Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SIMULATION STUDY OF DYNAMIC LOAD BALANCING FOR PROCESSOR SHARING SERVERS WITH FINITE CAPACITY UNDER GENERALIZED HALFIN-WHITT REGIMES

Matías Bonaventura
Rodrigo Castro

Departamento de Computación
FCEyN, UBA and ICC-CONICET
Ciudad Universitaria, Pabellón 1
Buenos Aires, C1428EGA, ARGENTINA

Matthieu Jonckheere

IC-CONICET
Ciudad Universitaria, Pabellón 2
Buenos Aires, C1428EGA, ARGENTINA

ABSTRACT

Defining efficient decentralized load balancing schemes exhibiting low memory and communication costs is an important ongoing topic. In particular, characterizing critical regimes where a system optimizes resource usage is uncharted territory. We consider here dynamic balancing schemes in a set of processor sharing servers with finite capacity. Guided by recent results for insensitive load balancing schemes, we applied a modeling and simulation strategy to characterize systematically and extensively several classes of balancing policies under various statistical conditions. We found that there is a class of efficient policies for which a common critical regime can be identified and interpreted as a generalization of the Halfin-Whitt-Jagerman regime for one-server systems. We also study the gap between full and partial information systems, and analyze the performance sensitivity to jobs' size distribution. This study is motivated by the network architecture in the ATLAS experiment at CERN, where load balancing plays a key role.

1 INTRODUCTION

Load balancing is an essential, often crucial mechanism to improve performance in call centers, server farms, and various applications that operate on parallel servers.

In the last decades new challenges have emerged given the enormous growth in the number computer nodes involved in an increasing number of applications. On top of defining efficient balancing schemes (in terms of delay and blocking probability), applications with a large number of servers typically require cheap schemes in terms of exchanged messages, required memory and computing efforts at the dispatcher. However, for systems with load balancing, the precise load regimes (ratio between the amount of incoming load and the service capacity) that can lead to very efficient utilization of resources remain largely uncharacterized.

As a consequence, there are no simple rules of thumb for dimensioning systems with *dynamic* load balancing schemes, i.e. when the decisions at the dispatcher depend on the instantaneous load (e.g. the number of active jobs in the system) with various possible types of load information and costs criteria. This fact becomes even more evident for large systems with asymmetric server speeds and blocking probability, where both the precise structure of optimal policies and their performance elude current theoretical knowledge and practical techniques.

Two main ideas are usually adopted to gain theoretical insights: to assume large scale networks to obtain asymptotic results using propagation of chaos, on the one hand, and to restrict the load balancing schemes to obtain computable bounds, on the other.

Since the 80s, strong attention has been given to mean-field results for different classes of networks with load balancing schemes. In particular, a great deal of research has been devoted to prove mean-field limits for schemes like *join the shortest of d among n queues* (also called *power of d*) where n is large, starting with the seminal work of Mitzenmacher (2001) and Vvedenskaya et al. (1996), and quickly complemented by

several papers on the mean-field behavior of such systems. Transient functional law of large numbers and propagation of chaos have been obtained for instance in Graham (2000) and Mukhopadhyay et al. (2015) for First In First Out (FIFO) scheduling, and more recently, propagation of chaos properties and asymptotic behavior of the number of occupied servers were obtained under very general assumptions (Bramson M. 2012). All these results concern systems without blocking and sensitive to the job size distribution.

Meanwhile, researchers have considered schemes that lead to the insensitivity of the queuing system to the jobs' size distribution. This route was first taken in Bonald et al. (2004), considering dynamic load balancing schemes that are insensitive to the job size distribution (see also Leino and Virtamo 2006, Pla et al. 2008 or Jonckheere and Mairesse 2010)

For small networks with a single class of traffic, it was shown that the insensitive load balancing (ILB) compares very accurately to optimal policies for a given job size distribution to estimate blocking probabilities, while delay estimations are a bit less accurate (Bonald and Proutière 2003; Bonald et al. 2004). The penalization imposed by reversibility is greater for multi-class networks while the sensitivity (of optimal sensitive policies) also deteriorates (Leino and Virtamo 2006; Jonckheere and Mairesse 2010).

Hence, a small to moderate price must be paid for robustness and simplicity. It is perhaps counter-intuitive that for models with infinite buffers, this price becomes very high. It was indeed proved that if the state space is infinite, and assuming absence of blocking, the optimal insensitive load balancing is static (i.e., it does not depend on the queue-length) and it is hence much less efficient than a state-dependent sensitive load balancing (Jonckheere 2006).

Both research directions described above shed light on the possible performance of dynamic load balancing, but also present strong limitations:

- The performance of the limiting system might not be informative. For instance the blocking probability or the delay will be 0 for a large class of policies,
- The price to impose insensitivity to the job size distribution is largely unknown.

