REAL-TIME PERFORMANCE MEASUREMENTS OF EPICS IOCCORE *

Shifu Xu, Martin R. Kraimer
Argonne National Laboratory, Argonne, Illinois 60439 USA

ABSTRACT

As the Experimental Physics and Industrial Control System (EPICS) is used in an increasing number of accelerator control systems, EPICS IOCcore is ported to a wider variety of OS platforms and thus the performance of EPICS IOCcore on different hardware and software platforms becomes more important. This paper provides real-time performance measurements of EPICS IOCcore on a VME hardware platform and on three different OS platforms: vxWorks, RTEMS, and Linux.

INTRODUCTION

EPICS Input/Output Controller core (IOCcore) software has been ported to several different operating systems (OSs) and many hardware platforms [1]. This paper compares the EPICS IOCcore runtime performance on one hardware platform (MVME2100 PowerPC) and three popular Operating Systems: vxWorks, RTEMS, and Linux. For Linux the following versions were tested: Linux 2.4.2 hard hat 2.0, standard Linux 2.4.30, and Linux 2.6.13. For Linux 2.6.13, the kernel was built both preemptive and non-preemptive.

Three real-time parameters are measured: interrupt, context switch, and total response latency. On Linux, more detailed interrupt latencies are measured: interrupt top half to bottom half, and interrupt bottom half to user space interrupt service routine.

To implement the tests, several software components were developed. In order to port to other operating systems or hardware platforms only, one component has to be implemented.

PARAMETERS MEASURED

Interrupt Latency

Interrupt latency is the time from interrupt generation until the interrupt service routine starts executing. Linux has an additional concept of interrupt "top half" and "bottom half". The top half interrupt handler, which executes with interrupts disabled, should complete quickly. If more work is required it can schedule a bottom half handler. On both vxWorks and RTEMS, the bottom half is replaced by a high priority thread. The VME support on Linux provides an additional feature, a user-level interrupt handler. A user-level handler is registered to receive a VME interrupt. POSIX signals are used to call the user-level handler. On Linux all of the following latencies are measured: top half, top half to bottom half, and bottom half to user-level Interrupt Service Routine (ISR).

Context Switch Latency

Context switch latency is the time from one context switching to another. The context switch time from a kernel space interrupt to user space-thread context is measured in this paper.

Total Latency

Total response latency is the time from an interrupt to when a waiting user thread begins execution.

MEASUREMENT RESULTS COMPARISON

The test IOC is heavily loaded, as described below, in all the tests. Four different values of each parameter are collected: minimum, median, maximum, and percentage of samples over some value. To make the results comparable, most tests were conducted on a private network for one hour. To look for network interference, some tests were run for 16 hours on a public network. Another test was run to measure user-level interrupt latency. Tables 1–5 show the results.

* Work supported by U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

All latencies are in microseconds. For vxWorks, two results are shown. One has the task priority lower than the vxWorks network task. The other has the task priority higher than the vxWorks network task.

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	Linux 2.4.30	9	12	72	0.000
	Linux 2.4.2hhl2.0	8	12	62	0.000
	Linux 2.6.13 non preemptive	9	13	80	0.000
	Linux 2.6.13 preemptive	8	13	71	0.000
	Linux 2.6.13 preemptive with user level ISR	15	21	77	0.000
	vxWorks 5.5 net task has higher priority	6	9	53	0.000
	vxWorks 5.5 net task has lower priority	6	8	56	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A
Public Network	Linux 2.6.13 preemptive	8	13	78	0.000
	vxWorks 5.5 net task has higher priority	6	9	60	0.000
	vxWorks 5.5 net task has lower priority	6	9	59	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A

Table 1: Interrupt Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	Linux 2.4.30	5	11	221	0.015
	Linux 2.4.2hhl2.0	5	8	218	0.001
	Linux 2.6.13 non preemptive	5	10	184	0.011
	Linux 2.6.13 preemptive	4	10	232	0.008
	Linux 2.6.13 preemptive with user level ISR	7	12	248	0.000
Public Network	Linux 2.6.13 preemptive	4	10	229	0.009

Table 2: Interrupt Top Half to Bottom Half Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	Linux 2.6.13 preemptive with user level ISR	91	99	567759	41.612

Table 3: Interrupt Bottom Half to User Level Interrupt Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	Linux 2.4.30	2	2	210	0.002
	Linux 2.4.2hhl2.0	2	3	210	0.001
	Linux 2.6.13 non preemptive	2	3	172	0.003
	Linux 2.6.13 preemptive	3	4	174	0.002
	Linux 2.6.13 preemptive with user level ISR	3	4	239	0.000
	vxWorks 5.5 net task has higher priority	5	11	503	0.013
	vxWorks 5.5 net task has lower priority	5	12	51	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A
Public Network	Linux 2.6.13 preemptive	3	4	228	0.003
	vxWorks 5.5 net task has higher priority	5	11	763	0.009
	vxWorks 5.5 net task has lower priority	5	11	56	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A

