Mathematical Problems of Computer Science 62, 43–51, 2024.
doi:10.51408/1963-0119

UDC 519.872

Advanced Queueing Model of a Multiprocessor Computing System

Artur P. Vardanyan

Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia
e-mail: artur.vardanyan@iiap.sci.am

Abstract

This paper presents an advanced queueing model for a multiprocessor computing system, where tasks require a random number of processors and are subject to constraints on waiting times in the queue. Unlike classical multi-server queueing systems, this model accounts for both resource requirements and queue waiting time restrictions, making it more suitable for real-world computing environments. By incorporating the probabilistic behavior of task arrival, service, and waiting constraints, the expressions are derived for key performance metrics, including the probabilities of task rejection and failure, system throughput, and resource utilization. An algorithm for determining the optimal queue length is also developed to enhance system efficiency by minimizing the probability of task losses. The proposed model provides a framework for analyzing and optimizing resource allocation in multiprocessor systems, improving their capability to handle dynamic and complex workloads.
Keywords: Multiprocessor System, Queueing System, Multi-server Queueing System, Waiting Time Restriction, Resource Utilization, System Efficiency, Performance Measures.
Article info: Received 29 September 2024; sent for review 15 October 2024; accepted 21 November 2024.

1. Introduction

In the rapidly evolving field of information technology, distributed and parallel high-speed computing systems have become essential due to advancements in several key areas, including multi-agent intelligent methods for information retrieval, data science for processing and storing vast amounts of data, and scientific research addressing complex challenges[1]. These systems are crucial for scientific and technological modelling, where there is a growing demand for increased accuracy, faster computations, and large-scale calculations.
Multiprocessor systems are often employed to tackle problems requiring large-scale computations that cannot be handled by a single processor. However, this leads to significant challenges in managing the simultaneous execution of multiprocessor tasks while ensuring the efficient utilization of system resources[2]. One of the critical issues in this context is the

development of effective scheduling algorithms for such systems.
Research on efficient scheduling algorithms for multiprocessor systems is a key area in computer science and parallel computing. Modern multiprocessor systems typically consist of heterogeneous processors with varying capabilities and performance characteristics. Additionally, applications are becoming increasingly diverse and dynamic, necessitating adaptive scheduling strategies that can efficiently allocate system resources while optimizing key performance metrics such as system throughput, service latency, and energy efficiency[3, 4].
As the development of scheduling algorithms continues to evolve, there is a growing need to incorporate additional metrics to enhance efficiency[5]. While optimization theory and machine learning tools are often employed to address these challenges, valuable insights can also be gained by analyzing these systems through the framework of queueing theory.

In this study, a multiprocessor system is modeled as a queuing system, where tasks require parallel servicing and constraints on task waiting times in the queue are explicitly considered.

2. Queueing Model

A computing system consisting of m processors (cores, cluster nodes, etc.) is modeled as a queueing system ($m \geq 1$). The system can queue a limited number of tasks, constrained by n waiting slots ($n \geq 1$).
Each task in the system is characterized by three random parameters (ν, β, ω):

- ν represents the number of computing resources (processors, cores, cluster nodes, etc.) required for servicing the task,
- β denotes the maximum time needed to service the task,
- ω is the maximum allowable waiting time for a task in the queue, after which the task leaves the system without being serviced.

The task is either accepted and placed in the service queue or denied service. Service denial may occur if there is no space in the queue or user-defined constraints (e.g., servicing time, waiting time, number of processors) cannot be met. The time required to service a task is partly conditional, meaning it is the maximum allowable value. The actual servicing time is random and may be shorter than the given maximum, allowing the order of service to change as tasks arrive or as services are completed.
The queuing model incorporates the cumulative distribution function of the exponential distribution constraints for task arrivals, servicing, and failures due to waiting time restrictions. These are defined by the intensities: a for the incoming task stream, b for task servicing, and w for task failure within the queue[6]. The model also considers normal distribution constraints for the random variable ν, representing the number of computational resources required to perform a task. The probability distribution is given by:

$$P(\nu = k) = \frac{1}{m}, \quad k = 1, 2, \ldots, m.$$

Tasks are serviced in the order they enter the system, following a FIFO discipline. Tasks that arrive when the queue is full are denied service.
To describe system transitions, the state of the system is defined by the number of tasks i

being serviced and j tasks waiting in the queue. The probability of the system being in this state is denoted by $P_{i,j}$. Since the number of possible states is finite, the system eventually reaches a stable operating mode, known as the steady state[7].
A system of equations was derived to describe the steady-state behavior of the queueing model.
A detailed description of the derivation, analysis, and development of a numerical algorithm for solving this system of equations is presented in [6]. This research provides an efficient and accurate method for computing the $P_{i,j}$ steady-state probabilities for the considered multiprocessor queueing system.

3. Performance Measures

To analyze the performance measures of the considered queueing system, it is first necessary to determine the effective arrival rate and the service rate of the tasks using the given system parameters and distribution functions for the time between arrivals, the service time, and the permissible waiting time. Unlike classical multi-server queueing systems, in this model, each task requires a specific number of service nodes and is subject to a restriction on queue waiting time. The permissible waiting time ω is modeled by an exponential distribution, while the number of required computational resources ν follows a uniform distribution. These considerations significantly impact the system's performance measures, including the probability of task rejection, the probability of queue task failure and the resource utilization factor. However, these considerations do not directly affect the service rate μ, which is based on only service time distribution. Specifically, the service rate μ is given by b in the considered queueing system.
To understand the real impact of these factors on system performance measures, it is first necessary to calculate the probabilities of task access rejection and task failure (i.e., when a task leaves the queue due to exceeding its waiting time) during system operation.
Upon task arrival, the system will reject access if the number of tasks waiting in the queue reaches its maximum capacity, n. In other words, the probability of task access rejection during system operation can be calculated by considering the probabilities of being in all system states where the queue is fully occupied. This probability of task access rejection, denoted by P_r, can be expressed as follows:

$$P_r = \sum_{i=0}^{m} \frac{aP_{i,n}}{a + ib + nw},$$

where a represents the arrival rate of tasks, b represents the service rate, w represents the rate at which tasks fail due to waiting too long in the queue, and $P_{i,n}$ for $i = 0, 1, \ldots, m$ are the steady-state probabilities of the system when i tasks are being serviced and the queue is fully occupied.
It is assumed that each task has a maximum allowable waiting time before being serviced, after which it leaves the system without being served. The probability of task failure during system operation, denoted by P_f, can be calculated using the system state probabilities. The formula to compute P_f is as follows:

$$P_f = \sum_{i=0}^{m} \sum_{j=1}^{n} \frac{jwP_{i,j}}{a + ib + jw},$$

where, as above, a represents the arrival rate of tasks, b represents the service rate, w represents the rate at which tasks fail due to waiting too long in the queue, and $P_{i,j}$ for $i = 0, 1, \ldots, m$ and $j = 0, 1, \ldots, n$ are the steady-state probabilities of the system when i tasks are being serviced and j tasks are waiting in the queue.
By using the above-obtained probabilities, the effective arrival rate λ in the considered queueing system can be calculated as follows:

$$\lambda = aP_a,$$

where P_a is the probability of task abandonment, which represents the likelihood that a task leaves the queue without being serviced, either due to exceeding its permissible waiting time or being rejected by the system because the queue is fully occupied.
Furthermore, P_a is expressed as follows:

$$P_a = (1 - P_r)(1 - P_f).$$

Queuing systems define the system throughput, which is closely related to the effective arrival rate, but they describe different aspects of the system's performance[8]. The system throughput is denoted by X and is defined as:

Definition 1.. *System Throughput is the rate at which tasks are completed and leave the system.*

