**A Beginners Look at Hadoop**

**By** [**David Poole**](http://www.sqlservercentral.com/Authors/Articles/David_Poole/3226/)**, 2015/01/01 (first published: 2013/06/05)**

**FROM:** [**http://www.sqlservercentral.com/articles/Hadoop/99135/**](http://www.sqlservercentral.com/articles/Hadoop/99135/)

I have been reading about Hadoop for a few years, but as my experience is exclusively Windows based, I have only recently got my hands dirty playing with the system.

My opportunity arose when two things happened.

* A colleague organized a Saturday morning boot camp on Linux.  As the words training & free are rarely combined I took this as a gift from the Gods!
* My company was reviewing its approach to click stream data and wanted to understand the technology options.

Even a modest web company will generate large amounts of click stream data so it is not surprising that it is usually mentioned as an example of "Big Data".  I have my own thoughts on what constitutes Big Data, but perhaps the best description of Big Data I have heard came from Buck Woody.  He described it as “*data that cannot be processed fast enough with the technology available today*”.  Perhaps a variation of this would be "technology you can afford today".

No matter which statement Hadoop offers a potential solution to the challenge of mining click stream data or indeed any suitably large pool of data The reason that Hadoop has acquired momentum in the market place is summarized below:

* It addresses situations where a single computer does not have the storage capacity and/or the computing power to solve the problem at hand.
* Although individual nodes are not particularly fast or their data processing particularly efficient Hadoops’ power comes from its ability to scale out and to do so massively and thus give huge performance.  This is particularly true of cloud based implementations.
* Scale out is on cheap off-the-shelf kit rather than enterprise grade servers thus the cost per node can be 10% or less that of a traditional server.
* Open source costs for both the Linux operating system and Hadoop technology stack.

**What is Hadoop?**

First a bit of history.  Back in 2004 Google published a white paper describing the Map-Reduce algorithm as a means of processing large and/or computationally expensive datasets.

It was [Doug Cutting](http://en.wikipedia.org/wiki/Doug_Cutting) and [Mike Cafarella](http://en.wikipedia.org/wiki/Mike_Cafarella) at [Yahoo that took that paper](http://www.wired.com/wiredenterprise/2011/10/how-yahoo-spawned-hadoop/) and implemented what we now call Hadoop back in 2005.  This was originally to be an extension of the Apache Nutch search facility.

|  |  |
| --- | --- |
| Hadoop at its core | Like most brilliant ideas the concept behind Hadoop is actually very simple.   * It is a file system written in Java * It is surrounded by a series of Java class libraries that act as an API and also carry out the mapping and reducing   The commands used to interact with the file system are based on the Linux file system commands.  For example, on Linux to display the contents of a file called readme.txt you would type the following:-  cat readme.txt |

To do the same thing in hadoop you would have to type the following:-

hadoop fs -cat readme.txt

It is quite important to remember this because Hadoop writes your data to three separate data nodes but your data is simply a file like any other.  You could log onto each individual node and use Linux to see your readme.txt file!

At its heart Hadoop is a system that allows you to scan immense files an as such is really a batch processor rather than a real-time analytics source.  However there are some caveats to this which I will go into later.

**Getting started**

Just to recap my starting point:

* 4 Hours of Linux experience
* No Hadoop experience
* No Java experience
* Negligble open-source experience

So hilariously ambitious though it may sound, I decided to try and use what I had learnt about Linux on an Saturday morning to see if I could actually get Hadoop up and running for myself. Why take on such a ludicrous challenge?  Well I wanted to discover a few things.

* How much could a complete novice achieve with Linux?
* What was the quality of support material to help a novice?
* Just how difficult is it to get started with an open source project?

So the best way to get to grips with technology is to attempt to use it.

My starting point was to download [Oracle Virtual Box](https://www.virtualbox.org/) so I could set up a virtual PC and install an [Ubuntu](http://www.ubuntu.com/) as my preferred flavour of Linux.

The next stage was to download and install [Hadoop](http://hadoop.apache.org/) from the Apache web site using both the instructions on the Apache site and also "[installing hadoop on ubuntu linux](http://mysolvedproblem.blogspot.co.uk/2012/05/installing-hadoop-on-ubuntu-linux-on.html)".

At this point I started to run into some difficulties that did turn out to  a repeating pattern as I attempted to install more and more of the optional ecosystem that surrounds Hadoop.

* Dependencies on other software components.  After some head scratching and a phone call to my Linux colleague the  first (of many) of these turned out to be CLASSPATH issues with Java and not too difficult to resolve.
* There are several articles on installing Hadoop on Linux.  There usability was extremely variable.

