ELT-scale real-time control on Intel Xeon Phi and many core CPUs

David R. Jenkins, Alastair G. Basden, and Richard M. Myers

CfAI, Department of Physics, Durham University, DH1 3LE, UK

ABSTRACT

The next generation of ELTs massively increase both the computational demands and complexity of AO real-time control systems requiring investigation into architectures that are both flexible and computationally performant. Current technologies such as FPGAs, GPUs and standard CPUs are either complex to develop for or would require far too many individual processing units for the demands of the RTC. The Intel Xeon Phi is proposed as a platform to support the computational demands of the real-time control systems for ELT scale telescopes and to provide a standard CPU like development environment. Other architectures have also been considered such as the IBM Power8 and many-core ARM CPU systems. The large number of processor cores and high bandwidth memory of the Xeon Phi and Power8 allow for massive parallelisation of the workload whilst also allowing these systems to act as socketed host CPUs. Being the host and not an accelerator removes many of the latency and/or bandwidth concerns that come with offloading and allows existing software to be compiled and run without any modifications; certain optimisations then allow the software to take full advantage of the system's hardware. A single Knights Landing Xeon Phi can currently demonstrate ELT scale SCAO running at a frame-rate $> 1kHz$ (which we hope to improve on) showing promise not only for it to act as the sole RTC processor in an SCAO system but also as a competent computational node in a distributed computing architecture for more complex AO regimes such as ELT MCAO, LTAO, and MOAO where the computational demands are much higher and the data sharing is much more complex. We will present an overview of the current generation Knights Landing Xeon Phi along with an overview of the investigation into the software and hardware optimisations to improve the performance of an ELT scale RTC system for many-core architectures.

Keywords: instrumentation: adaptive optics methods: numerical

1. INTRODUCTION

1.1 Adaptive Optics real-time control

Ground based astronomical telescopes suffer image degradation due to the optical aberrations introduced by Earth's turbulent atmosphere. Adaptive optics (AO)[1] is a technique that has been successfully deployed for many years which aims to combat the effect this has on scientific observations. The basic operation of AO involves detecting the shape of an incoming wavefront using a wave-front sensor (WFS) and using this information to revert the atmospheric perturbations by the use of an optical correcting element, usually a deformable mirror (DM). The wavefront measurements are normally approximated by sampling the local wavefront slope at a number of discreet points in the image plane, e.g., by the use of a Shack-Hartman WFS (SH-WFS) or a pyramid WFS. This technique then requires that the approximated wave-front be reconstructed from slope measurements using a typically computationally expensive approach.

The AO real-time controller (RTC) is the hardware and software responsible for operating the AO instrument, from taking WFS data and reconstructing the wavefront to delivering the resulting correction to the correcting element and everything in-between. Due to the conditions of Earth's atmosphere being dynamic and constantly changing, for AO to achieve adequate correction it must apply the wave-front corrections at a rate consistent

Further author information: (Send correspondence to D.R.J.)
D.R.J.: E-mail: d.r.jenkins@durham.ac.uk, Telephone: +44 (0)191 33 43485
A.G.B.: E-mail: a.g.basden@durham.ac.uk
R.M.M.: E-mail: r.m.myers@durham.ac.uk

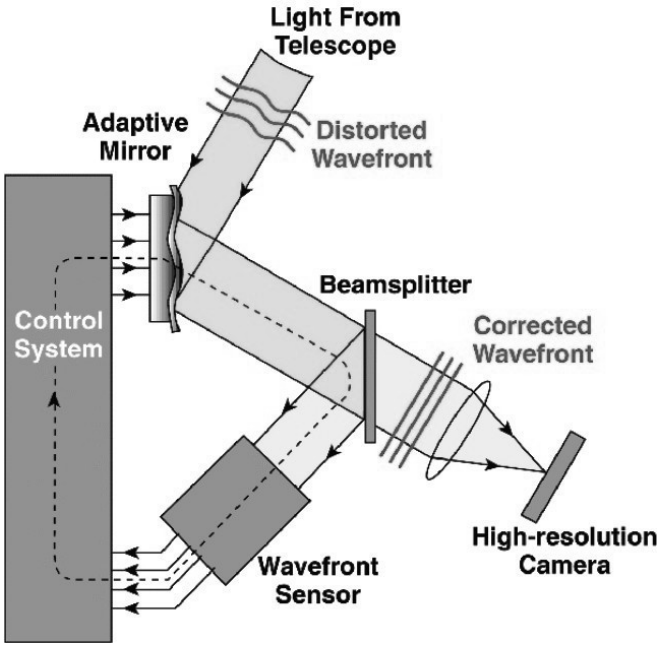

Figure 1. Schematic overview of closed loop Adaptive Optics.[2] The control system is responsible for taking WFS images, processing them, reconstructing the incident wavefront and delivering the corrections via the adaptive mirror.

