Energieforschungsprogramm

Publizierbarer Endbericht

Programmsteuerung:

Klima- und Energiefonds

Programmabwicklung:

Österreichische Forschungsförderungsgesellschaft mbH (FFG)

Endbericht

erstellt am

13/10/2023

Big Power Data: Transaktionssicherheit und Skalierbarkeit bei der Echtzeitverarbeitung von (Big) Power Data

Projektnummer: 39833532

Ausschreibung	7. Ausschreibung Energieforschungsprogramm
Projektstart	01/10/2021
Projektende	30/09/2023
Gesamtprojektdauer (in Monaten)	24 Monate
ProjektnehmerIn (Institution)	HAKOM Time Series GmbH
AnsprechpartnerIn	DI Stefan Komornyik
Postadresse	Lemböckgasse 61/Stiege 2/6. OG 1230 Wien
Telefon	+43 1 815 79 80 112
Fax	...
E-mail	stefan.komornyik@hakom.at
Website	hakom.at

BigPowerData (überschreiben)

Transaktionssicherheit und Skalierbarkeit bei der Echtzeitverarbeitung von (Big) Power Data (überschreiben)

AutorInnen:
Namen der mitwirkenden AutorInnen je ProjektpartnerInnen
Ricardo Mariense Wickert (HAKOM)
Martin Harrer (FG Hagenberg)
Olver Krauss (FH Hagenberg)
Christoph Praschl (FH Hagenberg)
Sebastian Pritz (FH Hagenberg)
Martina Zeinzinger (FH Hagenberg)
Javier Carredano (LeanXScale)
Ricardo Jimenez-Peris (LeanXScale)

1 Inhaltsverzeichnis

Es muss ein Inhaltsverzeichnis mindestens auf Überschriftebene 1 mit Seitenangabe erstellt werden!

Ein publizierbarer Endbericht sollte folgende Struktur (Index) besitzen und mindestens 15 Seiten/maximal 50 Seiten (inklusive Deckblatt, Inhalts-, Abbildungs- und Tabellenverzeichnis, exklusive Anhänge) haben.
Der Umfang des publizierbaren Endberichts von F&E-Dienstleistungen soll mindestens 50 Seiten/maximal 150 Seiten (inklusive Deckblatt, Inhalts-, Abbildungs- und Tabellenverzeichnis, exklusive Anhänge) betragenbesteht aus **mindestens 25 Seiten**.
Die unten angeführte **Darstellung ist eine Mindestanforderung** und kann bei Bedarf erweitert werden.
Vorrangiges Ziel der publizierbaren Berichte ist die Darstellung der wesentlichen Projektergebnisse.

Textformat

- Papierformat: A4 Hochformat
- Linker und rechter Rand: 2,5 cm
- Schriftformatierung: Arial, 11 Punkt, Zeilenabstand 1,3-fach
- Schriftformatierung für Tabellen: Arial, 10 Punkt
- Fußzeile: Seitennummerierung
- Definition der Überschriften bis zur 3. Ebene
  - Überschrift 1:
    Schriftformat: Arial, 16 Punkt und Fett
    Absatzformat: Abstand vor: 24 Punkt, Abstand nach: 12 Punkt
  - Überschrift 2:
    Schriftformat: Arial, 14 Punkt und Fett
    Absatzformat: Abstand vor: 24 Punkt, Abstand nach: 12 Punkt
  - Überschrift 3:
    Schriftformat: Arial, 11 Punkt und Fett
    Absatzformat: Abstand vor: 12 Punkt, Abstand nach: 6 Punkt

2 Einleitung

Aufgabenstellung
Schwerpunkte des Projektes
Einordnung in das Programm
Verwendete Methoden
Aufbau der Arbeit

Die Menge der Daten in der Energiewirtschaft wächst aufgrund der starken Zunahme von Sensoren und durch die Erhöhung der Abtastrate der Messungen aktuell dramatisch an. Entscheidend sind dafür die folgenden Aspekte:

- Smart Meter Ausrollung - Die Stromrichtlinie im 3. EU-Binnenmarktpaket fordert die Einführung von „intelligenten Messsystemen“, also Smart Meter, für alle Verbraucher. Es war geplant, dass mindestens 80 % aller Stromkunden bis spätestens 2020 einen Smart Meter erhalten müssen. Aus aktuellen Gründen ist die Ausrollung verzögert worden, perspektivisch entsteht hier ein immenser Datendruck.
- Ausbau der erneuerbaren Energieerzeuger – im Rahmen der Klima- und Energiepolitik der Europäischen Union wurde der Ausbau des Anteils von Energie aus erneuerbaren Quellen auf mindestens 32% festgesetzt.
- Ausbau der eMobility – im gleichen Zuge wurde die Senkung der Treibhausgasemissionen um mindestens 40% gegenüber 1990 beschlossen.
- Verfügbarkeit von hochfrequenten Power Quality Sensordaten – die mit dem starken Ausbau der erneuerbaren Energieerzeugung verbundenen Herausforderungen in Bezug auf Netzstabilität haben die Entwicklung hochfrequenter Sensoren beschleunigt.

Es ist die zentrale Herausforderung der Energiewirtschaft geworden, unter diesen dynamischen Rahmenbedingungen die Netzstabilität weiterhin dauerhaft zu gewährleisten. Dies gelingt nur dann, wenn 2 Voraussetzungen gleichermaßen gegeben sind:

- Die gemessenen Datenströme müssen in nahezu Echtzeit verarbeitet werden können. Umstände, die die Netzstabilität verringern müssen zur Echtzeit erkannt, und idealerweise bereits im Vorfeld, identifiziert werden.
- Die Messungen müssen nachvollziehbar sicher verarbeitet werden können – technisch gesehen gewährleistet dies die sogenannte transaktionale Konsistenz bei der Verarbeitung der Daten. Diese Eigenschaft gewährleistet, dass beim Einfügen, Verändern oder Löschen von Daten der Datenbestand weiterhin widerspruchsfrei ist, sodass keine inkonsistenten Daten auftreten.

Selbstverständlich ist es notwendig Störfälle lückenlos rekonstruieren zu können, um mögliche Ursachen zukünftig zu vermeiden und auch Maßnahmen in Zukunft verbessern zu können. Hierfür ist eine konsistente Datenlage die Grundvoraussetzung. Nicht zuletzt ist Transaktionssicherheit speziell bei einem Stromausfall notwendig, um abgebrochene Datenbanktransaktionen wieder zurückrollen zu können.

Der Löwenanteil der wesentlichen Datenströme in der Energiewirtschaft sind Zeitreihen – Datensätze, deren Schlüsselinformation der Zeitstempel ist. Die Verarbeitung solcher Zeitreihen in der Energiewirtschaft in Zeitreihenmanagementsystemen (Time Series Management – TSM) setzt die

Entwicklung domänenspezifischer Lösungen voraus. Nur so ist es möglich, von den Fortschritten von Online Transaction Processing (OLTP) -Datenbanken hinsichtlich Skalierbarkeit zu profitieren und auf diese Weise die notwendigen Durchbrüche in der Echtzeitverarbeitung und in der Anwendung von KI-basierten Algorithmen zu ermöglichen.
Um die Netzstabilität und die versorgungskritischen Prozesse weiterhin gewährleisten zu können, ist es erforderlich, die großen Mengen an Daten (Zeitreihen) in Echtzeit verarbeiten zu können. Deswegen wurde der Fokus des Projektes auf die Konsistenz und Skalierbarkeit der Transaktionen in dem verteilten Energiedaten-System gelegt.
Die im Projekt involvierten Partnerunternehmen haben ins Projekt mit ihren speziellen Nischentechnologien – Zeitreihenmanagement im Energiesektor einerseits und sowohl horizontal skalierbare als auch transaktional konsistente Datenbanken andererseits – spezielle Erfahrungen gebracht. Wissenschaftlich hat die FH Oberösterreich, Fakultät für Informatik, Kommunikation und Medien, die Entwicklung mitbegleitet und unterstützt.
Die folgenden 3 Aspekte wurden als Überziele für das Entwicklungsprojekt BIG POWER DATA definiert:

1. Entwicklung einer TSM-Lösung für den Energiesektor die mit horizontal skalierbaren Datenbanken mit transaktionaler (ACID) Konsistenz in Echtzeit funktioniert.
2. Ausbau einer entsprechenden Datenbanktechnologie zur optimalen Verarbeitung von Zeitreihen aus dem Energiesektor (Zusammenspiel von Transkationen über verteilte TSM- und Persistenz-Services).
3. Ermittlung der Grenzen anhand von Echtdaten für anspruchsvolle Anwendungsfälle aus der Energiewirtschaft für horizontal skalierbare Datenbanken.

