Keynote Speech:
Xen ARM Virtualization

VP Sang-bum Suh, Ph.D.
sbuk.suh@samsung.com
S/W Platform Team
DMC Research Center
SAMSUNG Electronics

2 November 2011, Seoul Korea

Xen Summit Asia 2011
Contents

- SEC Overview
- DMC R&D Center Overview
- Xen ARM Virtualization
SEC Overview
Corporate Philosophy

We will devote our people and technology to create superior products and services thereby contributing to a better global society.
History

- Established the company
- Started manufacturing B&W TV
- Ranked #1 in DRAM
- Developed the cellular telephone system
- Became market leader in flash memory
- Achieved leading share of LCD panel market
- Introduced mobile WiMAX technology (World’s 1st)
- Ranked #1 in TV market
- Ranked #2 in global handset market
- No.1 revenue in global electronics industry ($134B)
Business Divisions

Visual Display

Mobile Comm.

Memory

IT Solutions

Telecomm. Systems / HME

System LSI

Digital Appliances

Digital Imaging

LCD

© 2011 SAMSUNG Electronics Co.
## Recent Technology Leadership

### Pioneering new technologies

<table>
<thead>
<tr>
<th>Year</th>
<th>Event Description</th>
<th>Reference Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>World’s largest TV</td>
<td>Sep 2005</td>
</tr>
<tr>
<td>2006</td>
<td>World’s first HSDPA phone</td>
<td>May 2006</td>
</tr>
<tr>
<td>2007</td>
<td>World’s first 30nm 64GB NAND</td>
<td>2007</td>
</tr>
<tr>
<td>2008</td>
<td>World’s first HSUPA phone</td>
<td>Apr 2008</td>
</tr>
<tr>
<td>2009</td>
<td>World’s slimmest LED TV</td>
<td>Jan 2009</td>
</tr>
<tr>
<td>2010</td>
<td>World’s first 30nm 2GB DDR DRAM</td>
<td>Jan 2010</td>
</tr>
</tbody>
</table>
DMC R&D Center Overview
Core R&D Domain (1/3)

1. NG Comm. & Networking

Conduct research for NG communication systems & connectivity solutions in advance

- NG mobile comm. system
- Wired/Wireless connectivity
- NG broadcast & service technologies

2. Advanced Media Processing

Create NG multimedia devices using innovative technologies

- NG display & audio solution (UHD, 3D, Amp, Speaker)
- NG video/audio codec
- Realistic graphics
- Medical imaging
3. Convergence & Platform Solutions

Build a new kind of ecosystem for multi-device convergence & improve platform competitiveness

- Multi-device convergence (AllShare\(^1\), Smart Home)
- Mobile S/W platform (SLP)
- Cloud service platform

4. Intelligent/Emotional Interaction

Create customized intelligent/emotional UX

- UI identity for SEC’s device
- Multimodal interaction (Flexible & Ambient interface)
- NG UX (Context awareness)

1) AllShare: Integrated Service Solution of SEC (IT/Smart CE/Non-IT Devices)
5. Differentiated Device Solutions

Differentiate mobile device through innovative module solution & sensor application

- Camera SoC (DSC/CAM common)
- Mobile camera module
- Sensor application
- New function module (EMR) pen

6. Eco-friendly Solutions

Develop eco-friendly core technologies & create new business opportunities

- Energy management (HEMS, BEMS)
- Energy saving (printer, air conditioner)
- Life-care solution (Water/Air care, u-Health, etc.)
- Clean material

1) EMR: Electro Magnetic Resonance
Xen ARM Virtualization
Future Computing Trends

Changes in Computing

Closed Centralized Correct Info. Stationary
- Keyboard/Mouse
- Voice Call, SMS
- Centralized/Concentrated
- Known Comm. Entities

Open Distributed Correct+Timely Info. Mobile
- Augmented Reality
- Gesture
- Interactive 3D UI
- Eye-Tracking
- Manytouch
- Realtime Web
- Distributed/Scattered
- Unknown/Utrusted Comm. Entities

Sensor Network

Every Node as Both of Client/Server

Embedded

Single-core
- UC Berkeley Sensornet Chip (TI MSP430 8MHz core, 10KB RAM)
- Tiger 1GHz Single-core
- Dunnington 3GHz 6-core

