

PROCEEDINGS OF SPIE

[SPIDigitalLibrary.org/conference-proceedings-of-spie](https://spiedigitallibrary.org/conference-proceedings-of-spie)

Results of the NFIRAOS RTC trade study

Jean-Pierre Véran, Corinne Boyer, Brent L. Ellerbroek, Luc Gilles, Glen Herriot, et al.

Jean-Pierre Véran, Corinne Boyer, Brent L. Ellerbroek, Luc Gilles, Glen Herriot, Daniel A. Kerley, Zoran Ljusic, Eric A. McVeigh, Robert Prior, Malcolm Smith, Lianqi Wang, "Results of the NFIRAOS RTC trade study," Proc. SPIE 9148, Adaptive Optics Systems IV, 91482F (21 July 2014); doi: 10.1117/12.2057323

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2014, Montréal, Quebec, Canada

Results of the NFIRAOS RTC trade study

Jean-Pierre Véran^a, Corinne Boyer^b, Brent L. Ellerbroek^b, Luc Gilles^b, Glen Herriot^a, Daniel A. Kerley^a, Zoran Ljusic^a, Eric A. McVeigh^c, Robert Prior^c, Malcolm Smith^a, Lianqi Wang^b

^aNational Research Council Canada, 5071 W. Saanich Rd., Victoria, BC Canada V9E 2E7

^bTMT Observatory Corp., 1111 S. Arroyo Parkway, Pasadena, CA, USA 91101

^cUniversity of Victoria, 3800 Finnerty Rd, Victoria, BC Canada V8P 5C2

ABSTRACT

With two large deformable mirrors with a total of more than 7000 actuators that need to be driven from the measurements of six 60x60 LGS WFSs (total 1.23Mpixels) at 800Hz with a latency of less than one frame, NFIRAOS presents an interesting real-time computing challenge. This paper reports on a recent trade study to evaluate which current technology could meet this challenge, with the plan to select a baseline architecture by the beginning of NFIRAOS construction in 2014. We have evaluated a number of architectures, ranging from very specialized layouts with custom boards to more generic architectures made from commercial off-the-shelf units (CPUs with or without accelerator boards). For each architecture, we have found the most suitable algorithm, mapped it onto the hardware and evaluated the performance through benchmarking whenever possible. We have evaluated a large number of criteria, including cost, power consumption, reliability and flexibility, and proceeded with scoring each architecture based on these criteria. We have found that, with today's technology, the NFIRAOS requirements are well within reach of off-the-shelf commercial hardware running a parallel implementation of the straightforward matrix-vector multiply (MVM) algorithm for wave-front reconstruction. Even accelerators such as GPUs and Xeon Phis are no longer necessary. Indeed, we have found that the entire NFIRAOS RTC can be handled by seven 2U high-end PC-servers using 10GbE connectivity. Accelerators are only required for the off-line process of updating the matrix control matrix every ~10s, as observing conditions change.

Keywords: Adaptive optics, real-time computing, real-time control

1. INTRODUCTION

The Narrow Field IR Adaptive Optics System (NFIRAOS) is the first light facility adaptive optics (AO) system for the Thirty Meter Telescope (TMT) [1]. The NFIRAOS project is currently at the final design stage [2]. NFIRAOS is a multi-conjugate AO system. It has two large deformable mirrors with a total of more than 7000 actuators. For most observations, these deformable mirrors will be driven from the measurements of six 60x60 laser guide star (LGS) wave-front sensors (WFSs). These WFSs produce a total of 1.23Mpixels at 800Hz, from which the NFIRAOS real-time controller (RTC) needs to compute updated DM commands with a latency of less than one frame (1.25ms). This task represents quite a computational challenge.

In 2008-2009, TMT commissioned a conceptual design study for a Real Time Control (RTC) system. Two groups carried out independent studies and both proposed custom-designed solutions based on Field Programmable Gate Arrays (FPGAs) that could meet the requirements [3][4].

In 2013, TMT commissioned a new trade study to re-evaluate the conclusions from 2009 in light of the rapidly evolving technology that has become available since then, with the plan to select a baseline architecture by the beginning of NFIRAOS construction expected for 2014.

This paper presents the results of this trade study. Section 2 presents the RTC requirements and the criteria used to evaluate different potential solutions. Section 3 summarizes the different possible RTC algorithms. In section 4, we present the different hardware that we have investigated, which includes PC-servers with and without accelerators, AdvancedTCA and Open VPX. In section 5, we discuss our efforts at mapping the algorithms on each different type of hardware, and in section 6 we present the different candidate architectures that we have designed, for each type of

hardware. In section 7, we summarize our effort to verify some of these architectures through benchmarking, and in section 8, we present our evaluation of each of our candidate architectures. Our conclusions are presented in section 9.

2. TRADE STUDY CRITERIA

2.1 NFIRAOS RTC top-level requirements

NFIRAOS is a Multi-Conjugate AO system, which includes:

- two Deformable Mirrors (DMs) conjugated at 0km (DM0 – 3125 actuators) and 11.2km (DM11.2 – 4548 actuators),
- one Tip/Tilt Stage (TTS) serving as the mount for DM0,
- six Laser Guide Star (LGS) Shack-Hartmann wavefront sensors (WFSs) of order 60x60 sub-apertures
- up to three low order Infrared natural guide star wavefront sensors (OIWFS) within each NFIRAOS instrument,
- one high order visible Natural Guide Star Shack-Hartmann WFS (NGS WFS) of order 60x60, which is used for operation without LGS,
- one Truth Wavefront Sensor (TWFS) measuring a natural guide star at low bandwidth, which is used to calibrate for slow-varying biases due to temporal variations in the sodium layer profile in LGS AO mode,
- and the RTC, which processes the inputs from the various WFSs to compute the commands of the deformable mirrors and tip/tilt stage.

The RTC interfaces with additional telescope and AO sub-systems, including the AO Sequencer, the NFIRAOS Component Controller, the Laser Guide Star Facility System, the NFIRAOS instruments, and the Data Management System (DMS).

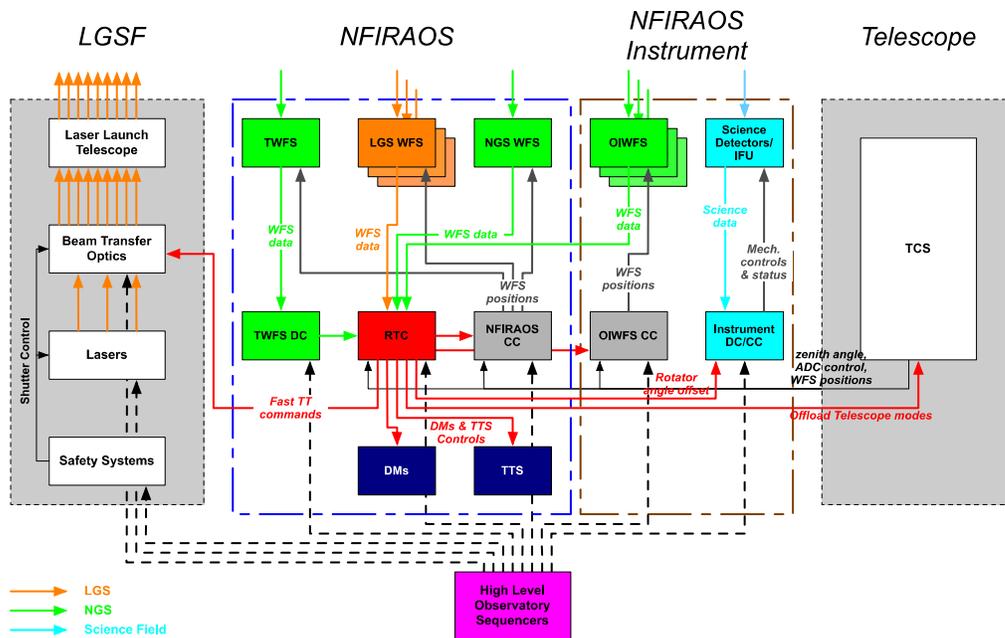


Figure 1: NFIRAOS top-level block diagram

In LGS mode, which is by far the most computationally demanding mode, the RTC processes 1.23Mpixels (2 bytes/pixel) from the six LGS WFS cameras and outputs a total of 7673 DM commands at a rate of 800 Hz. The main performance requirement for the NFIRAOS RTC is to have a total latency of no more than 1.2ms with a goal of 0.6ms, knowing that the LGS WFS CCDs are read in 0.5ms. Here latency is calculated from the time the WFS CCD read-out starts to the time all the DM commands are output to the DM Drive Electronics (DM). Since the DME is allocated 0.05ms to output the commands to the DM, the requirement corresponds to a full one frame latency, and the goal corresponds to one-half frame.

