

Enhacement of JMeter

B. Sushmitha
KIET

Dhruv Joshi
NIT Rourkela

Manisha Choudhury
NIT Rourkela

Naman Choudhary
NIT Jamshedpur

Shekhar Saurav
NIT Jamshedpur

Surabhi Mour
SVNIT Surat

03 July 2013



Apache
JMeterTM

Our Guides

Principle Investigator

Prof. D. B. Phatak

Project In-Charge

Mr. Nagesh Karmali

Mentors

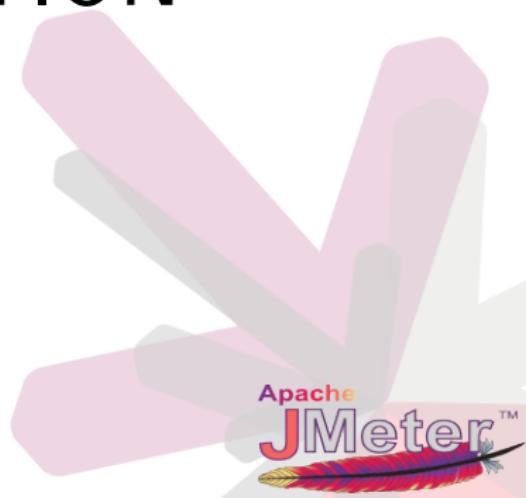
Miss. Silpa T.

Miss Firuza Aibara

Mr. Sukhdeo Gupta

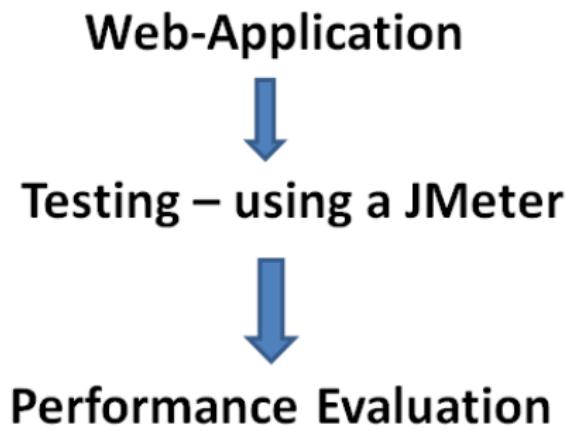


INTRODUCTION



Why do we use JMeter?

Considering a typical web-application, we could have 1000s of users trying to access it concurrently. We may not choose to employ 1000 testers for every such user for performance evaluation of the application.



More on Testing

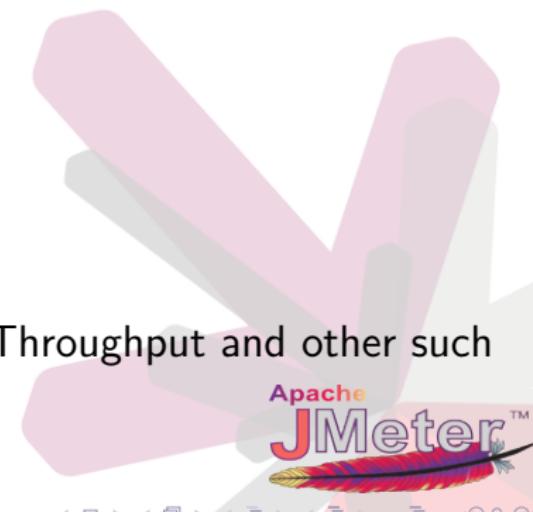
Testing Parameters:

- User
- Data
- Time

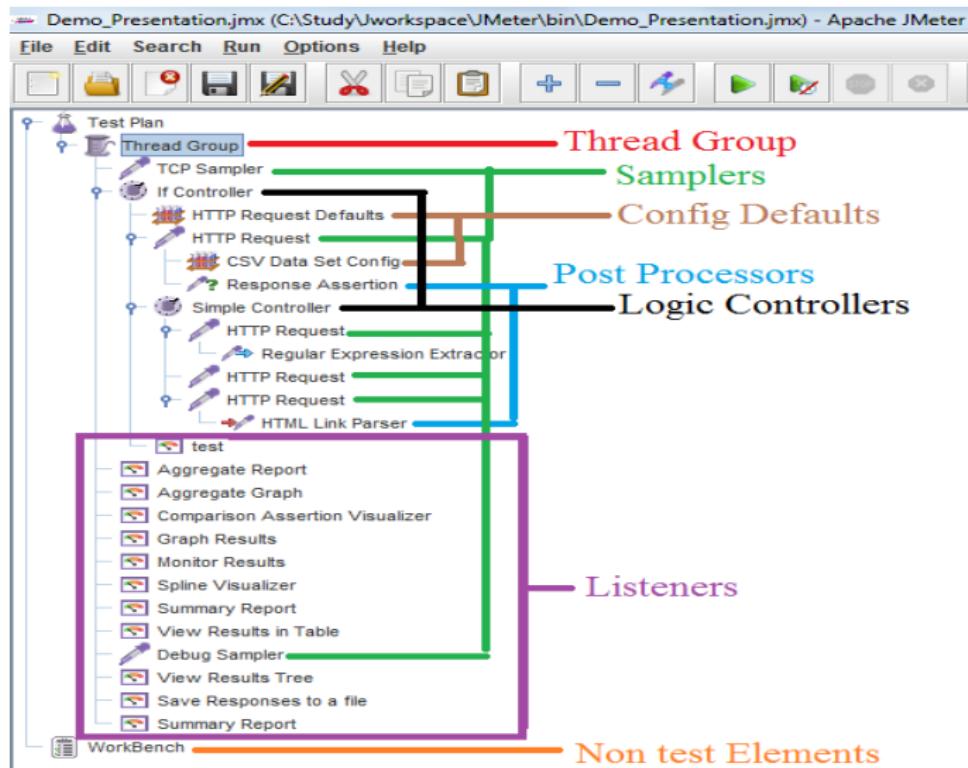
Performance testing Comprises of:

- ① Load Testing
- ② Stress Testing
- ③ Scalability Testing

and calculating Response Time, Latency, Throughput and other such Metrics.



A sample JMeter 'TESTPLAN'



Some JMeter Plugins

- **Thread group Plugins**

- Stepping Thread Group
- Ultimate Thread Group

- **Timeline Graph Plugins**

- Active Threads Over Time
- Response Times Over Time
- Response Latency Over Time
- Transactions per Second
- Server Hits per Seconds
- Bytes Throughput Over Time



JMeter Enhancements Implemented

- **Dynamic Bandwidth Throttling**

for the requests being sent, based on response error percentage



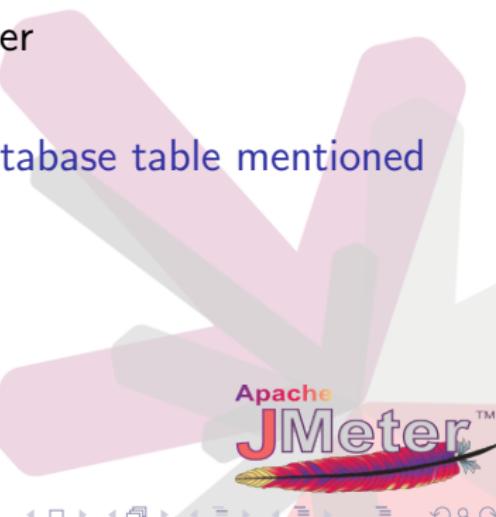
JMeter Enhancements Implemented

- **Dynamic Bandwidth Throttling**
for the requests being sent, based on response error percentage
- **IP Spoofing**
distinct IP addresses for each virtual user



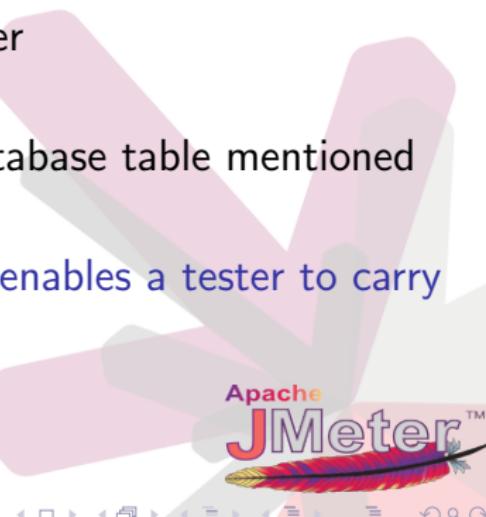
JMeter Enhancements Implemented

- **Dynamic Bandwidth Throttling**
for the requests being sent, based on response error percentage
- **IP Spoofing**
distinct IP addresses for each virtual user
- **Auto CSV Generation**
creating A .csv file directly from the database table mentioned



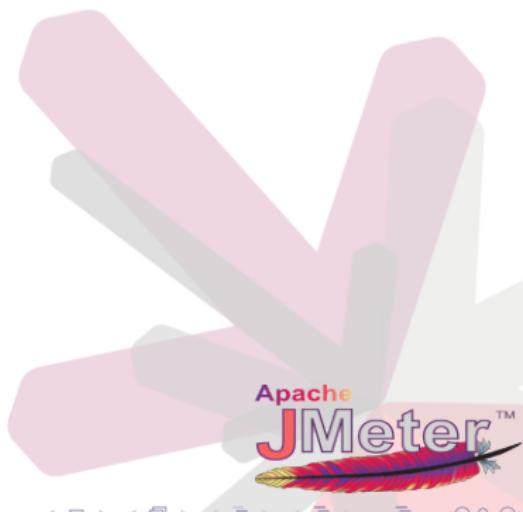
JMeter Enhancements Implemented

- **Dynamic Bandwidth Throttling**
for the requests being sent, based on response error percentage
- **IP Spoofing**
distinct IP addresses for each virtual user
- **Auto CSV Generation**
creating A .csv file directly from the database table mentioned
- **Automating TPC-C Testing**
Test script for Oracle and MySQL that enables a tester to carry out preliminary TPCC testing



- **Filtered Results Table**

Filters the sampler results, based on user specified parameters



- **Filtered Results Table**

Filters the sampler results, based on user specified parameters

- **Constant Increasing Timer**

Stepping Up time interval between Samples requested



- **Filtered Results Table**

Filters the sampler results, based on user specified parameters

- **Constant Increasing Timer**

Stepping Up time interval between Samples requested

- **Enhanced Assertion results**

Details of the Sampler passing or failing the test



- **Filtered Results Table**

Filters the sampler results, based on user specified parameters

- **Constant Increasing Timer**

Stepping Up time interval between Samples requested

- **Enhanced Assertion results**

Details of the Sampler passing or failing the test

- **SMTP Defaults**

A configuration element for setting data for SMTP Samplers under it

AutoCSV Generation



Application to be tested

The screenshot shows a Firefox browser window with the URL `localhost:8090/TestAMS/` highlighted by a red oval. The main content area displays a large image of a white airplane on a runway against a blue sky. Below the image, a pink banner contains the text "Airport Management System". A login form is visible, featuring fields for "Username" and "Password", and a "Login" button. The browser's toolbar at the bottom includes icons for various applications like Internet Explorer, Google Chrome, and Mozilla Firefox.

The Auto CSV Generation GUI

AutoCSV_test1.jmx (C:\Study\Uworkspace\JMeter\bin\AutoCSV_test1.jmx) - Apache JMeter (2.9)

File Edit Search Run Options Help

Test Plan

- Auto CSV Generation

Auto CSV Generation

Name: Auto CSV Generation

Comments:

Variable Name Bound to Pool

Variable Name: bbb

Connection Pool Configuration

Max Number of Connections: 10

Pool Timeout: 10000

Idle Cleanup Interval (ms): 60000

Connection Validation by Pool

Keep-Alive: True

Max Connection age (ms): 5000

Validation Query: Select 1

Database Connection Configuration

Database URL (without Database name): jdbc:mysql://localhost:3306/

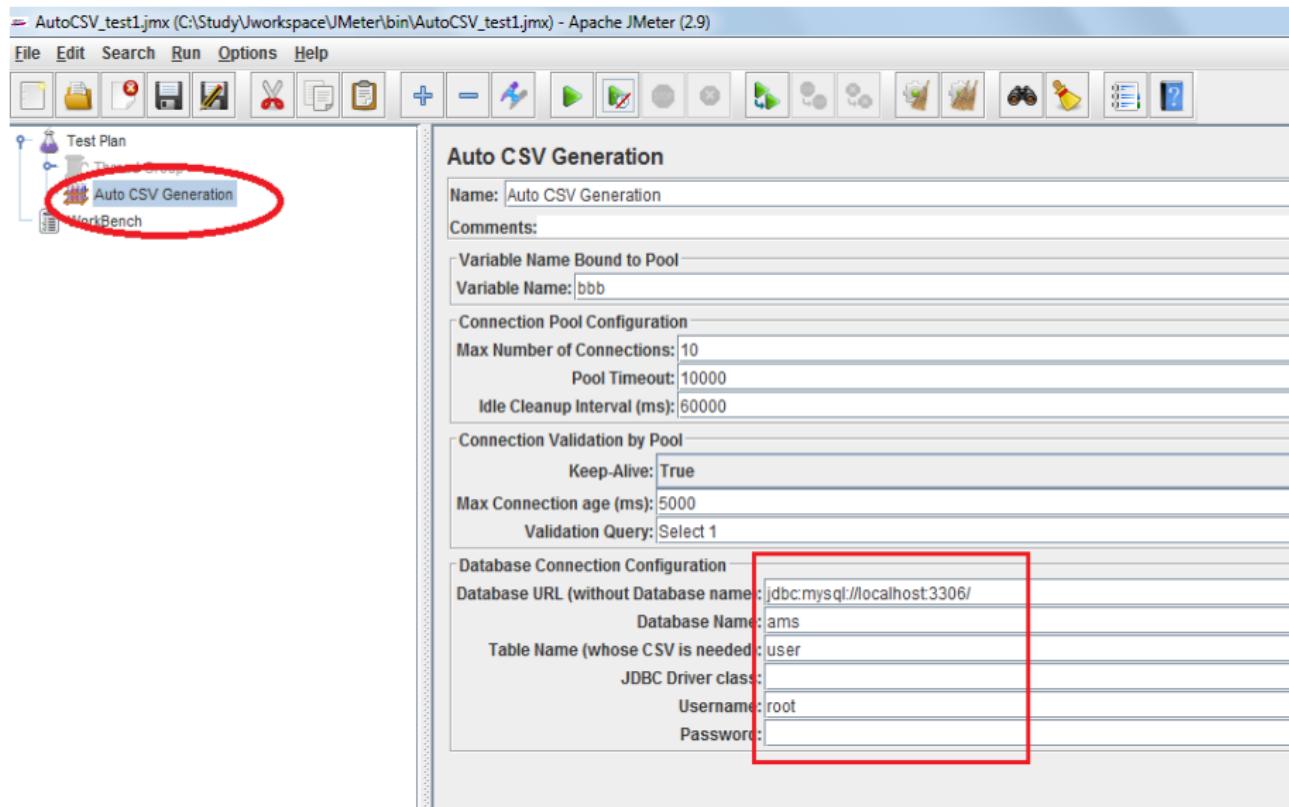
Database Name: ams

Table Name (whose CSV is needed): user

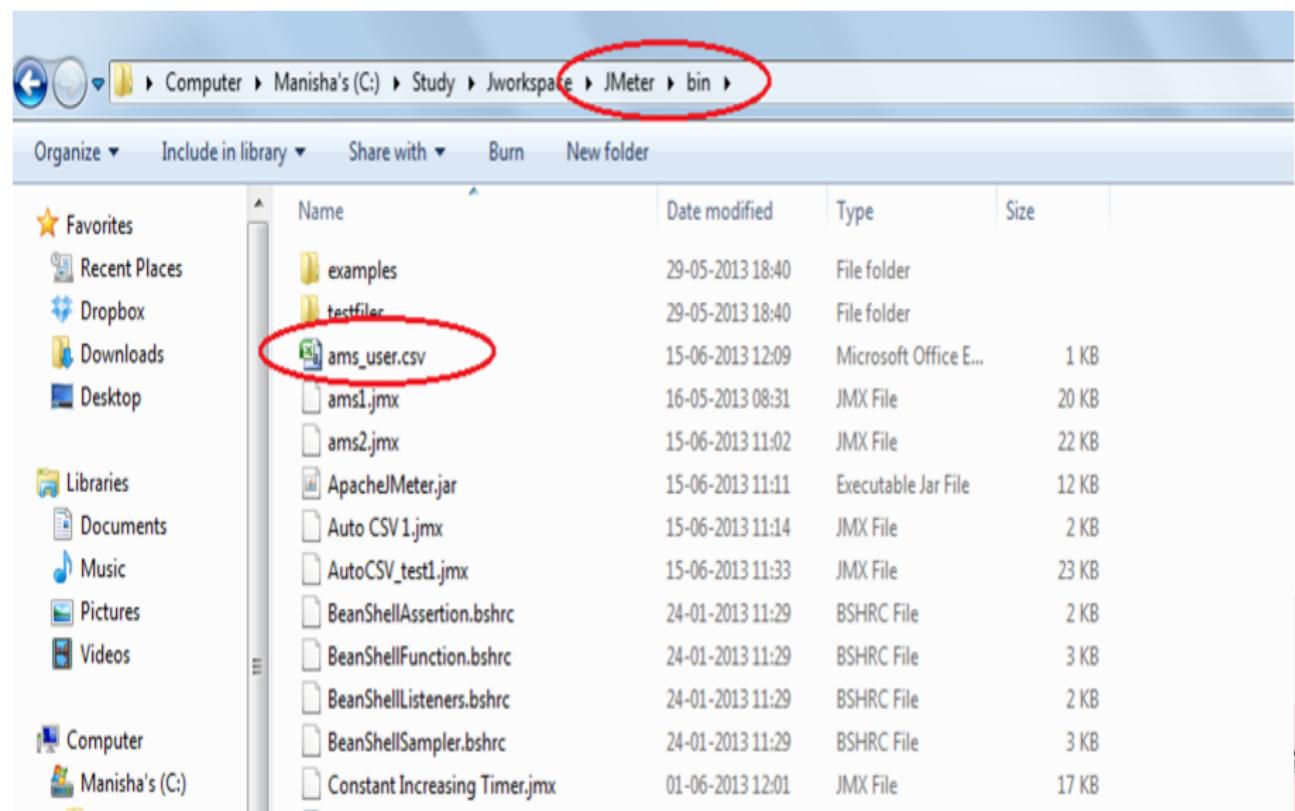
JDBC Driver class:

Username: root

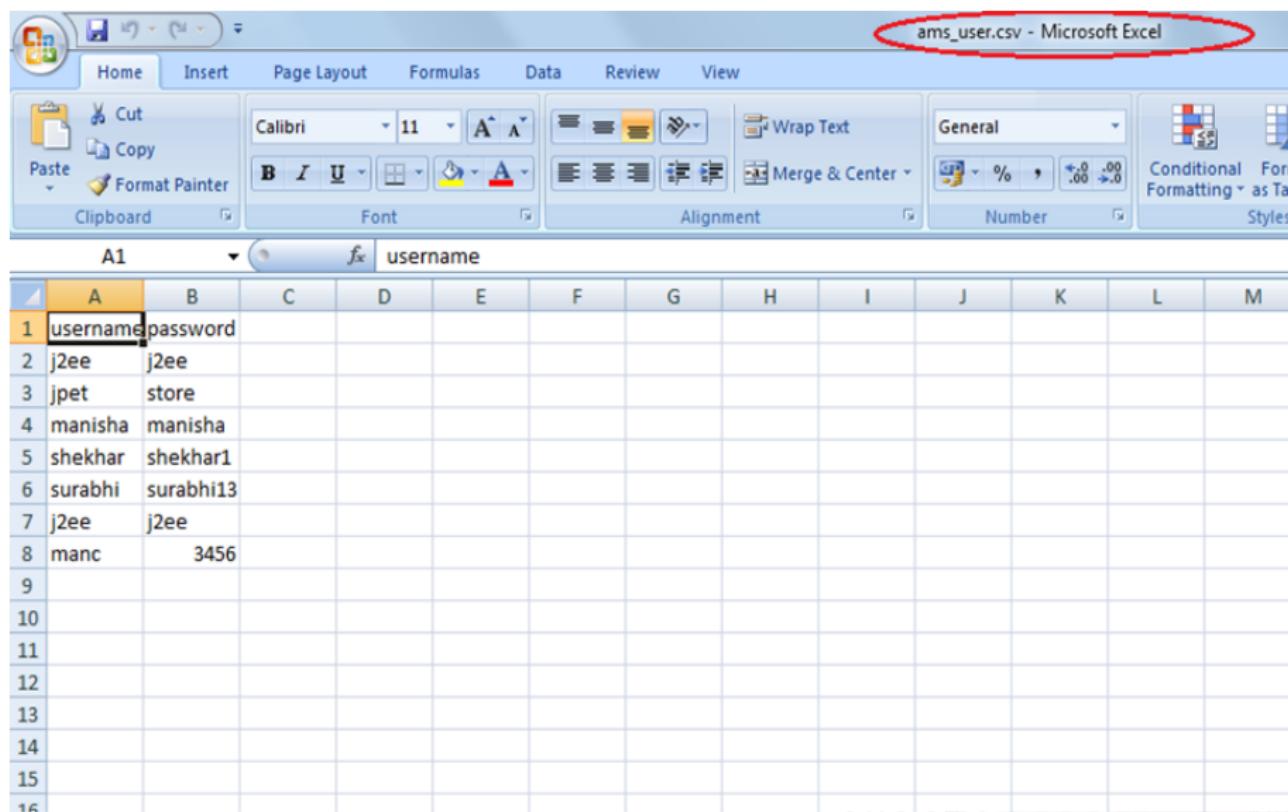
Password:



The .csv file generated in /bin folder



The ams_user.csv generated



The screenshot shows a Microsoft Excel spreadsheet titled "ams_user.csv - Microsoft Excel". The title bar is circled in red. The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, and View. The Home tab is selected, displaying various toolbar icons for clipboard operations (Cut, Copy, Paste), font styles (Calibri, 11pt, bold, italic, underline), alignment (Wrap Text, Merge & Center), number formats (General, %, .00, .00), conditional formatting, and styles.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	username	password											
2	j2ee	j2ee											
3	jpet	store											
4	manisha	manisha											
5	shekhar	shekhar1											
6	surabhi	surabhi13											
7	j2ee	j2ee											
8	manc	3456											
9													
10													
11													
12													
13													
14													
15													
16													



Test Plan

Thread Group

HTTP Sample 1

HTTP Sample 2

CSV Data Set Config

View Results Tree

Auto CSV Generation

WorkBench

CSV Data Set Config

Name: CSV Data Set Config

Comments:

Configure the CSV Data Source

Filename: ams_user.csv

File encoding:

Variable Names (comma-delimited): username,password

Delimiter (use 't' for tab):

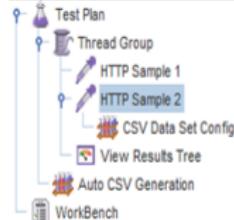
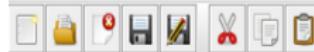
Allow quoted data?: False

Recycle on EOF ?: True

Stop thread on EOF ?: False

Sharing mode: All threads

The CSV data config element is added as child of HTTP Sampler



HTTP Request

Name: HTTP Sample 2
Comments:
Web Server
Server Name or IP: localhost Port Number: 8090
HTTP Request
Implementation: [dropdown] Protocol [http]: http Method: POST Content encoding: [dropdown]
Path: /TestAMS/validateLogin
 Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data for POST Browser-compatible headers
Parameters Post Body
Send Parameters With the Request:

Name:	Value
username	\${username}
pass	\${password}

Send Files With the Request:
File Path:

The HTTP sampler where the parameters of .csv file are set



Output - CSV data verification

AutoCSV_test1.jmx (C:\Study\workspace\JMeter\bin\AutoCSV_test1.jmx) - Apache JMeter (2.9)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Sample 1 HTTP Sample 2 CSV Data Set Config View Results Tree Auto CSV Generation WorkBench

View Results Tree

Name: View Results Tree
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Success

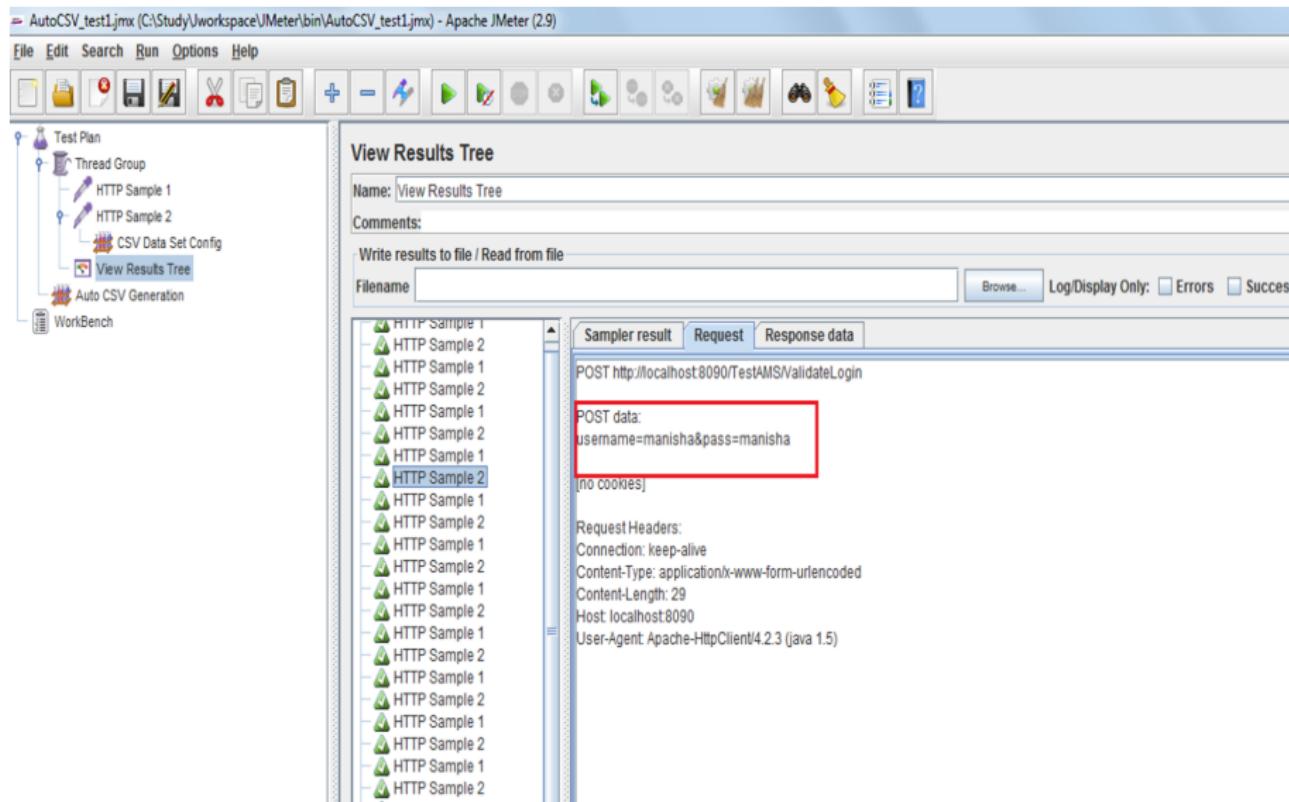
Sampler result Request Response data

POST http://localhost:8090/TestAMS/ValidateLogin

POST data:
username=manisha&pass=manisha

[no cookies]

Request Headers:
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Host: localhost:8090
User-Agent: Apache-HttpClient/4.2.3 (java 1.5)



Bandwidth Throttling



Bandwidth Throttling

- Bandwidth throttling is the intentional slowing of Internet service.



Bandwidth Throttling

- Bandwidth throttling is the intentional slowing of Internet service.
- In real world scenario, people use different web services at different bandwidths.



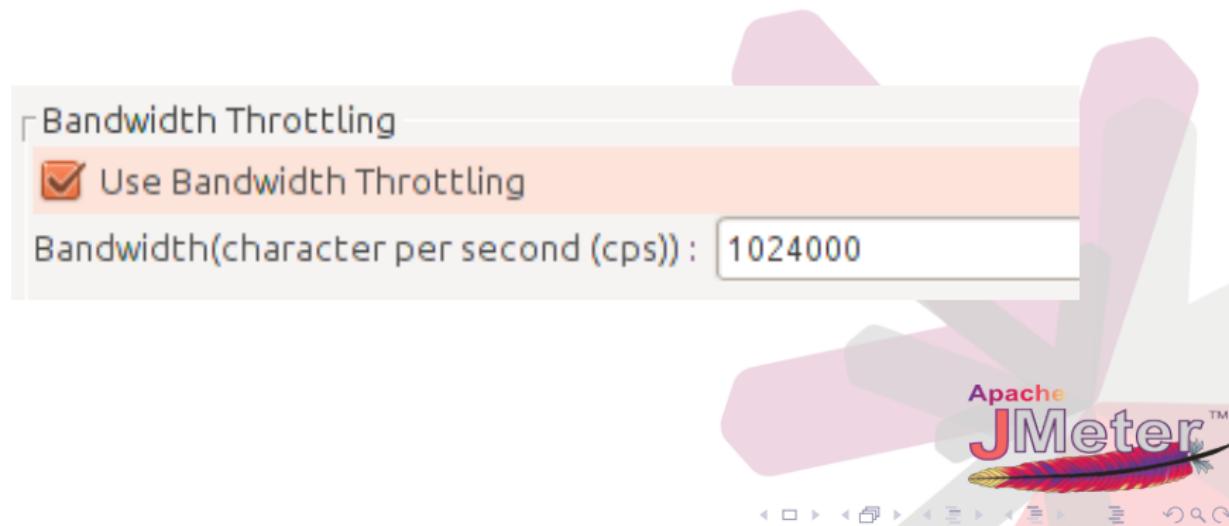
Bandwidth Throttling

- Bandwidth throttling is the intentional slowing of Internet service.
- In real world scenario, people use different web services at different bandwidths.
- Using Bandwidth Throttling, JMeter can be used to create test plans to simulate slower connections.



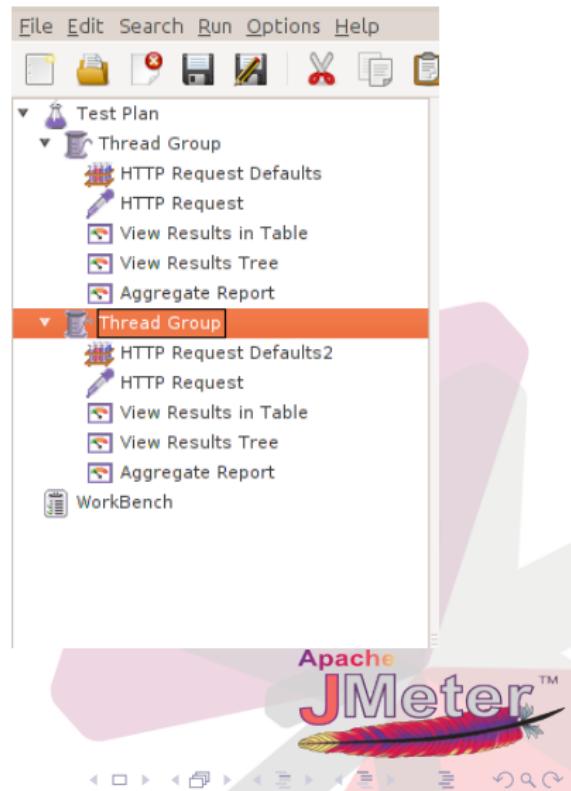
User Interface

- To use bandwidth throttling in JMeter, a gui component has been added to HTTP Request Defaults Config Element.



Test Plan

- Thread groups : 2
- Thread count : 5
- Loop count : 5
- HTTP Samplers : 1



HTTP Default Settings for Samplers

HTTP Request Defaults

Name:

Comments:

Web Server

Server Name or IP:

HTTP Request

Implementation: Protocol [http]:

Path:

Parameters

Name:

Detail

Proxy Server

Server Name or IP:

Optional Tasks

Retrieve All Embedded Resources from HTML Files Use conc

Embedded URLs must match:

Bandwidth Throttling

Use Bandwidth Throttling

Bandwidth(character per second (cps)):

HTTP Request Defaults

Name:

Comments:

Web Server

Server Name or IP:

HTTP Request

Implementation: Protocol [http]:

Path:

Parameters

Name:

Detail

Proxy Server

Server Name or IP:

Optional Tasks

Retrieve All Embedded Resources from HTML Files Use conc

Embedded URLs must match:

Bandwidth Throttling

Use Bandwidth Throttling

Bandwidth(character per second (cps)):

Result Table for Thread group 1

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	15:03:36.961	Thread Group 1-4	HTTP Request	3588	▲	14164	3347
2	15:03:36.961	Thread Group 1-1	HTTP Request	3600	▲	14164	3359
3	15:03:36.976	Thread Group 1-5	HTTP Request	3588	▲	14164	3347
4	15:03:36.959	Thread Group 1-2	HTTP Request	3614	▲	14164	3371
5	15:03:36.972	Thread Group 1-3	HTTP Request	3662	▲	14164	3340
6	15:03:40.594	Thread Group 1-2	HTTP Request	1035	▲	14164	794
7	15:03:40.571	Thread Group 1-1	HTTP Request	1064	▲	14164	823
8	15:03:40.636	Thread Group 1-3	HTTP Request	1008	▲	14164	767
9	15:03:40.572	Thread Group 1-5	HTTP Request	1075	▲	14164	834
10	15:03:40.553	Thread Group 1-4	HTTP Request	1747	▲	14164	1506
11	15:03:41.632	Thread Group 1-2	HTTP Request	751	▲	14164	510
12	15:03:41.680	Thread Group 1-3	HTTP Request	1049	▲	14164	808
13	15:03:41.679	Thread Group 1-1	HTTP Request	1057	▲	14164	816
14	15:03:41.682	Thread Group 1-5	HTTP Request	1060	▲	14164	819
15	15:03:42.303	Thread Group 1-4	HTTP Request	747	▲	14164	506
16	15:03:42.387	Thread Group 1-2	HTTP Request	745	▲	14164	504
17	15:03:42.733	Thread Group 1-3	HTTP Request	881	▲	14164	525
18	15:03:42.803	Thread Group 1-1	HTTP Request	862	▲	14164	541
19	15:03:42.803	Thread Group 1-5	HTTP Request	904	▲	14164	662
20	15:03:43.135	Thread Group 1-2	HTTP Request	845	▲	14164	604
21	15:03:43.054	Thread Group 1-4	HTTP Request	946	▲	14164	705
22	15:03:43.618	Thread Group 1-3	HTTP Request	875	▲	14164	633
23	15:03:43.724	Thread Group 1-5	HTTP Request	780	▲	14164	539
24	15:03:43.722	Thread Group 1-1	HTTP Request	810	▲	14164	569
25	15:03:44.119	Thread Group 1-4	HTTP Request	1059	▲	14164	818

Result Table for Thread group 2

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	15:03:36.983	Thread Group 2-1	HTTP2 Request	32450	green	14164	16449
2	15:03:36.996	Thread Group 2-2	HTTP2 Request	32438	green	14164	16437
3	15:03:37.007	Thread Group 2-4	HTTP2 Request	32428	green	14164	16428
4	15:03:37.008	Thread Group 2-5	HTTP2 Request	32428	green	14164	16427
5	15:03:37.002	Thread Group 2-3	HTTP2 Request	33412	green	14164	17411
6	15:04:09.437	Thread Group 2-1	HTTP2 Request	32400	green	14164	16399
7	15:04:09.439	Thread Group 2-2	HTTP2 Request	32398	green	14164	16397
8	15:04:09.441	Thread Group 2-4	HTTP2 Request	32398	green	14164	16397
9	15:04:09.442	Thread Group 2-5	HTTP2 Request	32398	green	14164	16397
10	15:04:10.418	Thread Group 2-3	HTTP2 Request	32397	green	14164	16396
11	15:04:41.849	Thread Group 2-5	HTTP2 Request	32522	green	14164	16521
12	15:04:41.841	Thread Group 2-1	HTTP2 Request	32530	green	14164	16529
13	15:04:41.842	Thread Group 2-2	HTTP2 Request	32530	green	14164	16529
14	15:04:41.847	Thread Group 2-4	HTTP2 Request	32525	green	14164	16524
15	15:04:42.818	Thread Group 2-3	HTTP2 Request	33397	green	14164	17396
16	15:05:14.374	Thread Group 2-5	HTTP2 Request	32407	green	14164	16406
17	15:05:14.375	Thread Group 2-1	HTTP2 Request	32406	green	14164	16405
18	15:05:14.380	Thread Group 2-4	HTTP2 Request	32401	green	14164	16400
19	15:05:14.379	Thread Group 2-2	HTTP2 Request	32402	green	14164	16401
20	15:05:16.219	Thread Group 2-3	HTTP2 Request	32426	green	14164	16425
21	15:05:46.784	Thread Group 2-1	HTTP2 Request	32412	green	14164	16412
22	15:05:46.785	Thread Group 2-2	HTTP2 Request	32416	green	14164	16415
23	15:05:46.783	Thread Group 2-5	HTTP2 Request	32425	green	14164	16425
24	15:05:46.784	Thread Group 2-4	HTTP2 Request	32428	green	14164	16427
25	15:05:48.650	Thread Group 2-3	HTTP2 Request	32401	green	14164	16400

Scroll automatically? Child samples? No of Samples 25 Latest Sample 32401 Average 32511 Deviation 267

Results Comparison

Bandwidth: 1MBps

Status	Bytes	Latency
Green	14164	3347
Green	14164	3359
Green	14164	3347
Green	14164	3371
Green	14164	3340
Green	14164	794
Green	14164	823
Green	14164	767
Green	14164	834
Green	14164	1506
Green	14164	510
Green	14164	808
Green	14164	816
Green	14164	819
Green	14164	506
Green	14164	504
Green	14164	525
Green	14164	541
Green	14164	662
Green	14164	604
Green	14164	705
Green	14164	633
Green	14164	539
Green	14164	569

Bandwidth: 1KBps

Status	Bytes	Latency
Green	14164	16449
Green	14164	16437
Green	14164	16428
Green	14164	16427
Green	14164	17411
Green	14164	16399
Green	14164	16397
Green	14164	16397
Green	14164	16396
Green	14164	16521
Green	14164	16529
Green	14164	16529
Green	14164	16524
Green	14164	17396
Green	14164	16406
Green	14164	16405
Green	14164	16400
Green	14164	16401
Green	14164	16425
Green	14164	16412
Green	14164	16415
Green	14164	16425
Green	14164	16427
Green	14164	16400

Aggregate Reports

Bandwidth: 1MBps

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP Request	25	1494	1035	3588	745	3662	0.00%	3.0/sec	42.1
TOTAL	25	1494	1035	3588	745	3662	0.00%	3.0/sec	42.1

Bandwidth: 1Kbps

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP2 Requ...	25	32511	32425	32530	32397	33412	0.00%	9.1/min	2.1
TOTAL	25	32511	32425	32530	32397	33412	0.00%	9.1/min	2.1



Dynamic Bandwidth Throttling



Dynamic Bandwidth Throttling

- ① DBT deals with the variation of bandwidth at runtime.



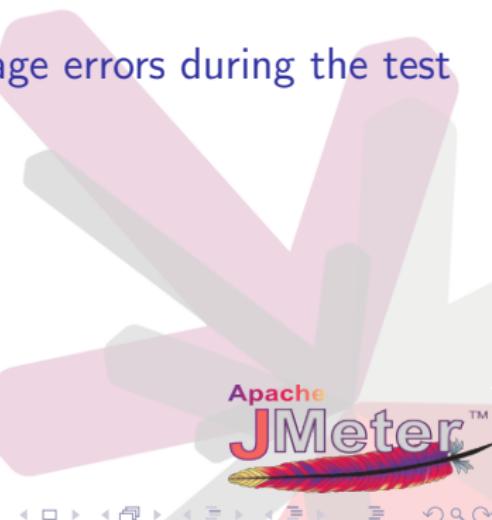
Dynamic Bandwidth Throttling

- ① DBT deals with the variation of bandwidth at runtime.
- ② DBT can be used to test performance of web services under varying bandwidth (load).



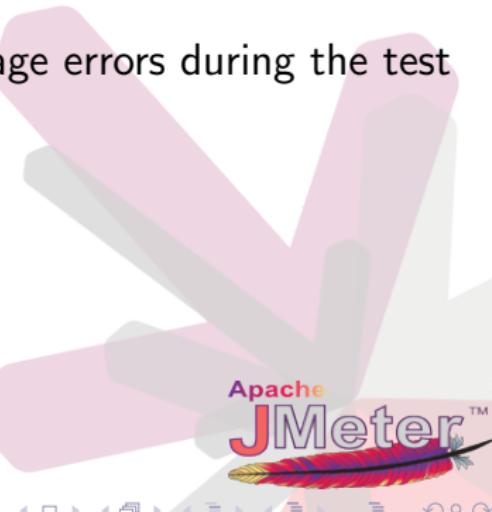
Dynamic Bandwidth Throttling

- ① DBT deals with the variation of bandwidth at runtime.
- ② DBT can be used to test performance of web services under varying bandwidth (load).
- ③ DBT can be used to measure and manage errors during the test at runtime.



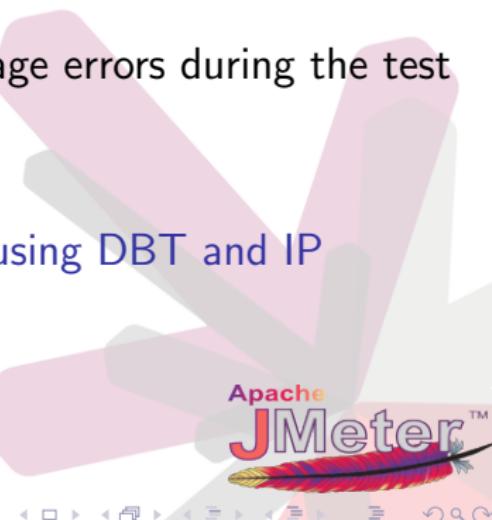
Dynamic Bandwidth Throttling

- ① DBT deals with the variation of bandwidth at runtime.
- ② DBT can be used to test performance of web services under varying bandwidth (load).
- ③ DBT can be used to measure and manage errors during the test at runtime.
- ④ Based on error rate.



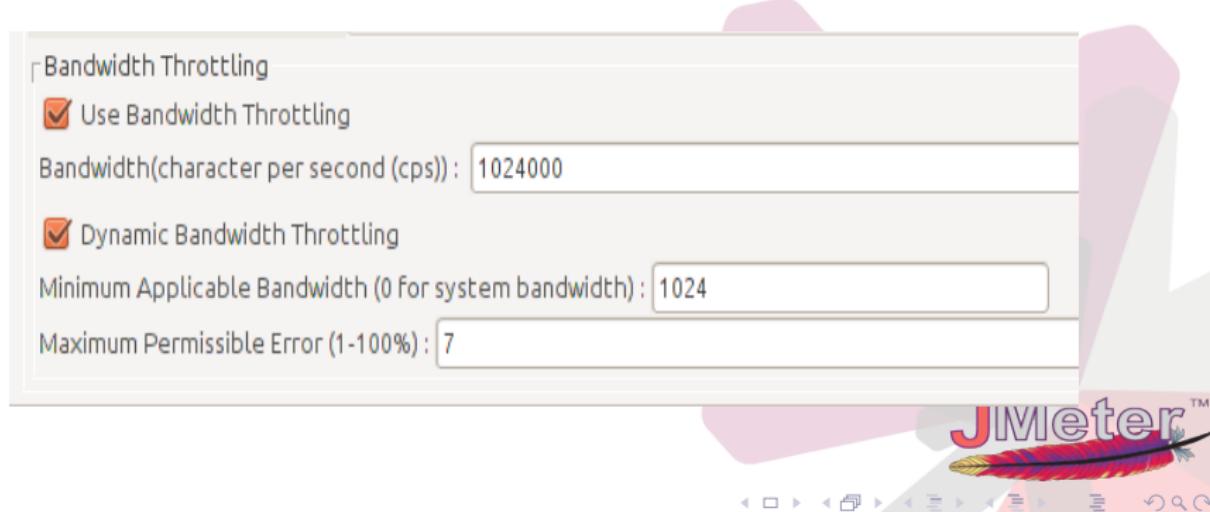
Dynamic Bandwidth Throttling

- ① DBT deals with the variation of bandwidth at runtime.
- ② DBT can be used to test performance of web services under varying bandwidth (load).
- ③ DBT can be used to measure and manage errors during the test at runtime.
- ④ Based on error rate.
- ⑤ A distributed testing can be simulated using DBT and IP spoofing.



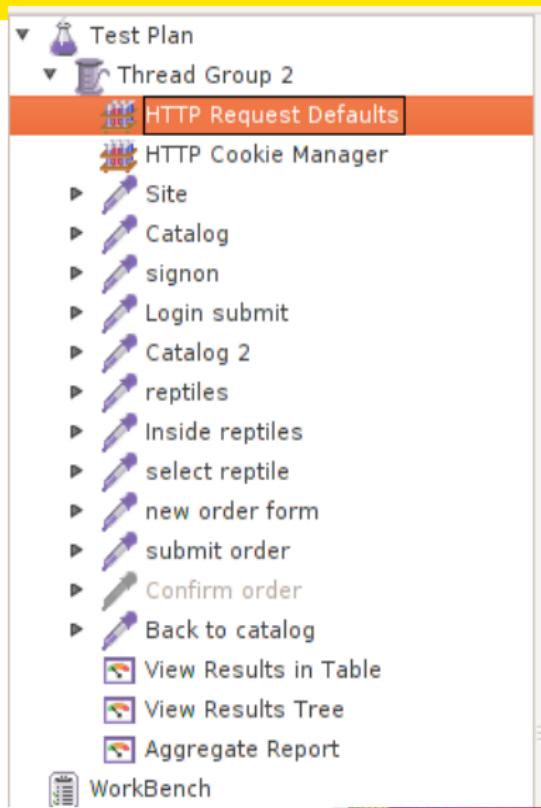
User Interface

To use Dynamic Bandwidth Throttling in JMeter, an extended GUI component has been added to Bandwidth Throttling in HTTP Request Defaults.



Test Plan

- Thread Group: 1
- Thread Count: 1000
- Ramp Up period: 0 sec
- Timeout period: 22 sec
- No. of Samplers: 11
- Permissible error: 7%
- Applicable bandwidth:
1MBps
- Minimum applicable
bandwidth: 1KBps



Aggregate Report

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename

Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
Site	1000	3431	3603	6341	8	8315	0.00%	82.9/sec	56.9
Catalog	1000	11100	10605	19757	44	35987	0.00%	22.5/sec	134.0
signon	1000	17621	21450	22039	549	33596	20.10%	12.9/sec	44.7
Login submit	1000	19773	20851	22037	1410	33216	9.00%	9.1/sec	33.0
Catalog 2	1000	21013	20961	21993	4290	36751	4.60%	6.9/sec	32.5
reptiles	1000	21312	20746	21841	10094	37571	2.90%	5.7/sec	26.8
Inside reptiles	1000	21670	21341	22020	18911	35739	5.80%	5.1/sec	23.5
select reptile	1000	21637	22030	22040	16554	27198	64.70%	5.2/sec	14.5
new order form	1000	21707	22028	22037	16557	23676	82.10%	5.3/sec	13.7
submit order	1000	21772	22028	22037	16605	23561	78.50%	5.4/sec	14.9
Back to catalog	1000	18737	19284	22030	11655	23560	28.40%	5.6/sec	22.9
TOTAL	11000	18161	21116	22037	8	37571	26.92%	39.2/sec	142.6

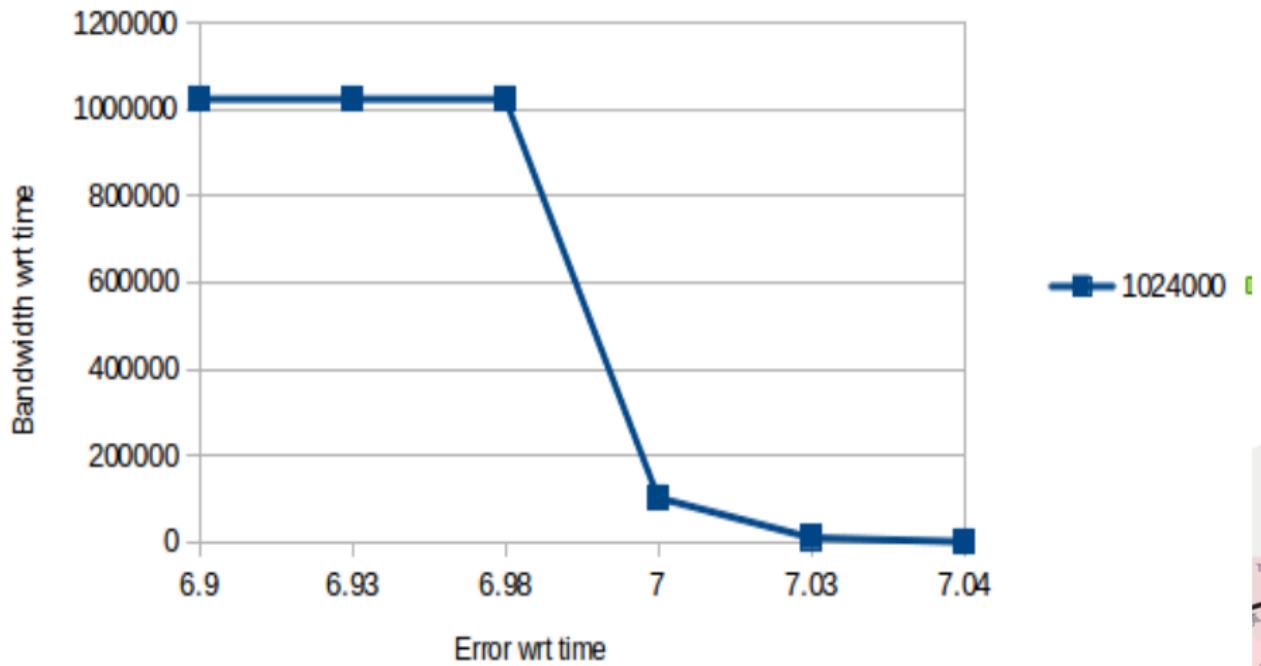


Aggregate Report

Error %	Throughput	KB/sec
0.00%	82.9/sec	56.9
0.00%	22.5/sec	134.0
20.10%	12.9/sec	44.7
9.00%	9.1/sec	33.0
4.60%	6.9/sec	32.5
2.90%	5.7/sec	26.8
5.80%	5.1/sec	23.5
64.70%	5.2/sec	14.5
82.10%	5.3/sec	13.7
78.50%	5.4/sec	14.9
28.40%	5.6/sec	22.9
26.92%	39.2/sec	142.6

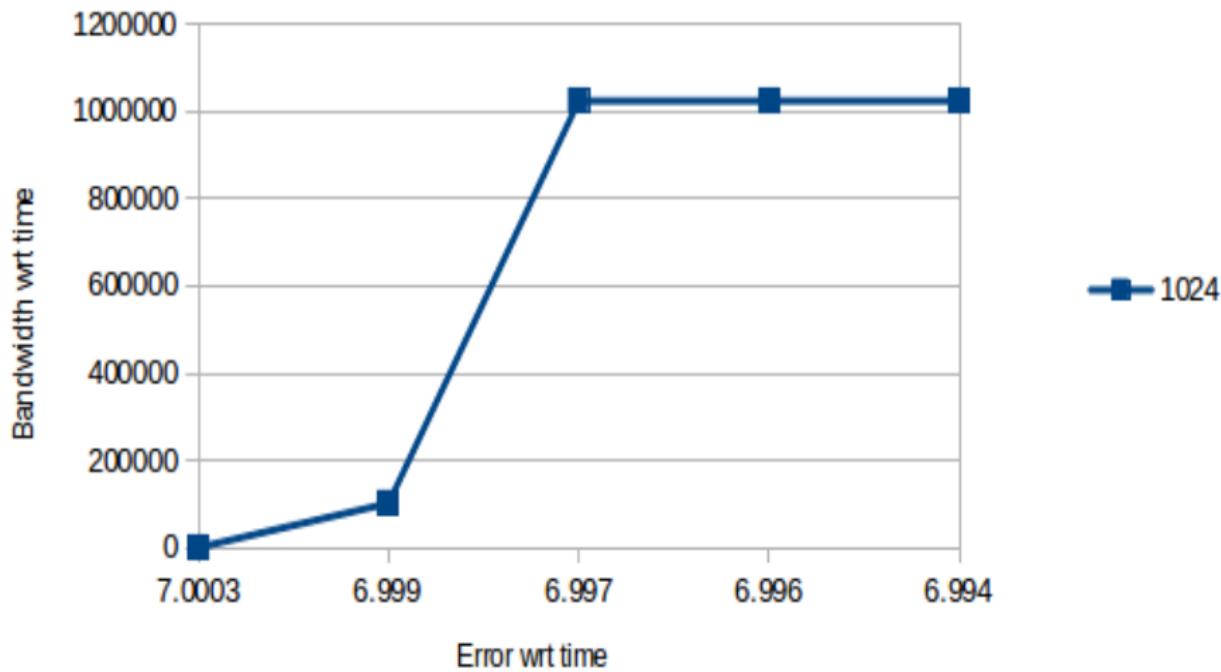
JMeter Log Report

Dynamic Bandwidth Throttling

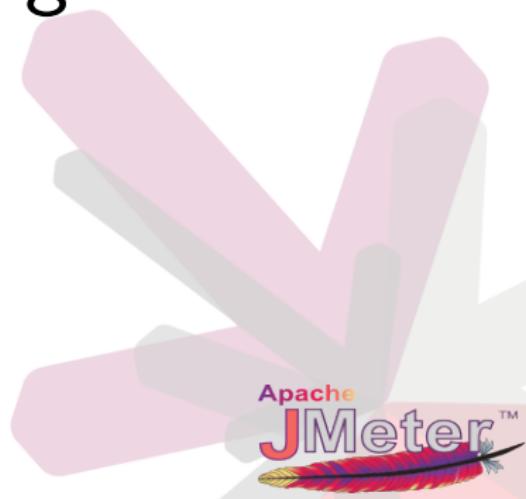


JMeter Log Report

Dynamic Bandwidth Throttling

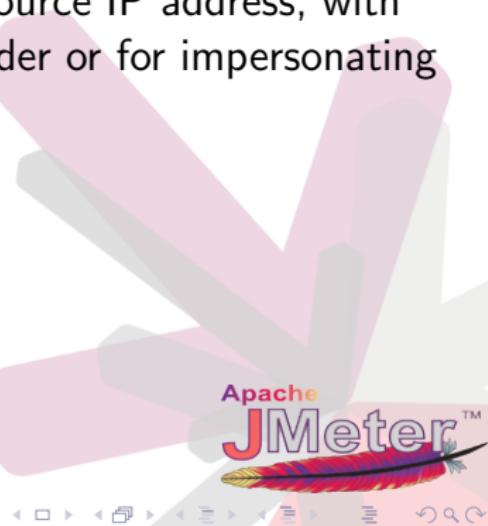


IP Spoofing



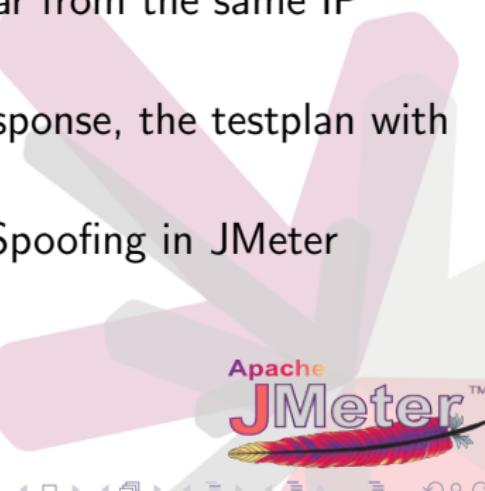
What is IP address Spoofing?

It is the creation of IP Packets with forged source IP address, with the purpose of concealing identity of the sender or for impersonating another computer system.



IP Spoofing in JMeter

- ① JMeter is capable of generating thousands of threads that act as virtual users
- ② On the server side, these requests appear from the same IP address on which JMeter resides
- ③ On servers which have IP dependent response, the testplan with a thousand virtual fails
- ④ To eliminate this drawback, we use IP Spoofing in JMeter



Without IP spoofing

Load balancing inactive

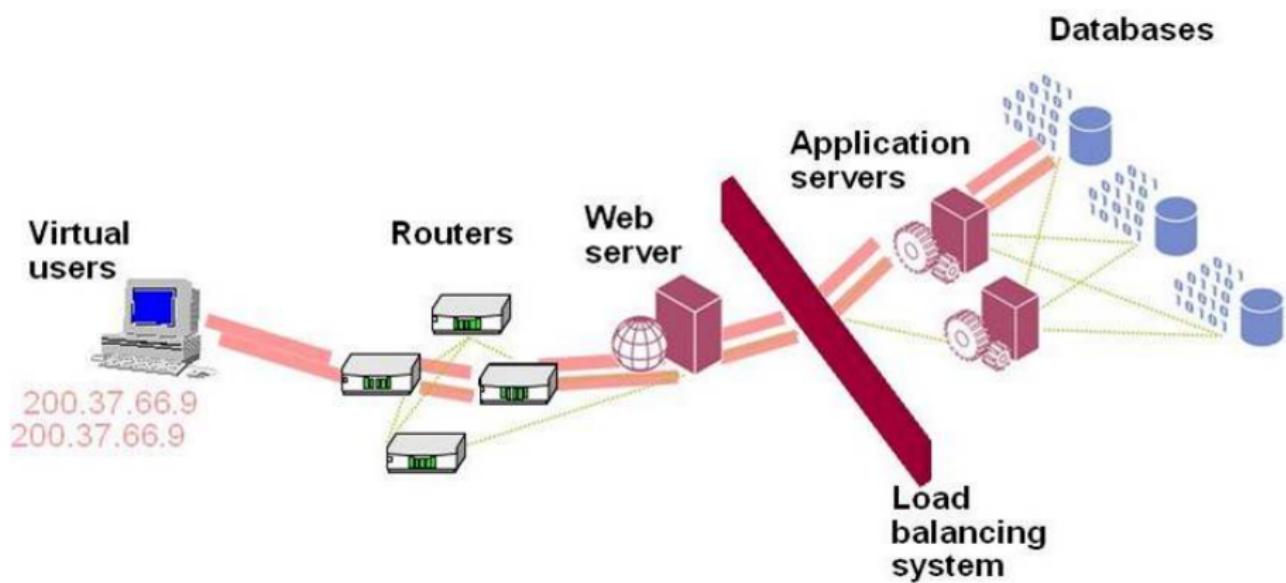
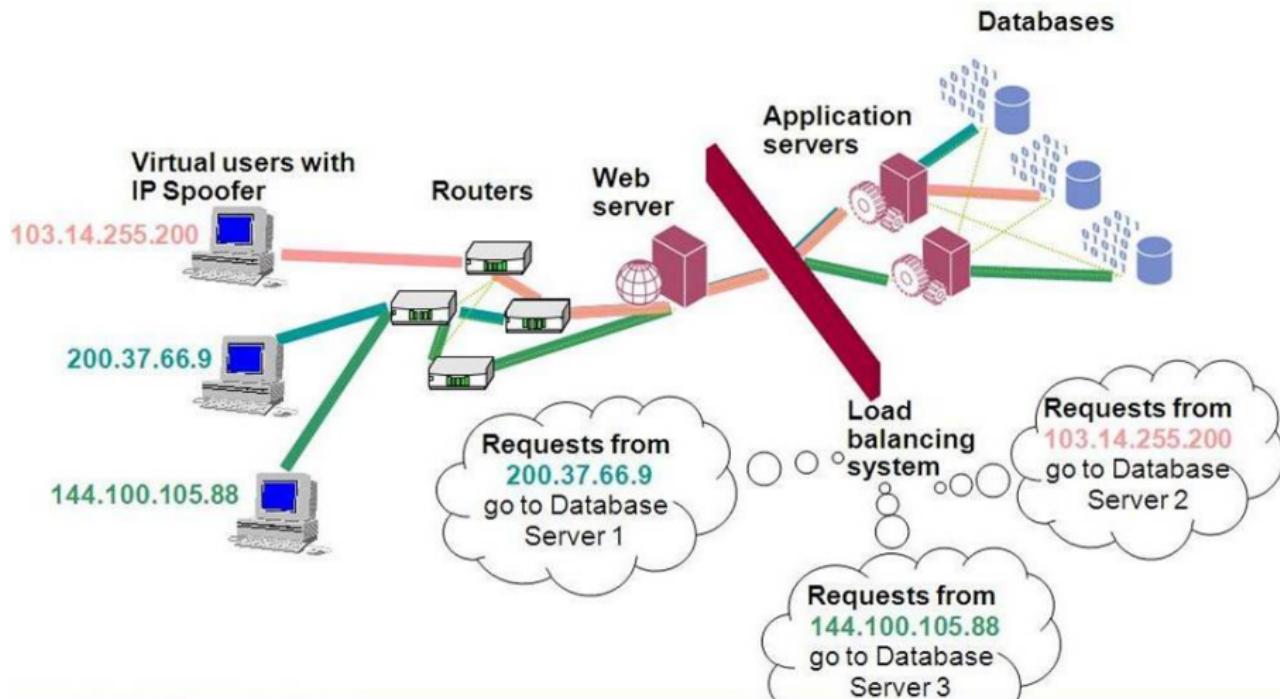


Figure: Server treating multiple requests without IP spoofing

With IP spoofing

Load balancing active



JMeter Implementation

IP and Subnet Details

We need to provide JMeter with IP address of the machine and the subnet it belongs to, and also specify the number of IP addresses required

IP allocation

JMeter internally allocates virtual IPs to the same machine, and each virtual user can send request from a distinct IP from newly allocated virtual IPs



Interface of IP Spoofing Config Element in JMeter

IP Spoofing

Name:

Comments:

IP Settings

IP Address: Subnet Mask: Number of Addresses:

Figure: GUI of IP Spoofing



Virtual IPs allocated to machine

Guake Terminal

```
wlan0:2 Link encap:Ethernet HWaddr 1c:4b:d6:43:8c:f7
      inet addr 192.170.0.2 Bcast:192.170.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      Test Plan
      IP Spoofing
      wlan0:3 Link encap:Ethernet HWaddr 1c:4b:d6:43:8c:f7
            inet addr 192.170.0.3 Bcast:192.170.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      View Results Tree
      wlan0:4 Link encap:Ethernet HWaddr 1c:4b:d6:43:8c:f7
            inet addr 192.170.0.4 Bcast:192.170.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            Subnet Mask: 255.255.255.0
            Create IP Addresses
      wlan0:5 Link encap:Ethernet HWaddr 1c:4b:d6:43:8c:f7
            inet addr 192.170.0.5 Bcast:192.170.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      wlan0:6 Link encap:Ethernet HWaddr 1c:4b:d6:43:8c:f7
            inet addr 192.170.0.6 Bcast:192.170.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

Figure: Server records same IP for each virtual user

Server Response Without IP Spoofing

access.log00095 (~/myDomain/JMeter/servers/AdminServer/logs) - gedit

```

access.log00095 x
36 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
37 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
38 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
39 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
40 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
41 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
42 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
43 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
44 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
45 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
46 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
47 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
48 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
49 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
50 127.0.0.1 - - [26/Jun/2013:00:55:00 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
51 127.0.0.1 - - [26/Jun/2013:00:55:01 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
52 127.0.0.1 - - [26/Jun/2013:00:55:01 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
53 127.0.0.1 - - [26/Jun/2013:00:55:01 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
54 127.0.0.1 - - [26/Jun/2013:00:55:01 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498

```

Figure: Server records same IP for each virtual user

Server Response With IP Spoofing

access.log (~/myDomain/JMeter/servers/AdminServer/logs) - gedit

```
access.log x
[192.170.0.1 - [03/Jul/2013:00:49:56 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.2 - [03/Jul/2013:00:49:56 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.4 - [03/Jul/2013:00:49:56 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.3 - [03/Jul/2013:00:49:56 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.5 - [03/Jul/2013:00:49:56 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.6 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.7 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.8 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.9 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.10 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.11 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.12 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.13 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.14 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.15 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.16 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.17 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.18 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.19 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
[192.170.0.20 - [03/Jul/2013:00:49:57 +0530] "GET /mybatis-jpetstore-6.0.1/ HTTP/1.1" 200 498
```

Figure: Server records distinct IP for each virtual user

Automating TPC-C Benchmarking



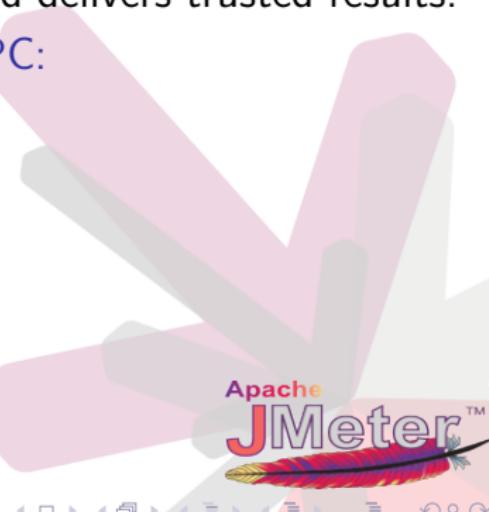
Automating TPC-C Benchmarking

- TPC- Transaction Processing Council It defines transaction processing and database benchmarks and delivers trusted results.



Automating TPC-C Benchmarking

- TPC- Transaction Processing Council It defines transaction processing and database benchmarks and delivers trusted results.
- Some benchmarks undertaken under TPC:
 - TPC-APP
 - TPC-H
 - TPC-C

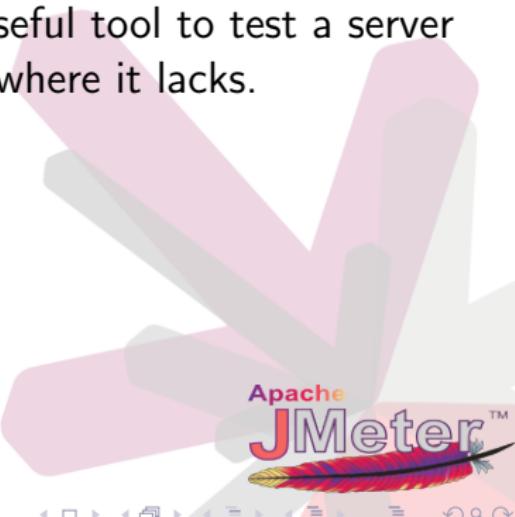


WHY TPC-C?

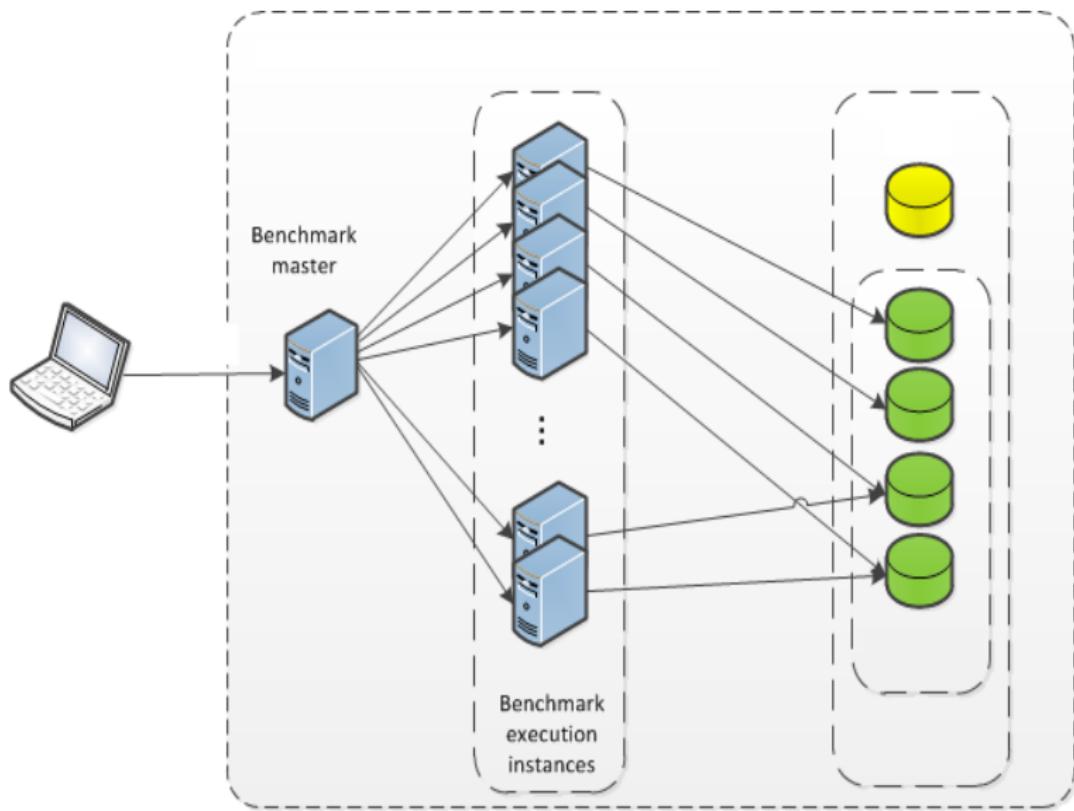
- A number of these benchmarks have been deprecated. TPC-C is currently in use and a rather complex process.
- A number of business houses use this benchmark to showcase their performance for OLTP transactions. It gives the measure of Server speed for online transaction processing.



- The actual benchmarking process is a time taking and a costly affair.
- A preliminary test would be a highly useful tool to test a server for performance and hence improve it where it lacks.

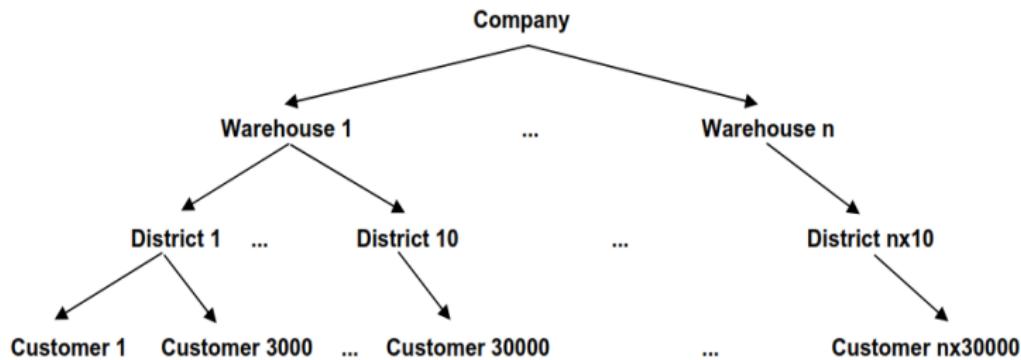


The Benchmarking Model



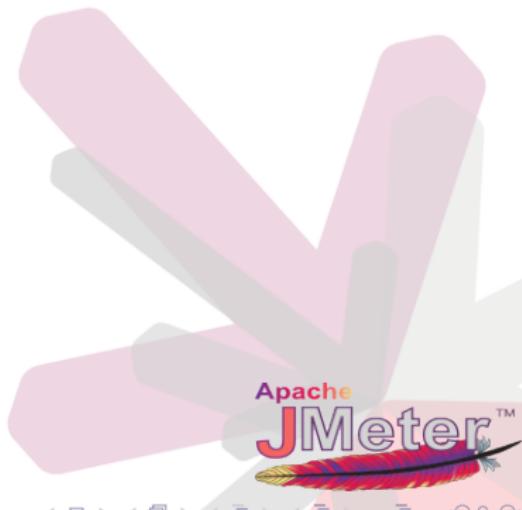
The Benchmarking Model

The model emulated for TPC-C



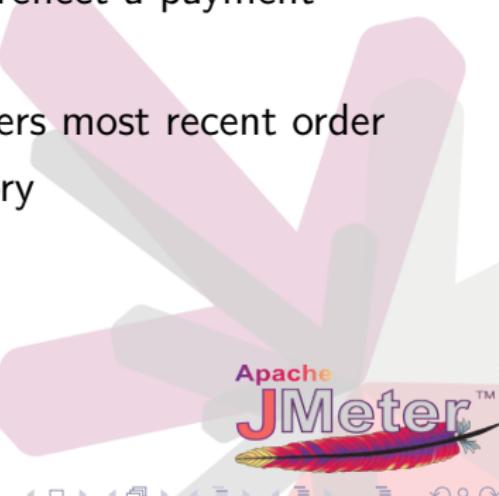
Tables in the TPC-C schema

- Item
- Warehouse
- History
- District
- Customer
- New Order
- Orders
- Order Line
- Stock

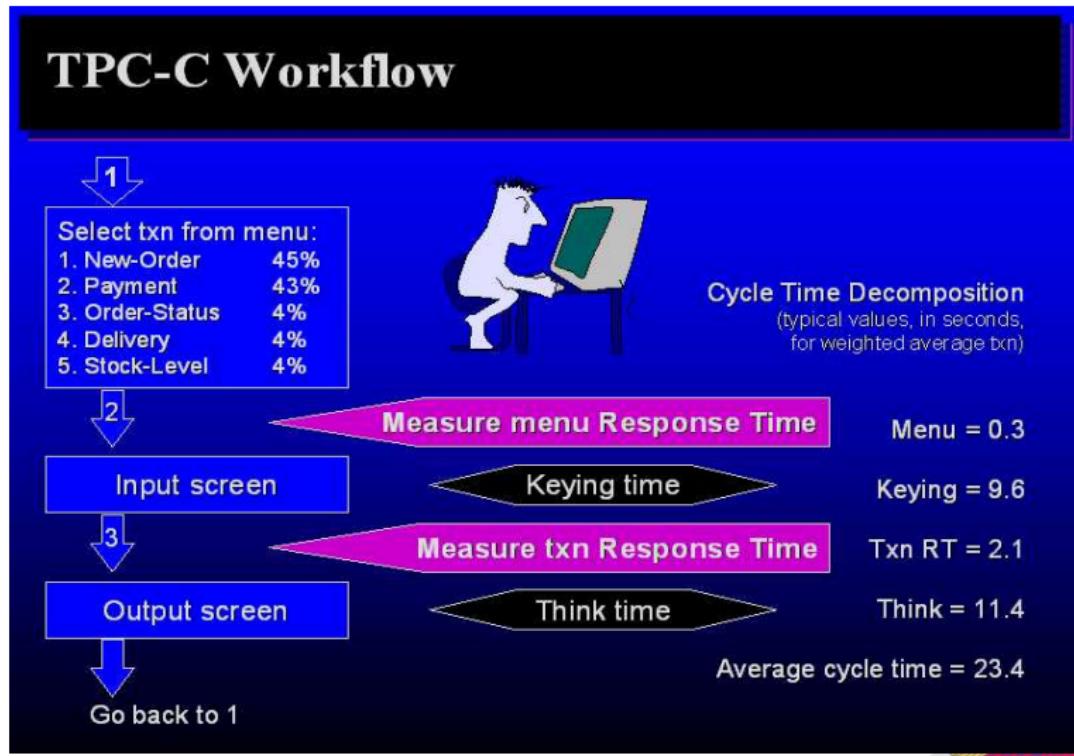


Transactions

- New-order: enter a new order from a customer
- Payment: update customer balance to reflect a payment
- Delivery: deliver orders
- Order-status: retrieve status of customers most recent order
- Stock-level: monitor warehouse inventory



TPC-C Workflow



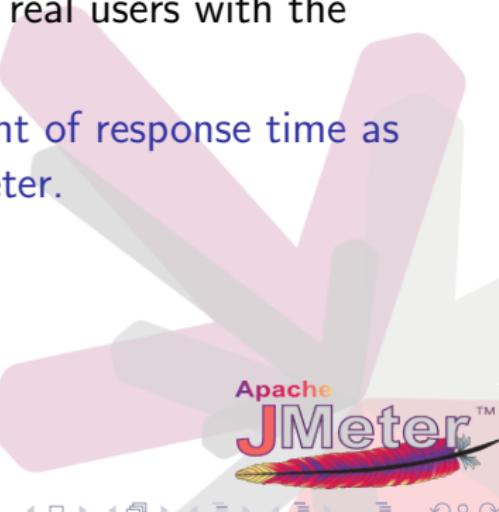
Why JMeter to automate??

- JMeter is already capable of spawning a large number of virtual users to simulate the interaction of the real users with the system under test.



Why JMeter to automate??

- JMeter is already capable of spawning a large number of virtual users to simulate the interaction of the real users with the system under test.
- The firing of a request and measurement of response time as well as throughput is embedded in JMeter.



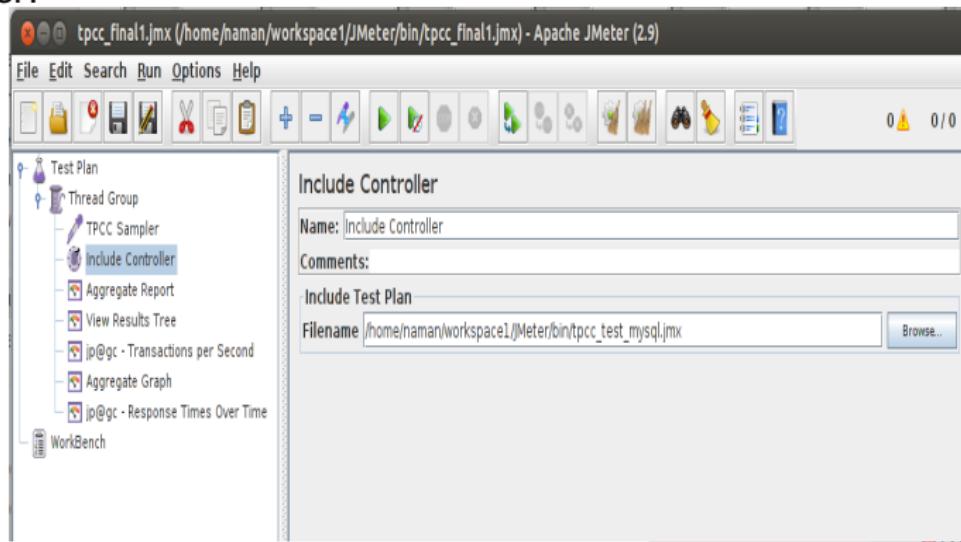
TPC-C Testing in JMeter

The screenshot shows the Apache JMeter 2.9 interface with the file "tpcc_final1.jmx" open. The left sidebar displays a hierarchical tree structure under "Test Plan": "Thread Group" contains "TPCC Sampler", "Include Controller", "Aggregate Report", "View Results Tree", "jp@gc - Transactions per Second", "Aggregate Graph", and "jp@gc - Response Times Over Time". Below "Thread Group" is "WorkBench". The main panel is titled "TPCC Sampler" and contains the following configuration fields:

Name:	TPCC Sampler
Comments:	
Configurations	
Database URL:	jdbc:mysql://localhost:3306/
Driver Class:	com.mysql.jdbc.Driver
Username:	root
Password:
Database Name:	TPCC
Number Of Warehouses:	2
<input checked="" type="radio"/> MySQL <input type="radio"/> Oracle	
<input type="button" value="Create Database"/> <input type="button" value="Start Test"/>	

Controllers

Include Controller is a component of JMeter used to run saved scripts in JMeter.



Test with 1 warehouse

The screenshot shows the phpMyAdmin interface for the 'tpcc_new' database. The left sidebar lists recent tables and the current table list: customer, district, history, item, new_order, orders, order_line, stock, and warehouse. The main area displays a table of 9 rows with columns for Action, Table, Rows, Type, Collation, Size, and Overhead. The 'customer' table has 29,998 rows and 21.1 MiB size. The 'district' table has 10 rows and 16 KiB size. The 'history' table has 20,290 rows and 2.5 MiB size. The 'item' table has 100,013 rows and 9.5 MiB size. The 'new_order' table has 5,919 rows and 368 KiB size. The 'orders' table has 33,545 rows and 4 MiB size. The 'order_line' table has 316,800 rows and 29.6 MiB size. The 'stock' table has 100,059 rows and 34.6 MiB size. The 'warehouse' table has 1 row and 16 KiB size. A summary at the bottom indicates 619,735 rows and 101.7 MiB size.

Action	Table	Rows	Type	Collation	Size	Overhead
Browse Structure Search Insert Empty Drop	customer	29,998	InnoDB	latin1_swedish_ci	21.1 MiB	-
Browse Structure Search Insert Empty Drop	district	10	InnoDB	latin1_swedish_ci	16 KiB	-
Browse Structure Search Insert Empty Drop	history	20,290	InnoDB	latin1_swedish_ci	2.5 MiB	-
Browse Structure Search Insert Empty Drop	item	100,013	InnoDB	latin1_swedish_ci	9.5 MiB	-
