RESEARCH ARTICLE

Performance Analysis of Energy-Efficient Reliable AIoT System Architectures

Shoma Nishio[1], **Yuta Makino**[1], **Tuan Phung-Duc**[1,*] **and Fumio Machida**[2]

[1] Department of Policy and Planning Sciences, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
[2] Department of Computer Science, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

Abstract

As many IoT systems deploy machine learning models to implement intelligent functions, the reliability and performance assurance of artificial intelligence and the Internet of Things (AIoT) system is becoming a crucial issue. While reliability of AIoT system outputs can be improved by redundancy using multiple input data, the system involves performance and energy overheads that may be unacceptable in real deployment under limited computing resources. To ensure the performance and energy-efficiency of AIoT systems, this paper proposes the queueing models for multi-input AIoT systems in two different architectures, namely the parallel and the shared architectures, and compares them with respect to several performance metrics. Through numerical and simulation studies, we show that the shared architecture has advantages in response time and energy consumption while maintaining the same reliability as the parallel architecture. Furthermore, we show that the proposed models yield more precise performance estimates than analyses based on simple queueing models, which do not capture the comparison process among multiple inference results.

Keywords: AIoT system, energy consumption, performance, queueing model, redundant architecture.

1 Introduction

Machine learning (ML) models have been extensively deployed to IoT systems in various domains. As IoT systems are increasingly empowered by ML, quality assurance for artificial intelligence and Internet of Things (AIoT) systems has become an emerging issue in system engineering. Outputs of ML models in real IoT systems are far from perfect because the data encountered in operation often differs from the training dataset. Therefore, system architects cannot ignore the uncertainty of ML model outputs in IoT devices and must consider the AIoT system's reliability with respect to prediction errors. There has been extensive research on the robustness of ML models to protect them from adversarial examples that are artificially generated samples to fool ML model predictions in the deployed systems [1–3]. In recent years, many ML testing techniques have also been proposed to minimize undesirable errors in ML components before deployment in real systems [4–6]. However, further efforts to improve the reliability of

Submitted: 15 December 2025
Accepted: 25 February 2026
Published: 15 March 2026

Vol. 1, **No.** 1, 2026.
10.62762/JSS.2025.386670

***Corresponding author:**
✉ Tuan Phung-Duc
tuan@sk.tsukuba.ac.jp

Citation
Nishio, S., Makino, Y., Phung-Duc, T., & Machida, F. (2026). Performance Analysis of Energy-Efficient Reliable AIoT System Architectures. *Journal of Systems Scalability*, 1(1), 6–22.

© 2026 by the Authors. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

ML-based system outputs at runtime are still needed for critical applications such as self-driving cars.

A multi-input ML system is an architectural approach that reduces erroneous outputs by combining multiple ML models and diverse inputs. The technique is also referred to as the N-version machine learning system in a recent study [7, 8]. The reliability improvements achieved by multiple versions of the ML architecture are studied analytically [7, 9] and experimentally [10–12]. While existing studies on redundant architecture approaches have focused solely on improving reliability, quite a few works address the performance and energy overhead issues that are not negligible in IoT system deployments. When we introduce multiple ML models to obtain redundant inference results, the processing cost increases, which clearly impacts system performance and energy consumption.

The parallel and shared architectures are two fundamental configurations of multi-input ML systems for improving reliability [13]. Both systems can exploit input data diversity to diversify ML model inferences and produce reliable outputs by comparing two inference results. For example, if two ML models disagree on their inference results, either output is considered an error, and the system can temporarily nullify the output or delegate the issue to a higher-level component. The reliability improvement is equally possible in two different architectures. However, the performance overheads are not equal: the parallel architecture runs two ML modules in parallel, whereas the shared architecture relies on a single shared ML module. The previous study used queueing analysis to show that the parallel architecture has an advantage in system throughput [13], which agrees with intuition, but the architecture's disadvantages were not rigorously investigated.

To fill this gap, this paper proposes comprehensive queueing models to evaluate the performance and energy consumption of parallel and shared architectures for multi-input AIoT systems comprising ML modules. The proposed models can analytically formulate performance metrics for AIoT systems, such as response time, throughput, and energy consumption. With the defined performance metrics, the advantages and disadvantages of the two architectures can be systematically analyzed through numerical and simulation experiments. The experimental results show that the shared architecture outperforms the parallel architecture in terms of expected response time and energy consumption. The performance models for the parallel and the shared architectures are based on a quasi-birth-death process and a GI/M/1-type Markov chain, respectively. The proposed models are more expressive than simple M/M/1 type queues. We also evaluate the approximated performance using the simple M/M/1 type queues and compare the results with the proposed models. The comparison results show that the approximated models can underestimate the performance of the parallel and the shared architectures. Our models provide a more precise evaluation of the AIoT system performance and energy, helping engineers to design a reliable system architecture.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 explains the system configurations of the parallel and the shared architectures. In Section 4, queueing models to analyze the performance of the two architectures are explained. Section 5 gives the definitions of the performance measures. Section 6 shows the numerical examples and simulation results. Section 7 provides a guide to choose the architecture based on our results. Finally, Section 8 describes the conclusions and future work.

2 Related work

While the quality of ML models is primarily evaluated by inference accuracy, reliability and performance are important quality metrics when ML models are deployed in IoT systems. The performance of ML-based systems has been studied in real-world applications such as face recognition on mobile devices [14, 15], object detection in autonomous vehicles [16, 17], and intrusion detection in IoT systems [18]. ML models used in real-time decision-making often have stringent requirements for low latency and high-throughput task execution [19]. An autonomous vehicle is a typical example of a latency-critical application that may encounter critical accidents if the requirement is not met [17]. The latency of ML-based computer vision tasks, such as object detection, can be improved by exploiting high-performance GPUs. However, the associated costs and energy consumption may not be acceptable in small embedded systems. We can also improve the latency of vision tasks by reducing the input image size, but it may compromise accuracy [20]. While many existing studies evaluated the performance of a single ML module, AIoT systems may consist of multiple ML modules with multiple

input data. A theoretical analysis of the performance of multi-input ML systems was presented in the previous study [13]. However, the study evaluated only throughput and did not analyze latency or energy consumption. In this paper, we newly define the latency and energy consumption metrics and compare the two architectures using these metrics.

Queueing models offer a powerful theoretical framework for analyzing a variety of systems, such as telecommunication, computer, and industrial engineering systems. The M/M/1 queue is the fundamental model for a system that receives requests following a Poisson arrival process and serves them with exponentially distributed service times [21]. Several performance metrics, such as expected response time, throughput, and server utilization, can be analyzed with the M/M/1 model. However, more comprehensive models are needed if we consider factors like a buffer, multiple types of jobs, and the concurrency of queues. To incorporate such factors, a quasi-birth-death process can provide a more general representation of complex queueing systems, which is also represented by a Markov chain [22]. The transition rate matrix of a quasi-birth-death process is block-tridiagonal. Such a structure has been observed and exploited in the performance analysis of many application fields like telecommunication systems [23], cloud computing infrastructure [24]. This paper applies the quasi-birth-death process to model the multi-input ML system in the parallel system configuration. The novel performance metrics for evaluating the response time and the energy consumption are defined. For the performance of the shared architecture, only one existing study [13] developed the GI/M/1-type Markov chain since the arrival stream of the shared architecture is regarded as a mixture of multiple types of data from different sources. However, the study only evaluates the throughput of the system because the response time distribution based on GI/M/1-type Markov chain cannot be easily derived analytically. In this study, we develop a simulation program to evaluate the response time distribution. In comparison with [13], numerical results are newly designed and updated. Furthermore, the response time distributions for both parallel system configuration and shared architecture are investigated by analysis and/or simulations. In addition, we also consider simple models that do not consider synchronization and comparison unit and show that these simple models cannot capture the performance of our proposed parallel and shared architectures and show that these simple models cannot capture the characteristics of our proposed architectures.

3 System configuration

We consider a reliable AIoT system adopting a multi-input ML system configuration. A multi-input ML system is a type of N-version ML system that uses multiple input data to diversify the inference results and enable reliable decision-making. Instead of relying on a single inference result by an ML module, a multi-input AIoT system determines the system output by comparing multiple inference results from ML modules. Diverse inference results can be generated by simply using slightly different input data (e.g., images captured from different sensors). For example, automated vehicles are typically equipped with multiple cameras and other sensors that can jointly enhance perception reliability and driving safety. Diverse images captured by RGB cameras can be used to build a multi-input ML system configuration.

However, considering the operation of the AIoT system with multiple inputs, multiple executions of ML inferences can affect system performance and energy consumption. The performance and energy overheads may vary depending on the choice of system configuration. For a system with two different sensory inputs, a straightforward configuration is to use duplicate ML modules, which we call the parallel architecture. Each ML module consumes input data from the corresponding data source and independently outputs inference results. On the other hand, we can adopt a configuration with a single ML module to produce two outputs from two different data sources. The configuration is referred to as the shared architecture. As the parallel and the shared architectures are two basic configurations that need to be selected in multi-input AIoT system design, we focus on comparing the two configurations in this paper. The following subsections explain the system models in detail.

3.1 Two input data sources

We consider a system with two distinct data sources that continuously supply data to ML modules for inference. Each data source generates slightly different data but represents the same object state in the real world (e.g., photo images captured by cameras from different angles). The data is supplied from the data sources independently and arrives in the buffers on the ML modules. Each input data is regarded as a

job for ML modules, and hence we refer to arriving data from the first and second sources as type-1 jobs and type-2 jobs, respectively. For the sake of architecture comparison based on queueing analysis, we assume the arrival processes of jobs of type 1 and type 2 follow the Poisson process with rate λ_1 and λ_2, respectively. The arrival process may depend on sensor types and/or system environments, which may not be well captured by a Poisson process. We consider the architecture comparison under a more complex data arrival process as future work.

