Algorithmen für verteilte Systeme in unbemannten Luftfahrzeugen

Artsiom Kaliaha

BACHELORARBEIT

eingereicht am

Fachhochschul-Bachelorstudiengang

Automotive Computing

in Hagenberg

im Februar 2021

Betreuung:

Mag. Dipl.-Ing. Dr. Andreas Müller B.Sc.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt. Die vorliegende, gedruckte Arbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Hagenberg, am 1. Februar 2021

Artsiom Kaliaha

Inhaltsverzeichnis

Kurzfassung

Drohnen ändern den Alltag auf überraschende Weise: sie retten in Katastrophengebieten, sichern die medizinische Versorgung in Konfliktzonen, reduzieren Verkehr und Emissionen und machen riskante Berufe sicherer. Heutzutage werden Drohnen vielseitig eingesetzt. Andere Gebiete, bei denen Drohnen zum Einsatz kommen, sind das Militär (Steuerung von Drohnenschwärmen), der Telekommunikationsbereich (schneller Aufbau von Netzwerken auf Nachfrage) und die Ausbildung (Aufnahme von 360-Grad-Videos).

Es gibt mehrere Forschungsrichtungen, die Funktionsweise einzelner Drohnen und Drohnenschwärmen optimieren. Eine von dennen ist Swarming. Swarming ist ein Bereich von Multiagentensystem, der Drohnen ermöglicht, bestimmte Tierarten nachzuahmen. Die Schwarmintelligenz ermöglicht es hunderten, wenn nicht tausenden von Drohnen, zusammenzuarbeiten, um herausfordernde Aufgaben zu erledigen. Heutzutage forscht die gesamte Robotikindustrie auf dem Gebiet der kollaborativen Robotik, bei der Roboter nebeneinander arbeiten und von Menschen trainiert werden. Schon in Naher Zukunft werden wir aber in der Zeit der Cloudrobotik leben, in der Roboter ohne menschlicher Interaktion als ein einziger Organismus agieren und denken werden. Fortschritte in den Bereichen der Künstlicher Intelligenz und der Cloudrobotik treiben die Schwarmtechnologie an und werden dazu beitragen, dass Drohnen nicht nur mit ihren Operatoren, sondern auch miteinander kommunizieren werden. Heutzutage wird jede Drohne von einem oder mehreren Menschen gesteuert. Die Drohnen von morgen werden möglicherweise überhaupt keine Operatoren brauchen. Forscher untersuchen Möglichkeiten, Menschen durch Künstliche Intelligenz und Algorithmen für verteilte Systeme zu ersetzen. Da eingebettete Systeme mit jedem Jahr leistungsstärker werden, haben sie genug Kapazitäten, um Sensordaten ohne Cloudverbindung zu bearbeiten, Rechenoperationen gleichmäßig untereinander aufzuteilen und Drohnen mit speziellen Funktionen im Schwarm redundant abzusichern.

Alle diesen Funktionen brauchen klassische Algorithmen für verteilte Systeme, die schon seit Jahren bei Cloudlösungen zum Einsatz kommen. Sie beruhen auf den Algorithmen aus den 80ern und 90ern, die immer noch in Produktivsystemen verwendet werden und als Inspiration für neue und optimalere Algorithmen dienen. Der Fokus dieser Bachelorarbeit liegt auf der Untersuchung verfügbarer Algorithmen für verteilte Systeme und auf der Recherche nach einem optimalen Concurrency Modell für robuste und fehlertolerante eingebettete Systeme.

Abstract

Drones change everyday life in surprising ways: they save in disaster areas, provide medicine in conflict zones, reduce traffic flow and emissions, and make risky jobs safer. Today, drones are used in many ways. Other areas where drones are used are the army (control of swarms of drones), the telecommunications sector (on demand network provisioning) and training (360-degree videos).

