Echtzeit Gerätedaten Nachrichtenübermittlung als Grundlage für datengetriebene Lösungen wie das IIoT

Lennart Severin

Hochschule Darmstadt - Fachbereiche Mathematik und Naturwissenschaften & Informatik

Zusammenfassung und Schwerpunkte

Die kontinuierlich steigende Abhängigkeit industrieller Unternehmen von digitalen Infrastrukturen führt zur Notwendigkeit, disparate und abhängige Applikationen in einer einheitlichen Struktur zu verbinden. *Enterprise Application Integration* (*EAI*) ist ein Prozess um Systeme in eine Struktur zu integrieren, sodass Informationen und Ressourcen geteilt werden können [1].

Ziel dieser Arbeit war es, basierend auf den Prinzipien von *EAI* eine entkoppelte, eventgetriebene und nachrichtenbasierte Integrationsarchitektur zu entwerfen. Diese Architektur sollte in die bestehende Infrastruktur von *Musterunternehmen* (*MU*) integrierbar sein. Weiterhin sollten Änderungen im Lebenszyklus von Geräten, welche von *MU* produziert werden, erkannt, eingeordnet und zur Verfügung gestellt werden. Solche Änderungen können beispielsweise die Produktion, der Verkauf oder eine Qualitätskontrolle eines Geräts sein und werden als Events bezeichnet.
Zur Umsetzung dieses Ziels wurde zunächst die bestehende Infrastruktur analysiert und eine geeignete nachrichtenbasierte Lösung identifiziert.
Anschließend wurden vorhandene Events, welche zu duplikativen Einträgen im Datenbestand führen, analysiert und mittels ausgewählter statistischer Methode eingeordnet. Im Fokus standen dabei Duplikate, die auf einen systematischen Fehler oder einen Tippfehler zurückzuführen sein könnten. Es sollte bestimmt werden, ob Abweichungen zwischen den Duplikaten so klein sind, dass eine automatische Zusammenführung angebracht ist.
Schließlich wurde eine Architektur konzipiert und Methoden zur Publizierung von Events identifiziert. Die erarbeitete Methode zur Einordnung der Events wurde in der Architektur implementiert und die Ergebnisse zur Initiierung weiterer Prozessschritte persistiert. Darüber hinaus wurde die entwickelte Architektur auf ihre Echtzeitfähigkeit und Skalierbarkeit unter verschiedenen Last- und Verarbeitungsszenarien geprüft. Diese Prüfung erfolgte mittels dreier Testreihen.

Die drei Schwerpunkte lassen sich wie folgt zusammenfassen:

1. Entwicklung einer in die bestehende Infrastruktur integrierbaren, eventgetrieben und nachrichtenbasierten Integrationsarchitektur.
2. Analyse und Einordnung von Duplikaten anhand ausgewählter statistischer Methoden.
3. Umsetzung einer Architektur anhand eines Anwendungsfalles und die Prüfung der Echtzeitfähigkeit und Skalierbarkeit unter verschiedenen Last- und Verarbeitungsszenarien.

Verfübare Daten

Der verfügbare Datensatz besteht aus 61,937,797 Einträgen aus den Jahren 2000 bis 2021. Selektiert nach doppelten Serialnummern entsprechen rund 360,000 der zu analysierenden Menge. In Tabelle 1 wird ein Auszug des Duplikatsdatensatzes mit ausgewählten Merkmalen dargestellt.

SN	MN	MT	OC	T	CN	CCC	CPON	FV	FB
725545006	56004119	KMAT	DTRANSRW01	Ident.Nr. 4226	30050497	CH		1.03.00	HART
725545006	56004142	HAWA	DTRANSRW01	Ident.Nr. 4226	30050497	CH	Mail H.R.		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3745101003	71034285	HAWA	BA304D	30-M6-6	34000944/G	GB	8034173		
3745101003	83402582	HAWA	BA304D	30-M6-6	34000944/G	GB	8034173		0 to

Tabelle 1: Auszug des Duplikatsdatensatzes mit den verbliebenen Merkmalen nach der Merkmalsselektion.

