Real-Time Wavefront Control for the PALM-3000 High Order Adaptive Optics System

Tuan N. Truong*[a], Antonin H. Bouchez[b], Richard G. Dekany[b] ,
Stephen R. Guiwits[a] , Jennifer E. Roberts[a], Mitchell Troy[a]
[a]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA 91109
[b]California Institute of Techonology, Pasadena, CA, USA 91125

ABSTRACT

We present a cost-effective scalable real-time wavefront control architecture based on off-the-shelf graphics processing units hosted in an ultra-low latency, high-bandwidth interconnect PC cluster environment composed of modules written in the component-oriented language of nesC. The architecture enables full-matrix reconstruction of the wavefront at up to 2 KHz with latency under 250 μs for the PALM-3000 adaptive optics systems, a state-of-the-art upgrade on the 5.1 meter Hale Telescope that consists of a 64x64 subaperture Shack-Hartmann wavefront sensor and a 3368 active actuator high order deformable mirror in series with a 241 active actuator tweeter DM. The architecture can easily scale up to support much larger AO systems at higher rates and lower latency.

Keywords: PALM-3000, AO, wavefront processor, GPU, CUDA, nesC

1. INTRODUCTION

PALM-3000[1] is the state-of-the-art upgrade to the successful Palomar Adaptive Optics System[1] on the 5.1 meter Hale Telescope, currently under development at JPL and Caltech. It is based on a 64x64 subaperture Shack-Hartmann wavefront sensor with 2x2 pixels per subaperture, and a new 3368 active actuator high-order Xinetics deformable mirror (DM) in series with the existing 241 active actuator "tweeter" DM. All of which are driven by a cost-effective scalable real-time wavefront control architecture based on off-the-shelf graphics processing units (GPUs) hosted in an ultra-low latency, high-bandwidth PC cluster environment composed of modules constructed in the component-oriented language of nesC[1]. This innovative architecture not only enables full-matrix reconstruction of the wavefront at up to 2 KHz with latency under 250 μs for PALM-3000, but can also scale up to support a much larger real-time AO system, such as the Thirty Meter Telescope (TMT), at higher rates and lower latency.

The remainder of the paper presents the architecture in order of importance of architectural objectives, starting with the most important. Section 2 describes a hardware architecture that fulfills our foremost objective of best price-performance. It discusses the design decisions made and the rationale behind them as we searched the solution space. Section 3 presents a software architecture that meets our secondary objective of a component-oriented architecture based on event-driven execution and split-phase operations. Section 4 presents the compute latency of the wavefront reconstruction based on the classical full matrix method. Section 5 concludes the paper.

2. HARDWARE ARCHITECTURE

Our main objective is to build an AO system that fulfills the high performance and low cost requirements of PALM-3000. Toward this end, we have examined a number of potential solutions that built around DSP-based and GPU-based architectures. Given the in-house experience with the Texas Instruments (TI) floating point DSP architecture from the successful current AO system[1], we began the evaluation with two commercially available DSP-based systems.

The first system we evaluated was based on fixed-point DSP boards from Lyrtech each featuring four 1-GHz TMS320C6416 DSPs from TI, organized in two clusters of two with the inter-cluster communication bandwidth limited

* Tuan.N.Truong@jpl.nasa.gov; phone 1 818-393-6151; fax 1 818-393-9471; Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA, USA 91109.

to 528 MB/s by the 64-bit 66MHz PCI bus connecting the clusters. Each DSP is provided with an impressive 8 GMACs of 16-bit peak processing power and 128 MB of dedicated off-chip memory with sustained memory bandwidth of 800 MB/s via a private 64-bit 100-MHz bus. This memory bandwidth unfortunately proved inadequate to compensate for the measly 1 MB of on-chip cache that equipped each DSP, given the large size(3368 x 6736) of the reconstruction matrix. As a result, the Lyrtech solution would require a relatively high number of DSPs. This is true for most, if not all, commercially available systems based on current generation of TI DSPs.

