DEEPSHORE

WHITE PAPER

PRIVATE BLOCKCHAIN:
WAS GILT ES BEIM AUFBAU ZU BEACHTEN?

PRIVATE BLOCKCHAIN: WAS GILT ES BEIM AUFBAU ZU BEACHTEN?

Die essenziellste Frage nach dem technisch-organisatorischen Aufbau eines privaten Blockchain-Systems kann im Grunde auf eine Fragestellung reduziert werden:

Wie ermittelt man die ideale Anzahl der Cluster-Knoten innerhalb einer geschlossenen Blockchain-Implementierung?

Diese Kernfrage zu beantworten, ist jedoch alles andere als trivial und die Antwort hängt von vielen Faktoren ab, die sich je Implementierung oder individueller Präferenz unterschiedlich darstellen können. Sie bildet aber genau deshalb eine sehr gute Ausgangssituation, um sich den vielfältigen Aspekten der Thematik zu nähern. Bei der folgenden Betrachtung handelt es sich aus diesem Grund auch um eine exemplarische Sicht, die die theoretischen Zusammenhänge beschreibt und auf Erfahrungen aus konkreten Projektdesigns basiert. Gewonnene Erkenntnisse können sich dabei aufgrund verschiedener Voraussetzungen zwischen einzelnen Installationen signifikant unterscheiden. Somit besteht die Notwendigkeit, sich im Vorfeld eines Systemaufbaus über dessen Zieldefinitionen und Rahmenbedingungen klar zu werden.

Die generelle Frage nach der Sicherheit von geschlossenen gegenüber öffentlichen Systemen wird in der folgenden Analyse gänzlich ausgeblendet. Eine Verortung der entsprechenden Grundgedanken findet sich in dem bereits veröffentlichen Beitrag aus dieser Reihe: „Sicherheit – private versus öffentliche Blockchain-Lösungen“. Bei allen nun folgenden Überlegungen sollte dennoch ein Fakt im Hinterkopf behalten werden:

Es gibt kein Computersystem auf der Welt, das absolute Sicherheit garantiert!

Es kann also nur darum gehen, wie groß eine Hürde oder die Wahrscheinlichkeit für erfolgreiche Manipulationen ist und was man tun muss, um sich unter Berücksichtigung der individuellen Situation „sicher genug“ zu fühlen.

Entscheidend für einen Systemaufbau sind meistens nachvollziehbare Gedanken und schlüssige Argumente gegenüber Dritten. Dabei kann es sich um einen Auditor, einen Wirtschaftsprüfer oder auch einen IT-Vorstand handeln, dem man Sinn oder Mehrwert einer Architektur erläutern muss. Also wie viele Blockchain-Knoten sollen es sein? Um das zu beantworten, bedarf es der Betrachtung mehrerer Dinge, deren Grundlagen bereits in vorangegangenen Beiträgen (Consensus 1, Consensus 2, Blockchain-Performance, Signatur) dieser Serie diskutiert wurden. Diese Erkenntnisse bilden die Grundlage für die nun folgenden Überlegungen. Dabei stehen folgende Themen im Fokus:

1.) Basisüberlegungen (CAP-Theorem)

2.) Consensus Algorithmus

3.) Mögliche Angriffsszenarien

4.) Betriebskonzepte

5.) Technische Eigenschaften und Konfigurationen der genutzten Technologie

Das Fundament – Blockchain, CAP-Theorem und Consensus

Eine wichtige Basisüberlegung ist, sich über einige Systemeigenschaften klar zu werden. Die klassische Blockchain verzichtet zugunsten von Verfügbarkeit und Ausfalltoleranz einzelner Instanzen auf die Datenkonsistenz im Sinne des CAP-Theorems (1). Das soll auch in unserem Beispiel der Fall sein. Aus diesem Grund gehen wir ab jetzt davon aus, dass unser Zielsystem hochverfügbar werden soll und zusätzlich auch noch den Verlust einzelner Entitäten verkraften kann.

Um alle nun folgenden Gedanken in einem einfach nachvollziehbaren und nicht zu komplizierten Rahmen zu halten, reduzieren wir die Anzahl der beteiligten Rechenzentren für unser Beispiel hypothetisch auf zwei. Dabei ist es unerheblich, ob sich diese „virtuell" bei einem Cloudanbieter oder in einer eigenen Umgebung (On-Premises) befinden. Die Anzahl der Rechenzentren verkompliziert lediglich die folgenden Schlüsse ein wenig, gleichwohl der Lösungsansatz zur Ermittlung der Knotenanzahl gleich bleibt.

