Real-Time Event Processing with Microsoft Azure Stream Analytics

Björn Beha und Suhay Sevinc

Zusammenfassung—Der Artikel befasst sich mit Microsofts Cloud-Plattform Azure und dem darauf angebotenen Dienst zur Echtzeit-Eventverarbeitung - Azure Stream Analytics. Die Arbeit konzentriert sich auf die Rolle des Service innerhalb der Plattform und dessen Leistungsumfang. Für eine kritische Auseinandersetzung mit dieser Technologie werden die offiziellen Angaben von Microsoft herangezogen, ausgewertet und mit der kosten- und lizenzfreien Lösung Apache Storm verglichen. Die Abhandlung beinhaltet eine Auswertung der Unterschiede dieser Untersuchung, welche unter anderem die Skalierbarkeit und Verfügbarkeit der Systeme betrachtet. Unter Berücksichtigung dieser Ergebnisse erfolgt eine Bewertung. Die Gegenüberstellung zeigt, dass sich die beiden Technologien nicht direkt vergleichen lassen. Azure Stream Analytics stellt ein fertigen Dienst dar (Software-as-a-Service), während ein Apache Storm-System über eine eigene Implementierung realisiert wird. Azure Stream Analytics zielt auf Domänenexperten ab, die kaum Programmierexpertise besitzen bzw. diese gar nicht benötigen, wohingegen Apache Storm fundierte Kenntnisse erfordert.

Index Terms—Microsoft, Azure, Stream Analytics, Real-Time, Event-Processing, Apache Storm

1 Einführung

„In the Internet of Things [...] the real value will remain in the services behind the scenes processing terabytes of data to help people live better lives.“ [1]

Wie aus dem Zitat von Dr. Floarea Serban hervorgeht, handelt es sich beim Internet der Dinge vor allem um die Dienste zur Verarbeitung der massiven Datenströme. Die zunehmende Anzahl von leistungsfähigen und intelligenten Geräten wie beispielsweise Sensoren, Smartphones etc., fördern laufend die Entwicklung von Anwendungen zur Datenstromanalyse, um Ereignisse in Echtzeit zu überwachen. Bereits heute sind über 20 Milliarden Geräte mit dem Internet verbunden und laut Prognosen sollen diese Zahlen bis in wenigen Jahren drastisch anwachsen [2]. Das damit verbundene Wachstum der zu verarbeitenden (nicht-endlichen) Datenströme, stellt eine Herausforderung dar. Nicht zuletzt, weil diese Daten in Echtzeit verarbeitet werden sollen und sie sich ständig in Bewegung befinden [3].

Um relevante Informationen aus einem Strom von Sensordaten auszuwerten, brauchen ereignisgesteuerte Systeme eine Automatisierung dieser Datenverarbeitung. Genau hier setzt Real-Time Stream Analytics an. Microsofts Cloud-basierte Lösung leitet aus einem unendlichen Datenfluss Muster und komplexe Ereignisse ab, wodurch es einem solchen System ermöglicht wird, auf gewisse Gegebenheiten entsprechende Maßnahmen anzuwenden. So ist beispielsweise das Abfangen kritischer Situationen in der Produktion möglich und es können potenzielle Ausfälle von Maschinen rechtzeitig erkannt und vermieden werden [4]. Jedoch offeriert nicht nur Microsoft eine Lösung im Bereich Complex Event Processing. Auch Apache Storm bietet hier über das Open Source-Konzept einen Ansatz.

Vorab wird kurz erläutert, wobei es sich um Complex Event Processing handelt. Dies ist für das weitere Verständnis der Arbeit wichtig. Anschließend erfolgt eine Beschreibung von Microsofts Cloud-Plattform. Auf diese baut das Kapitel auf, in welchem der Azure-Dienst Stream Analytics genauer beschrieben und auf dessen Features näher eingegangen wird. Daraufhin erfolgt der Vergleich verschiedener Aspekte der beiden Echtzeitverarbeitungssystemen sowie eine abschließende Bewertung. Diese Arbeit zielt auf Leser ab, welche sich mit dem Azure-Dienst auseinandersetzen möchten. Es werden hier keine tiefgreifenden Informationen geliefert.

Einen Überblick zum Thema liefert das Buch IoT Solutions in Microsoft's Azure IoT Suite [5]. Allerdings zeigt sich, dass es im Kontext von Azure Stream Analytics keine wissenschaftlichen Arbeiten von dritten gibt, die den Funktionsumfang dieser Technologie bezeugen und kritisch beanstanden. Weiterhin gilt es zu betonen, dass es sich bei Azure Stream Analytics um ein kommerzielles Produkt handelt, welches keine direkte Analyse über den Quellcode zulässt. Diese Arbeit stützt sich auf offizielle Literatur und Angaben von Microsoft [6] [5]. Folglich können keine exemplarisch wichtigen Arbeiten vorgestellt werden, die mit der Forschungsfrage des Artikels in Beziehung stehen.

