Benchmarking the performance of JMS on computer clusters

Branimir Radić Emir Imamagić

Department of Computer Systems,
University Computing Centre, Croatia
{bradic, eimamagi}@srce.hr

ABSTRACT

Job management systems (JMS) are systems in charge of distributing jobs on a cluster of computers. There is no generally accepted way or algorithm for measuring the performance of JMS. The most relevant characteristics of JMS on computer clusters are throughput and turn-around time. Throughput is the number of jobs that are completed on a system in a specified time. Turn-around time is the time needed for a job that is submitted to a system to be completed and the results of the completing job to be accessible. This paper describes our attempt to benchmark throughput of three frequently used JMS: Sun Grid Engine (SGE), Torque and Condor.

Introduction

When measuring the complete performance of computer clusters many aspects can be observed: speed of the interconnecting network, speed of the cluster itself measured in FLOPS/sec, speed of clusters data storage and other. In this paper we discus measuring of the performance of a specific component of a cluster - job management system.

Concrete results obtained by measuring the performance of three different JMS are presented.

JMS is part of a cluster that distributes jobs for execution on different nodes of a computer cluster. Role of a JMS is to optimize the use of clusters resources, enable creation of access and usage policy to different resources, hide complexity of a cluster from cluster users and provide a unique interface for accessing clusters. Performance of different JMS can be measured in **throughput** as a number of processed jobs on a single cluster in a unit of time, or **turn around time** – the elapsed time for different JMS on the same cluster for the same job.

There is no existing generally accepted way or algorithm for measuring the performance of a JMS on computer clusters. Our intention is to test the behavior of a cluster managed with different JMS when imposed the same workload consisting of a various benchmarks (which also measure other aspects of cluster performance).

Overview of JMS tested and testing tools used

During our work three different JMS where tested: Sun Grid Engine (SGE), Torque and Condor.

Condor [3] is a JMS intended primarily for the High Throughput Computing (HTC). Condors main advantages are: CPU harvesting, special *ClassAds* language for describing jobs and nodes and checkpointing and process migration.

Condor's orientation to HTC has certain drawbacks:

- Although Condor can be used for execution of parallel jobs, it is not his standard mode of operation and therefore enabling system for execution of parallel jobs takes serious adjustment.
- Checkpointing is possible only for programs that were linked to Condor libraries during compilation time.
- Checkpointing Capability applies only to serial jobs.

Sun Grid Engine (SGE) [1] has a well developed concept of calendars where the time when a certain resource will be unavailable can be defined and the job execution will be adjusted accordingly. SGE also enables CPU harvesting. Main drawback of the SGE is the lack of a simple way to define global queues for jobs and the lack of possibility to define a preemptive schedule for execution of jobs.

Tera-scale Open-source Resource and QUEue manager (Torque) [2] has an efficient interface for communication with parallel libraries, advanced scheduling and capability of using desired module as a job distributor. As a drawback there is no capability of defining complex properties of nodes.

NAS Parallel Benchmarks (NPB) [4] is a suite of benchmarks for cluster benchmarking [5]. NPB is comprised of eight different tests written in FORTRAN and C. NPB's main advantage is that for each test there are five different sizes, enabling NPB to be used to test desired aspect of a cluster by choosing an appropriate test, and enabling the choice of complexity by choosing between those five different sizes.

Testing

As there is no universally accepted benchmark or an algorithm for testing the performance of JMS, we had an opportunity to define the one that suited our needs. Main goal was to test throughput of clusters running different JMS. Throughput of a cluster can be measured by counting how many jobs can be executed in a certain time using different JMS or by measuring the time necessary for clusters using different JMS to execute same workloads. We decided to use the latter approach.

Test clusters were made of two-processor computers used as computing nodes for the cluster and one two-processor computer used as the front-end. Computers used where two HP Blade processor machines on 2.8 GHz with 2 GB of RAM. Condor cluster had three nodes and a front end computer, while Torque and SGE clusters had four nodes and a front end computer.

For the testing of JMS we decided to use different sets of NAS Parallel Benchmarks (NPB). NPB was identified as the most appropriate of the Cluster

benchmark suites [5] for testing the performance of JMS. There are 8 forms of tests that NPB benchmark uses to determinate performance of a cluster system. Tests vary in complexity and in parts of the cluster that are being tested by them (bandwidth, CPU performance etc.). Some tests put an emphasis on how cluster executes parallel jobs while other on how cluster executes serial jobs etc.

On each cluster controlled by a different JMS a number of tests was submitted. Tests varied in number of simple tests that they were comprised of and some tests used hyper threading while others did not. All tests were compiled with same libraries (even same versions of libraries) so that all the tests run on testing clusters were identical.

Complete description of each test can be seen in Table 1. All tests were repeated several times and the final results are an average of those measurements. Shorter tests were repeated more times and longer tests were repeated only a few times. All tests repeated only once would in total last approximately 44 hours of sole execution. This is the reason why longest tests could be repeated only several times.

