Evaluating Dispatching and Scheduling Strategies for Firm Real-Time Jobs in Edge Computing

Shaik Mohammed Salman*†, Alessandro Vittorio Papadopoulos*, Saad Mubeen*, and Thomas Nolte*
*Mälardalen University, Västerås, Sweden
†ABB AB, Västerås, Sweden

***Abstract*—We consider the problem of on-arrival dispatching and scheduling jobs with stochastic execution times, inter-arrival times, and deadlines in multi-server fog and edge computing platforms. In terms of mean response times, it has been shown that size-based scheduling policies, when combined with dispatching policies such as join-shortest-queue, provide better performance over policies such as first-in-first-out. Since job sizes may not always be known apriori, prediction-based policies have been shown to perform reasonably well. However, little is known about the performance of prediction-based policies for jobs with firm deadlines. In this paper, we address this issue by considering the number of jobs that complete within their deadlines as a performance metric and investigate, using simulations, the performance of a prediction-based shortest-job-first scheduling policy for the considered metric and compare it against scheduling policies that prioritize based on deadlines (EDF) and arrival times (FIFO). The evaluation indicates that in under-loaded conditions, the prediction-based policy is outperformed by both FIFO and EDF policies. However, in overloaded scenarios, the prediction-based policy offers slightly better performance.**

I. Introduction

Edge computing architectures allow real-time system designers to deploy algorithms on high-performance edge computing servers to improve response times and reduce energy consumption in embedded devices [1]–[3]. In addition, these edge deployments may need to support several different devices necessitating a multi-server system design. For such multi-server systems, dispatching and scheduling algorithms are important in satisfying response time requirements [4]. Specifically, when the objective is mean response times and the processing times of individual jobs are known apriori, multi-server variants of shortest-remaining-processing-time (SRPT) and preemptive shortest job first (SJF) are optimal under a central queue assumption [5]. Similarly, when jobs are dispatched on arrival, Mitzenmacher et al. [6] showed that SRPT and SJF outperformed the FIFO policy when combined with the join-shortest-queue (JSQ) dispatching policy. A limitation of these policies is that they require the knowledge of processing times of each job before its completion, which may not always be possible [7]. As an alternative, usage of predicted values has been investigated to achieve improved performance in terms of mean response times compared to size-oblivious policies such as first-in-first-out (FIFO) ordering [6], [8]. In a large-scale queuing system, Mitzenmacher [8] studied the impact of single-bit predictors that can indicate whether a job's processing time is above or below some threshold. They found that such predictors can provide benefits similar to those achievable with knowledge of exact processing times for Poisson arrivals and certain processing time distributions. While mean response time can be a useful measure for certain applications, some real-time applications impose constraints in terms of deadlines, where the output is only valid if it is generated before its deadlines. However, they can tolerate some missed deadlines [9]. Additionally, some of these algorithms may exhibit stochastic behavior in terms of processing times as well as inter-arrival times(For example, See [10], [11]) The performance evaluation of scheduling policies based on predicted processing times for such workloads with firm deadline requirements has received limited attention.

We address this by first considering the presence of a dual-bit job size predictor that can classify an incoming job into one of the four job size classes: small, medium, large, and very large. We choose a dual-bit prediction approach based on the intuition that the accuracy of such predictors may be better than the accuracy of predictors estimating individual job processing times. Secondly, we use the information the dual-bit predictor provides with a preemptive shortest-job-class-first (PSJF) scheduling policy. This policy orders jobs according to their job size classes, and jobs in each class are in FIFO order. Furthermore, we consider an on-arrival dispatching policy and leave evaluation of central queue approaches for future work.

Concretely, we formulate the following questions and address them using simulations.

1) What is the impact of on-arrival job acceptance and rejection based on response-time estimation of pending jobs on achievable throughput under various load conditions?
2) How does a dual-bit prediction-based PSJF scheduling policy compare against EDF, FIFO, and SRPT scheduling policies in terms of achievable throughput under various load conditions?
3) What is the impact of on-arrival server selection policy on achievable throughput under various load conditions when using dual-bit prediction-based PSJF scheduling policy?

