Performance Measurements of CEMon on an OSG Test Environment

Dec 14, 2006

Karthikeyan Arunachalam (University of Oklahoma)
Gabriele Garzolio (Fermilab)

Introduction

The primary objective for this study is to check the impact of deploying CEMon on OSG. This is done by studying machine parameters, such as load and memory usage, when running CEMon and GRIS, the LDAP-based Globus MDS v2 monitoring server.

CEMon and GRIS are two independent information systems that gather local information running a set of scripts called Generic Information Providers (GIP). Because the systems use two different publication infrastructures, they are both needed at this time. The LDAP server publishes information in LDIF format upon request (pull-model), while CEMon for OSG pushes information in classad format to a central information repository.

This study addresses some concerns reported by OSG site administrators in running CEmon on a typical OSG installation. Concerns include high load to the machine and memory usage: high load could be generated by calling the GIP scripts twice, by GRIS and CEMon; high memory usage could result from CEMon, which runs Web Services interfaces in a Tomcat engine.

Summary of conclusions:

We verified that running CEMon and GRIS together does not significantly impact the average load of the machine. We measured a load increase of 10% with respect to running GRIS only. Also, we have concluded that running CEMon alone uses more memory than GRIS, but does not generate a higher average load to the machine.

Test conditions:

We ran tests of the monitoring services under two conditions:

The first condition (round 1) simulates a "busy" OSG environment that interoperates with LCG. GRIS is queried continuously to simulate connections from about 100 LCG brokers. GRIS Caching parameters are configured as follows: freshness = 300; cache_ttl = 600. CEMon publishes information to a central resource selection server every 10 minute.

The second condition (round 2) is useful to compare monitoring technologies, as the different monitoring servers are configured to publish information at the same rate (once every 10 minutes, like cemon by default). CEMon test conditions are the same as for round 1.

Content

In **section (A)** of this report we describe the test environment. This includes information about the server, client and data collection.
Section (B) explains the test rounds and test cases, including the starting environment of the tests and background load.
Section (C) and **(D)** present test results for test round 1 and 2 respectively.
Section (E) deals with the details of the background load measurements.
Section (F) provides a conclusion for the report
Acknowledgements are contained in **section (G)** and appendix containing the data plots in **section (H)**.

A) Test environment

The test bed consists of two machines. The server machine runs the monitoring services; the client machine runs the clients to the GRIS monitoring service. The duration of each test case is 1 hour. Machine parameters (memory, load, etc.) are acquired every 10 seconds using Linux command line tools, such as 'uptime', 'ps', and 'top'.

These are the specification of the machines and the software installed.

Sever: ouhep1
Dual processor
Model name : Pentium III (Coppermine)
Cpu MHz : 996.593
Cache size : 256 KB
Memory : 1 GB
OS : Scientific Linux Release 3.0.4 (Fermi)

Client: ouhep5
Dual processor
Model name : Pentium III (Coppermine)
Cpu MHz : 996.578
Memory : 1 GB
OS : Scientific Linux Release 3.0.4 (Fermi)

Software configuration:
OSG version: 0.5.1
The jobmanager scripts used by GIP is condor.
GIP is NOT configured to publish SRM

B) Description of the tests

Two test conditions (rounds) were created and measured.

Test round 1

Objective: measure machine parameters when running monitoring servers on a "busy" OSG environment.

A total of 15 test runs were conducted. Each test run consisted of 4 test cases as explained below. Each test case was conducted for a duration of 1 hour, with data measurements taken every 10 seconds (360 per hour).

Test cases for Test round 1

1) Run CEMon only. CEMon publishes information in old classad format (OLD_CLASSAD dialect from the OSG_CE sensor) to the information repository of ReSS, the OSG Resource Selection Service. More information on the architecture of ReSS at http://osg.ivdgl.org/twiki/bin/view/ResourceSelection/
2) Run GIP by hand only. GIP commands are run continuously i.e. as soon as one command finishes, the same command is executed again.
3) Run GRIS only (run GRIS on server and ldapsearch on client). ldapsearch is run continuously from the client to simulate LCG brokers querying the OSG monitoring system.
4) Run Both CEMon and GRIS (run GRIS,CEMon on server and ldapsearch on client)

Test round 2

Objective: compare CEMon characteristics with LDAP server when publishing information at the same rate.

A total of 2 test runs were conducted, each test run consisting of 3 test cases. Each test case was conducted for a duration of 1 hour with data measurements taken every 10 seconds (360 per hour).

Test cases for Test round 2

1) Run CEMon only. These measurements are the same as in round 1
2) Run GIP by hand only. GIP commands are run once every 10 minutes .
3) Run GRIS only (run GRIS on server and ldapsearch on client). ldapsearch is run once every 10 minutes from the client machine.
4) Run Both CEMon and GRIS (run GRIS,CEMon on server and ldapsearch on client)

Starting condition for both test rounds and for all test cases

Server running : xinted, condor_collecor/negotiator, globus GK, NIS, NFS
Server NOT running: CEMon, LDAP server (i.e. GRIS or GRIS), condor jobs

These processes account for the background load - see section (E) for more details on background load.

C) Test results for round 1: simulation of an OSG site

This section presents measurements of load, percentage of CPU and memory usage for all four test cases.

1) Machine Load

The three subsequent numbers reported below refer to the load averaged over 1, 5, 15 minutes, as presented by 'uptime'. Note that the max value presented is mostly affected by the background.

Table 1

Test Case	**Description**	**Number of Measurements (Taken once every 10 seconds)**	**Average of system load averages for the past 'N' minutes**			**Maximum of system load averages for the past 'N' minutes**		
			N=1	**N=5**	**N=15**	**N=1**	**N=5**	**N=15**
1.	Run CEMon only	4785	0.53	0.55	0.58	8.07	2.94	1.67
2.	Run GIP only, continuously*	4686	1.83	1.74	1.47	8.65	4.78	3.49
3.	Run GRIS only, continuously	4772	1.14	1.18	1.28	7.20	4.96	3.62
4.	Run Cemon and Run GRIS continuously	4730	1.20	1.19	1.17	9.98	5.67	4.33

* The next instance of the process is started immediately after the current instance is finished.

2) %CPU and %MEM Consumption

These averages per run were calculated by measuring the load using the command 'top' once every 10 seconds. Note that the max value presented is mostly affected by the background.

Test case 1 - Run CEMon only

Average %CPU 0.806667
Average %MEM 4.68667
Max %CPU 0.9
Max %MEM 5.1

Test case 2 - Run GIP by hand only

These measurement were gathered by running GIP command continuously for 15 hours. During this time period, the command was run 759 times. The average values calculated this way is close to the average values measured when running the command once.

Average time per run: 71.15 seconds
Average %CPU (4686 total measurements using ps auxww): 92.1992
Max %CPU (4686 total measurements using ps auxww): 99.99
Average %MEM (4686 total measurements using ps auxww): 0.4

Test case 3 – run GRIS only

Average %CPU 24.76
Average %MEM 0.56
Max %CPU Max 25.5
Max %MEM Max 0.6

Test case 4 - run both CEMon and GRIS

CEMon:
Average %CPU 0.826667
Average %MEM 4.68667
Max %CPU 0.9
Max %MEM 5.1

GRIS:
Average %CPU 24.5467
Average %MEM 0.546667
Max %CPU 25.6
Max %MEM 0.6

D) Test results for round 2 - Comparison of monitoring system

As discussed in section B, under these test conditions servers gather information every 10 minutes. These measurements are mostly useful to compare the performance of different monitoring technologies, because all servers are configured to publish information at the same rate.

Table 2

Test Case	Description	Number of Measurements (Taken once every 10 seconds)	Average of system load averages for the past 'N' minutes			Maximum of system load averages for the past 'N' minutes		
			N=1	N=5	N=15	N=1	N=5	N=15
1.	Run CEMon only *	4785	0.53	0.55	0.58	8.07	2.94	1.67
2.	Run GIP only, by hand every 10 minutes	1432	0.56	0.54	0.53	4.92	1.66	1.00
3.	Run GRIS only, by hand every 10 minutes	1432	0.52	0.52	0.51	4.76	1.90	1.15
4.	Run CEMon. Run GRIS, by hand every 10 minutes	1432	0.60	0.58	0.51	5.63	2.61	1.51

* Data for CEMon is the same as that of round 1: no new measurements were taken.

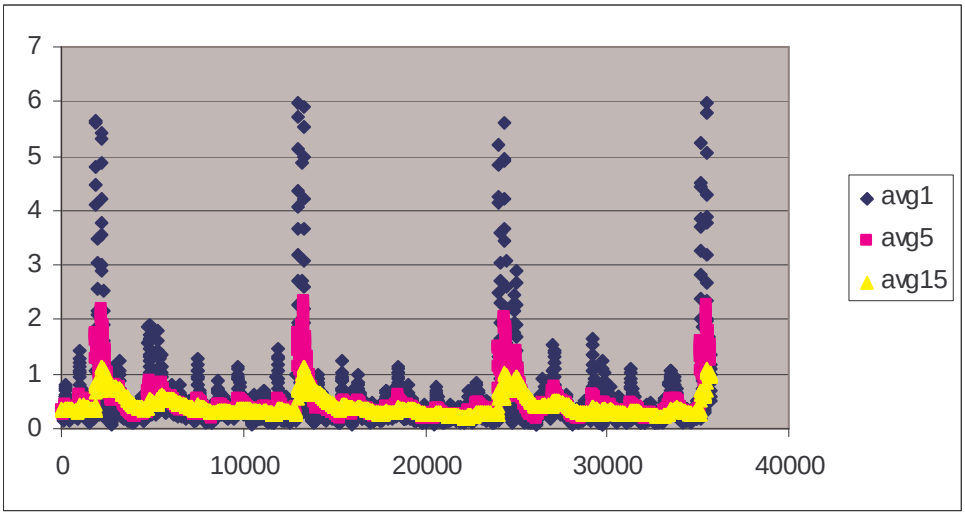
E) Background Load

We call 'background load' the load of the server machine during day to day operations. This is the list of processes running on the machine.

