HPTS 2019

GONG SHOW

Emcee: Peter Alvaro


GONG SHOW
Chuck Taylor
ALL STAR


Cartier
Cartier

Test Infrastructure for Storage Systems

Engineering Quality in Apache Cassandra

Scott Andreas | HPTS Gong Show

About Apache Cassandra

- Distributed database: non-relational row store.
- Designed for multi-DC active-active topologies.
- 1000+ instances, multiple petabytes per cluster.
- Dynamo model, Paxos for partition linearizability.
- Used by Netflix, Facebook, Uber, CERN,
  and over 1500 others.

A Story About Quality

Classification	Found By	JIRA Ticket
Data Loss	Preflight Testing	CASSANDRA-15004: Anti-compaction briefly removes sstables from the read path
Data Loss	Upgrade Tests	CASSANDRA-14958: Counters fail to increment on 2.X to 3.X mixed version clusters
Data Resurrection	Preflight Testing	CASSANDRA-14936: Anticompaction should throw exceptions on errors, not just log them
Corruption / Data Loss	OSS	CASSANDRA-14672: After deleting data in 3.11.3, reads fail: "open marker and close marker have different deletion times"
Corruption	Preflight Testing	CASSANDRA-14912: LegacyLayout errors on collection tombstones from dropped columns
Corruption	Fuzzing	CASSANDRA-14843: Drop/add column name with different Kind can result in corruption
Corruption	OSS Review	CASSANDRA-14568: CorruptSSTableExceptions in 3.0.17.1 (CASSANDRA-14568 v2) Static collection deletions are corrupted in 3.0 <-> 2.{1,2} messages
Corruption	Preflight Testing	CASSANDRA-14749: Collection Deletions for Dropped Columns in 2.1/3.0 mixed-mode can delete rows
Corruption	Preflight Testing	CASSANDRA-14568: Static collection deletions are corrupted in 3.0 -> 2.{1,2} messages
Incorrect Resp / Data Loss	Fuzzing	CASSANDRA-14861: Inaccurate sstable min/max metadata can cause data loss
Incorrect Resp / Data Loss	Diff Test	CASSANDRA-14823: Legacy sstables with range tombstones spanning multiple index blocks create invalid bound sequences on 3.0+ (#1193)
Incorrect Response	Fuzzing	CASSANDRA-14873: Missing rows when reading 2.1 SSTables in 3.0
Incorrect Response	Fuzzing	CASSANDRA-14838: Dropped columns can cause reverse sstable iteration to return prematurely
Incorrect Response	Diff Te	CASSANDRA-1 3: R cross index block boundaries can cause incomplete reverse results in some cases.
Incorrect Response	Diff Te	CASSANDRA-1 6: D the last UNF
Stability	Other	CASSANDRA-14991: SSL Cert Hot Reloading should defensively check for sanity of the new keystore/truststore before loading it
Stability	Preflight Testing	CASSANDRA-14794: Avoid calling iter.next() in a loop when notifying indexers about range tombstones
Stability	Preflight Testing	CASSANDRA-14 [illegible]
Stability	Preflight Testing	CASSANDRA-14657: Handle failures in upgradesstables/cleanup/relocatee
Behavior Change / Incorrect Rep	OSS	CASSANDRA-14638: Column result order can change in 'SELECT *' results when upgrading from 2.1 to 3.0 causing response corruption for queries using prepared statements...
Availability	Upgrade Tests	CASSANDRA-14919: Regression in paging queries in mixed version clusters
Concurrency	OSS (Datastax)	CASSANDRA-14871: Datastax - Severe concurrency issues in STCS,DTCS,TWCS,TMD.Topology,TypeParser
Concurrency	Preflight Testing	CASSANDRA-14554: LifecycleTransaction encounters ConcurrentModificationException when used in multi-threaded context
Correctness	OSS (Datastax)	CASSANDRA-14869: Datastax - Fix full ring range subtraction
Performance	Preflight Testing	CASSANDRA-14935: PendingAntiCompaction should be more judicious in the compactions it cancels
Performance	Fuzzing	CASSANDRA-14894: RangeTombstoneList doesn't properly clean up mergeable or superseded rts in some cases
Safety	Review	CASSANDRA-14824: Expand range tombstone validation checks to multiple interim request stages
Safety	Preflight Testing	CASSANDRA-14763: Fail incremental repair prepare phase if it encounters sstables from un-finalized sessions
Test Failure	Upgrade Tests	CASSANDRA-14920: Some comparisons used for verifying paging queries in dtests only test the column names and not values

30+ Critical Bugs Found + Fixed

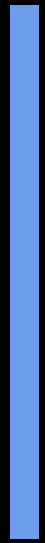
Data Loss, Corruption, Incorrect Response, More...

Time

Cultural Change

Sustained Investment in Quality

“Could Kyle still break it?”



“Yes”

Methodologies

- **Model-based testing:** formal specification of database, randomized generators, and a verifier to validate correctness of responses.
  - Soon: continuous execution across thousands of cores.
  - Also: Executed concurrently with fault injection.
- **Diff testing:** comparison of billions of queries between V1 and V2.
- **Replay Testing:** Capture and execution of shadow traffic and comparison.
- **Source Audits:** Targeted review of soft spots, phased replacement.

Process

- **Commitment to users:** stable “dot-zero” for critical applications.
- **Feature freeze** on upcoming major release.
  - Collective focus on quality.
- Published test and qualification plans.
- Establishment of quality metrics.
  - “Find rate” by methodology, “pass rate” by test.
  - Goal: Convergence across all methodologies.

Results

- A safe and stable release: Apache Cassandra 3.0.19.
- A path to Apache Cassandra 4.0 in 2020.

Goals

- An end to “waiting a year after major release to upgrade”
- Automated qualification infra + formal model (time + electricity = confidence)
- Faster cadence of safe, major releases toward evolving Cassandra to rival proprietary systems such as DynamoDB and BigTable.

Engineering Quality in Apache Cassandra

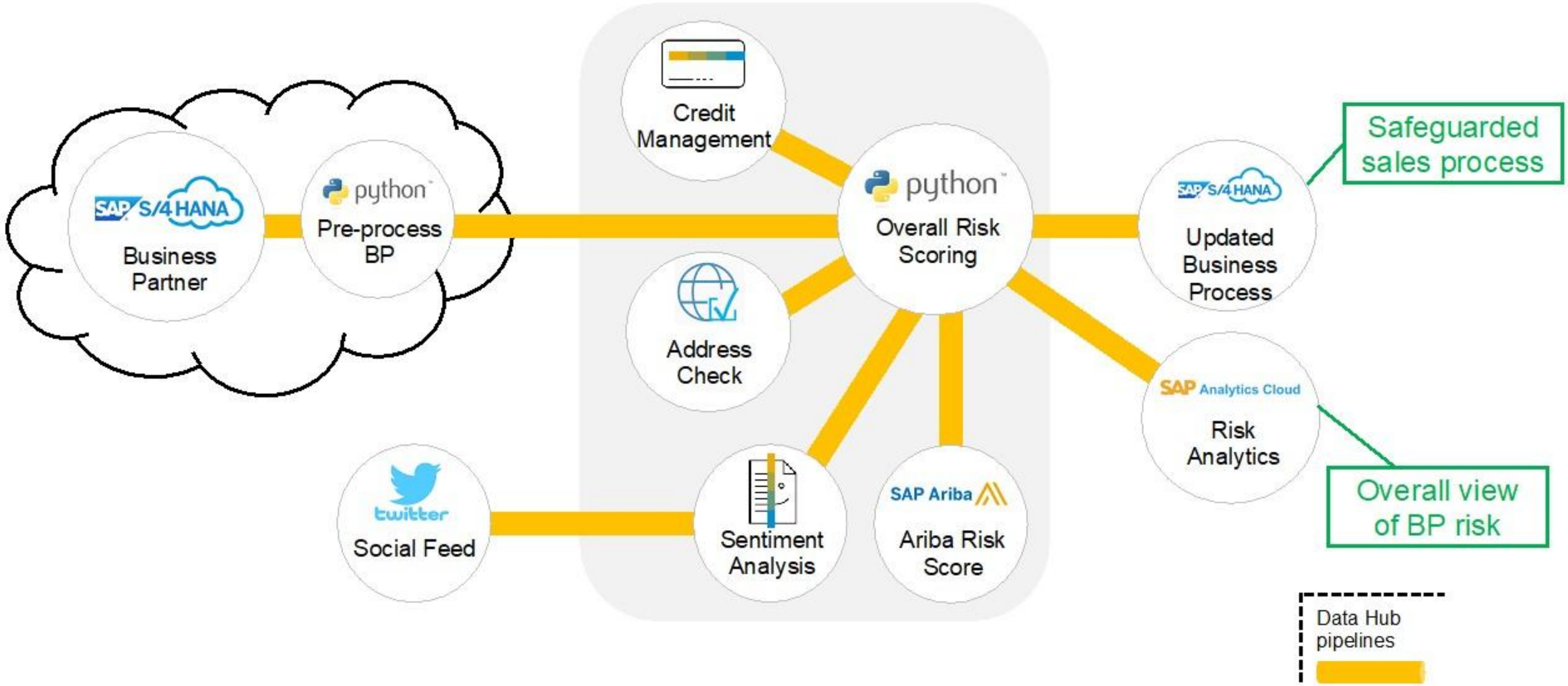
Engineering Quality in Apache Cassandra

Scott Andreas | HPTS Gong Show

Thomas Zurek (SAP)
DWs + Data Lakes
⇒ Data Hubs

Customer Risk Intelligence

The implementation: customer risk scored across all disparate data assets!



The Case for Latency-Aware Query Optimization in a Global DBMS

HPTS 2019 Gong Show

Presented by Rebecca Taft

CockroachDB

- Geo-Distributed
- SQL
- Scalable
- Resilient

Cockroach LABS

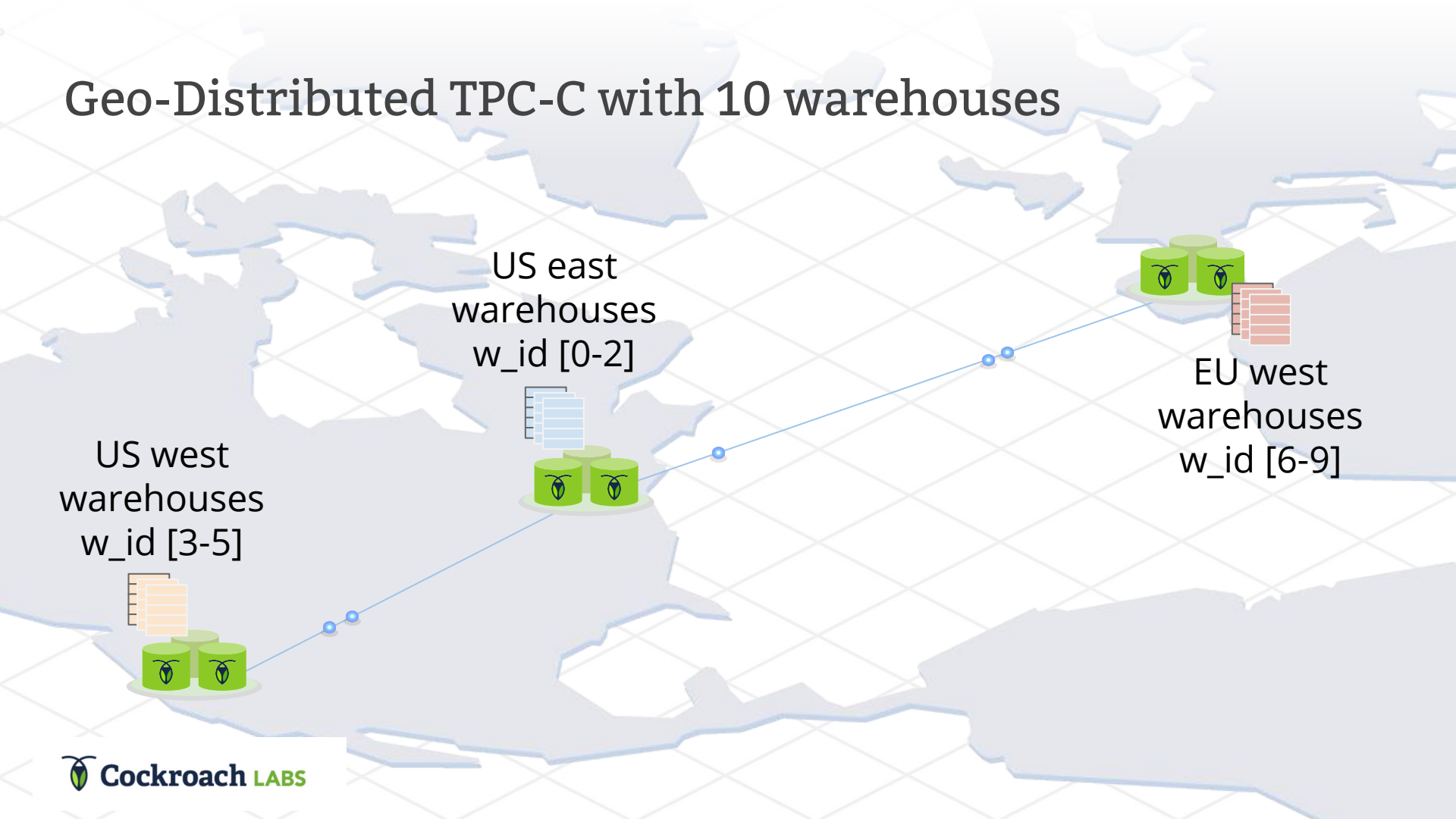
Locality-Aware SQL Optimization

- For OLTP, **both** latency and bandwidth matter
- Cost model must account for both

Global Kitchen Supply: The case of the defective vitamix blenders

"How many vitamix blenders in recent orders from LA were supplied from the warehouse in Toulouse?"

Geo-Distributed TPC-C with 10 warehouses



Cockroach LABS

Model with TPC-C

"How many vitamix blenders in recent orders from LA were supplied from the warehouse in Toulouse?"

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Customers from LA

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Customers from LA

Recent orders

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Customers from LA

Recent orders

Vitamix blender

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Customers from LA

Recent orders

Vitamix blender

Supplied by Toulouse

Model with TPC-C

```
SELECT
    sum(ol_quantity)
FROM
    order_line
WHERE
    ol_w_id = 5
    AND ol_d_id = 1
    AND ol_o_id > 3000
    AND ol_i_id = 49712
    AND ol_supply_w_id = 9;
```

How many?