In Jonckheere and Prabhu (2016), the intersection of both research directions was considered by studying the asymptotics of large networks (i.e., a finite but arbitrarily large number of servers) for ILB schemes. It was shown that a qualitative phase transition occurs at a critical load $\rho_c(n) = 1 - an^{-\theta/(\theta+1)}$ where θ is the buffer depth and a is a constant value. The blocking probability is exponentially small until $\rho_c(n)$, it then changes to order $O(\varepsilon n^{-1/2})$ at the critical load, and to a higher order thereafter. This generalizes the Halfin-Whitt-Jagerman (H-W-J) regime established for $M/M/n/n$ systems. In Halfin and Whitt (1981), Jagerman (1974), critical regimes for optimal use of resources were identified for single queues. In particular the blocking probability for $M/M/n/n$ systems is $O(n^{-1/2})$ only if the number of servers scales as $\rho + a\sqrt{\rho}$, where ρ is the load of the system. Before this critical regime the blocking probability is exponentially small, while it is of constant order after that critical point.

In Jonckheere and Prabhu (2016), it is hence shown that a rule similar to the popular *staffing rule* established for the $M/M/n/n$ system is valid, but must depend critically on the value of θ when dynamic load balancing is employed.

Whether this critical regime is a general phenomenon or not (i.e., is it verified for a large class of dynamic load balancing schemes ?) remains an important and completely open question that we investigate numerically in this work.

Our main contribution is to show, by means of extensive systematic simulations, that there exists a class of 'efficient' policies (i.e. optimal for some job size distribution, including for instance join the shortest queue) which share the same critical regime as the ILB policy, while more decentralized and less efficient policies (like power of d) might have different critical regimes. We also investigate the sensitivity of non-reversible policies that are sensitive to the job size distribution.

Finally, our last contribution consists of providing performance benchmarks for balancing policies with various types of information (e.g., the number of jobs in each server, the number of idle servers, a local information about the congestion of each server, etc.) for general service distributions.

This paper is organized as follows. In Section 2 we present a real case study at CERN that motivates our study of load-balancing strategies. In Section 3 we introduce the model under study, detail the different strategies, and mention relevant modeling and simulation aspects related to DEVS (our simulation framework of choice). Section 4 shows the simulation results and the critical load at which some strategies exhibit a phase transition in their blocking probability. In Section 5 we present conclusions and possible lines for future research.

2 MOTIVATING CASE STUDY

The ATLAS experiment at CERN (Collaboration 2008) hosts one of the four detectors at the Large Hadron Collider (LHC) where bunches of particles collide every 25 ns. The ATLAS detector measures and digitizes physical properties of the traveling particles (in units of information called Events) at a raw workload of about 80 TB/s. In order to assimilate such a huge throughput ATLAS relies on a complex layered system known as TDAQ (trigger and data acquisition) (Pozo Astigarraga et al. 2015) that combines custom electronics networked with farms of servers. TDAQ decides in real time whether each Event should be permanently stored for off-line analysis or safely discarded, achieving a rejection factor of about 40000x. A first-level trigger (L1) system uses electronics to filter events down to roughly 100 kHz. L1-accepted Events are temporarily stored in the Read-Out System (ROS), composed of approximately 100 server nodes.

A crucial element is the High Level Trigger Supervisor (HLTSV) node, that orchestrates and balances the assignment of each Event stored at the ROS nodes into one of approximately 2000 Trigger Processing Unit (TPU) servers, that compose the next filtering layer. This is a highly sensitive task from the load balancing perspective, since the HLTSV must assign Events complying with several constraints: it should be fast enough to avoid buffer overflows at the ROS layer, and it should distribute Events fairly among TPUs to avoid overloading resources.

A strong requirement is to avoid any piece of information to get discarded without being analyzed (i.e. interesting physics Events should not be discarded due to uncontrolled system conditions, such as overflown nodes in the network or farm nodes). The estimated averaged processing delay for an Event is around 300 ms (end to end, since it is first stored at the ROS farm layer until it gets fully analyzed by the TPU farm layer).

An inefficient load balancing strategy can saturate TPUs unnecessarily making delays grow, thus increasing the blocking probability and processing times.

Requirements, technologies and budget change on a continuous basis in the engineering process of the TDAQ system. Thus, varied design options should be considered for load balancing strategies. A key practical design question is what kind of balancing algorithm can offer the best combination of delay and blocking features for a given number of nodes or, conversely, what is the minimum number of nodes required to guarantee delay and blocking for a given strategy.

In the Discrete Event Simulation lab we apply modeling and simulation (M&S) research for the TDAQ system, assisting the TDAQ network and data-flow teams at CERN. We rely on the Discrete Event System Specification (DEVS) formal M&S framework (Zeigler et al. 2018) to develop models and tools through different life-cycles: Build and maintenance (of simulation models), Hypothesis and design (on the system) and Explore and discover (on simulation results) (Bonaventura et al. 2016). We also develop an ecosystem of related tools and methods, such as automated data analysis for continuous simulation validation (Foguelman et al. 2016) or automated topology generation for software defined networks simulation (Laurito et al. 2017).

In this context, an important goal is to verify, through simulation studies, different load balancing strategies guided by theoretical insights. In this work we implement all simulations with the DEVS M&S framework making the models of the investigated schedulers readily pluggable into our TDAQ system model.

