A Prototype Adaptive Optics Real-Time Control Architecture for Extremely Large Telescopes using Many-Core CPUs

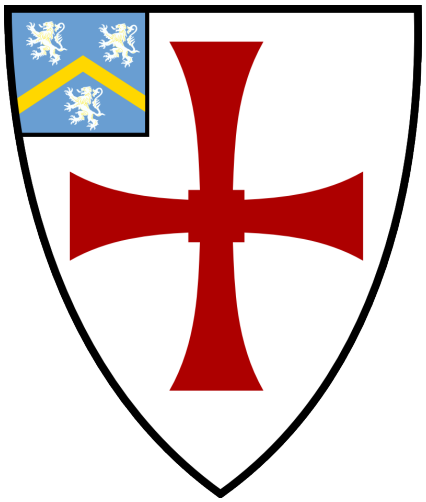
DAVID RICHARD JENKINS

A Prototype Adaptive Optics Real-Time Control Architecture for Extremely Large Telescopes using Many-Core CPUs

David Richard Jenkins

A thesis presented for the degree of

Doctor of Philosophy



Centre for Advanced Instrumentation

Durham University

United Kingdom

August 2019

A Prototype Adaptive Optics Real-Time Control Architecture for Extremely Large Telescopes using Many-Core CPUs

David Richard Jenkins

Submitted for the degree of Doctor of Philosophy

August 2019

Abstract

A proposed solution to the increased computational demands of Extremely Large Telescope (ELT) scale adaptive optics (AO) real-time control (RTC) using many-core CPU technologies is presented. Due to the nearly 4x increase in primary aperture diameter the next generation of 30-40m class ELTs will require much greater computational power than the current 10m class of telescopes. The computational demands of AO RTC scale to the fourth power of telescope diameter to maintain the spatial sampling required for adequate atmospheric correction. The Intel Xeon Phi is a standard socketed CPU processor which combines many (<64) low power cores with fast (>450GB/s) on-chip high bandwidth memory, properties which are perfectly suited to the highly parallelisable and memory bandwidth intensive workloads of ELT-scale AO RTC. Performance of CPU-based RTC software is analysed and compared for the single conjugate, multi conjugate and laser tomographic types of AO operating on the Xeon Phi and other many-core CPU solutions. This report concludes with an investigation into the potential performance of the CPU-based AO RTC software for the proposed instruments of the next generation Extremely Large Telescope (ELT) and the Thirty Meter Telescope (TMT) and also for some high order AO systems at current observatories.

Supervised by Alastair Basden and Richard Myers

Declaration

The work in this thesis is based on research carried out at the Centre for Advanced Instrumentation, Department of Physics, University of Durham, England. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is the sole work of the author unless referenced to the contrary in the text.

Some of the work presented in this thesis has been published in journals and conference proceedings - the relevant publications are listed below.

Relevant Publications

First author Publications:

ELT-scale adaptive optics real-time control with the Intel Xeon Phi Many Integrated Core Architecture, MNRAS, 478(3), August 2018, 3149–3158, Jenkins et al., 2018b

A many-core CPU prototype of an MCAO and LTAO RTC for ELT-scale instruments, MNRAS, 485(4), June 2019, 5142–5152, Jenkins et al., 2019

Conference Proceedings:

An ELT scale MCAO real-time control prototype using Xeon Phi technologies, SPIE, 10703, Jenkins et al., 2018c

Multi-node homogeneous Xeon Phi architecture for ELT scale Adaptive Optics RTC (Conference Presentation), SPIE, 10707, Jenkins et al., 2018a

ELT-scale real-time control on Intel Xeon Phi and many core CPUs, AO4ELT5, Jenkins et al., 2017

Acknowledgements

I would like to give special thanks to Alastair Basden and Richard Myers for their support as supervisors during my PhD. This PhD would not be possible without their input and teaching and I will always be grateful to them for helping me to begin my career as a researcher.

Both Gary McCallum and Claire Whitehill have been fantastic in helping me with network and computer issues and for organising travel and making sure I remember my seminar dates.

I would also like to thank the Durham Green Flash team for their help and motivation including James Osborn, Eddie Younger, Matthew Townson, Lazar Staykov, Deli Geng, Sofia Dimoudi, Andrew Reeves and Nigel Dipper.

Very special thanks to Mizuki, Saavi, Abi, Amrit, Jay, Nico, Ollie, Huizhe, Mark, Penny, Dougie, Alex, Daniel, Wenfeng, Zhentao, Wei, Stuie, Ruari, Vlad, Xuewen, Sam, Jonatan, Jack, Duncan, Laura, Andy, and all the others who helped me enjoy my time in Durham and made it an unforgettable time of my life.

Thanks to Saavi, Matt, Amrit, Abi and Jamie for organising the student journal club and representing CfAI student on the SSCC. Thanks to Tim Morris, Simon Morris, Ray Sharples, Richard Wilson, Tim Butterly, Larry Fitzpatrick, and Kieron O'Brien for their support as mentors and the encouragement they gave me. Thanks to Chris Saunter and Nigel Dipper for their excellent Masters project supervising, without which I would not have been doing the PhD in the first place. Thanks again to Richard Myers for the role of academic advisor during my undergraduate studies and recommending my for the PhD project. Thanks to David Barr et al. for beginning the work on many-core CPU AO RTC at Durham and laying the groundwork for my PhD. Thanks to Matt and Andrew for the thesis template, which took a lot of time out of the formatting. Thanks to Ollie Farley for his help with understanding turbulence profiling and helping me to make the Cn2 plot.

Finally thanks to my parents for making this all possible and supporting me through out my entire life.

Real-time Adaptive Optics work by the Durham group was supported by the European Union Horizon 2020 funded GreenFlash project, ID 671662, under FETHPC-1-2014, and is still supported by the UK Science and Technology Facilities Council (STFC) consolidated grant ST/P000541/1, and a STFC PhD studentship, award reference 1628730.

Contents

Nomenclature

AcO active optics

AO adaptive optics

AOF Adaptive Optics Facility

bpp bits per pixel

CCD charge-coupled device

CoG centre of gravity

CPU central processing unit

DARC The Durham Adaptive Optics Real Time Controller

DASP The Durham Adaptive Optics Simulation Platform

DM deformable mirror

DoF degrees of freedom

DSM deformable secondary mirror

DSP digital signal processor

EE encircled energy

ELT extremely large telescope

ESO ELT The Extremely Large Telescope

ESO European Southern Observatory

EU European Union

ExAO extreme AO

FPGA field programmable gate array

FWHM full width half maximum

GLAO ground layer AO

GMT Giant Magellan Telescope

GP-GPU general purpose graphics processing unit

GPU graphics processing unit

HDL hardware description language

HPC high performance computing

IP intellectual property

IRQ interrupt request

KNC Knights Corner

KNL Knights Landing

LBT Large Binocular Telescope

LGS laser guide star

LQG linear quadratic gaussian

LTAO laser tomographic AO

MCAO multi conjugate AO

MOAO multi object AO

MV minimum variance

MVM matrix-vector multiply

NGS natural guide star

NIC network interface controller

OS operating system

PMX poke matrix

POL pseudo open-loop

POLC pseudo open-loop control

PSF point spread function

Pyr-WFS pyramid WFS

RTC real-time controller

SCAO single conjugate AO

SH-WFS Shack-Hartmann WFS

SIMD single instruction multiple data

SPARTA Standard Platform for Adaptive optics Real Time Applications

SR Strehl ratio

STFC Science and Technology Facilities Council

TMT Thirty Meter Telescope

VLT ESO Very Large Telescope

WFS wavefront sensor

WHT William Herschel Telescope

Chapter 1

Introduction

1.1 Motivation

Ground based astronomical telescopes excel in a few areas that are extremely difficult to reproduce in space: they can be much more flexible and upgradeable, and most importantly they can employ much larger primary apertures. To be able to probe the furthest reaches of space, telescopes need to collect as much light as possible and with the next generation of extremely large telescopes (ELTs) the collecting area for optical and infra-red telescopes will be an order of magnitude greater than that available today. A larger aperture also gives an improved diffraction limit so that a greater range of spatial frequencies can be detected and thus smaller objects can be better resolved. However the biggest disadvantage to basing optical and infra-red telescopes on the ground is the need to see through Earth's turbulent atmosphere. The atmosphere is constantly changing, with turbulent air currents and the mixing of air with different temperatures, causing the incoming light from extraterrestrial objects to appear blurry and to lose definition.

All optical ground based telescopes with a primary mirror greater than a certain diameter are affected by atmospheric turbulence as it reduces the telescope diffraction limit to that of a much smaller telescope, the size of which is dependent on the strength of the atmospheric turbulence. This is known as the seeing limit, which is

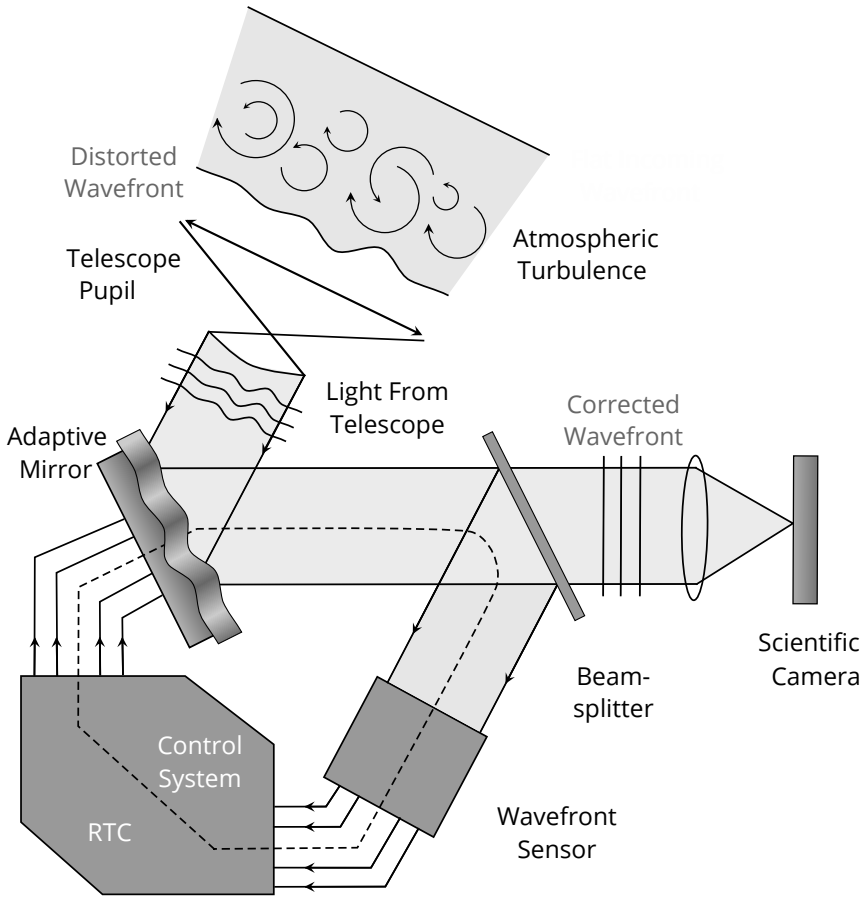


Figure 1.1: Standard closed-loop Adaptive Optics. The turbulent wavefront phase induced by the atmosphere is corrected by a deformable mirror (DM) using residual phase measurements provided by the wavefront sensor (WFS).

dependent on the wavelength of observation. Telescopes with larger primary apertures are affected to an even greater extent as the best seeing conditions for a certain observing site remain fairly constant and so the reduction in effective resolution is even more pronounced. For the three next generation ELT-scale telescopes, the 25m Giant Magellan Telescope (GMT, Johns et al., 2004), the 30m Thirty Meter Telescope (TMT, Stepp and Strom, 2004) and the 39m The Extremely Large Telescope (The ELT, Spyromilio et al., 2008)[1], mitigating the effect of the atmosphere is even more important and crucial to their scientific success.

[1]The Extremely Large Telescope (ESO ELT) will henceforth be referred to as the "ESO ELT" in this thesis, to distinguish from the more general term for extremely large telescopes (ELTs).

1.1.1 Adaptive Optics

Adaptive optics (AO, Babcock, 1953) is a widely-used technique that helps to negate the perturbing effects of the atmosphere and allows ground based telescopes to achieve imaging fidelity much closer to the diffraction limit than otherwise. AO has been widely used in Astronomical Instrumentation since its first civilian demonstration in 1989 with the deployment of the COME-ON system at the Haute-Provence Observatory in France (Merkle et al., 1990). The first UK AO implementation was the MARTINI partial-AO system (Doel et al., 1992; Myers et al., 1994; Sharples et al., 1994) developed by Durham University and deployed at the William Herschel Telescope (WHT) on the island of La Palma and saw its first light in 1992. Since then more complex AO systems have been developed, which have greatly increased the degree of atmospheric correction and the size of the corrected field of view.

The functionality of current AO systems can be split into 3 main parts: indirectly detecting the incoming wavefront, reconstructing the wavefront, and then applying corrections to mitigate the effects of the atmosphere. The detection and correction of the wavefront are usually performed by optical methods, using a wavefront sensor (WFS) for the detection, and a deformable mirror (DM) for correction. Wavefront reconstruction is a computational method. The basic idea behind the reconstruction is to attempt to "flatten" the wavefront from a point-source natural guide star (NGS) which picks up aberrations as it travels through the atmosphere. Therefore any deviations of the measured phase of the wavefront from a flat wavefront are considered perturbations and can be corrected by the DM.

The two most common types of WFS used on-sky are the Shack-Hartmann WFS (SH-WFS) and the pyramid WFS (Pyr-WFS) which work in fundamentally different ways. The SH-WFS measures the local slope of the wavefront at a discrete number of points across the pupil plane by the use of a lenslet array to create a regular grid of focussed spots made from focusing the light from smaller subapertures.

The Pyr-WFS focusses the light on the point of an optical pyramid, or similarly a double knife edge, which then forms 4 images of the pupil. These 4 images are processed to find the ratios of light at each pixel location which then gives the local wavefront slope at those points. The operation of both types of WFS is given in more detail in Chapter 2.

The DMs used in adaptive optics are commonly made up from a continuous facesheet mirror behind which an array of piezoelectric actuators that locally deform the facesheet. The "stroke", or working range, of these actuators is on the μm scale and so large perturbations in the wavefront can be difficult to correct. A method to improve the working range of a high order (large number of actuators) DM is to use it in conjunction with a low order DM, which can generally achieve high stroke values, in a "woofer-tweeter" configuration. This means that the low order DM corrects the stronger low-order wavefront aberrations first, such as wavefront tip and tilt, before the higher order DM corrects the higher-order aberrations.

1.1.1.1 Characterising the atmosphere

The relation between the strength of the atmospheric turbulence (and therefore the amount of wavefront perturbation) to the size of the PSF is known as the *seeing* limit. A description of the statistics of atmospheric turbulence was developed by Kolmogorov (Kolmogorov, 1991) by studying the mean-square velocity difference between two points in space . A number of assumptions about the atmosphere must be made in order to derive a velocity structure function which depends on the displacement of the two points. These are that the atmosphere is locally homogenous (the velocity depends on the displacement vector between the two points), the atmosphere is locally isotropic (the velocity depends on the magnitude of the displacement), and lastly that the turbulence is incompressible (divergence of velocity is zero, $\boldsymbol{\nabla} \cdot \mathbf{v} = 0$). A further assumption is that the temperature follows velocity as a passive additive (Tatarskii, 1971) which in turn leads to a 3D refractive index structure function.

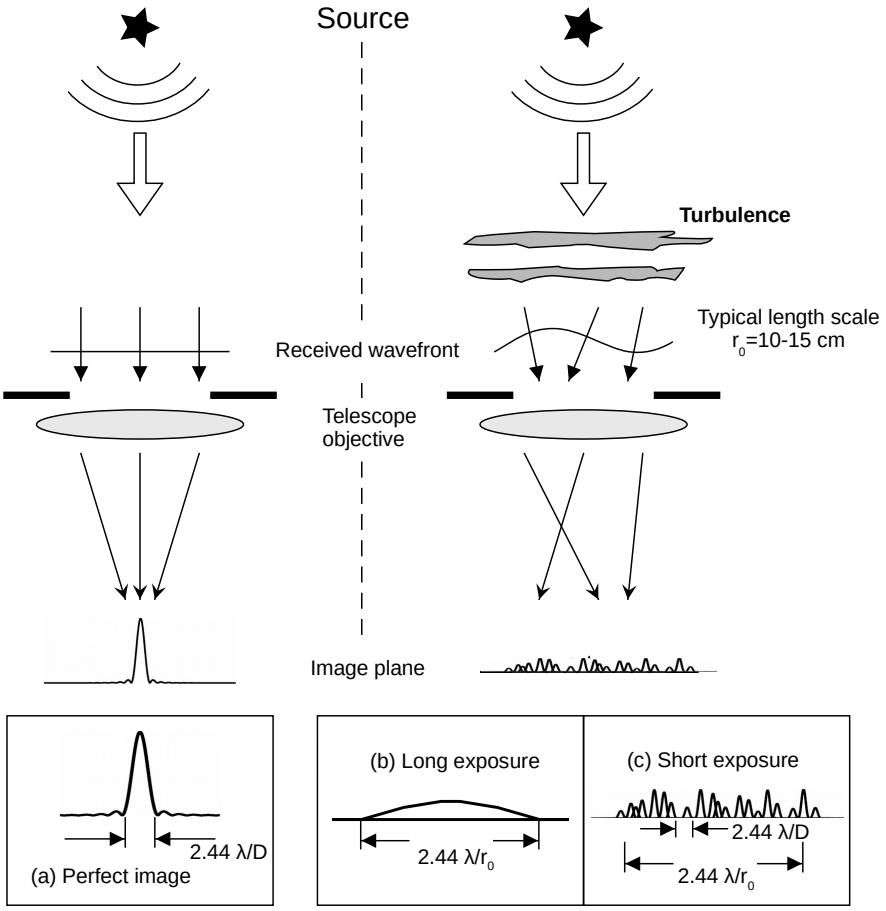


Figure 1.2: A diagram showing how the atmosphere affects the imaging resolution of astronomical observations. The turbulent layers perturb the incoming wavefront causing the short exposure images to be corrupted, which average out the perfect image of width $2.44\lambda/D$ to a larger PSF of width $2.44\lambda/r_0$.

The effect that turbulence has on an incoming wavefront is shown in Figure 1.2, showing that the width of the diffraction limited PSF is given by $2.44\lambda/D$ whilst the width of the seeing limited PSF is given by $2.44\lambda/r_0$.

The strength of the turbulence in the atmosphere which causes the wavefront perturbations can be defined by the Fried parameter (Fried, 1966), r_0, which has units of length. At good observation sites this is usually of order ~10-15 cm at a wavelength of around 500 nm at night time. The Fried parameter is related to the size of the PSF in the seeing limit in a very similar way to the relationship of the telescope diameter to the size of the PSF in the diffraction limit. Therefore the

effective diffraction limit for all telescopes with aperture diameters greater than r_0 will be constant for given seeing conditions and roughly equal to the diffraction limit of a telescope with diameter r_0. This means that without any AO correction the resolving power of larger and larger ground-based telescopes remains effectively constant at the particular seeing limit.

The atmosphere can generally be described as being made up of layers of turbulence at different altitudes and with varying strengths and characteristics (Tatarskii, 1961). The refractive index structure constant, C_n^2, defines the strength of turbulence as a function of altitude and can be used to find the strongest layers of turbulence. Even though it is called a constant, it is a constantly varying value depending on short and long timescales, geographic location, and currently there is no theoretical model accurate enough for all different situations. An equation to describe C_n^2 was derived by Hufnagel (Wolfe and Zissis, 1985) based on experimental observations,

$$C_n^2 = \{[(2.2 \times 10^{-53})h^{10}(W/27)^2]e^{-h/1000} + 10^{-16}e^{-h/1500}\} \exp[r(h,t)] \qquad (1.1)$$

where h is the altitude above sea level in meters, W is the wind correlating factor (Tyson, 2010) defined as,

$$W = \left[\left(\frac{1}{15\text{km}}\right)\int_{5\text{km}}^{20\text{km}} v^2(h)\mathrm{d}h\right] \qquad (1.2)$$

and $r(h,t)$ is a zero-mean homogenous Gaussian random variable as a function of altitude (h) and time (t). C_n^2 has units of $m^{-2/3}$. The W term is an important consideration in Equation 1.1, and it requires a model of the dependence of wind speed on altitude which governs its time sensitive nature.

As can be seen in Figure 1.3, the Paranal data show that a large portion of the strength of the atmospheric turbulence resides very close to the ground in what is known as the boundary layer. The boundary layer is significant as it is directly

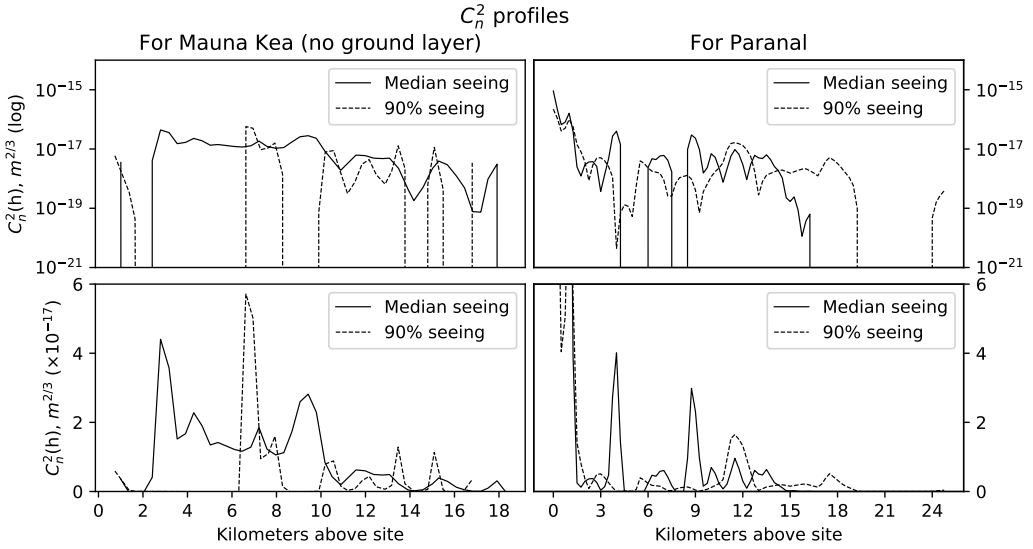


Figure 1.3: Mauna Kea (Ellerbroek and Tyler, 1998) and Paranal (Osborn et al., 2018) C_n^2 values as a function of height above the site. Median and 90th percentile seeing conditions correspond to r_0 values of 0.24 and 0.42 m for the Gemini North site on Mauna Kea (Racine et al., 1991) and 0.16 and 0.23 m for the VLT at Paranal (Osborn et al., 2018) at a wavelength of 0.5 µm.

influenced by the surface of the Earth, causing greater variations in temperature and complex mixing of air currents. An addition to Equation 1.1 which takes this into account was devised by Ulrich and results in the Hufnagel-Valley boundary (HVB) model (Ulrich, 1988),

$$C_n^2 = 5.94 \times 10^{-23} z^{10} e^{-z} (W/27)^2 + 2.7 \times 10^{-16} e^{-2z/3} + A e^{-10h} \tag{1.3}$$

where z is the altitude above sea level, h is the height above ground level, W is adjustable based on upper atmosphere wind conditions and A is a scaling constant.

The r_0 value for certain atmospheric conditions for a plane wave can be calculated by integrating the C_n^2 distribution along a line of sight (Fried, 1965),

$$r_0 = \left[0.423 k^2 \sec(\beta) \right] \int_0^L C_n^2(z) \mathrm{d}z \Bigg]^{-3/5} \tag{1.4}$$

where L is the path length ($L = \infty$ in astronomical applications), β is the zenith angle and the C_n^2 is able to vary with altitude z. The r_0 parameter is often known

as the "seeing cell size", with a "seeing cell" being a portion of atmosphere that acts as a lens and focuses the light due to the differences in refractive index in the turbulent air. The r_0 value is the size of the "seeing cell" in a 2-dimensional projection along the line of sight through a 3-dimensional cylinder of atmosphere. For astronomical observations of a given wavelength, it can be useful to know the typical night time median r_0 which can be approximated as (Fried and Mevers, 1974),

$$[r_0]_{median} = 0.114 \left(\frac{\lambda}{5.5 \times 10^{-7}} \right)^{6/5} \sec(\beta)^{-3/5} \tag{1.5}$$

by using Equation 1.4 because starlight arrives at the Earth's atmosphere as a plane wave. Here λ is the wavelength of the incident light and this is only suitable for a carefully selected site (Fried and Mevers, 1974). The wavelength dependence, $r_0 \propto \lambda^{6/5}$, means that at longer wavelengths the seeing is less severe and is therefore much easier to correct. Most current AO systems in use correct in the infrared, whilst visible light AO correction is much more difficult to achieve. As the diffraction limit is also wavelength dependent there is an even greater disparity between seeing limited and diffraction limited observations at shorter wavelengths, giving the potential for a much greater reduction of the effects of the atmosphere for visible light AO systems.

The median r_0 for a given observing site is an important consideration when first designing an AO system. However during operation of the AO instrument another important metric is the coherence time, τ_0, which is the maximum time delay between measuring the atmosphere and applying the correction that results in a mean square phase error of less than one radian. For the special case of a single turbulent layer with constant wind velocity, τ_0 can be defined by (Hardy, 1998)

$$\tau_0 = (6.88)^{-3/5} \frac{r_0}{v_w} = 0.314 \frac{r_0}{v_w} = \frac{0.134}{f_G} \tag{1.6}$$

where $f_G = 0.427(v_w/r_0)$ is the Greenwood frequency and is defined for the entire C_n^2 distribution with varying wind speeds as,

$$f_G = 2.31\lambda^{-6/5}\left[sec\beta \int_0^L C_n^2(z) v_w^{5/3}(z) dz\right]^{5/3} \tag{1.7}$$

For a typical observing site the Greenwood frequency would be measured to be of order $20-40\ Hz$ (Fried, 1990; Tyson, 2010) which gives τ_0 values of $3-7\ ms$. This determines the update rate of the AO system (the time between successive wavefront measurements), and typically the closed loop bandwidth of the AO system is taken to be roughly 10 times f_G (Greenwood and Fried, 1976).

Another important consideration when designing an AO system or choosing what type of AO to employ is the non-isoplanatic nature of the turbulence. The layers of the atmosphere can be considered as planes having various phase changes over their surface and orientated such that at zenith ($\beta = 0$) each plane is normal to the direction of propagation of a beam of light from an astronomical source (see Figure 1.4). While the atmospheric turbulence can be described statistically, and the statistics are same for different parts of the atmosphere, the light will be affected differently depending on its path through the atmosphere. Two different beams of light will pick up different phase aberrations as they travel along different paths through the atmosphere and a large beam will have varying aberrations across its diameter. This means that propagation through the atmosphere is described as anisoplanatic.

There are 5 main types of anisoplanatism that occur in different situations:

1. *Displacement* - displaced parallel beams
2. *Angular* - beams propagating at different angles
3. *Focal* - beams whose sources are at different distances from the receiver
4. *Temporal* - beams with a time delay between propagation

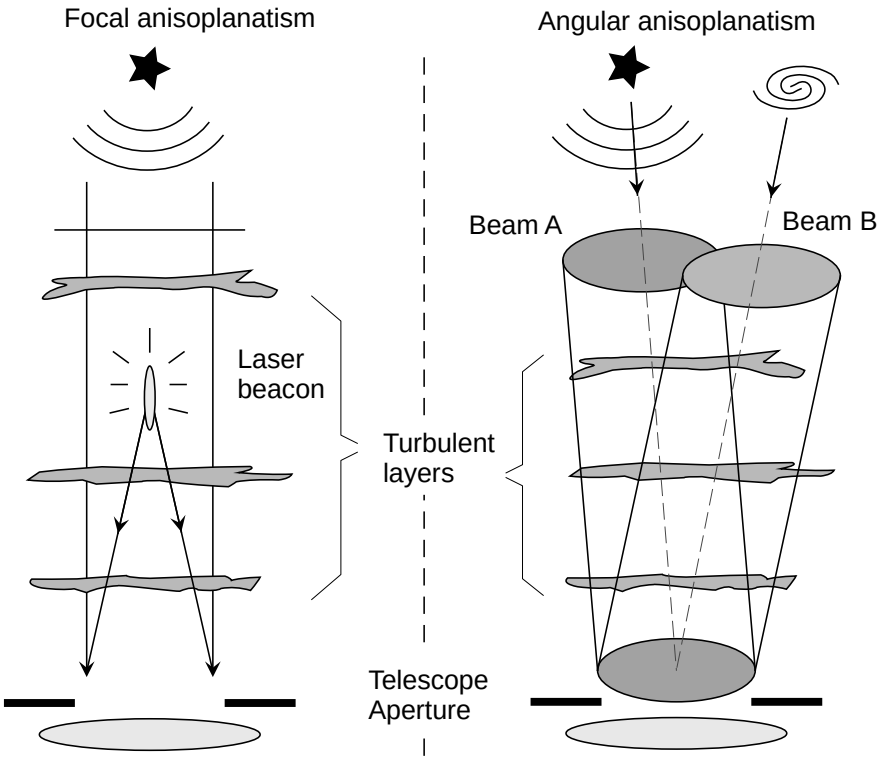


Figure 1.4: The two types of anisoplanatism that have the most effect on AO performance. Focal anisoplanatism affects the performance of LGS AO systems as it causes the "cone effect" whereby the laser doesn't sample the full portion of atmosphere that an NGS would. Angular anisoplanatism limits the sky-coverage of AO as the bright guide stars need to be close to the science object of interest.

5. *Chromatic* - beams with different wavelengths

For astronomical adaptive optics the most important types of anisoplanatism are angular anisoplanatism, focal anisoplanatism and temporal anisoplanatism. Angular anisoplanatism defines the maximum angle between an AO guide star and the scientific object of interest before the correction applied is no longer valid for that different line of sight. There needs to be a bright guide star available for AO within this angular distance from the object of interest and this therefore defines the allowable sky coverage of a natural guide star (NGS) AO system based on the distribution of suitable stars in the sky.

To improve the sky coverage a laser guide star (LGS) can be used to provide a bright

reference closer to the object of interest. However due to the finite altitude that a LGS spot appears at in the sky ($\approx 90km$ for a sodium laser) focal anisoplanatism reduces the available turbulence information that can be recovered using an LGS. The light from an astronomical source propagates through a cylindrical section of the atmosphere as the source is assumed to be at infinity. However as the LGS beacon is projected at a finite height, the light propagates through a conical section of atmosphere as seen in Figure 1.4. This means that the wavefront detected from an LGS beacon doesn't include all of the information that's required to correct for the full cylindrical section of atmosphere and it also produces a distorting projection of upper turbulent layers below the LGS altitude. Focal anisoplanatism is therefore known as the "cone effect" and is shown in Figure 1.4.

As mentioned above, the temporal anisoplanatism dictates the update rate of the AO system such that the corrections are applied within the necessary time window before the turbulence has evolved sufficiently to invalidate the wavefront measurements.

1.1.1.2 Representing the Aberrated Wavefront

The phase aberrations induced by atmospheric turbulence can be represented by a number of mathematical constructs. These define a 2-D map of the phase on a plane normal to the line of sight of the beam. The most common are either a power series representation in polar (ρ, θ) coordinates or the polynomials introduced by Zernike (1934), which are much better suited for atmospheric phase representation. The Zernike series of polynomials differs from the power series by being an orthonormal set over a circle, which can be useful when considering circular apertures in telescope design. The Zernike polynomials are composed of sums of power series terms along with the appropriate normalising factors and are given in more detail by Born and Wolf (1997). The Zernike polynomials are generally defined using two indices, (n, m), however an analysis of their suitability for describing atmospheric turbulence was conducted by Noll (1976) and a method for sequentially

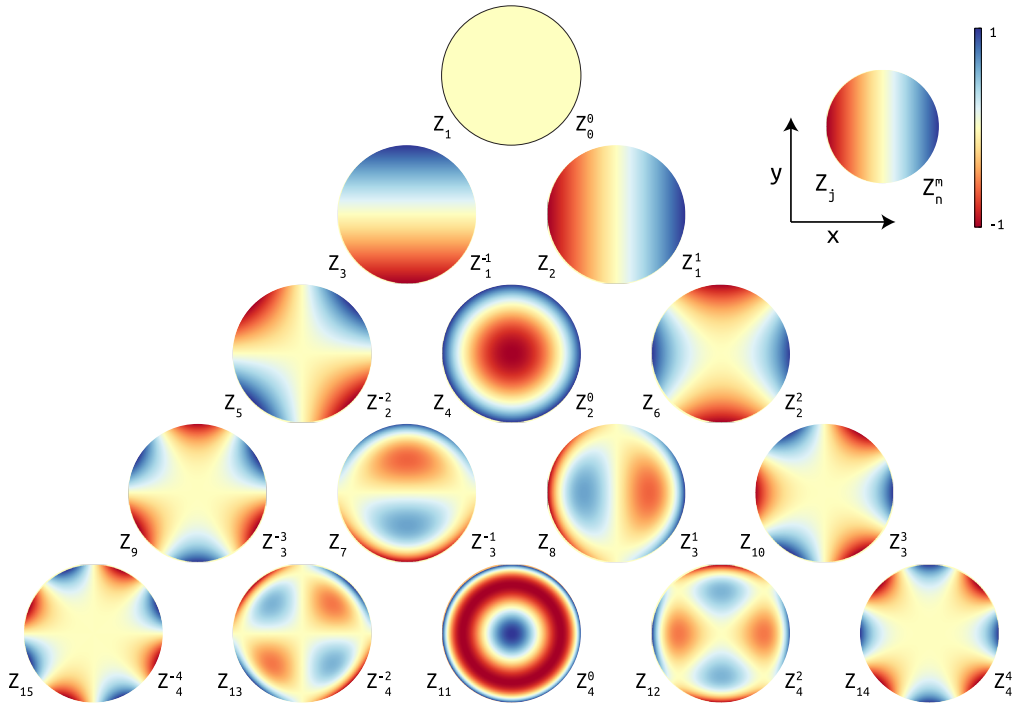


Figure 1.5: The first 15 Zernike modes indexed in both the standard n and m regime and with Noll j indexing (Jenkins, 2019).

indexing the Zernikes was devised that maps the two indices n and m to a single index j given by

$$j = \frac{n(n+1)}{2} + |m| + \begin{cases} 0, & m > 0 \wedge n \equiv \{0,1\} \pmod 4; \\ 0, & m < 0 \wedge n \equiv \{2,3\} \pmod 4; \\ 1, & m \geq 0 \wedge n \equiv \{2,3\} \pmod 4; \\ 1, & m \leq 0 \wedge n \equiv \{0,1\} \pmod 4. \end{cases} \tag{1.8}$$

The general Zernike series is represented by the following expression

$$\begin{aligned} \mathbf{\Phi}(\rho,\theta) = A_{00} + \frac{1}{\sqrt{2}} \sum_{n=2}^{\infty} A_{n0} R_n^0 \left(\frac{\rho}{R'}\right) \\ + \sum_{n=1}^{\infty} \sum_{m=1}^{n} [A_{nm} \cos m\theta + B_{nm} \sin m\theta] R_n^m \left(\frac{\rho}{R'}\right) \end{aligned} \tag{1.9}$$

where $n - m =$ even and R' is the radius of the circle over which the polynomials are defined. The A_{nm} and B_{nm} coefficients determine the strength of each Zernike

term and the series contains all aberration terms including piston (given by A_{00}) and tilt. The shape of the first 15 Zernike modes are shown in Figure 1.5 given by both the standard n and m regime and with Noll j indexing.

There are a number of properties of the Zernike series that make it very beneficial for use in adaptive optics (Tyson, 2010). One of the main benefits is that it allows a simple way to calculate the rms wavefront error directly from summing the coefficients of all non-piston terms of the wavefront,

$$(\Delta\mathbf{\Phi})^2 = \sum_{n=1}^{\inf} \sum_{m=0}^{n} \frac{A_{nm}^2 + B_{nm}^2}{2(n+1)} \tag{1.10}$$

where A_{nm} and B_{nm} are the coefficients of each Zernike polynomial Z_n^m.

1.1.1.3 Performance Estimation

To determine the performance of an AO system it is necessary to analyse all independent sources of wavefront error (residual uncorrected wavefront) and the overall performance can be estimated by summing in quadrature the individual errors. When all sources of error are uncorrelated the total residual wavefront phase error squared is simply given as the sum of their variances, (Hardy, 1998)

$$\sigma_{phase}^2 = \sum \sigma_i^2 \tag{1.11}$$

However in reality there are correlations between some of sources of error and so Equation 1.11 can lead to an overestimate of the total residual error which should be accounted for to achieve a more realistic prediction. Figure 1.6 shows a summary of the major sources of error along with their dependence on both external atmospheric parameters and on AO system parameters.

Once the total wavefront error variance has been estimated it is possible to calculate image quality metrics such as the Strehl ratio that will result from the system. The Strehl ratio is a common way to quantify the reduction in peak intensity when

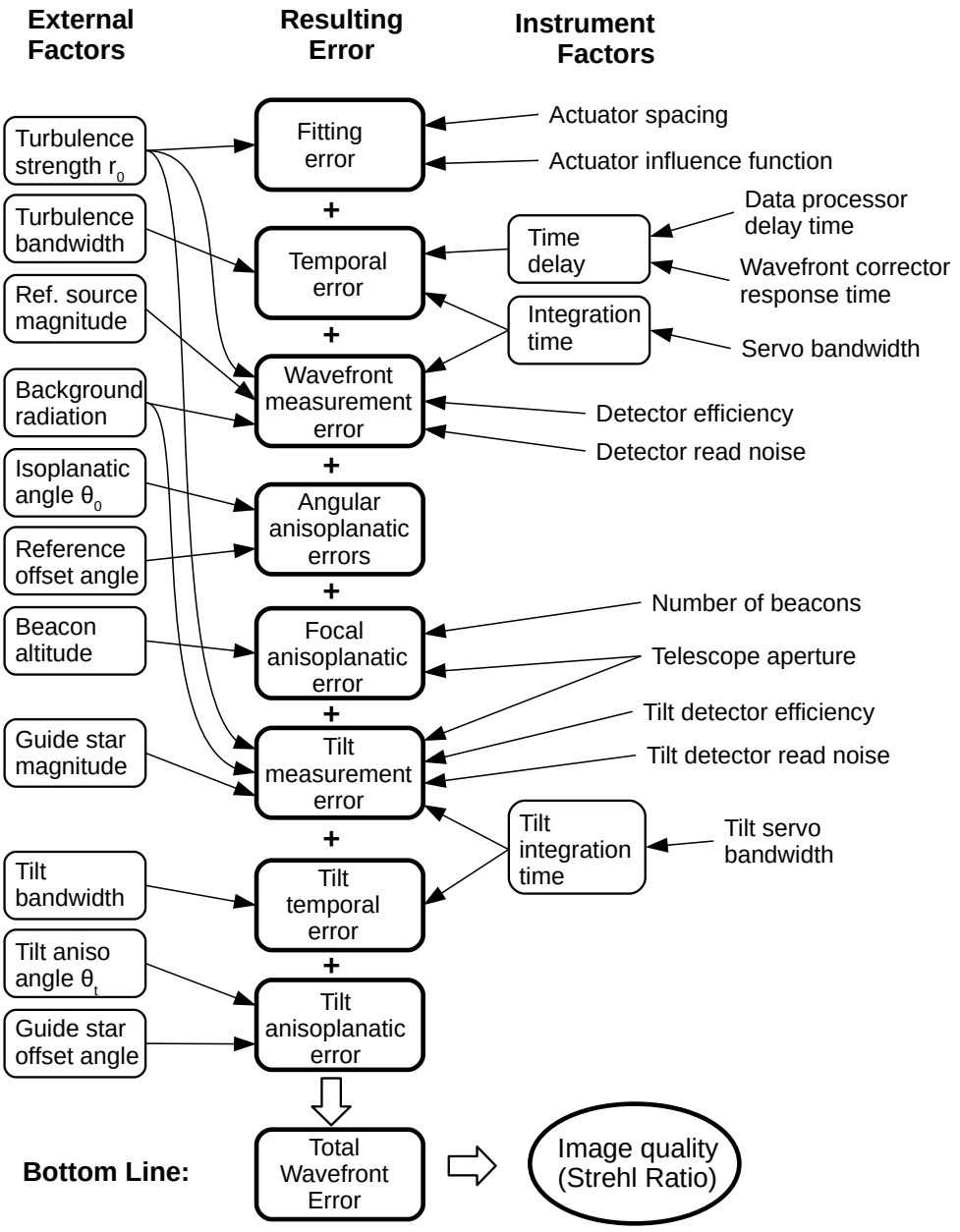


Figure 1.6: Main sources of wavefront error in adaptive optics, adapted from Hardy (1998).

aberrations are present in an optical system. It is defined as the ratio of the actual peak intensity to that of an ideal peak intensity obtained at the Airy image point when no aberrations are present. A common approximation of the Strehl ratio calculation was derived for small aberrations by Marechal (1947),

$$S \approx \exp\left\{-\sigma_p^2\right\} \tag{1.12}$$

where σ_p is the standard deviation of the wavefront phase. This is known as the "extended Marechal approximation". The Strehl ratio is now a relatively easy-to-calculate performance metric for an AO system as whole. However it does not completely represent the absolute performance of an imaging system in the following situations: when the contrast between the signal and the background is more important such as for exoplanet imaging; or when the important scientific/instrumental parameter is the coupling of light to a spectrographic slit or spatial element, the size of which can be significantly larger than the diffraction limit.

Other image quality metrics used in AO and astronomical observations include the full width half maximum (FWHM) and the encircled energy (EE). The FWHM is defined as the width of the PSF at half of its maximum amplitude. For a diffraction limited Airy disk, the FWHM is almost exactly the same as the radius of the first minimum and is therefore commonly used as a measure of the size of the PSF. The EE is defined the proportion of light within a given radius around the centroid of the PSF. A common use of EE in astronomical imaging is to determine the radius at which either 50 % or 80 % of the total energy is contained. For a diffraction limited Airy disk, the radius of the 50 % EE is almost exactly half the width of the FWHM and the 80 % EE is roughly the same as the position of the first minimum (first minimum has 83% EE).

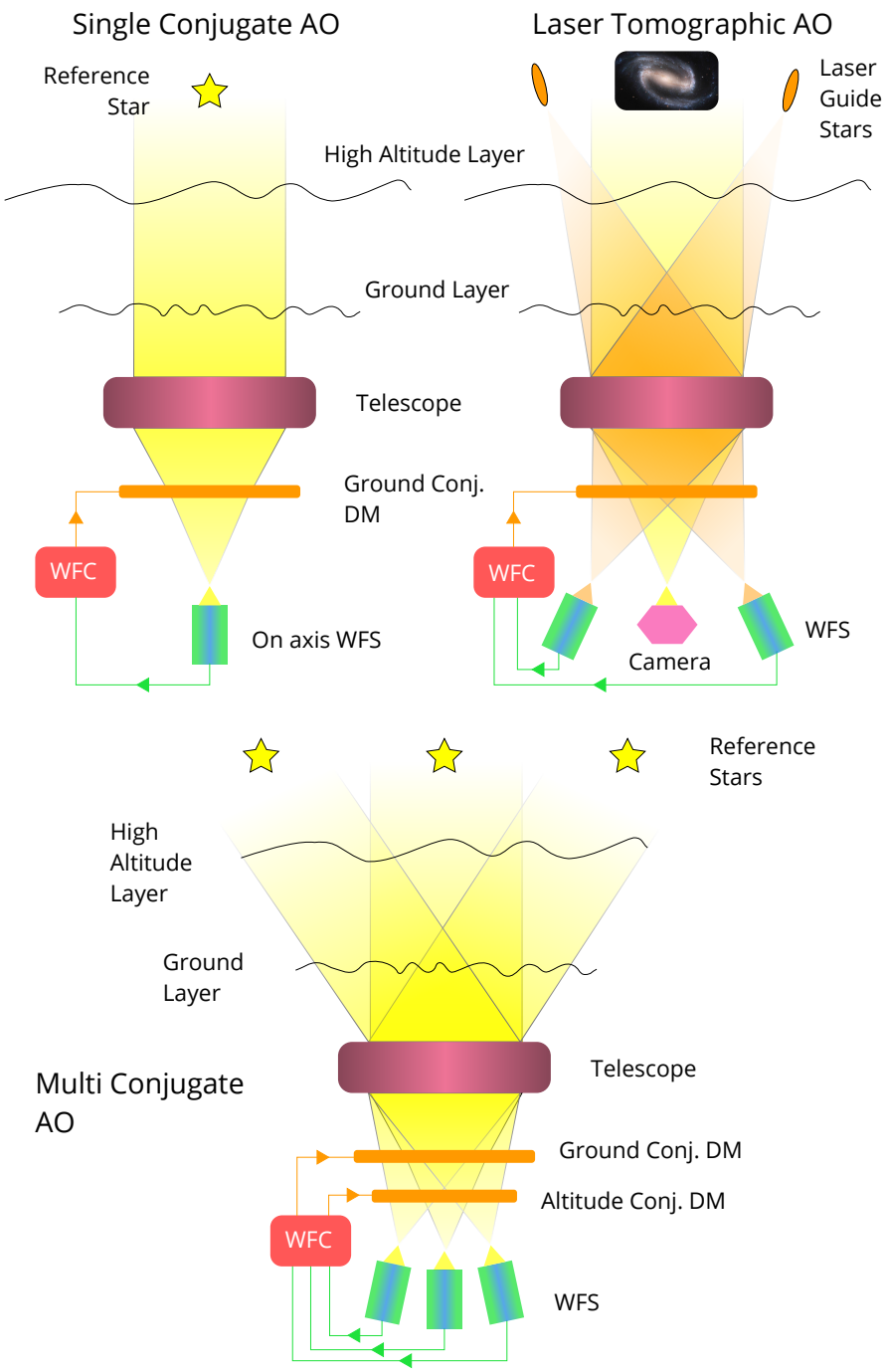


Figure 1.7: A visual comparison of SCAO, LTAO and MCAO. SCAO in general has one WFS and one DM. MCAO has multiple WFSs focussed on different guide stars and multiple DMs conjugated to different atmospheric layers. LTAO is similar to MCAO except it mainly uses LGS, it generally only has one DM and it corrects over a narrower FoV.

1.1.2 AO Classifications / Types of AO

There are different classifications of AO system types which can be chosen to suit the given parameters of the astronomical observations. These parameters include but are not limited to,

- the size of the corrected field of view required
- the location and therefore NGS availability of the observation on the sky
- the wavelength of observation
- the number of observation targets
- the performance metric to be optimised e.g SR, FWHM, contrast.

The types of AO available range from comparatively simple Single Conjugate AO (SCAO), which uses a single guide star for correction with a single DM, to more complex systems such as Ground Layer AO (GLAO, Rigaut, 2002), Multiple Conjugate AO (MCAO, Beckers, 1988; Dicke, 1975; Johnston and Welsh, 1994), Laser Tomographic AO (LTAO, Foy and Labeyrie, 1985; Fugate et al., 1991; Murphy et al., 1991; Tallon and Foy, 1990) and Multi Object AO (MOAO, Gendron et al., 2005; Rousset et al., 2010) which use multiple reference stars (NGS or LGS) and can include multiple correcting elements. A recent development in AO is a technique called Extreme AO (ExAO, Angel, 1994; Guyon, 2018; Nakajima, 1994; Stahl and Sandler, 1995) which aims to correct a single line of sight to achieve very high SR by running the AO system at higher updates rates and, in conjunction with a coronagraph, can achieve very high contrast levels for direct exoplanet imaging. Table 1.1 summarises the main characteristics of each of the AO types.

1.1.2.1 Single Conjugate AO

The most basic form of AO is single conjugate AO (SCAO), generally using a single WFS with a natural guide star to provide wavefront measurements to correct along

AO Type	Summary
SCAO	One main WFS (NGS or LGS), one main DM, high increase in SR for a narrow FOV, low sky coverage for NGS, increased sky coverage for LGS but reduced correction due to the cone effect
GLAO	Multiple WFSs (NGS or LGS), one main DM optically conjugate to the usually very strong ground layer of turbulence, low increase in SR for a very wide FOV
MCAO	Multiple WFSs (NGS or LGS), multiple DMs optically placed at different conjugate altitudes, medium to high increase in SR for a wide FOV
LTAO	Multiple WFS (NGS and LGS), one main DM, high increase in SR for a medium FOV, increased sky coverage, multiple LGS to reduce cone effect compared to LGS SCAO
MOAO	Multiple WFS (NGS or LGS), one DM per science target, high increase in SR for a narrow FOV around each science target
ExAO	(usually) One WFS (NGS or LGS), one high order main DM, extremely high increase in SR for a very narrow FOV, can provide very high contrast imaging

Table 1.1: A summary of each of the main types of AO available.

a single line of sight with a single DM. This is shown in Figure 1.7. An NGS is simply a bright star that is close enough to the science object of interest such that the light travels through as much of the same atmosphere as possible to ensure that the reconstruction is valid for the direction of the science object. Generally the isoplanatic patch size is small even for good seeing conditions ($\approx$10 mas at H-band) leading to a small corrected field of view for SCAO and due to the sparse distribution of stars in the sky bright enough for AO, NGS SCAO is greatly limited by sky-coverage. SCAO can also be used with a single LGS to increase sky-coverage, however due to the cone effect described in Section 1.1.1.1 the amount of correction is generally less than that available with an NGS. For LGS there is also the tilt determination problem (Rigaut and Gendron, 1992): because the LGS first needs to propagate upwards through the turbulent atmosphere, it is therefore not possible to recover the tilt of the wavefront from a LGS alone and so a NGS is still required for detecting tilt.

Due to the correction being applied solely through a single DM, the DM is placed in an optical plane conjugate to the ground layer and so all the turbulence detected by the WFS is collapsed to a single layer. The reconstruction algorithm is normally quite simple for SCAO as the NGS is considered to be at infinity; it should therefore have a flat wavefront as it first arrives at the Earth's atmosphere. Reconstructing the wavefront is then a case of finding the correction that should be applied to re-flatten the turbulent NGS wavefront and in the process also correcting the wavefront of the science object of interest. The mapping from WFS to DM can therefore be constructed by first applying a flat wavefront to the DM and then measuring the resulting wavefront when each actuator of the DM is actuated in turn. This is usually done off-sky with a flat reference source but can also be done on-sky (e.g. by using temporal modulation of the poke). This procedure creates a "poke matrix" or a mapping of DM commands to WFS measurements. Inverting this matrix then yields the mapping needed to go from wavefront measurements to actuator commands.

There have been and currently are many implementations of simple SCAO systems either operating on-sky or in laboratory environments for testing. Gemini Observatory has operated the ALTitude conjugate Adaptive optics for the InfraRed (ALTAIR, Herriot et al., 1998) system which has been available for observations since 2004 using NGS, with final commissioning of an LGS operation mode (Boccas et al., 2006) in 2007. The W.M. Keck Observatory operates a SCAO system on the Keck II telescope (Wizinowich et al., 1998), which has been operational since 1999, with an LGS upgrade (Wizinowich et al., 2006) in late 2004. The ESO Very Large Telescope (VLT) operates multiple different SCAO systems for its different instruments. The first AO system operational on the VLT was the Nasmyth Adaptive Optics System (NAOS, Rousset et al., 1998, 2003) which had its first light in 2001 and an LGS upgrade in 2004 (Kasper et al., 2004).

More recently there have been several very interesting implementations of SCAO that have improved upon first generation instruments. Robo-AO (Baranec et al.,

2011) achieved first light in 2012 at the 1.5m Palomar Observatory (Baranec et al., 2013; Riddle et al., 2014) before moving to the 2.1m telescope at Kitt Peak in 2015 (Jensen-Clem et al., 2017; Salama et al., 2016) and more recently to the 2.2m UH88 (Ashcraft and Baranec, 2018; Baranec et al., 2018) telescope on Mauna Kea. Robo-AO is the first fully autonomous LGS AO system and science instrument that has operated on-sky. It is capable of performing large scale surveys, monitoring long-term astrophysical dynamics and characterising newly discovered transients. It is able to do this all at the visible light diffraction limit of the 2m class telescopes it operates on.

Another visible light SCAO system is Magellan AO (MagAO, Close et al., 2010) currently in operation on the twin 6.5m Magellan telescopes in Chile, which saw first light in 2012 and first general science run in 2014 (Morzinski et al., 2014). As mentioned in Section 1.1.1.1, shorter wavelengths are affected more by atmospheric turbulence and so correcting in the visible spectrum is more difficult than the infrared and until recently has been rarely achieved. MagAO achieves this by the use of a high order deformable secondary mirror (DSM) and a modulating NGS pyramid WFS (Morzinski et al., 2014) running at up to 1000Hz. MagAO has been able to achieve up to 30% Strehl ratio (SR) in the visible (Close et al., 2014). MagAO is very similar to the twin First Light AO (FLAO, Quirós-Pacheco et al., 2010) systems on the Large Binocular Telescope (LBT) at Mt. Graham Arizona, FLAO#1 and FLAO#2, which achieved their first light in 2010 (Esposito et al., 2011) and 2011 (Esposito et al., 2012) respectively. The FLAO systems also use a high order DSM coupled with a high order pyramid WFS to deliver a >80% SR in the H-band (Esposito et al., 2011).

