CSE 520S Course Projects

TA for class CSE 520S, Fall, 09/05/2015

Chong Li
Homework clarification

- The workload in the given example is for synthetic workload only.

- It will be better to replace each job with a real workload in the benchmarks.

- Tune the CPU frequency as well as the jobs execution time!
Connecting to Linux Lab

- In your browser, access [https://linuxlab.seas.wustl.edu](https://linuxlab.seas.wustl.edu), and log in with WUSTL Key.
- Click on ‘Submit Job’. In the next page, choose Linux Desktop.
- Select the maximum run time and resolution for the VNC window, and hit ‘Submit Job’ button.
- Your browser will download a ‘xsession.jnlp’ application to start the session.
- Run `xsession.jnlp`
  - For Mac OS user, you may get a warning saying `xsession.jnlp` can’t be opened … Then, you need to
    - System Preferences ->Security & Privacy -> click on ‘Open Anyway’
- Unfortunately, you do **not** have sudo permission.
Project Requirements

- Run on Raspberry Pi, a Linux PC, or in a cloud.

- Difficulty varies for listed candidates - will take difficulty into consideration when grading.

- Will grade based on
  - project difficulty
  - quality and depth of work
  - workload distribution among team

- Milestones: proposal, demo1, demo2, final demo, report.

- Start early! Discuss with me and Dr. Lu
Project Topics

- Real-time scheduling evaluation
  - Timing API study
  - Scheduling latency measurement
  - Worst case execution time measurement
  - Real-time scheduling theory
- Kernel hacking
  - EDF scheduler porting
- Virtualization
  - Virtualization on Raspberry Pi
  - Real-time Xen
  - Light-weight virtualization
  - Measurement study on public cloud
- Real-time applications
Timing API Study

- RDTSC is a high-resolution timer on x86 platform
  - Important when measuring real-time applications

- On Raspberry Pi, you can use
  - Gettimeofday(), clock_gettime()
  - Or find (ARM-specific) high-level APIs to read performance counters
  - Or emulate RDTSC
    - Create a loadable kernel module to access to performance counters of Rpi

- On x86 or Rpi, do performance comparison of timing APIs
  - Resolution, overhead, offset with multi-core…
Scheduling Latency – Why?

- Scheduling latency: time until the highest priority task is executed

- Scheduling latency measurement
  - Hourglass to get scheduler behavior and scheduling latency
  - Cyclic test to measure scheduling latency

“Inferring Scheduling Behavior with Hourglass”
http://www.cs.utah.edu/~regehr/hourglass/

Basic idea
- Multiple threads recording fine-grained map of when each of its threads runs

Can also be used to get scheduling latency
However

- Record using rdtsc(), which is not available on Raspberry Pi
- Evaluation based on Linux 2.4 on a PC

Minimum requirement: Fig 3,4 in the hourglass paper

- Scheduler behavior: default, FIFO, RR, deadline, etc
- Different interference: idle, CPU workload, IO workload
“Using and Understanding the Real-Time Cyclictest Benchmark”
https://www.youtube.com/watch?v=f_u4r6ehZKY
https://rt.wiki.kernel.org/index.php/Cyclictest

“A Comparison of Scheduling Latency in Linux, Preempt_RT, and LITMUS-RT”

Basic idea
- External interrupt triggers, possible delay due to
  - IRQ handling, external interrupts, preemption, scheduler overhead, etc
Scheduling Latency – CyclicTest

- **However**
  - Need to cross compile on the Raspberry Pi platform

- **Minimum requirement: fig 4,5,6 in ospert13 paper**
  - Idle system (get baseline results)
  - CPU busy task
  - Cache busy task
  - I/O bound task
  - Combination of the above three
### Real-Time Tools – WCET

