

The Power of Clouds: The Power Metric for Cloud Auto-Scaling Under Attack

Jhonatan Tavori
Columbia University
New York, NY, U.S.
j.tavori@columbia.edu

Gil Zussman
Columbia University
New York, NY, U.S.
gil.zussman@columbia.edu

Leonard Kleinrock
UCLA
Los Angeles, CA, U.S.
lk@cs.ucla.edu

ABSTRACT

Modern cloud applications are composed of networks of distributed microservices that process user traffic. Unlike static queueing networks, cloud applications are inherently dynamic: service capacity can be adjusted through *auto-scaling* in response to traffic load surges and attacks, often within a short time. Prior work on worst-case attacks in queueing networks has focused mainly on static settings where the defender may react to attacks, at most, through traffic re-routing over fixed resources.

We propose to study dynamic auto-scaled services through Kleinrock’s *Power* metric [3], which captures the tradeoff between delay and throughput. The metric is closely aligned with the goal of auto-scaling: maintaining sufficient capacity to sustain performance while avoiding unnecessary over-provisioning of resources.

Using Power-optimality analysis, we study the *economic* impact of overload attacks. While auto-scaling improves the system’s ability to preserve latency under attack, it comes at the cost of provisioning additional resources. Using two simple network structures, we show how dynamic capacity control changes the damage-maximizing strategy of the attacker, compared to un-scaled static systems.

1. INTRODUCTION

Cloud microservices are typically managed through *auto-scaling* [1], a mechanism that dynamically adjusts their provisioned compute capacity in response to the observed request load. As distributed microservices commonly operate together to process user traffic in cloud applications, auto-scaling mechanisms continuously adapt capacity, increasing resources when services become overloaded and reducing them when capacity is underutilized.

Interestingly, this objective is closely aligned with Kleinrock’s *Power-optimal operating point* in queueing systems, which optimizes the tradeoff between throughput and delay in a single performance metric [3]. Power is the ratio of system’s “goodness” divided by “badness”, which in queueing systems can be expressed as the ratio of efficiency (utilization) to normalized response time [4]. Optimizing this metric yields the operating point that balances efficient utilization use against the delay caused by congestion. This operating point is well matched to auto-scaling, which aims to tune resource capacity so that services are provisioned

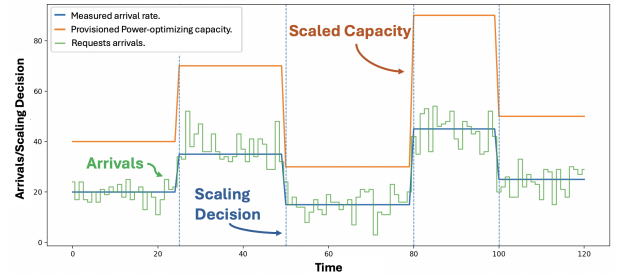


Figure 1: Illustration of auto-scaling under a time-varying arrival process. As the arrival $\lambda(t)$ changes, the auto-scaling target may track the Power-optimal $M/M/1$ operating point at $2\lambda(t)$.

enough to maintain performance, but not so much that resources are wasted through over-provisioning (see Fig. 1).

The Power-optimal operating point has been established in a series of works under several queueing models, making it an analytically meaningful basis for studying dynamic resource control in a variety of settings [4, 5].

Therefore, we propose to use the Power metric as a foundation for analyzing the operation of Power-optimized auto-scaled cloud networks. In this work, we study the *economic* impact of overload attacks, quantified through the additional capacity that must be provisioned in response.

Prior work on worst-case attacks in queueing systems has mainly considered static or only partially adaptive settings, where service capacity is fixed and the defender can react primarily through re-routing. In particular, [7] showed that damage-maximizing attacks tend to concentrate on specific parts of the network, namely on bottlenecks. Once auto-scaling mechanisms are introduced, however, the defender may respond by provisioning additional resources, thereby reducing the exposure of any single bottleneck. This response, while mitigating performance degradation, comes at a cost, as highlighted by [6]: part of the attack’s effect is shifted from performance damage to economic damage, since the additional capacity provisioned in response to the attack must itself be paid for.

