I J C T A, 9(41) 2016, pp. 727-736 ISSN: 0974-5572

AN ADAPTIVE RESOURCE ALLOCATION SCHEME IN COMPUTATIONAL GRID

Sukalyan Goswami[1] and Ajanta Das[2]

Abstract: Computational Grid solves computation-intensive problems in the fields of science and technology. It is composed of loosely coupled virtual resources. In computational grid, resource allocation scheme needs to be adaptive to cater to the problem of load balancing. The objective of this paper is to propose and subsequently prove by virtue of benchmark code execution results of an adaptive resource allocation scheme using Nearest Deadline First Scheduled (NDFS) algorithm. This adaptive methodology of resource allocation is necessary for balancing load in the computational grid. Service Quality Agreements (SQA) are met through proper scheduling of jobs when resource is capable of executing the submitted jobs, else SQA gets declined. This paper presents the simultaneous execution results of the benchmark codes of several instances of fast fourier transform (FFT) in grid test bed, built with Globus Toolkit 5.2.

Key Words: *Grid Computing, Load Balancing, Resource Allocation, Nearest Deadline First Scheduled (NDFS) algorithm.*

I. INTRODUCTION

Computational grid [2] requires load balancing along with efficient resource scheduling in order to cater to computationally intensive problem solving across geographically dispersed autonomous groups. Participating resources of the grid remain variedly loaded. So, workload balancing needs to be achieved to ensure higher number of deadline meets for the submitted jobs in the grid. This research aims to find a solution to this problem. This paper presents an adaptive resource allocation scheme using already proposed NDFS [3] algorithm. SQAs are met by proper job scheduling in ranked resources. The proposal has been justified by testing the enhanced algorithm in a grid test bed set up by Globus toolkit 5.2 [10] and benchmark code of FFT [11] has been executed to standardize the experimental results

Organisation of this paper is as follows: challenges in grid computing are represented in section 2. Section 3 presents taxonomy of job scheduling and load balancing policies. Section 4 presents proposed resource allocation scheme. Section 5 presents experimental results and discussion of benchmark codes. Section 6 concludes the paper.

II. CHALLENGES IN GRID COMPUTING

The motivation of this work is to provide an analytical survey for grid computing models and algorithms. There are a lot of challenges in Grid computing, which are described in the following points.

[1]*Department of Computer Science & Engineering, Institute of Engineering & Management, Kolkata, India, Email – sukalyan.goswami@gmail.com*
[2]*Department of Computer Science & Engineering,Birla Institute of Technology, Mesra, Kolkata Campus, India, Email – ajantadas@bitmesra.ac.in*

- *Distribution Management:*
  Due to the networked status in grid computing, the resources could be located in real far geographic locations. This implementation of distributed network adds a tremendous weight on the messages over the network and the data transferring capacity. The traffic failure, fault tolerance, and adaptability are key issues that need to be addressed in any grid solution.
- *Heterogeneity:*
  The resources constructing the grid environment are totally heterogeneous. The task scheduling and resource management activities have significant challenges in considering these heterogeneous resources and handling them in the grid life time.
- *Resource Failure:*
  The resource participation is voluntary in computational grid. Moreover, in this distributed environment, the resources may fail anytime, thereby increasing the probability of job deadline misses. This phenomenon makes the grid operation even more challenging.

Apart from the above mentioned challenges in grid, load prediction, distribution, management, and balancing are the prime focus of research nowadays. The balancing mechanisms need to satisfy the below mentioned properties:

- *Load metric:*

The load metric is the representation used to describe the load distribution of the resources.

- *Load communication:*

Load communication defines the method of information sharing; this information contains the load of the resources.

- *Transfer:*

It relates to the physical medium for transferring jobs between hosts.

There are two basic issues in the design of load balancing solution. The policy is the set of choices to balance the load and decide which tasks should be selected to run and where. Secondly the mechanism carries out the physical facilities to be used for remote task execution and provides any information required by the policies [1] [7].