- **WCET project / Benchmarks**

**Legend:**
- **I** = uses include files
- **E** = calls external library routines
- **S** = always single path program (no potential flow dependency on external variables, for a discussion, see [here](http://www.mrtc.mdh.se/projects/wcet/benchmarks.html))
- **L** = contains loops
- **N** = contains nested loops
- **A** = uses arrays and/or matrixes
- **B** = uses bit operations
- **R** = contains recursion
- **U** = contains unstructured code
- **F** = uses floating point calculation (some of the benchmarks define floating point variables that are used for time measurements, these have not been marked. These variables can be removed)
- **Bytes** = size of source code file
- **LOC** = lines of source code

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Description</th>
<th>Comments</th>
<th>I</th>
<th>E</th>
<th>S</th>
<th>L</th>
<th>N</th>
<th>A</th>
<th>B</th>
<th>R</th>
<th>U</th>
<th>F</th>
<th>Bytes</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm*</td>
<td>Adaptive pulse code modulation algorithm.</td>
<td>Completely well-structured code.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26852</td>
<td>879</td>
</tr>
<tr>
<td>bs</td>
<td>Binary search for the array of 15 integer elements.</td>
<td>Completely structured.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4248</td>
<td>114</td>
</tr>
<tr>
<td>bsort100</td>
<td>Bubblesort program.</td>
<td>Tests the basic loop constructs, integer comparisons, and simple array handling by sorting 100 integers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2779</td>
<td>128</td>
</tr>
<tr>
<td>cnt*</td>
<td>Counts non-negative numbers in a matrix.</td>
<td>Nested loops, well-structured code.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2880</td>
<td>267</td>
</tr>
<tr>
<td>compress*</td>
<td>Data compression program.</td>
<td>Adopted from SPEC95 for WCET-calculation. Only compression is done on a buffer (small one) containing totally random data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13411</td>
<td>508</td>
</tr>
<tr>
<td>cover*</td>
<td>Program for testing many paths.</td>
<td>A loop containing many switch cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5026</td>
<td>240</td>
</tr>
</tbody>
</table>
Real-Time Tools – WCET

MiBench [http://www.eecs.umich.edu/mibench/](http://www.eecs.umich.edu/mibench/)
- Provide ARM binaries

A set of 35 embedded applications divided into 6 suits
- Automotive and industrial control
- Consumer devices
- Office automation
- Networking
- Security
- Telecommunications
Real-Time Tools – WCET

- SNU real-time benchmarks
  http://www.cprover.org/goto-cc/examples/snu.html

- A set of 17 applications
  - Sqrt, qsort, crc check, insert sort, etc
Real-Time Tools – WCET

- Minimum requirement: Pick a representative subset of the applications, show execution time distribution for 1000 times
  - Idle system
  - CPU busy task
  - Cache busy task
  - I/O bound task
  - Combination

- Plot the results
  - CDF…
Real-Time Scheduling – Why?

- Scheduling latency and WCET benchmarks
  - Focus on single task/job
  - Measure wcet, scheduling overhead

- Real-time Scheduling
  - Focus on the scheduler
  - Validating real-time scheduling theory
    - Fix priority: Rate Monotonic
    - Dynamic priority: Earliest Deadline First
  - Show deadline miss ratio for the whole task set

- Question: How to decide the period/wcet/priority for each task?
Real-Time Scheduling – Case Study

- Open Repository for Real-Time Benchmarks
  http://www.cyphy.ece.mcgill.ca/Benchmark/Single%20Core.html

- Fuel injection system: 16 tasks, 46 shard data
- Task system: 46 tasks, 36 resources
- Fuel injection system: 90 function blocks, 106 communication links

Reference:
- “Optimal synthesis of communication procedures in real-time synchronous reactive models”
- “Synthesis of Multi-task Implementations of Simulink Models with Minimum Delays”
Real-time Scheduling – ARINC 653

- Defines a general-purpose application/executive software interface between the operating system of an avionics computer and the application software.
  - [http://repository.upenn.edu/cis_reports/898/](http://repository.upenn.edu/cis_reports/898/)
  - Workload detail in appendix
Real-Time Scheduling – Synthetic