The second system we evaluated was based on floating-point DSP boards from Bittware each featuring eight 500-MHz ADSP-TS201 TigerSHARC DSPs from Analog Devices, organized in two clusters of four. Each cluster is provided with 256 MB of shared memory for intra-cluster access via a common 64-bit 83.3-MHz cluster bus at aggregate sustained memory bandwidth, or equivalently intra-cluster communication bandwidth, of 667 MB/s. Like the Lyrtech, the Bittware clusters are also interconnected via a 64-bit 66-MHz PCI bus with identical sustained inter-cluster communication bandwidth of 528 MB/s. Unlike the Lyrtech, each DSP on a Bittware board offers 3 MB of on-chip memory and 3 GFLOPs of 32-bit peak processing power. Based on the performance specifications of the optimized floating point math library from Bittware, we estimated this peak realistically to be between 1 GMACs and 1.3 GMACs, which proved in the final analysis to be the limiting factor in determining the number of DSP boards in the Bittware system. Hence, both DSP-based systems were dropped from consideration due to the high cost, which exceeded not only the hardware budget but also the cost of either GPU-based system by at least 50%.

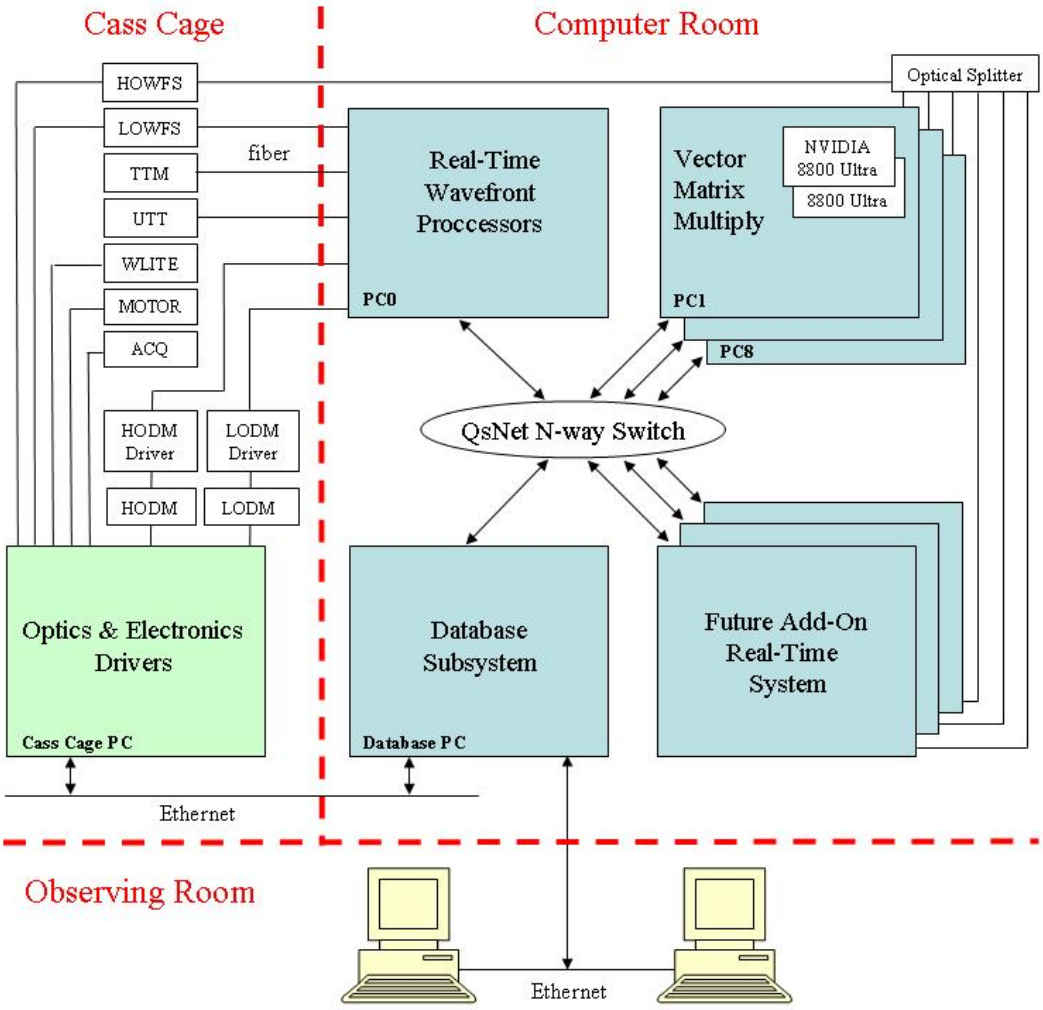


Figure 1: PALM-3000 Hardware Architecture

Figure 1 shows the PALM-3000 hardware architecture. Under this architecture, compute-intensive operations are off-loaded and accelerated by low-cost off-the-shelf GPU-based graphics cards hosted in a PC cluster interconnected with one or more ultra low-latency, high-bandwidth switches.

Consider for example the large vector-matrix multiplication (VMM) operation required for the high-order wavefront reconstruction on PALM-3000. The accelerating hardware consists of 16 retail NVIDIA 8800 Ultra graphics cards which are distributed over 8 dual-core Opteron PCs from HP, each hosting 2 cards, the maximum number of PCI Express (PCIe) x16 cards permitted by PCIe 1.1 Specification. Since our acquisition, PCIe 2.0 has been released with support for up to 4 such cards per PC.