All the parameters required by the RTC are provided by the Reconstructor Parameter Generator (RPG), which is part of the AO Sequencer, not the RTC. These parameters are optimized and updated as conditions change. However, some parameters requiring frequent updates such as the matched filters for pixel processing, are optimized within the RTC.

The full list of requirements for the NFIRAOS RTC is recorded in a Design Requirement Document (DRD).

2.2 Trade study criteria description

The trade study criteria are summarized in Table 1 and Table 2. They were set in advance of carrying out the trade study, in order to achieve the most unbiased comparison. The most important items in the DRD are specifically included in the criteria, and others are lumped into a generic line item: “compliance with requirements with minimal waivers.”

The columns of these tables are from left to right:

- Criterion, which is often a key requirement in the DRD
- Requirement itself that must be met – ideally a quantifiable and measurable value
- Goal is a tighter version of the requirement, which indicates desirable improvements over the requirements, if they can be done for low cost.
- Weighting qualitatively ranks the importance of the criteria in three bands: high, medium and low.

In Table 1 we present the very highest priority criteria. Table 2 contains items that are considered of lesser importance, even if the weighting is currently shown as high.

Table 1 Highest Priority Criteria for RTC trade-off study

Criterion	Requirement	Goal	Weighting
Accuracy	40 nm RMS WFE relative to FD-PCG3	20 nm	High
Latency (start of CCD readout to end of transfer to DME)	1.2 ms	0.6 ms	High
Latency jitter	No missed frames. Jitter <TBD	TBD	Medium
Availability (reliability)	0.1% loss of expected 3200 h science per year		High
Cost	Minimize		High
Schedule -- Delivery Date	Meets NFIRAOS schedule	6 months early	High
Risks (technical, schedule and costs)	medium	low	High
Hardware Maintainability	10 yr lifetime	15 yr	High
Software Maintainability	Acceptable to require a Specialist	General programming skills sufficient	High
Space	24 U		Medium
Power	1500 W		Medium

Table 2 Moderate Priority Criteria for RTC trade study

Criterion	Requirement	Goal	Weighting
Upgradability	NFIRAOS+	MOAO	Low
Ease of Verification	Module I/O comparison with MAOS	RTC testable with MAOS	Medium
Compliance with all requirements	Minimal waivers	No waivers	High
Telemetry	Telemetry is one single criterion		Low
Rate	3.5 GB/sec	5.0 GB/sec	
Data Size	90 TB	140 TB	
Rack Space, power	60U, 6 kW		
Updating RTC from RPG	No disturbance to closed loop		High
Observing Overhead	Overhead is one single criterion		Medium
Initializing Dead time	10 sec (TBC)	No better goal	
RPG load - update ready in time for dither	10 sec cadence	No goal	
Dithering dead time	5 asec in < 2 sec	No goal	
Flexibility	Software	Algorithms	Medium
Quality of Interfaces	Standards based	Industrial	High
Standards	Standards is one single criterion		Medium
Hardware standards compliance			
Quality of standards and documentation			
Maturity of standards and documentation			

The accuracy is the quadrature difference between the residual wave-front error achieved by the chosen algorithm on the chosen architecture, compared to the lowest wave-front error we can achieve in simulation (in median conditions). Reasons that can degrade the accuracy include rounding errors and/or resorting to an inferior algorithm.

The latency measures the elapsed time between the start of WFS readout and the end of sending the DM commands to the DM electronics. The latency includes the CCD read-out, which is 500us. Reducing the latency towards the goal will reduce servo-lag errors in NFIRAOS. Furthermore, short latency algorithms have the potential to permit longer readout times on the WFS CCD and therefore less read noise. They also can reduce data rates on the interface between WFSs and the RTC, possibly saving money or reducing risk.

For the jitter in latency between frames, the requirement is to not miss any frame. At this time, we are still evaluating whether this requirement should be tightened, or whether it could be loosened to allow missing a frame from time to time.

For reliability, no more than 0.1% of the 3200 hours (i.e. 3 hours) per year planned for scientific observing can be lost due to an RTC failure. What needs to be compared with this requirement is the product of the mean time between failures (MTBF) and mean time between repairs (MTTR) integrated over a year. We categorize failures in two types: those that can be fixed remotely (e.g. by rebooting, swapping servers, or changing observing mode) that, in our estimation, typically cost 30 minutes of downtime; and those that need the intervention of the day crew, which can only occur on the following day, for which the remaining of the night is lost. The estimate also takes into account the fact that NFIRAOS will only be used half of the nights at TMT on average.

The cost includes all the spares required to operate throughout the lifetime of NFIRAOS. It must, of course, be minimized.

The schedule of the NFIRAOS RTC must be such that it does not impact the development of the rest of NFIRAOS, with a goal of delivering 6 months in advance.

The technical risk is assessed according to the risk assessment framework developed at TMT. Using commercially available hardware with well-established costs and development methods reduces the risk.

Maintainability quantifies how difficult it will be to maintain the RTC hardware and software. This includes how difficult it is to diagnose a problem, and how difficult it is to correct it. For hardware, the quantity of spares and their expected availability is a factor, since spares can deteriorate with time. As well, heterogeneous hardware makes maintainability more difficult. For software, the requirement is that the software can be changed by a specialist, e.g. a GPU or FPGA programmer. The goal is that this can be done by general-purpose programmers, so that the observatory does not need to carry specialists on staff. This goal is enabled by general-purpose programming languages, and homogeneous development and operating environments.

For space, 24U is the current allocation for the RTC in the NFIRAOS electronic cabinet. However, a small amount of spare room exists, which could be allocated to the RTC. Hence this criterion has only a medium weighting. Same for the 1500W power budget, which comes from a top-level allocation split between the different NFIRAOS components.

In terms of upgradability, it is slightly desirable that the selected RTC design be upgradable for use in future order 120x120 NFIRAOS+, or MOAO system.

In terms of ease of verification, it would be desirable that the RTC could be interfaced directly with MAOS, the TMT AO simulation software, for testing, but it is required that accuracy tests be conducted with input and output files provided by MAOS.

For compliance, the requirement is that the RTC only needs very few waivers of requirements. In some cases, an otherwise attractive option may need more waivers to be used in NFIRAOS. But it would receive a lower score on this item. The goal is to meet all the requirements with no waivers

For updating the RTC in real time, the requirement is that downloading and swapping parameters (e.g. matrices) from the Reconstructor Parameters Generator (RPG) into the RTC shall take place without disturbing the closed loop. I.e. updating parameters should not cause jitter or lost cycles.

The “Observing overhead” has three components: (i) no more than 10s to initialize a new observation; (ii) no more than 10s to be ready for a <30 arcsec dither; and (iii) no more than 2 seconds dead time during a jitter of up to 5 arcsec.

For the “Flexibility criterion”, the requirement is that the software may be changed. The goal is that entire algorithms may be replaced. The intent is that the RTC may be revised as we learn more during the construction, commissioning and operation of NFIRAOS. New attractive algorithms may come along. Systems that can only support one type of algorithms are penalized.

The requirement is that the interfaces to WFSs, DMs, Telemetry, and RPG follow standards-based approaches, rather than custom protocols, formats, connectors etc., specific to NFIRAOS. The goal is that these are industrial standards, as opposed to consumer-product standards, which are not generally as reliable, nor as long-lived in the marketplace. This criterion will impact uptime of NFIRAOS as well as the long-term availability of knowledgeable staff, and the ability to procure spares if needed.

Use of standards is one single criterion with medium weight, and has three components.

- Hardware standards compliance: as above, the requirement is that all of the RTC, not just interfaces be designed using standards-based approaches, rather than custom protocols, formats, connectors etc., specific to NFIRAOS.
- Quality of standards and documentation: as above, this criterion will impact the long-term availability of knowledgeable staff, and the ability to procure spares if needed. Furthermore ideally these standards should be open and accessible. Documentation of some standards is more comprehensive and readable than others, which will affect the maintainability of NFIRAOS. Tutorial and reference documentation from a variety of sources is valuable.
- Maturity of standards and documentation: well proven standards, with extensive field usage and wide acceptance help ensure reliability of designs, and multiple sources of supply. With such field experience comes a variety of tutorial information, and more complete and accurate reference documentation. A good trade-off is required between too new (not quite mature yet) or too old (soon to be obsolete) standards.

