Chapter 4

Computing as a Service

"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers."

—Admiral Grace Hopper

The two simplest forms of cloud computing are perhaps the best known to scientists and engineers: storage and computing on demand. We covered storage in part I of the book; we turn next to cloud computing: computing as a service.

A cloud can provide near-instant access to as much computing and storage as you need, when you need it, along with the ability to obtain computers with the specific configuration(s) that you want. Furthermore, these capabilities are all just an API call or a few mouse clicks away. As we will see, computing services can be delivered in several different ways. The most basic is known in the cloud industry as **infrastructure as a service** (IaaS) because it provides virtualized infrastructure to its users. To understand what IaaS compute looks like, let's consider some typical scenarios.

- You need 100 CPUs now, rather than when your job reaches the head of the queue in your lab cluster.
- You need to access a GPU cluster or a computer with 1 TB memory, but such a resource does not exist in the lab.
- You need computers loaded with 10 different Linux variants to test the portability of your new software prior to distribution.

- You want a machine outside your firewall that you can share with your collaborators for a project.

Each scenario can be addressed in multiple ways by cloud computing. The question we look at in this chapter is how to determine the best approach and how to evaluate the pros and cons of picking a solution.

4.1 Virtual Machines and Containers

Public cloud data centers comprise many thousands of individual servers. Some servers are used exclusively for data services and supporting infrastructure and others for hosting your computations. When you compute in the cloud, you do not run directly on one of these servers in the way that you would in a conventional computational cluster. Instead, you are provided with a virtual machine running your favorite operating system. A **virtual machine** is just the software image of a complete machine that can be loaded onto the server and run like any other program. The server in the data center runs a piece of software called a **hypervisor** that allocates and manages the server's resources that are granted to its "guest" virtual machines. In the next chapter, we delve into how virtualization works, but the key idea is that when you run in a VM, it looks exactly like a server running whatever operating system the VM is configured to run.

For the cloud operator, virtualization has huge advantages. First, the cloud provider can provide dozens of different operating systems packaged as VMs for the user to choose from. To the hypervisor, all VMs look the same and can be managed in a uniform way. The cloud management system (sometimes called the **fabric controller**) can select which server to use to run the requested VM instances, and it can monitor the health of each VM. If needed, the cloud monitor can run many VMs simultaneously on a single server. If a VM instance crashes, it does not crash the server. The cloud monitor can record the event and restart the VM. User applications running in different VMs on the same server are largely unaware of each other. (A user may notice another VM when they impact the performance or response of their VM.)

We provide in chapter 5 detailed instructions on how to deploy VMs on the Amazon and Azure public clouds, and on OpenStack private clouds.

Containers are similar to VMs but are based on a different technology and serve a slightly different purpose. Rather than run a full OS, a container is layered on top of the host OS and uses that OS's resources in a clever way. Containers allow you to package up an application and all of its library dependencies and data

into a single, easy-to-manage unit. When you launch the container, the application can be configured to start up, go through its initialization, and be running in seconds. For example, you can run a web server in one container and a database server in another; these two containers can discover each other and communicate as needed. Or, if you have a special simulation program in a container, you can start multiple instances of the container on the same host.

Containers have the advantage of being extremely lightweight. Once you have downloaded a container to a host, you can start it and the application(s) that it contains quasi-instantly. Part of the reason for this speed is that a container instance can share libraries with other container instances. VMs, because they are complete OS instances, can take a few minutes to start up. You can run many more containers on a single host machine than you can effectively run the same number of VMs. Figure 4.1 illustrates the difference between the software stack of a server running multiple VMs versus a server running a single OS and multiple containers on a typical server in a cloud.