By using the Power-optimal operating point, we obtain a tractable way to study the scaling cost induced by overload attacks. We use it to examine how dynamic auto-scaling changes the damage-maximizing strategy of the attacker.

2. MODELING AND FORMULATION

A microservices based cloud application can be modeled as a queueing network $G = (N, L)$, where each node $i \in N$ represents a microservice and each directed link $(i, j) \in L$ represents a possible request transition from service i to service j . A similar network abstraction of microservice applications was previously used in [2].

2.1 Service Requests and Capacity

Requests may arrive externally to the application and may then transition through the services network according to the application logic. Let λ_i^{ext} denote the external arrival rate into service i , measured in user requests per second, and let $r_{i,j}$ denote the probability that a request completing service at node (microservice) i is routed to node j . The effective arrival rate at service i , denoted by λ_i , is induced jointly by the external arrivals and the internal transitions in the network. In standard flow-balance form, the effective arrival rates satisfy $\lambda_i = \lambda_i^{\text{ext}} + \sum_j \lambda_j r_{j,i}$.

Each microservice i is modeled as a queue with effective requests arrival rate λ_i , per-unit service rate μ_i , and allocated capacity c_i , so that its effective processing rate is $\mu_i \cdot c_i$.

2.2 Auto Scaling and Power

We consider cloud microservices as the processing nodes in our model. Thus, their allocated capacities may change dynamically over time through *auto-scaling* [1], rather than remain fixed.¹

As discussed in the introduction, we adopt the Power-optimal operating point as the basis for optimized auto-scaling decisions, since it captures the throughput-delay trade-off in a single quantity, $P(\rho)$, defined as:

$$P(\rho) = \frac{\rho}{\mu T(\rho)}$$

where ρ is the system utilization, $T(\rho)$ is the mean delay and $T(0) = 1/\mu$.

Let P^* denote the optimal Power value and ρ^* the corresponding optimal utilization. The auto-scaler adjusts the provisioned capacity at node i , c_i , accordingly so as to keep the service near ρ^* . We denote the capacity required at service i to maintain the Power-optimal operating point under effective arrival rate λ_i by $c_i^*(\lambda_i)$.

2.3 Attacking the System

We focus on overload attacks (e.g., DDoS attacks), where the attacker injects additional requests traffic into the system. We represent an attack by a vector $\mathbf{x} = (x_1, \dots, x_n)$, where $x_i \geq 0$ denotes the additional adversarial external arrival rate injected into service i . Accordingly, the external arrival rate into service i becomes $\lambda_i^{\text{ext}} + x_i$.

We denote by $\lambda_i(\mathbf{x})$ the effective arrival rate at service i under the attack $\mathbf{x}$. Once node i experience this change from λ_i to $\lambda_i(\mathbf{x})$, the auto-scaler responds by updating the capacity of service i from $c_i^*(\lambda_i)$ to $c_i^*(\lambda_i(\mathbf{x}))$, in order to restore the Power-optimal operating point.

The additional capacity required to restore the Power-optimal operating point at service i is thus

$$\Delta c_i(\mathbf{x}) = c_i^*(\lambda_i(\mathbf{x})) - c_i^*(\lambda_i).$$

¹This differs from classical capacity-assignment problems in routing networks, where node capacities are usually fixed during system operation.

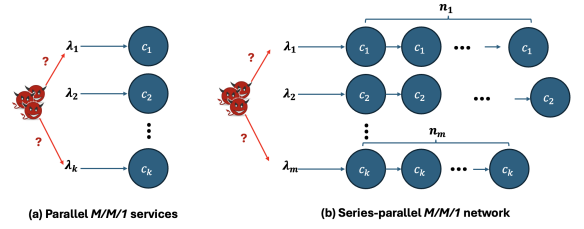


Figure 2: Two network structures considered in the analysis: parallel queues and parallel chains.

