U.P.B. Sci. Bull., Series C, Vol. 85, Iss. 3, 2023 ISSN 2286-3540

COMPARATIVE PERFORMANCE ANALYSIS OF FCFS, JOHNSON SEQUENCING AND DYNAMIC MAX-MIN JOHNSON SEQUENCING (DMMJS) ALGORITHM USING THREE SERVERS IN CLOUD COMPUTING ENVIRONMENT

Pallab BANERJEE[1], Sharmistha ROY[2]

This paper presents Dynamic Max-Min Johnson scheduling algorithm that offers the optimal sequence. Johnson's rule is perhaps the most classical algorithm in the scheduling field. Dynamic Max-Min Johnson scheduling algorithm gives the most excellent way to the system flow shop to reduce the makespan in polynomial time. The sequencing problem offers with determining a most useful series of acting a number of tasks via a finite number of machines in line with some pre-assigned order with a view to optimize the output. The goal is to decide on the most desirable order of performing the jobs in such a manner that the full elapsed time may be minimal.

Keywords: Cloud computing, Job scheduling, FCFS, Johnson scheduling, Dynamic Max-Min Johnson Sequencing algorithm.

1. Introduction

In today’s world Cloud computing is very wildly used architecture. In simple words, Cloud computing means permitting dynamic allocation of resources, strolling applications using data centers and networks to remote customers [1]. By using Cloud computing, it will help customers in many ways like, ability at runtime without buying new infrastructure, it will also help different business, cooperation and industry in storing their data properly. Currently many company [2] are providing cloud computing services like Google, Microsoft etc. Johnson sequencing rule basically helps in scheduling of different jobs. The primary motive of Johnson rule is to decrease the total idle time of machines & decrease the total time taken to finish all the different jobs. As there aren't any priority rules due to the fact all activity has equal priority, sequencing the jobs in line with the time taken may additionally decrease the total idle time taken by the different jobs [3]. DMMJS algorithm is widely adopted in complex engineering task and Scheduling problem to optimize Cost efficiency and productivity. In this paper, we are providing an algorithm which can adequately increase the availability of resource when demand of resource increases in parallel

[1] Ph.D Scholar, Faculty of Computing and Information Technology, Usha Martin University, Ranchi, India, e-mail: p.banerjee@umu.ac.in

[2] Associate Professor, Faculty of Computing and Information Technology, Usha Martin University, Ranchi, India, e-mail: sharmistha@umu.ac.in

processing of cloud environment. The second major contribution is to reduce the overhead of resource management in equally distributed data center to multiple tenant user infrastructures.

This research paper is organized as follows. Section 1 describes Introduction, Section-2 highlight major literature gap, Section-3 proposes major algorithm formulation of proposed Literary Review, Section -4 scatters the results with future scope and conclusion.

Nomenclatures	
λ	Average arrival rate of task
τ	Time between two successive arrivals of task
S_i	Service time of ith task
μ	Service rate
P_0	Probability that system is idle
L_q	In a queue mean number of tasks waiting
L_s	In a system mean number of tasks waiting
W_q	In a queue mean waiting time of tasks
W_s	In a system mean waiting time of tasks
Abbreviations	
CSP	Cloud Service Provider
SC	Service Centre
FCFS	First Come First Serve
RR	Round Robin
JS	Johnson Sequencing
DMMJS	Dynamic Max-Min Johnson Sequencing
DCC	Data Centre Controller

2. Literature review of the related work

The main research problem is the jobs allocation in cloud in the arena of cloud computing environment. The main issue is to minimize the waiting time while applying Task allocations algorithms and to gain maximum profit. The principal design is to decrease the execution time of the undertaking by utilizing a different task allocation algorithm [4]. There are various sorts of task allocation algorithm present in cloud climate which can't be applied to cloud environments, since cloud is a circled environment that contains heterogeneous structures. Some fundamental algorithms that are used in task allocations are FCFS algorithm in which the task which comes first will execute first. However, the main problem is the amount of time tasks spend waiting in the queue. A new algorithm, called Johnson Sequencing, was then proposed [5]. It fixed the problem with FCFS, which was the average amount of time tasks spent waiting, and it was followed by the Dynamic Max-Min Johnson Sequencing algorithm, which will further reduce task waiting times.

3. System design of Johnson Sequencing

This part deals with scheduling and queuing model as shown in fig.1.In scheduling phase Johnson Algorithm is applied to schedule at Task and M/M/c/K queuing is executed to find holding time. Here *n* number of task in the system. JS Calculation has been viewed as giving an ideal succession of occupations [6]. Models for lining frameworks like M/M/c/K are considered into a record to ascertain AWT. There is a large number of clients who submit requests to the cloud broker, and the requested resource may be infrastructure-, resource-, platform-, storage-, or software-based [7]. Cloud broker detects and has management skills as an intermediary service. The monitor records all user requests, jobs, and resource information for a predetermined amount of time [8]. Analyzer then determines the resources that are available. The required resources are provided in accordance with the SLA service terms and conditions, depending on their availability [9]. When the scheduler module receives information about the available resources, it schedules the tasks using the Johnson Algorithm, which determines the best order and minimizes the make span, hence reducing client wait times. The M/M/c/K system allows for the passing of scheduled jobs, which results in a variety of waiting periods that will be covered later . The sharing of the system may be maximized by efficient server use. By using the servers effectively, the system's sharing potential may be maximized [10].

