A Cost-Sensitive Adaptation Engine for Server Consolidation of Multitier Applications

Gueyoung Jung, Calton Pu

Georgia Institute of Technology

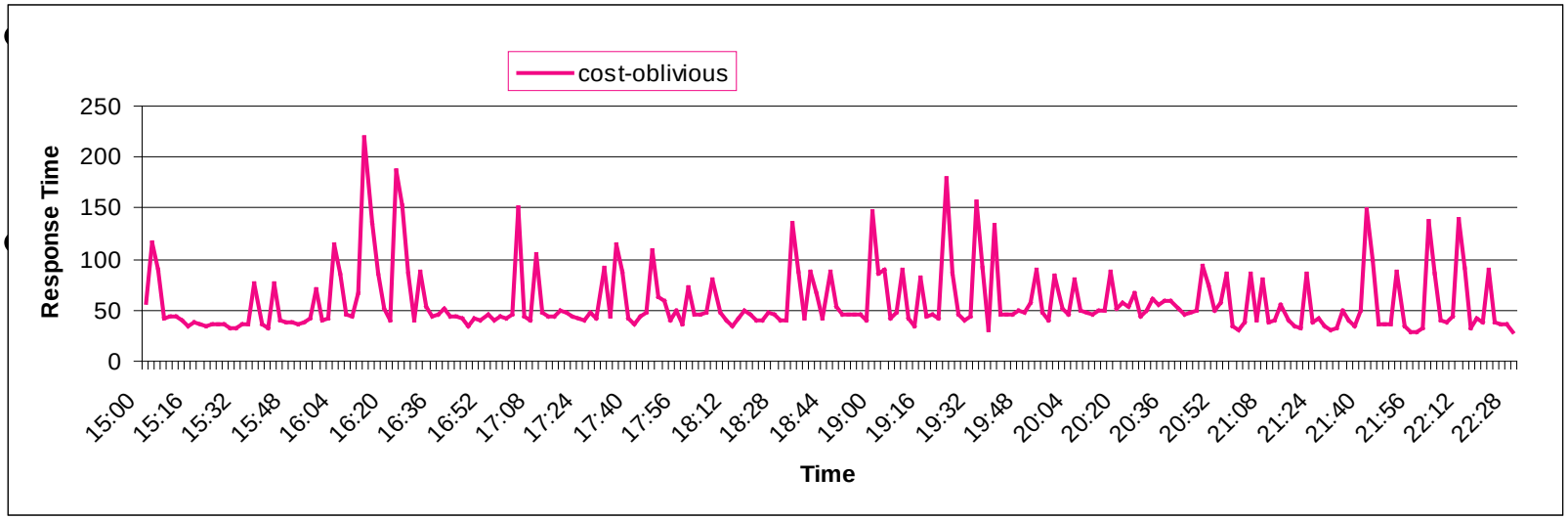
Kaustubh Joshi, Matti Hiltunen, Richard Schlichting

AT&T Labs Research

Context

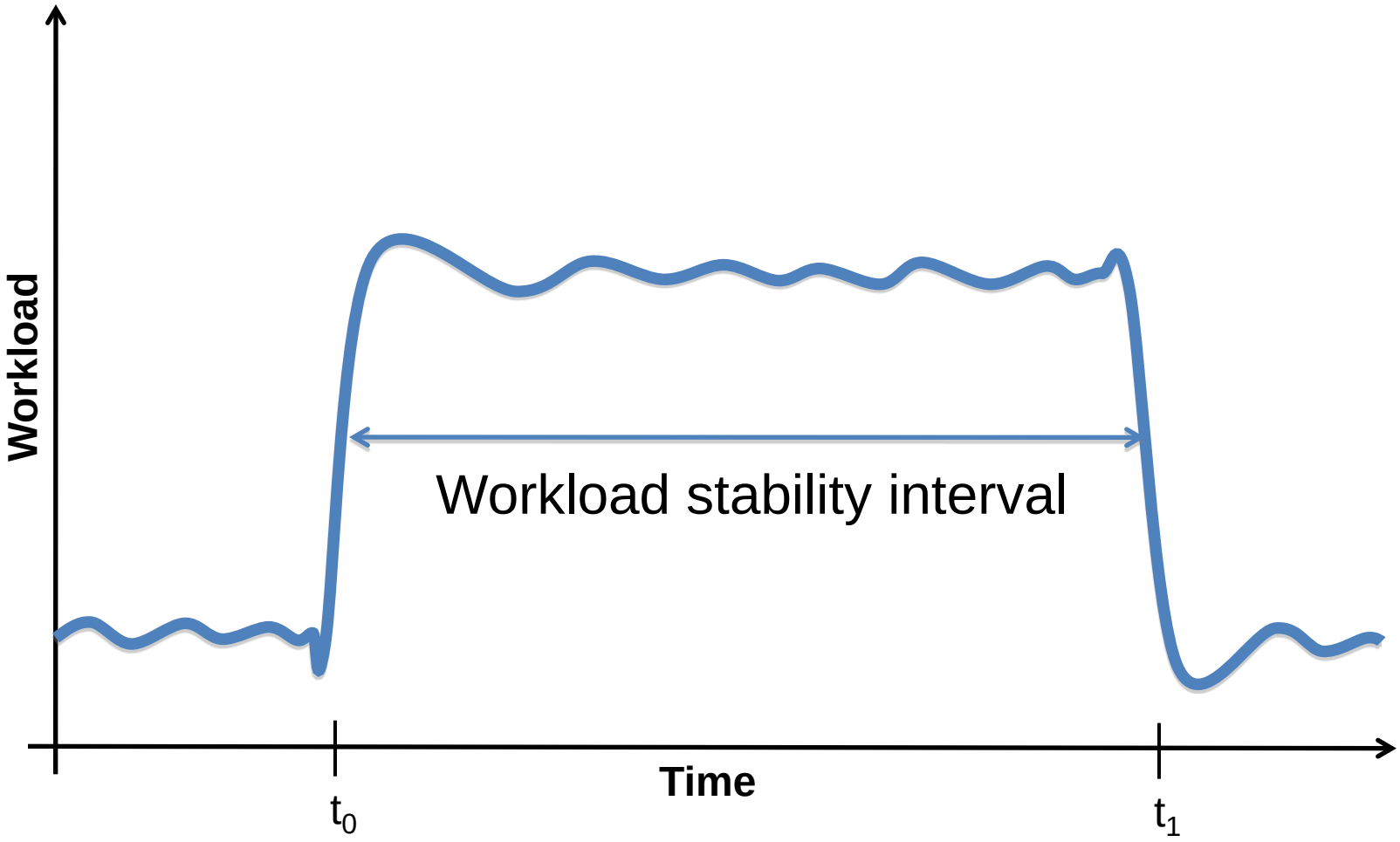
- Cloud infrastructures proving resource pool shared by multiple applications
- Multi-tier applications (e.g., web, application, database tier servers)
- Server consolidation through virtualization allowing each physical machine to host tier servers in isolated containers (VMs)
- Performance optimization in the context of end-to-end response time through resource configuration
- Resource configuration: CPU capacity tuning of VM, VM migration, and increase/decrease replication level

