EECS 4314

Advanced Software Engineering

Topic 13:

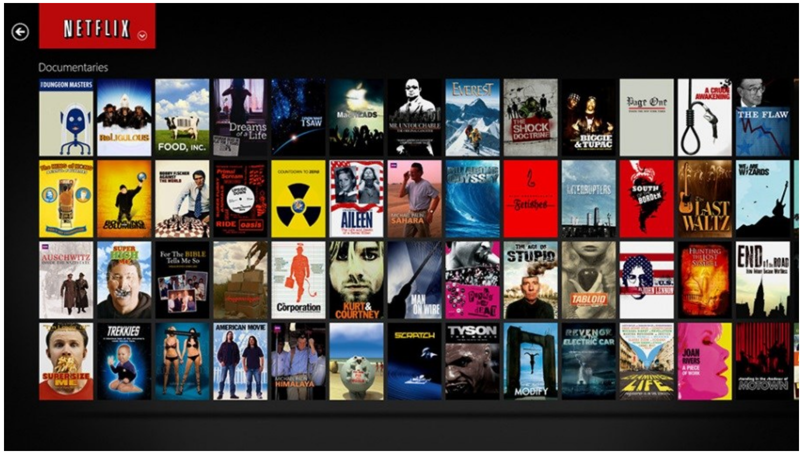
Software Performance Engineering

Zhen Ming (Jack) Jiang

Acknowledgement

- Adam Porter
- Ahmed Hassan
- Daniel Menace
- David Lilja
- Derek Foo
- Dharmesh Thakkar
- James Holtman
- Mark Syer
- Murry Woodside
- Peter Chen
- Raj Jain
- Tomáš Kalibera
- Vahid Garousi

Software Performance Matters



What is Software Performance?

"A performance quality requirement defines a metric that states the amount of work an application must perform in a given time, and/or deadlines that must be met for correct operation".

- Ian Gordon, Essential Software Architecture

Performance Metrics (1)

- Response time
  - a measure of how responsive an application or subsystem is to a client request.
- Throughput
  - the number of units of work that can be handled per unit of time (e.g., requests/second, calls/day, hits/hour, etc.)
- Resource utilization
  - the cost of the project in terms of system resources. The primary resources are CPU, memory, disk I/O and network.

Performance Metrics (2)

- Availability
  - the probability that a system is in a functional condition
- Reliability
  - the probability that a system is in an error-free condition
- Scalability
  - an application's ability to handle additional workload, without adversely affecting performance, by adding resources like CPU, memory and disk

Common Goals of Performance Evaluation (1)

Evaluating Design Alternatives

- Should my application service implement the push or pull mechanism to communicate with my clients?

Comparing System Implementations

- Does my application service yield better performance than my competitors?

Benchmarking

Common Goals of Performance Evaluation (2)

Performance Debugging

- Which part of the system slows down the overall executions?

Performance Tuning

- What are the configuration values that I should set to yield optimal performance?

Common Goals of Performance Evaluation (3)

Performance Prediction

- How would the system look like if the number of users increase by 20%?

what-if analysis

Capacity Planning

- What kind of hardware (types and number of machines) or component setup would give me the best bang for my buck?

Common Goals of Performance Evaluation (4)

Performance Requirements

- How can I determine the appropriate Service Level Agreement (SLA) policies for my service?

Operational Profiling

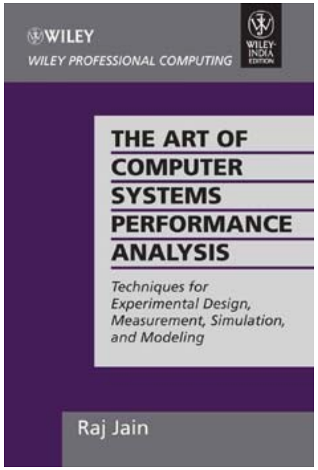
- What is the expected usage once the system is deployed in the field?

workload characterization

Performance Evaluation v.s. Software Performance Engineering

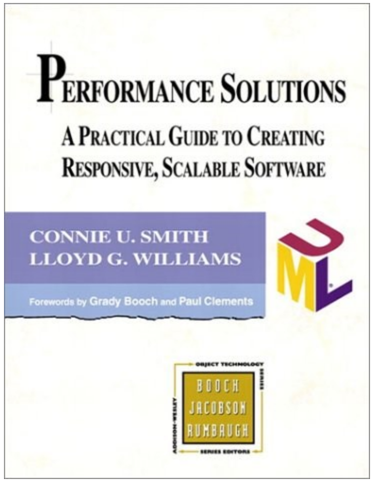
- *"Contrary to common belief, performance evaluation is an art. Like a work of art, successful evaluation cannot be produced mechanically."*

  - Raj Jain, 1991



- *"[Software Performance Engineering] Utterly demystifies the job (no longer the art) of performance engineering"*

  - Connie U. Smith and Lloyd G. Williams, 2001



When should we start assessing the system performance?

"It is common sense that we need to develop the application first before tuning the performance."

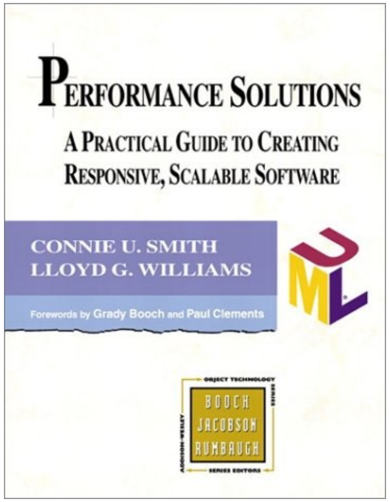
- Senior Developer A

Wrong!

Many performance optimizations are related to the system architecture. Parts or even the whole system might be re-implemented due to bad performance!

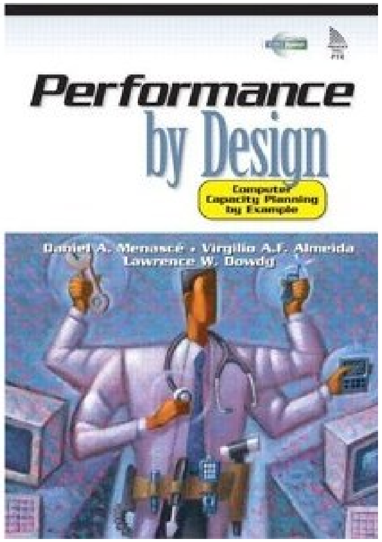
We should start performance analysis as soon as possible!

Originated by Smith et al.



To validate the system performance as early as possible (even at the requirements or design phase)

"Performance By Design"

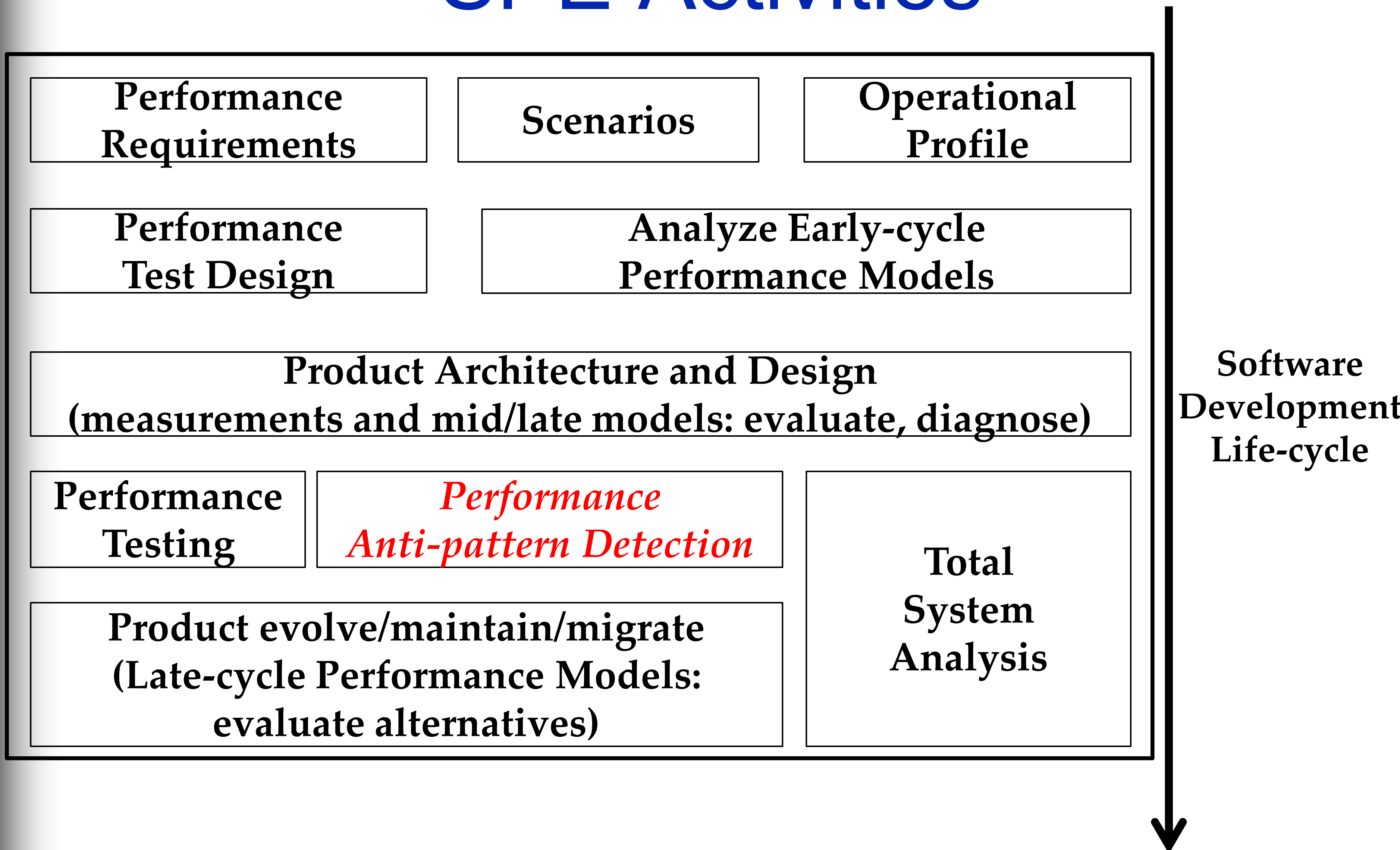


Software Performance Engineering

<u>Definition:</u> Software Performance Engineering (SPE) represents the entire collection of software engineering activities and related analyses used throughout the software development cycle, which are directed to meeting performance requirements.

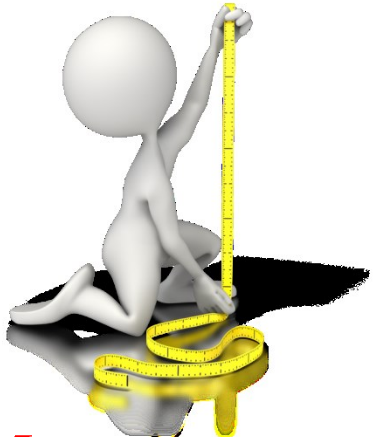
- Woodside et al., FOSE 2007

SPE Activities



[Woodside et al., FOSE 2007]

Three General Approaches of Software Performance Engineering



Measurement

Usually applies late in the development cycle when the system is implemented

Analytical Modeling

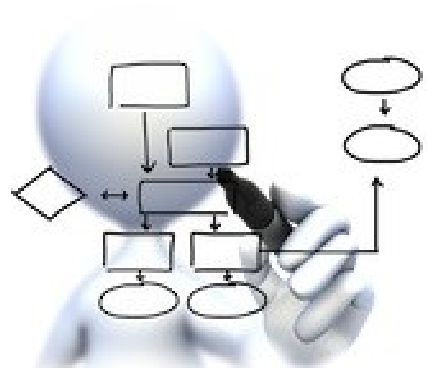
Usually applies early in the development cycle to evaluate the design or architecture of the system

Simulation

Can be used both during the early and the late development cycles

Three General Approaches of Software Performance Engineering

Measurement

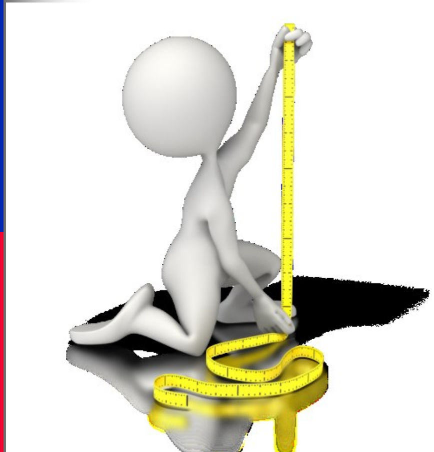


Analytical Modeling

Simulation

	Approaches		
Characteristic	**Analytical**	**Measurement**	**Simulation**
Flexibility	High	Low	High
Cost	Low	High	Medium
Believability	Low	High	Medium
Accuracy	Low	High	Medium

Three General Approaches of Software Performance Engineering



Measurement

Usually applies late in the development cycle when the system is implemented

Analytical Modeling

Usually applies early in the development cycle to evaluate the design or architecture of the system

Simulation

Can be used both during the early and the late development cycles

Convergence of the approaches

Books, Journals and Conferences

CMG Computer Measurement Group

spec

PERFORMANCE EVALUATION
An International Journal

ICSE 2013 San Francisco

ICSME 2014

ISSTA

ACM SIGMETRICS
special interest group on performance evaluation

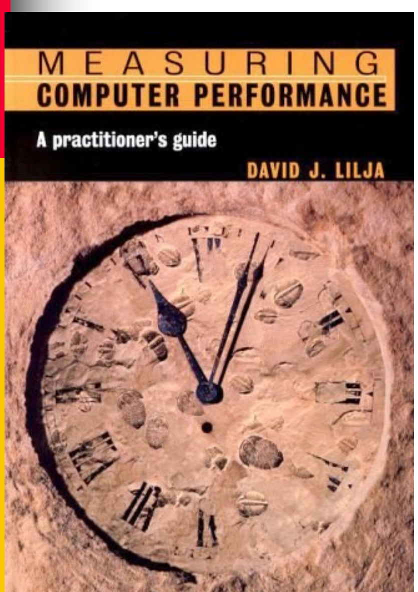
ICST

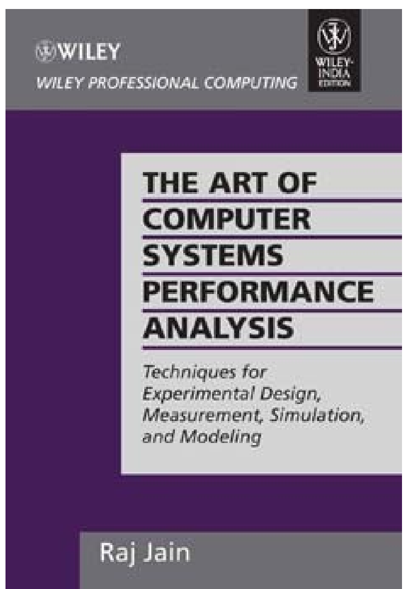
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING

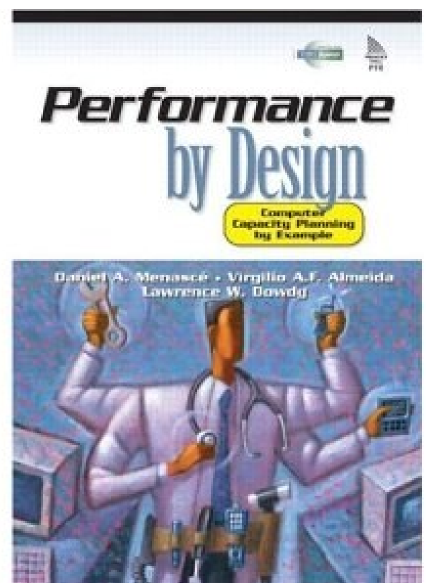
sosp

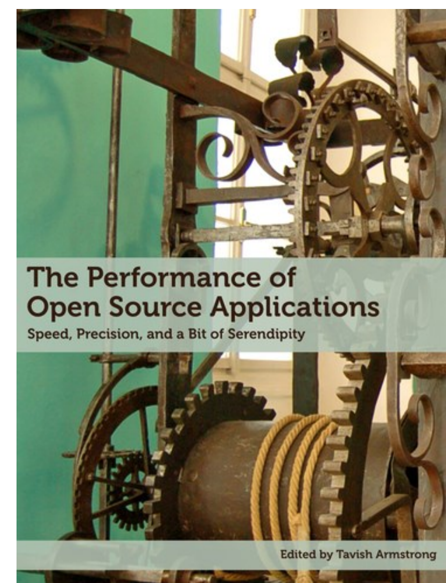
OSDI 14

FSE 2014









Roadmap

- Measurement
  - Workload Characterization
  - Performance Monitoring
  - Experimental Design
  - Performance Analysis and Visualization
- Simulation
- Analytical Modeling
  - Single Queue
  - Queuing Networks (QN)
  - Layered Queuing Networks (LQN)
  - PCM and Other Models
- Performance Anti-patterns

Performance Evaluation - Measurement

Measurement-based Performance Evaluation

Minimum # of experiments,
Maximum amount of information

testing, benchmarking,
capacity planning, etc.

Workload → **Experimental Design** → **Performance Measurement** → **Performance Analysis**

operational profile

light-weight performance monitoring and data recording

Operational Profiling (Workload Characterization)

An *operational profile*, also called a *workload*, is the expected workload of the system under test once it is operational in the field.

The process of extracting the expected workload is called *operational profiling* or *workload characterization*.

Workload Characterization Techniques

- Past data
  - Average/Minimum/Maximum request rates
  - Markov Chain
  - …
- Extrapolation
  - Alpha/Beta usage data
  - Interview from domain experts
  - …
- Workload characterization surveys
  - M. Calzarossa and G. Serazzi. Workload characterization: a survey. In Proceedings of the IEEE.1993.
  - S. Elnaffar and P. Martin. Characterizing Computer Systems' Workloads. Technical Report. School of Computing, Queen's University. 2002.

Workload Characterization Techniques - Markov Chain

```
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=ANTHONY%20
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=category&browse_category=11&browse_actor=&brow
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dslogin.jsp?username=user41&password=password HTTP/1.1" 200 2539 16
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=WILLIAM%20
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=category&browse_category=15&browse_actor=&brow
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=HILARY%20G
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=category&browse_category=6&browse_actor=&brows
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=title&browse_category=&browse_actor=&browse_ti
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=5961&item=646&quan=3&ite
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=41&item=4544&quan=1&item
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dslogin.jsp?username=user3614&password=password HTTP/1.1" 200 728 6
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dsbrowse.jsp?browsetype=title&browse_category=&browse_actor=&browse_ti
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=ELLEN%20GA
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dsbrowse.jsp?browsetype=category&browse_category=9&browse_actor=&brows
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=ANGELINA%2
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=JULIA%20TA
192.168.0.1 - [22/Apr/2014:00:32:29 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=3614&item=4717&quan=2&it
192.168.0.1 - [22/Apr/2014:00:32:31 -0400] "GET /dslogin.jsp?username=user13337&password=password HTTP/1.1" 200 1960 9
192.168.0.1 - [22/Apr/2014:00:32:31 -0400] "GET /dsbrowse.jsp?browsetype=title&browse_category=&browse_actor=&browse_ti
192.168.0.1 - [22/Apr/2014:00:32:31 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=13337&item=322&quan=2&it
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dslogin.jsp?username=user5414&password=password HTTP/1.1" 200 2579 10
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dsbrowse.jsp?browsetype=actor&browse_category=&browse_actor=GRACE%20BR
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=5414&item=198&quan=3&ite
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dsnewcustomer.jsp?firstname=RHVSQS&lastname=EBFMQDBVNM&address1=909823
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dsbrowse.jsp?browsetype=title&browse_category=&browse_actor=&browse_ti
192.168.0.1 - [22/Apr/2014:00:32:35 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=20001&item=7868&quan=3&i
192.168.0.1 - [22/Apr/2014:00:32:36 -0400] "GET /dslogin.jsp?username=user13713&password=password HTTP/1.1" 200 729 6
192.168.0.1 - [22/Apr/2014:00:32:36 -0400] "GET /dsbrowse.jsp?browsetype=category&browse_category=9&browse_actor=&brows
192.168.0.1 - [22/Apr/2014:00:32:36 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=13713&item=493&quan=3&it
192.168.0.1 - [22/Apr/2014:00:32:41 -0400] "GET /dslogin.jsp?username=user9011&password=password HTTP/1.1" 200 728 6
```

web access logs for the past few months

Workload Characterization Techniques - Markov Chain

192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dsbrowse.jsp?browsetype=title&browse_category=&browse_actor=&browse_title=HOLY%20AUTUMN&limit_num=8&customerid=41 HTTP/1.1" 200 4073 10

192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=5961&item=646&quan=3&item=2551&quan=1&item=45&quan=3&item=9700&quan=2&item=1566&quan=3&item=4509&quan=3&item=5940&quan=2 HTTP/1.1" 200 3049 177

192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /dspurchase.jsp?confirmpurchase=yes&customerid=41&item=4544&quan=1&item=6970&quan=3&item=5237&quan=2&item=650&quan=1&item=2449&quan=1 HTTP/1.1" 200 2515 113

Web Access Logs

Workload Characterization Techniques - Markov Chain

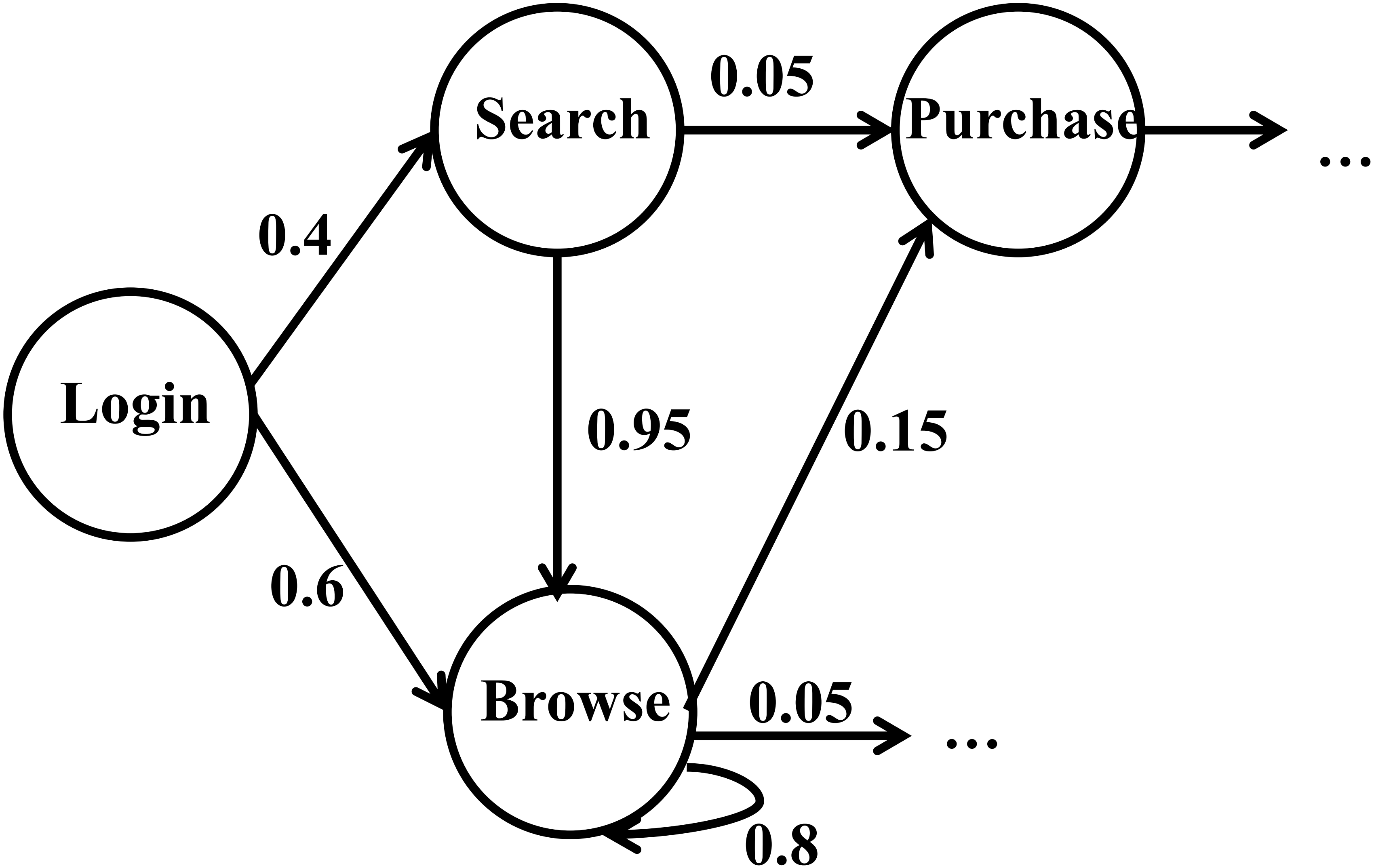
192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /**dsbrowse.jsp**?browsetype=title&browse_category=&browse_actor=&browse_title=HOLY%20AUTUMN&limit_num=8&**customerid=41** HTTP/1.1" 200 4073 10

192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /**dspurchase.jsp**?confirmpurchase=yes&**customerid=5961**&item=646&quan=3&item=2551&quan=1&item=45&quan=3&item=9700&quan=2&item=1566&quan=3&item=4509&quan=3&item=5940&quan=2 HTTP/1.1" 200 3049 177

192.168.0.1 - [22/Apr/2014:00:32:25 -0400] "GET /**dspurchase.jsp**?confirmpurchase=yes&**customerid=41**&item=4544&quan=1&item=6970&quan=3&item=5237&quan=2&item=650&quan=1&item=2449&quan=1 HTTP/1.1" 200 2515 113

For customer 41: browse -> purchase

Workload Characterization Techniques - Markov Chain



Experimental Design

- Suppose a system has 5 user configuration parameters. Three out of five parameters have 2 possible values and the other two parameters have 3 possible values. Hence, there are $2^3 \times 3^2 = 72$ possible configurations to test.
- Apache webserver has 172 user configuration parameters (158 binary options). This system has 1.8×10^{55} possible configurations to test!

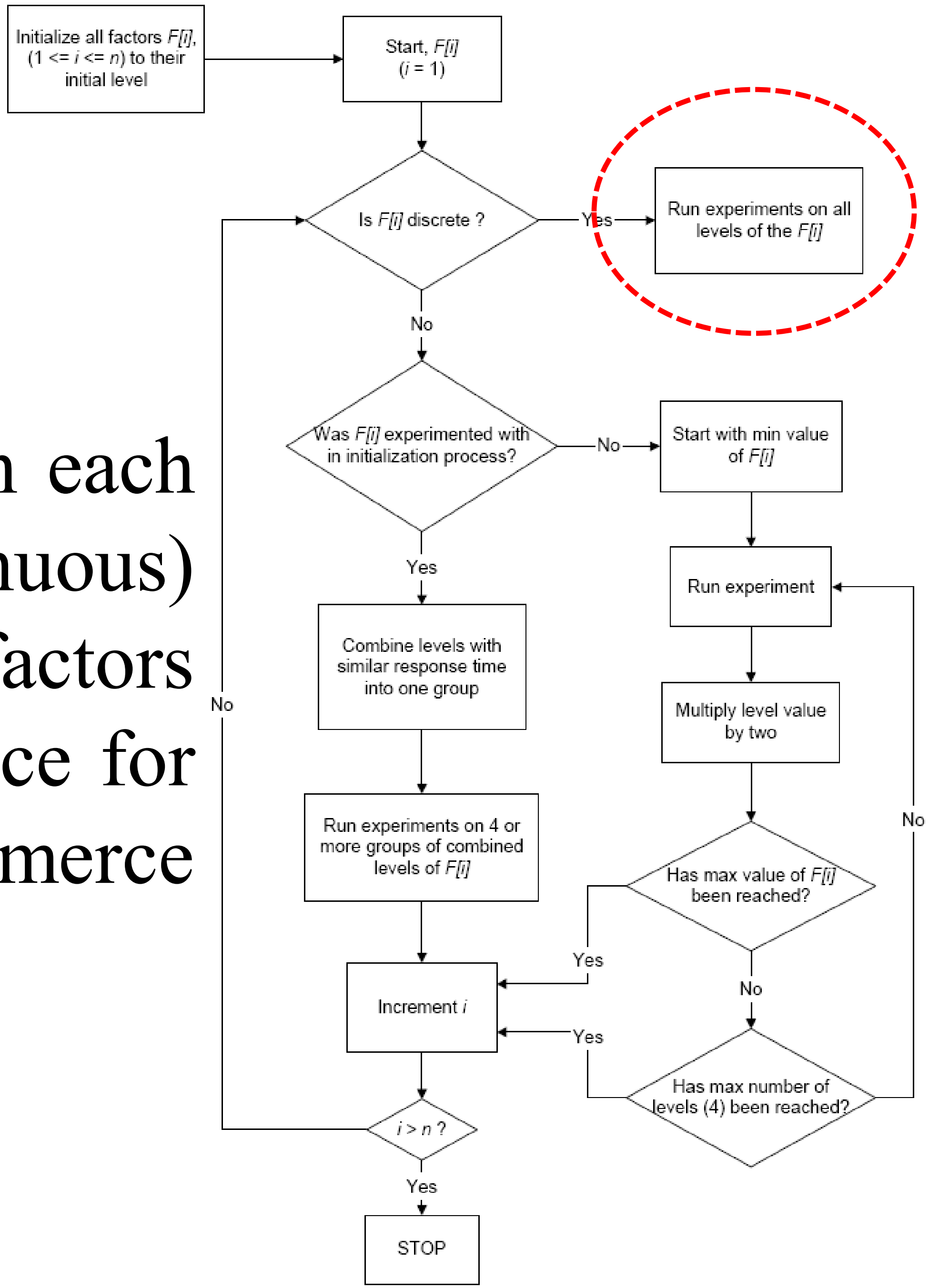
The goal of a proper *experimental design* is to obtain the maximum information with the minimum number of experiments.

