An Energy-efficient Scheduling Policy for Hypervisors on Asymmetric Multi-core

Ching-Chi Lin, Yi-Chung Chen
Institute of Information Science, Academia Sinica
Department of Computer Science and Information Engineering, National Taiwan University

You-Cheng Syu, Pangfeng Liu
Department of Computer Science and Information Engineering, National Taiwan University
Graduate Institute of Networking and Multimedia, Nation Taiwan University

Jan-Jan Wu
Institute of Information Science, Academia Sinica
Research Center for Information Technology Innovation, Academia Sinica

Po-Wen Cheng, Wei-Te Hsu
Information and Communications Research Laboratories, Industrial Technology Research Institute
Outline

- Introduction
  - Background
  - Motivation
- Virtual Core Scheduling Problem
  - Model
  - Solution
- Evaluation
- Conclusion
Background

- Asymmetric multi-core architecture.
  - Consists of cores with the same ISA but different computing capabilities and power characteristics.
    - ARM (big.LITTLE), Qualcomm (aSMP), Nvidia (vSMP), Samsung, MediaTek…etc.
  - Aim to achieve both performance and energy-efficient.
Motivation

- Design new scheduling algorithm for asymmetric multi-core platforms.
  - Exert the advantage of different types of cores.
  - Maximize power efficiency with modest performance sacrifices.
Hypervisor Scheduler

- Assigns the virtual cores to physical cores for execution.
  - Determines the execution order and amount of time assigned to each virtual core according to a scheduling policy.
  - Current solutions
    - Xen - credit-based scheduler
    - KVM - completely fair scheduler
Current Load-balancing Design

Load-balanced Hypervisor Scheduler

Performance Core

Performance Core

Power-efficient Core

Power-efficient Core
Proposed Design

Asymmetry-aware Hypervisor Scheduler

VM$_1$
- vCore$_0$
- vCore$_1$

VM$_2$
- vCore$_2$
- vCore$_3$

Performance Core
- Performance Core
- Power-efficient Core
- Power-efficient Core
Goal

- Design and implement a new hypervisor scheduler for asymmetric multi-core platform.
  - Achieve energy-saving while satisfying the resource requirement of each virtual core.
Assumptions

- The scheduling policy in the guest OS is already asymmetry-aware.
- The hypervisor is aware of the frequency of each virtual core.
Outline

- Introduction
  - Background
  - Motivation
- Virtual Core Scheduling Problem
  - Model
  - Solution
- Evaluation
- Conclusion
Virtual Core Scheduling Problem

- For every time period, given the operating frequency of each virtual core, the scheduler has to generate a scheduling plan such that
  - The power consumption is minimized.
  - Satisfy the resource requirements of virtual cores.
Scheduling plan

- The amount of time each virtual core should run on each physical core.

- The execution order of virtual cores on each physical core.
Power Model

- Relation between power consumption, core frequency, and load.
  - bzip2
Computing Resource

- Number of CPU cycles.
  - Multiplying the frequency by time.

- Our scheduling plan must satisfy the resource requirement of each virtual core.
  - Unless “fully utilized”.
    - Resource required from vCPUs > resources provided by pCPUs.
Optimization Problem

- **Objective function:**
  
  \[
  \min \left( \sum_{i=1}^{n} \text{Power}_i \right)
  \]

  - \(n\): number of physical core

- **Generate a set of \(a_{i,j}\).**
  - \(a_{i,j}\): the amount of time executing virtual core \(j\) on physical core \(i\) in a time interval.
  - Some constraints.
Solution

- Three phase scheduling
  - Phase 1: generate the amount of time each virtual core should run on physical cores.
  - Phase 2 & 3: determine the execution order of virtual cores on a physical core.
Phase 1

- Generate the amount of “time slice” each virtual core should run on physical cores.
- Linear programming
  - Greedy heuristic
    - Assign workloads to the most energy-efficient physical core with available resources.

<table>
<thead>
<tr>
<th></th>
<th>pCPU 1</th>
<th>pCPU 2</th>
<th>pCPU 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vCPU 0</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>vCPU 1</td>
<td>40</td>
<td>10</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>vCPU 2</td>
<td>0</td>
<td>50</td>
<td>30</td>
<td>...</td>
</tr>
</tbody>
</table>
Phase 2

- Determine the execution order of virtual cores on a physical core according to the result of phase 1.
- “Open Shop Scheduling Problem”
  - Can be solved in polynomial time if jobs can preempt each other.
Phase 3

- Reorder the *Execution Slices* to reduce migration overhead.

<table>
<thead>
<tr>
<th></th>
<th>$t_{100}$</th>
<th>$t_4$</th>
<th>$t_3$</th>
<th>$t_2$</th>
<th>$t_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>...</td>
<td>$T_4$</td>
<td>$x$</td>
<td>$x$</td>
<td>$T_4$</td>
</tr>
<tr>
<td>$x$</td>
<td>...</td>
<td>$x$</td>
<td>$T_3$</td>
<td>$x$</td>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>...</td>
<td>$T_2$</td>
<td>$T_1$</td>
<td>$T_4$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>...</td>
<td>$T_3$</td>
<td>$T_2$</td>
<td>$T_1$</td>
<td>$T_1$</td>
</tr>
</tbody>
</table>

- **Core$_0$ (Performance)**
- **Core$_1$ (Performance)**
- **Core$_2$ (Power-efficient)**
- **Core$_3$ (Power-efficient)**
Outline

• Introduction
  ◦ Background
  ◦ Motivation

• Virtual Core Scheduling Problem
  ◦ Model
  ◦ Solution

• Evaluation

• Conclusion
Experimental Environment

- ARM Juno board
  - 2 performance core A57
  - 4 power-efficient core A53

- Xen 4.5.0-rc
  - Dom0: dual-core VM
  - Dedicate 2 power-efficient to Dom0 VM.

- Workload: Coremark
  - Light, medium, heavy

- Baseline: Xen default credit-based scheduler
Workload Setting

- Two DomU: dual-core VMs.
- Two sets of input:
  - Case 1: Both VMs with light workloads.
  - Case 2: One VM with medium workloads, the other with heavy workloads.
Results

(a) Case 1

(b) Case 2
Results (Cont.)

<table>
<thead>
<tr>
<th></th>
<th>Energy (J)</th>
<th>Time (Sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Asymmetry-aware</td>
<td>4.948</td>
</tr>
<tr>
<td></td>
<td>Credit-based</td>
<td>9.817</td>
</tr>
<tr>
<td>Case 2</td>
<td>Asymmetry-aware</td>
<td>24.775</td>
</tr>
<tr>
<td></td>
<td>Credit-based</td>
<td>34.890</td>
</tr>
</tbody>
</table>

- Case 1: asymmetry-aware method is about 50.4% of that of credit-based method.
- Case 2: asymmetry-aware method uses 70.8% of energy used by the credit-base method.
Outline

- Introduction
  - Background
  - Motivation
- Virtual Core Scheduling Problem
  - Model
  - Solution
- Evaluation
- Conclusion
Conclusion

- We develop an energy-efficient scheduler for asymmetric multi-core platforms.
  - Generates energy-efficient scheduling plans that satisfy virtual core resource requirements.

- Will keep improving the scheduler and the related issues.
Thank you!