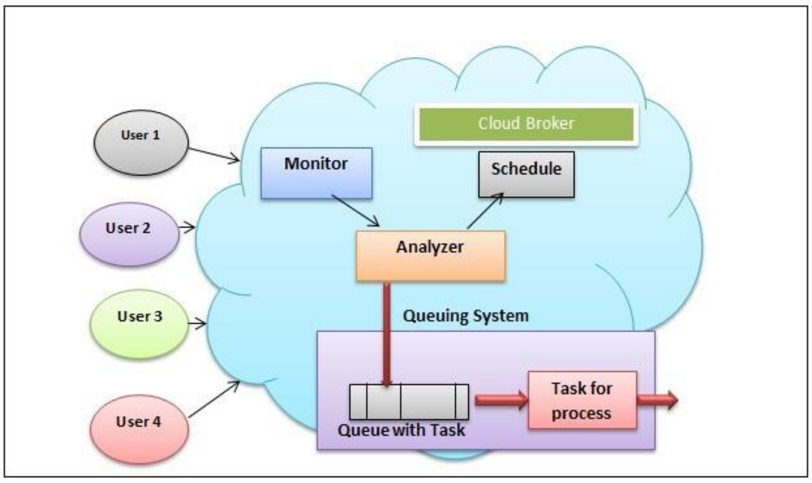


Fig. 1. System design of M/M/c/K Queuing system.

4. Flow chart of FCFS Algorithm

As seen in Fig. 2, FCFS operates under the tenet of "First Come, First Serve." The first task which comes first will be executed first. A non-primitive scheduling technique is used here [11]. In this diagram, the client requests are handled by the DCC, who then places them all in a FIFO queue and routes them to the FCFS virtual machines. The client request is carried out if there are available virtual machines.

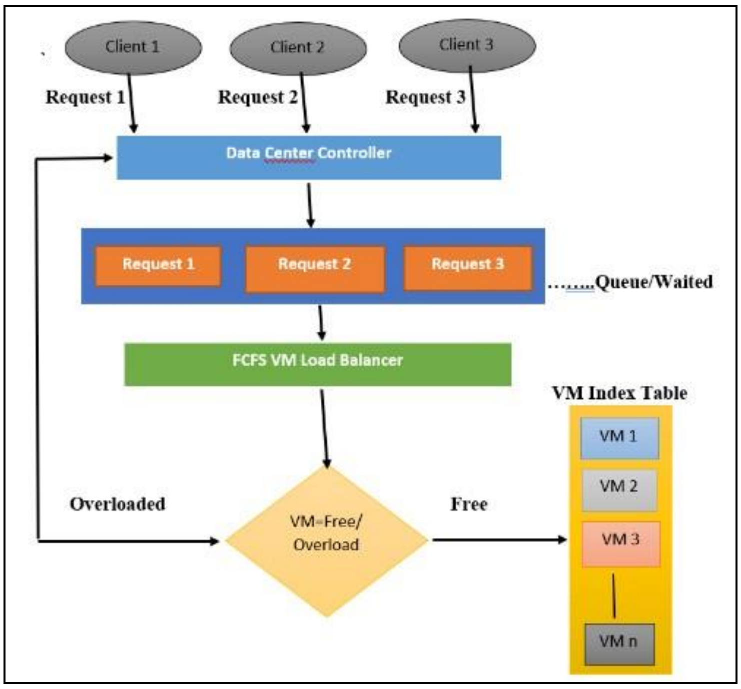


Fig.2.Flow Chart of FCFS Algorithm

5. Lining model for distributed computing climate

The study of the wonder of the holding line is known as queuing theory. By demonstrating entrance preparation, the service process, the number of servers, and the locations with the highest capacity, the fundamental queuing process is completely illustrated [12]. Recognizing that each work's method time should be seen as an exponential transmission It is possible to build an M/M/c/K lined demonstration with non-preemptive demonstration using two servers, S1 and S2, and five locations of holding up positions as capacity [13]. In this study, the queuing model and work scheduling are combined. The fact that page hits start at time zero has been taken into account. We require a rough estimate of "" in order to demonstrate this distribution. Considering that the interval between two subsequent arrivals is "," Therefore, we presume that "" = interval-arrival time.