3.2 Parallel architecture

The parallel architecture uses two ML modules in parallel, corresponding to two input data sources. We call module 1 and module 2 for serving jobs of type 1 and type 2, respectively. Figure 1 shows the system configuration. Each ML module has a buffer to keep the jobs and independently processes the input data on a first-come, first-served (FCFS) discipline (e.g., detect the object in the image data). We assume the processing times of ML modules are exponentially distributed with rates μ_1 and μ_2, respectively. Furthermore, we assume that the maximum number of jobs of type 2 is $K < \infty$ while the buffer size of module 1 is infinite. This exponential-distribution assumption and $K < \infty$ make the queueing models analytically tractable in this study, while we consider relaxing this assumption in future work. Once the two ML modules complete the processing, the inference results are compared at the comparison unit to determine the final system output (e.g., output the result if the two modules agree on the results). The processing time of the comparison unit is assumed to be exponentially distributed with rate μ. Since the modules' outputs need to be synchronized for comparison, the module that completes faster must wait for the other module to finish processing.

3.3 Shared architecture

The shared architecture uses a single ML module to process both types of jobs on an FCFS discipline. Figure 2 shows the system configuration. The aggregated job arrival process to the shared ML module becomes the Poisson process with rate $\lambda_1 + \lambda_2$. The ML module has a buffer of infinite size to hold jobs waiting to be processed. The processing time depends on the type of job. We assume that the processing time of a type 1 job and a type 2 job is exponentially distributed with rates μ_1 and μ_2, respectively. To compare the inference results of the two job types, once the jobs are completed, the results are sent to the

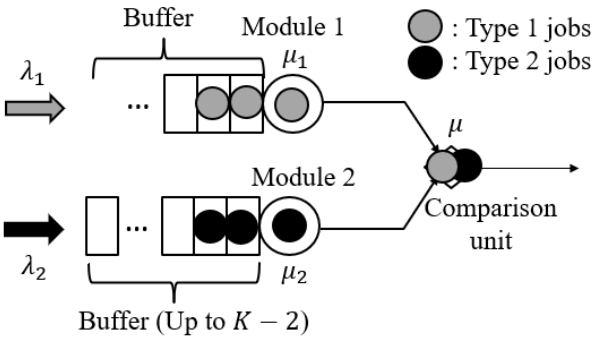

Figure 1. The system configuration of the parallel architecture [13].

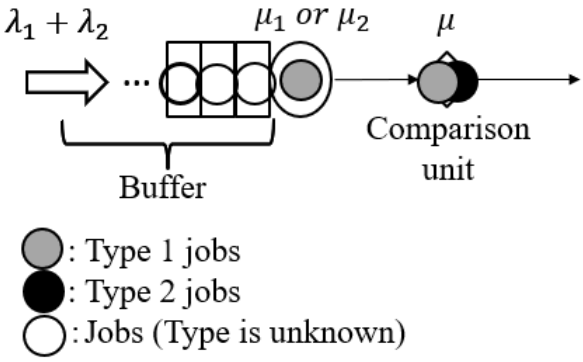

Figure 2. The system configuration of the shared architecture [13].

comparison unit. The processing at the comparison unit is the same as the parallel architecture. While the comparison unit is processing, the ML module pauses processing and waits for the comparison to complete. Jobs of type 1 and type 2 may not arrive alternately. When the ML module completes a job, it deletes the job from the head of the buffer one by one until it finds another job to compare.

4 Queueing models

In this section, we propose queueing models that represent the dynamics of job processing in a multi-input AIoT system, both in parallel and in shared architectures. First, we introduce the simple M/M/1 queueing models that approximate the queuing processes without considering the comparison unit. Next, we propose more comprehensive models that include the comparison unit process.

4.1 Simple queueing models

To compare the performance of the parallel and shared architectures without considering the comparison step, we can use a simple queueing analysis based

on M/M/1 queues (with infinite buffer sizes). As described in the previous section, we assume the job arrival process follows the Poisson process, while the service times follow the exponential distributions. The behavior of individual ML modules can be modeled using M/M/1 queues.

Parallel architecture. When we neglect the comparison unit of parallel architecture, the two ML modules can be modeled as two independent M/M/1 queues with arrival rates λ_1, λ_2 and service rates μ_1, μ_2, respectively.

Shared architecture. As the data arrives at a shared architecture from the two data sources, the job arrival process can be characterized by the Poisson process with rate $\lambda_1 + \lambda_2$. We can model the behavior of the shared architecture using an M/M/1 queue with arrival rate $\lambda_1 + \lambda_2$ and service rate μ_1, assuming $\mu_1 = \mu_2$. It should be noted that if $\mu_1 \neq \mu_2$, the service time distribution of an arbitrary job is not exponential anymore but a hyper-exponential distribution, and thus the analysis is more complicated. Here, for comparison purposes, we consider the simple case where $\mu_1 = \mu_2$ to utilize the known results of the M/M/1 queue.

4.2 Queueing model for parallel architecture

While simple queueing models can capture the abstract behavior of the Parallel architecture, a more comprehensive model is needed to account for the comparison process. Define $N(t)$ and $N_2(t)$ as the numbers of type 1 and type 2 jobs in the system at time t. Also, we denote $N_{m1}(t), N_{m2}(t)$, and $N_{c1}(t) \in \{0, 1\}$ as the states of module 1, module 2, and the comparison unit at t, respectively, where 0 represents the idle state and 1 represents the processing state. The state of the parallel architecture can be represented by $X_P(t) = (N(t), N_2(t), N_{m1}(t), N_{m2}(t), N_{c1}(t))$. $\{X_P(t); t \geq 0\}$ is a continuous-time Markov chain (CTMC) in the state space S_P representing the set of reachable states. S_P is given by $S_P = \cup_{i=0}^{\infty} \mathcal{L}_i^P$, where $\mathcal{L}_i^P$ represents level i defined below.

As the job arrival and processing can be regarded as a quasi-birth-death process [22], the infinitesimal generator of the CTMC is given by the following block matrices, with the state space into i parts by the number of type 1 jobs in the system.

$$Q_P = \begin{array}{c} \\ \mathcal{L}_0^P \\ \mathcal{L}_1^P \\ \mathcal{L}_2^P \\ \mathcal{L}_3^P \\ \mathcal{L}_4^P \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathcal{L}_0^P & \mathcal{L}_1^P & \mathcal{L}_2^P & \mathcal{L}_3^P & \mathcal{L}_4^P & \mathcal{L}_5^P & \cdots \end{array} \\ \begin{pmatrix} B_0 & C_0 & O & O & O & O & \cdots \\ A_1 & B_1 & C_1 & O & O & O & \cdots \\ O & A_2 & B_2 & C_2 & O & O & \cdots \\ O & O & A_3 & B_2 & C_2 & O & \cdots \\ O & O & O & A_3 & B_2 & C_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \end{array} \tag{1}$$

where $\mathcal{L}_i^P$ are the set of states with i type 1 jobs in the system, O is a zero matrix of appropriate size, and B_i, C_i, and A_i are the block matrices representing the state transitions within $\mathcal{L}_i^P$, the transitions with a job arrival (i.e., i is incremented), and the transitions with a job departure (i.e., i is decremented), respectively. The elements in $\mathcal{L}_0^P, \mathcal{L}_1^P, \mathcal{L}_i^P\ (i \geq 2)$ are sorted in the increasing order of $N_2(t)$ and the last three elements (i.e, $N_{m1}(t), N_{m2}(t), N_{c1}(t)$) are sorted in an appropriate order convenient for computation. The $\mathcal{L}_0^P, \mathcal{L}_1^P, \mathcal{L}_i^P$ are given as below.

$$\begin{aligned}
\mathcal{L}_0^P = {} & \{(0,0,0,0,0)\} \cup \cdots \cup \{(0,K,0,0,0)\} \cup \\
& \{(0,1,0,1,0)\} \cup \cdots \cup \{(0,K,0,1,0)\}, \\
\mathcal{L}_1^P = {} & \{(1,0,0,0,0)\} \cup \\
& \{(1,0,1,0,0)\} \cup \cdots \cup \{(1,K,1,0,0)\} \cup \\
& \{(1,1,0,1,0)\} \cup \cdots \cup \{(1,K,0,1,0)\} \cup \\
& \{(1,1,0,0,1)\} \cup \cdots \cup \{(1,K,0,0,1)\} \cup \\
& \{(1,1,1,1,0)\} \cup \cdots \cup \{(1,K,1,1,0)\} \cup \\
& \{(1,2,0,1,1)\} \cup \cdots \cup \{(1,K,0,1,1)\}, \\
\mathcal{L}_i^P = {} & \{(i,0,0,0,0)\} \cup \\
& \{(i,0,1,0,0)\} \cup \cdots \cup \{(i,K,1,0,0)\} \cup \\
& \{(i,1,0,1,0)\} \cup \cdots \cup \{(i,K,0,1,0)\} \cup \\
& \{(i,1,0,0,1)\} \cup \cdots \cup \{(i,K,0,0,1)\} \cup \\
& \{(i,1,1,1,0)\} \cup \cdots \cup \{(i,K,1,1,0)\} \cup \\
& \{(i,1,1,0,1)\} \cup \cdots \cup \{(i,K,1,0,1)\} \cup \\
& \{(i,2,0,1,1)\} \cup \cdots \cup \{(i,K,0,1,1)\} \cup \\
& \{(i,2,1,1,1)\} \cup \cdots \cup \{(i,K,1,1,1)\}.
\end{aligned}$$

All the elements of B_i, C_i, and A_i are specified in the Appendix. We consider the process $\{X_P(t); t \geq 0\}$ under the stability condition. We define $Q_P^* := A_3 + B_2 + C_2$ and ξ as the stationary distribution of Q_P^*, i.e., $\xi Q_P^* = 0, \xi e = 1$, where e represents a column vector of an approprite size whose components are all 1. The stability condition of $\{X_P(t); t \geq 0\}$ is given by

$\xi(C_2 - A_3)e < 0$. Under the stability condition, the stationary distribution of $X_P(t)$ for $(i,j,k,m,l) \in S_P$ is given by

$$\pi^P_{i,j,k,m,l} = \lim_{t\to\infty} P(X_P(t) = (i,j,k,m,l)) \tag{2}$$

The stationary distribution for the states with i type 1 jobs is expressed as $\pi^P_i = (\pi^P_{i,j,k,m,l})_{(i,j,k,m,l)\in\mathcal{L}^P_i}$. According to matrix analytic method, π^P_i is given as follows.

