User manual for CloudSpeller software

Dieter De Witte
IBCN-iMinds,
Ghent University,
Gaston Crommenlaan 8,
9050 Ghent,
Belgium
Supervisor: Prof. Dr. Jan Fostier

Contact: jan.fostier@intec.ugent.be

February 2014

Contents

1 The package **2**
1.1 Running the Alignment-free discovery Algorithm 3
1.1.1 Input format 3
1.1.2 Settings file 4
1.1.3 Output format 5
1.2 Postprocessing functionalities 6

2 Deploying CloudSpeller on the Amazon Cloud **7**
2.1 Amazon interface 7
2.2 CLI possiblities 7
2.3 Configuring your EMR cluster for CloudSpeller 7
2.3.1 Bootstrapping the AF simulation 8
2.3.2 Bootstrapping the AB simulation 8
2.3.3 Bootstrapping the Filtering simulation 8
2.3.4 KBest 9
2.4 Running CloudSpeller jobs with the AWS graphical webinterface 9

Chapter 1

The package

The CloudSpeller software is written in JAVA. The source code and precompiled JARS (in JDK6) can be downloaded from the website: `http://bioinformatics.intec.ugent.be/blsspeller/index.html`. To build the JARS from the source code the match between the main classes and the JARS is given in the table below.

Table 1.1: Correspondence between JARS and java main files

JAR	Corresponding source code file	functionality
CloudSpeller.jar	driver.FrameworkTest.java	De novo discovery algorithm
DBPM.jar	driver.DistributedPatternMatcher.java	Alignment-free pattern matcher
DBABPM.jar	driver.DistributedABPatternMatcher.java	Alignment-based pattern matcher
Filter.jar	postprocessing_Single.DBMotifFilter.java	Filtering the motif database on C, F and BLS-threshold
BestK.jar	postprocessing_Single.BestKMotifsSelector.java	Filtering the motif database for the top K motifs per permutation group

CloudSpeller.jar provides the main functionality: it's a motif discovery algorithm with the following properties:

- It is word-based: currently only the IUPAC motif model is supported. (a pwm filter is part of our future work)
- It is exhaustive: all possible words which are within a certain search space (defined in the settings file) are tested
- It is comparative: it uses the BLS score as a means to quantify motif conservation between orthologous promoter sequences
- It is distributed: the algorithm can be easily deployed both locally as on a Hadoop cluster (for example Amazon EMR)

The algorithm generates a database of motifs. The database contains for each motif the number of gene families in which it occurs, for a certain BLS-threshold, and a confidence score per BLS-threshold. The confidence scores measures how often a motif occurs, compared to a background model of control

motifs (permutations of the motif). If a motif occurs twice the number of times the confidence scores is 50%, five times more often 80%, ten times more often 90% and so fort.

The motif database allows the user limit the search for potentially interesting motifs. It is a preprocessing algorithm. The user can investigate the database with the Filter and BestK algorithms which will limit the set of results (these filters are very similar to a SELECT query in a relational database). To know the exact motif occurrences (i.e. gene family ID, gene ID, orientation, coordinates w.r.t. transcription start site) the user can use de alignment-free and the alignment-based pattern matcher.

1.1 Running the Alignment-free discovery Algorithm

To run the CloudSpeller software on a local system, the following command should be issued:

```
java -jar CloudSpeller.jar -settings settings.txt input/ output/
```

The software can also be tested on a Single Node hadoop setup. The details for setting up this type of virtual cluster are well destribed at: `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/` To run the software on a hadoop cluster first, the inputdata and the settingsfile need to be copied to the distributed filesystem.

```
bin/hadoop dfs -copyFromLocal sourcePath destinationPath
```

Next the software can be started as follows:

```
bin/hadoop jar CloudSpeller.jar -settings settings.txt input/ output/
```

Note that the output path cannot exist in advance.