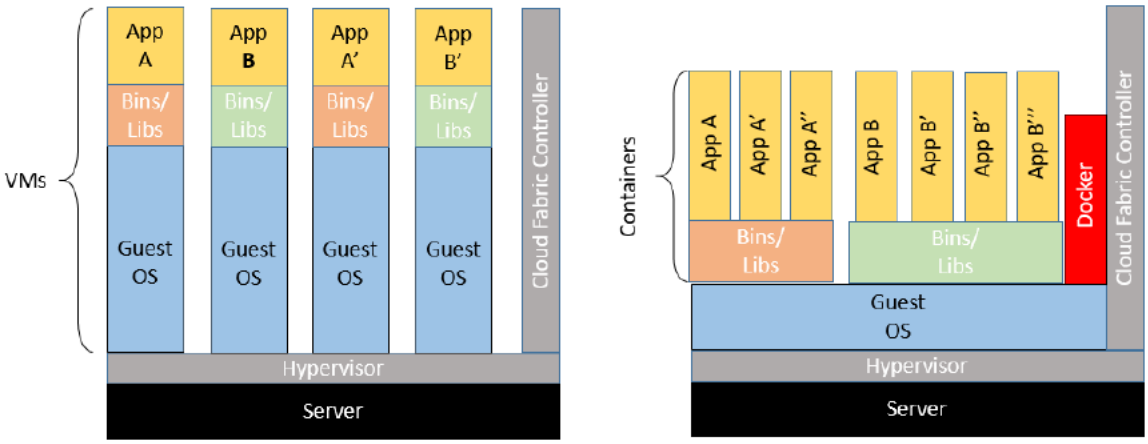

Figure 4.1: Virtual machines vs. containers on a typical cloud server.

Building a container to run a single application is simple compared with the task of customizing a VM to run a single application. All you need to do is create a script that identifies the needed libraries, source files, and data. You can then run the script on your laptop to test the container, before uploading the container to a repository, from where it can be downloaded to any cloud. Importantly, containers are completely portable across different clouds. In general, VM images cannot be ported from one cloud framework to another.

Containers also have downsides. The most serious issue is security. Because containers share the same host OS instance, two containers running on the same

Table 4.1: Virtual machines and containers, compared.

Virtual machines	Containers
Heavyweight	Lightweight
Fully isolated; hence more secure	Process-level isolation; hence less secure
No automation for configuration	Script-driven configuration
Slow deployment	Rapid deployment
Easy port and IP address mapping	More abstract port and IP mappings
Custom images not portable across clouds	Completely portable

host are less isolated than two VMs running on that host. Managing the network ports and IP addresses used by containers can be slightly more confusing than when working with VMs. Furthermore, containers are often run on top of VMs, which can exacerbate the confusion.

In chapter 6, we work though examples of using containers in some detail, describing in particular the Docker container system `docker.com`, how to run containers, and how to create your own containers.

Table 4.1 lists some of the the pros and cons of the virtual machine and container approaches. Needless to say, both technologies are evolving rapidly.

4.2 Advanced Computing Services

Cloud vendors such as Amazon, Microsoft, and Google have many additional services to help your research, including special data analysis clusters, tools to handle massive streams of events from instruments, and special machine learning tools. We discuss these services in part III of the book.

A common issue of concern to scientists and engineers is **scale**. VMs and containers are a great way to virtualize a single machine image. However, many scientific applications require multiple machines to process many data or to perform a complex simulation. You may already know how to run parallel programs on clusters of machines and you now want to know whether you can run those same programs on the cloud. The answer depends on the specifics of the application. Most high-performance parallel applications are based on the **Message Passing Interface** (MPI) standard [147]. As we describe in chapter 7, Amazon and Azure provide an extensive set of tools for building Linux MPI clusters.

Cloud users can also exploit parallelism in other ways. For example, **many task** (MT) parallelism [221] is used to tackle problems in which you have hundreds of similar tasks to run, each(largely) independent of the others. Another method is

called **MapReduce** [108], made popular by the Hadoop computational framework [258]. MapReduce is related to a style of parallel computing known as **bulk synchronous parallelism** (BSP) [139]. We cover these topics, also, in chapter 7.

