TAPAS

IST-2001-34069

Trusted and QoS-Aware Provision of Application Services

QoS-aware Application Server:
Experimental Evaluation of the Resource Utilization

Report Version: D11 supplement

Report Delivery Date: 31 March, 2005

Classification: Public Circulation

Contract Start Date: 1 April 2002 **Duration:** 36m

Project Co-ordinator: Newcastle University

Partners: Adesso, Dortmund – Germany; University College London – UK; University of Bologna – Italy; University of Cambridge – UK

Project funded by the European Community under the "Information Society Technology" Programme (1998-2002)

QoS-aware Application Server:

Experimental Evaluation of the Resource Utilization

Giorgia Lodi, Fabio Panzieri, Davide Rossi, Elisa Turrini

University of Bologna

Department of Computer Science

Mura A. Zamboni 7

I – 40127 Bologna

{lodig|panzieri|rossi|turrini}@cs.unibo.it

This Report supplements the D11 deliverable Report with additional experimental results, obtained from the evaluation of our implementation of the Configuration, Monitoring, and Load Balancing Services described in D11.

This additional evaluation has been carried out in order to i) assess the average resource utilization entailed by the use of the above mentioned Services, and ii) compare and contrast it with that of the standard JBoss application server.

For our experimental evaluation we have used the same infrastructure as that introduced in D11. It consisted of 5 application server instances running in 5 dedicated Linux machines, interconnected by a 1Gb Ethernet LAN. Each machine was a 2.66Ghz dual Intel Xeon processor, equipped with a 2GB RAM. In the experiments described below one of these machines was dedicated to implement the Load Balancing Service; the other four machines were used to host application server instances that could serve the client requests. The client program was implemented by the JMeter program running in a G4 processor under Mac OS X; this client machine was connected to the clustered servers via a 100Mb Ethernet subnet.

The clustered servers were hosting (homogeneously) the digital bookshop application mentioned in D11. This application provides its clients with such catalog operations as “choose_book”, “add_book_to_cart”, “remove_book_from_cart”, “buy_book”, “confirm_order”, which access and manipulate an application database. As the principal scope of our evaluation was to assess the performance of our middleware services, only, this application database was replicated and instantiated locally to each application server, in order to avoid that it become a bottleneck (issues of database consistency have been ignored, for the scopes of this evaluation).

For the purposes of this evaluation, we have assumed that an ASP, offering standard JBoss hosting services, adopt a *resource overprovision policy* in order to ensure that the application hosting SLA be honoured. We wish to show that, in contrast with this policy, the resource utilization can vary dynamically as needed, without causing hosting SLA violations, if JBoss extended with our TAPAS middleware services is used (i.e., we wish to show that the use of our services optimizes the use of the clustered resources).

In addition, we have assumed that the hosting SLA prescribe a maximum response time of 1s per catalog operation, and a 20 request per second (rps) minimal throughput. Thus, the following two principal *warning points* were set by the Configuration Service at cluster configuration time:

- a High Load warning point HL = 90%, indicating that when either the response time or the throughput reach this percentage of their relative SLA values (i.e. their *breaching points*), dynamic cluster reconfiguration is required, and a new node is to be *added* to the current cluster;
- a Low Load warning point LL = 30%, indicating that when either the response time or the throughput reach this percentage of their relative SLA values, dynamic cluster reconfiguration is required, and resources (i.e., nodes) can be *released* from the cluster as they are no longer necessary.

Based on the above assumptions, we developed a number of experiments in order to carry out our evaluation; each experiment consisted of a set of different tests, summarised below. In the first test, 10 clients were invoking concurrently the entire sequence of the digital bookshop operations, for 5 minutes. We run this test for 40 minutes, obtaining a sequence of 5 minutes snapshots, such as that depicted in Figure 1 below.

