Globally Distributed SQL Databases FTW

Henrik Engström
Architect @ Kindred Group
Twitter: @h3nk3

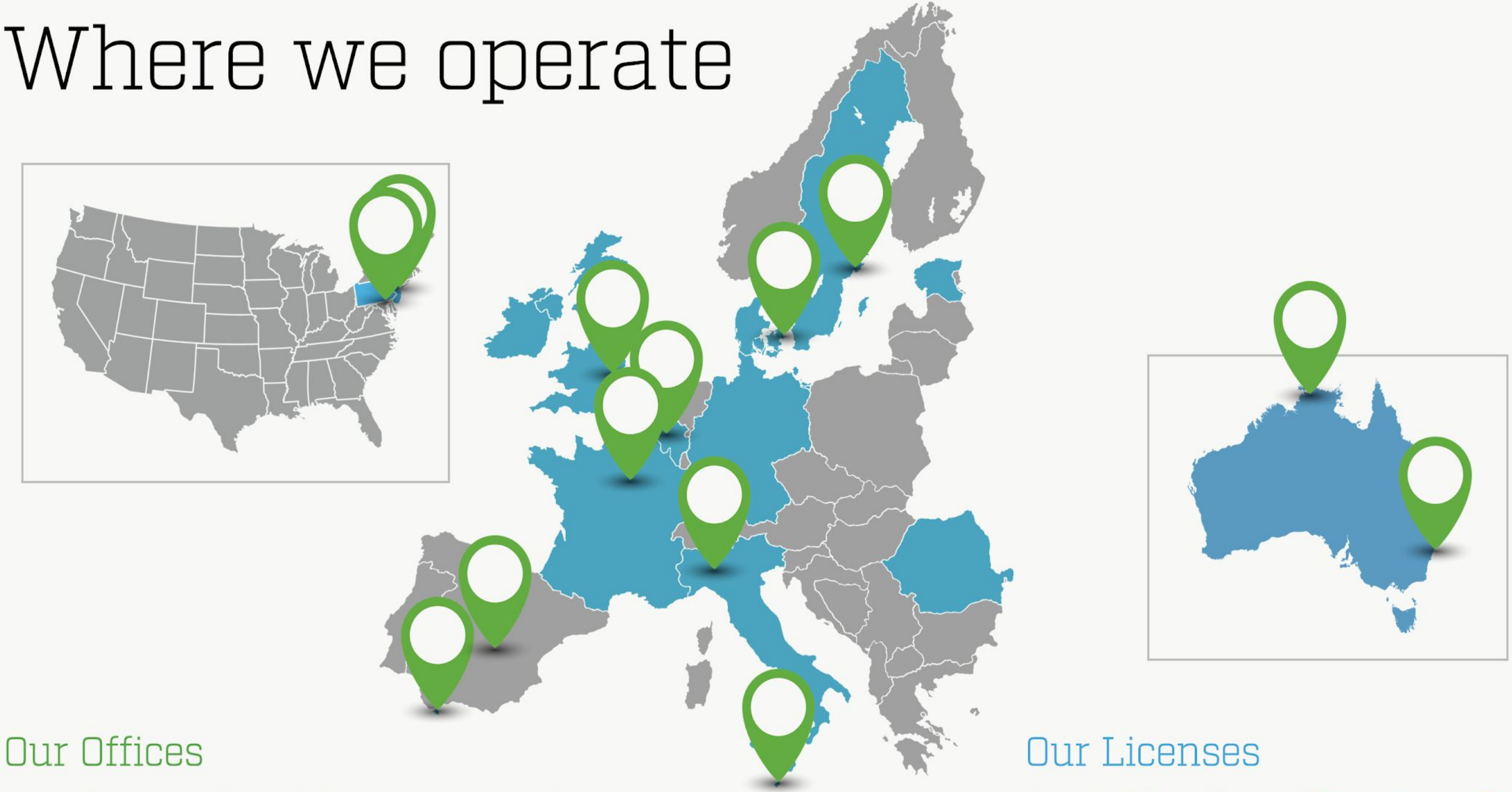
https://speakerdeck.com/h3nk3

~whois Kindred Group

- 11 gambling brands under one umbrella
  - Started as Unibet in 1997
  - 2018: £908m GWR, £203m EBITDA
  - 25 million users
- Microservice based, event-driven architecture based on Java
- Lots of transactions and data!
  - Example: Kindred vs. PayPal
    - Paypal Q2 in 2019: ~3.1 billion/tx
    - Kindred transactions handled: ~2.7 billion/quarter
- 1500 employees with ~500 in Tech spread over the globe
  - Tech hubs: Stockholm, London, Madrid, Sydney, Belgrade, Malta, Gibraltar
- Kindred is always looking for great engineers

kindred

Where we operate



Our Offices

Antwerp | Copenhagen | Darwin | Gibraltar | London | Madrid

Malta | Milan | New Jersey | New York | Paris | Stockholm | Sydney

Our Licenses

Australia | Belgium | Denmark | Estonia | France | Germany | Gibraltar | Ireland

Italy | Malta | New Jersey | Pennsylvania | Romania | Sweden | United Kingdom

~ less henrik_engstrom.txt

First program written in 1984 on an ABC 80

Professional developer since 1998

Consultant in gambling, finance, retail [1998-2010]

Principle Engineer @ Typesafe/Lightbend (Scala/Akka/observability) [2011-2018]

Architect @ Kindred Group [2019-]

henrik_engstrom.txt (END)

kindred

~ ls -l agenda

drwxr--r-- 3 root staff Feb 5 2020	Challenges with Data Consistency
drwxr--r-- 3 root staff Feb 5 2020	Data Consistency Definitions
drwxr--r-- 3 root staff Feb 5 2020	Application Tier Consistency
drwxr--r-- 3 root staff Feb 5 2020	NewSQL and CockroachDB

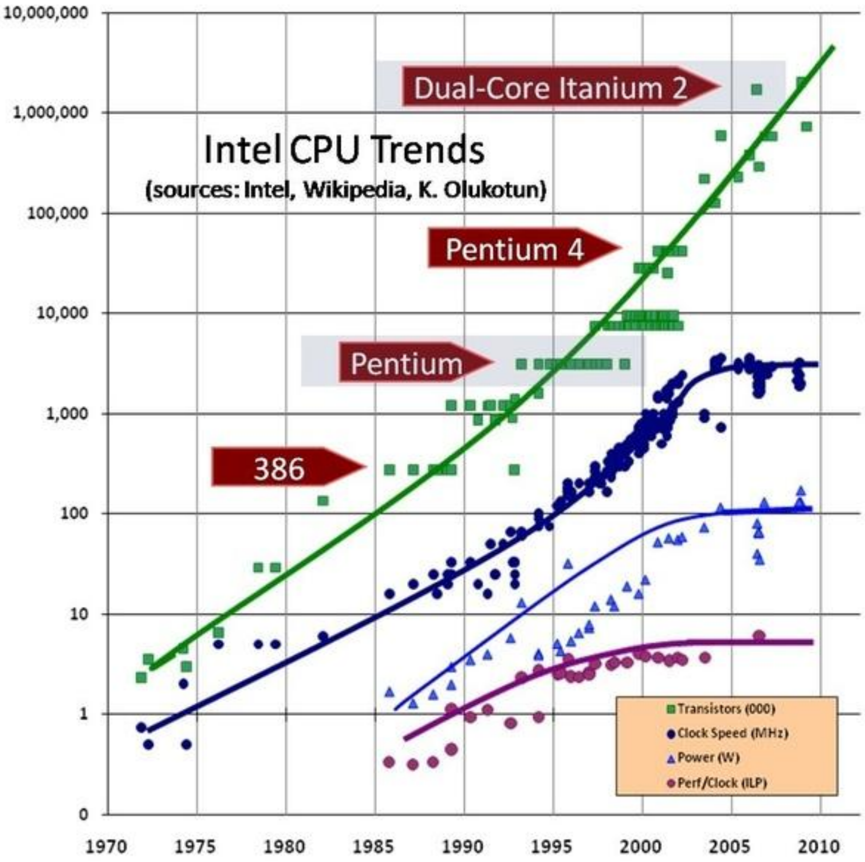
kindred

Challenges with Data Consistency

All images in this presentation are from https://www.pexels.com/ unless otherwise stated.

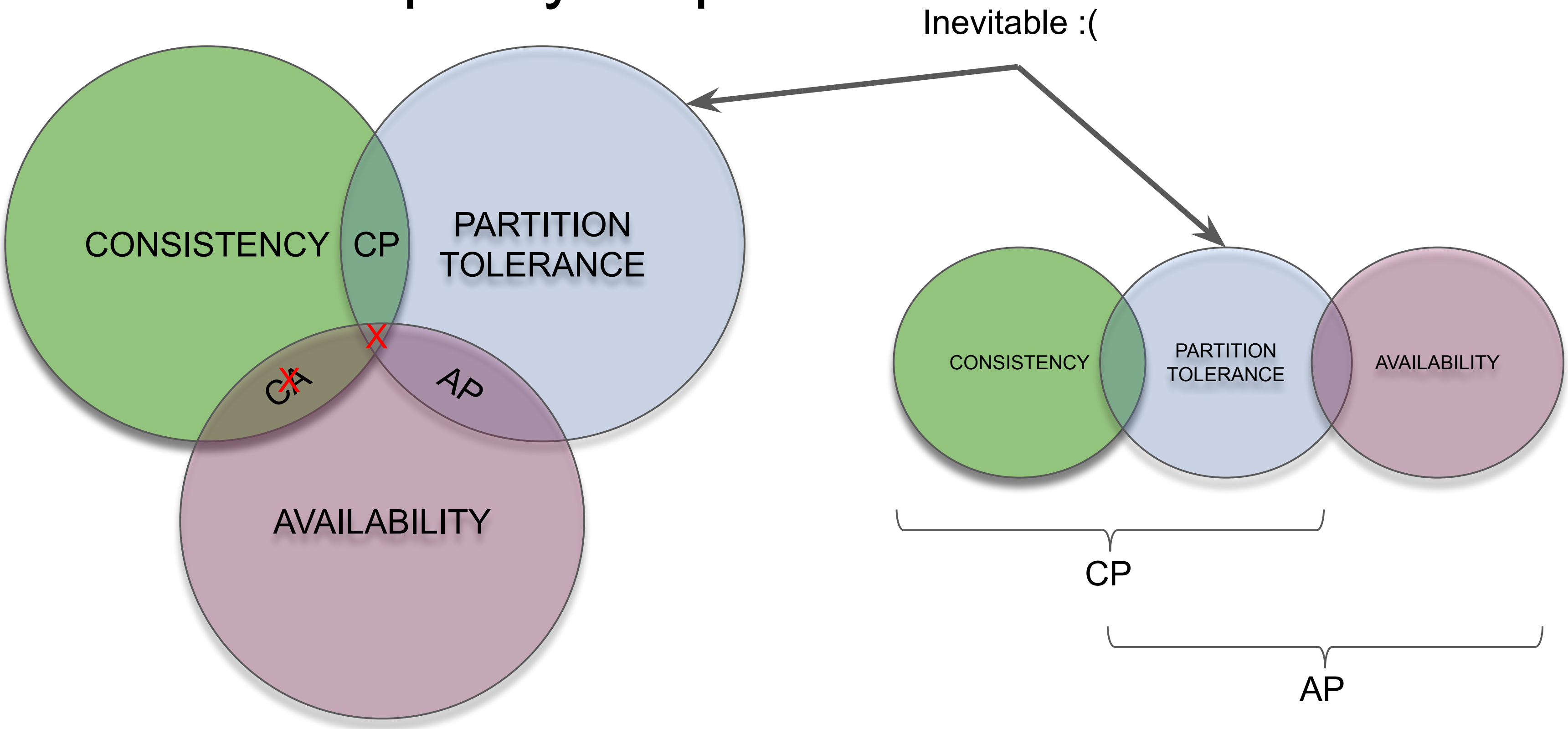
Back in mid 2000s

- CPU single core → multiple cores
- Big monolith HW → Smaller servers
- On-prem → “cloud”
- Distributed systems
- RDBMS and TX → EC and NoSQL
- CAP theorem



kindred

CAP - a.k.a. pick your poison



kindred

Pat Helland's "Life beyond distributed TX" quotes

Life beyond Distributed Transactions: an Apostate's Opinion [2007]

"Personally, I have invested a non-trivial portion of my career as a strong advocate for the implementation and use of platforms providing guarantees of global serializability."