Motivations

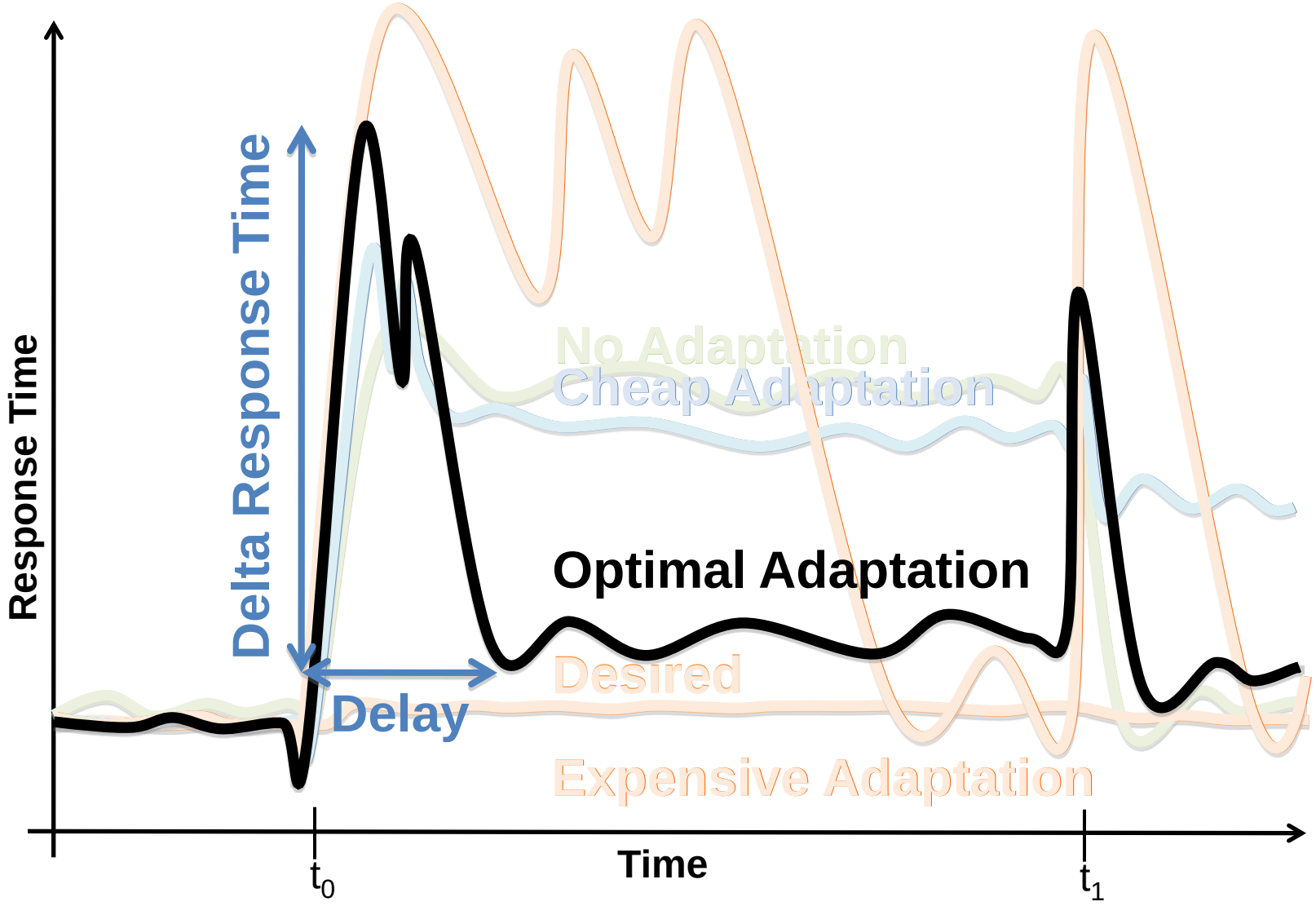


- Challenge: indiscriminate usage of adaptations can have significant impacts on the overall QoS of hosted applications.

