Team Number: 3

WSDM Project

…to serve and protect data.

Remy Duijsens, Jeroen Nelen, Raphael Frühwirth

TU Delft

Content

- Tech Stack
- Implementation
- Results
- Future improvements

Systems used

- Initial Design:
  - Flask
  - Redis
  - Synchronous
  - Docker-compose

- Final Design:
  - FastAPI
  - Google Spanner
  - Kafka
  - Asynchronous
  - Kubernetes

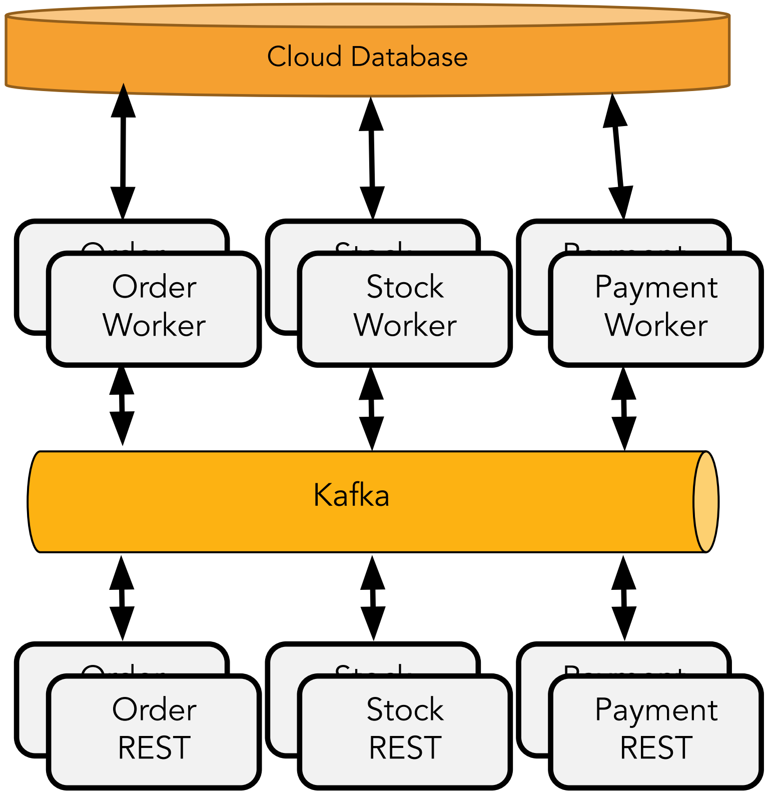
