EXPERIENCE PORTING TRIUMF'S 500 MEV CYCLOTRON CENTRAL CONTROL SYSTEM SOFTWARE TO INTEL'S 64-BIT ITANIUM RUNNING OPENVMS

K.S. Lee, E. Klassen, M.M. Mouat, J.J. Pon, P.J. Yogendran
TRIUMF 4004 Wesbrook Mall, Vancouver, B.C. Canada, V6T 2A3

ABSTRACT

The Central Control System (CCS) of TRIUMF's 500 MeV cyclotron runs on clustered Alphaservers using Hewlett Packard's (HP) OpenVMS. Recently HP announced their intention to standardize their new servers on Intel's 64-bit Itanium processor (I64). To test OpenVMS on Itanium at TRIUMF, an I64 based server was purchased and configured in an existing OpenVMS cluster. This new server was setup with a full development environment. Hardware issues for connecting to the cluster, to fibre channel disks, and to proprietary hardware interfaces were resolved and then software applications of the CCS were ported to the new hardware platform. In addition to porting the existing software, support was added for symmetric multiprocessing (SMP). The issues of hardware and software configuration, software porting, and performance are discussed, and the lessons learned are described.

INTRODUCTION

The TRIUMF Central Control System has seen many changes since the Cyclotron's first beam production in the 1970s. Initially the CCS used Data General SuperNova computers running a stand-alone real-time operating system called NATS. This was ported to a Digital VAX cluster environment running VMS in the early 1990s[1].

The evolution continues in the OpenVMS cluster environment. With the launching of newer, more powerful machines, VAX-only clusters became VAX-Alpha mixed clusters, then to Alpha-only clusters. HP has announced the planned retirement of the Alphaservers and its move of their 64-bit servers to the Intel 64-bit Itanium Family. Given that the Alpha was going end-of-life, we decided to explore migrating the Central Control System to the new Itanium platform.

For the migration to be successful, it has to satisfy several goals. Firstly, it has to be transparent. By this we mean that applications should have the same look and feel, independent of the hardware platform. Secondly, the migration cannot incur any loss of scheduled beam production or require any significant scheduled downtime. Lastly, it has to provide acceptable performance. Preferably, the migration path will involve incorporating an Itanium server into an existing production Alpha cluster, and then gradually moving towards Itanium-only clusters. The first step was to get an Itanium-based computer and set it up in our environment. Once the hardware and software infrastructure was in place, the task of porting the required software followed. The present situation is that the hardware and software infrastructures are in place and functioning well, and many of the applications have been ported.

ENVIRONMENT

Clustering

Before we had Itaniums or Alpha-only clusters, we ran mixed computer clusters that had both VAXes and Alphas. At that time we developed the environment to support applications for multiple hardware architectures in a transparent manner. When an application was activated or used, you could not tell whether you were on an Alpha or a VAX except perhaps

from the performance. A single source code was used for application programs. Multiple source codes, one for each hardware architecture, were required for a very small amount of low-level code that depended on the hardware IO. It was our desire to use this mixed cluster approach with only a single source code for our applications in the Alpha-Itanium clusters. The previously used techniques for directory structure and image activation were again employed.

Software

Because the Itaniums were new products, production-level code was not initially available in a number of areas but the software vendors were helpful by providing Beta releases of OpenVMS (from HP), the TCPIP software (Multinet from Process Software), and the messaging software (BEA MessageQ from BEA). Oracle is not presently available for OpenVMS on Itanium but is scheduled for release in the first quarter of 2006. Production versions of OpenVMS (version 8.2)[2] and Multinet (version 5.1) were available part way through our porting.

The X Windows is done using the normal OpenVMS techniques and is supported over TCPIP, DECnet, and LAT transports.

