# Fantastic Data Consistency Techniques and Where to Find Them

Gordon Hutchison
IBM Development
OpenLiberty
@gordhut

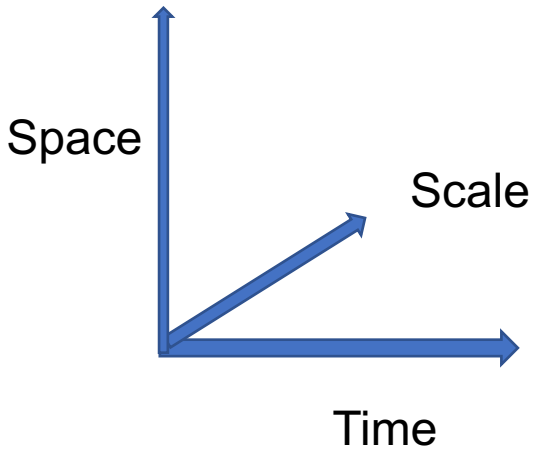
September 2019

- This session covers data consistency available today for distributed microservices.
- Globally consistent data is no longer a safe assumption, and what will ultimately replace it is not yet resolved. Are the techniques that are becoming popular now such as eventual consistency, sagas, and CQRS/event sourcing frameworks here to stay? Or will they be displaced, in turn, by techniques emerging in multiple-editor shared document systems such as conflict-free replicated data types (CRDTs)?
- If you are wondering how to cope without the ACID properties of transactions in distributed data as you move to microservices. This session will inform you of the palette of solutions and libraries currently available.

What forms of consistency do we deal with today?

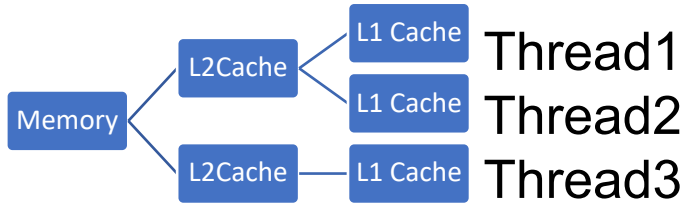
Open Liberty

# Consistency




## Local Scale

- Cache-Memory
  - Weakly Consistency
    - POSIX mutex
  - Across Java Threads
    - Volatile/Atomic
- Persistent Storage
  - Recoverable results
  - Repeatable Reads/Cursor Stability,

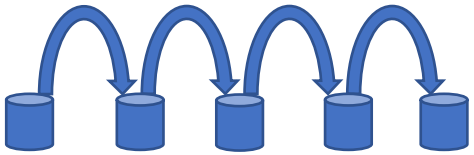



## Across Time

Always consistent view  Databases -

ACID

Eventually Consistent



## Across Space

Clusters – Consensus via Paxos etc,
XA – Coordinator+voting
Microservice Constellations

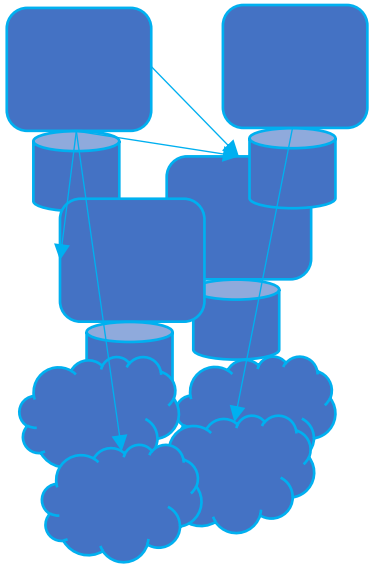
Open Liberty

# Distributed Microservices

An application composed from co-operating services.
Services may be taking part in multiple applications.

Services may be running:

On-prem
Cloud
3rd Party



Also

‘Cattle not pets’
Scale out rather than scale up
Sharding
Re-hashing
Geo-replication

All this leads to → Increased Data Distribution

Globally consistent data is no longer a safe assumption.

