A QUEUEING MODEL FOR VIDEO ANALYTICS APPLICATIONS OF SMART CITIES

Mani Sharifi
Abdolreza Abhari

Distributed Systems & Multimedia Processing Laboratory (DSMP lab), Department of Computer Science,
Ryerson University, Toronto, Ontario, CANADA

Sharareh Taghipour

The Reliability, Risk, and Maintenance Research Laboratory (RRMR Lab), Mechanical & Industrial Engineering Department,
Ryerson University, Toronto, Ontario, CANADA

ABSTRACT

This paper aims to find a proper methodology for evaluating job scheduling strategies for a data-intensive application such as video analytics applications used for smart cities that involve edge and cloud computing. To compare two simulation methods with the analytical modeling for such evaluation, we proposed a queueing model for a system consisting of some heterogeneous edge processors and one cloud processor and compared it with a simple simulation approach. We first defined the system's characteristics and developed a queueing model for the system to calculate the edges and cloud processors' working times. We use the state-space diagram of the system to determine the set of differential equations of the system and solved them to calculate the system components' performance measures. The results show that the proposed queueing model's computational time is significantly less than other existing techniques like the simulation.

1 INTRODUCTION

To provide a fast processing and network response time in the edge and cloud processing system used for different applications such as video analytics, the most important challenges are how to use online real-time dispatching strategies. For instance, Real-time video surveillance systems based on security cameras, aerial surveillance systems, smart cars, and video-based IoT devices (e.g., home cameras) need high computing video processing nodes close to cameras or user devices. In most cases, such real cases use a set of available Edge processors (EPs) for image or video processing, while a Cloud Processor (CP) supports the EPs. Our recent work proposed a task scheduling method to dispatch the jobs for edge nodes in the fog computing infrastructure for such real-time (i.e., real-time video analytics when job arrival times are fixed and there is a hard deadline for processing the tasks). Using the proposed method, the achieved results show that better network performance can be achieved in fog computing involving edge nodes and cloud computing than cloud computing (Sharifi et al. 2021). For the smart cities that include a mix of processing powers, such as CPU and GPU-based edge servers and public clouds, infrastructure networks may not provide the required latency when the number of cameras increases to hundreds of thousands to control traffic in big cities (Ananthanarayanan et al. 2017). Therefore, providing a model that takes into account the scalability and calculates the processors' performance levels in a significantly short time is needed.

In video analytics applications for smart cities, hundreds of thousands of cameras store their video frames on a database. Then, the stored data will be sent to a free processor for video processing based on demand. So, it can be assumed that the video processing for smart cities is not real-time. With this assumption, we proposed an analytic model based on a queuing system that can help have a good approximation of the processors' performance level to examine and optimize the data/task dispatching rules. Consider a video processing application that consists of several data processing jobs that arrive at random times with different sizes and should be dispatched among the existing EPs and CP. The edge processors are different in terms of processing speed and can handle a maximum of one job. It means that there is no queue for the edge processors, and if a job arrives when no edge processor is idle, the job should be processed on the cloud processor. The cloud processor can process different jobs in parallel, and its processing speed is much faster than the edge processors. Consider that the job number i arrives at the time t_i with the size of d_i, in which the arrival time and the size have uncertain values. For instance, in a video analytics application, the job's

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

size depends on the number of video frames defined in Kbytes. The processing power of the EPs are different from each other's and are significantly lower than the CP. The processing power of processor number j is considered as p_j.

In this paper, we work on a system consisting of a single queue that is together with a dispatching center, with m heterogeneous EPs, and one CP. The jobs arrive in the dispatching center in stochastic times and are dispatched among the idle EPs or CP, while there is no latency between the dispatching center and processors. The EP's can process only one job at the moment, and no queue is for the EPs. The CP can process the jobs in parallel, so when a job arrives and there is no idle EP, the job dispatches to the CP. We develop a queueing model to calculate the performance level of EPs and CP.

For this reason, we first define the states of the system. Then we use the state-space diagram of the under-studied system to determine the set of the differential equation of the system's states. Finally, by solving the set of differential equations, we calculate the states' probabilities and each processor's working time in a steady-state condition to determine each processor's performance level. To validate and compare the proposed method's result, we also used the simulation technique for the first-in-first-out (FIFO) dispatching strategy in two different scenarios.

