"Parallel Protein Classification with IBM BigInsights"

Semester Thesis

Department of Computer Science
University of Applied Science Rapperswil

Spring Term 2013

Authors:	Christof Büchi, Susanne Mathys
Advisor:	Prof. Josef M. Joller
Project Partner:	IBM Switzerland
External Co-Examiner:	Romeo Kienzler

Abstract

Big Data is an expanding topic in information technology based on the huge collection of data which is available today on IT systems all over the world. Processing huge amounts of large files and analyzing unstructured data in real time could bring advantages for institutions or enterprise which store a large volume of generated data from their transactions.
Dealing with the rapid growth of data and analyzing it is crossing the boundaries of the given IT infrastructures. *Google* and *Yahoo!* have introduced their own way how to handle such datasets. A completely new architecture beyond well-known established tools and principles is required to store massive data efficiently in storage and process them with minimal overhead.
Big Data systems and frameworks such as *IBM BigInsights* with *Hadoop* provide a distributed fault-tolerant file system running on commodity hardware. They also allow writing custom applications in *Java* based on the *MapReduce* principle.
How difficult would it be to perform classification with a given single processing application on a *Big Data* system? During our research we wanted to show that it is as simple as setting up a cluster and running the tool out of a bash script that is used within a *Hadoop* streaming job. We took a look at the overhead of using such a complex framework for processing simple applications in a parallel manner. We also had a scope to the scale out characteristics of the cluster size.

Declaration

We, Christof Büchi and Susanne Mathys declare:

- This term project and the work presented in it is our own, original work.
- All the sources we consulted and cited are clearly attributed. We have acknowledged all main sources of help.

Rapperswil, May 31, 2013

Christof Büchi

Susanne Mathys

Contents

1 Management Summary

1.1 Problem Definition

In this project we will solve a large scale text classification problem using the massive parallel data processing infrastructure based on *IBM InfoSphere BigInsights*. Focus will not be on the classification algorithm itself but on ease of use in implementing such a system based on already existing classifiers and data sources to be integrated. Therefore we will focus on scalability and reuse.

1.2 Solution Proposal

At the beginning the data has to be retrieved and imported into the system. Depending on the data source, there are many options (e.g. *BigInsights WebCrawler*, Open Source Tools, HandWritten Crawler, ..) and therefore an appropriate solution has to be chosen and implemented. Depending on the type of data to be analyzed, the appropriate storage system has to be chosen (e.g. files in *HDFS*, *HBASE*, *HIVE*, ..). Based on *Hadoop-Streaming*, a library capable of integrating any *UNIX* command-line application able to read and write to Standard-Input/Standard-Output the integration of existing classification algorithms/software packages should be shown.

1.3 Sample Use Case

As data source, a publicly available protein database will be used. In order to copy a large test data set of protein sequences including their annotations into *HDFS*. Based on this protein sequence data a classifier will be used to classify these proteins into the following subclasses based on their amino acid sequence: alpha, beta, alpha + beta, alpha/beta, and zeta (irregular). Since the implementation of the classifier itself is not in the scope of this work, the classification performance will not be considered.

1.4 Experiments

The experimental evaluation will be performed in two dimensions: data set size and cluster size. Based on these recordings, the scale out coefficient of the system will be determined. It will be checked whether it holds the theoretical assumption of linear scale out, and if not, a bottleneck analysis will be performed.

1.5 Future Work

A further extension to the sample use-case could be enriching the protein information with the publication references. Based on the referenced abstracts and the already classified proteins, a text analytics plugin (SystemT) provided by *IBM BigInsights* could be used to classify these abstracts as well. The correlation between the protein classifier and the text classifier can also be drawn.

2 Introduction

2.1 Objectives

During our research we covered the following goals:

- Demonstrating the simple use of *IBM Biginsights*
- Running an existing executable on the *IBM BigInsights* cluster
- Achieving linear scale out behavior in experiments
- Determining overhead of using *Hadoop* framework for parallel processing

For the present use case, a command-line tool was used to run the *Hadoop Streaming* [8] framework in a cluster without considering the parallel implementation of the tool.

2.2 Motivation

In the past years science moved forwards thanks to improved technical capabilities. In the field of genetics and genomics large amounts of raw data [**?**] has been retrieved. To take benefit of this collection and draw some conclusions, all the data has to be analyzed for further insights.
It is still difficult to handle massive datasets. A lot of time is invested to scale out known algorithms, such as classifying genoms and proteins. With our work, we want to show the simplicity of building a distributed system.

2.3 Ongoing situation

To process a big set of input data in parallel system bears some problems.
First: The more different hardware components are used, the more hardware failures [18] are likely to occur. Mitigating these can be handled by software.
Second: Big dataset have to be partitioned over several hosts for fault tolerance and raised I/O operations per second. Third: After parallel processing on different systems, a task is needed that aggregates the results of each system to one final result.
In 2008 [6] *Google* introduced the *MapReduce* principle for processing data on clusters. The *Hadoop* [17] framework implements this principle.
IBM released *InfoSphere BigInsights* which is based on *Hadoop* and brings some additional features [5] providing simplicity to manage *Big Data*.

2.4 Classification of proteins

There are numerous schemes to classify a protein in different groups. The challenge is that all implemented algorithms and tools do not provide as high a level of certainty and accuracy as classification of the proteins by human experts. On the other hand, humans have a high time consumption to complete such tasks.

2.4.1 Amino acids

Amino acids are essential elements of the organic system. Twenty three amino acids are known as proteinogenic. These amino acids are found in proteins. There is an official representation for amino acids which assigns a certain character to every amino acid.
Table 2.1 shows the IUB/IUPAC standard codes.[14]

Table (2.1) The IUB/IUPAC standard codes for amino acids

Code	**Meaning**
A	Alanine
B	Aspartic acid or Asparagine
C	Cysteine
D	Aspartic acid
E	Glutamic acid
F	Phenylalanine
G	Glycine
H	Histidine
I	Isoleucine
K	Lysine
L	Leucine
M	Methionine
N	Asparagine
O	Pyrrolysine
P	Proline
Q	Glutamine
R	Arginine
S	Serine
T	Threonine
U	Selenocysteine
V	Valine
W	Tryptophan
Y	Tyrosine
Z	Glutamic acid or Glutamine

2.4.2 Proteins

Proteins are built as a chain of different amino acids. Some proteins are well-known and their functions are clearly identified. A certain protein is classified by predicting its properties and function. To predict a proteins' function, an algorithm has to compare the sequence of amino acids with all sequences in a database of already known proteins.

2.4.3 Protein sequence databases

A number of databases which store information about proteins their amino acid sequences and properties.
The *UniProt* knowledge base supports two different databases:

- *Swiss-Prot* database [19], where all 539,616 entries are manually annotated and reviewed.
- *TrEMBL* database [20] with 32,153,798 entries, which are automatically annotated and are not beeing reviewed.

There are releases of the *Uniprot* database every four weeks. It is possible to download the whole database as FASTA-file from *Uniprot*'s FTP directory [4].

2.5 FASTA-format

The FASTA-format [7] is used to store a sequence of amino acids. There are two different kinds of records for each sequence:

- Header line
  Contains the description and identification of the sequence starting with the symbol ">" directly followed by an ID of the database and then followed by a blank and a description name.
- Sequence line
  One or more lines with the sequence of amino acids. Each line should not be longer than 80 characters. The amino acids are written in the IUB/IUPAC [14] standard code of amino acids.

```
>sp|P69905|HBA_HUMAN Hemoglobin subunit alpha OS

MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP
AVHASLDKFLASVSTVLTSKYR
```

Listing (2.1) FASTA-Format

2.5.1 Classification

The spotlight is set on two different classifications: structure analysis and sequence analysis.

2.5.2 Structural analysis

In structural analysis every amino acid is assigned in a different structural category. We focus on the three classes: alpha-Helix(H), beta-Strand(E) and Loops(C). For structural analysis we take advantage of the *RaptorX* command line tool.

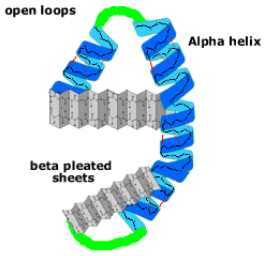

Figure (2.1) Protein secondary structure [3]

2.5.3 Sequence analysis

In sequence analysis, the given amino acid sequence will be compared or aligned to all known sequences and their properties. *InterProScan* will provide an ID and some references to the GO database, so that some information about the properties of the given sequence is returned.

3 System and Methods

3.1 Hadoop

Based on the idea of the MapReduce principle from *Google* [6], *Yahoo!* began to build an open-source alternative called *Hadoop*. *Hadoop* handles massive amounts of data in a distributed system.
It allows to develop applications based on the *Hadoop* framework to run their own tasks, which are controlled by a central administration-node. It is built with a small amount of administration-nodes and many worker-nodes. The administration-nodes are used as a job-tracker and namenodes to distribute and manage tasks on the worker-nodes. If one worker-node fails, the task is marked as incomplete and will be sent to another worker-node. This allows a network with heterogeneous system as a cluster which carry on the same work. This calculation is normally done on one system and sequential for all sequences.
The *Hadoop* Framework is partitioned into different components: *HDFS*, *MapReduce* and common utilities. The *HDFS* provides an filesystem-layer over all distributed nodes. *MapReduce* allows one to write custom software and to run map reduce jobs. Common utilities are used to manage, start and stop a *Hadoop* cluster.

3.2 Hadoop Streaming

Hadoop provides a utility called *Hadoop* Streaming [8] to build map or reduce tasks in other languages than *Java*. It is possible to build and run a map and reduce task without programming and configuring *Java* Classes.
Hadoop Streaming asks for a executable or script which reads and writes to stdin and stdout. The given task is then built, executed and monitored directly on the cluster.

3.3 IBM BigInsights

IBM BigInsights contributes an easy to use webconsole, web-installer and many analytic and developer-related tools. It contains *Apache Hadoop* with associated products for a ready-to-use environment. Futhermore, it bundles a specific version of *Hadoop* and its components such as *hive, flume, oozie, hbase* and others within one component, which can be selected during the installation. The following are the benefits [5] of using *IBM BigInsights*:

- Easy to use installer
- Realtime-view of cluster status and fine graded perspective on tasks and jobs
- Distribution of configuration-files over all nodes
- Scripts for cluster-management (adding node, healthcheck of cluster, complete cluster start and stop)
- Enhanced security with *LDAP*-access

- Flexible job scheduling mechanism
- Developer related functions such as eclipse plugin for linking references and automatic upload of code fragments
- Simple deployment and distribution of custom *java* applications on all nodes
- Already pre-deployed sample applications, such as wordcount

3.4 IBM BigSheets

IBM BigSheets allows one to use the output of the *MapReduce* task with a spreadsheet-like environment. For large output-files *IBM BigSheets* allows loading the data in a lazy-loading manner. It has many functions built-in such as counting rows, sorting output and building charts based on files inside *HDFS* based on *MapReduce* principle. *IBM BigSheets* is very useful where the data size is larger than normal spreadsheet tools can handle.

3.5 Classifiers

While researching the topic of classifying proteins, the following two classifiers were found: *InterProScan* and *RaptorX*. Both process FASTA-files with their own implemented algorithm. The tools were used within a *Hadoop Streaming* job to calculate the sequence of some proteins in a parallel manner.

