An Oracle White Paper
November 2010

Leveraging Massively Parallel Processing in an Oracle Environment for Big Data Analytics

Introduction

New applications such as web searches, recommendation engines, machine learning and social networking generate vast amounts of data in the form of logs, blogs, email, and other technical unstructured information streams. This data needs to be processed and correlated to gain insight into today's business processes. Also, the need to keep both structured and unstructured data to comply with government regulations in certain industry sectors requires the storage, processing and analysis of large amounts of data.

There are many ways to process and analyze large volumes of data in a massively parallel scale. Hadoop is an often cited example of a massively parallel processing system. This paper introduces the Hadoop framework, and discusses different methods for using Hadoop and the Oracle Database together to processes and analyzes a very large amount of data, taking advantage of the strengths of each solution.

What is Hadoop?

Hadoop is an Open Source implementation of a large-scale batch processing system. It uses the MapReduce framework introduced by Google by leveraging the concept of map and reduce functions well known used in Functional Programming. Although the Hadoop framework is written in Java, it allows developers to deploy custom- written programs coded in Java or any other language to process data in a parallel fashion across hundreds or thousands of commodity servers. It is optimized for contiguous read requests (streaming reads), where processing consists of scanning all the data. Depending on the complexity of the process and the volume of data, response time can vary from minutes to hours. While Hadoop can processes data fast, its key advantage is its massive scalability.

Hadoop leverages a cluster of nodes to run MapReduce programs massively in parallel. A MapReduce program consists of two steps: the Map step processes input data and the Reduce step assembles intermediate results into a final result. Each cluster node has a local file system and local CPU on which to run the MapReduce programs. Data are broken into data blocks, stored across the local files of different nodes, and replicated for reliability. The local files constitute the file system called Hadoop Distributed File System (HDFS). The number of nodes in each cluster varies from hundreds to thousands of machines. Hadoop can also allow for a certain set of fail-over scenarios.

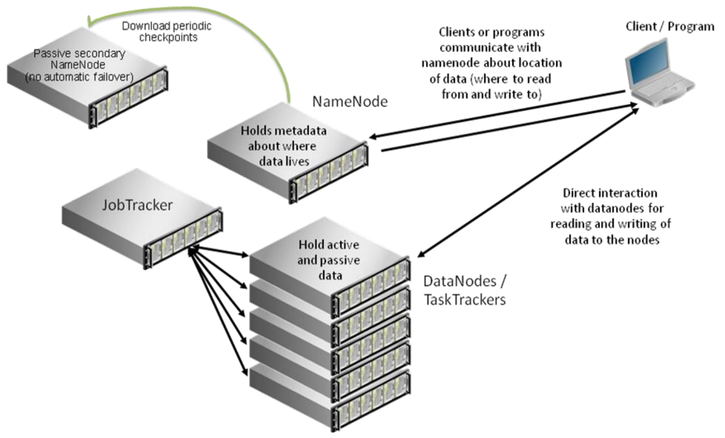

Download periodic checkpoints
Passive secondary NameNode (no automatic failover)
NameNode
Holds metadata about where data lives
Clients or programs communicate with namenode about location of data (where to read from and write to)
Client / Program
JobTracker
Hold active and passive data
DataNodes / TaskTrackers
Direct interaction with datanodes for reading and writing of data to the nodes

Figure 1. A high level overview of Hadoop cluster

Hadoop is currently being used for index web searches, email spam detection, recommendation engines, prediction in financial services, genome manipulation in life sciences, and for analysis of unstructured data such as log, text, and clickstream. While many of these applications could in fact be implemented in a relational database (RDBMS), the main core of the Hadoop framework is functionally different from an RDBMS. The following discusses some of these differences.

General Purpose vs. Point Solutions

While the main components of Hadoop provide a flexible framework, most Hadoop-based applications are very specific. Each program is custom written and specifically set up to do a certain task. The focus is on scanning hundreds or thousands of files in parallel to run a set of repetitive actions in a batch-oriented processing fashion. Access to Hadoop data wasn't originally intended for traditional business users, however, other open source projects, such as Pig and Hive have sprung up to provide an easier interface for users to access Hadoop data through very basic SQL alike syntax. However under the covers, the query formulations are converted to a series of MapReduce jobs to run in the Hadoop cluster. An RDBMS, on the other hand, is intended to store and manage data and provide access for a wide range of users.

Intermediate/real-time vs. batch

An RDBMS can process data in near real-time or in real-time, whereas MapReduce systems typically process data in a batch mode. Some MapReduce implementations have moved some processing to near real-time however.