Processes running : xinted, condor_collecor/negotiator, globus GK, NIS, NFS, GridCat jobs
Processes NOT running: CEMon, LDAP server (i.e. GRIS or BDII), condor jobs

The average of the background load was calculated using 3574 measurements.
Background load averages (averages of 1 min, 5 min, and 15 min average):
0.46 0.44 0.41

Background data plot: (3574 measurements)



A regular pattern of spikes were noted for the background load. These spikes might be related to GridCat jobs running on the machine via the gatekeeper. Because of the predominance of these spikes, we believe that subtracting the background from the measurement traces does not lead to reliable results.

F) Conclusions

1) The average load from the round 2 test cases 1,2,3 are within 10% of one another. Since these represent the average load to the machine when CEMon, GRIS, and GIP gather monitoring data at the same frequency, we conclude that running CEMon alone does not generate more load than the other services.

2) Running CEMon on the same machine where GRIS runs does not significantly increase the machine load. When continuously queried, running GRIS alone contributes to an average load of 1.1. The load when running the services together is 1.2.

3) The average load to the machine is smaller when running CEMon alone (avg. 0.5) than when running a GRIS that is queried continuously (avg. 1.1). Both servers generate lesser load than when running GIP by hand continuously (avg. 1.8): this is expected because the both servers cache data.

4) CEMon uses less %CPU than a GRIS that is queried continuously (0.8% vs. 24%). On the other hand, CEMon uses more memory (%4.7 vs. %0.5). This is not surprising because CEMon is run within Tomcat offering web services interfaces. These technologies are well known to be memory intensive.

G) Acknowledgments

Gabriele Garzolio (Fermilab)
Provided the motivation to conduct the test cases, defined what to measure and provided the overall test guidelines and clarifications.

Karthikeyan Arunachalam (University of Oklahoma)
Created the testing scripts, conducted the tests and analyzed the test results

Horst Severini (University of Oklahoma)
Provided the overall test environment. Participate with helpful insights during the tests and analysis of test results.

Tanya Levshina (Fermilab)
- Provided help with CEMon and GIP.

Joel Snow (Langston University)
- Provided guidelines for background load measurements.

Steven Timm (Fermilab)
- Provided feedback during the analysis of test results.

Pat Skubic, Mike Strauss (University of Oklahoma)
- Provided general help and support.

H) Appendix

1. How 'top' data was collected

```
cemonPid=`ps auxww|grep tomcat|grep java|grep -v grep|awk '{print $2}'`
CEMonTopData=`top -n 1 -b -p $cemonPid |egrep -i "java"`

gipPid=`ps auxww|grep lcg-info-generic|grep -v grep|awk
GIPTopData=`top -n 1 -b -p $gipPid |egrep -i "perl"`

grisPid=`ps auxww|grep "slapd -h"|grep 0:2135|grep -v
GRISTopData=`top -n 1 -b -p $grisPid |egrep -i "slapd"`
```

2. How 'ps auxww' data was collected

```
cemonPs=`ps auxww|grep java|grep tomcat|grep -v grep`
grisPs=`ps auxww|grep "slapd -h"|grep 2135|grep -v grep`
gipPs=`ps auxww|grep "lcg-info-generic"|grep "osg-info-generic.conf"|awk
'{print $1,$2,$3,$4,$9,$10}'`
```

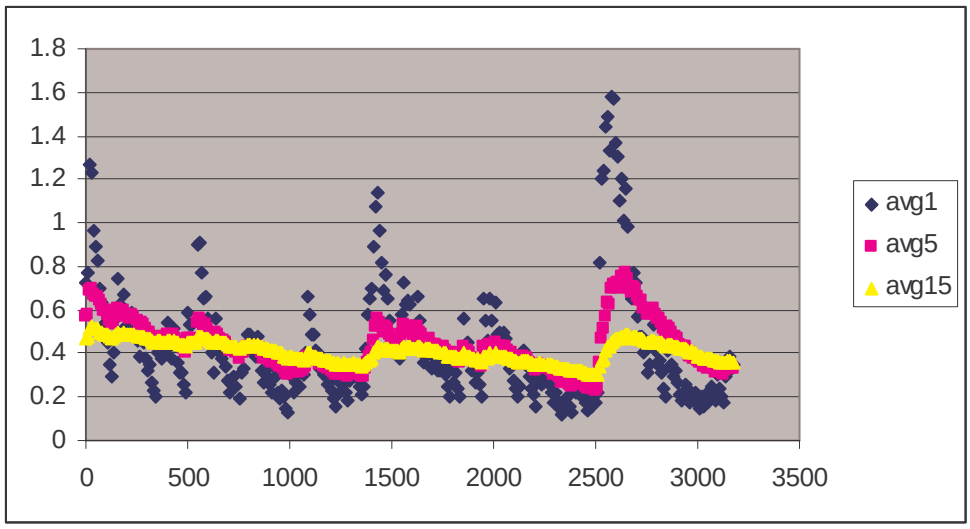
3. Other observations

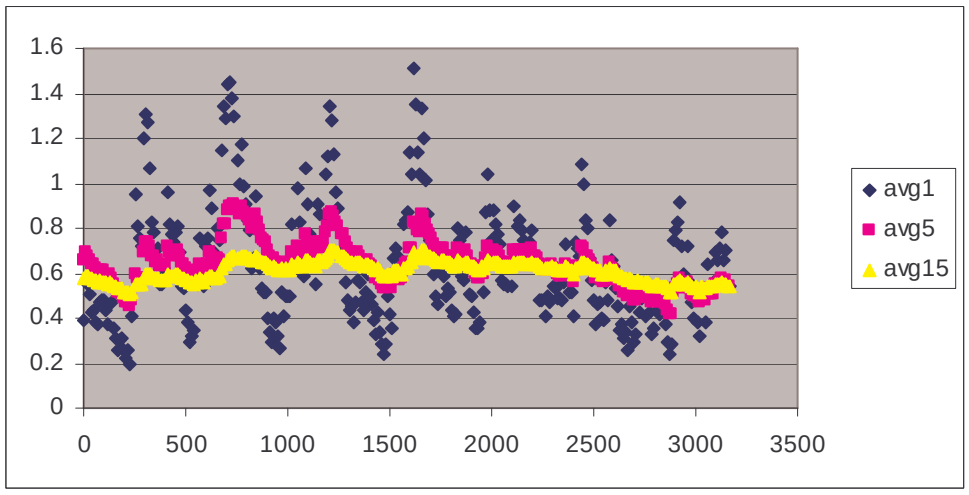
There were at least 3 instances noted where the ldapsearch from the Client hanged up the GRIS related 'slapd' process on the server. The process would consume a considerable percentage of the CPU. We have to kill it by an explicit kill command.

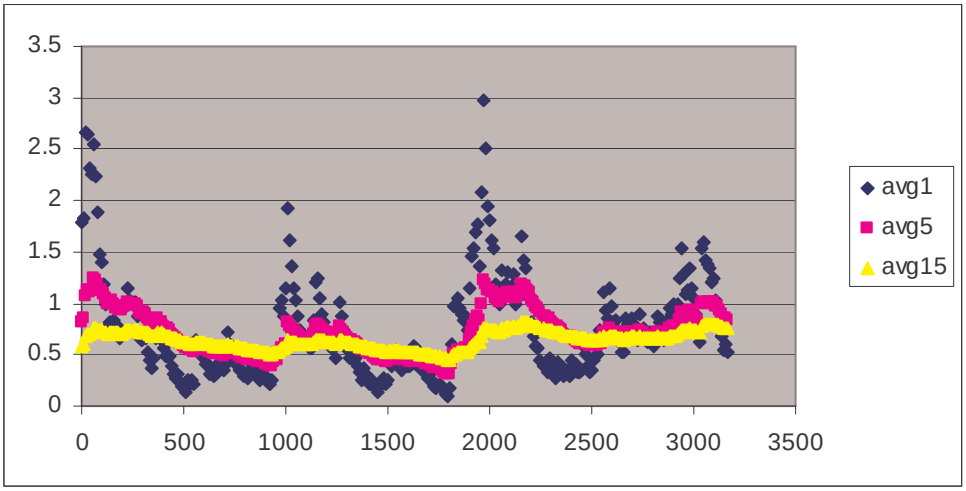
4. Data Plots

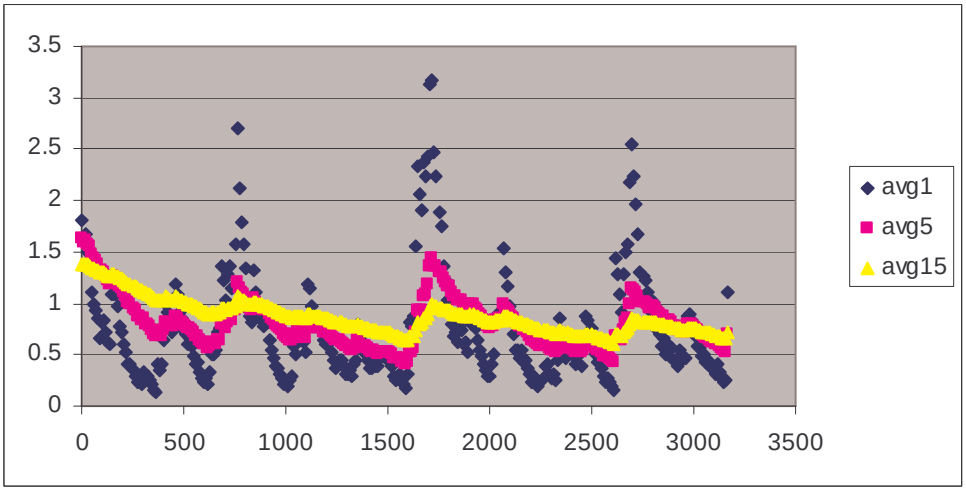
(5 Test runs (each representing an1 hour timeframe) were randomly chosen for the plots for each test case)

