Replication vs speculation for load balancing

Jonatha Anselmi

Queueing Systems Vol. 100
Special Issue: 100 views on queues

Replication vs Speculation for Load Balancing

Jonatha Anselmi

February 16, 2022

1 Introduction

In standard, or canonical, load balancing, a central dispatcher receives a stream of exogenous work units (jobs) and, immediately upon their arrival, sends each of them to one out of a number of distributed computational resources (servers). Upon service completion at its designated server, each job leaves the system. The main goal is to design load balancing algorithms that optimize some performance metric, e.g., the user-perceived delay, while ensuring an efficient use of resources, e.g., bandwidth, energy, etc. The underlying architecture is fully decentralized: there is no queueing at the central dispatcher and each server has its own queue. In cloud computing systems, this is motivated by the stringent latency requirements of modern applications, which can run on top of systems with thousands of servers. In this setting, scalability is a major concern. Popular examples of load balancing algorithms are Power-of-d and Join-the-Idle-Queue among several others [8].

Over the last decades, standard load balancing proved itself as a central theme in queueing theory. Nowadays, researchers and practitioners have taken a step forward and, on top of standard load balancing, they are embedding *replication* techniques to further improve user-perceived delays. In this respect, there are two underlying principles: either replicate, i.e., "*replicate a job $d \geq 1$ times upon its arrival and use the results from whichever replica responds first*" [4], or speculate, i.e., "*replicate a job $d \geq 1$ times as soon as the system detects it as a straggler*" [2]; here, a 'straggler' refers to a job taking longer than expected to complete. Within the former, which we refer to as "replication", all redundant replicas are usually canceled as soon as one completes or starts service. Within the latter, which we refer to as job "speculation" [10], a job is usually replicated following a *timeout rule*. Both principles have been successfully introduced to mitigate the impact on system performance of heavy-tailed job service

Jonatha Anselmi
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France. E-mail: jonatha.anselmi@inria.fr

times. From a practical standpoint, a service time may be large either because its size is inherently large, meaning that it requires a lot of computational resources, or because it encountered some unfortunate run-time phenomena that slowed or stalled its execution. There is increased interest in replication as it is employed by Google's BigTable, while the speculation principle is used in Apache Spark and Hadoop MapReduce.

2 Problem Statement

The replication and speculation principles above admit several implementations, all of which can be viewed as generalizations of standard load balancing: when the number of additional replicas of a given job boils down to zero and when the timeout in the timeout rule for speculation is set to infinity, the standard load balancing framework is recovered. It is then natural to investigate the performance induced by both replication and speculation strategies and compare their benefit with respect to their standard load balancing configuration; as we discuss below, there may not be any benefit at all! This is a challenging problem in queueing theory and a satisfactory understanding is currently missing. One of the technical difficulties consists in the stochastic analysis of the underlying Markov process, as one should keep track of the locations of all replicas of a given job. This makes the state description complex. In addition, the interesting case is when the service times of all replicas are *dependent* random variables, as one expects that the service time of a large (in size) job will remain large no matter how many times it is replicated [5]. Matter of fact, unless one consider service times with an exponential distribution, even a satisfactory characterization of the stability region induced by popular replication mechanisms is currently unclear; see [5,6,7] for further details.

2.1 Recent work on speculation

Recent theoretical works on speculation focus on static settings with a finite number of jobs and no queueing [1,9]. The queueing and stability behavior of job speculation is not well-understood. To the best of our knowledge, the first queueing network model for job speculation in a stochastic and dynamic setting has been proposed in [3], where the case $d = 1$ is considered. Here, each job is initially assigned to a single server by a frontend dispatcher. Then, when its execution begins, the server sets a timeout. If the job completes before the timeout, it leaves the network, otherwise the job is terminated and either relaunched or resumed at another server where it will complete. Concerning stability, it is found that under mild assumptions and also within heterogeneous servers, the underlying Markov process is positive Harris recurrent under the usual stability condition [3, Theorem 1], i.e., the nominal load at each queue is less than one. The key observation is that the resulting stability region differs from a queueing network implementing a standard load balancing scheme such as Power-of-d. Within classes of service time distributions of practical interest, it is then found that speculation can increase the stability region of the network when compared with standard load balancing and replication schemes, and a recipe for the design of optimal timeouts is also given.

2.2 Replication vs speculation vs standard load balancing

To the best of our knowledge, the first work that jointly compares *i)* replication, *ii)* speculation and *iii)* their corresponding standard load balancing configuration in a stationary regime is [3]. It is found that none of these three approaches always performs better than the others: the general answer strongly depends on the load conditions and on the level of variability in the job service times. In a light-load regime, replication provides better average response times. Under moderate to heavy loadings and in presence of heavy-tailed server slowdown, simulations indicate that speculation can significantly improve performance when compared with existing redundancy schemes and standard load-balancing systems. On the other hand, in the presence of light-tailed server slowdowns however, it is found that speculation performs worse than its corresponding standard load balancing configuration (where all timeouts are set to infinity).

3 Discussion

The replication and speculation principles discussed provide effective mitigation techniques for the straggler problem but which one is preferable under which conditions is not clear as the former may lead to significant additional resource costs while the latter may increase latency. In addition, both principles can be implemented in several ways. For instance, within speculation, a replicated job can be either resumed or re-executed from scratch, and the straggling job may be killed or not. Also, one is allowed to choose the number of extra replicas, where to place them and, for each of them, whether a timeout rule should be adopted again or not, etc. All of these questions/variations have not been addressed in the stationary regime. In our view, this will lead to a plethora of queueing models, most of which are yet to come in the queueing literature.

References

1. M. F. Aktaş and E. Soljanin. Straggler mitigation at scale. *IEEE/ACM Transactions on Networking*, 27(6):2266–2279, Dec 2019.
2. G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. Reining in the outliers in map-reduce clusters using mantri. In *Proc. of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, page 265–278, USA, 2010. USENIX Association.
3. J. Anselmi and N. Walton. Stability and optimization of speculative queueing networks. *IEEE/ACM Transactions on Networking*, pages 1–12, 2021.
4. J. Dean and L. A. Barroso. The tail at scale. *Commun. ACM*, 56(2):74–80, Feb. 2013.
5. K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. V. Houdt. A better model for job redundancy: Decoupling server slowdown and job size. *IEEE/ACM Transactions on Networking*, 25(06):3353–3367, nov 2017.
6. Y. Raaijmakers and S. Borst. Achievable stability in redundancy systems. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), Nov. 2020.
7. Y. Raaijmakers, S. Borst, and O. Boxma. Redundancy scheduling with scaled Bernoulli service requirements. *Queueing Systems*, 93(1):67–82, Oct 2019.
8. M. van der Boor, S. C. Borst, J. S. van Leeuwaarden, and D. Mukherjee. Scalable load balancing in networked systems: A survey of recent advances. *arXiv preprint arXiv:1806.05444*, 2018.
9. D. Wang, G. Joshi, and G. W. Wornell. Efficient straggler replication in large-scale parallel computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 4(2), Apr. 2019.
10. M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving MapReduce performance in heterogeneous environments. In *Osdi*, volume 8, page 7, 2008.