Waiting Queues Applied in Parallel Processing

Alecu Felician
Academy of Economic Studies, Bucharest, Romania
E-mail: alecu.felician@ie.ase.ro

Abstract

Queuing theory, the theory of congestion, is a branch of operations research which explores the relationships between demand of a service and the delays suffered by the users of the system.

The study of the queues is very useful in fields like telecommunications, traffic control, determining the sequence of computer operations, predicting computer performance, health services, airport traffic, airline ticket sales, mining industry, manufacturing systems.

The execution queue of a parallel system is managed by a special program named scheduler who allocates tasks to available processors as well as implements the queue order and priorities. The queuing theory is a very useful tool to predict the performances of computer systems in general and of the parallel ones in particular.

Introduction

According to the dictionary, a queue is a file or line of persons. As a verb, "to queue" means to form a line while waiting for something. The etymology is from the Latin *coda*, which means tail. As defined by Saaty, a queue, or a waiting line, involves arriving items that wait to be served at the facility which provides the service they seek.

The total turnaround time of a process is formed by the queue wait time and the elapsed execution time. The queuing theory teaches us that it is important to not only reduce the job elapsed time by using faster processors but also the queue time through effective scheduling and management resources.

Content

Fundamentals of Queuing Theory

A queuing system is composed by three generic elements:

1. The arrival process of users in the system;
2. The order in which users obtain access to the service facility, once they join the queue;
3. The service process.

It is obvious that there are countless variations of queuing systems and networks. This is why a code has been used to describe the best-understood queuing systems. The code has the form *A/B/m*, where *A* and *B* are letter symbols that indicate the probability distribution of arrival and service times and *m* is the number of identical parallel servers from the queuing system ($m \in [1, \infty)$). The standard code letters used to express the probability distributions in queuing theory are shown in table 1. Table 2 contains the abbreviations of the most common queue disciplines.

Symbol	Description
M	Poisson distribution
D	Deterministic distribution
E_k	Erlang distribution
H_k	Hyper exponential distribution
G	General distribution

Table 1 – Probability distributions

Abbreviation	Description
FIFO	First in, first out
FCFS	First come, first served
LIFO	Last in, last out
LCFS	Last come, first served
SIRO	Service in random order

Table 2 – Queue disciplines

Another important parameter in the description of a queuing system is the system capacity that indicates the maximum number of users that can be in the service facility and in the queue at any time. On the other hand, the queue capacity indicates the maximum number of users that can be in the queue alone.

The Fundamental Model

The fundamental model includes general features so that the real models used in real life can be viewed as simply special cases. The model assumes a serving facility having *m* parallel identical servers. Also, the arrivals in the system and the service completions are performed in a Poisson manner. The queuing system operates under a FIFO/FCFS queue discipline.

The best-known and most widely used models of queuing theory (M/M/1 and M/M/m with finite or infinite capacity) are just particular cases of the fundamental model.

Extensions and Variations

The extensions and variations of the classical models can be developed by multiplying the arrival rates and/or service rates with some time dependent values. In such a way it is possible to implement phenomena like *reneging* and *balking*. *Reneging* represents the situation when a user that already joined the queue become discouraged after a while and leaves the system without getting the service. *Balking* is the phenomenon in which a prospective user of the system decides not to join to the queue but to go elsewhere in order to obtain the service.

Such extensions and variations also allow to speed up the service whenever the queues grow very long.

Queuing Systems With Priorities

In the case of the classical models there is no distinction made among different classes of users with respect to the type or the length of the service requested. Priorities are used to differentiate among classes of users as shown in figure 1. The users from the system are separated into *r* distinct classes and each class is assigned to a priority number. By convention, the smaller the priority number, the higher the priority of the class.

As soon as the service for a particular user is completed, the system is choosing the next user from the class having the highest priority. Within classes, the queue disciplines are determined according to FCFS, LCFS, SIRO or another one.

In systems that are using *preemptive* priorities, a user having a higher priority is never waiting for a lower priority one. The service of the current user is interrupted immediately upon arrival of a user belonging to a class with higher priority than that of the present facility occupant. In some systems, service to the ejected user is continued from the point it was interrupted while in other systems the service is started again from the beginning. Also, in same cases, an ejected user will be assigned, in compensation, to a priority class higher than the original one. On the other hand, n*onpreemptive* systems never interrupt a service to a user, even if one with a higher priority arrived in the queue.

