Asian Journal of Information Technology 6 (5): 544-547, 2007


# Efficient Utilization of Computing Resources Using Highest Response Next Scheduling in Grid

K. Somasundaram, S. Radhakrishnan and M. Gomathynayagam
Department of CSE, A.K. College of Engineering, Krishnankoil, Tamilnadu, India

**Abstract:** Grid is a type of parallel and distributed computing environment. Alchemi is one of the open source toolkit for implementing grid environment. In Alchemi, Grid manager provide services associated with managing the execution of grid applications and their constituent thread. When a client node sends request to the manager or entry portal node, which distributes jobs among the executor or worker node. Here the threads are scheduled on a priority and First Come First Served (FCFS) basis. Most previous research on job scheduling for heterogeneous system considers the scenario where each job or task is mapped into single processor. In this study, we address the scheduling of parallel jobs in grid environment, where each site has a homogeneous cluster of processor, but processor at different sites has different speed. Here, we use Highest Response Next scheduling scheme where jobs are allotted to number of processor based on job's priority and processor's capability. This scheme is adaptive for local jobs and remote jobs without any loss of performance and highly adaptive for grid environment.

**Key words:** FCFS, Highest Response Next Scheduling Algorithm (HRN), grid computing, efficient utilization, resources

## INTRODUCTION

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities. According to the function, Grid is classified into three types: Computational Grid, Data Grid and Service Grid. Computational Grid is used to connect various computing resources on the network to construct a virtual high performance computer, which could offer high performance computing facilities (Ranganathan and Foster, 2002). The traditional computational Grid systems involve many technologies such as certification, task scheduling, communication protocols, fault tolerance and so on. The task of Grid resource broker and scheduler is to dynamically identify and characterize the available resources, select and allocate the most appropriate resources for a given job (Mitrani and Palmer, 2003). The resources are typically heterogeneous and locally administered and accessible under different local policies. Advance reservation (Foster *et al*., 2004) is currently being added to Portable Batch System (PBS).

Alchemi Grid Architecture (Akshay *et al*., 2005) has the following major components like: Manager, Executor, Owner and Cross Platform Manager. Manager manages the execution of grid application and provides services associated with managing thread execution. Here threads are scheduled on priority and FCFS basis. Executer accepts threads from manager and executes them. It can be either dedicated or non-dedicated. Grid applications are created by using owner component. Cross platform manager is an optional sub components of manager.

In this study, we present Highest Response Next (HRN) scheduling algorithm, where the highest priority job to choose suitable resource based on its CPU and Memory requirements.

## RELATED WORK

So far researchers have focused on allocating a single resource type (e.g., CPU usage) to jobs in the ready queue. The use of many scheduling algorithms has been limited due to restriction in application designs, runtime, system, or the job management system itself. Therefore, simple allocation scheme such as First Come First Serve (FCFS) or FCFS with First Fit back fill (FCFS/FF) are used in practice (Vijay *et al*., 2002).

Current jobs scheduling practices, support various resource allocations to a job and run to complete the scheduling. Scheduling policies are also heavily based on First-Come-First-Serve (FCFS) algorithms (Bitten *et al*., 2002). A FCFS scheduling algorithm allocates resources to jobs in the order that they arrive. The FCFS algorithm schedules the next job in ready queue as soon as sufficient system resources become available to meandl of

**Corresponding Author:** K. Somasundaram, Department of CSE, A.K.College of Engineering, Krishnankoil, Tamilnadu, India

the job requirements. The advantage of this provides the level of determinism on the waiting time of each job (Ememann *et al*., 2002). The disadvantage of FCFS shows up when the jobs at the head of the ready queue cannot be scheduled immediately due to insufficient system resources, but jobs further down the queue would be able to execute given the currently available resources. These latter jobs are essentially blocked from executing while the system resources are remaining idle (Hong *et al*., 2003).

AppLes (Fran *et al*., 1997) is a application level scheduler which builds agents for each application responsible for offering a scheduling mechanism. It concentrates to particular application whereas HRN focuses common to all applications

Nimrod/G (David *et al*., 2000; Rajkumar *et al*., 1993) is also an application level meta scheduler where a jobs to be submitted by a user and operate by selecting resources based on information available about those resources. It mainly concentrates on resource cost and deadline.