Our evaluation indicates that prediction-based PSJF provides no significant advantage over FIFO and EDF scheduling policy for the considered settings and under-loaded scenarios. However, under fully loaded and sustained overload conditions, it performs better than FIFO and EDF when used with an estimation-based admission policy.

II. Related Work

In on-arrival dispatching systems, dispatching policies determine the server on which an incoming job will be executed. Dispatching policies such as JSQ and its variants that require the knowledge of the number of pending jobs in each server is optimal with respect to mean response times [4]. However, gathering this information may introduce overheads depending on the number of servers and the network traffic. Alternatively, policies such as round-robin (RR) that do not require knowledge of pending jobs dispatch incoming jobs in a cyclic order. Consequently, they do not have the overhead associated with policies that require information about the pending jobs on each server. Several other policies, such as join-the-idle-queue and join-below-threshold, have been proposed to balance the trade-off between overheads and response times [12], [13].

Many studies have explored the potential for enhancing algorithm performance through machine-learned advice or predictions, including classical algorithms for online scheduling and load-balancing [14]. These prediction-augmented algorithms have been evaluated through competitive analysis under both accurate and possibly incorrect predictions [6], [8], [13], [15]. In online scheduling, some researchers have considered predicting job execution times [13], [16] and the ordering of jobs [15]. Mitzenmacher et al. [6] demonstrated via simulations that the benefits of using predictions in large distributed systems were retained if the predictions were reasonably precise. Based on the evaluations, they proposed selecting servers with the least number of pending jobs and using the predicted shorted processing job first policy for use in actual systems. Zhao et al. [13] extended the RMLF algorithm to use predicted job execution times. The prediction enhanced algorithm achieved performance close to that of SRPT when the prediction error was small and better than RMLF when the error was large. However, designing highly accurate predictors that predict the exact size of a job may be difficult. Consequently, we consider dual-bit predictors that coarsely classify an incoming job into one of the four distinct job classes.

In single-server settings, Gao et al. [17] developed scheduling strategies for firm semi-periodic real-time tasks. They introduced three control parameters to decide at run-time whether to interrupt a job before its deadline and considered four admission policies. Our work differs in that we consider a multi-server setting and estimate response times using the job execution time distribution and the number of pending jobs on a specific server on job arrival and allowing admitted jobs to stay in the queue until completion or reaching their deadline.

Several works within queuing theory analyzed the performance of EDF under different scenarios. Abhaya et al. solve a set of linear equations to calculate the mean delay for M/G/1/./EDF [18]. Kargahi et al. [19] provide bounds for estimating the deadline miss probabilities for M/M/k/./NEDF+G and M/M/1./EDF+G assuming a single queue system, unlike the dispatch on arrival policy we considered. Kargahi provided an analytical method in [20] to show the performance of parallel EDF queues for JSQ, the minimal expected value of unfinished work(MED), and threshold-based dispatching strategy but without any arrival time acceptance or rejection.

We previously evaluated the performance of a dispatching policy that relies on single-bit predictions of job processing times in conjunction with a non-preemptive FIFO scheduling policy where all jobs had a fixed relative deadline. [21]. Similarly, we evaluated the percentage of missed deadlines when jobs have individual deadlines using preemptive EDF policy in [22]. The work in the paper differs from our previous work in that we consider a prediction-based policy and compare it with EDF and FIFO ordering.

III. System Model

A. Job Model

Jobs arrive online following a Poisson process with an arrival rate λ adjusted accordingly to generate desired system load. We assume a Poisson process since this has been extensively used in the literature in analyzing queuing systems and models quite well the use case we consider. i.e., requests for job executions can arrive from several users, and each such user may have different inter-arrival times. The processing time of each job is drawn from a trimmed and discretized exponential distribution with a mean of 10 and lower bounded with value 1 and upper bounded by value 100. (i.e., ten times the mean value). While this may not be a realistic representation of many real-world workloads, it has been widely used in the literature in queuing systems, and the loss of accuracy compared to true exponential distribution may be acceptable since we only consider the average throughput rather than numerically precise response times. Additionally, each job's relative deadline d_i is drawn from a uniform distribution D with a range between five to ten times the mean value of the execution time distribution. We choose the considered ranges due to negative results associated with low laxity systems [15]. The relative deadline is revealed when the job arrives. Each job is assigned an absolute deadline on its arrival.

