L4-Linux Based System As A Platform For EPICS iocCore

J. Odagiri, N. Yamamoto and T. Katoh
High Energy Research Accelerator Organization, KEK

ICALEPCS 2001, Nov 28, San Jose
Contents

- Backgrounds
- Causes of latency in Linux kernel
- Real-time Linux and EPICS iocCore
- L4-Linux as a real-time platform
- Conclusions
Backgrounds

- iocCore
  - portable to multi-platforms in EPICS 3.14
- Linux
  - promising candidate for running EPICS
- iocCore runs on Linux
  - OS Interface libraries
  - POSIX 1003.1c (Pthread)
  - POSIX 1003.1b (real-time extension)
However, ...

- POSIX real-time extensions
  - Unpredictable latency
  - Not for hard real-time applications
  - Possible rare misses to the deadline

- Causes
  - Not in the libraries but in the Linux kernel
Causes of Latency

- Non-preempt-able kernel
  - Possibly up to 100 ms or more
- Disabling of interrupts
  - Typically, several hundreds of μs
- Address Space Switching
  - Tens of μs
Non-preempt-able Kernel

Interrupt

Kernel

Latency

High Priority Process

Low Priority Process
Interrupt Disabling

unsigned long flags;

save_flags(flags);
cli();

/* critical section */

restore_flags(flags);
Address Space Switching

- x86
- Page Directory Table
- User Space
- Kernel Space
- Physical Page
- Page Table
Impacts on the Latency

- Non-preempt-able Kernel
- Interrupt Disabling
- Address Space Switching
Two Different Approaches

- Kernel-level tasks / real-time scheduler
  - RTLinux
  - RTAI
  - ...
- User-level processes / low latency Linux
  - Low latency patches
  - Preempt-able kernel
  - ...
RTLinux / RTAI

- Definitely shortest latency!
  - Several tens of µs

- Free from all of the three obstacles
  - Nothing to do with non-preempt-able Linux
  - Not allow Linux to disable interrupts
  - No address space switching required
Real-time tasks in kernel space

Each Process has Kernel Space mapping
However, …

- Real-time tasks run in Linux kernel space
- Need to put iocCore into Linux kernel space
  - Seems to be unrealistic
  - Unfavorable even if possible
- User space is much better!
Preempt-able Linux kernel

- Converts SMP spin-lock to a preemption-lock for multiple processes on a single processor
- Latency of, *typically*, around 1 ms
  - OSI libraries can benefit from the improvement
  - Linux and iocCore can evolve independently
However, …

- Some parts of the kernel code might allow a process to run too long with holding a spin-lock.
- The average performance of SMP stays even if it happens.
- It directly affects the longest latency if the spin-lock is converted to make the kernel preempt-able.
- Checking up on the whole kernel code for such problematic parts should take much effort.
Two Opposite Extremes

- Kernel-level tasks / real-time scheduler
  - Not available to iocCore
  - Hard real-time
- User-level processes / low latency Linux
  - Available to run iocCore
  - Soft real-time
The “Third Way”

- User-level processes / real-time scheduler
  - Available to run iocCore
  - Hard real-time

- Possible candidates
  - LXRT (an option of RTAI)
  - L4-Linux
What is L4-Linux?

- A port of Linux on top of a real-time micro-kernel
- Dresden Institute of Technology / IBM Watson Research Center
- Running real-time applications with a time-sharing system
Process Preempts “Linux”

Interrupt

L4

Process

Linux

Process
An Obstacle in Memory Management

Two sets of page tables for a process

“clone” allocates new page tables in L4 space

“mlockall” does not prevent page faults
Interrupt Response

- Interrupt latency measurement
  - VME CPU board (Celeron 300MHz)
  - IDE disk I/O as a background
  - A VME module as an interrupt source
  - Real-time thread serving the interrupt
Interrupt Response

Thread Priority > Linux Server Priority

Number of Events

Latency (Full Scale: 1 ms)
OSI libraries for L4-Linux

- Implemented based only on L4 system calls, basically
  - threadSuspend(), semTake(), ...
- Rely on Linux for resource allocation
  - threadCreate(), semCreate(), ...
- Channel Access related tasks rely on Linux for TCP/IP socket services
Conclusions

- Linux kernel is not for real-time application
- Real-time tasks in kernel space are not available for iocCore threads
- Processes under low latency Linux remains at a soft real-time level
- Need to dispatch user-level processes under real-time kernel control
- L4-Linux is a possible candidate