$$\pi^P_i = \begin{cases} \pi^P_0 & (i=0), \\ \pi^P_1 & (i=1), \\ \pi^P_2 R_P^{i-2} & (i \geq 2). \end{cases} \tag{3}$$

In (3), R_P is the rate matrix of the quasi-birth-and-death process, which is the minimal non-negative solution of the following equation.

$$C_2 + R_P B_2 + R_P^2 A_3 = 0,$$

which is algorithmically solved [22]. The boundary vectors $\pi^P_0, \pi^P_1, \pi^P_2$ are determined by (4) and (5), where e^P_0, e^P_1, e^P_2 are column vectors of sizes $2K+1$, $5K+1$, $7K$ with all elements being 1.

$$\begin{cases} \pi^P_0 B_0 + \pi^P_1 A_1 = 0, \\ \pi^P_0 C_0 + \pi^P_1 B_1 + \pi^P_2 A_2 = 0, \\ \pi^P_1 C_1 + \pi^P_2 B_2 + \pi^P_2 R_P A_3 = 0. \end{cases} \tag{4}$$

$$\pi^P_0 e^P_0 + \pi^P_1 e^P_1 + \pi^P_2 (I - R_P)^{-1} e^P_2 = 1. \tag{5}$$

4.3 Queueing model for shared architecture

As in the parallel-architecture queueing model, we first define the states of the shared architecture. Define $L(t)$ as the number of any jobs waiting in the buffer at t. Let $N_m(t) \in \{0 \dots 7\}$ represent the state of the ML module where the values are defined as follows:

0. idle
1. only type 1 jobs have been processed
2. only type 1 job is in processing
3. type 1 jobs has been processed and type 2 job is in processing
4. only type 2 jobs have been processed
5. only type 2 job is in processing
6. type 2 jobs has been processed and type 1 job is in processing
7. both two types of jobs have been processed and waiting for comparison process

Also, define $N_{c2}(t) \in \{0,1\}$ as the state of the comparison unit, where 0 represents the idle state, and 1 represents the processing state. The state of the shared architecture can be represented by $X_S(t) = (L(t), N_m(t), N_{c2}(t))$. $\{X_S(t); t \geq 0\}$ is a CTMC in the state space S_S representing the set of reachable states. In particular, $S_S = \cup_{i=0}^{\infty} \mathcal{L}^S_i$, where $\mathcal{L}^S_i$ represents level i defined below.

As the process can be looked as a CTMC of GI/M/1 type (also called skip-free to the right) [22], we can derive the infinitesimal generator of the CTMC as the block matrix segmented by the number of jobs in the system.

$$Q_S = \begin{array}{c} \\ \mathcal{L}^S_0 \\ \mathcal{L}^S_1 \\ \mathcal{L}^S_2 \\ \mathcal{L}^S_3 \\ \mathcal{L}^S_4 \\ \mathcal{L}^S_5 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathcal{L}^S_0 & \mathcal{L}^S_1 & \mathcal{L}^S_2 & \mathcal{L}^S_3 & \mathcal{L}^S_4 & \mathcal{L}^S_5 & \cdots \end{array} \\ \begin{pmatrix} B_0 & C_0 & O & O & O & O & \cdots \\ B_1 & A_1 & A_0 & O & O & O & \cdots \\ B_2 & A_2 & A_1 & A_0 & O & O & \cdots \\ B_3 & A_3 & A_2 & A_1 & A_0 & O & \cdots \\ B_4 & A_4 & A_3 & A_2 & A_1 & A_0 & \cdots \\ B_5 & A_5 & A_4 & A_3 & A_2 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array} \tag{6}$$

where $\mathcal{L}^S_i$ are the sets of states with i jobs in the system given by

$$\begin{aligned} \mathcal{L}^S_0 =& \{(0,0,0)\} \cup \cdots \cup \{(0,6,0)\} \cup \\ & \{(0,0,1)\} \cup \cdots \cup \{(0,7,1)\}, \\ \mathcal{L}^S_i =& \{(i,2,0)\} \cup \{(i,3,0)\} \cup \{(i,5,0)\} \cup \{(i,6,0)\} \cup \\ & \{(i,2,1)\} \cup \{(i,3,1)\} \cup \{(i,5,1)\} \cup \{(i,6,1)\} \cup \\ & \{(i,7,1)\}. \end{aligned}$$

The diagonal blocks B_0 and A_1 represent the state transitions within $\mathcal{L}^P_i$, while C_0 and A_0 represent the state transitions with a job arrival and O is a zero matrix of appropriate size. The block matrices in the lower triangle represent the transitions with decreased jobs in the system. $A_i\,(i \geq 2)$ represents the transitions with $i-1$ decreased jobs, and $B_i\,(i \geq 1)$ represents the transitions with i decreased jobs. All the elements of each matrix are given in the Appendix.

Defining $A = \sum_{n=0}^{\infty} A_n$, A represents the infinitesimal generator of the Markov chain of the phase. Letting l denote the stationary distribution of this Markov chain, i.e., $lA = 0, le = 1$, the stability condition of $\{X_S(t); t \geq 0\}$ is given by

$$lA_0e + l\sum_{n=1}^{\infty}(1-n)A_ne < 0. \tag{7}$$

The stationary distribution of $X_S(t)$ for $(i,j,k) \in S_S$ is given by $\pi^S_{i,j,k} = \lim_{t\to\infty} P(X_S(t) = (i,j,k))$. The stationary distribution for the state with i jobs is expressed as $\pi^S_i = (\pi^S_{i,j,k})_{(i,j,k)\in\mathcal{L}^S_i}$.

$$\pi^S_i = \begin{cases} \pi^S_0 & (i=0), \\ \pi^S_1 R^{i-1}_S & (i \geq 1). \end{cases} \tag{8}$$

In (8) R_S is the minimal non-negative solution of the following equation:

$$O = \sum_{i=0}^{\infty} R^i_S A_i,$$

which is also numerically solved [22].

Furthermore, let e^S_0, e^S_1 denote the vector of size 15, 9 with all elements being 1. The boundary vectors π^S_0 and π^S_1 are determined by the following equations.

$$\begin{cases} \pi^S_0 B_0 + \pi^S_1(\sum_{i=1}^{\infty} R^{i-1}_S B_i) = 0, \\ \pi^S_0 C_0 + \pi^S_1(\sum_{i=1}^{\infty} R^{i-1}_S A_i) = 0. \end{cases} \tag{9}$$

$$\pi^S_0 e^S_0 + \pi^S_1 (I - R_S)^{-1} e^S_1 = 1. \tag{10}$$

5 Performance measures

To compare the performance of the parallel and the shared architectures, we define three performance measures; throughput, response time, and energy consumption. Table 1 shows the notations of measures defined for the simple and the proposed queueing models.

Table 1. Performance measures.

Measure	Architecture	Proposed	Simple
Throughput	Parallel	TP_P	$\widetilde{\mathrm{TP}}_P$
	Shared	TP_S	$\widetilde{\mathrm{TP}}_S$
Response time	Parallel	RT_P	$\widetilde{\mathrm{RT}}_P$
	Shared	RT_S	$\widetilde{\mathrm{RT}}_S$
Energy cons.	Parallel	EC_P	$\widetilde{\mathrm{EC}}_P$
	Shared	EC_S	$\widetilde{\mathrm{EC}}_S$

5.1 Throughput

Throughput is measured by the number of jobs processed per unit time.

Simple models. In order to make a fair comparison of the throughputs in the simple queueing models, the throughput is measured by the number of times that two jobs finish processing per unit time. We can express the throughputs of the simple queueing models $\widetilde{\mathrm{TP}}_P$ (for parallel architecture) and $\widetilde{\mathrm{TP}}_S$ (for shared architecture) as follows.

$$\widetilde{\mathrm{TP}}_P = \widetilde{\mathrm{TP}}_S = \frac{\lambda_1 + \lambda_2}{2} \tag{11}$$

Parallel architecture model. When the behavior of the parallel architecture is modeled as the CTMC shown in Section 4.2, the throughput is measured by the number of jobs processed at the comparison unit per unit time. As presented in [13], the throughput can be defined as

$$\mathrm{TP}_P = \left(\pi^P_1 e^{**}_{P1} + \sum_{i=2}^{\infty} \pi^P_i e^{**}_{P2}\right)\mu, \tag{12}$$

where e^{**}_{P1} and e^{**}_{P2} are column vectors whose elements are 1 if $N_{c1}(t) = 1$ in the corresponding state otherwise 0.

Shared architecture model. When the behavior of the shared architecture is modeled as the CTMC shown in Section 4.3, the throughput is measured by the number of jobs completed in the comparison process per unit time. As presented in [13], the throughput is expressed as

$$\mathrm{TP}_S = \left(\pi^S_0 e^*_{S0} + \sum_{i=1}^{\infty} \pi^S_i e^*_{S1}\right)\mu, \tag{13}$$

where e^*_{S0}, e^*_{S1} are column vectors given by

$$e^*_{S0} = (0,0,0,0,0,0,0,1,1,1,1,1,1,1,1)^T,$$
$$e^*_{S1} = (0,0,0,0,1,1,1,1,1)^T.$$

For elements in e^*_{S0}, e^*_{S1}, 1 corresponds to the state that the throughput is calculated.

5.2 Response time

The mean response time of a job is the average time from when the job enters the system to when it leaves.

Simple models. When we consider simple queuing models, the mean response time of the pair can be

derived from the response time distribution of M/M/1 model as follows.

$$\mathrm{RT}_P^{\sim} = \frac{(\mu_1-\lambda_1)^2+(\mu_2-\lambda_2)^2+(\mu_1-\lambda_1)(\mu_2-\lambda_2)}{(\mu_1-\lambda_1)(\mu_2-\lambda_2)(\mu_1-\lambda_1+\mu_2-\lambda_2)}, \tag{14}$$

$$\mathrm{RT}_S^{\sim} = \frac{3}{2(\mu_1-(\lambda_1+\lambda_2))}. \tag{15}$$

Formulae (14) and (15) are derived using the response time (sojourn time) distribution of an arbitrary job in an M/M/1 queue. For (14), let Y_1 and Y_2 denote the response time of a job in two M/M/1 queues with arrival and service rates (λ_1, μ_1) and (λ_2, μ_2), respectively. We have, $\mathrm{RT}_S^{\sim} = E[\max(Y_1, Y_2)]$ and after some calculations we obtain (14). As for (15) is obtained by the same formula $\mathrm{RT}_P^{\sim} = E[\max(Y_1, Y_2)]$, where Y_1 and Y_2 follow the same sojourn time distribution of an M/M/1 queue with arrival rate $\lambda_1+\lambda_2$ and service rate μ_1, as jobs of both types are fed to the same queue.