with the rate of change of the atmospheric turbulence, typically on the order of $500Hz$. This places strict requirements on the performance of the RTC so that it can meet the demand set out by the requirements of the AO instrument. An AO system's requirements are mostly defined by the parameters that quantify the strength and conditions of the atmospheric turbulence present at the site of observation. These parameters can change for different types of scientific observation or for different operating wavelengths and the effect they have on the AO requirements can vary based on the attributes of the telescope, e.g., telescope diameter.

The dependence of AO system requirements on telescope diameter is an extremely important consideration for the next generation of Extremely Large Telescopes (ELTs) with primary mirror diameters greater than 20m. The number of wavefront slope measurements and the number of correcting element actuators required for the accurate reconstruction and correction of the wavefront both scale to the second power of telescope diameter. However the problem size of the most common wavefront reconstruction technique scales linearly with both of these parameters resulting in an overall fourth power dependence on telescope diameter for the reconstruction computational requirements. This presents a great challenge in the process of designing an 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 exceeds the requirements of the AO system.

1.2 Real-time Controller Hardware

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 proficiency for the algorithms required. The computational requirements are largely dictated by the algorithm and the reconstruction problem size - with the computational complexity increasing as a fourth power of telescope diameter. This scaling requires significantly greater computational performance for the next generation ELT scale telescopes than has been required previously and presents the challenge of finding suitable hardware. The AO RTC hardware is generally composed of a device or devices which are tasked with processing the AO data from beginning to end, from receiving WFS data to sending DM commands. Historically this has been achieved with a variety of devices, including digital signal processors (DSPs),[3] field programmable gate arrays (FPGAs)[3],[4] central processing units (CPUs),[5] graphics processing units (GPUs)[5],[6] etc.

These devices 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 necessary, modifying the RTC software with the advantage that they are fast and can be deterministic. CPU based systems generally aren't as fast and can have increased latencies compared to the other devices, however they are much easier to develop for and are generally backwards compatible with common programming languages. For ELT scale systems two of the main challenges are firstly the scaling of these systems for the increased computational complexity, usually requiring many of these devices working in tandem, and secondly future proofing the development of the system for updated hardware.

The Intel Xeon Phi processor family combines many low power x86 CPU cores with high bandwidth memory to enable acceleration of highly parallelisable tasks. This allows it to leverage the benefits both of having a CPU based architecture and of having a highly parallelisable workflow such as that of GPUs. The high bandwidth memory is also important for common reconstruction techniques which have memory bandwidth-bound performance. These attributes of the Xeon Phi make it a very interesting candidate for ELT-scale AO RTC as they are easy to develop for and yet have many of the advantages of GPUs.

In this paper we present the preliminary findings of an investigation of the suitability of using the Intel Xeon Phi Many Integrated Core (MIC) architecture as the processor in AO RTC. Section 2 expands upon the motivation and requirements for an ELT scale AO RTC, section 3 summarises our efforts with adapting existing RTC software for use with the Xeon Phi platform, section 4 presents our results and section 5 concludes with a summary and a discussion of our work.

2. MOTIVATION

2.1 ELT-scale AO RTC

The AO real-time controller (RTC) is responsible for controlling all aspects of the AO loop, receiving images from the WFS cameras, calculating local wave-front slopes, reconstructing the wave-front and finally distributing the results to the correcting elements. The adaptive optics control loop needs to be operated in real-time to provide optimal correction; the time taken to measure, calculate and apply a reconstruction must be consistently low from frame to frame so that there are no variable delays in applying a correction.

The next generation of ELTs provide many new challenges for AO and AO RTC. The larger aperture diameters require many more degrees of freedom in the measurement of the wave-front phase and also in applying a suitable correction. The number of degrees of freedom increases with the 2nd power of telescope diameter to maintain a similar level of correction and so the jump from 10m class telescopes to 30–40m has a massive impact on the reconstruction demands of the RTC. The computational demands of current RTCs can be accommodated by standard server CPU systems, however for the next generations ELTs, these systems no longer provide the performance necessary to meet the requirements.

