Bench4Q Tool 1.2

A QoS Oriented E-commerce Benchmark

USER'S MANUAL

Revision Sheet

Release No.	Date	Revision Description
Rev. 1.0	09/24/2009	For Bench4Q Tool 1.0.0
Rev. 1.1	11/26/2009	For Bench4Q Tool 1.1.0
Rev. 1.2	05/10/2010	For Bench4Q Tool 1.2.X

Contents

1. Introduction

1.1 basic information

E-commerce，as a very popular business model, has grown substantially with the development of Internet technology. In the meantime, the QoS of e-commerce, critical to the success of e-commerce systems, is a major factor in the success of competition. While the infrastructures of an e-commerce system, such as application server, play an important role in QoS guarantees, how to evaluate the performance of these infrastructure techniques has become the focus of the issue.

Benchmark as a typical and impartial way to evaluate the e-commerce system is a key matter of people's concerns. However, the current e-commerce benchmarks that mainly concerning on transaction is lack of cares about customer's behavior. And they are lack of ability to measure the e-commerce's characteristic. Therefore, the current e-commerce Benchmark isn't an effective way to evaluate QoS guarantees of an e-commerce system.

Bench4Q is a new methodology for QoS oriented e-commerce benchmarking. Bench4Q has features to deduce a controllable and flexible representation of complex session-based workloads, and to simulate authentic customer behavior. What's more, we show that Bench4Q that mainly concerns on session, profit and QoS guarantee can provides an overall view of the system's performance.

Bench4Q Tool brings more conveniences in Bench4Q benchmarking. With this tool, we are offering a simply way for installation, configuration and metrics analysis.

1.2 Targeted Audience

This document is targeting two types of audience:

- People who just want to use right away the Bench4Q Tool. This is for those who will use the Bench4Q Tool to benchmarking the middleware.
- People who would like to modify Bench4Q Tool to fit their particular needs. You may want to change a little bit our Bench4Q Tool to add some functionality or replace a component with another one.

1.3 Structure of the document

This document will walk you over:

- A brief introduction to Bench4Q in section 2, for people who have never

used Bench4Q before.

- A brief introduction to Bench4Q Tool in section 3.
- Some basic information about Bench4Q Tool in section 4
- Bench4Q Tool Usage in section 5.

2. Bench4Q: A QoS Oriented E-commerce Benchmark

2.1 Overview

E-commerce(the use of the Internet to buy and sell goods and services)，as a very popular business model, has grown substantially with the development of Internet technology. In the meantime, the competition of e-commerce business has been more and more intense. E-commerce must provide a system, such as e-commerce website, in order to function. We have shown that in the context of web-based systems characterized by highly variable workload, it is difficult to find out how to make best use of current resource and how to find out the bottleneck. Finding a simple way to help e-commerce system provider with making decision has become the focus of the issue.

As a main way to evaluate the performance of e-commerce system, benchmark has become a key matter of people's concerns. The current e-commerce benchmark, such as TPC-W, mainly concerns on transaction. Without the ability of measuring QoS guarantee, it may mislead the user. By investigating the current e-commerce Benchmark, we find out the insufficient of them in measuring QoS guarantees. Then we introduce a new methodology for QoS oriented e-commerce benchmarking, named Bench4Q.

Bench4Q, as a QoS oriented e-commerce benchmark, has features to deduce a controllable and flexible representation of complex session-based workloads, and to simulate authentic customer behavior. What's more, we show that Bench4Q can be used to evaluate the system's performance and scalability.

Bench4Q has made many extensions of TPC-W specification for measuring QoS guarantees of e-commerce system infrastructures, especially of load simulation and metrics analysis.

2.2 What is TPC-W?

TPC Benchmark™ W (TPC-W) is a transactional web e-Commerce benchmark, introduced by the Transaction Processing Performance Council. TPC-W specifies an e-Commerce workload that simulates the activities of a retail website which produces heavy load on the backend database.

The workload is performed in a controlled internet commerce environment that simulates the activities of a business oriented transactional web server. The workload exercises a breadth of system components associated with such environments, which are characterized by:

- Multiple on-line browser sessions.

- Dynamic page generation with database access and update.
- Consistent web objects.
- The simultaneous execution of multiple transaction types that span a breadth of complexity.
- On-line transaction execution modes.
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships.
- Transaction integrity (ACID properties).
- Contention on data access and update.

The performance metric reported by TPC-W is the number of web interactions processed per second. Multiple web interactions are used to simulate the activity of a retail store, and each interaction is subject to a response time constraint.

TPC-W simulates three different profiles by varying the ratio of browse to buy: primarily shopping (WIPS), browsing (WIPSb) and web-based ordering (WIPSo). The primary metrics are the WIPS rate, the associated price per WIPS ($/WIPS), and the availability date of the priced configuration.

Users of the TPC-W benchmark can browse and order products from the website. In the case of TPC-W the products are books. The expected introduction or Home page is the first page user will see. It includes the company logo, promotional items and navigation options to the top best selling books, a list of new books, search pages, client's shopping cart, and order status pages. User can browse pages containing a list of new or best selling books grouped by subject, or perform searches against all books based upon a title, author or subject. A product page will give you detailed information for the book along with a picture of the book's front cover. User may order books by entering the order web pages. If user is a new customer s/he will have to fill out a customer registration page while for returning customers their information will be retrieved from the database and filled in automatically for them. User can change the quantity of the order or delete a book from the shopping cart. When user wishes to buy, s/he enters credit card information and submits the order. The system will obtain credit card authorization from a Payment Gateway Emulator (PGE), and present the user with an order confirmation page. At a later date user can view the status of his/her last order. Two additional web pages are provided for the system administrator to change book's front cover picture and price. This change is reflected in the new product book list. Essentially, the TPC-W benchmark provides basic functionality required of an Internet e-Commerce website.

