Strong and Efficient Consistency with Consistency-aware Durability

Seminar Supercomputer: Forschung und Innovation

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik

Sebastian Willmann, 05.12.2023

Betreuer: Janek Squar, Anna Fuchs

Gliederung

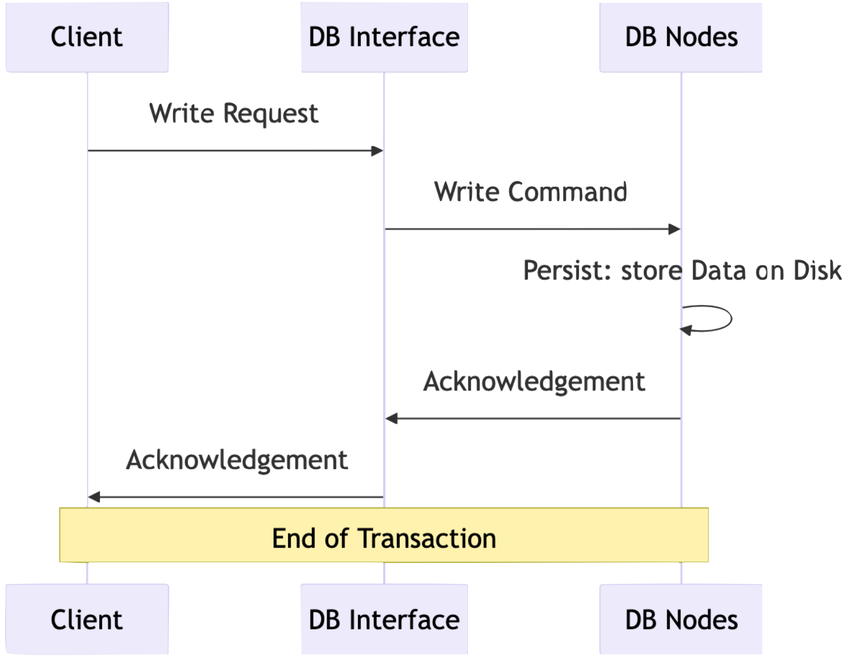
- Was ist Dauerhaftigkeit (Durability)?
- Was ist Konsistenz (Consistency)?
- Grundidee von CAD (Consistency-aware Durability)
- Funktionsweise
- Probleme, Nachteile und Hürden
- Umsetzung in ORCA
- Performance und Benchmarks
- Zusammenfassung
- Literatur

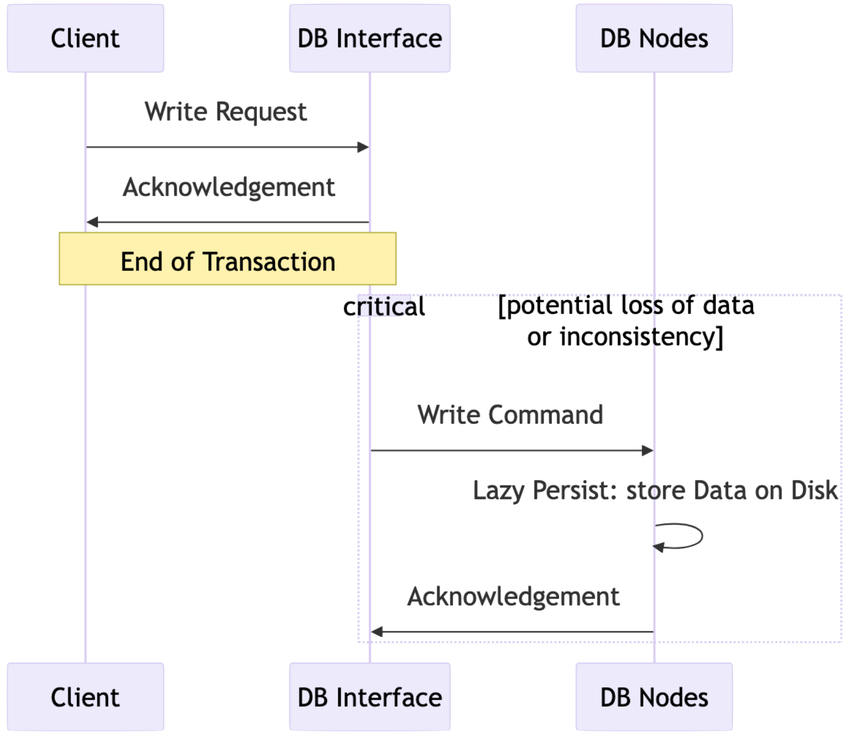
Was ist Dauerhaftigkeit (Durability)?

