There are several risks involved when a large organization relies solely on a single cloud provider. As

the short history of cloud computing shows, cloud services may be unavailable for a short or even an

extended period of time. Such an interruption of service is likely to negatively impact the organization

and possibly diminish or cancel completely the benefits of utility computing for that organization.

The potential for permanent data loss in case of a catastrophic system failure poses an equally great

danger.

Last but not least, a Cloud Service Provider (CSP) may decide to increase the prices for service and

charge more for computing cycles, memory, storage space, and network bandwidth than other CSPs.

The alternative in this case is switching to another provider. Unfortunately, this solution could be very

costly due to the large volume of data to be transferred from the old to the new provider. Transferring

terabytes or possibly petabytes of data over the network takes a fairly long time and incurs substantial

charges for the network bandwidth.

This chapter discusses the storage models supported by the cloud infrastructure provided by Amazon,

Google, and Microsoft; Chapter 8 covers the architecture of storage systems in general. Reliability is

a major concern, and here we discuss a solution that addresses both avoidance of vendor lock-in and

storage reliability.

A solution to guarding against the problems posed by the vendor lock-in is to replicate the data to

multiple cloud service providers. Straightforward replication is very costly and, at the same time, poses

technical challenges. The overhead to maintain data consistency could drastically affect the performance

of the virtual storage system consisting of multiple full replicas of the organization’s data spread over

multiple vendors. Another solution could be based on an extension of the design principle of a RAID-5

system used for reliable data storage.

A RAID-5 system uses block-level stripping with distributed parity over a disk array, as shown in

Figure 3.5(a); the disk controller distributes the sequential blocks of data to the physical disks and

computes a parity block by bit-wise XOR-ing of the data blocks. The parity block is written on a different

disk for each file to avoid the bottleneck possible when all parity blocks are written to a dedicated disk,

as is done in the case of RAID-4 systems. This technique allows us to recover the data after a single

disk loss. For example, if Disk 2 in Figure 3.5 is lost, we still have all the blocks of the third file, c1,

c2, and c3, and we can recover the missing blocks for the others as follows:

a2 = (a1) XOR (aP) XOR (a3)

b2 = (b1) XOR (bP) XOR (b3)

d1 = (dP) XOR (d2) XOR (d3)

. (39)

Obviously, we can also detect and correct errors in a single block using the same procedure. The

RAID controller also allows parallel access to data (for example, the blocks a1, a2, and a3 can be read

and written concurrently) and it can aggregate multiple write operations to improve performance.

The system in Figure 3.5(b) strips the data across four clusters. The access to data is controlled

by a proxy that carries out some of the functions of a RAID controller, as well as authentication and

other security-related functions. The proxy ensures before-and-after atomicity as well as all-or-nothing

atomicity for data access; the proxy buffers the data, possibly converts the datamanipulation commands, optimizes the data access (e.g., aggregates multiple write operations), converts data to formats specific

to each cloud, and so on.

This elegant idea immediately raises several questions: How does the response time of such a scheme

comparewith that of a single storage system?Howmuch overhead is introduced by the proxy?Howcould

this scheme avoid a single point of failure, the proxy? Are there standards for data access implemented

by all vendors?

An experiment to answer some of these question is reported in [5]; the Redundant Array of Cloud

Storage (RACS) system uses the same data model and mimics the interface of the S3 provided by AWS.

The S3 system, discussed in Section 3.1, stores the data in buckets, each bucket being a flat namespace

with keys associated with objects of arbitrary size but less than 5 GB. The prototype implementation

discussed in [5] led the authors to conclude that the cost increases and the performance penalties of

the RACS systems are relatively minor. The paper also suggests an implementation to avoid the single

point of failure by using several proxies. Then the system is able to recover from the failure of a single

proxy; clients are connected to several proxies and can access the data stored on multiple clouds.

It remains to be seen whether such a solution is feasible in practice for organizations with a very

large volume of data, given the limited number of cloud storage providers and the lack of standards for

data storage. A basic question is whether it makes sense to trade basic tenets of cloud computing, such

as simplicity and homogeneous resources controlled by a single administrative authority, for increased

reliability and freedom from vendor lock-in [67].

This brief discussion hints at the need for standardization and for scalable solutions, two of the many

challenges faced by cloud computing in the near future. The pervasive nature of scalability dominates all

aspects of cloud management and cloud applications; solutions that perform well on small systems are

no longer feasible when the number of systems or the volume of the input data of an application increases

by one or more orders of magnitude. Experiments with small test-bed systems produce inconclusive

results. The only alternative is to conduct intensive simulations to prove (or disprove) the advantages of a

particular algorithm for resource management or the feasibility of a particular data-intensive application.

We can also conclude that cloud computing poses challenging problems to service providers and

to users. The service providers have to develop strategies for resource management subject to quality

of service and cost constraints, as discussed in Chapter 6. At the same time, the cloud application

developers have to be aware of the limitations of the cloud computing model.