METIS AO RTC Prototype

Martin Kulas, Horst Steuer, Thomas Bertram, and Florian Briegel

Max Planck Institute for Astronomy, Heidelberg, Germany

ABSTRACT

Adaptive Optics (AO) will enable the Extremely Large Telescope (ELT) to perform observations close to the diffraction limit. As part of the Mid-infrared ELT Imager and Spectrograph (METIS) team, we develop a real-time computer (RTC) implementing the control loops for Single Conjugate AO (SCAO). The wavefront control loop has the tightest timing constraints: It receives the images from the wavefront sensor at up to 1 kHz, computes commands and sends them to the adaptive mirrors of the ELT. The sheer scale of the ELT in terms of the number of actuators driven by the SCAO system requires unprecedented demand of computational performance to run the wavefront control loop in hard real-time.

Here we present results from the preliminary design phase (PDR) and ongoing work towards a prototype for the real-time control system that is divided into a hard real-time core and a soft real-time cluster. We focus on the hard real-time computer (HRTC) and evaluate the performance of a prototype using Graphics Processing Units (GPU) as hardware accelerators. Using a matrix vector multiplication (MVM) based reconstructor, we show that the main timing requirement is fulfilled.

Keywords: AO, RTC, GPU, MUDPI, METIS, ELT

1. AO RTC IN METIS SCAO

METIS is the Mid-infrared ELT Imager and Spectrograph for the ELT.[1] Its single conjugate AO system (SCAO) will allow observations close to the diffraction limit. The block diagram in figure 1 depicts METIS SCAO and its environment.

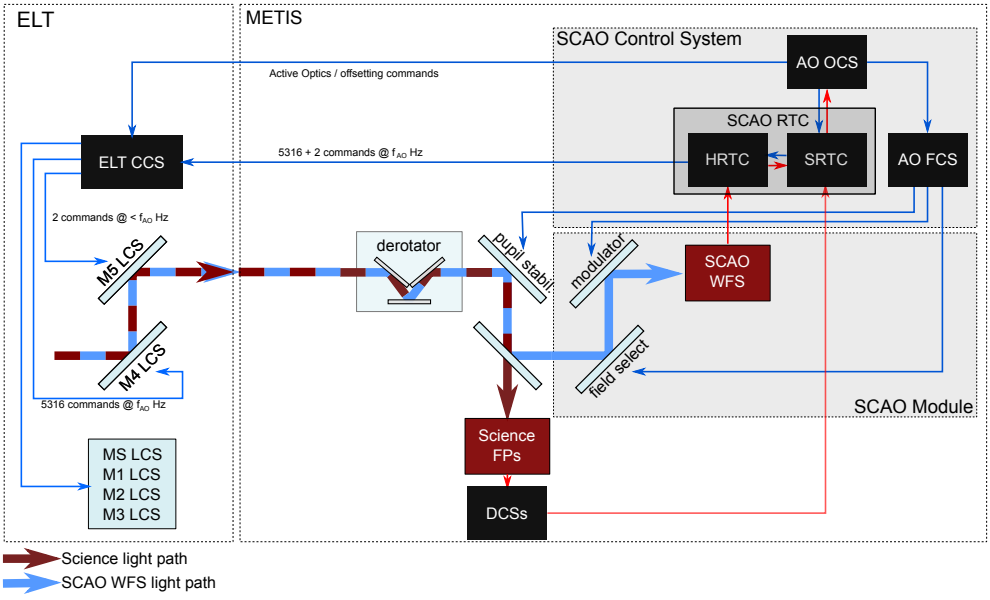


Figure 1. Block diagram of METIS SCAO and its environment.

The heart of the METIS SCAO is the real-time computer (RTC). It is split into the two following components:

- HRTC: Hard Real-Time Core for the main AO loop, namely wavefront control (WFC) loop. It receives frames from the wavefront sensor (WFS), computes corrective optics commands and sends them to the ELT Central Control System (CCS). Additionally, it sends telemetry data (e.g. calibrated WFS images) to the SRTC for further processing.
- SRTC: Soft Real-Time Cluster supervises the HRTC and improves its AO performance. It runs control loops using HRTC telemetry data and communicates with the AO Observation Coordination System (AO OCS) sending telemetry data and receiving configuration parameters.

The table below gives an overview of the key METIS SCAO properties that are relevant for the RTC:

WFS type	Pyramid WFS with 4 sides
WFS pupil size	93×93
Subapertures	6785 (i.e. 13,570 gradients)
WFS detector readout speed	1 kHz
Number of corrective optics actuators	$5316 + 2 = 5318$
Max. RTC computation time	$909\,\mu$s

According to ESO,[2] RTC computation time is defined as the "time elapsed between the first WFS data received at the RTC and the last command data transmitted by the RTC".

