Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.7, No.2, 2016

Task Scheduling for Multiprocessor Systems Using Queuing Theory

Dr.Kusay Hameed Al-Salami[1*] Zaid Taha Sawadi[2]
1.Department of Business Administration, College of Administrative and Financial Sciences, Cihan University, Erbil, Iraq
2.University of Mosul, Mosul, Iraq

Abstract
This research focuses on comparing different multi-processor task scheduling algorithms. Each algorithm has been simulated using one of queuing theory models in Operations Research (OR) science to evaluate its behavior and efficiency. The comparison includes an analysis of the behavior of central processing unit (CPU) when receiving number of jobs at four random job duration patterns that are; (random, ascending, descending, and volatile low-high). Microsoft Excel 2010 was used to form the data of each case, and the result shows convergence and divergence among the studied algorithms at different patterns. Also it has been found that the Fleischer algorithm is very efficient in enhancing and minimizing the waiting duration for each job at the total job queue of the CPU.
Keywords: Operations Research, Queuing Theory, Multiprocessor, Scheduling Algorithms, Simulation.

1. Introduction

In multiprocessor systems, an efficient scheduling of the paralleled program on to the processor; is vital for achieving a high performance and minimizing the entire execution time. The problem of multiprocessor scheduling can be stated as finding a schedule for general tasks to be executed. Thus the schedule length can be minimized and the jobs can be completed as early as possible. Task scheduling problem is a key factor for the parallel multiprocessor system to gain better performance, reduce processing time, and increase processor utilization (R. Al-Ekram 2004).

In order to highlight the issues mentioned in this section, the research aims to achieve the following objectives:

1. Studying the different ways of task arriving to the multi-processor scheduling.
2. Building a model to simulate the scheduled work and use it to apply different algorithms; that are used in the preparation of the queuing theory.
3. Finding the scheduling algorithms that are guaranteed to be not so far from the optimal algorithm.
4. Finding a scheduling algorithm that can increases the CPU efficiency, through keeping it running longer.

There is a great debate about how to ensure the availability of data at very high speed. This is done through using a number of processors on a single chip called (multi-core processer), or using separate processers on the same mother board. This issue highlights the following points:

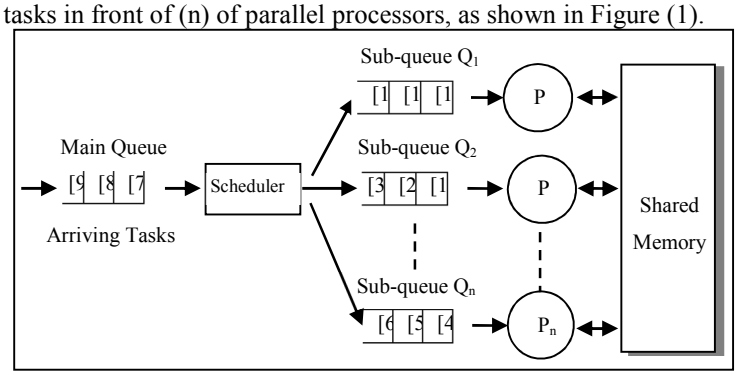
1. Data transmission speed between Dual-Core.
2. Tasks that are waiting to enter the processors for processing.

The research objectives will find answers to the following questions:

1. What is the optimal algorithm which is able to schedule the arrived tasks to a multi-processor?
2. What is the impact of using the optimal algorithm on improving the efficiency and increasing the speed of computer processors?

The importance of this research comes from:

1. Using queuing theory to solve the research's problem.
2. Task scheduler plays an important role in reducing the waiting time for tasks to be processed at CPU. This process was done previously with a possibility of sharing data on shared memory.
3. Scheduling set of tasks in front of (n) of parallel processors, as shown in Figure (1).



In Figure 1, it is clear that the main queue (Q) is receiving the arriving tasks, that are forwarded by the

scheduler to enter a sub-queue (Q_1, Q_2, ..., Q_n) based on one of the scheduling algorithms. Then, the tasks will be received by the available processors (P_1, P_2, ..., P_n), that are characterized by their shared memory used in all processors.

2. History and Definition of Queuing Theory

The first attempt to identify the concept of queuing was done by Erlang (1909). This study was implemented on the Copenhagen Telephone Company, for the purpose of avoiding the long waiting time for customers who are willing to receive communication services. The problem has been solved through providing a number of essential communication offices (K.H. Al-Salami 2013).

John (2009) also presented a different model for hospitals. He found that the use of OR techniques had a significant impact in reducing the waiting time for patients while receiving treatment (J. Pirolo *et al.* 2009).

Queues (or waiting lines) help in providing services in an orderly fashion. For instance, previously in the airport terminals, the passengers were waiting for a long time because they were forming separate queues in front of check-in counters. While nowadays, only one line of passengers feeding into several check-in counters. The airline management realized that a single line policy serves are better for passengers as well as employees. Such a conclusion has come from analyzing the way by which a queue is formed and the service is provided. The analysis is based on building a mathematical model representing the process of passengers' arrival to the queue, the rules that they should follow, and the time required to serve them (S. Yashkov 1989).

Queuing systems can be described as particular entities (users) that ask for services. In each serving system, one can distinguish the arrival process, service process and one or more station or server. The general assumption is that one station cannot at the same time serve two or more arrival entities. If the station is busy, the user has to wait for the service. During the service time, the entities can pass from one or more station before departing the system (P. Goran *et al.* 2008).

