IRMOS Tutorial

The IRMOS Real-Time Scheduler

by

Tommaso Cucinotta

cucinotta -at-domain- sssup.it

12th Real-Time Linux Workshop
Strathmore University
Nairobi, Kenya
October 26th, 2010
Outline

- About the IRMOS Project
- IRMOS Real-Time Scheduler
- Tutorial Session
- Conclusions
Outline

- About the IRMOS Project
- IRMOS Real-Time Scheduler
- Tutorial Session
- Conclusions
About IRMOS

- FP7 IP - Call ICT-2007.1.2
- 3-Years: Feb 2008 – Jan 2011
- 11 Partners - Total Cost: €12.9M
Distributed interactive real-time multimedia applications

- **Deployment of VSNs on PNs**
  - Given computing/network requirements
  - Respecting end-to-end timing constraints

![Diagram](image.png)
Two-Phase Approach

Modeling, Analysis, Planning
  Application Component Description
  Service Component Development & Packaging
  Application Development
  Service Design
  Discovery Negotiation
  Service Instantiation
  Service Component Configuration
  Execution & Monitoring
  Cleanup

Benchmarking

Design

Application Concretions
  Discovery Negotiation
  Reservation

Offline

Online
# Mechanisms

- **QoS-awareness and Real-time:**
  - Real-Time CPU Scheduling among VMs
  - Networking
  - Storage Access

- **Virtualization & Fault-tolerance**

- **Workflow Management/Monitoring**

- ...
Outline

- About the IRMOS Project
- IRMOS Real-Time Scheduler
- Tutorial Session
- Conclusions
IRMOS Scheduler

Features at a glance

- Resource Reservations
  - EDF-based scheduling (hard CBS)

- Hierarchical scheduling
  - Multiple tasks attached to same reservation
  - POSIX Fixed Priority scheduling inside each reservation

- Multi-processor reservations
  - Partitioned scheduling for improved efficiency
  - Migration of tasks among CPUs

- Simple admission control
IRMOS RT Scheduler
Design Goals

- Replace real-time throttling
- Tight integration in Linux kernel
  - Modification to the Linux RT scheduler
- Reuse as many Linux features as possible
  - Management of task hierarchies and scheduling parameters via cgroups
  - POSIX compatibility and API
- Efficient for SMP
  - Independent runqueues
IRMOS Scheduler

- Slice the available computing power into reservations

\[(Q_1, P_1)\]  \[(Q_2, P_2)\]  \[(Q_3, P_3)\]
Hierarchical Scheduling

• Partitioned CBS

- Fixed Priority Scheduling Of Tasks

- Fixed Priority Scheduling Of Tasks

- Partitioned Deadline-Based Scheduling Of Entities (groups)

CPU0
Multi-processor Scheduling

- **Group-wide POSIX Fixed-Priority**
  - SCHED_RR, SCHED_FIFO both possible
  - With M CPUs, if \( N \leq M \) partitioned reservations are scheduled, then the \( N \) highest priority tasks in the group concurrently run

```
  push
  pull
```

Tommaso Cucinotta – Real-Time System Laboratory – Scuola Superiore Sant'Anna – Pisa – Italy
Outline

- About the IRMOS Project
- IRMOS Real-Time Scheduler
- Tutorial Session
- Conclusions
Preliminary Operations
Compiling the Kernel

- Set-up compilation dependencies
  - <set-up proxy>
  - `apt-get install libncurses-dev numactl`

- Compiling the kernel (Nairobi)
  - `scp root@192.168.176.146:linux* .`
  - `tar -xjf linux-irmos.tar.bz2`
  - `cd linux-irmos`
  - `make menuconfig`
  - `make -j4`
Preliminary Operations

Installing the kernel

- make modules_install
- make install
- update-initramfs -c -k 2.6.35-fabio-irmos-1.2testing
- Update-grub2

Reboot the system

- At the boot loader menu, choose the new kernel (-fabio-irmos-1.2testing)

Pin application to core 0

- numactl -C 0 /path/to/application ...
Installation

- **Install binary kernel package(s)**
  - Download the RPM/DEB binary kernel packages
    - Go to the AQuoSA website: http://aquosa.sourceforge.net
    - Follow Downloads → Get AQuoSA → kernel-binaries

- **Compile from kernel sources**
  - Download Linux 2.6.30.10
    - tar -xjf linux-2.6.30.10.tar.bz2
    - cd linux-2.6.30.10
  - Download the IRMOS patch
    - Go to the AQuoSA website: http://aquosa.sourceforge.net
    - Follow Downloads → Get AQuoSA → kernel-patches
  - Apply the patch
    - patch -p1 < /path/to/linux-2.6.30.10-irmos-1.0.patch
Configuration

