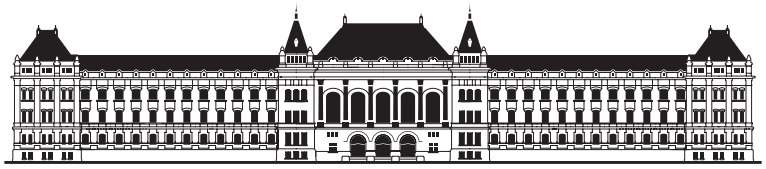

M Ű E G Y E T E M 1 7 8 2
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Fault Tolerant Systems Research Group

Critical Architectures Laboratory
Spring Semester 2019/2020

Distributed Data Storage in NoSQL Databases

Syllabus
v1.4

Author: Gábor Szárnyas
(szarnyas@mit.bme.hu)

March 31, 2020

1 Introduction

This document contains the theoretical and practical background for the *Distributed Data Storage in NoSQL Databases* sessions of the *Critical Architectures Laboratory* course. The purpose of this session is to demonstrate the practical side of replication and distributed data storage (lecture notes in Hungarian are available in [6]).

Big Data and the NoSQL Movement Since the 1980s, database management systems based on the relational data model dominated the database market. Relational databases have a number of important advantages: precise mathematical background, understandability, mature tooling and so on. However, due to their rich feature set and the strongly connected nature of their data model, relational databases often have scalability issues [5]. They are typically optimized for transaction processing, instead of data analysis (see *data warehouses* for an exception). In practice, these render them impractical for a number of use cases, e.g. running complex queries on large data sets.

In the last decade, large organizations struggled to store and process the huge amounts of data they produced. This problem introduces a diverse palette of scientific and engineering challenges, called *Big Data* challenges. These challenges spawned dozens of new database management systems. Typically, these systems broke with the strictness of the relational data model and utilized simpler, more scalable data models. These systems dropped support for the SQL query language used in relational databases and hence were called *NoSQL databases*[1] [1]. Because relational databases are not suitable for large-scale model-driven applications, we experimented with numerous NoSQL databases.

2 Concepts

2.1 Consistency in a Distributed System

2.1.1 The CAP-theorem

In 1999, Eric Brewer, a professor at Berkeley University published a set of informal requirements for a distributed system, called the CAP properties. Next year, in the keynote speech of PODC (Principles of Distributed Computing), he presented the CAP-conjecture [2]. The conjecture states that in any given moment, a web service can only guarantee two of the following properties: consistency, availability, partition tolerance. The concepts are roughly defined as follows [3]:

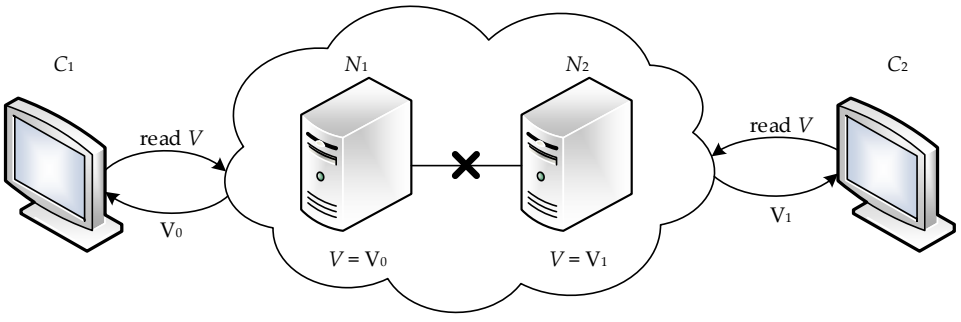

Figure 1: The V data item is not consistent, clients C_1 and C_2 see different values

[1]The database community now mostly interprets NoSQL as "not only SQL".

Consistency Consistency means whether and how a system is in a consistent state after the execution of an operation. A distributed system is typically considered to be consistent if after an update operation of some writer all readers see his updates in some shared data source. (Nevertheless there are several alternatives towards this strict notion of consistency as we will see below.) See Figure 1.

