APPLICATION OF A VIRTUALIZATION TECHNOLOGY TO VME CONTROLLERS

T. Masuda[#], T. Ohata, JASRI/SPring-8, Hyogo 679-5198, Japan
T. Fukui, RIKEN SPring-8 Joint-Project for XFEL, Hyogo 679-5148, Japan

Abstract

In the SPring-8 control framework MADOCA, a VME controller employs a remote procedure call (RPC) server process named equipment manager (EM) for device control. EM executes control commands from client applications one by one because it is a single-thread process. As a rapid and simple but effective approach to the concurrent execution of the EM processes, we apply the virtualization technology Solaris Containers to VME controllers. Solaris Containers consumes a small disk space (~17 MB) for a new virtual host creation, and all its virtual hosts can share and access VME I/O boards because it virtualizes operating system environment within the OS level. We do not need to modify any software including the EM framework, device drivers and client applications. The technology allows us not only to consolidate but also to logically partition the deployed VME controller. In the application to linac MCU control, we have successfully obtained good results of the increase in the throughput of control commands. This approach is expected to be one of the solutions for the concurrent execution of the EM process.

INTRODUCTION

The SPring-8 control framework MADOCA [1] employs client-server architecture based on Sun remote procedure call (RPC) for device control. An RPC server process named equipment manager (EM) is running on each VME controller operated by Solaris [2]. EM executes equipment control commands from client applications one by one because it is a single-thread process. A control command taking a long time for I/O completion in the EM process delays the next command execution. Therefore, requests for the parallel (or concurrent) execution of the EM process have increased recently.

Making the EM process multithreading is a major and popular approach. However, this is not an easy approach for us because the EM framework and API libraries for device access are not currently thread-safe.

Another approach is to run multiple EM processes on a VME controller. In this case, we have to assign a different RPC program number to each EM process because an RPC client cannot distinguish multiply running RPC servers with the same program number. It might be slightly confusing for us to manage the EM processes with many program numbers.

As a simple and effective approach to the concurrent execution of the EM process, we apply the virtualization technology Solaris Containers to VME controllers.

[#]masuda@spring8.or.jp

SOLARIS CONTAINERS

Solaris Containers is one of the virtualization technologies proposed by Sun Microsystems, Inc. and has been integrated into the Solaris 10 OS [3]. It virtualizes an operating system environment within the OS level and does not use virtual machine or hypervisor. Therefore, the overhead of virtualization is very low. A virtual environment called *a non-global zone* (or simply *a zone*) is an environment completely isolated from other zones. Each zone acts as a unique virtual host built on a single machine. The zones can be managed and controlled through *a global zone*, which is a default zone when Solaris 10 is installed.

When we apply Solaris Containers to VME controllers, it has to meet the following requirements:

- Each zone can run an EM process with the same RPC program number simultaneously for parallel (concurrent) equipment control.
- Each zone can share and access hardware devices including VME I/O boards.
- The virtualization technology does not consume large disk and memory space to create a new zone.

Concurrent EM Executions

We have confirmed concurrent EM executions on a VME controller by using Solaris Containers. We prepared Solaris 10-installed VME controller *host0* and created two non-global zones on it. They were named *host1* and *host2* as virtual hostnames. We simultaneously ran the same EM program on both zones, and called the EM processes from RPC clients, as shown in Fig. 1.

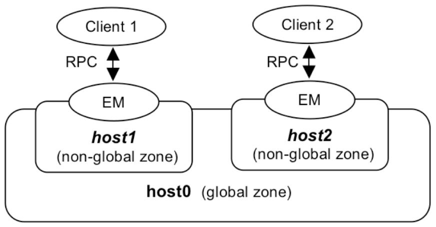


Figure 1: Example of concurrent execution of EM process.

These RPC clients succeeded in calling both EMs concurrently. This means that multiple EM processes can be run on a VME controller without changing the RPC program number. We can identify a specific EM using the virtual hostname of the zone, instead of the RPC program number. The virtual hostname simplifies the identification and management of the concurrently executed EM processes.

