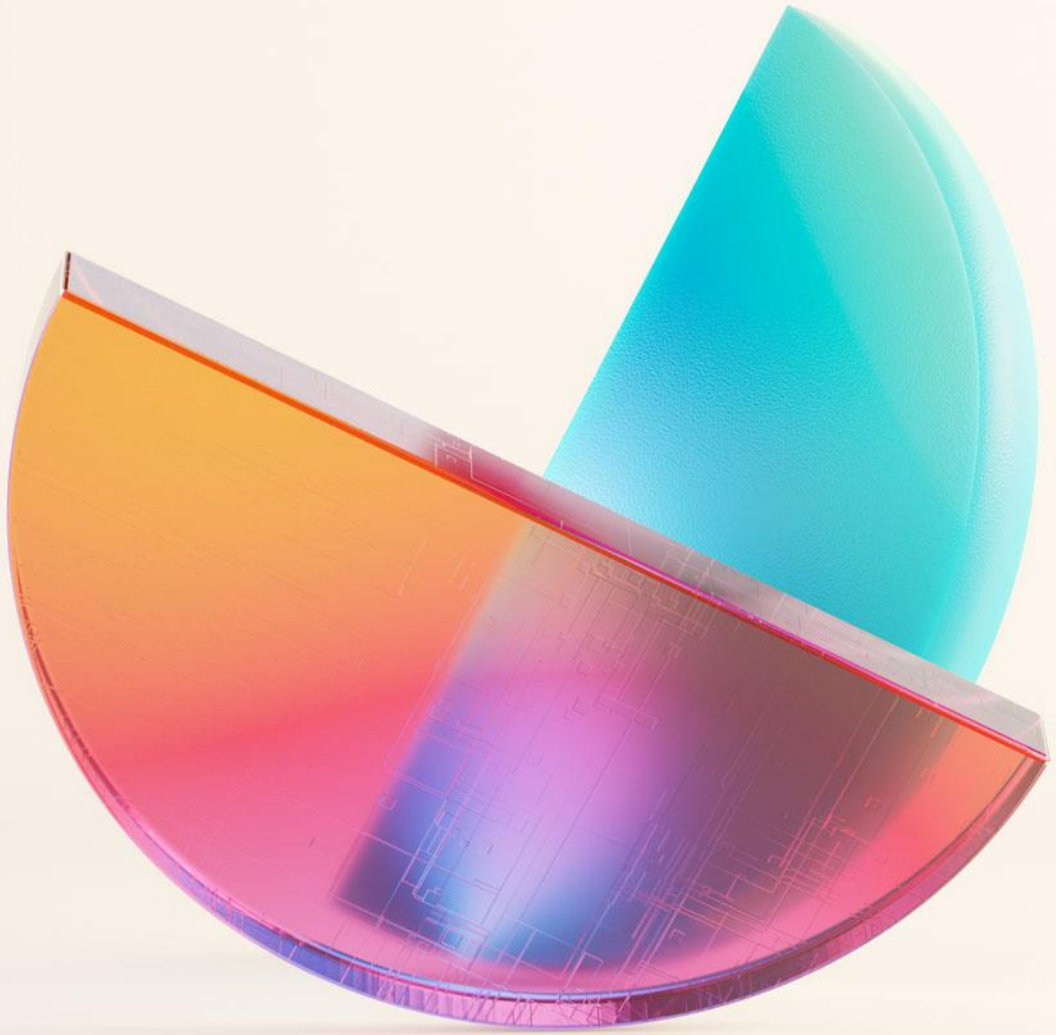


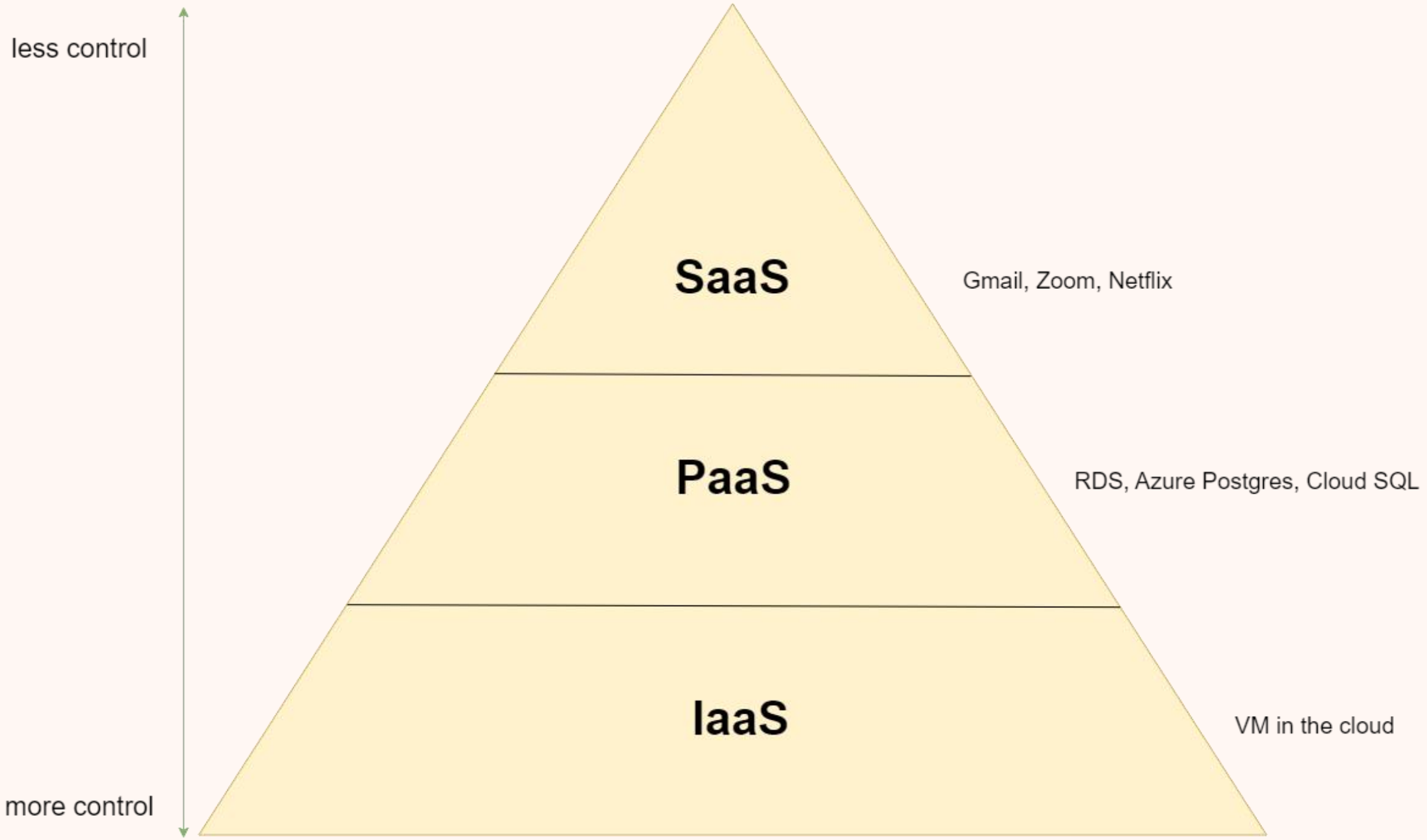
Best Practices in PostgreSQL Tuning: Navigating Key Performance Bottlenecks in the Cloud

Alicja Kucharczyk

Agenda

- What is PaaS
- Geographies and latency
- Evolving Technologies: Yesterday vs. Today
- Connections and myths
- Logging

What is PaaS?


less control
SaaS
Gmail, Zoom, Netflix
PaaS
RDS, Azure Postgres, Cloud SQL
IaaS
VM in the cloud
more control