Vorgehensweise

Zunächst wurde zur Konzeptionierung der Integrationsarchitektur, eine Analyse der bestehenden Infrastruktur durchgeführt. Zur Selektion einer geeigneten *Enterprise Messaging* (*EM*)-Lösung wurde eine Nutzwertanalyse verwendet [2]. Im Fokus der Analyse standen qualitative Merkmale wie die Zukunftsperspektive, Integrier- und Nutzbarkeit sowie die Funktionalität. Verglichen wurden Apache Kafka, SAP Event Mesh und Microsoft Event Hub. Die Analyse der Systeme erfolgte anhand von 10 Kriterien, wobei 4 Kriterien als erforderlich und 6 Kriterien als optional eingestuft wurden. Event Mesh erzielte den größten Nutzwert und wurde als Basis für die Konzeption verwendet.
Anschließend wurden die duplikativen Eintrage analysiert. Im Mittelpunkt stand die Identifizierung systematischer Zusammenhänge der Merkmale und Abweichungen zwischen den Ausprägungen. Zunächst wurde eine Merkmalsselektion basierend auf einem Vergleichsdatensatz ohne Duplikate durchgeführt. Von 21 vorhandenen Merkmalen wurden 12 verworfen.

Anschließend wurde die Damerau-Levenshtein-Distanz zur Bestimmung des Editierabstands zwischen den Merkmalsausprägungen des ersten Eintrags und etwaigen Duplikaten berechnet. Mithilfe dieser Distanz wird die minimale Anzahl an Operationen zur Transformation eines Strings in einen anderen String bestimmt. Die Besonderheit dieses Distanzmaßes ist es, neben Einfüge- und Löschoperationen auch die Transposition benachbarter Zeichen zu berücksichtigen [3]. Basierend auf Stichprobenuntersuchungen und unter Konsultation interner Experten wurden daraufhin Annahmen zu automatisch zusammenführbaren Einträgen getroffen. Es wurden Hypothesen zu den systematischen Zusammenhängen von 3 Merkmalskombinationen (7 Merkmale) sowie ein Grenzwert für den Editierabstand formuliert. Zur Prüfung der systematischen Zusammenhänge wurde einer Faktoranalyse durchgeführt [4]. Die Zuordnung der Daten und die Prüfung des Grenzwertes erfolgte mittels dem K-Means Clustering-Verfahren [5].
Basierend auf der initial selektierten *EM*-Lösung wurde anschließend die Zielarchitektur in allgemeiner Form und im Bezug zu dem Anwendungsfall konzipiert.

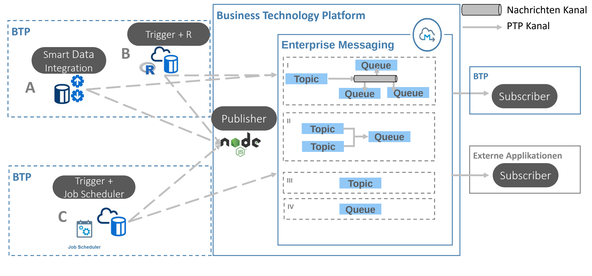
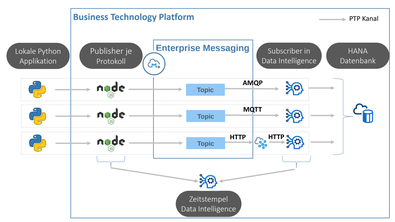
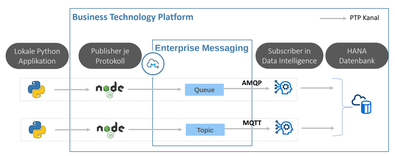


Abbildung 1: Aufbau der eventgetriebenen Integrationsarchitektur.