Customers from LA

Recent orders

Vitamix blender

Supplied by Toulouse

Plan chosen by CockroachDB v19.2

```
scalar-group-by
    ├── index-join order_line
    │    └── scan order_line@order_line_stock_fk_idx
    │         └── constraint: /6/5/3/2/1/4: [/9/49712/5/1/3001 - /9/49712/5/1]
    └── aggregations
         └── sum
              └── variable: ol_quantity
```

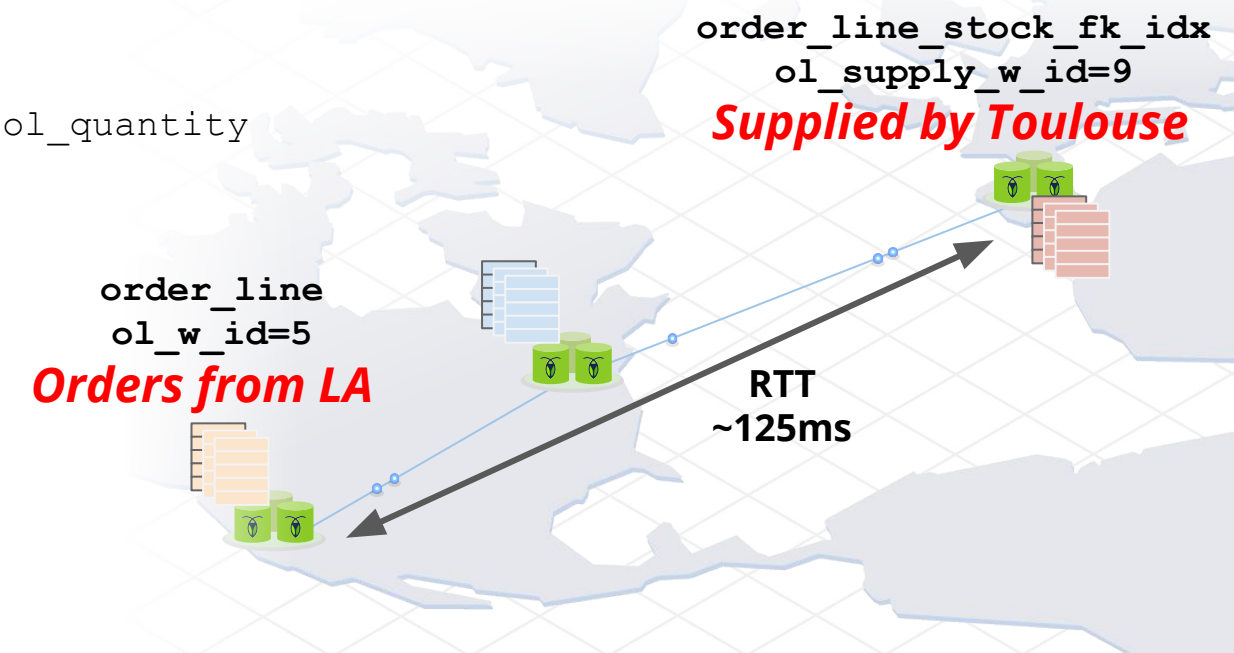
Time = 254 ms

Cockroach LABS

Plan chosen by CockroachDB v19.2

```
scalar-group-by
    ├── index-join order_line
    │    └── scan order_line@order_line_stock_fk_idx
    │         └── constraint: /6/5/3/2/1/4: [/9/49712/5/1/3001 - /9/49712/5/1]
    └── aggregations
         └── sum
              └── variable: ol_quantity
```

Time = 254 ms



Cockroach LABS

Plan chosen by CockroachDB v20.1

```
scalar-group-by
    ├── select
    │    ├── scan order_line
    │    │    ├── constraint: /3/2/-1/4: [/5/1 - /5/1/3001]
    │    │    └── flags: force-index=primary
    │    └── filters
    │         ├── ol_i_id = 49712
    │         └── ol_supply_w_id =
    └── aggregations
         └── sum
              └── variable: ol_quantity
```

Over 100x faster!

Time = 1.4 ms

Cockroach LABS

Summary

1.
Geo-distributed SQL optimizers must be locality-aware

2.
For OLTP workloads, must consider WAN bandwidth *and* latency

3.
Help us build our locality-aware optimizer. We are hiring!

Thank you.

becca@cockroachlabs.com
CockroachLabs.com
github.com/cockroachdb/cockroach

Presented by Rebecca Taft

Cockroach LABS

The Case for Converged Databases

Danica Porobic

Principal Member of Technical Staff

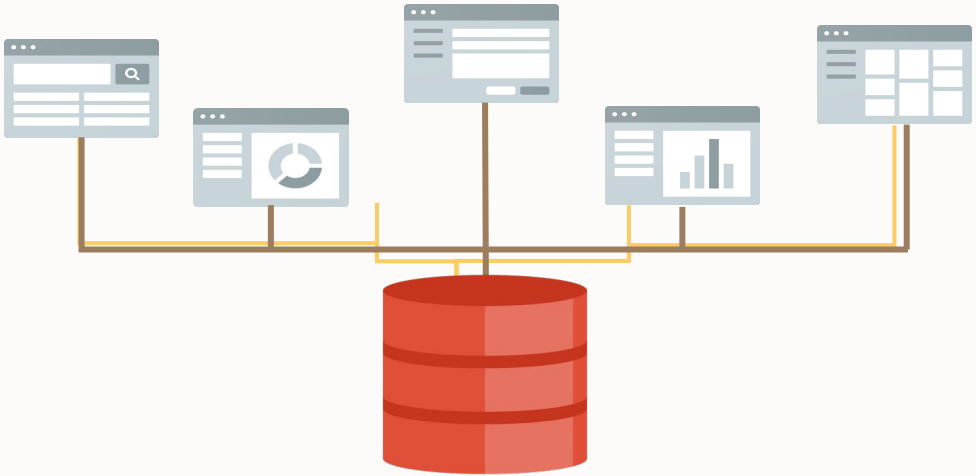
Oracle Database In-Memory

Safe harbor statement

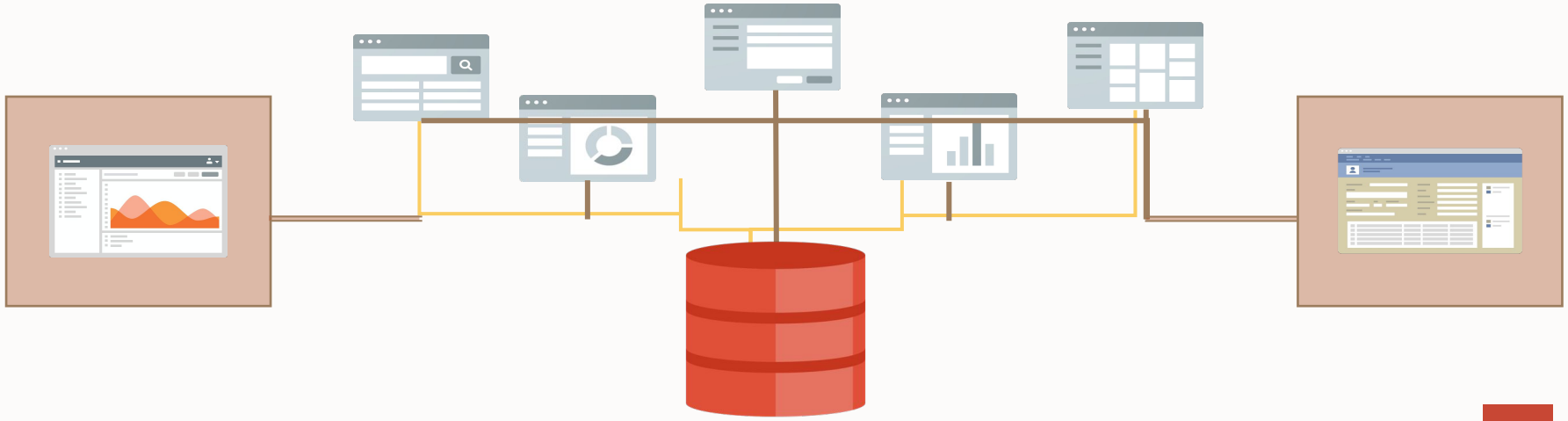
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Historically, developers built large monolithic applications using one data store



These applications and data stores become difficult to maintain, and unresponsive to change over time



This lead to an alternative approach, one that has a separate database per microservice

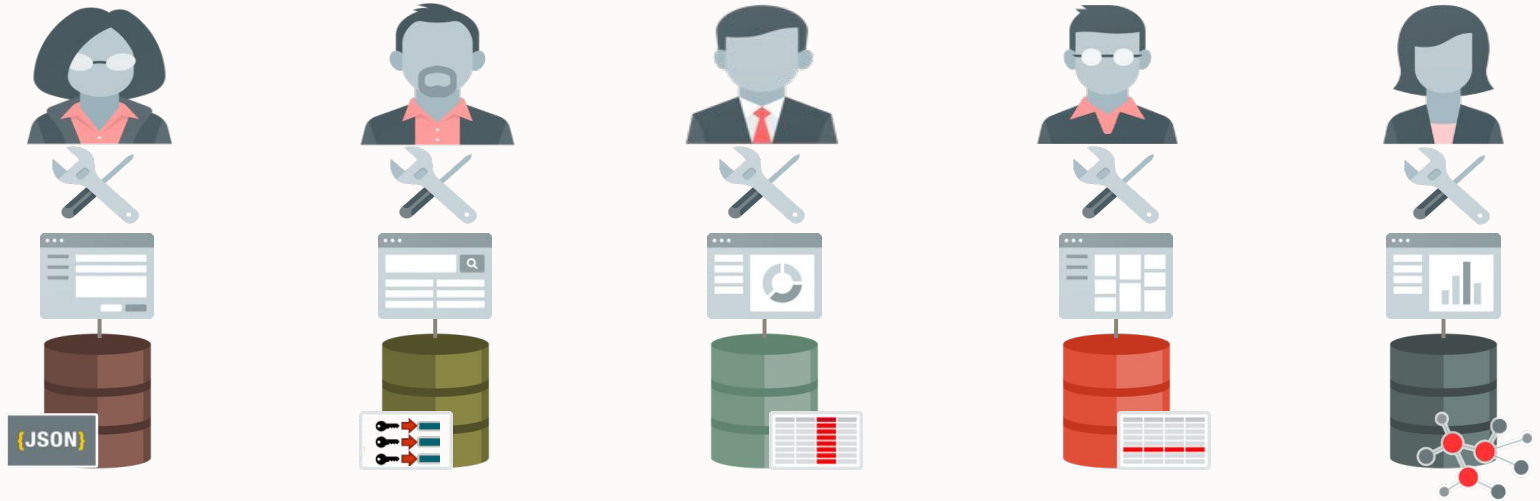
Each database is specialized for a workload or data type: document, key-value, analytic, relational, graph

The goal was to create service independence using best-of-breed databases for each workload or data type

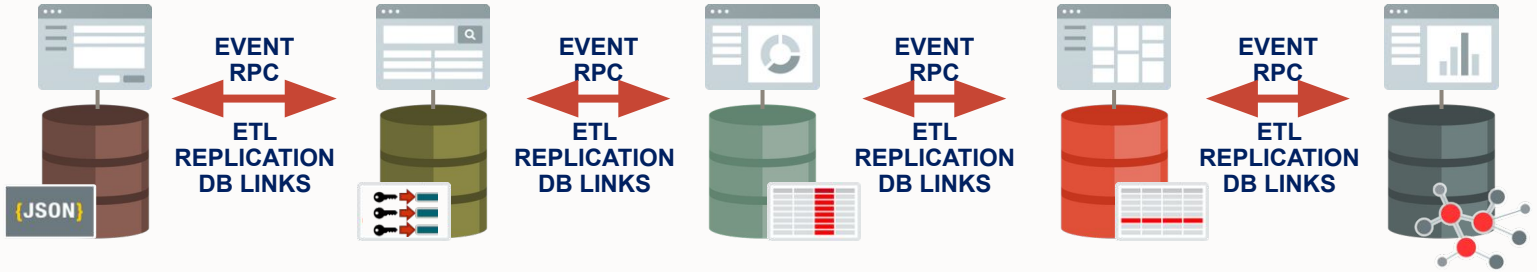
But each single-purpose database that is deployed fragments the overall data architecture

Single-purpose databases require apps to use their proprietary APIs, language, and transactions models instead of standards like SQL

Each single-purpose database has different operational needs and limitations, requiring unique management and skills

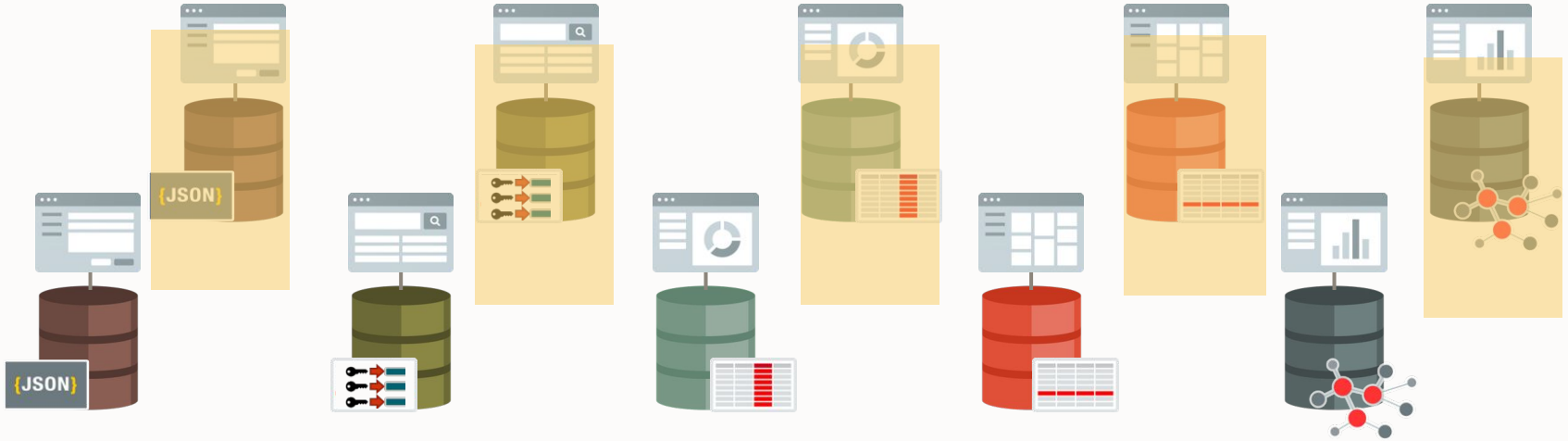


Data propagation is inherently difficult and causes unavoidable data delays and data divergence

