FluxNinja

Graceful Degradation: Keeping The Lights On When Everything Goes Wrong

Tanveer Gill
CTO, FluxNinja

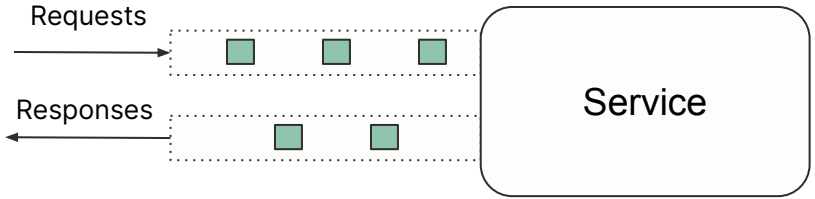
https://www.linkedin.com/in/gill-tanveer/

@GillTanveer89

Agenda

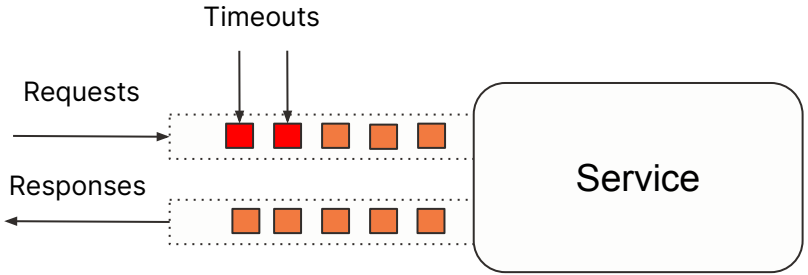
1. Understanding the Consequences of Lack of Load Management in Microservices
2. Throw more compute at the problem? Examining the Limitations of Auto-Scaling
3. Concurrency limits are effective but challenging to implement
4. Smart concurrency limits and prioritized load shedding for any service with Aperture

What happens when a service gets overwhelmed?

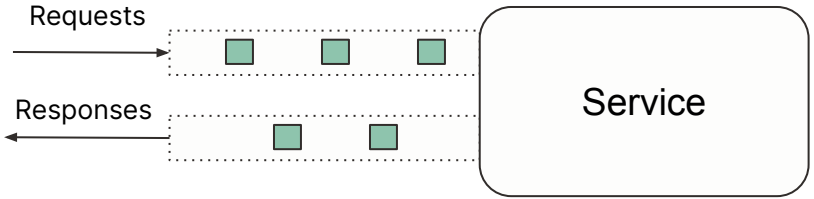


Regular load

Overload

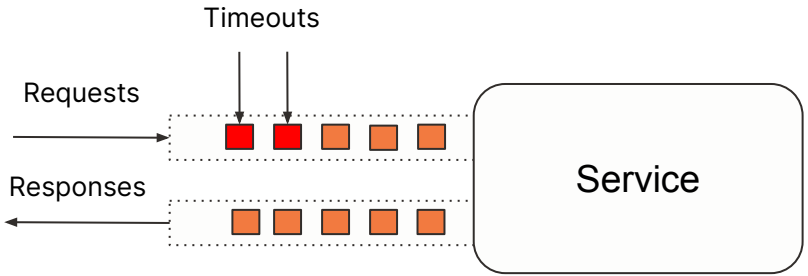


What happens when a service gets overwhelmed?



