Mattermost

Running Mattermost on YugabyteDB

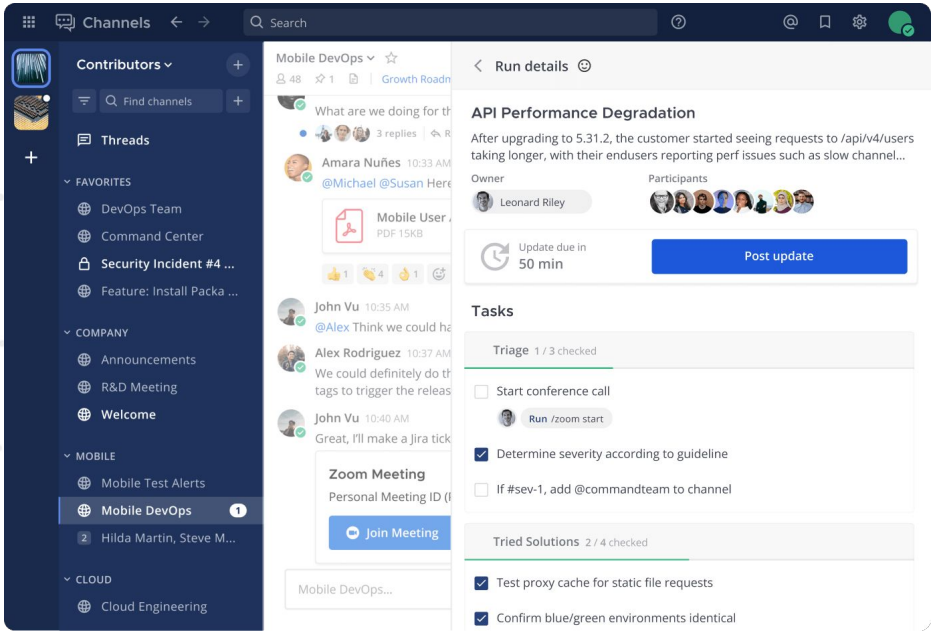
Jesús Espino, Software Engineer @ Mattermost

01

What is Mattermost?

Stats and good practices

Mattermost



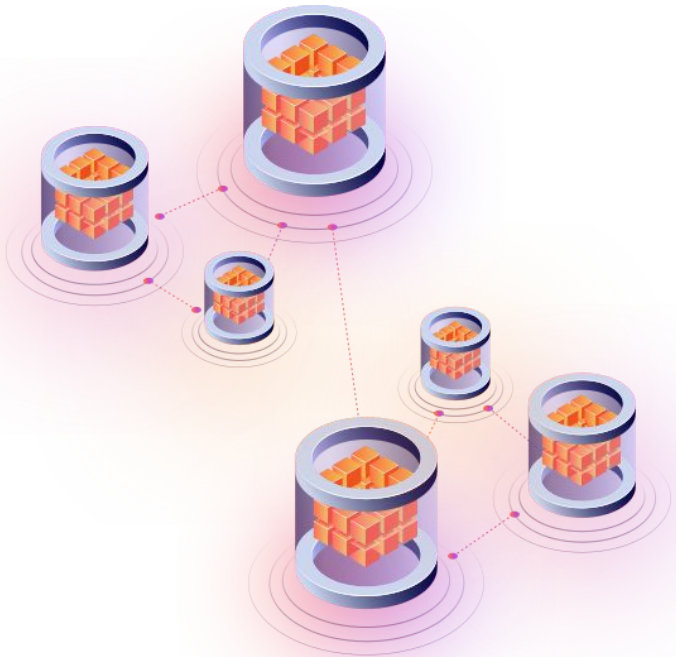
- Communication platform
- Open Source
- On-prem or Cloud
- Design for mission critical systems
- Focus on stability and security

02

Why a distributed database?

The mattermost use case

Why a distributed database?



- Align with Mattermost focus
- Potential for high scalability
- Geo partitioning

Why YugabyteDB?

