Building Adaptive Performance Models for Dynamic Resource Allocation in Cloud Data Centers

Jin Chen
University of Toronto

Joint work with Gokul Soundararajan and Prof. Cristiana Amza.
Today’s Cloud

Pay for your resources!

small, large, extra large instances

data stored, I/O rate, data transferred

$8 per user
Dream: QoS Cloud

Pay for your performance!

Average query latency < 1 sec

Customer: Our workload is usually stable, but there will be a few unpredicted peak times.

Cloud Admin: how many resources should we provision dynamically?
Cloud Admin: What If Question

What is the performance of this application given 2 CPUs, 4G RAM, 300 IOPS?
Challenges

● Performance interference
  - Between consolidated workloads
  - Uncontrolled resource sharing affects performance

● Increasingly complicated systems
  - Increase in # of resources, e.g. multi-level cache
  - System behavior varies, e.g. pre-fetching, background operations,…

● Workload itself causes variance/noise
Our Solution

- Build predictable systems in cloud
  - Partition critical resources
    - CPU, Memory, Network, Storage
- Build performance models for each app
  - Answer what if question on-line
- Dynamically allocate resources
  - Consolidate workloads in cloud
- Monitor and correct performance models
Build Predictable System in Cloud

Partition and allocate critical resources dynamically.

- Switch/Router
- Net Bandwidth
- Database Servers
- CPU, DB Buffer Pool
- Storage Array
- Storage Cache, Storage Bandwidth

Applications will have minimum interference from each other!
Performance Model for Application
Multi-level Resource Allocation

Allocate Resource to Meet SLOs per Application
Our Goal

• Challenge of building performance models
  - Large number of configurations to predict performance
    ● (CPU, Buffer Pool Size, Network, Storage cache, ...)
  - Increasingly complicated systems

• Build multi-level performance model with
  - Acceptable accuracy
  - In a short amount of time
  - Adapt to system changes
Outline

● Overview of different types of models
  – Analytical models
  – Black box models
● Our approach: Chorus
● Experimental results
Different Types of Models

Performance Models

Analytical
- Extensive domain knowledge
- Fast to get results
- Acceptable accuracy
- Difficult to adapt

Black Box
- Minimum domain knowledge
- May take a long time
- Higher accuracy
- Easy to adapt
Black Box Performance Models

2 level resources create a 3D surface

32x32=1024 samples, 11 days!

32 samples

Avg. Latency

Buffer Pool Size

Storage Cache

High Latency

Low Latency

May need long time to build black box models!
Outline

- Overview of different types of models
- Our approach: Chorus
  - Exploit incomplete expert knowledge
  - Leverage individual models
  - Automated the modeling process
  - Optimizations
- Experimental results
Exploit Partial Expert Knowledge

Performance Models

- Analytical
- Gray Box
- Black Box

- Deep domain knowledge
- Partial domain knowledge
- Minimum domain knowledge
- Difficult to adapt
- Easy to adapt
- Combine experimental samples
- Faster than black box models
- Easier to adapt than analytical
Expert Knowledge: Cache Inclusiveness

DB Buffer Pool

Storage Cache

LRU

Storage cache includes data in the buffer pool

I/Os: 6
Expert Knowledge: Cache Inclusiveness

DB Buffer Pool

Storage Cache

LRU

I/Os: 6

Buffer pool includes data in the storage cache
Approximate Single Cache Model (LRU)

I/Os: 6

Same Number of I/Os
Gray Box Multi-level Cache Model

\[ f(\text{CPU}, \text{Buffer pool size}, \text{storage cache size}, \text{storage bandwidth}) \]

Gray box model: Ignore the smaller cache size

\[ f(\text{CPU}, \text{Bigger Cache Size}, \text{storage bandwidth}) \]

Greatly reduce the # of configurations to predict!
Gray Box Curve Fitting Model

Analytical

\[ L_d(\rho_d) = \frac{L_d(1)}{\rho_d} \]

Gray box

\[ L_d(\rho_d) = \frac{\alpha}{\rho_d^\beta} \]
Outline

- Overview of different types of models
- Our approach: Chorus
  - Exploit incomplete expert knowledge: gray box model
  - Leverage individual models
  - Automated the modeling process
  - Optimizations
- Experimental results
Build Performance Model

Buffer Pool Size

Disk Bandwidth

Query Latency

11 days!