Weiterhin soll der Consensus für alle folgenden Betrachtungen mit einem einfachen Quorum arbeiten. Das heißt, jeweilige Mehrheiten innerhalb eines verteilten Systems entscheiden über „richtig" oder „falsch" einer Information bzw. einer Transaktion.

Consensus-Mechanismen, also die Fähigkeit der beteiligten Instanzen eines verteilten Systems eine Übereinkunft über die Korrektheit enthaltener Informationen zu treffen und zu bewahren, beschäftigen zur Zeit Forscher und Entwickler von Blockchain-Lösungen auf der ganzen Welt. Allen voran steht der überaus erfolgreiche Proof-of-Work – als Teil des Consensus, den die Bitcoin-Blockchain nutzt – wegen seines enorm hohen Energiebedarfes in der Kritik. Eine Auseinandersetzung mit den jeweiligen Ansätzen zur Abbildung eines Konsenses – und damit der Schaffung von Vertrauen – würde Bücher füllen und wird im Folgenden nur dort betrachtet, wo eine Relevanz in Bezug auf die Eingangsfrage gegeben ist (2). Konkret also nur, wenn es um die sinnvollste Anzahl von Knoten in einem nicht öffentlichen Blockchain-System geht.

Wir merken uns also als Rahmenbedingungen für unser Beispiel:

- Es gibt zwei Rechenzentren.
- Eine einfache Mehrheit (Quorum) reicht zur Konsensbildung.
- Das System soll hochverfügbar (Availability im Sinne des CAP-Theorem) sein.
- Das System soll den Ausfall einzelner Instanzen verkraften können (Partition Tolerance im Sinne des CAP-Theorem).

(1) Dies gilt nicht, wenn ein Block validiert und auf allen Knoten im Cluster erfolgreich verteilt wurde, sondern lediglich für den Zeitraum, bevor dieser Zustand erreicht wurde.

(2) Siehe auch Beitrag: Consensus made by RAFT – oder Paxos? & Blockchain verbraucht zu viel Strom – Performance, Consensus und das CAP-Theorem

Die Grenzen von verteilten Systemen: das CAP-Theorem

Verteilte Systeme sind Rechnerverbünde, deren Cluster unabhängig voneinander agieren, die sich aber nach außen als ein System präsentieren. Der Wert solcher Systeme für die Verarbeitung von großen Datenmengen ist groß, aber auch verteilte Systeme haben ihre Grenzen. Das CAP-Theorem von Eric Brewer (Informatiker, USA) definiert diese Grenzen anhand von drei zentralen Anforderungen an verteilte Systeme:

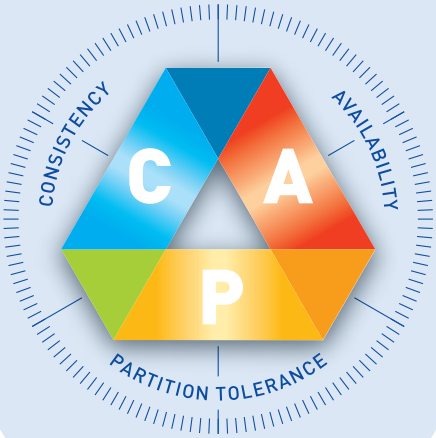
Consistency:
Volle Konsistenz ist gegeben, wenn alle Instanzen eines Clusters zum selben Zeitpunkt die selben Daten sehen.

Availability:
Volle Verfügbarkeit ist gegeben, wenn zu jedem Zeitpunkt alle Anfragen an das System beantwortet werden.

Partition Tolerance:
Volle Ausfalltoleranz ist gegeben, wenn das System auch bei Verlust von Partitionen arbeitsfähig bleibt.

DAS CAP-THEOREM:

In einem verteilten System ist es unmöglich, alle drei Anforderungen zu erfüllen:



Mögliche Angriffsszenarien – Byzantinischer Fehler

Betrachtet man mögliche Angriffsvektoren auf ein verteiltes System, landet man schnell bei dem Byzantinischen Fehlermodell (System-Attacke, Attacken-Abwehr). Ein Byzantinischer Angriff ist deshalb so herausfordernd, weil nicht klar ist, wer der Angreifer ist, da dieser sich quasi unbemerkt in ein System geschlichen hat und versteckt sein Unwesen treibt. An dieser Stelle sei noch einmal erwähnt, dass der Byzantinische Fehler in der Theorie darauf abzielt, ein System lahmzulegen oder inhaltlich zu manipulieren. Die Verteidigung gegen zum Beispiel inhaltliche Spionage, in einem geschlossenen, aber verteilten System, ist dabei eine weitere Herausforderung.

