Hadoop in cloud deployments

Overcome the challenges and maximize the advantages of a cloud-based Hadoop deployment

Martin C. Brown
Director of Documentation
Continuent

03 July 2014

Hadoop and the cloud seem to be ideal companions. They both contain flexible and distributed processing and storage, and they are married with a flexible instance system. They also enable you to grow and shrink your Hadoop cluster with your data and processing requirements. But this raises a variety of different management and scheduling issues. This article looks at all of these issues and more, as it describes the challenges and advantages of a cloud-based Hadoop deployment.

Understanding the scope of cloud deployments

The Hadoop system is a challenging environment to work with, but cloud deployments introduce additional levels of complexity because of the constraints (and freedoms) offered by the cloud environment.

For example, with Hadoop in the cloud, how do you cope with the variable cluster size and effective distribution of information? How do you effectively grow and shrink your cloud environment to cope with the Hadoop load you expect to process? And how do you schedule and control your jobs and processing to make the best of your cloud instances while they are available?

The advantages and disadvantages of cloud deployments can work for and against your use of Hadoop within these environments, depending upon the exact cloud service. Remember that the restrictions and limitations of a private cloud service differ enormously, compared to a public cloud service. If you are making use of your own VM environment or if you are using a solution such as OpenStack, you have much more flexibility to customize your services and functions.

To get the best out of Hadoop in the cloud, you first need to understand the cloud deployment solutions that exist and how they affect the Hadoop environment.
Service-based cloud deployments

InfoSphere BigInsights Quick Start Edition

InfoSphere BigInsights Quick Start Edition is a complimentary, downloadable version of InfoSphere BigInsights, IBM's Hadoop-based offering. Using Quick Start Edition, you can try out the features that IBM built to extend the value of open source Hadoop, like Big SQL, text analytics, and BigSheets. Guided learning is available to make your experience as smooth as possible including step-by-step, self-paced tutorials and videos to help you start putting Hadoop to work for you. With no time or data limit, you can experiment on your own time with large amounts of data. Watch the videos, follow the tutorials (PDF), and download BigInsights Quick Start Edition now.

Some cloud solutions are based entirely on a particular service into which data is loaded and processed. With IBM Bluemix® for example, you can configure a MapReduce service based on IBM InfoSphere® BigInsights™ that can process up to 20GB of information. But the size, configuration, and complexity of the Hadoop service is not configurable. Other service-based solutions offer the same sort of complexity.

You must select or configure the size of the service solution according to your requirements, because you might have no control over disk, I/O, CPU, or RAM availability.

The only way to determine your requirements is with testing. Add 25 percent or more to any calculations to provide room for spikes in usage and complexity.

(Virtual) Machine-based cloud deployments

Although the cloud environment is based entirely on machine or virtual machine-style deployments, Hadoop is installed in a virtual environment in much the same way as it would be on a physical machine. A range of configurable parameters can be configured, and these can change your options for cluster deployment.

In particular, consider the configuration for each node with regard to CPU, RAM, disk capacity, and disk I/O speed. Although it is true that a good Hadoop cluster deployment can hide minor differences between nodes, knowing the configuration can help you size your deployment for the standard support and for those times when you need to increase your speed and power.

As with any typical cloud-based deployment of a system or installation, the configuration depends on the following factors:

CPU

Either strict CPU count or arbitrary units. Unless you are deploying a YARN-based solution, consider deploying an identical configuration for all of the data and processing nodes within the cluster. This approach makes calculating the required size and capacity of the cluster easier. For YARN-based deployments, you can configure different nodes within the cluster to support and process different levels of CPU capacity. This method enables you, for example, to expand an existing cluster with a specific group of high-power CPU nodes for a given project.
RAM
All nodes should have at least 4GBs, but be aware that you might be limited by the available options. Also, keep in mind that you can gain performance benefits by ensuring there is some reserved space for file cache. Some solutions, such as HBase, can make use of additional memory.

Storage size
Ensure that you have separate volumes for the operating system and for the Hadoop storage, a practice that improves performance and makes extending the HDFS storage easier. Gauge your expected storage and standardize on a given size before scaling the cluster size. This method ensures a maximum even distribution across the cluster.

Disk I/O
The HDFS environment should limit the exposure of disk I/O, because all work is distributed across the cluster. However, do not unnecessarily constrict the disk I/O to the extent that a node is unable to access and process the data effectively. Within many cloud environments, the base or lowest disk I/O configuration can be so slow that it lowers the overall performance. Worse, if the disk I/O rate is not guaranteed to be above a particular level, you might experience a drop in performance in the middle of a processing job.

Network I/O
Hadoop requires a significant amount of network I/O to operate; each file is replicated at least twice in addition to the original write, and the data used during a MapReduce operation must be similarly transferred over the network. Within many cloud environments, network performance is limited and might be a limiting factor in your deployment.

Mixed-based cloud deployments
In some mixed-based cloud deployments, some elements are fixed and others are variable. In this situation you are able, within certain limits, to define specific machine capacities and simultaneously control the overall number of nodes. In these Hadoop cloud deployments, pick the right RAM and CPU combination, then scale your cluster to this configuration.

Expanding and shrinking Hadoop clusters
One of the most attractive aspects of the cloud environment is the ability to expand and shrink the size of the Hadoop cluster to cope with the expected loads and storage requirements of the jobs that are being submitted. In a cloud environment based on a service-based architecture, the expansion and contraction is normally managed through the control portion of the cloud service.

In general, expanding your cluster is easy, because you are adding more nodes to an existing configuration where the additional resources can be used more easily. Shrinking your cluster is more difficult and can cause a potential performance loss and job interruption.

Depending on your chosen cloud environment, the exact method with which you increase or decrease the size of your Hadoop cluster can change. Service-based cloud environments have some built-in scaling. Virtual machine-based units require deployment, installation of the software, and enabling within the cluster.
Using elastic Hadoop clusters

Truly elastic Hadoop clusters require a significant amount of work and management. Even with cloud services to start up and spawn larger numbers of nodes and then delete them later, the actual addition and management of the data is complex.

The big problem is that for the cluster to work on the problem with maximum efficiency, you really need to distribute the data and the workload.

Furthermore, consider the time these processes take. Even in the best case, the process to get individual nodes up and running and ready to start dealing with work requires five to ten minutes.

 Shrinking is always more complex because you have to avoid de-commissioning nodes that might be holding all of the replicas of the same block of data. To reduce the size of the cluster, you must first perform a rebalance and ensure that the data is correctly stored and replicas exist across the remainder of the nodes, before reducing again and repeating the process.

Expanding your Hadoop cluster

Expanding your cluster with new nodes is a common need, and the sequence is quite simple. In general, when you add nodes within a cloud environment, add new nodes of the same size and capacity as your existing nodes. This approach helps with future capacity planning. This general rule does not apply in the following cases:

- If you are planning to upgrade your nodes with more space, CPU, or RAM capacity
- If you are expanding your cluster before your cluster reaches 80 percent capacity. Increase your node count only when your current capacity reaches 80 percent. Failure to wait to expand your cluster might result in your running out of room.

After you have made the decision to add more nodes, the actual sequence is quite straightforward:

1. Add your new nodes to your cloud environment.
2. Install Hadoop on each new node. (Spawning a pre-installed node from an image is the easiest method.)
3. Add the new node information into the conf/slaves file on your master node.
4. Start the Hadoop processes shown in Listing 1

<table>
<thead>
<tr>
<th>Listing 1. Starting the Hadoop processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ hadoop-daemon.sh start datanode</td>
</tr>
<tr>
<td>$ hadoop-daemon.sh start tasktracker</td>
</tr>
</tbody>
</table>

Different Hadoop variants might have different steps, but these are the basic steps. You might also want to ensure that they are identified correctly by the master node by checking the dfs.hosts configuration. To check that they are identified correctly, run the code in Listing 2.

<table>
<thead>
<tr>
<th>Listing 2. Ensuring that Hadoop variant is identified correctly</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ hadoop mradmin -refreshNodes</td>
</tr>
<tr>
<td>$ hadoop dfsadmin -refreshNodes</td>
</tr>
</tbody>
</table>
This code should set the node up for MapReduce job processing, but it does not move any existing data. The next step is to ensure the file blocks are moved.

Redistributing stored blocks

You can move the stored data blocks around your cluster to make better use of the nodes you have added by using any of these methods:

1. Copy the files to a different directory. This action automatically redistributes the blocks, because the file is effectively rewritten into HDFS. Although this step requires additional work, it can be performed at the same time as your workflow.
2. Temporarily increase the replication size. The default is 3. Increasing this to 4 adds new block copies to the cluster. Reducing the size back to 3 removes the blocks from some machines.
3. Explicitly start a rebalance operation:

   $ start-balancer.sh

Remember that starting any kind of rebalance operation requires significant I/O and network transfer until the rebalance has completed.

Shrinking your Hadoop cluster

When shrinking your Hadoop cluster consider these factors:

- **Can you reduce your cluster safely?** If jobs are currently running, run a decommission process on each node. This process removes the node and stops any running task in a restartable fashion (using the status `KILLED_UNCLEAN`) so that the task can be rescheduled on an available node when the queue is next examined and the JobTracker re-assigns the task.

- **Can you free up space on your other nodes?** Remember that reducing the cluster size reduces the number of machines around which data can be replicated and increases the disk usage on the remaining nodes.

  **Note:** Never decommission more than one node at a time.

The temptation with cloud environments is to start 20 new nodes in your 100-node cluster to cope with a spike and then to remove them later. This approach opens the cluster to the risk of reducing or even completely removing stored data. The decommission process automatically redistributes replicas among the remaining nodes. Although you can reduce big blocks, doing so puts a significant load on the system. Instead, reduce the nodes in multiple blocks.

The safest way to decommission multiple nodes is to do it in a staged fashion. For example, with 20 nodes to remove, decommission three or five nodes at a time:

1. Add the nodes to be removed from the cluster to the `dfs.hosts.exclude` setting.
2. Run the `dfs` refresh to update the node list:

   $ hadoop dfsadmin -refreshNodes

3. Refresh the MapReduce configuration:
The nodes are now marked as decommissioned. The replicas are copied to other hosts in the cluster before the node is finally decommissioned and the machine can safely be removed.

Now repeat these steps for each additional block. Typically, this process takes longer than the expansion process, but it removes the potential for data loss.

**Upgrading your node configuration**

One of the major advantages of the cloud environment is the flexibility to change the configuration of your individual nodes, even to completely update and replace your nodes as required. You can make these changes in a staged fashion, combining the expansion and reduction processes described earlier. You can even make the process part of a planned expansion and reduction process for a specific job.

For example, to move 20 data nodes from four CPU systems to eight CPU systems, perform the following steps:

1. Add four new nodes with the new configuration.
2. Add them to the configuration.
3. Start the services.
4. Perform a rebalance.
5. Decommission four data nodes from the old configuration
6. Repeat these steps.

The result is a cluster that has first been expanded and then reduced by adding new configuration nodes and removing old configuration nodes.

**Job scheduling and distribution**

Determine when to expand and shrink your cluster based on your job scheduling needs and how much capacity you need to be able to cope with that process.

With the cloud model, you can sometimes gain an advantage by using multiple individual clusters rather than a single large cluster. You can enable scheduling of the work according to the complexity of the data and the size of the cluster. For example, a larger processing job might require more nodes but less storage, and others might require more storage but less individual nodes for processing.

When working in the cloud, try to maximize the cluster configuration without expanding or increasing the cluster size. Many tools are available to help, including basic job scheduling and making use of complex workflows such as the application manager within IBM InfoSphere BigInsights, or with Oozie. The goal is to get the maximum performance out of your cluster for the cost and to ensure that you do not overload the cluster to the point that it cannot easily be expanded or recovered.
Coping with storage and load spikes

Probably the most complex process is to understand how to cope with sudden storage and load spikes. Depending upon your deployment environment, the choices available to you might change significantly. For the factors that you can change, what do you do when you realize your cluster is running out of capacity?

The obvious argument is to try to avoid this specific problem in the first place. You might want to keep an eye on the capacity and make sure you have 20 to 30 percent free to be able to cope with the work. Running above 80 percent capacity is asking for trouble.

Determine the requirement

First determine whether you need disk or MapReduce capacity. The two have different properties. The inclination in both is to add more nodes, which is an approach that typically solves both problems, but at an expense that might be unnecessary.

If the problem is storage, consider adding more storage devices to your existing nodes. Some cloud environments offer this option without rebooting or changing your system. In that case, you can update the configuration of `dfs.datanode.data.dir` to include the newly mounted directory. This option is always much quicker and easier than expanding your cluster with new nodes.

If the problem is longer term, consider adding and replacing some of your existing nodes with nodes that have more capacity. This can be a worthwhile investment in the long run, because it will prevent more problems if you have future spikes.

If you need CPU power but are willing to wait on the storage requirement and not rebalance, add the new nodes, install Hadoop, and start the data node and task tracker processes, but do not perform a rebalance.

Determine if you can expand (and shrink) quickly enough to make a difference

If the spike is short in comparison to the expected length of the job, it is probably not worth the process of adding nodes, because it takes longer to create the new hosts than the time it takes to process the job.

Although there is no general rule, keep in mind that although a cloud deployment might enable you to increase your node capacity by 10 percent or even 100 percent, it might not increase the performance in an entirely linear fashion, especially within a cloud environment.

If your job is expected to run for 6 hours and you can effectively increase your cluster size by 50 percent within less than an hour, then it may be worth expanding it.

Can you also shrink when the spike completes?

Consider how much time it will take to reduce when the spike is over. Reduction requires additional time to decommission and redistribute the blocks across the remaining nodes of the cluster. The
graph in Figure 1 shows approximate times for the same job run over 10 nodes, and then run over increasing number of nodes including the time to add and decommission the nodes to the cluster, with overall times measured in hours.

**Figure 1. Approximate times for the same job run over 10 nodes**

You can see from this chart that adding 5 nodes is quick, and tripling the size of the cluster to 30 is simple, but it took an additional 2.5 hours to decommission the nodes. This process results in a savings of only one hour, but a tripling of the cost.

**Conclusion**

Deploying Hadoop within the cloud is about understanding the limitations of the cloud environment, coupled with the advantages of being able to dynamically expand and contract your cluster as the need arises. But the elastic nature is not without its pitfalls. Effective Hadoop deployment is therefore about understanding how long it takes to run your jobs, and how long the expansion and contraction process will take to maximize the time for your job execution within the cloud.
Resources

Learn

- **Analyze and optimize cloud cluster performance for Hadoop** (Yu Li, developerWorks, March 2011): This article shows you how to take full advantage of Hadoop by introducing Hadoop configurable parameters and using them to monitor, analyze, and tune the performance of your Hadoop cluster.
- Read an **Introduction to Hadoop on the cloud using BigInsights on BlueMix**.
- Watch the **Big Data: Frequently Asked Questions for IBM InfoSphere BigInsights video** to listen to Cindy Saracco discuss some of the frequently asked questions about IBM's Big Data platform and InfoSphere BigInsights.
- **Exploring your InfoSphere BigInsights cluster and sample applications** (Cynthia M. Sarraco et al., developerWorks, April 2012): Learn more about InfoSphere BigInsights' web console.
- **HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads**, Azza Abouzeid et al., Proceedings of the VLDB Endowment, 2(1), 2009: This paper explores the feasibility of building a hybrid system that takes the best features from both technologies.
- **MapReduce: Simplified Data Processing on Large Clusters**, Jeffrey Dean and Sanjay Ghemawat, OSDI, 2004

Get products and technologies

- Get the **IBM InfoSphere BigInsights Quick Start Edition**.
- Get **Hadoop MapReduce**.

Discuss

- Participate in **developerWorks blogs** and get involved in the developerWorks community.
About the author

Martin C. Brown

A professional writer for over 15 years, Martin (MC) Brown is the author and contributor to more than 26 books covering an array of topics, including the recently published *Getting Started with CouchDB*. His expertise spans myriad development languages and platforms: Perl, Python, Java, JavaScript, Basic, Pascal, Modula-2, C, C++, Rebol, Gawk, Shellscript, Windows, Solaris, Linux, BeOS, Microsoft WP, Mac OS and more. He currently works as the director of documentation for Continuent.