3.5.1 RaptorX - structural classification

With the help of co-examiner Romeo Kienzler we established a contact to Jinbo Xu [1].
Jinbo provided us the standalone *RaptorX-Classifier*, which we built into our *Hadoop* cluster.
RaptorX is used to predict the secondary structure [15] of a protein based on a pre-calculated library. The following command shows the usage of this classifier:

```
1 ./buildFeature -i query.seq -c 2
```

Listing (3.1) Raptor-Command

The output is a long list of information. The relevant information for our work are the secondary structures [15]. The structure is predicted from *PSIPRED* [15] based on the amino acid likelihood. The amino acids are grouped the following subclasses:

- H (alpha helix)
- E (beta strand)
- C (loops)

The following table 3.1 (page 8) shows the output of the classifier

1 Jinbo Xu, main-author of the RaptorX Classifier, http://ttic.uchicago.edu/~jinbo/

Table (3.1) Three-class secondary structure prediction

Nr	A-Acid	Structure	H-likehood	E-likelihood	C-likelihood
1	M	C	0.000	0.000	1.000
2	K	C	0.015	0.070	0.915
3	K	C	0.036	0.219	0.745
4	I	E	0.275	0.363	0.362
5	W	H	0.463	0.409	0.128
6	L	H	0.559	0.411	0.030
7	A	H	0.575	0.404	0.022
8	L	H	0.597	0.387	0.016
9	A	H	0.610	0.376	0.014
10	G	H	0.608	0.366	0.026
...	...	...	...	...	...

3.5.2 InterProScan - sequence analysis

InterProScan is the classifier provided by the European Bioinformatics Institute [1]. It predicts not only through one analysis, it also supports many different algorithms which can be used through a single interface. All algorithm could be used at the same time for an optimal prediction. Figure 3.1 shows all the supported algorithms on *InterProScan*.

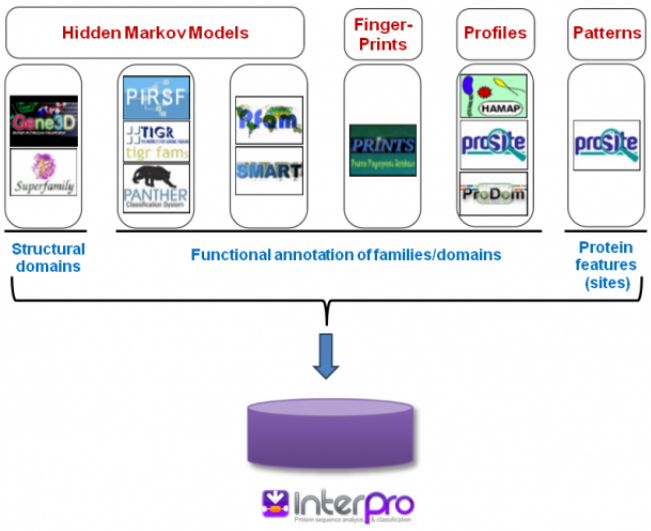

Figure (3.1) Supported algorithm on InterProScan [12]

Dr. Rémy Bruggmann [2] recommended to use the *PfamA-26.0* algorithm which makes a prediction about the appropriate family and domain of the input-sequence. The following command shows the usage of *InterProScan*:

1 The European Bioinformatics Institute http://www.ebi.ac.uk/
2 Dr. Rémy Bruggmann, Group Leader Bioinformatics University of Berne, http://www.bioinformatics.unibe.ch

```
./interproscan.sh -i query.fasta -appl PfamA-26.0 -f xml
```

Listing (3.2) InterProScan-Command

The command outputs an XML file with description about the family and domain:

```
<hmmer3-match evalue="2.5E-28" score="98.4">
  <signature ac="PF00042" desc="Globin" name="Globin">
  <entry ac="IPR000971" desc="Globin" name="Globin" type="FAMILY">
    <go-xref db="GO" id="GO:0005506"/>
    <go-xref db="GO" id="GO:0020037"/>
  </entry>
  <models>
    <model ac="PF00042" desc="Globin" name="Globin"/>
  </models>
  <signature-library-release library="PFAM" version="26.0"/>
  </signature>
  <locations>
    <hmmer3-location env-end="107" env-start="7" score="97.9"
     evalue="3.6E-28" hmm-start="1" hmm-end="108" hmm-length="0"
     start="7" end="107"/>
  </locations>
</hmmer3-match>
```

3.6 MapReduce tasks

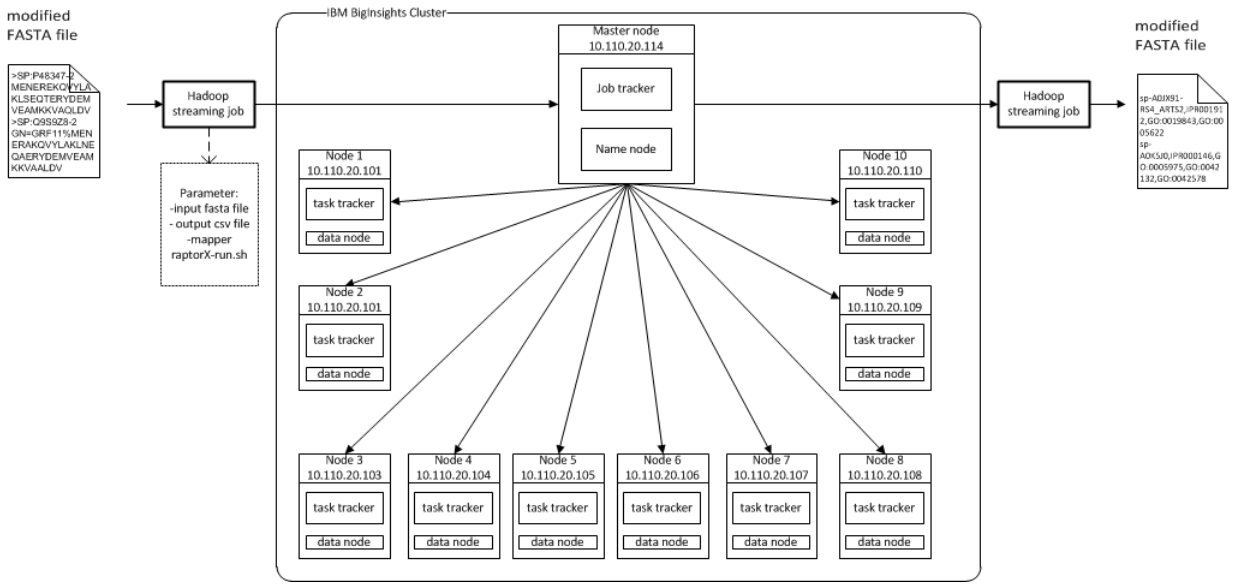

Figure (3.2) Workflow of Streaming job

Hadoop Streaming allows us using already existing executable binaries. *Hadoop Streaming* only support read and write from stdin/stdout. Given the executables we used, only working with files for input and output data we had to encapsulate those tools within a bash script. The script runs after the mapping-process and covers following necessary actions:

- Reads the input stream and converts the one-line input (see section 4.3) to the necessary FASTA-format(see section 2.5)
- Reads out the sequence-id of the stream for sequence recognition and creates a temporary-file
- Calls the classifier with this temporary-file
- Gets the output of the classifier and filters the necessary information
- Passes the id with classified information to stdout.

For all jobs only one reduce task is used to reduce all different output files on the workernodes to only one output file in *HDFS*.
Referencing to Listing A.2 (appendix p. 25) for more information about our implementation of the map task for the classifier *RaptorX*.
Further consult the Listing A.4 of *InterProScan* bash script (appendix p. 27) for more details.

4 Configuration and setup

To run the use-case on a *Hadoop* cluster the following tasks and steps are necessary:

- Setting up a cluster with *IBM BigInsights*
- Downloading all sequences in one large input-file
- Preparing and modifing the input file for saving on *HDFS*
- Loading data into *HDFS*
- Configuring a *MapReduce* job with *Hadoop Streaming* and running it on the system.
- Formatting *Hadoop Streaming* output to a CSV-output file for import in *IBM BigSheets*
- Interpreting the generated data

4.1 Hadoop cluster with IBM BigInsights

For a basic *Hadoop* runtime-environment, a single node cluster is needed at a minimum. *IBM BigInsights* provides a single-node-cluster installation as a test-environment or as a complete installation including worker-nodes through the installation wizard. The wizard made the installation much easier.

4.2 Cluster hardware configuration

Thanks to *IBM Switzerland* we executed our experiments on a bladecenter hardware cluster.

Figure (4.1) Test enviroment: hardware cluster located at IBM Switzerland

To conduct performance analyzes, we chose ten worker nodes with an identical performance. *Hadoop* has a primary focus on heterogeneous systems, to enable it to operate on different commodity hardware. However, for our use-case that was less relevant, since we wanted to illustrate scale out performance and the resulting overhead.

The following table lists hardware specification and the used role inside the *Hadoop* cluster for each blade.

Table (4.1) The servers of our test environment

Server	Model	Function	CPU	Memory	IP	FQDN
Blade1	IBM HS21	task node, data node	4	12 GB	10.110.20.101	Sa-biginsights-110-20-101-rh5.tec.app.ibm.com
Blade2	IBM HS21	task node, data node	4	12 GB	10.110.20.102	Sa-biginsights-110-20-102-rh5.tec.app.ibm.com
Blade3	IBM HS21	task node, data node	4	12 GB	10.110.20.103	Sa-biginsights-110-20-103-rh5.tec.app.ibm.com
Blade4	IBM HS21	task node, data node	4	12 GB	10.110.20.104	Sa-biginsights-110-20-104-rh5.tec.app.ibm.com
Blade5	IBM HS21	task node, data node	4	12 GB	10.110.20.105	Sa-biginsights-110-20-105-rh5.tec.app.ibm.com
Blade6	IBM HS21	task node, data node	4	12 GB	10.110.20.106	Sa-biginsights-110-20-106-rh5.tec.app.ibm.com
Blade7	IBM HS21	task node, data node	4	12 GB	10.110.20.107	Sa-biginsights-110-20-107-rh5.tec.app.ibm.com
Blade8	IBM HS21	task node, data node	4	12 GB	10.110.20.108	Sa-biginsights-110-20-108-rh5.tec.app.ibm.com
Blade9	IBM HS21	task node, data node	4	12 GB	10.110.20.109	Sa-biginsights-110-20-109-rh5.tec.app.ibm.com
Blade10	IBM HS21	task node, data node	4	12 GB	10.110.20.110	Sa-biginsights-110-20-110-rh5.tec.app.ibm.com
Blade14	IBM HS22V	job tracker, name node	16	30 GB	10.110.20.114	Sa-biginsights-110-20-114-rh5.tec.app.ibm.com

4.3 Preparing input dataset for HDFS

In *HDFS* all files are stored over different nodes. The default block-size in *IBM BigInsights* is configured with 128MB memory. To process an input dataset, *Hadoop* reads the file block by block and delivers this to the map task. There are different input-formats, per default the file content is provided line by line to the map task.
The input of the classifier has to be in FASTA-format [7]. However the FASTA-format consists of more than one line text-input for a protein sequence.
This problem was solved by writing a short C++ program that reads FASTA-input file which was downloaded from *UniProt* [4] and wrote out one line for each sequence. Whereas the one-line record had to be split afterwards back to FASTA-format, we added as delimiter symbol "%" after the header line to differ the headline with the sequence.
For more information about our program refer to the listing A.1 in the Appendix (page 25).

