Performance Analysis and Evaluation of Ternary Optical Computer Based on Asynchronous Multiple Vacations

Wang Xianchao (wxcdx@126.com)
Fuyang Normal University
Wang Xianchuan
Fuyang Normal University
Zhang Jie
Fuyang Normal University
Ling Man
Fuyang Normal University
Hou Dayou
Fuyang Normal University
Song Kai
East China JiaoTong University

Research Article

Keywords: task scheduling, processor allocation, tandem queueing, quasi birth and death process, rate matrix

Posted Date: October 21st, 2021

DOI: https://doi.org/10.21203/rs.3.rs-995288/v1

Performance analysis and evaluation of ternary optical computer based on asynchronous multiple vacations

Wang Xianchao[1*], Wang Xianchuan[1*], Zhang Jie[2*], Ling Man[1], Hou Dayou[1], Song Kai[3]

[1](*School of Computer and Information Engineering, Fuyang Normal University, Fuyang Anhui* 236037)

[2](*School of Mathematics and Statistics, Fuyang Normal University, Fuyang Anhui* 236037)

[3](*School of Information Engineering, East China Jiaotong University, Nanchang* 330013)

* Correspondence: wxcdx@126.com, xch_wang@126.com, zjp562@126.com

Abstract Ternary optical computer(TOC) has become a research hotspot in the field because of the advantages such as inherent parallelism, numerous trits, low power consumption, extendibility, bitwise allocability and dynamical bitwise reconfigurability. Meanwhile, its performance evaluation attracts more and more attentions from potential users and researchers. To model its computing ecology more accurately, this paper first builds a three-staged TOC service model by introducing asynchronous multi-vacations and tandem queueing, and then proposes a task scheduling algorithm and an optical processor allocation algorithm with asynchronous vacations of some small optical processors after dividing equally the entire optical processor into several small optical processors which can be used independently. At the same time, the analytical model was established to obtain important performance indicators such as response time, the number of tasks and utilization of optical processor, based on M/M/1 and M/M/n queuing system with asynchronous multi-vacations. In addition, relevant numerical simulation experiments are conducted. The results illustrate that the number of small optical processors, vacation rate and the number of small optical processors allowed to be on vacation have important effects on the system performance. Compared with synchronous vacation, asynchronous vacation not only ensures the system to obtain better maintenance but also improves the system performance to some degree.

Key words task scheduling; processor allocation; tandem queueing; quasi birth and death process; rate matrix

1. Introduction

With the development of science and technology, especially artificial intelligence, people express higher requirements for computing power. scientists have made the electrons travel a shorter distance in a shorter time for increasing the speed of a single CPU by miniaturizing electronic components to very small micron sizes. At the same time, they also developed the cloud computing platform such as the amazon cloud, Google cloud, Ali cloud and a new batch of supercomputers such as Summit, Sierra, Sunway Taihu Light, to improve the computing power of single computer. However, these technologies and strategies have not fundamentally changed the inherent bottlenecks of the electronic computer such as high energy consumption, low bandwidth and high latency. Therefore, many researchers have been exploring new types of computers, such as DNA computing, optical computing and quantum computing.

Among these novel computers, optical computers have been preferring in recent decades because of their low energy consumption, high bandwidth, free interconnect in three-dimensional space and high parallelism. For example, in the early 1970s, Heinz et al. mathematically investigated and studied matrix multiplication utilizing coherent optical correlation techniques and optical analog methods [1]. Especially, in 2010, Ambs summarized the 60-year adventure of optical computing [2]. In 2017, Zangeneh-Nejad et al. designed and realized a reconfigurable and highly miniaturized analog optical differentiator by using a half-wavelength plasma graphene film based on graphene-supported plasma wave characteristics [3]. In 2018, Rashed et al. developed an optoelectronic converter to realize all-optical computing operations based on nonlinear metamaterials [4]. In 2019, Zhou et al. implemented a multi-layer spatial optical differentiator by designing a deep neural network which can predict the reflection coefficient of the 12-layer film [5]. Consequently, optical computing remains a common focus.

Jin pioneered a ternary optical computer (TOC), which expresses information in ternary optical states, including no intensity light (NIL), vertically polarized light (VPL) and horizontally polarized light (HPL). A lot of significant achievements have been obtained. For instance, in hardware, Jin et al. proposed the TOC principle and its structure in 2003 [6, 7]. In 2008, Yan et al. put forward the decrease-radix design principle(DRDP) [8], which makes the construction of TOC processor normative and operable. Meanwhile, the processor constructed according to DRDP is dynamically reconfigurable. Multi-generation TOC hardware platforms have been constructed according to this theory. In 2017, the TOC platform SD16 with 192 trits was successfully constructed. In 2018, Jin et al. built an optical processor(OP) with 1152 trits by combining six SD16s [9]. To solve the carry delay problem, TOC adder [10-14] and multiplier [15, 16] were designed and realized based on MSD digital system. Literature [17] designed and implemented a positive/negative value judger for MSD data for further improving the three-valued optical processor. Literature [18] proposed the structure and theory of dual-space memory to solve the problem of data transmission in TOC. In software, Literature [19] presented the module structure and inter-module communication protocol of the TOC monitoring system, and [20, 21] deeply discussed its task management system and trit resources. [22-24] realized the programming application by designing and implementing of the operation-data file SZG in TOC; [25] implemented the seamless combination of C language and TOC by successfully transplanted the former to the latter. Meanwhile, the MSD digital multiplication program [16] and MPI programming technology [26] were also achieved on TOC. In numerical calculation, parallel carry free addition [12], multiplication [15, 16], division [27] and vector matrix multiplication [12] were realized on TOC. Literature [28] and [29] achieved fast Fourier transformation (FFT) and discrete Fourier transformation (DFT) on TOC, respectively. [30] implemented the algorithm to solve higher-order derivative by configuring the multipliers and adders on the TOC platform. [31] used the cellular automata on TOC to simulate a three-lane traffic flow. In addition, [32] designed and implemented a parallel artificial bee colony algorithm on TOC. These have made the research on TOC from theory to numerical application. It can be seen that some breakthroughs have been made.

However, there are few reports on another important research direction of TOC -- performance analysis and evaluation. Therefore, this paper analyzes the QoS(quality of service) performance indicators of TOC, proposes a three-staged service model of TOC, and analyzes and evaluates the performance of TOC based on the queuing theory. We focus on making the following contributions in this paper:

- This paper first builds a three-staged service model of TOC by connecting three queues -- receiving queue, scheduling queue and transmitting queue -- in series based on vacation queuing and tandem queuing after illustrating the computing paradigm of a TOC and its primary modules.
- Based on the equal partition strategy of OP [9], we propose a TOC task scheduling algorithm with asynchronous vacations of some small OPs(SOPs), a processor allocation algorithm and a processor recovery algorithm.
- We construct some analytical al models that evaluates the important performance indicators of TOC, including response time, number of tasks, and OP utilization. In particular, we build the analytical al model to solve the performance indicators of the second stage by using the quasi-birth and death process and rate matrix.
- We fully demonstrate the influence on TOC performance of different parameters such as the number of SOPs, the number of SOPs allowed to be on vacations and vacation rate by numerically simulating, and make an analysis of the reasons for the results.
- Finally, we illustrate the influence on TOC performance of different service models with vacations, including three-staged service model with asynchronous vacation (TSSMAV), four-staged service model with synchronous vacation (FSSMSV) and four-staged service model with asynchronous vacation (FSSMAV), by numerically simulating.

The rest of the paper is organized as follows. Section 2 chiefly describe the background and motivation of this

paper, in particular, the performance analysis based on different queueing systems. Section 3 introduce the three-staged service model with vacation queueing after presenting the computing paradigm of TOC. In Section 4, we primarily present a TOC task scheduling algorithm with some SOPs allowed to be on asynchronous vacation and a processor allocation algorithm. In Section 5, we focus on the construction of the performance analysis and evaluation model. Especially, we build the analytical al model to solve the performance indicators of the second stage, including response time, number of tasks, and OP utilization, by using the quasi-birth and death process and rate matrix. Section 6 and Section7 demonstrate the influence on TOC performance of different parameters and three service models with vacations by numerically simulating. Finally, Section 8 gives some concluding remarks and possible future research.

2 Background and motivation

In the early stage of TOC development, its performance has been concerned by researchers. For example, in 2010, Liu et al. designed and implemented a subsystem for measurement of the response time of optical processor in TOC [33]. However, it can only measure the response time of the computing component, they ignored the transmission time of the data from client to TOC, the one of computing results from TOC to the client and other performance indicators such as optical processor utilization and so on. In literature [34], a four-staged TOC service model was established for the first time, and the response time of TOC was analyzed and evaluated based on M/M/1 queuing system. It concluded that the speed of network is the bottleneck of TOC performance. [35, 36] also built a four-staged service model of TOC. However, the model is based on the complex queuing system composed of M/M/1, M/M/n, M^X/M/1 and M/M^B/1. They made a comparative analysis of the response times under the immediate scheduling strategy and the computing accomplished scheduling strategy. The results show that the latter is obviously superior to the former. Nevertheless, they did not consider the possible need for maintenance of TOC. To better model the computing ecology of TOC, [7] analyzed and evaluated the performance of TOC based on synchronous multi-vacations. And the results showed that the number of SOPs uniformly partitioned and vacation rate have a great impact on the system performance.

In 2018, a TOC with 1152 trits was built by combining six optical processors (each with 192 trits). It not only shows that TOC has a very good scalability, but also shows that the assumption of synchronous vacation of all SOPs in literature [7] is not enough to accurately describe the computing ecology of TOC and leads to the waste of processor resources. On the other hand, asynchronous vacation of partial SOPs is more suitable for the practical application of TOC. Consequently, this paper intends to analyze and evaluate the performance of TOC by taking asynchronous multi-vacations of some SOPS into consideration.

3 Three-staged service model of ternary optical computer

3.1 Computing paradigm of ternary optical computer

As can be seen in Fig. 1, the computing paradigm of TOC is Client/Server. The function of Client mainly includes two aspects. On the one hand, it submits the operation request to the Server. In Literature [9, 19, 34-36], Client resolves the operation into binary three-valued logical operations needed to implement the operation when users click on "Submit" button. Meanwhile, it obtains the number of logical operations, the computation amount of each logical operation and the total computation amount and send the operation request to the Server after transforming operands input by user into internal communication codes. Ant then, the preprocessing module in Server changes the internal communication codes into internal control codes which is used to implement optical computing. In this paper, the Client can be also used to improve the user experience and system performance in addition to providing input interface for users to enter operation and data. In other words, the preprocessing module

is deleted, as shown in Fig. 1. Thus, the operands in operation request are not in internal communication code but in internal control code. On the other hand, the Client displays the results sent by the Transmitter in the Server when it receives them.