Each NVIDIA 8800 Ultra features 576 GFLOPS on 128 612-MHz single-precision floating-point SIMD processors, arranged in 16 clusters of eight. Each cluster provides 8 K registers and 16 KB of shared on-chip memory organized in 16 banks. Accessing the shared memory is as fast as accessing a register as long as there are no bank conflicts. Also included, but not currently availed by our applications, per cluster is 64 KB of constant memory with a cache working set of 8 KB. The clusters are interconnected via a 384-bit memory bus, providing 103.7 GB/s memory bandwidth to 768 MB of global shared GDDR3 RAM.

It is this winning combination of supercomputing power and unparalleled memory bandwidth that earned NVIDIA GPUs the selection. In particular, it is the ease of programming coupled with unfettered access to the tremendous processing power of the GPU through a simple low-level C programming interface that won NVIDIA over the competing ATI GPU architecture which we also evaluated, though briefly. To aid developers in the software development, NVIDIA provides the CUDA[1] (Compute Unified Device Architecture) environment, which consists of the C cross compiler, debugger, host driver, two mathematical libraries of common usage, FFT and BLAS, and a plethora of code samples, in addition to the GPU API and its runtime.

The eight PCs along with PC0 and the Database PC are interconnected using a Quadrics QsNet^{II} 16-port switch with extra free ports reserved for future add-on systems. The switch delivers over 900 MB/s of user space to user space bandwidth each direction with latency under 2 μs for a total of 14.4 GB/s of bisectional bandwidth and broadcast capability. This scalable switch combined with the 1-to-N optical splitter that delivers the same complete frames to each of the N output ports permit not only future integration but also concurrent executions of additional real-time systems and accelerators.

As shown, high-bandwidth data, such as pixels received by the eight PCs from high-order and low-order wavefront sensors (HOWFS and LOWFS), and latency-sensitive commands, such as those issued by PC0 to high-order and low-order deformable mirrors (HODM and LODM), tip/tilt mirror (TTM) and uplink tip/tilt mirror (UTT), are transferred using fiber optics. Low-bandwidth data, such as acquisition camera (ACQ) pixels and hardware status, are sent from the Cass Cage PC to the Database PC via a dedicated 1Gbit Ethernet. Latency-tolerant commands, such as those issued by the Cass Cage PC to motors and white light, including ACQ and DM configurations, are sent via direct connections.