Many typical AO reconstruction techniques involve computing one or more large matrix-vector-multiplies (MVMs) where the dimensions of the matrix are defined by the number of degrees of freedom (DoF) in the system. A control matrix which contains information relating to how certain wavefront measurements correspond to the appropriate actuator commands is multiplied by a vector containing the wavefront measurements, the results of which can yield the wavefront shape required for correction. A control matrix of dimensions (NxM) is made up from N DoF of a wavefront measurement and M actuators in the correcting element. This makes the MVM very large for ELTs requiring on the order of N^4 calculations where $N \approx M$.

Because each individual calculation in the MVM is simple, a multiplication and an addition, the biggest bottleneck in the computation comes from not being able to fetch the control matrix from memory fast enough for processing. The hardware metric used to quantify this is memory bandwidth, which defines the maximum throughput of data from the random-access memory (RAM) into the CPU cores themselves. CPUs generally have comparatively low memory-bandwidth compared to GPUs and it is their main disadvantage when it come to RTC. Because the Xeon Phi has high memory bandwidth, which is comparable to high end GPUs, it is much better at working on large matrices and so helps with the reconstruction problem.

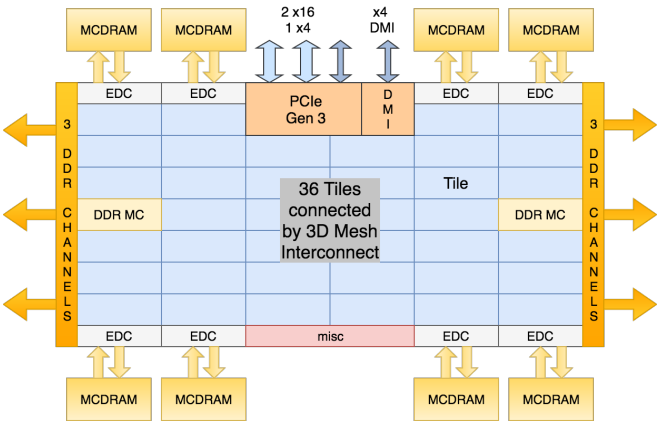

Figure 2. Schematic of Xeon Phi Knights Landing CPU showing the MIC architecture along with the high bandwidth MCDRAM.

2.2 Xeon Phi: Knights Landing

The Xeon Phi[7] is a range of High Performance Computing (HPC) processors from Intel with an emphasis on core counts and memory bandwidth. The high core counts aim to increase the throughput of parallel work flows with the high memory bandwidth allowing the processor to be sufficiently fed with data. Knights Landing (KNL) is the third generation of the Xeon Phi and is the first to be released in the socketed form factor. Previous Xeon Phis were accelerator cards only, KNL comes in two flavours, socketed host CPU and add-on PCI-E accelerator cards. For the purposes of AO RTC the socketed host will be the main aim of our investigation as it gives us a standard Linux software environment with the KNL acting as a standard CPU but with many more cores and much faster memory. It also eliminates the bottleneck of transferring data through the PCI bus to accelerator cards, which reduces the latency and jitter of the calculation and simplifies the data pipeline as the camera data is only needed by the host CPU. Previous investigations[8] have been made of AO RTC on the prior generation of the Xeon Phi Knights Corner accelerator card with results which showed promise for the future generations.

The most important thing to note about the Xeon Phi range of processors is that they use the x86 64-bit architecture and so are fully compatible with standard programming languages and tools. Intel officially supports 3 Linux distributions, Red Hat Enterprise Linux, CentOS and SuSe Enterprise Server with updated kernels that are optimised for the Xeon Phi. The benefit of this is that existing applications and tools can be ported to the Xeon Phi quickly and easily and some extra optimisation would then allow full use of the many cores and fast memory. This shows that the Xeon Phi can simply be used as an upgrade for existing x86 CPU based RTCs without having to completely re-write software for new architectures.

The KNL itself comes in four models in ascending order of performance, 7210, 7230, 7250, 7290, with the main differences being increased core counts from 64 up to 72, increased clock speed from 1.3 to 1.5 Ghz and increased memory bandwidth. There is also an optional on-chip Fabric interconnect available, Omni-Path, which would theoretically allow up to 100Gbps transfer speeds directly from the CPU to other Omni-Path devices; however for this study this is currently not being considered.

