Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments

Qi Zhang[1], Mohamed Faten Zhani[1], Shuo Zhang[3] Quanyan Zhu[2], Raouf Boutaba[1] and Joseph L. Hellerstein[4]

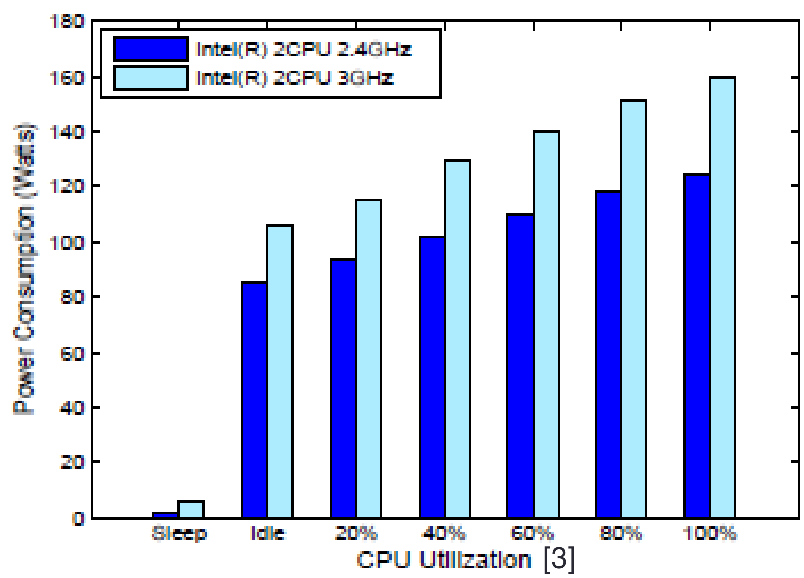
[1]University of Waterloo,
[2]University of Illinois at Urbana Champaign
[3]National University of Defense Technology,
[4]Google

ACM International Conference on Autonomic Computing (ICAC) 2012

Introduction

- Energy cost is an important concern of cloud providers
  - Accounts for 12 % of data center operational cost in 2010 [1]
  - Government policies for building energy-efficient (i.e. “Green”) computing platform

- Turning off servers is an effective way to minimize energy cost
  - An idle server consumes as much as 60% of its peak power usage



Related Work

- Dynamic capacity provisioning and load dispatching
  - Estimate the number of servers then distribute requests among them
  - Dynamically adjusting number of servers
- Differences from our work
  - Most of the existing work focuses on provisioning at application level
    - Trade-off between energy savings and scheduling delay
  - Usually do not consider the cost of turning on and off machines
  - Do not consider the fluctuation of electricity prices
  - Consider a single type of resource (i.e., CPU)

Motivation

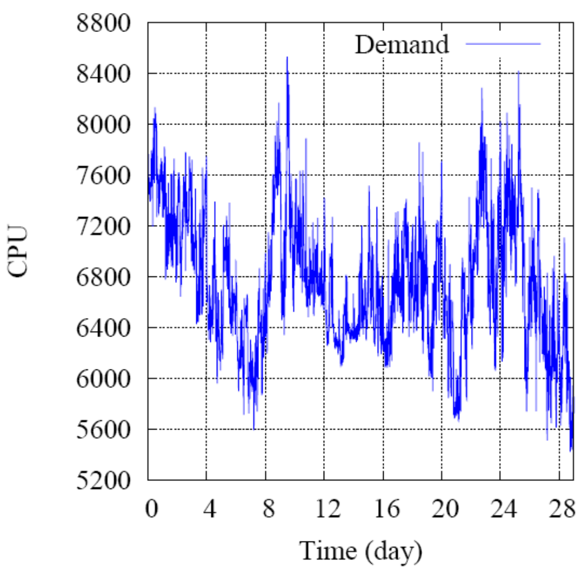
- To dynamically control data center capacity, one must consider the following factors:
  - The task arrival rate
  - Task requirements
    - Memory, cpu and disk
  - The cost of turning on and off servers
    - Wear-tear effect
  - The fluctuating energy prices

Trace Analysis

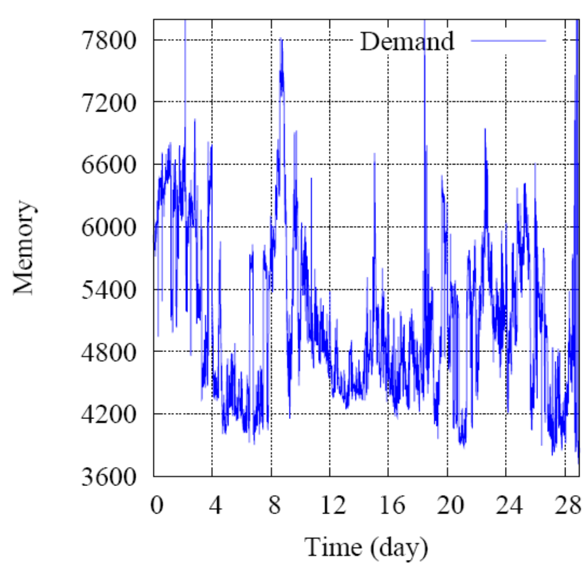
- Google's compute clusters execute millions of tasks on a daily basis
- Workload traces collected from a production compute cluster in Google over 29 days
  - ~ 12,000 machines
  - ~2,012,242 jobs
  - 25,462,157 tasks
- Applications are represented by jobs
- Each Job consists of one or more tasks
  - User-facing jobs: e.g., 3-tier web applications
  - Batch jobs: e.g., MapReduce jobs