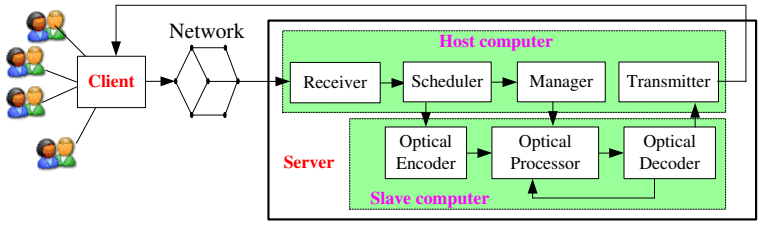

Fig. 1 The computing paradigm of a TOC and its primary modules.

Server is made up of Host computer(HC) and Slave computer(SC), and the HC mainly consists of four modules, including Receiver(R), Scheduler(S), Manager(M) and Transmitter(T). Receiver receives the task, i.e. operation request sent by the Client; Scheduler schedules the tasks according to certain strategy; Manager primarily allocates and recovers the SOPs of TOC; and Transmitter sends the operation results to the corresponding Client. In addition, the SC mainly consists of three modules, including optical encoder (OE), OP and its reconfigurable component, and optical decoder (OD). These modules coordinate with each other and organically form a whole TOC system to achieve the user's calculation.

3.2 Three-staged service model of ternary optical computer

Literature [9, 34-36] all constructed a four-staged TOC service model with vacations. According to the description in Subsection 2.1, this paper has built a three-staged TOC service model with vacations, as shown in Fig. 2. It can be seen that the model is connected from Stage 1 to Stage 3 in series. Moreover, each stage has its own queue. They are the receiving queue (RQ), scheduling queue (SQ) and transmitting queue (TQ) in turn. We assume that all queues are blocked request delay and the queuing rules of all queues are first-come-first-served (FCFS). The main function of each stage is as follows.

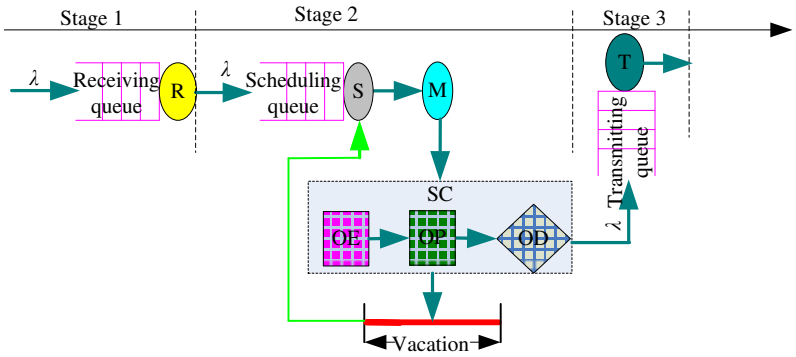

Fig. 2 The three-staged TOC service model with vacations.

- In Stage 1, operation requests submitted by users arrives to the Receiver R at an average arrival rate λ. R puts them into the RQ by the FCFS strategy. And then, it takes out the operation requests in the receiving queue RQ in turn when it is not empty, and sends them to the Scheduler S. Thus, the operation requests become tasks in TOC.
- Stage 2 consists of more functional modules, including the Scheduler S, Manager M and SC. The Scheduler S inserts the received tasks into the scheduling queue SQ in turn, and then schedules the tasks in SQ by some scheduling strategy. At the same time, it sends them to the optical encoder OE in SC. The Manager M allots a SOP to each of the just-scheduled tasks, allocates trit resources for each binary three-valued logic operation needed of each task. In the meantime, it finds the reconfiguration codes of the binary and three-valued logic

operations and sends the allocation information and reconfiguration information to the OP in the SC. The OE converts the electrical signals into optical signals, i.e. HPL, VPL and NIL, after receiving the data represented in internal control code. After the reconfiguration unit in the OP finishes the OP reconfiguration in full parallel by use of the reconfiguration codes, the OP can implement the optical computing when the optical signals generated by the OE pass through it. The optical decoder OD decodes the operation results, i.e. converts the optical signals into the electrical ones in internal control code, and then judges whether they are the final operation results. If so, they are sent to the Transmitter T in the HC; otherwise, they are sent to the encoder to participate in the next operation. On the other hand, the SC will start a vacation with random time according to a certain strategy for the SOP which just finishes a task and send the "vacation" signal to the Scheduler S if there is no task to calculate. It sends the "vacation is over" signal to S when the vacation is over. S schedule a task in SQ once again if there is a task in SQ; otherwise, the SOP goes on its next vacation.

- The Transmitter T in Stage 3 puts the received operation results into the TQ in turn, and then it takes out the operation results in TQ and sends them to the corresponding Client according to FCFS policy.

The three stages form a tandem queuing model. Compared with Literature [9, 34-36], the computing paradigm of TOC proposed in this paper has deleted the preprocessing module in the Server and the service model has only three stages. Therefore, we assert that this model will result in a significant performance improvement over the previous service models.

4 Task scheduling and optical processor management algorithm with asynchronous vacations of some SOPs

Considering the unique characteristics of TOC optical processor, such as numerous trits, extendibility, bitwise allocability and dynamical bitwise reconfigurability and so on, this paper similarly divides the entire optical processor into several SOPs for standbys according to the equal partition strategy. We assume that there are N trits and divide the consecutive trit resource into n SOPs which can be used dependently. And each of them has $N_s=\lfloor N/n \rfloor$ trits. Under the asynchronous vacation strategy, it is assumed that the maximum of SOPs allowed to be on asynchronous vacations is c. Obviously, each SOP has three states, including "idle", "busy" and "vacation", denoted by "0", "1" and "2," respectively. The Scheduler S executes the task scheduling algorithm shown in Algorithm 1 and the Manager M fulfills the processor allocation algorithm and recovery algorithm, shown in Algorithms 2 and 3, respectively.

Algorithm 1 Task scheduling algorithm with asynchronous vacations of some SOPs

- Step 1: Initialize the system parameters. Set the scheduling queue SQ to null. Meanwhile, set the length L_{SQ} of SQ, the number d of SOPs in vacation state and the number N_{pring} of tasks being processed all to 0.
- Step 2: When a task arrives, S inserts it in SQ by FCFS policy, increases L_{SQ} by 1, and jump to Step 3.
- Step 3: Judge whether there is an idle SOP. In other words, judge whether N_{pring} is less than n-d. If so, jump to Step 4; otherwise, jump to Step 9.
- Step 4: Schedule a task, i.e. send it to the SC, increase N_{pring} by 1, decrease L_{SQ} by 1, send the relevant information on the task to the Manager M, and jump to Step 5.
- Step 5: Judge whether L_{SQ} is equal to zero. If so, jump to Step 9; otherwise, jump to Step 3.
- Step 6: Decrease N_{pring} by 1 and jump to Step 7 when S receives the "i finished" signal sent by the SC, which means that the task on the i-th SOP is finished.
- Step 7: Judge L_{SQ} is greater than zero. If so, jump to Step 3; otherwise, judge whether d is equal to c. If so, send "i0" signal to M; otherwise, increase d by 1 and send "i2" signal to M. Jump to Step 9.
- Step 8: Decrease d by 1 and jump to Step 7 when S receives the "Vacation over" signal.

- Step 9: The algorithm is over.

As can be seen from Algorithm 1, not until there is no idle SOP or no task in SQ does S stop scheduling task. M carries out the recovery of processor resource when a task is finished and L_{SQ} is equal to zero. There are two altenatives for the SOP that just finished the task: it starts a vacation, that is to say, its state "1" is change into "2" if d is less than c; it becomes idle if d is equal to c. In other words, only when a SOP finishes a task, there is no task to be scheduled and d is not equal to c can it can start a vacation.

Algorithm 2 Processor allocation algorithm

- Step 1: Initialize the system parameters. Set the states $S[0\ldots n\text{-}1]$ of all SOPs to zero. j=0(j is used to point to the SOP to be scheduled).
- Step 2: j=j mod n, and judge whether S[j] is equal to zero. If so, S[j]=1 and jump to Step 4; otherwise, jump to Step 3.
- Step 3: Increase j by 1 and jump to Step 2.
- Step 4: k=1.
- Step 5: Judge whether k is greater the number N_{Log} of binary three-valued logic operations needed by the task. If so, jump to Step 7; otherwise jump to Step 6.
- Step 6: $N_k = \left\lfloor \frac{C_k}{C} \times N_s \right\rfloor$, where C_k(k=1,2,...,N_{Log}) is the computation amount of the k-th logic operation, and C=$\sum_{k=1}^{N_{Log}} C_k$. Increase k by 1, jump to Step 5.
- Step 7: The algorithm is over.

It can be seen that the algorithm still uses the proportional allocation strategy to allocate the trit resources of the SOP as Literature [9, 34-36] do, focusing on the synchronous completion of all binary three-valued logic operations in the task.

Algorithm 3 Processor recovery algorithm

- Step 1: Obtain "i" and state "0" or "2" by splitting the information received from S and assign them to I and s, respectively.
- Step 2: S[I]=s. Judge whether s is equal to zero. If so, jump to Step 3; otherwise, send "2" to the SC.
- Step 3: The algorithm is over.

It can be seen that Algorithm 3 achieves the recovery of SOPs by setting the states to "0" or "2". In other words, it not only recoveries the idle SOPs but also recoveries the SOPs on vacations.

5 Analytical model for performance analysis and evaluation

To analyze and evaluate the performance of TOC, we choose some primary performance indicators such as response time, the number of task and the OP utilization in the TOC. They are denoted as T, R and U, respectively. In addition, the response time is defined as the elapsed time from the submission of a request to the TOC until the Client receives the final output, i.e. the time it is serviced. It can be obtained via the following formula when the system is in equilibrium.

$$T = \sum_1^3 T_i \tag{1}$$

Where T_i(i=1,2,3) denotes the mean service time of Stage 1~3 shown in Fig. 2 and the service time is the sum of waiting time and computing time. Similarly, R is the sum of mean of tasks in each stage. Namely,

$$R = \sum_1^3 R_i \tag{2}$$

Where R_i(i=1,2,3) represents the mean of tasks in Stage 1~3.

