UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Jürmo Mehine

Large Scale Data Analysis Using Apache Pig
Master's Thesis

Advisor: Satish Srirama
Co-advisor: Pelle Jakovits

Author: .... „.....“ May 2011
Advisor: .... „.....“ May 2011
Co-advisor: .... „......“ May 2011

Permission for defense
Professor: .... „.....“ May 2011

Tartu, 2011

Table of Contents

Introduction

Currently the field of parallel computing finds abundant application in text analysis. Companies, such as Google Inc. and Yahoo! Inc., which derive revenue from delivering web search results, process large amounts of data and are interested in keeping delivered results relevant. Because these companies have access to a large number of computers and because the amount of data to be processed is big, the search query processing computations are often parallelized across a cluster of computers. In order to simplify programming for a distributed environment, several tools have been developed, such as the MapReduce programming model, which has become popular, because of its automatic parallelism and fault tolerance.

The SciCloud [1] project at the University of Tartu studies the adaption of scientific problems to cloud computing frameworks and Pig is one of the most widely used frameworks for processing large amounts of data. For instance Yahoo! Inc. [2] runs the Hadoop framework on more than 25 000 computers and over 40 percent of the tasks use Pig to process and analyze data. This demonstrates Pig's use in a commercial setting with data that does not fit on a single machine, but is distributed across tens or hundreds of machines to decrease processing time.

Pig operates as a layer of abstraction on top of the MapReduce programming model. It frees programmers from having to write MapReduce functions for each low level data processing operation and instead allows them to simply describe, how data should be analyzed using higher level data definition and manipulation statements. Additionally, because Pig still uses MapReduce, it retains all its useful traits, such as automatic parallelism, fault tolerance and good scalability, which are essential when working with very large data sets.

Because of its successful adoption in commercial environments, ease of use, scalability and fault tolerance, the SciCloud project decided to study the Pig framework to determine its applicability for large scale text analysis problems the project deals with. This work is a part of the research and its goal is to give a full understanding of how to use Pig and to demonstrate its usefulness in solving real life data analysis problems.

This is achieved by first describing how the Pig framework works and how to use the tools it provides. An overview is given of the Pig query language, describing, how to write statements in it and how to execute these statements. This information is illustrated with examples. Secondly a real world data analysis problem is defined. This problem is solved using Pig, all the steps taken in the solution are documented in detail and analysis results are interpreted. The example problem involves analyzing news feeds from the internet and finding trending topics over a period of time.

This work is divided into nine chapters. In chapter 1 the technologies used in the data analysis task are described. Chapter 2 gives an overview of how to use Apache Pig. Chapter 3 describes the example data analysis problem in more detail. Chapter 4 describes how to set up an environment to test the Pig applications, created as part of this work. Chapter 5 describes the implementation details of the data analysis application, which was created. Chapter 6 goes through the usage of the program step-by-step using example input data and analyzes the results of the program's work. Chapter 7 describes problems encountered in implementing the data analysis tool and their solutions and work-arounds. Chapter 8 has discussions about software products, which are related to Pig. Chapter 9 summarizes the work, giving its conclusions.

Chapter 1
Technologies Used

This chapter lists and describes different tools and standards used for accomplishing the goals stated in chapter 3. The roles, each of these technologies play within the system, are explained. Because Pig operates on top of the Hadoop infrastructure, it is important to give sufficient background information about Hadoop and the research that lead to the creation of Pig. Hadoop is used with the Eucalyptus open source cloud infrastructure in the University of Tartu's SciCloud project. These technologies are explained in detail below. In addition a description of the RSS web feed standard is given, because RSS files are used as input for the data analysis task proposed.

1. 1 SciCloud Project

Because Pig is oriented at processing large amounts of data in a distributed system, it is important to test it on a cluster of computers, not only a single machine. The SciCloud project provids the necessary hardware and software infrastructure to deploy Pig programs on a cluster.

The Scientific Computing Cloud or SciCloud project was started at the University of Tartu with the primary goal of studying the scope of establishing private clouds at universities. Such clouds allow researchers and students to utilize existing university resources to run computationally intensive tasks. SciCloud aims to use a service-oriented and interactive cloud computing model, to develop a framework, allowing for establishment, selection, auto scaling and iteroperability of private clouds. Finding new distributed algorithms and reducing scientific computing problems to the MapReduce algorithm are also among the focuses of the project [1, 3].

SciCloud uses the Eucalyptus open source cloud infrastructure. Eucalyptus clouds are compatible with Amazon EC2 clouds [4], which allows applications developed on the private cloud to be scaled up to the public cloud. The initial cluster, on which SciCloud was set up, consisted of 8 nodes of SUN FireServer Blade system with 2-core AMD Opteron processors. The setup has since been extended by two nodes of double quad-core processors with 32 GB memory per node and 4 more nodes with a single quad-core processor and 8 GB of memory each.

1. 2 Eucalyptus Cloud Infrastructure

Eucalyptus was designed to be an easy to install, „on premise“ cloud computing infrastructure utilizing hardware that the installing comapny (or in this case university) has readily available. Eucalyptus is available for a number of Linux distributions and works wth several virtualization technologies. Because it supports the Amazon Web Services interface, it is possible for Eucalyptus to interact with public clouds. [5]

Eucalyptus components include the Cloud Controller, Cluster Controller , Node Controller, Storage Controller, Walrus and Management Platform. The Cloud Controller makes high level scheduling decisions and manages the storage, servers and networking of the cloud. It is the entry point into the cloud for users. The Cluster Controller manages and schedules execution of virtual machines on nodes in a cluster. Node Controllers control the execution of virtual machines on a node. A Node Controller is executed on all nodes which host a virtual machine instance. The Storage Controller implements network storage. It can interface with different storage systems. Walrus is the Eucalyptus storage service created to enable storage of persistent data organized as buckets and objects. It is compatible with Amazon's S3 and supports the Amazon Machine Image (AMI). The Management Platform can be used as an interface to communicate with the different Eucalyptus services and modules as well as external clouds. [5]

1. 3 Hadoop Framework

Hadoop is a software framework for computations involving large amounts of data. It enables distributed data processing across many nodes. It is maintained by the Apache Software Foundation. The framework is implemented in the Java programming language and one of the main contributors to the project is Yahoo! Inc. Hadoop is based on ideas presented in Google's MapReduce and Google File System papers. This section outlines the basic principles of working with Hadoop and describes the MapReduce programming model, which forms the conceptual basis for Hadoop. [6]

Hadoop provides a part of the infrastructure in this project, because the main tool used for data processing and analysis, Pig, is built on top of it and utilizes the underlying MapReduce model to execute scripts. However, no part of the final application itself is written as MapReduce programs, because the idea is to demonstrate Pig as an abstraction layer on top of MapReduce. Before any distributed data processing can occur Hadoop has to be installed on the system.

1. 3. 1 MapReduce Programming Model

In order to understand, how Hadoop works and what kind of problems it is meant to solve, some knowledge is first required of the MapReduce programming model, it implements.

In 2004 Jeffrey Dean and Sanjay Ghemawat, two researchers at Google, Inc. published the paper „MapReduce: Simplified Data Processing on Large Clusters“ [7]. This work outlined a programming framework designed for processing large amounts of input data in a distributed file system. The proposed programming model and implementation make use of clusters of machines to split input tasks between different nodes, compute intermediate results in parallel and then combine those results into the final output. The parallelization of tasks allows for faster processing times while the programming framework hides the details of partitioning data, scheduling processes and failure handling from the programmer.

This subsection summarizes the paper „MapReduce: Simplified Data Processing on Large Clusters“ [7]. It will give a brief overview of the background of MapReduce, describe the details of the programming model as well as Google's implementation of it.

The motivation to create MapReduce came from Google's need to process large amounts of data across a network of computers. In order to do this effectively the solution would have to handle scheduling details, while empowering the user to only write the application code for a given assignment.

The inspiration for MapReduce comes from the map and reduce functions used in functional programming languages, such as Lisp. In functional programming these functions are applied to lists of data. The map function outputs a new list, where a user specified function is applied to each of the elements of the input list (e.g multiplying each number in a list by a given number). The reduce function combines the list elements by a user given function (e.g adding up all numbers in a list) into a new data object.

Google's MapReduce is implemented in the C++ programming language. It takes a set of input records and applies a map function to each of them. The map function is defined by the programmer and it outputs a list of intermediate records – the input for the reduce function. The reduce function takes the list of intermediate records and processes them in a manner specified by the programmer. The reduce function returns the final output.

These actions are done in a cluster of computers where different computers are assigned different chunks of input or intermediate data to process. The splitting and assigning of tasks is handled by an instance of the program called the master. The other instances running are called workers. The master also collects intermediate results and passes them on to workers to perform reduce tasks. The programmer using MapReduce only has to specify the two functions that will be applied to each record in the list in the map and reduce steps. Any other aspects of running the job, such as parallelism details and task splitting are handled by the MapReduce library.

MapReduce works with records defined as key-value pairs. In the map step initial keys and values are retrieved from the input data and intermediate pairs are produced. The reduce step is applied to all records with the same intermediate key, thereby creating the final output. An additional combine function may also be written to further process the intermediate data.

The input data is split into *M* pieces of typically 16 to 64 megabytes. Intermediate key-value pairs output by the map function are partitioned into *R* partitions using a function defined by the user. The program is run concurrently on several machines with one node running the master task. The master assigns map and reduce tasks to worker nodes. Workers assigned with a map task parse key-value pairs from input data and pass them to the user's Map function. Intermediate results output by map tasks are buffered in memory and periodically written to disk. The master assigns tasks to reduce workers by giving them the locations of intermediate key-value pairs to work on. The reduce worker sorts intermediate keys so that occurrences of the same key are grouped together. The MapReduce process is illustrated in figure *1*.

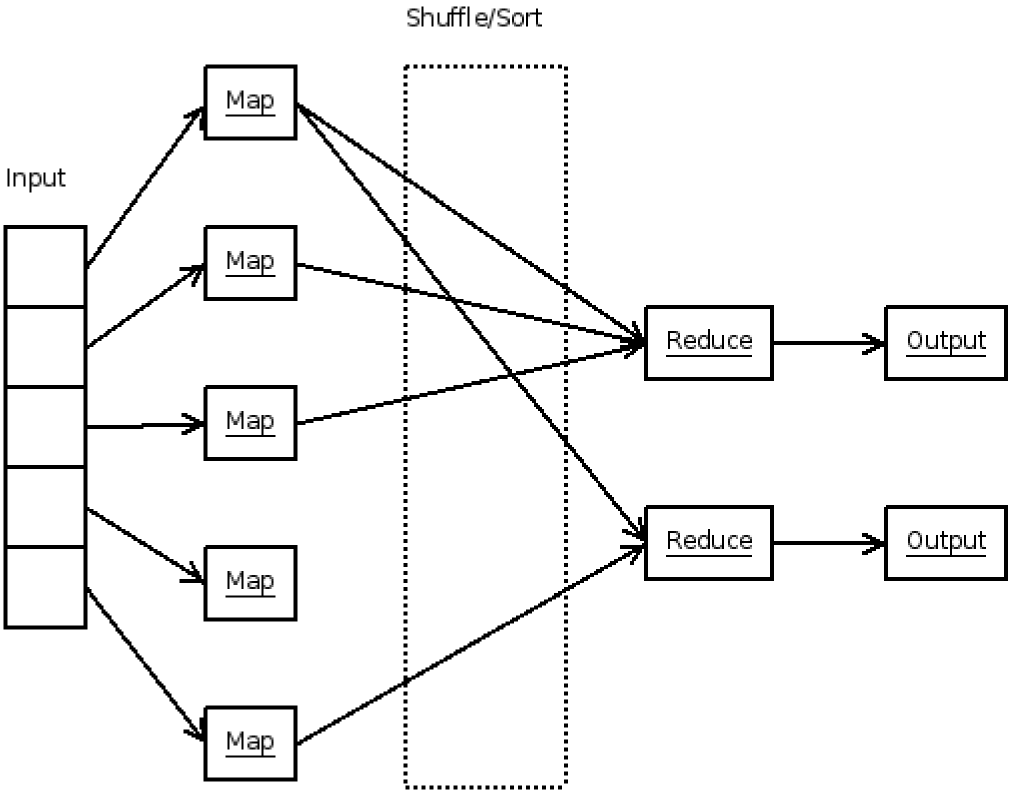

Figure 1. *Parallelization using MapReduce [8]*

The MapReduce library also has built-in fault tolerance to deal with failures of worker nodes. The master periodically pings all workers. If a worker does not respond, it is marked as failed. Any map tasks assigned to a failed node are reassigned and re-executed. Completed reduce tasks do not need to be re-executed as their results are written to the global file system.

