Optimal Service Elasticity in Large-Scale Distributed Systems

Debankur Mukherjee et al.

Eindhoven University of Technology

Analysis by Noah Johnson

College of Electrical Computer and Biomedical Engineering, University of Rhode Island

Contents

I. INTRODUCTION

Latency in server systems can be extremely detrimental on an end user's experience. In banking systems, for example, even a few milliseconds of delay can be catastrophic. However, where the bottleneck was once lack of resources due to underpowered hardware and prohibitive cost, this has changed in recent years. The bottleneck is now maximizing allocation of server resources while minimizing energy usage. This is, of course, a hard problem with no clear solution. In their paper, "Optimal Service Elasticity in Large-Scale Distributed Systems" Mukherjee et al. present a novel scheme for automatic load-balancing between individual servers in a large scale system. This scheme utilizes constant overhead for control messages per task in the system, and requires no global queue information, a common requirement in many noteworthy schema of the same type. They demonstrate that the scheme provides provably near-optimal task waiting time and energy wastage measures for large server farms. They further prove that the scheme provides optimal results in the limit as the number of servers tends to infinity, and that these results extend from traditional arrival distributions to phase-type distributions.

II. THE PROBLEM

As the complexity and overall size of distributed computing systems increases, the power loss and latency of the system become much more of a problem than the service resources available. This is the premise of service elasticity as defined in the paper. The problem is then as follows:

> **How do we dynamically scale the available server resources with the observed load conditions, while maintaining minimal energy loss from idle servers and satisfying several performance criteria?**

This is a difficult problem to solve, as turning servers on to meet a changing arrival process takes some amount of time, and thus the server resources available will always lag behind the needs of the observed system load, usually by an amount far exceeding acceptable system latency limits. However this is only true if the reactive balancing method used is naive. Generally, predictive models are preferred, and will yield better results, with several serious caveats. Typically, these schema require a centralized queue, which is not generally true in distributed computing systems. In addition, maintaining such a queue would entail significant, often prohibitive, computational overhead. So how do we efficiently balance incoming load on a distributed system in a strictly reactive manner, while maintaining First-Come-First-Serve policy at each of the servers?

III. TABS ALGORITHM EXPLANATION

Mukherjee et al. propose the following scheme, called "Token-based Auto Balance Scaling", or TABS. The most basic part of TABS is that given a distributed system, each server has a colored token, which it uses to tell a job dispatcher about its current state. The token can be one of 4 colors:

Red - Denoting that the server is in the "idle-off" state.

Green - Denoting that the server is in the "idle-on" state and is ready to accept jobs.

Yellow - Denoting that the server is "busy" (i.e. it has at least one job allocated to it, including the one it is working on.)

Orange - Denoting that the server is currently in "setup" mode, and cannot accept any jobs.

The servers in the system turn on and off according to several rules. When a server becomes idle, it sends a "green" (idle-on) message to the dispatcher, and enters the idle-on period, where it will stay for a period of time t_{idle} which follows the distribution $t_{idle} \sim Exp(\mu_{idle})$. If, after t_{idle} time units have passed and the server has received no new jobs, then it will shut itself off, and send a "red" (idle-off) message to the dispatcher. When a job arrives in the system, then the dispatcher will immediately look for a green server to send the job to (recall that the main goal of TABS is to minimize waiting time. A green server will have zero jobs queued, and thus any new job will experience zero waiting time). At this point the server in question will send a "yellow" message to the dispatcher, and will stay in the busy state until it has finished all of the jobs in its queue. Note that yellow servers *can* accept new jobs from the dispatcher.

So from the perspective of the dispatcher, when a job arrives, the dispatcher will choose a random green server and forward the job to that server. If there are not any green servers,

