Real-time and VMM

Real-time Xen for Embedded Devices

Chuck Yoo and team
Operating Systems Laboratory
Korea University
Feb. 25, 2009
Contents

• Motivation
• Background
  — Two key properties in real-time systems
• Challenges to Xen
  — determinism
  — predictability
• Development status
Motivations

why real-time support is important

• Various forms of new embedded systems
  – Real-time requirement

• Mobile phone to run both:
  – Communication stack like GSM, CDMA
  – Applications like UI, games

• Automobiles, Robots, UAV, and more
Determinism in Real-time System

- **Schedulability**
  - Theoretical foundation
  - Task $T_i = (p_i, c_i, d_i)$, 
    execution $c_i$ time within a deadline $d_i$, during each period $p_i$
  - Schedulability can be decided for a given scheduling policy and a real-time task set

- Real-time operating system performs schedulability test whenever the task set or the scheduling policy is changed
  - Performs admission control so that all the task in the system can meet the execution condition
  - In order to simplify the schedulability analysis at run-time, it fixes the task set and utilization a priori.
Schedulability Test

• For a given Task set $T = \{T_i(p_i, c_i, d_i)\}$ and Algorithm A, scheduler S checks whether all the tasks in the system meet the execution condition.

• $T$ is RM-schedulable if and only if

$$w_{i,k} \leq t,$$

where $t = kp_j, j = 1, 2, ..., i, k = 1, ..., \frac{p_i}{p_j},$

$$w_{i,k} = \frac{c_k}{p_k} t, \quad 0 < t < p_i.$$

• $T$ is EDF-schedulable if and only if

$$U = \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} p_i} b \leq 1.$$
Predictability in Real-time System

• **Interrupt is a key obstacle to predictability**
  - Interrupt is **asynchronous and unpredictable**
  - For predictability, need to make interrupt processing predictable

• **Real-time operating system adopts mechanisms to make it time-bound**
  - Disable another interrupt
  - Nested/prioritize interrupt
  - Polling I/O
  - Deferred (Delayed) processing
    - e.g.) bottom half, softirq, etc.
Xen: Physical time $\neq$ Virtual time

- Physical Machine $PM_1$
- Virtual Machine $VM_1$
- Scheduling over physical machine
- Scheduling over physical machine
- Physical time
- Virtual time
- Discrete system time in $PM_1$
- Discrete system time in $VM_1$
- Media player (24frm/s)
- Media player (12frm/s)
- Not defined
Xen: I/O model

• Does not provide predictable time-bound
• Split driver model is not suitable for real-time I/O
VMM for Embedded Systems

• **Xen-ARM** from Samsung Elec.  
  (first ARM porting)

• **L4** from Open Kernel Labs  
  (microkernel-based virtualization)

• **VLX** from VirtualLogix  
  (real-time support RTOS runs inside the VMM)

Similar issues inherited!
Challenge to Xen

How to Provide Determinism?
Scheduling in Xen

• SEDF, Credit schedulers
  - Simple EDF implements real-time EDF scheduler
    • User should define period, execution slice
  - Credit scheduler implements credit based fair scheduling with I/O boost
    • Supports multi-processor, load balancing
    • I/O BOOST priority to keep responsiveness

• Time keeping in Xen
  - Timer interrupts are distributed over VMs
    • Hardware timer interrupts are handled by the Xen, and delivered by via event channel
    • VM can be aware of time passage by virtual timer interrupt which has been sent by the Xen
      • Time flows only when the guest OS is scheduled by the VMM
How to Perform Schedulability Analysis?

• Problem

- Task scheduling should be performed by RTOS
  - It performs schedulability test, RT scheduling policy
  - Xen does not know the tasks in the RTOS

- However, RT guest is not aware of the physical time
  - RTOS is aware of the virtual time provided by the Xen
  - Deadline might be missed if the Xen does not timely schedule RT guest
Solution 1 - Integrated scheduling at VMM

• Scheduling at the lowest layer
  - At VMM?
    • VMM should be aware of all the real-time tasks in RT guest OS
      - e.g.) L4 task scheduling
    • What if multiple RT guest OS

  - In addition,
    • How to support GPOS
      - Scheduling analysis whenever GPOS creates a user process
      - Sorting all the tasks in all the guests for SEDF at run time should have a considerable overhead
    • Interfere GPOS scheduler
      - Local scheduler knows a best way to utilize the resources
Solution 2 - Workload aggregation

- Hierarchical scheduling framework
  - RT guest OS can preserve its own scheduling policy and schedulability
    - No global schedulability test for local task set change
  - Aggregate RT guest OS (its tasks) as one real-time task