2 LITERATURE REVIEW

During the past decade, the researcher focused on new approaches to processor scheduling to improve computer-based processing. In this regard, Hamayun and Khurshid (2017) developed a new approach for a preemptive shortest job first, which improves the efficiency of CPU for the real-time and time-sharing environment, which improved the drawbacks of the preemptive shortest job first scheduling algorithm. Ramezani et al. (2015) developed a comprehensive multi-objective optimization model for task scheduling, considering minimizing task transfer time, task execution cost, power consumption, and task queue length. Dash and Samantra (2016) proposed a dynamic average burst round-robin algorithm for CPU scheduling. Hicham and Lotfi (2017) applied artificial neural network algorithms to optimize CPU scheduling for cloud computing models. Kumari et al. (2017) investigated the design and implementation of a modified fuzzy-based CPU scheduling algorithm and presented a new set of fuzzy rules, which improved average waiting time and average turnaround time. Bitam et al. (2018) proposed a new bio-inspired optimization approach called Bees Life Algorithm for the job scheduling problem in the fog computing environment, which optimized the distribution of a set of tasks among all the fog computing nodes. Ma et al. (2019) proposed a deadline and cost-aware scheduling algorithm to minimize the execution cost of a workflow under deadline constraints in the infrastructure as a service model considering the virtual machine performance variation and acquisition delay. Doan et al. (2019) proposed scalable IoT video data analytics applications for Smart cities to end users, who could exploit scalability in both data storage and processing power to execute analysis on large or complex datasets. Harki et al. (2020) presented a review paper that was organized based on two distinct perspectives: the implementation strategies of CPU scheduling technique and criteria-based measures used. Grammenos et al. (2021) proposed an asynchronous iterative scheme that allows a set of interconnected nodes to distributively reach an agreement within a pre-specified bound in a finite number of steps and adopted the scheme context of task scheduling for data centers. The readers are referred to by combining the Markov models and queueing systems to scheduling problems (Grillo et al. 2015, Cheng et al. 2016, Shen et al. 2019, Sharifi and Taghipour 2020, Ghaleb et al. 2020, Sharifi and Taghipour 2021, Sharifi et al. 2021)

This paper tried to fill the literature gap by proposing a queueing-based process task scheduling with heterogeneous edge processors and a backup cloud processor used in smart cities with the assumptions discussed above. We first turn the processor's scheduling problem into a queueing model. Then we establish the set of differential equations between the system's states. Next, we solve the set of differential equations to calculate the state probabilities and processors' performance levels. Finally, we check the proposed model's connectivity by comparing the results with a simulation-based technique. Although the proposed queueing-based model is complicated in the design stage, it can measure the processors' performance levels

in a significantly short computational time compared with other techniques like simulation. The novelties of this paper are summarized as follows:

- Developing a queueing-based model for a smart city processing center considering dispatching center, hundreds of thousands of Edge processors, and Cloud,
- Deriving the general set of differential equations between the system's states,
- Measure the performance of the different dispatching strategies using the proposed queueing-based model's results and compare it with a simulation-based method. To the best of our knowledge, no similar paper compares analytics and simulation approaches for this problem.

3 QUEUEING MODEL

In this section, we first define the state of the system. Then we define the transition rates between the system's state. After that, we derive the set of differential equations between the system's states. Finally, we calculate all processors' performance levels, having the transition probabilities obtained by solving the set of differential equations.

3.1 The System's States

Consider a system consist of m EPs and one CP (in total $m+1$ processor). When the first job arrives, all processors are idle. So, the job can dispatch to all processors. Since we should pay processing costs for CP, we prefer to dispatch the jobs to EPs when an idle one is available. The state of the system is shown by a vector consist of m binary value and an integer one at the end of the vector, such as $[x_1\ x_2\ \dots\ x_m\ x_{m+1}]$. In this string, the first m elements are binary variables that define EPs' condition, and the last element is an integer variable that defines the condition of the CP. The value of $x_1;\ i = 1, \dots, m+1$, define the number of the job which is allocated to the processors. For example, for a system consist of three EPs and one CP, state $[0\ 1\ 1\ 3]$ is the state that EP number one is idle, EP number 2 and 3 are working, and three jobs are allocated to the CP.