B. Server Model

We consider a network of homogeneous servers. Each server has its own queue and executes jobs assigned to it according to the considered scheduling policy. When a job arrives, a dispatching policy selects a server and accepts or rejects the job based on the pending workload on the selected server. Each accepted job is added to the queue of the selected server. The jobs in the queue are ordered based on the scheduling policy. If a new job has a shorter deadline than the currently executing job, the scheduler preempts that job, adds it to its own queue, and starts executing the new job.

Server Selection Policy: We evaluate JSQ and RR server selection policies. Under JSQ, whenever a new job arrives, the server with the least number of pending jobs is selected, while in RR, the server is selected cyclically. We combine these policies with an online schedulability test to enable admission time control. If a job is deemed to be schedulable,

it is immediately sent to the selected server and is rejected otherwise.

Admission Policy: A schedulability test decides whether a job can be successfully scheduled on a server, given a scheduling policy and information about the pending jobs on the server. As jobs in our system are bound by a deadline, we must determine whether the job can meet its deadline on the selected server. Utilization-based schedulability tests that rely on worst-case processing time values can be used if the service cannot tolerate any deadline miss. However, if over-provisioning is a problem and deadline misses are tolerated, a low-overhead but less accurate test may be useful. If the jobs satisfy such a schedulability test, they are accepted into the system.

We now describe the policies used to admit or reject the jobs. We consider three policies based on how the job processing times are considered, (i) mean-approximation policy and (ii) clairvoyant policy, and (iii) admit-all policy.

Mean-approximation policy: In this policy, we use mean μ of the processing time distribution to estimate the response time f_i of a newly arrived job on the selected server i. If the number of pending jobs on this server is given by N_i, the estimated response time is given by

$$f_i = (N_i + 1) \cdot \mu. \tag{1}$$

Clairvoyant policy: In this policy, we assume the knowledge of exact processing times. The response time f_i on any server i is given by

$$f_i = x_j + \sum_{k=0}^{N_i} x_k, \tag{2}$$

where x_k is the exact processing time of each job k assigned to server i and x_j is the processing time of the newly arrived job.

For both the estimation methods, the admission test returns true if the following condition is satisfied:

$$f_i \leq d_i. \tag{3}$$

Admit-all policy: This policy dispatches each incoming job to a selected server and does not reject any job.

Scheduling Policies: We consider four different policies depending on the workload parameters, (i) FIFO policy, prioritizing jobs based on their arrival times, (ii) EDF, prioritizing jobs based on their deadlines (iii) PSJF, prioritizing jobs based on their predicted processing times and (iv) SRPT, prioritizing jobs based on their true remaining processing time.

- FIFO policy prioritizes jobs according to their arrival times, with ties broken arbitrarily. It executes jobs in a non-preemptive manner.
- EDF policy prioritizes jobs according to their absolute deadlines with preemptions. i.e., a newly added job can preempt a running job if its absolute deadline is lower than that of the running job.
- SRPT policy prioritizes jobs according to their remaining processing times with preemptions. Although this policy requires the exact processing time to be known, we use this policy as a baseline to compare against the size-aware prediction-based PSJF policy.
- PSJF policy prioritizes jobs according to their predicted job classes, with jobs from the shortest class executed first with preemptions enabled. Jobs within each class are in FIFO order.

IV. Simulation Methodology

We compare the performance of the scheduling policies for (trimmed) exponentially distributed job processing times, Poisson inter-arrival times, and uniformly distributed deadlines. We simulate 10000 time units and measure throughput as the ratio of the number of jobs completed within their deadlines and the total number of jobs released during this simulation interval. We repeat the simulation over ten runs and present the average throughput calculated over these ten runs.