Adaptation Benefit vs. Cost

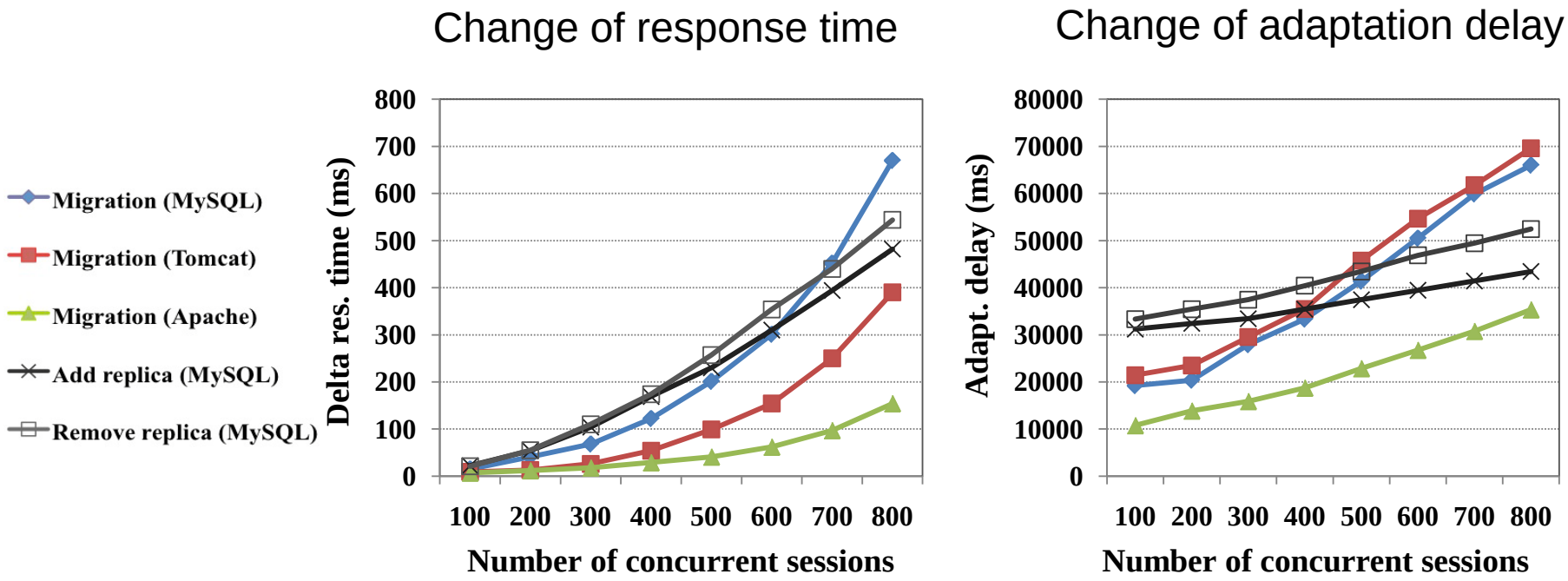


Adaptation Benefit vs. Cost



Adaptation Costs

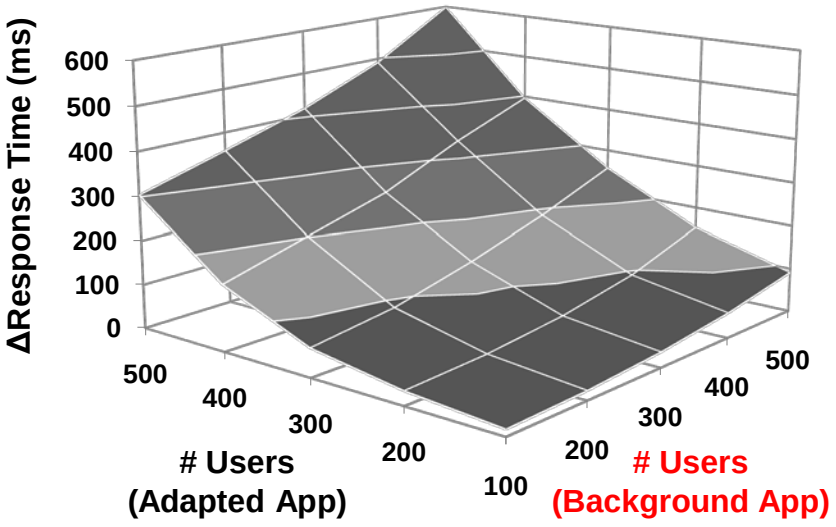
- Vary with workload, adaptation type, and performance characteristics of applications and their tier servers



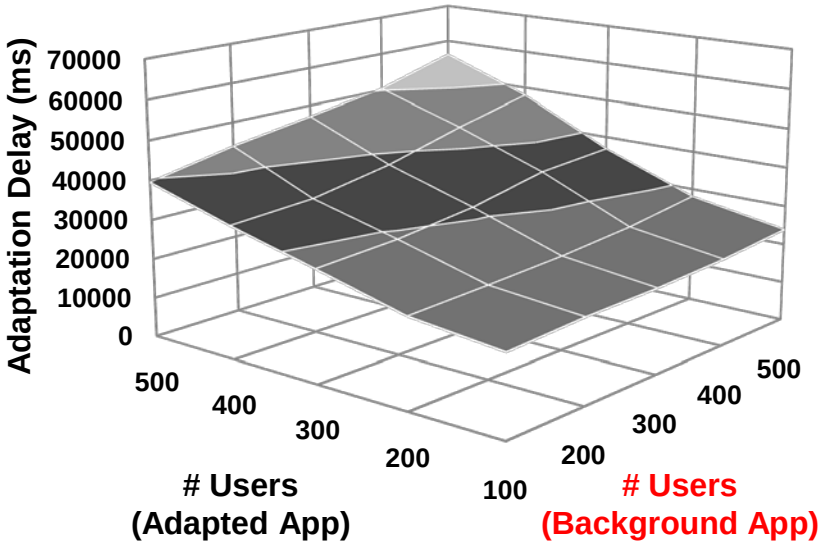
Adaptation Costs

- Affected by the workload of background applications sharing resources with adapted application (virtualization overheads)