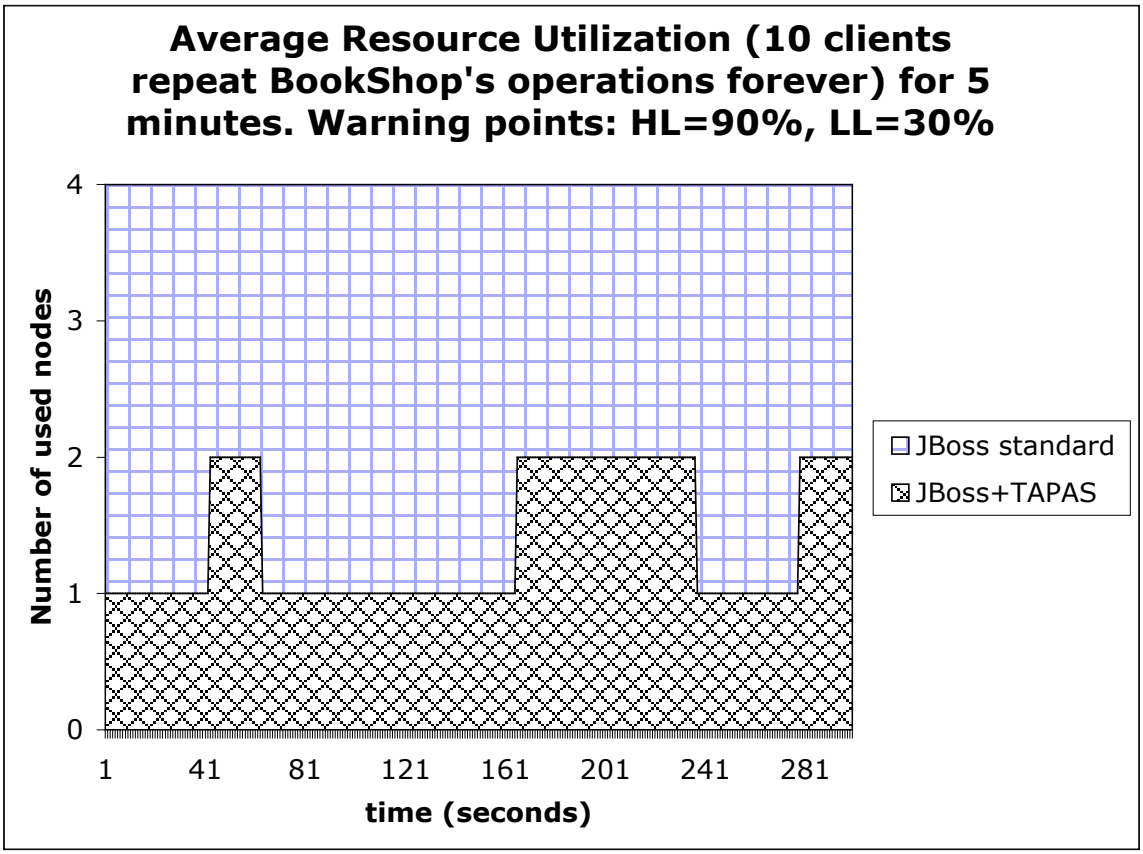

Figure 1: 1st test

Figure 1 shows that the number of used nodes in the cluster did not change for the standard JBoss configuration, as no dynamic cluster reconfiguration is provided by it. Thus, in order to meet the above mentioned SLA, the standard JBoss has to adopt resource overprovision, and allocate at least (and possibly 3) nodes to the hosted application for the entire duration of the test. In contrast, the TAPAS extended JBoss (JBoss+TAPAS, in Figure 1) dynamically uses from 1 to 2 nodes, only, depending on the load in different time intervals.

In the second test, the number of clients was augmented to 30. As in the previous test, each client invoked the entire sequence of bookshop application operations for 5 minutes. We run this test for 40 minutes, and observed a sequence of 5 minutes snapshots such as that depicted in Figure 2.

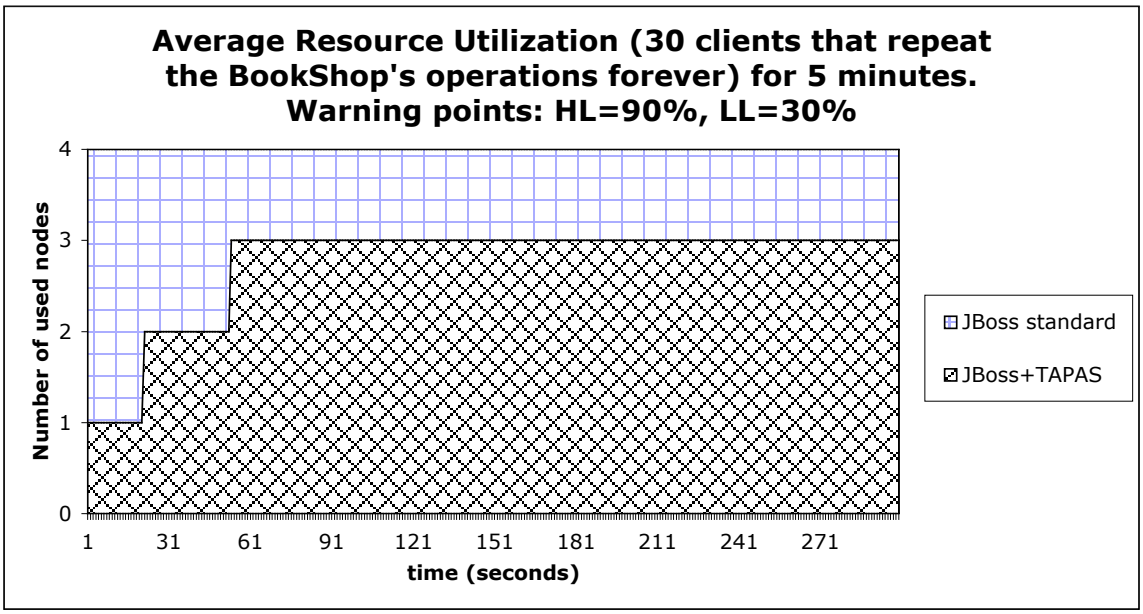

Figure 2: 2nd test

Figure 2 shows that the standard JBoss configuration, in order to meet the hosting SLA, must allocate at least 3 (and possibly 4) available nodes, regardless of the actual load. In contrast, the TAPAS extended JBoss starts with a small number of nodes (i.e., 1 node), and increases it (up to 3 nodes), as the load increases.

