Scalable Load Balancing in the Presence of Heterogeneous Servers

Kristen Gardner[1], Jazeem Abdul Jaleel[2], Alexander Wickeham[2], Sherwin Doroudi[2]
[1]Department of Computer Science, Amherst College, Amherst, MA
[2]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN

1. INTRODUCTION

In large-scale computer systems, deciding how to dispatch arriving jobs to servers is a primary factor affecting system performance. Consequently, there is a wealth of literature on designing, analyzing, and evaluating the performance of load balancing policies. For analytical tractability, most existing work on dispatching in large-scale systems makes a key assumption: that the servers are homogeneous, meaning that they all have the same speeds, capabilities, and available resources. But this assumption is not accurate in practice. Modern computer systems are instead heterogeneous: server farms may consist of multiple generations of hardware, servers with varied resources, or even virtual machines running in a cloud environment. Given the ubiquity of heterogeneity in today's systems, it is critically important to develop load balancing policies that perform well in heterogeneous environments. In this paper, we focus on systems in which *server speeds* are heterogeneous.

The dominant dispatching paradigm in the contemporary literature on large scale systems is the "power of d choices," wherein a fixed number (d) of servers are queried at random, and a dispatching decision is made among these servers. Unfortunately, the "power of d" policies that have been designed to perform well in homogeneous systems, such as Join-the-Shortest-Queue-d (JSQ-d) [3, 4] and Shortest-Expected-Delay-d (SED-d) can lead to unacceptably poor performance in the presence of heterogeneity (see Figure 1).

Our key insight is that there are two decision points at which "power of d" policies can use server speed information: first, when choosing which d servers to query, and second, when deciding where among the queried servers to send an arriving job. Alone, neither decision point appears to be enough to both ensure stability and achieve good performance. In combination, they allow for the design of a new class of powerful policies that benefit from server speed heterogeneity. We propose two new families of policies, called JIQ-(d_F,d_S) and JSQ-(d_F,d_S), that are inspired by classical "power of d" policies but use server speed information at both decision points. This enables them to outperform JSQ-d, SED-d, and other heterogeneity-aware policies in certain settings, as well as to maintain the full stability region.

2. MODEL

Our system consists of k heterogeneous servers. There are two classes of servers: k_F of the servers are "fast" servers and $k_S = k - k_F$ of the servers are "slow" servers. We let $q_F = \frac{k_F}{k}$ and $q_S = \frac{k_S}{k} = 1 - q_F$ denote the fraction of servers that are fast and slow respectively. Service times are independent with rate μ_F on fast servers and rate μ_S on slow servers, where the speed ratio $\mu_F/\mu_S > 1$. We assume that $\mu_F q_F + \mu_S q_S = 1$, so the system has total capacity k.

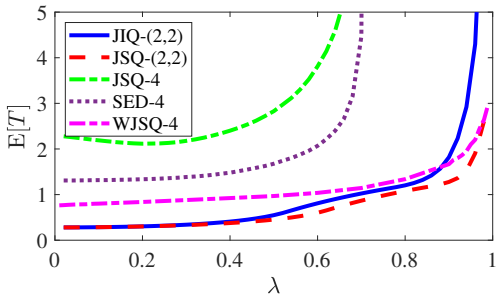

Figure 1: Mean response time as a function of λ under five policies, where $\frac{\mu_F}{\mu_S} = 10$ and $q_F = 0.2$.

Jobs arrive to the system as a Poisson process with rate λk. Upon arrival to the system, a job is dispatched immediately to a single server according to some policy. Each server works on the jobs in its queue in first-come first-served (FCFS) order.

We propose two families of dispatching policies: JIQ-(d_F, d_S) and JSQ-(d_F, d_S). Both families favor idle fast servers whenever possible, and both leverage the idea that slow servers are occasionally worth utilizing, and it is better to utilize them when idle rather than busy. Both policy families are parameterized by d_F and d_S, as well as by probabilities p_F and p_S; each setting for these four parameters defines a different specific policy within the family.

DEFINITION 1. *Under both **JIQ-**(d_F, d_S) and **JSQ-**(d_F, d_S), when a job arrives the dispatcher queries d_F fast servers and d_S slow servers, chosen uniformly at random without replacement. The job is then dispatched to one of the queried servers as follows:*

- *If any of the d_F fast servers are idle, the job begins service on one of them.*
- *If all d_F fast servers are busy and any of the d_S slow servers are idle:*
  - *With probability p_S the job begins service on an idle slow server.*

– *With probability* $1-p_S$ *the job is dispatched to a* ***chosen*** *fast server among the* d_F *queried.*

- *If all* $d_F + d_S$ *queried servers are busy:*
  - *With probability* p_F *the job is dispatched to a* ***chosen*** *fast server among the* d_F *queried.*
  - *With probability* $1-p_F$ *the job is dispatched to a* ***chosen*** *slow server among the* d_S *queried.*

The difference between the two policies lies in how a busy server (among those under consideration) is ***chosen****. Under JIQ-(d_F,d_S) the server is chosen uniformly at random. Under JSQ-(d_F,d_S) the server with the shortest queue is chosen, with ties broken uniformly at random.*

3. ANALYTICAL APPROACH

We first study *stability*, the property that each server experiences an average arrival rate less than its service rate; because all servers are work conserving in our model, this notion of stability is equivalent to the property that each server is idle a nonzero fraction of the time. This property is a necessary condition for achieving finite mean response time. We show that both policies achieve the maximum possible stability region, assuming that p_F and p_S are chosen optimally, and then derive more specific necessary and specific conditions for stability as $\lambda \to 1$. These results allow for generally distributed service times.