Multi-core
- [2009]
- ARM 2GHz 4-core
- Intel 4GHz 32-core

Many-core
- [2012]
- ARM 3GHz 8-core
- Intel 6GHz 128-core
- SensorNet Chip (128MHz core, 160KB RAM)

IT

Single-core

Multi-core

Many-core

“Privacy”

“Realtime”
Industry Trends

- Introduction of Virtualization Technology in Embedded Devices
- Strengthening of Smartphone Features

- Ubiquitous Instant Boot (Android quick boot)
- Wind River Acquisition (VxWorks, RTLinux)
- Symbian OS Open source (2010.02)
- Google Android
- Linux-based mobile OS (2010.01)
- MS Windows Phone 7 ('2010 4Q)
- Google Chrome OS ('2010 4Q)
- Apple iOS Sandbox
- Google Android Sandbox & Permission-based Access Control
- Google Chrome Browser Sandbox & Renderer Process Isolation

* RTM: Root of Trust Measurement

© 2011 SAMSUNG Electronics Co.
Why CE Virtualization?

1. **HW Consolidation:** AP(Application Processor) and BP(Baseband Processor) can share multicore ARM CPU SoC in order to run both Linux and Real-time OS efficiently.

2. **OS Isolation:** important call services can be effectively separated from downloaded third party applications by Xen ARM combined with access control.

3. **Rich User Experience:** multiple OS domains can run concurrently on a single smartphone.

---

**Diagram:**
- **AP SoC + BP SoC:** Consolidated Multicore SoC
- **Virtualization SW (Realtime Hypervisor):** Important services
- **H/W:** Secure Smartphone
- **Linux 1** and **Linux 2:** Rich Applications from Multiple OS
- **Secure Kernel:** Hypervisor
- **Hypervisor:** Hardware
Xen ARM Virtualization

**Goals**

- Lightweight virtualization for secure 3G/4G mobile devices
  - High performance hypervisor based on ARM processor
  - Fine-grained access control fitted to mobile devices

**Architecture of Xen ARM**

- Guest Domain
  - Application
  - Backend Drivers
  - Native Drivers
  - VM Interface

- Secure Xen ARM Hypervisor
  - Domain Manager
  - Resource Allocator
  - Access Control

- Hardware
  - Peripheral Devices
  - CPU
  - System Memory
  - UART
Xen ARM Virtualization

- **CPU virtualization**
  - Virtualization requires 3 privilege CPU levels, but ARM supports 2 levels
    - Xen ARM mode: supervisor mode (most privileged level)
    - Virtual kernel mode: User mode (least privileged level)
    - Virtual user mode: User mode (least privileged level)

- **Memory virtualization**
  - VM's local memory should be protected from other VMs
    - Xen ARM switches VM's virtual address space using MMU
    - VM is not allowed to manipulate MMU directly

- **I/O virtualization**
  - Split driver model of Xen ARM
    - Client & Server architecture for shared I/O devices
      - Client: frontend driver
      - Server: native/backend driver

Memory virtualization

VM's local memory should be protected from other VMs

- Xen ARM switches VM's virtual address space using MMU
- VM is not allowed to manipulate MMU directly

I/O virtualization

- Split driver model of Xen ARM
  - Client & Server architecture for shared I/O devices
    - Client: frontend driver
    - Server: native/backend driver
Performance Evaluation
Virtualization Overhead

Micro-benchmark Results

- Evaluation Environments: Samsung Blackjack Phone
  - CPU: Xscale PXA310, 624MHz
  - L1 Cache: 32KB + 32KB
  - L2 Cache: 256KB (Disabled)
  - Memory: 128MB
  - Guest OS: Linux-2.6.21

LMBENCH Micro Benchmark (latency)

- Higher is better

LMBENCH Micro Benchmark (Bandwidth)

- Higher is better

Lower is better

© 2011 SAMSUNG Electronics Co.
Virtualization Overhead Comparison

**Benchmark Results**

- **Evaluation Environments**: Samsung Blackjack Phone
  - CPU: Xscale PXA310, 624MHz
  - L1 Cache: 32KB + 32KB
  - L2 Cache: 256KB (Disabled)
  - Memory: 128MB
  - Guest OS: Linux-2.6.21

