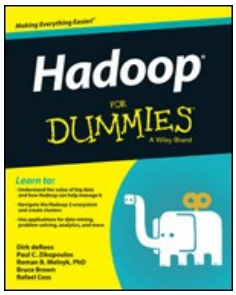

Hadoop for Dummies

by Dirk deRoos et al.

Chapter 16: Deploying Hadoop

In This Chapter

- Examining the components that comprise a Hadoop cluster
- Designing the Hadoop cluster components
- Reviewing Hadoop deployment form factors
- Sizing a Hadoop cluster

At its core, Hadoop is a system for storing and processing data at a massive scale using a cluster of many individual compute nodes. In this chapter, we describe the tasks involved in building a Hadoop cluster, all the way from the hardware components in the compute nodes to different cluster configuration patterns, to how to appropriately size clusters. In at least one way, Hadoop is no different from many other IT systems: If you don't design your cluster to match your business requirements, you get bad results.

Working with Hadoop Cluster Components

While you're getting your feet wet with Hadoop, you're likely to limit yourself to using a pseudo-distributed cluster running in a virtual machine on a personal computer. Though this environment is a good one for testing and learning, it's obviously inappropriate for production-level performance and scalability. In this section, we talk about what's involved in advancing to the next step. More specifically, we describe what a distributed cluster looks like, where multiple nodes are dedicated to data storage and processing.

Distributed Hadoop clusters normally follow the model shown in Figure 16-1. Redundancy is critical in avoiding single points of failure, so you see two switches and three master nodes. (We explain the latter number later in this chapter, in the section "Master nodes.") You also see two edge nodes for client applications and connectivity to resources outside the cluster, and a sufficient number of slave nodes to store your data sets. You see variations on this model when using multiple racks or processing techniques that need additional master nodes (HBase with its region servers, for example). We get into the specifics in later sections.

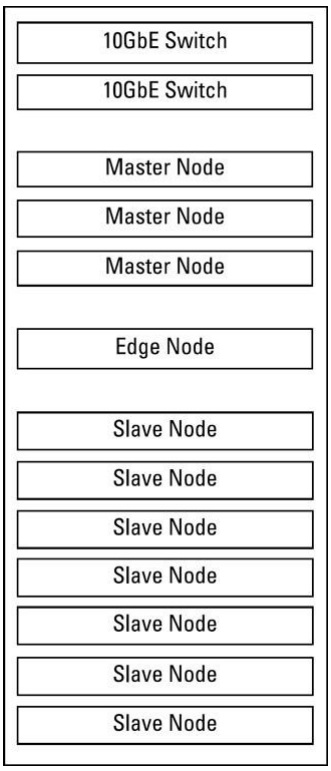

Figure 16-1: Typical components in a Hadoop cluster

Rack Considerations

A core principle of Hadoop is scaling out with additional slave nodes to meet increasing data-storage and -processing demands. In a scale-out model, you must carefully consider cluster design because dozens, and even hundreds, of slave nodes will ultimately need to be racked, powered, networked, and cooled.

Server form Factors

One of the first choices that IT architects will face when designing a Hadoop cluster is which of the following two form factors to use for Hadoop nodes:

- **Blade server**: Designed for maximum density, you can cram as many of these babies into one rack as possible. Blade servers fit into blade enclosures, which have many standard server components, like dedicated storage, networking, power, and cooling. These components are shared among the blade servers, which means that each individual blade server can be much smaller.

  Blade servers are an appealing choice on the surface because you could take a standard rack and deploy between 40 and 50 of these blade servers. The problem with using blades for Hadoop deployments is that they rely on certain shared components, which isn't in line with Hadoop's shared-nothing architecture, where each of the slave nodes are self-contained and have their own dedicated resources. More importantly, blades have little room for locally attached storage, often having no more than two or three drive bays. This is a non-starter for Hadoop, since slave nodes need much more dedicated storage capacity.

- **Rack server**: Complete servers with no shared components and room for hardware expansion, rack servers are the true choice for Hadoop because they're nicely self-contained. A rack server that's appropriately configured for being a Hadoop slave node typically occupies two RU, so you can fit 20 of them in a standard rack.

Cost of Ownership

When choosing and designing a slave node, your most important considerations are typically the initial procurement costs and the storage volume. However, the cost of ownership is also important. It's a fine balancing act, however, because choices affecting procurement cost, power consumption, cooling, hardware performance, and density are often in opposition. In the name of helping you make good choices, we offer some (quite specific) advice:

- **Reserve redundant power supplies for the master nodes**. Having redundant power supplies for slave nodes is overkill — a power supply failure in a slave node wouldn't greatly affect the cluster. However, having redundant power supplies on all slave nodes would increase power consumption and generate more heat.
- **Choose middle-of-the-road clock speeds for** slave **node CPUs**. CPUs with higher clock speeds not only cost more but also use more power and generate far more heat.
- **Choose rack servers that are designed for Hadoop**. With the rising popularity of Hadoop, all major hardware vendors now offer rack servers that are ideal slave nodes, with 12 to 20 drive bays for locally attached storage. Rack servers designed to work as Hadoop slave nodes are typically too big to fit into a form factor of one RU, but taking up two RUs can result in wasted space. For the more efficient use of space, certain hardware vendors have released rack servers that cram multiple slave nodes into a single chassis. As an example, in this compressed form, a standard rack can have as many as 27 slave nodes (even with network switches), where each slave node has room for 15 disk drives for HDFS. The upshot of this arrangement is much higher density and better use of space in the data center.