Another aspect of the Xeon Phi architecture that is of use in AO RTC is the introduction of wide 512bit vector CPU registers. These can be used to process up to 16 single precision floating point operations per CPU per clock cycle giving an even greater parallelisation advantage. Previous generations of processors included a maximum of 256bit wide vector registers and so this doubles the computational throughput. This advancement will also be included in Intel's upcoming standard server CPUs so any optimisation for this will be applicable to future non-Xeon Phi CPU based hardware.

3. METHODS

3.1 RTC Software

A large aspect of the time and effort required to design and produce an AO RTC stems from writing the software that controls the whole process. 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 simple 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. The Xeon Phi can run Linux and therefore in a traditional manner it can be programmed for like any other x86 CPU before it, using the same languages and tools readily available and by using the Intel and GNU compilers, programs can automatically be compiled with optimisations for the MIC architecture.

3.1.1 The Durham Adaptive Optics Real Time Controller

The Durham Adaptive Optics Real Time Controller (DARC)[5] 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 be extended for many different AO RTC scenarios such as for different AO regimes like SCAO and MOAO, and even for changing individual algorithms such as pixel calibration and wavefront reconstruction. The modular design also allows it to interface with many different devices for wavefront sensing and wavefront correction,[9] 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. Because of this our investigation will be concerned with optimising the current version of DARC for the Xeon Phi MIC architecture.

3.2 Main areas of Optimisation

As an x86 CPU the Xeon Phi shares many attributes with standard CPU hardware, however it is also very different from previous CPUs with it's many (≥ 64) low power cores, it's high bandwidth memory and the 512bit wide vector registers for improved Single Instruction Multiple Data (SIMD) performance. 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 thread synchronisation - to efficiently make use of the many cores, memory access - to optimise for the fast memory, and vectorisation - to take advantage of the wide vector registers.

3.2.1 BIOS, Operating system and kernel setup

The operating system (OS) installed on the Xeon Phi used in this paper is Centos Linux 7.3,[10] a kernel with a real-time patch from CERN[11] was also investigated but due to some technical issues it was not considered for this report. 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 involve turning off Intel Hyper-threading, which allows more logical threads to execute concurrently on hardware cores, this change was made so that each software thread can be pinned to a single hardware core so that no other processes utilise the same hardware. Other BIOS setting include Xeon Phi specific settings which relate to how the CPU handles memory addressing, information available online,[12] 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 Level-4 cache, or a mixture of the two; these modes are called '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 to and other options relating to CPU interrupts and different power and performance modes.

3.2.2 Multi-threading and synchronisation

Multi-core CPU systems have become the norm in recent years leading to DARC being built on a multi-threaded real-time core using the POSIX[13] 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 get a 'lock' on certain sections of code preventing other threads from accessing protected regions. If a thread calls the lock function on an unlocked mutex variable then that thread will be allowed to proceed and then the mutex becomes locked. Any other threads which attempt to lock this mutex will have to wait at the lock function until the mutex is unlocked. 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 which saves power and frees 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 to no overhead. However for the Xeon Phi MIC architecture with ≥ 64 low power cores, DARC is configured to execute a single thread per core with ¿ 50 threads needed for ELT scale SCAO. This cause problems when using mutexes and condition variables as the constant sleeping and waking of that many threads introduces large amounts of latency. The solution has been to use a structure similar to mutexes called spinlocks, which also protect critical sections of parallel code but instead of sleeping the waiting threads they sit there 'spinning' until they can proceed. This massively reduces the latency with many threads as it is then much faster to resume operation. Condition variables however do not work with spinlocks and so they are replaced where possible by simple volatile flag variables which through careful consideration maintain thread safety.

3.2.3 Vectorisation

The 512bit wide vector registers present on the Xeon Phi allow up to 16 single precision operations to be performed per cycle per CPU core. This is double the previous specification of 256bit 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. The first step is to make sure that array memory is aligned to cache boundaries, this 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[13] function call which aligns the amount of memory required on the specified boundary. The next step is to ensure that sections which can be vectorised such as for loops are written in such a way that the compiler can detect how to vectorise them, there is a guide produced by Intel which details the necessary steps.[14]

3.2.4 Reducing the memory bandwidth requirement

Because the MVM is memory bandwidth bound, due to the relatively simple mathematical operations but large data size, investigating ways to reduce the memory bandwidth dependence is an important consideration. A potential solution is to store the control matrix using half the memory. This is achieved by not storing the control matrix as 32-bit floating numbers which use 4 bytes of memory for each element but by storing it as 16-bit 'half' floats which only use 2 bytes per element. This can result in a loss of precision in the values but in some cases where the precision of wavefront measurements can sufficiently be stored as half-floats it can provide a boost to the maximum frame-rate.