Change of response time



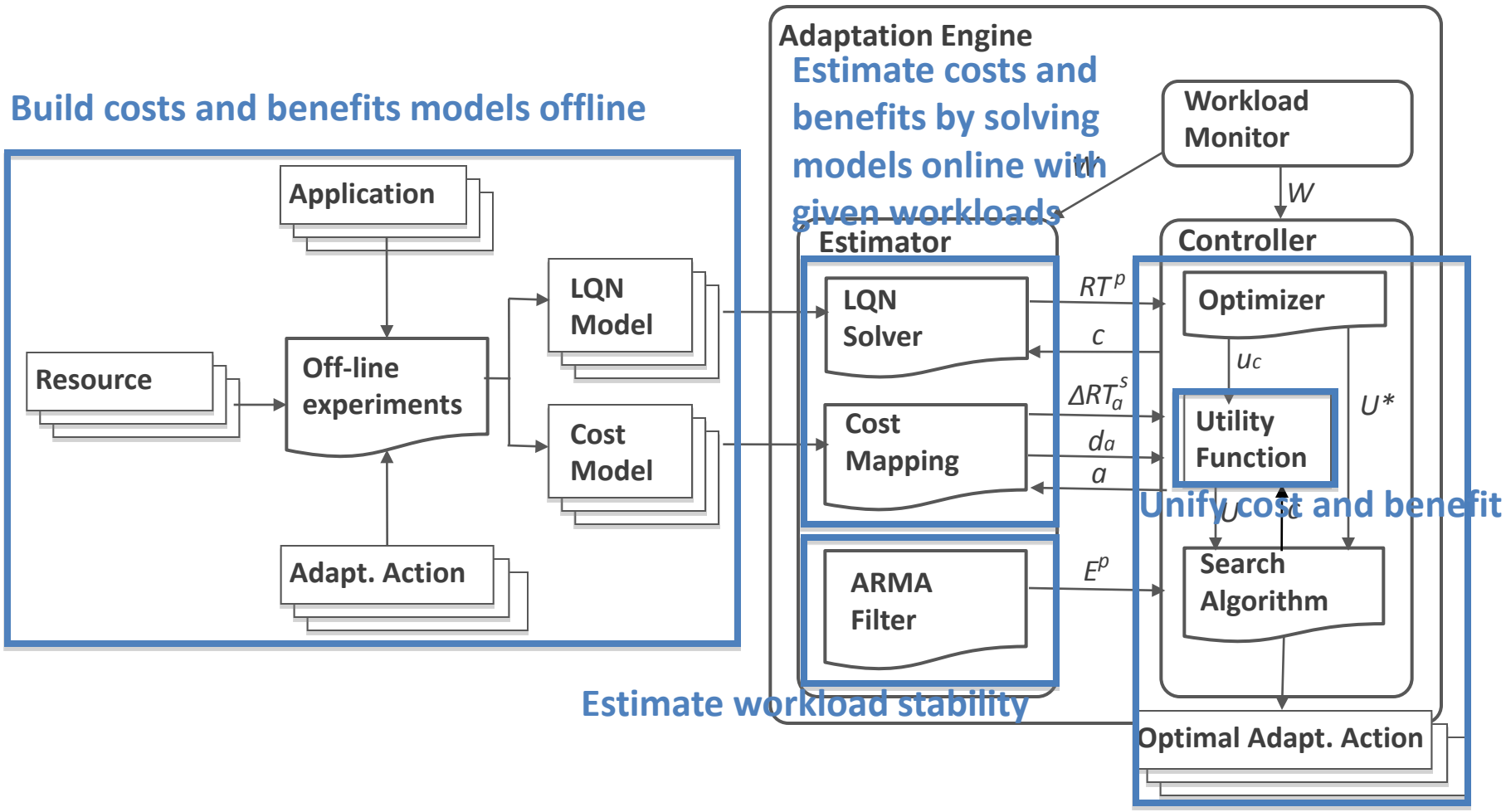
Change of adaptation delay



Contributions

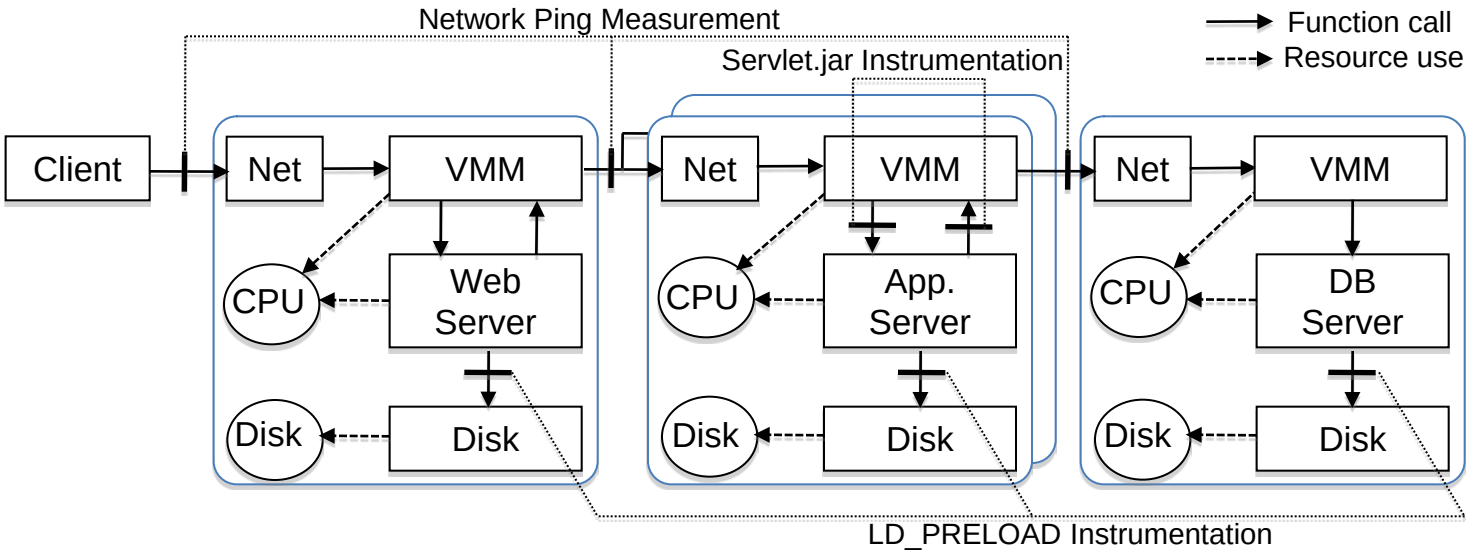
- Develop a framework to generate optimal balance between various adaptations' benefits and costs.
- Demonstrate the framework in dynamic resource configuration for multiple multi-tier applications in a server consolidation environment.
- Specifically, generate a set of optimal adaptation actions to maximize overall utility.

Architecture



Modeling

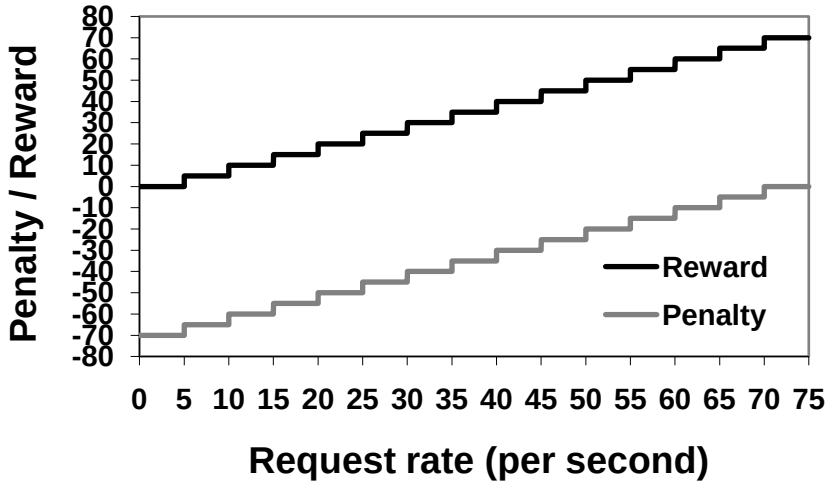
- Estimation of benefits: Given workload W and configuration c, estimate response time RT using layered queueing network models.



Modeling (cont.)

- Estimation of costs: Given workload W, configuration c, and adaptation action a, experimentally, iteratively measure delay d_a and delta response time ΔRT_a. Then generate a mapping table to be used online.
- Estimation of workload stability CW: Given workload history, estimate how long current workload will stay within a range using ARMA filter.

Utility Function



Each application s has its own SLA that provides TRT^s, base R^s and P^s.

R^s and P^s are factored by the intensity of workload W^s.