So average arrival rate

$\lambda = \frac{1}{E[\tau]}$ where E[τ] =Average inter-Arrival Time. (1)

Average Service time E(S)= $\frac{\sum_{i=0}^{n} Si}{n}$ (2)

and Service rate will be $\mu = \frac{1}{E(S)}$ (3)

Probability that the system is idle

$$P_0 = \left[\sum_{n=0}^{S-1} \frac{1}{n!} \left(\frac{\lambda}{\mu}\right)^n + \frac{1}{S!} \left(\frac{\lambda}{\mu}\right)^S \left(\frac{S\mu}{S\mu - \lambda}\right) \right]^{-1} \quad (4)$$

Average number of tasks waiting in the queue

$$L_q = \left[\frac{1}{(S-1)!} * \left(\frac{\lambda}{\mu}\right)^S * \left(\frac{\lambda\mu}{(S\mu - \lambda)^2}\right)\right] * P_0 \quad (5)$$

Average number of tasks in the system

$$L_s = L_q + \frac{\lambda}{\mu} \quad (6)$$

Average waiting time of tasks in queue

$$W_q = \frac{Lq}{\lambda} \quad (7)$$

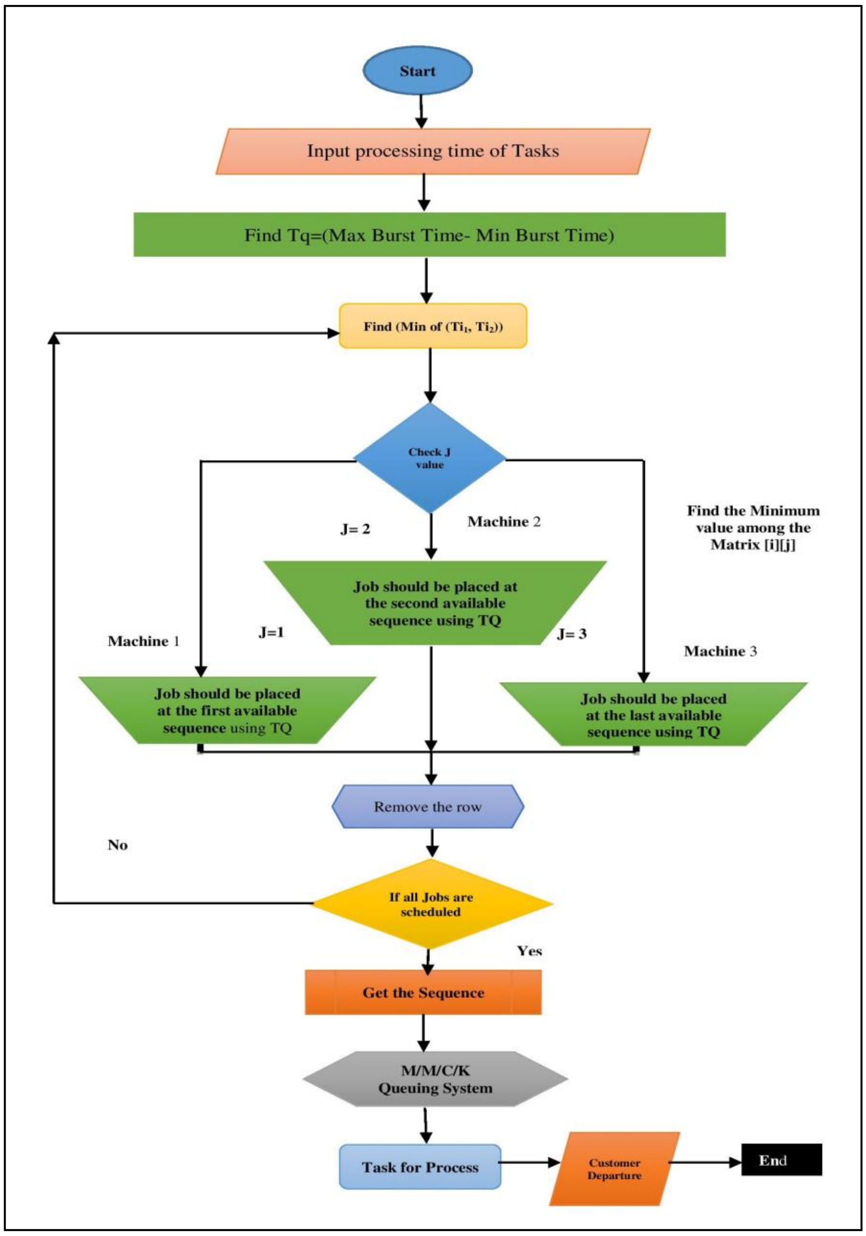
Average waiting time of tasks in the system

$$W_s = W_q + \frac{1}{\mu} \quad (8)$$

6. Objective of the study

Here, a system has been created where Dynamic Max-Min Johnson Sequencing has been used using three servers in the cloud to decrease the service time in the cloud. Let's suppose a batch has a variety of jobs. Creating a Gantt chart will help in calculating the service time. The total execution time of each task is shown in the Gantt chart. After the use of the Dynamic Max-Min Johnson Sequencing Rule in the system, the common range of clients within the queue and inside the machine could be decreased, in addition to the average waiting time inside the device and in the queue.

