EFFICIENT SCHEDULING STRATEGY USING COMMUNICATION AWARE SCHEDULING FOR PARALLEL JOBS IN CLUSTERS

A.Neela madheswari[1] and R.S.D.Wahida Banu[2]

[1]Department of Information Technology, KMEA Engineering College, Aluva, India
[2]Principal, Government College of Engineering, Salem, India

ABSTRACT

In the area of Computer Science, Parallel job scheduling is an important field of research. Finding a best suitable processor on the high performance or cluster computing for user submitted jobs plays an important role in measuring system performance. A new scheduling technique called communication aware scheduling is devised and is capable of handling serial jobs, parallel jobs, mixed jobs and dynamic jobs. This work focuses the comparison of communication aware scheduling with the available parallel job scheduling techniques and the experimental results show that communication aware scheduling performs better when compared to the available parallel job scheduling techniques.

KEYWORDS

Scheduling, simulation, communication aware scheduling, local scheduling, explicit coscheduling, implicit coscheduling

1. INTRODUCTION

Along with the power of parallelization in supercomputers and massively parallel processors comes the problem of deciding how to utilize this increased processing capability efficiently. In a parallel computing environment, multiple processors are available to execute collectively submitted jobs [7]. In this work, the concentration is given to jobs submitted by users.

While considering the user jobs that are to be scheduled under high performance computing environments such as clusters or super computers, it is important to take an ultimate care for selecting the scheduling strategy. Since scheduling plays an important role in improving the system performance. Otherwise, the system can be degraded due to the under utilization of existing resource, load imbalancing, more time taken for completing the jobs, etc.

We can classify the user jobs as two forms according to the nature of its execution. They are namely: serial jobs and parallel jobs. Serial job can contain single process whereas the parallel job can contain a set of processes. Serial jobs are independent jobs and these jobs can be scheduled to any of the available processors. The parallel jobs are classified into two types namely: batch and coscheduled. The processes involved in the batch jobs are independent whereas the processes involved in coscheduled jobs are depend among themselves. Batch jobs can be scheduled using any of the existing techniques such as [1], [3], [6], but for coscheduled jobs, if the same techniques mentioned in [1], [3] and [6], then the system performance can be slowed down due to communication overhead.

2. MOTIVATION

Traditional scheduling policies for parallel systems focus on treating differently for interactive versus batch jobs in order to maximize the utilization of an expensive system. Because it will reduces resource fragmentation and increases system utilization. Users are expected to provide nearly accurate estimates of job execution times [14].

Peter Strazdins and John Uhlmann has been compared local scheduling with gang scheduling. Experiments on a Beowulf cluster with 100Mb fast Ethernet switches are made comparing the gang scheduling with local scheduling. Results for communication-intensive numerical applications on 16 nodes reveal that gang scheduling results in slowdowns up to a factor of two greater for 8 simultaneous jobs. The application studies showed that, when considering scheduling policies, both memory usage and communication patterns have an important effect on overall throughput. These factors should not be neglected [2].

A generic framework for deploying coscheduling techniques has been proposed by Saurabh, et al. All the types of explicit coscheduling techniques such as dynamic coscheduling, spin block and periodic boost and a co-ordinated coscheduling were proposed. For all the coscheduling techniques, the mixture of jobs is considered such as CPU intensive and parallel jobs, I/O intensive and parallel jobs. For performance evaluation, execution time, overhead and slowdown were considered. The scheduling techniques perform well for CPU intensive and parallel jobs but for the I/O intensive and parallel jobs, performance degrades [1].

In general, the workload for job scheduling techniques is divided into two main categories namely: serial jobs and parallel jobs. The parallel jobs are again subdivided into batch jobs or coscheduled jobs. A batch job contains a set of independent processes, whereas coscheduled job contains a set of dependent processes. The dependencies may be due to communication, I/O or synchronization requirement.

The processes involved under a coscheduled job are related with themselves. The workflow of the processes of the parallel job can be in any of the types mentioned in [12], [5]. This work considered the flow of processes under coscheduled job to be of type workflow chain [5] as mentioned in Figure 1. The starting process should complete its execution and then the next process will start its execution and it continues till the last process under that coscheduled job and thus the execution will complete for every coscheduled job.

In some kind of workflows, this form of workflow chain is common. Consider the workflow available in High Energy Physics [8]. The HEPTrails workflow contains modules that support streaming workflows that process data as it becomes available and make the data available for further processing. Each module's individual operations are fine-grained, the module's communication overhead can be greater than operations themselves.

In many parallel job scheduling techniques, the workload considered is focused on serial jobs [9], serial and I/O intensive jobs [1], batch jobs [3], [10] and the performance is evaluated. But considering the communication intensive jobs is lagging. This work concentrates on scheduling the communication intensive jobs on clusters for reducing the communication overhead among various processors in clusters and thus to achieve better system utilization.

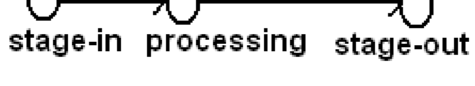

Figure 1. Workflow Chain

If the parallel jobs are in need of communicating between themselves, then using the existing techniques [1], [3], there will be performance degradation due to communication overhead. To avoid the communication overhead and to improve the performance of the cluster or any high performance computing environment, communication aware scheduling (CAS) is devised [15]. In CAS, coscheduled job is considered for parallel job. But the comparisons with the existing techniques are not done. This paper focuses the comparison of the communication aware scheduling with local scheduling (LS), extended implicit coscheduling (ICS) and extended explicit coscheduling (ECS) techniques.