5.1 Analytical model for the Receiver

We can use a server to implement the function of the Receiver R, that is, to receive the operation request sent

by the user. For the sake of simplification, we model Stage 1 as an M/M/1 queuing system with single request arrival and a request buffer of infinite capacity. The specific model is described as follows.

- Assume that the arrival of the requests follows a Poisson distribution, that is to say, the arrival rate follows a negative exponential distribution with parameter λ.
- The service times of R for the operation requests are independent and identically distributed random variables that follow a negative exponential distribution with parameter μ_1, which denotes the service rate of R.
- Service mechanism for R is FCFS policy.
- Denote the mean transmission speed of network and the mean traffic of requests that are submitted to R as ξ and D, respectively.

Thus, the state transition diagram of the continuous time Markov chain (CTMC) for the number of requests in Stage 1 is shown in Fig. 3, where the state m means the request number in R and there are m-1 requests in RQ to be received.

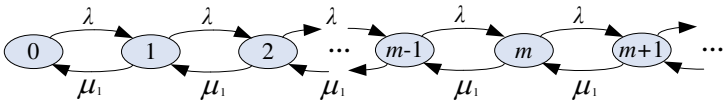

Fig. 3 State transition diagram for Stage 1 based on the M/M/1 queueing system.

Let $\rho_1 = \lambda/\mu_1$. [38-39] shows that R will reach an equilibrium state when $\rho_1 < 1$. Denote the probability of state m under its equilibrium as p_m(m=0,1,2,…) . According to Fig. 3, the steady-state equilibrium equation of the RQ system can be obtained as follows.

$$\begin{cases} \lambda p_0 = \mu_1 p_1, \\ (\lambda + \mu_1) p_m = \lambda p_{m-1} + \mu_1 p_{m+1}, \quad m \geq 1. \end{cases}$$

Then

$$p_m = \rho_1^m p_0, \ m \geq 1.$$

We can obtain the idle probability p_0 of the Receiver R, i.e. the probability of no request in R by use of the normalization equation $\sum_{m=0}^{\infty} p_m = 1$.

$$p_0 = 1 - \rho_1.$$

Then, the mean number of requests R_1 can be obtained

$$R_1 = \sum_{m=0}^{\infty} m p_m = \rho_1(1-\rho_1) \sum_{i=0}^{\infty} i \rho_1^{i-1} = \rho_1(1-\rho_1)\left(\frac{\rho_1}{1-\rho_1}\right)' = \frac{\rho_1}{1-\rho_1} = \frac{\lambda}{\mu_1 - \lambda} \tag{3}$$

We can obtain the mean service time T_1 of the RQ by Little's law [38-39], i.e. the service time = the number of requests in the system/the arrival rate of requests.

$$T_1 = \frac{R_1}{\lambda} = \frac{1}{\mu_1 - \lambda} \tag{4}$$

Where λ and μ_1 are the arrival rate of requests and the mean receiving rate in unit time, respectively. Obviously, $\mu_1 = \frac{\xi}{D}$. We can obtain the following formulas of R_1 and T_1 by substituting it into (3) and (4).

$$R_1 = \frac{\lambda}{\frac{\xi}{D} - \lambda}, T_1 = \frac{1}{\frac{\xi}{D} - \lambda} \tag{5}$$

5.2 Analytical model for Stage 2

The output of R is also a Poisson process with the parameter λ while it reaches equilibrium on the basis of Burke's theorem [38-39]. In other words, the arrival rate of tasks to Stage 2 is still λ. As described in Section 3, the whole OP of the TOC is uniformly divided into n SOPs. Therefore, we model Stage 2 as an M/M/n queueing system. Meanwhile, multi-vacations of partial SOPs are taken into consideration. The details of the model are described as follows.

- Denote the maximum of SOPs allowed to be on asynchronous vacations as c. Obviously, $1 \le c \le n$.
- Assume the mean computation amount, denoted as C, and $C=mD$, where m is a constant greater than 1.
- Denote the speed of the whole OP as σ. Assume that the computing times of TOC for the tasks are independent and identically distributed random variables which follow a negative exponential distribution with parameter μ_2, which means the service rate of the whole OP. That is to say, $\mu_2 = \frac{\sigma}{mD}$. And the service rate denoted as μ_{2s} of each SOP $\mu_{2s} = \sigma/(mnD)$.
- The vacations times of SOPs are independent and identically distributed random variables which follow a negative exponential distribution with parameter δ, which means the vacation rate of the SOPs.
- $L_v(t)$ and $V(t)$ indicate the number of tasks and the number of SOPs on vacation at time t in Stage 2 in equilibrium, respectively.
- The reconfiguration time is ignored because the full parallel reconfiguration takes very little time.
- The random variables μ_2, δ and λ are independent of each other.

Let $\rho_2 = \lambda/\mu_2$. Stage 2 will reach an balance state when $\rho_2 < 1$. According to the description above, at any time t, the number of SOPs on vacations $V(t) \leqslant c$, and there are $n-d$ idle SOPs for customers to be used. When $0 \le L_v(t) \le n-c$, the number d of SOPs on vacations reaches its maximum c, $L_v(t)$ SOPs are busy and the remainder are idle. When $n-c < L_v(t) \le n$, there are at least $n - L_v(t)$ SOPs that are on vacation and no idle SOP. When $L_v(t) > n$, there is no idle SOP. However, there may be SOPs which are on vacations.

Now we use the quasi birth-death (QBD) process [42-43] to obtain the task number R_2 and service time T_2 of Stage 2 and the utilization U of OP in equilibrium. Thus, $\{(L_v(t), V(t))\}$ constitutes a two-dimensional Markov process and a QBD process. And its state space Ω is as follows.

$$\Omega = \{(k,c) | 0 \le k \le n-c\} \cup \{(k,j) | n-c < k \le n, n-k \le j \le c\} \cup \{(k,j) | n \le k, 0 \le j \le c\}.$$

The states will change when a task is inserted into the scheduling queue SQ, a task is completed and a SOP vacation ends. For example, the state transition mechanism of n=6 and c=3 can be obtained by sorting the states by level, as shown in Fig. 4.

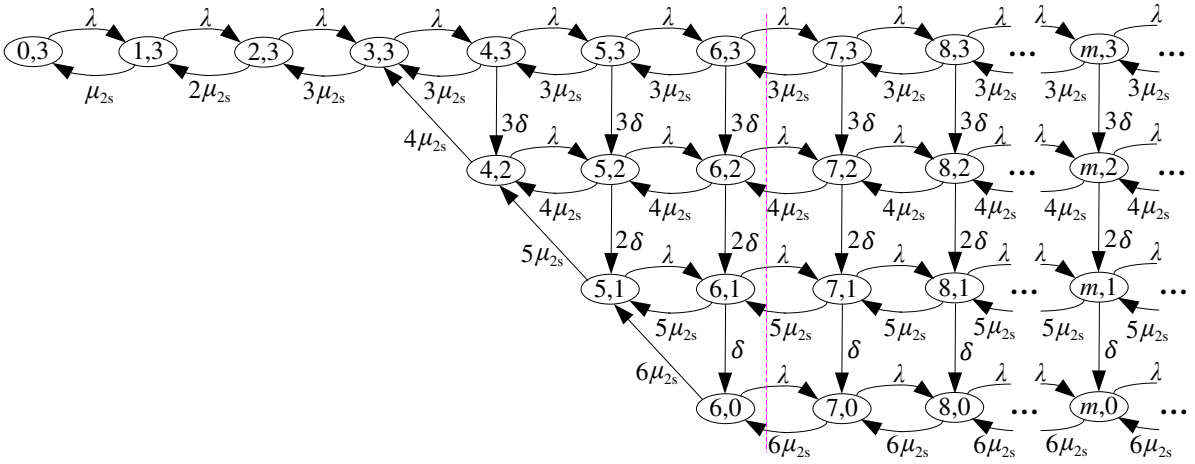

Fig. 4 State transition diagram based on M/M/6 queueing with three SOPs on asynchronous multi-vacations.

The four layers from bottom to top in Fig. 4 represent 0,1,2,3 SOP(s) on vacations, respectively. For example, the state (7,2) indicates that there are currently seven tasks, two SOPs on vacations and the other four busy SOPs in Stage 2. The state will transfer to three states, including (8, 2), (6, 1) and When a task enters the SQ, it will transfer to the state (8, 2) at the arrival rate of λ; the SC, i.e. TOC operates for four tasks at the service rate of $4\mu_{2s}$, so the state (7, 2) will transfer to state (6, 2) at the rate of $4\mu_{2s}$; the state (7, 2) will transfer to the state (6, 1) at the rate of 2δ if one of the vacations of the two SOPs is over. The state (5,1) indicates that there are currently five tasks in

Stage 2 and no task in SQ, with one SOP on vacation and the five busy SOPs. The state (5,1) will transfer to the state (4,2) at the rate of $5\mu_{2s}$ when a task is completed at the service rate of $5\mu_{2s}$.

We can obtain the following infinitesimal generative element matrix $\boldsymbol{G}$ of each state to the left of the dashed line in by sorting them from left to right and from top to bottom.

