Using Embedded Linux for Infrastructure Systems

Yoshitake Kobayashi

Embedded Linux Conference Europe 2014
14 Oct 2014
Scope of this presentation

- **Make a Place for collaboration**

- **NOTE**
  - This project currently belongs to CE workgroup, Linux Foundation
  - Anyone interested in this activities, please contact us
Outline

- Definition of Social Infrastructure Systems
- Problem statement
- Expected goal
- Project scope
- Activities
- Current status
Definition of Social Infrastructure Systems (SI systems)

- **Embedded systems which uses for the following infrastructure systems**
  - Transport infrastructure
  - Energy infrastructure
  - Water management infrastructure
  - Communications infrastructure
  - Earth monitoring and measurement systems

- **The system includes**
  - Automation systems
  - Controllers
    - PLC
    - Microcontrollers
    - Signal controller
    - Multi-purpose controller
    - Mini-server based controller (includes PC based controller)
  - Sensor network systems
An Example of Infrastructure System
IoT?

THE INTERNET OF THINGS
Infrastructure System becomes a part of IoT

THE INTERNET OF THINGS

Moving Forward

Controller (e.g. PLC)
Problem statement

- The SI systems will be work with other systems such as:
  - Cloud computing
  - IoT / IoE related systems or devices
Problem statement

- To connect with other systems, SI systems
  - Need to have same quality with new requirement
  - Need to have new technology to have new functionality

- New requirements
  - Ensure a complicated performance requirement
    - Nondeterministic and deterministic programs runs on a same board
  - Collaborate with cloud platform

- New functionality
  - Use fast non-volatile memory (e.g. MRAM, FeRAM)
Goal

- **Main goal**
  - Develop reference implementations for SI system

- **Reference implementation may includes:**
  - Linux kernel and filesystem for some target board
    - Target board: TBD
    - Open spec hardware is better for the implementation
  - Linux kernel related implementations should post to LKML for mainlining
  - Test-cases
Project scope

- **Security**
  - Protect from irregular activities (e.g. viruses)
    - Integrity check
  - Dependability
  - Robustness

- **Run-Time Maintenance / Update**
  - e.g. Kpatch / kGraft

- **Real-time**
  - Hard real-time
  - Multi-purpose real-time system

- **Tests**
  - Reliability test

- **Long Term Support**
  - VLTS = Very Long Term Support

- **Legal/Compliance Issue**

- **(Possibly) Functional Safety**

- **The other scopes to be discussed**
Activities

- **This project will be separated into several different sub-project by its key technology**

- **Initial activity**
  - Survey to determine current common key issues for SI systems
    - Create a mail list
    - F2F meeting at Linux Foundation’s conference
      - Embedded Linux Conference / LinuxCon
    - Workgroup meeting will be arranged
  - Reviewed by members of the CE Workgroup Architecture Group
Estimated output

- **Publicly available pages on the elinux wiki which includes the follows:**
  - A list of issues
  - Specification for reference implementations
    - A list of link to reference implementations
  - All kernel related implementation should post to LKML

- **Reference implementations for key issues will be available on GitHub or alternative service**
Current status

- Under survey to determine actual issues by discussion
- Start to call for participation
Project scope

- **Security**
  - Protect from irregular activities (e.g. viruses)
    - Integrity check
  - Dependability
  - Robustness

- **Run-Time Maintenance / Update**
  - e.g. kpatch / kGraft

- **Real-time**
  - Hard real-time
  - Multi-purpose real-time system

- **Tests**
  - Reliability test

- **Long Term Support**
  - VLTS = Very Long Term Support

- **Legal/Compliance Issue**

- **(Possibly) Functional Safety**

- **The other scopes to be discussed**
Project scope

- **Security**
  - Protect from irregular activities (e.g. viruses)
    - Integrity check
  - Dependability
  - Robustness

- **Run-Time Maintenance / Update**
  - e.g. kpatch / kGraft

- **Real-time**
  - Hard real-time
  - Multi-purpose real-time system

- **Tests**
  - Reliability test

- **Long Term Support**
  - VLTS = Very Long Term Support

- **Legal/Compliance Issue**

- **(Possibly) Functional Safety**

- **The other scopes to be discussed**
- **Real-time**
  - Improve scheduling granularity for SCHED_DEADLINE
  - Hardware resource isolation to run RT and GP process

- **Test**
  - Deta reliability evaluation for Linux filesystems

- **Other**
  - Long-term testing by accelerated kernel
Issue on SCHED_DEADLINE

- Difficult to keep task’s budget if the budget has micro seconds granularity

**Period: 1ms, Budget: 0.5ms**

**Step 1: Refill budget**
- **1.5ms**
  - `dl_timer`

**Step 2: Use budget**
- **1.5ms**
  - `task_tick_dl`
  - **1ms**
  - **2ms**

**Task execution**

- **Wakeup**
- **Period**
- **Wakeup**
- **Period**
Support for micro seconds granularity

When a task’s budget is less than 1ms, set HRTICK for the rest of budget

Period: 1ms, Budget: 0.5ms

Step1: Refill budget

Step2: Use budget

Source code is available on GitHub (https://github.com/ystk/sched-deadline)
Let’s play a movie

- **Original**
- **Modified**

- **Play the movie about 3 times faster than normal speed**
  - Requirement for a frame decoding
    - Period: 10ms, budget: 3.6ms
  - When a frame decode is not finished in a period, the player shows “DEADLINE MISS” on screen and add 3ms penalty

(c) copyright 2008, Blender Foundation
License: Creative Commons Attribution 3.0
- **Realtime**
  - Improve scheduling granularity for SCHED_DEADLINE
  - Hardware resource isolation to run RT and GP process
- **Reliability**
  - Deta reliability evaluation for Linux filesystems
- **Functionality**
  - Long-term testing by accelerated kernel
RT process and GP process runs on a board

**Goal**
- Run RT processes and GP processes on a hardware platform

![Diagram showing RT and GP processes on a board, with isolation from GP application's behavior.]
An example for isolation demo

**Point**

RT Control and UART interrupts are isolated from the Web server and other interrupts.