- Kommt aus dem ACI**D**-Konzept für Transaktionen
- „Once a transaction is committed, it cannot be abrogated“- Gray, 1981
- => Wie könnten die Daten verloren gehen, wenn die Transaktion vorbei ist?
- Man sagt, Daten sind konsistent, wenn sie (strom-) ausfallresistent sind
- In verteilten Systemen, entweder auf einer Mehrheit oder allen nichtflüchtigen Speichern
- Die Anwendung bestimmt das benutzte **Dauerhaftigkeits-Modell**

Was ist Dauerhaftigkeit (Durability)?

Dauerhaftigkeits-Modelle





Sofortige Dauerhaftigkeit

- Sehr sicher mit hoher Konsistenz
- Bis zu 10x langsamer als andere Modelle

Eventuelle Dauerhaftigkeit

- Sehr schnell, meistens der Standard
- Schlechte Konsistenz, fehleranfällig

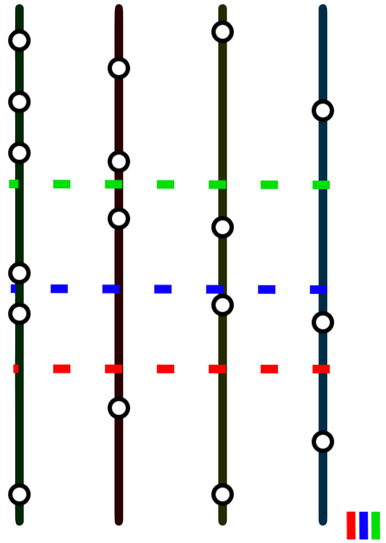
Was ist Konsistenz (Consistency)?

- Kommt aus A**C**ID: „The Transaction must obey legal protocols“ - Gray 1981
- Es wird nur die zeitliche, nicht die inhaltliche Konsistenz betrachtet
- „What does a read see given a previous set of reads and writes?“
- Sehr wichtig bei z.B. Wetterdaten-Speicherung, weniger wichtig bei z.B. CDN für statische Dateien
- Neues Konzept: Cross-Client monotonic reads
- Verschiedene Modelle je nach Anwendungsfall

Was ist Konsistenz (Consistency)?

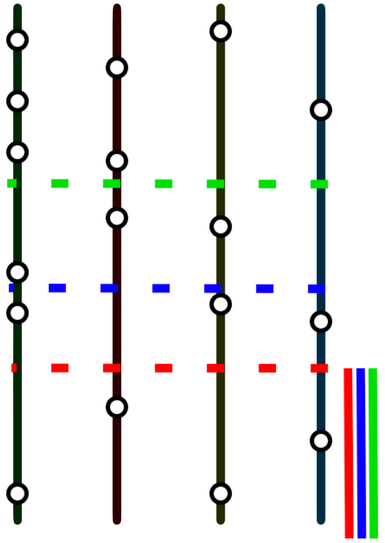
Verschiedene Konsistenz Modelle

<——Mehr Konsistenz, mehr Dauerhaftigkeit nötig Weniger Konsistenz, weniger Dauerhaftigkeit nötig ——>



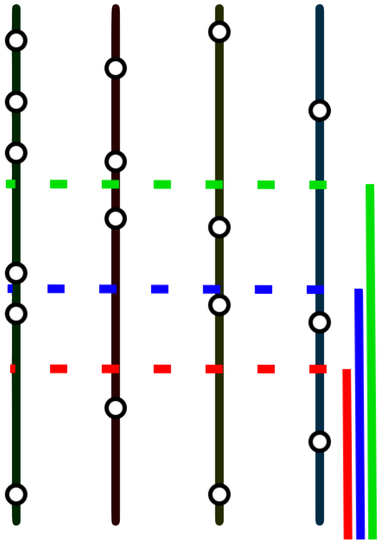
Linearizability

Perfekte Konsistenz, fordert aber sofortige Dauerhaftigkeit, simpel, sehr langsam