7. Dynamic Max-Min Johnson Sequencing Algorithm Flowchart


Start
Input processing time of Tasks
Find Tq=(Max Burst Time- Min Burst Time)
Find (Min of (Ti1, Ti2))
Check J value
J= 2
Machine 2
Find the Minimum value among the Matrix [i][j]
Job should be placed at the second available sequence using TQ
Machine 1
J=1
J= 3
Machine 3
Job should be placed at the first available sequence using TQ
Job should be placed at the last available sequence using TQ
Remove the row
No
If all Jobs are scheduled
Yes
Get the Sequence
M/M/C/K Queuing System
Task for Process
Customer Departure
End

8. System flow

In the system design, there are total 4 jobs and every job have a processing time which are available in a table shown below. A, B, C are the Machines and P1,P2,P3,P4 are the number of jobs.

Table 1

Processing time Matrix of Three Machines

Task or Jobs	**Machine A(MC_A)**	**Machine B(MC_B)**	**Machine C(MC_C)**
P1	AA_1	BB_1	CC_1
P2	AA_2	BB_2	CC_2
P3	AA_3	BB_3	CC_3
P4	AA_4	BB_4	CC_4

8.1 Converting 3 machines to 2 machines

Step 1: Find the Minimum of (AAi) , Maximum of (BBi) and Minimum of(CCi)

Step 2: Now verify the following condition. Minimum of (AAi) >= Maximum of (BBi) and Minimum of (CCi) >= Maximum of(BBi).If at least one condition is satisfied then we can convert 3 machines into 2 machines. If none of the condition are satisfied then the method cannot be applied.

Step 3: If any of the above condition is satisfied then, Converting MA, MB and MC into new machine GG and HH. For calculating Machine G add the processing time of (MCA) and (MCB) for each job. That is Machine [14].

$$GG = AAi + BBi$$

For calculating Machine H add the processing time of (MCB) and (MCC) for each job. That is Machine HH = CCi + BBi

Step 4: Now apply Johnson's sequencing Algorithm for n-jobs and 2-machines to determine the optimal sequence. Conversion of 3 Machine to 2 Machine is shown in table 2**.**

Table 2

Executing time Matrix

Job	**(MC_A)**	**(MC_B)**
P1	GG_1	HH_1
P2	GG_2	HH_2
P3	GG_3	HH_3
P4	GG_4	HH_4

8.2 Johnson's sequencing Algorithm for 2 machines.

Step 1: Check the GGi's and HHi's for i = 1, 2…..n and then calculate the minimum of [GGi , HHi]

Step 2: If the minimum be GGk for some i = k, do the kth job first of all. And, if this minimum be HHk for some i = r, do the rth job last of all [15].

Step 3: If there is a equal value for the minimum GGk = HHk , process the kth job first of all and the rth job last of all. If there is an equal value for the minimum occurs among the GGi's select the job corresponding to the minimum of HHi's and process it first of all. If the equal for min occurs among the HHi's take the job corresponding to the minof GGi's and process it last of all [16].

Step 4: Then, mark the jobs which is already done and then repeat from steps one to step three arranging the jobs next to first or next to last, until all the tasks will be completed [17].

Table 3

Initial Processing Time of different task

Tasks	**Processing Time of Server Machine1 (in Millisecond)**	**Processing Time of Server Machine2 (in Millisecond)**	**Processing Time of Server Machine3 (in Millisecond)**	**Burst Time**
P1	0.04	0.02	0.09	0.15
P2	0.03	0.02	0.09	0.14
P3	0.09	0.05	0.10	0.24
P4	0.02	0.01	0.06	0.09

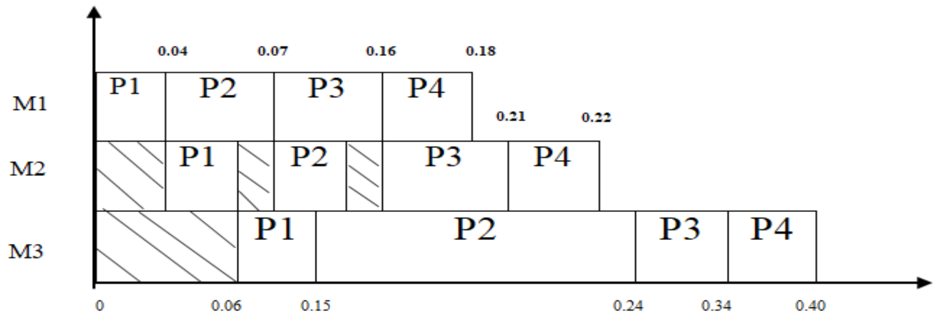


Fig.4. Gantt Chart of FCFS Algorithm

Now, calculating the service time of every process with the help of above Gantt chart.