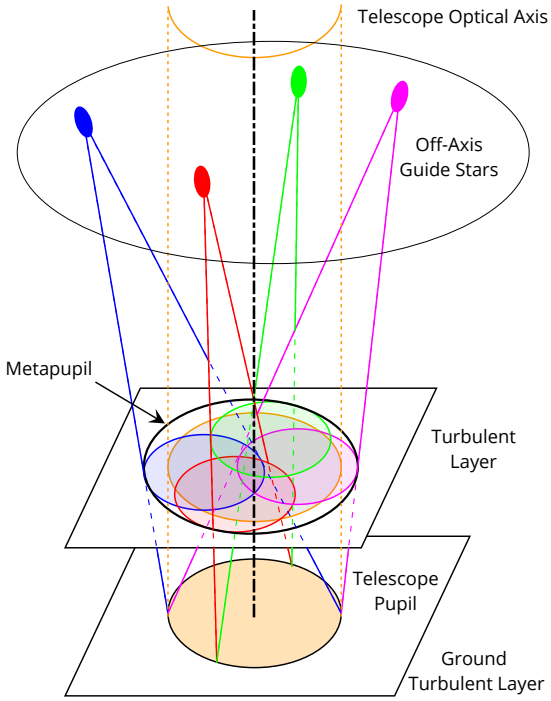
1.1.2.2 Laser Tomographic AO

Laser tomographic adaptive optics (LTAO) is one of the more simple extensions to SCAO to help improve performance. Its main aim is the reduce the focal anisoplanatism (cone effect) as described in Section 1.1.1.1. To achieve this, LTAO systems

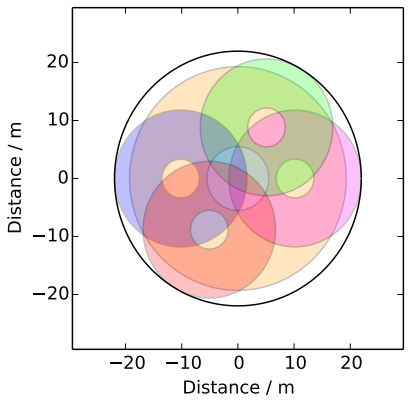
employ multiple laser guide stars such that the conical corrected lines of sight of each overlap to get a better measurement of the turbulence for the science object of interest. This is shown in Figure 1.7, which can be contrasted to the single LGS case demonstrating the cone effect shown in Figure 1.4. The corrected field of view of LTAO systems is generally kept quite small and similar to that of SCAO systems, however the use of multiple LGS gives much greater sky coverage and better correction than a single LGS can provide.

Due to the overlapping cross sections of the LGS beams, it is possible to measure the turbulence at different altitudes by considering the degree of correlation between the different LGS WFS measurements. This technique is know as tomography and, used with suitable reconstruction algorithms, yields greater correction due to the greater volume of turbulence that is corrected along the line of sight of the science target. Knowing the turbulence at different heights allows the corrected field of view to be directed towards the science object by considering the cross-section of the incident light through these layers. The geometry of the overlapping cross-sections can be seen in Figure 1.8. LTAO systems use a single DM to apply correction and so the turbulence measured along the required line of sight is collapsed to the altitude that the DM is conjugated to; usually the ground layer.

The VLT currently offers LTAO for the "Multi-Unit Spectroscopic Explorer" (MUSE McDermid et al., 2008) instrument using the GALACSI (Ströbele et al., 2012) AO system of the Adaptive Optics Facility (AOF Madec et al., 2018). The narrow field mode of the MUSE facility (MUSE instrument + GALACSI AO) uses 4 LGS to provide tomographic information for applying the corrections using the Adaptive Optics Facility (AOF) deformable secondary mirror (DSM Arsenault et al., 2006). MUSE achieved its first light in 2014, and achieved its first narrow field AO corrected light using LTAO in 2018.



a) 3-dimensional view of a four LGS LTAO setup



b) Simulated LGS overlap using DASP. The LGS beacons are set to a height of 90 km and the layer shown is at 35 km. The aperture is an ESO ELT-like 39 m primary with an 11 m central obscuration. The asterism has a radius of 60 mas with the four beacons positioned at 0, 60, 180 and 240 degrees.

Figure 1.8: Using tomography to measure the turbulence along the optical axis of the telescope by considering the overlapping cross section of the off-axis LGS to create a metapupil. (a) Shows a 3-dimensional schematic view of a 4 LGS LTAO setup. (b) Shows a the overlap of the LGS as simulated by DASP. Colours are for illustrative purposes only.

1.1.2.3 Multi Conjugate AO

Multi conjugate adaptive optics (MCAO) uses multiple DMs in the optical path conjugate to different altitudes in the atmosphere resulting in a wider corrected field of view compared with LTAO. It can operate solely with NGS or using a combination of NGS and LGS. Along with the multiple WFSs looking at different guide stars, it can use the tomographic information to correct for the strongest layers of turbulence. This is shown in Figure 1.7. The corrected field of view of MCAO is generally wider than that of LTAO as it takes into account multiple layers of turbulence allowing it to get a better correction along the different lines of sight, however the maximum amount of correction is generally lower than that achievable with LTAO.

There is currently only one MCAO system in operation on an 8-m class telescope. The Gemini Multi-Conjugate Adaptive Optics System (GeMS d'Orgeville et al., 2008) is located at the Gemini South Observatory where it achieved first light in 2011 (Rigaut et al., 2012). During commissioning it had already produced images with H band Strehl ratio in excess of 35% over fields of view of 85×85 arcsec, fulfilling the MCAO promise of wide field correction. The VLT demonstrated MCAO with its "Multi-Conjugate Adaptive Optics Demonstrator" (MAD, Marchetti et al., 2003) which was on-sky for 8.5 effective nights in 2007 (Marchetti et al., 2007). The result of MAD was a corrected field of view of almost 2 arcminutes with greater than 20% Strehl and peaks of up to 40% around the guide stars; a representative Strehl map is shown in Figure 1.9 (Marchetti et al., 2007) from MAD.

1.1.2.4 Other AO Types

There are 3 other major types of AO that won't be discussed in this thesis. The first is ground layer AO (GLAO Baranec et al., 2007) which is similar to LTAO except it uses a much wider asterism for either LGS or NGS to correct over a wider field of view. GLAO only considers the ground layer of turbulence such that the LGS can be spaced wider apart and so their cross sections only need to overlap up to a relatively low altitude. As shown in Section 1.1.1.1 the ground layer is a large source of turbulence: typically 0.5 to 0.67 of the total atmospheric turbulence is in the ground layer (Baranec et al., 2009) and there is a very high degree of overlap of the LGS and so the correction can be quite good over the large field of view. The wide field mode of GALACSI for the VLT MUSE instrument operates in GLAO mode, which had its first light in 2017. A representative Strehl map from the VLT comparing the sky coverage of SCAO, GLAO and MCAO is shown in Figure 1.9 Marchetti et al. (2007).

One of the most complicated types of AO is known as multi object AO (MOAO) which aims to correct along many different lines of sight at once by the use of multiple DMs, one for each line of sight. The CANARY (Myers et al., 2008) AO

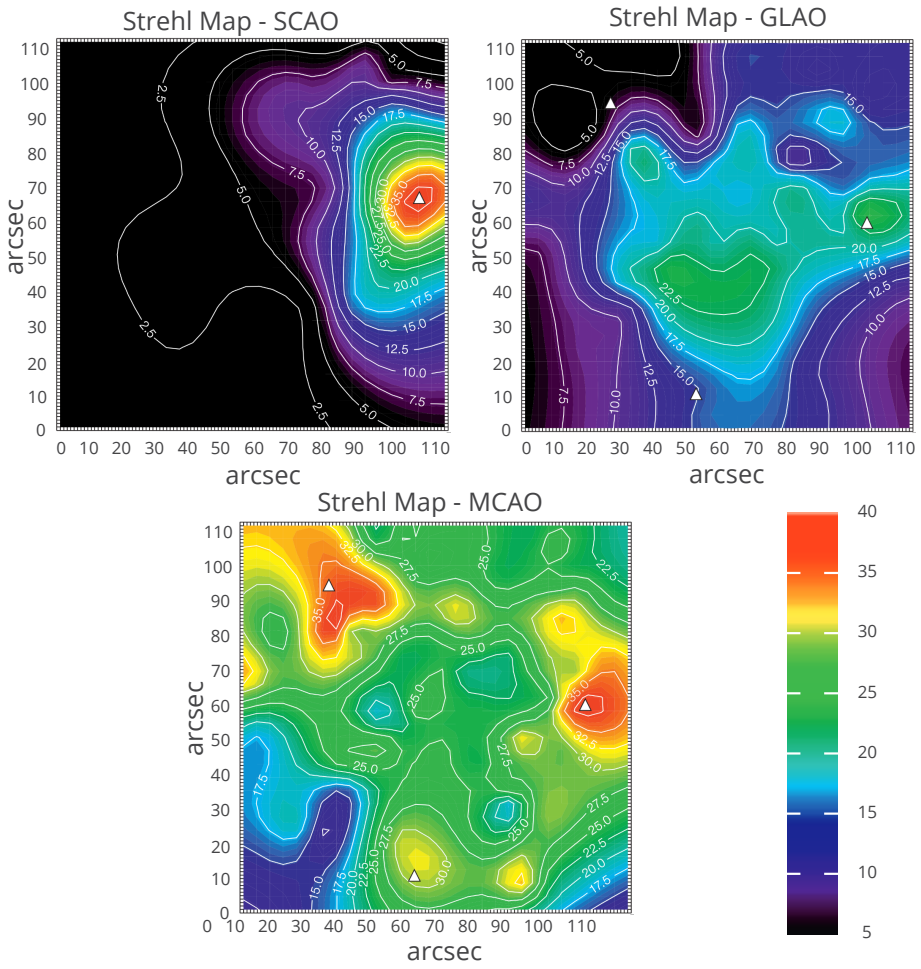


Figure 1.9: A comparison of representative Strehl maps from the VLT showing the Strehl (in % at 2.2μm) of SCAO (left), GLAO (right) and the MCAO result (bottom) of the MCAO Demonstrator (MAD) (Marchetti et al., 2007).

demonstrator on the WHT on La Palma demonstrated MOAO in 2010 (Gendron, E. et al., 2011). The first MOAO demonstrated on an 8-m class telescopes was the RAVEN demonstrator (Lardière et al., 2012) on the Subaru telescope which achieved first light in 2014 (Lardière et al., 2015). The reconstruction of MOAO combines multiple WFS measurements and tomography to calculate the corrections separately for each line of sight. Due to the large number of independent DMs, MOAO needs to run in an open-loop configuration, i.e the WFS don't see the corrections applied by the DMs and so the reconstruction step algorithm is generally quite different to those of closed loop AO.

The last major type of AO is known as extreme AO (ExAO). ExAO systems correct a very narrow field of view to deliver a very high Strehl ratio which is essential for the direct observation of exoplanets in conjunction with a chronograph to exclude the starlight for better contrast. ExAO achieves its high level of correction by running the AO loop faster, >1kHz, and using higher order WFS and DMs to be able to correct for high order aberration modes in the turbulent wavefront. Current ExAO instruments include the Gemini Planet Imager (GPI, Poyneer et al., 2014) on Gemini South which achieved first light 2014 (Macintosh et al., 2014), the Spectro-Polarimetric High-contrast Exoplanet Research instrument (SPHERE, Beuzit et al., 2019) on the VLT achieving first light in 2014 (Vigan et al., 2016) and the Subaru Coronagraphic Extreme Adaptive Optics system (SCExAO, Lozi et al., 2018) on the Subaru telescope which saw first light in 2017 (Currie et al., 2017).

1.1.3 ELT-scale AO

For the next generation of extremely large telescopes (ELTs), including the GMT, the TMT and the ESO ELT, good AO correction will become much more important but also much more difficult to achieve. The dependence of AO system requirements on telescope diameter is an extremely important consideration for the ELTs. The computational complexity of the conventional wavefront reconstruction algorithm scales with the fourth power of telescope diameter; this is due to the reconstruction problem size, which is governed by the total number of correcting elements in the DMs and the number of slope measurements from the WFSs, which both scale to the second power of telescope diameter. This presents a huge challenge in the process of designing a real-time controller (RTC) suitable for ELT scale AO, both in the choice of hardware suitable to process the computational demands and with producing software capable of delivering performance that meets the requirements of the AO system.

Many typical AO reconstruction techniques involve computing one or more large

Table 1.2: A summary of first light instruments planned for the ELTs and the types of AO they will use. Also included is the approximate problem size for the wavefront reconstruction for each, adapted and updated from Hippler (2018).

AO Type ↓	Approx.	Telescopes and Instruments ↓		
	problem size	ESO ELT, 39.3m	TMT, 30m	GMT, 24.5m
SCAO	$(9\text{k} \times 5\text{k})^1$	METIS, HARMONI,	NFIRAOS+IRIS	GMTNIRS
→(NGS)		MICADO		
MCAO	$(54\text{k} \times 6\text{k})^2$	MICADO-MAORY	NFIRAOS+IRIS	
→(NGS+LGS)				
LTAO	$(54\text{k} \times 5\text{k})^2$	HARMONI		GMTIFS,
→(LGS)			GMTNIRS	
GLAO	$(36\text{k} \times 5\text{k})^3$		WFOS	G-CLEF,
→(NGS)			GMACS	

[1]Assuming a single high-order WFS of dimensions $\approx 80 \times 80$. [2]Assuming 6× high-order LGS WFS. [3]Assuming 4× high-order NGS WFS.

matrix-vector multiplies (MVMs) where the dimensions of the matrix are defined by the number of input values and degrees of freedoms (DoFs) of the system. The control matrix, which contains information mapping given wavefront measurements to the appropriate actuator commands, is multiplied by an input vector containing the wavefront slope measurements, the results of which yields the wavefront shape required for correction. A MVM operation is defined as,

$$\mathbf{y} = A\mathbf{x} \rightarrow y_i = \sum a_{ij} x_j, \tag{1.13}$$

where $\mathbf{x}$ is the wavefront slope vector of length N, A is the control matrix of dimensions $N \times M$ and $\mathbf{y}$ is the resulting DM command vector of length M, x_i, a_{ij} and y_j are the elements of each respectively. This has computational complexity $\mathcal{O}(NM)$ ($\mathcal{O}(N^2)$ when $N \approx M$). Both N and M scale to the square of telescope diameter resulting in a total fourth power dependence on telescope diameter for the matrix size. Due to this, the MVM becomes very large for ELT-scale operation posing a big challenge for the wavefront reconstruction. The approximate dimensions of the problem size for each type of ELT-scale AO are given in Table 1.2.

A number of instruments utilising AO have been proposed for the first light of the three ELTs. For the ESO ELT, the "High Angular Resolution Monolithic Optical and Near-infrared Integral field spectrograph" (HARMONI, Thatte et al., 2016) instrument will utilise both SCAO and LTAO modes using LGS. The "Multi-Adaptive Optics Imaging Camera for Deep Observations" (MICADO, Davies et al., 2016) instrument will have an SCAO mode and the "Multi-conjugate Adaptive Optics RelaY" (MAORY, Diolaiti et al., 2016) module will provide wider field MCAO correction. Finally the "Mid-infrared ELT Imager and Spectrograph" (METIS, Brandl et al., 2016) will use SCAO firstly with an NGS and later with a single LGS. For the Thirty Meter Telescope (TMT), the first light instrument, the "Infrared Imaging Spectrograph" (IRIS Larkin et al., 2016), will utilise the "Narrow Field InfraRed Adaptive Optics System" (NFIRAOS, Herriot et al., 2014) which will provide SCAO and MCAO operating modes. This will use two DMs, six LGS WFSs and single high order NGS WFS for the SCAO mode. The TMT will also have the "Wide-Field Optical Spectrometer" (WFOS, Pazder et al., 2006) for first light which will operate either in seeing limited mode or with GLAO. The Giant Magellan Telescope (GMT), like the TMT and ELT, will utilise both active optics (AcO) and AO (Bouchez et al., 2014, 2018; McLeod et al., 2014), where the AcO is the active control of telescope optics to compensate alignment and figure errors and the AO is used for atmospheric correction working in three modes, NGS SCAO, GLAO and LTAO with 6 LGS. The GMTs AcO and AO will share the same WFSs and wavefront compensators (segmented primary and deformable secondary) and will be available for all instruments.

Table 1.2 shows an overview of the first light ELT-scale instruments that will be utilising AO corrections, adapted from Hippler (2018).

1.1.4 Real-time control of AO

An AO RTC is the hardware and software responsible for ensuring that the AO correction is computed and applied at the required update rate to effectively cor-

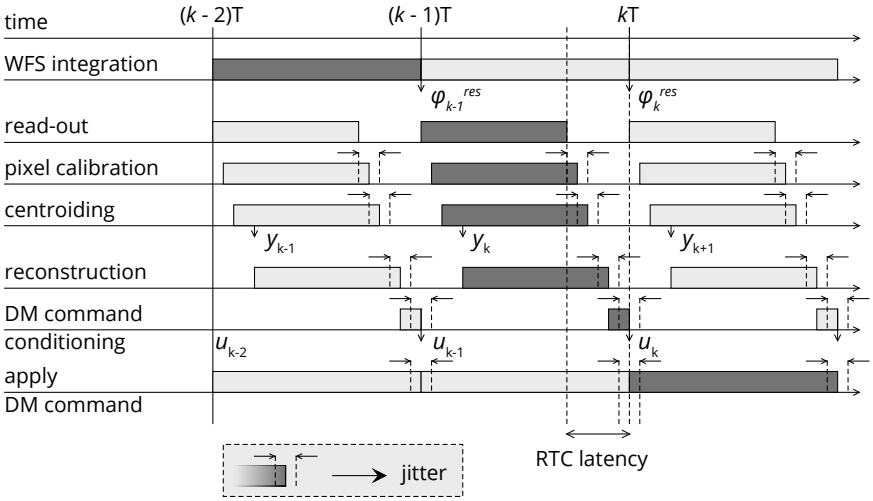


Figure 1.10: Two frame delay AO loop chronogram showing the overlapping computation times of the pipelined RTC operations. The jitter on the RTC latency is shown as the varying end times for each computation step which results in an overall jitter on the time the DM command is applied. The parameter k designates the time step, ϕ^{res} is the residual wavefront phase, y is the wavefront gradient vector, and u is the DM command vector.

rect the turbulent wavefront for the given atmospheric conditions. This is typically of order 1 kHz for visible wavelengths to ensure that correction is applied within an atmospheric coherence time. The RTC is labelled as the controller in Figure 1.1. Figure 1.10 shows the steps needed in the reconstruction pipeline. The RTC is responsible for receiving the WFS images after readout, calibrating the WFS images, computing the wavefront slopes, reconstructing the wavefront, conditioning the DM command using a control algorithm before finally delivering the command vector to the DM. These processes are described further in Chapter 2.

The wavefront reconstruction involves restoring the absolute phase values that are not seen when measuring the local wavefront slopes. If the slope measurements are used directly as a description of the residual wavefront phase, then they will add incoherently and therefore the angular resolution obtainable will be no greater than that of a single subaperture, λ/d, where d is subaperture diameter (Hardy, 1998). Wavefront reconstruction is therefore needed to restore the overall phase relationship between each subaperture that is essential to recover the full angular

resolution of the aperture, λ/D, where D is primary mirror diameter. The process of reconstructing the wavefront is to assemble the individual slope measurements into a continuous 3-dimensional representation of the wavefront. When the sub-apertures are correctly sized and phased, the AO system is capable of producing a potential D/r_0 improvement in angular resolution.

The most common and mathematically straight forward method of reconstructing the wavefront is to directly map the WFS slope measurements to DM actuator commands. This is known as zonal reconstruction as the actuator commands represent phase variations across the different zones of the face of the DM. This is in contrast to modal reconstruction which involves mapping the slope measurements to optical aberration modes defined by a given basis or series, such as the Zernike series described in Section 1.1.1.2. Modal reconstruction then requires a second operation to map the power of each aberration mode to DM actuator commands and is therefore more computationally demanding.

The temporal anisoplanatism described in Section 1.1.1.1 requires corrections to be performed in real-time; i.e. there is a defined time limit between measurement and correction within which the reconstruction must be computed and applied such that the AO performance is sufficient to correct for the atmospheric conditions. The real-time control of AO is therefore a fundamental aspect of the operation of AO and is an extremely important consideration in the design of an AO instrument. For the different types of AO described in Section 1.1.2, the computational complexity increases with each additional WFS or DM mirror in the system. An MVM operation as described above is needed to reconstruct the wavefront for each WFS sensor and map this to required number of DM actuator commands.

1.1.4.1 RTC Latency and Jitter

The *latency* of an RTC is a measure of the time taken to process a WFS image into DM commands and is given as the time between the RTC receiving the last

WFS pixel and delivering the DM command; this is shown in Figure 1.10. This is in contrast to the general AO correction bandwidth, which is determined by the frame-rate of the WFS, or the inverse of the integration time over which the WFS is averaging the guide star signal. The time taken between last pixel in to DM command out, is more appropriate from a RTC point of view as it encompasses only the time taken to process the frame without including the delays caused by sensor readout and DM settling; i.e. only the additional latency that is directly affected by the RTC.

The upper limit of latency required for the RTC is generally given by the atmospheric coherence time as described in Section 1.1.1.1. This places strict requirements on the underlying hardware of the RTC as it must be capable of computing the multiple steps required for reconstructing the DM commands at frequencies of order 1 kHz.

Another important metric for an AO RTC is *jitter*, which describes the variation in latency for the processing of each frame. As mentioned above, the corrections need to be applied in real time and so the ideal RTC latency for each iteration is a constant value which would always meet the requirements of the AO system. However the RTC processing is generally not deterministic and so the latency measurements form a distribution of values, and the jitter can refer to either the shape and width of this latency distribution or to the presence of large outliers in the latency measurements, with different consequences for the magnitude of each type.

The principle of jitter can be seen in Figure 1.10. A single large outlier, where the frame computation time may be instantaneously many times larger than the average, will delay the sending of the DM command and therefore it will no longer be valid for the continuously changing atmospheric conditions. It can also cause the RTC to miss the next frame received from the WFS. In a closed loop system this can have the effect of "resetting" the AO loop which then needs time to settle back to correcting the image. The overall shape of the frame time distribution will generally

have more of an effect on the average PSF size and the more the distribution tails off to longer frame times the larger the PSF will become. Latency and jitter become ever more important in the context of ExAO for extrasolar planet finding where the threshold for latency is much lower than that for other AO regimes and any amount of jitter will have a much greater detrimental effect.

1.1.4.2 ELT-scale AO RTC

Due to the increased problem size when computing the reconstructed wavefront and then preparing the resulting DM commands, the computational performance of the AO RTC for ELT-scale must be many times greater than the performance required for the largest optical telescopes today. The fourth power scaling of the reconstruction DoF with telescope diameter means that there is up to $250\times$ the number of operations required to process each frame of WFS data in an ELT-scale SCAO system. This places strict requirements not only on the computational performance of the AO RTC hardware but also on the interfaces required for the transfer of data from WFSs $\rightarrow$ RTC $\rightarrow$ DMs. All of the next generation ELTs will utilise AO not only in their scientific instruments but also for the alignment and co-phasing of their segmented primary mirror designs.

As mentioned above the computational requirements are even greater for the high order AO types such as LTAO and MCAO as they require the steps of image calibration, wavefront gradient calculation and wavefront reconstruction to be performed for each WFS. The size of each of the wavefront reconstruction calculations is also increased for each additional DM present in the system, with each resulting DM command vector needing to be conditioned appropriately before being delivered to the correcting devices. As shown in Section 1.1.3, there are many proposed instruments for the ELTs which will require the use of AO and so the development of an RTC platform capable of performing these operations in real-time is paramount to the success of these ELT instruments. Another important consideration due to the increased physical size of the telescope enclosures of the ELTs is the increase

in aberrations due to wind induced vibrations. This will have much more of an impact compared to current telescope designs and therefore will require proper characterisation, and additional algorithms will need to be added to the control processing to mitigate their effects on observations.

1.2 Real-time Controller Hardware

The large MVM calculations needed for the wavefront reconstruction is a memory-bandwidth bound problem. This is because the processing time during each AO RTC cycle is dependent on the rate at which the processing hardware is capable of reading the large control matrix from main memory due to it being too large (typically several GB) to be stored in the much faster but smaller cache. This favours computational architectures with large amounts of memory with high memory bandwidths. The Intel Xeon Phi (Intel, 2017a) is an example of such an architecture, with the Knights Landing (KNL) model having an on-chip 16 GB MCDRAM package, giving a measured memory bandwidth as high as $480\,\mathrm{GB\,s^{-1}}$, measured using the STREAM benchmark (McCalpin, 1995).

The first consideration when designing an AO RTC is to choose hardware that can meet the requirements of the AO system, both in terms of input and output (I/O) interfaces for the instruments and in terms of computational performance for the algorithms required. The computational requirements are largely dictated by the reconstruction problem size. Historically, a variety of hardware architectures have been used for AO RTC, including digital signal processors (DSPs, Fedrigo et al., 2006), field programmable gate arrays (FPGAs, Fedrigo et al., 2006; Rodríguez-Ramos et al., 2012), central processing units (CPUs, Basden et al., 2010) and graphics processing units (GPUs, Basden et al., 2010; Truong et al., 2012).

These architectures have proved capable for previous and current AO systems with varying advantages and disadvantages for each. The main disadvantage with DSPs, FPGAs and GPUs is the time cost associated with designing, writing and, if nec-

essary, modifying the RTC software. The main advantage of FPGAs and DSPs is deterministic behaviour. Due to their more general computing nature, CPU-based systems can be at a disadvantage when it comes to some specific computation problems, such as highly parallelisable problems which may be better suited for GPUs; however CPU systems have a large selection of programming tools and libraries to aid development, and are generally backwards compatible with common programming languages. For ELT-scale systems, two of the main challenges are scaling of these systems for the increased computational complexity (usually requiring many of these devices working in parallel), and future proofing the development of the system for updated hardware.

Within the past 5 years advances in processing technology have allowed AO RTCs to operate with the required latency for 8-10 m class telescopes on off-the-shelf server CPU and graphics processing unit (GPU) technologies. For the next generation of ELT-scale telescopes, GPUs have demonstrated the capability of accelerating the various computational tasks involved. However, though modern GPUs have enormous compute capabilities, they are limited by the fact that these only apply to tasks that can be sufficiently parallelised and they can only be used as an accelerator and not as the host processor. The latter fact generally means that the demanding computations must all be offloaded to the GPU for computation and the results then copied back to the host CPU, which can introduce an increase in latency and complicate the software so that it is no longer portable. Portable software code is important for being able to implement the RTC software and algorithms for different AO systems and on different processing hardware quickly and easily. A solution to the GPU offload problem has been developed by the Green Flash project (Gratadour et al., 2018) by utilising the technique of direct memory access (DMA) which can allow a network device to receive WFS images and copy them directly to the internal GPU memory for processing and then the results can be copied directly back for sending to the DMs.

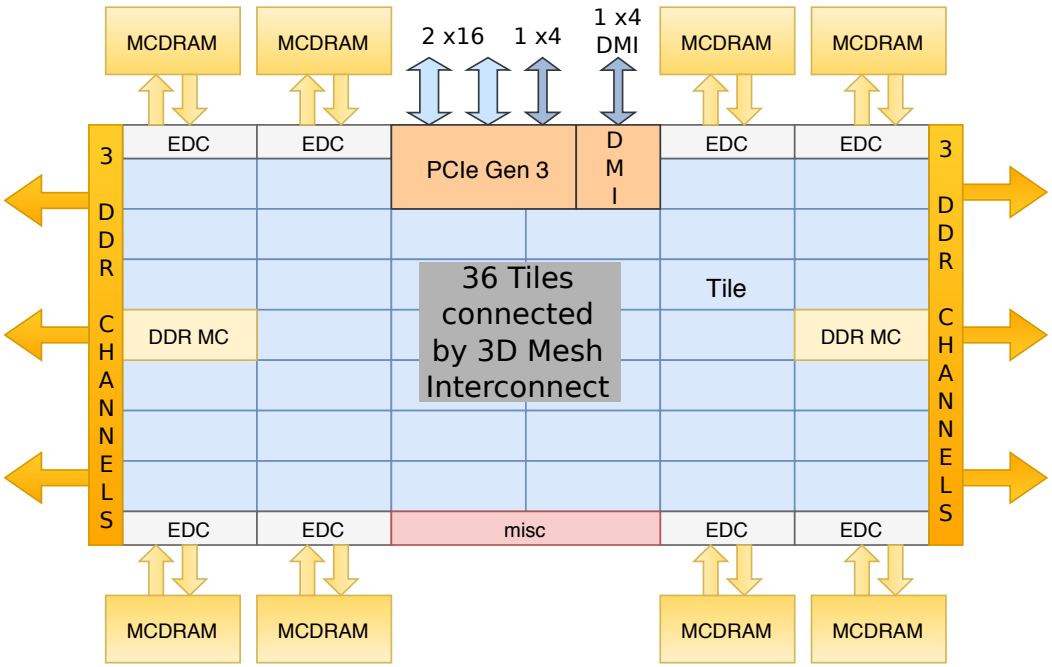


Figure 1.11: Schematic of Xeon Phi Knights Landing CPU showing the MIC architecture along with the high bandwidth MCDRAM. Each tile contains two CPU cores, two vector processing units per core and a shared 1MB of Level2 cache. (DDR MC = DDR memory controller, DMI = Direct Media Interface, EDC = MCDRAM controllers, MCDRAM = Multi-Channel DRAM (Intel, 2016))

1.2.1 CPU systems

1.2.1.1 Xeon Phi Knights Landing

The Intel Xeon Phi processor family combines many low power x86 CPU cores utilising wide 512 bit vector registers with high bandwidth on-chip memory to enable acceleration of highly parallelisable tasks, while keeping the cores sufficiently fed with data. These x86 cores use the backwards compatible x86 instruction sets which are used in the vast majority of Intel and AMD based CPU systems. This allows the Xeon Phi to leverage the benefits both of having a CPU-based architecture and of having a highly parallelisable work flow similar to that of GPUs. These attributes of the Xeon Phi make it a very interesting candidate for an ELT-scale AO RTC as they can be developed using conventional CPU programming techniques. However due to the relatively low performance of an individual Xeon

Table 1.3: Available Knights Landing models and their key specifications. The peak single precision (SP) TFLOPS is a theoretical calculation resulting from the core count, the clock speed (-200MHz for 512-bit vector operation), the vector register size, the number of vector process units per core and the number of floating point operations per fused-multiply-add. e.g for the 7210 model: $64 \times (1.3 - 0.2) \times 512/32 \times 2 \times 2 = 5325$ GFLOPS. The memory bandwidth is that as measured using the STREAM triad benchmark.

KNL Model	Core count	Base CPU Clock Speed (GHz)	Peak SP TFLOPS	Memory Bandwidth ($GB\,s^{-1}$)
7210	64	1.3	4.51	450
7230	64	1.3	4.51	-
7250	68	1.4	5.22	480
7290	72	1.5	5.99	-

Phi CPU core, properly utilising vectorisation and parallelism is essential for good performance.

Knights Landing (KNL) is the third generation Xeon Phi processor and is the first to be released in the self-booting socketed form factor, with a number of variants, as given in Table 1.3. A schematic diagram of the CPU layout of the KNL processor is shown in Figure 1.11. Previous generation Xeon Phi chips were available as accelerators only. The KNL processor can therefore be used just like a conventional server processor and can run the Linux operating system and standard software environment. Existing applications can be ported to the Xeon Phi quickly: recompilation is usually not even required, though code will not be well optimised in this case.

The KNL introduces additional instruction sets such as AVX-512 which can be utilised for improved performance. The AVX-512 instructions work with 512 bit CPU registers, which allow Single Instruction Multiple Data (SIMD) operation on 16 single-precision floating point numbers simultaneously per core, per instruction cycle. This improves the parallelisation advantage of the KNL architecture over previous processors, which included a maximum of 256 bit wide vector registers. This 512 bit register is also included in forthcoming (and most recent) standard

Intel CPUs, so any code optimisations made for this feature will also be applicable to future non-Xeon-Phi CPUs. However, it should be noted that for the KNL system, high utilisation of 512 bit instructions reduces the base core clock speed by 200Hz, which is taken into consideration in the peak SP TFLOPS calculated in Table 1.3.

The wavefront sensor cameras can be directly attached to a KNL server via the PCIe bus. This is an advantage over accelerator cards such as the previous generation Xeon Phi cards and GPUs, where, unless specific effort is taken (often requiring specific custom network cards, and low level device programming), image data must be transferred first to the CPU from the camera, and then to the accelerator (essentially 2 PCIe transfers, with increased latency and jitter). The previous generation of the Xeon Phi, the Knights Corner, was only available in an accelerator form factor and has also been investigated for AO RTC (Barr et al., 2015), which showed promise for continuing the investigation to future processor generations (i.e. the KNL).

1.2.1.2 Multi Socket CPU Systems

While the properties of the Xeon Phi KNL make it ideally suited for the processing of ELT-scale AO, there are also available other x86 CPU-based systems that can provide the required performance. Multi socket CPU systems incorporate two or more CPU packages onto a single motherboard to create a processing node with greater computational performance. Each CPU package has its own memory channels and using non-uniform memory access (NUMA), each CPU socket is able to access the memory channels of the others. However the main advantage of NUMA is that the sockets' memory channels are grouped into what are called NUMA nodes. This allows the operating system (OS) and the NUMA library to specify which NUMA nodes it should use for the allocation of specific blocks of memory. This can allow multi-threaded applications to not only specify which CPU cores to execute certain threads, but also which NUMA memory node the

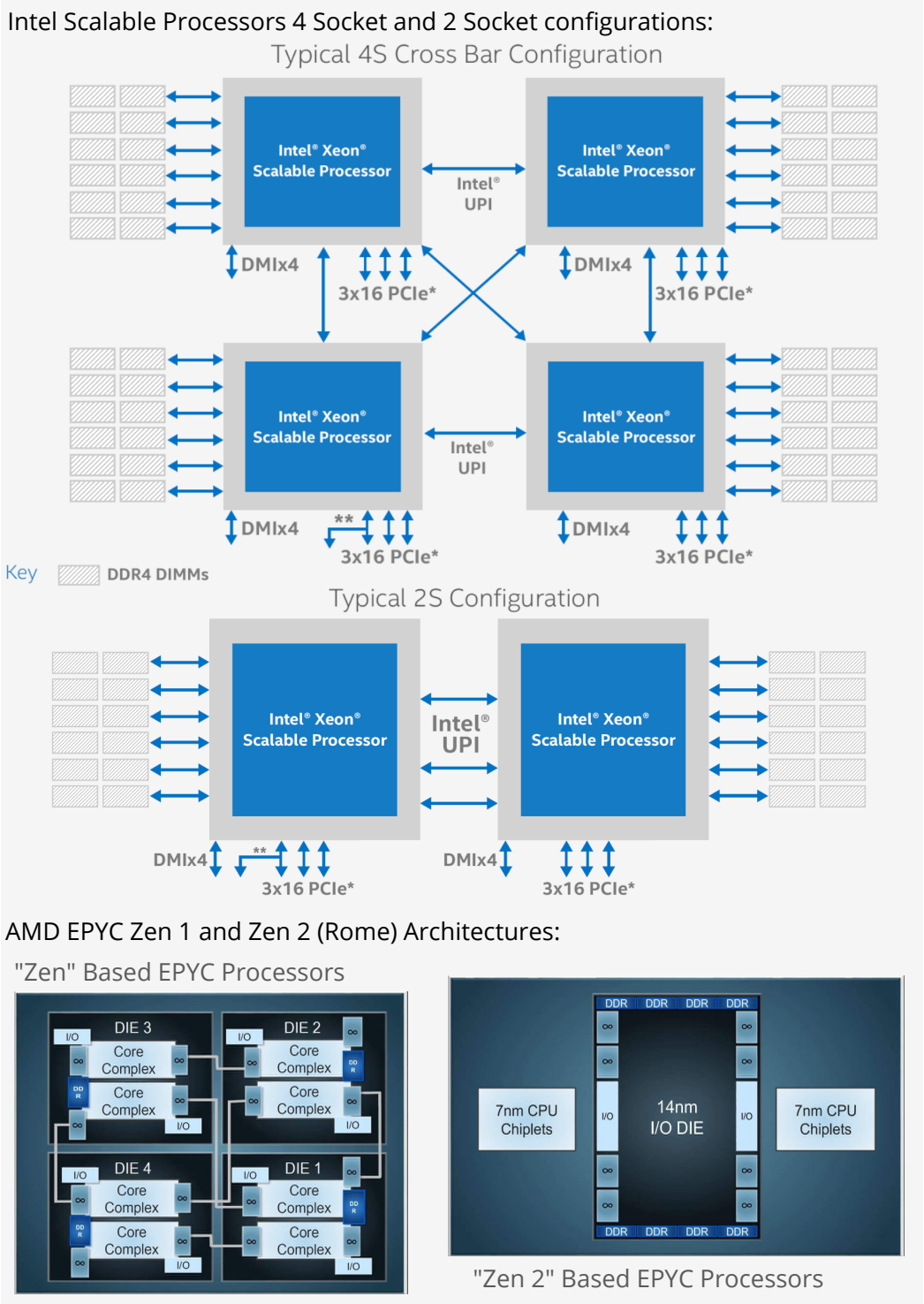


Figure 1.12: Many-core CPU architectures with NUMA topology to acheive the required memory bandwidth for ELT-scale AO RTC. The Intel configurations and the AMD architecture are described in Section 1.2.1.2.

thread should have its memory allocated to. In theory, multi-threaded software that uses the NUMA node allocation correctly can greatly increase the usable memory bandwidth by allocating the threads equally among each CPU socket and allocating the memory for each on the local NUMA nodes. The theoretical memory bandwidth available on NUMA systems is therefore the sum of the memory bandwidths of each individual socket.

Having multiple CPU packages on the same motherboard also allows the processors to work in parallel with much faster data transfers between them than if they were on separate nodes. It also simplifies the software environment as a single OS is able to schedule tasks and synchronise all processors. The main downside to multi-socket systems is that the link between each processor can become a bottleneck in the processing performance if a lot of data is needed to be shared between them. It is therefore necessary to reduce the amount of synchronisation and data transfers between processing threads to ensure peak performance and allocate large memory structures to the correct node to avoid threads needing to frequently access the memory on non-local nodes.

x86 CPUs: Intel and AMD The most common type of desktop, server and high performance computing (HPC) central processing unit (CPU) architecture is the x86 architecture. Due to its ubiquity there is a wide selection of tools and libraries available for x86 application development and a number of operating systems (OS's) are available. There is also continuous development of x86 CPUs from the two main vendors, Intel and AMD, meaning that there is a constant update to performance and capability, and, as mentioned above, the Xeon Phi is also based on this architecture. Both Intel and AMD currently produce multi-socket CPU systems with total memory bandwidth capabilities that are comparable to the Xeon Phi.

Intel CPUs are generally better performing than AMD CPUs, having greater clock speeds, more advanced feature sets, faster memory channels, and until recently

much better instruction-per-cycle, which is a measure of how many CPU instructions can be processed each CPU clock cycle. One of the main advantages of modern Intel CPUs relating to AO RTC is the support for the AVX-512 instruction set which was first introduced with the Xeon Phi, and some high-end Intel CPU models incorporate two of the AVX-512 vector processing units per CPU core, further increasing the potential for accelerating vectorisable workloads. The NUMA architecture of multi-socket Intel CPU systems is shown in Figure 1.12 for both a quad-socket (4S) and a dual-socket (2S) configuration. The CPU are connected via the UPI links and each has up to 6 memory channels available, giving a maximum of 24 memory channels in the 4S configuration.

AMD CPUs, whilst generally not performing as well as Intel's products, still have a few advantages. The biggest advantage when considering building an RTC for an AO system is the cost of the processors. AMD processors can be much cheaper than Intel CPUs with comparable specifications. AMD CPUs also have more memory channels per socket than Intel CPUs, with the current generation EPYC processors having up to 8-channels per socket, compared with 6 for Intel, giving a higher theoretical memory bandwidth per socket. The Zen architecture of the current AMD EPYC processors is shown in Figure 1.12. A single Zen based EPYC CPU is made up of four individual dies called Zeppelins, which themselves have two memory channels each and two core complexes. Each core complex has up to four active CPU cores, giving a maximum of 32 cores per CPU package. A downside of the Zen architecture is that each of the dies is a separate NUMA node with reduced memory bandwidths between each one. A dual-socket EPYC system will then have a total of 8 NUMA nodes which can complicate the memory allocation scheme and introduce extra latency between nodes.

The next generation "Zen 2" architecture, due for release in 2019, will simplify the memory access by consolidating all memory channels for a single CPU package onto a single I/O die and then having up to 8 connected CPU chiplets containing the CPU cores. This is shown in Figure 1.12. This gives each of the CPU cores

equal access to all 8 memory channels of the CPU package, potentially reducing latency and simplifying NUMA memory allocations.

ARM CPU Architecure Another interesting hardware candidate for AO RTC is the ARM architecture. First developed in the 1980s as co-processors it has since been developed into a fully fledged CPU architecture that is used primarily for small form factor low power devices such as smart phones, tablets and laptop computers. However due to recent advances in performance, and it being available to license by 3rd party manufacturers, the ARM architecture has seen use in a number of server and HPC related processors. One of the most recent announcements was the Fujitsu a64fx (Shimizu, 2018) which is a HPC processor aimed at supercomputing applications. Its specification includes up to 48 CPU cores with the ARM scalable vector extensions, with up to 512-bit vector registers similar to the Intel AVX-512 instructions. It will also have on-chip high bandwidth memory for a maximum memory bandwidth of 1024GB/s and the capability of performing native 16-bit floating point operations, which can reduce the memory bandwidth requirement. It is unlikely that this CPU will be available as a stand-alone system as it is being developed for integrating into a supercomputing cluster, however the fact that an ARM CPU with these specifications has been developed means that there is potential for similar devices to be available in the future.

A different type of ARM CPU device is the Mellanox BlueField smart network interconnect (Mellanox, 2018). This device is a network interface controller (NIC) with an integrated ARM based system-on-a-chip (SoC) with up to 16 ARM CPU cores and up to 16GB of on-board DDR4 memory. Whilst these specifications wouldn't make it suitable for processing the entire ELT-scale RTC pipeline, it does present an interesting capability as a wavefront processing unit (WPU). This means it would perform the less intensive operations of pixel calibration and wavefront gradient measurements and only transfer the wavefront slopes to the host processor to complete the wavefront reconstruction. As it is itself the NIC, the WFS pixels

arrive directly for immediate processing and due to the much smaller memory footprint of the wavefront gradients compared to the pixel data, the latency of data transfer to the host can be reduced substantially. The processing of network packets would also be offloaded to the Bluefield SoC which would further reduce the latency for devices with relatively weak single threaded performance such as the Xeon Phi. The precursor of this device was investigated for AO RTC use by Barr et al. (2015).

1.2.2 Hardware Accelerator Cards

1.2.2.1 General Purpose GPUs

In recent years there has been a push in the HPC world to utilise general purpose graphics processing units (GP-GPUs) more and more for the acceleration of highly vectorisable and parallelisable workloads. The processing of computer graphics involves streaming lots of data at a high rate whilst performing the same mathematical operations on large chunks of the data at once. This is known as the single instruction multiple data (SIMD) paradigm as it involves performing a single sequence of operations on many individual data points at once. This idea also works very well for large mathematical operations such as matrix multiplications and MVMs and so these graphics processing unit (GPU) devices are now used for accelerating more general purpose processing workloads.

A GP-GPU is inherently an add-in accelerator card to be used with a host CPU system and therein lies their most fundamental disadvantage for working on large data sets; the data need to first be transferred to the device before processing can begin. Previous investigations into using GPU systems to accelerate AO RTC have been conducted (Bitenc et al., 2018) by using the device solely to process the most computationally demanding aspect of the reconstruction pipeline. There have also been efforts to use GP-GPUs to fully process the entire RTC pipeline by using them in conjunction with separate dedicated network cards to allow data to be

copied directly from the memory of one to another without needing to go through the host CPU.

As mentioned in Section 1.2 the GreenFlash project developed a solution to the data transfer problem with GP-GPUs by using direct memory access to transfer the incoming pixel data directly to the GPU memory and process the entire RTC pipeline on the GPU. However this a non-standard solution involving collaboration with the manufacturers of the devices and using advanced techniques to ensure that the GPU can continually process the pipe-line. GPUs normally work by processing a certain set of input data using a kernel; the kernel defines the operations to be applied, their order and how the output should be returned. Normally, a host processor dispatches data and a kernel to the GPU devices and receives the output once the kernel has completed. The GreenFlash project developed a persistent kernel which is capable of operating continuously on data transferred directly from the network device without any input from the host CPU. This all has to be written in the specific proprietary language for the GPU device and is therefore non-portable to other devices. The feature that allows direct memory access is also not available on all GPU devices, even from the same manufacturers, and so this solution is very specific and difficult to transfer to other hardware.

1.2.2.2 Xeon Phi Knights Corner

Both the Knights Corner (KNC) and KNL Xeon Phi processors were available as co-processor accelerator cards in the same format as GPUs but KNL was the only one also available as a socketed host processor much like a standard CPU. The accelerator cards suffer from the same offloading problems as GP-GPUs mentioned above, however as the KNC is based on the x86 based architecture, like the vast majority of current server CPUs, standard programming code can be ported directly to it without the need for much, if any, modification. The main caveats with the KNC co-processors are that the many cores available are comparatively much slower individually than those of a standard CPU and while no modifications to

the source code are required (other than specifying the operations to be offloaded), certain optimisations still need to be considered in order to achieve optimal performance on these devices.

1.2.3 FPGAs and DSPs

Field programmable gate arrays (FPGAs) and digital signal processors (DSPs) have been used quite extensively for AO RTC. An FPGA is an integrated circuit processor that can be re-configured for different purposes, hence the name "field programmable". They tend to be integrated into an add-on accelerator form factor, similar to network devices and GPUs, however they can operate independently of a host CPU. They are made up of an array of programmable logic blocks connected with reconfigurable interconnects that allow the blocks to be wired up in different configurations. The configuration is written in a hardware description language (HDL) which is a specialised programming language to describe the structure and behaviour of digital logic circuits. Once the configuration has been created, the FPGA kernel then needs to be built and deployed to the device. This is generally a time consuming process, of order hours, and so rapidly prototyping and testing FPGA kernels is infeasible.

Due to the complexity of designing the HDLs and the steep learning curve of the language, there have been efforts to introduce high level languages that abstract a lot of the underlying functionality into easy to use programming tools, such as the FPGA add-in tool for LabVIEW and the QuickPlay software from Accelize. There are also libraries of complex functions available that are pre-made and optimised for certain tasks; these are referred to as intellectual property (IP) cores and are generally available to license from FPGA vendors and third-party IP suppliers. However due to the costs associated with licensing the IP cores that are most optimal for specific operations, it may not be an ideal solution for AO RTC which would need to license the IP for the lifetime of the system, which can be of order decades.

A DSP is generally much simpler than an FPGA; it's a specialised microprocessor which is optimised to carry out the large number of repeated mathematical operations that are needed to process digital input signals. They are much more efficient than general-purpose microprocessors programmed for the same task and are able to achieve better performance allowing them to process signals continuously in real-time. The DSP instruction sets are optimised for the mathematical operations that are needed in signal processing but the instruction sets would be considered highly irregular compared to those for general purpose processors. To achieve the best performance, the DSP software needs to be hand-written as even modern compilers are unable to properly optimise for the architectures. This is a significant drawback if the algorithms need to be changed and development can be a very time consuming task.

The main benefits to using FPGAs and DSPs is that they are very deterministic with their processing; the time taken to complete their preprogrammed routines is very consistent and results in an AO RTC that has very low jitter. They are also more efficient at performing the mathematical operations needed for processing the input signal, however they can struggle with more complex algorithms due to their more limited instruction sets compared with general purpose processors. It is also typical for DSPs to have lower memory bandwidth than other devices; it would therefore be extremely difficult to scale a DSP based system for ELT-scale AO RTC operation. The time and effort factor in designing the software also makes it impractical to design many different algorithms for different situations which makes them essentially fixed function processors after their initial implementation.

Even though FPGAs are often used independently, they can also be used with a CPU host machine such that they act as either an algorithm accelerator or as a smart network interface. This allows the FPGAs to perform the more simple operations that are continuously needed and the general purpose host machine can then perform the more complex algorithms which can be more easily modified. Until recently the computational performance of CPU based systems has not really

allowed this configuration and FPGA based AO RTCs have been implemented entirely on stand-alone FPGA and DSP hardware. There is also the possibility to use FPGAs with GPUs to leverage the benefits of each and overcome some of their more serious drawbacks.

1.3 Thesis Synopsis

Chapter 2 will expand upon some of the RTC concepts introduced in this chapter and give more detail on the theory behind the most commonly used reconstruction techniques. It will then go on to describe the RTCs that are used to control some of the AO systems described in Section 1.1.2, their suitability for ELT-scale, and other ELT-scale AO RTC investigations. It will conclude with a discussion of the suitability of many-core CPUs for the processing of ELT-scale AO RTC.

Chapter 3 will introduce the AO RTC software used in this thesis along with an implementation of a CPU-based software camera simulator which is used to simulate the pipelining of pixels over a network interface. It will then cover the optimisations made to the software and host machines to enable optimal performance for ELT-scale AO RTC.

Chapter 4 presents results and discussion of testing the updated and optimised RTC and camera simulator software in the SCAO regime of AO using Intel Xeon Phi hardware. Results are presented for both Shack-Hartmann and pyramid WFS processing and for the performance optimisations discussed in Chapter 3. The work in this chapter was initially presented in Jenkins et al. (2018b).

Chapter 5 introduces a prototype architecture for processing both MCAO and LTAO AO RTC on multiple nodes of Intel Xeon Phi hardware. It also presents results and discussion of testing the prototype architecture using the camera simulator to stream pipelined WFS images to all reconstruction nodes. The work in this chapter was initially presented in Jenkins et al. (2019).

Chapter 6 discusses performance evaluation of AO RTC for ELT-scale systems beginning with a comparison of classical least squares reconstruction and optimal

LQG predictive control. It will also presents results of using a reduced version of the MCAO and LTAO architecture described in Chapter 5 to process SCAO RTC on multiple CPU processing nodes and a discussion of the performance benefits at the cost of increased complexity. It will conclude with preliminary results of testing the AO RTC software on a number of NUMA aware multi socket systems similar to those discussed in Section 1.2.

Chapter 7 finally concludes with a discussion of the work presented in this thesis and a look ahead at the future investigations to continue the work.

Chapter 2

Real Time Control

All adaptive optics (AO) systems need to update the wavefront correction at a rate that is consistent with the given atmospheric conditions such that the turbulent phase hasn't evolved sufficiently to render the measurements obsolete. To make sure that the computations are completed in the time allotted the AO control should be processed in real-time: there should be a maximum acceptable computation time delay in which all frames should be processed. This is the basis for real time control of AO.

2.1 The Wavefront Reconstruction pipeline

A general instruction pipeline refers to a set of data processing elements connected in series where the output of one element is the input to the next. The processing of wavefront measurements into DM commands is made up of a number of distinct steps with the flow of data following a set path. The AO RTC pipeline is the process of transferring the wavefront information from the wavefront measurement devices, processing it to recreate the incident wavefront and using the information to build control commands to deliver to the wavefront correctors.