Life beyond Distributed Transactions: an Apostate's Opinion [2007]

”In general, application developers simply do not implement large scalable applications assuming distributed transactions. When they attempt to use distributed transactions, the project founder because the performance cost and fragility make them impractical.”

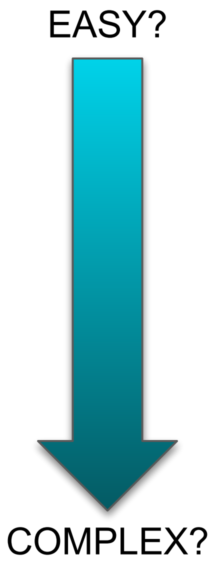
Life beyond Distributed Transactions: an Apostate's Opinion [2007]

“In a system which cannot count on distributed transactions, the management of uncertainty must be implemented in the business logic.”

Life beyond Distributed Transactions: an Apostate's Opinion [2007]

Evolution of Consistency

- Strong Consistency
  - ACID
    - Atomicity - all or nothing
    - Consistency - no violating constraints
    - Isolation - exclusive access
    - Durability - committed data survives crashes
  - Associated with RDBMS
- Eventual Consistency
  - ACID 2.0
    - Associative - Set().add(1).add(2) === Set().add(2).add(1)
    - Commutative - Math.max(1,2) === Math.max(2,1)
    - Idempotent - Map().put(“a”,1).put(“a”,1) === Map().put(“a”,1)
    - Distributed - symbolical meaning
  - Associated with NoSQL



Data Consistency Definitions

LAMIGO MONKEYS
LAMIGO MONKEYS
LAMIGO MONKEYS
LAMIGO MONKEYS
330
Lagoon

Baseball Rules in Code

```scala
case class Team(name: String, var points:Int=0) {
    def addPoint(inning: Int, p: Int): Unit = {
        points = points + p
        println(s"[$inning] $name team score $p") // simplified score reporting
    }
}
class Rules {
    val visitors = Team("Visitors")
    val home = Team("Home")
    for (i <- 1 to 9) { // innings
        var outs = 0
        while (outs < 3)
            play() match {
                case s: Score => visitors.addPoint(i, s.points)
                case Out => outs += 1
            }
        outs = 0
        while (outs < 3)
            play() match {
                case s: Score => home.addPoint(i, s.points)
                case Out => outs += 1
            }
    }
    def play(): Result = { // game simulation }
}
```

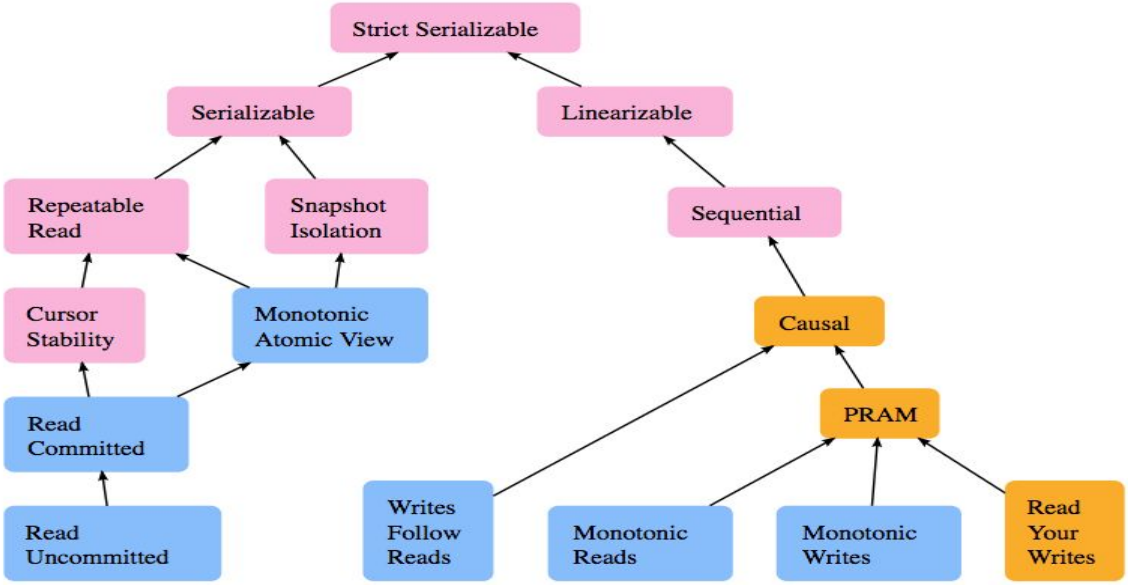
kindred

Let’s Play a Game

	1	2	3	4	5	6	7	8	9	RUNS
VISITORS	0	0	1	0	1	0	0			2
HOME	1	0	1	1	0	2				5

Strong Consistency	2-5
Eventual Consistency	0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-5, 1-4, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5
Consistent Prefix (~ Snapshot Isolation)	0-0, 0-1, 1-1, 1-2, 1-3, 2-3
Monotonic Reads	(after reading 1-3): 1-3, 1-4, 1-5, 2-3, 2-4, 2-5
Read Your Writes	(for writer): 2-5 (anyone else): 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3 ,1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5

Consistency Models Overview



Credit:
Aphyr @ jepsen.io
Peter Bailis @ http://www.bailis.org/blog/linearizability-versus-serializability/

kindred

Application Tier Consistency

Challenges in an EC World

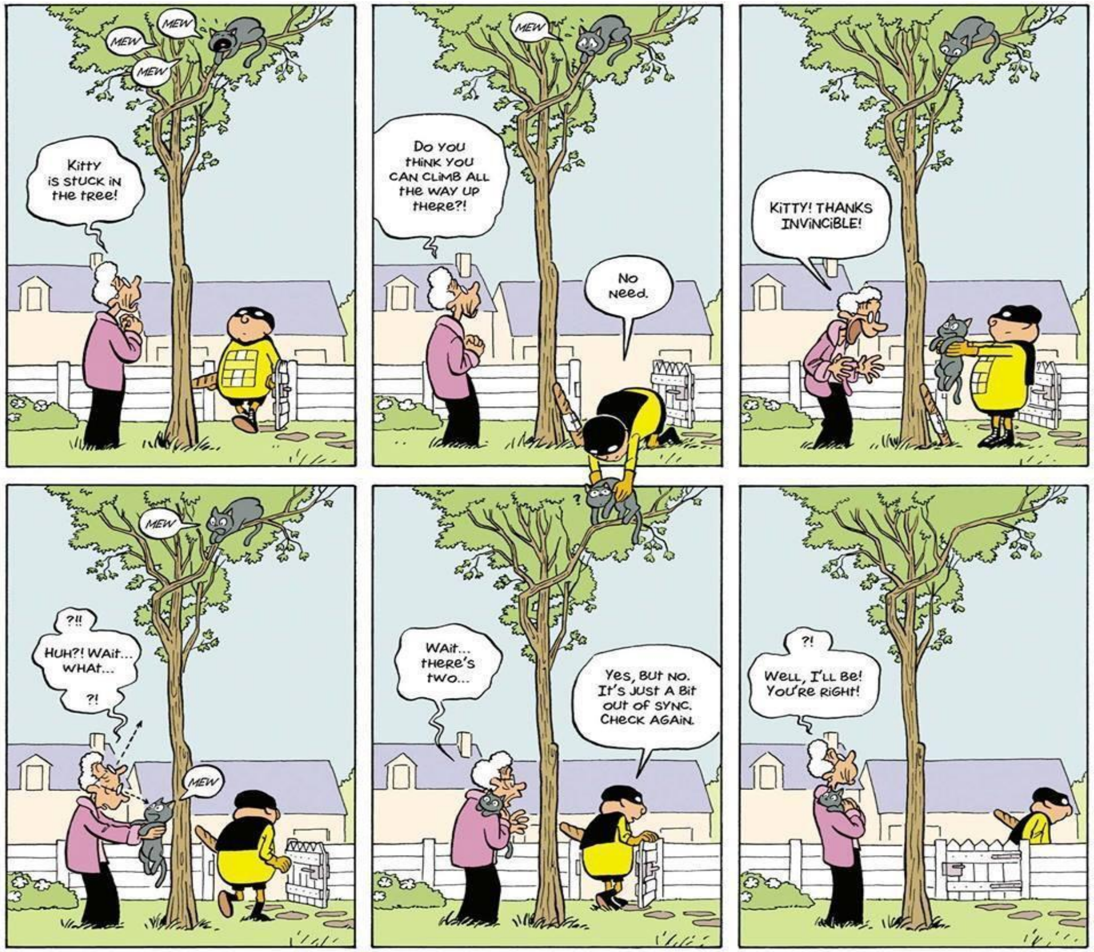


Image Credits

kindred

Building Highly-Scalable Distributed Systems is Easy

Just sprinkle some of the following onto your design:

- Eventual consistency
- Idempotence
- CQRS
- CRDTs
- The Saga Pattern
- ...and so on…

Also known as “*push the hard problems somewhere else*”™

kindred


Cockroach DB

NewSQL

- Coined in 2011
- Support the relational model
  - Lessons learned: SQL is pretty good to have and is ubiquitous
- Back to ACID 1.0 again
- Support fault tolerance *and* horizontal scalability
- Split databases into “chunks” and use Paxos or Raft for consensus
- Example of NewSQL databases:
  - Spanner (Google) - the flagship NewSQL DB
  - Azure Cosmos DB (Microsoft) - multiple consistency levels available
  - CockroachDB

kindred

“We believe that it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions.”

Spanner: Google’s Globally-Distributed Database [2012]

kindred

“Within Google, we have found that this [strong guarantees] has made it significantly easier to develop applications on Spanner compared to other database systems with weaker guarantees. When application developers don’t have to worry about race conditions or inconsistencies in their data, they can focus on what they really care about - *building and shipping* a great applications.”