Many queuing systems implement different priority rules for different classes, so that users from a high priority class may obtain preemptive service and users from middle classes may enjoy only nonpreemptive priority over users belonging to lower priority classes.

Queuing Systems and parallel processing

To obtain a faster execution time, a parallel program is usually divided into independent tasks that will be executed concurrently. Two tasks are independent each other if the same result is obtained if the tasks are executed sequentially in any order or in parallel.

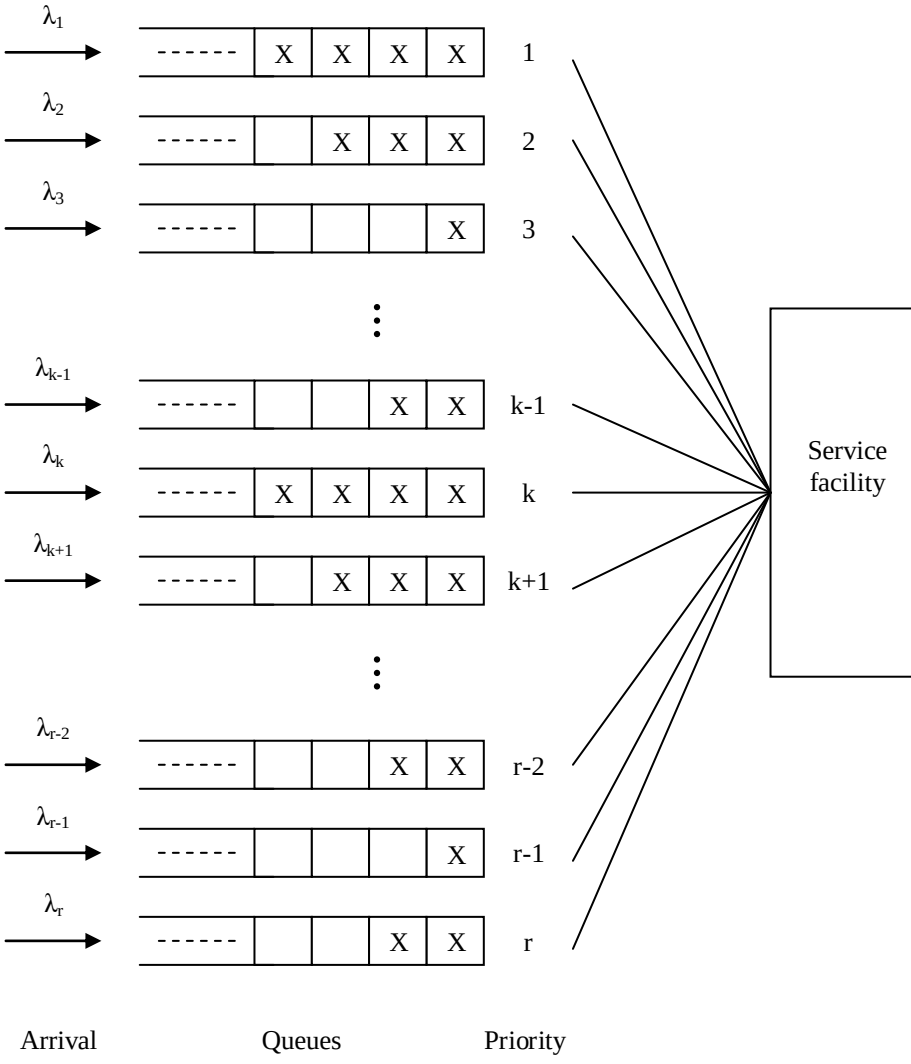

Figure 1 – Queuing System with *r* classes of priorities

Basically, an execution queue is a list containing ready to be executed processes. If this list is located in the system shared memory, the access to the list has to be done in a critical section in order to avoid the possibility of an execution of a process on two different processors in the same time. The mutual exclusion is used to access the shared resources and it is implemented using the classical mechanisms from uniprocesor systems like barriers, semaphores, monitors, etc. If a processor enters the critical section, it has exclusive access to the list of the ready to be executed processes. Based on the queue discipline and priorities, the processor will pick up a process from the list. Than, it will leave the critical section and will execute the selected process.