1.1.1 Input format

The input format consists of Gene Family (GF) records. Multiple records can be put in the same file, this avoids the overhead of opening and closing many small files. Note that a file ($< 64MB$) will be handled by a single mapper, this implies that the records in one file will be processed by a single slave node, no further splitting is possible. The simulations on the Monocot dataset were run with 10 GF records per file (this is enough to overcome the small file overhead without compromising Hadoop's dynamic load balancing). An example of a GF record:

```
iORTHO012025
((BD1G15540:0.2688,OS03G38480:0.2688):0.0538,(SB01G015830:0.086,ZM01G45340:0.086):0.2366);
4
BD1G15540 BD
ATTAAGTTTGTTACGAATCAACGCGCACTCCACTCCTCTATAGGAATTTTTCATAAAAACACTCGGCTTCTTTTC
OS03G38480 OS
GCATCAAGTATATATATAGTACTTTTGCAGTGCTATCGATCTACCGATGCATGATTACCTGCAAAAATTATCTAA
```

```
SB01G015830 SB
GGCTCTAGAAATGCAAAGCACAGCCCAAGCATAGAGTATTCTTTTGGTTGCTGCTCAAGGACAGACTCAACACCA
ZM01G45340 ZM
ACGTGGCAGAGATCAATTCACGCACCGCATCGCGGCGTGCTTGCTGATTTAGCCCCCTAACATTCCAAGAGAGGA
```

Next list provides an explanation of the diffferent lines in the record:

- Gene Family ID
- Newick tree format of phylogenetic tree containing all genes in the gene family. (note that no branches with length 0 should be added, paralog branch lengths are usually given a small number $1e-6$) Details about the Newick tree format can be found on wikipedia (we used the **popular** convention)
- Number of genes in family
- A number of gene sequence combinations:
  - gene $\mapsto$ geneID and speciesID,
  - sequence is a string with no newlines. For the alignment-based gene families the dashes ('-') and noncapital characters are handled by the algorithm.

1.1.2 Settings file

The settings file contains all the parameters for the CloudSpeller motif discovery run, an example is given below:

```
sMotifAlgorithm_Type=EXACT
sIndex_Structure=GST
sNode_Decoration_Type=BITS
sConservationScore=BLS
sBLS_Thresholds=15,50,60,70,90,95
iKmin=6
iKmax=12
iMax_Degenerate_Positions=3
sMotif_Alphabet=TWOFOLDSANDN
sFilter_Type=SIMULTANEOUS
iFamily_Cutoff=1
iConfidence_Cutoff=50
iBackground_Group_Size=1000
```

The settings file contains the name of the parameter, an equal sign and a value. The parameter name is precede by the parameter type: s for string, i for integer. We'll discuss the possibilities for the different parameters:

- sMotifAlgorithm_Type: can be EXACT (AF discovery), ABEXACT (AB discovery) and FAKE (for benchmarking purposes)
- sIndex_Structure: currently only GSTs (Generalized Suffix Trees) are supported. In principle the algorithm could also run with suffix arrays.

- sNode_Decoration_Type: The information stored in the internal nodes of the suffix tree can be stored as BITS (recommended) or SETS (requires more space and is slower)
- sConservationScore: currently only the BLS is implemented but the interface could be easily implemented to support different conservation metrics
- sBLS_Thresholds: the different thresholds in a comma-separated list.
- iKmin: the minimal motif length
- iKmax: the maximal motif length
- iMax_Degenerate_Positions: the number of positions where IUPAC symbols are allowed
- sMotif_Alphabet: the main choices here are BASEPAIRS, DONTCARES (ACGT and N), TWOFOLDSANDN (IUPAC alphabet without threefold degenerate characters)
- sFilter_Type: Simultaneously filters the output (before streaming to the database) on Confidence and Family cutoff (a confidence chart is only emitted if both cutoffs are satisfied for at least 1 BLS-threshold)
- iFamily_Cutoff: The minimal number of target gene families a motif must have in order for it to be stored in the database
- iConfidence_Cutoff: The minimal confidence score of a motif must have in order for it to be stored in the database