- Open Source
- Highly compatible with Postgres
- Good metrics
- Good administration
- Cloud hosting service

03

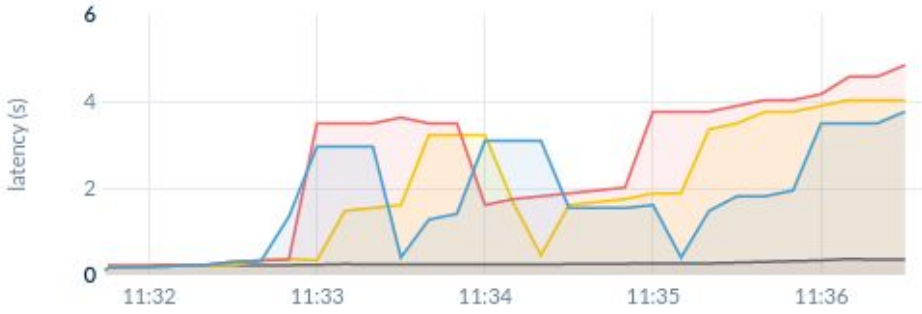
My first try

Another DB 3 years ago

Changes needed

```
.gitignore                                     |  1 +
Makefile                                       |  3 +++
app/export_test.go                             | 12 ++++++------
app/server.go                                  |  8 ++++++++
build/docker-compose-generator/main.go         |  1 +
build/docker-compose.common.yml                | 11 +++++++++++
build/docker-compose.yml                       |  5 +++++
docker-compose.makefile.yml                    |  8 ++++++++
docker-compose.yaml                            |  8 ++++++++
einterfaces/[redacted]                         | 12 ++++++++++++
model/config.go                                |  9 +++++----
model/license.go                               |  6 ++++++
store/searchtest/channel_layer.go              |  3 +++
store/searchtest/post_layer.go                 |  3 +++
store/searchtest/user_layer.go                 |  3 +++
store/sqlstore/channel_store.go                | 73 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++--------------
store/sqlstore/channel_store_categories.go     | 39 ++++++++++++++++++++++++++++++++++++++-
store/sqlstore/channel_store_test.go           |  3 +++
store/sqlstore/group_store.go                  |  2 +-
store/sqlstore/imports/placeholder.go          |  4 ++++
store/sqlstore/link_metadata_store.go          |  2 +-
store/sqlstore/oauth_store.go                  |  2 ++
store/sqlstore/plugin_store.go                 |  4 ++--
store/sqlstore/post_store.go                   | 22 ++++++++++++++++++----
store/sqlstore/preference_store.go             |  2 +-
store/sqlstore/reaction_store.go               |  2 +-
store/sqlstore/store.go                        |  3 +++
store/sqlstore/store_test.go                   | 10 ++++++++++
store/sqlstore/team_store.go                   | 20 +++++++++++++++-----
store/sqlstore/upgrade.go                      |  4 ++--
store/sqlstore/user_access_token_store.go      |  6 ++++++
store/sqlstore/user_store.go                   |  2 +-
store/store.go                                 |  5 +++++
store/storetest/channel_store.go               | 15 +++++++++++++++
store/storetest/mocks/Store.go                 | 14 ++++++++++++++
store/storetest/settings.go                    | 55 +++++++++++++++++++++++++++++++++++++++++++++++++++++++
store/storetest/store.go                       |  4 ++++
vendor/github.com/Masterminds/squirrel/go.mod  |  2 ++
vendor/gopkg.in/yaml.v2/go.mod                 |  8 ++++----
39 files changed, 349 insertions(+), 47 deletions(-)
```

Results



P99 latency	4563.4 ms
Queries per second	693.3

- Database collapsing in less than 2000 users
- A lot of back and forth with support
- A lot of specific SQL queries
- p99 performance

Lessons learnt

- Distributed databases are hard
- It is not a drop-in replacement
- You need to design your queries and indexes for your distributed database

04

Running on YugabyteDB

Changes needed

```
server/channels/db/migrations/postgres/000066_upgrade_posts_v6.0.up.sql |  5 +++--
server/channels/db/migrations/postgres/000111_update_vacuuming.down.sql | 10 ++++++++++
server/channels/db/migrations/postgres/000111_update_vacuuming.up.sql   | 11 ++++++++++
3 files changed, 24 insertions(+), 2 deletions(-)
```