YOU OWN THE CAR

= on-premises solution

LEASED CAR

= IaaS

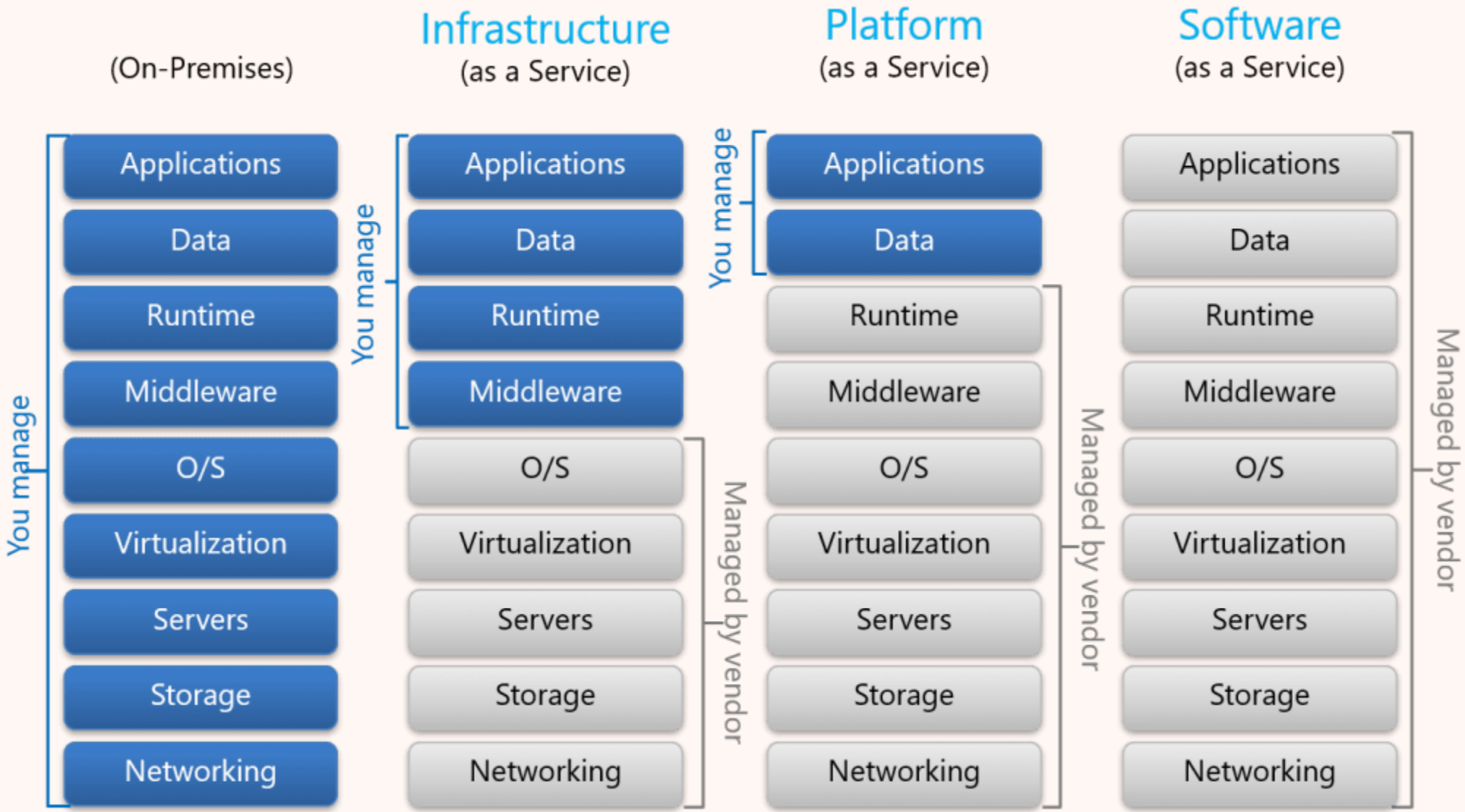
TAXI

= PaaS

BUS

= SaaS

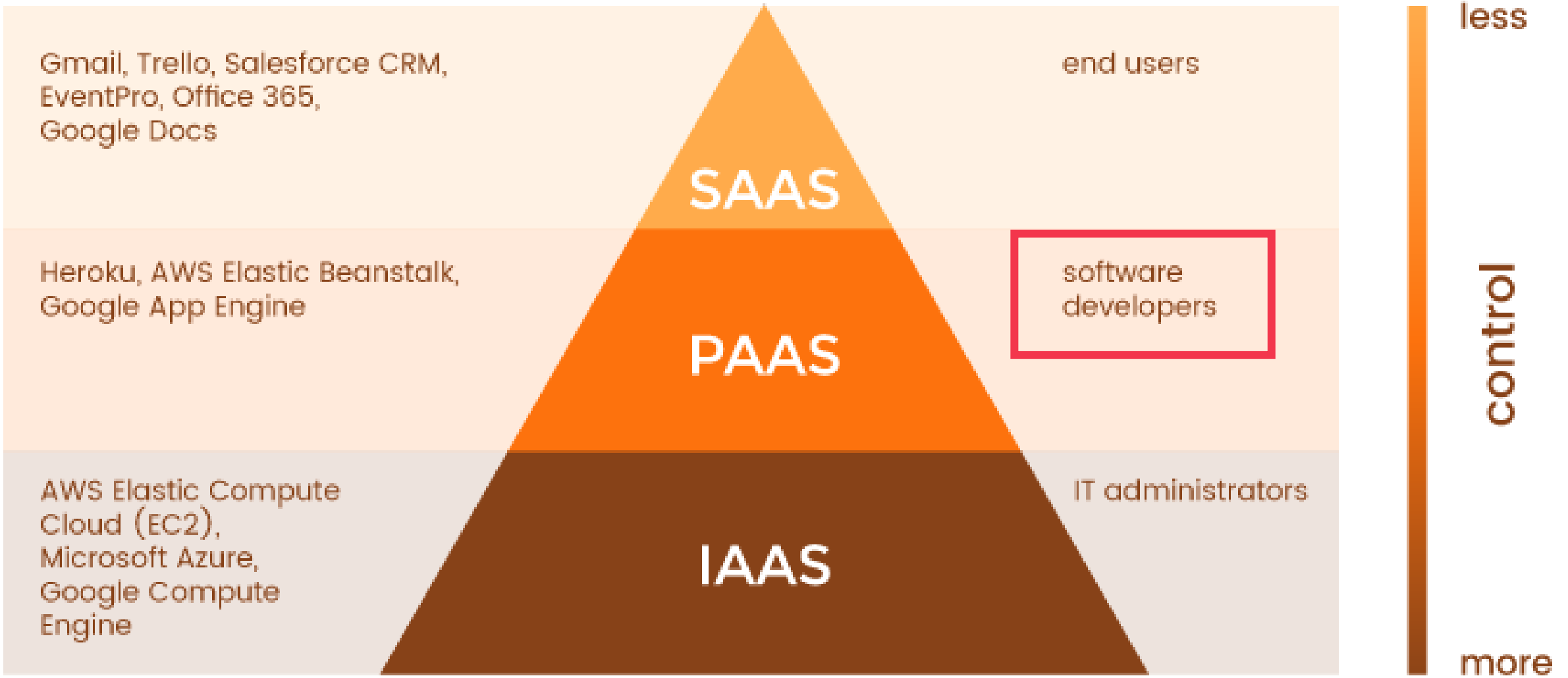
Source: Regpacks.com



Source: https://www.liquidweb.com/kb/saas-paas-iaas/

Postgres PaaS

Features/tasks	Managed by PaaS?
Minor upgrades	yes
Major upgrades	Partially, solution provided, testing and action on users
OS upgrades	yes
HA	yes
DR	Yes, some users actions needed
Parameter tuning	No, some sparse solutions might be provided
Autovacuum tuning	No, some sparse solutions might be provided
Query tuning	No, some tools might be provided
Support	Partially, limited to platform capabilities
Db maintenance (reindex, vacuum full, partitioning)	No, some tools might be provided
Data modeling	No



Source: https://rubygarage.org/blog/iaas-vs-paas-vs-saas

Postgres PaaS on major clouds

Service category	Service type	Google Cloud product	Google Cloud product description	AWS offering	Azure offering
Database	RDBMS	AlloyDB for PostgreSQL	Run transactional workloads 4x faster than standard PostgreSQL, and analytical queries up to 100x faster.	Amazon Aurora	Azure Cosmos DB for PostgreSQL, Azure SQL Database
Database	RDBMS	Cloud Spanner	Manage relational data with massive scale, strong consistency worldwide, and up to 99.999% availability.	Amazon Aurora	Azure SQL Database
Database	RDBMS	Cloud SQL	Manage relational data for MySQL, PostgreSQL, and SQL Server for workloads under 64 TB.	Amazon Relational Database Service (RDS), Amazon Aurora	Azure Database for MySQL and Azure Database for PostgreSQL