Experimental Design Terminologies

- The outcome of an experiment is called the ***response variable***.
  - *E.g.*, throughput and response time for the tasks.
- Each variable that affects the response variable and has several alternatives is called a ***factor***.
  - *E.g.*, to measure the performance of a workstation, there are five factors: CPU type, memory size, number of disk drives and workload.
- The values that a factor can have are called ***levels***.
  - E.g., Memory size has 3 levels: 2 GB, 6 GB and 12 GB
- Repetition of all or some experiments is called ***replication***.
- ***Interaction effects***: Two factors A and B are said to interact if the effect of one depends on the other.

Ad-hoc Approach

Iteratively going through each (discrete and continuous) factors and identity factors which impact performance for an three-tired e-commerce system.



[Sopitkamol et al., WOSP 2005]

Covering Array

- A *t-way covering array* for a given input space model is a set of configurations in which each valid combination of factor-values for every combination of t factors appears at least once.
- Suppose a system has 5 user configuration parameters. Three out of five parameters have 2 possible values (0, 1) and the other two parameters have 3 possible values (0, 1, 2). There are total $2^3 \times 3^2 = 72$ possible configurations to test.

A 2-way covering array

A	B	C	D	E
0	1	1	2	0
0	0	0	0	0
0	0	0	1	1
1	1	1	0	1
0	1	0	0	2
1	0	1	1	0
1	1	1	1	2
1	0	0	2	1
1	0	0	2	2

Covering Array and CIT

- There are many other kinds of covering array like: variable-strength covering array, test case-aware covering array, etc.
- *Combinatorial Interaction Testing* (CIT) models a system under test as a set of factors, each of which takes its values from a particular domain. CIT generates a sample that meets the specific coverage criteria (e.g., 3-way coverage).
- Many commercial and free tools: http://pairwise.org/tools.asp

[Yilmaz et al., IEEE Computer 2014]

Performance Measurement

- Types of performance data
- Performance Monitoring
  - Agent-less Monitoring
  - Agent-based Monitoring
- Measurement-based frameworks
- Performance measurement issues

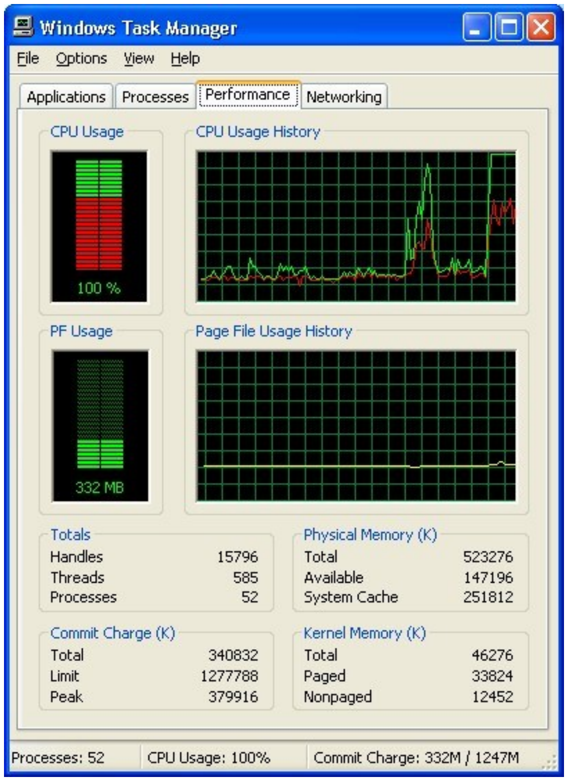
Performance Data

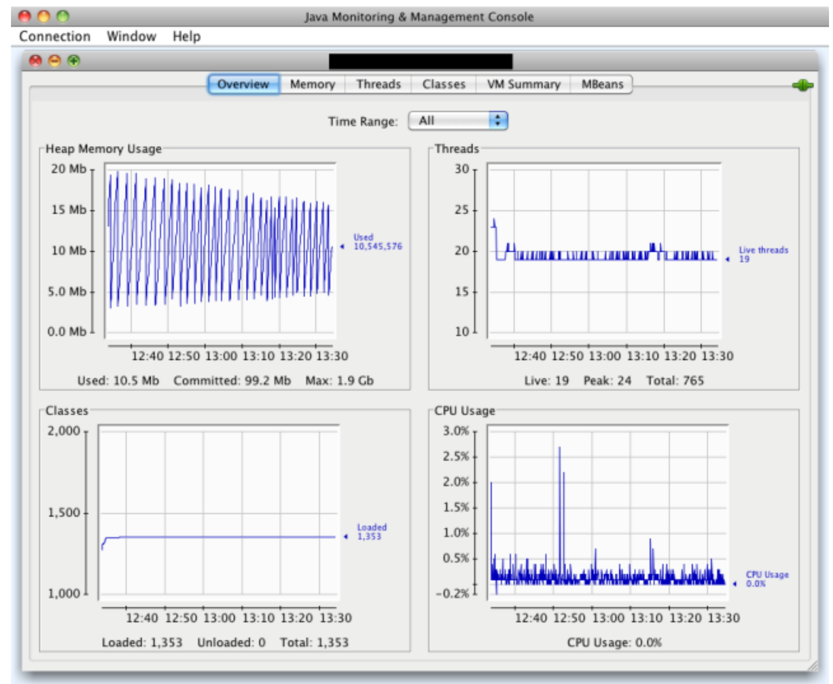
- System and application resource usage metrics
  - CPU, memory, network, etc.
- Application performance metrics
  - Response time, throughput, # of requests submitted and # of requests completed, etc.
- Application specific metrics
  - # of concurrent connections to the database, rate of database transactions, # of garbage collections
- Some of these data can be obtained directly, while others need to be derived (e.g., from logs).

Performance Monitor

- Monitors and records the system behavior over time
- Needs to be light-weight
  - Imposing as little performance overhead as possible
- Two types of performance monitoring approaches
  - Agent-less monitoring
  - Agent-based monitoring

Agent-less Monitoring Examples

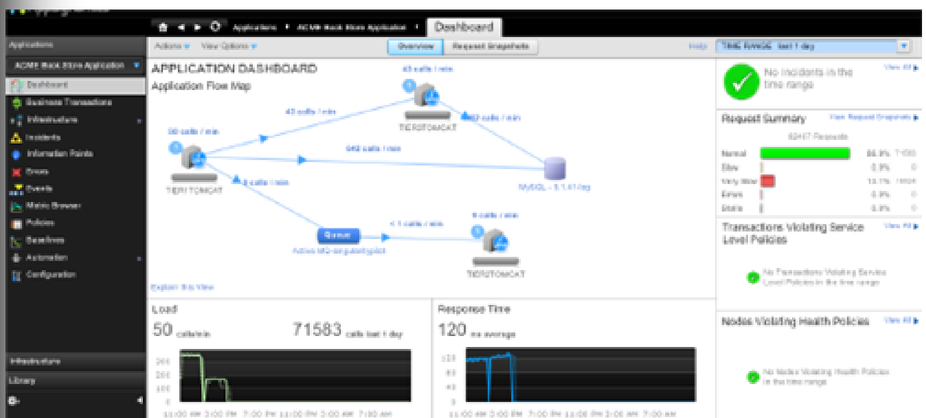


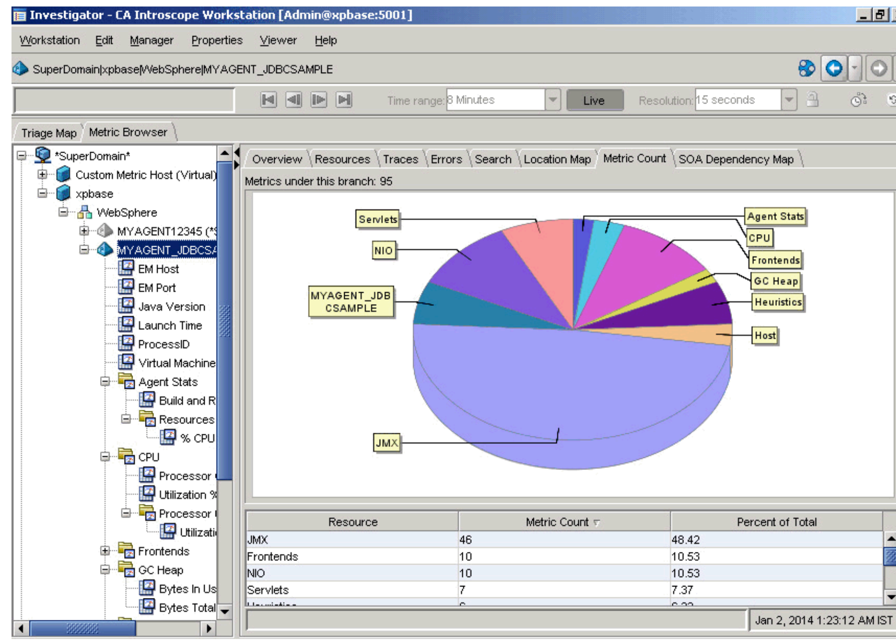


Task Manager **JConsole**

PerfMon (Windows), sysstat (Linux), top

Agent-based Monitoring Examples

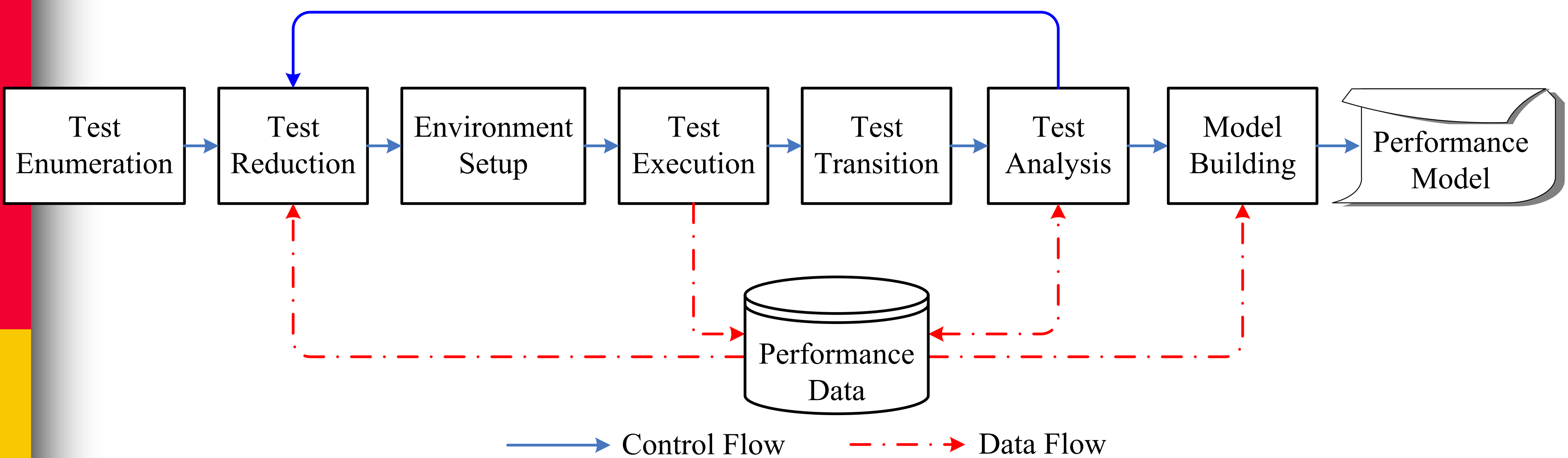




App Dynamics **CA Willy**

Dell FogLight, New Relic

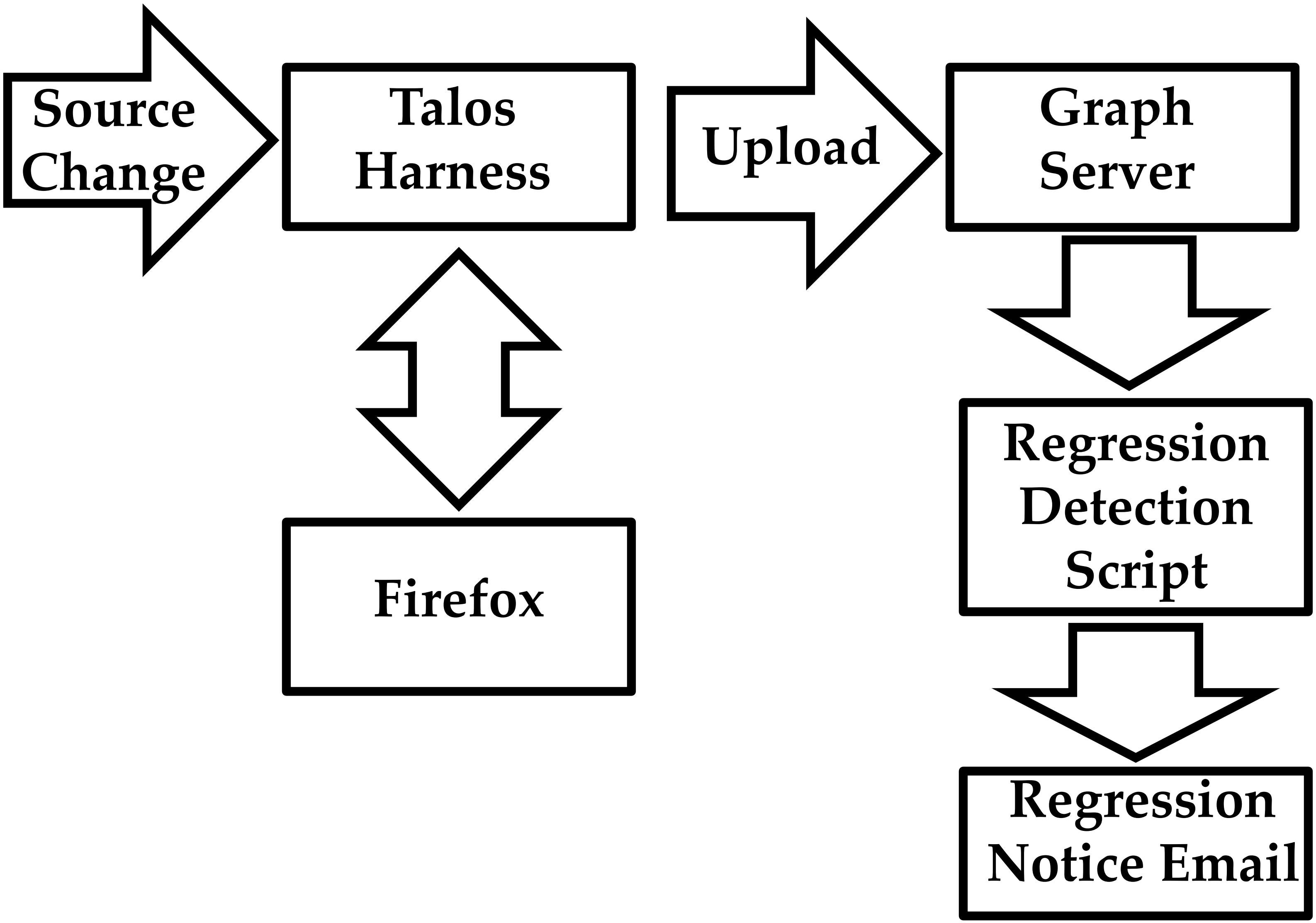
A Framework for Measurement-based Performance Modeling