Hardware

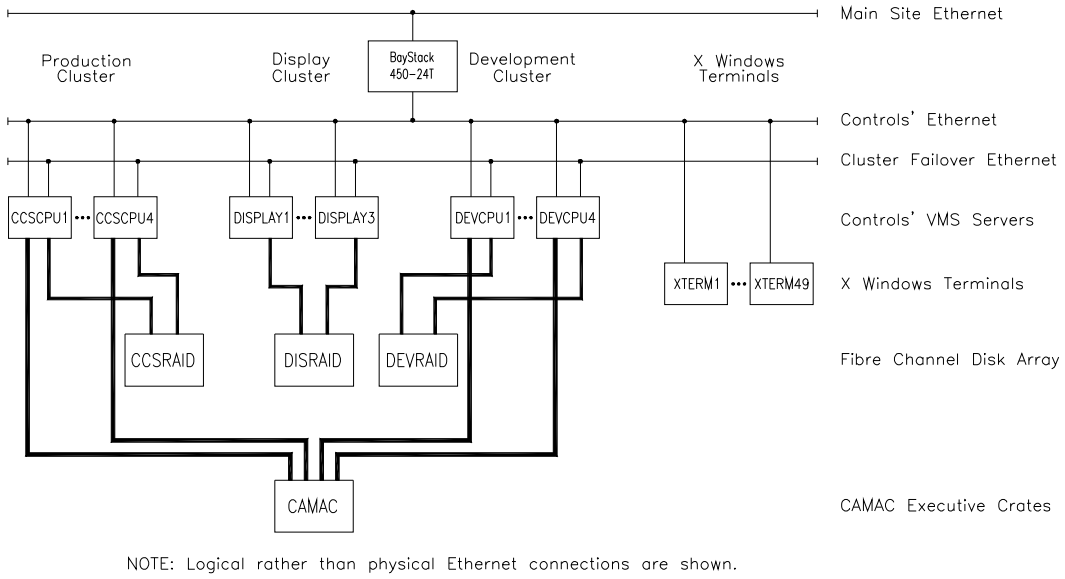

Figure 1 CCS hardware configuration

On the hardware side, an HP Integrity rx2600 server was used for the porting. It has 2 Intel Itanium processors running at 1.4GHz with 1.5 MB of cache and 4 GB of memory. OpenVMS V8.2 is installed as the operating system. The Itanium uses Intel's "EPIC" (Explicitly Parallel Instruction Computing) architecture. It relies heavily on the compilers to handle instruction scheduling and requires an increased number of registers. For example, the Itanium has 128 general registers compared to the Alpha's 32.

The rx2600 server has 4 64-bit/133 MHz PCI-X slots. The exclusive use of PCI-X slots initially caused a problem for our development. TRIUMF's PCI to CAMAC executive crate interface requires a 5 volt PCI slot and the rx2600 had only 3.3 volt slots. The solution to this issue was the purchase of a PCI-X to PCI expansion chassis from Magma[3]. Using the Magma expansion chassis, connections to the cyclotron's CAMAC systems were established.

The rx2600 was initially run standalone (not in a cluster) to minimize the impact of software related crashes during the development of the driver-level software. Eventually the rx2600 was incorporated into our existing, Alpha-only, OpenVMS Development Cluster. The rx2600 was connected to the fibre channel raid disks, to the site Ethernet, and to the Controls' Ethernet so that it had the same hardware infrastructure as the Alphaservers (figure 1). The fibre channel host bus adapter was a new model because the model available for the Alphas is not supported on Itanium. The rx2600 comes with support for two Ethernet connections, one of them with gigabit.

The console of the rx2600 is a complex interface that was found not to be user friendly.

SOFTWARE PORTING

The majority of the Central Control System's programs are written in Fortran, C and C++. Functionally, they can be divided into three categories: High-level applications, logical device support and low-level CAMAC access support.

High-Level Applications

Relatively few problems were encountered while migrating the high-level applications. Programs that compiled successfully using the latest version of compilers in Alpha generally had very little problem in compiling and linking in the Itanium[4]. The resulting object files and executable images are larger in size, typically double or triple that of the Alpha's.

One issue is the floating-point representation in the Itanium[5]. Its architecture supports IEEE floating point format only. The Alpha supports VAX F, G, and H floating-point formats and IEEE formats in hardware, but uses VAX formats by default. This means a format conversion is necessary if a floating-point data is being saved on disk using VAX format and later expected to be used in IEEE format.

One such example is the CCS data logging utility. It currently runs in an Alpha and saves the data onto the disk in VAX format. For the read-back program that runs in the Itanium, it needs to perform the conversion while reading in the data. This is achieved by setting CONVERT="VAXG' in the Fortran OPEN statement. Another option, though less efficient, is to use the /FLOAT in compiling the program. This means run-time code is used for the conversion.

Logical Device Support

For logical accelerator-device support, multiple layers of shareable images are deployed. Again, the experience has been that very few problems are encountered. A floating point format related issue is that for some run-time library routines that take a float-point argument, the type of floating point format needs to be specified. An example is the LIB$WAIT run-time routine. The default float-type is F_floating. For this routine to run properly on the Itanium using the IEEE floating-point format, the third argument (called float-type) must be set to LIB$K_IEEE_S or LIB$K_IEEE_T. Failure to do this results in an incorrect wait duration.