TPC-W benchmark specifies the application database schema, the set of 14 web interactions (web pages), and how their execution manipulates application data. It also specifies data consistency and transactional requirements, database scaling and population, and various other aspects, e.g., which web interactions must be covered by SSL. However, TPC-W neither provides sample implementation of the application, nor it prescribes any implementation methodic or limits it to a specific technology. Besides the application structure, TPC-W specifies all aspects of the workload profile, performance metrics of interest, and required structure of the official disclosure report.

2.3 Why Bench4Q?

Benchmarks have been the effective measures to evaluate performances of middleware products. Unlike traditional Web system, e-commerce system generates state-full responses according to sequences of requests. We refer to this class of systems as session-based systems. As a session-based system, e-commerce system is sensitive to QoS guarantees. The workload for these systems must be characterized with respect to sessions. However, the current benchmark specification, such as TPC-W, is not competent to be the QoS oriented e-commerce benchmark. The reasons were as follows:

1) TPC-W can’t simulate Internet customers, which are unlimited, unrelated to each other and sensitive to QoS;

2) TPC-W mainly concerns performance metrics such as hit rate and response times, but incapable of analyzing QoS metrics of E-commerce business, such as session related metrics.

Bench4Q offers a better way to evaluate the QoS guarantees of e-commerce system infrastructures.

3. Bench4Q Tool

3.1 QUICK Introductions

Bench4Q Tool is designed for Bench4Q benchmarking. Bench4Q Tool offers a convenient way to configure the test and analysis the test result.

3.2 Downloading Bench4Q

Bench4Q is available on the Internet at http://forge.ow2.org/projects/jaspte. You can find latest version there.

Bench4Q is distributed as zip file which you should expand using unzip, WinZip (http://www.winzip.com/) or similar.

3.3 Bench4Q Tool License

Bench4Q Tool is distributed according to the GNU Lesser General Public License.

Bench4Q Tool is a free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or any later version.

This source code is distributed "as is" in the hope that it will be useful. It comes with no warranty, and no author or distributor accepts any responsibility for the consequences of its use.

This version is a based on the implementation of TPC-W from University of Wisconsin - Madison. See COPYRIGHT file of "COPYRIGHT TPC-W in Java distribution" for license and copyright details.

This version used some source code of The Grinder. See COPYRIGHT file of "COPYRIGHT The Grinder" for license and copyright details.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

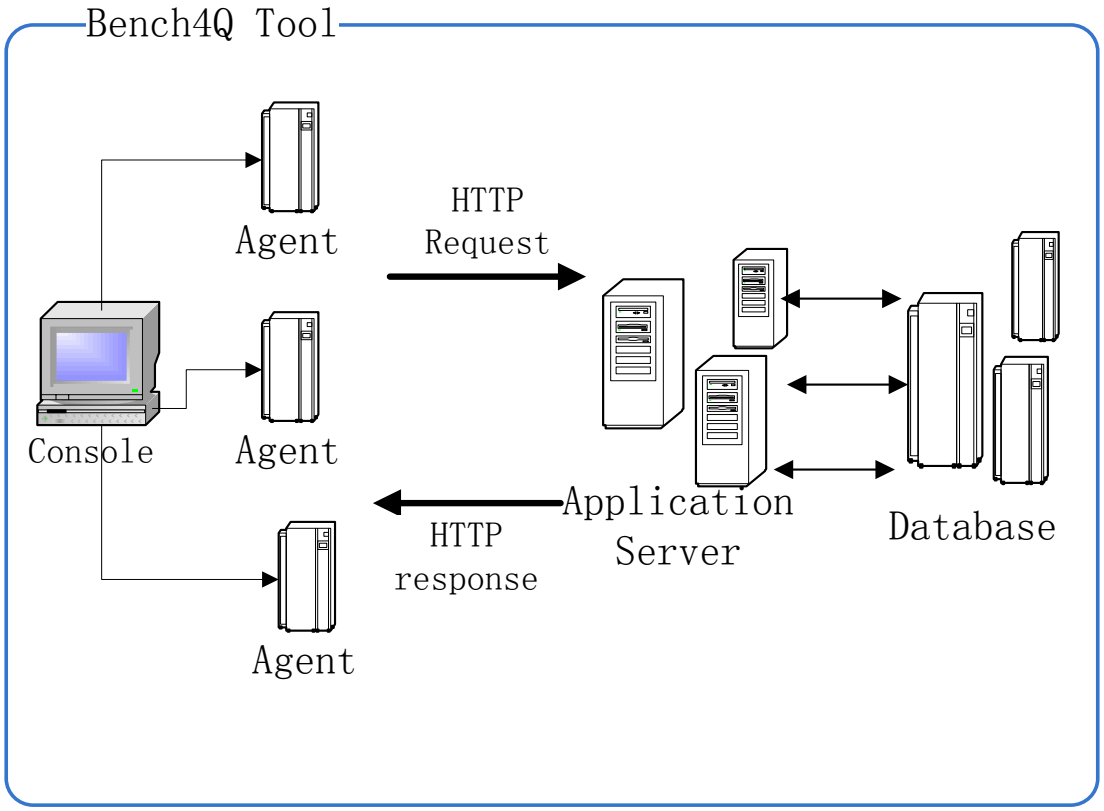
3) Neither the names of the copyright holders nor the names of the contributors may be used to endorse or promote products derived from this

software without specific prior written permission.

Bench4Q Tool includes PicoContainer (http://picocontainer.codehaus.org/). See the file LICENSE-PicoContainer for license and copyright details.