3. Characteristics of Queues Model

Queues System consists of three main elements, as explained in the following: (A.T. Hamdy 2007, S. Yashkov 1989)

1. Customers Access Process: Customers are those who are in need of the service. Customers have a (Source) where they are generated. This source may be (Finite) or (Infinite), and customers arrive at service stations individually or collectively, See Figure (2).
2. Customers Service Process: is performed through service channels available at the service station. There may be single service channel or several channels serving a number of customers at the same time. These channels are referred to as parallel. For example, many employees are there to receive messages at the post office. Similarly, the departure of customers may be either at specific times i.e. the service period for each customer is limited, or it is random where the number of departing customers is a separate random variable which has a probability distribution.
3. Service Method (or Principle): This is the method by which the customers are chosen to be serviced. The most common principles for customer servicing are:
   a. Service in the order of access i.e. the first customer who arrives is the first customer who is serviced. FIFO stands for First-In, First-Out.
   b. Service in the reverse order of access i.e. the first customer who arrives is the last customer who is serviced. (FILO) stands for First In Last Out.
   c. Random Service: Where the customers are chosen randomly to be serviced regardless of their time of access (SIRO) stands for Selection In Random Order.
   d. Priority Service: (PRI) where the priority is given to some customers.
4. Access and Service Distributions: The number of customers who arrive at the service station until time t or during the interval (0,t) is a random variable N(t), then the probability distribution function $a_n(t)$ for the random variable N(t) is given by Equation (1):

$$a_n(t) = pr\{N(t) = n\} = \frac{(\lambda t)^n e^{-\lambda t}}{n!}; n = 0,1,2,... \quad (1)$$

5. Distribution of Time Interval between Customers Access: the number of customers N(t) who arrive during the interval of time (0,t) is subject to Poisson distribution, and consider, as before, that the random variable T represents the interval of time between two consecutive accesses. Then probability density function f(t) for the time interval (t) between any two successive arrivals (where $t \geq 0$) is given by Equation (2):

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & ; \quad t \geq 0 \\ 0 & ; \quad t < 0 \end{cases} \quad (2)$$

The above mentioned function, as known, is probability density function for (negative) exponential distribution

with parameter λ.

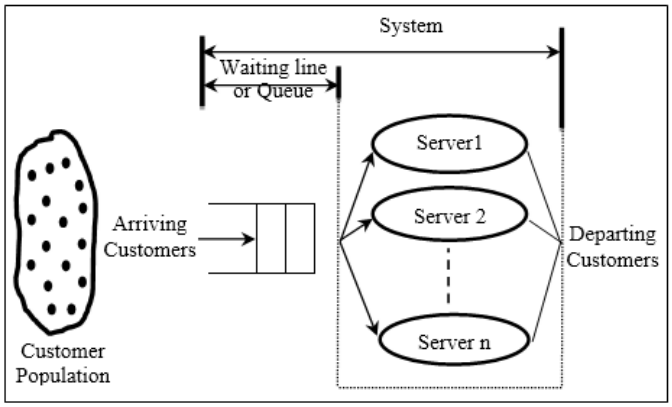


Figure 2: Service Center with *n* of Parallel Service Facilities
Source: W. Andreas, (1999) & A.T. Hamdy, (2007)

6. Departure Distribution: First, one has to notice that departure does not occur if the system is empty (does not contain customers) and the departure of one customer means the end of his service. So, if we suppose, as the case is in the access, that the departure occurs at rate μ, then the distribution of the number of completed services (i.e. the number of departing customers) during an interval of time (0,t) is Poisson distribution for the parameter μ_t. If we suppose that the system has started with N of customers who are departing the system with rate μ without allowing the entry of new elements into the system and if we represent $q_n(t)$ to the distribution of number of departing customers during (0, t), then:

$$q_n(t) = \frac{(\mu t)^n . e^{-\mu t}}{n!} \quad ; n = 1,2,..., N-1; \tag{3}$$

$$q_n(t) = 1 - \sum_{n=0}^{N-1} q_n(t) \tag{4}$$

This means, as in the case of access, that the distribution of interval of time between two consecutive services is the exponential distribution with parameter μ_t. Or, the probability density function of the distribution of service time for a customer is given by Equation (5):

$$S(t) = \begin{cases} \mu \quad e^{-\mu t} & ; t \geq 0 \\ 0 & ; t < 0 \end{cases} \tag{5}$$

7. Terms: As previously referred, access and departure have its own distributions, and customers may generate from either finite or infinite source. System capacity may be limited or unlimited, and the system may contain one channel or many parallel channels. In order to refer to these points, the following hexagonal code is used:

Where the characters in this code stand for the following: A / B / C / X / Y / Z

A: stands for the distribution of the number of customers who access the system. M code is usually used to refer that this number of customers follows Poisson distribution.

B: stands for the distribution of customers' number who depart the system. M code is usually used to refer that the number of departing customers follows the negative exponential distribution.

C: stands for the number of symmetrical parallel service channels C = 1, 2, 3,...,∞.

X: stands for the service principle and some service methods that are noticed: FIFO, FILO, PRI and SIRO. The code GD will be used to refer to general service principle.

Y: stands for the system capacity.

Z: stands for the source capacity.