3. System Design

An important goal of any parallel system scheduler is to promote the productivity of its users. To achieve high productivity, the scheduler has to keep its users satisfied and motivate them to submit more jobs. Due to high costs involved in deploying a new scheduler, it is uncommon to experiment with new designs in reality for the first time. Instead, whenever a new scheduler is proposed, it is first evaluated in simulation, and only if it demonstrates significant improvement in performance can then become a candidate for an actual deployment. The role of simulation is thus crucial for the choices made in reality [11]. Here the scheduling system is simulated and monitored using Java with version jdk1.6.

3.1. System Setup

The cluster system of 124 nodes is considered for system environment, in which one node act as a head node, while other nodes are compute nodes. The head node is responsible for scheduling the jobs to the compute nodes. Users have to submit the jobs to the head node. It is assumed that the scheduler is able to schedule a job per second. The entire cluster system is assumed to be fully connected and no communication delay exists. The head node or scheduler can schedule a job per second to the available nodes. The selection of node is based on the scheduling algorithm used. It is shown in Figure 2.

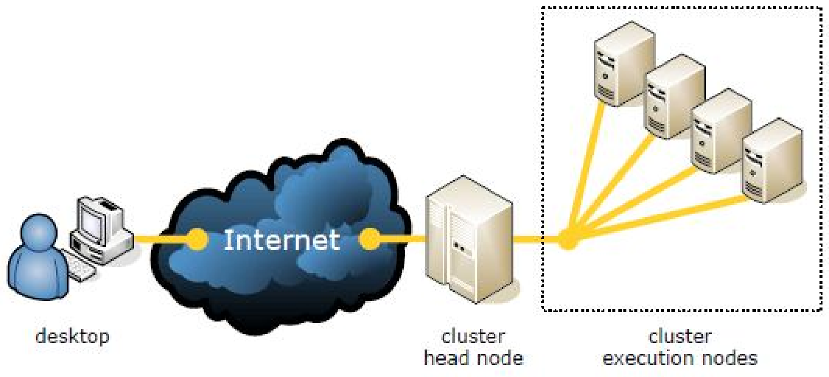

Figure 2. Architecture of a typical Cluster

3.2. Workload Consideration

The performance of a computer system depends not only on its design and implementation, but also on the workload to which it is subjected. Different workloads may lead to different performance and in some cases to different relative ranking of systems or designs. Using representative workloads is therefore crucial in order to obtain reliable performance evaluation results. One way to obtain representative workloads is to use real workloads from production systems [4]. Here the real workload from Grid5000 is considered for performance evaluation of the system [13].

4. SCHEDULING ALGORITHMS

There are various literatures that describe different scheduling algorithms. They are based on the workload consideration as the key point. Communication aware scheduling algorithm is devised by considering the different kinds of workloads such as serial jobs or parallel jobs [15]. The performance is evaluated using two parameters namely: idle time of nodes and wait time of jobs. There are three important parallel job scheduling algorithms namely: local scheduling, explicit coscheduling and implicit coscheduling [3].

4.1. Sample Scenario of Workload

To know the details of the algorithm in a clear manner, let us consider a sample workload of 6 jobs out of which the jobs with job id 1, 3 and 6 are serial and the jobs with job id 2, 4 and 5 are parallel in nature and their dependencies, runtime is shown in Table 1. Cosched_id shows the order in which the parallel job has to run. Let us consider we are having 4 compute nodes under which the given jobs are to be allocated and they are specified as n1, n2, n3 and n4.

Table 1. Sample Jobs with runtime

Job_id	Job type	Cosched_id	Job_id for graph	Runtime(seconds)
1	Serial	-	1	5
2	Parallel	1	21	1
2	Parallel	2	22	1
2	Parallel	3	23	1
2	Parallel	4	24	1
3	Serial	-	3	5
4	Parallel	1	41	2
4	Parallel	2	42	2
5	Parallel	1	51	1
5	Parallel	2	52	1
5	Parallel	3	53	1
6	Serial	-	6	5

4.2. Local Scheduling

In this scheduling strategy, the jobs are scheduled similar to First Come First Serve basis. Any incoming job is scheduled to the available nodes in a serial order i.e. the nodes from node 1 through node 123 are filled according to the nodes that are free. Any kinds of jobs can be scheduled using this strategy. Thus serial jobs, parallel jobs, mixed jobs and dynamic jobs are scheduled and their performance are evaluated by finding the idle time of nodes and wait time of jobs. For the sample workload mentioned in Table 1, the scheduling of jobs using local scheduling is given in Figure 3, with the assumption that scheduler will take one second for every job to submit to the worker node.

