Processing Cassandra Datasets with Hadoop-Streaming Based Approaches

ABSTRACT:

The progressive transition in the nature of both scientific and industrial datasets has been the driving force behind the development and research interests in the NoSQL model. Loosely structured data poses a challenge to traditional data store systems, and when working with the NoSQL model, these systems are often considered impractical and costly. As the quantity and quality of unstructured data grows, so does the demand for a processing pipeline that is capable of seamlessly combining the NoSQL storage model and a “Big Data” processing platform such as MapReduce. Although MapReduce is the paradigm of choice for data-intensive computing, Java-based frameworks such as Hadoop require users to write MapReduce code in Java while Hadoop Streaming module allows users to define non-Java executables as map and reduce operations. When confronted with legacy C/C++ applications and other non-Java executables, there arises a further need to allow NoSQL data stores access to the features of Hadoop Streaming. We present approaches in solving the challenge of integrating NoSQL data stores with MapReduce under non-Java application scenarios, along with advantages and disadvantages of each approach. We compare Hadoop Streaming alongside our own streaming framework, MARISSA, to show performance implications of coupling NoSQL data stores like Cassandra with MapReduce frameworks that normally rely on file-system based data stores. Our experiments also include Hadoop-C*, which is a setup where a Hadoop cluster is co-located with a Cassandra cluster in order to process data using Hadoop with non-java executables.

INTRODUCTION

With the increased amount of data collection taking place as a result of social media interaction, scientific experiments, and even e-commerce applications, the nature of data as we know it has been evolving. As a result of this data generation from many different sources, “new generation” data, presents challenges as it is not all relational and lacks predefined structures. As an example, blog sections for commercial entities collect various inputs from customers about their products from Twitter, Facebook, and social media outlets. However, the structure of this data differs vastly because it is collected from varied
sources. A similar phenomenon has arisen in the scientific arena, such as at NERSC where data coming from a single experiment may involve various sensors monitoring disparate aspects of a given test. In this circumstance, data relevant to that experiment as a whole will be produced, but it may be formatted in different ways since it originates from different sources.

EXISTING SYSTEM

In Existing System similar challenges existed before the advent of the NoSQL model, earlier approaches involved storing differently structured data in separate databases, and subsequently analyzing each dataset in isolation, potentially missing a “bigger picture” or critical link between datasets. Currently, NoSQL offers a solution to this problem of data isolation by allowing datasets, sharing the same context but not the same structure or format, to be collected together. This allows the data not only to be stored in the same tables but to subsequently be analyzed collectively. When non-uniform data grows to large sizes however, a distributed approach to analyze unstructured data needs to be considered. MapReduce has emerged as the model of choice for processing “Big Data” problems. MapReduce frameworks such as Hadoop offer both storage and processing capabilities for data in any form, structured or not. However, they do not directly provide support for querying the data. Growing datasets not only need to be queried to enable real time information collection and sharing, but also need to undergo complex batch data analysis operations to extract the best possible knowledge.

DisADVANTAGE OF Existing SYSTEM

✓ NoSQL data stores offer not only the potential of storing large, loosely structured data that can later be analyzed and mined as a whole, but they also offer the ability for queries to be applied on such data.
✓ This is especially beneficial when real time answers are needed on only slices of the stored data. Despite the presence of this valuable batch

PROPOSED SYSTEM

In Proposed System order to fully exploit “Big Data” sets, we need a software
pipeline that can effectively combine the use of data stores such as Cassandra with scalable distributed programming models such as MapReduce. In this paper we show two different approaches, one working with the distributed Cassandra cluster directly to perform MapReduce operations and the other exporting the dataset from the database servers to the file system for further processing. We introduce a MapReduce streaming pipeline for the latter approach and use two different MapReduce streaming frameworks, Hadoop Streaming and MARISSA, to show the applicability of our system under different platforms. Furthermore, we present a detailed performance comparison of each approach under various application scenarios. Our results are summarized in Section V to help users make informed decisions for processing large Cassandra datasets with MapReduce using non-Java executables.

ADVANTAGE OF PROPOSED SYSTEM

- Data Preparation for Hadoop Streaming is nearly 1.3 times faster than MARISSA.
- Data Transformation stage can lead to great reduction in data size. This reduction in data size helps the performance of stage MR2.
- MARISSA performance advantage is more visible in smaller data sizes.
- Data Processing for read intensive applications with MARISSA is nearly 1.5 times faster than Hadoop Streaming up to 256 million records.

ARCHITECTURE:
HARDWARE REQUIREMENTS:

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- Floppy Drive: 44 Mb.
- Monitor: 15 VGA Colour.

SOFTWARE REQUIREMENTS:

- Coding Language: Java 1.7, Hadoop 0.8.1
- Database: MySql 5
- IDE: Eclipse