Regression Performance Testing with mBrace

Version 1.0
08 September 2013
Author: Michael Kok

Contents

1 Performance Regression Testing 3
2 Comparing one transaction type in two releases 3
3 Comparing all 49 transaction types 5
4 Viewing total impact on capacities at high volume 7
5 Viewing change over more releases 8
6 Implementing the mBrace Method for Performance Regression Testing 9

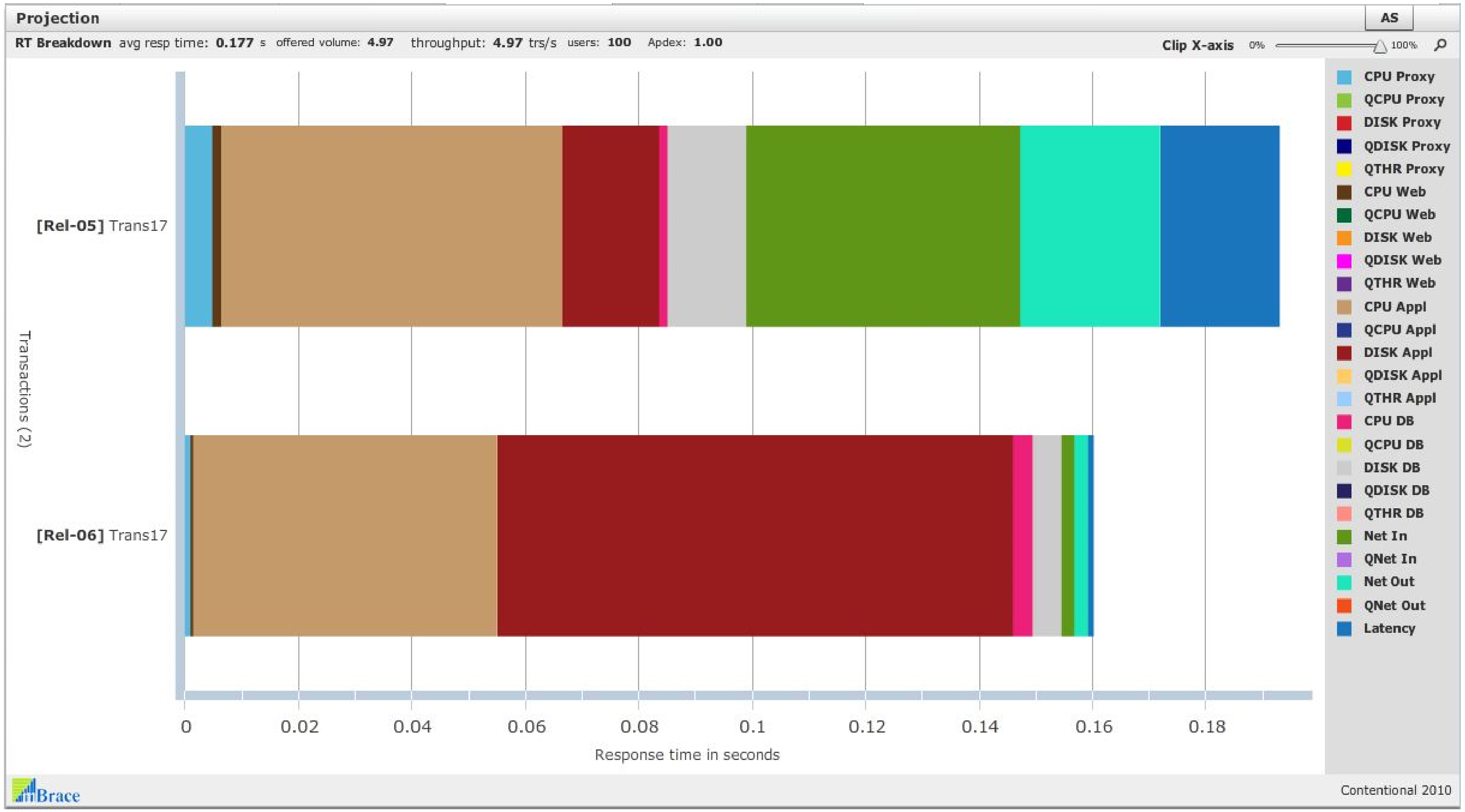
1 Performance Regression Testing

A regression test should be conducted at each new release of an application. Consequently the performance of the application should be subjected to a regression test to prevent poor performance. Commonly load tests are done in a production like acceptance test environment to the new release. For a large web site this can be an expensive job that must be repeated frequently. mBrace offers an attractive alternative. The application can be tested in a small test environment where measurements are taken, resulting in a complete picture of the Performance-DNA. This is fed into the mBrace model where one can compare the performance with the previous releases. Within minutes you have a complete overview of the performance anomalies of the new release. In the next sections a show case is presented based on a web application that is used at a high usage volume. 6 releases of the web application were measured and their performance-DNA's were captured and fed into the model. The application consists of over 40 transaction types and is operated on an extensive infrastructure chain with large capacities including Proxy server, Web server, Application server and Database server. The web site of this application serves over 7,000 concurrent users at peak times.

Screenshots of the mBrace model are used in the next sections to demonstrate Performance Regression Testing with mBrace.

2 Comparing one transaction type in two releases

The first figure compares the performance of 2 releases of only one transaction type. This is a simple first example that explains how this works with mBrace.


Projection

RT Breakdown avg resp time: 0.177 s offered volume: 4.97 throughput: 4.97 trs/s users: 100 Apdex: 1.00

Clip X-axis 0% 100%

Figure 1

Each of the two horizontal colored bars of Figure 1 represents the response time of one transaction type, Trans17. The colored parts represent a response time breakdown with respect to the time passed on each of the hardware resources of the infrastructure. The legend at the right explains the colors, e.g. light-brown represents the time spent for processing on the Appl server and red represents the time spent on disk IO on the Appl server. The upper bar shows the performance of Trans17 at Release 05 and the lower shows the same for Release 06. We can see that Release 05 has the longer response time. With Release 06 the response time has improved from 0.193 second to 0.160 second. We can also see that the time spent on disk IO (dark brown) on the application server has increased drastically while the time spent on the networks (two sorts of green and blue) has decreased significantly.
By comparing the patterns of the colored bars we can easily spot any anomaly and relate it to one or more parts of the response time.

3 Comparing all 49 transaction types

In this section all 49 transactions are compared for Release 05 and Release 06. A total of 98 transactions are included in one model of which 20 are shown in one window. The performance is shown when used by 7,200 users in total, equally divided over Release 05 and Release 06. Figure 2 shows one of the 5 windows that can show all of the transactions in the comparison. This window covers the transactions Trans21 through Trans32. The response times of all transactions have improved except for transaction Trans22, Trans29 and Trans32. Trans22 clearly is an anomaly. Instead of a decreasing, the response time drastically increased from 0.333 seconds to 1.121 seconds. This release of Trans22 displays excessive CPU usage and diskIO on the DB server. It is clear that something went wrong with the development of this transaction.

By scrolling through this window of the model while comparing the patterns of the coloured bars one can quickly inspect the new release completely.