In this paper we present our preliminary results in designing and developing a HRTC which can deliver the necessary performance given the constraints regarding problem size and timing described in the table above. In the following section, we describe our current design for the HRTC software. In Sec. 4 we describe the current setup for our performance evaluation including an analysis of the necessary data throughput. Preliminary results of the performance evaluation are presented in Sec. 5. Sec 6 gives an overview of our planned activities towards completing the RTC prototype and finally a conclusion of our findings is given in the last section.

2. RELATED WORK

Adaptive optics became an important technology in astronomy since the late 1980's when the first AO system for astronomical infrared images COME-ON (CGE Observatoire de Meudon ESO ONERA) was tested at the 1.52m OHP telescope in France. It later was deployed at the ESO 3.6m telescope at La Silla, Chile.[3] COME-ON had a deformable mirror with 19 actuators and its wavefront computer was able to process wavefront images at 200Hz using a least squares approach. This wavefront computer was equipped with a 32bit processor Motorola 68020,[4] which can deliver 5.36 million instructions per second (MIPS) at 33MHz.[5]

COME-ON was developed as a prototype-system for the European Very Large Telescope (VLT) and could be demonstrate, "that adaptive optics works reliably and repeatingly for the correction of astronomical infrared images".[6] The first adaptive optics system on the VLT is the Nasmyth Adaptive Optics System (NAOS) which had its first light in 2001.[7] The system was designed to process up to 144 active subapertures in the WFS images at 444Hz and drive 185 actuators.[8] Compared to the prototype COME-ON this meant a significantly increased demand on the computational performance. Here, the computation could not be handled by a single processor but the wavefront correction had to be distributed to four specifically designed boards, each holding two computation modules with three Texas Instruments DSP TMS320CS40 microprocessors each.[9] Each of these microprocessors can deliver up to 30 MIPS or 60 MFLOPS.[10]

While the ELT is currently under construction, adaptive optics systems for all of the first light instruments are being developed. Besides METIS, the first light instruments are the 'High Angular Resolution Monolithic Optical and Near-infrared Integral field spectograph' (HARMONI), 'Multi-AO Imaging Camera for Deep Observations' (MICADO), and the 'Multi-conjugate Adaptive Optics Relay for the ELT (MAORY)'. Due to the differing requirements regarding e.g., the latencies and the number of laser guide stars, different RTC architectures are being evaluated for these instruments.

Jenkins et al.[11,12] propose a CPU based approach and were able to show that for a ELT sized SCAO problem, they were able to achieve a latency of less than 600 μs using a Xeon Phi multi-core processor (<700 for a similar

AMD Epyc) system. They also proposed a multi node RTC architecture for the multi-conjugate optics(MCAO) and laser tomographic (LTAO) case as specified by MAORY and HARMONI, where each node of the eight nodes is based on a multi-core CPU system.

A different architecture is proposed by Clénet et al.[13] for the specification of MICADO: instead of distributing computational work to the cores of a CPU-only system, it is offloaded to GPU accelerator cards. During prototyping a Nvidia DGX-1 with eight Nvidia P100 GPUs was used[14] which can deliver up to 170 TFLOPS in half precision.[15] A similar GPU-based architecture was assessed for the MICADO-MAORY SCAO RTC.[16]

3. HRTC PROTOTYPE DESIGN

The HRTC has to run the main AO loop in hard-real time. To stay within this challenging constraint the use of hardware accelerators is necessary. For the prototyping activity, we chose a flexible software design which allows to test different hardware accelerator types with relatively low effort.

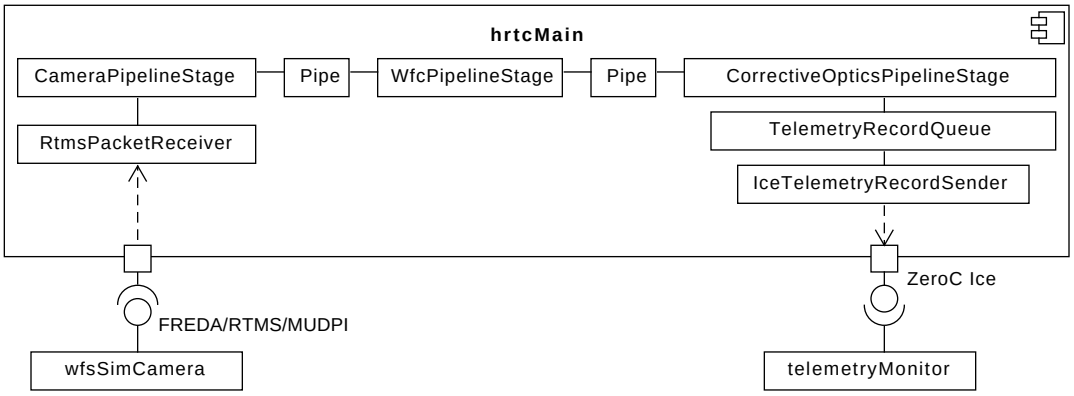


Figure 2. HRTC pipeline structure: `CameraPipelineStage` reconstructs the WFS camera frame from the received RTMS packages, then `WfcPipelineStage` computes the command vector and provides telemetry to `CorrectiveOpticsPipelineStage` which finally send the telemetry data to a telemetryMonitor.

