Fluid Limits and Optimal Task Assignment Policies for Locally Pooled Service Systems

Diego Goldsztajn

Fluid Limits and Optimal Task Assignment Policies for Locally Pooled Service Systems

Diego Goldsztajn

PhD affiliation: Eindhoven University of Technology, Mathematics and Computer Science Department
Current affiliation: Inria Center at Université Côte d'Azur, Network Engineering and Operations Team

1. ABSTRACT

Task assignment policies play a central role in many online applications, where service requests or tasks arrive over time and are distributed across parallel servers in a data center or cloud computing platform. The way in which the tasks are distributed across the servers has a tremendous impact on the performance perceived by users and the efficiency of server usage, which has attracted strong interest from our research community. Most of this interest has been directed to the so-called supermarket model, but real-life systems have features that fall outside of this standard framework, raising significant challenges which have been pursued in my doctoral thesis [3]. In particular, my thesis provides novel modeling frameworks and performance analysis techniques that enable the study of interactive online applications with fluctuating demand patterns and networked systems.

2. INTRODUCTION

Effective task assignment policies can drastically improve the quality of service provided to users and simultaneously increase server utilization. As a result, they have attracted substantial interest from our research community, which has focused on the *supermarket model*. In this standard model, a centralized entity called dispatcher assigns each incoming tasks to a server where the task is queued until completed. In this setting, maintaining a balanced distribution of the number of tasks across the queues is optimal under standard assumptions, as well as natural. An extensive survey of load balancing policies for this purpose is provided in [1].

The supermarket model was inspired by applications like web search, file transfer, online shopping and compute jobs. A key characteristic of these applications is that completing a task requires a fixed amount of work, which may depend on the task, but users are flexible regarding the amount of time that they have to wait until the task is finshed; e.g., a file transfer involves the transmission of a number of bits equal to the file size, but users do not know its duration in advance. Another key characteristic is that user satisfaction is determined by the successful completion of the task and its sojourn time, whereas the way in which the task progresses over time is hardly relevant. In particular, the tasks can be queued and need not start right after being submitted.

Interactive applications such as video streaming or online gaming are of a different nature, and thus require a new modeling framework. In these applications the experience of users depends on how task progress over time, the execution of tasks is expected to start rightaway and task durations tend to be predefined. Further, tasks are *elastic*, in the sense that the amount of processing resources required to execute a task can be adjusted over time to cope with congestion without altering duration. For example, a critical resource in video streaming is bandwidth and the amount of bandwidth required to transmit a video can be reduced by transmitting fewer bits per frame. While this can negatively impact video resolution, it is better than interrupting the transmission.

Another important characteristic of the supermarket model is that tasks are assigned to the servers by a dispatcher with full flexibility for selecting any server for each task. When all the servers are identical, this implies that servers with the same number of tasks are exchangeable from the perspective of the dispatcher. However, servers are not exchangeable in real-life data centers and cloud computing systems, which operate under many compatibility constraints between tasks and servers; e.g., artificial intelligence applications require different machine learning models that cannot all be loaded in each server. The absence of exchangeability generated by the compatibility constraints creates important challenges. Remarkably, the empirical distribution of the queue lengths is not Markovian even under exponential assumptions.

The above described differences between the supermarket model and the task assignment problems considered in my thesis are illustrated schematically in Table 1.

	Elastic sojourn Inelastic work	Inelastic sojourn Elastic work
Centralized arrivals Flexible routing	Supermarket model	Section 3
Distributed arrivals Constrained routing	Section 4	-

Table 1: Comparison of the standard supermarket model and the models studied in my doctoral thesis.