4.4 Load data to HDFS

To load some data into *HDFS IBM BigInsights* provides an upload function in the web console. This also can be done by simple line commands [10]:

```
1 hadoop fs -put localfile /user/hadoop/hadoopfile
```

Listing (4.1) HDFS put

4.5 Running tasks with Hadoop Streaming

Thanks to the *Hadoop Streaming* utility, a map task can be specified through any executable or script.
The following is an example of the command that was issued during performance tests:

```
1 hadoop jar /opt/ibm/biginsights/IHC/hadoop-streaming.jar -input /user/biadmin/
    fasta/input/db_dump_short -output output_10run -mapper /opt/scripts/raptorX-
    run.sh
```

Listing (4.2) Streaming Job

4.6 Configuration parameters for Hadoop Streaming

Within the *Hadoop Streaming* command it is feasible to adjust parameters of the *Hadoop* instance for the current running job.
In Table 4.2 (page 15) all parameters that were used by us are descripted.

Table (4.2) Used parameters for Hadoop Streaming job

Parameter	Description
-D mapred.map.tasks=*	defining number of Map Tasks based on input sequences
-D mapred.task.timeout=*	for RaptorX set timeout to a value above 5 min
-D keep.failed.task.files=true	If true, the files for failed tasks will be kept
-D mapred.map.max.attempts=*	maximal attempt of restart a failed task(default 4)
-D stream.non.zero.exit.is.failure=false	ignore non zero return code of executable(default false)

4.7 Generating output for IBM BigSheets

Hadoop writes the result of the mapping task to it's file-system. With *IBM BigInsights* these output files can be imported to *BigSheets* with a single click in the web console.
Simply write output to stdout in CSV format and import the file from *HDFS* to *IBM BigSheets*. The CSV from our project is generated by writing the sequence id and the information gained from the output of the classifier.

4.8 Scale out with nodes

IBM BigInsights provides shell scripts for managing the amount of nodes on a cluster. The following command removes nodes from the cluser:

```
1 /opt/ibm/biginsights/bin/removenode.sh sa-biginsights-110-20-110-rh5.tec.app.ibm.
    com
```

Listing (4.3) remove node

To evade the default replication factor within *HDFS*, the force option on the command has to be issued:

```
1 /opt/ibm/biginsights/bin/removenode.sh hadoop -f sa-biginsights-110-20-109-rh5.
    tec.app.ibm.com
```

Listing (4.4) force remove node

Using the force option automatically decrements the value of replica of files in *HDFS*. (If the number of nodes fall off the defined property.) The property is named dfs.replication and the default value is three.
To change the replication factor on files, the following command can be used:

```
1 hadoop dfs -setrep -R -w 1 /user/biadmin/
```

Listing (4.5) change replication factor to one for directory /user/biadmin

Adding nodes is also performed with scripts:

```
1 /opt/ibm/biginsights/bin/addnode.sh hadoop sa-biginsights-110-20-110-rh5.tec.app.
      ibm.com,pw
```

Listing (4.6) add node

Other management commands are starting and stopping the cluster:

```
1 /opt/ibm/biginsights/bin/start-all.sh
```

Listing (4.7) start the cluster

```
1 /opt/ibm/biginsights/bin/stop-all.sh
```

Listing (4.8) stop the cluster

5 Results

5.1 Facts

Below are the measurements of the experiments with the *InterProScan* tool

Table (5.1) Measurements InterProScan

sequences	native	1 node	5 nodes	10 nodes
1	117	133	138	137
5	585	617	136	139
10	1167	1217	253	251
40	4661	4842	1092	622
100	11673	12082	2541	1369
250	29146	30194	6449	3246
500	58314	62348	12828	6261
1000	116622	124537	25773	12497

Below are the measurements of the experiments with the *RaptorX* tool

Table (5.2) Measurements RaptorX

sequences	native	1 node	5 nodes	10 nodes
1	425	442	441	472
5	2128	2157	443	509
10	4304	4306	876	515
40	17026	17191	3459	1767
100	42569	44137	9065	4548
250	106427	110557	22409	11572

5.2 Cost of parallel processing

Figure 5.1 and 5.2 demonstrate the overhead of processing the sequences with *InterProScan* on our *Hadoop* cluster.

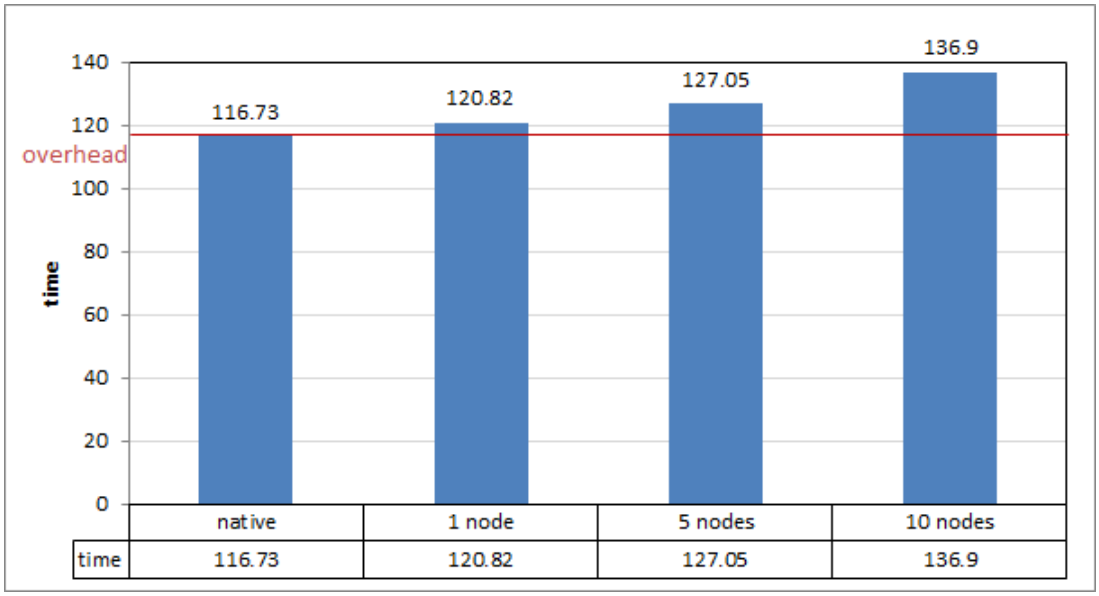

Figure (5.1) Mean-overhead of a InterProScan run with 100 various sequences

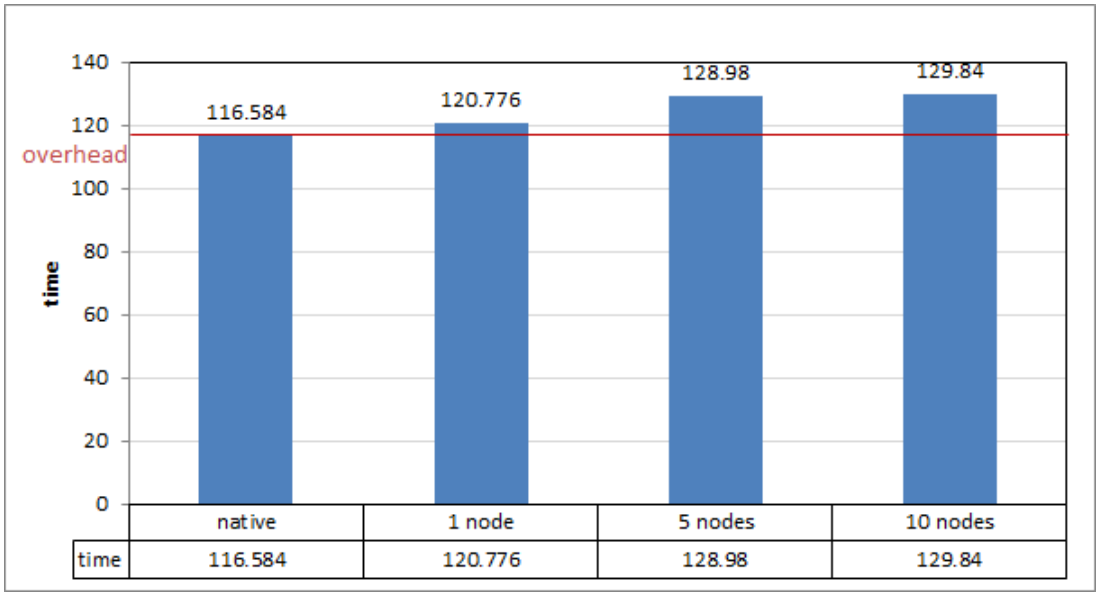

Figure (5.2) Mean-overhead of a InterProScan run with 250 various sequences

Figure 5.3 and 5.4 demonstrate the overhead of processing the sequences with *RaptorX* on our *Hadoop* cluster.

