# An Efficient Dynamic Scheduling Algorithm In Distributed System

Fahim Kawsar, Md. Shahriar Saikat, Dr. M. A. Mottalib.
Department of Computer Science and Information Technology
Islamic University of Technology

## ABSTARCT

This paper presents an algorithm for scheduling applications in large-scale, multi-user distributed systems. The approach is primarily targeted at systems that are composed of general-purpose workstation networks. Scheduling decisions are driven by the desire to minimize turnaround time while maintaining fairness among competing applications and minimizing communication overhead. The algorithm handles the task of resource management by dividing the nodes of the system into mutually overlapping subsets. Thus a node gets the system state information by querying only a few nodes subsequently can take scheduling decision.

*Keywords: Scheduling, distributed systems, mutually overlapping subsets, process migration.*

## 1. INTRODUCTION

Over the last decade distributed computing systems have attracted a great deal of attention. This is due, in part, to the technological advances in the design of sophisticated software and communication interfaces, the availability of low-cost processors and the rapid decline in hardware costs. The motivations for building distributed computing systems are many. Resource sharing, parallel processing, system availability and communication are four major reasons. By distributing a computation among various sites, processes are allowed to run concurrently and to share resources, but still work independently of each other.

Distributed systems 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 resource sharing by migrating a local process and executing it at a remote node of the network. A process may be migrated because the local node does not have the required resources or the local node has to be shut down. 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 resource 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 as CPU. While the problem of scheduling parallel applications on 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].

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 and each subset is assigned to every node of the system. A node queries only the nodes of its set to collect most of the nodes state information and thus take scheduling decision.

The motivation towards the development of this paper was to derive a semi distributed scheduling approach that exploits the dynamicity and stability requirements of a good scheduling technique. In addition our proposed method also satisfies the quick decision making capability and provides a balanced system performance with respect to scheduling overhead. Another aspect to consider in an algorithm is the amount of information that is maintained by each node in the system and how it is used. The state information about the total system can be used to reduce the message traffic in the network and to recover from failures. The performance of the algorithm presented here will be evaluated using the total number of messages required for a node to take scheduling decision as a criterion. Message traffic should be minimized in order to decrease the overhead in the communications network.

The remainder of the paper is organized as follows. Section 2 compares our work with existing research in distributed scheduling. Section 3 outlines the model on which our scheduling algorithm is based. The scheduling algorithm is presented in Section 4. Finally, Section 5 concludes the paper.

## 2. RELATED WORKS

Two approaches have been used for managing resources in a distributed computing system. In a static approach, information about the average behavior if the system is used, ignoring the current state of the system where as the attraction of the dynamic approaches is that they do respond to the systems state and so are able to provide better performance. This paper only addresses the dynamic approach. Dynamic Scheduling algorithm may be centralized or distributed,

In a *centralized* approach, one of the nodes functions as a central coordinator. The central coordinator is fully responsible for scheduling having all the information of the system. 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.

A *centralized* algorithm is characterized by these properties.

- ✓ Only the central node makes a decision
- ✓ Sometimes, central control may Migrate from node to node.
- ✓ All necessary information is held at the central node.

The main problem with this approach is the reliability issue that is a single point of failure

A *fully distributed* algorithm has these properties.

- ✓ All nodes have an equal amount of information.
- ✓ All nodes make a decision based solely on local information.
- ✓ All nodes bear equal responsibility for a final decision.
- ✓ All nodes expend equal effort in effecting a final decision.
- ✓ Usually, failure of a node does not cause system collapse.

Many algorithms are distributed, but few satisfy all of these properties.

Most existing dynamic homogeneous scheduling approaches target load balancing as the main motivation for dynamic reassignment 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. Willebeek-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 or distributed require frequent message exchanging between the nodes for collecting the system information for making 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 if we assume that information transfer policy is exchange messages when state changes [6] messages. The algorithm presented here requires far less message exchanging before making a scheduling decision. Our proposed algorithm requires only 2(k-1) messages to decide whether to execute a process locally or remotely where K=√N (approx).

## 3. FRAMEWORK MODEL

The idea presented in this paper is basically based on the concept of mutually overlapping subsets [Maekawa 1985] [7]. For N–number, 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

$$S_i \cap S_j \neq \emptyset$$

For any combination of i and j, $1 \leq i, j \leq N$.]