A compelling feature of the cloud is that it provides many ways to create highly scaled applications that are also interactive. The **Spark** system [265], originally developed at University of California Berkeley, is more flexible than Hadoop and is a form of BSP computing that can be used interactively from Jupyter. Google has released a service called **Cloud Datalab**, based on Jupyter, for interactive control of its data analytics cloud. The Microsoft Cloud Business Intelligence (**Cloud BI**) tool supports interactive access to data queries and visualization. We discuss these tools in chapter 8.

Managing your cloud computing resources can become complicated when you need to scale beyond a few VMs or containers. Keeping track of many processes spread over many cloud VMs is not easy. Fortunately, several new tools have been adopted by the public clouds to help with this challenge. For managing large numbers of containers, you can use the Docker **Swarm** tools `docker.com/products/docker-swarm` and Google's **Kubernetes container management** [26, 79] (which Google uses for its own container management). Many people use the venerable **HTCondor** system [243] to manage many task parallel computation. (HTCondor is used in the Globus Genomics system that we describe in chapter 11.) **Mesos** [154] provides another distributed operating system with a web interface that allows you to manage many applications in the cloud at the same time. All of these systems are already available, or can easily be deployed, on cloud platforms. We describe them in chapter 7.

One other computation service programming model that is common in cloud computing is **dataflow**. This model plays a significant role in the analysis of streaming data, as discussed in chapter 9.

4.3 Serverless Computing

An interesting recent trend in cloud computing is the introduction of **serverless computing** as a new paradigm for service delivery. As we show in the chapters ahead, computation and data analysis can be deployed in the cloud via a range of special services. In the majority of cases, the user must deploy VMs, either directly or indirectly, to support these capabilities. Doing so takes time, and the user is responsible for deleting the VMs when they are no longer needed. At times. however, this overhead is not acceptable, such as when you want an action to take place in response to a relatively rare event. The cost of keeping a VM running

continuously so that a program can wait for the event may be unacceptably high.

Serverless computing is similar to the old Unix concepts of a daemon and cron jobs, whereby a program is managed by the operating system and is executed only when specific conditions arise. In serverless computing, the user provides a simple function to be executed, again under certain conditions. For example, the user may wish to perform some bookkeeping when a new file is created in a cloud repository or to receive a notification when an important event occurs. The cloud provider keeps a set of machines running to execute these functions on the user's behalf; the user is charged only for the execution of the task, not for maintaining the servers. We return to this topic in chapters 9 and 18.

4.4 Pros and Cons of Public Cloud Computing

Public cloud computing has both pros and cons. Important advantages include the following.

- *Cost*: If you need a resource for only a few hours or days, the cloud is much cheaper than buying a new machine.
- *Scalability*: You are starting a new lab and want to start with a small number of servers, but as your research group grows, you want to be able to expand easily without the hassle of managing your own racks of servers.
- *Access*: A researcher in a small university or an engineer in a small company may not even have a computer room or a lab with room for racks of machines. The cloud may be the only choice.
- *Configurability*: For many scientific disciplines, you can get complete virtual machines or containers with all the standard software you need pre-installed.
- *Variety*: Public cloud systems provide access to a growing diversity of computer systems. Amazon and Azure each provide dozens of machine configurations, ranging from a single core with a gigabyte of memory to multicore systems with GPU accelerators and massive amounts of memory.
- *Security*: Commercial cloud providers have excellent security. They also make it easy to create a virtual network that integrates cloud resources into your network.
- *Upgradeability*: Cloud hardware is constantly upgraded. Hardware that you buy is out of date the day that it is delivered, and becomes obsolete quickly.

- *Simplicity*: You can manage your cloud resources from a web portal that is easy to navigate. Managing your own private cluster may require sophisticated system administration skills.

Disadvantages of computing as a service include the following.

- *Cost*: You pay for public cloud by the hour and the byte. Computing the total cost of ownership of a cluster of machines housed in a university lab or data center is not easy. In many environments, power and system administration are subsidized by the institution. If you need to pay only for hardware, then running your own cluster may be cheaper than renting the same services in the cloud. Another accounting oddity, perhaps peculiar to the U.S., is that some universities charge overhead on funds obtained from a federal funding source for public cloud but not when equivalent funds are used for hardware purchases.