Sharing VME I/O Boards

We have determined that all zones can share and access VME I/O boards. We prepared a Solaris 10-installed VME controller equipped with a GP-IB board and created the character device node */dev/gpib_0* in a global zone after installing a device driver for the GP-IB board. The driver was a Solaris driver typically used in a multi process environment. We created two zones on the global zone and added a *device* property to each zone to share the GP-IB board. Consequently, both zones can possess a */dev/gpib_0* node, which is actually a metadevice of /dev/gpib_0 on the global zone. As shown in Fig. 2, we simultaneously executed EM programs with GP-IB board accesses through metadevices on both zones.

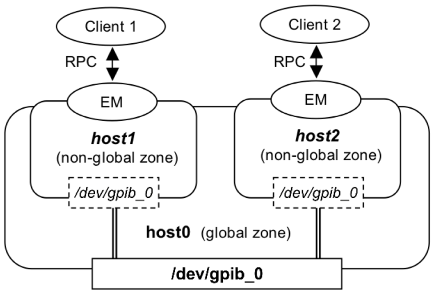


Figure 2: Sharing GP-IB board between two zones.

Both EMs successfully accessed the GP-IB board. Also, mutex control to the device driver was available. Thus, Solaris Containers allows us to share and access VME I/O boards through the device driver. We have to consider that sharing devices may violate the isolation between zones. However, this feature of Solaris Containers is very essential for applying the virtualization technology to VME controllers.

Low Resource Consumption

Generally, VME computers for device control do not have sufficient disk and memory space. Our VME CPU boards are typically equipped with 1GB Compact Flash and 256MB memory.

The installation of a new zone using *a sparse root zone* can save disk space because it shares many directories and files with a global zone. Our creation of a new zone consumes about 17MB disk space, while our installation of a global zone occupies about 420MB. Thus, the disk consumption of the new zone is small.

On the other hand, running a new zone consumes about 25MB memory space. This space is not markedly large; however, it cannot be ignored for the VME CPU board with 256MB memory. The VME controller that runs multiple zones should be equipped with at least 512MB memory.

APPLICATION TO VME CONTROLLERS

As its first applications, we have applied Solaris Containers to separate VME controllers logically and to consolidate PCs for network-based equipment control. Both applications employ universal pseudo devices (UPDs) widely used for network-based equipment in SPring-8 [4]. As a matter of course, we can apply these ideas to real devices, such as GP-IB boards and ADC boards, instead of the UPDs.

Consolidation of PCs

We have independently deployed two PCs for the control of a network-based DMM, Keithley 2701 [5]. One PC named *vacdiag6* was for temporary measurement related to the vacuum status of an 8GeV storage ring, and the other PC named *mondiagdcct* was for the diagnostics of a beam current monitor of the storage ring. A set of a UPD and a communication client (ComC) [4] for DMM control was installed into each PC, and both the EM process and the data acquisition process were run on it.

We have successfully consolidated these two PCs into one by using Solaris Containers. Two non-global zones named *vacdiag6* and *mondiagdcct* were created on a global zone, on which two UPDs were installed. The dedicated metadevice of the UPD has been created, and the dedicated ComC, EM process and data acquisition process have been run on each virtual host.

In this consolidation, we do not need to change the EM process, data acquisition program and client-side applications. The consolidation does not affect the system performance.

Logical Separation of VME Controllers

A network-connected pulse motor controller called a motor control unit (MCU) has been installed in a 1GeV linac [6]. Twenty MCUs have been deployed along with the length of the linac and controlled by three VME CPU boards named *limcu01*, *limcu02* and *limcu03*. The CPU boards have been equipped with a 1.1GHz PentiumM CPU and 256MB memory. They controlled MCUs through UPDs and ComCs. A set of a UPD and a ComC was dedicated to an MCU. We installed five sets into *limcu01*, eight sets into *limcu02* and seven sets into *limcu03*. Each VME ran one EM process and five data acquisition processes.

This approach of understanding the entire system was slightly confusing. For example, when trouble with data acquisition related to a MCU occurred, we did not know immediately which VME was in charge of the MCU.

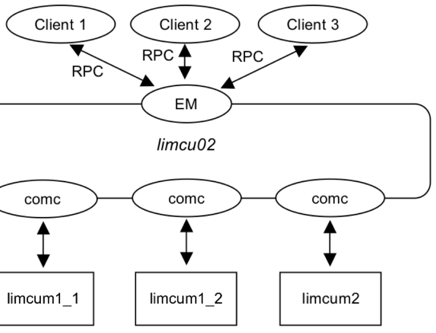
Thus, we have changed the composition of the system by using Solaris Containers. We have to increase memory size to 512MB. Solaris Containers has allowed us to logically separate the three VME controllers into twenty virtual hosts. Each virtual host is in charge of an MCU. We ran an EM process and a data acquisition process on each zone, and twenty EM processes and twenty data acquisition processes are running on the three VME controllers. The logical separations into twenty zones make the system comprehensible.

