2012 International Conference on Information and Network Technology (ICINT 2012)
IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore

Node Circulation in Cluster Computing Using ORCS

S.M.Krishna Ganesh [+]

Department of Computer Science and Engineering, Raja College of Engineering and Technology, Madurai

Abstract. The system of cluster computing aims in distributing the process among the CPU to increase speed of processing. The server in the cluster performs the monitoring and load balancing of the clients. Here we aim to schedule the load of the client using a process scheduling node allocation based on the client capability, in a fashion that resembles a single node system. The algorithm can be implemented in any system that supports process migration. The objective is to improve the efficiency of load balancing and scalability in the cluster environment. The paper measures the performance of the algorithm on a MOSIX GRID with maximum of 60 nodes. The paper presented an on-line, resource sharing Algorithm for Node allocations in a cluster and its performance. First, we showed an algorithm for Excess allocations in which nodes, including unclaimed nodes, are proportionally allocated to users. Since in our model nodes are not shared by different users, the algorithm was extended to achieve a Temporal Node allocation by node circulations. In another extension, node allocations were based on a hierarchical tree. We implemented the algorithm on a running MOSIX Grid and measured its performance on a 60 node shared cluster. We tested cases with equal proportions, non-equal proportions, with node circulations, a hierarchical setup and in a flexible configuration.

Keywords: cluster computing, clustering, load balancing, monitoring, process scheduling.

1. Introduction

The cluster is a collection of interconnected stand-alone computers working together as a single, integrated computing resource consisting of many of the same or similar types of machine. A Cluster is a group of terminals or workstation attached to common control unit. We need to create a cluster with maximum number of nodes. It takes the advantage of non-local resources - using available computer resources on a wide area network. Overcoming memory constrain-single computers have very finite memory resources for large problems, using the memories of multiple computers may overcome this obstacle. The main feature in cluster environment with automatic resource discovery, preemptive process migration, and a priority scheme among migration in maximizing memory and/or disk performance among the cluster nodes we choose many workstations with peer architecture. Briefly, resource discovery is performed by an on-line information dissemination algorithm that maintains a distributed bulletin boarding order to provide each node with the latest information about the availability and the state of grid-wide resources.

2. Optimal Resource Constraint Scheduling(ORCS)

In computer science, a scheduling algorithm is the method by which threads or processes are given access to system resources, usually processor time. This is usually done to load balance a system effectively. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking, or execute more than one process at a time. Scheduling algorithms are generally only used in a time slice multiplexing. The reason is that in order to effectively load balance a system, the kernel must be able to suspend execution of threads forcibly in order to begin execution of the next thread. The proposed algorithm used may be as simple as Prior round-robin which is shown in figure 2. In which each process is given equal time (for instance 1 ms, usually between 1 ms and 100 ms) in a cycling list. So, process A executes for 1 ms, then process B, then process C, then back to process A. More advanced algorithms take

[+] Corresponding author. Tel.: + +918124465349.
E-mail address: krishnaganeshsm@gmail.com.

into account process priority, or the importance of the process. This allows some processes to use more time than other processes. It should be noted that the kernel always uses whatever resources it needs to ensure proper functioning of the system, and so can be said to have infinite priority. In SMP systems, processor affinity is considered to increase overall system performance, even if it may cause a process itself to run more slowly. This generally improves performance by reducing cache thrashing.

1. Initially Si = Max_100
2. Let ri be the no of cluster nodes
3. The Ci be the capability of ri
4. Pi be the no of process on ri
5. Si/Ci * 100 =Mi ↔{M1,M2,……Mn}
6. Max (Mi) is allotted with Pi
7. The Pi exec: = Σ Pi€ ri → M

