How Cloudflare Was Able to Support 55 Milli Requests per Second With Only 15 Postgres Clusters

#32: Learn More - Awesome PostgreSQL Scalability (5 minutes)

NEO KIM
JAN 12, 2024

This post outlines how Cloudflare scaled to 55 million requests per second with 15 Postgres clusters. If you want to learn more, scroll to the bottom and find the references.

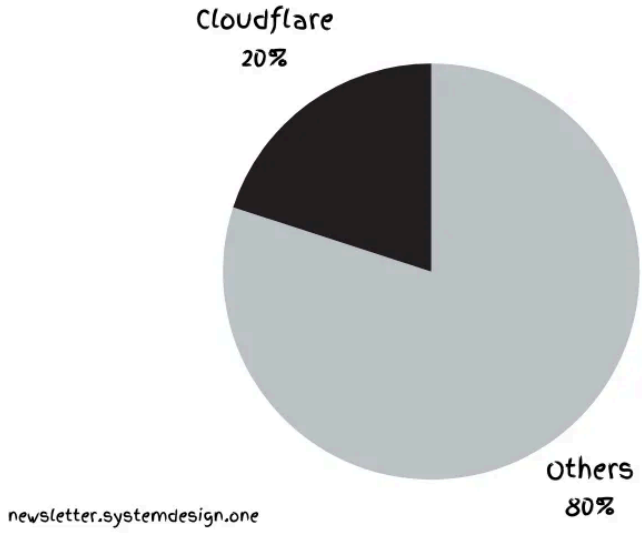
- *Share this post & I'll send you some rewards for the referrals.*

July 2009 - California, United States.

A team of entrepreneurs creates a content delivery network (**CDN**) to make the Internet faster and more reliable. And called it Cloudflare.

They faced various challenges in the early stages of development.

Yet their growth rate was spectacular.



Overall Internet Traffic

Now they serve 20% of Internet traffic at 55 million HTTP requests per second.

And they did it with only 15 PostgreSQL clusters.

A **cluster** is a group of database servers.

They use Postgres to store the metadata of services and handle the Online Transaction Processing (**OLTP**) workload.

Yet supporting many tenants having different load conditions within a cluster is a problem.

A **tenant** is an isolated data space for a specific user or group.

PostgreSQL Scalability

Here's how they use PostgreSQL at extreme scalability:

1. Resource Usage

Most clients compete with each other for Postgres connections.

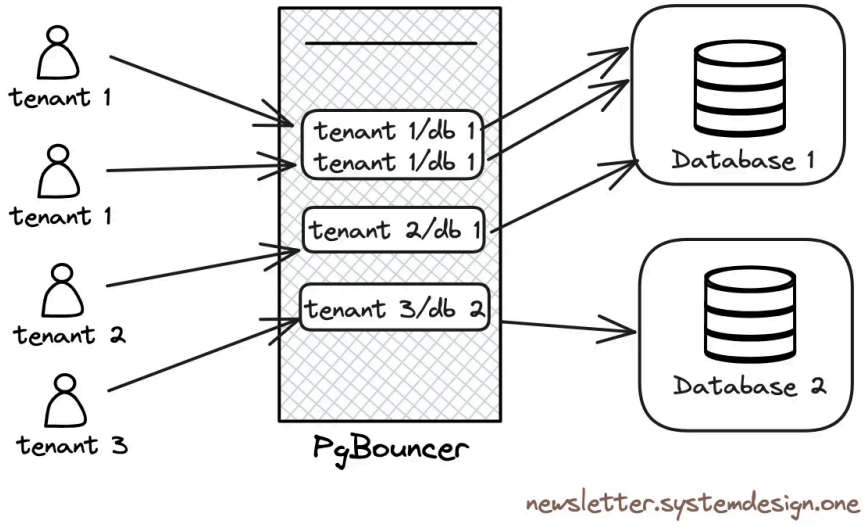
But Postgres connections are expensive because each connection is a separate process at the operating system (**OS**) level.

And each tenant has a unique workload, so it's difficult to create a global throttle value.

Besides throttling misbehaving tenants manually is a huge effort.

A single tenant might issue an expensive query. Thus they might starve the neighboring tenants and block them.

Also a query is difficult to isolate once it reaches the database server.



Connection Pooling With PgBouncer

So they use *PgBouncer* as a connection pooler in front of Postgres.

PgBouncer acts as a TCP proxy that holds a pool of connections to Postgres.

A tenant connects to the PgBouncer instead of Postgres directly. Thus it limits th number of Postgres connections to prevent connection starvation.