2 Stream Analytics und Complex Event Processing

Stream Analytics beschreibt grundlegend das Analysieren und Verarbeiten von Datenströmen, allerdings zur Laufzeit und nahezu in Echtzeit. Das bedeutet, dass die Zeit zwischen der Datenaufnahme und dem Ergebnis möglichst gering ist (typischerweise in Sekunden gemessen). Grundsätzlich geht es bei der Verarbeitung um das Filtern, Aggregieren und Suchen von Mustern. Die Resultate lassen sich anschließend an entsprechende Anwendungen weiterleiten. Stream Analytics gehört somit

zum Themenbereich Complex Event Processing [7].

Complex Event Processing (CEP) ist ein Sammelbegriff. Dieser umfasst Methoden, Techniken und verschiedene Werkzeuge, mit denen voneinander unabhängige Ereignisse stetig und zeitnah verarbeitet werden. Um auf Ereignisse zu reagieren, werden aus dem eingehenden Datenstrom benötigte Informationen extrahiert. Typische Reaktionen wären das Senden von Benachrichtigungen, einfache Aktionen oder Interaktionen mit Geschäftsprozessen [7].

Datenströme bestehen häufig zu einem Großteil aus irrelevanten Daten („noise“). Nur ein Bruchteil bilden dabei Nutzdaten („signal“). Ein Ereignisverarbeitungssystem filtert diese über geeignete Mechanismen heraus. Das ermöglicht eine Konzentration der relevanten Anteile, während beispielsweise Messfehler direkt aussortiert werden. Nachfolgend sind die Nutzdaten mit anderen Datenströmen in Bezug zu setzen. Durch diese Korrelation der einzelnen Sensordaten ergeben sich Ereignisse. Zusätzlich können die Daten mit Informationen (Meta-Daten) aus unterschiedlichen Datenbanken angereichert werden. Dies kann nützlich sein, um noch aussagekräftigere Ergebnisse zu erzielen. Dabei ist vor allem die zeitliche Komponente ein wichtiger Faktor. Für verschiedene Situationen ist es notwendig, dass Ereignisse in einem gewissen Zeitraum bzw. einer bestimmten Reihenfolge auftreten. Da Ereignisse, die in einem gewissen zeitlichen Abschnitt eingepflegt werden, von großer Bedeutung sein können, gibt es die Funktion der Zeitfenster. Abfragen auf einen unendlichen Ereignisstrom sind nur in bestimmten Ausschnitten denkbar [7]. Nach dieser Vorverarbeitung des Ereignisstroms, lassen sich die Daten gezielt nach definierten Mustern absuchen. Werden entsprechende Muster gefunden, kann das System darauf reagieren [8].

3 Microsoft Azure

Azure ist Microsofts Hybrid-Cloud-Plattform. Azure enthält eine stetig wachsende Anzahl an Tools, Diensten, Anwendungen und SQL- sowie NSQL-Datenbanken, die dem Nutzer über das Portal angeboten werden. Zudem bieten sie Dienste, mit welchen die Entwicklung intelligenter Anwendungen sowie die Forschung im Bereich Machine Learning ermöglicht wird. Nutzer können über Azure auf Hochleistungsgrafikprozessoren zugreifen, wodurch ihnen die Entwicklung im Bereich autonomes Fahren, Spracherkennung sowie Daten- und Videoanalyse erlaubt wird. Neben umfangreicher Prozessorleistung stellt Microsoft auch Speicherplatz zur Verfügung. Somit erlaubt Azure das Speichern und verknüpfen eigener Daten und Apps in der Cloud sowie lokal (daher Hybrid-Cloud) [9]. Über das Cloud-Computing-Konzept lassen sich IT-Infrastrukturen flexibel gestalten. Stichwörter hier sind Infrastructure-, Platform- und Application-as-a-Service. Auch Lösungen unter Einbeziehung von beliebige Entwicklungstools oder -sprachen werden dem Anwender ermöglicht [5].

Grundlage der Azure-Plattform ist ein wachsendes Netzwerk aus verteilten Rechenzentren. Somit kann eine hohe Verfügbarkeit gewährleistet werden. Die Sicherung der eigenen Anwendungen und Daten wird durch diese Hochverfügbarkeit und eine Notfallwiederherstellung gewährleistet. Durch kontinuierliche Investitionen in die neueste Infrastrukturtechnologie, bleibt Microsofts Plattform nachhaltig [5].

Derzeit liegt Microsoft hinter dem Marktführer Amazon, welches in Abbildung 1 illustriert wird. Auf der offiziellen Website von Azure wird die Frage gestellt ”Azure oder AWS?”. Microsoft gibt darauf eine prägnante Antwort: ”Azure ist die richtige Wahl” [10].