THEOREM 1. *Under both JIQ-(d_F,d_S) and JSQ-(d_F,d_S), for any values of* $d_F, d_S \geq 1$*, there exist choices of* p_F *and* p_S *such that the system is stable for* $\lambda < \mu_F q_F + \mu_S q_S = 1$.

THEOREM 2. *As* $\lambda \to 1$*, the system is unstable if* $p_F \neq \mu_F q_F$*, and the system is stable if* $p_F = \mu_F q_F$ *and* $p_S \geq \mu_S q_S$.

We next analyze the queue length distribution and mean response time under both JIQ-(d_F, d_S) and JSQ-(d_F, d_S). We assume that $k \to \infty$ and that in this limiting regime all servers' queue lengths become independent. This is a common assumption in the analysis of large-scale systems that has been proved for several related systems (see, e.g., [1]), and it allows us to treat a single queue as its own isolated system. We provide a brief sketch of our approaches to analyzing both policies:

- For JIQ-(d_F, d_S), we use a mean field approach and consider a tagged fast server and a tagged slow server in isolation. We derive λ_{IF}, λ_{IS}, λ_{BF}, and λ_{BS}, respectively the arrival rates to the tagged fast and slow servers when idle and when busy. We observe that the dynamics of a busy fast (respectively, slow) server are identical to those of an M/G/1 system with arrival rate λ_{BF} and the same service time distribution as in our system. This allows us to apply the Pollaczek-Khinchine formula to obtain the mean response times at the tagged fast and slow servers.
- For JSQ-(d_F, d_S), we assume exponentially distributed service times. We consider a tagged arrival and condition on whether the tagged arrival ultimately runs on a fast server or a slow server and on whether it has to wait in the queue. We formulate a system of differential equations for $f_i(t)$ (respectively, $s_i(t)$), the fraction of fast (respectively, slow) servers that have at least i jobs at time t. Solving this system of equations allows us to find the queue length distributions at fast and slow servers, and, from this, the mean response time for the tagged arrival.

For both policy families, we minimize mean response time by optimizing over p_F and p_S, assuming a fixed d_F and d_S.

4. RESULTS AND DISCUSSION

We compare mean response time under JIQ-(d_F,d_S) and JSQ-(d_F,d_S) to that under three other policies (results for our policies are analytical, while results for the following policies are simulated):

- Under **JSQ-**d, the dispatcher queries d servers uniformly at random and sends the job to the queried server with the shortest queue.
- Under **SED-**d, the dispatcher queries d servers uniformly at random and sends the job to the queried server at which it has the shortest expected delay.
- Under **WJSQ-**d (Weighted JSQ-d), the dispatcher queries d servers, where the probability that a server is queried is proportional to that server's speed, and sends the job to the queried server with the shortest queue [2].

We note that JSQ-d is heterogeneity-unaware, SED-d only uses heterogeneity information when dispatching, and WJSQ-d only uses heterogeneity information when querying.

Figure 1 illustrates our results in one particular case, when $\frac{\mu_F}{\mu_S}$ is high and q_F is low. Here, both JSQ-d and SED-d can lead to apparent instability. While WJSQ-d avoids instability at high load, performance under WJSQ-d still suffers at low load. In contrast, our policies always remain stable, and in some cases achieve better performance, by differentiating between fast and slow servers both when querying and when choosing where to dispatch among the queried servers.

Our work establishes that, in order to achieve good performance in heterogeneous systems, "power of d" dispatching policies should use heterogeneity information at two decision points: (1) when choosing which servers to query, and (2) when choosing where among the queried servers to dispatch a job. Ultimately, how best to distribute jobs among fast and slow servers depends jointly on the system load, the fraction of servers that are fast, and the relative speeds of the servers. Because there is no single right answer, policies designed for heterogeneous systems must be able to adapt to the system parameters. JIQ-(d_F,d_S) and JSQ-(d_F,d_S) do this by optimizing over the probabilistic parameters to choose the best allocation of jobs to fast and slow servers.

5. REFERENCES

[1] M. Bramson, Y. Lu, and B. Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Systems*, 71(3):247–292, 2012.

[2] H. Chen and H.-Q. Ye. Asymptotic optimality of balanced routing. *Operations research*, 60(1):163–179, 2012.

[3] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[4] N. Vvedenskaya, R. Dobrushin, and F. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Informatsii*, 32(1):20–34, 1996.