There are several research areas that focus on optimizing individual drones operation and swarms of drones control. One of these research topics is swarming. Swarming is a subfield of multi-agent systems that allows drones to mimic behavior of certain animal species. Swarm intelligence allows hundreds, if not thousands, of drones to work together to complete challenging tasks. Today the entire robotics industry is doing research in the field of collaborative robotics, which enables robots to work side by side with humans and to be trained by humans. In the near future, however, we will be living in the age of cloud robotics, in which robots will act and think as a single organism without human interaction. Advances in artificial intelligence and cloud robotics are driving swarm technology and will help drones communicate not only with their operators, but also with each other. Nowadays, each drone is controlled by one or more people. Tomorrow drones may not need operators at all. Researchers are investigating possibilities of replacing humans with artificial intelligence and algorithms for distributed systems. As embedded systems become more powerful each year, they get more and more capacity to process sensor data without cloud connection. Innovative approaches to building swarm systems suggest splitting computation evenly across all drones and redundantly securing drones which fulfill special functions in a swarm or carry unique hardware.

All of these functions require classic algorithms for distributed systems that were used in cloud solutions for years. They are based on the algorithms from the 80s and 90s, which are still used in production systems and serve as inspiration for more optimal algorithms. The goal of this bachelor thesis is to investigate available algorithms for distributed systems and find an optimal concurrency model for robust and fault-tolerant embedded drone systems.

Kapitel 1

Einleitung

Wir leben in der Zeit Künstlicher Intelligenz, Big Data und intelligenter Robotersysteme. Heutzutage ist die Vielfalt der Robotersysteme sehr beeindruckend: Roboter finden Einsatz in Form autonomer Autos, intelligenter Produktionstechnik und auch *Drohnen*. Vor 5 Jahren waren Drohnen noch manuell gesteuert, und erlaubten keine anspruchsvolle Rechenoperationen direkt auf der Hardware. Zukünftig sollen Drohnen leistungsfähiger sein und noch autonomer funktionieren und das nicht nur einzeln sondern auch in Schwärmen. Der Fokus dieser Arbeit liegt im Gebiet der Drohnenschwärme, genauer gesagt in der Betrachtung dieser als *verteiltes System*. Dadurch, dass Drohnen miteinander kommunizieren, können sie in einem Schwarm voll autonom und ohne Kollisionen fliegen. Die Kommunikation im Schwarm liefert in diesem Fall zusätzliche Informationen für die Funktion des autonomen Fliegens. Die Anwendung von Algorithmen für verteilte Systeme macht diese Bachelorarbeit auch dadurch interessant, dass diese Algorithmen mehr Anwendungsfälle für Interaktionen innerhalb eines Schwarmes und auch zwischen einem Schwarm und der Cloud bieten. Ein weiterer möglicher Anwendungsfall ist die Entlastung von Operatoren, die diese Drohnen normalerweise steuern. Die Entwicklung von Drohnen mit einem hohen Grad an Autonomie bietet den Operatoren im Laufe einer Mission bzw. eines Auftrages mehr Freiheit, andere nützliche Aufgaben zu erfüllen (Kamera ausrichten, Objekt beobachten, 3D Mapping des Geländes steuern, usw).

Fortschritte in Kommunikationstechnologien (vor allem 5G), Künstlicher Intelligenz und eingebetteter Hardware haben die zweite Welle von *Drohnentechnologie* ausgelöst. Manche Anwendungsfälle, die früher nicht denkbar waren, werden derzeit in Forschungsinstituten und Universitäten (in Europa vor allem EPFL [1] und ETH Zürich) erforscht. Innovative Startups versuchen, all diesen Anwendungsfälle auf den Markt zu bringen. Die Drohnenindustrie bietet bahnbrechende Anwendungsfälle und dadurch auch neue Herausforderungen.