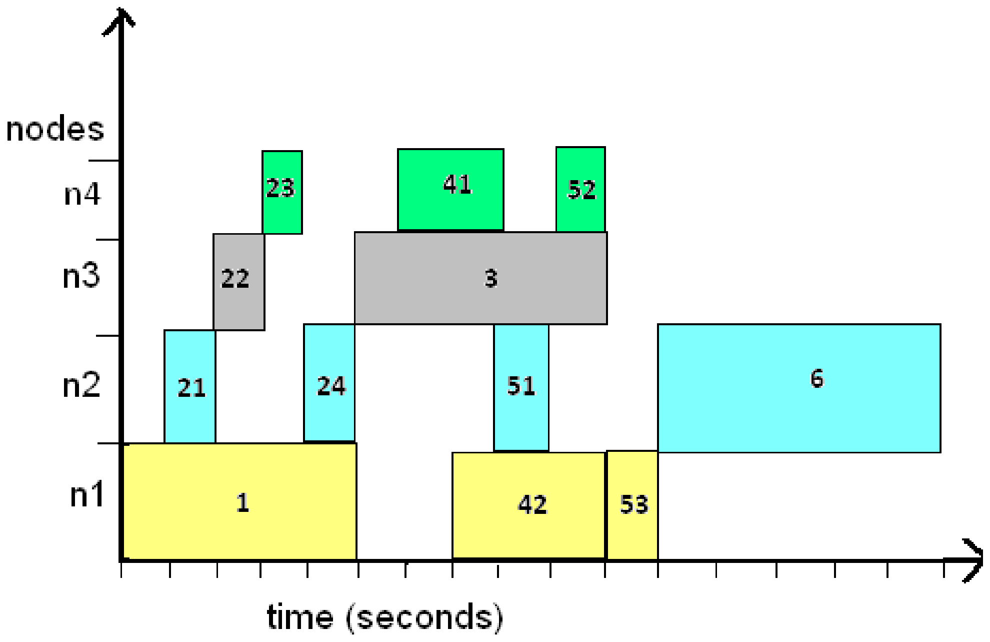

Figure 3. Local scheduling for sample workload

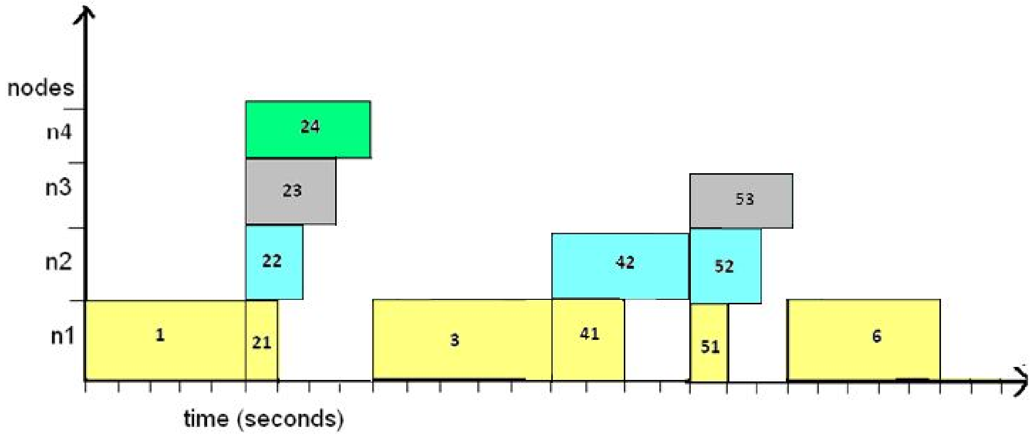

Figure 4. Explicit coscheduling for sample workload

4.3. Extended explicit coscheduling

This technique is developed specifically for scheduling the parallel jobs. Each parallel job is split into number of processes and every process is assigned to the processors at the same time. After the completion of a single parallel job, the next parallel job will enter into the system for execution. Here we are considering a system with 123 nodes for processing. A slight modification is made for this work in the algorithm and the name of algorithm is called Extended explicit coscheduling. Instead of scheduling a single parallel job, the system is assumed to schedule with n number of jobs if the system is ready to occupy n number of jobs at a time. For the sample workload mentioned in Table 1, the scheduling of jobs using explicit coscheduling is given in Figure 4 and using extended explicit coscheduling is given in Figure 5. While using explicit coscheduling, more number of time slots are left idle. Refer Figure 4. Hence the extension to this algorithm is made as extended explicit coscheduling where the idle time slots are less compared to the idle time slots of implicit coscheduling. Refer Figure 5.