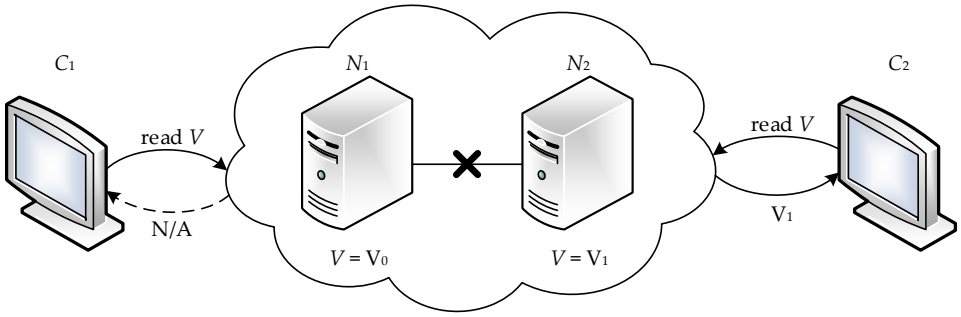

Figure 2: The V data item is not available for client C_1

Availability Availability means that a system is designed and implemented in a way that allows it to continue operation (i.e. allowing read and write operations) if e.g. nodes in a cluster crash or some hardware or software parts are down due to upgrades. See Figure 2.

Partition tolerance Partition tolerance is the the ability of the system to continue operation in the presence of network partitions. These occur if two or more "islands" of network nodes arise, which (temporarily or permanently) cannot connect to each other. Some people also understand partition tolerance as the ability of a system to cope with the dynamic addition and removal of nodes (e.g. for maintainance purposes; removed and again added nodes are considered an own network partition in this notion).

The conjecture was formalized and proved in 2002 by two researchers in MIT (Massachusetts Institute of Technology), Seth Gilbert and Nancy Lynch [4]. In this form, the CAP-theorem defines a limitation for all distributed systems.

In short, the CAP-theorem can be stated as follows: in a distributed system in the case of network partition, the operations of the system will not be atomic and/or the data elements will be unavailable.

Formally: in a distributed system running on an asynchronous network, the system cannot guarantee the following properties:

- availability,
- atomic consistency.

2.2 Replication

Replication, consistency and availability are strongly correlated. We use the following notations [7]:

- N – the number of nodes that store a replica of the data.
- W – the number of replicas that need to acknowledge the receipt of the update before the update completes.

- R – the number of replicas that are contacted when a data object is accessed through a read operation.

Quorum protocols enforce that the following inequalities holds:

1. $W > N/2$
2. $W + R > N$

The first condition guarantees that any two write quorums have a mutual node, which makes it possible to preserve the order of subsequent writes. The second guarantees that each read quorum and write quorum overlap, so that we will not read stale data.

2.3 Sharding

Sharding is the process of determining the location of each data item in a distributed system. Sometimes this process is called *partitioning* or *segmenting*.

References

[1] NoSQL Databases. `http://nosql-database.org/`, October 2013.

[2] Eric A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, PODC '00, pages 7–, New York, NY, USA, 2000. ACM.

[3] Christof Strauch. NoSQL Databases. `http://www.christof-strauch.de/nosqldbs.pdf`, October 2013.

[4] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002.

[5] Adam Jacobs. The pathologies of big data. *Commun. ACM*, 52(8):36–44, August 2009.

[6] Majzik István. Többpéldányos adatkezelés. `http://www.inf.mit.bme.hu/sites/default/files/materials/category/kateg%C3%B3ria/oktat%C3%A1s/msc-t%C3%A1rgyak/szolg%C3%A1ltat%C3%A1sbiztons%C3%A1gra-tervez%C3%A9s/13/SZBT-2013_EA05_tobbpeldanyos_adatkezeles.pdf`.

[7] Werner Vogels. Eventually consistent. *Commun. ACM*, 52(1):40–44, January 2009.