Source: https://cloud.google.com/docs/get-started/aws-azure-gcp-service-comparison

Geographies and latency

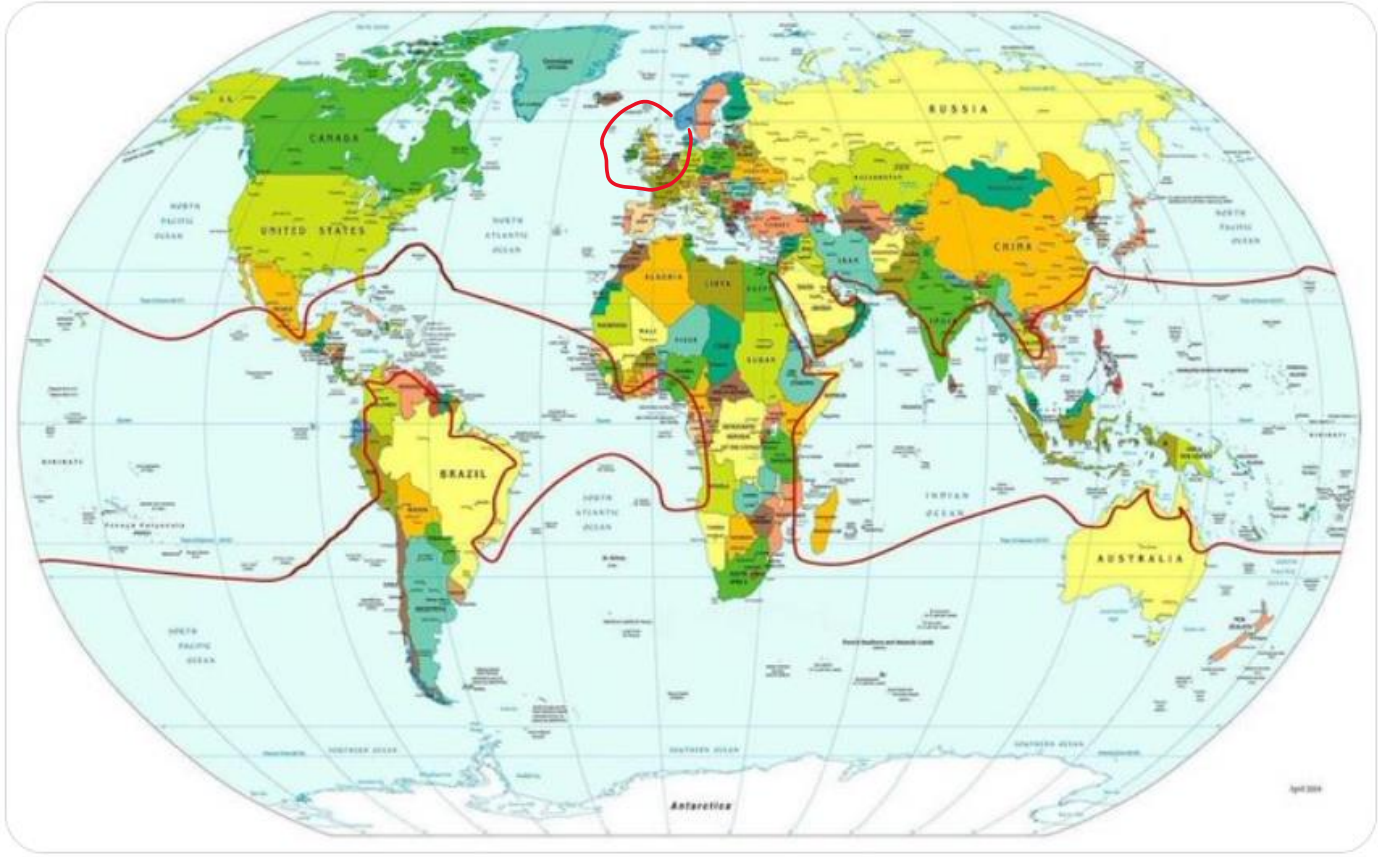
Chosing the regions

Patrick Pendergast @LPoPNH · 4 mar
Also in England around roughly the time of King Arthur, according to the Monty Python and the Holy Grail documentary.

Terrible Maps @TerribleMaps

Places where coconuts can grow

Przetłumacz wpis



10:30 PM · 4 mar 2024 · **471,7 tys.** Wyświetlenia

113 238 5 tys. 172

```
pgbench=> SELECT * FROM pg_stat_activity WHERE appl
ication_name='pgbench';
```

DB
South India

```
azureuser@UkSouthVM:~$ pgbench -c 100 -T120
```

Client
UK South

```
pgbench=> SELECT * FROM pg_stat_activity WHERE appl
ication_name='pgbench';
```

```
azureuser@UkSouthVM:~$ pgbench -c 100 -T120
```

```
datid            | 24798
datname          | pgbench
pid              | 26654
leader_pid       |
usesysid         | 24787
usename          | stefan
application_name | pgbench
client_addr      | 20.90.181.117
client_hostname  |
client_port      | 43558
backend_start    | 2024-03-12 12:56:17.754115+00
xact_start       | 2024-03-12 12:58:07.324484+00
query_start      | 2024-03-12 12:58:07.324484+00
state_change     | 2024-03-12 12:58:07.324742+00
wait_event_type  | Client
wait_event       | ClientRead
state            | idle in transaction
backend_xid      |
backend_xmin     |
query_id         | 6650430479887907887
query            | BEGIN;
backend_type     | client backend
```

UK South to South India

```
SELECT count(*), wait_event FROM
stat_activity GROUP BY wait_event ORDER BY
1 DESC;
```

```
count |  wait_event
-------+---------------
 25327 | ClientRead
   255 | transactionid
    38 |
    34 | WALSync
    17 | DataFileRead
     5 | WALWrite
     3 | tuple
```

```
SELECT count(*), state FROM stat_activity
GROUP BY state ORDER BY 1 DESC;
```

```
 count |        state
-------+---------------------
 19793 | idle in transaction
  5549 | idle
   337 | active
```

Actions

We recommend different actions depending on the causes of your wait event.

Topics

- Place the clients in the same Availability Zone and VPC subnet as the instance
- Scale your client
- Use current generation instances
- Increase network bandwidth
- Monitor maximums for network performance
- Monitor for transactions in the "idle in transaction" state

Place the clients in the same Availability Zone and VPC subnet as the instance

To reduce network latency and increase network throughput, place clients in the same Availability Zone and virtual private cloud (VPC) subnet as the RDS for PostgreSQL DB instance. Make sure that the clients are as geographically close to the DB instance as possible.

Source: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/wait-event.clientread.html

Availability Zones (AZs)