**LMBENCH Micro Benchmark**

- **Relative Performance**
  - Higher is better
  - Xen/ARM vs L4

**AIM7 Macro Benchmark**

- **Normalized Performance**
  - Native Linux
  - Xen/ARM
  - L4

**Evaluation Environments**: Samsung Blackjack Phone
- CPU: Xscale PXA310, 624MHz
- L1 Cache: 32KB + 32KB
- L2 Cache: 256KB (Disabled)
- Memory: 128MB
- Guest OS: Linux-2.6.21
Performance Comparison

Micro-benchmark Results

- **Evaluation Environments**: nVidia Tegra250
  - CPU: Cortex-A9 1GHz Dual Core
  - L1 Cache: 32KB + 32KB
  - L2 Cache: 1MB
  - Memory: 1GB
  - Guest OS: Linux-2.6.29

Lower is better

**LMBENCH Micro Benchmark (latency)**

- Native Linux
- Para-virtualized Linux
Real-time Performance

- **Evaluation Environment**

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/W (Tegra250)</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Cortex-A9 / 1GHz / Dual Core</td>
</tr>
<tr>
<td>RAM</td>
<td>1GB</td>
</tr>
<tr>
<td>S/W</td>
<td></td>
</tr>
<tr>
<td>Hypervisor</td>
<td>Xen ARM</td>
</tr>
<tr>
<td>Guest OS (DOM0)</td>
<td>Linux-2.6.29</td>
</tr>
<tr>
<td></td>
<td>(Running Busy Loop Task)</td>
</tr>
<tr>
<td>Guest OS (DOM1)</td>
<td>uC/OS-II</td>
</tr>
<tr>
<td></td>
<td>(Running RT Task : Cyclictest benchmark)</td>
</tr>
</tbody>
</table>

- **Cyclic test benchmark repeats**
  1. RT task sleeps for 10ms
  2. Timer interrupt will occur after 10ms
  3. Timer interrupt wakes up the RT domain (uC/OS-II)
  4. uC/OS-II preempts Xen ARM
  5. RT task is scheduled
  6. RT task logs timestamp

---

**CDF of Responsiveness (Periodic Timer Interrupt : 10ms)**

**Response Overhead (3us)**

<table>
<thead>
<tr>
<th></th>
<th>Native (uC/OS-II)</th>
<th>Xen ARM (uC/OS-II)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>9995</td>
<td>9996</td>
</tr>
<tr>
<td>Avg</td>
<td>9996.810169</td>
<td>9999.327119</td>
</tr>
<tr>
<td>Max</td>
<td>10000</td>
<td>10001</td>
</tr>
</tbody>
</table>

**Unit : usec**
Effectiveness of Access Control

Test Environment

- **net_atk**: UDP packet flooding (sending out UDP packets with the size of 44,160 bytes every 1000 usecs)
- **mtd_atk**: overwhelming NAND READ operations (scanning every directory in the filesystem and reading file contents)

### Test Cases

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Network I/O Test Cases</th>
<th>Storage I/O Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Attack</td>
<td>TcN0</td>
<td>TcS0</td>
</tr>
<tr>
<td>Under Attack (No I/O ACM)</td>
<td>TcN1</td>
<td>TcS1</td>
</tr>
<tr>
<td>Under Attack (20% I/O ACM Policy)</td>
<td>TcN2</td>
<td>TcS2</td>
</tr>
<tr>
<td>Under Attack (10% I/O ACM Policy)</td>
<td>TcN3</td>
<td>TcS3</td>
</tr>
</tbody>
</table>

CPU Utilization: Network

CPU Utilization: Storage

© 2011 SAMSUNG Electronics Co.
Effectiveness of Access Control:

- Throughput: Network
  - Effectiveness of our access control: throughput increase and power consumption decrease even under malware attack.