The third test consisted of 50 clients, each of which invoked the entire sequence of bookshop operations for 5 minutes. This test run for approximately 40 minutes; a 5 minutes snapshot of this test is illustrated in Figure 3. This Figure shows that the standard JBoss statically allocates the 4 available clustered nodes, as usual, since the beginning of the test, even though not all these nodes are always necessary. In contrast, the TAPAS extended JBoss increases dynamically the number of nodes in the cluster, as the load increases, and begins to release resources (i.e., nodes) as the load decreases (i.e., as some of the nodes become unnecessary).

In order to investigate further the effectiveness of this latter feature implemented by our services (i.e., the release of unnecessary resources, as load decreases), we carried out the two tests described below. In both these tests, initially 10 clients issued concurrently the entire sequence of bookshop operations. After a fixed time interval (i.e., a period of 2' and 7', respectively, in each test), the number of clients increased to 30, and then, after the same period in each test, it increased to 50 (needless to say, the clients always issued the entire sequence of bookshop operations). Then, the client load decreases, firstly from 50 to 30 clients; then, from 30 to 10 clients.

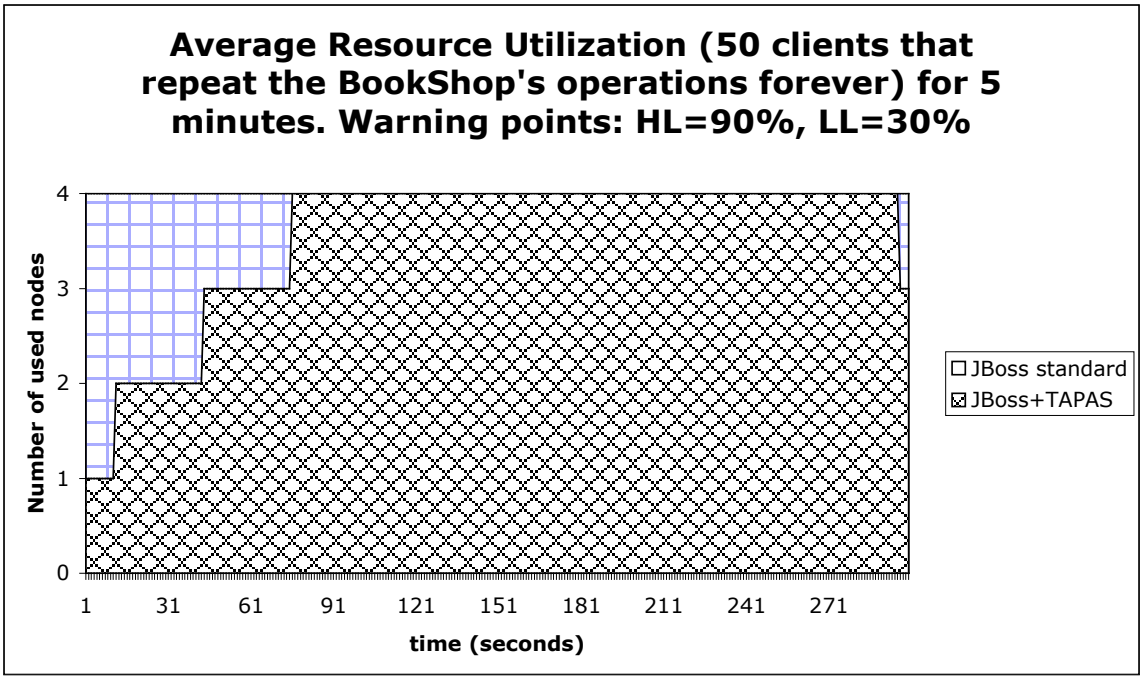

Figure 3: 3rd test

The results of these tests are shown in the Figures 4 and 5, below; in both these Figures, the client load distribution is depicted by the bold line. Figure 4 shows a 10' snapshot of a 90' test in which the client load varied with a 2' period. This figure shows that the standard JBoss allocates all the 4 available nodes, and maintains them allocated to the hosted application for the entire duration of the test, regardless of the actual client load. In contrast, the TAPAS extended JBoss dynamically augments the cluster size, as the client load increases, and dynamically releases clustered nodes, as the client load decreases.

Figure 5 shows a 35' snapshot of a 140' test in which the client load varied with a 7' period. It can be seen that the results shown in this Figure are coherent with those discussed above, and illustrated in Figure 4. Namely, Figure 5 shows that, yet again, the TAPAS extended JBoss optimizes resource allocation by dynamically adding resources to the cluster when necessary, and releasing those resources, as they become unnecessary. In contrast, the standard JBoss cluster configuration needs resource overprovision, in order to guarantee that the SLA be honoured.