Das Problem, das diese Arbeit löst befasst sich mit einer solchen Herausforderung, und zwar *der Synchronisation zwischen mehreren Knoten in einem Cluster von Drohnen*. Wenn Drohnen eine abstrakte Kommunikationstechnologie verwenden, muss ein Algo-

[1] Die École polytechnique fédérale de Lausanne ist eine technisch-naturwissenschaftliche Universität in Lausanne, Schweiz.

rithmus gewählt werden, der Daten von Drohnen in einem Schwarm zuverlässig synchronisiert. Um eine passende Lösung dafür zu finden, wurden bestimmte Algorithmen verglichen. Zwei Algorithmen, die sich bei der Betrachtung verteilter Systeme und der Synchronisation von Konten besonders auszeichnen, sind *Paxos* und *Raft*. Das Problem der Synchronisation zwischen den Knoten eines Systems wird in der Informatik auch *„Problematik der Replikation des Zustandsautomaten“* genannt. Langjährige Erfahrungen der Cloudprovider, die diese Algorithmen für mehrere Bereiche ihrer Infrastruktur einsetzen (vor allem Speicherdienste und *MapReduce* [2] Clusters), haben die Nutzbarkeit dieser Algorithmen bewiesen. Zwar werden beide Algorithmen in zahlreichen Produktivsystemen benötigt, die Meinungen über die jeweiligen Stärken und Schwächen der Algorithmen sind jedoch noch gespalten.

In einem engen Zusammenhang mit verteilten Systemen stehen *Concurrency Modelle*. Ergänzend zu den unabhängigen Fehlern der Teilkomponenten und dem Fehlen der Global Clock ist die Concurrency eine der wichtigsten Eigenschaften der verteilten Systeme. In dieser Bachelorarbeit werden die Eigenschaften mehrerer Concurrency Modelle verglichen, um ein dafür am besten geeignetes Modell für die Entwicklung verteilter Systeme zu finden.

Überblick über das Dokument: im 2. Kapitel 2 wird das Thema der Drohnen und der Interaktion von Drohnen in einem Schwarm betrachtet: was sind die Vor- und Nachteile bei der Verwendung eines Schwarms, welche Schwierigkeiten bringt die Steuerung eines Schwarms mit sich, welche theoretischen Frameworks gibt es für die Steuerung verteilter Agenten. Außerdem wird das Thema der verteilten Systeme behzandelt: theoretische Grundlagen werden erläutert und Algorithmen für verteilte Systeme verglichen. Anschließend werden Concurrency Modelle für die Implementierung verteilter und ausfallsicherer Systeme untersucht. Das letzte Kapitel der Arbeit 3 befasst sich mit der Umsetzung eines der besprochenen Algorithmen mit einem der ausgewählten Concurrency Modelle. In diesem Kapitel wird der ausgewählte Algorithmus im Detail analysiert und erklärt. Welche Entscheidungen wurden bei der Umsetzung getroffen und weshalb. Zudem zeigt dieses Kapitel, welche Vorteile die ausgewählten Werkzeuge bieten.

[2] MapReduce ist ein Programmiermodell für nebenläufige Berechnungen über große Datenmengen auf Computerclustern.

Kapitel 2

Analyse

Um die Problemstellung der Arbeit besser zu verstehen, muss man einige grundlegende Konzepte der *Schwarmrobotik* und verteilter Systeme verstehen. Außerdem gibt dieses Kapitel einen Überblick auf den aktuellen Stand der Technik, Optionen und Empfehlungen für technische Umsetzung.

2.1 Vorteile von Drohnenschwärmen

Heutzutage werden Drohnen entweder einzeln verwendet oder sie finden in einem Drohnenschwarm ihre Anwendung. Beide Verwendungsarten haben Vor- und Nachteile und umfassende Anwendungsfälle. Um den Weg von einer Drohne zu einem Drohnenschwarm zu verstehen, muss man zuerst die Forschungsrichtungen im Feld von UAVs (Unmanned Aerial Vehicles) betrachten. Zu UAVs gibt es vier Forschungsschwerpunkte, die an entsprechende Herausforderungen gebunden sind:

1. Ein einziges UAV muss stets in der Lage sein, bestimmte Gebiete und Flächen zu überfliegen.
2. Objekte und Ereignisse von Interesse müssen von dem UAV erkannt werden können.
3. Um einen Synergieeffekt zu erreichen, ist die Kommunikation zwischen mehreren Drohnen essenziell.
4. Damit das Arbeiten der Drohnen an einem gemeinsamen Ziel ermöglicht wird, gilt es die Koordination eines Netzwerks von UAVs zu ermitteln.