1.1.3 Output format

The algorithm generates 1 output file per reduce task. For the Monocot simulations the number of reduce tasks was 75, resulting in the same number of files. The files contain confidence charts: for each BLS-threshold the number of target families is given and the confidence score for this threhold. A typical output record looks as follows (1 record is a single line):

```
 AAAAAAAAAARW AAARWAAAAAAA 98 86 77 6 4 1
 22.959183673469386 24.41860465116279 29.87012987012987 33.333333333333336
50.0 100.0
```

- AAAAAAAAAARW: permutation group ID of the motif (this is the motif with the characters lexicographically sorted)
- AAARWAAAAAAA: the motif
- 6 integer numbers indicating the number of target gene families of the motif for the six BLS thresholds (15-50-60-70-90-95).
- 6 double numbers corresponding to the confidence score for the six BLS-thresholds.

1.2 Postprocessing functionalities

Filtering the motif database can be done on Confidence cutoff C, Family cutoff F and BLS-cutoff. The three constraints must be satisfied simultaneously for a certain motif-BLS threshold pair.

```
java -jar Filter.jar -C 90 -F 10 -BLS 50 input/ output/
```

Filtering can also be done per permutation group, the -KBest option indicates the maximum number of motifs which are reported per group. The 3 best motifs per permutation group, given a set of constraints.

```
java -jar KBest.jar -C 90 -F 10 -BLS 50 -KBest 3 input/ output/
```

Chapter 2

Deploying CloudSpeller on the Amazon Cloud

2.1 Amazon interface

2.2 CLI possiblities

Interacting with the Amazon web interface is easy but command line tools give the user more power. Therefore we recommend installing to commandline tools: s3cmd and emr cli. s3cmd lets you interact with the Amazon S3 Cloud Storage, this is where all your data goes, emr cli allows to to create MapReduce jobs from the commandline. s3cmd can be downloaded from `http://s3tools.org/s3cmd`, this webpage also explains the possibilities of this tool. The installation instructions for the emr cli can be found on the amazon docs: `http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-cli-install.html`. In the following section we will use the cli to demonstrate the use of CloudSpeller on AWS.

2.3 Configuring your EMR cluster for CloudSpeller

CloudSpeller can be run via the following command:

```
./elastic-mapreduce --create --name "Simulation"
--master-instance-type=m1.large --slave-instance-type m1.xlarge --num-instances 20
--log-uri s3n://dietersawsbucket/logs --enable-debugging
--jar s3n://dietersawsbucket/CloudSpeller.jar
--args "-D,mapred.reduce.tasks=75,-uri,s3n://dietersawsbucket/,-settings,
s3n://dietersawsbucket/settings.txt,s3n://myBucket/input,s3n://myBucket/output"
```

- This creates a named cluster with 1 master node (type m1.large) and 19 slave nodes (type m1.xlarge)
- A path for the simulation logs is provided and debugging is enabled.

- CloudSpeller.jar is the executable
- The number of reduce tasks is recommended to be slightly below 2 times the number of reducers (38 reducers)
- A uri is provided to tell Hadoop that the settings file can be located on amazon S3
- The settingsfile containing the simulation parameters

2.3.1 Bootstrapping the AF simulation

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
--args "-m,mapred.map.child.java.opts=-Xmx1000m,-m,io.sort.mb=600,
-m,mapred.reduce.child.java.opts=-Xmx3200m,
-m,mapred.tasktracker.map.tasks.maximum=7,-m,mapred.tasktracker.reduce.tasks.maximum=2,
-m,mapred.job.reuse.jvm.num.tasks=-1,-m,mapred.job.shuffle.input.buffer.percent=0.05"
```

The reduce function contains a hash map to store all motif permutations. 3200 MB corresponds to storage for $\frac{12!}{3!2!2!2!} \approx 10e+7$ permutations, which is the maximum number for simulations with IUPAC motifs between 6-12 bp and 3 degenerate characters. The two extra parameters allow the jvms to be reused, which limits the overhead and limit the in-memory sorting in the reduce phase since this is already memory intensive. The map functions only require memory to store the suffix trees, but these only require orders of megabytes. 600MB of the map heap is therefore reserved for the in-memory sorting of the map output records.

2.3.2 Bootstrapping the AB simulation

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
 --args "-m,mapred.map.child.java.opts=-Xmx1150m,-m,io.sort.mb=250,
 -m,mapred.reduce.child.java.opts=-Xmx3200m,
 -m,mapred.tasktracker.map.tasks.maximum=6,-m,mapred.tasktracker.reduce.tasks.maximum=2,
 -m,mapred.job.reuse.jvm.num.tasks=-1,-m,mapred.job.shuffle.input.buffer.percent=0.05"
```

