User-Friendly Performance Engineering with Java Modelling Tools (JMT)

Giuliano Casale
Department of Computing
Imperial College London, UK
About Myself

- Software performance engineering
  - Quality-aware DevOps
  - Performance measurement and inference
  - Modelling and resource management
- More infos at: https://wp.doc.ic.ac.uk/gcasale/
Agenda

- Introduction to Java Modelling Tools
- Getting started: modeling a simple server
- A real enterprise application
- Advanced features
- Mean-value analysis
Example: Website Capacity Planning

- Web request processed by multiple servers
  - Web server (e.g. Apache, IIS, ...)
  - Application server (e.g. Tomcat, Glassfish, ...)
  - Database server (e.g. MySQL, Oracle, ...)
- How to predict response time taking into account server inter-dependences?
Why not just linear predictions?

Data: TPC-W e-commerce store

Performance trends are non-linear!

Throughput [request/s]

<table>
<thead>
<tr>
<th>number of concurrent users</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
</tr>
<tr>
<td>40</td>
</tr>
</tbody>
</table>

Linear Predictor

Measured

“Knee”
Why is the problem difficult?

HTTP Requests in the WS (Web Server)

Observation period $T$

1 request in WS

0 requests in WS

3 requests in SUT
Why is the problem difficult?

- Operating systems schedule jobs in round robin
  - If $n$ requests run simultaneously, each will approximately receive $1/n$ of the CPU time
  - Process Sharing is a round robin where the quantum of time assigned to each request is infinitesimal

![Diagram showing CPU with 3 requests running and service time S of the yellow request]
Predictive Modelling in JMT

- Model applications using a queueing network
- Resources modelled as queues where requests arrive, get served after queueing, and leave

Arrivals -> Web server (Processor Sharing Queue, Models CPU round-robin scheduler) -> DB server (First-come First-served Queue, Dual-server queue, Models dual-CPU server) -> Storage server (First-come First-served Queue) -> Response
Introduction to Java Modelling Tools
Obtaining JMT

- Developed since 2002 by 10+ generations of PG and UG students at PoliMi and Imperial College
- JAR, source code and ANT build scripts at http://jmt.sourceforge.net/Download.html
- JMT is open source: GPL v2
  - size: ~4,000 classes; 21MB code; ~200k lines
- Users are welcome to contribute with bug reports and «Request a new feature»
Java Modelling Tools (http://jmt.sf.net)
JMT: Architecture

CLI XML

GUI

JABA, JMVA, JWAT

JSIMwiz

JSIMgraph

XML

XSLT

XML

JMT framework

jSIMengine

Status Update

JMT
JSIMgraph & JSIMwiz
JSIM – discrete-event engine

- Simulation components defined by 3 sections

  queueing station

- Discrete-event simulation of section messaging
Advanced statistical analysis is fully automated
Automated (but over-ridable) simulation stop

Intelligent filtering of simulation data
- R5 heuristics, spectral analysis, MSER-5 rule, ...
Solution of basic queueing networks
  Algorithmic solution requires seconds or less
Who uses JMT?

- JMT is for PE practice, teaching, and research
- Courses, papers and books help you get started
Getting Started with JSIMgraph
Multi-threaded Web server

\[ \lambda = 1\text{–}20 \, \text{r/s} \]

Clients \rightarrow \text{thread requests queue (inside the server)} \rightarrow \text{server} \rightarrow \text{CPU} \rightarrow \text{I/O} \rightarrow \text{sink}

\[ D_{\text{CPU}} = 0.010s \quad D_{\text{I/O}} = 0.047s \]

Threads = \([1, N]\)
Model definition (unlimited threads and buffer size)

\[ \lambda = 1 - 20 \text{ req/sec} \]
Input parameters: service time

Number of servers: 1

Mean service time:
- For service section 1: $\frac{1}{100} = 0.010\ s$
- For service section 2: $\frac{1}{21.277} \approx 0.047\ s$

Number of cores: 1

Service time distribution:
- Req: Load Independent, $\exp(100)$
- Req: Load Independent, $\exp(21.277)$
Input parameters: service distribution

probability density function: $f(x) = \lambda \exp(-\lambda x)$

Exponential $[\exp(\lambda)]$:

$$f(x) = \lambda e^{-\lambda x}$$

Selected Distribution: Exponential

External trace
Input parameters: arrival distribution

- **Sun**
- **Mon**

![Graph showing varying peaks of user activity with a window for editing Class1 distribution.](image)

**Burst**

Define two interval types A and B, their probabilities of occurrence, the distributions of the interval lengths and of the inter-event times.
System Response time \((\lambda=20\;\text{req/sec})\)