3 MODEL AND DIFFERENT LOAD BALANCING POLICIES

We consider a set of n processor sharing servers, with speed 1 and buffer size θ, receiving jobs according to a Poisson process of rate $\lambda = \rho.n$. The job size distribution is assumed generic with a finite first moment. We denote by $x = (x_1, \ldots, x_n)$ the number of jobs at each of the n servers. A dispatcher routes each incoming job to one of the servers according to a given load balancing strategy. We denote by $\lambda_i(x)$ the arrival rate at the i-th server describing the load balancing strategy. When the dispatcher sends an incoming job to a server with θ active concurrent jobs, the new request gets blocked, i.e. it is rejected from the system. Hence, the state space of the number of jobs in the system is finite and the system is always stable. We denote by B_n^θ the blocking probability of a system with n servers of capacity θ each (i.e. a maximum of θ jobs can be served simultaneously). We call *delay* to the service time spent by an arbitrary job within the system.

We now characterize the set of load balancing policies that shall be studied, which can be grouped into *centralized* and *decentralized* strategies. Centralized load balancing refers to policies having full information (about the number of jobs in each server) available at the dispatcher. Decentralized load balancing, on the contrary, refers to policies where the dispatcher manages only partial (or local) information. Usually centralized policies yield better service time and blocking probabilities while decentralized schemes minimize the utilization of communication channels between the processors and the dispatcher (which have limited capacity in real world systems). We will study the following schemes:

- **Join the shortest queue (JSQ)** This centralized strategy dispatches to one (of possibly many) shortest queue, breaking ties at random. It is optimal for a wide class of job size distributions but is also known not to be optimal for size distributions with high variance (see (Righter and Shanthikumar 1989) and references therein).
- **Insensitive load balancing (ILB)** This centralized strategy has been extensively studied in (Bonald et al. 2004; Jonckheere and Mairesse 2010) and more recently in (Jonckheere and Prabhu 2016). It has the desirable property to be insensitive to the job size distribution, i.e., the stationary measure of the number of concurrent jobs in each server depends only on the first moment of the job size distribution. The incoming job is routed to server i with the following probability:

$$p_i^{ILB}(x) = \frac{\theta - x_i}{\sum_{j=1}^n (\theta - x_j)}.$$

  This load balancing rule was proved to be optimal (in the sense that it minimizes the blocking probability for any convex criterion) in the set of insensitive load balancing for a single class of traffic in (Bonald et al. 2004).
- **Join the idle queue (JIQ)** This partially centralized strategy uses only as state information whether each server is idle or not (and dispatches at random otherwise). It is hence a first step towards decentralization while its efficiency is potentially much better than fully decentralized schemes.

- **Random (RND)** This completely decentralized strategy uses no information from the system and chooses at random a single server for each new incoming request.
- **Power of d (Pod)** This partially decentralized strategy corresponds to choosing at random a subset of d servers among n and then send to the shortest queue within this subset.
- **First-Finished-First-Assigned (FFFA)** This centralized strategy is the one currently implemented in the TDAQ farm. It imposes a small computing effort on the dispatcher. It assigns new jobs to servers in the same order in which they finish processing jobs. The bootstrap assignment starts with $n\theta$ random (unique) assignments.
- **Centralized Random (CRND)**. This strategy uses only as state information whether each server is fully busy or not (and dispatches at random otherwise).

Remark 1 (The particular case $\theta = 1$) When $\theta = 1$ all centralized policies coincide, and the system corresponds to the $M/M/n/n$ queue. Measures such as the mean delay and the blocking probability can be explicitly calculated and the critical regime corresponds to the well-known Halfin-Whitt-Jagerman regime.

3.1 Model Implementation in DEVS

We modeled the load balancing system described above in PowerDEVS (Bergero and Kofman 2011), a discrete event simulator that implements the DEVS mathematical formalism. PowerDEVS offers several libraries of predefined models, consisting of reusable blocks that can be interconnected to build more complex systems. We expanded the libraries to incorporate new load-balancing models making them available for other applications. In particular, the new models can be readily plugged into the TDAQ network model described in Section 2 to test different policies in varied TDAQ scenarios.

Figure 1 (bottom) shows the high-level view of the load balancing model as implemented in PowerDEVS. The *JobGenerator* model follows parameterizable probabilistic distributions to generate new jobs with desired sizes and send rates λ. New jobs are sent to the *Dispatcher* model that behaves according to the DEVS Graphs diagram (Christen et al. 2004) depicted in Figure 1 (top right). Upon receiving a new job, an external state transition is triggered by the *NewJob* arriving at the input port *In0*. This transition brings the model from the *Wait* state to the *SendAssignment* state. The Dispatcher will remain at the *Wait* state forever (its autonomous time advance is infinite, depicted as *ta=INF*) unless an external message arrives. Conversely, the lifetime of the *SendAssignment* state is zero (depicted with *ta=0*), meaning this is an instantaneous state, which will undergo an internal transition immediately. The *Dispatcher* relies on a load balancing *strategy* to decide which processor will handle each new job. This decision is made during the external transition (solid arrow), while the message with the decided assignment is sent out during the instantaneous internal transition (dotted arrow, back to *Wait*) through the output port *Out0*.