Diese Bachelorarbeit fokussiert sich auf Drohnenschwärme und auf die Koordination zwischen den einzelnen Drohnen. Betrachtet man eine Gruppe von Drohnen, stellt man fest, dass man einen Algorithmus benötigt, der die Funktionsweise dieser Gruppe optimiert, damit diese durch Zusammenarbeit ein gemeinsames Ziel erreicht. Unter anderem können folgende Probleme durch die Anwendung eine Schwarms gelöst werden:

Verteilung von Rechenoperationen (optimale Nutzung von Ressourcen). Das ist ein klassischer Anwendungsfall aus der Informatik. Keine einzelne Drohne hat

genug Rechenleistung, um eine schwierige Operation im Stand-Alone-Betrieb bzw. ohne Verbindung zur Cloud zu berechnen. Dadurch, dass es in einem Schwarm mehrere Drohnen gibt, kann die Arbeit aufgeteilt werden. So einen Ansatz kann man "Ad-Hoc Cloud" nennen. Unter der optimalen Nutzung von Ressourcen kann man auch den Austausch von lokalen Karten und Sensorwerten verstehen.

Verteilung von Aufgaben. Für eine bestimmte Art von Problemen, die Schwarmintelligenz zur Lösung benötigt, ist es wünschenswert, dass jede Drohne eine bestimmte Funktion übernimmt und durch ihre speziellen Fähigkeiten (künstliche Intelligenz) oder Ausrüstung (Sensoren und Aktuatoren) hilft, das gemeinsame Ziel zu erreichen. Aus der Sicht der Ausfallsicherheit, Robustheit und Zuverlässigkeit eines verteilten Systems ist es immer besser, Tasks auf mehrere Geräte zu verteilen. Dieser Zusammenhang wurde schon durch die Erfahrungen von Google mit verteilten Systemen bewiesen. Andererseits muss man mit dem Kommunikationsoverhead und der gestiegenen Komplexität bei der Entwicklung rechnen.

Ausfallsicherheit. Ein Drohnenschwarm ist robust gegen Teilausfälle und bietet mehr Flexibilität bei Missionen. Wie oben schon erwähnt, können Aufgabenbereiche auf einzelne Drohnen und Drohnengruppen aufgeteilt werden.

Kollektive Entscheidungsfindung. Jede Drohne im Schwarm kann bzw. darf nur einen Teil vom ganzen Problem kennen. Für die Entscheidungsfindung braucht ein automatisiertes System sehr häufig nicht nur das lokale Wissen, sondern auch die Vogelperspektive oder den "Meinungsaustausch" mit anderen Drohnen.

2.2 Autonomie im Drohnenschwarm

Die im vorherigen Subkapitel erwähnten Vorteile eines Schwarms werden von Softwarekomponenten einer Drohne definiert, da diese Softwarekomponenten das Kommunikationsmuster bestimmen. Ein wichtiger Aspekt dabei ist die Autonomie von Drohnen. Autonomie ermöglicht einzelne Drohnen eines Schwarms lokale Entscheidungen zu treffen.

2.2.1 Methoden zur Kommunikation im Drohnenschwarm

Um die Vorteile, die Autonomie einzelner Drohnen mit sich bringt, ausnutzen zu können, können folgende systematische Ansätze angewendet werden:

Multiagentensystem. Ein Multiagentensystem ist eine Klasse von Algorithmen, in der einzelne Agenten basierend auf vordefinierten Regeln und Einschränkungen miteinander interagieren. Dadurch wird ein kollektives Verhalten ermöglicht.

Die Interaktionen im System finden sowohl zwischen Agenten untereinander als auch zwischen Agenten und der Umgebung statt. Dabei spielt die sogenannte *„Reward Function"* eine wichtige Rolle. Diese beschreibt, wie sich ein Agent verhalten muss. Der *„normative Inhalt"* schreibt dem Agenten vor, wie er bestimmte Aufgaben lösen soll. Im Multiagentensystem kennt ein Agent den gesamten Problembereich bzw. -raum nicht und muss die Lösung deswegen durch Lernen herausfinden. Multiagentensysteme üben Selbstorganisation, komplexe Verhaltensweisen

und auch Kontrollparadigmen aus, obwohl individuelle Strategien von allen Agenten einfach sind.