Job P1: 0.15 - 0 = 0.15

Job P2: 0.24 - 0.04 = 0.20

Job P3: 0.34 - 0.07 = 0.27

Job P4: 0.40 - 0.16 = 0.24

Mean service time = E(s) = 0.86/4=0.215

Now, calculating the service rate " μ "= 1/(E(s)) = 1/0.215=4.6511

Po (for FCFS) = 0.30264

Table 4

In-Out time of Three Machines of different tasks in Johnson Sequencing Algorithm

Tasks	**Machine1**		**Machine2**		**Machine3**	
	IN TIME	**OUT TIME**	**IN TIME**	**OUT TIME**	**IN TIME**	**OUT TIME**
P4	0	0.02	0.02	0.03	0.03	0.09
P2	0.02	0.05	0.05	0.07	0.09	0.18
P1	0.05	0.09	0.09	0.11	0.18	0.27
P3	0.09	0.18	0.18	0.23	0.27	0.37

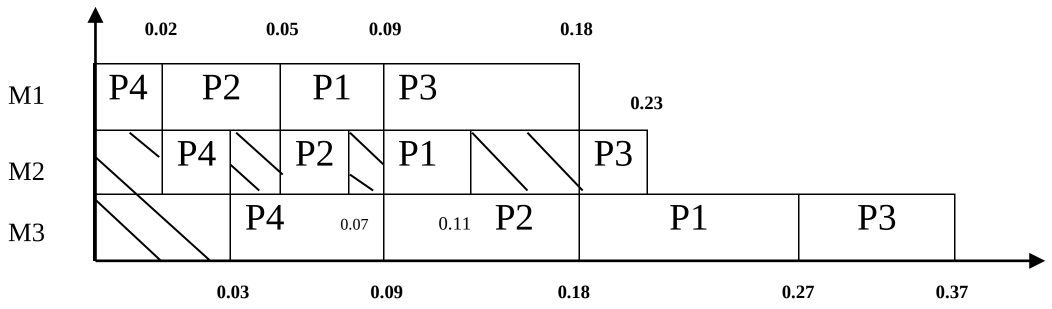


Fig.5. Gantt Chart of Johnson Sequencing Algorithm

Now, calculating the service time of every process with the help of above Gantt chart.

Job P4: 0.09 - 0 = 0.09

Job P2: 0.18 - 0.02 = 0.16

Job P1: 0.27 - 0.05 = 0.22

Job P3: 0.37 - 0.09 = 0.28

Mean service time = E(s)= 0.75/4=0.1875

Now, calculating the service rate " μ "= 1/(E(s)) = 1/0.1875=5.3333

Po (for JS) = 0.470592

For Dynamic Johnsons sequencing, Time quantum will be calculated first.

Now, calculating the Time quantum

TQ = MAX(BT) – MIN(BT)

TQ = 0.24 – 0.09 = 0.15

Now, using the time quantum to create Gantt chart

Table 5

In-Out time of Three Machines of different tasks of Dynamic MaxMin Johnson Sequencing Algorithm

Tasks	Machine1		Machine2		Machine3	
	IN TIME	OUT TIME	IN TIME	OUT TIME	IN TIME	OUT TIME
P4	0	0.09	0.09	0.62	Idle	Idle
P2	0.09	0.23	Idle	Idle	Idle	Idle
P1	0.23	0.38	Idle	Idle	Idle	Idle
P3	0.38	0.53	Idle	Idle	Idle	Idle

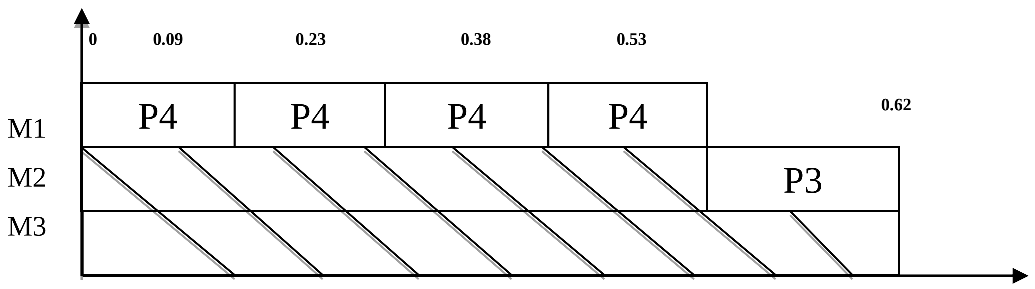


Fig.6. Gantt chart of dynamic max-min Johnson sequencing algorithm

Now, calculating the service time of every process with the help of above Gantt chart.