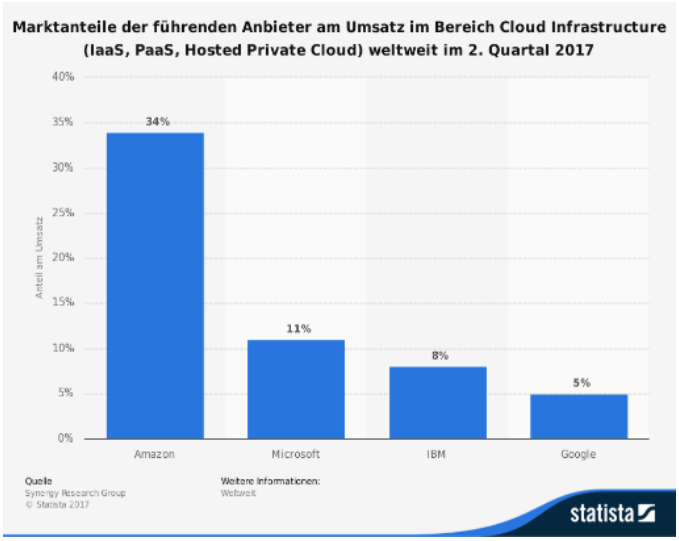

Abbildung 1. Marktanteil nach Umsatz [11]

Beide Plattformen haben ihre Vor- und Nachteile. Welche sich tatsächlich besser eignet, hängt von der Anforderung ab [12]. Was Azure allerdings zugute kommt, ist die ergonomische Benutzeroberfläche zum Aufsetzen komplexer Systeme. Dies ist durch das Verknüpfen verschiedener Dienste möglich. Die Schritt-für-Schritt-Anleitungen, die dem Anwender zur Verfügung gestellt werden, zeigen, dass die Einrichtung entsprechender Dienste, Server und Cloud-Instanzen mit wenig Aufwand durchzuführen ist. Somit macht es den Übergang zur Cloud-basierten Infrastruktur für viele Unternehmen unbeschwerter. Vor allem das nahtlose Zusammenarbeiten der angebotenen Dienste bzw. Azure-Instanzen erlaubt es, eine robuste Infrastruktur aufzubauen. Als Exempel kann hier der IoT Hub und die sich daran anschließende Ereignisverarbeitung Azure Stream Analytics dargelegt werden [12]. Eine Azure-Instanz besteht dabei aus einem virtuellen Server. Dieser wird über Hyper-V (Microsofts Hypervisor zur Visualisierung ihrer Server [13]) betrieben.

Microsoft bietet einsatzbereite Lösungen, die vollständig vorkonfiguriert sind. So lassen sich die häufigsten Szenarien im IoT-Bereich aufsetzen, ohne dass es Cloud-Wissen oder Software-Know-How bedarf. Abbildung 2 zeigt die fertige Lösung zur Überwachung der Telemetrie einiger Geräte.

Eine weitere verfügbare Lösung wäre der vorhersagbare

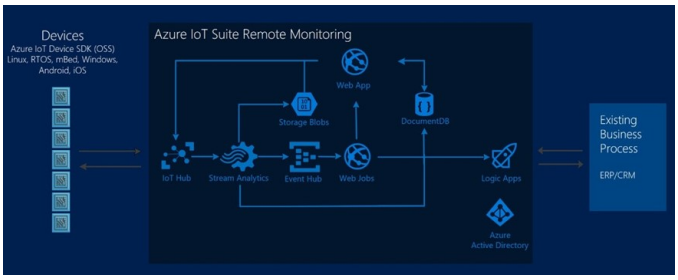

Abbildung 2. Fertige Remote Monitoring-Lösung [14]

Wartungsbedarf. Diese Systeme sind Teil der IoT bzw. Cortana Suite. Beide Angebote bestehen aus einer Reihe von Azure-Diensten [12].

4 Azure Stream Analytics

Vor allem im heutigen Industrie 4.0 Umfeld sammeln sich schnell umfangreiche Datenmengen. Diese können oft nur über einen (stark) begrenzten Zeitraum zurückgehalten werden, da sie ansonsten ihre Aussagekraft verlieren. Ab diesem Punkt muss eine Datenverarbeitung erfolgen. Hier bietet Azure Stream Analytics eine Lösung. Dieser Dienst erlaubt Einblicke in Daten, um darin beispielsweise nach Mustern oder Anomalien zu suchen und entsprechend darauf zu reagieren [5]. Azure Stream Analytics unterstützt dabei eine SQL-Ähnliche Abfragesprache, die Stream Analytics Query Language (SAQL), über welche die dynamischen Datenströme analysiert werden. Obwohl SAQL von der SQL-Abfragesprache abgeleitet wurde, enthält es spezielle Funktionen, die für die Stream-Verarbeitung notwendig sind [3]. Eine dieser Funktionen ist die Möglichkeit, Ergebnisse mithilfe von Zeitfenstern zu erhalten (siehe Fensterfunktionen). Ein typisches Echtzeitverarbeitungssystem, welches auf Stream Analytics und weiteren Azure-Diensten aufbaut, ist in Abbildung 3 aufgeführt.

