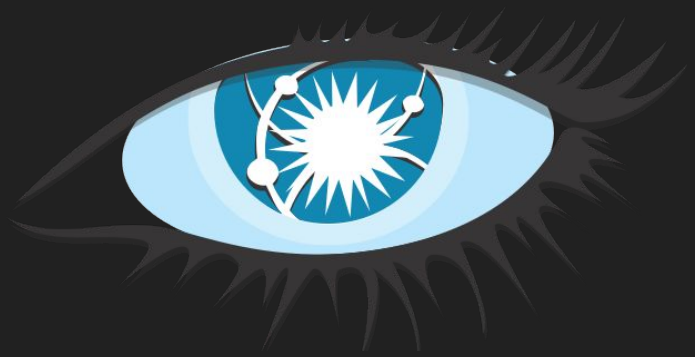
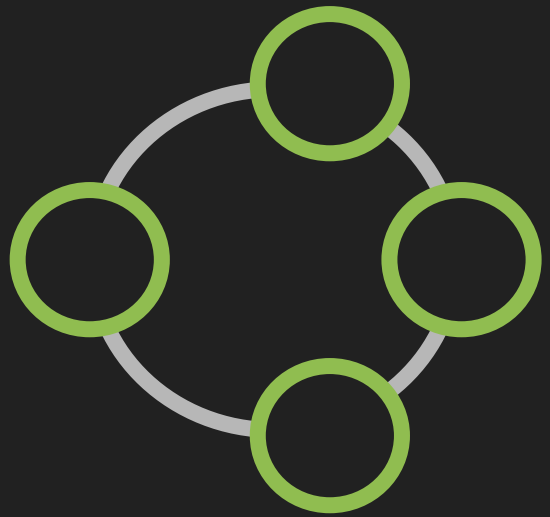
Transaktionsunterstützung