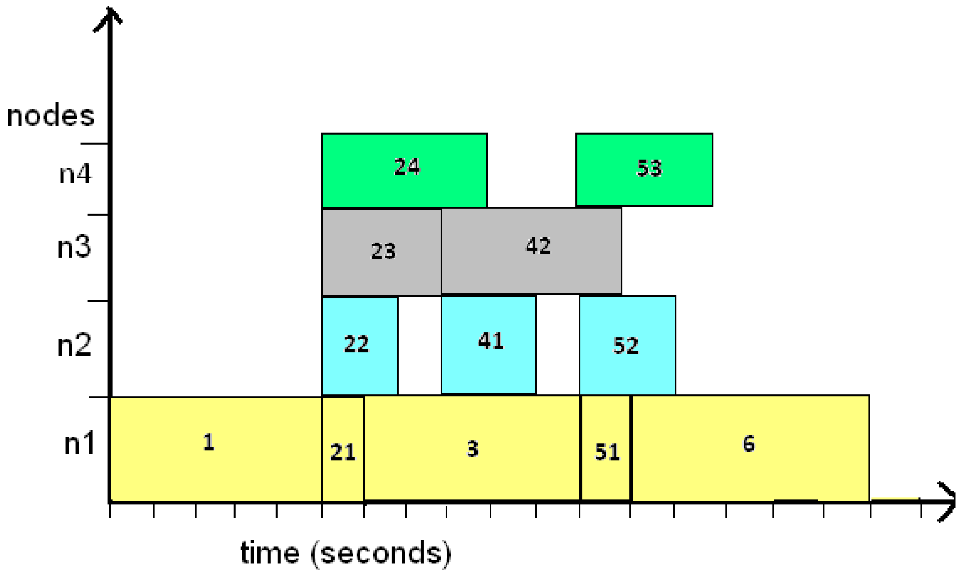

Figure 5. Extended explicit coscheduling for sample workload

4.4. Extended implicit coscheduling

This technique is developed specifically for scheduling the parallel jobs. Each parallel job is split into number of processes and every process is assigned to the processors but with small time difference between them. Here also the modification is carried out for this work i.e. instead of scheduling a single parallel job, the system can able to handle n number of jobs if the n number of jobs can be accommodated at a time. The algorithm contains the modification is called extended implicit coscheduling. For the sample workload mentioned in Table 1, the scheduling of jobs using implicit coscheduling is given in Figure 6 and using extended implicit coscheduling is given in Figure 7. While using implicit coscheduling, more number of time slots are left idle. Refer Figure 6. Hence the extension to this algorithm is made as extended implicit coscheduling where the idle time slots are less compared to the idle time slots of implicit coscheduling. Refer Figure 7.

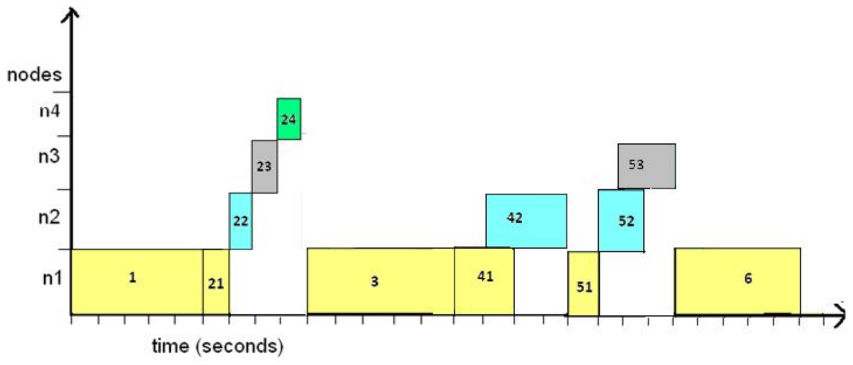

Figure 6. Implicit coscheduling for sample workload

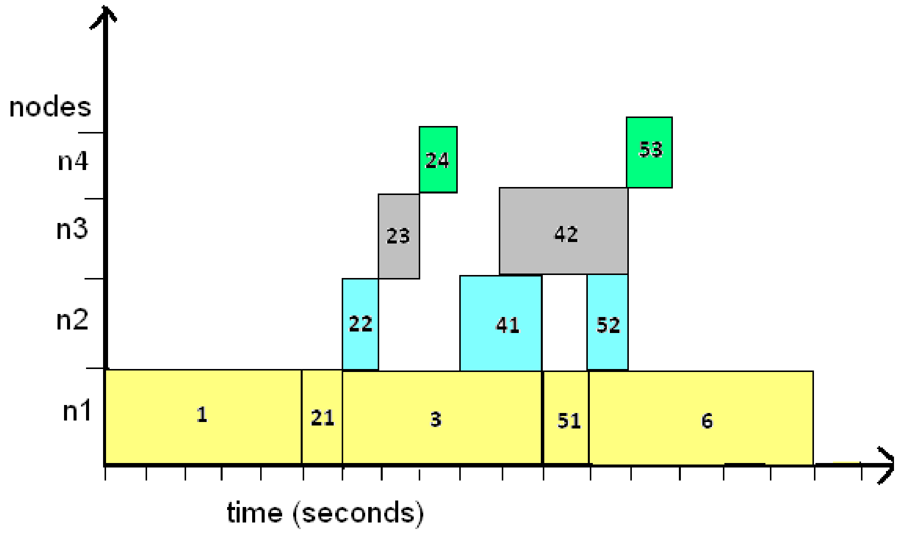

Figure 7. Extended implicit coscheduling for sample workload

4.5. Communication aware scheduling

This technique is developed for handling different kinds of workloads such as serial jobs, parallel jobs, mixed jobs and dynamic jobs [15]. For communication aware scheduling, the runtime for sample workload given in Table 1 is given in Figure 8.

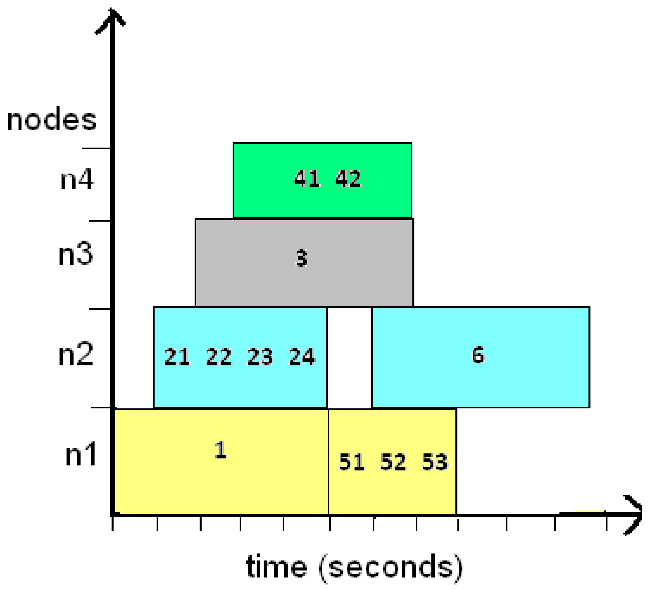

Figure 8. Communication aware scheduling for sample workload

According to the nature of the scheduling algorithms, the usage for various workloads is specified in the Table 2. 'A' specifies that the particular scheduling algorithm is applicable for the specified workload and 'NA' specifies that the particular algorithm is not applicable for that specified workload.