Parallel architecture model. When the behavior of the parallel architecture is modeled as the CTMC shown in Section 4.2, the mean response time of any job in the parallel architecture is given by

$$\mathrm{RT}_P = \frac{E[L_1]+E[L_2]}{\lambda_1+\lambda_2(1-P_b)}, \tag{16}$$

due to Little's law. P_b represents the blocking probability of a type 2 job, and $E[L_1]$ and $E[L_2]$ represent the average number of jobs of type 1 and type 2 in the system, respectively. $P_b, E[L_1], E[L_2]$ are expressed as follows.

$$P_b = \sum_{i=0}^{\infty} \pi^P_{(i,K)}, \tag{17}$$

$$E[L_1] = \pi_2^P(R_P(I-R_P)^{-2}+2(I-R_P)^{-1})e_{P12}+ \pi_1^P e_{P11}, \tag{18}$$

$$E[L_2] = \pi_0^P e_{P20} + \pi_1^P e_{P21} + \pi_2^P(I-R_P)^{-1}e_{P22}, \tag{19}$$

where e_{P11} and e_{P12} are column vectors of size $5K+1$ and $5K+7$, respectively, with all elements being 1. Also, e_{P20}, e_{P21} and e_{P22} are given by

$$\begin{aligned}
e_{P20} &= (0,1,\ldots,K,1,2,\ldots,K)^T,\\
e_{P21} &= (0,0,1,\ldots,K,1,2,\ldots,K,1,2,\ldots,K,1,2,\ldots \\
&\quad ,K,2,3,\ldots,K)^T,\\
e_{P22} &= (0,0,1,\ldots,K,1,2,\ldots,K,1,2,\ldots,K,1,2,\ldots, \\
&\quad K,1,2,\ldots,K,2,3,\ldots,K,2,3,\ldots,K)^T.
\end{aligned}$$

The expectations $E[L_1]$ and $E[L_2]$ are first expressed in terms of the stationary probabilities using the definition, and after some rearrangements using (3), (18), and (19) follow.

Shared architecture model. For the shared architecture modeled as the CTMC shown in Section 4.3, the mean response time is given by

$$\mathrm{RT}_S = \frac{E[L_s]}{\lambda_1+\lambda_2}, \tag{20}$$

due to Little's law.

$E[L_s]$ represents the average number of jobs in the system, and it is expressed as follows.

$$\begin{aligned}
E[L_S] =& \pi_0^S(e_{S10}+2e_{S20}+3e_{S30}+4e_{S40})+ \\
& \pi_1^S R_S(I-R_S)^{-2}e_{S1}+ \\
& \pi_1^S(I-R_S)^{-1}(e_{S1}+e_{S11}+2e_{S21}+3e_{S31}+4e_{S41}),
\end{aligned} \tag{21}$$

where e_{S1} is column vectors of size 9 and $e_{S10}, e_{S20}, e_{S30}$ and e_{S40} are column vectors of size 15 and $e_{S11}, e_{S21}, e_{S31}$ and e_{S41} are column vectors of size 9 and are given by

$$\begin{aligned}
e_{S10} &= (0,1,1,0,1,1,0,\ldots,0)^T,\\
e_{S20} &= (0,0,0,1,0,0,1,0\ldots,0)^T,\\
e_{S30} &= (0,0,0,0,0,0,0,0,1,1,0,1,1,0,0)^T,\\
e_{S40} &= (0,0,0,0,0,0,0,0,0,0,1,0,0,1,1)^T,\\
e_{S11} &= (1,0,1,0,\ldots,0)^T,\\
e_{S21} &= (0,1,0,1,\ldots,0)^T,\\
e_{S31} &= (0,0,0,0,1,0,1,0,0)^T,\\
e_{S41} &= (0,0,0,0,0,1,0,1,1)^T.
\end{aligned}$$

The expectation $E[L_S]$ is first expressed in terms of the stationary distribution, and using the matrix geometric form (8), (21) follows.

We can also derive the distribution of RT_S. Assuming that the number of jobs existing in the system when a tagged job arrives at the system is k, the state when $(k+1)$ jobs leave the system is the absorbing state. The duration from the time when the tagged job arrives until the time that the job leaves the system follows a phase-type distribution whose infinitesimal generator Q_S^* is expressed as follows. Here, the transition rate matrix for the temporary (transient) state, excluding the taboo state, is given by $\hat{Q}_S$.

$$Q_S^* = \begin{pmatrix} 0 & O \\ \hat{q}_S^T & \hat{Q}_S \end{pmatrix}, \tag{22}$$

where $\hat{q}_S^T = -\hat{Q}_S 1^T$. We define a_{k+1} as a vector appropriately arranged using steady-state

probabilities depending on the state reached when the tagged job arrives at the system, and $a = (a_1, a_2, \dots)$. Besides, we define a_0 as the sum of the steady-state probabilities that the job leaves the system immediately after the arrival. Therefore, the distribution function and probability density function of the response time of any job are defined as follows using the phase-type distribution whose initial state distribution is given by $a^* = (a_0, a)$.

$$F(t) = 1 - a\exp\{\hat{Q}_S t\}1^T, (t \geq 0), \tag{23}$$

$$f(t) = a\exp\{\hat{Q}_S t\}\hat{q}_S^T, (t > 0). \tag{24}$$

It should be noted that because we consider an infinite buffer, the size of the matrix $\hat{Q}_S$ is infinite, but we appropriately truncate it in numerical computation.

Furthermore, we define the response time of the pair as the response time of the pair of jobs that have completed the comparison process.

5.3 Energy consumption

We measure the energy consumption of a multi-input AIoT system by the amount of energy consumed in the ML modules and the comparison unit. Assuming that the system is implemented with the same hardware and the same model, we use the normalized measure such that the energy consumption in processing state is 1 and that in idle state is $\alpha \in (0, 1)$.

Simple models. For the simple queueing models, we do not consider the energy consumption in the comparison unit. Therefore, the energy consumption of the parallel architecture $\mathrm{EC}_P^{\sim}$ and the shared one $\mathrm{EC}_S^{\sim}$ can be calculated by

$$\mathrm{EC}_P^{\sim} = \frac{(1-\alpha)\lambda_1}{\mu_1} + \frac{(1-\alpha)\lambda_2}{\mu_2} + 2\alpha, \tag{25}$$

$$\mathrm{EC}_S^{\sim} = \frac{(1-\alpha)(\lambda_1+\lambda_2)}{\mu_1} + \alpha. \tag{26}$$

Parallel architecture model. Define $\mathrm{EC}_{Pm1}, \mathrm{EC}_{Pm2}$ and EC_{Pc} as the energy consumption of the module 1, module 2, and the comparison unit in parallel architecture, respectively, as follows.

$$\mathrm{EC}_{Pm1} = (1-\alpha)(\pi_1^P e_{P11}^* + \pi_2^P (I - R_P)^{-1} e_{P12}^*) + \alpha, \tag{27}$$

$$\mathrm{EC}_{Pm2} = (1-\alpha)(\pi_1^P e_{P20}^* + \pi_1^P e_{P21}^* + \pi_2^P (I - R_P)^{-1} e_{P22}^*) + \alpha, \tag{28}$$

$$\mathrm{EC}_{Pc} = (1-\alpha)(\pi_1^P e_{P31}^* + \pi_2^P (I - R_P)^{-1} e_{P32}^*) + \alpha. \tag{29}$$

where e_{P20}^* is a column vector of size $2K+1$ with all elements in $K+2 \sim 2K+1$ rows being 1 and the other elements being 0; e_{P11}^* is a column vector of size $5K+1$ and with all elements in $2 \sim K+2$ and $3K+3 \sim 4K+2$ rows being 1 and the other elements being 0; e_{P21}^* is a column vector of size $5K+1$ and with all elements in $K+3 \sim 2K+2$ and $3K+3 \sim 5K+1$ rows being 1 and the other elements being 0; e_{P31}^* is a column vector of size $5K+1$ and with all elements in $2K+3 \sim 3K+2$ and $4K+3 \sim 5K+1$ rows being 1 and the other elements being 0; e_{P12}^* is a column vector of size $7K$ and with all elements in $2 \sim K+2$ and $3K+3 \sim 5K+2$ and $6K+2 \sim 7K$ rows being 1 and the other elements being 0; e_{P22}^* is a column vector of size $7K$ and with all elements in $K+3 \sim 2K+2$ and $3K+3 \sim 4K+2$ and $5K+3 \sim 7K$ rows being 1 and the other elements being 0, and e_{P32}^* is a column vector of size $7K$ and with all elements in $2K+3 \sim 3K+2$ and $4K+3 \sim 7K$ rows being 1 and the other elements being 0. The total energy consumption of the parallel ML modules, which does not include the comparison unit, is given by

$$\mathrm{EC}_P = \mathrm{EC}_{Pm1} + \mathrm{EC}_{Pm2}. \tag{30}$$

Shared architecture model. Define EC_S and EC_{Sc} as the energy consumption of the ML module and the comparison unit in shared architecture, respectively, as follows.

$$\mathrm{EC}_S = (1-\alpha)(\pi_0^S e_{S10}^* + \pi_1^S (I - R_S)^{-1} e_{S11}^*) + \alpha, \tag{31}$$

$$\mathrm{EC}_{Sc} = (1-\alpha)(\pi_0^S e_{S20}^* + \pi_1^S (I - R_S)^{-1} e_{S21}^*) + \alpha. \tag{32}$$

where e_{S10}^*, e_{S20}^*, e_{S30}^*, and e_{S40}^* are column vectors of size 15 and e_{S11}^*, e_{S21}^*, e_{S31}^*, and e_{S41}^* are column vectors of size 9 and are given by

$$\begin{aligned}
e_{S10}^* &= (0,1,1,0,1,1,0,\dots,0)^T,\\
e_{S20}^* &= (0,0,0,1,0,0,1,0\dots,0)^T,\\
e_{S30}^* &= (0,0,0,0,0,0,0,0,1,1,0,1,1,0,0)^T,\\
e_{S40}^* &= (0,0,0,0,0,0,0,0,0,0,1,0,0,1,1)^T,\\
e_{S11}^* &= (1,0,1,0,\dots,0)^T,\\
e_{S21}^* &= (0,1,0,1,0,\dots,0)^T,\\
e_{S31}^* &= (0,0,0,0,1,0,1,0,0)^T,\\
e_{S41}^* &= (0,0,0,0,0,1,0,1,1)^T.
\end{aligned}$$

6 Experiments

We compare the parallel and the shared architectures for a multi-input AIoT system with respect to the performance metrics defined in Section 5. We conduct

both Monte Carlo simulations and numerical analyses to confirm the evaluation results. To show the effectiveness of the proposed models, we also compare the results obtained by the analysis of simple queueing models presented in Section 4.1. In the simulation, we started with the state that the system is empty, i.e., all the components of the Markov chains are 0. Then we simulate 1100000 events (transitions of the Markov chain) while the first 100000 events are discarded and the rest 1000000 events are used to take the average of the performance measures. With this setting, in all the experiments, we confirm high consistency between simulation and analysis for the same model.