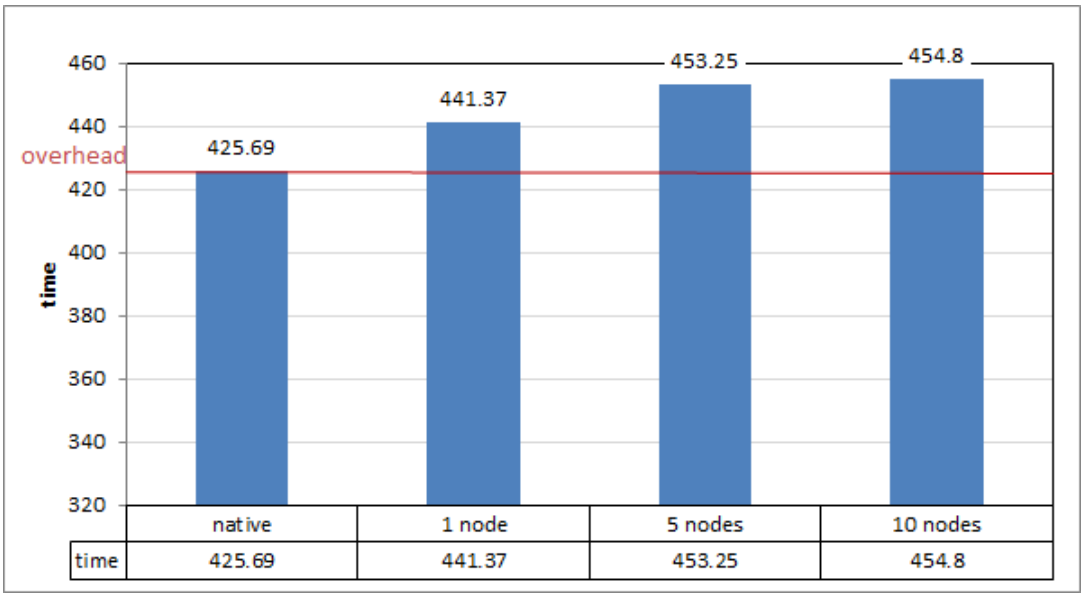

Figure (5.3) Mean-overhead of a RaptorX run with 100 equal sequences

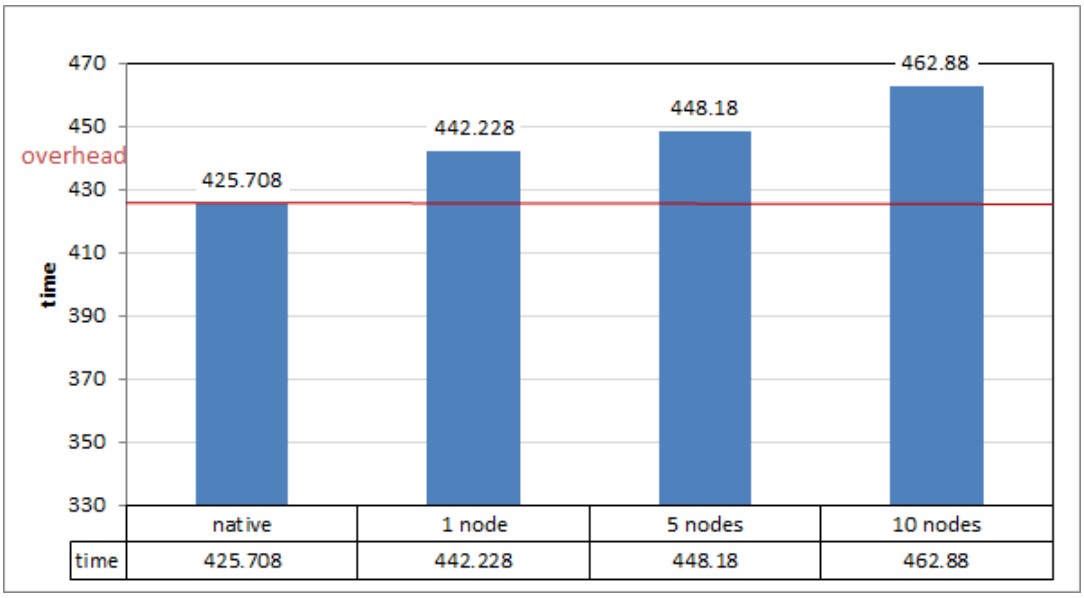

Figure (5.4) Mean-overhead of a RaptorX run with 250 equal sequences

The values are calculated by following formula:

```
measured time / amount of sequences * cluster size
```

Adding more nodes raises the CPU-power for handling more sequences in parallel. On the other hand managing more nodes will increase the average calculation time for one sequence, which results in a higher overhead per sequence.

5.3 Scale out

Scale out describes performance characteristics of adding more nodes to a cluster. This development is preferable linear, which means adding one more host to an already existing one node cluster results in double computing power. The additional needed power to distribute the tasks is called overhead and should be as less as possible.

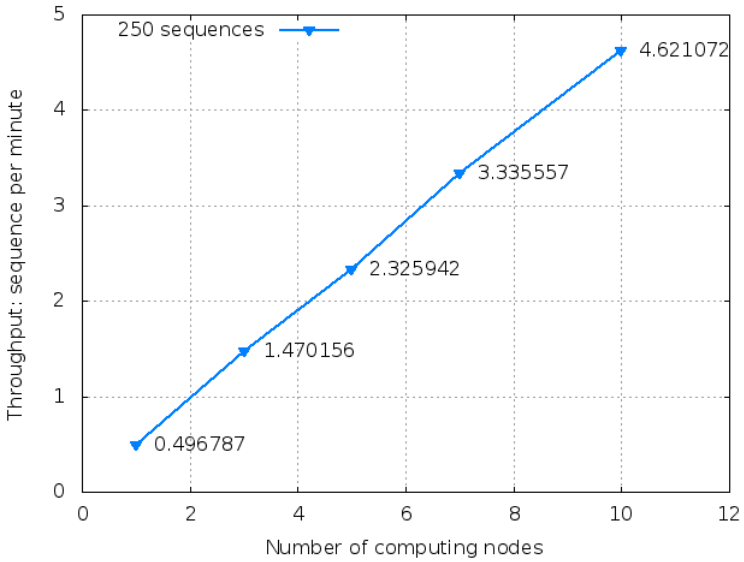

Figure (5.5) Throughput of InterProScan with 250 various sequences

As displayed in figure 5.5 the scale out behavior is almost linear. Adding more nodes results in linear less compution time. We normalized the measurement values with the formula:

```
number of sequences / (measured time / 60)
```

The calculated values describes the throughput. The throughput itself describes the possible sequence calculation per minute.
As displayed in figure 5.6 the scale out behavior on our cluster with the usage of *RaptorX* for equal 100 sequences and equal 250 sequences is also almost linear.

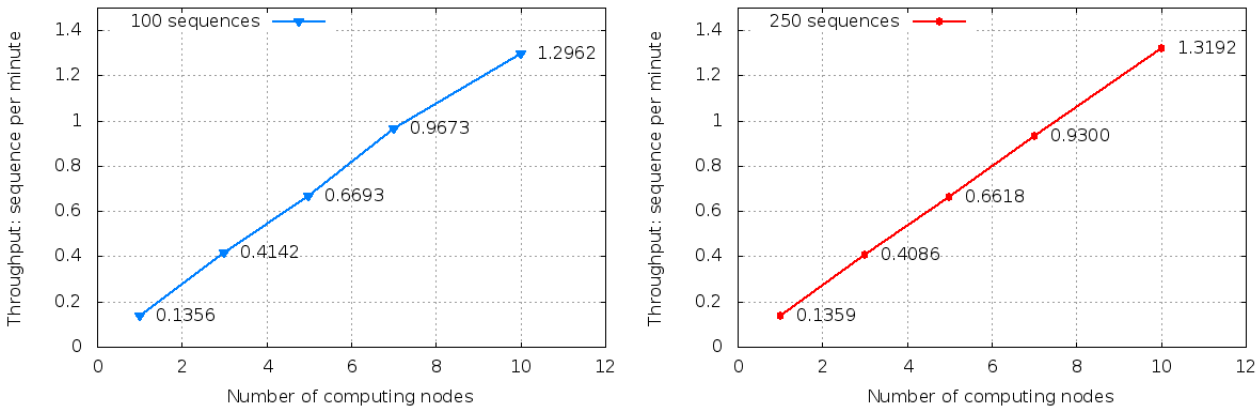

Figure (5.6) Throughput of RaptorX

5.4 Interpretation of the generated output files

5.4.1 InterProScan

As mentioned in section (System and Methods) *InterProScan* will provide us IDs of the *InterProScan* database and of the GO database. To give an overview of the used GO-IDs and IPR-numbers on the *InterProScan* output file, we ran the sample application "WordCount" on cluster.
The following charts expose the occurencye of the GO-ID that *InterProScan* reported for 500 randomly choosen sequences:

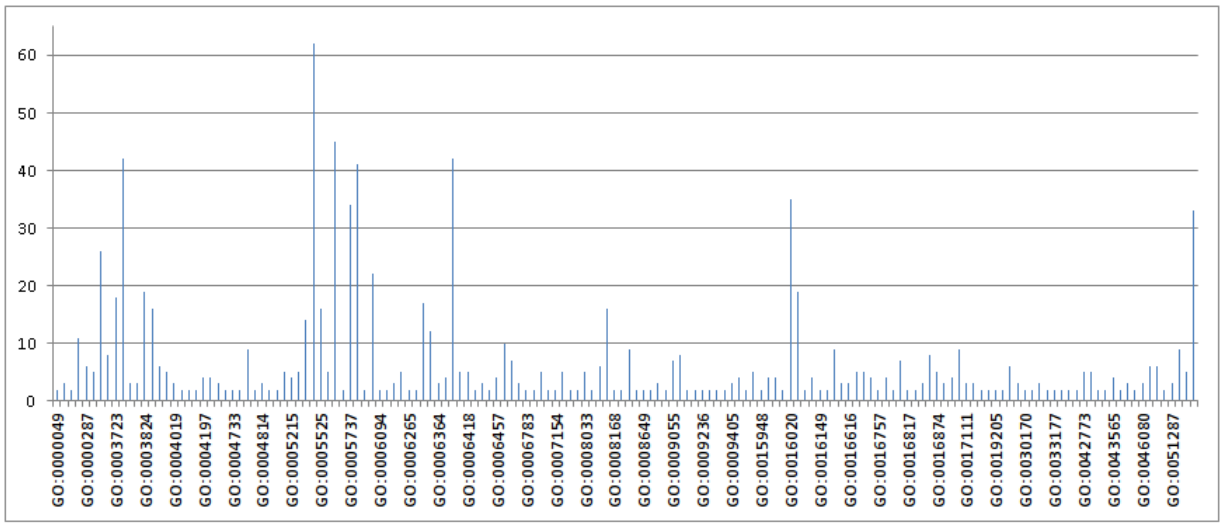

Figure (5.7) Occurency of GO-IDs

The GO-ID "GO:0005524" had the greatest occurence with 62 counts. The entry is associated with the property "ATP binding". The second most occurence was the GO-ID "GO:0005622" with the property "intracellular". 227 GO-IDs appread only once.
In Table B.1 (appendix p. 30) all GO-IDs with three or more occurences of GO-IDs in the output file are listed.
The following chart illustrate the frequency of the IPR-numbers that *InterProScan* reports from 500 sequences:

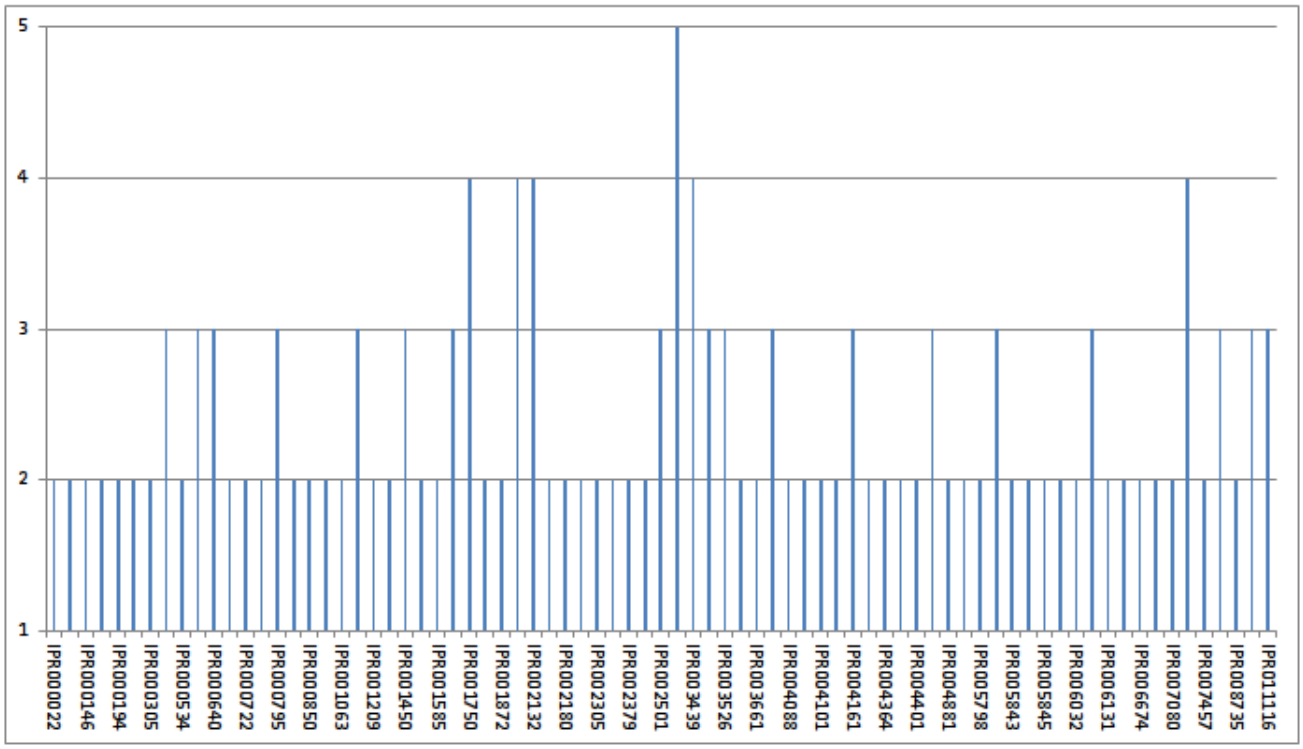

Figure (5.8) Occurency of IPR-numbers

Five of the 500 sequences belong to the domain "RNA-binding S4", which was reported through IPR-number IPR002942 and four of 500 sequences were asigned to the family "Ribosomal protein L5", which is indicated by IPR-number IPR002132. The IPR-numbers are more specific and detailed entries than the GO-IDs. They provide a family or a domain to a certain protein.
All IPR-numbers with two or more appearances in the output file of *InterProScan* run were listed in Table B.2 (appendix p. 33) and Table B.3 (appendix p. 34).