Capacity Planning

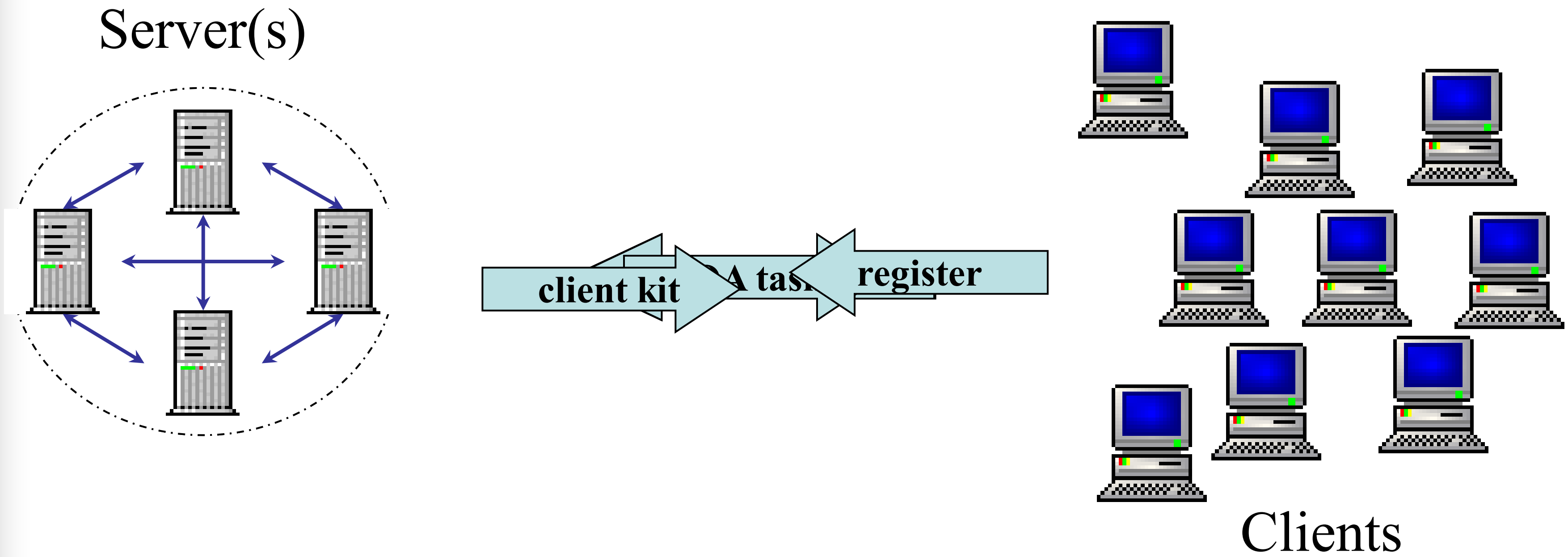
[Thakkar et al., WOSP 2008]

Talos
- Mozilla Performance Regression Testing Framework



[Talbert et al., http://aosabook.org/en/posa/talos.html]

Skoll – A Distributed Continuous Quality Assurance (DCQA) Infrastructure



- Server selects best task that matches client characteristics

Performance Regression Testing under different configurations

[Memon et al., ICSE 2004]

Measurement Bias

- Measurement bias is hard to avoid and unpredictable.
- **<u>Example 1</u>**: How come the same application today runs faster compared with yesterday?
- **<u>Example 2</u>**: Why the response time is very different when running the same binary under different user accounts?
- **<u>Example 3</u>**: Why the code optimization only works on my computer?

  - **Repeated measurement**
  - **Randomize experiment setup**

[Mytkowicz et al., ASPLOS 2010]

Evaluate 2010

Workshop on Experimental Evaluation of Software and Systems in Computer Science

Co-located with SPLASH'10 in Reno/Tahoe, Nevada, October 17-21, 2010.

News

- See you on Monday, October 18, in Reno!
- Now open: Evaluate Collaboratory (a collaborative web site for workshop participants and the broader community)
  - Check out, update, and comment on the position statements (account required)
  - Browse or contribute to the Experimental Evaluation Bibliography
  - Check out the Evaluate 2010 schedule, list of participants, location, lunch and dinner options, and more
- Keynote: Cliff Click
- Panel: Experimental evaluation in different areas of computer science. The panelists are leaders in evaluation methodologies in their respective

Call for Position Statements

We call ourselves 'computer scientists', but are we scientists? If we are scientists, then we must practice the scientific method. This includes a solid ex

In the last few years, researchers have identified disturbing flaws in the way that experiments are performed in computer science. For example, in the
As hardware and software grow more complex, this problem just gets worse.

This workshop brings together experts from different areas of computer science to discuss, explore, and attempt to identify the principles of sound ex

The workshop will consist of discussion sessions, which focus on themes such as data collection, data analysis, and reproducibility, with the goal to a

- What are the issues that are preventing proper experimental evaluation?
- How can we resolve these issues?
- We need more research in evaluation methodology. What should that research be?
- We need better tools to do sound experimental evaluation. How do we encourage investment in such tools?
- What are the principles and best practices that people are using in the different areas of computer science?
- How does the computer science curriculum need to be changed to prepare the next generation of computer scientists?

Performance Analysis

- Statistical analysis
  - Descriptive statistics
  - Hypothesis testing
  - Regression analysis
- Performance visualization
- Performance debugging
  - Profiling
  - Instrumentation

Comparing Two Alternatives

- Paired observations
  - E.g., there are six scenarios ran on two versions of the system. The response time are: {(5.4, 19.1), (16.6, 3.5), (0.6, 3.4), (1.4, 2.5), (0.6, 3.6), (7.3, 1.7)}
  - Paired student-t test
- Unpaired observations
  - E.g., the browsing scenarios were ran 10 times on Release A and 11 times on Release B
  - Unpaired student-t test
- Student-t tests assume that the two datasets are normally distributed. Otherwise, we need to use non-parametric tests
  - Wilcoxon signed-rank test for paired observations
  - Mann-Whitney U test for the unpaired observations

Comparing More than Two Alternatives

- For comparing more than two alternatives, we will use ANOVA (Analysis of variance)
  - E.g., There are 6 different measurements under 6 different software configurations.
- ANOVA also assumes the datasets are normally distributed. Otherwise, we need to use non-parametric tests (e.g., Kruskal-Wallis H test)

Statistical significance v.s. Performance impact

- The new design's performance may be statistically faster than the old version. However, it's only 0.001 seconds faster and will take a long time to implement. Is it worth the effort?

$$\text{Effect Sizes} = \begin{cases} \text{Trivial} & (\text{Cohen's D} \leq 0.2) \\ \text{Small} & (0.2 < \text{Cohen's D} \leq 0.5) \\ \text{Medium} & \text{if } 0.5 < \text{Cohen's D} \leq 0.8 \\ \text{Large} & 0.8 < \text{Cohen's D} \end{cases}$$

Cliff's δ -> Non-parametric alternative

Regression-based Analysis

- Can we find an empirical model which predicts the application CPU utilization as a function of the workload? For example,

$$CPU = b_0 + b_1 \times \#\ of\ browsing + b_2 \times \#\ of\ searching$$

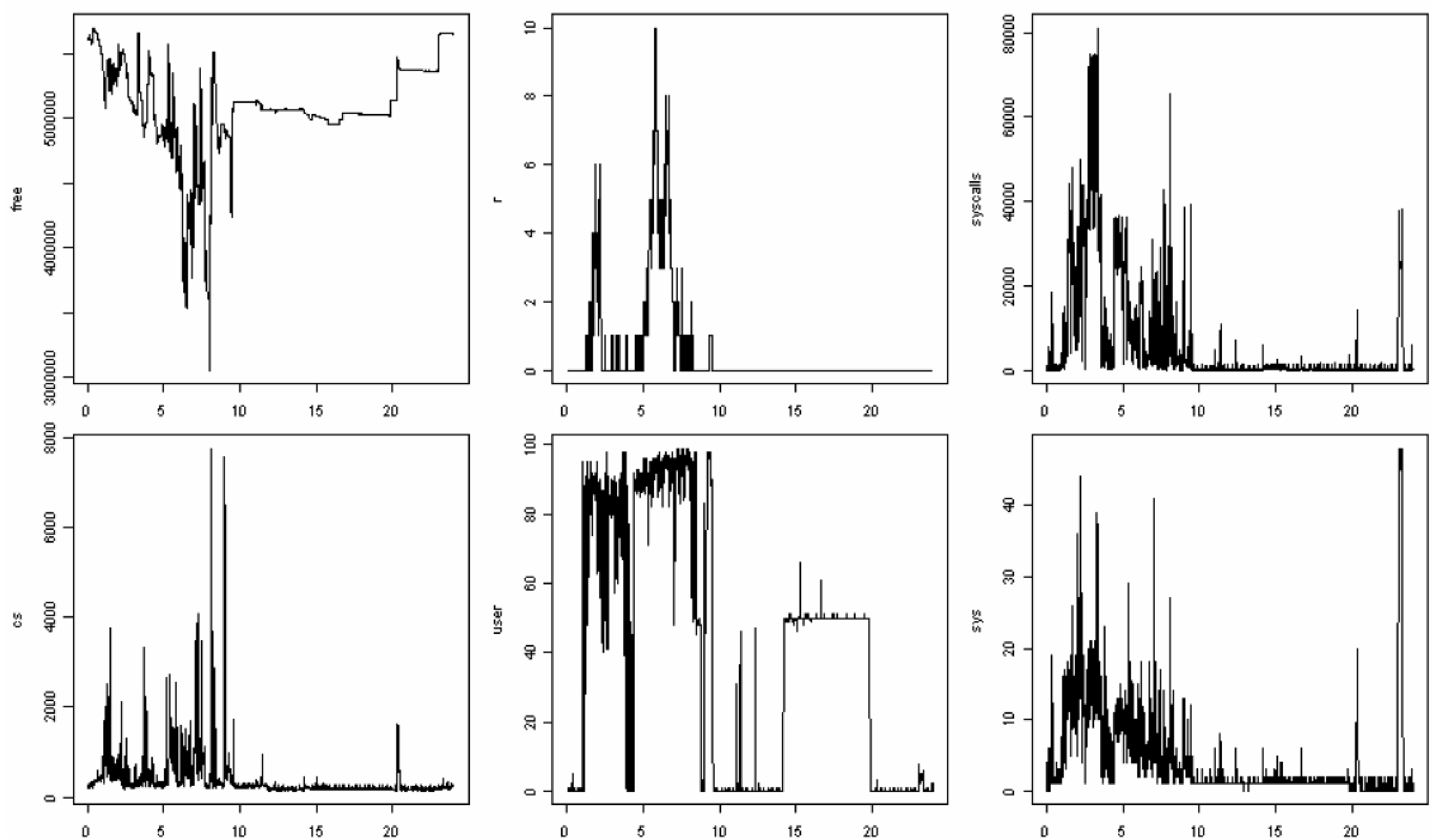
- Then we can conduct various what-if analysis?
  - (Field Assessment) What if the field workload is A, would the existing setup be able to handle that load?
  - (Capacity Planning) What kind of machine (CPU power) should we pick based on the project workload for next 3 years?

Are input variables linearly independent of each other?