III. TAXONOMY OF JOB SCHEDULING AND LOAD BALANCING POLICIES

In static job scheduling, a particular resource will be assigned a fixed job; even if after restarting the system, the same binding of task and resource is used without considering changes that might have occurred during the system's lifetime. Dynamic load balancing takes into consideration the system parameters, which may not be known before the execution. The dynamic balancing handles the run time conditions and takes the decision depending on the actual updates. TABLE I present the comparison between static and dynamic job scheduling and load balancing algorithms. Since, the loads of the resources in grid are changing continuously, dynamic scheduling results in better scheduling of jobs, minimizing the number of job deadline misses.

- Distributed vs. Centralized:
  In dynamic policy [9], the responsibility of making global and strategic decisions could be assigned to a central location that has all the needed info for taking these decisions. But another policy recommends that the decision should be shared by multiple distributed resources.
- Local vs. Global:

Local and global algorithms [7] fall under the distributed scheme since a centralized scheme should act globally. In a local policy, each resource polls other resources in its neighbours (in the same intra-grid).

- Cooperative vs. Non-cooperative:

  It is a distributed/global class and the scheduling activities co-operate with each other and transfer information between themselves, so it is called "cooperative" policy. The cooperation here is between scheduling managers for sharing information on different scheduling scopes. The co-operation between these tasks is very beneficial to be able to share information and up to date the information.

- One-time Assignment vs. Dynamic Reassignment:

  The one-time assignment is the policy that the task is assigned once to a resource when it is ready to run, but thereafter the concept of migration of this task from resource to another does not exist. But dynamic reassignment allows the system to migrate the tasks from resource to another depending on the resource status and overall system status.

- Adaptive vs. Non-Adaptive:

  Adaptive and non-adaptive schemes [8] are part of the dynamic reassignment job scheduling and load balancing policies. In adaptive scheme, scheduler takes into consideration the previous and current system performance and is affected by the earlier decisions or changes in the environment.

Features	**Static Scheduling**	**Dynamic scheduling**
Decision Time	Compilation time	Run time
Decision Criteria	Predefined resources and tasks specifications	Environmental runtime conditions and updates
Adaptability Feature	Not Applicable	Adapts with runtime environment
Scheduling Speed	Generally faster	Generally slower
Efficiency in load balancing	Not efficient	Yields better results with lesser number of job deadline misses

TABLE I **Static Job Scheduling v/s Dynamic Job Scheduling**

IV. RESOURCE ALLOCATION SCHEME

NDFS [3], [4], [5] is a dynamic process for resource allocation in Grid Environment where tasks with nearer deadlines (measured by time to complete or TTC) are scheduled before others. The scheduler here maintains a priority queue of tasks based on their TTC. It also checks the pending queue of tasks as soon as the size of the queue exceeds a certain limit (called the ‘critical size’) and adds tasks to the priority

queue according to TTC. Once the tasks are placed in the priority queue, the scheduler picks suitable resources according to their ranks from the available resource pool and allocates tasks to the resources so that an overall optimal makespan is achieved, while maintaining an overall optimal workload balance. Figure 1 depicts the flowchart of resource allocation scheme using NDFS. The estimation of the critical size of the queue is vital for the successful running of this algorithm. Otherwise, tasks with nearer deadlines might miss their deadlines while waiting in the queue, resulting in the violation of SQA. The critical size is estimated heuristically by checking the miss-to-Met (m2M) ratio, which determines how many deadlines are 'missed' for every deadline 'met'. The critical size begins with a predefined value that increases additively till the m2M ratio exceeds an upper limit (e.g., 0.1). After that critical size is halved till m2M ratio falls below the upper limit and then the entire process is repeated. The main challenge to implement this heuristic is the dynamic workload and deadline demand of the incoming tasks, which can make the m2M ratio highly dispersed with time. Job deadline being the major priority decision parameter, NDFS can ensure that almost all the deadlines are met provided that the total CPU utilization is within permissible limits.

V. EXPERIMENTAL RESULTS

A grid test bed is set up, for the purpose of this research work, consisting of three clients, three resources and a grid broker. Globus Toolkit 5.2 is used to set up the real grid environment. Java has been used as the programming language for implementation of this research work because of its extensive and robust support in the networking environment. Specifically, JAVA has inbuilt support available for message and data passing between the clients and the broker and the broker and the resources. The System Information Gatherer And Reporter (SIGAR) [12] API [12] is used to retrieve system parameters.