Open Liberty

# Why look beyond distributed transactions?

Cross server call/data flow is:

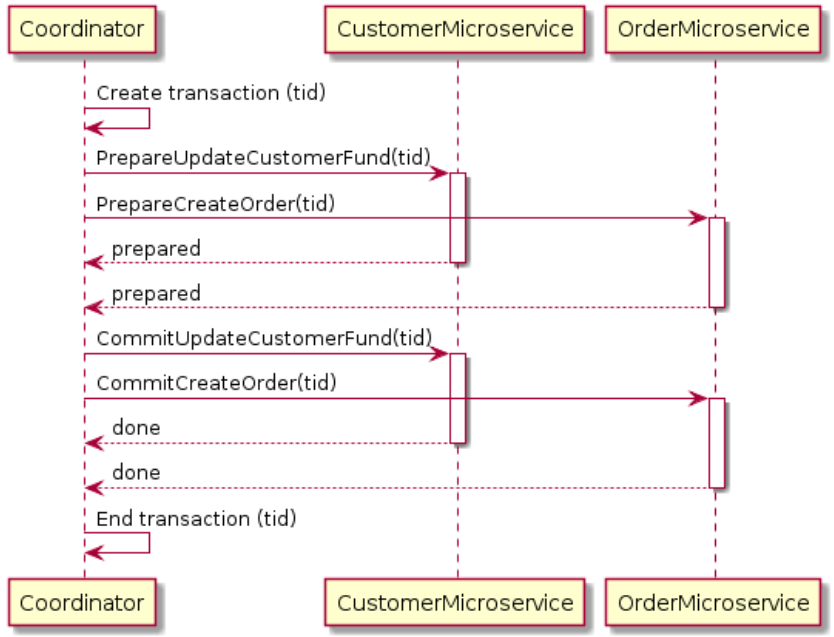
- finer grained
- more frequent
- more participants

Distributed transactions ‘hardening’ overhead more costly.

Middleware:

- multi-vendor
- rapid turnover
- polyglot-technologies

Distributed transactions/XA not supported by middleware

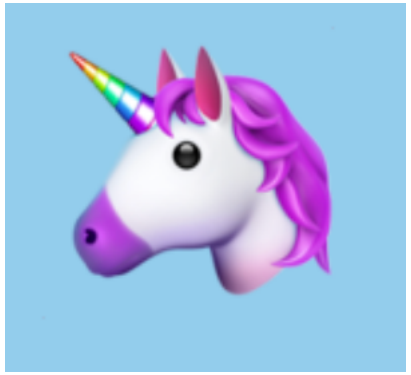



Blocking – Synchronous ‘long’ calls + locks…**blocking** increases **latency**

Design of data/lock ordering needed to avoid…**deadlock**

Open Liberty

# Consistency for Microservices



Bandwidth
‘Cattle not pets’
Polyglot software
Data as an asset
Consistency for ‘free’

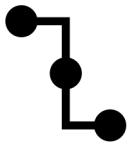
Latency
Blocking
Call outs
Server affinity
Software lock-in
Data inconsistency

Open Liberty

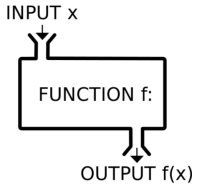
# Avoiding Contention Altogether?

Optimistic Locking

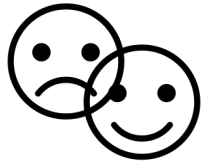
Sharding Aggregates with a Consistent Hash



Thread sharding and non-premptable tasks




Functional Programming



Actors (Akka)

# ‘Eventual Consistency’?


Funny cartoon on Eventual Consistency

https://twitter.com/mykola/status/1101337299525267457

Open Liberty

# Eventual Consistency

What can services rely on?

What has actually happened locally.

