Performanzanalyse von Key-Value-Datenspeichern

Prüfer: Prof. Dr.-Ing. Bernhard Mitschang

Betreuer: M.Sc. Dennis Przytarski

Kei Wai Lam
st159708@stud.
uni-stuttgart.de

Alexander Schäfer
st162498@stud.
uni-stuttgart.de

Sharon-Naemi Stiliadou
st154880@stud.
uni-stuttgart.de

Anwendungssoftware

13.08.2021

Agenda

MOTIVATION

VERWANDTE ARBEITEN

ERGEBNISSE

AUSBLICK

Agenda

MOTIVATION

VERWANDTE ARBEITEN

ERGEBNISSE

AUSBLICK

SQL vs. NoSQL

	SQL	NoSQL
Entwickelt	1970	ca. 2000
Schema	statisch/vordefiniert	flexibel
Modell	tabellenbasiert	dokumentenorientiert, graphbasiert, spaltenorientiert, Key-Value
Skalierbarkeit	vertikal	horizontal
Eigenschaften	ACID	BASE

[1]

SQL vs. NoSQL

	SQL	NoSQL
Entwickelt	1970	ca. 2000
Schema	statisch/vordefiniert	flexibel
Modell	tabellenbasiert	dokumentenorientiert, graphbasiert, spaltenorientiert, Key-Value
Skalierbarkeit	vertikal	horizontal
Eigenschaften	ACID	BASE

[1]

Was ist eine Key-Value Datenbank?

- Datenmodell: (*Key, Value*)-Paare
- Values werden über einen Key identifiziert
- Key ist eindeutig
- Values können einfach oder komplex sein z.B. Integer oder Array

Persistente vs. Nicht persistente Datenbank

Persistent	Nicht persistent
Die Eigenschaft Daten über die Laufzeit eines Programms oder Systems zu speichern. *Beispiel:* Die Daten werden auf der Festplatte gespeichert.	Die Eigenschaft, dass Daten nur über die Laufzeit eines Programms bestehen bleiben. *Beispiel:* In-memory Datenbanken

Persistente vs. Nicht persistente Datenbank

Persistent	Nicht persistent
Die Eigenschaft Daten über die Laufzeit eines Programms oder Systems zu speichern. *Beispiel:* Die Daten werden auf der Festplatte gespeichert.	Die Eigenschaft, dass Daten nur über die Laufzeit eines Programms bestehen bleiben. *Beispiel:* In-memory Datenbanken

Aufgabenstellung

Performanz von verschiedenen persistenten Key-Value-Datenspeichern anhand von synthetischen Testdaten analysieren

Aufgabenstellung

Performanz von verschiedenen persistenten Key-Value-Datenspeichern anhand von synthetischen Testdaten analysieren

1) Recherche zu bestehenden Key-Value-Datenspeichern
2) Eingruppierung in die jeweiligen Implementierungsstrategien
3) Kategorisierung der Key-Value-Datenspeicher anhand ihrer Eigenschaften
4) Generierung von synthetischen Testdaten mit Hashwerten als Schlüssel und Byte-Arrays mit Festlänge als Wert
5) Durchführung der Performanzanalysen für ausgesuchte Key-Value-Datenspeicher je Implementierungsstrategie

Agenda

MOTIVATION

VERWANDTE ARBEITEN

ERGEBNISSE

AUSBLICK

Vergleich von Benchmarks

Benchmark	Fokus	Letztes Update	Funktionen	Vorteile	Nachteile
InfluxDB Benchmark [2]	• Dokumenten-orientiert	• Juli 2021	• *Write (key, value)* • *Get (key)* • *Delete (key)*	• Vergleicht LevelDB • Delete	• Keine weiteren Key-Value Datenbanken
RocksDB Benchmark [3]	• RocksDB	• Juli 2021	• *Read(key)* • *Range scan(range, key)* • *Write(key, value)*	• Änderung pro Release sichtbar	• Keine anderen Datenbanken
LMDBJava [4]	• Java Key-Value Datenbanken	• Juni 2020	• *Write(value)* • *Get(key)* • *Get(Iterator)*	• Nur Key-Value Datenbanken	• Letztes Update
Yahoo! Cloud Serving Benchmark [5]	• NoSQL Systeme	• Februar 2021	• *Read(key)* • *Update (key, value)* • *Insert(key, value)* • *Scan (key, recordcount)*	• Unterstützt viele NoSQL Systeme	• Schlechte Dokumentation