Moreover, to ensure non-trivial performance of the grid, benchmark code of Fast Fourier Transform (FFT) is executed. Image processing and signal processing are the major domains in which FFT is extensively used. FFT ensures frequency domain computation is equally feasible as of time or spatial domain. As already known, fast fourier transform has a complexity of O (n^2) and requires n^2 complex multiplications and n(n-1) complex additions.

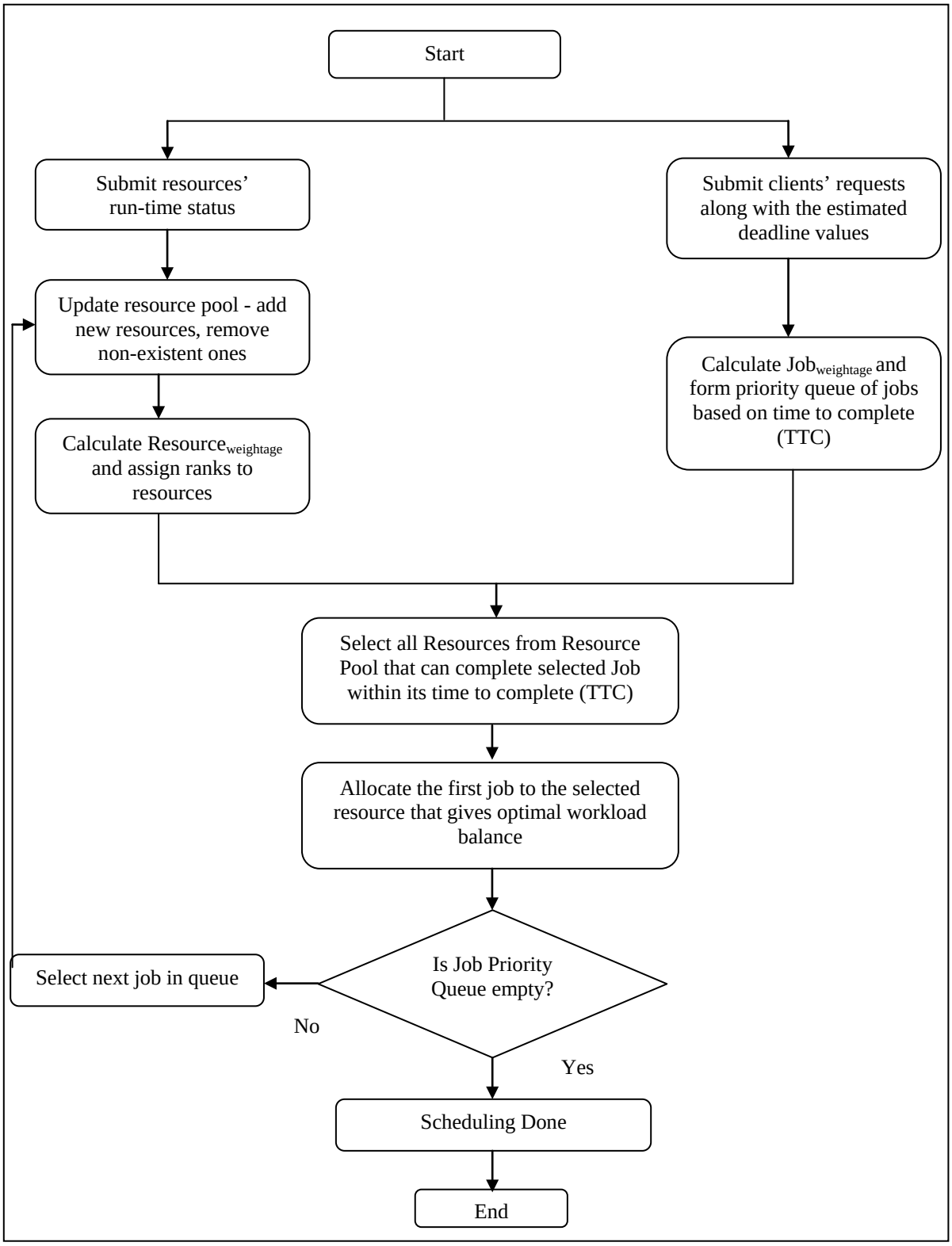

Figure 1 Flowchart of Resource Allocation Scheme

Hence, FFT is highly computation intensive job. Faster FFT computations take advantage of the periodic and symmetric properties of the complex exponential.