Load generation: We generate jobs based on the load following Eq. (4), where ρ is the load, μ is the mean processing time, and λ is Poisson arrival rate per server, and n is the number of servers. Specifically for the results presented in this paper, we fix μ to 10, set n to 2 and 4, and calculate λ for fixed values of load ρ. Specifically, we set ρ between 0.5 and 1.2 and incremented in steps of 0.1.

$$\rho = \frac{\mu}{n \cdot \lambda} \tag{4}$$

Assigning processing times: For each job, we first assign the true processing time generated according to the trimmed exponential distribution with μ set to ten. The lower bound is set to one, and the upper bound is set to one hundred. As we use integer time units for stepping through the simulation, we use the $ceil$ function to convert the generated floating point value to an integer. We note that this discretization changes the mean of the distribution and deviates from the true exponential distribution. However, considering a large sample size, i.e., a hundred thousand samples, this deviation can be ignored without significantly impacting the observed results.

Assigning predicted processing times: As we assume a dual-bit predictor, which classifies jobs into four distinct classes, defining the thresholds that can guide the classification is necessary. As a first approach, we consider quartile values of the exponential distribution and use them as threshold values. Once the true processing time has been assigned, we assign the predicted processing time equal to the discrete version of the quartile values. Specifically, we use values 3, 7, 17, and 47, the rounded quartile values of the trimmed exponential distribution with a mean of 10.

Assigning deadlines: To assign the relative deadline, we first generate a random number from a uniform distribution of a range of 5 to 10. This generated random number is multiplied by the mean of the processing time distribution, and the result is assigned as the deadline. If the true processing time value is greater than the assigned deadline, we reset the processing time to the mean of the distribution while retaining the deadline. We do this since this allows us to

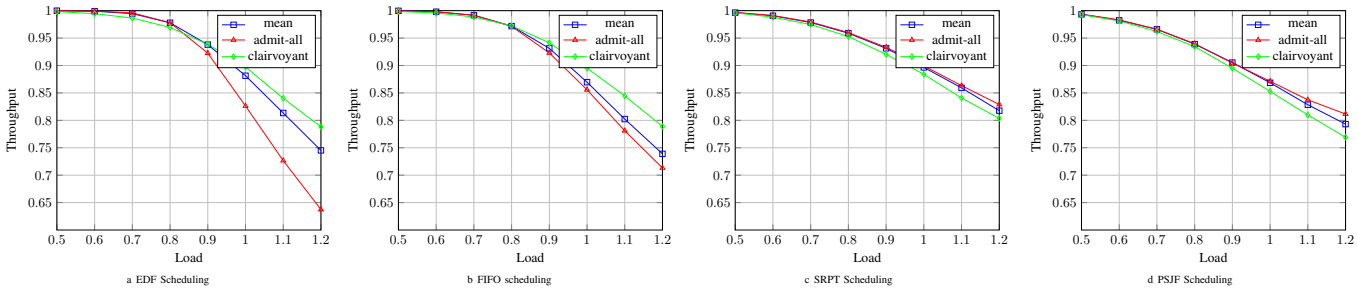

Fig. 1. Comparison of throughput under three different admission policies for round-robin dispatching and various scheduling policies

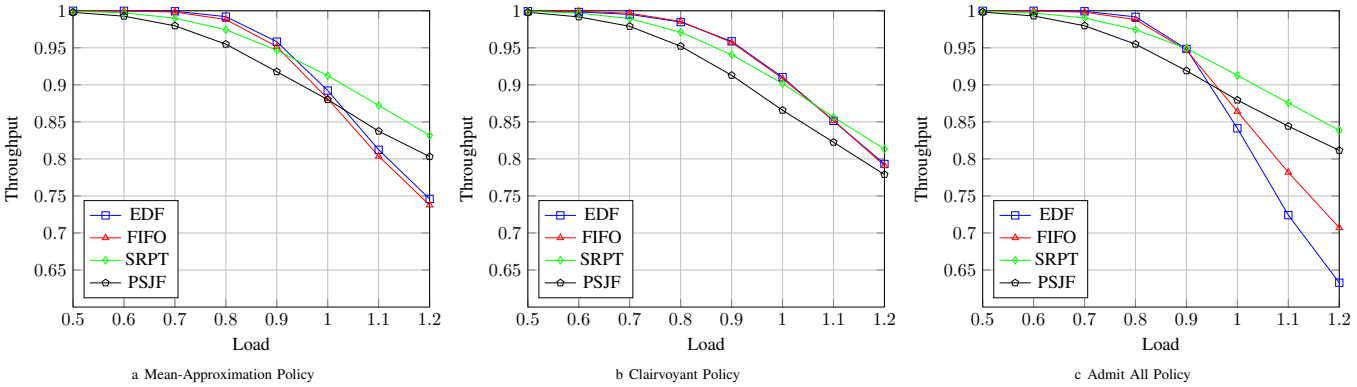

Fig. 2. Comparison of throughput under different scheduling and admission policies for RR dispatching

