

60017 PERFORMANCE ENGINEERING

Load testing

This lecture

- ▶ Benchmarking a distributed application
 - ▶ Case study: SPECjbb2015
- ▶ Load testing a distributed application

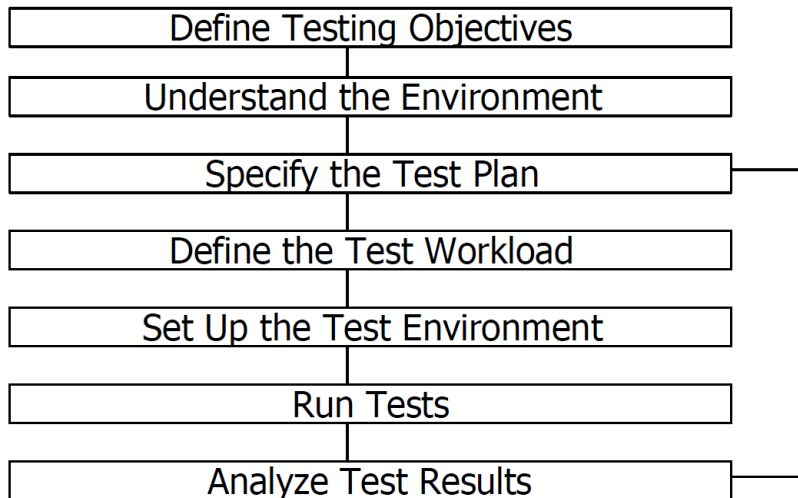
Introduction

- ▶ IT systems undergo regular cycles of performance testing:
 - ▶ to **certify** performance relatively to a standard workload (**benchmarking**)
 - ▶ to ensure that they **satisfy service level agreements (SLAs)** under increasing workload intensities (**load testing**)
- ▶ Performance testing involves injecting a chain of requests into the system and monitoring the results.
- ▶ The test is often carried out in a **controlled environment**, where the system can be monitored without noise.

Introduction

- ▶ Several **tools** exist to carry out the tests, differing for:
 - ▶ License (commercial vs open source)
 - ▶ Types of workloads injected into the system
 - ▶ Ability to customize the test
 - ▶ Parallelization support (e.g., multiple load injection points)
 - ▶ Degree of automation
- ▶ Each test collects several **performance measurements**, typically **response times**, **throughputs**, or **ad-hoc metrics**
 - ▶ *e.g., SAP benchmarks use a metric called SAPS.*
 - ▶ *100 SAPS equal completing 2000 order items per hour.*
 - ▶ *1000 SAPS equal completing 20,000 order items per hour.*

General testing methodology



Benchmarks vs micro-benchmarks

- ▶ A benchmark tests the response of the system against a **reference** workload.
- ▶ We focus on benchmarks for distributed applications that use **realistic workloads**, as opposed to micro-benchmarks.
- ▶ Micro-benchmarks are designed to test a **critical part** of the system in order to help optimising it.
 - ▶ *e.g., locking schemes, system call latency, kernel modules, ...*
- ▶ Micro-benchmarks tend to be just a few lines of code.
 - ▶ *e.g., quicksort, sieve of Eratosthenes, ...*
- ▶ Benchmarks are better suited than micro-benchmarks to assess if a system is fit for its **business purpose**.
 - ▶ Don't use a micro-benchmark to decide on hardware upgrades!

Timescale analysis

Consider the following execution timescale:

scale	runtime	workload/operation type
	> 10000 s	MapReduce, TPC-C, TPC-H, ...
	1...10000 s	SPECjvm2008, SPECjbb2015, ...
milli	1...1000 ms	single website request (dynamic content), ...
micro	1...1000 us	single website request (static content), ...
nano	1...1000 ns	complex CPU operations, cache miss, ...
pico	1...1000 ps	CPU pipelining, integer addition, ...