Die linke Spalte zeigt Sensoren, Geräte und andere Datenquellen einer IoT-Lösung. Diese senden permanent Daten über das Cloud Gateway an die Azure Hubs. Der Gateway-Dienst ist hierbei ein virtuelles Gerät und ermöglicht eine konsistente Verbindung. Die Hubs sind in der Lage Millionen von Ereignissen pro Sekunde zu verarbeiten. Von dort aus werden Ereignisse an weitere Anwendungen übergeben. Stream Analytics transformiert eingehenden Daten. Die Ausgaben lassen sich daraufhin auf eine Vielzahl von Endpunkten richten. Azure bietet einige Dienste, welche daran angeknüpft werden können [3].
Sowohl Ein- als auch Ausgaben, die ein Stream Analytics-Job benötigt, werden im Azure-Portal konfiguriert. Ein einzelner Stream Analytics-Job kann mehrere Eingaben enthalten. Durch Aggregation der Datenströme lassen sich kombinierte Abfragen durchführen. Dies kann mit einem JOIN in SQL verglichen werden. Dadurch, dass ein einzelner Stream Analytics-Job mit mehreren Ein- und Ausgängen konfiguriert werden kann, ermöglicht es den Aufbau umfangreicher Topologien [3].

4.1 Fensterfunktionen

Das Hauptfeature der Stream Analytics-Abfragesprache SAQL ist die Unterstützung von Fensterfunktionen. Nachfolgend werden die Funktionen aufgeführt [3]:

- TumblingWindow
- HoppingWindow
- SlidingWindow

Zeitfenster erlauben es für Datenströme temporale Vorgänge durchzuführen. Die Ausgabe eines Fensters ist ein einzelnes Ergebnis, welches auf der verwendeten Aggregatfunktion basiert (siehe Abbildung 4).

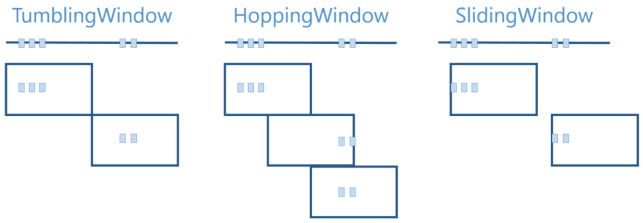

Abbildung 4. Fensterkonzepte [3]

Das Resultat enthält einen Zeitstempel vom Ende des Fensters. Ein Fenster wird stets über eine feste Länge definiert. Alle Ergebnisse sollten mit einer GROUP BY-Klausel entsprechend zusammengeführt werden [14]. Als Exempel dient eine Aggregatfunktion, die aufzeigt, wie viele Autos einer speziellen Farbe alle fünf Minuten eine Mautstelle passieren [3]. Mit TumblingWindow wird ein Datenstrom in einzelne Zeitsegmente unterteilt. Für diese unterteilten Segmente wird eine Funktion ausgeführt. Die Abfrage kann hier äquivalent zum oben beschriebenen Beispiel ausgedrückt werden. Bei den rollierenden Fenstern gibt es keine Überlappungen und ein Ergebnis kann nicht zu mehreren Fenstern gehören. Bei HoppingWindow wird immer für einen festen Zeitraum einen Sprung nach vorne durchgeführt. Grundsätzlich kann dieser Ansatz wie das TumblingWindow angesehen werden, bloß überlappen sich die wiederholenden Fenster. Durch diesen Umstand kann es sein, dass ein Ergebnis zu mehreren Fenstern gehören kann. Die Abfragen lassen sich ähnlich ausdrücken, allerdings kann hier unabhängig von der Fenstergröße eine Ausgabe erzeugt werden. Es lässt sich beispielsweise einmal pro Minute ausgegeben, wie viele Autos einer speziellen Farbe alle fünf Minuten diese Mautstelle anfahren. SlidingWindow hingegen erzeugt nur ein Ereignis, wenn entsprechende Anforderungen erfüllt sind [14]. Auch hier können Ergebnisse zu mehreren Fenstern gehören. Dieser Ansatz lässt die Abfrage zu, in welchen Fünf-Minuten-Zeitfenstern zehn oder mehr Autos einer speziellen Farbe die Mautstelle erreichen. SlidingWindow ist somit für den Umgang mit relativ spärlichen Ergebnismengen geeignet [3].

5 Lambda-Architektur

„Mithilfe des Lambda-Architektur-Konzepts haben wir eine Architektur geschaffen, für die immer stärker wachsende Datenberge und neue Anforderungen der Zukunft kein Problem darstellen.“ [15, S.2]