5.4.2 RaptorX

To interpret the output of the *RaptorX* tool, the frequency on the three different structures is counted and reported for each sequence.
The loop structure appeared at minimum five times in all of the 247 different sequences of amino acids. There were 10 sequences without a alpha helix structure and 30 sequences without a beta strand structure.
All details about the amount of these structures per sequences are listed in Table B.4 (appendix p. 35).

6 Discussion

6.1 Basic Hadoop

6.1.1 HDFS - replication and blocksize

An important idea behind the worker-node is locality of the data and the processing of the same data. It is important to process the data on the node where it resides. The locality is a big benefit for I/O-intensive applications. In best practices [11], the input data is split in 64 MB or 128 MB blocks. All of these blocks are replicated over three worker-nodes for fault tolerance.
In our case, the input size is much smaller than in usual use-cases: A 2000 sequence file uses about 1 MB of disk space and need 2.5 hours to process it with *InterProScan* on our 10node cluster.

In this situation *Hadoop* allows us the following options to tune I/O access time:

- Use a very small blocksize (around a few kilobytes) which results that our file is split into multiple blocks and those blocks could be local processed.
- Replicate our data to all nodes to guarantee that tasks would be scheduled on all nodes equal at the beginning of the *Hadoop Streaming* job.

We decided to not consider I/O access time. The time for the TCP connection to send a few characters and metadata in relation to the processing time was very small. The jobs were much more CPU-intensive then I/O-intensive.

6.1.2 CPU capacity utilization

During our research and experiments we had a lot of timeouts from the task-node. First we optimized our cluster for maximum performance. All four cores on our machines were used. The nodes had four tasks with 100 percent CPU usage. The task-tracker could not respond to heartbeats from the management node which resulted in the worker-node being marked as offline/unreachable. In default-settings of *IBM BigInsights*, the formula is "quantity of cores - 1" for the quantity of maximum mapping tasks. We ran one task per node with three cores (from available four).

6.2 Hadoop Streaming framework

6.2.1 Streaming mnnerism

The *Hadoop Streaming* framework allows to execute an external executable as a MapReduce job. Negative aspects of this option are that the executable has to be distributed to all worker-nodes. For a small executable such as the *UNIX* cat-command, the parameter -file= exists which copies the file to the worker-node. In our case that was inappropriate because our tools (*RaptorX* and *InterProScan*) using libraries in background with many thousands of files (totally a site of approximately 40 gigabytes).
A speciality of the *Streaming* framework is to provide the input data as stdin and receive the output of the executable as stdout. To receive the data from input stream is unhandy for executable which

uses a file for their calculations. Both of our used classifiers needed a file, because they could process different algorithms in background which read the file many times. That made a virtual file also impossible, because a virtual file based on a stream could only be read once. At second reading the data would be different to the former reading. To meet our requirements we had to pack our executable in shell scripts. In the beginning of the script we processed the input stream and wrote it to a temporary file, which was then used by the executable, and then the output file piped to stdout.

6.2.2 The Streaming adaptability

Hadoop Streaming allowed us to use all available *Hadoop* job parameters for our task. We used a few in our jobs such as "-D mapred.map.tasks=500". This was necessary because *Hadoop Streaming* has a default value of two map tasks. It is not the idea of *Hadoop* to generate large map tasks. For example: With a 500 sequence input file two nodes are calculating 250 sequence as one map task. *Hadoop* likes small tasks which can be redistributed to another node at failure. As a result, we ran at a minimum as many map tasks as sequences on our input file.

6.2.3 Failure handling

Hadoop has a primary viewpoint of failure handling. A failure does not mean a complete job-failure. It tells the management-node that the worker-node was unnable to complete the given task and the management-node distributes the task to another worker-node. If a failure occurs four times in a row, the task is killed but the job continues. If a task contains bad input, the task will fail every time it is executed. Executing it on a different node has no other result. It will fail again and again. There are some possibilities to tell *Hadoop* that a non zero return code of the executable has not to be marked as failed attempt. One is to set the property "mapred.max.map.failures.percent", which is unfortunately not supported by the *Hadoop* version *(Hadoop 1.0.3)* we used. The second option is to set the property "stream.non.zero.exit.is.failure=false" to ignore a non zero return code. We had to change this property dependent on the task. It was set to false only for the *RaptorX* runs. We also decremented the number of maximum task attempts "mapred.map.max.attempts" to a value of two because one bad input sequence could easily generate a useless server-run of about one hour when using *RaptorX*.

6.3 Conclusion

At the beginning, we were able to obtain initial success within short time. The good documentation of *Hadoop Streaming* let us take the first steps quickly. Without completely understanding the *Hadoop* ecosystem, we created some simple *MapReduce* jobs. Based on the information of the job tracker site, we gained insight into the complete task-handling and execution of *MapReduce* jobs. This disclosed the important facts about task trackers, datasets and the capabilities of the cluster. During our experiments, *IBM Biginsights* facilitated the distribution of cluster related settings.
The benefit of *Hadoop* is a flexible configuration of parameters to fit the different needs of use cases. In summary, there are about 160 available properties to configure for the cluster and its dependencies. Some of the default configurations were inaccurate for our usage. Nevertheless, setting the correct parameters was difficult and had to be based on experience [1]. We think it will going to be a time consuming task to find best practice values for other use-cases. We conclude it is not a considered task when attempting to install and use a *Big Data* system like *Hadoop*. Although *Hadoop* is discussed in a lot communities [16] [9], there are only a few statements about best practices [13] [2]. We expect that the increasing use of *Hadoop* in real production environments will improve knowledge about best practices.

Appendix A

Program listings and bash scripts

```
#include <iostream>
using namespace std;

int main() {

    std::string s{};
    int c = 0;
    while(getline(std::cin,s)) {
        if (s.front() == '>') {
            if (c != 0)
                std::cout << endl;
            else
                c = 1;
            std::cout << s << "%";
        } else {
            std::cout << s;
        }
    }
    return 0;
}
```

Listing (A.1) C++ Programm FASTA2oneLine

```
#!/bin/bash
# aufruf mit cat file | ./raptorX-run.sh >> output.csv

# 1.) read stdin
# 2.) for each sequenz save ID and store as $seqID, write ID to stdout as csv-
    detail-line
# 3.) create a file with the sequence in FASTA format in temp folder
# 4.) run raptorx, and filter output and writes out for csv-detail-line

#echo "seqID, H-value-count, E-value-count, L-value-count"
while read sequence
do

 # 2. Sequenz ID
 idx=`expr index "$sequence" " "`
 seqID=${sequence:1:$idx-2}
 seqID=$(echo "$seqID" | tr '|' '-' )
 echo -n "$seqID,"

 # 3. create FASTA file
 OIFS=$IFS
```

```
IFS='%'
arr2=$sequence
count=0
for x in $arr2
 do
  if [ $count -eq 0 ]
   then
     echo "$x" > /opt/temp/$seqID
   else
     echo "$x"| sed -e "s/.\{60\}/&\n/g" >> /opt/temp/$seqID
  fi
  count=`expr $count + 1`
 done
 IFS=$OIFS

# 4. raptorX && # 5. filter output
cd /opt/CNFsearch1.4/
./buildFeature -i /opt/temp/$seqID -o /opt/temp/$seqID.tgt -c 1 >> /dev/null &&
    /opt/scripts/raptorX-output-filtern.sh /opt/temp/$seqID.tgt

done
```

Listing (A.2) Bash-script RaptorX run

```
#!/bin/bash
#process last part of .tgt file fromraptorX
#filters secondary structure information
#and counts the occurance of H,E and L values and write it to STDOUT as csv-
    detail-line

swDO=''
countH=0
countE=0
countL=0

 while read line
 do
  echo $line | grep -q "^############ Original SS3+SS8+ACC file"
  if [ $? -eq 0 ]
   then
     swDO='1'
      read line
      read line
      read line
  fi

  if [ "$swDO" = "1" ]
   then
     if [ "${line:0:1}" != "#" ]
      then
        Hvalue=${line:0:5}
        Evalue=${line:6:5}
        Lvalue=${line:12:5}

         compare_result=`echo "$Hvalue > $Evalue" | bc`
         if [ $compare_result -eq 1 ]
             then
                compare_result=`echo "$Hvalue > $Lvalue" | bc`
```

```
                if [ $compare_result -eq 1 ]
                   then
            countH=`expr $countH + 1`
                   else
            countL=`expr $countL + 1`
                fi
         else # Hvalue < Evalue
            compare_result=`echo "$Evalue > $Lvalue" | bc`
        if [ $compare_result -eq 1 ]
               then
           countE=`expr $countE + 1`
               else
           countL=`expr $countL + 1`
               fi
         fi
      else
        swDO=''
      fi
  fi

 done < $1
echo "$countH"",""$countE"",""$countL"
```

Listing (A.3) Bash-script RaptorX output filter script

```
#!/bin/bash

# 1.) read stdin
# 2.) for each sequenz save ID and store as $seqID, write ID to stdout as csv-
    detail-line
# 3.) create a file with the sequence in FASTA format in temp folder
# 4.) run interproscan, filter output and writes out for csv-detail-line
while read sequence
do

 # 2. Sequenz ID
 idx=`expr index "$sequence" " "`
 seqID=${sequence:1:$idx-2}
 seqID=$(echo "$seqID" | tr '|' '-' )
 echo -n "$seqID,"

 # 3. create FASTA file
 OIFS=$IFS
 IFS='%'
 arr2=$sequence
 count=0
 for x in $arr2
  do
   if [ $count -eq 0 ]
    then
      echo "$x" > /opt/temp/$seqID
    else
      echo "$x"| sed -e "s/.\{60\}/&\n/g" >> /opt/temp/$seqID
   fi
   count=`expr $count + 1`
  done
  IFS=$OIFS
```

```
# 4. interproscan mit seqID file aufrufen
 cd /opt/interproscan-5-RC5/
35 echo "$seqID" >> /opt/temp/interproscan_log
 ./interproscan.sh -i /opt/temp/$seqID -appl PfamA-26.0 -f xml >> /opt/temp/
     interproscan_log && cat /opt/temp/$seqID.xml | /opt/scripts/interproscan-
     output-filtern.sh

done
```

Listing (A.4) Bash-script InterProScan run

```
1 #!/bin/bash
 # search all GO-IDs and IPR-IDs and write them to STDOUT
  while read line
  do
5 echo -n $line | grep -o -e "GO:[0-9]\+" -e "IPR[0-9]\+" | tr '\n' ,
  done
 echo -e "\n"
```

