Performance Evaluation of Grid Simulators Using Profilers

Hemant Kumar Mehta[1], Manohar Chandwani[2],
Devi Ahilya University,
Indore, India
[1]mehtahk@yahoo.com, [2]chandwanim1@rediffmail.com

Priyesh Kanungo,
Patel College of Science & Technology,
Indore, India
priyeshkanungo@hotmail.com

Abstract—**Profilers are the performance analysis tools that produce the traces of the execution of the applications developed in various programming languages. These traces can be analyzed to determine the performance of the applications. This paper compares the performance of two Grid simulators developed in Java namely the EcoGrid (developed by the authors of the paper) and GridSim. The profiler tool available with the NetBeans integrated development environment has been used. The scalability, memory usage and the runtime performance of the simulators are tested on the profilers. We showed that EcoGrid has much better performance as compared to GridSim.**

Keywords-EcoGrid; Grid Simulator; GridSim; Performance Evaluation; Profilers; NetBeans

I. Introduction

Generally, most of the runtime systems do not execute their tasks as per the performance requirement in their first execution. Once we ensure that performance issues exist with the application, we need to find out the code fragments consuming most of the resources thereby resulting in poor utilization. The major resources include processor time and memory space. The performance measurement is the key to the efficient execution of the various applications. Several tools are available to determine the performance of an application. Profilers are similar kind of tools that can identify computational performance and memory related issues. These tools range from the operating system based profiling tools and Java based profiling tools, to the third party profiling tools [9], [10]. In this paper, we consider the Java based profiling tools. We have used the profiler bundled with the NetBeans integrated development environment extensively used for Java language [1], [5], [7].

The process of measuring the performance of program is called benchmarking. Performance measurement is accomplished using available benchmarking tools. Another method used to test the performance of various applications is by comparing the applications' performance with other similar type of standard products. We have used this concept to compare the performance of EcoGrid which is a simulator developed by the authors of this paper as a test-bed for Grid scheduling algorithms [6]. We compare the performance of EcoGrid with an extensively used standard simulator called GridSim [8].

Both the simulators are developed in Java programming environment. Therefore, we have used the object oriented software development (OOSD) model to depict the profiling stage in Fig. 1. Generally, OOSD consists of four steps viz. analysis, design, coding and benchmarking or testing. In addition to these four steps, an additional step of profiling has now been added to the OOSD. This step determines the performance characteristics of the application being tested. The profiling is performed when the performance of the application has to be analysed.

The main features of profilers are:

- Profilers can determine the most frequently invoked methods.
- Profilers can determine the memory usage of an application.
- Profilers also determine the execution time taken by individual methods.
- Profilers should find the memory usage of various types of objects.
- Profilers should provide details regarding various threads created by the application.
- Memory leaks must be reported by the profilers as it is quite easy to thwart the garbage collector using the simple programming errors.
- Profilers should produce persistent data that can be analysed even when profilers are not running.
- Profiles can find total execution time of an application.
- Profilers are able to create the snapshot of the execution traces which can be analysed later.

The Java based profiling tools supports both command based as well as graphical user interface (GUI) based operations. The NetBeans integrated development environment supports the GUI based profiling operations. Various tasks that can be performed by profilers are monitoring the execution of an application, analysing the processors performance and analysing the memory usage. We use monitoring application when we want to profile the high level information about the target Java virtual machine (JVM) including thread activity and memory usage. Data regarding the applications' performance can be obtained by analyzing the CPU performance. These data includes the time to execute various methods and the number of times a method is invoked etc. Using the "analyze memory usage" option of NetBeans, the details regarding the allocation and garbage collection of the object can be ascertained.