- Implication to Xen
  - Need abstract interface to translate all the RT tasks to a single RT task
  - Slightly more overhead

Scheduleability Test with Abstract interface, i.e. periodic task model $\Gamma(\Pi, \Theta)$

Abstract interface

$\Gamma_1(\Pi, \Theta)$

Subsystem $S_1$

Preserve local scheduling policy, scheduleability test

RT task $T_1(\pi, e_i)$

$\Gamma_2(\Pi, \Theta)$

Subsystem $S_2$

Preserve local scheduling policy, scheduleability test

RT task $T_2(\pi, e_i)$
Aggregation

Periodic resource model

- $k = \max \left( \lceil (t-\Pi-\Theta) \rceil / \Pi, 1 \right)$
  - number of periods within $t$
  - i.e. $k = 3$ for the figure below
**Example of Aggregation**

- For RT guest R whose task set $T$ is $\{t_1(50, 7, 50), t_2(75, 9, 75)\}$ and algorithm $A$ is EDF.
  - Utilization of workload = 0.26
  - Let period $\Pi$, resource capacity $U = \Theta / \Pi$
  - Supply bound function (sbf): minimum resource supply from VMM
    \[
    \text{sbf}_{t}(t) = \begin{cases} 
    \frac{t - k + 1}{\Pi} - \Theta & \text{if } t \in (k + 1)\Pi - 2\Theta, (k + 1)\Pi - \Theta \\
    \frac{k - 1}{\Pi} - \Theta & \text{otherwise}, \\
    \end{cases}
    \]
    where $k = \max \left\{ \frac{t - \Theta}{\Pi}, 1 \right\}$.
  - Demand bound function (dbf): total resource demand for the workload
    \[
    \text{dbf}_{EDF,W,t}(t) = \frac{t}{p_i} \cdot e_i. \\
    \]
  - Gray region in the graph presents feasible resource allocation
    \[
    \forall 0 < t \leq \text{LCM}_W, \text{dbf}_{EDF,W,t}(t) \leq \text{sbf}_{R,t}(t), \\
    \]
    where $\text{LCM}_W$ is the least common multiple of $p_i$ for all $T_i \in W$.

- Let assume that $\Pi = 10$,
  - We need to allocate at least 2.8 (as $\Theta$) to R
  - Rounding it and we get $\Gamma(10, 3)$

---

I. Shin, I. Lee “Compositional real-time scheduling framework with periodic model,”
ACM Trans. on Embedded Computing Systems Vol.7 (3), 2008
To make aggregation work

- To incorporate the hierarchical scheduling framework
  - Need a tool to calculate with the given task set

- Resource supply bound \( \geq \) Resource demand bound
  - Xen should provide enough resources to RTOS

  - RTOS calculates demand bound
    (total resource demand to meet the schedulability)
  - Xen provides CPU resource to RTOS at least to resource supply bound
    (minimum resource supply for a corresponding scheduling policy)
HOW TO DEAL WITH PREDICTABILITY?
How to Handle Interrupts in Time-bounded fashion?

- **Problem**
  - *Xen’s domain model splits physical I/O processing from virtual machine*
  - **Additional latency**
  - **Unbounded processing time**
    - *Dom 0 may be kept busy i.e. target guest may not be scheduled immediately*
Solution 1 – Putting RT guest at Dom 0

• At Dom 0
  - Special control/driver domain
  - For faster I/O response
    • Good for timely interrupt processing

• Cons
  - Another non-RT VM’s I/O processing is affected
    • Hard to support schedulability analysis
      (non-RT guest processing, which is not time-bounded)
  - RT guest can be overloaded with too many interrupts

Diagram:
- RT-Dom 0
- Dom U (non-RT guest)
- Xen
- Interrupt for Dom U (non-RT guest)
Solution 2 - Adopting Real-time Sublayer (RTL)

- Let RT guest access physical devices directly
  - RT devices are dedicated to RT guests
  - Interrupt service routine (ISR) of RT guest should be minimal
- Add real-time sublayer to provide time-bounded interrupts for RT guest
  - Handle RT guest’s interrupt first
  - Throttle interrupt processing for RT guest and non-RT guest
    - Selectively deliver interrupts to dom0 by monitoring incoming interrupts
    - To avoid unbounded interrupt processing for non-RT guest (e.g., network flooding)
More on RTL

- RTL controls the incoming rate of the aggressive interrupts
  - discard interrupts when too many
    if current interrupt incoming rate > maximum interrupt rate threshold

- Max. interrupt rate could be derived from available CPU bandwidth from schedulability analysis
Development status