In the case of master failure the MapReduce computation is aborted and the calling program will have to restart it.

The master uses information about the locations of input data in the cluster and tries to assign map tasks to nodes, which contain a copy of the corresponding input data. If this is not possible, the master tries to schedule the task to a machine near the one with the right input data in the network. This is done in order to save time by limiting network traffic between nodes.

A common problem with many MapReduce jobs is some nodes taking a long time to finish a task. This can happen for a number of reasons, such as other processes running on the machine. The Google MapReduce implementation uses the following mechanism to deal with this problem. When the job nears completion the master schedules backup executions of the remaining in-progress tasks to ensure that they are completed in reasonable time.

Because sometimes it is useful to combine the data before sending it to the reduce function, the MapReduce library allows the user to write a combiner function, which will be executed on the same node as a map task. The combiner function is useful when there is significant repetition in the intermediate keys. The combiner is used to partially merge these results before sending them over the network to the reducer. Typically the same function is used for the reduce and combine tasks, but the combine result is written to an intermediate file which will be sent to a reduce task, while the reduce output is written to a final output file.

One feature of the library is that it allows the user to specify the format of the input data and how to convert it into key-value pairs. In most cases the input is handled as text files, meaning the keys are line numbers and the value is the string contained within this line. However any other way of reading input data can be defined by implementing a different reader interface.

1. 3. 2 Hadoop Implementation

Hadoop consists of three sub-projects: Hadoop Common, Hadoop Distributed File System (HDFS) and MapReduce. Projects closely connected to Hadoop include, in addition to Pig, Hive, Mahout, ZooKeeper and HBase [9]. Some of these projects are described in chapter 8, Related Work. This sub-section gives an overview of Hadoop's design focusing on Hadoop MapReduce and HDFS.

Hadoop components and sub-projects are joined through Hadoop Common which provides the utilities needed for Hadoop tools to work together. Hadoop Common contains the source code and scripts necessary to start Hadoop. [6]

HDFS stands for Hadoop Distributed File System. Because HDFS is meant to run on clusters with large numbers of nodes, it was designed to detect and automatically recover from hardware failures. HDFS was created for large data sets with sizes in gigabytes and terabytes. It should scale to hundreds of nodes on a cluster. In terms of computations, involving the data, HDFS is designed in a way to perform computations on a node that is close to the input data it processes, thus saving time. Files in HDFS are split into blocks, typically of size 64 MB. Each block is replicated across several nodes to ensure data availability if a node goes offline. [10]

Hadoop's MapReduce engine is based on Google's MapReduce specification. The MapReduce Job Tracker assigns jobs to available Task Tracker nodes, which run Map and Reduce tasks. If a Task Tracker fails, its tasks are rescheduled. Job Tracker failure causes the current job to stop. Unlike Google's MapReduce which is written in C++, the Hadoop project is implemented in Java. [6]

1. 4 Pig

The core tool employed in this work is Pig. It is used to perform the data processing and analysis tasks necessary to solve the proposed problem.

Pig was originally a research project at Yahoo! Inc. and later became an independent sub-project of Apache Software Foundation's Hadoop project. Pig is a large scale data analysis platform with a high level data description and manipulation language. It was designed to make use of the MapReduce programming model implemented in Hadoop [11]. As of September 2010 Pig is no longer a Hadoop sub-project, but a top-level Apache project [9].

Pig abstracts away from the MapReduce model by introducing a new high-level language for data aggregation. A common use of MapReduce is aggregating information from large text files such as log files or HTML documents. Pig acts as a uniform interface for many common data aggregation tasks (counting, filtering, finding records by certain parameters etc). This means that the programmer no longer has to write MapReduce programs in Java, but instead can write shorter Pig Latin queries or batch scripts to accomplish the data manipulation tasks needed. The Pig platform then generates, optimizes and compiles MapReduce tasks from the Pig Latin statements and commits them to Hadoop for execution. Pig Latin queries can also be run in local mode without parallelism [9].

A detailed description of Pig usage is given in chapter 2.

1. 5 RSS Standard

News stories published online are used as input for the data analysis problem specified in chapter 3. The stories are gathered by downloading RSS web feed files. This section gives an overview of the format RSS files follow.

RSS is a format specification for web syndication. The acronym stands for Really Simple Syndication [12]. „Web syndication“ is a phrase commonly used to refer to the act of creating and updating web feeds on a website, showing summaries of the latest additions to the content of this site [13]. Many online news sources as well as blogs publish web feeds using the RSS standard and these are what provide the input for the application that was developed during the course of this project. The files containing the RSS feeds of several news sites were periodically downloaded for later processing and analysis, using the implemented application.

RSS is based on XML, the Extensible Markup Language. The RSS standard defines, what XML elements and attributes have to be present in a feed document, and how a feed reader or aggregator should interpret them. The latest version of the RSS standard at the time of this writing is 2.0.11 [12]. A description of some of the elements in the specification follows.

The top-level element of any RSS document is the „rss“ element, which is required to have a „version“ attribute with the value „2.0“. It must also contain exactly one „channel“ element. The „channel“ element has three obligatory elements: „description“, „link“ and „title“ and a number of optional elements. The „description“ element contains a summary of the feed, „link“ has the URL of the feed publishing web site and the title of the feed is held in the „title“ element [14].

Among the optional elements of „channel“ there can be any number of „item“ elements. These contain data about the content additions published on the web site. Usually an „item“ element is added to the feed every time a new content item (blog post, news story etc.) is published on the site. Among the possible sub-elements of „item“ are „title“, „description“, „link“ and „pubDate“. Of these, it is required to include either a „title“ or a „description“. The „title“ contains the headline of the entry. The „description“ may contain a summary of the item or its full content – this decision is left up to the publisher. The „pubDate“ element contains the publication date of the item. All date and time values in RSS have to be formatted in accordance with the RFC 822 Date and Time Specification with one difference: in RSS a four-digit year number notation is allowed as well as a two digit one. An example of a date-time value in RSS is „`Mon, 05 May 2011 12:11:00 +0200`“. The „link“ element contains the URL of the actual published content [14].

An example of a valid RSS 2.0 web feed document follows.

```
<?xml version="1.0"?>
<rss version="2.0">
     <channel>
          <title>The Daily Aggregator</title>
          <link>http://example.com</link>
          <description>The Latest News</description>
          <lastBuildDate>Fri, 06 May 2011 08:32:00
GMT</lastBuildDate>
          <webMaster>mail@example.com</webMaster>
          <item>
               <title>Life Discovered on Mars</title>
               <link>http://example.com/news/life-on-
mars</link>
               <description>Scientists have discovered
intelligent life forms on Mars.</description>
               <pubDate>Fri, 01 Apr 2011 01:00:00
GMT</pubDate>
          </item>
          <item>
               <description>There are currently no news to
report</description>
               <pubDate>Fri, 01 Apr 2011 02:30:00
```

```
GMT</pubDate>
          </item>
     </channel>
</rss>
```

Chapter 2
Pig Usage

This chapter provides instructions on how to use the Apache Pig tool. The use of Pig Latin data types as well as some of the most common functions is explained. Also different approaches to running Pig scripts are discussed.

2. 1 The Pig Latin Language

The Pig query language is called Pig Latin. A Pig Latin program is a series of statements, where each statement does one of three things: reading data from the file system, applying transformations to data, outputting data. The language is like SQL in that it allows the user to run queries on relations. The exact structure of data is discussed below. Keywords available in Pig Latin include JOIN, GROUP and FILTER [15, 16]. All Pig Latin usage instructions, presented in this section, are compiled based on information in the resource [16].

The „`LOAD`" statement allows the user to read data from the file system into Pig. Typical usage looks like this:

```
Data = LOAD '/dir/input_data.txt' USING PigStorage(',') AS
(field1:chararray, field2:int, field3:float);
```

In this example „`Data`" is a variable to refer to the relation just created. This is similar to a SQL table or view name. „`/dir/input_data.txt`" is the full path to the file, from which input is being read.

„`PigStorage`" is a storage function. This means it is used to describe, how to parse the data being read and in what format to save it, when it is written. The „`PigStorage`" function adds each line of an input file as a record to the new relation, treating the lines as fields, separated by the string it receives as a parameter (in this example, a comma). If omitted, the separator string defaults to a tab („`\t`").

The „`AS`" clause is optional and, if given, it allows the programmer to specify aliases for the fields in the relation, as well as their data types. In the example above the lines are split into three fields with aliases „field1“, „field2“, „field3“ and data types „chararray“, „int“ and „float“ respectively. The aliases are used to refer to fields later when manipulating data. If they are not specified the fields can still be accessed using the positional notation identifiers $0, $1, $2 etc.

The simple data types, that Pig uses, are „int“ and „long“ for signed integer values and „float“ and „double“ for floating point values. In addition Pig has two array types: „chararray“ for UTF-8 strings and „bytearray“.

The complex data types in Pig are: tuple, bag and map. A map is a set of key-value pairs. A tuple is an ordered set of fields. A bag is a set of tuples. All Pig relations are bags. Relations are similar to tables in a relational database management system. In this comparison the tuples in a relation would correspond to individual rows in a database table. One of the differences, however, is that all tuples in a bag are not required to have the same number of fields. Also, a tuple can contain other tuples as well as bags in its fields. A bag contained in a tuple is also called an inner-bag, while relations are called outer-bags. In this work the words „relation“, „bag“ and „table“ are all used to refer to outer-bags.

In Pig Latin syntax tuples are enclosed in parentheses. The values of their fields are separated by commas:

```
(first value,7,1.2F)
```

Inner-bags are enclosed in curly brackets and their tuples are separated by commas:

```
{(first value, 7, 1.2F),(second value, 8, 3.0F)}
```

Maps are key value pairs enclosed in straight brackets and separated by commas. The key and value are separated by a pound sign:

```
[key1#value1,key2#value2]
```

The „FOREACH“ operator is used to apply a function to every tuple in a relation. An example of the usage of „FOREACH“ looks like this:

```
NewData = FOREACH Data GENERATE $0 + $1;
```

This example takes the first and second field in every tuple in the relation „Data“, adds them up and saves the sums in a new relation called NewData.

If tuples in a a relation contain inner-bags, it is sometimes useful to know the number of tuples in these bags. This can be achieved using the „FOREACH“ operator together with the „COUNT“ function as shown below:

```
NewData = FOREACH Data GENERATE $0, COUNT($1);
```

This saves into „NewData“ the values in the first column of „Data“ and the corresponding numbers of tuples in the second column.

It is a common task to reduce the number of records studied to only those that fit certain criteria. In SQL this is accomplished by using the „WHERE“ keyword and specifying the conditions the output records must satisfy. For this same task Pig uses the „FILTER“ operator.

```
FilteredData = FILTER NewData BY $1 > 10;
```

This examples goes through the „NewData“ relation and saves in „FilteredData“ only those records, where the value in the second field is greater than ten.

