Resource Management

- A resource can be a logical, such as a shared file, or physical, such as a CPU (a node of the distributed system). One of the functions of a distributed operating system is to assign processes to the nodes (resources) of the distributed system such that the resource usage, response time, network congestion, and scheduling overhead are optimized.
- There are three techniques for scheduling processes of a distributed system:

  1) ***Task Assignment Approach,*** in which each process submitted by a user for processing is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance.

  2) ***Load-balancing approach,*** in which all the processes submitted by the users are distributed among the nodes of the system so as to equalize the workload among the nodes.

  3) ***Load-sharing approach,*** which simply attempts to conserve the ability of the system to perform work by assuring that no node is idle while processes wait for being processed.

  The task assignment approach has limited applicability to practical situations because it works on the assumption that the characteristics (e.g. execution time, IPC costs etc) of all the processes to be scheduled are known in advance.

Desirable features of a good global scheduling algorithm

- ## No a priori knowledge about the processes

  Scheduling algorithms that operate based on the information about the characteristics and resource requirements of the processes pose an extra burden on the users who must provide this information while submitting their processes for execution.

- ## Dynamic in nature

  Process assignment decisions should be dynamic, I.e., be based on the current load of the system and not on some static policy. It is recommended that the scheduling algorithm possess the flexibility to migrate a process more than once because the initial decision of placing a process on a particular node may have to be changed after some time to adapt to the new system load.

- ## Quick decision making capability

  Heuristic methods requiring less computational efforts (and hence less time) while providing near-optimal results are preferable to exhaustive (optimal) solution methods.

- ## Balanced system performance and scheduling overhead

  Algorithms that provide near-optimal system performance with a minimum of global state information (such as CPU load) gathering overhead are desirable. This is because the overhead increases as the amount of global state information collected increases. This is because the usefulness of that information is decreased due to both the aging of the information being gathered and the low scheduling frequency as a result of the cost of gathering and processing the extra information.

- ## Stability

  Fruitless migration of processes, known as processor thrashing, must be prevented. E.g. if nodes n1 and n2 observe that node n3 is idle and then offload a portion of their work to n3 without being aware of the offloading decision made by the other node. Now if n3 becomes overloaded due to this it may again start transferring its processes to other nodes. This is caused by scheduling decisions being made at each node independently of decisions made by other nodes.

Desirable features of a good global scheduling algorithm

Scalability

A scheduling algorithm should scale well as the number of nodes increases. An algorithm that makes scheduling decisions by first inquiring the workload from all the nodes and then selecting the most lightly loaded node has poor scalability. This will work fine only when there are few nodes in the system. This is because the inquirer receives a flood of replies almost simultaneously, and the time required to process the reply messages for making a node selection is too long as the number of nodes (N) increase. Also the network traffic quickly consumes network bandwidth. A simple approach is to probe only m of N nodes for selecting a node.

Fault tolerance

A good scheduling algorithm should not be disabled by the crash of one or more nodes of the system. Also, if the nodes are partitioned into two or more groups due to link failures, the algorithm should be capable of functioning properly for the nodes within a group. Algorithms that have decentralized decision making capability and consider only available nodes in their decision making have better fault tolerance capability.

Fairness of service

Global scheduling policies that blindly attempt to balance the load on all the nodes of the system are not good from the point of view of fairness of service. This is because in any load-balancing scheme, heavily loaded nodes will obtain all the benefits while lightly loaded nodes will suffer poorer response time than in a stand-alone configuration. A fair strategy that improves response time of the former without unduly affecting the latter is desirable. Hence load-balancing has to be replaced by the concept of load sharing, that is, a node will share some of its resources as long as its users are not significantly affected.