Hadoop + IBRIX: High Availability Hadoop

Johannes Kirschnick, Steve Loughran, Julio Guijarro

OST/Labs

Johannes.kirschnick@hp.com, steve.loughran@hp.com, julio.guijarro@hp.com

Abstract

Apache Hadoop, an open source re-implementation of Google's MapReduce framework and the MapReduce parallel programming framework, is becoming the de-facto standard for large scale data mining of unstructured data. Hadoop is complimented by a highly scalable file system, the Hadoop Distributed File System (HDFS). HDFS has reliability and security problems which make it not adequate for enterprise applications, such as data warehousing. For our enterprise customers we proposed to replace HDFS with the HP IBRIX file system, which offers in addition to reliability and security a noticeable storage cost advantage over HDFS, giving HP a clear differentiation and advantage over similar offerings.

Problem statement

The falling cost of storage, gigabit networking and “commodity” servers is transforming data warehousing. HP and its customers can now afford to store Petabytes of “low-value" data and extract value from it.

Apache Hadoop (1), an open source re-implementation of Google's MapReduce (2) framework can work with this data, using the MapReduce parallel programming framework to efficiently distribute computation across large numbers of server nodes, each of which can store data and run work near this data. Figure 1 presents a high level overview of how a MapReduce job gets executed in Hadoop.

With clusters of thousands of servers, each with many disks, failures become the common and therefore the storage layer and work scheduler must be designed to handle failures. Furthermore, network bandwidth within a cluster can be a scarce resource. Hadoop addresses this with a layered architecture: a distributed file system layer, which serves as the backend store for data and the MapReduce layer which executes work on these machines, whereas each layer can separately handle machine failures.

The risk of hardware failures influenced the design of HDFS, the bundled file system, which copies data across non-RAIDed disks in the worker machines; moving the failure management into the application itself. The MapReduce framework runs alongside the same nodes that store data, so that complex jobs can be broken up into a sequence of Map and Reduce tasks which are scheduled to run on the machines which physically hold particular pieces of data, thus sending computation to the data and limiting excess bandwidth usage.

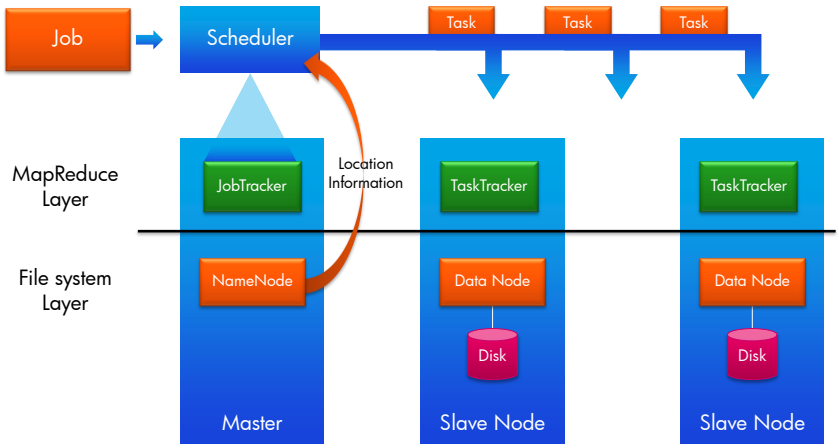

Figure 1 Hadoop architecture

Unfortunately, the HDFS file system, although designed to be fault tolerant, features one single point of failure, the NameNode; the central file system index server. A crash of the NameNode server will take the entire cluster down. HDFS is also insecure: anyone who can run code in the data centre can access all the data, which implies that anyone who can submit a MapReduce job has access to all the data. These limitations prevent its acceptance in the enterprise; which want high available, large data processing clusters.

Our solution

We make use of HP IBRIX (3) as an alternative file system which offers in addition to reliability and security a noticeable storage cost advantage over HDFS. IBRIX is a proven high performance, secure, and highly scalable distributed file system which delivers fault tolerance through RAIDed storage. It can be deployed in a number of different configurations, trading off performance and resilience. IBRIX is composed of a number of segment servers, which each have access to underlying physical storage. A number of segment servers can be grouped together to create a file system which can be access by clients using, a native protocol between the segment servers, or exported via a number of standard transports (NFS, CIFS, Webdav).

To expose the file system to Hadoop, we configured it to run in a "couplet mode", where two segment servers can see each other's physical storage in an active-active configuration. While they see the same storage all the time, only one of them will be responsible for the storage at one time, failing over if needed.

Each server running the segment servers will also run parts of the MapReduce layer, the task tracker, to retain the location aware scheduling of tasks near to the data.

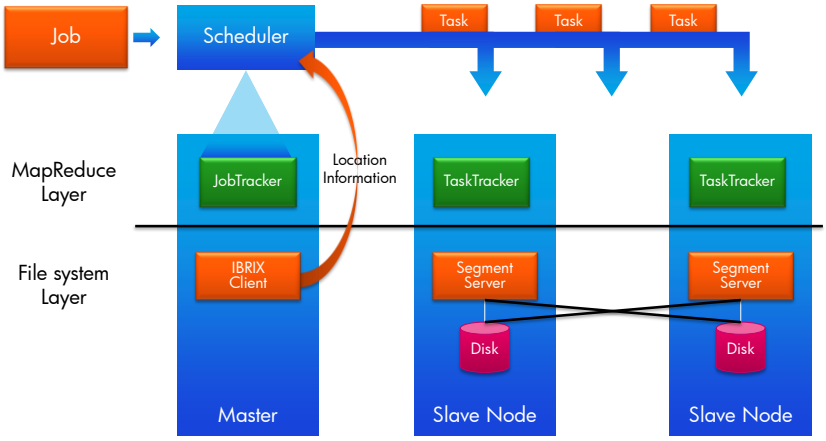

Figure 2 Hadoop running on top of the IBRIX files system

Our main contribution is the integration between Hadoop and IBRIX. The main key feature that Hadoop demands from a file system is to be able to place computation near to the data and therefore the biggest challenge was to expose the location of files retained in the IBRIX distributed file system to the scheduler running inside the Hadoop JobTracker.