For ELT-scale AO systems, the majority of the computation time is spent in the reconstruction of the turbulent wavefront and incorporating this into the control

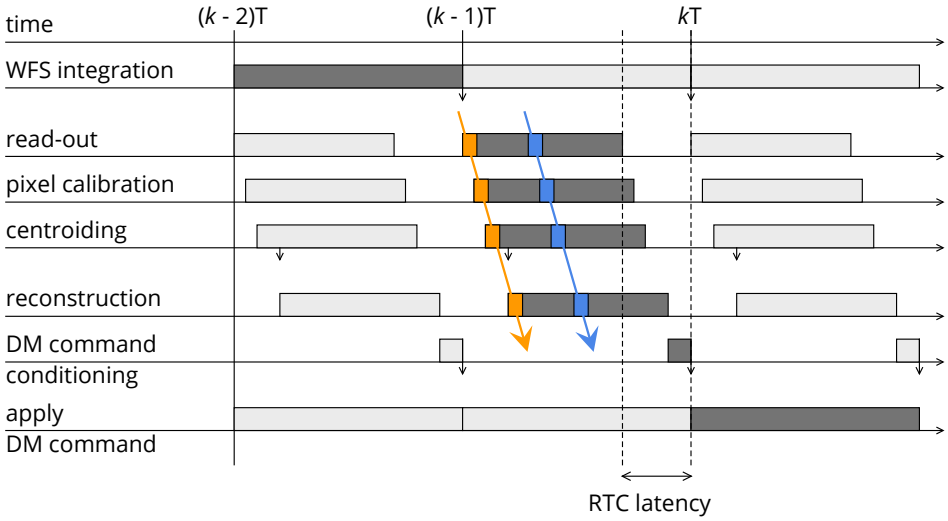


Figure 2.1: An AO RTC two-frame delay AO loop chronogram, showing the pipeline of information as it arrives from the WFS and is processed to obtain the DM command. Single pieces of data, e.g. as shown in orange and blue, are pipelined by different processing threads.

law that controls the time evolution of DM commands. This entire process can be split into 5 main sub-processes:

- WFS image acquisition
- WFS image calibration
- Wavefront gradient calculation
- Wavefront reconstruction
- Conditioning the DM command with the given control law

These are needed for all types of AO operation and the data pipeline is shown in Figure 2.1. Section 2.1 describes these 5 main processes and Section 2.2 gives a description of the most common control law used as well as two other more complex control laws which are used to improve the correction. The descriptions that follow are based on the operation of closed-loop SCAO unless otherwise stated.

2.1.1 WFS Imaging

The first step in the RTC pipeline is to acquire the pixel data from the WFS cameras. These WFS cameras can have different interface formats, and so most current and previous AO systems have been bespoke, designed to work with a specific camera. In recent years however it has been more desirable to utilise detectors from camera vendors which use industrial camera interfaces such as Camera Link or GigE Vision. However the European Southern Observatory are developing their own camera interface to cope with the very high data rates of some of the WFS cameras to be used on ESO ELT. This camera interface, named MUDPI, will use the User Datagram Protocol (UDP) network communications protocol to transmit the frame data from the WFSs to the RTC hardware.

In order to improve the AO performance, a very useful technique when processing the WFS images is to pipeline the processing of pixel data, and to process small packages (subapertures) as soon as enough data has arrived, rather than waiting for the full image. This will be especially important for ELT-scale AO systems as they will generally require larger image formats that will take more time to read-out and transfer. Pipelining then allows the RTC to utilise the read-out and transfer time for processing, further reducing the apparent latency. This thesis will henceforth only consider the situation where pipelining can achieved, which involves the transfer of image data in fixed size chunks or packets, smaller than a full frame, which can be processed by the RTC as soon as they arrive.

Regardless of the individual camera interfaces used for acquiring the WFS images, there are several properties of the camera image transport that need to be considered. These include the bits per pixel (bpp), the individual packet size, and the location of the read-out ports on the detector, which affects the order that pixels arrive. The bpp informs the RTC how many bits of data are used per pixel to store the intensity information. The detector will usually record pixel values of 8-16 bpp and format them as either 8 bit or 16 bit integer values. These have a range of 256

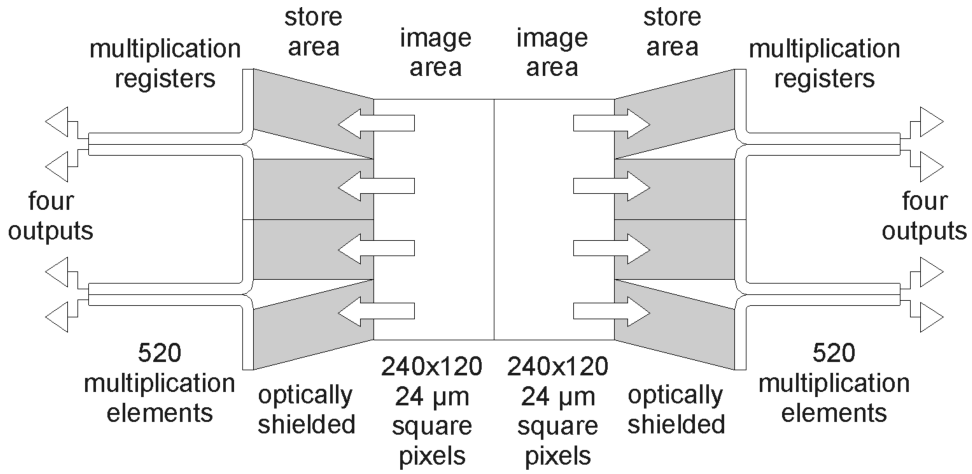


Figure 2.2: Schematic of the e2v technologies 240x240 pixel L3Vision CCD220 showing the 4 detector read-out ports on each side of the device. The layout of the read-out ports needs to be taken into consideration when receiving the WFS pixels. (Downing et al., 2018)

or 65536 counts respectively. The bpp is a measure of the range of intensity values that can be retrieved from the exposure. The bpp and packet size information is vital for the RTC so it can properly store the image data as the packets arrive and convert it into floating point values for further processing.

If a charge-coupled device (CCD) detector is used in the WFS, the number of read-out ports of the detector affects the ordering of pixels during the pixel transfer. The pixels are generally read out line by line and a WFS camera detector with a single read-out port will read out the pixels contiguously (all in order) from top to bottom and so the RTC simply needs to place the pixel data in a contiguous section of memory as it arrives. However to create larger sensors and improve the read-out speed it is often helpful to have multiple read-out ports on a single detector. For example the Teledyne e2v CIS124 Large Visible Sensor Module (LVSM, Downing et al., 2018; Jorden et al., 2018), which will be used for the ELTs SH-WFS cameras, has a usable 800×800 pixel area with its readout split into the upper and lower sections of the device meaning that the pixels from the top and bottom will be interleaved during the read-out. Another sensor to be used with ESO ELT is the e2v technologies L3Vision CCD220 (Jorden et al., 2018) which will be used for the

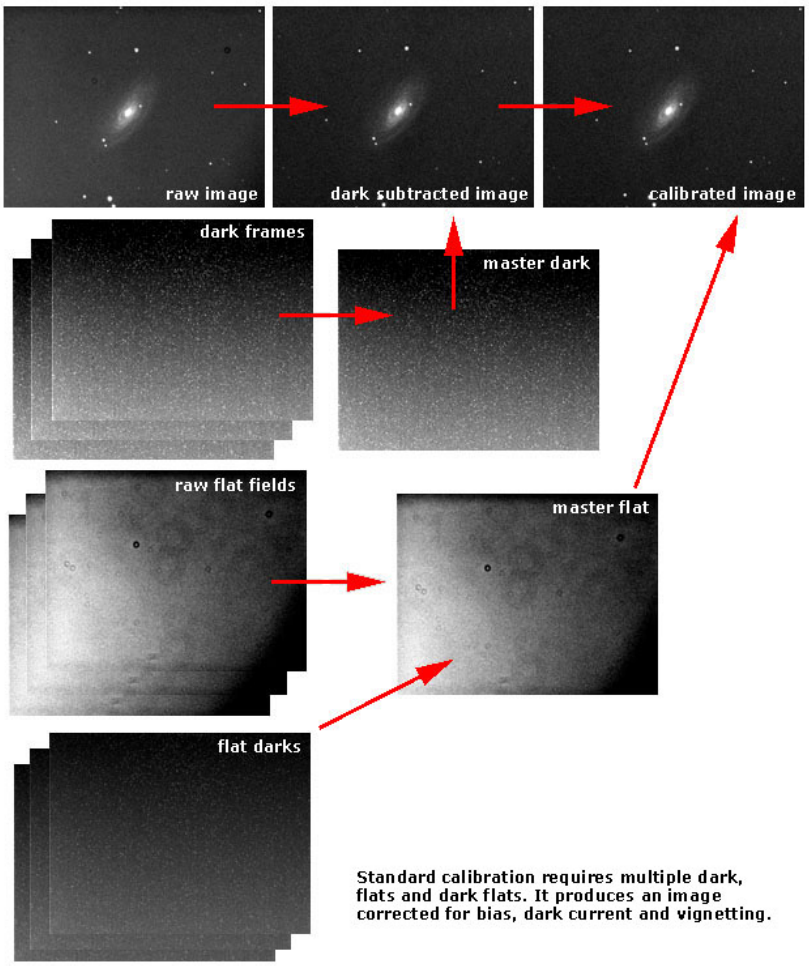


Figure 2.3: An example of image calibration for an astronomical image; WFS calibration is a similar procedure. (Ré, 2019, reproduced with permission)

Pyr-WFS and has a usable 240×240 pixel area with a total of eight read-out ports, four on each side of the device; this can seen in Figure 2.2. As the pixels arrive out of order with rows from the both top and bottom interleaved, the RTC must be aware of the format that the read-out takes such that it can properly store the pixels in the correct order.

2.1.2 Image Calibration

Once the pixels have arrived at the RTC, the first step in the image processing pipeline is the pixel calibration step. This is necessary to reduce the amount of

noise and unwanted signals in the image. The noise present in the image comes from a number of different sources. The first of these is the camera sensor itself which will have a characteristic dark noise and read out noise. The read-out noise cannot necessarily be corrected using calibration as the amount will vary per pixel and from frame to frame. The dark noise however can be corrected by using a dark frame which is an average of several images taken when there is no light incident on the detector, hence the name "dark" frame. The dark frame will capture any fixed-pattern noise from the detector which is constant from frame to frame and is caused by differing dark currents in each pixel.

Another source of noise or unwanted signal comes from distortions in the optical path and from irregularities of the detector surface. These cause unwanted intensity variations from pixel to pixel. These can be artefacts resulting from vignetting of the field or simply shadows from dust in the optical path. This can be mitigated by taking a flat field image which is an average of several images taken when there is a uniform field of light evenly distributed across the field of view. This information is then combined with the dark frame as described above to calibrate both the sensor dark noise and the optical system's flat field variations.

Some sensors, such as CCDs that use all-gates pinning (AGP, Bosiers et al., 1993) can have a very low dark current but will still have bias associated with each pixel, i.e. a non-uniform pixel-to-pixel deviation in the signal received from the camera. This bias is independent of exposure and so can be represented by a bias frame which is, ideally, an average of several images taken with an exposure of zero length.

The last major source of unwanted signals comes from the sky background, which is the signal measured by the detector when looking at an empty patch of sky taken a short time before or after observations. The source of this unwanted light can be from general light diffusion from the atmosphere or from light pollution from nearby sources.

The calibrated images are usually obtained using the following equation,

$$\text{Calibrated} = ((\text{Raw} - \text{Dark}) / \text{Flat}) - \text{Background} \tag{2.1}$$

Once the image has been calibrated, for AO RTC image processing it can also be beneficial to perform a thresholding of the pixel vales, setting any pixel values that fall below a certain threshold to either zero or a pre-determined minimum value. This can be an important way to remove any un-calibrated noise that is still present in very small pixel values and ensures that no values fall below zero. Figure 2.3 shows an example calibration process for astronomical images; this is similar to WFS image calibration.

2.1.3 WFS Slope Calculation

Once the WFS image has been calibrated, the images must be further processed to obtain the wavefront information. As mentioned in Section 1.1.1 the two most common types of WFS sensor used in astronomical AO are the Shack-Hartmann WFS (SH-WFS) and the pyramid WFS (Pyr-WFS) which both function by measuring the local wavefront slopes at a number of discrete points across the pupil. These are currently the only WFSs being considered for first light ELT-scale AO. However other types of WFS exist such as curvature WFSs (Roddier, 1988) and interferometric WFS such as ZELDA (N'Diaye et al., 2013), CAWS (Bharmal et al., 2012) and the Mach–Zehnder interferometer (Delacroix et al., 2015). Interferometric WFS can be used for continuous calibration of AO systems at a slower update rate than the main RTC rate. In this role they are not a primary driver of RTC specification or design.

2.1.3.1 Shack-Hartman WFS Processing

The operation of a SH-WFS is demonstrated in Figure 2.4. It shows that the images obtained by the SH-WFS camera form a regular grid of subaperture spots and the

deviation of each spot from the centre of its subaperture gives the local wavefront slope at that point on the pupil. The number of SH-WFS subapertures needed across the pupil for general correction is given roughly by the telescope primary mirror diameter divided by the average seeing, r_0, at the observing site, $N \approx D/r_0$. This is so that when projected onto the size of the primary mirror, each subaperture is roughly r_0 in diameter. The reason for this is to ensure that the image formed by each subaperture is in general unaffected by modes of aberration of greater order than the first 3 modes of piston, tip and tilt (see Section 1.1.1.2). The individual subaperture images will therefore be approximately diffraction limited, giving a well defined image spot whose position is solely affected by the local wavefront gradient.

The SH-WFS lenslet optics are designed such that for the number of subapertures and the detector used, there are equal numbers of pixels dedicated to each subaperture. This information is then required by the RTC software such that the pixels of each subaperture can be copied into the right memory location before the slope computation can begin. Once the subaperture pixels have been copied to the correct memory location, the local slope for each one needs to be calculated. There are three major types of wavefront gradient calculation for SH-WFS that are used depending of the specific AO system in use. These are centre of gravity (CoG), correlation wavefront sensing and matched filter gradient calculations

Centre of gravity The most common slope computation algorithm for SH-WFS is a CoG calculation over all pixels in each of the subapertures. The CoG involves summing the intensity values of each pixel multiplied by their coordinates within the subaperture along both axes of the pixel array independently. The resulting slope values are then divided by the total flux in the subaperture to normalise their amplitudes. The CoG equation is defined as,

$$\mathbf{S}_{\mathrm{CoG}} = (x_c, y_c) = \frac{\sum_{ij} X_{ij} I_{ij}}{\sum I_{ij}} \tag{2.2}$$

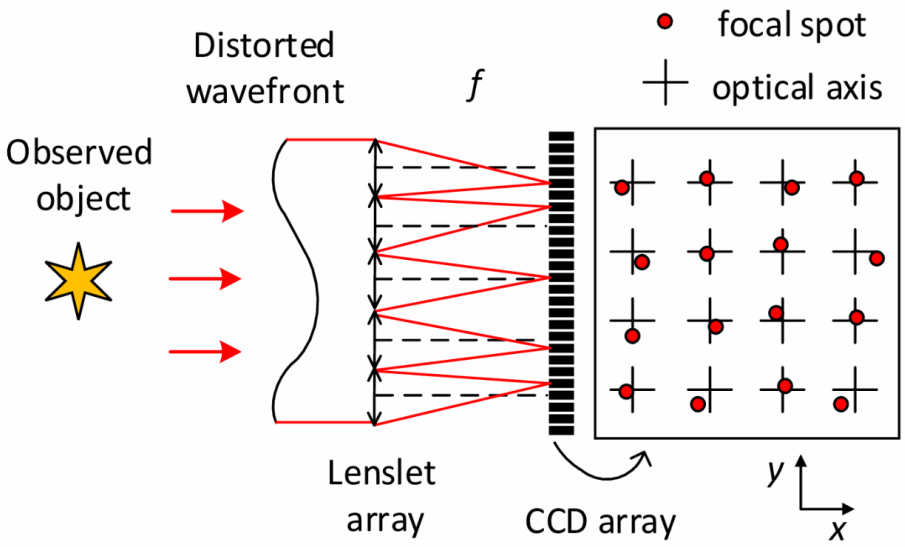


Figure 2.4: (Li and Li, 2018) The operation of a Shack-Hartmann WFS, showing the lenslet array and the subapertures on the detector. A deviation in the local wavefront slope causes a subapertures spot to move on the detector. The size of the subapertures projected on to the telescope pupil ensures that the main aberrations seen by a single subaperture are tip and tilt.

where $\mathbf{S}_{\text{CoG}} = (x_c, y_c)$ are the x and y slope values, X_{ij} are the pixel coordinates and I_{ij} are the pixel intensity values.

The results of the CoG then need to be shifted by half a pixel value to arrive at the final wavefront slope for each subaperture. Some post-processing of the SH-WFS slopes is often required once they have been computed. This involves subtracting a reference slope from each of the subaperture slopes to remove any systematic bias in the system causing an un-perturbed wavefront to give off-centre slope values. The reference slopes are calculated by using a calibration source to detect the neutral CoG of the subapertures when a flat wavefront is incident on the WFS.

A downside to using the CoG centroiding algorithm for the SH-WFS is that it becomes less effective for sensing on extended objects instead of the more typical point-source like guide stars. The CoG algorithms essentially determine the location around which the majority of the flux is located within each subaperture and for a point source this is assumed to be the centre of the PSF formed by the lenslet.

However for sensing on extended sources such as LGSs the PSF is no longer well defined, and so tracking the movement of the CoG of flux is no longer a reliable method for calculating the local wavefront slope. This is important, for example in solar AO where the WFS slopes are measured using patterns of extended structure on the solar surface itself. It's also important for LGS WFS, where the PSFs obtained on the WFS images are elongated due to the laser being launched off-axis from the subaperture lines of sight. If the elongation of the LGS spots is sufficient, the PSF will be truncated by the finite size of the subaperture reducing the effectiveness of the CoG algorithm.

Correlation wavefront sensing To overcome the limitation of CoG centroiding in solar AO, the technique of correlation wavefront sensing is used (Townson, 2016). This involves calculating a correlation image for each subaperture by cross-correlating a reference image with a subaperture image. The deviation of the peak intensity in the correlation image from the centre then indicates the shift between the reference and the sub-aperture images. There are many different techniques to perform the cross-correlation itself, one such technique is the square difference function which is simply calculated for each pixel, i, j, using ,

$$\sum_{x,y}(Im(x,y) - Ref(x+i, y+j))^2, \tag{2.3}$$

where Im represents the sub-aperture image and Ref represents a reference image. This corresponds to a least squares exploration of the possible alignments of the two images (Townson, 2016). Once the correlation image is formed, the CoG algorithm can be used to determine the shift of the images and therefore the local slope. The downsides to correlation wavefront sensing are the computational requirements of computing the cross-correlation, then computing the CoG and finally needing to create and update the reference images for each subaperture as the view of the extended object changes. Updating the reference slopes is then non-trivial, see Basden et al. (2014).

Matched filter An alternative technique that has been investigated for dealing with the elongation of the LGS spots, which is especially relevant for the large apertures of the ELTs, is matched filtering of the WFS subapertures. The matched filtering algorithm for WFS proposed by Gilles and Ellerbroek (2006) involves generating the matched filter on-sky by continual measurements and averaging of the LGS spots. Once the matched filter has been computed, the process of determining the slope is by computing the dot product of the match filter with the pixel values of each subaperture and then normalising by dividing by the total flux per subaperture,

$$\mathbf{S}_{\mathrm{MF}} = (x_c, y_c) = \frac{R \cdot I}{\sum I_{ij}}, \tag{2.4}$$

where $\mathbf{S}_{\mathrm{MF}} = (x_c, y_c)$ are the x and y slope values, R is the matched filter and I is the sub-aperture image. This is not any more computationally demanding than the CoG; however for ELT-scale LGS WFS the spot elongation changes significantly across the pupil and so ideally each subaperture needs a unique matched filter which needs continually updating as atmospheric conditions change. Matched filtering has been proposed for the TMT AO and a number of experimental demonstrations have been conducted to explore its efficacy for ELT-scale wavefront sensing (Basden et al., 2017).

Regardless of the slope computation method employed, one of the benefits of SH-WFS processing is that the slopes for each subaperture can be computed completely independently and therefore can be pipelined along with the pixel transfer. In practice this means that when using a WFS detector that can stream pixels row by row, once enough rows of pixels have arrived the processing of the first row of subapertures can begin. Figure 2.5 shows the pipelining of pixels for the SH-WFS. This allows much of the subaperture processing time to overlap with pixel transfer and therefore reduces the effect that RTC processing has on the overall AO loop latency.

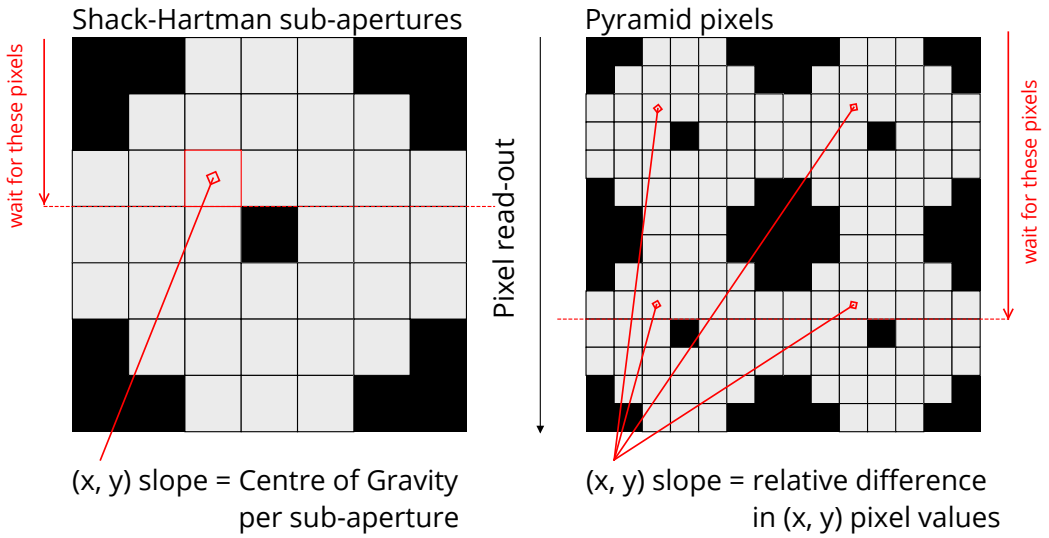


Figure 2.5: Comparison of the pixel pipelining for SH-WFS and Pyramid WFS image layouts. Each slope measurement from a SH-WFS is taken as the centre of gravity of the pixels values in each subaperture, slopes can be calculated as soon as all subaperture pixels have arrived. For the Pyramid WFS the slopes are calculated as the relative x and y differences between corresponding pixels from each pupil quadrant, no slopes can be computed until at least half the frame has been read. This difference makes pipelining of slopes much less effective when using a Pyramid WFS. Here we have assumed that the pixels are read from the top of the detector to the bottom, however the principle is the same for other read-out regimes.

2.1.3.2 Pyramid WFS Processing

Pyr-WFS slope computation differs from the more conventional SH-WFS, as described above, in the fundamental approach to detecting the local wavefront slopes across the telescope aperture, shown in Figure 2.6. The images received from a Pyr-WFS show the four split images of the telescope pupil in each quadrant which are made by focusing the PSF of the guide star onto the point of an optical pyramid or similarly the leading edge of an optical double knife-edge. The difference in intensity at each pixel location in each of the quadrants gives the local wavefront slope at that point in the pupil, these are calculated for each pixel position by,

$$\mathbf{S}_{\text{pyr}} = (x_c, y_c) = \left[\left(\frac{I_2 + I_4 - I_1 - I_3}{I_1 + I_2 + I_3 + I_4} \right), \left(\frac{I_1 + I_2 - I_3 - I_4}{I_1 + I_2 + I_3 + I_4} \right) \right], \tag{2.5}$$

where $\mathbf{S}_{\text{pyr}} = (x_c, y_c)$ are pyramid slopes and I_1, I_2, I_3 and I_4 are the pixel intensity

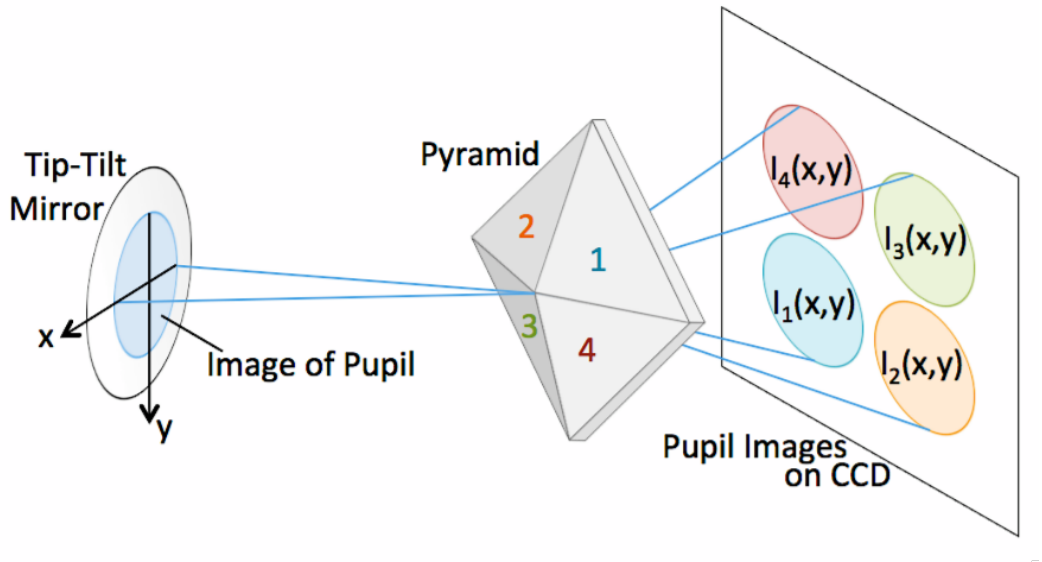


Figure 2.6: (van Kooten, 2016) The operation of a pyramid WFS, showing the optical pyramid and the four pupil images on the detector. A deviation in the local wavefront slope causes the intensity of complementary pixels to vary between the 4 quadrants. The size of the pixels projected on to the telescope pupil ensures that the main aberrations seen by a single set of complementary pixels are tip and tilt.

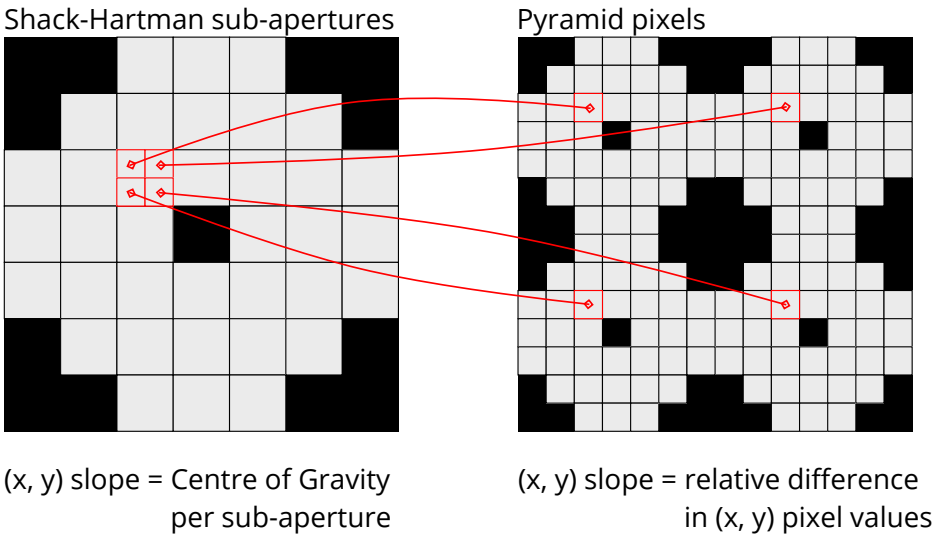


Figure 2.7: The organisation of wavefront data in the images from both a SH-WFS(left) and Pyr-WFS(right). The SH-WFS is made of a 7×7 array of subapertures, each of which is a quad-cell meaning that the subapertures are 2×2 pixels in size. The Pyr-WFS has the same dimensions as the SH-WFS and the arrows show the how wavefront data is distributed in both WFS types.

values in each of the four quadrants. This is similar to the slope calculation in a quad-cell SH-WFS where each subaperture is only 2×2 pixels. The relationship between distribution of wavefront data in a Pyr-WFS and a quad-cell SH-WFS is shown in Figure 2.7. Because the value of each equivalent pixel in the four quadrants is needed to calculate the slope values, the pipelining of the slope computation with pixel transfer isn't as efficient as that for the SH-WFS; this is shown in Figure 2.5.

The important information needed by the RTC for the slope calculation of the Pyr-WFS is the number of pixels per quadrant and the offset of the quadrant from the edges of the image. The slope computation then involves summing the pixel values in two adjacent quadrants and then subtracting the pixel values from the other two; this is done in both directions to get both the x and y slope values. These then need to be normalised by dividing by the total flux in the image. Whilst Pyr-WFS processing can benefit from the streaming of pixels so that the slope computation can be pipelined, this is not a efficient as with the SH-WFS as the slope computation cannot begin until at least half of the image has arrived; this is shown in Figure 2.5.

Another disadvantage to Pyr-WFS is that for a point source guide star the focused image needs to be rapidly moved on the tip of the pyramid to reduce non-linear effects due to diffraction; this is called modulation. This usually involves moving the spot in a circular motion around the tip of the pyramid such that the signal received during a single integration contains one or more of these movement cycles. With increasing modulation, drawing a larger circle around the tip of pyramid, the linearity of the Pyr-WFS increases, though the sensitivity decreases. With small modulation the sensitivity can be greater than that of SH-WFSs (Esposito and Riccardi, 2001). The reduction in sensitivity with more modulation also means that there is a reduction in sensitivity for wavefront sensing on extended sources, as the modulation essentially blurs the point source out during a single integration. Therefore currently no LGS Pyr-WFS are planned for the ELTs.

There are however benefits to using Pyr-WFS over SH-WFS. The first is the increase in sensitivity at low modulations as mentioned above. The lack of a lenslet array means that the subapertures are therefore defined by the detector pixels in each quadrant, so for fainter sources the number of subapertures can be reduced simply by binning the pixels to increase the signal. For extended sources such as LGSs, a Pyr-WFS doesn't suffer from truncation of the PSF which can occur due the finite size of SH-WFS subapertures. The reduction in sensitivity due to extended sources is also seen in SH-WFSs. The Pyr-WFS also requires fewer pixels for the same number of subapertures as compared to the SH-WFS which reduces the read noise and dark current effects as described in Section 2.1.2.

2.1.4 Wavefront Reconstruction

Once the slope values have been calculated from the WFS measurements, the next step in calculating the correction is to reconstruct the wavefront. In the case of closed-loop AO operation, this can be a fairly simple step which is achieved by mapping the slope values calculated from the WFS directly to actuator commands for the DM. The result of this is a wavefront described in actuator space, meaning that it gives the actuator commands that produce a DM configuration that matches the incident wavefront. In practice this transformation is achieved by a MVM operation between the slope vector and a control matrix which defines the mapping. A reconstruction MVM is given as,

$$\mathbf{y} = A\mathbf{x} \rightarrow y_i = \sum a_{ij} x_j, \tag{2.6}$$

where $\mathbf{x}$ is the wavefront slope vector of length N, A is the control matrix of dimensions $N \times M$, $\mathbf{y}$ is the resulting DM command vector of length M and x_i, a_{ij} and y_j are the elements of each respectively. The control matrix is constructed by calculating the inverse of the interaction matrix, the "poke matrix", which is formed by recording the response of WFS slope measurements when each of the

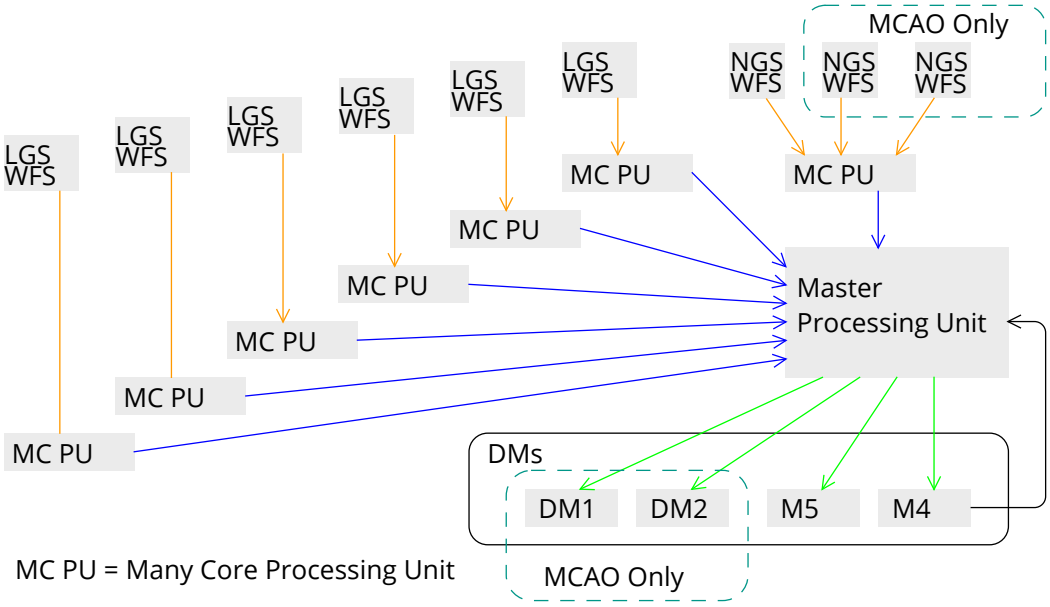


Figure 2.8: A multi processing node AO RTC architecture showing how for more complicated AO system such as MCAO and LTAO, the individual WFS can be processed seperately and the final result combined by a master processing node. This technique is further explained in Chapter 5.

DM actuators is actuated, or "poked", in turn. This is expanded upon further in Section 2.2.1.

For the more complicated closed-loop AO types such as GLAO, MCAO and LTAO, reconstructing the wavefront in this way can be very computationally demanding and so more efficient techniques have been proposed for ELT scale (e.g. Rosensteiner, 2012). However the majority of the processing for each WFS can be completed independently and the computational requirements for each are on the same scale as a SCAO systems' computational requirements. Therefore it is possible to process each WFS on a separate processing node before combing the results of the individual reconstructions. This method of processing is shown in Figure 2.8 and is further explained in Chapter 5. In the case of MCAO with multiple DMs the control matrix is constructed such that the wavefronts at the right altitudes are mapped to the correct set of DM actuators. More complicated reconstructors combine the wavefront reconstruction with a number of control steps such as with linear quadratic gaussian (LQG) control, which is explained in more detail in

Section 2.2.2.

Tomographic wavefront reconstruction In normal closed-loop operation of AO, the WFS measures the residual wavefront after the correction has already been applied to the wavefront and so the residual slopes themselves do not contain information about the atmospheric phase perturbations. However another requirement of tomographic reconstructors is the need for open-loop WFS measurements to acquire information about the atmospheric conditions. As described in Section 1.1.2.2, tomographic reconstruction involves considering information about the strength and altitude of the different turbulent layers that make up the Earth's atmosphere.

To retrieve information about the atmosphere from closed-loop slopes, a technique called pseudo open-loop control (POLC) can be used. This uses the previous DM command along with the residual slopes to calculate the pseudo open-loop (POL) slopes. The POL slopes are normally calculated by multiplying the previous DM command by the interaction matrix to get the open loop slopes that would have resulted in that DM shape, and then adding on the current residual slope values:

$$s_n^{\mathrm{POL}} = s_n^{\mathrm{RES}} + P \cdot a_{n-1}, \tag{2.7}$$

where s are the wavefront slopes (POL and residual respectively), P is the interaction matrix (which can be measured in a conventional way by poking the DM), and a are the actuator demands from the previous frame (n-1) (Basden et al., 2019). This specific method of calculating the POLC is an explicit calculation of the POL slopes, however POLC can also be achieved implicitly without actually calculating the POL slopes themselves. This implicit POLC technique is described in Basden et al. (2019). For an AO system with M slope values and N actuator values, where $M > N$ and usually $M \approx 2N$, using implicit POLC decreases computation time by reducing the number of operations from $N \times M$ for the explicit POLC to just N^2 as the POL slopes aren't explicitly required in the POLC calculation.

2.1.5 Applying the Correction

The last step in the AO RTC pipeline is to apply any control to the DM commands to condition them ready for sending to the DMs. This can involve simply applying a gain to the DM commands which scales them such that the specific control law is adhered to. The classic SCAO control uses the method of least-squares which is described in more detail in Section 2.2.1. As mentioned above there are also more complex control regimes that can be used to overcome some of the limitations of the classic minimum variance (MV) control such as predictive techniques to overcome the two-frame delay between measurement and correction and vibration filtering to mitigate non-atmospheric perturbations to the system.

Before sending the commands to the DM hardware it can be important, depending on the specific DMs used, to further condition the commands. This can involve scaling of the commands to appropriate values for the DM actuators, clipping of the commands to make sure they are within the required range and adding a bias to the commands to compensate for DM effects such as creep, hysteresis or non-linearities in actuator movements. A DM can interface with the RTC hardware via different methods (Basden et al., 2016) including, but not limited to, PCIe digital-to-analogue converter cards, which take the DM commands and transmit the correct voltages for the actuators directly to the hardware, as well as ethernet based devices and USB devices.

The ESO ELT's integrated deformable mirror, M4, will use an Ethernet interconnect (Chiozzi et al., 2018) and will perform further conditioning on the DM commands to ensure that they adhere to the constraints of the hardware (Vernet et al., 2014). This ultimately means that the actual shape taken by the DM may slightly vary from that which the RTC commands have computed. The RTC therefore requires feedback from M4 to ensure that the current DM shape can be taken into account in the tomographic calculations. Using the multiple processing node approach as shown in Figure 2.8 and the POLC method as described above, this

feedback only needs to be received by the master processing unit which combines the individual reconstructions and integrates the previous DM shape into the next command.

2.2 Wavefront Reconstruction Techniques

2.2.1 Classical MVM Control

Most AO systems work in a closed-loop configuration, that is the WFSs measure the residual wavefront phase errors which result from applying a correction to the incident turbulent wavefront phase. A closed-loop control AO feedback loop is shown in Figure 2.9, and shows the relationship between the DM actuator values u and the incident turbulent wavefront phase ϕ^{tur}. The ultimate aim of an AO RTC is to produce DM commands such that the wavefront errors due to ϕ^{tur} are fully corrected; in practice, however, this is not possible and so AO control techniques aim to minimise the wavefront error of ϕ^{res} via various means.

Due to the finite time steps between measurement and correction, the variables used in the wavefront reconstruction calculations are not continuous but in fact discrete and are defined as the value of the variable averaged over one time step, T. Using the variable k to define the current time-step gives the notation,

$$x_k = \frac{1}{T}\int_{(k-1)T}^{kT} x(t)dt, \tag{2.8}$$

where x_k is the variable x averaged over the time period $[(k-1)T, kT)$.

This discrete nature is illustrated in Figure 1.10, which shows an AO loop with a two-frame delay. During frame $[(k-2)T, (k-1)T)$ the residual wavefront is integrated by the WFS resulting in ϕ^{res}_{k-1} which is used to calculate y_k during $[(k-1)T, kT)$, which in turn is used to calculate u_k. The residual phase, ϕ^{res}_k, at

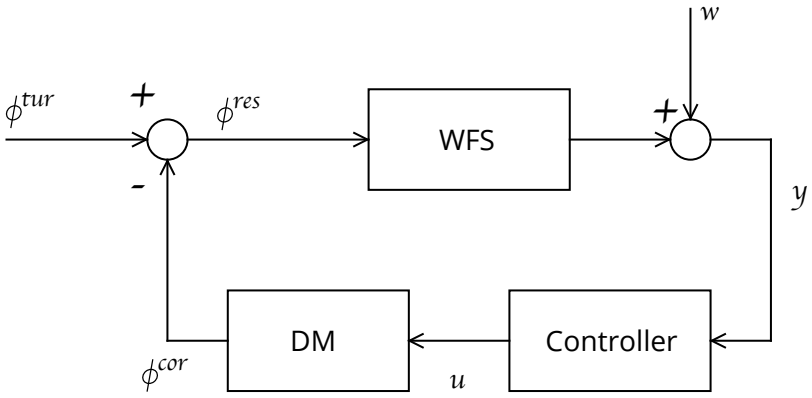


Figure 2.9: Block-diagram of an AO disturbance rejection feedback loop. The input turbulent phase, ϕ^{tur}, is corrected by subtracting the corrected phase, ϕ^{cor}, which gives the residual phase, ϕ^{res}. The WFS measures the residual phase giving slope measurements y which are integrated by the controller to produce actuator commands u for the DM that produces a new corrected phase to correct the turbulent phase for the next iteration.

time step k is the sum of the turbulent phase and the negative corrected phase at that time step, $\phi_k^{res} = \phi_k^{tur} - \phi_k^{cor}$.

2.2.1.1 Least-squares Reconstruction

The most basic form of AO closed-loop control is the least-squares MVM reconstructor which involves simply calculating the DM commands directly from the WFS slope measurements. In least-squares control the WFS measurements y for time step k are defined by,

$$y_k = G_y u_{k-1} + w_k, \tag{2.9}$$

where G_y is the DM-to-WFS influence matrix (poke matrix), u_{k-1} are the DM commands for the previous time step and w_k is an additive Gaussian measurement noise. The poke matrix gives a direct mapping of DM actuator commands to WFS slope values and can be computed by actuating each of the DM elements in turn

and recording the WFS response. This then gives an AO control law defined by,

$$u_k = u_{k-1} + (G_y)^{-1} y_k, \tag{2.10}$$

where $(G_y)^{-1}$ is the AO control matrix. This shows how classical AO control calculates the DM commands directly from WFS slope measurements as well as integrating the DM command from the previous time step.

In general the matrix G_y is not directly invertible as it is not normally a square matrix and may also be singular. This requires that the pseudo-inverse of the matrix be calculated using the method of least squares which minimises the merit function defined as,

$$\chi^2 = \sum_{i=1}^{M} \left[y_i - \sum_{k=1}^{N} a_k B_{ik} \right]^2 \tag{2.11}$$

e.g for matrix $[B]$ in $(y) = [B](a)$, the sum of the squares of the differences between the actual value of y and the estimated value of y from $[B]a$ which is equivalent to solution of a given by,

$$a = [B^t B]^{-1} [B^t] y, \tag{2.12}$$

where $B^t = \text{Transpose}(B)$ (Tyson, 2010). Combining this with Equation 2.10 gives the least squares control equation,

$$u_k = u_{k-1} + [(G_y)^t (G_y)]^{-1} [(G_y)^t] y_k. \tag{2.13}$$

where $[(G_y)^t (G_y)]^{-1} [(G_y)^t]$ is the least squares pseudo-inverse of $[G_y]$. This method of least squares breaks down when $[(G_y)^t (G_y)]$ is singular and is therefore not directly invertible, in which case the *singular value decomposition* (SVD) method can be used (Tyson, 2010).

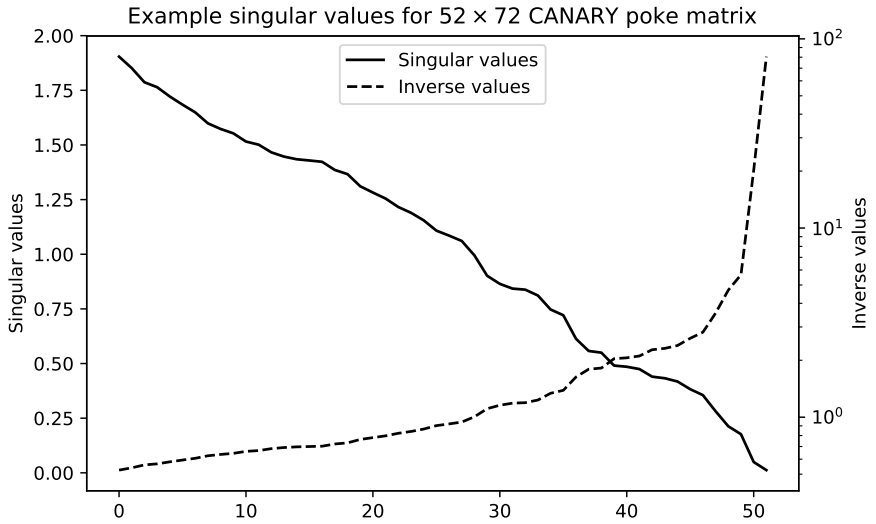


Figure 2.10: An example distribution of the singular values obtained via the SVD algorithm described in Section 2.2.1. These values are for a 52×72 poke matrix from the CANARY AO demonstrator (Basden, 2011).

The SVD method involves splitting up a matrix, such as an $M \times N$ matrix $[B]$, into the product of three matrices, $[B] = [U][\bar{W}][V^t]$. Here $[U]$ is also an $M \times N$ matrix, $[V]$ is a square $N \times N$ matrix and $[\bar{W}]$ is an $N \times N$ diagonal matrix given by,

$$[\bar{W}] = \begin{pmatrix} w_1 & & & 0 \\ & w_2 & & \\ & & \ddots & \\ 0 & & & w_N \end{pmatrix}, \tag{2.14}$$

The diagonal values w_i of $\bar{W}$ are called the singular values of B and are positive and real. The matrices U and V are orthogonal which gives the properties $UU^T = U^TU = 1$ and therefore $U^{-1} = U^T$. The columns of U and V form an orthonormal set and are called the left and right singular vectors of B respectively.

The elements of $[U]$, $[\bar{W}]$, and $[V]$ can be calculated from SVD computational algorithms. The inverse of $[B]$ is then simply given as $[B]^{-1} = [V][\bar{W}]^{-1}[U^t]$ where $[\bar{W}]^{-1}$ is also a diagonal matrix with elements that correspond to the inverse of the elements of $[\bar{W}]$, i.e $([\bar{W}]^{-1})_{ii} = ([\bar{W}]_{ii})^{-1}$. A representative example of singular

values for a 250×250 matrix and their inverse values are shown in Figure 2.10. Due to the orthogonality of U and V their inverses are trivial to calculate.

However if the matrix $[B]$ is singular then at least one of the values of $[\bar{W}]$ will be zero and so the corresponding value in $[\bar{W}]^{-1}$ will be a singularity. The SVD method allows the calculation of a solution that is *closest* to the least squares solution by replacing any singularities in $[\bar{W}]^{-1}$ with zero (Tyson, 2010). Combining the SVD method and Equation 2.13 gives us the SVD control equation,

$$u_k = u_{k-1} + [V][\bar{W}]^{-1}[U^t]y_k, \tag{2.15}$$

with $[V][\bar{W}]^{-1}[U^t] = [G_y]^{-1}$, where as before the elements of $[U]$, $[\bar{W}]$, and $[V]$ can be calculated from SVD computational algorithms (Tyson, 2010).

2.2.1.2 Minimum Variance Control

The WFS model described in Equation 2.9 is in general an oversimplification of the actual response of the WFS. Due to this the least-squares wavefront reconstruction algorithm described above performs poorly for some AO applications. The WFS measurements y are actually a function of the atmospheric turbulence and not of the pre-existing actuator commands that must be nullified to obtain a perfect wavefront. The DM actuator commands u required are ones that compensate for the turbulence induced wavefront error which is in general not the same as the solution to the best fit to the WFS measurements y. Minimum variance (MV) wavefront reconstruction can provide an optimal solution in the minimisation of the variance of the residual wavefront error when the statistics of the atmospheric turbulence profile and the WFS measurement noise are known (Ellerbroek, 2002).

From Figure 1.10 we can see that the controls, u_k, will provide the corrected phase

for the time period $[kT, (k+1)T)$ and so the corrected phase is given by,

$$\phi_{k+1}^{cor} = Nu_k, \tag{2.16}$$

where N is the mirror influence matrix which maps actuator commands to the corresponding wavefront phase and is defined by which ever basis the phase is represented in e.g. Zernike modes as described in Section 1.1.1.2. The residual wavefront ϕ_k^{res} that remains after the turbulent wavefront ϕ_k^{tur} been corrected by the DM actuator commands u_{k-1} is then given by,

$$\phi_k^{res} = \phi_k^{tur} - \phi_k^{cor} = Hx_k - Nu_{k-1}, \tag{2.17}$$

where N is the influence matrix as above, x is the vector of the turbulent phase values and H is the influence matrix that associates with the vector x. The vector x is similar to the DM command vector u, in that it describes the incident turbulent phase values at discrete points on the atmospheric phase screen. The effects of the two phase vectors, x and a, on the corrected phase ϕ^{cor} are assumed to be linear. This can be numerically evaluated by tracing rays through the atmospheric phase screens and the DM conjugate planes to compute the interaction matrices H and N, as described in Ellerbroek (2002). The vector x is a random variable with zero mean and finite second-order statistics, which are typically modelled using the Kolmogorov or von Kármán spectrum, see Section 1.1.1.1.

The WFS model given by Equation 2.9 can be now be replaced with,

$$y_k = Gx_{k-1} + w_k, \tag{2.18}$$

where G is the phase-to-WFS influence matrix and w_k now has finite second-order statistics. The elements of G can be numerically evaluated similarly to the elements of H and N by tracing rays from the guide star(s) through the turbulent

atmospheric phase screen to the WFS sub-apertures, as described in Ellerbroek (2002). This can also be expressed as,

$$y_k = D\phi_{k-1}^{res} + w_k, \tag{2.19}$$

where D is the WFS matrix that maps the incident wavefront phase to slope measurements. This gives the WFS measurements as a function of ϕ_k^{res} and not solely of the previous DM actuator commands u_k.

The minimum variance AO control law can then be defined similarly to Equation 2.10 by,

$$u_k = u_{k-1} + Ey_k, \tag{2.20}$$

where E is the wavefront reconstruction matrix. The minimum variance reconstructor E_* is then value of E that minimises the expected value of the mean square piston-removed wavefront error, denoted by σ^2, averaged over the statistics of the phase profile x and the WFS measurement noise w_k. σ^2 is related to ϕ^{tur} by

$$\sigma^2 = (\phi^{tur})^T W \phi^{tur}, \tag{2.21}$$

where W is a symmetric, positive-semidefinite matrix. The elements of W can be defined such that the value of σ^2 is equal to the mean square piston-removed value of a continuous phase profile obtained by interpolating a smooth function through the values of ϕ^{tur} specified on the discrete set of grid points x (Ellerbroek, 1994, 2002). This definition of E_* can be generalised as,

$$E_* = \text{argmin}_E \left\langle \sigma^2 + k||u||^2 \right\rangle, \tag{2.22}$$

where the angle brackets $\langle\cdot\rangle$ denote ensemble averaging over the statistics of noise and turbulence. The regularization term $k||u||^2 = ku^Tu$ must be included (with

a very small value of k) to avoid singularities if the subspace of DM actuator commands having no effect on σ^2 is not a priori known. A full derivation of the reconstructor can be found in Ellerbroek (2002).

2.2.2 Optimal LQG Control

As shown in Section 2.2.1 and Figure 1.10, classical AO control is limited due to the discrete time delay between measurement and correction. LQG control improves upon classical control methods by making a better prediction of the turbulent phase at the time-step that the correction is to be applied. Equation 2.16 shows how ϕ_{k+1}^{cor} which is applied at time $[kT, (k+1)T)$ is computed from u_k, which is itself computed from ϕ_{k-1}^{res} at time $[(k-1)T, kT)$, demonstrating the two-frame delay shown in Figure 1.10. LQG control instead calculates u_k from a prediction of the turbulent phase ϕ_{k+1}^{tur} at time $[kT, (k+1)T)$, i.e in the future with respect to u_k.

Starting from the fact that the correction applied by the DM should match the turbulent phase profile so that they cancel each other out and so minimise the residual phase, we can rewrite Equation 2.16 as,

$$u_k^{lq} = (N^t N)^{-1} N^t \phi_{k+1}^{tur}, \tag{2.23}$$

where u_k^{lq} is the optimal full information feed-forward control and $(N^t N)^{-1} N^t$ is the pseudo-inverse of the DM influence matrix, assuming with no loss of generality that $N^t N$ is invertible (Kulcsár et al., 2012). This shows that if we know ϕ_{k+1}^{tur}, i.e. in the full information regime, we can calculate the optimal commands to correct this incident turbulent phase profile. In reality however we don't generally know what the future turbulent phase will be and so we need to predict it based on past

measurements, giving rise to,

$$u_k^{opt} = (N^t N)^{-1} N^t \widehat{\phi}_{k+1|k}^{tur}, \tag{2.24}$$

where u_k^{opt} are optimum DM commands based on the prediction of the turbulent phase $\widehat{\phi}_{k+1|k}^{tur}$ and the $x_{k+1|k}$ notation represents the variable x at time-step $k+1$ given previous measurements at time-step k. This predicted turbulent phase is approximated in the simplest case by the vector valued auto-regressive model, AR(1), which models the turbulent phase as,

$$\phi_{k+1}^{tur} = A^{tur} \phi_k^{tur} + v_k, \tag{2.25}$$

where A^{tur} is a diagonal matrix which enables the adjustment of the cut-off frequency for each turbulent mode according to priors (Kulcsár et al., 2012) and is given in the Zernike basis by,

$$a_{ii} = \exp\left(-\frac{0.3(n(i)+1)VT}{D}\right), \tag{2.26}$$

where $n(i)$ is the radial order of the $i-th$ Zernike mode, V is the wind speed norm, T is the sampling period and D is the telescope diameter.