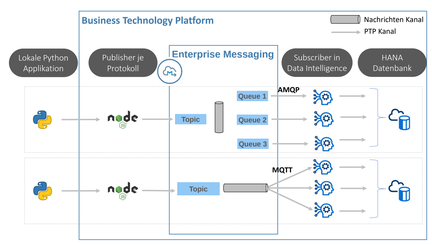
In Abbildung 1 ist die erarbeitete Zielarchitektur für den Anwendungsfall der Detektion und Publikation von Änderungen in den Gerätestammdaten dargestellt. Die linksseitig aufgezeigten Varianten A, B und C sind Mechanismen, welche Änderungen detektieren und an einen Publisher oder direkt an das *EM*-System senden. Nach Empfang der Nachricht stehen die Subskriptionsvarianten I bis IV zur Verfügung, um die Nachricht für weitere Applikationen vorzuhalten, zu vervielfältigen und zuzustellen. Als Subscriber wurde eine Applikation zur Transformation, Einordnung und Persistierung der Daten in der *Business Technology Platform* (*BTP*) implementiert und verwendet. Die Allgemeine Architektur wird durch Ersetzen der linksseitigen Publiziervarianten A bis C mit einer beliebigen internen oder externen Applikation erreicht.
Im letzten Teil dieser Arbeit wurde die erarbeitete Architektur anhand dreier Testreihen auf ihre Echtzeitfähigkeit und Skalierbarkeit geprüft. Weiterhin wurden Hypothesen über das Verhalten des Systems unter diversen Lastszenarien aufgestellt und unter folgenden Konfigurationen überprüft: Testlänge von 5 und 30 Minuten, bis zu drei Protokolle, alle Subskriptionsvarianten I bis IV, bis zu drei aktive Subscriber und 35 bis 280 Nachrichten pro Minute. Weiterhin wird zwischen den Testreihen durch eine dedizierte Zeitstempelabfrage unterschieden. In Testreihe 1 (Abb. 2a) wird eine solche Abfrage durchgeführt. In Testreihen 2 (Abb. 2b) und 3 (Abb. 2c) hingegen, wird die Zeitsynchronisierung der *BTP* verwendet. Alle Applikationen und Services der *BTP* werden von Amazon Web Services gehostet. Zur Durchführung der Testreihen wurde die Detektion und Publikation der Änderungen durch ein lokales Skript ersetzt.



(a) Testreihe 1: Kurzzeit-Test



(b) Testreihe 2: Langzeit-Test

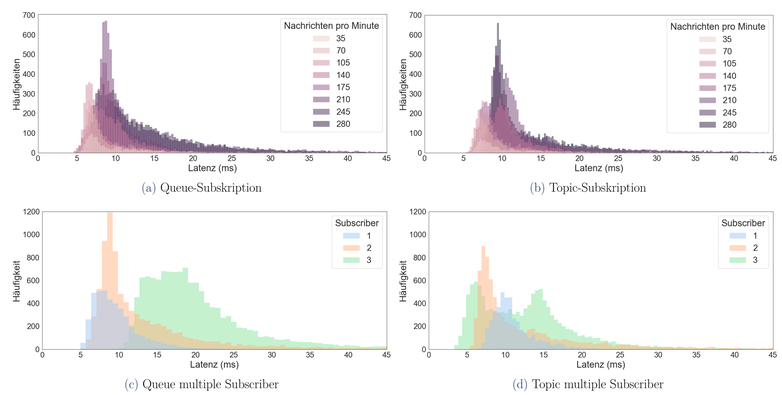


(c) Testreihe 3: Subscriber-Varianten-Test

Abbildung 2: Schematischer Aufbau der Testreihen 1 bis 3.

