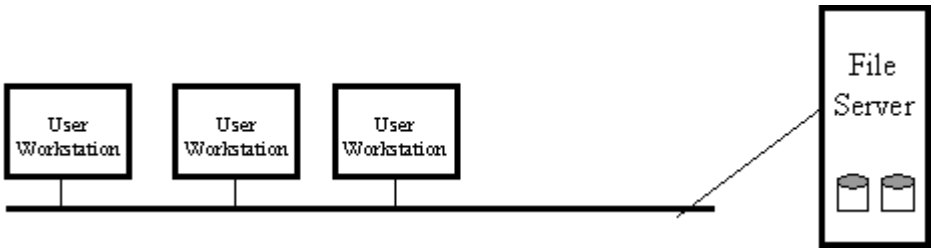
Hardware Models

The Workstation-Server Model

In the Workstation-Server Model the distributed system is composed of a network of workstations where each workstation provides local processing capability and an interface to the network. The model associates a workstation with each user of the system from which local and/or remote processing can be initiated. Some workstations may be assigned dedicated roles as file servers, print servers, time servers, name servers or secure authentication servers. Workstations may be diskless, depending only on the communication interface and the operating server nodes to obtain various system services.



Diskless workstations have many advantages. They are cheap, quiet and easy to maintain. Backing up the disks on a single file server is simpler than backing up a collection of individual disks. It also means the file system on every workstation looks the same for each user when they log on, which facilitates user mobility.

Diskless workstations place a burden on the communication system and centralized file services can result in a bottleneck which mitigates against scale. Sometimes a local disk is useful because it can be used to store frequently used executables in the form of a cache or it can be used to store temporary files generated during file processing applications or it could store virtual memory pages locally improving performance.

In the Workstation-Server Model, idle workstations can be used to improve performance by sending them work which can be done in parallel with other activities.

In Unix systems a remote shell command allows a process to be started by a remote shell on a chosen workstation. The format of the command is "rsh machine command". Notice that the user must identify the chosen machine. This places responsibility on the user for identifying idle workstations and also for keeping track of all machine names. The process cannot be removed until it is finished without destroying its computation. The process must also work in the environment of the remote machine, including a possibly different file system.

What we would like is simply to execute "command" at our workstation and have the system automatically find an idle workstation and run the process there providing a seamless environment for it. To this end, modern efforts at enabling remote computation concentrate on how to find/identify an idle workstation, how to migrate a process elsewhere when required and how to achieve this in a transparent way so that the process's computation is not affected by system activity.

The Processor Pool Model

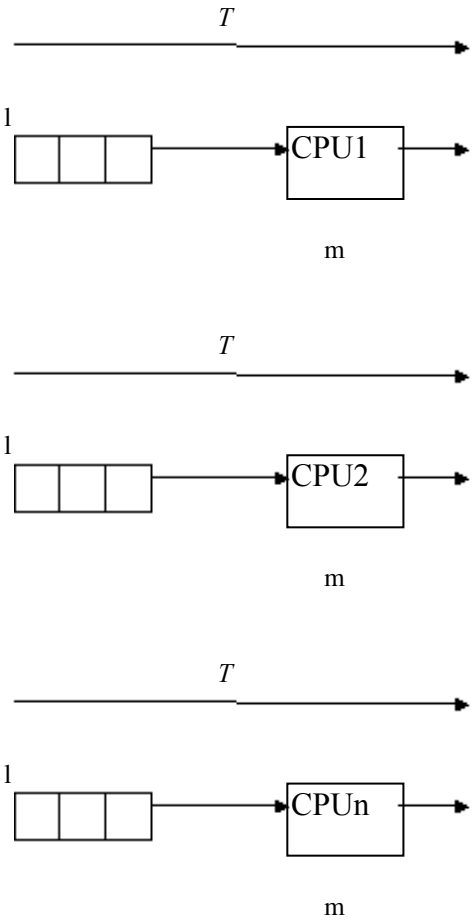
A processor pool consists of a large bank of CPUs each with its own local memory or possibly operating with shared memory. In this model, all processing is done by the processor bank, with users initiating computation through graphics terminals. Other operating system services may be provided by dedicated server machines. User processes are submitted to the processor pool which can dynamically allocate the necessary number of CPUs to perform the task. This is quite similar to the traditional timesharing approach except the processor bank is composed of low-cost microprocessors and can be expanded as required. There is another cost benefit in that only cheap X terminals need be supplied to the user and there are maintenance advantages similar to those for diskless workstations.

The main argument for the processor pool model comes from queuing theory.
If the average number of requests per second to a system is l and it can process m requests per second then it can be proven that the mean time between issuing a request and getting a complete response, *T*, is related to l and m by the formula:-

$$T = \frac{1}{m - l}$$

Of course m must be greater than or equal to l for the system to be able to cope with the load.

Now if we take n processors, each with its own queue then we get:-

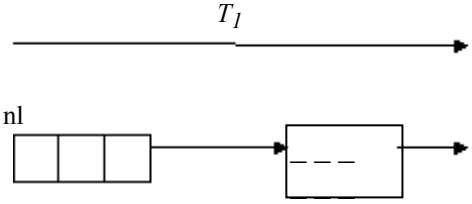


where

$$T = \frac{1}{m - l}$$

for each processor.

Now if we combine the processors into a processor pool with a single queue we get:-



where

$$T_1 = \frac{1}{nm - nl} \frac{T}{n}$$ i.e a Faster Mean Response Time

Assuming of course that all processors can be made equally busy all of the time.

Which Model is Best?

The choice of architectural model depends on the nature of the workload. Algorithms with a large amount of inherent parallelism such as simulations would be better suited to executing on closely coupled parallel architectures. If the primary use of the system is for resource sharing or supporting integrated personal productivity applications, or providing fault tolerance then the workstation model is more suited.

A hybrid based on both models, i.e. personal workstations with a processor pool resource available for intensively parallel computations may be a suitable solution.