Basically, n length FFT job when sub-divided into two n/2 length FFT jobs will take fewer computations. FFT algorithms exploit these computational savings collectively. Hence, fewer computations will ensure that overall execution time of the submitted job, i.e., makespan will get reduced by a considerable extent. So, clients' nodes with less computational capabilities have to send computation intensive FFT jobs to the grid. Figure 2, Figure 3 and Figure 4 show the output of successful job allocation in the grid. Figure 2 shows the execution activity of the resource, Resource1 (192.168.30.4), where the job, "fft.java" is executed. Figure 3 shows successful job allocation and detailed activities of Broker. Broker received specifications from three resources, Resource1 (192.168.30.4), Resource2 (192.168.30.5) and Resource3 (192.168.30.6). Then it calculates rank metrics and assigns subsequent ranks to all these resources. In this experiment, Broker receives "fft.java" as Job file with its specific parameters from the Client (192.168.30.3). Broker then calculates rank metric for job and pushes it into job pool. At this time, bi-partite agreement, SQA is signed between Broker and Client. Then this figure also shows allocation details. The job "fft.java" is scheduled in Resource1 (192.168.30.4). Finally Broker receives execution output from the same resource and it sends the details to the Client. Figure 4 represents the execution activity of Client. In this experiment, Client (192.168.30.3) sends "fft.java" as job file to the Broker. After proper investigation, SQA is signed. Finally it receives the output details from Broker. It also checks whether quality of service is achieved or not. Output is displayed for test case purpose and as specimen. Moreover, in order to maintain privacy and security, it is not mandatory to display the output of jobs.
Restricted provision of job scheduling in computational grid environment is represented in Figure 5. The job requirements submitted by the client in this scenario are more than the available computational power of the resources. So, broker declines the SQA. Then it notifies the client that, provisioning of execution of this particular job is not possible at that point in time in computational grid environment.