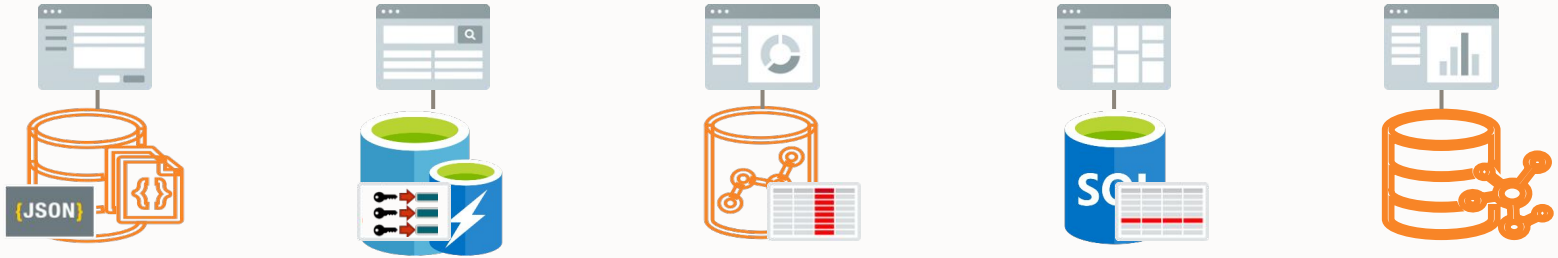


Separate security policies must be implemented in every database and must be re-implemented when app or policies change

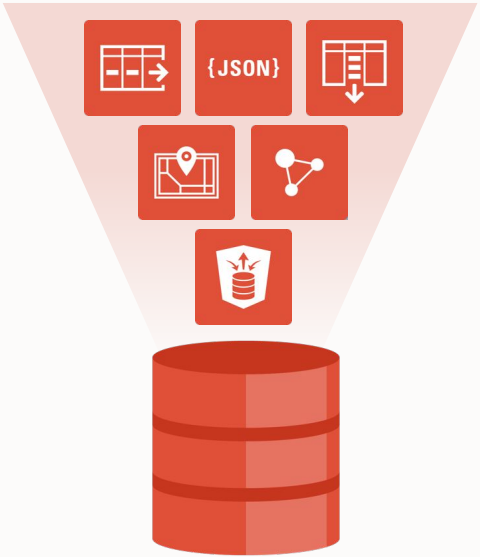
High availability and scalability mechanisms and configuration are specific to each single-purpose database



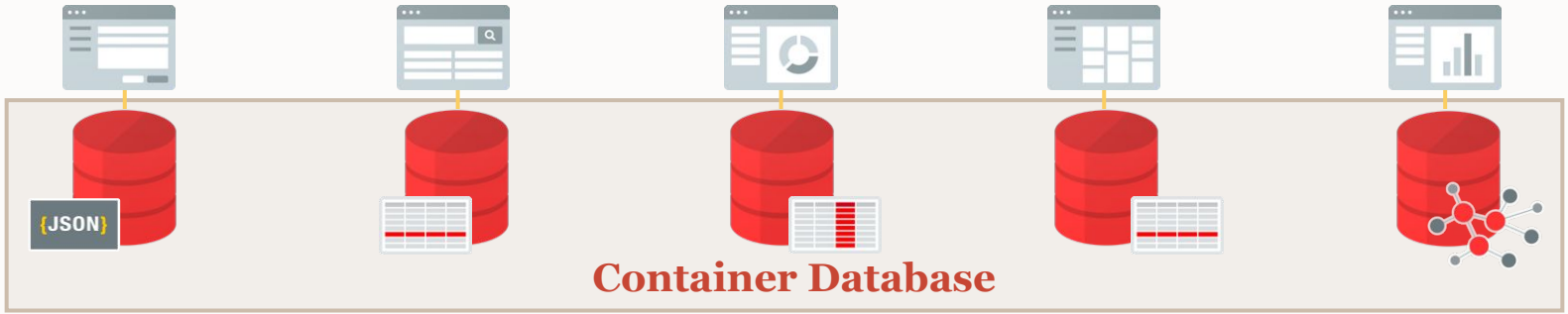
Cloud providers offer different proprietary cloud services that require apps to change when you change cloud



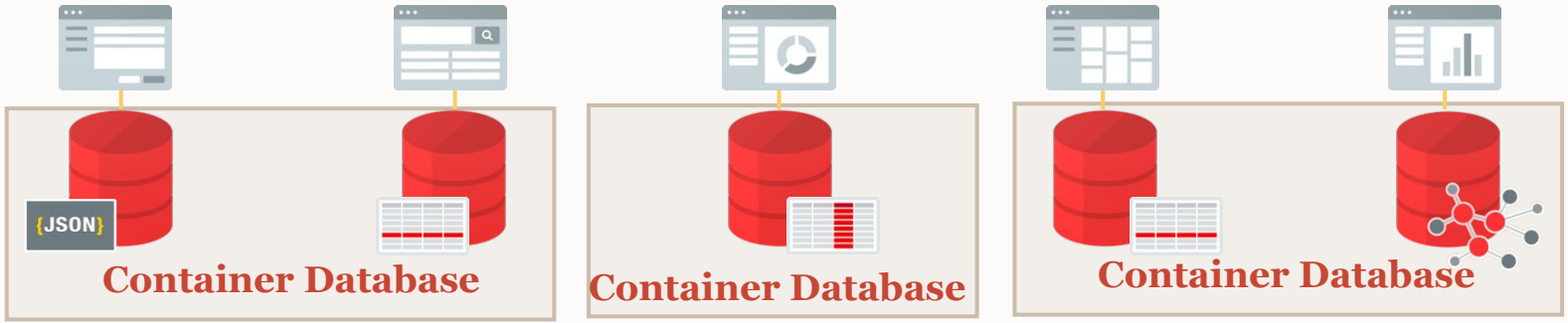
Integrating fragmented databases to create a complete, available, secure, and scalable solution is complex and custom



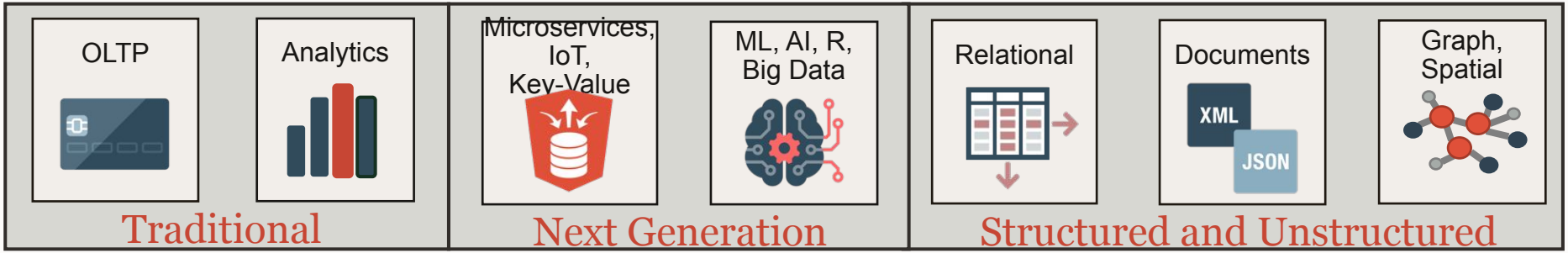
Converge many databases, data models, and apps into one container database



Or use multiple container databases



A Multi-Model Database Radically Simplifies Data Management



Single Database Engine Supports all Workloads and Data

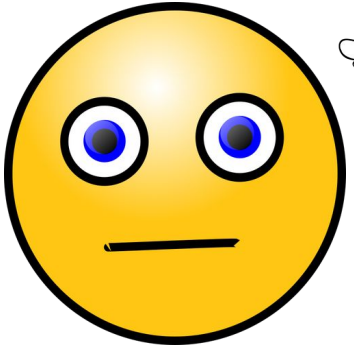
Thank you!

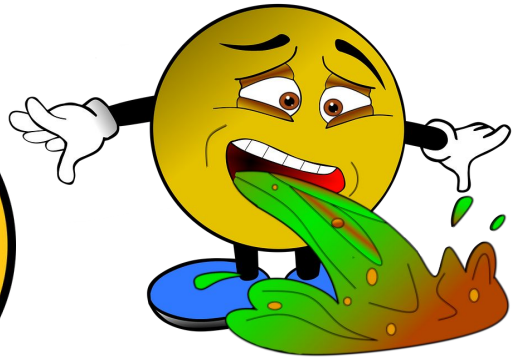
Training for Speech Recognition on Coprocessors

Sebastian Baunsgaard, Sebastian B. Wrede, ***Pınar Tözün***
IT University of Copenhagen

IT UNIVERSITY OF COPENHAGEN

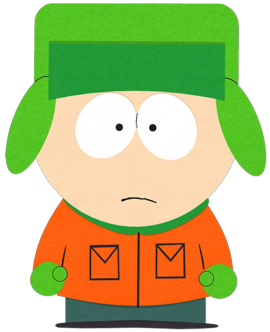
reactions people who know me give when i say i work on speech recognition





how did i get into this?

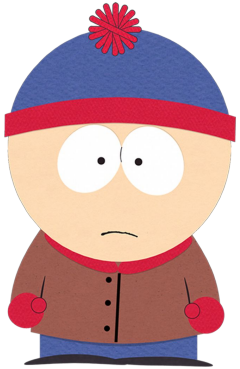
sebastians ***me***



Could you supervise our MSc thesis?

What would you like to work on?

Automatic speech recognition



Why are you talking to me?

We want to make it scalable

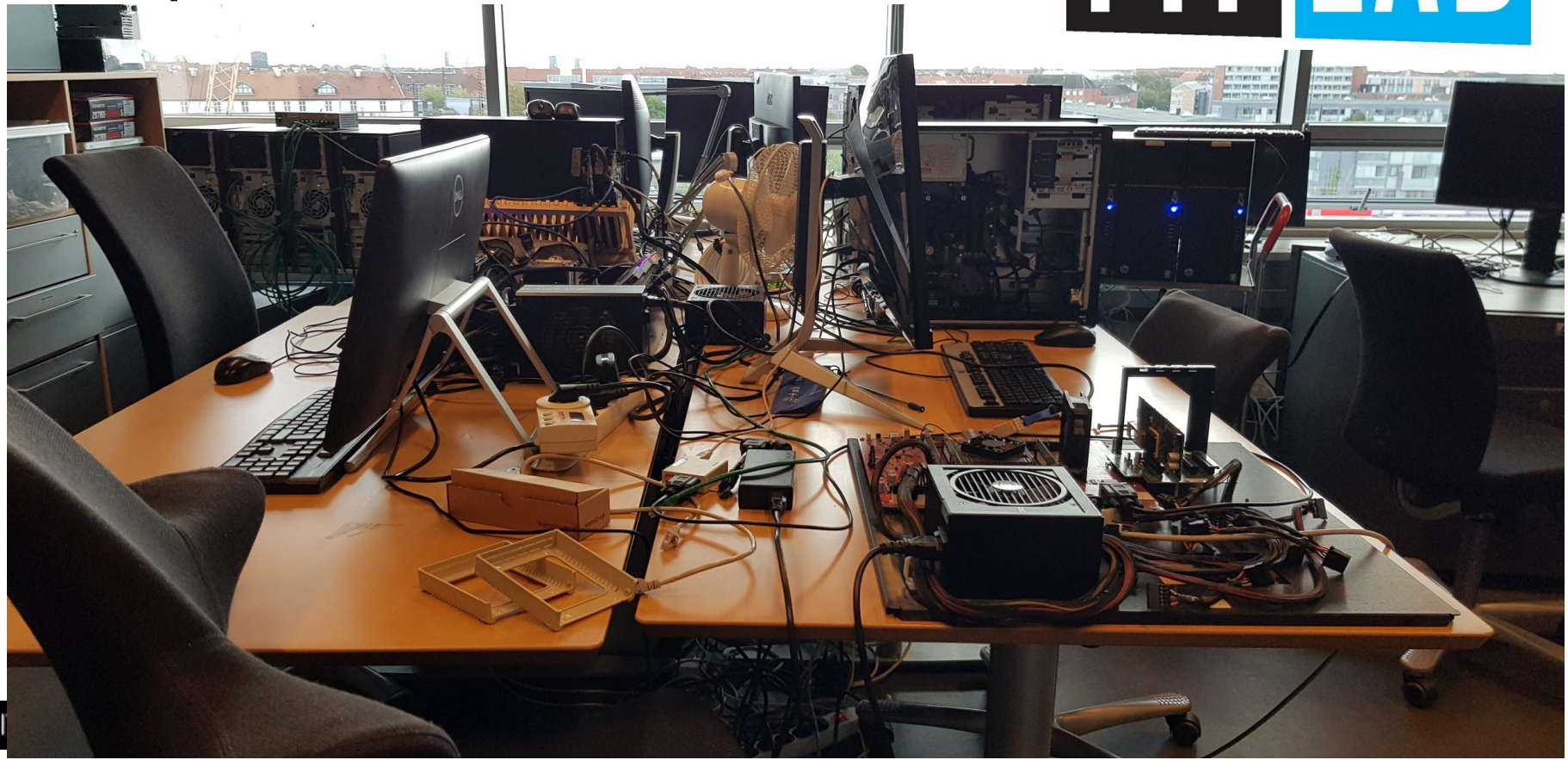
ok then

needed to add some hardware dimension, though

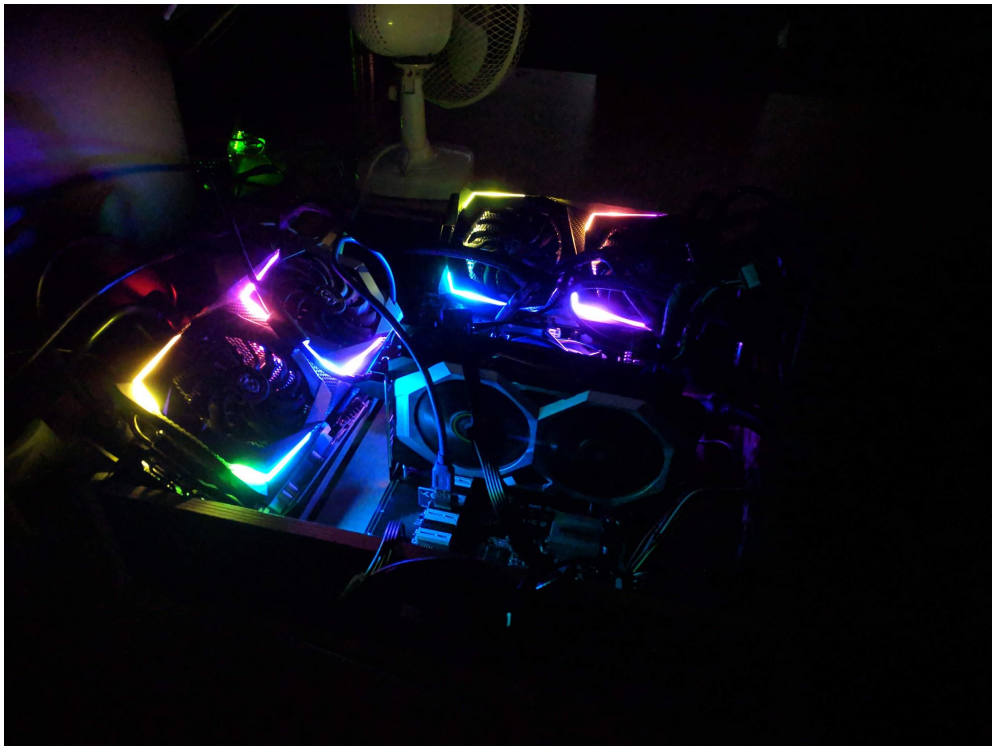
but most student's attitude when i talk about hardware is like ..