8. Efficiency "Performance" Criteria: through which one can study and analyze the queue systems and then reset or reconstruct these systems to reach the required targets. In this concern, there is an interest in the following criteria:
   1) The rate of time interval, which a customer waits in the system and is represented by W.
   2) The rate of time interval, which a customer waits in the queue and is represented by W_q.
   3) Customers' rate number in the system, which represented by L.
   4) Customers' rate number in the queue, which represented by L_q.

4. Markovian Systems

The common characteristic of all Markovian systems is that all interesting distributions, namely the distribution of the inter-arrival times and the distribution of the service times are exponential distributions and thus exhibit the

Markov property (A. Ivo *et al.* 2001, F. Taher *et al.* 2006, W. Andreas 1999).

(M_i/M/C):(NPRP/∞/∞) Model: This model assumes that all customers have the same distribution of service time, regardless of who owns the absolute merit. Also all service channels have the same exponential distribution to the service time μ, access to queue K with the greatest priority be the access rate λ_k and follow the Poisson distribution for each values (k= 1,2, ..., m) (A.T. Hamdy 2007).

5. Review of Control Processing Unit

Control Processing Unit (CPU) is responsible to perform all calculations and logical operations (program implementation). Usually the efficient use of processing unit (CPU Utilization) is measured depending on the tasks that fall in it, which are in the status of implementation (Running). The (Input / Output) operations need more time to be completed, compared with the implementation time of arithmetic and logical operations. The CPU in single-user operating systems; spend most of the time in waiting (Idle), which leads to low CPU efficiency. In a multi-user operating systems (multiple programs), the efficiency of the CPU must be raised using specific methods, such as allocating a specific time period for each work (program), or moving to carry out the following work when the previous work is in operation (I/O). The scheduling affects the behavior and efficiency of the system because it determines the processes that have to be waited or implemented (Z.M. Al-Kathe 2004).

6. Review of Simulation

Simulation can be defined as a technique that imitates the operation of a real-world system as it evolves over time. This is normally done by developing a simulation model. A simulation model usually takes the form of a set of assumptions about the operation of the system, expressed as mathematical or logical relations between the objects of interest. In contrast to the exact mathematical solutions available with most analytical models, the simulation process involves executing or running the model through time (W. Winston 2003).

The simulation technique provides an acceptable methodology for the study of complex systems. It includes the study of tasks' waiting time in single or multi processor systems, using different scheduling strategies. It can also build systems, that are used for pilot studies and testing, to solve complex issues with minimal cost and effort (J. Pirolo *et al.* 2009).

The systems are classified as discrete or continuous (W. Winston 2003):

1. In discrete system the variables are changing only at discrete or countable time points. A bank is an example of a discrete system, since the variables are changing only when a customer arrives or when a customer finishes being served and departs. These changes take place at discrete time points.
2. In continuous system the variables are changing continuously over time. A chemical process is an example of a continuous system. Here, the status of the system is changing continuously over time. Such systems are usually modeled using differential equations.

There are two types of simulation models presented in the following (W. Winston 2003):

1. A static simulation model is a representation of a system at a particular time point. It usually refers to a static simulation as a Monte Carlo simulation.
2. A dynamic simulation is a representation of a system as it evolves over time.

Within these two classifications, a simulation may be deterministic or stochastic. The deterministic simulation model contains no random variables; which the stochastic simulation model contains one or more random variables. Discrete and continuous simulation models are similar to discrete and continuous systems. Such models are called discrete-event simulation models. Discrete-event simulation concerns the modeling of a stochastic system as it evolves over time, in which the variables change only at discrete time points.

Most discrete-event simulation models have stochastic elements that mimic the probabilistic nature of the system under consideration. The focus here is on the modeling and generation of arrival processes. Stochastic models of arrival processes could be used to model the arrival times of customers to a queue, jobs to a job shop, or demands in an inventory system. These stochastic models are generalized to model events that occur over time. A close match between the arrival process model and the true underlying probabilistic mechanism associated with arrivals to the system of interest; is required for successful simulation input modeling (M.L. Lawrence 2006).

An example of the simulation process has been presented recently by M. Arashpour *et al.* (2015); in which a special purpose simulation model was created to apply the rate-driven production in the house building environment. This model focuses on assessing the qualitative performance of alternative project planning and control strategies. It analyses the theoretical aspects of rate-driven production and its consequential practical concerns. The superiority of rate-driven production in the construction environment was observed through minimizing the errors in determining the optimal quantity of houses under construction. Furthermore, different simulation experiments were developed to evaluate numerous what-if scenarios in construction environment. Finally, a simulation template was created to reach the maximum of production and limit the quantity of houses under construction based on the obtainable capability and real demand.

7. Multiprocessor Tasks Scheduling

Scheduling is the task of allocating resources (machines) to activities (jobs) optimally. A scheduling problem is characterized by the properties of the jobs, machines, processing constraints and the optimality criterion. The properties of the jobs involve the processing time of the jobs, the release dates, deadlines and priorities. The properties of the machines are the number of machines, processing speeds, and machine configuration (parallel or series). There can be a number of constraints for processing the jobs in the machines, e.g. precedence, penalties for lateness, capability of preemption; offline or online arrival of jobs (A. Susanne *et al.* 2000, R. Al-Ekram 2004). In 2005, Jorge and Vernon used multiprocessor scheduling algorithms efficiently and effectively to simulate the clinical trials process. Several treatment processes were included in these tests. Also Novel algorithm was used to simulate the multiprocessor using Round-Robin Scheduling. Despite of its complexity compared with the use of NaÏve Round-Robin algorithm scheduling, it has been proven that this is similar to the steering tasks of (CPU) processors. This method focuses on building the model without focusing on the details of the world outside the model (R. Jorge *et al.* 2005).