3.3 Best Case ELT SCAO

To complement the results of DARC optimised for Xeon Phi, we also present results of a simple RTC simulator which performs computation of the necessary algorithms, but without the complexities of a full RTC. This is to show the best case performance of an ELT scale RTC on the Intel Xeon Phi hardware. The simple simulator uses the OpenMP API[15] for it's multithreading and performs pixel calibration on fake image data, centroiding of the calibrated pixels, an MVM reconstruction of the centroids and finally introduces a gain to the final result. There is no pipelining of the data here and no camera interface so all centroids 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.

4. RESULTS

The performance of an AO RTC is generally defined by the time taken for it to process each frame, where a frame is single image from a WFS and the processing involves calibrating the pixels, computing the centroids and reconstructing the wavefront before sending the results to a correcting element. There are different ways of determining 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 core 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.

The results presented will use this method of timing and for situations where there is no real camera, and therefore no pipelining, the results are timed from the beginning of pixel calibration to when the final DM commands are ready. The main performance metric used to define the latency of the RTC will be average time taken to process a frame and will be referred to as the frametime. Where applicable the performance will also be quoted as the average framerate which in this case is equal to the inverse of the average frametime because each frame is processed independently and in sequence. To define the jitter, the RMS error of the framerate will be used quoted with the average frametimes.

4.1 SCAO best case simulator

Figure 3 shows the frametime results of the SCAO best case simulator, the figure shows the minimal number of outliers and also the small spread of the distribution. The average frametime $0.77 \pm 0.02ms$ is which corresponds to an average framerate of $1300 \pm 30Hz$.

4.2 DARC SCAO computation

Figure 4 shows the frametime results of the DARC simulating ELT scale SCAO without a real camera attached, the figure shows a small number of regular single frame outliers and also the small spread of the bulk of the distribution. The average frametime is $0.905 \pm 0.009ms$ which corresponds to an average framerate of $1110 \pm 10Hz$.

4.3 DARC SCAO computation with camera

Figure 5 shows the frametime results of DARC simulating ELT scale SCAO with pixels arriving from a real gigE vision based camera running at the camera's maximum framerate of 966Hz. The figure shows the minimal number of outliers and also the small spread of the distribution. The average frametime is $1.04 \pm 0.01ms$ which corresponds to an average framerate of $970 \pm 10Hz$.

5. DISCUSSION

5.1 Suitability of Xeon Phi for ELT scale AO RTC

The results of the best case SCAO simulator in Figure 3 show that the Xeon Phi Knights landing is capable of performing the computational requirements for ELT-scale SCAO with an average frame time of $0.77 \pm 0.02ms$. This meets the requirement of exceed the minimum atmospheric coherence time for ELT sites which is on the order of 1.0 ms. The RMS jitter, which is taken as the standard error on the mean frametime, is $16.3\mu s$ which is much less than the preliminary requirement of $100\mu s$ for the entire AO system, giving plenty of room for potential jitter from other aspects of the AO system. The instantaneous peak-to-peak jitter, which is measures as the maximum difference in frametime between subsequent frames is $107\mu s$, however this does include the 3 outliers at the beginning of the data set which would not necessarily be present in a full AO system with more investigation into reducing systematic jitter. Without this value, the instantaneous peak-to-peak jitter becomes $88.8\mu s$ which now falls just below the preliminary requirement of $100\mu s$.

