Generating Adaptation Policies for Multi-Tier Applications in Consolidated Server Environments

Gueyoung Jung, Calton Pu

College of Computing
Georgia Institute of Technology

Kaustubh Joshi, Matti Hiltunen, Richard Schlichting

AT&T Labs Research

Outline

- Research Challenges
- Our Contributions
- Overview
- Approach
  - Formal Problem Statement
  - Application Modeling
  - Optimization
  - Rule-Set Construction
- Summary

Challenges

Problem: Dynamically configuring systems to adapt to changing conditions

- Hybrid approach to bridge both worlds
  - On-line controller (with stochastic models, reinforcement learning, control theory)

  ☞ lack transparency and predictability

  - Rule-based expert systems

  ☞ hard to maintain linkage to underlying systems

Challenges (contd.)

Focus: Dynamic resource provisioning in virtualized consolidated data centers hosting multiple multi-tier applications

- Promising cost-effectiveness and high resource utilization

However, with unpredictable workloads

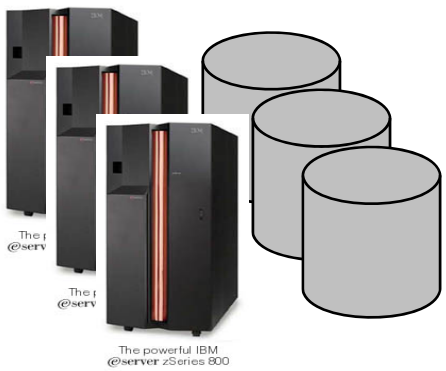
- More complex models required
- Larger optimization space of possible system configurations
- More sophisticated adaptation policies required

Runtime Resource Management

Making decisions
- Evaluating monitoring results
- Adaptive systems

Acting/Adapting
- Start/stop processes (e.g., adjust replication degree of a component)
- Migrate processes
- Adjust CPU allocation (e.g., virtual machine technology)

Monitoring
- Request Rates
- Resource utilization
- Response times etc.



Shared Resource Pool with applications

How to Use Models for Decision Making

Making decisions

Model Inline (MIL)

- Model(s) evaluated **at runtime** given current system workload as input
- The rewards for alternative configurations can be calculated to determine **a better configuration**

Model Offline (MOL)

- Model(s) evaluated **before system deployment** using various workloads as inputs
- **Optimal configuration** determined for each different workload mix
- **Adaptation rules** generated based on model outputs

MOL vs. MIL

MOL

- Rules can be inspected by system administrators and domain experts
- Resulting rules can be used by existing rule-based management systems
- The time and compute intensive model evaluation is out of critical path (offline), while rules can be evaluated very quickly at runtime