978-1-4244-5586-7/10/$26.00 © 2010 IEEE

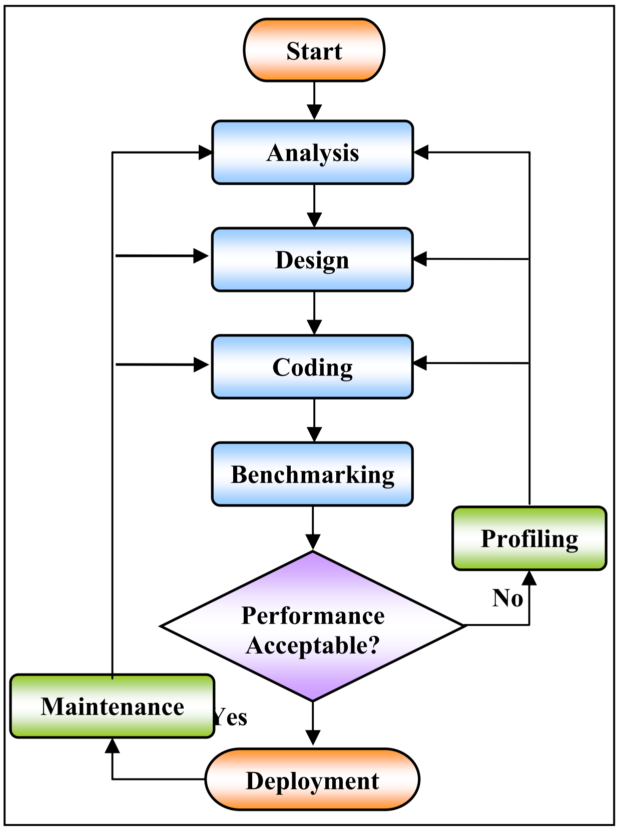

Figure 1. Performance measurement and enhancement process

NetBeans also provides the facility to take snapshots of the execution of application at various stages of execution. The three options of taking snapshot are: when a thread of application enters or leaves a specified line of code, at regular time intervals or when particular condition is met. The Java based profilers are developed on top of the Java Management Extension API [4].

II. OVERVIEW OF JAVA BASED SIMULATORS

A. *EcoGrid*

EcoGrid, developed by the authors of this paper, is dynamically configurable and supports resource modelling, advance reservation of the resource and integration of new scheduling policy. EcoGrid is object oriented which can be used for testing the Grid scheduling algorithms. Java applications can also be executed on it. In EcoGrid, a number of parameters have been included for economy based scheduling. These parameters are dynamically configurable as per the need of specific Grid environment. Although, the simulator is primarily developed for testing the economy based scheduling algorithms, we can also use it for non-economy based scheduling algorithms by using one of the configuration parameters [6]. In order to execute Java applications, the users can submit their java program using the given interface. The architecture of EcoGrid is discussed in subsequent paragraphs.

The Grid environment supported in EcoGrid contains various components required in the Grid. The architecture of EcoGrid is shown in Fig. 2. Different components of EcoGrid are: Configuration Manager (CM), Random Number Generator (RNG), Load Generator (LG), Resource Calendar (RC), Computer Node (CN), Computer Cluster (CC), Media Directory (MD), Grid Process (GP), Grid (G), Grid Scheduler (GS), Statistical Analyzer (SA) and the Grid Data Provider (GDP). The DB & FS represents database and file system. The components of EcoGrid are being described below:

1) Configuration Manager (CM): The simulator is dynamically configurable. Configuration manager is responsible for dynamically configuring the Grid simulator. Several methods have been included to read and update the values of the parameters.

2) Random Number Generator (RNG): This component provides the methods for random number generation that are based on the various statistical distributions namely Poisson distribution, Exponential distribution, Normal distribution and Uniform distribution.

3) Resource Calendar (RC): The resource calendar element of the simulator is used by the computer node to define its basic properties. It stores the details regarding the load, price, booking status, list of holidays of the time zone etc. for the resource. The load and price are having different values for weekdays and for holidays.

4) Load Generator (LG): This module generates the processing load for the system. We can also configure the parameters of the load generator for varying work load situations.

5) Statistical Analyzer (SA): The statistics generated by the system is analyzed by this component. This module provides the facility to perform statistical computations.

6) Grid Data Provider (GDP): This component is designed to separate the data access functionality from the core Grid functionality. The class will be used to facilitate data access from various sources. Data can be stored in flat files, database software or XML files.