GrAds (Sathish *et al*., 2002) and Utopia (Songnian *et al*., 1993) are Global meta scheduler. It maintains account information about resources and other jobs. The resulting is the better application performance and resource utilization.

## HRN SCHEDULING APPROACH

In this approach, jobs are allotted based on its priority, its CPU and Memory requirements. We have used Highest Response Next (HRN) scheduling algorithm, to correct some of the weakness in both Shortest Job First and FCFS. In particularly, the excessive bias against longer jobs and the excessive favoritism toward short new job. HRN is a non preemptive discipline in which the priority of each job is a function not only of the job's service time but also of the amount of time the job has been waiting for service. Once a job gets the CPU cycle, it runs to completion.

## HRN SCHEDULING MODEL

HRN is a non preemptive and priority based scheduling. Here priority is calculated dynamically. Dynamic priority in HRN is calculated according to the following formula:

$$Tp_i=(Tw_i+Ts_i) / Ts_i$$

Where

Tpi = Priority of each job. (i=0,1,..n Jobs)
Twi = Waiting time of each Jobs (i=0,1,..n Jobs)
Tsi = Service time of each jobs (i=0,1,..n Jobs)

In HRN, Service time is also calculated dynamically with starting and finishing time of each jobs.

$$Ts_i= Tf_i - Tst_i$$

Where,

$Ts_i$ = Service time of each jobs (i=0, 1, n Jobs)
Tfi = Finishing time of each Jobs (i=0,1,..n Jobs)
$Tst_i$ = Starting time of each jobs (i=0,1,… n Jobs)

Because the service time appears in the denominator, shorter jobs will get preference. But the waiting time appears in numerator, longer jobs that have been waiting and also give favorable treatment to them. Hence the total system's response time is calculated as:

$$R=\sum_{i=0}^{n} Tw_i + Ts_i$$

The following Pseudo code shows the algorithm of Highest Response Next scheduling.

```
procedure main()
begin:
        int j=0;
        for i=1 to n jobs
        begin:
        initially all jobs service time = 0;
        j= (j+1)%10000;
        if (j==0)
        begin:
                job is in processing state;
               call processQ sub procedure;
else
                job is added in queue;
                call add process sub procedure;
end;
end;
end;
end procedure;
```

```
procedure addprocess(unsigned servicetime)
begin:
for i=1 to n jobs;
begin:
if(current job's service time= = 0)
begin
current job's service time= service time;
        current job's waittime=0;
        else
        slot is filled
end;
end;
end;
end procedure;
```

```
procedure processQ()
begin:
    low=0;
    for i=1 to n jobs
    begin:
    if (job's service time!=0)
    begin:
    priority = ( job's waiting time + job's service time)/ (job's
    service time);
    if ((i==1) || (low < priority))
    begin:
        low = priority;
        id=I;
    end;
        job's wait time ++;
    end;
    end;
    if(id==-1)
        begin:
        return;
        end;
        job's service time--;
        if (job's service time ==0)
        begin:
        job is removed from the slot;
        end;
    end;
    end procedure;
```

Table 1: CPU, memory and priority table of each jobs

| Scheduling epochs | FCFS | HRN |
|---|---|---|
| 1 | j0,j1 | J0,j2 |
| 2 | j2 | J3,j1 |
| 3 | j3 | J5,j4 |
| 4 | j4,j5 | |

Table 2: Job queue ordering

| Jobs | CPU requirement | Memory requirement | Waiting time | Service time | Priority TP= (Ts+ Tw)/Ts |
|---|---|---|---|---|---|
| J0 | 8 | 4 | 0 | 0.125 | 1 |
| J1 | 4 | 2 | 0.125 | 0.0625 | 4 |
| J2 | 7 | 16 | 0.1875 | 0.5 | 2 |
| J3 | 11 | 20 | 0.6875 | 0.625 | 3 |
| J4 | 1 | 12 | 1.3125 | 0.375 | 5 |
| J5 | 1 | 10 | 1.6875 | 0.3125 | 6 |