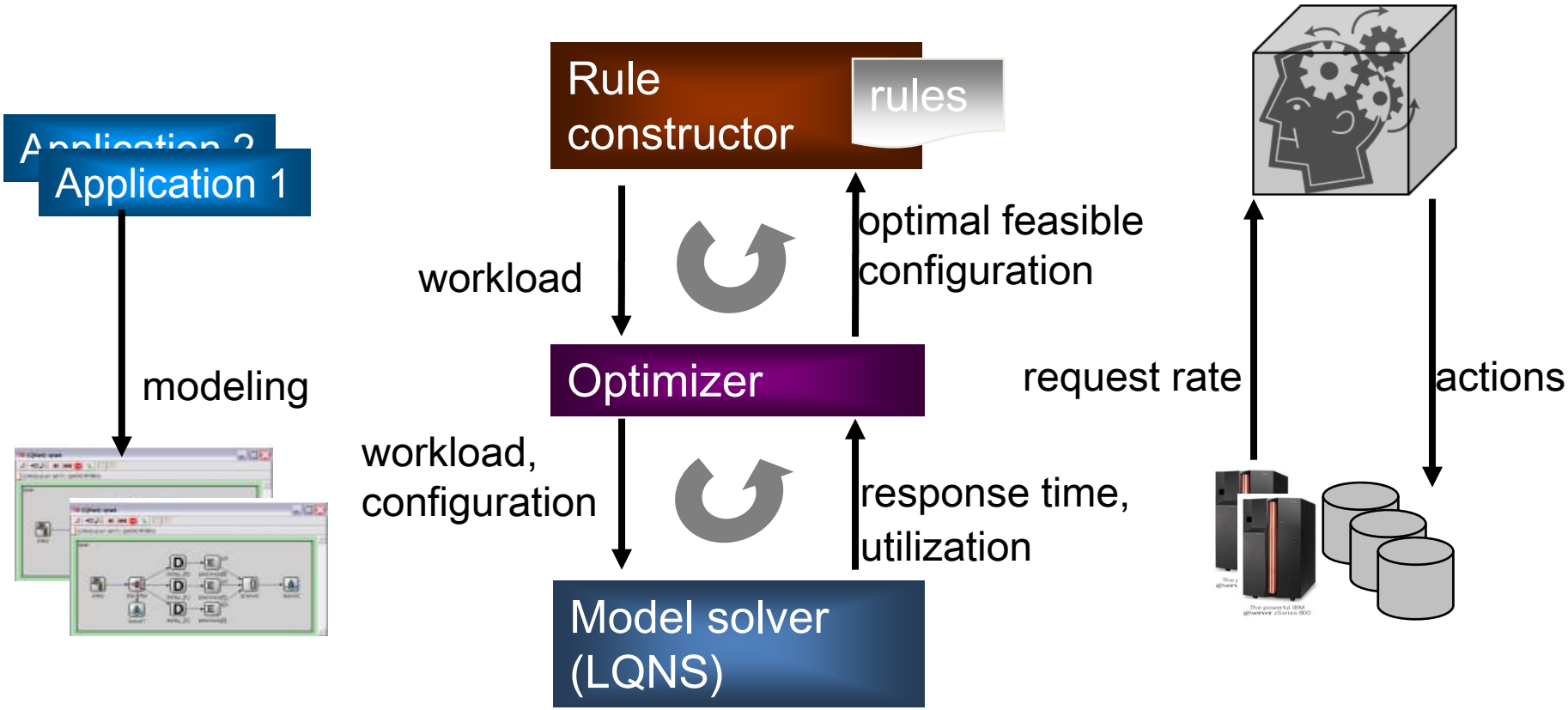
MIL

- Potentially more accurate (evaluate exactly with the current load and configuration)
- Model parameters can be updated at runtime

MOL in Action: Our Contributions

- Proposing a hybrid approach
  - ☞ Best of both worlds
- Automatically generating adaptation rules offline
  - ☞ Layered queueing models for Xen-based virtualization
  - ☞ Novel optimization technique
- Efficiently supporting an adaptive system for dynamic resource provisioning
  - ☞ Fine-grained resource (re)allocation
  - ☞ Compact, human-readable
  - ☞ Accurate even with a subset of possible scenarios

Approach Overview



Formal problem statement, then discuss steps bottom up.

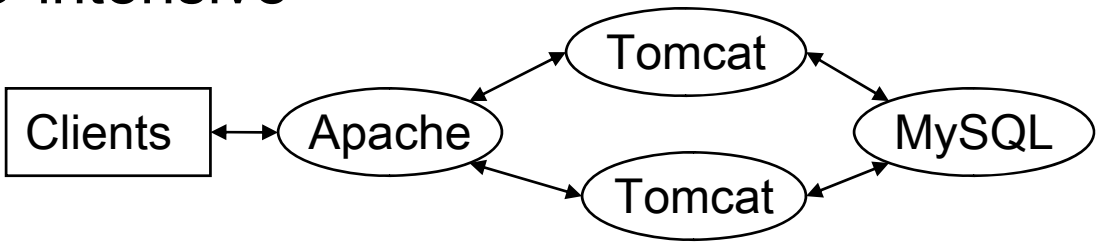
Formal Problem Statement (1/2)

Given

- A set of computing resources *R*
- A set of applications *A*:
  - each application consists of a set of components (tiers)
  - each component has a set of possible replication degrees
  - each application supports multiple transaction types
- For each transaction type:
  - transaction graph describing interactions between components and service time in each component
- For each transaction of each application:
  - Utility function defined using the desired (mean) response time and the reward/penalty for meeting/missing this time

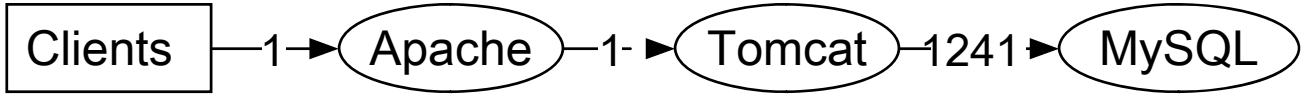
Example: RUBiS

RUBiS: a Java-based 3 tier auction site benchmark, CPU-intensive

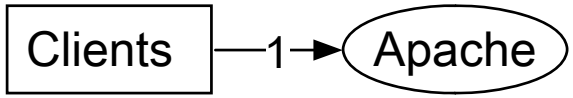


26 different transaction types with different behaviors

AboutMe Transaction



Home Transaction



Formal Problem Statement (2/2)

Measured at runtime:

- Workload (request rate) of each transaction type

Goal:

- Configure the set of applications A on the resources R
- Maximize utility $U = \sum_{i \in A} \sum_{j \in Ti} U_{ij}$

$$U_{ij} = w_{ij} R_{ij} [or\ P_{ij}] (TRT_{ij} - MRT_{ij})$$

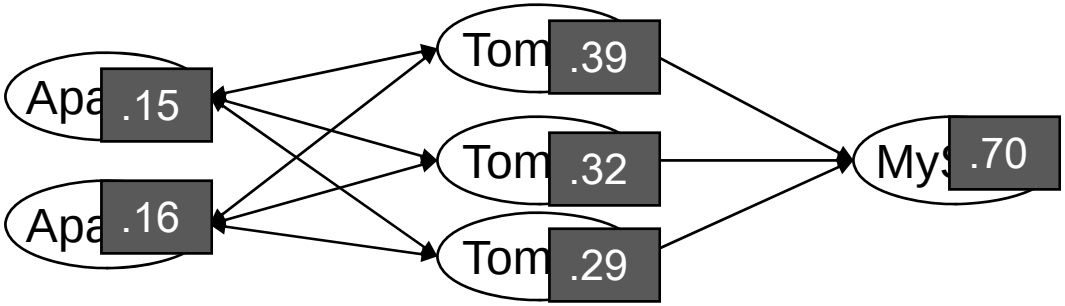
- Configuration:
  - Degree of replication for each component
  - Virtual machine parameters (e.g., CPU reservation)
  - Placement of VMs on the physical machines R

  (a VM contains a replica of a component of an application)

