Energy-Efficient Mapping Technique for Virtual Cores

Yu-Chia Lin, Chuan-Yue Yang, Che-Wei Chang, Yuan-Hao Chang, Tei-Wei Kuo, and Chi-Sheng Shih
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Introduction

- Virtualization technique: A layer of abstraction between computer hardware systems and the software running on them
  
  - Benefits
    - Increase resource utilization
    - Improve the deployments of administrators
  
  - Categories
    - Platform virtualization
    - Resource virtualization
    - Application virtualization
Support the DVS functionality in virtual machines

Modify the virtualization technique for embedded systems
Related Work

- **High performance**
  - Manage CPU-Intensive resource in virtualization environments
    - CPU resource guarantee\(^1\) and fairness\(^2\)
  - Increase performance in multiple-core platforms based on virtualization\(^3\)

- **Security**
  - Provide safe execution environments\(^4\) and guarantee code integrity\(^5\)
  - Security operating systems\(^6,7\)

---

\(^1\) VMWare ESX Server 2: ESX Server Performance and Resource Management for CPU-Intensive Workload, VMWARE White Paper
\(^2\) Credit-Based CPU Scheduler, [http://wiki.xensource.com/xenwiki/CreditScheduler](http://wiki.xensource.com/xenwiki/CreditScheduler)
\(^6\) the VFiasco project, [http://os.inf.tu-dresden.de/vfiasco/](http://os.inf.tu-dresden.de/vfiasco/)
Related Work (cont.)

- **Embedded system: μ-kernel**
  - Provide a high performance design\(^8\)
  - Improve the dependability by unmodified drivers in virtual machines\(^9\)

---


Related Work (cont.)

- Energy-related researches of virtualization environments
  - Energy efficiency in enterprise systems
    - On-line power management \(^1\)
    - Energy consumption minimization under the response time guarantee\(^2\)
  - Energy management for each virtual machine\(^3\)

Motivation

- The significant growing on the number of cores per system
  - Develop or deploy multi-core platforms in an effective system design
- The consideration of the DVS functionality in virtual cores
  - Avoid timing constraint violations
Motivation (cont.)

- Minimize the energy consumption under the requirements of virtual cores
  - Model the DVS needs of a virtual core
  - Map the DVS needs of virtual cores into the DVS settings of physical cores
  - Design a scheduling algorithm to satisfy the needs of virtual cores

→ Propose a power-aware virtualization system
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Power-Aware Virtualization System

- Virtual core model
- Hypervisor functionalities
  - Minimize the energy consumption
  - Manage virtual cores with virtual core functions
  - Guarantee the needs of each virtual cores
- Application extension
  - Multi-core extension
  - Implementation considerations
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Virtual Core Model

- A virtual core
  - In user’s view: an independent physical core
    - execute its own tasks with a given virtual frequency
  - In hypervisor’s view: a single server
    - have task execution transparent to the hypervisor

- Virtual core model on a Constant-Bandwidth-Server-based model with Hard-Reservation (CBS-HR)$^1,2$
  - $C$: Maximum budget (execution cycles)
  - $T$: Replenish period (time)
  - $F_v$: User-specified operating frequency of a core

- The behavior of a virtual core
  - Guarantee $C$ execution cycles during every replenish period $T$
  - Determine the $C$ and $T$ according to the user requirement

Virtual Core Model (cont.)

- Determine the maximum budget $C$ for a given replenish period $T$ from the desired operating frequency $F_v$
  \[
  \frac{C}{T} \geq F_v \quad C \geq F_v \cdot T
  \]

- Determine the replenish period $T$ to meet the timing constraints
  - Real-time guarantee for real-time tasks
  - Response time guarantee for non-real-time tasks
Virtual Core Model – Real Time Constraint

Lemma 1

- Given a set of real-time tasks \( \{\tau_1, \tau_2, ..., \tau_n\} \), the timing constraints of these tasks can be met with a virtual core by setting

\[
T = \gcd(p_1, p_2, ..., p_n) \quad \& \quad C \geq \sum_{i=1}^{n} \frac{c_i}{p_i} \times T
\]

if the real-time tasks are scheduled under the Earliest-Deadline-First (EDF) scheduling policy

\[
C_k = \left\lfloor \frac{p_k}{T} \right\rfloor C = \frac{p_k}{T} \sum_{i=1}^{n} \frac{c_i}{p_i}
\]

Virtual Core Model – Response Time Constraint

Lemma 2

Given a tolerable response time delay $\sigma$, the delay of the response time on the virtual core can be no more than $\sigma$ if

$$T \leq \frac{\sigma}{1 - F_v / F_p} \quad \text{and} \quad C \geq F_v \cdot T$$

$$R_d = (T - \frac{C}{F_p}) + \frac{\Delta \cdot F_v}{F_p} - \Delta$$
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
DVS-Based Energy Efficient Policy

- To minimize the energy consumption
  - Provide functionalities for users to utilize virtual cores
  - Support an admission control mechanism to guarantee the needs of virtual cores
- Scale up the frequency of a virtual core
  - Determine a physical frequency to serve the user’s requirement
- Scale down the frequency of a virtual core
  - Determine a physical frequency to minimize the energy consumption
Hypervisor Functionalities

- Virtual core creation function
  - Input: required frequency $f$, required timing constraint
  - Output: a new virtual core $\{C, T, F_v\}$

- Virtual core deletion function
  - Input: a selected virtual core
  - Output: none

- Virtual core adjustment function
  - Input: a selected virtual core, new required frequency $f$
  - Output: the selected virtual core with new $\{C, T, F_v\}$

\[
F_v = f \\
T = \gcd(p_1, p_2, \ldots, p_n) \\
\text{or } T = \frac{\sigma}{1 - F_v / F_p} \\
C = F_v \cdot T
\]
Hypervisor Functionalities (cont.)