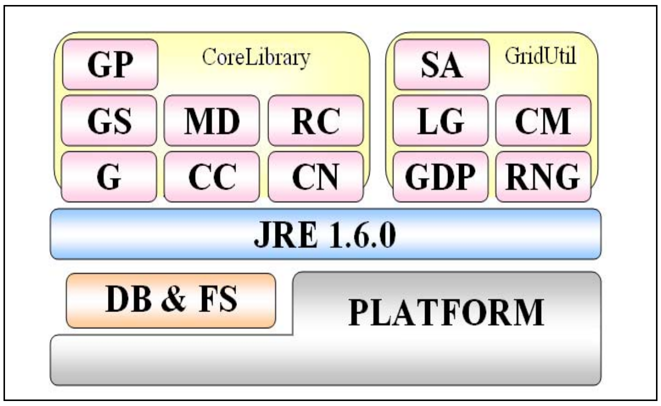

Figure 2. Architecture of EcoGrid

7) Computer Cluster (CC): The computer cluster is the collection of the computer nodes to represent the processing unit of the Grid. Cluster will publish its details in the media directory to facilitate the resource consumers.

8) Media Directory (MD): As the name suggests, this component acts as the media service to store the details of the service providing clusters, which can be accessed by the resource consumers. It also provides resource matching algorithm to determine most suitable cluster and most suitable node. Media directory contains two databases: mediaFileProvider containing the details of the service providing cluster and the mediaFileProcess containing the details of the negotiated processes.

9) Grid Process (GP): Grid process represents a task to be executed on Grid. The arrival time for the process follows the Poisson distribution. The details required regarding the process are: execution time (executionTime), input data file size (inputFileSize), starting date (arrivalDate), deadline date (deadlineDate), deadline miss limit (deadlineMissLimit) budget, optimization factor (optimizationFactor) and negotiation factor for time & cost (timeNegotiationFactor & costNegotiationFactor).

10) Grid Scheduler (GS): Grid scheduler represents the set of algorithms to schedule the processes on the clusters and the nodes. We have designed the Grid scheduler in such a manner that the users can dynamically plug their own scheduling algorithms into the system.

11) Grid (G): The Grid represents a set of cluster to perform the task. It initializes the clusters, nodes in the cluster and the various resources on a node.

12) Computer Node (CN): The basic resource in the Grid is represented by the computer node component. This component consists of the hard disk, main memory and the processor. Based on the speed of the processor, a weight is assigned to the computer node, so that it can represent higher class of the processing unit as compare to the computer nodes with normal weight (=1).

B. GridSim

GridSim is a popular and well established discrete event simulator written on top of the "SimJava" API [2]. This is a powerful simulator that mainly supports the economy based scheduling algorithms. GridSim supports simulation of computational Grid as well as the data Grid. Various extensions to GridSim are released over the period of time. The extensions include concept of advance reservation of resources, failure detection, improved network structure & topology and buffer management over the network. The GridSim models each of the resource as a separate thread of execution [8].

III. Performance Evaluation

We are testing performance of the simulators for three parameters. These parameters include the memory requirement, CPU time and scalability. The analysis of memory involves both the main memory and the virtual memory. The test of scalability will consider both the number of resources that can be modelled and the number of processes that can be executed on these resources. The scalability and memory usage is tested as a function of the number of resources as well as the number of processes. We performed all experiments on a system having Intel Core 2 Due E7400 @ 2.80 GHz processor. The system has 2 GB RAM.

A. *Test of Scalability*

The scalability of Grid simulators is tested for the maximum possible number of resources that can be configured on the simulator. The GridSim is based on the multithreaded model and spawns a new thread for every new resource to be created. Due to this the GridSim can only scale-up to a certain limit depending on the configuration of the systems. From the experiments we have shown that EcoGrid can simulate huge number of resources as compared to the GridSim.

1) Scalability of number of resources: We have tested both the simulators for maximum possible number of resources that can be created. The value of number of processes fixed 16000. From the Table I and Fig. 3 we observe that, EcoGrid can supports ten times more resources than GridSim. GridSim models each resource as a separate thread of execution. Maximum number of thread that can be created by a process is limited and depends on the configuration of the workstation. The considered workstation can create maximum 1380 threads for GridSim while the number of ten times large for EcoGrid.