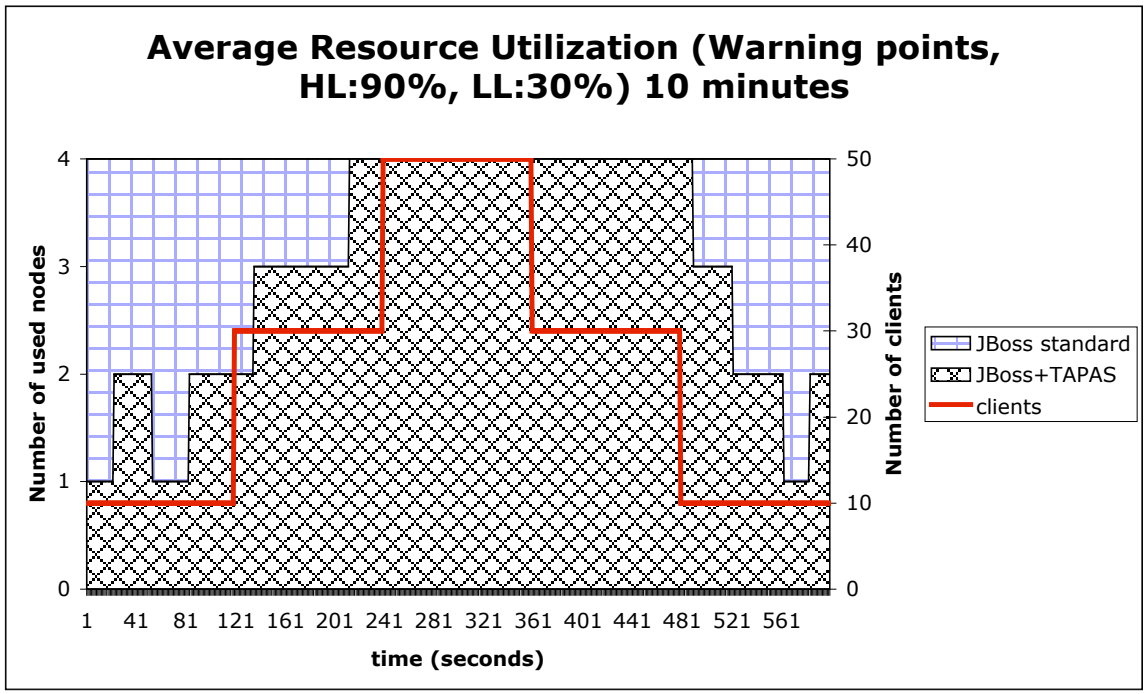

Figure 4: 4th test

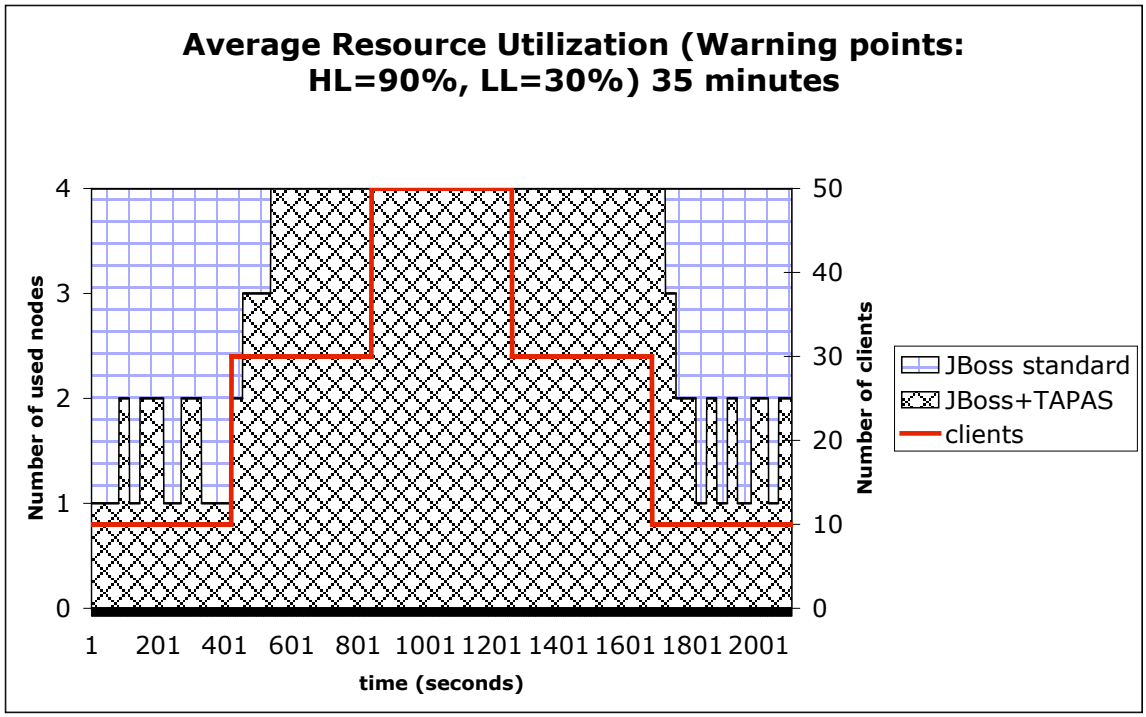

Figure 5: 5th test

In summary, the last two tests above show that the TAPAS middleware uses, on average, from 2 to 3 nodes, by saving approximately 30% of the available resources, when compared to the standard JBoss; Figures 6 and 7 illustrate this results.