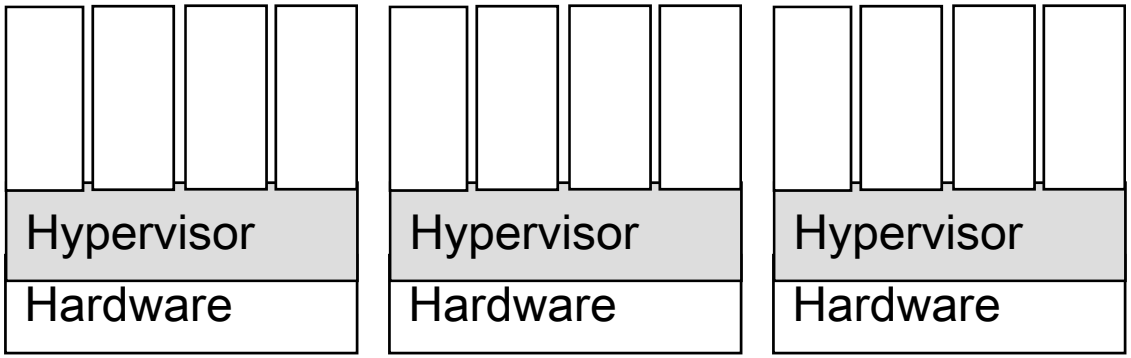
Example: RUBiS

Application components:

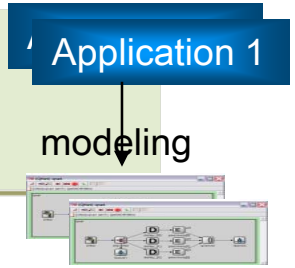
Logical configuration:



Physical configuration:

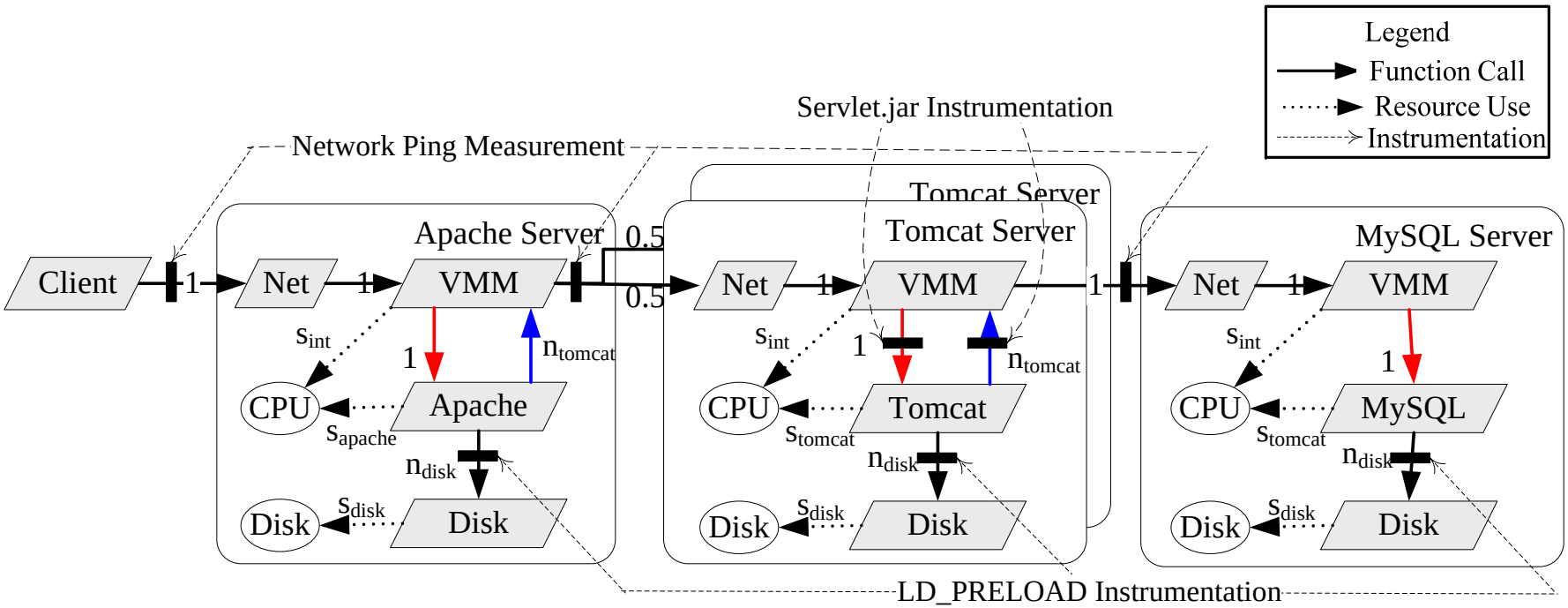


Application Modeling



- Modeling layered, multi-tier software systems
- Modeling virtualization overhead
  - Xen-based virtual machine monitor
- Modeling blocking phenomenon
  - Synchronous calls between components
- Arbitrary mixes of transaction types in workload

Layered Queueing Model



Modeling CPU, network I/O, Disk I/O, and the per-message delay of virtual machine monitor