Also constant opening and closing of database connections are expensive. PgBo avoids it through persistent connections.

Besides they use PgBouncer to throttle tenants issuing expensive queries at runt

2. Thundering Herd Problem

Thundering herd occurs when many clients query a server concurrently. It degra database performance.

Thundering Herd

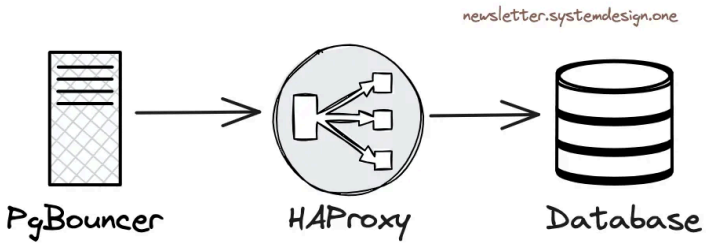
The state of an application gets initialized when redeployed. It results in the application creating many connections to the database at once.

Thus it causes a thundering herd as tenants compete with each other for Postgr connections.

They use *PgBouncer* to throttle the number of Postgres connections created by specific tenant.

3. Performance

They don't run Postgres in the cloud. Instead on bare metal servers without virtualization for high performance.



Load Balancing Traffic Between Database Instances

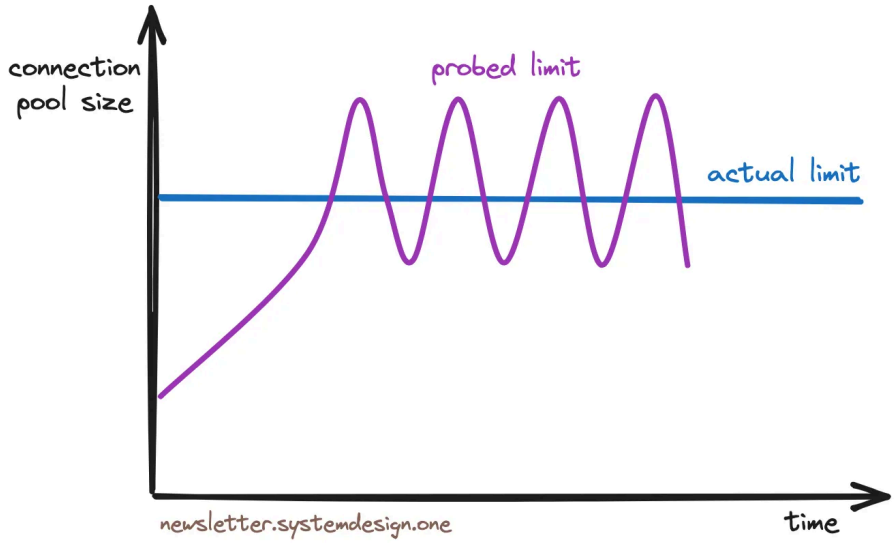
They use *HAProxy* as an L4 load balancer.

PgBouncer forwards queries to HAProxy.

And HAProxy load balance traffic across primary and secondary database read replicas.

4. Concurrency

The performance degrades if there are many tenants issuing concurrent queries



Congestion Avoidance Algorithm Throttling Tenants

So they use the *TCP Vegas congestion avoidance algorithm* to throttle tenants. I the optimal number of packets that can be sent concurrently.

The algorithm does that by sampling each tenant's transaction round trip time (**F** to Postgres.

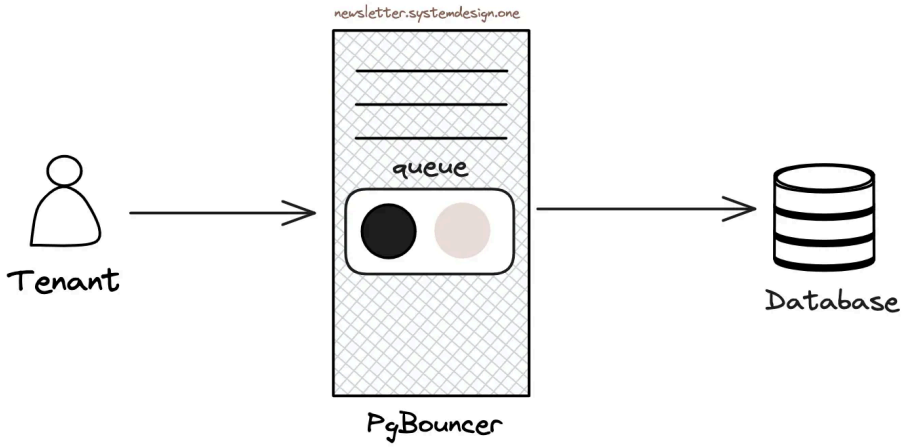
It then increases the connection pool size as long as the RTT doesn't degrade.

