Dispatching and scheduling multiserver jobs for throughput optimality

Jonatha Anselmi [a], Josu Doncel [b]

[a] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG. Grenoble 38000, France
[b] UPV/EHU, University of the Basque Country. Leioa 48940, Spain

ABSTRACT

We consider the problem of dispatching and scheduling an infinite stream of multiple classes of jobs to a set of single-server parallel queues. Each job requires the simultaneous utilization of multiple servers. Our objective is to identify a dispatching algorithm (used by a central dispatcher) and a scheduling discipline (used by each server) that induces throughput optimality, i.e., the mean response time of jobs is finite whenever the system load is less than one. It is known that this problem is not trivial even in the case where all servers share a centralized queue, i.e., no job dispatching. We show that throughput optimality can be obtained by dispatching jobs to queues according to probabilities provided that i) jobs of a given class respect some constraint on the server geometry and ii) the scheduling discipline prioritizes jobs with the largest server need. Due to a connection with the M/G/1 priority queue, we provide an exact analytical expression for the mean response time of jobs. Finally, we discuss the connection of our model with redundancy systems, where jobs do not necessarily require the simultaneous utilization of servers.

1. INTRODUCTION

In the multiserver job model, incoming jobs occupy multiple servers simultaneously for the entire job duration. The number of servers a job requires is usually referred to as *server need* of that job. Recently, the interest for this type of concurrent jobs in the context of queueing systems has increased significantly due to a recent trace of Google Borg scheduler [16], where it is shown that server needs can vary by five orders of magnitude across jobs. In existing multiserver job queueing models, jobs wait in a central queue before being processed and then leave the system permanently upon service completion. The primary challenge is to design simple scheduling policies that ensure *throughput optimality*. Roughly speaking, this means that the mean response time of jobs is finite when the system load is less than one. For instance, First-Come-First-Served is not throughput optimal in general. To see this, assume that a job that requires the utilization of one server arrives when the system is empty and, immediately after, a job that requires the utilization of all the servers arrives; all the servers except for one will be idle while the first job is in service. Within First-Come-First-Served, a sufficient condition for stability is given in [19]. In these references, the authors also show that throughput optimality is achieved in a limiting regime where the number of servers grows to infinity. More recently, other throughput optimal scheduling policies have been proposed. Examples are the family of BackFilling policies, Max-Weight and ServerFilling; see, e.g., [7, 8]. A further challenging problem related to the multiserver job model is the development of simple analytical formulas for the mean response time [9], though this objective is often out of reach. Consequently, existing analyses only focus on mean response time within practically relevant limiting regimes. These regimes offer not only approximations of mean response time in prelimit settings but also reveal structural properties, such as asymptotic optimality. In this respect, [7,8] derive simple asymptotic formulas for ServerFilling and ServerFilling-SRPT in a heavy-traffic regime, where the arrival rate approaches its nominal value while all other parameters remain constant. Also, [10] examines mean waiting times under FCFS and the Smallest-Need-First policy in the many-server heavy-traffic limit.

In this work, we address the aforementioned challenges by considering a system with parallel queues. Specifically, instead of assuming a centralized queue, we assume that each server has its own queue. This decentralized version of the multiserver job model yields the additional problem of deciding which queue a job should join upon its arrival. The authors in [15] consider that the servers have finite buffer size and incoming jobs are placed to one of the servers with enough place. They provide an algorithm that achieves throughput optimality. In this work, we assume that servers have infinite buffer size (i.e., the number of waiting customers in the queue is unbounded) and we provide a job dispatching *and* scheduling scheme that is able to achieve not only throughput optimality, but also mean response time characterization. Within the proposed model, each job is placed by a central dispatcher to servers upon arrival according to probabilities that depend on the server need of the incoming job, and each server simply prioritizes jobs with the largest server need.

The remainder of the article is organized as follows. In Section 2 we describe the model under investigation. In Section 3, we present our dispatching and scheduling scheme as well as our main results. In Section 4, we discuss the connection of our work with redundancy and bandwidth sharing systems. Finally, we draw the conclusions if this work in Section 5.

Copyright is held by author/owner(s).

2. MODEL DESCRIPTION

We consider a queueing system where an infinite stream of jobs needs to be processed by N parallel servers.

2.1 Jobs

Jobs arrive in the system following a Poisson process with rate λ and can be partitioned into C *classes*. A job is of class i with probability α_i, for all $i = 1, \dots, C$. Class-i jobs require the simultaneous possession of $n_i \leq N$ servers for a random amount of time equal in distribution to the random variable D_i, for all $i = 1, \dots, C$. After being processed, each job leaves the system.