The AR(1) model shown in Equation 2.25 shows how the turbulent phase from one time frame depends on the turbulent phase from the previous frame, however it doesn't include any actual prior measurements. Applying a Kalman filter to the AR(1) model produces a prediction which then depends on the delay- and control-free measurements, $\mathscr{S}_k = \{s_0, ..., s_k\}$, with $s_k = y_{k+1} + DNu_{k-1}$, resulting in a prediction of the form,

$$\widehat{\phi}_{k+1|\mathscr{S}_k}^{tur} = A^{tur} \widehat{\phi}_{k|\mathscr{S}_{k-1}}^{tur} + L_k(s_k - D\widehat{\phi}_{k|\mathscr{S}_{k-1}}^{tur}), \tag{2.27}$$

where L_k is the Kalman gain (Kulcsár et al., 2012). Due to the stationary model used and because the estimate is to be used for infinite horizon LQG (Kulcsár et al., 2012), the time-varying Kalman gain L_k can be replaced, with no loss of optimality, by the steady state counterpart,

$$L_\infty = A^{tur}\Sigma_\infty D^t(D\Sigma_\infty D^t + \Sigma_w)^{-1}, \tag{2.28}$$

where Σ_w is the covariance matrix of the measurement noise in Equation 2.19 and Σ_∞ is the solution to the discrete algebraic Riccati equation (DARE),

$$\Sigma_\infty = A^{tur}\Sigma_\infty(A^{tur})^t + \Sigma_v - A^{tur}\Sigma_\infty D^t(D\Sigma_\infty D^t + \Sigma_w)^{-1}D\Sigma_\infty(A^{tur})^t, \tag{2.29}$$

where Σ_v is the covariance matrix of the noise term in Equation 2.25. A solution to Equation 2.29 can be found by using a DARE solver included in some mathematical programming languages, e.g MATLAB, PYTHON/SCIPY.

The next step is to combine Equations 2.16, 2.19, and $\phi^{res}_{k-1} = \phi^{tur}_{k-1} - \phi^{cor}_{k-1}$ to give,

$$y_k = D\phi^{tur}_{k-1} - DNu_{k-2} + w_k, \tag{2.30}$$

which can be substituted into Equation 2.27 via the delay- and control-free measurements $s_k = y_{k+1} + DNu_{k-1}$ resulting in,

$$\widehat{\phi}^{tur}_{k+1|k+1} = A^{tur}\widehat{\phi}^{tur}_{k|k} + L_\infty(y_{k+1} - D\widehat{\phi}^{tur}_{k|k} + DNu_{k-1}), \tag{2.31}$$

or equivalently,

$$\widehat{\phi}^{tur}_{k|k} = A^{tur}\widehat{\phi}^{tur}_{k-1|k-1} + L_\infty(y_k - D\widehat{\phi}^{tur}_{k-1|k-1} + DNu_{k-2}). \tag{2.32}$$

This result is then used to calculate the predicted turbulent phase at the next step, $\widehat{\phi}^{tur}_{k+1|k}$, by,

$$\widehat{\phi}^{tur}_{k+1|k} = A^{tur}\widehat{\phi}^{tur}_{k|k} \tag{2.33}$$

The equations for $\widehat{\phi}^{tur}_{k|k}$ and $\widehat{\phi}^{tur}_{k+1|k}$ are then simply the non-trivial part of the Kalman filter in predictor form adjusted to the state space model given by,

$$x_{k+1} = Ax_k + Bu_k + \xi_k \quad \text{and} \quad y_k = Cx_k + w_k, \tag{2.34}$$

where,

$$\begin{gathered} x_k = \begin{pmatrix} \phi^{tur}_k \\ \phi^{tur}_{k-1} \\ u_{k-1} \\ u_{k-2} \end{pmatrix}, \quad A = \begin{pmatrix} A^{tur} & 0 & 0 & 0 \\ I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ I \\ 0 \end{pmatrix}, \\ \xi_k = \begin{pmatrix} v^t_k & 0 & 0 & 0 \end{pmatrix}^t, \quad C = \begin{pmatrix} 0 & D & 0 & -DN \end{pmatrix}. \end{gathered} \tag{2.35}$$

And so the predicted turbulent phase is then given by,

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + L_\infty(y_k - \hat{y}_{k|k-1}), \tag{2.36}$$

where $\hat{y}_{k|k-1}$ is the best estimate of the model output given $\mathscr{S}_{k-1}$ and is given by,

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1}, \tag{2.37}$$

where L_∞ is as given in Equation 2.28.

The state space model demonstrated thus far in Equations 2.34, 2.35, and 2.36 is not however the only model that can be used to describe and then predict the state of the system. Smaller state vectors, x_k, can also be used (Kulcsár et al., 2012) and as the vector given in Equation 2.34 includes two occurrences of u and ϕ^{tur} it is a non-minimal state representation. The reason to use the non-minimal model

described is due to the fact that the control matrices A and B in Equation 2.35 do not depend on the DM parameter which makes the model structurally simple and the choice of state-vector also allows an adaptation to more complex models and configurations (Kulcsár et al., 2012).

2.2.3 Mitigation of Vibrations in AO

The LQG AO control described in Section 2.2.2 doesn't immediately account for telescope vibrations. However by inserting in the model additional states corresponding to spring-mass subsystems, this observer-based control can also filter out and/or compensate for telescope vibrations. This can be achieved by defining a global phase, ϕ_k^{glob}, which includes the turbulent phase and also a phase perturbation, ϕ_k^{vib}, due to vibrations,

$$\phi_k^{glob} = \phi_k^{tur} + \phi_k^{vib}. \tag{2.38}$$

This allows the additive vibrations to be straightforwardly included as perturbations in the state vector and estimated in the same way as the turbulence (Petit et al., 2008); the models then only need to be modified to explicitly describe the impact of vibrations on the phase. The vibrations can be modelled as a dampened oscillatory signal generated by a forcing function at the natural frequency of the vibrations to be compensated for, as described in Petit et al. (2008). This then leads to a a second order auto-regressive model, AR(2),

$$\phi_k^{vib} = a_1\phi_{k-1}^{vib} + a_2\phi_{n-2}^{vib} + \Xi_k, \tag{2.39}$$

where the coefficients a_1 and a_2 are given by,

$$a_1 = 2e^{-K\omega_0 T}\cos(\omega_0 T\sqrt{1-K^2}), \quad a_2 = -e^{-2K\omega_0 T}. \tag{2.40}$$

where K is the damping coefficient, $\omega_0 = 2\pi f_{vib}$ is the natural vibration frequency, T is the sampling period and Ξ_k is the, in general unknown, forcing function modelled simply as Gaussian white noise.

This vibration induced phase profile can then be included in the state space model with a modified state space vector given by,

$$x_k = \begin{pmatrix} \phi_k^{vib^t} & \phi_{k-1}^{vib\ t} & \phi_k^{tur^t} & \phi_{k-1}^{tur\ t} & u_{k-1}{}^t & u_{k-2}{}^t \end{pmatrix}^t, \tag{2.41}$$

and the measurement equation, Equation 2.19, now takes into account the global phase so that it becomes,

$$y_k = D\phi_{k-1}^{res} + w_k = D(\phi_{k-1}^{tur} - \phi_{k-1}^{cor}) + w_k \Rightarrow y_k = D(\phi_{k-1}^{glob} - \phi_{k-1}^{cor}) + w_k. \tag{2.42}$$

The state space model matrices, A, B, C etc, as shown in Equation 2.35 can easily be modified for this new vector and the estimation of the vibration and turbulence is still provided by an equation of the form of Equation 2.36. Correction is also performed similarly, by projecting both the turbulent phase and the phase perturbation caused by vibrations onto the DM (Petit et al., 2008).

The most popular alternative methods for vibration rejection are the H_2 and H_∞ frequency based approaches which have been shown to perform similarly to LQG (Guesalaga et al., 2013). It has been demonstrated by simulation (Kulcsár et al., 2006) that the LQG control described here, without vibration mitigation, has been able to increase the SR of a PSF to 71% from the 69% obtained using the classical techniques. Other laboratory simulations (Petit et al., 2008) have shown that LQG vibration mitigation has been able to increase an 81% SR, measured without vibration mitigation, up to 90% with an equivalent vibration free measurement of 91%. H_2 synthesis has been demonstrated (Guesalaga et al., 2013) to give reductions of up to 50% in the variance of residuals in off-line runs and also an improvement of around 30% in on-line runs, although the on-line results are inconclusive. LQG

control has also demonstrated benefits in wide-field AO control (Kulcsár et al., 2012) with test bench results giving LQG control an 81% SR for a an off-axis star with relative separation of 20% compared to the 34% SR obtained for the same star with classical AO control.

These results show the potential for LQG and H_2/H_∞ control to improve the resolution of seeing limited observations and also to allow for the characterisation and mitigation of telescope vibrations. Further research and study into these methods is currently being conducted with the aim of producing an AO control system suitable for Extremely Large Telescopes for which vibrations are expected to play a large role in the perturbations of the detected wavefront.

Chapter 3

Many-core CPU RTC and ELT-scale Optimisations

3.1 Current RTCs and their Suitabilty for ELT-scale

As described in Section 1.2 current AO RTCs are implemented on multiple hardware types and running different implementations of RTC software. One of the most widely used RTC systems is the European Southern Observatory (ESO) Standard Platform for Adaptive optics Real Time Applications (SPARTA, Fedrigo et al., 2006) platform, which is a set of tools and definitions which can be used to build AO RTC systems. The hardware defined by SPARTA uses FPGAs for the wavefront processing and DSPs for the reconstruction step. Section 1.2 discusses how both FPGAs and DSPs are generally more complex to work with than CPUs and therefore more time is required in the initial development, and it makes it difficult for further modifications to the RTC algorithms to be made. There is also the problem of the limited processing power per device with these technologies and so scaling such a system to the ELT-scale as described in Section 1.1.4.2 becomes very difficult.

Other current AO RTC implementations will use either GP-GPU technologies or standard CPU systems. However the data transfer from host to accelerator makes

the use of GP-GPUs infeasible for ELT-scale AO since the amount of data to be transferred increases massively compared to current AO systems. Standard CPU systems also suffer not only from their reduced computational performance but also from their very limited memory bandwidth compared to either GP-GPU technologies or many-core and multi-socket CPU solutions. Table 3.1 shows a comparison of the computational performance, memory bandwidth and number of processing nodes required for ELT-scale for the different hardware technologies available. The measured memory bandwidths shown in Table 3.1 were gathered using a modified version of the STREAM benchmark (McCalpin, 1995) which used NUMA aware memory allocations and `pthread` multi-threading. This was not as optimal as the standard STREAM benchmark when used on a non-NUMA system, however it does allow a direct comparison between the different hardware platforms.

3.2 Other ELT-scale Investigations

Due to the challenges involved in the processing of ELT-scale AO there are a number of other investigations ongoing and complete to discover a solution to the RTC processing problem.

GreenFlash was an European Union (EU) Horizon2020 funded project to investigate different HPC technologies for the facilitation of ELT-scale AO RTC processing. It concentrated on three main technologies, GP-GPU, many-core CPUs, and FPGAs to be used for the entire RTC pipeline. The Green Flash GP-GPU solution consisted of a hybrid FPGA-GP-GPU design where the FPGA receives WFS and transfers it directly to the GP-GPU using direct memory access. The GP-GPU also uses a persistent kernel, as described in Section 1.2.2.1, allowing it to continue the RTC processing without any host CPU intervention. These two techniques reduced the latency due to memory transfers from host to accelerators that are normally present in GP-GPU solutions and the persistent kernel allows all processing to occur on either the FPGA or GP-GPU, further reducing latency

Processor Type	Representative Example	Computational Performance (SP TFLOPS)	Memory Bandwidth (GBs^{-1}) T	M	Nodes Required (6×WFS)	Price per unit (USD)
GPU	NVIDIA V100	14.9	900	-	3	10,600
Xeon Phi	KNL 7250	5.2	480[1]	432	6	3,400
Xeon Phi	KNL 7210	4.5	450[1]	385	6	3,400
Intel CPU	Platinum 8180 ×4	22.9	512	362	6	40,000[2]
Intel CPU	Platinum 8180 ×2	11.5	256	182	12	20,000[3]
Intel CPU	Gold 5120 ×2	2.3	230	139	8	3,100[3]
AMD CPU	EPYC 7351 ×2	3.0	341	294	6	2,500[3]
FPGA	Intel Stratix 10 MX2100	6.3	512	-	6	~14,000[4]

Table 3.1: A comparison of computational performance, theoretical and measured memory bandwidth and nodes required for ELT-scale AO, for the different hardware types available for AO RTC. The T and M columns for memory bandwidth refer to theoretical and measured respectively. DSPs are not included as it is difficult to find specifications and in general their computational and memory bandwidth performance are far behind the other processor types.

[1]Measured using starboard STREAM, no theoretical available. [2]Price is for the four CPUs. [3]Price is for the two CPUs. [4]Price is for a development kit with $256GBs^{-1}$ memory bandwidth.

from host interruptions. The Green Flash many-core CPU solution was based on work conducted in this thesis and will be discussed in Chapter 4 and Chapter 5. The FPGA investigation proposed a cluster of many FPGAs running in parallel. However due to the complexity involved in developing both the software and hardware required, this was not considered a viable solution for ELT-scale AO RTC with the technology available.

Other more direct investigations are being conducted at other institutions with the aim of developing working RTCs for the proposed ELT instruments. This includes the NFIRAOS RTC for the TMT as well as RTCs proposed for the ESO ELT instruments, HARMONI, MAORY and METIS. The main technology being inves-

tigated for all of these efforts are many-core CPU systems. The ESO has investigated many-core CPUs for the next iteration of the SPARTA platform, SPARTA2 (Fedrigo and Donaldson, 2010) and so they are very much in favour of the use of this technology for the first light ESO ELT instruments.

3.3 Suitability of Many-core CPUs for AO RTC

3.3.1 Reducing Latency and Improving Jitter

For an MVM AO reconstructor the main computational burden lies with the reconstruction itself, transforming WFS slope measurements to DM commands, and its complexity scales linearly with the product of the total number of WFS subapertures and number of DM actuators. After centroiding of the WFS spots, the slope of each subaperture is described by two values, a displacement of the spots from centre along two perpendicular axis, commonly referred to as the x and y values of each slope. The poke matrix (PMX) is therefore a 2 dimensional matrix of size $(M \times N)$ where M corresponds to the total number of actuators and N to the total number of slope values. The AO control matrix has the same dimensions as the PMX and since the number of subapertures and actuators increases with the square of telescope diameter, the size of the control matrix increases with the fourth power of telescope diameter.

An MVM operation can be parallelised fairly straightforwardly as the computation can be split up into smaller parts all of which can be calculated concurrently with the results of each combined at the end to produce the final result. The most efficient way of calculating the multi-threaded MVM computationally is by computing the multiplication in a column-major fashion, as described in Figure 3.1. This allows the calculation of each part of the matrix to be associated with the corresponding part of the WFS slope measurements and not the entire vector, meaning that calculations can begin before all of the WFS sensor data has arrived.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix}$$

Figure 3.1: A comparison of the standard row-major (blue) and column-major (red) MVM calculation types. The row major method requires the full input vector to produce an entire single element of the output vector. The column major method only requires a subset of the input vector and produces a partial result for each of the output vector elements; once all input vector elements have been processed, the elements of the output vector will be complete. The column major method is used for the wavefront reconstruction MVM operation when the pixels are pipelined as not all elements of the input slope vector will be ready for processing at once.

The idea of splitting the incoming data into smaller parts, each of which can be operated on separately, is known as pipelining and is a fundamental function of an efficient AO RTC in all aspects of operation, globally reducing the dependence on memory bandwidth and meaning less time spent idle waiting for data.

Accelerating the AO RTC in off-load mode to an accelerator card such as a GPU or a Xeon Phi co-processor requires copying data to and from the accelerator for every WFS image. The pipelining of data reduces the impact of this process on the latency as only a small amount of slope data needs to be copied at a time before processing can commence and the results can be copied back whilst other parts are being processed. However there is no way to entirely eliminate the impact of the offload on the latency and it can also have an effect on the jitter of the frame times as the offload is an additional process in the loop adding complexity to the code and another source of uncertainty. One method of reducing offload latency being investigated is to copy the WFS data directly to the accelerator without having it go through the host CPU and then sending the resulting DM commands directly from the GPU, as shown in Figure 3.2. However this method further increases both the complexity of the software and the hardware by requiring a separate custom device to deliver the data to the accelerator as currently no commercial options

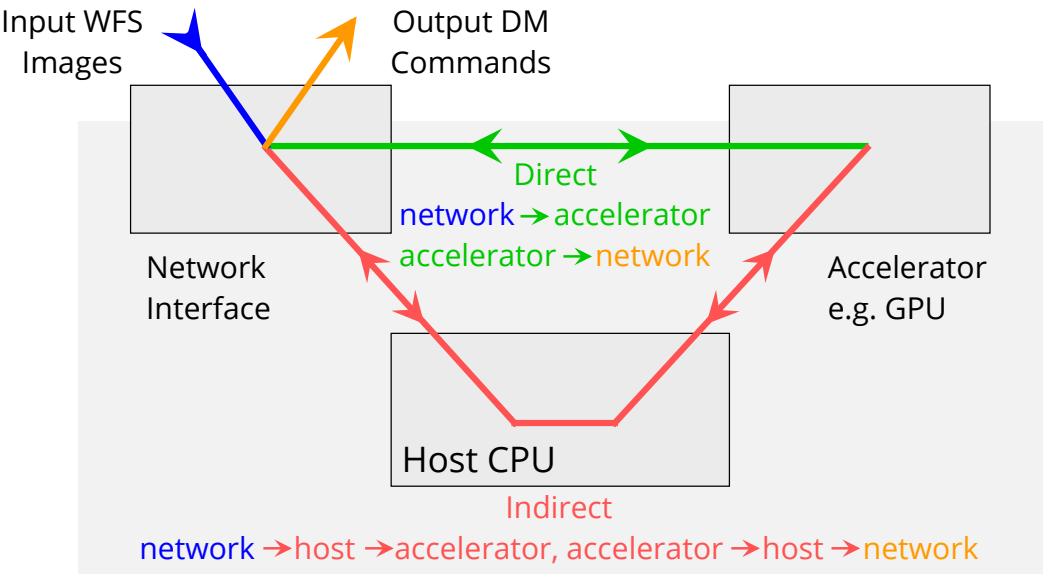


Figure 3.2: A comparison of the standard indirect data transfer through the host CPU and the more efficient direct to acclerator transfer sheme. A downside to the latter is that it currently involves non-portable software and the use of proprietary libraries.

exist.

The most straightforward way of reducing the latency and jitter introduced by offloading the data is to completely remove the accelerator card from the process and do all computations on the host CPU. For current 8-10m class telescopes this is easily achievable for a relatively simple AO system with 1 or 2 WFSs/DMs as the AO RTC can be wholly run on existing CPU hardware at the required latency. However for larger AO systems such as those needed for the next-generation ELT scale telescopes and those needed for more complex AO instruments using multiple WFSs and DMs, current CPU technology is unable to provide the computational performance required to reduce the latency and jitter of these systems to the required levels.

A large component of the time and effort required to design and produce an AO RTC stems from development of the control software. For technologies such as DSPs, FPGAs and GPUs, this can be extremely time consuming and require specific technological expertise, without any guarantee that the software will be in any

way compatible with future devices. CPU program development is comparatively more straightforward with a choice of well documented and easily accessible programming languages to choose from which are compatible with a wide range of CPU-based platforms.

3.4 Best case performance for ELT-scale SCAO RTC

In order to determine the best performance achievable with the Xeon Phi devices described in Chapter 1, a highly optimised algorithmic RTC was developed, i.e. a simple software solution which performs all necessary RTC algorithms using optimised library functions. However the software doesn't interface with a camera or DM hardware and so its operation is not pipelined and not user configurable. Therefore, although this RTC cannot be used in a real AO system, it gives some idea of the minimum frame computation time (or maximum frame rate) which can be achieved using given hardware. We note that an investigation using a full on-sky tested RTC is introduced in later sections.

The simple simulator uses the OpenMP API (OpenMP Architecture Review Board, 2015) for multi-threading, and performs pixel calibration on fake image data, centroiding of the calibrated pixels, an MVM reconstruction of the centroids and finally introduces a gain factor to the final result. The slope measurements are computed as if all pixels are available at once. This is the minimum computational requirement of an SCAO RTC and gives a base-line for best-case expected performance of the Xeon Phi. Figure 3.3 gives an overview of this system. Results of this best-case simulator are presented in Chapter 4.

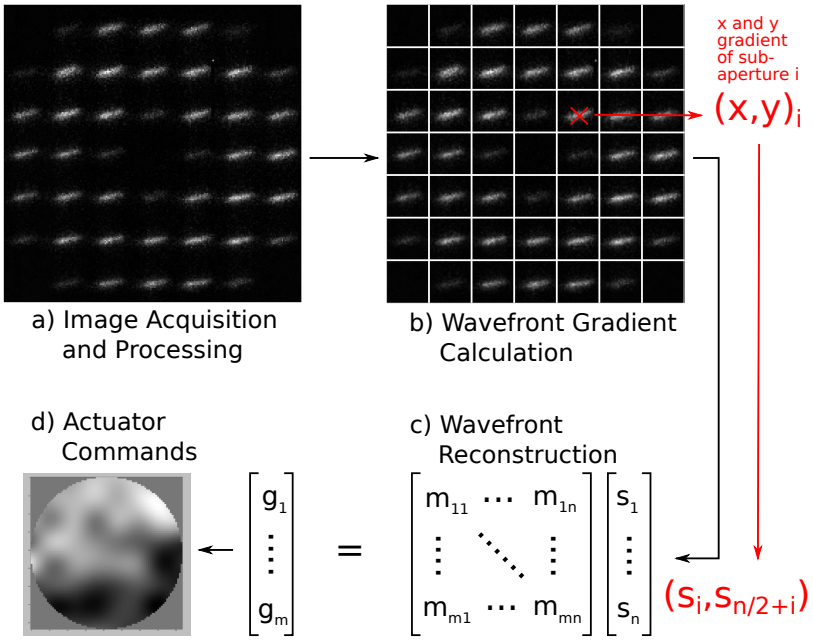


Figure 3.3: A figure showing the basic RTC operations, including a) image acquisition and processing (background subtraction, flat field application and threshold application), b) local wavefront gradient computation (using a centre of gravity algorithm), c) wavefront reconstruction (using a Matrix Vector Multiplication, MVM) and d) output of actuator commands. A thread will process a defined set of subapertures from beginning to end. For each subaperture, the local wavefront gradients are placed in a slope vector such that all the x gradients come first and then the y gradients ($(x,y)_i$ -> $(s_i, s_{n/2+i})$). The result of the MVM is a vector of actuator commands which can be reformatted to show the resulting shape of the correcting element.

3.5 The Durham Adaptive Optics Real Time Controller

This thesis presents the optimisation of a real on-sky tested RTC for the Xeon Phi KNL in the form of the The Durham Adaptive Optics Real Time Controller (DARC, Basden et al., 2010). DARC is a freely available and on-sky proven AO RTC software package written in the C and PYTHON programming languages. It is built upon a modular real-time core which allows it to be extended for many different AO RTC scenarios such as for different AO regimes like SCAO and MOAO and allows individual algorithms such as pixel calibration and wavefront reconstruction to be replaced or modified. The modular design also allows it to interface with many

different devices for wavefront sensing and wavefront correction (Basden et al., 2016), making it flexible enough to be used in almost any AO situation.

Because DARC is built on the C and PYTHON programming languages it can be compiled and run on many different systems including the socketed Xeon Phi, x86 (Intel & AMD), IBM POWER8 (Basden, 2015) and ARM processors. Within DARC, wavefront sensor images are processed in parallel, with subapertures being processed as soon as enough pixels have arrived at the computer. DARC uses a horizontal processing strategy (Basden et al., 2010) which allocates a similar workload to each thread, with threads being responsible for processing of a subaperture from start to finish (including calibration, slope calculation and partial reconstruction). This means that AO latency can be reduced, since by the time the last pixels arrive at the computer, the majority of the processing for that frame has already been completed. Here, we consider the optimisation of DARC for use with the Xeon Phi architecture, and report on performance investigations.

3.6 Optimisations for many-core operation

An x86 CPU the Xeon Phi shares many attributes with standard CPU hardware. However, it is also very different from previous CPUs with its many ($\geq$ 64) low power cores, and its high bandwidth memory due to the integrated MCDRAM and the 512 bit wide vector registers for improved SIMD performance. The SIMD processing paradigm is extremely important when considering large computational problems that can be easily vectorised. While most software developed for standard CPU systems can be compiled and run on Xeon Phi hardware with no alterations, to make the most of the new features, some optimisations are needed to best utilise the available hardware. These include:

1. thread synchronisation, to make efficient use of all cores
2. memory access, to optimise for the fast memory

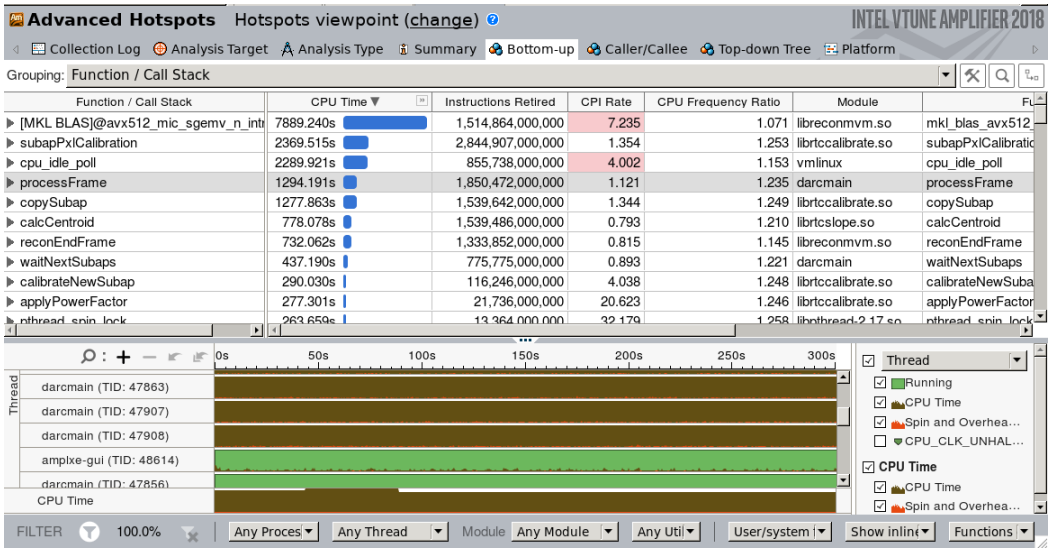


Figure 3.4: An example screenshot of the Intel VTUNE AMPLIFIER software and it shows the computation performance of the application. This shows that the AVX512_MIC_SGEMV function call is taking most of the processing time, which is used for the reconstruction MVM.

3. vectorisation, to take advantage of the wide vector registers.

3.6.1 Software Profiling

To analyse the utilisation of the hardware and therefore the efficiency of the programming, software profiling methods were used. The software profiling involves executing the RTC application at the same time that the profiling software is running and (depending on the type of software) it can access different information about the state and execution of the RTC software. The three main profiling tools used were Intel's VTUNE AMPLIFIER, PERF, and HTOP. Each was used depending on which information was required and how much impact the profiling should have on the application performance. Due to the way the software profilers work they can reduce the performance of the application being tested, and depending on how much information is gathered, the performance drop can be quite severe.

VTUNE AMPLIFIER was by far the most comprehensive profiling tool used. It works with the Intel compiler to show the impact of each function call in the application

and how much time the hardware spends executing each instruction. Because of the amount of data collected, the impact on performance of the VTUNE software on the RTC was very high, reducing the performance by as much as 40%. This drop in performance for real time software potentially means that the data it collects are not necessarily representative of how the RTC spends its time during normal operation. Therefore this tool was mainly used to discover any functions that seemed to be taking up more CPU time than expected and also to find any areas of code that were being inefficiently vectorised by the compiler.

The most straight forward way to use VTUNE AMPLIFIER involves starting the RTC software like normal and using the VTUNE GUI to attach to the running process by specifying the process ID. The software will begin to collect profiling data until stopped by the user or a pre-determined time limit has been reached. The VTUNE GUI shows a comprehensive breakdown of the time spent in each section of code. Figure 3.4 shows an example of the functionality of the VTUNE software.

The next step in the profiling involved using the Linux PERF tool, which, when used with the top argument, displays the percentage of time the system spends on different function and instruction calls. This tool is less intrusive than VTUNE and so the impact on the performance of the RTC is reduced. Because of this, it is a useful tool to use when debugging modifications made to the source code if the RTC is not performing as expected. The final tool used is the HTOP utility which is similar to the built-in Linux top utility, however it also displays a handy visualisation of the utilisation of each of the processor cores. This tool has very little impact on performance and therefore is used regularly to inspect the multi-threading of the RTC and to ensure that specific threads are using the resources as they should. Figure 3.5 shows an example of the functionality of the PERF and HTOP utilities.

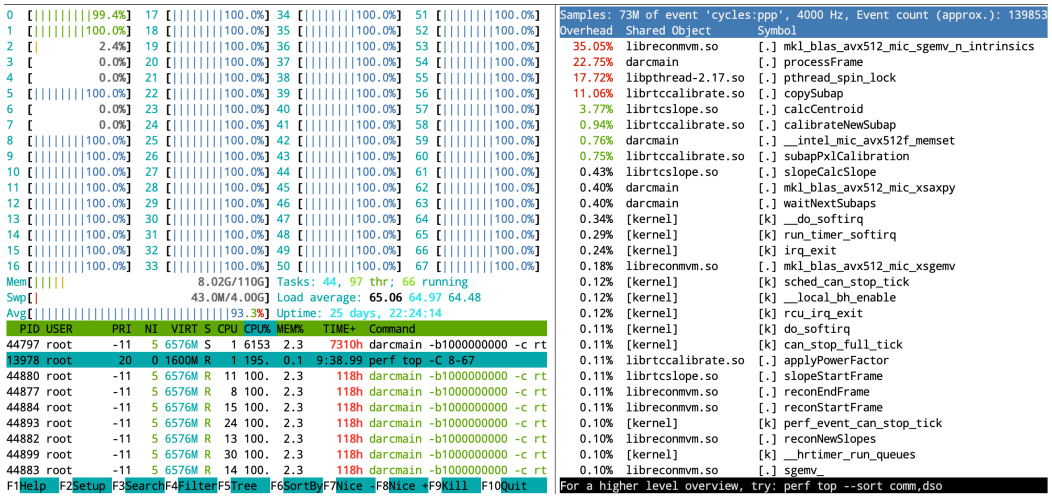


Figure 3.5: An example screenshot of the HTOP (left) and PERF (right) utilities and how they show a profile of the DARC RTC execution. This is for 60 reconstruction threads executing on cores 8-67 as seen in the HTOP utility. The PERF utility shows that the MKL_BLAS_AVX512_MIC_SGEMV function call is taking most of the processing time; this is used for the reconstruction MVM.

3.6.2 Multi-threading of Subaperture Processing

Multi-core CPU systems have become the norm in recent years leading to DARC being developed using a multi-threaded real-time core with the POSIX ('The-Open-Group', 2016) pthread library. The main method of ensuring thread synchronisation has been by the use of pthread mutexes and condition variables. A mutex is a mutual-exclusion variable which allows threads to 'lock' a certain section of code, preventing other threads from accessing these protected regions. If a thread calls the lock function on an unlocked mutex variable, then that thread will be allowed to acquire the mutex lock. Any other threads which attempt to lock this mutex will have to wait at the lock function until the mutex is unlocked.

Condition variables provide a powerful facility to synchronise different threads by using mutexes. If a thread has acquired a mutex lock it can call a condition wait function that is associated with that specific mutex variable. That thread will then proceed to release the mutex lock and then wait until the condition variable has been signalled; other threads can also wait on the same condition variable. To

release the waiting threads a condition variable can either be signalled or broadcast by a non-waiting thread. A signal will release a single waiting thread which will then reacquire the mutex before proceeding with execution. A broadcast will wake all threads that are waiting and one at a time the waiting threads will reacquire the mutex and be allowed to continue.

If multiple threads are waiting at a mutex then they will proceed one by one as the mutex is repeatedly locked and unlocked by the preceding thread. A thread waiting at a mutex will generally be descheduled by the operating system scheduler and put to sleep, reducing power consumption and freeing up the hardware for other threads to be scheduled. This works well for low order multi-core systems with 2-16 CPU cores, as it allows for more threads than physical CPU cores and the simultaneous descheduling and rescheduling of these few threads when they are waiting at the same mutex has little overhead.

However, for the Xeon Phi MIC architecture with ≥ 64 low power cores, DARC needs to be configured to execute a single thread per core with > 48 threads to achieve maximum performance for ELT-scale SCAO (Figure 3.6). This can cause problems when using mutexes and condition variables as the constant sleeping and waking of this large number of threads significantly increases latency. The solution that we have developed is to use a structure similar to mutexes called spinlocks, which also protect critical sections of parallel code but instead of sleeping and descheduling, threads simply wait until they can proceed. This waiting process constantly consumes CPU cycles but this increases the system's responsiveness. This helps to reduce the computation latency when using a large number of processing threads as each thread can resume operation without needing to wake from a sleep state.

Unfortunately, the condition variables described above do not work with spinlocks and so we replace these where possible by simple volatile flag variables, taking care to ensure that thread safety is maintained.

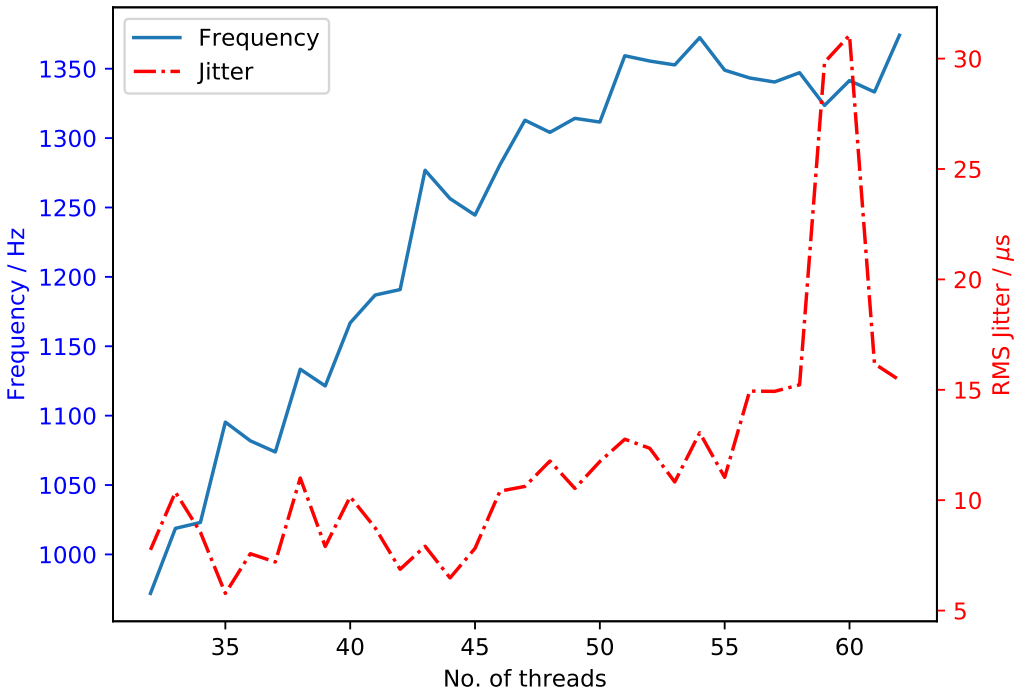


Figure 3.6: A measure of the average frequency of DARC running with different numbers of threads on Xeon Phi KNL, given in Hz. Also shown is the RMS jitter of the frame time data used given in µs. The increases seen in the frequency at threads counts of 35, 38, 43, 47, 51, 54 are most likely due the thread allocation used to aid in vectorisation described in Section 3.6.3.1.

3.6.2.1 Explicit Subaperture Thread Allocation

As described in Chapter 1 the latency of an RTC is defined as the time between the arrival of the last WFS pixel at the RTC to the time that the final DM command is delivered; reducing this interval is therefore essential to improving the performance of the RTC. The different options for handling and processing the pixel stream are shown in Figure 3.7, with each subsequent option reducing the RTC latency.

As each DARC thread processes its subapertures from beginning to end, they must be allocated a specific set of subapertures to process. The most simple and naive way of assigning subapertures would be to divide them equally amongst the threads as shown in Figure 3.7(c). However since each thread will spend an equal amount of time to complete its processing, the threads will complete processing of their

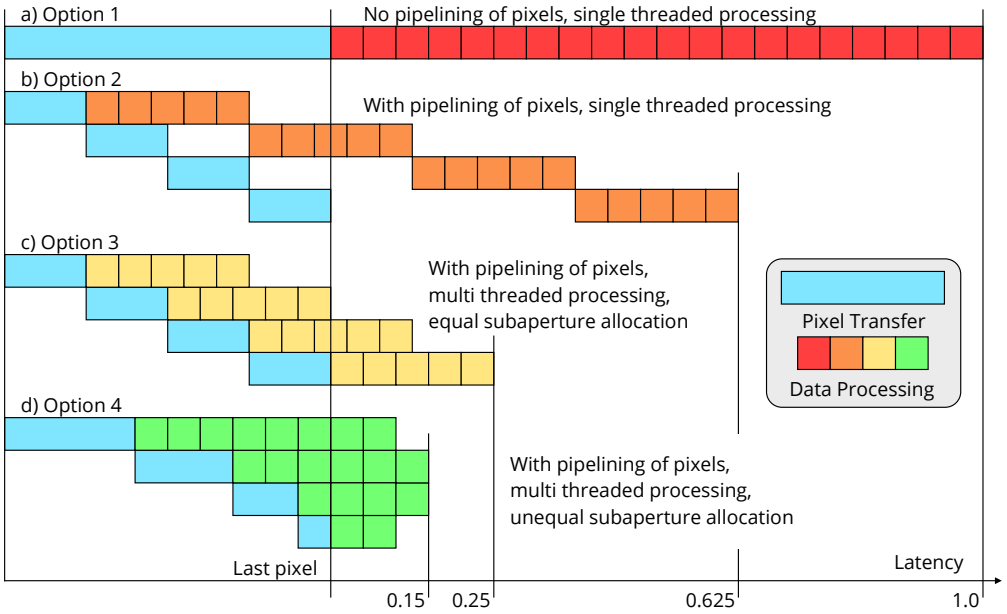


Figure 3.7: A comparison of the latency introduced via various pixel handling techniques. It shows that minimal latency is achieved via pipelining of the reconstruction using threads which process unequal numbers of subapertures such that they finish processing at roughly the same time.

allocated subapertures asynchronously. Threads that process the earlier arriving subapertures will finish ahead of the others, with the end time of each dictated by the pipe-lining of the arriving pixels.

Seen in Figure 3.7(d) is an option whereby the subapertures are not allocated equally among the threads. Rather the threads that process earlier subapertures are given more work to do and the later ones are given less. This ensures that the threads finish their work at roughly the same time and so helps to reduce the time between the last pixel arriving and the final DM command being sent out. However, the time waiting for pixels also changes as processing more subapertures requires waiting for more pixels, which can be seen in the different sized blocks for pixel waiting. This is a lot more complex in practice as there will not be a one-to-one relation between the number of subapertures and the number of pixels that need to be waited for.

3.6.2.2 Batch Processing of Subapertures

The method of allocating specific subapertures to specific threads, as described above, fixes the number of subapertures and the order that they are processed for each thread. With the correct tuning and optimisation this approach can reduce the RTC jitter as the local caching of information reduces the number of times that data needs to be transferred between each CPU core. However for pipelined WFS pixels the rate at which the data arrives for each row of pixels can vary for different WFSs and therefore the optimal number of subapertures to be processed by each thread will change. If this isn't taken into account then the latency will be worse.

A different approach is to instead split the subapertures into a number of batches and allow the threads to process a batch as soon as the batch and thread are ready. This means that each subaperture will be processed by different CPU cores each frame and so the jitter is likely to increase due to the caching of data no longer being efficient. However this does provide a benefit to overall RTC latency without any optimisation needed of the subaperture allocation, i.e. the OS scheduler is allowed to optimise the subaperture allocation at runtime. For pipelined pixels, the most simple and efficient way of splitting up the subapertures into batches is to put each row, or groups of rows, of subapertures into their own batch. This is because each row of pixels is read out as a whole and so the pixels for an entire row of subapertures will be available for processing almost simultaneously. This method of subaperture allocation should be more efficient at reducing latency, because as long as there are adequate processing resources, the subapertures will processed as soon as they arrive.

3.6.3 MVM Optimisations

3.6.3.1 Vectorisation

The 512 bit wide vector registers present on the Xeon Phi allow up to 16 single precision (SP) operations to be performed per cycle per CPU core. An operation in this case can be a fused-multiply add (FMA) operation which combines an addition and a multiplication, allowing up to 16 SP additions and 16 SP multiplications per instruction cycle. This is double the previous specification of 256 bit vector registers allowing a theoretical 2X speed up for vectorisable computations. Vectorisation is generally handled by the compiler: depending on the level of optimisation chosen at compile time, a certain amount of auto-vectorisation will occur. However, steps can be taken to aid the compiler and investigate where vectorisation occurs or does not occur. Essentially, if the compiler is able to detect that vector or matrix operations include 16-float boundaries at the same points, then these operations can be vectorised. This therefore usually means that by aligning memory to the nearest 64 bytes, vectorisation will be aided.

The allocation of subapertures to specific threads within DARC can be optimised such that each thread processes a multiple of 16 slope measurements when calculating its own section of the wavefront reconstruction MVM. As each subaperture has 2 slope measurements, x and y, we therefore ensure that the subapertures are allocated to threads such that each thread processes a multiple of 8 subapertures as a single chunk.

Alignment of array memory to page cache boundaries is important so that the data required for the vectorised instructions can be loaded into the registers efficiently and with the right ordering. This can be done when allocating memory for the arrays using the `posix_memalign` ('The-Open-Group', 2016) function call which aligns the amount of memory required at the specified boundary. The next step is to then ensure that sections which can be vectorised are written in such a way that

the compiler can apply auto-vectorisation; Intel provides a guide which details the necessary steps (Intel, 2012).

3.6.3.2 16-bit Floating Point Control Matrix

Because the wavefront reconstruction MVM is a memory bandwidth bound operation, due to the relatively simple mathematical operations but large data size, investigating ways to reduce the memory bandwidth dependence is an important consideration for ELT-scale AO RTC. A potential solution is to store the control matrix using 16 bit floating point format, rather than the conventional 32 bit format. The format used for the 16 bit floats is the IEEE 754 specification for binary16 (IEEE, 2008) which reduces the exponent from 8 to 5 bits and the mantissa from 23 to 10 bits.

This change does result in some loss of precision in the control matrix, however the available precision is still greater than that of the wavefront slope measurements (which are based on integer-valued detector measurements) and is therefore still considered sufficient for the reconstruction (Basden et al., 2010). Every AO loop iteration, this control matrix is then loaded into CPU registers, converted to 32 bit format for operations (necessary since the Xeon Phi cannot perform 16 bit floating point mathematical operations), and the DM vector computed. The reduction in memory bandwidth required can therefore reduce AO system latency.

3.6.4 Reduction of Partial DM Vectors

As each DARC thread processes a set of subapertures from beginning to end, the result of each thread's execution is a partial DM vector corresponding to those subapertures. To combine these results into a final DM command vector they must be all be summed together. Previously DARC has achieved this by using a mutex to lock the final DM command vector whilst each thread adds its vector into it in turn. This works well for small numbers of threads on fast cores. However

for the KNL case where there are more threads on comparatively slower cores, this serial addition is a processing bottleneck and a large source of extra latency.

A solution that I have developed is a branching algorithm which allows groups of threads to add their partial DM vectors and so each group can work in parallel. A group will be defined by a spin-lock and a thread-barrier. Within the group a thread will get the lock whilst it copies its partial DM vector into a temporary output array and once each thread has copied its partial vector it waits at the barrier for the other threads in the group to finish. At the completion of a group's work, one thread from each group will take ownership of the temporary output array and move on to the next group. This can be seen in Figure 3.8 where each group adds its partial DM vector into the red box before that moves down to the next group where the process is repeated until the final DM command vector results.

This can help reduce the computational latency of an ELT-scale SCAO system by up to 200 µs or 18 %. The benefit of the algorithm is reduced in the pipe-lined case where it can help to reduce the latency by up 20 µs or 5 %. This difference is likely due to the individual threads finishing their execution at slightly different times when they receive pipe-lined pixels. For an optimal unequal subaperture allocation as described in Section 3.6.2.1, however, this branching algorithm would reduce the waiting time for each thread.

3.7 Host Optimisation and Tuning

3.7.1 Tuning the OS, Kernel and BIOS for Low Latency RTC

The operating system (OS) installed on the Xeon Phi used in this thesis is CentOS Linux 7.3 (The-CentOS-Project, 2001). To obtain the best low latency and low jitter performance various changes have been made to the default settings of the BIOS, the operating system and the kernel. The main changes to the BIOS settings

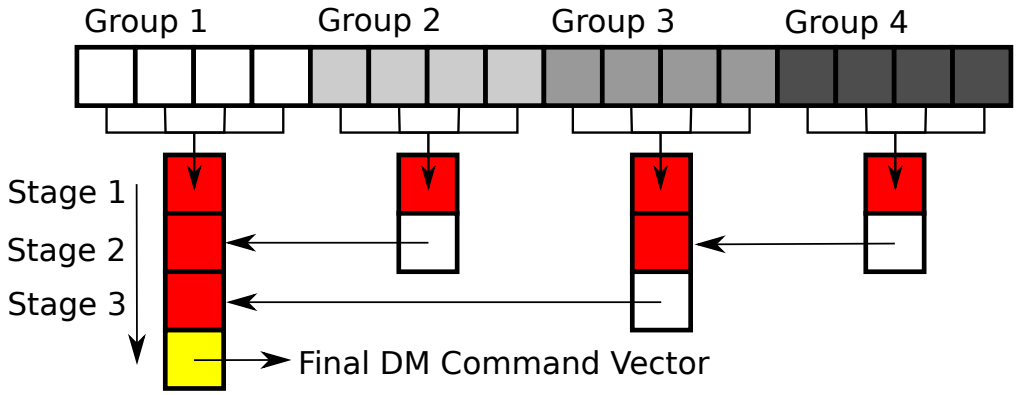


Figure 3.8: A schematic of the branching vector addition algorithm for a 4->2->2 situation with 16 threads, the first stage involves groups of four threads adding up their partial DM vectors, stages 2 and 3 reduce the resulting temporary DM commands to final DM command vector. This example allows up to 4 vector additions to happen in parallel and a total of 3 sequential stages instead of simply adding up all 16 threads' partial DM vectors sequentially. For larger thread counts, the effect is even more pronounced.

involve turning off Intel Hyper-threading, which allows more logical threads to execute concurrently on hardware cores. Removing Hyper-threading allows each software thread to be pinned to a single hardware core and removes scheduling inefficiencies caused when cores switch between different Hyper-Threads. During initial testing a Linux kernel with a real-time patch was considered. The real-time patch attempts to increase the kernel's real time response and allows the scheduler to pre-empt tasks to allow processes with higher priorities to proceed. However I discovered that with the tuning described above a real-time kernel was not required and in some cases degraded performance or even caused the system to crash.

Other BIOS settings include Xeon Phi specific settings which relate to how the CPU handles memory addressing, with information available online (Intel, 2015), and different modes which determine how the fast Multi-channel DRAM (MCDRAM) is allocated, either accessible like standard RAM, reserved for the OS as a large last level cache (LLC), or a mixture of the two; these modes are termed 'flat', 'cache', and 'hybrid' respectively.

OS and kernel setup refers to options such as isolating certain CPU cores so that

the OS doesn't schedule any program to run on these cores without specific instruction, and also to other options relating to CPU interrupts and different power and performance modes. The main kernel options used are:

- `isolcpus=[corelist]` - specify isolated CPU cores
- `nohz_full=[corelist]` - stop certain CPU core ticks whenever possible
- `idle=poll` - improve the performance of waking up idle cores
- `irqaffinity=[corelist]` - specify cores that handle interrupt requests (IRQs)
- `nohalt` - turns off some power saving functions

The `isolcpus` option isolates all but the first 2 CPU cores from the OS scheduler such that processes must be explicitly allocated to them. This prevents the OS from potentially interrupting the simulator processes. The `nohz_full` option sets the specified CPUs whose tick will be stopped whenever possible, which can reduce the number of scheduling-clock interrupts and reduce jitter. The `irqaffinity` options set the specified CPUs to handle interrupt requests (IRQs). This can reduce jitter by allowing the necessary interrupts to be processed on the correct CPU cores. The `nohalt` option tells the kernel not to use certain power saving functions which reduces interrupt wake-up latency and can improve performance for real-time systems. Finally the `idle=poll` option forces a polling idle loop that can slightly improve the performance of waking an idle CPU at the expense of power consumption. A comprehensive description of the kernel command line parameters can be found at The Linux Kernel (2019).

During our testing, we have identified that best performance is achieved with the CPU set to Quadrant memory addressing mode, and the MCDRAM was set to 'flat' mode. In 'flat' mode, the MCDRAM is visible to the CPU on a separate NUMA (Section 1.2.1.2) node from the standard RAM and so this must be addressed either by explicitly allocating the memory in the program (using a NUMA library), or

by executing the program on the specific NUMA node to make use of the fast MCDRAM. In this report the MCDRAM was allocated by running software with the `numactl` command with the `-membind=nodes` option, ensuring that the entire RTC is allocated on this NUMA node. On the Xeon Phi, the MCDRAM is 16 GB in size, which is sufficient to fit a whole ELT-scale RTC.

3.7.2 Compiler Tuning

There are multiple compilers available for compiling software written in the C programming language to target x86 hardware. During initial testing, two compilers were considered to achieve the best performance of the AO RTC. These were the Intel C compiler, `icc`, and the GNU's Not Unix (GNU) C compiler, `gcc`. By far the main benefit to using `gcc` is that it is the default Linux compiler and is therefore widely available, it is also completely free to use and modify under the GNU General Public License (GPL). It is being constantly updated to incorporate new features such as the Intel AVX-512 instruction set. Intel's `icc` is not open source and not free to use, being available only as part of a paid license subscription to the Intel Parallel Studio XE or Intel System Studio packages. Intel do however offer a free version of these packages to students and classroom educators which was used to compile software used for this thesis.

For optimal compilation with either `icc` or `gcc`, certain compiler flags were necessary to achieve best performance on the Intel Xeon Phi. The `-O3` compiler flag (GNU; Intel, 2017b) was used with both compilers as it enables the most aggressive automatic compiler optimisations including vectorisation, inlining of function calls and optimising loop structures. The `gcc` specific flags used were

- `-mavx512f -mavx512er -mavx512cd -mavx512pf` - enable AVX-512
- `-march=knl` - optimise for the Xeon Phi KNL
- `-mfma` - ensure fused-multiply add (FMA) operations are used

- `-finline-functions` - attempt to inline functions
- options to statically link the Intel Math Kernel Library (MKL)
  - The options to enable MKL for `gcc` have been omitted for brevity

To enable AVX-512, `gcc` needed to be of version 7 or above and so version 7.3.1 was installed manually, as the default CentOS `gcc` version is only 4.8.5. The Intel MKL is used to accelerate common basic linear algebra subroutines (BLAS) such as MVMs using pre-compiled optimised libraries.

The `icc` specific compiler flags used were

- `-static-intel` - link intel libraries statically
- `-xMIC-AVX512` optimise for the many integrated core (MIC) architecture
- `-fma` - ensure fused-multiply add (FMA) operations are used
- `-align` - attempt to align memory allocations to natural boundaries
- `-mkl=sequential` - dynamically link the Intel math kernel library (MKL)

The shared Intel libraries were statically linked to avoid having to install the compiler software package on every target system; this was needed as the free student license had a limit to the number of machines it could be installed on simultaneously.

It was found that `icc` provided the best performance for the AO RTC. This is as expected due to the number of optimisations available for the Intel platform. However the performance difference was only of order 10-15 % and so depending on the dimensions of the AO system, it may be more beneficial to use the free and open GNU `gcc`. Intel's `icc` was used for the compilation of all software used in this thesis unless otherwise stated.

3.8 CPU-based Network Camera Simulator

Results presented in this thesis were obtained from the DARC software using either a real camera for pipelined pixels or a CPU camera simulator machine. The early camera simulator results used the Aravis GigE Vision Library (AravisProject, 2018) to both transmit and receive the pixels. However this library is not optimised for large frame and high rate camera simulations and so instead a camera simulator based on the proposed standard for ESO ELT WFS (Downing et al., 2018) that streams pixels using UDP packets was developed. The UDP camera simulator is controlled at runtime of the simulator executable and the receiver software simply waits to receive the packets. In this way we can define the parameters of the camera simulator separately from the receiving of the camera stream, and it no longer requires a heartbeat thread to keep the camera sending packets, which we found could interfere with the pixel stream.