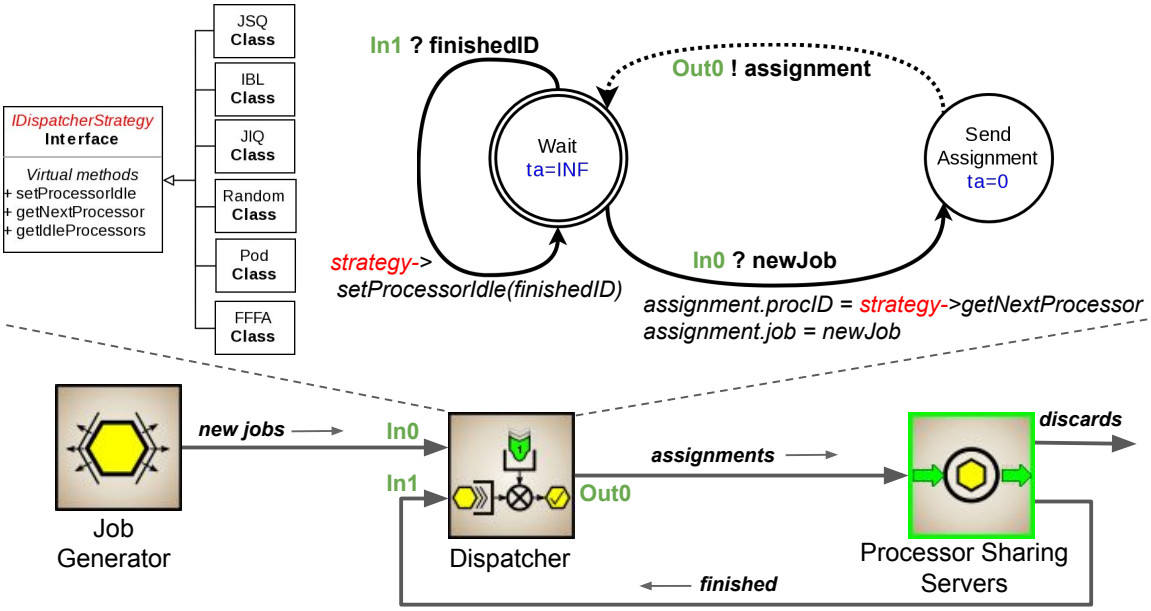


Figure 1: Model implementation in PowerDEVS (bottom) and Dispatcher model state machine (top).

The *Dispatcher* also receives events through the port *In1* whenever a server finishes processing a job, in which case it notifies the ID of the processor to the *strategy*. All strategies described in the previous section are implemented following a single class hierarchy (as shown in Figure 1, top left) and they all implement a common *IDispatcherStrategy* interface, which decouples the *Dispatcher* and *strategy* logics. This modeling

approach facilitates the addition of new strategies, which should only implement the common interface, without changing the *Dispatcher* logic nor the rest of the system.

The *Processor Sharing Servers* model uses Vectorial DEVS (Bergero and Kofman 2014) to create automatically N instances of a same DEVS model (depicted with a green border), possibly with different values for its parameters. Each server model, identified by its index i, has a parameterizable finite capacity θ_i and processing power c_i (the latter fixed all to 1 in this work).When servers receive assigned jobs from the *Dispatcher* the current number of jobs being processed x_i is checked: if it exceeds θ_i the job is discarded, otherwise the job is accepted and processed.

4 NUMERICAL SIMULATION AND RESULTS

We start the study with simulations for the ILB policy in the vicinities of the critical regime (described in Jonckheere and Prabhu 2016) for which the following result was proved, giving an asymptotic closed form expression to compute the blocking probability for the limiting system:

Theorem 1 For $a \in (-\infty, \infty)$, let

$$n\rho = n + an^{\frac{1}{\theta+1}}. \tag{1}$$

Then,

$$\lim_{n\to\infty} B_n^\theta n^{\frac{\theta}{\theta+1}} = \left[\int_0^\infty \exp\left(au - \frac{u^{(\theta+1)}}{(\theta+1)!}\right) du\right]^{-1}. \tag{2}$$

In the sequel, we consider the following as the *normalized blocking probability*

$$\bar{B}_n^\theta = B_n^\theta n^{\frac{\theta}{\theta+1}}.$$

Our first simulations check how the blocking probability in the pre-limit (i.e., for fixed large n) compares to the theoretical limit defined above for the ILB policy. Figure 2 shows the ILB blocking probabilities for systems with different number of servers, with the case $n = \infty$ representing the theoretical limiting formula (2). Note that the x-axis depends on the a parameter which is a handy link to the load $n\rho$ according to formula (1) that facilitates looking at limiting values when $n \to \infty$.

Figure 2(a) shows results for high loads, where the blocking probability of the simulated systems get closer to the theoretical system as n increases. Note that the expected asymptotic blocking probability for large ρ ($B_n^\theta = 1$) is not predicted by the formula (which shows that the limits in n and ρ do not commute here).

For light loads, the formula predicts quite well the blocking probabilities for all systems (independently of the number of servers).

Figure 2(b) is a close-up view around the critical load $a = 0$ with a log scale in the ILB blocking probability. The figure shows that the theoretical formula predicts very well the inflexion point for the ILB blocking probability in the critical regime. The formula precisely predicts ($\bar{B}_n^\theta = 0.5$) for all systems (independently of the number of servers) in the critical load ($a = 0$). For loads in the vicinity of $a = 0$ the theoretical predictions get precise for $n > 100$ and deteriorate for large $|a|$.