In the following, n_i (a constant) and D_i will be respectively referred to as "server need" and "service time" of class-i jobs. Without loss of generality, we assume that server needs are decreasing.

2.2 Servers

Let $\mathcal{S} := \{1, \dots, N\}$ denote the server set. Each server works at unitary speed and has its own queue. For this reason, the terms server and queue will be used interchangeably. For now, servers are free to process jobs according to any scheduling discipline provided that they somehow coordinate among themselves in order to respect the constraint that jobs hold the simultaneous possession of multiple servers.

2.3 Dispatcher

Upon the arrival of a class-i job, a central dispatcher selects n_i servers and immediately routes it to such servers for processing – the server selection procedure is specified later. Dispatching decisions are taken instantly at job arrival times and are irrevocable.

2.4 Technical assumptions

We assume that the stochastic sequences of job interarrivals, service times, and classes are i.i.d. and independent.

Let $d_i := \mathbb{E}[D_i]$, $\rho_i := \lambda \alpha_i n_i d_i / N$ and $\rho := \sum_{i=1}^{C} \rho_i$. For the system *load* ρ, we assume that

$$\rho < 1, \tag{1}$$

which is necessary to ensure the stability of the underlying Markov process, in the sense of positive Harris recurrence. Note that (1) is the usual stability condition "the overall arrival rate is less than the overall service rate".

Though it restricts the applicability of our results, we assume that the overall number of servers and all server needs are powers of some integer $k \geq 2$. This assumption has been also considered in [8] and, furthermore, it has been observed that the specific scenario where $k = 2$ occurs in several real workloads, e.g., [1].

ASSUMPTION 1. *The number of servers N and the server needs n_i, for all $i = 1, \dots, C$, are all powers of some positive integer $k \geq 2$.*

2.5 Problem statement

Within the setting described above, a dispatching algorithm is a rule employed by the central dispatcher that selects the servers that will process each job, and a scheduling discipline is a rule employed by each server that selects a job in its queue to process at any point in time. A *Dispatching and Scheduling* (D&S) scheme is the combination of a dispatching algorithm and a scheduling discipline. In this paper, our primary objective is to identify a D&S scheme that ensures throughput optimality.

DEFINITION 1. *We say that a* D&S *scheme is* throughput optimal *if the resulting mean response time, i.e., the mean time spent by jobs in the system, is finite whenever* (1) *holds true.*

For illustration purposes, let us consider two natural examples of D&S schemes and show that they are not throughput optimal. In both examples, we assume 4 servers and 3 classes. In the following illustrations, jobs of class 1, 2 and 3 are represented in green, red and blue, respectively. Also, we assume that $n_i = 2^{3-i}$, for $i = 1, 2, 3$ and that the system is initially empty.

EXAMPLE 1 (FIRST-COME-FIRST-SERVED SCHEDULING). *Assume that the first arriving job is of class 3 and needs to be routed to Server 3. Immediately after, assume that a class-1 job arrives, which is sent to all servers. We represent in Figure 1a the behavior of this system when the scheduling First-Come-First-Served is applied. As it can be seen, jobs of class 1 do not start service before the completion of the job at Server 3. This situation clearly leads to a waste of capacity and throughput optimality cannot be guaranteed.*

EXAMPLE 2 (RANDOM DISPATCHING). *Assume that a class-2 job arrives and, immediately after, another one of the same class. Under a probabilistic routing, the first job can be placed in Server 1 and Server 2, whereas the second one can be placed in Server 2 and Server 3. In Figure 1a, we represent the behavior of this system when this occurs. As it can be seen, the job in Server 3 will not start service until the previous job finishes being served at Server 1 and Server 2. This also leads to a waste of the capacity of the system, which implies that a* D&S *scheme with a randomized dispatching is not throughput optimal.*

3. OUR APPROACH

We define a particular D&S scheme.

3.1 Dispatching and scheduling scheme

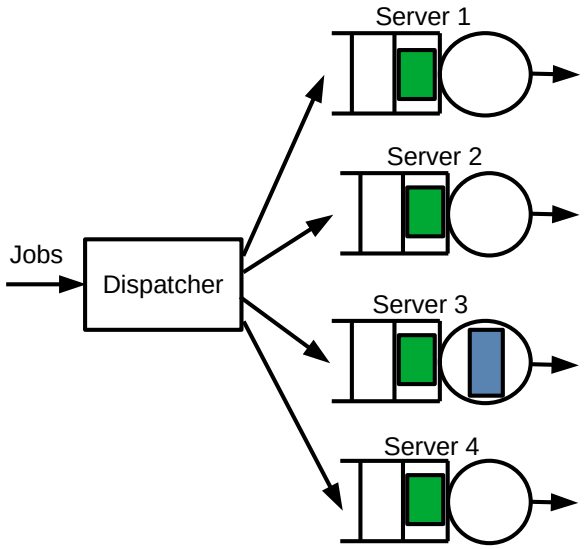
For each job class $i = 1, \dots, C$, let

$$\mathcal{S}_i := \{\{1, \dots, n_i\}, \{n_i + 1, \dots, 2n_i\}, \dots, \{N - n_i, \dots, N\}\},$$