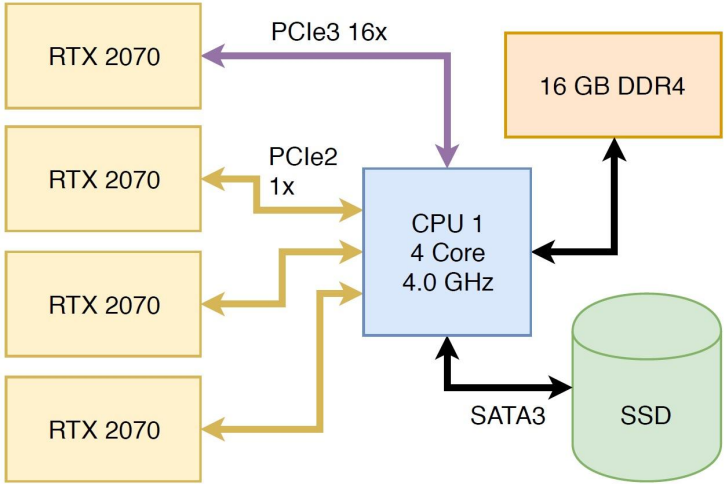
so i placed sebastians into ...

PIT LAB

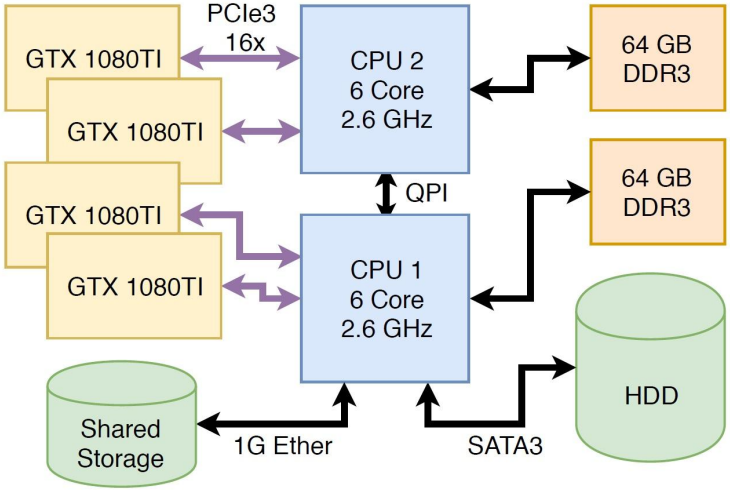


a month later, they built this

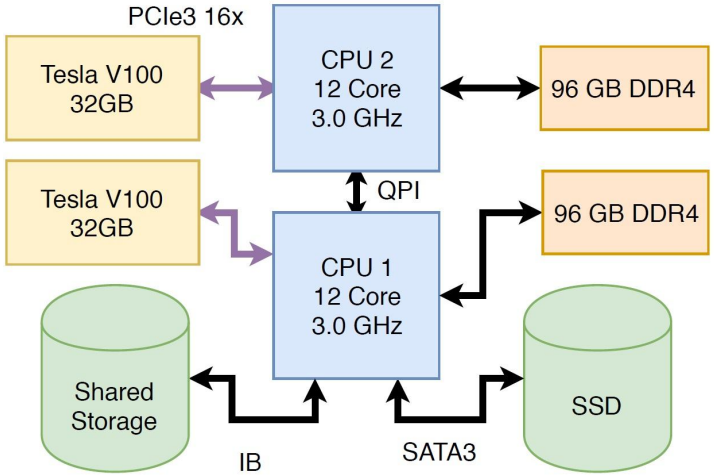




sys1$

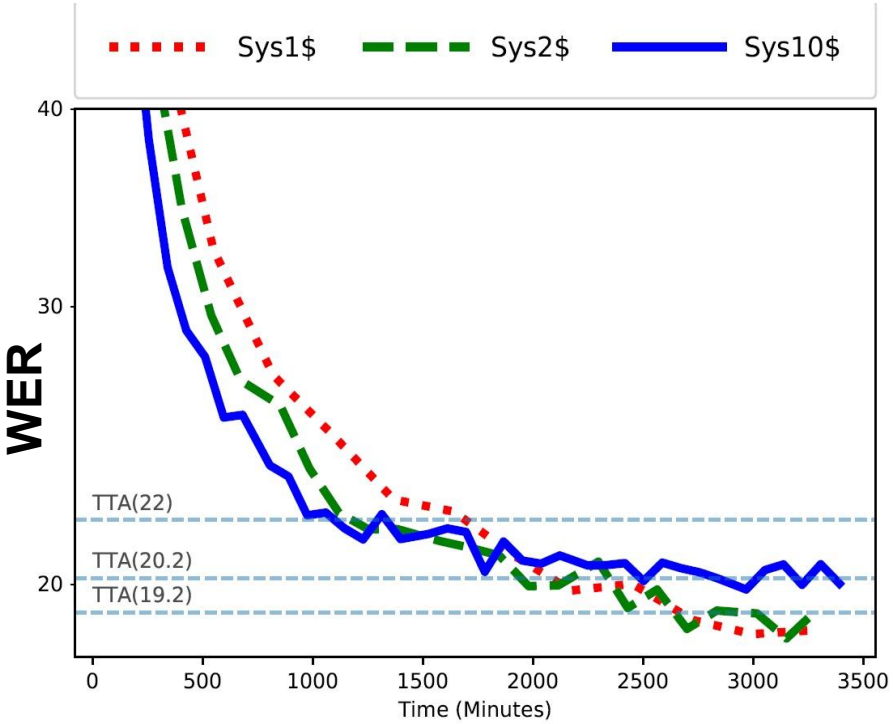


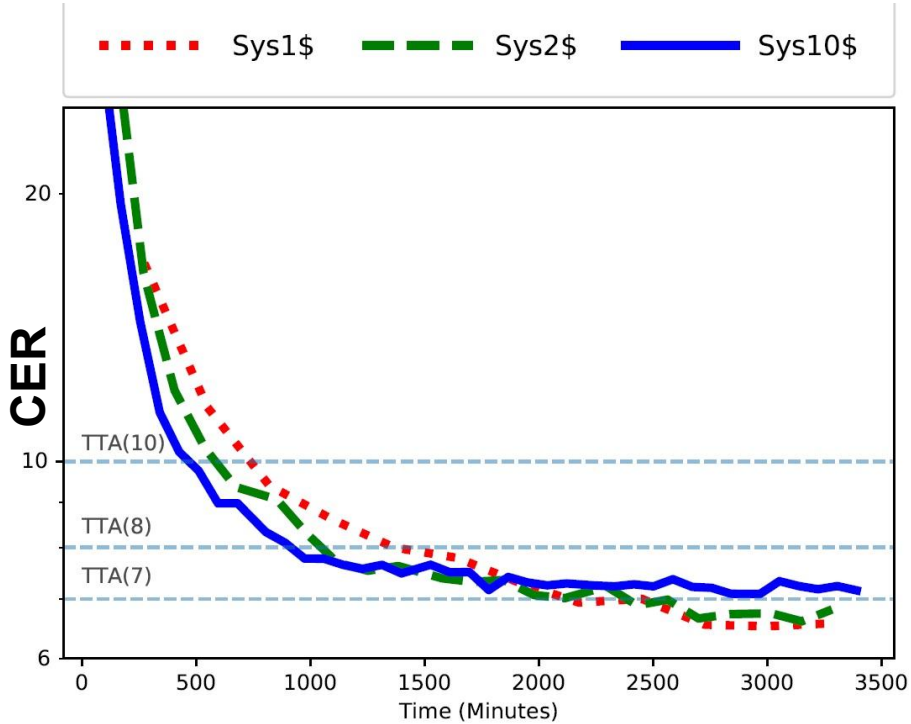
sys2$



sys10$

training of acoustic model based on neural networks on co-processors - time-to-accuracy





conclusion

- very powerful co-processors more and more widely available for machine learning
- but takes a lot to exploit, no free lunch as usual
- need to invest further in improving ML libraries for accelerating model training on heterogeneous hardware
- on the other hand, low-budget platforms may be good enough for your needs

Fault-tolerance is not a technical problem

Josh Leners, Two Sigma

Fault-tolerance is not a technical problem

Josh Leners, Two Sigma

This is not a financial talk:

This document is being distributed for informational and educational purposes only and is not an offer to sell or the solicitation of an offer to buy any securities or other instruments. The information contained herein is not intended to provide, and should not be relied upon for, investment advice. The views expressed herein are not necessarily the views of Two Sigma Investments, LP or any of its affiliates (collectively, “Two Sigma”). Such views reflect the assumptions of the author(s) of the document and are subject to change without notice. The document may employ data derived from third-party sources. No representation is made by Two Sigma as to the accuracy of such information and the use of such information in no way implies an endorsement of the source of such information or its validity.

The copyrights and/or trademarks in some of the images, logos or other material used herein may be owned by entities other than Two Sigma. If so, such copyrights and/or trademarks are most likely owned by the entity that created the material and are used purely for identification and comment as fair use under international copyright and/or trademark laws. Use of such image, copyright or trademark does not imply any association with such organization (or endorsement of such organization) by Two Sigma, nor vice versa

Fault-tolerance ~~is not a technical problem~~
is an epistemological problem

Fault-tolerance ~~is not a technical problem~~ is an epistemological problem

Techniques (e.g., consensus) expand our knowledge

“When a single replica fails, we won’t lose data and can still make progress.”

“We’ve survived a simulated partition”

“The cable cleaners unplugged a rack last week and we were OK”

Fault-tolerance ~~is not a technical problem~~ is an epistemological problem

Techniques (e.g., consensus) can't provide judgment

"What's this *k* parameter?"

"Should I Paxos all the things?"

"What's the impact of failure?"

Fault-tolerance ~~is not a technical problem~~ is an epistemological problem

Byzantine fault tolerance won't save you

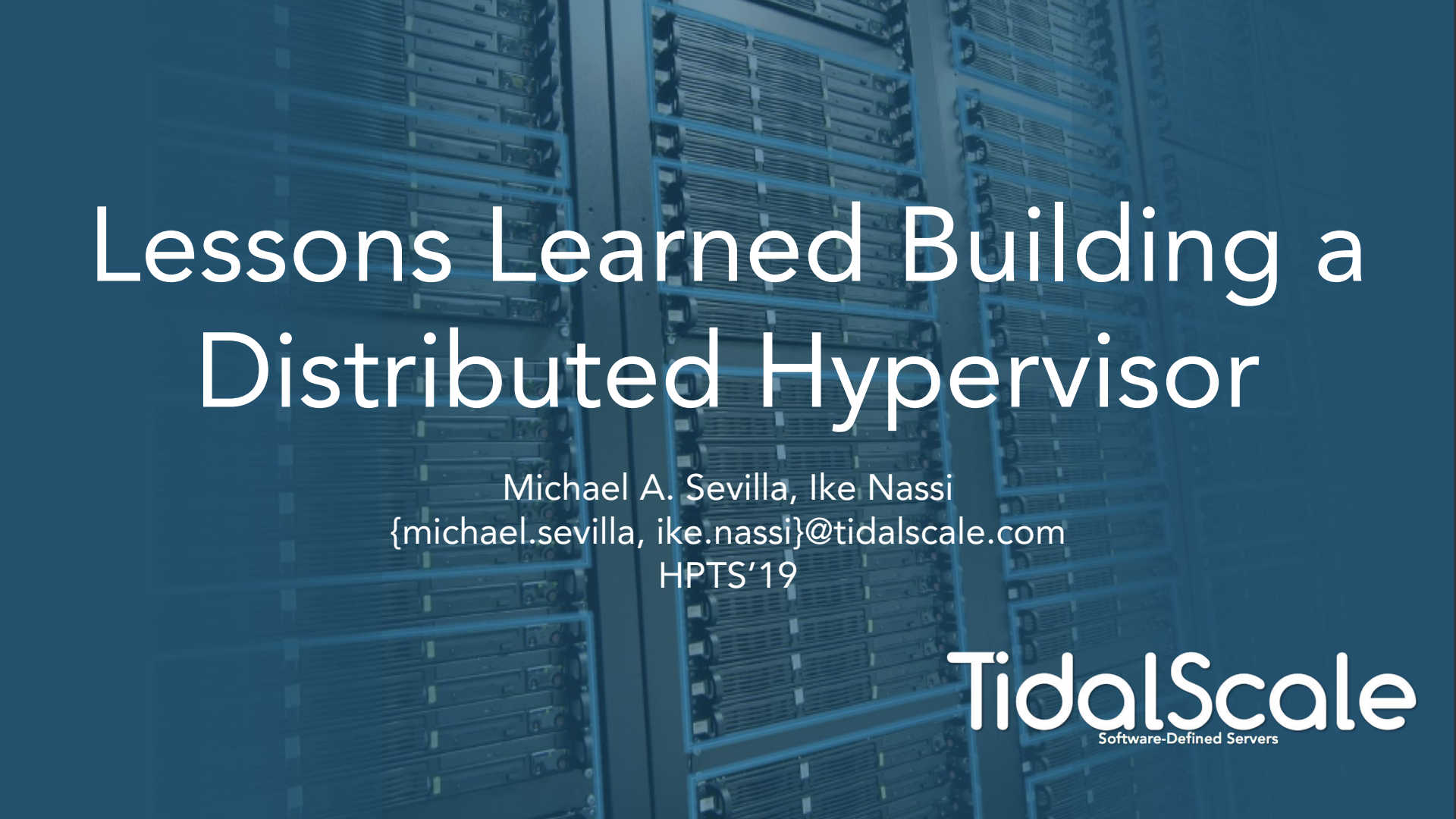
"Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults"
Clement et al.