3. RTC ALGORITHMS

In LGS mode, the most computationally intensive tasks consist of two steps:

- Pixel processing, which takes in the WFS pixels and computes the position of the spots in each sub-aperture (slopes)
- Wave-front reconstruction, which takes in the slopes and computes the new commands to be applied to both DMs

In the NFIRAOS architecture, pixel processing involves correcting each pixel for dark current and flat field, and applying a matched filter to the image of each sub-aperture to produce the X and Y slopes. The total number of sub-apertures to be processed is 17,376, with between 6x6 and 6x15 pixels per sub-aperture (depending on the location of the sub-aperture). This corresponds to about 1.23Mpixels per frame, with two bytes per pixel. This results in $17,376 \times 2 = 34,752$ slopes.

Wave-front reconstruction involves reconstructing the incoming wave-front on 6 layers located at different altitudes above the telescope (tomography step), and deriving the 7,673 new commands to be applied to both DMs (fitting step). The slopes s are related to the DM commands via a linear equation: $s = Ga$, where G is a 34,752 row by 7,673 column matrix.

The tomography step is performed by a minimum variance reconstructor that uses as prior the expected covariance matrix of the incoming atmospheric turbulence. The fitting step is a projection that optimizes performance (wavefront variance and Strehl ratio) averaged over the specified science field of view (FoV).

The wave-front reconstruction consists in inverting G in the minimum variance sense. Two classes of algorithms have been selected to solve this problem. The first one is a straightforward matrix-vector multiply MVM of the form $a=Es$, where E is an inverse matrix that encapsulates both tomography and fitting steps. The second class comprises several iterative algorithms make use of the fact that (i) the G matrix is very sparse and (ii) the covariance of the turbulence can be approximated by a block diagonal matrix. During the course of the NFIRAOS design, we have formulated three different algorithms for the tomography step: the Conjugate Gradient (CG), the Fourier Domain Preconditioned Conjugate Gradient (FDPCG), and the Block-Gauss Seidel with inner Conjugate Gradient (BGS-CG). For the fitting step, which involve significantly fewer operations than the tomography step, we have only considered the CG. Describing these algorithms in details is well beyond the scope of this paper. However Table 3 provides a top level summary of the number of operations and memory footprint that these algorithms require. For the iterative algorithms, the number of iterations was chosen as the minimum required to achieve proper accuracy: 30 iterations for CG (CG30), 3 iterations for FDPCG (FD3), one iteration of BGS with 20 iterations of CG for each block (BGS-CG20) and 4 iteration for CG in the fitting step (CG4).

Table 3 Number of operations (in Million of Multiply and Accumulates per frame) and memory footprint of the different wave-front reconstruction algorithms, and for different layer samplings. “total” means the sum of tomography (tomo) and fitting.

1	Algorithm	(MMACs/frame)		Memory (MB)	
2	B	C	D	E	F
3	Nos (nb of oversampled layers)	2	6	2	6
4	CG30 tomo	166.4	218.7	10.6	
5	FD3 tomo	72.8	181.4	20.4	48.1
6	BGS-CG20 tomo	42.2	98.8	47.5	68.9
7	CG4 Fitting	17.9		44.6	
8	CG30 total	184.3	236.6	55.2	
9	FD3 total	90.6	199.2	64.9	92.7
10	BGS-CG20 total	60.1	116.7	92.0	113.5
11	MVM total	237.2		909.6	

We see that MVM has the largest requirements both in terms of memory and in terms of number of operations. This is because the reconstruction matrix E is fully dense. Iterative algorithms require an order of magnitude or less memory:

CG30 requires the least memory, but about the same number of operations as MVM, whereas BGS-CG20 requires the least amount operations, but somewhat more memory.

4. HARDWARE ARCHITECTURES

We have investigated three different hardware architectures: PC servers, AdvancedTCA and Open VPX.

4.1 PC server-based architectures (CPUs, GPUs and Xeon Phis)

PC servers are widely available for consumer and industrial applications. CPUs are the heart of PC servers, and thus the heart of modern computing. CPUs typically include several cores, as many as 12 on modern machines, and each core can execute 8 floating point operations simultaneously in one clock cycle. High-end modern CPUs typically run at a clock rate of 2.7GHz, so the maximum theoretical computation speed is $2.7 \times 12 \times 8 = 259$ GFlops (Giga Floating point operations per second – one such operation can be a multiply and accumulate or MAC). In practice however, such a computation rate can be seldom achieved because the computation coefficients cannot be brought from the memory to the compute registers at that speed. CPUs can access a limited amount of on-board memory (cache) with very high transfer rate and very low latency, but the size of this memory is limited. Data not in the cache must be brought in from the main external memory. This significantly reduces the transfer rate, and increases the latency. Table 4 shows the characteristics of the E5-2600, which is the high-end series of Intel CPUs. The current version is v2 and it has 30MB of L3 cache (shared between all the cores) and a maximum bandwidth to the main memory of 59.7 GB/sec. The table also shows the characteristic of the previous version, and what is expected from the future version, showing a clear trend of increased number of cores, increased cache size and increased memory bandwidth. Note that the much more expensive but higher grade E7 series provides somewhat higher performance, and could be used if needed.

Table 4: Intel E5-2600 Evolution

E5 Version	v1	v2	v3	v4
Design Name	Sandy Bridge	Ivy Bridge	Haswell	Broadwell
Release Date	Q1, 2012	Q3, 2013	Exp. Q3, 2014	2015
Max. Cores	8	12	14	18
Max. L3 Cache	20 MB	30 MB	35 MB	45 MB
RAM	DDR3-1600	DDR3-1866	DDR4-2133	DDR4-2400
RAM Bandwidth	51.2 GB/sec	59.7 GB/sec	68.2 GB/sec	76.8 GB/sec
Manufacturing Process	32 nm	22 nm	22 nm	14 nm

To increase the compute power, CPUs can be supplemented with accelerator cards, to which the CPUs can offload calculations. GPUs are the most common accelerators. They connect to the CPU via the PCI-Express bus. GPUs are usually programmed in CUDA. They pack in a single chip many compute engines, can offer computing power and memory bandwidth significantly higher than CPUs. For example, the Tesla K20X can achieve 3.95 TFlops of peak single precision processing power, and the bandwidth to the 6GB on-board device memory can reach 250 GB/s (a limited amount of faster cache memory is also available). A potential bottleneck for GPUs is the PCI-Express (PCI-E) bus, which needs to be used to bring data onto the GPU, and has a bandwidth of only 16 GB/s (PCI-E 3.0) and a latency of ~11us per transfer.

Another type of accelerators that we have considered is the Intel Xeon Phi. The Xeon Phi also connects to the host CPU via the PCI-Express bus. However, their architecture is more like that of a CPU with a higher number of cores and higher memory bandwidth but reduced clock speed. In our study, we used the Xeon Phi 7120P, which was the top of the line in the summer 2013. It has 61 cores running at 1.238 GHz, 30.5 MB of cache, 16GB of RAM with a maximum bandwidth of 352 GB/s. The maximum theoretical processing power for single precision floating point operations is 2.4 TFlops, so comparable to that of a GPU.

4.2 AdvancedTCA-based architectures

AdvancedTCA is a computing standard primarily geared at the telecommunication industry. AdvancedTCA systems consist of a number of printed circuit boards (or blades) hosted in a dedicated chassis (or shelf) that provides a very high level of connectivity between the blades, via backplanes offering point-to-point connections (no data bus).

The main reason for investigating AdvancedTCA architectures in the context of the NFIRAOS RTC is, beyond the intrinsic capabilities of these architectures, to benefit from the recent development at NRC of the Kermod Board. The Kermod XV6 is the most powerful AdvancedTCA compute blade ever built. It has been specifically designed to tackle the most demanding signal processing applications that exist, and its primary application will be the real-time digital processing of signals detected by the future radio-telescopes (correlators). The Kermod XV6 packs eight Xilinx Virtex-6 SX475T FPGAs, delivering an outstanding 8.8 TeraMACs solely from their DSP48E1 dedicated multiply-accumulate engines. Each FPGA interfaces with two DDR-3 SDRAM SODIMM modules, capable of supporting up to 4 GBytes, for an aggregate memory capacity of 64 GBytes. The bandwidth to this memory is limited however, to a maximum 6.4GB/s per memory module, or 12.8GB/s per FPGA. The blade connects with the backplanes of the AdvancedTCA shelf at rates exceeding 500 Gb/s, making for highly-efficient clusters, with up to 128 FPGAs in a single chassis. Within the Kermod blades, direct communications between FPGAs can be established, at bandwidth up to 2 GB/s.

4.3 Open VPX-based architectures

Open VPX is a relatively new standard that replaces VME and VXS. It is mainly targeted to defense applications and can operate tough environments (temperature, shock and vibrations), with the ability to replace modules in the field (two level maintenance). It has also a high bandwidth density and is based on robust standards. Just like the AdvancedTCA standard, a VPX computers consists of a chassis with a high speed backplane, which supports various computing engines such as CPUs, GPUs and FPGAs, in the form of computing blades.

