CloudPowerCap: Integrating Power Budget and Resource Management across a Virtualized Server Cluster

Yong Fu, Anne Holler, Chenyang Lu
Yong & Chenyang are from WUSTL
Anne is from VMware
Background: Introduction of Server Power Caps

- **Problem**
  - Datacenter **server racks** on ave 40% **underutilized** due to power issues
    - **Empty slots**: Rack slots are left empty
    - **Dark Power**: Power allocated to servers in rack cannot be fully utilized
  - Underutilization incurs substantial ongoing **costs**
    - **Private**: amortized capital for provisioned power/cooling 18% -> 7%
    - **Hosting**: often charge for provisioned not used power -> 40%

- **Root cause**
  - Server labeled **max** power draw >> **peak** power draw achievable

- **Solution**
  - Vendors [e.g., HP, Dell, IBM] are shipping support for **server power caps**
    - Hardware/firmware guarantees peak power draw <= power cap setting
    - Operator sets server power caps so sum in rack <= rack power budget
CloudPowerCap: Manage Server Power Caps

• Server power caps present new management opportunity
  • Avoid admin maintaining each server’s share of rack power budget
  • Support safe rack power budget reallocation btw hosts at runtime
    • Power cap setting: BMC operation, host responds in < 1 ms

• CloudPowerCap idea:
  • Automate allocating rack power budget to servers in the rack
    • Dynamically reset host power caps as needed by host workloads
    • Challenge: How to determine workloads’ needs?
CloudPowerCap: Determining Workload Needs

- We observe that many datacenters
  - Use their racks of hosts to run workloads in virtual machines [VMs]
  - Run cloud resource management [CRM] which determines workload needs
    - Respect for constraints
      - to handle VM operating requirements & user business rules
    - Admission-controlled resource reservations
      - to support service-level agreements [SLAs]
    - Computation of and scheduling for resource entitlements
      - to achieve fair-share scheduling
    - Entitlement load-balancing
      - to maintain headroom for bursts in resource demands
    - VM consolidation & power-off excess hosts; power-on when needed
      - to improve power proportionality
- CloudPowerCap approach to determine VM workload needs:
  - Interoperate with CRM system, leverage its model of workload needs
CloudPowerCap: Manage Server Power Caps to meet VM workload needs

• Previous server power cap management work
  • Didn’t include interoperation w/CRM, which lessens its usability
    • Could impact respect for constraints, SLAs, fairness
    • Could reduce burst performance and power proportionality

• CloudPowerCap+CRM -> novel capabilities for power-capped hosts:
  • Improved constraint satisfaction via powercap allocation
  • Reduced entitlement balancing overhead via powercap ent balancing
  • Enhanced power availability for bursts via powercap redistribution

• Note: CloudPowerCap not designed to reduce power consumption
  • Designed to redistribute rack power budget btw hosts to support CRM
  • In some cases, this results in power savings
Server Power Caps impact effective CPU capacity

- Server power consumption roughly proportional to CPU utilization

\[ P_{\text{consumed}} = P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}})U \]

- Worst-case; P-states (DVFS) & C-states reduce power consumption
- Idle power includes non-CPU elements (e.g., memory, network, disk)
- Peak power is that consumed at 100% CPU utilization at peak P-state
- Utilization is computed relative to full CPU capacity at peak P-state

- Setting a host power cap limits worst-case attainable CPU capacity

\[ P_{\text{cap}} = P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}})(C_{\text{capped}} / C_{\text{peak}}) \]

\[ C_{\text{capped}} = C_{\text{peak}}(P_{\text{cap}} - P_{\text{idle}})/(P_{\text{peak}} - P_{\text{idle}}) \]

- Host resource scheduler may get more capacity for given host power cap
Rack Power Budget Resource Trade-offs

- Trade-offs possible in allocating a rack’s power budget to its hosts
  - Consider host comprising 12 CPUs, each 2.9 GHz, as follows:

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Nameplate</th>
<th>Peak</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.8 GHz</td>
<td>96 GB</td>
<td>400 W</td>
<td>320 W</td>
<td>160 W</td>
</tr>
</tbody>
</table>

- Rack w/8 KW power budget can support different maximum capacities
- CloudPowerCap manages racks w/host power cap below peak

<table>
<thead>
<tr>
<th>Power Cap(W)</th>
<th>Num Hosts</th>
<th>CPU Total (GHz)</th>
<th>CPU Ratio</th>
<th>Memory Total (GB)</th>
<th>Memory Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>20</td>
<td>696</td>
<td>1.00</td>
<td>1920</td>
<td>1.00</td>
</tr>
<tr>
<td>320</td>
<td>25</td>
<td>870</td>
<td>1.25</td>
<td>2400</td>
<td>1.25</td>
</tr>
<tr>
<td>285</td>
<td>28</td>
<td>761</td>
<td>1.09</td>
<td>2688</td>
<td>1.40</td>
</tr>
<tr>
<td>250</td>
<td>32</td>
<td>626</td>
<td>0.90</td>
<td>3072</td>
<td>1.60</td>
</tr>
</tbody>
</table>
CloudPowerCap Powercap Distribution Examples

- Powercap Allocation
  - Provide resources to allow CRM to satisfy VM-VM affinity constraint
Background: Powercap Distribution Examples

• Powercap Entitlement Balancing
  • Enable entitlement balancing to enhance response to demand bursts (below)
  • Reduce entitlement balancing overhead by avoiding some migrations
Background: Powercap Distribution Examples