Also, can you really just give up once you go over *k* failures?

Fault-tolerance ~~is not a technical problem~~ is an epistemological problem

Opportunities:

1. Formalize propagation of impact
2. A global forum for sev0 incidents
3. Formal methods in the wild

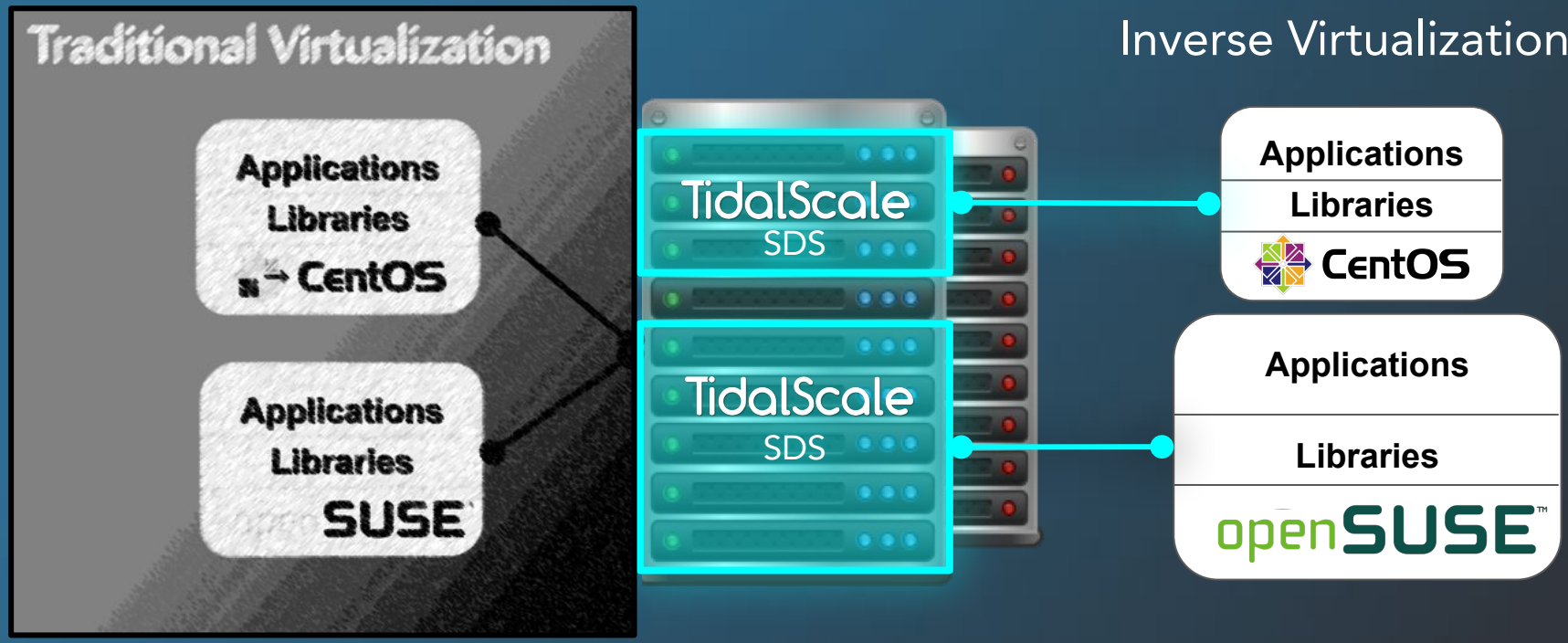


Lessons Learned Building a Distributed Hypervisor

Michael A. Sevilla, Ike Nassi
{michael.sevilla, ike.nassi}@tidalscale.com
HPTS'19

TidalScale
Software-Defined Servers

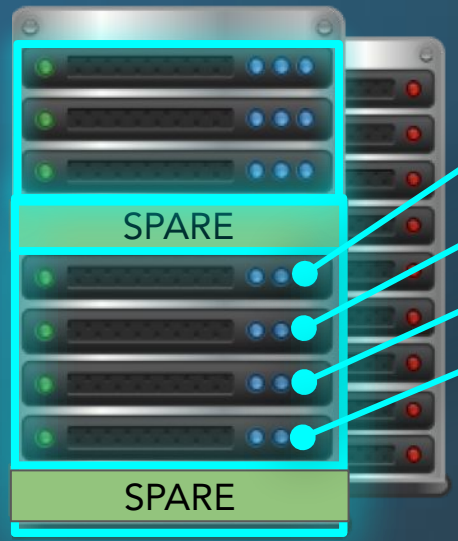
TidalScale: Software-Defined Servers (SDS)



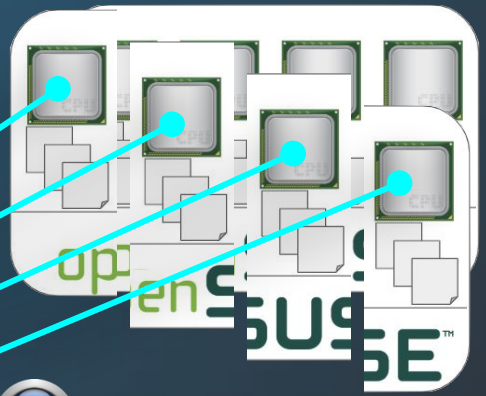
Michael Sevilla, Ike Nassi HPTS'19 {michael.sevilla, ike.nassi}@tidalscale.com

~~Challenges:~~ New, Exciting Opportunities

1. Detect failure
2. Evict resources from server
3. Repair server and re-introduce



New Model for Distribution



- > Nodes = >… Reliable
- 0 downtime (SLAs, upgrades, etc)
- IO: granularity of compute
- DB: NUMA compatibility

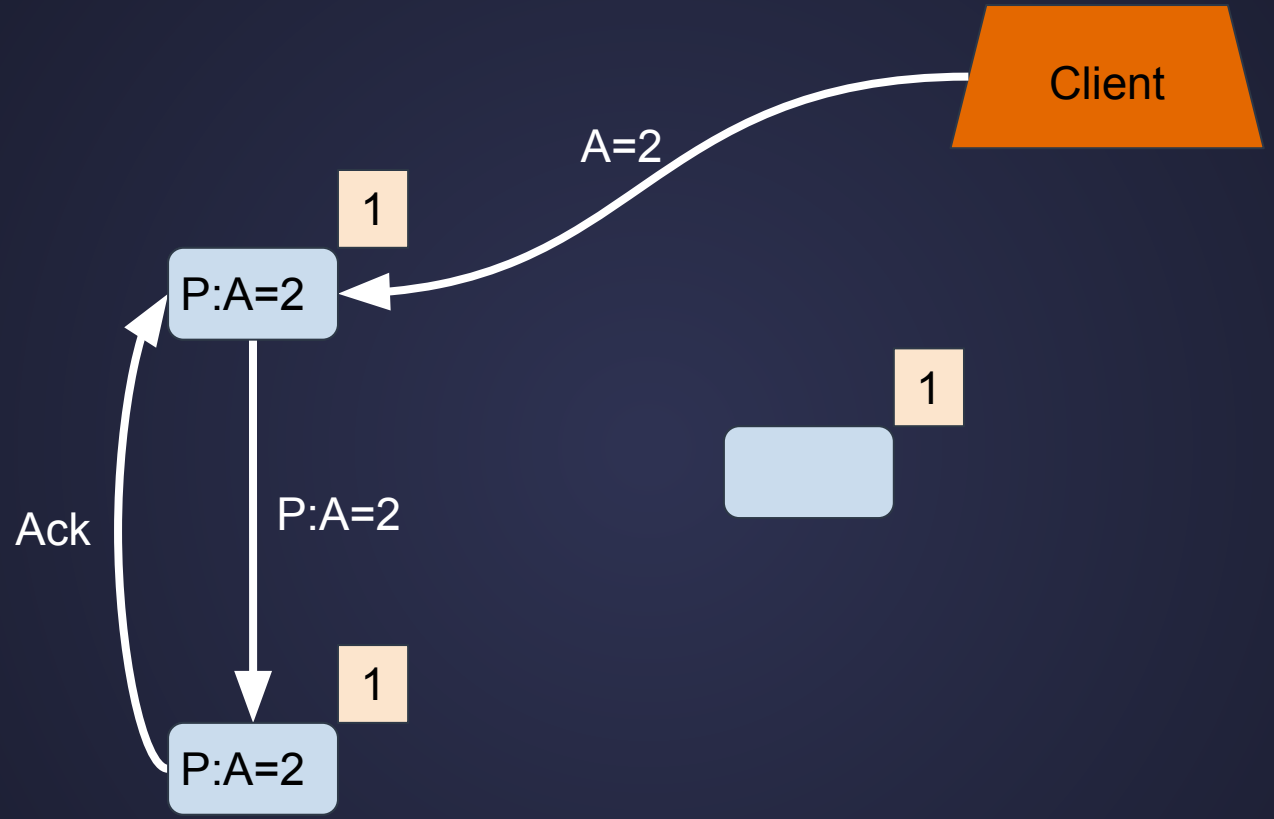
Consensus

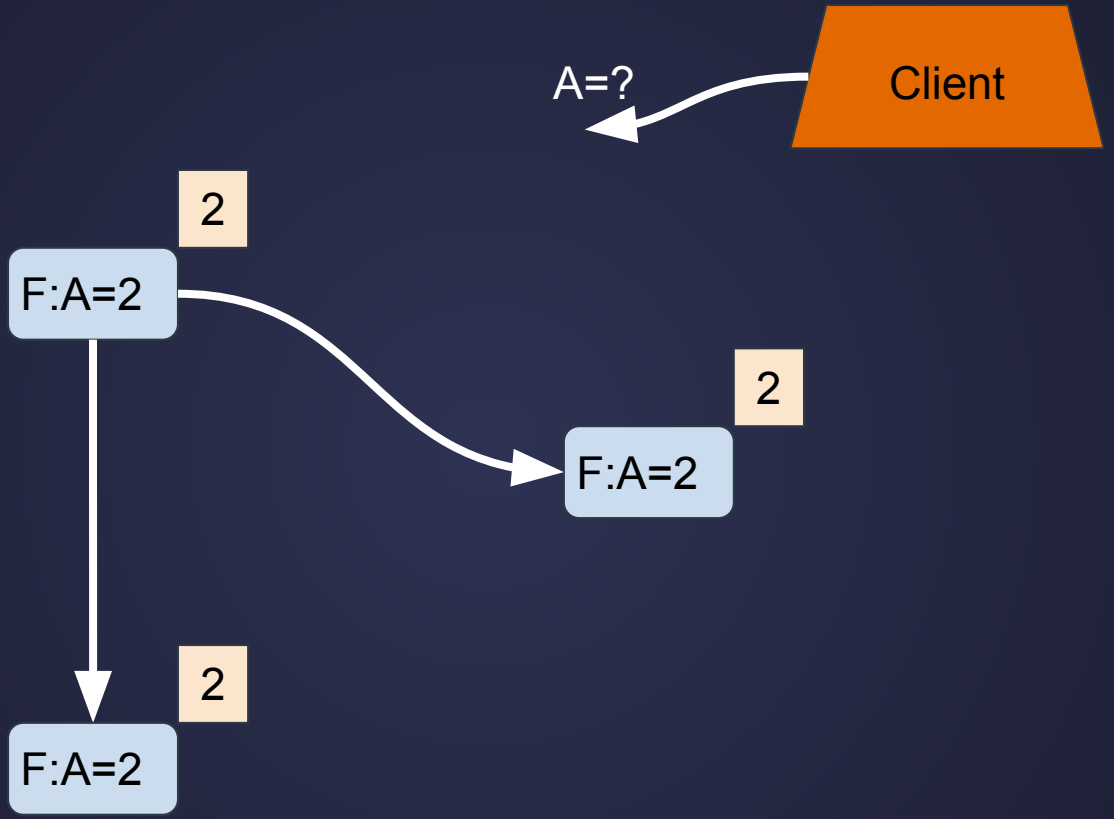
vs

Consistency

Sugu Sougoumarane

Co-creator Vitess, CTO @ PlanetScale


Client
A=2
1
P:A=2
1
Ack
P:A=2
1
P:A=2


A=?
Client
2
F:A=2
2
F:A=2
2
F:A=2

Our Options?

