REAL-TIME PROCESS CONTROL ON MULTI-CORE PROCESSORS

M. Ishii#, T. Furukawa, T. Matsumoto
JASRI/SPring-8, Hyogo, Japan

Abstract

Real-time control is essential for a low level RF and timing system to have beam stability in an accelerator operation, however, it is difficult to optimize priority control of multiple processes with real-time (RT) and time-sharing (TS) classes on a single-core processor. For example, it is not possible to log into the operating system if a real-time class process occupies the resource of a single-core processor. Recently, multi-core processors have been utilized for equipment control. We studied the process control for multiple processes running on multi-core processors. After several investigations, we confirmed that a stable operating system could run under a heavy load on multi-core processors. It is possible to achieve a real-time control response of the order of milliseconds in a fast control system such as an event synchronized data acquisition system. Additionally, we measured the response performance between client and server processes using the MADOCA II framework, the next-generation MADOCA. In this paper, we present details of the tunings required for real-time process control on multi-core processors and measurement results of MADOCA II.

INTRODUCTION

The control systems for SPring-8 and SACLA adopt the MADOCA framework [1]. These equipment controls introduce the VMEbus system based on the IA architecture Solaris 10. VME single-core CPU boards, such as the SANRITZ SVA041, are in use. In the MADOCA framework, a basic software scheme on a VME CPU board consists of an equipment control process, some processes to write polling data in memory, and a server process to send data from the memory to a database. Additionally, several fast feedback processes run in fast and complex control systems such as the LLRF system, undulator control system, and event synchronized data acquisition system. Recent control systems have a tendency to increase the number of processes running on a host.

Conventionally, programs are developed with wait-to-release CPU resources by using *sleep()* or the timeout function of *select()*. The Solaris system clock frequency can be set up to 1000 Hz with high resolution. To satisfy a control interval of less than 1 millisecond, it is an easy solution to install a busy-wait process. However a busy-wait should not be used if it is necessary to avoid 100% CPU occupation on a single-core processor. If a real-time process enters an infinite loop on a single-core processor, it becomes impossible to log into the operating system. It is difficult to optimize priority control of multiple processes with real-time and time-sharing classes on a single-core processor.

Recently, a VME multi-core CPU board has come into use for equipment control. We studied the process control of multiple processes running on multi-core processors.

#ishii@spring8.or.jp

OPERATION VERIFICATION OF MULTI-CORE PROCESSORS

We studied two models of VME multi-core CPU board: a XVB601 (GE Intelligent Platforms), featuring the Intel Core i7-620UE 1.06 GHz, and a VP717 (Concurrent Technologies), featuring the Intel Core i7-620LE 2.0 GHz. Both have dual-core processors with low power consumption, and support Intel Hyper-Threading Technology. These VME CPU boards allow four processors to appear to the host operating system, and the maximum value of CPU utilization per processor is 25%. We investigated CPU sharing and process states under high workloads on Solaris 10. The high workload test program was a simple infinite loop: *while (1)*. If this process runs on a single-core processor, the operating system hangs. We used this test program for the operation verification of multi-core processors.

Scheduling Class

By default, Solaris uses a time-sharing (TS) scheduling class, however, it also offers a real-time (RT) scheduling class. The RT scheduling class uses system priorities in a different range from the Fair Sharing Scheduler (FSS). Therefore, the FSS can coexist with the RT scheduling class within the same processor. TS class processes are controlled by the FSS. We studied the following cases.

- When a TS test program runs on four processors, the process is running on any processor. The process is not allocated a specific one. The CPU utilization of a process reaches 25%.
- When four TS test programs run on four processors, these processes are running on any processor among the four. A process is not allocated to a specific one. One among the four processes is placed in the run queue. When five TS test programs run on four processors, two of the five processes are placed in the run queue. The CPU utilization of a process is 20%. In this situation, it is possible to log into the operating system.
- When three RT test programs run on four processors, these processes are running on any processor among the four. A process is not allocated to a specific one. The CPU utilization of a process reaches 25%. In this situation, it is possible to log into the operating system. When four RT test programs run, the operating system hangs.

ISBN 978-3-95450-139-7

Processor Binding

Solaris can bind a process to a specific processor. We also studied the following cases.