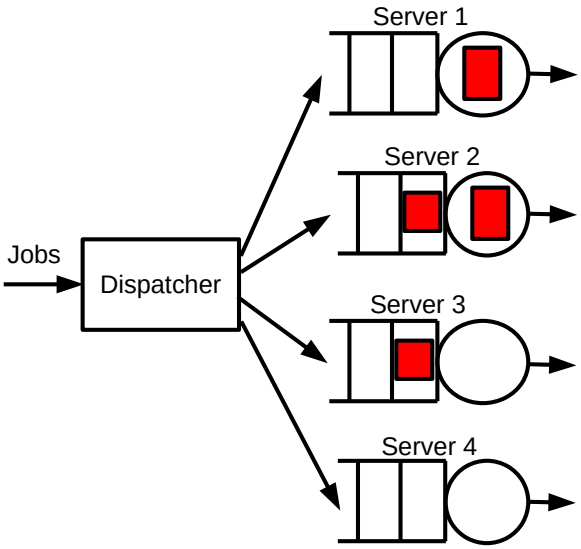
denote a partition of the server set $\mathcal{S}$. Note that each element of the partition contains exactly n_i servers, i.e., the server need of class-i jobs. Then, the proposed scheme works as follows:

1. *(Dispatching rule)* Upon arrival of a class-i job, choose randomly $s \in \mathcal{S}_i$ and send the job to the servers in s;
2. *(Scheduling rule)* Process jobs with the largest server needs first (smallest job class index), preemptively and breaking ties following the order of their arrival.

The dispatching rule above ensures that jobs are processed by specific blocks of servers and that all servers are equally



(a) The scheduling FCFS does not provide a D&S scheme that is throughput optimal.



(b) The random dispatching does not provide a D&S scheme that is throughput optimal.

Figure 1: Examples of natural dispatching and scheduling schemes.

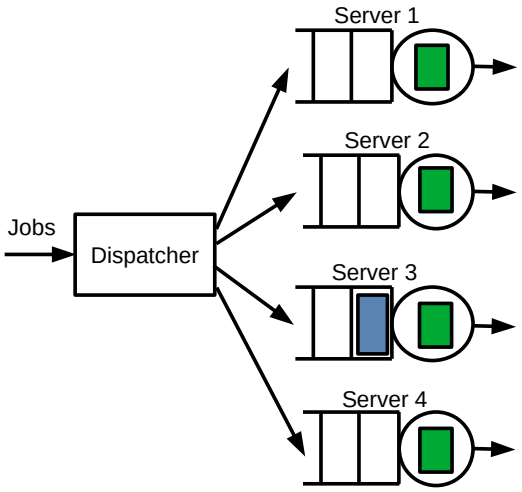


Figure 2: Our D&S scheme for the instance of Example 1.

loaded. If we endow servers with a notion of distance among them, then servers inside each block are close to each other. In serverless computing terminology, one can think of these blocks as containers (or pods) and servers as replicas [2, 3].

The performance of our D&S scheme for the instance considered in Example 1 is represented in Figure 2. Under our scheme, we remark that $\mathcal{S}_2 = \{(1,2),(3,4)\}$. Therefore, since $(2,3) \notin \mathcal{S}_2$, we have that a job of class 2 cannot be routed to Server 2 and Server 3 and, as a consequence, the behavior described in Example 2 cannot be given under our D&S scheme.

Within Assumption 1, the D&S scheme above ensures that each job holds the simultaneous possession of a number of servers that corresponds to its server need. This happens naturally without the servers having to communicate with each other. Also, the proposed scheme ensures that no processing capacity is lost whenever a job needs to be processed.

3.2 Main result

Since our scheme ensures full capacity usage at all times, each queue is work-conserving, and throughput optimality is obtained. In fact, letting R denote the mean response time induced by the above D&S scheme, we have the following more general result.

THEOREM 1. *Let Assumption 1 hold. If $\rho < 1$, then*

$$R = \sum_{i=1}^{C} \alpha_i R_i,$$

where

$$R_i := \left(\frac{\sum_{j=1}^{i} \rho_j r_j}{1 - \sum_{j=1}^{i} \rho_j} + d_i \right) \frac{1}{1 - \sum_{j=1}^{i-1} \rho_j}, \qquad (2)$$

and

$$r_j := \frac{\lambda}{2} \sum_{k=1}^{j} \alpha_k \mathbb{E}[D_k^2].$$

PROOF. The key observation is that the proposed D&S scheme ensures that each queue is an M/GI/1 queue with the preemptive-resume priority rule. This holds by Assumption 1, which allows for a "perfect nesting" of jobs inside servers. In this queue, there are C job classes. Class-1 jobs have the highest priority, class-2 jobs the second highest priority and so on. Because of the probabilistic dispatching, class-i jobs have arrival rate $\lambda \alpha_i \frac{n_i}{N}$ and service times distributed as D_i. Therefore, the load of class-i jobs is ρ_i. This queueing system has been widely studied in the literature [12]. It is well known that the mean response time of class-i jobs is given by the formula in (2) whenever $\rho < 1$. Note that r_j is the residual service time (upon arrival) of a class-j job, $j = 1, \ldots, C$. □

4. RELATION WITH REDUNDANCY AND BANDWIDTH SHARING MODELS

The proposed model admits an interpretation in the context of redundancy (or replication) systems and the bandwidth-sharing model proposed in [13]. In this section, we discuss such a relationship.

4.1 Redundancy systems

It is well-known that redundancy is a powerful technique used in computer and communication systems to decrease latency. Under this approach, upon arrival of a job, multiple replicas (or copies, clones, etc.) of the job itself are placed at different servers. The number of copies of a job is usually called the redundancy level. The service time across replicated jobs can be i.i.d. or equal. In the former case, it is said that copies are i.i.d. whereas in the latter the copies are identical. Redundant replicas may be canceled when the first replica completes service (cancel-on-completion) or when the first replica initiates service (cancel-on-start). Within some architectures and load conditions, replicas are not canceled at all as the cost of sending cancellation messages may not pay off [17]. Many researchers have been interested in analyzing the stability of redundant systems and we point to [4] for a recent survey.

Multiserver job and redundancy systems share the property that each job requires the utilization of multiple servers. However, the main difference between these models is that redundancy systems do not require to use these servers simultaneously. Under the proposed scheme, we observe that both the redundancy with identical copies and multiserver job models are identical. As a consequence, the results of this work extend to redundancy systems with identical copies, i.e., our results show that, for a redundancy system with identical copies, we can provide a dispatching and scheduling scheme that is throughput optimal and such that the mean response time of jobs is known.

The D&S scheme proposed in this work is called Most-Redundant-First and First-Come-First-Served (MRF-FCFS) in the context of redundancy systems. In [5], the authors provide partial results about the stability and performance optimality of MRF-FCFS with identical copies and a *nested* topology, which is a generalization of the topology considered in our work. Given the double interpretation of our work in terms of multiserver jobs and redundancy systems, we conclude that the result of Theorem 1 characterizes the stability condition and provides the expression of the mean response time of jobs of the MRF-FCFS policy with identical copies and a nested topology when the condition of Assumption 1 is verified.

4.2 Bandwidth sharing model

The proposed model is closely related to the well-known bandwidth-sharing model for elastic traffic introduced in [13]. To illustrate this connection, consider fixing the dispatching rule to a uniformly random policy. In this case, the model presented here becomes a special instance of the bandwidth-sharing model, where each server represents a link in a network and each multi-server job functions as a flow occupying multiple links simultaneously.

Over the past decades, extensive research has explored the performance of the bandwidth-sharing model, including throughput optimality, mean response time, and other metrics. For example, [14] demonstrates that the bandwidth-sharing model with general job-size distributions under an alpha-fair scheduling policy achieves maximal stability at the fluid scale. Additionally, studies on heavy-traffic moments and the job count distribution for this model, under proportionally fair scheduling, have been conducted in [11, 18], though they often assume phase-type service times. A notable feature of the proportionally fair policy is its insensitivity to job size distribution variance in the heavy-traffic regime, a feature not typically seen in policies like FCFS or Largest-Need-First. For more details, we point the reader to [6, 18].

The above studies focus on network capacity allocations within the class of weighted α-fair allocations, aiming to extend the behavior of Processor Sharing to a network context. In our model, however, network capacity is not shared; each job occupies the full capacity of required resources during processing. This distinction makes our model structurally different. Additionally, a significant limitation of the bandwidth-sharing model is the lack of an exact formula for mean response time, contrasting with our approach. On the other hand, a limitation of the proposed approach is that insensitivity does not hold, as it is clear from Theorem 1.