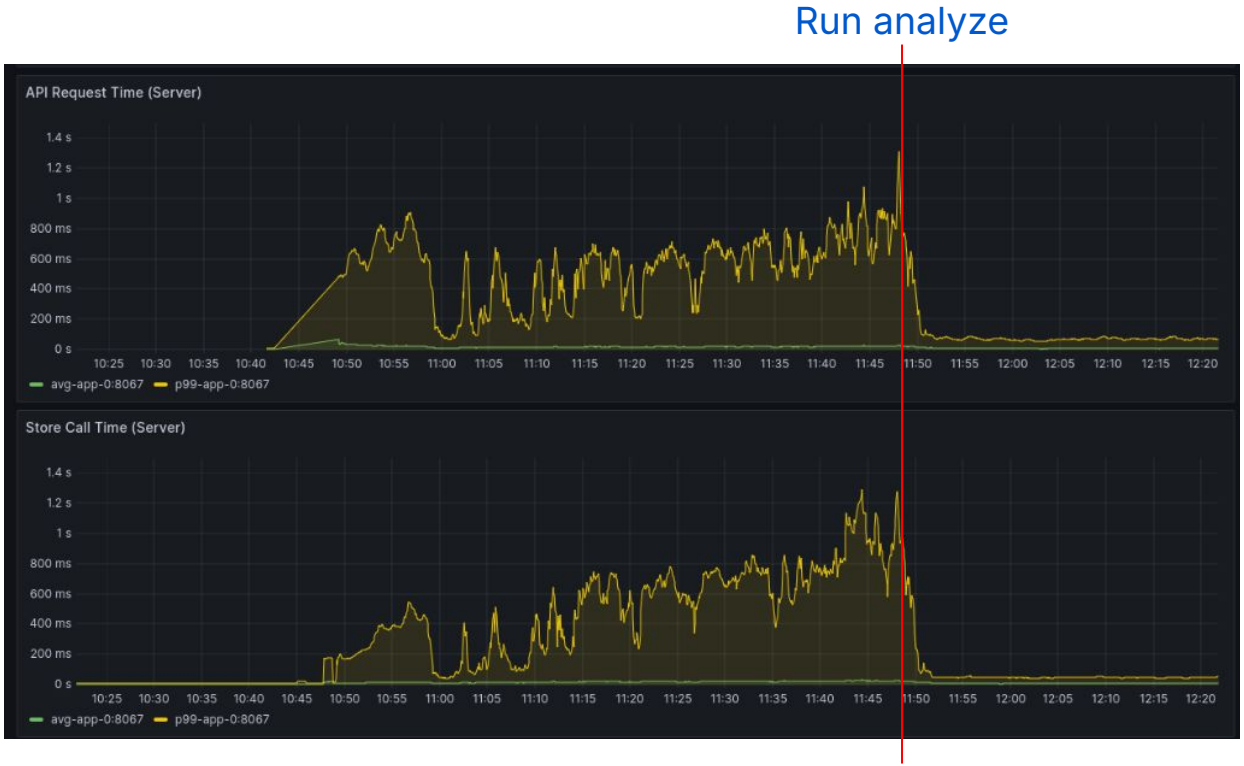
Results first try

- Database collapsing at 2000 users
- Performance degrading over time
- Both avg and p99 growing over time
- Weird query plans applied