• Ported uC/OS-II over XenARM platform
  – Freescale iMX21
• Being implemented
  – Hierarchical scheduler
  – Tools for schedulability analysis
  – RTL
• Preview at
  http://www.youtube.com/watch?v=Vli9zb62
Conclusion

- Real-time support is important
- Xen can do it
THANKS!
Real-time and VMM

- Real-time Xen for Embedded Devices

Chuck Yoo and team
Operating Systems Laboratory
Korea University
Feb. 25, 2009
Contents

• Motivation
• Background
  – Two key properties in real-time systems
• Challenges to Xen
  – determinism
  – predictability
• Development status
Motivations

why real-time support is important

• Various forms of new embedded systems
  – Real-time requirement
• Mobile phone to run both:
  – Communication stack like GSM, CDMA
  – Applications like UI, games
• Automobiles, Robots, UAV, and more
Determinism in Real-time System

- **Schedulability**
  - Theoretical foundation
  - Task $T_i = (p_i, c_i, d_i)$, execution $c_i$ time within a deadline $d_i$, during each period $p_i$
  - Schedulability can be decided for a given scheduling policy and a real-time task set

- Real-time operating system performs schedulability test whenever the task set or the scheduling policy is changed
  - Performs admission control so that all the task in the system can meet the execution condition
  - In order to simplify the schedulability analysis at run-time, it fixes the task set and utilization a priori.
Schedulability Test

- For a given Task set $T = \{T_i(p_i, c_i, d_i)\}$ and Algorithm $A$, scheduler $S$ checks whether all the tasks in the system meet the execution condition.

- $T$ is RM-schedulable if and only if
  \[ w_i \left\lfloor \frac{t}{p_i} \right\rfloor \leq t \]
  where $t = kp_j$, $j = 1, 2, \ldots, i, k = 1, \ldots, \left\lfloor \frac{p_j}{p_i} \right\rfloor$,
  \[ w_i \left\lfloor \frac{t}{p_i} \right\rfloor = c_k \left( \frac{t}{p_k} \right), \quad 0 < t < p_i. \]

- $T$ is EDF-schedulable if and only if
  \[ U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1. \]
Predictability in Real-time System

• Interrupt is a key obstacle to predictability
  – Interrupt is asynchronous and undeterministic
  – For predictability, need to make interrupt processing predictable

• Real-time operating system adopts mechanisms to make it time-bounded
  – Disable another interrupt
  – Nested/prioritize interrupt
  – Polling I/O
  – Deferred (Delayed) processing
    • e.g.) bottom half, softirq, etc.
Xen: Physical time $\neq$ Virtual time

- Scheduling over physical machine
  - Physical Machine $P_{M_1}$
  - Physical time
  - Discrete system time in $P_{M_1}$

- Scheduling over virtual machine
  - Virtual Machine $V_{M_1}$
  - Virtual time
  - Discrete system time in $V_{M_1}$

Physical time

Discrete system time

Not defined
Xen: I/O model

- Does not provide predictable time-bound
- Split driver model is not suitable for real-time I/O
VMM for Embedded Systems

- Xen-ARM from Samsung Elec.  
  (first ARM porting)
- L4 from Open Kernel Labs  
  (microkernel-based virtualization)
- VLX from VirtualLogix  
  (real-time support RTOS runs inside the VMM)

Similar issues inherited!
Challenge to Xen

HOW TO PROVIDE DETERMINISM?
Scheduling in Xen

- **SEDF, Credit schedulers**
  - Simple EDF implements real-time EDF scheduler
    - User should define period, execution slice
  - Credit scheduler implements credit-based fair scheduling with I/O boost
    - Supports multi-processor, load balancing
    - I/O BOOST priority to keep responsiveness

- **Time keeping in Xen**
  - Timer interrupts are distributed over VMs
    - Hardware timer interrupts are handled by the Xen, and delivered by via event channel
    - VM can be aware of time passage by virtual timer interrupt which has been sent by the Xen
      - Time flows only when the guest OS is scheduled by the VMM
How to Perform Schedulability Analysis?