Quorum Read

Read from Leader

Wait till update

Use Consensus for

Durability

Preventing Divergence

High Availability

Consistency

“Transactions” in a Microservices World: The Saga Continues …

Pranta Das
Founder & CEO
Das Coders

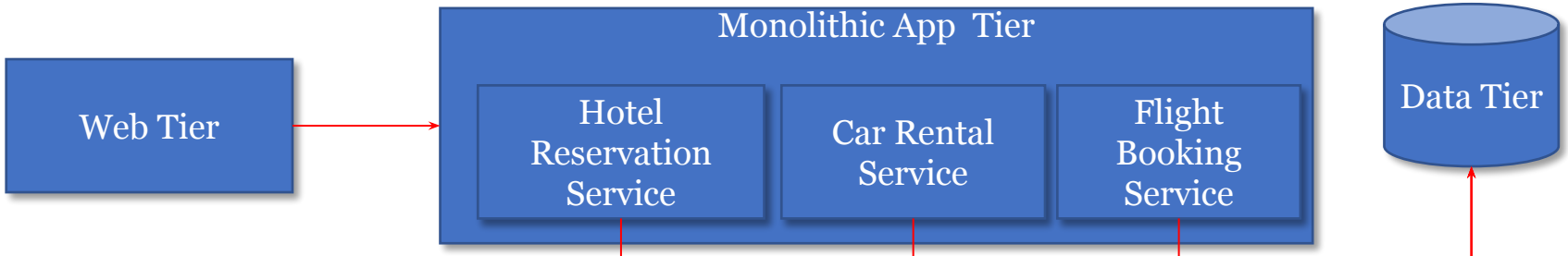
18th International Workshop on High Performance Transaction Systems (HPTS)
November 3-6, 2019

History of Sagas

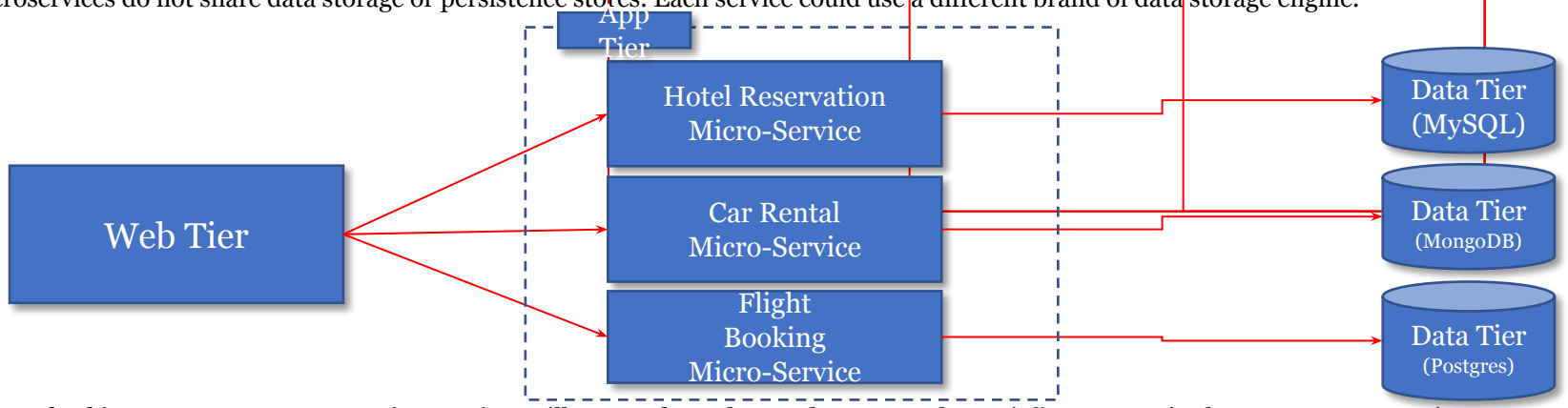
- First proposed in a research paper titled “Sagas” by *Hector Garcia-Molina and Kenneth Salem*, Dept. of Computer Science at Princeton University, submitted on 7, January 1987. Also appeared in SIGMOD '87 Proceedings of the 1987 ACM SIGMOD international conference on Management of data (Pages 249-259) in San Francisco, California, USA — May 27 - 29, 1987:
  - https://dl.acm.org/citation.cfm?id=38742
- This advanced transaction model became popular in Enterprise Application Integration (EAI) systems, which had Long-Lived Transactions (LLTs), in the late 1990’s and early 2000’s.
- I had written a paper on a hybrid model called CHAT (CrossWorlds Hybrid Asynchronous Transactions), that borrowed concepts from the Saga model and the ConTract model, 16 years ago at HPTS 2003:
  - https://drive.google.com/file/d/1Tm3oNmGiuf16tDhIBFTEUHDxt_GsOoFW/view

Why are Sagas making a comeback?

- As monolithic applications are getting split-up into microservices, each microservice is responsible for making its own independent data storage and persistence decisions.
- For example, a standard 3 tier application in single monolithic process, such as the one, shown below, uses a single monolithic data store engine:

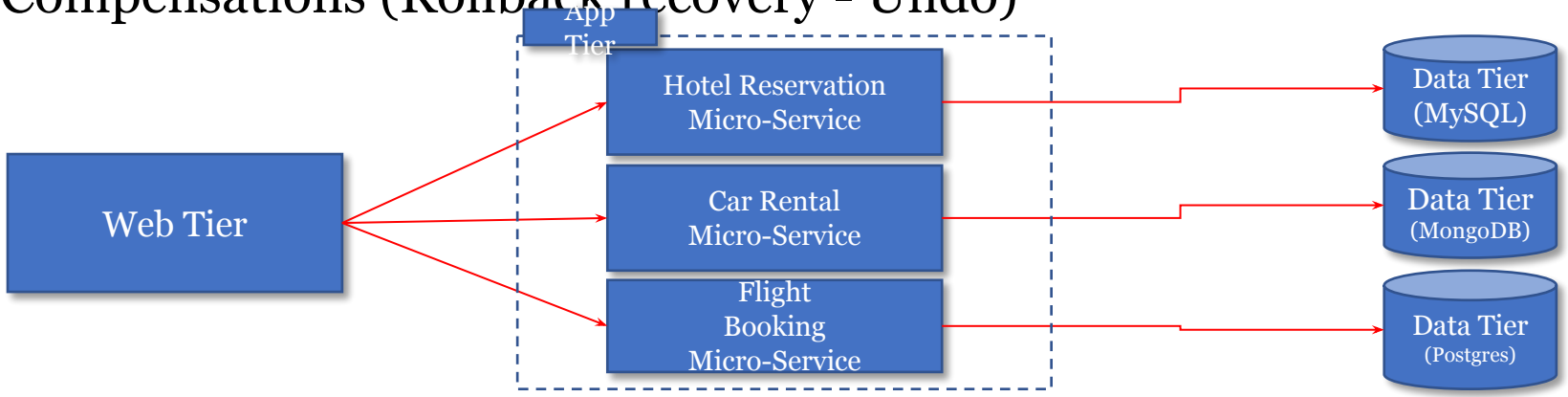


- Microservices do not share data storage or persistence stores. Each service could use a different brand of data storage engine.



- So standard in-process ACID across microservices will not work. And 2-PC does not scale – $O(n^2)$ messages in the worst case scenario.

Each Saga has a bunch of Sub-Transactions and associated Compensations (Rollback recovery - Undo)



Backward Recovery (Requires Compensations to be Idempotent)

:Begin-Saga

Sub-Transaction-1: Make-hotel-reservation

If failed – go to :Abort-Saga (since Compensation Stack is Empty)

Otherwise Push Make-hotel-reservation onto Compensation-Stack

Sub-Transaction-2: Rent-a-car

If failed – Pop Compensation Stack – Undo Make-hotel-reservation and go to :Abort-Saga

Otherwise Push Rent-a-car onto Compensation-Stack

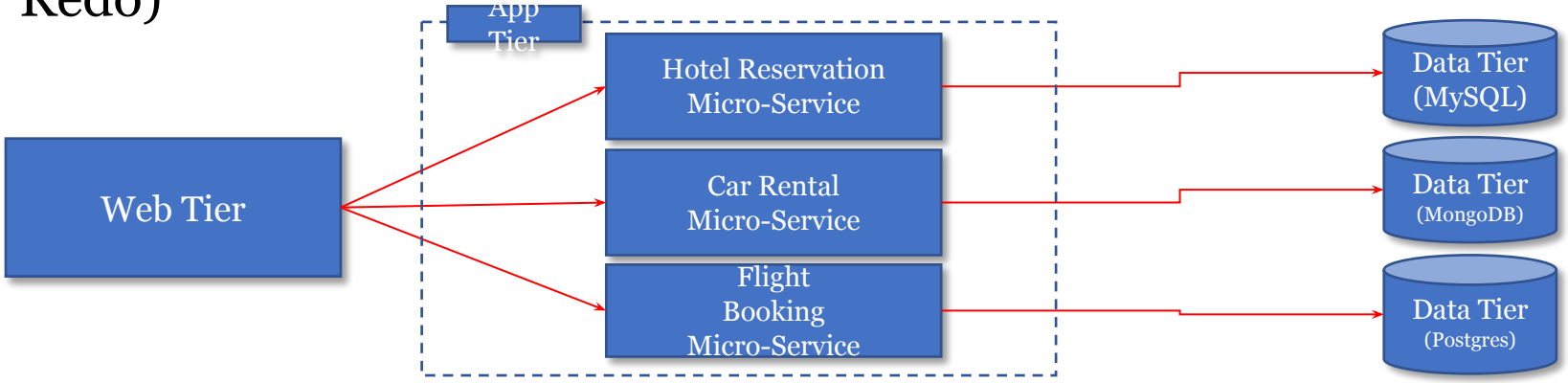
Sub-Transaction-3: Book-a-flight

If failed – Pop Compensation Stack twice – Undo Rent-a-car, Undo Make-hotel-reservation and go to :Abort-Saga

:End-Saga-Success

:Abort-Saga

Each Saga has a bunch of Sub-Transactions (Roll-Forward Recovery - Redo)



Forward Recovery (Requires Sub-Transactions to be Idempotent)

:Begin-Saga

Sub-Transaction-1: Make-hotel-reservation

If failed – retry "indefinitely" with exponential-backoff until it succeeds.

Sub-Transaction-2: Rent-a-car

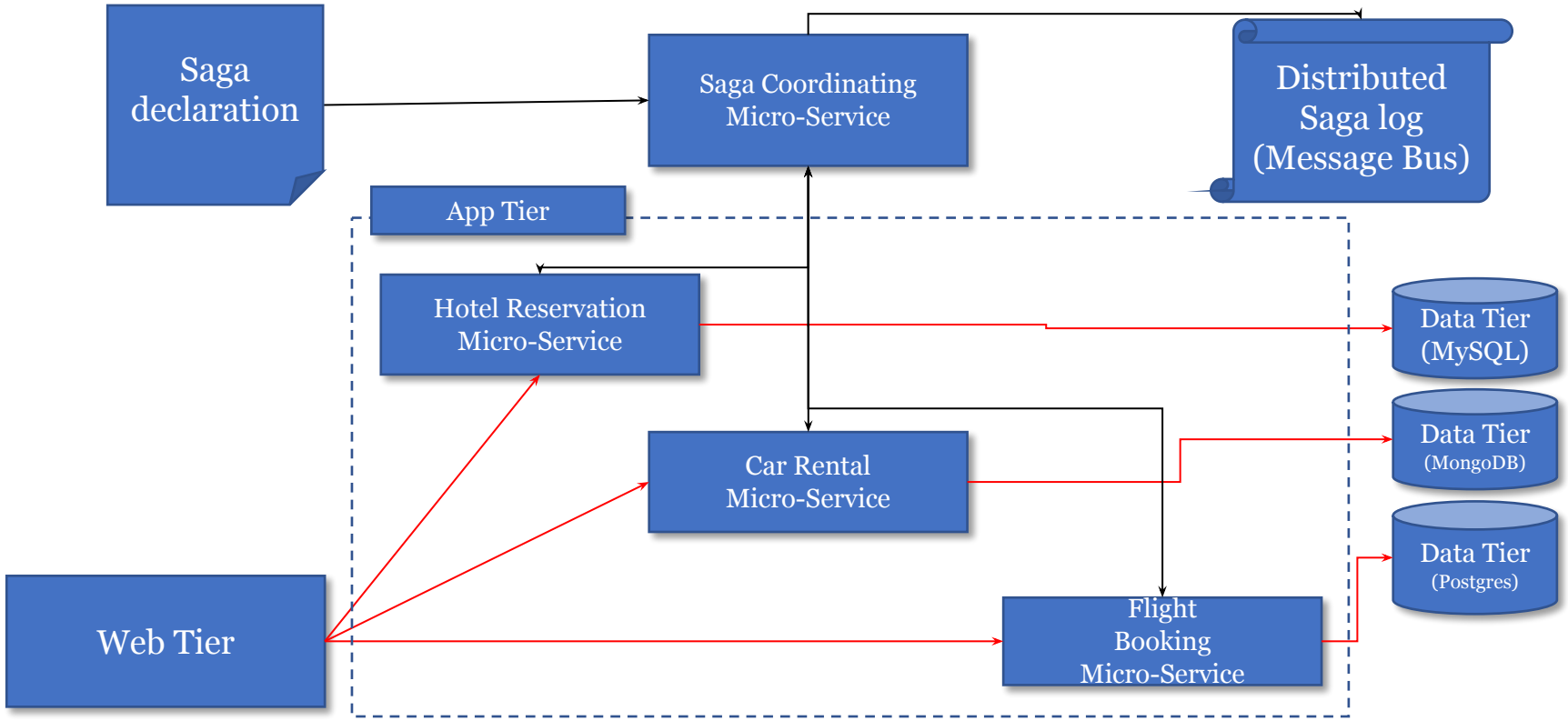
If failed – retry "indefinitely" with exponential-backoff until it succeeds.

Sub-Transaction-3: Book-a-flight

If failed – retry "indefinitely" with exponential-backoff until it succeeds.