**Workload (Web clients)**

Pandaboard

- Web server
- RT control
- Core0
- Core1
- Eth0
- SD
- UART

Puppy

- Sensors (Gyroscope, etc.)
- Motor
- CAN

- Sensor values
- Serial-CAN converter
- Motor torque

RS232C
DEMO
An issue while control the robot

Latency for RT app while running with GP application

Evaluation
Environment
Hardware
- Pandaboard
  - ARM Dual core 1.2GHz
  - Memory 1GB
  - SDcard

OS
- Linux 3.6.11.6-RT38

Application
- RT App
  - Periodic task
    Period = 300 micro sec
- GP App
  - Random I/O on SDcard

This latency cause a failure
Realtime performance improvement with Timer-Shielding

Issue
Some kernel task does not able to allocate to target core

Timer management

Hardware Partition Management (Create / Modify / Delete)

API to manage hardware partitions

Prohibit to register non RT related timers

Migrator

Migrate non RT related timers

Timer-Shield OFF

Timer-Shield ON

Embedded Linux Conference Europe 2014
An issue while control the robot

Latency for RT app while running with GP application

Evaluation
Environment
Hardware
- Pandaboard
  - ARM Dual core 1.2GHz
  - Memory 1GB
  - SDcard

OS
- Linux 3.6.11.6-RT38

Application
- RT App
  - Periodic task
    Period = 300 micro sec
- GP App
  - Random I/O on SDcard
- **Realtime**
  - Improve scheduling granularity for SCHED_DEADLINE
  - Hardware resource isolation to run RT and GP process

- **Test**
  - Deta reliability evaluation for Linux filesystems

- **Other**
  - Long-term testing by accelerated kernel
Data reliability test

- **Power off while running writer processes**
- **Verify all data**
  - File size
  - Contents

Test case is available on GitHub (https://github.com/ystk/fs-test)
# Results of data reliability tests on Linux filesystems

**Point 1:** An filesystem has different characteristics on different kernel.

**Point 2:** 2.6.33 has high error rate on ordered and writeback mode.

**Point 3:** Ext4-journal and Btrfs has good results.

<table>
<thead>
<tr>
<th>File system types</th>
<th>Error rate [%]</th>
<th>File size mismatch rate</th>
<th>Data mismatch rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT3-ORDERED</td>
<td>0.00</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>EXT3-JOURNAL</td>
<td>0.00</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>EXT4-JOURNAL</td>
<td>82.4%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>EXT4-ORDERED</td>
<td>43.2%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>EXT4-WRITEBACK</td>
<td>13.3%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>JFS</td>
<td>45.9%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>XFS</td>
<td>43.4%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>XFS</td>
<td>41.4%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
<tr>
<td>BTRFS</td>
<td>43.2%</td>
<td>Blue</td>
<td>Purple</td>
</tr>
</tbody>
</table>
Realtime
- Improve scheduling granularity for SCHED_DEADLINE
- Hardware resource isolation to run RT and GP process

Test
- Detla reliability evaluation for Linux filesystems

Other
- Long-term testing by accelerated kernel
Long-term running test by accelerated kernel

Issue

- Long-term testing takes really long time
  → We want results as fast as possible

Accelerate

Things that NOT able to be accelerated.
- CPU frequency
- HDD or SSD access speed
- Network link speed
  ...

Accelerate clock counter
jiffies = jiffies + (1 * ratio)

Detect clock counter related errors (e.g. time_t)
Evaluation with accelerated kernel

**Xdaliclock works as a stopwatch**

About 40 times faster to get 100% CPU usage

returned incorrect value after about 450 days
Enough?

- Real-time latency is short enough
- Data is safe enough
- System works long enough
Summary

- There are many issues need to be solved for SI system
  - CE workgroup is considering to have sub project for it

- Current status
  - Under survey for actual issues
  - Start call for participation

- Initial activity will be the follows:
  - Clarifying issues to be solved
  - Create a mail list
  - Have F2F meetings at Linux Foundation’s conference

- Estimated output
  - Publicly available pages on the elinux wiki or Linux Foundation’s web site which includes the follows:
    - A list of issues
    - Specification for reference implementations
      - A list of link to reference implementations
      - All kernel related implementation should post to LKML
    - All source code for key issues will be opened
Discussion

- Any comments and suggestions are welcome

- Contact information
  - To get the latest information, please send an email to the following adderss:
    - Yoshitake Kobayashi  yoshitake.kobayashi@toshiba.co.jp
    - Noriaki Fukuyasu  fukuyasu@linuxfoundation.org
Demo (Not Isolated)
Demo (Isolated)