Dynamic Load Balancing in Distributed System: An Efficient Approach

Md. Abdur Razzaque[1] and Choong Seon Hong[2]
Department of Computer Engineering, Kyung Hee University
1, Seocheon, Giheung, Yongin, Gyeonggi, Korea, 449-701
[1]m_a_razzaque@yahoo.com
[2]cshong@khu.ac.kr

Abstract

In this paper, the problem of distributing load of a particular node over ***m*** identical nodes of a distributed computing system for minimizing turnaround time is studied first. Then an efficient technique is presented for dynamically scheduling jobs in large-scale, multi-user distributed computing systems that provides a balanced system performance with respect to the scheduling overhead. The nodes are scheduled independently and asynchronously with distinct execution initiation times corresponding to their earliest instant of being overloaded. The technique handles the task of resource management by dividing the nodes of the system into mutually overlapping subsets and thereby a node gets the system state information by querying only a few nodes. The approach is primarily targeted at systems that are composed of general purpose workstation computers having identical processors. Process scheduling decisions are driven by the desire to minimize turnaround time while maintaining fairness among competing applications and minimizing communication overhead. The performance analysis of the technique shows that it significantly reduces the total number of messages required for a node to take scheduling decision.

1. Introduction

Scheduling is an important aspect for efficient resource utilization of a system. Distributed computing systems (DCS) are characterized by resource multiplicity and system transparency [1]. Every distributed system consists of a number of resources interconnected by a network. Besides providing communication facilities, it facilitates sharing computation power by migrating a local process and executing it at a remote node of the network. A process may need to be migrated because of the shortage of required resources at the local node or the local node has to be shut down for some reason. A process may also be executed remotely if the expected turnaround time needs to be better. From a user's point of view, the set of available resources in a distributed system acts like a single virtual system. Hence when a user submits a process for execution, it becomes the responsibility of the resource manager of the distributed operating system to control the assignment of resources to processes and to route processes to suitable nodes of the system according to these assignments. A resource can be logical, such as a shared file, or physical such is CPU. While the problem of scheduling parallel applications in distributed computing systems is already well explored, most existing approaches focus on dedicated, centralized or distributed environments. From a scheduling perspective, first approach addresses a single point of failure whereas the later one increases communication overhead to a great extent [2, 3].

The idea presented in this paper typically sits between these two approaches. The entire system is divided into number of subsets equal to the number of nodes. Each node in the system is assigned a subset. A node queries only the member nodes of its set to collect system state information and thus takes scheduling decision.

The amount of state information maintained at each node of the DCS and how it is used, is an important parameter for developing distributed load balancing algorithms. The state information about the total system can be used to reduce the message traffic in the network and to recover from failures. Message traffic should be minimized in order to decrease the overhead in the communications network. The performance of the proposed technique will be evaluated using the total number of messages required for a node to take scheduling decision as a criterion and turnaround time of application execution.

The major contributions of the paper include: (i) a novel workload migration technique proposed by considering the homogeneous and dynamic features of distributed computing environments (ii) dynamic and stable technique that satisfies the quick decision making capability and provides a balanced system performance with respect to scheduling overhead and (iii) the experimental results show that the proposed migration scheme outperforms common-used schemes with respect to reducing the communication cost and the application execution time.

The remainder of the paper is organized as follows. Section 2 features existing research works in distributed scheduling. Section 3 outlines the model on which our migration technique is based. The scheduling technique is presented in Section 4. Section 5 discusses about the performance of the proposed technique. Finally, Section 6 concludes the paper.

2. Related Works

We term real-time resource allocation and scheduling algorithms either static or dynamic, depending on the timing of allocation and scheduling are performed. In a static approach, information about the average behavior of the system is used, ignoring the current state of the system. The times at which the tasks start execution are determined a priori. Static algorithms [7, 8, 9, 10] are often used to schedule periodic tasks with hard deadlines. Whereas the attraction of the dynamic approaches [4, 11, 12, 14, 15, 17, 18, 21] is that they do execute schedules online. They respond to the systems current state and dynamically determine the feasibility of scheduling new tasks without jeopardizing the guarantees that have been provided for the

previously scheduled tasks, so are able to provide better performance. This paper only addresses the dynamic approach.