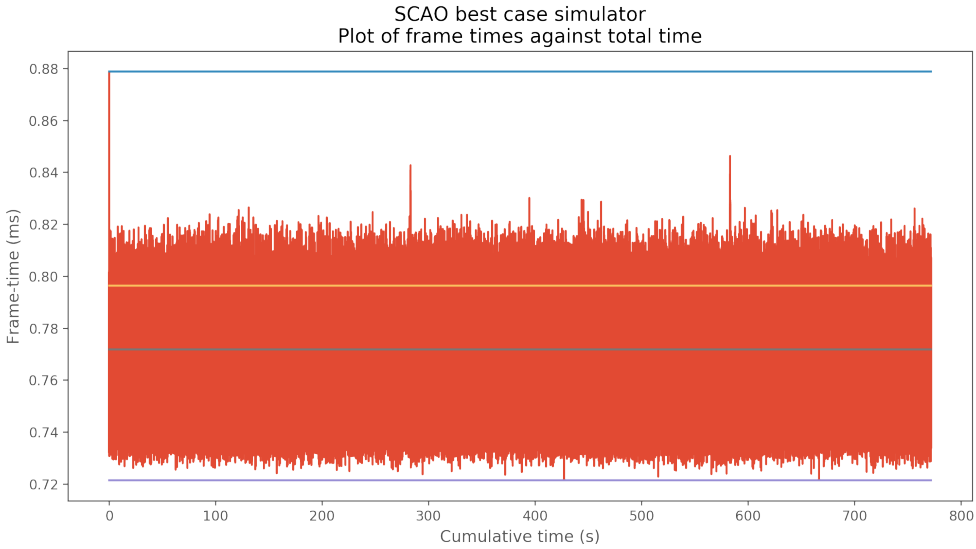

Figure 3. Frametime results of SCAO best case simulator for 10^6 frames with an average frametime of $0.77 \pm 0.02ms$ which corresponds to an average framerate of $1300 \pm 30Hz$. The horizontal lines from top to bottom are: maximum frametime, 99th percentile, average frametime and minimum frametime.

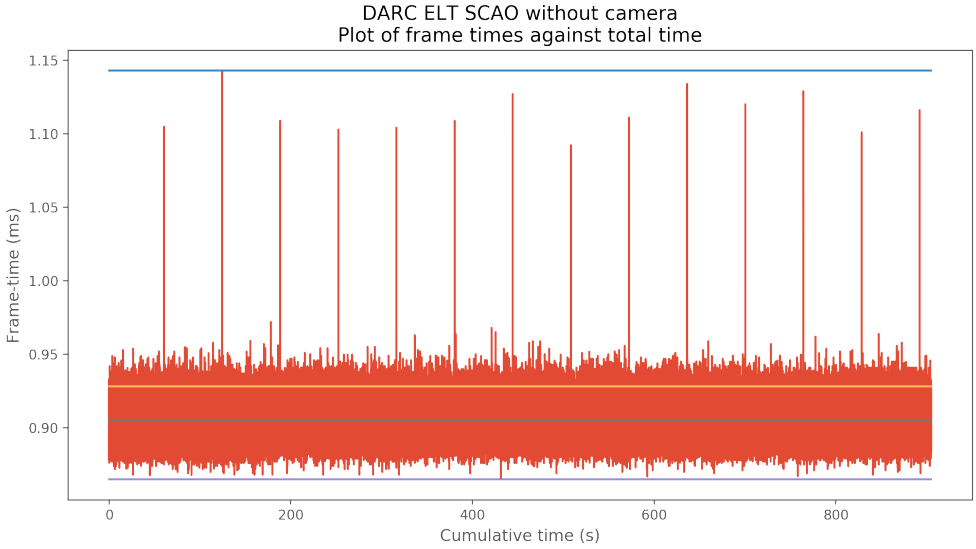

Figure 4. Frametimes for DARC SCAO no camera for 10^6 frames with an average frametime of $0.905 \pm 0.009ms$ which corresponds to an average framerate of $1110 \pm 10Hz$. The horizontal lines from top to bottom are: maximum frametime, 99th percentile, average frametime and minimum frametime.