Regular load

Overload

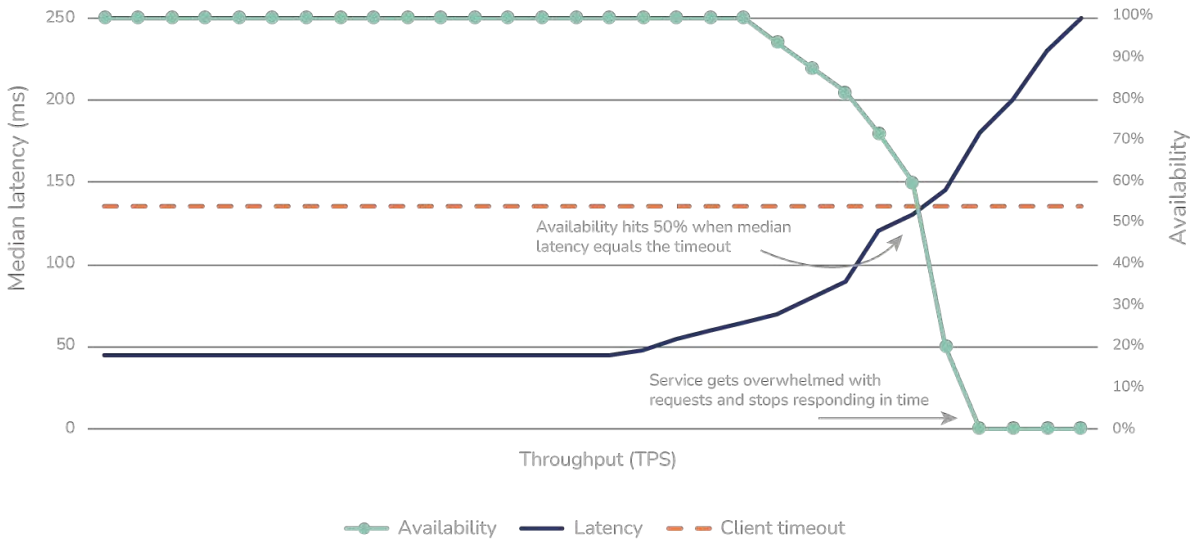


Little's Law

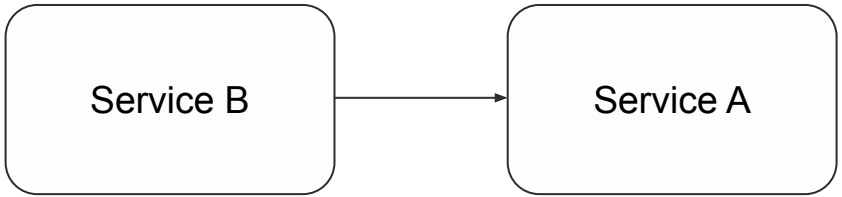
$L = \lambda W$

L = Requests in progress
λ = Average Throughput
W = Average Response time

Visualizing the Impact of Overload on a Service

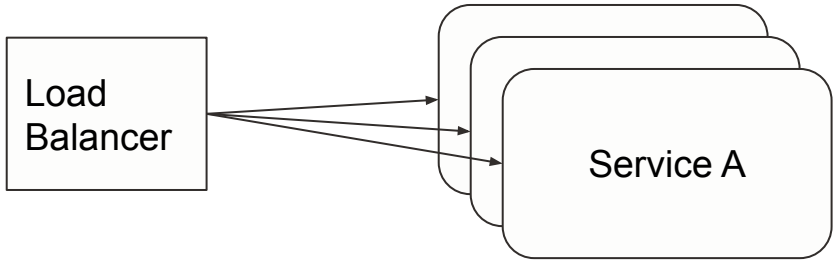


Overloads lead to cascading failures

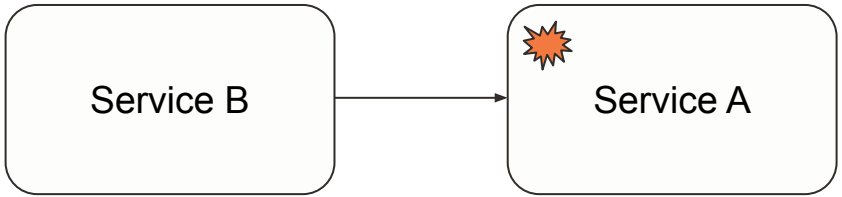


Across services

Laterally within a service

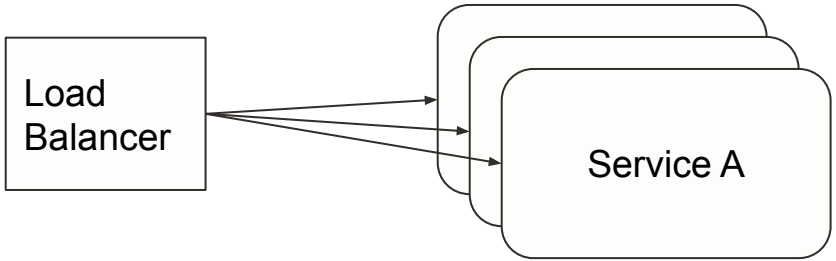


Overloads lead to cascading failures

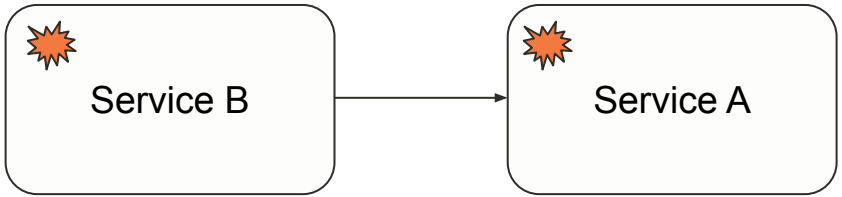


Across services

Laterally within a service

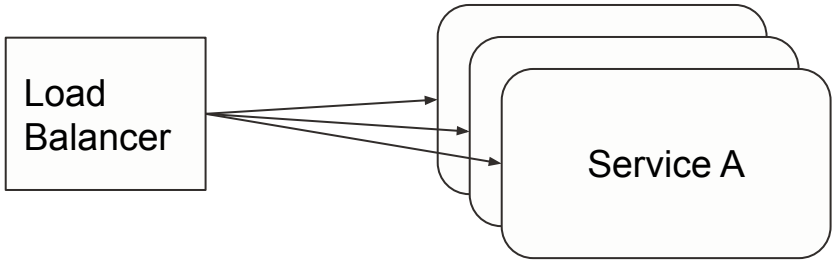


Overloads lead to cascading failures

