CROWNBench: A Grid Performance Testing System Using Customizable Synthetic Workload

Xing Yang, Xiang Li, Yipeng Ji, and Mo Sha

School of Computer Science, Beihang University, Beijing, China
{yangxing, lixiang, jiyipeng, shamo}@act.buaa.edu.cn

Abstract. The Grid middleware must be developed iteratively and incrementally, so Grid performance testing is critical for middleware developers of Grid system. Considering the special characters of Grid system, in order to gain meaningful and comprehensive results of performance testing, it is necessary to implement testing on real Grid environment with various types of workload. CROWNBench, as described in this paper, is a system for helping Grid middleware developers to evaluate middleware design and implement using customizable synthetic workload. Middleware developers can customize testing workload basing on the model of Grid workload derived from real workload traces, including its structure and parameters, and then workload is synthesized automatically and contained jobs will be submitted by CROWNBench in a distributed manner. CROWNBench defines several metrics for measuring Grid performance as automatic testing results. The experiment, which used CROWNBench to test the performance of Grid system with CROWN Grid middleware, shows that the system already finished have accomplished its prospective goal. It can implement Grid performance testing in an efficient, flexible, controllable, replayable and automatic way to help middleware developers evaluate and improve their products effectively.

Keywords: Grid computing, performance testing, synthetic workload.

1 Introduction

Grid is a new computing infrastructure, and is developing quickly since proposed in 1990s. Performance testing is needed by people including Grid system designers, developers of Grid middleware, application developers and system users. Comparing to traditional distributed system, such as cluster, the content and implication of Grid system performance is different because of its characteristics, for instance, heterogeneity, dynamics and the special way for sharing resources. Accordingly, performance testing methods and tools especial for Grid system are necessary.

The greatest motivation of our work is to provide a performance testing tool for Grid middleware developers, firstly for CROWN [1] middleware developers, to validate function and performance of their designing and implementing. To gain meaningful results, the Grid system, including physical resources, system architecture and workload, used in test should be close to the target system. Because testing we discuss and implement here is real testing which is executed on real circumstance, physical resources and system architecture are exactly the same as the target system. Thus, what we are focusing on in our system is workload used in testing.

Workload impacts Grid system performance greatly. Usually, there are three kinds of workloads can be used in testing to assess the performance of Grid systems: real Grid workload, synthetic Grid workload and benchmark workload. Synthetic workload, which is basing on workload model derived from real workload traces, is regarded to be a more appropriate candidate in our performance testing system.

In this paper, we present CROWNBench, a system for Grid performance testing using customizable synthetic workload. In order to gain flexibility and universality, CROWNBench allows testers to customize their workload model, including application and job submission rules, basing on workload statistical model extracted from real workload trace. In this way, Grid middleware developers can acquire comprehensive performance data of Grid system by running various kinds of workload. What is more, testing procedure can be controlled and replayed easily.

2 Related Works

Even though workload in CROWNBench system could be customized, synthetic workload should still be basing on workload model and its contained elements and parameters. So real workload traces of some certain Grid systems should be well studied and concluded to a model, which could depict general characteristics of real workload. Workload models of kinds of computer systems have been well studied and researched [2] [3]. Even Grid system workload model has been started to research on some Grid system testbeds to evaluate Gird system performance [4] [5]. Base general rules in these models are extracted. And for testing performance capacity of Grid system, amount and distribution of workload should also be tunable to simulate real workload in different time and on different Grid testbeds.

GRASP [6] is a project for testing performance and validating dependency of Grid system by using probes, some low level benchmarks; NGB[7] is developed for Grid from NPB. It contains a suite of benchmark the structure of which is described by data flow graph; the benchmarks of GridBench [8] cover multiple levels benchmarks, including Micro–benchmark and Micro–kernel. GRENCHMARK [9] is a Grid performance testing framework with synthetic workload. But only four settled applications are supported to synthesize workload, and customizable workload is not supported.

Comparing to projects discussed above, CROWNBench provides tools to analyze real workload trace and maintain testing environment, which are both not covered in these projects. Analyzing real workload trace could find out approximate workload statistical model, and testers could use it to customize the synthetic workload close to real workload and acquire meaningful result of Grid performance testing by running it. Maintaining an isolated testing environment for a Grid performance testing could eliminate the influence among different performance testing and minimize the influence which is exerted on working Grid by performance testing.