- Maximize $U = (CW - \sum_{a^k \in A} d_{a^k}) \sum_{s \in S} u^s_{c_{i+1}} + \sum_{a^k \in A} (d_{a^k} \sum_{s \in S} u^s_{c^{k-1},a^k})$

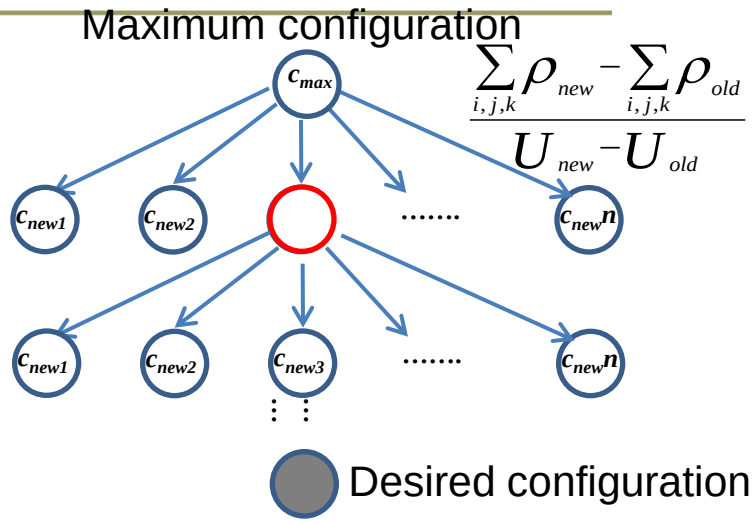
  Benefit utility — Cost utility

  Where CW is the estimated length of stability interval.

Optimization

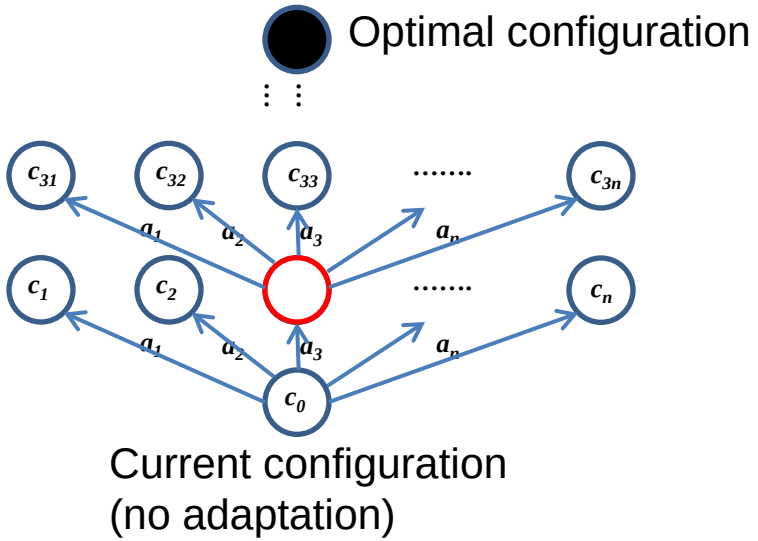
- Search the desired configuration using bin-packing and gradient-based search algorithms.

  - not consider adaptation costs.



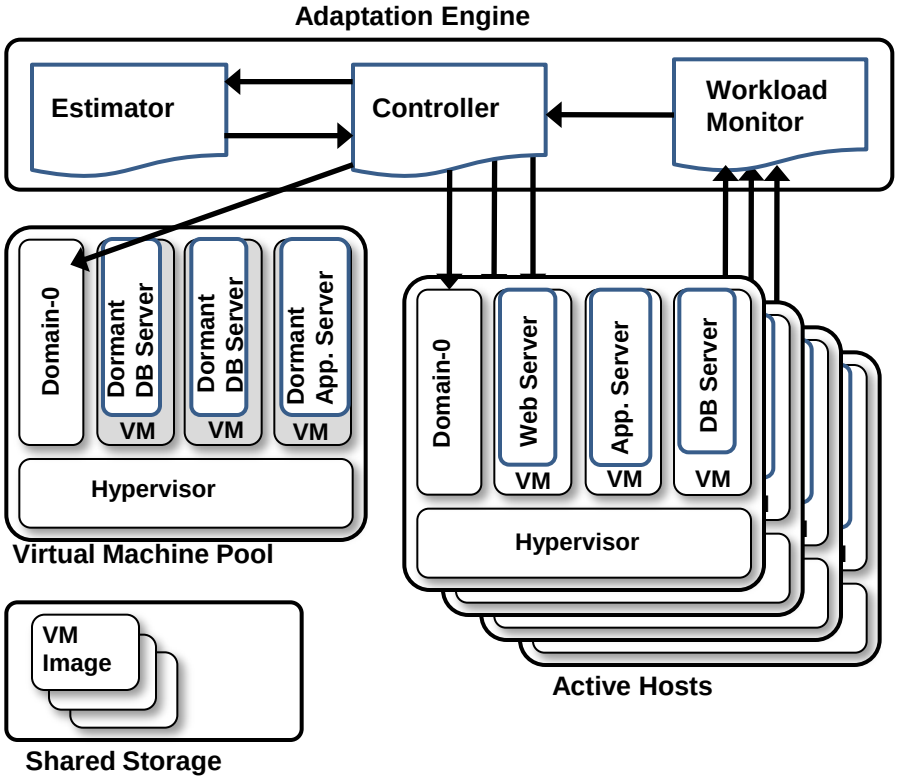
- Generate optimal actions using a best first search algorithm.

  - A* graph search algorithm.



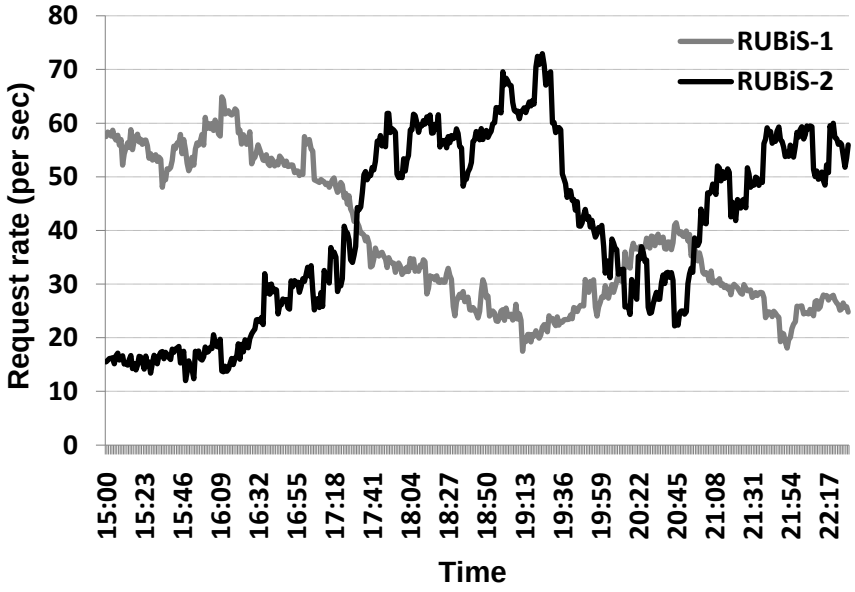
Test-bed Architecture

- Develop a small virtualized data center
- Deploy multiple 3-tier RUBiS applications

