Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Vorlesung 12 – NoSQL DBs

Foto: Quirin Leppert

Themenübersicht

Data Integration

- Data Formats (csv, XML, json)
- Extract, Transform, Load
- Object Relation Mapper (ORM)
- Staging

Data Processing

- Relationale Datenbanken
- **nicht-relationale Datenbanken**
- Resource Description Framework (RDF)
- Ontologien
- Data Warehouse

Data Modelling

- Serialisierung
- OPC UA
- MQTT
- Pub/Sub
- Data pipelines
  - Apache Airflow
  - gRPC

Web-Service Architektur

- Front-End
- Backend for Frontend (BFF)
- Micro Services
- Docker Container

Security

- Security ist wichtig in allen Phasen der Softwareentwicklung und Datenbereitstellung.

Übung

- Erstellung eines Daten-Modells einer prozesstechnischen Anlage
- Statische Daten
- Dynamische Daten
- Auswertung der Daten

NoSQL DBs

NoSQL DBs

- Bereits behandelt in DSCB240 - VL3 - NoSQL-Datenbanken
- NoSQL (englisch für Not only SQL deutsch: „Nicht nur SQL“) bezeichnet Datenbanken, die einen nicht-relationalen Ansatz verfolgen und damit mit der langen Geschichte relationaler Datenbanken brechen. Diese Datenspeicher benötigen keine festgelegten Tabellenschemata und versuchen Joins zu vermeiden. Sie skalieren dabei horizontal. Im akademischen Umfeld werden sie häufig als „strukturierte Datenspeicher“ (engl. structured storage) bezeichnet.

Quelle: https://de.wikipedia.org/wiki/NoSQL

NoSQL DBs

Anwendungsbeispiele

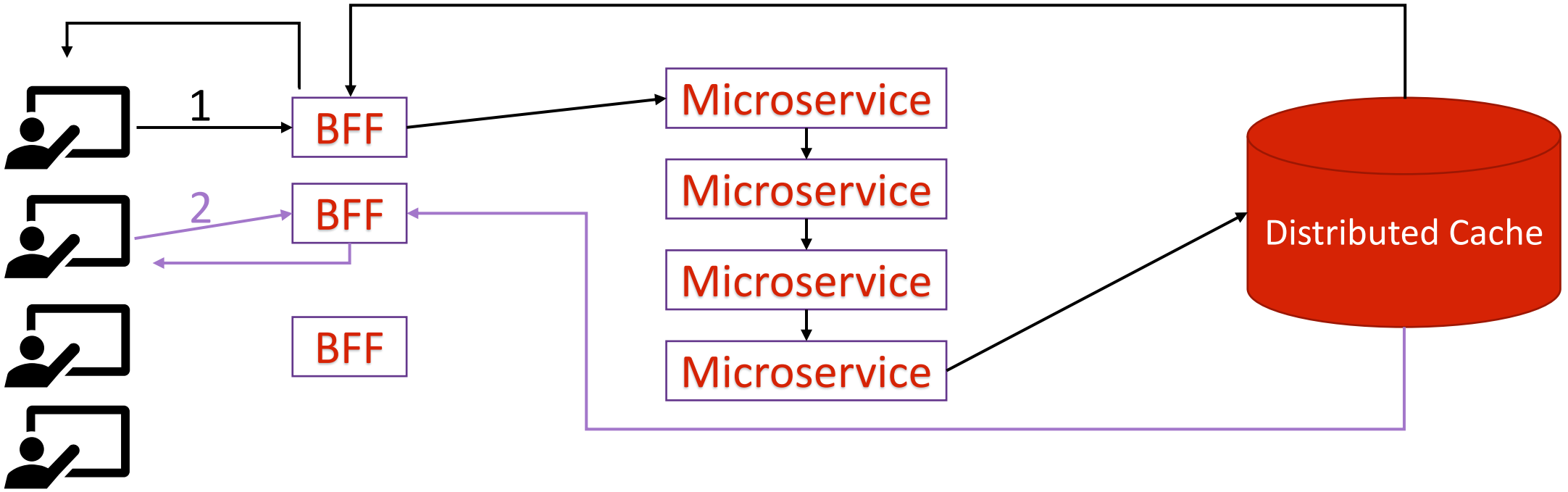
+ Nutzung als Distributed Cache für Web-Services
- Vergleich MSSQL und memurai (redis für Windows)
+ NoSQL-Datenbank als Timeseries-Datenbank
- Beispiel influxDB