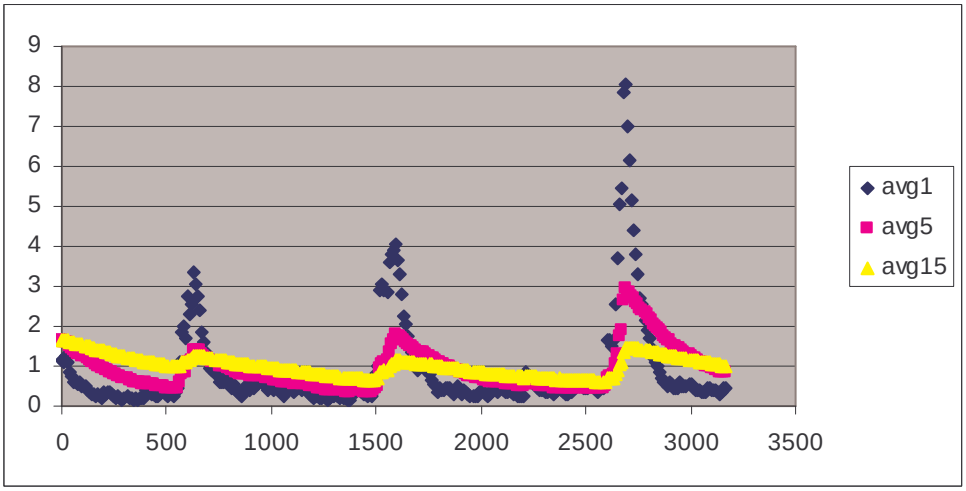
Load vs Time Plots for Test case 1 – Time is in seconds



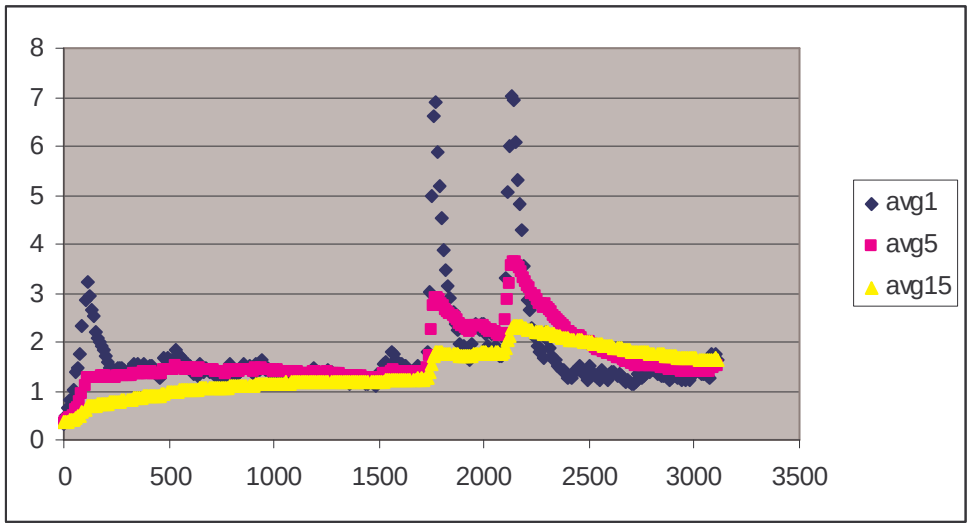


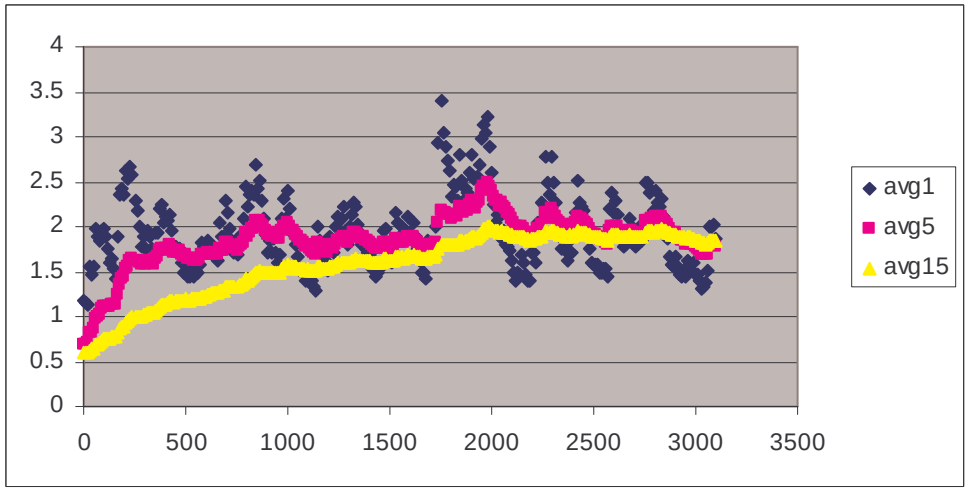

3.5
3
2.5
2
1.5
1
0.5
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

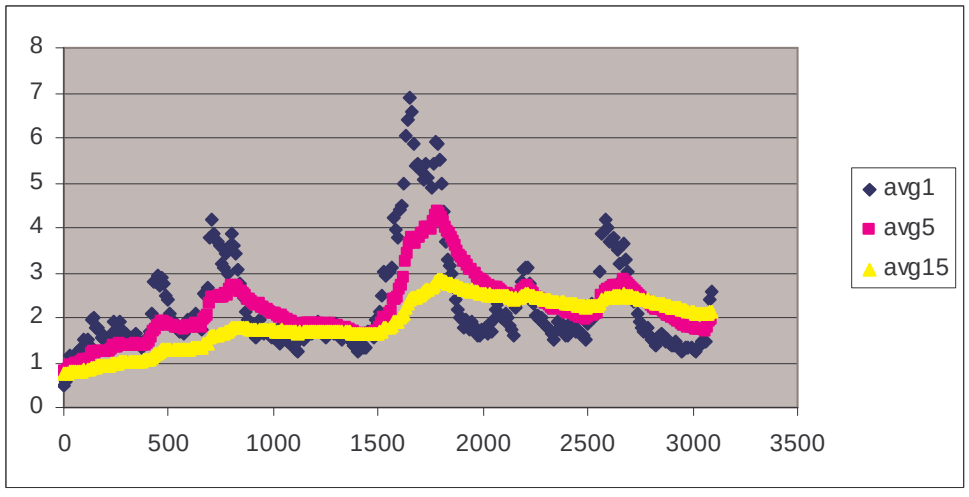

3.5
3
2.5
2
1.5
1
0.5
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

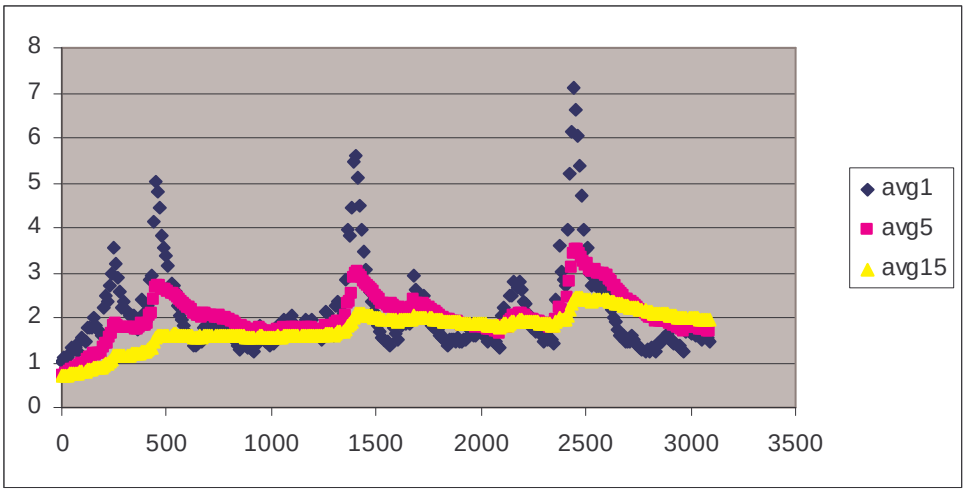

9
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

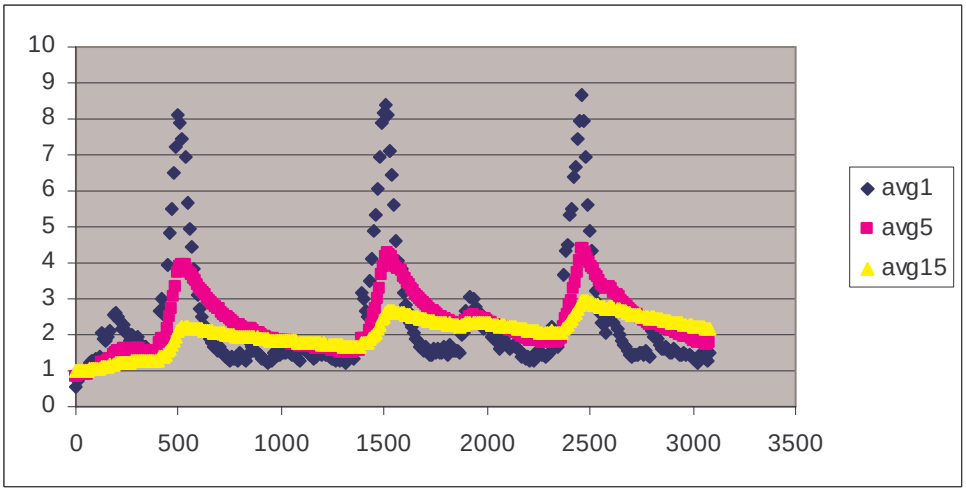
Load vs Time Plots for Test case 2 – Time is in seconds