Structure vs. Program

While an RDBMS stores data in a set of tables with rows and columns, a Hadoop programmer is free to structure data in any manner desired. This means that an RDBMS enforces a strict structure when loading data, whereas Hadoop data loads simply store without any predefined structure. Because an RDBMS has a structure (tables and columns) it is quite easy to express what information should be retrieved from that table. Within Hadoop, the programmer has to interpret and write that interpretation in the MapReduce program itself. This explains why loading data into Hadoop can be faster than loading data into an RDBMS.

The advantage of the Hadoop approach is that more formats can be handled, however with more complexity comes a greater need for expertise in handling such data in a programming language. These programs are typically a lot more involved than the general SQL statements used in an RDBMS.

Single vs. Multiple Access Strategy

For a Hadoop job to process a user request, data is always scanned in parallel. This allows a Hadoop system to scale by simply adding more nodes. However, if there is ever a need to process a subset of the data, or access some data randomly, then a full scan is still required. At that point, a single data access strategy becomes a limitation. Within the Oracle RDBMS, on the other hand, there are various specialized access strategies to ensure certain operations can be done very quickly - by index lookup for example.

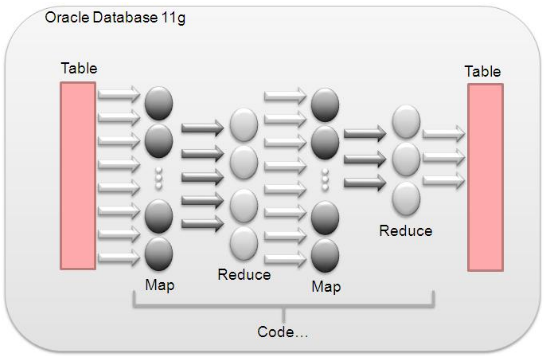
While at times a single data access strategy satisfies all query aspects, more often, a multiple data access strategy is required. Hadoop is optimized for contiguous read requests and can only append new data with no capability to do any updates or deletes.

Hadoop and Oracle are Complementary

The Oracle Database and a Hadoop cluster can be very complementary. There are several ways to integrate the two solutions; the nature of the data, the type of workload, amount of computer equipment, manpower skills, and service level requirements from the end user will dictate the optimal approach.

Oracle Database provides the flexibility to leverage programming language functionality within the database without having to write complex SQL statements by using user defined functions known as Table Functions.

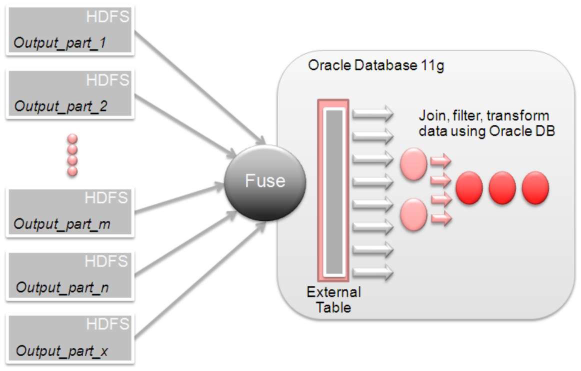
The MapReduce programming model can be implemented within the Oracle Database using Parallel Pipelined Table Functions and parallel operations. It is possible to write parallel processing tasks as database queries using user defined Table Functions to aggregate or filter the data. Pipelined Table Functions were introduced in Oracle 9i as a way of embedding procedural logic within a data flow. At a logical level, a Table Function is a user defined function that appears in the FROM clause of a SQL statement and operates like a table returning a stream of rows.



This mechanism provides an alternative for SQL developers to perform very complex processing in a procedural way, not easily expressed with SQL. It also follows the MapReduce paradigm, enabling massively parallel processing within the realm of the database.

Feeding Hadoop Data to the Database for Further Analysis

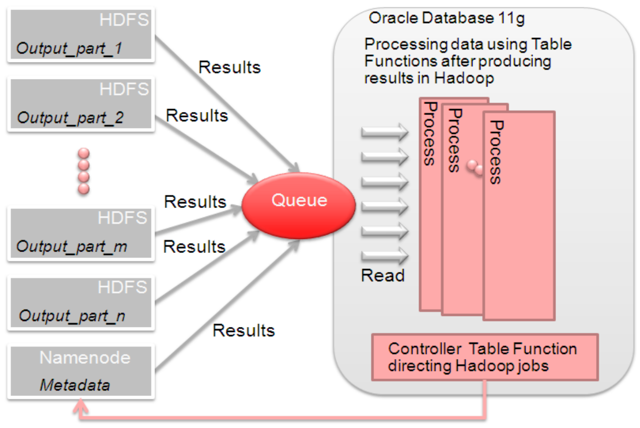
External tables present data stored in a file system in a table format, and can be used in SQL queries transparently. Hadoop data stored in HDFS can be accessed from inside the Oracle Database by using External Tables through the use of FUSE (File system in User Space) project driver to provide the application programming interface between HDFS and the External Table infrastructure. Using the External Table makes it easier for non-programmers to work with Hadoop data from inside an Oracle Database.