Source: Oracle (adapted)

- ▶ Fast-executing operations can be difficult to isolate inside measurements of complex long-running operations.
 - ▶ $10s + 10ns \approx 10s$
- ▶ Micro-benchmarks help 'zooming' on fast-executing operations, acting similarly to a measurement **microscope**.

Properties of benchmarks

- ▶ Benchmarks are more complex than micro-benchmarks.
- ▶ Non-profit organizations such as TPC and SPEC work with vendors to develop standard industry benchmarks.
- ▶ A good benchmark should ensure that the workload is:
 - ▶ **Repeatable**: if the benchmark is run twice on the same system, results will be very similar.
 - ▶ **Interpretable**: it must be easy to tell what the results mean and how well the system performed.
 - ▶ **Comparable**: following a given protocol, we can compare results on two systems and draw meaningful conclusions.
 - ▶ **Vendor-neutral**: the benchmark should avoid to systematically favour systems from a certain company.
 - ▶ **Transparent**: what the benchmark does is well documented.

SPEC Benchmarks



Standard Performance Evaluation Corporation

[Home](#) [Benchmarks](#) [Tools](#) [Results](#) [Contact](#) [Site Map](#) [Search](#) [Help](#) [f](#) [in](#) [tw](#) [g+](#)

Benchmarks

- Cloud
- CPU
- Graphics/Workstations
- ACCEL/MPI/OMP
- Java Client/Server
- Mail Servers
- Solution File Server
- Power
- Virtualization
- Web Servers
- Results Search

SPEC's Benchmarks

Cloud



- SPEC Cloud_1aaS 2016**
[\[benchmark info\]](#) [\[published results\]](#) [\[order benchmark\]](#)
SPEC's first benchmark suite to measure cloud performance SPEC Cloud_1aaS 2016's use is targeted at cloud providers, cloud consumers, hardware vendors, virtualization software vendors, application software vendors, and academic researchers. The benchmark addresses the performance of infrastructure-as-a-service (1aaS) public or private cloud platforms. The benchmark is designed to stress provisioning as well as runtime aspects of a cloud using I/O and CPU intensive

- ▶ SPEC is a non-profit organization known for its benchmarks for CPU, Java, and data center infrastructure.
- ▶ Its membership includes almost 150 organizations, spanning ICT industry and academia (... including Imperial College.)

Some popular SPEC Benchmarks

- ▶ [SPEC CPU2006](#): integer and floating-point CPU operations
- ▶ [SPECjbb2015](#): multi-tier Java applications
- ▶ [SPEC Cloud_IaaS 2016](#): MapReduce, Cassandra
- ▶ [SPECjms2007](#): message-oriented middlewares
- ▶ [SPECjvm2008](#): java runtime
- ▶ [SPECpower_ssj2008](#): server and equipment power usage
- ▶ ...

TPC Benchmarks



he TPC is a non-profit corporation focused on developing data-centric benchmark standards and disseminating objective, verifiable performance data to the industry... T

[Document Search](#)[Member Login](#)

[Home](#)[About the TPC](#)[Benchmarks](#)[Enterprise BMs](#)[TPC-C](#)[TPC-DS](#)[TPC-E](#)[TPC-H](#)[TPC-VMS](#)[Express BMs](#)[TPC-X-BB](#)[TPC-X-HS](#)[TPC-X-V](#)[Common Specifications](#)[TPC-Pricing](#)[TPC-Energy](#)[Submission Checklist](#)[Obsolete BMs](#)[TPC-A](#)[TPC-App](#)[TPC-B](#)[TPC-D](#)[TPC-R](#)[TPC-VW](#)

Active TPC Benchmarks

TPC-C

TPC-C simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered around the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service.

TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

TPC-C performance is measured in new-order transactions per minute. The primary metrics are the transaction rate (tpmC), the associated price per transaction (\$/tpmC), and the availability date of the priced configuration.

[more >>](#)

- ▶ TPC is a non-profit corporation focussed on data-centric benchmarking, e.g., database, Big data, decision support, ...
- ▶ TPC's membership includes major database vendors.
- ▶ Members can submit performance benchmarking results to a public database.