Configure, compile, install kernel
- make menuconfig; make -j4; sudo make modules_install; sudo make install

Needed kernel compile-time options
- RT_GROUP_SCHED
- GROUP_SCHED
- CGROUPS
- CGROUP_SCHED
- EXPERIMENTAL
- CONFIG_PREEMPT (recommended)
- CGROUP_CPUACCT (recommended)
- SCHED_DEBUG (recommended)
- HIGH_RES_TIMERS
- HZ_1000 (suggested)

Run-time tunable options (recommended)
- mount -t debugfs debugfs /sys/kernel/debug/
- echo NO_RUNTIME_BALANCING > /sys/kernel/debug/sched_features
- echo HRTICK > /sys/kernel/debug/sched_features
Group Management

- **Pre-requisite at run-time: mount cgroups**
  - mkdir /cg
  - mount -t cgroup -o cpu cgroup /cg

- **Reduce runtime for root-level tasks**
  - echo 200000 > /cg/cpu.rt_rt_task_runtime_us
  (root-group runtime remains at default of 950000)

- **Create group, with reservation of 10ms every 100ms**
  - mkdir /cg/g1
  - echo 100000 > /cg/g1/cpu.rt_period_us
  - echo 10000 > /cg/g1/cpu.rt_runtime_us
  - echo 100000 > /cg/g1/cpu.rt_task_period_us
  - echo 10000 > /cg/g1/cpu.rt_task_runtime_us

- **Attach task with tid=1421**
  - echo 1421 > /cg/g1/tasks

- **Detach task**
  - echo 1421 > /cg/tasks

- **Attach process with pid=1700**
  - for tid in `ls /proc/1700/task`; do echo $tid > /cg/g1/tasks; done

- **Destroy group**
  - rmdir /cg/g1
POSIX Priorities

- After attachment to a group
  - tasks are still SCHED_OTHER

- Need to set real-time scheduling
  - Class (SCHED_RR or SCHED_FIFO)
  - Priority

- Task 1421 at SCHED_RR with rtprio=20
  - `chrt -r -p 20 1421`

- Return at SCHED_OTHER
  - `chrt -o -p 1421`

- Process 1700 at SCHED_RR
  - `for tid in `ls /proc/1700/task`; do chrt -o -p $tid; done`
Things easier with AQuoSA

- Create group, with reservation of 10ms every 100ms
  - qres create q=10000 p=100000
  - SID of new server: 1
- Attach task with tid=1421
  - qres attach sid=1 tid=1421
- Detach task
  - qres detach sid=1 tid=1421
- Attach process with pid=1700
  - qres attach sid=1 pid=1700
- Destroy group
  - qres destroy sid=1
- All-in-one command
  - qres create q=30000 p=100000 attach tid=1421
- Wrap an entire process at start
  - qres create q=30000 p=100000 wrap qemu-kvm ...

AQuoSA qosres API

AQuoSA scheduler

IRMOS scheduler

Aquosa manager

ioctl() cgrees
Outline

- About the IRMOS Project
- IRMOS Real-Time Scheduler
- Tutorial Session
- Conclusions
Conclusions

- **IRMOS real-time scheduler**
  - Hierarchical EDF/FP scheduling
  - Multi-processor reservations
  - User-space interface based on cgroups

- **Things to improve**
  - Kernel-space ↔ User-space communication
  - Access-control model
  - Add a desired budget for adaptive reservation
  - Soft reservations

- **We're working on it**
  - Stay tuned
    - [http://aquosa.sourceforge.net](http://aquosa.sourceforge.net)
    - [http://retis.sssup.it/~tommaso](http://retis.sssup.it/~tommaso)
    - [http://retis.sssup.it/~fabio](http://retis.sssup.it/~fabio)
Related Publications

- **Hierarchical Multiprocessor CPU Reservations for the Linux Kernel**
  F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari
  OSPERT 2009, Dublin, Ireland, June 2009

- **Access Control for Adaptive Reservations on Multi-User Systems**
  T. Cucinotta
  RTAS 2008, St. Louis, MO, United States, April 2008

- **Self-tuning Schedulers for Legacy Real-Time Applications**
  T. Cucinotta, F. Checconi, L. Abeni, L. Palopoli
  EuroSys 2010, Paris, April 2010

- **Respecting temporal constraints in virtualised services**
  T. Cucinotta, G. Anastasi, L. Abeni
  RTSOAA 2009, Seattle, Washington, July 2009
Thank you!

Tommaso Cucinotta
Scuola Superiore Sant'Anna
cucinotta at sssup.it

Further Information
http://www.irmosproject.eu

The research leading to these results has received funding from the European Commission Seventh Framework Programme FP7/2007-2011 under grant agreement n° 214777