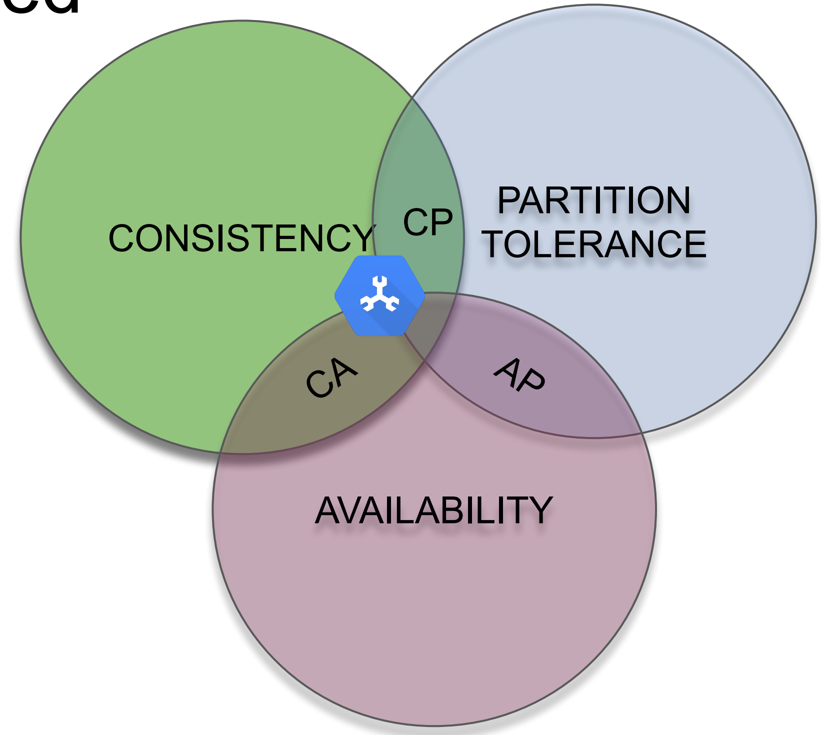
Life of Cloud Spanner Reads & Writes

kindred

“Real world almost-infinite scale applications would love the luxury of a global scope of serializability as is promised by two phase commit and other related algorithms.”

Life beyond Distributed Transactions: an Apostate's Opinion [2007]

CAP Revisited



Spanner, TrueTime & The CAP Theorem [2017]

CockroachDB Overview

Formerly open-sourced DB based on and inspired by Spanner concepts.

Some CRDB highlights:

- Distributed, replicated, transactional key-value store
- Uses so-called ranges and Raft for consensus
- No atomic clocks required - uses HLC and NTP
- Geo-distribution capabilities
- Full SQL support - Postgresql dialect
- Built-in Change Data Capture functionality (The Outbox Pattern)
- Jepsen tested and approved
- Cloud native

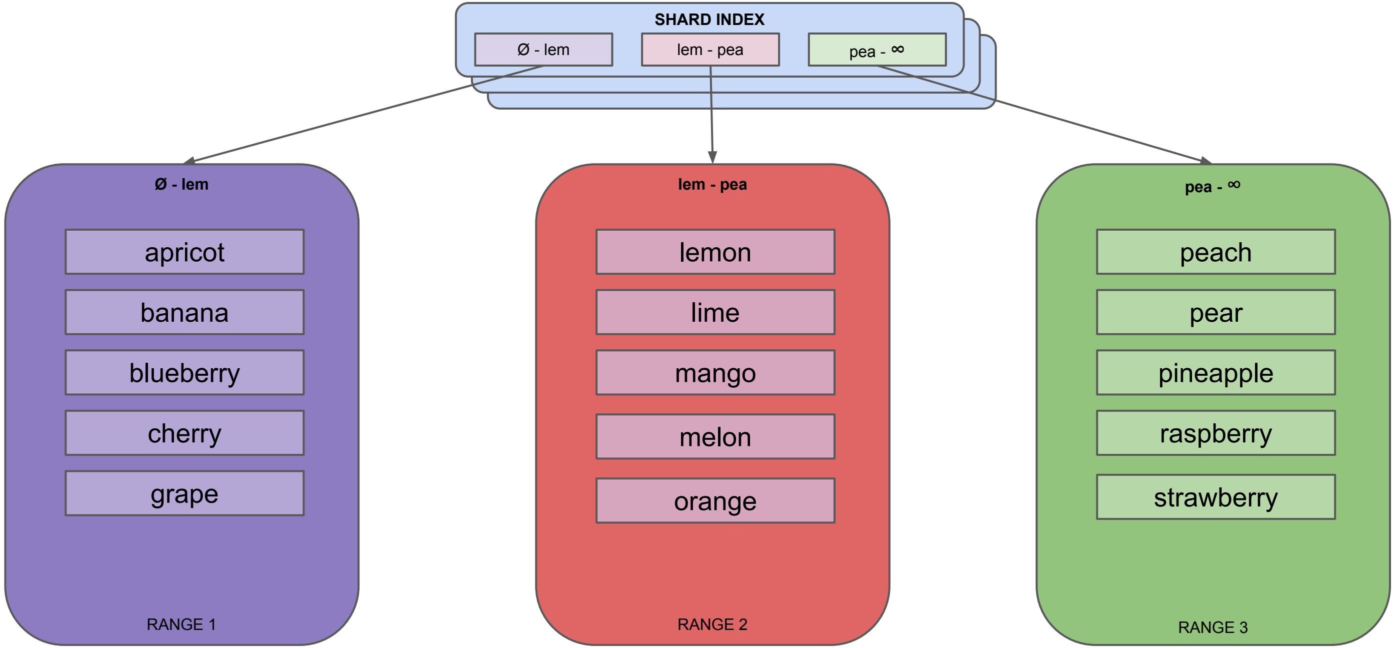
Ranges and Replication

- Ranges
  - Ranges (64MB) are the units of replication
  - Raft for consensus
  - Each range is a Raft group
  - Minimum of 3 nodes required
- Leaseholders - 1 per range
  - Read/write control
  - Follow-the-workload
- Replica Placement Algorithm:
  - Space
  - Diversity
  - Load
  - Latency

kindred

CRDB Replication Layer: Sharding and Index

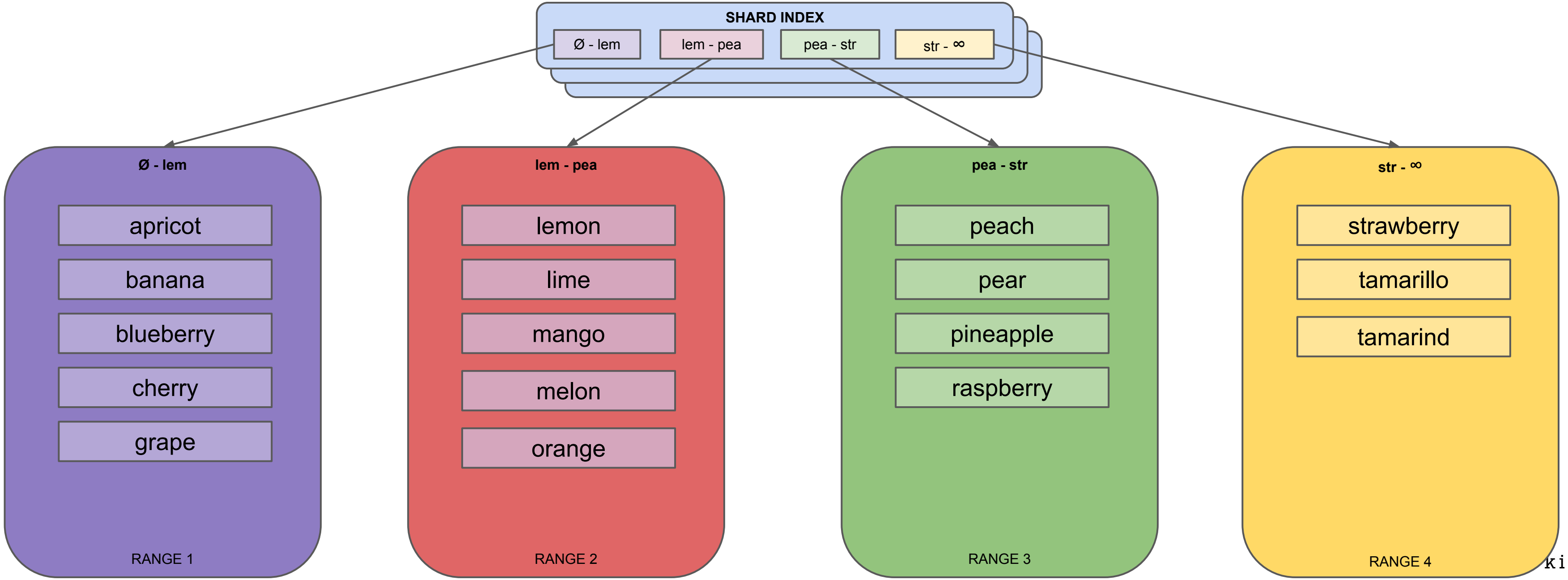
- Data is split up into 64MB ranges - each holding a contiguous range of keys
- An index maps from key to range ID