The effective type of the queuing systems used to describe the execution of a parallel program depends by the probability distribution of the service facility servers. The most common model used is the M/M/m one because usually the service distribution is a Poisson one. An M/G/m model has to be implemented if processes can use unlimited time to complete their execution. For a data-parallel program, the same instructions will be executed on different data sets and the final result will be obtained by combining the partial ones. This is why we can assume that the services are distributed exponentially negative so an M/M/m model could be successfully used.

A grid is a collection of machines that contribute any combination of resources as a whole. It tries to create the illusion of a virtual single powerful computer instead of a large collection of individual systems connected together. A grid network can be considered like a single center where the service demands are scheduled on the individual computers from the grid. There is a single queue of jobs for all processors from the network. The total service rate of n processors from the grid can be significantly les then n times the rate of a single processor because of competition for software locks (such of those controlling access to the shared queue of jobs) and interference in accessing shared resources. On the other hand, the effective service rate of the grid network is not constant but depends on the number of jobs queued at center. Let's consider a four processor system. Ideally, if four or more jobs desire service at the center, all four processors can be kept busy, and the effective service rate of the center is its maximum rate. However, if less than four jobs are queued at the center, some of the machines will be idle, and so the effective service rate will de reduced correspondingly. Figure 2 graphs effective service rate as a function of the queue length for a four-processor system. Service rates increase with queue length until all four systems are busy, after which increasing the number of jobs contending for the processors does not result in any increase in effective service rate. The dashed line illustrates the ideal growth in service rate and the other one represents the effect of contention.

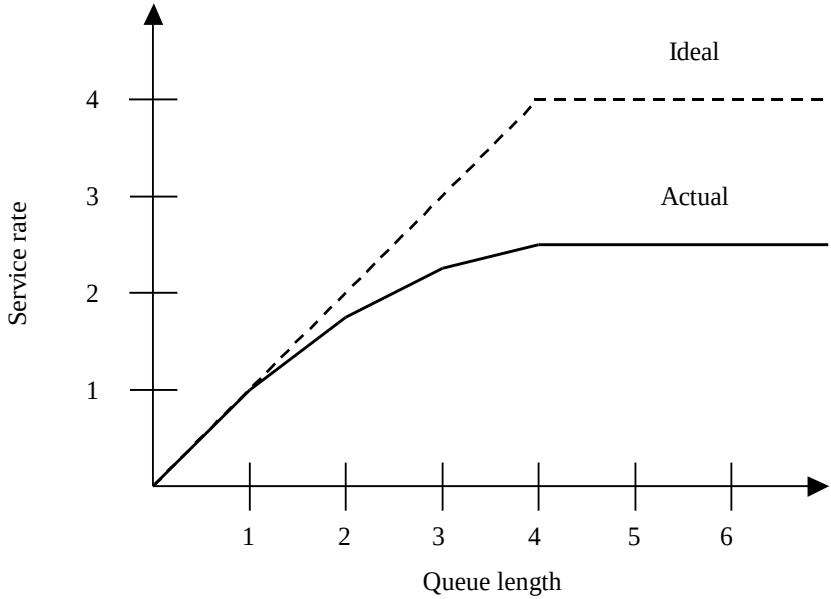

Figure 2 – Service Rate Function of a Four Processor Grid Network

The queuing theory is a very useful tool to predict the performances of computer systems in general and of the parallel ones in particular. The average quantities of interest (expected waiting time in the queue and in the system, expected total number of users in the queue and in the system) offer us a clear picture about the system performances and the ways to improve them. The queuing theory will help us to find out how to tune the system parameters in order to increase the system efficiency.

References

1. G. Boldur-Lățescu, I. Suciu, E. Țigănescu, *Operational Research with Applications in Economy*, A.S.E., 1990
2. Gh. Dodescu, B. Oancea, M. Raceanu, *Parallel Processing*, Ed. Economica, Bucharest, 2002
3. D. Gross, C. M. Harris, *Fundamentals of Queuing Theory*, Wiley, NY, 2003
4. H. F. Jordan, H. E. Jordan, *Fundamentals of Parallel Computing*, Prentice Hall, 2002
5. J. Joseph, C. Fellenstein, *Grid Computing*, Prentice Hall, 2003
6. A. S. Tanenbaum, *Distributed Operating Systems*, Prentice Hall, 1995