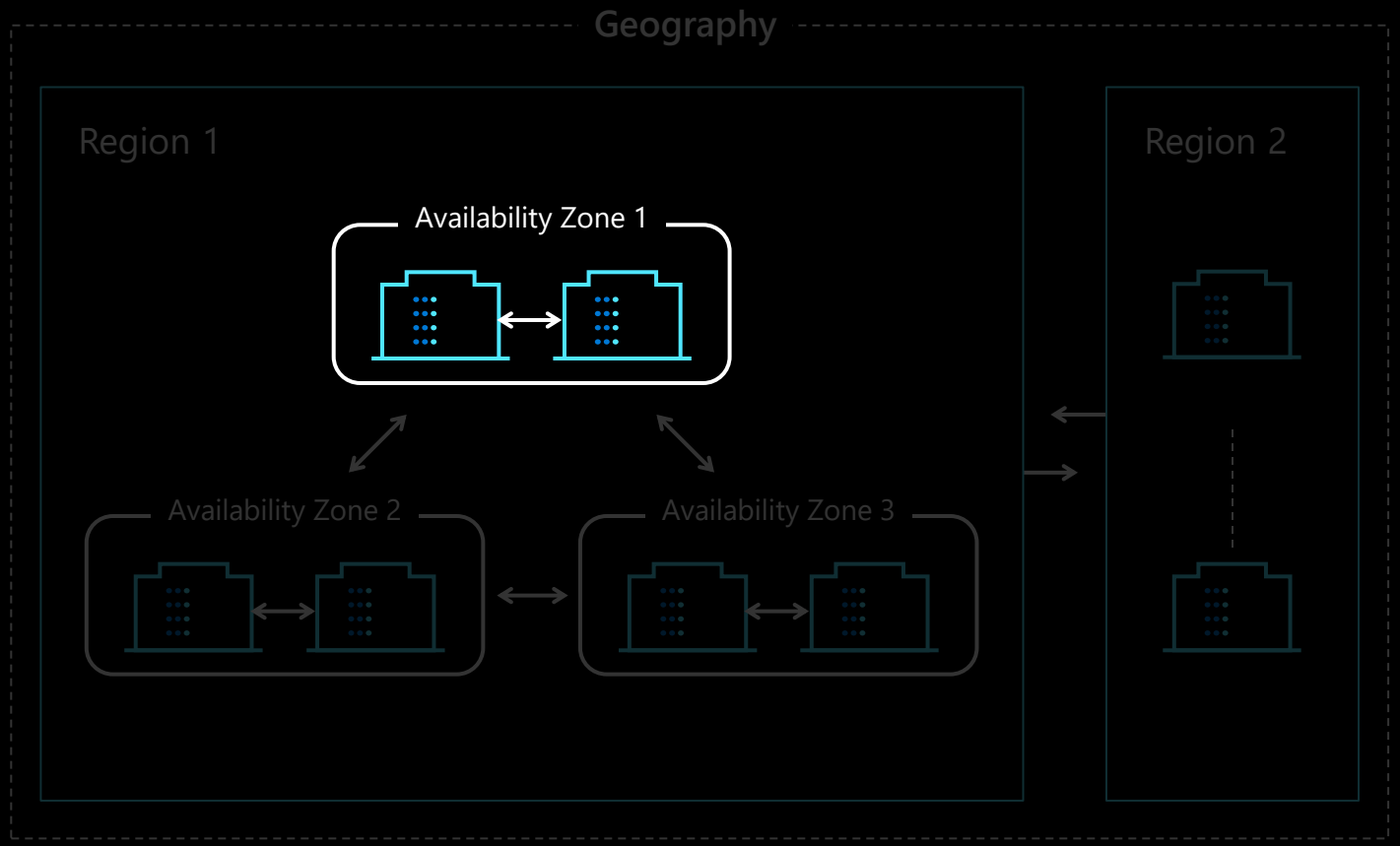
„An Availability Zone (AZ) is one or more discrete data centers with redundant:

- power,
- networking,
- and connectivity

in an AWS Region.

All AZs in an AWS Region are interconnected with high-bandwidth, low-latency networking, over fully redundant, dedicated metro fiber providing high-throughput, low-latency networking between AZs. All traffic between AZs is encrypted. The network performance is sufficient to accomplish synchronous replication between Azs."

Source: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/



```
datid            | 24968
datname          | pgbench
pid              | 17638
leader_pid       |
usesysid         | 24787
usename          | stefan
application_name | pgbench
client_addr      | 20.90.181.117
client_hostname  |
client_port      | 60952
backend_start    | 2024-03-12 18:55:36.582148+00
xact_start       | 2024-03-12 18:57:36.117534+00
query_start      | 2024-03-12 18:57:36.117534+00
state_change     | 2024-03-12 18:57:36.117548+00
wait_event_type  | Client
wait_event       | ClientRead
state            | idle in transaction
backend_xid      |
backend_xmin     |
query_id         | 6650430479887907887
query            | BEGIN;
backend_type     | client backend
```

Same AZ

```
SELECT count(*), wait_event FROM
stat_activity GROUP BY wait_event ORDER BY
1 DESC;
```

```
 count |      wait_event
-------+---------------------
 21913 | WALWrite
 18872 | ClientRead
  2796 | transactionid
  1298 | DataFileRead
  1274 |
   426 | WALSync
   184 | WALInsert
   125 | tuple
```

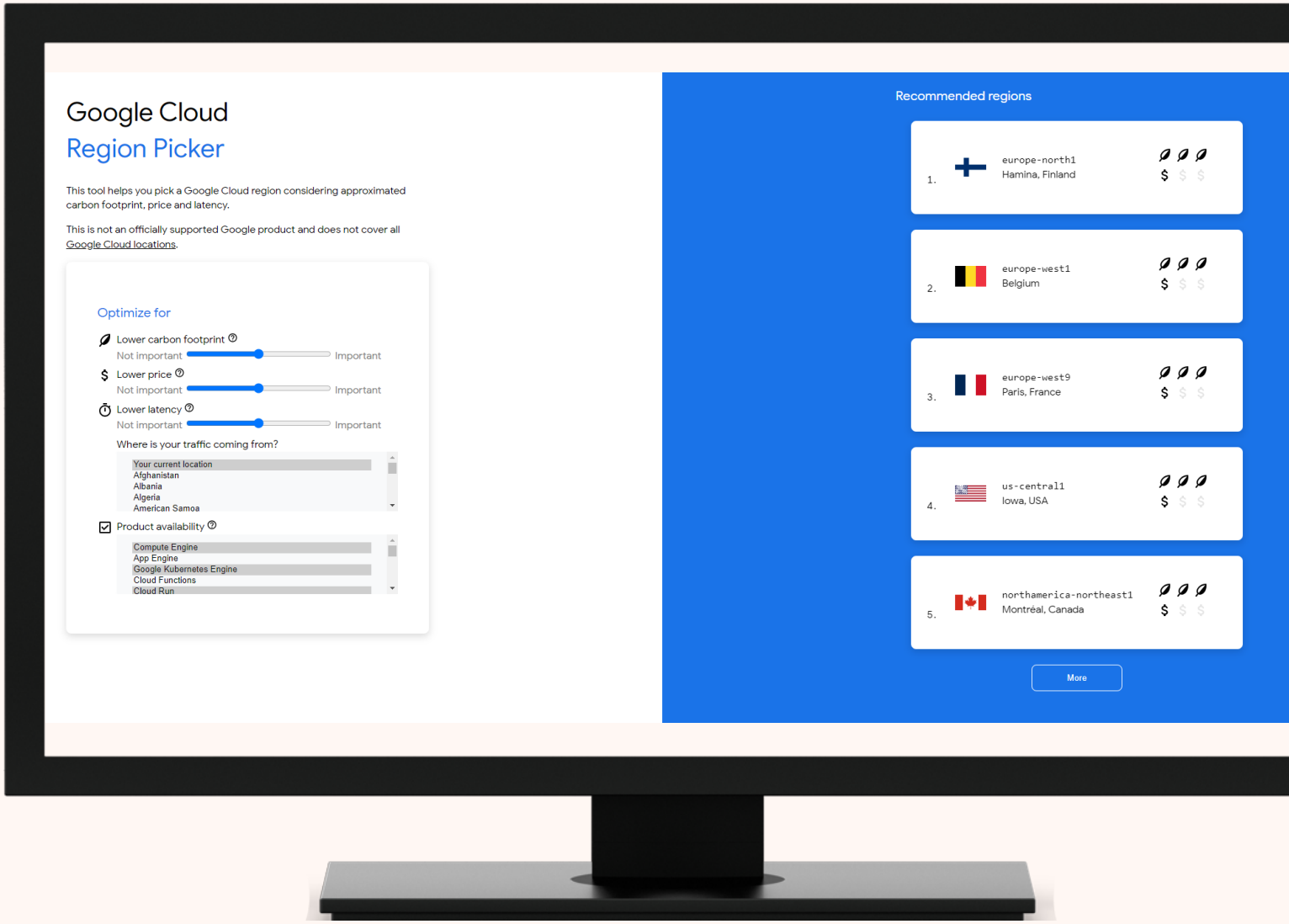
```
SELECT count(*), state FROM stat_activity
GROUP BY state ORDER BY 1 DESC;
```

```
 count |        state
-------+--------------------
 27795 | active
 18062 | idle in transaction
  1120 | idle
```