6.1 Throughput

As the throughput of the parallel and the shared architectures was evaluated in [13], we only highlight the comparison results and show new results here. The previous study observed that the parallel architecture generally achieves higher throughput than the shared one, especially when the service rate is high. The result can be simply interpreted as the difference in the computing resources available in the parallel architecture, which is twice as large as the shared one. However, it is noted that the throughput is not proportional to the number of modules. The throughput of the parallel architecture with two ML modules is approximately 1.4 times compared to the throughput of the shared architecture [13]. It is caused by the difference in the processing at the comparison unit.

Since we also consider the simple queuing models that give the approximated performance evaluations, we compare $\mathrm{TP}_P, \mathrm{TP}_S$ with $\mathrm{TP}_P^{\sim}, \mathrm{TP}_S^{\sim}$ (the throughput for the simple queueing models) under the same parameter conditions. For the parallel architecture, we fix $\mu = 50, K = 80, \lambda_1 = 1.4, 2.4, \lambda_2 = 1.5, 2.5$. For the shared architecture, we fix $\mu = 50, \lambda_1 = \lambda_2 = 1.5, 2.5$. Figures 3 and 4 show the comparison results of the parallel and the shared architectures, respectively, by varying the value of $\mu_1 = \mu_2$.

In Figure 3, we observe that the value of TP_P is not much different from the value of the simple model $\mathrm{TP}_P^{\sim}$ for the same parameter setting. On the other hand, when we look at Figure 4, the value of TP_S is about $\frac{2}{3}$ of the value of the simple model $\mathrm{TP}_S^{\sim}$ regardless of the value of $\mu_1 = \mu_2$. The simple queueing model can be used to obtain the approximated throughput for the parallel architecture, but it is not accurate for estimating the throughput of the shared architecture.

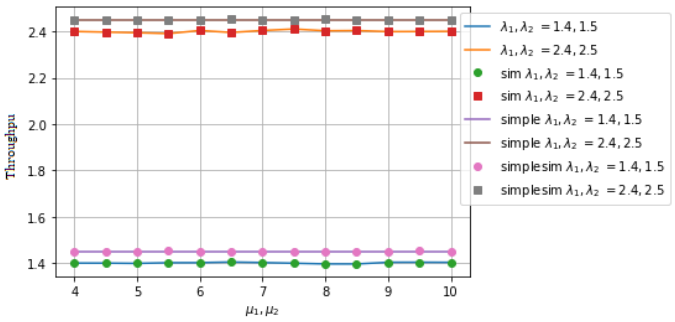

Figure 3. TP_P and the throughput of the simple model $\mathrm{TP}_P^{\sim}$ by varying the value of μ_1, μ_2.

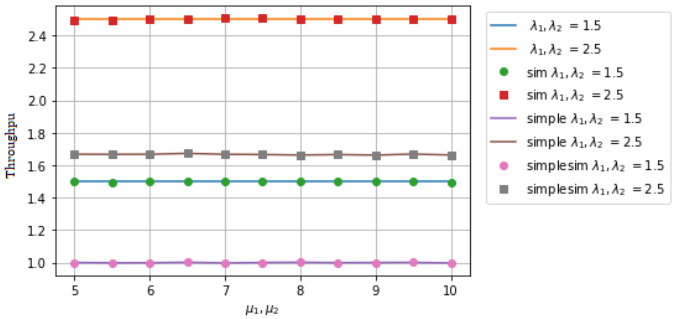

Figure 4. TP_S and the throughput of the simple model $\mathrm{TP}_S^{\sim}$ by varying the value of μ_1, μ_2.

6.2 Response time

Next, we consider the response time. We evaluate the response time of any job by varying the value of $\mu_1 = \mu_2$ with $\mu = 50, K = 80$ while in the parallel architecture, λ_1 is fixed to 0.9, 1.4, 1.9, 2.4 and λ_2 is fixed to 1.0, 1.5, 2.0, 2.5 and in the shared architecture, λ_1, λ_2 are fixed to 1.0, 1.5, 2.0, 2.5. Figures 5 and 6 show the results of the response times of the parallel and the shared architectures, respectively. As we can observe, the value of RT_P and RT_S decrease as the value of μ_1, μ_2 increases. The value of RT_P is almost insensitive when μ_1, μ_2 is larger than a certain value. When $\mu_1, \mu_2 > 3$, the value of RT_S is much smaller than the value of RT_P regardless of the value

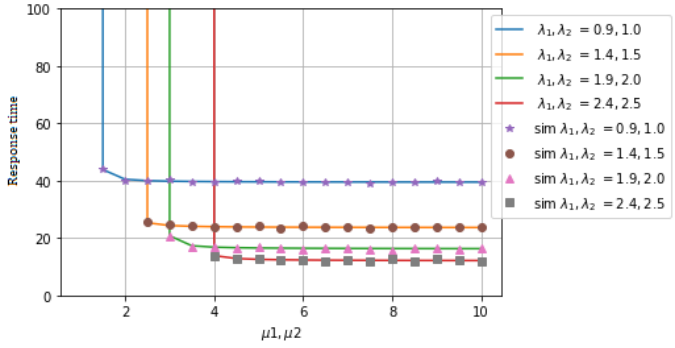

Figure 5. Response time of any job of the parallel architecture RT_P by varying the value of μ_1, μ_2.

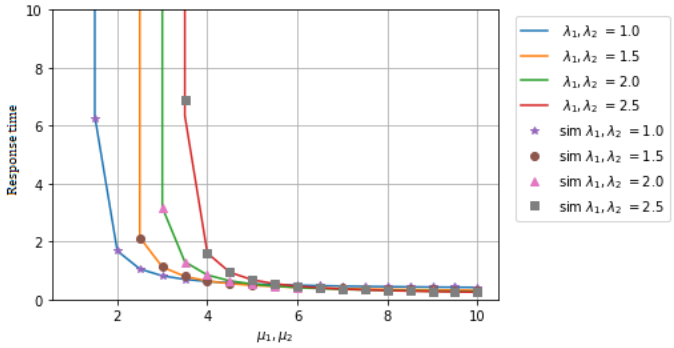

Figure 6. Response time of any job of the shared architecture RT_S by varying the value of μ_1, μ_2.

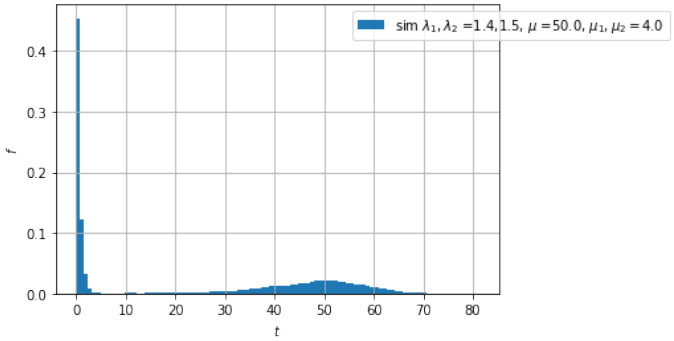

Figure 7. Distribution of response time of any job of the parallel architecture by varying the value of μ_1, μ_2.

of λ_1, λ_2. The results show the advantage of the shared architecture in the mean response time measure. It should be noted that RT_S calculated here represents the mean response time for all jobs, including those that may be temporarily deferred due to being the same type as the preceding job.

Next, we compute the distribution of the response time of any job with $\mu = 50, \mu_1 = \mu_2 = 4.0, K = 80, \lambda_1 = 1.4, \lambda_2 = 1.5$ in the parallel architecture and $\mu = 50, \mu_1 = \mu_2 = 4.0, \lambda_1, \lambda_2 = 1.5$ in the shared architecture. Figures 7 and 8 show the results of the parallel and the shared architectures, respectively.

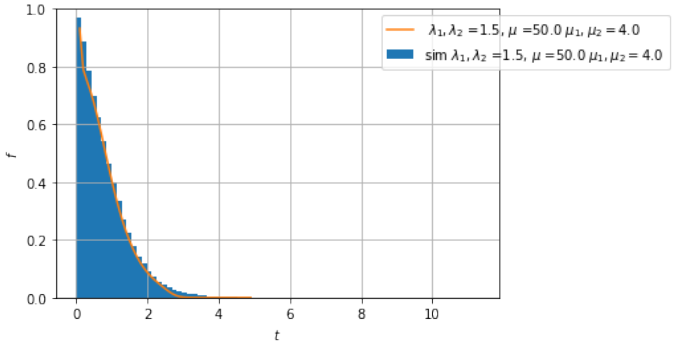

Figure 8. Distribution of response time of any job of the shared architecture by varying the value of μ_1, μ_2.