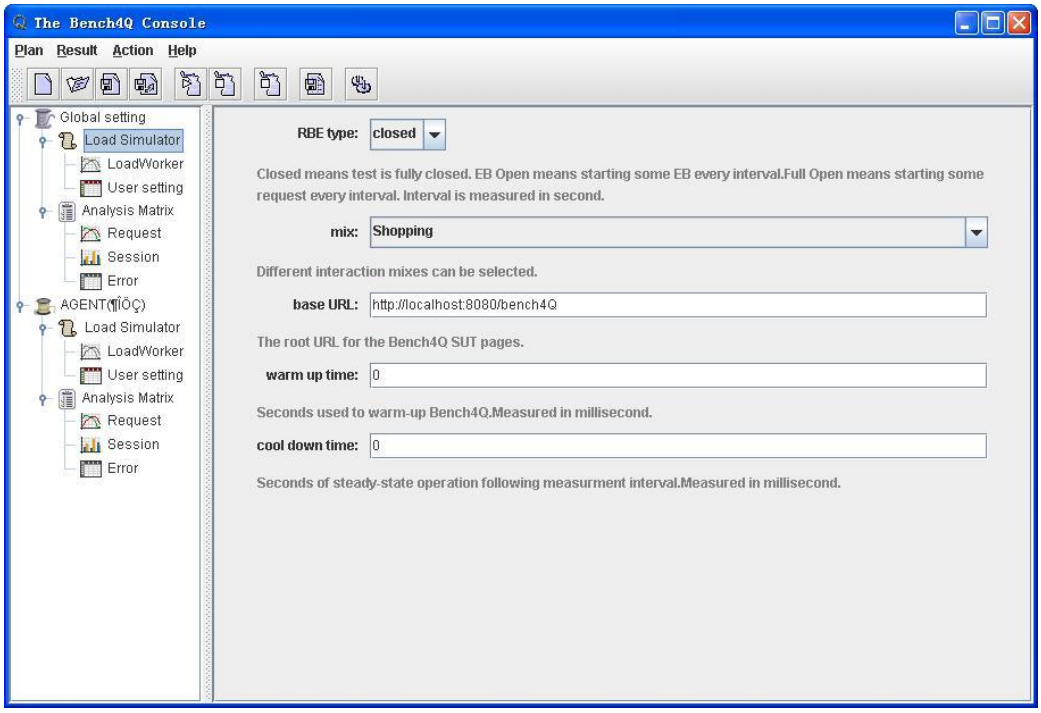
3.4 Bench4Q Tool Design

Bench4Q tool composed of three parts, console, agent and SUT(system under test), showed in the following picture:



3.4.1 Console

The console configures the test, collate and display result. The main frame of console windows is showed in the following picture.



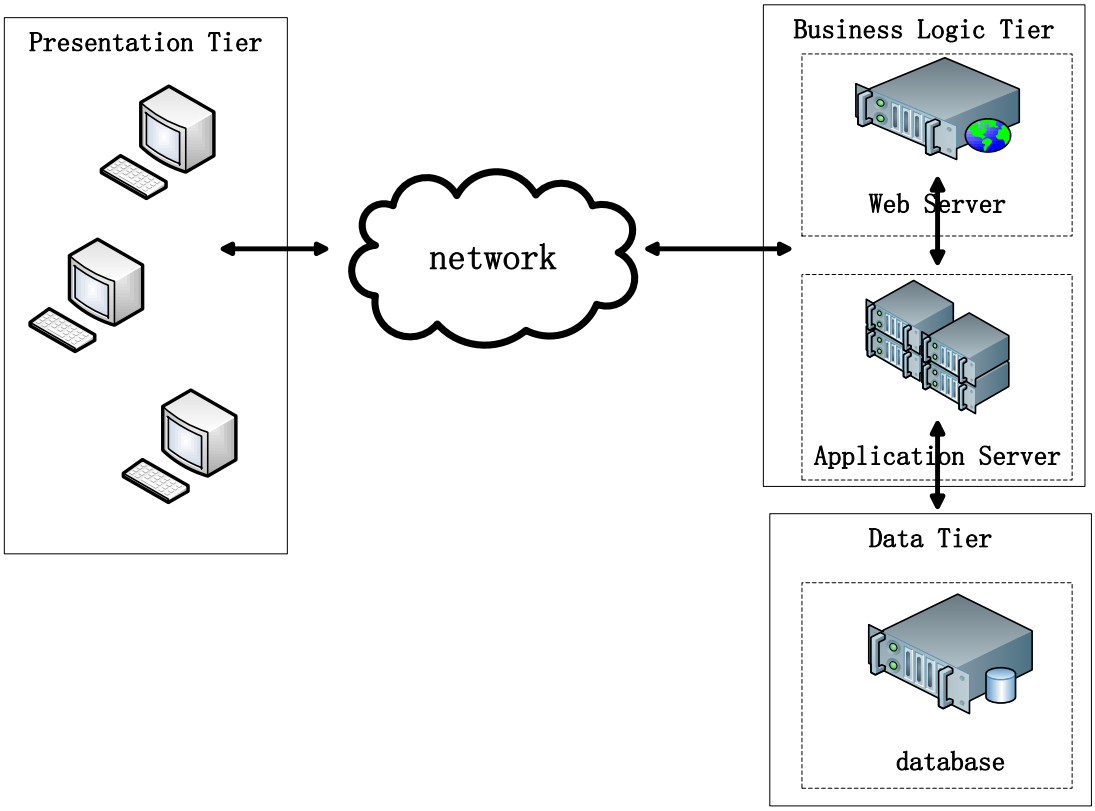
3.4.2 Agent

The agent is the real work part. It fallows the console's instrument, and generates load as the console configured.

For heavy duty testing, you can start an agent on each of several load injector machines. The stats of agent can be controlled using the console. There is little reason to run more than one agent on each load injector, but you can if you wish.

3.4.3 SUT

E-commerce must provide a system, such as e-commerce website, in order to function. Traditional e-commerce system is structured on some infrastructures, such as database, web server and application server. The common architecture shows as following picture：



SUT (System Under Test) contains the business logic to simulate the tested scenario. The SUT comprises all components which are part of the “application” being simulated. This includes network connections, Web Servers, application servers, database servers, etc. The SUT of Bench4Q is based on a distribution of the TPC-W Benchmark Java Implementation originated of PHARM at the University of Wisconsin - Madison.

The SUT of Bench4Q is an online bookstore, which is a typical B2C application. It simulates 14 business operations, 8 of which involving database interactive.