Trace Analysis (cont'd)

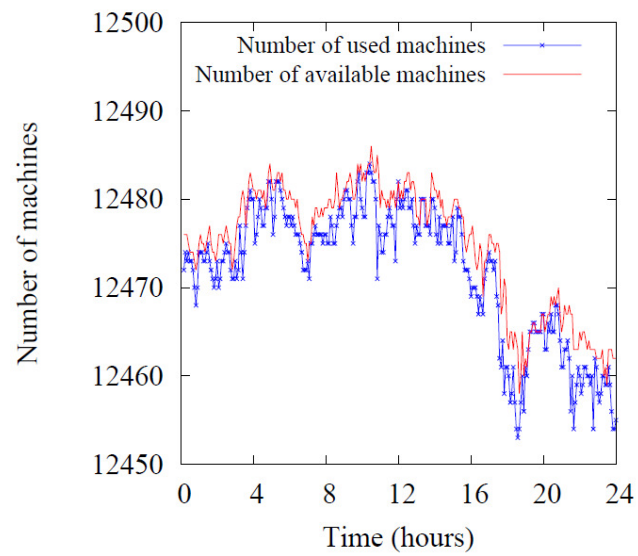
- The fluctuation of resource demand in data centers creates opportunities for dynamically turning on and off servers



CPU Demand over 30 days



Memory Demand over 30 days



Machine availability over 24 hours

Figure: Total resource demand in Google's Cluster Data Set

Trace Analysis (cont'd)

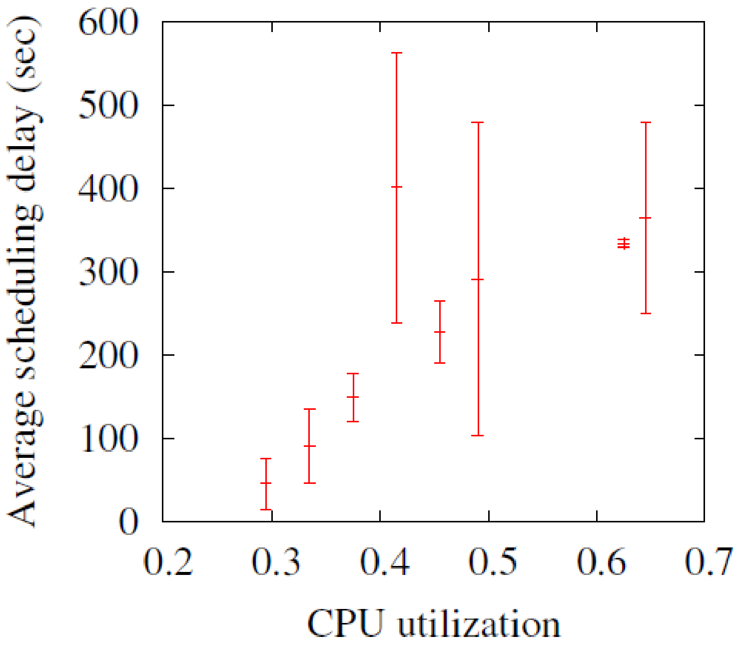
- There is a trade-off between utilization and scheduling delay

