SPARTA roadmap and future challenges

Enrico Fedrigo[1a], Robert Donaldson[a]
[a]European Southern Observatory, Karl-Schwarzschild-strasse 2, 85748 Garching, Germany;

ABSTRACT

SPARTA, the ESO Standard Platform for Adaptive optics Real Time Applications, provides a generic decomposition in functional blocks that can be applied, unchanged, to a variety of different AO systems, ranging from very small single conjugate AO with less than 100 actuators to much bigger and faster systems. For AO systems under development, SPARTA provides an implementation for all those functional blocks that are mapped to currently available technologies. The E-ELT with its instruments poses new challenges in terms of cost and computational complexity. Simply scaling the current SPARTA implementation to the size of E-ELT AO system would be unnecessary expensive and in some cases not even feasible. So, even if the general architecture is still valid, some degree of re-implementation and use of new technologies will be needed. This paper analyses the new general requirements that the E-ELT and its instruments will pose and introduces promising technology and solutions that could replace the current ones and show how the SPARTA architecture could evolve to address those new requirements.

Keywords: real time, real time computer, adaptive optics, control systems

1. INTRODUCTION

During the phase A of the E-ELT instrument studies, some consortia have considered using adaptive optics either provided by the telescope system, or by a separate AO module or by an internal AO sub-system. Those have shown a remarkable degree of variability in terms of required computing power, as shown in Table 1.

Table 1: Main requirements for the Phase-A instruments with AO and telescope systems

	AO Class	AO Module	WFS	Size	DM	size	Frq	Complexity
EAGLE	MOAO	Internal	6+4	84x84	1 x IFU	84x84	250	200 GMAC[2]
EPICS	XAO	Internal	1	210x210	1	211x211	2500	6 TMAC
MICADO	SCAO MCAO	Internal MAORY	1	84x84	1	7200	1000	80 GMAC
METIS	LTAO SCAO	ATLAS Internal	1	84x84	1	7200	1000	80 GMAC
MAORY	MCAO	-	6	84x84	3	7200 1700 2200	500	370 GMAC[3]
ATLAS	LTAO	-	6	84x84	1	7200	500	240 GMAC[4]
Telescope1	N-GLAO	Internal	3	68x68	1	7200	500	26 GMAC
Telescope2	L-GLAO	Internal	4	68x68	1	7200	500	26 GMAC

For comparison, Table 2 presents what SPARTA delivers for the AOF in terms of computing power, which does not indicate a limit of the performance of the platform itself but what the particular instance requires.

[*] enrico.fedrigo@eso.org

[2] Only 1 IFU, total complexity can be obtained by multiplying the given number by 20 and adding an LGS-GLAO. Even if the total complexity of EAGLE approaches the one of EPICS, EAGLE is intrinsically parallel since all 20 IFUs are independent and the technological issue is put on the data distribution network rather than to the bare bone RTC.

[3] Assumes straight MVM WFS->DM, compatible with POLC with off-line extra DMxDM projection

[4] Same as (3)

Adaptive Optics Systems II, edited by Brent L. Ellerbroek, Michael Hart, Norbert Hubin, Peter L. Wizinowich,
Proc. of SPIE Vol. 7736, 77364O · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.857109

Table 2: VLT second generation AO instruments, SPARTA delivered computing power

	AO Class	AO Module	WFS	Size	DM	size	Frq	Complexity
SPHERE	XAO	Internal	1	40x40	1	1377	1500	5.2 GMAC
AOF	LTAO	Internal	4	40x40x4	1	1170	1000	11.8 GMAC

Figure 1 shows the complexity as a function of time assuming arbitrary first light dates for the various instruments starting from the foreseen telescope first-light in 2018. Order of instruments is arbitrary. Systems currently in operation in Paranal are also shown, placed on their respective commissioning date. One can recognize 3 clusters, one with GLAO+SCAO modes, MAORY/ATLAS/EAGLE and separately EPICS. Bars account for a range of different requirements.
The two lines centered in the AOF coordinate describe the expected technological evolution in terms of the so-called Moore's Law.

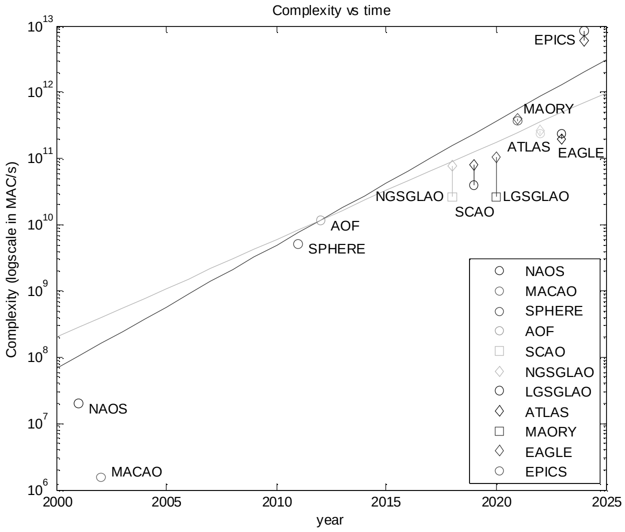

Figure 1: E-ELT systems complexity in comparison with VLT systems. Lines are prediction based on Moore's law, curved line includes economical limitations.