The main features of this version are listed as follows:

1. This version implements TPC-W spec version 1.8.

2. All VARCHAR database rows are limited to size <=255.

3. In the ITEMS table, I_THUMBNAIL and I_IMAGE are varchars, keeping the name of the image file.

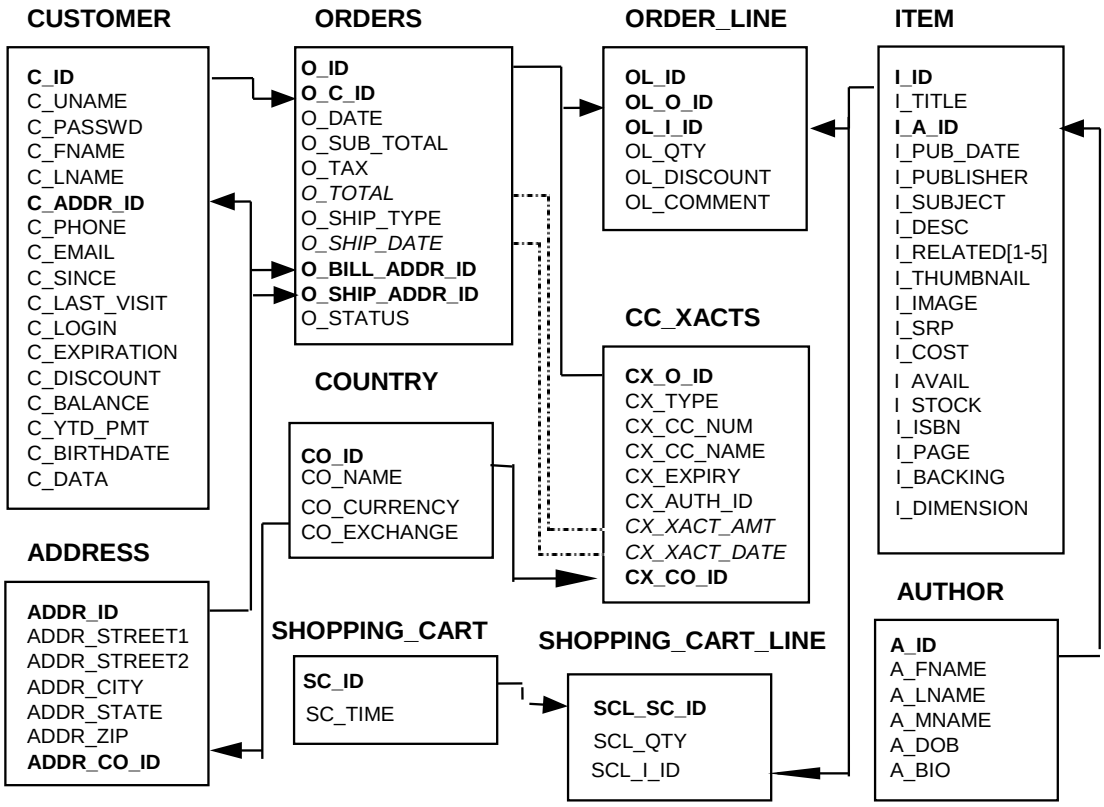
4. There are 1000 item images and thumbnails required.

5. No Payment Getaway Emulator is used (clause 2.7.3.3), so the CC_XACTS credit card processing record is created right away during the order creation.

6. No SSL is used, all connections are insecure.

The components of Bench4Q database are defined to consist of ten separate and

individual base tables. The relationships among these tables are defined in the entity-relationship diagram shown below:

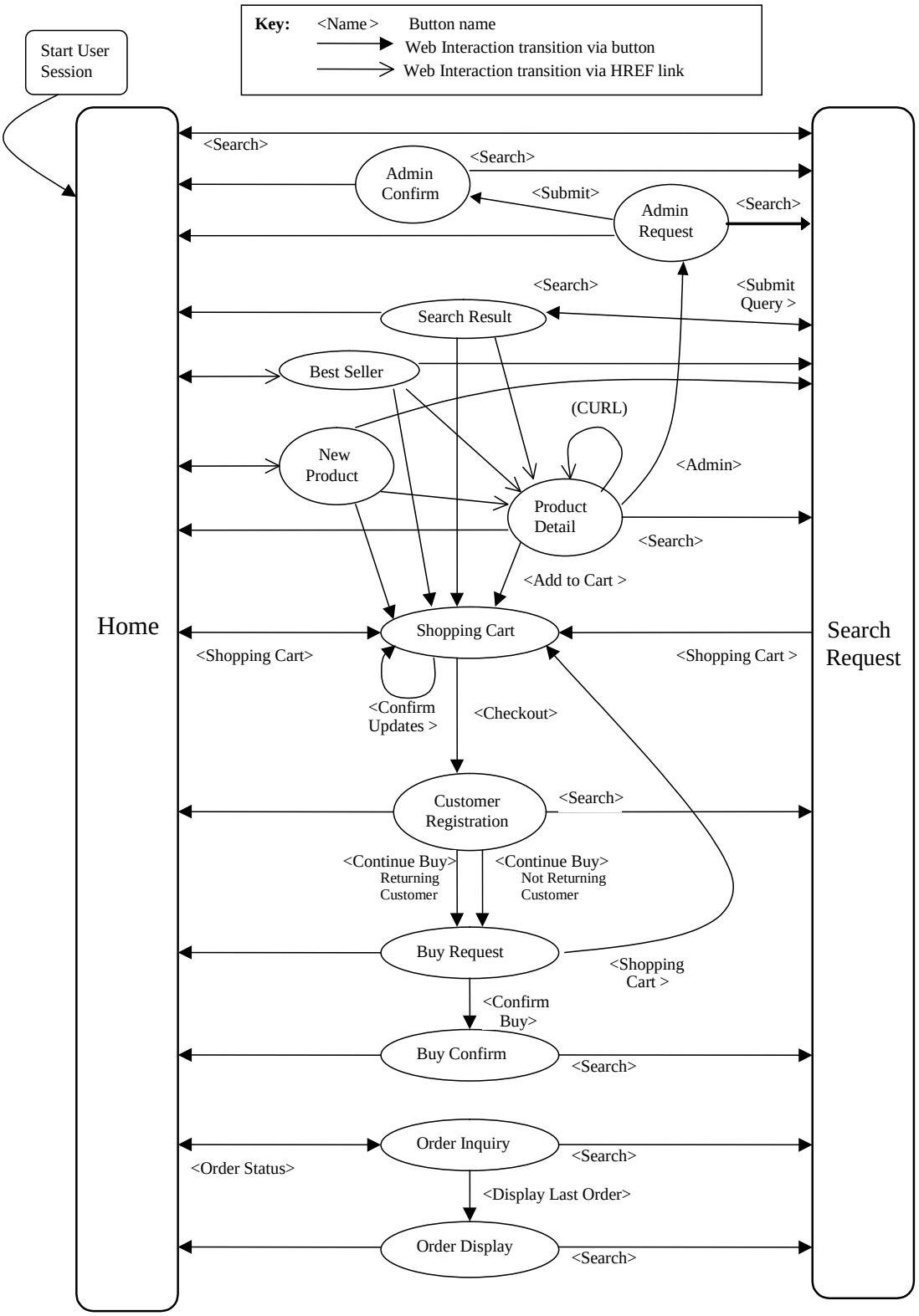


For each table that composes the database, the cardinality of the initial population is specified as follows:

Table Name	Cardinality (in rows)	Typical Row Length (in bytes)	Typical Table Size (in bytes)
CUSTOMER	2880 * (number of EB)	760	2,188,888 k
COUNTRY	92	70	6.44 k
ADDRESS	2 * CUSTOMER	154	887,040 k
ORDERS	.9 * CUSTOMER	220	570,240 k
ORDER_LINE	3 * ORDERS	132	1,026,432 k
AUTHOR	.25 * ITEM	630	1,575 k
CC_XACTS	1 * ORDERS	80	207,360 k
ITEM	1k, 10k, 100k, 1M, 10M	860	8,600 k

The following diagram shows the sequencing of web interactions. During a User Session a sequence of web interactions will be performed. That sequence corresponds to a traversal of this diagram. Each node in the diagram contains the name of a web interaction type (Home, Best Seller, Search Request, etc.). An arrow between two nodes A and B (i.e., A $\rightarrow$ B) indicates that after performing web interaction A, it is possible for an Emulated Browser to next perform the web

interaction B. When there are multiple arrows leaving a node, the arc that is chosen for the next web interaction is determined probabilistically.



4. Getting Started

In order to run a Bench4Q test, you will need to get some basic information of the Bench4Q Tool.

4.1 Requirements

- As Bench4Q tool is written in Java, to run Bench4Q tool, a Java Virtual Machine (JVM) is needed.
- A IBM DB2 and the JDBC driver are needed.
- A JavaEE Application Server is needed.

4.2 Bench4Q Tool Files Structure

Doc	documentation (steps & tips). Also some background documentation.
SUT	The SUT(System Under Test, in this application it is a online book store) web project.
Bench4Q Tool	Bench4Q Tool, it is the load generator.
DB2 Generator	The jar used to generate DB2 data.

4.3 Installation

1) Everything required to run The Bench4Q is in the zip file labelled grinder-version.zip. all you need to do is unpack the package.
2) Then you need to install a IBM DB2 and a JavaEE Application Server.
3) Initiate the database using DB2 Generator. First you need to create a database, and then configure the db.properties file. Then run the DBPopulate.jar file. This process may take several hours.
4) Deploy the SUT on your JavaEE Application Server.The web application looks for datasource using JNDI, so you need to configure database.properties file in WEBAPPROOT\WEB-INF\classes\org\bench4Q\servlet.

4.4 How do I start Bench4Q

It's easy:

1. Create a bench4Q.properties file. This file specifies general control information (how the worker processes should contact the console).
2. Set your CLASSPATH to include the bench4Q.jar file which can be found in the lib

directory.

```
java org.bench4Q.Console
```

3. Start the console on one of the test machines:

```
java org.bench4Q. Agent
```

4. For each test machine, do steps 1 and 2 and start an agent process:
5. The agent will look for the bench4Q.properties file in the local directory. If you like, you can also specify an explicit properties file as the first argument. For example:
6. The console does not read the bench4Q.properties file. It has its own options dialog (choose the help/Options menu option) which you should use to set the

```
java org.bench4Q.Agent propertieFile
```

communication addresses and ports to match those in the bench4Q.properties files.

7. The console process controls can be used to trigger agent to start test. Each agent process then generate load as the console configured.
8. After the test done, you can collect result of agent, then some picture will be showed at the console.

4.5 Console and Agent conmmunication Configuration

AGENT

In bench4Q.properties file, the address of console is configured as follows:

```
bench4Q.consoleHost=133.133.133.123
bench4Q.consolePort=6372
```

The agent will look for the bench4Q.properties file, and connect console using this address.

CONSOLE

You can choose the help/Options menu option to configure the address of console.

5. Bench4Q Tool Usage

After you have deployed all parts of Bench4Q Tool, you can start running tests.

5.1 Test Configurations

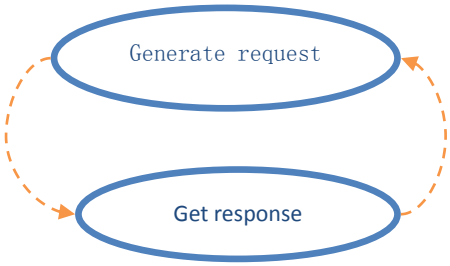
To run a test, you need to know the actual meaning of Bench4Q parameters.

5.1.1 RBE Type

There are two RBE types, closed and open.