kindred

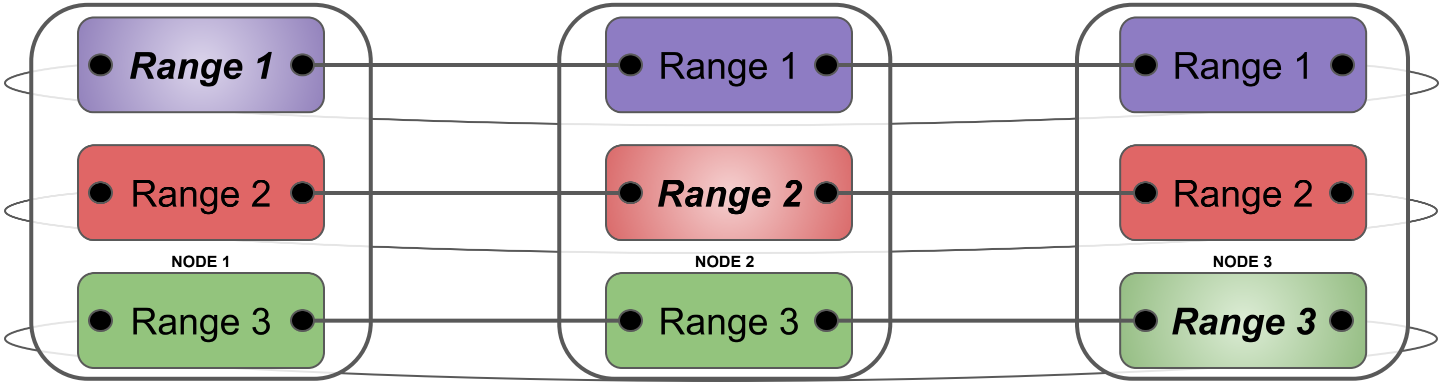
CRDB Replication Layer: Split

- Split when range is out of space or too hot

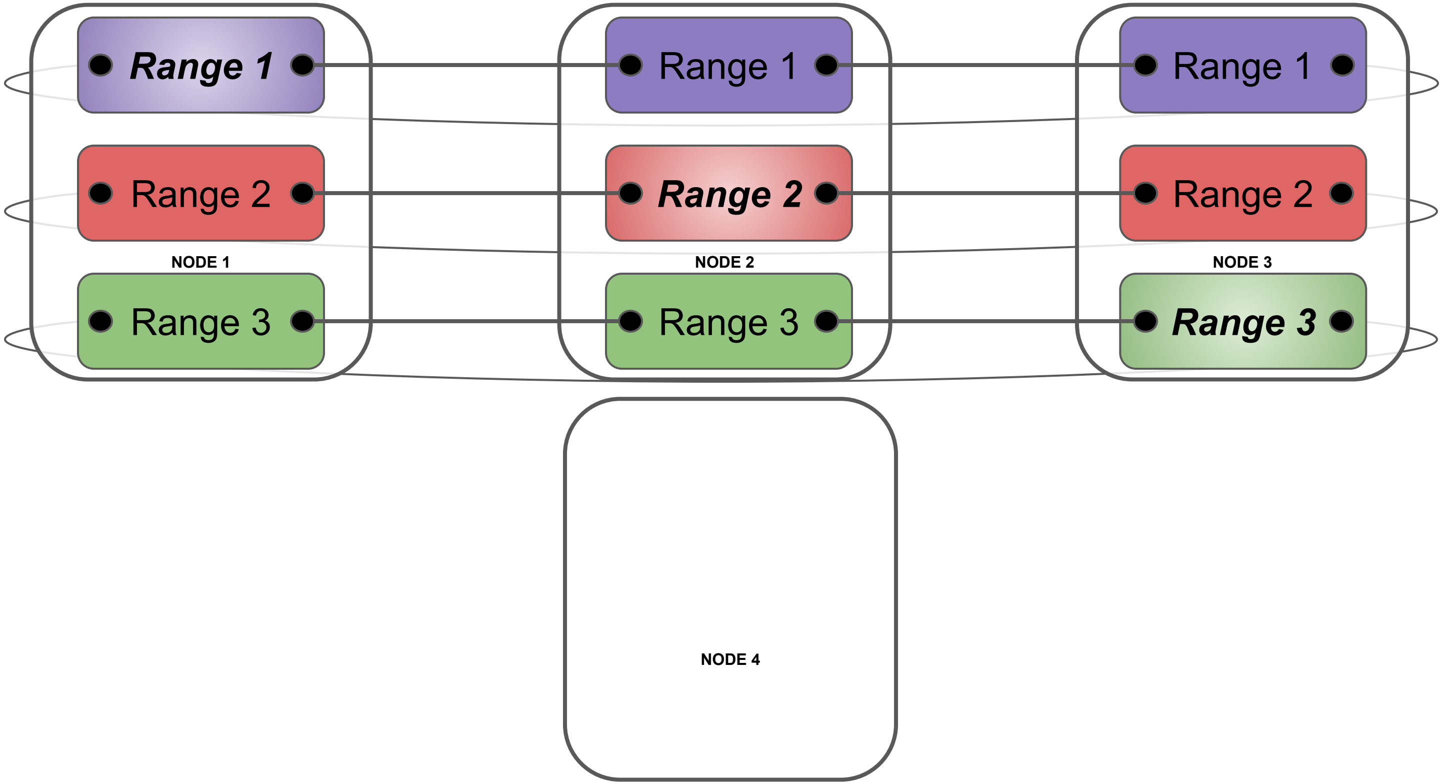


kindred

CRDB Replication Layer: Placement

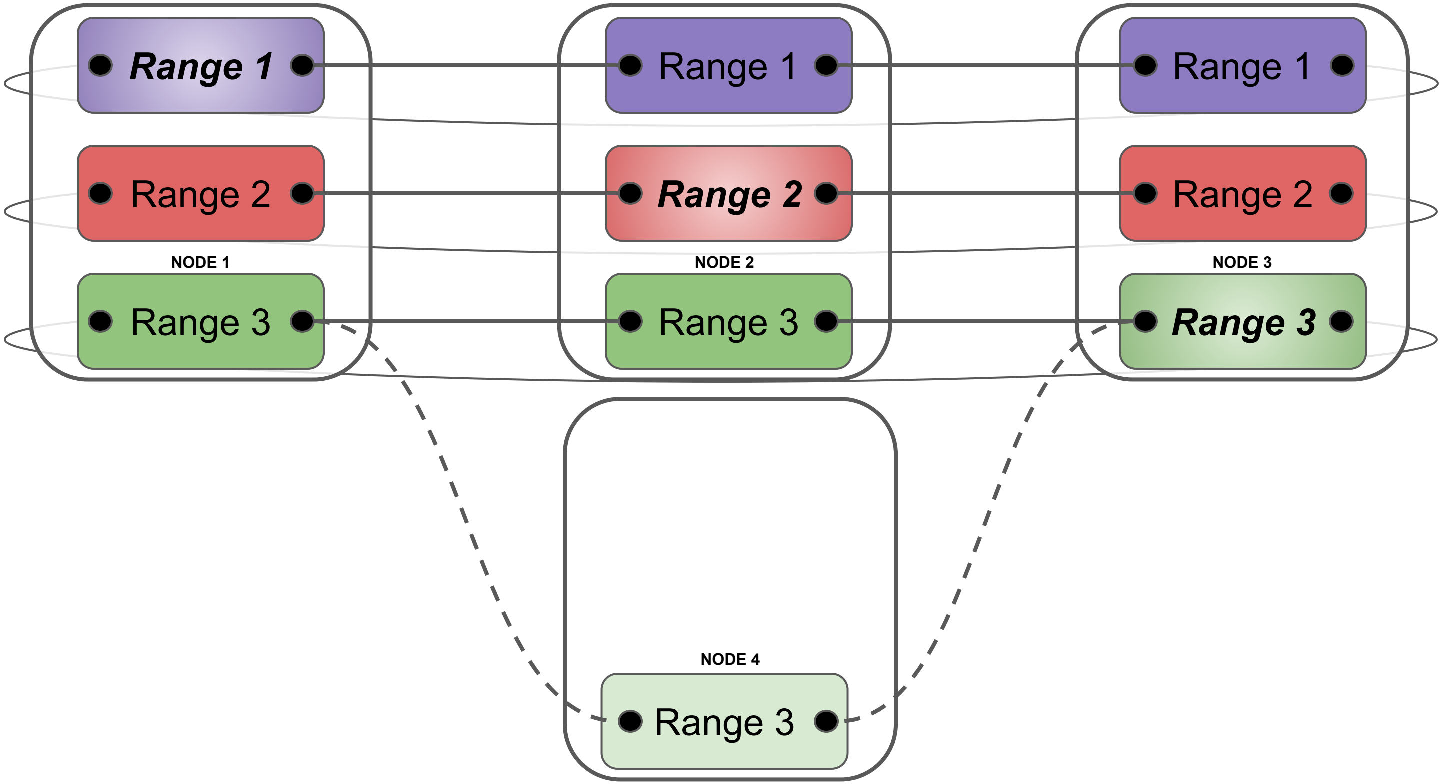


CRDB Replication Layer: Rebalancing

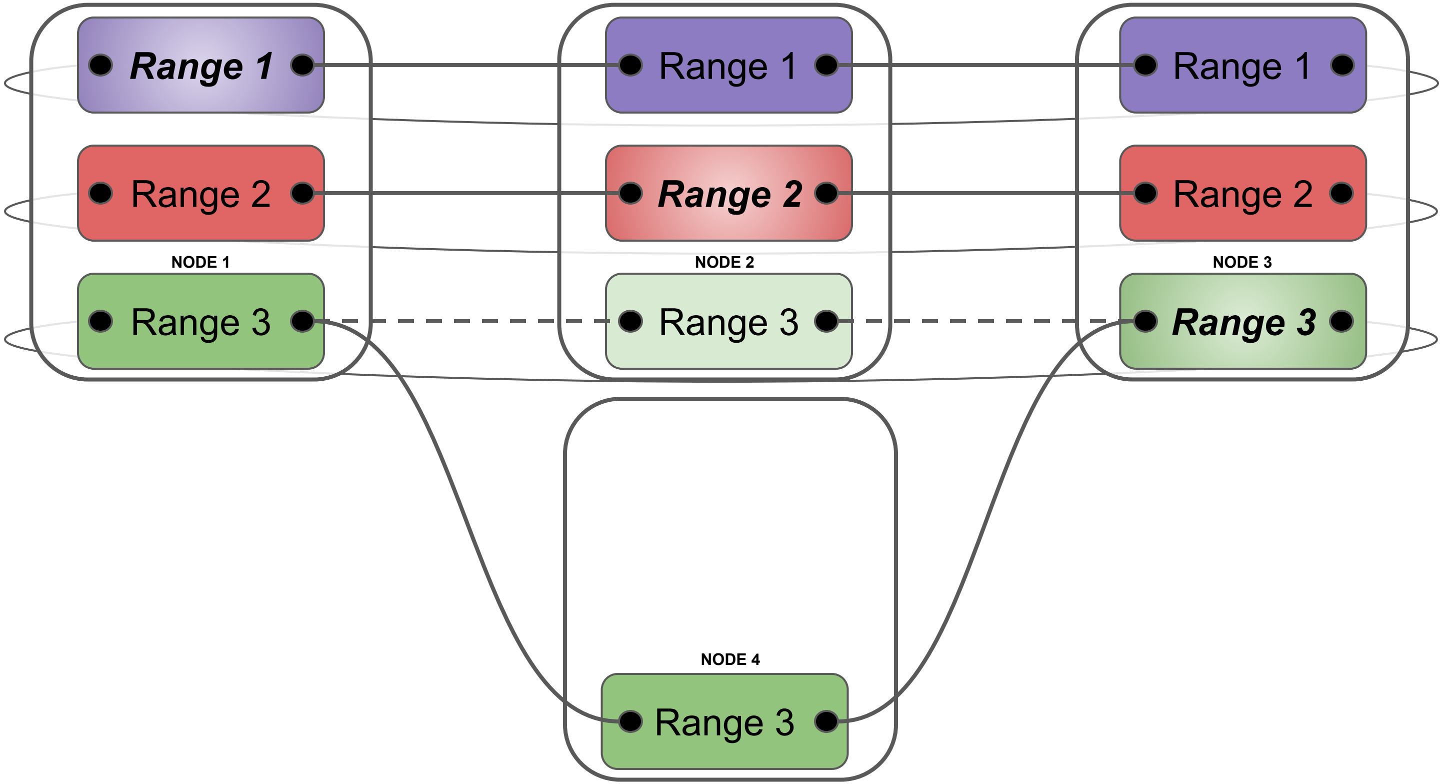


kindred

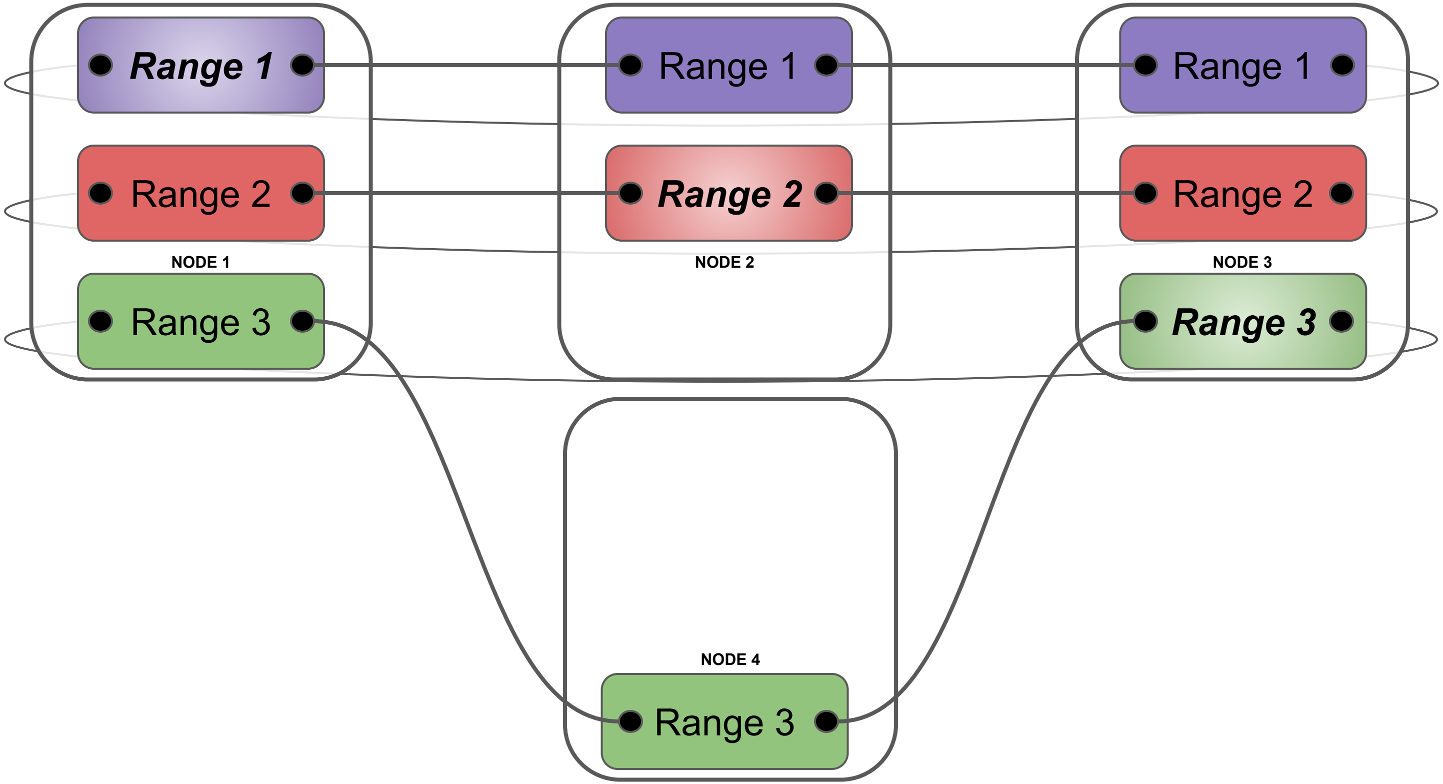
CRDB Replication Layer: Rebalancing



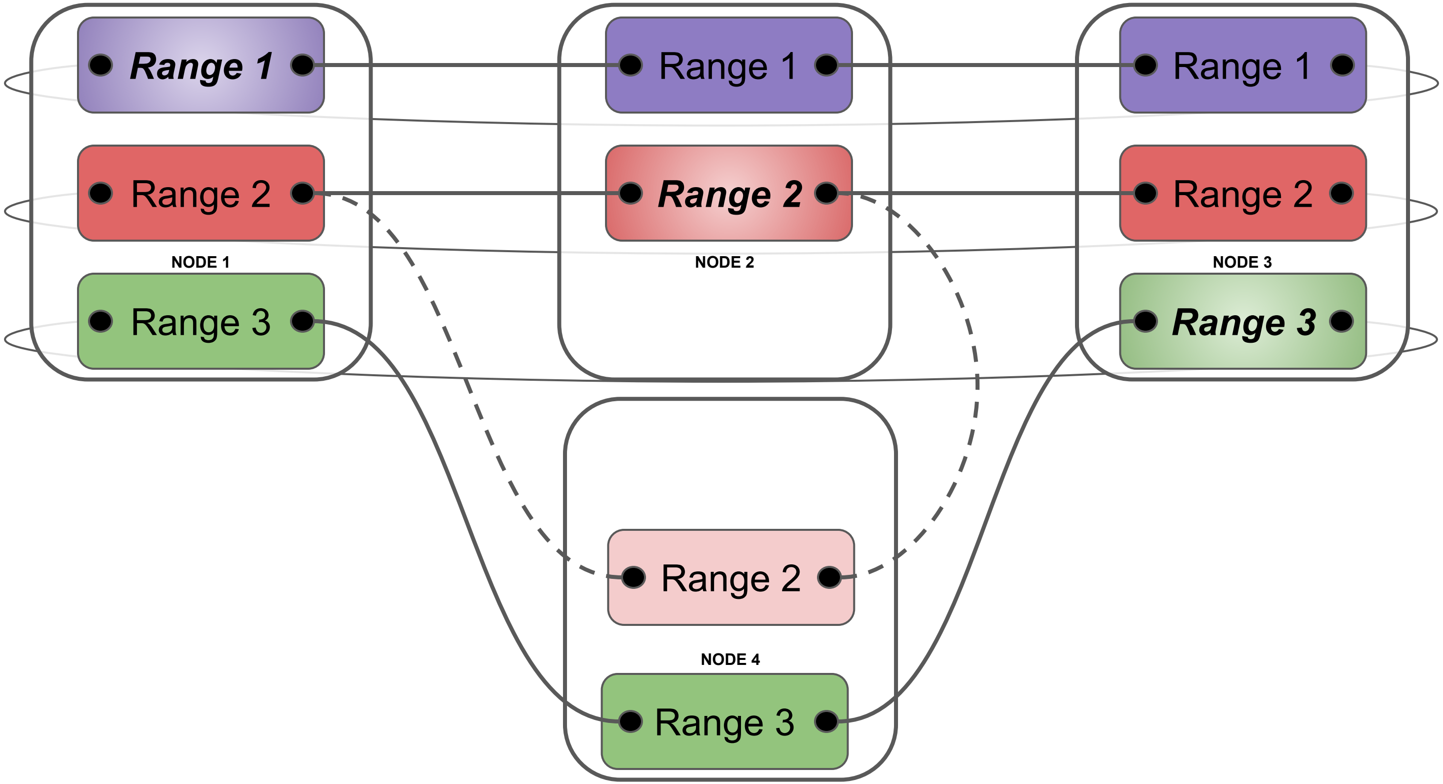
CRDB Replication Layer: Rebalancing



CRDB Replication Layer: Rebalancing