Some popular TPC Benchmarks

- ▶ TPC-C: database transactions (short-running, read/write)
- ▶ TPC-H: analytical database queries (long-running, read-only)
- ▶ TPC-DS: Big data analytics (e.g., Hadoop/Spark)
- ▶ TPC-DI: data integration and transformation
- ▶ ...

Example: benchmark comparability

TPC-H - Top Ten Performance Results

Version 2 Results As of 24-Jan-2017 at 6:39 PM [GMT]




Note 1: The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons. The TPC-H results shown below are grouped by database size to emphasize that only results within each group are comparable.


Note 2: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

☒ All Results ☐ Clustered Results ☐ Non-Clustered Results Currency United States - Dollar (USD) ▼





100 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1		Dell PowerEdge R720xd using EXASolution 5.0	1,582,736	.12 USD	NR	09/24/14	EXASOL EXASolution 5.0	EXASOL EXACluster OS 5.0	09/23/14	Y

300 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1		Dell PowerEdge R720xd using EXASolution 5.0	2,948,721	.12 USD	NR	09/24/14	EXASOL EXASolution 5.0	EXASOL EXACluster OS 5.0	09/23/14	Y

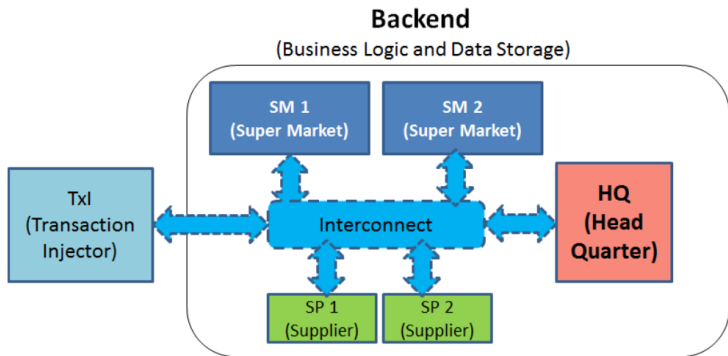
1,000 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1		Dell PowerEdge R720xd using EXASolution 5.0	5,246,338	.14 USD	NR	09/24/14	EXASOL EXASolution 5.0	EXASOL EXACluster OS 5.0	09/23/14	Y
2		HPE ProLiant DL380 Gen9	678,492	.64 USD	NR	07/31/16	Microsoft SQL Server 2016 Enterprise Edition	Microsoft Windows Server 2012 R2 Standard Edition	03/24/16	N
3		Cisco UCS C460 M4 Server	588,831	.97 USD	NR	12/16/14	Microsoft SQL Server 2014 Enterprise Edition	Microsoft Windows Server 2012 R2 Standard	12/15/14	N
4		HPE ProLiant DL380 Gen9	543,102	.69 USD	NR	07/31/16	Microsoft SQL Server 2016 Enterprise Edition	Microsoft Windows Server 2012 R2 Standard Edition	03/09/16	N

Learning by example: SPECjbb2015

- ▶ Benchmarks for multi-tier applications are rather similar.
- ▶ We therefore focus on a case study: SPEC Java Business Benchmark 2015 (SPECjbb2015).
- ▶ **Goal of SPECjbb2015:** evaluate performance and scalability of **environments** for Java business applications.
- ▶ Using SPECjbb2015 you can decide if you have a properly sized IT infrastructure in terms of servers, network, JVMs, ...
- ▶ Simulates a supermarket company IT infrastructure
 - ▶ Handling sales in local supermarkets and online purchases
 - ▶ Managing coupons/discounts and customer payments
 - ▶ Managing receipts, invoices and user database
 - ▶ Interacting with suppliers for replenishing the inventory
 - ▶ Mining operations data in the company headquarters