If they are highly correlated, keep only one of them in the model

Performance Visualization

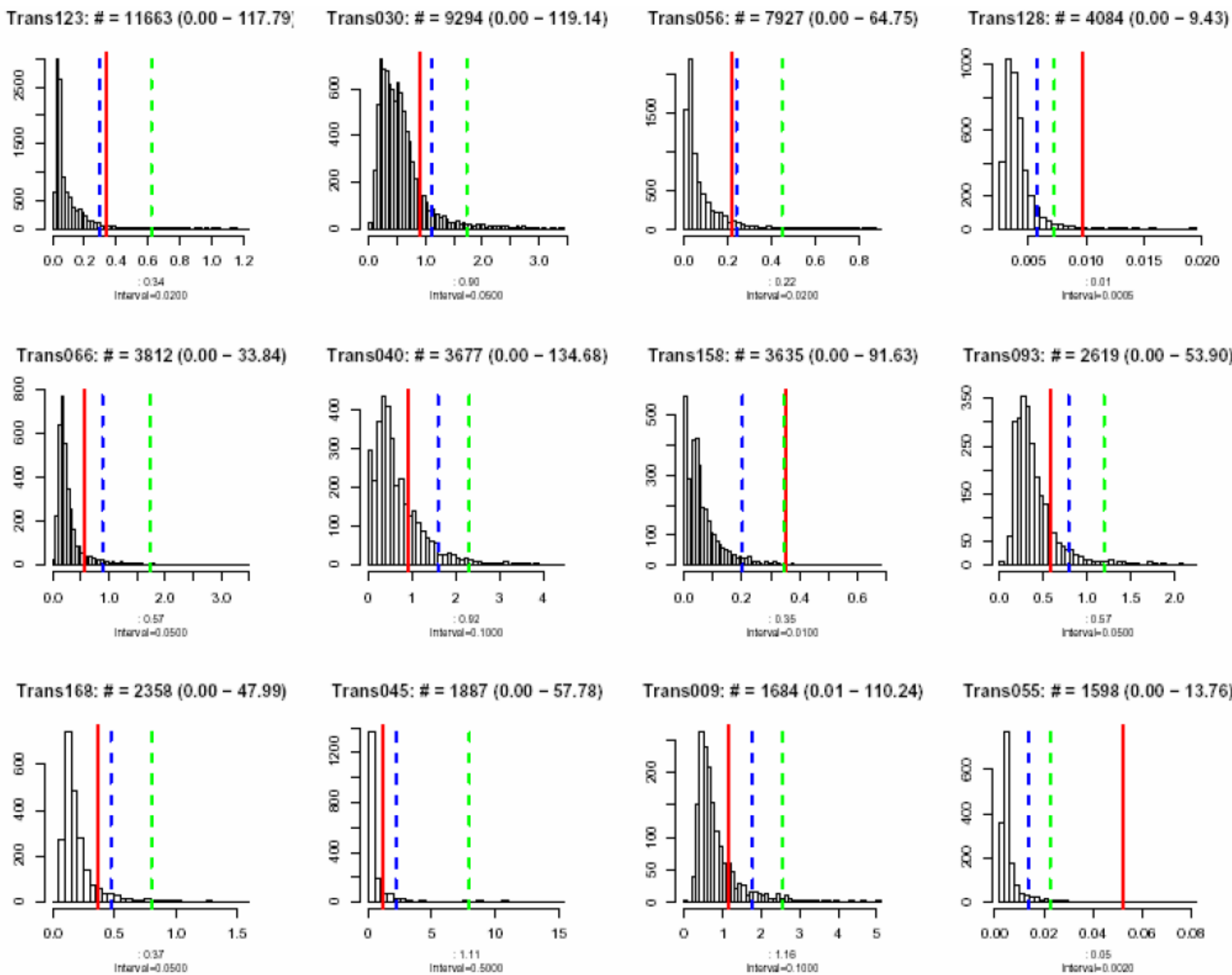
Line Plots



Metrics plots from vmstats

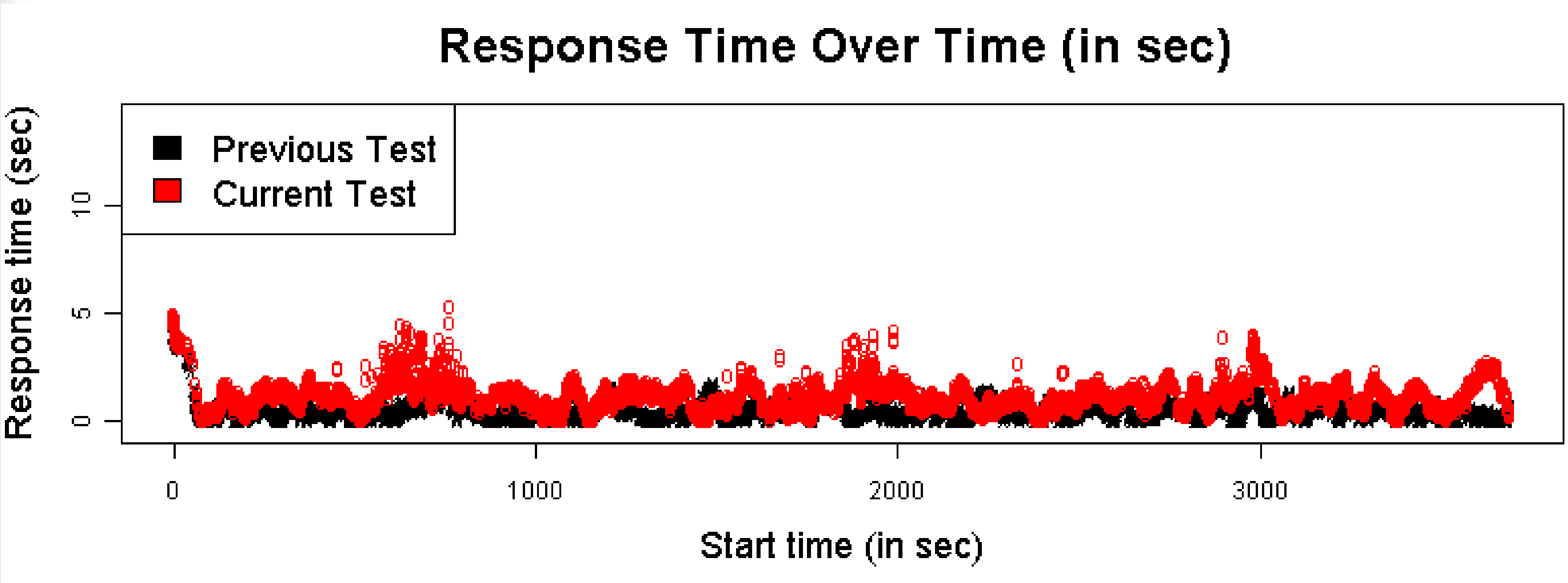
[Holtman, 2006]

Histogram Plots



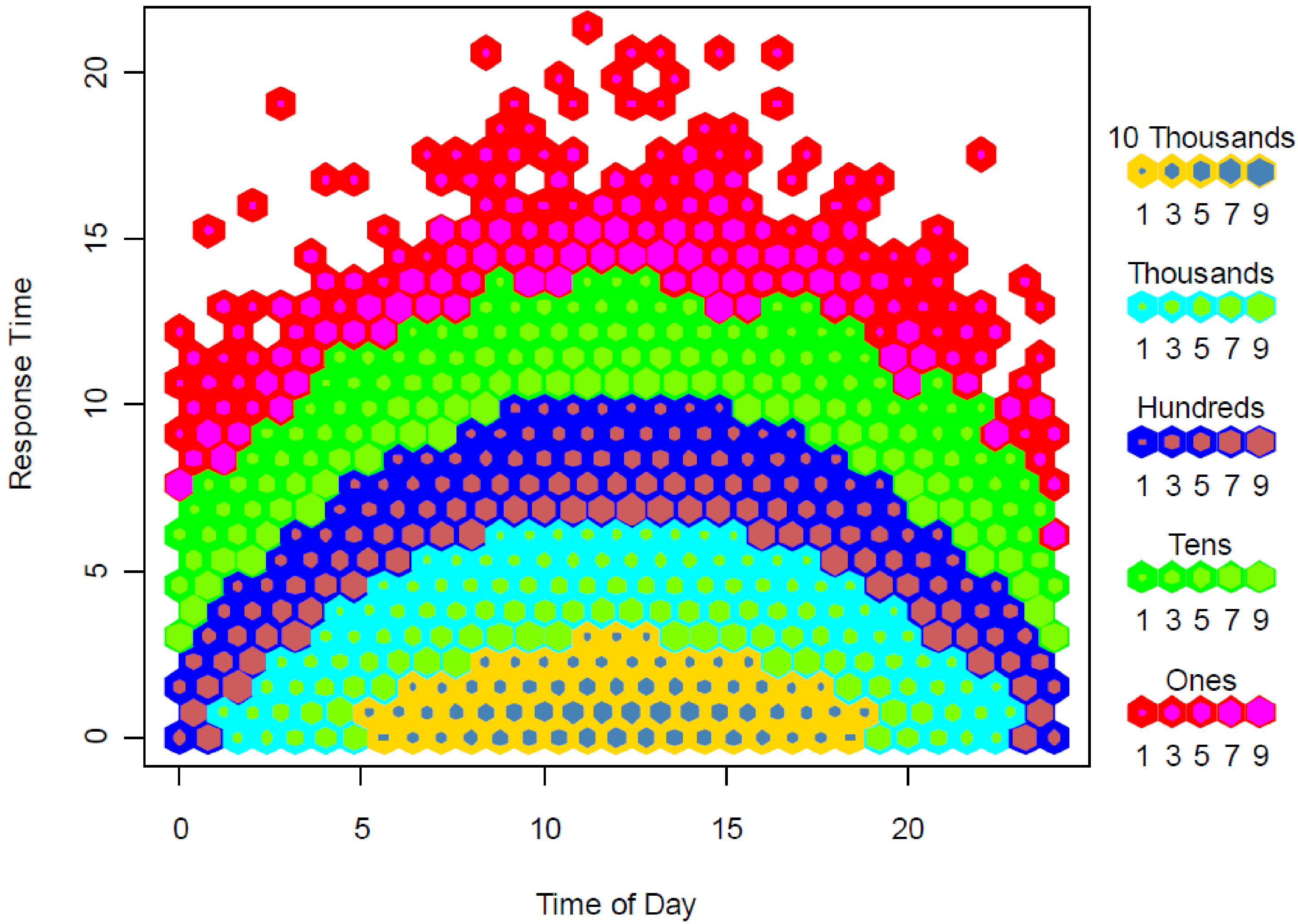
[Holtman, 2006]

Scatter Plot



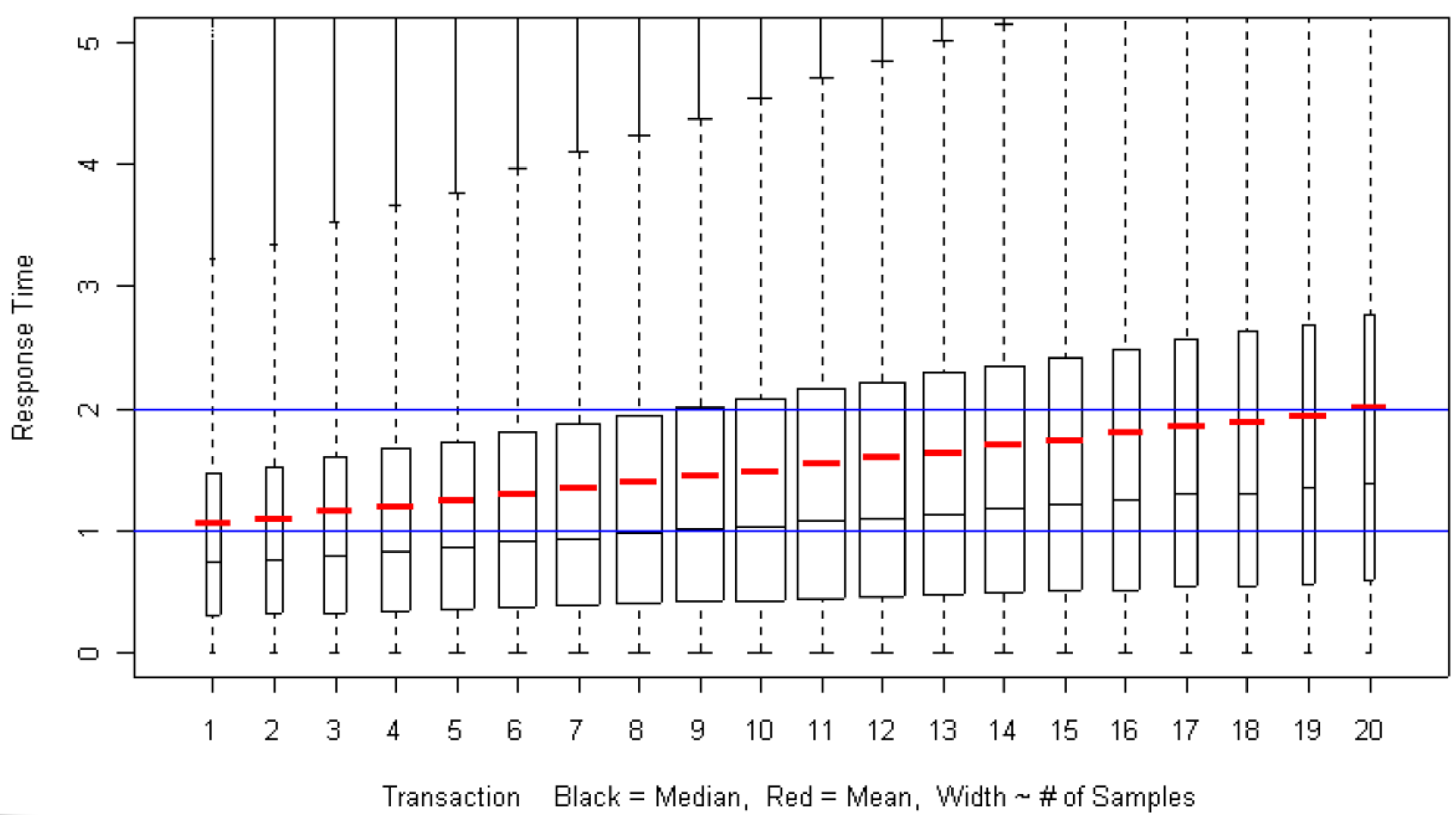
[Jiang et al., ICSM 2009]

Hexbin Plot



[Holtman, 2006]

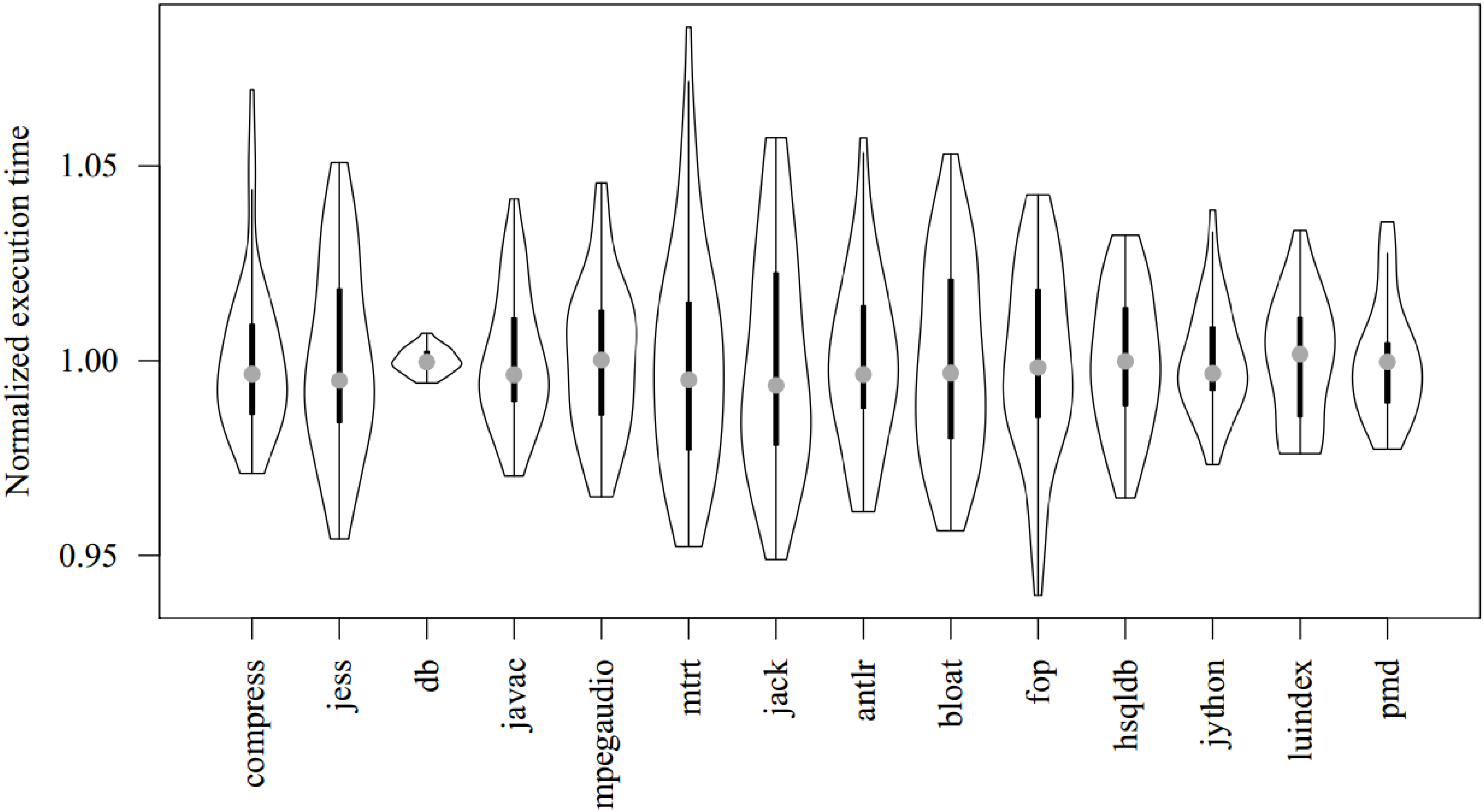
Box-Plot



Width of the box is proportional to the square root of the number of samples for that transaction type