Die Methoden der empirischen Forschung wurden im Projekt angewandt, und zwar ein Modell wurde vorgeschlagen und eine Reihe an Experimenten (Performance Messungen der unterschiedlichen Datenbanksysteme) wurden definiert, durchgeführt und gemessen. Die Messergebnisse wurden mit statistischen Methoden ausgewertet, um die Performance dieses Modells zu messen, basierend auf einem repräsentativen Fall. Somit wurde das Modell validiert und das Verfahren wiederholt.
Im Projekt wurden spezielle Benchmark-Methoden umgesetzt, um eine Evaluierung der angestrebten Ziele sicherzustellen. Das Entwicklungskonsortium hat sich darauf verständigt, neben IoT-Benchmarks [33, 34] auch einen Ansatz, angelehnt an COST (Configuration that Outperforms a Single Thread) dem Projekt zu Grunde zu legen [24]. HAKOM ist seit dem Launch von PowerTSM (https://www.powertsm.com/) in der Lage in der Cloud Big Data Skalierung für ihren Service anzubieten. Daraus entstand eine auf HAKOM ausgerichtete Benchmark Suite, die auch den Overhead durch die TSM-Plattform vermessen kann und somit ermöglicht, dass man die Kosten über die Applikations- und Persistierungsschicht hinweg optimieren kann. Der Overhead von Aggregierungen kann zusätzlich zu Read und Write ausgewiesen werden [35, 36]. Die Möglichkeit durch diese Art der Messung die Skalierung zwischen der Persistierungs- und Applikationsschicht auszubalancieren, geht über das ursprüngliche Projektziel hinaus. Die Anlehnung an COST wurde beibehalten und auf unterschiedliche Datenbanksysteme und die darüberliegende TSM-Schicht angewendet.
Damit die Ziele des Projektes erreicht werden können, wurden die folgenden Arbeitspakete definiert:

1. Projektmanagement (HAKOM)
2. Transaktionen in verteilten Energiedaten-Systemen (HAKOM, LeanXScale)
3. Dynamische Skalierung unterschiedlichen Servicetypen (HAKOM, FH Oberösterreich)

4. Umsetzung einer Konfigurations- und Überwachungslösung (HAKOM, FH Oberösterreich)
5. Pilotierung einer skalierbaren TSM-Lösung für Big Power Data (HAKOM)

Die Aufgaben mancher Arbeitspakete waren eng miteinander verbunden und sind meistens gleichzeitig gelaufen. Das Projektmanagement umfasste die gesamte Laufzeit des Projektes und beinhaltete die Koordination der Aufgaben, deren Kontrolle und Kommunikation zwischen den Projektpartnern. Auf der Entwicklungsseite wurden Anpassungen in der Architektur vom HAKOM Framework vorgenommen, so dass ein LeanXScale Connector entstehen könnte. Darüber hinaus wurde ein Benchmark Framework entwickelt, das ermöglicht, die Datenlast automatisch zu generieren und die unterschiedlichen Datenbanksysteme mit und ohne TSM einheitlich zu vergleichen. Dadurch wurden die Grundlagen geschaffen, die Konfigurations- und Monitoring-Möglichkeiten bei HAKOM TSM in Richtung Platform-as-a-Service (PaaS) bzw. eventuell zu Software-as-a-Service (SaaS) weiterzuentwickeln.

3 Inhaltliche Darstellung

Inhaltliche Darstellung wurde nach der Beschreibung der Ergebnisse aus jedem Arbeitspaket aufgebaut.

3.1 Projektmanagement

Zur Abstimmung der Konsortialpartner über den Projektfortschritt und anstehende Probleme wurden regelmäßige Wochenmeetings durchgeführt. In diesen fand weiterhin auch Know-How-Transfer statt und sie wurden auch zur Abstimmung von Implementierungsvarianten genutzt, wenn diese für alle relevant waren. Daher fielen auch in Zeitraum nach dem Zwischenbericht keine Reisekosten für Konsortialmeetings an. Die Kosten werden auf Seiten der FH Hagenberg für Konferenzbesuche genutzt. Neben der Publikation für die Vermessung von Single-Nodes [35] konnte in der zweiten Projekthälfte ein Paper zu den Messungen mit parallelen Zugriffen eingereicht werden [36]. DI (FH) Dr. Florian Eibensteiner und DI (FH) Martina Zeinzinger vom Embedded Systems Lab haben auch in der zweiten Projekthälfte DI Dr. Andreas Stöckl und DI Rimbert Rudisch-Sommer ersetzt.
DI (FH) Dr. Florian Eibensteiner und DI (FH) Martina Zeinzinger vom Embedded Systems Lab ergänzen daher die Mitglieder der Forschungsgruppen AIST und WinLab im Bereich Maschine Learning und Zeitserien, weil bei BIG POWER DATA der spezielle Aspekt der Zeitserien [1, 2, 3, 4, 6 ,8, 12, 13, 14, 15] im Zentrum der Forschung steht. Die Stundenunterschreitung in der ersten Projekthälfte konnte gut gemacht werden, sodass am Projektende ausgeglichen budgetiert wurde.
Prof. Martin Harrer ist leider am 15. Juni 2023, verstorben. Als Kontakt für den Projektabschluss steht Oliver Krauss (bereits im Projektteam) bereit.

Fazit

Die angestrebten Messungen für skalierende ACID-Transaktionen im Vergleich zwischen mehreren Datenbanken mit konkurrierenden Ansätzen konnten mit Datenbanken aus dem Bereich SQL, NoSQL und NewSQL, jedoch nicht mit LeanXcale umgesetzt werden. Messungen von Read, Write, Aggregationen und Overhead aus der TSM-Schicht können getrennt ausgewiesen und mit einer einheitlichen Messmethode mit einem definierten Hardwaresetup durchgeführt werden. Die

Messergebnisse liegen roh als JSON-Dateien, in Tabellenform und als, mittels Python erstellten, Plots vor.