3.2 Transition Rates

At the beginning of the process, the system is in the state $[0\ 0\ \dots\ 0\ 0]$. The model starts allocating the jobs to the processor from the first idle one. So, the first job will be allocated to EP number one, and the system moves to state $[1\ 0\ \dots\ 0\ 0]$. The model consists of two types of transitions. The first type is related to job dispatching between the processors with the transition rate of λ (which is the rate of the arrival jobs). The second type of transition is related to the job's processing which depends on the job's size (d) and the processor's speed (p_j) and calculated as $\mu_j = p_j/d$. Consider the state $[x_1\ x_2\ \dots\ x_m\ x_{m+1}]$. Based on the values of $x_j;\ j = 1, \dots, m+1$ the system will move to different other system states. Consider that y is a variable that refers to the place of the first zero on the system's state. Now consider that the system moves to state $[x`_1\ x`_2\ \dots\ x`_m\ x`_{m+1}]$ during a transition. Based on the values of x_i, two different cases may happen as follows:

- **Case 1: At least on zero is in the first m elements of the system's state**

In this case, the value of $x`_j$ and the transition rates are provided in table 1.

Table 1: Transition rates and the values of $x`_j$ for case one.

Transition rate	**Values of $x`_j$**

λ	$x`_y = 1$ $x`_j = x_j$ for all values of j except y
$x_k.\mu_k;\ \ for\ \ k = \{1, \dots, m+1\} - \{y\}\ if\ x_k \neq 0$	$x`_k = x_k - 1$ $x`_j = x_j$ for all values of j except k

- **Case 2: There is no zero is in the first m element of the system's state**

In this case, the value of $x`_j$ and the transition rates are provided in table 2.

Table 2: Transition rates and the values of $x`_j$ for case two.

Transition rate	**Values of $x`_j$**
λ	$x`_{m+1} = x_{m+1} + 1$ $x`_j = x_j$ for $j = 1, \dots, m$
$x_k.\mu_k;\ \ for\ \ k = \{1, \dots, m\}\ and\ (k = m+1\ if\ x_{m+1} \neq 0)$	$x`_k = x_k - 1$ $x`_j = x_j$ for all values of j except k

3.3 Set of Differential Equations

For determining the set of differential equations between the system's states, a simple rule is available. Consider that the current state of the system is shown by S. The set of states that the state S is reachable from them is shown by A, and the set of states that are reachable from state S is shown by B. Consider that A consist of a states that shown by $(A_1, \dots, A_a)$, and B consist of b states that shown by $(B_1, \dots, B_b)$. The state-space for this state is presented in Figure 1.

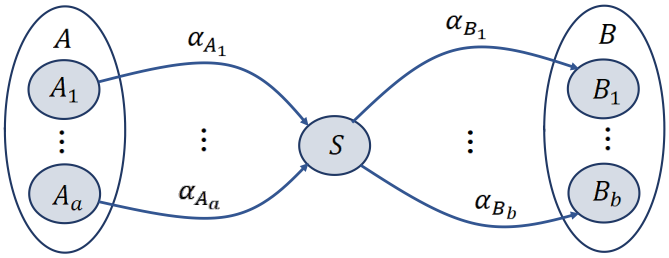

Figure 1: state-space diagram of state S.

In Figure 1, α_{A_i} is the transition rate from the state A_i to state S, and α_{B_i} is the transition rate from state S to state A_i. For the state S, the differential equation is calculated in Equation (1) as follows:

$$P'_S + \left(\sum_{i=1}^{b} \alpha_{B_i}\right) \times P_S = \sum_{i=1}^{a} \left(\alpha_{A_i} \times P_{A_i}\right). \qquad (1)$$

Considering the same logic, we can calculate the set of differential equations between the system's states.

3.4 The Processors' Performance Levels

By solving the set of differential equations, the real-time probability of each state can be calculated. In each state, if $x_j = 1$, it means that the processor number j is working. So, the probability that the processor number j is working calculates by multiplying the value of x_j to the state probability and summing up for all system's states. The working probability of each processor is equal to the performance level of the

processor. For example, if for a specific processor, the working probability at a fixed time is equal to p, it means that the processor $\%100 \times p$ of time is working. So, the usage performance of the processor is equal to $\%100 \times p$.