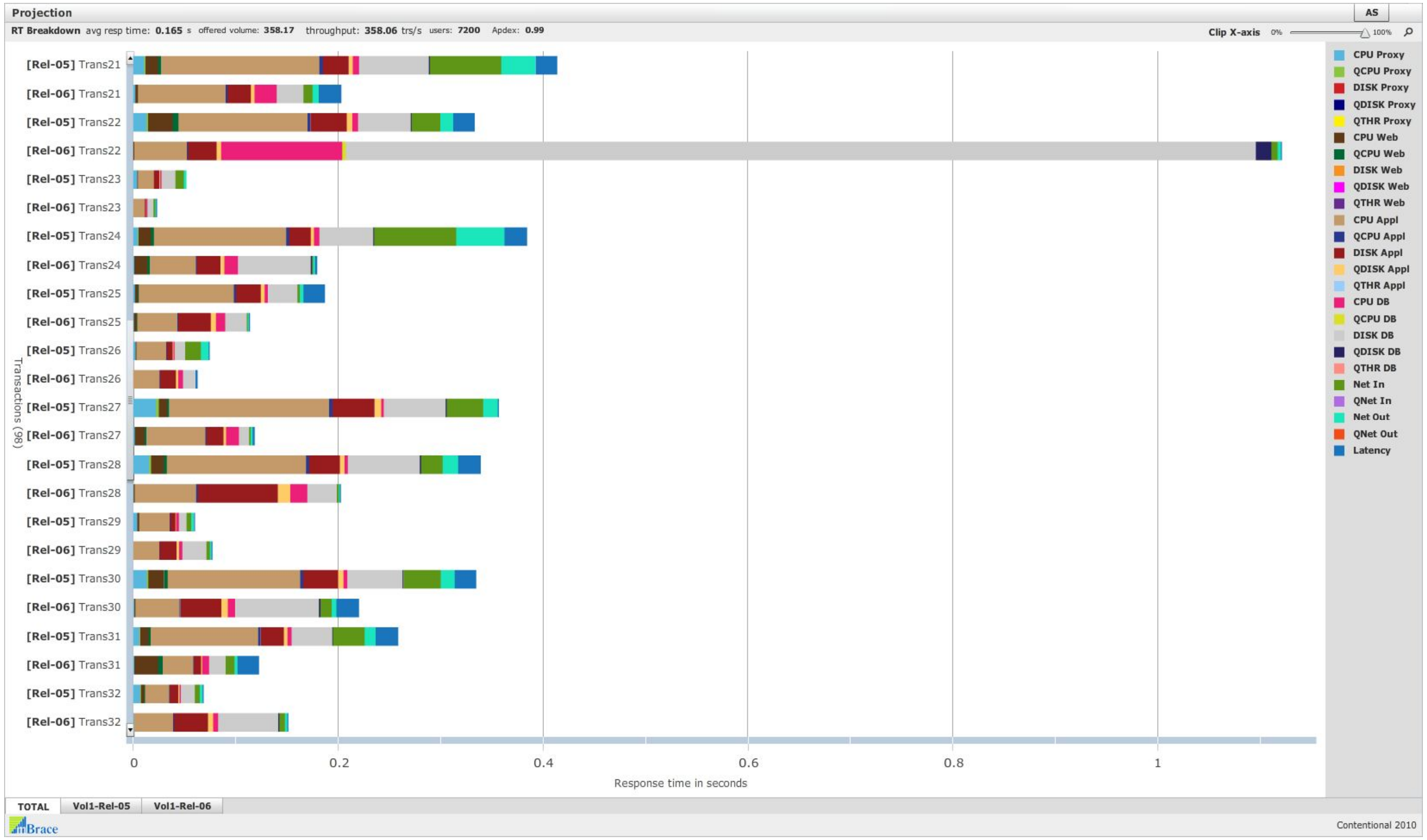