2. THE MOORE'S LAW

Considerations about future trends in computing cannot avoid citing the Moore's law[5].
The Moore's law describes a long-term trend in the history of computing hardware, in which the number of transistors that can be placed on an integrated circuit has doubled approximately every two years. The actual period was about 20 months. The purple line uses the '20 months' rule, while the yellow line uses '24 months'. In both cases the complexity of almost all instruments lies below or around the expected trend in technological evolution, with the notable exclusion of EPICS. It is also to be noted that centering the lines around AOF is rather arbitrary since this is not the limit of the current technology in its incarnation (SPARTA), but instead is what it delivers in one of its instances. Limits are actually higher, but even increasing the height of the two Moore's law line would not reach the requirements posed by EPICS.
Is the Moore's law to be believed for the future?
The literature abounds in papers treating this subject in details. First of all it has to be noted that the Moore's law consider the growth in the number of components that can be squeezed in the same area and that does not always translate into greater practical CPU performance. Therefore the technological evolution predicted in Figure 1 does not immediately refer to more powerful architectures, but simply to *denser* CPUs. There are cases where a roughly 45% increase in processor transistors have translated to roughly 10–20% increase in processing power (see [2]).
There are more factors limiting the computer power that can be obtained from a densely populated piece of silicon, such as internal bandwidth and storage speed. In fact CPU speed has improved as has the memory size. However memory access speed has failed to keep up resulting in the real bottleneck for high performance systems.
For years, processor makers consistently delivered increases in clock rates and instruction-level parallelism, so that single-threaded code executed faster on newer processors with no modification. Now, to manage CPU power dissipation, processor makers favor multi-core chip designs, and software has to be written in a multi-threaded or multi-process manner to take full advantage of the hardware, with the CPU speed stabilized at around 3GHz.
There are several papers on the subject, mostly focusing on the insurmountable technological barriers that will be reached, like the size of an atom. Certainly it looks like the speed limit has been reached, since in the last years processors stabilized their clocks at 3 GHz.

[5] Part of the text in the following paragraphs/sections extracted or adapted from Wikipedia.

There might be more fundamental limits. Sodan et al. ([1]) take the economical perspective. The current Moore's law would account for indefinite exponential growth that would eventually exceed the Gross World Product. The actual growth is the effect of two forces, the push to miniaturize and put more components on the unit of area and the exponential cost to fabricate those more and more complex chips, that also grows exponentially. Therefore limitations must be expected in the Moore's law for component density that should settle around a rate for doubling of the density every 8 years. Depending on the economic model used, the knee could happen between 2020 and 2025.

2.1 Technological Evolution: What To Expect

Given the previous arguments, we need to take a more realistic Moore's law (20 months for doubling components), then we need to add a factor to map from components to actual computing power (an heuristic factor 1.6) plus the long term economical limitations that add another dumping factor of 4.

We also consider that a concept based on current technology (SPARTA platform) is available and that with minor changes could serve the entry level NGS/LGS GLAO system. Therefore in Figure 2 we see a band in which we could reasonably expect the technological advances will bring current state-of-the-art computing machinery without requiring significant changes.

The revised forecast puts some more instruments outside the band of expected technological evolution. The plot says also that the technology used in MACAO or NAOS could not have been used today to build SPHERE and the AOF by simply waiting for advancement in computing power. In fact these curves imply that the technological evolution brought about a factor 10 improvement in 10 years, which is more or less what is observed if one compares MACAO CPU with currently available single board computers, which is what we call SPARTA Light, a low-end AO system. SPHERE and AOF are built using different technologies.

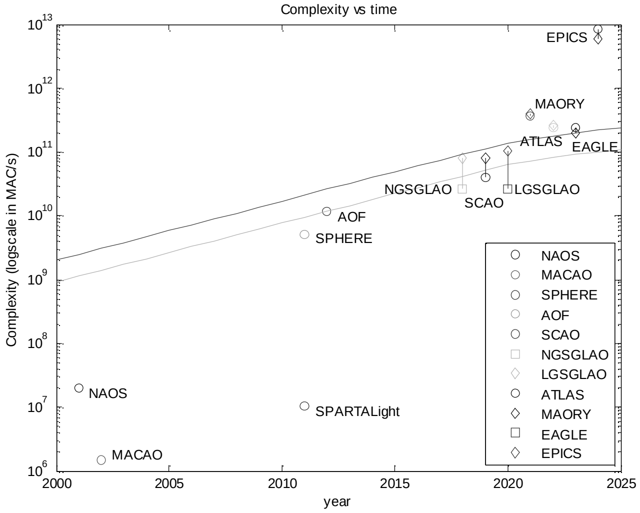

Figure 2: Corrected Moore's law for computing power with long term economical limitations

This is again what is to be done to build the SPARTA2 platform, a platform that could address the requirements of all E-ELT AO instruments.