The camera simulator is implemented in the C programming language in an effort to make it both low level and as easy to modify and develop as possible. The underlying networking uses packet sockets, which are used to receive or send raw packets at the device driver (OSI Layer 2) level (Kerrisk, 2018). This allows minimal overhead from the kernel when sending and receiving packets as most of the protocol implementation can be programmed in user space on top of the physical layer. The type of socket used here is SOCK_DGRAM which does not have the link level header removed by the network stack and so is not quite as low level as SOCK_RAW packets.

The operating system on the simulator machine is Ubuntu 16.04 on top of a Linux 4.4.0 generic kernel. A low-latency Linux kernel was investigated but was observed to provide no discernible performance benefit. Some OS, Kernel and network level tuning was performed to improve performance. Some of the steps taken involved:

- isolating most CPU cores from the OS scheduler,

- specifying the core affinity for the NIC interrupts and camera threads,
- tuning both the NICs and linux network stacks UDP buffer sizes using the linux `ethtool` command and the `sysctl` utility,
- and setting the CPU power settings to "performance".

The camera simulator software is used only to simulate the pipelined transfer of pixels to the RTC and so the unique images that were streamed by each camera were created ahead of time via AO simulations to properly construct images for the type and dimensions of AO system tested. The images were stored on a PCIe fast raid storage array consisting of 4 Samsung 960 EVO NVMe solid state drives (SSDs) which provided transfer rates $> 4.2\text{GBs}^{-1}$ which is sufficient to allow the pixel data to be streamed directly from storage. The simulator software is set up such that both the inter-packet delay and inter-frame delay can be set independently. This not only allows different camera frame rates to be simulated, it also allows the readout time to be adjusted to better reflect that of a real camera, rather than just sending out the packets as soon as possible. As the inter-packet delay needs to have microsecond precision the timing is achieved via the Linux `timer_fd` utility which uses file descriptors to achieve a repeatable high precision timer.

The simulator hardware consists of a 2012 Intel Xeon E5-2650 dual socket system with 8 CPU cores and 32GB of DDR3-1600MHz RAM per socket with a base CPU frequency of 2.0GHz. The network devices used are 2 PCIe Intel Ethernet X710-DA4 Network Interface Controllers (NICs) with 4 10GbE ports each where a single camera simulator stream will have exclusive use of one of these 8 interfaces. In a multi-socket CPU system, certain PCI-e lanes are physically connected to a single CPU socket. The NICs were therefore installed in the host such that each was local to a different CPU socket and then the camera threads for each interface were assigned CPU cores on the local socket.

Some BIOS settings were tuned for the simulator, Intel HyperThreading was turned off to allow a single processing thread exclusive use of a CPU hardware core and

the power settings were tuned to performance settings. Linux kernel settings, as described above, were tuned such that the relevant command line options were `isolcpus=2-15 nohz_full=2-15 nohalt idle=poll`.

3.8.1 UDP Camera DARC Module

To interface with the UDP camera simulator described above it was necessary to implement a camera module for DARC to receive and process the image packets. This is implemented in DARC with a worker thread that performs the data transfer: it listens on a network socket, receives the packets, checks that they originate from the camera simulator, copies the data into a buffer and increments the received pixel count. The subaperture processing threads meanwhile wait for the correct number of pixels required to begin their portion of the processing, once enough pixels have arrived they are copied into the DARC raw pixel buffer for further processing. Depending on the data type of the pixels, usually either 8 bit or 16 bit, and the order that pixels are read out from the detector (see Section2.1.1) the data might need to be converted and/or reordered when being copied to the raw pixel buffer.

Chapter 4

SCAO Demonstrator: Single Node SCAO

In this chapter I present results of the investigation into optimising the DARC RTC software for many-core CPU operation as described in Chapter 3. First is presented the results of the best case simulator as described in Section 3.4. This is followed by an investigation of storing the AO RTC control matrix as 16 bit floating point values for the purpose of reducing the memory bandwidth requirement. Results of the DARC RTC software interfacing with a real camera and the simulated camera as described in Section 3.8 are presented. Finally results are presented for the batch subaperture allocation described in Section 3.6.2.2, the implicit POLC for the SCAO case as described in Section 2.1.4, and for a long time period operation of an ELT-scale SCAO configuration. Results and discussion in this chapter have been previously published in Jenkins et al. (2018b) and Jenkins et al. (2019).

The performance of an AO RTC is generally defined by the time taken for it to process each frame, where a frame is a single WFS image, and the processing includes calibrating the pixels, computing the centroids and reconstructing the wavefront before sending the results to a correcting element. There are different ways of defining the amount of time that it takes an RTC to process a frame and it depends on the definition of when a frame starts and when it ends.

Timing for the entire AO RTC loop is generally taken from the time the first pixels arrive at the RTC hardware to the time when the last DM commands have been sent to the correcting element. This encompasses the entire computation of the RTC especially when the main loop is pipelined, i.e., the processing is done for groups of pixels as they arrive due to the way the image sensors read-out the pixels. However, in the case of a fast RTC and a slower camera, RTC latency will be artificially lengthened by periods waiting for camera pixels to arrive. Therefore here, we use the traditional definition of RTC latency, defined by the time taken between last camera pixels arriving at the RTC and the last DM demand being computed. This definition can therefore be computed entirely within the RTC hardware, though does not include delays due to the capture of camera pixels (e.g. by a frame grabber card, and transfer to the computer memory), or delays due to time taken for DM demands to leave the computer and arrive at the DM. In addition to latency, we also report the maximum stable RTC loop rate, i.e. the fastest rate at which the RTC can operate stably.

In cases where we present the maximum RTC rate, i.e. without a camera attached, we define latency as the inverse frame time: in this case, the latency represents the minimum computation time for the RTC loop. The jitter of the AO RTC is defined to be the variation in latency which is presented as both rms jitter and also peak-to-peak (worst case) jitter.

4.1 The Best Case Simulator on Xeon Phi

We configure the best case SCAO RTC simulator as described in Chapter 3 in an ELT configuration with 80×80 subapertures, a $0.25 \times D$ central obscuration and 10×10 pixels per subaperture. This results in 4708 active subapertures, and therefore 9416 slope measurements. The number of DM actuators is 5170, based on a circular 81×81 actuator DM aperture. This system was tested using an Intel Xeon Phi 7250 system as described in Table 3.1.

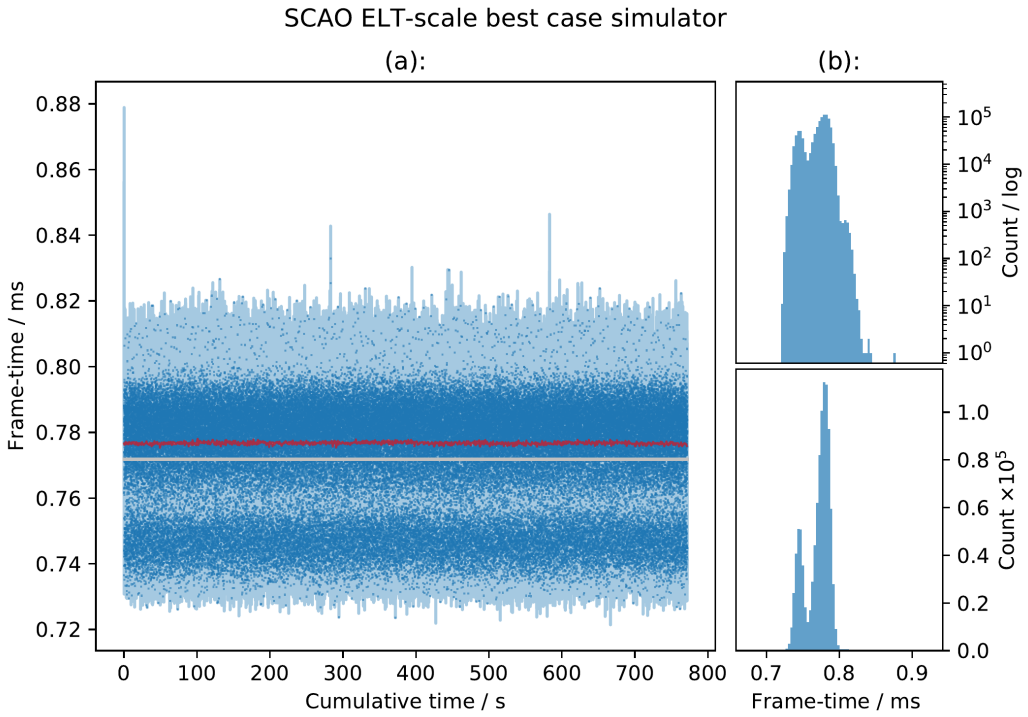


Figure 4.1: a) Frame time results of SCAO best case simulator for 10^6 frames with an average frame time of $770 \pm 20\,\mu s$ which corresponds to an average frame rate of $1300 \pm 30\,Hz$. The horizontal silver line is the average frame time, the red line shows a running median for every 1000 frame times. b) A histogram of the frame times in (a).

Figure 4.1 shows the frame time results of the SCAO best case simulator for 10^6 frames. The figure shows the minimal number of outliers and also the small spread of the distribution. The average frame time of $770 \pm 20\,\mu s$ corresponds to an average frame rate of $1300 \pm 30\,Hz$. This is shorter than a typical atmospheric coherence time, and therefore would be suitable for an ELT-scale SCAO RTC. The rms jitter is $16.3\,\mu s$, which is about $2\,\%$ of the mean frame time, and would have insignificant impact on AO performance (Pettazzi et al., 2012). The maximum instantaneous peak-to-peak jitter between consecutive frames is $107\,\mu s$, including the startup measurements, or $88.8\,\mu s$ during the long-term measurements.

4.2 DARC on Xeon Phi for ELT scale AO RTC

Figure 4.2 shows the frame time results of DARC configured for ELT-scale SCAO (in a similar configuration as above however with 5316 actuators, to mimic the ESO ELTs M4 Adaptive mirror, and 4632 subapertures for a total of 9264 slope measurements) though without a physical camera connected, measured over 10^6 frames. It can be seen in Figure 4.2(a) that for the system using an Intel motherboard (model S72000, 7250 processor), there are a small number of regular single frame outliers which add about 200-250 µs to the frame time, roughly every 63.75 s. We have determined that these events are due to the Intel system management interface on the motherboard, which periodically polls the processor for information. There appears to be no way in which this can be turned off. The presence of these interrupts can be verified using this code:

1. for SEC in ‘seq 0 200‘; do echo -n "$SEC "; rdmsr -p 0 -d 0x34; sleep 1; done,

which has been used to confirm their presence on the Intel S72000 motherboard used in this report.

Figure 4.2(b) shows results taken using a Ninja Development platform Xeon Phi using a Supermicro motherboard (model K1SPE with 7210 processor). Here it can be seen that these 64 s period events are not present. It is therefore important to take care when evaluating motherboards suitable for AO RTC. Histograms of both measurements are shown in Figure 4.2(c), the difference in mean frame time between the two distributions is due to the specification of the processors used in each motherboard; 1.4 vs. 1.3 GHz clock speed, 480 vs. 450 GB s^{-1} memory bandwidth (Table 1.3). From this figure, it can be seen that the distribution of latency measurements is approximately Gaussian, except for the outliers.

Therefore, DARC is able to operate ELT-scale SCAO with a 930 ± 10 µs frame time, corresponding to a 1070 ± 10 Hz maximum frame rate. When the 64 s events

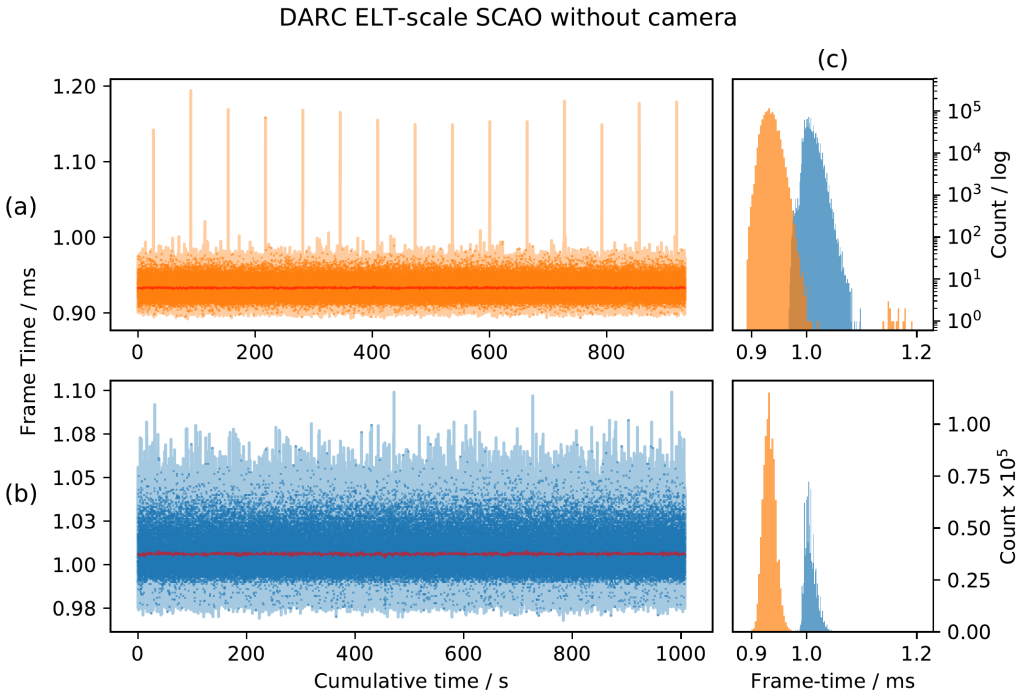


Figure 4.2: a) Frame times for DARC SCAO on an Intel Motherboard with no camera for 10^6 frames with an average frame time of 930 ± 10 µs which corresponds to an average frame rate of 1070 ± 10 Hz. b) As for (a), except for a Supermicro motherboard with an average frame time of 1010 ± 10 µs which corresponds to an average frame rate of 990 ± 10 Hz. c) Histograms of the frame times are presented in (a) and (b), for both a log scale (top) and a non-log scale (bottom). The horizontal silver lines in (a) and (b) are the average frame times for each distribution, the red lines show a running median for every 1000 frame times.

are included, the instantaneous peak-to-peak jitter over a million frames is 263 µs, while ignoring these events reduces the peak-to-peak jitter to 92.7 µs and the RMS jitter is only 11.4 µs. The Ninja development platform can operate ELT-scale SCAO with a 1010 ± 10 µs frame time, corresponding to a 990 ± 10 Hz maximum frame rate. This is a lower maximum performance than the Intel motherboard system due to the difference in processor specification, which is as expected.

4.3 Storing the control matrix as 16 bit floating point values

Being able to store the control matrix as 16 bit floats could potentially reduce the memory bandwidth requirement of the RTC by a factor of two, compared with storing it as standard single precision floats. Most CPU systems available, however, are unable to operate on 16 bit floats directly as they lack the proper arithmetic units to do so; and so an alternative method was devised. This involves converting the control matrix, stored as 16 bit float values, at execution of the MVM 32 bit float values, which the CPU is able to process. This is enabled via the Xeon Phi intrinsic instructions that allow converting 16 bit values 16 at a time and loading them directly into the 512 bit vector registers where they are ready for the FMA MVM operations. This ensures that there is a minimum overhead for the transfer of 32 bit values to conserve memory bandwidth.

To achieve this functionality a custom implementation of the MVM algorithm was designed using only Intel intrinsic functions to load to matrix, convert it to 32 bit floating point values, and compute the MVM operation. This was needed as the Intel MKL library that is used to calculate the reconstruction MVM in previous results is unable to load 16 bit floating point values. To be able to use it, the values would need to be converted to 32 bit before each call to the MKL library. This would not be ideal as MKL works best on larger MVM problem sizes and converting a large amount of the control matrix to 32 bit would defeat the purpose of storing it as 16 bit floats. Also making too many calls to the library would vastly increase the latency.

A similar custom 32 bit MVM implementation, simply loading the data instead of converting it, shows that this algorithm isn't as optimised as MKL. It gives an average frame time of 995 ± 6 µs with an RMS jitter of 5.96 µs which can be compared to results that use MKL on the same processor/motherboard of 930 ± 10 µs

from Figure 4.2.

The algorithm that uses a 16 bit control matrix decreases this average frame time to 902 ± 6 µs with an RMS jitter of 5.60 µs. The need to use a less optimised custom MVM and the need to convert to 32 bit floats introduces extra overhead which greatly reduces the potential gain. These results show that this implementation of storing and converting the 16 bit control matrix increases performance by only 3 % over the best case MKL results.

The next iteration of Xeon Phi after KNL, Knights Mill (KNM) which is available now, includes support for Intel variable precision operations (vector neural network instruction, VNNI) which include intrinsic instructions that can directly operate on 16 bit integer values by addition and multiplication to produce an accumulated 32 bit integer sum. This would require some conversion of the control matrix and slope values to fixed-point 16 bit precision integers but could reduce the memory bandwidth requirement without introducing a costly 16 to 32 bit conversion for each control matrix value at execution time. Simulations have shown that 16 bit fixed-point values would just be sufficient to provide the required precision (Basden et al., 2010), however when taking into account a real system with misalignments it may not be adequate.

The VNNI functionality comes via a redesigned vector processing unit (VPU) which also introduces quad FMA (QFMA) instructions; these allow sequential FMA to accumulate over four sets of calculations within a single instruction cycle. This has the potential to double the theoretical number of SP FMA operations possible per instruction cycle, increasing the theoretical peak SP-FLOPS by a factor of two over KNL. There are caveats to this, however, due to the instruction pipeline of the VPUs; a factor of two speed up is therefore unlikely. The QFMA operations should definitely benefit the highly vectorisable MVM and may make the 16 bit VNNI unnecessary; investigation into KNM VNNI and QFMA instructions could be useful for future work.

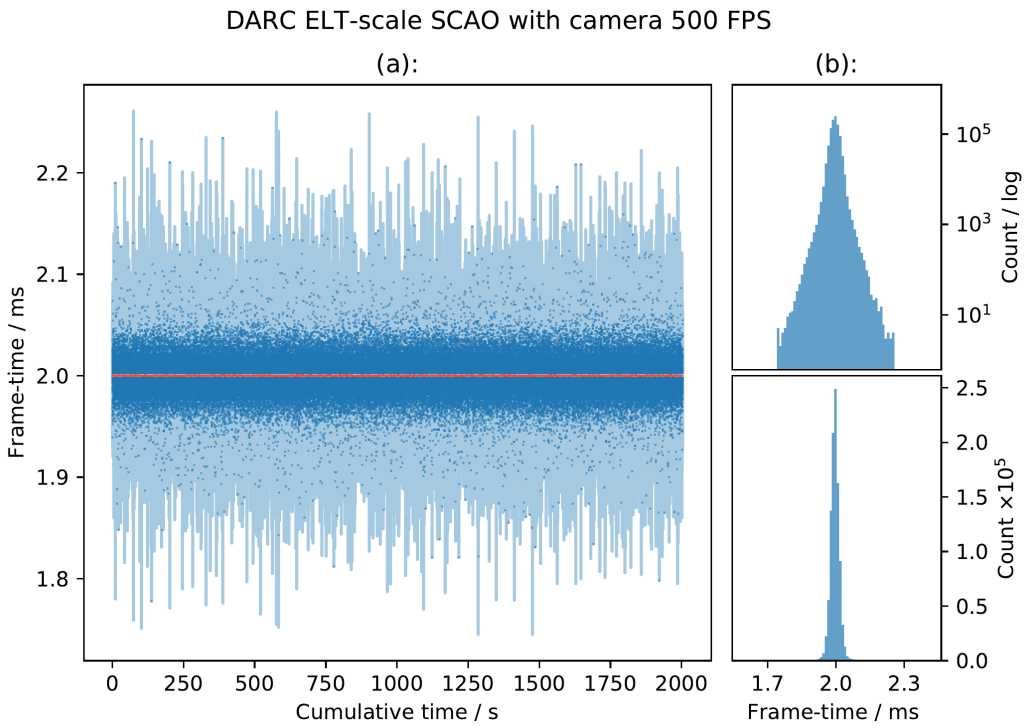


Figure 4.3: a) Frame times of DARC SCAO with a real wavefront sensor camera operating at 500 Hz for 10^6 frames, the red line shows a running median for every 1000 frame times. The average frame time is 2000 ± 20 µs. b) A histogram of the frame times, showing the distribution of jitter.

4.4 DARC SCAO with a real WFS camera

Figure 4.3 and Figure 4.4 show two sets of frame time results for DARC configured for ELT-scale SCAO, with pixels arriving from a real 10GigE Vision based camera running at 500 Hz and at the camera's maximum frame rate of 966 Hz respectively. The camera is an Emergent Vision Technologies HS2000M, delivering 100 pixels per subaperture. The figures show minimal numbers of outliers and also a small spread in the distributions, with rms jitters of 20.1 µs and 13.8 µs for 500 Hz and 966 Hz respectively, which is similar to that when DARC operates without a real camera.

The 64 s events due to the Intel motherboard are visible in the data for 966 Hz, however they are not seen in the data for 500 Hz. This is likely due to the reduced

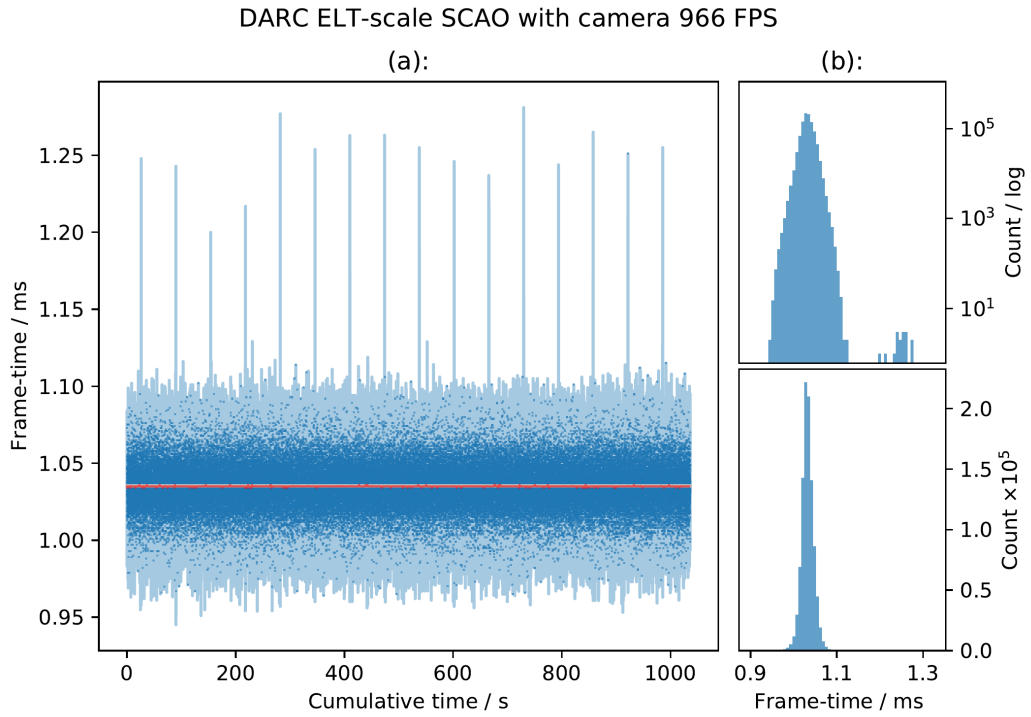


Figure 4.4: a) Frame times of DARC SCAO with a real wavefront sensor camera operating at 966 Hz for 10^6 frames, the red line shows a running median for every 1000 frame times. The average frame time is 1040 ± 10 µs. b) A histogram of the frame times, showing the distribution of jitter.

computational demands for SCAO at 500 Hz and so the CPU has ample time to process the interrupts without affecting DARC. The maximum instantaneous peak-to-peak jitter is 510 µs for 500 Hz and 163 µs for 966 Hz excluding the 64 s events, over one million frames.

The shapes of the histograms in Figure 4.3 and Figure 4.4 are quite different, the differences are more pronounced because they are plotted on a log scale. They show that for the camera operating at 500 Hz there is a high narrow peak at the mean of the distribution with relatively small numbers of frames spread out to either side. This gives the distribution its low RMS jitter but a relatively high relative instantaneous peak-to-peak jitter.

For interfacing with the real camera we used a modified version of the Aravis GigE Vision Library (AravisProject, 2018), which enables access to the pixel stream,

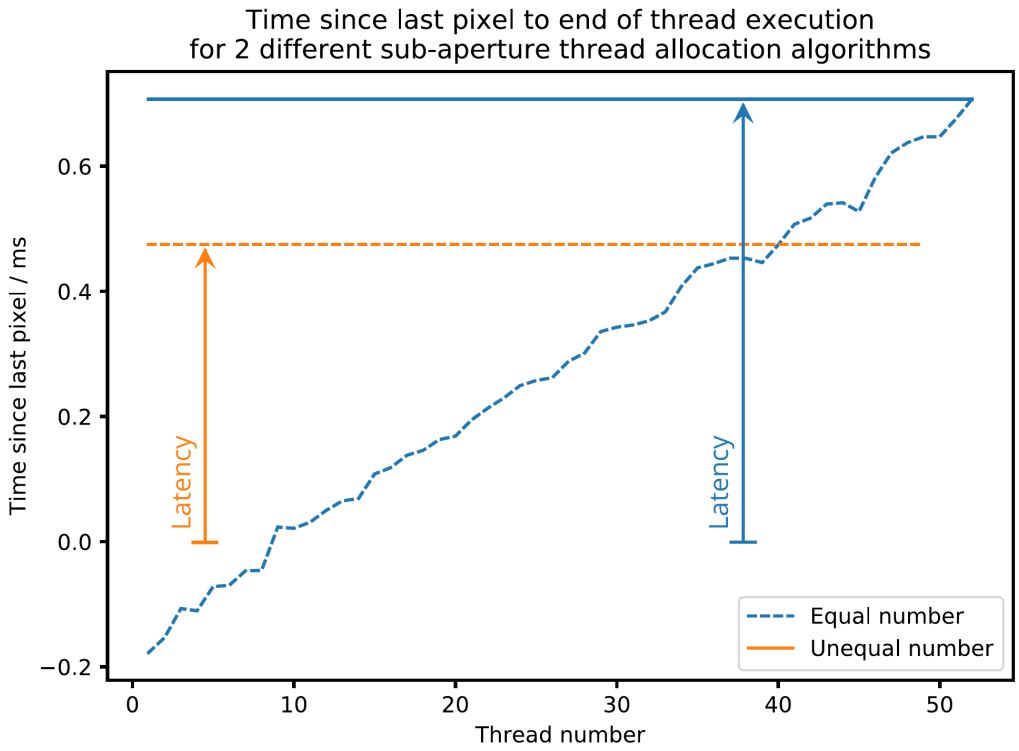


Figure 4.5: A measure of the time from the last pixel arriving from the camera to end of thread computation per thread. 'Equal number' shows the results for a naive subaperture allocation whereby each thread processes an equal number of subapertures. 'Unequal number' shows allocation by a simple algorithm which gives more work to threads which are processing subapertures whose pixels arrive earlier.

rather than waiting until the entire frame has been delivered. In this way, DARC can begin processing subapertures as soon as enough pixels have arrived, reducing latency. As the latency of an RTC is defined as the time between last pixel arriving and the final DM command being sent out, reducing this time improves the performance of the RTC.

For the case of multi-threaded AO RTC software it is important to ensure that none of the individual processing threads are taking substantially longer to finish processing than the others; as this will reduce the overall latency performance. Figure 4.5 shows the time taken for the DARC processing threads to finish processing their subapertures from the time the last pixel arrives from a real camera; these are average times for 10^5 frames. Figure 4.5 (Equal number) is for the case described in

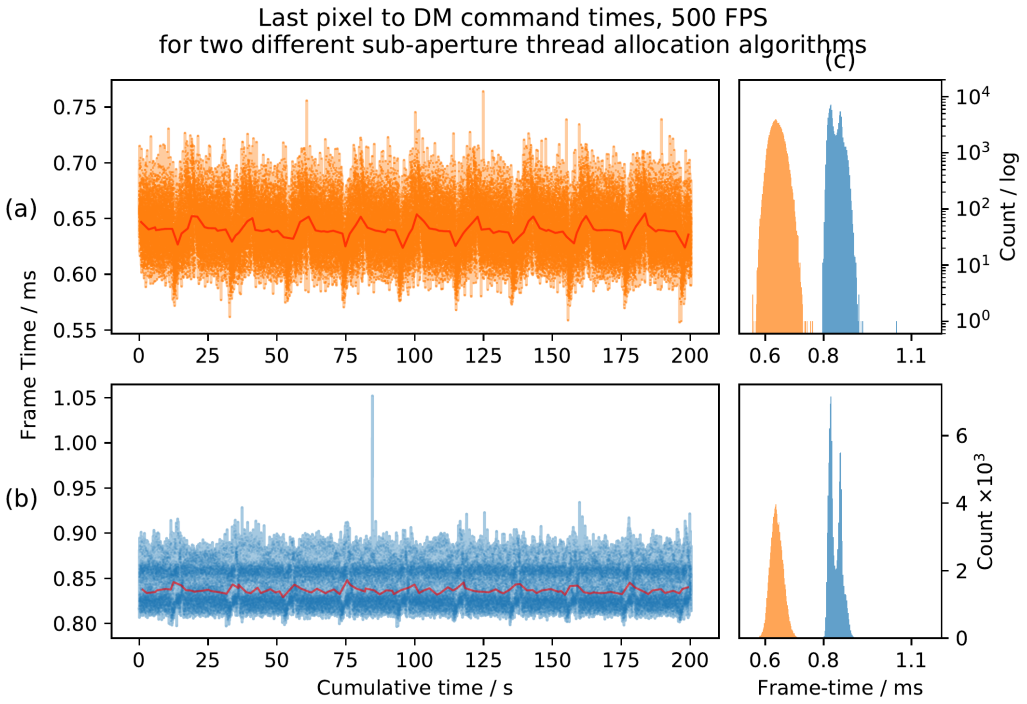


Figure 4.6: Latency measurements (time between last pixel received to DM demand ready) for DARC operation with a real wavefront sensor camera at ELT-scale at 500 Hz. a) shows results when using the unequal subaperture thread allocation. b) shows results for the equal subaperture thread allocation, Figure 4.5. c) shows histograms for each distribution using a log scale (top) and a non-log scale (bottom). The horizontal silver lines in (a) and (b) are the average frame times for each distribution, the red lines show a running median for every 1000 frame times.

Figure 3.7(c) and Figure 4.5 (Unequal number) is for the Figure 3.7(d) case. Both sets of data are taken with the real camera operating at 500 Hz. Figure 4.6 shows that for the situation with equal numbers of subapertures the mean RTC latency for 10^5 frames is 840 ± 20 µs, and for unequal numbers, the mean RTC latency is 640 ± 20 µs. Figure 4.5 shows a modest improvement in RTC latency, bringing the latency below that of the best case simulator and demonstrates the different end of thread execution times described in Figure 3.7. These results show that DARC on the Xeon Phi can operate SCAO at ELT-scales with a real camera.

The algorithm used to assign the unequal numbers of subapertures is a very basic implementation with a linearly decreasing subaperture count per thread. This algorithm will be explored further to find the optimal subaperture allocation to

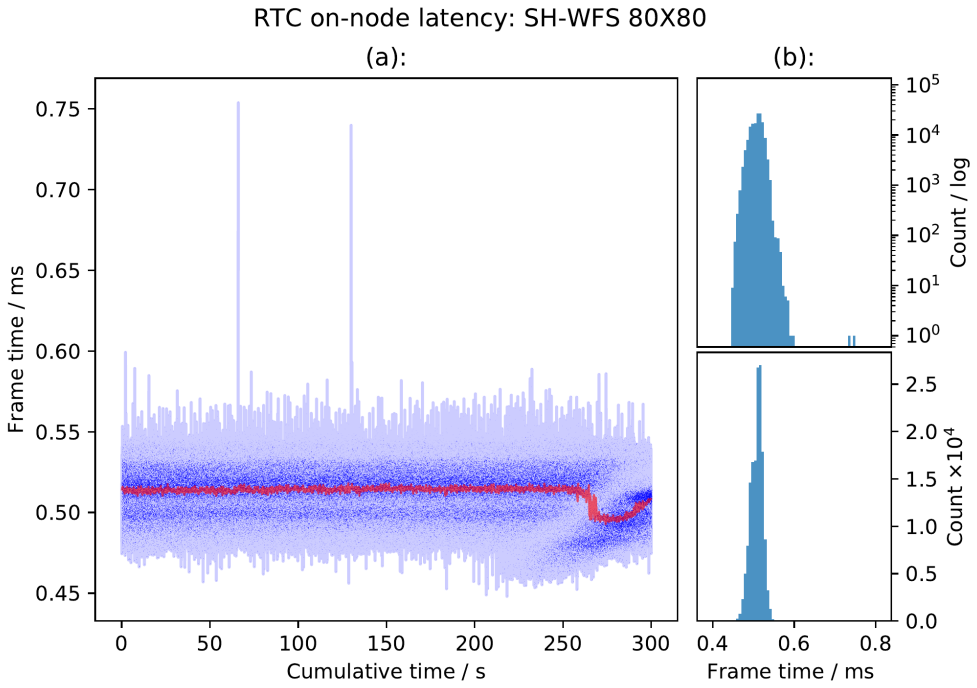


Figure 4.7: Frame time results for an SCAO setup using a SH-WFS type WFS slope calculation with 80 subapertures across the pupil. (a) shows the frame time distribution and (b) shows histograms of the data both on a log scale (top) and a non-log scale (bottom). Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is 511 ± 15 µs and the red line shows a running median for every 1000 frame times. The strange behaviour exhibited at 250 s is rare and due to it causing a reduction in the latency, albeit with a brief bi-modal distribution, it shouldn't have any negative affects on the AO performance.

improve latency for desired frame rates and different read-out rates.

4.5 DARC SCAO with the UDP camera simulator

Figure 4.7 shows frame time and latency results for DARC running an SCAO RTC on an Intel Xeon Phi 7250 with an attached simulated camera using the UDP pixel streaming method as described in Section 3.8. These results are for a SCAO setup with a single 80×80 SH-WFS with 4616 valid subapertures and an ELT-like M4 + M5 DM configuration with a total of 5318 actuators. The reconstruction therefore, is a single MVM of dimensions 5318×9232. There are 300 seconds worth of data corresponding to 1.5×10^5 frames at a frame rate of 500 Hz; the average latency is

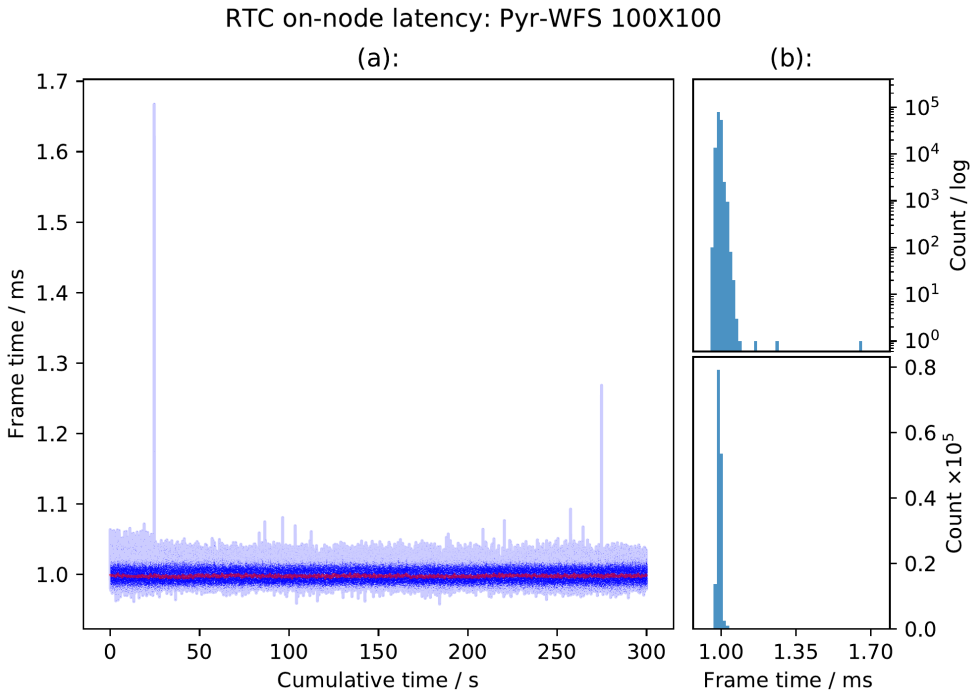


Figure 4.8: Frame time results for an SCAO setup using a Pyramid type WFS slope calculation with 100 pixels across each quadrant. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is $998 \pm 10\mu s$ and the red line shows a running median for every 1000 frame times.

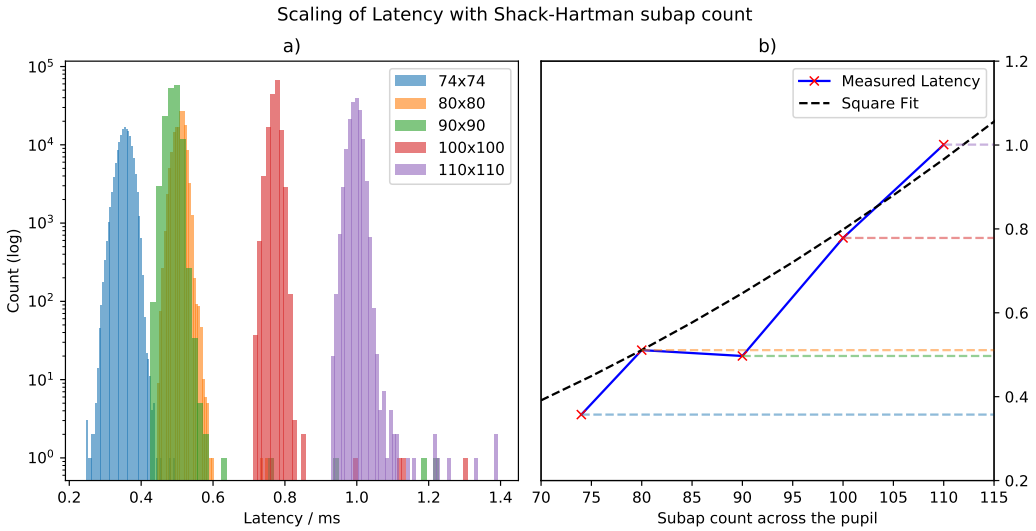


Figure 4.9: The scaling of latency with subapertures across the pupil for the Shack-Hartman type WFS. a) shows histograms of frame time data for each test while the relationship of the mean values is shown in b). The latency is that for the entire RTC operation from pixels to DM commands as shown in Figure 3.3, measured from the time the last pixel arrives until the DM command is ready. The relevant values for the data are shown in Table 4.1.

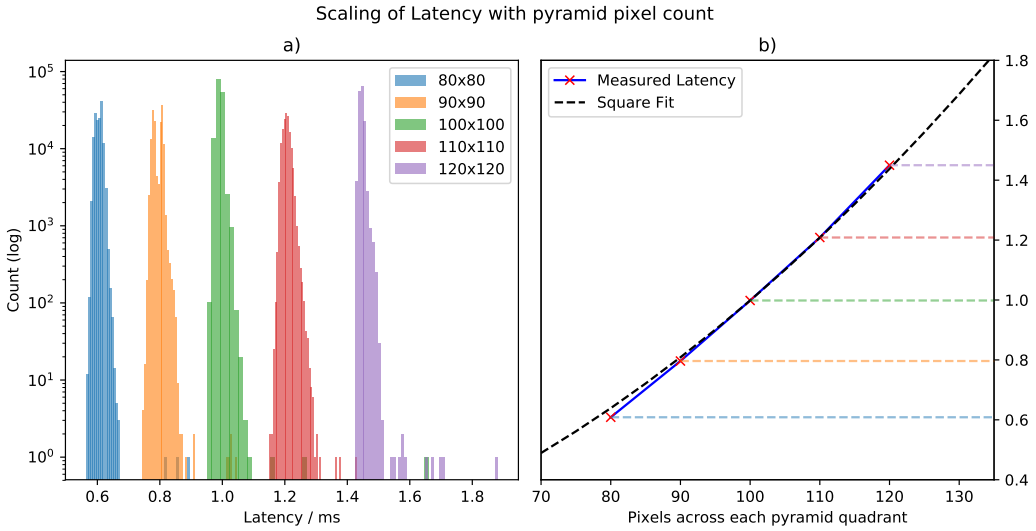


Figure 4.10: The scaling of latency with pixels across the pupil for the Pyramid type WFS. a) shows histograms of frame time data for each test while the relationship of the mean values is shown in b). The latency is that for the entire RTC operation from pixels to DM commands as shown in Figure 3.3, measured from the time the last pixel arrives until the DM command is ready. The relevant values for the data are shown in Table 4.1.

Table 4.1: Latency, RMS jitter and largest outliers results for all of the data presented in Sections 4.5 and 4.6. For all results the a) columns correspond to results from 1.5×10^5 continuous iterations and where no frame rate is given at 500 Hz for a total time of 300 s for each test case. The b) columns correspond to results from a subset of no less than 2×10^4 continuous iterations chosen from the larger a) data sets for a total time of 40 s each where no frame rate is given. The b) column subsets were chosen to avoid any large outliers that result from simulated camera delays to give a better representation of the "steady" latency.

	Mean Latency	RMS Jitter (μs)		Largest Outlier (μs)	
AO Mode	(μs)	a)	b)	a)	b)
Pyr-WFS 80×80	609	11	11	898	653
Pyr-WFS 90×90	796	16	16	1051	862
Pyr-WFS 100×100	998	10	9	1667	1049
Pyr-WFS 110×110	1209	12	12	1433	1292
Pyr-WFS 120×120	1450	8	8	1886	1508
SH-WFS 74×74	357	18	16	495	436
SH-WFS 80×80	511	15	15	754	589
SH-WFS 90×90	497	14	14	1237	559
SH-WFS 100×100	778	12	11	1314	821
SH-WFS 110×110	1001	14	14	1397	1065
Batch Allocation 500 Hz	348	12	-	605	-
Batch Allocation 700 Hz	406	47	-	1190	-

measured at $511 \pm 15 \mu s$. This compares favourably with similar results of $640 \pm 20 \mu s$ from Jenkins et al. (2018b), with a reconstructor of dimensions 5170×9416. The problem size used here has been adjusted to better reflect the potential dimensions of actual ELT instruments using more up to date information (Biasi et al., 2016; Correia, 2018).

By default when the camera simulator operates at 500Hz, the delay between the sending of each UDP packet is set to a finite value such that the total read-out time is 1400 µs. This was chosen to emulate the readout of the ESO LVSM cameras as described in Downing et al. (2018). However for results presented in later sections where other frame rates were used, the simulated camera was operated with a zero time delay between the sending of packets to keep the results consistent. The latencies obtained with the zero time delay are generally greater than reported here where the finite inter-packet delay was used. This is because the last pixels arrive at the RTC sooner and so there is less time to process the RTC operation during read-out, resulting in a greater RTC latency. All results in this section should be assumed to use the 1400 µs read out unless otherwise stated.

Figure 4.8 shows frame time results for a similar AO setup as above but using a Pyr-WFS instead of a SH-WFS, with the differences between the WFS processing as described in Section 2.1.3. There are 300 seconds worth of data corresponding to 1.5×10^5 frames at a frame rate of 500Hz, and the average latency here is measured at $998 \pm 10 \mu s$. The dimensions of 100×100 pixels for the Pyr-WFS and 80×80 subapertures for the SH-WFS were chosen as these are the likely dimensions that would be used for real a WFS on ELT-scale instruments (Hippler, 2018). The SH-WFS dimensions result from technological limitations of the sensors being developed by ESO and the required dimensions of each subaperture (Schreiber et al., 2018). The Pyr-WFS dimensions were chosen based on what is being targeted for the ESO ELT first light instrument HARMONI (Thatte et al., 2014) SCAO mode (Schwartz et al., 2018).

Figures 4.9 and 4.10 show the latency scaling of both the SH-WFS RTC and Pyr-

WFS RTC against subapertures/pixels across the pupil. Due to the use of a simulated camera as described in Section 3.8, there are some unavoidable latency spikes that result from delays of the pixel transmission from the simulated camera. The mean latencies, RMS jitters and largest outliers are shown in Table 4.1 for data sets containing 1.5×10^5 samples at 500Hz. Table 4.1 also shows the RMS jitter and largest outliers for each case for a reduced subset of the data containing at least 2×10^4 continuous iterations. These reduced sets were chosen to eliminate any major outliers resulting from camera delays to give a better idea of the "steady" latency distribution.

For all of these results the input image sizes are kept constant for each type of WFS processing used; images of 800×800 pixels are used for the SH-WFS and 240x240 for the Pyr-WFS and either the number of pixels per subaperture is reduced to provide more subapertures or a smaller area within the input image is used for reduced numbers of subapertures. In this way the pixels received are kept constant between the different WFS and only the calibration, centroiding and reconstruction are affected by the different dimensions. We can see that for the Pyr-WFS the scaling matches very closely to a square fit, which is as expected; because the change in degrees of freedom in the reconstruction scales with the second power of the number of pixels across the pupil.

For the SH-WFS results shown in Figures 4.9 there is no clear fitting to a square fit and the case of 90×90 subapertures actually has a lower latency than the 80×80 subapertures case. We believe this is due to the reduced number of pixels per subaperture when using 90×90 subapertures. Because all of the SH-WFS tests use the same simulated camera image dimensions of 800×800 pixels, the 90×90 subapertures case is only using 8×8 pixels per subaperture compared to 10×10 pixels per subaperture for the 80×80 case. This reduces the perceived latency in two ways; firstly by the reduced pixel processing required and secondly by the fact that the latency is measured as the time from when the last pixel arrives and due to the way this is measured, the timestamp is taken when the full 800×800

image has arrived which is later than when the processing has completed for the 90×90 subapertures case. The 8×8 pixel subapertures would also be processed more efficiently by vectorisation due to the 512 bit wide vector registers allowing 16 values to be computed simultaneously.

If the latencies for the Pyr-WFS in Table 4.1 are compared directly with the latencies from the SH-WFS for the same WFS dimensions, we see that the SH-WFS overall results in reduced latencies. This is a result of the reduced pipelining efficiency of the Pyr-WFS vs. the SH-WFS as described in Section 2.1.3 and shown in Figure 2.5.

4.6 Batch Subaperture Allocation

The results presented thus far have used a fixed subaperture allocation for each thread as described in Chapter 2, using the unequal subaperture allocation described in Section 3.6.2.1. Here I present similar results using the batch processing of subapertures as described in Section 3.6.2.2. The batches are devised as described in Section 3.6.2.2, by putting each row of subapertures into a separate batch. The 80×80 subaperture SH-WFS case gives 80 individual batches with varying numbers of subapertures in each one due to the circular aperture and central obscuration. The algorithm then lets each thread process a batch as soon as it is ready and with 54 total reconstruction threads as before, at least 26 of the threads will process more than one batch. The strength of this method is that the balancing of work between threads is done automatically at run time and little ahead of time optimisation is needed.

Figure 4.11 shows results of the batch allocation with the configuration as described above with the simulated camera operating at 500 Hz. The overall RTC latency has been decreased by up to 160 µs compared to the explicit subaperture allocation as described above. The jitter on the RTC has remained roughly constant compared to other subaperture allocation method. Figure 4.12 shows results of the batch

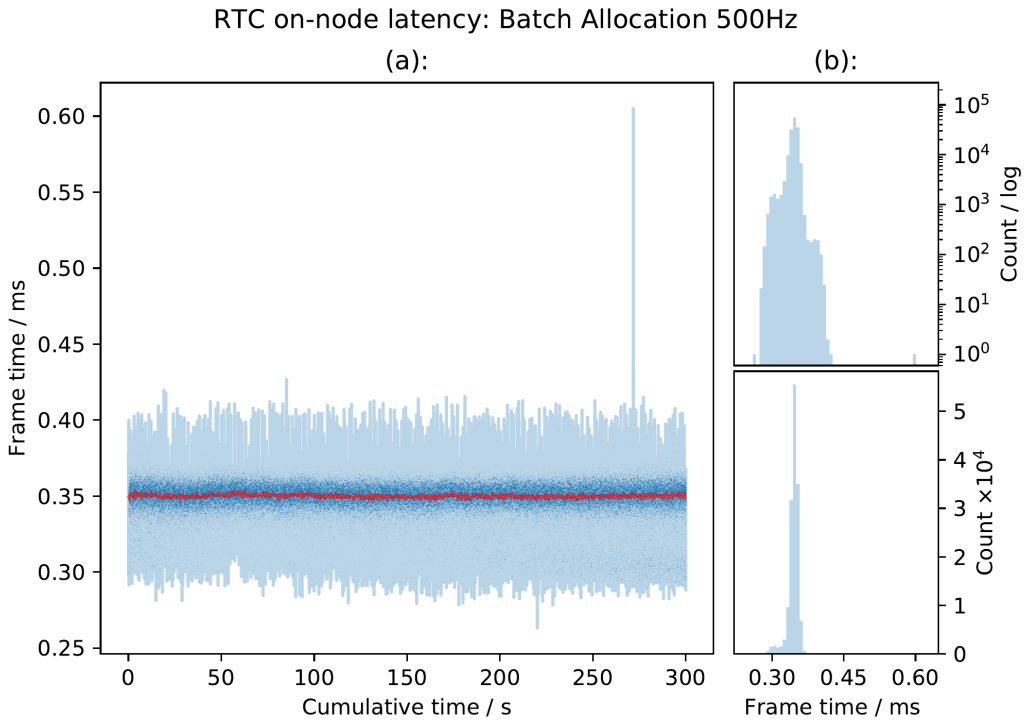


Figure 4.11: Frame time results for an SCAO setup using a Shack-hartman type WFS slope calculation using the batch subaperture allocation scheme described in Section 2.2. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is $348 \pm 12\mu s$ and the red line shows a running median for every 1000 frame times.

allocation with the configuration as described above with the simulated camera operating at 700 Hz. For these results the simulated camera was operating with a zero inter-packet delay resulting in a greater average RTC latency. The jitter on the results obtained at 700 Hz is noticeably worse than the results obtained at 500 Hz.

This method does however reduce the maximum computational no-camera performance compared with the equal subaperture allocation due to the number of chunks being different to the number of threads and so without the pipelining of pixels, some threads need to process more subapertures than others. The best performance would be achieved with a compromise between the two methods whereby the rows of subapertures are split into chunks and they are assigned to specific threads such that the workload is balanced and each thread processes the same

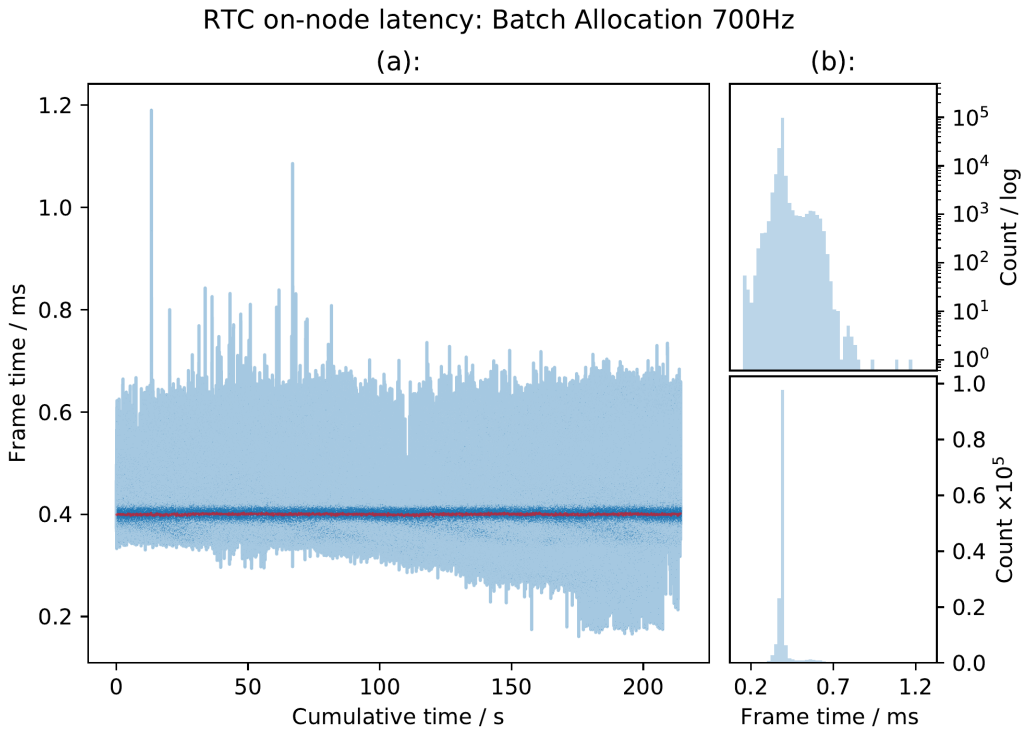


Figure 4.12: Frame time results for an SCAO setup using a Shack-hartman type WFS slope calculation using the batch subaperture allocation scheme described in Section 2.2. Results shown are for 1.5×10^5 iterations at 700 Hz for a total time of $214.3s$. The mean latency is $406 \pm 47\mu s$ and the red line shows a running median for every 1000 frame times.

subapertures each frame.

