Many cloud services are based on online transaction processing (OLTP) and operate under tight latency

constraints. Moreover, these applications have to deal with extremely high data volumes and are expected

to provide reliable services for very large communities of users. It did not take very long for companies

heavily involved in cloud computing, such as Google and Amazon, e-commerce companies such as

eBay, and social media networks such as Facebook, Twitter, or LinkedIn, to discover that traditional

relational databases are not able to handle the massive amount of data and the real-time demands of

online applications that are critical for their business models.

The search for alternate models with which to store the data on a cloud is motivated by the need

to decrease the latency by caching frequently used data in memory on dedicated servers, rather than

fetching it repeatedly. In addition, distributing the data on a large number of servers allows multiple

transactions to occur at the same time and decreases the response time. The relational schema are of little

use for such applications in which conversion to key-value databases seems a much better approach.

Of course, such systems do not store meaningful metadata information, unless they use extensions that

cannot be exported easily.

A major concern for the designers of OLTP systems is to reduce the response time. The term

memcaching refers to a general-purpose distributed memory system that caches objects in main memory

(RAM); the system is based on a very large hash table distributed across many servers. The memcached

system is based on a client-server architecture and runs under several operating systems, including

Linux, Unix, Mac OS X, and Windows. The servers maintain a key-value associative array. The API

allows the clients to add entries to the array and to query it. A key can be up to 250 bytes long, and a

value can be no larger than 1MB. The memcached system uses an LRU cache-replacement strategy.

Scalability is the other major concern for cloud OLTP applications and implicitly for datastores.

There is a distinction between vertical scaling, where the data and the workload are distributed to

systems that share resources such as cores and processors, disks, and possibly RAM, and horizontal

scaling, where the systems do not share either primary or secondary storage [66].

Cloud stores such as document stores and NoSQL databases are designed to scale well, do not exhibit

a single point of failure, have built-in support for consensus-based decisions, and support partitioning

and replication as basic primitives. Systems such as Amazon’s SimpleDB, discussed in Section 3.1;

CouchDB (see http://couchdb.apache.org/), or Oracle NoSQL database [277] are very

popular, though they provide less functionality than traditional databases. The key-value data model is very popular. Several such systems, including Voldemort, Redis, Scalaris, and Tokyo cabinet, are

discussed in [66].

The “soft-state” approach in the design of NoSQL allows data to be inconsistent and transfers the

task of implementing only the subset of the ACID properties required by a specific application to the

application developer. The NoSQL systems ensure that data will be “eventually consistent” at some

future point in time instead of enforcing consistency at the time when a transaction is “committed.”

Data partitioning among multiple storage servers and data replication are also tenets of the NoSQL

philosophy7; they increase availability, reduce response time, and enhance scalability.

The name NoSQL given to this storage model is misleading. Michael Stonebreaker notes [335] that

“blinding performance depends on removing overhead. Such overhead has nothing to do with SQL, but

instead revolves around traditional implementations of ACID transactions, multi-threading, and disk

management.”

The overhead of OLTP systems is due to four sources with equal contribution: logging, locking,

latching, and buffer management. Logging is expensive because traditional databases require transaction

durability; thus, every write to the database can be completed only after the log has been updated.

To guarantee atomicity, transactions lock every record, and this requires access to a lock table. Many

operations require multithreading, and the access to shared data structures, such as lock tables, demands

short-term latches8 for coordination. The breakdown of the instruction count for these operations in

existing DBMSs is as follows: 34.6% for buffer management, 14.2% for latching, 16.3% for locking,

11.9% for logging, and 16.2% for hand-coded optimization [157].

Today OLTP databases could exploit the vast amounts of resources of modern computing and communication

systems to store the data in main memory rather than rely on disk-resident B-trees and heap

files, locking-based concurrency control, and support for multithreading optimized for the computer

technology of past decades [157]. Logless, single-threaded, and transaction-less databases could replace

the traditional ones for some cloud applications.

Data replication is critical not only for system reliability and availability, but also for its performance.

In an attempt to avoid catastrophic failures due to power blackouts, natural disasters, or other causes

(see also Section 1.6), many companies have established multiple data centers located in different

geographic regions. Thus, data replication must be done over a wide area network (WAN). This could

be quite challenging, especially for log data, metadata, and system configuration information, due to

increased probability of communication failure and larger communication delays. Several strategies are

possible, some based on master/slave configurations, others based on homogeneous replica groups.

Master/slave replication can be asynchronous or synchronous. In the first case the master replicates

write-ahead log entries to at least one slave, and each slave acknowledges appending the log record as

soon as the operation is done. In the second case the master must wait for the acknowledgments from

all slaves before proceeding. Homogeneous replica groups enjoy shorter latency and higher availability

than master/slave configurations. Any member of the group can initiate mutations that propagate

asynchronously.

flexible one tailored to the specific requirements of the applications. Sometimes the data management

ecosystem of a cloud computing environment integrates multiple databases; for example, Oracle integrates

its NoSQL database with the HDFS discussed in Section 8.6, with the Oracle Database, and with

theOracle Exadata. Another approach, discussed in Section 8.10, is to partition the data and to guarantee

full ACID semantics within a partition, while supporting eventual consistency among partitions.