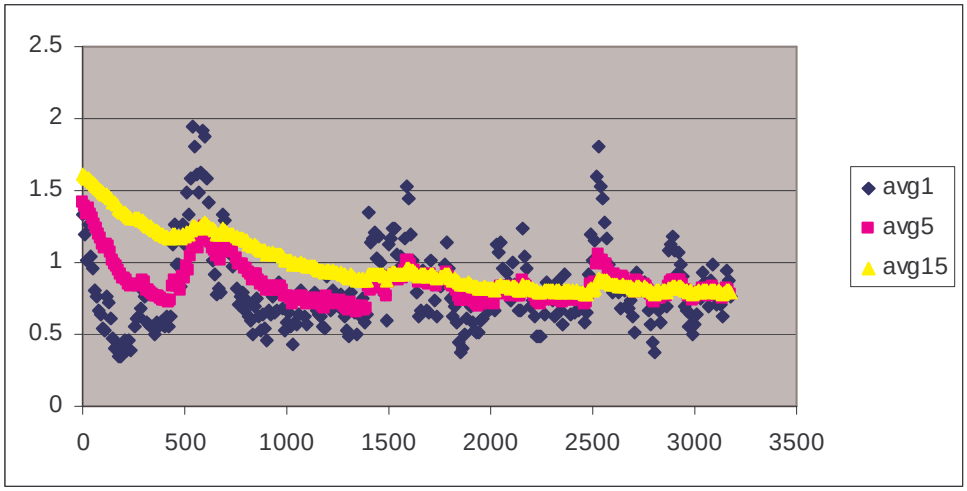


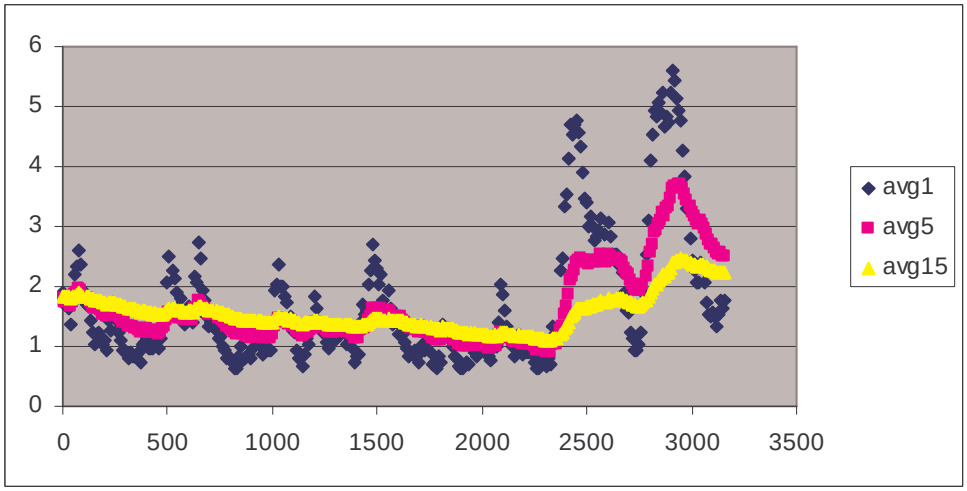


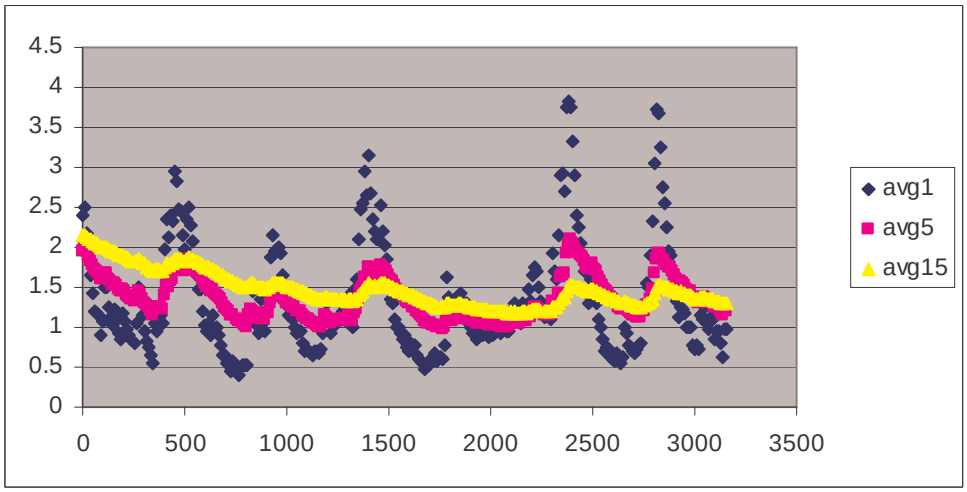

8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

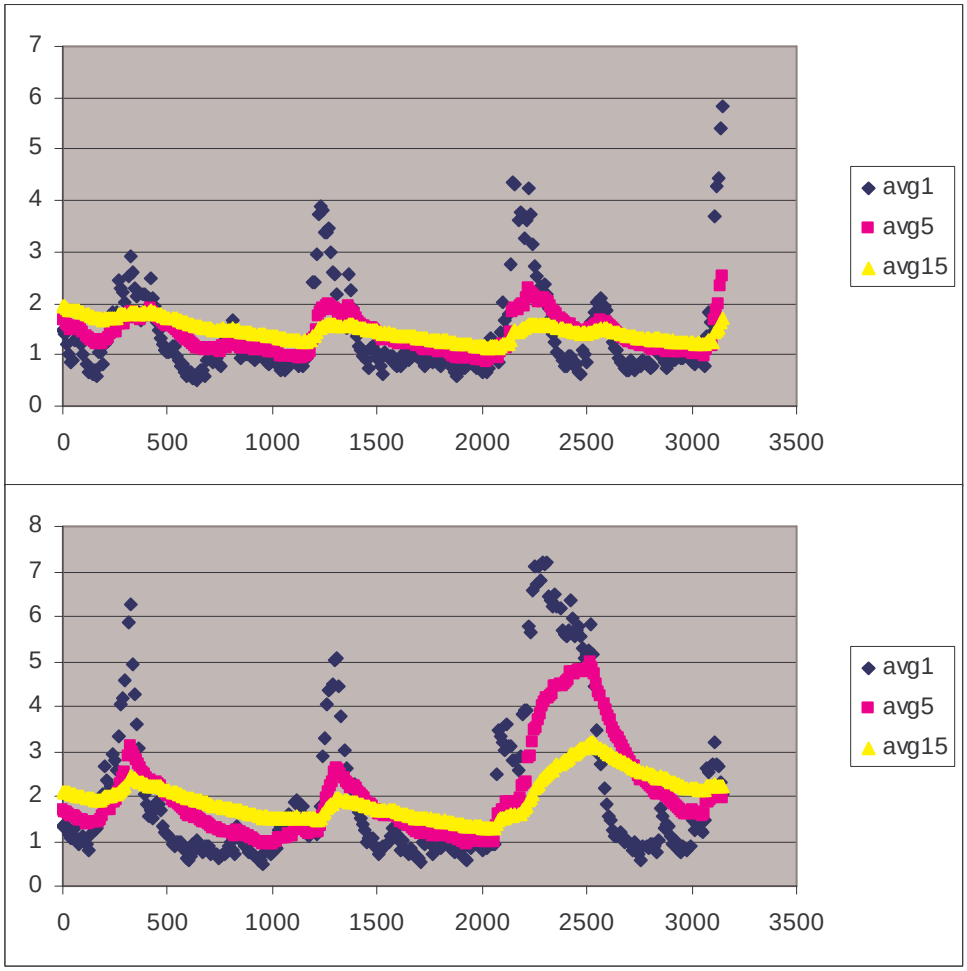

10
9
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

Load vs Time plot for Test case 3 – Time is in seconds

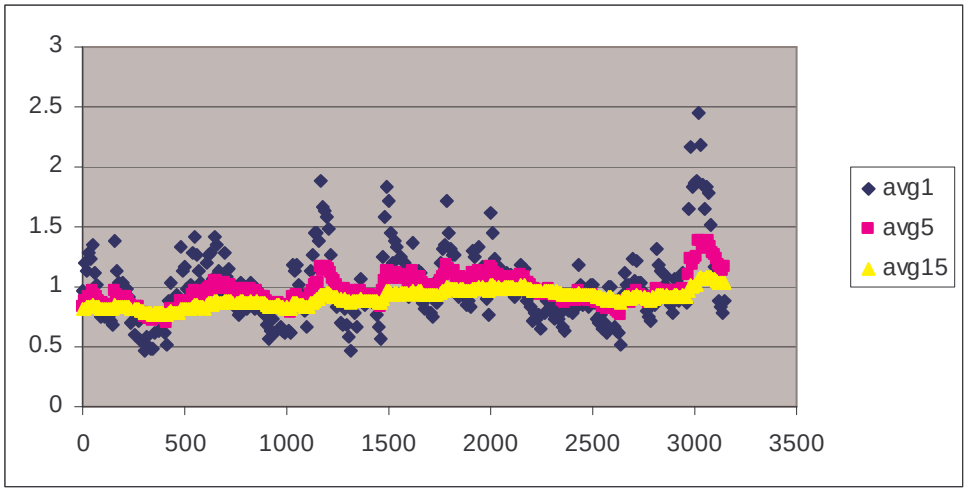


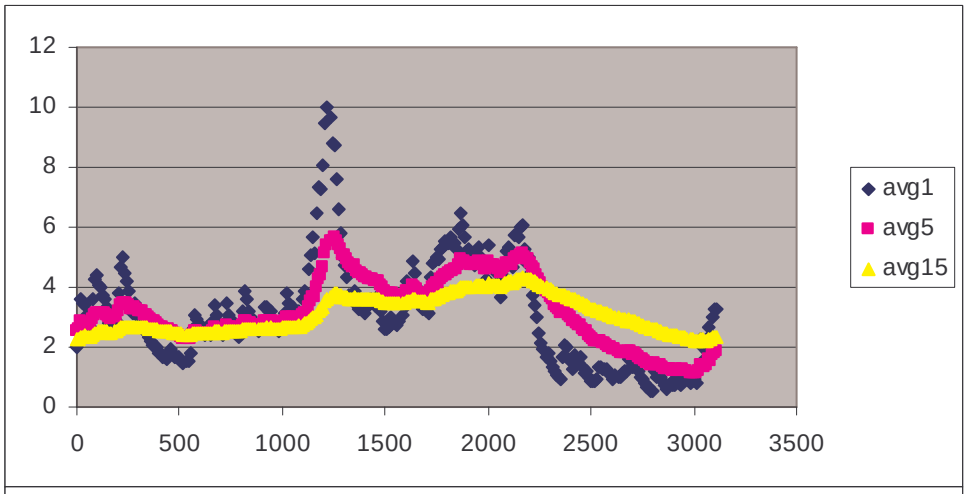


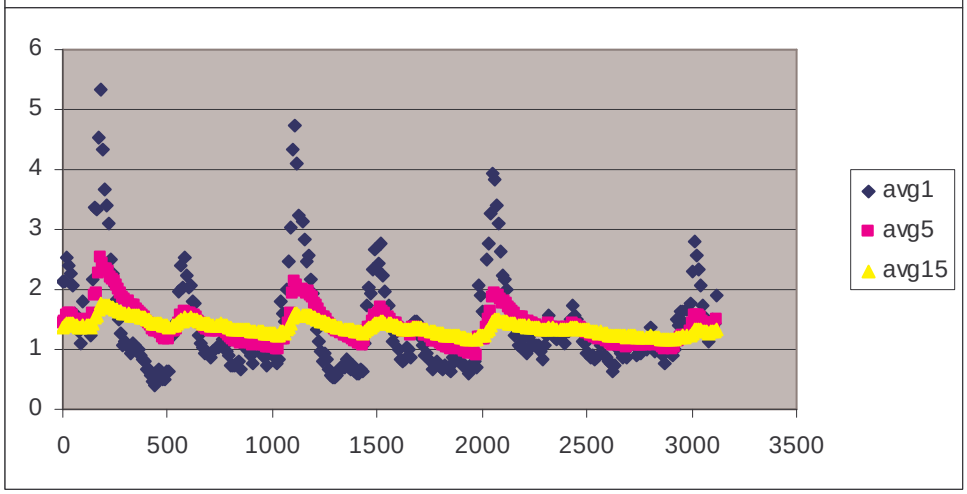


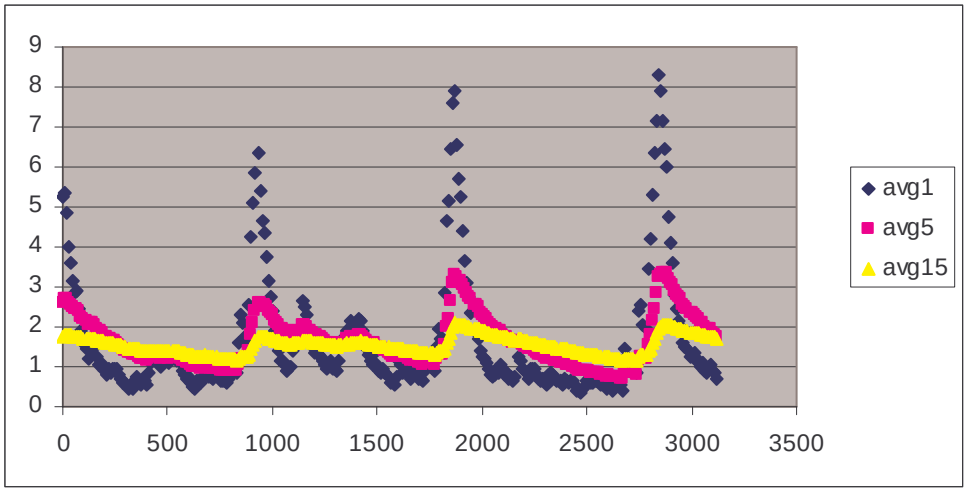

7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

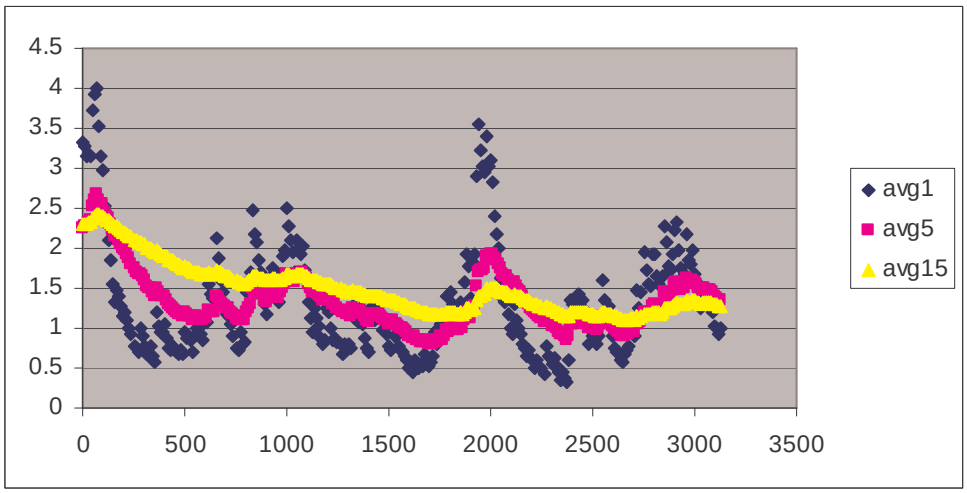
Load vs Time plot for Test case 4 – Time is in seconds



