Introduction

Vojtěch Horký Peter Libič Petr Tůma

2010 – 2023

This work is licensed under a "CC BY-NC-SA 3.0" license. Created to support the Charles University Performance Evaluation lecture. See `http://d3s.mff.cuni.cz/teaching/performance-evaluation` for details.

Contents

1 Introduction

Syllabus

Purpose

It is all about measuring how fast (or slow) your software runs and what can you do to figure out why.

- Goals of performance evaluation
- What to measure
- How to measure
- How to process the data
- How to present the data
- Simulation and modeling

Course Outcome

Experiment Design
- Selecting relevant metrics
- Using appropriate measurement procedures
- Selecting appropriate workload configuration

Experiment Evaluation
- Selecting appropriate method (statistical and visual)
- Understanding the results of statistical methods
- Presenting the results to others

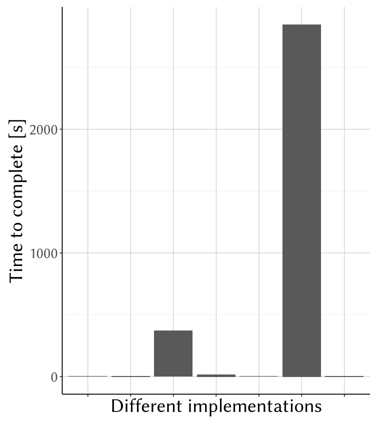
Simulation and Modeling
- Understanding the principles and uses

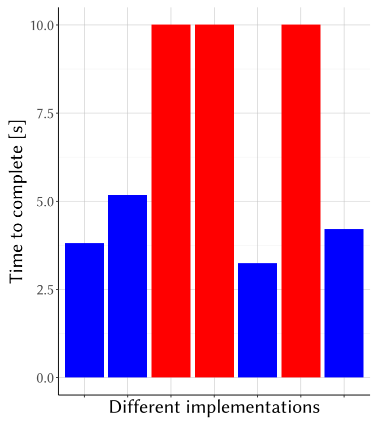
– Understanding the limits of the methods

How Hard Can It Be ?

Experiment

Implement an application that reads an XML file with <section> and <xref> tags, and outputs a list of cross references grouped per section.





[1]

How Hard Can It Be ?

Experiment

Select the faster of two (black box) implementations of the same interface.

2 Adminstration

Prerequisites

Computer Systems

Compilers Operating Systems Computer Architecture Operating Systems Course (NSWI004) is more than enough

Statistics

Statistics useful but not required Only applied knowledge R system

[1]Based on Horký et al.: Utilizing Performance … doi:10.1145/2668930.2688051

Programming

Examples will be mostly in Java and C

Activities

Self Study

In general not much self study required. Might provide a paper or two to read before some (later) lectures.

Lectures

Me showing slides (boring) and hopefully you discussing (exciting) !

Labs

Once roughly every two weeks, check webpage for dates. Assignments to practice lecture content.

Grading

Labs

Each assignment earns 0-10 points. Need 80 % to earn course credit. Backup assignments not available !

Exam

Prepare and execute a performance evaluation experiment of your choice. During the exam, you will present the experiment and the results. Exam discussion will be focused on:

- Understanding of the presented methods
- Correct evaluation of the experiment results
- Basic knowledge of the internals of the used tools

Study Materials

Online

Slides available on the webpage as slides and handouts. Links to papers occasionally provided on the slides. School digital library subscriptions should work. Other common resources:

- Tool documentation (R, PAPI, perf, ...)
- Blogs (Alexey Shipilev, Cliff Click, ...)

Books

- Recent
  - Gregg: Systems Performance
  - Kounev et al.: Systems Benchmarking ...
- Classics
  - Lilja: Measuring Computer Performance ...
  - Jain: The Art of Computer Systems Performance Evaluation

Contact

- Petr Tůma petr.tuma@d3s.mff.cuni.cz

Webpage

`http://d3s.mff.cuni.cz/teaching/nswi131`

Acknowledgments

Contributors to earlier versions of the course and the materials:

- Vlastimil Babka
- Lubomír Bulej
- Vojtěch Horký
- Tomáš Kalibera
- Peter Libič

3 Motivation

What is it about ?