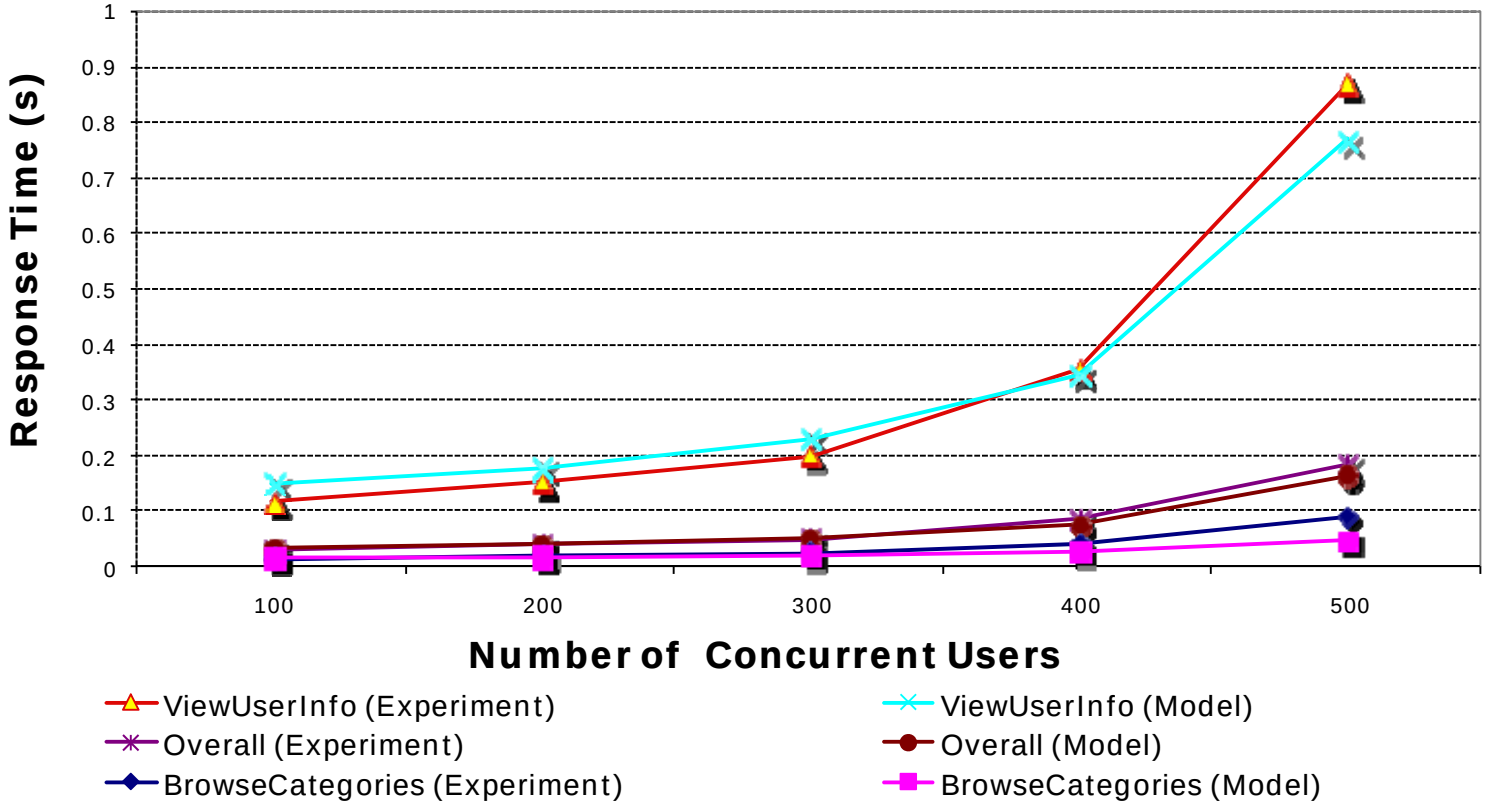
Parameterizing models at training phase without intrusive instrumentation

Model Validation (1/4)

Model predicts:

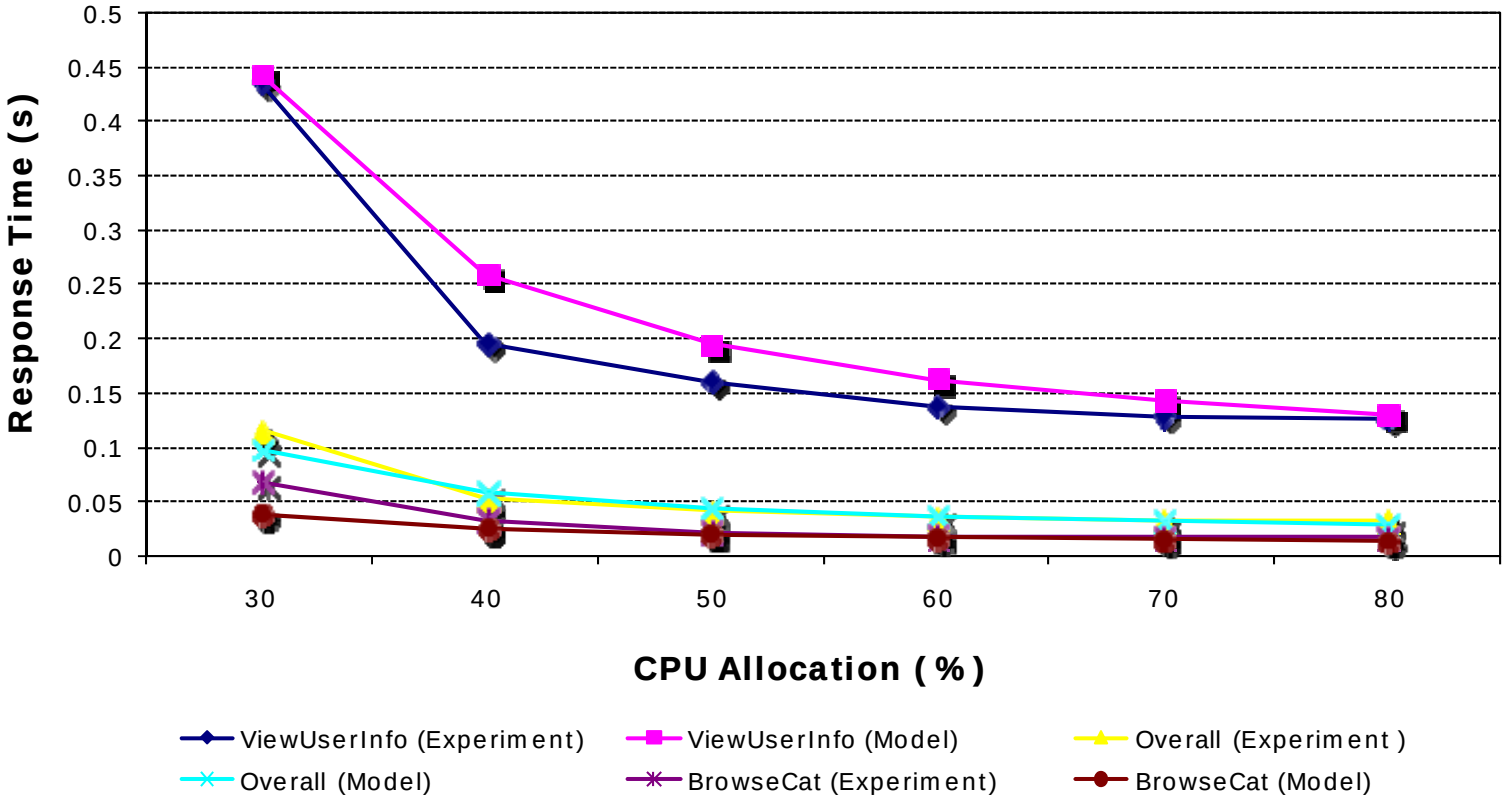
- Mean response time for each transaction type given a configuration and request rate
- CPU utilization of each application component given a configuration and request rate

Model Validation (2/4)



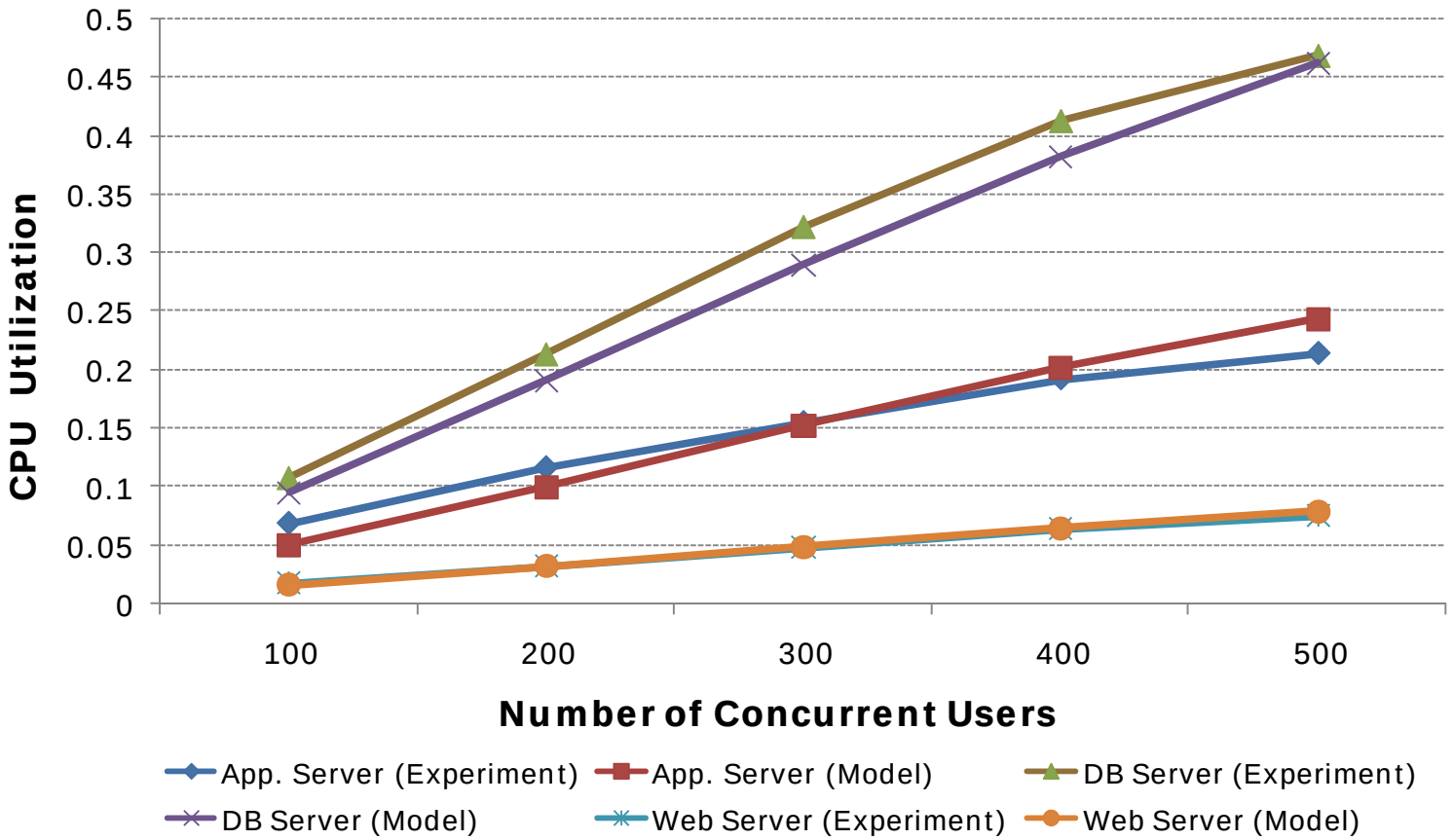
Model predicts response time at different request rates (fixed configuration)