3 Grid Performance Testing System

3.1 Grid Performance

When discussing performance of distributed systems, we are usually focusing on the quality and quantity of resources they providing. Grid is regarded as a virtual computer in [10], and performance testing of Grid system is to test performance of this virtual computer constituted by each layers of Grid system while it is running.

Compared to traditional distributed system, performance of physical resources is not the only content of Grid system performance. Grid middleware performance is a very important part in the whole Grid system, especially when we consider requirements of CROWNBench. Throughout development, the middleware must be developed iteratively and incrementally. On each milestone, performance of middleware should be tested on real circumstance. After that, results of performance metrics, which defined before, are required to validate the design and implement of Grid middleware.

3.2 Grid Performance Testing System

Usually, there are three ways of testing Grid performance, model analysis, emulating and real testing respectively. Comparing to real testing, former two have difficulties in emulating dynamic behaviors of the Grid system. Thus, we choose the way, implementing Grid performance testing on real Grid environment to gain performance results in CROWNBench.

In order to fulfill our system goal, helping middleware developers to evaluate and compare performance of different middleware versions, Grid system performance testing should be implemented on same physical resources and Grid environment structure deployed with different middleware version. As a result, performance data could be acquired to compare the performance of different middleware version, to testify whether the new one is better than the old ones.

As a Grid performance testing system, CROWNBench faces challenges in three aspects, and now our contributions also locate in them: generating meaningful and comprehensive workload for Grid performance testing. They will be described in detail in 5.4.

4 Grid Workload Model

Generally, Grid workload includes all jobs submitted to Grid within a period of time. A given real workload on some certain period of time and certain condition is not necessarily representative of workloads on other times and conditions. Thus, Gird workload model should be derived from traces of history workloads, and then be studied as the basis of workload synthesizing.

When we study Grid workload model from a higher view, there are two aspects discussed in our system. Firstly, arrival process and submission process of a certain kind of jobs in workload would be stable in a long term. Distribution of interarrival times and submission nodes are derived from this stable rule, and are used to model Grid workload and customize synthetic workload. Secondly, job running time is also an aspect in Grid workload model as well as in traditional parallel and distributed system workload. In our system, job running time will be measure after testing running as turnaround time. So it will not be settled down before testing.

5 CROWNBench

5.1 System Architecture

CROWNBench is a system for Grid performance testing with customizable synthetic workload, which has been used on Grid deployed with CROWN Grid middleware. Figure 1 shows its architecture.

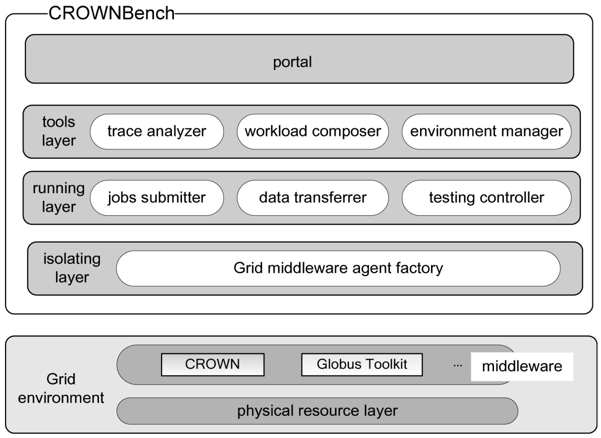


Fig. 1. CROWNBench system architecture

CROWNBench is built upon physical resources and Grid middleware. It is isolated from certain Grid middleware instance by **Grid middleware agent factory**. Layers in CROWNBench above isolating layer do not know which type of Grid middleware was deployed above physical resource layer in Grid environment. In running layer, there are three components. When testing starts running, **job submitter** will submit jobs in synthetic workload automatically. **Data transferrer** takes the responsibility to transfer related data, including testing running files and results data, among nodes in Grid environment. **Testing controller** provides a general testing running platform in heterogeneous Grid environment. In tools layer, trace analyzer, workload composer and environment manager work before testing starts. **Trace analyzer** extracts statistical model of real workload by analyzing workload trace. **Workload composer** synthesizes workload for testing through definition of testers. E**nvironment manager** maintains a testing environment for a performance testing by building up a testing environment before testing starts and clearing it after testing finishes. Testers could use CROWNBench through system **portal**, including edit testing environment, view workload analyzing results and customize synthetic workload.