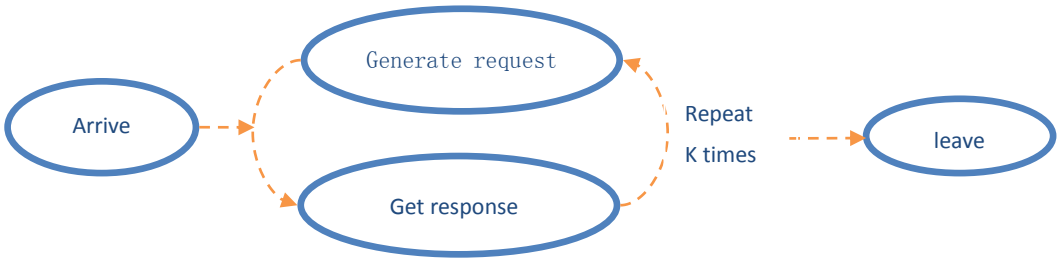
CLOSEED

Fixed number of user sit at the website forever. Each user has the same behavior:



OPEN

Each user visit website, make K request, get response and then leave:



5.1.2 Mix

The percentage of each web interaction executed during each measured interval must following the mix as defined in following table.

Web Interaction	Browsing mix	Shopping mix	Ordering mix
Browse-related	**95.00**	**80.00**	**50.00**
Home	29.00	16.00	9.12
New products	11.00	5.00	0.46
Best sellers	11.00	5.00	0.46
Product detail	21.00	17.00	12.35
Search request	12.00	20.00	14.53
Search result	11.00	17.00	13.08
Order-related	**5.00**	**20.00**	**50.00**
Shopping cart	2.00	11.60	13.53
Registration	0.82	3.00	12.86
Buy request	0.75	2.60	12.73
Buy confirm	0.69	1.20	10.18
Order inquiry	0.30	0.75	0.25
Order display	0.25	0.66	0.22
Admin request	0.10	0.10	0.12
Admin confirm	0.09	0.09	0.11

5.1.3 Warmup and Cooldown

Warm up: Seconds used to warm-up the Agents. The result of this phase of test is not considered. The default warm up time is 0.

Cool down: Seconds used to cool-down the Agents. The result of this phase of test is not considered. The default warm up time is 0.

5.1.4 Load Fluctuation Control

The load fluctuation represents the concurrency feature of business and it has great effects on the choice of tuning policy of a B2C server. Bench4Q simulate such load fluctuation by accumulating the simulated loads from multiple Load Workers. Each Load Worker simulates a controlled load by the following parameters:

- ✓ Base Load: Fixed amount of EBs the Load Worker will be generated in each period during the test.
- ✓ Radom Load: Random amount of EBs will be generated with the bLoad.

- ✓ Rate: The rate of base load changes. It can be positive, negative or zero. It's positive means the base load is increasing every second. It's negative means the base load is decreasing every second. While the rate is zero, the load is fixed.
- ✓ Trigger Time: The time for the Load Worker to start generate its load.
- ✓ Duration: The duration for the Load Worker to generate its load.

By composite the simulated loads from multiple Load Works, the following typical load fluctuations of B2C businesses can be simulated.

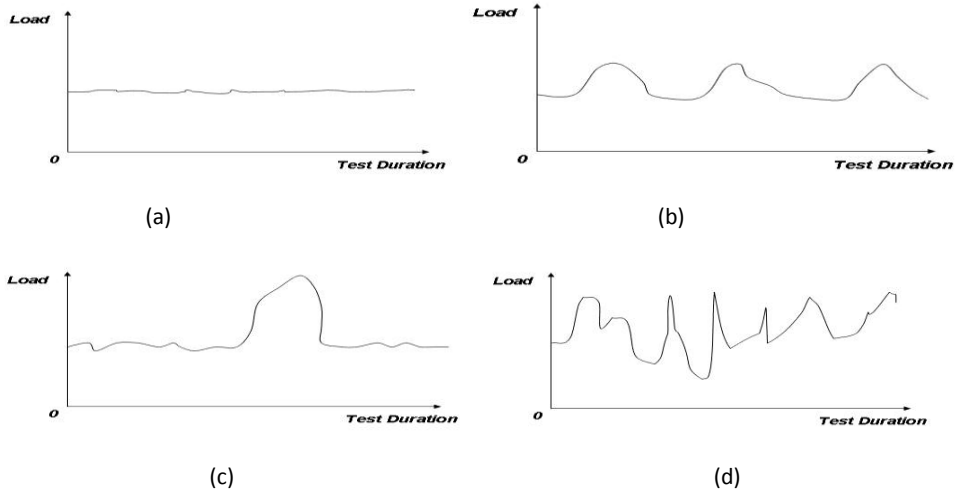


Fig 6. Some typical load fluctuations of B2C

5.1.5 Think Time

The Think Time (TT) is defined by:

TT = T2 - T1

where:

T1 and T2 are measured at the EB;

T1 = time measured after the last byte of the last web interaction is received by the EB from the SUT; and

T2 = time measured before the first byte of the first HTTP request of the next web interaction is sent by the EB to the SUT.

Each Think Time must be taken independently from a negative exponential distribution. Each Think Time, T, must be computed from the following equation:

T = - ln(r) *μ

where:

ln = natural log (base e)

r = random number, with at least 31 bits of precision, from a uniform distribution such that (0 < r 1)

μ = 7 to 8 seconds inclusive.

The parameter is used to increase (>1.0) or decrease (<1.0) think time. While the parameter is set to 0, it means there's no think time.

5.1.6 Tolerance Time

Patience, customer tenacity and importance of transaction affect the behavior of a B2C customer. Correspondingly, we extend the TPC-W by simulating QoS sensitive behavior of B2C customers as follows:

Latency Tolerance (LT) measures the time a customer will wait for a response before change his behavior (for example, give up the ongoing session). Once the request processing time is over its LT, the Load Simulator will discard the current session.

Definition of tenacity for online behavior:

type	Web interaction	tolerance
Unimportant	Home、New Products、Best Sellers、Product Detail、Search Request、Search Results	8 seconds
Important	Shopping Cart、Customer Registration、Buy Request、Buy Confirm、Order Inquiry、Order Display、Admin Request、Admin Confirm	80 seconds

This parameter is used to increase (>1.0) or decrease (<1.0) think time. While the parameter is set to 0, it means there's no tolerance time limitation.

