Scalable Services: Understanding Architecture Trade-off

Markus Klems and Stefan Tai

Karlsruhe Institute of Technology, Germany
{markus.klems, stefan.tai}@kit.edu
<http://www.kit.edu>

Abstract. Creating Internet-scale services is a critical challenge for many organizations today. Data storage is a key component and factor to scalability, and data partitioning and replication along with loose coupling and simple service interfaces have become successful architecture guidelines to preventing scalability issues.

Partitioning realizes incremental scalability by splitting up large data sets and distributing smaller data shards across multiple servers. Replication and loose coupling help tolerate server failures and improve service availability. Replica synchronization, however, can produce substantial volumes of server-to-server traffic and delay service response time.

Distribution of storage infrastructure, consequently, provokes fundamental trade-off challenges, known as the strong CAP principle: only two out of the three properties of a distributed system, strong consistency (C), high availability (A), and partition-tolerance (P), can be achieved at the same time. For example, a transactional database on a single node provides CA without P, a distributed database system with pessimistic locking provides CP without A, and the Domain Name System provides AP without C. The weak CAP principle generalizes the strong CAP principle by characterizing the trade-off as a continuum instead of binary choices. In particular, relaxing consistency requirements and trading consistency for higher availability has become a successful *modus operandi* for Internet-scale systems.

Key-value data stores provide storage capabilities for a wide range of applications and services, from Amazon's shopping carts to Zynga's social gaming engine. We explore common mechanisms employed by Internet-scale key-value data stores, such as Dynamo, Cassandra, and Membase, and discuss how key-value data stores are used in support of representative Internet applications and services.

To evaluate and compare eventually consistent data stores, metrics and benchmarking tools are needed. We review metrics proposed by the distributed systems community and argue for a novel consistency benchmarking model as a systematic approach to measure relaxed consistency.

Keywords: Service-Oriented Computing, Cloud Computing, Data Store Architecture.