Other parties informed (events)

Reconciliation is asynchronous to write

Weaker than

linearizable (read ALL completed writes)

strict serializability (of transactions)

Commands (versus Queries)

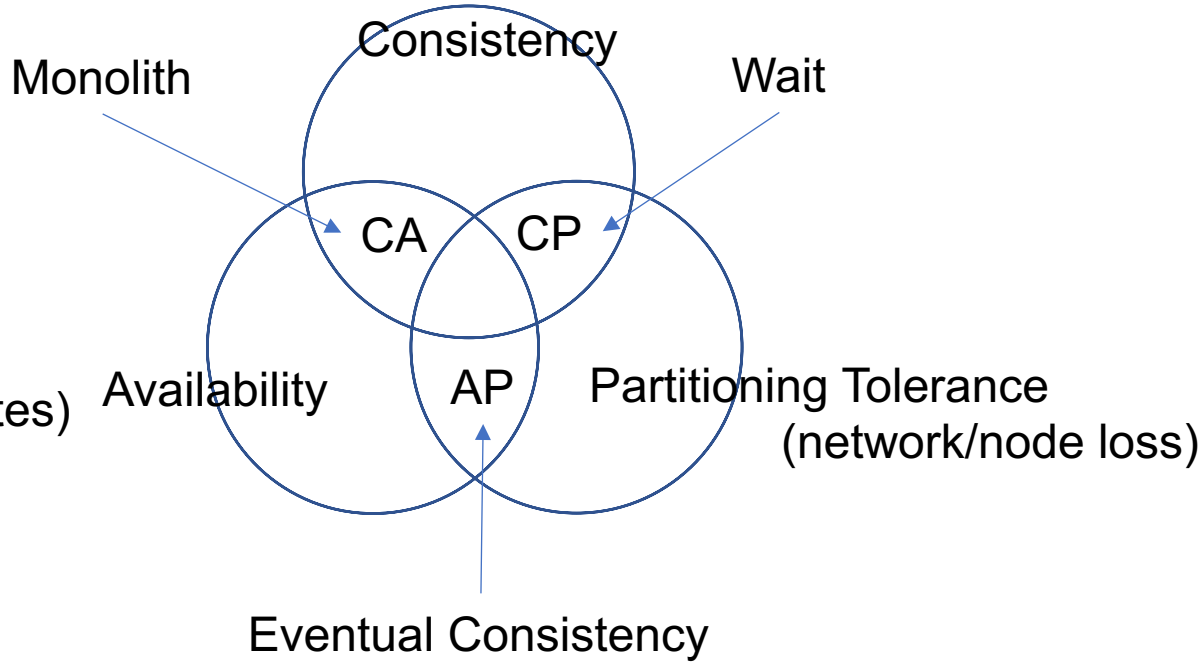
Resolve conflicts by…

Consensus (coordinator)

Backing up (sagas undo/redo)

Distributed common rules (no ‘conflicts’)

CAP Theorem



‘Sagas’?

Open Liberty

# Long Running Activities and Compensation

Sagas.

Server Local Transactional Steps
Expose Intermediate State Tentatively
Transactional outbox for event distribution
Backtrack if necessary via compensations

Sub-step-**A**tomicity,

Local-**C**onsistency

Lumpy/leaky/temporary-**I**solation

**D**urability only after final commit

LRA

Narayana

Microservices.io: saga (Eventuate)

Choreography - each local transaction publishes events that trigger other services