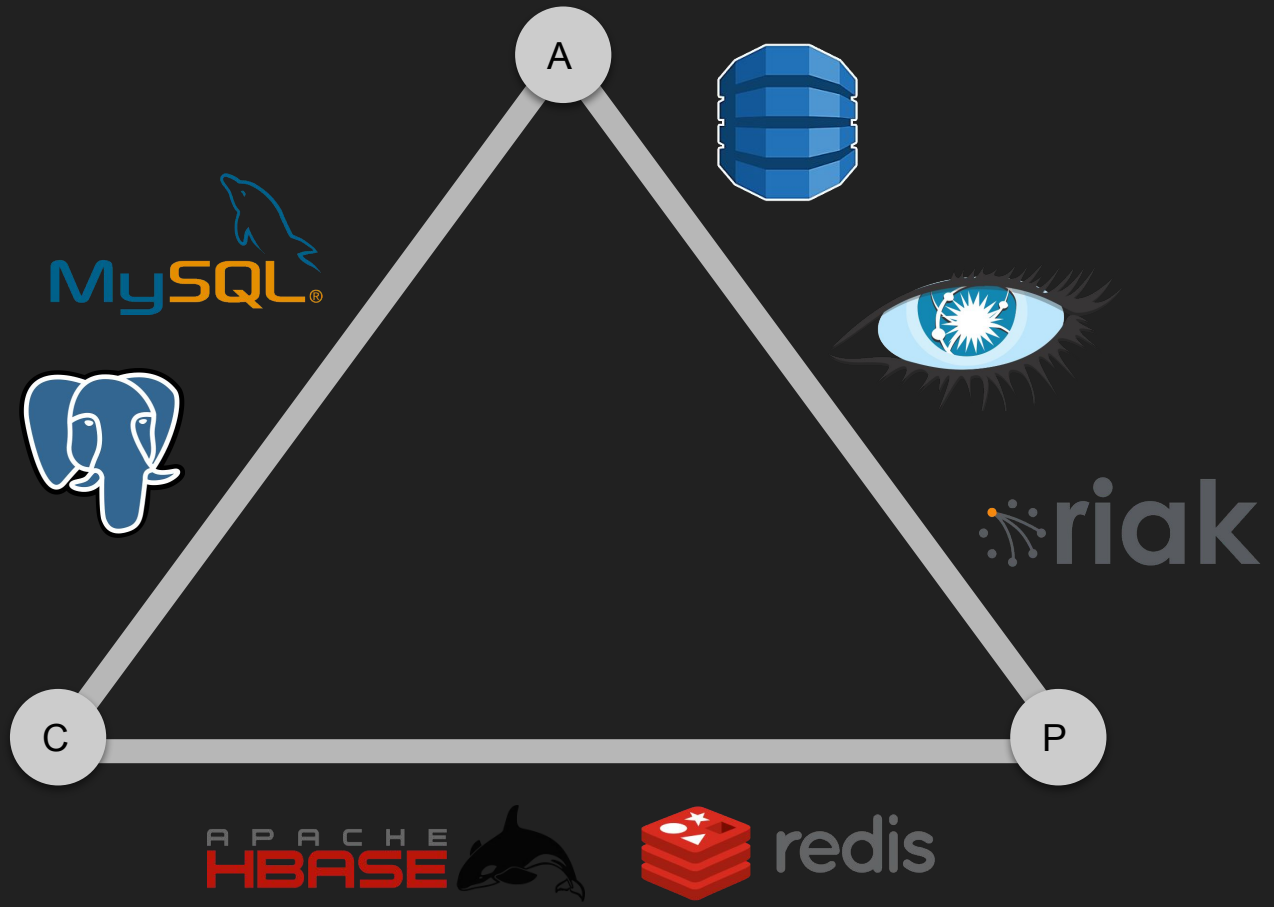
Cassandra Unboxed

Inhalt

- Position im CAP Theorem
- Tunable Consistency
- Lightweight Transactions
- Paxos
- Batches
- Hinted Handoff
- Tracing
- Fazit



Cassandra im CAP Theorem



Bilder: Wikipedia.org
https://www.youtube.com/watch?v=82wuPR5exmM

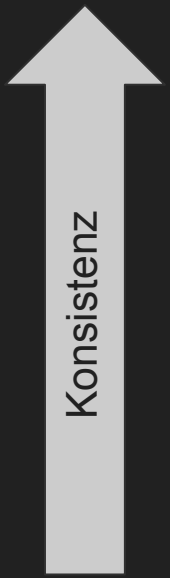
Wiederholung: Tunable Consistency

- ALL
- EACH_QUORUM
- QUORUM
- LOCAL_QUORUM
- ONE
- TWO
- THREE
- LOCAL_ONE
- ANY

Wiederholung: Tunable Consistency

Kein Option gegen

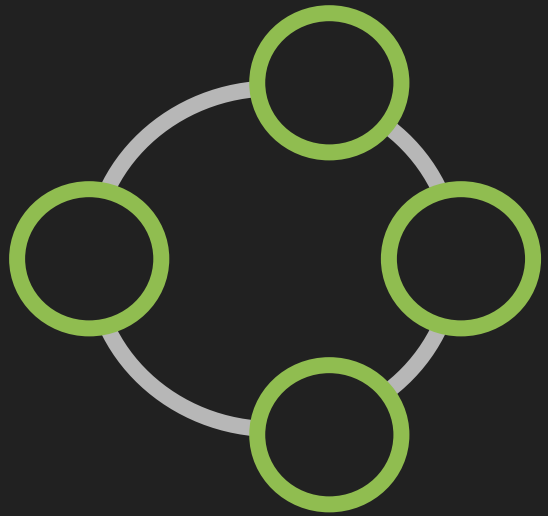
Race Conditions



- ALL
- EACH_QUORUM
- QUORUM
- LOCAL_QUORUM
- ONE
- TWO
- THREE
- LOCAL_ONE
- ANY

Transaktionen in Cassandra

- Keine klassischen Transaktionen im Sinne von ACID
- Nicht trivial in verteilten Systemen
- Aber zwei Mechanismen für ein gewisses transaktionales Verhalten
- Lightweight Transactions für serial isolation
- Batches für Atomacy



Lightweight Transactions (LWT)

- Compare-And-Set (CAS) oder Read-before-Write
- Setzt Konsistenz Level auf **`SERIAL`** (`== QUORUM`)
- Linearizable consistency - auf eine Partition beschränkt
- Latenz vervierfacht sich ⇒ LWTs sehr teuer
- Blockiert keine normale Lese- und Schreiboperationen, lediglich andere LWTs

```
[INSERT | UPDATE] ... IF ...[<, <=, >, >=, !=] ... ;
```

LWT Beispiele

```
INSERT INTO USERS (email, password)
values ('nosql@hbrs.de', 'a3cca2b' )
IF NOT EXISTS;
```