In 2013 (K.H. Al-Salami 2013) developed a new multiprocessors algorithm to improve the speed of computer processor. This is done depending on the operations research and statistical techniques. The new algorithm presents good results compared to the approved algorithms in computer processors.

Furthermore, an effective real-time multiprocessor scheduling algorithm was presented by H. Alhussian *et al.* (2014). In this algorithm the fairness rule was relaxed to preclude the massive number of generated preemptions and migrations. This can help in tolerating the deadline misses as a few number of tasks preemptions and migrations will be produced. This algorithm used a global job queue controlled by an increasing laxity. The minimum task's laxity value, the higher priority in executing the task is scheduled. Finally, it has been found that there is little quantity of task preemptions and migrations compared to the common and well known algorithms in this field.

The online multiprocessor scheduling algorithms described as the following (R. Al-Ekram 2004):

1. Job properties: **n** jobs, each one have a release date and a processing time available on release date.
2. Machine properties: **m** identical parallel machines.
3. Constraints: Non-preemptive and online arrival

Some of the online multiprocessor scheduling algorithms are presented in the following (R. Al-Ekram 2004):

1. **Graham's List Scheduling:** Graham's List scheduling is the oldest online multiprocessor scheduling algorithm, introduced in the mid-sixties. This algorithm always assigns an incoming job to the least loaded machine. The competitive ratio of this algorithm is (2 – 1/m) for all m. It assumes that the jobs are given in a linear list, which is equivalent to jobs that are arriving in an online manner.
2. **Bartal's Algorithm:** After a quarter century from introducing Graham's List algorithm, Bartal proposes an algorithm with a competitive ratio of (2 – 1/70) ≈ 1.986 for all (m>70). The problem with Graham's List algorithm is trying to keep a balanced load among all machines. So when a large job arrives it makes the make-span will be far from the optimal. The idea behind Bartal's algorithm is to maintain an imbalance among the loads of the machines, so that when a large job arrives it can always be assigned to a machine with lighter load. More specifically the algorithm tries to keep approximately 44.5% of the machines lightly loaded. Any new small job will be assigned to one of them if it still preserves the goal. Otherwise, it will be assigned to a heavily loaded machine, which will always maintain the goal. When a large job arrives assigning it to a lightly loaded machine may violate the goal, but if it is assigned to a heavily loaded machine; the make-span will be increase immediately. In such case the only choice is to assign the job to a lightly loaded machine.
3. **Karger's Algorithm:** Karger introduced an algorithm that out performs List for any (m ≥ 6) and is at most 1.945 competitive for all m. When placing a new job, Bartal's algorithm can only choose either the first machine from the list of short machines or the first machine from the list of tall machines according to an imbalance criterion. Karger's algorithm is more general than Bartal's algorithm, as it chooses a machine from the full list of machines to place a new job; according to an imbalance criterion.
4. **Albers's Algorithm:** Albers's algorithm is similar but simpler than the previous two algorithms. It improves the competitive ratio to 1.923 for all (m ≥ 2). At any time the algorithm maintains a list L_l of (m/2) short machines and list L_h of (m/2) tall machines. A new job is either assigned to the shortest machine of L_1 or the shortest machine of L_2, depending on the sum ratio of heights machines in the two lists.
5. **Fleischer's Algorithm:** The algorithm given by Fleischer has better competitiveness than the previous algorithms as (m ≥ 64) and the competitive ratio approaches to: $1 + \sqrt{\frac{1+\ln 2}{2}} < 1.9201$. This is the best known deterministic result so far for the given online multiprocessor scheduling problem.

All algorithms that provide an improvement over Graham's List algorithm are based on the idea of maintaining an imbalance of load among the machines. When a new job arrives it is scheduled either on the shortest machine as long as it is short enough or on a certain taller machine if it can be done safely. Choosing a machine is done based on some specific imbalanced criteria related the algorithms (R. Al-Ekram 2004).

8. Empirical Section

This section presents using the simulation with a queue model (M_i/M/C):(NPRP/∞/∞) based on the models' priority in providing services. The proposed model matches the multiple processors that have been discussed previously in this research. In this section also the simulation process will be discussed based on different numbers of processor cycles (No. of cycles). Furthermore, the default values of simulations have been generated using three different cycle numbers; that are (5, 10 and 15 cycles). These numbers have been applied for scheduling four algorithms namely; (Albers, Bartal, Fleischer and Graham's List). It has been assumed that (164 jobs) will be arrived on the available processors.[1]

To obtain the desired results, (Microsoft Excel 2010) was used, and the necessary simulation processes for the required number of cycles were generated. This is done in order to reach the random values of the processor; that are (1 ≤ time in service ≤ 30). A random integer is returned in each run of Excel program,[2] where four different forms of value generation are used; that are (random, ascending, descending & volatile low-high). These forms utilized to generate values for the number of sessions that may encounter the processor (representing the pressure on the processor method). Then the generated numbers entered to Excel based on the algorithm type, in order to get the result of tasks scheduling on the processor.