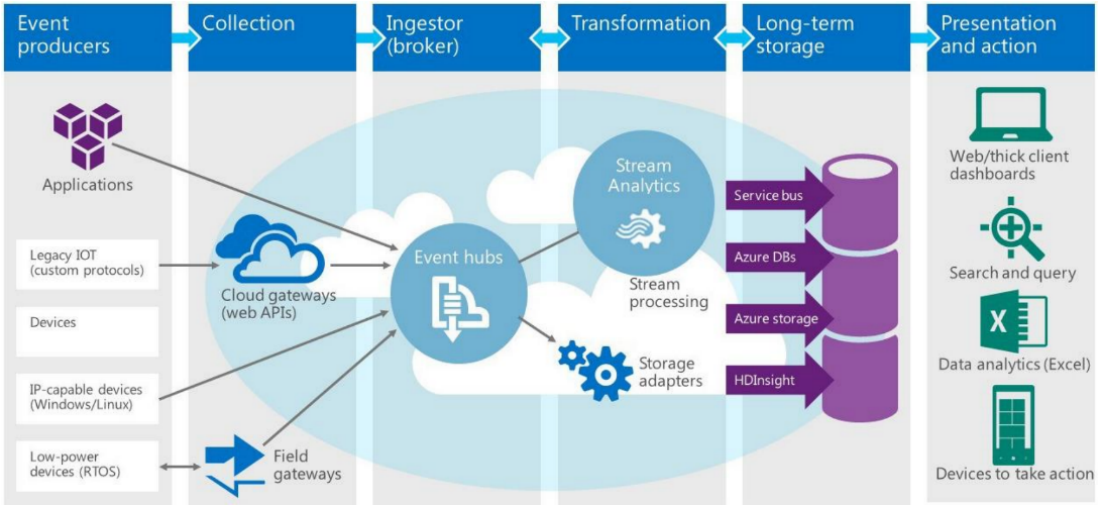

Abbildung 3. Typisches Echtzeitverarbeitungssystem [3]

Wie aus dem Zitat hervorgeht, ist die Lambda-Architektur ein Ansatz, um die entstehenden Herausforderungen bei der Echtzeitverarbeitung großer Datenmengen zu bewältigen. Diese Big-Data-Architektur weist eine geringe Latenz zwischen Ein- und Ausgabe auf und eignet sich für das Arbeiten mit hohen Rechenkomplexitäten. Aufgrund dessen findet sie sich in Stream Analytics-Frameworks wieder [6]. Die Lambda-Architektur ist ein populäres Big-Data-Pattern und beschreibt den konzeptionellen Aufbau einer Big-Data-Architektur, bestehend aus einem zentralen Eintrittspunkt. Die Architektonik besteht aus zwei Kernschichten: Batch- und Speed-Layer. Von einem Eintrittspunkt werden Daten an diese Ebenen gesendet.

Batch Layer. Diese Schicht berücksichtigt bei der Verarbeitung sämtliche Daten und achtet auf das Erbringen exakter Ergebnisse. Im Gegensatz zum Speed Layer müssen die Daten nicht im Speicher gehalten werden. Die Schicht ermöglicht die Durchführung von Fensterfunktionen. Die Ergebnisse des Batch Layer werden im Serving Layer abgespeichert.

Speed Layer. Da Daten in der Echtzeit-Ereignisverarbeitung ihre Aktualität verlieren, muss es eine Möglichkeit zur direkten Verarbeitung geben. Der Speed Layer hat zur Aufgabe, möglichst aktuelle Daten zu liefern. Dabei wird nicht auf Datenvollständigkeit und -korrektheit geachtet. Durch die hohe Frequenz, in welcher die Daten empfangen werden, ist eine vollständige Verarbeitung nicht möglich bzw. gar nicht notwendig. Verschickt ein Temperatursensor Messwerte in Millisekunden-Abständen, wäre auch eine Abarbeitung jedes tausendsten Sensorwerts ausreichend. Ein Ausschöpfen sämtlicher Daten würde an dieser Stelle keinen Mehrwert bedeuten. Hier geht es um eine Minimierung der Verzögerungslücke, welche durch den Batch Layer verursacht wird. Die Verarbeitung der Daten geschieht im besten Fall vollständig in-Memory. Auch dieser Layer sendet die Ergebnisse an den Serving Layer.

Serving Layer. Der Serving Layer weist nach Empfang der Daten einen aktuellen Zustand auf. Eine Vereinigung der Schichten führt zu einer Echtzeitverarbeitung der Daten, welche sich im Serving-Layer befinden.

Die Umsetzung der Lambda-Architektur erfolgt in vielen Fällen nicht in reiner Form. Heterogene Systemlandschaften sehen dieser zwar ähnlich, weisen allerdings eine andere Interaktion zwischen den Komponenten auf, da beispielsweise verschiedene Anwendungsfälle unterschiedliche Schwerpunkte haben [16]. Dieser Ansatz hat allerdings den negativen Aspekt, dass Teile der Berechnungen doppelt ausgeführt werden müssen. Abhilfe für diesen Nachteil könnte die Kappa-Architektur schaffen.

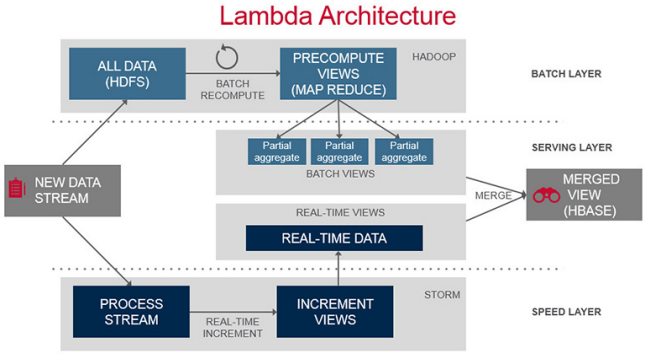

Abbildung 5. Aufbau der Lambda-Architektur

6 Stream Analytics und Apache Storm im Vergleich

Die beiden Technologien als Produkt lassen keinen direkten Vergleich zu. Azure Stream Analytics stellt bereits eine funktionsfähige Software dar, die über das Portal in Form eines Dienstes bezogen wird. Apache Storm hingegen muss vorab Aufgesetzt werden und beinhaltet keine Features wie eine SQL-ähnliche Abfragesprache und spezielle Funktionen für

die Ereignisverarbeitung. In diesem Abschnitt werden jedoch einige Aspekte aufgeführt, die sich gegenüberstellen lassen.