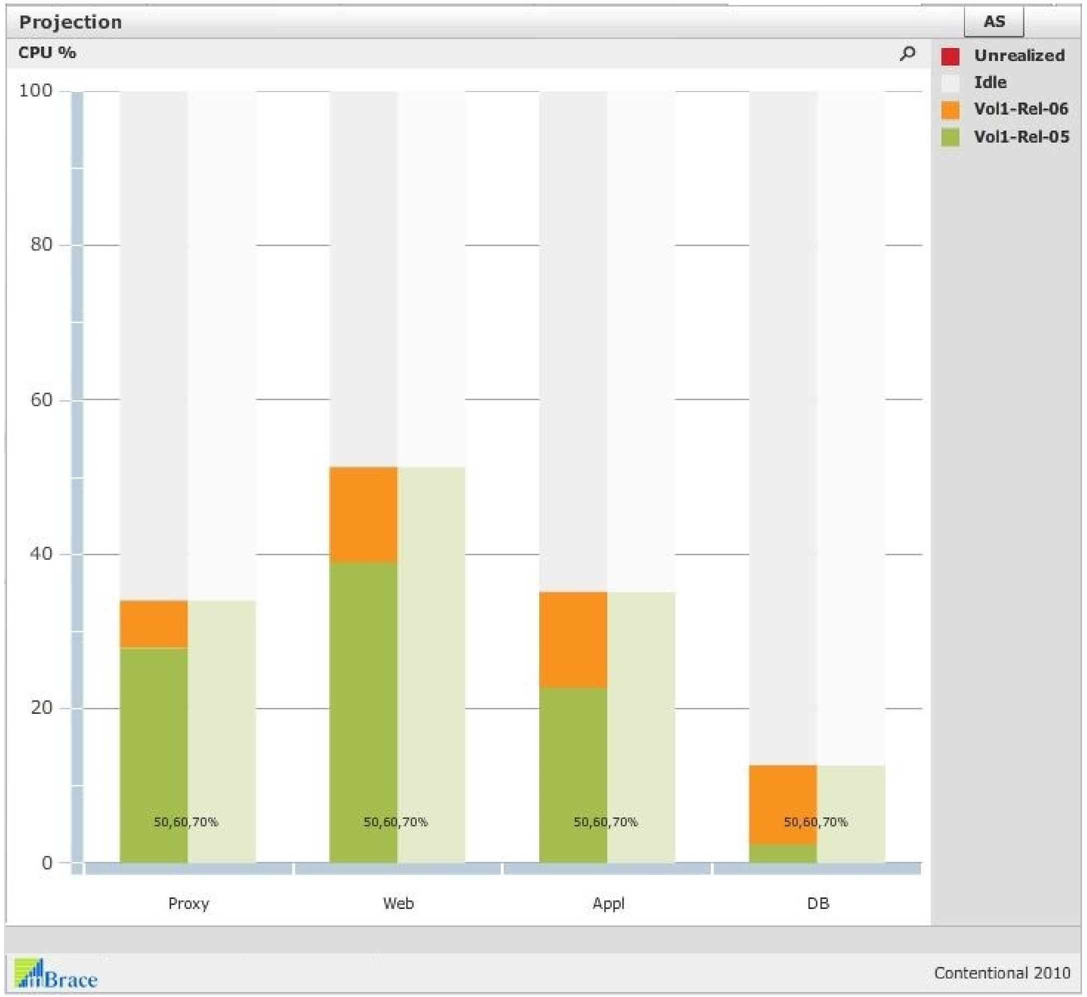