The development in technology and computers has led to increasing the speed of processors to process more operations, but in general measurement unit (Hz):

$$Hz = \frac{cycles}{unit(second)} \quad \dots (6)$$

$$let\ \mu = 5\ MHz = (5 \times 10^6)\ \frac{cycles}{second}$$

Where Equation (6) is assumed that the processor speed (5 MHz) of Intel processor, and this value represents the service rate (μ) provided by the processor to service operations up to that. Equation (7) shows how many seconds needed in each cycles:

$$\frac{1}{Hz} = \frac{1}{\mu} = \frac{1}{\mu \frac{cycles}{second}} = \frac{1}{\mu}\frac{second}{cycles} = cycle\ time \quad \dots (7)$$

Where $(\frac{1}{\mu})$ represents the cycle time, and the arrival rate (λ) is measured by the unit ($\frac{\text{Instruction}}{\text{unit (second)}}$), and this must be unify with the units (μ) because the computer processor works in this way, and in the future; it is easy to calculate the value of traffic density parameter (ρ), as in Equation (8):

$$\lambda = \frac{sum(\lambda)}{\frac{latest\ arrival}{time\ in\ cycle}} \times \frac{cycle}{time} = \frac{sum(\lambda)}{\frac{latest\ arrival}{time\ in\ cycle}} \times \frac{1}{\mu} = \frac{\mu \times sum(\lambda)}{\frac{latest\ arrival}{time\ in\ cycle}}\frac{cycles}{second} \quad \dots (8)$$

Where:

1. Latest arrival time in cycles: is the time needed to access the system as a whole, so any new task will not be received.
2. Cycle's time: is the number of seconds per cycle.
3. Sum (λ) = Te (cycles): is the total rate of accessing the system.
4. Sum (λ_1): is the total rate for accessing the first column (1st processor).
5. Sum (λ_2): is the total rate for accessing the second line (2nd processor).
6. Te = Execution cycles time = Time to execution = Time in service: instruction's time need for implementation = the amount of time (number of cycles) for each instruction.
7. Cycles: is the electrical pulse.

These instructions are usually need specific time for implementation, which is typically in the range between (1 ≤ Te ≤ 30). The integer value represents the number of instruction cycles on the processor. It is clear that the time needed for service delivery is between (1 ≤ times in service ≤ 30).

At this stage the value of (ρ) can calculated as Equation (9):

$$\rho_i = \frac{\lambda_i}{\mu} = \frac{\frac{\mu \times sum(\lambda_i)}{latest\ arrival\ time}}{\mu} = \frac{sum(\lambda_i)(cycles)}{\frac{latest\ arrival}{time}\ (cycles)}$$

$$\therefore \rho = \sum_{i=1}^{n=2} \rho_i \times \frac{non\ stabile}{rate} = (\rho_1 + \rho_2) \times \frac{non\ stabile}{rate} \quad \dots (9)$$

Here the (non-stable rate) represents the average coefficient of unbalanced loads at each processor, which is

[1] Note: In the beginning, it is assumed to access 50 tasks to study the behavior of the system. After that, the tasks were increased to 100, and then to 150. At 164 tasks, the system got the stability in his behavior.
[2] M.I. Al-Adue, p.165, 2000.

multiplied by (ρ) value. The distribution of loads on processors has uneven policy, so many algorithms are transforming the task scheduler (jobs arrived) to one processor, and the other processor will receive a lighter task that need more time for implementation.

9. Simulating the Multiprocessor Work Using (M_i/M/C):(NPRP/∞/∞) Model

This model will be applied using all the preliminary data that are necessary to calculate the whole rules. An example of this model is the case study which applied to Fleischer algorithm. In this case study, the arrival rate (λ) for the main line has been postulated to each change (15 cycles). Also the service process time for each task is vary from others (volatile Low-High), which represents the (process service time (Te), and is limited to values between ($1 \leq Te \leq 30$). It is assumed that the arrival of 164 instruction is through (645 cycles), the number of channels (C = 2), and the value of service rate (μ = 5M Hz = 5.00E + 06 cycles / second).

Based on that, the efficiency measurements of performance can be obtained for a constant service rate queuing (k). This is explained using figures (3) to (9):

1. Estimating the arrival rates for each queue:

$$\lambda_1 = \frac{\mu \times sum(\lambda_1)}{\substack{latest\ arrival \\ time\ in\ cycle}} \frac{cycles}{second} = \frac{(5.00E+06)\times 247}{645} = (1.91E+06)\frac{cycles}{second}$$

$$\lambda_2 = \frac{\mu \times sum(\lambda_2)}{\substack{latest\ arrival \\ time\ in\ cycle}} \frac{cycles}{second} = \frac{(5.00E+06)\times 1842}{645} = (1.43E+07)\frac{cycles}{second}$$

$$\lambda = \frac{\mu \times sum(\lambda)}{\substack{latest\ arrival \\ time\ in\ cycle}} \frac{cycles}{second} = \frac{(5.00E+06)\times 2089}{645} = (1.62E+07)\frac{cycles}{second}$$

$$\because\ \mu = (5.00E+06)\frac{cycle}{second}\ \ is\ constant, so:$$

$$\rho_1 = \frac{\lambda_1}{\mu} = \frac{1.91E+06}{5.00E+06} = 3.83E-01,\ \rho_2 = \frac{\lambda_2}{\mu} = 2.86,\ \rho = \frac{(\rho_1+\rho_2)}{2} = 1.62$$

2. Estimating the value of E {ξ0}:

$$E\{\xi_0\} = \frac{1}{c\mu\left(\rho^{-c}(c-\rho)(c-1)!\ \sum_{n=0}^{c-1}\frac{\rho^n}{n!} + 1\right)} = 7.25E-08$$