6.1 Systemlandschaft

Azure Stream Analytics stellt ein abgeschlossenes CEP-System dar. Hingegen Apache Storm ist lediglich ein Stream Processing-Framework, welches für Berechnungen auf kontinuierlichen Datenströmen optimiert ist [14]. Somit ist es ein kein CEP-System an sich, aber ein wichtiger Bestandteil davon. Um dies zu erlangen ist die Verknüpfung von weiteren Tools wie Kafka, Trident und Esper [26] erforderlich [24].

6.2 Lizenzmodell und Kosten

Azure Stream Analytics ist ein kommerzieller Dienst, welcher nur innerhalb des Azure-Portals genutzt werden kann. Hier kommt das Pay-As-You-go-Geschäftsmodell zum Einsatz, welches den Nutzern die Möglichkeit gibt, ausschließlich für den verursachten Datenverkehr zu bezahlen [17]. Desto höher das Datenvolumina, desto teurer wird die Nutzung. Allerdings steht bei Azure nicht nur das Kontingent der verarbeiteten Daten im Vordergrund. Die bereits erwähnten Suits können auch über ein monatliches Abonnement erworben werden. Damit können sowohl Kleinunternehmen als auch Großunternehmen dieses Produkt nutzen. Microsoft hat zudem die Cortana Analytics Suite für Unternehmen jeder Größe entwickelt. Kleinere Unternehmen können davon profitieren, dass keine Investitionen oder teure Software-Lizenzen für den Start benötigt werden. Größere Unternehmen können von der Skalierbarkeit der Azure Cloud-Plattform profitieren [14]. Apache Storm hingegen steht unter der Apache-Lizenz und ist somit kostenlos nutzbar [18].

6.3 Skalierbarkeit und Hardwareanforderung

Da Azure sämtliche Hardwareressourcen bereitstellt, bestehen für die Nutzer keine speziellen Hardwareanforderungen. Welche Anforderung Stream Analytics an die genutzte Cloud-Infrastruktur zugrunde legt, bleibt Microsoft vorbehalten und ist nicht bekannt [19]. Die Skalierung wird seitens Microsoft automatisch geregelt, sodass ein Event-Verkehr von einem Gigabyte pro Sekunde verarbeitet werden kann [20]. Apache Storm dagegen ist eine Standalone-Applikation und kann daher auf eigenen Systemen betrieben werden. Laut Apache kann mit 24 Gigabyte Arbeitsspeicher und 2-Intel-Prozessoren ein Kontingent von 100 Million Byte-Messages verarbeitet werden [21]. Auch hier sind keine konkreten Hardwarebedingungen gegeben. Diese sind vom Anwendungsfall und der Anzahl Knoten abhänig [19].

6.4 Verfügbarkeit

Beide Analyse-Plattformen versprechen hier eine Verfügbarkeit von 99.9 Prozent. Für eine Hochverfügbarkeit bietet Azure eine Wiederherstellungsmöglichkeit. Bei Apache Storm trägt der Benutzer die Verantwortung der Sicherheit seiner Daten [19] [21].

6.5 Entwicklungsumgebung

Stream Analytics unterstützt zwei Ansätze, einen Auftrag zu erstellen: Azure-Portal und REST. Im ersten Fall kann der Benutzer über eine entsprechende Benutzeroberfläche einen Job erstellen [22]. Die zweite Möglichkeit geht über eine REST-Schnittstelle, welche über Quellcode angesprochen wird. Somit ist es möglich via eigener Implementierung Jobs zu erstellen, diese zu automatisieren und zu re-/konfigurieren [23]. Apache Storm erfordert ein eigenverantwortliches Aufsetzen des gesamten Systems [24].

6.6 Abfragesprachen

Wie bereits erwähnt stellt Microsoft eine SQL-ähnliche Sprache zur Verfügung, mit welcher Abfragen erfolgen. In Kombination mit Scope definiert ASQL eine neue deklaravative Big-Data-Sprache [25]. Somit werden temporale Operatoren unterstützt. In Apache Storm sind Abfragesprachen nicht standardmäßig enthalten. Hierfür muss der Anwender Programmcode schreiben, was Programmierkenntnisse voraussetzt. Auch Fensterfunktionen werden nicht unterstützt und erfordern eine eigene Implementierung [19]. Allerdings kann Apache Storm durch Einsatz von Fremdbibliotheken an Abfragesprachen, wie zum Beispiel "Esper" erweitert werden [26].