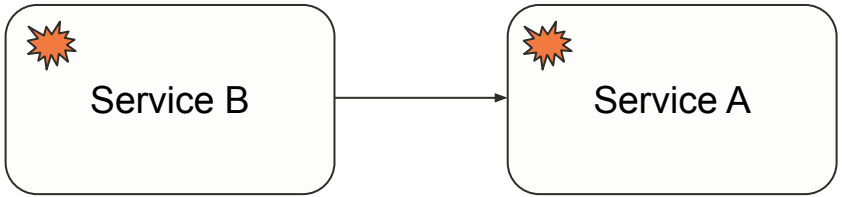


Across services

Laterally within a service

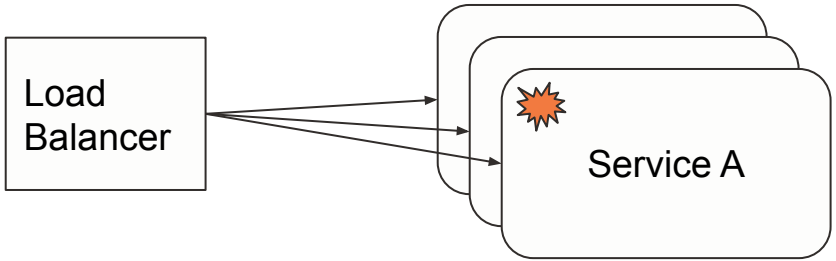


Overloads lead to cascading failures

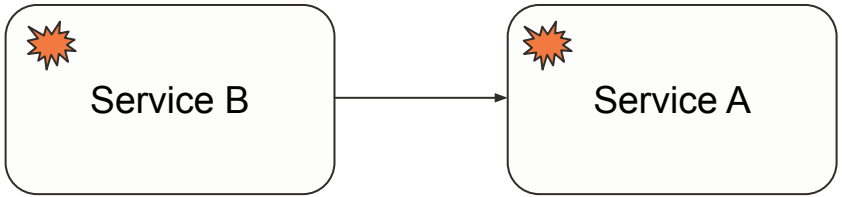


Across services

Laterally within a service

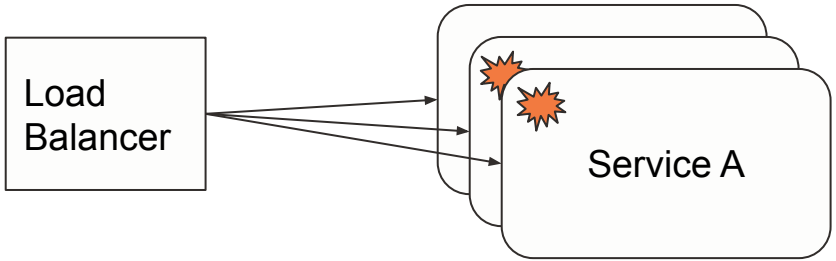


Overloads lead to cascading failures

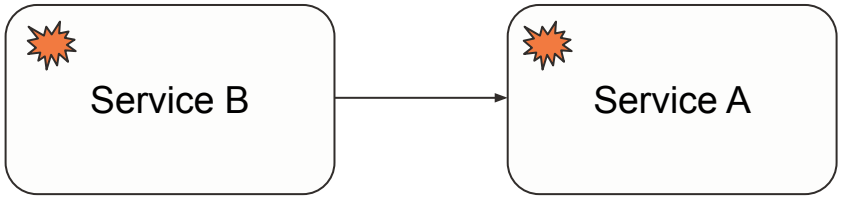


Across services

Laterally within a service

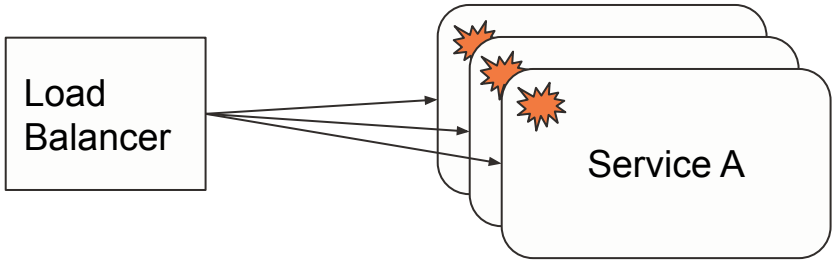


Overloads lead to cascading failures



Across services

Laterally within a service



Limitations of auto-scale

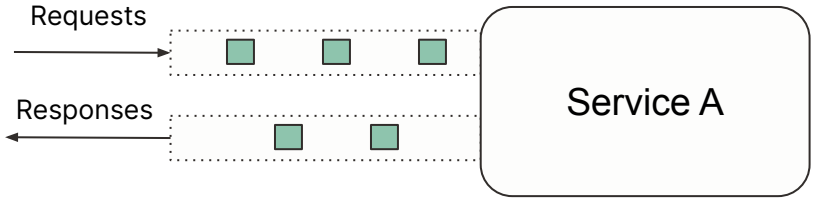
- Slow to respond
- Limited by resource usage quotas