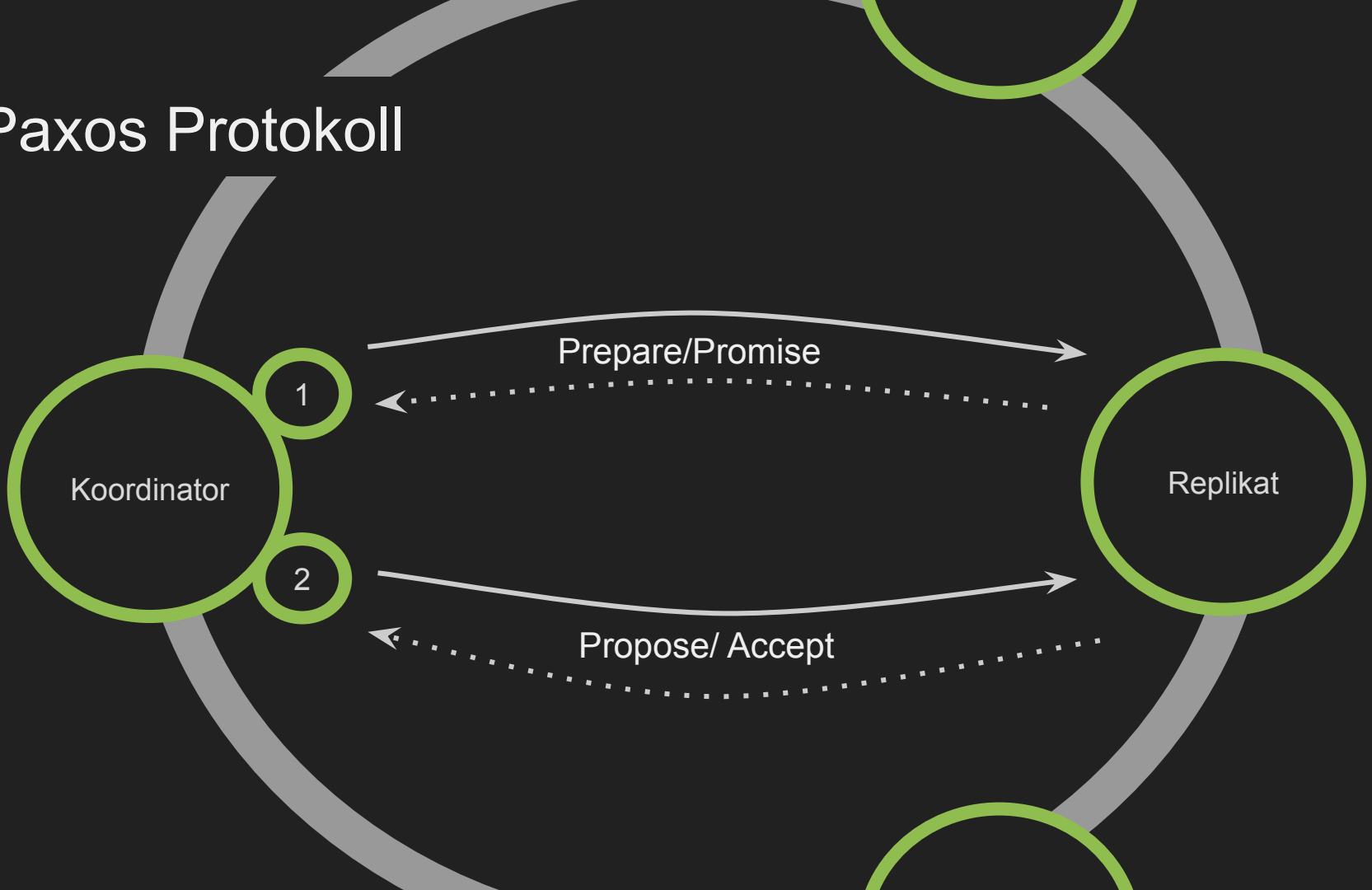
```
UPDATE users SET
reset_token = null, password = 'newpassword'
WHERE email = 'nosql@hbrs.de'
IF reset_token = 'some-generated-reset-token';
```

https://www.datastax.com/blog/2013/07/lightweight-transactions-cassandra-20

LWT Background: Paxos Protokoll

- Frühere Versuche mit Sperrverfahren ⇒ Lost-Updates
- Seit Version 2.0 ⇒ Paxos
- Ermöglicht Einigung auf einen Wert ohne Master
- Alternative zum klassischen Zwei-Phasen-Commit Protokoll
- Erweiterung des Protokolls für Cassandras LWT/CAS
- Status der Transaktionen in system.paxos Tabelle