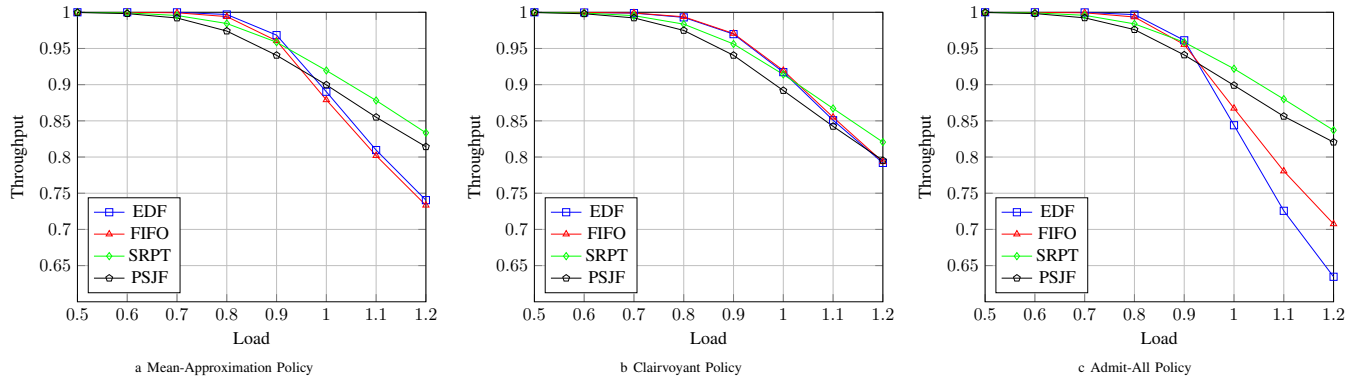

Fig. 3. Comparison of throughput under different scheduling and admission policies for JSQ dispatching

keep the generated jobs, as it eliminates the problem of having unschedulable jobs even before they are released.

V. Evaluation

A. Performance of Admission Policies

We compared the performance of the scheduling policies with and without admission control policies. As a baseline, we considered a clairvoyant policy that knows the exact size of each pending job on a chosen server. The estimated response time of an incoming job is then calculated using eq.(2). We also considered a low complexity mean approximation policy where the response time is estimated using eq.(1). In addition to this, we considered the scenario where all incoming jobs are admitted and dispatched to specific servers depending on the dispatching policy. The achieved throughput with RR dispatching when using two servers is shown in Fig. 1. When using EDF as the scheduling policy, rejecting jobs using the admission control policies has a limited impact on achievable throughput when the load is below 0.9. However, the benefits of admission control are seen when the load is increased to 1.0, with even the mean-approximation policy achieving a six percent higher average throughput than the admit-all policy. The difference, however, is reduced under the FIFO scheduling policy. Similarly, when considering SRPT and the prediction-based PSJF policy, the impact of the admission control policies remains negligible until a load value of 0.9. When the load increases to 1.0 and above, admitting all jobs and mean-approximation policy-based admission control provide slightly better throughput than the clairvoyant policy. Based on these observations, we can conclude that on-arrival