5.2 Grid Performance Metrics

W stands for workload during a period of time. The number of applications included in workload is $|W|$. Job J_i includes $|J_i|$ tasks, $T_1...T_{|J_i|}$. For a task T_i, it could have 0 to $(|J_i|-1)$ pre-tasks. It can run when these tasks finished. Similarly, T_i could have 0 to $(|J_i|-1)$ post tasks, which could run simultaneously only when T_i finished.

Because of characters of Grid system, a ratio of successful jobs to whole jobs can be looked as a metric of Grid system performance, especially in Grid system where testers look job correctness more important than processing speed.

JS is the successful job rate for workload W. For an included job J, TS is the successful tasks rate:

$$JS = \sum_{J \in w \wedge J succeed} 1 \quad / |W| \tag{1}$$

$$TS = \sum_{T_i \in J \wedge T_i succeed} 1 \quad / \sum_{T_i \in J \wedge pretasksofT_i finised} 1 \tag{2}$$

Average successful task rates, shown in (3), can be computed to evaluate whole Grid system performance:

$$\overline{TS} = \sum TS_i / |W| \tag{3}$$

In Grid system which pays attention to quality of service, turn-around time TT of a job (time from submission to finishing) and each component processing time could be chosen as metrics. In a special scenario, workload contains only one kind of application. (4) is a metric of high level Grid system performance:

$$\overline{TT} = \sum_{J_i \in W} TT_i / |W| \tag{4}$$

Processing time of a certain component can be looked as a metric of that component's performance. For example, (5) can be used for evaluating performance of scheduling component in Grid middleware.

$$\overline{ST} = \sum_{J_i \in W} ST_i / |W| \tag{5}$$

5.3 CROWNBench Testing Process

A typical testing process with CROWNBench is described below as Figure 2.

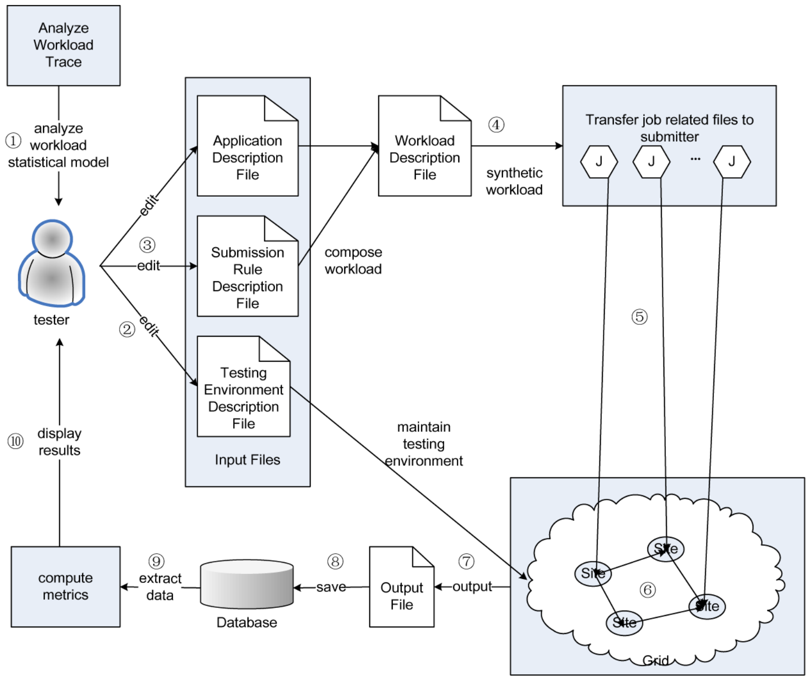

Analyze Workload Trace
① analyze workload statistical model
tester
edit
③ edit
② edit
Application Description File
Submission Rule Description File
Testing Environment Description File
Input Files
Workload Description File
compose workload
④ synthetic workload
Transfer job related files to submitter
J J ... J
⑤
maintain testing environment
⑩ display results
compute metrics
⑨ extract data
Database
⑧ save
Output File
⑦ output
Site
Site
Site
Site
⑥
Grid

Fig. 2. CROWNBench system testing process

As shown above, workload statistical model could be extracted after analyzing workload trace①; Testers could define testing environment structure in topology description file②, and then testing environment will be built by environment manager; Basing on workload trace analyzing result, testers could edit Application Description File, which contains detailed tasks and working flow and dependency among them; The other file testers should edit is Submission Rule Description File, in which submission statistic rule is set③; Workload Composer composes workload into Workload Description File for testing from Application Description File and Submission Rule Description File④; Jobs Submitter parses workload and submits each job from appointed node and in appointed time⑤; Testing Controller and script which describe the workflow of an application will control the running order of tasks ⑥; When testing is finished, output file was generated⑦; After Data Transferrer transfers result files to the testing control node, result will be extracted and saved into database⑧; After computing test results of metrics⑨; Final results will be fed back to testers⑩.