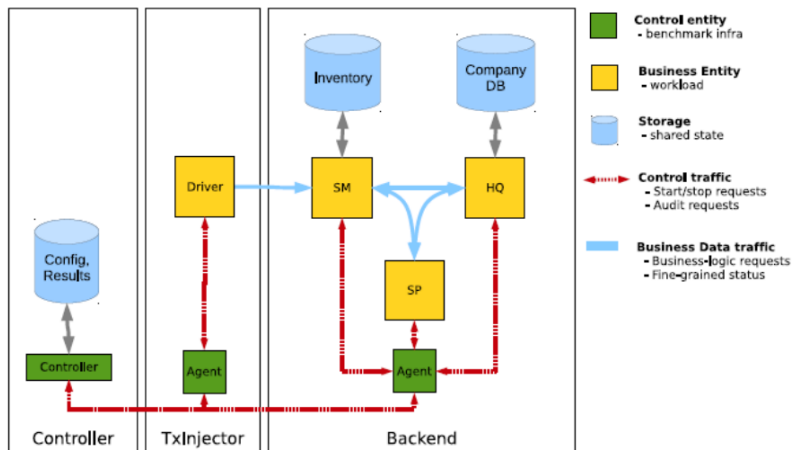
SPECjbb2015: supermarket application



SPECjbb2015: main features

- ▶ SPECjbb2015 provides:
 - ▶ An [implementation](#) of the supermarket application based on the [latest Java technologies](#).
 - ▶ Guidelines to [install and configure](#) the application on different operating systems and hardware architectures.
 - ▶ A mechanism to [simulate](#) the execution of the supermarket application on the user's IT infrastructure.
 - ▶ A mechanism to [scale](#) the workload injection to large transaction volumes.
 - ▶ Monitoring, collection, and filtering of the [results](#).
 - ▶ Generation of result reports including [summary metrics](#).

SPECjbb2015: benchmark architecture



SPECjbb2015: controller

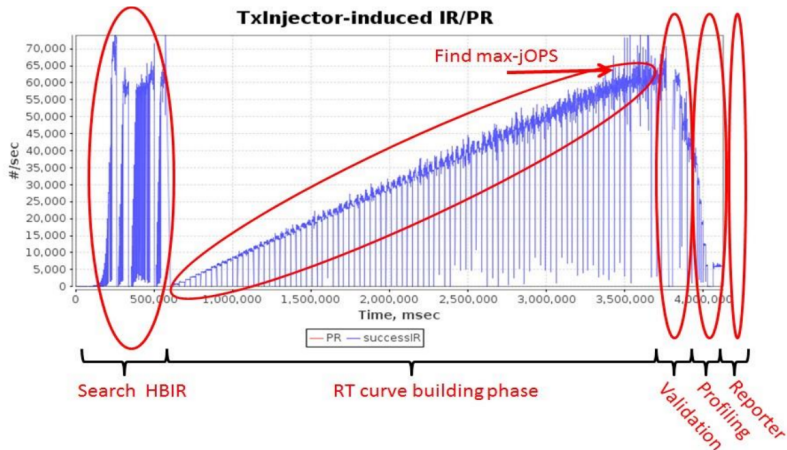
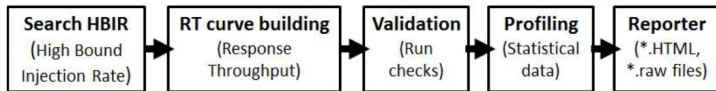
SPECjbb2015 works as follows:

- ▶ A **Controller** agent setups the initial environment according to a user-specified configuration.
- ▶ The benchmark uses a single Controller.
- ▶ The Controller has the following responsibilities:
 - ▶ **Synchronise** start/stop of the load injectors
 - ▶ Coordinate the system through **benchmarking phases**
 - ▶ **Dispatch work** to the load injectors
 - ▶ **Send heartbeats** to stop/restart the test in case of node failure
 - ▶ Collect, aggregate, and store the **test results**
- ▶ The Controller also controls **backend agents** that start/stop/configure the application.