- When five TS test programs and a TS test program bound to a processor run on four processors, the CPU utilization of the bound process is ~14% and the CPU utilization of the others is ~18%. Five processes share four processors.
- When five TS test programs run on four processors and a TS test program and an RT test program bound to same processor run, the CPU utilization of a bound RT process reaches 25% and the CPU utilization of a bound TS process is 0%. The other TS processes share the other three processors.

The non-binding TS processes can share all processors, but a bound TS process is allocated to only a specific processor. Therefore, under the control of FSS, a bound process is allocated a smaller CPU resource than the other processes. If high performance is required, the binding of multiple processes should be done carefully.

Processor Set

Solaris can group multiple processors as a set. We formed a processor set by grouping three processors and studied the following cases.

- When three TS test programs bound to a processor set run on four processors, the CPU utilization of a process reaches 25% and the total CPU utilization reaches 75%. When four TS test programs bound to a processor set run on four processors, the CPU utilization of a process reaches 25% and the total CPU utilization reaches 75%. One among the four processes is placed in the run queue.
- When a TS test program with three threads bound to a processor set runs on four processors, the CPU utilization of a process reaches 75%. When a TS test program with four threads bound to a processor set runs on four processors, the CPU utilization of the process reaches 75%. One of the four threads is placed in the run queue.

Features of Multi-core Processors

Our studies revealed the following features of multi-core processors.

- Even if processes numbering more than the processors are running on multi-core processors, the operating system continues to run stably.
- It is effective to bind an RT process to a processor. However, it is not effective to bind a TS process to a processor.
- A process or thread can occupy only one processor.
- The priority control of processes is extremely easy to achieve by setting a high priority process to the RT class and binding the process to a processor.

On a single-core processor, if an RT process goes out of control, the method to recovery the system is to shut the power off. However, on multi-core processors, it is possible to log into the operating system and kill the problem process and thus, the recovery of the system is faster.

INSTALLATION OF MULTI-CORE PROCESSORS

At the beam commissioning stage of SACLA, the accelerator repetition rate was 1-10 Hz, and the single-core CPU boards were installed in all VMEbus systems [2]. Currently, the accelerator repetition rate is 20 Hz, however, it will go up to 60 Hz. In the LLRF control system, two fast feedback control processes and three basic MADOCA processes are running on a single-core processor. One more process will be added into this system. It is difficult to run these processes stably on a single-core CPU board. Therefore, the VME CPU boards are currently being replaced with multi-core CPU boards.

Beam steering magnets require fast device control. We set the tick to 0.1 ms in Solaris running on the VME single-core CPU board and controlled the devices using sleep-wait. However, at the tick of 0.1 ms, Solaris became unstable. Therefore, we installed busy-wait processes running on VME multi-core CPU boards and set the tick to 1 ms. The system is now stable [3]. We concluded that a VME multi-core CPU board is suitable for fast device control.

PERFORMANCE MEASUREMENT OF MADOCA II

Next-generation MADOCA (MADOCA II) was developed in 2012 [4]. MADOCA II is based on message-oriented control, just as MADOCA was. However, MACDOCA II adopts the ZeroMQ socket [5] as the protocol for internal process communication and remote communication. MADOCA II also increases the number of processes that may be run on a device control computer, (generally a VME CPU board). In MADOCA, a single Equipment Manager (EM) ran on a host. In contract, multiple EMs for each device can be run on a host in MADOCA II.

We measured the performance of the round trip time (RTT) of message transmission in Solaris 10 on a VME CPU board, VP717. Figure 1 shows the software scheme for performance measurement. The Message Server

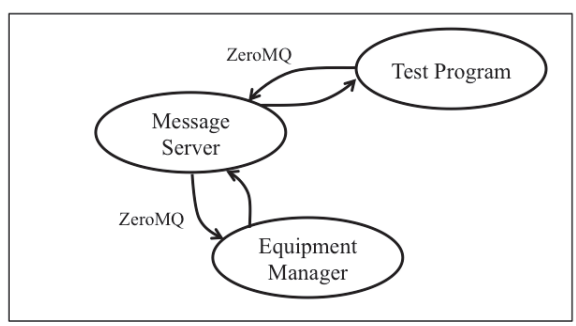

Figure 1: Software scheme for performance measurement on a VME CPU board.