Pgbench results

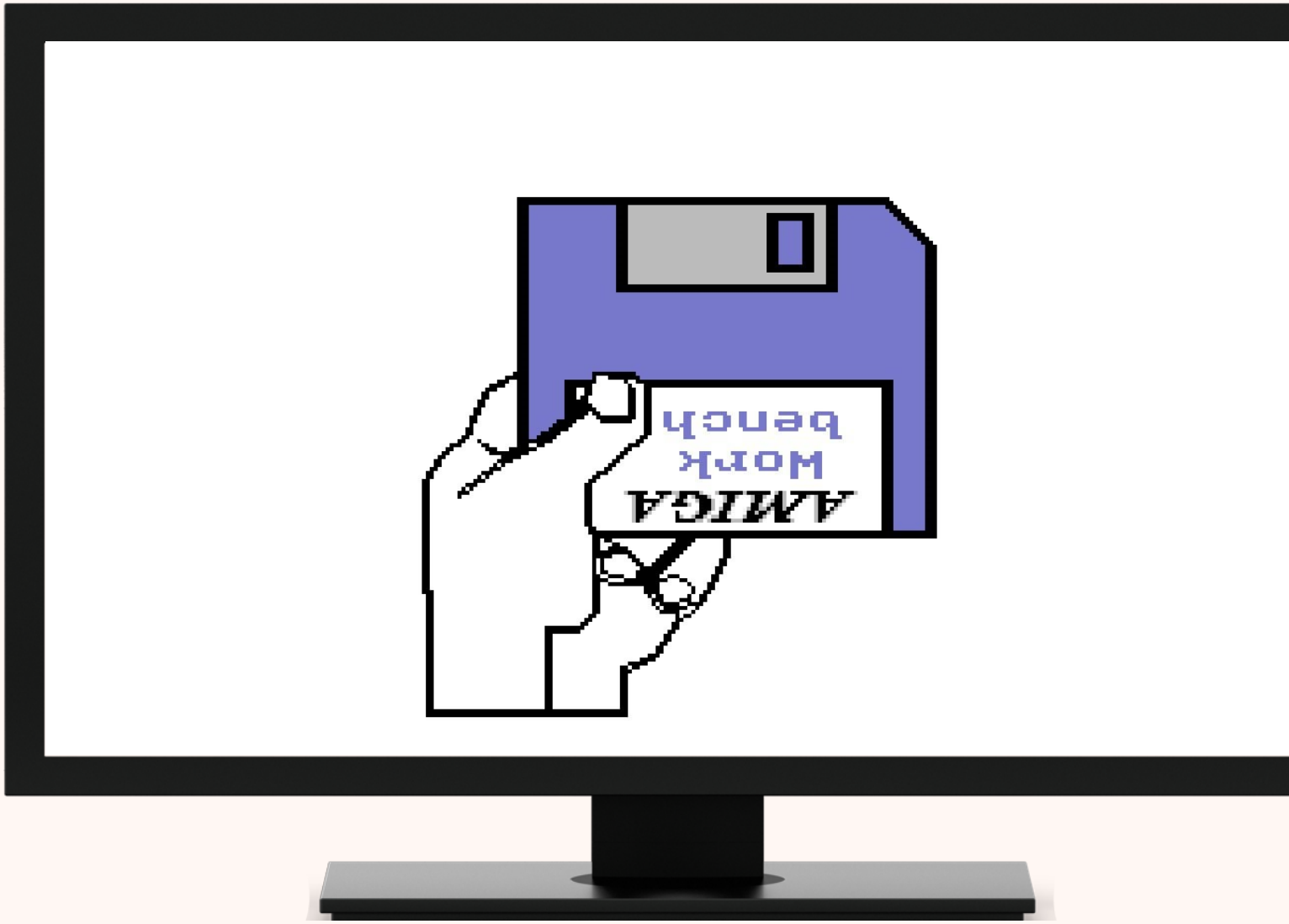
	UK South to South India	Same zone
number of transactions actually processed	3 213	497 525
latency average	1 044.792 ms	23.901 ms
tps	96	4 184

Google Cloud Region Picker



Evolving Technologies: Yesterday vs. Today

source: retrodata.se



Compute

Compute + storage

Compute

Compute resources are pre-allocated and billed per hour based on vCores configured.
Note that high availability is supported for only General purpose and Memory optimized tiers.

Compute tier

- ○ Burstable (1-20 vCores) - Best for workloads that don't need the full CPU continuously
- ○ General Purpose (2-96 vCores) - Balanced configuration for most common workloads
- ◉ Memory Optimized (2-96 vCores) - Best for workloads that require a high memory to CPU ratio

Compute Processor

- ○ AMD
- ◉ Intel

Compute size: Standard_E32ds_v4 (32 vCores, 256 GiB memory, 20000 max iops)

Standard_E48ds_v5 (48 vCores, 384 GiB memory, 20000 max iops)
Standard_E64ds_v5 (64 vCores, 512 GiB memory, 20000 max iops)
Standard_E96ds_v5 (96 vCores, 672 GiB memory, 20000 max iops)
Standard_E2ds_v4 (2 vCores, 16 GiB memory, 3200 max iops)
Standard_E4ds_v4 (4 vCores, 32 GiB memory, 6400 max iops)
Standard_E8ds_v4 (8 vCores, 64 GiB memory, 12800 max iops)
Standard_E16ds_v4 (16 vCores, 128 GiB memory, 20000 max iops)
Standard_E20ds_v4 (20 vCores, 160 GiB memory, 20000 max iops)
Standard_E32ds_v4 (32 vCores, 256 GiB memory, 20000 max iops)
Standard_E48ds_v4 (48 vCores, 384 GiB memory, 20000 max iops)
Standard_E64ds_v4 (64 vCores, 432 GiB memory, 20000 max iops)
Standard_E2s_v3 (2 vCores, 16 GiB memory, 3200 max iops)
Standard_E4s_v3 (4 vCores, 32 GiB memory, 6400 max iops)
Standard_E8s_v3 (8 vCores, 64 GiB memory, 12800 max iops)

Storage

The storage you provision is the amount of
Note that storage cannot be scaled down

Storage type ⓘ

Storage size ⓘ

Performance Tier (preview) ⓘ

Storage Auto-growth ⓘ

High availability

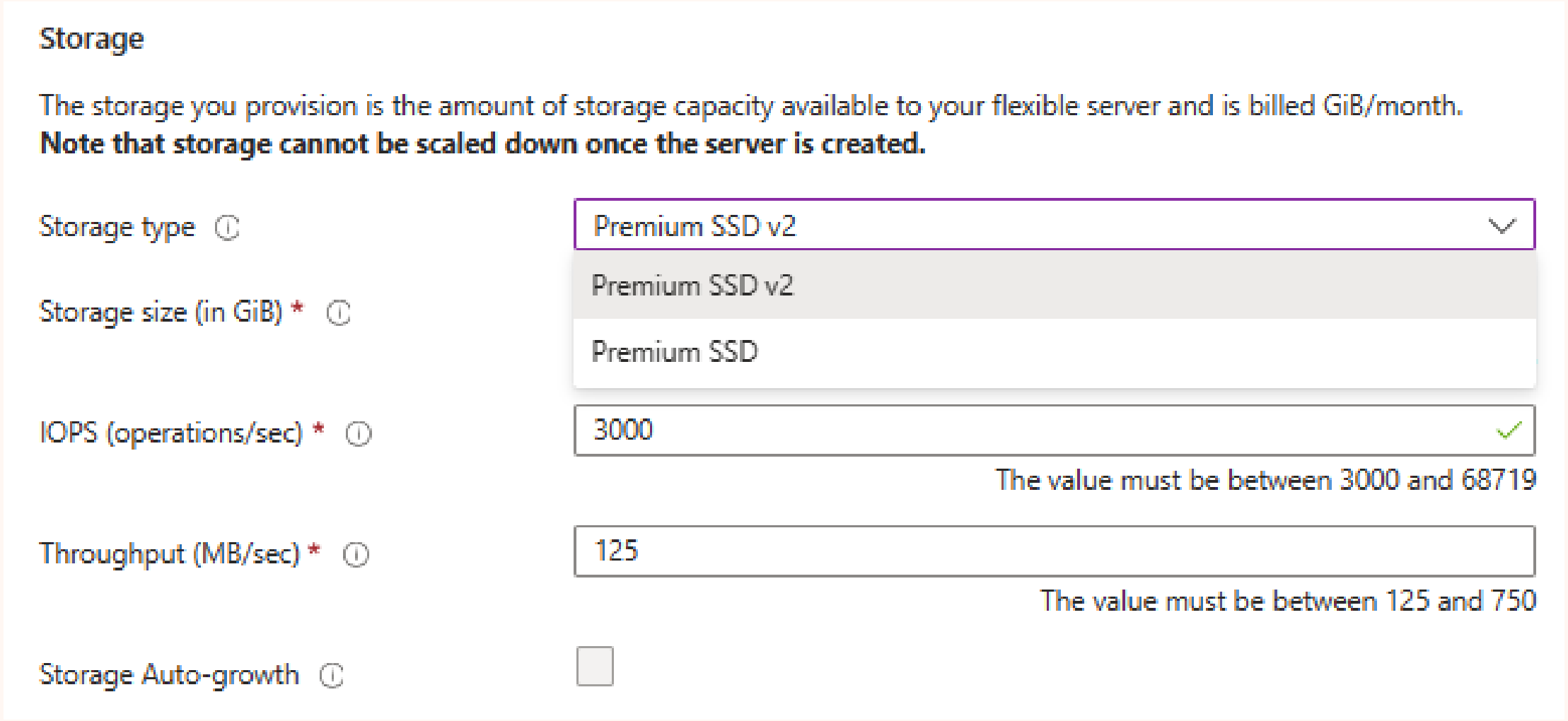
Zone redundant high availability deploys a