We have already seen that CPU evolution has stabilized the clock frequency at around 3GHz since the performance gap between processors and memory limits the gains possible from further increasing processor frequency. The design direction currently employed for performance increases uses available chip space for multithreaded and multicore CPUs.

Today we have 8-cores processors and we expect an increase in the number of cores on chip in the following years, leading to CPUs of 16 or more cores in the near future[6]. However, as described in [1] and [2], there are strong limitations in the growth in computing power by the amount of real estate required for the cache and the crossbar switch. Caches are the most important feature for enhancing modern CPU performance because of the gap between CPU speed and memory-access times. The dominant approach to mitigating this gap exploits available chip space to provide more on-chip cache memory. This is an important design factor. Algorithms like MVM can deliver the full performance of the CPU only if the matrix can be hosted in the cache. The other critical component, which also depends on the cache architecture, is the interconnect, directly impacting the performance of the processor. The interconnect links together all on-chip components: core, caches and memory controllers. Nowadays the interconnect is implemented as a crossbar switch in order to reduce latency and contention.

However, the interconnect can become expensive: an 8 × 8 crossbar on-chip can consume as much area as five cores and as much power as two cores, therefore limiting the total number of cores that can be put on a single chip. Moreover, this component is power-hungry. Finally, also the number of off-chip pins is limited and put significant constraints on the scalability and programmability of multicore/multithreaded chips, as they impact the transfer rate of data to and from the cores. There is currently no technology in sight to drastically increase the pin count. Since more cores must be kept busy with instructions and data, the future for many-core designs may be limited.

[6] This text and following paragraphs extracted from referenced papers.

3. AREAS OF RESEARCH

In the previous section we have shown that simply waiting for the technological evolution to follow the Moore's law is rather risky. In the next sections we analyze which technologies we must focus on in order to arrive to a design of a real time computer able to meet the requirements listed in Table 1.

3.1 High Performance Computing Units

We have benchmarked that a modern system (Dual Nehalem or Core i7/Xeon with 4 cores each) can compute the matrix-vector multiply of the GLAO system in about 15 milliseconds, therefore it is not suitable for implementing a system like that (the requirement is for at least 2 milliseconds for 500Hz, 1 considering idle time activities). The SCAO computation completes, in the same architecture, in 22 milliseconds.

Even if conceivable for the telescope system RTCs, a CPU-based architecture will not scale up to the need of the instruments due to the overhead of distributing the computation across several machines given the inefficiency of CPU-based data communication. It is true that CPU like Nehalem would be placed in the mainstream course of technological evolution, therefore Moore's Law would apply to them. We have also shown in the previous sections that the expectations on future technological evolution based on Moore's Law are very optimistic.

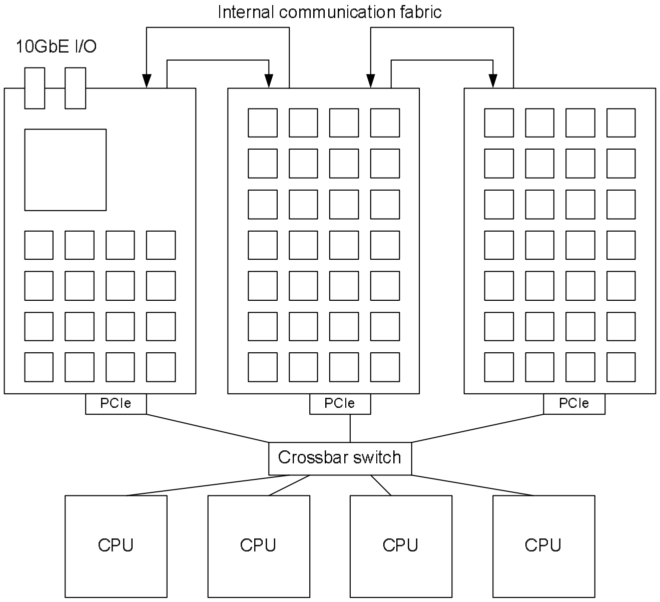

Figure 3: Concept for an FPGA array

Therefore we need to shift the architecture towards more performing parallel systems and employ more massively FPGAs because they are massively parallel, fully deterministic, they can integrate I/O with high speed and extremely low latency and they can be coupled with several banks of memory and run parallel memory I/O operations. Figure 3 shows a possible concept in which a front-end board hosts 2 10GbE transceiver and a large FPGA to manage all the I/O including the IP stack. The front-end board hosts also a number of smaller FPGAs for computation. A fast low-latency bus can connect in daisy chain more boards featuring a similar but large array of FPGAs. The system is managed via PCIe by the host computer.

For low-cost/low-performance systems, FPGA-based I/O can be coupled with CPUs or other co-processing boards, while for high-performance systems a massive array of FPGAs can be used. An aggressive architecture like the one proposed here will raise the starting point of the Moore's law in Figure 2 such that even EPICS can be targeted if not reached.

C-to-VHDL

Given the importance of FPGAs for future high-performance systems, one must consider the difficulties of programming such devices. They require a specialized language, VHDL, which is derived from the chip manufacturing domain.