5.2 Metrics Analyzer

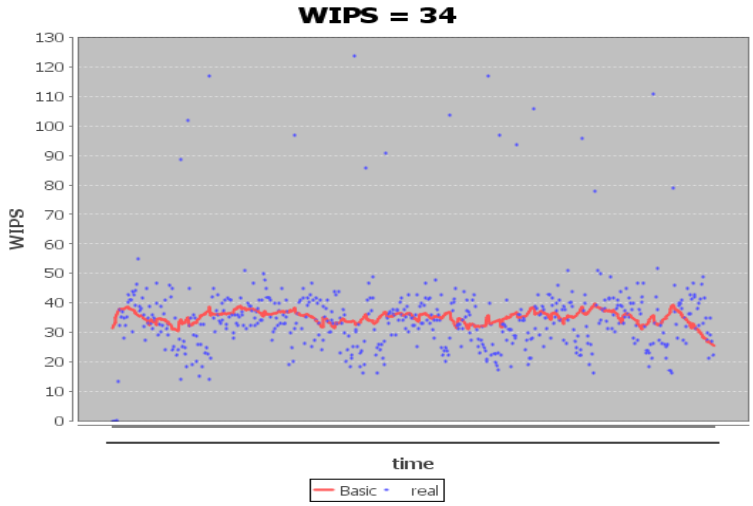
TPC-W mainly focuses on performance metrics, which include hit ratio per second (WIPS) and response time (WIRT). Although these metrics are important to evaluate the performance of middleware, the QoS metrics about the B2C business should be more concerned. The main QoS metrics should include:

- Session based metrics: Compared with performance metrics, a B2C business concerns more that whether sessions, especially profit sessions can be completed. Theses metrics include total sessions, failure sessions, profit sessions, failure profit sessions and the most possible requests leading to failure sessions.
- Customer classes based metrics: QoS differentiation is a popular requirement of a B2C business, which means that an important customer should receive better QoS than a less important customer, especially when the B2C server has been overloaded. So benchmark should provide QoS

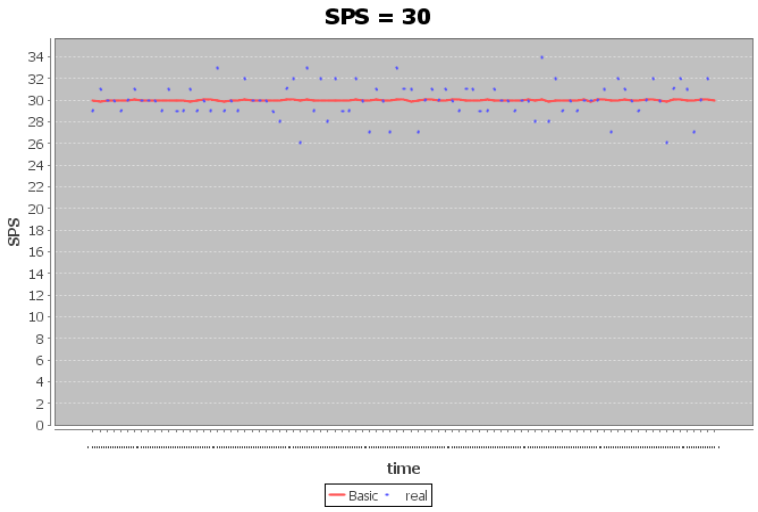
metrics according to the customer classes.

- Profit based metrics: This is the final target of a B2C business [11], so the B2C benchmark should provide the profit based metrics, such as the total profits gained and the profits lost.

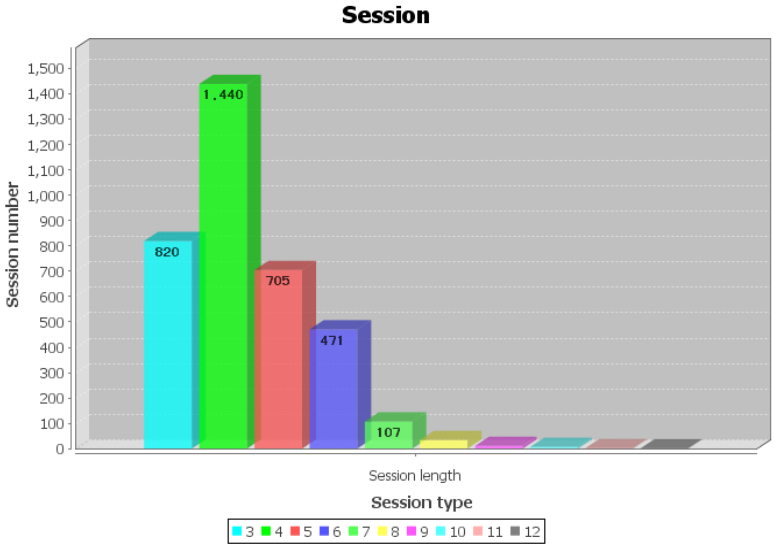
All the benchmarking result can be showed in picture format, showed as follows:



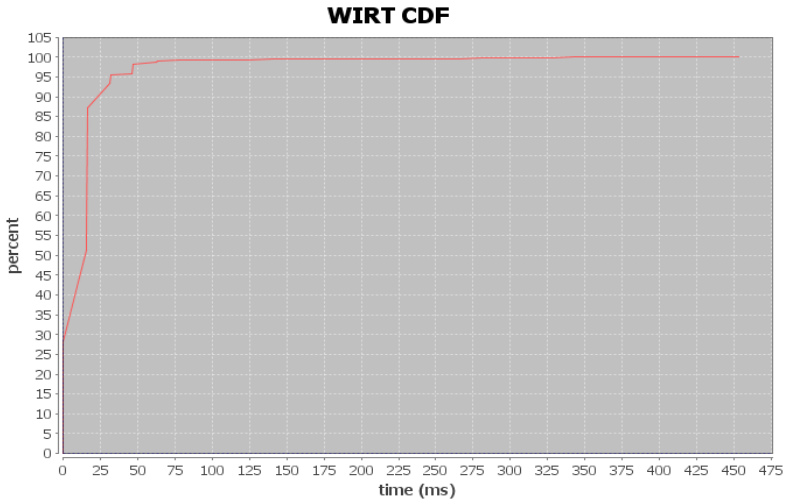
WIPS(web interactions processed per second): WIPS are computed as the total number of Web interactions requested and completed successfully within a measurement interval divided by the length of that interval in seconds.