3. SOFTWARE ARCHITECTURE

This section presents the software architecture of the PALM-3000 system. It describes the decomposition of the software using three different architectural views each depicting different aspect of the system. The architecture is built upon many structuring concepts and execution model of the TinyOS[1] operating system, namely, a component-based architecture, structured event-driven execution model, and split-phase operations. Like TinyOS, the PALM-3000 software system uses active messages for the communication method and is written entirely in the component-oriented language of nesC. It achieves its distributed processing through the use of bidirectional proxy components that are interposed between user components and remote provider components.

Figure 2 shows a system context diagram that sets the stage for our discussion in this section. The diagram depicts a high level overview of the system (P3K-AO) and its required interfaces to a number of external systems, which include science instruments, Telescope Control System (TCS), Laser Guide Star (LGS), Beam Transfer Optics (BTO) and the system operator a.k.a. the AO operator.

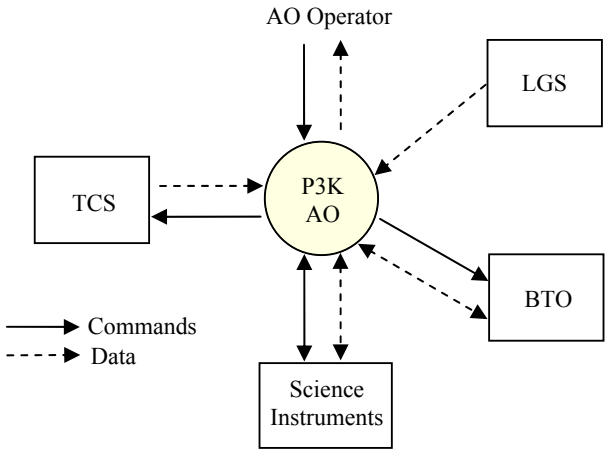


Figure 2: PALM-3000 System Context Diagram.

3.1 Functional View

The functional decomposition of the P3K-AO system is illustrated in Figure 3. As shown, the system is comprised of four major functional subsystems:

- AO Command Processor (AOCP)
- AO Database Subsystem (AODB)
- High-Order Wavefront Processor (HOWFP)
- Low-Order Wavefront Processor (LOWFP)

The solid arrows in the figure denote the flows of commands containing requests from users to providers, while the dashed arrows denote the flows of data containing results returned to users via asynchronous callbacks upon completion of the requests. Commands controlling the system are sent to AOCP, but those requesting for data are directed to AODB instead. AODB provides two methods of data delivery, *push* and *pull*. The push capability enables data to be automatically pushed to the users without polling. To achieve real-time delivery, AODB also offers the publish-subscribe capability.

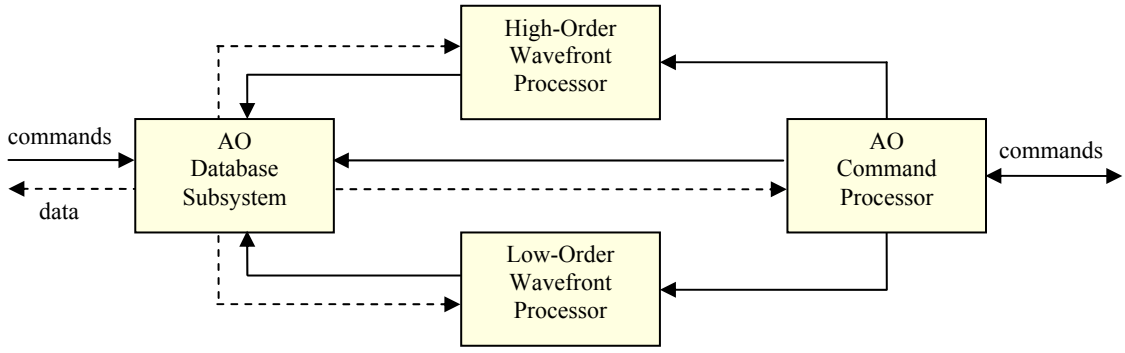


Figure 3: P3K-AO Functional Decomposition

It is important to point out that all data returned to users *originate* from AODB, not from any constituent component of AOCP, HOWFP, and LOWFP. As such, AODB provides the only avenue of access to AO data. This design decision requires basically that all processing results produced upon completion by data providers (publishers) be recorded at