To eliminate duplicate tuples from a bag the „DISTINCT“ keyword is used. This has the effect of making sure, that each tuple is represented in the bag exactly once.

```
DistinctData = DISTINCT Data;
```

Another task familiar from SQL is that of joining tables. This essentialy means outputting a new table by finding a column in either table, being joined, and merging the rows where the values in these columns are the same. The equivalent syntax in Pig Latin looks like this:

```
NewData = JOIN Rel1 BY Field1, Rel2 BY Field2;
```

In this example relations „Rel1“ and „Rel2“ are joined by matching rows where „Field1“ in „Rel1“ equals „Field2“ in „Rel2“.

Pig allows both inner and outer joins. If not specified otherwise the join is an inner join. The above example query can be rewritten to use an outer join as follows:

```
NewData = JOIN Rel1 BY Field1 LEFT OUTER, Rel2 BY Field2;
```

In this case a left outer join is used. The keywords „RIGHT“ and „FULL“ are also permitted to achieve other types of outer joins.

2. 2 Running Pig

In chapter 3 a data processing and analysis problem will be solved, using the tools provided by Pig. First, however, an introduction must be made to the technical details of running Pig. This section gives instructions on how to get started using Pig. It is based on information found in the resources [15] and [17].

Pig Latin statements can be executed either from a batch script or interactively using the Grunt shell. The user can also choose whether to run Pig in MapReduce mode, to have the actions parallelized using Hadoop, or to execute statements in local mode without parallelization.

The Grunt shell is an interactive command line program for processing Pig Latin statements. Once Pig has been installed, it can be started with the command „`pig`“. In order to specify the execution mode („MapReduce“ or „Local“), an additional command line parameter „`-x`“ with a value must be specified at startup.

```
$ pig -x local
```

is used to to run Pig in local mode.

```
$ pig -x MapReduce
```

is used to run Pig in MapReduce mode. When running Pig without the execution mode parameter, MapReduce mode is chosen as the default. After starting Pig the user will be presented with the Grunt shell prompt, which looks like this:

```
grunt>
```

The user can now enter, one by one, whatever Pig Latin statements the user wants and they will be executed.

There are two ways to execute a Pig batch script. One is to start Pig and give the path to the script file as the argument:

```
$ pig myscript.pig
```

or:

```
$ pig -x local myscript.pig
```

The other way of running a script is to first start the interactive Grunt shell and then issue the „exec“ or „run“ command to start the batch job:

```
$ pig

grunt> exec myscript.pig;
```

Pig programs also accept parameters from the command line. The values of these parameters can be substituted into the script. On the command line the parameters take the following format:

```
-param param1=value1 -param param2=value2
```

In the script file their values are accessed and substituted in the following manner:

```
new_data = FILTER data BY field = '$param1';
```

In this example the value of „param1“ is substituted for the string, against which the „FILTER“ operation matches values in the relation.

The program's parameters may also be read from a file, whose name is specified at the command line. The parameters in the file have to be in the following format:

```
param1 = value1

param2 = value2
```

The path to the parameter file is given on the command line with the „-param_file“ flag.

In addition to these methods, a programmer can also embed Pig Latin statements into Java programs. In order to do so the programmer needs to add the Hadoop and Pig core library jar files to the Java project and import classes from them as necessary.

To run Pig queries from Java code, an instance of the „PigServer“ class has to be created and the execution mode („local“ or „mapreduce“) specified. Single queries can be run by calling the function „registerQuery“ on the PigServer instance and providing the Pig Latin statement as the argument. To store a relation in the file system the „store“ method has to be called on the PigServer object, and the name of the relation and the output path have to be provided as arguments.

The Pig API also allows executing Pig Latin code stored in external script files from a Java application. For this purpose the PigServer class has the „registerScript“ method, which takes the path to the external script as its parameter. Command line parameters for the script can be provided in the form of Java HashMap objects. [18]

An important thing to note about Pig, is how and when it actually executes the MapReduce jobs created from Pig statements. In general Pig does not read in any actual data until it reaches a „DUMP“ or „STORE“ statement in the script. Upon reaching this point it will read the data and process it. [15]

2. 3 Usage Example

To better understand Pig Latin, a complete data processing script written in Pig Latin is presented in this section, as an example of Pig usage. A data processing task is defined and solved using a Pig script. Comments on how the program works, its input and output data are also provided.

The input for the sample script is provided in the form of text files. The text files hold, in a tab separated format, a database of video game scores and their owners. The first file „score.in“ holds records of scores, achieved in different games, along with the name of the game and the person, who owns the score. A typical line in this file may look like this:

```
Alice     Asteroid Storm 2432
```

In this case „Alice“ is the name of the player, „Asteroid Storm“ is the game and the score is 2432 points.

The second file, named „player.in“, contains the names and ages of players. It can contain lines such as this:

```
Alice     15
```

Here Alice is the name and the age is 15 years. This data is processed to find the average age of the players of different games. The data used in this example is fictitious and serves only to illustrate Pig usage.

Two pieces of knowledge are sought from this data: who holds the highest score for each separate game and what is the average age for players of each game. In a real study this data could be useful for a game development company to determine, which demographics different types of games appeal to. The script file is named „high-score.pig“. The following is the sample Pig script listing.

```
scores = LOAD '$score_input' USING PigStorage('\t') AS (name:
chararray, game: chararray, score: int);

game_scores = GROUP scores BY game;

top_game_scores = FOREACH game_scores {

     top_scores = ORDER scores BY score DESC;

     top_score_bag = LIMIT top_scores 1;

     GENERATE FLATTEN(top_score_bag);

}

DUMP top_game_scores;


players = LOAD '$player_input' USING PigStorage('\t') AS
(name:chararray, age:int);

player_scores = JOIN players BY name, scores BY name;

grouped_player_scores = GROUP player_scores BY game;

average_ages = FOREACH grouped_player_scores GENERATE group,
AVG(player_scores.age) AS age:float;

DUMP average_ages;
```

The script features two parameters to be given from the command line: „`score_input`“ and „`player_input`“. These are used to communicate to the script the locations of the two input files. To run it on the files „score.in“ and „player.in“ (both located in the file system directory „/path“), the script is run using the following command:

```
$ pig -param score_input=/path/score.in -param
player_input=/path/player.in high-score.pig
```

During the execution of this script the contents of the file „score.in“ are first read and the entries there saved in the Pig relation called „scores“. The columns in the relation are given the names: „name“, „game“ and „score“. A grouping is then performed on the „scores“ relation to separate the results for different games into groups. This results in the creation of the relation „game_scores“, which holds game names in one column and inner-bags, consisting of all the entries containing this game, in another.

The next statement demonstrates the use of nested blocks in Pig. Scores for every game are ordered and only the top score tuple is generated as output. The following „DUMP“ statement outputs the resulting bag to the terminal. In a real world Pig application it is usually more practical to save the output into files using the „STORE“ command. Below is an excerpt from the output.

```
(Bob,Alien Fighter,1500)

(Alice,Asteroid Storm,2432)
```

Next, the second input file, „player.in“ is read. Now the „players“ and „scores“ relations are joined and again grouped by game. With data about ages and games now in the same relation, it is possible to determine the average playing age for each game using the „AVG“ function. Below is a sample of the output generated.

```
(Space Run,15.0)

(Planet Rider,18.5)
```

The above script can also be executed from a Java application. Below the source code for a single class Java application, which uses the script, is given.

```
import java.util.HashMap;

import org.apache.pig.PigServer;

class HighScore {

     public static void main(String[] args) throws Exception
{

          PigServer pigServer = new PigServer("local");

          HashMap <String,String> params = new HashMap
<String,String>();
            params.put("score_input", "/path/score.in");
            params.put("player_input", "/path/player.in");
            pigServer.registerScript("high-score.pig", params);
            pigServer.registerQuery("under_age = FILTER
average_ages BY age < 18;");
            pigServer.store("under_age", "under_age.out");
      }
}
```

When running the Java program a PigServer instance is created using local execution mode. The necessary input file for the script are given in the form of a Java HashMap object containing parameter names and values. In addition to running „high-score.pig“, the Java program also performs an extra operation on a relation created in the script. It filters the „`average_ages`“ bag to find only the tuples where the „`age`“ value is less than 18. The resulting relation is finally saved to the output file „under_age.out“.

Chapter 3
The Problem

Many web sites publish news feeds to keep their readers informed of updates. Because this is common practice, standards have been developed, dictating the exact format, which news feeds need to follow, in order to be correctly read by feed aggregators. These formats include RSS and Atom standards. For the purpose of studying the applicability of tools provided by Pig to large scale data analysis, a problem, using RSS files as input, was proposed and solved using these tools. This chapter specifies the details of the exact data analysis and processing problems solved.

3. 1 Expected Results

This section describes in-depth the questions, to which the data analysis application needs to answer. Three different kinds of results are expected from the application:

1. a list of popular words in news stories by date
2. the number of co-occurrences of certain words in news stories
3. a list of news items matching a given search query.

The first result to look for in the data, is which words occur more frequently in the news. These results are grouped by date, to give an idea, how the topics covered by news sources change over a period of time. As a major world event (e.g a war, elections or a natural disaster) starts getting coverage on the news, the words commonly used to describe this event will become more frequent in news stories. With the passage of time the coverage of the event diminishes, causing a drop in the number of occurrences of the respective words in the news. The data analysis application has to find these changes and, upon querying, give a list of the most frequent relevant words in the news for every day over a period of time.

To allow for more detailed querying and trends analysis, functionality is needed to find out, by date, how often some words occur together in news stories. The program accepts as its parameters an arbitrary number of words and counts all the news stories, grouped by date, where these words occur together. These results are also grouped by news source to show, how coverage of a topic differs from one publication to another.

Also search functionality is created to find all news items that match a given regular expression. Whereas the above described functionality only outputs frequencies in the occurrences of words, the regular expression gives full news stories. The user gets to specify a string to look for and the program will search all its stored news stories for it and output any, which match the expression. This provides the user with flexible full-text search functionality.

3. 2 Constraints

The solution presented is not intended to address all issues of automated web feed analysis, but instead to serve as an example of Pig usage. Therefore certain constraints were established to limit the scope of the final application. The purpose of these limitations is to focus attention on the core aspects of the problem while ignoring non-essential issues.

For the purpose of simplicity only feeds following the RSS format were used as input. This was done to avoid having to adapt the data processing algorithms to different kinds of input (e.g Atom feeds). Not all websites publishing RSS feeds follow the standard with complete accuracy so only ones, which adhere to the standard closely were selected for analysis.

Full news stories are usually not published in RSS feeds. Instead they contain the story's title, a synopsis and a link to the full story. Although accessing the full news story would give more complete insight into the event described, the scope of this problem is limited to only looking at the title and description given in the feed. This is done to avoid the additional problems that following news links would introduce. Because websites have different designs, all news sources would have to be first manually analyzed to identify the portion of the page that holds the actual news story. This information would have to be integrated into the data processing algorithms to analyze it. Also, after the processing method for a web site has been established, this information is subject to change, which means a change to the web page design could break the algorithm. Alternatively machine learning techniques could be applied to automate the process of identifying the necessary segment of the web page, but this approach falls outside the scope of the work.

Of all possible news sources only a few are selected to test the application. These are all sources that publish news in English. Other languages are avoided to prevent the need for translating search keywords. Using more than one language in the input data would cause difficulties in comparing analysis results, because a search word in one language may not return meaningful results in another. While integrating an automated translation module into the system would help solve this problem, it is not within the scope of the work.

The system collects news stories from sources in different time zones. However, for reasons, which are discussed in detail in chapter 7, the publishing times of news stories are not normalized to a single uniform time zone. This causes inaccuracies in the output data, but the loss of precision is relatively small when the size of the time window is one day.

Chapter 4
Application Deployment

The applications created were deployed on an instance of a SciCloud image. This chapter gives instructions on how to set up the applications as well as the tools used to run them on SciCloud. Information on initializing the working environment and configuring the applications is given here.