Figure 2 depicts the building blocks of our HRTC program called `hrtcMain`. The program is designed as a pipeline with several functional stages. Currently, the only input to `hrtcMain` are WFS camera frames sent over Ethernet via RTMS protocol (Real-Time MUDPI Streaming). At a later stage in the development process, additional inputs like control parameters provided by the SRTC will be implemented. Currently the only output sent by `hrtcMain` are the telemetry records as described in Sec. 4.1.2. Albeit DM command vectors are being produced by the program, in this stage of the development process we chose only to send telemetry data, as they incur a larger traffic on the network and they should be enough to demonstrate the feasibility of the approach.

The building blocks of `hrtcMain` itself are described in the following table:

`RtmsPacketReceiver`	Receives data packages from the WFS camera over Ethernet via RTMS protocol (Real-Time MUDPI Streaming).
`CameraPipelineStage`	Reconstructs the WFS camera frame from the received data packages and provides them via a pipe to the `WfcPipelineStage`.
`WfcPipelineStage`	This stage computes the corrective optics commands and is computationally expensive. It calibrates the WFS frame by subtracting pixel-wise offsets and background and correcting for gain. It then computes the WFS signal from the four pupils of the pyramid sensor before calculating the command vector by a matrix vector multiplication.
`CorrectiveOpticsPipelineStage`	Sends out the corrective optics commands and inserts the telemetry records into the `TelemetryRecordQueue`.
`IceTelemetryRecordSender`	Instances of this class transmit the telemetry records via ZeroC Ice[17] to a telemetry monitor.

The computationally most demanding stage in the pipeline is the `WfcPipelineStage`. As such it is designed as an abstract class so that by inheriting from this base class, programmers are able to implement the WFC computation with different hardware accelerators. Currently, two specialized classes exists: one class supports GPUs and one class contains a reference implementation in plain C++.

4. HRTC PERFORMANCE EVALUATION SETUP

One design goal of the METIS HRTC is supporting hardware accelerators like GPUs and NUMA computer systems. For the results presented in this paper, GPUs are used as hardware accelerators. We use a single x86-64 server with an AMD Ryzen Threadripper 2950X CPU and NVIDIA GPUs as hardware platform and implement the main loop in a single operating system process. For the implementation of the main loop, we used C++11 and NVIDIA CUDA 10.0.

Figure 3 depicts the experimental deployment. The pixel stream was generated by an operating system process simulating a WFS camera. Both the HRTC main process and the WFS simulation camera process were located in the same workstation. Both processes communicated via the IPv4 socket interface using RTMS packets. The telemetry stream was sent out to a telemetry sink over a 10GbE.

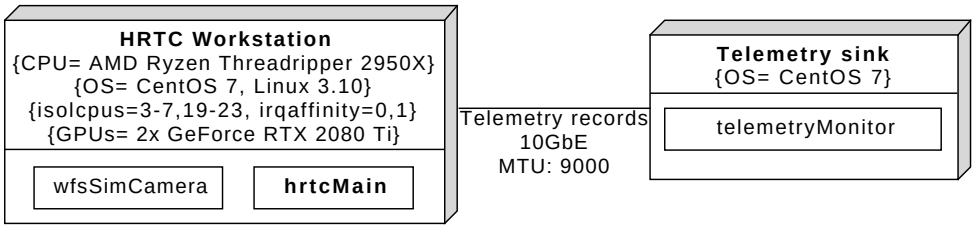


Figure 3. Experiment deployment: the AO main loop and the simulated WFS camera as data provider reside in the same physical system, while the telemetry monitor is deployed on a different computer.

4.1 Memory Throughput Analysis

In the following, an analysis of the theoretical memory throughput is presented.

4.1.1 WFS Pixel Stream

The wavefront sensor generates frames with a rate of up to 1 kHz. The detector region of interest is 192 ×186 pixels = 35,712 pixels with 4 Byte per pixel. These frames are fragmented into RTMS packets according to the ESO FREDA (inFraRED cAmera) packet description:[18]

- 1 RTMS leader packet: 54 Byte
- 28 RTMS payload packets:
  - 27 RTMS payload packets: 5170 Byte
  - 1 RTMS payload packet for the remaining pixels: 4658 Byte
- 1 RTMS trailer packet: 42 Byte

The total frame size on the UDP interface is 144,344 Byte. Thus, the wavefront sensor generates a pixel stream with a throughput of approx. 145 MByte/s which has to be consumed by the HRTC.