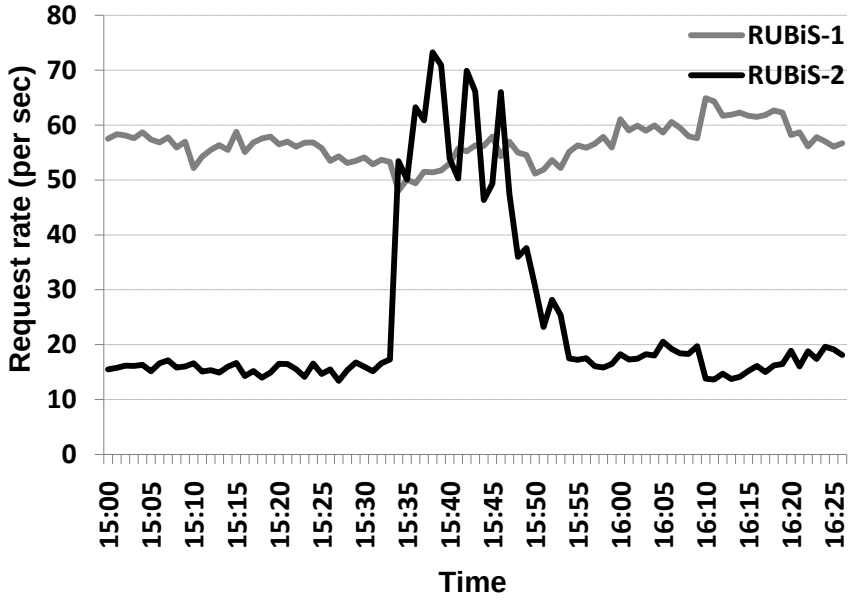


Workloads

Time of day trace

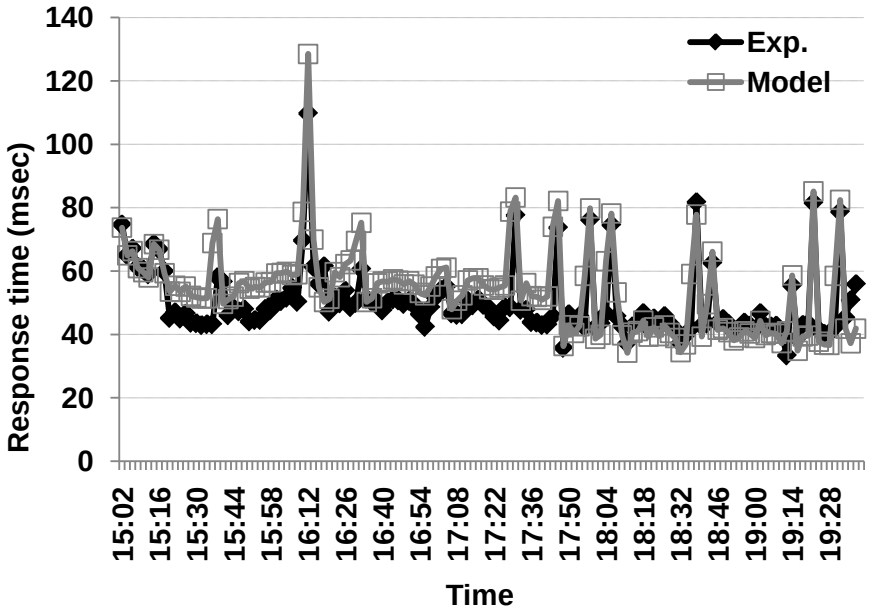


Flash crowd trace

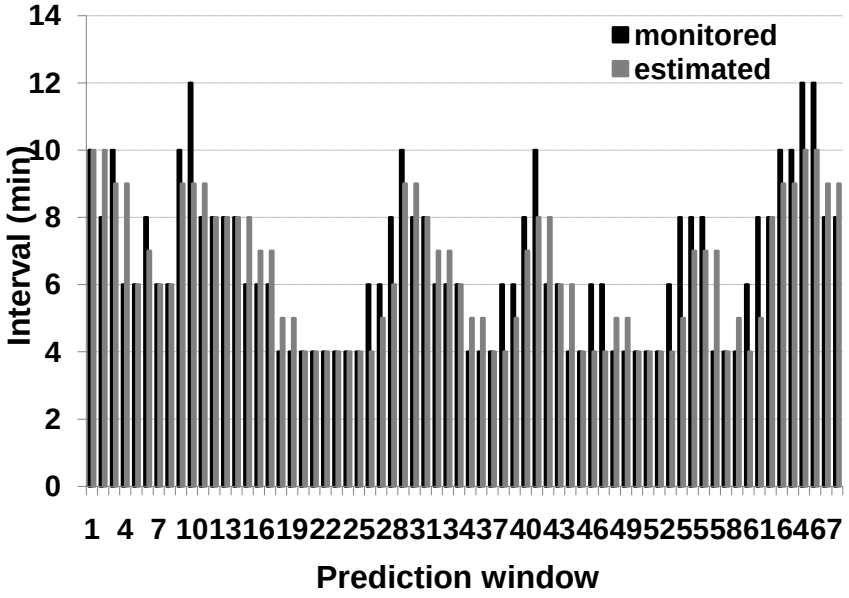


Model Accuracy

Estimation error of benefits and costs is around 15%



Estimation error of workload stability is around 15%



Comparison Evaluation

- Compare 5 strategies

  NA: No adaptation. A Static configuration used

  CO: Cost-Oblivious. No consideration of adaptation costs

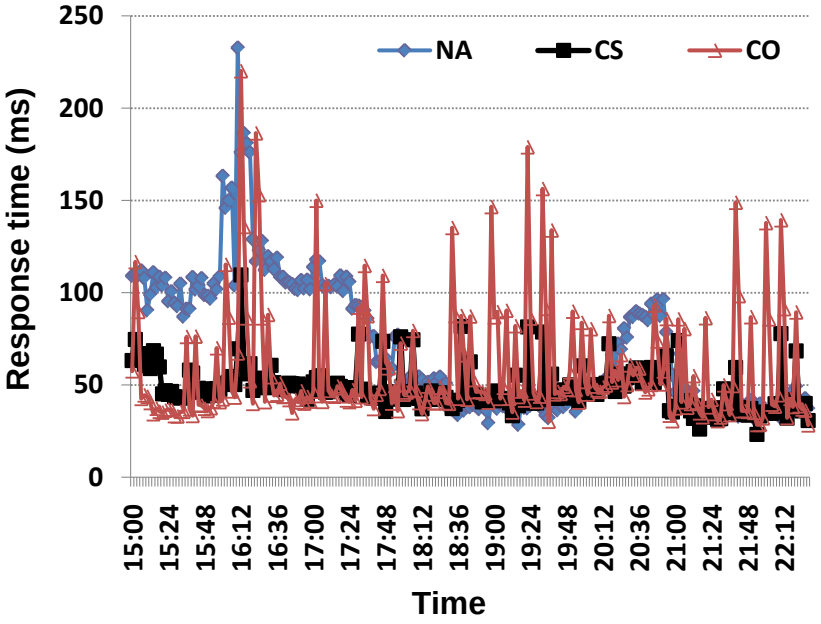
  Oracle (Desired): Simulation with adaptation costs = 0