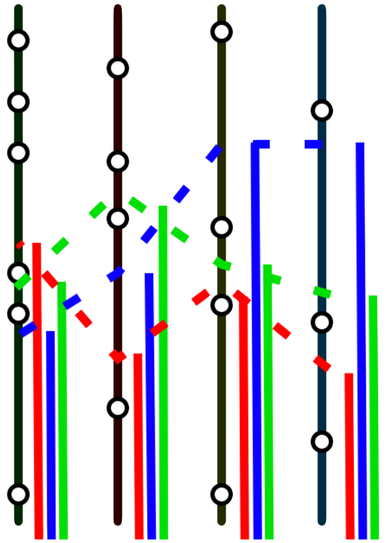
Cross-Client Monotonic Reads

Gute Konsistenz, komplexe Dauerhaftigkeit nötig, ist also aufwendiger, schnell



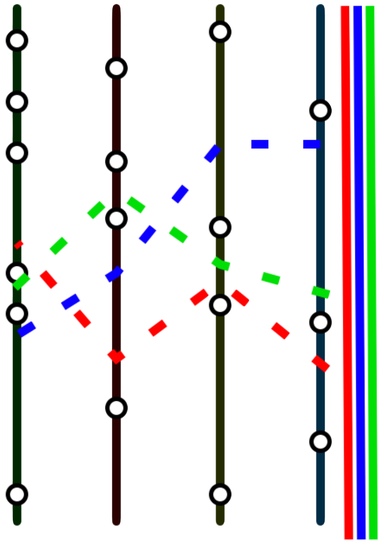
Monotonic Reads

Gute Konsistenz, Dauerhaftigkeit OK, etwas aufwendig, gute Geschwindigkeit



Causal Consistency

Akzeptable Konsistenz, Dauerhaftigkeit OK, nicht sehr aufwendig, sehr gute Geschwindigkeit



Weak Consistency

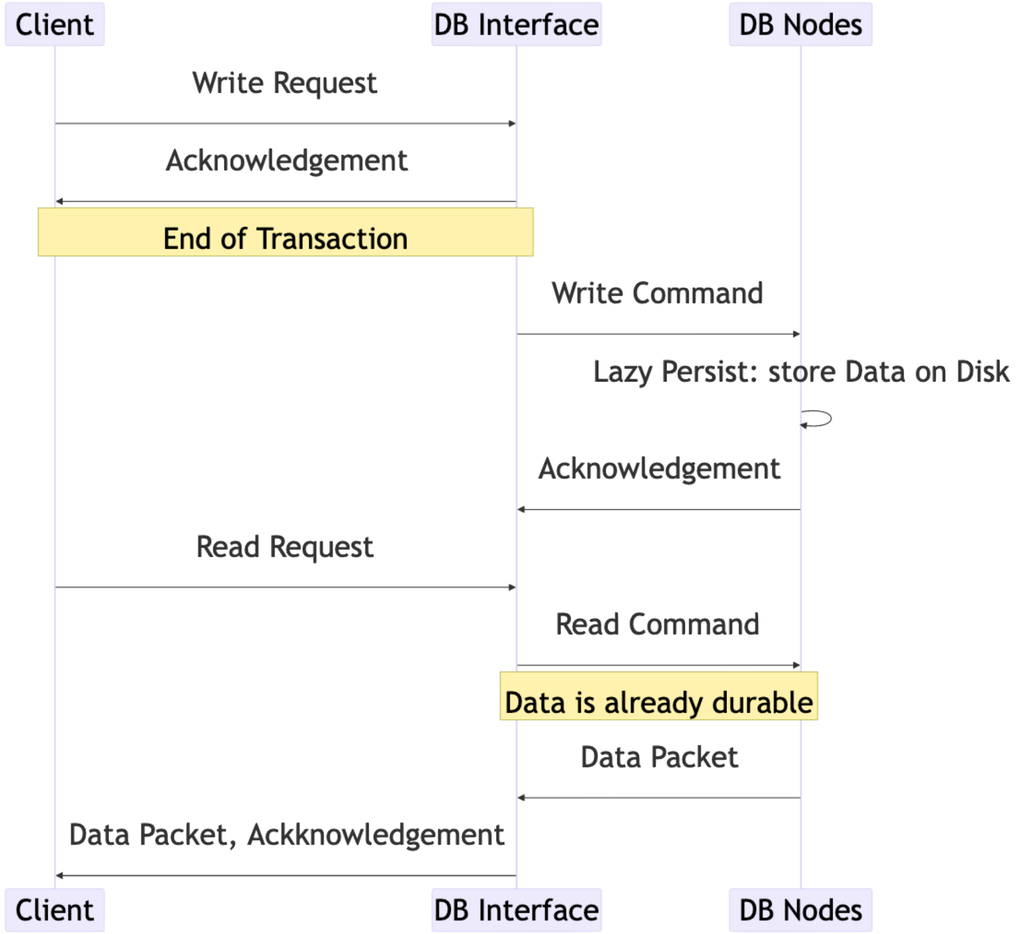
Sehr schlechte Konsistenz, irrelevante Dauerhaftigkeit, sehr einfach, extrem gute Geschwindigkeit, meist Standard

