12.2.2.3 Hypervisor Products

Several hypervisor products have been released since the late 1990s. One of the early hypervisors was the product by VMware. It started off as a Stanford University Research project that was then commercialized. Essentially, VMware developed one of the early virtualization software for the cloud. As stated in [VM], VMware software provides a completely virtualized set of hardware to the guest OS. VMware software virtualizes the hardware for a video adapter, a network adapter, and hard disk adapters. The host provides pass-through drivers for guest USB, serial, and parallel devices. Thus, VMware VMs are portable between computers because every host looks nearly identical to the guest. VMware provides desktop virtualization, server virtualization, cloud virtualization, and applications. Its desktop product suite includes VMwareWorkstation and VMwareFusion. Its server products include VMwareESX and VMotion. Its cloud products include VMwarevCloud. Its application products include VMware vFabric tc Server.

Another notable hypervisor product is XEN. Like VMware, it started as a research project at the University of Cambridge, England and is marketed through Citrix. XEN Hypervisor runs just on top of the hardware and traps all calls by VMs to access the hardware. Domain 0 (Dom0) is a modified version of Linux that is used to manage the other VMs. Domain U (DomU) is the user domain in XEN. DomU is where all of the untrusted guest OSs reside. DomU is broken into two parts: Para-virtualized Domains (PV) and Hardware Assisted Virtualized Machine (HVM) Domains. A Para-virtualized domain is a modified OS which is aware that it is a VM and can achieve near-native performance. HVM Domain is a VM that runs OSs that have not been modified to work with Dom0. PVs are given read-only access to memory and any updates are controlled by the hypervisor. HVMs are given a shadow page table because they do not know how to work with noncontiguous physical address spaces. I/O (input/output) management is controlled by Dom0. PVs share memory with Dom0 through which they can pass messages.

There are several white papers and articles on both VMware and XEN. Details can be found in [MATH09]. We will discuss security details of these hypervisors in Chapter 17.

12.2.3 Cloud Frameworks

This section discusses the cloud service providers as illustrated in Figure 12.4. We have utilized these frameworks in our experimental work. Our experimental systems will be discussed in Part IV.

12.2.3.1 Hadoop, MapReduce Framework

As stated in [AWS], Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license. It enables applications to

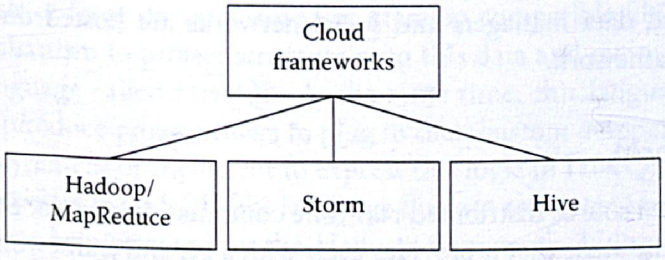

Figure 12.4 Cloud frameworks.

work on numerous machines and generates massive amounts of data. Hadoop was derived from Google's MapReduce and GoogleFileSystem (GFS) papers. Hadoop is written in Java.

Hadoop consists of the Hadoop Common, which provides access to the file systems supported by Hadoop. For effective scheduling of work, every Hadoop-compatible file system should provide location awareness: the name of the rack where a worker node is. Hadoop applications can use this information to run work on the node where the data is, and, failing that, on the same rack/switch, so reducing backbone traffic. HDFS uses this when replicating data, to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure so that even if these events occur, the data may still be readable.

A small Hadoop cluster will include a single master and multiple worker nodes. The master node consists of a JobTracker, TaskTracker, NameNode, and DataNode. A slave or *worker node* acts as both a DataNode and TaskTracker, though it is possible to have data-only worker nodes, and compute-only worker nodes; these are normally only used in nonstandard applications. In a larger cluster, the HDFS is managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures. This prevents file system corruption and reduces loss of data.

HDFS is a distributed, scalable, and portable file system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single datanode; a cluster of datanodes form the HDFS cluster. Above the file systems comes the MapReduce engine, which consists of one JobTracker, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible. With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data reside, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network. If a TaskTracker fails or times out, that part of the job is rescheduled.

We have used the Hadoop/MapReduce framework in all of our cloud implementations to be discussed in later chapters. That is, many of our prototypes

fact that (i) they do not provide adequate security mechanisms to protect sensitive data and (ii) they do not have the capability to process massive amounts of semantic web and geospatial data.

We are utilizing the state-of-the-art hardware, software, and data components and building an infrastructure that handles the inadequacies of the current cloud computing infrastructures. In particular, we (i) use modern hardware parts (e.g., secure coprocessors) to improve the performance due to incorporating additional security functionalities, (ii) integrate open-source software parts as well as custom-developed software parts to support query operations on complex data, (iii) support fine-grained access control and reference monitor support to provide security for complex data, and (iv) provide strong authentication mechanisms for cloud computing. The infrastructure facilitates several areas of computer and information science as well as social science research, including security and privacy, semantic web, geospatial information management, and social network analysis.