5.4 Key Techniques of CROWNBench

Analyzing Workload Trace. If testers could acquire some knowledge about statistical model of workload in Grid of which performance to be tested, they could customize synthetic workload which will be more close to real workload in the Grid. In order to extract statistical model of real workload, CROWNBench analyzes data in logs. We assume Grid workload to be analyzed is periodical, stable and self similar [11] as the precondition of our analyzing, which prove our analyzing to be meaningful and feasible. Data in logs are processed in below steps.

1. Logs processing. Job records of some certain type of application are chosen, in which job submitting time and submitting user are extracted for analyzing;
2. Data filtering. Abnormal records should be filtered to improve the accuracy of analyzing results. We use the algorithm discussed in [12] to find out abnormal data;
3. Distribution fitting. Firstly, for each distribution, parameters should be estimate by the moment matching method. Secondly, using χ^2 method, tests for goodness of fit for each distribution and choose the most suitable one.

Composing Workload. Testers define submission statistical rules for each job submitting node, which defines distribution of jobs interarrival times. After they edit Submission Rule Description File (Figure 3), Workload Composer generate synthetic workload according to Submission Rule Description File, and then jobs in workload will be serialized in lines as several “<request>”s in Workload Description File (Figure 4) in time order. This file will be parsed by Jobs Submitter.

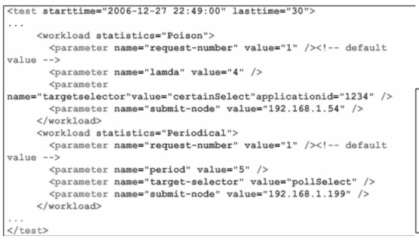

```
<test starttime="2006-12-27 22:49:00" lasttime="30">
...
     <workload statistics="Poison">
       <parameter name="request-number" value="1" /><!-- default
value -->
       <parameter name="lamda" value="4" />
       <parameter
name="targetselector"value="certainSelect"applicationid="1234" />
       <parameter name="submit-node" value="192.168.1.54" />
     </workload>
     <workload statistics="Periodical">
       <parameter name="request-number" value="1" /><!-- default
value -->
       <parameter name="period" value="5" />
       <parameter name="target-selector" value="pollSelect" />
       <parameter name="submit-node" value="192.168.1.199" />
     </workload>
...
</test>
```

Fig. 3. Submission rule description file

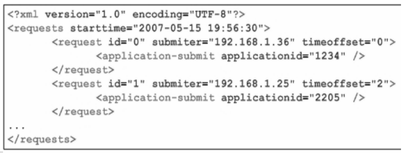
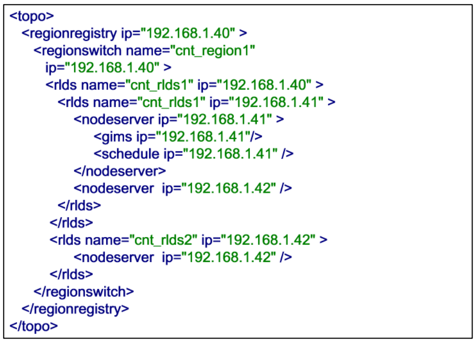

```
<?xml version="1.0" encoding="UTF-8"?>
<requests starttime="2007-05-15 19:56:30">
        <request id="0" submiter="192.168.1.36" timeoffset="0">
                <application-submit applicationid="1234" />
        </request>
        <request id="1" submiter="192.168.1.25" timeoffset="2">
                <application-submit applicationid="2205" />
        </request>
...
</requests>
```

Fig. 4. Workload description file

Maintaining Testing Environment. To eliminate the influence among different performance testing and minimize the influence which is exerted on Grid environment by performance testing, CROWNBench maintains a testing environment for a performance testing by building an isolated testing environment before testing starts and clearing it after testing finishes. Concretely, a suit of Grid middleware was deployed and configured on each node in basing on its status in Grid environment according to Testing Environment Topology File (Figure 5), and middleware and related files and database will be deleted after performance testing.