- Open to other workload, as long as they are based on real-world real-time applications
  - Video decoding
    - Frame rate of 24 HZ to 60 HZ
    - Task period of 33 ms to 17 ms
  - Audio play
  - Robotics control, sensor getting data
Real-Time Scheduling – Requirement

- Select/Generate several task sets, assign each task priority according to rate monotonic priority scheme
- Perform the theory analysis
  - You will learn it later in the class
- Run the task set on raspberry pi or PC
- Compare real results with theoretical guarantees
  - Does scheduling latency/overhead matters?
  - Is theory tight for schedulability analysis?
Real-Time Scheduling – Requirement

- Other extensions
  - Priority inheritance
  - Harmonic workload (scheduling bound is 100%)
  - Aperiodic tasks
  - Default scheduler without theory guarantees
  - User level EDF scheduler
  - …

- Distribution of
  - Response time / deadline
  - Deadline miss ratio
Summary: Real-Time Scheduling

- Timing API study (optional)
- Scheduling latency measurement
  - Hourglass, CyclicTest
- Worst case execution time benchmarks
  - WCET benchmark, MiBench, SNU
- Real-time scheduling theory
  - Real-world workload example

Example proposal
- Demo 1: scheduling latency measurement (timing API study)
- Demo 2: worst case execution time measurement
- Final demo: task set schedulability analysis & experiment
Kernel Hacking – Why?

- Raspberry Pi has no EDF scheduler
  - Linux on RPi only supports fixed priority
  - Dynamic priority (EDF) provides better schedulability test
  - Existing patches provides EDF scheduler for x86 system, but not for ARM
Kernel Hacking – Litmus^RT


- Add another set of schedulers on top of FIFO and RR
  - Global EDF
  - Partitioned EDF
  - Cluster EDF
  - Pfair
Modified the task structure to support sporadic task model

```c
+struct rt_task {
    + lt_t exec_cost;
    + lt_t period;
    + lt_t relative_deadline;
    + lt_t phase;
    + unsigned int cpu;
    + unsigned int priority;
    + task_class_t cls;
    + budget_policy_t budget_policy; /* ignored by pfair */
    + release_policy_t release_policy;
};
```
Kernel Hacking – Sched_deadline

- An implementation of EDF in Linux Kernel
  http://gitorious.org/sched_deadline

- Kernel/sched_dl.c; Include/kernel/sched.h

```c
struct sched_dl_entity {
    struct rb_node       rb_node;
    int nr_cpus_allowed;

    /*
     * Original scheduling parameters. Copied here from sched_param_ex
     * during sched_setscheduler_ex(), they will remain the same until
     * the next sched_setscheduler_ex().
     */
    u64 dl_runtime;     /* maximum runtime for each instance */
    u64 dl_deadline;    /* relative deadline of each instance */
    u64 dl_period;      /* separation of two instances (period) */
    u64 dl_bw;          /* dl_runtime / dl_deadline */
```
Kernel Hacking – Expectations

- Identify the parts that changed to add a EDF scheduler

- Port it to run on Raspberry Pi

- Compare it with user-level EDF scheduler
  - “Bringing Theory Into Practice: A Userspace Library for Multicore Real-Time Scheduling”, RTAS 2013

- Perform real-time schedulability test for EDF scheduler

- This is not easy
  - You have to replace some x86 specific functions in the kernel
  - Will grade based on your effort
Virtualization – Why?

- Virtualization is widely used in clusters, cloud computing or embedded systems
  - Provides resource isolation, security, high availability
  - Introduces overhead, may result in long latency, extra resource consumption
Virtualization on ARM

- Very hot topic in virtualization now

- Xvisor: light weight open source virtualization for ARM
  - [http://xhypervisor.org/pdf/Embedded_Hypervisor_Xvisor_A_comparative_analysis.pdf](http://xhypervisor.org/pdf/Embedded_Hypervisor_Xvisor_A_comparative_analysis.pdf)