3.2 Transaktionen in verteilten Energiedaten-Systemen

Die Arbeitspakete 2,3 und 4 sind in der Implementierung an einigen Stellen eng verknüpft und lassen sich daher nicht vollständig getrennt betrachten.
Im Projektzeitraum wurde der HAKOM Framework erweitert, um eine flexiblere Architektur bezüglich der Persistenzschicht zu ermöglichen. Diese Entwicklung führte zur Umsetztung eines HAKOM LeanXcale Connectors, der es ermöglicht, Datenbank ohne Stored Procedures anzusprechen. Diese Implementierung steht beispielhaft für weitere Datenbankanbindungen zu Systemen, die keine Stored Procedures unterstützen. Über Stored Procedures erfolgt das Mapping des Kundendatenmodells auf das HAKOM-Datenmodell. Das Mapping muss für Systeme wie LeanXcale in eine eigene Zwischenschicht gezogen werden. In einem ersten Schritt erfolgte eine Anbindung ohne Mapping, bei der das HAKOM-Datenmodell für den Kunden ohne Mapping umgesetzt wird. Diese Variante wurde auch für vergleichende Performancemessungen herangezogen, um den Overhead von HAKOM TSM getrennt von der eigentlichen Schreib- und Leselast auf der Datenbankebene vermessen zu können. Dadurch wird es möglich unterschiedliche Datenbanksysteme in Bezug auf ihre Fähigkeiten unter ACID zu skalieren zu untersuchen. Um einschätzen zu können, wie gut die Skalierung funktioniert, wurden im ersten Schritt die Lastmessungen gegen Single-Nodes durchgeführt und dann später mit Ergebnissen mit parallelem Zugriff verglichen. Zusätzlich befindet sich derzeit eine finale Publikation in Einreichung, welche diese Ergebnisse weiters um Cluster Lösungen ergänzt. Dadurch lässt sich in einem an COST [24] angelehnten Verfahren der Skalierungsoverhead ermitteln.
Alle Skalierungslösungen, die für den Antrag einer Evaluierung unterzogen wurden, weisen Skalierungsoverhead auf, der in seinem Ausmaß von der speziellen Implementierung abhängig ist. Kommen bei Skalierung ACID-Transaktionen ins Spiel, wird das Ausmaß der Skalierungseinbußen auch von verteilten Sperren bestimmt. Die Fähigkeit von LeanXcale [18], hier einen Ausweg zu bieten wurde durch die Messungen im Vergleich zu anderen Datenbanksystemen aus dem Umfeld relationaler, NoSQL und NewSQL Datenbanken [44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54], verglichen. MongoDB hat mit der Version 5.0 einen speziellen Zeitseriendatentyp eingeführt und unterstützt den Nutzer seit 5.6 besser beim DataCleansing. Es war ursprünglich angedacht beide Versionen zu vergleichen, um die Auswirkungen auf die Performancesteigerung zu messen. Aufgrund des Zeitplans war die Vermessung der älteren Version jedoch nicht möglich. MongoDB orientiert sich beim neuen Ansatz an LSM-Trees, die auch in Cassandra und CrateDB, das auf Lucene basiert, zum Einsatz kommt. Dadurch wird das Schreiben aus dem RAM auf Disk optimiert. TimescaleDB optimiert PostgreSQL in Bezug auf das Arbeiten mit Zeitserien.

Fazit

Im Projekt wurde der HAKOM LeanXcale Connector entwickelt, um die Datenbank ohne Stored Procedures anzusprechen und die Unterstützung von Transaktionen in einem verteilten Zeitreihenmanagementsystem zu verbessern. Die Anbindung ohne Mapping ermöglichte Performancemessungen, um den Overhead von HAKOM TSM zu untersuchen. Es wurden

Skalierungslösungen hinsichtlich der Skalierungsmöglichkeiten evaluiert. TimescaleDB und MongoDB optimieren zusätzlich das Arbeiten mit Zeitserien.

3.3 Dynamische Skalierung unterschiedlichen Servicetypen

Die Arbeitspakete 2, 3 und 4 sind in der Implementierung an einigen Stellen eng verknüpft und lassen sich daher nicht vollständig getrennt betrachten.
Im Zwischenbericht wurde bereits angemerkt, dass alle Skalierungslösungen, die einer Evaluierung unterzogen wurden, einen Skalierungsoverhead aufwiesen. Verschärft wurden Probleme mit Skalierungsoverhead durch die Berücksichtigung von verteilten ACID Transaktionen, bei denen auch das Ausmaß an Skalierungseinbußen durch verteilte Sperren bestimmt ist. Hierfür wurde eine Messuite entwickelt, die LeanXcale [18] mit anderen relationalen, NoSQL und NewSQL Datenbanken vergleichbar macht. Zusätzlich zur Single-Node Messung wurde die Messuite nun auf Multi-Node Messungen mit Cluster-Systemen in Datenbanken erweitert. Weiters werden spezielle Datentypen der Datenbanken z.T. in Betracht gezogen (so z.B. Incrementdatentyp in LeanXcale oder Zeitseriendatentypen in MongoDB und TimescaleDB). Details können der Veröffentlichung in [55] entnommen werden, die zur Publikation eingereicht wird (derzeit noch in Review).
Für die Erweiterungen wurden auch die AWS EC2 Instanzen erweitert. Es gibt pro Cluster mehrere Worker-Nodes, die für die Datenhaltung verantwortlich sind, und (Abhängig von der Datenbank) ein Management-Node dient zur Lastverteilung. Mehrere Client-Applikationen greifen sowohl schreibend als auch lesend auf die Datenbank zu, um die Skalierungsfaktoren der Datenbanksysteme vergleichbar messen zu können. Bei den Messungen wurde sichergestellt, dass ausreichend Last auf die Datenbanken ausgeübt wird.
Basierend auf der Skalierung wurde eine Analyse umgesetzt, die entscheidet, wann eine Skalierung aufwärts (System ist zu stark ausgelastet) oder abwärts (System ist nicht ausgelastet und Knoten können reduziert werden) relevant wird. Dieses Scaling ist für TimescaleDB nicht notwendig, da der Coordinator ein Bottleneck darstellt, das ein Skalieren der Lastverteilung verhindert (siehe Grafik *Prozessor-Auslastung der Clusterknoten*, der Blaue Coordinator ist voll ausgelastet, aber kein einziger der Worker ist unter voller Last).
Für die anderen Datenbanken wurde eine Scaling Methode entwickelt, die basierend auf der durchschnittlichen CPU Last aller Worker-Knoten entscheidet, ob Knoten hinzugefügt oder entfernt werden sollen. Die Methode ignoriert derzeit noch die Kosten des hoch/herunterfahrens eines Workers und wurde noch nicht in die Infrastruktur überführt. Die anderen Werte wie RAM und Speicher wurden nicht berücksichtigt, da sich CPU als das einzige Bottleneck für die DB Schreib und Lesevorgänge gezeigt hat.
Das Ergebnis der Scaling Methode kann für die Datenbank MongoDB in Grafik *Skalierungs-Entscheidung eingesehen werden.* Wie beim Lesen mit 3 Worker-Knoten zu sehen ist wird vor Systemlast, als auch nach dem Ende des Benchmarks ein Downscaling empfohlen, zu voller last jedoch ein Upscaling, das bei 5 bzw. 7 Knoten jeweils kürzer ist, weil die Last bereits ausreichend verteilt wird. Beim Write ist die Last auf 5 bzw. 7 Knoten bereits ausreichend verteilt, bei 3 Knoten wird jedoch ein Upscaling für fast den gesamten Verlauf empfohlen.