Ein Agent im Multiagentensystem hat einige wichtige Eigenschaften: *Autonomie* (ein Agent ist teilweise unabhängig, selbstbewusst und autonom), *lokaler Überblick* (kein Agent hat einen globalen Überblick und das System ist für einen einzelnen Agent zu komplex) und *Dezentralisierung* (kein Agent wird als Leader bezeichnet / gewählt).

Verteilte Problemlösung. Kooperative verteilte Problemlösung ist ein Netzwerk von halbautonomen Verarbeitungsknoten, die zusammenarbeiten, um ein Problem zu lösen. Dabei geht es um die Untersuchung der Problemaufteilung, Unterproblemaufteilung, Synthese vom Ereignis, Optimierung des Problemlösers und Koordination. Es ist eng mit der verteilten Programmierung und der verteilten Optimierung verbunden.

In der verteilten Problemlösung arbeiten mehrere Agenten daran, ein spezifisches Problem zu lösen. Das wichtigste in diesen Systemen ist, dass Kooperation erforderlich ist, weil kein einziger Agent genug Informationen, Wissen und Fähigkeiten hat, um das Problem zu lösen. Die eigentliche Herausforderung besteht in der Sicherstellung, dass die Informationen so aufgeteilt werden, dass die Agenten einander ergänzen und nicht miteinander im Konflikt stehen. Ein Algorithmus für die verteilte Problemlösung muss ein größeres Problem in Teilaufgaben unter Berücksichtigung der räumlichen, zeitlichen oder funktionalen Aspekte gliedern.

Ein Algorithmus für verteilte Problemlösung muss unter folgenden Einschränkungen handeln:

1. Kein Knoten hat genug Informationen, um das Problem selbstständig zu lösen.
2. Im System gibt es weder eine globale Steuerung, noch einen Speicher. Steuerung und Speicher sind verteilt.
3. Die Berechnung von Operationen auf einer lokalen CPU ist grundsätzlich schneller und weniger aufwändig als die Aufteilung derselben Operationen auf verteilte Systeme.
4. Darüber hinaus muss das zu lösende Problem modular sein. Außerdem darf es keinen einzigartigen Knoten geben, da dies zu einem *Bottleneck* im System führen kann. Ein solcher *einzigartiger Knoten* wäre jener, der im Cluster oder im Schwarm eine spezielle Rolle erfüllt. Eine solche Aufgabe könnte beispielsweise die eines Leaders oder eines Koordinators sein.

Schwarmintelligenz / Schwarmrobotik. Schwarmintelligenz ist das kollektive Verhalten von dezentralisierten, selbstorganisierten, natürlichen oder künstlichen Systemen. Das Konzept wird sehr oft im Bereich der Künstlichen Intelligenz eingesetzt.

Systeme mit Schwarmintelligenz bestehen typischerweise aus einer Population von Agenten und Boids, die untereinander und mit der Umgebung kommunizieren. Inspiration für solche Systeme kommt aus der Natur, vor allem aus biologischen Systemen. Die Agenten folgen sehr einfache Regeln. Obwohl es keine zentralisierte Steuerung gibt, führen lokale und bis zu einem gewissen Grad zufällige Interak-

tionen zur Entstehung eines intelligenten und globalen Verhaltens. Ein klassisches Beispiel aus der Natur sind Ameisenkolonien.

An dieser Stelle sieht man, dass die Grenzen zwischen Multiagentensystem, verteilter Problemlösung und Schwarmintelligenz verschwimmen können. In der Tat ist das Bilden einer klaren Grenze sehr schwierig.