- The queuing delay q can be modeled by
  - A linear function

$$q(u) = a \cdot u + b$$

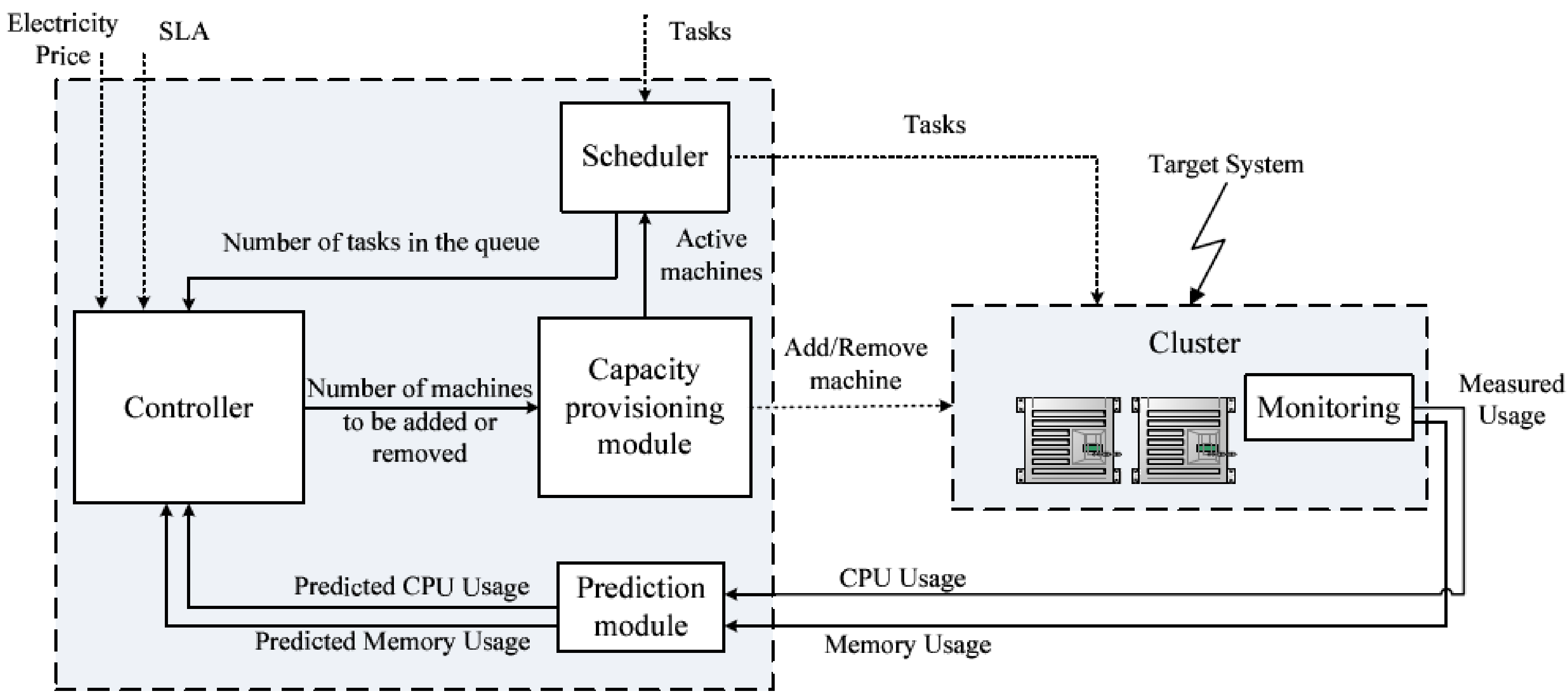
  - A delay function for M/M/1 queuing delay

$$q(u) = a \cdot \frac{u}{1-u} + b$$



Scheduling Delay vs. Utilization

System Architecture



Our Solution

- A control-theoretic solution to the dynamic capacity provisioning problem
  - Predict the demand as well as capacity required to handle the demand
  - Formulate the problem as an optimal control problem
    - Minimizing the total energy cost while meeting requirement
    - Meeting performance requirement measured by the average queuing delay, (estimated as a function of the cluster utilization)

- Derive an expression for the optimal required capacity given electricity prices and queuing delay objectives
    - Solve an optimization problem that minimizes the sum of electricity cost and SLA penalty cost

Optimal Capacity

- Derive an expression for the optimal required capacity given *electricity prices* and *queuing delay* objectives
- Solve an *optimization problem* that minimizes the sum of SLA penalty cost and energy cost :

$$\min_{x_k \in \mathbb{R}+} \underbrace{N_k p^{SLA} \left(q\left(\max_{r \in R} \left\{ \frac{G^r}{x_k C^r} \right\} \right) - \bar{d} \right)^+}_{\text{SLA Penalty Cost}} + \underbrace{p_k^{power} x_k \left(E_{idle} + \sum_{r \in R} \alpha^r \frac{G^r}{x_k C^r} \right)}_{\text{Energy cost}}$$

$$\text{s.t. } 0 \leq x_k \leq M_k$$

⇨ This problem can be solved directly using KKT conditions, assume a M/M/1 queuing model is used for $q(\cdot)$

$$x_k^* = w_k + \sqrt{\frac{N_k p^{SLA} a w_k}{p_k^{power} E_{idle}}}.$$

Designing the Controller

- At time k, predicting the future usage over a prediction window $[k, k+H]$ using time-series method (e.g., ARIMA)
- Compute the optimal capacity x_k^* required at each step in $[k, k+H]$
- Design a control algorithm that tracks the reference value x_k^*. We model it as a linear quadratic control problem:

$$\min_{u_k \in \mathbb{R}} J_k = \sum_{h=1}^{H} \underbrace{Q(e_{k+h|k})^2}_{\text{Tracking Error}} + \underbrace{R(u_{k+h|k})^2}_{\text{Reconfiguration Cost}}$$

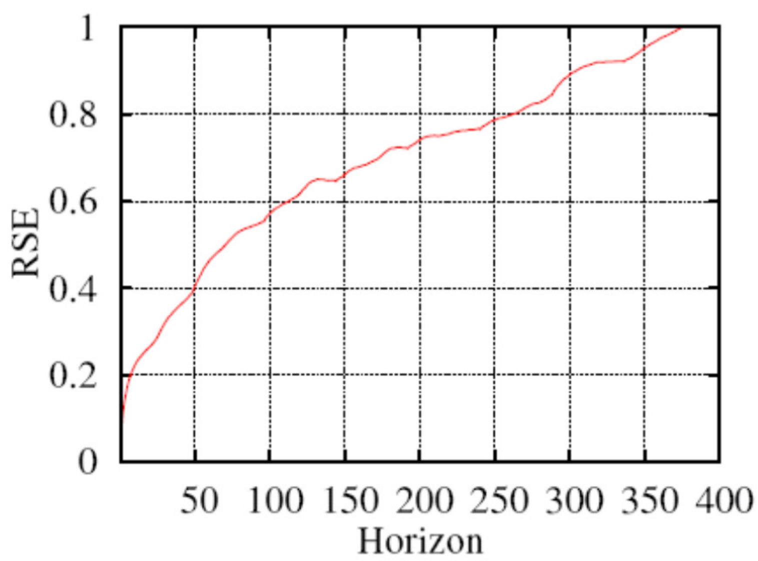
$$\begin{aligned}
\text{s. t.} \quad & x_{k+h+1|k} = x_{k+h|k} + u_{k+h|k}, && \forall 0 \leq h \leq H-1 \\
& e_{k+h|k} = x_{k+h|k} - x^*_{k+h|k}, && \forall 1 \leq h \leq H \\
& 0 \leq x_{k+h|k} \leq N, && \forall 1 \leq h \leq H
\end{aligned}$$