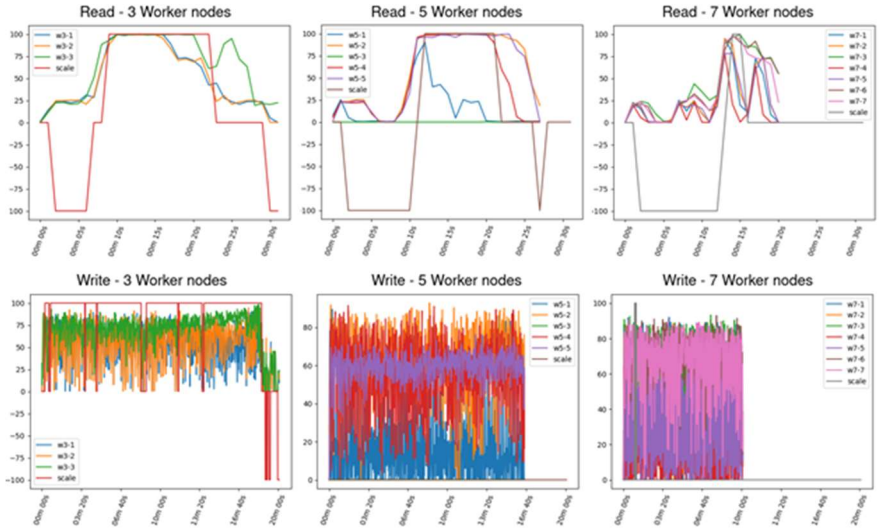

Abbildung 1: Skalierungs-Entscheidung. Basierend auf der Last wird für die Knoten entschieden, ob die Anzahl der Knoten der Datenbank ausreichend (Wert 0), upscaling (100) oder downscaling (-100) notwendig ist. Read / write für die verschiedenen Knoten sind auf die gleiche Zeitlänge angepasst.

Um die Ergebnisse hier zu demonstrieren, wurden exemplarisch die Datenbanken TimescaleDB (RDBMS), MongoDB (NoSQL) und CockroachDB (NewSQL) herangezogen, um jede der drei Kategorien zu inkludieren. Diese Datenbanken wurden dann mit variabler Cluster-Größe betrieben und verglichen, wobei Update- und Delete-Operationen aufgrund der geringeren Relevanz und langsamen Zugriffe ausgenommen wurden. Die Ressourcen-Auslastung in den Grafiken betrifft immer das gesamte Cluster – eine CPU-Auslastung von 100% würde also bedeuten, dass jeder einzelne Knoten im Cluster voll ausgelastet ist.
Den nachfolgenden Grafiken kann entnommen werden, dass TimescaleDB als Vertreter der relationalen Datenbanken mit diesem Aufbau schlecht bzw. gar nicht skaliert, da der sogenannte Access-Node das Bottleneck darstellt, und nur die damit verbundenen Worker-Nodes skaliert/hinzugefügt werden können, welche aber geringe Auslastung aufweisen: Zusätzliche Clients verbessern die Performance nicht. Im Vergleich zu den anderen Systemen ist es zudem nicht möglich, mehrere Access-Nodes hinzuzufügen – es verbleibt nur die vertikale Skalierung des Access-Node oder die Auslagerung der Tätigkeit des Access-Node in den Programmcode. Auf letzteres wurde aufgrund der mangelnden Vergleichbarkeit verzichtet.
Sowohl CockroachDB und MongoDB zeigen allerdings das, für horizontale Skalierung, übliche Verhalten: Mit zunehmender Cluster-Größe, steigert sich die Performanz der Schreib- und Lesezugriffe. Es wurde initial sichergestellt, dass der Cluster mit der niedrigsten Konfiguration während Schreibzugriffen eine ausreichend hohe Auslastung auf den Worker-Knoten aufweist, um subsequent die Vorteile der Skalierung zu zeigen.
Allgemein ist zu sagen, dass TimescaleDB zwar an sich sehr effizient ist, aber keine Performanceverbesserungen erreicht werden. Dies führt dazu, dass es die anderen Lösungen bei kleiner Clustergröße schlägt, aber durch fehlende Skalierung von MongoDB und CockroachDB eingeholt wird bzw. werden kann, wie die Grafiken es zeigen bzw. andeuten. Auch der oft erwähnte Skalierungs-Overhead ist den Grafiken unweigerlich zu entnehmen; obwohl sich die Cluster-Größe von 3 auf 7 mehr als verdoppelt (Faktor 2,33), erhöht sich die Leistung weder bei Read, noch bei Write, um denselben Faktor.