4.7 SCAO POLC

As described in Section 2.2, to perform any tomographic reconstruction of the atmospheric structure it is necessary to either directly measure the turbulent atmosphere in open-loop configuration or to reconstruct the POL wavefront from the residual wavefront and the previous DM shape. This can either be calculated explicitly or implicitly as shown in Basden et al. (2019); here we present the RTC latency of an ELT-scale SCAO DARC configuration with both types of POLC computation and a comparison with a no-POLC setup. The results were obtained with the same ELT-scale configuration used throughout this chapter and using the Intel

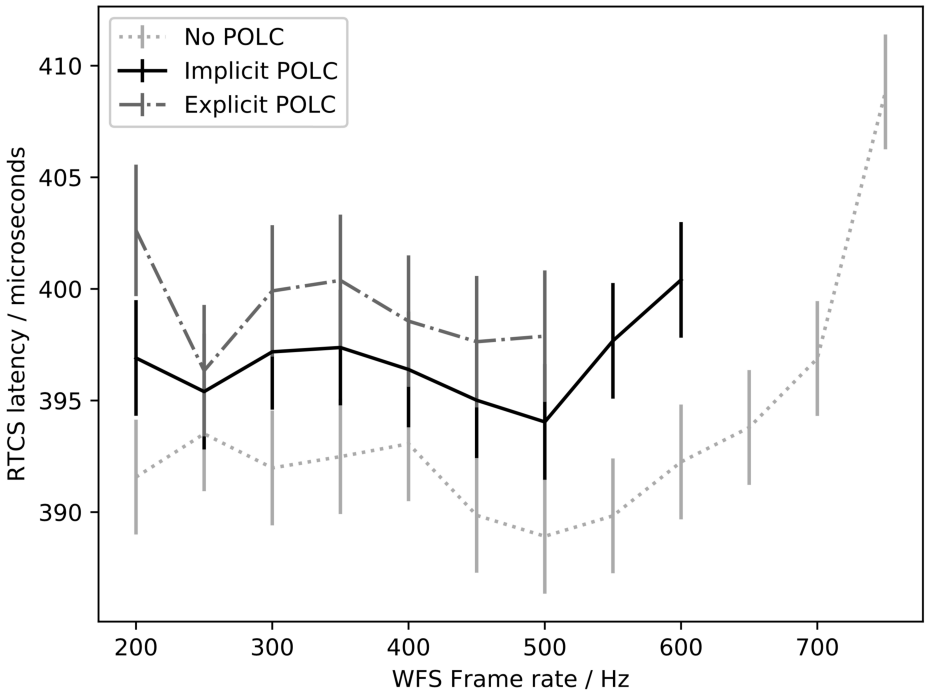


Figure 4.13: The RTC latency as a function of frame rate for the explicit and implicit POLC methods compared with the case where no POLC compuation occurs. The explicit POLC method is unable to operate at frame rates exceeding 500 Hz whilst the implicit POLC method extends the maximum frame rate to 600 Hz (Basden et al., 2019).

Xeon Phi 7250 hardware. Results were obtained with the UDP camera simulator at increasing WFS frame rates until each different mode was unable to keep up and began to drop frames. The camera simulator inter-packet delay was set to zero to keep results comparable across the different frame rates.

Figure 4.13 shows a comparison of the two different POLC methods and no-POLC operation as described above with the results displayed in Table 4.2. As can be seen from the figure, both POLC modes are unable to keep up with the no-POLC operation at the higher frame-rates and the RTC latency for the POLC modes is generally increased at all frame rates. With no-POLC the RTC is able to operate at up to 750 Hz with the latency having a steady increase above 500 Hz. The implicit POLC is able to achieve a maximum frame rate of 600 Hz and also shows a steady increase in latency above 500 Hz. The explicit POLC mode is only able to achieve a

Table 4.2: Latency and RMS jitter results for the SCAO POLC data shown in Figure 4.13 and discussed in Section 4.7. For all results the means and RMS are computed from 1.5×10^5 continuous iterations. Exp refers to results of the explicit POLC method, Imp refers to implicit POLC and None is for the case where no POLC computation occurred.

		Mean Latency (μs)			RMS Jitter (μs)		
		Exp	Imp	None	Exp	Imp	None
Frame Rate (Hz)	200	403	397	392	8	8	7
	250	396	395	394	9	8	9
	300	400	397	392	7	7	9
	350	400	397	393	7	8	9
	400	399	396	393	7	7	9
	450	398	395	390	7	7	9
	500	398	394	389	8	9	7
	550		398	390		7	9
	600		400	392		8	9
	650			394			9
	700			397			8
	750			409			28

maximum frame-rate of 500 Hz before it begins to drop frames. These results show that the implicit POLC method does indeed have a smaller cost on the operation of the RTC and allows SCAO POLC to achieve a maximum frame-rate of 600 Hz on the Xeon Phi hardware.

4.8 Long Time Period AO RTC Operation

The continuous operation of AO systems over long periods is extremely important for situations where the astronomical observations require an AO corrected beam for long integration times. During these time periods, any interruption in the AO performance of the system can have dramatic effects on the final image quality of long exposure images. This requires the AO RTC to provide wavefront corrections with consistent latencies to match the requirements due to the constantly changing atmospheric conditions during the course of observations. Here I present long time period results of the DARC RTC software to demonstrate its stability over time periods exceeding 8 hours.

Table 4.3: A table showing the percentages of the frame times for the long period results that fall within and below multiples of the RMS of the distribution; the latencies are shown in Figure 4.14. The values for 'Within' fall within the range $(\mu - n\sigma, \mu + n\sigma)$, where μ is the mean latency of 335 µs, σ is the RMS jitter of 16 µs, and n is a multiple of σ. The values for 'Below' are the for latencies $(< \mu + n\sigma)$.

	Deviation From the Mean					
Percentage (%)	0σ	1σ	2σ	3σ	4σ	5σ
Within	0.00	65.81	97.18	99.72	99.97	99.99
Below	48.25	83.73	98.68	99.93	99.99	99.99

Figure 4.14 shows latency results for 1.5×10^7 frames for an ELT-scale SCAO configuration using a SH-WFS and the batch subaperture allocation as described in Section 4.6. This is operating at 500 Hz for a total time of 8.3 hours; the WFS images are delivered at this rate by the UDP camera simulator described in Chapter 3. Also shown in Figure 4.14 (b) is a 1.8×10^6 frame subset of these results which corresponds to one hour of continuous operation. This highlights a regular spike in the latency measurements that occurs at 5 minute intervals. These correspond to frame drops detected from the camera simulator and so are not representative of real results using a deterministic camera.

The mean latencies for both distributions shown in Figure 4.14 are both measured at 335 ± 16 µs; this is the same regardless of whether the regular latency spikes due to the camera simulator are removed. Table 4.3 shows percentages for the latency values of the full distribution that fall within certain time limits. The percentages are given for the number of latency values that fall within multiples of the RMS value from the mean and also for the number of latency values that fall below the mean plus multiples of the RMS jitter. This is because for AO RTC we are not particularly concerned with latencies that are less than the requirements, it is only frames that take longer than required that will affect AO RTC performance. These results demonstrate that the DARC RTC software is stable over long periods of time and that 99.73 % of the frame latencies fall within 3 standard deviations from the mean. They also show that 99.93 % of the frame latencies fall below 3 standard deviations above the mean.

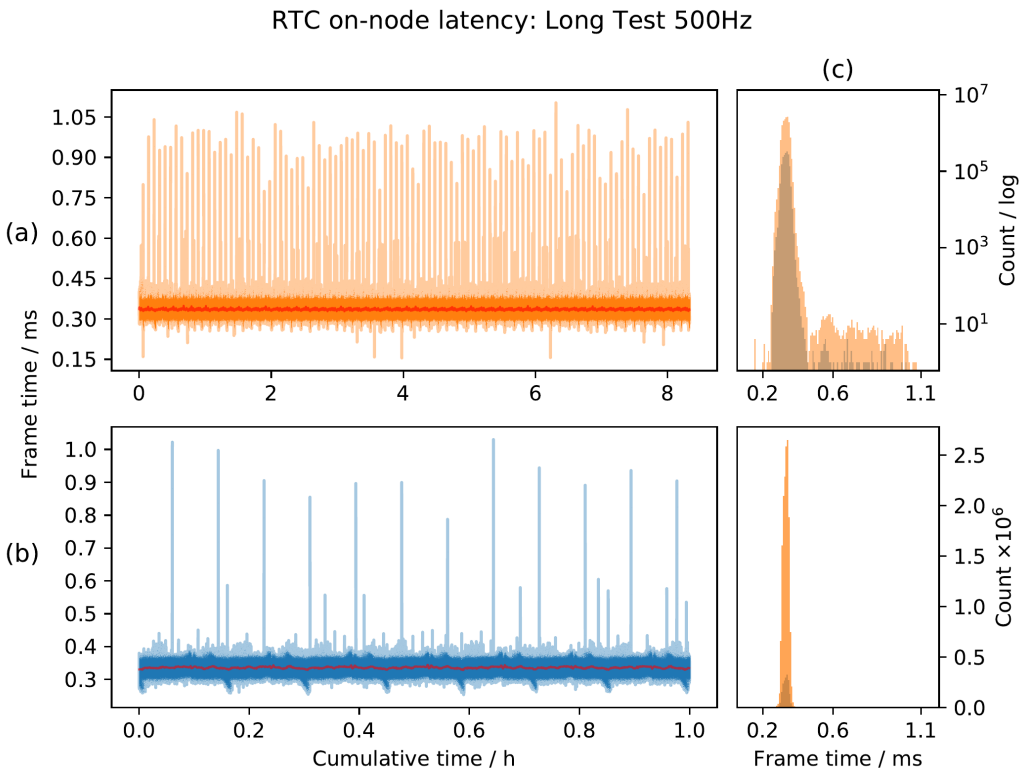


Figure 4.14: Frame time results for an SCAO setup using a Shack-hartman type WFS slope calculation using the batch subaperture allocation as described in Section 2.2. Results shown in (a) are for 1.5×10^7 iterations at 500 Hz for a total time of 8.3 hours. Results in (b) show a 1.8×10^6 frame subset of these results, which corresponds to one hour of continuous operation; this is to highlight the regular latency spikes at 5 minute intervals due to the camera simulator. (c) shows histograms of the data sets shown in (a) and (b) with a log scale (top) and a non-log scale (bottom). The non-log scale demonstrates that the vast majority of frame times fall within a very narrow distribution around the mean.

4.9 Chapter Summary

The chapter presents results of testing the DARC AO RTC software for SCAO systems and utilising the CPU camera simulator. The results show that a single Intel Xeon Phi Knights Landing processing node is able to compute the AO RTC for ELT-scale SCAO systems with latencies as low as 348 ± 12 µs for a WFS frame rate of 500 Hz, and has the ability to process WFS images delivered at up to 750 Hz. The RTC latency was measured for the processing of both a SH-WFS and a Pyr-WFS configuration, with an investigation of how the latency for each scales

with the dimensions of the WFSs.

Different subaperture allocation schemes were investigated with the most efficient for pipe-lined WFS operation being an un-equal subaperture allocation. The effect of POLC computation for the SCAO case was explored, involving both explicit and implicit POLC methods. It was found that both methods reduce the maximum framerate that the RTC could achieve on the given hardware, however the implicit POLC had much less of an effect on the maximum achievable performance.

A long time period test demonstrates the stability of the CPU-based RTC software, giving consistently low latencies over the full 8 hour duration. The camera simulator does introduce extra latency due to missing frames however, and the Xeon Phi's single threaded performance does hinder its ability to process serial tasks and especially network related tasks.

Chapter 5

MCAO Demonstrator: Multi-node Xeon Phi Cluster

In this chapter I present results of an investigation into designing a multi-node many-core CPU architecture for the processing of MCAO and LTAO RTC. First is presented the process of designing the architecture based on the DARC software and its many-core optimisations as described in Chapter 3. This is followed by results of testing the prototype architecture for the MCAO and LTAO cases using seven Xeon Phi processing nodes and a camera simulator as described in Section 3.8 to deliver unique pre-simulated WFS images to each node. Results are presented of an investigation into how the latency is affected whilst streaming telemetry during RTC operation for the MCAO case. Finally results are presented for the MCAO implicit POLC calculation as described in Section 2.1.4. Results and discussion in this chapter have been previously published in Jenkins et al. (2019).

5.1 Prototyping an MCAO and LTAO RTC

As mentioned in Chapter 1 MCAO and LTAO generally differ from SCAO by the number of WFSs and DMs used. Therefore in the context of the RTC they are both much more computationally demanding and more complex than the simple SCAO

Table 5.1: A comparison of the specifications of ELT-scale SCAO, MCAO and LTAO, the values are the most current known specifications for the HARMONI SCAO mode, the MAORY MCAO mode and the HARMONI LTAO mode respectively.

Mode	SCAO-SH	SCAO-Pyr	MCAO	LTAO
Target Frame rate (Hz)	700	1000	500	500
LGS number	0	0	6	6
LGS subaperture geometry	N/A	N/A	80×80	80×80
LGS pixel geometry	N/A	N/A	10×10	10×10
LGS total subapertures	N/A	N/A	4616×6	4616×6
LGS image format	N/A	N/A	800×800	800×800
NGS number	1	1	3	1
NGS type	SH-WFS	Pyramid	SH-WFS	SH-WFS
NGS subaperture geometry	80×80	100×100	2×2	2×2
NGS pixel geometry	10×10	N/A	100×100	100×100
NGS total subapertures	4616	4616	4×3	4
NGS image format	800×800	240×240	240×240	240×240
DM number	1	1	3	1
Total DM modes	$5316 + 2$	$5316 + 2$	$5316 + 2$ $+2 \times 500$ $+6 \times 2$	$5316 + 2$ $+6 \times 2$

case. When prototyping an RTC architecture for MCAO and LTAO we decided to design it based on the ELT first-light instruments, the Multi-conjugate Adaptive Optics RelaY (MAORY, Ciliegi et al., 2018) and the HARMONI LTAO mode (Neichel et al., 2016). For both of these instruments, the most demanding aspects will be the reconstruction of 6 laser guide star (LGS) WFSs operating at 500Hz. The parameters for these instruments and a comparison with an SCAO system are shown in Table 5.1. Targeting proposed ELT MCAO and LTAO instruments allows us to demonstrate more realistic test cases and puts better constraints on the design of the architecture.

One of the main benefits of designing a CPU-based RTC lies in the flexibility and generality that the CPU architecture provides. The product of this is that the software optimisations and modifications detailed in Chapter 3 for SCAO can be readily applied to different AO types such as MCAO and LTAO. Therefore

the architecture we propose for these AO regimes is an extension of the SCAO case by scaling the software and hardware to match the increased computational complexity.

For the MAORY MCAO and HARMONI LTAO parameters detailed in Table 5.1 the computational demands for each LGS WFS are similar to those of the SH-WFS SCAO case tested in Section 4.5; albeit with increased DM actuator DoFs due to additional DMs and laser-pointing tip-tilt mirrors but targeting a reduced framerate. We therefore decided that for each LGS WFS the processing of the WFS images to reconstructed wavefronts can be done in a very similar way to the SCAO procedure in Section 4.5 by processing a single LGS WFS per processing node.

The NGS parameters demand far fewer computational resources than the SCAO case demonstrated on a single node in Section 4.5, therefore we propose that the NGS WFS processing, 3 NGS for MAORY and 1 for HARMONI, can be achieved by a single instance of DARC on a single processing node. The partial DM commands resulting from these calculations then need to be summed together to produce a single DM command vector for all of the required DMs. This will be achieved by having a separate "master" processing node which can receive partial DM commands from the reconstruction nodes and perform any post-processing that may be required before delivering the final actuator commands to the DMs.

Figure 5.1 shows the prototype architecture for MCAO/LTAO RTC using the ideas described above. There are a total of 7 reconstruction nodes to process all the WFSs required for the either the MAORY MCAO case or the HARMONI LTAO case. An 8th master node is used for summing the partial results from each reconstruction node and processing them for sending to the DMs. Due to the way the ESO ELT M4 will operate (Xompero et al., 2018), the correction applied by M4 will not necessarily be the same as that which is delivered to M4 from the RTC. It will therefore be necessary for the RTC of an ELT instrument to receive feedback from M4 such that it can incorporate the actual correction applied for the previous iteration. In

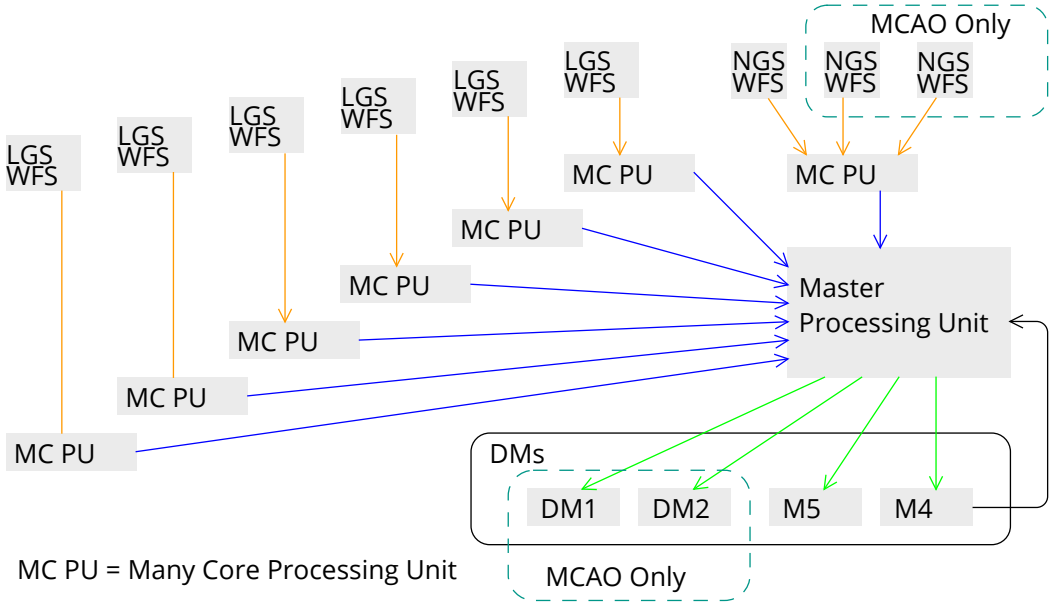


Figure 5.1: The proposed architecture for the MCAO and LTAO multi node RTC. The 6 LGS WFSs are each processed by a single many-core node. The NGS are all processed on the same node, three for MAORY and one for HARMONI. The master node sends out the final DM commands once it has finished summing and processing the partial vectors. The master node should also able to receieve feedback from the ESO ELT M4 to integrate the actual M4 shape used in the next command.

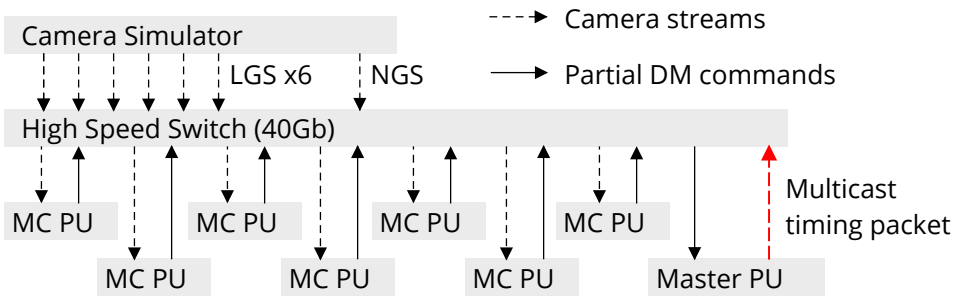


Figure 5.2: The lab test setup for the MCAO and LTAO architecture. The simulated camera streams 6 LGS WFSs and either 1 or 3 NGS WFSs through a high speed switch to the processing units. The many-core processing units send their partial DM commands calculated from the individual WFSs to the master node for summing and further processing. For the tests to determine the overall latency of the system, the master node multicasts a timing packet for all other nodes to time stamp the end of frame.

our prototype architecture, this will be processed by the master processing unit, shown in Figure 5.1.

Figure 5.2 shows a lab test set up derived from the prototype architecture shown in Figure 5.1. It shows the simulated camera delivering the required camera streams to each reconstruction node over a high speed network. The master processing node communicates with the reconstruction nodes over a separate high speed network on the same physical switch but using a different VLAN and different network interconnects to the camera streams. For our lab test setup the full RTC latency is calculated on each reconstruction node by measuring the time between when it receives the last pixel from its camera stream to the time it receives a timing packet from the master which is multicast to the reconstruction nodes when it has completed the current frame. As the cameras use the proposed ESO MUDPI packet format they stamp each image with a frame number which is then propagated through to the master and back through the timing packet to be able to match the correct DM command to the camera frames.

5.1.1 UDP cameras simulator setup for MCAO/LTAO

For the testing of the architecture described above, the UDP camera simulator was configured to stream the 7 individual pixel streams required, one for each of the reconstruction nodes. The camera streams were configured such that they delivered previously simulated frames stored on the fast storage of the simulator machine. The simulated frames were made with The Durham Adaptive Optics Simulation Platform (DASP), configured for a MAORY type AO system as described in Table 5.1 in open loop. Open loop operation was chosen due to the difficulty of constructing a closed-loop reconstructor that produced valid WFS images, and closed loop WFS images were not required.

Each of the six LGS wavefront sensors were separately simulated with an asterism

Table 5.2: Latency, RMS jitter and largest outliers results for all of the data presented in this chapter. For all results other than LTAO with buffer swap the a) columns correspond to results from 1.5×10^5 continuous iterations at 500 Hz for a total time of 300 s for each test case. The b) columns correspond to results from a subset of no less than 2×10^4 continuous iterations chosen from the larger a) data sets for a toal time of 40 s each. The b) column subsets were chosen to avoid any large outliers that result from simulated camera delays to give a better representation of the "steady" latency. The LTAO with buffer swap results are for 1.5×10^4 continuous iterations at 500 Hz for a total time of 30 s, there are no "steady" results for this case as the outliers here are a result of the parameter swap itself and not from an external factor.

	Mean Latency	RMS Jitter (μs)		Largest Outlier (μs)	
AO Mode	(μs)	a)	b)	a)	b)
Full MCAO	985	33	29	4465	1235
Full LTAO	894	29	28	4434	1174
MCAO telemetry	1085	32	30	2988	1466
MCAO POLC	1090	45	44	2880	1312
MCAO 6 LGS	992	47	46	3498	1143
MCAO 5 LGS	979	46	44	2870	1261
MCAO 4 LGS	969	45	45	3305	1285
MCAO 3 LGS	943	43	42	2817	1182
MCAO 2 LGS	951	42	43	2858	1119

where each LGS was projected 60 mas[1] off-axis equally spaced around the central axis. For simplicity and because simulated LGS elongation was not required, the LGS were simulated as point sources so that no elongation of the spots was present in the images. The NGS WFS images were projected 90 mas off-axis and for the LTAO tests, only one of the NGS WFS image sets were used. The atmosphere was simulated using a 35 layer atmospheric model as described in Sarazin et al. (2013) with an r_0 value of 0.137 m and an outer scale of 10 m. Figure 5.3 shows one of the WFS images used with the camera simulator for the MCAO and LTAO prototype and Figure 5.4 has a more detailed view of the first quadrant of the image.

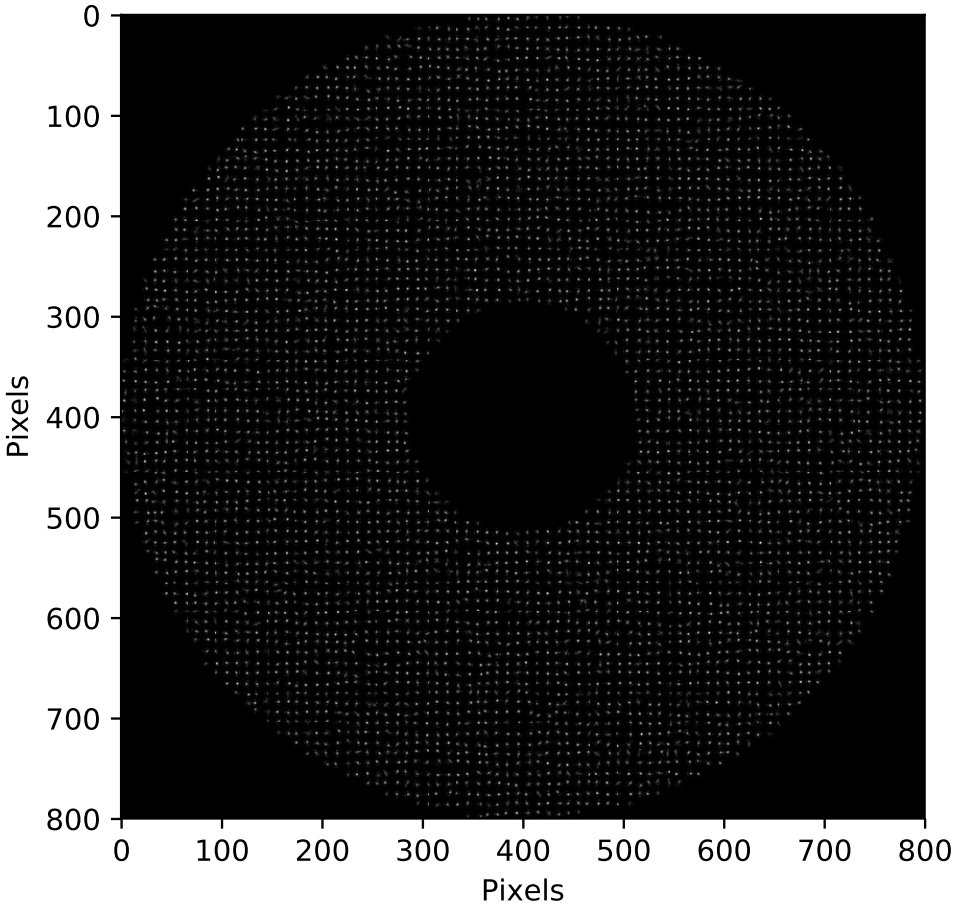


Figure 5.3: Simulated ELT-scale LGS WFS images without elongation.

5.2 Results of testing the prototype

When testing the MCAO prototype described in Section 5.1 certain considerations needed to be made with regards to the timing of the individual frames. Ideally we would want to measure the time from the last pixel into the RTC until the time the DM command is ready, however we discovered that for a multi-node CPU-based architecture it is difficult to synchronise the clocks between nodes with enough precision using our available hardware such that timestamps generated on

[1] milliarcsecond (mas)

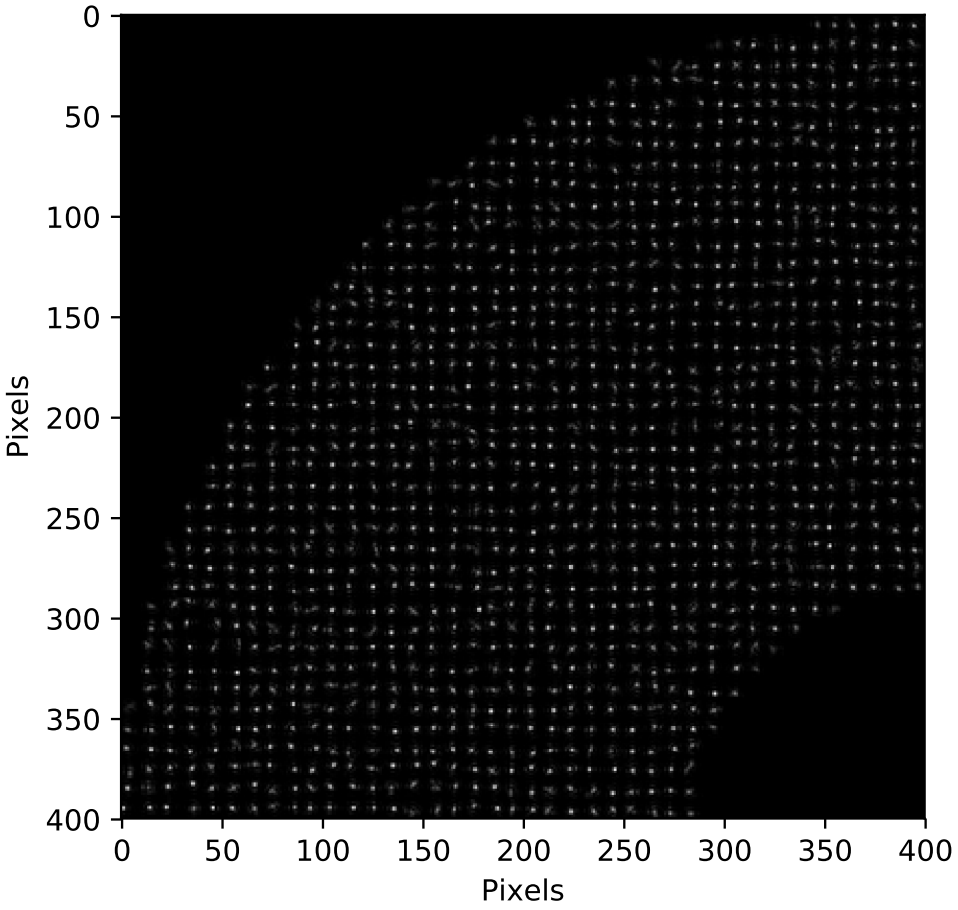


Figure 5.4: A zoom in on the first quadrant of a simulated ELT-scale LGS WFS images without elongation.

each node can be directly compared. Instead we have to rely on only comparing timestamps generated on individual nodes and so the full RTC latency is calculated as the time between a reconstruction node receiving the last pixel from its camera stream and the time it receives a timing packet from the master node indicating that the final DM command is ready, which does result in a slightly pessimistic measurement.

Ideally the full RTC latency would be measured externally with a device that is able to record the time between when the camera finishes sending a frame and the time when the corresponding DM command is received from the master node. This

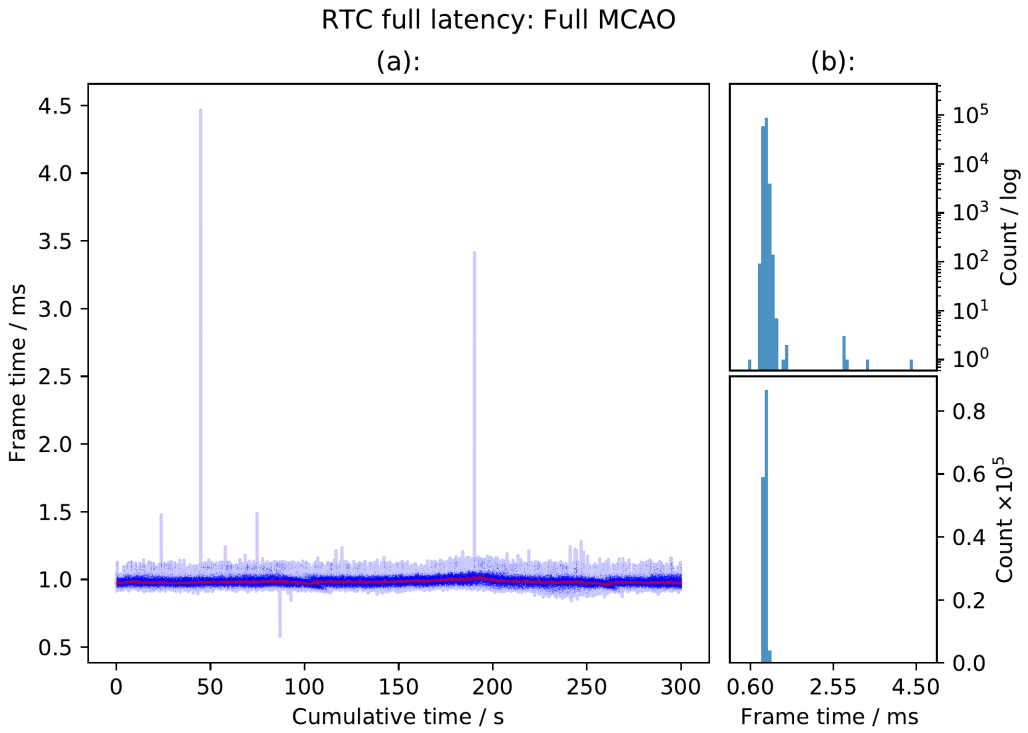


Figure 5.5: Latency results for the full MCAO setup as described in Section 5.1 and Table 5.1. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s. This is a measure of the time between when the last pixel arrives at a reconstruction node and when it receives the timing packet from the master node. The large outliers are result of delays from the CPU-based simulated cameras, which is compounded by the fact that this timing data includes delays from all 7 simulated camera streams.

process would make the RTC latency measurement easier. However the method described above is no less valid as a measure of the RTC latency.

For all the MCAO and LTAO tests described in this report, the system architecture used is shown in Figure 5.1 and network interconnects are as shown in Figure 5.2. Each of the 7 reconstruction nodes uses a single instance of the DARC software to receive an 800×800 pixel camera stream which it processes from pixels to a partial DM command for that WFS. This processing is done in a very similar way to the SH-WFS SCAO case as described in Chapter 2 and in Section 4.5; the main differences are the way in which the reconstruction matrix is constructed, and the DM software library that is used to send the partial DM command vector to the master node. The master node itself runs a separate instance of DARC which

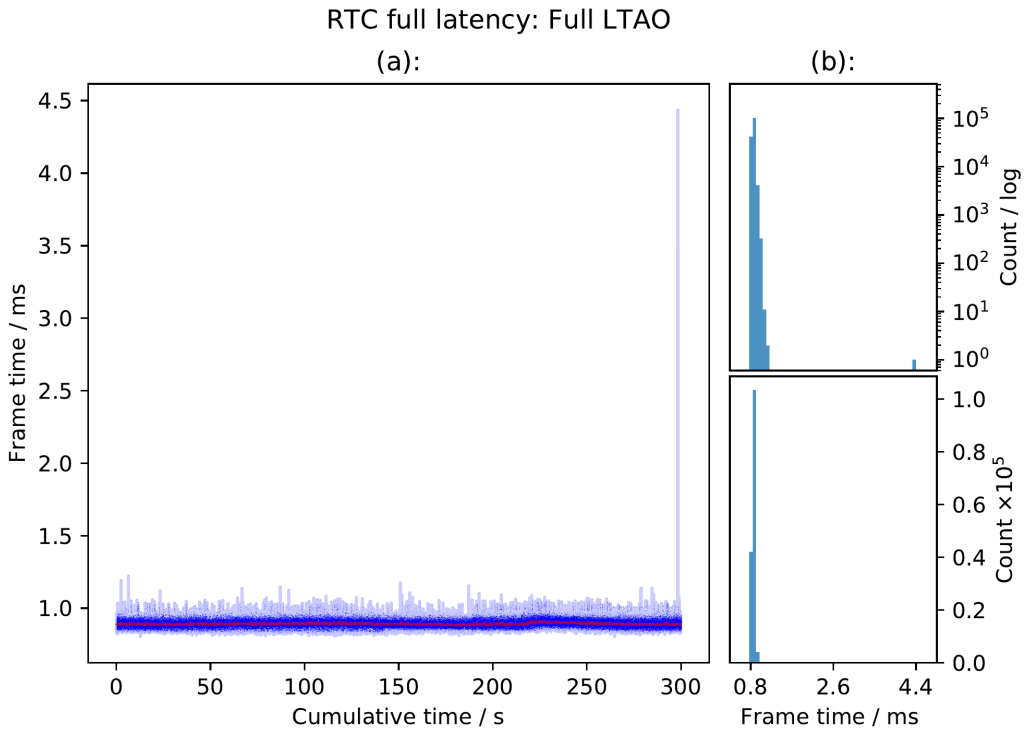


Figure 5.6: Latency results for the full LTAO setup as described in Section 5.1 and Table 5.1. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s. This is a measure of the time between when the last pixel arrives at a reconstruction node and when it receives the timing packet from the master node. The large outliers are result of delays from the CPU-based simulated cameras, which is compounded by the fact that this timing data includes delays from all 7 simulated camera streams.

receives the partial DM commands, sums them together, processes the result and once finished it sends a timing packet to the other nodes.

Figure 5.5 and Figure 5.6 show RTC latency plots for the MCAO and LTAO test cases respectively. They show the timing data from one of the reconstruction nodes of the 7 during full operation, which includes the transmission and receiving time of the DM timing packet and can therefore be considered slightly pessimistic. The ping latency has been measured to be ≈30 µs. The specifications used for each are shown in Table 5.1 and the mean latency of the MCAO case is $985 \pm 33\mu s$ and $894 \pm 29\mu s$ for the LTAO case. It can be seen that there are 2 major outliers in the latency for the MCAO test and one for the LTAO test which are a result of delays introduced from the CPU-based simulated cameras. The number of frame losses

however is acceptable considering that there is at worst one frame drop per 150 s total integration time.

As well as testing the full 8 node MCAO RTC we also tested the architecture with different numbers of LGS WFSs, and therefore reconstruction nodes, combined with the master node. Table 5.2 shows the results for the MCAO case where there are less than the full number of reconstruction nodes for the MAORY specification. Results are shown for the cases where there are 2, 3, 4, 5 and 6 LGS reconstruction nodes feeding partial DM commands to the master node. As can be seen from the results the total latency varies only slightly by the reduction in processing nodes, showing that the solution is scalable at least up until the desired number of nodes for ELT-scale MCAO.

5.2.1 Effect of streaming RTC telemetry on latency

Another very important aspect of AO RTC operation involves the streaming of telemetry during operation, either for concurrent processing so as to update the reconstruction matrices or reference centroids or purely for saving data to disk for later analysis. DARC employs circular buffers to store telemetry when requested during operation and also has the capability to read from these buffers and send the telemetry to wherever is necessary. All the timing data used in this thesis is gathered by using the telemetry streaming functionality of DARC. There are three buffers which are read for every timing measurement used in this chapter; an RTC time buffer which stores frame times, an RTC status buffer which stores various status information and and RTC DM time buffer which stores the timing data for the receipt of the timing packet from the master node.

The status buffer is used to retrieve the timestamps for when the last pixel has arrived and also the timestamp for when each node has delivered its partial DM command. The DM time buffer is populated by a process which listens for packets from the master node and takes a time stamp on arrival. A common iteration

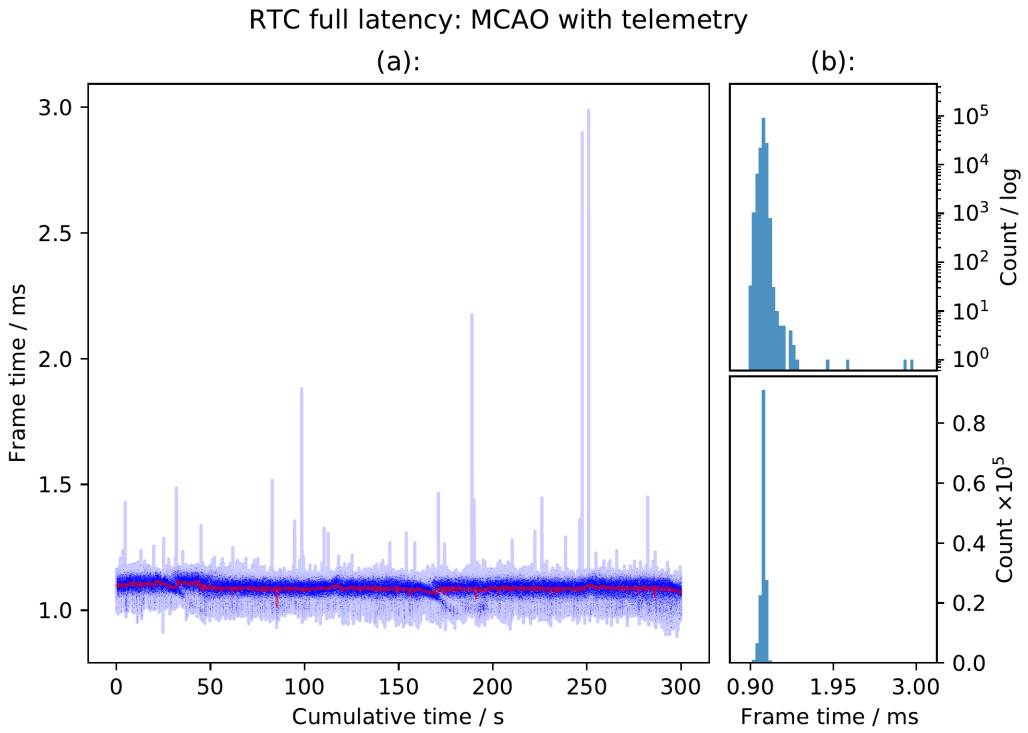


Figure 5.7: Latency results for an MCAO setup as described in Section 5.1 whilst both centroid and DM command telemetry is taken for all nodes as described in Section 5.2.1. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

number between the two buffers originating from the simulated camera is used to synchronise the data. In this way we can match up the DM command timing packet sent from the master to the image frame received from the simulated camera and calculate the full RTC latency.

Figure 5.7 shows the RTC latency for an MCAO setup for a case when slope telemetry and partial DM command telemetry are also streamed from the reconstruction nodes during operation. The mean latency is measured at $1085 \pm 32\mu s$ and there are several outliers which result from delays due to the simulated camera streams. There are also a number of relatively small outliers in this data, $< 1.5ms$, compared to the case without slope and DM telemetry which results from the taking of telemetry itself.

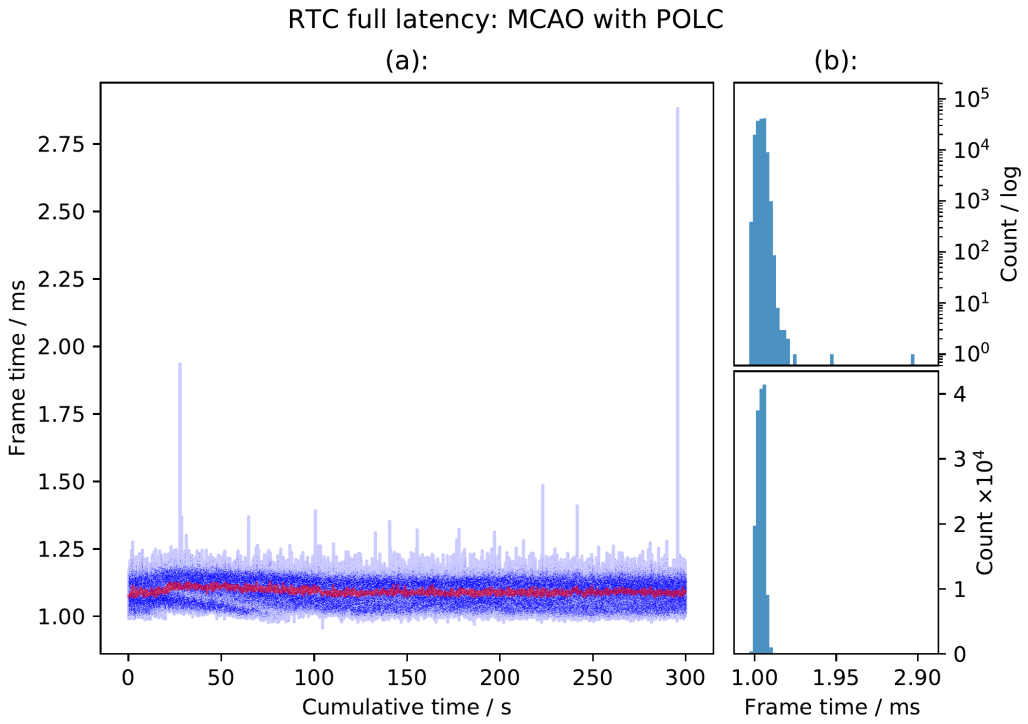


Figure 5.8: Latency results for an MCAO setup as described in Section 5.1 whilst implicit POLC is computed on the master processing node as described in Section 5.2.2. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

5.2.2 Effect of pseudo-open loop control on latency

All of the types of AO used in this report would generally be used in closed-loop operation, that is, the atmospheric wavefronts are corrected before the residual phase error is measured by the WFSs. This approach means that the wavefront phase errors measured by the WFSs are smaller than those measured in open loop and so the WFSs can be tuned for finer precision. Also, during closed-loop operation, errors in the measurement, correction, and reconstruction can be dynamically removed by the feedback of the system. The downside to closed-loop AO is that the slopes measured by the WFS no longer give a measurement of the actual atmospheric wavefront phase. The slopes measured in open-loop can be used to retrieve information about the atmospheric conditions which are required for reconstruction algorithms that take the current atmospheric statistics into account.

To get around the lack of open-loop slopes in closed-loop operation, it is possible to reconstruct pseudo-open loop (POL) (Piatrou and Gilles, 2005) slopes from the DM commands and resulting closed-loop slopes. As described in Chapter 2, for reconstruction algorithms that rely on POL control (POLC) there are two ways in which the POL slopes can be incorporated into the final reconstruction result. They can either be calculated explicitly and used directly in the algorithms or the effects of POLC can be incorporated into the final reconstruction implicitly without first calculating the actual POL slopes. The benefits of implicit POLC are massively reduced computational requirements. Explicit POLC requires the POL slopes to be computed before the wavefront reconstruction computation. However implicit POLC only needs the final DM command as calculated by the master node and so the POLC computation can be done there.

We have implemented the implicit POLC calculation for the MCAO and LTAO operation of DARC on the master node. The POLC is calculated for the next frame after the DM command is ready and so it should have minimal impact on the overall latency. The summing of partial DM commands on the master node is calculated by a single thread and so there are enough computational resources remaining to calculate the implicit POL, which is a single MVM, without affecting latency. Figure 5.8 shows the RTC latency for an MCAO setup for a case when the master node is performing POLC computation using 32 threads. The mean latency is measured at $1090 \pm 45 \mu s$ and there is a single large outlier which results from delays due to the simulated camera streams.

Chapter 6

AO RTC Performance Evaluation

Chapter 4 and Chapter 5 demonstrate the use of the Intel Xeon Phi processors and the DARC software to achieve ELT-scale AO RTC operation. This chapter will further explore the performance of AO RTC and evaluate different techniques for increasing the performance of an AO RTC. This chapter begins with a description and results of a simple simulation showing the comparison of a first order LQG reconstructor with the standard minimum variance MVM reconstructor. The chapter will then expand upon some of the results presented in the earlier chapters including results of NUMA aware SCAO (Jenkins et al., 2019), SCAO RTC on multiple processing nodes, the effect of parameter changes during operation of a SCAO and LTAO system (Jenkins et al., 2019), and concluding with a breakdown of the timing of the individual RTC sub-processes as described in Chapter 2.

These considerations must be taken into account for the deployment of the next generation ELTs due to the challenges that they pose. The LQG control presents a solution to help reduce the effects of vibrations on the telescope structure; other many-core CPU systems need to be considered due to the unavailability of future Xeon Phi processors; and updating of AO parameters during RTC operation is crucial for any real telescope deployment.

6.1 Improving the correction with optimal control

For the next generation of ELTs, by far the the most computationally demanding aspect of the real-time control is the reconstruction of the turbulent wavefront. Section 2.2.1 details the most common reconstruction method, which is also the simplest, by directly mapping the WFS measurements to DM commands by using the least squares solution of the influence equation, Eq. 2.16. This method doesn't utilise any statistics or direct measurements of the atmospheric turbulence and so is a non-optimal control solution. The LQG method described in Section 2.2.2 however is an example of optimal control which uses information about the atmosphere in the reconstruction. It also takes into account previous measurements in order to make a prediction of the atmospheric conditions when the DM command is to be applied.

However the LQG control is significantly more computationally intensive than the standard single MVM approach of the least-squares reconstruction. In practice, the LQG reconstruction requires multiple MVM operations to incorporate the atmospheric statistics and to perform the prediction step as described in Section 2.2.2. In this section a comparison of the AO correction performance as corrected Strehl ratio is presented for the two reconstruction methods shown in Section 2.2. The methods are compared by simulating the AO correction of two reconstruction methods using AO simulation software.

To begin investigation into the performance of LQG control in AO, a very simple AO simulation was created to compare the MVM control with the optimal LQG control directly, by running both methods on exactly the same simulated wavefront data. The simulation consists of a set up phase, whereby all the required matrices needed for the duration of the test are calculated, including the interaction (poke) matrix and control matrix for the MVM method and the various matrices required by LQG as described in Section 2.2.2. The simulation then enters the main AO loop where successive areas of turbulent phase are corrected by each method

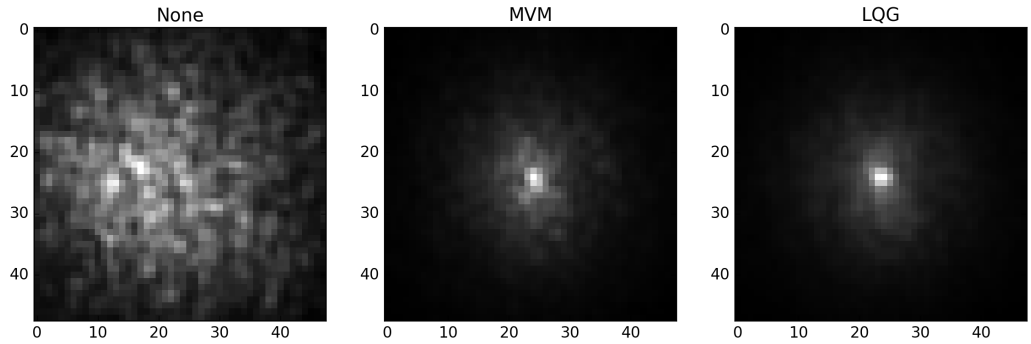


Figure 6.1: Long exposure images of 400 frames from a simple AO simulation including images for no correction (None), MVM control and LQG control. The Fried parameter of the simulated turbulence was set at 0.2m.

		Fried Parameter (m)		
		0.2	0.5	1.0
SR (%)	MVM	2.4	44.9	77.9
	LQG	2.7	37.0	70.3
	None	0.6	3.0	12.1

Table 6.1: The SRs of simulated long exposures of 400 frames for a simple SCAO AO simulation showing results for LQG control, MVM control and no control (None) for different values of the Fried parameter.

and a long-exposure image is constructed for each. The criterion for determining the correcting performance of each algorithm is the measure of the Strehl ratio of the long-exposure results, which would approximate the image attainable in a real-world situation. The simulation makes extensive use of DASP (the Durham Adaptive Optics Simulation Platform) to simplify certain aspects of the simulation such as the creation of the turbulent phase and simulating the action of the WFSs and DMs.

The SCAO simulation for both types of reconstruction was set up with the following parameters:

- Telescope diameter of 4.2m (William Herschel Telescope)
- 7×7 WFS subapertures

- 8×8 DM actuators
- 32 Zernike modes to represent the wavefront
- Kolmogorov turbulence statistics
- Wind speed of $10\,\mathrm{m\,s^{-1}}$
- Turbulence outer scale of 30 m
- Wavelength of 640 nm

Figure 6.1 shows three long-exposure images: one with no AO correction (None), one created using the least-squares MVM method and the other with the optimal LQG control method. Strehl ratios are shown in Table 6.1. The results show the mean Strehl ratios for the methods using different values of the Fried parameter.

The results shown in Figure 6.1 and Table 6.1 agree with the statement that LQG control with a first order model, such as the AR1 model used here, will be able to correct to a similar degree as the classical MVM reconstructor (Kulcsár et al., 2012). However the LQG results here are not optimised due to the extra parameters involved, which require more tuning and characterisation, compared to the simple MVM case. Therefore the performance we see is reduced. Higher order models, such as the AR2 model, are able to correct for the turbulent phase to a greater extent, which could be the target of further investigation. Due to the more computationally demanding nature of LQG control its use has been limited to smaller AO system sizes and so it would be an ideal candidate for acceleration by the Xeon Phi discussed in Section 1.2.1.1. The main computational difference between LQG and the MVM methods is the reconstruction of the DM commands for each frame in the main RTC loop: whilst the MVM method has an eponymous single matrix-vector multiply (MVM), LQG control has multiple MVMs and matrix additions to compute for each frame as described in Chapter 2. This however should still be applicable for acceleration with the Xeon Phi and other many-core CPU systems.