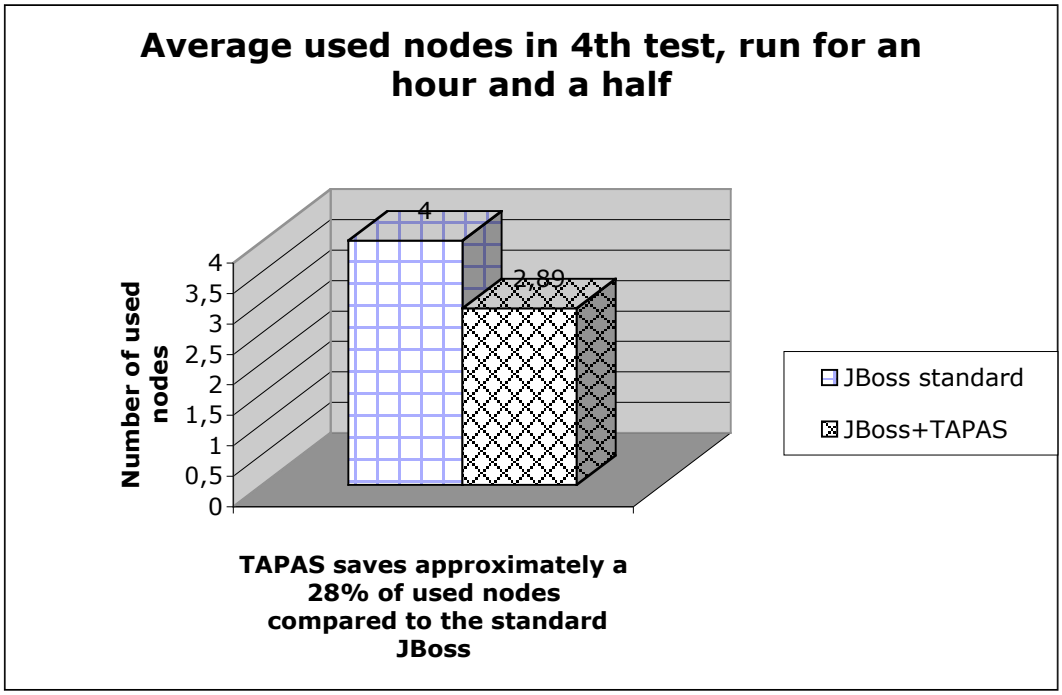

Figure 6: Average number of nodes in 4th test

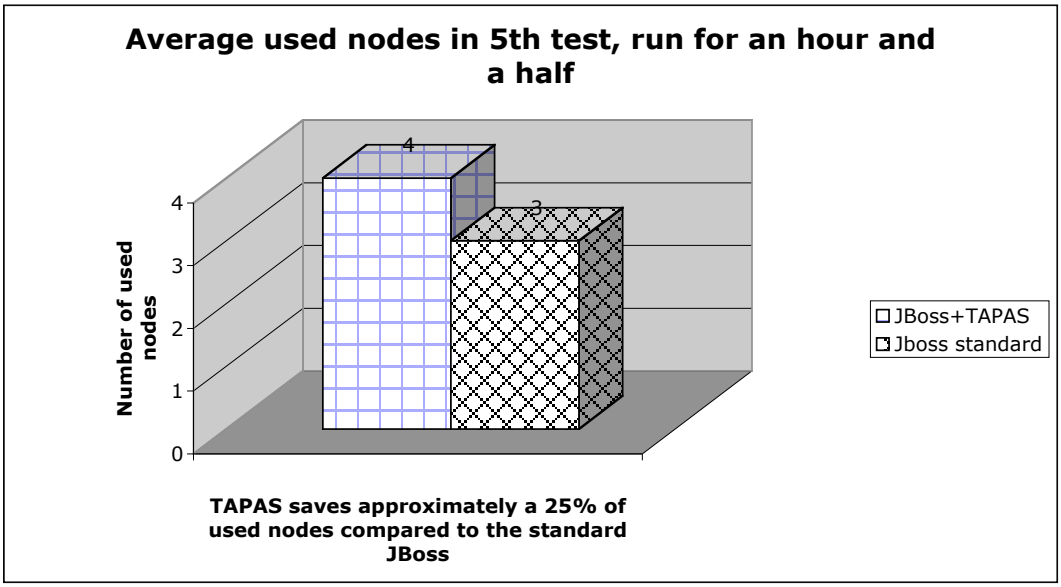

Figure 7: Average number of nodes in 5th test

The following experiment aims at evaluating the load balancing mechanisms implemented by our TAPAS extended JBoss. Specifically, this experiment assesses the *number of used nodes* when *round robin* load balancing is implemented by our TAPAS extended JBoss. The result is compared and contrasted with that obtained when *adaptive* load balancing is used, instead.