AODB. To facilitate the recording of data, the publishers are provided with virtual local access to AODB through bidirectional proxy components. The AODB subsystem is built upon Berkeley DB, a fast, reliable, highly configurable open source, embeddable, non-relational database engine that stores and retrieves records consisted of key/value pairs. Each key in AODB is comprised of a data ID and a timestamp. Accurate timestamping is provided to within 1 μs resolution through the use of synchronized hardware clocks.

3.2 Process View

Figure 4 depicts the decomposition of the P3K-AO software system in terms of processes. We begin by describing the decomposition of the AODB subsystem. At the center of the diagram is the AO Data Recorder (AODR) process receiving real-time data from publishers and saving them to a backing store managed by Berkeley DB. As shown, the publishers include the HOWFP, VMM, LOWFP, AOCP and AODD processes as well as hardware components. Once the writing is completed, AODR immediately notifies the AO Data Server (AODS) of the event. Acted as a data distributor, AODS responds to such event by *pushing* the published data to subscribers, which include AOCP, the AO operator, or any user that requests data push. This publisher-distributor-subscriber architecture offers a number of advantages. It allows addition of new subscribers to the real-time control closed loop, while relieving publishers of the unnecessary burden of distributing data. Also, by giving AODS direct access to the entire Berkeley DB database as shown in the diagram, it enables subscribers to process data at their own rates without loss. User applications may concurrently access the backing store to retrieve stored data at any time by calling the Berkeley DB API directly via the pull (polled) method.

