SOFT REAL TIME CONTROL WITH CLIENT/SERVER CONTROL SYSTEM

Y. Furukawa, Spring-8/JASRI, 1-1-1 Kouto, Sayo-cho, Hyogo, JAPAN.

Abstract

Real-time properties have studied for client/server control system on single CPU system with Linux and Solaris operating system (OS) with real-time scheduler. Time jitters were within one msec for Linux OS and for Solaris OS on the MADOCA control system[1] that is the SPring-8 standard control system (CPU was 1.6GHz Intel Atom processor). These results are small enough for many synchrotron radiation experiments such as x-ray diffraction experiments with continuous scanning method. The client application can be described using scripting language, so real-time applications are developed and modified easily. The system has been used in the diffuse scattering beamline at the SPring-8.

INRODUCTION

There are many request on real time controls with msec order time resolution on synchrotron radiation experiments, such as scanning micro probe XRF, continuous scanning x-ray diffraction experiments, etc. In these applications, exact timing is not required because the counting results can be normalized by each step time or integrated intensity of incident x-ray. So the sub-msec order soft real time controls are suitable for these appllications.

To realize real-time application, real time operating system (OSs) has been used, it is, however, difficult to develop the real time applications on theses OSs because it required low-level (device driver or kernel level) software development and there are poor development support tools.

Modern OSs, like Linux or Solaris, have been improved its real time properties and became to be used for real time applications. Under these OSs, soft real time can be realized only set the framework software and these applications to use real time schedulers, such as RT-class on Solaris or FIFO and round robin scheduler on Linux.

There are many single program implementations to realize the real time properties. It requires the detailed knowledge for device control libraries and frame work, it is hard task for x-ray beamline scientist because most of them are not specialist of the control software.

If real time applications can be described using simple scripting languages, many non control specialist can develop the real time applications. It is possible if the client/server type system provides real time properties. In this paper, results of the real time property measurements in the case of the MADOCA control system on the single CPU system and it has enough for the synchrotron radiation experiments.

MEASUREMENTS OF THE REAL TIME PROPERTIES

Real time property measurements were made on Solaris 10 and Linux (vanilla kernel 2.6.34 and real time patch[2] applied kernel 2.6.33.7-rt29). In the Solaris case, parameter hires_tick=1 was set in /etc/sysconfig for 1 msec tick. For the Linux case, tickless kernel and 100Hz tick were set in kernel parameters. All the software were installed on the Atom Z530 (1.6GHz) processor based control sysmte called “Blanc-4” developed at the SPring-8[3]. The blanc-4 has 512MByte main memory and 16Gbye flash memory based storage. All the softwares were set RT-class in the Solaris case (using priocntl command) or FIFO scheduling for the both Linux case (using chrt command).

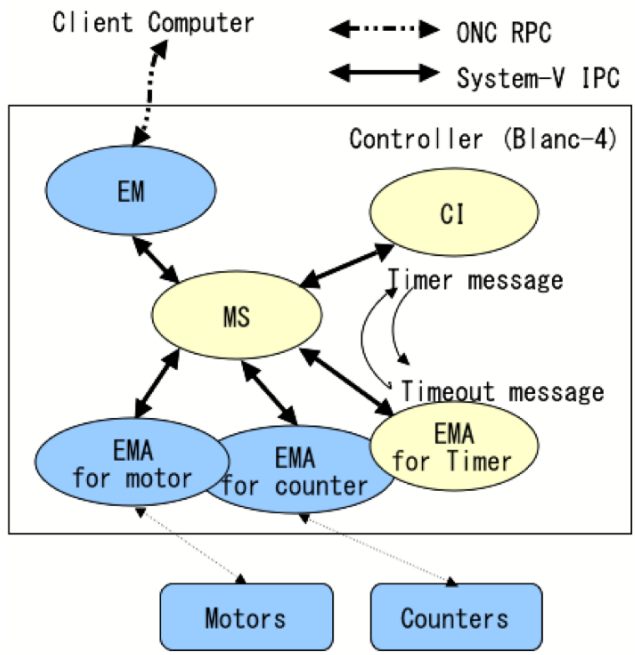

Figure 1: Software scheme of the measurements.