- $u_{k+1|k}$ represents the controller action (number of servers to turn on and off) to be performed at time k

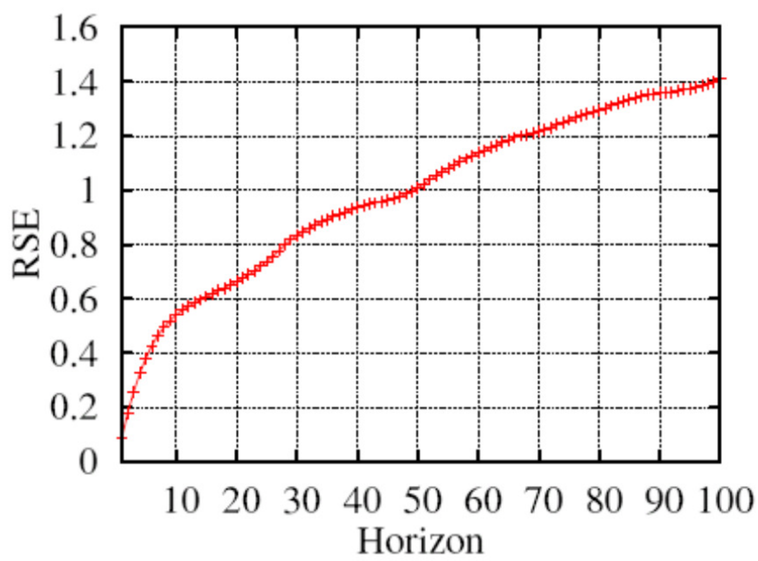
Experiments

- Usage Prediction
  - ARIMA model - ARIMA (2,1,1)
  - Performance metric : *Relative Squared Error* (RSE)

$$RSE_h = \frac{\sum_{k=1}^{T} \left[G_k - G_{k+h|k}\right]^2}{\sum_{k=1}^{T} \left[G_k - \mu\right]^2}$$