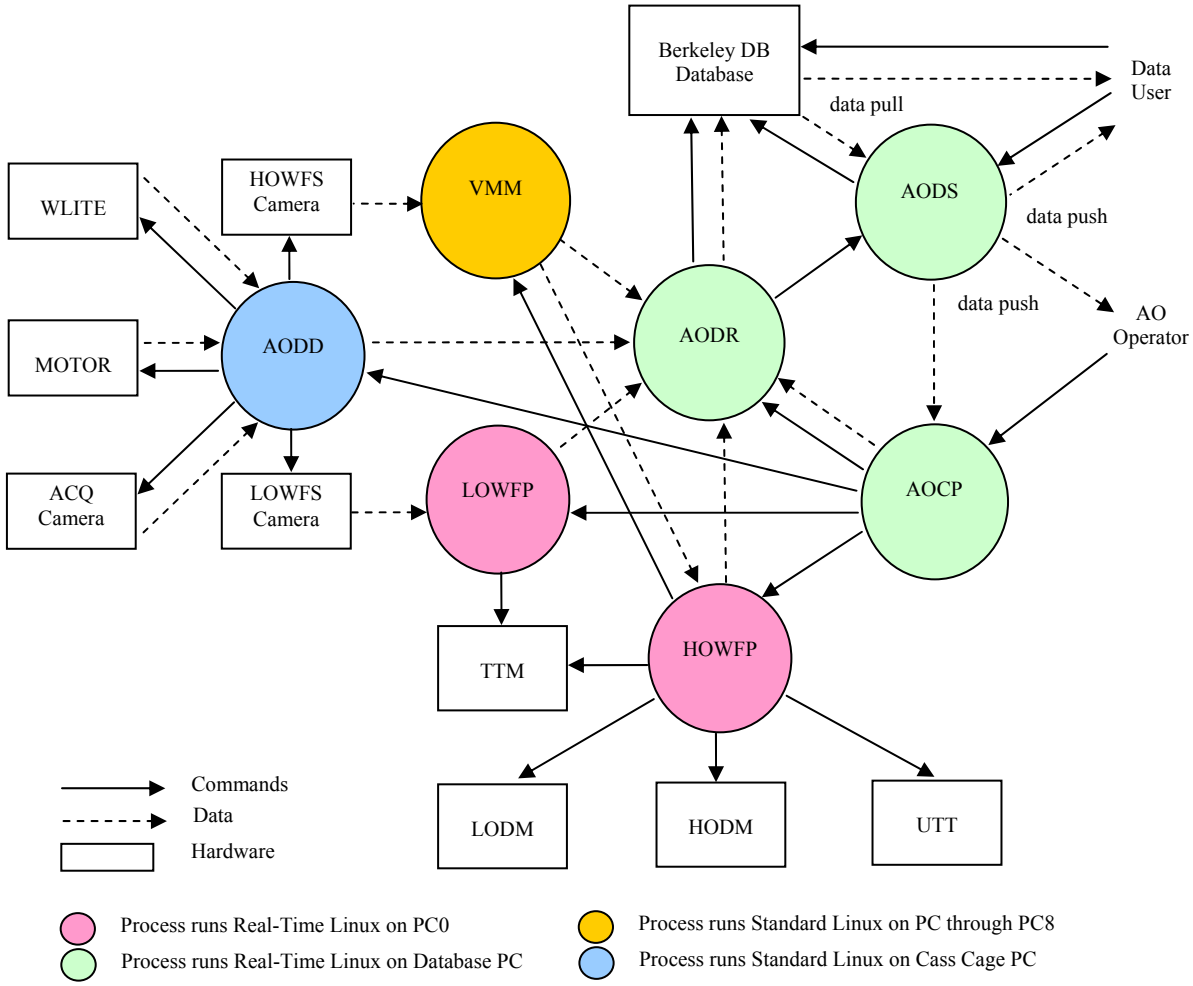


Figure 4: PALM-3000 Process View

The AO command processor is split into two processes with the main process AOCP hosted on the Database PC performing the bulk of the processing. The other "helper" process AODD is hosted on the Cass Cage PC providing virtual connections to the hardware that must reside on the Cass Cage due to hardware limitations. The splitting is necessary in order to minimize the communication latency since the Cass Cage PC cannot be directly connected to the Quadrics switch given the long distance separating them.

The high-order wavefront processor consists of nine processes, though only two are apparent from the diagram. The first eight processes execute the same code on PC1 through PC8 to help parallelize the vector-matrix multiplication (VMM) operation, while the primary process HOWFP, responsible for VMM summing, DM command calculation and interfacing to hardware and AOCP, executes on PC0.

The low-order wavefront processor consists of only one process, named LOWFP in figure 5, and is computationally much less intensive, due to the sufficiently small matrix size required for the low order reconstruction. The LOWFP performs its VMM using the second dual-core processor. To ensure HOWFP has precedence over LOWFP and both meet the real-time requirements, the host PC0 runs a real-time operating system.