5. CONCLUDING REMARKS AND FUTURE WORK

We have investigated job dispatching and scheduling in a parallel queueing system where jobs require the simultaneous possession of multiple servers. For this scheme, we show that it achieves throughput optimality and we provide an analytical expression of the mean response time of jobs. As observed in Section 4, the proposed model has a double interpretation, for which we have the following concluding remarks:

- Within the multiserver job interpretation, to the best of our knowledge, our model is the first to consider the decentralized case where each server has its own queue, which opposes to the more common case where all jobs share a central queue [8, 9].

- Within the job replication interpretation, the proposed scheme does not need the exchange of control messages between servers. Also, it should be clear that Theorem 1 provides an upper bound on the mean response time obtained if the underlying system would implement the cancel-on-start replica-cancellation policy.

As future work, we would like to explore the optimal D&S scheme in the multiserver job model. Another interesting is to analyze under which conditions the performance of redundancy systems and the multiserver job model coincide. Finally, we believe that an interesting research line consists of studying the performance of load-balancing techniques in parallel server systems for the multiserver job model.

6. ACKNOWLEDGEMENTS

This work has been partially funded by the Department of Education of the Basque Government through the Consolidated Research Group MATHMODE (IT1456-22) and the Grant PID2023-146678OB-I00 funded by the Spanish Ministry of Science, Innovation and Universities.

7. REFERENCES

[1] The San Diego Supercomputer Center (SDSC) SP2 log. https://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_sp2/index.html. Accessed: 2024-04-14.

[2] Knative docs v1.3. https://knative.dev/docs/, 2024. Online; accessed: 2024-08-21.

[3] Knative scale bounds. https://kubernetes.io/docs/concepts/, 2024. Online; accessed: 2024-08-21.

[4] E. Anton, U. Ayesta, M. Jonckheere, and I. M. Verloop. A survey of stability results for redundancy systems. In *Modern Trends in Controlled Stochastic Processes: Theory and Applications, V. III*, pages 266–283. Springer, 2021.

[5] E. Anton, R. Righter, and I. M. Verloop. Efficient scheduling in redundancy systems with general service times. *Queueing Systems*, 106(3):333–372, 2024.

[6] Y. Fu and R. J. Williams. Stability of a subcritical fluid model for fair bandwidth sharing with general file size distributions. *Stochastic Systems*, 10(3):251–273, 2020.

[7] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf. WCFS: a new framework for analyzing multiserver systems. *Queueing Syst. Theory Appl.*, 102(1–2):143–174, oct 2022.

[8] I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Optimal scheduling in the multiserver-job model under heavy traffic. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(3), dec 2022.

[9] M. Harchol-Balter. The multiserver job queueing model. *Queueing Syst. Theory Appl.*, 100(3–4):201–203, apr 2022.

[10] Y. Hong and W. Wang. Sharp waiting-time bounds for multiserver jobs. In *Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '22, page 161–170, New York, NY, USA, 2022. Association for Computing Machinery.

[11] W. N. Kang, F. P. Kelly, N. H. Lee, and R. J. Williams. State space collapse and diffusion approximation for a network operating under a fair bandwidth sharing policy. *The Annals of Applied Probability*, 19(5):1719 – 1780, 2009.

[12] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. Wiley, 1976.

[13] L. Massoulié and J. W. Roberts. Bandwidth sharing and admission control for elastic traffic. *Telecommun. Syst.*, 15(1–2):185–201, Nov. 2000.

[14] F. Paganin, A. Tang, A. Ferragut, and L. L. H. Andrew. Network Stability Under Alpha Fair Bandwidth Allocation With General File Size Distribution. *IEEE Transactions on Automatic Control*, 57(3):579–591, March 2012.

[15] K. Psychas and J. Ghaderi. Randomized algorithms for scheduling multi-resource jobs in the cloud. *IEEE/ACM Transactions on Networking*, 26(5):2202–2215, 2018.

[16] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes. Borg: the next generation. In *Proceedings of the fifteenth European conference on computer systems*, pages 1–14, 2020.

[17] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Low latency via redundancy. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, page 283–294, New York, NY, USA, 2013. Association for Computing Machinery.

[18] W. Wang, S. T. Maguluri, R. Srikant, and L. Ying. Heavy-traffic insensitive bounds for weighted proportionally fair bandwidth sharing policies. *Math. Oper. Res.*, 47(4):2691–2720, Nov. 2022.

[19] W. Wang, Q. Xie, and M. Harchol-Balter. Zero queueing for multi-server jobs. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(1), feb 2021.