Fig. 1: ORC Algorithm Steps

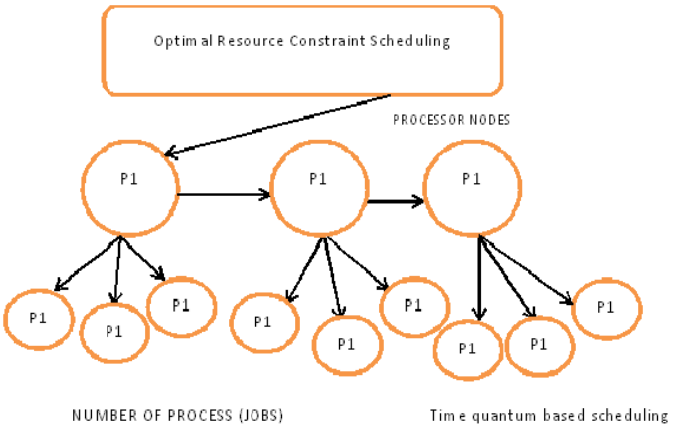

Fig. 2: ORC Scheduling basic frame work

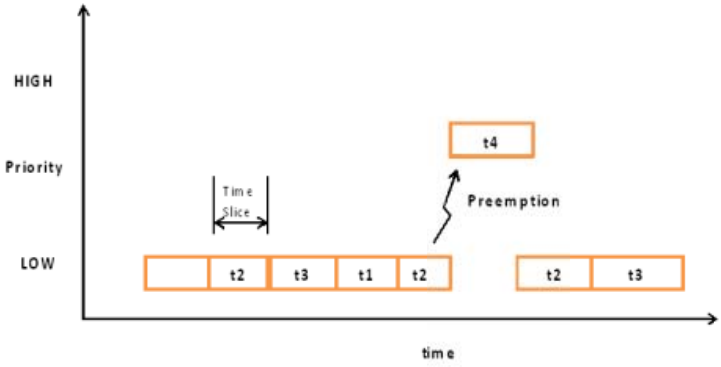

Fig. 3: Process Preemption with priority.

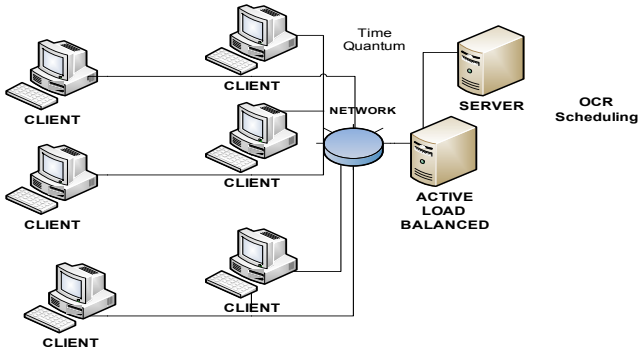

Fig. 4: The Load Balancing scheme in cluster

Let Si be the server RAM with maximum memory which is utilized by the ri number of cluster nodes effectively. The capacity of the nodes be Ci and is taken by dividing the RAM capacity and is utilized effectively. The maximum percentage is done with highest priority. Based on which the process are executed correspondingly to improve the scalability and load balancing. The simplest best-effort scheduling algorithms are round-robin, fair queuing (a max-min fair scheduling algorithm), proportionally fair scheduling and maximum throughput. If differentiated or guaranteed quality of service is offered, as opposed to best-effort communication, we use optimal resource constraint scheduling to improve the load balancing and scalability. Figure 3 below shows the OCR scheduling based on the time quantum compilation and pre-emption of process based on the priority. The main objective is to schedule the process dynamically among the clients by the server to improve the scalability and load sharing which could minimize the overhead. A scheduling algorithm which repeatedly runs through a list of users, giving each user the opportunity to use the central processing unit in succession. The existing scheduling may not be desirable if the size of the jobs or tasks are strongly varying. A process that produces large jobs would be favored over other processes. This problem may be solved by time sharing i.e. by giving each job a time slot or quantum (its allowance of CPU time), and interrupt the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process. Thus the ORC improves the efficiency. Example: The time slot could be 100 milliseconds. If a job1 takes a total time of 250ms to complete, the priority scheduler will suspend the job after 100ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (100ms each), job1 will get another allocation of CPU time and the cycle will repeat. This process continues until the job finishes and needs no more time on the CPU

2.1. Performance analysis of ORC scheduling

The process scheduling in cluster environment includes the round robin scheduling for the process to be distributed among the clients and the corresponding order is queued for execution. Thus the order of execution increases the performance of the load effectively shared across the network and the achieve the scalability. The tables below show the average waiting and turnaround time of the number of process in the total number of system executed correspondingly based on ORC algorithm. We consider a total of 5 systems and the maximum of 25jobs executed and minimal WT=10 and TT= 30 with high degree of accuracy. The ultimate aim is to improve the efficiency of the algorithm effectively and perform load balancing among the shared cluster nodes. The process in each processor is executed with high speed and the load of the server is distributed across the client effectively. The load balancing among the cluster nodes are distributed by the scheduling to reduce server load and improve the scalability of the systems effectively. This scheduling of the jobs to be run on the server is performed effectively. With improved efficiency and performance as shown in graph.The Following tables indicate the calculation of the average waiting and turnaround time.