PERFORMANCE

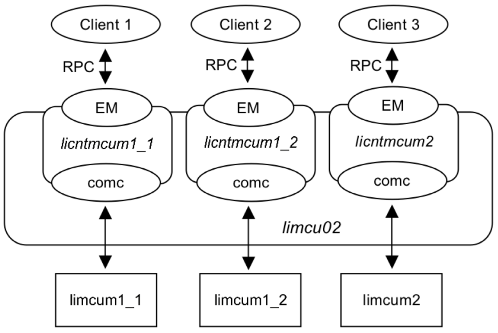
In the above-mentioned application of the linac MCU control, the EM processes are concurrently running on the VME controller. We measured the throughput of control commands sent to *limcu02*.

The measurement environment is shown in Fig. 3. Figure 3(a) shows the previous system. A single EM process that manages eight MCUs is running on *limcu02*. Figure 3(b) shows the current system on which eight zones are created. An EM process is executed on each zone, and eight EM processes are concurrently running on *limcu02*. We prepared three client applications for sending control commands continuously only to *limcum1_1* for 1024 times, only to *limcum1_2* for 1000 times and only to *limcum2* for 950 times. These client applications take about 100ms to communicate with the MCUs. We simultaneously executed these three client applications on three different operator consoles and measured the completion time.

(a)



(b)



Figures 3: Performance measurement environment of linac MCU control system. Figures 3(a) and 3(b) show the previous and current systems, respectively.

Table 1 shows the results of the performance measurements in both the previous and current systems. The concurrent execution based on Solaris Containers successfully increases the total throughput of the current system. The results indicate that the concurrent execution can increase CPU utilization. This approach is expected to be effective for increasing the efficiency of the VME controllers when faster CPU or multicore CPU boards will be used in the future.

Table 1(a) Completion time (sec) in the previous system

	client 1	**client 2**	**client 3**	**CPU utilization (idle/kernel/user)**
solo	122	119	113	48/51/1 (%)
trio	353	351	226	48/51/1 (%)

Table 1(b) Completion time (sec) in the current system

	client 1	**client 2**	**client 3**	**CPU utilization (idle/kernel/user)**
solo	122	119	113	48/51/1 (%)
trio	122	119	113	0/98/2 (%)

SUMMARY AND FUTURE PLANS

We have succeeded in the concurrent executions of the EM processes on VME controllers by using Solaris Containers. This is a rapid and simple but effective approach because we do not need to change the EM framework and RPC program number. Each virtual host can share real devices including VME I/O boards. The concurrency increases the control system throughput and enhances VME CPU utilization.

We will apply the technology to partition a consolidated VME controller into some logical hosts related to equipment groups, such as magnet power supplies, vacuum controllers and beam monitors. We will be able to consolidate VME controllers but separate them into some logical hosts.

We will test the resource management function of Solaris Containers. This will allow us to allocate hardware resources, such as CPUs and memory, to each zone. We will also change the scheduling policy of each zone.

In addition, we will examine Solaris Containers for Linux Application [7]. We may use it for Linux-based applications mainly to control network-connected devices.

REFERENCES

[1] R. Tanaka et al., "The first operation of control system at the SPring-8 storage ring", Proc. of ICALEPCS'97, Beijing, China, 1997, p. 1.
[2] http://www.sun.com/software/solaris
[3] http://www.sun.com/software/solaris/virtualization.jsp
[4] M. Ishii et al., "A Software Framework to Control a Network-Connected Equipment as a Pseudo Device", Proc. of ICALEPCS'03, Gyeongju, Korea, 2003, p. 512.
[5] http://www.keithley.com
[6] T. Masuda et al., "Upgrade of the SPring-8 Linac Control by Re-engineering the VME Systems for Maximizing Availability", Proc. of ICALEPCS'03, Gyeongju, Korea, 2003, p. 295.
[7] http://sun.com/solaris/scla.jp