Quelle: https://de.wikipedia.org/wiki/NoSQL

NoSQL DBs
Distributed Cache für Web-Services

Architektur

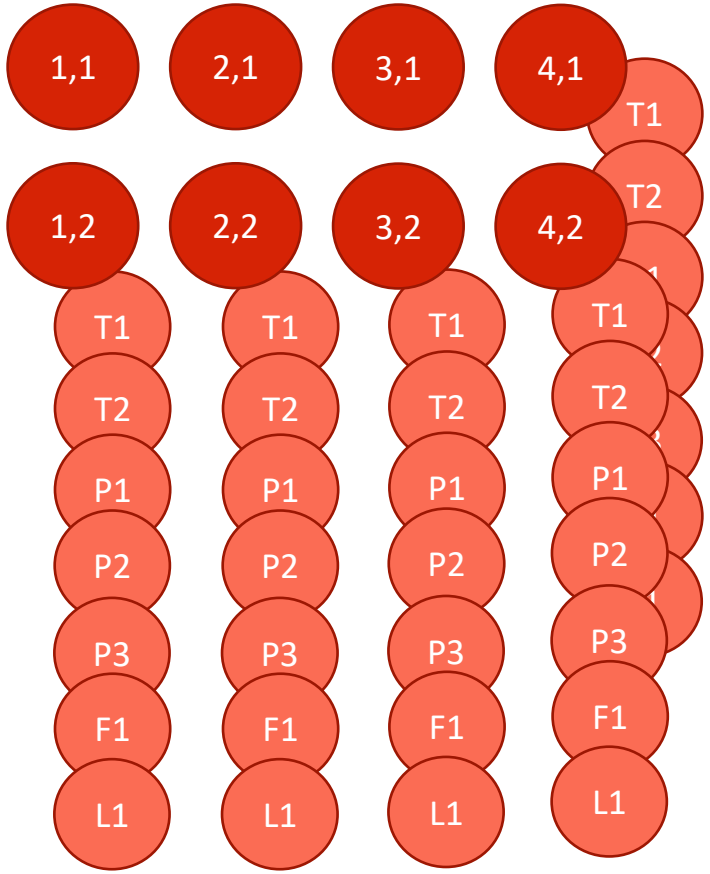
- Beispiel: komplexe Berechnung angestoßen durch einen Nutzer wird im Distributed Cache gespeichert und kann anschließend allen Nutzern auf Anfrage sofort zur Verfügung gestellt werden.



NoSQL DBs
Distributed Cache für Web-Services

Beispiel: self-organizing map (SOM) (1)