Accordingly, we measure the *economical damage* of an attack by the total additional capacity required to restore Power-optimal operation across the network:

$$D(\mathbf{x}) = \sum_{i=1}^n \Delta c_i(\mathbf{x}).$$

The attacker seeks to choose $\mathbf{x}$ so as to maximize $D(\mathbf{x})$, subject to its attack-budget constraints.²

3. DAMAGE-MAXIMIZING ATTACK

In this section, we assume that each service behaves as an $M/M/1$ queue. That is, the arrival process is Poisson, and the service time of a request processed at microservice i is exponentially distributed with mean $1/\mu_i c_i$.

For the $M/M/1$ queueing system, Kleinrock [4] showed that Power is maximized at $P^* = 1/4$, attained at $\rho^* = 1/2$. In our setting, this is equivalent to

$$\mu_i c_i = 2\lambda_i. \quad (1)$$

The result was later shown to hold under any stable Jackson networks as well, where the Power-optimal regime is likewise characterized by load $\rho^* = 1/2$ at each node [5].

Under an attack $\mathbf{x}$, which increases the effective arrival rate at service i by x_i , we have $\lambda_i(x) = \lambda_i + x_i$. Thus, by Eq. (1), in order to maintain the Power-optimal operating point the auto-scaler must now satisfy: $\mu_i c_i^*(\lambda_i(x)) = 2\lambda_i(x) = 2(\lambda_i + x_i)$. It follows that to maintain the Power-optimality at node i , the additional capacity required is

$$\Delta c_i(x) = c_i^*(\lambda_i(x)) - c_i^*(\lambda_i) = \frac{2x_i}{\mu_i}. \quad (2)$$

We begin by considering the attacker's damage-maximizing strategy for a set of parallel $M/M/1$ queues (see Fig. 2a). We then study the corresponding strategy in a parallel series-network consisting of m parallel series-networks whose lengths may differ (see Fig. 2b).

3.1 Parallel Queues

We consider a set of n parallel $M/M/1$ queues, of the same service type (i.e., $\mu_i = \mu_j$ for any $1 \leq i, j \leq n$), under an attack $\mathbf{x}$. By Eq. (2), restoring to the Power-optimality at node i requires additional capacity

$$\Delta c_i(\mathbf{x}) = \frac{2x_i}{\mu}$$

²A common budget constraint limits the total injected attack traffic, e.g., $\sum_i x_i \leq B$. Under such a constraint, the attacker seeks to maximize the induced damage $D(\mathbf{x})$ for a given attack effort rate of B malicious requests.

where μ is the shared service rate. Hence, the total damage cost over the entire n queues is:

$$D(\mathbf{x}) = \sum_{i=1}^n \Delta c_i(\mathbf{x}) = \frac{2}{\mu} \sum_{i=1}^n x_i. \quad (3)$$

Looking at Eq. (3), we observe that:

Remark 1. For parallel queues with identical service type, the total damage cost depends only on the total attack size $\sum_{i=1}^n x_i$, and not on how the attack is distributed across the queues.

3.2 Series Networks of Different Lengths

We next consider a series-parallel topology consisting of m parallel series (chain) networks, where series network i consists of n_i sequential $M/M/1$ queues and carries effective arrival rate λ_i .

Under an attack $\mathbf{x}$, which targets the first node in each chain i , all its nodes carry the same flow $\lambda_i(x) = \lambda_i + x_i$. Thus, considering Eq. (2), the total cost of restoring series network i to its Power-optimal operating cost is: $\Delta C_i(x) := n_i \Delta c_i(x) = \frac{2n_i}{\mu} x_i$.

Hence, the total damage cost over all m chains is

$$D(\mathbf{x}) = \sum_{i=1}^m \Delta C_i(x) = \frac{2}{\mu} \sum_{i=1}^m n_i x_i. \quad (4)$$

Remark 2. In a series-parallel topology, the damage induced by attacking series network i scales linearly with its length n_i . Thus, under a fixed attack budget, longer series networks induce larger damage per unit of attack.