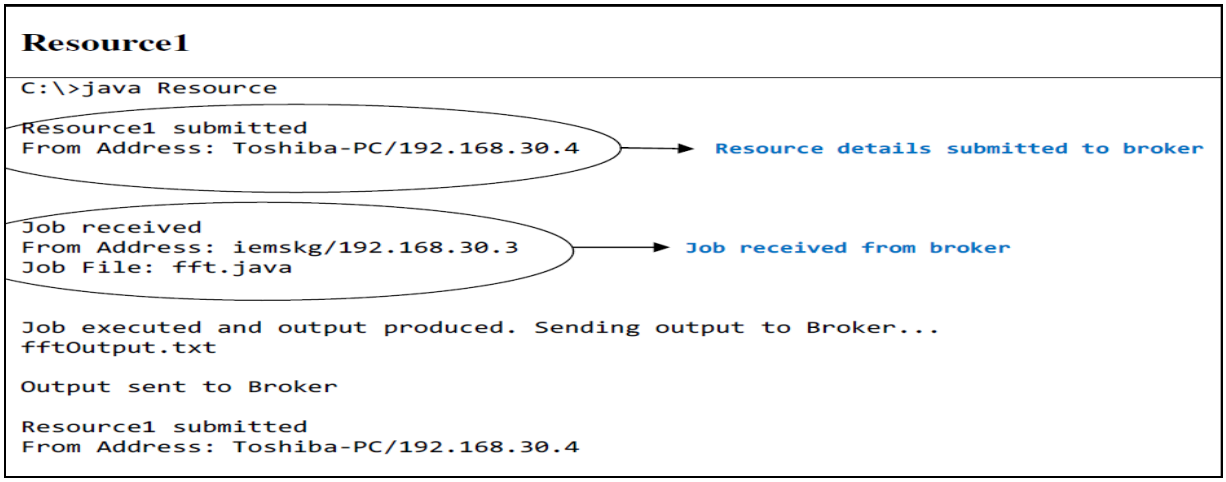

Figure 2 Successful Job Allocation - Resource

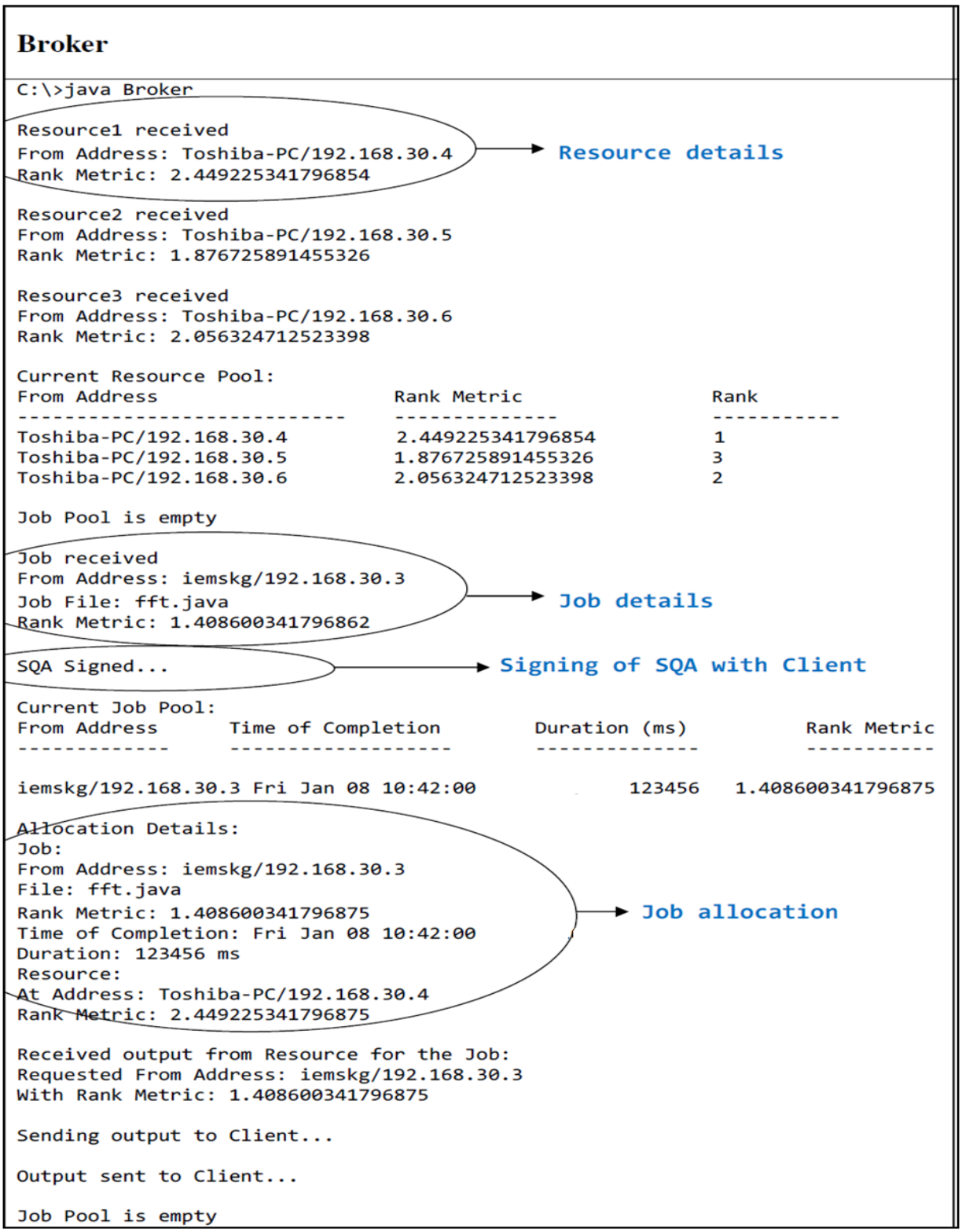

Figure 3 Successful Job Allocation - Broker

Client

```
C:\>java Job fft.java Duration 123456 ms

Job submitted
From Address: iemskg/192.168.30.3
Job File: fft.java
```

