D-FACTOR: A Quantitative Performance Model of Application Slow-down in Multi-Resource Shared Systems

Presenter: Youngjae Kim
June 14th 2012

Seung-Hwan Lim¹,², Jae-Seok Huh¹, Youngjae Kim¹, Galen M. Shipman¹, and Chita R. Das²

¹Oak Ridge National Laboratory
²Pennsylvania State University
A norm in a computing system: multiple concurrent workloads

Enterprise-scale system: server consolidation

Desktop system or Smartphone: multiple programs

Computing systems are running multiple workloads. Applications slow down due to resource contentions.

How can we estimate the slow-down of multiple concurrent workloads in multi-resource systems?
Estimating the slow-down of applications due to interference.

Empirical Method
- Measure the slow-down with other workloads.
  - Representative workloads
  - Statistically similar workloads

Analytical Method
- Queuing model
  - Based on well-established theory.
  - However, to enhance accuracy more detailed information on resource usage is often required.
- Linear Sum
  - The simplest analytical model

We extend the linear sum model to estimate the slow-down of applications due to resource contention.
The non-linear slow-down in multi-resource systems

Experiments

CPU workload: CPU job consists of arithmetic operations only
Dedicated to run on a single-core CPU

I/O workload: Each I/O job randomly reads two 2GB of files (RAM = 4GB)
Both CPU and I/O workloads take 100 sec without the presence of other workloads.

Linear sum model fails to explain multi-resource contention.
D-Factor (*Dilation Factor*) model

Estimates the slow-down of jobs due to contention for multiple resources in a system
D-factor model extends linear sum.

Objective

We want to describe the slow-down of applications in multi-resource systems.

Design Constraints

To maintain the simplicity instead of the perfection.
To easily use in existing schedulers.

Our Approach

We extend the linear sum model. However, it has the following limitation.

The linear sum is for single-resource systems.
However, the basis of many scheduling algorithms requires to consider multi-resource system environment.
An Overview of D-factor Model Framework

D-factor model explains the expected slow-down when applications are concurrently running.

$\lambda$ is a quadratic function of loading vectors in the D-factor model.
Outline

Introduction

How to describe jobs and machines
- Dilation factor; job and job slices; and loading vector

How to estimate running times

Validation results

Conclusions & Future work
Each fraction of a job will be dilated by resource contention.

Stand-alone behavior

Co-located behavior

*System model: Single CPU system
Dilation Factor, $\lambda$

$$\lambda = \frac{\text{Running Time w/ Other Jobs}}{\text{Stand-Alone Running Time}}$$

$$\lambda_1 = \lambda_2 = \frac{7}{5} = 1.4$$
**Dilation Factor**
Slow-down due to resource contention

**Definition 1: Dilation Factor**
Dilation factor $\lambda$ is the expectation of the factor of dilated completion time due to the resource contention, denoted by

\[
\text{Dilation Factor } \lambda = \frac{T}{\tau}
\]

- Running time with other jobs
- Stand-alone running time
Machine: serves multiple jobs with shared system resources

A portion of job-1 at a certain time

CPU
- Job 1
- Job 5

Memory
- Job 2

Disk I/O
- Job 3
- Job 6

Network I/O
- Job 4

A job may contend for multiple system resources with other jobs in its overall execution.
Definition 2. Job slice and Job

Job slice: a hypothetical fraction of a job that accesses one resource

Job: a sequence of job slices

Assumptions

• A job is a sequence of job slices.
• A job slice accesses only one resource for a hypothetical one-unit time.
• The service time of each job slice does not change by interference.
• No idle period between job slices.
• Jobs are independent to each other, i.e., different processes.
Job: described by resource access probabilities

2 Resources (CPU and I/O) in a system

Job slice: accesses single resource.

$P_i = (P_{cpu}, P_{I/O})$

$P_1 = (0.6, 0.4)$
Resource probability vector $P_1$ for Job 1

$P_2 = (0.4, 0.6)$
Resource probability vector $P_2$ for Job 2
Definition 3. **Loading vector**: A loading vector consists of elements that represent the portion of time in accessing each resource during execution of a job.

Loading vector: the statistical characterization of a job

\[ p_1 = (0.6, 0.4) \]
**Loading Matrix**: Describes the Set of Jobs in a System

$n$ jobs

$m$ resources

$j^{th}$ job

Probability of accessing resource $i$ by job $j$ during its execution

Loading vector of job $j$, $p_j$

Total loading vector, $\bar{p} = \sum_{j=1}^{n} p_j$
Outline

Introduction

How to describe jobs and machines

How to estimate running times

• An example: n-jobs in 2-resource
• By-products
  • How to obtain loading vectors of jobs
  • How to reduce to linear sum

Validation results

Conclusions & Future work
**Dilation Factor Theorem**

**Theorem 1:** Given a job set on a machine characterized by the loading vectors $p_j$, the dilation factors, $\lambda_j = \frac{T}{\tau}$, are given by

$$\lambda_j = 1 + p_j \cdot \bar{p} - p_j \cdot p_j$$

- **Factor of the service time of the job without interference**
- **Sum of the probability of interference with all the jobs**
  $$\bar{p} = \sum_{j=1}^{n} p_j$$
- **The probability of the interference with itself**

**Intuitions:**
Due to the resource contention, each job slice will be dilated such that from $\delta$ to $\delta +$ waiting time while other jobs are served in the resource.
Theorem 1: Given a job set on a machine characterized by the loading vectors $p_j$, the dilation factors, $\lambda_j = T / \tau$, are given by

$$\lambda_j = 1 + p_j \cdot \bar{p} - p_j \cdot p_j$$

This job slice will only wait for job 1’s job slice.