Table 4: Context Switch Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	Linux 2.4.30	21	35	1185	0.676
	Linux 2.4.2hhl2.0	23	29	250	0.030
	Linux 2.6.13 non preemptive	25	36	272	0.046
	Linux 2.6.13 preemptive	25	34	269	0.045
	Linux 2.6.13 preemptive with user level ISR	121	132	567814	100.000
	vxWorks 5.5 net task has higher priority	13	20	513	0.013
	vxWorks 5.5 net task has lower priority	13	21	80	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A
Public Network	Linux 2.6.13 preemptive	25	34	373	0.051
	vxWorks 5.5 net task has higher priority	12	20	772	0.009
	vxWorks 5.5 net task has lower priority	13	20	82	0.000
	RTEMS 4.6	N/A	N/A	N/A	N/A

Table 5: Total Response Latency

BENCHMARK PLATFORM

Hardware Platform

All tests were performed on a MVME2100 Motorola VMEbus CPU board. This is a 200-MHz CPU based on PowerPC603e core with 32 megabytes of SDRAM. The host is an x86-based PC running Fedora Core 2.6.10 Linux. The tftp server, ftp server, and DHCP daemon run on the host, and the root file system NFS of the Linux target is mounted on it. The debug port of MVME2100 is connected to the host serial port. One terminal running minicom on the host serves as the display window of the Linux target.

Operating System Evaluated

The following OSs are evaluated: vxWorks 5.5, RTEMS 4.6 snapshot, Linux 2.6.13, Linux 2.4.30, and Linux 2.4.2hhl2.0.

vxWorks 5.5 is a commercial real-time operating system. It has a high-performance real-time kernel including multitasking, preemptive priority scheduling, intertask synchronization and communication facilities, and interrupt handling support.

RTEMS 4.6 is an open-source, commercial-grade, real-time operating system [2]. It is a full-featured RTOS that supports a variety of open API and interface standards. It has the following kernel features: multitasking capabilities; event-driven, priority-based preemptive scheduling; priority inheritance; responsive interrupt management; etc.

Linux is an open source OS. The Linux 2.6 version has the preemptive kernel option for building a kernel with low latency requirements. This option reduces the latency by making all the kernel code preemptive. It also has a real-time scheduler. Linux 2.4.2hhl2.0 is MontaVista Software's Hard Hat Linux 2.0 Journeyman Edition. It has a real-time scheduler.

Platform Set-up

Cross-compiler tools [3] were built for compiling the Linux kernel and applications. Network services dhcpd, tftpd, ftpd, nfsd were installed for communication. Root file system and busybox tool [4-5] were built for the target Linux system. Efforts were made to configure all the Linux kernels in order to get better performance. EPICS base configuration files were created for the PowerPC603e platform on Linux.

MVME2100 BSP and VMEbus driver

vxWorks provides a BSP (Board Support Package) for MVME2100, but Linux does not. There is a BSP patch file that can be applied to build a Linux system for the MVME2100 under Linux 2.4.2hhl2.0 [6]. Some changes were made based on this patch file so that it could be used on other evaluated Linux platforms.

GE Fanuc provides VME bus driver for their products under Linux [7]. Some changes had to be made to the driver for the MVME2100.

TEST SOFTWARE

Generic Driver

An asynDriver [8] port driver called samplerDriver does the following:

- Implements the asynDriver interfaces asynInt32 and asynInt32Array.
- Computes all the statistics reported in the tables above.
- Communicates with the OS-dependent driver via the interface described in the next subsection.

The EPICS device support is the devEpics support provided by asynDriver.

Interface Between Generic and OS/Platform-Dependent Drivers

The interface between the generic and the OS-dependent drivers is defined as follows:

```
typedef int (*getSampleType)(void *samplerPvt,
        delayMeasure *pdelayMeasure);
typedef struct delayMeasure {
   unsigned long interruptLatency;
   unsigned long topToBottom;
   unsigned long bottomToUserInterrupt;
   unsigned long contextSwitch;
   unsigned long totalResponse;
}delayMeasure;
typedef struct samplerManager{
   int (*registerSampler)(const char *name,
```

```
        getSampleType getSample,void *samplerPvt);
    int (*removeSampler)(void *samplerPvt);
}samplerManager;
extern   samplerManager *psamplerManager;
```

samplerDriver implements sampleManager. The OS-dependent driver must implement getSample and call registerSampler. The OS dependent driver just needs to collect one sample of delayMeasure. The OS-independent driver repeatedly calls the OS-dependent driver to collect multiple samples.