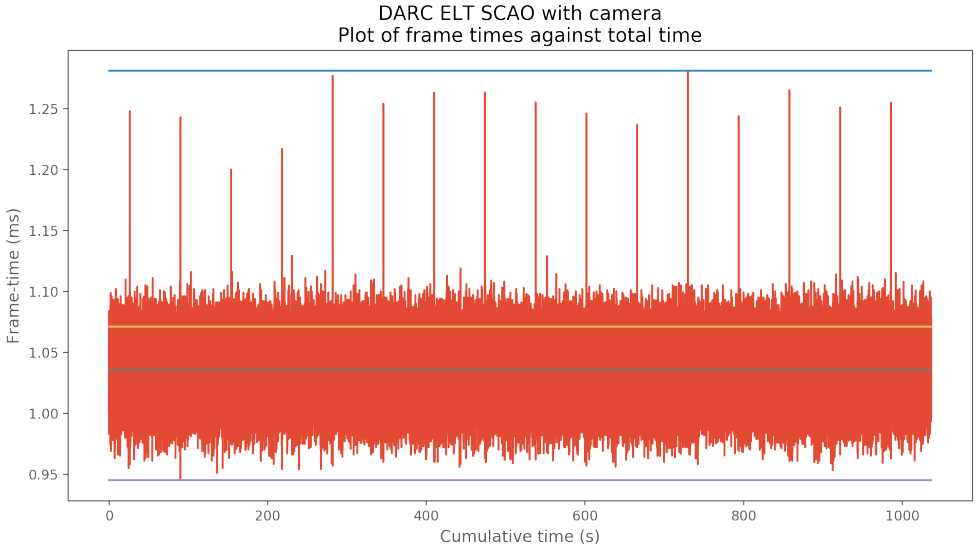

Figure 5. Frametimes of DARC SCAO with camera for 10^6 frames with an average frametime of $1.04 \pm 0.01ms$ which corresponds to an average framerate of $970 \pm 10Hz$. The horizontal lines from top to bottom are: maximum frametime, 99th percentile, average frametime and minimum frametime.

5.1.1 DARC on Xeon Phi for ELT scale AO RTC

The results described in Sections 4.2 and 4.3 for a mature AO RTC on the Xeon Phi are consistent with those found in Section 4.1. The frametimes for DARC with no camera, as shown in Figure 4, give the current best performance of the computation along with the other hard-real time aspects which are required for a real RTC. The average frametime of $0.905 \pm 0.009ms$ still comes in below the average minimum atmospheric coherence time on the order of $1.0ms$. This results included the regular outliers as seen in Figure 4 which occur every 64 seconds during the measurement period; these outliers are presumed to be due to an interrupt running on the CPU due to the period in seconds being equal to number of hardware cores. Due to the relatively few outliers in the sample (14 in 10^6) ignoring these values makes no discernible difference to the average frametime and RMS jitter of $9.22\mu s$ however it does decrease the instantaneous peak-to-peak jitter from $249\mu s$ to $73.9\mu s$, now falling below the preliminary requirement of $100\mu s$.

For DARC running with a real camera attached, results in Section 4.3, the frametime results in Figure 5 show similar structure to the results for DARC without a camera as shown in Figure 4. The results include the regular outliers that occur once every 64 seconds which are presumed to be due to an interrupt running on the CPU. The average frametime is $1.04 \pm 0.01ms$ which corresponds to an average framerate of $970 \pm 10Hz$, comparable to the maximum camera framerate of $966Hz$. This framerate seems slightly higher than the camera's framerate due to rounding. This results shows that DARC on the Xeon Phi can keep up with a real camera running at nearly 1kHz which is the likely target for ELT-scale SCAO. The RMS jitter is $13.8\mu s$ which is on the same order as the RMS jitter of DARC with no real camera. Similar to the results of darc without a real camera the average frame and RMS jitter are unaffected by removing the regular outliers due to the relatively few outliers in the sample. The instantaneous peak-to-peak jitter however does change with the removal of the outliers, from $304\mu s$ to $163\mu s$ which in this case does not fall below the preliminary requirement of $100\mu s$ but as this exact hardware and camera is very unlikely to used in a real AO system, the average framerate is the most important metric to show that hardware of this type is capable of performing AO RTC at ELT scales.

5.2 Future Work

The optimisations of DARC presented in this paper are far from exhaustive, they focus on the multi-threading and reconstruction aspects and provide us with adequate performance for ELT SCAO. However they can still be improved upon by further investigation[16] into other aspects of the real-time pipeline such as the pixel calibration

and centroiding parts. There are also more novel algorithms in use in AO which could benefit from acceleration by Xeon Phi such as more complex reconstruction techniques like Linear Quadratic Gaussian (LQG) reconstruction and less computationally expensive techniques such as the Cumulative Reconstructor with domain decomposition (CuRe-D).

Another area where Xeon Phis may be applicable is in AO simulation for ELT scale. Currently full ELT scale simulations are not capable of being run even close to real-time,[17] so that they require running ahead of time and then the results can then be analysed once the simulations are finished. This is not such a problem now before the ELTs are built, but when they are operational it will be very helpful to have full simulations which can be run in real-time alongside actual observations, to help better understand the atmospheric conditions and to better tune the parameters of the AO system for optimal performance.