(a) CPU

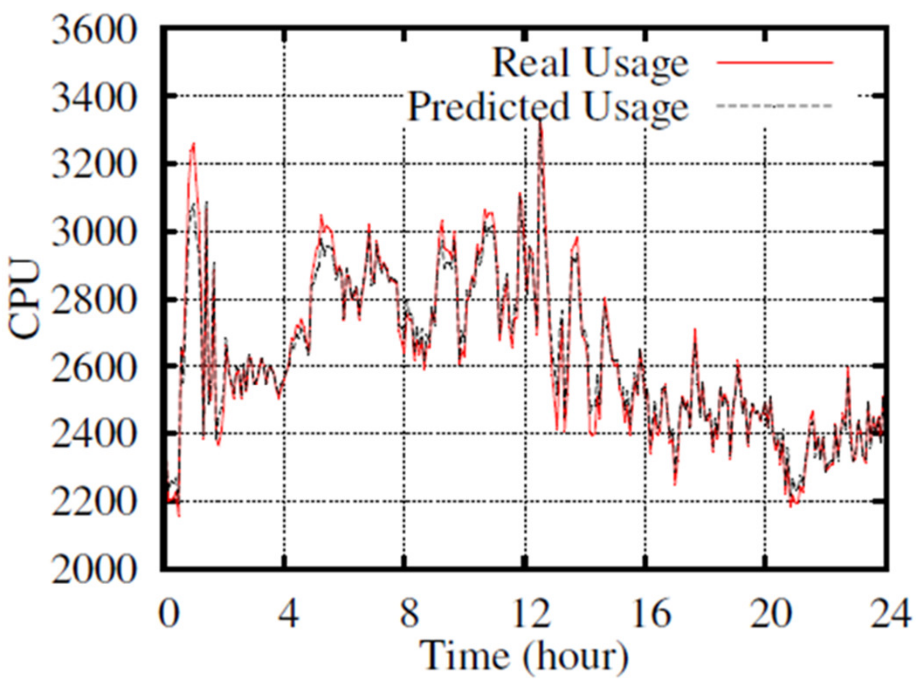


(b) Memory

Experiments

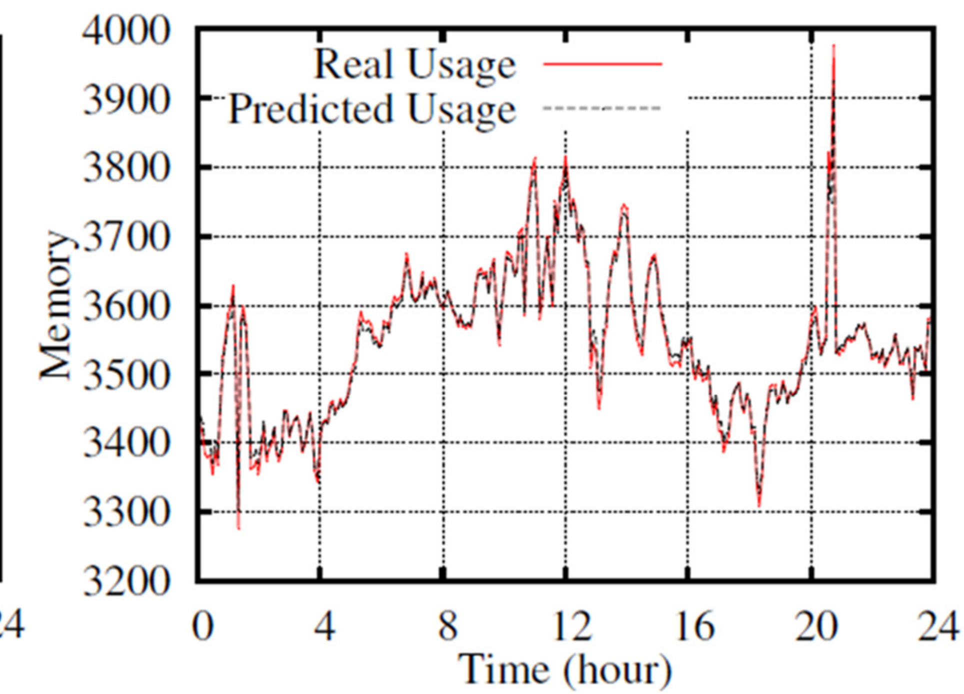
- Usage Prediction
  - ARIMA model ARIMA (2,1,1)
  - One-step prediction

RSE$_1$ = 0.062



CPU

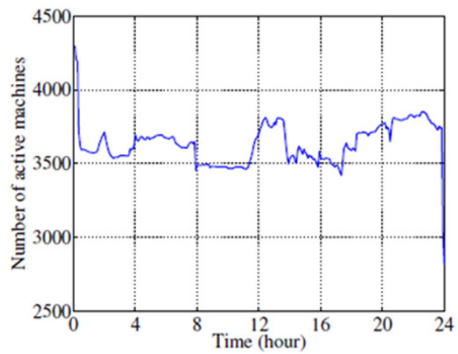
RSE$_1$ = 0.086



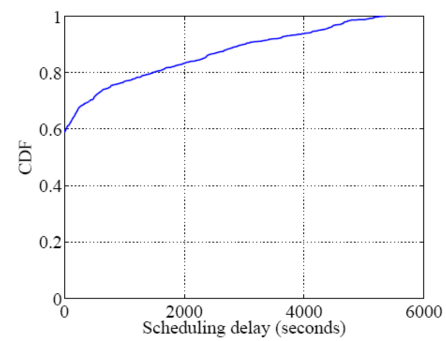
Memory

Experiments

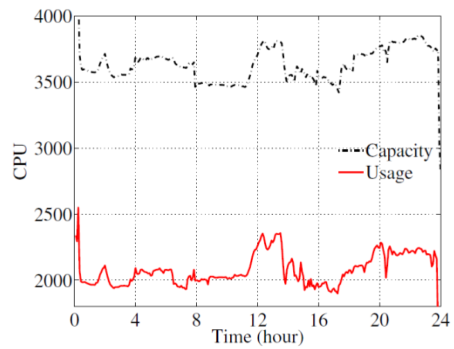
- Simulation Setup
  - Traces from a Google compute cluster
  - Reconfiguration cost R=0.1
  - Desired average scheduling delay: 10 seconds