Browse Structure Search Insert Empty Drop	new_order	5,919	InnoDB	latin1_swedish_ci	368 KiB	-
Browse Structure Search Insert Empty Drop	orders	33,545	InnoDB	latin1_swedish_ci	4 MiB	-
Browse Structure Search Insert Empty Drop	order_line	316,800	InnoDB	latin1_swedish_ci	29.6 MiB	-
Browse Structure Search Insert Empty Drop	stock	100,059	InnoDB	latin1_swedish_ci	34.6 MiB	-
Browse Structure Search Insert Empty Drop	warehouse	1	InnoDB	latin1_swedish_ci	16 KiB	-
9 tables	Sum	619,735	InnoDB	latin1_swedish_ci	101.7 MiB	0 B

Check All / Uncheck All With selected:

Print view Data Dictionary

Create table

Procedures

The screenshot shows the phpMyAdmin interface for the 'tpcc_new' database. The left sidebar lists tables: _new, customer, district, history, item, new_order, orders, order_line, stock, and warehouse. A 'Create table' button is also present. The main panel displays a table titled 'Routines' with the following data:

Name	Action	Type	Returns
DELIVERY	Edit Execute Export Drop	PROCEDURE	
NEWORD	Edit Execute Export Drop	PROCEDURE	
OSTAT	Edit Execute Export Drop	PROCEDURE	
PAYOUT	Edit Execute Export Drop	PROCEDURE	
SLEV	Edit Execute Export Drop	PROCEDURE	

A 'New' button and an 'Add routine' link are located at the bottom of the routines list.



Test with 33 Warehouses

The screenshot shows the phpMyAdmin interface for a database named 'tpcc'. The main area displays a table of 9 tables with their respective details:

Table	Action	Rows	Type	Collation	Size	Overhead
customer	Browse Structure Search Insert Empty Drop	~987,934	InnoDB	latin1_swedish_ci	715.8 MiB	-
district	Browse Structure Search Insert Empty Drop	330	InnoDB	latin1_swedish_ci	80 KiB	-
history	Browse Structure Search Insert Empty Drop	~1,220,619	InnoDB	latin1_swedish_ci	107.7 MiB	-
item	Browse Structure Search Insert Empty Drop	~100,086	InnoDB	latin1_swedish_ci	9.5 MiB	-
new_order	Browse Structure Search Insert Empty Drop	~295,463	InnoDB	latin1_swedish_ci	12.5 MiB	-
orders	Browse Structure Search Insert Empty Drop	~1,009,529	InnoDB	latin1_swedish_ci	105.2 MiB	-
order_line	Browse Structure Search Insert Empty Drop	~10,032,724	InnoDB	latin1_swedish_ci	0.9 GiB	-
stock	Browse Structure Search Insert Empty Drop	~3,294,549	InnoDB	latin1_swedish_ci	1.1 GiB	-
warehouse	Browse Structure Search Insert Empty Drop	33	InnoDB	latin1_swedish_ci	16 KiB	-
9 tables	Sum	~16,942,127	InnoDB	latin1_swedish_ci	3 GiB	0 B

Below the table, there are buttons for 'Check All / Uncheck All' and 'With selected:'. Further down, there are links for 'Print view' and 'Data Dictionary', and a 'Create table' form with fields for 'Name:' and 'Number of columns:'.

JDBC Configuration

The screenshot shows the JMeter interface with a Test Plan tree on the left and a detailed configuration dialog on the right.

Test Plan Tree:

- Test Plan
 - Thread Group
 - JDBC Connection Configuration
 - Loop Controller
 - Random Order Controller
 - stock level
 - Keying Timer
 - Think Timer
 - delivery
 - order_status
 - NEW_ORDER
 - payment
 - payment
 - payment

JDBC Connection Configuration

Name: JDBC Connection Configuration

Comments:

Variable Name Bound to Pool

Variable Name: mypool

Connection Pool Configuration

Max Number of Connections: 0

Pool Timeout: 10000

Idle Cleanup Interval (ms): 60000

Auto Commit: True

Transaction Isolation: DEFAULT

Connection Validation by Pool

Keep-Alive: True

Max Connection age (ms): 5000

Validation Query: Select 1

Database Connection Configuration

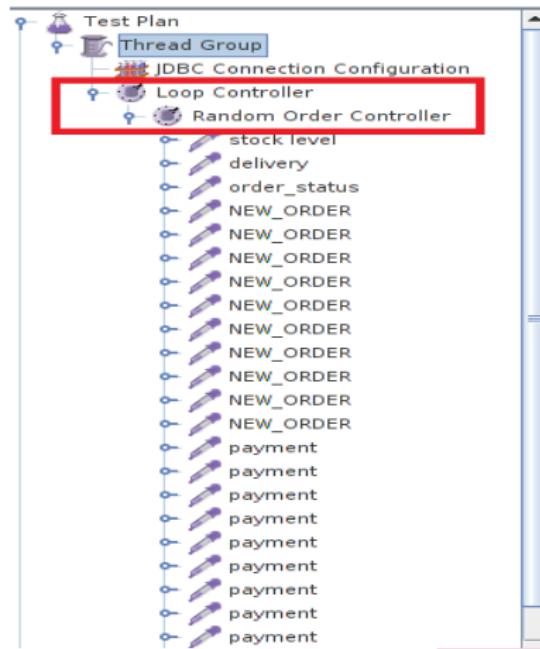
Database URL: \${__geturl()}

DBC Driver class: \${__getdriver()}

Username: \${__getusername()}

Password: *****

Controllers



Timers

The image shows the JMeter interface. On the left, the Test Plan tree is displayed with the following structure:

- Test Plan
 - Thread Group
 - JDBC Connection Configuration
 - Loop Controller
 - Random Order Controller
 - stock level
 - Keying Timer
 - Think Timer
 - delivery
 - order_status

Gaussian Random Timer

Name: Think Timer

Comments:

Thread Delay Properties

Deviation (in milliseconds): 5000

Constant Delay Offset (in milliseconds): 5000



Transaction Call

prelims_tpcc.jmx (C:\Users\surabhi\workspace\JMeter\bin\prelims_tpcc.jmx) - Apache JMeter (2.9)

File Edit Search Run Options Help

JDBC Request

Name: payment

Comments:

Variable Name Bound to Pool

Variable Name: mypool

SQL Query

Query Type: Callable Statement

call payment(?,?);

Parameter values: \${__counter(tpcc)},\${__counter_dist_tpcc()},\${__counter(tpcc)},\${__counter_dist_tpcc()},\${__getCustomerID()},\${__byname()},\${__randomNumberTpcc}

Parameter types: INTEGER,INTEGER,INTEGER,INOUT INTEGER,INTEGER,DECIMAL,INOUT VARCHAR,OUT VARCHAR,OUT VARCHAR,OUT VARCHAR,OUT VARCHAR

Variable names:

Result variable name:

12:28 PM
6/15/2013

Windows Taskbar icons: Start, File Explorer, File, Internet Explorer, Help, JMeter, Enhacement of JMeter, Paint.

Function Helper

JDBC Request

Name: payment

Comments:

Variable Name Bound to Pool

Variable Name: mypool

SQL Query

Query Type: Callable Statement

Function Helper

Choose a function: _BeanShell

Name:	Value
formattedDouble	
getCurrentTime_tpcc	
getCustomerID	
getitemId	
getLastname	
getURandLNameRun	
GetWarehouseID	
intSum	

Detail Add Delete

Copy and paste function string: _____ Generate

Parameter values: \${__counterTpcc()}, \${__counter_dist_tpcc()}, \${__counterTpcc()}, \${__counter_dist_tpcc()}, \${__getCustomerID()}, \${__byname()}, \${__}

Parameter types: INTEGER,INTEGER,INTEGER,INOUT INTEGER,INTEGER,DECIMAL,INOUT VARCHAR,OUT VARCHAR,OUT VARCHAR,OUT VARCHAR

Variable names: _____

Result variable name: _____

Procedure Request

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename Log/Display Only: Errors Successes

	Sampler result	Request	Response data
NEW_ORDER			
payment			
NEW_ORDER			
payment			
NEW_ORDER			