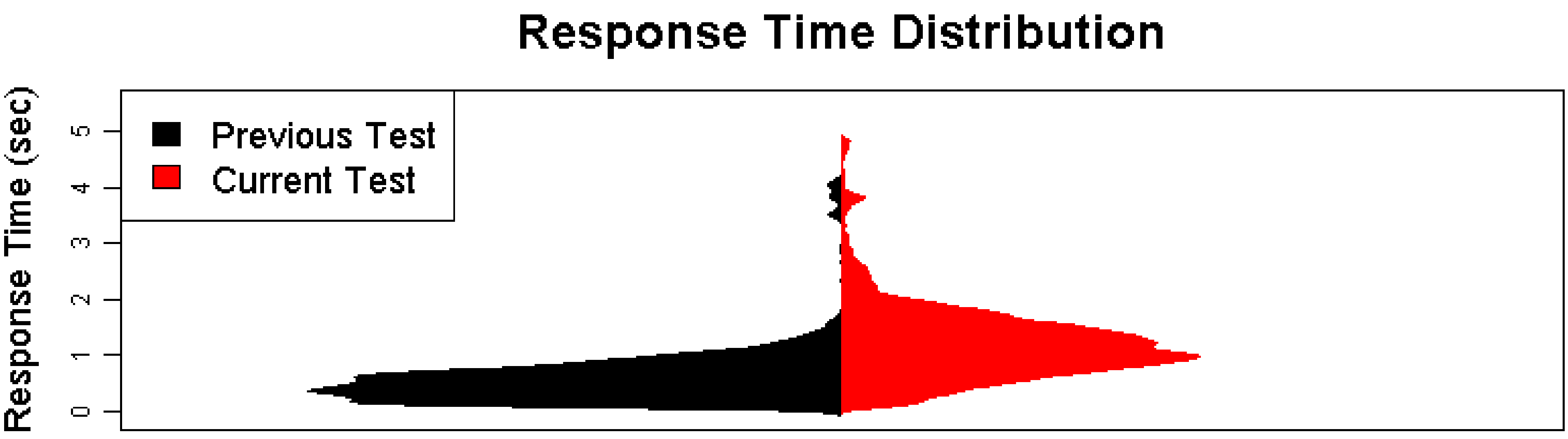
[Holtman, 2006]

Violin Plot



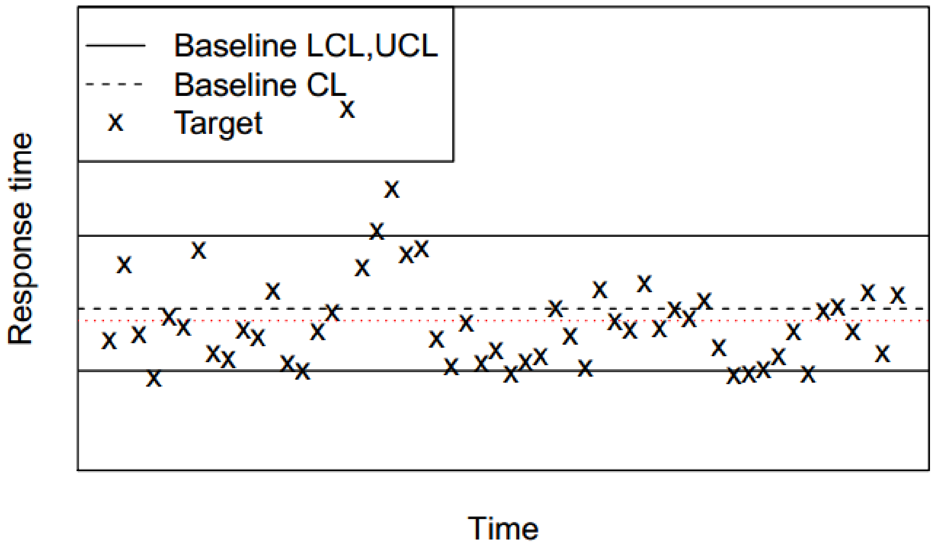
[Georges et al., OOPSLA 2007]

Bean Plot

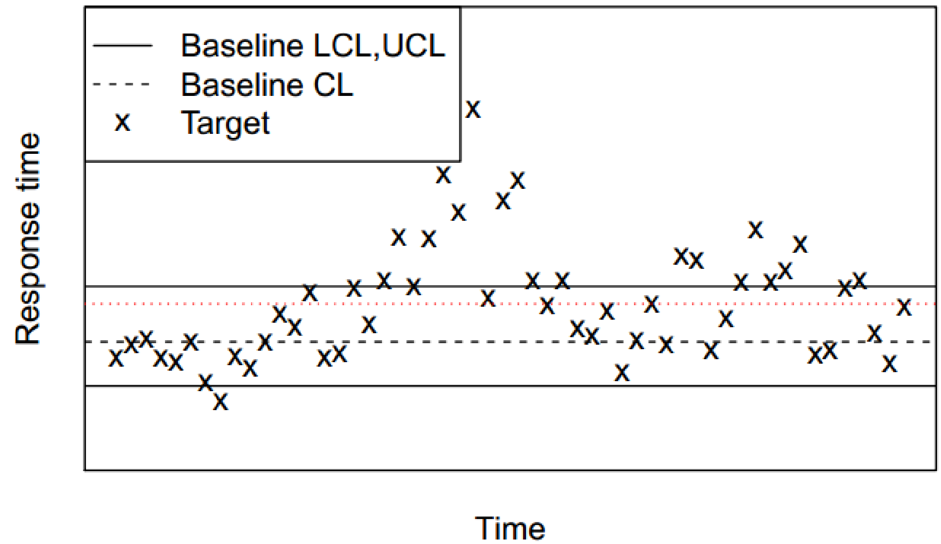


[Jiang et al., ICSM 2009]

Control Charts



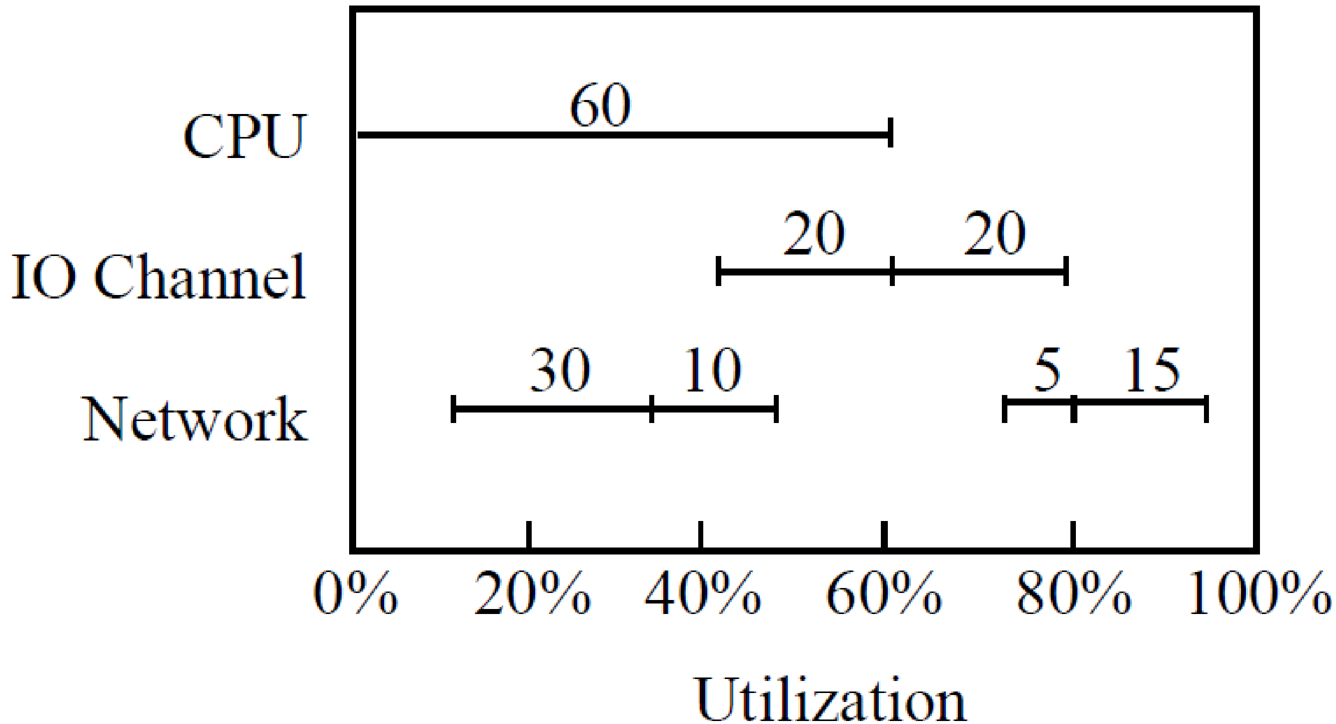
(a) Normal operation



(b) Out-of-control

[Nguyen et al., APSEC 2011]

Gantt Chart



Shows the relative duration of a number of (Boolean) conditions

[Jain, 1991]

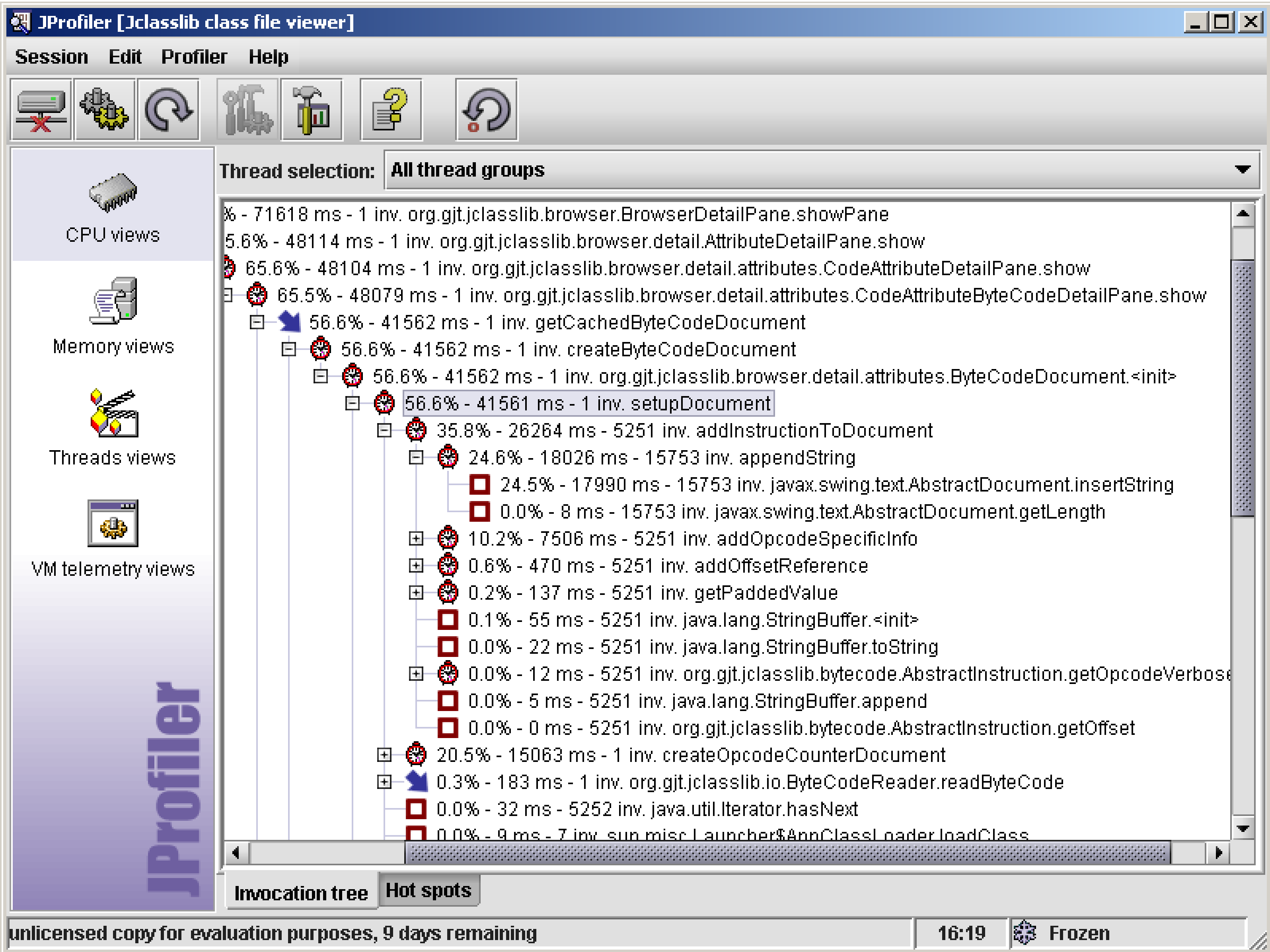
Instrumentation

- Source code level instrumentation
  - Ad-hoc manual instrumentation,
  - Automated instrumentation (e.g., AspectJ), and
  - Performance instrumentation framework (e.g., the Application Response Time API)
- Binary instrumentation framework
  - DynInst (http://www.dyninst.org/),
  - PIN (http://www.dyninst.org/)
  - Valgrind (http://valgrind.org/)
- Java Bytecode instrumentation framework
  - Ernst's ASE 05 tutorial on "Learning from executions: Dynamic analysis for software engineering and program understanding" (http://pag.csail.mit.edu/~mernst/pubs/dynamic-tutorial-ase2005-abstract.html)