$\boldsymbol{G}=$

	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(4,2)	(5,3)	(5,2)	(5,1)	(6,3)	(6,2)	(6,1)	(6,0)
(0,3)	$-\lambda$	λ											
(1,3)	μ_{2s}	$-(\lambda+\mu_{2s})$	λ										
(2,3)		$2\mu_{2s}$	$-(\lambda+2\mu_{2s})$	λ									
(3,3)			$3\mu_{2s}$	$-(\lambda+3\mu_{2s})$	λ								
(4,3)				$3\mu_{2s}$	$-(\lambda+3\mu_{2s}+3\delta)$	3δ	λ						
(4,2)				$4\mu_{2s}$		$-(\lambda+4\mu_{2s})$		λ					
(5,3)					$3\mu_{2s}$		$-(\lambda+3\mu_{2s}+3\delta)$	3δ		λ			
(5,2)						$4\mu_{2s}$		$-(\lambda+4\mu_{2s}+2\delta)$	2δ		λ		
(5,1)						$5\mu_{2s}$			$-(\lambda+5\mu_{2s})$			λ	
(6,3)							$3\mu_{2s}$			$-(\lambda+3\mu_{2s}+3\delta)$	3δ		
(6,2)								$4\mu_{2s}$			$-(\lambda+4\mu_{2s}+2\delta)$	2δ	
(6,1)									$5\mu_{2s}$			$-(\lambda+5\mu_{2s}+\delta)$	δ
(6,0)									$6\mu_{2s}$				$-(\lambda+6\mu_{2s})$

And $\boldsymbol{G}$ can be written as a partitioned triangular matrix.

$$\boldsymbol{G}=\begin{pmatrix}\boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & & & & & & \\ & \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{C}_2 & & & & & & \\ & & \boldsymbol{B}_3 & \boldsymbol{A}_3 & \boldsymbol{C}_3 & & & & & \\ & & & \ddots & \ddots & \ddots & & & & \\ & & & & \boldsymbol{B}_n & \boldsymbol{A}_n & \boldsymbol{C}_n & & & \\ & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} & & \\ & & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} & \\ & & & & & & & \ddots & \ddots & \ddots \end{pmatrix},$$

where

$$\boldsymbol{A}_k=\begin{cases}\big(-(\lambda+k\mu)\big), 0\le k\le n-c, \\ \begin{pmatrix}-h_c & c\delta & & & \\ & -h_{c-1} & (c-1)\delta & & \\ & & \ddots & \ddots & \\ & & & -h_{n-k-1} & (n-k-1)\delta \\ & & & & -(\lambda+k\mu)\end{pmatrix}_{(k-n+c+1)\times(k-n+c+1)}, n-c<k\le n\end{cases}$$

$$h_m=\lambda+(n-m)\mu+m\delta,\qquad 0\le m\le c;$$

$$\boldsymbol{B}_k=\begin{cases}(k\mu), 1\le k\le n-c, \\ \begin{pmatrix}(n-c)\mu & & & \\ & (n-c+1)\mu & & \\ & & \ddots & \\ & & & (k-1)\mu \\ & & & k\mu\end{pmatrix}_{(k-n+c+1)\times(k-n+c)}, n-c<k\le n;\end{cases}$$

$$\boldsymbol{C}_k=\begin{cases}(\lambda), 0\le k\le n-c, \\ \begin{pmatrix}\lambda & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda\end{pmatrix}_{(k-n+c+1)\times(k-n+c+1)}, n-c<k\le n;\end{cases}$$

$\boldsymbol{C}=\lambda\boldsymbol{I}$, $\boldsymbol{I}$ is the identity matrix of order c+1. And $\boldsymbol{A}$ and $\boldsymbol{B}$ are the following square matrixes.

$$\boldsymbol{A}=\begin{pmatrix} -h_c & c\delta & & & \\ & -h_{c-1} & (c-1)\delta & & \\ & \ddots & \ddots & & \\ & & & -h_1 & \delta \\ & & & & -(\lambda+n\mu) \end{pmatrix}_{(c+1)\times(c+1)};$$

$$\boldsymbol{B}=\begin{pmatrix} (n-c)\mu & & & \\ & (n-c+1)\mu & & \\ & & \ddots & \\ & & & n\mu \end{pmatrix}_{(c+1)\times(c+1)}.$$

When $\rho_2 < 1$, the least nonnegative solution

$$\boldsymbol{R}=\begin{pmatrix} r_{00} & r_{01} & r_{02} & \cdots & r_{0,c-1} & r_{0c} \\ & r_{11} & r_{12} & \cdots & r_{1,c-1} & r_{1c} \\ & & r_{22} & \cdots & r_{2,c-1} & r_{2c} \\ & & & \ddots & \vdots & \vdots \\ & & & & r_{c-1,c-1} & r_{c-1,c} \\ & & & & & \rho_2 \end{pmatrix} \tag{7}$$

of the following matrix equation

$$\boldsymbol{R^2B}+\boldsymbol{RA}+\boldsymbol{C}=\boldsymbol{0} \tag{6}$$

is called rate matrix [43-44]. Where $r_{kk}(0 \le k \le c)$ of $\boldsymbol{R}$ is the root in (0,1) of the following equation

$$(n-c+k)\mu x^2-[\lambda+(n-c+k)\mu+(c-k)\delta]x+\lambda=0,\ 0 \le k \le c \tag{8}$$

and $r_{cc}=\rho_2<1$. In addition, The off-diagonal elements of $\boldsymbol{R}$ satisfy the following equation

$$(n-c+k)\mu\sum_{i=j}^{k} r_{ji}r_{ik}-(\lambda+(n-c+k)\mu+(c-k)\delta)r_{jk}+(c-k+1)\delta r_{jk-1}=0,$$

$$0 \le j \le c-1, j+1 \le k \le c.$$

Let (L_v,V) represent the steady-state limit of the process ($L_v(t)$,$V(t)$), and

$$p_{kj}=P\{L_v=k,V=j\}=\lim_{t\to\infty}P\{L_v(t)=k,V(t)=j\},(k,j)\in\Omega.$$

Thus, the distribution P_k of (L_v,V) can be denoted as

$$P_k=\begin{cases} K\alpha_k=K\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k, 0 \le k \le n-c, \\ K\boldsymbol{\alpha}_k=K\left(\alpha_{kc},\alpha_{k,c-1},\dots,\alpha_{k,n-k}\right), n-c+1 \le k \le n, \\ K\boldsymbol{\alpha}_c\boldsymbol{R}^{k-c}, n<k \end{cases} \tag{9}$$

Where K is the constant factor, and $K=\left[\sum_{i=0}^{n-c}\frac{1}{i!}\left(\frac{\lambda}{\mu}\right)^i+\sum_{i=n-c+1}^{n-1}\boldsymbol{\alpha}_i\boldsymbol{e}+\boldsymbol{\alpha}_n(\boldsymbol{I}-\boldsymbol{R})^{-1}\boldsymbol{e}\right]^{-1}$, $\boldsymbol{e}$ is a column vector whose elements are all 1, and $\alpha_0,\alpha_1,\dots,\alpha_{n-c},\boldsymbol{\alpha}_{n-c+1},\dots,\boldsymbol{\alpha}_n$ is the positive solution of the following equations

$$(p_0,p_1,\dots,p_{n-c},\boldsymbol{p}_{n-c+1},\dots,\boldsymbol{p}_n)\boldsymbol{B}(\boldsymbol{R})=\boldsymbol{0} \tag{10}$$

Where $\boldsymbol{p}_k=(p_{kc},p_{k,c-1},\dots,p_{k,n-k})$, $n-c+1 \le k \le n$, and

$$\boldsymbol{B}(\boldsymbol{R})=\begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & \\ & \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{C}_2 & \\ & & \ddots & \ddots & \ddots \\ & & & \boldsymbol{B}_{n-1} & \boldsymbol{A}_{n-1} & \boldsymbol{C}_{n-1} \\ & & & & \boldsymbol{B}_n & \boldsymbol{RB}+\boldsymbol{A} \end{pmatrix}.$$

According to the formula (10), the joint distribution p_{kj} of (L_v,V) can be obtained when the system reaches equilibrium. And R_2 can be get by use of the following formula

$$R_2 = \sum_{k=0}^{\infty} k \sum_{j=0}^{c} p_{kj} \tag{11}$$

Similarly, the mean service time T_2 of the SQ can be calculated by Little's law, i.e. $T_2 = \frac{R_2}{\lambda}$. Moreover, the utilization U of OP can be obtained by the following formula

$$U = \frac{R_2}{n} \tag{12}$$

Finally, we can obtain the mean V of SOPs which are on vacations by the following formula

$$V = \sum_{j=1}^{c} j \sum_{k=0}^{\infty} p_{kj}.$$

5.3 Analytical model for Stage 3

According to the Literature [44], the task arrival rate in TQ is equal to the output of Stage 2, i.e. λ. Assume that the mean traffic from the Transmitter T to the corresponding client is D/2. Meanwhile, we model this stage as an M/M/1 queuing system. Let $\rho_3 = \frac{\lambda D}{2\xi}$. Similarly, we can obtain R_3 and T_3 by the following formulas when $\rho_3 < 1$.

$$R_3 = \frac{\lambda}{\frac{2\xi}{D} - \lambda}, \quad T_3 = \frac{1}{\frac{2\xi}{D} - \lambda} \tag{13}$$

We can obtain the number R of tasks and the response time T of the whole system by substituting R_1, R_2, R_3 into (1) and T_1, T_2, T_3 into (2).

6 Performance analysis and evaluation by numerical simulation

To verify the correctness and effectiveness of the proposed model for system performance analysis, we conduct some experiments by simulating numerically on the above model. Meanwhile, this paper analyzes the influence of the parameters such as the request arrival rate λ, the maximum c of SOPs on vacations and vacation rate δ on system performance. In addition, we consider the impacts of different vacation models on TOC performance.

6.1 Parameter Settings

Task arrival rate $\lambda \in \{0.002i | 1 \leq i \leq 20, i \in \mathrm{N}\}$, the average transmission speed of network $\xi = 50$ MB/s, the mean traffic of tasks D=0.5 GB, optical computing speed of TOC, i.e. OP σ=5 GB/s, the number of SOPs n=6, the maximum of SOPs allowed to be on vacations c=2, vacation rate of SOPs $\delta = 0.1$ and the times of computation amount compared with the mean traffic m=200. Certainly, these parameters are illustrative, that is, they can be modified.

6.2 Influence on system performance of arrival rate

The experimental results are shown in Table 1 and Fig. 5 by numerically simulating the proposed model by means of the above parameters. T_1 and R_1, T_3 and R_3 all increase linearly with the increase of λ, as can be seen in Fig. 5(a) and Fig. 5(e), Fig. 5(c) and Fig. 5(g), respectively. The reason is that they are all increasing functions of λ, as shown in formulas (5) and (13). Similarly, T_2, T and V all decrease with the increase of λ, as shown in Fig. 5(b), Fig. 5(d) and Fig. 5(i), respectively. The main reason is that the high task arrival rate can result in the state change from "vacation" to "busy", that is, the number of SOPs on vacations decreases with the increase of λ, as shown in Fig. 5(i). In particular, V gradually reduces from 2.0000 to 1.3827 when λ increases from 0.002 to 0.032. We can think that there is one SOP whose state changes from "vacation" to "busy". Thus, the increase of busy SOPs inevitably leads to a decrease of T_2. On the other hand, as shown in Fig. 5(a)~(c) and Table1, T is mainly affected by T_2, so it decreases with the increase of λ. Most interestingly, R_2, R and U all tend to increase first and then

decrease with the increase of λ, as shown in Fig. 5(f), Fig. 5(h) and Fig. 5(j), respectively. The reason is that low task arrival rate does not make the states of SOPs change from “vacation” to “busy”, causing R_2 to increase with the increase of λ. When λ reaches a certain threshold, about 0.032, one of the two SOPs on vacations changes its state from "vacation" to "busy" after its vacation is over, which makes R_2 gradually reduce. Similarly, R is mainly influenced by R_2, as shown in Fig. 5(e)~(g), so it also tends to increase first and then decrease with the increase of λ. According to formula (12), U is a function of R_2 when n is determined, so it, like R_2, tends to increase first and then decrease. In short, T tends to decrease with the increase of λ, while R and U tend to increase first and then decrease.

