Towards Energy Proportionality for Large-Scale Latency-Critical Workloads

David Lo†*, Liqun Cheng*, Rama Govindaraju*, Luiz André Barroso*, Christos Kozyrakis†

† Stanford University * Google Inc.
Motivation

- **Energy proportionality**
  - Servers are far less energy efficient at low and medium utilizations
  - Servers are underutilized due to diurnal load patterns

- **Large-scale latency-critical workloads**
  - Web search, social networking, etc.
  - Strict guarantees on tail latency and workload complexity precludes previous power management techniques
Executive Summary

- Energy waste is caused by overachieving on performance

- **Solution: Match power to Service Level Objective (SLO)**
  - End-to-end SLO latency monitoring
  - Fine-grain power saving mechanism (i.e. RAPL)

- Built dynamic controller for large-scale latency-critical workloads
  - 20-30% power savings on production Google search without SLO violations
Outline

- Energy proportionality vs. latency-critical workloads
- Recovering energy proportionality: iso-latency
- PEGASUS: QoS aware dynamic controller
Energy proportionality vs. latency-critical workloads

The case for latency-aware fine-grain power management
OLDI workloads

- On-line Data Intensive (OLDI) workloads are user-facing workloads that mine massive datasets across many servers
- Strict Service Level Objectives (SLO): e.g. 99%-ile tail latency is 5ms
- High fan-out with large distributed state
- Extremely challenging to perform power management

Workload we evaluate on:

- search: Query serving portion of production Google search
Search topology

Q

Front end

Web Server

Back end

Parent

…

Parent

…

Parent

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf

Leaf
The challenge of energy proportionality

![Graph showing energy proportionality](image)

- Measured
- Ideal EP
- Dynamic EP

1.5x power of dynamic EP

% of peak power vs. % of peak cluster load

Search full system power
The need for energy proportionality

- Diurnal variation in cluster load and power for search across a 24 hour period
- Cluster not fully utilized half the time
- Gap between measured power and EP curves represent potential savings
Previous cluster-level power management

- Consolidate load on fewer servers during low utilization
  - Issue: state of OLDI applications cannot fit on fewer servers

- Use very low power idle modes
  - Issue: OLDI request rate is always too high, e.g. >1k requests/sec

- Batch requests to form long enough idle periods
  - Issue: OLDI applications cannot tolerate msec exit times and batching delays
Previous machine-level power management

- CPU utilization based DVFS
  - Changes p-states based on CPU utilization

**Issue:** causes SLO violations
Weakness of current DVFS schemes

- CPU utilization is a poor proxy for workload latency
- To meet SLO, must be latency-aware

Need to rethink approach!
- New policy
- New control mechanism
- New controller
Recovering energy proportionality: iso-latency

Trading end-to-end latency slack for immense power savings
Motivating assumption

- Beating the end-to-end SLO is no better than meeting it
  - The end-user only cares if the web page takes a long time to load
  - If the page loads in 0.25sec vs. 0.50sec, user does not notice
Latency opportunities

![Graph showing search latency vs. cluster load with significant latency slack!](image)

Overall query latency

% of maximum cluster load

0% 20% 40% 60% 80% 100%

Significant latency slack!
Iso-latency power management

- **Key idea:** Trade end-to-end latency slack for power savings
- Use power management mechanisms to keep the workload performing just well enough to avoid SLO violations
  - Need end-to-end latency feedback from workload
    - Most OLDI workloads have ways of measuring this
  - Need fine-grained power management mechanisms
Problem: p-states are not fine grained

Latency vs. cluster load for various p-states

Overall query latency

% of maximum cluster load

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

- Cannot save power here
- 10-20% wasted power

p-0
p-1
p-2
Solution: RAPL

- RAPL: Running Average Power Limit
- Fine-grained: power limit increments as small as 0.125W
- Fast: <1ms delay to apply new limit
- Effective: Dynamic Voltage Frequency Scaling (DVFS) behind the scenes to meet the power limit
  - More fine-grained than p-states
  - Can even modulate between multiples of base clock frequencies
Advantages of fine-grain control

Latency vs. load for various p-states

New RAPL states

New operating points

Overall query latency

% of maximum cluster load

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Iso-latency potential: power

- Close to ideal EP
- Beat dynamic EP!
Iso-latency potential: power savings

20-30% savings
Power and Energy Gains Automatically Saved from Underutilized Systems

QoS aware dynamic controller
PEGASUS description

- Real-time dynamic controller for iso-latency
- Use RAPL as knob for power
- Measures latency slack and sets uniform power limit across all servers
- Power is set by workload specific policy
Example PEGASUS policy for search

- **L** = Measured instant latency
- **T** = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

### Action

<table>
<thead>
<tr>
<th><strong>L</strong></th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60T</td>
<td>Lower power by 7%</td>
</tr>
<tr>
<td>0.85T</td>
<td>Lower power by 1%</td>
</tr>
<tr>
<td>1.00T</td>
<td>Keep current power</td>
</tr>
<tr>
<td>1.35T</td>
<td>Increase power by 7%</td>
</tr>
<tr>
<td></td>
<td>Set max power</td>
</tr>
</tbody>
</table>

**Notes:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Legend:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Figure:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization

**Graph:**

- $L$ = Measured instant latency
- $T$ = SLO target
- Use instant latency for quick corrections
- Violating SLO latency triggers fail-safe
- Constants determined through empirical optimization
Evaluation methodology

- **Workload parameters**
  - **SLO metric**: 30 second average latency
  - Traffic pattern and user queries derived from anonymized search logs
  - Index derived from production search index

- **Evaluate on several cluster sizes**
  - **Small**: tens of machines, use full 24hr trace
  - **Production**: thousands of machines, use 12hr portion

- **Measure full cluster power and SLO latency**
Small cluster results: power over time

Achieve dynamic energy proportionality!
Small cluster results: power comparison

PEGASUS power savings

30% power savings half the time

Savings over baseline (%) vs. Time
Production cluster results: power over time

Closes EP gap, but not to the same extent as small cluster
Production cluster results: power comparison

PEGASUS power savings

Savings over baseline (%) vs. Time

10-20% power savings
Improving PEGASUS scalability

- Production cluster sees “tail at scale” for server utilization
  - At peak load, 0.2% nodes at 100% load while 50% nodes at <85% load
  - Caused by popular queries hitting a few shards
  - **Issue:** Hot nodes set lower bound on power limits for everyone

- **Idea:** hierarchical control
  - **Global:** sets latency targets instead of power limits
  - **Local:** decides amount of power needed to meet target latency
Hierarchical PEGASUS design

Local latency feedback

Local latency targets

Application Performance Monitor

PEGASUS controller
Estimated hierarchical PEGASUS results

Can save power on less-loaded nodes

Dynamic EP recovered
Conclusion

- Halfway there to fully energy proportional systems
- **Iso-latency**: Use SLO metrics and fine-grain power control
  - Save up to 30% power
  - Meet/exceed energy proportionality targets
- **PEGASUS achieves iso-latency benefits**
  - Up to 20% savings on production cluster
  - Be aware of tail at scale effects