Tables. 1: Calculating the Average waiting time and Turnaround time

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	3	6	14	20	22
AVG TT	21	32	43	54	65

SYSTEM 1: AVG WT= 13 AVG TT=23

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	25	35	40	45	55
AVG TT	21	42	54	66	83

SYSTEM 2: AVG WT= 40 AVG TT=53

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	24	44	65	85	107
AVG TT	32	64	87	106	141

SYSTEM 3: AVG WT=65 AVG TT=76

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	32	63	96	129	159
AVG TT	44	78	110	137	171

SYSTEM 4: AVG WT=96 AVG TT=102

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	43	77	121	155	204
AVG TT	56	103	130	167	204

SYSTEM 5: AVG WT= 112 AVG TT=132

NODES	SYS1	SYS2	SYS3	SYS4	SYS5
AVG WT	56	112	151	215	261
AVG TT	69	129	179	239	284

SYSTEM 6: AVG WT=159 AVG TT=178

2.2. Experimental study

The experimental study here shows the process speed of execution and the corresponding turnaround time with high degree of efficiency. The Figure 5 shows the graph showing the process and the time of processing which indicates the speed of execution correspondingly. Table1: shows the corresponding waiting time and turnaround time of the process executed. By implementing this algorithm number of process on the node are scheduled effectively. This scheduling helps to minimize the overload in the server and we increase the number of cluster nodes and their process being executed. The graph displays the increase in node and the corresponding process with increased performance. The graph displays comparison with FCFS, SJF and Round Robin Scheduling as follows