Betrachtete Paper

[6] Brian F. C., Adam S., Erwin T., Raghu R., Russell S.: Benchmarking cloud serving systems with YCSB - https://dl.acm.org/doi/pdf/10.1145/1807128.1807152 , Juni 2010

[7] Pedro M., Maryam A., Filipe A. S.: A Study over NoSQL Performance - https://www.researchgate.net/publication/332028074_A_Study_over_NoSQL_Performance , April 2019

[8] Yishan L., Sathiamoorthy M.: A performance comparison of SQL and NoSQL databases – https://ieeexplore.ieee.org/document/6625441 , August 2013

[9] Omoruyi O., Kennedy O., Nsikan N., Charles N., Samuel J., Uzairue S.: Performance Benchmarking of Key-Value Store NoSQL Databases – https://www.researchgate.net/publication/330653733_Performance_Benchmarking_of_Key-Value_Store_NoSQL_Databases , Dezember 2018

[10] Veronika A., Jorge B., Pedro F.: Which NoSQL Database? A Performance Overview - https://www.researchgate.net/publication/292025334_Which_NoSQL_Database_A_Performance_Overview , Januar 2014

[11] John K., Ian G., Neil E., Patrick D., Kim P., Chrisjan M.: Performance Evaluation of NoSQL Databases: A Case Study- https://www.researchgate.net/publication/275033854_Performance_Evaluation_of_NoSQL_Databases_A_Case_Study , Februar 2015

[12] Hristo K., Emanuela M.: Performance Study of SQL and NoSQL Solutions for Analytical Loads - https://www.researchgate.net/publication/265964446_Performance_Study_of_SQL_and_NoSQL_Solutions_for_Analytical_Loads , September 2014

Paper

Vorteile:

- neben den „Standard“- Workloads wird zusätzlich Skalierbarkeit untersucht [6,11]
- „Vorbereitungsphase“ wird getestet [11]
- Delete wird getestet [11]

Nachteile:

- wenig (persistente) Key-Value-Datenbanken werden untersucht [6-12]
- Verfügbarkeit (Availability) wird nicht untersucht [6]
- Replizierbarkeit wird nicht untersucht [5]
- Implementierungsstrategie wird nicht beachtet [6-12]

Agenda

MOTIVATION

VERWANDTE ARBEITEN

ERGEBNISSE

AUSBLICK

Kriterien zur Wahl der Datenbanken

- LevelDB
- Ehcache
- MapDB
- Riak
- SSDB
- Voldemort
- Aerospike
- Tarantool
- Couchbase
- Redis
- Scalaris
- Tokyo Cabinet
- Kyoto Cabinet
- Hazelcast
- MVStore
- Amazon Dynamo
- RocksDB

Kriterien zur Wahl der Datenbanken

1. Persistenz

- LevelDB
- Ehcache
- MapDB
- Riak
- SSDB
- Voldemort
- Aerospike
- Tarantool
- Couchbase
- Redis
- Scalaris
- Tokyo Cabinet
- Kyoto Cabinet
- Hazelcast
- MVStore
- Amazon Dynamo
- RocksDB

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?

- LevelDB
- Ehcache
- MapDB
- Riak
- SSDB
- Voldemort
- Aerospike
- Tarantool
- Couchbase
- Redis
- Scalaris
- Tokyo Cabinet
- Kyoto Cabinet
- Hazelcast
- MVStore
- Amazon Dynamo
- RocksDB

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?