Cloud Spanner

APACHE kafka

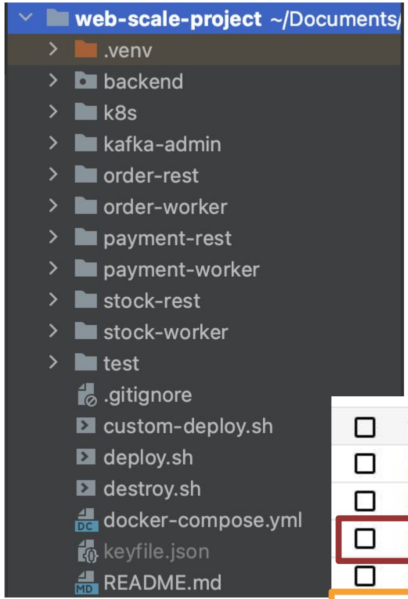
FastAPI

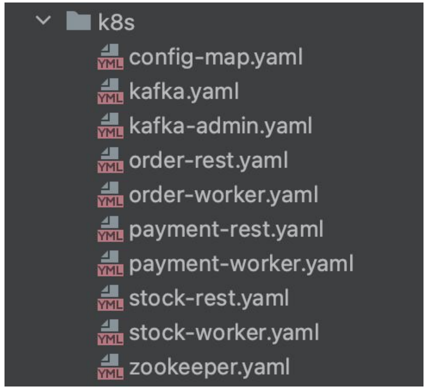
System Implementation

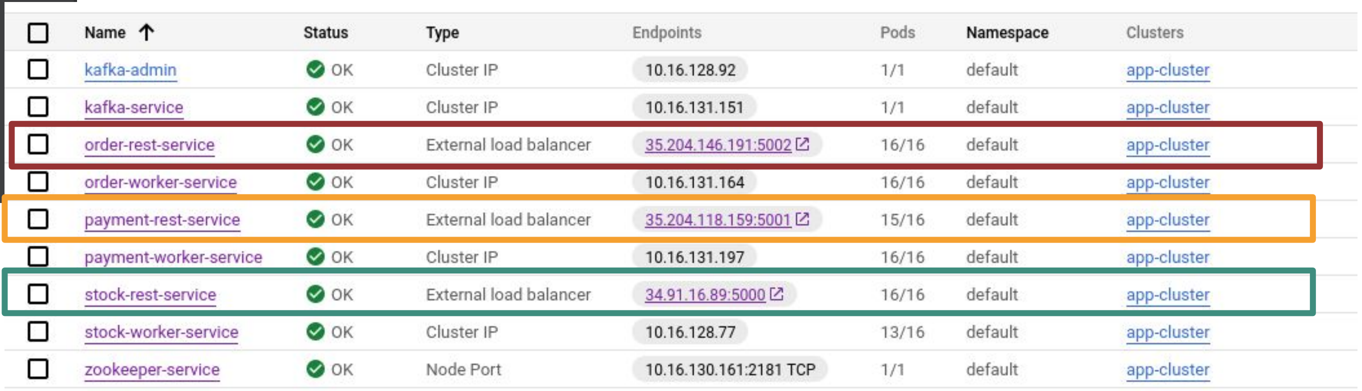
- Brokers and partitions provide scalability.
- Each REST type has its own topic.
- Each worker type has it own topic.
- Partitions are created to allow parallel task execution on worker nodes.
- Scalable and Fault-Tolerant