In particular, for critical loads lower than $a = -5$ the theoretical formula gets a significant bias and predicts much lower blocking probabilities than those obtained with simulations. It is interesting to observe that all simulated systems start their phase transition (passing from an exponentially small blocking to a polynomially small blocking) at approximately the same normalized load (at $a \simeq -5$) and with approximately the same normalized blocking probability ($\bar{B}_n^\theta \simeq 10^{-6}$). As expected, this is not predicted by the asymptotic formula (2) (which does not depend on n).

4.1 Performance Comparisons

A performance comparison for different policies is presented in Figures 2c and 2d for blocking probabilities and mean service time, respectively. All strategies present small blocking probabilities when compared to the random strategy, and all start increasing quickly after $\rho > 1$. Regarding the mean service time, CRND shows the worst processing times. JIQ, JSQ and Power-of-20 are the best performing policies and show a close-to-optimal delay with low loads and a rapid increase in the critical regime. The rest of the policies dwell in the middle with a softer increase of the delay for higher loads. Under heavy traffic conditions

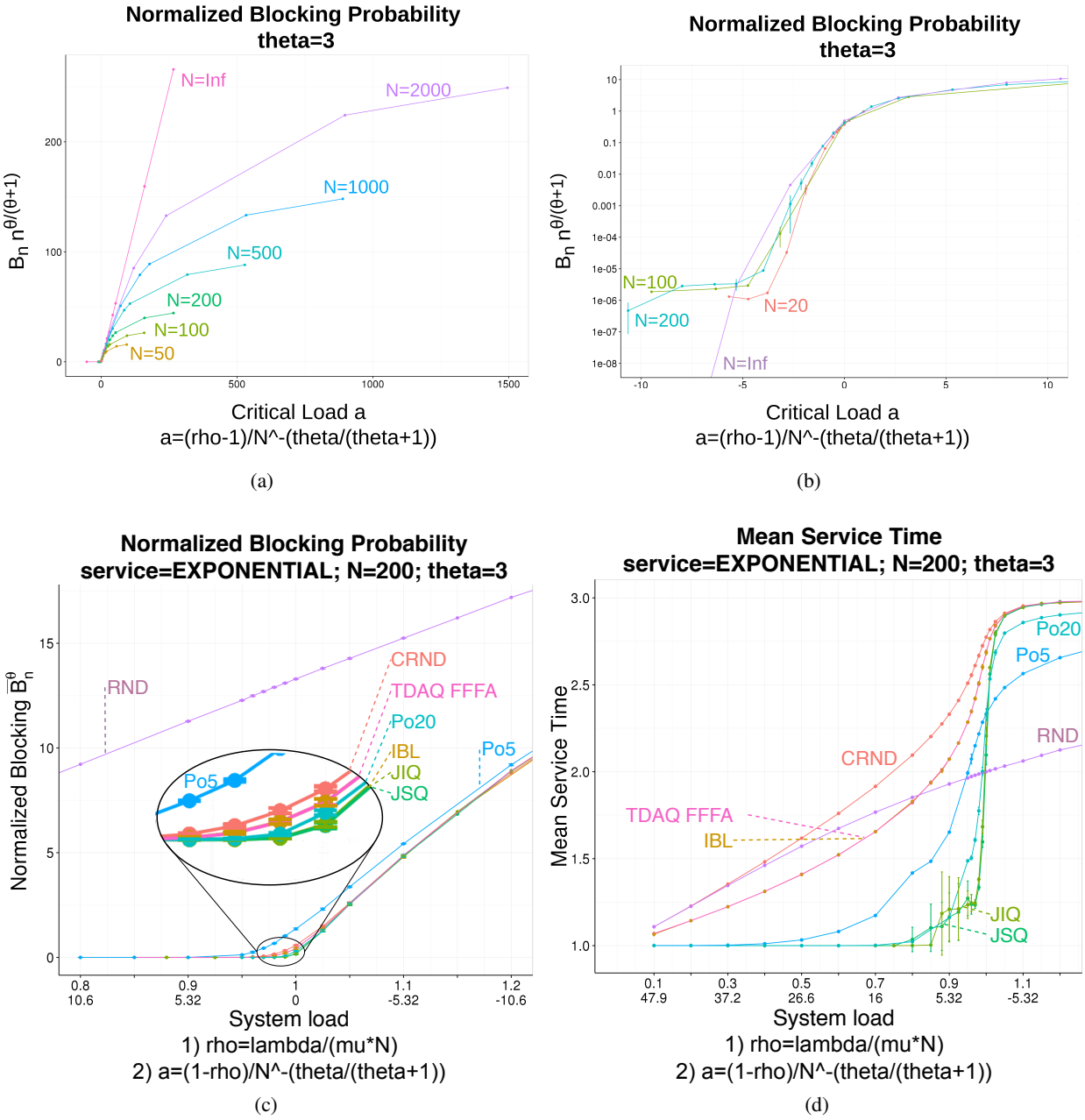


Figure 2: Normalized blocking probability. (a) and (b) for IBL with different number of servers. (c) and (d) for different strategies. Theta=3, Service=exp(1). Bars represent the standard deviation.