TABLE I. Performance with Varying Number of Resources (Number of Processes=16000)

Number of Resources	Execution Time on GridSim (in Seconds)	Execution time on EcoGrid (in Seconds)
10	11	36
50	12	36
100	13	37
200	15	39
300	17	40
400	19	42
500	23	43
600	26	44
700	30	46
800	34	48
900	39	49
1000	43	50
1100	50	52
1200	56	53
1300	63	55
1380	69	56
1600	Execution not possible	59
2000		66
2500		72
3000		79
4000		95
8000		153
10000		182
12000		212
14000		241

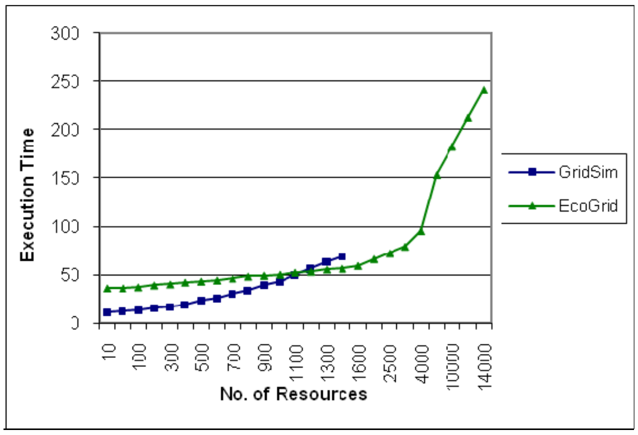

Figure 3. Performance with varying number of resources (number of processes =16000)

We observe from the Table 1 and Fig. 3 is that initially as the number of resource are few (i.e. up to 1100), the GridSim has a better performance. However with the increase in number of resources EcoGrid gives much better results. This is mainly due to the fact that, GridSim first creates all the threads to represents each resource resulting in huge memory consumption. Therefore, its performance degrades with the increase in the number of resources. On the other hand, the performance of EcoGrid does not depend on the number of resources.

2) Scalability of number of processes: In second experiment, number of resources are fixed whereas number of processes are increased. The value of number of resources is fixed at 1380, as this is the largest number of resource supported by the GridSim on the workstation under consideration. Here also, we observe from the Table II and Fig. 4 that, the Gridsim can execute around 180000 processes while EcoGrid is found to support more than 4,00,000 processes. The performance of EcoGrid remains unaffected even if the processing workload increases up to a particular limit. On the other hand, the performance of the GridSim is degrades exponentially with the increase in the number of processes.

B. *Memory Usage:*

To perform the experiments for memory usage, we fixed the number of processes to the value 14000. From the Table III and Fig. 5, we observe that initially the memory usage of the GridSim is very low but gradually as the number of resources increases its performance starts degrading. GridSim creates a separate thread for each of the resources. The control data of every thread is stored in a thread stack, and its size varies with the versions of java but the minimum thread stack size is 256 KB [3]. Moreover Java allocates the space required by the objects declared in the programs on the heap. The value of memory used for the GridSim shown in the Table III is the sum of the heap space and the sum of stack size allotted to each of the threads.

TABLE II. PERFORMANCE WITH VARYING NUMBER OF PROCESSES (NUMBER OF RESOURCES =1380)

Number of Processes	Execution Time on GridSim (in Seconds)	Execution Time on GridSim (in Seconds)
2000	19	56
4000	26	56
6000	33	56
8000	38	56
10000	45	56
12000	52	57
14000	60	57
16000	67	57
18000	104	57
20000	Execution not possible	57
22000		57
24000		57
26000		57
28000		58
30000		58
32000		58
40000		58
80000		60
100000		61
200000		66
400000		75

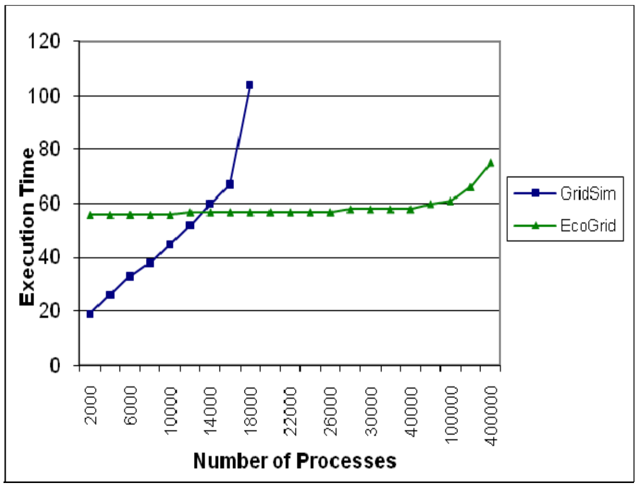

Figure 4. Performance with varying number of processes (number of resources=1380)