4. 1 Environment Setup

This section describes how to start an instance of an image on SciCloud. All instructions, presented in this section, are based on information in the SciCloud Manual [19].

Before any work can be started with SciCloud, the person wishing to deploy the project first has to apply for an account on the cloud. This can be done on the web page at https://scicloud1.mt.ut.ee:8443/ Once access has been granted, the user can log in to SciCloud at the same address and download the Eucalyptus credentials. When the credentials have been downloaded, the user must unpack them into a „.euca“ folder. In that folder, using the Bash command line, the environment can be set up with the command

```
$ source eucarc
```

A key pair must then be created and registered with the system. This is what the Euca2ools „`euca-add-keypair`“ command is for. The user has to specify a key pair name and a file to save it to, thus:

```
$ euca-add-keypair user > user.private
```

The „`euca-run-instances`“ command is used for running instances of images. It allows the user to specify the key name, number of instances and the type of instance to run as values of the command's „`-k`“, „`-n`“ and „`-t`“ flags respectively. The ID of the instance is required as the command's parameter. An example of running this command is as follows:

```
$ euca-run-instances -k user -n 1 -t m1.large emi-590A1342
```

To connect to a running instance of the image, the SSH remote login client is used. For a successful login the private key file has to be referred to in the value of the „-i“ flag. The user should log in to the IP address of the instance as the „root“ user as exemplified below

```
$ ssh -i user.private root@172.17.37.187
```

To upload and download data to and from the instance the „scp“ remote file copy program can be used, again, using the private key file name as the identity credential. To copy directories recursively, use the „-r“ flag with the „scp“ command.

```
$ scp -i user.private -r /path/dir/ root@172.17.37.187:/root
```

4. 2 Tools Setup

Once logged in to the started instance, the necessary tools can be set up. Firstly, a software repository has to be added to the list of software sources. This is done by adding the following line to the file „/etc/apt/sources.list“ on the instance

```
deb http://archive.canonical.com/ lucid partner
```

The file can be edited using a terminal-based text editor, such as „nano“ or „vim“, or with an „echo“ command redirecting the output to append to the file. This allows for installing several necessary packages, which are by default not available, on the instance. To update the package list, run the command

```
# apt-get update
```

Now it is possible to install the required software. This is done with the command

```
# apt-get install sun-java6-jdk wget ssh
```

Next Hadoop has to be installed on the virtual machine. For this a Hadoop version, packaged in a „tar.gz“ archive, is downloaded from one of Apache's download mirrors. The version used for this project is 0.20.2

```
# wget
http://www.ecoficial.com/apachemirror//hadoop/common/hadoop-
0.20.2/hadoop-0.20.2.tar.gz
```

Once the download has finished, the archive can be unpacked with the command

```
# tar xvzf hadoop-0.20.2
```

Because Hadoop needs to be able to log into the local machine without a password using the SSH client, an RSA key is generated using the „ssh-keygen“ utility and added to the list of authorized keys.

```
# ssh-keygen -t dsa -P "" -f /root/.ssh/id_dsa

# cat /root/.ssh/id_dsa.pub >> /root/.ssh/authorized_keys
```

Next Pig can be installed similarly to the way Hadoop was installed earlier. Again a „tar.gz“ archive containing the software has to be downloaded and unpacked. The version of Pig used in this case is 0.8.1.

```
# wget http://www.fightrice.com/mirrors/apache/pig/pig-
0.8.1/pig-0.8.1.tar.gz

# tar zxvf pig-0.8.1.tar.gz
```

Binary files and scripts required for running Hadoop are located in the directory „/root/hadoop-0.20.2/bin“. The directory holding the binary file required to run Pig is „/root/pig-0.8.1/bin“. By default neither of those directories is in the machine's „PATH“ enviroment variable. This means that the programs can only be run by giving the full path to the file to execute. To make running the programs more convenient, the „PATH“ variable should be modified in the „/root/.bashrc“ file to include the executable directories for Pig and Hadoop. The directive to be added to the file, is as follows:

```
export PATH=$PATH:/root/hadoop-0.20.2/bin:/root/pig-0.8.1/bin
```

Now each time the Bash command line is started, the necessary commands will be available.

It is also useful to set the „JAVA_HOME“ environment variable in „/root/.bashrc“:

```
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
```

When the supporting infrastructure has been set up, the news collection and analysis applications can also be installed on the instance. The applications are available on the CD accompanying this thesis. They can be uploaded to the instance using the „scp“ remote file copy program.

4. 3 Applications Installation

On the CD the file „News.jar“ and the directories „pig“, „src“ and „crawler“ can be found. „News.jar“ holds the built Java application meant for news analysis. The „pig“ directory contains separate Pig scripts, all of which the „News“ program needs to run. The directory „crawler“ contains the feed collector application files.

The crawler application does not have to be set up on the cloud, as the tasks it performs are not computationally expensive enough, to benefit significantly from parallelism. If the crawler is set up separately from the application instance, then the collected files have to be later uploaded to the cloud for analysis.

Whether installing on the cloud or a separate machine, certain parameters have to be configured for the crawler to run properly. The crawler root directory contains a directory with the name „sources“. This is by default, where downloaded files get saved. The location can be changed by changing the value of the „`SOURCES_PATH`“ variable in the „config.tcl“ file. To cause the program to download files, a directory has to be created within „sources“ for each of the sources, from which the user wishes to obtain news. A „.info“ file has to be written and saved into each one of the source directories. The „.info“ files contain configuration parameters for the download sources. The parameters necessary are the name of the download source and the URL of the news feed file. The parameters have to be on separate lines with the parameter name and value separated by a space character as shown in this example:

```
name bbc

url http://feeds.bbci.co.uk/news/rss.xml
```

Example source directories, including „.info“ files, for three sources, used for testing the application, have been set up in the „crawler/sources“ directory on the CD accompanying this thesis.

To run the program use the Tcl programming language interpreter „tclsh“ (this has to be installed beforehand), giving the path to „crawler.tcl“ as the parameter. This will cause only a single download for each configured source. To collect news continuously, a scheduling utility, such as Cron, has to be configured to run the program at regular intervals.

For the News program to properly work, „News.jar“ has to be uploaded to the instance along with the „pig“ folder. The jar file and „pig“ directory have to be located in the same directory, because the News application makes the assumption that the scripts, it needs to run, can be found there. If feed files have been collected separately, the „sources“ folder, holding those files, must also be copied to the instance. When running the application the Hadoop core, Pig core and Piggybank library jar files have to be included in the application's class path as they are not compiled with „News.jar“.

4. 4 Saving the Image

When the instance has been modified with custom software and data, it can be saved as a new image on SciCloud. This section describes the process of bundling an instance and uploading the resulting image, based on the resource [19].

To bundle the instance as a new image the user's „.euca“ folder must first be uploaded to the instance's „/mnt“ directory. Then the user must connect to the instance via SSH, navigate to the „.euca“ folder and execute the commands in the file „eucarc“.

```
# cd /mnt/.euca
```

```
# source eucarc
```

The next step is to bundle the instance using the „`euca-bundle-vol`“ command

```
# euca-bundle-vol --cert ${EC2_CERT} –privatekey
${EC2_PRIVATE_KEY} --user 000202048240 –ec2cert
${EUCALYPTUS_CERT} --no-inherit --kernel ${kernelID} –ramdisk
${ramdiskID} -d /mnt/tmp -s 3072
```

The „kernelID“ and „ramdiskID“ parameters can be obtained using the „`euca-describe-instances`“ command.

Next, the bundle has to be uploaded to SciCloud and the bucket name registered:

```
# euca-upload-bundle -b ubuntu.euca-scicloud-pigtest-v1 -m
/mnt/tmp/image.manifest.xml
```

```
# euca-register ubuntu.euca-scicloud-pigtest-
v1/image.manifest.xml
```

In the example „image.manifest.xml“ is the name of the image's manifest file, which is located in the bundle's parent folder („/mnt/tmp“).

An example image has been created on SciCloud with the data analysis application already set up. This image has the manifest „ubuntu.euca-scicloud-pig-v3/image.manifest.xml“ and ID „emi-914B1414“. The relevant data and software is found in the „`root`“ user's home directory „/root“ on the image. This directory contains the Pig and Hadoop installation folders „pig-0.8.1“ and „hadoop-0.20.2“. It also contains the application file „News.jar“ and the scripts, it calls, in the folder „pig“. Finally a set of collected data is also in the „/root“ folder for the purpose of testing the application.

The Crawler program along with a collection of feed files collected between March 17, 2011 and May 11, 2011, can be found at the SciCloud image with the manifest „ubuntu.euca-scicloud-jyrmo-crawler/image.manifest.xml“ and ID „emi-4E751749“. On this image the Cron utility has also been configured to run the crawler once every hour. This means that as soon as an instance of the image is launched, the process of collecting news at regular intervals, continues.

Chapter 5
Application Description

This chapter describes the implementation details of the data collection, processing and analysis tools that were created in the course of this project. Different tools were created for different aspects of the task, the central focus being on utilizing Pig Latin for data processing and analysis. The program design decisions for the different parts of the application are explained in the following sections.

For data collection, a tool, separate from the rest of the system, was created. The processing and analysis phase are handled together in a single Java application, which interacts with the user through the command line. This application was used to dynamically string together independent Pig scripts and individual queries to produce the desired output. In addition a custom tool was implemented and used to create visualizations for analysis results.

5. 1 Data Collection Tool

The data collection tool was implemented in the dynamic programming language Tcl. Its purpose was to gradually build up a collection of RSS feed documents from different online news sources. This collection would later be processed using Pig to extract news items and their publication dates.

Although the collection tool is referred to, in this work, as a web crawler, it is arguable whether it actually fits that description, because it does not, in fact, follow links to web sites, but instead only downloads and saves web feed documents from preset sources. The word „feed aggregator“ or „reader“ may describe the program better.

Before running the crawler a directory for storing the downloaded files has to be created and referred to in the program's configuration file. This directory has to contain a sub-directory for each RSS channel, from which data is to be downloaded. These directories are going to hold all the downloaded files from their respective feeds. All of these directories also need to contain the hidden metadata file „.info“. This file contains information about the feed being downloaded that the collection tool (and later the processing application) needs to know to correctly work. This information includes the name, which is used to refer to this news source, the URL of the feed, a counter value which is incremented each time the feed document is downloaded, and the UNIX timestamp value of the most recent occasion, on which the feed file was downloaded. This metadata is stored line by line in the „.info" file with the parameter and value on the same line separated by a comma. The collection program reads the file, interprets the data as necessary and writes an updated version of the information to the file after each successful download. The following is an example of the content of a „.info" file.

```
name bbc

url http://feeds.bbci.co.uk/news/rss.xml

counter 553

lastupdate 1304938804
```

The counter value is appended to the downloaded file name as a suffix to avoid overwriting existing files.

The way the crawler works is by reading the path to the news sources directory from the configuration and iterating over all of its sub-directories. The „.info" file is read from each directory to determine the URL of the file to download. The file content is downloaded and a new file with the name „`file<counter>.rss`" is created in the directory. Here `<counter>` is the value of the „`counter`" parameter in the „.info" file. This value is then incremented and the value of the „`lastupdate`" parameter is also updated. The „.info" file is then overwritten with the updated data. This process is repeated for all directories in the main sources directory. The source code for the data collection program can be found on the CD included with this work.

Because news need to be gathered continuously, the collection program has to be run at regular intervals. It is also reasonable to automate the execution instead of doing it manually every once in a while. The automation is achieved using the Cron Unix utility. The application was installed on an instance of a SciCloud image and a Crontab entry was created to run the application at one hour intervals. All output (such as error messages) from the script was redirected to a log file, to enable the detection and troubleshooting of problems, if they were to occur.

