Power Management for Virtualized Servers

Dr. Xiaorui Wang

Slides are adapted based on the slides from Yefu Wang and Ripal Nathuji

Recap of Last Week

- Power/thermal-aware utilization control
  - Guarantees real-time deadlines while achieving power savings.
  - Provides double guarantees on system temperature and timeliness.

- Energy management for servers and server clusters
  - Uses DVFS and request batching to reduce energy while achieving desired performance.
  - Uses DVFS / cluster reconfiguration / multiple sleep states to minimize the power consumption of a tier web server cluster.

Power-Aware Server Design

- Concerns of today’s data centers.
  - 1. Service-level agreements (SLAs)
  - 2. Power consumption of servers
  - Operating costs: $30 billion for electricity costs worldwide
  - Failures: power capacity overload, overheating, etc.

- Traditional solutions.
  - Meet SLAs while minimizing power consumption.
    - e.g., transition hardware power state to guarantee response time.
      - Too slow —> speed up CPU.
      - Too fast —> slow down CPU for power savings.

New Challenge: Virtualization

- Several virtual machines share a physical machine.
- Every virtual machine runs its own OS on its own virtual hardware.

Reasons for virtualization:
  - Cost saving
  - Application isolation

Power Management for Virtualized Servers?

- The CPU speed can be lower.
- I need the CPU to be faster.
- Which power state should I use?

Power-Efficient Response Time Guarantees for Virtualized Enterprise Servers

Yefu Wang, Xiaorui Wang, Ming Chen, and Xiaoyun Zhu

University of Tennessee, VMware
Two-layer Control Architecture

Layer 1: Load Balancing Controller (1)
- Goal: all the VMs can have the same relative response time.
- Relative response time of each VM – average = 0
- Actuator: CPU resource allocation (weight)

Layer 2: Response Time Controller (1)
- Assumption: Load balancing controller is working correctly.
  - So all the VMs have the same relative response time.
  - Relative to its maximum allowed value: e.g., 80ms/100ms = 0.8
- Goal: Control response time by adjusting CPU power states.
  - Average relative response time <1
  - Can be extended to control worst-case response time or 90 percentile response time.
- Actuator: DVFS (dynamic voltage and frequency scaling)

Layer 1: Load Balancing Controller (2)
- Modeling
  - Input: Change in weight (n x 1 vector, \( \Delta w(k) \))
  - Output: (Relative response time of a VM) – (average of all VMs) (n x 1 vector, \( e(k) \))
  - Method: System identification
    \[
    e(k) = \sum_{i=1}^{n} A_i e(k-i) + \sum_{i=1}^{n} B_i \Delta w(k-i)
    \]

Layer 1: Load Balancing Controller (3)
- LQR controller design
  - Set points: 0 for each VM
  - Model augmentation: \( v(k) = \Delta w(k-1) + e(k-1) \)
  - Cost function:
    \[
    J = \sum_{k=0}^{\infty} \left( x'(k) Q(k) x(k) + \Delta x'(k) R(k) \Delta x(k) \right)
    \]
  - Controller:
    - Larger Q: faster response to variations; Larger R: less sensitive.
    - Solve the optimization problem to get the feedback matrix \( F \).

Layer 2: Response Time Controller (2)
- Modeling
  - Input: Relative CPU frequency
    - 1 means the highest CPU frequency
  - Output: Average relative response time
  - Method: System identification
    \[
    \Delta r(k) = a_1 \Delta r(k-1) + b_1 \Delta f(k-1)
    \]
  - Controller
    - Set point: <1 (0.88 in experiments)
    - SISO PI (proportional-integral) controller
Coordination between Two Controllers

- Two controllers may affect each other.
  - Decouple them by using different control periods.
  - Control period of the RT controller is selected to be much longer than the settling time of the load balancing controller.
  - So that all the VMs can achieve load balancing (i.e., response time).

Baselines

- OPEN
  - Fixed CPU allocation to each VM
  - CPU uses a fixed frequency
- SIMPLE
  - Traditional power-ware design without VM considerations
  - Fixed CPU allocation
  - Response time controller: the same as the response time controller in Step 2
  - Difference between SIMPLE and two-layer control solution
    - Load balancing

Experimental Setup

- Test-bed
  - A server with Intel Xeon X5365 processor
  - OS: Linux (Fedora Core 8)
  - VM monitor: Xen 3.1
  - 3 VMs running Apache servers on the server.
  - Another server emulates concurrent HTTP clients.
- Control components:
  - Response time monitor: One for each VM
  - CPU resource allocator: Xen Credit Scheduler
  - CPU frequency modulator: Intel's SpeedStep
  - Two controllers
  - Power meter

Exp 1: Load Balancing Only

- An unpredictable workload increase to VM2 from time 300s to 600s.
- All three VMs are controlled to have the same relative response time.
- System with load balancing controller will adapt to the workload change.

Exp 2: Two-layer Control Solution vs. OPEN

- Two-layer control solution saves power.
- OPEN has a response time unnecessarily shorter than required.
- An unpredictable workload increase to VM2 from time 300s to 600s.

Exp 3: Two-layer Control Solution vs. SIMPLE

- An unpredictable workload increase to VM2 from time 300s to 600s.
- In SIMPLE, VM2 violates its response time requirement.
Conclusions

- Power and performance are both important.
- Traditional power-aware solutions do not work for virtualized environments.
- This paper presents a two-layer control solution.
- Guarantees response time while saving power for virtualized servers.

VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems

Ripal Nathuji and Karsten Schwan
Georgia Institute of Technology

Motivations

- Problems of power management in virtualized systems
  - VMs use different existing power management policies that are developed for non-virtualized systems.
  - Need to maintain VM isolation and independence with resource sharing (e.g., DVFS).
  - Provide maximum power management support despite hardware heterogeneity for VM migration purpose.

 VirtualPower provides a framework that transparently supports guest VMs’ power management requests.

Problems and VirtualPower Solutions

- Problem 1: VMs’ power management policies may ask for different numbers of power states?
  - Solution: soft virtualized PM states

- Problem 2: maintain VM isolation and independence?
  - Solution: provides a VPM channel for Dom0 to arbitrate power state change requests and decide hardware power states

- Problem 3: Hardware only has limited power management capabilities (e.g., limited # of DVFS levels)?
  - Solution: hardware scaling + soft scaling + consolidation for maximized support

Problem 1 and Solution: VPM States

- VPM states
  - Virtualized “soft” states
  - Provide consistent view of manageability across migrations
  - No need to change the power management policies of guest VMs

Problem 2: Isolation + Independence

- How to arbitrate the power state change requests from the VMs?
- Conduct power management for one VM without affecting other VMs
Problem 2 Solution: VPM Channels

- Forward VM policy actions to management domain
- Dom0 manages hardware power states to satisfy all the VMs
- No need to modify OS

Problem 3: Limited Hardware PM Benefits

- Guest VMs have conflicts with shared resources.
- Time multiplexing (round robin) has high overhead and not good for multi-core.
- Multiple cores also have correlation that may cause conflicts.

Problem 3 Solution: VPM Mechanisms

- Soft scaling restricts resource allocations
  - Adjust time slices allocated to each VM: half freq → half time slice
  - Soft scaling achieves power saving by putting CPU idle
  - Multiple soft scaled virtual resources can be consolidated
  - Achieves more power savings with degraded performance

VirtualPower Architecture

VirtualPower has three components
- VPM states: to deal with migration across heterogeneous servers
- VPM channel: to deal with isolation and power state translation
- VPM mechanisms: to deal with limited hardware support

VPM State Definitions

<table>
<thead>
<tr>
<th>VPM State</th>
<th>Hardware Freq</th>
<th>Soft (Scheduler)</th>
<th>Work-conserving</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0GHz</td>
<td>2.0GHz</td>
<td>Yes</td>
<td>No</td>
<td>80ms</td>
</tr>
<tr>
<td>2.0GHz</td>
<td>2.0GHz</td>
<td>No</td>
<td>Yes</td>
<td>74ms</td>
</tr>
<tr>
<td>1.0GHz</td>
<td>1.0GHz</td>
<td>No</td>
<td>Yes</td>
<td>56ms</td>
</tr>
<tr>
<td>800MHz</td>
<td>800MHz</td>
<td>No</td>
<td>No</td>
<td>50ms</td>
</tr>
</tbody>
</table>

- Hardware only supports 2 freq levels.
- With soft scaling, VirtualPower can support 5 or more power states.

Evaluation with Different PM Rules

- How VirtualPower can support different rules?
  - Transactional workloads
    - VM policy: monitors transaction processing rate and selects power state based upon “slack”
  - Tiered web service (RUBiS)
    - VM policy: Linux on-demand governor
  - VM consolidation in heterogeneous server systems
Transactional Workloads: Meeting Varying Demands

- Single VM: Obvious power benefits for reduced rates
- Multi-VM/VPM rules can obtain substantial savings across VMs with identical or different demands

RUBiS: Utilizing Different VPM Rules

- More realistic multi-tier web service applications
- Necessary to use different VPM rules for different applications
- VPM rules can be sophisticated: adaptive, complex analysis, learning methods
- Simple’s power efficiency is low because on-demand governor is too slow
- Complex improves power efficiency by smoothing VPM state requests

Consolidation with Heterogeneous Systems (1)

- Three dual core platforms, four deployed VMs
- Heterogeneous systems
- Workloads require full performance of P4 core
- PM-G policy heuristic: utilize more power efficient hardware (Core2)

Consolidation with Heterogeneous Systems (2)

- Migrate two VMs to Core2 system
- Local PM-L policy on Core2 performs soft scaling based upon observed requests
- Soft scaling provides room for further consolidation

Consolidation with Heterogeneous Systems (3)

- The other two VMs are migrated to Core2 system.
- Soft scaling and consolidation can utilize heterogeneous systems for further power savings.

Conclusions

- Power management in virtualized systems
  - Transparently leverage existing application policies
  - Deal with heterogeneity in hardware/manageability
  - Maintain isolation and independence
  - Obtain power savings with VM resource sharing

- Solutions/contributions
  - Virtualized “soft” PM states
  - VPM channels and mechanisms
## Comparison of the Two Papers

<table>
<thead>
<tr>
<th></th>
<th>Two-layer control solution</th>
<th>VirtualPower</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What was proposed?</strong></td>
<td>A unified VM PM policy to replace existing PM solutions</td>
<td>A framework to support VM's own PM policies</td>
</tr>
<tr>
<td><strong>How to conduct power management (resolve conflicts)?</strong></td>
<td>Load balancing to achieve unanimous PM agreement</td>
<td>Satisfy the PM request of each VM using hardware and soft scaling</td>
</tr>
<tr>
<td><strong>Knobs to manage power</strong></td>
<td>DVFS, CPU resource</td>
<td>DVFS, CPU resource (soft scaling), consolidation</td>
</tr>
<tr>
<td><strong>Power management policy</strong></td>
<td>Feedback control to minimize power for response time guarantees</td>
<td>No policy was proposed. Can support various existing policies.</td>
</tr>
<tr>
<td><strong>Considered multi-core?</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>