- LevelDB
- Ehcache
- MapDB
- Riak
- SSDB
- Voldemort
- Aerospike
- Tarantool
- Couchbase
- Redis
- Scalaris
- Tokyo Cabinet
- Kyoto Cabinet
- Hazelcast
- MVStore
- Amazon Dynamo
- RocksDB

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?
3. Implementierungsstrategie

Bibliotheken:

In dieser Kategorie sind die Key-Value Datenspeicher enthalten, die sich einbinden lassen und danach genutzt werden können.

Prozesse, die skalierbar sind:

In diese Kategorie fallen die Key-Value Datenspeicher, die als ein eigener Prozess ausgeführt werden und sich dynamisch bei steigender oder sinkender Last an den Workload adaptieren.

Mischform:

In diese Kategorie fallen die Key-Value Datenspeicher, die sich als Bibliothek einbinden lassen oder auch als Prozess gestartet werden können, der skalierbar ist.

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?
3. Implementierungsstrategie

Bibliothek

- LevelDB
- MapDB
- MVStore

Skalierbarer Prozess

- Riak
- Aerospike
- Voldemort
- Couchbase
- Ehcache
- SSDB

Mischform

- RocksDB + Rocksplicator

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?
3. Implementierungsstrategie

Bibliothek

- LevelDB
- MapDB
- MVStore

Skalierbarer Prozess

- Riak
- Aerospike
- Voldemort
- Couchbase
- Ehcache
- SSDB

Mischform

- RocksDB + Rocksplicator

Kriterien zur Wahl der Datenbanken

1. Persistenz
2. Wie aktuell sind die Datenbanken?
3. Implementierungsstrategie

Bibliothek

- LevelDB
- MapDB
- MVStore

Skalierbarer Prozess

- Riak
- Aerospike
- Voldemort
- Couchbase
- Ehcache
- SSDB

Mischform

- RocksDB + Rocksplicator

Gewählte Datenbanken

Bibliothek(1):

- MapDB
- MVStore

Skalierbarer Prozess(2):

- Voldemort
- Riak
- Aerospike

Mischform aus (1) + (2):

- RocksDB + Rocksplicator

Eigenschaften

Clusterfähigkeit:

Bezeichnet die Fähigkeit mehrerer Server oder Instanzen, eine Verbindung mit einer einzigen Datenbank herzustellen.

Integrationskomplexität:

Die Schwierigkeit die Datenbank zu installieren und in eine Anwendung zu integrieren.

Skalierbarkeit:

Unter der Skalierbarkeit von Datenbanken versteht man die Fähigkeit der Datenbank mit variabler Anzahl Zugriffen und großer Datenmengen umgehen zu können.

Replizierbarkeit:

Die Eigenschaft zur Verteilung und Erstellung von Kopien derselben Daten, um Verfügbarkeit, Latenz und Skalierbarkeit zu unterstützen.

Eigenschaften (Bibliothek)

Datenbank	Clusterfähigkeit	Integrations-Komplexität	Skalierbarkeit	Replizierbarkeit
MapDB	X	• direkt einbinden z.B als Maven Dependency	• basiert auf den Java Collections (Threads)	X
MVStore	X	• direkt einbinden z.B als Maven Dependency	• basiert auf den Java Collections (Threads)	X

Eigenschaften (Skalierbarer Prozess)

Datenbank	Clusterfähigkeit	Integrations-komplexität	Skalierbarkeit	Replizierbarkeit
Voldemort	✓	• Server kann entweder im Programm, durch Kommandozeile oder durch war Datei gestartet werden	• Reads und Writes skalieren horizontal	• Consistent Hashing • Multi-version concurrency control
Riak	✓	• für Windows zusätzliche Virtualisierung nötig	• Riak Ring	• Multi-Cluster Replikation
Aerospike	✓	• Docker nutzen	• Multi-threaded, eine oder mehrere Instanzen können auf mehrere Cores verteilt werden	• Master-Slave Zuweisung • synchronisierte Replikation • Rack-Awareness

Eigenschaften (Mischform)

Datenbank	Clusterfähigkeit	Integrations-komplexität	Skalierbarkeit	Replizierbarkeit
RocksDB + Rocksplicator	✓	• Docker nutzen • direkt einbinden z.B als Maven Dependency	• Auf mehreren Cores laufen lassen um Workloads zu verteilen	1. Asynchron 2. Semi-synchron 3. Synchron