Master Nodes

The master nodes host the various storage and processing management services, described in this list, for the entire Hadoop cluster:

- **NameNode**: Manages HDFS storage. To ensure high availability, you have both an active NameNode and a standby NameNode. Each runs on its own, dedicated master node.
- **Checkpoint node (or backup node)**: Provides *checkpointing* services for the NameNode. This involves reading the NameNode's edit log for changes to files in HDFS (new, deleted, and appended files) since the last checkpoint, and applying them to the NameNode's master file that maps files to data blocks. In addition, the Backup Node keeps a copy of the file system namespace in memory and keeps it in sync with the state of the NameNode. For high availability deployments, do not use a checkpoint node or backup node — use a Standby NameNode instead. In addition to being an active standby for the NameNode, the Standby NameNode maintains the checkpointing services and keeps an up-to-date copy of the file system namespace in memory.
- **JournalNode**: Receives edit log modifications indicating changes to files in HDFS from the NameNode. At least three JournalNode services (and it's always an odd number) must be running in a cluster, and they're lightweight enough that they can be colocated with other services on the master nodes.
- **Resource Manager**: Oversees the scheduling of application tasks and management of the Hadoop cluster's resources. This service is the heart of YARN.
- **JobTracker**: For Hadoop 1 servers, handles cluster resource management and scheduling. With YARN, the JobTracker is obsolete and isn't used. We mention it because a number of Hadoop deployments still haven't migrated to Hadoop 2 and YARN.
- **HMaster**: Monitors the HBase region servers and handles all metadata changes. To ensure high availability, be sure to use a second HMaster instance. The HMaster service is lightweight enough to be colocated with other services on the master nodes. In Hadoop 1, instances of the HMaster service run on master nodes. In Hadoop 2, with Hoya (HBase on Yarn), HMaster instances run in containers on slave nodes.
- **Zookeeper**: Coordinates distributed components and provides mechanisms to keep them in sync. Zookeeper is used to detect the failure of the NameNode and elect a new NameNode. It's also used with HBase to manage the states of the HMaster and the RegionServers. As with the JournalNode, you need at least three instances of Zookeeper nodes (and always an odd number), and they're lightweight enough to be colocated with other services on the master nodes.

Figure 16-2 shows an example of how Hadoop 2 services can be deployed.

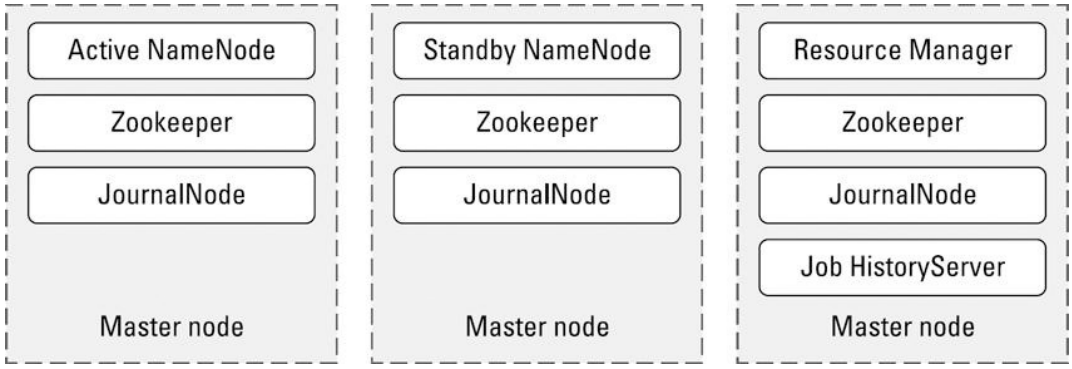

Figure 16-2: Hadoop 2 services deployed on master nodes

Here we've got three master nodes (with the same hardware), where the key services Active NameNode, Standby NameNode, and Resource Manager each have their own server. There are JournalNode and Zookeeper services running on each server as well, but as we mentioned earlier, these are lightweight and won't be a source of resource contention with the NameNode and Resource Manager services.

Figure 16-3 shows what master nodes look like for Hadoop 1 deployments.

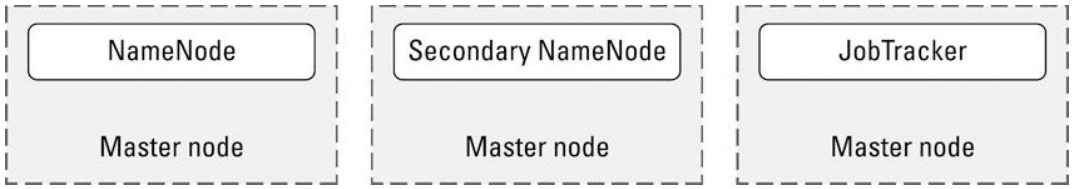

Figure 16-3: Hadoop 1 services deployed on master nodes

The principles are the same for Hadoop 1, where you need a dedicated master node for the NameNode, Secondary NameNode, and JobTracker services.

If you plan to use HBase with Hoya in Hadoop 2, you don't need any additional services. For Hadoop 1 deployments using HBase, see Figure 16-4 for the deployment of services on the Hadoop cluster's master nodes.

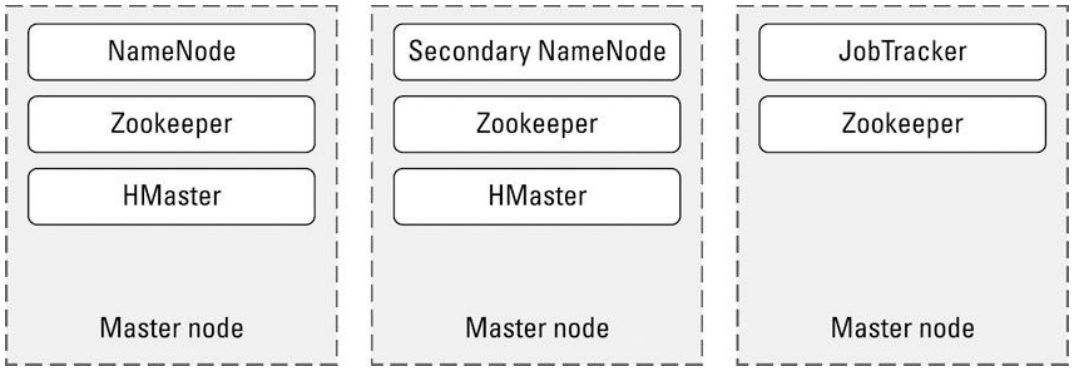

Figure 16-4: Hadoop 1 services deployed on master nodes, with HBase

There are two differences when comparing these master servers to the Hadoop 1 master servers without HBase support: here we need two HMaster services (one to coordinate HBase, and one to act as a standby) and Zookeeper services on all three master nodes to handle failover. If you intend to use your Hadoop 1 cluster only for HBase, you can do without the JobTracker service, since HBase does not depend on the Hadoop 1 MapReduce infrastructure.

