PACMan: Performance Aware Virtual Machine Consolidation

Alan Roytman (UCLA)
Joint work with:
Aman Kansal (Microsoft Research), Sriram Govindan (Microsoft),
Jie Liu (Microsoft Research), Suman Nath (Microsoft Research)
Motivation

- Server cost is the largest expense for data centers
- Data centers operate at very low utilization
  - Eg. Microsoft: over 34% servers at less than 5% utilization (daily average). US average 4%.
- VM Consolidation increases utilization, decreases idling costs

Idle power = 50 to 70%

Adding more work to active server is more efficient

[Chen et al, NSDI’ 08]
Motivation

- But VM consolidation **degrades performance** due to interference in the memory hierarchy
  - Interference occurs throughout memory hierarchy (e.g., multiple cores can share a cache)

![Diagram showing interference in the memory hierarchy](image)

Motivation

Goal: Consolidate intelligently to trade-off energy efficiency and performance

- How do we minimize resource cost while staying within a performance bound?
  - (e.g., minimize energy consumption or active machines)
- How do we maximize the worst case performance?
  - (e.g., Map–Reduce)
Talk Outline

- Introduction
- Performance Aware Consolidation Manager
  - Performance-Mode: Minimize Energy Under Constraint
  - Energy-Mode: Minimize Maximum Degradation
- Experimental Results
- Conclusions and Future Work
First Problem: Perf–Mode Example

Each machine incurs a cost of 50 for being active, plus 10 per VM assigned.
Total cost of schedule = 6 \times (50 + 10) = 360
First Problem: Perf–Mode Example

Machine 1

A
50+10+10 = 70

B

Machine 2

C
50+10+10+10 = 80

D

E

Machine 3

F
50+10 = 60

Degradation

A
B
50+10+10 = 70

C
D
E

F
50+10 = 60

Machine 1

A
B
C
50+10+10+10 = 80

Machine 2

D
E
F
50+10+10+10 = 80

✓
Minimize Energy Under Performance Constraint

- We have $n$ VMs, along with a degradation constraint $D \geq 1$, machines with $k$ cores
- We are given feasible sets $|S| \leq k$ (all VMs experience degradation at most $D$)
- Each set $S$ has a cost $w(S)$ (e.g., energy)
- Goal: $\min_{\text{partitions}} \sum_S w(S)$
We give a polynomial time optimal solution for the two-core case.

Bad news: for $k \geq 3$ cores, this problem is NP-Complete.

Good news: we design and analyze an approximation algorithm with approximation ratio $\alpha = H_k \approx \ln(k)$.

We can solve it close to optimal!
Multi-Core Case

- This problem is approximable within a factor 
  \[ \alpha = H_k = \sum_{i=1}^{k} \frac{1}{i} \approx \ln(k) \]

- This means, for all inputs: 
  \[ w(ALG) \leq H_k w(OPT) \]

- Proof similar to the \( k \)-Set Cover Problem

- Need two assumptions:
  
  **Closure Under Subsets:** \( S \) feasible implies any subset \( T \subseteq S \) is feasible

  **Monotonicity:** If \( S \subseteq T \), then \( w(S) \leq w(T) \)
First consider the case when all costs are 1 (minimizing cost = minimizing # machines)

Algorithm:
- Sort sets (ascending order) according to $\frac{1}{|S|}$
- Greedily pick disjoint sets going down the list
Algorithm Example

Suppose there are $n = 5$ VMs and $k = 3$ cores

<table>
<thead>
<tr>
<th>$S$</th>
<th>{A,B}</th>
<th>{A,C}</th>
<th>{B,C}</th>
<th>{A,B,C}</th>
<th>{D,E}</th>
<th>{A}</th>
<th>{B}</th>
<th>{C}</th>
<th>{A,B,D}</th>
<th>{A,B,E}</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{</td>
<td>S</td>
<td>}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{2}$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

Sorted order:
{A,B,C} {A,B} {A,C} {B,C} {D,E} {A} {B} {C}

Solution uses two machines
The proof generalizes to the case when the costs of sets can be arbitrary!