6.7 Debugging-Unterstützung

Zum Debuggen steht ein Protokoll zur Verfügung, welches den Auftragsstatus und die Auftragsvorgänge aufzeigt. Jedoch können Benutzer die Debugg-Inhalte nicht anpassen. Der Anwender kann keine personalisierte Logdatei erstellen. Auf Seitens Apache Storm gibt es kein automatisches Logging. Dies liegt vollständig in der Hand des Entwicklers. Er kann das Logging dafür frei gestalten [27].

6.8 Datenformate

Azure Stream Analytics beschränkt sich auf die Datenformate Avro, JSON und CSV. Apache Storm hingegen kann beliebige Datenformate nutzen. Das ermöglicht eine dynamischere Entwicklung [5].

7 Diskussion

Storm genießt einen großen Einfluss in der Big-Data-Community und ist äußerst skalierbar, fehlertolerant und kann in bestehende Technologien integriert werden. Große Namen wie Twitter greifen darauf zurück, um ihre Datenmengen in Echtzeit zu verarbeiten. Allerdings ist die Einrichtung von Apache Storm keinesfalls trivial. Anfängliche Investitionen in Hardware und womöglich Software können anfallen. Dies ist der große Vorteil von Azure Stream Analytics, welches im Grunde keine Einrichtung und/ oder Wartung erfordert, da es von Microsoft als Dienst (Software-as-a-Service) angeboten wird. Nicht jeder Anwender verfügt über fundamentale Programmierkenntnisse. Zumal Storm an sich keine CEP Engine bietet. Hier muss ein weiteres Produkt wie Esper mit eingebunden werden, um auch komplexe Ereignisse über Regelwerke und Interpreter zu erkennen und zu verarbeiten. Dies ist bei Azure Stream Analytics nicht

der Fall ist. Die Engine steckt bereits im Dienst. Daher ist Azure Stream Analytics ein profitables Produkt für Unternehmen, die einen umfangreichen initialen Aufwand nicht stemmen wollen oder können. Dies mag unter anderem an mangelndem Expertenwissen liegen. Das Einrichten eines Azure Stream Analytics-Jobs beschränkt sich hier auf wenige Klicks im Azure-Portal. In ihm werden sämtliche Ein- und Ausgaben konfiguriert sowie entsprechende Abfragen ausgeführt. Auf die Skalierbarkeit wird bei Azures Diensten viel Wert gelegt. Ein einzelner Job kann bereits Millionen Ereignisse pro Sekunde verarbeiten. Da es mit anderen Diensten auf der Azure-Plattform angeboten wird, erlaubt es außerdem ein nahtloses Kombinieren verschiedener Azure-Services. Dazu zählen die in diesem Artikel aufgelisteten Produkte (Azure Storage, Azure SQL Database und Microsoft Power BI). Wie aus der Arbeit hervorging, werden auch einige, häufig vorkommende Szenarien als vorkonfigurierte Lösung angeboten um die anfänglichen Hürden noch weiter zu reduzieren.

Durch die von SQL abgeleitete Abfragesprache können Nutzer bereits eine bekannte Sprache verwenden, um die Daten zu analysieren oder persistent zu speichern usw. Es zielt zunehmend auf Analysten ab, da keine Programmierexpertise vorausgesetzt wird. Zudem fällt das Aufsetzen von bestimmten Umgebungen vollständig weg. Jedem Nutzer wird dabei ein Ansprechpartner delegiert.

Während bei Storm die Verarbeitungslogik mit Spouts und Bolts umgesetzt wird, weiß man bei Azure Stream Analytics nicht direkt, wie die Daten intern verarbeitet werden. Einiges deutet darauf hin, dass auch Azure Stream Analytics ähnlich vorgeht. Logisch betrachtet bleibt auch nur ein solcher Ansatz. Entsprechende Agents werden hier die Ausführungseinheiten darstellen. Für die Echtzeitverarbeitung der Daten wird auch bei Microsofts Lösung auf die Lambda-Architektur gebaut. Die Untersuchungen ergaben, dass die Kappa-Architektur trotz potentiell schnellerer Verarbeitung keine Alternative darstellt. Für einige Szenarien ist es erforderlich, Daten über einen gewissen Zeitraum zwischenzuspeichern. Genaue Einblicke gewährt Microsoft hier nicht. Der Anwender muss allerdings auch nicht wissen, was unter der Haube steckt. Kein Software-Know-How ist erforderlich. Es ist möglich eine entsprechende Lösung zu konfigurieren, ohne zu wissen, wie sie intern funktioniert. Bei Storm hingegen braucht man umfangreiche Einblicke in die Technologie, da ansonsten das Implementieren einer Lösung unmöglich wäre.

8 Ausblick

Obwohl Azure Stream Analytics bereits einige Zeit auf dem Markt ist, gibt es keinerlei wissenschaftliche Artikel, welche sich kritisch mit der Thematik auseinandersetzen. Es sind lediglich von Microsoft zur Verfügung gestellte Literatur, Beispiele und Dokumentationen zu finden. Aufgrund der stark begrenzten Quellen, war es leider nicht möglich tiefgreifende Einblicke zu erhalten. Die offiziellen Angaben der Anbieter klingen sehr vielversprechend, doch eine Überprüfung dieser Werte steht noch aus. Auch eine Frage bleibt offen. Wie bereits erläutert, unterstützt Azure Stream Analytics drei Fensterfunktionen. Den Autoren stellt sich die Frage, warum drei unterschiedliche Zeitfenster angeboten werden, jedoch keine Längenfenster. Grundsätzlich reicht auch eine Zeitfensterfunktion, welche nur über den zeitlichen Versatz unterschiedlich konfiguriert werden könnte.