The reduce function is identical to that in the AF simulation, so the same requirements hold. Due to the fact that a motif can occur in multiple aligned windows in a certain gene family, the motifs in the AB mapper can only be emitted after scanning the full set of sequences. This requires storing the motif data and therefore the heapsizes of the mapper are mainly used for this purpose. The in-memory sorting in the map phase is also limited to 250 MB.

2.3.3 Bootstrapping the Filtering simulation

An example of a filtering simulation:

```
./elastic-mapreduce --create --name "SimulationName"
--master-instance-type=m1.small --slave-instance-type m1.xlarge --num-instances 2
--log-uri s3n://myBucket/logs --enable-debugging
--jar s3n://myBucket/Filter.jar
--args "-C,70,-F,5,-BLS,70,s3n://myBucket/input,s3n://myBucket/output"
```

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
--args "-m,mapred.map.child.java.opts=-Xmx1000m,
-m,mapred.tasktracker.map.tasks.maximum=8,-m,mapred.job.reuse.jvm.num.tasks=-1"
```

To understand this command we discuss one line at a time:

- This creates a named cluster with 1 master node (type: m1.small) and 1 slave node (type m1.xlarge).
- A path for the simulation logs is provided and debugging is enabled.
- Filter.jar is the executable (full s3 path is required)
- The Filter tool allows 3 cutoff parameters: -C (confidence threshold), -F (minimal number of target gene families), -BLS (minimal BLS-threshold).
- input and output map are required.
- Filter.jar is a map-only script. Each map slot has a maximum heap size of 1000 MB, 8 map slots are available per task node and the task jvm are reused for efficiency reasons.

2.3.4 KBest