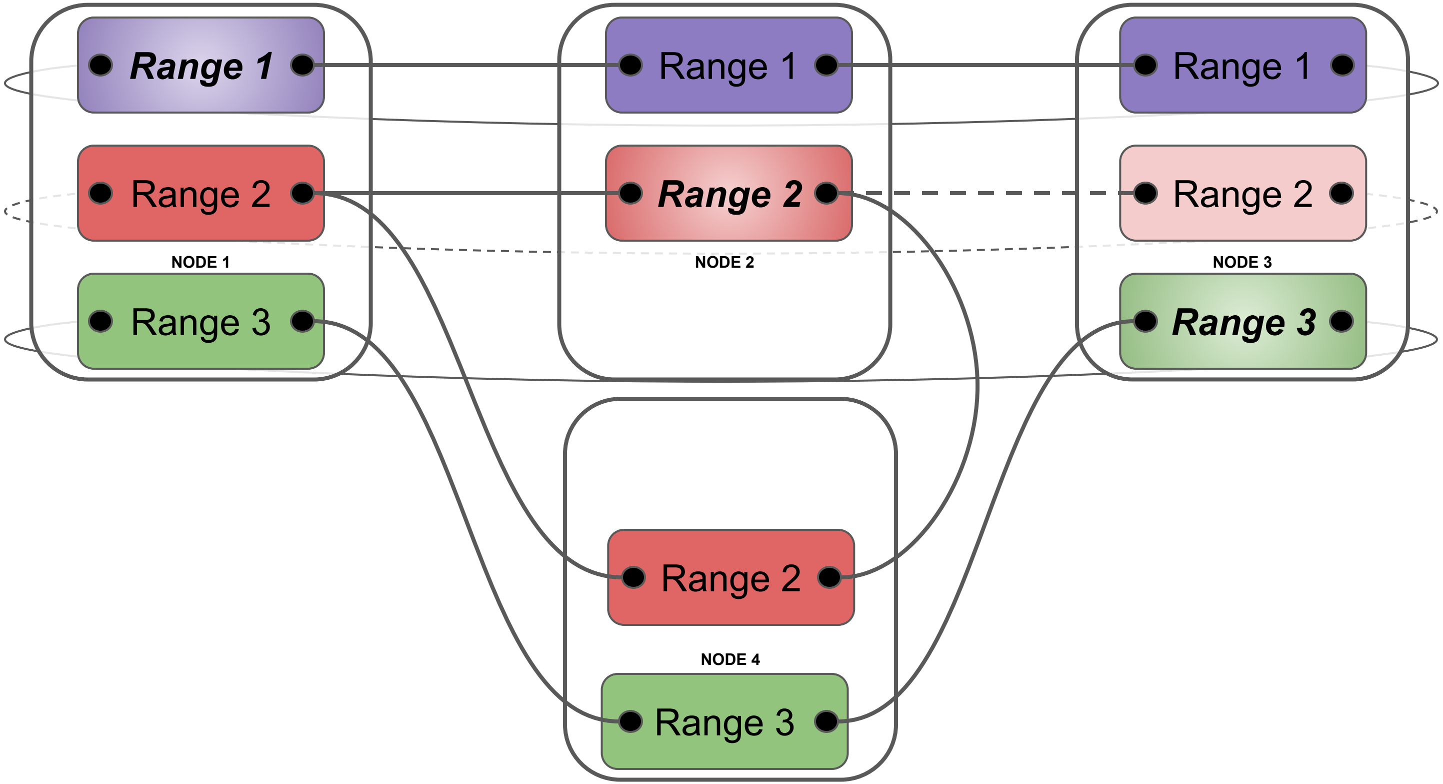


CRDB Replication Layer: Rebalancing



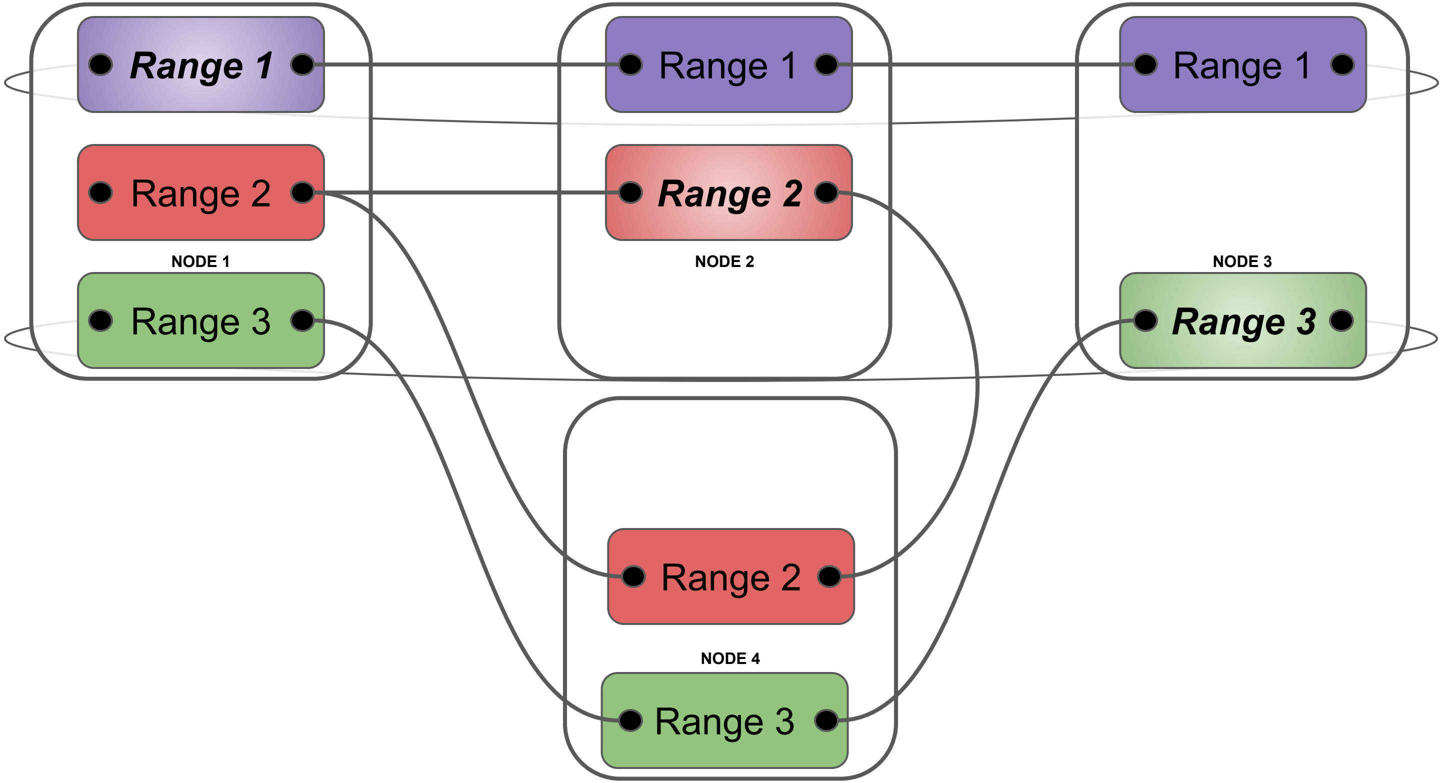
kindred

CRDB Replication Layer: Rebalancing



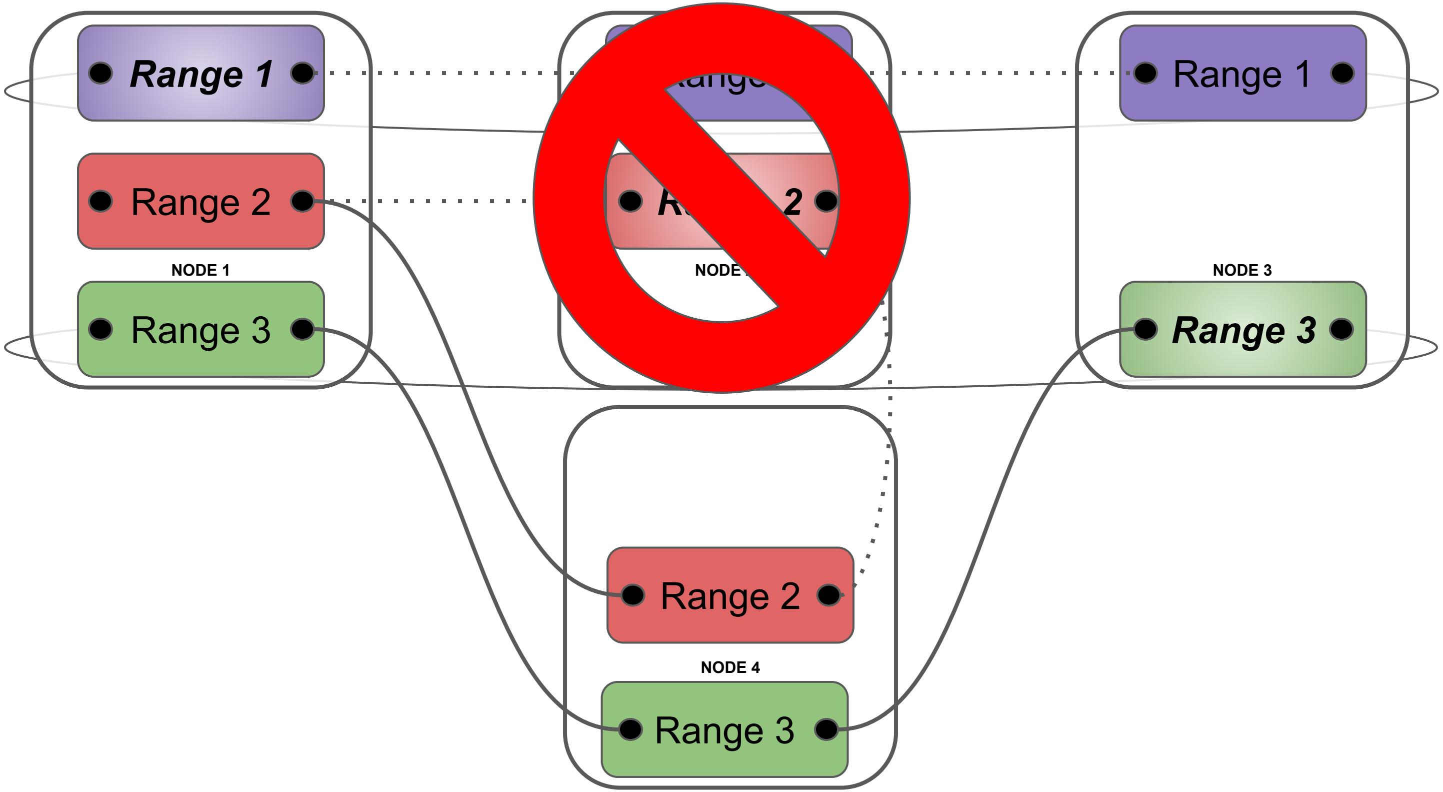
kindred

CRDB Replication Layer: Rebalancing

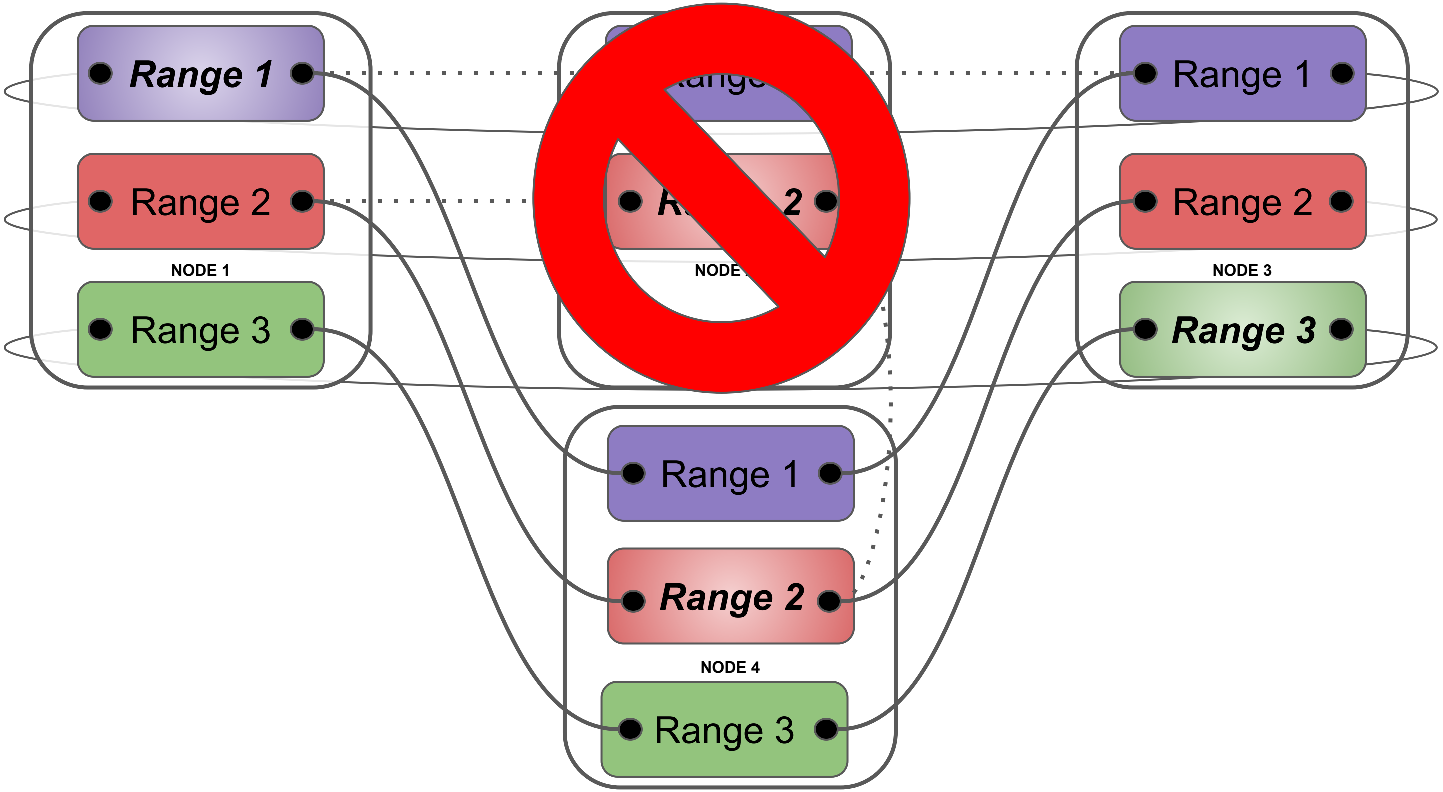


kindred

CRDB Replication Layer: Recovery

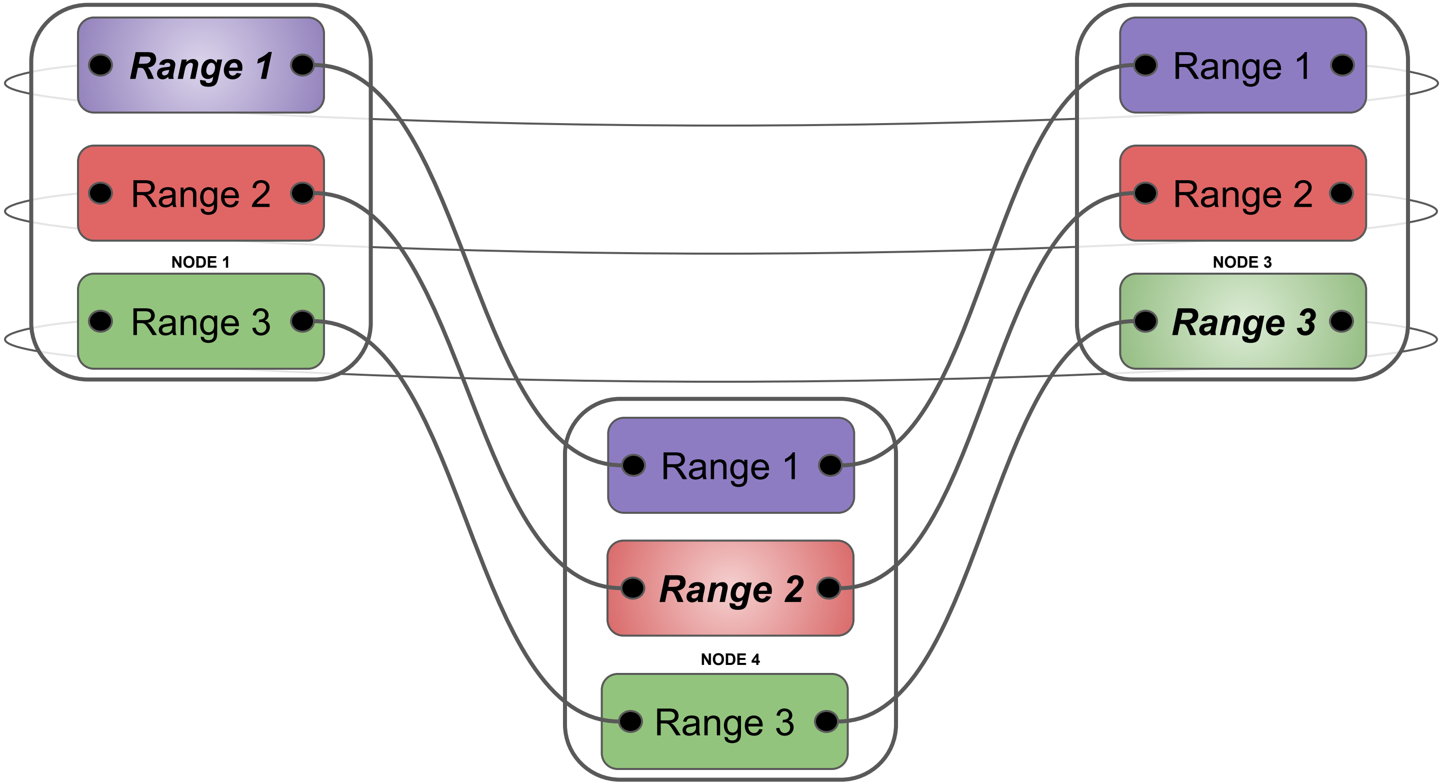


CRDB Replication Layer: Recovery



kindred

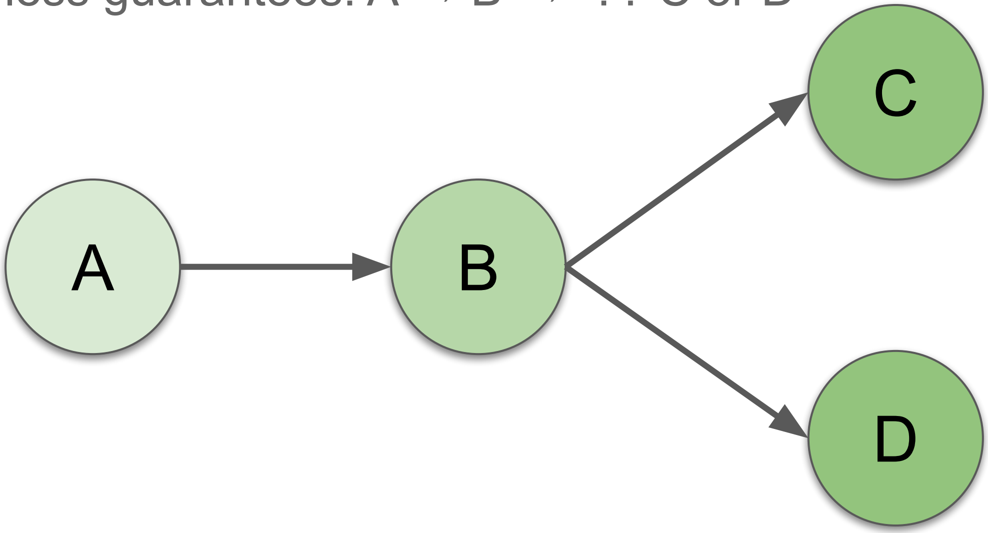
CRDB Replication Layer: Recovery



Short Digression About Time

Ordering

- Ordering - A happens before B - implies a notion of time