The reason I am not detailing the steps that eventually got Hadoop up and running for me is that the rate of change in the Hadoop ecosystem is incredibly fast.  Instructions that probably worked fine when the various authors wrote them ceased to work as the version numbers incremented.  Reproducing the steps will simply add to the canon of articles that used to work!

All in all it took a day to get a working Hadoop environment running on my virtual instance.  Having gained experience from this I believe I could get a single instance up and running in about two hours.

Hadoop does come with some simple examples that you can use to test that it is working.  These are simple word counts but will at least let you proove that the system is working.

I recommend taking a look at the [Yahoo Hadoop Tutorial](http://developer.yahoo.com/hadoop/tutorial/index.html) as a good primer for the system.

**What is SQOOP and HIVE?**

With SQOOP think BCP for Hadoop.  It is a bi-directional bulk importer that allows you to grab data from a SQL database and put it into Hadoop or extract data from Hadoop and push it to a SQL intance.

The Microsoft implementation of [SQOOP](http://www.microsoft.com/en-gb/download/details.aspx?id=27584) I used is now no-longer available as it is part of the main Apache SQOOP 1.4 distribution. To export data from Adventureworks.Person.Address the SQOOP command would be as follows:-

sqoop.cmd export -connect "jdbc:sqlserver://YourDBServer:1433;database=Adventureworks;user=AWUser;password=AWPass" --table Person.Address --hive-import

Just like BCP SQOOP does allow records to be imported using a SQL query against the source system.

By using the --hive-import switch the structure of the originating SQL Server table is copied into the HIVE metadata store to allow us to use HIVE to query the data at a later date.

I tested SQOOP against all versions of SQL Server available to me an found it to work with all versions from SQL2005 onwards.

| **Version** | **Success** |
| --- | --- |
| 2000 | NO |
| 2005 | YES |
| 2008 | YES |
| 2008R2 | YES |
| 2012 | YES |

Potentially you could use a Hadoop cluster as an online archive facility for your traditional SQL Server estate.

Hive itself allows you to use a subset of MySQL commands (plus a few extensions) to interrogate data in a Hadoop cluster.

**SQOOP behaviour and profiling SQL**

Some behaviors became apparent when using SQOOP.  I carried out various tests on data sets up to 100 million rows and made the following discoveries:-

* SQOOP expects there to be a single column primary key field in the source table which it will use by default if present.
* Where there is no primary key SQOOP will have to be told which field will act as a unique key on which to split the load across the Hadoop nodes.
* If there is no unique key then Hadoop will have to be told to use only ***a single node*** with obvious impact on performance
* Prior to starting the upload Hadoop throws a number of queries at SQL Server to establish structure of the source data.
* SQOOP seems to assume the primary/unique key to be contiguous.  Evidence for this is the way it populates the nodes
  + Determine the MIN/MAX value for the key
  + Divide the difference by the number of nodes
  + Run a query WHERE key BETWEEN x AND y

**Structure discovery query**

SQOOP runs a query to the affect that SELECT \* FROM table WHERE 1=0.  This brings back a list of fields in the source table.

It then runs SELECT column\_name FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME=table name which ultimately achieves the same thing.

This does no harm but is a curious behaviour.

**Batch import size**

 The default behavior of Hadoop seems to be using 1 reducer per CPU core so irrespective of the size of the data SQOOP simply divided the data by the number of cores and ran a SELECT against the source database. This meant that 100 million records were simply split into 4 separate queries selecting 25 million records each.

This is clearly going to have a big impact on the source system so particular attention must be given to the Hadoop (and supporting component) cluster configuration.

**What is HIVE?**

Hive allows you to use SQL to query a Hadoop cluster.  As mentioned earlier the dialect is a subset of MySQL called HiveQL. When you run HiveQL Hive generates and then runs the Java classes to carry out the Map Reduce functions corresponding to your SQL statements.

It is possible to access the source code to these Java classes and use you Java skills (which I don't have) to optimise them further. The main difference is that HIVE has far simpler data types.

| **SQL Type** | **Hive Type** |
| --- | --- |
| TINYINT | TINYINT |
| SMALLINT | INT |
| INT | INT |
| BIGINT | BIGINT |
| DATETIME | STRING |
| SMALLDATETIME | STRING |
| VARCHAR | STRING |
| VARCHAR(MAX) | STRING |
| NVARCHAR | STRING |
| BIT | BOOLEAN |
| FLOAT | DOUBLE |
| DECIMAL | DOUBLE |
| MONEY | DOUBLE  Neither SQL Types should be used as these are not ANSI SQL compliant |
| SMALLMONEY |
| BINARY  VARBINARY | Neither of these two data types is supported by the current version of SQOOP/HIVE |

**HIVE gotcha**

HIVE is reliant on the metadata store to record the structural metadata necessary to allow SQL to be used against flat files. I spent a considerable amount of time getting a suitably large volume of data into my Hadoop/Hive installation and was shocked to find that it appeared to have lost my data. I quickly established that the data itself had not been lost but the structural metadata appeared to have vanished.