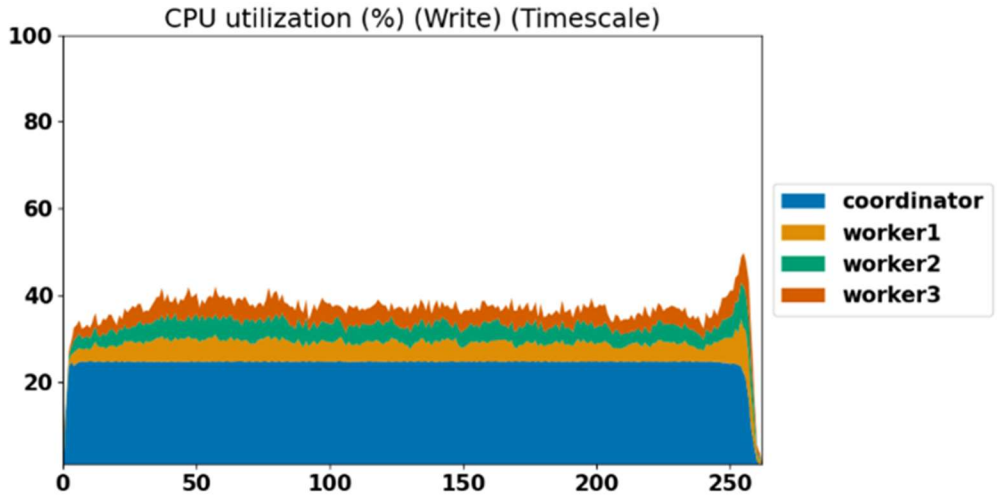

Abbildung 2: Prozessor-Auslastung-Write_Timescale

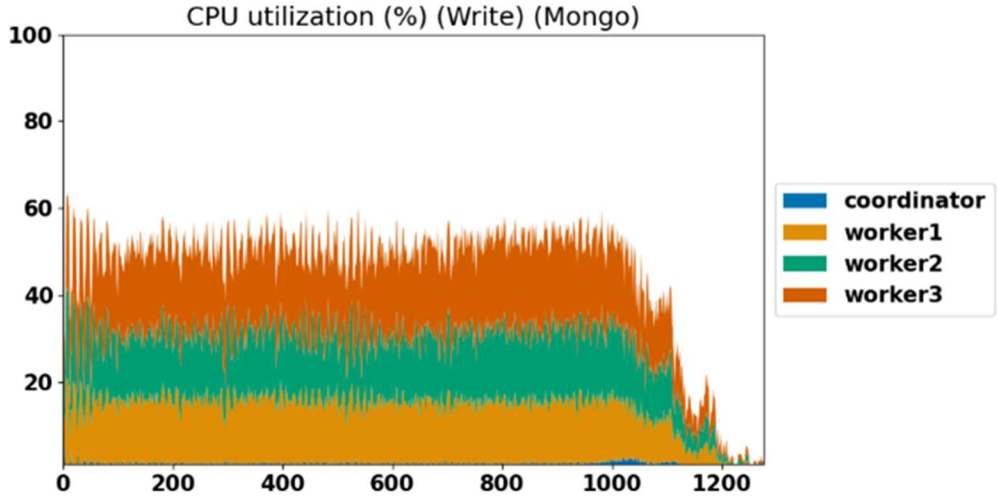

Abbildung 3: Prozessor-Auslastung-Write_Mongo

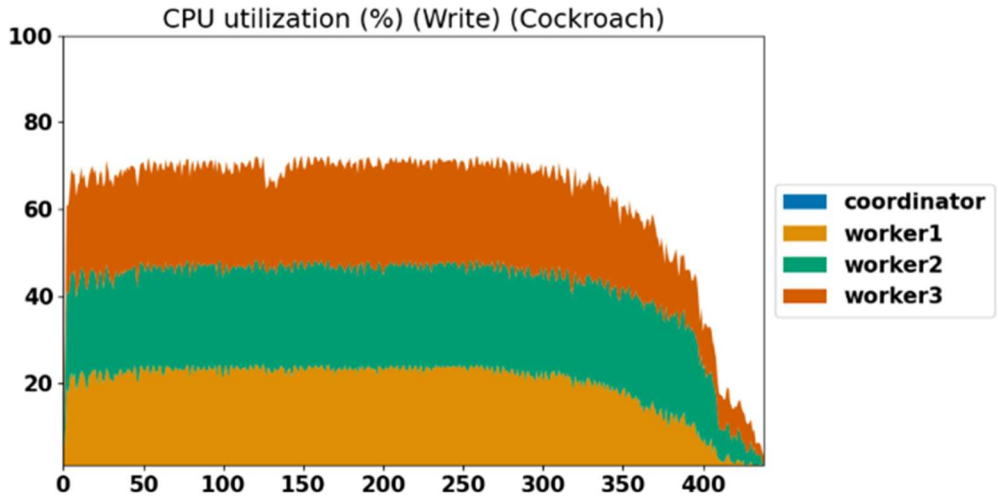

Abbildung 4: Prozessor-Auslastung-Write_Cockroach

Prozessor-Auslastung der Clusterknoten – Es ist zu erkennen, dass bei Timescale der Access-Node das Bottleneck ist. Sowohl MongoDB als auch CockroachDB weisen, abseits vom Coordinator, eine hohe Auslastung

pro Worker auf. Konkret sind es 55-70% Gesamauslastung, was einer Auslastung von 73%-95% pro Worker entspricht.

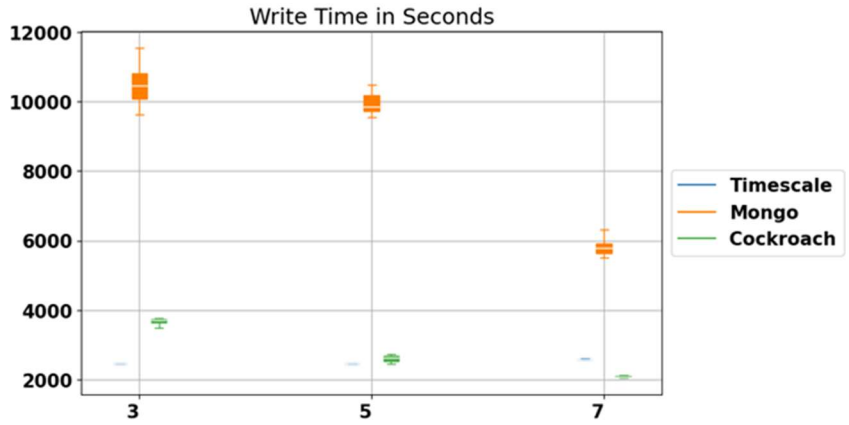

Abbildung 5: WriteTime . Die benötigte Zeit, um alle Daten im System zu persistieren. Je niedriger, desto besser.

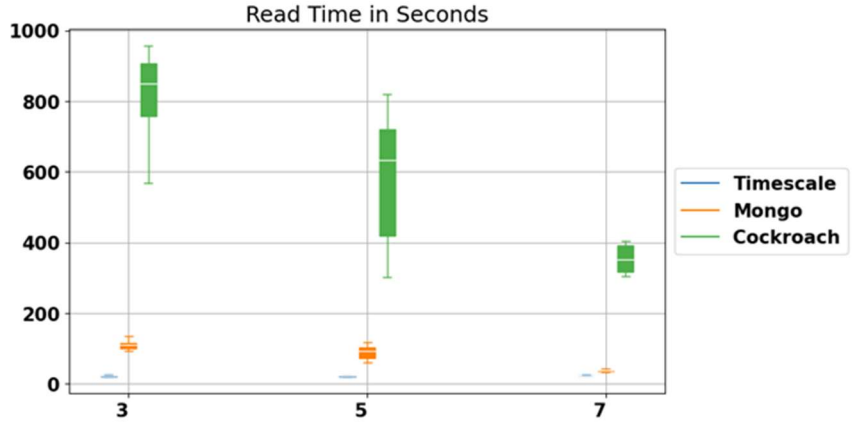

Abbildung 6: ReadTime. Die benötigte Zeit um 50% der geschriebenen Datenpunkte über ein SELECT Statement abzufragen. Je niedriger, desto besser.

Im Hinblick auf die Verarbeitungsschicht bestand ein wesentlicher Teil dieses Arbeitspakets darin, das TSM-Framework so zu portieren, dass es leichtere Bibliotheken im Kontext des .NET-Standards für webbasierte Anwendungen verwendet und die Bereitstellung des Dienstes durch orchestrierte Container-Instanzen ermöglicht. Darüber hinaus wurde eine Architektur für den Einsatz des HAKOM-Dienstes hinter einem Load Balancer implementiert, mit geeigneten Mechanismen, um die Anzahl der benötigten Instanzen entsprechend der Last im System zu definieren. Schließlich wurde die Architektur in mehreren Punkten verbessert, um eine effizientere Nutzung der Ressourcen zu ermöglichen und die Leistung auch in Szenarien mit nur einem Knoten zu steigern. Diese Verbesserungen sind Bestandteil des in Arbeitspaket 5 erwähnten Einsatzes im Zusammenhang mit dem PaaS-Angebot von HAKOM (POWER TSM).

Fazit

Es wurde eine Messuite entwickelt, um LeanXcale mit anderen Datenbanken vergleichbar zu machen, einschließlich Multi-Node-Messungen mit Cluster-Systemen. Dabei wurden spezielle Datentypen der Datenbanken berücksichtigt. Für die Erweiterungen wurden auch die AWS EC2 Instanzen erweitert, um ausreichend Last auf die Datenbanken ausüben zu können. Es wurde eine Skalierungsmethode entwickelt, die entscheidet, wann eine Skalierung aufwärts oder abwärts relevant wird, jedoch wurde

dieser Scaler noch nicht für die Datenbanken umgesetzt. Die Aufgabe der automatischen auslastungsabhängigen Skalierung wurde bisher nicht erfüllt.
Das Arbeitspaket ist (bzgl. Kosten) 100% abgeschlossen. Da die Skalierungsmessungen sich deutlich komplexer herausgestellt hat als geplant, war eine volle Umsetzung der Auslastungsabhängigen Skalierung basierend auf des Deciders umgesetzt.