Paxos Protokoll



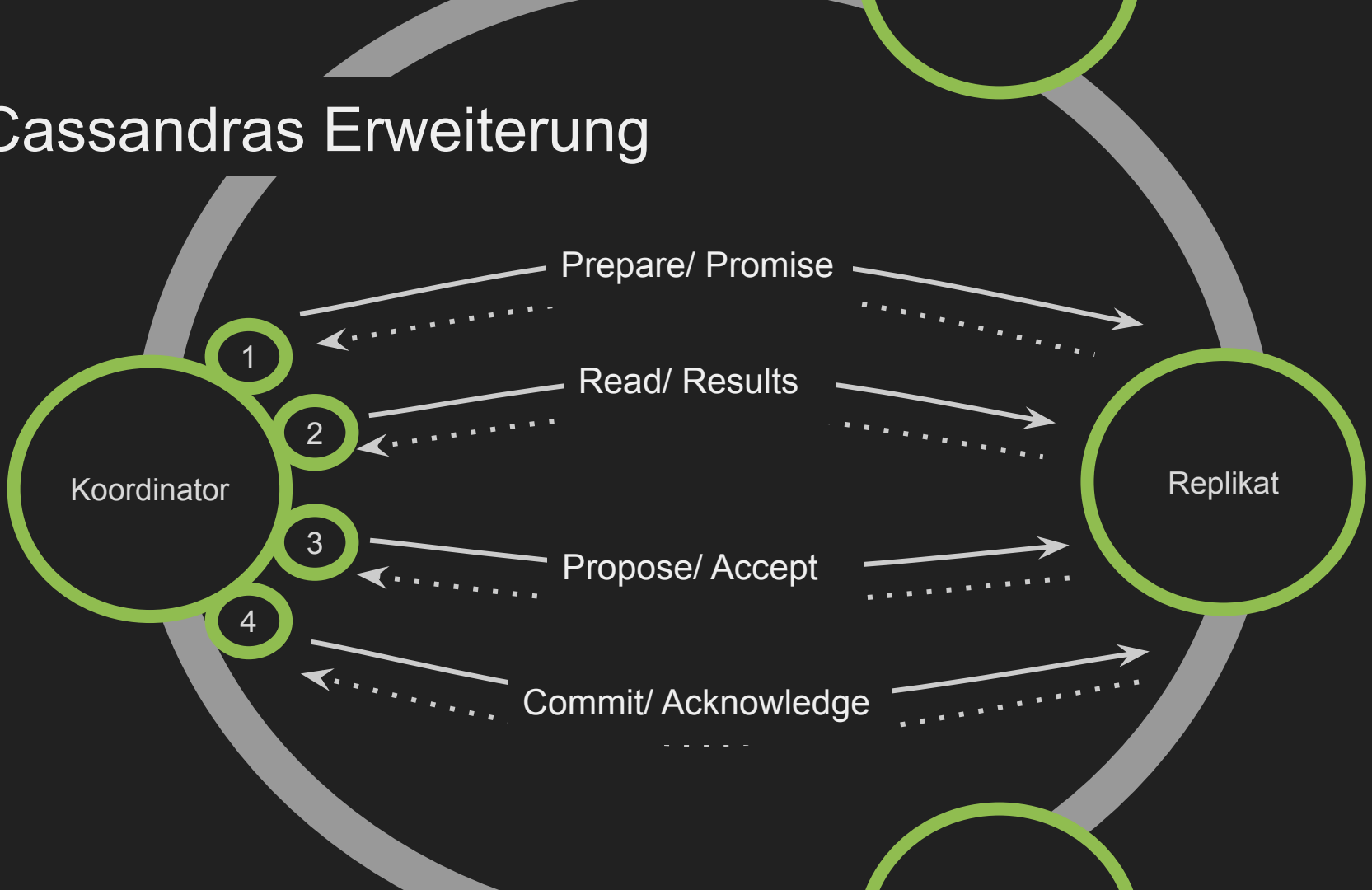
Cassandras Implementierung/ Erweiterung

- Akzeptierter Wert soll daher in Cassandra Speicher

  ⇒ Zusätzliche Phase am Ende: Commit/ Acknowledge
- Wegen CAS muss dieser Wert auch gelesen werden können

  ⇒ Weitere Phase: Read/ Results
- Paxos State kann bei Konflikten zurück gesetzt werden

Cassandras Erweiterung



Batches

- Für eine einzelne Partitionen nativ Atomar (All-Or-Nothing) & Isoliert
- Atomarität auch über mehrere Partitionen hinweg konfigurierbar ⇒ *batchlog*
- Keine Isolationsgarantie für alle Partitionen ⇒ Dirty Read
- Status der atomaren Batches in system.batchlog Tabelle
- Mögliche Operationen: `INSERT, UPDATE, DELETE`

```
BEGIN BATCH
   statement ;
   [statement; ...]
APPLY BATCH;
```

Kombination aus LWT und Batch

- Durchaus möglich
- Bedingung (IF) muss true zurückgeben, oder Batch schlägt fehl

```
BEGIN BATCH
    INSERT INTO ... VALUES ...  IF NOT EXISTS;
    INSERT INTO ... VALUES ... ;
APPLY BATCH;
```

Fazit

- LWTs und Batches sorgen für hohe Performance Einbußen
- Nur mit Bedacht nutzen und wenn keine andere Möglichkeit existiert

Für transaktionsorienterte Anwendungen
ist Cassandra eher ungeeignet

https://blog.pythian.com/cassandra-use-cases/

Vielen Dank

Fragen?