It turned out that when you use the default metastore (Apache Derby) it is sensitive to the Linux directory you are in when you start a Hive session.  Quitting Hive, navigating to the correct directory and starting a new Hive session revealed the table in all its glory.

It is possible to use a small MySQL database as the metastore.  As this relies on a JDBC connection string then theoretically you could use SQL Server for the metastore though I haven't had time to test it.

**Beyond Hadoop and HIVE**

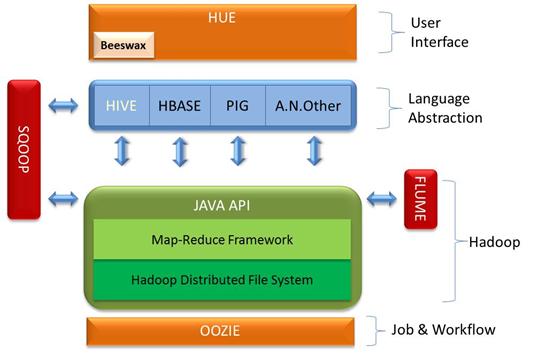
I was encouraged by my success with Hadoop and Hive but unfortunately it was a case of pride coming before a fall.

I had no success installing PIG and made the BIG mistake of running sudo apt-get update which broke my installation 10 minutes before I was supposed to demonstrate my findings.

**Cloudera**

When Yahoo started off with Hadoop they spun up a subsidiary called Hortonworks.  This is still going strong today and it is this distribution that is at the heart of Microsoft's HDInsight service and server. Doug Cutting, one of the co-inventors of Hadoop, went on to Cloudera who produce a complete downloadable distribution of the Hadoop ecosystem.

To go further I decided to download the Cloudera distribution to see how far I could get.



With Cloudera it was simply a case of downloading a virtual image and attaching it to my Virtual Box.  I found I had the full Hadoop ecosystem available to me with everything (except SQOOP) properly configured ready for me to use.

Since I put together the above graphic other elements have been added to the Hadoop ecosystem.

Facebook released [Corona](http://gigaom.com/2012/11/08/facebook-open-sources-corona-a-better-way-to-do-webscale-hadoop/) as an open source job tracker because Hadoop simply wasn't scalable enough for them. In fact the Map-Reduce framework itself has become "Yarn" which is effectively Map-Reduce Version 2.0!  I have listed as many of the Hadoop ecosystem components as I can below.

| **Term/Technology** | **Description** |
| --- | --- |
| Hadoop | Consists of three main parts           The distributed file system HDFS on which files are stored           The Map-Reduce framework that handles the splitting and merging of tasks.           The API layer to enable external programs to interact with the Map Reduce algorithm. |
| Oozie | Is a workflow scheduler that enables the various Map-Reduce jobs to be scheduled and run.  Oozie can be used to run jobs written in the various language abstractions or as Java programs themselves. |
| SQOOP | This is a command line utility to enable data from the more traditional RDBMS’s into Hadoop. In addition to importing direct to Hadoop it can also import using HIVE and therefore populate the HIVE metadata store thus mimicking the RDBMS structure.  The tool can also pump data from HDFS back to the relational store thus the results of a massive Map-Reduce job can be populated back into the RDBMS. |
| Flume | Framework components that can be installed, primarily on web servers, to stream data into Hadoop.  At present the key use for this is transferring web logs into Hadoop though the potential exists to expand this out into a poor mans’ complex event processing. |
| ZooKeeper | In essence a distributed transaction co-ordinator |
| Hue | A user interface to the Hadoop cluster that also includes other user interfaces such as Beeswax |
| Beeswax | A user interface for HIVE to allow simple query and data tasks to be accomplished. The interface will be familiar to anyone who has used PHPMyAdmin to administer MySQL. |
| HIVE | Allows interrogation of files on the Hadoop cluster using a dialect of SQL similar to MySQL.  Fundamental to HIVE is the “metastore” database which records the structure of the Hadoop files. This metastore can be local to a directory or shared across all users. |
| HBase | Effectively a column based store sitting on top of Hadoop thus combining scalability with high performance. |
| PIG | Short for Pig Latin. A simple language that will be familiar to procedural programmers. |
| Whirr | A framework to allow Hadoop nodes to be spun up in the cloud if required. |
| Mahout | A machine learning and data mining toolset |
| Yarn | This is effectively Map-Reduce version 2. |
| Avro | The communication framework used within Hadoop for exchange data in a compact binary form. Although used by Hadoop it does have a wider technological application |
| Impala | Real-time relational store overlaying either HDFS or HBASE.  This product was released on October 2012 and is intended to give Parallel Data Warehouse like capabilities. |

**Hadoop and Performance**

When I first started looking at Hadoop it did not have the concept of data location awareness.  This meant that if you ran a query against a Hadoop cluster it ran the cluster against every node. I was curious about how Hadoop would perform in real life vs SQL Server so a colleague with AWS experience took my sample 100 million records and uploaded it onto a large 4 node Elastic Map Reduce cluster and got the results back from a query in 14 seconds.