Many existing dynamic algorithms are often used to schedule aperiodic tasks and rarely used to schedule periodic tasks. Periodic tasks are mostly scheduled by static scheduling algorithms in the previous research. Dynamic scheduling algorithms may again be divided into two categories: centralized and distributed. In a centralized approach, one of the nodes functions as a central coordinator. The central coordinator, having all the information of the system, is fully responsible for scheduling. In a distributed approach, the decision making for scheduling is distributed across the entire system consisting of several nodes. Each node, having the information of the entire system, is responsible for its own decision [2, 3, 6].

Most existing dynamic homogeneous scheduling approaches target load balancing as the main motivation for dynamic reassignment of jobs and differ according to their accuracy and the amount of processor load information they exchange [3]. Zhou's algorithm [4] balances load by periodically requiring each processor to inform other processors of load changes. Willebeck-LeMair and Reeve [5] presented four scheduling policies that dynamically balance load without using global information, instead considering load only on neighboring processors.

Both the approaches centralized and distributed require frequent message exchanging between the nodes for collecting the system information in order to make scheduling decision. For a fully distributed system with N nodes for making a scheduling decision a node has to exchange approximately $2(N-1)$ messages [6] if we assume that information transfer policy is to exchange messages when state changes.

Cavanaugh et. al. in [11] proposed a resource management framework and an algorithm for dynamic resource management in distributed real-time system. Assuming that the multiple real-time and non-realtime processes are active throughout the system, the authors developed a set of software components for the framework to ensure efficient resource management of the system. This paper did not investigate the complexity of the proposed resource management algorithm neither in terms of time required nor the number of messages exchanged between nodes.

Zeng et. al. in [12] studied an optimal scheduling problem of divisible loads originating from single site in a computer system with arbitrary network configuration. They have presented a generic mathematical model and therefore proposed a distributed algorithm to solve the general divisible loads scheduling problem. But the message complexity of the algorithm is very high (N^2) and also fault-tolerant issues are not addressed here.

In [13] the problem of distributing m interacting tasks over N identical processors for minimizing job completion time is examined. The processors are independently and asynchronously scheduled corresponding to their earliest instant of availability. The incorporation of the effects of asynchronous execution starts for processors with different availability times extends the applicability of the results to the common situations in multiprocessor and distributed systems where scheduling policy dictates that, in the interest of improved processor utilization, an individual processor should start execution as soon as the processor becomes available. As a result, the distribution of load modules over the subset of engaged processors is generally uneven, with late available processors receiving less modules than processors with early availability.

C. Wang et. al. in [20] exploited the utilization of intelligent agents to cope with the dynamic load balancing. They proposed a distributed scheduling architecture oriented to grid service to enable load balancing and fault tolerance. In this model, the functionalities are implemented by applying intelligent agent technology in grid environments.

The technique presented here requires much less message exchanging for making a scheduling decision. Our proposed technique requires only $2(K-1)$ (where $K=\sqrt{N}$ approx.) messages to decide whether to execute a process locally or remotely at a different node.

3. The System Model

The effectiveness of a new technique depends on the suitability of the model as well as the validity of the assumptions made about the computing environment. The technique presented here is based on the following assumptions and conditions for the distributed environment:

- All nodes in the system are assigned unique identification numbers from 1 to N.
- All the nodes in the system are fully connected.
- The method used as the Load Estimation policy would be the measure of CPU utilization of the nodes [8]. CPU utilization is defined as the number of CPU cycles actually executed per unit time.
- Process transfer policy that determines whether to execute a process locally or remotely is implemented by the double threshold policy by Alonso and Cova [8].
- The node sends its state information to other nodes only when its state switches from normal load region to either overload or under load region [8].

When reviewing an algorithm, attention should be paid to the assumptions made about the communications network. This is very important because nodes communicate only by exchanging messages with each other. The following aspects about the reliability of the underlying communications network should be considered [8].

- Message delivery is guaranteed.
- Message-order is preserved.
- Message transfer delays are finite.
- The topology of the network is known.

The idea presented in this paper is basically based on the concept of mutually overlapping subsets presented by

Maekawa [7]. For a network with *N*-number of nodes, we create *N* different subsets of size *K* (*K*=√*N* approx.) such that each subset overlaps every other subset. Each of the subsets is assigned to different numbers.
That is, for each number i, we define a subset S_i such that

$$S_i \cap S_j \neq \Phi$$

For any combination of i and j, 1 <= i , j <= N
Here is an example of mutually overlapping subsets for the integers 1 to 13:

N={1,2,3,4,5,6,7,8,9,10,11,12,13}
|N|=13
|N|=K(K-1)+1 So K=4
First 4 numbers are = {1,2,3,4}
The remaining numbers {5,6,7,8,9,10, 11,12,13}
The matrix along with the single number