Table 2. Scheduling Algorithms Usage

Types of Workloads	**Scheduling Algorithms**			
	LS	**ECS**	**ICS**	**CAS**
Serial	A	NA	NA	A
Parallel	A	A	A	A
Mixed	A	NA	NA	A
Dynamic	A	NA	NA	A

5. EXPERIMENTAL RESULT

The performance of communication aware scheduling is evaluated by comparing the performance with local scheduling, extended explicit coscheduling and extended implicit coscheduling techniques based on the wait time of jobs and idle time of nodes. The entire work is classified according to the type of jobs considered.

5.1. Scheduling Serial Jobs

Here the first 300 serial jobs from Grid5000 workload are considered. The scheduler is filled with first 300 jobs. At the end of scheduling, the average idle time of nodes and the average wait time of jobs are calculated using communication aware scheduling and local scheduling, the results are shown in Figure 9 and Figure 10 respectively.

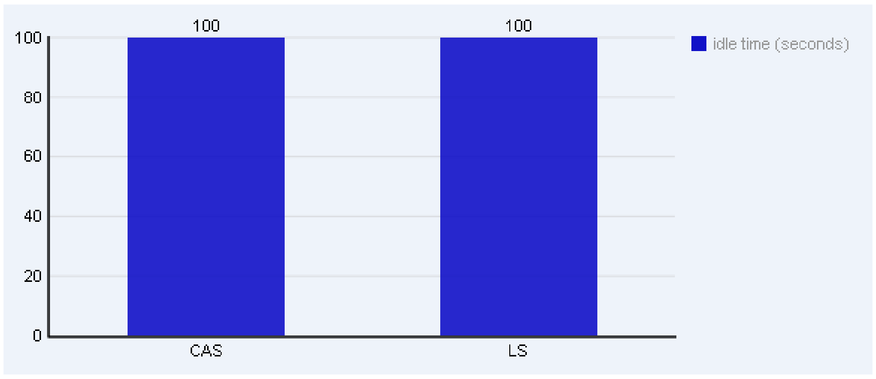

Figure 9. Average idle time of nodes at the end of scheduling serial jobs

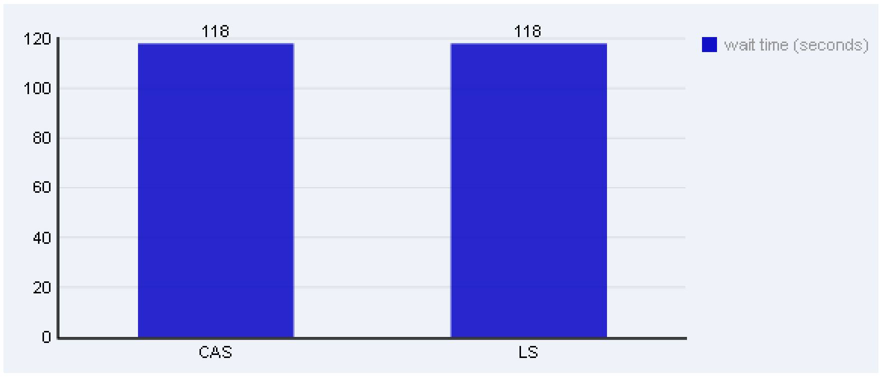

Figure 10. Average wait time of jobs at the end of scheduling serial jobs

5.2. Scheduling Parallel Jobs

Here the first 300 parallel jobs are from Grid5000 is considered. The scheduler is filled with first 300 parallel jobs. At the end of scheduling, average idle time of nodes and the average wait time of jobs are calculated using communication aware scheduling, local scheduling, extended explicit coscheduling and extended implicit coscheduling, the results are shown in Figure 11 and Figure 12 respectively.