4.4 10GbE connectivity

In addition to the main hardware, the connectivity between all the components needs to be resolved. It is advantageous to define a single data transfer protocol for communications between all the different NFIRAOS modules, as it makes prototyping and testing easier. The main requirements for selecting a common protocol are:

- Low latency
- Availability and industry support.
- Well defined evolution road map defined
- Ability to implement at reasonable cost.
- Reasonable total cost of ownership

The two data protocols that are capable of fulfilling the task and are supported by the industry are 10Gbps Ethernet (10GbE) and FDR-10 Infiniband (Fourteen Data Rate).

Infiniband data protocol operates at extremely low latency, 100ns in some cases. This feature is important in High Performance Computing systems and data centers.

On the other hand 10GE is not as good as infiniband when latency is considered but with a few hundreds of nanoseconds delay it could be used in NFIRAOS. These delays are acceptable when RTC operates at 800MHz. This data protocol is most widely utilized in today's telecom and computing industry. There are plenty of vendors producing switches, routers, adapter and cables to select from. New generations, 40GE and 100GE, are defined and are already being used. Finally 10GbE cost per port is less than Infiniband.

Our decision was to use 10GbE.

5. ALGORITHM MAPPING

Mapping an algorithm onto an architecture consists of examining the requirements of the algorithms given in section 3, and finding the layout that can meet these requirements, based on the architecture characteristics outlines in section 4.

5.1 MVM mapping

MVM requires 237.2 MMACs/frame. At 800Hz frame rate, this leads to 190GFlops of required computing power. As discussed above, in terms of raw computation, this rate is in fact fairly modest and can, in theory, be achieved by even a single modern CPU. The problem is that the ~1GB worth of matrix coefficient need to all be fetched from memory within one frame, which would require a ~800GB/s memory bandwidth. As discussed above, none of the compute engines can achieve such a bandwidth. Therefore, the work needs to be split on several compute engines.

The nice feature about MVM is that the computation is very easy to parallelize. The work can be easily split between as many compute engines as required. Each compute engine processes one, several, or a portion of a WFS, producing partial DM command vectors, and all the partial DM command vectors need to be summed at the end of all the calculations by a central machine. One useful feature of the MVM approach is that the MVM can be carried out column-wise. The calculation can therefore start as soon as the first WFS slope is available, and the 500us it takes to read the WFS CCDs can be fully utilized.

5.2 Iterative algorithms mapping

The problem with mapping iterative algorithms is that 1) the computational load and/or memory footprint is somewhat too large to fit into one processor and its high speed memory; and 2) when the work is split between several processors, communications of intermediate results between the processors is required, after each iteration or even more often. Another downside is that, unlike the MVM where calculations can start as soon as the first pixels are read out of the WFS CCDs, of iterative algorithm can only accomplish very little (part of the first iteration) until all the slopes are computed. In NFIRAOS, this results in an overhead of 500us, which is the time to read all the WFS, and which is a significant fraction of the requirement.

Earlier attempt to map iterative algorithms onto GPUs have been unsuccessful [5]. On a single NVIDIA GTX 580 GPU, the fastest algorithm was found to be FDPCG, but it would take 2.6ms to complete the tomography step. And the DM fitting step took 4.5ms, for a total exceeding 7ms, a far cry from the requirement. Also, splitting the work across two GPUs did not improve these timings.

5.3 Region-based CG30

During the course of the trade-study, we have attempted to reformulate the CG30 algorithm so that it could be split onto several processors while minimizing the amount of data that needed to be exchanged between the processors. The intended target were the FPGA processors on the Kermode board. The reformulated algorithm is called region-based CG30 (RBCG30). The details of RBCG30 will be published elsewhere, but we present a summary now.

In RBCG30, the aperture planes (on which the WFS measurements are taken) and the layer planes (on which the turbulence profile is reconstructed) are separated into regions. For the sake of discussion, we choose four regions, as shown in Figure 2.

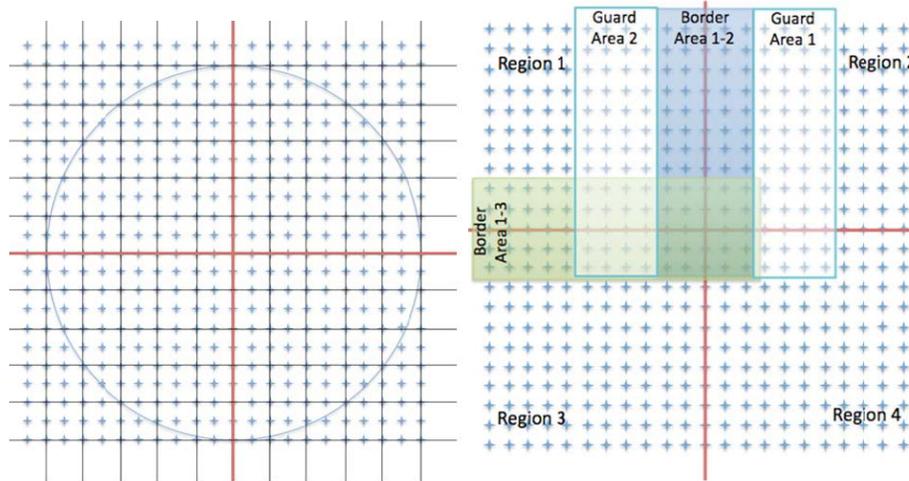


Figure 2: Layout out of the region-based CG30. Left: aperture plane. Right: layer planes, including some border and guard areas. Plus symbols correspond to phase points that are estimated during the tomography reconstruction process.

In the RBCG30, a separate processor (FPGA) handles each region. In order to carry out the calculations in a region, the process needs to have a copy of the values from the neighboring regions that lie in the border and guard areas. The guard area has a fixed size (4 points), but the width of the border area increases with the altitude of the layer, from just one point wide for the ground layer (and the aperture planes) to 8 points for the highest (15.5km) layer, seen at a 60 degree zenith angle (maximum zenith angle for NFIRAOS observing). At the end of each iteration, the processors need to exchange values from the guard and border regions, but in the NFIRAOS configuration, this is only 2772 floating point values. This is only a small fraction of the total number of values that are handled during the CG calculations, so the required communication bandwidth between the processors is minimized.

6. ARCHITECTURES CONSIDERED IN THE TRADE STUDY

Our experience with trying to implement iterative algorithms on GPUs led us to conclude that such algorithms were not suitable for PC-based architectures. It is worth noting that this assessment is not as definite as it sounds because i) we have not considered RBCG30 in on a PC-based architecture; and ii) further optimization and/or newer hardware might make the implementation more feasible. For example, when comparing the memory requirements from Table 3 with the projected amount of cache memory available in CPUs that will be available in the short term (Table 4), we see that the footprint of the CG-based reconstructor might well fit in the cache memory of a single CPU, soon, if not today, and that the computing requirements are also almost within the reach of a single CPU. However, given the limited time and resources to conduct our trade study, we have decided to only consider MVM for the PC-based architectures. We have only considered iterative algorithms for the AdvancedTCA/Kermode architecture, for which the cost of implementing a parallel MVM layout would be prohibitive (See below).

6.1 PC-Server architectures

MVM on PC-servers is accomplished by splitting the work on as many servers as required. One additional server is used for additional RTC tasks such as summing all the DM partial commands. One last server is used as spare, which can be remotely powered if needed. A high-end telecom switch handles the communication between the servers and the WFSs, DMs, etc, and ensures that telemetry data are properly recorded. The switch that we have selected in the 52-port 7150S-52 from Arista.

All the connectivity is handled by 10Gb Ethernet.

MVM on PC-Servers only (all CPUs)

The CPU-only architecture is laid out in Figure 3.