Listing (A.5) Bash-script InterProScan output filter script

Appendix B

Streaming job output files

Table (B.1) Table of GO-ID and their Term name

GO-ID	Occ.	Term name
GO:0005524	62	ATP binding
GO:0005622	45	intracellular
GO:0003735	42	structural constituent of ribosome
GO:0006412	42	translation
GO:0005840	41	ribosome
GO:0016020	35	membrane
GO:0005737	34	cytoplasm
GO:0055114	33	oxidation-reduction process
GO:0003677	26	DNA binding
GO:0005975	22	carbohydrate metabolic process
GO:0003824	19	catalytic activity
GO:0016021	19	integral to membrane
GO:0003723	18	RNA binding
GO:0006351	17	transcription, DNA-dependent
GO:0003899	16	DNA-directed RNA polymerase activity
GO:0005525	16	GTP binding
GO:0008152	16	metabolic process
GO:0005515	14	protein binding
GO:0006355	12	regulation of transcription, DNA-dependent
GO:0000166	11	nucleotide binding
GO:0006508	10	proteolysis
GO:0004812	9	aminoacyl-tRNA ligase activity
GO:0008270	9	zinc ion binding
GO:0016491	9	oxidoreductase activity
GO:0017038	9	protein import
GO:0051536	9	iron-sulfur cluster binding
GO:0003700	8	sequence-specific DNA binding transcription factor activity
GO:0009058	8	biosynthetic process
GO:0016868	8	intramolecular transferase activity, phosphotransferases
GO:0006520	7	cellular amino acid metabolic process
GO:0009055	7	electron carrier activity

GO:0016787	7	hydrolase activity
GO:0000287	6	magnesium ion binding
GO:0003924	6	GTPase activity
GO:0008137	6	NADH dehydrogenase (ubiquinone) activity
GO:0019843	6	rRNA binding
GO:0046872	6	metal ion binding
GO:0046983	6	protein dimerization activity
GO:0003676	5	nucleic acid binding
GO:0003968	5	RNA-directed RNA polymerase activity
GO:0005198	5	structural molecule activity
GO:0005506	5	iron ion binding
GO:0005576	5	extracellular region
GO:0006164	5	purine nucleotide biosynthetic process
GO:0006415	5	translational termination
GO:0006418	5	tRNA aminoacylation for protein translation
GO:0006810	5	transport
GO:0007165	5	signal transduction
GO:0008033	5	tRNA processing
GO:0015078	5	hydrogen ion transmembrane transporter activity
GO:0016620	5	oxidoreductase activity, acting on the aldehyde or oxo group of donors, NAD or NADP as acceptor
GO:0016740	5	transferase activity
GO:0016874	5	ligase activity
GO:0042773	5	ATP synthesis coupled electron transport
GO:0043039	5	tRNA aminoacylation
GO:0055085	5	transmembrane transport
GO:0004197	4	cysteine-type endopeptidase activity
GO:0004222	4	metalloendopeptidase activity
GO:0005215	4	transporter activity
GO:0006396	4	metalloendopeptidase activity
GO:0006457	4	protein folding
GO:0010181	4	FMN binding
GO:0015986	4	ATP synthesis coupled proton transport
GO:0015991	4	ATP hydrolysis coupled proton transport
GO:0016114	4	terpenoid biosynthetic process

GO:0016743	4	carboxyl- or carbamoyltransferase activity
GO:0016773	4	phosphotransferase activity, alcohol group as acceptor
GO:0016887	4	ATPase activity
GO:0043565	4	sequence-specific DNA binding
GO:0000105	3	histidine biosynthetic process
GO:0003747	3	translation release factor activity
GO:0003774	3	motor activity
GO:0004019	3	adenylosuccinate synthase activity
GO:0004252	3	serine-type endopeptidase activity
GO:0004814	3	arginine-tRNA ligase activity
GO:0006139	3	nucleobase-containing compound metabolic process
GO:0006364	3	rRNA processing
GO:0006420	3	nucleobase-containing compound metabolic process
GO:0006526	3	arginine biosynthetic process
GO:0008685	3	2-C-methyl-D-erythritol 2,4-cyclodiphosphate synthase activity
GO:0009405	3	pathogenesis
GO:0016597	3	amino acid binding
GO:0016616	3	oxidoreductase activity, acting on the CH-OH group of donors, NAD or NADP as acceptor
GO:0016829	3	lyase activity
GO:0016876	3	ligase activity, forming aminoacyl-tRNA and related compounds
GO:0017111	3	nucleoside-triphosphatase activity
GO:0019028	3	viral capsid
GO:0020037	3	heme binding
GO:0030976	3	thiamine pyrophosphate binding
GO:0045263	3	proton-transporting ATP synthase complex, coupling factor F(o)
GO:0046080	3	dUTP metabolic process
GO:0051287	3	NAD binding

Table (B.2) Table of IPR-number and their Domain

IPR-number	Occ.	Domain
IPR002942	5	RNA-binding S4 domain
IPR001750	4	NADH:ubiquinone/plastoquinone oxidoreductase
IPR001912	4	Ribosomal protein S4/S9, N-terminal
IPR003439	4	ABC transporter-like
IPR000352	3	Peptide chain release factor class I/class II
IPR000640	3	Translation elongation factor EFG, V domain
IPR000795	3	Elongation factor, GTP-binding domain
IPR001450	3	4Fe-4S binding domain
IPR003526	3	2-C-methyl-D-erythritol 2,4-cyclodiphosphate synthase
IPR004161	3	Translation elongation factor EFTu/EF1A, domain 2
IPR005824	3	KOW
IPR006073	3	GTP binding domain
IPR011115	3	SecA DEAD-like, N-terminal
IPR011116	3	SecA Wing/Scaffold
IPR011130	3	ecA preprotein, cross-linking domain
IPR012947	3	Threonyl/alanyl tRNA synthetase, SAD
IPR013842	3	GTP-binding protein LepA, C-terminal
IPR018449	3	NIL domain
IPR020069	3	Ribosomal protein L9, C-terminal
IPR020070	3	Ribosomal protein L9, N-terminal
IPR000022	2	Carboxyl transferase
IPR000073	2	Alpha/beta hydrolase fold-1
IPR000157	2	Toll/interleukin-1 receptor homology (TIR) domain
IPR000194	2	ATPase, F1/V1/A1 complex, alpha/beta subunit,nucleotide-binding domain
IPR000305	2	GIY-YIG nuclease superfamily
IPR000534	2	Semialdehyde dehydrogenase, NAD-binding
IPR000713	2	Mur ligase, N-terminal
IPR000722	2	RNA polymerase, alpha subunit
IPR000793	2	ATPase, F1/V1/A1 complex, alpha/beta subunit, C-terminal
IPR000836	2	Phosphoribosyltransferase domain
IPR001347	2	Sugar isomerase (SIS)
IPR002319	2	Phenylalanyl-tRNA synthetase
IPR002379	2	V-ATPase proteolipid subunit C-like domain
IPR003594	2	Histidine kinase-like ATPase, ATP-binding domain
IPR003661	2	Signal transduction histidine kinase
IPR004088	2	K Homology domain, type 1
IPR004100	2	ATPase, alpha/beta subunit, N-terminal
IPR004101	2	Mur ligase, C-terminal
IPR004115	2	GAD domain
IPR004364	2	Aminoacyl-tRNA synthetase, class II (D/K/N)
IPR004365	2	Nucleic acid binding, OB-fold, tRNA/helicase-type
IPR005139	2	Peptide chain release factor
IPR005798	2	Cytochrome b/b6, C-terminal
IPR005843	2	Alpha-D-phosphohexomutase, C-terminal
IPR005844	2	Alpha-D-phosphohexomutase, alpha/beta/alpha domain I

IPR005845	2	Alpha-D-phosphohexomutase, alpha/beta/alpha domain II
IPR005846	2	Alpha-D-phosphohexomutase, alpha/beta/alpha domain III
IPR006131	2	Aspartate/ornithine carbamoyltransferase,Asp/Orn-binding
IPR006132	2	Aspartate/ornithine carbamoyltransferase,carbamoyl-P-binding
IPR006674	2	HD domain
IPR007066	2	RNA polymerase Rpb1, domain 3
IPR007080	2	RNA polymerase Rpb1, domain 1
IPR011260	2	RNA polymerase, alpha subunit, C-terminal
IPR011261	2	DNA-directed RNA polymerase, dimerisation
IPR011262	2	DNA-directed RNA polymerase, insert domain
IPR011576	2	Pyridoxamine 5'-phosphate oxidase-like, FMN-binding domain
IPR013022	2	Xylose isomerase-like, TIM barrel domain
IPR013106	2	Immunoglobulin V-set
IPR013221	2	Mur ligase, central
IPR013317	2	Chromosomal replication control, initiator
IPR013520	2	Exonuclease, RNase T/DNA polymerase III
IPR018164	2	Alanyl-tRNA synthetase, class IIc, N-terminal
IPR018484	2	Carbohydrate kinase, FGGY, N-terminal
IPR018485	2	Carbohydrate kinase, FGGY, C-terminal
IPR018948	2	GTP-binding protein TrmE, N-terminal
IPR020603	2	MraZ domain
IPR021131	2	Ribosomal protein L18e/L15P
IPR022666	2	Ribosomal Proteins L2, RNA binding domain
IPR022669	2	Ribosomal protein L2, C-terminal
IPR024567	2	Ribonuclease HII/HIII domain
IPR024951	2	Sulphate adenylyltransferase catalytic domain
IPR025867	2	tRNA modification GTPase MnmE C-terminal domain
IPR025980	2	TP-sulfurylase PUA-like domain

Table (B.3) Table of IPR-number and their Family

IPR-number	**Occ.**	**Family**
IPR002132	4	Ribosomal protein L5
IPR000630	3	Ribosomal protein S8
IPR001114	3	Adenylosuccinate synthetase
IPR001705	3	Ribosomal protein L33
IPR002501	3	Pseudouridine synthase II
IPR003509	3	Uncharacterised protein family UPF0102
IPR004506	3	tRNA-specific 2-thiouridylase
IPR008180	3	DeoxyUTP pyrophosphatase
IPR000146	2	Fructose-1,6-bisphosphatase class 1/Sedoheputulose-1,7-bisphosphatase
IPR000276	2	G protein-coupled receptor, rhodopsin-like
IPR000850	2	Adenylate kinase
IPR001015	2	Ferrochelatase
IPR001063	2	Ribosomal protein L22/L17
IPR001209	2	Ribosomal protein S14
IPR001518	2	Argininosuccinate synthase
IPR001585	2	Transaldolase

IPR001763	2	Rhodanese-like domain
IPR001872	2	Peptidase A8, signal peptidase II
IPR002146	2	ATPase, F0 complex, subunit B/B', bacterial/chloroplast
IPR002180	2	6,7-dimethyl-8-ribityllumazine synthase
IPR002220	2	Dihydrodipicolinate synthetase-like
IPR002305	2	Aminoacyl-tRNA synthetase, class Ic
IPR002423	2	Chaperonin Cpn60/TCP-1
IPR004214	2	Conotoxin
IPR004401	2	Nucleoid-associated protein YbaB
IPR004881	2	Ribosome biogenesis GTPase RsgA, putative
IPR006032	2	Ribosomal protein S12/S23
IPR007457	2	Fe(II) trafficking protein YggX
IPR008735	2	Beta-microseminoprotein
IPR013025	2	Ribosomal protein L25/L23
IPR018317	2	Queuosine biosynthesis protein QueC