The technical contribution of our work is a cloud that supports fine-grained access control, storage of encrypted and sensitive data, complex query processing for massive data sets, and authentication mechanisms. This cloud is the first of its kind that we have developed for applications such as assured information sharing with massive amounts of data and consists of state-of-the-art hardware, software, and data components. It is utilized by our research projects on semantic web data management, assured information sharing, and automated reference monitors, among others. It should be noted that the development of our infrastructure is being carried out with funding from the AFOSR as well as the NSF and it is a work in progress.

The organization of this chapter is as follows: in Section 32.2, we will describe our infrastructure. Integrating with other infrastructures will be discussed in Section 32.3. Research enhancement with our infrastructure is discussed in Section 32.4. Education and performance aspects are discussed in Section 32.5. This chapter is summarized in Section 32.6. Figure 32.1 illustrates the contents of this chapter. Figure 32.2 illustrates the components of the infrastructure.

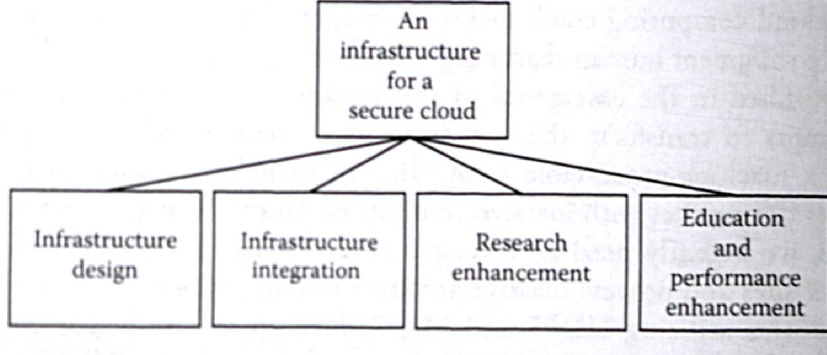

Figure 32.1 An infrastructure for a secure cloud.

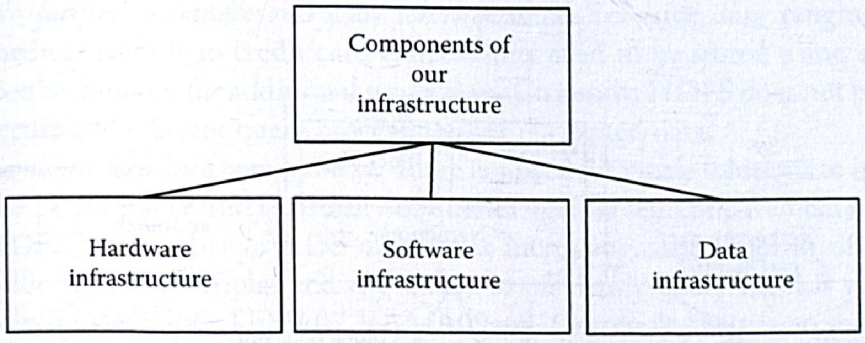

Figure 32.2 Components of our infrastructure. (From Hamlen, K.W., Kantarcioglu, M., and Khan, L. Security Issues for Cloud Computing, *This paper appears in the International Journal of Information Security and Privacy*, Namati, H. (ed.), 4(2), 36–48. © 2010, IGI Global. With permission.)

32.2 Description of the Research Infrastructure

32.2.1 Background

32.2.1.1 The Need for Our Infrastructure

There is a need to securely store, manage, share, and analyze massive amounts of data. For example, we may want to analyze multiple years of stock market data statistically to reveal a pattern or to build a reliable weather model based on several years of weather and related data. To handle such massive amounts of data distributed at many sites (i.e., nodes), we need an infrastructure with scalable hardware and software components. The cloud computing model has emerged to address the explosive growth of web-connected devices and handle massive amounts of data. It is defined and characterized by massive scalability and new Internet-driven economics. Hadoop is emerging as a superior software solution for cloud computing together with the integrated software parts such as Mahout, Hama, and MapReduce [MAHO, HAMA, CHU07, DEAN04]. Infrastructures such as HP's Open Cirrus test bed are utilizing HDFS. However, while such infrastructures have the advantages that come with Hadoop, they also have the limitations of Hadoop. These include the inability to handle complex data and have inadequate security protections. Owing to the fact that there is no infrastructure that can handle petabyte data sets consisting of semantic web and geospatial data as well as to provide secure storage and access to these data, we are developing such an infrastructure. Our infrastructure consists of a hardware component, a software component, and a data component.