Profiling

- Profilers can help developers locate “hot” methods
  - Methods which consume the most amount of resources (e.g., CPU)
  - Methods which take the longest
  - Methods which are called frequently
- Examples of profilers:
  - Windows events: xperf
  - Java applications: xprof, hprof, JProfiler, YourKit
  - .net applications: YourKit
  - Dump analysis: DebugDiag (Windows dumps), Eclipse Memory Analyzer (Java heap dumps)
  - Linux events: SystemTap/Dtrace, lttng

JProfiler


JProfiler [Jclasslib class file viewer]
Session Edit Profiler Help
Thread selection: All thread groups
CPU views
Memory views
Threads views
VM telemetry views
JProfiler
% - 71618 ms - 1 inv. org.gjt.jclasslib.browser.BrowserDetailPane.showPane
5.6% - 48114 ms - 1 inv. org.gjt.jclasslib.browser.detail.AttributeDetailPane.show
65.6% - 48104 ms - 1 inv. org.gjt.jclasslib.browser.detail.attributes.CodeAttributeDetailPane.show
65.5% - 48079 ms - 1 inv. org.gjt.jclasslib.browser.detail.attributes.CodeAttributeByteCodeDetailPane.show
56.6% - 41562 ms - 1 inv. getCachedByteCodeDocument
56.6% - 41562 ms - 1 inv. createByteCodeDocument
56.6% - 41562 ms - 1 inv. org.gjt.jclasslib.browser.detail.attributes.ByteCodeDocument.<init>
56.6% - 41561 ms - 1 inv. setupDocument
35.8% - 26264 ms - 5251 inv. addInstructionToDocument
24.6% - 18026 ms - 15753 inv. appendString
24.5% - 17990 ms - 15753 inv. javax.swing.text.AbstractDocument.insertString
0.0% - 8 ms - 15753 inv. javax.swing.text.AbstractDocument.getLength
10.2% - 7506 ms - 5251 inv. addOpcodeSpecificInfo
0.6% - 470 ms - 5251 inv. addOffsetReference
0.2% - 137 ms - 5251 inv. getPaddedValue
0.1% - 55 ms - 5251 inv. java.lang.StringBuffer.<init>
0.0% - 22 ms - 5251 inv. java.lang.StringBuffer.toString
0.0% - 12 ms - 5251 inv. org.gjt.jclasslib.bytecode.AbstractInstruction.getOpcodeVerbose
0.0% - 5 ms - 5251 inv. java.lang.StringBuffer.append
0.0% - 0 ms - 5251 inv. org.gjt.jclasslib.bytecode.AbstractInstruction.getOffset
20.5% - 15063 ms - 1 inv. createOpcodeCounterDocument
0.3% - 183 ms - 1 inv. org.gjt.jclasslib.io.ByteCodeReader.readByteCode
0.0% - 32 ms - 5252 inv. java.util.Iterator.hasNext
Invocation tree
Hot spots
unlicensed copy for evaluation purposes, 9 days remaining
16:19
Frozen

How JProfiler works

- Monitors the JVM activities via the JVM Tool Interface (JVMTI) to trap one or more of the following event types:
  - Lifecycle of classes,
  - Lifecycle of threads,
  - Lifecycle of objects,
  - Garbage collection events, etc.
- System overhead v.s. the details of system behavior
  - The more the types of monitored events, the higher the total number of events collected by the profiler, the slower the system is (higher overhead)
  - To reduce overhead:
    - the profiler is recommended to run under sampling mode, and
    - select only the "needed" types of events to monitor

http://resources.ej-technologies.com/jprofiler/help/doc/index.html

Evaluating the Accuracy of Java Profilers

- This paper shows that four commonly-used Java profilers (*xprof*, *hprof*, *jprofile*, and *yourkit*) often disagree on the identity of the hot methods.
- The results of profilers disagree because
  - They are run under the "sampling" mode
  - The samples are not randomly selected
    - They are all "yield point-based" profilers
    - The "observer effect" of profilers => Using a different profiler can lead to differences in the compiled code (dynamic optimizations by the JVM) and subsequently differently placed yield points

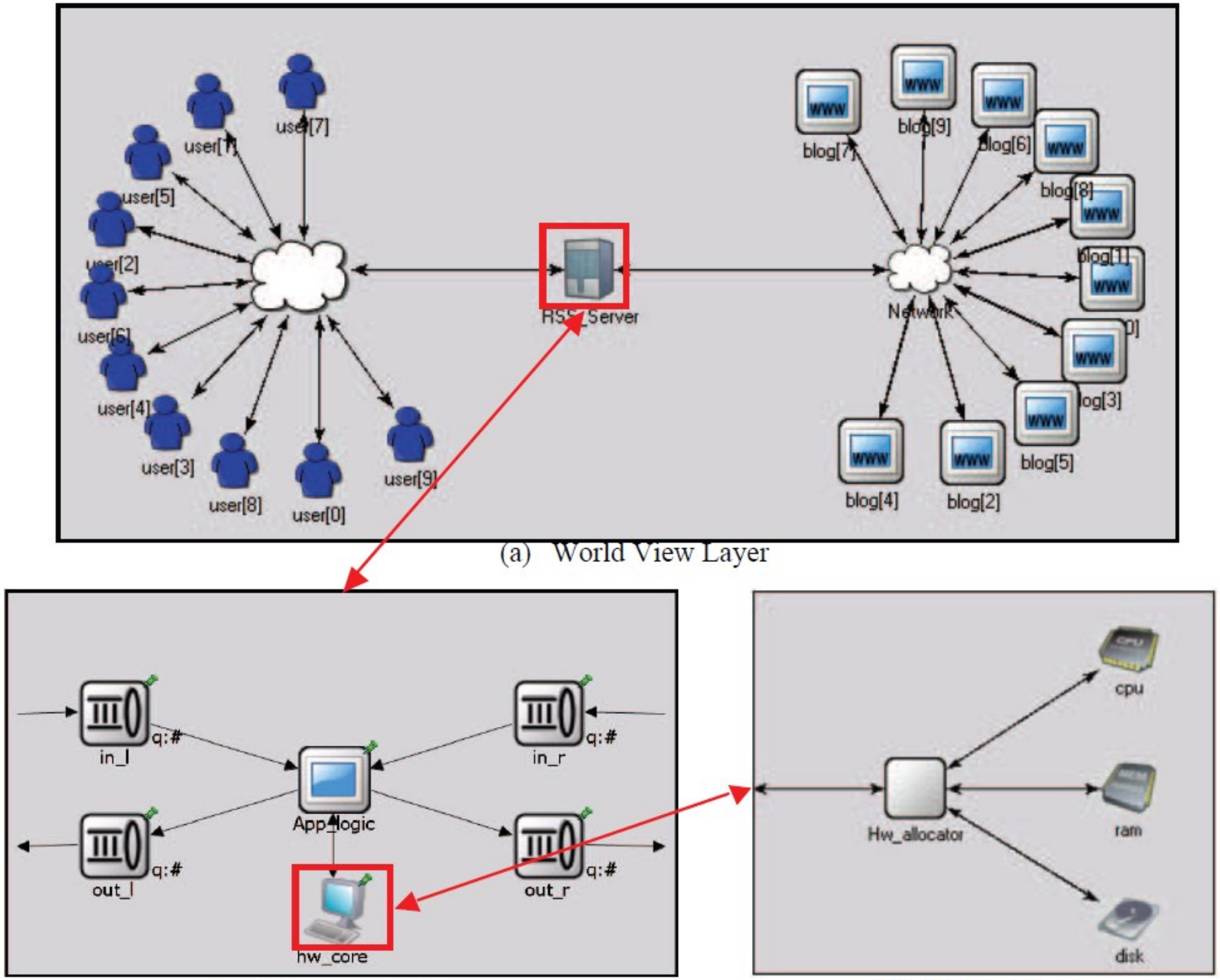
[Mytkowicz et al., PLDI 2010]

Performance Evaluation
- Simulation

Simulation

- A simulation model can be used
  - when during the design stage or even some components are not available; or
  - when it is much cheaper and faster than measurement-based approach (Simulating an 8-hour experiment is much faster than running the experiment for 8 hours.)
- However, the simulation models
  - usually take longer to develop than the analytical models, and
  - are not as convincing to practitioners as the measurement-based models

Evaluating the Performance Impact of Software Design Changes



(a) World View Layer

(b) Component Layer

(c) Physical Layer

Developed using the OMNetT++ framework

[Foo et al., MESOCA 2011]

Simulation

- Used extensively in computer networking. It is also gaining popularity in SPE, especially when it is used to solve performance models.

- Popular simulation frameworks
  - NS2 network simulator: http://isi.edu/nsnam/ns/
  - OMNeT++ network simulation framework: http://www.omnetpp.org/
  - OPNet: http://www.riverbed.com/products/performance-management-control/opnet.html

OMNeT++

OPNET®
Making Networks and Applications Perform™

Performance Evaluation
- Analytical Modeling

Performance Models

- ***Performance models*** describe how system operations use resources and how resource content affects operations.

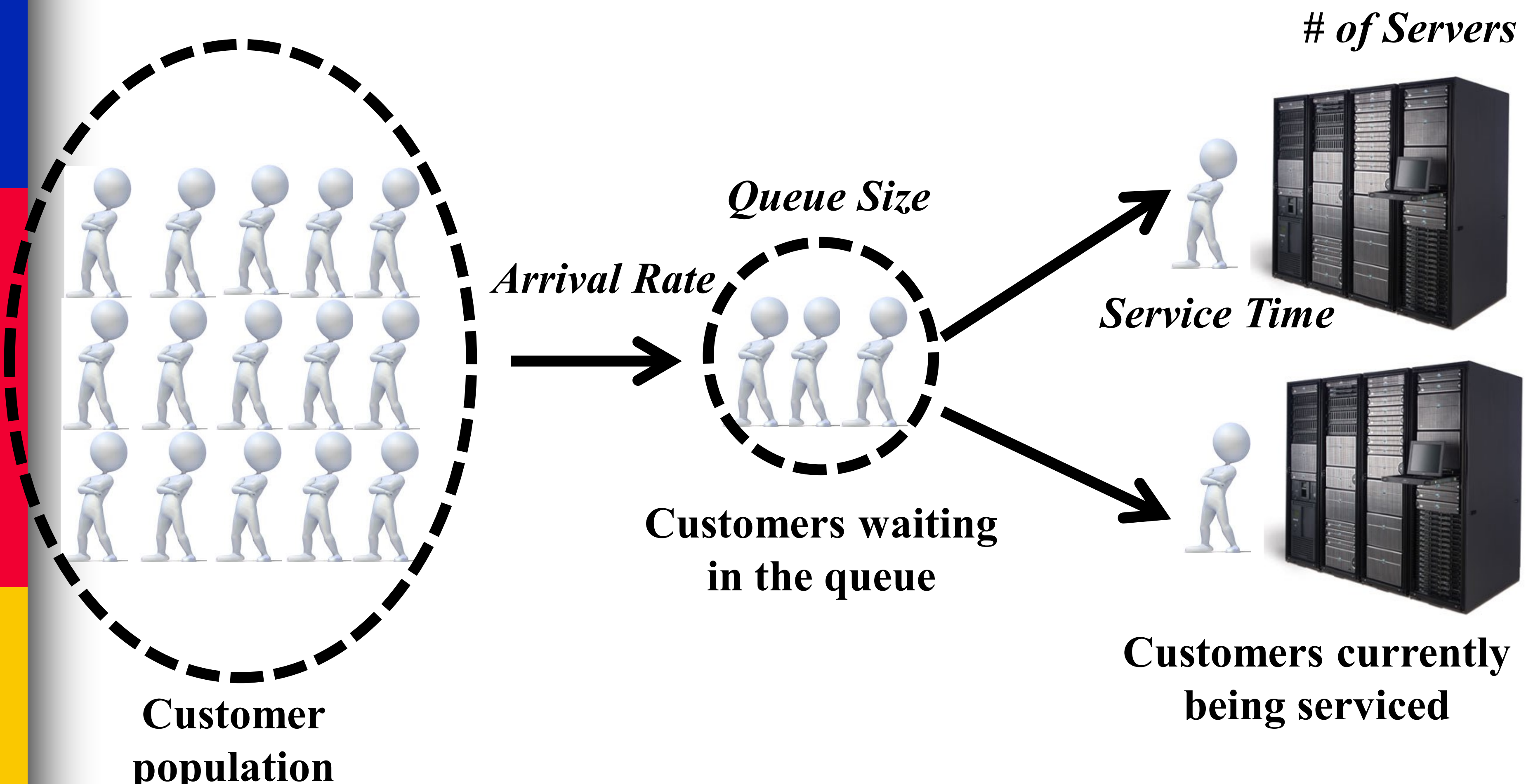
- ***Early-cycle performance models*** can predict a system's performance before it's build, or assess the effect of a change before it's carried out.

  During the requirements or design stages

- ***Late-cycle performance models*** explore amongst various architecture and configuration alternatives to support the evolution of these large software systems.