REMEMBER When people talk about hardware for Hadoop, they generally emphasize the use of *commodity* components — the inexpensive ones, in other words. We don't recommend taking that route for master nodes. Because you have to plunk down for only a few master nodes (typically, three or four), you aren't hit by multiplying costs if, for example, you decide to use expensive hard disk drives. Keep in mind that, without master nodes, there is no Hadoop cluster. Master nodes serve a mission-critical function, and even though you need redundancy, you should design them with high availability and resiliency in mind.

Recommended Storage

For Hadoop master nodes, regardless of the number of slave nodes or uses of the cluster, the storage characteristics are consistent. Use four

900GB SAS drives, along with a RAID HDD controller configured for RAID 1+0. SAS drives are more expensive than SATA drives, and have lower storage capacity, but they are faster and much more reliable. Deploying your SAS drives as a RAID array ensures that the Hadoop management services have a redundant store for their mission-critical data. This gives you enough stable, fast, and redundant storage to support the management of your Hadoop cluster.

Recommended Processors

At the time of this writing, most reference architectures recommend using motherboards with two CPU sockets, each with six or eight cores. The Intel Ivy Bridge architecture is commonly used.

Recommended Memory

Memory requirements vary considerably depending on the scale of a Hadoop cluster. Memory is a critical factor for Hadoop master nodes because the active and standby NameNode servers rely heavily on RAM to manage HDFS. As such, we recommend the use of error-correcting memory (ECC) for Hadoop master nodes. Typically, master nodes need between 64GB and 128GB of RAM.

The NameNode memory requirement is a direct function of the number of file blocks stored in HDFS. As a rule, the NameNode uses roughly 1GB of RAM per million HDFS blocks. (Remember that files are broken down into individual blocks and replicated so that you have three copies of each block.)

The memory demands of Resource Manager, HMaster, Zookeeper, and JournalNode servers are considerably less than for the NameNode server. However, it's good practice to size the master nodes in a consistent fashion so that they're interchangeable in case of hardware failure.

Recommended Networking

Fast communication is vital for the services on master nodes, so we recommend using a pair of bonded 10GbE connections. (In case networking jargon is new to you, GbE stands for GigaBit Ethernet.) This bonded pair provides redundancy, but also doubles throughput to 20GbE. For smaller clusters (for instance, less than 50 nodes) you could get away with using 1 GbE connectors.

Slave Nodes

In a Hadoop universe, slave nodes are where Hadoop data is stored and where data processing takes place. The following services enable slave nodes to store and process data:

- **NodeManager**: Coordinates the resources for an individual slave node and reports back to the Resource Manager.
- **ApplicationMaster**: Tracks the progress of all the tasks running on the Hadoop cluster for a specific application. For each client application, the Resource Manager deploys an instance of the ApplicationMaster service in a container on a slave node. (Remember that any node running the NodeManager service is visible to the Resource Manager.)
- **Container**: A collection of all the resources needed to run individual tasks for an application. When an application is running on the cluster, the Resource Manager schedules the tasks for the application to run as container services on the cluster's slave nodes.
- **TaskTracker**: Manages the individual map and reduce tasks executing on a slave node for Hadoop 1 clusters. In Hadoop 2, this service is obsolete and has been replaced by YARN services.
- **DataNode**: An HDFS service that enables the NameNode to store blocks on the slave node.
- **RegionServer**: Stores data for the HBase system. In Hadoop 2, HBase uses Hoya, which enables RegionServer instances to be run in containers.

Figure 16-5 shows the services deployed on Hadoop 2 slave nodes.

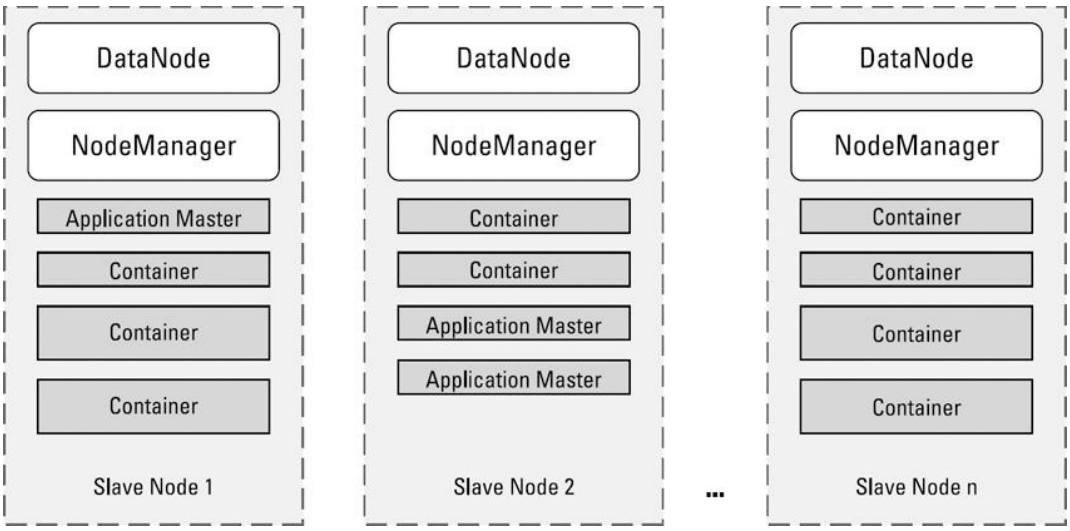

Figure 16-5: Services deployed on Hadoop 2 slave nodes

Here, each slave node is always running a DataNode instance (which enables HDFS to store and retrieve data blocks on the slave node) and a NodeManager instance (which enables the Resource Manager to assign application tasks to the slave node for processing). The container processes are individual tasks for applications that are running on the cluster. Each running application has a dedicated ApplicationMaster task, which also runs in a container, and tracks the execution of all the tasks executing on the cluster until the application is finished.