SPECjbb2015: load injectors

- ▶ One or more transaction injectors (**TxInjectors**) send workload to the system.
- ▶ Each injector has the following duties:
 - ▶ Control a **JVM thread pool**, where each thread is responsible for issuing requests to the system and waiting for replies.
 - ▶ Ensure that each thread waits some time before issuing the next request (so-called **think time**).
 - ▶ Inject load using a suitable **transport protocol** (e.g. Java NIO).
 - ▶ **Report results** to the Coordinator.

SPECjbb2015: execution phases

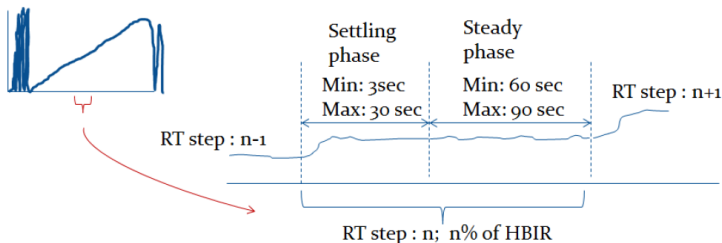


SPECjbb2015: execution phases

- ▶ **Search HBIR**: experiments aimed at determining the maximum sustainable request rate for the system.
 - ▶ Knowing the HBIR prevents the system from becoming unstable during testing (e.g., due to disk swapping).
 - ▶ At the end of the HBIR phase, the benchmark re-initializes data structures.
- ▶ **RT curve building**: transaction rates are increased linearly (+1% each step) and kept constant for a fixed period, recording transaction response times.
- ▶ **Validation**: the validity of the experiment is established, for example checking failed transactions.
- ▶ **Profiling**: monitoring data is collected and analysed.
- ▶ **Reporter**: a report or web page is generated automatically to visualize the results.

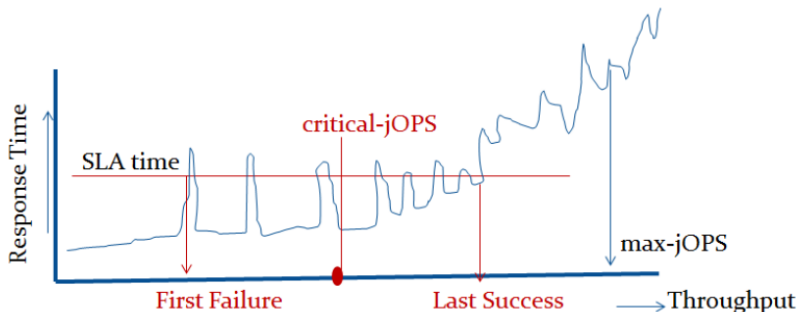
SPECjbb2015: transient and steady phases

- ▶ At each step of RT curve building a settling phase is used to ensure that the system adapts to the new load. This is also called the **transient phase**.
- ▶ The transient phase ends when the injection rate matches the target rate $\pm 1\%$.
- ▶ Only the **steady phase** data is used to measure performance.



SPECjbb2015: summary performance metrics

- ▶ **max-jOPS**: maximum sustainable injection rate, above which the system does not converge to steady-state within 90s.
- ▶ **critical-jOPS**: mean of injection rates of first failure and of last success of SLA (99th percentile on response time)



SPECjbb2015: limitations

- ▶ The benchmark is meant to test only the platform (runtime, middleware, ...) and the infrastructure (servers, network, ...) for the reference application.
- ▶ Creating a test workload for an arbitrary application can be done in two ways:
 1. **customize** an existing benchmark
 - ▶ e.g., *SAP-H* is a customised *TPC-H* for SAP applications.
 2. **use** a customisable testing tool for that application
 - ▶ e.g., *Apache Jmeter*, *Selenium*, *Azure PaaS testing*, ...
- ▶ The second method is the subject of **load testing**.