Enable high availability ⓘ

Backups

Save

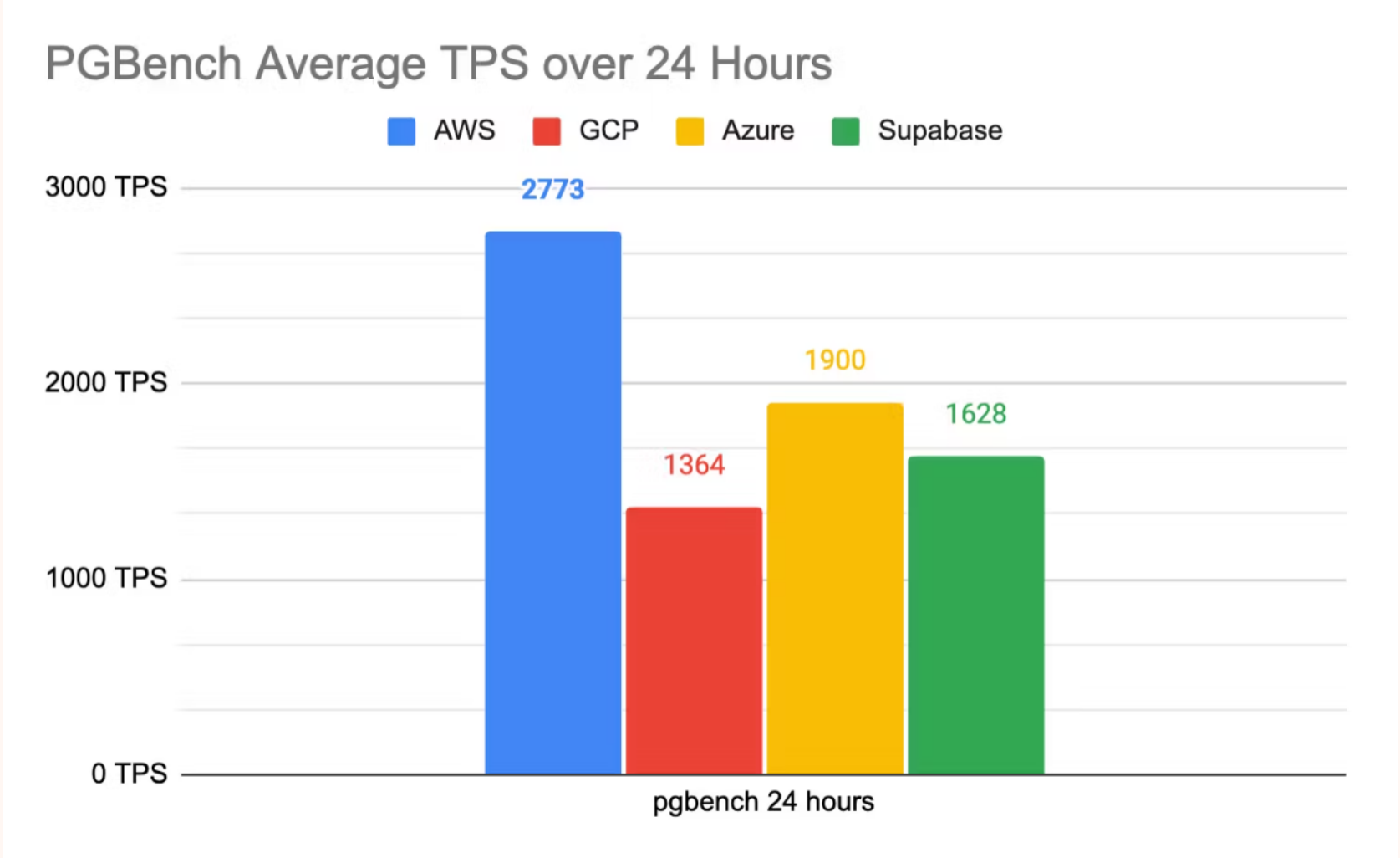
Storage


Storage
The storage you provision is the amount of storage capacity available to your flexible server and is billed GiB/month.
Note that storage cannot be scaled down once the server is created.
Storage type
Premium SSD v2
Premium SSD v2
Premium SSD
Storage size (in GiB) *
IOPS (operations/sec) *
3000
The value must be between 3000 and 68719
Throughput (MB/sec) *
125
The value must be between 125 and 750
Storage Auto-growth

Performance

	16	15	14	13	12
Abbreviated Keys	Yes	Yes	Yes	Yes	Yes
Asynchronous Commit	Yes	Yes	Yes	Yes	Yes
Automatic plan invalidation	Yes	Yes	Yes	Yes	Yes
Background Checkpointer	Yes	Yes	Yes	Yes	Yes
Background Writer	Yes	Yes	Yes	Yes	Yes
Base backup throttling	Yes	Yes	Yes	Yes	Yes
CREATE STATISTICS - most-common values (MCV) statistics	Yes	Yes	Yes	Yes	Yes
CREATE STATISTICS - multicolumn	Yes	Yes	Yes	Yes	Yes
CREATE STATISTICS - "OR" and "IN/ANY" statistics	Yes	Yes	Yes	Yes	No
Cross datatype hashing support	Yes	Yes	Yes	Yes	Yes
Distributed checkpointing	Yes	Yes	Yes	Yes	Yes
Foreign keys marked as NOT VALID	Yes	Yes	Yes	Yes	Yes
Frozen page map	Yes	Yes	Yes	Yes	Yes
Full Text Search	Yes	Yes	Yes	Yes	Yes
Hash aggregation can use disk	Yes	Yes	Yes	Yes	No
Hashing support for DISTINCT/UNION/INTERSECT/EXCEPT	Yes	Yes	Yes	Yes	Yes
Hashing support for FULL OUTER JOIN, LEFT OUTER JOIN and RIGHT OUTER JOIN	Yes	Yes	Yes	Yes	Yes
Heap Only Tuples (HOT)	Yes	Yes	Yes	Yes	Yes
Improved performance for sorts exceeding working memory	Yes	Yes	No	No	No
Improved window function performance	Yes	Yes	No	No	No
Incremental sort	Yes	Yes	Yes	Yes	No
Incremental sort for SELECT DISTINCT	Yes	No	No	No	No
Incremental sort for window functions	Yes	Yes	Yes	No	No

