TÉCNICO LISBOA

Cloud Computing

Cloud Computing

Group 40
Final Report

77875 – Filippo Campagnaro

46425 – Rodrigo Bruno

67074 – Samuel Bernardo

Master's Telecommunications and Informatics Engineering

December 12, 2013

Contents

Introduction

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

1 Map Reduce implementation

1.1 Map

1.2 Combiner

1.3 Reducer

1.4 Incremental logs

2 Databases

2.1 PostgreSQL

2.2 DynamoDB

3 Web Site

We developed a web application to allow simple queries over the output data of the MapReduce application. The web application is developed using php and javaScript. Those languages are the most used for simple web applications to allow the user to access and querying a database. In fact in our first prototype we store the results of the MapReduce application in a postreSQL database. Anyway with php is also simple accessing to a BigTable on DynamoDB, as we did for the final application. Three kind of queries are permitted:

3.1 User Interface

With the website the user can access the data inserting the request specifications in the input fields. To permit this there are three form blocks in the interface, one for each query. After the right insertion of the specifications in the input fields the request is sent to a php page due to accessing the data stored in a webserver. The web interface is shown in figure 1 avali-

able at: server url: `ec2-54-200-205-56.us-west-2.compute.amazonaws.com`, load balancer url: `ec2-54-201-119-18.us-west-2.compute.amazonaws.com/`.

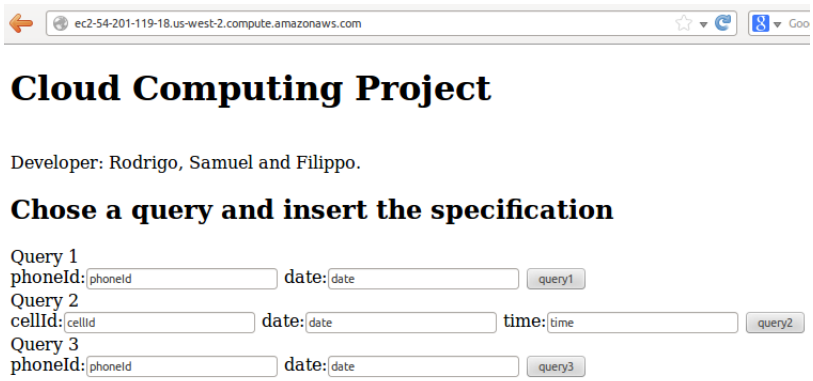

Figure 1: Illustrative screenshot of index.html

3.2 PHP: connection with data

For retrieving the data stored in the web server (in a DynamoDB BigTable) we use a php page. To allow stress test with software like 'siege' or 'jmeter', the querying specifications are passed to this page with the GET method, directly written in the URL of the page. Getting the attributes, the data are retrieved by the server. For the second and the third query the retrieved data are directly shown to the user, instead for the first query the data are stored in an hidden field for the last elaboration at the client web browser. The listing of the php page is attached in the code 3.2 and an example of the web application is shown in figure 2, available using the `http://ec2-54-200-205-56.us-west-2.compute.amazonaws.com/query.php?id=C1&date=2013%2F10%2F22&time=08&submit=query2`