- The remaining utilization $U_c$ of the physical core
  - $U_c = 100 - \sum_{i=1}^{n} U_{v,i}$
- Admission control mechanism
  - When the hypervisor creates a new virtual core or adjusts the frequency of an existing virtual core
  - When the physical frequency is changed

If $U_c \geq 0$ then accept the request, else reject the request.
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Multi-Core Extension

- Map multiple virtual cores into a single physical core
- Maintain the remaining utilization for each physical core
  - Assign virtual cores to physical cores according to the utilization of the virtual core with the best-fit policy
Implementation Considerations

- Implementation overhead
  - Types
    - Management overhead ($O_m$): the duties of the hypervisor
      - A fixed utilization added into the utilization of each virtual core
    - Context switch overhead ($O_s$): the context switch between the executions of two virtual cores
      - A fixed number of execution cycles added into the maximum budget of each virtual core
  - Effects
    - The tradeoff between the response time guarantee and the overhead
    - The modification in the virtual core model
Implementation Considerations (cont.)

- The relationship between the replenish period $T$ and the implementation overhead
  - Extra execution time for a virtual core

\[ T \leq \frac{\sigma}{1 - \frac{F_v}{F_p}} \]

- The modification in the virtual core model
  - Add extra execution cycles into the maximum budget $C$

\[ C' = C + O_s + O_m \cdot T \cdot F_p \]
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Implementation

- Implementation target
  - Hypervisor: L4:Fiasco\(^1\) and L4Env\(^2\)
  - Guest operating systems: L4Linux\(^3\) and μC/OS II

- Hypervisor modifications
  - Virtual core model (CBS-HR)
  - Virtual core functions server
  - Energy-efficient policy server
  - DVS functionality server
  - L4 system call extension

[2] L4Env - An Environment for L4 Applications, Operating Systems Research Group Technische Universität Dresden, drops@os.inf.tu-dresden.de
Implementation (cont.)

- L4Linux
- μC/OS II
- μC/OS II
- DVS Module
- Policy
- Virtual Core Functions Server
- L4 System Call
- Virtual Core
- Scheduler
- Virtual Core Functions
- Ready Queue
- Update
- Schedule
- Timer Handler
- L4Env
- L4:Fiasco

Note: The diagram illustrates the components and their relationships within the context of Guest Operating Systems.
Evaluation Environment

- Platform: TI Davinci DVEVM
  - MPU core: ARM926ejs
  - Memory: 256MB
  - Test pins for MPU power measurement
  - A 64-bit hardware timer for counting

- Power measure: Agilent 34970A

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Power (mw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>297</td>
<td>32.40</td>
</tr>
<tr>
<td>283.5</td>
<td>30.93</td>
</tr>
<tr>
<td>270</td>
<td>29.45</td>
</tr>
<tr>
<td>256.5</td>
<td>27.98</td>
</tr>
<tr>
<td>243</td>
<td>26.51</td>
</tr>
<tr>
<td>229.5</td>
<td>25.04</td>
</tr>
<tr>
<td>216</td>
<td>23.56</td>
</tr>
<tr>
<td>202.5</td>
<td>22.09</td>
</tr>
</tbody>
</table>
Evaluation Results

- Energy-efficient policy
  - Check the remaining utilization of the physical core every time it changed
  - Scale the physical frequency to the one which satisfies the user’s requirement

<table>
<thead>
<tr>
<th>Virtual Core</th>
<th>OS</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 L4Linux</td>
<td>Booting of the L4Linux</td>
</tr>
<tr>
<td>2</td>
<td>1 μC/OS II</td>
<td>Mathematics program</td>
</tr>
<tr>
<td>3</td>
<td>1 μC/OS II</td>
<td>Sorting program</td>
</tr>
</tbody>
</table>

The Workloads of Virtual Cores

Power Trace

The Workloads of Virtual Cores

Time (ms)
### Evaluation Results (cont.)

**The overhead**

- **Implementation**

<table>
<thead>
<tr>
<th>Virtual Core</th>
<th>OS</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 L4Linux</td>
<td>Booting of the L4Linux</td>
</tr>
<tr>
<td>2</td>
<td>1 μC/OS II</td>
<td>Mathematics program</td>
</tr>
<tr>
<td>3</td>
<td>1 μC/OS II</td>
<td>Sorting program</td>
</tr>
</tbody>
</table>

**Overhead of Virtual Core Implementation**

<table>
<thead>
<tr>
<th></th>
<th>Original Fiasco</th>
<th>Virtual Core Model</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time (ms)</td>
<td>8536.0755</td>
<td>8948.8492</td>
<td>412.7737 (4.84%)</td>
</tr>
</tbody>
</table>

**Virtual core functions**

**The Overhead of Virtual Core Functions**

<table>
<thead>
<tr>
<th></th>
<th>Creation</th>
<th>Deletion</th>
<th>Adjustment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time (ms)</td>
<td>6.606</td>
<td>6.603</td>
<td>7.131</td>
</tr>
</tbody>
</table>

- Performance degradation less than 5%
Outline

- Introduction
- Power-Aware Virtualization System
  - Virtual Core Model
  - Hypervisor Functionalities
  - Application Extension
- Implementation & Evaluation
- Conclusion
Conclusion

- A power-aware virtualization environment
  - Propose a virtual core model to map the DVS needs and guarantee the requirements of virtual cores
  - Propose a scheduling mechanism and an admission control to emulate the needs of virtual cores

- Future Research
  - Explore the energy management mechanism of other hardware components for virtualization, such as main memory
  - Study the management and resource allocation designs that map a single virtual core to multiple physical cores
Q & A

- Thanks for yours listening!