Low-Level CAMAC Access Support

The low-level CAMAC access support consists of two modules: a user-written system service and a CAMAC device driver. Together they handle activities to the multiple executive crate CAMAC system.

The user-written system service (UWSS) is basically a privileged shareable image. It provides standard IEEE calls such as CDREG and CFSA as well as lower level hardware-register access to the CAMAC interface. Its porting posed the biggest challenge for the software migration to Itanium**.** A similar task was undertaken when this code was ported from OpenVMS 32-bit VAX assembler code to OpenVMS 64-bit Alpha C code. Porting the Alpha C code to Itanium C code was considerably less work.

The Alpha version of the UWSS uses the IO address mapping approach. IO registers are mapped into memory space and register access is treated as a memory cycle. Since the PCI device control registers are assigned to the swizzle space of the PCI memory, their locations are platform dependent. This version also only supports the uni-processor environment. To prevent other processes interrupting during a CAMAC cycle, the priority level of the executing process is raised to the highest level (31) during the required part of the CAMAC cycle.

For the Itanium, the UWSS was modified to use the platform independent IOC$READ_IO and IOC$WRITE_IO calls for the PCI access. To support the symmetric multiple processor (SMP) environment, merely raising interrupt priority level (IPL) on the local processor to block other activities is not enough, so a device lock is used to synchronize device access for multiple processors. This is done by calling the DEVICE_LOCK and DEVICE_UNLOCK system calls with the appropriate device lock ID. This dynamic spinlock has to be allocated from the nonpaged pool memory when the UWSS was first installed.

Since the CAMAC cycle is executing under kernel mode in elevated IPL, in this case IPL 21, page faulting is not allowed during that time. Together with the fact that the UWSS is installed as writable for gathering run-time diagnostic data, locking the whole image, even though much simpler, is not an option. Care is taken to ensure that no data is required to be paged in during the CAMAC cycle. During performance testing, the new version of the UWSS is found to be 50% fast than the current UWSS version for executing the CFSA call.

The second component is a CAMAC device driver, which handles the incoming CAMAC interrupts generated by the CAMAC modules[6]. The TRIUMF built PCI-CAMAC hardware interface is auto-configured during boot time with information stored in the file SYS$SYSTEM:SYS$USER_CONFIG.DAT. The device driver's porting went quite smoothly, only the device name needed to be modified. In the Alpha, the name is CMA0. In the Itanium the device name CMxx was already taken, so the new name JCA0 was chosen.

Third Party Software

The Central Control System uses some third party software products, namely, Oracle for handling the device database, BEA MessageQ for inter-computer, inter-process communication and MultiNet for the TCP/IP environment. As mentioned earlier, a beta release of BEA MessageQ and a production release of Multinet are available. So far, no problems have been seen with software that uses these products. A recompile and relink of our applications has run normally. When Oracle for OpenVMS Itanium is available it will be tested.

PERFORMANCE

We are currently satisfied with the performance that the Alphas provide in running our applications. Clearly, faster performance is better but a migration to Itaniums should provide at least similar performance to the Alphas. Overall performance comprises many aspects including components such as IO times to accelerator devices, disk access times, internal processing times, the operating system's multitasking overhead times, ethernet performance, etc. We operate in an environment where the two primary Alphaservers typically run more than 200 processes each and these processes do a mixture of accelerator device access IO, calculation, display IO, and disk IO. Accelerator device access performance is an important issue. Device access, in our case to CAMAC, can be viewed from the perspective of having two components, the hardware IO time, and the internal CPU processing time. The hardware IO time is determined in part by the external hardware, and thus partially independent of processor type. The internal CPU processing time is strictly processor dependent. We do device access at two levels, a faster low-level access (CFSA) which is specified by the

geographic address, and a higher object oriented level which is specified by the device and its channel type. The higher-level call uses the lower level call.

CAMAC hardware cycle timing data has been collected in the Itanium server and an Alpha server. The Alpha is a DS10L with a clock rate of 617 MHz and is similar to the DS10s that we use in our Production Cluster. The results are shown in Table 1 and 2. As expected, the PCI expansion chassis has added some overhead for each CAMAC access as shown. The CAMAC cycle hardware time for the rx2600 Itanium server was found to be slower than that of the Alpha DS10L but is still considered acceptable. We expect the Itanium cycle time to improve when a new interface, which is under development, is completed. The new interface takes advantage of the Itanium's 64-bit PCI-X IO bus.