- Powercap Redistribution
  - Provide burst headroom after host power-off during low demand period
CloudPowerCap Design and Implementation

- Design: Interacts with each CRM phase to support its goals
  - Operates on host/VM model used by CRM to determine actions
  - Modifies hosts’ CPU capacity when changing hosts’ power-cap
  - Issues actions in appropriate order wrt other CRM actions
- Implemented w/VMware Distributed Resource Scheduler (DRS)
  - CloudPowerCap could be implemented similarly wrt other CRMs
CloudPowerCap Implementation

- DRS [VMware Technical Journal 2012]
  - Runs every 5 min or on request (e.g., choose VM power-on host)
  - Operates on snapshot representing hosts & VMs under mgmt
  - Recommends actions w/expected impact simulated on snapshot

- For CloudPowerCap interoperation with DRS
  - Set snapshot host CPU capacity to reflect host power cap value
  - Added new action type to change a host’s power cap value
    - Changes snapshot host CPU capacity accordingly
  - Changed DRS to treat CPU capacity as elastic up to peak
    - Subject to available power budget
  - Introduced CloudPowerCap phases to support DRS phases
Experiments

- Run on DRS simulator
  - Characteristics
    - High fidelity; used to develop & test all DRS algorithm features
    - Allows flexibility & precision in expressing scenarios; repeatable
    - Simulates behavior of cluster of specified hosts and VMs
      - Hosts: core count; CPU GHz / core; memory size; idle, peak power
      - VMs: vCPUs & memory size, time series CPU & memory workloads
  - Models:
    - Operation of hosts & VMs, including host hypervisor scheduler
    - Execution of DRS recommendations, updating the cluster state
    - vMotion overhead, incl. memory access patterns, I/O bandwidth
    - CPU & memory given to VMs over time, reported as payload
Experiments

- CloudPowerCap compared with two static strategies from rack budget table slide
  - StaticHigh: Each host power cap statically set to its peak of 320W
    - Maximizes the amount of CPU capacity available for given rack power budget
    - Doesn’t enable using power for more memory & I/O capacity w/lower CPU
  - Static: Each host power cap statically set to 250W
    - Allows more servers to be placed in the rack
    - Allows more memory and I/O capacity w/lower CPU capacity

- CloudPowerCap compared with these static strategies in three scenarios
  - Headroom balancing
    - Supporting DRS headroom balancing while avoiding vMotion overhead
  - Flexible resource capacity
    - Enabling trade-off between powering CPU and memory capacity at runtime
  - Standby host power redistribution
    - Redistributing powered-off host power, running hosts can better handle bursts
Experiment: Headroom Rebalancing

(a) CloudPowerCap

(b) Static
Experiment: Headroom Rebalancing

<table>
<thead>
<tr>
<th></th>
<th>CPU Payload Ratio</th>
<th>Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPC</td>
<td>0.99</td>
<td>0</td>
</tr>
<tr>
<td>Static</td>
<td>0.89</td>
<td>7</td>
</tr>
<tr>
<td>StaticHigh</td>
<td>1.00</td>
<td>0</td>
</tr>
</tbody>
</table>
Experiment: Flexible Resource Capacity

(a) CloudPowerCap

(b) Static
Experiment: Flexible Resource Capacity

<table>
<thead>
<tr>
<th></th>
<th>CPU Ratio</th>
<th>Mem Ratio</th>
<th>Trading Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPC</td>
<td>1.24</td>
<td>1.28</td>
<td>1.00</td>
</tr>
<tr>
<td>Static</td>
<td>1.21</td>
<td>1.28</td>
<td>0.62</td>
</tr>
<tr>
<td>StaticHigh</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Experiment: Standby Host Power Reallocation

<table>
<thead>
<tr>
<th>CPU Payload Ratio</th>
<th>Migration</th>
<th>Power Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPC</td>
<td>1.00</td>
<td>10</td>
</tr>
<tr>
<td>Static</td>
<td>0.98</td>
<td>19</td>
</tr>
<tr>
<td>StaticHigh</td>
<td>1.00</td>
<td>10</td>
</tr>
</tbody>
</table>
Conclusions

• CloudPowerCap
  • Can manage rack power budget \( \geq \) sum of the hosts peak power
  • Effectively by interoperating w/Cloud Resource Management system
    • Reinforces and enhances the operation of the CRM system
    • Supports flexible trade-off between CPU, memory, I/O
Questions?
Implementation: Interoperation w/DRS phases

- Constraint Violation Correction: Powercap Allocation
  - Determine min powercap setting needed on each host for reservations
  - Allocate remaining rack power budget as needed to enable vio correction
  - If correction actions, redivvy rack power budget to rebalance headroom
Implementation: Interoperation w/DRS phases

• Entitlement Balancing: Powercap Entitlement Balancing
  • While NE imbalance > imbalance threshold, use progressive filling
    • Lower powercap of host w/lowest NE, Raise for host w/highest NE
    • Stop powercap entitlement balancing when power cap hits peak
  • Address any remaining NE imbalance w/DRS load balancing phase
Implementation: Interoperation w/DRS phases

- Power Management: Powercap Redistribution
  - If low demand leads DRS to recommend VM consolidation and host power-off
    - CloudPowerCap redistributes powered-off hosts’ power to those remaining on
      - For better burst handling
  - If high demand leads DRS to recommend host power-on and VM distribution
    - CloudPowerCap generates prerequisite host powercap recommendations
      - May involve redistribution of power budget from powered-on hosts