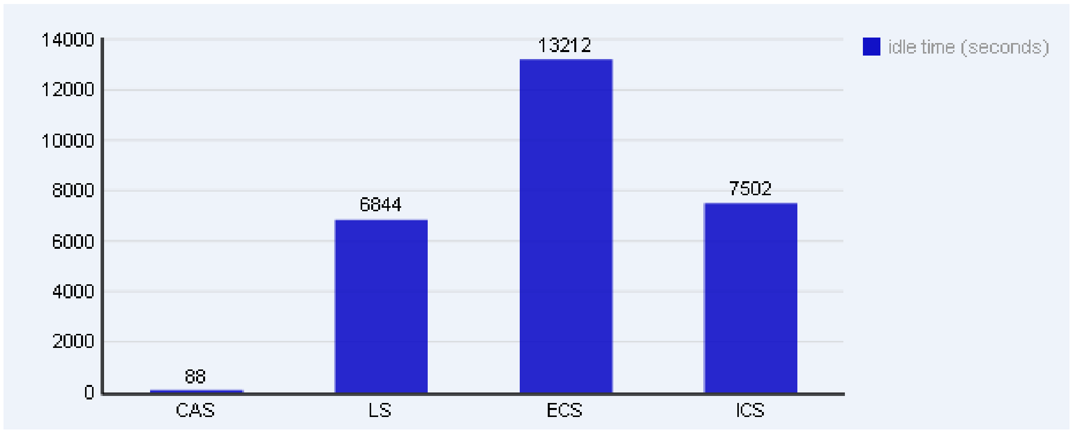

Figure 11. Average idle time of nodes at the end of scheduling parallel jobs

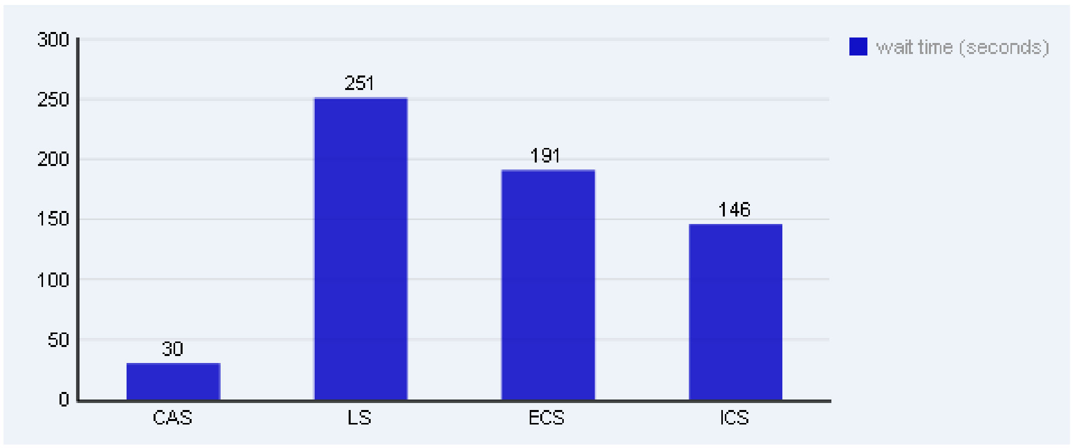

Figure 12. Average wait time of jobs at the end of scheduling parallel jobs

5.3. Scheduling Mixed Jobs

Here equivalent mixture of 150 serial jobs and 150 parallel jobs is considered. The scheduler is designed such that it can able to schedule 25 equivalent mixture of serial and parallel jobs in an alternative manner. At the end of scheduling, the average wait time of jobs and average idle time of nodes are calculated using communication aware scheduling and local scheduling and the results are shown in Figure 13 and Figure 14 respectively.

