VSDB

Space Based Computing

Ausarbeitung

Daniel Dimitrijevic Thomas Traxler

13. Dezember 2013

5AHITT

Inhaltsverzeichnis

1 Erklärung

1.1 Wofür steht Space-Based Computing?

Space-Based Computing (fortan SBC) hat seine Ursprünge im parallel programming, und stellt dabei hauptsächlich ein Datenorientiertes Modell zur Koordination dar. Wie aus dem Namen eindeutig hervorgeht handelt es sich hierbei um ein Modell welches auf 'Spaces' basiert. Ein Space ist hierbei gleichbedeutend mit einem (logischen) Ort auf welchem Daten von mehreren Komponenten geteilt verwendet werden. Diese Komponenten können im einfachsten Fall 'write', 'read' und 'take' Aktionen ausführen. Ein Write steht hierfür für das zur Verfügung Stellen eines neuen Datenteil an alle anderen Komponenten in diesem Space, ein Read für das Lesen von Daten ohne diese zu entfernen und ein Take für das Lesen und entfernen einer Datei aus dem Space, auch destructive read genannt. Im einfachsten Fall hat ein Space nun das Datenmodell eines Tuples.
Eine Hauptanforderung die an einen Space hier nun gestellt wird ist das persistente Aufbewahren aller Daten die sich in ihm befinden, auch bei Systemausfällen. Wie das SBC modelliert und implementiert wird kann sehr stark variieren, je denn gegebenen Anforderungen und Wünschen entsprechend. Was hierbei vor allem variiert ist die Zahl der Geräte auf dem der Space implementiert ist und die Zahl der Geräte die auf diesen Space zugreifen, diese Zahlen sind prinzipiell, wenn nicht durch den Anwendungsfall anders umgesetzt, voneinander unabhängig und befinden sich jeweils im Bereich von 1 bis n.
Das SBC-System stellt hierbei eine logische zentrale Einheit dar welche nicht spezifiziert wo sie genau überall Physikalisch vorhanden ist. Der Ort wo eine Datei schlussendlich wirklich physikalisch abgespeichert wird kann dabei nach verschiedensten Methoden ausgewählt werden, in der simpelsten Form wird das selbe Prinzip wie bei Tuples angewendet oder es werden andere Prinzipien verwendet wie FIFO, LIFO, keys, geo-coordinates oder noch kompliziertere, auch das ist vom gegebenen Anwendungsfall abhängig.

1.1.1 Unterschied zwischen SBC und Cloudcomputing

Um nun der Verwechslung von SBC und Cloudcomputing vor zu Beugen sie hier nun gesagt, zu aller erst behandelt SBC lediglich die Daten und Cloudcomputing stellt noch viele weitere Ressourcen (wie zB. Rechenleistung) zur Verfügung. Auch zu Cloud-Storrage kann man abgrenzen, Cloud-Storrage behandelt nicht nur die interne Abspeicherung der Daten sondern auch vor allem wie diese dem Benutzer zur Verfügung gestellt und abstrahiert werden, bei SBC geht es hauptsächlich um die interene Verteilung und Kommunikation der Komponenten untereinander und stellt dabei einen Datenkanal zwischen

den Komponenten des Spaces dar beziehungsweise zur Verfügung.

1.2 Einsatzbereiche

2 Grundlegende Prinzipien

2.1 Space-Based Computing Paradigmen

2.1.1 Tuple Spaces

2.2 Mapping

2.3 EAI

3 Im genaueren betrachtet

3.1 Tuple Spaces

Tuple Spaces(Linda)

Ein Tupel Raum ist eine Ausführung des assoziativen Gedächtnisses Modell für verteilte / paralleles computing. Es bietet eine Bibliothek von Tupeln, auf die gleichzeitig zugegriffen werden kann. Tupel sind Begriffe, mit null oder mehr Argumente und einem Schlüssel.

Die Sammlung von Tupeln unterstützt einige Grundfunktionen, wie zB das Hinzufügen eines Tupels in den Raum (Schreiben) und Entfernen eines Tupels aus dem Raum (nehmen). Das Tupel Sammlung wird von einem Netzwerk über meherer Server aufbewahrt und gemanaged. Mehrere Threads können zur selben Zeit auf den selben Raum zugreifen. Nun gehen wir einwenig auf Linda ein.