In a stable system (where the queue doesn't grow indefinitely over time), the system throughput is typically equal to the effective arrival rate. This is because the system can handle the incoming tasks without accumulating an infinite queue, meaning that every task that arrives is eventually processed. For the considered queueing system, the system throughput is the same as the effective arrival rate:

$$X = \lambda.$$

Next, to derive a formula for the utilization factor ρ in the queueing system under consideration, it is first necessary to define the utilization factor in the context of this multiprocessor queueing system. Since, each task requires a random number of computational resources, which affects how much of the system's total resources are utilized on average, the utilization factor will be defined as follows:

Definition 2.. *Utilization is defined as the ratio of the service capacity demand to the total service capacity of the system:*

$$\rho = \frac{Service\ Capacity\ Demand}{Total\ Service\ Capacity}$$

As each task requires a random number of processors, the system's effective utilization factor must consider the expected number of processors required by a task, denoted as $E[\nu]$. Thus, the service capacity demand is determined by $\lambda \times E[\nu]$. This accounts for the fact that each task might occupy multiple number computational resources simultaneously.
For a system with m processors, where each processor has a service rate μ, the total service capacity is $m \times \mu$.
Thus, the refined utilization factor formula becomes:

$$\rho = \frac{\lambda E[\nu]}{m\mu}.$$

Given that the distribution of ν is uniform from 1 to m, the expected value $E[\nu]$ is determined as follows:

$$E[\nu] = \sum_{k=1}^{m} kP(\nu = k) = \frac{1}{m}\sum_{k=1}^{m} k = \frac{m+1}{2}.$$

By substituting the derived values for λ, μ and $\mathbb{E}[\nu]$, a formula is obtained for the utilization ρ:

$$\rho = \frac{a(m+1)P_a}{2mb}.$$

The formula for calculating the utilization factor ρ is refined to be consistent as a measure of system resource usage.

4. Optimal Service Parameters

This section presents the process of determining the optimal configuration for the modeled queueing system based on given parameters and distributions. As discussed in the previous section, for the specified distributions of system parameters, it is possible to evaluate the probabilities of task rejection and task failure while waiting in the queue. This allows for estimating the probability that an incoming task will not be serviced by the system. This probability is a key indicator of the number of tasks unserviced by the system and can be calculated as follows:

$$P_l = P_r + (1 - P_r)P_f,$$

Thus, the question arises as to how this probability can be minimized, which would reduce the number of unserviced tasks by the system. It is evident that the value of P_l depends on the number of processors (m) in the multiprocessor system, the queue length (n), the task arrival rate (a), the service rate (b), and the task failure rate in the queue (w). Considering this, the objective is to minimize P_l by varying the queue length, identifying the queue length at which P_l takes its minimum value for the given system parameters while ensuring that the utilization factor is less than one:

$$\begin{cases} P_l \to \min \\ \rho < 1 \end{cases}$$

It is evident that when there is no queue, the rejection probability P_r has a certain value, while there is no concept of P_f probability(in this case, P_f is assumed to be zero for simplicity). As the queue length increases, the rejection probability P_r tends toward 0, and the failure probability P_f approaches 1, as follows:

$n = 0$	$n \to \infty$
$0 < P_r \leq 1$	$P_r \to 0$
$P_f = 0$	$P_f \to 1$

An algorithm has been developed and implemented in Python, to determine the optimal queue length for a multiprocessor system with a given number of processors, task arrival, service, and failure rates. The algorithm initially sets the queue length to $n = 1$ and calculates the probability P_l. At each subsequent step, the calculation is repeated with the queue length increased by one, and the new P_l value is compared to the previous value. If the new value is smaller, the algorithm continues this cycle. When the P_l value from the previous step is smaller than the current step, the queue length is set to the previous value,

and the system utilization factor is calculated for that queue length. If the utilization factor is less than one, the algorithm terminates, setting the current n as the optimal queue length for the given system parameters. If the utilization factor is greater than or equal to one, the algorithm ends with a message indicating that the given parameters don't support effective system performance. Thus, with this queue length, the system can achieve an optimal service configuration, minimizing the probability P_l of unserviced tasks for the given parameters. The pseudocode of the algorithm consists of two blocks and is presented below:

Algorithm 1 System Performance Measures Computation

Input: n, m, a, b, w
Output: P_r, P_f, ρ

$p \leftarrow \text{SteadyStateProbabilities}(n, m, a, b, w)$
$P_r, P_f \leftarrow 0, 0$
$i, j \leftarrow 0, 0$
for each $x \in p$ **do**
  if $j == n$ **then**
    $P_r \leftarrow P_r + \dfrac{a \cdot x}{a + i \cdot b + j \cdot w}$
  end if
  $P_f \leftarrow P_f + \dfrac{j \cdot w \cdot x}{a + i \cdot b + j \cdot w}$
  $j \leftarrow j + 1$
  if $j == n + 1$ **then**
    $j \leftarrow 0$
    $i \leftarrow i + 1$
  end if
end for
$P_a \leftarrow (1 - P_r) \cdot (1 - P_f)$
$\rho \leftarrow \dfrac{aP_a \cdot (m + 1)}{2 \cdot m \cdot b}$
return P_r, P_f, ρ

Algorithm 2 Optimal Queue Length Computation

Input: m, a, b, w
Output: n_{opt} (the optimal queue length)

$n_{opt} \leftarrow 1, n \leftarrow 1$
$P_r, P_f, \rho_{opt} \leftarrow \text{SystemPerformanceMeasures}(m, n, a, b, w)$
$M_1 \leftarrow P_r + (1 - P_r) \cdot P_f$
$n \leftarrow n + 1$
$P_r, P_f, \rho \leftarrow \text{SystemPerformanceMeasures}(m, n, a, b, w)$
$M_2 \leftarrow P_r + (1 - P_r) \cdot P_f$
while $M_1 > M_2$ **do**
  $n_{opt} \leftarrow n$
  $n \leftarrow n + 1$
  $M_1 \leftarrow M_2$
  $\rho_{opt} \leftarrow \rho$
  $P_r, P_f, \rho \leftarrow \text{SystemPerformanceMeasures}(m, n, a, b, w)$
  $M_2 \leftarrow P_r + (1 - P_r) \cdot P_f$
end while
if $\rho_{opt} \geq 1$ **then**
  Raise "InvalidParameterException: These parameters don't support effective system performance!"
else
  Print "The optimal queue length is n_{opt}"
  return n_{opt}
end if

5. Conclusion

The study presented in this paper offers a refined approach to modeling multiprocessor systems using advanced queueing theory and accounting for variability in task requirements and queue constraints. By analyzing the steady-state probabilities and deriving metrics such as task rejection and failure rates, effective arrival rates, and system utilization, insights into optimizing system performance are gained. The proposed algorithm for finding the optimal queue length minimizes task losses while maintaining efficient resource usage. This work contributes to the field of multiprocessor queueing theory, providing tools for better scheduling and resource allocation strategies in high-performance computing environments. Future work may explore extensions of this model to systems with heterogeneous resources or additional performance constraints, further enhancing the model's applicability in real-world scenarios.

References

[1] L. N. Bhuyan and S.Sinha, "Design of Parallel and Distributed Systems", *Springer, Cham,* pp. 12-250, 2023.

[2] V. Sahakyan and A. Vardanyan, "About the possibility of executing tasks with a waiting time restriction in a multiprocessor system", *AIP Conference Proceedings*, 2757.1, pp. 030003, 2023. DOI: https://doi.org/10.1063/5.0135784.

[3] D.Bertsimas and D.Gamarnik, "Queueing Theory: Classical and Modern Methods", *Dynamic Ideas*, Belmont, pp. 126-586, 2022.

[4] F. P. Kelly, S. Zachary and I.Ziedins, "Stochastic Networks: Theory and Applications", *Clarendon Press*, Oxford, pp. 16-298, 2020.