Table (B.4) Secondary structure prediction

sequence ID	alpha helix	beta strand	loops
sp A0JX91 RS4 ARTS2	71	27	110
sp A0RRG4 END4 CAMFF	103	27	152
sp A1AFE8 FETP ECOK1	44	1	46
sp A1B9H4 RL27 PARDP	0	36	54
sp A1JP96 XNI YERE8	109	25	117
sp A1RED3 RL15 SHESW	21	27	96
sp A1RT13 SYA PYRIL	328	134	430
sp A1UZV8 COXX BURMS	212	2	86
sp A1VER5 MNME DESVV	195	67	195
sp A1VYR0 EFP CAMJJ	8	87	94
sp A2AKB9 DCA10 MOUSE	0	202	364
sp A2BUD7 NU1C PROM5	257	6	109
sp A3Q6V2 RL332 MYCSJ	0	24	30
sp A4FZ98 COFD METM5	96	56	157
sp A4IIT5 OLM2A XENTR	107	122	413
sp A4QKC9 RR12 BARVE	7	36	80
sp A4W778 QUEA ENT38	65	78	213
sp A4WD10 MNTH ENT38	311	1	100
sp A4XHM5 RISB CALS8	74	25	57
sp A4YXQ6 SYA BRASO	338	126	427
sp A5F5P1 SECA VIBC3	459	52	392
sp A5G6H1 LPXK GEOUR	108	61	191
sp A5UBA3 Y3135 HAEIE	304	61	186
sp A6L0A5 PNP BACV8	211	131	369
sp A6LPU7 RS9 CLOB8	45	20	65
sp A6VKC6 RL7 ACTSZ	59	11	52
sp A7FME9 TAL YERP3	165	22	130
sp A7MFI9 TIG CROS8	201	64	167

sp A7VN14 MSMB2 PROFL	15	23	71
sp A7X3H9 Y1729 STAA1	105	64	116
sp A7ZJJ0 MOAA ECO24	99	36	194
sp A8ES24 HIS4 ARCB4	79	52	104
sp A8H1E5 HCP SHEPA	254	40	260
sp A8HS90 NDK AZOC5	59	18	63
sp A8KZF0 PDXT FRASN	60	49	94
sp A8SEC9 PSBB CERDE	149	88	271
sp A8Z4G7 MURC STAAT	149	94	194
sp A9A5I4 RL24 NITMS	33	37	98
sp A9LYX9 F16PA NEIM0	95	70	159
sp A9MEJ1 PURR SALAR	140	50	151
sp B0VBY5 MNMA ACIBY	80	96	201
sp B1HMT3 GLMM LYSSC	167	63	220
sp B1I956 MRAY STRPI	203	19	104
sp B1IIL4 RSGA CLOBK	75	69	148
sp B2A4C3 RL33 NATTJ	0	20	29
sp B2VJA4 PURT ERWT9	125	88	179
sp B3Q5X2 RL20 RHOPT	90	0	29
sp B3R808 Y3032 CUPTR	35	33	63
sp B4T750 NANA SALNS	137	35	125
sp B5F4C2 YJHX SALA4	27	17	41
sp B5F5J1 ARGO SALA4	191	0	20
sp B5FJK3 RL24 SALDC	0	49	55
sp B5XAM2 ICT1 SALSA	70	22	99
sp B5XJ20 PUR9 STRPZ	176	75	264
sp B5XJP1 NANE STRPZ	91	37	106
sp B5YH59 DNAK THEYD	213	111	308
sp B7GJC8 UPPP ANOFW	199	0	74
sp B7H9V0 Y577 BACC4	97	38	104
sp B7J6C1 PUR7 ACIF2	65	44	131
sp B7L649 RISB ECO55	72	26	58
sp B7M3B3 MTFA ECO8A	110	17	138
sp B7VH36 Y2060 VIBSL	12	7	40
sp B8GVN4 RL9 CAUCN	63	42	89
sp B8HYL7 DAPA CYAP4	131	34	129
sp B8ZSA6 RS14Z MYCLB	29	0	32
sp B9LPW4 RPON HALLT	34	0	30
sp C0QB49 PURA DESAH	121	74	234
sp C0QQN7 RS8 PERMH	31	35	70
sp C0SDJ3 COQ4 PARBP	156	0	129
sp C1A051 ASSY RHOE4	154	68	177
sp C1CPR4 RNH3 STRZT	126	46	121
sp C1ETQ0 Y4566 BACC3	99	35	91
sp C4ZRB8 PYRB ECOBW	115	40	156
sp C5A083 K6PF ECOBW	136	51	133

hline sp C5CV58 DCD VARPS	12	71	105
sp E1WAB4 ORGB SALTS	152	22	52
sp F7D4X9 SIR5 MONDO	91	29	195
sp O22431 RL10 PINTA	43	44	141
sp O23324 APS3 ARATH	145	67	253
sp O24385 CPI7 SOLTU	0	66	112
sp O31609 YJBK BACSU	49	61	80
sp O43002 SC61B SCHPO	38	7	57
sp O61069 KAD TRYBR	93	13	103
sp O74759 NTO1 SCHPO	228	27	512
sp O79213 CYB OCETR	228	0	152
sp O94225 HOSM PENCW	168	49	257
sp P01667 KV3AF MOUSE	0	36	75
sp P03330 GAG WMSV	202	4	306
sp P09125 MSP8 EIMAC	12	15	232
sp P0A6H9 CLS ECOL6	229	80	177
sp P0A7I2 RF1 ECO57	177	29	154
sp P0AA49 YFDV ECOLI	258	0	56
sp P0AAS2 YLAC SHIFL	85	22	49
sp P0ABB4 ATPB ECOLI	119	96	245
sp P0C2Z9 ATPH ORYSA	69	0	12
sp P0C5Q6 YM94A YEAST	17	28	29
sp P0CG46 CKBR CONSL	54	0	49
sp P0CP10 NAR1 CRYNJ	188	71	391
sp P13198 LMP1 EBVR	160	0	226
sp P15276 ALGP PSEAE	97	0	255
sp P21238 CPNA1 ARATH	267	66	253
sp P22087 FBRL HUMAN	42	70	209
sp P25508 COCA1 BOVIN	0	0	86
sp P25779 CYSP TRYCR	112	57	298
sp P27581 ADH2 DROAR	112	35	107
sp P29363 THRC PSEAE	214	59	196
sp P31552 CAIC ECOLI	160	86	271
sp P36512 UDB13 RABIT	250	62	219
sp P40219 OSW5 YEAST	83	2	63
sp P42528 ARP3 DICDI	113	59	246
sp P46950 SNG1 YEAST	266	30	251
sp P46953 3HAO RAT	27	99	160
sp P48377 RFX1 MOUSE	285	10	668
sp P53573 ETFA BRAJA	91	76	147
sp P53948 YNF7 YEAST	0	55	55
sp P55919 CXCR1 GORGO	213	15	122
sp Q72J71 VATF THET2	45	18	41
sp P57087 JAM2 HUMAN	7	144	147
sp P57103 NAC3 HUMAN	279	201	447
sp P58063 TRUB CAUCR	61	79	170
sp P59511 ATS20 MOUSE	93	385	1428

sp P61806 DAD1 MESAU	81	2	30
sp P76241 YEAM ECOLI	102	45	126
sp Q566C7 NUDT3 RAT	32	48	88
sp B4SJB0 GLYA STRM5	177	30	210
sp P83038 HDAC4 CHICK	378	33	669
sp P84391 UL13 EHV1V	124	64	406
sp P93746 EC1 ARATH	0	9	75
sp P96602 DCTR BACSU	100	37	89
sp Q00408 VHUD METVO	51	21	62
sp Q00758 SP5B BACSU	457	0	61
sp Q01JR9 MRS2D ORYSI	211	17	206
sp Q02944 URED KLEPN	29	118	123
sp Q02DE9 ATPL PSEAB	73	0	12
sp Q02VS5 MUTL LACLS	185	107	364
sp Q03667 MIC17 YEAST	60	0	96
sp Q03DS8 GLPK PEDPA	172	54	278
sp Q03QT7 TRUB LACBA	67	74	162
sp Q06199 YL456 YEAST	34	62	108
sp Q07ZS6 UVRC SHEFN	241	81	288
sp Q086C3 NUSB SHEFN	90	0	44
sp Q0ARD7 PDXH MARMM	56	58	108
sp Q0C5F3 SUCC HYPNA	140	78	179
sp Q0G9M8 RPOC1 LIRTU	211	69	401
sp Q0TQ59 BIOB CLOP1	133	23	163
sp Q0VSV6 HEMH ALCBS	148	42	151
sp Q13478 IL18R HUMAN	50	228	263
sp Q14032 BAAT HUMAN	81	105	232
sp Q1CFH7 METN2 YERPN	114	86	143
sp Q1CNS4 G6PI YERPN	255	40	253
sp Q1GBJ2 RPOA LACDA	76	81	155
sp Q1I0V1 VP1 MPRVN	171	692	1034
sp Q1J570 GATB STRPF	194	43	242
sp Q1PEW8 FB127 ARATH	106	0	57
sp Q21276 YZVL CAEEL	295	20	171
sp Q21L55 Y1312 SACD2	89	41	179
sp Q256C3 CLPX CHLFF	149	49	223
sp Q27288 OBP2 HELVI	118	0	44
sp Q2FIC3 MNHA1 STAA3	537	19	245
sp Q2JPT2 TRPF SYNJB	74	38	121
sp Q2K4E6 Y3534 RHIEC	149	0	55
sp Q31XD9 RL19 SHIBS	22	51	42
sp Q31ZS3 YCHJ SHIBS	45	22	85
sp Q39837 ALB1 SOYBN	34	17	68
sp Q3B8E9 IFT43 XENLA	72	0	129
sp Q3K0D3 AROD STRA1	89	41	95
sp Q3Z0K4 COBT SHISS	165	21	173

sp Q47LL2 RL15 THEFY	22	28	99
sp Q493M5 CSRA BLOPB	14	27	20
sp Q4FVP3 SSTT PSYA2	290	7	103
sp Q4KKT0 DNAA PSEF5	265	32	216
sp Q58350 Y940 METJA	248	0	70
sp Q5DU25 IQEC2 MOUSE	337	23	1118
sp Q5E218 MURI VIBF1	114	39	109
sp Q5HG77 TKT STAAC	271	54	337
sp Q5KQS4 FETC GLOBR	49	88	187
sp Q5LLT4 TRUB RUEPO	64	75	164
sp Q5M5L0 ARGJ STRT2	131	84	182
sp Q5PBP7 CTAA ANAMM	240	0	101
sp Q5RA17 RRP7A PONAB	102	38	140
sp Q5VYY2 LIPM HUMAN	172	43	208
sp Q5XTY7 RL17 FELCA	66	26	92
sp Q63811 CANB2 MOUSE	89	9	81
sp Q63H76 RS8 BACCZ	30	36	66
sp Q6ANN6 NUON DESPS	355	0	115
sp Q6FPK6 RGI1 CANGA	31	35	91
sp Q6FSP5 CSM2 CANGA	91	32	91
sp Q6GJ03 SARX STAAR	93	5	21
sp Q6GZM8 097R FRG3G	107	0	30
sp Q6LME3 Y3228 PHOPR	36	34	55
sp Q70RT7 CYB PLAMN	226	1	152
sp Q71W03 Y2748 LISMF	71	29	76
sp Q73PM9 RL2 TREDE	3	75	198
sp Q746Q5 RSMG GEOSL	82	42	93
sp Q7N364 END4 PHOLL	107	29	144
sp Q7TX80 THTR2 MYCBO	79	29	212
sp Q7V336 LEUC PROMP	150	64	255
sp Q7VJY0 YIDC HELHP	201	87	303
sp Q7VKF7 RS4 HAEDU	75	28	105
sp Q7VZG1 KCY BORPE	102	24	97
sp Q7YQM2 AFF2 PANTR	189	2	1081
sp Q7Z2W4 ZCCHV HUMAN	145	113	644
sp Q7Z408 CSMD2 HUMAN	13	1151	2323
sp Q82S93 COAX NITEU	93	61	102
sp Q83CV9 DEF1 COXBU	39	51	80
sp Q87E80 RL23 XYLFT	16	32	52
sp Q88AI5 MSRA PSESM	53	28	134
sp Q88QV5 PQQB PSEPK	53	68	182
sp Q8CTB2 METN1 STAES	116	90	135
sp Q8E5P8 GLMS STRA3	228	109	267
sp Q8FQ24 ATPF COREF	162	0	28
sp Q8JTH2 PHOSP ABLVH	105	13	179
sp Q8KE85 Y805 CHLTE	54	20	37