4 NUMERICAL EXAMPLE

To show how to use the above-mentioned method in a simplified way and calculate the processors' performance level, we focused on a system consisting of two EPs and one Cloud. The arrival rate of the job is equal to $\lambda = 0.2$ which means that the expected value between two consecutive arrivals is equal to 5 seconds. To have a fair comparison between the queueing-based and simulation (which will be provided in Section 4.1) techniques, we consider that the size of the jobs has an Exponential distribution with an average of 1000 kilobytes for the simulation method. The processors' performance speed is considered 100, 50, and 500 kilobytes per second, respectively. So, the processing rate (service rate) of the processors is equal to $\mu_1 = 100/1000 = 0.1$, $\mu_2 = 50/1000 = 0.05$, and $\mu_3 = 500/1000 = 0.5$, respectively. The states of the system for a maximum of 4 under-processed jobs are presented in Table 3. For more than four under-processed jobs, we can use the same logic of the last row of Table 3 to determine the system's states.

Table 3: States of the system for a maximum of four under-processed jobs.

Number of under-processed jobs	States
0	[0 0 0]
1	[1 0 0], [0 1 0], [0 0 1]
2	[1 1 0], [1 0 1], [0 1 1], [0 0 2]
3	[1 1 1], [1 0 2], [0 1 2], [0 0 3]
4	[1 1 2], [1 0 3], [0 1 3], [0 0 4]

The state-space diagram of the instance is presented in Figure 2. In this figure, all blue arrows define the transitions due to the job's dispatching with the transition rate of λ.

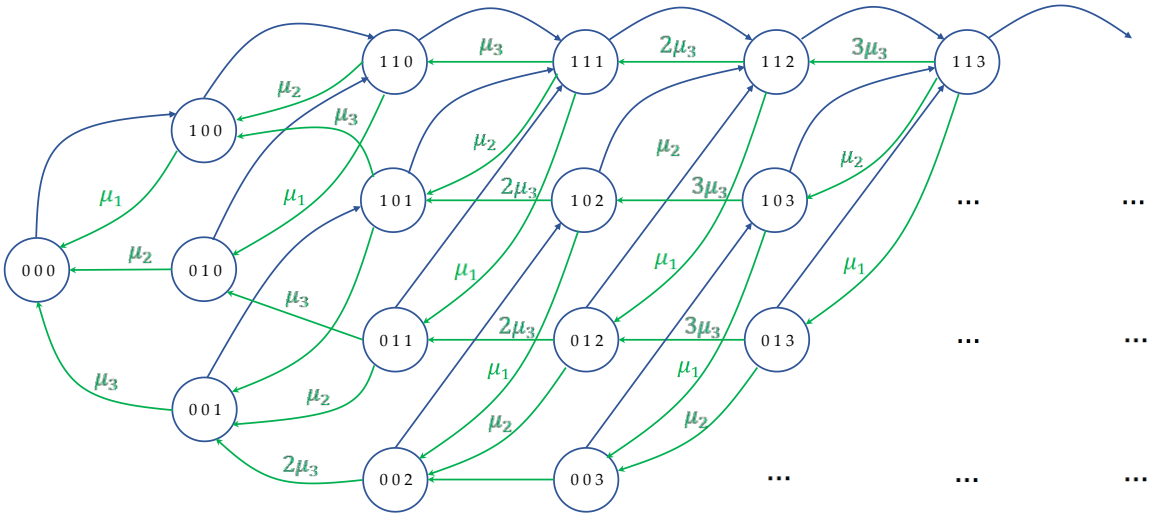

Figure 2. The state-space diagram of a system consists of two EPs and one CP.

Considering the state-space diagram of Figure 2 and the transition rates between the system's status, the differential equation set between the system's rates is presented in Equations (2) to (6).