Testablauf

Generierung von Testdaten:

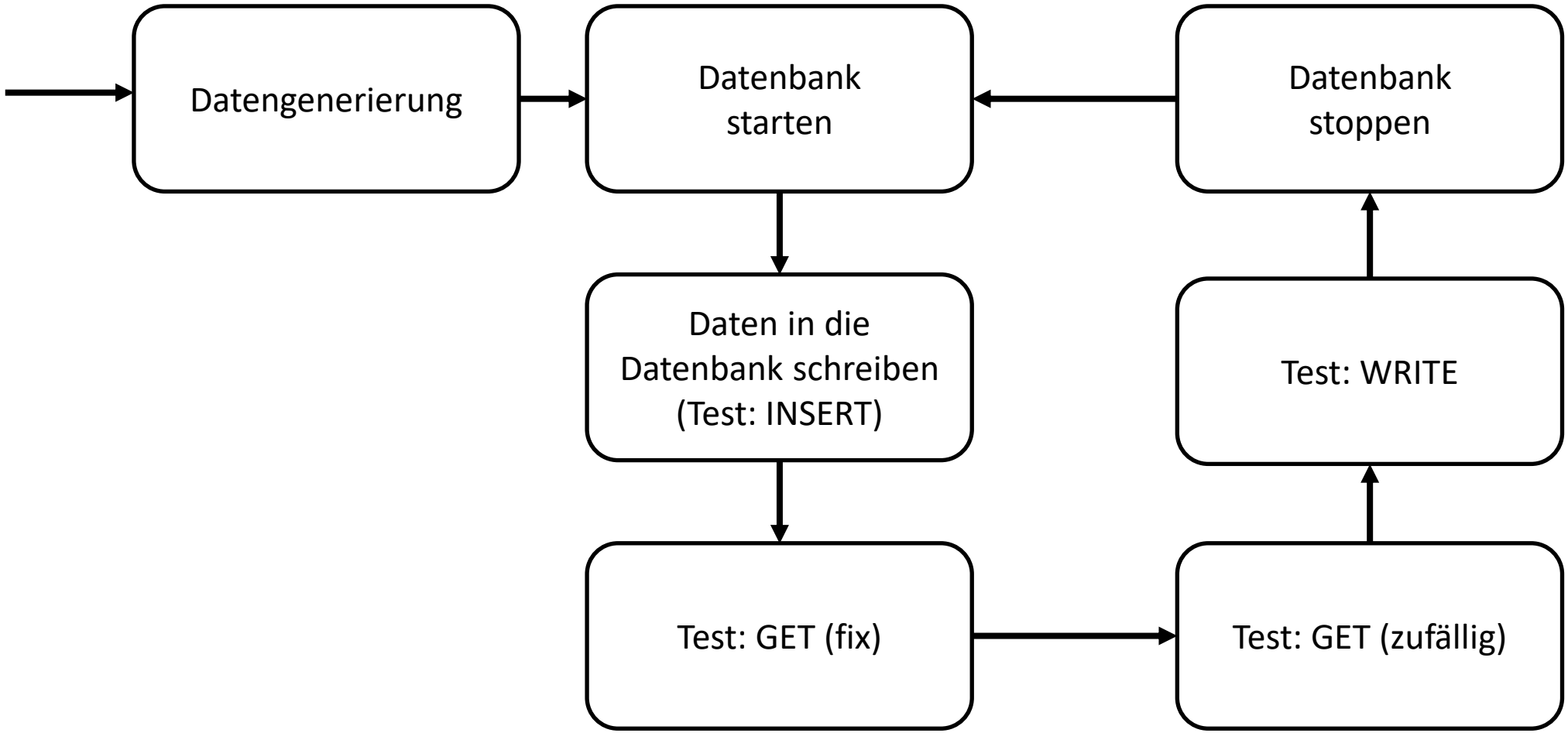
- Key ist 4 Bytes lang
- Key → Hashwert
- Value → Byte Array der Länge 4, 8, 16 KB