3. LOCALLY POOLED SYSTEMS

The model considered in my thesis for studying interactive online applications is based on [5, 6, 8] and is represented in Figure 1. In this model, tasks arrive over time and must be immediately and irrevocably assigned to a processing unit by a dispatcher, but instead of servers with dedicated queues, the processing units are local pools of resources where tasks are served in parallel. The duration of a task is not affected by the number of tasks contending for resources at the same local pool, but the *utility* derived from executing a task does depend on the amount of resources allocated to the task. As

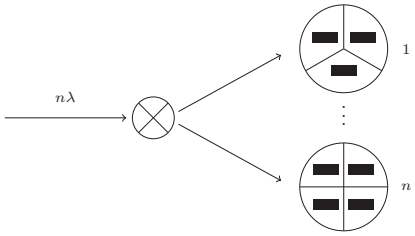

Figure 1: Schematic of a locally pooled service system. The big circles represent n local pools of resources, the black rectangles represent tasks and the crossed circle represents the dispatcher. Resources are equitably shared by tasks.

noted earlier, these features are characteristic of interactive applications such as video streaming, where a local pool of resources is hardware or software that handles multiple user sessions in parallel, and the utility derived from each session is the quality of service provided to the user.

Each local pool of resources behaves as an infinite-server pool, where tasks are served in parallel with independent and identically distributed durations; e.g., in a streaming service the distribution of task durations is mostly given by the duration and popularity of the available videos. All the server pools together form an infinite-server system where the sojourn times of tasks and the total number of tasks at any given time are independent of the way in which tasks are dispatched. However, the task assignment policy determines the distribution of the tasks across the server pools and this has a strong impact on the utility derived from tasks, which depends on the level of congestion at each server pool.

The natural objective is to maximize the aggregate utility provided by all the server pools when the utility derived from a single server pool is a function of the number of tasks in the server pool. For instance, suppose that the quality of service experienced by a user depends on the amount of allocated resources through a function g. If server pool i has r_i units of processing resources and is serving x_i tasks, then the total utility provided by this server pool is

$$u_i(x_i) = x_i g\left(\frac{r_i}{x_i}\right), \tag{1}$$

where r_i/x_i is the amount of resources allocated to each task. The utility functions that we consider need not have the above form, we only assume that u_i is concave; if g is concave and nondecreasing, then (1) is concave.

3.1 Homogeneous systems

In [6, 8] we consider homogeneous systems where all the server pools have the same utility function. If x_i denotes the number of tasks that server pool i has at some given time, then the aggregate utility provided by the system is

$$\sum_{i=1}^{n} u_i(x_i) = \sum_{i=1}^{n} u_1(x_i). \tag{2}$$

If u_i is as in (1), then (2) subsumes the family of α-fair utility functions for bandwidth-sharing networks [14, 20]. Further, (1) and (2) cover a wide range of concave utility functions that have been proposed in [9, 10] as proxies for quality of service in the context of video streaming.

The aggregate utility (2) is Schur-concave with respect to $x = (x_1, \ldots, x_n)$ if u_1 is concave. Thus, keeping a balanced distribution of the number of tasks across the server pools is optimal. A result proved in [12, 19] implies that Join the Shortest Queue (JSQ) is the optimal load balancing policy when task durations are exponentially distributed. However, this policy incurs a prohibitive communication overhead.

	Single-server dynamics	Infinite-server dynamics
Power-of-d policies	[13, 16, 22]	[17]
Token-based policies	[11, 21]	Section 3.1

Table 2: Literature on power-of-d and token-based load balancing policies for different service dynamics.

It was proved in [17] that power-of-d policies achieve a similar performance as JSQ in suitable asymptotic regimes where d and n approach infinity. Instead, we define in [8] a token-based policy that uses an admission threshold ℓ_n, and also prove that it has the same fluid and diffusion limits as JSQ under exponential assumptions, completing Table 2.

3.1.1 Fluctuating demand patterns

Suppose that tasks arrive at rate $n\lambda$ and the average task duration is $1/\mu$. If $\rho = \lambda/\mu$, then the threshold value that yields the same fluid and diffusion limits as JSQ is $\lfloor \rho \rfloor$. This value is typically unknown and time-varying since it depends on the current level of demand. While fluctuating demands are prevalent in practice, their impact on performance has been rarely analyzed rigorously. Nonetheless, in [6, 8] we design and analyze a learning scheme that uses only the tokens stored at the dispatcher for adjusting the admission threshold $\ell_n(t)$ over time to find the optimal value. Also, we establish a fundamental trade-off between the degree of load balance that can be achieved and the stability of the admission threshold, which is related with communication overhead. Some of the results derived in [6] apply even when task durations are Coxian distributed.