sp Q8LPJ4 AB2E ARATH	186	96	323
sp Q8NVL8 LUKL2 STAAW	27	129	194
sp Q8WJ37 MATK ANEAN	142	73	288
sp Q8WVZ7 RN133 HUMAN	73	65	238
sp Q8WYJ6 SEPT1 HUMAN	151	48	168
sp Q8Y2E1 KPRS RALSO	93	66	157
sp Q8Y4B6 ATP6 LISMO	167	8	63
sp Q8Y9D6 METX LISMO	122	47	199
sp Q8YFN7 RL9 BRUME	57	41	91
sp Q8YIS0 Y373 BRUME	49	0	5
sp Q8ZCC1 HDA YERPE	111	20	108
sp Q8ZGV8 BETI YERPE	146	0	52
sp Q925N2 SFXN2 MOUSE	183	11	128
sp Q96RD1 OR6C1 HUMAN	162	25	125
sp Q99WT9 ESSC STAAN	446	301	732
sp Q9BPB1 O226A CONTE	35	0	37
sp Q9BW60 ELOV1 HUMAN	170	16	93
sp Q9DHP6 VHR2 YLDV	14	84	80
sp Q9FI78 HST ARATH	108	88	237
sp Q9GQ38 MAB21 CAEBR	152	40	172
sp Q9HM28 Y042 THEAC	61	58	86
sp Q9KD76 LEPA BACHD	146	130	333
sp Q9M4C0 RR4 HAPHO	61	28	113
sp Q9MUL9 CYST MESVI	189	14	66
sp Q9N0Z0 CXCR6 CERAT	206	15	122
sp Q9NR97 TLR8 HUMAN	148	119	774
sp Q9PA83 RL1 XYLFA	68	44	120
sp Q9PTU1 DBX1A DANRE	37	4	273
sp Q9RQQ9 DIVL CAUCR	340	169	260
sp Q9U0M8 YPF06 PLAF7	142	24	418
sp Q9UTI7 TYSY SCHPO	160	72	393
sp Q9Y6L6 SO1B1 HUMAN	295	38	358
sp Q9ZKW7 MURJ HELPJ	392	0	68
sp Q9ZPY1 PPOX2 ARATH	40	64	94

Appendix C

Project documentation

C.0.1 Milestones

- MS0 19.02.2013 Project start, Kick-Off meeting
- MS1 13.04.2013 Status meeting with Prof. Joller and Romeo Kienzler
- MS2 07.05.2013 Setup hardware cluster
- MS3 14.05.2013 Start experiments
- MS4 28.05.2013 Hand over for abstract and A0 poster
- MS5 31.05.2013 Project end

C.0.2 Week by week breakdown

week 1 (18.02)
setting up a wiki website and redmine
getting an overview about all the required parts in the documentation

week 2 (25.02)
getting started with the subject of MapReduce
research about the classifier *RaptorX*
overview about *Hadoop* framework, *HDFS* and *Streaming*
phone-call mit Romeo und Rémy Bruggmann, introducing *InterProScan* analysis tool

week 3 (04.03)
reseach about *InterProScan* tool
gaining information about uniprot/swissprot database and how we can get the files
research for FASTA-file format
getting started with documentation
tool *RaptorX* is runnable on CLI with one sequence
setup a virtual cluster with *IBM BigInsights*

week 4 (11.03)
two single node clusters were installed and configured for first step and try out *Hadoop Streaming*
the input file in FASTA-format has two sequences,
but files in Hadoop are read block by block and are splitted line after line,
so input file has to be transformed before loading it into *Hadoop*

week 5 (18.03)

starting with writing bash scripts
meeting with Rémy Bruggmann about possible interpretation of *InterProScan* output files

week 6 (25.03)
bash script which writes every FASTA-sequence in one-line. But it runs extremly slow.
we write down the required steps in our bash script which will be used as mapper function on the *Hadoop Streaming* job

week 7 (01.04)
replace the bash script for one-line FASTA-record by a C++ program
run a *Streaming* job with 15 sequences and getting an output file with the filtered information of *RaptorX* output file
run *InterProScan* on cluster
some of the sequences are difficult to calculate for *RaptorX* and take extremly long
experiments should show overhead of using *Hadoop* vs. native runs and scale out has to be linear

week 8 (08.04)
setting up eclipse with *BigInsights* plugin
writing a FASTA-input file reader is not so simple, because different *Hadoop* versions supports different classes of lineReader

week 9 (15.04)
update documentation according the template we get from Prof. Joller
change properties of *InterproScan* for single threaded runs
RaptorX run on a cluster with more than one node

week 10 (29.04)
InterProScan runs correctly on 1 node cluster
updating documentation
week 11 (06.05)
setting up hardware cluster and distribution of all required files for *RaptorX* and *InterProScan* on all nodes
running tests with bash scripts on a cluster with more than one node

week 12 (13.05)
plan the measurements
review documentation and discusion on the conclusion section at the documentation
restart experiments on cluster, because of incorrect settings on *InterProScan* properties and too many map tasks per node.
try out *RaptorX* runs with additional parameters because of the failing tasks

week 13 (20.05)
analysing experiment-results
updating documentation
started with design of poster and writing the abstract

week 14 (27.05)
completing documentation

Appendix D

CD content

Table (D.1) CD content

file	**description**
./semesterThesisBuechiMathys.pdf	Documentation
./sa-Poster.ppt	A0 Poster
./summary.doc	summary of our project
./Documentation/	LATEX-Documentation-files
./fasta-sequences/	Used input files for experiments
./gnuplot/	gnuplot source files
./scripts/	used bash scripts

List of Figures

List of Tables

Glossary

FQDN
: *FQDN* means fully qualified domain name which represents the system/node in the domain name system hierarchy.

GO database
: *GO* database stores ontology and annotation files which are contributed by the *GO Consortium*. the following url points to the online websearch at the *GO* Database: http://amigo.geneontology.org/cgi-bin/amigo/go.cgi

GO-ID
: Every entry in the *GO* database has an unique identification number. It starts witch the term "GO:" and is followed by a number.

HDFS
: *HDFS* (Hadoop Distributed File System) is a part of the *Hadoop* framework. It is a dristributed file system which is fault tolerant and designed to store big datasets in parts of the different nodes of a cluster.

IPR-number
: Every entry in the *InterProScan* database has a unique identification number. It starts witch the term "IPR:" and is followed by a number.

MapReduce
: *MapReduce* is a pattern introduced by *Google* for calculating huge dataset on parallel systems.

Bibliography

[1] allthingshadoop.com.
Tips, tricks and pointers when setting up your first hadoop cluster to run map reduce jobs.
http://allthingshadoop.com/2010/04/28/map-reduce-tips-tricks-your-first-real-cluster/, 2010.
[Online; accessed 28-May-2013].

[2] Milind Bhandarkar, Suhas Gogate, and Viraj Bhat.
Hadoop performance tuning: a case study.
http://stevereads.com/papers_to_read/hadoop_performance_tuning_a_case_study.pdf.
[Online; accessed 28-May-2013].

[3] J. G. Burrell.
Click4biology - secondary structure.
http://click4biology.info/c4b/7/images/7.5/tertiary.gif, 2002.
[Online; accessed 30-May-2013].

[4] UniProt Consortium.
ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete.
[Online; accessed 30-April-2013].

[5] IBM Cooperation.
http://www-01.ibm.com/software/data/infosphere/biginsights/features.html.
[Online; accessed 28-May-2013].

[6] Jeffrey Dean and Sanjay Ghemawat.
Mapreduce: simplified data processing on large clusters.
51(1):107–113, 2008.

[7] The National Center for Biotechnology Information.
Fasta-format.
http://blast.ncbi.nlm.nih.gov/blastcgihelp.shtml.
[Online; accessed 28-May-2013].

[8] The Apache Software Foundation.
Hadoop Streaming.
http://hadoop.apache.org/docs/stable/streaming.html.
[Online; accessed 27-May-2013].

[9] The Apache Software Foundation.
Hadoop user groups.
http://wiki.apache.org/hadoop/HadoopUserGroups.
[Online; accessed 30-May-2013].

[10] The Apache Software Foundation.
hdfs commands.
http://hadoop.apache.org/docs/r1.1.2/file_system_shell.html.

[Online; accessed 30-April-2013].

[11] The Apache Software Foundation.
Performancetuning hadoop wiki.
http://wiki.apache.org/hadoop/PerformanceTuning.
[Online; accessed 30-May-2013].

[12] The European Bioinformatics Institute.
overview of the different databases used on interpro.
http://www.ebi.ac.uk/training/online/sites/ebi.ac.uk.training.online/files/user/84/documents/figure_interpro.png.
[Online; accessed 30-May-2013].

[13] Shrinivas B. Joshi.
Apache hadoop performance-tuning methodologies and best practices.
In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, pages 241–242, New York, NY, USA, 2012. ACM.

[14] Shonda A. Leonard.
IUPAC/IUB Single-Letter Codes Within Nucleic Acid and Amino Acid Sequences.
John Wiley Sons, Inc., 2002.

[15] Dapeng Li, Tonghua Li, Peisheng Cong, Wenwei Xong, and Jiangming Sun.
A novel structural position-specific scoring matrix for the prediction of protein secondary structures.
Bioinformatics, 2011.

[16] Yahoo! Developer Network.
Apache hadoop: Best practices and anti-patterns.
http://developer.yahoo.com/blogs/hadoop/apache-hadoop-best-practices-anti-patterns-465.html, 2010.
[Online; accessed 28-May-2013].

[17] Owen O'Malley.
Introduction to hadoop, 2008.

[18] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso.
Failure trends in a large disk drive population.
In *Proceedings of the 5th USENIX conference on File and Storage Technologies*, pages 2–2, 2007.

[19] UniProtKB Swiss-Prot release 2013/04.
http://web.expasy.org/docs/relnotes/relstat.html.
[Online; accessed 30-April-2013].

[20] UniProtKB TrEMBL release 2013/04.
http://www.ebi.ac.uk/uniprot/TrEMBLstats.
[Online; accessed 30-April-2013].