Orchestration - an orchestrator tells the participants what to execute

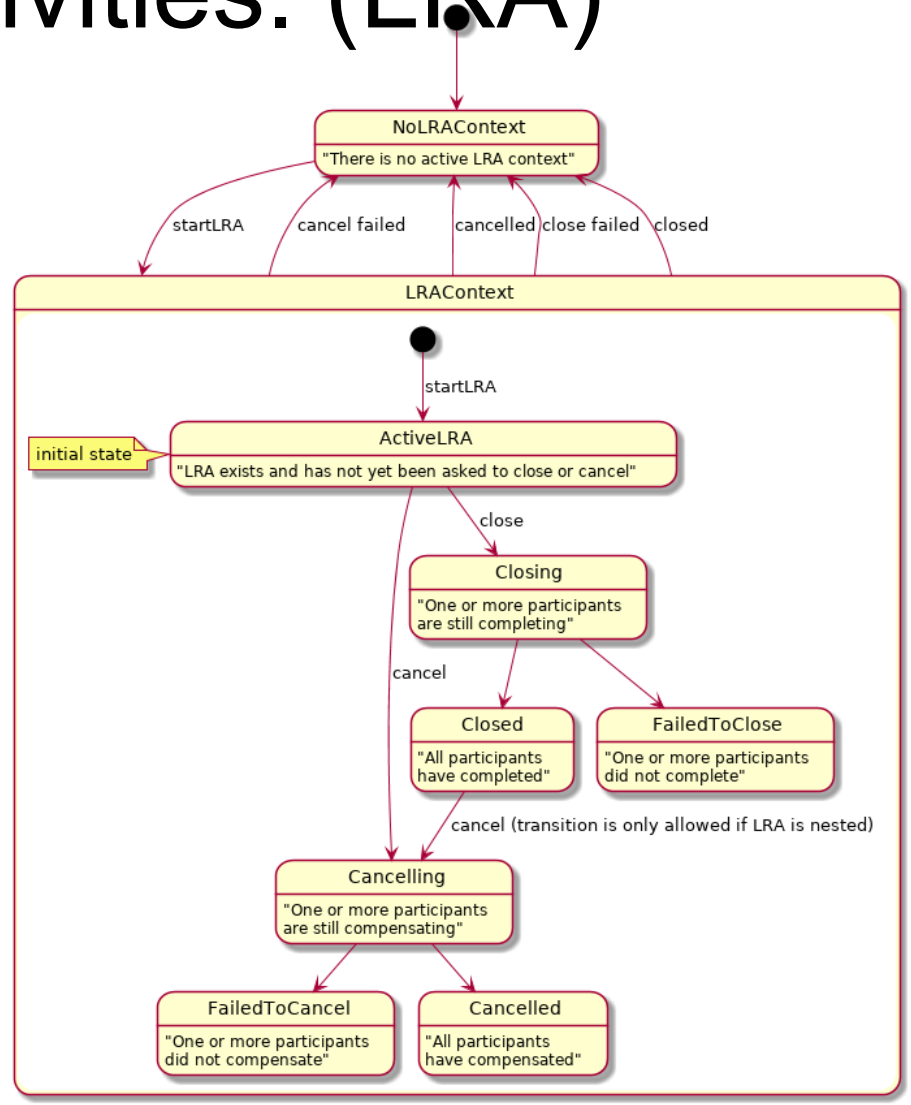
‘Sagas’?

Open Liberty

# Microprofile Long Running Activities. (LRA)

| Annotation | Description |
|---|---|
| @LRA | Controls the life cycle of an LRA. |
| @Compensate | Indicates that the method should be invoked if the LRA is cancelled. |
| @Complete | Indicates that the method should be invoked if the LRA is closed. |
| @Forget | Indicates that the method may release any resources that were allocated for this LRA. |
| @Leave | Indicates that this class is no longer interested in this LRA. |
| @Status | When the annotated method is invoked it should report the status. |

```
@Compensate @Path("/compensate")
@PUT public Response compensateWork(
            @HeaderParam(LRA_HTTP_CONTEXT_HEADER) URI lraId,
                                    String userData) {
            return Response.ok(
                        ParticipantStatus.Compensated.name()).build();
}
```



Open Liberty

# CQRS Architectures and Frameworks

(Often associated with DDD and **Event Sourcing**)

Commands - update state (handled by Aggregates)
Queries - do not update state

Any microservice can be scaled independently
>1 view of data
Routing - Axon splits out
Commands(persistent-hash),
Events(1->N),
Queries(1ofN)

Axon

Commands and Queries' state have different consistency needs.

Commands Update State
Single location– transactional
XA
or eventually consistent event sourced

Open Liberty

# Event Sourcing

All (stimuli of) changes to state captured via an ‘event’

Events are immutable

Events are logged/stored
indexed by time, consistency points

State of system can be arrived at solely by processing each event.

Events can be resent and rerouted
(cf reactive manifesto)

Groups of events sent or not sent (‘transactional’)
Event logs can be augmented by snapshots.