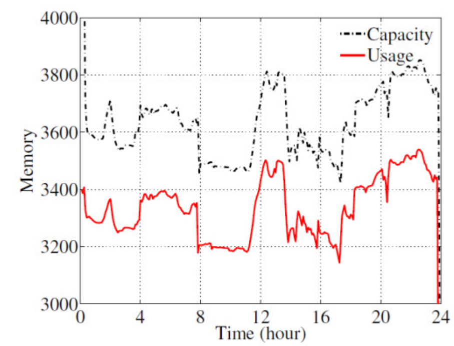
Number of machines over 24 hours



CDF of Task Scheduling Delay



(a) CPU

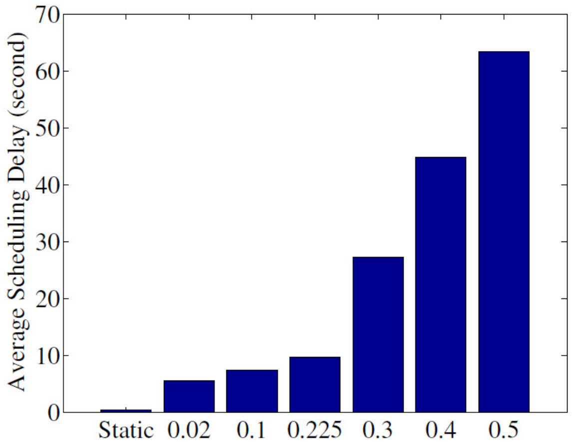


(b) Memory

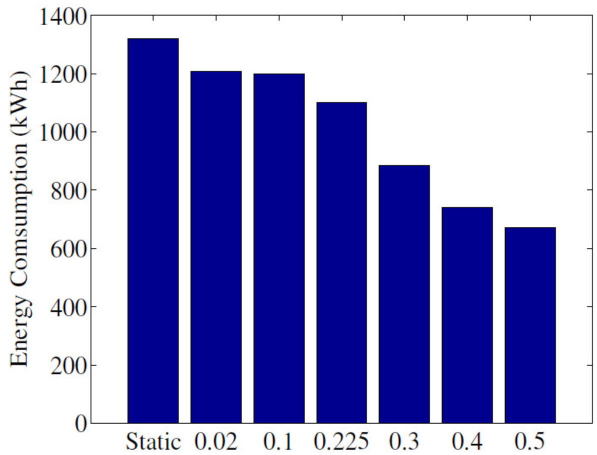
Capacity vs. Actual Resource Usage in the Cluster

Experiments

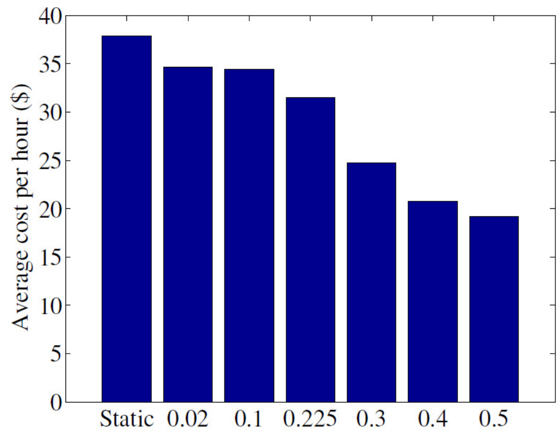
- Effect of the reconfiguration cost on the solution



Average Scheduling Delay as a Function of R



Energy saving as a Function of R



Average cost per hour.

Conclusion

- Dynamic capacity provisioning can achieve substantial energy savings in cloud data centers
- We proposed a control-theoretic solution that dynamically adjusts server allocations according to both demand and resource price
  - ⇨Reduction of18.5 % in energy costs while meeting the SLA requirement in terms of scheduling delay
- Experiments using Google workload traces demonstrate the effectiveness of our approach

Future work

- Heterogeneity is a major challenge for resource management in Cloud computing environments
  - Machines have heterogeneous capacities and capabilities
  - Applications have diverse resource characteristics, priority and performance objectives

➢ How to leverage machine heterogeneity and job arrival patterns to save energy, while meeting job performance objectives?

➢ How to design scheduling algorithms that consider workload heterogeneity?

➢ Many research opportunities exist for designing heterogeneity-aware resource management schemes

Questions?

References

1. Technology research - Gartner Inc. 和 http://www.gartner.com/it/page.jsp?id=1442113
2. http://perspectives.mvdirona.com/
3. Chen et al., Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services, NSDI 2008

Backup Slides

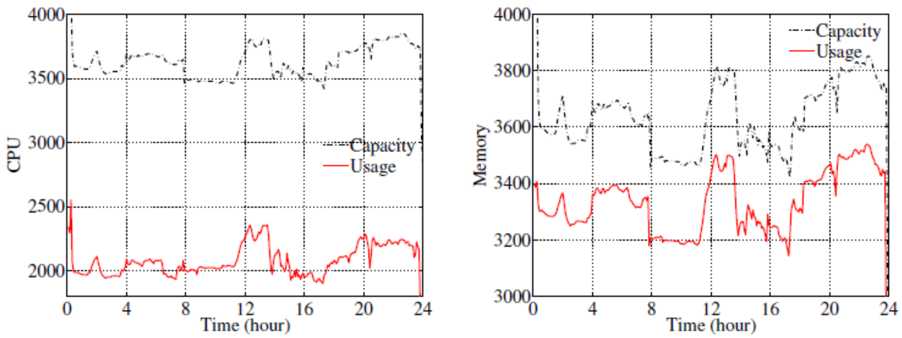
Future work (cont'd)

- Optimizing workload performance and resource efficiency using migration
  - Live migration is a well known technique for online workload management
  - How to use migration effectively given heterogeneous workload characteristics?

Dynamic Energy-Aware Capacity Provisioning

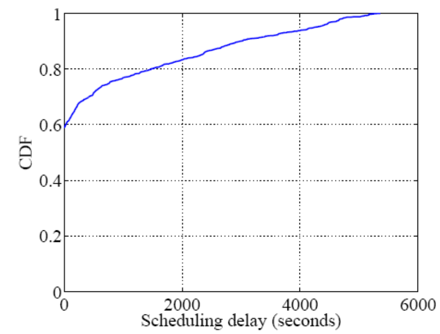
Experiments

- Controller performance (R=0.1)