Here is an example of mutually overlapping subsets of the integers 1 to 13:

**N=13**
**N= {1,2,3,4,5,6,7,8,9,10,11,12,13 }**
**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}**
**$S_4$ = {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: Maekawa's Algorithm

## 4. PROPOSED ALGORITHM

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

**i.** All nodes in the system are assigned unique identification numbers from 1 to N.
**ii.** All the nodes in the system are fully connected.
**iii.** The method used as the Load Estimation policy would be the measure of CPU utilization of the nodes [8]. Central Processing

Unit (CPU) utilization is defined as the number of CPU cycles actually executed per unit time.

**iv.** Process transfer policy which determines whether to execute a process locally or remotely is implemented by the double threshold policy [Alonso and Cova 1988] [8]

**v.** The node send its state information to other nodes only when it's state switches from normal load region to either over load or under load region. [8]

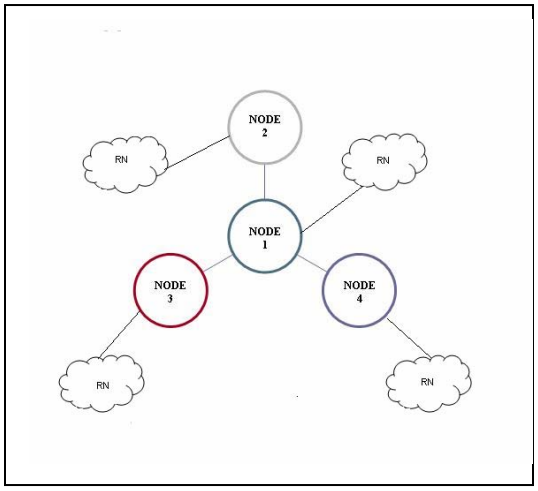
When reviewing an algorithm, attention should be paid to the assumptions made about the communications network. This is very important because nodes communicate only by exchanging messages with each other. The following aspects about the reliability of the underlying communications network should be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

- Message delivery guaranteed.
- Message-order preservation.
- Message transfer delays are finite, but unpredictable.
- The topology of the network is known.

The framework model stated in the proceeding 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 different a node that is considered as its request set. 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 query its request set members then that node will be able to get most of the nodes state information Because the nodes of its request have the last known state information of their request set nodes which are not included in inquirer request set. Let's see an example.

Considering a system with 13 nodes, from the figure 1 we see that the request sets of the node 1,2,3 & 4 are

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




RN: Rest of the Network (Transparent)
Fig 2: Nodes Connectivity

So the node 1 will exchange its state change messages only with node 2, 3 and 4 and from this 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 and based on the collected state information it can quickly makes its scheduling decision. The algorithm 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 node to get a list of participating nodes.
• Assign itself a unique node number.
• Have its node number placed on all other nodes' list of participating

When a node leaves the network, it must notify all other nodes of its intention. In the

meantime, it cannot participate in any communication.

### 4.1 PERFORMANCE GAIN

The performance gain of our algorithm 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 where K=√N (approx). This reduces network traffic to a great extent. It also makes quick scheduling decision about the assignment of processes. It also provides greater reliability then 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 example given, 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 it's request set it can take scheduling decision quickly enough that is whether to run a process locally or remotely. The algorithm provides the scheduling decision influencing information quite fairly. The algorithm has a high scalability factor because the inquirer receives fairly small number of messages considering the total node number for making a decision.

## 5. CONCLUSION

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 visible that from number of message exchanging perspective our proposed algorithm provides a significant performance. The limitation of this algorithm is that 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 a while making a decision. But as the number of nodes increases in a system the information of the missing nodes is insignificant, for example for a system with 7 nodes 1 nodes information is missing, a system with 31 nodes 7 nodes information is missing. 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 may be 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(6-1)+2*7=24 where as in traditional systems the required message is 2(31-1) = 60.As the node number increases the amount of message requirement also decreases. It is noticeable here is that this version still provides far better performance than the previous one. As the performance of a distributed system is heavily influenced by the scheduling mechanism, the algorithm presented in this paper may lead to a balanced system performance as scheduling decision is performed quite fairly with less communication overhead.

## 6. REFERENCES


[1] J. XU 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 OLDEHOEFT, R.R., "Operating Systems, Advanced Concepts," Benjamin-Cummings, 1987.

[4] S. ZHOU. A trace driven simulation study of dynamic load balancing. IEEE Trans. Software Engineering 14(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] MAEKAWA, M., "A sqrt(n) algorithm for mutual exclusion in decentralized systems," ACM Transactions on Computer Systems, vol 3, no. 2, May 1985, pp. 145-159
[8] PRADEEP K SINHA", Distribute Operating Systems, Concepts and Design", IEEE Press, 1998