Trotz dem nicht offengelegten Hintergrund bzgl. der präzisen Funktionsweise von Azure Stream Analytics sind die Autoren der Meinung, dass es eine gewinnbringende Plattform darstellt. Künftig wird die Entwicklung dieses Dienstes verfolgt. Der bisherige Mangel an wissenschaftlichen Artikeln könnte sich noch ändern, da schließlich immer mehr Geräte Teil einer IoT-Lösung werden und somit immer mehr Daten das Internet förmlich überfluten und analysiert werden müssen. Interessant bleibt auch die Entwicklung im Machine Learning-Bereich. Hier ergeben sich mächtige Erweiterungen zum Erkennen von Anomalien.

Literatur

[1] Dr Floarea Serban and Dr. Yves Brise, "Complex event processing: Eine strategische big data entscheidung," 2014. [Online]. Available: https://ipt.ch/complex-event-processing-eine-strategische-big-data-entscheidung/

[2] Statista, "Iot: number of connected devices worldwide 2012-2025 — statista," 2017. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[3] J. Prosise, "Understanding azure stream analytics," 2016.

[4] P. Tracy, "What is complex event processing and why is it needed for iot?" 2016. [Online]. Available: https://www.rcrwireless.com/20161122/fundamentals/cep-iot-tag31-tag99

[5] S. Klein, *IoT Solutions in Microsoft's Azure IoT Suite*. Berkeley, CA: Apress, 2017.

[6] B. Familiar and J. Barnes, *Business in Real-Time Using Azure IoT and Cortana Intelligence Suite*. Berkeley, CA: Apress, 2017.

[7] Gesellschaft für Informatik, "Complex event processing (cep)," 2009. [Online]. Available: https://gi.de/informatiklexikon/complex-event-processing-cep/

[8] A. Bruening, "Event-processing und stream-analyse," 2016.

[9] Microsoft, 2017. [Online]. Available: https://azure.microsoft.com/de-de/blog/stream-analytics-compliance/

[10] ——, "Stream analytics - datenanalyse in echtzeit — microsoft azure," 2017. [Online]. Available: https://azure.microsoft.com/de-de/services/stream-analytics/

[11] Statista, 2017. [Online]. Available: https://de.statista.com/statistik/daten/studie/779775/umfrage/

[12] P. Tsai, 2016. [Online]. Available: https://www.computerwoche.de/a/aws-versus-azure-die-vor-und-nachteile-aus-sicht-von-it-experten,3323128

[13] searchdatacenter, 2017. [Online]. Available: http://www.searchdatacenter.de/definition/Microsoft-Hyper-V

[14] Microsoft, 2017. [Online]. Available: https://azure.microsoft.com/

[15] OPITZ CONSULTING, "Big data für macher - lambda-architektur und hadoop framework in der praxis — ein fachartikel von opitz consulting," 2017. [Online]. Available: https://www.opitz-consulting.com/fileadmin/user_upload/Collaterals/Artikel/it-management-2017-10_big-data-fuer-macher_berle_sicher.pdf

[16] Lukas Berle, OPITZ CONSULTING Deutschland GmbH, "Lambda-architektur in der praxis," 2017.

[17] Microsoft, "Pricing - stream analytics — microsoft azure," 2017. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/stream-analytics/

[18] Apache, "Apache license, version 2.0," 2004. [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0.html

[19] Microsoft, "Analyseplattformen: Vergleich von apache storm mit stream analytics," 2017. [Online]. Available: https://docs.microsoft.com/de-de/azure/stream-analytics/stream-analytics-comparison-storm
[20] ——, "Introduction to stream analytics," 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction
[21] Apache, "Project information," 2017. [Online]. Available: http://storm.apache.org/about/scalable.html
[22] Microsoft, "Stream analytics rest api," 2017. [Online]. Available: https://docs.microsoft.com/de-de/rest/api/streamanalytics/?redirectedfrom=MSDN
[23] ——, "Erstellen eines auftrags zur verarbeitung von datenanalysen für stream analytics," 2017. [Online]. Available: https://docs.microsoft.com/de-de/azure/stream-analytics/stream-analytics-create-a-job#next-steps
[24] Apache, "Apache storm," 2017. [Online]. Available: http://storm.apache.org/
[25] Microsoft, "Stream analytics query language reference," 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn834998.aspx
[26] espertech, "Esper," 2016. [Online]. Available: http://www.espertech.com/esper/
[27] community.hortonworks, "Debugging an apache storm topology - hortonworks," 2016. [Online]. Available: https://community.hortonworks.com/articles/36151/debugging-an-apache-storm-topology.html