Performance Evaluation of Computer Systems

Performance

Expressed using mostly time related metrics such as response time, latency, throughput (operations per second), but also more exotic metrics such as memory footprint, power consumption, thermal budget.

Evaluation

Measurement of real or modeled systems. Analysis of (mathematical) models.

What is it about ?

Performance Evaluation of **Computer Systems**

Computer Systems

We are interested in performance of *software applications.* But it includes effects of other components, which are difficult to eliminate from the analysis, like:

- Libraries
- Virtual machine
- Operating system, hypervisor
- Hardware (processor, memory, caches ...)

Why take the course I

Development

Making appropriate decisions in development. How to decide which library, algorithm or optimization is more suitable for the application ? Measurements possibly tricky ... Problems:

- Repeated measurement results change. Is the difference just noise or is one solution really better ?
- Which of the different results is representative ? How many measurement repetitions should we execute ?
- What is it we see in the results ? Is it server response time or network capacity ?

Why take the course II

Operations

Tuning parameters for best performance. For example number of threads, heap memory size, network buffer size. Identification of bottlenecks. Problems:

- Ranking alternatives in presence of noise.
- Knowing what to configure and measure.

Show `java -XX:+PrintFlagsFinal` to illustrate tunables.

Why take the course III

Research

Research hypotheses require validation. In complex systems theoretical validation may not be feasible. We are then left with conducting experiments. Problems:

- How to derive general conclusions from specific experiments ?
- How to make experiments reasonably reproducible ?
- How to avoid wrong conclusions ?

4 Goals of Performance Evaluation

High Level Goals

Performance evaluation is usually not a self-serving goal.

General Systems

- Create a system with best possible performance under given constraints (efficiency).
- Create a system with given performance at minimum cost (capacity planning).

Real-Time Systems

- Create a system that will (always) respond within deadline.
- Create a system that allows easy worst-case execution time analysis.

We are interested mostly in experimental methods to achieve these goals.

Tasks Solved Using Performance Evaluation

Typical tasks solved using performance evaluation include:

- Assessing Performance Cost of Features
- Comparing of Design Alternatives
- Guiding System Tuning
- Testing Performance
- Debugging Performance
- Estimating Worst Case Performance
- …

For each task, different techniques may be useful.

Assessing Performance Cost of Features

Task

Measure or estimate what (performance) impact would adding a feature have. Also possibly viewed as comparing system before and after adding a feature.

Example questions:

- Does it make sense to add more computing units ? caches ? memory ?
- Does it make sense to add or remove a specific compiler optimization ?
- What is the performance impact of upgrading or replacing a software component ?

Typical problems:

- What if the system is not yet available ?
- How to detect and express platform dependent results ?

Comparison of Alternatives

Task

Select the best of available design or configuration alternatives. Rarely single criterion (speed), typically compromise (speed, cost, power consumption, maintenance cost).

Example questions:

- What is the best hardware configuration (from a given list) for an application ?
- What components (libraries, operating system, database) to use for an application ?

Typical problems:

- What if the system is not yet available ?
- Interaction between alternatives (software selection may depend on hardware).

Guiding System Tuning

Task

Given a configurable system, guide the tuning procedure to achieve optimum configuration. Done mostly during or after deployment rather than during development.

Example questions:

- What configuration settings matter ?
- How to set a particular parameter (buffer size, thread pool size, heap size, scheduler parameters, ...) ?

Typical problems:

- Interaction between settings.
- Impacts can be counterintuitive.
- Testing all possible combinations is often impossible, too many settings and too many legal values.

> Use `java -XX:+PrintFlagsFinal` to see JVM tunables in effect.

> See `https://chriswhocodes.com` for a list of JVM command line options.

Testing Performance

Task

Given a system, develop and execute a test suite that provides developer feedback. Testing possible at many levels such as unit testing or integration testing.

Example questions:

- Does a particular component meet the performance requirements ?

Typical problems:

- What are the actual requirements ?
- Too many measurements take too much time.
- Handling platform dependent requirements difficult.
- Realistic test conditions needed for realistic measurements.