Beim Byzantinischen Fehler gilt die „>2/3“-Regel, dass heißt, mehr als 2/3 der vorhandenen Instanzen müssen loyal gegenüber dem Cluster sein, um voll arbeitsfähig zu bleiben (oder umgekehrt dürfen nur weniger als 1/3 der Knoten infiziert sein). Dekliniert man, dieser ersten Erkenntnis folgend, die mögliche Anzahl der Instanzen durch, ergibt sich folgendes Bild:

ANZAHL DER KNOTEN	ERKENNBARKEIT DES BYZANTINISCHEN FEHLERS
	Nicht gut, da „Single Point of Failure“. (3)
	Nicht gut, da durch die Übernahme einer Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich: 1/3 > 1/3).
	Nicht gut, da durch die Übernahme einer einzelnen Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich, da 1/3=1/3).
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann.
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/5 > 1/3) kann nicht kompensiert werden. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/6 = 1/3) kann nicht kompensiert werden kann. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust von bis zu zwei Instanzen durch das System verkraftet werden kann (2/7 < 1/3).

TABELLE 1 – ANZAHL KNOTEN NACH BYZANTINISCHEM FEHLERMODELL

(3) Auch eine Blockchain bewegt sich im Spannungsfeld des CAP-Theorems. In der Regel gelten für Implementierungen nicht Consistency, jedoch Availability und Partition Tolerance (im Sinne des CAP-Theorems).

Dieses Modell lässt sich nach dem skizzierten Muster beliebig fortführen und es ergibt sich ein erstes Bild zu der Frage, welche Anzahl von Knoten sinnvoll oder eher nicht sinnvoll sein könnte. Blendet man an dieser Stelle die Verteilung der Instanzen auf verschiedene Rechenzentren und damit auch die verbundenen Verfügbarkeitsmodelle aus, wäre ein Cluster mit mindestens 4 Knoten die erste sinnvolle Zahl. Zumindest aus der Perspektive, ein verteiltes System so aufzubauen, dass es überhaupt robust gegen einen Byzantinischen Angriff sein kann.

Betrieb – Administration

Kommen wir nun zum nächsten Punkt, der bei der zu eruierenden Entscheidung in Betracht gezogen werden sollte. Hierbei handelt es sich um die Frage, wie die Blockchain generell organisiert ist. Um den Nutzen eines dezentralen Systems ohne Master voll auszuschöpfen, sollte auch an dieser Stelle keine alleinige Masterinstanz vorhanden sein, die zum Beispiel die Konfiguration der Infrastruktur des Clusters allein verändern kann [4]. Somit muss es mindestens zwei administrative Instanzen geben, die gemeinsam die Infrastruktur kontrollieren. Eine Verteilung auf die Schultern von zwei Administratoren ist möglich, würde im tatsächlichen Betrieb aber zu einer Herausforderung führen. Entweder könnte eine Person allein eine Entscheidung (beispielsweise über die Erhöhung der Anzahl von Instanzen) treffen, was wiederum zu einem brisanten Angriffsvektor führen würde, da eine Person durch das Hinzufügen von Knoten das Mehrheitsgefüge beeinflussen könnte. Oder aber man wäre stets von der zeitgleichen Verfügbarkeit von beiden Administratoren abhängig, um Veränderungen vorzunehmen. Urlaub und Krankheit sprechen gegen letzteres Modell. Würde man die Aufgabe auf drei Personen verteilen, könnte man über eine Administrator-Mehrheit (zwei von drei) arbeiten. Dann müssten jedoch noch zwei Personen gleichzeitig verfügbar sein, aber ein krankheitsbedingter Ausfall bzw. die Urlaubsplanung könnte sich etwas entspannen. Alternativ wäre auch eine organisatorische Vertretungsregelung bei nur zwei Parteien denkbar. An diesem Punkt zeigt sich schnell, dass es keine einfache und perfekte Lösung gibt. Wenn wir von unserer initialen Annahme mit zwei Rechenzentren/Cloudservices ausgehen, wählen wir für unser Beispiel einfach das Zwei-Rollenmodell und gehen davon aus, dass es eine funktionierende Vertreterregelung innerhalb einer Rolle gibt. Folglich sind immer beide Stellen besetzt und Entscheidungen werden stets gemeinsam getroffen.