4.1.2 Telemetry Stream

The AO loop generates telemetry that can be used to estimate system parameters like e.g. seeing and to optimize AO loop parameters like e.g. gains. In each AO loop cycle, the HRTC generates a telemetry record that is composed of the following elements:

- Frame counter: 4 Byte
- Corrective optics command vector: 5318×4 Byte = 21,272 Byte
- WFS signal vector: $13{,}570 \times 4$ Byte = 54,280 Byte
- Subaperture intensities: 6785×4 Byte= 27,140 Byte
- Calibrated frame: $192 \times 168 \times 4$ Byte + 2×4 Byte= 129,032 Byte
- Timestamps: 7×8 Byte = 56 Byte

The total telemetry record size is 231,784 Byte which theoretically produces a data stream of approx. 231 MByte/s on the Ethernet excluding the network protocols' overhead when sent to the `telemetryMonitor`.

4.1.3 Data Transfer between Host and Hardware Accelerator

Hardware accelerators are commercial off-the-shelf (COTS) GPUs, namely two instances of GeForce RTX 2080 Ti. Both GPUs were inserted in PCIe 3.0 slots with 16 lanes which theoretically enables approx. 16 GByte/s throughput in both directions between the GPUs and the CPU. No NVlink was used to connect the GPUs directly.

The memory throughput provided by PCIe (approx. 16 GByte/s) significantly exceeds the memory throughput demand by both the WFS pixel stream (approx. 145 MByte/s) and the telemetry stream (approx. 231 MByte/s).

We confirmed the achievable memory throughput of the PCIe bus by measuring it with the NVIDIA utility `bandwidthTest`. As shown in the table below, the achieved memory throughputs fall short of the specified throughput of 16 GByte/s but suffices for the 376 MByte/s needed in this stage of the prototype.

source to destination	**Memory throughput [GByte/s]**
Host to device	12.8
Device to host	13.2
Device to device	519.2

4.1.4 MVM Memory Throughput

The METIS SCAO PDR baseline reconstructor is based on MVM. The reconstruction matrix size is 5318 × 13,570 × 4 Byte = 288,661,040 Byte. Thus, the required memory throughput for loading the reconstruction matrix in each loop cycle is approx. 289 MByte / 0.909 μs = 318 GByte/s. Note that this data is loaded only from random access memory (RAM) or caches to the processing units in each loop cycle and does not need to traverse any network connection with this frequency. While the reconstruction matrix in the final version of the prototype would need to be updated and sent over the network every few seconds, in the current prototype this is not implemented yet, and the matrix is loaded into the RAM at the beginning of the experiment.

Compared with loading the reconstruction matrix, the memory throughput demand by the WFS pixel stream and the telemetry stream is negligible. The same applies to other AO loop tasks like pixel preprocessing: they require a much smaller memory throughput than the wavefront reconstruction task. Overall, the wavefront reconstruction MVM dominates the memory throughput demand.

The used GPU GeForce RTX 2080 TI provides a theoretical memory bandwidth of 616 GByte/s.[19] Our measured memory throughput of approx. 519 GByte in section 4.1.3 satisfies the memory throughput demand.

4.2 Timings

The METIS HRTC generates a telemetry record in each AO loop cycle. Part of the telemetry record are the timestamps depicted in figure 4 and defined in the following table:

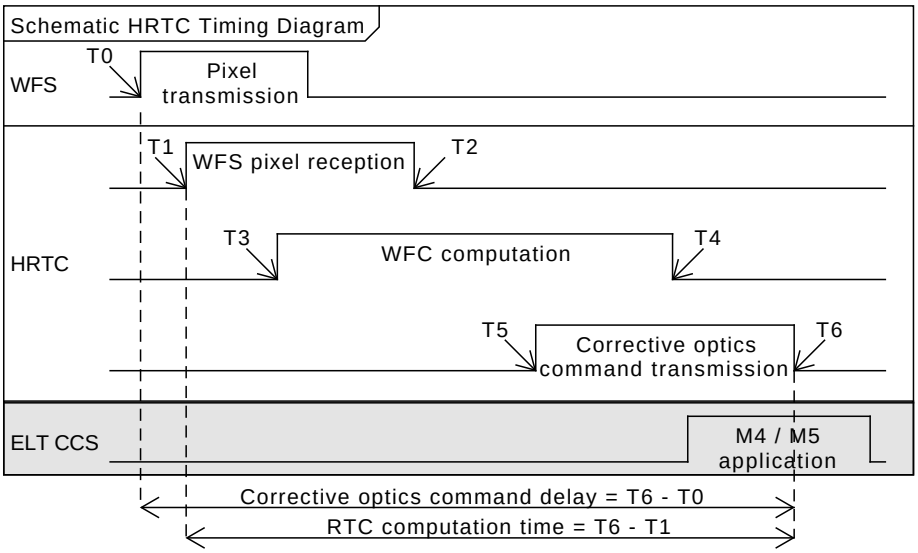


Figure 4. Schematic METIS HRTC timing diagram.