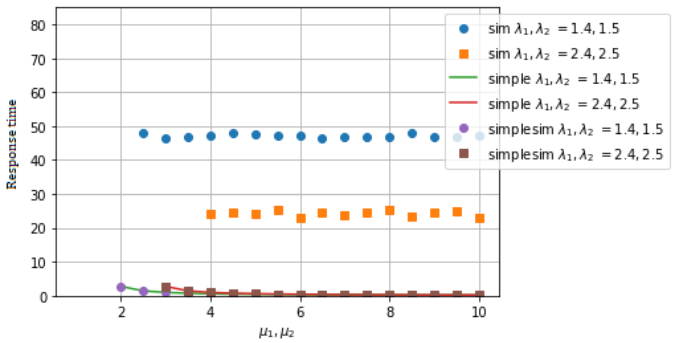

Figure 9. Response time of pair of the parallel architecture RT_P and the simple model $\widetilde{\mathrm{RT}_P}$ by varying the value of μ_1, μ_2.

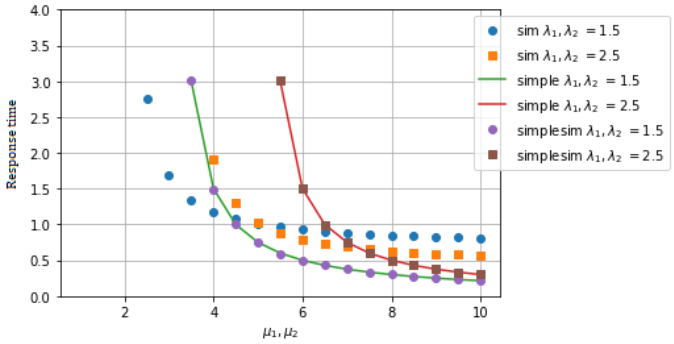

Figure 10. Response time of pair of the shared architecture RT_S and the simple model $\widetilde{\mathrm{RT}_S}$ by varying the value of μ_1, μ_2.

Finally, we compare the response times of the parallel and the shared architectures with $\mu = 50, K = 80$ and the results computed from the simple queueing models. For the parallel architecture, λ_1 is fixed to 1.4, 2.4 and λ_2 is fixed to 1.5, 2.5, while for the shared architecture, λ_1, λ_2 are fixed to 1.5, 2.5. Figures 9 and 10 show the comparison results of the parallel and the shared architectures, respectively, by varying the value of $\mu_1 = \mu_2$.

We observe that the value of RT_P is much smaller than the value of the simple model $\widetilde{\mathrm{RT}_P}$ for the same parameter setting. On the other hand, the value of RT_S is not much different from the value of the simple model $\widetilde{\mathrm{RT}_P}$. The results imply that our proposed model is especially necessary for the accurate evaluation of the response time of the parallel architecture.

From Figures 9 and 10, it is also confirmed that the mean response time of the parallel architecture is much longer than that of the shared one. It should be noted that the mean response time for the shared architecture did not account for jobs in front of the buffer, disregarded because they were the same type

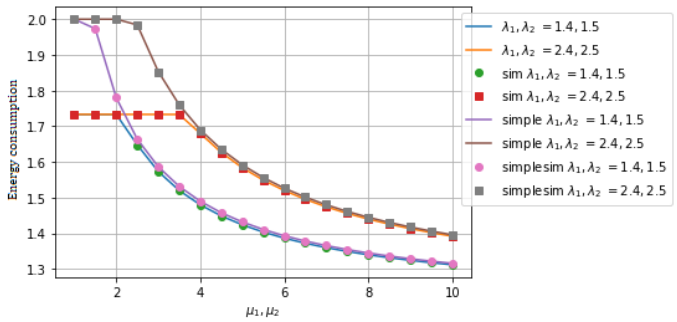

Figure 11. Energy consumption of the ML modules of the parallel architecture EC_P and the simple model $\mathrm{RT}_P^{\sim}$ by varying the value of μ_1, μ_2.

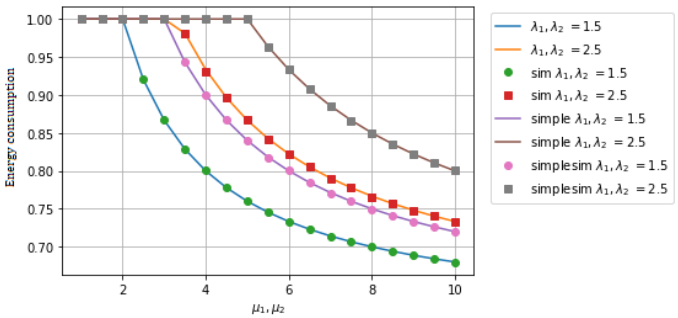

Figure 12. Energy consumption of the ML module of the shared architecture EC_S and the simple model $\mathrm{EC}_S^{\sim}$ by varying the value of μ_1, μ_2.

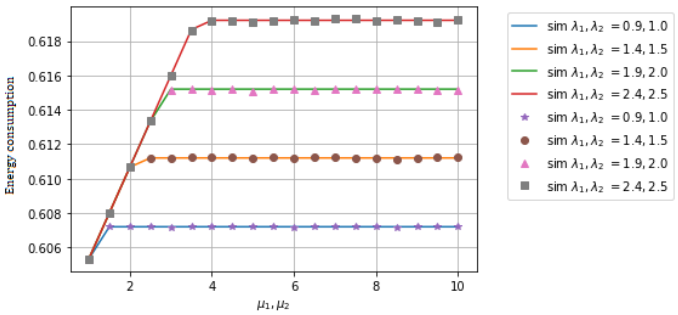

Figure 13. Energy consumption of the comparison unit of the parallel architecture EC_{Pc} by varying the value of μ_1, μ_2.

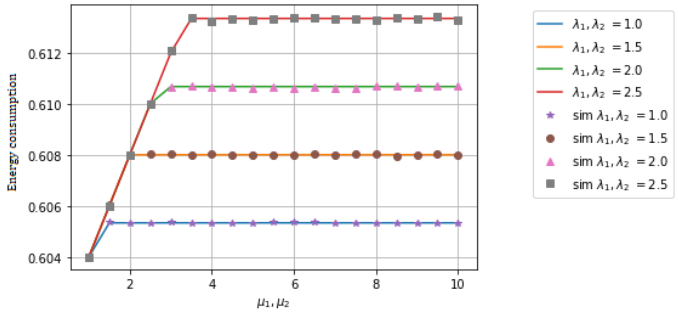

Figure 14. Energy consumption of the comparison unit of the shared architecture EC_{Sc} by varying the value of μ_1, μ_2.

as the completed job. Furthermore, the response time of a pair (that completed the comparison) is calculated as the time of the job that entered the system earlier, to ensure a safe evaluation. This complements the observation in Figures 5 and 6.

6.3 Energy consumption

Finally, we compute the energy consumption of the ML modules in the parallel and the shared architectures by varying the value of $\mu_1 = \mu_2$ with $\mu = 50, K = 80$. For the parallel architecture λ_1 is fixed to 1.4, 2.4 and λ_2 is fixed to 1.5, 2.5, while for the shared architecture λ_1, λ_2 are fixed to 1.5, 2.5. Figures 11 and 12 show the results of the parallel and the shared architectures, respectively. The results from the simple queueing models are also shown in the graphs. The value of EC_P and EC_S increases as the arrival rates $\lambda_1 = \lambda_2$ increase, while the value decreases as the value of μ_1, μ_2 increases to some extent, regardless of the parallel and the shared architectures. It is clear that the shared architecture has an advantage over the parallel one in terms of energy consumption. This result is also confirmed by an empirical study [25]. We observe that the value of EC_P is not much different from that of the simple model $\mathrm{EC}_P^{\sim}$ for the same parameter value if μ_1, μ_2 are large enough. On the other hand, the value of EC_S is smaller than that of the simple model $\mathrm{EC}_S^{\sim}$ for the same parameter value. Using the simple queueing model is inappropriate, especially when evaluating the energy consumption of the shared architecture.

Next, we evaluate the energy consumption of the comparison unit in the parallel and the shared architectures by varying the value of $\mu_1 = \mu_2$ with $\mu = 50, K = 80$. For the parallel architecture, λ_1 is fixed to 0.9, 1.4, 1.9, 2.4 and for the shared architecture λ_1, λ_2 are fixed to 1.0, 1.5, 2.0, 2.5. Figures 13 and 14 show the results of the parallel and the shared architectures, respectively. As the value of $\mu_1 = \mu_2$ increases, the value of EC_{Pc} and EC_{Sc} increases at a certain value. We observe that the value of EC_{Pc} is smaller than the value of EC_{Sc} regardless of λ_1, λ_2. The shared architecture is much more energy efficient than the parallel one, despite its lower throughput.

7 A guide to reliable AIoT system design

The goal of quality design for multi-input AIoT systems is to achieve high reliability and performance with a smaller energy consumption. From the numerical and simulation results on the proposed queueing models, we provide a guide to select the relevant architecture as follows.

As introduced in Section 3, both the parallel and

shared architectures determine the final output after comparing the two predictions for the two types of jobs. If the two prediction results are not consistent, either one of the predictions is wrong, and hence the comparison unit does not produce the final output. Therefore, the probability that the system produces the error output can be decreased by adopting either the parallel or the shared architectures. The increased reliability is attributed to the input data diversity from two data sources (i.e., sensors), and hence we can consider both architectures to improve the reliability.

In terms of throughput, the previous study showed that the parallel architecture is preferable to the shared one, especially when the processing rate μ_1, μ_2, and μ are large [13]. However, if the response time and the energy consumption are more concerned in the applications, the shared architecture is considered a better option according to our results. For example, autonomous vehicles or robots may have stringent limitations on their computing resources and latency [17], and hence a multiple-input architecture in a parallel configuration may not be acceptable. The shared architecture, as a type of multi-input ML system exploiting input data diversity, can be considered an efficient design option that can improve reliability with lower performance overheads. On the other hand, the parallel architecture is considered another type of multi-input AIoT system that potentially exploits model diversity as well as input data diversity by employing different ML models. The resulting architecture is also called a double-model double-input (DMDI) system that can achieve higher output reliability than the shared architecture [7]. However, the reliability enhancement is fundamentally attributed to the model diversity between the two ML models, which requires a dedicated reliability assessment using diversity metrics [8]. If the performance and energy overheads estimated by our models are acceptable, the parallel architecture remains an option, given its higher throughput and potential reliability improvements.