### System Number of Customers
Average customer number for each chosen class.

<table>
<thead>
<tr>
<th>Station Name:</th>
<th>-- Netwo</th>
<th>Class Name:</th>
<th>-- All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf.Int/Max Rel.Err:</td>
<td>0.95 / 0.</td>
<td>Analyzed samples:</td>
<td>15000</td>
</tr>
<tr>
<td>Min:</td>
<td>8.3130</td>
<td>Max:</td>
<td>9.200</td>
</tr>
<tr>
<td>Average value:</td>
<td>8.7567</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The simulator failed to compute this measure with the specified confidence interval and maximum relative error because the number of samples needed is greater than the max allowed. The actual simulation parameters are different from the default values of parameters.
$\lambda = 1-20 \text{ req/s, unlimited threads & queue size (JSIMgraph)}$

$U_{I/O} = \lambda D_{I/O} = 20 \times 0.047 = 0.94 \text{ (exact)}$

Utilization of I/O

$R = 0.784 \text{ s (sim)}$

System Response time

$R = 0.795 \text{ s (exact)}$

$X = 19.86 \text{ r/s}$

Throughput

same as $\lambda$

No limitations

System Power
Finite Capacity Regions – FCR

- Modelling thread limits via FCRs
FCR parameters

- **Global capacity of the FCR**: 15
- **Max number of requests per class in the FCR**: infinite (for both constraints)
- **Drop the requests when the region capacity is reached**: true (for both constraints)
Queue-length at the web server

unlimited

15 threads

10 threads

5 threads
What is the impact of admission control?

$\lambda = 20 \text{ r/s}$

clients

queue for threads with finite capacity (outside the server)

Blocking After Service policy

sink

server

D$_{\text{server}} = 0.047 \text{s}$

threads = 5

drop policy
Block After Service (BAS) blocking

Station Name: Server

Station queue policy: Non-preemptive Scheduling

Queue Policy:
- Class: Class0
- Queue Policy: FCFS
- Drop Rule: BAS blocking

Station with finite capacity:
- max num. customers: 5

Selection of the BAS policy:

BAS policy:
- Requests are blocked in the sender station when the max capacity of the receiver is reached
## Comparison Results

<table>
<thead>
<tr>
<th>λ=20 req/s</th>
<th>#req</th>
<th>Resp Time</th>
<th>Util</th>
<th>Tput req/s</th>
<th>Drop</th>
<th>Queue and Server stations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qsize=∞ Ser=5, queue</td>
<td>0</td>
<td>16.11</td>
<td>0.77</td>
<td>0.95</td>
<td>20.06</td>
<td>0</td>
</tr>
<tr>
<td>Qsize=∞ Ser=5, BAS</td>
<td>11.03</td>
<td>4.77</td>
<td>0.53</td>
<td>0.24</td>
<td>19.82</td>
<td>0</td>
</tr>
<tr>
<td>Qsize=5 drop Ser=5, BAS</td>
<td>0.94</td>
<td>3.82</td>
<td>0.05</td>
<td>0.20</td>
<td>18.76</td>
<td>1.14</td>
</tr>
<tr>
<td>Qsize=∞ Ser=5, drop</td>
<td>0</td>
<td>2.34</td>
<td>0.136</td>
<td>0.812</td>
<td>17.16</td>
<td>2.866</td>
</tr>
</tbody>
</table>
Modelling a Real Enterprise Application
ERP Transaction Lifecycle

Client-Side

Application Server

DB Server

Request arrives

Admission control

Load context in memory

CPU

Data access

CPU

Response arrives

Network latency (1)

Queueing time

Worker Thread

Worker thread admission time

Service time (1)

Simultaneous Resource Possession

Service time (2)

DB query time (1)

Service time (3)

DB query time (2)

Network latency (2)

Server Response time

Request Response time
Modelling Abstraction

Client-Side

Request arrives

Network latency (1)

Queueing time

Server-Side

Request arrives

Admission control

Worker Thread

Load context in memory

Application

CPU

Server Steps

Data access

CPU+I/O

DB Server Steps

Data access

CPU+I/O

DB query time (1)

DB query time (2)

Network latency (2)

Response arrives

Response arrives

Server admission time

Service time (1)

Service time (2)

Service time (…)

Request

Response time

Request

Response time

Server

Response time

Client
Modelling Multi-Tier Applications

N=300 app users

WorkerThreads

Think Time

Load context

Send to jMVA

simulate

Exponential Distributions

4 Servers (Cores)

Zload = 0.015s

Scpu = 0.072s

Sdb = 0.032s

Appl. Server CPUs

Database

FCR

Application CPU

PS scheduling

Exponential Distributions

Global WorkerThreads Properties

Region name: WorkerThreads

Region capacity: 4

FCR Capacity

FCR Admission Policy

Class specific WorkerThreads Properties

Class: Transactional workload

Capacity: infinite

Drop: false
Example: Closed Model of ERP Software

Gallery of Advanced JSIM Features
Multiclass Models

- Many types of concurrent workloads can be specified (e.g., FTP, HTTP, HTTPS, ...)
- Support for external arrivals (open) or fixed job populations (closed)
Class-Switching