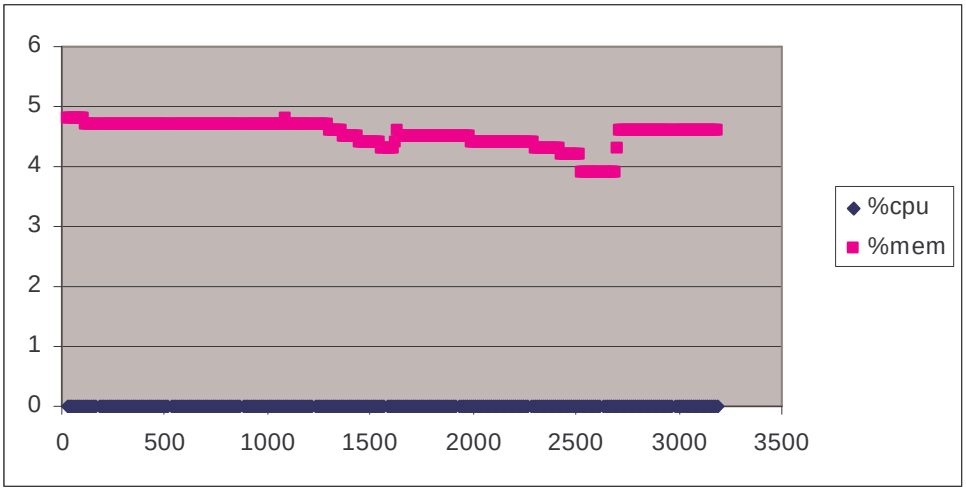


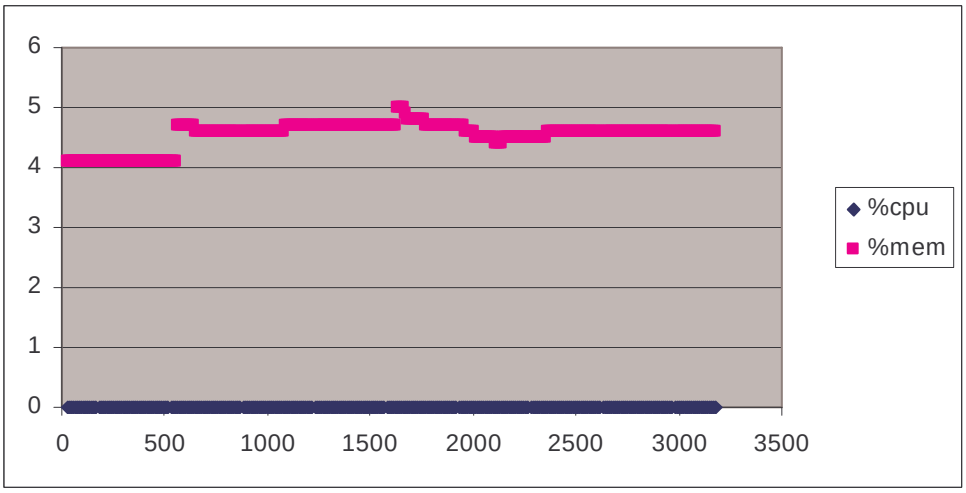


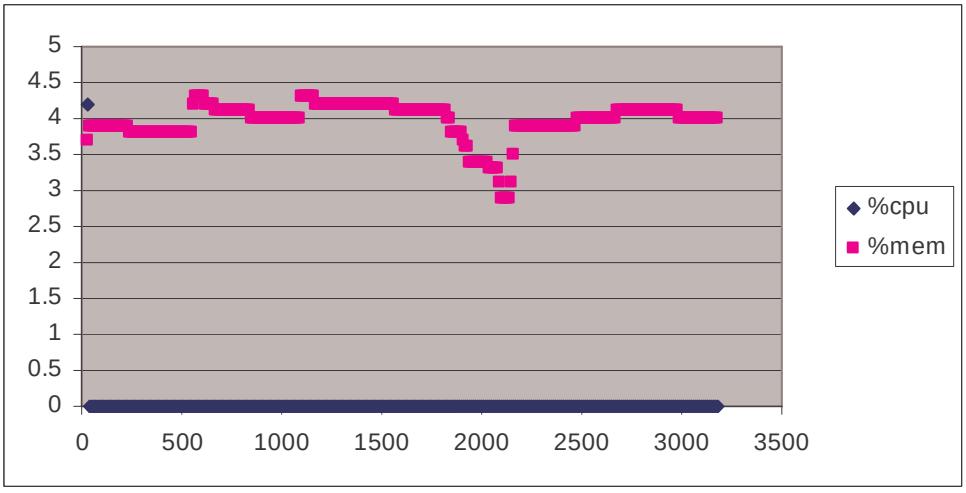

9
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

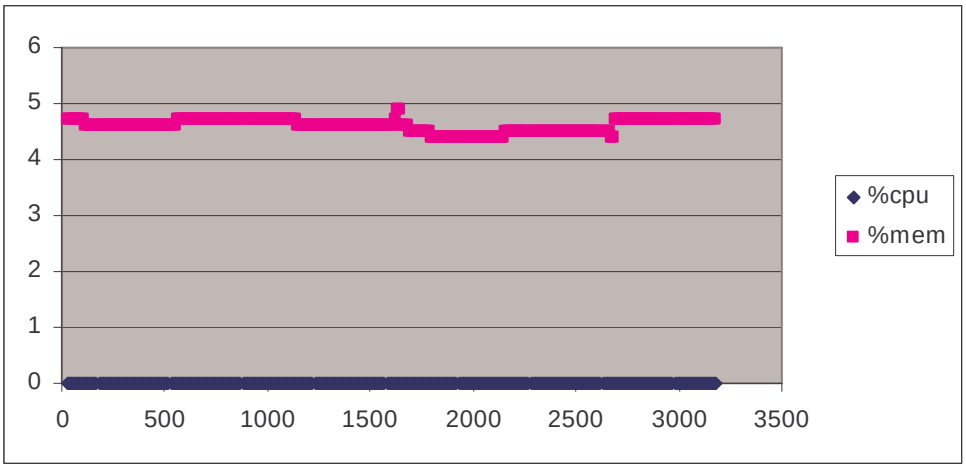

4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
500
1000
1500
2000
2500
3000
3500
avg1
avg5
avg15

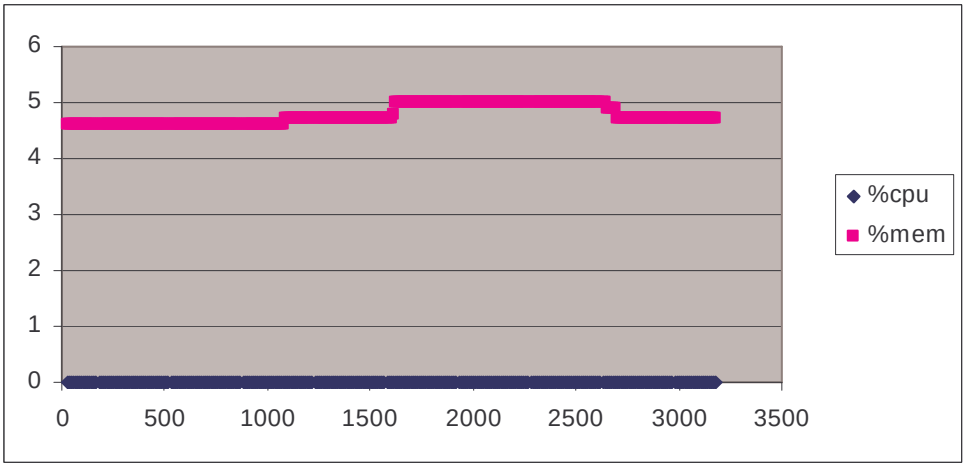
Test case 1 – Time (in seconds) vs %CPU, %MEM plot:



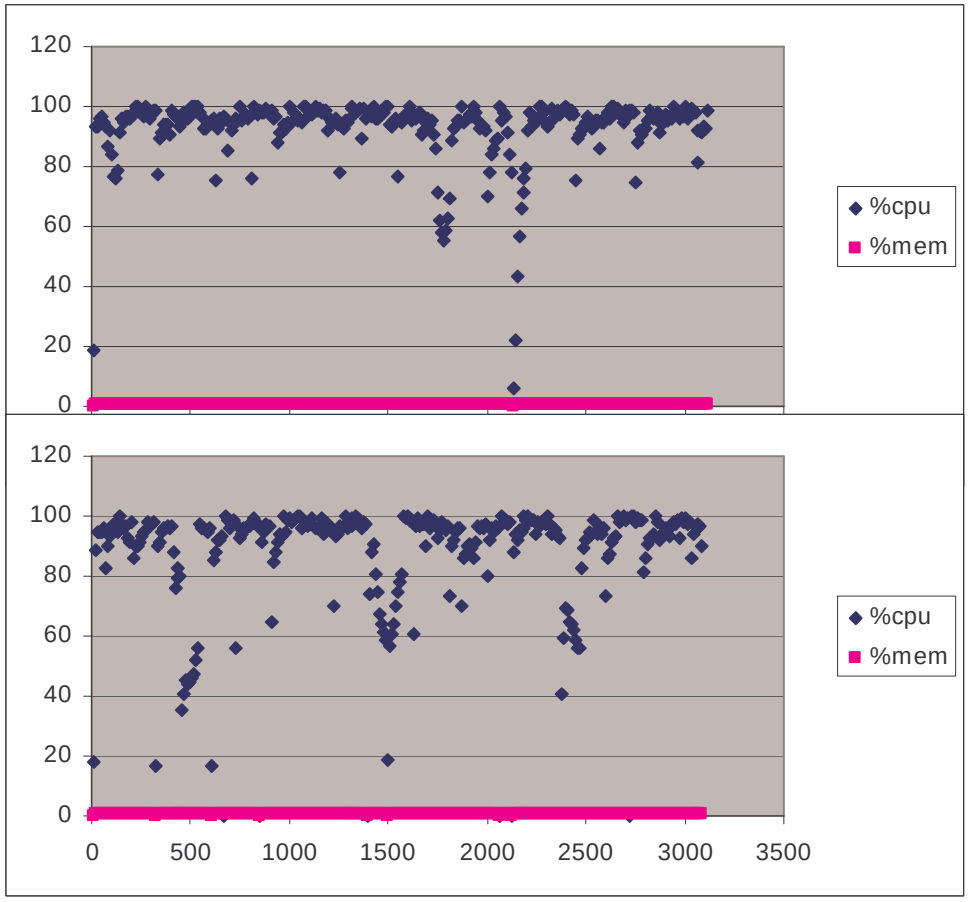


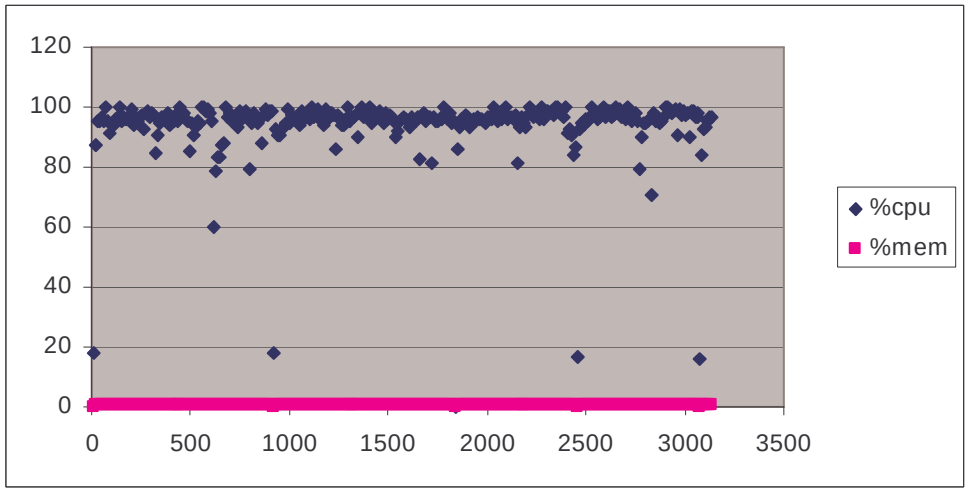


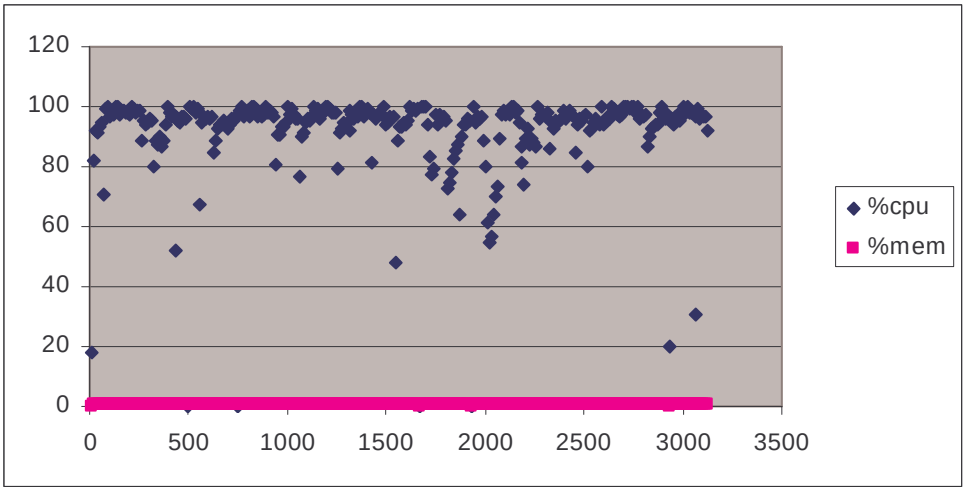

6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

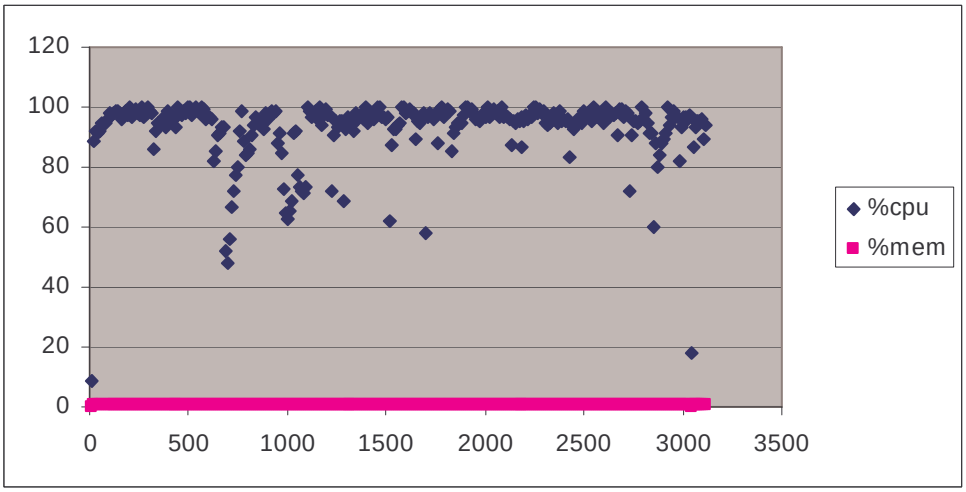

6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

Test case 2 – Time (in seconds) vs %CPU, %MEM plot:

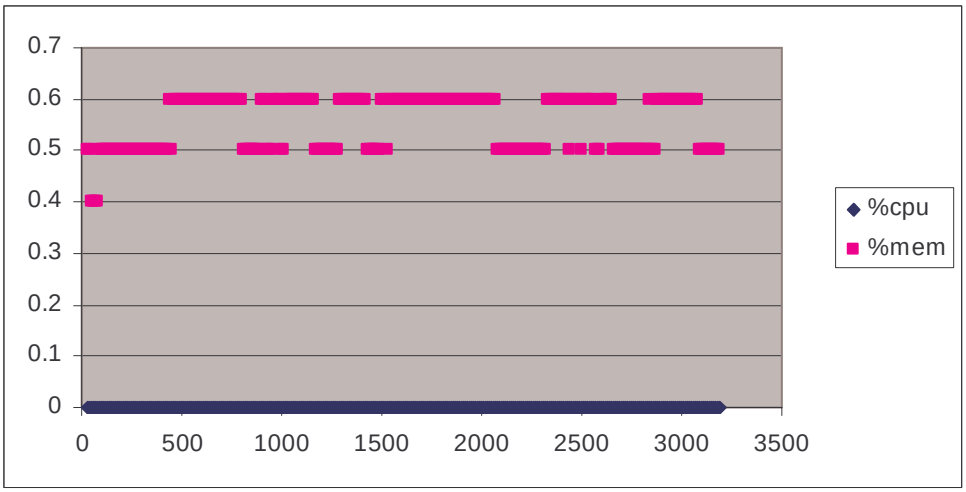


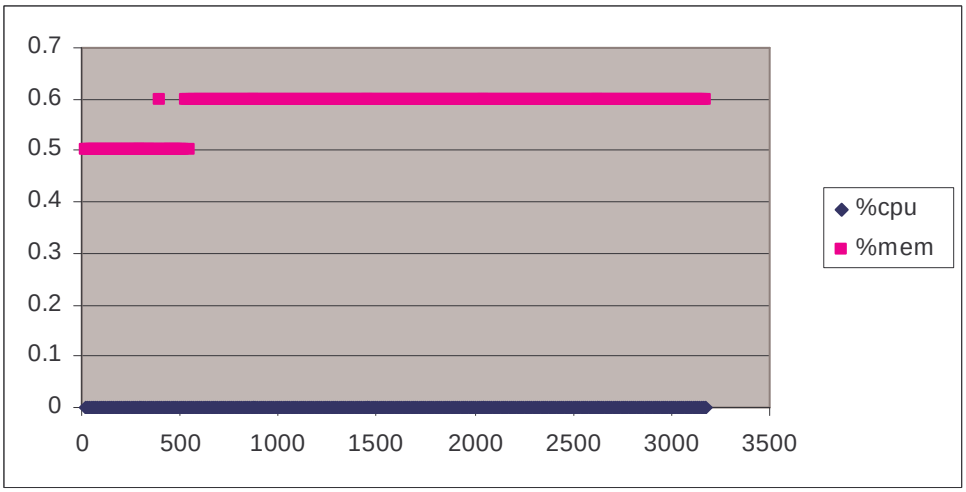


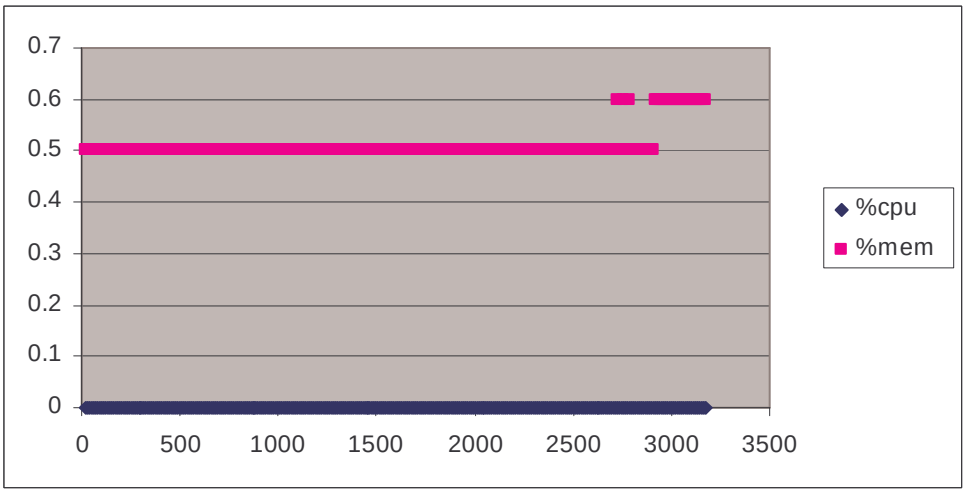

120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

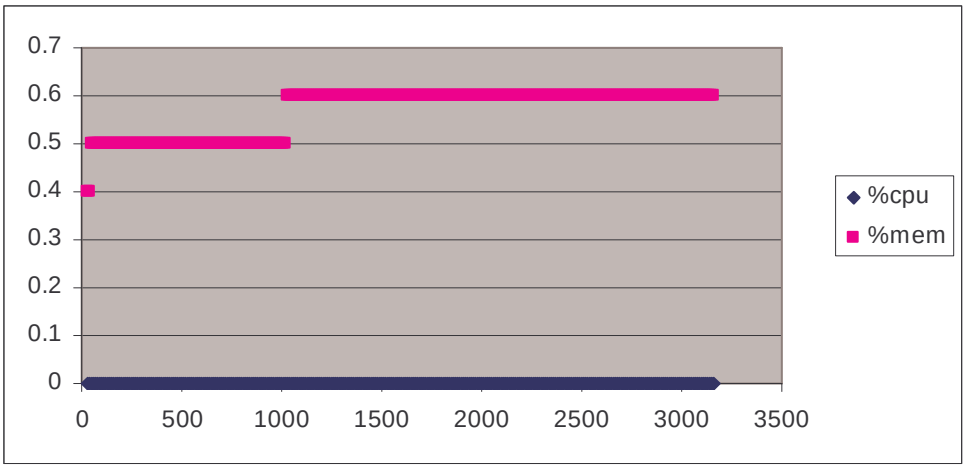

120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

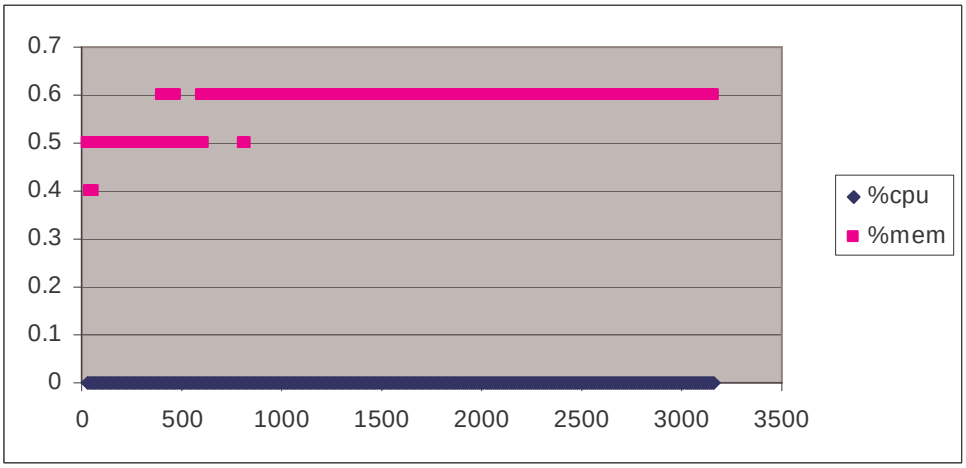
Test case 3 – Time (in seconds) vs %CPU, %MEM plot:



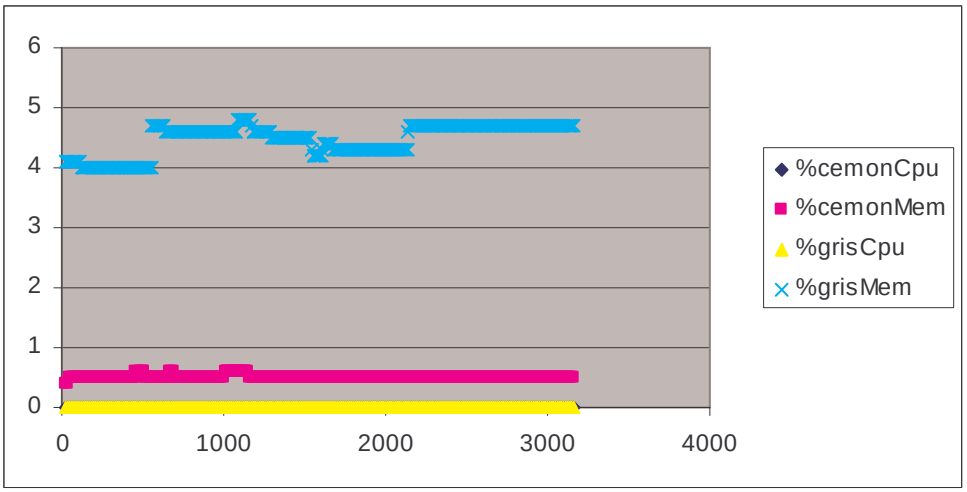


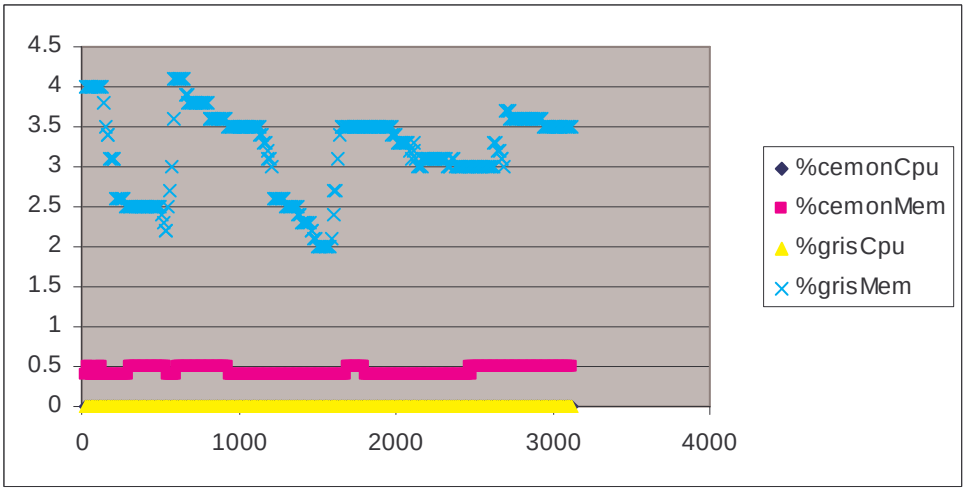