2.2.2 Einblick in zukünftige Systeme

Moderne Systeme sind meist mit der Cloud verbunden oder brauchen zumindest eine kurze Synchronisation mit einem Cloud-Server. Nehmen wir an, dass eine Drohne eine Mission erfüllen soll, was sowohl Kommunikation zwischen Drohnen als auch zwischen dem Schwarm und der Cloud erfordert. Alle eingebetteten Systeme sind bis zu einem gewissen Grad in ihren Ressourcen (Rechenleistung, Akku, Konnektivität, Speicher) eingeschränkt. Dennoch wäre es sinnvoll, *Missionsinformationen* zwischen der Cloud und einer einzelnen Drohne (dem sogenannten Leader) zu synchronisieren und mittels eines Algorithmus im Schwarm zu verteilen.

Dies widerspricht dem Regel, dass jeder einzigartige Knoten in einem verteilten System ein potenzielles Bottleneck ist, da die Präsenz eines speziellen Knotens die Durchsatzrate senken und die Ausfallsicherheit des Gesamtsystems verringern kann. Diese Herausforderungen müssen natürlich mit einem klugen Algorithmus ausgeglichen werden, womit sich diese Arbeit auch befasst.

In der Zukunft sollen Systeme wie Drohnenschwärme selbstorganisierend und autonom agieren. Laut den Forschungsergebnissen der Technischen Hochschule Lausanne (EPFL) sind allerdings mit aktuellem Stand der Technik in der Robotik 100% selbstorganisierende Systeme weniger effizient als die Systeme, in denen Kommunikation zwischen Knoten möglich ist. Das heißt, dass die Synchronisation von Knoten im Schwarm eine gültige und in der Branche durchaus akzeptierte Lösung ist.

2.3 Analyse von Algorithmen für verteilte, ausfallsichere Systeme

Wie bereits im vorherigen Kapitel festgestellt, ist für das Funktionieren eines Schwarms die *Replikation* von Daten zwischen den Knoten erforderlich. Diese Aufgabe umfasst das Themengebiet der verteilten Systeme. Die Auswahl eines Algorithmus beeinflusst die Robustheit des Gesamtsystems und wird in diesem Kapitel betrachtet.

Verteilte Systeme finden in mehreren Teilen der IT-Infrastruktur Einsatz. Häufige Einsatzbereiche sind:

1. Dateisysteme
2. Datenbanken
3. Key-Value Stores (Schlüssel-Werte-Datenbank oder eine verteilte Hashtabelle)
4. Queues (Verarbeitung von Datenströmen, keine einfache Datenstruktur in diesem Kontext)

Diese genannten Systeme haben zwar unterschiedliche Anwendungsgebiete, die potenziell auftretenden Probleme sind allerdings sehr ähnlich:

Fehlertoleranz. Um die Fehlertoleranz zu garantieren, muss ein System mehrere *Replica* aller Daten besitzen und beim Ausfall der *Hauptreplica* auf eine andere reibungslos und ohne Datenverlust umsteigen können. Die Anzahl von Replicas soll immer eine ungerade Zahl sein, typischerweise werden drei oder fünf Replicas gebildet. Eine ungerade Zahl verhindert eine gleiche Aufteilung von Stimmen zwischen 2 Gruppen von Replicas. Gibt es eine Mehrheit aller Knoten, die den gleichen Wert haben, werden diese Daten als korrekt gewertet.

Zeit für Wiederherstellung. Ein Algorithmus für das verteilte System muss die Zeit für die Wiederherstellung verkürzen und intelligent auf mögliche Ausfälle reagieren. Der Algorithmus muss das Cluster unter neuen Einschränkungen umstrukturieren.

Erreichbarkeit. Die Erreichbarkeit eines Systems stellt fest, ob ein Knoten als abgestürzt oder als laufend betrachtet werden soll. In verteilten Systemen gibt es allerdings keine Möglichkeit zu überprüfen, ob ein Knoten abgestürzt ist oder das Netzwerk dieses Knotens überfordert ist. In modernen Systemen wird das Problem mittels *Timer* gelöst. Nach einem bestimmten Timeout wird der Knoten als nicht mehr erreichbar markiert. Nach welcher Zeit ein Knoten als nicht erreichbar betrachtet wird, ist für jede Konfiguration des Clusters spezifisch.