Table 1 The system performance indicators with different task arrival rates

λ	R_1	R_2	R_3	R	T_1	T_2	T_3	T	V	U
0.002	0.020 4	0.240 0	0.010 1	0.270 5	10.204 1	120.000 5	5.050 5	135.255 1	2.000 0	0.040 0
0.004	0.041 7	0.480 0	0.020 4	0.542 1	10.416 7	120.005 4	5.102 0	135.524 1	1.999 9	0.080 0
0.006	0.063 8	0.720 1	0.030 9	0.814 8	10.638 3	120.015 2	5.154 6	135.808 1	1.999 1	0.120 0
0.008	0.087 0	0.960 1	0.041 7	1.088 7	10.869 6	120.012 1	5.208 3	136.090 0	1.996 8	0.160 0
0.010	0.111 1	1.199 5	0.052 6	1.363 3	11.111 1	119.951 2	5.263 2	136.325 4	1.991 7	0.199 9
0.012	0.136 4	1.437 1	0.063 8	1.637 3	11.363 6	119.760 3	5.319 1	136.443 1	1.982 1	0.239 5
0.014	0.162 8	1.670 8	0.075 3	1.908 9	11.627 9	119.345 5	5.376 3	136.349 8	1.966 5	0.278 5
0.016	0.190 5	1.897 6	0.087 0	2.175 0	11.9048	118.600 0	5.434 8	135.939 6	1.943 5	0.316 3
0.018	0.219 5	2.113 4	0.098 9	2.431 9	12.195 1	117.413 4	5.494 5	135.103 1	1.911 5	0.352 2
0.020	0.250 0	2.313 6	0.111 1	2.674 7	12.500 0	115.680 5	5.555 6	133.736 0	1.869 7	0.385 6
0.022	0.282 1	2.492 8	0.123 6	2.898 4	12.820 5	113.307 5	5.618 0	131.746 0	1.817 0	0.415 5
0.024	0.315 8	2.645 2	0.136 4	3.097 4	13.157 9	110.217 2	5.681 8	129.056 9	1.753 2	0.440 9
0.026	0.351 4	2.765 1	0.149 4	3.265 9	13.513 5	106.351 1	5.747 1	125.611 7	1.677 7	0.460 9
0.028	0.388 9	2.846 8	0.162 8	3.398 4	13.888 9	101.669 8	5.814 0	121.372 7	1.590 7	0.474 5
0.030	0.428 6	2.884 6	0.176 5	3.489 6	14.285 7	96.153 2	5.882 4	116.321 3	1.492 2	0.480 8
0.032	0.470 6	2.873 5	0.190 5	3.534 6	14.705 9	89.798 1	5.952 4	110.456 4	1.382 7	0.478 9
0.034	0.515 2	2.809 0	0.204 8	3.528 9	15.151 5	82.616 7	6.024 1	103.792 3	1.262 7	0.468 2
0.036	0.562 5	2.686 8	0.219 5	3.468 8	15.625 0	74.633 7	6.097 6	96.356 3	1.132 6	0.447 8
0.038	0.612 9	2.503 6	0.234 6	3.351 1	16.129 0	65.884 3	6.172 8	88.186 2	0.993 2	0.417 3
0.040	0.666 7	2.256 5	0.250 0	3.173 1	16.666 7	56.411 4	6.250 0	79.328 0	0.845 1	0.376 1

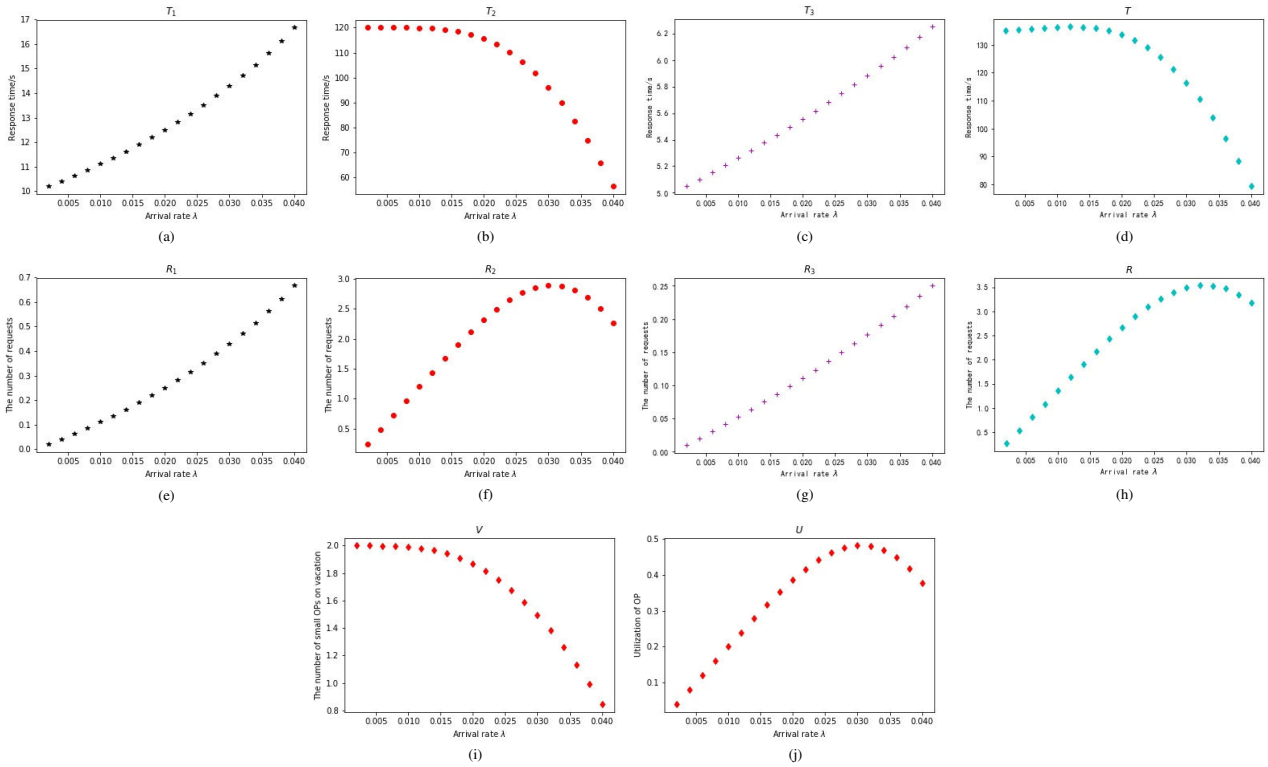

Fig. 5 Influence on the system performance of different task arrival rates.

6.3 Influence on system performance of the number of SOPs

For the parameters in Section 6.1, we consider the system performance indicators including response time T, task number R and OP utilization U when n is equal to 4, 5 and 6 and the other parameters are unchanged. The relevant results are shown in Table 2 and Fig. 6.

We give a special explanation about OP utilization. The SOPs differ in size after the whole OP is uniformly divided into 4, 5 and 6 parts. Therefore, U needs to be modified to obtain the real OP utilization when n is equal to 4 and 5. The correction formula for the utilization is as followed.

$$U_{\mathrm{r}} = rU. \tag{14}$$

Where U_{r} is the OP utilization after being corrected and r is the correction coefficient. r is equal to 1.5 and 1.2, respectively when n is 4 and 5.

As can be seen from Table 2 and Fig. 6, the response times T with different number of SOPs show the same trend with the increase of the task arrival rate λ and the response times for each arrival rate increase with the increase of n. And the task numbers and OP utilizations tend to increase first and then decrease with the increase of arrival rate. Meanwhile, the task numbers for each arrival rate increase with the increase of n while the OP utilizations reduce. The reason is that the processing speed of the whole OP is unchanged and the processing speed of each SOP is larger when n is smaller. In other words, the reduction of n will lead to an improvement in system performance.

Table 2 The system performance indicators when the number n of small OPs is 4, 5 and 6, respectively

λ	n=4				n=5				n=6		
	T	R	U	U_{r}	T	R	U	U_{r}	T	R	U
0.002	95.305 6	0.190 6	0.040 0	0.060 0	115.259 5	0.230 5	0.040 0	0.048 0	135.255 1	0.270 5	0.040 0
0.004	95.684 0	0.382 7	0.080 2	0.120 2	115.548 4	0.462 2	0.080 0	0.096 0	135.524 1	0.542 1	0.080 0
0.006	96.056 1	0.576 3	0.120 4	0.180 6	115.856 4	0.695 1	0.120 1	0.144 1	135.808 1	0.814 8	0.120 0
0.008	96.330 6	0.770 6	0.160 5	0.240 8	116.138 9	0.929 1	0.160 1	0.192 1	136.090 0	1.088 7	0.160 0
0.010	96.419 0	0.964 2	0.200 1	0.300 2	116.325 0	1.163 3	0.199 9	0.239 9	136.325 4	1.363 3	0.199 9