Load testing

- ▶ **Load testing** generalizes benchmarking to work with **user-defined workloads**.
- ▶ Often used by developers to check application scalability.
- ▶ Load testing tools allow to:
 - ▶ Simulate loads using **load injectors** or **emulated browsers**.
 - ▶ Emulate realistic user **think times** between request submissions.
 - ▶ Emulate **user abandonment** when requests take too long.
 - ▶ **Record** a user interaction for later reply.
 - ▶ Support **test configuration parameters**:
 - ▶ test duration
 - ▶ abandonment threshold
 - ▶ workload intensity
 - ▶ mix of requests to be submitted
 - ▶ ...

Video: emulated browsers

Firefox +

Connecting...

104.40.216.241 http://104.40.216.241:8443/eCommerce/control/newcat...

State/Province*

Virginia

Allow Address Solicitation

Phone Numbers

	Country	Area Code	Contact Number	Extension	Allow Solicitation
Home phone					
Business phone					
Fax number					
Mobile phone					

E-Mail Address

E-Mail Address*

9713517@domain.com

Allow Solicitation

User Name

Use E-Mail Address

User Name*

user-1430899713517

Password

Password*

Repeat password to confirm*

Password Hint

his is my password hint

Cancel Save

W3C CSS W3C XHTML 1.0

About Us

Copyright (c) 2001-2015 The Apache Software Foundation - www.apache.org

Powered by Apache OFBiz

<https://www.doc.ic.ac.uk/~gcasale/content/eb.mp4>

Example: Apache JMeter

- ▶ A wide-spread open source tool for load testing supporting:
 - ▶ Web (HTTP, HTTPS) traffic
 - ▶ Web services (SOAP)
 - ▶ Microservices
 - ▶ JDBC database transactions
 - ▶ Messaging services (JMS)
 - ▶ Mail (SMTP, POP3, IMAP)
 - ▶ Shell scripts execution
 - ▶ Thread pools
 - ▶ Distributed testing
 - ▶ Results collection and analysis
- ▶ Commercial versions exist in the cloud, e.g.: Blazemeter.

Example: Apache JMeter - workload specification

Apache JMeter (2.4 r961953)

File Edit Run Options Help

0 / 0

SubmitFormTest
50Users
submit
WorkBench

HTTP Request

Name: submit-signupform

Comments:

Web Server

Server Name or IP: localhost Port Number: 8080

Timeouts (milliseconds)

Connect: Response:

HTTP Request

Protocol (default http): Method: POST Content encoding:

Path: /signup.do

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data for HTTP POST

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?
username	test	<input type="checkbox"/>	<input checked="" type="checkbox"/>
password		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Add Delete

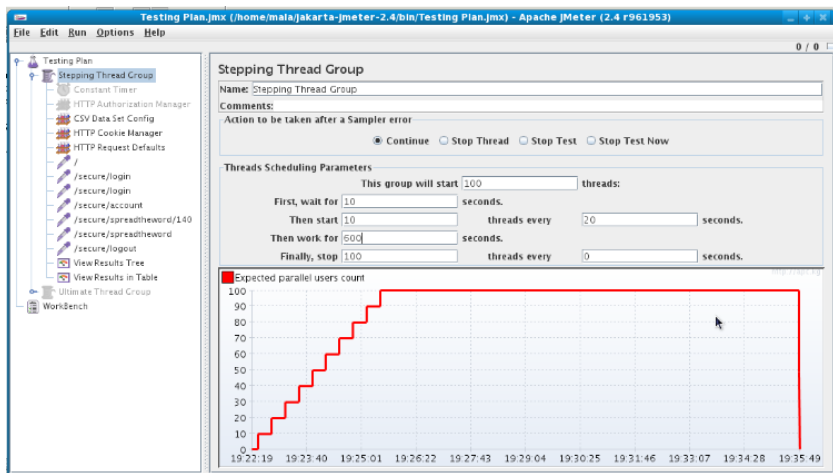
Send Files With the Request:

File Path:	Parameter Name:	MIME Type:

Add Browse... Delete

Proxy Server

Example: Apache JMeter - thread pool specification



Source: Google