Skalierbarkeit und Durchsatzrate. Die Skalierbarkeit und Durchsatzrate in verteilten Systemen wird durch "Scaling up" Verfahren erreicht, wenn zur Lösung eines Problems oder für eine bessere Fehlertoleranz mehrere Knoten zusammengebracht werden. Für ein klassisches "Scaling up" Verfahren braucht man dagegen einen neuen, leistungsfähigeren Rechner, der alle Anfragen abarbeiten kann. Eine weitere Lösungsmöglichkeit stellen Algorithmen dar. Ein Algorithmus, der Leseoperationen von allen Knoten eines Clusters erlaubt, ist viel effizienter als der Algorithmus, der Leseoperationen nur von einem einzigen Knoten im Cluster erlaubt. Dieser einzige Knoten ist leicht zu überfordern und wird durch eine Hohe Nutzung des Netzwerks öfter als andere Knoten ausfallen.

Synchronisation und Timestamping. Timestamping hilft bei der Auflösung von Konflikten, beispielsweise zwischen Einträgen auf unterschiedlichen Knoten einer verteilten Datenbank bzw. eines verteilten Dateisystems. Der Eintrag mit dem höheren Timestamp hat Vorrang und soll einen anderen Eintrag überschreiben. Das Problem mit den Timestamps ist die Desynchronisation von Uhren auf allen Knoten. Mit modernen Technologien ist eine vollständige Synchronisation zwischen allen Knoten nicht umsetzbar. Selbst für Google, das mehrere Datenzentren mit Atomuhren und GPS-Fehlerkorrektur betreibt, ist das derzeit unmöglich. Die *TrueTime API* liefert zwar für höchst genaue Timestamps einen Zeitabstand, diese sind allerdings nicht eindeutig.

Eine weitere Lösung bietet die sogenannte *„Logische Anordnung von Operationen"*, welche Algorithmen zur Verfügung stellt. Es werden logische Zusammenhänge zwischen Operationen festgestellt und dadurch eine Linearisierung jener Operationen erreicht. (Anordnung dieser Operationen, als ob sie nacheinander ausgeführt wären).

Bis auf die Erreichbarkeit können all diese Probleme durch die Kombination spezialisierter Algorithmen gelöst werden. Die Replikation von Daten zwischen mehreren Knoten eines Schwarms fällt in die Klasse der Konsens-Algorithmen. Diese Klasse von Algorithmen löst ein fundamentales Problem der verteilten Systeme und Multiagentensysteme: die allgemeine Zuverlässigkeit des Systems beim Vorhandensein einer größeren Anzahl fehlerhafter Knoten. Dies erfordert eine Koordinierung der Knoten, um einen Konsens zu erreichen bzw. sich über einen Wert zu einigen, der für eine Rechenoperation nötig ist. Mögliche Anwendungen von Konsens-Algorithmen umfassen:

1. Anordnung von Transaktionen einer Datenbank
2. Cloud Computing
3. Lastverteilung (wie Load Balancing für Internetdienste)
4. Blockchain
5. Steuerung von UAVs (und generell mehreren Robotern / Agenten)

Um die richtige Entscheidung bei der Auswahl eines Algorithmus für den Konsens zu treffen, muss man zuerst die dahinter liegenden theoretischen Grundlagen verstehen, die auch Auswahlkriterien für Konsens-Algorithmen darstellen.

2.3.1 Konsens in verteilten Systemen

Der Konsens über einen Wert erfordert eine Einigung zwischen einer Reihe von Knoten. Einige der Knoten können ausfallen oder auf andere Weise unzuverlässig funktionieren. Deswegen muss der Konsens-Algorithmus fehlertolerant sein. Die Knoten haben die Aufgabe, die geltenden Werte zu ermitteln, miteinander zu kommunizieren und sich auf einen einzigen Wert zu einigen.

Konsens ist ein grundlegendes Problem bei der Steuerung von Multiagentensystemen. Definition von Konsens besagt, dass alle Knoten sich auf einen Mehrheitswert einigen müssen. In diesem Zusammenhang ist es erforderlich, dass sich im verteilten System eine Mehrheit bildet. Die Mehrheit ist immer um einen Knoten größer als die Hälfte aller Knoten, die am Wahlprozess teilnehmen. Ein oder mehrere fehlerhafte Knoten können das resultierende Ergebnis jedoch so verzerren, dass möglicherweise kein Konsens oder ein falsches Ergebnis erzielt wird.