Due to it being a highly parallelisable process, it could be processed by multiple processing nodes to achieve the necessary performance. As demonstrated in Chapter 5, multi-node RTC computation is a viable method to process large ELT-scale AO problems sizes at the required rates. Section 6.3 presents results of processing a single SCAO WFS across two processing nodes, which could be similarly applied to the LQG control technique.

6.2 Further Investigation of the RTC software

6.2.1 Camera Simulator Performance

The ESO MUDPI camera simulator (described in Chapter 3) is unable to provide completely jitter free cameras streams, being based on CPU technology which is non-deterministic. Figure 6.2 shows two representative frame time distributions from the camera simulator whilst it was delivering 7 individual camera streams for an MCAO or LTAO setup as described in Chapter 5. For 10 random distributions similar to those shown, the number of outliers can vary from 0-11 over the 1.5×10^5 frame for 300 s of total running time, the largest outlier is never more than twice the frame time. For the periods of low jitter the RMS jitter is very low and of order 5 µs. Each of these data sets were collected after restarting the camera simulator software and the amount of jitter present in each run can vary significantly. This introduces random high latency spikes into some of the AO RTC timing data presented in this thesis. The camera induced latency spikes were minimised by waiting a small amount of time to determine if a certain run met a minimum stability criterion. Once the camera simulator was running the amount of jitter varied little and so once a stable run was found, it was used for as many RTC tests as possible.

Due to the simulator software needing to time the inter-packet delay to microsecond precision and to deliver up to 7 individual camera streams at high frame rates, the CPU system used is not an ideal candidate. The workload of the simulator is very

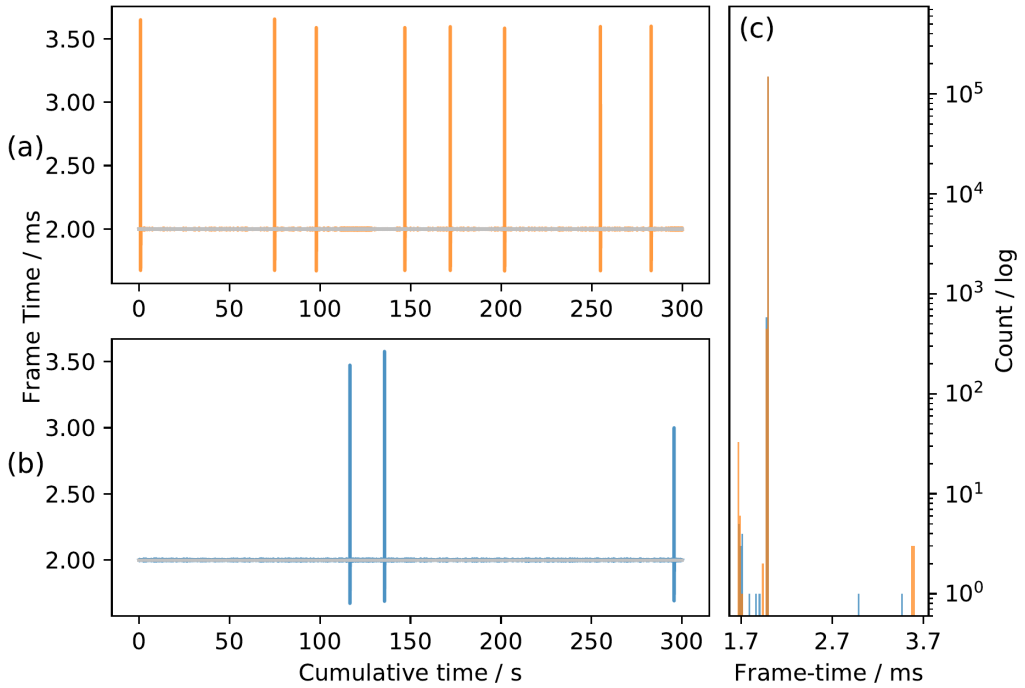


Figure 6.2: Frame time results from the camera simulator software showing two representative samples of frame time distributions while the camera simulator is delivering the 7 individual camera streams as needed by the MCAO and LTAO RTC architectures described in Section 5.1. (a) shows a frame time distribution with 8 frame drops whilst (b) shows a comparatively better distribution with only 3 frame drops. (c) shows histograms of the data. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s.

different to that of the AO RTC system and so a more modern single socket CPU system with faster cores and lower latency memory could potentially improve the camera simulator's jitter performance. We note that the jitter originating from the camera simulator would not be present in real cameras as they are usually deterministic.

6.2.2 Effect of on-the-fly changes to RTC parameters on latency

An important aspect of any real on-sky RTC is the ability to change parameters during operation, for example to update the matrix used in the reconstruction process or to update the reference centroids. DARC has the ability to set parameters through two different means. The first is a command line utility called "darcmagic"

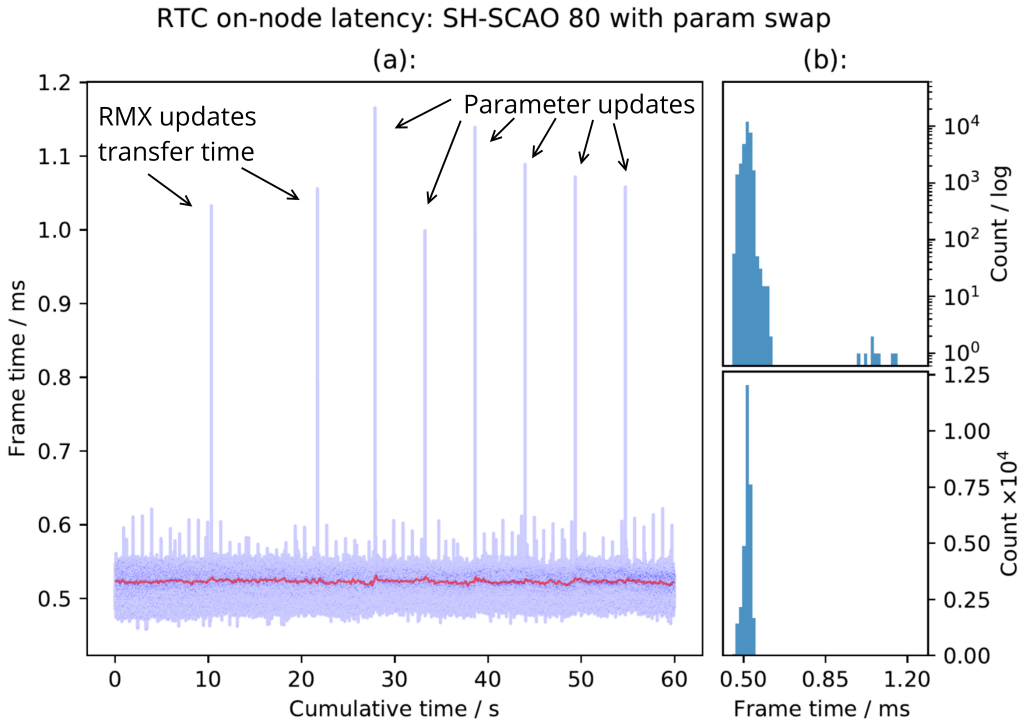


Figure 6.3: Latency results for an SCAO setup as described in Section 4.5 whilst two types of buffer swap are performed as described in Section 6.2.2. As can be seen, the buffer swap causes a $\approx 650\mu s$ spike in the latency whenever it is performed. This is for 3×10^4 iterations at 500 Hz corresponding to a total cumulative time of 60 s to highlight the 10 second transfer time of the first two parameter changes of the control matrix.

Table 6.2: Latency, RMS jitter and largest outliers results for SCAO and LTAO with parameter swap detailed in Section 6.2.2. The results are for 1.5×10^4 continuous iterations at $500Hz$ for a total time of $30s$.

AO Mode	Mean Latency (μs)	RMS Jitter (μs)	Largest Outlier (μs)
LTAO with buffer swap	1034	31	1949
SCAO with buffer swap	522	19	1165

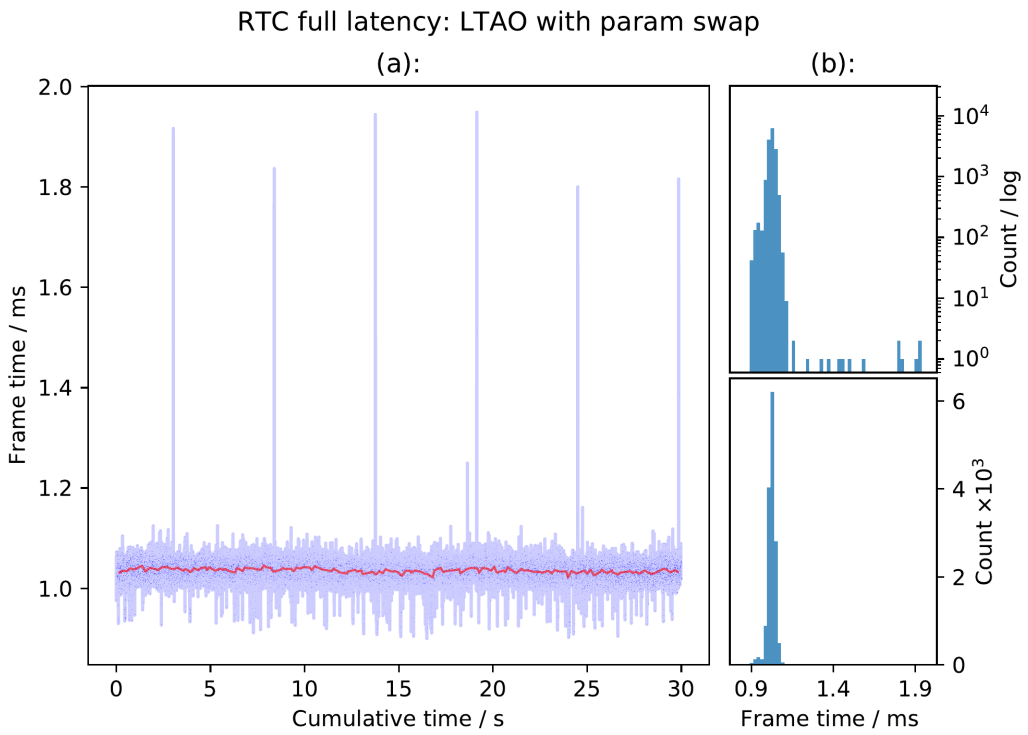


Figure 6.4: Latency results for an LTAO setup as described in Section 5.1 whilst a periodic buffer swap is performed as described in Section 6.2.2. As can be seen, the buffer swap causes a $\approx 800\mu s$ spike in the latency whenever it is performed. This is for 1.5×10^4 iterations at 500 Hz corresponding to a total cumulative time of 30 s to highlight the disturbance every 5 s.

which can change simple parameters such as string and scalar values directly from the command line and more complex parameters by loading values from configuration files. The second is a Python interface which is loaded as a module and can be used for more complex scripting of parameter updates; this is used by the "darcmagic" command line utility to interface with the running RTC.

DARC uses a double buffer approach to handle parameter switching with the buffers containing all the necessary parameters for RTC operation. One buffer is read by the RTC during operation and the second can be modified by a user to include any required new values without affecting the running processes. Once the necessary changes have been made to the second buffer, DARC is instructed to perform a buffer swap, which causes it to start reading values from the second buffer instead of the first. The process that DARC uses to handle a buffer swap involves a flag

in the main processing loop which instructs the first thread beginning a new frame that a buffer swap is required. This thread performs some checks, updates some information and then replaces the buffer pointer. Because all of the processing threads will read from the buffer this needs to be thread safe and so all other threads are temporarily blocked while the buffer is swapped.

The double buffering of the parameters reduces the effects of a parameter change by allowing the majority of the change to happen during general operation without affecting latency. However on a platform like the Xeon Phi, which excels on multi-threaded performance, the single threaded buffer swap can have a noticeable impact on the latency of the frame during the swap. Figure 6.3 shows the effect of a buffer swap on a SCAO system set up as described in Table 5.1. The first 2 latency spikes seen are due to changing the control matrix of size 5318×9232 which takes approximately 10 seconds for the transfer to happen. The other latency spikes are due to a periodic change of a single value parameter every five seconds. The size of the latency spikes shows that for different size parameters the amount of jitter introduced is the same and the spike only occurs once the internal RTC buffer is actually swapped.

Figure 6.4 shows the latency for an LTAO type system set up as described in Table 5.1 whilst a buffer swap is set to occur on one of the reconstruction nodes every five seconds. There is a clear impact on the latency which corresponds to a $\approx$800 µs spike to latency when the buffer swap occurs. Here the relative size of the latency spikes is increased from the SCAO case above as this is the full LTAO RTC system as described in Section 5.1. This essentially results in a frame drop for every buffer swap due to the average latency being 1034 µs for this particular case.

The relatively large effect of a buffer swap on the latency of the LTAO system is partly caused by the fact that the Xeon Phi has poor single threaded performance and partly because of the need for all reconstruction nodes to be synchronised by the master node, compounding any adverse effects of the swap. The parameter

change performed here was for a single valued scalar parameter, however due to the double buffered approach of DARC, changing more complex parameters such as arrays or matrices shouldn't have any more impact on the latency, as all copying of data can occur concurrently with RTC operations. The rate of data transfer can be reduced by copying the data in small batches so as to have little effect on the memory bandwidth of the system.

A future improvement for DARC would involve introducing more robust methods to change parameters. This would involve a facility to only update a single parameter without causing a full buffer swap which would reduce the size of the jitter event as only the new parameter would need to be checked for validity before RTC operations can continue. Another important feature update would be the ability to perform the buffer swap at a given future frame number. This would be necessary for properly synchronising the update on multi-node AO systems such as MCAO and LTAO to ensure that all processing nodes swap buffers simultaneously. The new parameter would be copied to each node and only after the specified frame number is reached would the parameter be used in the AO calculations.

The DARC RTC software also has a buffer interface which can continuously update parameters, and in particular array parameters, at the beginning of each frame without a buffer swap. This could be utilised to achieve the required functionality mentioned above.

6.3 Multi-node Xeon Phi SCAO

The results presented in Chapter 4 show that the Intel Xeon Phi is capable of processing an ELT-scale WFS at up to 800 Hz. However due the CPU's relatively poor single threaded performance it is not capable of processing the large data streams at a greater rate. It has been suggested that for the ESO ELT, a goal for WFS frame rate for SCAO operation is 1 kHz, which has not yet been possible on a single Xeon Phi node. Here we describe a method for processing a single ELT-scale

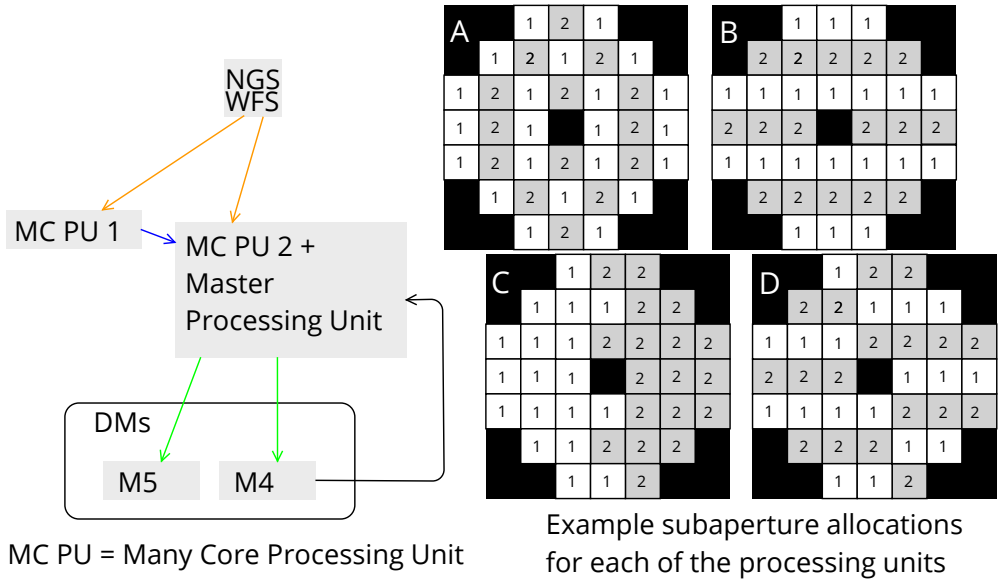


Figure 6.5: The architecture of the multi-node SCAO system setup as described in Section 6.3, this is an example for the ESO ELT. The WFS is multicast to the two processing units each of which process half of the total WFS subapertures, the master processing unit receives the partial DM vectors before combining them and delivering them to the DMs.

SCAO WFS across two separate Xeon Phi processing nodes, followed by results of the investigation.

As the camera simulator uses the UDP protocol to transfer the image frame packets, it is straightforward to specify a multicast address for the destination of the images. Multicast addresses fall within a set range within a network's available IP addresses and, if supported by the networking hardware, allow devices to subscribe to a group specified by the address and receive any valid packets sent to that address. The processing of multicast packets is usually performed within a network switch with negligible impact on packet latency and jitter. It is therefore possible to have two or more processing nodes subscribe to a multicast address and receive a single simulated pixel stream simultaneously. The DARC software can then be configured such that each node will process a unique set of subapertures and distribute the processing load accordingly. This demonstrates the flexibility of using a CPU-based AO RTC.

The configuration of the multi-node SCAO RTC is similar to that of the MCAO

and LTAO architecture in Chapter 5 and is shown in Figure 6.5. It uses the same master node concept as the other architectures, however now the master node is also one of the processing nodes to reduce the amount of data transfers required. The combined reconstruction and master processing node operates two separate instances of the DARC software, one of which is configured similarly to the other reconstruction node and the other configured to receive the partial DM vectors and combine them to form the final command vector.

The reconstruction nodes are configured such that each one processes alternating subapertures in the subaperture rows, as shown in layout A in Figure 6.5. This layout of the subaperture allocation per node was chosen as entire rows of subapertures become available to process simultaneously due to the pipelining of pixels from the camera simulator. Alternative layouts were considered, such as each node processing entire alternating rows of subapertures as shown in layout B in Figure 6.5. However the subaperture allocation shown in layout A in Figure 6.5 delivered the best performance. The layout which splits each row exactly in half down the centre, with one node processing the first half and the other the second half as shown in layout C in Figure 6.5, was expected to provide better performance than the other layout, however it was found to be unable to deliver similar latencies as layout A. This is because the striding of subapertures within a row limits cache misses in the process of copying each subaperture into contiguous memory.

Figure 6.7 and Figure 6.8 show the full latency distributions for the multi-node SCAO architecture operating at 500Hz and 700Hz respectively. This latency is measured between the last pixel arriving at the combined reconstruction and master node the DM command becoming ready. Figure 6.6 shows the on-node latency of one of the reconstruction nodes, which includes the time taken from receiving the last pixel to delivering the partial DM vector to the master node. The latency and jitter values for these results, as well as for other frame rates can be seen in Table 6.3.

In this configuration the multi-node system was able to process the simulated cam-

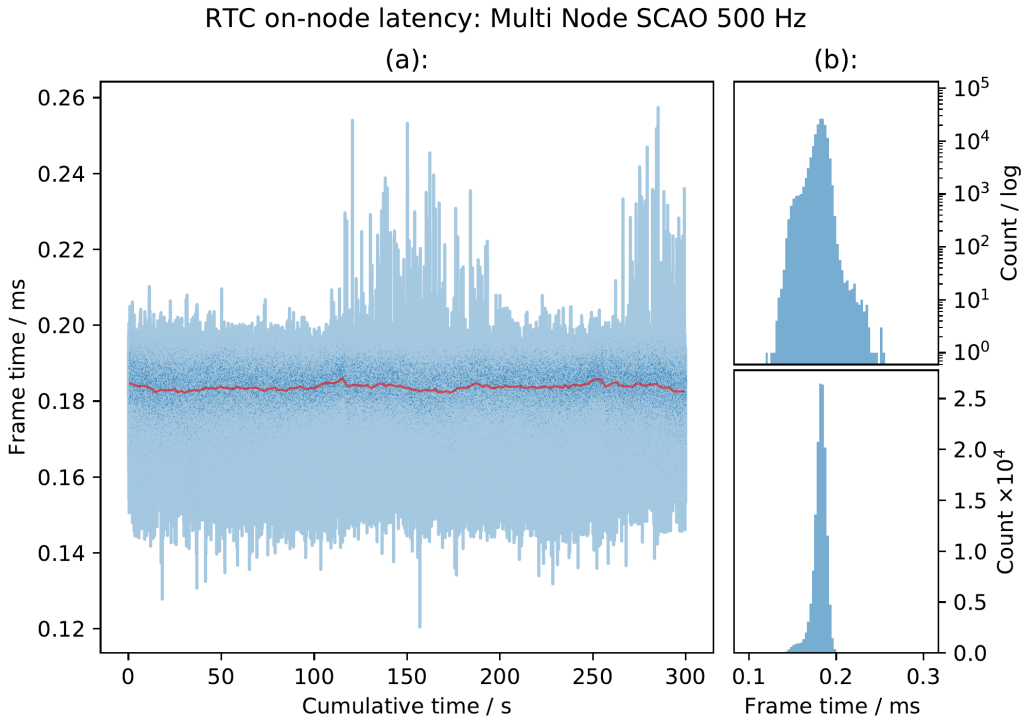


Figure 6.6: On-node latency results for the multi-node SCAO setup as described in Section 6.3 operating at 500Hz. This latency is for the processing of half the total subapertures for the WFS and includes the time between when the last pixel arrives and until the partial DM vector is ready. These results are for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

Frame Rate	Latency (μs) ± Jitter (RMS)		
	On node	Master node	Single node
500	183 ± 8	376 ± 19	348 ± 12
600	186 ± 6	384 ± 16	
700	186 ± 6	380 ± 20	406 ± 47
800	187 ± 6	383 ± 20	
900	185 ± 6	380 ± 26	
936	184 ± 6	442 ± 84	

Table 6.3: Full latency and on-node latency results for the multi-node SCAO setup described in Section 6.3 for different WFS frame rates. The on-node latency includes the time take to process half of the total WFS subapertures from last pixel arriving to delivery of the partial DM vector. The full latency includes the time taken to process all WFS subapertures from arrival of the last pixels to when the full DM vector is ready.

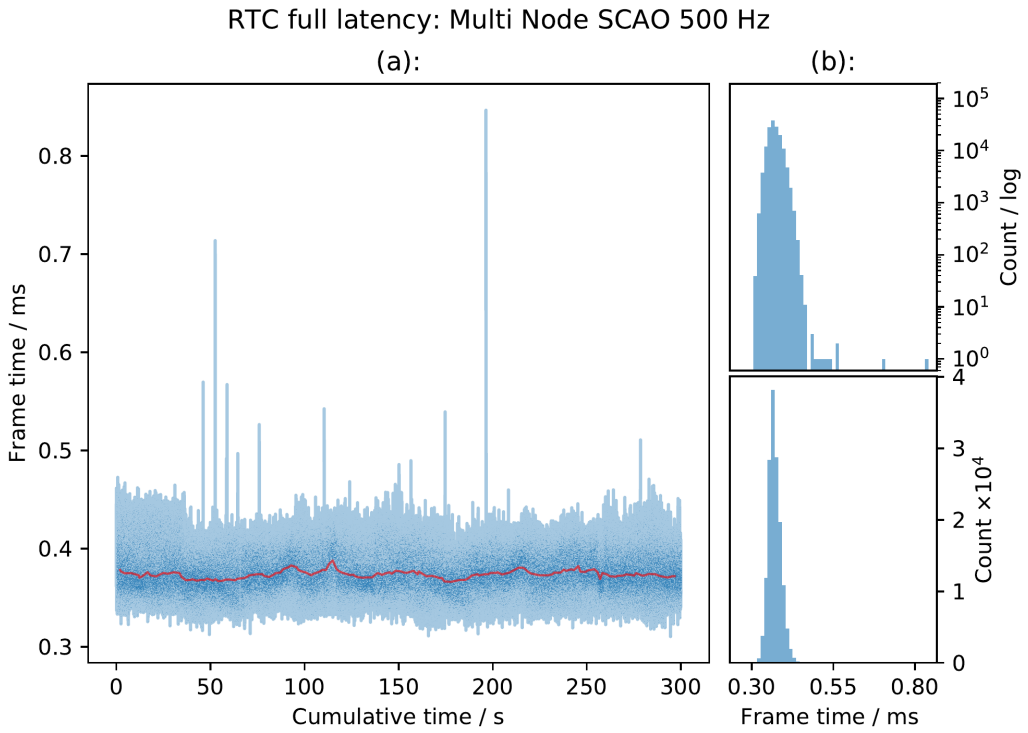


Figure 6.7: Full latency results for the multi-node SCAO setup as described in Section 6.3 operating at 500Hz. This latency is for the processing of all subapertures for the WFS and includes the time between when the last pixel arrives and until the final DM vector is ready. These results are for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

era stream at its maximum achievable frame rate of 936 Hz, though with an increased latency and jitter compared to the slower frame rates. The full latency achievable with the multi node configuration is similar to the latency achievable with a single node, as shown in Chapter 4 and Table 6.3. However it does enable processing of WFSs operating at increased frame rates compared to a single node, which was able to achieve a maximum WFS frame rate of 750 Hz. This demonstration proves that multi-node SCAO is possible with a similar software architecture used for MCAO and LTAO and for observatories and instruments that use an MCAO or LTAO mode, the necessary processing hardware will be readily available for the multi-node SCAO.

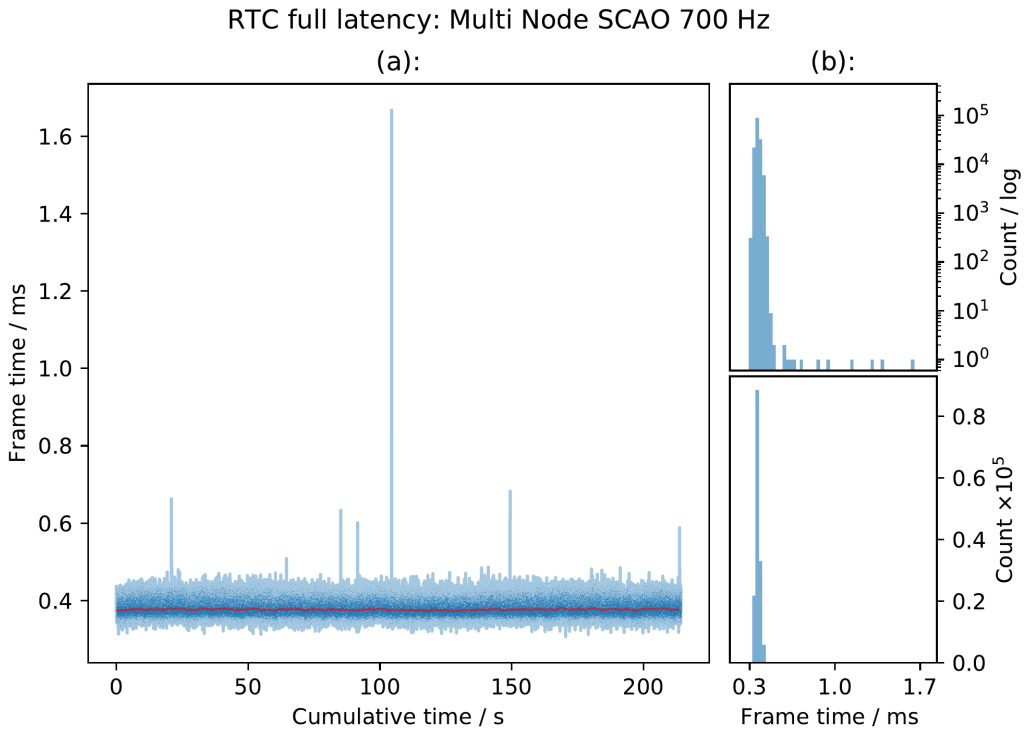


Figure 6.8: Full latency results for the multi-node SCAO setup as described in Section 6.3 operating at 700 Hz. This latency is for the processing of all subapertures for the WFS and includes the time between when the last pixel arrives and until the final DM vector is ready. These results are for 1.5×10^5 iterations at 700 Hz corresponding to a total cumulative time of 214.3 s.

6.4 Other many-core CPU systems

6.4.1 NUMA-aware DARC

Most of the results presented in this report are obtained from Intel Xeon Phi CPU systems as described in Jenkins et al. (2018b). However the Xeon Phi platform has been discontinued and so it is unlikely to be considered as a candidate for real AO RTC hardware. One of the main reasons for choosing a CPU-based RTC is that the software and the optimisations made for many-core operation are not specific to a single CPU architecture or vendor. This thesis has demonstrated the flexibility of a CPU-based AO RTC system, with many different AO system types profiled. Here we investigate RTC performance on different hardware platforms.

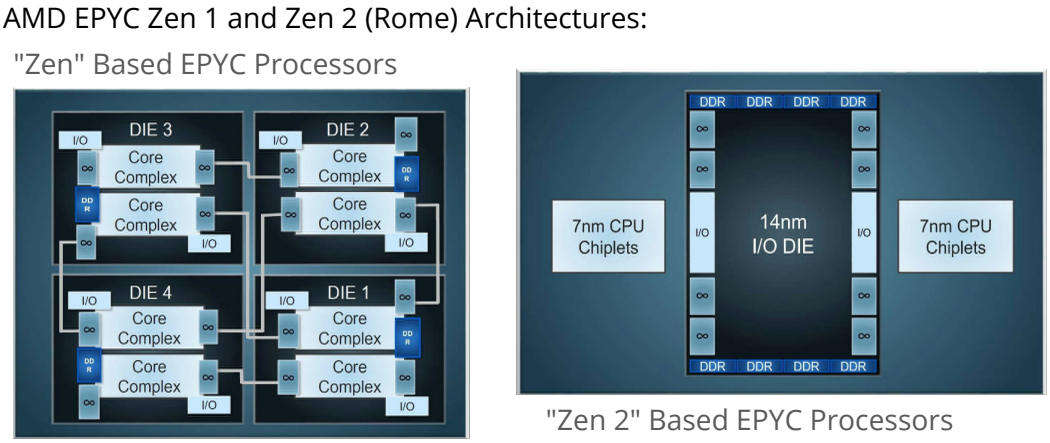


Figure 6.9: The AMD Zen and Zen 2 EPYC architectures showing the NUMA node topology, each of the core complexes is made up of two CPU cores sharing a single memory channel.

The main advantage the Xeon Phi has over traditional CPU systems is the large memory bandwidth of the MCDRAM which can be accessed by all CPU cores equally. For other CPU systems the memory bandwidth of a single CPU package ($<200\,\mathrm{GB\,s^{-1}}$) is much less than that available to the Xeon Phi ($480\,\mathrm{GB\,s^{-1}}$). However it is possible to increase the memory bandwidth of traditional CPU systems by using multiple CPU sockets per system and utilising the NUMA properties of the system. With software that takes into account the NUMA architecture it is possible to multiply the memory bandwidth of each socket by the number of sockets in the system as described in Chapter 1.

6.4.1.1 AMD EPYC: NUMA-aware DARC with pipelining

Here we consider DARC running on an AMD EPYC 7351 dual socket system using the camera simulator as described in Chapter 3 to deliver pipelined pixels for an ELT-scale SCAO system configuration. Figure 6.10 shows frame time and latency results which can be compared directly to the Xeon Phi results presented in Chapter 4. There is 300 s of continuous measurements corresponding to 1.5×10^5 frames at a framerate of 500 Hz and the average latency is measured at $616 \pm 17\,\mu\mathrm{s}$.

The source code and RTC parameters are identical for the two different CPUs with

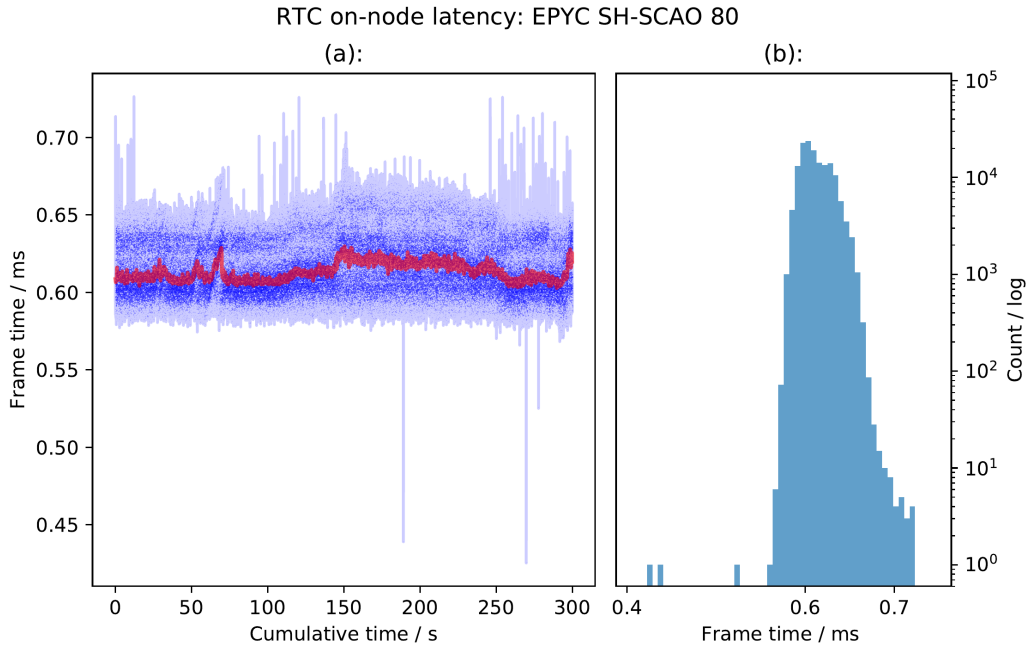


Figure 6.10: Frame time results for an SCAO setup using a SH-WFS type WFS slope calculation with 80 subapertures across the pupil for an AMD EPYC system as described in Section 6.4. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is $616 \pm 17 \mu s$.

the only differences coming from the compiler options (no AVX512 instructions for the EPYC) and the configuration of the threading and NUMA aware memory allocation for the two different platforms. The EPYC processor used for these results has 16 cores and 64GB of DDR4 2667 MHz memory per socket. The core topology of the EPYC CPUs is such that each CPU has four NUMA regions with each region having 4 CPU cores and 16 GB of memory each, as shown in Figure 6.9. The RTC software uses the NUMA information of the CPU to allocate memory for the RTC control matrix on the nodes relevant to each CPU core. The Linux Kernel and OS is tuned in a similar way to the Xeon Phi, with the major differences being the OS itself (Ubuntu 16.04 for EPYC vs. CentOS for the Xeon Phi) and that simultaneous multi-threading (Hyper-Threading) is turned on for the EPYC system as it provides better performance and allows 8 threads per NUMA node.

The maximum theoretical memory bandwidth of the EPYC system is 341 GB s^{-1}. Using the STREAM benchmark the maximum attainable memory bandwidth was

measured at 200 GB s^{-1} using a NUMA-aware version of the STREAM benchmark. This is 58 % of the theoretical maximum, which is determined by considering the 8 DDR4 memory channels per socket running at 2667 MHz. This is less than half of the measured memory bandwidth of 480 GB s^{-1} of the Xeon Phi 7250. It is therefore expected that the performance of the EPYC will be less than that of the Xeon Phi for the memory bandwidth bound RTC operations. However these results show that the software can be readily used on different CPU platforms and that performance is as expected based on the knowledge that the main RTC operations are memory bandwidth bound.

Other multi-socket CPU systems would also be suitable for ELT-scale AO RTC such as the Intel Xeon Scalable processors, which in a quad socket configuration can provide comparable maximum memory bandwidth to the Xeon Phi when the NUMA regions are taken into account. Multi-socket systems also benefit from generally running at a higher base CPU frequency than the Xeon Phi and so their single threaded performance is better. A dual-socket EPYC system with the required memory bandwidth can be purchased for a similar price to the Xeon Phi, making it the most likely substitute. For the Intel Xeon Scalable processors, due to their reduced memory channels per socket, a quad socket system would be required to match the memory bandwidth and this can increase the per node costs to over 4× that of comparable EPYC or Xeon Phi systems as shown in Table 3.1.

Next generation AMD EPYC processors will introduce a new architecture (Papermaster, 2018) that simplifies the core topology of the system and will be built on a smaller 7nm process node to provide better energy efficiency. This involves introducing a 9-die architecture which includes 8 compute chiplets and a single I/O interface die such that each CPU core can access all memory channels equally. This is different to the current design where each of the 4 NUMA regions has 8 cores and 2 memory channels each and so only those 8 cores can access the full bandwidth of those 2 channels. The new architecture will reduce the relative complexity of NUMA memory management and allow more efficient interleaving of memory over

all 8 memory channels, which will reduce the latency of the EPYC results shown in Figure 6.10, as each CPU socket will be a single NUMA node. The memory for these next-generation processors will likely be clocked faster, at up to 3200 MHz compared to the current maximum of 2667 MHz, increasing memory bandwidth to a theoretical 410 GB s^{-1} for a dual socket system.

6.5 Latency Contribution of RTC Processes

Here is presented an analysis of the contribution to the overall RTC latency from each of the 5 RTC sub-processes described in Section 2.1. As described in Section 6.1, the wavefront reconstruction is the most time consuming process in an ELT-scale AO RTC. However for the large WFS image frames needed for ELT-scale SH-WFSs, the image calibration and centroiding steps can also have a significant contribution to the overall latency. With a pipelined camera stream, a large portion of the pixel processing latency can be reduced but due to their inherent serial nature, it is difficult to efficiently accelerate these processes with vectorisation and multi-threading.

The results presented in this section compare the computational latency of different CPU systems for the RTC processes described in Chapter 2, and where necessary use NUMA-aware DARC operation. Due to the unavailability of the camera simulator for all systems, the results show the computation performance without pixel pipelining. The computational latency refers to the time needed to complete the processes described in Section 2.1. To achieve the best computational latency on each platform the subapertures were allocated to each processing thread equally, similar to Option 3 in Figure 3.7. However each thread can begin processing simultaneously due to the lack of pixel pipelining.

The computational latencies were measured for each hardware platform with the AO systems parameters for the ELT-scale SCAO system as described in Chapter 4. The latencies were measured by taking timestamps at the beginning and end of

processing for each of the individual RTC processes. All the timing data was gathered using the `clock_gettime` POSIX function call and collected through the standard DARC status buffer telemetry. Data was gathered for the Xeon Phi 7210 and 7250 systems, the Intel Xeon Platinum 8180, the Intel Xeon Gold 5120 and the AMD EPYC 7351 as described in Chapter 1. The Intel Platinum system is a 4-socket system and so data was gathered for the full 4-socket configuration and also for a 2-socket configuration to compare with the 2-socket Intel Gold and AMD EPYC systems.

The number of processing threads has varied across the different systems in an attempt to achieve the best performance on each one, with the number used for each shown in Table 6.4. Due to the large number of processing threads for each system, it was difficult to accurately and consistently make timing measurements for each of the RTC processes without adding significant overhead and increasing the latency. Therefore the timing results for the processes are not exact due to each thread operating asynchronously on unique sets of subapertures.

Figure 6.11 shows a comparison of the latency measurements for the different hardware systems. It includes the time taken to complete the five processes of calibration, centroiding, reconstruction, preparing the DM command and then sending the DM command via UDP network transfer. Table 6.4 shows the values for when each of the five process are completed as well as RMS jitter values. Each timing measurement is the median value from 1.5×10^5 frames. Figure 6.12a shows the latency distributions for the completion time for each of the five process for the Intel Xeon Phi 7250 system. Figure 6.12b shows similar results for the Intel Platinum 8180 quad socket configuration.

The results in this section show that CPU systems other than the Xeon Phi are capable of delivering ELT-scale AO RTC performance. A quad socket Intel Platinum system can complete the RTC computation in less than half the time of Xeon Phi whilst the dual socket Intel Platinum configuration is also capable of increased performance compared to the Xeon Phis. The AMD EPYC and Intel Gold dual

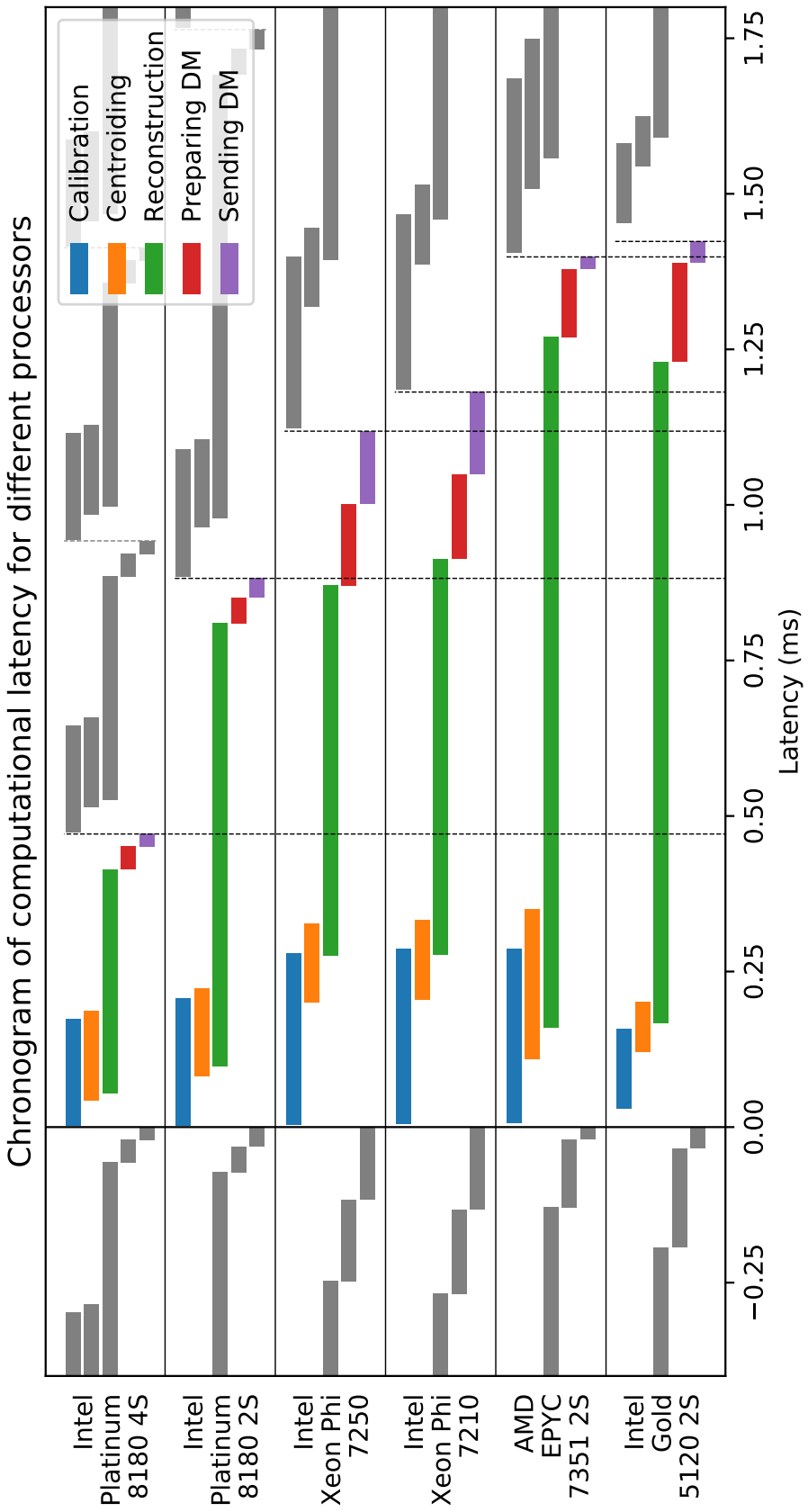


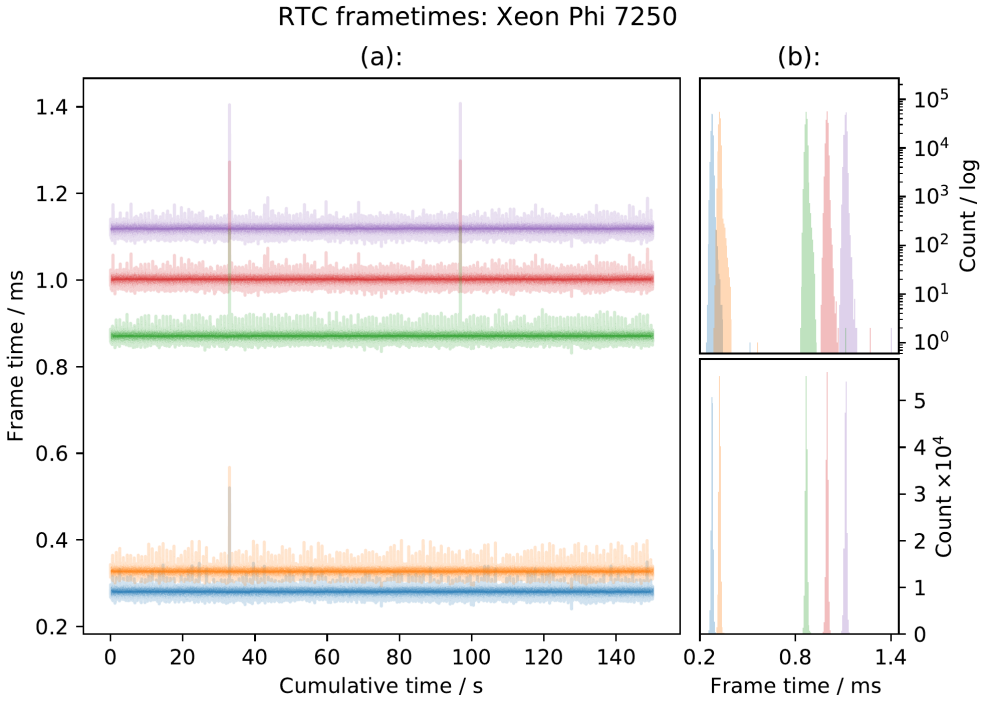
Figure 6.11: A comparison of the computational latency for different hardware platforms. The processors are described in Chapter 3 and the latencies shown are for the different computational processes described in Chapter 2.

Hardware Platform	No. of Threads	Time from start-of-frame to completion of each process				
		Calibration	*Centroiding*	*Reconstruction*	*Preparing DM*	*Sending DM*
Intel Xeon Platinum 4S	48	173 ± 9	186 ± 9	414 ± 11	450 ± 12	471 ± 12
Intel Xeon Platinum 2S	48	207 ± 9	223 ± 9	810 ± 8	851 ± 8	882 ± 9
Intel Xeon Phi 7250	54	280 ± 6	327 ± 7	871 ± 7	1001 ± 7	1118 ± 7
Intel Xeon Phi 7210	54	286 ± 6	333 ± 7	914 ± 8	1049 ± 8	1181 ± 8
AMD EPYC 7351 2S	60	287 ± 11	350 ± 23	1270 ± 18	1379 ± 18	1399 ± 18
Intel Xeon Gold 2S	48	158 ± 05	200 ± 06	1230 ± 08	1389 ± 15	1423 ± 15

Table 6.4: The median values of the latency measurements and the RMS jitter for the computational latency results presented in Section 6.5. This shows the latency from start-of-frame until completion of each of the RTC processes for the different hardware platforms. Figure 6.11 illustrates these results.

socket systems are not quite able to match the performance of the Xeon Phis, however their much lower cost compared to the Intel Platinum keep them cost competitive without sacrificing too much latency. These results show that even though the Xeon Phi processors have been discontinued there is still CPU-based hardware available to process ELT-scale AO RTC.

a Intel Xeon Phi 7250



b Intel Xeon Platinum 8180 4S

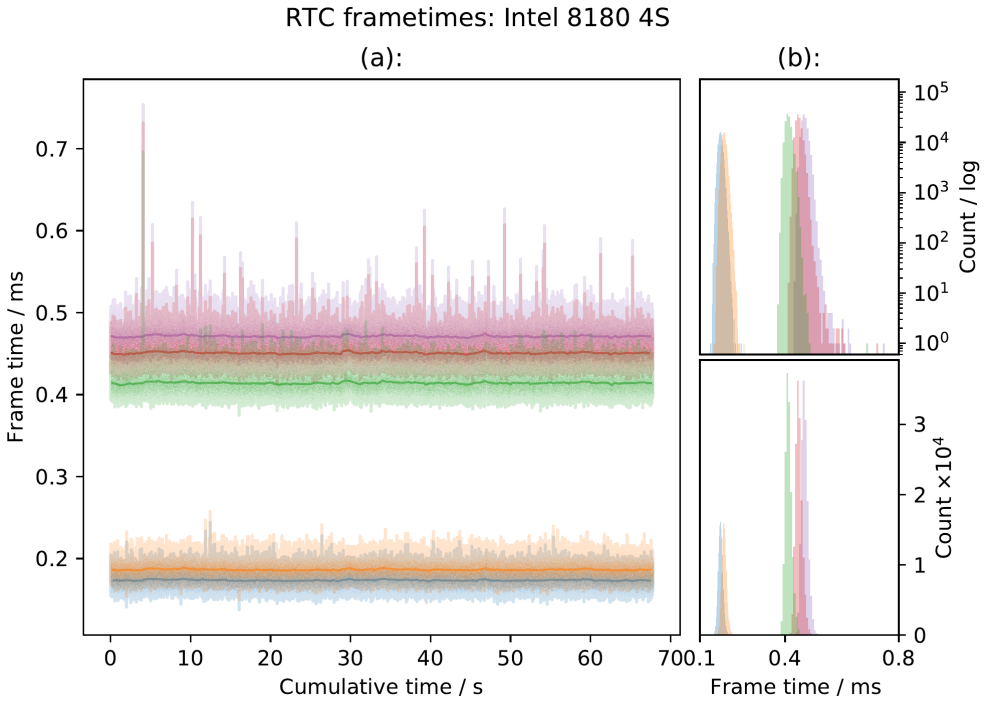


Figure 6.12: Computational latency results for the individual subprocess of DARC for both the Intel Xeon Phi 7250 (top) and the Intel Xeon Platinum 8180 quad socket system (bottom). This is using a SCAO configuration with a SH-WFS type WFS slope calculation with 80 subapertures across the pupil. Each distribution corresponds to the time from start-of-frame to the completion of each of the five RTC processes shown in Figure 6.11 and median latencies for each process are presented in Table 6.4. Results shown are for 1.5×10^5 iterations. Colours correspond to the legend shown in Figure 6.11.

Chapter 7

Conclusions and Future work

7.1 The Challenges of ELT-scale AO RTC

The next generation of ELT-scale telescopes brings with it many challenges in the design and implementation of AO systems. All three of the planned ELTs will use AO as an integral tool for their successful operation, and, due to the scaling of the AO real-time control problem size with telescope diameter, the computational requirements are greatly increased compared to currently operating AO RTC systems. Further considerations such as the size of corrected field of view required for observation, the number of science targets to be observed simultaneously, and the increase in dynamic aberrations due to telescope wind shake will dictate the AO system types needed and the wavefront reconstruction techniques to be employed.

This thesis presents an investigation into a many-core CPU-based AO RTC architecture to achieve the computational performance necessary to help mitigate the increased effects of the atmosphere on ELT-scale astronomical observations.

7.2 Many-core CPUs with the DARC AO RTC

The Intel Xeon Phi Knights Landing CPU was investigated for the processing of ELT-scale AO RTC. It has up to 72 CPU cores with 16GB of high bandwidth

memory, which is ideal for accelerating the highly parallelisable and vectorisable wavefront reconstruction problem. The DARC RTC software is a mature on-sky tested modular application that has been optimised and adapted for many-core CPUs and for ELT-scale AO RTC. It is written in the standard C and PYTHON programming languages. The optimisation process was not specific to any particular type of CPU system and did not involve any non-standard programming practices, but used only widely available tools and libraries. A UDP based networked camera simulator was developed to deliver simulated pipe-lined WFS images to the CPU-based DARC RTC. This is written in the C programming language and used only general Linux libraries and commands to achieve its functionality.

7.2.1 ELT-scale SCAO RTC

The results of testing the DARC AO RTC software for SCAO systems and utilising the CPU camera simulator is presented in Chapter 4. The results show that a single Intel Xeon Phi Knights Landing processing node is able to compute the AO RTC for ELT-scale SCAO systems with latencies as low as $348 \pm 12\,\mu s$ for a WFS frame rate of 500 Hz and has the ability to process WFS images delivered at up to 750 Hz. The RTC latency was measured for the processing of both a SH-WFS and a Pyr-WFS configuration, with an investigation of how the latency for each scales with the dimensions of the WFSs. Different subaperture allocation schemes were investigated, with the most efficient for pipe-lined WFS operation being an unequal subaperture allocation. The effect of POLC computation for the SCAO case was explored, involving both explicit and implicit POLC methods. It was found that both methods reduce the maximum framerate that the RTC could achieve on the given hardware, however the implicit POLC had much less of an effect on the maximum achievable performance.