- Total and Partial Ordering
  - Total - full ordering guarantees: A → B → C and A → B → D
  - Partial - less guarantees: A → B → ?? C or D



kindred

Clock Alternatives for Distributed Systems

- Perfectly Accurate Global Clocks
    - Google's TrueTime is almost perfect: ~7ms range
- Imperfect Local Clocks
    - Total ordering on the local level - monotonic updates
    - No ordering between nodes
    - More "real-world" like than Google
- No Physical Clocks Available
    - Fallback: use a logical "clock" instead

kindred

Logical: Lamport Clock

- A counter incremented when:
  - The process executes
  - The process receives a message

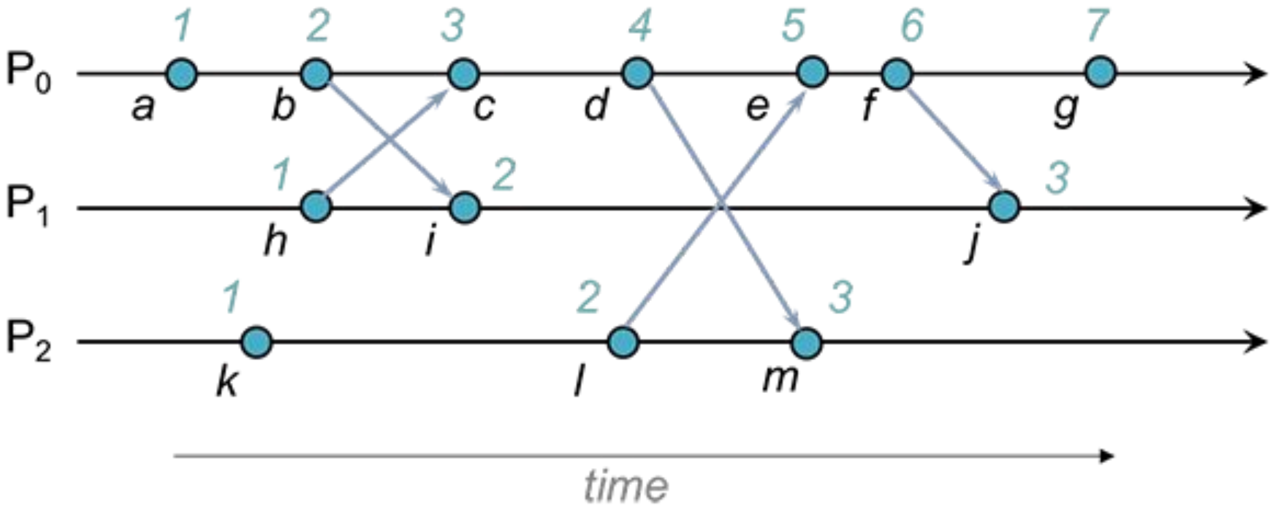


Image credit: https://www.cs.rutgers.edu/~pxk/417/notes/clocks/index.html

kindred

Logical: Vector Clock

- A counter incremented when:
  - The process executes
  - The process receives a message using `max(local, received)`

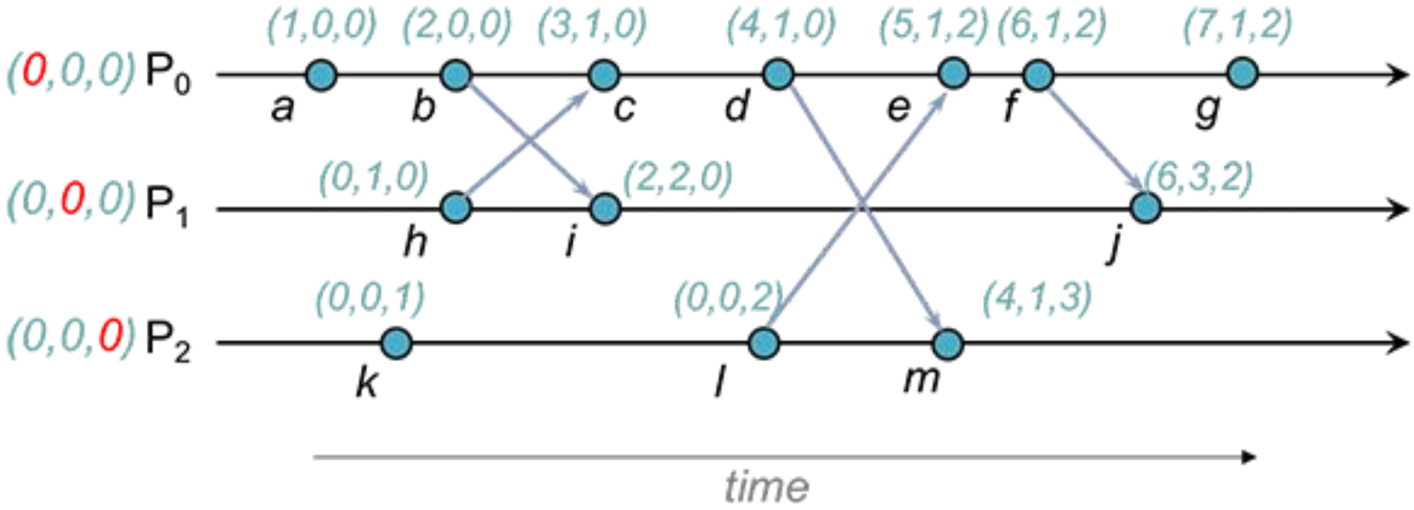
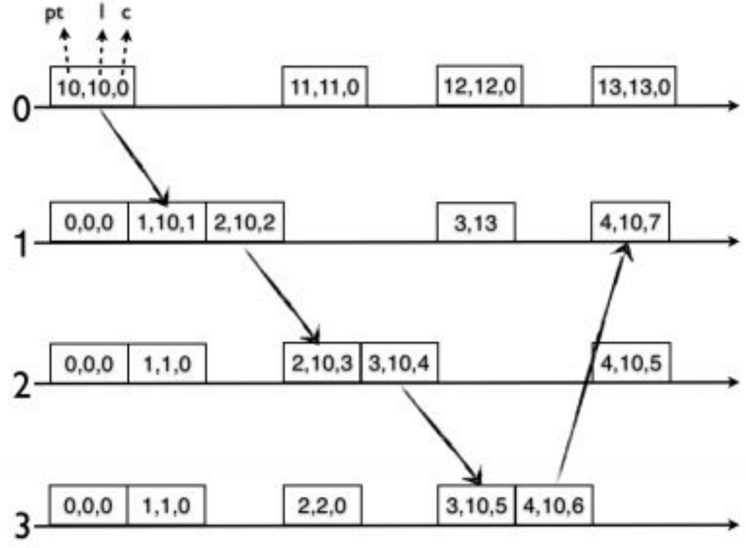


Image credit: https://www.cs.rutgers.edu/~pxk/417/notes/clocks/index.html

Hybrid Logical Clocks (HLC)

