Weather Search System

Cloud Computing Project - 2012-13
MERC/EMDC
IST

October 29, 2012

1 Introduction

The goal of the cloud computing project is to develop a weather data processing and search system.

Students who wish to pursue extensions or alternatives to this project are welcome to propose them.

The system should be organized in three components:

- A MapReduce application running the weather data processing algorithm which imports logs from weather stations and summarizes the weather data into a database;
- A web application allowing simple queries over the output of the MapReduce application stored in the database mentioned above.
- A load balancing mechanism to manage the load among multiple instances of the web application.

1.1 Input Data

The system will receive data in the form of a folder containing text files with log entries from weather stations from several cities. Each line of each log file contains a measurement from a weather station listing city, date, hour, temperature, humidity, rainfall separated by commas:

- *city*: A string with the name of the city, e.g. Lisboa
- *date*: A string with the date of the sample, e.g. 2012-10-12
- *time*: The time of the sample, e.g. 17:54:01
- *temperature*: The temperature in degrees Celsius, e.g. 17
- *humidity*: The measured humidity as a percentage, e.g. 90
- *rainfall*: The amount of rainfall in the last hour (in mm), e.g. 5

An example of a log entry would be: *Lisboa, 2012-10-12, 17:54.01, 17, 90, 5.* Each weather station takes new measurements every hour and adds to its log one new entry per hour.

You can use the log files as the input for the MapReduce application or you can first transfer the data into another storage system.

1.2 MapReduce Application

The MapReduce application will be used to summarize the data from the log files. It will calculate for each city the maximum, minimum and average temperature and humidity for each day; for each day the total rainfall should also be calculated. These results should be obtained in the most efficient way possible by the MapReduce application.

It should be possible to run the MapReduce application periodically to take into account new log files arriving at the input files folder.

The choice of the system used to store the output of the MapReduce application is free. The selected storage system can be updated directly by the MapReduce application or you may resort to some intermediate transfer mechanism. You should take into account that querying this storage system may become a bottleneck for the web application component. The schema for storing the MapReduce results is a choice of the groups. The system should maximize the number of queries that are supported by the database without requiring additional calculations.

1.3 Web Application

The web application should be a simple application that allows users to query the database by submitting into a text search box the city, the date (e.g. 2012-10-29), the parameter (rainfall - R, temperature - T, humidity - H) and the type of value (minimum - MIN, maximum - MAX or average - AVG). Obviously only those queries that your database supports should be allowed. For example, by default the system does not support asking for the hour of maximum rainfall in a day since only the daily total is calculated by MapReduce. You should test that the queries can also be submitted as a URL so that the website can be tested using a script.

1.4 Load Balancing

The implemented solution should be supported by a load balancer. The load balancer distributes request among different instances of the web application. It should detect that the web app is overloaded and start new instances and load balance requests to those instances. Your system should allow configuring the web application in a way that load at the web application can be simulated, e.g. by having a parameter that indicates how long to wait before answering a web request.

1.5 Implementation

The system and any of its parts can be implemented on a single machine or using an Amazon Web Services (AWS) account that will be provided. AWS provides components for running MapReduce, storing data, running computing instances and load balancing. It is up to each individual group to decide how and where to implement each of the solution's components.

2 Checkpoint

For the checkpoint, students should submit the solution of the MapReduce problem and a prototype version of the web application. The code submitted for the checkpoint will be evaluated on the following week's labs. The checkpoint submission should be made in the Fenix system until 17:00 on November 16th 2012.

3 Final Submission

The final submission should include the solution's code and a report describing the implemented solution and any measurements that support the design decisions and configurations. Each group should also submit a script that performs queries on the system with enough load to trigger the load balancing mechanism. The code of the final submission should be well organized and adequately commented. The final submission should be made in the Fenix system until 17:00 on December 14th 2012.