With HBase on Hadoop 2, the container model is still followed, as we can see in Figure 16-6.

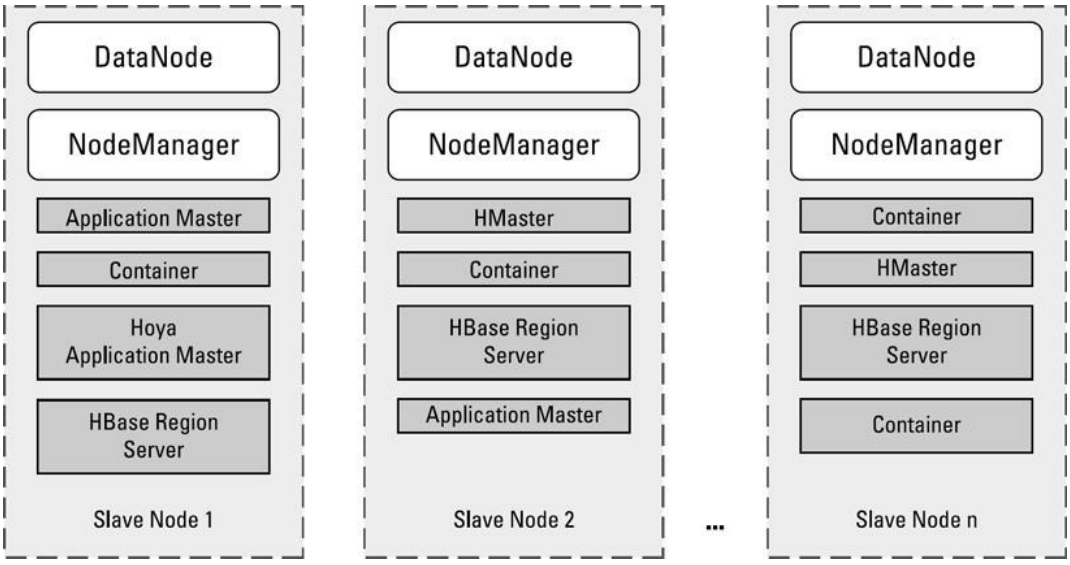

Figure 16-6: Services deployed on Hadoop 2 slave nodes, including HBase

HBase on Hadoop 2 is initiated by the Hoya Application Master, which requests containers for the HMaster services. (You need multiple HMaster services for redundancy.) The Hoya Application Master also requests resources for RegionServers, which likewise run in special containers.

Figure 16-7 shows the services deployed on Hadoop 1 slave nodes.

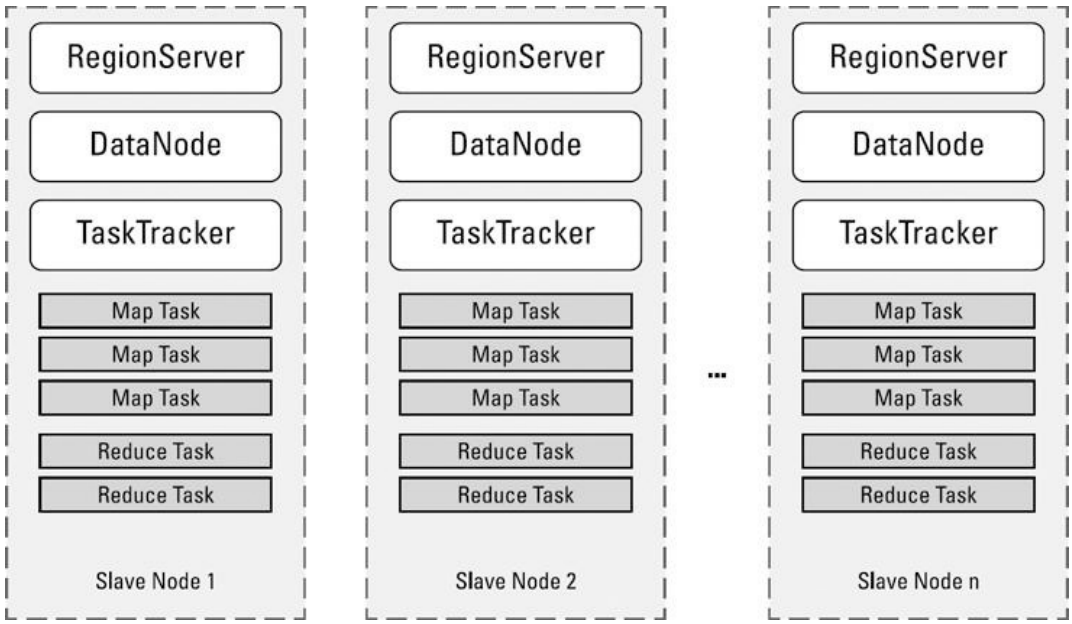

Figure 16-7: Services deployed on Hadoop 1 slave nodes

For Hadoop 1, each slave node is always running a DataNode instance (which enables HDFS to store and retrieve data blocks on the slave node) and a TaskTracker instance (which enables the JobTracker to assign map and reduce tasks to the slave node for processing). Slave nodes have a fixed number of map slots and reduce slots for the execution of map and reduce tasks respectively. If your cluster is running HBase, a number of your slave nodes will need to run a RegionServer service. The more data you store in HBase, the more RegionServer instances you'll need.