Grundidee von CAD

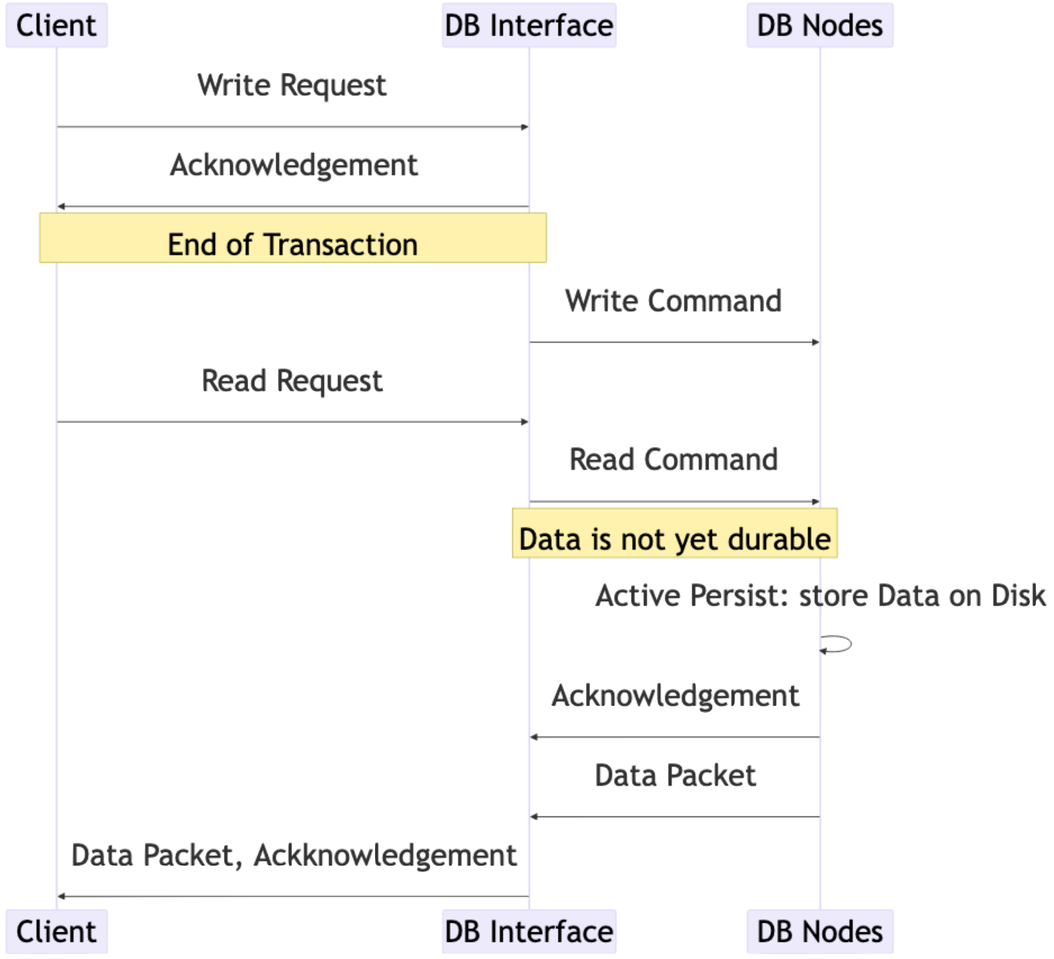
Wann genau ist etwas dauerhaft?