(4) Dies verbietet sich auch in manch rechtlichem Kontext, wenn es um die Systemsicherheit im Sinne der GoBD nach deutschem Recht geht. Dabei sollte es nicht möglich sein, dass eine einzelne Person ein System einfach manipulieren kann.

Legt man diese Erkenntnis über die Tabelle, die aus den Erkenntnissen der Byzantinischen Fehlertoleranz resultiert, erhält man folgendes Bild:

ANZAHL DER KNOTEN	ERKENNBARKEIT DES BYZANTINISCHEN FEHLERS	AUFTEILUNG BEI TECHNISCHEM BETRIEB MIT 2 ENTITÄTEN
	Nicht gut, da „Single Point of Failure“.	Bei 2 Admin-Entitäten nicht sinvoll
	Nicht gut, da durch die Übernahme einer Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich: 1/3 > 1/3).	je Admin = 1 Knoten
	Nicht gut, da durch die Übernahme einer einzelnen Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich, da 1/3=1/3).	Admin a = 2 Knoten Admin b = 1 Knoten
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann.	Admin a = 2 Knoten Admin b = 2 Knoten
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/5 > 1/3) kann nicht kompensiert werden kann. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.	Admin a = 3 Knoten Admin b = 2 Knoten
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/6 = 1/3) kann nicht kompensiert werden kann. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.	Admin a = 3 Knoten Admin b = 3 Knoten
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust von bis zu zwei Instanzen durch das System verkraftet werden kann (2/7 < 1/3).	Admin a = 3 Knoten Admin b = 4 Knoten

TABELLE 2 – ANZAHL INSTANZEN UND ADMINISTRATOREN

Folgt man dem Konzept der Byzantinischen Fehlertoleranz, könnte auch ein Administrator der Angreifer selbst sein. Somit könnte er allein zwar keine Änderungen am System durchführen und manipulieren, aber er wäre theoretisch in der Lage, die ihm zugewiesenen Knoten zu manipulieren und unter seine Kontrolle zu bringen. Denkt man diesen Gedanken zu Ende, dürfte ein Administrator allein auch nur Zugriff auf eine Anzahl von Instanzen haben, die kleiner als 1/3 der Gesamtzahl ist. Somit würde sich die theoretische Zahl der Administratoren auf mindestens vier erhöhen, um ein kleinst-mögliches System wirklich abzusichern. Warum? Nach dem Byzantinischen Fehlermodell wären vier Knoten die erste sinnvolle Anzahl von Instanzen bei der man überhaupt in der Lage wäre, einen Angriff auf eine singuläre Instanz abzuwehren. Würden die vier

Clusterinstanzen auf drei Administratoren aufgeteilt, hätte mindestens eine Person 2/4, also mehr als 1/3 der Instanzen unter ihrer Kontrolle und könnte damit das System zum Erliegen bringen.

In der Realität einer privaten Blockchain-Implementierung wird dieser Gedanke jedoch eher eine untergeordnete Rolle spielen, da die organisatorischen Aufwände zum Systembetrieb unverhältnismäßig ansteigen würden, um einen doch eher theoretischen Mehrwert an Sicherheit zu generieren. Man darf bei diesem Gedanken nicht vergessen, dass eine heimliche Attacke durch einen Angreifer durchaus komplex und aufwendig zu realisieren ist. Sollte ein Administrator über die kriminelle Energie und gleichermaßen über die Fähigkeiten verfügen, solch einen Angriff zu lancieren, würde er gegebenenfalls auch andere bzw. einfachere Wege finden, seinen Auftraggeber zu schädigen. Insofern ist es sinnvoll, die Anzahl der einem Admin zugewiesenen Knoten nicht in eine direkte Abhängigkeit zum Byzantinischen Fehlermodell zu stellen, sondern lediglich darauf zu achten, dass ein Administrator alleine nicht in der Lage ist, grundlegende Mechanismen des Systems durch einseitige Änderungen vorzunehmen (wie Neukonfiguration des Consensus oder auch das Hinzufügen neuer Instanzen). Das Vier-Augen-Prinzip entspricht dem in der Wirtschaft heute gängigen Sicherheitsstandard für den Betrieb großer Systeme und sollte der Mindeststandard sein.