3. Expected waiting time in queue (W_q) for a task in queue (k) at the moment t:

$$W_q^{(k)} = \frac{E\{\xi_0\}}{(1-S_{k-1})(1-S_k)}\ ,\quad k = 1,2,\dots,m$$

$$where\quad S_0 \equiv 0\quad and\quad S_k = \sum_{i=1}^{k}\frac{\lambda_i}{C\mu} < 1\ ,\quad for\ all\ k$$

$$When\ k=1,\quad S_0 = 0,$$

$$S_1 = \frac{\lambda_1}{C\mu} \times \frac{non\ stabile}{rate} = \frac{1.91E+06}{2(5.00E+06)} \times 0.1891 = 3.62E-02$$

$$W_q^{(1)} = \frac{E\{\xi_0\}}{(1-S_0)(1-S_1)} = 7.52E-08\ Second$$

It is possible to multiply the expected value of task waiting time with the service rate (μ) in order to convert the units of measurement from (second) to (cycles / time).

$$or\ \ W_q^{(1)} \times \mu = (7.52E-08)(5.00E+06) = (3.76E-01)\ Cycle\ Time$$

$$When\ k=2,\ S_0 = 0\ \ \&\ \ S_1 = 3.62E-02,\quad S_2 = \left(S_1 + \frac{\lambda_2}{C\mu}\right) \times \frac{non\ stabile}{rate} = 3.62E-02$$

$$W_q^{(2)} = \frac{E\{\xi_0\}}{(1-S_1)(1-S_2)} = 1.04E-07\ Second$$

$$or\ \ W_q^{(2)} \times \mu = (1.04E-07)(5.00E+06) = (5.20E-01)\ Cycle\ Time$$

4. Expected number of tasks that provide service (L_q) in queue (k):

$$L_q^{(k)} = \lambda_k . W_q^{(k)}$$

$$\therefore L_q^{(1)} = \lambda_1 . W_q^{(1)} = (1.91E+06)(7.52E-08) = 1.44E-01\ cycle\ \ \&\ \ L_q^{(2)} = \lambda_2 . W_q^{(2)} = 1.48\ cycle$$

5. Expected number of tasks that provide service in system (L_q):

$$L_q = \lambda . W_q = (1.62E+07)(1.01E-07) = 1.63\ cycle$$

6. Expected waiting time in system (W_s) for a task in queue (k) at the moment t:

$$W_s^{(k)} = W_q^{(k)} + E_k\{t\} = W_q^{(k)} + \left(\frac{Expected\ service}{time\ per\ customer}\right) = W_q^{(k)} + \frac{1}{\mu}$$

$$\therefore W_s^{(1)} = W_q^{(1)} + \frac{1}{\mu} = 7.52E-08 + \frac{1}{5.00E+06} = 2.75E-07\ Second$$

It is possible to multiply the expected value of waiting time for a task with the service rate (μ) in order to convert the units of measurement from (second) to (cycles / time).

$or \quad W_s^{(1)} \times \mu = 1.38 \; Cycle \; Time$

$\& \; W_s^{(2)} = W_q^{(2)} + \frac{1}{\mu} = 1.04E-07 + \frac{1}{5.00E+06} = 3.04E-07 \; Second, \quad or \quad W_s^{(2)} \times \mu = 1.52 \; Cycle \; Time$

7. Expected time of a task in the system at the moment t:

$$W_s = \sum_{k=1}^{m} \frac{\lambda_k}{\lambda} . W_s^{(k)} = \frac{\lambda_1}{\lambda} . W_s^{(1)} + \frac{\lambda_2}{\lambda} . W_s^{(2)}$$

$$= \frac{1.91E+06}{1.62E+07} . (2.75E-07) + \frac{1.43E+07}{1.62E+07} . (3.04E-07) = 3.01E-07 \; Second$$

$$or \; Ws(t) \times \mu = (3.01E-07)(5.00E+06) = 1.5 \; Cycle \; Time$$

8. Expected value of number of tasks in the system, which comes from queue (k):

$$L_s^{(k)} = \lambda . W_s^{(k)}$$

$$\therefore \; L_s^{(1)} = \lambda . W_s^{(1)} = 4.46 \; cycle \; , \& \; L_s^{(2)} = \lambda . W_s^{(2)} = (1.62E+07)(3.04E-07) = 4.92 \; cycle$$

9. Expected value of the number of tasks in the system:

$$E \begin{bmatrix} number \\ of \; jobs \; in \\ system \end{bmatrix} = L_s = \lambda . W_s = 4.87 \; cycle$$

Figures from (3) to (9) are describing the above result.