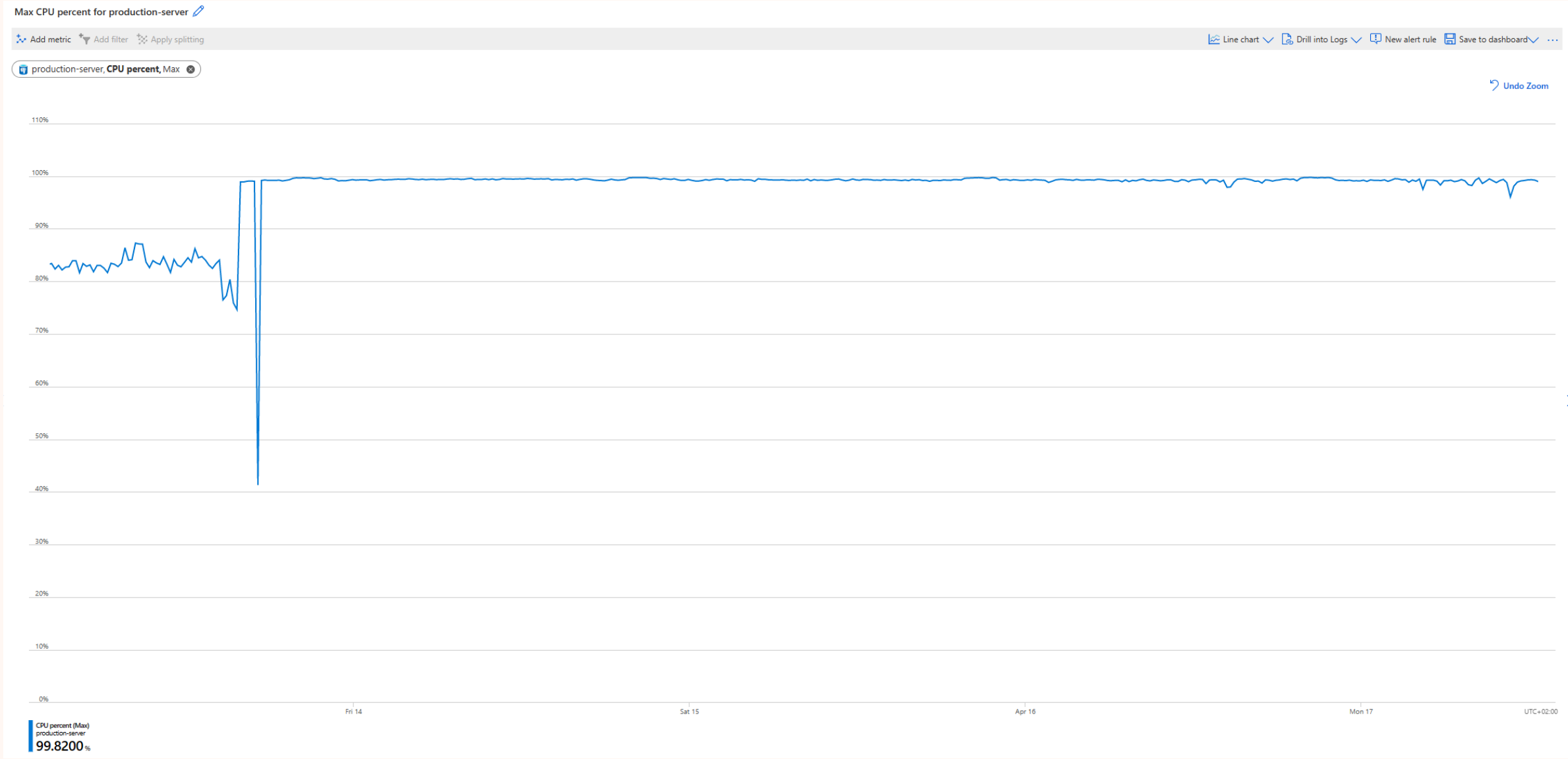
And benchmarking...



Source:https://blog.peerdb.io/comparing-postgres-managed-services-aws-azure-gcp-and-supabase

Connections

CPU usage


Max CPU percent for production-server
Add metric
Add filter
Apply splitting
Line chart
Drill into Logs
New alert rule
Save to dashboard
production-server, CPU percent, Max
Undo Zoom
110%
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Fri 14
Sat 15
Apr 16
Mon 17
UTC+02:00
CPU percent (Max)
production-server
99.8200 %

Number Of Database Connections

You can often support more concurrent users by reducing the number of database connections and using some form of connection pooling. This page attempts to explain why that is.

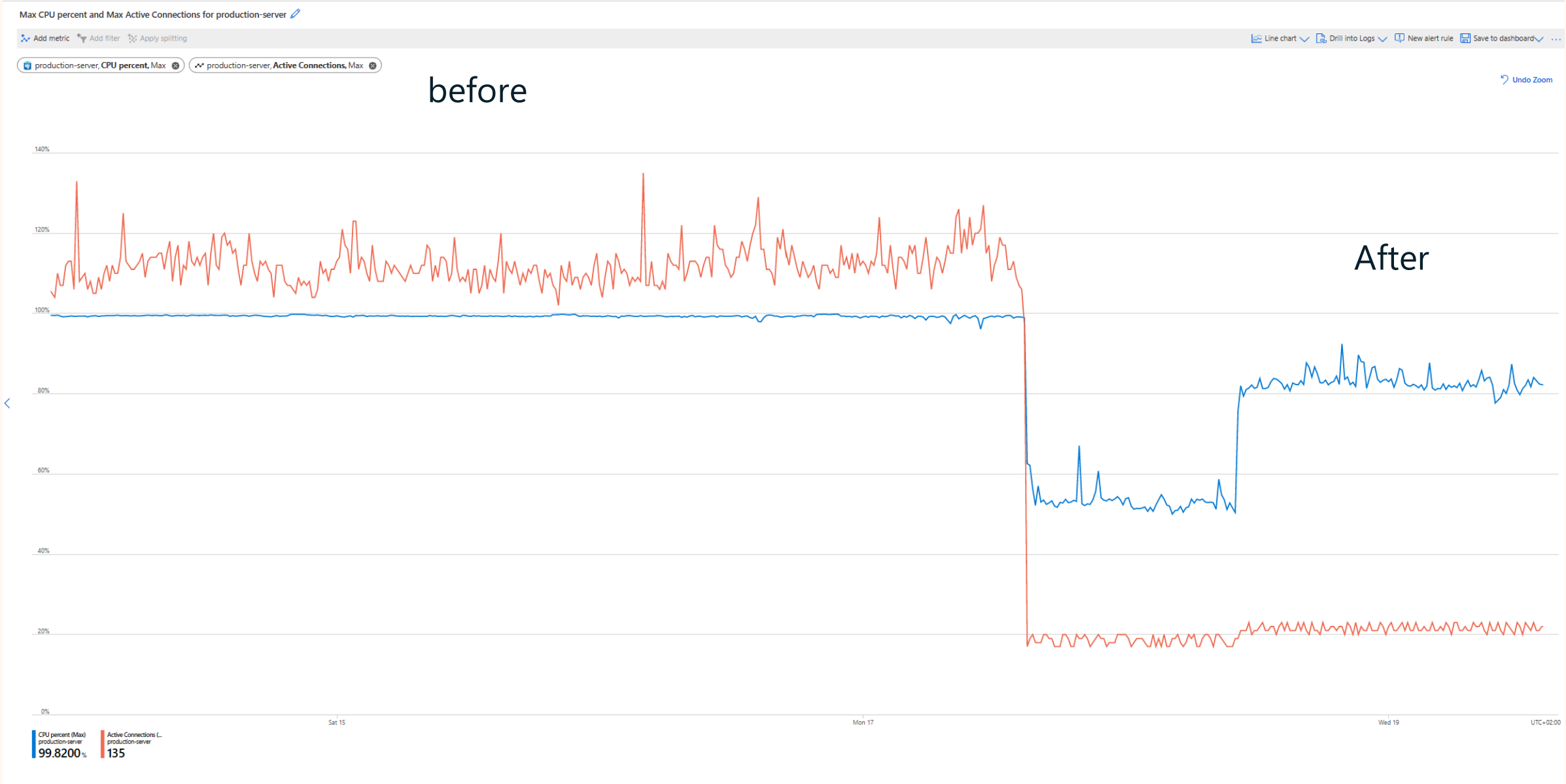
Contents [hide]

Summary

A database server only has so many resources, and if you don't have enough connections active to use all of them, your throughput will generally improve by using more connections. Once all of the resources are in use, you won't push any more work through by having more connections competing for the resources. In fact, throughput starts to fall off due to the overhead from that contention. You can generally improve both latency and throughput by limiting the number of database connections with active transactions to match the available number of resources, and queuing any requests to start a new database transaction which come in while at the limit.

Contrary to many people's initial intuitive impulses, you will often see a transaction *reach completion sooner* if you queue it when it is ready but the system is busy enough to have saturated resources and *start it later* when resources become available.