$$P'_{[000]} + \lambda \times P_{[000]} = 0 \tag{2}$$

$$P'_{[00x]} + (\lambda + x \times \mu_3) \times P_{[00x]} = \mu_1 \times P_{[10x]} + \mu_2 \times P_{[0x1]} + (x+1) \times \mu_3 \times P_{[00(x+1)]} \tag{3}$$

$$P'_{[01\,x]} + \left(\lambda + \mu_2 + x \times \mu_3\right) \times P_{[01\,x]} = \mu_1 \times P_{[11x]} + \left(x+1\right) \times \mu_3 \times P_{\left[01\,(x+1)\right]} \tag{4}$$

$$P'_{[10\,x]} + \left(\lambda + \mu_1 + x.\mu_3\right) \times P_{[10x]} = \lambda \times P_{[00x]} + \mu_2 \times P_{[11x]} + \left(x+1\right) \times \mu_3 \times P_{\left[10\,(x+1)\right]} \tag{5}$$

$$P'_{[11x]} + \left(\lambda + \mu_1 + \mu_2 + x \times \mu_3\right) \times P_{[11x]} = \lambda \times \left(P_{[01x]} + P_{[10x]} + P_{\left[11\,(x-1)\right]}\right) + \left(x+1\right) \times \mu_3 \times P_{\left[11\,(x+1)\right]}. \tag{6}$$

The performance level of all processors can be calculated using Equations (7) to (9) as follows:

$$Pl_{EP_1} = \sum_{x=0}^{\infty} \left(P_{[10\,x]} + P_{[11\,x]}\right) \tag{7}$$

$$Pl_{EP_2} = \sum_{x=0}^{\infty} \left(P_{[01\,x]} + P_{[11\,x]}\right) \tag{8}$$

$$Pl_{CP} = \sum_{x=1}^{\infty} \left(P_{[00\,x]} + P_{[01\,x]} + P_{[10\,x]} + P_{[11\,x]}\right). \tag{9}$$

The range of x in equations (3) to (7) is $x = 1, \ldots, \infty$. For solving the set of differential equations presented above, we codded all equations in MATLAB R2021a on a laptop with Intel(R) Core (TM) i7-4500U CPU @ 1.80GHz 2.39 GHz laptop with 8.00 GB RAM. For this reason, we cute-of the set of equations for $x \geq 10$ and calculates the probabilities for the set of equations with $x \leq 9$. The results show that this cutting-off has no effects on the states' probabilities, and the sum of the states' probabilities for the states with $x \leq 9$ still is an equal one. We run the codes for $t = 0, 10, 20, \ldots, 1000$, and the performance levels for different values of t are presented in Figure 3.

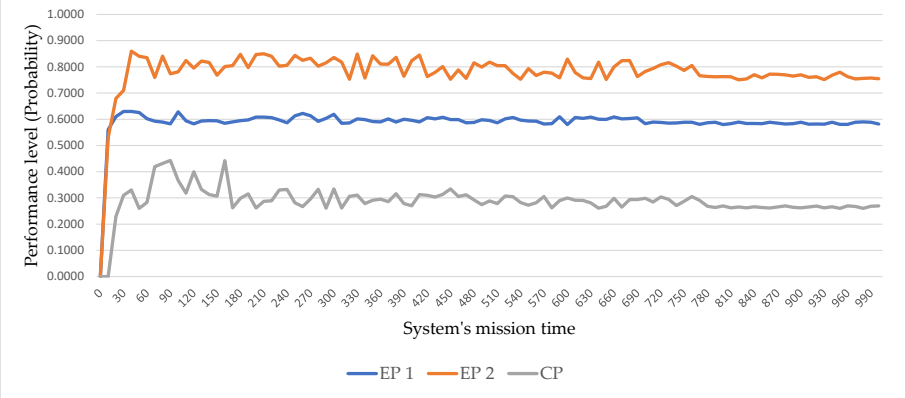

Figure 3: Performance levels of the processors.

As shown in figure 3, the EP1, EP2, and CP's performance levels are equal to 0.5851, 0.7649, and 0.2624, respectively. It means that EP1 is processing the jobs in %58.51 of the time, while this value for EP2 and EC is equal to %76.49 and %26.24. In this instance, the priority of jobs' dispatching is the EP with the highest processing speed. If we want to dispatch the job processors with the lowest processing speed, we need to order the EPs ascendingly. The new strategy results are equal to %68.58 for EP1, %58.61 for EP2, and %31.33 for CP. The total computational times for calculating the processors' performance level for all values of t were less than 1 second for both instances.

4.1 Comparing the Proposed Model with Simulation

To compare the presented model with simulation, we first solved the same instances by a simulation technique. We coded the simulation model in MATLAB R2021a and ran each instance 100000 times. The flowchart of the used simulation technique is provided in figure 4.