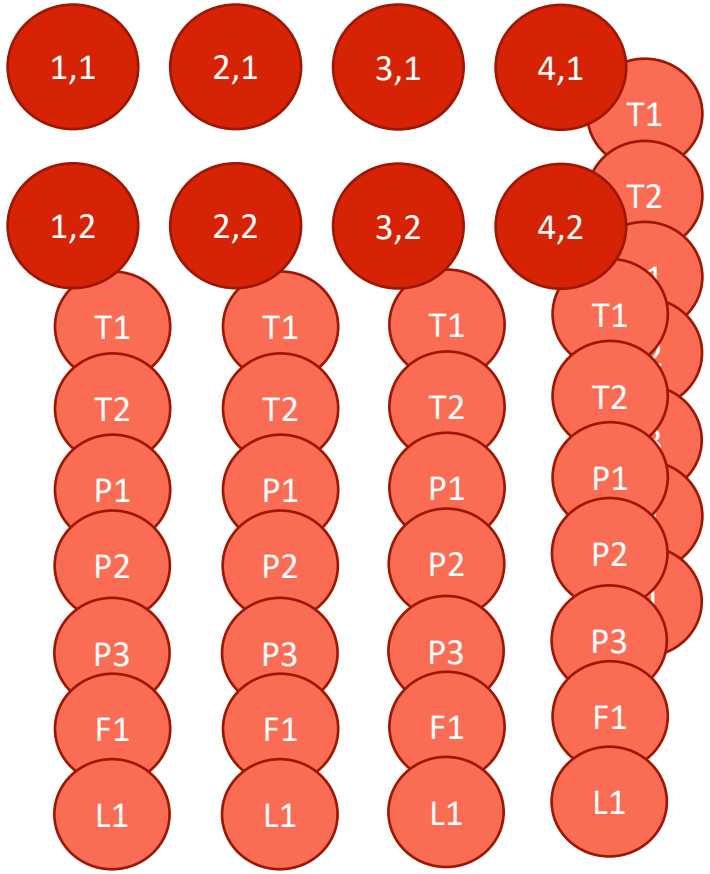
+ Verfahren zum unsupervised Clustering von Daten (Messwerten)
+ Nutzung z.B. zur Anomalieerkennung
+ Eingangsvektor z.B. Temperatur 1 (T1), Temperatur 2, Druck 1 (P1), Druck 2, Druck 3, Durchfluss 1 (F1), Füllstand 1 (L1), …
+ Eine selbst-organisierende Karte besteht aus (x,y)-Neuronen
+ An jedem Neuron hängt ein Vektor mit repräsentativen Werten (Gewichte) für den jeweiligen Messwert
+ Beim Training werden
- alle Gewichte mit Zufallszahlen initialisiert,
- alle Trainingsdaten viele tausend Mal mit der SOM verglichen; das Neuron mit dem jeweils geringsten euklidischem Abstand ermittelt; dieses Neuron und seine umgebenden Neuronen an den Trainingsdaten-Vektor angepasst
- im Laufe des Trainings die Umgebung und die Stärke der Anpassung verringert



NoSQL DBs
Distributed Cache für Web-Services

Beispiel: self-organizing map (SOM) (2)

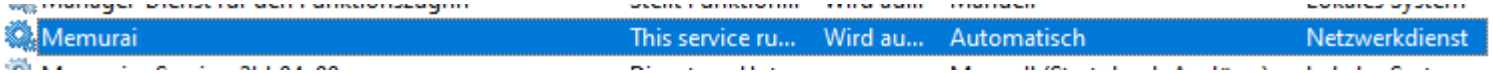
- Speicherung der SOM in einer relationalen Datenbank aufwendig, wenn jeder Wert in einem Tabellenfeld gespeichert und die 3-Dimensionalität abgebildet wird.
- Nutzung vieler Fremdschlüssel erforderlich
- Einfache Lösung: Serialisierung der SOM im Binärformat und Speicherung als Key-Value-Pair.
- Größe der Karte je nach Anwendung bis zu 1000 Neuronen; bei z.B. 100 Messwerten folgen daraus 100.000 Gewichte



NoSQL DBs
Distributed Cache für Web-Services

Installation von memurai

- kompatibel zu Redis 5.0.14
- Download https://www.memurai.com/get-memurai → „Free Download“
- Ausführen der Datei Memurai-Developer-v2.0.5.msi
- Installation als Windows Service



- Nutzung von redis-py zur Kommunikation mit der Datenbank

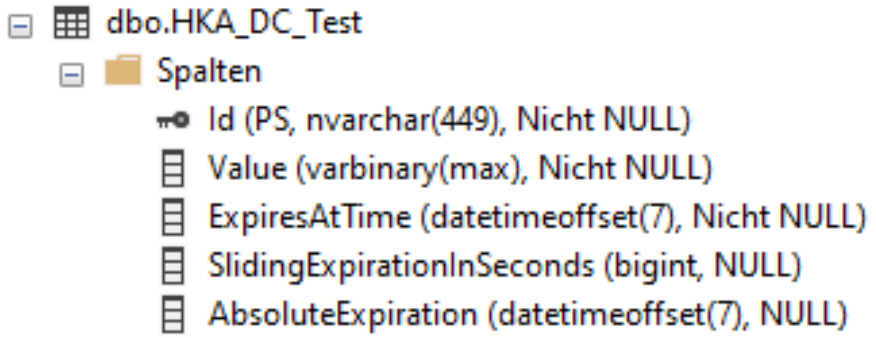
Quelle: https://www.memurai.com/get-memurai