  1-hour: Adaptation every 1 hour

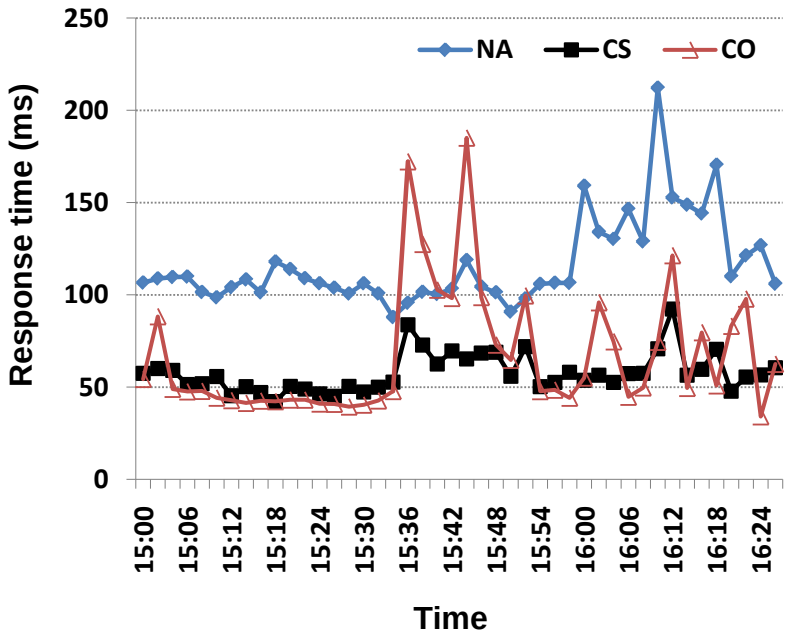
  CS: Cost-Sensitive.

End-to-End Response Times

Time of day trace

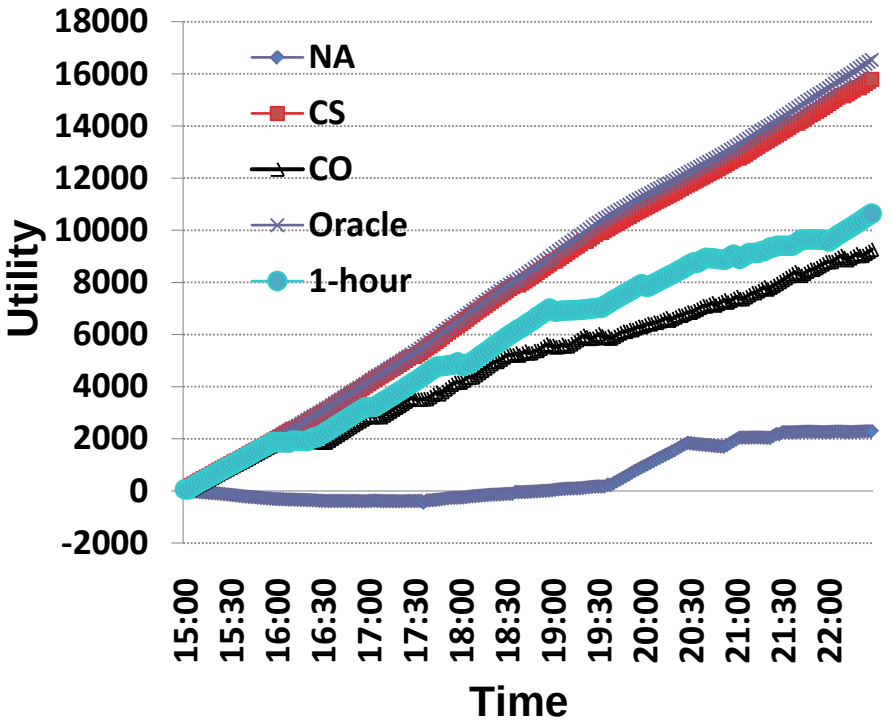


Flash crowd trace

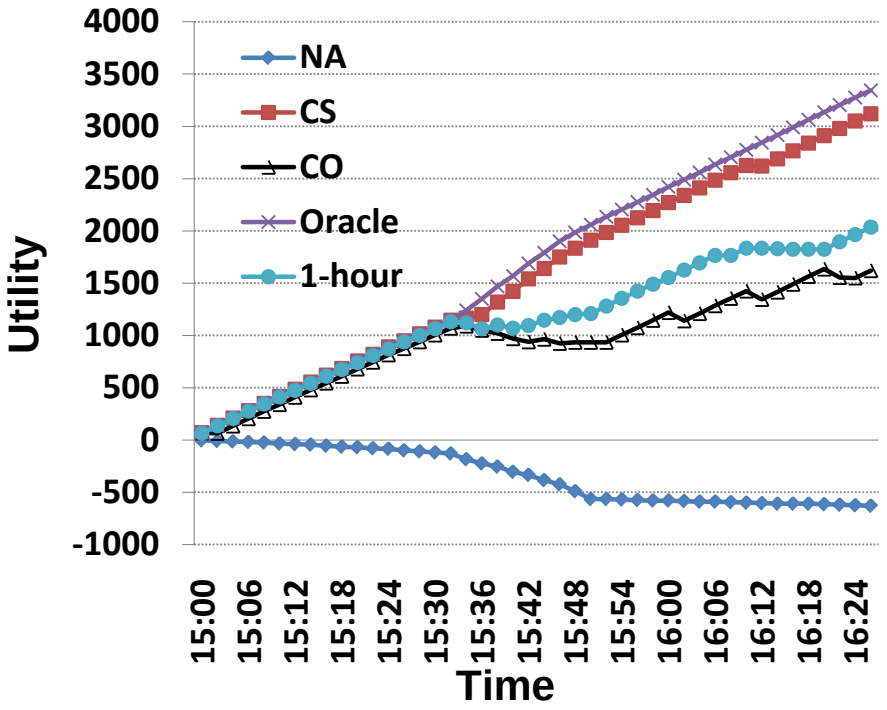


Cumulative Utilities

Time of day trace



Flash crowd trace



Adaptation Actions

The number of adaptation actions triggered in flash crowd scenario

Type of Action	CS	CO
CPU Increase and Decrease	14	36
Add MySQL replica	1	4
Remove MySQL replica	1	4
Migrate Apache	4	10
Migrate Tomcat	4	10
Migrate MySQL	0	2

Conclusion

- Adaptation actions such as VM replication and migration can impose significant performance costs.
- Our approach makes smart decision on ***when*** and ***how*** to act to enhance the satisfaction of response time SLAs.