Projection
AS
RT Breakdown avg resp time: 0.165 s offered volume: 358.17 throughput: 358.06 trs/s users: 7200 Apdex: 0.99
Clip X-axis 0% 100%
[Rel-05] Trans21
[Rel-06] Trans21
[Rel-05] Trans22
[Rel-06] Trans22
[Rel-05] Trans23
[Rel-06] Trans23
[Rel-05] Trans24
[Rel-06] Trans24
[Rel-05] Trans25
[Rel-06] Trans25
[Rel-05] Trans26
[Rel-06] Trans26
[Rel-05] Trans27
[Rel-06] Trans27
[Rel-05] Trans28
[Rel-06] Trans28
[Rel-05] Trans29
[Rel-06] Trans29
[Rel-05] Trans30
[Rel-06] Trans30
[Rel-05] Trans31
[Rel-06] Trans31
[Rel-05] Trans32
[Rel-06] Trans32
Transactions (98)
0
0.2
0.4
0.6
0.8
1
Response time in seconds
CPU Proxy
QCPU Proxy
DISK Proxy
QDISK Proxy
QTHR Proxy
CPU Web
QCPU Web
DISK Web
QDISK Web
QTHR Web
CPU Appl
QCPU Appl
DISK Appl
QDISK Appl
QTHR Appl
CPU DB
QCPU DB
DISK DB
QDISK DB
QTHR DB
Net In
QNet In
Net Out
QNet Out
Latency
TOTAL
Vol1-Rel-05
Vol1-Rel-06
Brace
Contentional 2010

Figure 2

4 Viewing total impact on capacities at high volume



Sometimes it is not immediately clear whether a change has much impact. A small change in response time may not hurt the user, but what is the impact at high volume? What does it mean for the capacities of the servers? Figure 3 compares the capacity demands on the CPU's of each of the servers at the usage volume of 7,200 concurrent users.