5. 2 Data Processing

The data analysis and processing functionality is implemented in the form of a Java application. The Java program acts as a wrapper for a number of Pig scripts. It reads user input from the command line and calls different Pig scripts and individual Pig queries accordingly. The data processing instructions are spread across ten files, which the Java program strings together and executes on the Pig server in order to produce the output. How exactly the wrapper program works, is described in section 5.4.

The main data processing tasks are:

1. the extraction of news item entries from web feed files
2. the extraction of the contents and metadata of news items
3. formatting and cleaning up the input data
4. composing a relation mapping single words to the news items they appear in

These are handled in the files: „ReadXml.pig“, „OrderNews.pig“ and „CreateWordMap.pig“. All these files are found on the included CD in the directory „pig“.

The first thing that has to be done, before any analysis operations can be executed, is to read in the raw input data and to format it in a way that Pig can understand. The application is given the file system path to the sources folder containing all the downloaded feed documents. When using the PigStorage loading function (the default loading function in Pig Latin [16]), as described in section 2. 1, data is read from files line by line and the text fields separated by a specified string (a single tabulation character by default) become the values in the created relations columns. This behavior is undesirable when reading files in an XML-based format such as RSS. When reading XML it is instead favorable to treat the contents between a beginning and an end tag as a single entity, even though they may be spread across different lines in the file. Although Pig version 0.8.1, which was used in this project, does not have a built-in XML loading function, there is one in the „Piggybank“ library, a collection of Pig user-defined functions [20]. This library is also included with the application. The XML loader takes as its parameter the tag name of an element and produces a single-column relation, where every row contains an occurrence of this element in the input files [21]. The XML loader is used to extract all „item“ elements from the input files as shown below:

```
items = LOAD '$input_path' USING
org.apache.pig.piggybank.storage.XMLLoader('item') AS
(item:chararray);
```

As the next step, news item information from the „item“ XML fragments has to be extracted and saved in table form as a Pig relation. For this extraction the „REGEX_EXTRACT“ function is used. „REGEX_EXTRACT“ allows the programmer to specify a regular expression to extract a specific string from text [16]. This function is run on each previously obtained „item“ XML fragments to acquire for each of them the text between the beginning and end tags for the following sub-elements of item: „link“, „title“, „description“ and „pubDate“.

```
$source_name = FOREACH items GENERATE
     REGEX_EXTRACT(item, '<link>(.*)</link>', 1) AS
link:chararray,
     REGEX_EXTRACT(item, '<title>(.*)</title>', 1) AS
title:chararray,
     REGEX_EXTRACT(item, '<description>(.*)</description>',
1) AS description:chararray,
     REGEX_EXTRACT(item, '<pubDate>(.*)</pubDate>', 1) AS
pubdate:chararray,
     '$source_name' AS source:chararray;
```

The contents of those elements are saved in corresponding columns of the new relation. This relation has the following schema:

```
items: {link:chararray, title:chararray,
description:chararray, pubdate:chararray, source:chararray}
```

The „link“ value is later used as a unique key for the news stories. The „source“ column holds the name of the news source associated with the item.

Once all the news items are saved in tabular format, the schema needs to be further refined to facilitate the analysis steps. At this point no new data has to be read in, the necessary changes can be inferred from the existing relation. Because news are collected at relatively short intervals, two sequentially downloaded files from the same feed may have overlap among the published items. For this reason a „DISTINCT“ query is run in this step, to eliminate all duplicate items.

Among other things the time zone info is separated from the rest of the date and time info, because Pig has problems processing time zone data. The publication time info is converted to ISO 8601 format. This allows for easier further processing as Piggybank has functions for processing ISO 8601 dates and times, but not for processing RFC 822 format dates, which RSS follows [12, 26]. The news items are also ordered by their publication date. In version 0.8.1 Pig only supports ordering by fields of simple data types, such as „int“ and „chararray“ [16]. ISO 8601 time values are saved with the type „chararray“ and Pig orders all „chararray“ values lexicographically when given the „ORDER BY“ instruction. Because in reality a chronological ordering of tuples is desired instead of a lexicographical one, the ISO 8601 time value is also converted to a Unix timestamp value, which is saved with the „long“ data type. Ordering on this timestamp value amounts to the chronological order of the news item records in the relation. The resulting relation's schema looks like this:

```
ordered_news: {link:chararray, title:chararray,
description:chararray, isotime:chararray, isodate:chararray,
unixtime:long, timezone:chararray, source:chararray}
```

One of the tasks performed in the analysis stage is that of finding words which occur together in a news story. For this purpose a relation is created that maps each word occuring in a news story to that story's link, which is treated as the unique identifier for the news item. This relation also holds the publication date of the item and its publisher's identifier string. The „TOKENIZE“ function is used to extract all words from the title and description of a story. The exact statement used is the following:

```
words = FOREACH texts GENERATE flatten(TOKENIZE(LOWER(text)))
AS word:chararray, link, isodate, source;
```

This is the schema for the word map relation:

```
words: {word:chararray, link:chararray, isodate:chararray,
source:chararray}
```

5. 3 Data Analysis

Once the data has been processed and saved into bags with acceptable schemas, the analysis can begin. The analysis tasks performed are the following:

1. counting the occurrences of all words in the news by date
2. counting the co-occurrences of sets of user-specified words in the news by date

Both these tasks make use of the word map relation described above. The first task is handled in the file „WordCount.pig“ and the second one is split between „FilterWordMap.pig“ and „FindTrend.pig“.

The word count functionality is meant to generate a list of a number of the most commonly occuring words in the news, for each day, over the entire period of time, during which data has been gathered. First some words need to be filtered out of the word map. These are words, that appear commonly in all news sources, but do not indicate the popularity of any particular topic. Examples include words such as „the“, „a, „an“, „to“, „for“ etc. All words in the relation are checked against a regular expression and all tuples matching this regular expression are filtered out. The filtered words are then grouped by date and word. The occurrences of each individual word within each of these groups, are counted.

```
grouped = GROUP filtered_words BY (isodate, word);

counts = FOREACH grouped GENERATE group,
COUNT(filtered_words) AS count:long;
```

Also in each group the words are ordered starting from the most frequent one and limited to only include a number (specified in an argument to the script, the wrapper application sets this as 20) of words, which appear in the top of the list.

```
day_counts = GROUP counts BY group.isodate;

day_counts_limited = FOREACH day_counts {

     ordered_counts = ORDER counts BY count DESC;

     limited_counts = LIMIT ordered_counts $limit;
```

```
    GENERATE group, limited_counts;
};
```

These lists are written to the output file.

The main goal of the program is to allow the user to track the co-occurrences of a set of terms of their choosing. This is achieved by selecting from the word map, those records, which contain the words. This goal could also be accomplished by matching news stories against a user specified regular expression. The use of regular expressions would also save some computational overhead as the word map relation would not have to be created for this news aggregation task. However the use of a mapping relation comes with certain advantages. Namely, once the word map has been created, it can be reused and word lookup from there happens faster, than the matching of regular expressions against all news stories, because Pig enables optimization by indexing the word map. Regardless, as part of the application a regular expression based news search module was also implemented. This is described in detail in the next section.

To find co-occurrences of words, a new bag is first created for each of these words. This bag consists of all the tuples in the word map relation which contain the word.

```
$word = FILTER words BY word == '$word';
```

An inner join is then performed on the created bags using the „link“ field, creating a bag, which contains only the records for the news items, in which all of the given search terms occur. Two different output bags are generated: one, which holds the counts of the co-occurrences of the words grouped by date, and another, which groups the counts by date as well as news source.

5. 4 Text Search

To allow the user to search the database of news gathered for news containing some specific string, a text search module was implemented using Pig. The file „SearchRegex.pig“ contains the code for this functionality.

The search is performed on the „`ordered_news`“ relation, which was described above. The script takes as a parameter the regular expression string to match. It then performs a

„FILTER“ operation on the „ordered_news“ bag, generating only the tuples, in which the „title“ or „description“ field match the expression specified earlier.

```
results = FILTER ordered_news BY title MATCHES '$regex' OR
description MATCHES '$regex';
```

These matches are written into an output location in the file system.

While the data analysis functionality gives insight into general trends in the news, the search functionality offers more concrete results in the form of individual news item summaries. The search can be used in congress with analysis by first identifying a relevant topic and then looking at what exactly has been written by different news sources, on this subject.

5. 5 The Wrapper Program

Pig Latin does not have any control flow statements nor modularity [23]. For this reason, a Java program was written as a wrapper to the data processing, analysis and search functionality. This application ties together all the functionality described above, by calling Pig scripts and running individual queries according to user input. The user interaction is implemented in the form of a command line user interface.

When the program is run, the user is first prompted for what type of source the news should be read from. The options are to read from collected XML files or from a file containg a previously saved ordered news Pig relation.

In both cases the user is asked for the path to the input directory. In the case of loading XML files this is the path to the directory containing the different news source download directories. Those direcotries have to include the „.info“ metadata files, described above, as the program reads the unique name, assigned to the news source, from it. In the case of existing news table loading, the specified input path is assumed to contain a news table in the form of a file, where tuples are separated by newline characters and fields by tabulation characters. The directory location needs to be given without the final slash („/“) character.

The way XML files are read and transformed into a relation, was described above in the Data Processing section. How and when a news relation is written into the file system for later reuse, will be covered further on down in this section. Immediately following, though, is the description of the process of reading in a news table from a previously saved text file.

The table reading process is handled in the script file „ReadNewsTable.pig“, which really only contains a single statement. This is the „`LOAD`“ statement, which reads in the specified input file. The assumption is made that the input file holds the records in a tab-separated format.

The next piece of input, asked from the user, is which task they wish to perform. The options here are: generating lists of the most popular words in the news by date, finding trends for co-occuring word sets and performing a text search.

If either of the first two options is selected, the user will be presented with an additional question about how to retrieve the word map. Because both the word occurrence counting and topic trend finding scripts operate on the word map relation, the user is expected to decide, whether to create the relation from the news table, which has been read in, or to read in the word map from a previously saved file. The program then calls the appropriate script for the task.

To create a word map the program calls the „CreateWordMap.pig“ script described above. The reading of a previously saved word map happens similarly to the news table reading and is handled in the script file „ReadWordMap.pig“. The way a word map is saved for future use, is described below in this section.

With the word count task the user is asked for the output directory path, which is the final piece of input, required from the user, in this use case. The „WordCount.pig“ script is called and the output is saved in the user specified output path in a directory named „counts.out“.

If the topic trend identification task was chosen, then, besides inquiring about the output path, as described above, the user is also prompted to enter at least two search words separated by spaces. These words represent the topic, whose popularity the user wishes to track. The program executes the „FilterWordMap.pig“ script once for each of these words, using the word as the parameter and then runs a query joining the results of the filter operations on their „link“ field. Finally „FindTrend.pig“ is run generating and outputting the results into files. In the user specified output path, a new directory is created. The name of this directory is composed of the search terms separated by underscore characters. Within this new directory the overall topic trend results are saved in the directory „trends.out“ and the results, broken down by news source, in „source_trends.out“.

The third possible task to perform is running a text search.For this the user is asked for the output path and the search query string. These parameters are passed on to the „SearchRegex.pig“ script, which implements the search functionality. The output from the search is saved in the output path, in a new directory named „search.out“.

Aside from console user input accepted at run time, the program also takes a maximum of two arguments from the command line. The first of these is the execution mode – either „local“ or „mapreduce“. If not specified otherwise, this defaults to „local“. The second argument is the path to an output directory.

If the second parameter is specified, then the program writes out not only the results, the user is requesting, but also some intermediate processing results, which can be reused later. These intermediate results are the news table and the word map relations. The program checks if the output path has been specified and if so, runs an extra store operation after creating the news table. The table is saved in a the output path, in a directory named „news.out“. Similarly, if running either the word count or topic trend task, the „`words`“ relation is saved into a directory named „words.out“.