- May contribute to load amplification at dependencies

Limitations of auto-scale

- Slow to respond
- Limited by resource usage quotas
- May contribute to load amplification at dependencies

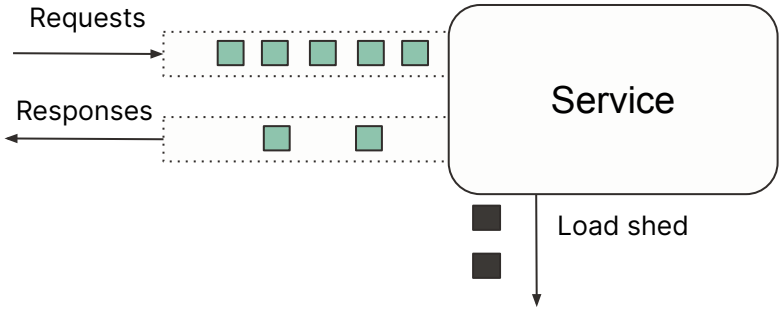
> *"This traffic surge caused a classic cascading failure scenario. Numerous backing services for the API—Cloud Datastore, Pokémon GO backends and API servers, and the load balancing system itself—exceeded the capacity available to Niantic's cloud project. The overload caused Niantic's backends to become extremely slow (rather than refuse requests), manifesting as requests timing out to the load balancing layer"*
>
> — Google SRE Workbook: Case Study 1: Pokémon GO on GCLB