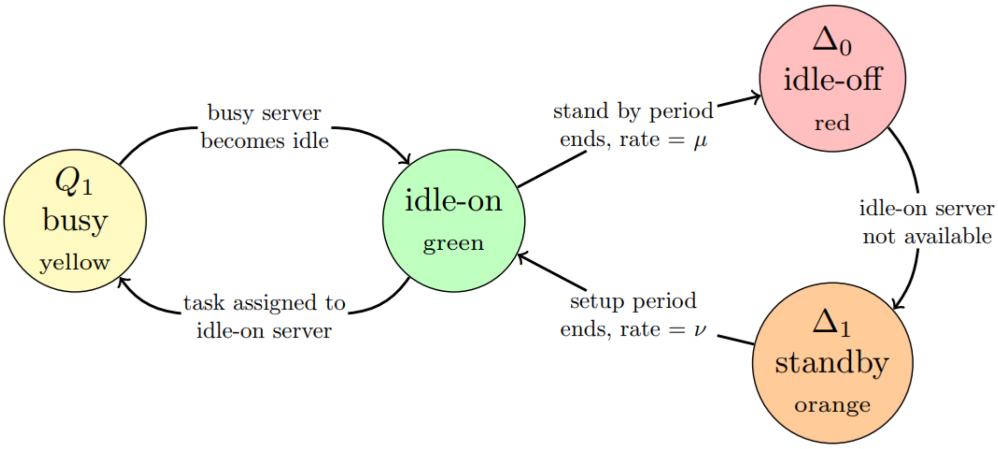


Figure 1: Server Decision flow as implemented in TABS.

then the dispatcher will choose a random yellow server and forward the job there. Then, if there are any red servers, then the dispatcher will send a message to a random red server and tell it to turn on, at which point it will enter a setup period t_{setup} which follows the distribution $t_{setup} \sim Exp(\nu)$. It is worth noting that red and orange servers will never be sent jobs, but green and yellow servers can be sent jobs (with affinity for the former), this is done to minimize job waiting time as much as possible. It is also worth noting that the setup procedures are never aborted, even if green servers become available.

With the algorithm defined, we will now prove the asymptotic optimality of TABS; whereby both the mean waiting time and mean energy wastage ($\mathbb{E}[\mathrm{W}^N]$ and $\mathbb{E}[\mathrm{Z}^N]$, respectively) vanish as $N \to \infty$.

IV. PROOFS

A. Definitions

Given: a distributed computing system with N total servers, each with exponential service process;

$$\{S(t)\}_{t\geq 0} \sim Exp(\mu_{service}),$$

and the Poisson arrival process;

$$\{A(t)\}_{t\geq 0} \sim Exp(N\lambda),$$

and buffer size $B = 1$, such that the overall system can be defined as an M/M/N/2N system, as seen in Figure 2;

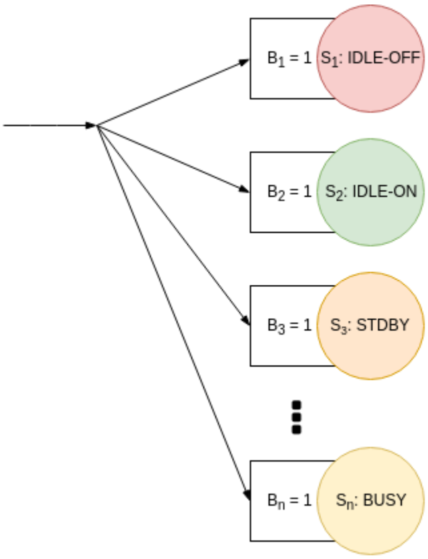


Figure 2: An M/M/N/2N Server system with TABS.

Define:

$$q_i^{(N)}(t) = \frac{Q_i^{(N)}(t)}{N} \tag{1}$$

$$\delta_0^{(N)}(t) = \frac{\Delta_0^{(N)}(t)}{N} \tag{2}$$

$$\delta_1^{(N)}(t) = \frac{\Delta_1^{(N)}(t)}{N} \tag{3}$$

Where Eqn. 1 is the fraction of N servers that have at least i jobs queued at time t, Eqn. 2 is the fraction of N servers that are in idle-off mode at time t, and Eqn. 3 is the fraction of N servers that are in setup mode at time t. Now we can say that the process $\left\{\mathbf{Q}^N(t), \Delta_0^N(t), \Delta_1^N(t)\right\}_{t\geq 0}$ fully describes the evolution of the system and is Markovian. However, the analysis of such a process is extremely difficult due to the coupling between all the service processes, and thus we resort instead to asymptotic analysis and consider the situation when $N \to \infty$, and use the above quantities to define the following vector quantities;

$$\mathbf{q}^N(t) = \left(q_1^{(N)}(t), \ldots, q_B^{(N)}(t)\right) \tag{4}$$

$$\boldsymbol{\delta}^N(t) = \left(\delta_0^{(N)}(t), \delta_1^{(N)}(t)\right) \tag{5}$$

Now, using Eqn. 4 and Eqn. 5, we define $\mathbf{E}$;

$$\mathbf{E} = \left\{(\mathbf{q}, \boldsymbol{\delta}) \in [0,1]^{B+2} : q_i \geq q_{i+1}, \forall i, \delta_0 + \delta_1 + \sum_{i=1}^{B} q_i \leq 1\right\} \tag{6}$$

Eqn. 6 defines the space of all possible system states with respect to time, such that $(\mathbf{q}^N(t), \boldsymbol{\delta}^N(t)) \in \mathbf{E}, \forall t$.