- Was bedeuten verschiedene Modelle (Dauerhaftigkeit und Konsistenz) für den Client?
- **Daten müssen erst dauerhaft sein, wenn sie gelesen werden**
- => Read/Write Performanz von Eventueller Dauerhaftigkeit
- => Hohe Konsistenz
- => Guter Throughput, weil von mehreren Nodes gelesen werden kann
- Aber: keine perfekte Dauerhaftigkeit, weil Daten vor einem Read verloren gehen können
- Nebeneffekt: Cross-Client Monotonic Reads

Funktionsweise



Wenn Daten schon dauerhaft sind, ist der Read wie erwartet sofort erfolgreich



Wenn Daten noch nicht dauerhaft sind, werden sie erst dauerhaft gemacht

Probleme, Nachteile, Hürden

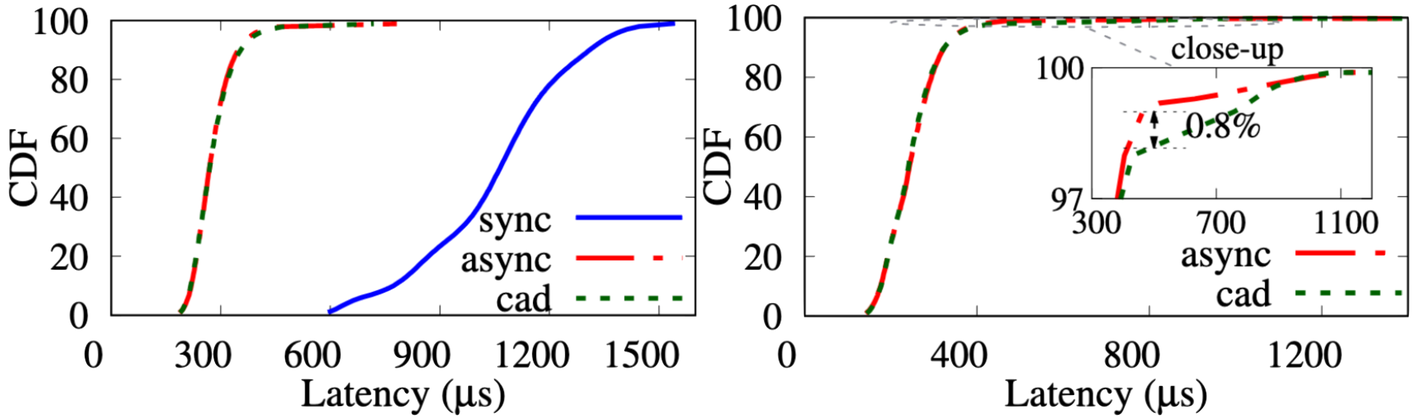
- Nicht Fail-safe; Writes können durch Ausfälle auch nach ACK gelöscht werden, wenn die Daten noch nicht lazily dauerhaft gemacht wurden und es noch keinen Read gab
- Das nötige System ist komplexer
- Nur eine Node als Interface zu benutzen bedeutet niedrigen Throughput, aber mehrere Interfaces sind schwieriger zu synchronisieren
- Dauerhaftigkeit? Auf allen Knoten => System so langsam wie der langsamste Knoten; Auf nicht allen Knoten => Es könnte zu stale Reads kommen
- => ORCA als Umsetzung von CAD

Umsetzung in ORCA

- CAD in ZooKeeper
- Leader-Based Majority System: Ein „Controller“, der den Zustand feststellt; Daten sind dauerhaft, wenn sie auf einer Mehrheit der Nodes dauerhaft sind
- Zwischen Leader und Nodes wird ein Heartbeat mit Infos ausgetauscht
- CCMRs sind immer garantiert, es kann aber zu stale reads kommen, Verfügbarkeit solange nicht mehr als die Hälfe der Nodes gleichzeitig ausfallen
- Benutzung eines Indexes um den Zustand linear dazustellen
- „Active Set“ von Nodes (mehr als Hälfte), die Dauerhaftigkeit definieren und Reads durchführen dürfen