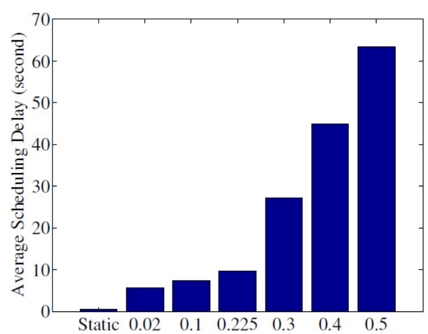
(a) CPU (b) Memory

Capacity vs. Actual Resource Usage in the Cluster

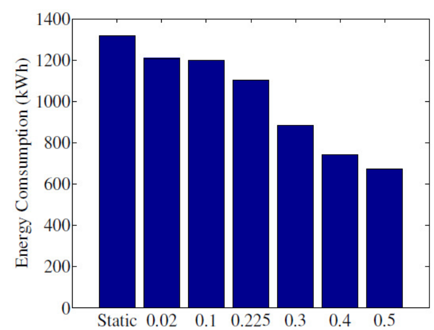


CDF of Task Scheduling Delay

- Effect of the reconfiguration cost on the solution

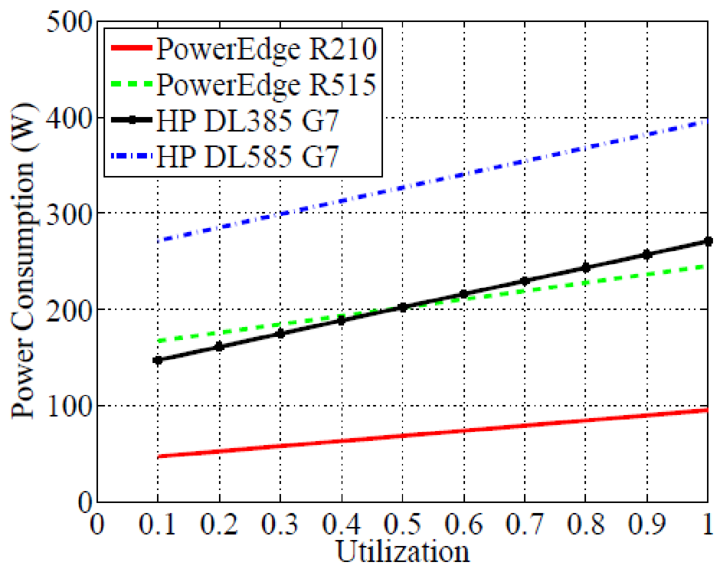


Average Scheduling Delay
as a Function of R

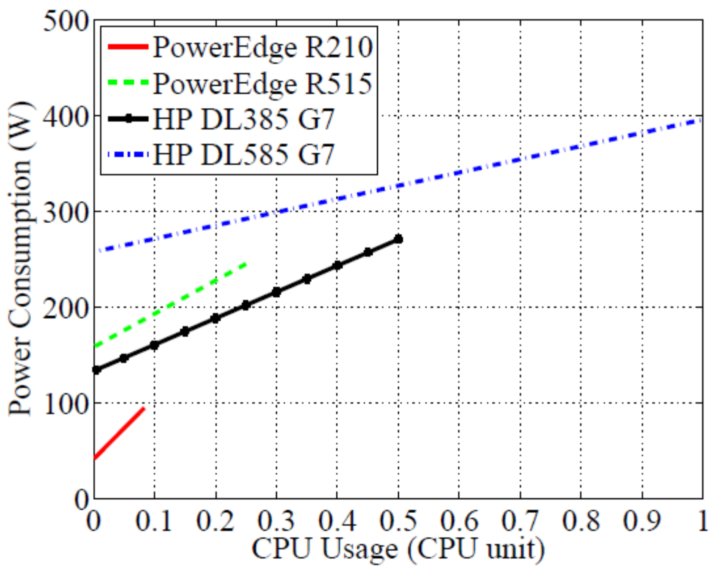


Energy saving
as a Function of R

Machine heterogeneity
Energy consumption

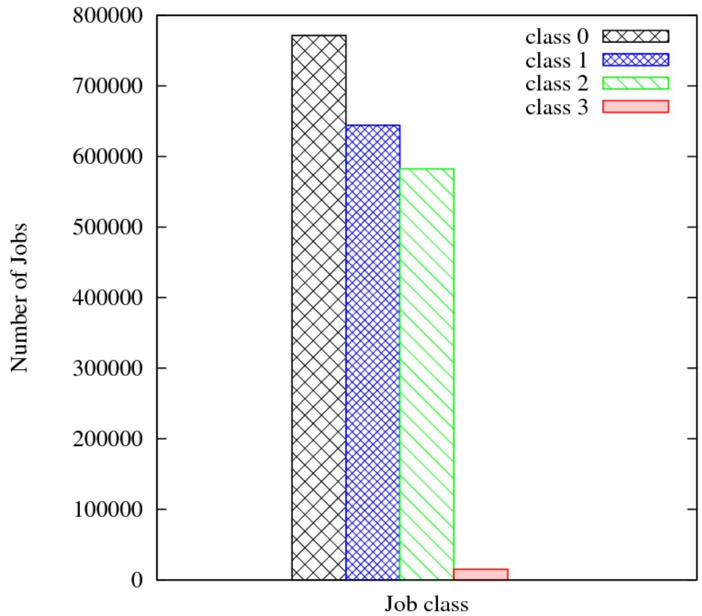


Energy consumption vs. Utilization

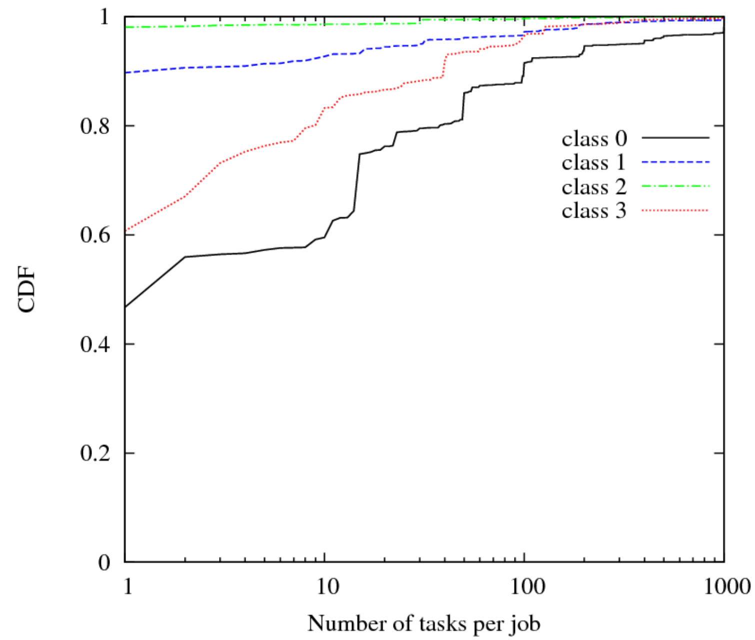


Energy consumption vs. Number of used cores (Normalized)