[5] J. Anamika, J. Madhu, D. Bhardwaj, "Controllable multiprocessor queueing system", *Applications of Mathematical Modeling, Machine Learning, and Intelligent Computing for Industrial Development*, pp. 61-76, 2023.

[6] V. Sahakyan and A. Vardanyan, "A Computational Approach for Evaluating Steady-State Probabilities and Virtual Waiting Time of a Multiprocessor Queuing System", *Programming and Computer Software*, Volume 49, pp. S16S23, 2023. DOI: https://doi.org/10.1134/S0361768823090098.

[7] J. F.Shortle, J. M.Thompson, D. Gross and Harris C. M., *Fundamentals of Queueing Theory*, John Wiley and Sons, New York, pp. 35-475, 2018.

[8] H. Takagi, "Appendix A: Derivation of Formulas by Queueing Theory" *Spectrum Requirement Planning in Wireless Communications*, John Wiley & Sons Ltd, pp. 199-218, 2008. DOI: https://doi.org/10.1002/9780470758946.app1.

Բազմապրոցեսորային հաշվողական համակարգի ընդլայնված հերթերի մոդել

Արթուր Պ. Վարդանյան

ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ, Երևան, Հայաստան
e-mail: artur.vardanyan@iiap.sci.am

Ամփոփում

Այս հոդվածում ներկայացվում է բազմապրոցեսորային հաշվողական համակարգի ընդլայնված հերթերի մոդել, որտեղ առաջադրանքները պահանջում են պատահական քանակով պրոցեսորներ և ենթակա են հերթում սպասման ժամանակի սահմանափակումների: Ի տարբերություն դասական բազմասերվերային հերթերի համակարգերի, այս մոդելը հաշվիէառնում ինչպես ռեսուրսների պահանջները, այնպես էլ հերթում սպասման ժամանակի սահմանափակումները, ինչն առավել հարմար է իրական հաշվողական միջավայրերի համար: Ներառելով առաջադրանքների մուտքի, սպասարկման և սպասման սահմանափակումների հավանականային բնույթը, արտածվել են համակարգի կարևոր կատարողական ցուցանիշների՝ առաջադրանքների մերժման և ձախողման հավանականությունների, համակարգի թողունակության և ռեսուրսների օգտագործման գործակցի, գնահատման արտահայտություններ: Մշակվել է նաև հերթի օպտիմալ երկարությունը որոշող ալգորիթմ, որը բարելավումէհամակարգի արդյունավետությունը՝ նվազեցնելով առաջադրանքների կորստի հավանականությունը:

Բանալի բառեր՝ Բազմապրոցեսորային համակարգ, հերթերի համակարգ, բազմասերվերային հերթերի համակարգ, սպասման ժամանակի սահմանափակում, ռեսուրսների օգտագործում, համակարգի արդյունավետություն, կատարողականության չափեր:

Расширенная модель очередей многопроцессорной вычислительной системы

Артур П. Варданян

Институт проблем информатики и автоматизации НАН РА, Ереван, Армения
e-mail: artur.vardanyan@iiap.sci.am

Аннотация

В данной статье представлена расширенная модель очередей для многопроцессорной вычислительной системы, где задачи требуют случайного количества процессоров и подчиняются ограничениям на время ожидания в очереди. В отличие от классических многосерверных систем очередей, эта модель учитывает как требования к ресурсам, так и ограничения на ожидание в очереди, что делает её более подходящей для реальных вычислительных сред. С использованием вероятностного подхода к моделированию поступления, обслуживания задач и ограничениям на ожидание были получены выражения для ключевых показателей производительности, включая вероятности отказа и сбоя задач, пропускную способность системы и загрузку ресурсов.

Также был разработан алгоритм для определения оптимальной длины очереди, позволяющий повысить эффективность системы за счёт минимизации вероятности потерь задач.

Ключевые слова: Многопроцессорная система, система очередей, многосерверная система очередей, ограничение времени ожидания, использование ресурсов, эффективность системы, показатели производительности.