SVM

High Latency

Low Latency
Leverage Individual Models

Buffer Pool Size

Disk Bandwidth

Query Latency

5 days! 3 days (manually)!

Have we simplified the management problem?
Iteratively Training

- Expand Samples
- Rank Models
- Build Ensemble
- Build Perf. Model

Refine if necessary
Optimizations of Chorus

- Train models from history
  - Train new workload from similar saved workloads’ models
  - Similarity test; only train regions with low accuracy

- Prune configurations
  - Find the boundary configurations meeting SLA
  - Cut configurations with less resources than boundary ones
Outline

- Overview of different types of models
- Our approach: Chorus
  - Exploit incomplete expert knowledge: gray box model
  - Leverage individual models: ensemble learning
  - Automated the modeling process: iteratively training
  - Optimizations: history and pruning
- Experimental results
Evaluation Platform

MySQL
- CPU
- Buffer Pool

Storage
- NBD
- Cache
- Quanta
- Disk

Linux
- Block Layer
- NBD

Network

Linux
- Block Layer
- NBD
- /dev/sdb

SCSI

Disk
/dev/nbd1

Disk
Disk
Disk
Chorus Composition

• Gray box models
  - G-LR: region based linear model
  - G-INV: inverse shape based curve fitting model
Chorus Composition

• Gray box models
  - G-LR: region based linear model
  - G-INV: inverse shape based curve fitting model

• Analytical models
  - CPU and DISK

• Black box models
  - B-SVM: support vector machine regression
  - B-CR: use average as prediction results per region
Evaluate Prediction Accuracy

Accuracy: the percent of good predictions
e.g. 3/5 = 60%

Good predictions if predicted value in [0,1 std]
Orion Workload

Accuracy of Predictions

Percentage of Total Training Samples

Chorus
B-SVM
B-CR
G-INV
G-LR

1 ~ 2 STD
0 ~ 1 STD
Orion Workload

Composition of Chorus

<table>
<thead>
<tr>
<th>Percentage of Total Training Samples</th>
<th>15%</th>
<th>30%</th>
<th>60%</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-LR</td>
<td>10%</td>
<td>7%</td>
<td>10%</td>
</tr>
<tr>
<td>G-INVB-CRB-SVM</td>
<td>90%</td>
<td>93%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Legend:
- G-LR
- G-INVB-CRB-SVM
- B-CR
- B-SVM
TPC-C Workload

Accuracy of Predictions

Percentage of Total Training Samples

Chorus
B−SVM
B−CR
G−INV
G−LR

1 ~ 2 STD
0 ~ 1 STD
TPC-W Workload

Accuracy of Predictions

Percentage of Total Training Samples

15% 30% 60%

Chorus
CPU
DISK

1 ~ 2 STD
0 ~ 1 STD
Conclusion

- Key to implement QoS Cloud
  - Build predictable systems
  - Build accurate performance models

- Our Chorus can build accurate models:
  - Exploit partial domain knowledge: gray box model
  - Leverage individual models: ensemble learning
  - From scratch and from history
  - Allocate resources properly in QoS Cloud
Lessons and Future Work

• QoS Cloud is still difficult
  - Some workload is hard to model with high accuracy
  - May relax application goals, e.g. allow larger variations

• Feedback loop between admin. and Chorus
  - Chorus suggests new model, and admin verifies it.
  - Admin adds new model, and Chorus verifies it.
End.

Thanks!