Table of Contents

INTRODUCTION

A distributed scheduler is a resource management component of a distributed operating system that focuses on judiciously and transparently redistributing the load of the system among the individual units to enhance overall performance.

Users submit tasks at their host computers for processing.

WHY LOAD DISTRIBUTION ALGORITHM NEEDED ?

The need for load distribution arises in such environments because, due to the random arrival of tasks and their random CPU service time requirements, there is a good possibility that several computers are idle or lightly loaded and some others are heavily loaded, Which would degrade the performance.

In real life applications there is always a possibility that one server or system is idle while a task is being waited upon at another server.

COMPONENTS OF A LOAD DISTRIBUTION ALGORITHM

1. Transfer Policy

Transfer policy indicates when a node (system) is in a suitable state to participate in a task transfer.

The most popular proposed concept for transfer policy is based on a optimum threshold.

Thresholds are nothing but units of load. When a load or task originates in a particular node and the number of load goes beyond the threshold T, the node becomes a sender (i.e. the node is overloaded and has additional task(s) that should be transferred to another node).

Similarly, when the loads at a particular node falls bellow T it becomes a receiver.

2. Selection Policy

A selection policy determines which task in the node (selected by the transfer policy), should be transferred.

If the selection policy fails to find a suitable task in the node, then the transfer procedure is stopped until the transfer policy indicates that the site is again a sender.

3. Location Policy

This policy selects the node where the selected task is to be transferred. Simplest approach to find such node is Polling. In polling, one node calls another node to determine whether it is in a state of load sharing. Nodes can be called serially or parallaly. A node can be selected for polling randomly or on nearest neighbor basis. An alternative to polling is to distribute a query to all nodes to find the available nodes.

4. Information Policy

This policy is responsible for determining when the system state is to be corrected, from where it is to be corrected and what information is to be corrected.

It is of three types:

4.1 Demand Driven

In this policy, the node starts correcting the state of other nodes when it becomes a sender or receiver. This policy is basically a dynamic policy. This policy can be sender initiated, receiver initiated or symmetrically initiated. In sender initiated, a sender looks for receiver to give their load, In receiver initiated, receiver searches for sender, take their load, and symmetrically initiated is the combination of both where the collection of the system state is started whenever there is need of extra processing power.

4.2 Periodic

In this policy, the nodes exchange their system information periodically and on the basis of this information, task transfer is made. This policy does not adapt its activities according to system state change for example if the system is already overloaded then exchanging the system state information periodically will further worsen the situation.

4.3 State Change Driven Policy

In this policy, a node disseminates its state information to other nodes whenever its state changes by certain degree. This policy differs from demand driven policy as in this case, the nodes disseminate their state information rather than collecting the state information of other nodes. Under centralized approach, a node disseminates its information t centralized collection point whereas in case of decentralized approach, a node disseminate to peers.

TYPES OF LOAD DISTRIBUTION ALGORITHM

1. Classification According to Approach

Load distribution algorithms can be classified as

1. Static
2. Dynamic or adaptive.

1. Static

Static schemes are those when the algorithm uses some priori information of the system based on which the load is distributed from one server to another.

The disadvantage of this approach is that it cannot exploit the short term fluctuations in the system state to improve performance. This is because static algorithms do not collect the state information of the system.

These algorithms are essentially graph theory driven or based on some mathematical programming aimed to find a optimal schedule, which has a minimum cost function. But unfortunately Gursky has shown that the problem of finding an optimal schedule for four or more processing elements is NPhard.

2. Dynamic

Dynamic scheduling collect system state information and make scheduling decisions on these state information.

An extra overhead of collecting and storing system state information is needed but they yield better performance than static ones.

2. Load Sharing and Load Balancing

Although both type of algorithms strive to reduce the likelihood of unshared state i.e. wait and idle state, load balancing goes a step further by attempting to equalize loads at all computers.

Because a load balancing algorithm involves more task transfers than load sharing algorithms, the higher overhead incurred by load balancing types may outweigh its potential improvement.

3. Initiation Based

In general the algorithms are also categorized on which node initiates the load distribution activity. The variations are sender initiated, receiver initiated or symmetrically initiated (by both sender and receiver). A sender initiated algorithm was studied by Eager et. al. and a receiver initiated algorithm was studied where as a symmetrically initiated algorithm was adopted Moreover an adaptive stable symmetrically initiated algorithm was put forward in and a stable sender initiated algorithm was discussed All the load distribution algorithms are based on one of more of the types discussed above