Chapter 6
Usage Example

This chapter walks through all the stages of the data analysis process, from the user's perspective, using the tools created over the course of this project. Data collection is demonstrated along with the process of identifying trends in the news. Visualizations are offered, illustrating the trends, to give a better understanding of the output data. The text search functionality is also demonstrated.

6. 1 Data Collection

The „Crawler“ application is used to perform raw data collection. It was set up to collect news from three online news sources. The sources and their corresponding feed URL-s are:

1. BBC News - http://feeds.bbci.co.uk/news/rss.xml
2. CNN.com - http://rss.cnn.com/rss/edition.rss
3. The New York Times - http://feeds.nytimes.com/nyt/rss/HomePage

For each of these a directory was created in the crawler's sources directory and configuration parameters set in the „.info“ files as described above. The program was deployed on an instance of a SciCloud image with the Cron utility set to execute it at one hour intervals. Over the span of time between March 17, 2011 and May 11, 2011, during which the instance was online, a collection of RSS files was accumulated. These files provided the raw input for the analysis tasks.

6. 2 Identifying Frequent Words

The descriptions given in this chapter are based on the process of running the news analysis program on an instance on SciCloud. The assumption is made that the instance has been prepared for work as described in the Application Deployment chapter. An image configured in the appropriate way, has been created and registered on SciCloud, it has the ID „emi-914B1414“. All commands are run on the command while in the „`root`“ user's home directory „/root“.

To identify frequent words by date, the News program is run for the first time. To avoid later rework of reading in data from XML files, it is desirable to save some intermediate data, aside from the final result. For this purpose the output path for the intermediate data is given as the second parameter to the program on the command line. When running on a cluster of more than one node, the first parameter should be set to „mapreduce“, because by default the application is run in local mode, without taking advantage of the parallelism available. The „`java`“ command is used to run the application and all relevant libraries included in the classpath, because no libraries are bundled with „News.jar“.

```
# java -cp hadoop-0.20.2/hadoop-0.20.2-core.jar:pig-
0.8.1/pig-0.8.1-core.jar:pig-
0.8.1/contrib/piggybank/java/piggybank.jar:News.jar
ee.ut.ds.pig.News mapreduce /root
```

When prompted for the approach of reading in data, the choice is made to read the data from XML files. This is the only possible option when running the program for the first time, as no Pig table has yet been serialized in the file system, to read from. To the next question, about where to find the input files, the user gives the path to the sources folder. In the current example this is „/root/sources“.

Now the user picks the task to perform – in this case that task is finding popular words by date. Additionally the program prompts the user for the output directory path. This is also specified as „/root“, so the output directory is created in the working directory. Hadoop MapReduce jobs are then run, outputting the news table into „/root/news.out“, the word map relation into „/root/words.out“ and the final output into „/root/counts.out“.

The size of all the files gathered was about 100 MB. The size on disk of the generated word map files is about 30 MB.

From the final output the most popular words by day can be read. The following, for instance, are seven of the most frequent words in the news on April 29, 2011 (formatted for clarity)

```
wedding, 41

royal, 37

prince, 30
```

```
william, 26
kate, 24
people, 23
middleton, 19
```

The number after each word represents the number of times the word occurred in the news that day. April 29, 2011 was the day when the wedding of Prince William, Duke of Cambridge and Catherine Middleton took place [24], therefore words such as the ones presented above appeared disproportionally frequently within news stories on that day. The words are also more frequent on the days, leading up to, and the days immediately after the wedding.

To contrast the above example here is another outtake from the same output file, this time giving the seven top words for the date May 2, 2011

```
bin, 64
osama, 54
laden, 49
death, 32
president, 26
killed, 24
obama, 21
```

This shows, that three days after the royal wedding, a new topic has overtaken the wedding in popularity. May 2, 2011 was the day, when United States President Barack Obama announced the death of al-Qaeda leader Osama bin Laden [25]. The news was widely covered by different news sources, including the ones, from which data was gathered for this work. As a result, the words above appeared more than others on May 2 and the days immediately following the announcement.

6. 3 Identifying Trends

The experiment, described in the previous section, of identifying, which words are more frequently featured, gave some insight into how different topics gain and lose popularity over time. To describe particular trends with more precision, the News application is used to run the „topic trend“ task. The output from the previous task can be used to make educated guesses, as to what words to use to define a topic. The process of identifying and plotting the popularity of topics in news stories is the subject, with which this section is concerned. The „royal wedding“ and „death of Osama bin Laden“ cases from above are used as example data for testing.

This time, when running the News program, the second command line parameter, identifying the path for intermediate output, is omitted, because in the previous example the intermediate data was already saved. Instead the saved output is now reused to save time, which would otherwise be spent on parsing data from XML files. Therefore, when asked by the program, the choice is made, to read in news data from an existing table in a file, as well as to read in the words table. When prompted for the news input path, the path „/root/news.out“, identifying a folder created in the previous execution of the application, is entered. Similarly the path „/root/words.out“ is given as the input data location for the word map relation.

After specifying the above data and choosing to perform the topic trend finding task, the user is finally expected to enter at least two search words separated by spaces. In the first example run, those words are „royal“ and „wedding“. A new directory is created to the specified output path. The directory's name consists of all the search terms separated by underscores, in this case „royal_wedding“. In that folder two output directories, „trends.out“ and „source_trends.out“, are created. The first of these holds the numbers of co-occurrences of the words in news overall, the second has those results broken down by news source. Here are some sample lines from „trends.out“

```
2011-04-21T00:00:00.000Z 5

2011-04-22T00:00:00.000Z 6

2011-04-23T00:00:00.000Z 7
```

The following lines are from „source_trends.out“

```
(bbc,2011-05-04T00:00:00.000Z)      1

(bbc,2011-05-10T00:00:00.000Z)      2
```

```
(cnn,2011-04-24T00:00:00.000Z)      1
```

The task of extracting the co-occurrence frequencies by date was also run using the search words „osama“, „bin“ and „laden“. To illustrate the results for both cases, graphs were created based on the output data.

Figure *2* graphs the overall popularity of the „royal wedding“ topic (data derived from „/root/royal_wedding/trends.out“). It indicates that the news were consistently interested in the wedding before the event. The days leading up to the wedding saw a rise in the number of stories mentioning the two words and the popularity peaked on 29 April, the wedding day. Interest in the topic faded soon afterwards.

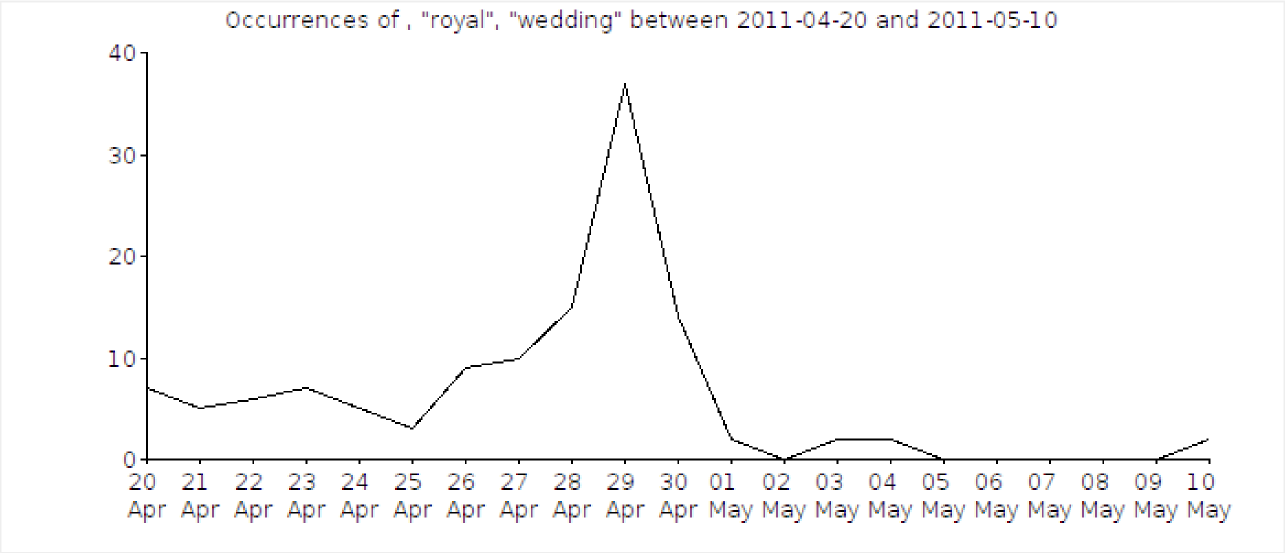

Figure 2. *Overall occurrences of the words „royal“ and „wedding“ in the news*

Figure *3* depicts the differences in the amount of coverage given to the wedding between different news sources („/root/royal_wedding/source_trends.out“). The news source generally giving the most coverage to the topic throughout the time period observed, is BBC News. This could be due to the fact that BBC is a British news source and for them the wedding is local news. In contrast both CNN and New York Times are American publications, meaning the event does not bear quite as much significance for them.

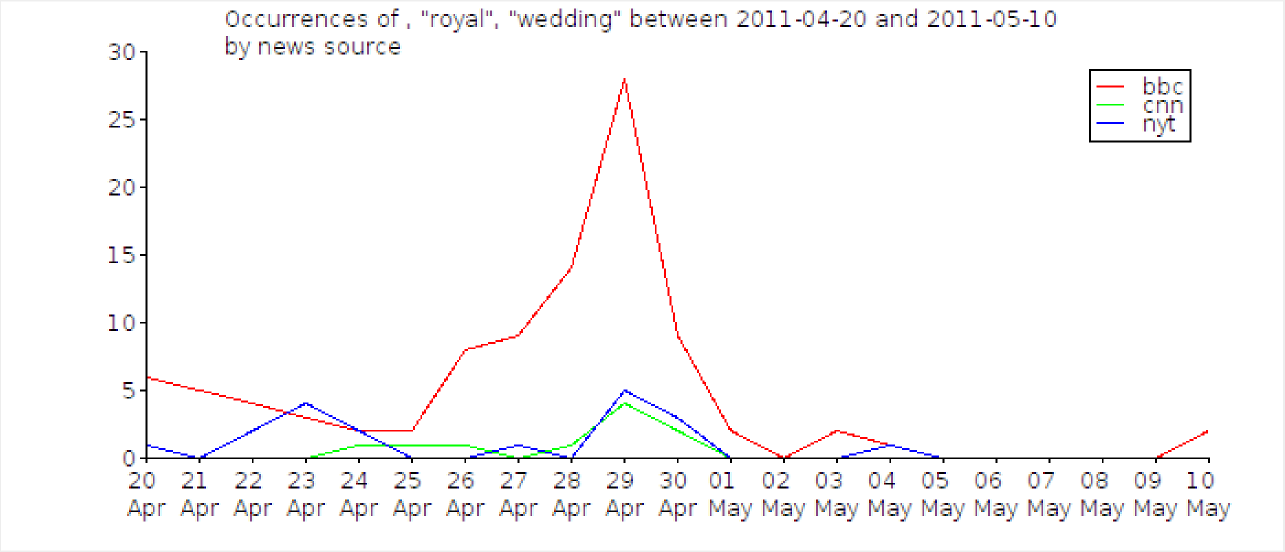

Figure 3. *Co-occurrences of the words „royal" and „wedding" in different news sources*

The co-occurrence frequencies of the words „osama", „bin" and „laden" are plotted in figure *4*. The plot was drawn across the same time period from 20 April to 10 May, as with the wedding example given earlier, to highlight some differences in the two cases. Unlike the event of the wedding, which was announced long ahead of time, with the killing no indication was given beforehand. This can be seen from the graph, as in the days leading up to the event no mention of the topic was featured in the news. As the official announcement was made on May 2, the number of occurrences suddenly went from zero to nearly 60. Referring back to figure *2*, it is worth noting that the May 2 spike in news about Osama bin Laden, coincides with a drop to zero in the number of news featuring the words „royal" and „wedding".