This ends up in a complex development environment and a long development cycle as well as very specialized expertise.

On the one hand the authors' current experience has been extremely positive: we have been able to subcontract the development of FPGA code and receive results fully compliant with very tough specs since the beginning. Something not as easily achievable with CPU-based development, where average performance is very easy to reach, but top performance takes much longer time.

However recent technological developments allow for a quantum jump in the FPGA development practices, that is C-to-VHDL compilers. Those tools are now affordable and become more and more performant. This allows having a single development of high performance components that can run on a FPGA for extreme performance and on a CPU for low-cost/low-performance with a single code base, with clear gains in development and maintainability costs. It also allows complex mathematics to be implemented in FPGA within a reasonable development time, allowing algorithm debugging in C within a Linux host and later translation to FPGA for performance.

GPU Servers

An array of cluster machines is used to process data produced by the real time box on the fly. For some AO systems, current mainstream CPUs might be good enough to serve this purpose, but starting from moderately complex systems like ATLAS and MAORY that might not be enough, definitely not for EPICS. GPUs as coprocessors within a CPU-based high-end server shall be considered and a prototype evaluated. Future developments might allow feeding a GPU from a fast I/O card and therefore would enable its use in a real-time environment.

3.2 Interconnects

The RTC computing modules, including wave-front sensor camera and controller as well as mirror drive electronics, must be connected by an efficient, fast, low-latency deterministic network.

The most promising interconnects for RTC Computing Modules (i.e.: complete systems, like computers) is Ethernet. Ethernet has a very solid roadmap and it is used everywhere. Complex switches are available that can support all needed topologies (1:1, 1:n, n:1). 10Gb Ethernet is already commodity today, 40 GbE and 100 GbE is under heavy development and hardware modules (including FPGA IP cores) implementing the draft standard are already available.

However Ethernet has not been designed for real time applications and actually foresees a degree of non real-time in its specifications. Ethernet is never real-time when used over a bus: its low-level protocol, CSMA/CD, has to deal with multiple concurrent accesses to the same medium by different devices and some randomness is added to minimize collisions. However Ethernet can be used as a point-to-point connection between a client (the RTC Computing Module) and a switch. This is an isolated 2 peer full-duplex communication link that does not suffer from any collision, therefore can perform in real-time. It is then the characteristics of the client and the switch to actually turn a carefully designed Ethernet network into a real-time deterministic network. For this purpose we can use our array of FPGAs. All clients participating in the deterministic network will have an FPGA to manage the communication and to ensure the lowest possible latency. One topic of study is certainly stream processing using Ethernet/UDP packets[7].

3.3 Algorithms[8]

The Adaptive Optics Real Time Computer for the E-ELT instruments has to perform several intensive computing tasks that can be grouped in to two families: low-latency and high-latency tasks. The high-latency tasks have to deal with the high data throughput but, in general there is no requirement to complete the computation in a short time. Those are tasks that can be generally be implemented in standard high-end servers and sometimes the tasks can be tuned to the available computing power by means of sub-sampling the real-time data stream. Typical high-latency tasks are calibration functions, performance estimation, system monitoring and so on. At the present time these tasks are not of concern.

Low-latency tasks are the core of the functionality of the Real-Time Computer, the wave-front control: the RTC must acquire sensor data and compute mirror positions in the shortest possible time. The most time-consuming aspect of the low-latency real time control is the wave-front reconstruction, or the projection of the wave-front residual measurements to the mirror space or equivalent (an intermediate modal space). The traditional solutions to wave-front reconstruction have been to use a direct solution for $Ax=b$ where b is the vector of wave-front gradients (measurements, so known) and x is the vector of wave-front phase (unknown). A general solution when A is not invertible is the standard least square estimate, that can also be written through the Moore-Penrose pseudoinverse $x=A^{+}b$. Although A is generally sparse, the matrix A^{+} is not necessarily sparse and so the application of the direct solution has computational cost $O(n^2)$.

Many initiatives are active in the AO community to try to reduce the complexity of the solution of the reconstruction by exploiting the sparsity of the matrix or the prior knowledge of the statistics of the solution or both.

Two big families of methods have been developed or are under study: iterative and direct methods.

Most iterative methods studies so far are variations of the well known Conjugate Gradient with a pre-conditioner to improve the convergence speed. The Conjugate Gradient method is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive-definite. As an iterative method, the CG constructs at each step a search direction built out of the current residue and orthogonal to all previous search directions, in a way similar to the Gram-Schmidt ortho-normalization algorithm. To make the speed more competitive,

[7] This is about the use and location of the CRC in the packets at various levels, Ethernet or MAC frame (at the end) or UDP (at the beginning).

