

Einstieg	Replikation	Availability	Consistency	CALM Theorem	CRDTs	Quellen	Abschluss	Einstieg	Replikation	Availability	Consistency	CALM Theorem	CRDTs	Quellen	Abschluss	Einstieg	Replikation	Availability	Consistency	CALM Theorem	CRDTs	Quellen	Abschluss
----------	-------------	--------------	-------------	--------------	-------	---------	-----------	----------	-------------	--------------	-------------	--------------	-------	---------	-----------	----------	-------------	--------------	-------------	--------------	-------	---------	-----------

Willkommen bei Verteilte Systemen!

Ziele

Von Datenbanken
über Webdienste
bis zu p2p und Sensornetzen.

- Ihr kennt verschiedene Arten der Replikation.
- Ihr versteht, dass Replikation zu Inkonsistenzen führen kann.
- Ihr kennt das CALM Theorem.
- Ihr versteht, dass Koordination vermieden werden kann und dies zu einfacheren Systemen führt.

Heute: Replikation, CALM und CRDTs.
Versprich nur, was du halten kannst.

Ablauf heute

- Replikation
- Was ist Availability?
- Welche Konsistenzmodelle gibt es?
- Lässt sich Koordination vermeiden?

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Replikation

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Replikation

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Single Leader

Replikation

Speichern von Kopien auf mehreren Maschinen, die über Netzwerk verbunden sind.

Gründe für Replikation:

- Geographische Skalierung: Daten eines Nutzers näher am Nutzer → Verringerung der Latenz
- Anwendung funktioniert trotz ausgefallenen Knoten.
- Größenmäßige Skalierung: Mehr Nutzer können die Anwendung gleichzeitig verwenden.¹

Annahme: Gesamter Datensatz passt auf eine Maschine → Keine Partitionierung (Sharding)

Übersicht Replikation

3 Arten von Replikation werden unterschieden:

- Single Leader
- Multi Leader
- Leaderless

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Single Leader

- Replika: Knoten, der eine Kopie speichert
- Leader: Eine Replika mit Schreibrecht
- Schreiben: Anfrage an Leader
- Leader schreibt lokal
- Sendet geänderte Daten an alle anderen Replikas (Follower)
- Follower speichern die Änderungen lokal
- Lesen auch von Followern

Hierarchie ähnlich zu NTP.

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Multi Leader

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Multi Leader

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Multi Leader

Multi Leader Replication

Nachteile Single Leader

- Leader nicht erreichbar ⇒ keine Änderungen
- Einzelner Leader → Flaschenhals

Anwendungen

- Progressive Apps: Offline arbeiten
- Kollaborative Apps: Etherpad, Cryptpad, Google Docs etc.

Nachteil Multi Leader

- Lösung von Schreibkonflikten nötig

Leaderless Replication

- Verbreitet durch Amazons Dynamo DB
- Auch Riak, Cassandra, Voldemort
- Writes auf jedem Knoten
- Meist „Quorum“ Reads and Writes.

Quorum

- Sende jeden write und read an n Knoten
- write ist erfolgreich wenn w Knoten ihn bestätigen
- read ist erfolgreich wenn r Knoten ihn bestätigen

Quorum Bedingung: $w + r > n$:

- garantiert Überlapp zwischen w-Knoten und r-Knoten
- $w < n$ kann bei ausgefallenen Knoten schreiben
- $r < n$ kann bei ausgefallenen Knoten lesen
- $w > \frac{n}{2}$ kann write-write Konflikte vermeiden

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Multi Leader

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

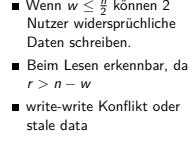
Multi Leader

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Quorum: Write-Write-Konflikte vermeiden

Zusammenfassung Replikation



- Wenn $w \leq \frac{n}{2}$ können 2 Nutzer widersprüchliche Daten schreiben.
- Beim Lesen erkennbar, da $r > n - w$
- write-write Konflikt oder stale data

- Single, Multi, Leaderless
- (a)synchrone Replikation
- Inkonsistenzen möglich
- Quorum Bedingung: $r + w > n$

- Total Available / High Available
- Sticky Available
- Unavailable

Literatur: Highly Available Transactions: Virtues and Limitations Bailis et al. (2013).