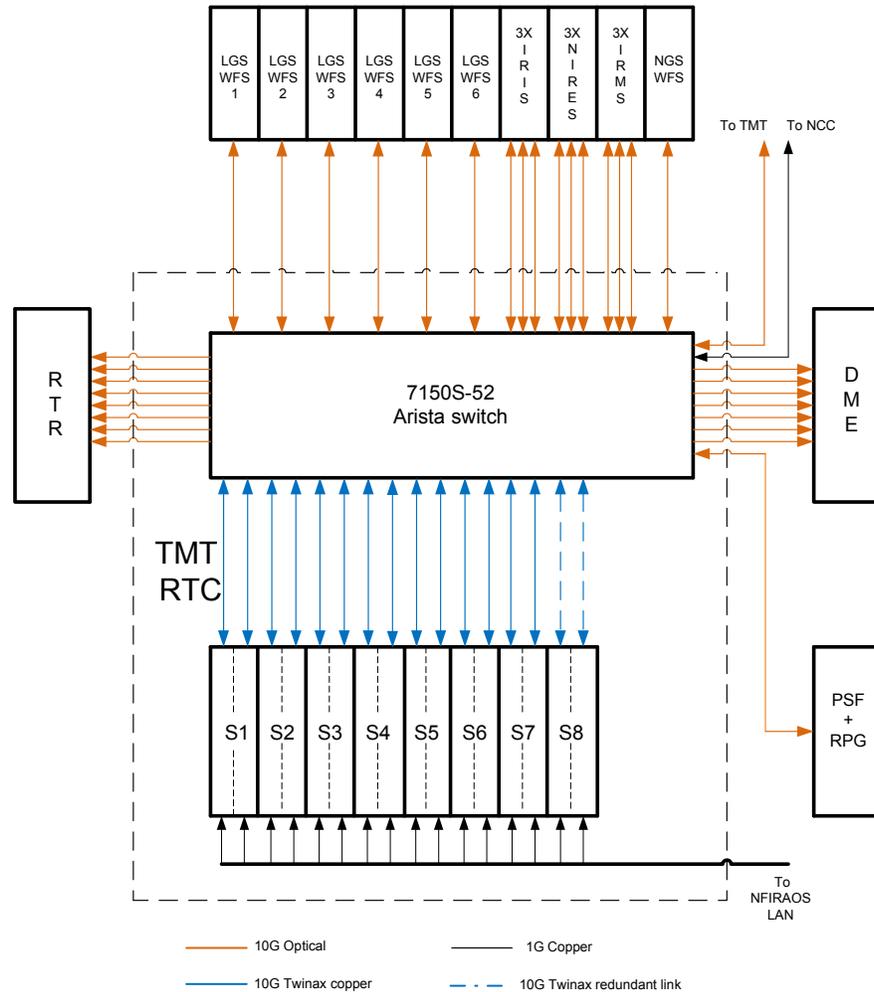


Figure 3: Block diagram used for the CPU-only MVM architecture.

There are 8 servers (S1...S8). S1-S6 handle one WFS each, S7 handles the summing of the DM commands, and S8 is a hot spare. The servers are 2U Supermicro dual node 6017TR-TF with two X9DRT-F motherboards. At the time of our trade study (2013), each motherboard contained 2 E5-2680 Intel Xeon processors with 8 cores. However, at the time of writing (Spring 2014), we would use the newer E5-2697 v2 with 12 cores. Each server therefore has 4 CPUs to handle one LGS WFS.

MVM on a PC-Server + GPU

The CPU+GPU architecture is the same as the CPU-only architecture, except that only 5 servers are required: 3 servers handle 2 LGS WFS each, one handle the summing of the DM commands, and one is a hot spare. The servers are 2U Supermicro single node 2027GR-TRFT with onr X9DRG-HTF motherboard. The motherboard has 2 E5 Intel Xeon processors + 2 K10 NVIDIA modules on x16 PCIe slots (each K10 module contains 2 GPUs). The block diagram for each server is presented in Figure 4.

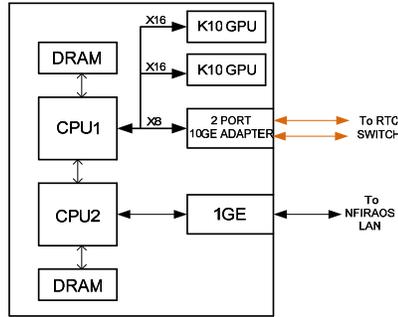


Figure 4: block diagram of a server in the CPU+GPU architecture

MVM on PC-Server + Xeon Phi

This configuration is the same as the GPU configuration above, except that the GPUs are replaced by Intel 7120P Xeon Phi.

6.2 AdvancedTCA + Kermode Board architectures

For the AdvancedTCA + Kermode Board architecture, we have considered two algorithms: MVM and RBCG. For this architecture, the connectivity to the WFS, DMs and Recorder is handled by sFPDP, using FMC sFPDP mezzanine card on the Kermode boards.

MVM on AdvancedTCA + Kermode Board

The architecture for the MVM on Kermode Board is presented in Figure 5.

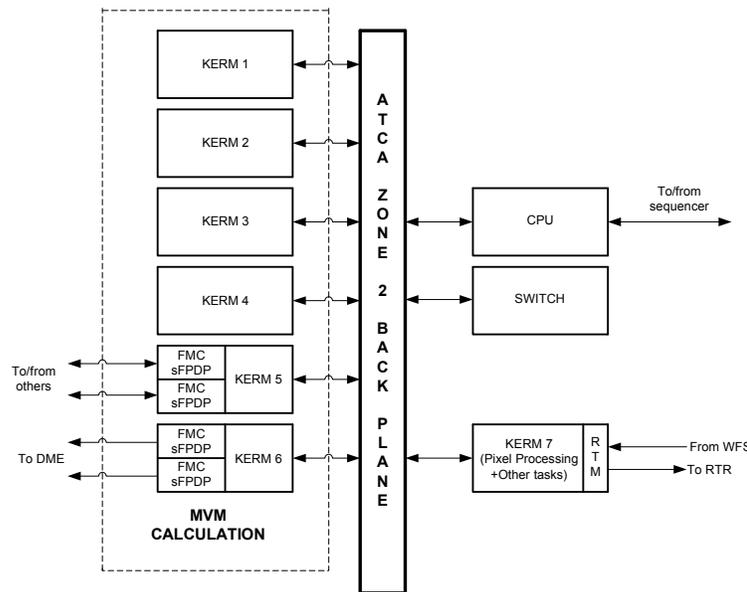


Figure 5: Block diagram for MVM on AdvancedTCA + Kermode Board architecture

Seven Kermode boards are required in order to achieve the required memory bandwidth. They are hosted in a 14 slot slot ATCA Chassis ASYS00001 from Schroff. Also hosted in this chassis is a CPU card AT8050-2 and a Switch card AT8904-2, both from Kontron. A Rear Transmission Module (RTM) need to be designed for I/O interfacing, but this is a very simple design consisting of an RTM PCB, a number of optical transceivers and a connector. The RTM module (backplane) can be easily customized to fit Kermode's Zone 3 connector. The Zone 2 back plane provides 20Gbps full mesh connectivity between boards in the chassis.

RBCG30 on AdvancedTCA + Kermode Board

RBCG30 can be implemented on one Kermode board only, using 4 of the 8 FPGAs available. Each FPGA handles one of four regions, and 2 GB/s connections are established between adjacent FPGAs, as shown in Figure 6.

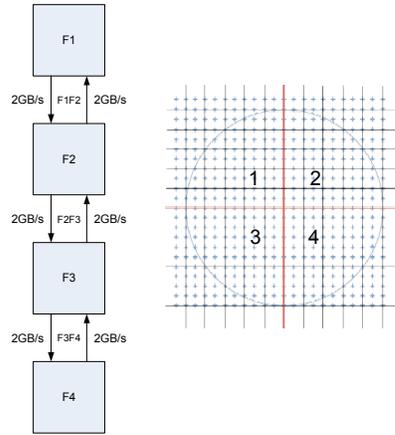


Figure 6: FPGA layout for the RBCG30 algorithm

The transfer of 2,772 floating-point numbers between each adjacent region can occur in only 11us. This includes the double loading of the communication lanes, due to the need to go through F2 and F3 to communicate between F1 and F4. The computations can be pipelined with the transfers, with points that do not require information from the guard and border areas being computed while the transfers occur. In order to keep up with the data flow, each FPGA would have to achieve 0.13TMAC/s, which is close to one half of the 0.250TMAC/s that is available in each FPGA, and therefore seems achievable. Therefore the total computation time for RCG30 is $30 \times 11 = 330 \mu s$. This time could be halved by using 8 FPGAs (and 8 regions) instead of 4. The 8 FPGAs would be split into two groups of 4, with each group in a different Kermode board.

6.3 Open VPX architecture

MVM on Open VPX

MVM could be implemented in an Open VPX architecture according to the block diagram in Figure 7.