Ziel von Linda ist es, Prozessen einer Anwendung zu erlauben, miteinander zu kommunizieren, ohne identifikations Informationen des Datensatzes zu wissen. Früher ging Linda von einem Tupelraum als einer abstrakten Umgebung aus. Verschiedene nebenläufige Prozesse eines verteilten Programms kommunizieren über einen gemeinsamen Tupelraum dadurch, dass jeder dieser Prozesse diesem Tupelraum beliebig Tupel hinzufügen und Tupel daraus entfernen kann.

Ein Prozess A lässt einem Prozess B Information zukommen, indem er eine gebündelte Menge von Werten, ein Tupel, im Tupelraum ablegt. Prozess B kann anschließend das Tupel lesen oder es aus dem Tupelraum entfernen, womit der Kommunikationsakt abgeschlossen ist. Der Prozess A benötigt weder einen Namen, eine Adresse oder sonstige identifizierende Information von Prozess B; für Prozess A ist es völlig irrelevant, ob Prozess B oder irgendein beliebiger anderer Prozess, mehrere Prozesse oder kein Prozess das Tupel liest. Derjenige Prozess, der das Tupel des Prozesses A entnimmt, muss nicht einmal zur gleichen Zeit wie Prozess A aktiv sein oder existieren. Das von A generierte Tupel ist von seinem Erzeugerprozess vollkommen unabhängig, das bedeutet, dass Prozess A zum Zeitpunkt der Entnahme seines Tupels durch Prozess B schon lange beendet sein kann.

Die hieraus resultierende zeitliche und aufgrund der Verteilung auch räumliche Entkoppelung ermöglicht einen Entwurf verteilter Protokolle, die flexibel und robust auf die Herausforderungen verteilter Programmierung durch Latenz, erhöhten Synchronisationsaufwand und mögliche Teilausfälle des Systems reagieren können.

3.2 Triple Spaces

Wie Tuple Spaces will auch Triple Spaces auf parallelen Verarbeitung von Anfragen auf Daten. Es übernimmt das Virtuel shared Memory

Zusammensetztung aus: Tuple Spaces Semantic Web Web Service

Semantic Web Technologie bietet Vorteile bei der Verwaltung und Anpassung von Kommunikation durch semantische Annotationen. Durch die Erstellung von Ontologien bieten sie Konsens Terminologien und erleichtern interoperability. „Matchmaking“ von semantischtisch verknüpften Daten kann die Notwendigkeit von direkten Zuordnungen von Kommunikationsbedarf reduzieren.

Web-Service Technologie bietet eine "Virtuelle Komponenten-Modellßur vereinheinfachung der heterogenenen Welt von Komponenten. Es erlaubt es, bestehende Funktionalität zu nutzen ohne die Last der Mittelware spezifischen Eigenheiten, wie die invokation-Mechanismus, Transport-Protokoll usw. Web Services sind eine gut verstandenes comkommunikationsstruktur und-Architektur für Unternehmensanwendungen. Letztlich Web Dienstleistungen sind ein weiterer großer Schritt in Richtung Lösung des EAI Problem.

3.3 Reliable Message

Reliable Message verhält sich wie Tuple Space computing. Nur das man hier nicht von Tuples spricht sondern von Messages. Die Reliable Message Technologie geht von mehreren Channels aus die die geteilten Daten beinhalten und jeder dieser Channel kann von einem einem oder mehreren Server „unterstützt“ werde d.H. das die geteilten Daten auf einem oder mehreren Servern liegen und damit abgesichert werden können.

4 Namhafte Implementierungen

4.1 JavaSpaces

JavaSpaces ist eine Spezifikation des Konzepts Object Spaces in der Programmiersprache Java. Ein Object Space ist hierbei ein assoziativer Speicher von verteilten, über das Netz erreichbaren Objekten. Kommunikationspartner (peers) kommunizieren ausschließlich indirekt über diese Objekte (stateful communication and coordination). Dadurch etabliert der JavaSpace einen äktiven, verteilten Datenraum", wie er in keiner anderen Technologie geschaffen wird (traditionelles Grid-Computing). Einige Ansätze der Jini-Technologie kommen hierbei zur Anwendung. Bei der Idee, die sich hinter den JavaSpaces verbirgt, handelt es sich nicht um eine revolutionäre Neuerung, sondern sie basiert im Wesentlichen auf den Linda TupelSpaces.