Generally, these tasks are finished manually. In CROWNBench, we developed a tool, WSAnt, to implement automatically. Building and clearing of testing environment



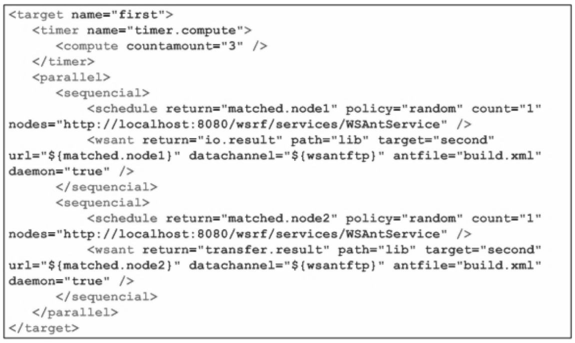
```xml
<topo>
  <regionregistry ip="192.168.1.40" >
    <regionswitch name="cnt_region1"
      ip="192.168.1.40" >
      <rlds name="cnt_rlds1" ip="192.168.1.40" >
        <rlds name="cnt_rlds1" ip="192.168.1.41" >
          <nodeserver ip="192.168.1.41" >
              <gims ip="192.168.1.41"/>
              <schedule ip="192.168.1.41" />
          </nodeserver>
          <nodeserver  ip="192.168.1.42" />
        </rlds>
       </rlds>
       <rlds name="cnt_rlds2" ip="192.168.1.42" >
          <nodeserver  ip="192.168.1.42" />
       </rlds>
    </regionswitch>
  </regionregistry>
</topo>
```

Fig. 5. Testing environment topology file

process are implemented in ANT script, and WSAnt controls its running in remote node by using web service technology. In this way, the whole process is implemented in a more automatic, accelerated and simple manner.

Application's Workflow Control. To control application workflow, we chose a flexible script tool, Ant [13], to avoid a large mount of developing works. Ant is a free open source tool of Apache Software Foundation. CROWNBench uses it to control workflow in applications for following reasons: at first, it is developed by JAVA, so it is platform independent and testing applications written by it can be integrated with Grid services frame easily; Ant and its additional project Ant-contrib [14] provide tasks and dependencies which can be described by XML. So testers' description of tasks' dependencies and workflow can easily mapping to relations between tasks of Ant. We can extend any Grid task to Ant task, and construct them to Grid application. Parameters of those tasks can be defined to change the amount of workload.

In order to support complex workflow of Grid application, CROWNBench extended Ant, and added some new tasks and conditions. These tasks have been collected in workflow control dictionary of CROWNBench. Figure 6 is a fraction of a sample workflow control script.



```xml
<target name="first">
   <timer name="timer.compute">
      <compute countamount="3" />
   </timer>
   <parallel>
      <sequencial>
          <schedule return="matched.node1" policy="random" count="1"
nodes="http://localhost:8080/wsrf/services/WSAntService" />
          <wsant return="io.result" path="lib" target="second"
url="${matched.node1}" datachannel="${wsantftp}" antfile="build.xml"
daemon="true" />
      </sequencial>
      <sequencial>
          <schedule return="matched.node2" policy="random" count="1"
nodes="http://localhost:8080/wsrf/services/WSAntService" />
          <wsant return="transfer.result" path="lib" target="second"
url="${matched.node2}" datachannel="${wsantftp}" antfile="build.xml"
daemon="true" />
      </sequencial>
   </parallel>
</target>
```

Fig. 6. Workload control script

6 CROWNBench Performance Testing Experiments

6.1 Experiment Environment

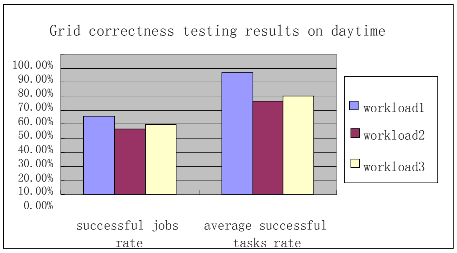
Two suites performance testing experiments with CROWNBench are implemented on CROWN testbed, which is deployed on 30 tree-structured nodes in our research institute. The first experiment is done to study the different influence on Grid system performance caused by different Grid workload model instances. We did the second one to find whether our system could gain meaningful performance results with synthetic workload comparing to real workload.