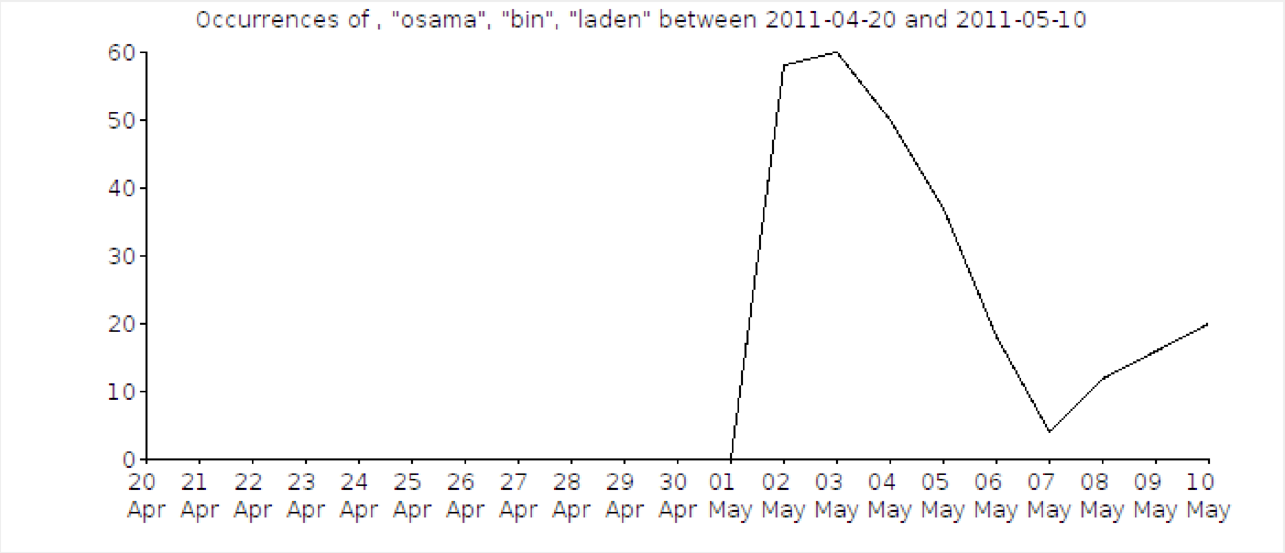

Figure 4. *Co-occurrences of the words „osama“, „bin“ and „laden“ in the news*

Figure 5 has the results for the Osama bin Laden topic grouped by news source. It shows that all three sources gave the topic a similar amount of coverage with BBC's coverage peaking at 25 occurrences on May 2 followed by CNN and New York Times on the next day.

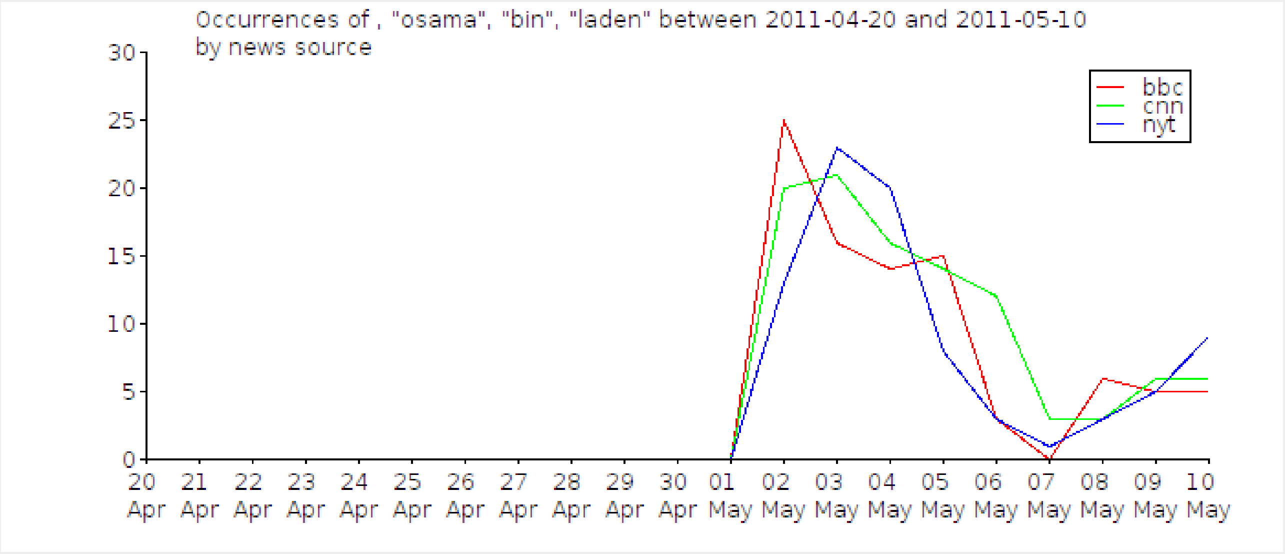

Figure 5. *Co-occurrences of the words „osama“, „bin“ and „laden“ in different news sources*

6. 4 Searching for News

This section goes through the process of finding news stories, which match a certain search query string. The News application is run once more, this time to demonstrate the regular expression search functionality.

As was the case with the trend identifying functionality, the existing news table is again used in the regular expression search. In this case, however, the word map is not used and the user is not asked whether to create it or read it from a file. The user is required to specify an output path and a search string.

It has already been established that the royal wedding was a popular topic over a period of time in April and May 2011. The search task intends to find out what exactly is being written on the subject. In particular, the user may be interested in, how the wedding was celebrated and enters the regular expression „celebrate.*William.*Kate“. As a result all tuples from the news relation, having a title or description, which matches this expression, are written to the output location. One such news story has the following title: „Guests arriving for royal wedding“. The story's description reads: „Thousands of people throng central London to celebrate the wedding of Prince William and Kate Middleton at Westminster Abbey.“ The link indicates the web page address, where the full news item can be read: „http://www.bbc.co.uk/go/rss/int/news/-/news/uk-13229961“

Chapter 7
Problems Encountered

This chapter describes the various problems encountered during the development of the data analysis system. The solutions and workarounds, which were used to overcome the problems, are also discussed.

7. 1 Data Collection Frequency

For collecting data, it was important to choose the optimum size for the period of time, after which downloads would be repeated. The size had to be picked so that the amount of data overlap in any two consecutively downloaded files would be minimal, but that no news stories would actually be omitted.

In reality minimizing the overlap is not a simple goal to achieve. Different sites update their feeds with different regularity and would therefore have to be handled case-by-case. The RSS standard defines the optional element „ttl“ or „time to live“, which lets the aggregator know, how long (in minutes), before the feed should be requested again [14]. However, even accounting for the time to live, the problem of data duplication remains, because feeds are usually updated in small increments and not by replacing every single item in the feed's „channel“ element. Also, only two of the three news feeds aggregated, featured the „ttl“ element.

For the purpose of simplicity, a different approach was taken to the collection process. The RSS files were downloaded at one hour intervals. According to [14] „By convention, most aggregators check an RSS feed for updates once an hour“. At this stage no duplicates were removed. This was instead done when the data was first read in for analysis using Pig. To avoid having to perform this time consuming task every time the analysis program is run, the option is given to save the relation without duplicates into the file system, so it could later be read in again.

The ideal solution to the problem would involve downloading files at short intervals to avoid data loss (using the configuration, which the application was tested with, a one hour period was enough to achieve this). The second part of the solution would be to build a database of news incrementally, adding new items every time feeds are downloaded. In this case the duplicate entries could also be removed after each download. Creating an application with this kind of architecture, was however, beyond the scope of this work.

7. 2 XML Parsing

One of the problems faced, when building the system stemmed from the nature of the input data. As stated above, RSS is an XML based format [12]. Pig is well-suited to read input given as lines of text with fields separated by a given string. This is easily done using the „`PigStorage`“ loading function [16]. There are also no problems extracting data from lines of input with a pre-determined format using regular expressions. The „`MyRegExLoader`“ function available in the „Piggybank“ library is useful for that purpose [26]. Data in XML format, however, fits neither of those cases.

The „Piggybank“ library also contains an „`XMLLoader`“ load function meant for reading data from files in XML format. This function takes as its parameter a single string, representing the name of the element to extract. It loads all occurrences of this element into a variable as fields in a single column Pig relation. [27] The content of the field is the entire unaltered string representing the element, including the beginning and end tags, as well as any nested XML markup within the content of the tag. The content of the element does not get automatically converted into nested tuples within the element tuple.

The elements, nested in the string, which was loaded in from the input file, have to be extracted into tuples in a separate statement. Pig has no builtin functions for extracting content from snippets of XML code. Neither is there any such function in the „Piggybank“ library. This is why, when implementing the application, the decision was made to use regular expressions to parse the necessary data from the XML snippets. The „`REGEX_EXTRACT`“ [16] function is employed for this purpose. Using regular expressions to parse XML is an unreasonably low-level approach, but it has to be taken, because Pig does not have any built-in method for XML parsing.

7. 3 Time Zone Information

Some problems were faced when trying to extract date and time data from RSS files. While the RSS standard specifies to a degree, how data in a feed document has to be represented, it still leaves room for some ambiguities in the format. This means that any feed aggregating program has to account for the different ways, in which publishers interpret the standard. To alleviate the effects of this problem, news sources were chosen, which follow the standard in a similar way, so that a minimal amount of modifications would have to be made, to adapt the system from one source to another. This still left a problem with interpreting time zone data.

Pig's built-in data types do not include a date or time type [16]. The dates in RSS have to conform to the RFC 822 standard [12]. RFC 822 allows for time zones to be expressed as three-letter acronyms as well as offsets, in hours, from the Universal Time (or Greenwich Mean Time) [28]. The three news sources monitored, used the first option. This caused problems when parsing the date and time into Pig. The user defined function „`CustomFormatToISO`“, available in the „Piggybank“ library, is used to convert „pubDate“ values into ISO 8601 format. It works by using a template format (e.g „`EEE, dd MMM yyyy HH:mm:ss`“), specified by the programmer, to which the „pubDate“ value is fitted and transformed to ISO 8601. Internally „`CustomFormatToISO`“ uses the the „Joda-Time“ Java library [22]. The „Joda-Time“ conversion class „DateTimeFormat“ does not support the parsing of three-letter time zone names [29]. Even if it did, however, the three letter designations are not a unique way of identifying a time zone, as pointed out in [30]: „the same abbreviation is often used for multiple time zones (for example, "CST" could be U.S. "Central Standard Time" and "China Standard Time")“.

For the reasons stated above the application ignores time zone information in publication dates and only saves the date and time, as if the time zone was always Coordinated Universal Time (UTC) [31]. This causes small inaccuracies in output data. In the interests of future development, the time zone information is still saved in the news table, even though it is not actually used. Future iterations of the application may offer a solution to this problem.

Chapter 8
Related Work

This work deals with data analysis using the Apache Pig tool. Pig is not the only tool built on top of Hadoop, which performs the kinds of tasks described here. This chapter briefly discusses other MapReduce based data aggregation frameworks, pointing out their similarities and differences.

Hive is a data warehousing infrastructure, which runs on top of Hadoop. It provides a language called Hive QL to organize, aggregate and run queries on the data. Hive QL is similar to SQL, using a declarative programming model [32]. This differentiates the language from Pig Latin, which uses a more procedural approach [33]. In Hive QL as in SQL the desired final results are described in one big query. In contrast, using Pig Latin, the query is built up step by step as a sequence of assignment operations.