On-going Work

- Integrate power management into the problem formulation by considering power consumption as another cost.
- Handle large-scale setup by extending the framework to multi-level hierarchical control, where each level represents different time scale and scope of control.

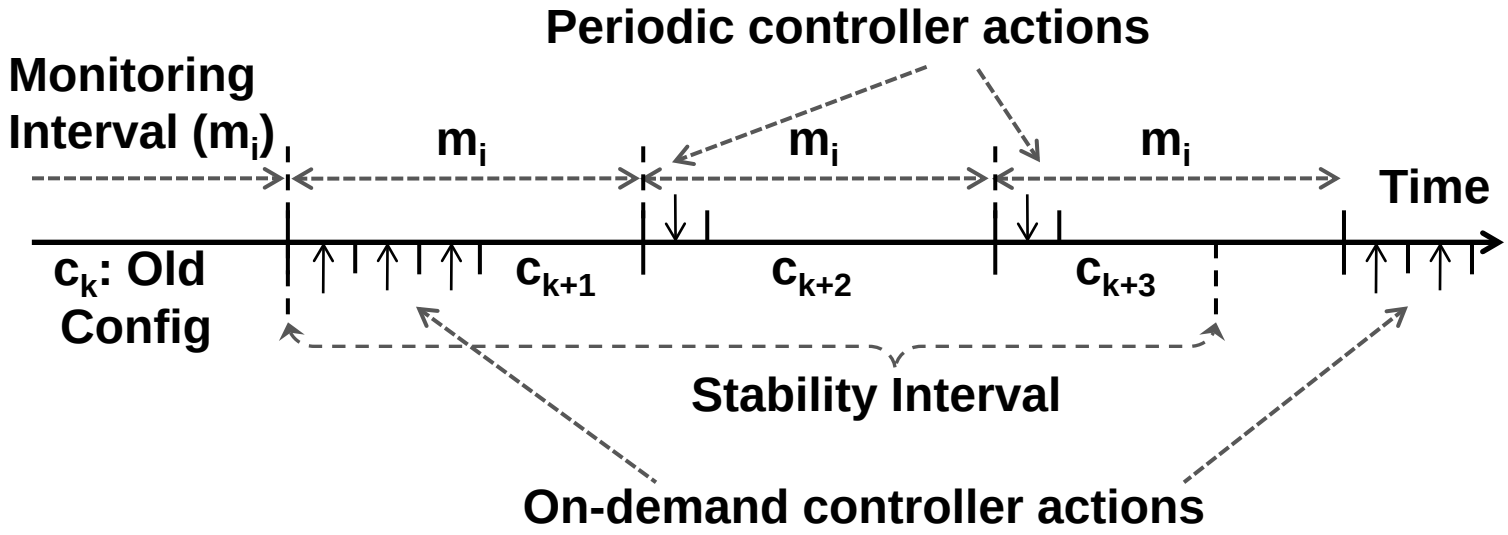
Georgia Tech

at&t

Questions?

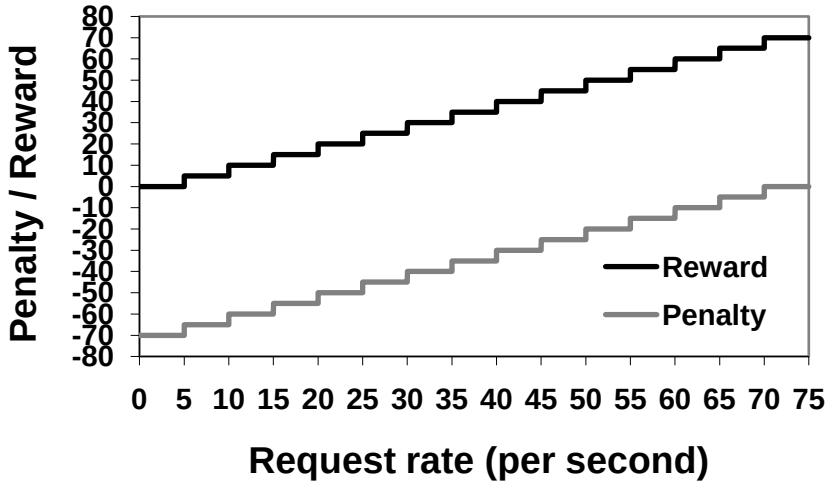
Multi-Level Contol

Periodic controller generates relatively cheap actions for short intervals against any workload changes.



On-demand controller generates relatively expensive actions for longer interval against a certain degree of workload changes.

Utility Function (detail)



Each application s has its own SLA that provides TRT^s, base R^s and P^s.
R^s and P^s are factored by the intensity of workload W^s.

- $u^s = 1[RT^s \leq TRT^s]\ R^s(W^s)/M - 1[RT^s > TRT^s]\ P^s(W^s)/M$
  where M is the length of unit interval.
- Cost $u^s{}_a$ can be computed by replacing RT^s with
  $RT^s{}_a = RT^s + \Delta RT^s{}_a$ in the above equation.
- Maximize $U = (CW - \sum_{a^k \in A} d_{a^k}) \sum_{s \in S} u^s{}_{c_{i+1}} + \sum_{a^k \in A} (d_{a^k} \sum_{s \in S} u^s{}_{c^{k-1},a^k})$

Optimization Algorithm

1. Start from current configuration c_0.
2. Compute the upper bound utility u^*.
3. Explore adaptation actions including "do nothing" if the node is candidate node.
4. Compute utility of each node and store it to V.
   for each intermediate node, using u^*.
   e.g., $U_1=(CW\text{-}d_{a_1})\ u^* + d_{a_1}\ u_{a_1}$
5. Sort nodes in V by decreasing utility and select the first node.
6. If the node is a candidate node, then stop search and return actions used to reach the node.
   Otherwise, keep searching from the node.