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Total Available / High Available

Sticky Available

- Antwort erhält, wer einen korrekten (nicht versagenden) Server kontaktieren kann
- Auch bei Netzwerkpartitionen zwischen Servern

- Antwort erhält, wer einen Server kontaktieren kann, der den gesamten, dem Nutzer bekannten Zustand beinhaltet
- Auch bei Netzwerkpartitionen zwischen Servern

- Daten auf mehrere Server repliziert
- Jede Replika enthält alle Daten
- Nutzer kontaktiert immer denselben Server
- ⇒ Sticky Available

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

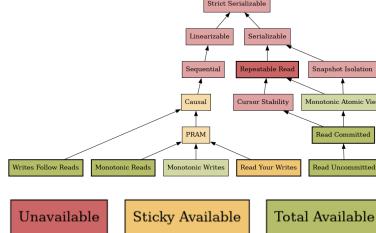
Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Availability

Unavailable

Consistency

System ist nicht verfügbar bei Netzwerkpartitionen.



Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Consistency

Consistency und Availability - Bewertung

- Spektrum möglicher Consistency und Availability
- Verschiedene Teile eines Systems können verschiedene Anforderungen haben
- Informierte Entscheidungen treffen!
- Unsere Anwendung muss nicht in jedem Fall 100% konsistent sein.
 - Manchmal reicht eine Entschuldigung.
 - Aber angrifffbar! (s. ACID Rain Paper (Warszawski und Bailis, 2017))
- Können wir uns auf Angaben von Herstellern verlassen?

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Quellen

Arne Babenhausenheide und Carlo Götz Replikation, CALM und CRDTs

Abschluss



Beispiele

- Können wir den Shopping Cart einer Webanwendung monoton gestalten?
Idee: Wir modellieren den Inhalt des Shopping Cart als Set.
Das Hinzufügen eines Artikels ist damit monoton und wir benötigen keine Koordination.
- Aber wie können wir einen Artikel aus dem Shopping Cart entfernen?
Idee: Wir verwalten ein Add-Set und ein Remove-Set.

Shopping Cart: Bewertungen

- Verwaltung des Shopping Carts ohne Koordination.
Brauchen Koordination, sobald wir den Einkauf tatigen.
 - > mussen sicherstellen, dass alle Anderungen vorher gesehen wurden.
 - > Koordination im System reduzieren, nicht komplett verhindern

Weitere CRDTs

- PN-Set: Counter für jedes Element, Wert des Counter entscheidet über Set-Zugehörigkeit.
 - 2P2P Graph: Je ein 2P Set für Knoten und Kanten.
 - Verschiedene Implementierungen von Registern.
 - Datentypen für kollaborative Textbearbeitung.
→ Conflict-free_replicated_data_type#Known_CRDTs



Zusammenfassung

- CRDTs können genutzt werden, um Koordination zu vermeiden oder zumindest einzuschränken.
 - CRDTs benötigen eine Form der Garbage Collection, um performance zu erhalten (z.B. Schnappschuss — „Keyframe“).
 - Garbage collection benötigt wiederum Koordination.
 - CRDTs werden in Verteilten Systemen eingesetzt: Riak, Redis, Dynamo
 - Ein nur auf CRDTs aufbauendere Algorithmus ist Koordinationsfrei



Quellen

- Highly Available Transactions: Virtues and Limitations; ([Bailis et al., 2013](#))
 - Weak Consistency: a generalized theory and optimistic implementations for distributed transactions; ([Adya and Liskov, 1999](#))
 - Designing Data-Intensive Application: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems; ([Kleppmann, 2017](#))
 - Keeping CALM: when distributed consistency is easy; ([Hellerstein and Alvaro, 2019](#))
 - A Comprehensive study of Convergent and Commutative Replicated Data Types; ([Shapiro et al., 2011](#))
 - Architecture of Open Source Applications; ([Brown and Wilson, 2011](#))



Ich wünsche Ihnen unkoordinierten Erfolg!



Verweise:

- theory and optimistic implementations for distributed transactions. PhD thesis:
<http://pmg.csail.mit.edu/papers/adya-phd.pdf>.

Bailis, P., Davidson, A., Fekete, A., Ghodsi, A., Hellerstein, J. M., and Stoica, I. (2013). Highly available transactions: Virtues and limitations. *Proc. VLDB Endow.*, 7(3):181–192.

Brown, A. F. and Wilson, G. (2011). The architecture of open source applications. PDF from <http://vgc.poly.edu/~juliana/pub/vistrails-aosa.pdf>.

Hellerstein, J. M. and Alvaro, P. (2019). Keeping CALM: when distributed consistency is easy. *CoRR*, abs/1901.01930.



Verweise 11

- Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

Shapiro, M., Preguiça, N., Baquero, C., and Zawirski, M. (2011). A comprehensive study of convergent and commutative replicated data types. PDF from
<https://hal.inria.fr/inria-00555588/document> or
<https://dsf.berkeley.edu/cs286/papers/crdt-tr2011.pdf>.



Verweise

- attacks on database-backed web applications. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD '17*, page 5–20, New York, NY, USA. Association for Computing Machinery.