32.2.1.2 Hadoop for Cloud Computing

A major part of the software component of our infrastructure is HDFS that is a distributed Java-based file system with the capacity to handle a large number of nodes storing

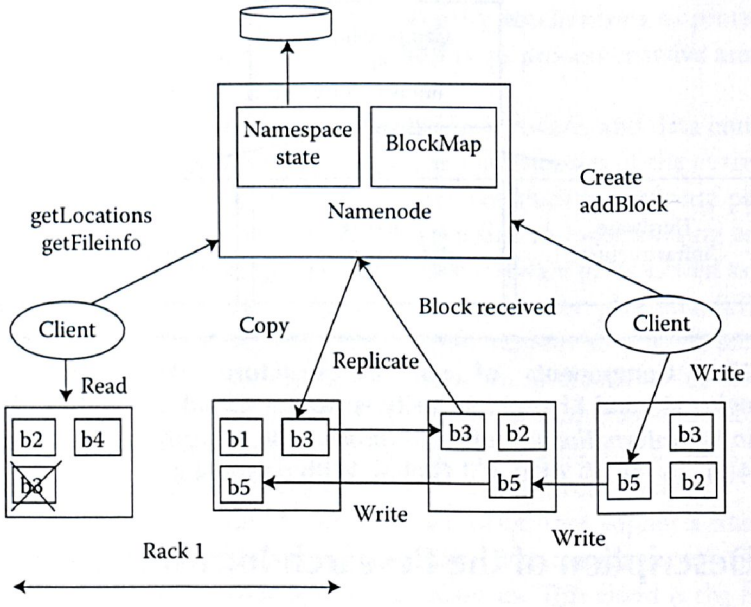

Figure 32.3 HDFS architecture.

petabytes of data. Ideally, a file size is a multiple of 64 MB. Reliability is achieved by replicating the data across several hosts. The default replication value is 3 (i.e., data are stored on three nodes). Two of these nodes reside on the same rack whereas the other node is on a different rack. A cluster of data nodes constructs the file system. The nodes transmit data over HTTP and clients access the data using a web browser. The data nodes communicate with each other to regulate, transfer, and replicate data.

HDFS architecture is based on the master–slave approach (Figure 32.3). The master is called a Namenode and contains metadata. It keeps the directory tree of all files and tracks from which data are available which node across the cluster. This information is stored as an image in the memory. Data blocks are stored in Datanodes. The Namenode is the single point of failure as it contains the metadata. So, there is an optional secondary Namenode that can be setup on any machine. The client accesses the Namenode to get the metadata of the required file. After getting the metadata, the client directly talks to the respective Datanodes to obtain data or to perform IO (input/output) actions [HADO]. On top of the file systems, there exists the *Map/Reduce engine*. This engine consists of a JobTracker. The client applications submit Map/Reduce jobs to this engine. The JobTracker attempts to place the work near the data by pushing the work out to the available Task Tracker nodes in the cluster.

32.2.1.3 Inadequacies of Hadoop

At the time of writing, the current infrastructures utilizing Hadoop have the following limitations:

1. *No facility to handle encrypted sensitive data:* Sensitive data ranging from medical records to credit card transactions need to be stored using encryption techniques for additional protection. Currently, HDFS does not perform secure and efficient query processing over encrypted data.
2. *Semantic web data management:* There is a need for viable solutions to improve the performance and scalability of queries against semantic web data such as RDF. The number of RDF datasets is increasing. The problem of storing billions of RDF triples and the ability to efficiently query them is yet to be solved [MUYS06, TESW07, RAMA09a-c]. Currently, there is no support to store and retrieve RDF data in HDFS.
3. *No fine-grained access control:* HDFS does not provide fine-grained access control. There is some work to provide access control lists for HDFS [ZHAN09]. For many applications such as assured information sharing, access control lists are not sufficient and there is a need to support more complex policies.
4. *No strong authentication:* A user who can connect to the JobTracker can submit any job with the privileges of the account used to set up the HDFS. Future versions of HDFS will support network authentication protocols such as Kerberos for user authentication and encryption of data transfers [ZHAN09]. However, for some assured information-sharing scenarios, we need PKIs to provide digital signature support.

32.2.2 Infrastructure Development

We are developing an infrastructure for secure cloud computing. It consists of a hardware component (includes 800 TB—terabytes) of data storage on a mechanical disk drive, 2400 GB (gigabytes) of memory and 100 commodity computers, a software component (includes Hadoop), and a data component (includes a semantic web data repository). This infrastructure provides the following support to researchers: (a) efficient storage of encrypted sensitive data, (b) store, manage, and query massive amounts of data, (c) fine-grained access control, and (d) strong authentication. The development of the infrastructure consists of four parts to address each of the limitations of HDFS (as shown in Figure 32.4):

1. Incorporate secure coprocessor (SCP) parts to the hardware component to enable efficient encrypted storage of sensitive data.
2. Incorporate several parts to the software component including Mahout [MAHO], Hama [HAMA], Jena [JENA], and Pellet [PELL] as well as develop parts for SPARQL query processing over HDFS.
3. Incorporate an XACML [MOSE05] implementation part to the software component for fine-grained access control.
4. Incorporate a part for flexible authentication mechanisms.