While Pig and Hive are meant to perform similar tasks, the argument is made in the resource [34], that Pig is better suited for the data preparation phase of data processing, while Hive fits the data warehousing and presentation scenario better. The idea is that as data is incrementally collected, it is first cleaned up using the tools provided by Pig and then stored. From that point on Hive is used to run ad-hoc queries analyzing the data. In this work the incremental buildup of a data warehouse is not enabled and both data preparation and querying are performed using Pig. The feasibility of using Pig and Hive in conjunction remains to be tested.

Zebra is a sub-project of Pig which provides a layer of abstraction between Pig Latin and the Hadoop Distributed File System [35]. Zebra allows a Pig programmer to save relations in a table-oriented fashion (as opposed to flat text files, which are, normally used) along with meta-data describing the schema of each relation [36]. As such the News application would benefit from its use, because the program would not have to specify the schema of the relation each time peviously saved data is read in from a text file. Currently the program does not utilize Zebra for storage nor retrieval.

A unit testing framework, called PigUnit, has been developed for Pig. It allows a programmer to write Java code, which runs tests on Pig Latin scripts. The tests can be run using JUnit or a similar Java testing framework. [37] The Pig code created in this work was tested without the help of PigUnit.

Apache HBase is a data base engine built using Hadoop and modeled after Google's BigTable. It is optimized for real time data access from tables of millions of columns and billions of rows. Among other features, HBase offers support for interfacing with Pig and Hive. [38] The Pig API features a storage function for loading data from an HBase data base [39], but in this work the data was read from and written to flat HDFS files, because the data amounts were too small to necessitate the use of HBase.

A MapReduce based data collection and monitoring system called Chukwa has been developed on top of Hadoop. Chukwa is mainly aimed at processing log files, especially from Hadoop and other distributed systems [40]. Because Chukwa is meant mostly for the narrow area of log data processing, not general data analysis, the tools it offers are not as diverse as Pig's and not as well suited for the tasks performed in this work.

Chapter 9
Conclusion

This work introduced the data processing and analysis tool Pig. It explained the principles involved in programming in the Pig Latin query language and how it is related to the MapReduce programming model.

To demonstrate the usefulness of Pig, a data analysis problem was specified and solved. The problem involved identifying trends in online news to find how the popularity of different topics changes over time and how the coverage of topics differs from one news source to another. The news were gathered from RSS news feeds using an automated tool.

To solve the problem, a data analysis application was designed, implemented and tested. The implementation, deployment and usage details of this application were presented. Some of the results from running the application were also presented and illustrated with graphs.

The applicability of Pig for large scale data analysis was demonstrated. Pig provides a useful abstraction on top of MapReduce. It allows to write data manipulation statements and querys in a high-level language and thanks to its underlying MapReduce model it is able to automatically parallelize and scale the operations performed providing support for work with very large data sets. This makes the framework applicable to the problems SciCloud deals with. Pig also has some disadvantages, e.g its current support for XML-based formats is incomplete and it is difficult to work with date and time formats.

In its current state the data extraction functionality was tested on a sample set of news feeds, but in the future, extraction queries could be refined to facilitate data collection from more sources. The application could also be expanded by implementing different text analysis algorithms. The possiblities of integrating Pig with other Hadoop based frameworks are also worth studying.

Raamistiku Apache Pig Kasutamine Suuremahulises Andmeanalüüsis

Magistritöö

Jürmo Mehine

Resümee

Käesolev magistritöö kirjeldab andmete paralleeltöötluseks mõeldud tarkvararaamistiku Apache Pig kasutamist. Esitatud on konkreetne andmeanalüüsi ülesanne, mille lahendamiseks raamistikku kasutati. Selle töö eesmärk on näidata Pig-i kasulikkust suuremahuliseks andmeanalüüsiks.

Raamistik Pig on loodud töötama koos paralleelarvutuste tegemise infrastruktuuriga Hadoop. Hadoop realiseerib MapReduce programmeerimismudelit. Pig käitub lisa-abstraktsioonitasemena MapReduce-i kohal, esitades andmeid relatsiooniliste tabelitena ning lubades programmeerijatel teha päringuid, kasutades Pig Latin päringukeelt.

Pig-i testimiseks püstitati andmeanalüüsi ülesanne, mis oli vaja lahendada. Üheks osaks ülesandest oli RSS veebivoogudest kogutud uudistest päevade kaupa levinumate sõnade tuvastamine. Teine osa oli, suvalise sõnade hulga puhul, kogutud uudistest leidmine, kuidas muutus päevade kaupa selle sõnade hulga koosesinemiste arv uudistes. Lisaks tuli Pig-i kasutades realiseerida regulaaravaldisi rakendav teksti otsing kogutud uudiste seast.

Probleemi lahendusena realiseeriti hulk Pig Latin keelseid skripte, mis töötlevad ja analüüsivad kogutud andmeid. Funktsionaalsuse kokku sidumiseks loodi programmeerimiskeeles Java raamprogramm, mis käivitab erinevaid Pig skripte vastavalt kasutaja sisendile. Andmete kogumiseks loodi eraldi rakendus, mida kasutati regulaarsete intervallide järel uudisvoogude failide alla laadimiseks.

Loodud rakendust kasutati kogutud andmete analüüsiks ja töös on esitatud ka mõned analüüsi tulemused. Tulemustest võib näha, kuidas teatud sõnade ja sõnakombinatsioonide esinemissagedused muutuvad seoses sellega, kuidas sündmuste, mida need sõnad kirjeldavad, aktuaalsus suureneb ja väheneb.

References

[1] S. N. Srirama, O. Batrashev, E. Vainikko, SciCloud: Scientific Computing on the Cloud, in: The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGrid 2010), p. 579.

[2] Pig Wiki, „PoweredBy“, http://wiki.apache.org/pig/PoweredBy, last visited 20.05.2011

[3] SciCloud Wiki, SciCloud, http://dougdevel.org/scicloud/trac/wiki/SciCloud, last visited 20.05.2011

[4] Wikipedia, Eucalyptus, http://en.wikipedia.org/wiki/Eucalyptus_%28computing%29, last visited 20.05.2011

[5] Eucalyptus Systems, Inc., Eucalyptus Open-Source Cloud Computing Infrastructure – An Overview, August 2009, http://opendata.socrata.com/download/dz9u-pfa3/application%252Fpdf, last visited 20.05.2011

[6] Wikipedia, Hadoop, http://en.wikipedia.org/wiki/Hadoop, last visited 20.05.2011

[7] J. Dean , S. Ghemawat, MapReduce: Simplified data processing on large clusters, in Proc. of the 6th OSDI, Dec. 2004

[8] R. B. Donkin, Apache Software Foundation, Hadoop And Friends, http://people.apache.org/~rdonkin/hadoop-talk/hadoop.html, last visited 20.05.2011

[9] Hadoop, Welcome to Apache Hadoop, http://hadoop.apache.org/, last visited 20.05.2011

[10] D. Borthakur, Hadoop, HDFS Architecture Guide, http://hadoop.apache.org/hdfs/docs/r0.21.0/hdfs_design.html, last visited 20.05.2011

[11] Wikipedia, Pig (programming language), http://en.wikipedia.org/wiki/Pig_%28programming_language%29, last visited 20.05.2011

[12] RSS Advisory Board, RSS 2.0 Specification, http://www.rssboard.org/rss-specification, last visited 20.05.2011

[13] Wikipedia, Web syndication, http://en.wikipedia.org/wiki/Web_Syndication, last visited 20.05.2011

[14] RSS Advisory Board, Really Simple Syndication Best Practices Profile, http://www.rssboard.org/rss-profile, last visited 20.05.2011

[15] Pig 0.8.1 Documentation, Pig Latin Reference Manual 1, http://pig.apache.org/docs/r0.8.1/piglatin_ref1.html, last visited 20.05.2011

[16] Pig 0.8.1 Documentation, Pig Latin Reference Manual 2, http://pig.apache.org/docs/r0.8.1/piglatin_ref2.html, last visited 20.05.2011

[17] Pig 0.8.1 Documentation, PigSetup, http://pig.apache.org/docs/r0.8.1/setup.html, last visited 20.05.2011

[18] Pig 0.8.1 API, Class PigServer, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/PigServer.html, last visited 20.05.2011

[19] SciCloud Wiki, SciCloud Manual, http://dougdevel.org/scicloud/trac/wiki, last visited 20.05.2011

[20] Pig Wiki, Piggy Bank – User Defined Pig Functions, http://wiki.apache.org/pig/PiggyBank, last visited 20.05.2011

[21] Pig 0.8.1 API, Class XMLLoader, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/piggybank/storage/XMLLoader.html, last visited 20.05.2011

[22] Pig 0.8.1 API, Class CustomFormatToISO, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/piggybank/evaluation/datetime/convert/CustomFormatToISO.html, last visited 20.05.2011

[23] Pig Wiki, Turing Complete Pig, http://wiki.apache.org/pig/TuringCompletePig, last visited 21.05.2011

[24] Wikipedia, Wedding of Prince William and Catherine Middleton, http://en.wikipedia.org/wiki/Wedding_of_Prince_William_and_Catherine_Middleton, last visited 21.05.2011

[25] Wikipedia, Death of Osama bin Laden, http://en.wikipedia.org/wiki/Death_of_Osama_bin_Laden, last visited 21.05.2011

[26] Pig 0.8.1 API, Class MyRegexLoader, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/piggybank/storage/MyRegExLoader.html, last visited 21.05.2011

[27] Pig 0.8.1 API, Class XMLLoader, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/piggybank/storage/XMLLoader.html, last visited 21.05.2011

[28] RFC 822, Standard for ARPA Internet Text Messages http://asg.web.cmu.edu/rfc/rfc822.html, last visited 21.05.2011

[29] Joda time 1.6.2 API, Class DateTimeFormat, http://joda-time.sourceforge.net/api-release/org/joda/time/format/DateTimeFormat.html, last visited 21.05.2011

[30] Java Platform, Standard Edition 6, API Specification, Class TimeZone http://download.oracle.com/javase/6/docs/api/java/util/TimeZone.html, last visited 21.05.2011

[31] Wikipedia, Coordinated Universal Time, http://en.wikipedia.org/wiki/Coordinated_Universal_Time, last visited 21.05.2011

[32] Hive, front page, http://hive.apache.org/, last visited 21.05.2011

[33] C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins, Pig Latin: A Not-So-Foreign Language for Data Processing, in SIGMOD'08, June 9-12, 2008

[34] Alan Gates, Yahoo! Hadoop Blog, Pig and Hive at Yahoo!, http://developer.yahoo.com/blogs/hadoop/posts/2010/08/pig_and_hive_at_yahoo/, last visited 21.05.2011

[35] Pig 0.8.1 Documentation, Zebra Overview, http://pig.apache.org/docs/r0.8.1/zebra_overview.html, last visited 21.05.2011

[36] Pig Wiki, Apache Zebra Wiki, http://wiki.apache.org/pig/zebra, last visited 21.05.2011

[37] Pig 0.8.1 Documentation, PigUnit – Pig script testing simplified, http://pig.apache.org/docs/r0.8.1/pigunit.html, last visited 21.05.2011

[38] HBase, This is Apache HBase, http://hbase.apache.org/, last visited 22.05.2011

[39] Pig 0.8.1 API, Class HbaseStorage, http://pig.apache.org/docs/r0.8.1/api/org/apache/pig/backend/hadoop/hbase/HBaseStorage.html, last visited 22.05.2011

[40] Chukwa 0.4 Documentation, Overview, http://incubator.apache.org/chukwa/docs/r0.4.0/index.html, last visited 22.05.2011

Appendices

Appendices are included on the accompanying CD. The CD contains:

- the Crawler application, including source code and sample configuration (in the folder „crawler“)
- the News application compiled as a jar file („News.jar“)
- Java source code for the News application (in the folder „src“)
- individual Pig scripts used by the News application (in the folder „pig“)