0.012	96.243 0	1.154 9	0.238 7	0.358 0	116.328 6	1.395 9	0.239 2	0.287 0	136.443 1	1.637 3	0.239 5
0.014	95.739 4	1.340 4	0.275 6	0.413 4	116.058 7	1.624 8	0.277 4	0.332 8	136.349 8	1.908 9	0.278 5
0.016	94.860 6	1.517 8	0.310 1	0.465 1	115.428 2	1.846 9	0.313 9	0.376 7	135.939 6	2.175 0	0.316 3
0.018	93.575 2	1.684 4	0.341 5	0.512 2	114.360 2	2.058 5	0.348 0	0.417 6	135.103 1	2.431 9	0.352 2
0.020	91.865 6	1.837 3	0.369 1	0.553 6	112.791 9	2.255 8	0.378 9	0.454 7	133.736 0	2.674 7	0.385 6
0.022	89.727 0	1.974 0	0.392 1	0.588 1	110.676 4	2.434 9	0.405 8	0.487 0	131.746 0	2.898 4	0.415 5
0.024	87.164 9	2.092 0	0.410 0	0.614 9	107.982 7	2.591 6	0.427 9	0.513 5	129.056 9	3.097 4	0.440 9
0.026	84.193 1	2.189 0	0.422 1	0.633 1	104.695 5	2.722 1	0.444 3	0.533 1	125.611 7	3.265 9	0.460 9
0.028	80.832 0	2.263 3	0.427 9	0.641 9	100.813 1	2.822 8	0.454 2	0.545 1	121.372 7	3.398 4	0.474 5
0.030	77.106 8	2.313 2	0.427 0	0.640 6	96.345 8	2.890 4	0.457 1	0.548 5	116.321 3	3.489 6	0.480 8
0.032	73.046 2	2.337 5	0.419 1	0.628 7	91.314 1	2.922 0	0.452 2	0.542 6	110.456 4	3.534 6	0.478 9
0.034	68.681 0	2.335 2	0.403 8	0.605 7	85.746 1	2.915 4	0.439 1	0.526 9	103.792 3	3.528 9	0.468 2
0.036	64.043 5	2.305 6	0.380 9	0.571 3	79.676 2	2.868 3	0.417 3	0.500 7	96.356 3	3.468 8	0.447 8
0.038	59.166 3	2.248 3	0.350 2	0.525 3	73.143 5	2.779 5	0.386 4	0.463 7	88.186 2	3.351 1	0.417 3
0.040	54.082 2	2.1633	0.311 7	0.467 5	66.189 9	2.647 6	0.346 2	0.415 4	79.328 0	3.173 1	0.376 1

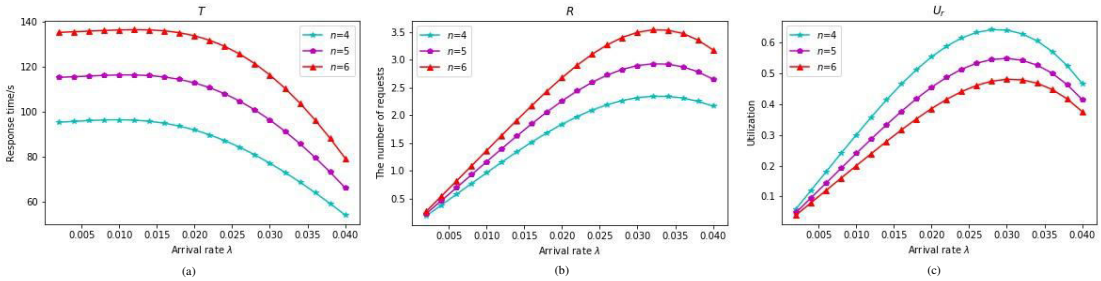

Fig. 6 Influence on the system performance of the number n of SOPs.

6.4 Influence on system performance of the maximum of SOPs allowed to be on asynchronous vacations

For the parameters in Section 6.1, we similarly consider the system performance indicators including response time T, task number R and OP utilization U when c is equal to 1, 2 and 3 and the other parameters are unchanged. The relevant results are shown in Table 3 and Fig. 7.

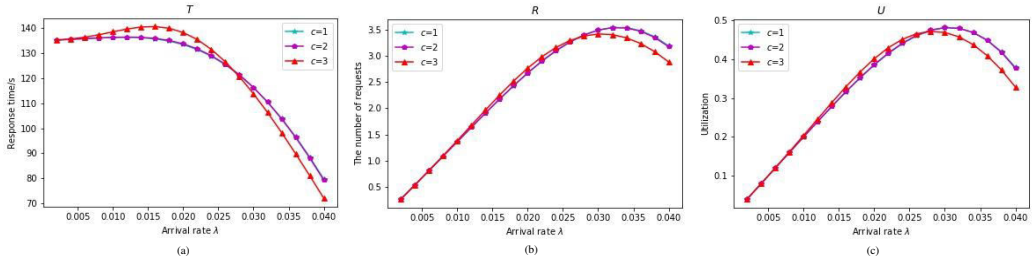

Fig. 7 Influence on the system performance of the number c of small OPs allowed to be on vacations.
(a) Response time T; (b)The number R of requests; (c)Utilization U of OP

Table 3 The system performance indicators when the number c of SOPs allowed to be on vacations is 1, 2 and 3, respectively

λ	c=1			c=2			c=3		
	T	R	U	T	R	U	T	R	U
0.002	135.3379	0.2707	0.0400	135.2551	0.2705	0.0400	135.2715	0.2705	0.0400
0.004	135.6015	0.5424	0.0800	135.5241	0.5421	0.0800	135.6885	0.5428	0.0801
0.006	135.8681	0.8152	0.1200	135.8081	0.8148	0.1200	136.3841	0.8183	0.1206
0.008	136.1159	1.0889	0.1599	136.0900	1.0887	0.1600	137.3773	1.0990	0.1617
0.010	136.3010	1.3630	0.1997	136.3254	1.3633	0.1999	138.5486	1.3855	0.2036
0.012	136.3565	1.6363	0.2392	136.4431	1.6373	0.2395	139.6714	1.6761	0.2460
0.014	136.1969	1.9068	0.2779	136.3498	1.9089	0.2785	140.4638	1.9665	0.2881
0.016	135.7250	2.1716	0.3155	135.9396	2.1750	0.3163	140.6429	2.2503	0.3288
0.018	134.8405	2.4271	0.3512	135.1031	2.4319	0.3522	139.9690	2.5194	0.3668
0.020	133.4465	2.6689	0.3844	133.7360	2.6747	0.3856	138.2731	2.7655	0.4007

0.022	131.4554	2.8920	0.4141	131.7460	2.8984	0.4155	135.4685	2.9803	0.4291
0.024	128.7936	3.0910	0.4395	129.0569	3.0974	0.4409	131.5457	3.1571	0.4508
0.026	125.4031	3.2605	0.4596	125.6117	3.2659	0.4609	126.5597	3.2906	0.4650
0.028	121.2430	3.3948	0.4735	121.3727	3.3984	0.4745	120.6109	3.3771	0.4709
0.030	116.2889	3.4887	0.4802	116.3213	3.4896	0.4808	113.8271	3.4148	0.4683
0.032	110.5322	3.5370	0.4789	110.4564	3.5346	0.4789	106.3474	3.4031	0.4570
0.034	103.9785	3.5353	0.4687	103.7923	3.5289	0.4682	98.3107	3.3426	0.4371
0.036	96.6457	3.4792	0.4490	96.3563	3.4688	0.4478	89.8472	3.2345	0.4087
0.038	88.5621	3.3654	0.4191	88.1862	3.3511	0.4173	81.0743	3.0808	0.3722
0.040	79.7645	3.1906	0.3784	79.3280	3.1731	0.3761	72.0941	2.8838	0.3278

As can be seen from Table 3 and Figure 7, *T*, *U* and *R* show the same trend with increase of λ when *c* is assigned with different values. That is, they all increase first and then decrease. However, *R* and *U* both increase more significantly. When *c* is equal to 1 and 2, the values of *T*, *R* and *U* for a certain arrival rate are basically identical, respectively. The performance indicators of *c* equal to 3 are significantly lower than those of it equal to 1 and 2 when $\lambda > 0.028$, and the response time *T* of *c* equal to 3 is significantly higher than that of the other two when $\lambda < 0.028$. The reason is that when the lower arrival rate doesn't change the states of the three SOPs on vacation, i.e. make one of them run. And these SOPs enter running state one by one with the increase of λ. At the same time, all of the SOPs can be maintained when they are on vacation, which further improves system performance. Therefore, this gives us the optimal option for the number of the allowed vacation SOPs. If $\lambda > 0.028$, *c* is 3; otherwise, *c* is 2(considering energy savings on vacations).

6.5 Influence on system performance of vacation rate

For the parameters in Section 6.1, we similarly consider the system performance indicators including response time *T*, task number *R* and OP utilization *U* when δ is equal to 0.000 1, 0.001, 0.01, 0.1 and 1.0 while the other parameters are unchanged. The relevant results are shown in Table 4 and Fig. 8.

Table 4 The system performance indicators when vacation rate δ is 0.0001, 0.001, 0.01 and 1.0, respectively

λ	$\delta = 0.0001$			$\delta = 0.001$			$\delta = 0.01$			$\delta = 1.0$		
	T	*R*	*U*	*T*	*R*	*U*	*T*	*R*	*U*	*T*	*R*	*U*
0.002	135.258 2	0.270 5	0.040 0	135.258 0	0.270 5	0.040 0	135.256 8	0.270 5	0.040 0	135.254 6	0.270 5	0.040 0
0.004	135.566 6	0.542 3	0.080 2	135.563 7	0.542 3	0.080 2	135.546 9	0.542 2	0.080 2	135.518 1	0.542 1	0.080 2
0.006	135.987 8	0.815 9	0.120 4	135.974 6	0.815 8	0.120 4	135.902 4	0.815 4	0.120 4	135.783 4	0.814 7	0.120 4
0.008	136.551 3	1.092 4	0.160 5	136.515 3	1.092 1	0.160 5	136.327 5	1.090 6	0.160 5	136.026 9	1.088 2	0.160 5
0.010	137.210 0	1.372 1	0.200 1	137.137 9	1.371 4	0.200 1	136.775 8	1.367 8	0.200 1	136.203 4	1.362 0	0.200 1
0.012	137.818 3	1.653 8	0.238 7	137.705 0	1.652 5	0.238 7	137.146 0	1.645 8	0.238 7	136.246 6	1.635 0	0.238 7
0.014	138.125 9	1.933 8	0.275 6	137.990 7	1.931 9	0.275 6	137.290 3	1.922 1	0.275 6	136.073 4	1.905 0	0.275 6
0.016	137.784 4	2.204 6	0.310 1	137.697 3	2.203 2	0.310 1	137.032 8	2.192 5	0.310 1	135.591 3	2.169 5	0.310 1
0.018	136.360 4	2.454 5	0.341 5	136.482 0	2.456 7	0.341 5	136.189 8	2.451 4	0.341 5	134.705 4	2.424 7	0.341 5
0.020	133.354 8	2.667 1	0.369 1	133.995 8	2.679 9	0.369 1	134.591 0	2.691 8	0.369 1	133.324 6	2.666 5	0.369 1
0.022	128.231 9	2.821 1	0.392 1	129.932 9	2.858 5	0.392 1	132.097 6	2.906 1	0.392 1	131.367 0	2.890 1	0.392 1
0.024	120.465 9	2.891 2	0.410 0	124.088 7	2.978 1	0.410 0	128.616 0	3.086 8	0.410 0	128.762 4	3.090 3	0.410 0
0.026	109.632 0	2.850 4	0.422 1	116.422 6	3.027 0	0.422 1	124.105 3	3.226 7	0.422 1	125.454 6	3.261 8	0.422 1
0.028	95.602 2	2.676 9	0.427 9	107.108 8	2.999 0	0.427 9	118.579 6	3.320 2	0.427 9	121.401 4	3.399 2	0.427 9
0.030	78.963 8	2.368 9	0.427 0	96.552 8	2.896 6	0.427 0	112.104 6	3.363 1	0.427 0	116.574 2	3.497 2	0.427 0
0.032	61.666 1	1.973 3	0.419 1	85.352 6	2.731 3	0.419 1	104.789 3	3.353 3	0.419 1	110.957 4	3.550 6	0.419 1
0.034	46.918 8	1.595 2	0.403 8	74.197 2	2.522 7	0.403 8	96.775 9	3.290 4	0.403 8	104.546 3	3.554 6	0.403 8
0.036	36.859 8	1.327 0	0.380 9	63.733 1	2.294 4	0.380 9	88.227 2	3.176 2	0.380 9	97.346 2	3.504 5	0.380 9
0.038	31.011 9	1.178 5	0.350 2	54.445 3	2.068 9	0.350 2	79.314 6	3.014 0	0.350 2	89.370 3	3.396 1	0.350 2
0.040	27.919 3	1.116 8	0.311 7	46.596 7	1.863 9	0.311 7	70.207 5	2.808 3	0.311 7	80.638 8	3.225 6	0.311 7