- Problem
  - Task scheduling should be performed by RTOS
    - It performs schedulability test, RT scheduling policy
    - Xen does not know the tasks in the RTOS
  - However, RT guest is not aware of the physical time
    - RTOS is aware of the virtual time provided by the Xen
    - Deadline might be missed if the Xen does not timely schedule RT guest

![Diagram](image-url)
Solution 1 - Integrated scheduling at VMM

- **Scheduling at the lowest layer**
  - At VMM?
    - VMM should be aware of all the real-time tasks in RT guest OS
      - e.g.) L4 task scheduling
    - What if multiple RT guest OS
  - In addition,
    - How to support GPOS
      - Scheduling analysis whenever GPOS creates a user process
      - Sorting all the tasks in all the guests for SEDF at run time should have a considerable overhead
    - Interfere GPOS scheduler
      - Local scheduler knows a best way to utilize the resources
Solution 2 - Workload aggregation

- Hierarchical scheduling framework
  - RT guest OS can preserve its own scheduling policy and schedulability
    - No global schedulability test for local task set change
  - Aggregate RT guest OS (its tasks) as one real-time task
  - Implication to Xen
    - Need abstract interface to translate all the RT tasks to a single RT task
    - Slightly more overhead

Schedule Test with Abstract interface, i.e. periodic task model \( \Gamma_i(\Pi, \Theta) \)

Preserve local scheduling policy, schedulability test

RT task \( T_1(p, i, e_1) \)

RT task \( T_2(p, i, e_2) \)
Aggregation
Periodic resource model

- $k = \max\left( \left\lceil \frac{t-(\Pi-\Theta)}{\Pi} \right\rceil, 1 \right)$
  - number of periods within $t$
  - i.e. $k = 3$ for the figure below
**Example of Aggregation**

- For RT guest $R$ whose task set $T$ is $\{t_1(50,7,50), t_2(75,9,75)\}$ and algorithm $A$ is $EDF$.
  - Utilization of workload = 0.26
  - Let period $\Pi$, resource capacity $U = \Theta / \Pi$
  - Supply bound function (sbf): minimum resource supply from VMM
    \[
    sbf(t) = \begin{cases} 
    k - \Theta, & \text{if } t \in [k\Pi, (k+1)\Pi - \Theta] \\
    k - 1\Theta, & \text{otherwise}, 
    \end{cases}
    \]
    where $k = \max\{\frac{t - \Theta}{\Pi} \}$
  - Demand bound function (dbf): total resource demand for the workload
    \[
    dbf_{EDF}(T, t) = \sum_{i \in T} \frac{t \cdot p_i}{T_i}. 
    \]
    \[
    \forall 0 < t \leq LCM_W, dbf_{EDF}(W, t) \leq sbf_{EDF}(t), 
    \]
    where $LCM_W$ is the least common multiple of $p_i$ for all $T_i \in W$.

- Let assume that $\Pi = 10$.
  - We need to allocate at least 2.8 (as $\Theta$) to $R$.
  - Rounding it and we get $\Gamma(10, 3)$.
To make aggregation work

- To incorporate the hierarchical scheduling framework
  - Need a tool to calculate with the given task set

- Resource supply bound $\geq$ Resource demand bound
  - Xen should provide enough resources to RTOS
    - RTOS calculates demand bound
      (total resource demand to meet the schedulability)
    - Xen provides CPU resource to RTOS at least to resource supply bound
      (minimum resource supply for a corresponding scheduling policy)
Challenge to Xen

HOW TO DEAL WITH PREDICTABILITY?
How to Handle Interrupts in Time-bounded fashion?

• Problem
  - Xen’s domain model splits physical I/O processing from virtual machine
    • Additional latency
    • Unbounded-processing time
      - Dom 0 may be kept busy, i.e., target guest may not be scheduled immediately
Solution 1 — Putting RT guest at Dom 0

- At Dom 0
  - Special control/driver domain
  - For faster I/O response
    - Good for timely interrupt processing

- Cons
  - Another non-RT VM’s I/O processing is affected
    - Hard to support schedulability analysis
      (non-RT guest processing, which is not time-bounded)
  - RT guest can be overloaded with too many interrupts
Solution 2 - Adopting Real-time Sublayer (RTL)

- Let RT guest access physical devices directly
  - RT devices are dedicated to RT guests
  - Interrupt service routine (ISR) of RT guest should be minimal
- Add real-time sublayer to provide time-bounded interrupts for RT guest
  - Handle RT guest’s interrupt first
  - Throttle interrupt processing for RT guest and non-RT guest
    * Selectively deliver interrupts to dom0 by monitoring incoming interrupts
    * To avoid unbounded interrupt processing for non-RT guest
      e.g. network flooding
More on RTL

• RTL controls the incoming rate of the aggressive interrupts
  - discard interrupts when too many
    if current interrupt incoming rate > maximum interrupt rate threshold

• Maximum interrupt rate could be derived from available CPU bandwidth from schedulability analysis
Development status

• Ported uC/OS-II over XenARM platform
  – Freescale iMX 21
• Being implemented
  – Hierarchical scheduler
  – Tools for schedulability analysis
  – RTL
• Preview at
  http://www.youtube.com/watch?v=Vli9zb62
Conclusion

• Real-time support is important
• Xen can do it
THANKS!