Ergebnisse

Hinsichtlich der Duplikatsannahmen konnte mithilfe der Faktoranalyse gezeigt werden, dass diejenigen Merkmalskombinationen mit systembedingten Zusammenhängen hohe Faktorladungen in jeweils einem Faktor aufweisen. Basierend auf dem K-Means Clustering-Verfahren wurde ein Cluster identifiziert, welches sich klar von den übrigen Clustern abgrenzt und aus 87.5% der angenommenen Duplikate besteht. Damit konnte der angenommene Grenzwert für automatisch zusammenführbare Einträge bestätigt und 12.16% aller Duplikate nach 2005 zugeordnet werden.
Mit Testreihe 1 wurde gezeigt, dass signifikante Unterschiede der Latenz zwischen HTTP und AMQP sowie HTTP und MQTT vorliegen. Weiterhin wurde eine Überlastung des Systems ab etwa 245 Nachrichten pro Minute identifiziert, welche auf die dedizierte Zeitstempelabfrage zurückzuführen ist. Alle Protokolle zeigten Latenzzeiten zwischen 200 und 450 Millisekunden für bis zu 245 Nachrichten pro Minute und erfüllen somit die Echtzeitforderung des Anwendungsfalles von Sekunden bis Minuten.
In Testreihe 2 wurde entgegen der Erwartung gezeigt, dass eine Queue-Subskription mindestens gleiche oder kleinere Latenzzeiten wie eine Topic-Subskription aufweist. Weiterhin konnte für steigende Nachrichten pro Minute ausgehend von 35 bis 280 (200% der Spitzenbelastung des produktiven Systems), keine durchgängig signifikante Steigerung der Latenz abgeleitet werden. Insbesondere die Randbereiche der Nachrichten pro Minute zeigten Abweichungen von der Annahme.
Mit Testreihe 3 konnte die Skalierbarkeit des Systems für bis zu drei parallel aktiven Subscribern gezeigt werden. Insbesondere die Queue-Subskription zeigte signifikante Unterschiede bei einer steigenden Anzahl von Subscribern. In Folge dessen wurde der Vergleich zwischen den Subskriptionsvarianten mit multiplen Subscribern durchgeführt. In diesem Fall zeigte die Topic-Subskription bei drei aktiven Subscribern signifikant kleinere Latenzzeiten als die Queue-Subskription. Dies könnte auf den technologischen Unterschied von der Topic- und Queue-Subskription zurückzuführen sein. Eine Queue hält die Daten solange vor, bis diese konsumiert wurden. Eine Topic hingegen stellt die Nachricht den aktiven Subscribern zu und löscht diese Nachricht anschließend. Sowohl Testreihe 2 als auch Testreihe 3 zeigten Latenzzeiten unter 20 Millisekunden in jeder Konfiguration, vgl. Abbildung 3.



(a) Queue-Subskription (b) Topic-Subskription

(c) Queue multiple Subscriber (d) Topic multiple Subscriber

Abbildung 3: Häufigkeitsverteilung der Latenz von Testreihe 2 (3a, 3b) und 3 (3c, 3d). Bei Testreihe 3 wurden 140 Nachrichten pro Minute gewählt, gemäß der Spitzenbelastung des produktiven Systems.

Fazit und Ausblick

Somit wurde eine eventgetrieben und nachrichtenbasierte Integrationsarchitektur erarbeitet, die skalierbar und in allen Kategorien echtzeitfähig ist. Zusätzlich wurde eine Methode zur Einordnung von Duplikaten erarbeitet, welche durch eine automatische Zusammenführung bereinigt werden können. Die Ergebnisse werden persistiert und stehen zur Initialisierung weiterer Prozesse zur Verfügung. Dies ermöglicht eine nahtlose Einbindung in bestehende Integrationsprozesse und eine Minimierung falscher Einträge für den zukünftigen Datenbestand.
Hinsichtlich der *EM*-Lösung könnten Varianten und Methoden zur Publizierung von businessgetriebenen Events untersucht werden. In diesem Kontext sollte eine Methode zur Provisionierung unterschiedlicher Anwendungsfälle konzipiert werden, um die Kapazitäten des *EM*-Systems entsprechend zu dimensionieren. Einhergehend mit der Provisionierung, könnten Mechanismen zur Datensicherung bei Systemausfall untersucht und entwickelt werden.

Quellen

[1] M. Fowler, *Patterns of Enterprise Application Architecture*. USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[2] C. H. Kepner and B. B. Tregoe, "The new rational manager: An updated edition for a new world," *Princeton, N. J. (P. O. Box)*, vol. 704, 1997.

[3] C. Zhao and S. Sahni, "String correction using the damerau-levenshtein distance," *BMC Bioinformatics*, vol. 20, p. 277, Jun 2019.

[4] R. Cudeck, "10 - exploratory factor analysis," in *Handbook of Applied Multivariate Statistics and Mathematical Modeling* (H. E. Tinsley and S. D. Brown, eds.), pp. 265–296, San Diego: Academic Press, 2000.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.