System Implementation - Overview

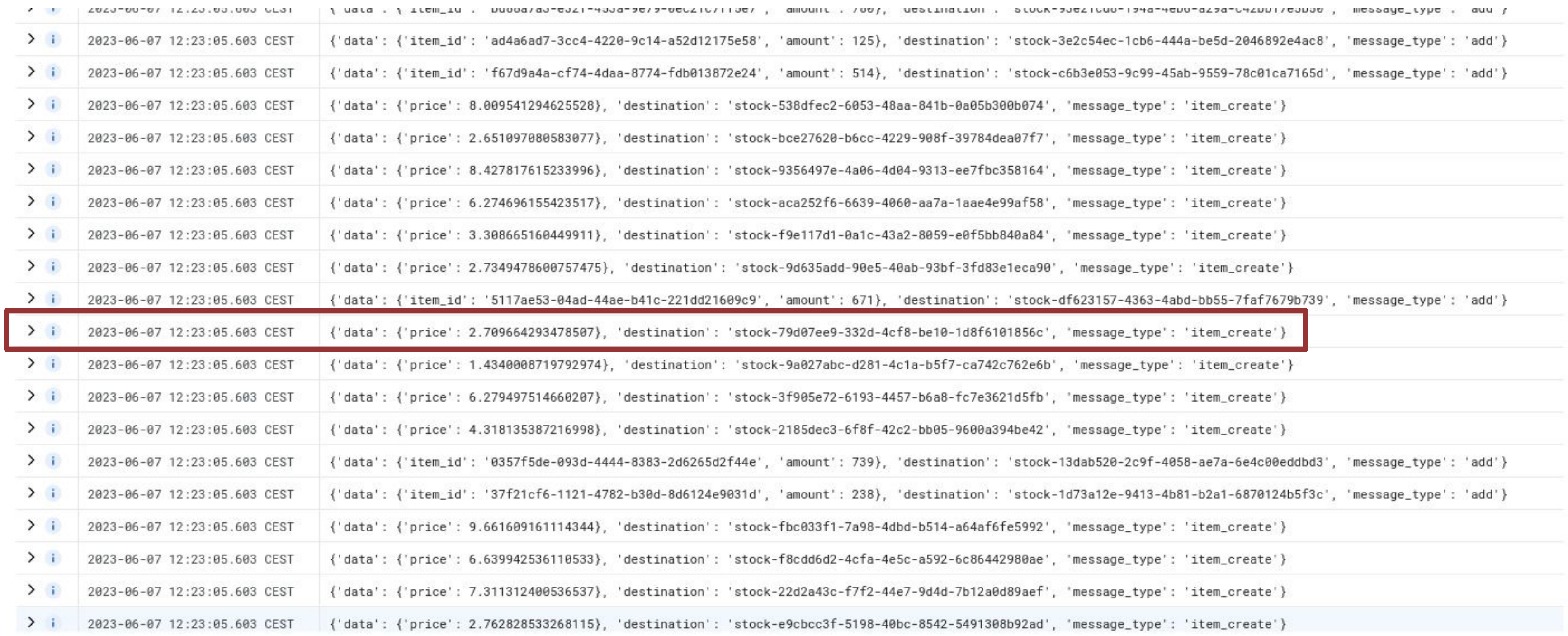

```
web-scale-project ~/Documents/
  .venv
  backend
  k8s
  kafka-admin
  order-rest
  order-worker
  payment-rest
  payment-worker
  stock-rest
  stock-worker
  test
  .gitignore
  custom-deploy.sh
  deploy.sh
  destroy.sh
  docker-compose.yml
  keyfile.json
  README.md
```


```
k8s
  config-map.yaml
  kafka.yaml
  kafka-admin.yaml
  order-rest.yaml
  order-worker.yaml
  payment-rest.yaml
  payment-worker.yaml
  stock-rest.yaml
  stock-worker.yaml
  zookeeper.yaml
```



Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
kafka-admin	✓ OK	Cluster IP	10.16.128.92	1/1	default	app-cluster
kafka-service	✓ OK	Cluster IP	10.16.131.151	1/1	default	app-cluster
order-rest-service	✓ OK	External load balancer	35.204.146.191:5002	16/16	default	app-cluster
order-worker-service	✓ OK	Cluster IP	10.16.131.164	16/16	default	app-cluster
payment-rest-service	✓ OK	External load balancer	35.204.118.159:5001	15/16	default	app-cluster
payment-worker-service	✓ OK	Cluster IP	10.16.131.197	16/16	default	app-cluster
stock-rest-service	✓ OK	External load balancer	34.91.16.89:5000	16/16	default	app-cluster
stock-worker-service	✓ OK	Cluster IP	10.16.128.77	13/16	default	app-cluster
zookeeper-service	✓ OK	Node Port	10.16.130.161:2181 TCP	1/1	default	app-cluster