Beispiel:

Key = sha256(1)

Value = sha256(1), sha256(1), sha256(1), .. bis 4KB Größe erreicht ist

Testablauf


Datengenerierung
Datenbank starten
Datenbank stoppen
Daten in die Datenbank schreiben (Test: INSERT)
Test: WRITE
Test: GET (fix)
Test: GET (zufällig)

Agenda

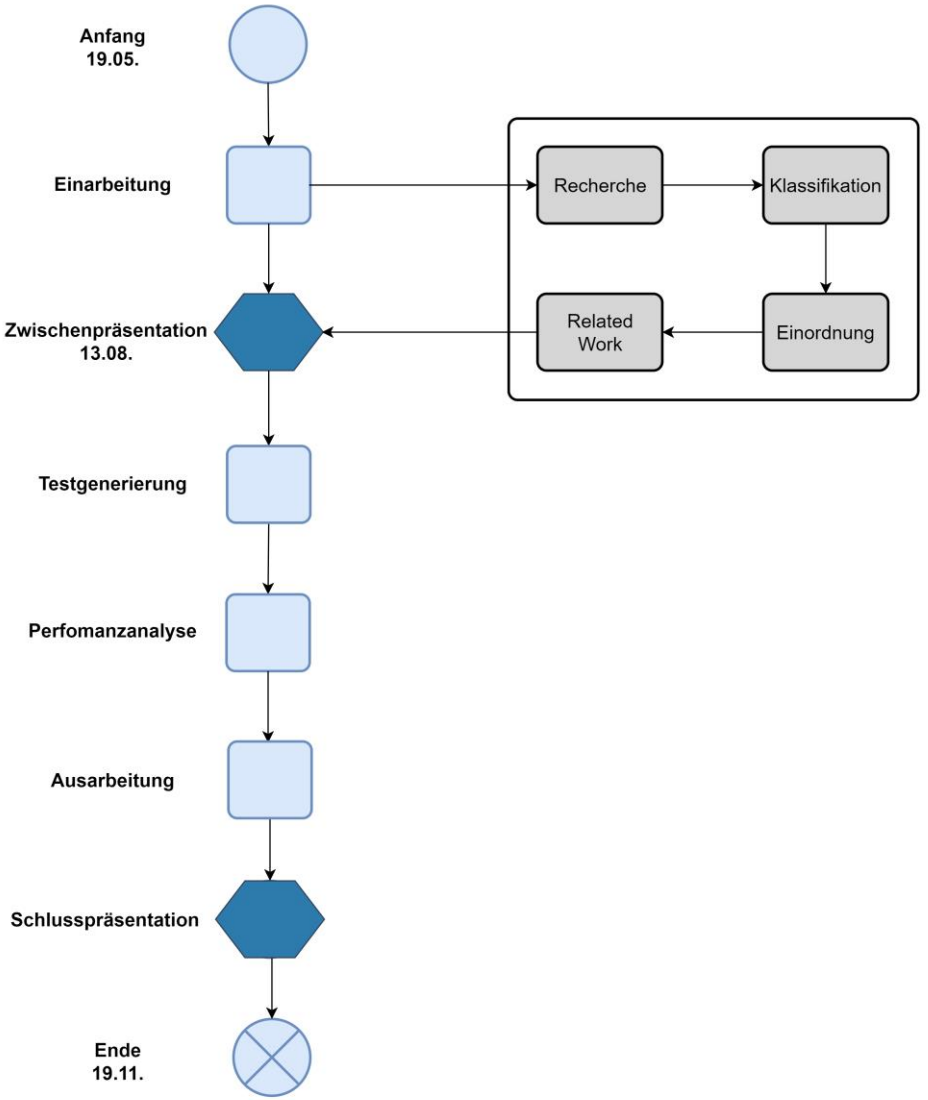
MOTIVATION

VERWANDTE ARBEITEN

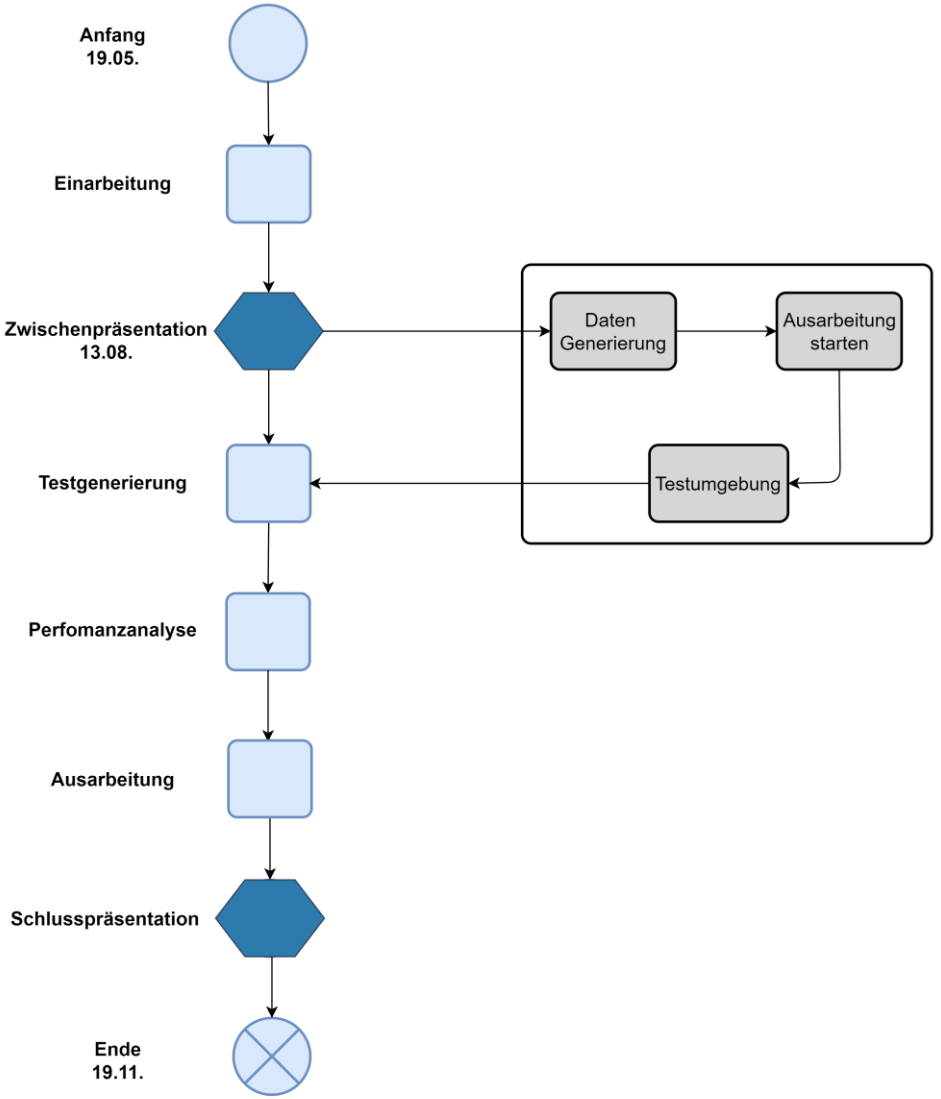
ERGEBNISSE

AUSBLICK

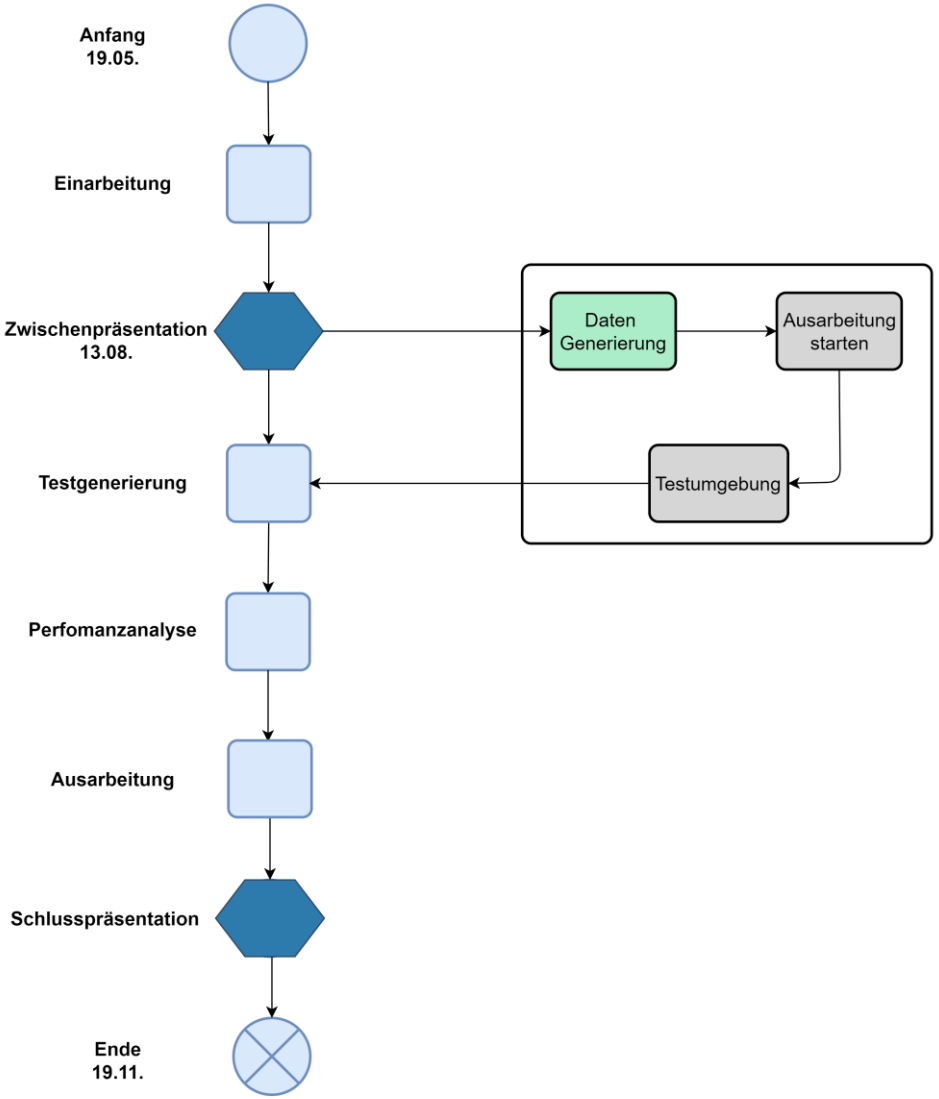
Ausblick


Anfang
19.05.
Einarbeitung
Recherche
Klassifikation
Zwischenpräsentation
13.08.
Related
Work
Einordnung
Testgenerierung
Perfomanzanalyse
Ausarbeitung
Schlusspräsentation
Ende
19.11.