7.2.2 ELT-scale MCAO and LTAO RTC

An architecture for the processing of ELT-scale MCAO and LTAO system types was presented in Chapter 5. This scales the DARC RTC across multiple processing nodes by processing each of the MCAO or LTAO WFSs independently in the same way as the SCAO RTC computation is performed in Chapter 4. The partial results of the wavefront reconstruction for each WFS can then be combined by a master processing node to acquire the final DM command vector for the entire system. This shows the flexibility of the CPU-based DARC RTC software; with no modifications to the base RTC software and only the implementation of different software modules and configuration the RTC software can achieve the performance required for ELT-scale MCAO and LTAO operation. For a cluster of seven Xeon Phi Knights Landing processing nodes, an ELT MAORY-like MCAO RTC system was capable of achieving RTC latencies of as low as $985 \pm 33\,\mu s$; for operation with a simulated camera at a WFS frame rate of 500 Hz. Similarly, for an ELT HARMONI-like LTAO RTC configuration, results of $894 \pm 29\,\mu s$ were achieved.

7.2.3 Considerations for ELT-scale AO Operation

Chapter 6 presents results of the investigation into other considerations for ELT-scale AO, such as different wavefront reconstruction techniques that can help mitigate telescope induced vibrations and the use of other many-core CPU processors for the computation of the AO RTC. A comparison is made between the standard MVM wavefront reconstruction and the optimal LQG control method. The architecture for a multi-node SCAO RTC configuration, whereby a single WFS is processed by two CPU nodes, is presented. This technique can be used to reduce the latency of a SCAO RTC configuration compared to the results presented in Chapter 4 and allow WFSs frame rates of up to 966 Hz. This multi-node approach could also be used to achieve the required latencies with the more computationally demanding reconstruction techniques such as LQG control, building upon the

flexibility of the CPU-based DARC RTC demonstrated in Chapter 5.

Also presented are results similar to those discussed in Chapter 4, for an ELT-scale SCAO configuration, but by using an AMD EPYC dual-socket CPU system for the processing of the RTC. This demonstrates that the software and techniques are applicable to many-core CPU systems other than the Intel Xeon Phi. The AMD EPYC CPU system demonstrated an RTC latency of $616 \pm 17\,\mu s$ at a WFS frame rate of 500 Hz. These results are consistent with the difference in specification between the Xeon Phi Knights Landing and EPYC CPU; specifically the memory bandwidth of each. Finally Chapter 6 concludes with an investigation of the computational performance of the DARC RTC software on different CPU systems. This demonstrates again the flexibility of the software being used with different processor architectures and also the scaling of performance due to the different specifications of each system.

Table 7.1 shows the cost of each of the available hardware technologies including many-core CPU systems, this is representative of future trends for each manufacturer. As seen the AMD EPYC CPU systems are by far the most cost effective option for scaling the hardware for ELT-scale operation.

7.3 Future work

To complete a full investigation into all the AO RTC system types proposed for the next generation ELTs, an architecture for a MOAO RTC needs to be considered. Due to the flexibility of CPU-based RTC, this could be accomplished with an architecture similar to that shown in Figure 7.1; proposed by Basden et al. (2019). Due to the number of DMs required for MOAO, each DM would use a single processing node to complete the wavefront reconstruction along its line of sight. A subset of the DM processing nodes are used to process the WFSs to obtain the wavefront slope values which are then distributed amongst all the DM processing nodes allowing them to complete the full wavefront reconstruction.

Processor Type	Representative Example	Computational Performance (SP TFLOPS)	Memory Bandwidth (GBs^{-1}) T	M	Nodes Required (6×WFS)	Price per unit (USD)
GPU	NVIDIA V100	14.9	900	-	3	10,600
Xeon Phi	KNL 7250	5.2	480[1]	432	6	3,400
Xeon Phi	KNL 7210	4.5	450[1]	385	6	3,400
Intel CPU	Platinum 8180 ×4	22.9	512	362	6	40,000[2]
Intel CPU	Platinum 8180 ×2	11.5	256	182	12	20,000[3]
Intel CPU	Gold 5120 ×2	2.3	230	139	8	3,100[3]
AMD CPU	EPYC 7351 ×2	3.0	341	294	6	2,500[3]
FPGA	Intel Stratix 10 MX2100	6.3	512	-	6	~14,000[4]

Table 7.1: A comparison of computational performance, theoretical and measured memory bandwidth and nodes required for ELT-scale AO, for the different hardware types available for AO RTC. The T and M columns for memory bandwidth refer to theoretical and measured respectively. DSPs are not included as it is difficult to find specifications and in general their computational and memory bandwidth performance are far behind the other processor types.

[1]Measured using starboard STREAM, no theoretical available. [2]Price is for the four CPUs. [3]Price is for the two CPUs. [4]Price is for a development kit with $256GBs^{-1}$ memory bandwidth.

Future upgrades to the DARC software would involve the improved parameter updating methods as described in Chapter 5, especially the functionality to enable synchronous parameter switching amongst all reconstruction nodes in multi-node RTC configurations.

Further exploration of the available many-core CPU systems would be useful to determine their performance in the multi-node RTC configurations. The Xeon Phi range of processors has now been discontinued and so other systems would be required for the actual deployment of ELT-scale AO RTC systems.

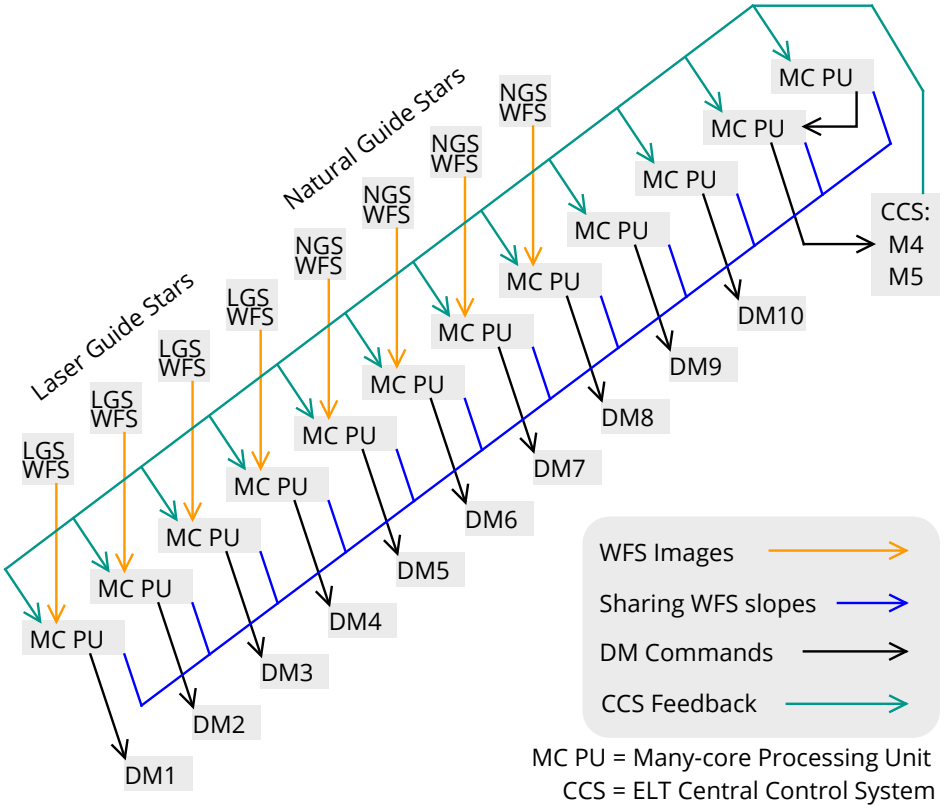


Figure 7.1: (a) A possible ELT MOAO real-time control system architecture based on 12 CPU nodes. The dark blue lines represent the multicast of slope measurements between nodes. The green lines represent distribution of CCS feedback when implicit POL is used; the CCS is the ELT Central Control System which controls the telescope mirrors M4 and M5 among other functions. The orange lines represent WFS pixel flow and the black lines represent DM command flow. (Basden et al., 2019).

7.3.1 Future Developments

With the next generation of ELT-scale telescopes on the horizon, it is a very interesting time in adaptive optics with ongoing developments in both general AO instrumentation and in the RTC domain. Due to the results achieved in this thesis, developing an RTC for ELT-scale SCAO, GLAO, LTAO or MCAO operation can now be considered an engineering challenge rather than a research challenge; the performance necessary for ELT-scale operation has been demonstrated with open-source software running on commercial off-the-shelf (COTS) CPU hardware. The greater challenges involved with MOAO and ExAO require further research to de-

termine a suitable architecture for ELT-scale operation. The ExAO problem might be better suited to a GPU based solution rather than a CPU based one if the data interconnect problem can be solved reliably using a COTS solution. GPUs have increased computational performance compared to CPUs which may be necessary to efficiently process the high order ExAO systems at the necessary frame rates, typically > 3 kHz, but are disadvantaged as they are unable to behave as a host processor.

In the near future it will be interesting to see the developments of many-core CPU based RTC systems, both for the ELT-scale instruments and for current observatories. It will be interesting to see how other AO groups approach the problem and to see their solutions. I would be interested in investigating the other AO RTC software packages, such as CACAO (Guyon et al., 2018), COSMIC (Gratadour, 2018) and HEART (Dunn et al., 2018), and their performance for ELT-scale AO, both on current hardware and also on the upcoming Zen 2 based EPYC CPUs from AMD.

The Zen 2 based EPYC Rome CPUs will be available with up to 64 CPU cores and 8 DDR4 memory channels running at 3200 MHz per socket. This memory speed is 20% greater than that available on the current generation of Zen 1 based EPYC Naples processors, giving a potential 20% boost in memory bandwidth performance. These specifications make it potentially feasible to process a single ELT-scale WFS at up to 1 kHz or multiple smaller dimension WFSs for current observatories at higher frame-rates.

Looking further ahead towards the future I am interested to see how the smart network interface controllers (NICs) such as the Mellanox Bluefield devices, mentioned in Section 1.2.1.2, perform in the context of AO RTC. The smart NICs are ideally suited for performing the duties of a WFS processing unit to process the incoming pixels to wavefront slope values before passing the data on the host CPU for the reconstruction step. This can reduce the impact of the data transfer latency of the PCI bus by requiring much less data to be copied into main CPU memory and can

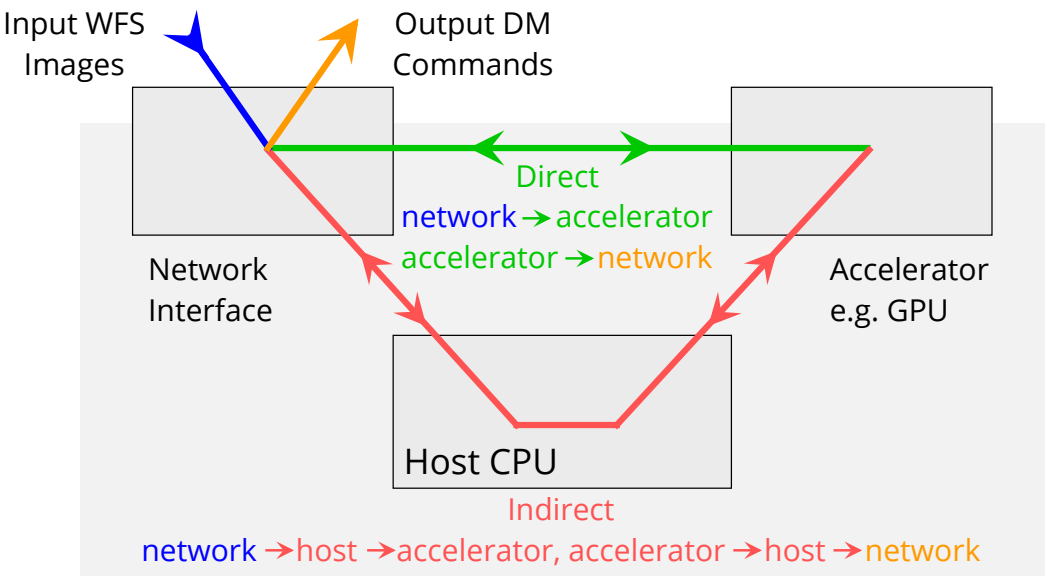


Figure 7.2: A comparison of the standard indirect data transfer through the host CPU and the more efficient direct to acclerator transfer sheme. A downside to the latter is that it currently involves non-portable software and the use of proprietary libraries.

free up the host CPU cores to further accelerate the reconstruction step. They are also a potential COTS solution to the GPU interconnect problem by allowing the host CPU memory to be bypassed and therefore removing the latency overhead of extra data transfers as shown in Figure 7.2.

The CPU technologies used in this report are standard off the shelf products that have been developed without the specific problem of AO RTC in mind. The CPU vendors cater mostly to the server and HPC markets, the requirements of which don't overlap significantly with the needs of AO RTC. Based on the experience I gained as outlined in this report I would like to see in the future CPU hardware that combines the strengths of the Xeon Phi processors with the greater single core performance of standard server CPUs. This is almost realised with the Zen 2 based EPYC Rome CPUs due their high core count, up to 64 cores per socket, and their high memory bandwidth, up to 410 $\mathrm{GB\,s^{-1}}$ theoretical for a dual-socket system.

The Xeon Phi processor was ahead of its time in terms of its core count and memory bandwidth capabilities and it will be some time before a comparable single socket

system will be available. To replace the Xeon Phi I would like to see a many core Intel Xeon or AMD EPYC design with 64 cores able to operate with at least a 4 GHz clock speed with at least an 8 GB pool of on-chip high bandwidth memory with a memory bandwidth exceeding $1\,\mathrm{TB\,s^{-1}}$. This CPU should be compatible with a four socket motherboard configuration with high speed interconnects between each of the sockets.

I believe this type of system would provide the best performance, not only for ELT-scale AO RTC, but for AO RTC in general. The fast cores would give single threaded tasks much greater performance than on the Xeon Phi processors and the on-chip high bandwidth memory would reduce the impact of the significantly memory bandwidth bound reconstruction step. This type of CPU architecture coupled with a smart network interconnect acting as a WFS processing unit would be ideally suited for tackling the problem of ELT-scale AO RTC.

7.4 Final Remarks

This thesis demonstrates the capability of the CPU-based DARC RTC platform to enable the acceleration of AO RTC to achieve the required performance for the next generation of ELT-scale AO systems. A CPU-based RTC provides the flexibility and performance necessary to scale to the ELT AO problem size without introducing unnecessary complexity or the need to rely on non-standard technologies or programming techniques. It will be interesting to see the development of the AO RTC systems for the ELTs and hopefully some of the ideas presented in this thesis will be used for their successful operation.

Bibliography

Angel, J.R.P. Ground-based imaging of extrasolar planets using adaptive optics. *Nature*, 368:203–207, Mar. 1994. doi: 10.1038/368203a0.

AravisProject. Aravis, 2018. URL `https://github.com/AravisProject/aravis`. [Online; accessed 21-January-2018].

Arsenault, R., Biasi, R., Gallieni, D. et al. A deformable secondary mirror for the vlt, 2006. URL `https://doi.org/10.1117/12.672879`.

Ashcraft, J. and Baranec, C. Simulated Guide Stars: Adapting the Robo-AO Telescope Simulator to UH 88. In *American Astronomical Society Meeting Abstracts #231*, volume 231 of *American Astronomical Society Meeting Abstracts*, page 152.13, Jan. 2018.

Babcock, H.W. The Possibility of Compensating Astronomical Seeing. *Publications of the Astronomical Society of the Pacific*, 65:229, 1953. ISSN 0004-6280. doi: 10.1086/126606. URL `http://iopscience.iop.org/article/10.1086/126606`.

Baranec, C., Lloyd-Hart, M., Milton, N.M. et al. Astronomical imaging using ground-layer adaptive optics, 2007. URL `https://doi.org/10.1117/12.732609`.

Baranec, C., Hart, M., Milton, N.M. et al. ON-SKY WIDE-FIELD ADAPTIVE OPTICS CORRECTION USING MULTIPLE LASER GUIDE STARS AT THE MMT. *The Astrophysical Journal*, 693(2):1814–1820, mar 2009.

Baranec, C., Riddle, R., Ramaprakash, A.N. et al. Robo-ao: An autonomous laser adaptive optics and science system. In *Imaging and Applied Optics*, page AWA2. Optical Society of America, 2011. doi: 10.1364/AOPT.2011.AWA2. URL `http://www.osapublishing.org/abstract.cfm?URI=AOPT-2011-AWA2`.

Baranec, C., Riddle, R., Law, N. et al. Rise of The Machines: First Year Operations of The Robo-AO Visible-Light Laser-Adaptive-Optics Instrument. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, page E48, Sept. 2013.

Baranec, C., Chun, M., Hall, D. et al. The robo-ao-2 facility for rapid visible/near-infrared ao imaging and the demonstration of hybrid techniques, 2018. URL `https://doi.org/10.1117/12.2312835`.

Barr, D., Basden, A., Dipper, N. et al. Reducing adaptive optics latency using many-core processors. 2015. URL `http://dx.doi.org/10.20353/K3T4CP1131546`.

Basden, A. The durham ao real-time controller and the canary implementation. In *Imaging and Applied Optics*, page AMB1. Optical Society of America, 2011. doi: 10.1364/AOPT.2011.AMB1. URL `http://www.osapublishing.org/abstract.cfm?URI=AOPT-2011-AMB1`.

Basden, A., Geng, D., Myers, R. et al. Durham adaptive optics real-time controller. *Applied optics.*, 49(32):6354–6363, November 2010. URL `http://dro.dur.ac.uk/10424/`.

Basden, A.G. Investigation of power8 processors for astronomical adaptive optics real-time control. *Monthly Notices of the Royal Astronomical Society*, 452(2):

1694–1701, 2015. doi: 10.1093/mnras/stv1396. URL +http://dx.doi.org/10.1093/mnras/stv1396.

Basden, A.G., Myers, R. and Butterley, T. Considerations for EAGLE from Monte Carlo adaptive optics simulation. *Appl. Optics*, 49:G1–G8, May 2010.

Basden, A.G., Chemla, F., Dipper, N. et al. Real-time correlation reference update for astronomical adaptive optics. *Monthly Notices of the Royal Astronomical Society*, 439(1):968–976, 01 2014. ISSN 0035-8711. doi: 10.1093/mnras/stu027. URL https://doi.org/10.1093/mnras/stu027.

Basden, A.G., Atkinson, D., Bharmal, N.A. et al. Experience with wavefront sensor and deformable mirror interfaces for wide-field adaptive optics systems. *Monthly Notices of the Royal Astronomical Society*, 459(2):1350–1359, 2016. doi: 10.1093/mnras/stw730. URL +http://dx.doi.org/10.1093/mnras/stw730.

Basden, A.G., Bardou, L., Bonaccini Calia, D. et al. On-sky demonstration of matched filters for wavefront measurements using ELT-scale elongated laser guide stars. *Monthly Notices of the Royal Astronomical Society*, 466(4):5003–5010, 01 2017. ISSN 0035-8711. doi: 10.1093/mnras/stx062. URL https://doi.org/10.1093/mnras/stx062.

Basden, A.G., Jenkins, D., Morris, T.J. et al. Efficient implementation of pseudo open loop control for adaptive optics on Extremely Large Telescopes. 03 2019. doi: 10.1093/mnras/stz918. URL https://doi.org/10.1093/mnras/stz918.

Beckers, J.M. Increasing the Size of the Isoplanatic Patch with Multiconjugate Adaptive Optics. In Ulrich, M.H., editor, *European Southern Observatory Conference and Workshop Proceedings*, volume 30 of *European Southern Observatory Conference and Workshop Proceedings*, page 693, 1988.

Beuzit, J.L., Vigan, A., Mouillet, D. et al. SPHERE: the exoplanet imager for the Very Large Telescope. *arXiv e-prints*, art. arXiv:1902.04080, Feb 2019.

Bharmal, N.A., Myers, R.M., Basden, A.G. et al. An interferometric wavefront sensor for high-sensitivity low-amplitude measurements, 2012. URL `https://doi.org/10.1117/12.925988`.

Biasi, R., Manetti, M., Andrighettoni, M. et al. E-elt m4 adaptive unit final design and construction: a progress report. In *Astronomical Telescopes + Instrumentation*, volume 9909, pages 9909 – 9909 – 16, 2016. doi: 10.1117/12.2234735. URL `https://doi.org/10.1117/12.2234735`.

Bitenc, U., Basden, A.G., Dipper, N.A. et al. Suitability of gpus for real-time control of large astronomical adaptive optics instruments. *Journal of Real-Time Image Processing*, 14(4):743–751, Apr 2018. ISSN 1861-8219. doi: 10.1007/s11554-017-0702-7. URL `https://doi.org/10.1007/s11554-017-0702-7`.

Boccas, M., Rigaut, F., Bec, M. et al. Laser guide star upgrade of Altair at Gemini North. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6272 of *Proc.SPIE*, page 62723L, June 2006. doi: 10.1117/12.670842.

Born, M. and Wolf, E. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 1997. ISBN 9780521639217. URL `https://books.google.co.uk/books?id=fbwTQwAACAAJ`.

Bosiers, J., Roks, E., Peek, H.L. et al. Frame-transfer ccd's with all-gates pinning: Device modeling and dark-current evaluation. In *Proc. 1993 IEEE Workshop on Charge-Coupled Devices and Advanced Image Sensors*, 1993.

Bouchez, A.H., Acton, D.S., Biasi, R. et al. The giant magellan telescope adaptive optics program, 2014. URL `https://doi.org/10.1117/12.2057613`.

Bouchez, A.H., Angeli, G.Z., Ashby, D.S. et al. An overview and status of gmt active and adaptive optics, 2018. URL `https://doi.org/10.1117/12.2314255`.

Brandl, B.R., Agócs, T., Aitink-Kroes, G. et al. Status of the mid-infrared e-elt imager and spectrograph metis, 2016. URL `https://doi.org/10.1117/12.2233974`.

Chiozzi, G., Kiekebusch, M., Kornweibel, N. et al. The elt control system, 2018. URL `https://doi.org/10.1117/12.2312164`.

Ciliegi, P., Diolaiti, E., Abicca, R. et al. Maory for elt: preliminary design overview. In *Astronomical Telescopes + Instrumentation*, volume 10703, pages 10703 – 10703 – 10, 2018. doi: 10.1117/12.2313672. URL `https://doi.org/10.1117/12.2313672`.

Close, L.M., Gasho, V., Kopon, D. et al. The Magellan Telescope Adaptive Secondary AO System: a visible and mid-IR AO facility. In *Adaptive Optics Systems II*, volume 7736 of *Proc.SPIE*, page 773605, July 2010. doi: 10.1117/12.857924.

Close, L.M., Males, J.R., Follette, K.B. et al. Into the blue: AO science with MagAO in the visible. In *Adaptive Optics Systems IV*, volume 9148 of *Proc.SPIE*, page 91481M, Aug. 2014. doi: 10.1117/12.2057297.

Correia, C. Architecture of elt 1st light instruments' hard real time computing facility with xeon-phis, 2018.

Currie, T., Guyon, O., Tamura, M. et al. Subaru/SCExAO First-light Direct Imaging of a Young Debris Disk around HD 36546. *Astrophysical Journal, Letters*, 836:L15, Feb. 2017. doi: 10.3847/2041-8213/836/1/L15.

Davies, R., Schubert, J., Hartl, M. et al. MICADO: first light imager for the E-ELT. In *Ground-based and Airborne Instrumentation for Astronomy VI*, volume 9908 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 99081Z, Aug 2016. doi: 10.1117/12.2233047.

Delacroix, C., Langlois, M.P., Loupias, M. et al. Development of an elt xao testbed using a mach-zehnder wavefront sensor: calibration of the deformable mirror, 2015. URL `https://doi.org/10.1117/12.2189094`.

Dicke, R.H. Phase-contrast detection of telescope seeing errors and their correction. *"The Astrophysical Journal"*, 198:605–615, jun 1975. doi: 10.1086/153639.

Diolaiti, E., Ciliegi, P., Abicca, R. et al. Maory: adaptive optics module for the e-elt, 2016. URL `https://doi.org/10.1117/12.2234585`.

Doel, A.P., Dunlop, C.N., Major, J.V. et al. MARTINI - Sensing and control system design. In Ealey, M.A., editor, *Active and Adaptive Optical Components*, volume 1543 of *Proc.SPIE*, pages 472–478, Jan. 1992. doi: 10.1117/12.51202.

d'Orgeville, C., Daruich, F., Arriagada, G. et al. The gemini south mcao laser guide star facility: getting ready for first light, 2008. URL `https://doi.org/10.1117/12.788970`.

Downing, M., Amico, P., Brinkmann, M. et al. Update on development of wfs cameras at eso for the elt. volume 10703, pages 10703 – 10703 – 14, 2018. doi: 10.1117/12.2314489. URL `https://doi.org/10.1117/12.2314489`.

Dunn, J., Kerley, D., Smith, M. et al. The real-time controller (rtc) for the narrow field infrared adaptive optics system (nfiraos) for tmt final design, 2018. URL `https://doi.org/10.1117/12.2314226`.

Ellerbroek, B. and Tyler, D. Adaptive optics sky coverage calculations for the gemini-north telescope. *PASP*, 110(744):165–185, 1998. ISSN 00046280, 15383873. URL `http://www.jstor.org/stable/10.1086/316120`.

Ellerbroek, B.L. First-order performance evaluation of adaptive-optics systems for atmospheric-turbulence compensation in extended-field-of-view astronomical telescopes. *J. Opt. Soc. Am. A*, 11(2):783–805, Feb 1994. doi: 10.1364/JOSAA.11.000783. URL `http://josaa.osa.org/abstract.cfm?URI=josaa-11-2-783`.

Ellerbroek, B.L. Efficient computation of minimum-variance wave-front reconstructors with sparse matrix techniques. *J. Opt. Soc. Am. A*, 19(9):1803–1816, Sep

2002. doi: 10.1364/JOSAA.19.001803. URL http://josaa.osa.org/abstract.cfm?URI=josaa-19-9-1803.

Esposito, S. and Riccardi, A. Pyramid Wavefront Sensor behavior in partial correction Adaptive Optic systems. *Astronomy and Astrophysics*, 369:L9–L12, Apr. 2001. doi: 10.1051/0004-6361:20010219.

Esposito, S., Riccardi, A., Pinna, E. et al. Large binocular telescope adaptive optics system: new achievements and perspectives in adaptive optics, 2011. URL https://doi.org/10.1117/12.898641.

Esposito, S., Riccardi, A., Pinna, E. et al. Natural guide star adaptive optics systems at lbt: Flao commissioning and science operations status, 2012. URL https://doi.org/10.1117/12.927109.

Fedrigo, E. and Donaldson, R. Sparta roadmap and future challenges, 2010. URL https://doi.org/10.1117/12.857109.

Fedrigo, E., Donaldson, R., Soenke, C. et al. SPARTA: the ESO standard platform for adaptive optics real time applications. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6272 of *Proc.SPIE*, page 627210, June 2006. doi: 10.1117/12.671919.

Foy, R. and Labeyrie, A. Feasibility of adaptive telescope with laser probe. *Astronomy & Astrophysics*, 152:L29–L31, Nov. 1985.

Fried, D.L. The effect of wavefront distortion on the performance of an ideal optical heterodyne receiver and an ideal camera, 1965.

Fried, D.L. Optical resolution through a randomly inhomogeneous medium for very long and very short exposures. *J. Opt. Soc. Am.*, 56(10):1372–1379, Oct 1966. doi: 10.1364/JOSA.56.001372. URL http://www.osapublishing.org/abstract.cfm?URI=josa-56-10-1372.

Fried, D.L. Greenwood frequency measurements. *J. Opt. Soc. Am. A*, 7(5):946–947, May 1990. doi: 10.1364/JOSAA.7.000946. URL `http://josaa.osa.org/abstract.cfm?URI=josaa-7-5-946`.

Fried, D.L. and Mevers, G.E. Evaluation of ro for propagation down through the atmosphere. *Appl. Opt.*, 13(11):2620–2622, Nov 1974. doi: 10.1364/AO.13.002620. URL `http://ao.osa.org/abstract.cfm?URI=ao-13-11-2620`.

Fugate, R.Q., Fried, D.L., Ameer, G.A. et al. Measurement of atmospheric wavefront distortion using scattered light from a laser guide-star. *Nature*, 353:144–146, Sept. 1991. doi: 10.1038/353144a0.

Gendron, E., Assémat, F., Hammer, F. et al. Falcon: multi-object ao. *Comptes Rendus Physique*, 6(10):1110 – 1117, 2005. ISSN 1631-0705. doi: https://doi.org/10.1016/j.crhy.2005.10.012. URL `http://www.sciencedirect.com/science/article/pii/S1631070505001660`. Multi-Conjugate Adaptive Optics for very large telescopes.

Gendron, E., Vidal, F., Brangier, M. et al. Moao first on-sky demonstration with canary. *A&A*, 529:L2, 2011. doi: 10.1051/0004-6361/201116658. URL `https://doi.org/10.1051/0004-6361/201116658`.

Gilles, L. and Ellerbroek, B. Shack-Hartmann wavefront sensing with elongated sodium laser beacons: centroiding versus matched filtering. *Applied Optics*, 45:6568–6576, Sept. 2006. doi: 10.1364/AO.45.006568.

GNU. Options that control optimization. URL `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`.

Gratadour, D. The cosmic rtc platform, 2018. URL `https://indico.obspm.fr/event/57/contributions/211/attachments/178/197/dgratadour-cosmic.pdf`.

Gratadour, D., Morris, T., Biasi, R. et al. Prototyping ao rtc using emerging high performance computing technologies with the green flash project, 2018. URL https://doi.org/10.1117/12.2312686.

Greenwood, D.P. and Fried, D.L. Power spectra requirements for wave-front-compensative systems∗. *J. Opt. Soc. Am.*, 66(3):193–206, Mar 1976. doi: 10.1364/JOSA.66.000193. URL http://www.osapublishing.org/abstract.cfm?URI=josa-66-3-193.

Guesalaga, A., Neichel, B., O'Neal, J. et al. Mitigation of vibrations in adaptive optics by minimization of closed-loop residuals. *Optics express*, 21(9):10676–10696, 2013. ISSN 1094-4087 (Electronic). doi: 10.1364/OE.21.010676.

Guyon, O. Extreme adaptive optics. *Annual Review of Astronomy and Astrophysics*, 56(1):315–355, 2018. doi: 10.1146/annurev-astro-081817-052000. URL https://doi.org/10.1146/annurev-astro-081817-052000.

Guyon, O., Sevin, A., Gratadour, D. et al. The compute and control for adaptive optics (cacao) real-time control software package, 2018. URL https://doi.org/10.1117/12.2314315.

Hardy, J. *Adaptive Optics for Astronomical Telescopes*. Oxford Series in Optical & Ima. Oxford University Press, 1998. ISBN 9780195090192. URL https://books.google.co.uk/books?id=-0aAWyckS_8C.

Herriot, G., Morris, S., Roberts, S.C. et al. Innovations in gemini adaptive optics system design, 1998. URL https://doi.org/10.1117/12.321684.

Herriot, G., Andersen, D., Atwood, J. et al. Nfiraos: first facility ao system for the thirty meter telescope, 2014. URL https://doi.org/10.1117/12.2055525.

Hippler, S. Adaptive Optics for Extremely Large Telescopes. *arXiv e-prints*, Aug. 2018.

IEEE. Ieee standard for floating-point arithmetic, Aug 2008.

Intel. A guide to auto-vectorization with intel c++ compilers, 2012. URL https://software.intel.com/en-us/articles/a-guide-to-auto-vectorization-with-intel-c-compilers. [Online; accessed 27-September-2017].

Intel. Memory modes and cluster modes: Configuration and use cases, 2015. URL https://software.intel.com/en-us/articles/intel-xeon-phi-x200-processor-memory-modes-and-cluster-modes-configuration-and-use-cases. [Online; accessed 27-September-2017].

Intel. Intel xeon phi processor 7200 family memory management optimizations, 2016. URL https://software.intel.com/en-us/articles/intel-xeon-phi-processor-7200-family-memory-management-optimizations. [Online; accessed 27-September-2017].

Intel. Intelxeon phi processors, 2017a. URL https://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors.html. [Online; accessed 27-September-2017].

Intel. Step by step performance optimization with intel c++ compiler, 2017b. URL https://software.intel.com/en-us/articles/step-by-step-optimizing-with-intel-c-compiler.

Jenkins, D., 2019. URL https://commons.wikimedia.org/wiki/File:ZernikePolynome6.svg. "ZernikePolynome6", Reordered the tip-tilt modes and resized the legend by d.r.jenkins@durham.ac.uk, https://creativecommons.org/licenses/by-sa/3.0/legalcode.

Jenkins, D., Basden, A.G., Myers, R.M. et al. Multi-node homogeneous xeon phi architecture for elt scale adaptive optics rtc (conference presentation). volume 10707, 2 2018a. doi: 10.1117/12.2313943. URL https://doi.org/10.1117/12.2313943.

Jenkins, D.R., Basden, A. and Myers, R.M. Elt-scale real-time control on intel xeon phi and many core cpus. 2017. doi: 10.26698/AO4ELT5.0046.

Jenkins, D.R., Basden, A. and Myers, R.M. Elt-scale adaptive optics real-time control with the intel xeon phi many integrated core architecture. *Monthly Notices of the Royal Astronomical Society*, 478(3):3149–3158, 2018b. doi: 10.1093/mnras/sty1310. URL `http://dx.doi.org/10.1093/mnras/sty1310`.

Jenkins, D.R., Basden, A.G., Myers, R.M. et al. An elt scale mcao real-time control prototype using xeon phi technologies. volume 10703, pages 10703 – 10703 – 7, 1 2018c. doi: 10.1117/12.2312845. URL `https://doi.org/10.1117/12.2312845`.

Jenkins, D.R., Basden, A. and Myers, R.M. A many-core cpu prototype of an mcao and ltao rtc for elt-scale instruments. *Monthly Notices of the Royal Astronomical Society*, 2019.

Jensen-Clem, R., Duev, D.A., Riddle, R. et al. The performance of the robo-AO laser guide star adaptive optics system at the kitt peak 2.1 m telescope. *The Astronomical Journal*, 155(1):32, dec 2017.

Johns, M., Angel, R., Shectman, S. et al. Status of the giant magellan telescope (gmt) project. volume 5489, pages 5489 – 5489 – 13, 2004. doi: 10.1117/12.550741. URL `http://dx.doi.org/10.1117/12.550741`.

Johnston, D.C. and Welsh, B.M. Analysis of multiconjugate adaptive optics. *Journal of the Optical Society of America A*, 11:394–408, Jan. 1994. doi: 10.1364/JOSAA.11.000394.

Jorden, P., Bourke, D., Cassidy, R. et al. Teledyne e2v sensors optimised for ground-based and space applications, 2018. URL `https://doi.org/10.1117/12.2310097`.

Kasper, M.E., Charton, J., Delabre, B. et al. Lgs implementation for naos, 2004. URL `https://doi.org/10.1117/12.551650`.

Kerrisk, M. Packet(7) linux programmer's manual, 2018. URL `http://man7.org/linux/man-pages/man7/packet.7.html`.

Kolmogorov, A.N. Dissipation of energy in the locally isotropic turbulence. *Proceedings: Mathematical and Physical Sciences*, 434(1890):15–17, 1991. ISSN 09628444. URL `http://www.jstor.org/stable/51981`.

Kulcsár, C., Raynaud, H.F., Petit, C. et al. Optimal control, observers and integrators in adaptive optics. *Optics express*, 14(17):7464–7476, 2006. ISSN 1094-4087. doi: 10.1364/OE.14.007464.

Kulcsár, C., Raynaud, H.F., Petit, C. et al. Minimum variance prediction and control for adaptive optics. *Automatica*, 48(9):1939–1954, 2012. ISSN 00051098. doi: 10.1016/j.automatica.2012.03.030. URL `http://dx.doi.org/10.1016/j.automatica.2012.03.030`.

Lardière, O., Nash, R., Markes, J.P. et al. Final opto-mechanical design of Raven, a MOAO science demonstrator for Subaru. In *Adaptive Optics Systems III*, volume 8447 of *Proc.SPIE*, page 844753, July 2012. doi: 10.1117/12.927176.

Lardière, O., Ono, Y., Dave, A. et al. On-sky results of Raven, a MOAO science demonstrator at Subaru Telescope. In *Adaptive Optics for Extremely Large Telescopes IV (AO4ELT4)*, page E77, Oct. 2015.

Larkin, J.E., Moore, A.M., Wright, S.A. et al. The infrared imaging spectrograph (iris) for tmt: instrument overview, 2016. URL `https://doi.org/10.1117/12.2232212`.

Li, Z. and Li, X. Fundamental performance of transverse wind estimator from shack-hartmann wave-front sensor measurements. *Opt. Express*, 26(9):11859–11876, Apr 2018. doi: 10.1364/OE.26.011859. URL `http://www.opticsexpress.org/abstract.cfm?URI=oe-26-9-11859`.

Lozi, J., Guyon, O., Jovanovic, N. et al. Scexao, an instrument with a dual purpose: perform cutting-edge science and develop new technologies, 2018. URL `https://doi.org/10.1117/12.2314282`.

Macintosh, B., Graham, J.R., Ingraham, P. et al. First light of the Gemini Planet Imager. *Proceedings of the National Academy of Science*, 111:12661–12666, Sept. 2014. doi: 10.1073/pnas.1304215111.

Madec, P.Y., Arsenault, R., Kuntschner, H. et al. Adaptive optics facility: from an amazing present to a brilliant future..., 2018. URL `https://doi.org/10.1117/12.2312428`.

Marchetti, E., Hubin, N.N., Fedrigo, E. et al. MAD the ESO multi-conjugate adaptive optics demonstrator. In Wizinowich, P.L. and Bonaccini, D., editors, *Adaptive Optical System Technologies II*, volume 4839 of *Proc.SPIE*, pages 317–328, Feb. 2003. doi: 10.1117/12.458859.

Marchetti, E., Brast, R., Delabre, B. et al. On-sky Testing of the Multi-Conjugate Adaptive Optics Demonstrator. *The Messenger*, 129:8–13, Sept. 2007.

Marechal, A. *Rev. d'Optique*, 26, 257, 1947.

McCalpin, J.D. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, Dec. 1995.

McDermid, R.M., Bacon, R., Bauer, S. et al. Muse: A second-generation integral-field spectrograph for the vlt. In Kaufer, A. and Kerber, F., editors, *The 2007 ESO Instrument Calibration Workshop*, pages 325–336, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-76963-7.

McLeod, B., Bouchez, A.H., Espeland, B. et al. The giant magellan telescope active optics system, 2014. URL `https://doi.org/10.1117/12.2056435`.

Mellanox. Mellanox bluefield smartnic, 2018. URL `http://www.mellanox.com/related-docs/prod_adapter_cards/PB_BlueField_Smart_NIC.pdf`.

Merkle, F., Rousset, G., Kern, P. et al. First diffraction-limited astronomical images with adaptive optics. *Proc.SPIE*, 1236 pt 1(1990):193–202, 1990. ISSN 0277786X. doi: 10.1117/12.19189. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-17044445365{&}partnerID=40{&}md5=cd3a9dfeb3694b08517e302d7de5e13f`.

Morzinski, K.M., Close, L.M., Males, J.R. et al. MagAO: Status and on-sky performance of the Magellan adaptive optics system. In *Adaptive Optics Systems IV*, volume 9148 of *Proc.SPIE*, page 914804, July 2014. doi: 10.1117/12.2057048.

Murphy, D.V., Primmerman, C.A., Zollars, B.G. et al. Experimental demonstration of atmospheric compensation using multiple synthetic beacons. *Optics Letters*, 16:1797–1799, Nov. 1991. doi: 10.1364/OL.16.001797.

Myers, R.M., Doel, A.P., Dunlop, C.N. et al. Astronomical adaptive optics system for use on a 4-m-class telescope at optical wavelengths. In Ealey, M.A. and Merkle, F., editors, *Adaptive Optics in Astronomy*, volume 2201 of *Proc.SPIE*, pages 437–446, May 1994. doi: 10.1117/12.176077.

Myers, R.M., Hubert, Z., Morris, T.J. et al. Canary: the on-sky ngs/lgs moao demonstrator for eagle, 2008. URL `https://doi.org/10.1117/12.789544`.

Nakajima, T. Planet detectability by an adaptive optics stellar coronagraph. *ApJ*, 425:348–357, Apr. 1994. doi: 10.1086/173990.

N'Diaye, M., Dohlen, K., Fusco, T. et al. Calibration of quasi-static aberrations in exoplanet direct-imaging instruments with a Zernike phase-mask sensor. *Astronomy & Astrophysics*, 555:A94, Jul 2013. doi: 10.1051/0004-6361/201219797.

Neichel, B., Fusco, T., Sauvage, J.F. et al. The adaptive optics modes for harmoni: from classical to laser assisted tomographic ao. volume 9909, pages 9909 – 9909

– 15, 2016. doi: 10.1117/12.2231681. URL https://doi.org/10.1117/12.2231681.

Noll, R.J. Zernike polynomials and atmospheric turbulence∗. *J. Opt. Soc. Am.*, 66(3):207–211, Mar 1976. doi: 10.1364/JOSA.66.000207. URL http://www.osapublishing.org/abstract.cfm?URI=josa-66-3-207.

OpenMP Architecture Review Board. OpenMP application program interface version 4.5, 2015. URL http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf. [Online; accessed 27-September-2017].

Osborn, J., Wilson, R.W., Sarazin, M. et al. Optical turbulence profiling with Stereo-SCIDAR for VLT and ELT. *MNRAS*, 478:825–834, July 2018. doi: 10.1093/mnras/sty1070.

Papermaster, M. Amd next horizon, 2018. URL https://www.amd.com/system/files/documents/next_horizon_mark_papermaster_presentation.pdf.

Pazder, J.S., Roberts, S., Abraham, R. et al. Wfos: a wide field optical spectrograph for the thirty meter telescope, 2006. URL https://doi.org/10.1117/12.672712.

Petit, C., Conan, J.M., Kulcsár, C. et al. First laboratory validation of vibration filtering with LQG control law for adaptive optics. *Optics express*, 16(1):87–97, 2008. ISSN 1094-4087. doi: 10.1364/OE.16.000087. URL http://www.ncbi.nlm.nih.gov/pubmed/18521135.

Pettazzi, L., Fedrigo, E. and Clare, R. Impact of latency and jitter on the performance of adaptive optics systems for elts, 2012. URL https://www.eso.org/sci/meetings/2012/RTCWorkshop/1_2_fedrigo.pdf.

Piatrou, P. and Gilles, L. Robustness study of the pseudo open-loop controller for multiconjugate adaptive optics. *Appl. Opt.*, 44(6):1003–1010, Feb 2005. doi: 10.1364/AO.44.001003. URL http://ao.osa.org/abstract.cfm?URI=ao-44-6-1003.

Poyneer, L.A., De Rosa, R.J., Macintosh, B. et al. On-sky performance during verification and commissioning of the Gemini Planet Imager's adaptive optics system. In *Adaptive Optics Systems IV*, volume 9148 of *Proc.SPIE*, page 91480K, July 2014. doi: 10.1117/12.2057092.

Quirós-Pacheco, F., Busoni, L., Agapito, G. et al. First light ao (flao) system for lbt: performance analysis and optimization, 2010. URL `https://doi.org/10.1117/12.858208`.

Racine, R., Salmon, D., Cowley, D. et al. Mirror, dome, and natural seeing at CFHT. *PASP*, 103:1020–1032, Sept. 1991. doi: 10.1086/132920.

Ré, P. CCD ASTROPHOTOGRAPHY by Pedro Ré, 2019. URL `http://www.astrosurf.com/re/standard.jpg`. [Online; accessed 21-March-2019].

Riddle, R., Baranec, C., Law, N.M. et al. Robo-AO: Initial results from the first autonomous laser guide star adaptive optics instrument. *Contributions of the Astronomical Observatory Skalnate Pleso*, 43:190–199, Mar. 2014.

Rigaut, F. Ground Conjugate Wide Field Adaptive Optics for the ELTs. In Vernet, E., Ragazzoni, R., Esposito, S. et al, editors, *European Southern Observatory Conference and Workshop Proceedings*, volume 58 of *European Southern Observatory Conference and Workshop Proceedings*, page 11, 2002.

Rigaut, F. and Gendron, E. Laser guide star in adaptive optics - The tilt determination problem. *Astronomy & Astrophysics*, 261:677–684, Aug. 1992.

Rigaut, F., Neichel, B., Boccas, M. et al. GeMS: first on-sky results. In *Adaptive Optics Systems III*, volume 8447 of *Proc.SPIE*, page 84470I, July 2012. doi: 10.1117/12.927061.

Roddier, F. Curvature sensing and compensation: a new concept in adaptive optics. *Appl. Opt.*, 27(7):1223–1225, Apr 1988. doi: 10.1364/AO.27.001223. URL `http://ao.osa.org/abstract.cfm?URI=ao-27-7-1223`.

Rodríguez-Ramos, L.F., Chulani, H., Martín, Y. et al. FPGA-based real time controller for high order correction in EDIFISE. In *Adaptive Optics Systems III*, volume 8447, page 84472R, jul 2012. doi: 10.1117/12.925352.

Rosensteiner, M. Wavefront reconstruction for extremely large telescopes via cure with domain decomposition. *J. Opt. Soc. Am. A*, 29(11):2328–2336, Nov 2012. doi: 10.1364/JOSAA.29.002328. URL `http://josaa.osa.org/abstract.cfm?URI=josaa-29-11-2328`.

Rousset, G., Lacombe, F., Puget, P. et al. Design of the nasmyth adaptive optics system (naos) of the vlt, 1998. URL `https://doi.org/10.1117/12.321686`.

Rousset, G., Lacombe, F., Puget, P. et al. Naos–the first ao system of the vlt: on-sky performance, 2003. URL `https://doi.org/10.1117/12.459332`.

Rousset, G., Fusco, T., Assemat, F. et al. EAGLE multi-object AO concept study for the E-ELT. In *Adaptative Optics for Extremely Large Telescopes*, page 02008, 2010. doi: 10.1051/ao4elt/201002008.

Salama, M., Baranec, C., Jensen-Clem, R. et al. Robo-ao kitt peak: status of the system and deployment of a sub-electron readnoise ir camera to detect low-mass companions, 2016. URL `https://doi.org/10.1117/12.2233741`.

Sarazin, M., Le Louarn, M., Ascenso, J. et al. Defining reference turbulence profiles for e-elt ao performance simulations. In Esposito, S. and Fini, L., editors, *Proceedings of the Third AO4ELT Conference*, Firenze, 2013. INAF - Osservatorio Astrofisico di Arcetri. ISBN 978-88-908876-0-4. doi: 10.12839/AO4ELT3.13383.

Schreiber, L., Feautrier, P., Stadler, E. et al. The maory laser guide star wavefront sensor: design status. volume 10703, pages 10703 – 10703 – 9, 2018. doi: 10.1117/12.2314467. URL `https://doi.org/10.1117/12.2314467`.

Schwartz, N., Sauvage, J.F., Correia, C. et al. Analysis and mitigation of pupil discontinuities on adaptive optics performance. volume 10703, pages 10703 –

10703 – 12, 2018. doi: 10.1117/12.2313129. URL `https://doi.org/10.1117/12.2313129`.

Sharples, R.M., Doel, A.P., Dunlop, C.N. et al. Co-Phasing of Segmented Mirrors Using Neural Networks. In Merkle, F., editor, *European Southern Observatory Conference and Workshop Proceedings*, volume 48 of *European Southern Observatory Conference and Workshop Proceedings*, page 475, Jan. 1994.

Shimizu, T. Post-k supercomputer with fujitsu's original cpu, a64fx powered by arm isa, 2018. URL `https://www.fujitsu.com/global/Images/post-k_supercomputer_with_fujitsu's_original_cpu_a64fx_powered_by_arm_isa.pdf`.

Spyromilio, J., Comerón, F., D'Odorico, S. et al. Progress on the European Extremely Large Telescope. *The Messenger*, 133:2–8, Sept. 2008.

Stahl, S.M. and Sandler, D.G. Optimization and Performance of Adaptive Optics for Imaging Extrasolar Planets. *ApJ*, 454:L153, Dec. 1995. doi: 10.1086/309777.

Stepp, L.M. and Strom, S.E. The Thirty-Meter Telescope project design and development phase. In Ardeberg, A.L. and Andersen, T., editors, *Second Backaskog Workshop on Extremely Large Telescopes*, volume 5382 of *Proc.Spie*, pages 67–75, July 2004. doi: 10.1117/12.566105.

Ströbele, S., La Penna, P., Arsenault, R. et al. GALACSI system design and analysis. In *Adaptive Optics Systems III*, volume 8447 of *Proc.SPIE*, page 844737, July 2012. doi: 10.1117/12.926110.

Tallon, M. and Foy, R. Adaptive telescope with laser probe - Isoplanatism and cone effect. *Astronomy & Astrophysics*, 235:549–557, Aug. 1990.

Tatarskii, V.I. *Wave Propagation in Turbulent Medium*. McGraw-Hill, 1961.

Tatarskii, V.I. *The effects of the turbulent atmosphere on wave propagation*. 1971.

Thatte, N.A., Clarke, F., Bryson, I. et al. Harmoni: the first light integral field spectrograph for the e-elt. volume 9147, pages 9147 – 9147 – 11, 2014. doi: 10.1117/12.2055436. URL https://doi.org/10.1117/12.2055436.

Thatte, N.A., Clarke, F., Bryson, I. et al. The E-ELT first light spectrograph HARMONI: capabilities and modes. In *Ground-based and Airborne Instrumentation for Astronomy VI*, volume 9908 of *Proc.SPIE*, page 99081X, Aug. 2016. doi: 10.1117/12.2230629.

The-CentOS-Project. Centos linux, 2001. URL https://www.centos.org/about/. [Online; accessed 27-September-2017].

The Linux Kernel. The kernel's command-line parameters, 2019. URL https://www.kernel.org/doc/html/latest/admin-guide/kernel-parameters.html.

'The-Open-Group'. The open group base specifications issue 7, 2016. URL http://pubs.opengroup.org/onlinepubs/9699919799.2016edition/. [Online; accessed 27-September-2017].

Townson, Mathew, J. Correlation wavefront sensing and turbulence profiling for solar adaptive optics, 2016.

Truong, T.N., Bouchez, A.H., Burruss, R.S. et al. Design and implementation of the palm-3000 real-time control system. In *Proc.SPIE*, volume 8447, pages 8447 – 8447 – 9, 2012. doi: 10.1117/12.927867. URL http://dx.doi.org/10.1117/12.927867.

Tyson, R. *Principles of Adaptive Optics, Third Edition*. Series in Optics and Optoelectronics. CRC Press, 2010. ISBN 9781439808597. URL https://books.google.co.uk/books?id=x1PUYBvHHqcC.

Ulrich, P. Hufnagel-valley profiles for specified values of the coherence lenght and isoplanatic patch angle. *W. J. Schafer Associates*, WJSA/MA/TN-88-103, 1988.

van Kooten, M. Investigation of alternative pyramid wavefront sensors. Master's thesis, University of Victoria, 2016.

Vernet, E., Cayrel, M., Hubin, N. et al. On the way to build the m4 unit for the e-elt, 2014. URL `https://doi.org/10.1117/12.2056281`.

Vigan, A., Bonnefoy, M., Ginski, C. et al. First light of the VLT planet finder SPHERE. I. Detection and characterization of the substellar companion GJ 758 B. *A&A*, 587:A55, Mar. 2016. doi: 10.1051/0004-6361/201526465.

Wizinowich, P.L., Acton, D.S., Gregory, T. et al. Status of the W.M. Keck Adaptive Optics Facility. In Bonaccini, D. and Tyson, R.K., editors, *Adaptive Optical System Technologies*, volume 3353 of *Proc.SPIE*, pages 568–578, Sept. 1998. doi: 10.1117/12.321714.

Wizinowich, P.L., Le Mignant, D., Bouchez, A.H. et al. The W. M. Keck Observatory Laser Guide Star Adaptive Optics System: Overview. *Publications of the ASP*, 118:297–309, Feb. 2006. doi: 10.1086/499290.

Wolfe, W.L. and Zissis, G.J. *The infrared handbook*. 1985.

Xompero, M., Briguglio, R., Pariani, G. et al. Fitting error analysis and performance evaluation of m4 deformable mirror. volume 10703, pages 10703 – 10703 – 11, 2018. doi: 10.1117/12.2310105. URL `https://doi.org/10.1117/12.2310105`.

Zernike, F. Diffraction theory of the knife-edge test and its improved form, the phase-contrast method. *Monthly Notices of the Royal Astronomical Society*, 94: 377–384, Mar. 1934. doi: 10.1093/mnras/94.5.377.

Colophon

This thesis was typeset with LaTeX 2_ε. It was created using the *memoir* package, maintained by Lars Madsen, with the *madsen* chapter style. The font used is Latin Modern, derived from fonts designed by Donald E. Kunith.