Timestamp name	Definition
T0	This timestamp contains the closest time to start the transmission of the first FREDA pixels over the network. The HRTC extracts this timestamp from the FREDA leader packet.
T1	Pixel reception start timestamp. Generated when HRTC receives the first FREADA payload packet.
T2	Pixel reception end timestamp. Generated when HRTC receives the FREDA trailing packet.
T3	WFC pipeline stage start timestamp.
T4	WFC pipeline stage end timestamp.
T5	Corrective optics pipeline stage start timestamp.
T6	Corrective optics pipeline stage end timestamp.

The delay definitions below are derived from the timestamps and are used in the performance valuation experiments to analyze the impact of the different stages on the overall performance:

Delay name	Definition
First WFS pixel transfer delay	T1 - T0
Camera pipeline stage delay	T2 - T1
WFC pipeline stage delay	T4 - T3
Corrective optics pipeline stage delay	T6 - T5
RTC computation time	T6 - T1
Corrective optics command delay	T6 - T0

Note that Fig. 4 shows overlapping time segments for most of the delays. This is motivated by the fact that the underlying algorithms theoretically can start working on partial data before the previous stage has finished. This can be used to decrease the overall delay from pixel data reception to command vector transmission, but in our case this has not been implemented. In the current version of our prototype, the different time segment are sequential. This means that the 'first WFS pixel transfer delay (T1-T0)' in our case corresponds to the pixel transmission delay.

5. HRTC PERFORMANCE EVALUATION

We evaluated the performance of the HRTC prototype using the setup described in the previous section. In our experiment the AO loop performed 300,000 cycles. For METIS AO, the RTC computation time has to be less than 909 μs as mentioned in section 1. Thus, in this experiment we measured the total HRTC computation time per frame. Additionally we were interested in a break down of the computation time on the different stages for further optimizations. Finally, we tested the data throughput on the network to ensure that our Ethernet setup can cope with the data rates and that the computation is successfully conducted on all frames and none are dropped.

5.1 Completeness of Frame Transmission and Processing

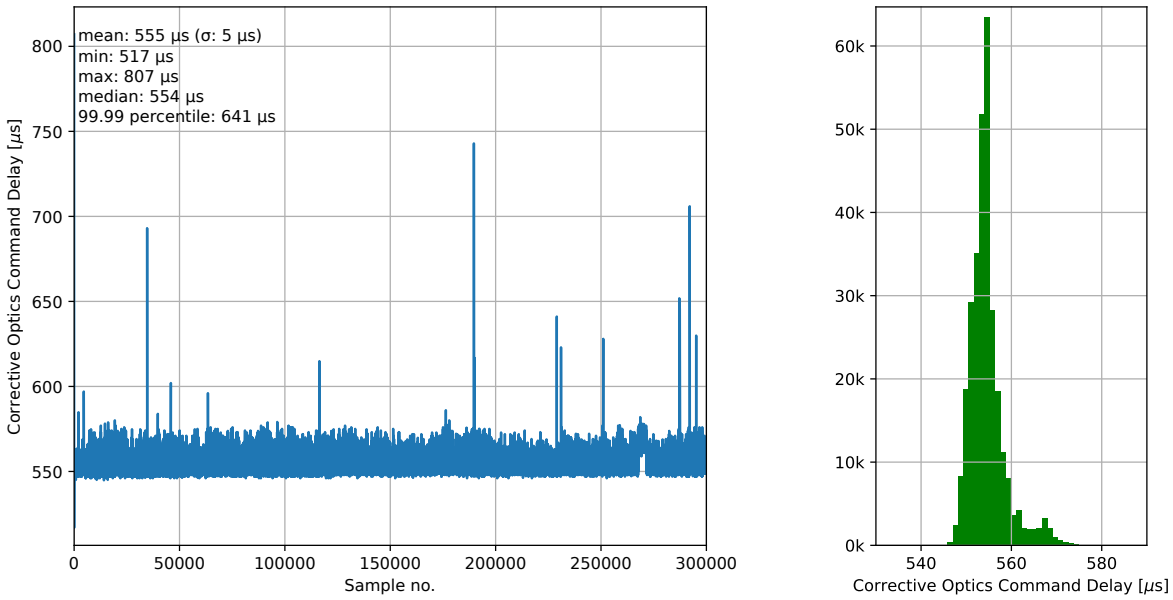
Due to the hard real-time constraints under which the HRTC has to perform, there is not enough time to repeat sending of packets lost during transmission over Ethernet. Additionally the HRTC pipeline is designed such that if a stage stalls for any reason, data frames are skipped instead of being queued for later processing. Ideally, all frames should be processed in time without dropping any.