SPS(session completed per second):SPS are computed as the total successful session within a measurement interval divided by the length of that interval in seconds.



Session length: session length shows the session number divided by the session length.



WIRT(Web Interaction Response Time): WIRT showed in CDF. From the CDF diagram, user can get the basic Web Interaction Response Time situation.

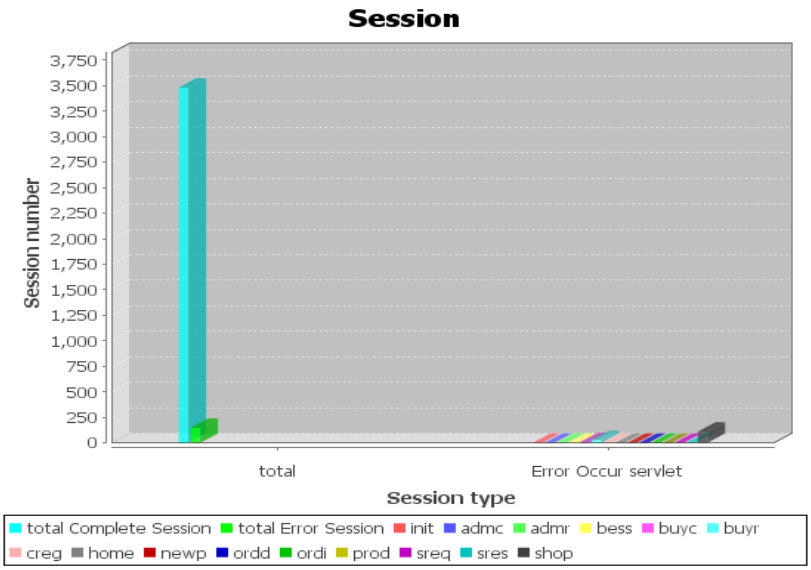
The Web Interaction Response Time (WIRT) is defined by:

WIRT = T2 - T1

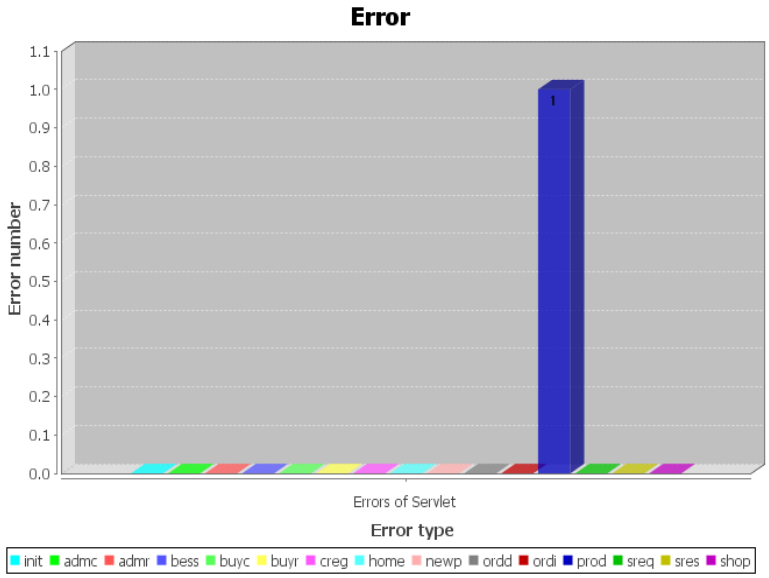
where: T1 and T2 are measured at the EB;

T1 = time measured before the first byte of the first HTTP request of the web interaction is sent by the EB to the SUT;

T2 = time measured after the last byte of the last HTTP response that completes the web interaction is received by the EB from the SUT.



Session: this diagram shows the basic information of session, including total successful session, error session, benefit session, etc.



ERROR: this diagram shows the error happened during the measurement interval. The diagram catalogs the errors by web interaction.

6. Getting Involved

Any input or personal view for improving and/or developing Bench4Q is welcome. Bench4Q, as an open source project, also welcomes external contributions.

APPENDIX

The schema of Bench4Q Tool's parameters:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="bench4Q">
        <xs:annotation>
            <xs:documentation></xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="testName" type="xs:string" />
                <xs:element name="testDescription" type="xs:string" />
                <xs:element name="rbe" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="interval"
                                type="xs:double" />
                            <xs:element name="prepair"
                                type="xs:integer" />
                            <xs:element name="cooldown"
                                type="xs:integer" />
                            <xs:element name="out" type="xs:string" />
                            <xs:element name="tolerance"
                                type="xs:double" />
                            <xs:element name="retry" type="xs:integer" />
                            <xs:element name="thinktime"
                                type="xs:double" />
                            <xs:element name="mix">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration
                                            value="shopping" />
                                        <xs:enumeration
                                            value="ordering" />
                                        <xs:enumeration
                                            value="browsing" />
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="slow" type="xs:double"
                                minOccurs="0" />
                            <xs:element name="getImage"
```

```
                                type="xs:boolean">
                            </xs:element>
                            <xs:element name="baseURL" type="xs:string" />
                            <xs:element name="ebs"
                                maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="baseLoad"
                                            type="xs:integer" />
                                        <xs:element name="randomLoad"
                                            type="xs:integer" />
                                        <xs:element name="rate"
                                            type="xs:integer" />
                                        <xs:element name="triggerTime"
                                            type="xs:integer" />
                                        <xs:element name="stdyTime"
                                            type="xs:integer" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="rbetype">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="open" />
                                    <xs:enumeration value="closed" />
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```