6.2 Experiment Ⅰ

In experiment we compose three applications (①②③) with different structure by using three basic tasks, which have been provided by CROWNBench, including file transferring, float computing and I/O operation. ①② all contain sequential and parallel workflow, ③ is pure chain of tasks. In this term, CROWNBench is used to study influence on Grid system performance given by job submission frequency. Three kind of workload described by matrix and testing results showed below.

$$\begin{bmatrix} J_2 & J_2 & J_2 & J_1 & \cdots & J_2 \\ J_1 & J_2 & J_3 & J_3 & \cdots & J_1 \\ J_3 & J_1 & J_2 & J_2 & \cdots & J_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & J_3 \\ J_3 & J_3 & J_1 & J_2 & J_1 & J_2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} \begin{bmatrix} J_2 & 0 & J_2 & 0 & \cdots & 0 \\ 0 & J_2 & 0 & J_1 & \cdots & J_2 \\ J_1 & 0 & J_3 & 0 & \cdots & 0 \\ 0 & J_2 & 0 & J_3 & \cdots & J_1 \\ J_3 & 0 & J_2 & 0 & \cdots & 0 \\ 0 & J_1 & 0 & J_2 & \cdots & J_1 \\ J_3 & 0 & J_2 & 0 & \cdots & 0 \\ 0 & J_2 & 0 & J_3 & \cdots & J_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & J_3 & 0 & J_2 & 0 & J_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ J_2 & J_2 & J_2 & J_1 & \cdots & J_2 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ J_1 & J_2 & J_3 & J_3 & \cdots & J_1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ J_3 & J_1 & J_2 & J_2 & \cdots & J_1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ J_3 & J_2 & J_2 & J_3 & \cdots & J_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ J_3 & J_3 & J_1 & J_2 & J_1 & J_2 \end{bmatrix}$$

Fig. 7. Workload① workload② workload③



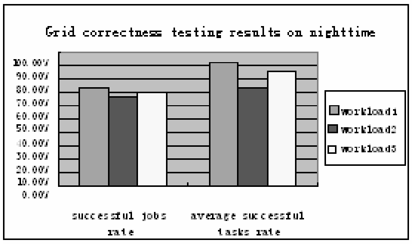


Fig. 8. Correctness testing results(daytime) **Fig. 9.** Correctness testing results (nighttime)

In daytime, besides testing workload, normal running workloads on Grid system are heavy. This situation will be eased during nighttime. Correspondingly, performance of Grid system should also be different in common sense, and it is also revealed by experiments. We can find in both of two experiments that long lasting and continuous workload will give Grid system a more great impact.

In this experiment, workload are same as ones used in first experiment, but they all contain only one application (③) for simplifying testing workload model and computing metrics of Grid system Qos. The testing results are showed as Figure 10.

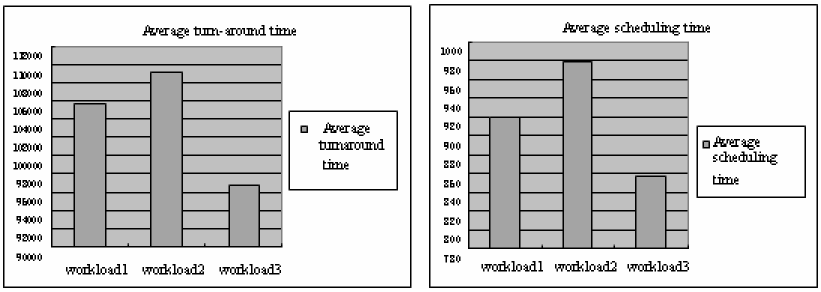


Fig. 10. Grid system Qos testing results (nighttime)

From testing results listed above, we can make some probable conclusions. Jobs submission frequency and lasting time exert influence on both running correctness and Qos of Grid system. Those workloads with higher jobs submission frequency and longer lasting time will spend longer average turnaround time and scheduling time, and also generate greater performance jitter (variance).