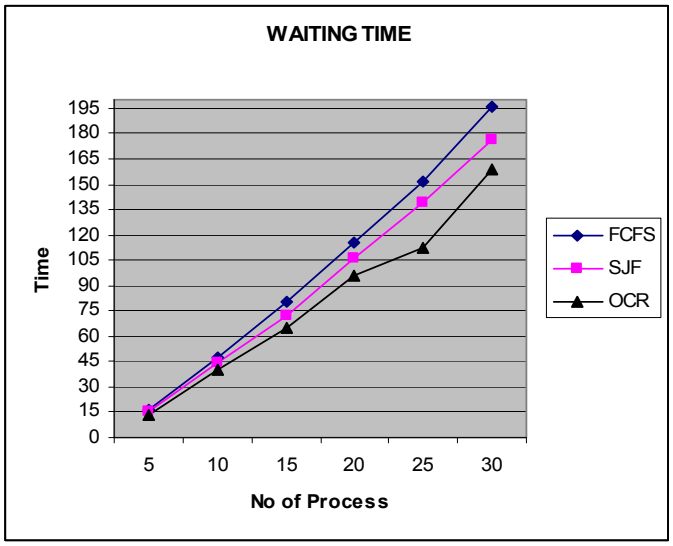

Fig. 5: Graph showing the Process Vs Turnaround time of processing

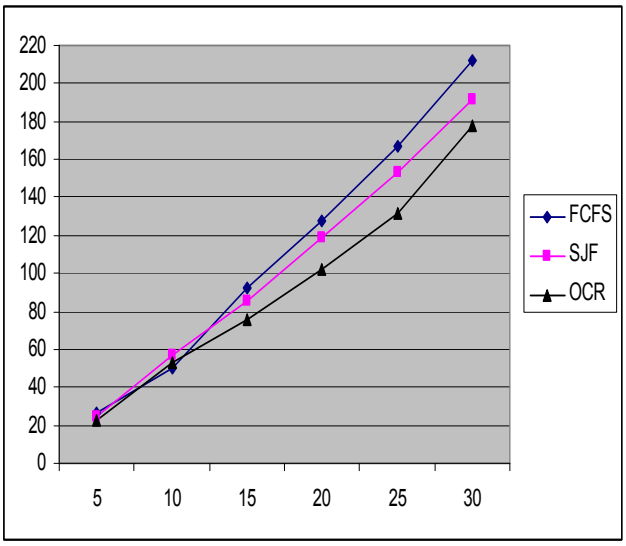

Fig. 6: Graph showing the Process Vs Waiting Time of processing

Table. 2: Time showing the Waiting time of the running process

PROCESS	FCFS	SJF	RR
5	27	25	23
10	50	57	53
15	92	85	76
20	128	119	102
25	167	153	132
30	212	192	170

Table. 3: Time showing turnaround running process

PROCESS	FCFS	SJF	RR
5	17	15	13
10	47	44	40
15	80	72	65
20	115	106	96
25	152	139	112
30	196	176	159

3. Conclusion

The paper presented an on-line, resource sharing Algorithm for Node allocations in a cluster and its performance. First, we showed an algorithm for Excess allocations in which nodes, including unclaimed nodes, are proportionally allocated to users. Since in our model nodes are not shared by different users, the algorithm was extended to achieve a Temporal Node allocation by node circulations. In another extension, node allocations were based on a hierarchical tree. We implemented the algorithm on a running MOSIX Grid and measured its performance on a 60 node shared cluster. We tested cases with equal proportions, non-equal proportions, with node circulations, a hierarchical setup and in a flexible configuration. The result shows the performance evaluation of the running process and CPU utilization the work presented in this paper can be extended in several ways. First, our algorithm can be generalized to handle different resources, such as nodes with several CPUs (SMPs), nodes with different CPU speeds or different amounts of memory. Another possible extension is to support time-sharing of nodes.

4. Acknowledgements

First of all i thank the almighty for giving me the knowledge and courage to complete the research work successfully. I like to thank the anonymous reviewers for their insightful remarks and valuable suggestions,

which were very helpful in improving the article. I express my gratitude to the respected PMJF.Lion.G.Nagarajan, Chairman, Raja College of Engineering & Technology, Madurai for allowing me to do the research work internally. Also i acknowledge the support provided by Dr.V.Sekaran M.E,Ph.D Principal, Raja College of Engineering & Technology,Madurai. I thank my parents, friends and collegues for their support and encouragement.

5. References

[1] A. C. Arpaci-Dusseau and D. E. Culler, “Extending Proportional-Share Scheduling to a Network of Workstations”,PDPTA, 1997.

[2] Barak, A. Shiloh and L. Amar, “An Organizational Grid of Federated MOSIX Clusters”, *CCGrid- 05,* Cardiff , 2005.

[3] J.F.C.M de Jongh, ”Share Scheduling in Distributed Systems”, *PhD Thesis*, Delft Tech. university, 2002.

[4] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham and R. Neugebauer, “Xen and the Art of Virtualization”, *OSDI*, 2003.

[5] S.D. Kleban and S.H. Clearwater, ”Fair Share on HighPerformance Computing Systems: What Does Fair Really Mean?” *CCGrid-03*, 2003.

[6] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang and B. A. Huberman, “Tycoon: an Implementation of a Distributed Market-Based Resource Allocation System”,*TR arXiv:cs.DC/0412038*, HP Labs, 2004.

[7] Jemal H. Abawajy and S. P. Dandamudi, “Distributed Hierarchical Co-ordination of Workstation Clusters,”IEEE PARELEC’2000, Trois-Rivierres, Quebec, Canada.

[8] S. P. Dandamudi and T.K. Thyagaraj, "A Hierarchical Processor Scheduling Policy for Distributed-Memory Multiprocessor Systems," In Proc. of HiPC'97, December 18-21, 1997, Bangalore, India.

[9] D. E. Bernholdt et. all. A component architecture for high-performance scientific computing. to appear in Intl. J. High-Performance Computing Applications.

[10] E. Gabriel et all. Open MPI: Goals, concept, and design of a next generation mpi implementation. In 11th European PVM/MPI Users’ Group Meeting, 2004.