```
[Callable Statement] call payment(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);
1,8,1,9,2177,0,504,PRESPRIPRI,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2013-06-15 12:29:41
INTEGER,INTEGER,INTEGER,INTEGER,INOUT INTEGER,INTEGER,DECIMAL,INOUT VARCHAR,OUT VARCHAR,OUT
VARCHAR,OUT VARCHAR,OUT CHAR,OUT CHAR,OUT VARCHAR,OUT VARCHAR,OUT CHAR,OUT CHAR,OUT CHAR,OUT
VARCHAR,OUT CHAR,OUT VARCHAR,OUT VARCHAR,OUT VARCHAR,OUT CHAR,OUT CHAR,OUT CHAR,OUT DATE,INOUT
CHAR,OUT DECIMAL,OUT DECIMAL,INOUT DECIMAL,OUT VARCHAR,DATE
```



Response

JMeter Team ()

Enhancement of JMeter

03 July 2013

66 / 78

The screenshot shows the JMeter interface with the 'View Results Tree' configuration selected in the left sidebar. The configuration includes:

- Name: View Results Tree
- Comments:
- Write results to file / Read from file
- Filename: (empty)
- Browse... button

The main panel displays the 'View Results Tree' results. The tree structure shows multiple 'payment' requests, each containing 'Keying Timer' and 'Think Timer' elements. The results list contains numerous entries for 'NEW_ORDER' and 'payment' samplers, along with other variables like 'order_status'. The results list is as follows:

- [5] 2177
- [8] ABLEATIONATION
- [9] MONZT5SqpRroon0
- [10] SbDIRYVFRfwby
- [11] QsftcmM5XnHxCmP7FkKE
- [12] j
- [13] 345911111
- [14] 860PHK269qlY
- [15] Nhsbe9uUHPOCJSxK
- [16] 72tLHOpe82q
- [17] jq
- [18] 382911111
- [19] X82m9Bq1BVOXW3ur
- [20] OE
- [21] Q7UhjVJgGSXGNxDcnv
- [22] 807lcGsWYNQ6Yx5
- [23] TAJRKR0NdeYk8H
- [24] g2
- [25] 680511111
- [26] 8351593852376079
- [27] 2013-06-15
- [28] GC
- [29] 50000.00
- [30] 0.4200
- [31] 494.00

At the bottom, there are buttons for 'Text' and 'CSV' output formats, and a checkbox for 'Scroll automatically?'. The status bar at the bottom right shows 'TM'.

Aggregate Report

Aggregate Report

Name: Aggregate Report

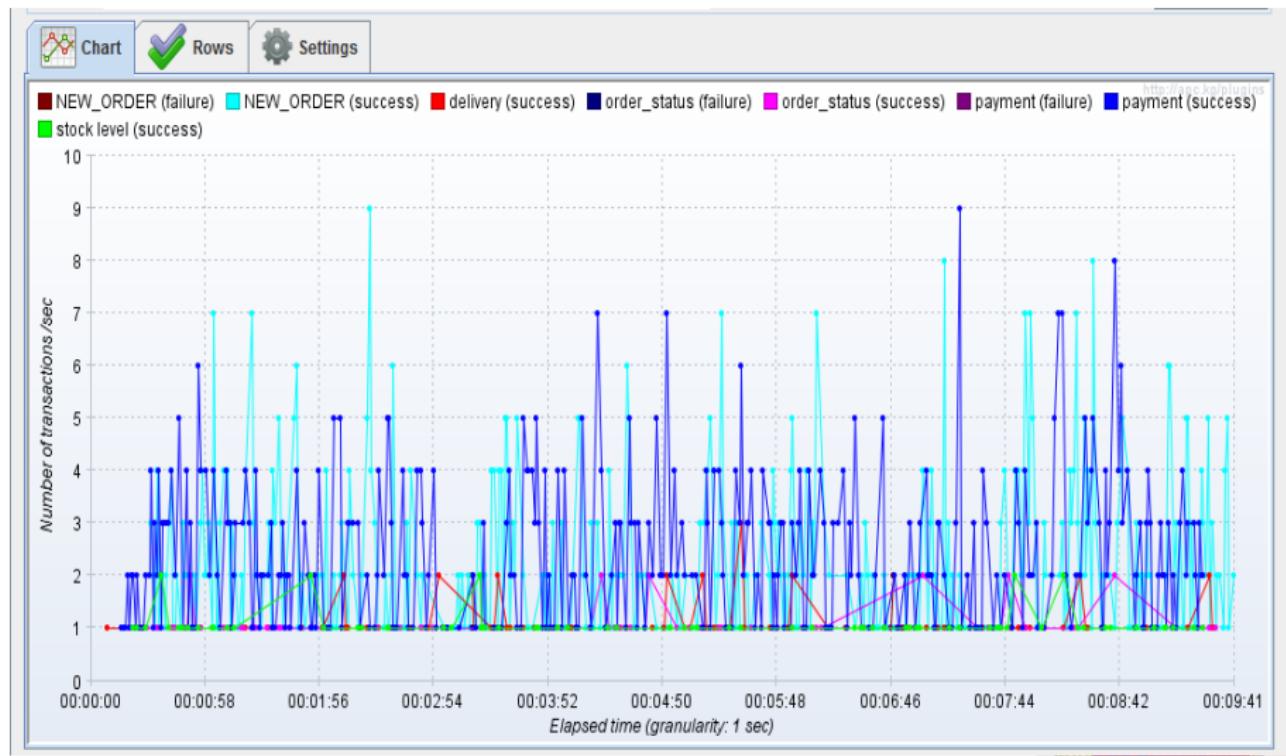
Comments:

Write results to file / Read from file

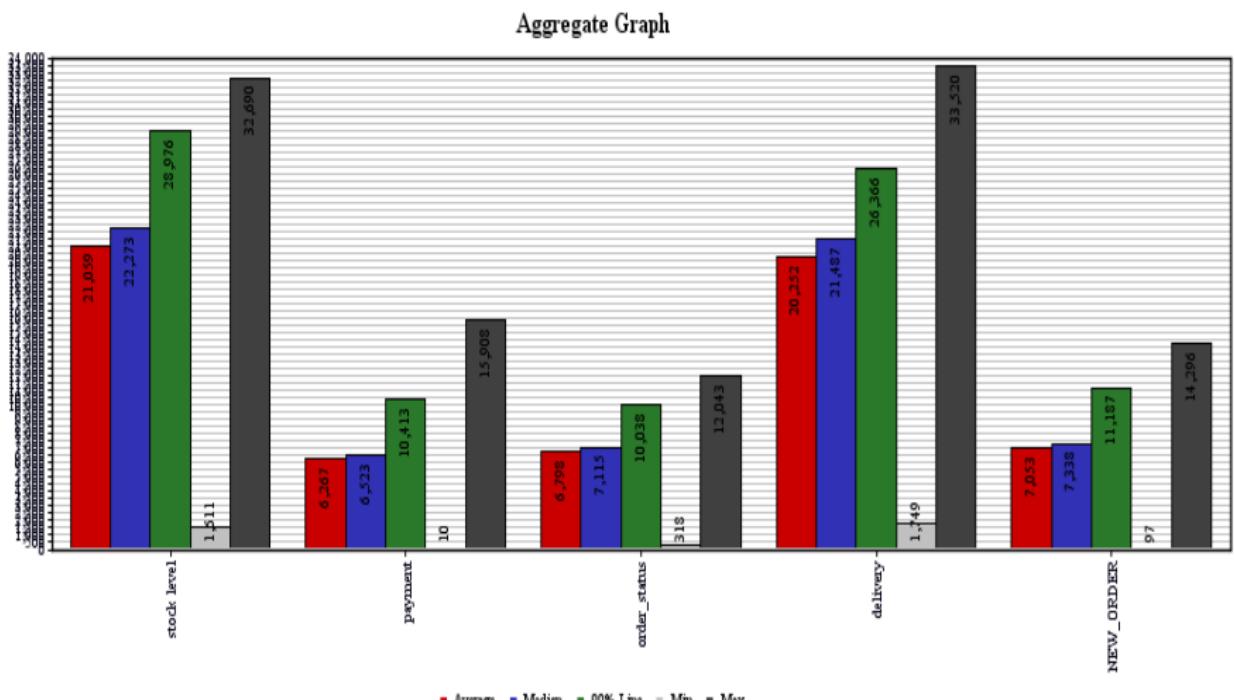
Filename Browse... Log/Display Only: Errors Successes

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
payment	74	30	27	59	8	149	0.00%	14.8/min	.1
NEW_ORDER	67	85	76	120	8	632	0.00%	13.4/min	.0
order_status	4	29	30	52	4	52	0.00%	53.3/min	.0
delivery	6	146	125	174	104	216	0.00%	2.9/min	.0
stock level	6	62	6	12	4	335	0.00%	2.9/min	.0
TOTAL	157	59	43	120	4	632	0.00%	30.0/min	.2

Transactions per second

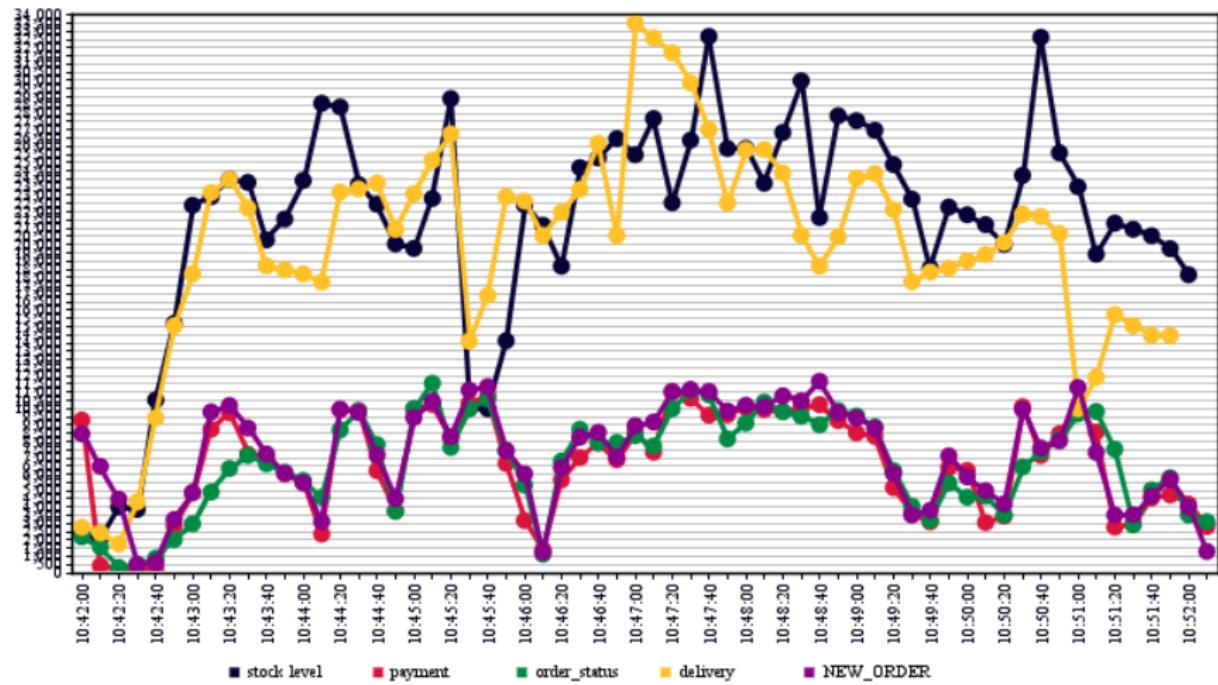


Aggregate Graph



Response time over time

Response Time Graph



Challenges

① Auto CSV Generation:

Generating CSV file for a particular table from a specified Database



Challenges

① Auto CSV Generation:

Generating CSV file for a particular table from a specified Database

② Dynamic Bandwidth Throttling:

Changing bandwidth in runtime and determining a reliable parameter (Percentage Error) to implement throttle bandwidth



Challenges

- ① Auto CSV Generation:
Generating CSV file for a particular table from a specified Database
- ② Dynamic Bandwidth Throttling:
Changing bandwidth in runtime and determining a reliable parameter (Percentage Error) to implement throttle bandwidth
- ③ IP Spoofing:
Finding an array of un-used IPs and automating allocation of virtual IPs for a system

Challenges

- ① Auto CSV Generation:
Generating CSV file for a particular table from a specified Database
- ② Dynamic Bandwidth Throttling:
Changing bandwidth in runtime and determining a reliable parameter (Percentage Error) to implement throttle bandwidth
- ③ IP Spoofing:
Finding an array of un-used IPs and automating allocation of virtual IPs for a system
- ④ Automating TPC-C testing:
Making tpcc testing capable of being run with different databases incorporating all the standards into JMeter

Future Work

- ① Incorporating other Benchmarking support such as TPC-H, TPC-E etc. into JMeter.



Future Work

- ① Incorporating other Benchmarking support such as TPC-H, TPC-E etc. into JMeter.
- ② Automation of the test scripts, as in user may not have to create the test script, and Jmeter can itself do it for the tester by techniques such as web crawling, etc.



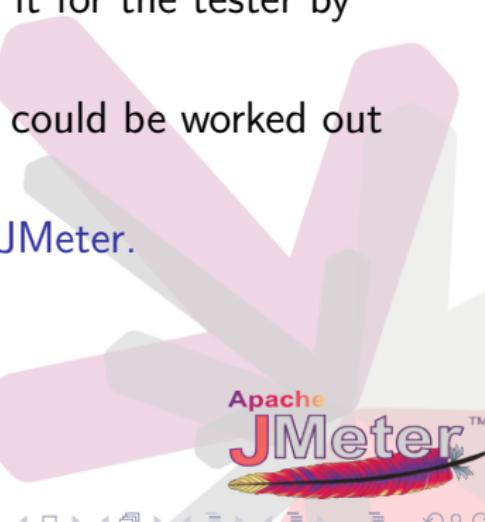
Future Work

- ① Incorporating other Benchmarking support such as TPC-H, TPC-E etc. into JMeter.
- ② Automation of the test scripts, as in user may not have to create the test script, and Jmeter can itself do it for the tester by techniques such as web crawling, etc.
- ③ The instability of JMeter on large loads could be worked out with some solution.



Future Work

- ① Incorporating other Benchmarking support such as TPC-H, TPC-E etc. into JMeter.
- ② Automation of the test scripts, as in user may not have to create the test script, and Jmeter can itself do it for the tester by techniques such as web crawling, etc.
- ③ The instability of JMeter on large loads could be worked out with some solution.
- ④ Bringing large download efficiency into JMeter.



Future Work

- ① Incorporating other Benchmarking support such as TPC-H, TPC-E etc. into JMeter.
- ② Automation of the test scripts, as in user may not have to create the test script, and Jmeter can itself do it for the tester by techniques such as web crawling, etc.
- ③ The instability of JMeter on large loads could be worked out with some solution.
- ④ Bringing large download efficiency into JMeter.
- ⑤ Better analysis of the results produced by JMeter via some complex graphs and better comparison between different graph results.



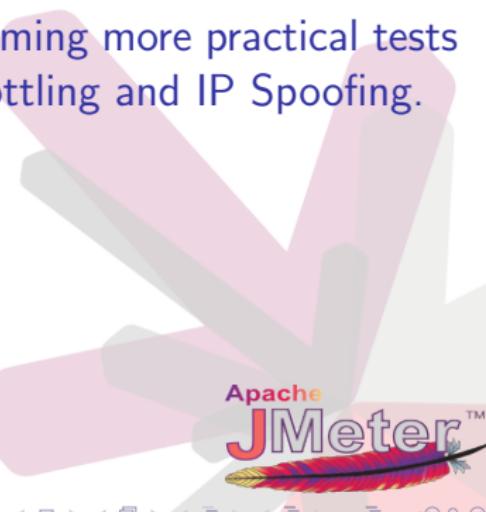
Conclusion

- The basic aim of the project was to enhance JMeter with the introduction of some new features and overcome some drawbacks.



Conclusion

- The basic aim of the project was to enhance JMeter with the introduction of some new features and overcome some drawbacks.
- We have made JMeter capable of performing more practical tests with the introduction of bandwidth throttling and IP Spoofing.

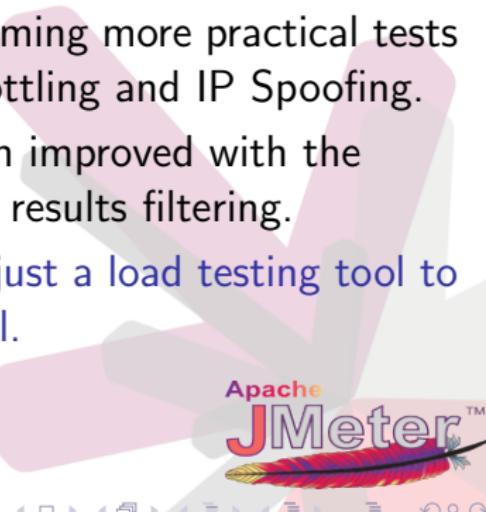


Conclusion

- The basic aim of the project was to enhance JMeter with the introduction of some new features and overcome some drawbacks.
- We have made JMeter capable of performing more practical tests with the introduction of bandwidth throttling and IP Spoofing.
- The user friendliness of JMeter has been improved with the introduction of auto csv generation and results filtering.

Conclusion

- The basic aim of the project was to enhance JMeter with the introduction of some new features and overcome some drawbacks.
- We have made JMeter capable of performing more practical tests with the introduction of bandwidth throttling and IP Spoofing.
- The user friendliness of JMeter has been improved with the introduction of auto csv generation and results filtering.
- Finally Jmeter has been extended from just a load testing tool to a Preliminary TPC-C benchmarking tool.

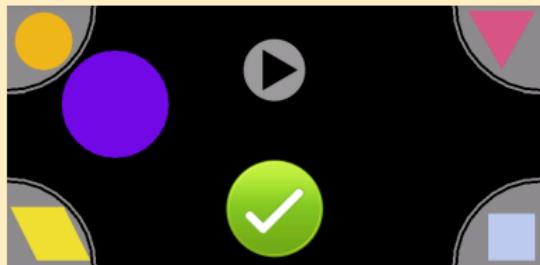


References

- [1] E. H. Hallili, *Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites.* Packt Publishing Ltd., 2008.
- [2] "Apache Jmeter. Available at <http://jmeter.apache.org/>," February 2013.
- [3] " Apache Jmeter usermanual. Available at [/http://jmeter.apache.org/usermanual/component_reference.html#introduction](http://jmeter.apache.org/usermanual/component_reference.html#introduction)."
- [4] "Installation of Jmeter Googlecode plugins <http://www.code.google.com/p/jmeter-plugins/>," February 2012.
- [5] "Jmeter Googlecode plugins testplan [http://shantonusarker.blogspot.in/2013/05/introduction-to-jmeter-google-plugin.html/](http://shantonusarker.blogspot.in/2013/05/introduction-to-jmeter-google-plugin.html)," May 2013.
- [6] " Standard Benchmarks for database systems. Available at www.tpc.org/information/sessions/sigmod/sigmod97.ppt," 1997.
- [7] "TPC-C Benchmark standard specification. Available at www.tpc.org/tpcc/spec/tpcc_current.pdf," February 2010.
- [8] "Benchmark overview of TPC-C. Available at http://globalsp.ts.fujitsu.com/dmsp/Publications/public/Benchmark_Overview_TPC-C.pdf," October 2003.
- [9] "Tpc-c benchmark kit. available at <http://docs.openlinksw.com/virtuoso/tpcc.html>."
- [10] "comparisons between LoadRunner vs. Grinder vs. JMeter. Available at <http://blackanvil.blogspot.in/2006/06/shootout-load-runner-vs-grinder-vs.html>," June 2006.
- [11] "Jmeter Plug-in Results [http://www.methodsandtools.com/tools/jmeterplugins.php/](http://www.methodsandtools.com/tools/jmeterplugins.php)," May 2012.

Educational Application

Learning Shapes



A simple application to teach young minds the sense of shapes and colors. This app tests the recognition skill shapes and colors among children

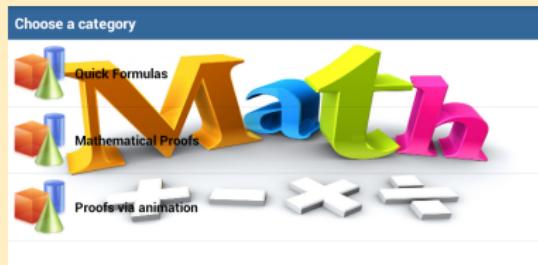
Traingles for High School



An application to aid learning about Triangles, allows construction of Triangles, also drawing Triangles from user Input

Educational Application

Mathematics Playground



Best friend of any student during examination for quick revision of the formulas. The detailed proofs and animation helps the student in recalling the logic behind the formula

Base Conversion

Input Base : 2
11011|

Output Base : 16

Showing Description :

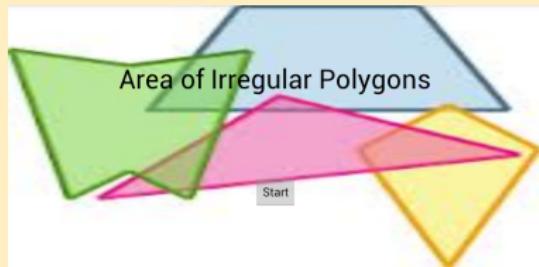
Converting input to decimal (base 10) :

Step 1 : 1 x (2⁰) +
Step 2 : 1 x (2¹) +
Step 3 : 0 x (2²) +
Step 4 : 1 x (2³) +
Step 5 : 1 x (2⁴) +

This application teaches Base Conversion, it describes the conversion procedure from any positive integral base to any other positive integral base

Educational Application

Area of Polygons



This application teaches methods to calculate area of any Polygon. Polygon is drawn by user input and area is calculated and displayed with explanation

Currency Converter

Euro:	5
Rupee:	392.5
US Dollar:	7.35
British Pound:	3.585
Japanese Yen:	825.96

Convert

This application provides the methods of currency conversion as well as the details of the country to which the currency belongs.

Enhancement of JMeter - Team Members



B. Sushmitha



Dhruv Joshi



Manisha Choudhury



Naman Choudhary



Shekhar Saurav



Surabhi Mour