To achieve this objective we developed a file system abstraction that is capable of querying the file system in real time for the location of a given file and presenting this information back in the format expected. One of IBRIX characteristics is that it can automatically rebalance the cluster, by moving files around, to improve performance and or reliability, which our solution can appropriately deal with.

Evidence the solution works

We evaluated our setup by running a scaled down version of TerraSort, using 1GB of randomly created records and sorting them. Figure 3 shows the total time it took to sort the data, while varying the number of worker nodes available in the cluster to show the scalability of the solution.

The tests were executed on the HP internal Cells as a service (4) cloud prototype in a virtual environment. Whereas the actual performance figures should not be used for comparison, it should be noted that IBRIX using the location aware scheduler relatively outperforms the non-location aware version, proving that scheduling work near the data works as expected. Furthermore, IBRIX relatively outperforms HDFS, but we attribute that to the small size of the dataset and generally expect a similar performance between the two file systems. By using IBRIX as an alternative file system we remove the single point of failure and add strong security features whilst still providing high performance data access.

Since IBRIX uses RAIDed storage to secure data, we get increased storage efficiency per node over HDFS which replicates data by default three times, as outlined in Table 1, which reverts into noticeable cost saving. This storage savings represent a big advantage for our solution since HDFS is typically deployed to store and process PetaBytes of data.

Table 1 Storage efficiency of RAID vs. HDFS (on a single node) with default replication

Disks	RAID 5	RAID 6	HDFS
3	67%	-	33%
4	75%	50%	33%
5	80%	60%	33%
6	83%	67%	33%
7	86%	71%	33%
8	88%	75%	33%
9	89%	78%	33%
10	90%	80%	33%

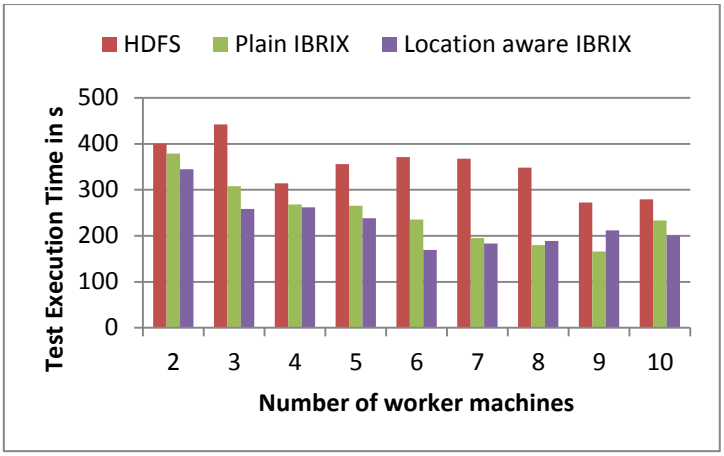

Figure 3 Performance of Sorting 1GB of random records

On the other hand, where this advantage becomes a disadvantage is that the use of multiple replicas of the same file by HDFS can potentially increase the performance of MapReduce Jobs, since IBRIX reduces the needed number of replicas, it therefore reduces number of jobs that can be run over the same data in parallel, while still taking advantage of the data locality.

Competitive approaches

Several competitors exist in this space, providing high performance analytics solutions. Notably Netezza (5) provide very powerful data mining solutions. Although they provide performance tuned algorithms which outperform similar implementations on Hadoop, Hadoop plus IBRIX still offers a cheaper TCO for storing PetaByte size data and analyzing it.

Current status

We successfully implemented an adapter for IBRIX using the existing Hadoop file system API such that unmodified MapReduce jobs can be submitted, transparently taking advantage of the HP proprietary file system.

The work has created a lot of internal interest for providing secure, fault tolerant, large scale data processing. NeoView are looking into providing real-time operational Business Intelligence and facilitate the ability to get data from multiple data sources, sort it, transformations it and load it into data warehouses. Hadoop is seen as a solution for transforming the data before it is loaded in the much more expensive NeoView, but it requires a highly resilient file system, as data should never be lost. Furthermore, other customers, like Vodafone 360 are eager to get into the Hadoop data processing world, but they consider security a must and demand rigid access controls for file access. To showcase our development we are able to dynamically bring up a virtual Hadoop cluster on demand on the Cells as a Service platform, running on top of IBRIX, in a matter of minutes.

Next steps

The Storage Works Division is trying to leverage our work in conjunction with Cloudera, a company which provides enterprise support for Hadoop, to certify IBRIX as a supported file system for Hadoop.

Furthermore, we are pursuing the idea of dynamically bringing up virtual Hadoop cluster to only bring up compute nodes on persisted data when needed. For this to work efficiently we would like include the ability to create on demand replicas of files transparently inside of IBRIX, thus increasing the performance comparative to HDFS, retaining the mentioned security features.

References

1. **Hadoop.** Apache Hadoop. [Online] http://hadoop.apache.org/.
2. *MapReduce: simplified data processing on large clusters.* **Dean, Ghemawat.** 1, s.l. : Commun. ACM, 2008, Vol. Commun. ACM.
3. **IBRIX.** [Online] www.hp.com/go/ibrix.
4. **CaaS.** Cells as a Service project (CaaS) . [Online] https://twiki.hpl.hp.com/bin/view/AIL/CellsAsAService.
5. **Netezza.** [Online] http://www.netezza.com/.