Server Type	**Read**	**Write**	**Test**	**Software Timeout**	**LAM Handling**
rx2600 (Itanium) with PCI Expansion Chassis	17.1	18.5	9.5	552	74.6
DS10L (Alpha) with PCI Expansion Chassis	13.9	11.8	6.4	549	52.2
DS10L without PCI Expansion Chassis	9.0	5.1	4.7	337	44.0

Table 1: CAMAC cycle hardware timing (units in microseconds)

Server Type	**Read**	**Write**	**Test**
rx2600 (Itanium) with PCI Expansion Chassis	27.0	30.6	21.7
DS10L (Alpha) with PCI Expansion Chassis	22.5	22.6	17.3
DS10L without PCI Expansion Chassis	16.1	12.7	14.4

Table 2: CFSA call timing (units in microseconds)

To explore the performance issues of using symmetric multiprocessor systems, throughput measurements in the Itanium and Alpha servers were carried out and are shown in Table 3. As more CAMAC processes are made to run simultaneously, the effective CFSA READ time in the Itanium server decreases and becomes comparable to that of the DS10L. This is the result of optimization of the CAMAC IO operations with the dual processors in the rx2600 and each processor executing its own instruction stream.

Number of concurrent Camac processes	**1**	**2**	**3**	**4**	**5**
rx2600 effective CFSA timing	27.2	21.5	21.4	21.1	21.2
DS10L effective CFSA timing	23.6	23.3	23.0	23.2	23.2
rx2600 effective device access timing	36.0	26.5	26.3	26.0	25.8
DS10L effective device access timing	40.0	39.7	38.5	38.6	38.6

Table 3:Throughput measurement (units in microseconds)

Because we operate in a multiprocess environment, we are particularly concerned with the overall throughput. The dual processor Itaniums provide a higher effective throughput, that is a lower effective cycle time, than our Alphas. The Alphas would also benefit by being dual processor. For us, the important issue is that this Itanium can provide acceptable overall throughput. In general applications with activities such as disk IO, computation, and then display IO (X windows), we found the Itanium rx2600 to be between 25% and 75% faster than the DS10s. This comparison is not meant to compare Alphas and Itaniums in general but just our present situation.

FUTURE WORK

We expect that the performance of our IO can be improved by taking advantage of the Itanium's 64-bit PCI bus. Using the wider bus we can change the multiple register writes to just one write and the multiple register reads to just one read. This requires a redesign of our PCI-to-CAMAC hardware interface and that project is underway.

Oracle related applications will be tested and their performance examined when Oracle for OpenVMS Itanium becomes available.

Because the porting has proceeded well, the remaining applications will also be migrated to the Itanium platform.

SUMMARY

An rx2600 Itanium server was purchased and configured in an existing OpenVMS cluster in the TRIUMF's Central Control System. This new server was setup with a full development environment. The PCI-X IO slots in the rx2600 were not compatible with TRIUMF's PCI to CAMAC interface but an expansion chassis solved this problem.

Software porting is progressing well. A new version of CAMAC access system routines was written to take advantage of the SMP architecture of the Itanium server. The resulting CAMAC timing for a single process is found to be slower than that of the Alpha DS10L. However, this is made up by the increased CAMAC IO throughput with the dual processor capability.

Porting software from OpenVMS Alpha to OpenVMS Itanium was found to usually be just a recompile and relink. There were some issues related to the floating-point formats and the low-level device access software was significantly modified. The last issue was the rx2600's console which is not user friendly but can be used.

Overall the porting has gone well and further movement in the migration will be pursued.

REFERENCES

[1] M.M. Mouat, Proc ICALEPCS-93, Berlin, Germany, Proc. 107-111.
[2] HP OpenVMS Version 8.2 Upgrade and Installation Manual, HP, Jan 2005.
[3] PCI Expansion System User's Manual- 4 Slot Series, Magma, 2003.
[4] Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers, HP, Jan 2005.
[5] OpenVMS floating-point arithmetic on the Intel Itanium architecture, HP, Technical White Paper, http://h71028.www7.hp.com/ERC/downloads/i64-floating-pt-wp.pdf , May 2003.
[6] K.S. Lee Proc. ICALEPCS –97, Beijing, China, Proc. 237-239