Technische Eigenschaften und Konfigurationen der genutzten Technologie

Genauso wichtig wie die vorangegangenen Überlegungen ist die Frage, ob und wie ein Blockchain-Fork im Rahmen einer potenziellen „Split Brain"-Situation (netzwerktechnische Trennung der Rechenzentren) vermieden werden kann. Diesem Problem liegt die Annahme zugrunde, dass aus Verteilungs- und Sicherheitsgründen nicht alle Instanzen eines Clusters in derselben Lokation/Netzwerksegment betrieben werden. Nehmen wir einmal an, dass alle Knoten gleichverteilt auf unsere zwei Rechenzentren A und B ihre Arbeit verrichten und diese beiden Entitäten unabhängig voneinander lauffähig sind. Sollte die Verbindung zwischen den Standorten verloren gehen, würde jeweils eine Teilmenge des Gesamtclusters innerhalb eines Rechenzentrums autark weiterarbeiten können. Hätten wir ein Cluster mit vier Knoten, von dem jeweils zwei Instanzen in Lokalität A und zwei Instanzen in Lokalität B installiert sind, würden sich bei einem Netzwerkausfall nur jeweils zwei Knoten sehen. Sofern diese beiden vom Rest isolierten Instanzen in der Lage wären, einen Consensus herzustellen, könnten in jedem Rechenzentrum disjunkte Blockchains fortgeschrieben werden (5). Es entsteht ein Fork (6). Wenn die Netzwerkverbindung wiederhergestellt würde, würde das System bemerken, dass es über zwei unterschiedliche Ketten verfügt. Was nun? Ein gängiges Verfahren ist, dass

(5) Dieser Überlegung liegt die Annahme zugrunde, dass ein System sein Quorum (Mehrheit) auf Basis der grade verfügbaren Instanzen berechnet. Sofern ein 4-Knoten Cluster in zwei Hälften geteilt wird, wäre das Quorum 2 von 2 Knoten je Rechenzentrum (51 % von 2 > 1). D.h. in jedem Rechenzentrum könnten beide verbleibende Knoten autark vom Rest des Systems weiterarbeiten.

(6) siehe auch: CAP-Theorem

die längere Kette einfach „gewinnt“ und als gültig übernommen wird. Was passiert aber mit den Daten der kürzeren Kette? Diese Frage muss sich jeder Verantwortliche dringend vor einer technischen Implementierung stellen und prüfen, welchen Mechanismus die verwendete Blockchain hier wählt, bzw. welche fachlichen Konsequenzen sich daraus ergeben.

Um ein solches Szenario zu vermeiden, sollte ein Fork generell vermieden werden. Das kann durch eine geeignete Konfiguration der Blockchain in Kombination mit der Infrastrukturwahl geschehen. Der Consensus könnte zum Beispiel auf die absolute Zahl (und nicht auf die relative Zahl) aller Cluster-Knoten bezogen werden. Demnach könnten in dem vorangegangenen Beispiel zwei Instanzen allein keinen Konsens erlangen, für den Fall das sie netzwerktechnisch vom Rest getrennt werden, da zwei nicht die Mehrheit von vier ist. Dies würde jedoch dazu führen, dass dieses 4-Knoten-System nicht mehr hochverfügbar im Sinne des CAP-Theorems arbeiten könnte, da der Ausfall eines Rechenzentrums (und damit von zwei Instanzen) nicht kompensiert werden könnte. Ergo, im Fehlerfall könnten auch keine neuen Blöcke entstehen. Geht man davon aus, dass einer der großen Vorteile einer Blockchain-Implementierung dessen Verfügbarkeit und die Ausfalltoleranz einzelner Partitionen ist, wären vier Knoten in zwei Rechenzentren keine gute Wahl, sofern sich das System so verhält, wie grade beschrieben wurde. Bei fünf Instanzen in derselben Infrastruktur könnte das Problem umgangen werden. Aber dann gelte es abzuwägen, denn nach dem Byzantinischen Fehlermodell böte dieses System keinen Vorteil gegenüber der 4-Knoten-Variante. In diesem Fall würde sich also der Wechsel zur nächst höheren und ungeraden Instanzenanzahl lohnen, die „sinnvoll“ gemäß der Byzantinischen Fehlertoleranz ist. In dem Fall also 7. Im konkreten Beispiel würden sich also drei und vier Knoten auf die beiden Rechenzentren aufteilen und durch jeweils eine administrative Entität kontrolliert werden. Bei solch einem Modell wird quasi per Design vermieden, dass ein Blockchain-Fork überhaupt entstehen kann, ohne dabei die Systemverfügbarkeit einzuschränken. Eine entsprechende Zusammenfassung aller Gedanken findet sich in Tabelle 3 – Anzahl Instanzen, Administratoren und Rechenzentren.