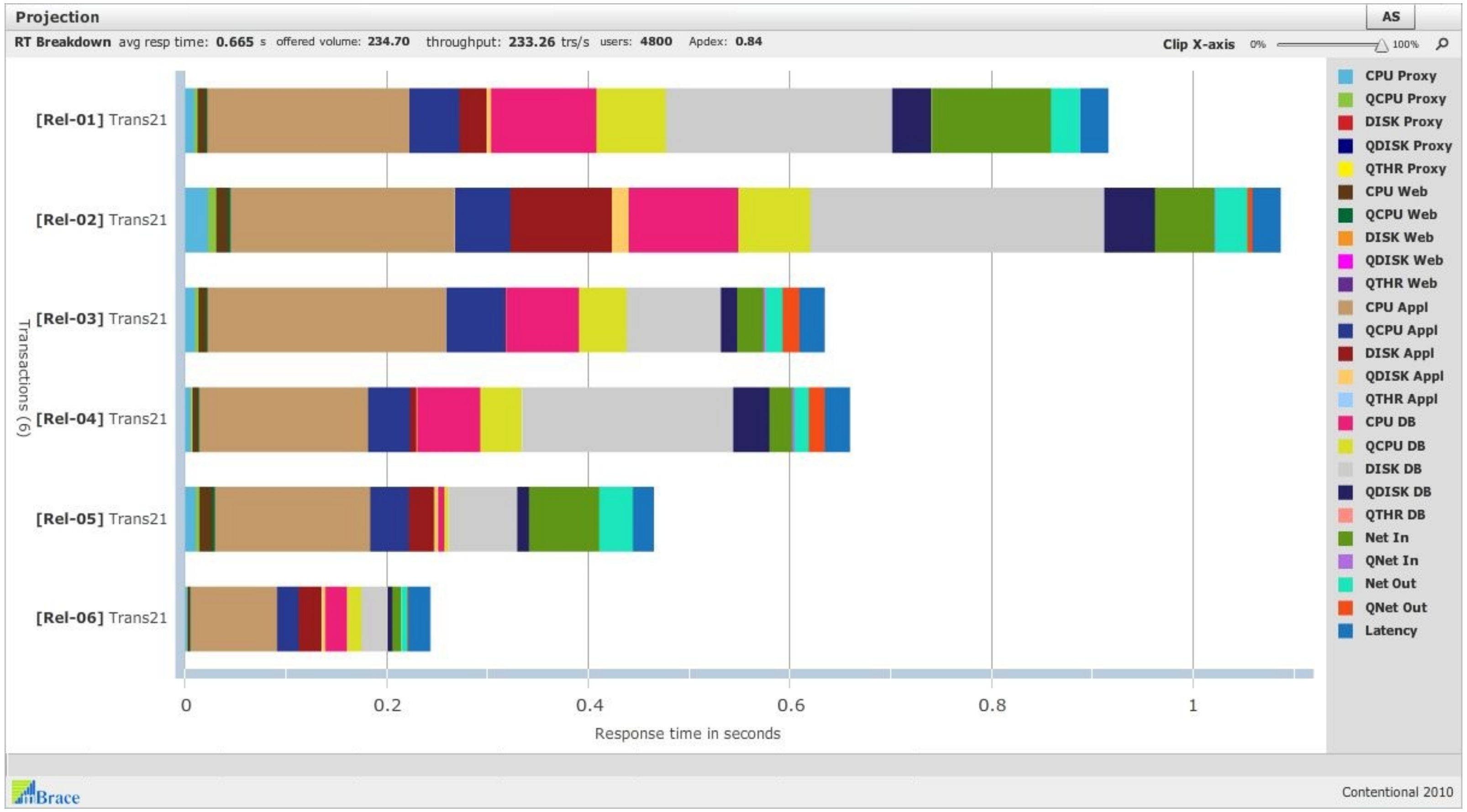
The figure presents a vertical coloured bar for the CPU's of each of the servers. The colours correspond with the releases, green for Releases 05 and orange for Release 06. Release 6 clearly has much lower capacity demands on all servers except de database server.

Notice that the capacity demands are shown by the utilisation percentage on the resource. One needs to know that the various resources do not necessarily have the same capacities. E.g. the Proxy server has 4 CPU's while the Application server consists of a cluster of 12 servers, each with 4 CPU's. Only a small fraction of that will be needed for Release 06.

Figure 3

5 Viewing change over more releases

Next figure shows how the response time of Trans21 evolved over six releases. Response time gradually decreased from 0.91 seconds to 0.22 seconds.



Projection

RT Breakdown avg resp time: 0.665 s offered volume: 234.70 throughput: 233.26 trs/s users: 4800 Apdex: 0.84

Figure 4

6 Implementing the mBrace Method for Performance Regression Testing

The process of measuring each new release can be easily implemented. One or two testers can conduct the measurements, analyze the results and complete the regression test within two days for successive releases.