Client submits job details to broker

```
SQA Signed...
```

Signing of SQA with broker

```
Job Output:
x
--------------------
0.6353633994776804
-0.26910735478092773
-0.6193071349681862
-0.8094158806991367

y = fft(x)
--------------------
-1.0624669709705703
1.2546705344458666 - 0.5403085259182091i
1.0945794999895586
1.2546705344458666 + 0.5403085259182091i

z = ifft(y)
--------------------
0.6353633994776804
-0.26910735478092773 - 1.6542177670443986E-17i
-0.6193071349681862
-0.8094158806991367 + 1.6542177670443986E-17i

c = cconvolve(x, x)
--------------------
1.222867509963577
0.6605921326309543 - 4.150996579734887E-17i
-0.05939733686319171
-0.6952262413279611 + 4.150996579734887E-17i

d = convolve(x, x)
--------------------
0.40368664939583454 - 5.551115123125783E-17i
-0.3419619275161129 - 2.0754982898674436E-17i
-0.7145514047911508 - 8.296495469682836E-17i
-0.695226241327961 - 9.026731956384122E-17i
0.8191808605677425 + 5.551115123125783E-17i
1.0025540601470673 - 2.0754982898674436E-17i
0.655154067927959 + 8.296495469682836E-17i
-1.1102230246251565E-16 + 1.3177728536119009E-16i
C:\>
```

