PERFORMANCE PRDICTION OF DATA PARALLEL PROGRAMS USING QUEUING THEORY

Felician ALECU, *Academy of Economic Studies*, Bucharest

Abstract: *Queuing theory is the mathematical study of waiting lines. The etymology is from the Latin coda, which means tail. As a verb, "to queue" means to form a line while waiting for something. A queue, or a waiting line, is a file or line of persons that wait to be served at the facility that provides the service they seek. The study of waiting system 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 called scheduler who allocates tasks to available processors as well as implements the queue order and priorities.*

Keywords: Queuing theory, waiting systems, parallel processing, scheduling, data parallelism.

1. INTRODUCTION

According to the dictionary, a queue is a file or line of persons. .Queuing theory, also known as the theory of congestion, is the mathematical study of waiting lines.

A queuing system is formed from three generic elements:

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

Each user of the queuing system passes through the queue where he may remain for a period of time and then is processed by a single server because of the parallel arrangement of the service facility stations. Once a user has left the server, after obtaining the service, he is considered to have left the queuing system as well.

It is obvious that there are countless variations of queuing systems. This is why a code has been used to describe the best-understood queuing systems. The code has the form $A/B/m$ (or $A|B|m$), 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 ($m \in [1, \infty)$).

Table 1 shows the standard code letters used to express the probability distributions in queuing theory.

Table 1. **Probability distributions**

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

Other standard abbreviations are also used to indicate the most commonly encountered queue disciplines, as shown in the Table 2.

Table 2. **Queue disciplines**

Abbreviation	Description
FIFO	First In, First Out
FCFS	First Come, First Served
LIFO	Last In, First 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 queuing network is a set of interconnected queuing systems. The user sources for some of the queuing systems in the network may be other queuing systems in the same network.

For a long period of time, scientific applications (like numerical simulations of complex systems) represented the most important users of parallel computing technology. Today, the developments in the parallel computers field are driven by the commercial applications that require a computer to be able to process large amounts of data in sophisticated ways. These data-intensive applications will define the parallel configurations that will be used in the future.

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.

A data parallel program is a program that executes in parallel the same set of instructions over different data sets.

2. APPLICATIONS OF QUIUEING THEORY IN PARALLEL AND DISTRIBUTED PROCESSING

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. 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.

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.

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 single 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.

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.

3. PERFORMANCE PREDICTION OF DATA PARALLEL PROGRAMS

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, where *m* represents the number of the processors from the parallel system.

A data parallel program can be executed in a parallel environment (a single computer containing multiple processors) or in a distributed one (multiple uniprocessor computers connected through the network). In both cases it is needed that one processing node (processor or computer) to be used as a master. This node will divide the data space into different pieces that will be concurrently processed by the regular nodes. This node will be also used to run the scheduler – a special program that implements the queue discipline and the priority classes according with a scheduling policy. Such a system is considered like a single center and the service demands are scheduled on the individual computers from the grid. There is a single queue of jobs for all processors from the system.

The initial data space will be divided into multiple blocks that will be sent by the scheduler to the different processing nodes. The same program will run in every node so the same set of instructions will be applied. Finally, the partial results will be also managed and assembled by the master node in order to obtain the final results.

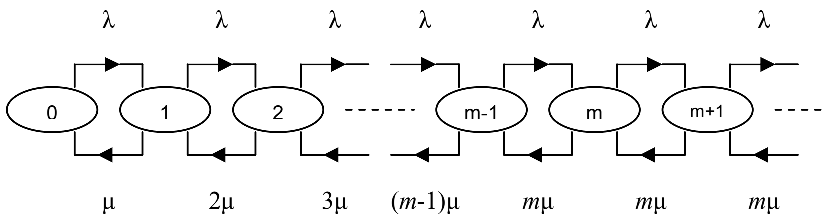
We will presume that the processing nodes from the system have almost the same characteristics. This is why we can consider all nodes as operating at the same service rate, μ, computed based on the average performance of individual processors.