System Implementation

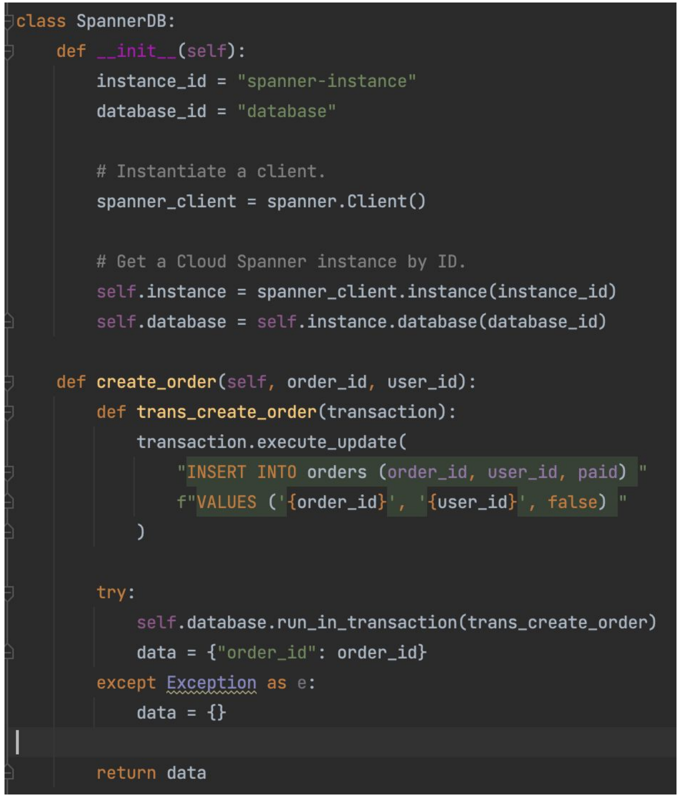


```
2023-06-07 12:23:05.603 CEST   {'data': {'item_id': 'ad4a6ad7-3cc4-4220-9c14-a52d12175e58', 'amount': 125}, 'destination': 'stock-3e2c54ec-1cb6-444a-be5d-2046892e4ac8', 'message_type': 'add'}
2023-06-07 12:23:05.603 CEST   {'data': {'item_id': 'f67d9a4a-cf74-4daa-8774-fdb013872e24', 'amount': 514}, 'destination': 'stock-c6b3e053-9c99-45ab-9559-78c01ca7165d', 'message_type': 'add'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 8.009541294625528}, 'destination': 'stock-538dfec2-6053-48aa-841b-0a05b300b074', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 2.651097080583077}, 'destination': 'stock-bce27620-b6cc-4229-908f-39784dea07f7', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 8.427817615233996}, 'destination': 'stock-9356497e-4a06-4d04-9313-ee7fbc358164', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 6.274696155423517}, 'destination': 'stock-aca252f6-6639-4060-aa7a-1aae4e99af58', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 3.308665160449911}, 'destination': 'stock-f9e117d1-0a1c-43a2-8059-e0f5bb840a84', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 2.7349478600757475}, 'destination': 'stock-9d635add-90e5-40ab-93bf-3fd83e1eca90', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'item_id': '5117ae53-04ad-44ae-b41c-221dd21609c9', 'amount': 671}, 'destination': 'stock-df623157-4363-4abd-bb55-7faf7679b739', 'message_type': 'add'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 2.709664293478507}, 'destination': 'stock-79d07ee9-332d-4cf8-be10-1d8f6101856c', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 1.4340008719792974}, 'destination': 'stock-9a027abc-d281-4c1a-b5f7-ca742c762e6b', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 6.279497514660207}, 'destination': 'stock-3f905e72-6193-4457-b6a8-fc7e3621d5fb', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 4.318135387216998}, 'destination': 'stock-2185dec3-6f8f-42c2-bb05-9600a394be42', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'item_id': '0357f5de-093d-4444-8383-2d6265d2f44e', 'amount': 739}, 'destination': 'stock-13dab520-2c9f-4058-ae7a-6e4c00eddbd3', 'message_type': 'add'}
2023-06-07 12:23:05.603 CEST   {'data': {'item_id': '37f21cf6-1121-4782-b30d-8d6124e9031d', 'amount': 238}, 'destination': 'stock-1d73a12e-9413-4b81-b2a1-6870124b5f3c', 'message_type': 'add'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 9.661609161114344}, 'destination': 'stock-fbc033f1-7a98-4dbd-b514-a64af6fe5992', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 6.639942536110533}, 'destination': 'stock-f8cdd6d2-4cfa-4e5c-a592-6c86442980ae', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 7.311312400536537}, 'destination': 'stock-22d2a43c-f7f2-44e7-9d4d-7b12a0d89aef', 'message_type': 'item_create'}
2023-06-07 12:23:05.603 CEST   {'data': {'price': 2.762828533268115}, 'destination': 'stock-e9cbcc3f-5198-40bc-8542-5491308b92ad', 'message_type': 'item_create'}
```