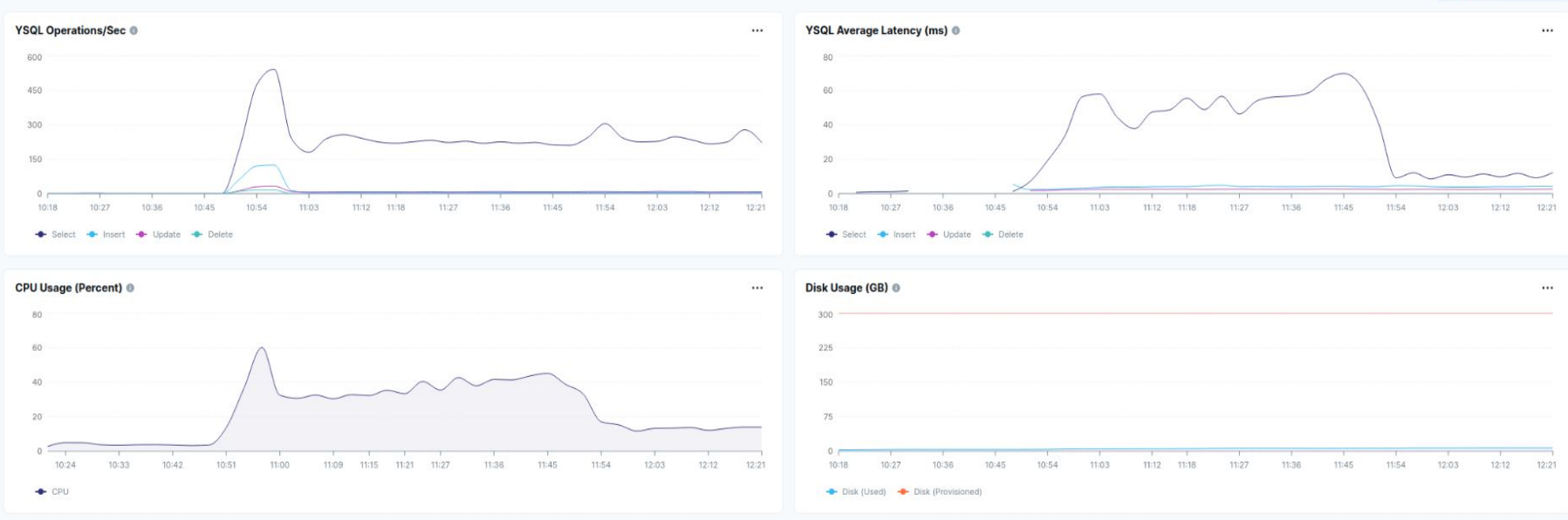
Solution to my problems

- Run “Analyze”

After running analyze



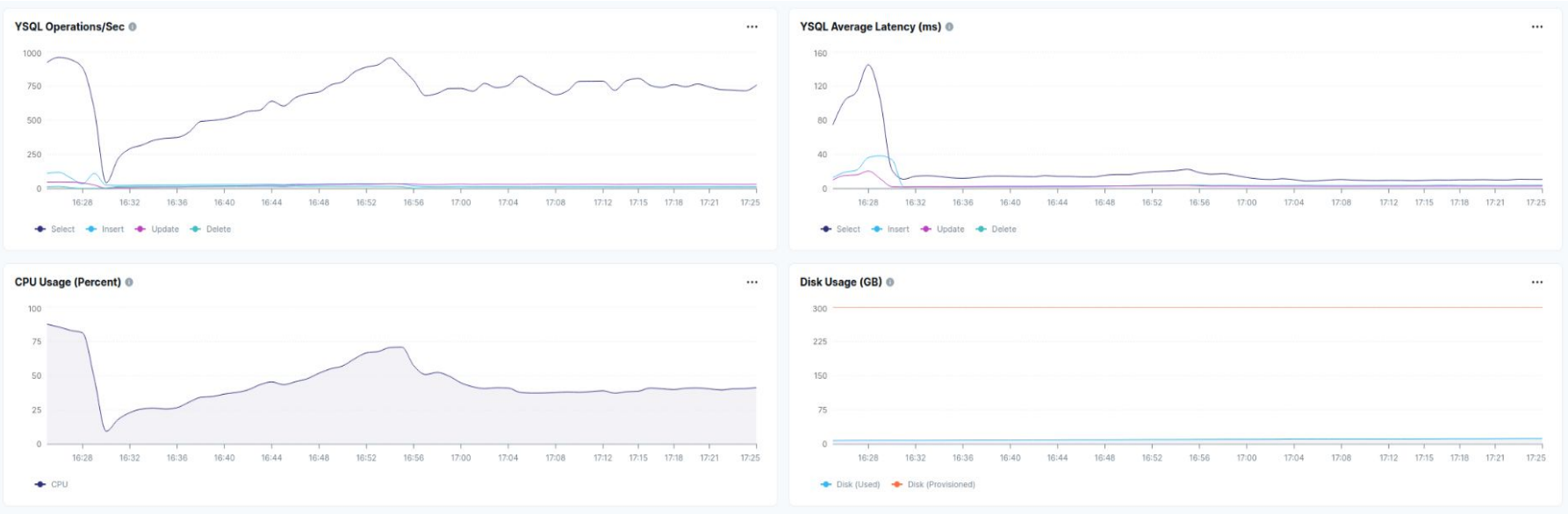
Results first try



Results second try

- Database working at 6000 users
- Performance stable over time

Results first try



Lessons learnt

- It can be drop-in replacement
- It worked great for us without changes
- YugabyteDB requires Analyze to make smart decisions about your query plans

05

Load tests by YugabyteDB

MatterMost open-source load test gives equivalent cost performance

YugabyteDB Aeon - List Pricing ~$1500

- 12 CPU (3 nodes x 4 CPU) with Load Balancer
- With ElasticSearch
- Maxconns, Idleconns=24
- For peak of **17800** users