This experiment is based on the assumptions, concerning the SLA and the warning points, introduced earlier, and has consisted of two tests. In the first test we compared and contrasted the round robin load balancing strategy with the adaptive strategy, by running our 5th test, previously described (i.e., from 10 to 50 clients accessing the electronic bookshop, with a 7' period of load variation). The results of this test, for the round robin policy, are depicted in the earlier Figure 5. Figure 9 below shows the results we have obtained by using our adaptive load balancing policy (described in D11).

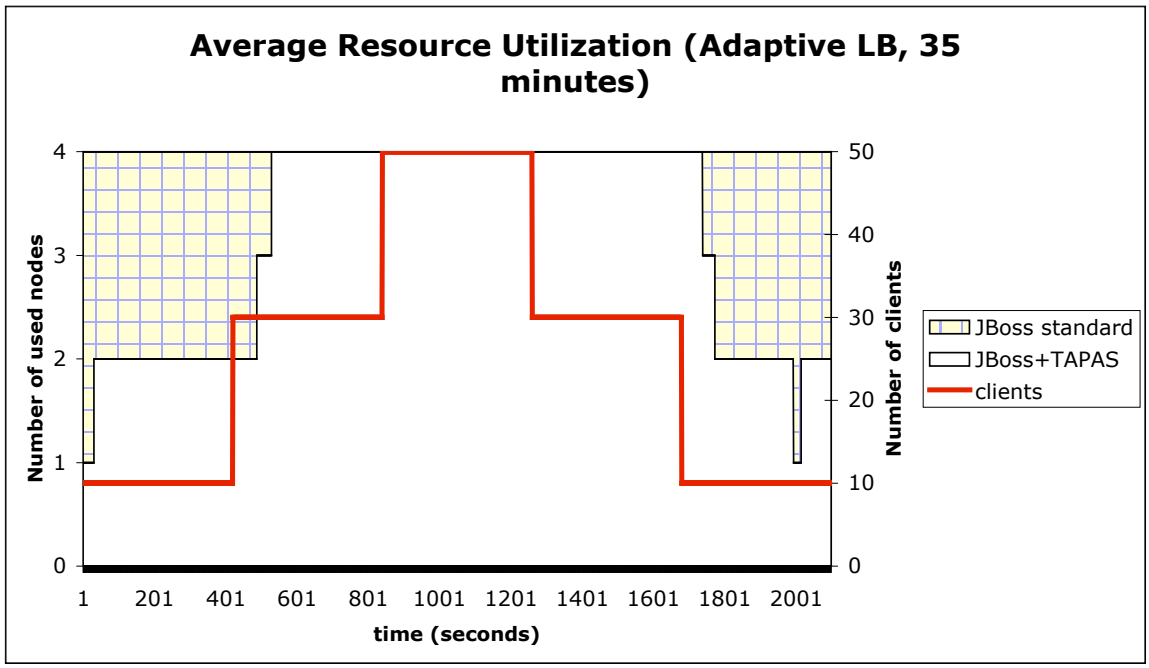

Figure 9: Adaptive load balancing results

Figures 5 and 9 show that the average number of used nodes is approximately the same by applying either strategies. Hence, in order to assess the effectiveness of our adaptive strategy, we injected additional load in one of the clustered nodes (as our adaptive load balancing operates based on the actual load of each node).

Thus, we carried out a second test. The test case was the same as that introduced above, with the exception that, on one of the clustered nodes, additional load was injected by 5 clients using a computational intensive application running in that particular node. The resulting average resource utilization is shown in the Figures 10 and 11.

Figure 10 shows that the round robin policy in our TAPAS extended JBoss may cause poor clustered resource allocation. After allocating all the available resources, the number of allocated nodes is maintained the same as that used in case of maximum, even though the load decreases. We have observed that this occurs because the response time (not shown in Figure 10) experienced by the clients never falls below the LL warning point (the 30% of the SLA specified response time).

In contrast, Figure 11 shows that the use of our adaptive strategy, in order to balance the client load in a cluster of TAPAS extended JBoss nodes, optimises the allocation of those nodes. In fact we have observed that, using adaptive load

balancing, the response time decreased (below the LL threshold) as the load decreased; hence, resources were released, as no longer necessary.

