# Real-time process control on multi-core processors

M. Ishii, Y. Furukawa, T. Matsumoto
JASRI/SPring-8

MOPPC128

ICALEPCS2013, San Francisco, California
October 07, 2013

## Motivation of study

The control systems for SPring-8 and SACLA adopt the MADOCA framework. 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,
- a server process to send data from the memory to a database. Additionally
- several fast feedback processes.

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.

We studied the process control of multiple processes running on multi-core processors.

## Operation verification of multi-core processors

We studied two models of VME multi-core CPU board.

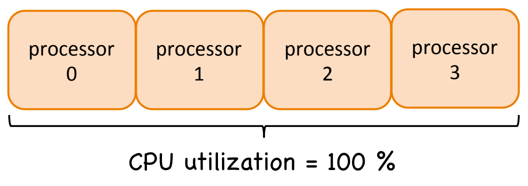
| XVB601 | VP717 |
|---|---|
| GE Intelligent Platforms | Concurrent Technologies |
| Intel Core i7-620 UE 1.06 GHz | Intel Core i7-620 LE 2.0 GHz |

- dual-core processors with low power consumption
- support Intel Hyper-Threading Technology
  - These VME CPU boards allow four processors to appear to the host operating system.

We investigated CPU sharing and process states under high workloads on Solaris 10 using this test program.

```
main() {
      while (1);
}
```

Test program

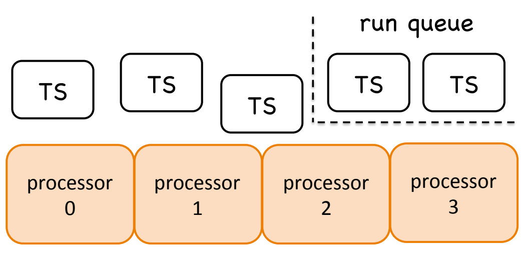


### *Scheduling class*

Solaris supports a time-sharing (TS) class and a real-time (RT) class. TS class processes and RT class processes can coexist on the same processor.

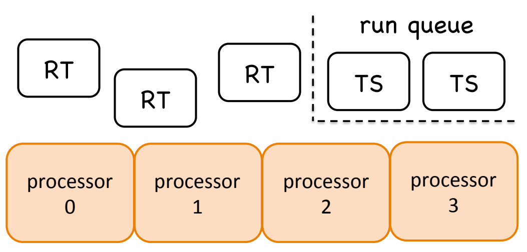
Five TS programs run on four processors.

- 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.



Three RT programs and two TS programs run on four processors.

- Two TS processes are placed in the run queue. The CPU utilization of a RT process reaches 25%, sum of two TS processes is 25%.

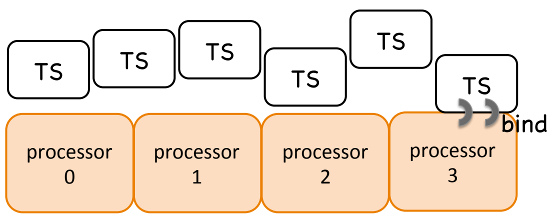


### *Processor binding*

Solaris can bind a process to a specific processor.

Five TS programs and a TS 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.

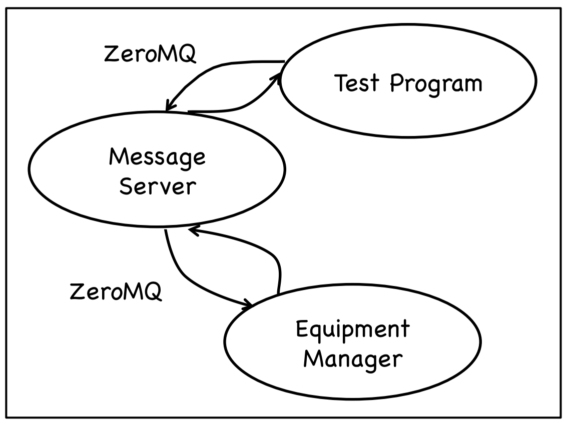


### *Features of multi-core processors*

- 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.

## Performance measurement of MADOCA II

We measured the performance of the round trip time (RTT) of message transmission on Solaris 10, and used a VME CPU board, VP717.

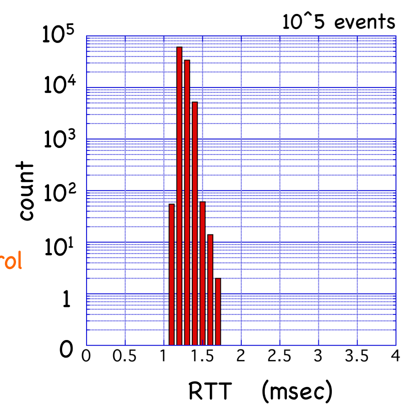


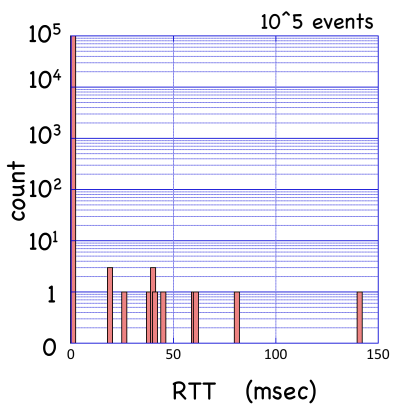
The Message Server (MS2) and EM are necessary components of MADOCA II. In MADOCA II, all messages go through MS2; therefore, the priority control of MS2 is important.

Statistics of RTT

| | MS2 (RT) with binding<br>EM (RT)<br>TP (RT) | MS2 (TS)<br>EM (TS)<br>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 |

The SD of RTT is extremely small.
1 ms < RTT < 2 ms
- Good performance for Real time control





## Summary

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 message transmission RTT 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.

ishii@spring8.or.jp