An Infrastructure for Automating Large-scale Performance Studies (Elba under-the-hood)

Calton Pu
Professor and J.P. Imlay Chair in Software
CERCS, Georgia Institute of Technology

Many PhD, MS, Undergrad students
Many company collaborators and support from several companies, particularly from Fujitsu and Intel.
Importance of Predictable Performance

• Extra delay of just 100ms could result in roughly 1% loss in sales.

• Additional delay of just 500ms could reduce revenues by 20%.

• IDC reported performance to be top 3 user considerations for Clouds.
Cloud Computing & Performance

• Cloud is a **black-box** for many users.
• Application providers face non-trivial performance challenges.
• One of the most effective ways to understand a cloud is to measure it:
  – Run measurement studies and collect data
  – Maybe it’s the only way
Large-scale Experimental Measurements

• **Goal:** Use large-scale experimental data as “real” (predictive) models for performance and scalability constraints;

• **Challenges:**
  – Deployment complexity due to configuration dependencies
  – Large state space: Many configuration options
  – Huge amount of data: >1GB/experiment, semi-structured data
Automating Large-scale Experiments

– Create
  • Prepare the platform, deploy and configure application.

– Manage
  • Start application, execute workload, data collection.

– Analyze
  • Data analysis (visualization) and building hypothesis.
Performance Measurement Workflow

- Code Generator
- Experiment Driver
- File Store
- Experstore
- Data Extractor & Data Processor
- Data Analyzer
Expertus – Code Generator

• **Idea**: Generate scripts to automatically create, manage and analyze the experiments from user-friendly specification files.

• Key Challenges:
  – From abstract mapping to concrete scripts
  – Heterogeneity of hardware and software components
  – Flexible customization needed in experiments

• Solution platform: XML + XSLT + AOP
Code Generation Pipeline
Template Types

• Base Templates
  – To generate OS, cloud, and user independent resources.
  – A template for each possible action (e.g., deploy-tomcat) and resource (e.g., httpd.conf).
  – Created by identifying output variances (each of which becomes an aspect).

• Aspect Templates
  – Customize to meet application, cloud, and user needs.
  – Can contain one or more advices.
  – Nested pointcuts – an advice can add zero or more pointcuts.
Expertract - Automated Data Extractor

• Performance logs from various monitors, semi-structured
  – Potentially, a custom data parser for each experiment

• Log file format with many variations
  – Monitoring tools (e.g., dstat, sar, o-profiler ...) and parameter settings

• ETL (Extract, Transform, Load) tools insufficient by themselves
  – Need to figure out the actual log layout
Most Common File Formats

- One header
- Multiple headers with sequentially corresponding data
- Multiple headers with non-sequential corresponding data
- Multiple headers appear randomly in the file and data is entirely non-sequential
Experstore - A Flexible Data Warehouse

• Tables are created *on-the-fly* based on the data.
• Why not static tables? (global schema too big)
  – Several monitoring programs
  – Many different parameter settings: e.g., 2 core vs. 4 cores
• Why not column based tables?
  – Would be too many tables (over 20000 tables per experiment).
  – (# Workload) * (# Nodes) * (# Resources).
• Our solution – A hybrid approach:
  – Create small tables to store related data, for example a table to store CPU data that consists of user, sys, idle etc...
Static and Dynamic Tables

- Experiment
- Nodes
- Workloads

Resource Mapping (Dictionary)

- CPU
- Memory
- ApacheLogs
- StoriesOfTheDay
Data to Schema Mapping

• Mapping performance data to a table (to-be-created) in the data warehouse.
  – Which columns (row) to read ?.
  – How to format (e.g., datetime) ?.
  – What to include/exclude ?.
  – Which parser to use ?.
• Specifying resources for a given node.
  – e.g., CPU1, CPU2, network, IO etc ...
• Mapping result directories to workloads.
  – e.g., “2009-11-29@11-25-40” → 1000-RO
• Mapping log files to a node/resources.
  – e.g., mod_jk.log → request processing time at Apache.
Data Analysis - Web Portal

• Aid the analysis of large amount of data.
• Identify patterns, trends and relations.
• Control the way in which data is represented.
• Data from seemingly unrelated sources could be easily compared against each other.
Web Portal-2D
Web Portal-3D

3D graph for 'Graph - 4'
Z: 1:1
Y: 1:10
X: 1:1
Support for R Framework

Generated Script

drv <- JDBC("com.mysql.jdbc.Driver",
    "C:/mysql/mysql-connector-java-5.1.7-bin.jar",
    identifier.quote="``")

conn <- dbConnect(drv,
    "jdbc:mysql://elbafs.cc.gatech.edu:3313/elba",
    "elba", "elba")

d = dbGetQuery(conn, "select user from TAB133397700575200219B_CPU0 where dictionaryid='334'")

hist(d$user, breaks=20, col="white",
    xlab="CPU Utilization",
    main="Histogram for CPU (user)"")
Wide Applicability

- Over 500 different hardware configurations.
- Over 10,000 software configurations.
- Over 100,000 nodes.
- Many clouds (e.g., Emulab, EC2, OpenCirrus, Georgia Tech cluster, and Wipro).
- Many representative applications (e.g., RUBBoS, RUBiS, CloudStone, and over 10 OLTP benchmarks).
## High-Level Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Emulab</th>
<th>EC2</th>
<th>OpenCirrus</th>
<th>Elba</th>
<th>Wipro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiments</td>
<td>9215</td>
<td>1436</td>
<td>480</td>
<td>3567</td>
<td>120</td>
</tr>
<tr>
<td>Nodes</td>
<td>102687</td>
<td>25848</td>
<td>4987</td>
<td>9865</td>
<td>430</td>
</tr>
<tr>
<td>Configurations</td>
<td>392</td>
<td>86</td>
<td>28</td>
<td>163</td>
<td>8</td>
</tr>
</tbody>
</table>
Specification Changes vs.
Changes in Generated Code

Lines Changed – Specification

Lines Changed – Generated Code
Number of Nodes vs. Generated Code

- Total Number of Lines vs. 
- # Nodes

- 16
- 20
- 35
- 43
- 65
- 86

- Other
- XML
- Shell Script
2013 Elba Publications

Ongoing Work

• Extending the data parser to support additional data formats.
• Extending the data warehouse to use No-SQL databases.
• Extending the visualization tool to support more customizable graphing capabilities.
Conclusions

• Help researchers efficiently creating, storing and analyzing performance measurement data.

• Open new opportunities and enable large-scale experiments above and beyond manual application testing.