Job P4: 0.09 - 0 = 0.09

Job P2: 0.23 - 0.09 = 0.14

Job P1: 0.38 - 0.23 = 0.15

Job P3: 0.62 - 0.38 = 0.24

Mean service time = E(s) = 0.62/4=0.155

Now, calculating the service rate " μ "= 1/(E(s)) = 1/0.155=6.451612

Po (for DMMJS) = 0.53698

9. Results

Tables 6, 7, and 8 have each been connected with the findings from the associated formulae. Here, the DMMJS Algorithm reduces service time and average waiting time when compared to the FCFS and Johnson Sequencing. Taking into account the results from formulae (1), (2), (3), (4), (5), (6) (7) and (8). The connection is between Lq, Ls, Wq, and Ws. Fig.7 through 10 show the average number of jobs and average length of the line. A Dynamic Max-Min Johnson sequencing algorithm can reduce it in the system instead of FCFS and the

Johnson sequencing technique. First, we compute the average arrival rate (λ), and then we figure out the average service time (E), which makes the service rate equal to $\mu = \frac{1}{E(S)}$.. The probability that the system is idle (P_0) will then be determined using the formula (4). Using the value of (P_0) from the formula (5), the average number of tasks waiting in the queue (Lq) is then determined. Following that, the system's average number of tasks (Ls) is determined using the preceding Lq findings from the formula (6). Following that, the formula (7) is used to compute the average waiting time for tasks in the queue, and finally, from the formula (9) is used to determine the average waiting time for tasks in the system (8).

Table 6

Results by using FCFS

	(Lq) in milliseconds	**(Ls)in milliseconds**	**(Wq) in milliseconds**	**(Ws) in milliseconds**
λ=2	.00078307	.43078307	.000391535	.2153915
λ=3	.009428535	.654434035	.0024550175	.21745575
λ=4	.018074	.878085	.0045185	.21952

Table 7

Results by using Johnson Sequencing

	(Lq) in milliseconds	**(Ls)in milliseconds**	**(Wq) in milliseconds**	**(Ws) in milliseconds**
λ=2	.0006749	.3756749	.00033745	.187849
λ=3	.00769375	.57019375	.0020078	.189519425
λ=4	.0147126	.7647126	.00367815	.19118985

Table 8

Results by using Dynamic Max-Min Johnson Sequencing(DMMJS)

	(Lq) in milliseconds	**(Ls)in milliseconds**	**(Wq) in milliseconds**	**(Ws) in milliseconds**
λ=2	.000136	.310360	.000068	.155068
λ=3	.003568	.46868	.000909	.155909
λ=4	.007	.627	.00175	.15675

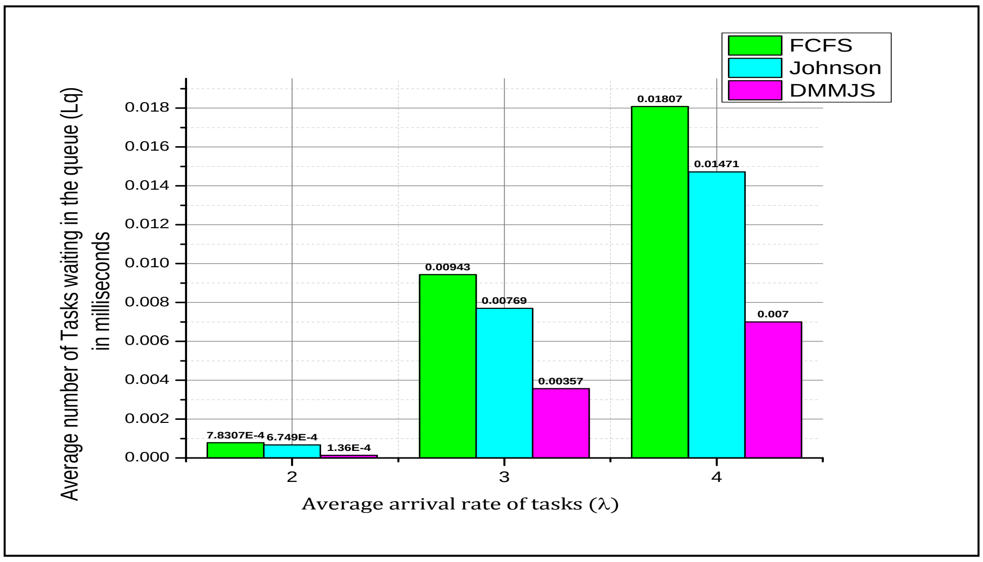


Fig.7. L_q versus Arrival rate of tasks

In this fig.7 the results of the average number of tasks waiting in the queue is shown. Here we can see that a smaller number of tasks are waiting in Dynamic Max-Min Johnson sequencing as compared to Johnson Sequencing and FCFS task scheduling.