REMEMBER The hardware criteria for slave nodes are rather different from those for master nodes (refer to the "Master nodes" section earlier in this chapter); in fact, the criteria don't match those found in traditional hardware reference architectures for data servers. Much of the buzz surrounding Hadoop is due to the use of commodity hardware in the design criteria of Hadoop clusters, but keep in mind that *commodity* hardware does not refer to consumer-grade hardware. Hadoop slave nodes still require enterprise-grade hardware, but at the lower end of the cost spectrum, especially for storage.

Recommended Storage

Enterprise storage is normally configured as a RAID array, but with Hadoop, the optimal configuration is the almost comically simple JBOD — Just a Bunch Of Disks. That's right — JBOD is just a bunch of disks that are directly (and independently) connected with the slave node's motherboard. For Hadoop slave nodes, you need two sets of hard disk drives: one set for the operating system and the other set for HDFS. Two 500GB SATA drives are sufficient for the operating system. Most hardware manufacturers have released rack servers specially designed for Hadoop, which enable individual slave nodes to house an additional 12 to20 drives for dedicated HDFS storage. Be sure to choose large form-factor (LFF) drives (3½ inches) for HDFS storage, because they have a higher capacity and are less expensive. At the time of this writing, 3TB SATA LFF drives are the most practical and cost-effective choice, though 4TB SATA drives will likely become a common choice.

Keep in mind the following information about Hadoop slave node storage:

- Twelve 3TB drives provide 36 terabytes of raw storage for your Hadoop cluster, which enables you to store 12 terabytes of data in HDFS, given the default replication factor of 3.
- For efficient and cost-effective performance, ensure a 1:1 ratio of CPU cores to drives dedicated to HDFS.
- Though the drives used for slave nodes are in the more economical commodity class, it's practical to connect them with a faster and more stable controller. Because many Hadoop workloads are I/O bound, we recommend using a SAS 6 gigabytes-per-second controller.

TIP Never use an operating system drive for HDFS, because it compromises performance.

Recommended Processors

At the time of this writing, dual-socket servers that have Intel Ivy Bridge processors that are clocked between 2 and 2.5 GHz represent the best balance of performance and cost for slave nodes. And, as we mentioned in the guidelines for storage earlier in the "Recommended storage" section, the number of drives you choose should be consistent with the number of CPU cores present — we recommend you should maintain a 1:1 ratio. If you're using 12 drives for HDFS, for example, you use two 6-core CPUs, and if you're using 16 drives, you use two 8-core CPUs. This configuration is practical for many applications, but if you're going to run processor-intensive workloads and you need fast performance, you can maintain a higher ratio of CPU cores to HDFS drives, for example 3 CPU cores for every two drives. As with any performance optimization exercise, at some point you will hit a bottleneck. For example, if you increase the ratio of CPU cores to HDFS drives too much, you will find your applications spending most of their time waiting for disk read or write operations.

Recommended Memory

For most workloads, considering the nature of the processor and disk specifications given in the previous two sections, 48GB of RAM is sufficient for slave nodes. For maximum performance, however, you must fully populate the RAM channels for the slave node processors. For example, a dual-core server with three RAM channels per processor will have 48GB of RAM divided between six 8GB memory modules (DIMMs).

TECHNICAL STUFF If you're not up on RAM channels, here's a quick primer. Most modern motherboard chipsets now use multi-channel memory. This enables the CPU to access its memory in parallel, which increases the data transfer speed by as many times as there are channels. For example, data transfers between the memory and CPU on servers with triple-channel memory architecture will be three times as fast as servers with single-channel memory architecture. The catch here is that for multi-channel memory to work well, each slot for the memory channel must be populated with an identical memory module.

For Hadoop clusters where you know that the workload will be memory intensive (for example, HBase deployments), we recommend doubling the number of DIMMs, for a total of 96GB of RAM per slave node (as per the preceding example, which is twelve 8GB DIMMs).

Recommended Networking

For slave nodes, we recommend a pair of bonded network connections, to provide redundancy *and* to double throughput. The deciding factor here is speed. If your cluster's slave nodes have 48GB or more dedicated to HDFS, we recommend 10GbE connections to be able to handle the data transfer demands that arise from dense storage. Otherwise, we recommend 1GbE connections.

REMEMBER A key concept in good cluster design is the separation of duties between master nodes and slave nodes. Their purposes are radically different, and their design patterns reflect this. For production clusters, do not give in to the temptation to add DataNode and NodeManager (or TaskTracker, for Hadoop 1) servers to your master nodes. Keep separate elements separate, in other words.

Edge Nodes

Edge nodes are the interface between the Hadoop cluster and the outside network. For this reason, they're sometimes referred to as *gateway* nodes. Most commonly, edge nodes are used to run client applications and cluster administration tools. They're also often used as staging areas for data being transferred into the Hadoop cluster. As such, Oozie, Pig, Sqoop, and management tools such as Hue and Ambari run well there. Figure 16-8 shows the processes you can run on Edge nodes.

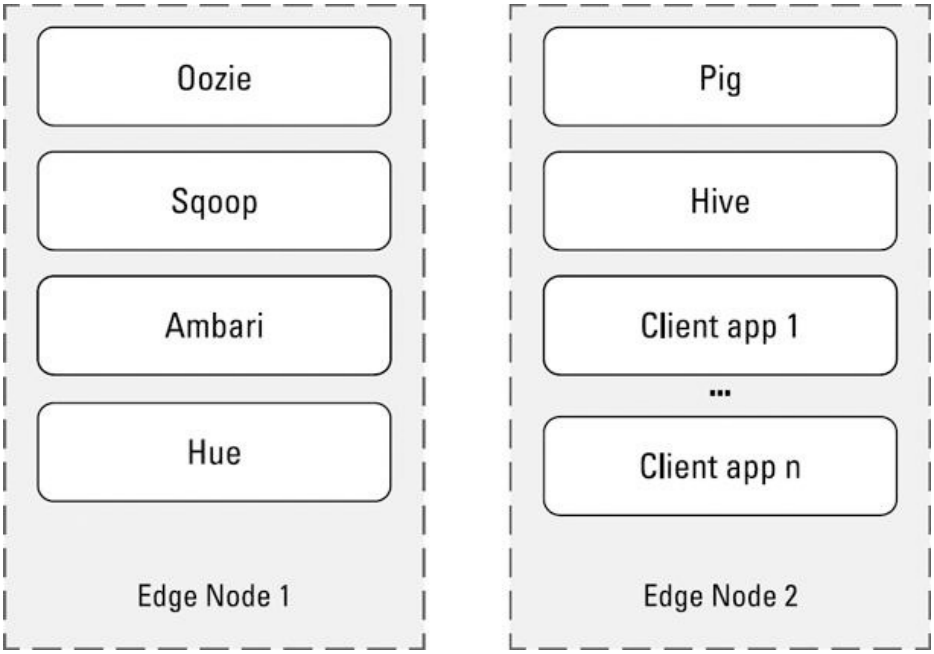

Figure 16-8: Services deployed on edge nodes

Edge nodes are often overlooked in Hadoop hardware architecture discussions. This situation is unfortunate because edge nodes serve an important purpose in a Hadoop cluster, and they have hardware requirements that are different from master nodes and slave nodes. In general, it's a good idea to minimize deployments of administration tools on master nodes and slave nodes to ensure that critical Hadoop services like the NameNode have as little competition for resources as possible.

WARNING! You should avoid placing a data transfer utility like Sqoop on anything but an edge node, as the high data transfer volumes could risk the ability of Hadoop services on the same node to communicate. The messages Hadoop services exchange are their lifeblood, so high latency means the whole node could be cut off from the cluster.

Figure 16-8 shows two edge nodes, but for many Hadoop clusters a single edge node would suffice. Additional edge nodes are most commonly needed when the volume of data being transferred in or out of the cluster is too much for a single server to handle.

Recommended Storage

For edge nodes in a Hadoop cluster, use enterprise class storage. For edge nodes focused on administration tools and running client applications, we recommend using four 900GB SAS drives, along with a RAID HDD controller configured for RAID 1+0.

Edge nodes oriented to ingesting data obviously need much more storage space, so you can add drives to the edge node. In this case, use LFF SAS drives because much higher capacities are available, as compared to smaller form-factor SAS drives.

Recommended Processors

A general-purpose edge node would be well served by a processor configuration similar to one used for slave nodes — specifically, a dual-socket server with Ivy Bridge processors clocked at between 2 and 2.5GHz.

Recommended Memory

For most workloads we see on edge nodes, 48GB of RAM is sufficient.

Recommended Networking

To enable communication between the outside network and the Hadoop cluster, edge nodes need to be multi-homed into the private subnet of the Hadoop cluster as well as into the corporate network.

TECHNICAL STUFF A multi-homed computer is one that has dedicated connections to multiple networks. This is a practical illustration of why edge nodes are perfectly suited for interaction with the world outside the Hadoop cluster. Keeping your Hadoop cluster in its own private subnet is an excellent practice, so these edge nodes serve as a controlled window inside the cluster.

For edge nodes that serve the purpose of running client applications or administration tools, we recommend two pairs of bonded 1GbE network connections: one pair to connect to the Hadoop cluster and another pair for the outside network.

Edge nodes oriented to handling high inbound and outbound data transfer rates will need two (or more) pairs of bonded 10GbE network connectors: one pair to connect to the Hadoop cluster and another pair for the outside network or specific data ingest sources.

Networking

As with any distributed system, networking can make or break a Hadoop cluster: Don't "go cheap." A great deal of chatter takes place between the master nodes and slave nodes in a Hadoop cluster that is essential in keeping the cluster running, so we definitely recommend enterprise-class switches.

For each rack in your cluster, you need two top-of-rack (ToR) switches, for both redundancy and performance. We recommend using 10GbE for ToR switches.

TECHNICAL STUFF ToR switches are network switches that connect all the computers in a rack together. You normally see them at the very top of a rack, which is why people say "top-of-rack." An alternative networking approach is to use end-of-row (EoR) switches but, we don't see this very often. The ToR approach is simpler from a networking perspective for growing clusters. For example, adding slave nodes and additional racks is far easier with ToR switches than EoR.

When you have more than three racks, you need at least two core switches (again, primarily for redundancy, but also for performance). These core switches handle massive amounts of traffic, so 40GbE is a necessity.

TIP If you're building or expanding a cluster to span multiple racks, we strongly recommend engaging networking experts who are familiar with Hadoop, your future growth plans, and your workload. Bad networking can severely hamper performance, but it can also make future growth painful and expensive.

Hadoop Cluster Configurations

Many of the decisions you need to make in terms of the composition of racks and networking are dependent on the scale of your Hadoop cluster. It has three main permutations, as discussed in the following three sections.

Small

A single-rack deployment is an ideal starting point for a Hadoop cluster, as shown in Figure 16-9.

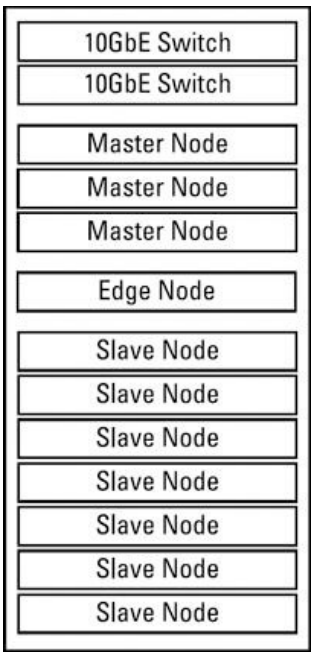

Figure 16-9: Single-rack Hadoop deployment

Here, the cluster is fairly self-contained, but because it still has relatively few slave nodes, the true benefits of Hadoop's resiliency aren't yet apparent.

Medium

A medium-size cluster has multiple racks, where the three master nodes are distributed across the racks, as shown in Figure 16-10.

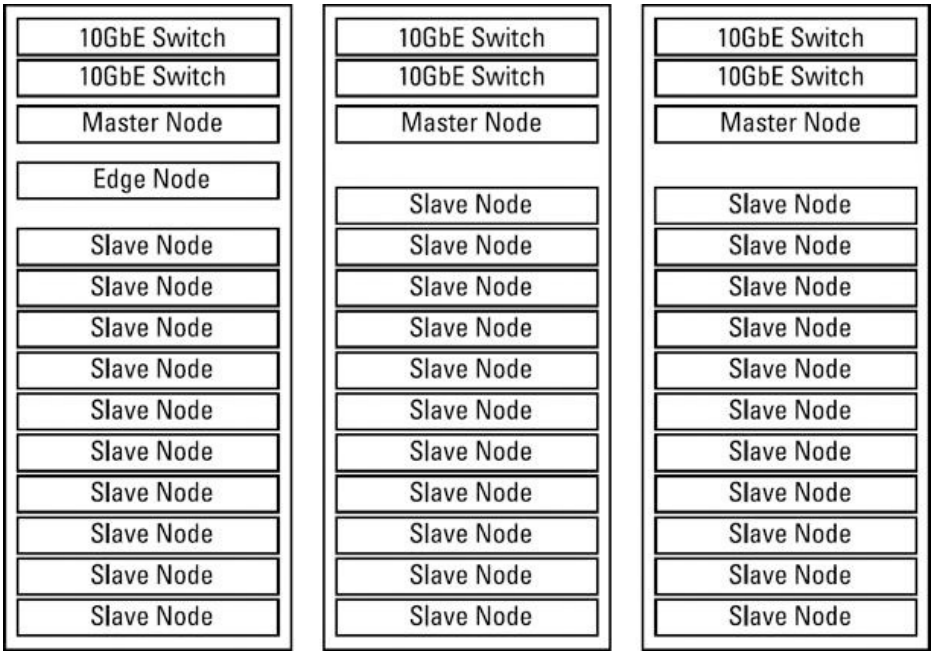

Figure 16-10: Three-rack Hadoop deployment

Hadoop's resiliency is starting to become apparent: Even if an entire rack were to fail (for example, both ToR switches in a single rack), the cluster would still function, albeit at a lower level of performance. A slave node failure would barely be noticeable.

Large

In larger clusters with many racks, like the example shown in Figure 16-11, the networking architecture required is pretty sophisticated.

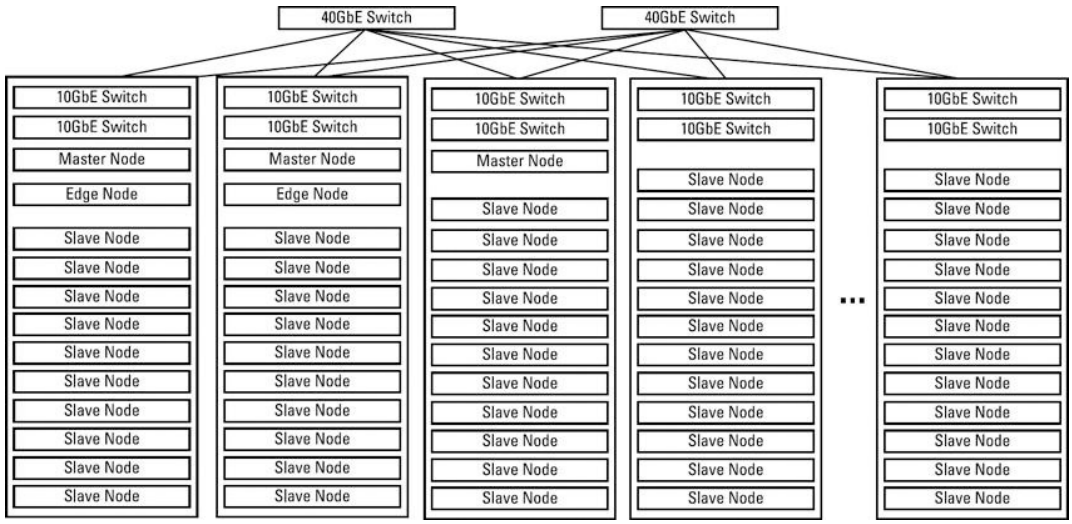

Figure 16-11: Large-scale Hadoop deployment

Regardless of how many racks Hadoop clusters expand to, the slave nodes from any rack need to be able to efficiently "talk" to any master node.

As the number of slave nodes increases to the point where you have more than three racks, additional racks are composed only of slave nodes, aside from the ToR switches. If you're using HBase heavily on your cluster, you may add master nodes to host additional HMaster and Zookeeper services. If you graduate to a truly massive scale, where you have hundreds of slave nodes, you may need to use the HDFS federation capabilities so that large portions of your data sets are managed by different NameNode services. (For more information on HDFS federation, see Chapter 4.) For every additional Active NameNode, you will need a corresponding Standby NameNode and two master nodes to host these servers. With HDFS federation, the sky is truly the limit in terms of how far you can scale out your clusters.

Alternate Deployment Form Factors

Though Hadoop works best when it's installed on a physical computer, where the processing has direct access to dedicated storage and networking, Hadoop has alternative deployments. And though they are less efficient than the dedicated hardware we describe earlier in this chapter, in certain cases alternatives are worthwhile options.

Virtualized Servers

A major trend in IT centers over the past decade is virtualization, where a large server can host several "virtual machines" which look and act like single machines. In place of dedicated hardware, an organization's entire set of applications and repositories is deployed on virtualized hardware. This approach has many advantages: The centralization of IT simplifies maintenance, IT investment is maximized because of fewer unused CPU cycles, and the overall hardware footprint is lower, resulting in a lower total cost of ownership.

Organizations in which IT deployments are entirely virtualized sometimes mandate that every new application follow this model. Though Hadoop can be deployed in this manner, essentially as a virtual cluster (with virtual master nodes and virtual slave nodes), performance suffers, partially because for most virtualized environments, storage is SAN-based and isn't locally attached. Because Hadoop is designed to work best when all available CPU cores are able to have fast access to independently spinning disks, a bottleneck is created as all the map and reduce tasks start processing data via the limited networking between the CPUs and the SAN. Since the degree of isolation between virtualized server resources is limited (virtual servers share resources with each other), Hadoop workloads can also be affected by other activity. When your virtual server's performance is affected by another server's workload, that's actually known in IT circles as a "noisy neighbor" problem!

Virtualized environments can be quite useful, though, in some cases. For example, if your organization needs to complete a one-time exploratory analysis of a large data set, you can easily create a temporary cluster in your virtualized environment. This method is often a faster way to gain internal approval than to endure the bureaucratic hassles of procuring new dedicated hardware.

As we experiment with Hadoop, we often run it on our laptop machines via a virtual machine (VM). Hadoop is extremely slow in this kind of environment, but if you're using small data sets, it's a valuable learning and testing tool.

Cloud Deployments

Variations of virtualized environments are cloud computing providers such as Amazon, Rackspace, and IBM SoftLayer. Most major public cloud providers now have MapReduce or Hadoop offerings available for use. Again, their performance is inferior to deploying your cluster on dedicated hardware, but it's improving. Cloud providers are making Hadoop-optimized environments available where slave nodes have locally attached storage and dedicated networking. Also, hypervisors are becoming far more efficient, with reduced overhead and latency.

Don't consider a cloud solution for long-term applications, because the cost of renting cloud computing resources is significantly higher than that of owning and maintaining a comparable system. With a cloud provider, you're paying for convenience and for being able to offload the overhead of provisioning hardware. However, the cloud is an ideal platform for testing, education, and one-time data processing tasks. We

use public cloud offerings often for proof-of-concept exercises in Hadoop, and we're able to easily conjure up a made-to-order cluster in a matter of minutes.

REMEMBER Aside from performance and cost considerations, you have regulatory considerations with public cloud deployments. If you have sensitive data, which must be stored either in-house or in-country, a public cloud deployment isn't an option. In cases like this, where you need the convenience of a cloud-based deployment, a private cloud is a good option, if it's available.

Sizing Your Hadoop Cluster

Sizing any data processing system is as much a science as it is an art. With Hadoop, you consider the same information as you would with a relational database, for example. Most significantly, you need to know how much data you have, estimate its expected growth rates, and establish a retention policy (how long to keep the data). The answers to these questions serve as your starting point, which is independent of any technology-related requirements.

After you determine how much data you need to store, you can start factoring in Hadoop-specific considerations. Suppose that you have a telecom company and you've established that you need 750 terabytes (TB) of storage space for its call detail record (CDR) log files. You retain these records to obey government regulations, but you can also analyze them to see churn patterns and monitor network health, for example. To determine how much storage space you need and, as a result, how many racks and slave nodes you need, you carry out your calculations with these factors in mind:

- **Replication**: The default replication factor for data in HDFS is 3. The 500 terabytes of CDR data for the telecom company in the example then turns into 1500 terabytes.
- **Swap space**: Any analysis or processing of the data by MapReduce needs an additional 25 percent of space to store any interim and final result sets. (The telecom company now needs 1875 terabytes of storage space.)
- **Compression**: The telecom company stores the CDRs in a compressed form, where the average compression ratio is expected to be 3:1. We now need 625 terabytes.
- **Number of slave nodes**: Assuming that each slave node has twelve 3TB drives dedicated to HDFS, each slave node has 36 terabytes of raw HDFS storage available, so the company needs 18 slave nodes.
- **Number of racks**: Because each slave node uses 2RU and the company in the example needs three master nodes (1RU apiece) and two ToR switches (1RU apiece), you need a total of 41RU. It's 1RU less than the total capacity of a standard rack, so a single rack is sufficient for this deployment. Regardless, no room remains for growth in this cluster, so it's prudent to buy a second rack (and two additional ToR switches) and divide the slave nodes between the two racks.
- **Testing**: Maintaining a test cluster that's a smaller scale representation of the production cluster is a standard practice. It doesn't have to be huge, but you want at least five data nodes so that you get an accurate representation of Hadoop's behavior. As with any test environment, it should be isolated on a different network from the production cluster.
- **Backup and disaster recovery**: Like any production system, the telecom company will also need to consider backup and disaster recovery requirements. This company could go as far as to create a mirror cluster to ensure they have a hot standby for their entire system. This is obviously the most expensive option, but is appropriate for environments where constant uptime is critical. At the least expensive end of the spectrum (beyond not backing up the data at all), the telecom company could regularly backup all data (including the data itself, applications, configuration files, and metadata) being stored in their production cluster to tape. With tape, the data is not immediately accessible, but it will enable a disaster recovery effort in the case that the entire production Hadoop cluster fails.

TIP As with your own personal computer, when the main hard disk drive fills with space, the system slows down considerably. Hadoop is no exception. Also, a hard drive performs better when it's less than 85 to 90 percent full. With this information in mind, if performance is important to you, you should bump up the swap-space factor from 25 to 33 percent.