Store Call Time (on YugabyteDB)		
Users	**Average**	**P99**
8900	< 5ms	< 25ms
13200	< 8ms	< 50ms
16100	< 12ms	< 100ms

AWS Aurora - List Pricing ~$1350

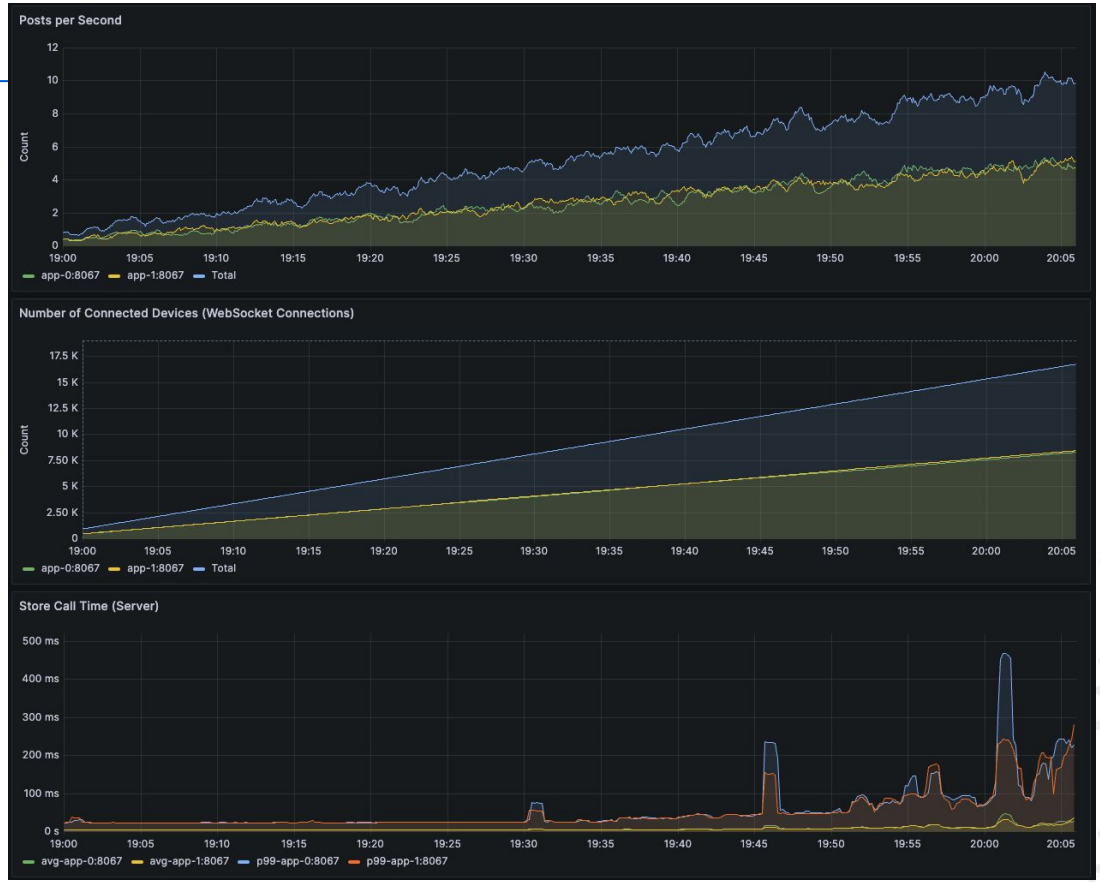
- 2 x db.r7g.xlarge (8vCPU) I/O optimized
- With ElasticSearch
- With storage, insights and backups
- For peak of **17100** users

Done by the Yugabyte team:

- Zoe Chan
- Mark Peacock

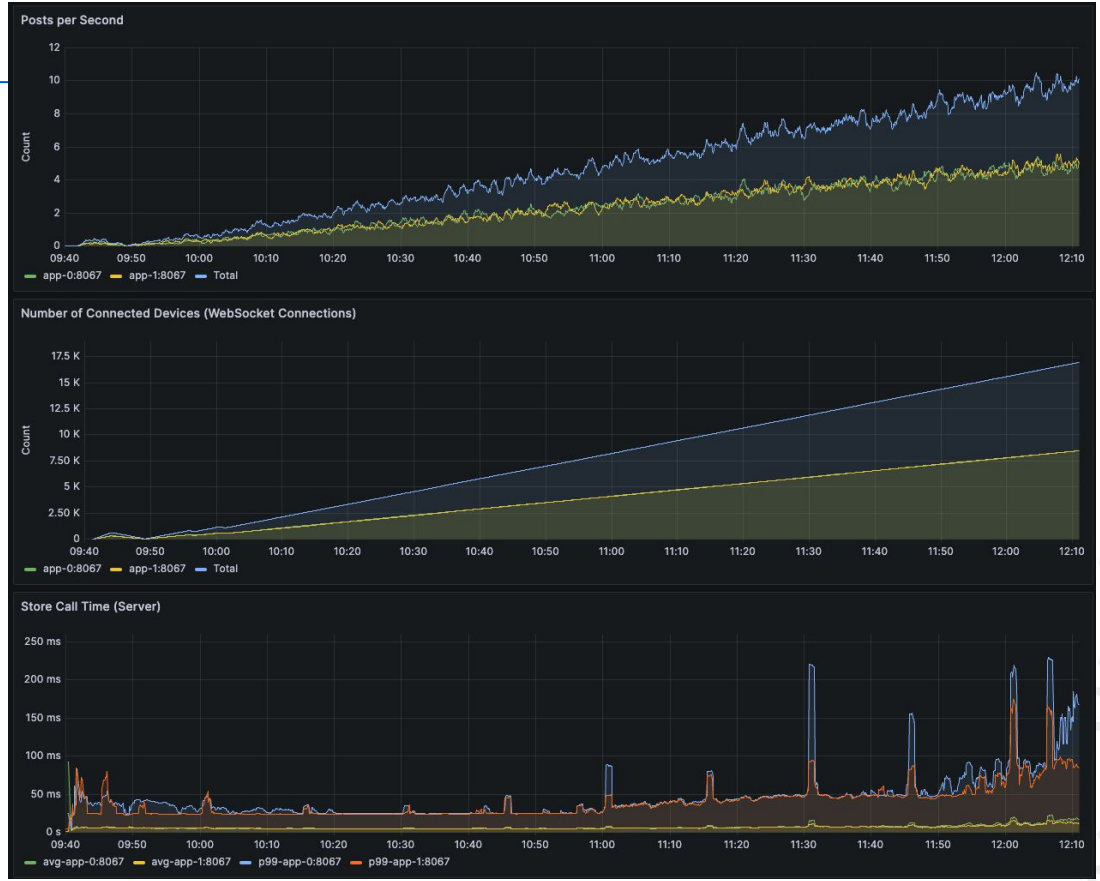
Test Results - 1

Users	Average	P99
8900	< 5ms	< 25ms
13200	< 8ms	< 50ms
16100	< 12ms	< 100ms
17800	Supported Users	