ISBN 978-3-95450-139-7

(MS2) and EM are necessary components of MADOCA II. In MADOCA II, all messages go through MS2; therefore, the priority control of MS2 is important. A test program (TP) requests the "current time" to the EM via MS2 and the EM replies to the TP via MS2. The RTT is the time that the TP sends message request and receives the message reply. This software scheme for measurement is the simplest that can be run on MADOCA II. We also measured the performance in various other cases by changing a scheduling class and a processor bind of the MS2, EM, and TP. We set the tick to 1 ms in Solaris. Case 1 is the RTT when the MS2, EM, and TP are set to the RT class and the MS2 is bound to a processor. Case 2 is the RTT when the MS2, EM, and TP are set to the default TS class. Table 1 shows the statistics of the RTT in each case. Figure 2 shows the measured results for case 1 and Figure 3 shows the measured results for case 2. In case 1, the standard deviation of the RTT is extremely small, and the RTT is between 1 and 2 ms. This is good performance for real time control requiring precise time accuracy. Case 2 is not suitable for real time control. However, over 99.8% of the RTT are ~1 ms. It is acceptable for a control system such as the motor control of beamline components and data logging with the order of seconds.

Table 1: Statistics of RTT

	MS2 (RT) with binding **EM (RT)** **TP (RT)**	**MS2 (TS)** **EM (TS)** **TP (TS)**
Minimum	1.131 ms	1.029 ms
Maximum	1.76 ms	141.985 ms
Average	1.293 ms	1.141 ms
Median	1.286 ms	1.126 ms
Standard deviation	0.047	0.67

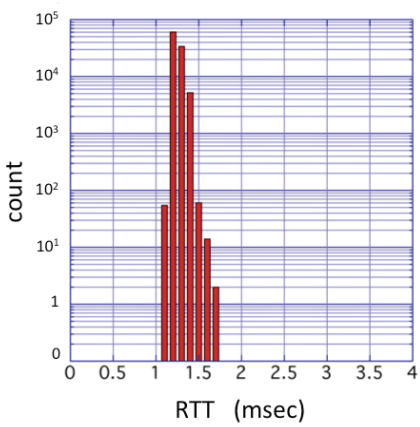

Figure 2: RTT When the MS2, EM, and TP are set to the RT class and the MS2 is bound to a processor.

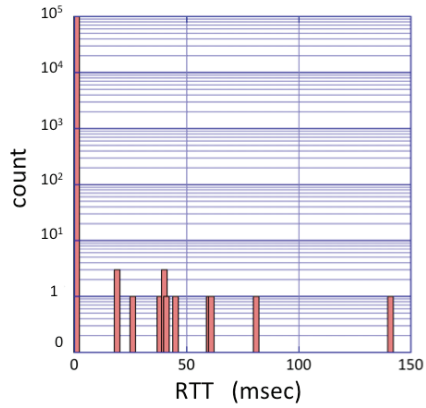

Figure 3: RTT when the MS2, EM and TP are set to the TS class.

SUMMARY

Multi-core processors are now being utilized for equipment control. We investigated the process control of multiple processes running on multi-core processors. Even if an RT process goes out of control on multi-core processors, the operating system continues to run stably. A process or thread can occupy only one processor. Additionally we measured the performance of RTT of message transmission in the MADOCA II framework running on multi-core processors. We determined that RTT is between 1 and 2 ms by the adjustment of process control. This is suitable for real time control. Multi-core processors are an essential resource for constructing real time control systems.

ACKNOWLEDGMENT

We would like to thank Dr. R. Tanaka, Dr. A. Yamashita, Mr. M. Kago, and other colleagues in the JASRI Controls and Computing Division for useful discussions regarding MADOCA-II.

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] R. Tanaka et al., "Inauguration of the XFEL facility. SACLA, in SPring-8", Proc. of ICALEPCS2011, Grenoble, France, 2011, p585.
[3] T. Otake et al., "Magnet power supply control program developed for SACLA/SPring-8", Proc. of the 8th Annual Meeting of the Particle Accelerator Society of Japan, Tsukuba, Japan, 2011, p 539.
[4] T. Matsumoto et al, "Next-Generation MADOCA for The SPring-8 Control Framework", Proc. of ICALEPCS2013, San Francisco, USA, 2013, in these proceedings.
[5] http://www.zeromq.org

ISBN 978-3-95450-139-7