Queuing Systems and Parallel Processing

Pre-Assistant Lecturer Felician ALECU
Economy Informatics Department, A.S.E. Bucharest

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. Queuing theory is very useful in telecommunications, traffic control, determining the sequence of computer operations, predicting computer performance, health services, airport traffic, airline ticket sales, mining industry, manufacturing systems. In our days, it is very common to have systems where users remain stationary at separated locations while the servers visit them and provide the required service. Such a system is called spatially distributed queue. The execution queue of a parallel system is managed by the scheduler who allocates tasks to available processors as well as implements the queue order and priorities.
***Keywords**: Queuing system and theory, queuing network, spatially distributed queue, scheduling, parallel systems efficiency.*

A queuing system is a generic model that comprises three elements: a user source, a queue and a service facility that contains one or more (possibly an infinite number of) identical servers in parallel. Each user of the queuing system passes through the queue where he may remain for a period of time (positive, possibly zero) and than is processed by a single server because of the parallel arrangement of the servers. Once a user has left the server, after obtaining the service, the user is considered to have left the queuing system as well. [Bol90]

A queuing system is formed from three generic elements (figure 1):

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.

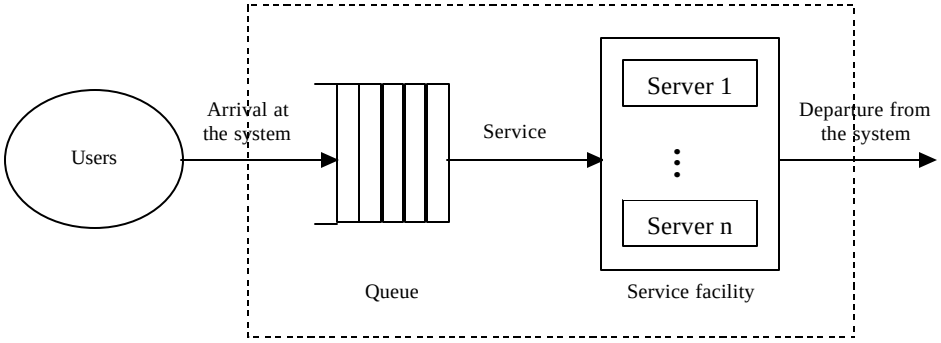

Fig. 1. Queuing system

A queuing network is a set of interconnected queuing systems.[Gro03] The user sources for some of the queuing systems in the network may be other queuing systems in the same network (figure 2).

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 ($m \in [1, \infty)$).

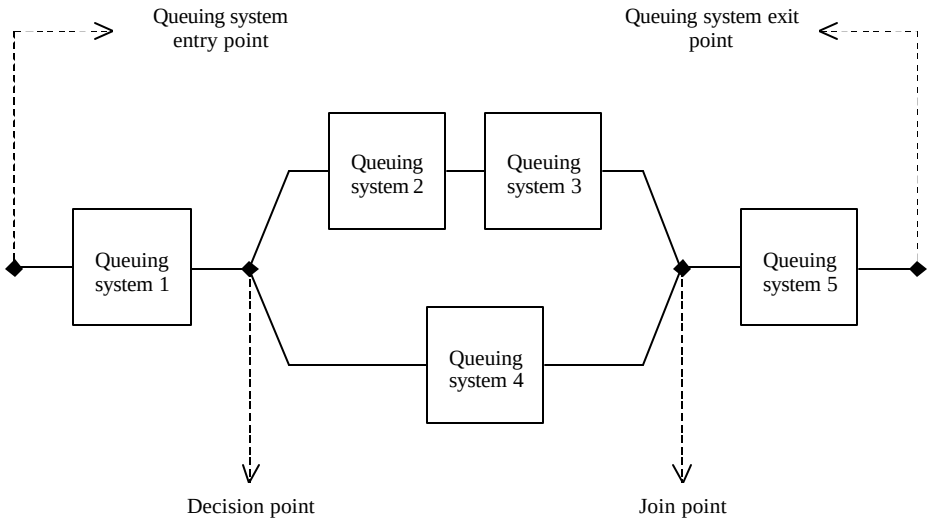

Fig. 2. Queuing network

Table 1 shows the standard code letters used to express the probability distributions in queuing theory. Table 2 contains the abbreviations of the most common queue disciplines.

Table 1 – Probability distributions

SYMBOL	DESCRIPTION
M	Poisson distribution
D	Deterministic distribution
E_k	Erlang distribution
H_k	Hyper exponential distribution
G	General distribution

Table 2 – Queue disciplines

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

The system capacity, another important parameter in the description of a queuing system, 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.