3.3 Physical View

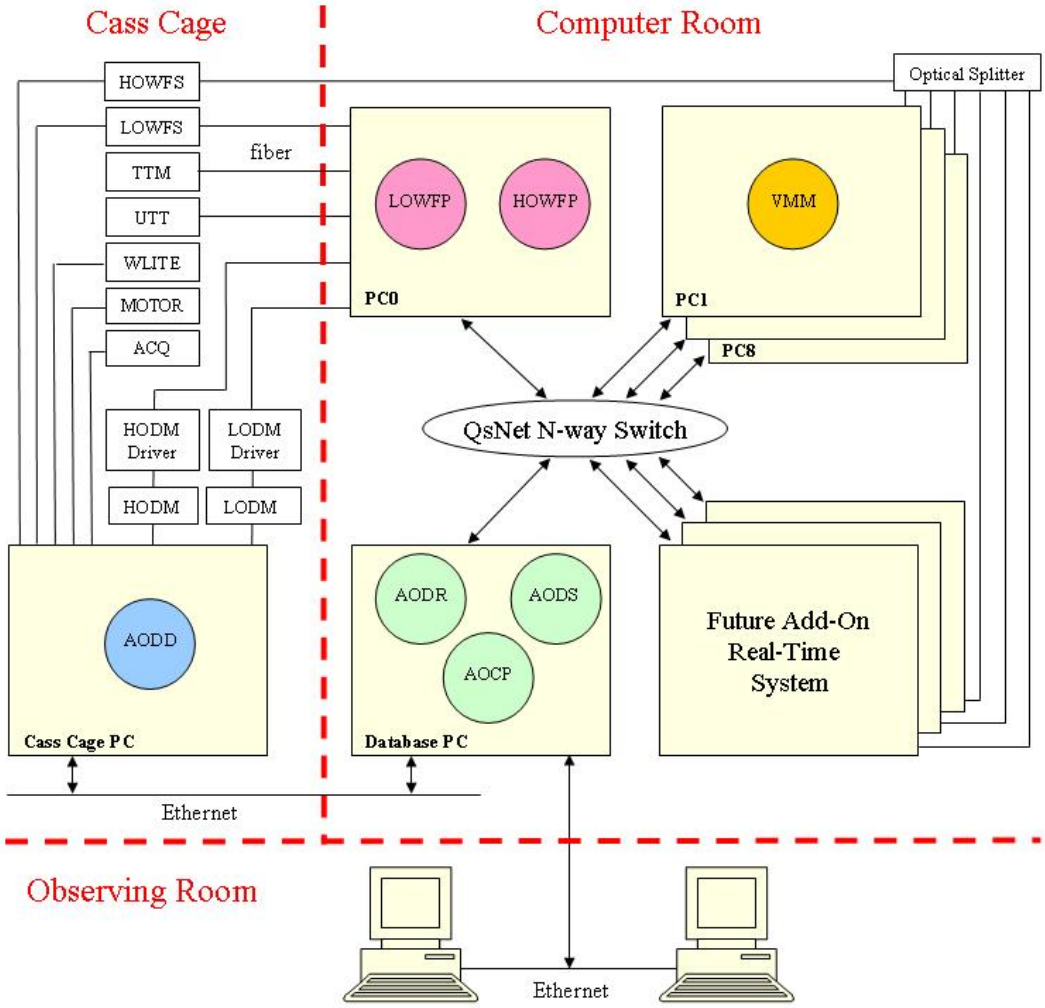
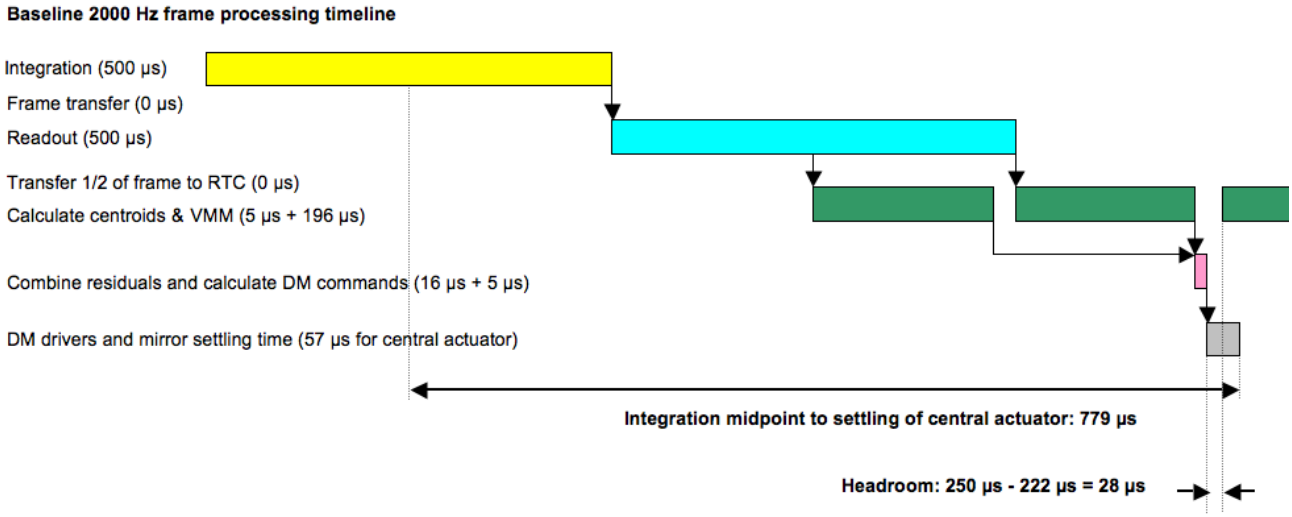