  Academic researchers may also have the option of accessing a national facility such as Jetstream, Chameleon, or the European science cloud. The cost here is the work involved in writing a proposal.

- *Variety*: The cloud does not provide every type of computing that you may require, at least not today. In particular, it is not a proper substitute for a large supercomputer. As we show in chapter 7, both Amazon and Azure support the allocation of fairly sophisticated high performance computing (HPC) clusters. However, these clusters are not at the scale or performance level of the top 500 supercomputers.

- *Security*: Your research concerns highly sensitive data, such as medical information, that cannot be moved outside your firewall. As stated above, there are ways to extend your network into the cloud, and the cloud providers provide HIPAA-compliant facilities. However, the paperwork involved in getting approval to use these solutions may be daunting.

- *Dependence* on one cloud vendor (often referred to as vendor lock-in). This situation is changing, however. As the public clouds converge in many of their standard offerings and compete on price, moving applications between cloud vendors has become easier.

Another way to think about the cost of computing is to weigh the cost of computing for an *individual*, who may use only a few hours of computing per week, versus for an entire *institution*, such as a university or large research center, which

may in aggregate use many tens of thousands of hours per week. One sensible solution is for the institution to sign a long-term contract with a public cloud provider that allows the institution to pick up the tab for its researchers. There are several ways to make this approach economically attractive to both the institution and the cloud provider. For example, universities can negotiate cloud access as part of a package deal involving software licenses or institutional data back-up.

An institution may also have the resources and expertise required to build a private mini-data center, in which it can deploy an OpenStack cloud that it makes available to all of its employees. Depending on the institution and its workloads, this approach may be more cost effective than others. It may also be required for data protection reasons. This approach leaves open the possibility of a hybrid solution, in which you spill over to the public cloud when the private cloud is saturated. This is often referred to as **cloud bursting**. This hybrid solution has become a common model for many large corporations and is well supported by cloud providers such as Microsoft and IBM. Aristotle `federatedcloud.org` is an example of an academic cloud that supports cloud bursting.

4.5 Summary

Clouds can provide computing resources as a service, at scales ranging from a single virtual machine with one virtual core and a few gigabytes of memory to full HPC clusters. The type and scale of service you use depend on the nature of the application. The following are examples.

- You may need only an extra Linux or Windows server to run an application, and you do not want to saddle your laptop with the extra load. In this case, a single VM running on a multicore server with a large memory is all you may need. You deploy it in a few minutes and remove it when you are done.

- You want to run a standard application such as a database to share with other users. In this case, the easiest solution is to run a containerized instance of your favorite database on a VM designed to run containers or on a dedicated container hosting service.

- You have an MPI-based parallel program that does not need thousands of processors to get the job done in a reasonable amount of time. In this case the public clouds have simple tools to spin up a HPC cluster that you can use for your job.

- You have a thousand tasks to run that produce data you need to analyze interactively. This may be a job for Spark with a Jupyter front end or Hadoop if it can be rendered as a MapReduce computation.
- If the thousand-task computations are more loosely coupled and can be widely distributed, then HTCondor is a natural choice.
- If you are processing large streams of data from external sources, a dataflow stream processing tool may be the best solution.

Other considerations, notably cost and security, can also enter into your choice of resource and system. We discuss security in chapter 15.

4.6 Resources

Each of the major public cloud vendors provides excellent tutorials for using its computing services. In addition, entire books have been written on each topic covered in this chapter. Four that we particularly like are: *Programming AWS EC2* by Jurg van Vliet and Flavia Paganelli [251]; *Amazon Web Services in Action*, by Andreas Wittig and Michael Wittig [263]; *Programming Google App Engine with Python*, by Dan Sanderson [231]; and *Microservices, IoT and Azure: Leveraging DevOps and Microservice Architecture to deliver SaaS Solutions* by Bob Familiar [119]. We point to additional resources in the chapters that follow.