TRACON
Interference-Aware Scheduling for Data-Intensive Applications in Virtualized Environments

Ron C. Chiang
H. Howie Huang
Department of Electrical and Computer Engineering
School of Engineering and Applied Science

Nov. 16, 2011
Cloud Service Revenue

Source: Gartner
Physical Machines

Virtual Machines
Performance Interference

- Normalized App1 runtime in VM1 while in VM2 running App2 consumes various CPU & I/O resources
- Small interference while competing for different resources
- Predictable from fair sharing when competing for CPU
- “Unpredictable” performance when competition comes to I/O
It Really Happens

- Amazon EC2 small instance
  - 1.7GB Memory
  - 1 EC2 compute unit
  - 160 GB instance storage

- Server slowdown by co-located data-intensive applications
TRACON
Task and Resource Allocation Control framework

I/O interference modeling
- Identify critical system parameters
- Quantify relations between performance metrics and parameters

Interference-aware scheduling
- Effectively utilize the interference models in scheduling
- Develop heuristic algorithms
  - Optimal job scheduling in parallel and distributed computing environment is NP-complete
Current Approaches for Performance Analysis

Mostly focus on

- **Non-virtualized** environments
- **Computation-intensive** applications
  - Model CPU utilization and performance

Examples

- **Q-Clouds**: deal with CPU bound workload
  [Nathuji et al 2010]
- **pSciMapper**: power-aware consolidation framework
  [Zhu et al 2010]
TRACON Architecture

- **Scheduler**
  - VM Query interference
  - Host info

- **Interference Models**
  - Generate models
  - Collect workload statistics

- **Model Training**
  - VM info
  - Query interference

**Hosts**

- Host
- Host
- Host
Model Variables

- **Performance metrics**
  - Runtime
  - **IOPS** (I/O throughput)
    - Important in exploring I/O interference

- **VM characteristics**

  - **domU**
    - CPU
    - Reads/sec
    - Writes/sec

  - **dom0**
    - CPU
    - dom0 handles domU’s I/O

**Model Training**

Dependent variable

Independent variables

Statistics of a target benchmark

Statistics of a series of microbenchmarks
Weighted Mean Method (WMM)

- Principle Component Analysis (PCA) transforms data into a number of uncorrelated variables, or called Principle Components (PC).
- PCs capture most important dynamics in the data.

- Calculate Euclidean distances between data points in the space spanned by the first four PCs.
- Choose three nearest data points and uses the reciprocal of their distances as the weights to get the predicted response.
Polynomial Models

• Linear models (LM)
• Non-linear models (NLM)
  • Use the Gauss-Newton method

• Model selection by stepwise algorithm
  • Iteratively fitting models with different sets of variables

• Evaluate goodness of fit with Akaike information criterion (AIC)
  • Provide the scores for examining the trade-off between model accuracy and flexibility

$$2 \times (\# \text{ of parameters}) - 2 \times \ln(\text{max. likelihood})$$

Lower AIC value indicates a better fit
# Data-Intensive Benchmarks

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Description</th>
<th>I/O intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastn</td>
<td>Bioinformatics</td>
<td>DNA sequence similarity searching</td>
<td>6</td>
</tr>
<tr>
<td>blastp</td>
<td>Bioinformatics</td>
<td>Protein sequence similarity searching</td>
<td>3</td>
</tr>
<tr>
<td>web</td>
<td>Server app.</td>
<td>Web server workload benchmark</td>
<td>2</td>
</tr>
<tr>
<td>email</td>
<td>Server app.</td>
<td>Email server workload benchmark</td>
<td>1</td>
</tr>
<tr>
<td>compile</td>
<td>Software dev.</td>
<td>Linux kernel compilation</td>
<td>4</td>
</tr>
<tr>
<td>dedup</td>
<td>System admin.</td>
<td>Data compression and deduplication</td>
<td>7</td>
</tr>
<tr>
<td>freqmine</td>
<td>Data mining</td>
<td>Frequent itemset mining</td>
<td>5</td>
</tr>
<tr>
<td>video</td>
<td>Multimedia</td>
<td>H.264 video encoding</td>
<td>8</td>
</tr>
</tbody>
</table>

- Can be extended to more benchmarks
- Light, median, and heavy I/O workloads are generated with the Gaussian dist.
Prediction Error

**Runtime prediction**

NLM has good prediction accuracy on runtime

**IOPS prediction**

NLM is also good on IOPS prediction

\[
\text{predicted value} - \text{actual value}
\]

\[
\text{actual value}
\]
Model Adaption

- Train initial *blastn* model on local storage
- Move to machines with iSCSI network storage

- Changing storage devices results in dramatic increases in prediction error
- Prediction errors are reduced by re-building models with new samples

![Graph showing prediction error vs. number of new samples](image)
Interference-Aware Scheduling

- Minimum Interference Online Scheduler (MIOS)
- Minimum Interference Batch Scheduler (MIBS)
- Minimum Interference miXed Scheduler (MIX)
## FIFO

<table>
<thead>
<tr>
<th>VM Pair</th>
<th>A+B</th>
<th>A+C</th>
<th>A+D</th>
<th>B+C</th>
<th>B+D</th>
<th>C+D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pred. runtime</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Total runtime = (A+B) + (C+D) = 20
Minimum Interference Online Scheduler (MIOS)

<table>
<thead>
<tr>
<th>VM Pair</th>
<th>A+B</th>
<th>A+C</th>
<th>A+D</th>
<th>B+C</th>
<th>B+D</th>
<th>C+D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pred. runtime</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

VM_A VM_B VM_C VM_D

Total runtime = (A+D) + (B+C) = 17
Recall: FIFO result is 20
Minimum Interference Online Scheduler

1. Take the first VM in queue
2. Get host machines status
   Repeat above procedure until no task in queue
3. Query interference

- Based on the minimum completion time heuristic
- Quick decision and small overhead
- Only consider limited assignments
Minimum Interference Batch Scheduler (MIBS)

- Based on the Min-Min heuristic
- Find a better match from tasks in batch

1. Take the first VM in queue
2. Find a VM has the least interference with the first VM (first min)
3. Get host machines status
4. Assign to the host with the least interference (second min)
Speedup by $\text{MIBS}_{\text{RT}}$ and $\text{MIBS}_{\text{IO}}$

**medium I/O** - Over 40% speedup
$\text{MIBS}_{\text{IO}}$ is better b/c it effectively increases I/O utilization

**light I/O** – 30% speedup from small interference

**heavy I/O** - Limited speedup b/c full utilization

---

**Number of physical machines**

- $\text{MIBS}_{\text{RT}}$-Light I/O
- $\text{MIBS}_{\text{IO}}$-Light I/O
- $\text{MIBS}_{\text{RT}}$-Medium I/O
- $\text{MIBS}_{\text{IO}}$-Medium I/O
- $\text{MIBS}_{\text{RT}}$-Heavy I/O
- $\text{MIBS}_{\text{IO}}$-Heavy I/O
Scalability

- Schedulers do work in a large scale
- MIBS has better scalability than MIOS
- With MIX as an upper bound, MIBS performs closely
Summary

I/O interference models

• I/O interference effects on runtime and IOPS can be better modeled by non-linear relations
• Dom0’s CPU is critical in virtual I/O interference modeling

Interference-aware schedulers

• 50% improvement on total job runtime
• 80% improvement on job throughput

Future works

• Online modeling and scheduling
• Real-world implementation and testing
Thank You

Emails: {rclc, howie}@gwu.edu