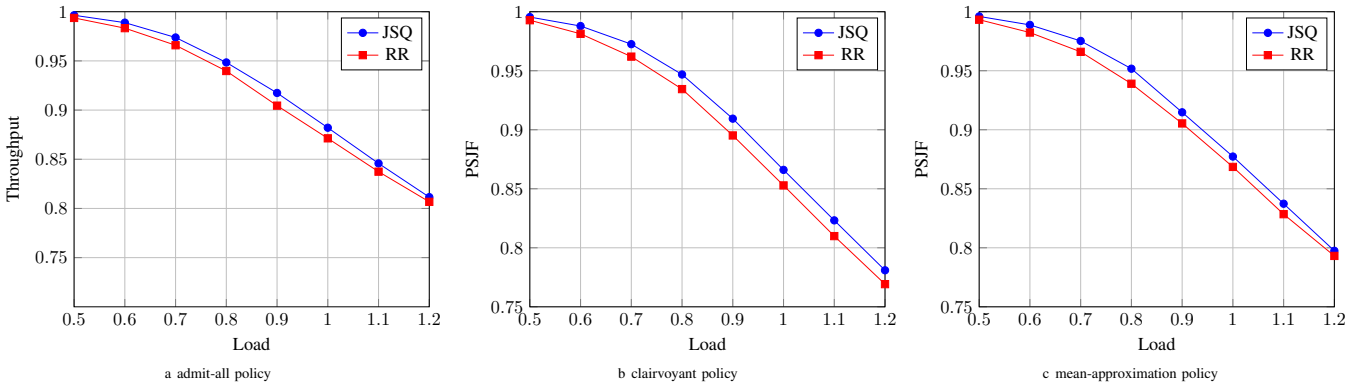

Fig. 4. Comparison of throughput under different dispatching schemes for PSJF scheduling policy

admission control policies provide no significant advantage when the system load is below 0.9 when combined with any of the considered scheduling policies. If the system load is above 0.9, admission control policies perform better when combined with size-oblivious EDF and FIFO scheduling policies. When combined with size-aware PSJF and SRPT, it is better to admit all jobs rather than reject some jobs to achieve a slightly better average throughput.

B. Impact of Scheduling Policy

Fig. 2 and Fig. 3 provide a comparison of the achieved throughput of the considered scheduling policies when combined with different admission control policies for RR and JSQ dispatching, respectively. When using the mean approximation policy and admit-all policy, we can observe that as load increases, the throughput reduces for all the scheduling algorithms. When the load is below 0.9, size-oblivious scheduling policies perform slightly better than the baseline SRPT policy, while the prediction-based PSJF performs the worst. However, when the load is close to 1.0, all the policies perform similarly. Under overload conditions, PSJF performs better than both EDF and FIFO, while SRPT performs the best. The performance of EDF is worst under the admit-all policy. When using the clairvoyant policy, PSJF performs worst at high loads, with EDF and FIFO performing better than SRPT.

C. Performance of Dispatching Policies

We compared the impact of JSQ and RR dispatching policies on the performance of various scheduling and admission control policies. Fig. 4 shows the throughput achieved with PSJF scheduling and various admission control policies. We observe that the difference in performance due to the dispatching policy is almost negligible, with JSQ only slightly outperforming RR. This is seen consistently across all load values. This behavior is also consistent for size-aware and size-oblivious scheduling policies, irrespective of the admission policy.

D. Discussion

We compared the performance of various combinations of admission control and scheduling policies, including a coarse prediction-based shortest job policy for applications whose jobs exhibit variability in inter-arrival times, processing times, and deadlines. The results show that for non-asymptotic conditions, i.e., when the number of servers is limited to two and four, PSJF and SRPT provide no advantage over EDF and FIFO policies when the load is below 0.9. However, size-aware policies provide better throughput under overloaded conditions, with even the coarse-grained PSJF performing better than both FIFO and EDF when combined with mean approximation and the admit-all admission policies. Additionally, JSQ and RR perform similarly for all combinations with no significant difference in performance. This indicates that the overheads of JSQ can be avoided with very little loss by choosing RR. Moreover, suppose it can be established that the load in the system will stay below 0.9. In that case, the simple FIFO policy with an admit-all policy can be used instead of the other policies with relatively higher computational complexities. Another interesting observation is that using size-aware policies for admission control and scheduling provides worse performance. However, it must be noted that we only considered the exponential distribution for processing time distribution, and the observations may not apply to a different distribution. In addition to this, we assumed an ideal predictor that is fully accurate with zero inaccurate job size classifications. This may not be realizable in practice, and further investigation is needed to study the impact of inaccuracies. Another missing parameter is network communication and the lack of consideration of end-to-end deadlines, which may be important in practical systems.