NoSQL DBs
Distributed Cache für Web-Services

Übung 30-distributed-cache

- Nutzung von Redis als Distributed Cache
- Nutzung von MSSQL-Server als Distributed Cache
- Vergleich von Redis und MSSQL bezüglich Performance
- 01-write-read-redis.py: Testen der Verbindung zur Memurai-DB
- 02-SQL-Create-Distributed-Cache.sql: Anlegen einer Tabelle für einen Distributed Cache
- 03-compare-redis-SQL-small-chunks.py: Performance-Test mit Schreiben und Lesen kurzer Streams
- 04-compare-redis-SQL-large-chunks.py: Performance-Test mit Schreiben und Lesen großer Streams

- Die Tabelle für den Distributed Cache des MSSQL-Servers enthält eine Spalte Id für den Schlüssel (Key), eine Spalte für den Value und Spalten für die Dauer der Speicherung. Nach Ablauf der Speicherzeit werden die entsprechenden Zeilen im Cache gelöscht.



Quelle: https://www.memurai.com/get-memurai; https://github.com/redis/redis-py; https://docs.microsoft.com/en-us/aspnet/core/performance/caching/distributed?view=aspnetcore-6.0

NoSQL DBs
Time series DB - InfluxDB

Beispiel InfluxDB

+ InfluxDB ist ein Open Source Datenbankmanagementsystem (DBMS), speziell für Zeitreihen (engl. time series). Es wird von der Firma InfluxData entwickelt und vertrieben.
+ Warum Nutzung der InfluxDB
- Einfache Installation unter Windows
- #1 at DB-Engines – Ranking nach Popularität

Quelle: https://de.wikipedia.org/wiki/InfluxDB; https://db-engines.com/en/ranking/time+series+dbms

NoSQL DBs
Time series DB - InfluxDB

Installation InfluxDB

+ Download der Datei influxdb2-2.1.1-windows-amd64.zip unter https://docs.influxdata.com/influxdb/v2.1/install/?t=Windows
+ Extrahieren der Dateien
+ Als Administrator: Verschieben der Dateien in den Ordner C:\Program Files\InfluxData\influxdb
  In diesem Ordner befinden sich drei Dateien: influxd.exe u.a.
+ Start der Datenbank z.B. aus der Powershell
  Ausführen der Datei influxd.exe
+ Einrichten der Datenbank
- Username und Passwort setzen und merken!
- Organization: HKA
- Bucket: DataEngineering

Set up InfluxDB through the UI

1. With InfluxDB running, visit localhost:8086.
2. Click **Get Started**

Set up your initial user

1. Enter a **Username** for your initial user.
2. Enter a **Password** and **Confirm Password** for your user.
3. Enter your initial **Organization Name**.
4. Enter your initial **Bucket Name**.
5. Click **Continue**.

+ Abrufen des Tokens:
- "Load Data" → "API Tokens", click on <USER> token and copy your token

Quelle: https://docs.influxdata.com/influxdb/v2.1/install/?t=Windows;

NoSQL DBs
Time series DB - InfluxDB

+ Daten sind organisiert in Buckets

Data elements

+ Timestamp: _time column, epoch nanosecond
+ Measurement: _measurement column, measurement name that describes the data
+ Fields bestehend aus
- _field column: key, name of the field, Datentyp string
- _value column: Wert, Datentyp string, float, integer, boolean
- Fields werden nicht indiziert
+ Field set
- A field set is a collection of field key-value pairs associated with a timestamp.
+ Tags (Key und Value) (optional)
- Tags are indexed
- Beschreibung der Daten
- Beispiel: Abfrage, wann 23 Bienen gezählt wurden:

```
from(bucket: "bucket-name")
|> range(start: 2019-08-17T00:00:00Z, stop: 2019-08-19T00:00:00Z)
|> filter(fn: (r) => r._field == "bees" and r._value == 23)
```