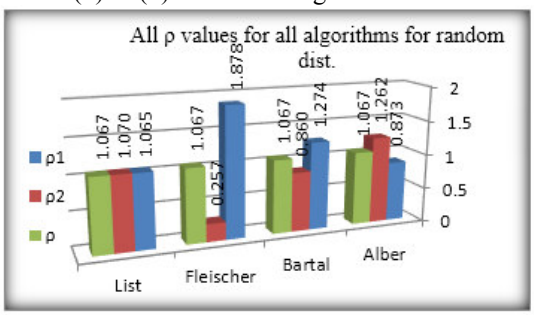


Figure 3: Traffic density parameters for all algorithms (random distribution)

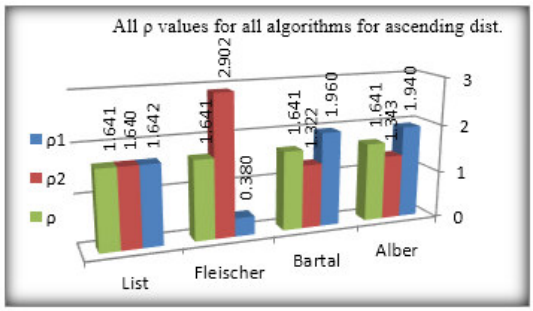


Figure 4: Traffic density parameters for all algorithms (ascending distribution)

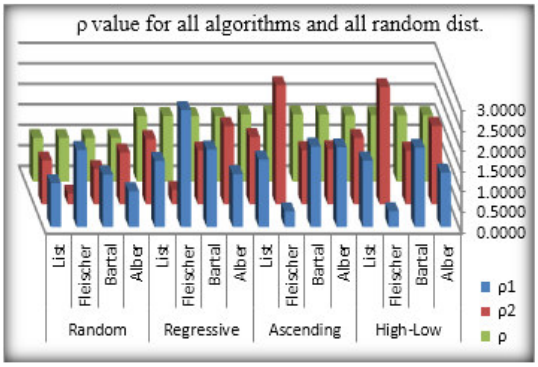


Figure 5: Behavior of (ρ) for all algorithms combined with all dist. (Te)

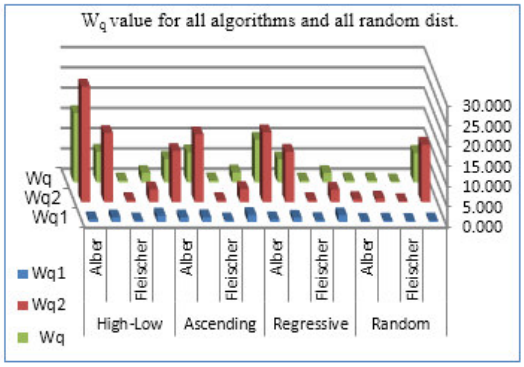


Figure 6: Behavior of W_q for all algorithms and distributions Te combined together

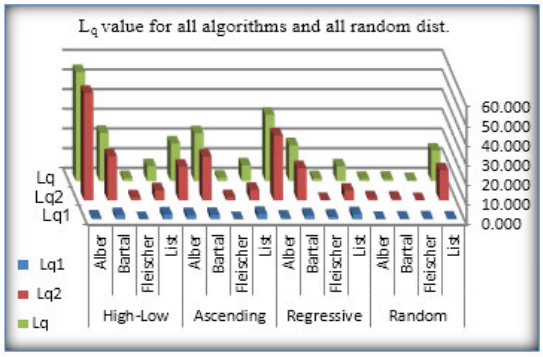


Figure 7: Behavior of L_q for all algorithms and distributions Te combined together

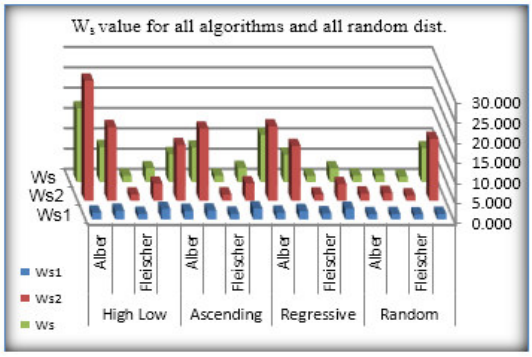


Figure 8: Behavior of W_s for all algorithms and distributions Te combined together

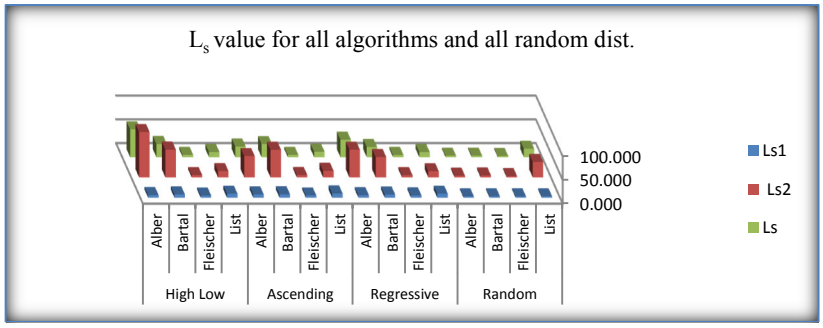


Figure 9: Behavior of L_s for all algorithms and distributions Te combined together

10. Conclusion

The following conclusions are expected to summarize the idea of this project:

1. The simulation techniques provide an acceptable methodology for the research of complex queuing systems.
2. This research shows how to use the simulation with a queuing priority models like $(M_i/M/C):(NPRP/\infty/\infty)$ to evaluate the different multi-processor task scheduling algorithms behavior and efficiency.
3. Different processor cycle's numbers were used based on generating default values. This is done using three different values for cycle's number (5, 10 and 15 cycles). Four algorithms have been applied (Albers, Bartal, Fleischer and Graham's List), and it has been assumed arriving (164 required jobs) to the available processors.
4. The overall traffic density parameter (ρ) has fixed values, which means that changing the algorithm type does not affect the processor's behavior, the distribution type, and the number of cycles.
5. All the algorithms that provide an improvement over the List algorithm are based on the idea of maintaining an imbalance of load among the machines.
6. The relationship between parameters under study with the number of cycles for all algorithms is presented in table (1).
7. From table (1), it is clear that Fleischer algorithm is the optimal algorithm to schedule tasks arriving to a multi-processor. It can reduce the total execution time.
8. The alternative algorithms that can be used are Albers and Bartal, while the Graham's List algorithm is the worst one.