Managing load with concurrency limits

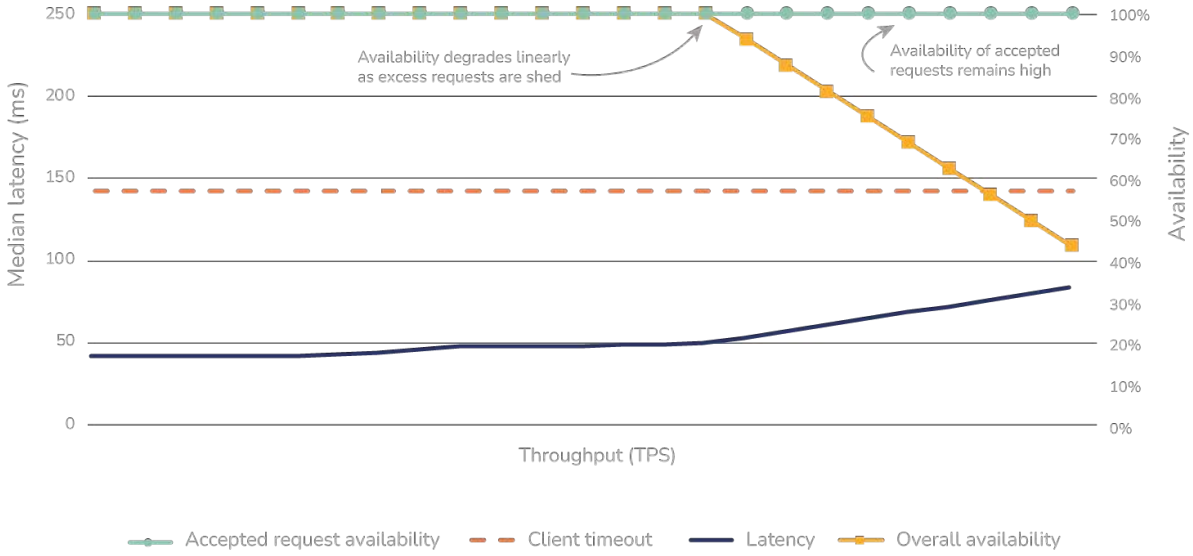


Regular load

Overload



Visualizing the effect of concurrency limits



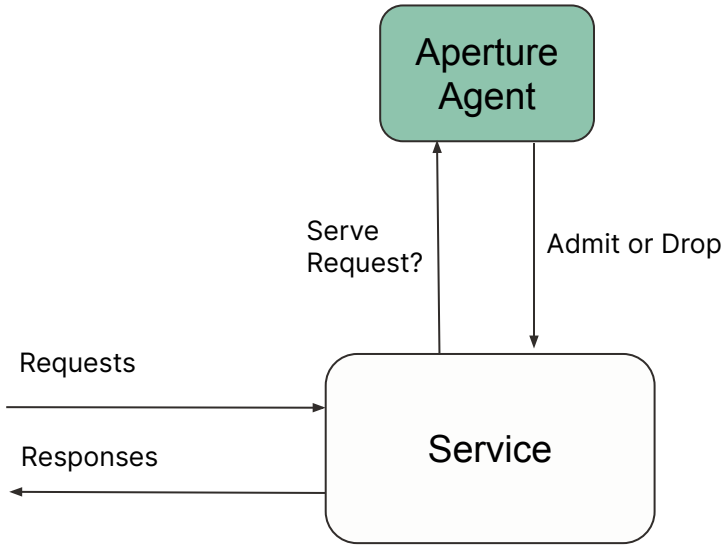
Challenges with implementing concurrency limits

- Determining the ideal maximum number of concurrent requests
  - Setting the limit too low can result in rejected requests
  - Setting the limit too high can lead to slow and unresponsive servers
- Difficulty in determining the ideal value in a constantly changing microservices environment