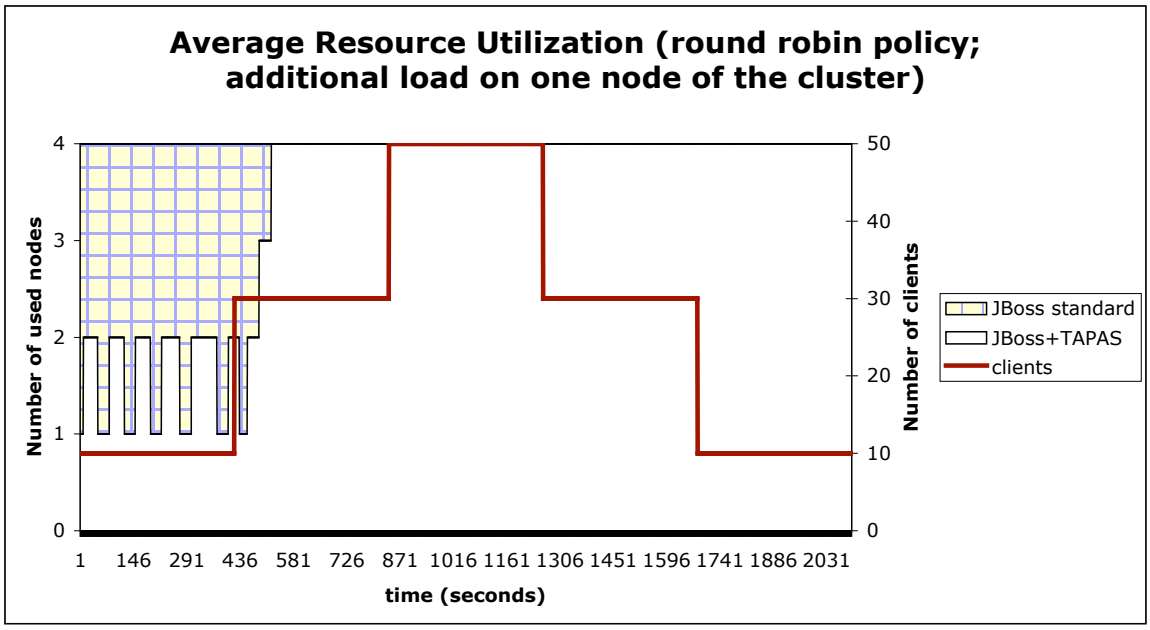

Figure 10: Round Robin strategy with additional load on a node.

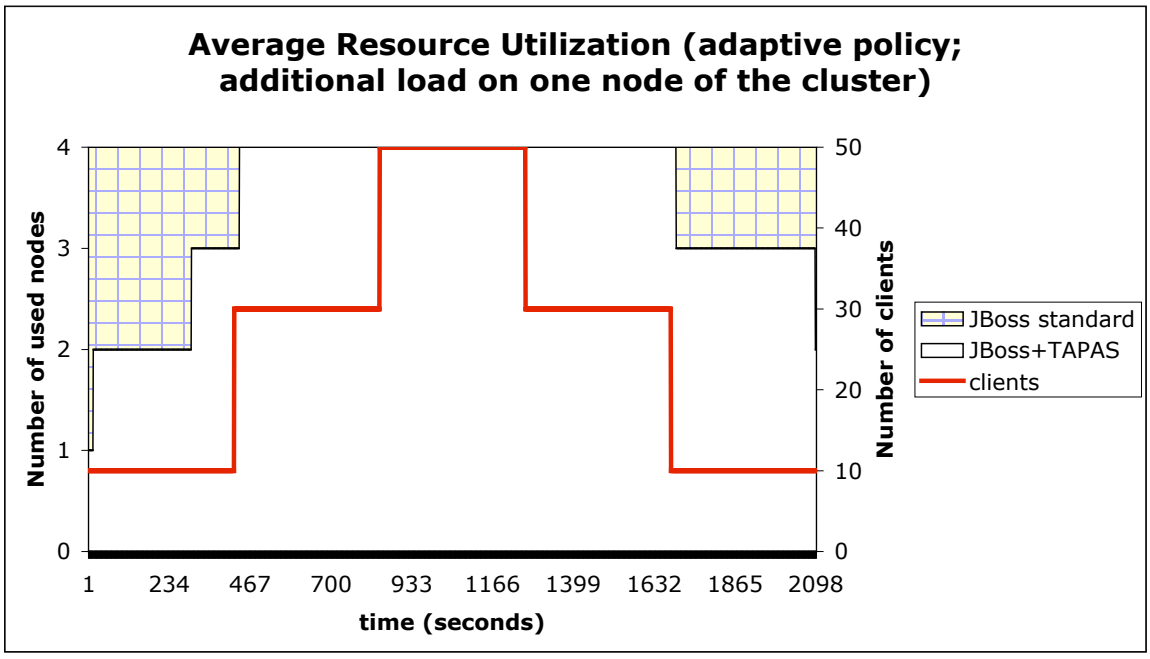

Figure 11: Adaptive strategy with additional load on a node.

The last experiment was carried out in order to assess the effect of the round robin and the adaptive load balancing strategies on the cluster response time and throughput. The cluster configuration for this experiment consisted of 3 nodes, only, implementing our TAPAS extended JBoss; specifically, 2 nodes were serving the client requests, and a node was used as dedicated load balancer. In addition, we injected, during the test, the additional load generated by 10 clients accessing the

computationally intensive application mentioned earlier in one of the two nodes serving the client requests.

The results of this experiment are depicted in the Figures 12 and 13. These Figures show that both the response time and the throughput of our clustered servers can improve (of 20%, approximately) if adaptive, rather than round robin, load balancing is used within the cluster.