($\rho \geq 1$), all policies behave almost similarly with delays close to the maximum given by $\mu\theta$. The Random policy shows an interesting (perhaps surprising) trade-off between a reasonable delay and a (very) high blocking probability.

4.2 Critical Regimes for Efficient Policies

In this section, we provide important conclusions suggesting the possible validity of the generalization of the Halfin-Whitt-Jagerman scaling for a large class of efficient policies. This leads to a convenient rule of thumb given by the formula (2) that implies choosing n servers for an incoming load of $n + an^{\frac{1}{\theta+1}}$.

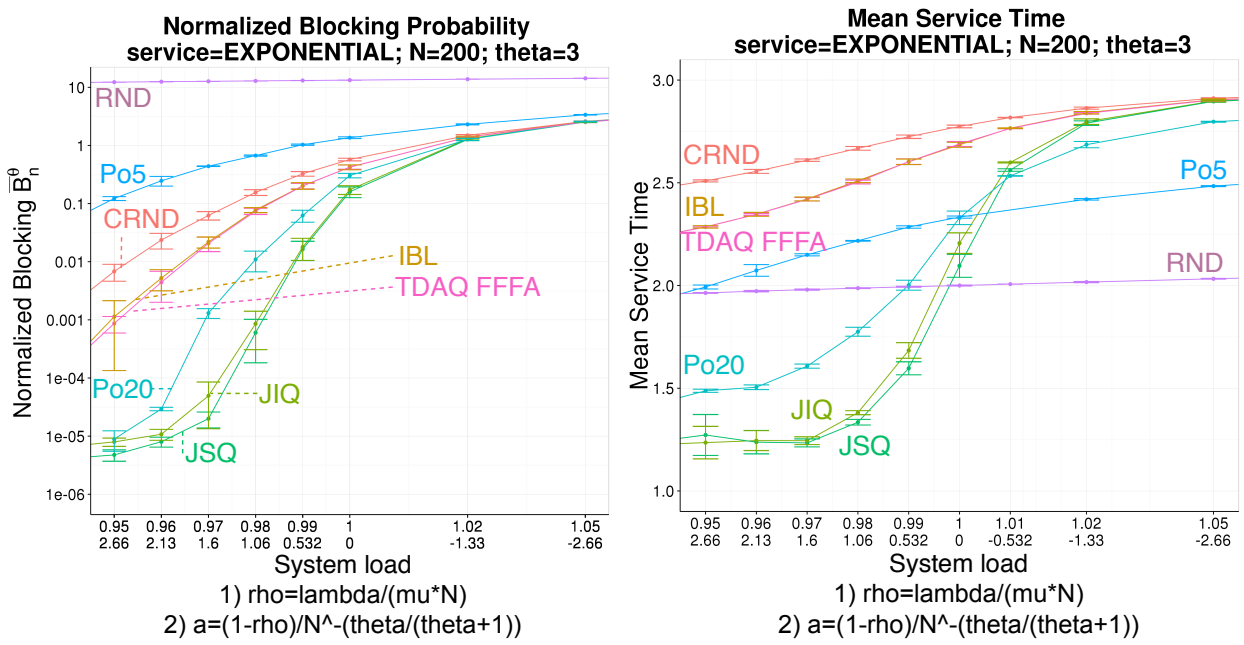
We therefore define the class $\mathscr{C}$ of policies such that

$$\lim_{n\to\infty} B_n^{\theta} n^{\frac{\theta}{\theta+1}} = \kappa(a) \in (0,\infty).$$

For this class, the blocking probability is exactly of order $O(n^{\frac{-\theta}{\theta+1}})$ for large n, with $\kappa(a)$ the corresponding constant depending of the normalized load a. By definition, the ILB policy belongs to $\mathscr{C}$.

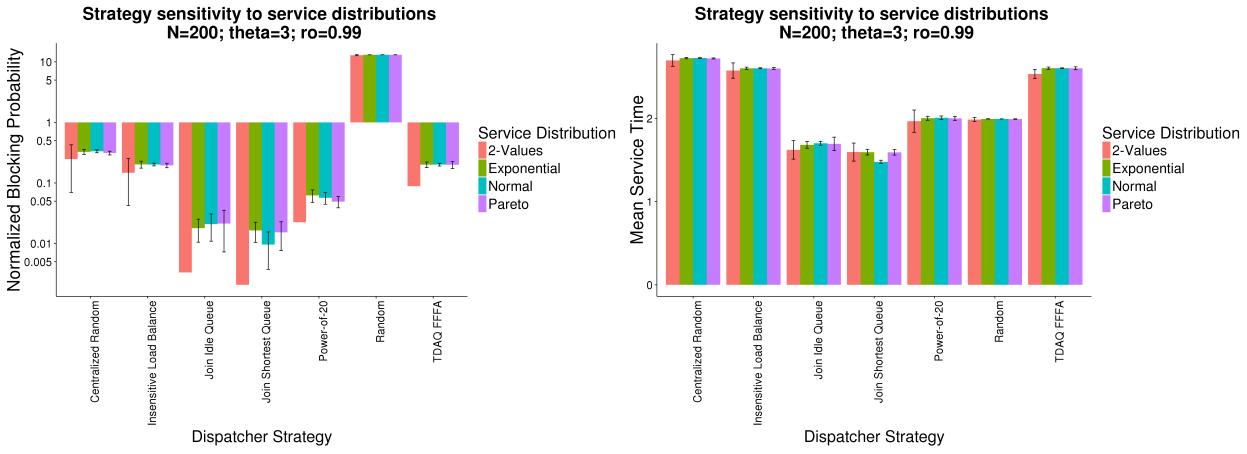
One way to check if a policy belongs to $\mathscr{C}$ is to plot the normalized blocking probability and to verify whether it transitions from very small values to $O(1)$ as a changes. The simulations in Figure 3 show the normalized blocking probability and evidences that there indeed exist consistent classes of policies that do share the same phase transition (i.e. same critical load depending on n and θ). Also, various policies with less efficiency do not share the same characteristics. For those efficient policies, the dimensioning rule of thumb could be applied.