System Implementation - Spanner

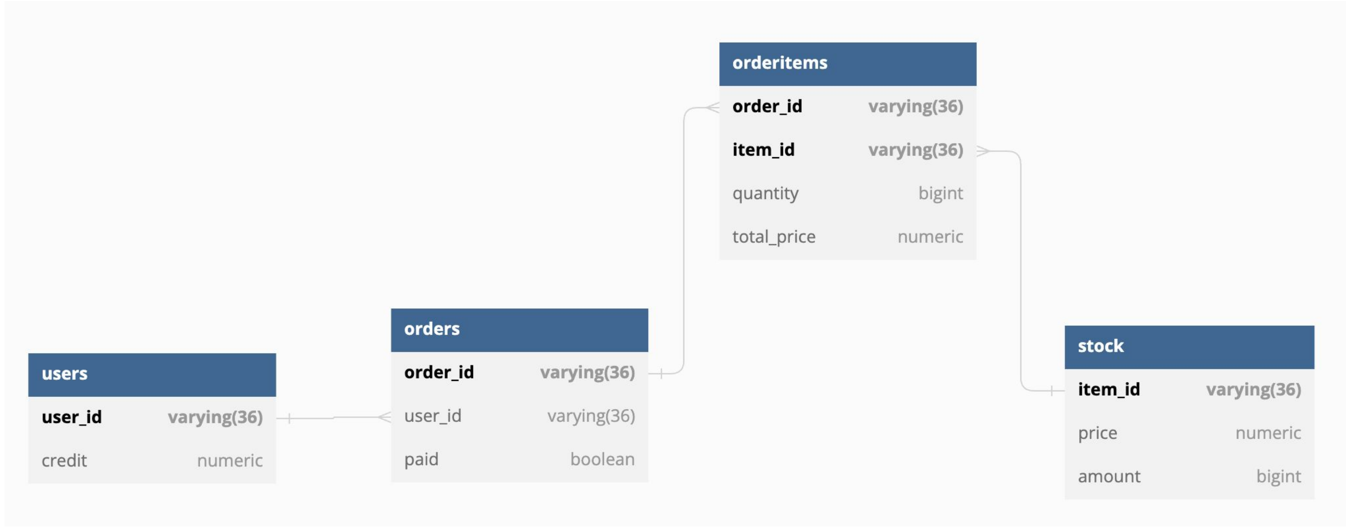
- Within the worker of Kafka
- SQL queries
- Indexing
- Transactions

-> Consistency vs Performance

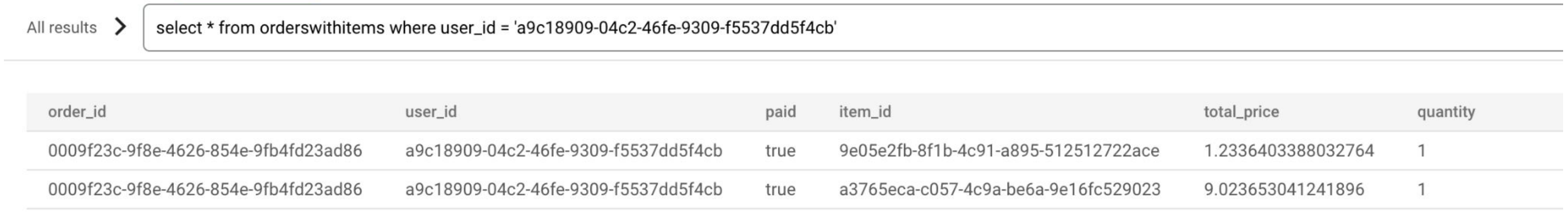


```
class SpannerDB:
    def __init__(self):
        instance_id = "spanner-instance"
        database_id = "database"

        # Instantiate a client.
        spanner_client = spanner.Client()

        # Get a Cloud Spanner instance by ID.
        self.instance = spanner_client.instance(instance_id)
        self.database = self.instance.database(database_id)

    def create_order(self, order_id, user_id):
        def trans_create_order(transaction):
            transaction.execute_update(
                "INSERT INTO orders (order_id, user_id, paid) "
                f"VALUES ('{order_id}', '{user_id}', false) "
            )

        try:
            self.database.run_in_transaction(trans_create_order)
            data = {"order_id": order_id}
        except Exception as e:
            data = {}

        return data
```