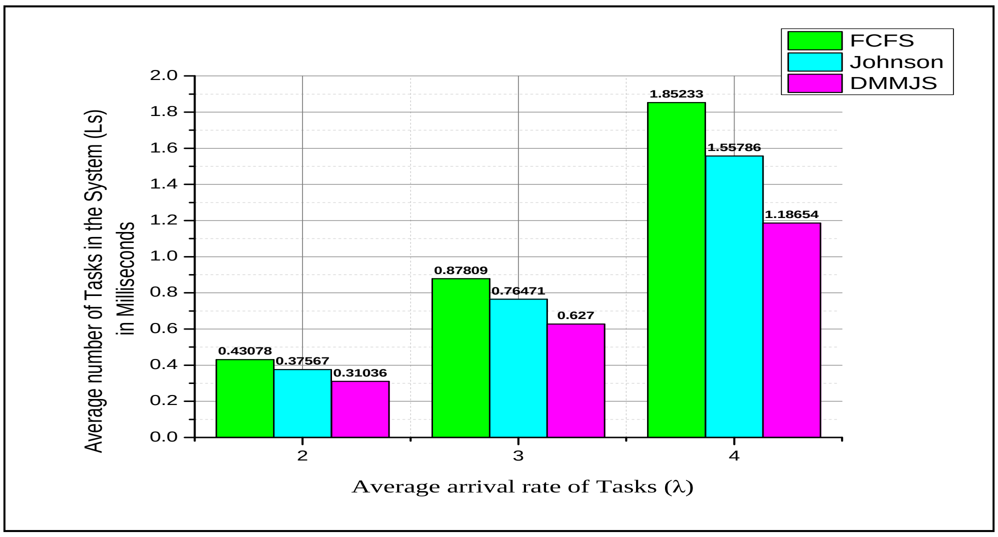


Fig.8. Ls versus Arrival rate of tasks

In Fig.8 and Fig.9, the results of average number of tasks in the queue and system is shown. Here we can see that a smaller number of tasks are waiting in Dynamic Max-Min Johnson sequencing as compared to Johnson Sequencing and FCFS task scheduling.