Figure 5: PALM-3000 Physical View

Figure 5 depicts the mapping of processes onto the underlying hardware, reflecting the system's distributed aspects. As mentioned earlier, AODR receives not only high-volume latency-sensitive data published by HOWFP and LOWFP on PC0 via a high-throughput, ultra-low latency, 16-port interconnect provided by the Quadrics switch, but also low-volume, latency-tolerant optics and electronics data as published by AODD on the Cass Cage PC. Once received and saved to the backing store, AODS pushes the stored data to interested subscribers, which include AOCP on the same machine and data users on a separate gigabit Ethernet network from the Cass Cage PC. As shown and noted earlier, the Cass Cage PC has no direct connection to the Quadrics switch due to the limitation on the cabling distance, resulting in the splitting of the AO command processor into two processes, namely AODD and AOCP.

4. PERFORMANCE

This section presents the expected compute latency of the wavefront reconstruction based on the full matrix vector multiplication method using the input matrix of 3456 actuator values by 8192 centroid values (i.e., 4096 subapertures with 2 centroid values per 2x2 pixel subaperture). Figure 6 shows the processing timing line at 2 KHz frame rate for 128x128 pixel frames. All times shown were theoretically determined from vendors' published performance, except the times to calculate centroids, VMM, and DM commands which were measured on actual hardware. As shown, at the 2000 Hz frame rate, the system can tolerate jitters of up to 28 µs.



PALM-3000 compute latency at 2000 Hz frame rate

5. CONCLUSIONS

We presented a cost-effective scalable real-time wavefront control architecture based on off-the-shelf graphics processing units hosted in an ultra-low latency, high-bandwidth interconnect PC cluster environment composed of nesC modules. We determined through benchmarking on actual hardware that the architecture can support full-matrix reconstruction of the wavefront at up to 2 KHz with latency under 250 µs. With a larger interconnect and additional GPUs, we expect the architecture will easily support a much larger AO system at higher rates and lower latency.

ACKNOWLEDGEMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the California Institute of Technology and the National Aeronautics and Space Administration. Support for evaluation hardware was generously provided by Defense University Research Instrumentation Program (DURIP).

REFERENCES

[1] Dekany, R.G., et al., “PALM-3000: visible light AO on the 5.1 meter telescope”, Proc. SPIE Vol. 6272, 2006.

[2] Troy, M., et al., “Palomar Adaptive Optics Project: Status and Performance”, Proc. SPIE Vol. 4007, p. 31-40, 2000.

[3] Gay, D., et al., “The nesC language: A holistic approach to networked embedded systems”, Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, p. 1-11, 2003.

[4] Truong, T., et al., "Real-Time Wavefront Processors for the Next Generation of Adaptive Optics System: A design and Analysis", Proc. SPIE Vol. 4839, p. 911-922, 2003.

[5] NVIDIA, “CUDA Programming Guide”, version 1.0, June 23, 2007.

[6] Levis, P., “TinyOS Programming”, http://www.tinyos.net, October 27, 2006.