Table of Contents

Introduction to distributed scheduling

Scheduling refers to the execution of non-interactive processes or tasks at designated times and places around a network of computer.

Distributed scheduling refers to the chaining of different jobs into a coordinated workflow that spans several computers. For example: – you schedule a processing job on computer1 and computer2, and when these are finished you need to schedule a job on computer3, this is distributed scheduling.

Motivation

Locally distributed system consists of a collection of autonomous computers, connected by a LAN.
In a locally distributed system, there is a good possibility that several computers are heavily loaded while others are ideal or lightly loaded.

If we can move jobs around, the overall performance of the system can be maximized.

A distributed scheduler is a resources management component of a distributed operating system that focuses on judiciously and transparently redistributing the load of the system among the computers to maximize the overall performance.

- A distributed system may have a mix of heavily and lightly loaded system.
- Tasks arrive at random intervals.
- CPUs service time requirements are also random.
- Migrating a task to share or balance load can help.

System may be heterogeneous in terms of CPU speed and resources. The distributed system may be heterogeneous in terms of load in different system.

Even in homogeneous distributed system a system may be idle even when a task is waiting for service in other system.

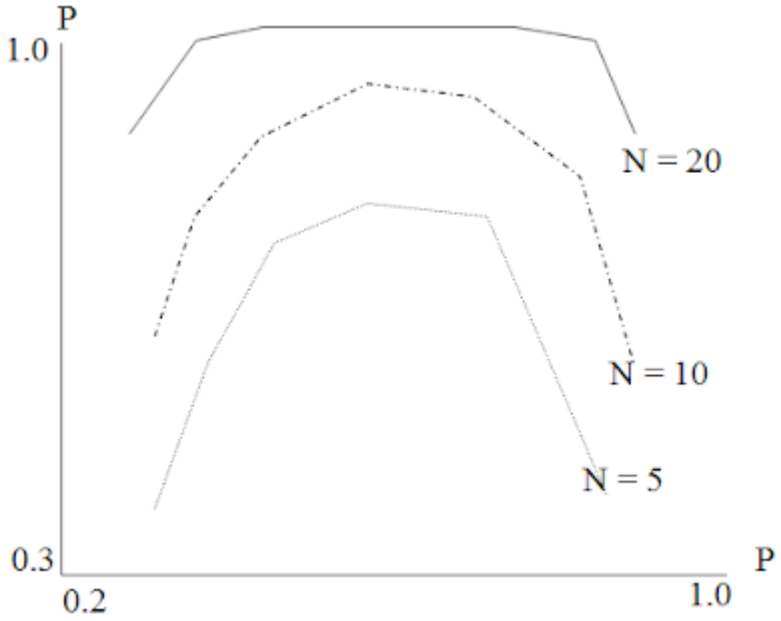
Consider a system of N identical, independent M/M/1 servers. Let P be the probability that the system is in state in which at least 1 task is waiting for service and at least 1 server is idle.

Let P be the utilization of each servers. We can estimate P using probabilistic analysis and plot a graph against system utilization.

For moderate system utilization, Value of P is high, i.e. at least 1 node is idle. Hence, performance can be improved by sharing of task.

What is performance?

Average response time.



Types of Algorithms

1. Static load distribution algorithms

Decisions are hard-coded into an algorithm with a priori knowledge of system.

2. Dynamic load distribution

Use system state information such as task queue length, processor utilization.

3. Adaptive load distribution

Adapt the approach based on system state.(e.g.) Dynamic distribution algorithms collect load information from nodes. Even at very high system loads.

Load information collection itself can add load on the system as messages need to be exchanged.

Adaptive distribution algorithms may stop collecting state information at high loads.

Balancing v/s Sharing

1. Load balancing

- Equalize load on the participating nodes.
- Transfer tasks even if a node is not heavily loaded so that queue length on all
- Nodes are approximately equal.
- More number of tasks transfers, might degrade performance.

2. Load sharing

- Reduce burden of an overloaded node.
- Transfer tasks only when the queue length exceeds a certain threshold.
- Less number of task transfers.
- Anticipatory task transfer: Transfer from overloaded nodes to ones that are likely to become idle/highly loaded
- More like load balancing, but may be less number of transfers.