Debugging Performance

Task

Given a (deployed) system that may exhibit performance anomalies, detect the anomalies, locate the causes and develop fixes.

Example questions:

- Is the observed performance reasonable ?

- What are the slow configurations or instances ?
- What are the performance critical system elements ?
- And the killer question: How can this be happening ?

Typical problems:

- Measuring a live system can distort behavior.
- How to develop smaller test cases that reveal the problem.
- Figuring out what is happening requires extensive knowledge.
- We want to avoid rather than understand problems.

Estimating Worst Case Performance

Task

Help estimate (induce and measure) the worst possible performance. This is tricky because real-time system design needs hard guarantees.

Example questions:

- What workload will trigger the worst possible performance ?
- Does an optimization improve the worst case ?
- Can any completeness guarantees be made ?
- How to design for testability ?

Typical problems:

- Guarantees (very) hard to get.
- Worst case performance not composable.
- Proper randomization of experimental conditions

5 Techniques of Performance Evaluation

Techniques Overview

Range of applicable techniques depends on circumstances.

Lifecycle Phase

- Design time with no implementation
- Development time with partial implementation
- Deployment time with full implementation and running application

Available Information

- Design plans with estimates
- Actual software implementation
- Actual performance measurements

Constraints

- Accuracy requirements
- Portability requirements
- Cost of conducting experiments

Performance Modeling

Create a Model of the System

- Mathematical (statistical) model or representation
- Reflects only selected characteristics of the system or environment
- Common model types include Queueing Networks, Petri Nets, or even low level models such as Markov Chains

Evaluate (Solve) the Model

- Analytically evaluate performance properties of the model
- If analytical solution is impossible or too expensive, use simulation or numerical methods

Simple Model Example

How many service desks should a bank have to service 60 customers per hour if it takes 5 minutes to service one customer ?

Queueing Theory model with Little's Law. $Utilization = Throughput \times ServiceTime$

- Assume stable system
- Throughput 1 per minute
- Utilization $1 \times 5 = 5$
- Serving on average 5 customers in parallel

Performance Modeling Pros and Cons

Pros

- Can reveal problems at design time
- Repeatable (if the model is stable)
- Possibly can analyze corner cases (worst case)
- Relatively fast and cheap (unless complex math required)

Cons

- It is easy to design models too complex to be analyzed
- Too simple models may be ignoring important factors
- Many factors are unknown (closed source or poor documentation)
- And all this implies low accuracy and therefore low trust in results

Simulation

Simulate Missing Parts of the System

- Users (humans) interacting with the system
- Missing SW or HW components interacting with the component of interest
- Expensive or not yet available HW the system will run on (emulation)
- Or even the whole system

And assume the results are representative of real system.

Simulation Pros and Cons

Pros

- Can take into account more details than (analytical) modeling, making the results more realistic
- Can reduce the cost by substituting missing or expensive parts
- Flexible, allows parameter space search
- For some cases, simulator can monitor more metrics than real device

Cons

- To reduce simulation time, the precision often has to be reduced
- If realistic, not always repeatable
- Slower and more expensive than modeling
- Potentially error-prone, errors hard to detect (for example random number generator problems)

Performance Measurement

Measuring (Parts of) the Implemented System

- Either exactly in the way it will be used

- Or using common or expected workload

Analyzing the Results

- Repeating measurements (or their parts) if needed
- Estimating precision using statistical methods
- Presenting the results in aggregate numerical or visual form (plots)

Complete System Measurement

Measurement of the whole (potentially simulated) system using representative workload

- Most reliable results
- High cost, needs implementation, may take a long time

Separate Component Measurement

Each component is measured separately, possibly in isolation. This entails the following steps:

- Devising performance test for an individual component
- Characterizing the demands of the application on the component
- Putting these two together (especially difficult to assess interactions)

Performance Measurement Pros and Cons

Pros

- Result are (possibly) most representative of the real systems
- Systems with closed source or specification are not a problem
- Relatively simple analysis

Cons

- Needs finished implementation and real hardware
- Harder to identify causes of the results
- Some metrics difficult to measure
- Needs careful experiment design
- May not reveal corner cases
- Results are hard to generalize