Client receives job output from broker

Figure 4 Successful Job Allocation - Client

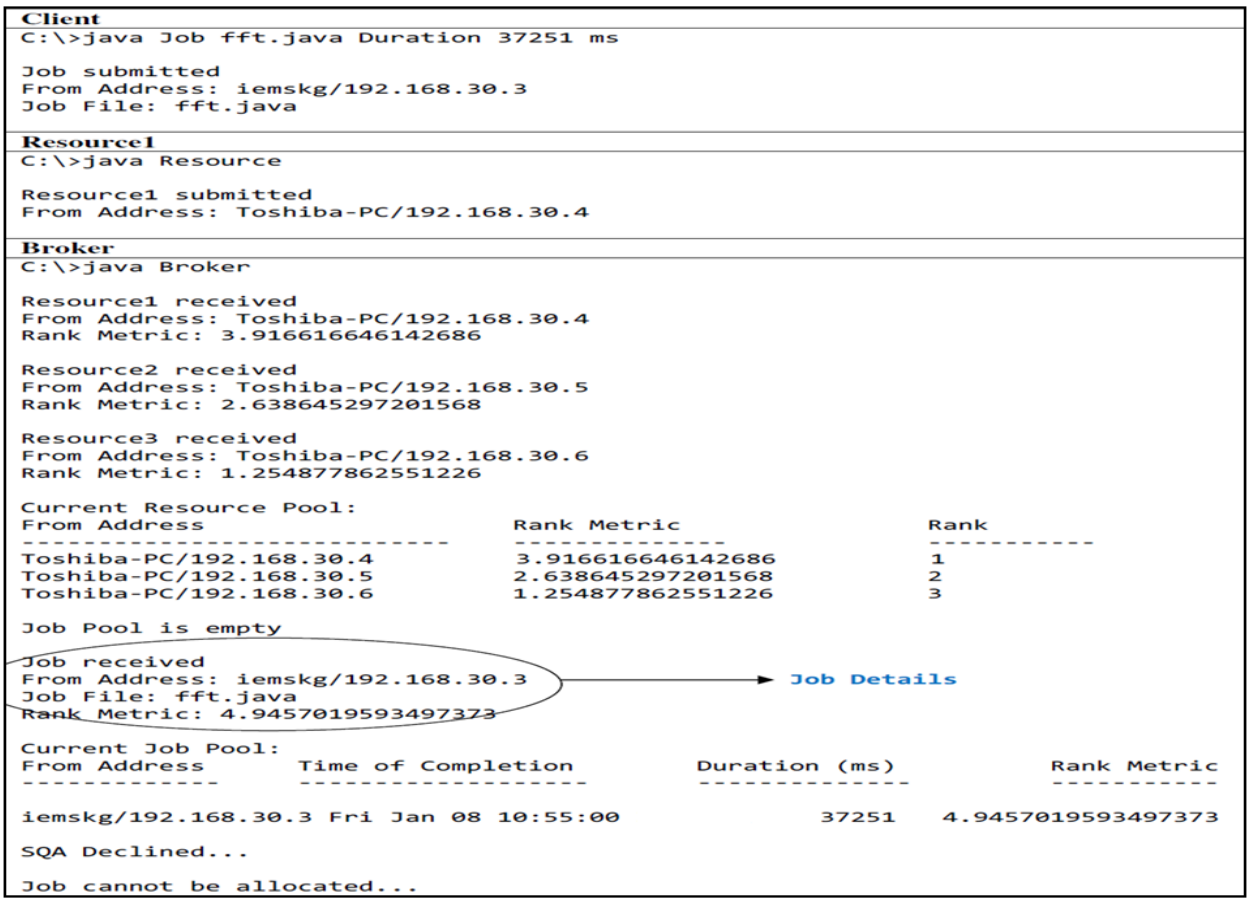

Figure 5 Restricted Provision of Scheduling

VI. CONCLUSION

Computational grid, being made of heterogeneous resources, facilitates computation-intensive scientific problem-solving. But, job scheduling and load balancing are the two major areas of concern in grid. Failure of resources during execution of jobs makes the situation even more complicated in grid. Major challenge faced in computational grid environment is workload balancing among the participating resources. This research aims to find a solution to this problem. This paper presents a resource allocation scheme using NDFS algorithm. In order to test the algorithm, a grid test bed is set up using Globus toolkit. This paper also presents the execution results of the computation intensive benchmark code of FFT. The scope of this paper is limited to submission and execution of single job in computational grid environment. In future this research will also deal with submission and execution of multiple jobs in the grid. This result demonstrates that at the time of scheduling of the specific job, load of each resources are balanced in computational grid environment. By the dynamic nature of grid environment, resources can fail at any time. In these circumstances obviously the resources are not balanced and the quality of the service is not maintained accordingly. So in future, this research will handle this failure situation where the job is executing at that point of time.

REFERENCES

[1] Buyya, R., Murshed, M.: GridSim: a toolkit for the modelling and simulation of distributed management and scheduling for Grid computing. The Journal of Concurrency and Computation: Practice and Experience 14 pp. 13–15, 2002.

[2] Foster, I., Kesselman, C., Tuccke, S.: The Anatomy of the Grid. International Journal of Supercomputer Applications, 2001.
[3] Goswami, S., Das, A.: Deadline Stringency Based Job Scheduling in Computational Grid Environment. In: Proceedings of the 9th INDIACom; INDIACom-2015. 11th to 13th March, 2015.
[4] Goswami, S., Das, A.: Handling Resource Failure towards Load Balancing in Computational Grid Environment. In: Fourth International Conference on Emerging Applications of Information Technology (EAIT 2014) at Indian Statistical Institute, Kolkata during Dec 19-21, 2014.
[5] Goswami, S., De Sarkar, A.: A Comparative Study of Load Balancing Algorithms in Computational Grid Environment. In:Fifth International Conference on Computational Intelligence, Modelling and Simulation, 2013, pp 99-104.
[6] Goswami, S., De Sarkar, A.: Service Oriented Load Balancing Framework in Computational Grid Environment. International Journal of Computers and Technology, Volume 9, Number 3, pp 1091 – 1098, 2013.
[7] Kant Soni, V., Sharma, R., Kumar Mishra, M.: An analysis of various job scheduling strategies in grid computing. In:2nd International Conference on Signal Processing Systems (ICSPS), 2010.
[8] Keerthika, P., Kasthuri, N.: A hybrid scheduling algorithm with load balancing for computational grid. International Journal of Advanced Science and Technology Vol.58, (2013), pp.13-28.
[9] Rajavel, R.: De-Centralized Load Balancing for the Computational Grid Environment. In: International Conference on Communication and Computational Intelligence, Tamil Nadu, India, 2010.
[10] Globus Toolkit. http://toolkit.globus.org
[11] http://introcs.cs.princeton.edu
[12] https://github.com/hyperic/sigar