	2	2	2
1	5	6	7
1	8	9	10
1	11	12	13

Jump-1 Diagonal	Jump-2 Diagonal
5, 9, 13	5,10,12
6,10,11	6, 8, 13
7, 8,12	7, 9, 11

The subsets are:

S_1={1,2,3,4}	S_2={2,5,8,11}
S_3={3,7,8,12}	S_4={4,7,9,11}
S_5={1,5,6,7}	S_6={2,6,9,12}
S_7={2,7,10,13}	S_8={1,8,9,10}
S_9={3,5,9,13}	S_{10}={3,6,10,11}
S_{11}={1,11,12,13}	S_{12}={4,5,10,12}
S_{13}={4,6,8,13}	

Fig. 1: Formation of mutually overlapping subsets using Maekawa's algorithm.

4. Proposed Technique

The basic technique is presented first as below and then its enhancement is described in the following subsections.

4.1 Basic Technique

The framework model stated in the preceding section forms the basis of our scheduling technique. We divide an *N* node system into *N* different mutually overlapping subsets. Each subset is assigned to a different node. The assigned set is considered as the request set of the corresponding node. Before making a scheduling decision a node query only the member nodes of its request set.
It is understandable from the discussion of the mutually overlapping subsets that if a node does query its request set members then that node will be able to get most of the nodes state information. The member nodes of the inquirer request set have the last known state information of the member nodes of their respective request sets. But these nodes typically are not included in inquirer's request set. So through its request set's member nodes the original inquirer gets overall system state information. Let's consider an example:
Considering a system with 13 nodes, from the figure 1 we see that the request sets of the nodes 1, 2, 3 & 4 are as follows:

S_1 = {1, 2, 3, 4} S_2 = {2, 5, 8, 11}
S_3 = {3, 7, 8, 12} S_4 = {4, 7, 9, 11}

So the node 1 will exchange its state change messages only with node 2, 3 and 4. From these nodes it can also acquire the state information of nodes numbered 5, 7, 8, 9, 11 and 12 and can update its system state table. So, whenever node 1 is overloaded it can migrate its local processes to execute remotely to one of the nodes of 2, 3, 4, 5, 7, 8, 9, 11 & 12 whose last known state is under loaded [6, 8]. So node 1 does not need to communicate explicitly with all the nodes for making a migrating decision. Based on the collected state information it can quickly make its scheduling decision.

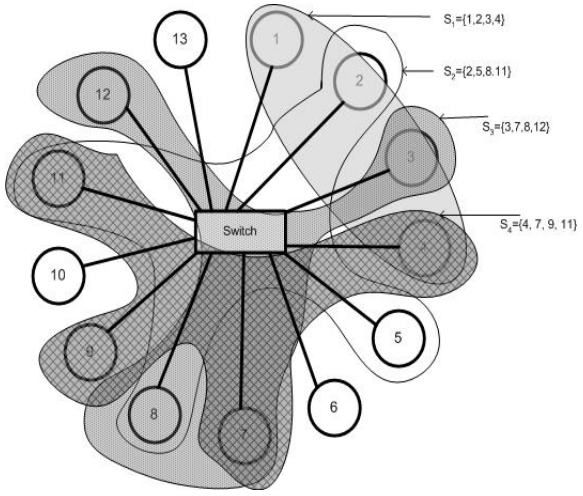

Fig. 2: Network connectivity (Only first four mutually overlapping subsets are shown)

A typical network with first four mutually overlapping subsets is shown in Fig. 2. These nodes may differ in configuration from each other such as, speed characteristics and number of resources etc. However, we assume that they have the same processing capabilities. It is also assumed that each node can receive loads from any other node. Because of the use of multi-port communication model, it can handle several communication sessions and receive loads from two or more neighboring nodes simultaneously. The current load of a node would be calculated from the CPU utilization. Every node has an upper threshold of CPU utilization, beyond which the node would be regarded as overloaded and similarly whenever the CPU utilization degrades to lower threshold value, the node is then under loaded. Thus, a node can decide whether to migrate some of its loads to

other nodes or not, and or how much load can it receive from other nodes. For load migration, each node intelligently selects its processes from available pool of processes so that minimum processing and communication overhead is occurred.
The technique reacts towards the addition or removal of a node from the system in the following manner:
When a new node is added to the network, it must-

- Interrogate some active nodes to get a list of participating nodes.
- Assign itself a unique node number (which is one greater than the current total node number).
- Initiate all nodes to recalculate their request set for the modified system by the addition of this node.