3.4 Umsetzung einer Konfigurations- und Monitoringslösung

Die Arbeitspakete 2, 3 und 4 sind in der Implementierung an einigen Stellen eng verknüpft und lassen sich daher nicht vollständig getrennt betrachten.
Für die Verknüpfung der HAKOM-Plattform mit der Persistierungsschicht ist der HAKOM Connector von zentraler Bedeutung. Für diesen wurde (wie im Zwischenbericht angeführt) ein Mapping zwischen dem Connector und den darunterliegenden zu messenden Datenbanken erstellt. In relationalen Datenbanken wird dieses Mapping über Stored Procedures durchgeführt, die auf der jeweiligen Zieldatenbank direkt ausgeführt werden. Für Datenbanken wie LeanXcale, die zwar ein SQL-Interface anbieten aber keine Stored Procedures unterstützen, wurde der HAKOM Connector angepasst. Auch für das LeanXcale-KiVi-Interface, das für performantes Schreiben verwendet wird, wurde ein Mapping im HAKOM Connector implementiert. Diese zweistufige Form des HAKOM-Connectors ist also eine zwingende Voraussetzung, um bei HAKOM TSM auch NewSQL und NoSQL-Speicher unterstützen zu können, die nicht auf dem HAKOM Datenmodell aufsetzen, sondern dieses Mapping außerhalb der Persistierungs-Schicht anbieten müssen.
Die entwickelte Mess-Suite ermöglicht Messungen mit und ohne dieses Mapping, um für alle Anwendungsfälle objektive Vergleiche zu ermöglichen. Die Mess-Suite sitzt zwischen den HAKOM-Connector und der jeweiligen Zieldatenbank.
Um die Notwendigkeit zur Skalierung nachvollziehbar darstellen zu können, werden bei den Messungen neben dem Durchsatz und der Schreibgeschwindigkeit auch KPIs (Key Performance Indikatoren) wie CPU- und Memory-Verbrauch mit NetData erfasst. NetData wurde aus mehreren Tools, die evaluiert wurden, als beste Wahl festgelegt. Um Last unabhängig von echten Kundendaten von HAKOM in ausreichendem Ausmaß bereitstellen zu können, wurde auf Sensordaten, die von FH OÖ Campus Hagenberg in einem anderen Forschungsprojekt generiert werden, sowie auf künstlich generierte Daten, zurückgegriffen. Zeitserien verhalten sich unabhängig von ihrem Ursprung sehr ähnlich beim Schreiben und Aggregieren, daher ist die Anwendung dieser „neutralen“, DSGVO-konformen Daten zulässig. Um mit verschiedenen Datentypen testen zu können wurde die Mess-Suite so erweitert, dass beliebige Datenmodelle zum Vergleich zwischen den Datenbanken herangezogen werden können. Eine Konfiguration der Systemauslastung und -nutzung ist primär durch das Skalieren der EC-2 Instanzen möglich, in denen sich die Knoten der Datenbanken in dem Cluster einhängen (manche Datenbanken erlauben dies nur mit einem Neustart des Datenbankclusters). Die Skalierungsregeln werden automatisch in Form eines Deciders errechnet. Dieser kann jederzeit neu trainiert werden, und insbesondere bezüglich der Zielmetriken (DB Neustart ja/nein?, Zeit die ohne/unter Last ist, CPU Limits, …) rekonfiguriert werden. Wie bereits angemerkt wurde die automatische Skalierung nicht für die untersuchten Datenbank implementiert.
Fazit

Die entwickelte Mess-Suite ermöglicht das Monitoring des Datenflusses und der Datenqualität durch Messungen mit und ohne Mapping zwischen dem HAKOM Connector und den Zieldatenbanken. KPIs wie CPU- und Memory-Verbrauch werden während der Messungen erfasst, vom NetData Webinterface abgefragt und nachfolgend aufbereitet, um die Systemauslastung und -nutzung zu überwachen. Die Konfiguration der Metriken und Skalierungsregeln erfolgt automatisch durch einen Decider, der jedoch die automatische Skalierung noch nicht für die untersuchten Datenbanken implementiert hat.
Das Arbeitspaket ist 100% abgeschlossen, lediglich die Anwendung der Skalierungsregeln wird nicht unterstützt.

3.5 Pilotierung einer skalierbaren TSM-Lösung für Big Power Data

Durch die bereits umgesetzten Punkte in AP2, AP3 und AP4 wurden die Grundlagen geschaffen, die Konfigurations- und Monitoring-Möglichkeiten bei HAKOM TSM in Richtung Platform-as-a-Service (PaaS) bzw. eventuell zu Software-as-a-Service (SaaS) weiterzuentwickeln.
Die Integration der Unterschiedlichen Services fand in AWS auf EC2-Instancen statt. Es wurde durch die Wahl von AWS EC2 Instanzen bereits auf leichte Portierbarkeit und Vergleichbarkeit über Cloud- und on-premise-Plattformen hinweg Rücksicht genommen. Das Ziel war, dass die Artefakte des Projekts und die notwendigen Neuentwicklungen innerhalb von HAKOM TSM und POWER TSM nicht nur datenbankunabhängig, sondern auch unabhängig von Cloud-Anbietern sein sollten. Die Elemente, die sich auf die Skalierung der Rechen-/Zeitreihenverarbeitungsschicht in POWER TSM beziehen, wurden außerdem in der Microsoft Azure Cloud eingesetzt. Auch on-premise Anwendungsfälle sind durch diese Infrastrukturentscheidung umsetzbar. Im Detail müssen eventuell Anpassungen an die jeweilige Cloud- oder on-premise Plattform vorgenommen werden, die das initiale Aufsetzen der Services betrifft.
Es wurden umfangreiche Lasttests mit synthetischen Zeitseriendaten, sowie Echtdaten aus Onlinequellen durchgeführt. Um realistisches Verhalten zu simulieren, wurden zwischen einem und zehn Client-Systeme simuliert, die mit mehreren Millionen Datensätzen auf das DBMS zugreifen (aufgrund der Kosten wurde auf > 10 Clients verzichtet). Die Ergebnisse wurden in den Publikationen [35, 36] bereits veröffentlicht und auf internationalen Konferenzen von FH Hagenberg präsentiert, weiters befindet sich eine Publikation für das Lastverhalten bei mehreren Datenbankknoten derzeit in einem Journal in Veröffentlichung [55].
Darüber hinaus laufen derzeit Tests mit realen Kundendaten als Teil der Erprobung von HAKOMs POWER TSM-Cloud-Angebot. Weitere Entwicklungen innerhalb des Frameworks, die z. B. für die Handhabung von Multi-Tenant-Szenarien erforderlich sind, aber nicht in den Rahmen des BigPowerData-Projektes fallen, werden derzeit noch umgesetzt.

Fazit

Es wurde eine Mess-Suite entwickelt, deren Implementierung es ermöglicht, eine Vielzahl an DBMS in verschiedenen Konfigurationen zu testen und miteinander zu vergleichen. Die Verbindung der Cloudlösung AWS, mit dem Programmcode und den Konfigurationsdateien ermöglicht es, durch Änderung von spezifischen Einstellungen in diesen Dateien die Anzahl der Knoten im Cluster, sowie die Anzahl der schreibenden Clients zu steuern. Sämtliche Logik in Bezug auf Verwaltung (Starten, Herunterfahren, Überprüfen auf Erreichbarkeit) und Start der Datenbanklösung (Starten des Dienstes allgemein, sowie Konfigurieren des Clusters) ist automatisiert; einzig und allein das Anlegen neuer EC2

Instanzen (falls größere Cluster oder DBMS benötigt werden), Hinterlegen des SSH-Keys und das Installieren der notwendigen Datenbanken, wofür vorgefertigte Bash-Skripts erstellt wurden, sind manuell und einmalig zu erledigen. Im Rahmen der derzeitigen Benchmarks wurden bis zu 10 Clients und 8 Cluster-Knoten getestet (1 Access-Node + 7 Worker-Nodes).Eine Implementierung dieses Frameworks für produktive Zwecke als Teil von HAKOMs PaaS-Angebot ist derzeit im Entstehen und unterstreicht die Tragfähigkeit des gewählten Ansatzes.