The results obtained in [6] are qualitatively similar to fluid limits but apply even when the threshold process ℓ_n is not tight. Instead of establishing process-level convergence, we derive asymptotic bounds for the distance between a process and a constant, and prove that these bounds approach zero over time; this leads to similar conclusions as proving a fluid limit and showing that the fluid dynamics have a global attractor. The threshold processes ℓ_n in [6] may not be tight in the Skorohod J_1-topology since the learning scheme can trigger updates in quick succession, with the time between consecutive updates vanishing in the limit. Similar quickly varying controls with discrete values are fairly common, and the approach in [6] could be used for their analysis.

3.2 Heterogeneous systems

In [5] we study systems as the one depicted in Figure 1 but we assume that server pools can be of different classes. The duration of a task is still determined by the application and does not depend on the class of the server pool where the task is served, but the utility function associated with each server pool is class-dependent. The class of a server pool could reflect the total amount of processing resources that it has; e.g., if the utility functions are as in (1), then r_i depends on the class of the server pool. We assume that the n server pools are split into m classes and that all the server pools of class i have the same utility function u_i. Apart from

being all concave, utility functions associated with different classes can be entirely different, with different shapes.

The heterogeneity of the utility functions implies that the aggregate utility provided by the entire system is generally not Schur-concave anymore. As a result, it is generally not optimal to keep a balanced distribution of the number of tasks across the server pools. In fact, load balancing policies can be *highly suboptimal* in this heterogeneous setup.

In general, the optimal distribution of the tasks at a given time depends intricately on the set of utility functions and the total number of tasks in the system. Let $\alpha_n(i)$ and $\boldsymbol{q}_n(t,i,j)$ be the fraction of server pools of class i and the fraction of server pools of class i that have at least j tasks at time t, respectively. If we define the overall utility as

$$u\left(q_n(t)\right) = \sum_{i=1}^{m}\sum_{j=1}^{\infty} u_i(j)\left[q_n(t,i,j) - q_n(t,i,j+1)\right],$$

then the aggregate utility at time t is just $nu\left(q_n(t)\right)$. The total number of tasks in server pools of class i is equal to the sum of $nq_n(t,i,j)$ over $j \geq 1$. Given that the total number of tasks in the system is $n\kappa$ for some $\kappa \geq 0$, the distributions of $n\kappa$ tasks across the server pools that maximize the aggregate utility solve the following optimization problem:

$$\begin{aligned} \underset{q}{\text{maximize}} \quad & u(q) \\ \text{subject to} \quad & \sum_{i=1}^{m}\sum_{j=1}^{\infty} q(i,j) = \kappa, \\ & 0 \leq q(i,j+1) \leq q(i,j) \leq q(i,0) = \alpha_n(i). \end{aligned} \tag{3}$$

We show that this problem always admits a solution with a particular structure that depends on the marginal utilities defined as $\Delta(i,j) = u_i(j+1) - u_i(j)$.

If tasks arrive as a Poisson process of intensity $n\lambda$ and have mean duration $1/\mu$, then the mean number of tasks in steady state is $n\rho$. We show that if κ is set equal to ρ, then the solution of (3) upper bounds the mean overall utility that any task assignment policy can achieve in steady state. It is worth noting that this upper bound is nonasymptotic and applies to generally distributed task durations.

Using the structure of solutions of (3), we design two task assignment policies that are shown to asymptotically achieve the upper bound if task durations are exponential. The first policy dispatches every incoming task greedily to a server pool that provides the largest increase of utility. This policy called Join the Largest Marginal Utility (JLMU) has the same implementation issues as JSQ, and in fact coincides with JSQ when all the utility functions are identical.