When a node leaves the network, it must notify all other nodes of its intention. In the meantime, it cannot participate in any communication.

4.2 Enhancement

The limitation of this basic technique is that while taking scheduling decision some nodes information is missing, as they do not include in any of the sets. For example, in the case of 13 node system each node misses 3 nodes information while making a decision. But as the number of nodes increases in a system the information of the missing nodes becomes insignificant. For instance, a system with 7 nodes misses only 1 node's information; systems with 31 nodes miss 7 nodes information and so on. However, from the definition of the distributed scheduling technique it is desirable that each node should have only a partial view of the entire system, which minimizes our limitation. One way to completely eliminate this problem is to communicate explicitly with the missing nodes. For example, a system with 31 nodes will miss 7 nodes information. If we explicitly communicate with these nodes then the required messages will be 2(6-l) + 2*7 = 24 [According to proposed technique for 31 node system we require 2*(6-1) messages for getting system wide information as (√31) = 6 (approx.) and 2*7 messages for communicating with missing nodes explicitly] where as in traditional systems the required messages is very high, 2(31-1) = 60. As the number of node increases, the amount of explicit message exchanging requirement also decreases.

The task migration technique presented in this paper is partially distributed, not every node participates in making scheduling decision. Even though the process of deciding the node at which a process would be executed is not fully distributed over all nodes but the task of decision making is distributed for each node in turn. Finally, the technique is fair with respect to load balancing and scheduling of process migration.

5. Performance Comparison

The proposed technique is tested in a distributed computing system having 150 nodes, numbered from 1 to 150 and was configured as specified in section 3. The codes written for implementing the proposed technique were installed in all the nodes first. Then some experiments are carried out to compare the performance of the proposed technique with the traditional one. The experimental values taken for producing graphs are mean of repeated experiments.
The performance gain of our technique is significant from overhead perspective. For a traditional *N* node distributed system each node need to exchange *2(N- 1)* messages for updating system management table thus making a scheduling decision. The attractive potentiality of this newly proposed technique is that we need to exchange only *2(K-1)* messages.
Fig. 3 shows that the experimental result matches very well with the theoretical estimations and this reduces network traffic to a great extent. It also makes quick scheduling decision about the assignment of processes. Fig. 4 shows that scheduling decision time required by traditional algorithm is very high and almost incomparable with the proposed technique for higher number of nodes. This happens for the traditional algorithm as because it does need to communicate each and every process explicitly for collecting present system load information and therefore message communication time adds up high which in turn makes the scheduling decision delayed. On the other hand, the number of nodes in a subset of the proposed method increases slowly (K=√N) as the number of total nodes in the system increases and thereby it can make scheduling decision quickly.
The proposed technique provides greater reliability than centralized approach or distributed approach as node crash does not result in the down of the entire system. Rather the node misses only partial information for making scheduling decision. For example in the 13-node system, if node 2 crashes then only node 5's information is missing. Based on the state information gathered by node 1 from the nodes of its request set, it can take scheduling decision quickly enough whether to run a process locally or remotely. The technique provides the scheduling decision influencing information quite fairly. It has a high scalability factor because the inquirer receives fairly small number of messages considering the total node number for making a decision.