Pg will usually complete the same 10,000 transactions faster by doing them 5, 10 or 20 at a time than by doing them 500 at a time. Determining exactly how many should be done at once varies by workload and requires tuning.


Max CPU percent and Max Active Connections for production-server
Add metric
Add filter
Apply splitting
Line chart
Drill into Logs
New alert rule
Save to dashboard
production-server, CPU percent, Max
production-server, Active Connections, Max
before
Undo Zoom
140%
120%
100%
80%
60%
40%
20%
0%
After
Sat 15
Mon 17
Wed 19
UTC+02:00
CPU percent (Max)
production-server
99.8200 %
Active Connections (...
production-server
135

Logging

Logging

- Yes, you still need it
- You really do!
- On all major clouds you have access to postgres logs, the way to obtain them might differ.

What to log?

- log_duration - Causes the duration of every completed statement to be logged.
- log_min_duration_statement - Causes the duration of each completed statement to be logged if the statement ran for at least the specified amount of time.
- log_statement - Controls which SQL statements are logged. Valid values are none (off), ddl, mod, and all (all statements).

Log_duration

2024-03-28 21:29:15.462 CET [116948] postgres@anon LOG: duration: 10.376 ms
2024-03-28 21:29:15.463 CET [116954] postgres@anon LOG: duration: 2.334 ms
2024-03-28 21:29:15.466 CET [116947] postgres@anon LOG: duration: 9.485 ms
2024-03-28 21:29:15.468 CET [116944] postgres@anon LOG: duration: 5.102 ms
2024-03-28 21:29:15.471 CET [116957] postgres@anon LOG: duration: 2.448 ms
2024-03-28 21:29:15.473 CET [116951] postgres@anon LOG: duration: 3.739 ms
2024-03-28 21:29:15.475 CET [116955] postgres@anon LOG: duration: 4.327 ms
2024-03-28 21:29:15.476 CET [116950] postgres@anon LOG: duration: 2.253 ms
2024-03-28 21:29:15.476 CET [116956] postgres@anon LOG: duration: 3.043 ms

log_min_duration_statement

2024-03-28 21:32:53.385 CET [117987] postgres@anon LOG: duration: 5.603 ms statement: SELECT increment(1)

2024-03-28 21:32:53.388 CET [117990] postgres@anon LOG: duration: 3.153 ms statement: SELECT increment(1)

2024-03-28 21:32:53.391 CET [117986] postgres@anon LOG: duration: 5.373 ms statement: SELECT increment(1)

2024-03-28 21:32:53.394 CET [117993] postgres@anon LOG: duration: 1.522 ms statement: SELECT increment(1)

2024-03-28 21:32:53.394 CET [117991] postgres@anon LOG: duration: 2.092 ms statement: SELECT increment(1)

2024-03-28 21:32:53.396 CET [117994] postgres@anon LOG: duration: 2.091 ms statement: SELECT increment(1)

Log_statement

2024-03-28 21:35:38.053 CET [118823] postgres@anon LOG: statement: SELECT increment(1)
2024-03-28 21:35:38.054 CET [118826] postgres@anon LOG: statement: SELECT increment(1)
2024-03-28 21:35:38.054 CET [118817] postgres@anon LOG: statement: SELECT increment(1)
2024-03-28 21:35:38.062 CET [118828] postgres@anon LOG: statement: SELECT increment(1)
2024-03-28 21:35:38.063 CET [118825] postgres@anon LOG: statement: SELECT increment(1)

And much more

- pgAudit
- pg_stat_statement
- pgBadger
- …

For troubleshooting

log_line_prefix = '%t [%p]: [%l-1] db=%d,user=%u,app=%a,client=%h '

log_connections = on

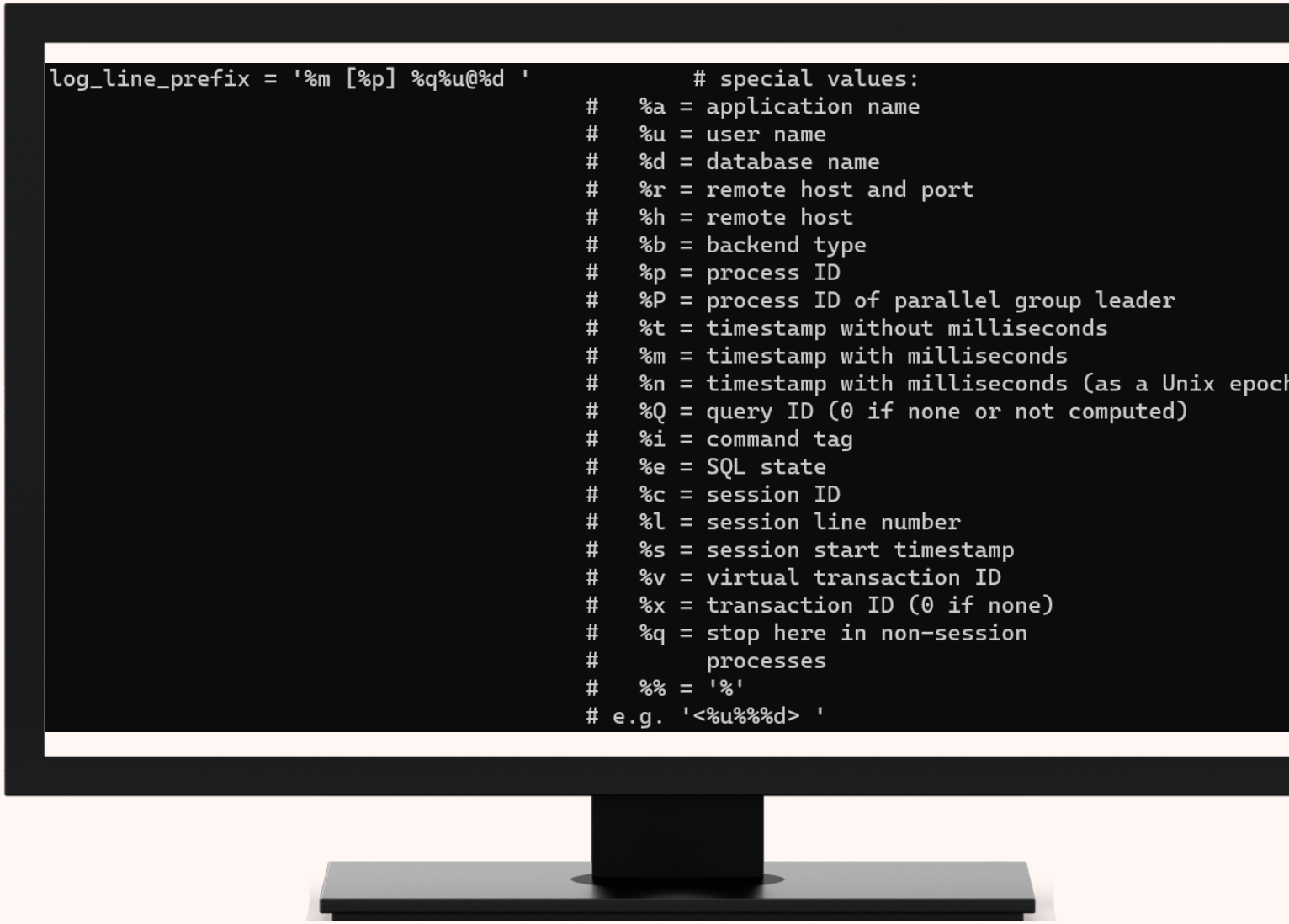
log_disconnections = on

log_lock_waits = on

log_temp_files = 0

log_autovacuum_min_duration = 0

log_line_prefix



References

- PaaS explained
- Azure Regions
- Compare AWS and Azure services to Google Cloud
- AWS RDS wait events
- https://www.postgresql.org/about/featurematrix/#performance
- https://wiki.postgresql.org/wiki/Number_Of_Database_Connections
- Comparing Postgres Managed Services: AWS, Azure, GCP and Supabase

Save the date
June 11-13, 2024

POSETTE:
An Event for Postgres

2024 (formerly Citus Con)

A free & virtual developer event

Save the Date → aka.ms/posette-cal

Microsoft

Got 3 minutes?
We'd love your input
on some of our Postgres work

Get your FREE socks
@ Microsoft booth

Thank you