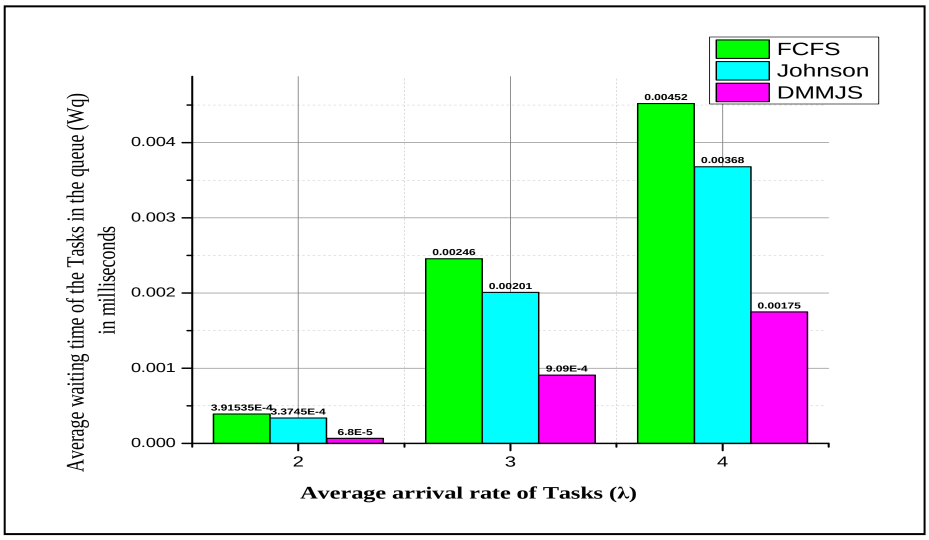


Fig. 9. Wq versus Arrival rate of tasks

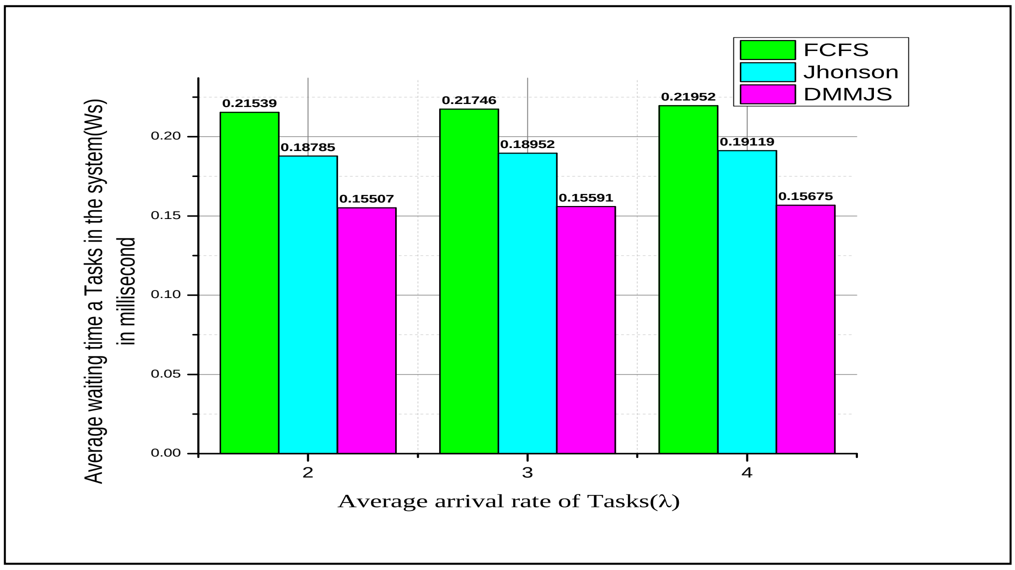


Fig.10. W_S versus Arrival rate of tasks

In this Fig.10 the results of average waiting time of tasks in the system is shown. Here we can see that a smaller number of tasks are waiting in Dynamic Max-Min Johnson sequencing as compared to Johnson Sequencing and FCFS task scheduling.

10. Conclusions

Cloud computing is a term that is used often in research and academia nowadays. Utilizing virtualized computer resources, cloud providers distribute them in accordance with user needs. In this work, we've covered queuing models with many servers and finite capacities as well as scheduling techniques. The Dynamic Max-Min Johnson algorithm was easily created in a suitable setting. Here, we've demonstrated how to obtain a highly valuable sequence using the Dynamic Johnson Sequencing set of rules.

REFERENCES

[1] *Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021).* Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm and Evolutionary Computation, 62, 100841.

[2] *Singh, R. M., Paul, S., & Kumar, A. (2014).* Task scheduling in cloud computing. International Journal of Computer Science and Information Technologies, 5(6), 7940-7944.

[3] *K. Ye, X. Jiang, D. Ye, and D. Huang, "*Two Optimization Mechanisms to Improve the Isolation Property of Server Consolidation in Virtualized Multi-core Server," in Proceedings of 12th IEEE International Conference on High Performance Computing and Communications, 2010, pp. 281– 288Zhao, W.; and Stankovic, J.A. (1989).

[4] *C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A.Warfield,* "Live migration of virtual machines," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, 2005, pp. 286.

[5] *Pal, S. and Pattnaik, P.K. (2013).* Classification of virtualization environment for cloud computing. Indian Journal of Science and Technology, 6(1), 3965-3971.

[6] *Saidi, K., Bardou, D.* Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities. Cluster Comput (2023). https://doi.org/10.1007/s10586-023-04098-4

[7] *Spicuglia, S. Chen, L.Y.; Binder, W. (2013).* Join the best queue: Reducing performance variability in heterogeneous systems. Sixth International Conference on Cloud Computing (CLOUD 2013), Proceedings IEEE, 139-146.

[8] *Guo, L.; Yan, T. Zhao, S.; and Jiang, C. (2014*). Dynamic performance optimization for cloud computing using M/M/m queuing system. Journal of Applied Mathematics, 2014, 1-8.

[9] *Cheng, C; Li, J and Wang, Y (2015).* An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing. Tsinghua Science and Technology, 20(1), 28-39.

[10] *Kuo, R.J; and Cheng, C. (2013).* Hybrid meta-heuristic algorithm for job shop scheduling with due date time window and release time. The International Journal of Advanced Manufacturing Technology, 67(4), 59- 71.

[11] *Jang, S. H., Kim, T. Y., Kim, J. K., & Lee, J. S. (2012).* The study of genetic algorithm-based task scheduling for cloud computing. International Journal of Control and Automation, 5(4), 157-162.

[12] *Awad, A. I., El-Hefnawy, N. A., & Abdel_kader, H. M. (2015).* Enhanced particle swarm optimization for task scheduling in cloud computing environments. Procedia Computer Science, 65, 920-929.

[13] *Ali, A., Iqbal, M. M., Jamil, H., Qayyum, F., Jabbar, S., Cheikhrouhou, O., ... & Jamil, F. (2021).* An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing. Sensors, 21(13), 4527.

[14] *Shah-Mansouri, H., Wong, V. W., & Schober, R. (2017).* Joint optimal pricing and task scheduling in mobile cloud computing systems. IEEE Transactions on Wireless Communications, 16(8), 5218-5232.

[15] *Banerjee, P., & Roy, S. (2021, October).* An Investigation of Various Task Allocating Mechanism in Cloud. In 2021 5th International Conference on Information Systems and Computer Networks (ISCON) (pp. 1-6). IEEE.

[16] *Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021).* Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm and Evolutionary Computation, 62, 100841.

[17] *Ababneh, J. (2021).* A hybrid approach based on grey wolf and whale optimization algorithms for solving cloud task scheduling problem. Mathematical Problems in Engineering, 2021, 1-14.