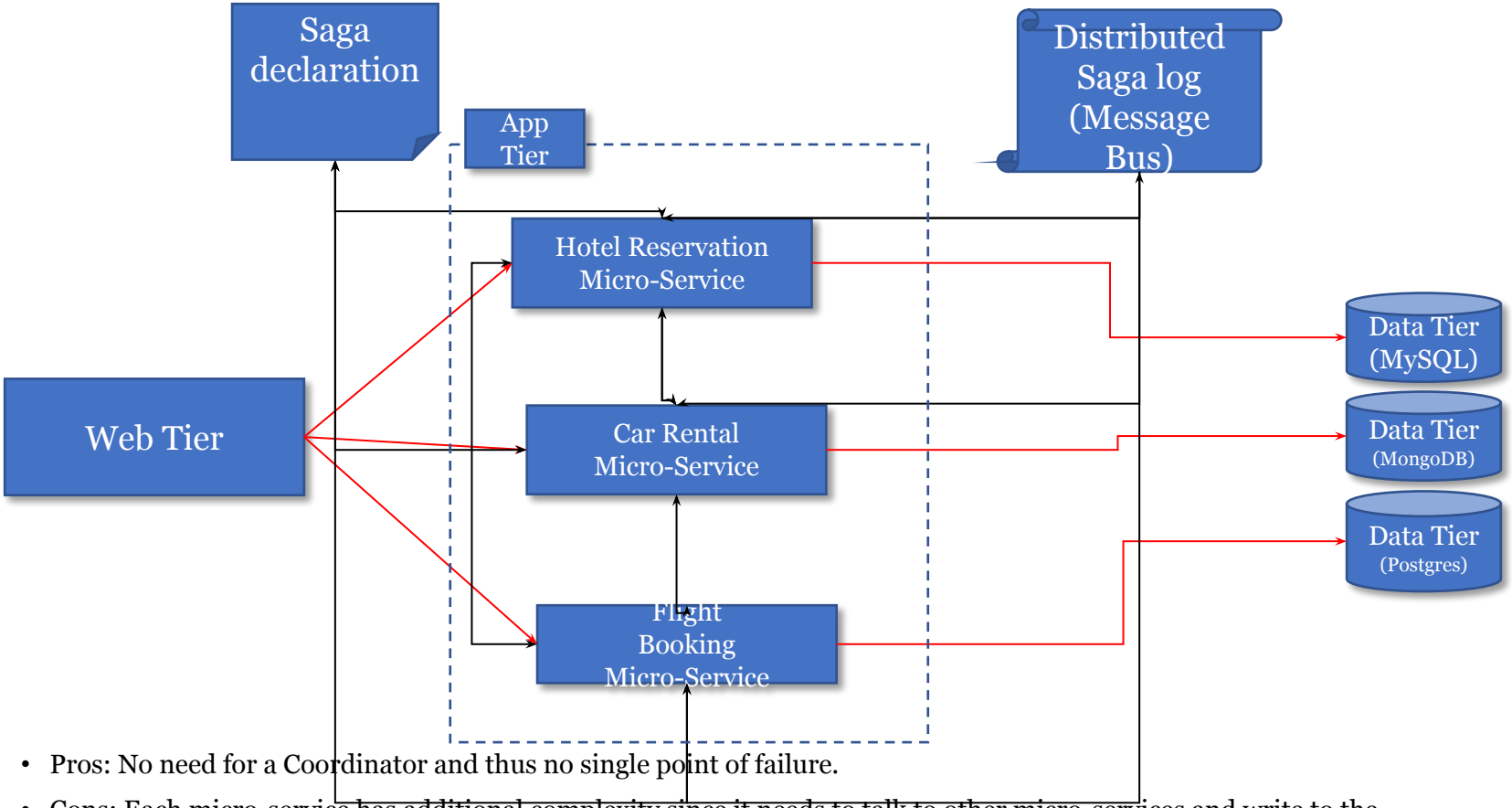
:End-Saga-Success

Pattern-1: Centrally Coordinated Saga Execution - A specialized Saga microservice coordinates the Saga and talks to all participant microservices



- Pros: Individual micro-services need not have to deal with Saga execution.
- Cons: Coordinator micro-service (albeit stateless) becomes a single point of failure

Pattern-2: Distributed Saga Execution – Each microservice runs it's part of the Saga and communicates with its peers



- Pros: No need for a Coordinator and thus no single point of failure.
- Cons: Each micro-service has additional complexity since it needs to talk to other micro-services and write to the distributed saga log

What if a compensation fails?

- If a compensation fails to execute, then the Saga enters a Heuristic state.
- In such cases, sometimes the only way to fix the Saga to bring it to a consistent state may be through manual repair. This may be via a phone-call or email to the system support staff to fix the problem.

Other recent literature on this subject:

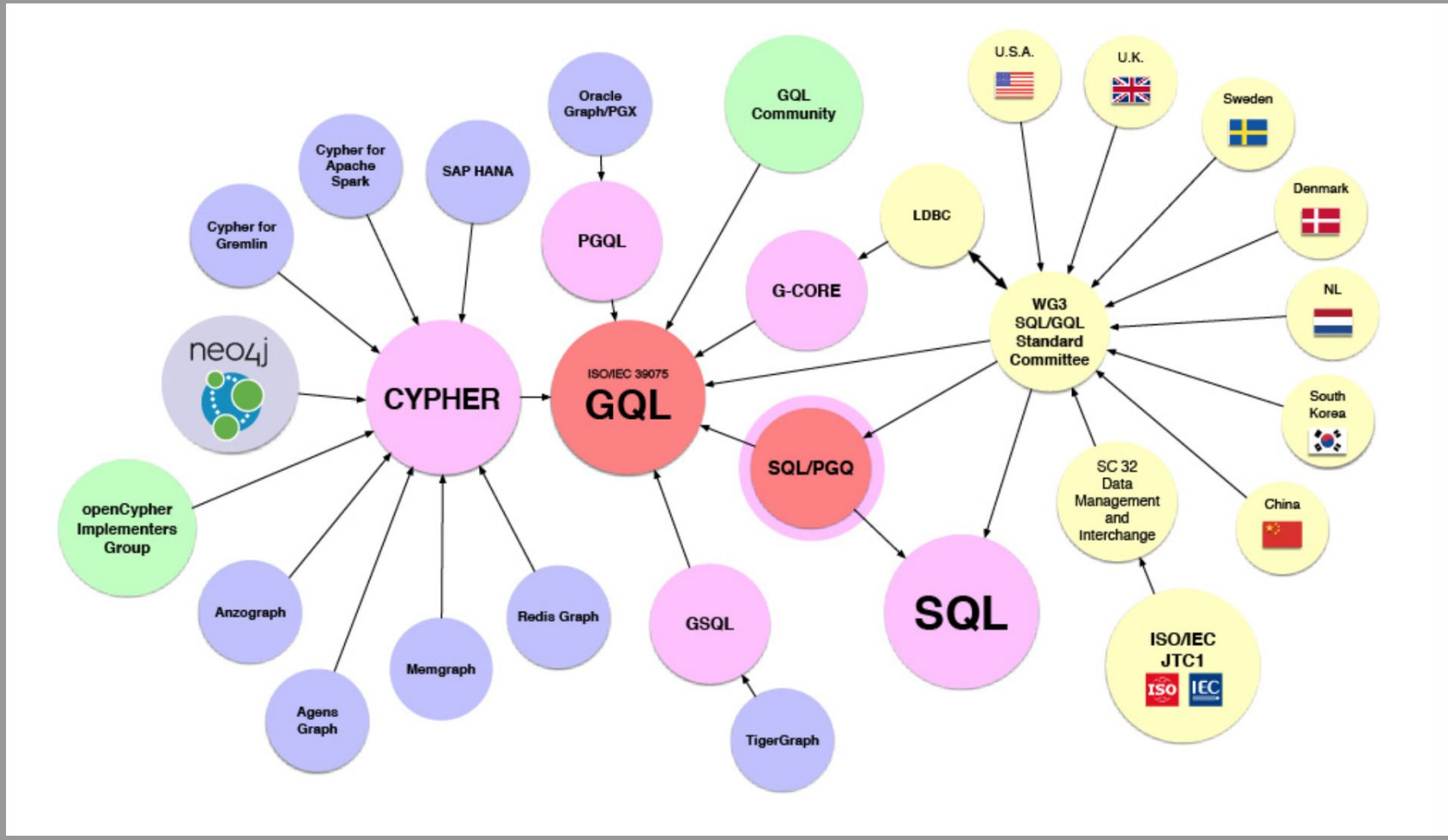
- Chris Richardson's article on Microservices patterns: https://microservices.io/patterns/data/saga.html
- Microsoft Azure Architecture Patterns: https://docs.microsoft.com/en-us/azure/architecture/patterns/compensating-transaction
- Caitie McCaffrey's talk at JonTheBeach @ JontheBeach (2017)

https://www.slideshare.net/JontheBeach/distributed-sagas-a-protocol-for-coordinating-microservices

GQL Graph Query Language
A new ISO/IEC standard project

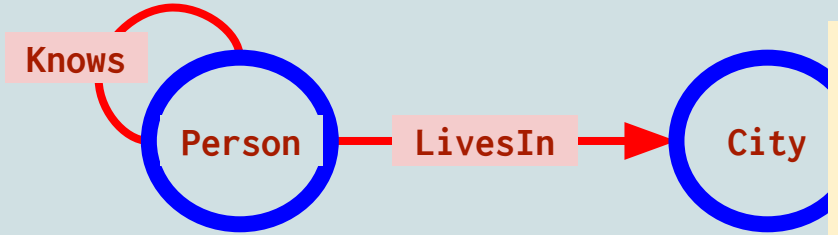
Alastair Green, Neo4j Query Languages Team
Vice-chair Linked Data Benchmark Council

HPTS 2019, 4 November


Oracle Graph/PGX
GQL Community
U.S.A.
U.K.
Sweden
Cypher for Apache Spark
SAP HANA
Denmark
Cypher for Gremlin
PGQL
LDBC
G-CORE
NL
neo4j
CYPHER
ISO/IEC 39075
GQL
WG3 SQL/GQL Standard Committee
South Korea
SQL/PGQ
SC 32 Data Management and Interchange
China
openCypher Implementers Group
Anzograph
Redis Graph
GSQL
SQL
ISO/IEC JTC1
Memgraph
Agens Graph
TigerGraph

```
CREATE GRAPH SocialNetwork (
    (Person {name STRING, dob DATE}),
    (City {name STRING}),

    (Person)-[LivesIn]->(City),
    (Person)=[Knows]=(Person)
)
```




```
CREATE VIEW Cities {
    FROM SocialNetwork
    MATCH (tail)-[edge]->(city::City)
    PROJECT GRAPH tail, edge, city
}

FROM Cities
MATCH ()-[connections]->()
PROJECT DISTINCT type(connections)
```

ISO/IEC **GQL standard**

September 2019

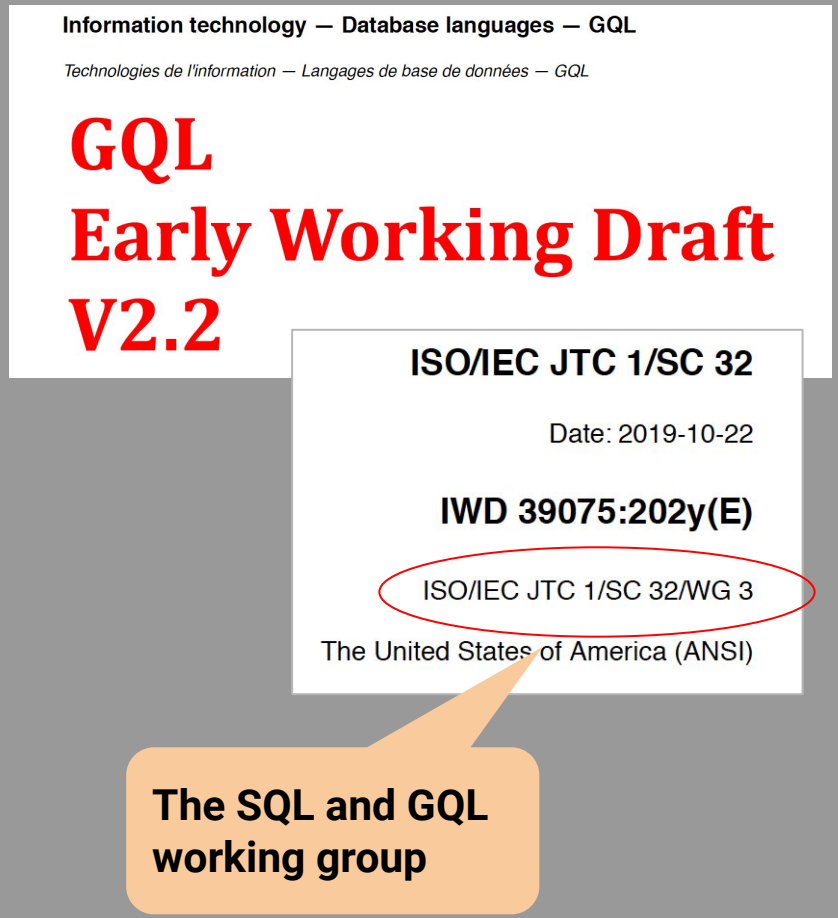
Ballot on new project proposal closes

10 countries for, 4 abstain, 1 against

7 countries volunteer experts including U.S.A., China, U.K.

Graph Query Language **GQL**

First international standard Database Languages project since SQL in 1987



Why GQL isn’t SQL reshuffled

Graph queries can examine structure, without knowing types or values

```
FROM Cities
MATCH ()-[connections]->()
PROJECT DISTINCT type(connections)
```

Don Chamberlin said that SQL pushed 100s of lines of CODASYL into a few lines

“Cypher: the best way to state a join”

```
FROM Cities
MATCH (p:Person)-[LivesIn]->(c:City WHERE c.name = “Berlin”),
      (p)=[Knows]=(friends:Person)
PROJECT p, friends
```

GQL is *a* graph language: not *the* graph language

GQL is a **declarative query language** that understands the property graph data model.

The body of a graph procedure can be written in GQL ...

There can be other graph languages: procedural, traversal, network algos ...