[8] Parts of this paragraphs have been derived from related Wikipedia entries. Contains also contributions from Nazim Bharmal from the University of Durham.

an alternative formulation of the initial problem can be stated by multiplying both sides of the problem equation by the pre-conditioner C, so that $C^{-1}A$ has a smaller conditioning number and therefore the PCG method a faster convergence.
Several PCG methods are under study, among which (non-exhaustive list):

- **Fourier Domain**. For the operation of a Fourier-Domain Preconditioned Conjugate Gradient algorithm, a pre-conditioning step replaces one part of the conjugate gradient algorithm. This replacement involves a transform to/from the frequency domain via a Fourier transform. This, at best, improves the overall FD-PCG speed to $O(n\,\log n)$.
- **Fractal Iterative Method (FRIM).** This is also a PCG algorithm where the pre-conditioning occurs using a set of Karhunen-Loève modes. Since the turbulence in the atmosphere has a power law relation within the inertial range ($l_0<x<L_0$), it can be described as a fractal process therefore enabling a recursive approximation of the projection on such modes. This sort of approach yield an algorithm with speed $O(n)$.
- **Multigrid (MG)**. Multigrid (MG) can also be used as a pre-conditioner. A multigrid method with an intentionally reduced tolerance can be used as an efficient preconditioner for an external iterative solver like PCG. The solution may still be obtained in $O(N)$ time as well as in the case where the multigrid method is used as a solver. Since the speed of MG is $O(n)$, the speed of MG-PCG is expected to reach this limit, and this has been demonstrated.

Other iterative algorithms exist, like:

- **Successive Over Relaxation (SOR).** The method of successive over-relaxation (SOR) is a variant of the Gauss–Seidel method for solving a linear system of equations, resulting in faster convergence. Given the fundamental problem Ax=b, A'A can be decomposed into a diagonal component D and a strictly lower and upper triangular components L and U. The system of linear equations can then be rearranged with the addition of a relaxation factor and turned into an iterative algorithm. It has been suggested that SOR can be competitive with a conjugate gradient algorithm in terms of speed, achieving a super-linear converge in some cases, but also be implemented as a $O(n)$ process with straightforward parallelization.
- **Multigrid.** Multigrid methods in numerical analysis are a group of algorithms for solving differential equations using a hierarchy of discretizations and they can be used both as solvers as well as pre-conditioners. The main idea of multigrid is to accelerate the convergence of a basic iterative method by global correction from time to time, accomplished by solving a coarse problem. This approach has the advantage over other methods that it often scales linearly with the number of discrete nodes used. That is: It can solve these problems to a given accuracy in a number of operations that is proportional to the number of unknowns. Although generally applied to partial differential equations with a geometrical background, extensions like the algebraic multigrid methods construct their hierarchy of operators directly from the system matrix, and the levels of the hierarchy are simply subsets of unknowns without any geometric interpretation. Thus, AMG methods become true black-box solvers for sparse matrices It can be combined with the wavelets method.

Iterative can lead to a significant reduction of the computational complexity at the cost of the error in the approximation. Non determinism due to the unknown number of iterations to reach a certain error threshold can be removed by establishing up-front a fixed number of iterations designed to achieve a good average performance, at the expense of reconstruction error variability from frame to frame. Clearly an iterative method used in a closed loop system can benefit from the fact that, if the loop is well behaved, the solution should not be far, in terms of number of iterations, from the previous step solution. Iterative algorithms clearly suffer from the inherent serialization of their implementation: the second iteration must wait for the first to complete, therefore wasting some unavoidable idle time like the read-out time of the detector. For this reason an iterative algorithm must be much faster than a direct method to be convenient, since most of the time spent to compute a direct solution would be spent in the idle time, only a small fraction (1/40th to 1/100th depending on the system) after that, when the iterative algorithm would run.
For this reason, it must be emphasized that iterative algorithms have very little chance to succeed for systems like EPICS (1/100 factor), given the tiny fraction of time available to process and enormous amount of data. For that, direct or single-pass methods shall be preferred in order to maximize the use of the read-out time of the detector while processing.
Amongst those, we can list (non-exhaustive list):

- **Fourier Transform**. The use of a Fourier transform to solve for a wave-front relies on the simple identification that a differential in the spatial domain transforms to a multiplication in frequency domain. Hence a division of wave-front gradients and inverse transform can lead to a (direct) wave-front solution. The fastest FT algorithm (DFT) limits the speed to $O(n\log n)$.
- **Wavelets**. The wavelet transform performs multi-resolution analysis of a given signal. The wavelet transform is often compared with the Fourier transform, in which signals are represented as a sum of sinusoids. The main difference is that wavelets are localized in both time and frequency whereas the standard Fourier transform is only localized in frequency. Wavelets often give a better signal representation using multiresolution analysis, with balanced resolution at any time and frequency. The discrete wavelet transform is also less computationally complex, taking O(N) time as compared to O(N log N) for the fast Fourier transform.

Other direct methods are under study. In particular Austria, as part of the accession to ESO, has offered to work on novel approaches on the reconstruction by using the expertise of mathematical institutes in Linz. This project is expected to add a new point of view to the AO control problem and therefore the list of algorithms might become longer. The Austrian team is targeting different architectures to cover all the instrument concepts given in Table 1.