```
./elastic-mapreduce --create --name "KBest"
--master-instance-type=m1.small --slave-instance-type m1.xlarge --num-instances 6
--log-uri s3n://myBucket/logs --enable-debugging
--jar s3n://myBucket/BestKMotifsInMapper.jar
--args "-C,50,-F,0,-BLS,50,-KBest,3,s3n://dietersawsbucket/outputFullSim,s3n://dieters
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
--args "-m,mapred.map.child.java.opts=-Xmx1500m,-m,io.sort.mb=300,-m,mapred.reduce.chi
mapred.tasktracker.map.tasks.maximum=7,-m,mapred.tasktracker.reduce.tasks.maximum=7,-m
```

2.4 Running CloudSpeller jobs with the AWS graphical webinterface

The parameters for the different executables can be found in the previous sections. Here we show graphically how to use the webinterface to run simulations on AWS.

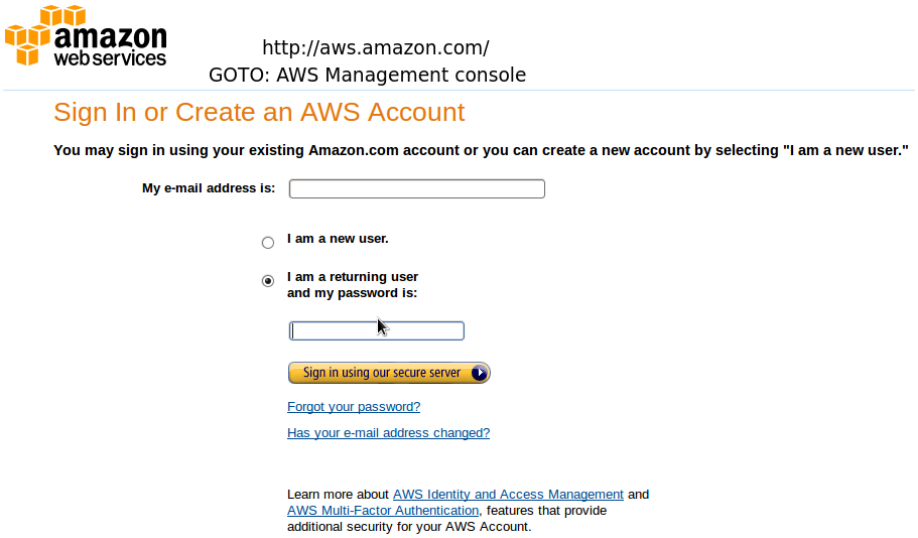

Figure 2.1: After creating an AWS account sign in on `aws.amazon.com` to get acces to the AWS Management Console

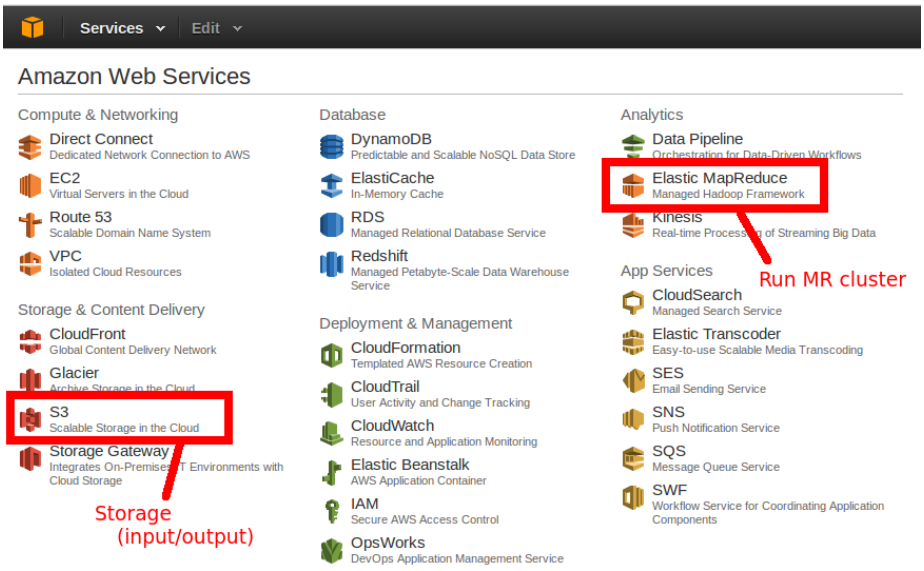

Figure 2.2: CloudSpeller relies on two services: S3 as a storage facility and EMR for running a Hadoop Simulation. In S3 create a bucket and upload the jar, input directory and settings file

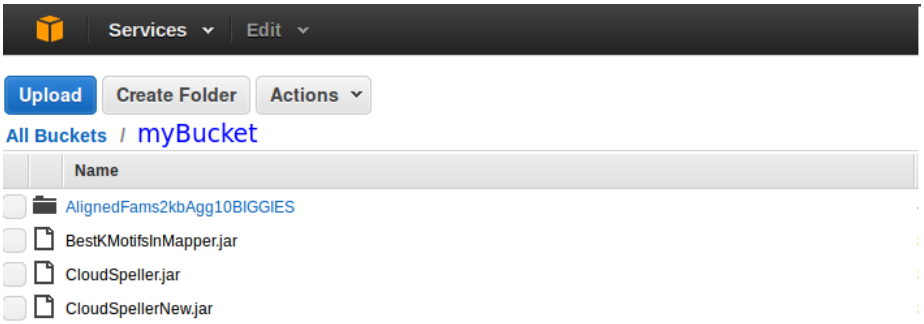

Figure 2.3: A screenshot showing the S3 interface containing a number of files. Use the Upload button (top left) to add more files.

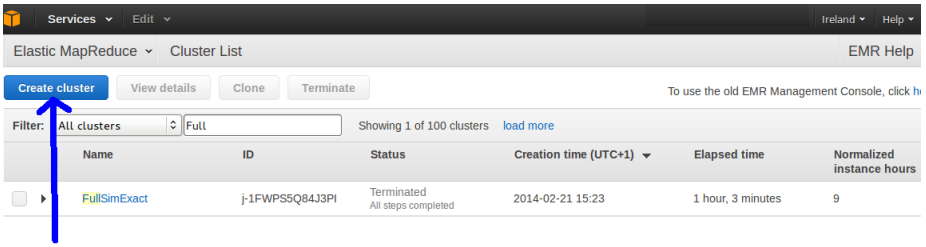

Figure 2.4: A screenshot showing the current EMR interface. Click on create cluster to configure a hadoop cluster to run CloudSpeller

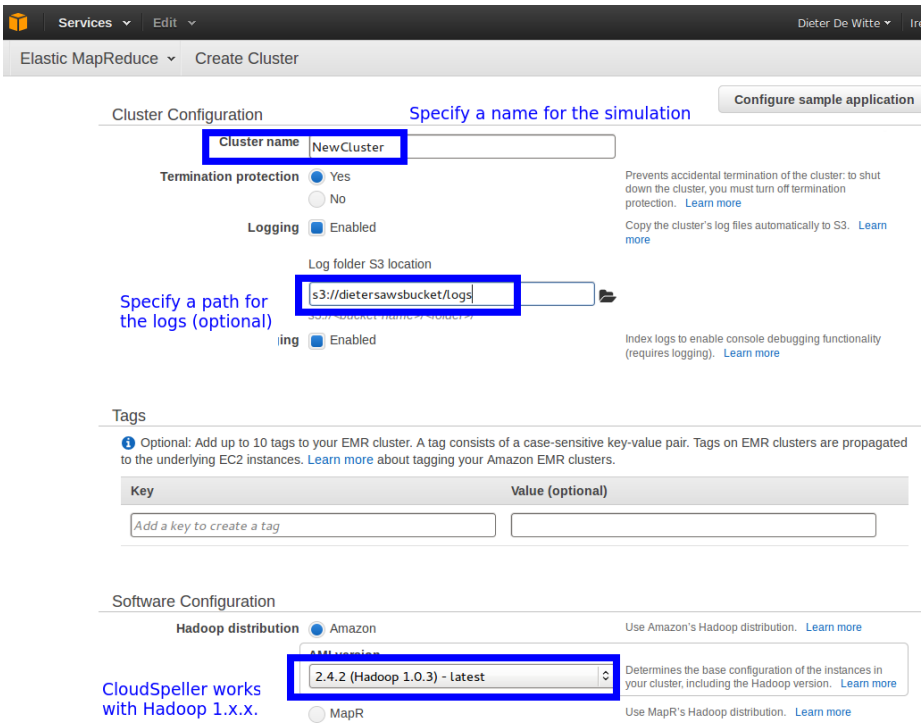