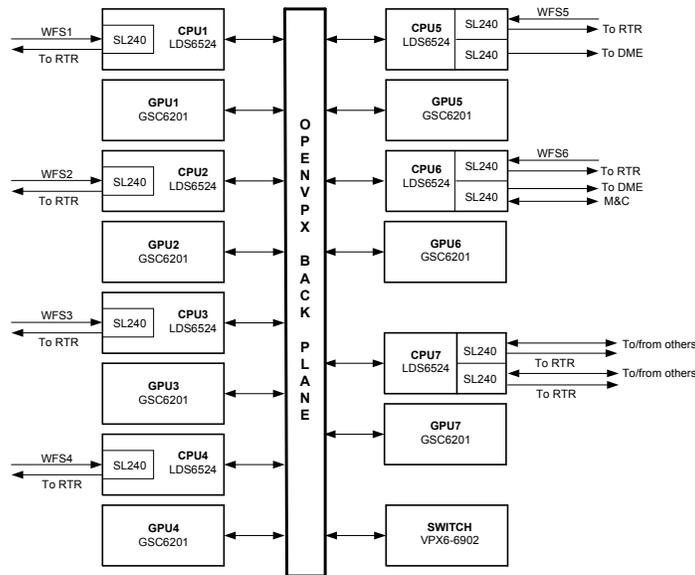


Figure 7: Functional block diagram of an MVM implementation in OpenVPX.

This computing platform utilizes military grade equipment packed in an OpenVPX chassis. This system computes MVM in the same fashion as the server based platform. There are 7 CPU cards and 7 GPU cards. Each GPU card contains two high performance MXM GPU modules. A 16 slot chassis is needed for this purpose. An OpenVPX back plane provides connectivity between the boards. An XMC style card, SL240, is selected to interface with WFSs and other instruments.

RBCG30 on OpenVPX

Our design is based on the SCFE-V6-4QSFP-OVPX card from Mercury Systems, which contains three V6SX475 FPGAs, the same as in the Kermod board, Each board also hosts two FMC bays for front panel interfacing. The design fits in a six slot chassis and its functional block diagram is presented in Figure 8.

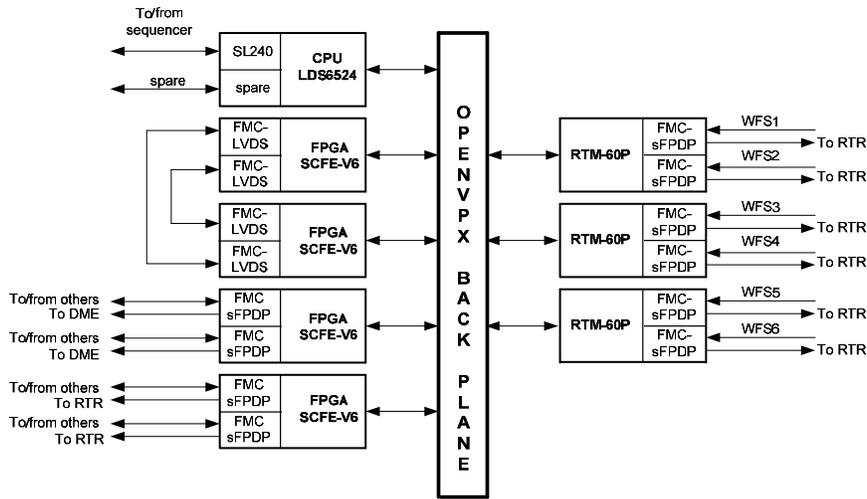


Figure 8: Functional block diagram of a RBCG30 implementation in OpenVPX.

A 4 or 6 region implementation can be achieved by using 2 (resp. 3) FPGAs from 2 cards, that are connected using their FMC LVDS interconnect cards, as shown in Figure 9.

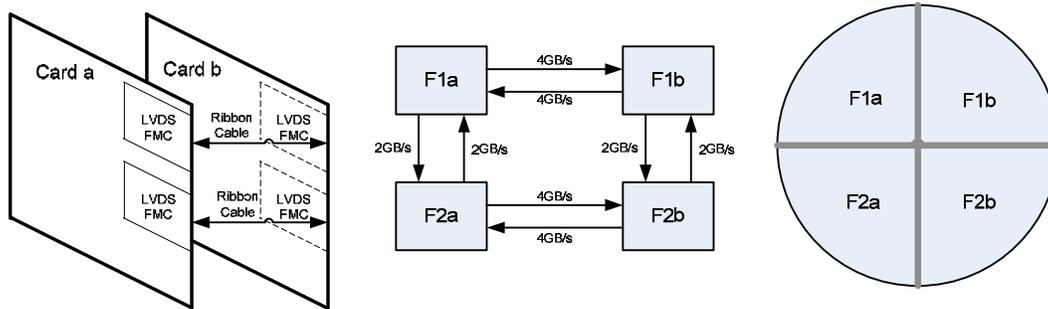


Figure 9: FPGA inter-connection layout for RBCG30 in OpenVPX

With this layout, the estimated time to complete the 30 iterations is 210us if 4 regions/FPGAs are used, and 165us if 6 FPGAs are used.

6.4 Cost, power consumption and MTBF evaluations

The cost, power consumption and MTBF of each system presented above have been evaluated based on the published values as of November 2013. Our findings are summarized in Table 5.

Table 5: Cost, power consumption and MTBF for our proposed architectures

	Cost (kilo-USD)	Power (kW)	MTBF (year)
PC-Servers CPUs only	172	5.7	1.2
PC-Servers + GPUs	101	3.7	0.77
PC-Servers + Xeon Phis	164	5.7	1.1
AdvancedTCA - MVM	532	2.5	1.25
AdvancedTCA – RGCG30	109	0.5	4.5
Open VPX - MVM	305	3.5	1.12
Open VPX – RBCG30	144	1.0	1.31

For the PC-based – MVM systems, using GPUs reduces cost and power consumption, but also MTBF, due to the lower MTBF of the GPU modules. Implementing MVM on AdvancedTCA or Open VPX increases the cost significantly, due to the higher price of the components. However, the power consumption is lower, for about the same MTBF. AdvancedTCA and Open VPX seems attractive if they can run RGCG30: because the unit count is low, the cost and power consumption is quite a bit lower, and the MTBF is significantly higher for AdvancedTCA.

7. BENCHMARKING

We have conducted extensive benchmarking for our three proposed PC-based systems, and limited benchmarking for our proposed AdvancedTCA systems. No benchmarking was performed for our proposed Open VPX system, as this solution did not look very attractive for NFIRAOS, as discussed in section 8.

7.1 PC server - CPU only benchmarking

Benchmarking of our PC Server – CPU only system is described in details in a dedicated paper [6]. Here is a summary.

Our benchmark machine replicates one motherboard of one of the servers noted S1...S8 in Figure 3. It has two E5-2697 v2 CPUs, with 12 cores per CPU. As discussed in section 4.1, this is the latest Intel CPU available as of Spring 2014. We are therefore benchmarking the work required to process one half of one LGS WFS. A similar machine, connected via 10Gb Ethernet, sends pixels at the same rate as the LGS WFS would, and receives the DM commands once the computation is performed. Using the same machine to send and receive allows for accurate timings.

The benchmarked machine receives pixels from half of one LGS WFS (one quarter of a WFS is assigned to each CPU), performs pixel processing and the part of the MVM related to these pixels and each CPU sends its DM commands back. The elapsed time (round-trip time) is calculated from the time the first pixel is sent to the time the last DM command is received.

A number of system tunings have been performed in order to achieve reliable real-time performance. These include:

- Turning hyper-threading off
- Using the Linux operating system with the real-time patch
- Isolating one core on each CPU to run all the non-RTC tasks
- Configuring the Ethernet connection to use 9000 byte jumbo packets and setting the interrupt throttle rate for the 10Gb Ethernet adapter to 0
- Assigning the interrupts originating from the 10Gb Ethernet devices to a specific core
- Using the Linux “tuned” package to set the system profile to minimize latency
- Assigning specific cores for specific tasks, including 2 cores for pixel reading, 2 cores for pixel processing and 16 cores for the MVM.
- Using UDP datagrams rather than TCP streams

The results of a 12 hour run for processing half of an LGS WFS on 2 CPUs at 800Hz are shown in the form of an histogram in Figure 10.

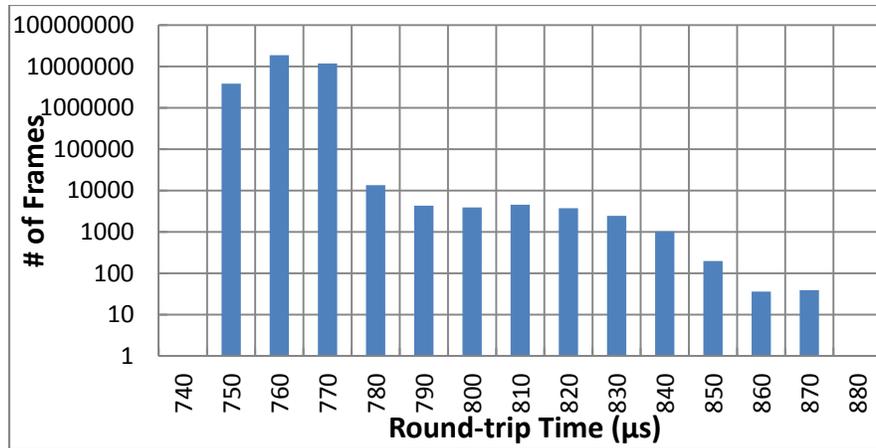


Figure 10: Histogram of end-to-end round-trip times for a 12 hours run

The average round-trip time is 766.5us, and the worst case is 877.0us, well below the requirement of 1200us. We have also conducted tests in which the control matrix was changed on the fly. We have found that swapping control matrices was slowing down the processing of the frame at which the swapping occurred, but by only ~120us, which still meets the requirements. The impact is relatively modest, thanks to the ability to pre-fetch the new coefficients from memory during the “dead time” after the new DM commands are computed and before the new frame arrives.