VI. Conclusion

We considered the problem of on-arrival dispatching and scheduling firm real-time jobs in multi-server settings. We evaluated the performance of the predicted job scheduling policy using coarse-grained predictions in terms of average throughput using simulations under non-asymptotic conditions. Our evaluation shows that the prediction-based size-aware policy does not offer significant benefits compared to policies that prioritize based on deadlines or arrival times in under-loaded conditions. However, in overloaded scenarios, it performs slightly better than other policies. In summary, our

study provides valuable insights into selecting dispatching and scheduling policies for edge computing systems to meet the needs of firm real-time applications.

REFERENCES

[1] J. Zilic, A. Aral, and I. Brandic, "Efpo: Energy efficient and failure predictive edge offloading," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019, pp. 165–175.

[2] J. Zilic, V. De Maio, A. Aral, and I. Brandic, "Edge offloading for microservice architectures," in *Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–6. [Online]. Available: https://doi.org/10.1145/3517206.3526266

[3] M. Jansen, A. Al-Dulaimy, A. V. Papadopoulos, A. Trivedi, and A. Iosup, "The spec-rg reference architecture for the edge continuum," 2022. [Online]. Available: https://arxiv.org/abs/2207.04159

[4] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[5] I. Grosof, Z. Scully, and M. Harchol-Balter, "Srpt for multiserver systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 2, pp. 9–11, 2019.

[6] M. Mitzenmacher and M. Dell'Amico, "The supermarket model with known and predicted service times," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, pp. 2740–2751, 11 2022.

[7] M. Harchol-Balter and Z. Scully, "The most common queueing theory questions asked by computer systems practitioners," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 3–7, 2022.

[8] M. Mitzenmacher, "Queues with small advice," in *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*. SIAM, 2021, pp. 1–12.

[9] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[10] M. Alcon, H. Tabani, L. Kosmidis, E. Mezzetti, J. Abella, and F. J. Cazorla, "Timing of autonomous driving software: Problem analysis and prospects for future solutions," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 267–280.

[11] G. C. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*. IEEE Comput. Soc, 2-4 Dec. 1998, pp. 286–295.

[12] X. Zhou, F. Wu, J. Tan, Y. Sun, and N. Shroff, "Designing low-complexity heavy-traffic delay-optimal load balancing schemes: Theory to algorithms," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–30, 2017.

[13] Y. Zhao, R. Zhou, and H. Zeng, "Design optimization for real-time systems with sustainable schedulability analysis," *Real-Time Systems*, vol. 58, no. 3, pp. 275–312, 2022.

[14] M. Purohit, Z. Svitkina, and R. Kumar, "Improving online algorithms via ml predictions," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[15] A. Lindermayr and N. Megow, "Permutation predictions for non-clairvoyant scheduling," in *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 357–368. [Online]. Available: https://doi.org/10.1145/3490148.3538579

[16] Z. Scully, I. Grosof, and M. Mitzenmacher, "Uniform bounds for scheduling with job size estimates," *arXiv preprint arXiv:2110.00633*, 2021.

[17] Y. Gao, G. Pallez, Y. Robert, and F. Vivien, "Dynamic scheduling strategies for firm semi-periodic real-time tasks," *IEEE Transactions on Computers*, vol. 72, pp. 55–68, 1 2023.

[18] V. Gamini Abhaya, Z. Tari, P. Zeephongsekul, and A. Y. Zomaya, "Performance analysis of edf scheduling in a multi-priority preemptive M/G/1 queue," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2149–2158, 2014.

[19] M. Kargahi and A. Movaghar, "A method for performance analysis of earliest-deadline-first scheduling policy," *Journal of Supercomputing*, vol. 37, no. 2, pp. 197–222, 2006.

[20] ——, "Dynamic routing of real-time jobs among parallel edf queues: A performance study," *Computers & Electrical Engineering*, vol. 36, no. 5, pp. 835–849, 2010.

[21] S. M. Salman, V. L. Dao, S. Mubeen, A. Papadopoulos, and T. Nolte, "Scheduling firm real-time applications on the edge with single-bit execution time prediction," in *26th International Symposium On Real-Time Distributed Computing (ISORC)*, 2023.

[22] S. M. Salman, S. Mubeen, A. Papadopoulos, and T. Nolte, "Dispatching deadline constrained jobs in edge computing systems," in *27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2023.