Table 3: FCFS scheduling

| | | FCFS | |
|---|---|---|---|
| Epochs | Jobs to be completed | Tot.CPU requirement | Memory requirement |
| 1 | j0, j1 | 12 | 6 |
| 2 | j2 | 7 | 16 |
| 3 | j3 | 11 | 20 |
| 4 | j4, j5 | 2 | 22 |

## PERFORMANCE ANALYSIS OF HRN WITH FCFS

As an example given in the Table 1, it shows the memory and CPU requirements, waiting and service time of each jobs. The job allocation scheme maps the 6 jobs in job queue to two resource system with 16 CPU and 32 Gbytes of memory. Assume that the order in the job queue represents the order of arrival and that each job requires the different amount of execution time. Under the assumption, a job allocation scheme would select a set of jobs for execution during the scheduling epochs. The number of epochs required to schedule all jobs in the job queue is used to compare different job allocation scheme. Table 2 shows the jobs allocated to each scheduling epochs for FCFS and HRN. The FCFS allocation scheme allocates jobs 0 and 1 in first epoch, but it can not allocate job 2, due to total CPU requirement of three jobs being greater than the system provides (8+4+7 >16). In FCFS, jobs are processed as it is coming and it takes approximately 4 epochs to complete all the jobs.

In HRN, jobs are allocated based on its priority and resource availability. Hence lowest service time jobs will get highest priority to get CPU for processing. For ex, jobs j0, j2 are completed in first epoch. So, the CPU and Memory requirements of first epoch are only (8+7< 16) and (4+16< = 20). Jobs j3 and j1 are completed in second epoch and jobs j5 and j6 are completed with in third epoch only. So over all TAT (Turn Around Time) of HRN is less than over all TAT of FCFS.

Table 3 and 4 shows the total CPU requirement and memory requirement for FCFS and HRN scheduling respectively. From the table, the HRN scheduling algorithm completes all the jobs within three epochs and forth epochs it ready to accept new jobs, but in FCFS takes four epochs to complete all the jobs.

Table 4: HRN scheduling

| | | HRN | |
|---|---|---|---|
| Epochs | Jobs to be completed | Tot.CPU Requirement | Memory Requirement |
| 1 | j0, j2 | 15 | 20 |
| 2 | j1, j3 | 15 | 22 |
| 3 | j4, j5 | 2 | 22 |
| 4 | - | - | - |

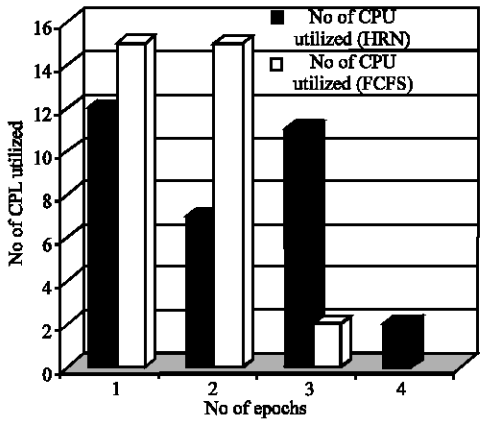


Fig. 1: CPU utilization of FCFS and HRN

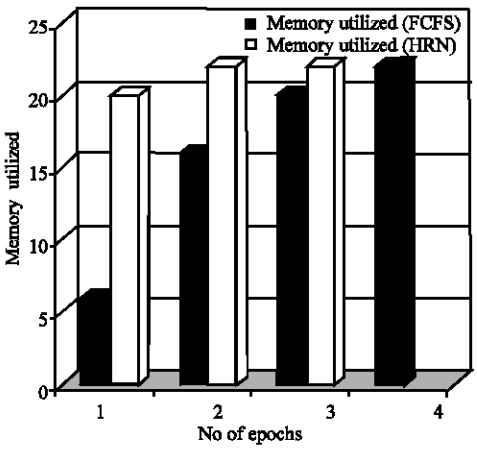


Fig. 2: Memory utilization of FCFS and HRN

Figure 1 and 2 shows the CPU utilization and memory utilization of FCFS and HRN scheduling. The system provides maximum of 16 CPUs, but FCFS can utilize the maximum of 12 CPUs due the nature of its scheduling. The CPU utilization of FCFS comes only 75% only. In HRN with priority, it utilize maximum of 15 CPUs and complete all the jobs within three epochs and its CPU utilization comes around 94%. From this result, we can conclude that HRN with priority will effectively utilize the available resources and complete all the jobs quickly. This algorithm will effectively work well in grid environments.