Wie im oberen Absatz schon angedeutet, gibt es aber auch technische Mechanismen, die mit einem Chain-Fork umgehen können. So gibt es Implementierungen, die zum Beispiel Transaktionen der kürzeren Kette nicht verwerfen, sondern diese wieder als neue noch zu validierende Transaktionen an die längere Kette hängen. Das bedeutet allerdings, dass Transaktionen, die vorher einem Client durch die kürzere Kette als „validiert“ gemeldet wurden, nun plötzlich wieder „offen“, gegebenenfalls sogar kurzzeitig von außen nicht sichtbar sind. Ferner ergibt sich das theoretische Risiko, dass diese Daten durch die Verkettung unglücklicher Umstände vollständig verloren gehen. Das könnte zum Beispiel der Fall sein, wenn sich ein Hardware-Defekt bei genau den Instanzen ereignete, welche zuvor die kürzere Kette gerechnet hatten. Genauer gesagt, müsste sich ein Defekt genau in dem Augenblick ereignen, an dem die längere Kette nach der Wiedervereinigung das Regiment im Cluster übernommen hat und die „alten“ Informationen der kurzen Kette, zeitgleich zu „neuen“ (noch nicht validierten) Einträgen erklärt werden, bevor diese im Cluster repliziert

wurden. Das mag recht unwahrscheinlich klingen, ist aber nicht ausgeschlossen. – Ein Sechser im Lotto ist auch sehr unwahrscheinlich und trotzdem wird der Jackpot regelmäßig geknackt.

ANZAHL DER KNOTEN	ERKENNBARKEIT DES BYZANTINISCHEN FEHLERS	AUFTEILUNG BEI TECHNISCHEM BETRIEB MIT 2 ENTITÄTEN	AUFTEILUNG UND HA/FORK BEI 2 RECHEN-ZENTREN
	Nicht gut, da „Single Point of Failure“.	Bei 2 Admin-Entitäten nicht sinvoll	Nicht sinnvoll
	Nicht gut, da durch die Übernahme einer Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich: 1/3 > 1/3).	je Admin = 1 Knoten	RZ A: 1 Knoten RZ B: 1 Knoten
	Nicht gut, da durch die Übernahme einer einzelnen Instanz das System zum Erliegen kommen könnte (keine Quorumbildung möglich, da 1/3=1/3).	Admin a = 2 Knoten Admin b = 1 Knoten	RZ A: 2 Knoten RZ B: 1 Knoten
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann.	Admin a = 2 Knoten Admin b = 2 Knoten	RZ A: 2 Knoten RZ B: 2 Knoten Fork möglich oder keine HA
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/5 > 1/3) kann nicht kompensiert werden kann. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.	Admin a = 3 Knoten Admin b = 2 Knoten	RZ A: 3 Knoten RZ B: 2 Knoten Fork kann vermieden werden; HA möglich
	Bedingt sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust einer Instanz durch das System verkraftet werden kann. Der Verlust von 2 Instanzen (2/6 = 1/3) kann nicht kompensiert werden kann. Es ergibt sich somit kein Mehrwert gegenüber einem 4-Instanzen-Cluster im Sinne der Byzantinischen Fehlertoleranz.	Admin a = 3 Knoten Admin b = 3 Knoten	RZ A: 3 Knoten RZ B: 3 Knoten Fork möglich oder keine HA.
	Sinnvolle Anzahl im Sinne einer Byzantinischen Fehlertoleranz, da der Verlust von bis zu zwei Instanzen durch das System verkraftet werden kann (2/7 < 1/3).	Admin a = 3 Knoten Admin b = 4 Knoten	RZ A: 3 Knoten RZ B: 4 Knoten Fork kann vermieden werden; HA möglich