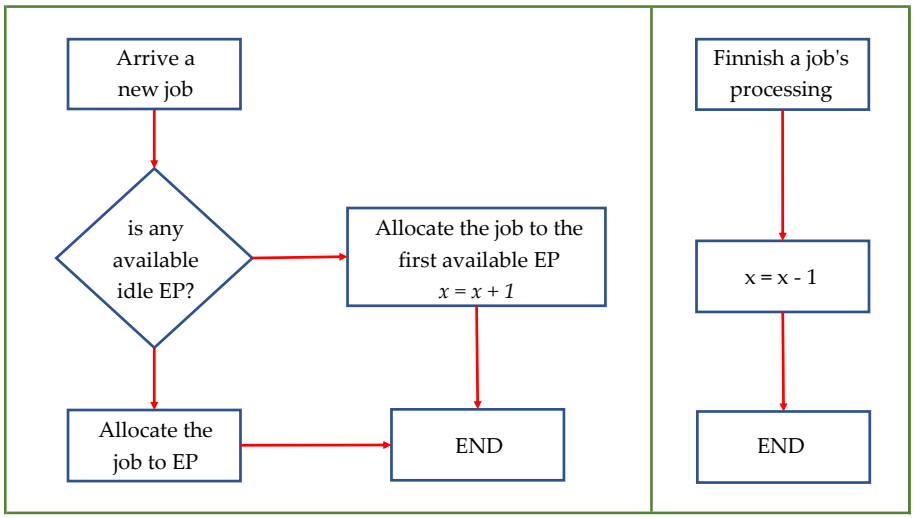

Figure 4: The flowcharts of the used simulation technique.

The primary instance is the FIFO dispatching strategy when the jobs are dispatched among idle processors based on their descending processing speed (from highest to lowest). After 1000 hours of the system's mission time, the simulation technique results are equal to %59.13 for EP1, %76.83 for EP2, and %25.80 for CP. The computational time for this instance is equal to 7.42 seconds. The second solved instance is the FIFO dispatching strategy when the jobs are dispatched among idle processors based on their ascending processing speed (from lowest to highest). After 1000 hours of the system's mission time, the simulation technique results are equal to %68.13 for EP1, %59.09 for EP2, and %32.02 for CP. The computational time for this instance is equal to 8.18 seconds. When comparing the queuing model and simulation, the queuing system is more favorable if the problem is simple. For complex queuing systems such as video analytics discussed above, simulation (more technically known as discrete-event simulation) has been used most of the time.

To better compare these two techniques in terms of average computational times, we compare ten different examples with 3 to 30 EPs. The parameters of these ten examples, such as the EPs and CP processing rate, the average time interval between two consecutive arrivals, and the average size of the jobs, are produced randomly. The computational time of these examples is presented in table 4. moreover, the detailed result of the example with 20 EPs is provided in table 5. Although we did not provide the performance level of the processors in these ten examples, the results for both techniques were equal considering two decimal numbers. For the example with 20 Eps, the initial parameters are as $\lambda = 10$, the average size of the jobs is considered as 2000 kilobytes, and the CP's performance level is equal to $\mu_{CP} = 1$. The performance level of the EPs is provided in Table 5.

Table 4: The average computational time of both methods in solving ten examples.

Number of EPs	Average computational times (Seconds)	
	Queueing-based	Simulation
3	1.13	35.03
4	1.31	41.73
5	1.43	44.50
6	1.71	100.77
7	2.17	127.21
8	3.13	144.89
9	5.51	189.90

10	8.16	255.86
20	27.13	1155.61
30	86.14	4816.16

As is reported in Table 4, the computation time of the queueing-based technique is significantly lower than the simple simulation-based technique.

Table 5: Processing rate and performance level (%) of the processor for the example with ten Eps.

EP	1	2	3	4	5	6	7	8	9	10
μ	0.500	0.460	0.435	0.415	0.395	0.375	0.330	0.320	0.295	0.275
Performance level	95.22	95.18	95.06	94.87	94.70	94.51	94.72	94.49	94.48	94.45
	11	12	13	14	15	16	17	18	19	20
μ	0.245	0.225	0.210	0.200	0.180	0.140	0.125	0.110	0.080	0.050
Performance level	94.67	94.75	94.81	94.71	94.94	95.82	96.01	96.30	97.18	98.18

The performance level of CP for the solved problem is equal to 82.92%. Moreover, as is mentioned before, the output of both techniques was equal to two decimal numbers. In this work, our results show that the proposed queueing-based model can calculate the exact value of the processor's performance level in a significantly less computational time compared with the simulation technique.