B. Fluid Limit for Exponential Service Distributions

Here we will show that, in the case where $\lambda(t) \triangleq \lambda, \forall t$, the system state space $\mathbf{E}$ has a fixed point, and lead into the optimality proof.

Assume that:

$$\left(\mathbf{q}^N(0), \boldsymbol{\delta}^N(0)\right) \xrightarrow{d} (\mathbf{q}^\infty, \boldsymbol{\delta}^\infty) \in \mathbf{E} \tag{7}$$

as $N \to \infty$. Then,

$$\left\{\mathbf{q}^N(t), \boldsymbol{\delta}^N(t)\right\}_{t\geq 0} \xrightarrow{d} \{\mathbf{q}(t), \boldsymbol{\delta}(t)\}_{t\geq 0} \in \mathbf{E} \tag{8}$$

Note that the process $\{\mathbf{q}(t), \boldsymbol{\delta}(t)\}_{t\geq 0}$ is deterministic, and fully describes the system in the limit ($N \to \infty$). In §3.1, Mukherjee et al. derive several integral equations which describe the fluid limit of the system as described thus far. Please refer there for the derivation, as it is highly involved. The equations are as follows:

$$q_i(t) = q_i^\infty + \int_0^t \lambda(s) p_{i-1}(\mathbf{q}(s), \boldsymbol{\delta}(s), \lambda(s))ds - \int_0^t (q_i(s) - q_{i+1}(s))ds, i = 1, \ldots, B \tag{9}$$

$$\delta_0(t) = \delta_0^\infty + \mu \int_0^t u(s)ds - \xi(t) \tag{10}$$

$$\delta_1(t) = \delta_1^\infty + \xi(t) - \nu \int_0^t \delta_1(s)ds \tag{11}$$

where:

$$u(t) = 1 - q_1(t) - \delta_0(t) - \delta_1(t) \tag{12}$$

$$\xi(t) = \int_0^t \lambda(s)(1 - p_0(\mathbf{q}(s), \boldsymbol{\delta}(s), \lambda(s0)))\mathbb{1}_{[\delta_0(s)>0]}ds \tag{13}$$

and for any $(\mathbf{q}, \boldsymbol{\delta}) \in \mathbf{E}, \lambda > 0$:

$$p_0(\mathbf{q}, \boldsymbol{\delta}, \lambda) = \begin{cases} 1, & \text{if } u = 1 - q_1 - \delta_0 - \delta_1 > 0, \\ \min\left\{\lambda^{-1}(\delta_1\nu + q_1 - q_2), 1\right\}, & \text{otherwise,} \end{cases} \tag{14}$$

$$p_i(\mathbf{q}, \boldsymbol{\delta}, \lambda) = (1 - p_0(\mathbf{q}, \boldsymbol{\delta}, \lambda))\,(q_i - q_{i+1})\,q_1^{-1}, i = 1, \ldots, B \tag{15}$$

Where $u(t)$ is the asymptotic fraction of unused resources at time t, $\xi(t)$ is the asymptotic fraction of server setups started in the interval $[0,t]$, $p_i(\mathbf{q}, \boldsymbol{\delta}, \lambda)$ is the instantaneous fraction of incoming jobs assigned to some server with queue length q_i when the system state is $(\mathbf{q}, \boldsymbol{\delta})$.

C. Fixed Point in System State Space

From these equations, we can derive that there is a fixed point within $\mathbf{E}$, in the following location:

$$\delta_0^* = 1 - \lambda, \qquad \delta_1^* = 0, \qquad q_1^* = \lambda, \qquad q_i^* = 0 \tag{16}$$

It is very important to note that these fixed point values are completely independent of the values ν and μ. This implies that for any values of ν and μ, the system will always reach this equilibrium point as $t \to \infty$.