- Throughput: Storage
  - Throughput increase:
    - Seq.out
    - Seq.in
    - Rand.seek

- Power Consumption
History of Xen ARM

'04  x86 Xen Hypervisor Release (Cambridge University)

'08  Xen ARM 1st Release: ARM9 Xen Hypervisor, Mini-OS (Samsung)

'09  Xen ARM 2nd Release: Paravirtualized Linux kernel (v2.6.24), Xen tool (Samsung)

'10  Xen ARM 3rd Release: ARM11 MPCore Support (Samsung)

'11  Xen ARM 4th Release: Cortex-A9 MPCore Support (Samsung)

Xen ARM Open Source Community


Supported Hardware & Guest OS (Stand-alone Version)

- ARM926EJ-S (i.MX21, OMAP5912)
- Xscale 3rd Generation Architecture (PXA310, Samsung SGH- i780)
- ARM1136/ARM1176(Core Only)
- Goldfish (EQMU Emulator)
- Versatile Platform Board
- ARM11 MPCore (Realview PB11MP)
- Tegra250

- Linux v2.6.11, v2.6.18, v2.6.21, v2.6.24, v2.6.27 (multicore supported)
- uC/OS-II
Future Roadmap of Xen ARM

- '11: Finish initial merge
- '12: Lightweight version of Xen tools
- '13: Cortex-A15 Support

Integration of Xen ARM with mainline (80% completed)
  - Rebased on the recent xen-unstable.hg
  - Many parts of the Xen ARM has been rewritten for the integration.

- Dynamic domheap allocation
  - Support of “pseudo-physical to machine translation” is ongoing.

- Dynamic xenheap expansion
  - Xenheap could be expanded on demand
    - Initially Xen ARM reserves 1MB(1 Section) of memory for heap
Xen ARM Development / Contribution Model

Development / Contribution Model

- xen-devel mailing (Review)
- xen-unstable.hg
- ARM Specific Patches
- Commit
- Pull
- xen-arm.git
- Xen arm mailing (Review)
- Xen ARM Developers
- Commit
- ARM Specific Patches
Issues

- **Xen-Tools**
  - Porting to ARM architecture is required
    - Currently libxc does not support ARM architecture.

- **Real-time**
  - Implementing Real-time Scheduler
    - How does the VMM knows which domain requires real-time scheduling?
  - Implementing VMM Preemption
    - How to minimize interrupts and event latency within the view of VM? (for VM perspective)

- **Access Control**
Thank You!
Issue: Xen-Tools

- Lightweight version of Xen-tools
  - Python-based xend/xm too heavy for small devices.

- Lightweight version of xend/xm for embedded devices
  - Adopt Plug-in architecture
  - To avoid re-compilation when new virtual device introduced.

<table>
<thead>
<tr>
<th>Python-based Xm/Xend</th>
<th>Memory Usage</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Several tens of MB</td>
<td>Several seconds</td>
</tr>
<tr>
<td></td>
<td>Several hundreds of KB.</td>
<td>&lt; 1 second</td>
</tr>
</tbody>
</table>

SW Platform Team.
© 2011 SAMSUNG Electronics Co.
**Issue: Real-time vs. Throughput**

- **Evaluation Environment**

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/W (Tegra250)</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Cortex-A9 / 1GHz / Dual Core</td>
</tr>
<tr>
<td>RAM</td>
<td>1GB</td>
</tr>
<tr>
<td>S/W</td>
<td></td>
</tr>
<tr>
<td>Hypervisor</td>
<td>Xen ARM</td>
</tr>
<tr>
<td>Guest OS (DOM0)</td>
<td>Linux-2.6.29 (Running Busy Loop Task)</td>
</tr>
<tr>
<td>Guest OS (DOM1)</td>
<td>uC/OS-II (Running RT Task: Cyclictest benchmark)</td>
</tr>
</tbody>
</table>

- **Cyclictest benchmark repeats**
  1. RT task sleeps for 10ms
  2. Timer interrupt will occur after 10ms
  3. Timer interrupt wakes up the RT domain (uC/OS-II)
  4. uC/OS-II preempts Xen ARM
  5. RT task is scheduled
  6. RT task logs timestamp