OS/Platform-Dependent Driver

Several OS-dependent drivers were created for the MVME2100. For a given test only one can be loaded into the IOC.

Other OS/Platform dependent drivers could be created to test other Operating Systems and/or hardware platforms.

caput and caget

These are the Channel Access clients that put a load on the IOC. caput is started by issuing the following command: caput numberRecords pause. It runs until stopped. It implements a counter for a value to put. It continuously does the following:

1. increments the counter value
2. writes the value to each of numberRecords records
3. sleeps for pause seconds
4. repeats 1)

caget is started via the command: caget numberRecords. It sets a monitor of each of numberRecords records. It also runs until it is stopped.

Thus every time caput issues a put it causes each record to process and raise a monitor that will send the new values to caget.

Linux Kernel Module

The measurement software uses the MVME2100 Universe II chip to generate interrupts and the 64-bit time base register of the PowerPC microprocessor for counting. On vxWorks and RTEMS, there is a routine sysBusIntGen() to generate interrupt. On Linux, the VMEbus driver kernel module provides a routine vme_interrupt_generate(), which is used to generate interrupts. There are functions gettimeofday (user space) and do_gettimeofday (kernel space) on Linux, which can be used to create time stamps. On vxWorks, an assembly language routine was written to create time stamps.

A Linux kernel module was created to communicate with the VMEbus driver kernel module. It has an Interrupt Service Routine (ISR) and an Interrupt Bottom Half (IBH) routine. The ISR shares its interrupt with the Universe II chip VME interrupt. The IBH can wake up the user-level interrupt service routine through kernel signal mechanism.

TEST DATABASE AND CHANNEL ACCESS CLIENTS

IOC Database

EPICS database files sampler.db and sampler.dbd are loaded into the IOC. Sampler.dbd includes all the support (record, device, etc.) needed to run the test. Sampler.db contains five waveform records storing real-time parameter values. It also contains longin, longout, calc, and fanout records. The longin records are used for various sample values, e.g., total samples, maximum, median values of the real-time parameters, etc. The longout records are used for issuing command on the MEDM screen, e.g., start and clean button. The calc records are for calculating the percentage. A total_samples record is processed by “I/O Intr”, and a start record is periodically scanned. Many other records are processed passively by record total_samples using fanout records or forward links.

To produce a highly loaded IOC, a database file containing 5000 records is created by the program createdb.c. It is loaded into IOC by st.cmd and used by caput and camonitor.

MEDM Display

A MEDM interface displays the measurement results. Each real-time parameter is displayed by a

Cartesian plot. A menu button chooses the periodic scan time. A StartOnce button can be used to take one set of samples, and a clear button clears all the results collected. A text field sets the number of samples to take with each scan, and another field displays the total samples collected. Other fields provide more detailed information.

caput and testcamonitor

caput is started as follows: caput 5000 5 and caget as caget 5000. Thus every five seconds a burst of channel access puts are sent to the IOC. This causes 5000 records to process, raise a monitor, and send the results to caget. This puts a heavy load including network activity on the IOC. Since this takes less than five seconds, the IOC undergoes a burst of activity and then only the sampling code runs.

RESULTS ANALYSIS AND CONCLUSION

The results show that vxWorks has excellent real-time performance. Unfortunately the RTEMS latencies were not available in time for this paper.

The average performance of Linux 2.6.13 is similar to Linux 2.4.30, but the maximum total response latency is greatly reduced.

The recent Linux 2.6 kernel looks promising for use in real-time applications. It might have better performance if the real-time preemptive patch is applied. No big differences have been seen between the low-latency preemptive kernel and the non-preemptive kernel of Linux 2.6 at the time this paper was written.

Switching context involved saving the CPU's registers and loading a new state, flushing the caches, and changing the virtual memory mapping. Context switch latency is highly architecture dependent and different hardware may get different results.

If these tests were repeated or if the tests lasted for a longer time, the maximum value of each parameter could be higher.

On Linux all of the latency measurements are of interest. The application real-time requirements determine how interrupts should be handled. At one extreme, user-level interrupt handling will be sufficient. At the other extreme, the top-level interrupt handler will have to process most interrupts and only involve the bottom half and user threads at appropriate times.

REFERENCES

[1] http://www.aps.anl.gov/epics
[2] http://www.rtems.org/
[3] http://kegel.com/crosstool/
[4] http://www.busybox.net/about.html
[5] http://buildroot.uclibc.org/
[6] http://www.aps.anl.gov/asd/people/anj/mvme2100.html
[7] http://www.gefanuc.com/en/documents/vmisft-7433-3.3.tar.gz
[8] http://www.aps.anl.gov/epics/modules/soft/asyn