- Alternative choices
  - Xen ARM, KVM ARM
Virtualization on ARM

- Configured Xvisor (or other virt tech) to run on Raspberry Pi

- Performance evaluation compared to non-virt environment
  - Disk read/write speed
  - CPU and memory
  - Network throughput/latency
  - VMM scheduler study and real-time scheduling evaluation

- Reference:
  - [http://xhypervisor.org/pdf/Embedded_Hypervisor_Xvisor_A_comparative_analysis.pdf](http://xhypervisor.org/pdf/Embedded_Hypervisor_Xvisor_A_comparative_analysis.pdf)
  - “KVM/ARM: Experiences Building the Linux ARM Hypervisor”
  - “Performance overhead of KVM on Linux 3.9 on ARM Cortex-A15”
    - [http://www.mrtc.mdh.se/vtres2013/papers.html](http://www.mrtc.mdh.se/vtres2013/papers.html)
Real-Time Xen

- Xen: open source virtualization platform

- RT-Xen: real-time scheduling for VMs on Xen
  - Initiated and maintained by CPSL and UPenn
  - Appeared in Xen 4.5 as RTDS scheduler
  - Benchmark RTDS:
    - scheduling latency and WCET evaluation
    - (compositional) real-time scheduling analysis

- Reference
Light-weight Virtualization

- Docker: running applications in software containers
  - Providing a layer of abstraction and automation of operating-system-level virtualization
  - Light-weight compared to KVM/Xen

- Performance evaluation compared to non-virt environment
  - Disk read/write speed
  - CPU and memory
  - Network throughput/latency
  - Real-time scheduling evaluation

- Benchmarks:
  - “An Updated Performance Comparison of Virtual Machines and Linux Containers”
  - Memcached
    https://community.centminmod.com/threads/centos-6-6-memcached-1-4-22-docker-image.2348/#post-11009
Measurements on Public Cloud

- Public cloud: Amazon EC2, Windows Azure, VMware cloud …

- Performance measurement for the VMs on a public cloud
  - Disk read/write speed
  - CPU and memory
  - Network throughput/latency
  - Performance fluctuation
  - Real-time scheduling evaluation

- Benchmarks:
  - “Cloudcmp: Comparing Public Cloud Providers”
Real-Time Applications – Why?

Many interesting applications on Raspberry Pi

  - Automation, sensing and robotics
  - Gaming
  - Graphics, sound and multimedia
  - Media centers
  - Networking and servers

Latency-sensitive applications
Real-Time Applications

- Find a application that has real-time requirements

- Measure it under different scenarios
  - Different schedulers: default CFS, RT with priorities
  - Different interference workload: cpu, io, cache, etc
  - Stress test the system

- Metrics:
  - End-to-end delay, predictability…

- **Quantify** the impact
  - It is ok to find existing projects from Raspberry Pi forum
Real-Time Applications – Examples

- “Soft Real-Time on Multiprocessors: Are Analysis Based Schedulers Really Worth It?”
  https://wiki.litmus-rt.org/litmus/Publications

- Modified a video player to measure deadline miss ratios

- Plays 58 best effort video under different schedulers
Real-Time Applications – Examples

- Cloud gaming: put all computation in the server
  - Easy configuration, maintenance
  - [http://gaminganywhere.org/](http://gaminganywhere.org/)

- VoIP (Voice over IP network)
  - Asterisk

- RTSP (Real-Time Stream Protocol)
  - Live555, Darwin …
Real-Time Applications – Examples

- Messaging middleware
  - RabbitMQ, ActiveMQ, Kafka, NSQ…

- Distributed memory access
  - Memcached, Redis
Project Topics

- Real-time scheduling evaluation
  - Timing API study
  - Scheduling latency measurement
  - Worst case execution time measurement
  - Real-time scheduling theory

- Kernel hacking
  - EDF scheduler porting

- Virtualization
  - Virtualization on Raspberry Pi
  - Real-time Xen
  - Light-weight virtualization
  - Measurement study on public cloud

- Real-time applications
Pointers

- Real-time benchmarks

- WCET tool challenge