The arrival rate of the customers in the waiting system is equal with the speed of the program used to divide the data space into multiple blocks and it mainly depends by the desired granularity of the process. This is why we can consider that the arrival rate of the customers is constant and equal with λ. Different block size will generate different values for the parameter λ.

The queuing system that describes better such a system is the *M/M/m* one with infinite capacity – a parallel system composed by *m* processors working in parallel and having a Poisson probability distribution for arrival and service rates. The queue discipline is considered to be FIFO (First In, First Out) or FCFS (First Come, First Served).If all the stations of the service facility are busy, the next consumer will be assigned to the first station that become free. If more than one stations are free, the next customer will be arbitrary assigned to one of them. The maximum number of customers that can be present in the service facility in the same time is equal with *m*, the number of stations. The probability distribution function (*pdf*) for incoming data sets is considered to be negative exponential with mean *1/λ* and the pdf for service times is negative exponential with mean *1/μ*.

Figure 1 shows the state transition diagram for our *M/M/m* model with infinite capacity. The system state is described by the state variable *n*, the total number of users in it. So, we can say that the system is in the state *x* whenever there are *x* users in the waiting system.

Figure 1. **State transition diagram**



The customers of our particular system are actually data blocks and the service facility is formed by the processors present in the system. Based on the properties of the *M/M/m* model, we can obtain the average quantities of interest regarding our waiting system.

The average number of customers in the queue can be computed using the following formula:

$$\overline{N}_q = \frac{1}{\sum_{n=0}^{m-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^m}{m!} \frac{1}{1-(\lambda/m\mu)}}_{0} \frac{(\lambda/\mu)^m}{m!} \frac{\lambda}{m\mu} \frac{1}{(1-\lambda/m\mu)^2}$$

One of the best known results of the queuing theory is the Little's formula that basically says that the average number of the customers can be obtained by multiplying the arrival rate with the average time spent by a customer in the system:

$$\overline{N} = \lambda \cdot \overline{W}$$

A special form of the same formula can be used to obtain the average number of customers from the queue based on the service rate and the average time needed to a customer to reach the service facility.

$$\overline{N}_q = \lambda \cdot \overline{W}_q$$

Using the Little's formula, the average quantities of interested can be expressed as the following:

- the average time spent by a customer in queue - $\overline{W}_q = \frac{\overline{N}_q}{\lambda}$
- the average time spent by a customer in the system - $\overline{W} = \overline{W}_q + \frac{1}{\mu} = \frac{\overline{N}_q}{\lambda} + \frac{1}{\mu}$
- the average number of the customers from the queuing system - $\overline{N} = \lambda \cdot \overline{W} = \lambda \frac{\overline{N}_q}{\lambda} + \lambda \frac{1}{\mu} = \overline{N}_q + \frac{\lambda}{\mu}$

4. CONCLUSIONS

Using the queuing theory background (see the above formulas), it is possible to predict a data parallel program performances and to have a complete picture of the time needed to obtain the final results without running it on a real system. It will be possible to perform a tuning operation over the main system parameters, like the dimension of the block and the type of the processors to be used, in order to achieve the desired performances.

These kinds of results could also help the developers to design the system scheduler and the data parallel program in such a way in which the final results will be obtained in the shortest time possible and at an acceptable level of costs.

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

1. S. Asmussen, *Applied Probability and Queues*, Springer, 2003
2. D. Gross, C. M. Harris, *Fundamentals of Queuing Theory*, Wiley, New York, 2004
3. Gh. Dodescu, B. Oancea, M. Răceanu, *Parallel Processing*, Economic Publishing House, Bucharest, 2004
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. G. Boldur-Lăţescu, I. Suciu, E. Ţigănescu, *Operational Research with Applications in Economy*, A.S.E. Publishing House, 1990