4 Ergebnisse und Schlussfolgerungen

1. Die aktuelle Mess-Suite ermöglicht es, Last sowohl an die Applikationsebene (HAKOM TSM) anzulegen bzw. nur die Persistierungsschicht (Datenbanklayer) zu vermessen. Noch können nur LeanXcale, PostgreSQL und jede Datenbank, für die bereits ein HAKOM-Connector existiert, auf diesem Weg vermessen werden. Allerdings kann durch die Ergänzung um weitere HAKOM-Connectoren jedes beliebige Endsystem angebunden werden. Dafür steht eine Mess-Suite zur Verfügung, die über eine beliebige Anzahl an Knoten ausreichend Last erzeugen kann, dass Datenbanken in der definierten Konfiguration in die Skalierung gezwungen werden. Als Vergleichsdatenbanken für LeanXcale für Single-Node und für Cluster stehen unterschiedliche Datenbanken zur Verfügung, um den Skalierungsoverhead sowohl beim Schreiben als auch beim Lesen einordnen und bewerten zu können. Dazu gehören PostgreSQL mit der Skalierungslösung TimeScaleDB [45], InfluxDB [46], das am Single-Node das Ranking unangefochten anführt, MongoDB [47] in der aktuellen Version mit spezieller Zeitserienunterstützung, CrateDB [50], CockroachDB [51,53], DolphinDB [54] und YugaByteDB [52]. MongoDB wurde in einer Version mit spezieller Unterstützung von Zeitseriendaten über einen neuen Datentyp vermessen und in einer älteren Version, der diesen Datentypen noch nicht unterstützt hat.
2. Weitere vergleichende Messungen sowohl am Single-Node als auch mit horizontaler Skalierung wurden durchgeführt und lassen einen Vergleich betreffend der Ressourceneffizienz und des Skalierungsoverheads der einzelnen Datenbanken sowohl schreibend als auch lesend zu. Die Datenbanken wurden so gewählt, dass relational, NoSQL und NewSQL vertreten sind und so ein Vergleich der grundlegenden Skalierungsansätze mit ACID-Overhead in Bezug auf Ressourceneffizienz möglich ist.
3. Analog zu der Erweiterung des Datenbank-Setups auf mehrere Knoten (= Cluster) zur horizontalen Skalierung, wurde auch der Datenbankzugriff auf mehrere Clients aufgeteilt, die sich ebenfalls in der Cloud/AWS befinden. Durch den parallelen Zugriff wird zum einen ein realistischerer Use-Case zur Verfügung gestellt (mehrere Benutzer greifen gleichzeitig zu) und zum anderen Ressourcen lastige Stresstests ermöglicht, die für die COST-Metrik notwendig sind.
4. Die Vermessung von Aggregierungen wurde verbessert, sodass Aggregierung, Read und Write getrennt ausgewiesen wird. Die Mess-Suite ist damit aktuell noch besser ausgelegt, auch hier unterschiedliche Ansätze und Konfigurationen vergleichen zu können.
5. Die Einbindung von anderen Datenformaten als dem von HAKOM bereits vorgegebenen Datenformat für den Benchmark wurde ermöglicht. Damit kann die Benchmark Suite auch für das Vermessen von Zeitserien allgemein, und für andere Anwendungsfälle, verwendet werden. Da

die Möglichkeit angedacht wird, das Projekt in großen Teilen als Open-Source zu stellen, ist das eine wichtige Erweiterung, damit Kunden die Ergebnisse bei Bedarf mit eigenen Daten selbst nachvollziehen können.

Probleme

1. Probleme gab es mit dem speziellen HAKOM Connector für LeanXcale, der noch nicht in vollem Funktionsumfang zur Verfügung stand.
2. Aufgrund von Einschränkungen hinsichtlich der Schreiboperationen, sowie von Instabilität bei zu großen Datenmengen wurde LeanXcale ohne horizontale Skalierung, also nur als Single-Node vermessen. Durch die Einschränkungen war es nicht möglich einen sinnvollen Vergleich mit anderen Clustern durchzuführen. Es ist zudem anzumerken, dass, aufgrund der erwähnten Probleme, selbst für die Single-Node Konfiguration nur Ergebnisse für die proprietäre KiVi Schnittstelle, und ohne parallelen Zugriff vorliegen [36].

5 Ausblick und Empfehlungen

1. Im Projekt konnte festgestellt werden, dass eine einzelne Thread-Datenbankinstanz für eine Reihe von Anwendungsfällen bereits sehr große Aufnahmevolumina bewältigen und angemessene Lesegeschwindigkeiten liefern kann. Dennoch gibt es interessante Open-Source-Alternativen und die Entwicklung ACID-konformer horizontal skalierender Datenbanktechnologien bieten viel Potenzial. Diese sind jedoch mit zahlreichen Herausforderungen in der Implementierung verbunden. Die Schwierigkeiten, mit denen LeanXcale bei der Portierung seiner Datenbank-API auf die .NET-Umgebung konfrontiert war, lassen darauf schließen, dass diese alles andere als trivial sind.
2. Die Leistungsanforderungen und die berechneten Benchmarks hängen sehr stark von der Datenverteilung und dem Verhältnis von Schreiben- zu Lesevorgängen ab und sind daher sehr anwendungsfallspezifisch. Es wird empfohlen, das Benchmark-Framework zu nutzen und es an die spezifischen Anforderungen bestimmten Projekts anzupassen, bevor man sich auf eine bestimmte Wahl einer Datenbankkonfiguration festlegt.
3. Das Benchmark-Framework würde von einer leichter konfigurierbaren GUI profitieren.

6 Literaturverzeichnis

1. Andreas Theissler. Detecting anomalies in multivariate time series from automotive systems. PhD thesis, Brunel University School of Engineering and Design PhD Theses, 2013. (kein Datum).
2. Jean-Marc Bardet, Faure Cynthia, Jérôme Lacaille, and Madalina Olteanu. Unequal time series clustering applied on flight data. (kein Datum).
3. Chad A Steed, William Halsey, Ryan Dehoff, Sean L Yoder, Vincent Paquit, and Sarah Powers. Falcon: Visual analysis of large, irregularly sampled, and multivariate time series data in additive manufacturing. Computers & Graphics, 63:50–64, 2017.
4. Samuel Da Silva, Milton Dias Junior, and Vicente Lopes Junior. Structural health monitoring in smart structures through time series analysis. Structural Health Monitoring, 7(3):231–244, 2008.

5. Tomohiko Inui, Hiroki Kohno, Yohei Kawasaki, Kaoru Matsuura, Hideki Ueda, Yusaku Tamura, Michiko Watanabe, Yuichi Inage, Yasunori Yakita, Yutaka Wakabayashi, et al. Use of a smart watch for early detection of paroxysmal atrial fibrillation: validation study. JMIR cardio, 4(1):e14857, 2020.
6. Alyson S Barratt, Karl M Rich, Jude I Eze, Thibaud Porphyre, George JGunn, and Alistair W Stott. Framework for estimating indirect costs in animal health using time series analysis. Frontiers in veterinary science,6:190, 2019.
7. Davis Blalock, Samuel Madden, and John Guttag. Sprintz: Time series compression for the internet of things. Proceedings of the ACM onInteractive, Mobile, Wearable and Ubiquitous Technologies, 2(3):1–23,2018.
8. Wim Van der Ham, Michel Klein, Seyed Amin Tabatabaei, Dilhan JThilakarathne, and Jan Treur. Methods for a smart thermostat to estimatethe characteristics of a house based on sensor data. Energy Procedia,95:467–474, 2016.
9. Yunsun Kim and Sahm Kim. Forecasting charging demand of electric vehicles using time-series models. Energies, 14(5):1487, 2021.
10. Sidra Mehtab and Jaydip Sen. A time series analysis-based stock priceprediction using machine learning and deep learning models. arXivpreprint arXiv:2004.11697, 2020.
11. José Portela González, Antonio Munoz San Munoz San Roque, andEstrella Alonso Perez. Forecasting functional time series with a newhilbertian armax model: Application to electricity price forecasting.IEEE Transactions on Power Systems, 33(1):545–556, 2017.2 https://www.citusdata.com/
12. Gianluca Bontempi, Souhaib Ben Taieb, and Yann-A¨el Le Borgne. Machine learning strategies for time series forecasting. In European business intelligence summer school, pages 62–77. Springer, 2012.
13. Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov,Jason Lines, and Franz J Kir´aly. sktime: A unified interface for machinelearning with time series. arXiv preprint arXiv:1909.07872, 2019.
14. Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guil-laume Androz, Chester Holtz, Marie Payne, Roman Yurchak, MarcRußwurm, Kushal Kolar, et al. Tslearn, a machine learning toolkit fortime series data. J. Mach. Learn. Res., 21(118):1–6, 2020.
15. Artur Romazanov and Irina Zakharova. Clustering of indoor temperature time series. In 2021 International Conference on Electrical, Computer,Communications and Mechatronics Engineering (ICECCME), pages 1–4, 2021.
16. Alina Barbulescu, Cristian Stefan Dumitriu, and Florentina-LoredanaDragomir. Detecting aberrant values and their influence on the time se-ries forecast. In 2021 International Conference on Electrical, Computer,Communications and Mechatronics Engineering (ICECCME), pages 1–5, 2021.
17. Sergio Di Martino, Luca Fiadone, Adriano Peron, Alberto Riccabone,and Vincenzo Norman Vitale. Industrial internet of things: persistencefor time series with nosql databases. In 2019 IEEE 28th InternationalConference on Enabling Technologies: Infrastructure for CollaborativeEnterprises (WETICE), pages 340–345. IEEE, 2019.
18. Patrick Valduriez, Ricardo Jimenez-Peris, and M. Tamer Ozsu. Dis-tributed database systems: The case for newsql. In Transactions onLarge-Scale Data-and Knowledge-Centered Systems XLVIII, pages 1–15. Springer, 2021.
19. Clinton Gormley and Zachary Tong. Elasticsearch: the definitive guide:a distributed real-time search and analytics engine. O'Reilly Media,Inc., 2015.
20. Josiah Carlson. Redis in action. Simon and Schuster, 2013.
21. Jim Webber. A programmatic introduction to neo4j. In Proceedings ofthe 3rd annual conference on zystems, programming, and applications:software for humanity, pages 217–218, 2012.

22. Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi.Time series databases and influxdb. Studienarbeit, Université Libre deBruxelles, 12, 2017.
23. Matthew Aslett. What we talk about when we talk about newsql. 6Travanj, 2011.
24. Frank McSherry, Michael Isard, and Derek G. Murray. Scalability! But at what COST? In 15th Workshop on Hot Topics in Operating Systems (HotOS XV), Kartause Ittingen, Switzerland, May 2015. USENIXAssociation.
25. Charles W. Bachman. The programmer as navigator. Commun. ACM,16(11):653–658, nov 1973.
26. Edgar Frank Codd. A relational model of data for large shared databanks. Communications of the ACM, 26(1):64–69, 1983.
27. Sandra L Emerson, Marcy Darnovsky, and Judith Bowman. Thepractical SQL handbook: using structured query language. Addison-Wesley Longman Publishing Co., Inc., 1989.
28. Theo Haerder and Andreas Reuter. Principles of transaction-orienteddatabase recovery. ACM computing surveys (CSUR), 15(4):287–317,1983.
29. Jing Han, Ee Haihong, Guan Le, and Jian Du. Survey on nosqldatabase. In 2011 6th international conference on pervasive computingand applications, pages 363–366. IEEE, 2011.
30. Jaroslav Pokorny. Nosql databases: a step to database scalability in webenvironment. International Journal of Web Information Systems, 2013.
31. Neal Leavitt. Will nosql databases live up to their promise? Computer, 43(2):12–14, 2010.
32. Innocent Mapanga and Prudence Kadebu. Database management systems: A nosql analysis. International Journal of Modern CommunicationTechnologies and Research, 1(7):265849, 2013.
33. Yuanzhe Hao, Xiongpai Qin, Yueguo Chen, Yaru Li, Xiaoguang Sun,Yu Tao, Xiao Zhang, and Xiaoyong Du. Ts-benchmark: A benchmarkfor time series databases. In 2021 IEEE 37th International Conferenceon Data Engineering (ICDE), pages 588–599. IEEE, 2021.
34. Rui Liu and Jun Yuan. Benchmarking time series databases with iotdb-benchmark for iot scenarios. arXiv preprint arXiv:1901.08304, 2019.
35. Christoph Praschl, Sebastian Pritz, Oliver Kraus, Martin Harrer, A Comparison Of Relational, NoSQL and NewSQL Database Management Systems For The Persistence Of Time Series Data, ICECCME 2022
36. Sebastian Pritz, Martina Zeinzinger, Christoph Praschl, Oliver Kraus, Martin Harrer, ID 948 Performance Impact of Parallel Access of Time Series in the Context of Relational, NoSQL and NewSQL Database Management Systems, ICECCME 2023
37. G. Meen, “The time-series behavior of house prices: a transatlantic divide?” Journal of housing economics, vol. 11, no. 1, pp. 1–23, 2002.
38. J. D. Cryer and N. Kellet, Time series analysis. Springer, 1991.
39. A. Moniruzzaman and S. A. Hossain, “Nosql database: New era of databases for big data analytics-classification, characteristics and comparison,” arXiv preprint arXiv:1307.0191, 2013.
40. K. Chooruang and K. Meekul, “Design of an iot energy monitoring system,” in 2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE). IEEE, 2018, pp. 1–4.
41. G. La Tona, M. Luna, A. Di Piazza, and M. C. Di Piazza, “Towards the real-world deployment of a smart home ems: A dp implementation on the raspberry pi,” Applied Sciences, vol. 9, no. 10, p. 2120, 2019.
42. A. Pavlo and M. Aslett, “What’s really new with newsql?” ACM Sigmod Record, vol. 45, no. 2, pp. 45–55, 2016.
43. D. G. Chandra, “Base analysis of nosql database,” Future Generation Computer Systems, vol. 52, pp. 13–21, 2015.
44. T. P. G. D. Group. Postgresql. [Online]. Available: https://www.postgresql.org/
45. Timescale. Timescaledb. [Online]. Available: https://www.timescale.com/

46. InfluxData. Influxdb. [Online]. Available: https://www.influxdata.com/
47. MongoDB. Mongodb. [Online]. Available: https://www.mongodb.com/
48. LeanXcale. Leanxcale. [Online]. Available: https://www.leanxcale.com/
49. DolphinDB. Dolphindb. [Online]. Available: https://www.dolphindb.com
50. Crate.io. Cratedb. [Online]. Available: https://crate.io
51. Cockroach Labs. Cockroachdb. [Online]. Available: https://www.cockroachlabs.com/
52. YUGABYTE. Yugabytedb. [Online]. Available: https://www.yugabyte.com/
53. Cockroach Labs. Time-travel queries. [Online]. Available: https://www.cockroachlabs.com/docs/stable/as-of-system-time.html
54. DolphinDB. Dolphindb - setatomiclevel. [Online].Available: https://www.dolphindb.com/help/FunctionsandCommands/CommandsReferences/s/setAtomicLevel.html
55. Christoph Praschl, Sebastian Pritz, Martina Zeinzinger, Benito Lo Bue, Oliver Kraus, Martin Harrer, Distributed Time Series! But at what COST?, [Submitted to Software MDPI]

7 Anhang

8 Kontaktdaten

ProjektleiterIn DI Stefan Komornyik
Institut/Unternehmen HAKOM Time Series GmbH
Kontaktadresse (Adresse, Tel/Fax, e-mail; Webpage des Instituts/Unternehmen; Webpage des gegenständlichen Projekts, falls vorhanden) Lemböckgasse 61, 1230 Wien, Österreich, Home - HAKOM Power TSM
Auflistung der weiteren Projekt- bzw. KooperationspartnerInnen Name / Institut oder Unternehmen
Christoph Praschl, Oliver Krauss FH Hagenberg
Ricardo Jimenez-Peres, LeanXScale