  Uses data from the measurement-based approach

Basic Components of a Queue



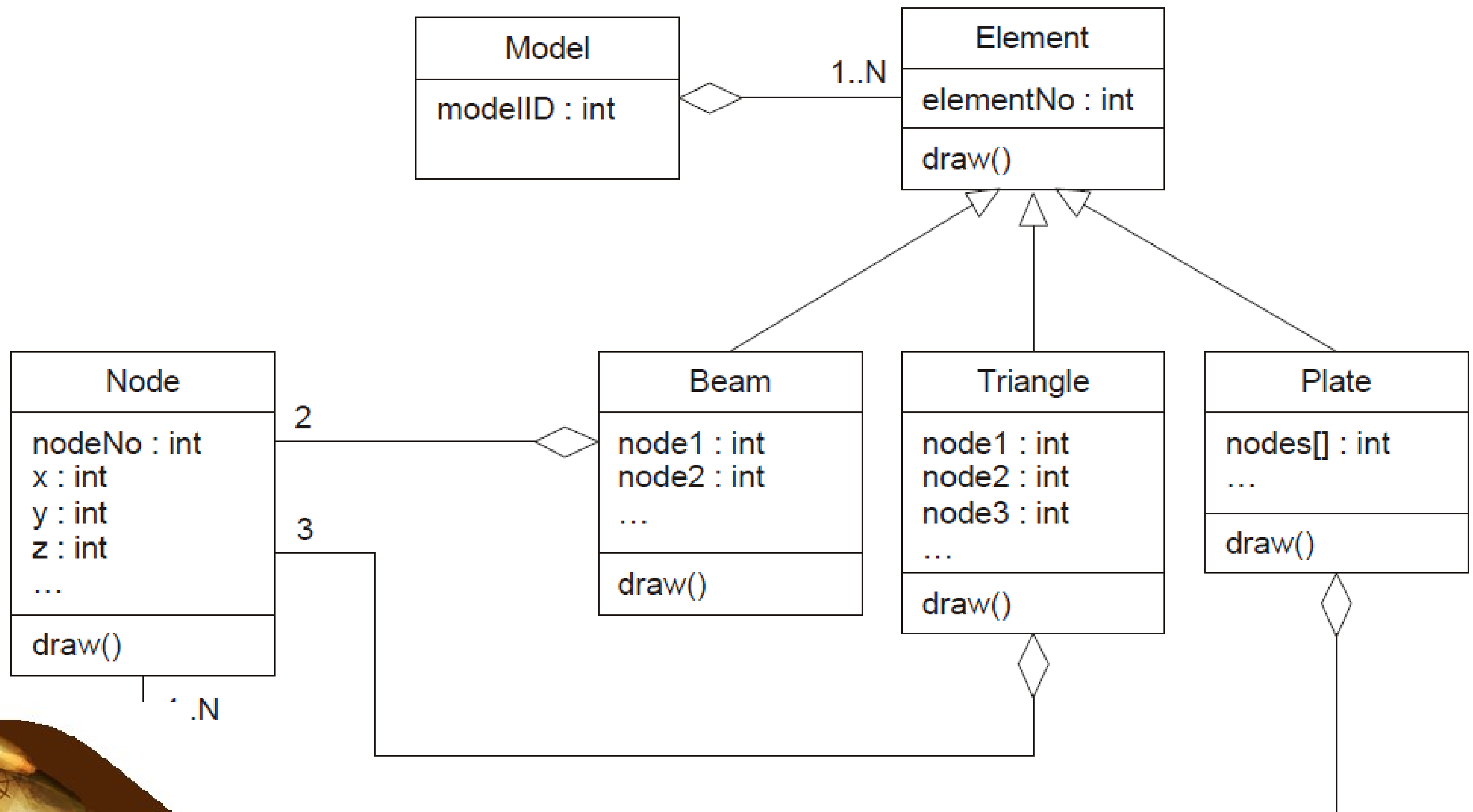
Performance Anti-patterns

Performance Anti-Patterns

- A *pattern* is a common solution to a problem that occurs in many different contexts. Patterns capture expert knowledge about “best practices” in software design in a form that allows that knowledge to be reused and applied in the design of many different types of software.
- An anti-pattern documents common mistakes made during software development as well as their solutions.
- A performance anti-pattern can lie at the
  - *software architecture or design* level, or
  - *code* level

[Smith et al., WOSP 2000]

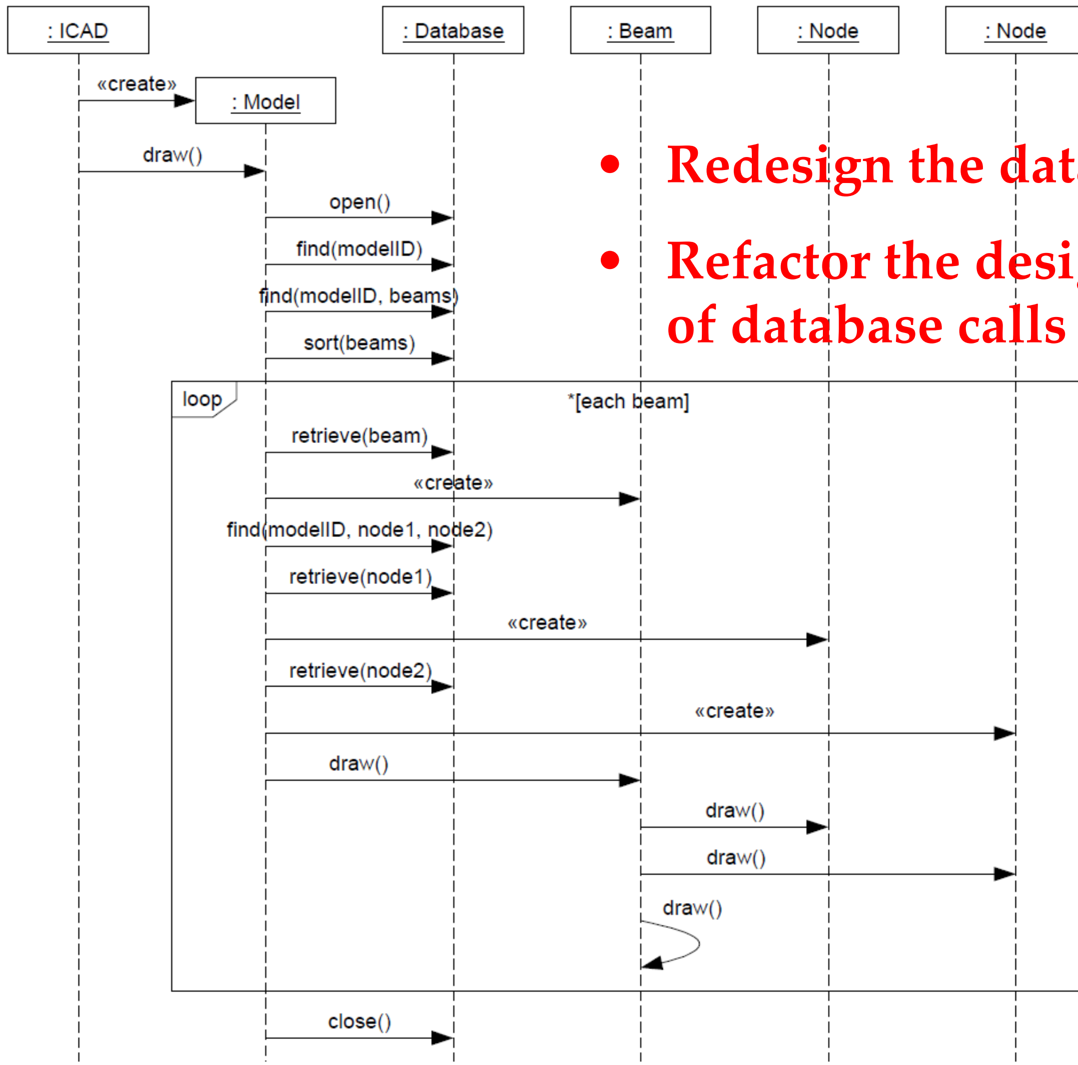
Design-level Anti-patterns - Circuitous Treasure Hunt



[Smith et al., WOSP 2000]

Design-level Anti-patterns - Circuitous Treasure Hunt

- Redesign the database schema
- Refactor the design to reduce the # of database calls



[Smith et al., WOSP 2000]

Code-level Anti-patterns - Repetitive Computations

Question: where is the redundant computation?

```
// Simplified from the XYPlot class in JFreeChart
public void render(...) {
  for (int item = 0; item < itemCount; item++) { // Outer Loop
    renderer.drawItem(...item...); // Calls drawVerticalItem
  }
}
// Simplified from the CandlestickRenderer class in JFreeChart
public void drawVerticalItem(...) {
  int maxVolume = 1;
  for (int i = 0; i < maxCount; i++) { // Inner Loop
    int thisVolume = highLowData.getVolumeValue(series, i).intValue();
    if (thisVolume > maxVolume) {
      maxVolume = thisVolume;
    }
  }
  ... = maxVolume;
}
```

A JFreeChart Performance bug

[Nistor et al., ICSE 2013]

Detecting architecture/design level anti-patterns

- Define software performance requirements for the system (response time, throughput and utilization)
- Encode the studied system architecture into the PCM with service demands and workload
- Encode the design/architecture level performance anti-patterns using rules
- Analyze the performance of the PCM model to see if it violates any performance requirements
- (If there are performance requirements violated,) Detect the performance anti-patterns using the encoded rules

[Trubianiet et al., JSS 2014]

Mining Historical Data for Performance Anti-patterns

- Randomly sampled 109 real-world performance bugs from five open source software systems (Apache, Chrome, GCC, Mozilla and MySQL)
- <u>**Static Analysis**</u>: Encode them as rule-checkers inside LLVM

Mozilla Bug 515287 & Patch	**What is this bug**

XMLHttpRequest::OnStop(){
  //at the end of each XHR
  …
-- *mScriptContext->GC();*
}
nsXMLHttpRequest.cpp
	This was not a bug until Web 2.0, where doing garbage collection (*GC*) after every XMLHttpRequest (XHR) is too frequent.
	
	It causes Firefox to consume **10X more CPU** at idle GMail pages than Safari.

An example of a Mozilla bug – Intensive GCs

[Jin et al., PLDI 2012]

Performance Anti-patterns - Repetitive Loop Iterations

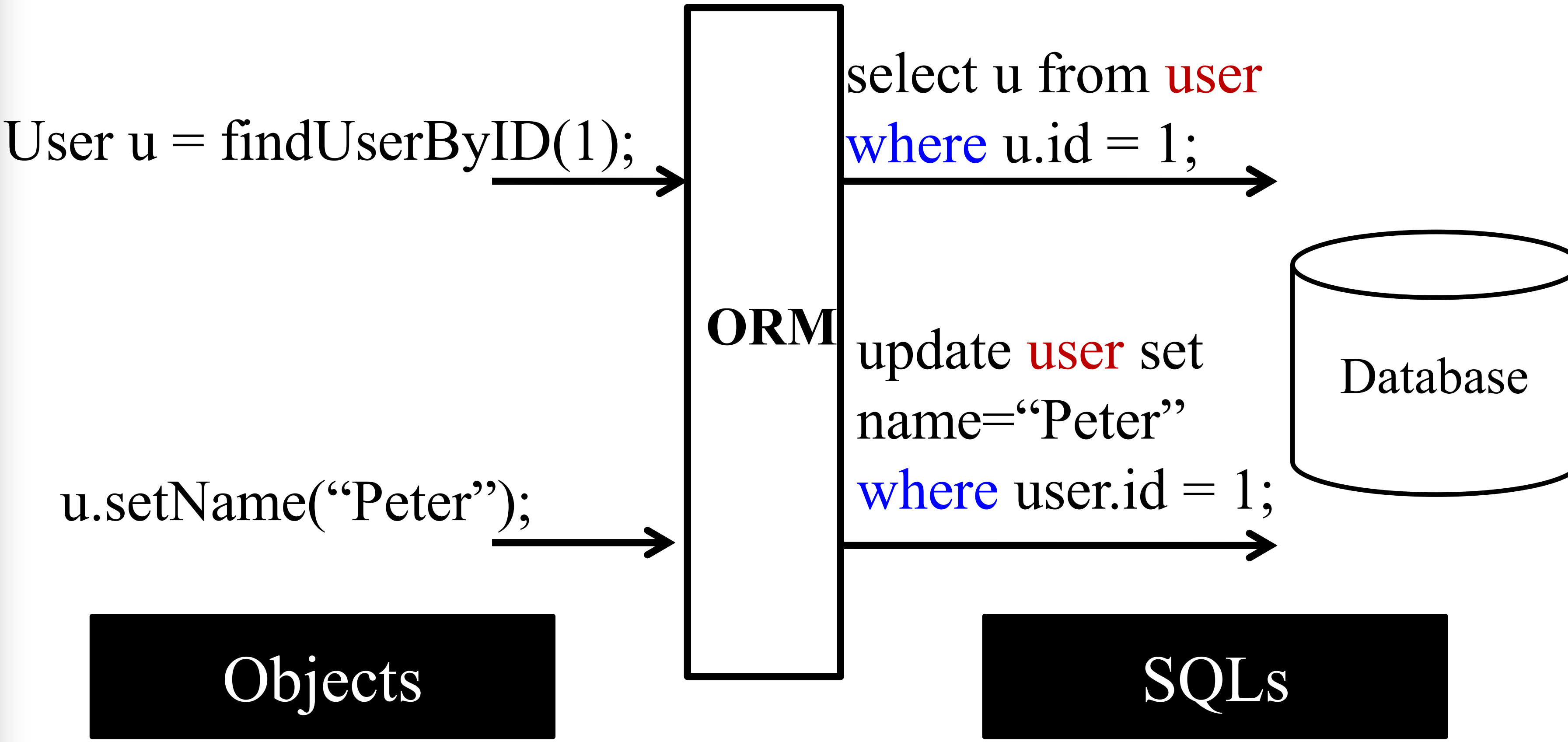
- **Dynamic Analysis**: Using the *soot* framework to detect similar memory access in the loops

```
// Simplified from the XYPlot class in JFreeChart
public void render(...) {
  for (int item = 0; item < itemCount; item++) { // Outer Loop
    renderer.drawItem(...item...); // Calls drawVerticalItem
  }
}
// Simplified from the CandlestickRenderer class in JFreeChart
public void drawVerticalItem(...) {
  int maxVolume = 1;
  for (int i = 0; i < maxCount; i++) { // Inner Loop
    int thisVolume = highLowData.getVolumeValue(series, i).intValue();
    if (thisVolume > maxVolume) {
      maxVolume = thisVolume;
    }
  }
  ... = maxVolume;
}
```

A JFreeChart Performance bug

[Nistor et al., ICSE 2013]

Accessing the Database Using ORM



[Chen et al., ICSE 2014]

Performance Anti-patterns in Hibernate

```
Company company = em.find(Company.class, companyID=1);
for (Department d : company.getDepartment()) {
        List<Employee> e = d.getEmployee();
        for (Employee tmp : e) {
                tmp.getId();
        }
}
```

select c **from company** c **where** c.ID = 1
select e **from employee** e **where** e.ID = departmentID.1
select e **from employee** e **where** e.ID = departmentID.2
…
select e **from employee** e **where** e.ID = departmentID.n

[Chen et al., ICSE 2014]

Performance Anti-patterns in Hibernate

```
@Fetch(FetchMode.SUBSELECT) private List<Employee> employee
Company company = em.find(Company.class, companyID=1);
for (Department d : company.getDepartment()) {
        List<Employee> e = d.getEmployee();
        for (Employee tmp : e) {
                tmp.getId();
        }
}
```

```
select c from company c where c.ID = 1
select * from employee e where e.departmentID
        = (select departmentID where department.company.id = 1)
```

	20 Department, 10 Employee	200 Department, 10 Employee	20000 Department, 10 Employee
Before (ms)	282 ms	1238ms	20462ms
After (ms)	214ms (+24%)	715ms (+42%)	6382ms (+69%)

[Chen et al., ICSE 2014]