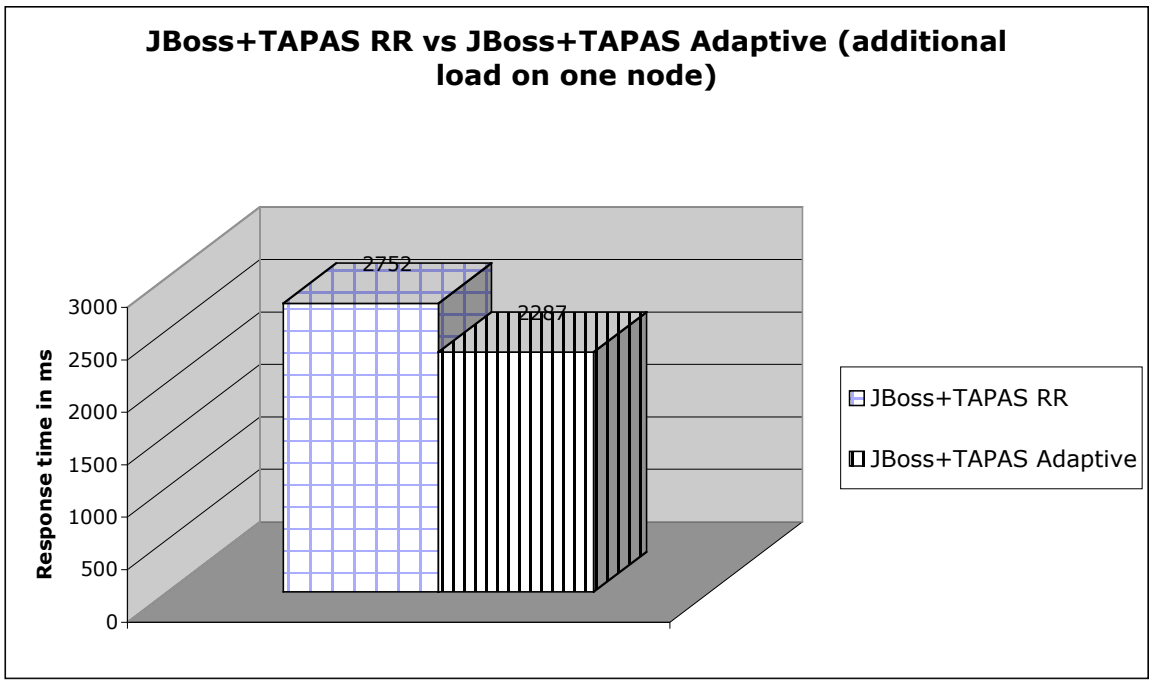

Figure 12: TAPAS Round Robin vs Adaptive: Response Time

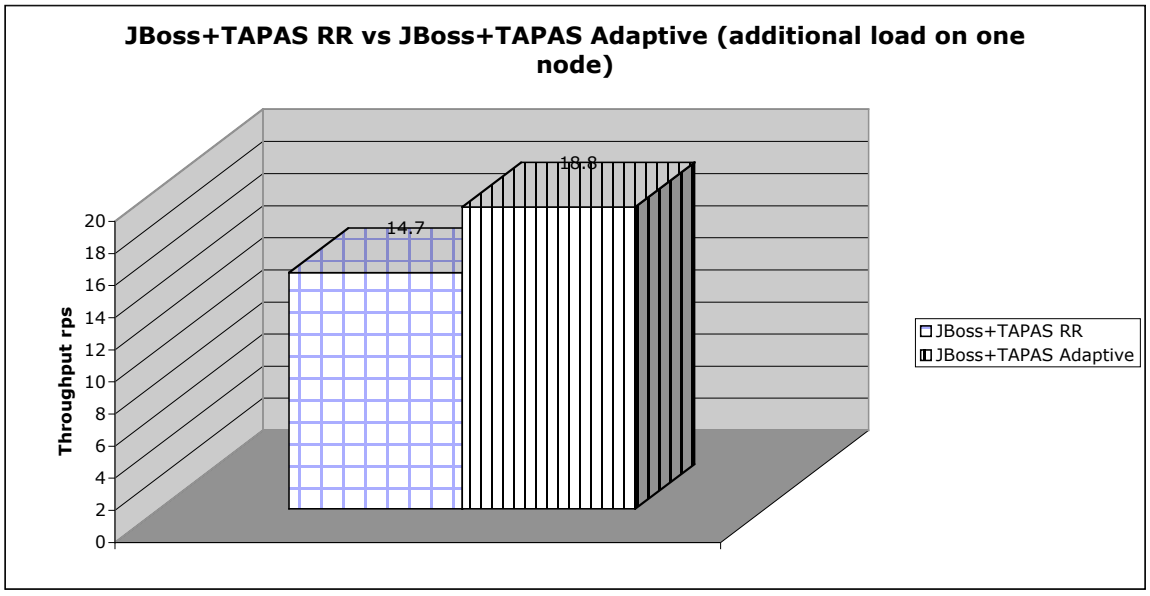

Figure 13: TAPAS Round Robin vs TAPAS Adaptive: Throughput