4. SPARTA2 CONCEPT

The concept of SPARTA2 reuses the general architecture of SPARTA, its design principles and several components improved in the areas described in the previous sections. As for SPARTA, also SPARTA2 is divided in three main components: the real time box, the co-processing cluster and the instrument workstation.

4.1 Instrument Workstation

The SPARTA2 Instrument Workstation is an ESO-standard Instrument Workstation[9] hosting the processes which implement the external interface to the SPARTA2 RTC, provides coordination with other software operating in the E-ELT, e.g. scientific instruments and TCS, and hosts the user interfaces for the SPARTA2 RTC. This computer will host display tools to show the status of the system as well as the configuration of the wave-front sensors and corrective optics. Given the size of the related devices, advanced techniques and dedicated hardware (OpenGL and advanced video cards) might be used. This component at the moment is not cause of concerns and seems technologically feasible.

4.2 Co-processing Cluster or Supervisor

SPARTA2 will reuse SPARTA software as much as possible. The SPARTA supervisor uses CORBA as command/reply middleware and DDS as real-time and non-real-time data distribution middleware. The future E-ELT standard software will provide both mechanisms and at least DDS will be present. If a CORBA implementation will not be available for SPARTA2, the related middleware will either be ported to the new command/reply standard or an interface will be provided at the Instrument Workstation level via gateways, in the very same way SPARTA interfaces with the VLT today. The following table lists the functions of the SPARTA middleware and their reusability in SPARTA2.

What	SPARTA	SPARTA2	Notes
Command/Reply	CORBA	CORBA or equivalent	To be ported or interfaced if CORBA not available under E-ELT
Real-time data distribution	DDS	DDS	
Event/log system	DDS	DDS	
Alarm system	DDS	DDS	
Configuration Database	CDMS	CDMS	SPARTA specific, hosts big matrices
File format for configuration data	FITS	FITS	
File format for real time data	FITS	FITS	
Real-Time Box command protocol	SPARTA Driver	New	Could be CORBA

[9] Will follow E-ELT standards

Real-Time box data protocol	Custom DDS-friendly	DDS/RTPS	Must ensure no extra workload on RT Box, no publisher side “features”

The general SPARTA architecture with device objects modeling and controlling components of the real time box can be fully maintained. The major concern would be the size of the maps to manage, bigger than in SPARTA. The same concern is shared for the CDMS evolution. The following table highlights some of the SPARTA control objects, their size and ratio with the current SPARTA implementation versus future needs.

SPARTA Object	CDMS Object	SPARTA Size		SPARTA2 Size	
		System	Size	System	Size
ACQ	Dark	SPHERE	0.1 MB	SCAO	5.6 MB
		AOF	0.1 MB	MAORY	5.6 MB
				EPICS	0.5 MB
REC	CM	SPHERE	13.6 MB	SCAO	316 MB
		AOF	46.4 MB	MAORY	2.9 GB
				EPICS	10.0 GB

None of the previous numbers is cause of concerns since files of the size of 10GB are certainly manageable. However managing those maps in memory might create troubles, but servers equipped with large memory are available today, at a cost.

SPARTA Data Tasks can also be maintained, but they will clearly require more memory and more throughput. The 10GbE adoption will allow for 10 times more network traffic and we can assume we can equip the servers with 10 times more memory as we have today (now less than 10GB, for SPARTA2 100GB) and 10 times more computing power.

The latter might not be easily available, given the considerations developed in the previous sections. In fact the latest Intel Nehalem-EX (Xeon 7560) does not always outperform the CPU in use in current SPARTA cluster, the Xeon 5560/5670. However the new Nehalem-EX can pack more CPUs with more cores each within a single motherboard, totaling 32 hardware cores in one machine against 8 hardware cores of current Xeon generation.

Therefore the SPARTA Data Task functions are a concern. A possible solution might be the use of GPUs as co-processors in systems like the one pictured in Figure 4.

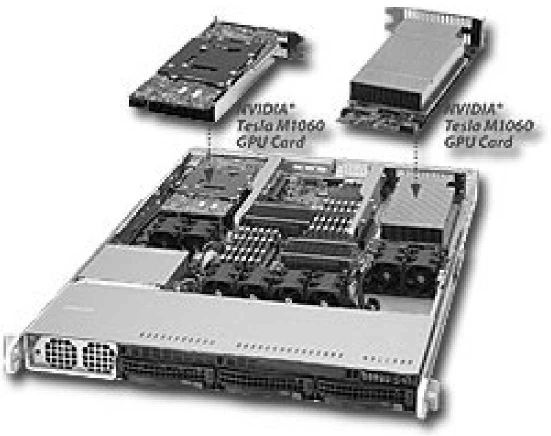

Figure 4: SuperMicro dual-Nehalem + dual-GPU server.

Clearly EPICS is a concern, given the volume of data to process, and an adequate cluster to process real-time data produced by the real-time box is, at the moment, not conceivable without significantly reducing the requirements of the cluster functions from what SPARTA performs for similar systems for the VLT like SPHERE.