In contrast, the second policy is computationally lighter and generalizes the threshold-based policy of Section 3.1, but instead of just one threshold uses one threshold for each server pool class. Learning the optimal value for all the m thresholds is more challenging than in the homogeneous case. However, using the structure of the solutions of (3), we prove that all the optimal threshold values can be found simultaneously through a single *one-dimensional* search in a suitably defined totally ordered space.

4. NETWORKED SERVICE SYSTEMS

Data centers and cloud computing systems operate under many compatibility constraints between tasks and servers; e.g., content delivery networks must spread content items across many edge servers due to storage limitations. Such constraints are not captured by the standard supermarket model, where tasks arrive at a centralized dispatcher that has full flexibility to assign the tasks to the servers. In [4, 7] we assume instead that the tasks arrive in a distributed way and there exist graphical dispatching constraints.

4.1 Load balancing with dynamic graphs

The analysis of the standard supermarket model hinges critically on the fact that servers with the same number of tasks are exchangeable, but this symmetry is lost when the dispatching decisions are graphically constrained, which complicates the analysis drastically. The main challenge is that compatible servers have strong interactions that make standard mean-field techniques inapplicable. Recent papers have made significant progress in studying constrained load balancing models, e.g., [15, 18, 23]. Nevertheless, they focus on limiting regimes such that the average degree diverges, which makes the strength of interactions between pairs of compatible servers vanish in the limit. The situation where the degrees are uniformly bounded is mathematically more challenging, and more relevant in practice since dispatching devices can handle a limited number of servers.

Results in [2] for interacting particle systems suggest that the limit of the empirical queue length distribution is not tractable when the servers are connected by a graph with bounded degrees. While interactions between neighboring servers are nonvanishing if the graph is static, they average over time if the graph is resampled from a random graph law in a suitable way. We proved this in [4], where we derived a fluid limit when the resampling rate of the graph goes to infinity at an arbitrarily slow rate; notably, the number of arrivals between successive resampling times may diverge.

The fluid dynamics in [4] are globally attracted by

$$q^*(0) = 1, \quad q^*(i) = \lambda q^*(i-1)\varphi\left(q^*(i-1)\right) \quad \text{for} \quad i \geq 1;$$

$q^*(i)$ is the fraction of servers with at least i tasks, λ is the arrival rate per server and φ is the generating function of the limiting degree distribution. We showed that the stationary distribution of the system converges to q^* with the size of the graph, and analyzed the impact of the degree distribution on performance. We proved that regular graphs achieve the best performance when the average degree cannot exceed a given value, and established a phase transition in the tail of the queue length distribution. It decays geometrically when the degree distribution has mass at zero, no matter how small, and doubly-exponentially otherwise. Therefore, *loss in routing freedom* for even a tiny fraction of the tasks carries a severe penalty in tail decay, even if the vast majority of the tasks are assigned in a fully flexible way.

4.2 Saturation in skewed neighborhoods

In the context of static constraints, we considered in [7] systems with multiple dispatchers connected to servers by a bipartite graph. Our results in [7] are the first to cover dispatchers with bounded numbers of compatible servers. We proved that local structures in the bipartite graph, called skewed neighborhoods, create long queues at specific servers in steady state. In order to derive this result we developed new drift-based techniques and stochastic couplings.

The canonical example of a skewed neighborhood is the dandelion network shown in Figure 2. More generally, the neighborhood of a server is skewed if it contains a diverging

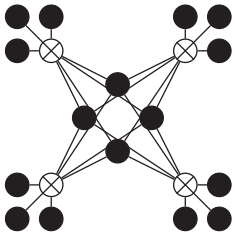

Figure 2: Dandelion network with dispatchers and servers represented by crossed and black circles, respectively. The central servers saturate as the number of dispatchers goes to infinity with the boundary servers per dispatcher fixed.

number of dispatchers with degrees uniformly bounded by some given constant. Even though skewed neighborhoods are favored by task heterogeneity and server specialization, we proved that they can even arise in symmetric random setups, e.g., random bipartite graphs. These results have the potential for improving the efficiency of processing networks, by distributing machine learning models and content items in a way that avoids skewed neighborhoods.