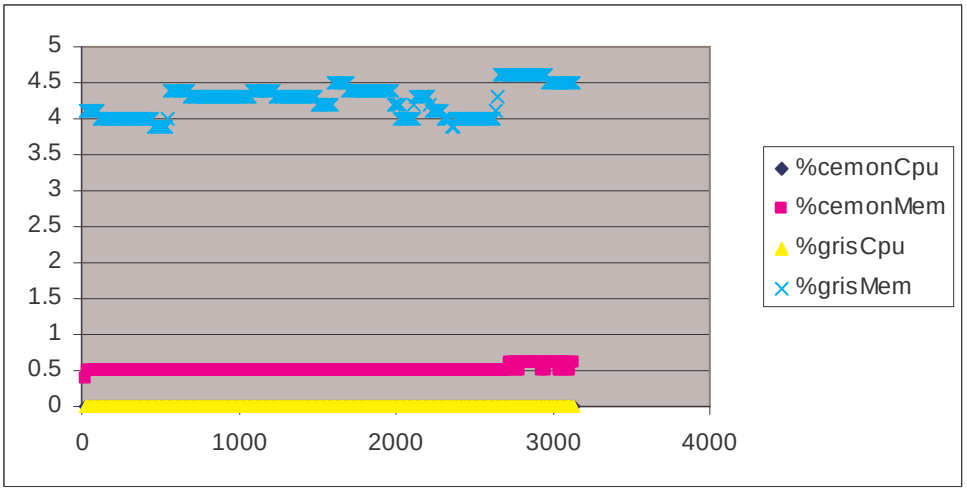

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

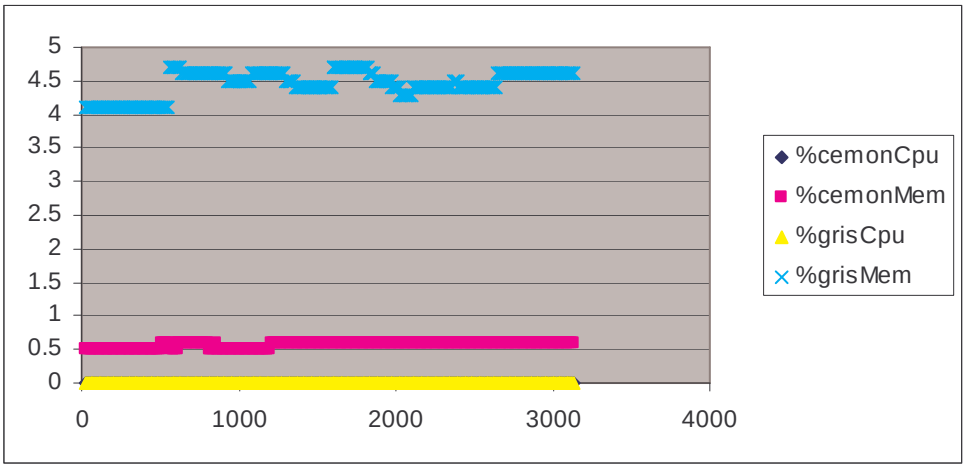

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500
3000
3500
%cpu
%mem

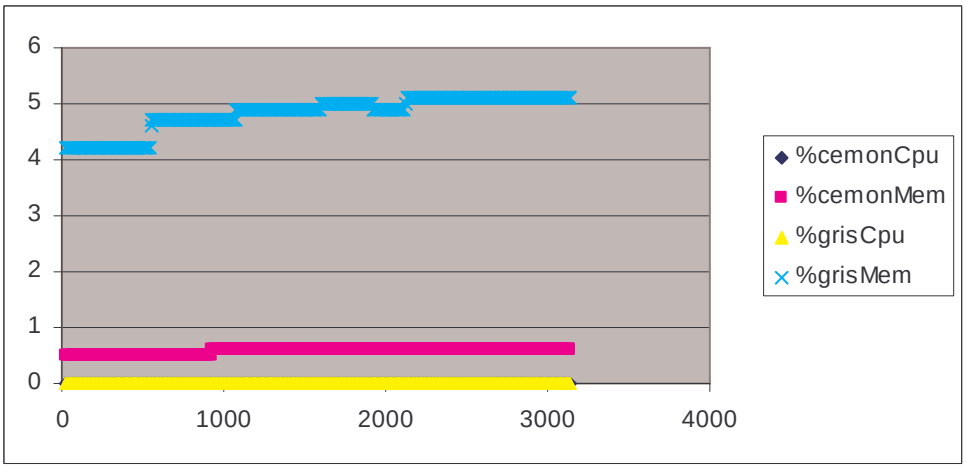
Test case 4 – Time (in seconds) vs %CPU, %MEM plot:








5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1000
2000
3000
4000
%cemonCpu
%cemonMem
%grisCpu
%grisMem


6
5
4
3
2
1
0
0
1000
2000
3000
4000
%cemonCpu
%cemonMem
%grisCpu
%grisMem