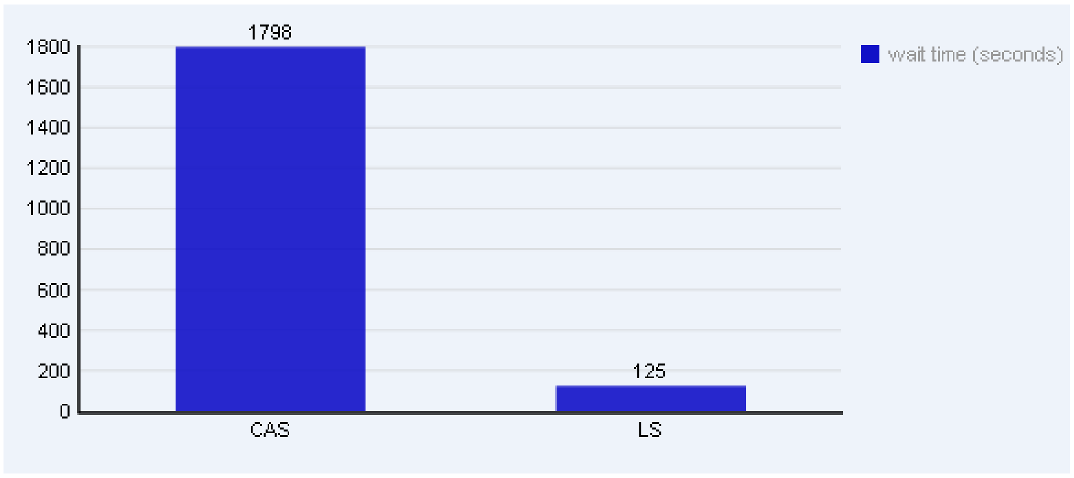

Figure 13. Average wait time of jobs at the end of scheduling mixed jobs

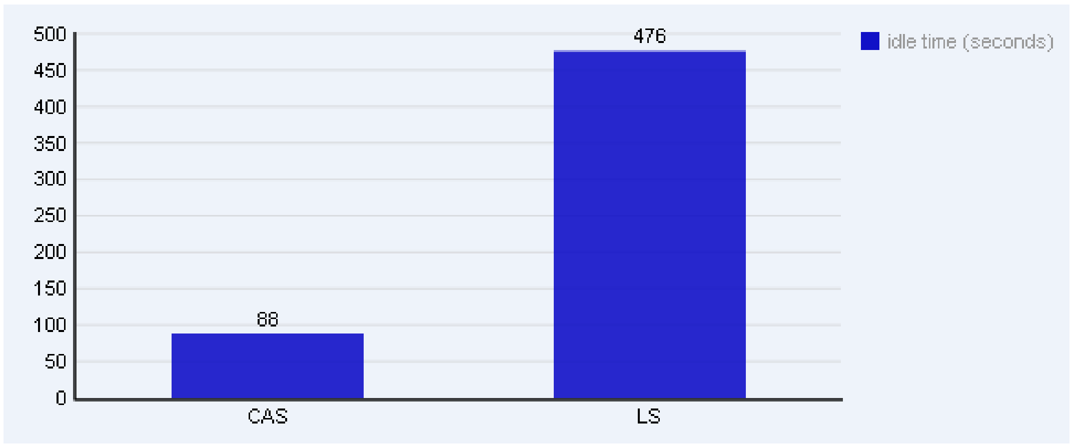

Figure 14. Average idle time of nodes at the end of scheduling mixed jobs

5.4. Scheduling dynamic jobs

If the user is not sure about the number of serial or parallel jobs, then he can classify the incoming jobs under this category. The scheduler can able to schedule any job combination whether it is only serial, only parallel, equal mixture of serial and parallel or any mixture of serial and parallel job types. For this kind of workload, the scheduler is filled with random 300 jobs from Grid5000 workload. At the end of scheduling, the average wait time of jobs and average idle time of nodes are calculated using communication aware scheduling and local scheduling and the results are shown in Figure 15 and Figure 16 respectively.

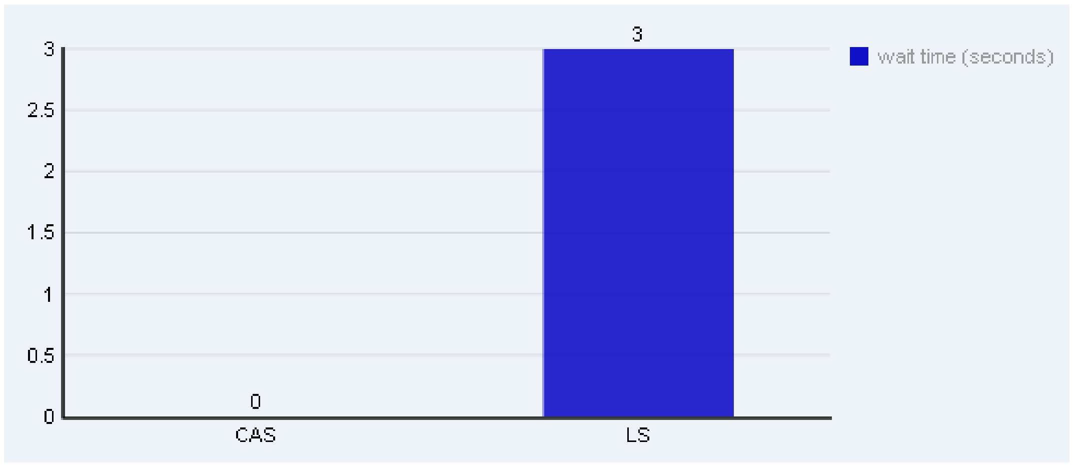

Figure 15. Average wait time of jobs at the end of scheduling dynamic jobs

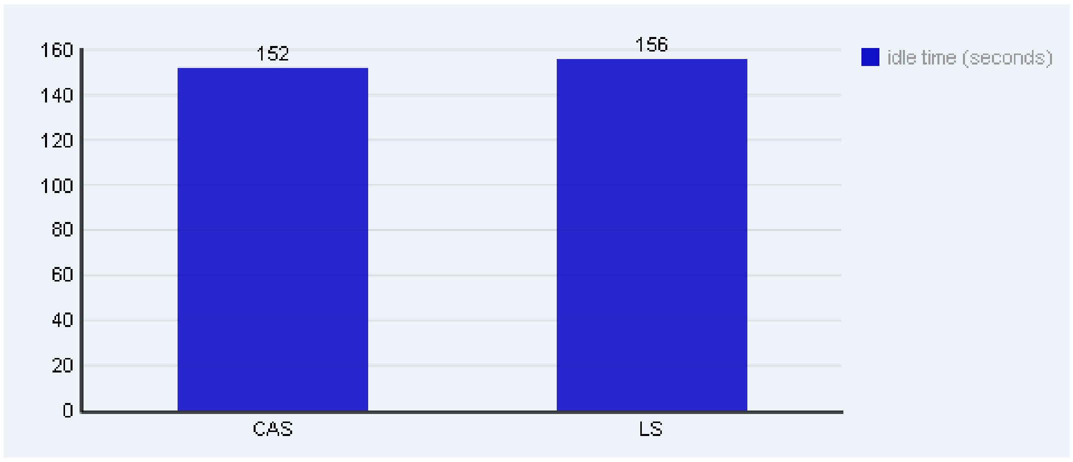

Figure 16. Average idle time of nodes at the end of scheduling dynamic jobs

We can compare the performance according to the types of jobs while scheduling. While scheduling serial jobs, by considering Figure 9, it is observed that average idle time of nodes is 100 seconds for communication aware scheduling and for local scheduling also the same value obtained. From Figure 10, it is observed that average wait time of jobs is 118 seconds for communication aware scheduling and for local scheduling also the same value obtained.

While scheduling parallel jobs, by considering Figure 11, it is observed that average idle time of nodes is 88 seconds for communication aware scheduling whereas for local scheduling, it is 6844 seconds, for extended explicit coscheduling, it is 13212 seconds and for extended implicit coscheduling, it is 7502 seconds. From Figure 12, it is observed that average wait time of jobs is 30 seconds for communication aware scheduling whereas for local scheduling, it is 251 seconds, for extended explicit coscheduling, it is 191 seconds and for extended implicit coscheduling, it is 146 seconds.