5. REFERENCES

[1] M. van der Boor, S. C. Borst, J. S. H. van Leeuwaarden, and D. Mukherjee. Scalable load balancing in networked systems: A survey of recent advances. *SIAM Review*, 64(3):554–622, 2022.

[2] A. Ganguly and K. Ramanan. Hydrodynamic limits of non-Markovian interacting particle systems on sparse graphs. *arXiv preprint arXiv:2205.01587*, 2024.

[3] D. Goldsztajn. *Fluid limits and optimal task assignment policies for locally pooled service systems*. PhD thesis, Eindhoven University of Technology, 2023.

[4] D. Goldsztajn, S. C. Borst, and J. S. H. van Leeuwaarden. Load balancing with sparse dynamic random graphs. *arXiv preprint arXiv:2305.13054*, 2023.

[5] D. Goldsztajn, S. C. Borst, and J. S. H. van Leeuwaarden. Utility maximizing load balancing policies. *Stochastic Systems*, 13(2):211–246, 2023.

[6] D. Goldsztajn, S. C. Borst, and J. S. H. van Leeuwaarden. Learning and balancing unknown loads in large-scale systems. *Mathematics of Operations Research*, Articles in Advance, 2024.

[7] D. Goldsztajn, S. C. Borst, and J. S. H. van Leeuwaarden. Server saturation in skewed networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(2):1–37, 2024.

[8] D. Goldsztajn, S. C. Borst, J. S. H. van Leeuwaarden, D. Mukherjee, and P. A. Whiting. Self-learning threshold-based load balancing. *INFORMS Journal on Computing*, 34(1):39–54, 2022.

[9] H. Hu, X. Zhu, Y. Wang, R. Pan, J. Zhu, and F. Bonomi. QoE-based multi-stream scalable video adaptation over wireless networks with proxy. In *2012 IEEE International Conference on Communications (ICC)*, pages 7088–7092. IEEE, 2012.

[10] V. Joseph, S. C. Borst, and M. I. Reiman. Optimal rate allocation for video streaming in wireless networks with user dynamics. *IEEE/ACM Transactions on Networking*, 24(2):820–835, 2016.

[11] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg. Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation*, 68(11):1056–1071, 2011.

[12] R. Menich and R. F. Serfozo. Optimality of routing and servicing in dependent parallel processing systems. *Queueing Systems*, 9:403–418, 1991.

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

[14] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.

[15] D. Mukherjee, S. C. Borst, and J. S. H. van Leeuwaarden. Asymptotically optimal load balancing topologies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):1–29, 2018.

[16] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, and P. A. Whiting. Universality of power-of-d load balancing in many-server systems. *Stochastic Systems*, 8(4):265–292, 2018.

[17] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, and P. A. Whiting. Asymptotic optimality of power-of-d load balancing in large-scale systems. *Mathematics of Operations Research*, 45(4):1535–1571, 2020.

[18] D. Rutten and D. Mukherjee. Mean-field analysis for load balancing on spatial graphs. *ACM SIGMETRICS Performance Evaluation Review*, 51(1):27–28, 2023.

[19] P. D. Sparaggis, D. Towsley, and C. Cassandras. Extremal properties of the shortest/longest non-full queue policies in finite-capacity systems with state-dependent service rates. *Journal of Applied Probability*, 30:223–236, 1993.

[20] R. Srikant and L. Ying. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[21] A. L. Stolyar. Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Systems*, 80:341–361, 2015.

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

[23] W. Weng, X. Zhou, and R. Srikant. Optimal load balancing with locality constraints. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(3):1–37, 2020.

6. AUTHOR BIOGRAPHY

Diego Goldsztajn completed his PhD in Mathematics and Computer Science at Eindhoven University of Technology in 2023, advised by Prof. Sem Borst and Prof. Johan van Leeuwaarden. He is currently a postdoctoral researcher at the Inria Center at Université Côte d'Azur. His research has focused on developing stochastic models and scaling limits for analyzing the performance of large decision systems. The research carried out during his PhD was supported by the NWO Gravitation-grant NETWORKS-024.002.003, and has been recognized with the Applied Probability Trust Prize.