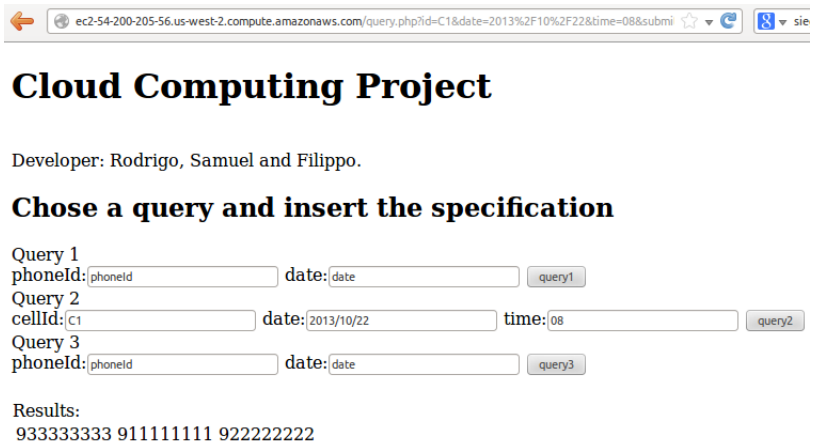

Figure 2: Second query in query.php

```
query.php
<?php
//AWS connection
require '../aws/aws-autoloader.php';
use Aws\DynamoDb\DynamoDbClient;
$client=DynamoDbClient::factory(array(
  'key' => 'AKIAJSRJRTI5GRQNROMA',
  'secret' => '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',
  'region' => 'us-west-2'
));
//FORM GET STUFF
$submit = $_GET['submit'];
$date = $_GET['date'];
$id = $_GET['id'];
$dateId=$date."-".$id;
$time = "";
$sql = "";
//DynamoDb getItem
if ($submit!="query2"){
  if($submit == "query1"){
    $result = $client ->getItem(array(
        'ConsistentRead' => true,
        'TableName' => 'CN_logs',
        'Key' => array('date-id' => array('S'=>$dateId)),
        'AttributesToGet' => array('value')
        ));
  $output = $result['Item']['value']['SS'];
  }
  else{
    $result = $client ->getItem(array(
      'ConsistentRead' => true,
      'TableName' => 'CN_logs',
      'Key' => array('date-id' => array('S'=>$dateId)),
      'AttributesToGet' => array('number')
    ));
    $output = $result['Item']['number']['SS'];
  }
}
else{
```

```
$time = $_GET['time'];
$dateId=$dateId.":".$time;
$result = $client ->getItem(array(
      'ConsistentRead' => true,
      'TableName' => 'CN_logs',
      'Key' => array('date-id' => array('S'=>$dateId)),
      'AttributesToGet' => array('value')
      ));
$output = $result['Item']['value']['SS'];
}
echo "<div id='queryResult'>Results: <input type='hidden' id='queryNumber' value='$submit'/>";
echo "<div id='out'>";
if($submit!="query1"){
  echo "<table>";
  for ($i=0;$i<sizeof($output);$i++)
    echo "<tr><td>$output[$i]</td></tr>";
  echo "</table></div>";
}
else{
  echo "</div><input type='hidden' id='campo' value='";
  for ($i=0;$i<sizeof($output);$i++)
    echo $output[$i];
  echo"'/>";
}
echo "</div>";
?>
```

3.3 JavaScript: last elaboration

As we saved the data in DynamoDB in a common table, the first query data needs one more elaboration step before be shown to the user. In fact we decided to elaborate the network logs by an unique MapReduce application and store them in the most generic way in a common BigTable. To do this we decide to store int the fields of the cells visited by a phoneId in a day also the time for having the sequence of the cells order by time. The first query require only the ordered sequence of the cells, not the time. That sequence is stored in a hidden field in the web page and by a javaScript function the time is deleted from the sequence before been shown to the user. This function is called when the page sent by the server is loaded in the web browser of the user, for doing less work as possible in the server.

```
<body onload="init();">
Results:
<input id="queryNumber" type="hidden" value="query1">
<div id="out"> C2 C1</div>
<input id="campo" type="hidden" value="00:24:00;C2 14:12:15;C1">
```

```
function init(){
if(document.getElementById("queryNumber").value=="query1")
document.getElementById('out').innerHTML=elaborateString(document.getElementById('campo').value);
}
function elaborateString(inputString){
  input=inputString.split(' ');
  output="";
  for (index=0;index<input.length;index++){
    item = input[index];
    item=item.split(";");
    output+=" "+item[item.length-1];
  }
  return output;
}
```

4 Load balancing and autoscaling

4.1 Instance

4.2 Autoscaling

4.3 Load balancing

5 Evaluation and Plots

6 Conclusion