Table (1): Relationship between parameters under study

Parameter	List	Bartal	Albers	Fleischer
ρ_1 ρ_2 ρ	With an inverse relationship with the number of cycles	With an inverse relationship with the number of cycles	With an inverse relationship with the number of cycles	With an inverse relationship with the number of cycles
W_{q1} W_{q2} W_q	Approach for Bartal Positive relationship Positive relationship	The highest anomalous value	Inverse relationship Approach for Bartal Approach for Bartal	Inverse relationship Inverse relationship Inverse relationship
L_{q1} L_{q2} L_q	The highest anomalous value	The highest anomalous value	Inverse relationship Positive relationship The highest anom.v.	Inverse relationship Inverse relationship Inverse relationship
W_{s1} W_{s2} W_s	The highest anom.v. Positive relationship Positive relationship	The highest anomalous value	Inverse relationship The highest anom.v The highest anom.v	Inverse relationship Inverse relationship Inverse relationship
L_{s1} L_{s2} L_s	The highest anomalous value	The highest anomalous value	Inverse relationship Approach for Bartal Approach for Fleischer	Inverse relationship Inverse relationship Inverse relationship

References

A. Ivo, and J. Resing, Queuing Theory, Department of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands, February 14, 2001.

A. Susanne, and S. Bianca, "An Experimental Study of Online Scheduling Algorithms", 2000, available for download at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.8580&rep=rep1&type=pdf .

A.T. Hamdy, Operations Research: An Introduction, by Pearson Education, Inc., 8th edition, ch. 15, 2007.

F. Taher, and K. Al-Salami, "The simulated open and closed queuing network systems", The National Burean for Research and Development, Journal of basic and applied sciences, Vol.16, Issue No.2, 2006, pp.86.

H. Alhussian, N. Zakaria; and F. Azmadi, An Efficient Real-Time Multiprocessor Scheduling Algorithm, Journal of Convergence Information Technology, January 2014.

J. Pirolo, and others, Utilization of Discrete Event Simulation in the Prospective Determination of Optimal Cardiovascular Lab Processes, Proceedings of the 2009 Winter Simulation Conference M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds., 2009.

K.H. Al-Salami, The use of operations research techniques to improve queues scheduling efficiency to enhance computer processing speed, Ph.D. thesis, Omdurman Islamic University, 2013.

M. Arashpour, R. Wakefield, N. Blismas; and B. Abbasi, Quantitative Analysis of Rate-Driven and Due Date–Driven Construction: Production Efficiency, Supervision, and Controllability in Residential Projects, Journal of Construction Engineering and Management, July 2015.

M.I Al-Adue, Microprocessors Intel80186 – Intel8086 – Z80 – Intel8085 – Pentium – Intel80486 – Intel80386 – Intel80286, International House Cultural Investments, 1st edition, Egypt, 2000.

M. L. Lawrence, Handbooks in Operations Research and Management Science, Vol. 13, Simulation, North-Holland is an imprint of Elsevier, 1st edition, 2006.

P. Goran, P. Nikola, and M. Zoran, "Application of the Markov Theory to Queuing Networks", UDC 519.217: 519.61, FACTA University, Mechanical Engineering Vol.6, No.1, 2008, pp. 45 – 56.

R. Al-Ekram, "Online Multiprocessor Scheduling", 2004, available for down-load at http://arnetminer.org/dev.do?m=downloadpdf&url=http://arnetminer.org/pdf/PDFFiles2.

R. Jorge, and R. Vernon, "Efficient implementation of multiprocessor scheduling algorithms on a simulation tested", Software-Practice and Experience, 35:27–50, published online 18 October 2004 in Wiley Inter Science, 2005, available for download at (www.interscience.wiley.com).

S. Yashkov, Queuing Systems: Processor showing queues - some progress in analysis, 1989.
http://citeseerx.ist.psu.edu/showciting;jsessionid=AC9343553AC92C658F26EF1716EA8133?cid=5090576

W. Andreas, A Short Introduction to Queuing Theory, Technical University Berlin, Telecommunication Networks Group, July 21, 1999.

W. Winston, Operations Research Applications and Algorithms, 4th edition, ch. 21, 2003.

Z.M. Al-Kathe, The basic concepts in operating systems, Arab society for publication and distribution library, 1st edition, 2004.

First Author – Kusay H. Al-Salami, B.Sc. in Computer & Statistics (1994), M.Sc. Operations Research (1998), Ph.D. Statistics (2013). 12 years in IT Dep. – Garyounis Uni. - Benghazi - Libya, 1 year in Baghdad College of Economic Sci. Uni.– Computer Eng. Dep.- Baghdad - Iraq, currently working in Business Adm. Dep. - Cihan Uni.- Erbil - Iraq since 2013. I have 11 published papers in differences journals. One book in the printing stage entitled "Quantitative Methods for financial and Banking Sciences".

Second Author – Zaid Taha Sawadi, B.Sc. in Management Information Systems from the University of Mosul/Iraq (2010). M.Sc. in Information Technology Management from the University of Technology/ Malaysia (2013). zaid.sawadi@gmail.com