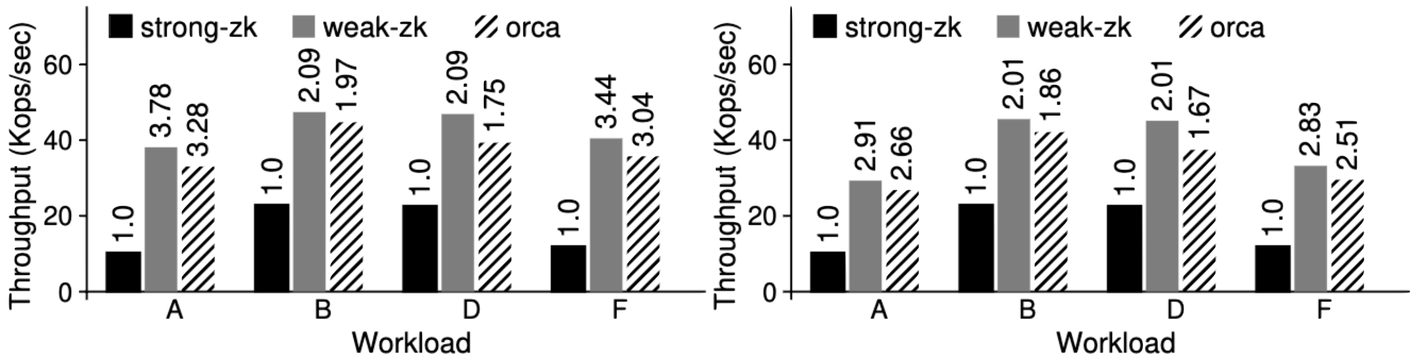
Performance und Benchmarks

ORCA vs andere Konsistenz-Modelle



(a) YCSB-A (write latencies)

(b) YCSB-B (read latencies)



(i) Baseline: async replication & persistence

(ii) Baseline: async persistence

(b) YCSB Macro-benchmark

- Read/Write Latenzen vergleichbar mit Eventueller Dauerhaftigkeit und Schwacher Konsistenz
- Throughput knapp unter Eventueller Dauerhaftigkeit und Schwacher Konsistenz

Performance und Benchmarks

Korrektheit von ORCA

- ORCA hält sich an CCMR, selbst wenn alle Knoten ausfallen oder hohe Latenzen haben
- Bei hohen Latenzen, zum Beispiel bei Geo-distributed Systems kann es aber zum Lesen von veralteten Daten kommen (Immer noch CCMR)

System	Outcomes (%)	
	Correct	Non-monotonic
weak-ZK	17	83
strong-ZK	100	0
sync-ZK-all	63	37
ORCA	100	0

(a) Async persistence

System	Outcomes (%)	
	Correct	Non-monotonic
weak-ZK	4	96
strong-ZK	100	0
sync-ZK-all	63	37
ORCA	100	0

(b) Async replication & persistence

Outcome(%)	Location-tracking			Retwis		
	weak-ZK	strong-ZK	ORCA	weak-ZK	strong-ZK	ORCA
Inconsistent	13	0	0	8	0	0
Consistent (old)	39	0	7	20	0	12
Consistent (latest)	48	100	93	72	100	88

Zusammenfassung

- CAD als Modell zwischen Dauerhaftigkeit und Konsistenz
  - Performance vergleichbar mit schwacher Dauerhaftigkeit und Konsistenz
  - Starke Dauerhaftigkeit und starke Konsistenz (Cross-Client Monotonic)
  - Guter Ersatz für Systeme, die Eventuelle Dauerhaftigkeit benutzen
- ORCA als Umsetzung von CAD
  - Hoher Throughput durch Active Set; Mehrere Knoten für Reads
  - Funktioniert in Case Study mit versprochener Performance

Literatur

- https://www.usenix.org/system/files/fast20-ganesan.pdf (Genesan, 2020)
- https://doi.org/10.1007/BF01784241 (Ahamad, 1995)
- http://people.eecs.berkeley.edu/~kubitron/courses/cs262a-F21/handouts/papers/theTransactionConcept.pdf (Gray, 1981)

Bildquelle:

- https://www.usenix.org/system/files/fast20-ganesan.pdf