- A job can change class during the simulation
- Requests issued by a browser change over time

![Diagram](image.png)

<table>
<thead>
<tr>
<th>CS Strategies</th>
<th>Class0</th>
<th>Class1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.6 (60%)</td>
<td>0.4 (40%)</td>
</tr>
<tr>
<td></td>
<td>0.8 (80%)</td>
<td>0.2 (20%)</td>
</tr>
</tbody>
</table>
Fork-Join

- Synchronizations between jobs often needed
  - MapReduce, RAID, In-Memory Databases, ....
- JSIM supports fork-join constructs
Load-Balancing

- Router element allow to specify smart routing
  - Join-the-shortest-queue
  - Probabilistic routing
  - Round-robin
  - Join least utilized queue
  - ...

![Diagram of load balancing with routers and web servers]
Service requirements are allowed to vary in JSIM and JMVA with requests in the queue.
Performance Indices - Statistics

- Probability of more than X waiting jobs?
- Or that the response time is less than Y?
Performance Indices - Logging

- Simulation events can be traced in CSV files

```
LOGGERNAME;TIMESTAMP;JOB_ID;CLASS_ID;INTERARRIVALSAMECLASS;INTERARRIVALANYCLASS;SIMULSTART_TIME
req arrive;0.009420010041266842;253625;Transactional workload;;
req leave;0.0217557654334812;253625;Transactional workload;;
req arrive;0.031032734664243056;253498;Transactional workload;;
req leave;0.0491599532909814;253498;Transactional workload;;
req arrive;0.07727161520772474;253542;Transactional workload;;
```
What-If Analysis

- Perform repeated executions automatically

![Image of a graph showing the relationship between confidence interval range and a ratio between assumed service time at Web Server 1 and the initial one. The graph indicates an increasing trend with error bars.]

10 jSIM invocations
What-If Analysis with JMVA & JABA
Many algorithms have been proposed to analyse models faster than simulation. MVA can take seconds where SIM take minutes. Approximations can take milliseconds. Useful to explore a large range of configurations. e.g., bottleneck identification.
Execution time [ms] at Storage 2 (class 1 bottleneck)

Execution time [ms] at Storage 1 (class 2 bottleneck)

Storage 3: potential system bottleneck?
JMVA: Solutions

Choice of solution algorithm

What-if analysis
Select a control parameter if you want to change the models with its values changing it. The performance indices will be recalculated.

MVA
RECAL
MoM
CoMoM
Approximate
Chow
Bard-Schweitzer
AQL
Linearizer
JABA: Bottleneck Identification

Common saturation sector

Fraction of class 2 jobs that saturate two resources concurrently
JMVA: What-If

Throughput

Response times
What’s Next?
Coming Soon

- New release (0.9.2) due in approx. 2 months
  - First release of statistics in jSIM
  - Many minor fixes across 18 months of work
- Beyond 0.9.2:
  - Migration to MAVEN & jUnit
  - Large-scale MVA for sparse models
  - More scheduling policies: SJF, LJF, SRPT, ...
  - SLA Violations
Questions?

g.casale@imperial.ac.uk
https://wp.doc.ic.ac.uk/gcasale/
FCR support also threading levels differentiated for each class

Service demands can be estimated using linear regression of utilizations vs throughputs

- [http://www.doc.ic.ac.uk/~gcasale/content/pdp08robust.pdf](http://www.doc.ic.ac.uk/~gcasale/content/pdp08robust.pdf)

or response time vs queue-length

- [http://www.doc.ic.ac.uk/~gcasale/content/valuetoools09estimating.pdf](http://www.doc.ic.ac.uk/~gcasale/content/valuetoools09estimating.pdf)

see example in next two slides for a simple model of a web server
Example: Apache Web Server

Methodology:

- Convince the administrator to enable logging... often the most difficult task!
- Observe the web server for a time period of length T (the longer the better)
- Collect utilization logs and Apache access log, e.g.:

  ...

Example: Apache Web Server

- From the logs, obtain total number of arrivals in $T$ (denoted by $A$) and mean utilization $U_{apache}$
- $\lambda = A / T$ is now the requests / second that arrive at the webserver
- Estimate service demand $D_{apache} = U_{apache} / \lambda$
- Use open model to make predictions
  - e.g., predict change of response times if CPU is upgraded to run 20% faster ($D_{apache-new} = 0.80 \ D_{apache}$)
- For more complex models, see http://www.doc.ic.ac.uk/~gcasale/content/pdp08robust.pdf