## CONCLUSION

HRN is a new scheduling algorithm in Grid which provides more responses with time, memory and CPU requirements. We have identified that the HRN model for scheduling system is much adaptive for Grid environment. In future, we can modify the HRN at priority level for reducing the HRN's over all Turn Around Time (TAT) of job completion from 3 epochs to 2 epochs and avoid memory and CPU wastage in each epochs.

## REFERENCES

Akshay Luther, Rajkumar Buyya, Rajiv Ranjan and Srikumar Venugopal, 2005. Alchemi: A. NET-based Enterprise Grid Computing Systems, Proceedings of the 6th International Conference on Internet Computing (ICOMP), Las Vegas, USA., pp: 27-30.

Akshay Luther, Rajkumar Buyya, Rajiv Ranjan and Srikumar Venugopal, 2005. Alchemi: A. NET-based Grid Computing Framework and its Integration into Global Grids. Proceedings of the 6th International Conference on Internet Computing (ICOMP), Las Vegas, USA., pp: 27-30.

Bitten, C., J. Gehring *et al*., 2000. The NRW-Meta Computer: building block for a worldwide computational Grid, proceeding of the 9th Heterogeneous Computing Workshop, pp: 31-40.

David Abramson, Jon Griddy and Lew Kotler, 2000. High Perfromance Paremetric Modeling with Nimrod/G: Killer application for the Global Grid?, Int International Parallel and Distributed Processing Symposium (IPDPS), Cancun, Mexico, pp: 520-528.

Ememann, C., V. Hamscher *et al*., 2002. On Advantageous of Grid Computing for parallel job scheduling, proceeding 2nd IEEE/ACM Int'. Symp. On cluster computing and the Grid (CCGRID 2002), Berlin, IEEE Press.

Foster, I. *et al*., 2004. The Grid 2003 Production Grid : Principles and Practice, 13th International Symposium on High Performance Distributed Computing.

Fran Berman, F. and R. Wolsky, 1997. AppLes Project: A Status Report, Proceeding of 8th NEC Research Symposium, Germany.

Hong Zhang Shan, Leonid Oliker *et al*., 2003. Job super Scheduler Architecture and Performance in Computational Grid Environments ACM: Version, pp: 15-21.

Mitrani, I. and J. Palmer, 2003. Dynamic Server Allocation Heterogenous Clusters, 1st International working conference on Heterogeneous Networks, Ilkley,UK.

Rajkumar Buyya, David Abramson, Jon Giddy, Nimrod/G, 2000. An Architecture for a Resource Management and Scheduling System in a Global Computational Grid, In: Proceedings of the HPC ASIA the 4th International Conference on High Perfromance Computing in Asia-Pacific Region, Beijing, China, IEEE Computer Society Press, USA.

Ranganathan, K and I. Foster, 2002. Decoupling Computation and Data Scheduling in Data Intensive Applications, 11th International Symposium on High Perfromance Distributed Computing, Ediburgh, Scotland, Condor Project, Condor-G.

Sathish S. Vadhiyar and Jack J. Dongarra, 2002. A Meta Scheduler for the Grid, In: Proceeding of the 11th IEEE International Symposium on High Perfromance Computing, Edinburgh, Scottland, IEEE. Computer Soc., pp: 343.

Songnian Zhou, Xiaohu Zheng, Jingwen Wang and Pierre Delisle, 1993. Utopia: A Load-Sharing facility for large heterogeneous distributed computing systems, Software-Practice and Experience, 23: 1305-1336.

Vijay Subramanian, Rajkumar Kettimuthu *et al*., 2002. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests, IEEE.