Figure 2.5: When configuring a cluster you should specify a name, a log path, a hadoop version (we use Hadoop 1.x.x). To be continued..

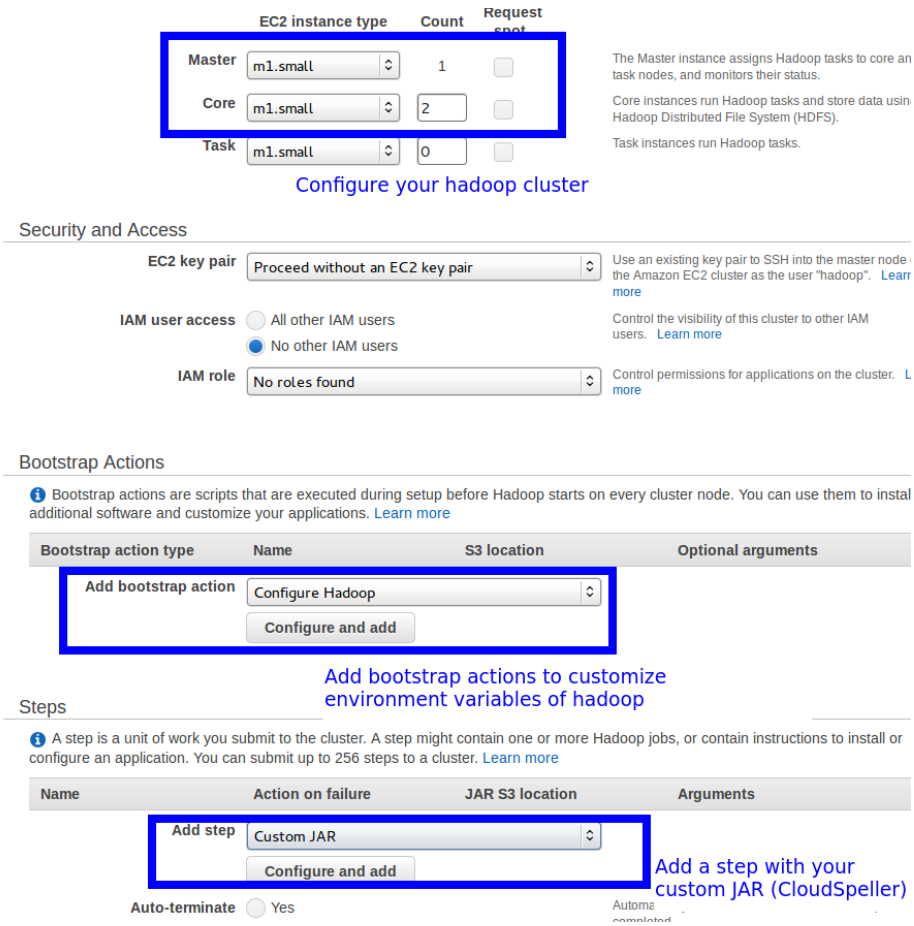

Figure 2.6: Choose the number and type of nodes for your simulation (CloudSpeller was run with $1 + 19$ m1.xlarge nodes). Add a bootstrap action (configure hadoop) with the same parameters as shown in the CLI version (previous section). This is important since the heap sizes of the task jvms need to be set correctly otherwise the simulation might crash.

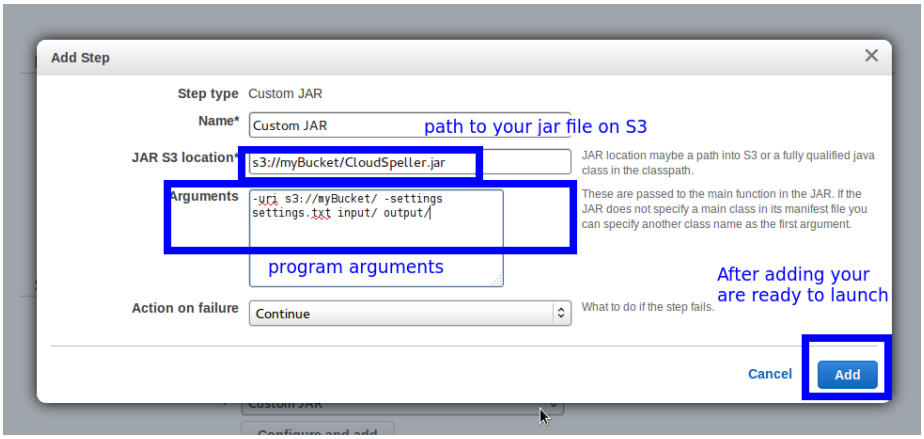

Figure 2.7: Finally add a Custom Jar Step, specifying a path to the jar file and the program arguments of the jar. After adding this step you can start the cluster. The graphical interface will give you information on the simulation progress