5 CONCLUSION AND FURTHER STUDIES

This paper was inspired by the lack of comparisons between analytical and simulation methods for evaluating the smart cities' video analytics applications. The simplified system used for smart cities consists of m heterogeneous edge processors and one backup cloud processor. There is no queue for the edge processors in the simplified system, and the jobs that arrive and face no idle edge processors are allocated to the cloud processor. A novel queuing-based method was presented to model the simplified system. In the queuing model, we proposed the general formulas for calculating the set of differential equations between the system's states. Then we solved this set of differential equations and calculated the performance level of all processors. Finally, we used a simple event-driven simulation technique for the simplified model for comparing the proposed model. We solved two scenarios of the first-in-first-out (FIFO) dispatching strategy and showed that the queueing-based method could calculate the exact values of the processors' performance level same as a simulation but in a significantly less processing time.

This work's novelty compares the new queuing-based model presented for a video analytics application in smart cities and compares it with a simulation-based approach. The results show that the presented model can achieve similar results in a short amount of time.

For future studies, two-direction can be considered. The first direction is related to the real-time video analytics with hard deadline characteristics used for smart cities. We considered the video applications with the jobs that can be divided into the subtasks and submitted to the edge and Cloud (Pudasaini and Abhari, 2019; Pudasaini and Abhari, 2020). We plan to use the proposed queuing system when dividing a job into sub-tasks that may provide better performance. The second direction considers the scalability of the systems with much more processing nodes and examining the smart cities applications beyond the video analytics when having latency between their units to draw the problem nearer to the real-time IoT 5G network of smart cities applications.

Acknowledgments
This research was undertaken, in part, thanks to funding from Ryerson University, Faculty of Science Dean's Research Fund for Post-Doctoral Fellowship (DRF-PDF), and the Canada Research Chairs Program.

REFERENCES

Ananthanarayanan, G., P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. 2017. "Real-Time Video Analytics: The Killer App for Edge Computing". *Computer*, Vol 50(10), pp. 58-67.

Bitam, S., S. Zeadally, and A. Mellouk. 2018. "Fog Computing Job Scheduling Optimization Based on Bees Swarm". *Enterprise Information Systems*, Vol 12(4), pp. 373-397.

Cheng, H., Z. Su, N. Xiong, and Y. Xiao. 2016. "Energy-Efficient Node Scheduling Algorithms for Wireless Sensor Networks Using Markov Random Field Model". *Information sciences*, Vol 329, pp. 461-477.

Dash, A. R., and S. K. Samantra. 2016. "An optimized round robin CPU scheduling algorithm with dynamic time quantum". *arXiv preprint arXiv*:1605.00362.

Doan, M. P., H. H. Huynh, and H. X. Huynh. 2019. "A Scalable IoT Video Data Analytics for Smart Cities". *EAI Endorsed Transactions on Context-aware Systems and Applications*, Vol 6(19), pp. 1-9.

Ghaleb, M., S. Taghipour, M. Sharifi, and H. Zolfagharinia. 2020. "Production and Maintenance Scheduling for a Single Degrading Machine with Deterioration-Based Failures". *Computers & Industrial Engineering*, Vol 143, 106432.

Grammenos, A., T. Charalambous, and E. Kalyvianaki. 2021. "CPU Scheduling in Data Centers Using Asynchronous Finite-Time Distributed Coordination Mechanisms". *arXiv preprint arXiv*:2101.06139.

Grillo, S., A. Pievatolo, and E. Tironi. 2015. "Optimal storage scheduling using Markov decision processes". *IEEE Transactions on Sustainable Energy*, Vol 7(2), pp. 755-764.

Hamayun, M., and H. Khurshid. 2015. "An Optimized Shortest Job First Scheduling Algorithm for CPU Scheduling". *Journal of Applied Environmental and Biological Sciences*, 5(12), 42-46.

Harki, N., A. Ahmed, and L. Haji. 2020. "CPU scheduling techniques: A review on novel approaches strategy and performance assessment". *Journal of Applied Science and Technology Trends*, 1(2), pp. 48-55.