While scheduling mixed jobs, by considering Figure 14, it is observed that average idle time of nodes is 88 seconds for communication aware scheduling whereas for local scheduling, it is 476 seconds. By considering the Figure 13, it is observed that average wait time of jobs is 1798 seconds for communication aware scheduling whereas for local scheduling, it is 125 seconds. While scheduling dynamic jobs, by considering Figure 16, it is observed that average idle time of nodes is 152 seconds for communication aware scheduling and for local scheduling, it is 156 seconds. From Figure 15, it is observed that the average wait time of the jobs is 0 second for communication aware scheduling and for local scheduling, it is 3 seconds.

6. CONCLUSION

Hence we can conclude that communication aware scheduling performs equal to local scheduling in case of serial jobs. In the case of parallel jobs, communication aware scheduling performs better than local scheduling, extended explicit coscheduling and extended implicit coscheduling techniques. For mixed jobs, communication aware scheduling gave more wait time compared to local scheduling whereas in dynamic jobs, communication aware scheduling performs well when compared to local scheduling. Hence it is concluded that communication aware scheduling is well suited for parallel job scheduling when compared to local scheduling, extended explicit coscheduling and extended implicit coscheduling techniques. Also, communication aware scheduling is able to handle different kinds of job types.

REFERENCES

[1] Saurabh Agarwal, Gyu Sang Choi, Chita R. Das, “Co-ordinated Coscheduling in Time-Sharing Clusters through a Generic Framework”, Proceedings of the IEEE International Conference on Cluster Computing CLUSTER, pp.84-91, 2003.

[2] Peter Strazdins, John Uhlmann, “A Comparison of Local and Gang Scheduling on a Beowulf Cluster”, In Proceedings of IEEE International conference on Cluster computing, pp.55-62, 2004.

[3] Eitan Frachtenberg, Dror G. Feitelson, Fabrizio Petrini, Juan Fernandez, “Adaptive parallel job scheduling with flexible coscheduling”, IEEE Transactions on Parallel and Distributed Systems, Vol. 16, pp.1066- 1077, 2005.

[4] Dan Tsafrir, Dror G.Feitelson, “Instability in Parallel Job Scheduling Simulation: The role of Workload Flurries”, IEEE, 2006.

[5] L.Schley, “Prospects of Co-allocation Strategies for a LightWeight Middleware in Grid Computing”, Proceedings of the 20th International Conference on Parallel and Distributed Computing and Systems PDCS, ACTA Press, pp. 198-205, 2008.

[6] Georgios L.Stavrinides, Helen D.Karatza, “Performance Evaluation of Gang scheduling in Distributed Real-Time Systems with Possible Software faults”, IEEE SPECTS, pp.no:1-7, 2008.

[7] Richard A.Dutton, Weizhen Mao, Jie Chen, William Watson III, “Parallel Job Scheduling with Overhead: A Benchmark Study”, IEEE, 2008.

[8] Andrew Dolgert, Lawrence Gibbons, Christopher D.Jones, Valentin Kuznetsov, Mirek Riedewald, Daniel Riley, Gregory J.Sharp, Peter Wittich, “Provenance in High-Energy Physics Workflows”, Computing in Science and Engineering (CiSE), Issue 3, pp. 22-29, May/June, 2008.

[9] Sudha S.V., Thanushkodi K, “An Approach for Parallel Job Scheduling using Nimble Algorithm”, IEEE International Conference on Computing, Communication and Networking, 2008.

[10] Zafeirios C.Papazachos, Helen D.Karatza, “Performance Evaluation of Gang Scheduling in a Two-Cluster System with Migrations”, IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp.1-8, 2009.

[11] Edi Shmueli, Dror G.Feitolson, “On simulation and design of Parallel-Systems Schedulers: Are we doing the Right Thing?”, IEEE Transactions on Parallel and Distributed Systems, vol.20, No.7, pp.983-996, July 2009.

[12] Adan Hirales, Jose-Luis Gonzalez-Garcia, Andrei Tchemykh, “Workload Generation for Trace based Grid Simulations”, First International SuperComputing Conference, ISUM, March, 2010.

[13] Grid’5000 team, Dr.Franck Cappello, http://gwa.ewi.tudelf.nl/pmwiki/, 2010.

[14] M.Balajee, B.Suresh, M.Suneetha, V.Vasudha Rani, G.Veerraju, "Preemptive Job Scheduling with Priorities and Starvation cum Congestion Avoidance in Clusters", IEEE International Conference on Machine Learning and Computing, pp.255-259, 2010.

[15] A.Neela madheswari, R.S.D.Wahida banu, "Static and dynamic job scheduling using communication aware policy in cluster computing", International Journal of Computers and Electrical Engineering, Vol 39, Issue 2, pp. 690-696, 2013.

Authors

A.Neela madheswari received her B.E., during 2000 and her M.E., during 2006 in Computer Science and Engineering. She is pursuing her Ph.D., in Computer Science and Engineering from Anna University, Tamil Nadu, India. Her research interests include Parallel computing and Web technologies.

R.S.D.Wahida Banu received her B.E., during 1981 and her M.E., during 1985. She completed her Ph.D., during 1998 and currently working as Principal, Government College of Engineering, Salem, Tamil Nadu, India. Her research interests include Distributed and Parallel Computing, Networks and Communications and Biomedical Engineering.