Also, the results in Figure 3a showing the normalized blocking probability for IBL as compared to other strategies, indicate that the theoretical formula (2) could be applied in practical scenarios as an estimate of low/high bounds for the blocking probability. For example, in the TDAQ case study, IBL exhibits blocking probabilities very similar to FFFA, and it is expected not to be far from the theoretical closed formula (easier to calculate) for a large number of servers (n $\sim$ 200).



(a) (b)

Figure 3: Normalized Blocking Probability (a) and Mean Service Time (b) for different strategies in the critical regime (logarithmic scale). N=200, theta=3, service=exp(1).



(a) (b)

Figure 4: Sensitivity of load balancing strategies to service time distribution.

Finally, in view of the simulation results we can propose the following conjectures:

Conjecture 1 The JSQ policy, the JIQ policy and the FFFA policy belong to $\mathscr{C}$.

Conjecture 2 The Power of d policy does not belong to $\mathscr{C}$ for any fixed d (i.e., not depending on n).

Conjecture 3 There exists $d(n) \leq \sqrt{n}$ such that the Power of $d(n)$ policy belongs to $\mathscr{C}$.

4.3 Sensitivity

Figure 4 compares the performance of several policies for different job size distributions (with mean equal to 1 in all cases). A generic conjecture for processor sharing servers (and more generally for any symmetric scheduling) is that all the limits are insensitive to job size distribution as $n \to \infty$. But it remains an open problem to quantify the sensitivity for fixed n. Our simulations showed that this sensitivity is limited in all policies for $N \leq 50$, and almost inexistent for $N \geq 50$. This contribution provides insights on the price of imposing insensitivity to the job size distribution. This implies that for medium to large systems ($N > 50$), dynamic load balancing schemes are robust to the statistical variation in the service distribution, at least for blocking probabilities.

5 CONCLUSIONS AND FUTURE WORK

Using extensive and systematic simulations, we reached the following findings.

1. There exists a non-trivial critical regime corresponding to a generalization of the Halfin-Whitt regime for a large class of centralized load balancing. This is the desirable regime of operation in order to best take advantage of the available resources. This regime has been completely characterized for the ILB strategy but remains an open theoretical problem for other load balancing strategies. Based on our simulation results, we conjectured that the scaling parameters should be essentially the same for a large class of policies including ILB and JSQ.
2. In this critical regime, we systematically observed that the blocking probabilities of efficient centralized policies are very similar. There is a mild gap of performance with partially decentralized policies which becomes very large when comparing against purely random load balancing.
3. We empirically observed a non-trivial trade-off between blocking probability and delays which could make decentralized policies attractive for applications where a certain amount of blocking can be allowed.
4. Finally, we found that the sensitivities of blocking probability and mean service time to the service time distribution vanishes quickly as the number of servers grows. Therefore, it is reasonable to conjecture asymptotic insensitivity in the studied critical regime.

Our simulations provide new evidence indicating that it is possible to design novel decentralized load balancing policies that could reach the same (asymptotic) performance as centralized policies. These results and the conjectures in Section 4.2 can have a significative impact for load balancing applications, as we studied a regime where resources are fully exploited while blocking probability stays arbitrarily low.. We plan to continue studying these regimes both theoretically and using simulations, for instance broadening the scope to include different distributions for the arrival process.

The general study in this work can contribute to our motivating case study, the TDAQ system at CERN, both from theoretical and practical points of view. The general conjectures derived can orient future sizing tasks for the TDAQ farm, and the new load balancing simulation models can now be incorporated into the existing TDAQ models.

REFERENCES

Bergero, F., and E. Kofman. 2011. "PowerDEVS: A Tool for Hybrid System Modeling and Real-Time Simulation". *Simulation* 87(1-2):113–132.

Bergero, F., and E. Kofman. 2014. "A vectorial DEVS Extension for Large Scale System Modeling and Parallel Simulation". *Simulation* 90(5):522–546.

Bonald, T., M. Jonckheere, and A. Proutière. 2004, June. "Insensitive Load Balancing". *ACM Sigmetrics Performance Evaluation Review* 32(1):367–377.