The processing time of job j’s job slice dilates according to the probability of resource contention.
2-Resource, \( n \) Identical Jobs

**Theorem 2:** Assume \( n \) 2-resource identical jobs with the loading vector given by \((p, 1-p)\). Then, the dilation factors are identically given by

\[
\lambda = 1 + (n - 1)(p^2 + (1 - p)^2)
\]

Intuitions: When we take non-requested resources out of consideration, the loading vector \( p = (p, 1-p) \).
How to profile applications

Measure the resource usage

• Not discussed in this study.

Measure the slow-down with two instances of the application.

Measure the slow-down with another well-known application.

• Included in this study.
Obtain $\lambda$ from measurements

Obtain the element of resource-1, $p$

Obtain the vector $p = (p, 1-p)$

Measure $\tau$ by running one instance of job $j$

Measure $T$ by running $n$ instances of job $j$

Substitute $\lambda$ into the equation

Solve a quadratic equation

Dilation Factor $\lambda = \frac{T}{\tau}$
1-resource jobs: linear completion time

**Theorem 3:** Given a job set, $J$, on a machine with only one resource, the total completion time of jobs, $T(J)$ is given by the linear sum of individual job completion times, that is,

$$T(J) = \sum_{j \in J} T(j)$$

Linear sum is a special case of the dilation factor theorem
Dilation Factor is the slow-down.

We explain the relationship between the job profile, loading vector $p$ and dilation factor $\lambda$.

We demonstrate that we can profile jobs and estimate slow-down of jobs before locating them.

Measure $T$ and $\tau$ to obtain job profile.

Dilation Factor $\lambda = \frac{T}{\tau}$

With $p$ and $\tau$, estimate $T$

Running time with other jobs

Stand-alone running time
Outline

Introduction

How to describe jobs and machines

How to estimate running times

Validation results
  • Workloads
  • System specification
  • Synthetic workloads
  • Application Benchmark : FileBench (fileserver/varmail)
  • MapReduce : identical jobs/non-identical jobs

Conclusions and Future work
Validating D-Factor Model

D-factor model can provide

1. More accurate estimation of the completion times of co-hosted jobs than the linear sum model
2. More efficient utilization of the system resource
3. Better predictable performance with existing scheduling algorithms than with the linear sum model

Experimental Setup

Experimented with synthetic and realistic workloads
Experimented on native Linux and Xen-based VM environment
Ran 40 times for each case and presented average values
# Description of Workloads

<table>
<thead>
<tr>
<th>Workload</th>
<th>CPU</th>
<th>Mem</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>I/O</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>FileBench</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fileserver</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Mailserver</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>MapReduce</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort (1GB)</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Grep</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>PiEstimator</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>
# System specification

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Two single-core 64bit AMD 2.4GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>4GB</td>
</tr>
<tr>
<td>Shared Storage</td>
<td>NFS, disk images for Xen</td>
</tr>
<tr>
<td>Local Storage</td>
<td>Ultra320 SCSI</td>
</tr>
<tr>
<td>Network</td>
<td>1Gbps Ethernet to NFS, 10Gbps Infiniband between nodes</td>
</tr>
<tr>
<td>vCPU (Dom0)</td>
<td>Runs on both CPUs</td>
</tr>
<tr>
<td>vCPU (VMs)</td>
<td>Runs on one CPU</td>
</tr>
<tr>
<td>RAM/VM</td>
<td>256MB</td>
</tr>
<tr>
<td>I/O (VM)</td>
<td>TAP:AIO (bypasses buffer cache of Dom-0)</td>
</tr>
<tr>
<td>Kernel</td>
<td>Linux 2.6.18</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>Xen 3.4.2</td>
</tr>
</tbody>
</table>
Validation: Synthetic workloads

CPU: consists of arithmetic operations only
IO: reads two 2GB files

Native Linux

Virtualized environment

Completion Time (sec)

Completion Time (sec)

CPU: consists of arithmetic operations only
IO: reads two 2GB files
Validation: FileBench workloads

Each workload hosted in separate virtual machines.

D-factor can estimate the slow-down of each job while Linear sum can’t. Recall that D-factor is an extension of Linear sum.
Validation: MapReduce workloads

Identical workloads often show the same phased behavior, which is hard to be explained with D-factor, which increases error rates as the number of instances increases.

A 17 node Hadoop cluster results (1 master, 16 slaves)

Identical MapReduce

<table>
<thead>
<tr>
<th>Completion Time (sec)</th>
<th>1sort</th>
<th>2sort</th>
<th>3sort</th>
<th>4sort</th>
<th>1grep</th>
<th>2grep</th>
<th>3grep</th>
<th>4grep</th>
<th>1pi</th>
<th>2pi</th>
<th>3pi</th>
<th>4pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-factor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error < 16 %
Validation: MapReduce workloads

A 17 node Hadoop cluster results (1 master, 16 slaves)

Since heterogeneous workloads are more independent than identical workloads, error rates becomes smaller than identical workloads.
Summary

Performance Model:

We proposed a novel completion time model of jobs for shared service systems

We modeled a job by a resource usage vector, called **loading vector**

We showed that dilation factor of application slow-down can be modeled in a quadratic function of loading vectors.

Model Validation

We validated our proposed model with experiments using synthetic and realistic workloads.

How to use the Model in systems

We showed how to profile jobs and estimate the overall completion times of jobs in shared service systems
Future Work

Extending space-shared resources
  (e.g., memory caches)

Developing a job scheduler with D-factor model

More validation with multi-core system
Questions?

Contact info

Youngjae Kim (PhD) / Seung-Hwan Lim (PhD)

kimy1@ornl.gov / lims1@ornl.gov

Oak Ridge National Laboratory