4.3 Real Time box

The SPARTA2 RTC box implements the hard real-time low latency adaptive optics control loop. This loop comprises the following tasks:

- **Wavefront Processing**
  Conversion of the input pixel stream into a wavefront measurement in the form of slopes.
- **Reconstruction**
  Reconstruction of the wave-front and calculation of the delta actuator positions to be applied by means of matrix vector multiplication.
- **Control**

Control of the final actuator positions by application of a filter such as IIR or Kalman.

- **Projection**
  An optional module to project the control space vector into the DM space if they were different.

The real time box is clearly the most critical component because it is not easy to:

- Acquire large amount of data and distribute them to processing unit in an extremely short time
- Collect results and distribute them to the next stage or to the actuators in an extremely short time.
- Deliver constant and reliable performance as far as execution time is concerned (jitter)
- Pack enough computing power in a reasonable space with a real-time low-latency fabric connecting them

These properties are normally not shared by CPU-based systems. By relaxing latency and jitter requirements one could conceive a system based only on CPUs and mainstream high-end servers, but that would require 15 to 20 machines for the 3 entry-level systems (N/L GLAO, SCAO) with uncertain results in terms of latency and jitter. Certainly all the instruments above SCAO will be more risky or even out of reach, for sure EPICS.
What we propose in following sub-sections is a scalable architecture that can deliver at different degrees of performance by employing a progressively larger number of FPGA modules. As the following sections will describe, the concept of SPARTA2 is extremely flexible and scalable allowing mixing different solutions for each single module, not just for a single instrument.

RT-Box Level 0: All CPU

The real time box is either a standard high-end multi-CPU/multi-core server equipped with one or two 10GbE adapters.
All SPARTA2 RT-Box modules run in the CPU in their SW-only version, derived from the high-performance FPGA version by means of the use of a C-to-VHDL compiler for the FPGA version.
The input is taken from the 10GbE adapter and the output is sent either on the same or on a second 10 GbE adapter.
A SPARTA2 standard command and data interface is offered either through a 1GbE or a separate 10GbE adapter.
The final real time properties will heavily depend on the choice of the operating system, threading libraries, I/O architecture.

The current SPARTA real-time box code is somehow reusable. SPARTA has a CPU version of the WPU, the reconstructor and the controller as well, but all with the following limitations:

- They are designed to run on VxWorks: some porting would be needed, although extensive use of POSIX calls should be favorable to a porting to a Linux-like OS.
- Each of the modules is conceived to run on a single machine. Multi-CPU machines and multi-core machines can be easily exploited by the underlying mathematical libraries like MKL, but their design is inherently single-machine based. In order to have a multiple-machine reconstructor some modifications would be needed.
- To reach high performance, the presence of PowerPC is assumed, therefore the use of Altivec. This shall be replaced by equivalent Intel instructions (SIMD instructions are available as well on Intel) or replaced by libraries like MKL, as done in the SPARTA supervisor software. Some modifications are needed.
- The internal IPC for the CPU version of SPARTA is based on the SPARTA Light specifications which, in turn, is based on the SPARTA RTDFL, which is a TCP-based protocol based on topics. This can migrate to DDS, but some modifications are needed.

However, high performance on standard hardware does not come cheap. One should consider that the top performing hardware, a multi CPU server based on Nehalem-EX or Xeon 7500 costs about 25 kUSD and its performance is not double what the previous plain Nehalem would perform at half the cost. This is a general trend today: more performance does not come anymore at the same price.

RT-Box Level 1: Communication layer via FPGA, processing via CPU

Architecturally this option is very similar to what described in the previous section. However an FPGA would accelerate the processing of the UDP packets offloading that part from the CPU. Currently TOE engines only serve TCP/IP, as their name reveals (TCP Offload Engine). An FPGA managing the UDP communication would offer similar features as TOE gaining in latency. A further improvement in latency would come from the ability of the FPGA to interact directly with the CPU cache, should this be possible in the near future. The rest of the architecture would be identical to the previous section and subject to the same considerations.

RT-Box Level 2: Processing with accelerators

The use of the CPU in previous sections (level 0 and 1) could be replaced by accelerators. I/O will be managed by FPGA cards that would better interface with accelerators. Accelerators can be of two types:

- In-socket FPGA (or socket fillers): one stack of FPGAs replaces one or more CPU in a multi-CPU system. I/O is performed by a separate FPGA card that is directly connected to the socket filler. This in turn can do processing or communicate with the other CPUs in a fast and optimized way. Current products interface at FSB level, therefore benefitting from the faster speed of the North Bridge. However they would suffer from the same problems of the first two levels in the sense that external data must be first stored in memory and then the CPU can access to them. Future products based on QPI could be able to talk directly to the CPU cache, therefore dramatically reducing the latency. The CPU would then be used as a fast DSP only operating with data in cache.
- Future developments of PCI Express might allow direct communication of devices on the cross bar, therefore the FPGA-based card performing I/O could talk directly to a GPU that would execute dedicated code programmed by the master CPU, like MVM. That would be very efficient and probably more powerful than the previous solution.

No current SPARTA real-time box software could be re-used, certainly not with the GPU option. Some reuse might be possible for the in-socket filler, but a low degree of reuse is expected.