D. Performance Metrics for $\mathbb{E}[Z^N]$

We mentioned earlier that one of the quantities we wanted to optimize for was wasted energy ($\mathbb{E}[Z^N]$), however we have not defined it. Let us first define $\mathbb{E}[P^N]$ to be the power usage of the N^{th} server in the system. Note that the power usage of an idle-on ("green") server is less than one which is starting up ("orange") or a busy one ("yellow"), while an idle-off ("red") server uses 0 Watts. Let us then define P_{full} to be the power used by orange and yellow servers, and to be the power used by green servers. [1] In order for the system to be stable, at least a fraction of the total servers λ must be on in order to handle the incoming tasks. Thus, the lower bound on energy usage per server is λP_{full}. It follows naturally that we should define the wasted energy as the energy used minus the lower bound;

$$\mathbb{E}[Z^N] = \mathbb{E}[P^N] - \lambda P_{full} \tag{17}$$

E. Proof of TABS Asymptotic Optimality - Waiting Time

Proof. When $\lambda < 1$, for any $\mu > 0, \nu > 0$ as $N \to \infty$:

$$\begin{aligned}
&\mathbb{E}[W^N] = \frac{\mathbb{E}[L^N]}{N\lambda}, \text{ where } \mathbb{E}[L^N] \triangleq \sum_{i=2}^{B} Q_i^N \\
&\mathbb{E}[W^N] = \frac{\sum_{i=2}^{B} Q_i^N}{N\lambda} \\
&\mathbb{E}[W^N] = \frac{\sum_{i=2}^{B} q_i^N}{\lambda} \\
&\mathbb{E}[W^N] = \frac{\sum_{i=2}^{B} q_i^*}{N\lambda}, \text{ where } q_i^* = 0, \forall i \geq 2 \text{ as } N \to \infty \\
&\mathbb{E}[W^N] \to 0 \text{ as } N \to \infty
\end{aligned}$$

□

F. Proof of TABS Asymptotic Optimality - Wasted Energy

First, let U^N equal the number of idle-on servers:

$$U^N \triangleq N - Q_1^N - \Delta_0^N - \Delta_1^N$$

[1] Note that while yellow servers can be considered to be doing useful work (i.e. processing jobs), green servers can be seen as the primary source of wasted energy. This will be important during the proof.

Proof. When $\lambda < 1$, for any $\mu > 0, \nu > 0$ as $N \to \infty$:

$$
\begin{aligned}
&\mathbb{E}[P^N] = \frac{1}{N}\mathbb{E}\left[\left(Q_1^N + \Delta_1^N\right) P_{full} + U^N P_{idle}\right] \\
&\mathbb{E}[P^N] = \mathbb{E}\left[\left(q_1^N + \delta_1^N\right) P_{full} + u^N P_{idle}\right] \\
&\mathbb{E}[P^N] \to \left(q_1^* + \delta_1^*\right) P_{full} + u^* P_{idle} \text{ where } q_1^* = \lambda, \delta_1^* = 0, u^* = 0 \text{ as } N \to \infty \\
&\mathbb{E}[P^N] \to (\lambda + 0)\, P_{full} + 0P_{idle} \text{ as } N \to \infty \\
&\mathbb{E}[P^N] \to \lambda P_{full} \text{ as } N \to \infty \\
&\mathbb{E}[Z^N] \triangleq \mathbb{E}[P^N] - \lambda P_{full} \\
&\mathbb{E}[Z^N] \to \lambda P_{full} - \lambda P_{full} \text{ as } N \to \infty \\
&\mathbb{E}[Z^N] \to 0 \text{ as } N \to \infty
\end{aligned}
$$

□

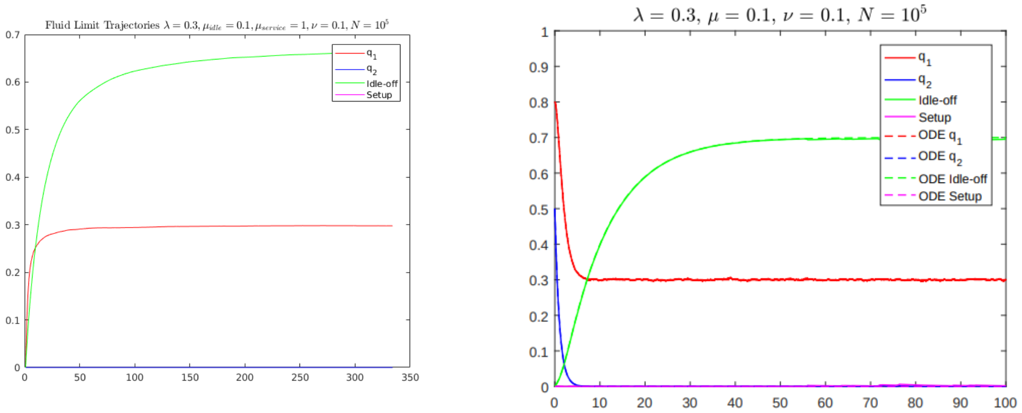
Here we have proven the asymptotic optimality of TABS, both in terms of waiting time, and wasted energy. These results can also be proven for the case of phase-type arrival distributions where $\lambda(t) \neq \lambda \forall t$, which Mukherjee et. al prove as well.