TABELLE 3 – ANZAHL INSTANZEN, ADMINISTRATOREN UND RECHENZENTREN

Zusammenfassung

Erkenntnisse, die sich aus den skizzierten Überlegungen ableiten lassen: In unserem fiktiven Beispiel sollte ein System verfügbar und ausfalltolerant im Sinne des CAP-Theorems sein. Weiterhin wurde die Nutzung zweier getrennt voneinander organisierter Rechenzentren angenommen. Der Cluster sollte ferner mit einem einfachen Mehrheits-Consensus arbeiten, der auf Basis der absoluten Anzahl von Instanzen ermittelt wird. Unter diesen Gesichtspunkten bietet sich eine initiale Implementierung mit zunächst sieben Clusterknoten an. Das System würde dabei von wenigstens zwei administrativen Instanzen betrieben, die übergreifende Konfigurationen (zum Beispiel Anzahl neuer Knoten) und Consensus-Einstellungen nur einstimmig beschließen können. Somit ergibt sich eine Architektur, die hochverfügbar und auch robust gegen den Ausfall einzelner Partitionen ist. Sogar eine einfache Attacke im Sinne des Byzantinischen Angriffs könnte abgewehrt werden und die Entstehung von Chain-Forks (und damit potenzieller Inkonsistenzen/Datenverluste) wäre unter normalen Umständen ausgeschlossen.

Neben diesen technischen und organisatorischen Überlegungen spielt in der realen Welt natürlich auch die Summe der Replikationen und damit die Kostenstruktur der n-fach vorgehaltenen Daten eine Rolle. Da es sich hierbei aber um eine rein wirtschaftliche Betrachtung handelt, wird auf eine Bewertung dieses Aspektes verzichtet. Klar ist, dass die endgültige Entscheidung für oder gegen eine Architektur auch unter ökonomischen Gesichtspunkten zu treffen ist. Insofern spielt die Abwägung zwischen Sicherheit und Aufwand bzw. den Kosten einer Lösung natürlich eine Rolle bzw. ist sie individuell zu treffen.

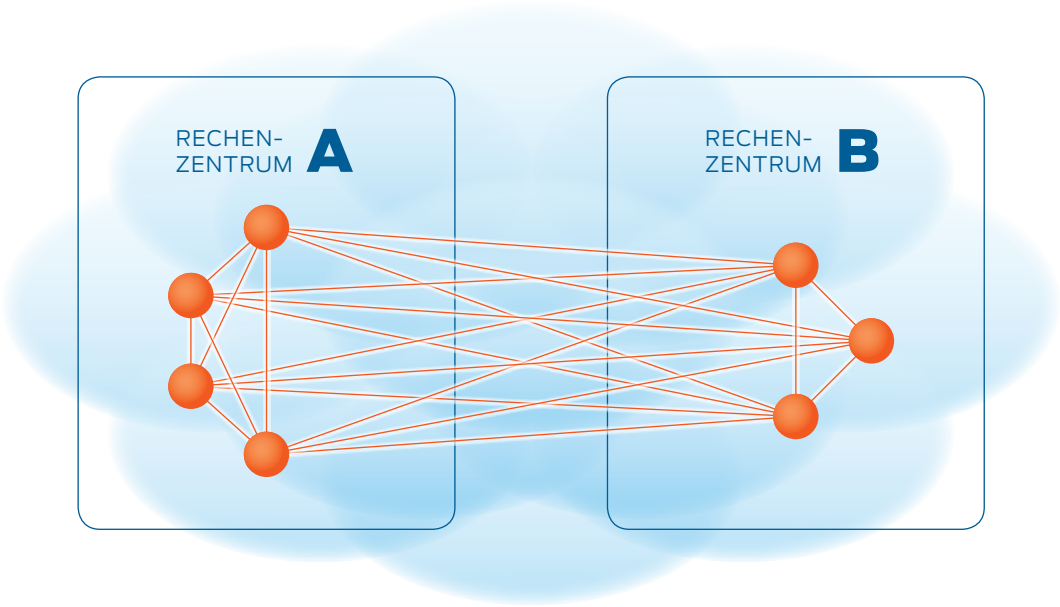


ABBILDUNG 2 – CLUSTER MIT 7 KNOTEN IN ZWEI RECHENZENTREN

KONTAKT:

DEEPSHORE

Deepshore GmbH
Baumwall 3, 20459 Hamburg
Tel +49 40 46664 296
www.deepshore.de