Application Heterogeneity: Job Priority and Size



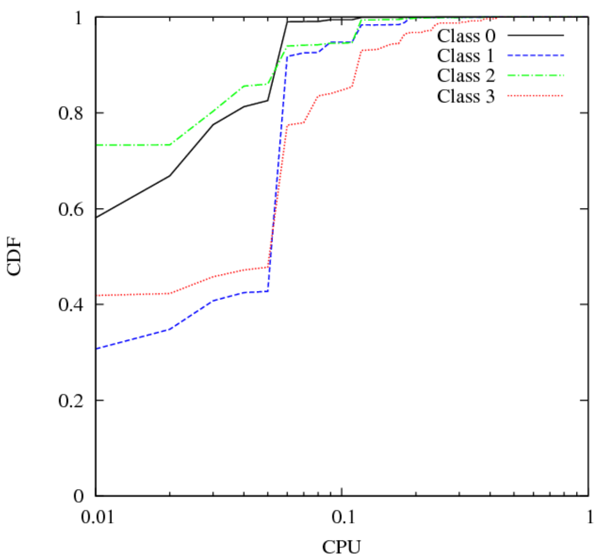
Number of jobs per class



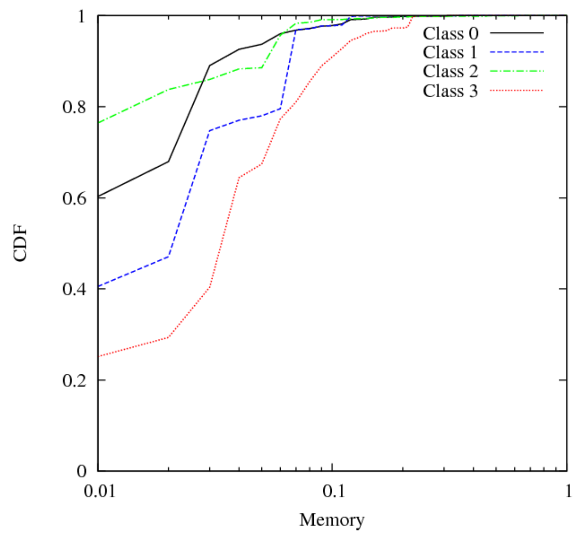
CDF of Number of tasks per job

- Most of the jobs have low priority
- Most of the jobs consists of <10 tasks, but a few of them have more than 1000 tasks

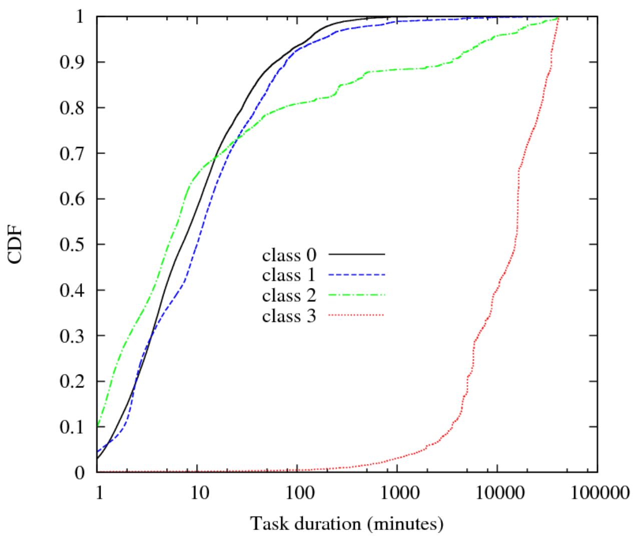
Application Heterogeneity: Task Size and Duration



CDF of Task CPU Requirement



CDF of Task Mem Requirement



CDF of Task Duration

- Most of the tasks require little resources, a few of them require a lot of resources
- Most of the tasks are short (<10 min), a few tasks are really long