This seemed exciting until I BCP'd the data into a test SQL 2008R2 instance on my lap top and achieved the same thing in 4 seconds!

There are some key points to bear in mind with these results:

* Hadoop is designed for situations where there is simply too much data for a single machine to hold or process in a reasonable timeframe.
* My example code was simple string extraction and was not a challenge for either Hadoop or SQL Server.  In reality we were constrained in our example by our knowledge of PIG
* AWS disk performance is notoriously mediocre.
* Time to get a significant amount of data into a cloud provider can be significant.  I took minutes to BCP data into my SQL Server but hours to upload the same data to Amazon.

Some vendors have sought to address Hadoop's performance, notably [Rainstor](http://rainstor.com/).

**Rainstor**

Rainstor are a company with a specialist database designed to operate in the 100 Petabyte scale.  This can be deployed on Hadoop to give data location awareness. It employs massive compression techniques to reduce the number of Hadoop nodes necessary to carry out a particular task.

|  |  |
| --- | --- |
| **Compression technique** | **Compression** |
| LZO | 3x |
| Compressed Relational | 6x |
| GZIP | 7x |
| Column DB | 8x |
| Rainstor | 40x |

Compression allows the number of Hadoop nodes to be hugely reduced and therefore the purchase cost and administrative burden of managing a large number of servers. As Hadoop nodes are added to facilitate storage rather than compute power the extra CPU load due to compressing data is easily absorbed. The company claim that the combination of their compression engine and data location awareness can massively reduce the query time of a large Hadoop cluster.

Most Big Data solution vendors have reached the conclussion that compression is essential in Big Data solutions.

**Potential uses for Hadoop with a traditional SQL server installation**

Bog standard Hadoop may be slow but the whole point of it is that it runs on relatively cheap commodity hardware.  Installing it on enterprise servers backed by a SAN defeats the whole point of the exercise. Where I see it playing a part in companies without a Big Data problem is in data profiling.  Having seen the demands that the SSIS data profiler places on traditional infrastructure, even with relatively modest amounts of data, the prospect of having a few Hadoop nodes grinding away at a profiling problem is appealing.

Complex ETL also strikes me as being an area where Hadoop has potential.  By complex ETL I mean the sort of data extraction that would normally require complicated script components. The general comment from the ETL tool industry is that although Hadoop could be used as a massively scalable ETL toolkit it has taken years for vendors to build their tools.  It will take a similar time for people to build the equivalent Hadoop components.  I rather think this misses the point.

**Hadoop and security**

Hadoop has not been designed with security in mind.  It was originally built boost up a search engine whose intent is to reveal data not lock it down.

I have spoken to some implementers who have a specially isolated Hadoop cluster with absolutely no connections to the outside world or indeed their own networks.  Data is loaded by plugging in and mounting data lockers containing the relevant data then copying the contents into the cluster.

Remember that users of Hadoop have exactly the same file privileges as Hadoop itself.

**So what have I learnt?**

New tools and products for the Hadoop ecosystem are emerging and evolving at a rapid rate.  This indicates the level of interest and investment that is being made in Hadoop. On the downside, keeping track of such a fast moving ecosystem is extremely challenging.  What we know today will be out-of-date tomorrow, which poses significant challenges if you are going to integrate Hadoop into your existing ecosystem. With that in mind and the number of components in the Hadoop ecosystem go with a pre-canned distribution of Hadoop.

Hadoop is not "free".  It may have a zero purchase cost but when it comes to "Total Cost of Ownership" things start to get a bit murky.  If you have huge amounts of data then the cost of hardware and the housing and management of that hardware is going to be a challenge.

If you are going to use the cloud then getting data to the cloud implementation is not going to be a easy unless you pay for a dedicated VPN.

By all means play with Hadoop or indeed any technology but if you go beyond playing then a competent 3rd party is essential to get you through the learning curve.

Hadoop is not fast.  Yes there are configuration options and special builds but you have to consider your use case.  In my use case I could achieve my aims far faster and in a cost effective manner with a single instance of SQL Server.

If I might use an analogy if SQL Server is a BMW 535 Estate (station wagon) capable of carrying one ton then Hadoop is a 40 ton truck.  Up to 1 ton the BMW wins hands down in all scenarios.  From 1 to 5 tons it depends on the load and distance and from 5 tons up to 40 tons the truck wins every time.