A graph program is made up of language modules that share types and procedure signatures

Learning from the weaknesses, actual and perceived, of SQL

Why SQL doesn’t suck

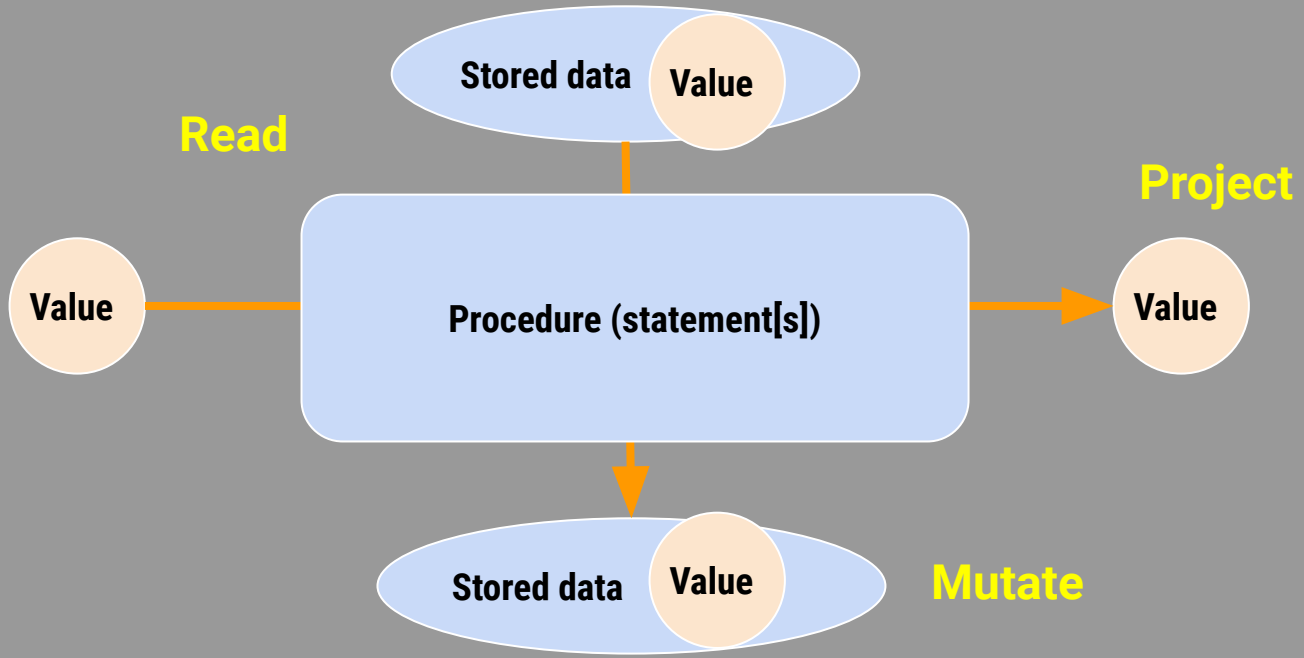
SQL SELECT is a function over a table, returning a table

Composition (closure over tables) enables Spark’s mixture of SQL clauses and user code. A DataFrame is a table, and everything in Spark is a chain of functions transforming tables to tables, using relational algebra and SQL syntactic units

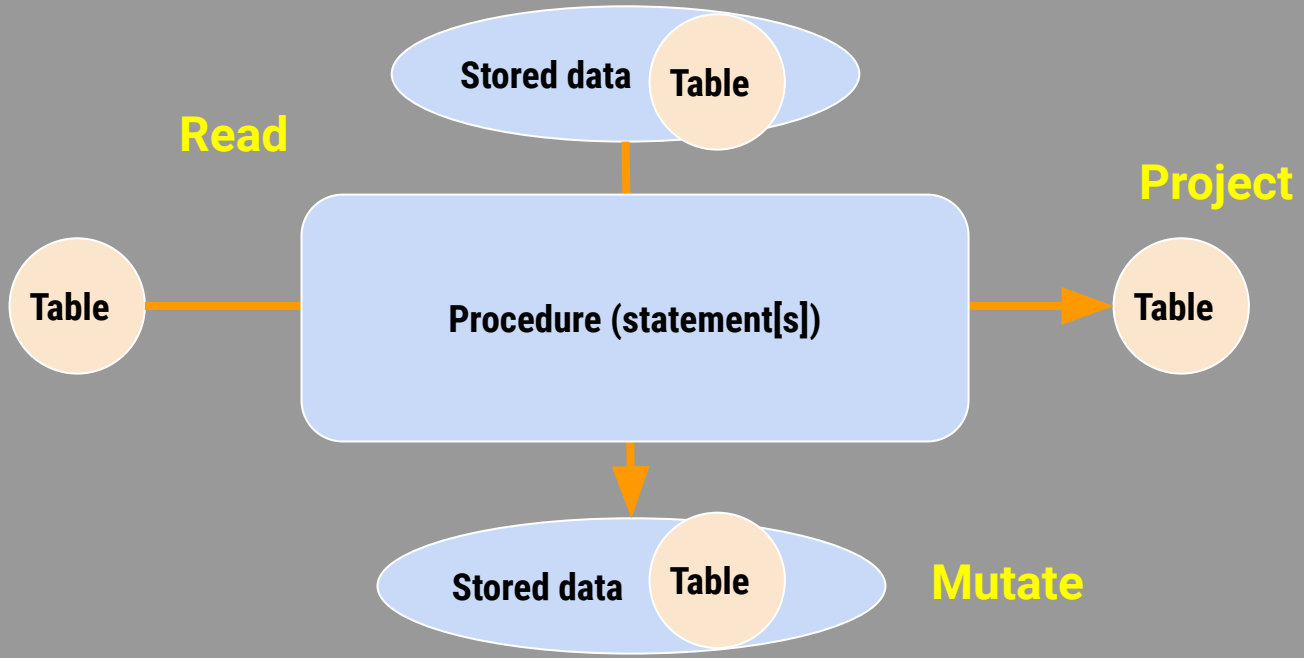
Property graph languages started out as Graph ➜Table projections

But they need to grow up, and allow Graph ➜Graph projections

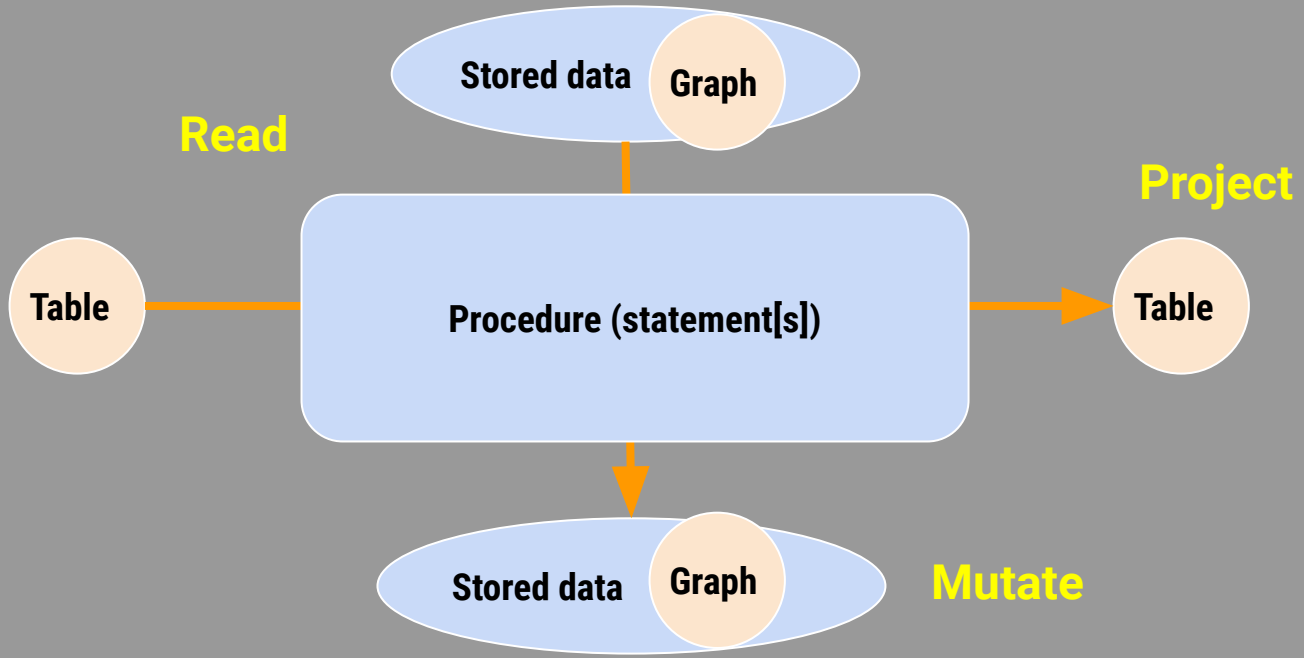
DB query language model



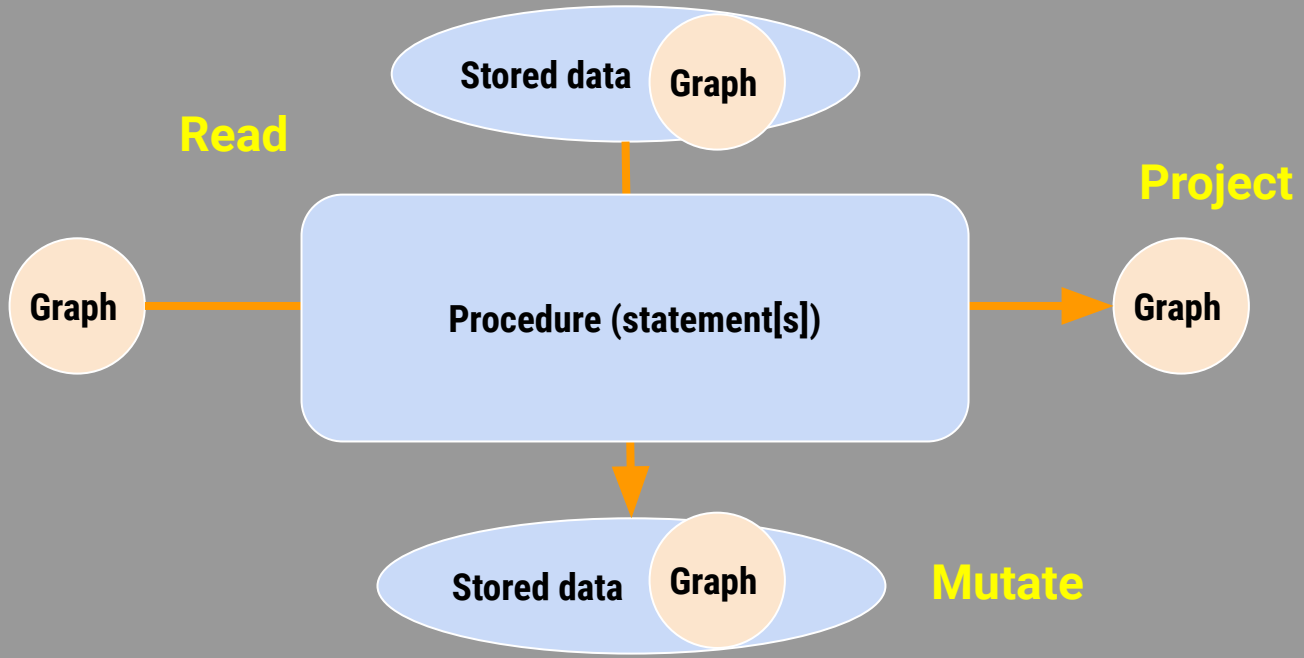
SQL query language model



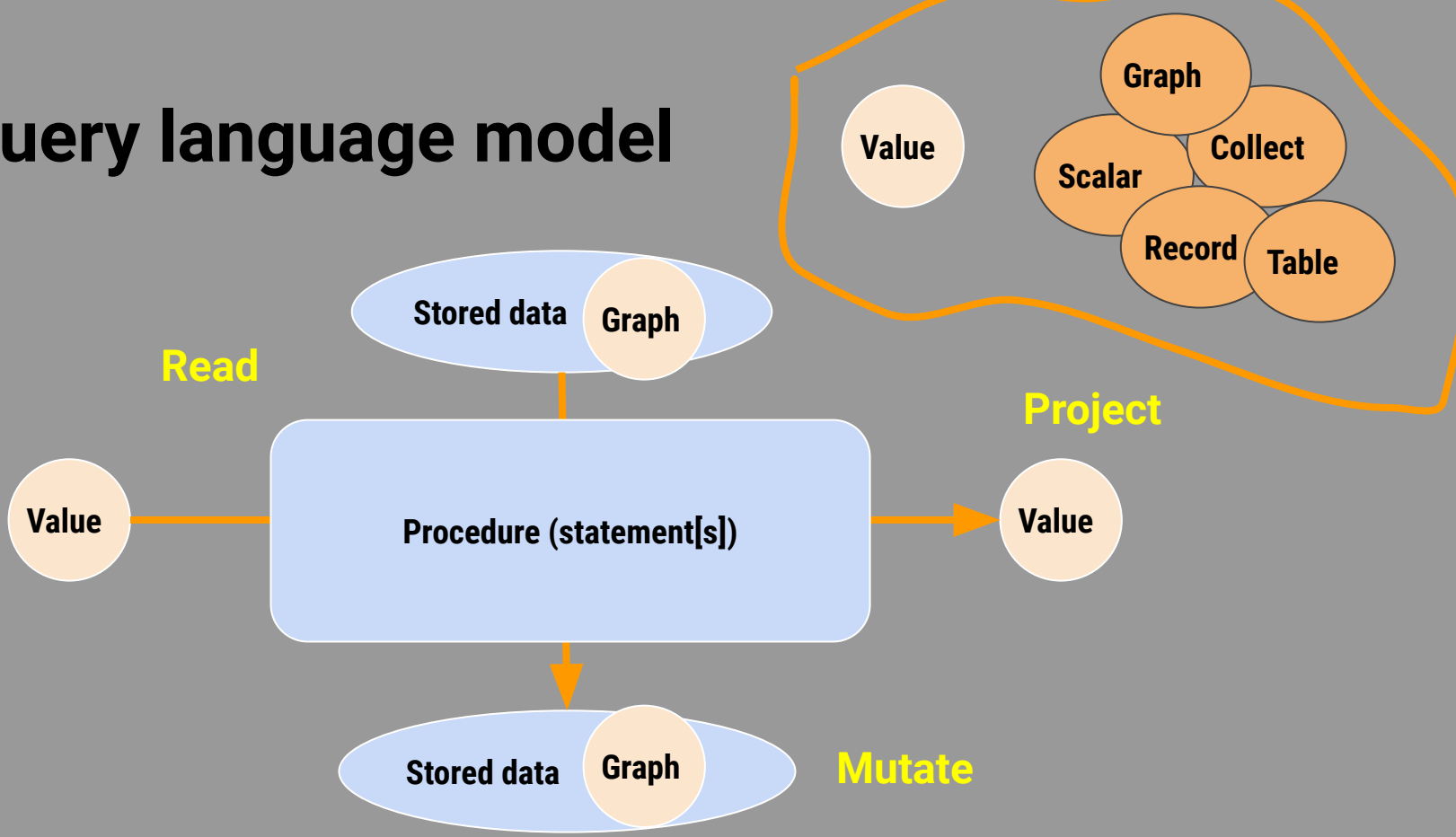
Cypher query language model



Pure graph query language (G-CORE) model



GQL query language model



GQL will codify the state of the art in graph data

- Superset of industrial property graph data models
- Full CRUD for databases *and* read/project for analytics engines
- Multi-statement procedures
- Nested and cascading procedure composition
- Table projections and graph projection/query composition
- RPQs as well as fixed patterns, for existence and data
- “Closed” schema for graphs
- Graph catalog with named graphs and named queries (views)
- Transaction demarcation support and atomic transactions
- Client-service sessions
- Access control: NOT YET

	GQL 1.0	SQL/PGQ 202x	RDF/SPARQL 1.1
Node and edge properties	✓	✓	~ nodes only
Data Query: fixed pattern	✓	✓	✓
Existential Query: RPQ	✓	✓	✓
Data Query: RPQ	✓	✓	~ endpoints only
Table projection	✓	✓	✓
Graph projection	✓		✓
Insert, update, delete	✓		✓
Named graphs	✓	✓	✓
Read-only graph views	✓		
Updatable views	~ simple only		
Omnigraphs	✓		
Catalog of graphs	✓	✓	
Graph schema	✓		✓
Transaction demarcation	✓	✓	

Thank you!