--- title: "Tools & Concepts for Cloud Deployments" author: ["Christopher B. Hauser"] institute: ["Institute of Information Resource Management, Ulm University"] subject: "Markdown" tags: [Markdown, Example] titlepage: true graphics: true mainfont: Open Sans mainfontoptions: BoldFont=Open Sans Bold mainfontoptions: ItalicFont=Open Sans Italic mainfontoptions: BoldItalicFont=Open Sans Bold Italic

date: SummerSchool 2019, Curitiba subtitle: "Solution for Exercise 7" --- # Answers to questions

Lesson 1: Container Orchestration with Kubernetes

Questions: Kubernetes

What Featuers does Kubernetes provide?

* Automatic binpacking * Self-healing * Horizontal scaling * Service discovery and load balancing * Automated rollouts and rollbacks * Secret and configuration management * Storage orchestration * Batch execution

Where in our Cloud Stack do you place Kubernetes?

| Cloud Stack | Example | Deployment Tool | | --- | --- | --- | | **Application Component** | Mediawiki | Dockerfile/Bash | | **Containers** | Docker | Kubernetes | | **Virtual Resource** | Instance m1.small | - | | **Cloud Platform** | OpenStack | - |

Lesson 2: Concept of Reliability and Availability

Questions: Consistency Guarantees

What is ACID? What is eventual consistency?

ACID: Atomicity, Consistency, Isolation, Durability - describes that transactions with various changes or queries to a data store runs as one atomic action, while the database guarantees strict consistency, parallel running actions don't harm each other by isolating them, and that written changes are durably persisted and never lost. These guarantees are very strict and are implemented by locking mechanisms, which decreases availability and performance.

Eventual consistency - describes that changes will be visible at an unspecified point in time in the future to other queries. This guarantee is in contrast to ACID, but allows an increase on availability and performance.

What are the three categories of CAP?

Concistency, **A**vailability, **P**artition tolerant - and only two (but never three) of these three features can be achieved at the same time by a data store. While ACID always guarantees consistency, it will harm the availability in case of (network) failures, which lead

to partitioning of a distributed data store (CA). On the other hand, eventual consistency will rather choose availability and tolerate partitioning instead of consistency guarantees (AP).

What are examples for CA data store? What are examples for AP? And for CP?

* CA: critical data like banking, ticket shops * AP: social media content where eventual consistency is enough * CP: huge big data applications where single parts can fail but failing the whole would be too costly

Questions: Distributed Database Systems

What is the difference between replication and sharding/partitioning?

Replication copies data on other locations, while sharding splits the data in pieces and distributes them to locations. Distributed NoSQL database systems often use both at the same time, while relational database systems (e.g. MySQL) often don't shard the data but allow replication.

How does replication relate to the CAP theorem?

Replication can improve the availability property and can be used to recover in case of failures or partitions. Replication hence has direct influence to the AP properties.

How does sharding relate to the CAP theorem?

Sharding distributes the workload to many nodes in parallel, and hence influences the performance (availability). In case of failures or partitions not 100% of the data is lost but only one shard (partition tolerance).

How would the perfect Cloud database system look like?

The perfect Cloud database would have the following characteristics:

* highly, linearly scalable * enough consistency guarantees depending on the application data * flexibly add and remove hosts to the database system, depending on the query load * automatic recovery of failed hosts Yet, the CAP theorem limits the practical possibilities of this wish list. Further, scaling in/out means to transport data, which takes time and affects the performance and availability of the database system.