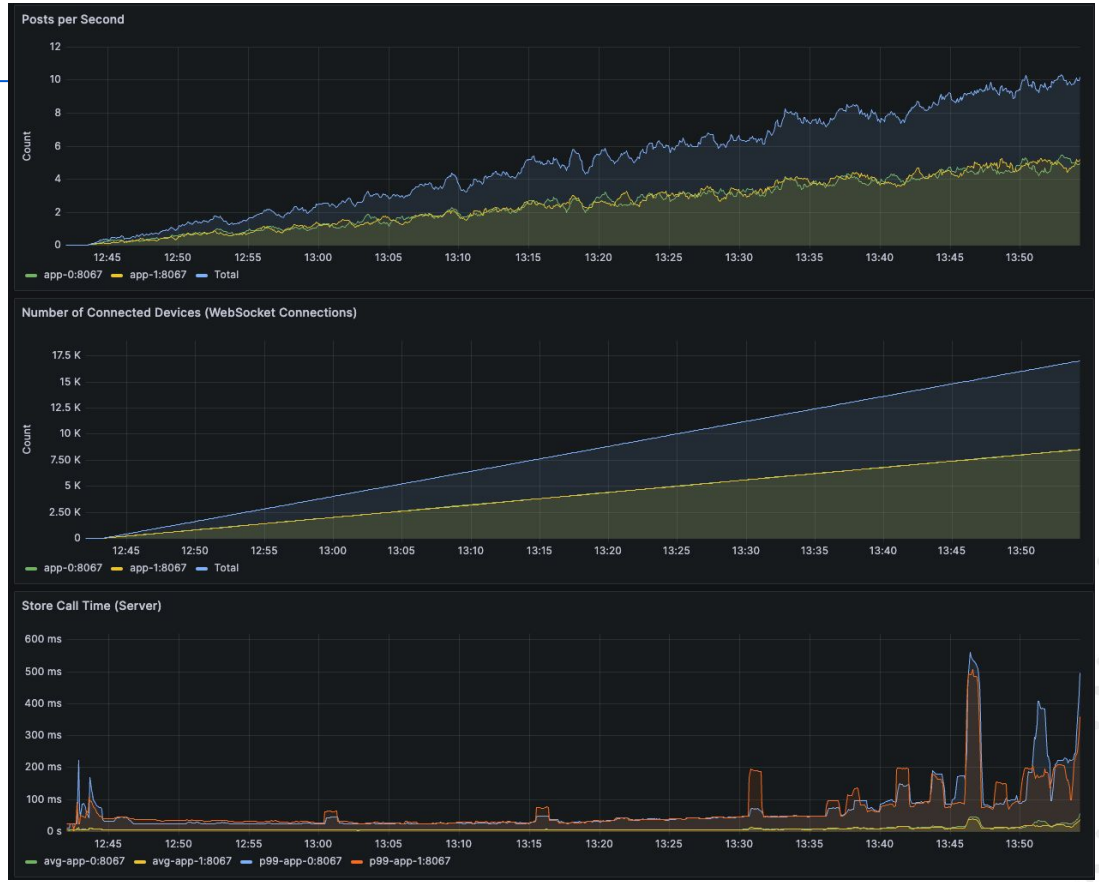
Test Results - 2

Users	Average	P99
7700	< 5ms	< 25ms
14400	< 8ms	< 50ms
16400	< 12ms	< 100ms
17300	Supported Users	



Test Results - 3

Users	Average	P99
8500	< 5ms	< 30ms
12700	< 8ms	< 50ms
15600	< 12ms	< 100ms
17900	Supported Users	



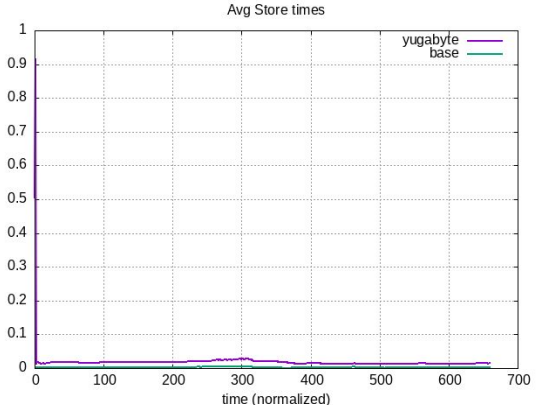
07

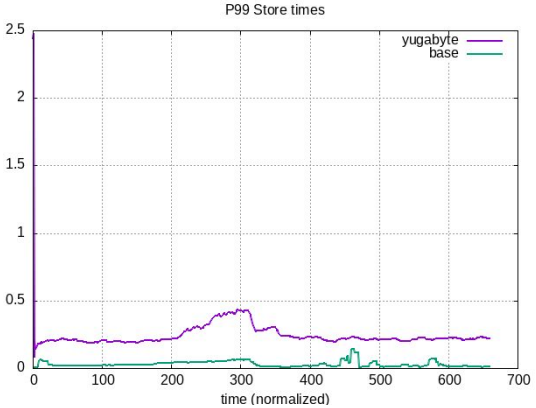
Comparison with PostgreSQL

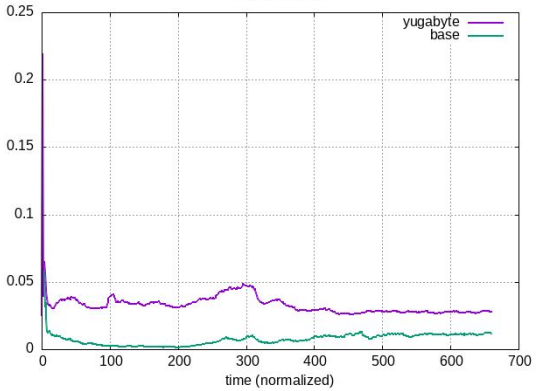
Mattermost with Postgres vs with Yugabyte

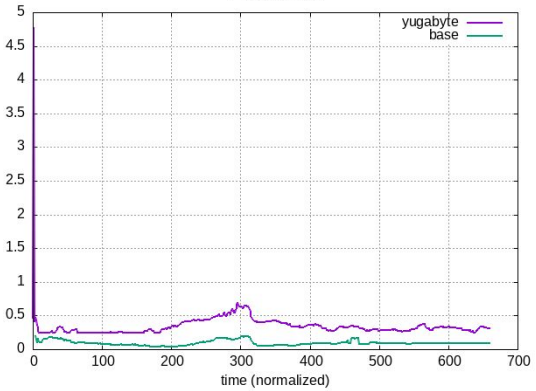
- Postgres:
  - Aurora
  - 1 node
  - 2 vCPU
  - 16 GB ram
- YugabyteDB
  - 3 nodes
  - 24 vCPU
  - 96 GB ram

Mattermost with Postgres vs with YugabyteDB

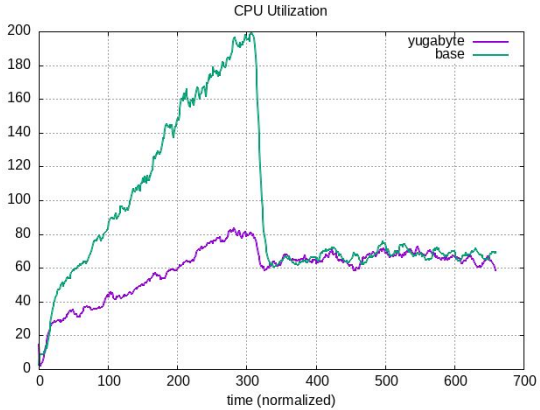


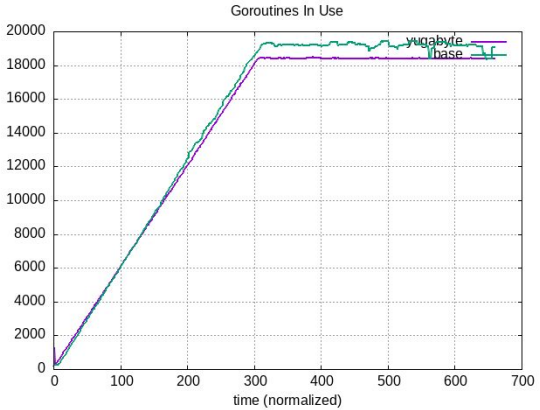


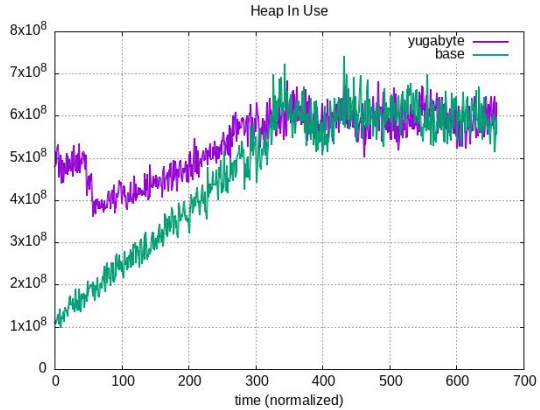


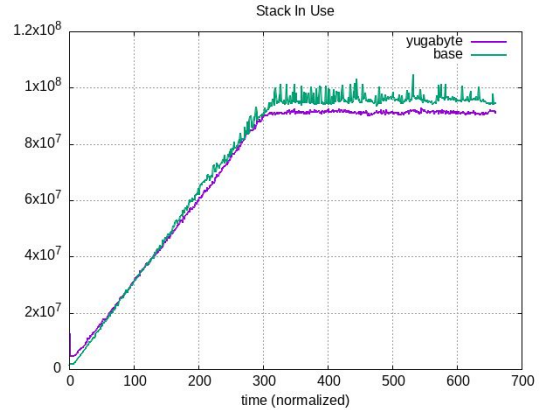


Mattermost with Postgres vs with YugabyteDB









Conclusions

- Mattermosts works and scales on YugabyteDB
- YugabyteDB is very compatible with Postgres
- Postgres is able to handle more with less resources
- YugabyteDB has higher latencies in general
- All previous conclusion are expected and is the price to pay for having the benefits of the distributed database

Mattermost

Thank you.