Kafka
Apache Pulsar
Axon Event Bus/Store

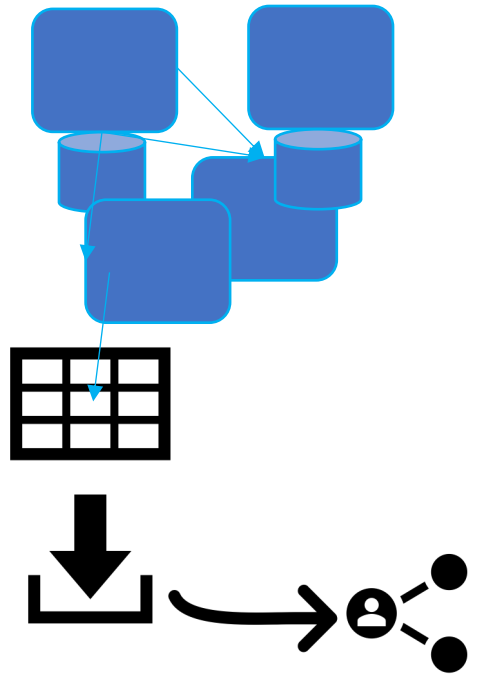
Eventuate
EventStore
DB2 Event Store
(Cloud vendor equivalents)

Open Liberty

# Change Data Capture (CDC)

Work from ‘underneath’ the database

Extracts changes to data from database logs/journals.
Whitelist-Publish → events → Subscribers
Change Events sent reliably over, for example, Kafka
Can be preceded by opening a Cursor over a table (timestamp )
Once cursor’s data is transmitted
Change Events in log/journal from timestamp are forwarded



Database vendor solutions
( e.g. Oracle CDC/GoldenGate, IBM Infosphere, Azure )

Debezium

Open Liberty

# CALM Conjecture

**Consistency And Logical Monotonicity (CALM).**

*A program has an eventually consistent, coordination-free execution strategy if and only if it is expressible in monotonic datalog.*
http://db.cs.berkeley.edu/papers/cidr11-bloom.pdf

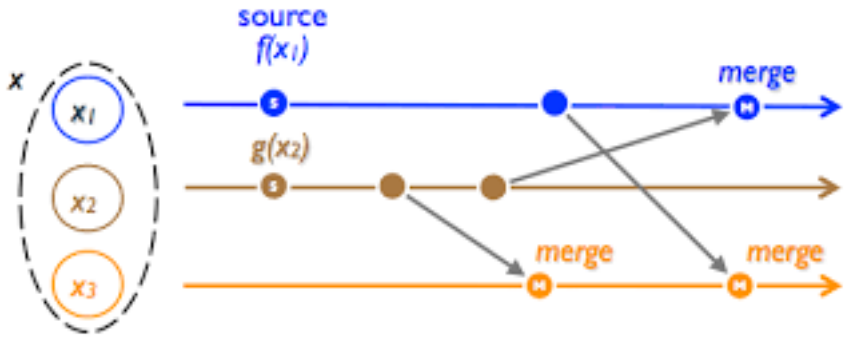
(Datalog is a Prolog like language for describing data operations
https://en.wikipedia.org/wiki/Datalog )

Open Liberty

# Conflict-free Replicated Data Type

Fundamental Data Type:
- built up from basic data structures:
  - flag, register, set, list, graph, tree, document, json

Forms of replication:
- a local replica (work offline and resync later)
- multiple editors (e.g. Boxnote, GoogleDoc )
- multiple copies/devices ( clustered state )




Conflict Free:
- ‘Collisions’ still exist but…
- A fixed set of rules exist for resolving any collisions
  - without the need for consensus resolution.
    - e.g. Timestamped last writer wins
    - GPS/Atomic clocks, UUIDs
- Convergence…towards **a** result
  - not necessarily the exact result any 1 actor wants!