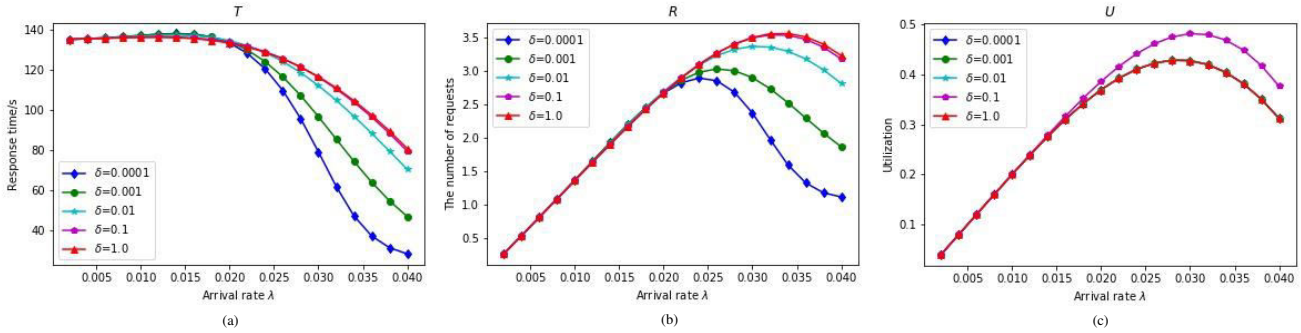

Fig. 8 Influence on the system performance of vacation rate δ.

As can be seen from Table 4 and Fig. 8, vacation rate δ has little or no effect on system performance when task arrival rate λ is low, for example, λ is less than 0.020; when λ is high, vacation rate has a significant effect on the response time T and task number R. In other words, both of them for each arrival rate remarkably decrease with the decrease of δ. The main reason is that smaller vacation rate means fewer vacations per unit time. Obviously, the lower vacation number makes the tasks just arriving be processed timely, improving system performance. In addition, as can be seen from Table 4 and Fig. 8(c), the OP utilization U is improved when δ is 0.1 and the task arrival rate is higher; the utilization curves for the other values of δ is essentially coincident since the utilizations for each arrival rate are identical. As a result, small vacation rate can improve system performance.

7 Performance comparison under different vacation models

In this section, we shall compare the main system performance indicators such as response time T, task number R, and OP utilization U under different vacation models. These models include TSSMAV proposed in this paper, FSSMSV proposed in Literature [9] and FSSMAV obtained by replacing the synchronous vacation in Literature [9] with the asynchronous vacation in this paper. The parameters in Section 6.1 are all unchanged and a parameter τ, i.e. the speed of electronic computer is added. Let $\tau = 3$ GB/s. And the relevant results are shown in Table 5 and Fig. 9.

Table 5 The performance comparison under different vacation models

λ	TSSMAV			FSSMAV			FSSMSV		
	T	*R*	*U*	*T*	*R*	*U*	*T*	*R*	*U*
0.002	135.255 1	0.270 5	0.040 0	135.505 1	0.271 0	0.040 0	143.388 2	0.286 8	0.040 0
0.004	135.524 1	0.542 1	0.080 0	135.774 2	0.543 1	0.080 0	142.006 9	0.568 0	0.080 0
0.006	135.808 1	0.814 8	0.120 0	136.058 3	0.816 4	0.120 0	140.995 0	0.846 0	0.120 0
0.008	136.090 0	1.088 7	0.160 0	136.340 2	1.090 7	0.160 0	140.275 6	1.122 2	0.160 0
0.010	136.325 4	1.363 3	0.199 9	136.575 8	1.365 8	0.199 9	139.798 6	1.398 0	0.200 1
0.012	136.443 1	1.637 3	0.239 5	136.693 5	1.640 3	0.239 5	139.535 5	1.674 4	0.240 2
0.014	136.349 8	1.908 9	0.278 5	136.600 3	1.912 4	0.278 5	139.476 7	1.952 7	0.280 5
0.016	135.939 6	2.175 0	0.316 3	136.190 1	2.179 0	0.316 3	139.628 4	2.234 1	0.321 2
0.018	135.103 1	2.431 9	0.352 2	135.353 7	2.436 4	0.352 2	140.012 1	2.520 2	0.362 4
0.020	133.736 0	2.674 7	0.385 6	133.986 7	2.679 7	0.385 6	140.664 2	2.813 3	0.404 5
0.022	131.746 0	2.898 4	0.415 5	131.996 8	2.903 9	0.415 5	141.637 0	3.116 0	0.447 9
0.024	129.056 9	3.097 4	0.440 9	129.307 8	3.103 4	0.440 9	143.001 9	3.432 0	0.493 1
0.026	125.611 7	3.265 9	0.460 9	125.862 6	3.272 4	0.460 9	144.855 3	3.766 2	0.541 0
0.028	121.372 7	3.398 4	0.474 5	121.623 6	3.405 5	0.474 5	147.327 5	4.125 2	0.592 6
0.030	116.321 3	3.489 6	0.480 8	116.572 3	3.497 2	0.480 8	150.599 7	4.518 0	0.649 3
0.032	110.456 4	3.534 6	0.478 9	110.707 5	3.542 6	0.478 9	154.930 7	4.957 8	0.713 5
0.034	103.792 3	3.528 9	0.468 2	104.043 5	3.537 5	0.468 2	160.705 4	5.464 0	0.788 2
0.036	96.356 3	3.468 8	0.447 8	96.607 5	3.477 9	0.447 8	168.525 2	6.066 9	0.878 5

0.038	88.186 2	3.351 1	0.417 3	88.437 5	3.360 6	0.417 3	179.388 0	6.816 7	0.992 6
0.040	79.328 0	3.173 1	0.376 1	79.579 4	3.183 2	0.376 1	195.085 3	7.803 4	1.000 0

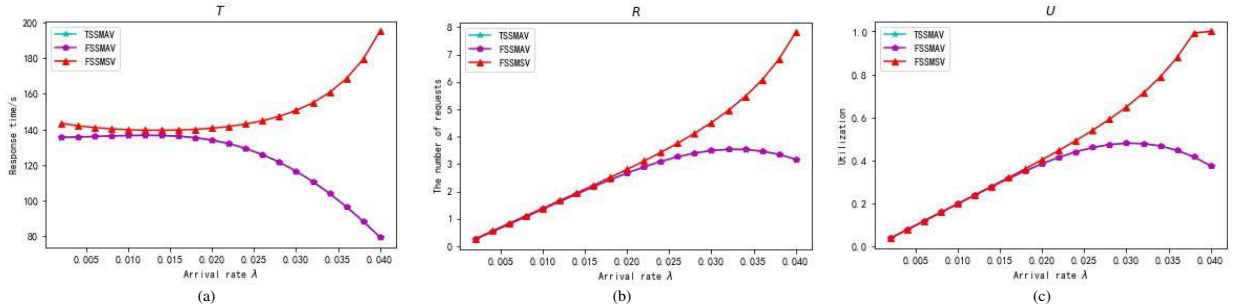

Fig. 9 Influence on the system performance of different vacation models.

It can be easily seen that T first reduces slightly and then increases with the increase of λ while R and U under FSSMSV all increase. Meantime, response times under TSSMAV and FSSMAV both decrease with the increase of λ and R and U under TSSMAV and FSSMAV both trend to increase first and then decrease. Moreover, the three performance indicator curves under TSSMAV and FSSMAV are basically coincident, respectively. The reason is as follows. The potential factors, including n, c and δ, that affect U are unchanged, so U under TSSMAV and FSSMAV are identical, as shown in Table 5. Compared with TSSMAV, FSSMAV is added the data preprocessing stage so that the R and T under the latter are slightly higher than those of the former.

In particular, the increasing trend of U under FSSMSV does not mean that FSSMSV can improve system performance. On the contrary, it will degrade the system performance. The too high utilization will result in a higher risk of downtime because the system will not be maintained effectively at high utilization. Of particular importance, each of the three indicators for each arrival rate under FSSMSV is greater than the other two and each of the three indicators for each arrival rate under TSSMAV is not more than the other two. Therefore, TSSMAV outperforms the other two vacation models in the system performance of TOC.

8 Conclusions and future work

Performance analysis and evaluation of a TOC has been paid more and more attentions of both its customers and its providers. To more accurately model the calculation ecology of the TOC for performance analysis and evaluation of the TOC, this paper built a three-staged service model of the TOC by introducing the asynchronous multi-vacation queuing and tandem queuing. And the vacation is referred to the asynchronous vacations of the SOPs. Moreover, for each SOP, it doesn't start a vacation until it services exhaustively and the number of vacation SOPs is less than the maximum of SOPs allowed to be on vacations. Meanwhile, this paper proposed a task scheduling algorithm with partial SOP asynchronous multi-vacation and OP management algorithm including a OP allocation algorithm and a OP recovery algorithm.