5.2.1 MAORY prototype

This paper has only discussed the SCAO regime of AO for ELT-scale but the next generation telescopes all have more complex multi-conjugate or multi-object AO systems planned for which suitable RTCs will be required. One of these systems is the ELTs Multi-conjugate Adaptive Optics Relay (MAORY) which will employ 6 Laser Guide Stars (LGS) along with three Natural Guide Stars (NGS) using 4 DMs for the correction, increasing the computation requirements by more than 6 times that of ELT SCAO. The preliminary design[18] for this involves multiple Xeon Phi systems working in parallel to process the images from each WFS independently before combining the reconstructions and delivering to the DMs. This presents significant challenges in addition to those for preparing DARC for ELT-scale SCAO.[16]

5.3 Conclusion

The results presented show that the Intel Xeon Phi is capable of performing the computational requirements of a full on-sky tested AO RTC for ELT scale with frame-rates and latencies that meet or exceed the current estimated preliminary requirements. Further more it has been shown that with a real camera, the RTC is capable of performing at similar frame-rates albeit with somewhat increased jitter which isn't seen as a problem at this stage. A more detailed analysis is in progress[16] and further work will take place to extend this research to other areas of AO RTC.

ACKNOWLEDGMENTS

This work builds upon previous investigation into AO RTC on Xeon Phi hardware conducted by David Barr.

Real-time AO work by the Durham group is supported by Green Flash and the STFC Consolidated Grant, and my own effort by an STFC PhD studentship and Durham University.

REFERENCES

[1] Babcock, H. W., "The Possibility of Compensating Astronomical Seeing," *Publications of the Astronomical Society of the Pacific* **65**, 229 (1953).

[2] Max, C., "Introduction to adaptive optics and its history," (2001). [Online; accessed 27-September-2017].

[3] Fedrigo, E., Donaldson, R., Soenke, C., Myers, R., Goodsell, S., Geng, D., Saunter, C., and Dipper, N., "Sparta, the eso standard platform for adaptive optics real time applications - art. no. 627210," (07 2006).

[4] Rodríguez-Ramos, L. F., Chulani, H., Martín, Y., Dorta, T., Alonso, A., and Fuensalida, J. J., "FPGA-based real time controller for high order correction in EDIFISE," in [*Adaptive Optics Systems III*], **8447**, 84472R (jul 2012).

[5] Basden, A., Geng, D., Myers, R., and Younger, E., "Durham adaptive optics real-time controller.," *Applied optics.* **49**, 6354–6363 (November 2010).

[6] Truong, T. N., Bouchez, A. H., Burruss, R. S., Dekany, R. G., Guiwits, S. R., Roberts, J. E., Shelton, J. C., and Troy, M., "Design and implementation of the palm-3000 real-time control system," *Proc.SPIE* **8447**, 8447 – 8447 – 9 (2012).

[7] Intel, "Intel xeon phi processors," (2017). [Online; accessed 27-September-2017].

[8] Barr, D., Basden, A., Dipper, N., and Schwartz, N., "Reducing adaptive optics latency using xeon phi many-core processors," *Monthly Notices of the Royal Astronomical Society* **453**(3), 3222–3233 (2015).

[9] Basden, A., "The durham ao real-time controller and the canary implementation," in [*Imaging and Applied Optics*], *Imaging and Applied Optics* , AMB1, Optical Society of America (2011).

[10] The-CentOS-Project, "Centos linux," (2001). [Online; accessed 27-September-2017].

[11] CERN, "Linux @ cern," (2016). [Online; accessed 27-September-2017].

[12] Intel, "Memory modes and cluster modes: Configuration and use cases," (2015). [Online; accessed 27-September-2017].

[13] 'The-Open-Group', "The open group base specifications issue 7," (2016). [Online; accessed 27-September-2017].

[14] Intel, "A guide to auto-vectorization with intel c++ compilers," (2012). [Online; accessed 27-September-2017].

[15] OpenMP Architecture Review Board, "OpenMP application program interface version 4.5," (2015). [Online; accessed 27-September-2017].

[16] Jenkins, D., Basden, A., and Myers, R., "Elt-scale adaptive optics real-time control with the intel xeon phi many integrated core architecture," *in preparation for MNRAS* (2017).

[17] Basden, A., "A real-time simulation facility for astronomical adaptive optics," **439**, 2854–2862 (Apr. 2014).

[18] Jenkins, D., Basden, A., and Myers, R., "Elt-scale mcao real-time control preliminary design for intel xeon phi mic architecture," *in preparation* (2018).