Die Gründe, warum JavaSpaces eingesetzt werden, sind vielfältig. Meist wird Skalierbarkeit und Verfügbarkeit bei gleichzeitiger Reduzierung der Gesamtkomplexität angestrebt.

4.2 Corso

Corso(Co-ORdinated Shared Objects) basiert auf Forschungsarbeiten der Technischen Uni Wien. Software Entwicklung in heterogenen verteilten Systemen bringt immer zusätzlich Komplexität. • Lokatoin: Addressierung und Lokalisierung von Ressourcen • Replikation: Verwalten von Daten Kopien an dislozierten Orten • Transaktion: Synchronisationsmechanismen konkurrierender Zugriffe auf verteilte Ressourcen • Skalierbarjeit: Die Möglichkeit zur transparenten Erweiterung des Systems im Zuge von wachsenden Anforderungen und Belastung an das Gesamtsystem • Lastverteilung: Die faire und gleichmäßige Verteilung von Aufgaben an verteilte Komponenten • Fehlersicherheit: Die kompensation aufallender Rechner und Persistzenz von flüchtigen Daten Corso ist eine Middleware. Corso erleichtert die Addressierung und vereinfacht das arbeiten mit Spaces.

4.2.1 Virtual shared Memory

Ein Virtuals shared Memory stellt einen konzeptionell hohne Abstraktionslevel für den Datenaustasch in verteilten Systemen dar. Es stellt einen Raum dar auf dem parallel verteilte Prozesse eine konsistente Sicht haben. Die verteilten Objekte werden für Speicherung von Informationen, Kommunikation, und Synchronisation von Prozessen verwendet.

4.3 XVSM

XVSM (eXtensible Virtual Shared Memory). XVSM ist eine Middleware technology die Daten in "Spaces" speichert und für andere Peers teilt. Dieser Weg bringt einige Vorteile wie die Daten werden auf mehreren verschiedenen Computern verteilt, damit wird die ausfallswahrscheinlichkeit der Daten reduziert.

XVSM ist keine middleware sondern eine technology die implementiert und verwendet werden kann. Unterschied zum Linda(javaSpaces) ansatz ist, dass die Daten mit eintregen, coordinaten und containern leichter gefunden werden können.

4.4 TinySpaces

Tinyspaces setzt auf das XVSM prinzip auf und benutzt das .Net Framework. Da es bei dem XVSM Prinzip noch nicht vollkommen mit allen Implementierungen kompatibel war erschuf man Tinyspaces. TinySpaces soll es erleichtern mit space based computing zu arbeiten. Da es unnötige schichten entfern. TinySpaces nutzt die unterstützung von Contracts die sagen wie man ein bestimmtes Interface ansprechen soll. Durch die Nutzung von Contracts ist es möglich die angesprochenen Componenten zu ändern. Da man bei der programmierung der Aplication darauf zielen sollte das Alle Componenten auf denn Contract aufbauen sollten. TinySpaces setzt auf CAPI-1 auf damit sie denn wechsel von Componenten während der Laufzeit zu ändern.

4.5 GSpaces

Geht von dem selben Ansatz aus wie JavaSpaces(Tuple Spaces) hat aber einen großen Unterschied. GSpaces hat zwar Tupel aber benutzt diese auch mit Replikationsrichtlinien. Bei einem Aufruf wird aber ein lokaler Aufrufshändler aufgerufen und dann wird nachgeschaut welche Operation ausfgeführt werden muss. Bei einem aufruf wird erst mal nachgeschaut welche Richtlinien angewendet werden müssen. Nachdem man die Auswahl getroffen hat wird die Anfrage weiter an einen Verteilungsmanager weitergeleitet. Wenn dann zB gelesen werden soll und es auf Master-Slave-Richtlinie herrscht wird dann einfach die Read Operation auf dem lokalem Datensatz(Slice) gelesen. Falls es aber eine write Operation ist muss der Verteilungsmanager beim Master Server anfragen ob er schreiben darf und dann kann er erst weiter schreiben.

5 Conclusio

6 Quellen