We focused on the building of analytical models for performance analysis and evaluation of the TOC after selecting some primary performance indicators such as the number of tasks, response time and OP utilization. We built some mathematical models for the numbers of tasks and the service times in the first and third stage based on the M/M/1 queuing system. Particularly, we established the analytical models for the performance indicators, the number of tasks, service time and OP utilization in the second stage based on M/M/n queuing system with asynchronous multi-vacation and solved them by introducing the QBD process. Thus, we obtained the number of tasks in the TOC system and the response time by summing them, respectively.

Finally, we obtained the system performance indicators by numerically simulating on these models. And the results showed that the response time drops with the increase of the task arrival rate while the number of tasks and

the OP utilization increase first and then decrease. And the smaller the number of SOPs is and the smaller the vacation rate is, the higher the system performance is. In addition, the maximum of SOPs allowed to be on vacations has an important impact on the system performance. Moreover, if the number of SOPs is 6, the performance is optimal when the maximum of SOPs allowed to be on vacations is 2 and the arrival rate is not greater than 0.028 or when the maximum of SOPs allowed to be on vacations is 3 and the arrival rate is greater than 0.028. At the same time, the system performance was compared under different vacation models, such as FSSMSV, TSSMAV and FSSMAV. The results showed that the performance under TSSMAV is superior to the other two. Therefore, compared with synchronous vacations, asynchronous vacations can not only make the system better maintained but also improve system performance to some extent. The next step is to study how to optimize the parameters to achieve the optimal TOC performance and further improve user experience.

Acknowledgments: The authors thank Yi Jin in Shanghai University for providing the optical platform.

Compliance with ethical standards

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

Funding: This research was funded by the Project of National Natural Science Foundation of China, grant numbers 61672006 and 61862023, the Key Project of Natural Science Research in Anhui, grant numbers KJ2019A0533 and KJ2019A0535.

Conflicts of Interest: The authors declare that they have no conflict of interest. And the funders had no role in the design of the study.

Informed Consent: All authors have read and agreed to the submitted version of the manuscript.

Author Contributions

Conceptualization, Xianchao Wang; methodology, Jie Zhang and Xianchao Wang; software, Xianchuan Wang and Dayou Hou; validation, Xianchuan Wang and Man Ling; writing—original draft preparation, Xianchuan Wang; writing—review and editing, Xianchao Wang; visualization, Kai Song; project administration, Xianchao Wang; funding acquisition, Xianchao Wang, Kai Song and Xianchuan Wang.

References

[1] R. A. Heinz, J. O. Artman, S. H. Lee, Matrix multiplication by optical methods, Appled Optics, 1970, 9: 2161–2168.

[2] P. Ambs, Optical computing: A 60-year adventure, Advances in Optical Technologies, 2010, 372652, doi:10.1155/2010/372652.

[3] F. Zangeneh-nejad, A. Khavasi, B. Rejaei, Analog optical computing by half-wavelength slabs, Optics Communications, 2018, 407: 338-343.

[4] A. N. Z. Rashed, A. E.-N. A. Mohammed, W. F. Zaky, et al, The switching of optoelectronics to full optical computing operations based on nonlinear metamaterials, Results in Physics, 2019, 13: 102152.

[5] Y. Zhou, R. Chen, W. Chen, et al, Optical analog computing devices designed by deep neural network, Optics Communications, 2020, 458: 124674.

[6] Y. Jin, H. C. He, Y. T. Lü, Ternary optical computer principle, Science China F Information Science, 2003, 46(2): 145–150.

[7] Y. Jin, H. C. He, Y. T. Lü, Ternary optical computer architecture, Physica Scripta, 2005, 118: 98–101.

[8] J. Y. Yan, Y. Jin, K. Z. Zuo, Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer, Science China F Information Science, 2008, 51(10): 1415–1426.

[9] X. C. Wang, J. Zhang, S. Gao, et al, Performance analysis and evaluation of ternary optical computer based on a queueing system with synchronous multi-vacations, IEEE Access, 2020, 8: 67214–67227.

[10] Y. F. Shen, L. Pan, Principle of a one-step MSD adder for a ternary optical computer, Science in China (Series F), 2014, 57(1): 012107-1-10.

[10] J. J. Peng, R. Shen, Y. Jin, et al, Design and implementation of modified signed-digit adder, IEEE Transactions on Computers, 2014, 63(5): 1134-1143.

[12] X. C. Wang, J. J. Peng, M. Li, et al, Carry-free vector-matrix multiplication on a dynamically reconfigurable optical platform, Applied Optics, 2010, 49(12): 2352-2362.

[13] Y. Jin, Y. F. Shen, J. J. Peng, et al, Principles and construction of MSD adder in ternary optical computer, Science China F Information Science, 2010, 53(11): 2159–2168.

[14] K. Song, L. P. Yan, Design and implementation of the one-step MSD adder of optical computer, Applied Optics, 2012, 51(7): 917–926.

[15] K. Song, G. Chen, Q. Q. Jin, et al, Design of MSD multiplier for ternary optical computer processor based on minimum module, Optics Communications, 2019, 448: 33-42.

[16] Q. Xu, X. C. Wang, C. Xu, Design and implementation of the modified signed digit multiplication routine on a ternary optical computer, Applied Optics,

2017, 56(16): 4661–4669.

[17] H. L. Zhang, J. Zhou, S. L. Zhang, et al, Design and implementation of positive and negative discriminator of MSD data for ternary optical processor, Journal of Computer Research and Development, 2017, 54(6): 1391-1404.(in Chinese)

[18] S. Ouyang, J. J. Peng, Y. Jin, et al, Structure and theory of dual-space storage for ternary optical computer, Science China(Information Science), 2016, 46(6): 743-762(in Chinese).

[19] X. C. Wang, J. J. Peng, S. Ouyang, Control method for the optical components of a dynamically reconfigurable optical platform, Applied Optics, 2011, 50(5): 662-670.

[20] K. Song, Y. Jin, Overall plan and design of the task management system of ternary optical computer, Journal of Shanghai University, 2011, 15(5): 467-472.

[21] Y. Jin, S. Ouyang, K. Song, et al, Management of many data bits in ternary optical computers, Science China(Information Science), 2013, 43(3): 361-373. (in Chinese)

[22] S. L. Zhang, J. J. Peng, Y. F. Shen, et al, Programming model and implementation mechanism for ternary optical computer, Optics Communications, 2018, 428:26-34.

[23] Y. Jin, S. L. Zhang, S. Li, et al, The computing-data file: A key technology of applying ternary optical computer, Journal of Shanghai Jiaotong University, 2019, 53(5): 584-592.

[24] S. Li, Y. Jin, Y. J. Liu, Initial SZG file generation software for ternary optical computer, Journal of Shanghai University (Natural Science), 2018, 24(2): 181-191. (in Chinese)

[25] H. Gao, Y. Jin, K. Song Extension of C language in ternary optical computer[J]. Journal of Shanghai University (Natural Science), 2013, 19(3): 280-285. (in Chinese)

[26] Q. Zhang, Y. Jin, K. Song, et al, MPI programming based on ternary optical computer in supercomputer, Journal of Shanghai University (Natural Science), 2014, 20(2): 180-189. (in Chinese)

[27] Q. Xu, Y. Jin, Y. F. Shen, et al. MSD iterative division algorithm and implementation technique for a ternary optical computer, Science China(Information Science), 2016, 46(4): 539-550. (in Chinese)

[28] J. J. Peng, X. Y. Wei, X. F. Zhang, et al, Implementation of parallel FFT algorithm on a ternary optical computer, Science China(Information Science), 2017, 47(7): 846–862. (in Chinese)

[29] J. J. Peng, Y. Y. Fu, X. F. Zhang, et al, Implementation of DFT application on ternary optical computer, Optics Communications, 2018, 410: 424-430.

[30] K. Song, Q. Q. Jin, G. Chen, et al, Algorithm on higher-order derivative based on ternary optical computer[J]. IEEE Access, 2020, 8: 64499-64513.

[31] S. L. Zhang, Y. F. Shen, Z. Y. Zhao, Design and implementation of a three-lane CA traffic flow model on ternary optical computer, Optics Communications, 2020, 470: 125750.

[32] S. Li, W. J. Li, H. H. Zhang, et al, Research and implementation of parallel artificial bee colony algorithm based on ternary optical computer, Automatika, 2019, 60(4): 422-431.

[33] S. B. Liu, Y. Jin, J. J. Peng, et al, The response time measurement system of optical computer component, International Conference on Information Engineering and Computer Science, NJ: IEEE, 2009: 1986-1990.

[34] X. C. Wang, S. L. Zhang, M. Zhang, et al, Performance analysis of a ternary optical computer based on M/M/1 queueing system, International Conference on Algorithms and Architectures for Parallel Processing, Switzerland AG: Springer, Cham, 2017: 331–344.

[35] Q. Xu, X. C. Wang, Service model and performance analysis of ternary optical computer based on complex queuing system, Journal of National University of Defense Technology, 2017, 39(2): 140–145. (in Chinese)

[36] X. C. Wang, S. L. Zhang, S. Gao, et al, Response time of a ternary optical computer that is based on queuing systems, The Journal of Supercomputing, 2019: 1–20.

[37] X. C. Wang, J. Zhang, S. Gao, et al, Response time of a ternary optical computer based on complex queueing system with synchronous multi-vacations, International Conference on Applications and Techniques in Cyber Security and Intelligence, Switzerland AG: Springer, Cham, 2020: 1180–1188.

[38] D. Gross, J. F. Shortie, J. M. Thompson, et al, Fundamentals of queueing theory(Fourth Edition), New Jersey:John Wiley & Sons, Inc, 2008.

[39] L. Kleinrock, Queueing systems:theory,vol 1, New York: Wikey-Interscience, 1975.

[40] P. J. Burke, The output of a queueing system, Operations Research, 1956, 26 (6) :699-704.

[41] W. J. Stewart, Probability, Markov chains, queues, and simulation-the mathematical basis of performance modeling, New Jersey: Princeton University Press,

2009.

[42] N. S. Tian, Q. L. Li, The M/M/c queue with PH synchronous vacations, System Science and Mathematical Sciences, 2000, 13(1):7-16.

[43] M. F. Neuts, Matrix-geometric solutions on stochastic models-an algorithmic approach, Baltimore and London:The Johns Hopkins University Press, 1981.

[44] N. S. Tian, Z. G. Zhang, Vacation queueing models theory and applications, New York: Springer, 2006.

[45] P. J. Burke, The output process of a stationary M/M/s queueing system, Annals of Mathematical Statistics, 1968, 39(4) :1144-1152.