More often than not, Hadoop data is unstructured and contains lots of irrelevant material that may not be practical to store in an Oracle Database. However, correlating the data from Hadoop with data in an Oracle data warehouse can yield interesting results. Hadoop files stored in HDFS can be easily accessed using External Tables from an Oracle Database.

Leveraging Hadoop Processing From the Database

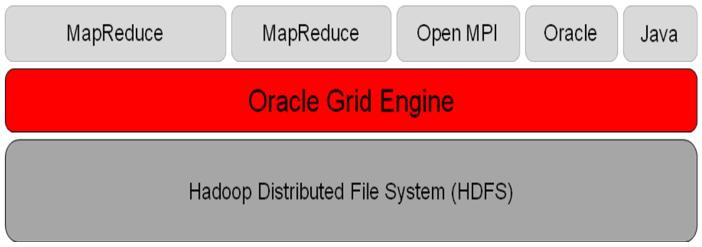
In the event that you need to process some data from Hadoop before it can be correlated with the data from your database, you can control the execution of the MapReduce programs through a table function using the DBMS_SCHEDULER framework to asynchronously launch an external shell script that submit a Hadoop MapReduce job. The table function and the MapReduce program communicate using Oracle's Advanced Queuing feature.



The scenario here is similar to the previous one where, despite storing the data outside of the database, there is a need to integrate the data with the Oracle Database. However, in this scenario, when there is a need to synchronize the timing of the data to be processed, one should use Table Functions and Oracle Advanced Queue features.

Drive Enterprise-Level Operational Efficiency of Hadoop Infrastructure with Oracle Grid Engine

All the computing resources allocated to a Hadoop cluster are used exclusively by Hadoop, which can result in having underutilized resources when Hadoop is not running. Oracle Grid Engine enables a cluster of computers to run Hadoop with other data-oriented compute application models, such as Oracle Coherence, Message Passing Interface (MPI), Open Multi-Processing (OpenMP), etc. The benefit of this approach is that you don't have to maintain a dedicated cluster for running only Hadoop applications.



Oracle Grid Engine provides an enterprise environment to deploy a Hadoop cluster along with other applications. It is a powerful business-driven workload manager that creates a layer of abstraction between your applications and the computing resources to provide an efficient way to match a workload to available resources and schedule jobs according to business policies, thus providing further scalability, flexibility and full accounting of resource utilization. In addition, Oracle Grid Engine has the capacity to scale out to cloud resources on-demand, and set idle or underutilized machines into a reduced power mode to conserve energy.

Conclusion

There are many ways to massively process data in parallel outside of a relational database. Hadoop follows the MapReduce programming paradigm to write programs to process the data. While Hadoop provides flexibility, the solution is very specific and requires software development skills to build a query to access the data. On the other hand, an RDBMS is intended to store and manage data, and use standard SQL language to provide access for a wide range of users.

Hadoop and Oracle are two different technologies that complement each other. Oracle Database features such as External Tables, Parallel Pipelined Table Functions, and Advanced Queuing provide ways to integrate Hadoop and Oracle Database. In addition, Oracle Grid Engine provides the means to harness the power of a Hadoop deployment. The benefits you can achieve are the following:

- By using Hadoop and then feeding the data to the Database for further analysis you can get more insight by correlating your data from Hadoop Cluster presented as an external table and your enterprise data residing in an Oracle Database.
- By leveraging Hadoop processing from the Database you can take advantage of the power of Oracle RDBMS at the same time to simplify the analysis of your data stored in a Hadoop Cluster by streaming data directly from Hadoop with Oracle Queue and Table Function.
- By driving enterprise level operational efficiency of Hadoop infrastructure with Oracle Grid Engine you can gain efficiency and return on assets by repurposing specialized cluster frameworks for other non Hadoop workloads.

References

- Integrating Hadoop Data with Oracle Parallel Processing
- In-Database Map-Reduce
- Oracle Grid Engine: An Overview

White Paper Title
November 2010
Author: Pedro Lay
Contributing Authors: Jean-Pierre Dijcks

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com