4. DISCUSSION

Auto-Scaling Cost

In static settings, where capacity cannot change dynamically, damage is typically measured through latency, and concentrated attacks have been shown to maximize it. This follows from the fact that the $M/M/1$ queueing delay is convex in utilization: when capacity is fixed, a small increase in load causes some damage, but pushing the same queue further can induce a much larger increase in delay.

In our setting, however, capacity is dynamic and re-adjusted by auto-scaling, so the attacker no longer exploits a fixed bottleneck in the same way. In particular, due to the $M/M/1$ Power-optimal operating point, we've seen that the cost inflicted by the attack depends only on the total injected traffic, and not on how that traffic is distributed across the parallel queues³. This suggests that dynamic capacity control makes the network more resilient, in the sense that it reduces the exposure of specific bottlenecks and thus changes the strategy of the worst-case attack.

Attacker's Approach: Keep the Pipe Just Fuller?

When considering auto-scaling mechanisms, it is important to account for their operational dynamics. In practice, scaling actions may take some time, and during the scaling process resources may be temporarily unavailable or less effective while instances are created, removed, or resized. Thus,

³Once one considers series networks, however, the damage cost is multiplied by the series length, since the attack-induced load must be absorbed at every queue along the series rather than only at a single bottleneck.

even if the steady-state damage cost depends mainly on the total attack size, the transient effect of repeated small attacks may still be significant.

This suggests a different attack strategy from the classical concentrated overload. If there is no clear steady-state benefit to focusing on a single service, and the cost-effectiveness is roughly proportional to attack size, then a natural policy is to "keep the pipe just fuller". That is, rather than attempting to overwhelm the system at once, the attacker may invest in repeated small bursts that push services slightly above their Power-optimal operating point, repeatedly triggering scaling actions and control overhead. Such attacks may not cause immediate collapse, but can repeatedly destabilize the system and accumulate operational cost over time.

Horizontal Scaling: The $M/M/m$ Model

Common scaling techniques in cloud systems include *vertical scaling*, where the capacity of existing instances is increased, and *horizontal scaling*, where additional service instances are added [1]. Beyond the $M/M/1$ model, which captures capacity scaling at a single-server service, the $M/M/m$ model is therefore also of interest, as it captures a microservice implemented by multiple parallel servers.

We see from [3] that when $m \rightarrow \infty$, there are, on average, approximately m customers in the system at optimum Power. I.e., the system aims to *keep the pipe just full, but no fuller* [4]. This suggests that the attack strategies discussed here may extend to $M/M/m$ settings, particularly in cloud environments where m can be large. A precise analysis of this attack-induced cost is the subject of ongoing work.

Further Work

Several extensions remain for future work. It is natural to extend the analysis beyond the considered topologies to a general network structure. Further, our model does not account explicitly for small scaling delays, which may substantially affect the attack dynamics in practice. As discussed above, it would be valuable to extend the analysis to richer queueing models including $M/M/m$ and $M/G/*$ systems.

5. REFERENCES

- [1] Amazon Web Services. Aws auto scaling. <https://aws.amazon.com/autoscaling/>, 2026.
- [2] H. Cao, X. Liu, H. Guo, J. He, and X. Liu. Queueflower: Orchestrating microservice workflows via dynamic queue balancing. In *Proc. IEEE ICWS'24*. IEEE, 2024.
- [3] L. Kleinrock. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *ICC 1979; International Conference on Communications, Volume 3*, pages 43–1, 1979.
- [4] L. Kleinrock. Internet congestion control using the power metric: Keep the pipe just full, but no fuller. *Ad hoc networks*, 80:142–157, 2018.
- [5] G. Rubino. On kleinrock's power metric for queueing systems. In *Proc. ICCCN'11*. IEEE, 2011.
- [6] M. Sides, A. Bremler-Barr, and E. Rosensweig. Yo-yo attack: vulnerability in auto-scaling mechanism. In *Proc. ACM SIGCOMM'15*, pages 103–104, 2015.
- [7] J. Tavori and H. Levy. How to attack and congest delay-sensitive applications on the cloud. In *Proc. IEEE INFOCOM'23*, May 2023.