JMS must be tested with same tests compiled using same compiler programs and linked with same libraries. This has been achieved without difficulties and all forms of NPB tests were successfully compiled with same parallel libraries on all three clusters (all the test which were used). For each form of tests and for each test a separate executable file had to be created so that compilation times can't influence the final results. Libraries that were used for those tests are explained in the Table 1.

Table 1 Description of tests

Name of the test	Description of the test
Hyper_short	Set of 28 NPB serial tests. Hyper threading is switched on.
no_hyper_short	Set of 28 NPB serial tests. Hyper threading is switched off
Hyper_long	Set of 28 NPB serial tests repeated 12 times. Hyper threading is switched on.
no_hyper_long	Set of 28 NPB serial tests repeated 12 times. Hyper threading is switched off.
long_test	Set of 28 NPB serial tests repeated 144 times. Hyper threading is switched on.
mpich_hyper	Set of 70 NPB parallel tests which use 2, 4 or 8 processors. Hyper threading is switched on.
mpich_no_hyper	Set of 70 NPB parallel tests which use 2, 4 or 8 processors. Hyper threading is switched off.
mpich-mpd / lam hyper	Set of 70 NPB parallel tests which use 2, 4 or 8 processors. On rocks-SGE cluster MPICH-MPD was and on OSCAR-PBS LAM/MPI was run. Hyper threading is switched on.
mpich mpd/lam no_hyper	Set of 70 NPB parallel tests which use 2, 4 or 8 processors. On rocks-SGE cluster MPICH-MPD was and on OSCAR-PBS LAM/MPI was run. Hyper threading is switched off.

mixed	Set of 28 NPB serial tests and 70 NPB parallel tests which use 2, 4 or 8 processors. Serial test were run 32 times and parallel were run 8 times.

Results

In this part, results of the tests of different JMS are given and interpreted. Although NPB automatically generates files in which performance measurement results of different parts of cluster is written these results will not be discussed. The results contained in the report files that NPB generates do not describe the characteristics of JMS.

In Table 2 results generated by our tests are presented. Numbers written in the table show the average time in seconds that a given set of NPB jobs take to be completed on the cluster running a certain JMS.

Table 2 Results of measurements

Test	Number of jobs	SGE	Torque	Condor
Hyper_short	28	**896**	1029	1080
no_hyper_short	28	1010	**909**	1020
Hyper_long	336	3777	**3612**	3780
no_hyper_long	336	5053	**5001**	6660
Long_test	4032	39471	**38594**	-
mpich_hyper	70	1279	**1006**	-
mpich_no_hyper	70	2776	**1802**	-
mpich-mpd / lam hyper	70	934	**933**	-
mpich mpd/lam no_hyper	70	1830	**1651**	-
mixed	1456	**16223**	17148	-

Results of tests indicate that all three JMS Condor, SGE, and Torque work successfully under high load. Condor system was tested with a limited number of tests because it was not possible to run parallel jobs on it. When observing the results of the tests, the fact that Condor cluster had one less node then the other cluster must be taken into consideration. With that in mind, seeing that execution of tests with Condor JMS lasted only a bit longer then the same tests with other JMS

means that Condor showed indisputably best throughput when working with serial jobs.

Jobs which were executed with hyper threading were completed significantly faster than the same jobs executed without hyper threading. Throughput with hyper threading is much higher for serial jobs because JMS start several jobs simultaneously and so the overhead created by JMS is considerably reduced. This combined with the fact that NPB jobs require only a small amount of resources (memory, disk space) explains why hyper threading increases throughput. Separately behavior of all three systems was tested with jobs which require high quantities of resources. In that test hyper threading reduced the throughput.

SGE and Torque have similar results with Torque showing a bit better performance when working with a homogeneous set of jobs and SGE works a bit better than Torque when processing a combination of serial and parallel jobs.

Conclusions

Testing influence of JMS on throughput didn't result in discovering the best JMS but rather proved that every JMS performs best when faced with problems it was designed to solve. Condor performs better in distributing serial jobs to the cluster, while SGE and Torque handle parallel jobs more effectively. SGE is more successful in distributing heterogeneous sets of jobs, while Torque works better with homogeneous sets.

It is hard to switch between different JMS every time a different type of job is submitted for execution on a cluster. However, knowledge obtained in this paper can and should be used to optimize performance of computer grids comprised of several clusters, and should help with the choice of JMS needed for a certain type of jobs. Grid scheduling system could make sure that serial jobs be submitted to a Condor controlled cluster, and parallel jobs to SGE and Torque. Scheduling system should also make sure that the jobs that require little resources be executed on a cluster that uses hyper threading.

Some work should also be done on integrating capabilities which already exist in certain JMS in order to improve general performance of clusters and enable clusters to be used in a simple and intuitive way.

References

[1] Grid engine home page, http://gridengine.sunsource.net/

[2] Torque home page, http://www.supercluster.org/projects/torque/

[3] Condor project home page, http://www.cs.wisc.edu/condor/

[4] NAS Parallel Benchmarks home page, http://www.nas.nasa.gov/Software/NPB/

[5] Cluster Benchmarks Web Page, http://www.mfn.unipmn.it/~mino/cluster/benchmarks/