- A combination of *physical* and *logical* clocks

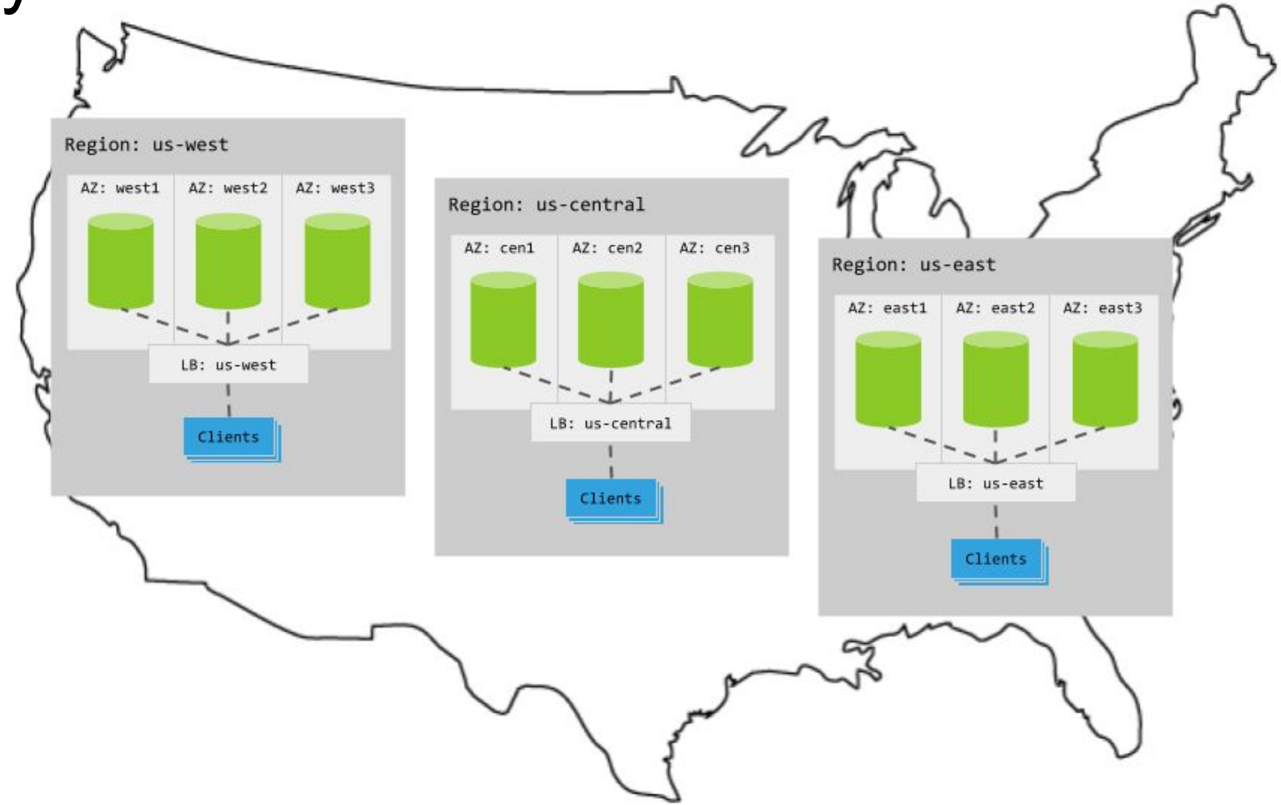


pt: Physical Time
l: Logical Time (keeps max pt seen)
c: Causality

Paper and Image credit: https://cse.buffalo.edu/tech-reports/2014-04.pdf

kindred

Geo keys



kindred

Geo keys

- Set location when starting node:

```
cockroach start --locality=region=us-east ...
```

- Partition by list in table definition:

```
PARTITION BY LIST (city)
  (PARTITION us-east VALUES IN ('NYC','DC'),
   PARTITION us-central VALUES IN ('Denver', 'SLC'),
   PARTITION us-west VALUES IN ('LA','SF'),
   PARTITION DEFAULT VALUES IN (default));
```

- Partition by range in table definition:

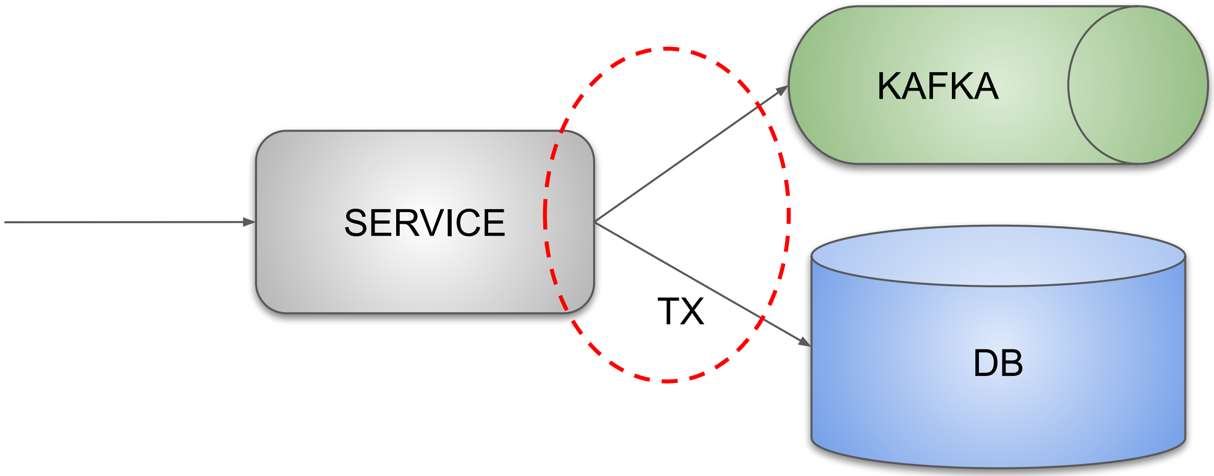
```
PARTITION BY RANGE (expected_graduation_date)
  (PARTITION graduated VALUES FROM (MINVALUE) TO ('2017-08-15'),
   PARTITION current VALUES FROM ('2017-08-15') TO (MAXVALUE));

> ALTER PARTITION current OF TABLE students CONFIGURE ZONE USING
constraints='[+ssd]';
```

kindred

CockroachDB and SQL

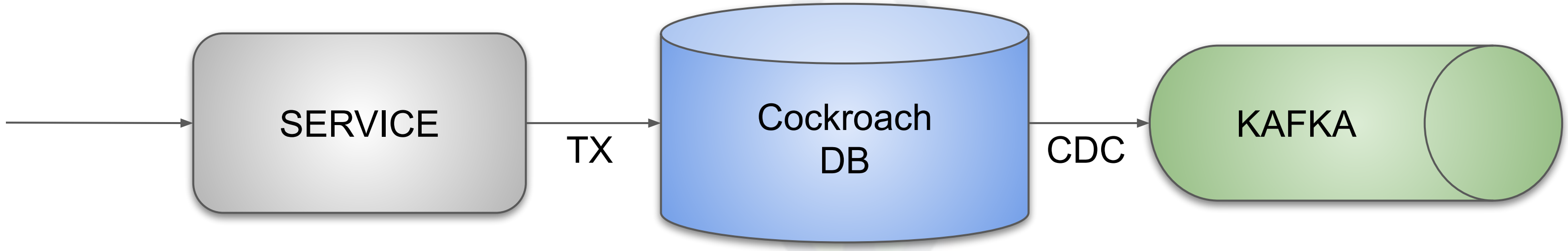
- Implements a large portion of the ANSI SQL standard
- Uses the PostgreSQL wire protocol
  - PostgreSQL-compatible drivers
  - GORM (GoLang)
  - Hibernate (Java)
- *Full* ACID support
- Distributed SQL (DistSQL) optimization tool for some queries
- Lots of effort put into performance!

The Dual Writes Problem



kindred

Change Data Capture (CDC)

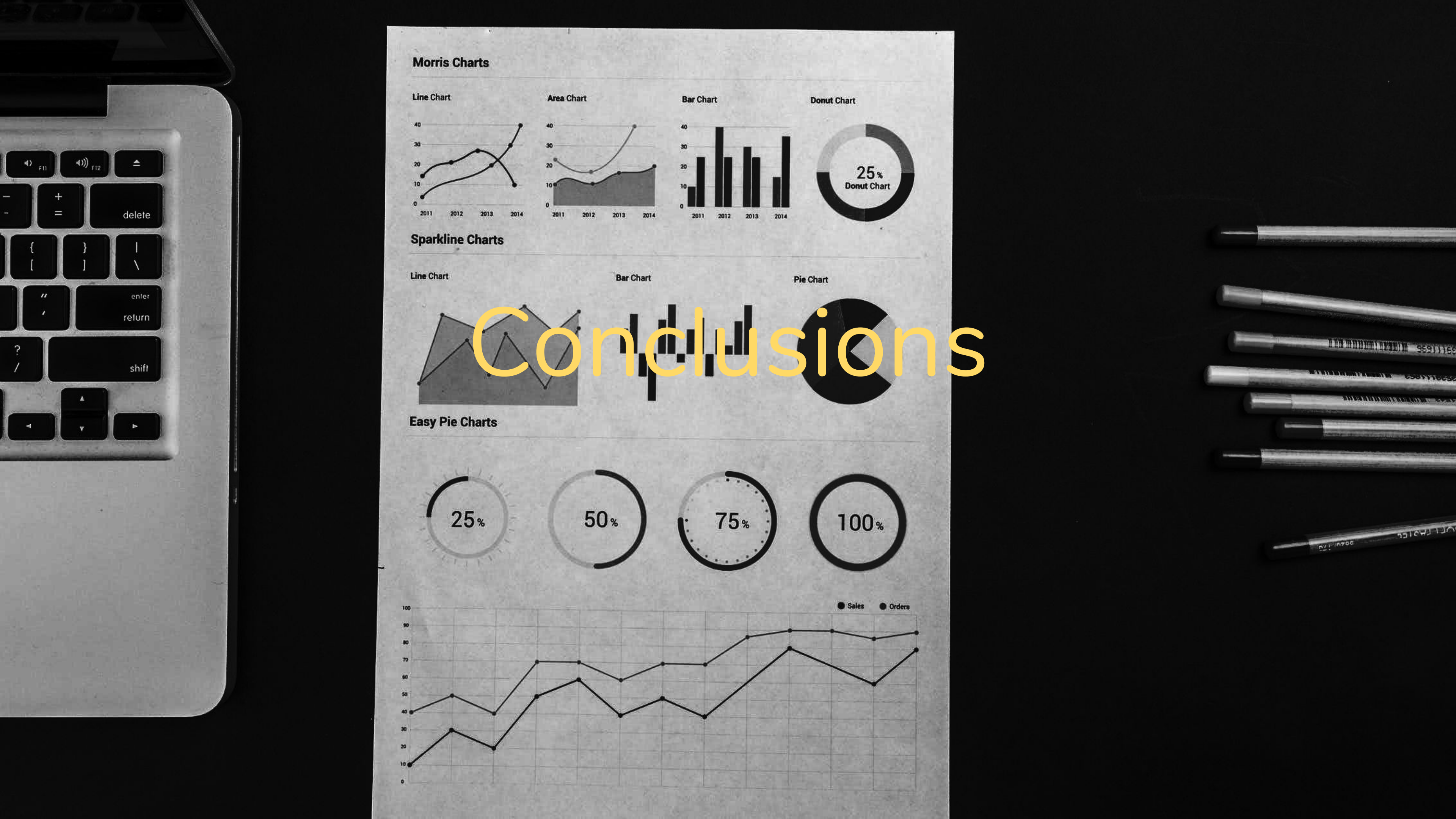


kindred

Challenges with Distributed Systems

- Performance/Latency
  - Going single homed → multi homed always comes with a cost (quorum price)
  - Some ping time examples
    - Stockholm - Dublin: 61.8 ms
    - New York - Tokyo: 215.9 ms
    - Frankfurt - Singapore: 150.2 ms
    - Sydney - Paris: 281.3 ms
  - CockroachDB mitigation: Use geo-positioning of data
- Observability (reasoning) hard in distributed systems

Conclusions



Questions to Ponder

- What role(s) do you play in the baseball match?
- How scalable do you need to be?
- What does your average engineering profile look like?
  - Remember what Google said about Eventual Consistency
- How much SQL do use? How much would you like to use?
- Are you able and willing to use the cloud?
- *Let the above guide you to the right solution.*


Decide how hot or cold you need the water

THANK YOU FOR LISTENING

Twitter: @h3nk3
Slides: https://speakerdeck.com/h3nk3