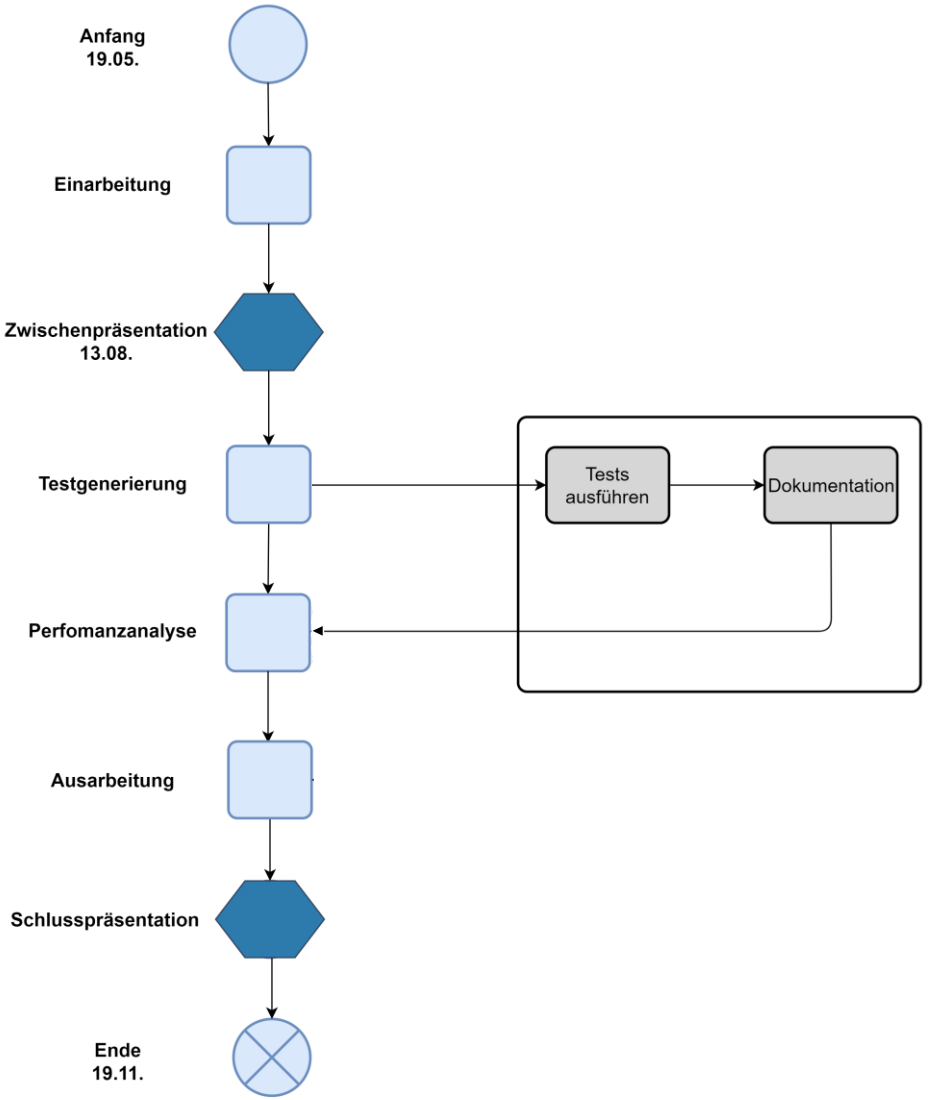
Ausblick


Anfang
19.05.
Einarbeitung
Zwischenpräsentation
13.08.
Daten
Generierung
Ausarbeitung
starten
Testumgebung
Testgenerierung
Perfomanzanalyse
Ausarbeitung
Schlusspräsentation
Ende
19.11.

Ausblick


Anfang
19.05.
Einarbeitung
Zwischenpräsentation
13.08.
Daten
Generierung
Ausarbeitung
starten
Testumgebung
Testgenerierung
Perfomanzanalyse
Ausarbeitung
Schlusspräsentation
Ende
19.11.

Ausblick


Anfang
19.05.
Einarbeitung
Zwischenpräsentation
13.08.
Testgenerierung
Tests
ausführen
Dokumentation
Perfomanzanalyse
Ausarbeitung
Schlusspräsentation
Ende
19.11.

Vielen Dank für Ihre Aufmerksamkeit !

Quellen

[1] Phiri, Hazael & Kunda, Douglas. (2017) A Comparative Study of NoSQL and Relational Database -View of A Comparative Study of NoSQL and Relational Database (icict.org.zm)

Benchmarks:

[2] InfluxDB - influxdata/influxdb-comparisons: Code for comparison write ups of InfluxDB and other solutions (github.com) (letzter Zugriff: 01.08.2021)

[3] RocksDB - Performance Benchmarks · facebook/rocksdb Wiki · GitHub (letzter Zugriff: 01.08.2021)

[4] LMDBJava - GitHub - lmdbjava/benchmarks: Benchmark of open source, embedded, memory-mapped, key-value stores available from Java (JMH) (letzter Zugriff: 01.08.2021)

[5] YSCB - brianfrankcooper/YCSB: Yahoo! Cloud Serving Benchmark (github.com) (letzter Zugriff: 01.08.2021)