In our experiment, we tested if any frames were dropped. All WFS frames were received successfully by the `CameraPipelineStage`. Also, no telemetry record was lost during transmission but all records were transferred successfully to the `telemetryMonitor` indicating that all stages of the pipeline stayed within their timing constraints and no packets were lost during transmission.

5.2 Telemetry Stream

We expected the telemetry stream to incur a data throughput of at least 231 MByte/s (see Sec. 4.1.2) based on the payload alone. We measured a data throughput of approx. 246 MByte/s on the Ethernet interface. Thus, roughly 15 MByte/s was protocol overhead of ZeroC Ice.

5.3 Timings



(a) (b)

Figure 5. Corrective optics command delay time series (a) and histogram (b). Histogram is cropped as the outliers are not visible on this scale.

In the experiment, we measured a time series of corrective optics command delay per frame. Figure 5 shows the time series and the corresponding histogram.

The mean value of the corrective optics command delay is 555 μs and the 99.99th percentile is 641 μs, i.e. 30 corrective optics commands took longer than 641 μs. The maximum value is 807 μs. Since the RTC computation time is less than the corrective optics command delay, the maximum RTC computation time of 909 μs was fulfilled in this experiment with 807 μs.

5.4 Mean Shares of Corrective Optics Command Delay

The HRTC telemetry record contains seven timestamps. They provide a way to measure the delays that sum up to the corrective optics command delay. Figure 6 depicts the mean shares of corrective optics command delay. Most of the delays correspond to the delays described in Sec. 4.2. The two unnamed delays T3-T2 and T5-T4 correspond to the delay introduced by the pipes between stages (c.f. Fig. 2) and are negligible with a percentage of 0.6%.

As expected, the WFC computation is the largest part of the corrective optics command delay (about 70 %). The wait for reception of all WFS pixels is approx. 120 μs and takes a share of about 22 %. The WFS frame transmission occurs without Ethernet communication but with IPv4 interprocess communication (IPC) through the loopback device. The number is close to the expected value when using 10GbE (115 μs), i.e. its resembles the possible final setup with 10GbE.

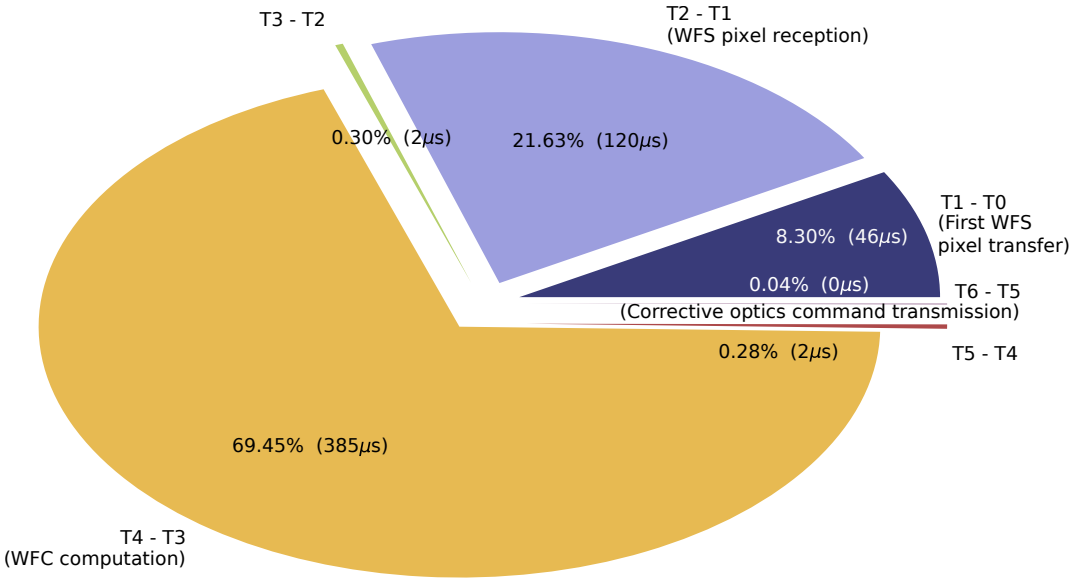


Figure 6. Mean shares of corrective optics command delay.

6. FUTURE WORK

The presented results are part of ongoing work to realize a fully functional RTC prototype fulfilling all METIS SCAO requirements. For the final design review, a number of activities are planned to improve the prototype with regard to functionality and to improve our understanding of its performance.

Currently, the prototype performs a wavefront reconstruction only using a static set of parameters. In order to achieve a more realistic prototype status, the following activities are planned:

1. Implement a temporal controller. So far, only the wavefront reconstruction is done. Depending on the computational demand of the temporal controller, more than two GPUs will be necessary in the HRTC.
2. Implement HRTC parameter update during closed loop operation. Until now, the HRTC works with fixed parameters. This limitation is fine for first tests. In operation, it will be necessary to update parameters like control matrices in order to optimize the AO performance. This will increase the network traffic and will significantly impact the memory throughput demand. Amongst other things, it has to be checked if the parameters can be updated without affecting the real-time control loop.
3. Design the SRTC. The HRTC is supervised and optimized by the SRTC. The necessary SRTC algorithms need to be designed and implemented. The communication between HRTC and SRTC needs to be implemented.

In the current design of the prototype, several modules reside in the same workstation, e.g., the simulated WFS camera and `hrtcMain`. In the final deployment, these modules will be distributed to different systems on a network which will introduce additional latencies due to communication lag. During the prototyping phase, we intend to simulate the final deployment as close as reasonable and foresee the following tasks:

1. Implement a ELT Central Control System (CCS) simulator in order to transmit corrective optics commands to a remote host and to receive the M4/M5 echo.
2. Setup a distributed SCAO system with clock synchronization as depicted in figure 7. Such a setup is close to the reality at the ELT because the interprocess communication is delayed by an Ethernet network. The precision time protocol (PTP) is used to synchronize clocks in sub-microsecond range.

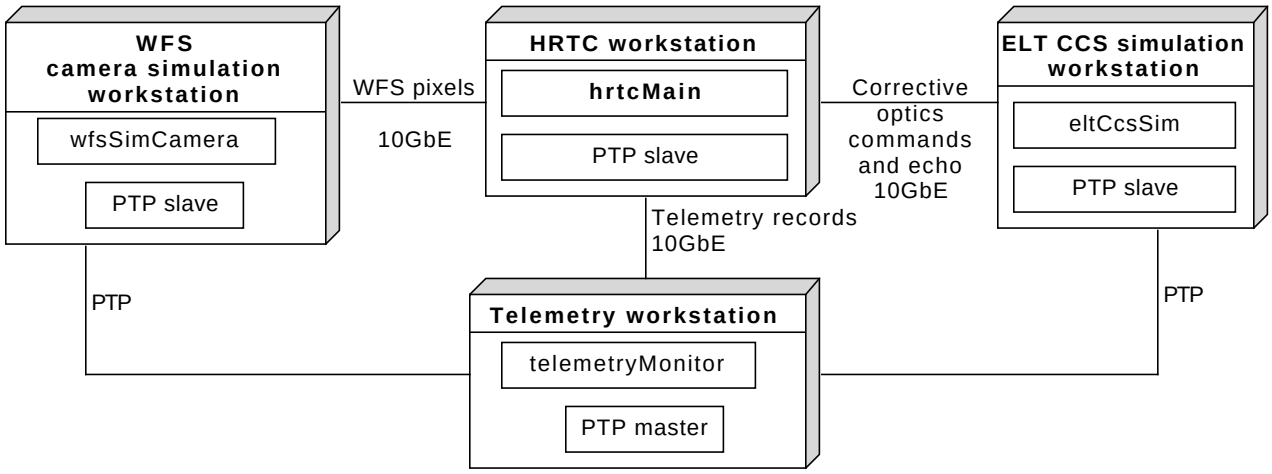


Figure 7. Distributed experiment setup.

Additionally to the functional improvements, we intend to optimize the prototype with regard to computation time. For the final phase of the prototyping the following activities are planned:

1. Eliminate outliers. Figure 5 shows a small number of outliers that come close to the limit of RTC computation time. We contemplate to use a real-time Linux kernel to counteract the outliers.
2. Further optimize the algorithms. We see room for improvement regarding memory management and parallelization of the algorithms to decrease overall computation time. This might be a necessary activity when the full control algorithm including temporal control has been implemented.

It is planned to deliver an full HRTC prototype in the final design review which is currently planned for 2021.

7. CONCLUSION

The sheer scale of the Extremely Large Telescope requires an unprecedented computational effort to drive the adaptive optics within the given timing constraint running the AO main loop at up to 1 kHz. In this paper we present first results of the prototyping activity for a hard real-time computer within in the METIS project. We created a test environment that is flexible enough to test different hardware accelerators. Here we discuss a single-node HRTC prototype using two COTS GPUs, but the design also allows multi-core (NUMA) CPUs.

Based on a MVM-reconstructor without temporal controller, we analyzed the theoretical memory throughput straining both the network but also the local memory and came to the conclusion that the chosen hardware setup using a single node with two GPUs should suffice to stay within the requirements.

We tested our theoretical findings in an experiment using two computers on a network, where a simulated WFS camera resides on the same node as the HRTC main computation, and a telemetry monitor on a different node receives the telemetry stream. For the given setup, we found that we can indeed process the WFS pixel stream and produce a telemetry data stream at a loop rate of 1kHz. The mean computation time of 555 μs stays well within the budget of 909 μs defined by the requirements. We observed a number of outliers, but these did not exceed the budget, either.

The presented results are a snapshot of ongoing work towards the final prototype. In the following months, we intend to add a temporal controller to get a final estimate of the demand regarding computational throughput. Furthermore, we intend to create a more realistic distributed deployment including a simulated ELT CCS workstation, and deploying the simulated WFS camera on its own node.

REFERENCES

[1] Brandl, B. R., Absil, O., Agócs, T., Baccichet, N., Bertram, T., Bettonvil, F., van Boekel, R., Burtscher, L., van Dishoeck, E., Feldt, M., Garcia, P. J. V., Glasse, A., Glauser, A., Güdel, M., Haupt, C., Kenworthy, M. A., Labadie, L., Laun, W., Lesman, D., Pantin, E., Quanz, S. P., Snellen, I., Siebenmorgen, R., and van Winckel, H., "Status of the mid-ir elt imager and spectrograph (metis)," **10702** (2018).

[2] *E-ELT Instrument Adaptive Optics Real-Time Computer Timing and Latency Considerations.*

[3] Kern, P., Lena, P., Gigan, P., Fontanella, J.-C., Rousset, G., Merkle, F., and Gaffard, J.-P., "Come-on: an adaptive optics prototype dedicated to infrared astronomy," in [*New Technologies for Astronomy*], **1130**, 17–28, International Society for Optics and Photonics (1989).

[4] Kern, P., Merkle, F., Gaffard, J. P., Rousset, G., Fontanella, J. C., and Lena, P., "Prototype of an adaptive optical system for astronomical observation," in [*Real-Time Image Processing: Concepts and Technologies*], **860**, 9–15, International Society for Optics and Photonics (1988).

[5] "Motorola 68020." https://en.wikipedia.org/wiki/Motorola_68020. (Accessed: 2 October 2019).

[6] Merkle, F. and Hubin, N. N., "Adaptive optics for the european very large telescope," in [*Active and Adaptive Optical Systems*], **1542**, 283–292, International Society for Optics and Photonics (1991).

[7] "NAOS." http://www.lesia.obspm.fr/NAOS.html?var_recherche=naos. (Accessed: 2 October 2019).

[8] Brandner, W., Rousset, G., Lenzen, R., Hubin, N., Lacombe, F., Hofmann, R., Moorwood, A., Lagrange, A.-M., Gendron, E., Hartung, M., et al., "Naos+ conica at yepun: first vlt adaptive optics system sees first light," *The Messenger* **107**, 1–6 (2002).

[9] Rabaud, D., Chazallet, F., Rousset, G., Amra, C., Argast, B., Montri, J., Dumont, G., Sorrente, B., Madec, P.-Y., Gendron, E., et al., "Naos real-time computer for optimized closed-loop and online performance estimation," in [*Adaptive Optical Systems Technology*], **4007**, 659–670, International Society for Optics and Photonics (2000).

[10] "Texas Instruments TMS320C40." http://www.ti.com/product/TMS320C40?keyMatch=TMS320C40. (Accessed: 2 October 2019).

[11] Jenkins, D. R., Basden, A. G., and Myers, R. M., "A many-core cpu prototype of an mcao and ltao rtc for elt-scale instruments," *Monthly Notices of the Royal Astronomical Society* **485**(4), 5142–5152 (2019).

[12] Jenkins, D., Basden, A., and Myers, R., "A prototype architecture for elt-scale mcao and ltao based on many core cpu technologies (poster presented at ao4elt6, quebec, canada)," (2019).

[13] Clénet, Y., Buey, T., Gendron, E., Hubert, Z., Vidal, F., Cohen, M., Chapron, F., Sevin, A., Fédou, P., Barbary, G., et al., "The micado first-light imager for the elt: towards the preliminary design review of the micado-maory scao," in [*Adaptive Optics Systems VI*], **10703**, 1070313, International Society for Optics and Photonics (2018).

[14] Gratadour, D., Morris, T., Biasi, R., Deneux, H., Bernard, J., Buey, J.-T., Doucet, N., Ferreira, F., Laine, M., Perret, D., et al., "Prototyping ao rtc using emerging high performance computing technologies with the green flash project," in [*Adaptive Optics Systems VI*], **10703**, 1070318, International Society for Optics and Photonics (2018).

[15] "NVIDIA DGX-1 datasheet." https://images.nvidia.com/content/technologies/deep-learning/pdf/61681-DB2-Launch-Datasheet-Deep-Learning-Letter-WEB.pdf. (Accessed: 9 October 2019).

[16] Ferreira, F., Sevin, A., Bernard, J., and Gratadour, D., "Micado-maory scao rtc system prototyping: assessing the real-time capability of gpu (poster presented at ao4elt6, quebec, canada)," (2019).

[17] "ZeroC Ice: a remote procedure call framework." https://zeroc.com/products/ice. (Accessed: 2 October 2019).

[18] *Wavefront Sensor Cameras MUDPI Data Packet Description.*

[19] "GeForce 20 series." https://en.wikipedia.org/wiki/GeForce_20_series. (Accessed: 2 October 2019).