Software scheme based on the MADOCA control framework is shown in Fig.1. Each program communicate using system-V IPC (message queue). Command Interpreter (CI)[4], used as a client software, issued messages to the Message Server (MS). The MS transfers the control message to the Equipment Manager Agent (EMA) which controls actual devices and send back a result message to the CI via the MS. In the measurement, the EM was set as a timer, which returns a result message to the client (CI) after sleeping a given time by the message from the CI as shown in Fig. 2. The time

durations from send a message to receive the result message were measured for 1,000,000 loops.

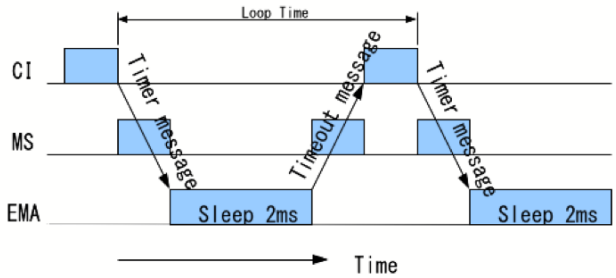

Figure 2: Time chart of measurements.

The results of time measurement are shown in Fig.3, 4 and 5 as a function of trail number for Solaris, Linux 2.6.34 and Linux 2.6.33-rt29 for first 30,000 loops. The statistics of the results were summarized in the Table 1. For the case of vanilla kernel of the Linux is not suitable for the real time applications. For the case of Solaris, time deviation is with in 0.8msec, it can be applicable some synchrotron radiation experiments. Time deviation for the Solaris 10 seems to come from SYTEM-class processes that have higher priority than RT-class processes.

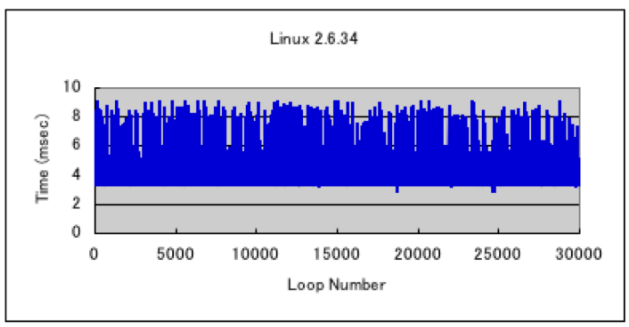

Figure 3: Result of loop time measurement for Linux-2.6.34

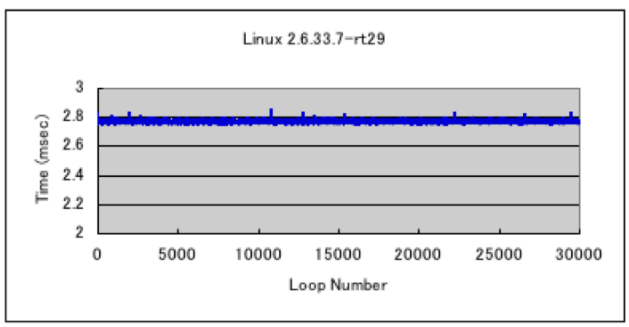

Figure 4: Result of loop time measurement for Linux-2.6.33.7-rt29

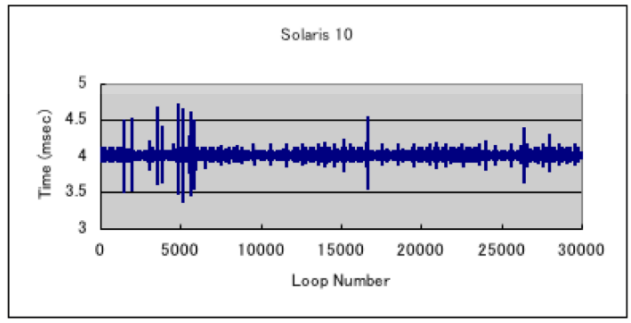

Figure 5: Result of loop time measurement for Solaris 10

Table 1: Statistics of results

OS/kernel	Meam time (msec)	Standard deviation (msec)	Min. (msec)	Max. (msec)
Linux-2.6.34	3.917	1.30	685.6	2.698
Linux-2.6.33.7-rt29	2.759	0.0091	2.731	2.969
Solaris 10	4.000	0.0175	3.349	4.417

Results for the RT-patched Linux kernel is with in 0.1msec and it is good enough for most synchrotron radiation experiments like scanning XRF, x-ray diffraction experiments. In the vanilla kernel is not pre-empt if the process is in the kernel space, while in the RT-patched kernel, the process is pre-empt in both kernel space and user space, so in the RT-patch kernel is assign the CPU time to real time process faster.