V. SIMULATION RESULTS

Mukherjee et al. corroborate their results with the following simulations:

A. Convergence of Sample Paths to Fluid-Limit Trajectories

The trajectories of the fluid limit quantities are analyzed under several arrival distributions (i.e. Constant arrival rate, Sinusoidal arrival rate, and Hyperexponential arrival rate). The results for the time-varied arrival scenario was most interesting, as it considered how TABS might perform in a real-world computing system, where load will very rarely be constant, and can generally be modeled by a sinusoidal process. Most notably, the resultant trajectories in the sinusoidal case show that the system's reactive ability is immense, as the modeled arrival rate varies extremely quickly with respect to the mean service time, yet the TABS scheme is able to balance the load extremely effectively.



(a) My Results

(b) Mukherjee et al.'s Results

Figure 3: Fluid Limit Trajectories for constant arrival rate

In both the constant arrival and hyperexponential scenarios, the simulations illustrate that the TABS scheme will always reach the fluid limit, regardless of the distribution of arrivals. I replicated the case where $\lambda(t) \triangleq \lambda = 0.3, \forall t$. My simulations (shown in figure 3) the same asymptotic limits as Mukherjee et al. yet the initial evolution of the system differs. This is due to the fact that my simulations assumed that the entire system began as idle-on. I would say this is a meaningful comparison, because it shows through simulation that the TABS scheme is entirely insensitive of initial conditions, and proves that there is a system-wide fixed point.

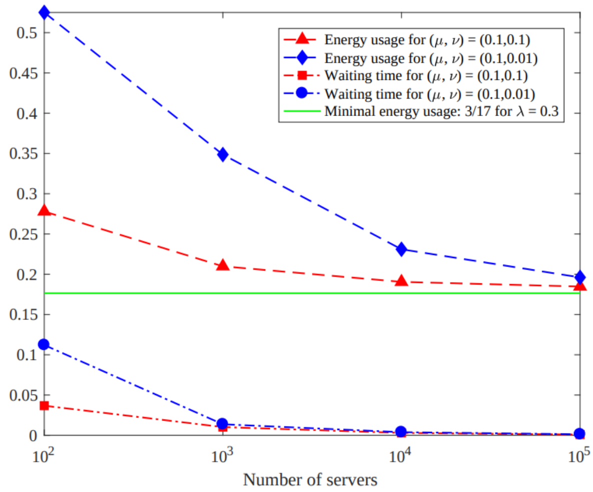


Figure 4: Convergence of $\mathbb{E}[W^N]$ and $\mathbb{E}[Z^N]$ under TABS for various values of N

B. Convergence of Steady-State Performance Metrics to Fluid-Limit Values

Mukherjee et al. also show that the performance metrics (i.e. Waiting Time and Wasted Energy) vanish in the limit as $N \to \infty$. They show notably that this convergence occurs for different values of μ and ν, however the rate of convergence for $\mathbb{E}[Z^N]$ is severely effected for small values of ν, as seen in Figure 4.

VI. FURTHER WORK

While this is a phenomenal and comprehensive work, there are several extensions that I could see being explored. I would be very interested in combining this scheme with a Power-of-d choices type algorithm for finding the best green or yellow server to use. I believe that there could be significant improvement in latency, but the additional overhead of calculating these choices negates the benefit of not having to retain global queue information, however since we only need to keep queue information for d yellow servers at each time, and eventually the yellow servers will vanish in the limit, then application of power-of-d should help this scheme to converge faster. However, this is mere speculation and I have no simulations yet to corroborate my hypothesis. If I am able to find the time, I will definitely be exploring this extension.