Relational DB Schema



Custom view:

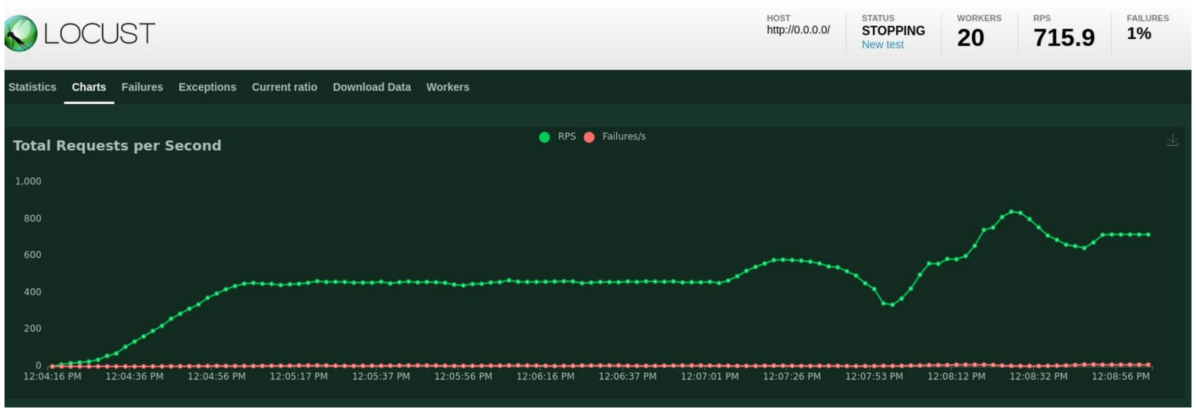


```
select * from orderswithitems where user_id = 'a9c18909-04c2-46fe-9309-f5537dd5f4cb'
```

order_id	user_id	paid	item_id	total_price	quantity
0009f23c-9f8e-4626-854e-9fb4fd23ad86	a9c18909-04c2-46fe-9309-f5537dd5f4cb	true	9e05e2fb-8f1b-4c91-a895-512512722ace	1.2336403388032764	1
0009f23c-9f8e-4626-854e-9fb4fd23ad86	a9c18909-04c2-46fe-9309-f5537dd5f4cb	true	a3765eca-c057-4c9a-be6a-9e16fc529023	9.023653041241896	1

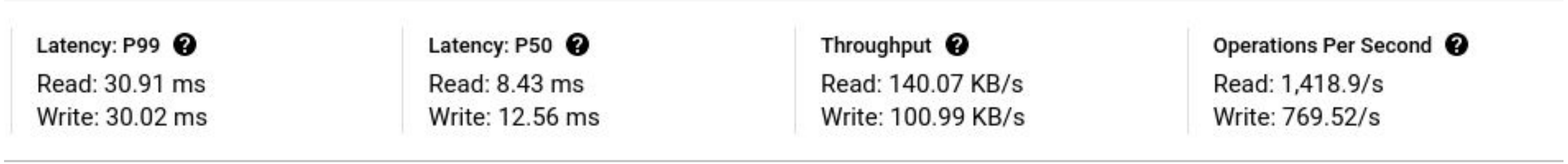
RESULTS - Benchmark

Benchmark Setup	Peak Req / Sec	Scalable?	Robust?
RESTful Microservices + Redis	300	No	No
Kafka + Redis	0 (Unstable)	No	Yes
Kafka + Spanner	950	Yes	Yes