APPLICATION TO THE CONTINUOUS SCANNING X-RAY DIFFRACTION MEASUREMENT

As an application of the real-time controls, continuous scanning diffraction measurement system has been developed with Linux-2.6.33.7-rt29 system. A schematic view of the measurement system is shown in Fig.6. Diffracted X-ray by the sample is counted using x-ray detector and the detector is scanned using stepper motor. The x-ray counts are recorded as a function of the detector angle and from an analysis of the result, atomic structure is obtained.

In a conventional way, step scan was used, i.e, before counting a x-ray intensity, the detector was moved some angle. It had a dead time to waiting for end of detector motion. In continuous scanning method there is no overhead, it is, however, required msec order timing accuracy because counting duration is a few ten msec to a few seconds.

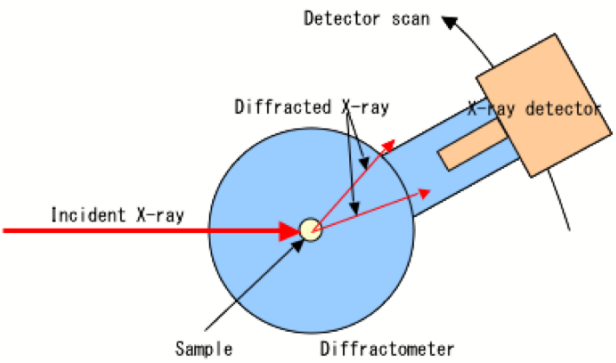

Figure 6: Schematic view of the diffration measurement.

A test of the continuous scanning was made using 1.00MHz clock instead of the x-ray counting, so the timing accuracy could be checked by uniformity of the counteing results. The result of the continuous scanning for 100msec step is shown in Fig.7. The speed of stepper motor rotation was 1000 pulse/sec. Deviation of the counting results is within 0.3%, this is good enough for most x-ray diffraction measurements. The 0.3% deviation of the counting data is corresponding to 0.3 msec timing deviation.

The counting result is not 100,000 but around 104000 counts, this measn the each loop time is 104 msec and it take 4 msec to obtaining motor position and counter data. This can be adjustable by changing the timer sleep time.

There are periodical spike on the counting data in the Fig.7. The period of the spike is about 1000 pulse, i.e. 1 sec. A motor position backing-up script was running at the same time, so the access racing to the stepper motor controller occurred. Under these racing condition to the device, timing deviation is small enough, less than required 1msec.

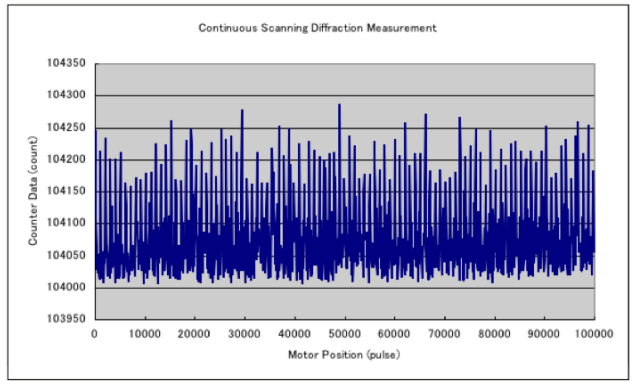

Figure 7: The result of the continuous scan for 100msec step with 1MHz input.

CONCLUSION

Real-time properties for the client/server system on Linux and Solaris OS were investigated and for Solaris 10 and RT-patched Linux case, it is shown that there are good timing accuracy. Especially for the RT-patched Linux, timing deviation is within 0.3msec.

To develop the client program, a scripting language can be used, so real-time software development becomes very easy. Note that some scripting languages invoke garbage collection and it deteriorates the real-time property. The CI is designed not to cause garbage collection.

REFERENCES

[1] R.Tanaka S. Fujiwara, T. Fukui, T. Masuda, A. Taketani, A. Yamashita, T. Wada and W. Xu, Proc of ICALEPCS'95 (1995) p.201
[2] http://www.kernel.org/pub/linux/kernel/projects/rt/
[3] M. Ishii and T. Ohata, Proc. ICALEPCS2009 (2009), p.465..
[4] Y.Furukawa, M.Ishii, T.Nakatani and T.Ohata, Proc. ICALEPCS2001 (2001), p.349