- e.g., $w(S) = c_f + \sum_{j \in S} d_j$, $w(S) = \max_{j \in S} d_j$

New Algorithm:

- Sort sets (ascending order) according to $\frac{w(S)}{|S|}$
- Greedily pick disjoint sets going down the list
We can solve the two-core case optimally and efficiently

For more cores, the problem is NP-Complete

We give an asymptotically tight approximation algorithm with $\alpha \approx \ln(k)$

The algorithm is greedy and easy to implement
Max Degradation

Second Problem: Energy–Mode Example
Minimizing Maximum Degradation

- Input is similar to before: \( n \) VMs, \( m \) machines, \( k \) cores

- For a set \( B \) of VMs, VM \( j \in B \) experiences degradation \( d_j^B \geq 1 \)

- New Objective Function:

\[
\text{Goal: Minimize } \max_{1 \leq i \leq m} \max_{j \in S_i} d_j^{S_i} \quad (S_i \text{ is the set of VMs on server } i)
\]
For two cores, the problem is polynomial-time solvable

We give an inapproximability result for this problem

We give heuristics since the problem is provably difficult to approximate
We implement a greedy heuristic:
- Start from an arbitrary initial schedule
- For all ways of swapping VMs, go to the schedule with smallest sum of maximum degradations
- We set number of swaps to be $G = (k - 1) \cdot (m - 1)$
Experimental Setup

- Small inputs:
  - $n = 16$ VMs, on servers with $k = 4$ cores
  - Can compute optimal solution for small instances

- Large inputs:
  - Up to $n = 1000$ VMs, on servers with $k = 4$ cores
  - Compare solutions against a lower bound

- Use real–world degradations with SPEC CPU 2006 applications (lbm, soplex, povray, sjeng)
Experiments: Perf–Mode (Small Inputs)

- We use costs $w(S) = c_f + \sum_{j \in S} d_j^S$, where $c_f = 4$
- Comparison against OPT
- Naïve leaves every other core empty, which is the current practice [Mars–Tang–Hundt–Skadron–Soffa 2011]
Experiments: Perf-Mode (Core Use)
Experiments: Perf–Mode (Large Inputs)

- Comparison against lower bound

![Graph showing energy overhead vs number of VMs for Proposed and Naïve methods. The Proposed method shows a lower energy overhead compared to the Naïve method across all numbers of VMs.]
Experiments: Energy-Mode (Small Inputs)

- Comparison against OPT
- Up to $G = (k - 1) \cdot (m - 1) = 9$ swaps
- Naïve solution randomly places VMs, error bars show standard deviation for 10 runs
Experiments: Energy-Mode (Large Inputs)

- Reduction in degradation relative to naïve solution
- Up to 1000 VMs
Amortized cost calculation for data centers

22% reduction in costs when comparing Performance-Mode algorithm to current practice

For 10MW data centers, costs are reduced from $2.8M to $2.2M per month (costs are related to energy expenditure)

James Hamilton estimate, Reference: http://perspectives.mvdirona.com/content/binary/OverallDataCenterCostAmortization.xlsx
Related Work

- [Jiang–Shen–Chen–Tripathi 2008]
  - Consider minimizing sum of degradations
  - 2-core case is poly-time solvable
  - $k$-core is NP-Complete for $k \geq 3$ (give heuristics)

- [Tian–Jiang–Shen 2009]
  - Consider different length tasks, allow migrations

- [Jiang–Tian–Shen 2010]
  - Proactive co-scheduling, heuristic runtime scheduler
Conclusion

- Give a provably near-optimal algorithm such that resource waste is minimized.
- Important for energy minimization to consider cache interference.
- Even small percentage improvement can have huge practical impact.
Future Work

- Energy–Mode: consider variable number of swaps while incurring cost for each swap
- Consider online versions of all variants
- Perform more experiments on real data centers
Thank You!

Questions?