7.2 PC server + GPU benchmarking

Similar end-to-end benchmarking has been carried out for our PC server + GPU architecture. The results have already been published [5], and are summarized below. The test also uses two servers connected with a 10GbE link: one server sends pixel and receive DM commands and one server processes the pixel and computes the DM commands. The processing server is equipped of 2 GTX 580 GPUs, which are the consumer-grade equivalent (similar single precision computing performance and memory bandwidth) of the professional grade Tesla K10 GPUs that we carry in our design (these are built for high performance computing with high reliability and are certified for use in server environments). The benchmarking therefore simulates the processing of a full WFS, and also includes control matrix swapping every 8000 steps (sending the new control matrix from CPU to GPU is spread over a few steps, 10 columns each). The end-to-end results are shown in Figure 11. The average round-trip time is 0.88ms, the best case is 0.83ms, and the worst case is 0.97ms. This is well below the requirement of 1.2ms.

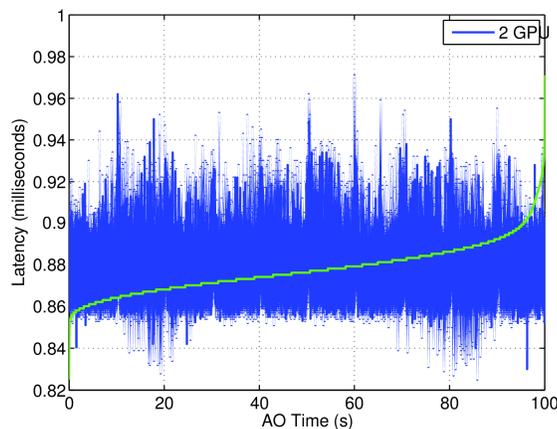


Figure 11: End-to-end round-trip times for a 9 hours run. The green curves shows the results sorted from low to high.

7.3 PC server + Xeon Phi benchmarking

Benchmarking of our PC Server + Xeon Phi system is described in details in a dedicated paper [6]. Here is a summary.

Our benchmark uses the same server platform as the CPU-only tests reported in section 7.1, but now each of the two CPU is connected to a Xeon Phi (7120P, as described in section 4.1). This set-up is equivalent to the PC Server + GPU set-up, except that Xeon Phis are used instead of GPUs. The server now handles one full LGS WFS: the CPUs compute the slopes from the pixels, and each Xeon Phi computes the MVM for half of the slopes. In order to keep the code simple, we used the OpenMP 4.0 programming framework, with which computations can be assigned to the Xeon Phi with simple offload directives. Our first attempt was to offload the MVM calculation in one single call. This requires that all the slopes are computed, and can only start after the whole CCD is read, 500us after the first pixel arrives. The time between when the offload command is issued and when the CPU receives the last DM command from the Xeon Phi is shown in Figure 12.

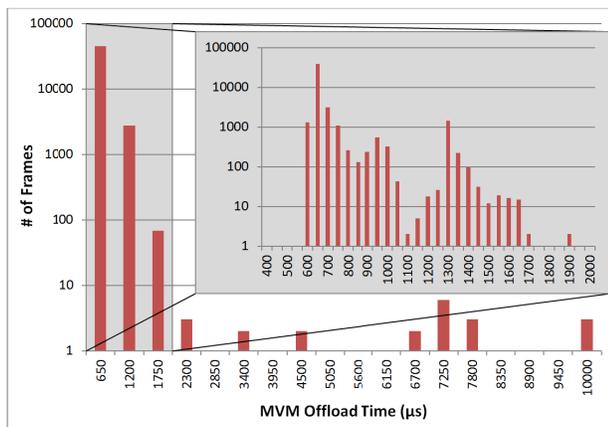


Figure 12: MVM execution time using a single offload call to the Xeon Phi to process half of one LGS WFS. The inset shows the results for times less than 2ms.

The results show an unacceptable level of latency, with some measurements of up to 10ms. Also these results do not account for the extra 500us required to compute all the slopes. We have tried to use two offload commands, which allows starting when only half of the pixels are received, but the latency results got even worse. We believe that the average latency could be reduced by using lower level communication routines (no OpenMP). However, we do not believe that this could solve the problem of the “slow frames”, which is probably due to the fact that the Xeon Phi support software does not have at the moment a real-time kernel.

7.4 Control matrix computation benchmarking

The MVM algorithm relies on another machine computing the control matrix, and updating it every ~10s. This machine is not part of the NFIRAOS RTC, and is located in the observatory control room. The FDPCG iterative algorithm is used to compute the control matrix [5]. This process can be parallelized, and has been benchmarked on GPUs. The results show that for updating the control matrix, where the previous control matrix can be used as warm start, only 10 FDPCG iterations are required, and they can be accomplished in ~9 seconds on 8 GPUs. At the beginning of a new observation when no warm start is possible, 100 iterations are required. This will take 90 seconds, which is well under the 5 minutes allowed to be ready for a new observation. Once the control matrix is calculated, it takes about 1s to download it over a 10GbE connection.

7.5 RBCG30 on Kermode Board benchmarking

We have benchmarked critical modules of the FPGA design for the RBCG30 by implementing within the Xilinx design tools. Unlike CPU-based compute engines, FPGAs are very deterministic, and therefore their performance can be simulated very accurately. The layout of the FPGAs handling region 2 and region 3 is shown in Figure 13. The FPGAs handling regions 1 and 4 are similar.

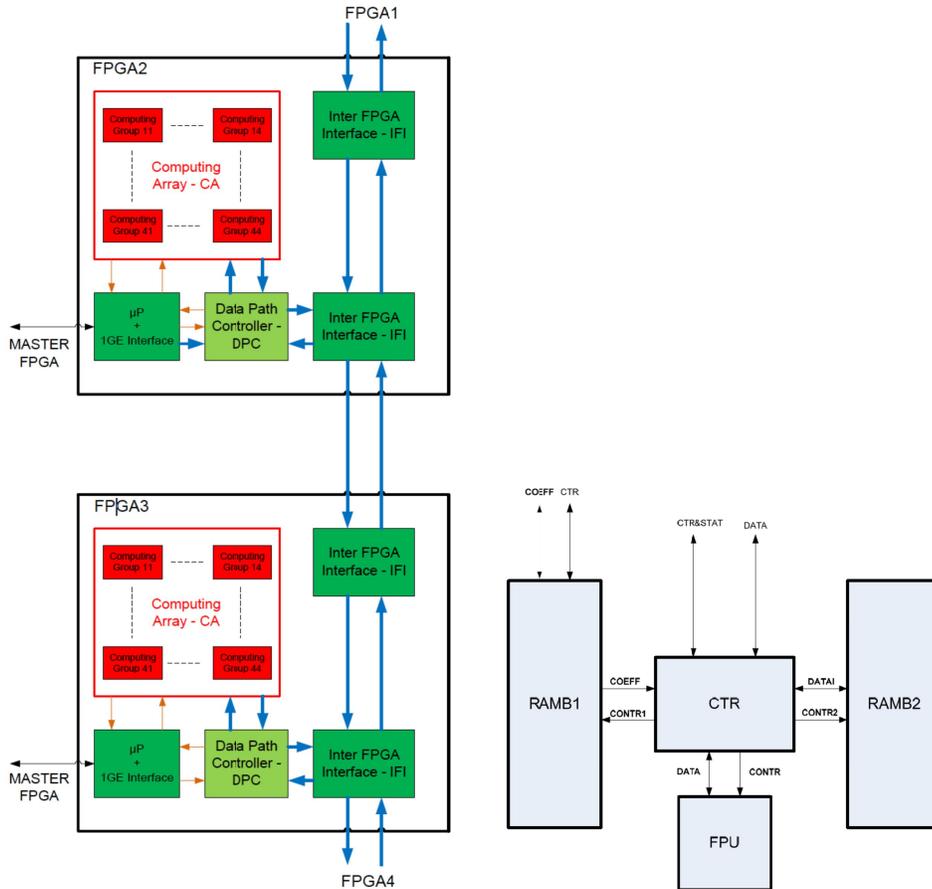


Figure 13: High Level FPGA RGCG30 design (left), and design of a compute element (right)

The most critical part is the computing array. It is made up of 16 computing groups, and each computing group has 256 computing elements. The layout of a computing element is shown in Figure 13 (right). The FPU block contains Floating Point Single Precision Adder and Multiplier.