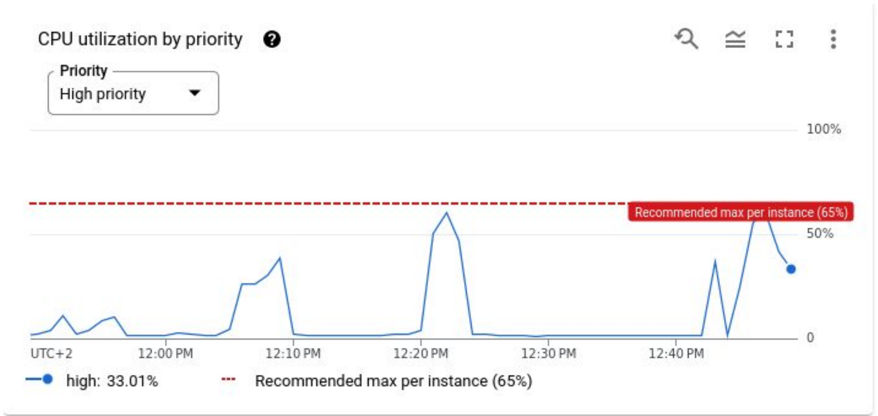


RESULTS - Spanner

- Results



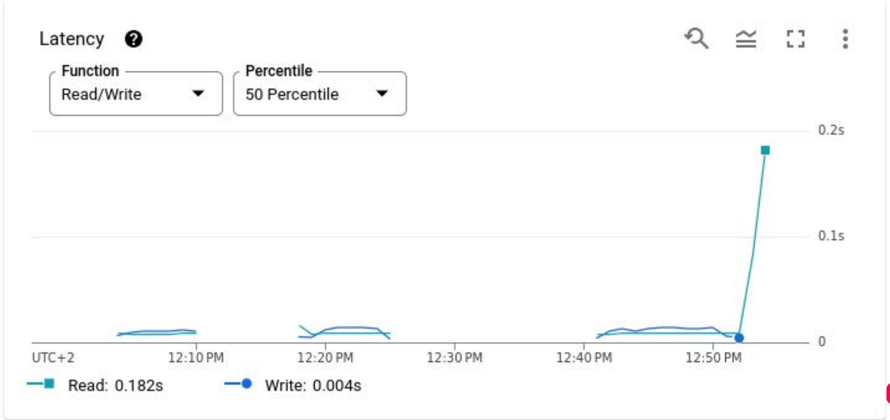
Latency: P99	Latency: P50	Throughput	Operations Per Second
Read: 30.91 ms	Read: 8.43 ms	Read: 140.07 KB/s	Read: 1,418.9/s
Write: 30.02 ms	Write: 12.56 ms	Write: 100.99 KB/s	Write: 769.52/s



CPU utilization by priority

Priority: High priority

● high: 33.01% --- Recommended max per instance (65%)



Latency

Function: Read/Write

Percentile: 50 Percentile

■ Read: 0.182s ● Write: 0.004s

RESULTS - Conclusion

- Kafka -> Extremely fast and scalable
- Async REST nodes and horizontal scaling -> good endpoint scaling
- Lightweight worker nodes -> provides scalability
- Spanner -> Difficult to get good performance
  - -> Yet easily scalable, Linearly
  - -> Region matters(!)
  - -> Efficient queries matter
- Google cloud performance questionable

2 months more?

More Robust implementation of Spanner

- SQLAlchemy + Static typed Python
- Query optimization
- Maybe use CockroachDB. Keep control

Move away from Google Cloud Platform:

- Docker images registry caching
- Crazy Expensive
- Too complex service group & access management

Move to mature tech stack (eg. Java Spring)

Thank you for listening.
Questions?