Bonald, T., and A. Proutière. 2003. "Insensitive Bandwidth Sharing in Data Networks". *Queueing Systems* 44(1):69–100.

Bonaventura, M., D. Foguelman, and R. Castro. 2016. "Discrete Event Modeling and Simulation-Driven Engineering for the ATLAS Data Acquisition Network". *Computing in Science & Engineering* 18(3):70–83.

Bramson M., Lu Y., P. B. 2012. "Asymptotic Independence of Queues Under Randomized Load Balancing". *Queueing Systems* 71:247–292.

Christen, G., A. Dobniewski, and G. Wainer. 2004. "Modeling State-Based DEVS Models in CD++". In *Proceedings of MGA, advanced simulation technologies conference*, 105–110. Arlington Virginia, USA.

Collaboration, A. 2008. "The ATLAS Experiment at the CERN Large Hadron Collider". *Journal of Instrumentation* 3(08):S08003.

Foguelman, D. J., M. Bonaventura, and R. D. Castro. 2016. "MASADA: A Modeling and Simulation Automated Data Analysis Framework for Continuous Data-Intensive Validation of Simulation Models". In *Proceedings of the European Simulation and Modeling Conference*, Volume 30, 34–42. SIANI, University of Las Palmas, Spain.

Graham, C. 2000. "Chaoticity on path space for a queueing network with selection of the shortest queue among several". *Journal of Applied Probability* 37(1):198–211.

Halfin, S., and W. Whitt. 1981. "Heavy-Traffic Limits for Queues with Many Exponential Servers". *Operations Research* 29(3):567–588.

Jagerman, D. L. 1974. "Some Properties of the Erlang Loss Function". *Bell System Technical Journal* 53(3):525–551.

Jonckheere, M. 2006. "Insensitive Versus Efficient Dynamic Load Balancing in Networks Without Blocking". *Queueing Systems* 54(3):193–202.

Jonckheere, M., and J. Mairesse. 2010. "Towards an Erlang Formula for Multiclass Networks". *Queueing Systems* 66(1):53–78.

Jonckheere, M., and B. J. Prabhu. 2016, June. "Asymptotics of Insensitive Load Balancing and Blocking Phases". *ACM Sigmetrics Performance Evaluation Review* 44(1):311–322.

Laurito, A., M. Bonaventura, M. E. Pozo Astigarraga, and R. Castro. 2017. "TopoGen: A network Topology Generation Architecture with Application to Automating Simulations of Software Defined Networks". In *Proceedings of the Winter Simulation Conference*, edited by C. Victor et al., Volume 50, 1049–1060. Piscataway, New Jersey: IEEE.

Leino, J., and J. Virtamo. 2006. "Insensitive Load Balancing in Data Networks". *Computer Networks* 50(8):1059–1068.

Mitzenmacher, M. 2001. "The Power of Two Choices in Randomized Load Balancing". *IEEE Transactions on Parallel and Distributed Systems* 12(10):1094–1104.

Mukhopadhyay, A., A. Karthik, R. R. Mazumdar, and F. Guillemin. 2015. "Mean Field and Propagation of Chaos in Multi-Class Heterogeneous Loss Models". *Performance Evaluation* 91:117 – 131. Special Issue: Performance 2015.

Pla, V., J. Virtamo, and J. Martinez-Bauset. 2008. "Optimal Robust Policies for Bandwidth Allocation and Admission Control in Wireless Networks". *Computer Networks* 52(17):3258–3272.

Pozo Astigarraga, M., E. ATLAS Collaboration et al. 2015. "Evolution of the ATLAS Trigger and Data Acquisition System". In *Journal of Physics: Conference Series*, Volume 608, 012006. IOP.

Righter, R., and J. G. Shanthikumar. 1989. "Scheduling Multiclass Single Server Queueing Systems to Stochastically Maximize the Number of Successful Departures". *PEIS* 3:323–333.

Vvedenskaya, N. D., R. L. Dobrushin, and F. I. Karpelevich. 1996. "Queueing System With Selection of the Shortest of Two Queues: An Asymptotic Approach". *Problems of Information Transmission* 32(1):15–27.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation 3rd Edition: Discrete Event and Iterative System Computational Foundations*. Elsevier.

AUTHOR BIOGRAPHIES

MATÍAS BONAVENTURA is a MASc in Computer Science and a PhD student in the Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires and a project associate with the ATLAS TDAQ group (CERN). His research interests are hybrid continuous/discrete modeling and simulation of networked computing systems. His email address is mbonaventura@dc.uba.ar.

MATTHIEU JONCKHEERE received his PhD in applied mathematics from the Ecole Polytechnique (Paris, France). He later completed a postdoctorate at CWI (Amsterdam) and became an assistant professor at Eindhoven University of Technology. He is now a CONICET researcher and professor at the University of Buenos Aires. He is a co-founder of the startup Aristas SRL. http://matthieujonckheere.blogspot.com.ar/ and his email address is mjonckhe@dm.uba.ar.

RODRIGO CASTRO is a Professor in the Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, head of the Simulation Lab, and a researcher at CONICET. His research interests include simulation and control of hybrid systems. His email address is rcastro@dc.uba.ar.