A full computing array was implemented in the Xilinx development tool. It was found that it could run at 450MHz, therefore achieving 0.1152TMAC/s. This would lead to 12us to compute one CG iteration, or 360us for the 30 iterations. This well meets our requirements.

We have not pursued further benchmarking for this architecture.

8. TRADE STUDY SCORING

8.1 AdvancedTCA architecture with Kermode Board

Based on our study, we have decided to reject this architecture.

Our main concerns with this solution were:

- The hardware cost, which is higher than the other solutions.
- Time to repair: even though the MTBF of the hardware is highest of all the architectures, the time to repair is also the highest, because there will be no on-line live spares. The lost time due to failure was estimated at 0.04%, well below our 0.1% budget, but it is an order of magnitude greater than the other architectures.
- Programming time and programming skills: as demonstrated by our attempts at benchmarking, programming FPGAs is labor intensive and requires specialized programming skills. Also, the iterative algorithms that would be implemented on the Kermode board are more complex than the straight MVM.

A few strong points could be identified however. These were:

- Low energy consumption, the only architecture that actually meets our power budget.
- Higher flexibility, since in theory, a variety of algorithms could be implemented on the platform.
- Savings in the RPG, since iterative algorithms avoid the computationally expensive task to compute the control matrix.

However, we have found that these advantages do not outweigh the drawbacks.

8.2 Open VPX architecture

We have also decided to eliminate the Open VPX architecture. Disadvantages include the fact that there are only few vendors on the market, that the hardware is relatively expensive, and that the hardware tends to be an older generation. These disadvantages appear to outweigh the fact that VPX is rugged and reliable and that it is heavily standard based.

8.3 PC-server based architectures

In the end, we only retained the PC-based solution.

Cost, power consumption and MTBF have already been presented in section 6.4. The PC+GPUs is the least expensive option for the hardware cost. However, the software development costs are estimated to be roughly the same for the three candidate architectures, and are much higher than the hardware cost (the total RTC cost is estimated to \$6M). Therefore the difference in hardware cost is rather insignificant.

The three candidate architectures meet our space requirement, with the PC+GPUs solution being the most compact (11U instead of 17U for the other ones). The PC+GPU solution has also the least power consumption, but still does not meet our power budget. As discussed previously, some margin exists within NFIRAOS to increase the power allocation for the RTC. The main concern here is in fact the potential heat leakage through the electronics enclosure, which only has a 10 kW cooling capacity.

The MTBF for each candidate architecture has been turned into expected lost time. Because all the architectures have on-line spares, the lost time is very low, much lower than our requirement. The lowest lost time was achieved by the CPU-only solution at 0.004%, but the other two architectures were only marginally higher at 0.005%. So the lower MTBF of the GPUs does not translate in significant increased loss time under our operational assumptions.

Based on the MTBF, cost of replacement parts and energy consumption, we have attempted to estimate a cost of ownership through the full 15 year lifetime. We have found that the PC+GPU solution had a lowest cost (~\$300k), followed by the PC+Xeon Phi (~\$400k), the most expensive being the PC+CPU only solution (~\$500k). This is because even though GPUs are less reliable, they are cheaper to replace and win because of their lower power consumption. However, although we have not exactly quantified it, we estimate that the order is reversed for the cost of software upgrade (which will be required to keep up with hardware obsolescence), with the PC+GPU being the most expensive, because of the specialized programming language and the more heterogeneous layout.

9. CONCLUSION

Our study has shown that the commercial PC-servers with 10Gb Ethernet connectivity and running an MVM algorithm have now reached a sufficient level of maturity to meet the NFIRAOS RTC requirements, and that we no longer need to resort to more specialized platforms such as VPX or AdvancedTCA. This conclusion is grounded in a detailed analysis and extensive end-to-end benchmarking, which also includes the benchmarking of the calculations to regularly update the matrix used in MVM by the RPG.

We are very confident that both the PC+CPU-only and the PC+GPU architectures we are proposing can work, and, based on our scoring table, it is difficult to identify a clear winner. For the PC+Xeon Phi architecture, while we find that it could work in theory, our benchmarking results currently show that the timing requirements are not met, with large jitter occurring. Between PC+CPU-only and PC+GPU, it is difficult to pick a clear winner.

Table 6 presents the criteria where the three proposed architectures differ the most.

Table 6: Key comparison criteria between the three server-based architectures

	All CPU	CPU+GPU	CPU + Phi
Pluses	<ul style="list-style-type: none"> •Homogeneity •Ease of programming •Life cycle risk •Real-time kernel 	<ul style="list-style-type: none"> •Power •Cost •Computation margin 	<ul style="list-style-type: none"> •Computation margin (in theory)
Minuses	<ul style="list-style-type: none"> •Computation margin •Cost 	<ul style="list-style-type: none"> •Heterogeneity •Reliability •Ties with consumer market 	<ul style="list-style-type: none"> •Heterogeneity •Cost •New product •Sole source •Benchmarking results

The main appeal of the all-CPU architecture is that it is conceptually the simplest and the most homogeneous. It is the easiest to program since it does not require the specialized programming skills that the other accelerator-based platforms require. Due to limited shell life, it will not be possible to stock spares for the entire life span of NFIRAOS for any of the proposed architectures. However, CPU servers are such mainstream systems that they are the most likely to continue to have compatible newer versions available. Continuous support of real-time operating system kernels, which is important to ensure low jitter, is also extremely likely. The downside of the all-CPU architecture is that it offers the least computational margin and comes with a higher cost.

The CPU+GPU architecture is more compact, provides more computational margin at a lower cost and for a lower power. However, the reliance on third-party GPU accelerator cards increases the heterogeneity of the system, and therefore its complexity (programming, connectivity, etc). Even though all the three architectures meet the TMT requirement for lost observing time due to failure, the CPU+GPU architecture has also a lower MTBF, and therefore will require more maintenance at the observatory. Also, the GPU technology has strong ties with consumer markets, which makes its evolution somewhat less predictable.

The CPU+Xeon Phi architecture provides, in theory, even more computational margin than the CPU+GPU architecture, simply because more servers are required, and therefore more CPUs are available for additional calculations. However, benchmarking has shown that latency was an issue that might not be resolvable. This architecture remains more heterogeneous than the all-CPU architecture, because it needs to run its own operating system. This solution is also the most expensive, and the Xeon Phi co-processor is a relatively new product available from one company only. Finally, as discussed above, this architecture did not pass our benchmarking tests.

Overall, we feel that the all-CPU architecture is the most attractive solution at this time. Lesser complexity resulting from a more homogeneous platform reduces the cost and schedule risks when building the RTC, and facilitates maintenance and trouble-shooting during operation. The risk of obsolescence is also better mitigated due to the more likely availability of compatible spares during the lifetime of NFIRAOS. Our opinion is that these risk reductions, although difficult to quantify, are worth the slight increase in cost and power consumption.

REFERENCES

- [1] Sanders, G. H., "Thirty Meter telescope project update," In Ground-based and Airborne Telescopes V Proc. of SPIE, Vol. 9145, (2014).
- [2] Herriot, G., et al., "NFIRAOS: first facility AO system for the Thirty Meter telescope," In Adaptive Optics Systems IV Proc. of SPIE Vol. 9148, (2014).
- [3] S. Browne et al., "A Real-Time Controller Architecture for the Multi-Conjugate Adaptive Optics System on the Thirty Meter Telescope", 1st AO4ELT conference - Adaptive Optics for Extremely Large Telescopes, oral presentation only (2010)
- [4] Hovey, G. et al., "An FPGA Based Computing Platform for Adaptive Optics Control", 1st AO4ELT conference - Adaptive Optics for Extremely Large Telescopes 07006 (2010)
- [5] Wang L., Design and Testing of GPU based RTC for TMT NFIRAOS, In: Proceedings of the Third AO4ELT Conference. Simone Esposito and Luca Fini, eds. Firenze, Italy, 26-31 May 2013.
- [6] Smith, M., et al., "Benchmarking hardware architecture candidates for the NFIRAOS real time controller," In Adaptive Optics Systems IV Proc. of SPIE Vol. 9148, (2014).