Challenges with implementing concurrency limits

- Determining the ideal maximum number of concurrent requests
  - Setting the limit too low can result in rejected requests
  - Setting the limit too high can lead to slow and unresponsive servers
- Difficulty in determining the ideal value in a constantly changing microservices environment
- **Need for dynamic and adaptive concurrency limits to adapt to changing workloads and dependencies**

Implementing concurrency limits with Aperture

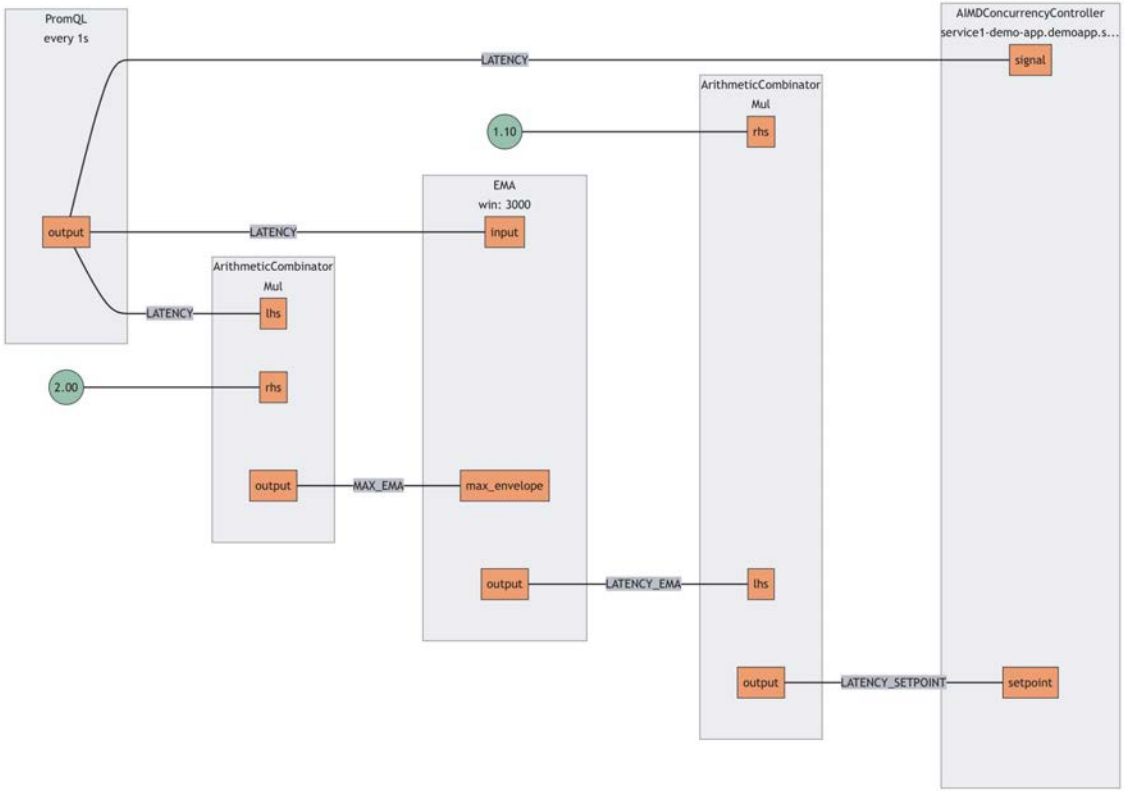


APERTURE

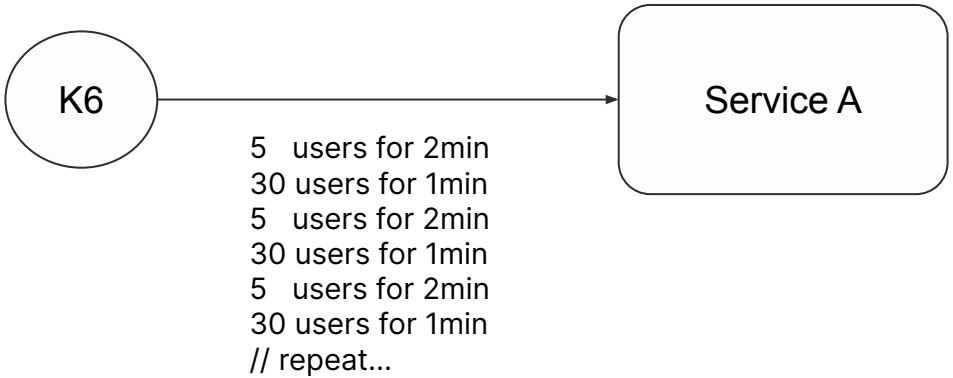
FluxNinja

- Open source reliability automation platform
- Declarative policy language visualized as a circuit graph
- Used for concurrency & rate limiting, workload prioritization, auto-scaling