Hicham, G. T., and E. Lotfi. 2017. "Comparative Study of Neural Networks Algorithms for Cloud Computing CPU Scheduling". *International Journal of Electrical and Computer Engineering*, Vol 7(6), pp. 3570.

Kumari, R., V. K. Sharma, and S. Kumar. 2017. "Design and Implementation of Modified Fuzzy Based CPU Scheduling Algorithm". *arXiv preprint arXiv*:1706.02621.

Ma, X., H. Gao, H. Xu, and M. Bian. 2019. "An IoT-Based Task Scheduling Optimization Scheme Considering the Deadline and Cost-Aware Scientific Workflow for Cloud Computing". *EURASIP Journal on Wireless Communications and Networking*, Vol 2019(1), pp.1-19.

Pudasaini D. and A. Abhari. 2019. "Scalable Pattern Recognition and Real Time Tracking of Moving Objects". *2019 Spring Simulation Conference (SpringSim19)*. pp. 1-12. Arizona, Tucson, United States, IEEE.

Pudasaini, D. and A. Abhari. 2020. "Scalable Object Detection, Tracking and Pattern Recognition Model Using Edge Computing". *SpringSim 2020 Virtual Conference*. Edited by F. J. Barros, X. Hu, H. Kavak, and A. A. Del Barrio. pp. 1-10. Fairfax, Virginia, United States, IEEE.

Ramezani, F., J. Lu, J. Taheri, and F. K. Hussain. 2015. "Evolutionary Algorithm-Based Multi-Objective Task Scheduling Optimization Model in Cloud Environments". *World Wide Web*, Vol 18(6), pp. 1737-1757.

Sharifi, M., and S. Taghipour. 2020. "Joint Optimizing the Production Sequence and Maintenance Plan for a Single-Machine Multi-Failure System". In *2020 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1-6. Palm Springs, California, USA, IEEE.

Sharifi, M., and S. Taghipour. 2021. "Optimal Production and Maintenance Scheduling for a Degrading Multi-Failure Modes Single-Machine Production Environment". *Applied Soft Computing*, 107312.

Sharifi, M., A. Abhari, and S. Taghipour. 2021. "Modeling Real-time Application Processor Scheduling for Fog Computing", *ANNSIM'21, July 19-22, 2021, Fairfax, VA, USA* (Accepted)

Sharifi, M., S. Taghipour, and A. Abhari, 2021. "Inspection Interval Optimization for a k-out-of-n Load Sharing System under a Hybrid Mixed Redundancy Strategy". *Reliability Engineering & System Safety*, 107681.

Shen, L., X. Yang, J. Wang, and J. Xia. 2019. "Passive Gain-Scheduling Filtering for Jumping Linear Parameter Varying Systems with Fading Channels Based on the Hidden Markov Model". *Proceedings*

of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering, Vol 233(1), pp. 67-79.

AUTHOR BIOGRAPHIES

Dr. Mani Sharifi is a post-doctoral research fellow at both DSMP and RRMR Lab at Ryerson University. He holds a B.Sc. degree, M.Sc. degree, and Ph.D. degree in Industrial Engineering from Tehran Research and Science, Azad University, Iran. His area of interest includes Reliability and Maintenance Optimization, Scheduling, and Statistical Analysis. Email address: manisharifi@ryerson.ca

Dr. Abdolreza Abhari is a Professor in the Department of Computer Science at Ryerson University and director of the DSMP lab (http://dsmp.ryerson.ca/). He holds a Ph.D. in Computer Science from Carleton University and a B.Eng. in Computer Engineering from the Sharif University of Technology, Canada, and Iran. His research interests include data science, web social networks, AI and agent systems, network simulation, and distributed systems. Email address: aabhari@ryerson.ca

Dr. Sharareh Taghipour is an Associate Professor and the RRMR Lab Director at the Department of Mechanical and Industrial Engineering at Ryerson University. She is a Tier 2 Canada Research Chair in Physical Asset Management. She obtained her Ph.D. in Industrial Engineering from the University of Toronto. She received her B.Sc. in Mathematics and Computer Science and her M.Sc. in Industrial Engineering from the Sharif University of Technology, Iran. Email address: sharareh@ryerson.ca