6.3 Experiment Ⅱ

We had learned from jobs observing component in CROWN Grid middleware that the arrival process of AREM jobs, an application used to predict a day's weather in the future, appears to be Poison distribution in a long term, and parameters of distribution are different among periods in a day. Such observing and analyzing processes are finished by component out of CROWNBench system, so it is beyond discussion of this

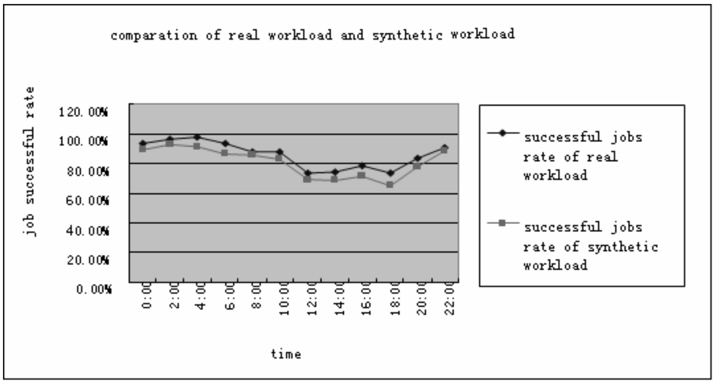


Fig. 11. Successful jobs rate of real workload and synthetic workload

paper. In this application, data firstly is fetched through I/O operation and then is processed. We used CROWNBench to compose synthetic workload using basic tasks of float computing and I/O operation and tuning parameters of Poison to emulate real AREM workload in each two hours period in a day. Then we chose a day to do performance testing every 2 hours. And we compared the average jobs rate of synthetic workload to real workload to evaluate the effectiveness of our system.

The Figure 11 show successful jobs rate of real workload and synthetic workload respectively. Its effect could be showed by results of this experiment.

7 Conclusion

CROWNBench is an automatic Grid system performance testing system using customizable synthetic workload. It allows testers to define testing workload by constructing workload and application with tasks which can be defined by them. Main contributions of CROWNBench are concluded as:

1. Testers can customize their Grid applications running for Grid performance testing. Grid performance testing is not limited to scenario running certain kinds of applications;
2. Testers can define jobs submission statistical laws to compose testing workload, and then load them to Grid system in a usual manner;
3. Running testing workload can be controlled and replayed. Because user can define their applications and tasks with probes, low level performance data of a certain component of system could also be collected to study and improve the performance of the component;
4. The whole procedure of testing is automatic, because testers only need to take part in works of editing Application Description File, Submission Rule Description File and Testing Environment Topology File.

References

1. China Research and Development Environment Over Wide-area Network, `http://www.crown.org.cn/`
2. Lublin, U., Feitelson, D.: The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. http://citeseer.nj.nec.com/lublin01workload.html
3. Calzarossa, M., Serazzi, G.: Workload Characterization a Survey. Proc. Of the IEEE 81(8), 1136–1150 (1993)
4. Li, H.: Performance Evaluation in Grid Computing: A Modeling and Prediction Perspective. In: ccgrid. Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), pp. 869–874 (2007)
5. Iosup, A., Dumitrescu, C., Epema, D., Li, H., Wolters, L.: How are real grids used? The analysis of four grid traces and its implications. In: proceedings of 7th IEEE/ACM Intl. Conference on Grid Computing (Grid 2006) (2006)
6. Khalili, O., He, J., Olschanowsky, C., Snavely, A., Casanova, H.: Measuring the Performance and Reliability of Production Computational Grids. In: Grid Computing Conference (2006)

7. Frumkin, M., Van der Wijngaart, R.F.: NAS Grid Benchmarks: A Tool for Grid Space Exploration. Cluster Computing 5(3), 247–255 (2002)
8. Tsouloupas, G., Dikaiakos, M.: GridBench: A Tool for Benchmarking Grids. In: Proceedings of the 4th International Workshop on Grid Computing (Grid 2003), pp. 60–67 (2003)
9. Iosup, A., Epema, D.H.J.: GrenchMark: A Framework for Analyzing, Testing, and Comparing Grids. In: Proc. of the 6th IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGrid 2006), May 2006, pp. 313–320. IEEE Computer Society Press, Los Alamitos (2006)
10. Németh, Z., Gombás, G., Balaton, Z.: Performance Evaluation on Grids: Directions, Issues, and Open Problems. In: Parallel, Distributed and Network-Based Processing (2004)
11. Lublin, U., Feitelson, D.G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel & Distributed Comput. 63(11), 1105–1122 (2003)
12. Calzarossa, M., Serazzi, G.: Workload characterization: a survey. Proc. IEEE 81(8), 1136–1150 (1993)
13. The Apache, A.N.T.: Project, `http://ant.apache.org`
14. Ant-Contrib Tasks, http://ant-contrib.sourceforge.net