Algorithmen, die den Konsens lösen, wurden entwickelt, um mit einer begrenzten Anzahl fehlerhafter Knoten umzugehen. Ein Knoten wird als fehlerhaft bezeichnet, wenn in diesem Knoten ein Fehler auftritt, durch den der Knoten nicht mehr erreichbar wird. Diese Algorithmen müssen eine Reihe von Anforderungen erfüllen, um nützlich zu sein. Ein Konsens-Algorithmus, der fehlerhafte Knoten toleriert, muss folgende Eigenschaften erfüllen:

Termination. Jeder korrekte Knoten entscheidet sich für einen Wert.

Integrity. Wenn alle korrekten Knoten denselben Wert vorschlagen, muss sich jeder korrekte Knoten für diesen Wert entscheiden.

Agreement. Jeder korrekte Knoten muss sich auf den gleichen Wert einigen.

All diese Kriterien werden durch die grundlegende Eigenschaften verteilter Systeme geprägt. Von diesen Eigenschaften zeichnet man vor allem aus:

Atomarität. Die wichtigste Eigenschaft aller verteilten Systeme besteht darin, dass ein System auf einen externen Beobachter wie ein einziger Rechner wirken soll. Das System verbirgt Implementierungsdetails und dient als eine Abstraktionsschicht.

Widerstandsfähigkeit. Wie es schon erwähnt wurde, braucht ein System mehrere Replicas, um einem Ausfall zu widerstehen. Die Widerstandsfähigkeit, also die Anzahl der Fehler, die ein System überleben kann, wird mit folgender Formel ausgerechnet:

$$T = (n - 1)/2$$

bei der

T : Anzahl der überstandenen Fehler
n : Anzahl der Knoten im Cluster

Kapitel 3

Entwicklung

Kapitel 4

Fazit

Anhang A

Technische Informationen

Anhang B

Ergänzende Inhalte

Auflistung der ergänzenden Materialien zu dieser Arbeit, die zur digitalen Archivierung an der Hochschule eingereicht wurden (als ZIP-Datei).

B.1 PDF-Dateien

Pfad: /

thesis.pdf . . . . . Finale Master-/Bachelorarbeit (Gesamtdokument)

B.2 Mediendaten

Pfad: /media

*.ai, *.pdf . . . . . Adobe Illustrator-Dateien
*.jpg, *.png . . . . . Rasterbilder
*.mp3 . . . . . Audio-Dateien
*.mp4 . . . . . Video-Dateien

B.3 Online-Quellen (PDF-Kopien)

Pfad: /online-sources

Reliquienschrein-Wikipedia.pdf

Anhang C

Fragebogen

Anhang D

LaTeX-Quellcode

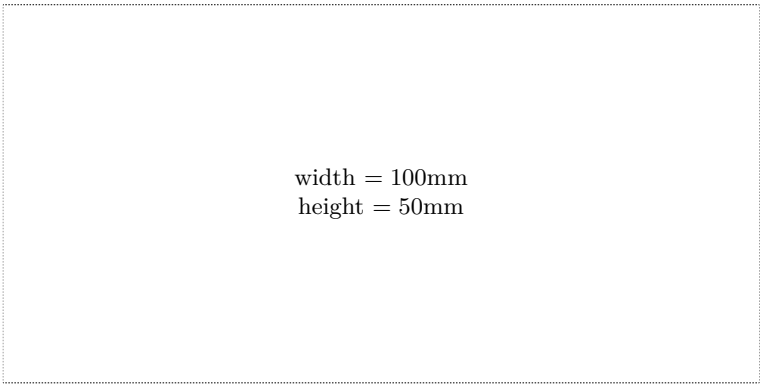
Quellenverzeichnis

Online-Quellen

[1] *Reliquienschrein*. Sep. 2018. URL: https://de.wikipedia.org/wiki/Reliquienschrein (besucht am 28. 02. 2019).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —