A Real-time Control Computer for the E-ELT

Document: GF-PDR-09

Supervision strategy

Version 1.1

18th January 2016

Change Record

Version	Date	Author(s)	Remarks
0.1	11 Jan 2016	D. Gratadour	Initial skeleton version
0.9	15 Jan 2016	N. Doucet	Major revisions
1.0	18 Jan 2016	D. Gratadour	Final version
1.1	19 Jan 2016	F. Ferreira	Minor revisions

Applicable Documents (AD)

These are the Green Flash PDR documents

No.	Title	Reference	Issue	Date
AD01	Introduction	GF-PDR-01		
AD02	Management plan and WP definition	GF-PDR-02		
AD03	Requirements Specification	GF-PDR-03		
AD04	System Architecture	GF-PDR-04		
AD05	Distributed GPUs for real-time HPC	GF-PDR-05		
AD06	FPGA Solution for hard real-time	GF-PDR-06		
AD07	Interconnect Strategy	GF-PDR-07		
AD08	Interface Control Document	GF-PDR-08		
AD09	Supervision Strategy	GF-PDR-09		

Reference Documents (RD)

These are documents external to the Green Flash project

No.	Title	Reference	Issue	Date

Acronyms and abbreviations

Table 1 Acronyms and Abbreviations

AD	Applicable Document
AO	Adaptive Optics
CANARY	Durham/LESIA on-sky AO demonstrator
CPU	Central Processing Unit
CUDA	NVIDIA GPU based software development language
DARC	Durham AO Real-time Controller
DDS	Data Distribution Service
DM	Deformable Mirror
DRAGON	Durham laboratory-based AO demonstrator bench
ELT	Extremely Large Telescope
E-ELT	European ELT
ESO	European Souther Observatory
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HLS	High Level Synthesis
HPC	High Performance Computing
MIC	Many Integrated Core
MVM	Matrix-Vector Multiplication
NIC	Network Interface Controller
PCIe	Peripheral Component Interconnect express
RD	Reference Document
RTC	Real-Time Control
RTL	Register Transfer Level
SIMD	Single Instruction Multiple Data
SPARTA	ESO VLT AO Real-time Control System
SHERE	VLT Planet finder instrument
UDP	User Datagram Protocol
UK ATC	United Kingdom Astronomical Technology Centre
VLT	Very Large Telescope
WFS	Wave-Front Sensor
WP	Work Package

Table of Contents

1 Scope

This document specifies the supervision strategy that will be adopted in Green FLASH.

Green FLASH targets the prototyping of a RTC for an E-ELT tomographic AO system dimensioning. In this document we concentrate on the most computationaly challenging part of the supervisor module : the AO loop optimization aiming at providing the real-time AO data pipeline with regular tomographic reconstructor matrix update.

2 Supervision strategy for tomographic AO systems

A critical subsystem of the AO RTC is the supervisor module. Its role is to feed the RT box at a regular rate with a *reconstructor* matrix, computed from a statistical analysis of the measurements. This process involves the direct inversion of the WFS measurements dense covariance matrix, the size of which being 90k x 90k in our target E-ELT first light instrument. The computational load for the inversion of this dense symmetric matrix, through Cholesky factorization, is quite significant at this scale but can be handled efficiently using GPUs[1].

We propose to reuse the GPU platform, implemented as a demonstration of concept for the RT box, as a testbed for the supervisor module. The performance of Cholesky inversion will be assessed on this platform reusing the work described above but on an ARM + GPU platform relying programming models like OmpSs. The achievable update rate to the RT box will be demonstrated depending on the cluster dimensioning. The porting effort will be part of an ongoing collaboration with the Extreme Computing Group at KAUST University[2]. More details on this activity is given in the next subsection.

This demonstration of performance relates to the objective 1.4 of the project.

Finally we propose to build a full functionality demonstrator for an AO RTC, using the core components developed in the prototyping phase and scalable to the specifications of the first light E-ELT instruments. The output of the performance assessment of the various prototypes will be used to define the final architecture to be tested. While the RT box and the supervisor module will be tested independently, their interconnection is a critical aspect of this phase of the project. A large (64 Gbytes) matrix must be uploaded to the RT box at a regular rate without interrupting / impacting the AO loop. We will again rely on the smart interconnect concept previously described to implement a custom approach for this regular loop optimization process and measure the achievable rate. This will provide critical inputs to the instruments preliminary design studies in term of achievable AO performance.

Because the final sensors and deformable optics won't be available by the end of the prototyping phase, it is mandatory to implement a simulation strategy providing the essential tools to validate the various features of the final demonstrator. A comprehensive program to develop a real-time simulator is included in the green Flash project. It relies on the use of an efficient simulation code, with various levels of accuracy leading to various simulation frame rates, and the realistic emulation of the interfaces and data transfer protocols from the sensors and to the optics. This work will be based on existing tools (software and hardware) developed at Observatoire de Paris and University of Durham.

Thanks to a low precision simulation mode, the RT box performance determinism will be assessed at

1 A. Charara et al., Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2014, DOI: 10.1109/SC.2014.27

2 http://ecrc.kaust.edu.sa/Pages/Home.aspx

maximum frame rate under realistic conditions. In this mode, the simulator is mainly publishing data like a WFS camera would do, at maximum frame rate, and is using the RT box new command vector uploads to measure its response time. A medium precision simulation, including loop retro-action but a simple model for WFS images, will be used to assess the system output reliability at a quasi-real-time rate. The goal here is to validate the control strategy performance in terms of achievable AO correction. A slow high precision simulation mode, will be available to validate the core algorithms and operation protocols under realistic conditions but at reduced frame rate.

This full functionality demonstration relates to obj. 3.1, 3.2 and 3.3 of the project.

The most compute intensive task is the computation of the *reconstructor* matrix. It involves the inversion of a large covariance matrix either through matrix factorization or through direct inversion. In the latter case, an embarrassingly parallel problem with a numerical complexity scaling with N^3, an optimized implementation should optimize the usage of computing cores rather than the memory bus. However, the underlying algorithms require frequent synchronizing of global communications, which represents a bottleneck that limits performance. This bottleneck will be further exacerbated when concurrency will reach billions of core in Exascale systems and should be addressed as a general issue for the HPC community. Concerning the AO application, optimizing the compute performance means larger update rate of the *reconstructor* matrix hence better image quality at the output of the telescope and incidentally larger science return. Given the matrix size (up to 100k x 100k), and the requirements in terms of update rate (at the scale of the minute), the compute performance should be of the order of 150 to 200 TFLOPS. Considering current GPUs theoretical peak performance for double precision matrix-matrix multiply (at the core of the Cholesky approach for matrix inversion) and ideal scalability in distributed memory configurations, this figure means clustering about 100 GPUs. Our goal is to reduce this number so as to reduce the cost of the infrastructure and maximize its energy efficiency. The AO RTC supervisor module is thus the ideal testbed to assess optimized implementations on a HPC facility of rather moderate size but with strong constraints on time-to-solution.

A collaboration with the MORSE project (Matrices Over Runtime System at Exascale[3]) has allowed us to developed an optimized pipelined approach for the computation of the *reconstructor* matrix[4]. The goal of the MORSE project is to design linear algebra methods that achieve the fastest possible time to an accurate solution on large-scale multicore systems with hardware accelerators, using all the processing power that future high end homogeneous and heterogeneous systems can make available. It relies on the use of a dynamic run time system to schedule computational tasks simultaneously on various compute devices and a data flow programming model based on the use of direct acyclic graphs for an efficient scheduling in which the tasks are executed out-of-order and scheduled according to a critical path for the execution. This efficient approach has allowed us build a new implementation that outperforms asymptotically previous state-of-the-art implementations up to 13-fold speedup.

With Green Flash, new challenges and new opportunities for these innovative programming models designed for Exascale computing could be studied. On one hand, to push further the energy efficiency, we propose to use ARM processors as host CPUs in the supervisor module's nodes. This requires to use new programming models supporting this architecture such as OmpSs[5]. We plan on developing our collaboration with the MORSE project on that aspect and propose the supervisor prototype of

3http://icl.cs.utk.edu/morse/

4 A. Charara et al., Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2014, DOI: 10.1109/SC.2014.27

Green Flash as a test-bench for the performance assessment of some core components of this library. On the other hand, we believe that the concept of smart interconnect could be efficiently exploited through the use of such programming models. This collaboration will allow us to study how this concept could be integrated in these programming approaches and how it would impact the specifications of these products (available IP cores / blocks, drivers, protocols) for a wider application spectrum.

Despite the importance of emerging massively parallel hardware that is driving a revolution in HPC, it is very often the application software that represents a huge investment in manpower and a resource for the future. This is equally true for commercial packages and for large science applications. Too often, such software is lost when it cannot be ported to newly emerging hardware. This will be a major issue for HPC in the coming decade and for large science projects such as the E-ELT that must be maintained for decades to come. There is also a substantial challenge in writing or modifying application software to take full advantage of parallel processing techniques. This challenge is to write software that is naturally parallel, is abstracted from the hardware and, perhaps above all, can be easily and cheaply maintained over long periods. One approach to all of these problems emerged from the Khronos group in the form of the Open Computer Language, OpenCL. This is an open standard for parallel programming of heterogeneous systems.

The ideal outcome of making use of OpenCL for application software is for it to be immediately portable to different accelerator hardware. Estimating the extent to which this is possible is a current research activity at the CfAI in Durham and will form a part of the evaluation process for the ecosystem for the RTC prototype. Some of the questions that we aim to answer are: Can the same code be run on CPUs, GPUs, MICs and FPGAs with minimal changes? How efficient is such code compared to that optimized by tools dedicated to specific hardware? What is the roadmap for the provision of OpenCL support into the future? The last of these must not be overlooked since the HPC industry must provide support for OpenCL along with emerging new hardware if such an approach is to be practical. Whilst it may not be practical for all of the application code in the prototype RTC to be in OpenCL, there is at the very least a likely role for it in providing an abstraction layer between such software and the RTC hardware. Currently, all of the accelerator hardware mentioned above either is already or is moving towards providing support for OpenCL. This has allowed us to start an investigation into the implementation of AO specific algorithms in OpenCL on multiple hardware platforms. We have already demonstrated that, for one specific AO algorithm, near identical OpenCL code can be run on GPU hardware from the two major suppliers, NVidia and AMD. This evaluation will be extended to include MIC and FPGA accelerators many of which also now support OpenCL.

3 Supervision pipeline definition

3.1 Algorithm definition

The supervisor module provides a tomographic reconstructor matrix given measurements from the wavefront sensors. From those measurements we compute an experimental covariance matrix (called C_{mm_e}) between all the measurements, and since it is not possible to get an infinite amount of measurement, this matrix is biased and an other covariance matrix (C_{mm_f}) should be fitted to the first one. The fit is done using a least square minimization and a known model ("learn" step). It provides a set of fitted parameters used to compute the fitted matrix ("apply" step). The covariance matrix between the measurements and the measurement of a truth sensor (C_{tm}) is also computed

5 https://pm.bsc.es/ompss

from the fitted parameters in the "apply" step. Finally, we compute the product of the C_{tm} by the inverse of the C_{mm_f} ($R' = C_{tm} . C_{mm\_f}^{-1}$) and the tomographic reconstructor as the product of a command matrix C_{mat} by the previous computed matrix (R' in the figure below). The command matrix C_{mat} is matrix is given and does not need to be computed in this module.

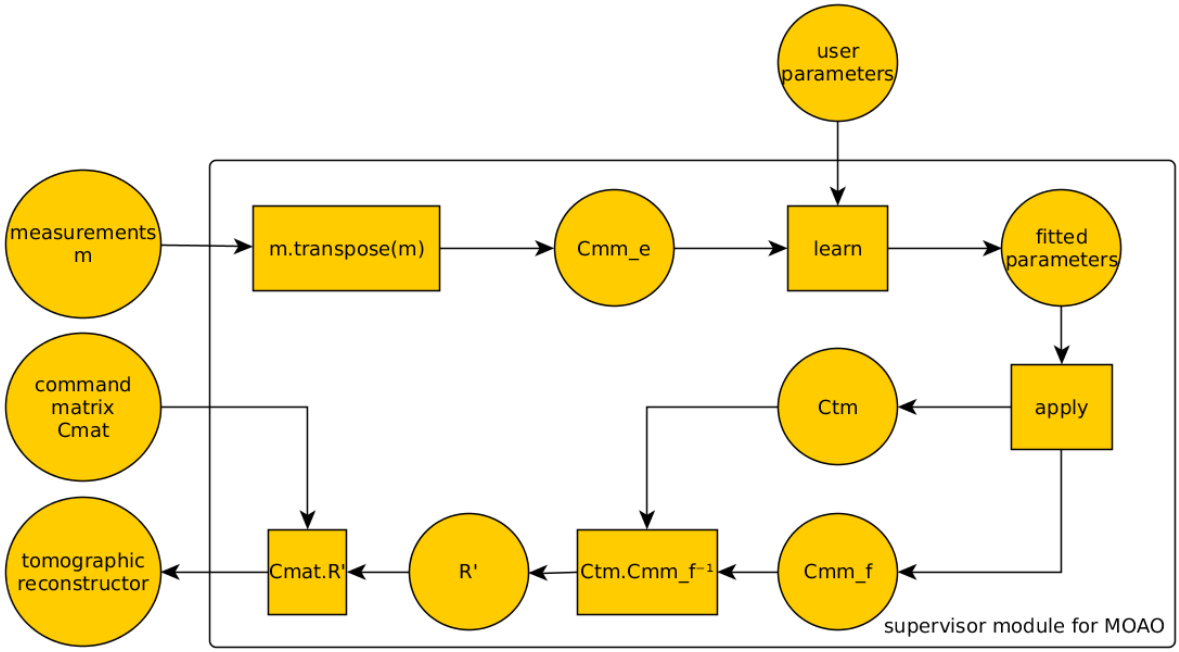

Figure: supervisor module for MOAO

The figure above represents the supervisor module pipeline in the Multiple Object Adaptative Optic (MOAO) case. The Multi Conjugate Adaptative Optics (MCAO) pipeline is similar, except that the C_{tm} matrix is a concatenation of multiple covariance matrices (between measurements and truth sensors).

Another implementation of the apply step return a modified C_{mm_f} noted $C_{\phi m}$ so that the product of the C_{tm} by the inverse of $C_{\phi m}$ is the tomographic reconstructor. This implementation requires more parameters for the apply step but the command matrix is no longer needed nor the last matrix matrix multiplication.

Due to the size of the covariance matrix (up to 100k x 100k), two issues arise: the memory required to store the matrices and the time needed to complete the computation of the tomographic reconstructor. The operations of the supervisor module that are the most concerned are the inversion of the fitted covariance matrix and the learn step.

The inverse of the covariance matrix (C_{mm_f}) can be computed using a Cholesky factorization. In this case, three steps are required. First a Cholesky factorization, then the inversion of the factorized matrix end finally a symmetric matrix-matrix products. But we might also want to filter some eigenvalues, so a eigenvalue decomposition is also be considered, in this case the steps are an eigenvalue decomposition then filter and symmetric matrix-matrix multiplications.

The "learn" step works fine on small cases using the levenberg-marquardt algorithm but this one requires a lot of memory in particular because it needs the Jacobian of the matrix to fit (in our case the covariance matrix of the measurements). Some other methods will be soon tested like the quasi-Newton like BFGS method, or jacobian free algorithm like Newton-Krylov will be studied.

3.2 Standard libraries

In order to benefit from the computing power of GPUs, relying on linear algebra packages is necessary. Those libraries include the BLAS (Basic Linear Algebra Subprograms) like cublas-XT[6] a cuda BLAS for multiple GPUs , clBLAS[7] an openCL BLAS or blasX[8] an improved cuda based BLAS proposes a two levels hierarchical tile caches[9] (this list is not exhaustive as well as the other lists of this section).

In fact we wont be using directly BLAS library because the module require more advanced linear algebra operations like Cholesky factorization. Instead, we will consider high performance linear algebra package, that provide those operations.

- MAGMA[10] is a dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures, including"Multicore+GPU" systems. MAGMA provide versions for CUDA, OpenCL and Mics. MAGMA is working with StarPU runtime system. We currently have a partnership with developers of this library.
- viennaCL[11] an incomplete linear algebra library on multicore systems providing support for CUDA, OpenCL and MIC
- CULA[12] is a GPU accelerated linear algebra (with multi GPUs support) but only simple precision library is available for free and support is mentioned for openCL or MIC.
- LibFlame[13] is a portable library for dense matrix computations, providing much of the functionality present in LAPACK. In fact, libflame includes a compatibility layer, FLAPACK, which includes a complete LAPACK implementation, containing its own runtime system: superMatrix. The documentation indicate that GPU routine are implemented.

The previous libraries does not include MPI parallelism for such parallelism we need to look for other libraries like:

- dplasma[14] a dense linear algebra library aiming for distributed systems (MPI parallelism) with GPU accelerators and also provide support for Xeon Phi

6 https://developer.nvidia.com/cublas

7 https://github.com/clMathLibraries/clBLAS

8 https://github.com/linnanwang/BLASX

9 https://arxiv.org/pdf/1510.05041.pdf

10 http://icl.cs.utk.edu/magma/

11 http://viennacl.sourceforge.net/

12 http://www.culatools.com/

13 http://www.cs.utexas.edu/~flame/web/libFLAME.html

14 http://icl.cs.utk.edu/dplasma/index.html

- PETSc[15] a library providing routines and data structures for scientific application. PETSc support MPI, GPUs (CUDA and OpenCL) and hybrid MPI-GPU parallelism. Using PETSc with OpenCL require viennaCL library.

In order to schedule task properly, improve data management and communication, we need a runtime system. Some of the libraries mentioned above contains one but we can try to switch to another on like OmpSs a high-level, task-based, parallel programming model supporting SMPs, heterogeneous systems (like GPGPU systems) and clusters.

3.3 Possible hardware implementations

Considering the size of a single covariance matrix (up to 100k x 100k), the memory requirements are quite large too (up to 80 Gbytes), hence this module require distributed memory. We can consider to use large but few CPUs (fat nodes) or on the contrary more of smaller CPUs (ARM) to store all the data necessary to the execution of the module. A third possibility would be to store the covariance matrix on GPUs or Xeon Phis persistently. This last option may need more effort in terms of development but may also be profitable in term of communication time. For instance using tile algorithm, operating on multiple tile stored on the same device wont need any communication.

All these hardware implementation will be considered in the Green Flash roadmap, the standard CPU option being considered as the reference in terms of compute performance and energy efficiency.

3.4 Efficient link to the real-time data pipeline

Regular updates of the reconstructor matrix to the real-time data pipeline are required in order to follow the evolution of atmospheric turbulence conditions. As described in document AD03. We will assess such offloads as the full functionality prototype will be assembled (task 8.3 of the project, see AD02) and based on either a standard interconnect solution or on the smart interconnect solution developed in the project (see AD07).

4 Implementation plan

The work on the supervisor module spreads over several WP and several tasks of the project. They are detailed in the following (see AD02 for a complete list of WP).

4.1 *Task 4.2: Supervisor module based on GPUs*

Objectives. The goal of this task is to address the most compute intensive task of the supervisor module: the computation of the *reconstructor* matrix and more generally the achievable performance of .large matrix (up to 100k x 100k) inversion in a pipeline for the computation of the *reconstructor* matrix on an ARM + GPU cluster. The targeted the compute performance is of the order of 150 to 200 TFLOPS and our goal is to reach this performance with a minimum number of GPUs on an energy efficient architecture.

On the algorithms side, we will rely on an existing collaboration with the MORSE project that has allowed us build a new implementation that outperforms asymptotically previous state-of-the-art implementations on x86 architectures with an up to 13-fold speedup. We propose to extend this study to a distributed configuration relying on ARM processors as host CPUs, reusing the hardware assembled in the context of the previous task.

15 http://www.mcs.anl.gov/petsc/

In the supervisor module, the smart interconnect features could be used to boost performance and the concept study of integrating these smart features in the programming model could be tested in this task, in relation with the developments led in task 8.3.