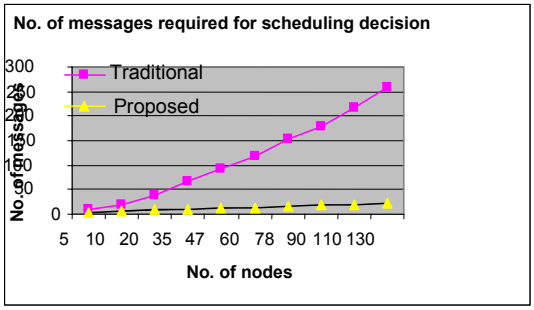

Fig. 3: The no. of messages required for making scheduling decision

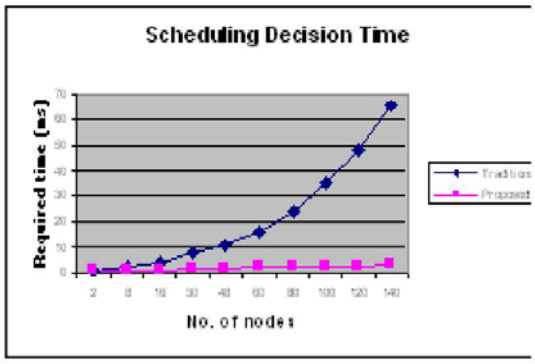

Fig. 4: Time required for taking scheduling decisions

6. Conclusions

The motivation towards the development of this paper was to develop a dynamic distributed scheduling technique that reduces the communication overhead involved in decision making. It is shown that from the number of message exchanging perspective our proposed technique provides a significant performance. Noticeable here is that this modified technique will provide far better performance than the traditional one. As the performance of a distributed system is heavily influenced by the scheduling mechanism, the technique presented in this paper may lead to a balanced system performance as scheduling decision is performed quite fairly with less communication overhead.

References

[1] J. Xl and K. Hwang, Dynamic load balancing for parallel program execution on a message passing multi computer., Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, pp.402—406, 1990.

[2] Andrew S. Tanenbaum, "Distributed Operating Systems", Prentice Hall, 1995

[3] Maekawa, M.; Oldehoeft, A.E.; and Oldeholit, R.R., "Operating Systems, Advanced Concepts," Benjamin-Cummings. 1987.

[4] S. Zhou, A trace driven simulation study of dynamic load balancing. IEEE Trans. Software Engineering14 (9): 1327-1341, Sep. 1988.

[5] J. M. Smith, A survey of process migration mechanisms. ACM Operating Systems Review 22(3), 28—40, Jul. 1988.

[6] Silberschatz, A. and Peterson, J.L., Operating System Concepts, Addison-Wesley, Alternate edition, 1988.

[7] H. S. Stone, Multiprocessor Scheduling with Aid of Network Flow Algorithms, *IEEE Trans. Software Eng.* vol. 3, no. 1, 1977, pp. 85-93

[8] V. M. Lo, Heuristic Algorithms for Task Assignments in Distributed Systems, *IEEE-Trans. on Computers,* vol. 37, no. 11, November 1988.

[9] S. Bokhari, Partitioning Problems in Parallel, Pipelined, and Distributed Computing, *IEEE Trans. Computers,* vol. 37, no. 1, January 1988, pp. 48-57.

[10] B. Indurkhya, H. Stone, L. Xi-Chen, "Optimal Partitioning of Randomly Generated Distributed Programs," *IEEE Trans. Software Engg,* vol. SE-1, March 1986, pp. 483-495.

[11] C. Cavanaugh, Radhika A., Dynamic Resource Management Algorithm for a Distributed Real-time System, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) 2005.

[12] Z. Zeng and B. Veeravalli, Divisible Load Scheduling on Arbitrary Distributed Networks via Virtual Routing Approach, Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS'04) 2004.

[13] C. Banino, 0. Beaumont, L. Carter, Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms, IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 4, April 2004

[14] S. Genaud, A. Giersch, F. Vivien, Load-Balancing Scatter Operations for Grid Computing, International Parallel and Distributed Processing Symposium (IPDPS'03) April 22 - 26, 2003 Nice, France

[15] Z.Lan, V.Taylor, and G.Bryan, Dynamic load balancing of SAMR applications on distributed systems, Proc. of SC2001, Denver, CO, 2001.

[16] Z.Lan, V.Taylor, and G.Bryan, Exploring cosmology applications on distributed environments, Journal of Future Generation Computer Systems, pages 839--847, 2003.

[17] Y.F. Hu, R.J. Blake, and D.R. Emerson, An optimal migration algorithm for dynamic load balancing, Concurrency: Practice and Experience, vol. 10, issue 6, (1998), pp. 467-- 483.

[18] Yawei Li and Zhiling Lan, A Novel Workload Migration Scheme for Heterogeneous Distributed Computing, IEEE International Symposium on Cluster Computing and the Grid, pp. 1055-1062, 2005

[19] J. Cab, D. P. Spooner et al., Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling, Proc. of Parallel and Distributed Processing Symposium. 2003.

[20] C. Wang, D. Chen, H. Hu, Intelligent Agent-Based Scheduling Mechanism for Grid Service, Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004, pp. 16-21

[21] Chun-Hsien Liu, Chia-Feng Li, Kuan-Chou Lai, Chao-Chin Wu, A Dynamic Critical Path Duplication Task Scheduling Algorithm for Distributed Heterogeneous Computing Systems, Proceedings of the 12th IEEE International Conference on Parallel and Distributed Systems (ICPADS'06) 2006