A stationary located server (like an ATM) where the users queue up to be served represents the classical perspective of a queuing system. In our days, it is very common to have systems where users remain stationary at separated locations while the servers visit them and provide the required service. Such a system is called spatially distributed queue. An example of a spatially distributed queuing system is the ambulance service.

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.

Any computer, sequential or parallel, implements waiting queues to properly manage the access to the system shared resources like processor, memory, peripheral devices and so on. Usually, there is a queuing system for each shared resource from the system. [Chi95] The resource represents the server and the tasks that try to access the resource concurrently form the users of the waiting system. If the shared resource is the processor, the waiting queue is known as execution queue. Execution queues make the transition from the sequential programming to the parallel one.

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 uni-processor 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.

A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. Based on this definition, a parallel computer could be a supercomputer with hundreds or thousands of processors or could be a network of workstations.

According to Tanenbaum, a distributed system is a set of independent and interconnected computers that appear to the user as a single one. The computers can communicate and collaborate to each other using software and hardware interconnecting components. The computers have to be independent and the software has to hide individual computers to the users. Multiprocessors (MIMD computers using shared memory architecture), multi-computers connected through static or dynamic interconnection networks (MIMD computers using message passing architecture) and workstations connected through local area network are examples of such distributed systems.

A distributed operating system is an operating system used on a distributed system. It is the extension for multiprocessor architectures of multitasking and multiprogramming operating systems. A distributed operating system is a special kind of software used on a distributed system. It manages the system-shared resources used by multiple processes, the process scheduling activity (how processes are allocated on available processors), the communication and synchronization between running processes and so on.

Multiprocessors are known as tightly coupled systems and multi-computers as loosely coupled systems. The software for parallel computers could be also tightly coupled or loosely coupled. Combining loosely and tightly coupled hardware and software we can identify four distributed operating systems categories but the loosely coupled software and tightly coupled hardware case is not met in practice (table 3).

Table 3 – Types of distributed operating systems

		Software Type	
		loosely coupled	tightly coupled
Hardware Type	loosely coupled	*Network operating systems*	*Real distributed operating systems*
	tightly coupled	-	*Multiprocessing operating systems*

A multiprocessing operating system is a multitasking operating system running on a multiple processor system. There is a single list of ready to run processes. When a new process is ready to run it is added to the list located in the shared memory area. Any process can access the list. When a processor is free, it extracts a process from the list and executes it. The operating system has to implement mutual exclusion mechanisms (semaphores, monitors, locks or events using various protocols) in order to protect the concurrent accesses to the ready to run processes list. Using these mechanisms, a processor has exclusive access to the processes list and it can extract the first list entry. Then, the list is released and the process is executed. Because the running time of a parallel program on such a system is finite, we can presume that the program will be divided into a finite number of processes. The queuing model associated to a parallel system running a multi-

processing operating system will be an infinite capacity one. The service facility is formed from m servers running in parallel, where m is the number of the processors from the system.

On the opposite side there are network operating systems and real distributed operating systems. A copy of the operating system is running in every processing node so the distributed system has as many waiting queues as the number of the processors. In every processing node there is a copy of the operating system that manages the processor execution queue. Assuming that the execution time of a parallel program is finite, we can presume that the program will be divided into a finite number of processes too. This is why we can associate to each processing node a queuing model with a single server and arrivals from a finite population. The queuing systems at the nodes level are interconnected together into a queuing network because the processors have to communicate and synchronize each other. The processes may have specific dependencies that may prevent them from executing in parallel in all cases. For example, a process may require the output produced by certain task and it cannot be executed until that prerequisite task has completed executing.

The waiting systems presented above could become queuing models with arrivals from an infinite population if the parallel program needs a very long time to complete its execution and the number of the generated processes will increase substantially.

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.

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 averages 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.

Usually, the system scheduler manages the queuing system. Its main responsibility is to scheduled tasks to the system processors but it also implements the queue discipline and the priority classes.[Dod02]

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.

References

[Bol90] G. Boldur-Latescu, I. Suciu, E. Tiganescu, *Operational Research with Applications in Economy*, A.S.E., 1990

[Chi95] I. Chiorean, *Parallel Computing*, Ed. Albastra, Cluj-Napoca, 1995

[Dod02] Gh. Dodescu, B. Oancea, M. Raceanu, *Parallel Processing*, Ed. Economica, Bucharest, 2002

[Gro03] D. Gross, C. M. Harris, *Fundamentals of Queuing Theory*, Wiley, New York, 2003