The memory used by GridSim mainly represents stack size used by the thread created for each resource. However, the memory used by the EcoGrid is the heap space that is used to allocate the objects created during the experiments. We have optimized the environment for all the experiments performed to evaluate the performance. As the GridSim is mainly using the thread stack memory, we have reduced the thread stack size for the Java. Reducing the thread stack size results in the creation more number of threads and ultimately more number of resources for GridSim. EcoGrid mainly uses the heap space in the memory, therefore we have increased the heap size for performing the experiments for EcoGrid. This results more number of resources modeled in the EcoGrid.

TABLE III. MEMORY USAGE WITH FIXED NUMBER OF PROCESSES (=14000)

Resources	Memory Used in GridSim (in MB)	Memory Used in EcoGrid (in MB)
10	53.5	144
50	67.5	134
100	78	141
200	104	156
300	129	168
400	158	171
500	181	166
600	207	185
700	232	196
800	260	189
900	286	187
1000	309	195
1100	338	206
1200	360	213
1300	387	211
1380	413	214
2000	Execution not possible	227
4000		379
10000		733
14000		871

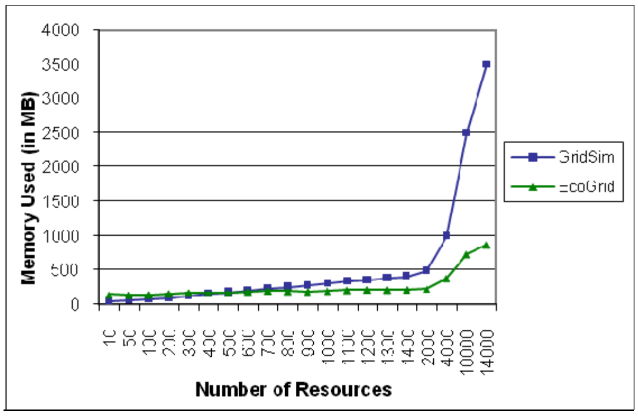

Figure 5. Memory usage with the fixed number of processes

IV. CONCLUSION

The profilers are the tools used for benchmarking and performance evaluation. In this paper, performance of EcoGrid (Grid simulator developed by the authors of this paper) is compared with a standard Grid simulator called GridSim. EcoGrid outperforms GridSim in scalability, memory utilization and runtime performance. It is found to be capable of running much larger number of processes and use much larger number of resources as compared to GridSim. Therefore, it can be effectively used as test-bed for Grid scheduling algorithms.

REFERENCES

[1] S. Elbaum and M. Diep, "Profiling deployed software: assessing strategies and testing opportunities," IEEE Transactions on Software Engineering, vol. 31, no. 4, pp. 312-327, Apr. 2005

[2] F. Howell and R. McNab, "SimJava: a discrete event simulation package for Java with applications in computer systems modeling", In Proc. 1st International Conference on Web based Modeling and Simulation, San Diego, CA, pp. 41-60, 1998.

[3] J. Gosling, B. Joy, G. Steele and G. Bracha, Java(TM) Language Specification, Addison Wesley, 3rd ed., 2005.

[4] The Java Management Extension website. [Online]. Available: http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/ -2009.

[5] The Java website. [Online]. Available: http://www.java.com -2009

[6] H. Mehta, P. Kanungo, and M. Chandwani, "EcoGrid: a dynamically configurable object oriented simulation environment for economy-based grid scheduling algorithms," unpublished.

[7] The NetBeans website. [Online]. Available: http://www.netbeans.org/ -2009.

[8] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, "A toolkit for modeling and simulating data Grids: an extension to GridSim," Concurrency and Computation: Practice and Experience, Wiley Press, New York, USA, pp. 1591-1609, Sep. 2008.

[9] S. Wilson and J. Kesselman, Java(TM) Platform Performance: Strategies and Tactic, Prentice Hall PTR, 1st ed., 2000.

[10] The YourKit Profiler website [Online]. Available: http://www.yourkit.com/ -2009.