RT-Box Level 3: All FPGA

The real-time box is a collection of servers, each of them hosting several densely populated FGPA boards. One of them performs I/O to/from the outside world and manages the communication between the FPGA boards inside the server using dedicated communication lines. The host CPUs in the server are used to monitor and manage the FPGA boards.

The servers are connected with each other by means of 10 GbE (or faster).

The I/O card in each server receives the data to process from the network and distributes them to its local FPGAs and to the other boards hosting more FPGAs (see Figure 3). All FPGAs in all boards perform a dedicated function (like MVM) and return the result to the master I/O card for transmission.

This solution is scalable by changing the number of FPGA boards in one server and by stacking together more servers and partitioning the problem.

It is expected that the MVM part (the reconstructor in the classical approach) of all foreseen instruments except EPICS can be implemented by a single server with up to 4 boards, including 1 IFU of EAGLE.

A 1KHz SCAO system would require a server with 2 boards, a 500Hz GLAO system with sensor space averaging would require a server with only 1 board.

EPICS would require a larger system with 50 servers populated with 4 boards each. For this system, synchronization between the various servers would be the issue to solve.

However this solution is extremely scalable since it is capable of scaling up to the biggest system using the same basic module. This solution is currently under study, but it is based on technology currently available at chip level, not yet integrated in commercially available boards.

Hybrid RT-Box

All previous paragraphs described various solutions for the technology to be used for the SPARTA2 real time box. From the external interfaces, all those solutions are compatible, therefore it is conceivable to build a SPARTA2 system in which the real-time box is made by different solutions.

For instance, most likely the WPU might need to be built in a ruggedized form factor, therefore the best solution for the WPU would be an FPGA-based card on an embedded chassis like a 3U VPX. Products of this type are already available and the current SPARTA WPU can be upgraded to the size requested by the E-ELT.

The reconstructor and the controller for small scale systems like GLAO (sensor space averaged) could be implemented using a variation of a CPU-based system, likely with FPGA boards to offload UDP management.

Bigger systems could use the all-FPGA module described in the level 3 for the reconstructor, and a CPU-accelerated system as described in level 2 for the controller. A further all-FPGA module could be used for back projection into mirror space in case modal control would be used.

5. CONCLUSIONS

We have shown that the range of requirements for the conceived E-ELT instruments is very large and that waiting for newer and faster computing units, CPUs or other, in view of a “first light” rather far in the future, might be risky. We have shown that the trend of computer evolution as captured by the Moore’s law might not hold for long time and a change in the growth rate of computing units is expected in the near future. We therefore identified the need of considering more aggressive solutions in several domains, from the high performance computing units (from CPUs, to GPUs and FPGAs), to the interconnect technology (10-40-100 GbE) and as well the algorithms.

We have presented a concept for the evolution of ESO current real time platform SPARTA to the E-ELT era, that includes different levels of complexity and performance still maximizing the software reuse both from the current SPARTA and within the levels of the SPARTA2 concept.

REFERENCES

[1] Sodan, J. Machina, A. Deshmeh, K. Macnaughton, B. Esbaugh, “Parallelism via multithreaded and multicore CPUs”, IEEE Computer, March, 24-32 (2010).

[2] R. Kumar, V. Zyuban, and D.M. Tullsen, “Interconnections in Multicore Architectures: Understanding Mechanisms, Overheads, and Scaling,” *Proc. 32nd Ann. Int’l Symp. Computer Architecture* (ISCA 05), ACM Press, 408-419 (2005)

[3] G. Rousset, “EAGLE MOAO system conceptual design and related technologies”, Proc. SPIE Astronomical Instrumentation 7736-27, (2010).

[4] M. Kasper et al, “EPICS: direct imaging of exoplanets with the EELT”, Proc. SPIE Astronomical Instrumentation 7735-84, (2010).

[5] R. Davies, “MICADO: the adaptive optics imaging camera for the E-ELT”, Proc. SPIE Astronomical Instrumentation 7735-80, (2010).

[6] R. Stuik, “The METIS AO system: bringing extreme adaptive optics to the mid IR”, Proc. SPIE Astronomical Instrumentation 7736-127, (2010).

[7] E. Diolaiti, “Conceptual design of the multi-conjugate adaptive optics module for the European Extremely Large Telescope”, Proc. SPIE Astronomical Instrumentation 7736-26, (2010).

[8] T. Fusco at al, “ATLAS: the LTAO system for the E-ELT: design, performance, and sky coverage”, Proc. SPIE Astronomical Instrumentation 7736-11, (2010).

[9] M. Tallon et al., “Fractal iterative method for fast atmospheric tomography on extremely large Telescopes”, Proc. SPIE Astronomical Instrumentation 7736-32, (2010).

[10] L. Ellerbroek and C. R. Vogel, “Inverse problems in astronomical adaptive optics”, Inverse Problems, (2009).

[11] L. Gilles and B. L. Ellerbroek, “Split atmospheric tomography using laser and natural guide stars”, J. Opt. Soc. Am. A, 2427–2435 (2008).