8 Conclusion

In this paper, we proposed queueing models for a multi-input AIoT system under two types of processing schemes: the parallel and the shared architectures. To compare the architectures, we defined the performance metrics on the queueing models, which are throughput, response time, and energy consumption. The numerical and simulation results showed that the shared architecture had an advantage in the response time and energy consumption, while the parallel architecture achieved a higher throughput. We also showed that the proposed models could make more precise performance estimation than the conventional simple queueing models, and hence would be useful in the design of reliable AIoT systems.

We focused on the basic components of a multi-input AIoT system that has two input sources in this work. As AIoT can be configured with more inputs and more ML modules, in future work, we consider extending the models for parallel and shared architectures with more inputs and more modules.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported in part by the JSPS KAKENHI under Grant 18K18006 and Grant 22K17871.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[2] Huang, X., Kwiatkowska, M., Wang, S., & Wu, M. (2017, July). Safety verification of deep neural networks. In *International conference on computer aided verification* (pp. 3-29). Cham: Springer International Publishing. [CrossRef]

[3] Wong, E., & Kolter, Z. (2018, July). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning* (pp. 5286-5295). PMLR.

[4] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering, 48*(1), 1–36. [CrossRef]

[5] Tian, Y., Pei, K., Jana, S., & Ray, B. (2018, May). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering* (pp. 303-314). [CrossRef]

[6] Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., & Tonella, P. (2020). Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering, 25*(6), 5193-5254. [CrossRef]

[7] Machida, F. (2019, June). N-version machine learning models for safety critical systems. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* (pp. 48-51). IEEE. [CrossRef]

[8] Machida, F. (2023). Using diversities to model the reliability of two-version machine learning systems. *IEEE Transactions on Emerging Topics in Computing, 12*(3), 810-825. [CrossRef]

[9] Xie, M., Dai, Y. S., & Poh, K. L. (2004). *Computing system reliability: models and analysis*. Boston, MA: Springer Us. [CrossRef]

[10] Machida, F. (2019, December). On the diversity of machine learning models for system reliability. In *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 276-27609). IEEE. [CrossRef]

[11] Takahashi, M., Machida, F., & Wen, Q. (2022, November). How data diversification benefits the reliability of three-version image classification systems. In *2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 34-42). IEEE. [CrossRef]

[12] Latifi, S., Zamirai, B., & Mahlke, S. (2020, June). Polygraphmr: Enhancing the reliability and dependability of cnns. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 99-112). IEEE. [CrossRef]

[13] Makino, Y., Phung-Duc, T., & Machida, F. (2021, June). A queueing analysis of multi-model multi-input machine learning systems. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* (pp. 141-149). IEEE. [CrossRef]

[14] Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012, July). Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *2012 IEEE symposium on computers and communications (ISCC)* (pp. 000059-000066). IEEE. [CrossRef]

[15] Powers, N., Alling, A., Osolinsky, K., Soyata, T., Zhu, M., Wang, H., Ba, H., Heinzelman, W., Shi, J., & Kwon, M. (2015). The cloudlet accelerator: Bringing mobile-cloud face recognition into real-time. In *2015 IEEE Globecom Workshops (GC Wkshps)* (pp. 1–7). IEEE. [CrossRef]

[16] Collin, A., Siddiqi, A., Imanishi, Y., Rebentisch, E., Tanimichi, T., & de Weck, O. L. (2020). Autonomous driving systems hardware and software architecture exploration: optimizing latency and cost under safety constraints. *Systems Engineering, 23*(3), 327-337. [CrossRef]

[17] Zhao, H., Hari, S. K. S., Tsai, T., Sullivan, M. B., Keckler, S. W., & Zhao, J. (2021, October). Suraksha: A framework to analyze the safety implications of perception design choices in avs. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)* (pp. 434-445). IEEE. [CrossRef]

[18] Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications, 111*(4), 2287-2310. [CrossRef]

[19] Nishihara, R., Moritz, P., Wang, S., Tumanov, A., Paul, W., Schleier-Smith, J., Liaw, R., Niknami, M., Jordan, M. I., & Stoica, I. (2017). Real-time machine learning: The missing pieces. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (pp. 106–110). Association for Computing Machinery. [CrossRef]

[20] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017, July). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3296-3297). IEEE. [CrossRef]

[21] Trivedi, K. S., & Bobbio, A. (2017). *Reliability and availability engineering: Modeling, analysis, and applications*. Cambridge University Press.

[22] Latouche, G., & Ramaswami, V. (1999). *Introduction to matrix analytic methods in stochastic modeling*. Society for Industrial and Applied Mathematics.

[23] Phung-Duc, T., Akutsu, K., Kawanishi, K. I., Salameh, O., & Wittevrongel, S. (2022). Queueing models for cognitive wireless networks with sensing time of secondary users. *Annals of Operations Research, 310*(2), 641-660. [CrossRef]

[24] Akutsu, K., Phung-Duc, T., Lai, Y. C., & Lin, Y. D. (2022). Analyzing vertical and horizontal offloading in federated cloud and edge computing systems. *Telecommunication Systems, 79*(3), 447-459. [CrossRef]

[25] Wakigami, K., Machida, F., & Phung-Duc, T. (2024). Empirical architecture comparison of two-input machine learning systems for vision tasks. *Formal Aspects of Computing, 36*(4), 1-19. [CrossRef]

Appendix

A Definition of infinitesimal generators

We provide the complete definitions of block matrices used in the infinitesimal generators Q_P and Q_S presented in Section 4.2 and 4.3, respectively. In the following, the element in the ith row from the top and

jth column from the left of a block matrix is called the (i, j) element of that matrix. For instance, the (i, j) element of a block matrix A is denoted as $(A)_{i,j}$.

A.1 Parallel architecture model

In this section, we describe each of the block matrices used in the infinitesimal generator Q_P. First, B_0 is a $(2K + 1)$-order square matrix that represents the transition of states such as the number of type 2 jobs or states of module and comparison processing when there are no type 1 jobs in the system. Each element $(B_0)_{i,j}$ is defined as follows.

$$(B_0)_{i,j} = \begin{cases} \lambda_2 & i = \{2, ..., K, K + 2, ..., 2K\}, j = i + 1, \\ \lambda_2 & i = 1, j = K + 2, \\ \mu_2 & i = \{K + 2, ..., 2K + 1\}, j = i - K, \\ \Phi^0_{i,j} & i = j, \\ 0 & (otherwise), \end{cases}$$

where $\Phi^0_{i,j} = -\left(\sum_{j\neq i} B_{0,i,j} + \lambda_1\right)$.

B_1 is a $(5K + 1)$-order square matrix that represents the transition of states when there is one job of type 1 in the system. Each element $(B_1)_{i,j}$ is defined as follows.

$$(B_1)_{i,j} = \begin{cases} \lambda_2 & i \in I_1, j = i + 1, \\ \lambda_2 & (i, j) = (1, K + 3), (2, 3K + 3), \\ & (2K + 3, 4K + 3), \\ \mu_2 & i = \{K + 4, ..., 2K + 2\}, \\ & j = i + 3K - 1, \\ \mu_2 & i = \{3K + 3, ..., 4K + 2\}, j = i - 3K, \\ \mu_2 & i = \{4K + 3, ..., 5K + 1\}, \\ & j = i - 2K + 1, \\ \mu_2 & i = K + 3, j = 2K + 3, \\ \mu_1 & i = \{4, ..., K + 2\}, j = i + 4K - 1, \\ \mu_1 & i = \{3K + 3, ...4K + 2\}, j = i - 2K, \\ \mu_1 & i = 3, j = 2K + 3, \\ \Phi^1_{i,j} & i = j, \\ 0 & (otherwise), \end{cases}$$

where $\Phi^1_{i,j} = -\left(\sum_{j=1}^{2K+1} A_{1,i,j} \sum_{j\neq i} B_{1,i,j} + \lambda_1\right)$, and I_1 is the set defined as follows.

$$I_1 := \{3, ..., K + 1, K + 3, ..., 2K + 1, 2K + 4, ..., 3K + 1, \\ 3K + 3, ..., 4K + 1, 4K + 3, ..., 5K\}.$$

B_2 is a $7K$-order square matrix that represents the transition of states when there are two or more jobs of type 1 in the system. Each element $(B_2)_{i,j}$ is defined as follows.

$$(B_2)_{i,j} = \begin{cases} \lambda_2 & i \in I_2, j = i + 1, \\ \lambda_2 & (i, j) = (1, K + 3), (2, 3K + 3), \\ \lambda_2 & (i, j) = (2K + 3, 5K + 3), \\ & (4K + 3, 6K + 2), \\ \mu_2 & i = \{K + 4, ..., 2K + 2\}, j = i + 5K - 2, \\ \mu_2 & i = \{3K + 3, ..., 4K + 2\}, j = i - 3K, \\ \mu_2 & i = \{5K + 3, ..., 6K + 1\}, j = i - 3K + 1, \\ \mu_2 & i = \{6K + 2, ..., 7K\}, j = i - 2K + 2, \\ \mu_2 & i = K + 3, j = 4K + 3, \\ \mu_1 & i = \{4, ..., K + 2\}, j = i + 6K - 2, \\ \mu_1 & i = \{3K + 3, ...5K + 2\}, j = i - 2K, \\ \mu_1 & i = 3, j = 4K + 3, \\ \Phi^2_{i,j} & i = j, \\ 0 & (otherwise), \end{cases}$$

where $\Phi^2_{i,j} = -\left(\sum_{j=1}^{5K+1} A_{2,i,j} \sum_{j\neq i} B_{2,i,j} + \lambda_1\right)$, and I_2 is the set defined as follows.

$$I_2 := \{3, ..., K + 1, K + 3, ..., 2K + 1, 2K + 4, ..., 3K + 1, \\ 3K + 3, ..., 4K + 1, 4K + 4, ..., 5K + 1, 5K + 3, ..., 6K, \\ 6K + 2, ..., 7K - 1\}.$$

C_0 is a matrix of size $(2K+1)\times(5K+1)$ that represents the transition of the number of jobs of type 1 from 0 to 1. Each element $(C_0)_{i,j}$ is defined as follows.

$$(C_0)_{i,j} = \begin{cases} \lambda_1 & i = \{1, ..., K + 1\}, j = i + 1, \\ \lambda_1 & i = \{K + 2, ..., 2K + 1\}, j = i + 2K + 1, \\ 0 & (otherwise). \end{cases}$$

C_1 is a matrix of size $(5K + 1) \times (7K)$ that represents the transition of the number of jobs of type 1 from 1 to 2. Each element $(C_1)_{i,j}$ is defined as follows.

$$(C_1)_{i,j} = \begin{cases} \lambda_1 & i = \{1, ..., 2K + 2, 3K + 3, ..., 4K + 2\}, \\ & j = i, \\ \lambda_1 & i = \{2K + 3, ..., 3K + 2\}, j = i + 2K, \\ \lambda_1 & i = \{4K + 2, ..., 5K + 1\}, j = i + 2K - 1, \\ 0 & (otherwise). \end{cases}$$

C_2 is a $7K$-order square matrix that represents the transition of the number of in-system jobs of type 1 from i to $i + 1$ $(i \geq 2)$, defined as

$$C_2 = diag(\lambda_1, ..., \lambda_1).$$

A_1 is a matrix of size $(5K+1)\times(2K+1)$ that represents the transition of the number of jobs of type 1 from 1 to 0. Each element $(A_1)_{i,j}$ is defined as follows.

$$(A_1)_{i,j} = \begin{cases} \mu & i = \{2K+3, ..., 3K+2\}, j = i-2K-2, \\ \mu & i = \{4K+3, ..., 5K+1\}, j = i-3K-1, \\ 0 & (otherwise). \end{cases}$$

A_2 is a matrix of size $(7K) \times (5K+1)$ that represents the transition of the number of jobs of type 1 from 2 to 1. Each element $(A_2)_{i,j}$ is defined as follows.

$$(A_2)_{i,j} = \begin{cases} \mu & i = \{2K+5, ..., 3K+2\}, j = i+2K-2, \\ \mu & i = \{4K+3, ..., 5K+2\}, j = i-4K-1, \\ \mu & i = \{5K+3, ..., 6K+1\}, j = i-4K, \\ \mu & i = \{6K+2, ..., 7K\}, j = i-3K+1, \\ \mu & (i,j) = (2K+3, 1), (2K+4, 2K+3), \\ 0 & (otherwise). \end{cases}$$

A_3 is a $7K$-order square matrix that represents the transition of the number of Type 1 jobs from i to $i-1$ $(i \geq 3)$. Each element $(A_3)_{i,j}$ is defined as follows.

$$(A_3)_{i,j} = \begin{cases} \mu & i = \{2K+5, ..., 3K+2\}, j = i+4K-3, \\ \mu & i = \{4K+3, ..., 5K+2\}, j = i-4K-1, \\ \mu & i = \{5K+3, ..., 6K+1\}, j = i-4K, \\ \mu & i = \{6K+2, ..., 7K\}, j = i-3K+1, \\ \mu & (i,j) = (2K+3, 1), (2K+4, 4K+3), \\ 0 & (otherwise). \end{cases}$$

A.2 Shared architecture model

Next, we describe each of the block matrices used in the infinitesimal generator Q_S. First, A_0 is a 9th-order square matrix that represents the transition of the number of jobs in the buffer from i to $i+1$ $(i \geq 1)$, defined as follows

$$A_0 = diag(\lambda_1+\lambda_2, ..., \lambda_1+\lambda_2).$$

B_0 is a 15th-order square matrix that represents the transition of states when the number of jobs in the buffer is zero. Each element $(B_0)_{i,j}$ is defined as follows.

$$\begin{aligned} &(B_0)_{1,3} = (B_0)_{5,7} = (B_0)_{8,10} = (B_0)_{12,14} = \lambda_1, \\ &(B_0)_{1,6} = (B_0)_{2,4} = (B_0)_{8,13} = (B_0)_{9,11} = \lambda_2, \\ &(B_0)_{3,2} = (B_0)_{7,8} = (B_0)_{10,9} = (B_0)_{14,15} = \mu_1, \\ &(B_0)_{4,8} = (B_0)_{6,5} = (B_0)_{11,15} = (B_0)_{13,12} = \mu_2, \\ &(B_0)_{8,1} = (B_0)_{9,2} = \cdots = (B_0)_{15,8} = \mu. \end{aligned}$$

Denoting $I := \{1, 2, ..., 15\}$, $J := \{1, 2, ..., 9\}, i \in I$, we define the diagonal components of B_0 as follows.

$$(B_0)_{i,i} = -\left(\sum_{j \in I \setminus \{i\}} (B_0)_{i,j} + \sum_{j \in J} (C_0)_{i,j} \right).$$

The other elements of $(B_0)_{i,j}$ are all 0.

C_0 is a matrix of size 15×9 that represents the transition of the number of jobs in the buffer from 0 to 1. Each element $(C_0)_{i,j}$ is defined as follows.

$$\begin{aligned} &(C_0)_{3,1} = (C_0)_{4,2} = (C_0)_{6,3} = (C_0)_{7,4} = (C_0)_{10,5} \\ &= (C_0)_{11,6} = (C_0)_{13,7} = (C_0)_{14,8} = (C_0)_{15,9} = \lambda_1 + \lambda_2. \end{aligned}$$

The other elements of $(C_0)_{i,j}$ are all 0.

B_k $(k \geq 1)$ is a matrix of size 9×15 that represents the transition of the number of jobs in the buffer from k to 0. Each element $(B_k)_{i,j}$ is defined as follows.

$$\begin{aligned} (B_k)_{1,2} &= (B_k)_{5,9} = \mu_1(\tilde{\lambda_1})^k, \\ (B_k)_{1,4} &= (B_k)_{5,11} = \mu_1(\tilde{\lambda_1})^{k-1}\tilde{\lambda_2}, \\ (B_k)_{3,7} &= (B_k)_{7,14} = \mu_2(\tilde{\lambda_2})^{k-1}\tilde{\lambda_1}, \\ (B_k)_{3,5} &= (B_k)_{7,12} = \mu_2(\tilde{\lambda_2})^k. \end{aligned}$$

where $\tilde{\lambda_1} = \frac{\lambda_1}{\lambda_1+\lambda_2}$ and $\tilde{\lambda_2} = \frac{\lambda_2}{\lambda_1+\lambda_2}$.

Also, we define the following elements $(B_1)_{i,j}$.

$$\begin{aligned} &(B_1)_{4,10} = \mu_1\tilde{\lambda_1}, \ (B_1)_{4,13} = \mu_1\tilde{\lambda_2}, \\ &(B_1)_{2,10} = \mu_2\tilde{\lambda_1}, \ (B_1)_{2,13} = \mu_2\tilde{\lambda_2}, \\ &(B_1)_{9,3} = \mu\tilde{\lambda_1}, \ (B_1)_{9,6} = \mu\tilde{\lambda_2}. \end{aligned}$$

The other elements of $(B_k)_{i,j}$ $(k \geq 1)$ are all 0.

A_k $(k \geq 2)$ is a 9th-order square matrix that represents the transition of the number of jobs in the buffer decreasing by $k-1$. Each element $(A_k)_{i,j}$ is defined as follows.

$$\begin{aligned} (A_k)_{1,2} &= (A_k)_{5,6} = \mu_1(\tilde{\lambda_1})^{k-2}\tilde{\lambda_2}, \\ (A_k)_{3,4} &= (A_k)_{7,8} = \mu_2(\tilde{\lambda_2})^{k-2}\tilde{\lambda_1}. \end{aligned}$$

Also, we define the following elements $(A_2)_{i,j}$.

$$\begin{aligned} &(A_2)_{4,5} = \mu_1\tilde{\lambda_1}, \ (A_2)_{4,7} = \mu_1\tilde{\lambda_2}, \ (A_2)_{2,5} = \mu_2\tilde{\lambda_1}, \\ &(A_2)_{2,7} = \mu_2\tilde{\lambda_2}, \ (A_2)_{9,1} = \mu\tilde{\lambda_1}, \ (A_2)_{9,3} = \mu\tilde{\lambda_2}. \end{aligned}$$

The other elements of $(A_k)_{i,j}$ $(k \geq 2)$ are all 0.

Shoma Nishio received Batchelor's Degree in Policy and Planning Sciences from the University of Tsukuba in March 2022. His research interest is Queueing Theory and its application. He is currently with DIGITAL VORN Co., Ltd.

Yuta Makino received Batchelor's and Master's Degrees in Policy and Planning Sciences from the University of Tsukuba in March 2019 and March 2021, respectively. His research interest is Queueing Theory and its application. He is currently with NTT EAST.

Tuan Phung-Duc is a Professor at Institute of Systems and Information Engineering, University of Tsukuba. He received a Ph.D. in Informatics from Kyoto University in 2011. He was on the Editorial Board of the Journal of the Operations Research Society of Japan. Currently, he is serving as a Co-Editor-in-Chief of Queueing Models and Service Management and on editorial board of several other journals. He served as Guest Editor of four special issues of Annals of Operations Research. He was the Chairman of the 10th International Workshop on Retrial Queues (WRQ2014) and the TPC co-chair of the 23rd International Conference on Analytical, and Stochastic Modelling Techniques and Applications (ASMTA2016), TPC co-chair of The 13th and 14th International Conference on Queueing Theory and Network Applications (QTNA2018, QTNA2019), General co-chair of EAI VALUETOOLS2020 - 13th EAI International Conference on Performance Evaluation Methodologies and Tools, and General chair of ASMTA/EPEW: The 26th International Conference on Analytical and Stochastic Modelling Techniques and Applications / 17th European Performance Engineering Workshop. Dr. Phung-Duc received the Research Encourage Award from The Operations Research Society of Japan in 2013. His research interests include Applied Probability, Stochastic Models and their Applications in Performance Analysis of Telecommunication and Service Systems.

Fumio Machida is an associate professor of Computer Science Department in University of Tsukuba. Before the current position, he was a principal researcher at NEC Corporation. He was a visiting scholar in the Department of Electrical and Computer Engineering at Duke University in 2010. He was a recipient of the young scientists' prize of Japan in 2014. His research interests include modeling and analysis of system dependability, software aging and rejuvenation, and reliability of machine learning systems. He is a senior member of the IEEE and the member of ACM.