_time	_measurement	location	scientist	_field	_value
2019-08-18T00:00:00Z	census	klamath	anderson	bees	23
2019-08-18T00:00:00Z	census	portland	mullen	ants	30
2019-08-18T00:06:00Z	census	klamath	anderson	bees	28
2019-08-18T00:06:00Z	census	portland	mullen	ants	32

+ Series:
- Series key: implizit gebildet aus measurement, tag set, and field key
- A **series** includes timestamps and field values for a given series key
+ A **point** includes the series key, a field value, and a timestamp

Quelle: https://docs.influxdata.com/influxdb/v2.1/reference/key-concepts/; https://docs.influxdata.com/influxdb/v2.1/reference/key-concepts/data-elements/

NoSQL DBs
Time series DB - InfluxDB

Design Principles

+ Time-ordered data
- To improve performance, data is written in time-ascending order.
+ Strict update and delete permissions
- To increase query and write performance, InfluxDB tightly restricts **update** and **delete** permissions. Time series data is predominantly new data that is never updated. Deletes generally only affect data that isn't being written to, and contentious updates never occur.
+ Handle read and write queries first
- InfluxDB prioritizes read and write requests over strong consistency. InfluxDB returns results when a query is executed. Any transactions that affect the queried data are processed subsequently to ensure that data is eventually consistent. Therefore, if the ingest rate is high (multiple writes per ms), query results may not include the most recent data.
+ Schemaless design
- InfluxDB uses a schemaless design to better manage discontinuous data. Time series data are often ephemeral, meaning the data appears for a few hours and then goes away. For example, a new host that gets started and reports for a while and then gets shut down.
+ Datasets over individual points
- Because the data set is more important than an individual point, InfluxDB implements powerful tools to aggregate data and handle large data sets. Points are differentiated by timestamp and series, so don't have IDs in the traditional sense.
+ Duplicate data
- To simplify conflict resolution and increase write performance, InfluxDB assumes data sent multiple times is duplicate data. Identical points aren't stored twice. If a new field value is submitted for a point, InfluxDB updates the point with the most recent field value. In rare circumstances, data may be overwritten. Learn more about duplicate points.

Quelle: https://docs.influxdata.com/influxdb/v2.1/reference/key-concepts/design-principles/

NoSQL DBs
Time series DB - InfluxDB

Übung mit InfluxDB

- 01-write-to-influxDB.py: Schreiben eines Werts in die Datenbank
- 02-query-data-from-influxDB.py: Lesen eines Werts aus der Datenbank
- 03-query-data-MSSQL.py: Lesen aller Zeitreihen aus einer SQL-Datenbank zur Demonstration des Datenumfangs
  Backup der MSSQL-Datenbank in der Datei HKA_SmA.bak,
  siehe https://bwsyncandshare.kit.edu/apps/files/?dir=/HKA-DataEngineering/Databases&fileid=1727967835
- 04-ETL-MSSQL-to-influxDB-iterator.py bzw. 04-ETL-MSSQL-to-influxDB.py:
  Überführen von Daten aus der MSSQL-Datenbank in die InfluxDB.
- 05-query-max-data-from-influxDB.py: Auslesen der Maximal-Werte aus der InfluxDB
- 06-query-max-data-from-MSSQL.py: Auslesen der Maximal-Werte aus der InfluxDB
- 07-influxDB-to-pandas.py: Auslesen der Maximal-Werte aus der InfluxDB

Quelle: https://docs.influxdata.com/influxdb/v2.1/api-guide/client-libraries/python/;

Hochschule Karlsruhe
University of
Applied Sciences

www.h-ka.de