**Domain level resolution still sometimes needed!**

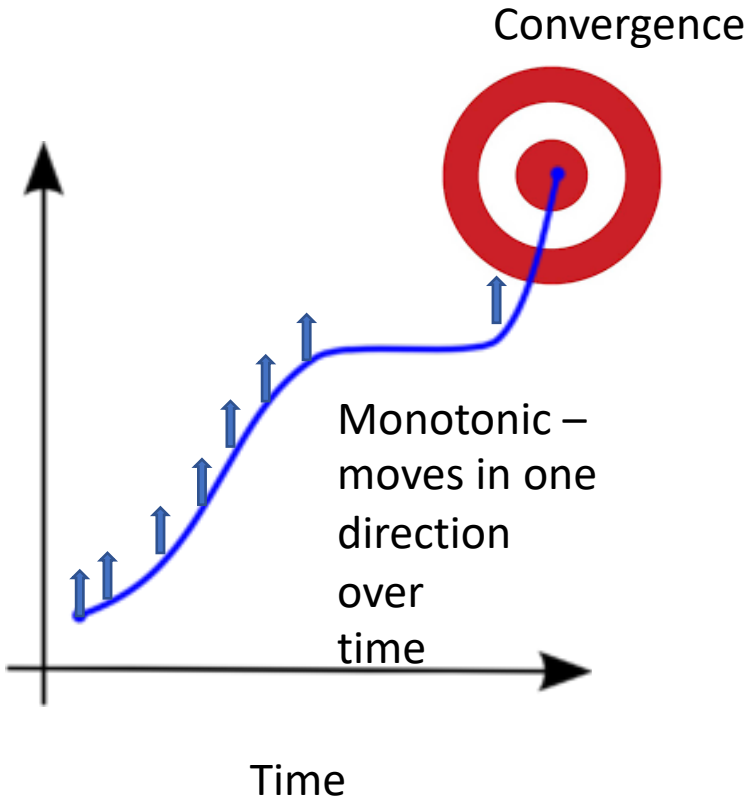
Open Liberty

# CRDTs continued…

…convergence

Data structures progress ‘monotonically’ towards convergence:

**Operations** ‘concurrent’ ones are commutative
(can arrive in different orders at different sites)
or…
**State** forms a ‘semi-lattice’
Internal data-structure to support this, indexing
Retired UUIDs (Tombstones) etc.
A ‘view’ function separates internal/external view




Causal Consistency
Born->Live->Die
Operations ‘pend’ until their prereq data/ops’ UUIDs turn up

Array [X,Y,Z] represented by [X<-Y<-Z] – ”insert after” operations

Open Liberty

# Interesting Commercial Libraries

(Not a personal or professional recommendation for any listing)

CDC
- Debezium

CQRS
- Axon
- Eventuate

CRDTs
- Lightbend
  - Akka distributed data
  - “Stateful Serverless”: CloudState
- Amazon Dynamo/Apache Cassandra
- Redis Enterprise - https://redislabs.com/?s=CRDT
- Riak - https://riak.com/distributed-data-types-riak-2-0/
- Microsoft Cosmos DB
  - “multiple, well-defined conflict resolution modes that guarantees strong convergence.”


Check YOUR Database!

# Let’s talk open source, Java, Cloud and … cool socks at booth #3101

- **Hands-on QuickLabs**
- Cloud-native development with Kabanero
- Getting started with Open Liberty
- Akka and Kubernetes: a symbiotic love story
- Intro to Kabanero and Spring

## Open Liberty

Experience the power of this cloud runtime for building cloud native apps in a fun, interactive game, Liberty Bikes. Open Liberty provides fully compatible implementations of Jakarta EE and Eclipse MicroProfile.

## Cloud-native Java

Talk to IBMers who are actively contributing to the open Java communities including: Open Liberty, Eclipse OpenJ9, Jakarata EE, Eclipse MicroProfile, Spring, Reactive and more.

## Journey to Cloud

Explore open source Kabanero which brings together Knative, Istio, and Tekton with new open projects Codewind, Appsody, and Razee into an end-to-end solution to architect, build, deploy, and manage the lifecycle of Kubernetes-based applications.