**CDF of Responsiveness (Periodic Timer Interrupt: 10ms)**

<table>
<thead>
<tr>
<th></th>
<th>Native (uC/OS-II)</th>
<th>Xen ARM (uC/OS-II)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>9995</td>
<td>9996</td>
</tr>
<tr>
<td>Avg</td>
<td>9996.810169</td>
<td>9999.327119</td>
</tr>
<tr>
<td>Max</td>
<td>10000</td>
<td>10001</td>
</tr>
</tbody>
</table>

**Unit: usec**
## Issue: Access Control

### sHype, XSM and our ACM

<table>
<thead>
<tr>
<th>Access Control Policies</th>
<th>sHype [SAI05]</th>
<th>XSM [COK06]</th>
<th>Xen ARM ACM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Control Policies</td>
<td>Flexible based on Flask (TE and Chinese Wall)</td>
<td>Flexible based on Flask (TE and Chinese Wall, RBAC, MLS, and MCS)</td>
<td>Flexible based on Flask (TE and proprietary policy)</td>
</tr>
<tr>
<td>Objects of Access Control</td>
<td>Virtual resources and domain management</td>
<td>Physical/virtual resources and domain management</td>
<td>Physical/virtual resources and domain management</td>
</tr>
<tr>
<td>Protection against mobile malware-based DoS attacks</td>
<td>N/A</td>
<td>N/A</td>
<td>Memory, battery, DMA, and event channels are controlled by ACM</td>
</tr>
<tr>
<td>Access control to objects in each guest domain</td>
<td>Enforced by ACM at VMM</td>
<td>Enforced by ACM at VMM</td>
<td>Enforced by ACM at each domain (for performance reason)</td>
</tr>
<tr>
<td>Etc</td>
<td></td>
<td></td>
<td>Xen ARM specific hooks</td>
</tr>
</tbody>
</table>
## Comparison of ARM vs. x86 Virtualizability

<table>
<thead>
<tr>
<th>Comparison</th>
<th>x86</th>
<th>ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring Compression (Protection mechanisms)</td>
<td>Segmentation and Paging</td>
<td>Paging and Domain Protection</td>
</tr>
<tr>
<td>Cache Architecture</td>
<td>PIPT</td>
<td>VIVT / VIPT / PIPT</td>
</tr>
<tr>
<td>I/O</td>
<td>I/O Instructions + memory-mapped I/O</td>
<td>Only memory-mapped I/O</td>
</tr>
<tr>
<td># of privilege levels</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Mobile Malware

• Number of mobile malware
  – More than 420 mobile phone viruses (2008)
  – Tens of thousands of infections worldwide

• Concerns about mobile phone security – by market

Source: F-Secure

Source: McAfee

Mobicom'09, September 20-25, 2009, Beijing, China
Current Status of Xen ARM

Changeset

Common files which have been modified

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xen</td>
<td>Rules.mk</td>
<td>- override TARGET_SUBARCH := $(XEN_TARGET_ARCH)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ override TARGET_SUBARCH := $(XEN_TARGET_SUBARCH)</td>
</tr>
<tr>
<td>xen/common</td>
<td>page_alloc.c</td>
<td>Add reserve_boot_pages() function</td>
</tr>
<tr>
<td>xen/drivers</td>
<td>Makefile</td>
<td>Exclude x86 dependent device drivers when Xen is built for ARM architecture</td>
</tr>
<tr>
<td>xen/include/public</td>
<td>Xen.h</td>
<td>Add preprocessor macros to include arch-arm.h header file.</td>
</tr>
<tr>
<td>xen/include/xen</td>
<td>libelf.h</td>
<td>Add preprocessor macros to support ARM architecture.</td>
</tr>
</tbody>
</table>

New files

- We wrote xxx files for ARM architecture
Xen ARM Access Control

- Protect unauthorized access to system resources from a compromised domain

- 37 access control enforcers in hypercalls

- Flexible architecture based on Flask
  - Currently, 5 access control models supported (TE, BLP, Biba, CW, Samsung Proprietary)

- Access control of the resources
  - Physical resources (TE, Samsung Proprietary)
    - Memory, CPU, I/O space, IRQ
  - Virtual resources (TE, BLP, Biba)
    - Event-channel, grant table
  - Domain management (CW)
    - Domain creation/destroy