Model Validation (3/4)



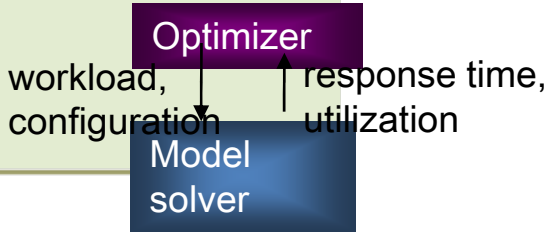
Model predicts response time at different CPU allocations

Model Validation (4/4)



Model predicts CPU utilization at different tiers at different request rates (fixed configuration)

Optimization



For a given workload, find the configuration with the maximum utility

Huge parameter space to explore, NP-Complete problem

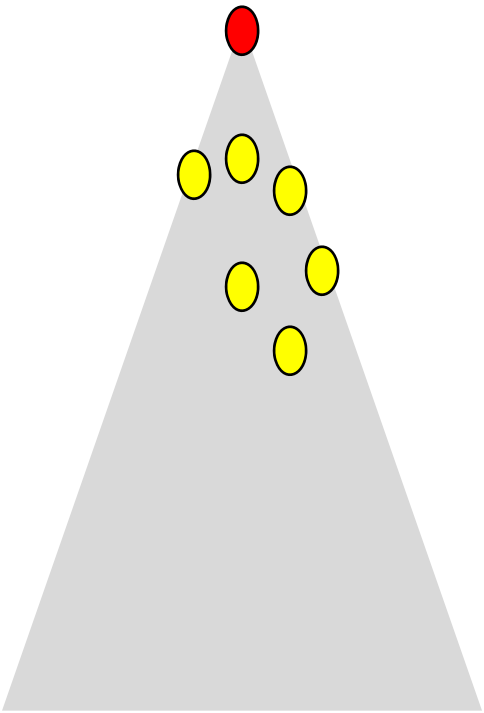
Observations:

- Response time is monotonic function of number of replicas and CPU fraction

Key Techniques:

- Decouple logical configuration from physical component placement
- Start from maximum configuration, search an optimal path that fits logical configuration into physical resources

Optimization Algorithm



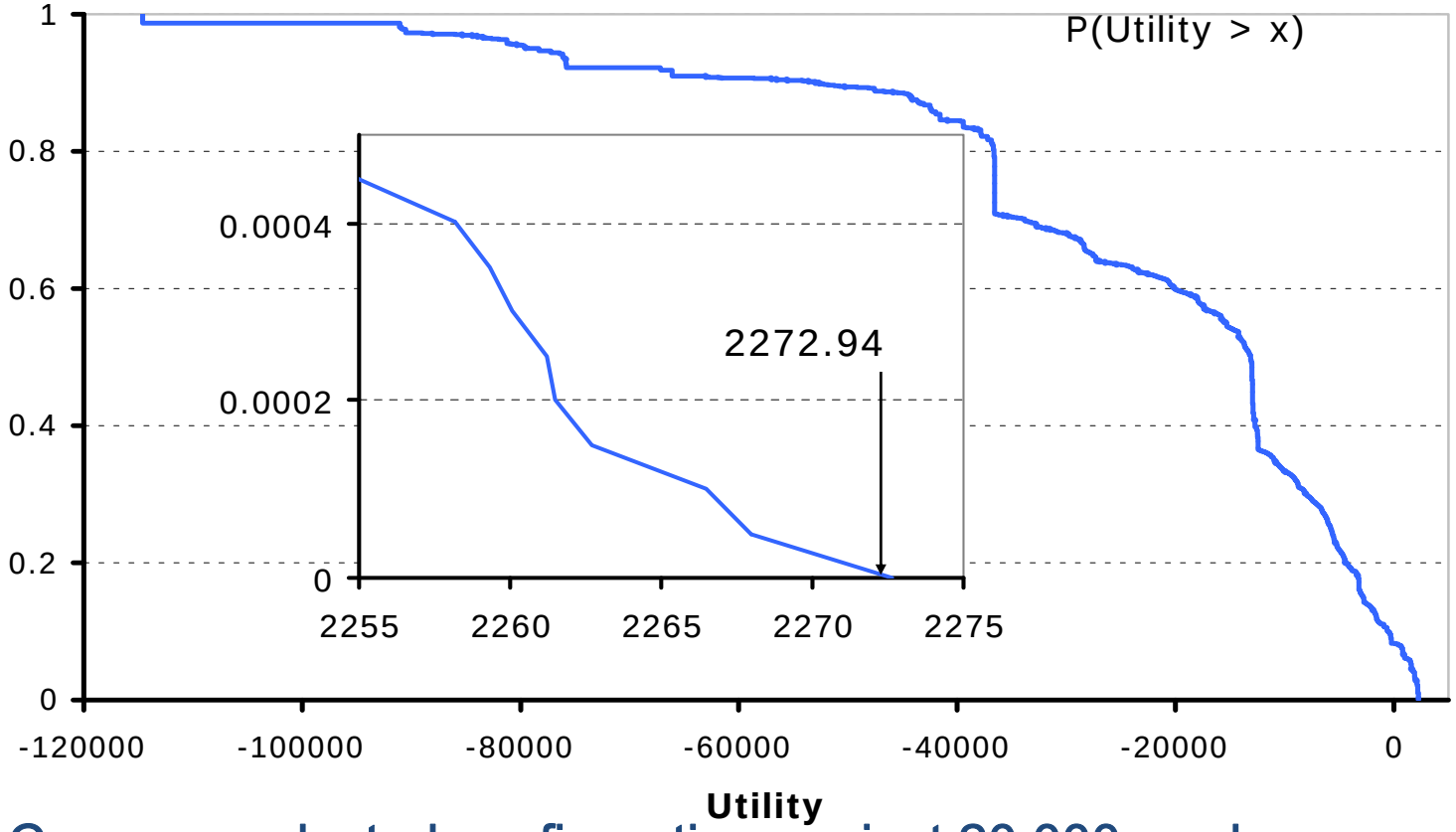
Maximum configuration:

- Each component of each application has the maximum number of replicas, each with 100% of a CPU of their own.
- Use model solver to get actual resource utilizations ρ and the response times (for calculating utility U).

Algorithm:

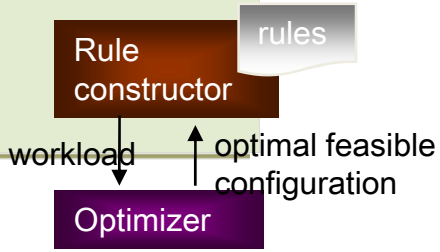
1. Use *bin-packing* algorithm to find out if the utilizations ρ can be fitted in the actual resources R.
2. If not, evaluate possible alternatives for reducing utilization:
   - Reduce number of replicas for some component
   - Reduce CPU fraction for some virtual machine by 5%
3. Determine the actual utilizations and utility for the different options.
4. Choose the one that maximizes: $$\frac{\sum_{i,j,k} \rho_{new} - \sum_{i,j,k} \rho_{old}}{U_{new} - U_{old}}$$
5. Repeat until configuration found

Optimality of Generated Policies



Compare selected configuration against 20,000 random configurations

Rule-Set Construction

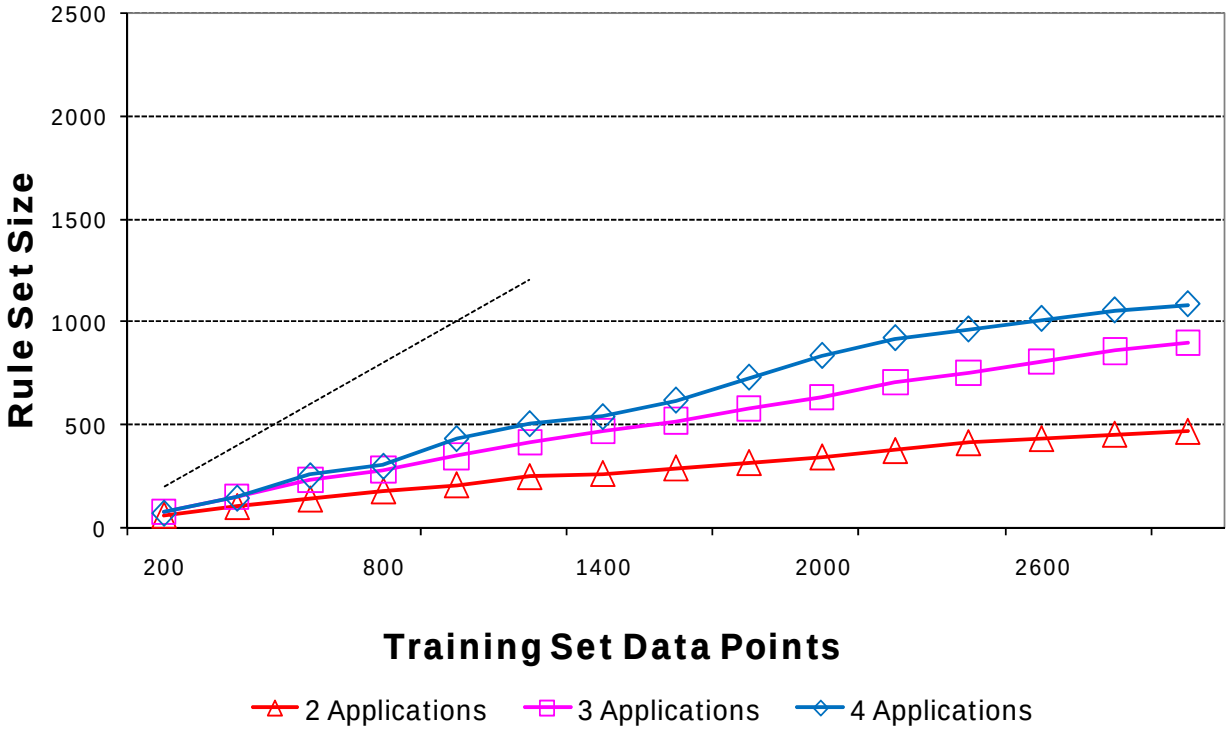


Rule Constructor:

- Randomly generates a set of workloads *WS* based on SLA for each application
- Invokes optimizer to find optimal configuration *c* for each $w \in WS$
- Gives (*w*, *c*) pairs (*raw rule-set*) still need interpolation for
- Use decision tre
  decision tree
- Linearize into ne

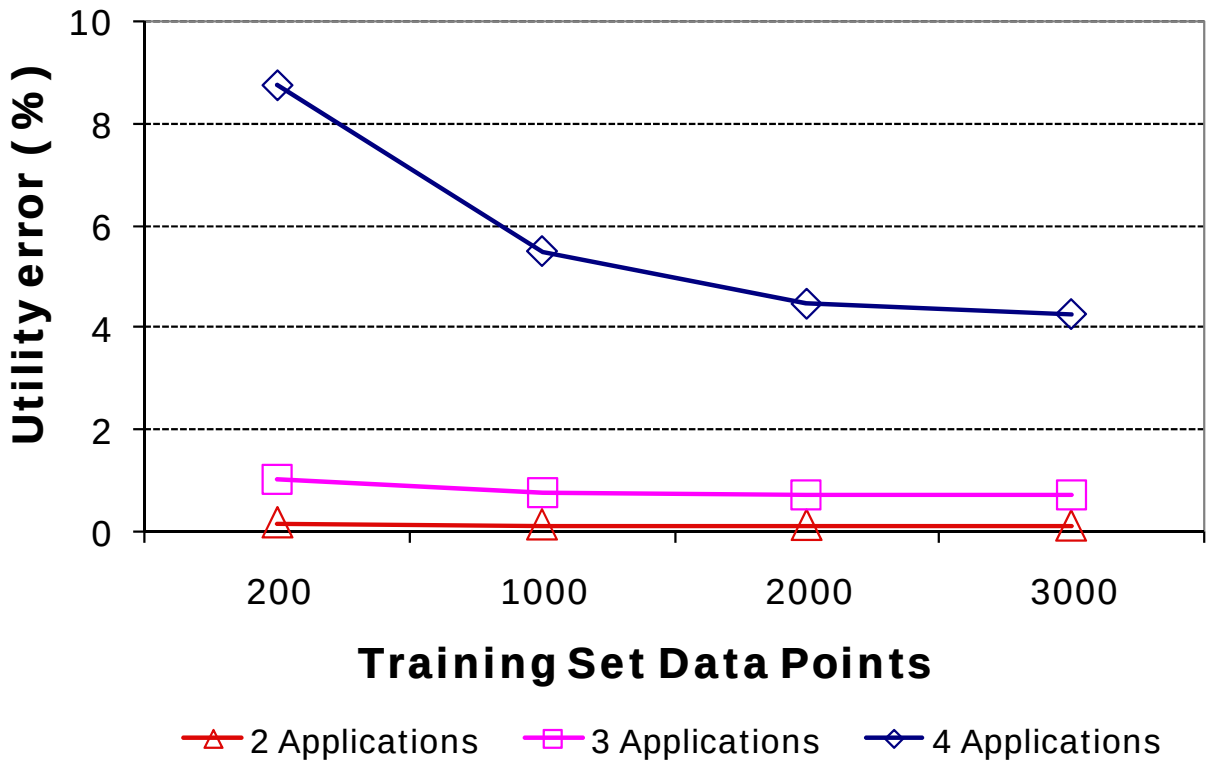
```
if (app0-Home <= 0.113882)
    if (app1-Home <= 0.086134)
        if (app1-Browse > 0.051189)
            if (app0-Home > 0.023855)
                if (app1-Browse <= 0.175308)
                    if (app0-BrowseRegions <= 0.05698)
                        config = "h0a0c2h1a1c2a0c0h2a0c1a1c1a1c0";
                    if (app0-BrowseRegions > 0.05698)
                        if (app1-Browse <= 0.119041)
                            if (app1-Browse <= 0.086619)
                                config = "h0a0c2h1a1c2a0c0a1c0h2a0c1a1c1";
.......
```

Size of Rule-Set



The size of the rule set increases when the number of training set data points increases

Utility Error



The utility error decreases, and then stabilizes, with number of training set data points

Related Work

Dynamic Provisioning of Enterprise Applications:

- Tesauro et al.: Reinforcement learning, coarse-grained (host level)
- Chandra et al.: Closed model, fine-grained but for single server applications
- Urgaonkar et al.: Queueing model, coarse-grained (host level) for multi-tier applications
- Bennani et. al: Queueing model and search, coarse-grained
- Many works use queueing models for multi-tier applications, but no special provisioning for virtualization in the model

Summary

- Dynamic resource management crucial for server consolidation
- Development of adaptation policy rules a challenging problem
- Propose a hybrid approach based on offline modeling for policy generation
- Automatic offline policy generation is viable and can be accurate even with moderate computational effort

Thank you! Questions?