This task includes:

- an analysis of the compatibility of the RT-box design for the supervision and possible evolution of the configuration to adapt for the supervisor need (in particular the role of smart interconnects)
- the deployment of the supervisor core code on this platform relying on optimized dense linear algebra libraries
- performance assessment of the overall supervision pipeline with respect to the scale of the facility in terms of throughput and energy consumption
- implement and test new optimized supervision strategies on this architecture based on innovative programming models taking advantage of the smart features in the interconnect

4.2 *Task 4.3: Xeon Phis as alternative solution*

Objectives. This task will investigate the use of the Xeon Phi as an alternative acceleration technology for the hard real-time data processing in the RTC prototype.

The Xeon Phi is a Many Integrated Core (MIC) technology that is emerging as an alternative to GPU based accelerator cards. Whilst this processor has fewer ‘cores’ than GPUs, the cores provided are very similar to the standard X86 cores of a multi-core CPU processor. This means that the Phi can be programming in a very similar way to a standard processor. Code that has been optimised for a standard processor can be easily ported to this device. This is in contrast to the additional effort and expertise required to make use of the proprietary development API known as CUDA that is provided by NVidia, the leading manufacturer of GPUs. The roadmap of these MIC processors may be very important to HPC over the next few years. At present, they are provided as a PCIe accelerator card. However, it is very possible that such processors will merge with standard processor designs and take on the latter role. It is already possible to run an operating system in a stand-alone way on the Phi which is not the case for GPUs.

This task will entail investigating the Phi both as an accelerator and as a stand-alone processor. It can take various roles within the prototype RTC system, as a part of the hard real-time data pipeline, as an accelerator for soft real-time calculations and as an accelerator for real-time simulations. The full performance of the technology in all of these roles will be investigated and compared to that of GPUs. Since the Phi may be used running an operating system, the measurement of jitter in a phi-based data pipeline will be key. As well as absolute performance, the energy efficiency and relative cost of a Phi based solution will be investigated along with the scalability of such a system to provide for a high order system mapped to multiple Phi cards.

One key result of this evaluation will be the ease of programming of this technology. The Phi is provided with an API for OpenCL as well as dedicated compilers. The various models for the development of software for the Phi will be implemented and compared with each other and with those available for GPUs.

This task will be designed in such a way that a standard series of tests can be applied to additional MIC technology that emerges over the duration of this project. An example is the Tilera multi-core technology that is already available and that will be assessed for real-time HPC applications in a very similar manner to the Phi.

4.3 *Task 7.3: Algorithms and libraries*

Objectives. This task concentrates on the definition of the various algorithms and libraries to be used in the AO RTC pipeline, including the real-time box and the supervisor module.

At the level of the RT box, while the baseline algorithm is the MVM with a *reconstructor* matrix, a number of other control strategies exist providing either better or more homogeneous image quality or faster implementations. In this task, the goal is to provide the framework, in terms of available libraries for the targeted hardware and algorithms, to implement these advanced control schemes on the various architecture options. An analysis of the portability of these algorithms will be led based on the availability of standard libraries. Emphasis will be put on the ARM architecture as well as the possibility to port these approaches to the FPGA either using OpenCL or relying on optimized C code, ported to the FPGA using optimized HLS from QuickPlay. Targeted control approaches include:

- Linear Quadratic Gaussian control (LQG)
- Fractal Iterative *reconstructor* (FriM)
- Fourier based AO control

At the level of the supervisor module, the main challenge resides in the computation of the tomographic *reconstructor* which involves the inversion of a large matrix (typically100k x 100k elements). To perform this computation efficiently, we have developed an optimized approach based on the use of GPUs as accelerators and using an optimized linear algebra library: MORSE which will be assessed in task 5.2 of WP5. In this task, we propose to:

- explore alternative libraries to perform the tomographic *reconstructor* computation
- derive new algorithms and approaches for the tomographic problem either to increase AO performance or to reduce execution time
- study innovative supervising strategies, designed to be coupled to the alternative control schemes mentioned above

As in the previous case, the possibility of implementing these supervising strategies will be assessed on the basis of the availability of relevant libraries for the targeted architecture. We believe that the concept of smart interconnect could be efficiently exploited through the use of such programming models. Our existing collaboration with the MORSE project will allow us to study how this concept could be integrated in these programming approaches and how it would impact the specifications of these products (available IP cores / blocks, drivers, protocols) for a wider application spectrum.