- Integrates with popular language frameworks
- Supports seamless insertion via service mesh (Envoy)

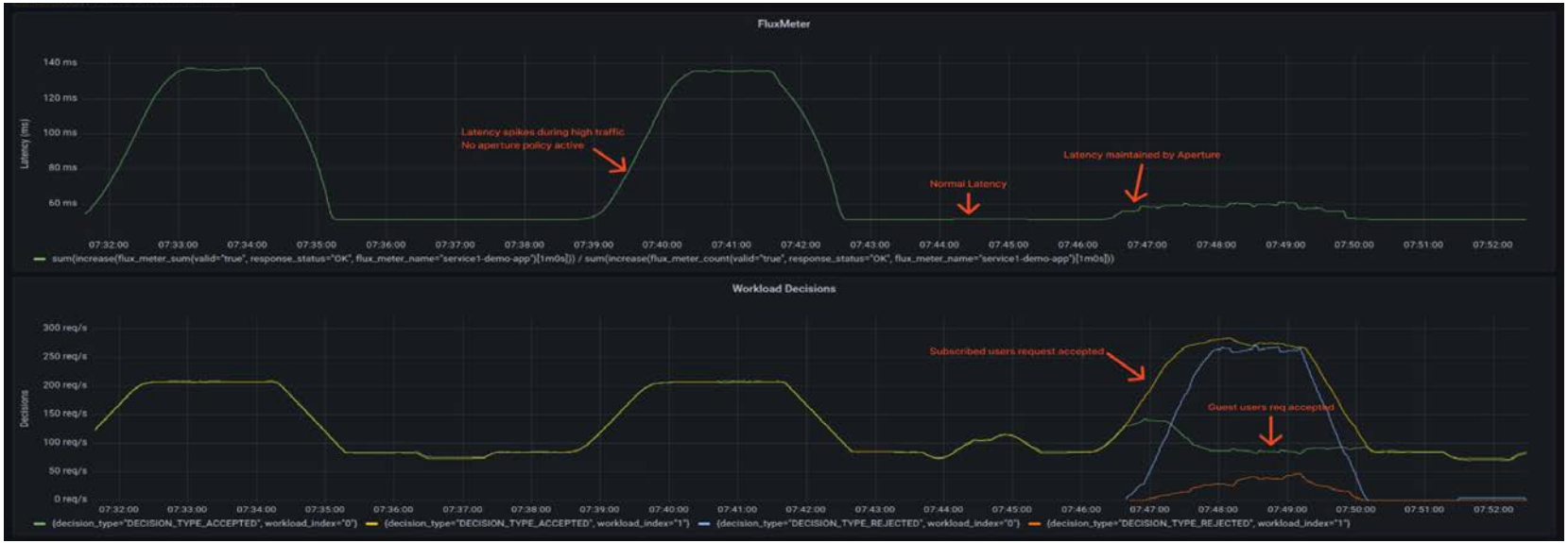
Aperture Policy



Test traffic scenario



Performance before and after Aperture's policy



Prioritized Load Shedding

Policy: Subscribed users are 4x higher priority than guest users

Conclusion

- Designing microservices to gracefully handle overload improves application reliability
- Aperture is a platform for reliability automation
- Aperture brings rate limits, concurrency limits, and load-based auto-scaling to any service
- Aperture integrates with Prometheus and does continuous signal processing on metrics for control and automation
- Check out the Aperture project on GitHub and provide feedback: https://github.com/fluxninja/aperture