Thus it throttles traffic before resource starvation happens.

5. Ordering Queries

They rank queries at the PgBouncer layer using queues.

The queries get ordered in the queue based on their historical resource consum
Put another way, the queries that need more resources get moved to the end of
queue.



Ordering Queries in Priority Queue

Also they enable priority queueing only in peak traffic to prevent resource starva
In other words, a query doesn't wait forever at the end of the queue in normal tra

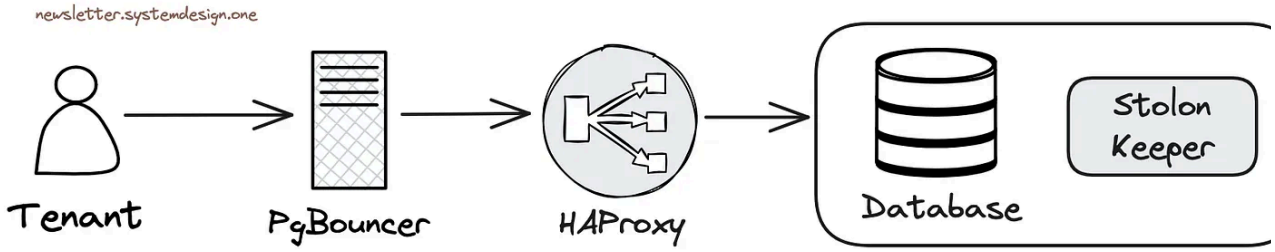
This approach improves the latency of most queries.

Yet a higher latency is observed for tenants that issue expensive queries in peak traffic.

6. High Availability

They use the *Stolon* cluster manager for the high availability of Postgres.

Stolon replicates data across Postgres instances.



High Availability of Database With Stolon

Besides Stolon elects Postgres leaders and does failover in peak traffic.

They replicate a database cluster across 2 regions for high availability.

While each cluster consists of 3 database servers within a single region.

The writes get routed to the primary region.

And data get asynchronously replicated to the secondary region. The reads are r to the secondary region.

Besides they test connectivity between components to find network partitions proactively.

They do chaos testing for resilience. And set up extra network switches and rout avoid network partitions.

They use the *pg_rewind* tool for synchronizing Postgres clusters. It replays missi writes to the primary server that came back online after a failover.

They run more than a hundred database servers including replicas.

And offer services like DNS Resolver, Firewall, and DDoS Protection.

They use a combination of OS resource management, queueing theory, congest algorithms, and even statistics for PostgreSQL scalability.

👋 PS - Are you unhappy at your current job?

And preparing for system design interviews to get your dream job can be stressf

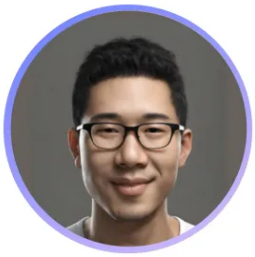
Don't worry, I'm working on content to help you pass the system design intervie make it easier - you spend only a few minutes each week to go from 0 to 1. Yet p subscription fees will be higher than current pledge fees.

So pledge now to get access at a lower price.

"This newsletter is great to level up your system design knowledge." Gregor

Consider subscribing to get simplified case studies delivered straight to your inb

Type your email... Subscribe



Follow me on LinkedIn | YouTube | Threads | Twitter | Instagram | Bluesky

Thank you for supporting this newsletter. Consider sharing this post with your fri and get rewards. Y'all are the best.

References

- Source taken from Vignesh Ravichandran
- Performance isolation in a multi-tenant database environment
- Cloudflare: Performance isolation in multi-tenant DB
- An introduction to stolon: cloud native PostgreSQL high availability
- Cloudflare dishes up the stats on internet traffic in 2023
- Open sourcing our fork of PgBouncer
- PostgreSQL pg_rewind documentation

152 Likes • 9 Restacks

← Previous Nex

Discussion about this post

Comments Restacks

Write a comment...

Marcos F. Lobo 12 Jan 2024

Liked by Neo Kim

It would be interesting to know, related to the performance section, what kind of discs the using in the baremetal Postgres servers. I could imagine they are some advanced SSD disl was working at CERN I witnessed changing the discs for the storage for Ceph

LIKE (3) REPLY

1 reply by Neo Kim

Neha Thakur 12 Jan 2024

Liked by Neo Kim

Learned some exciting features of postgres

LIKE (3) REPLY

1 reply by Neo Kim

13 more comments...