Data Intensive Computing – Handout 1

Sun Grid Engine and others

- Since 2001, open-source.
- In 2009 Sum bought by Oracle – Oracle Grid Engine, no longer open-source.
- Two major open-source forks, one of them (Son of Grid Engine) still active.

`qsub`: Submits a job for execution

- `-b [y|n]` binary or a script
- `-cwd` keep current working directory
- `-v variable[=value]` defines or redefines environment variable
- `-V` export all environment variables
- `-N name` set name of the job
- `-o outpath` set file with the standard output of the job; default `$JOB_NAME.o$JOB_ID`
- `-e outpath` set file with the standard error of the job; default `$JOB_NAME.e$JOB_ID`
- `-j [y|n]` merge standard output and error
- `-sync [y|n]` wait till the job finishes
- `-l mem_free=1G` set required amount of memory
- `-l h_vmem=1G` set maximum amount of memory; stop job if exceeded
- `-hold_jid comma_separated_job_list` jobs that must finish before this job starts
- environmental variable `JOB_ID`
- environmental variable `JOB_NAME`

Array jobs:

- `-t 1-n` start array job with n jobs numbered $1 \dots n$
- environmental variable `SGE_TASK_ID` (sometimes `GE_TASK_ID` in different SGE versions)
- output and error files `$JOB_NAME.[eo]$JOB_ID.$TASK_ID`
- `-t m-n[:s]` start array job with jobs $m, m+s, ..., n$
- environmental variables `SGE_TASK_FIRST`, `SGE_TASK_LAST`, `SGE_TASK_STEPSIZE`
- `-tc j` run at most j jobs simultaneously
- `-hold_jid_ad comma_separated_job_list` array jobs that must finish before this job starts; task i depends only on task i of specified jobs

`qstat`: List of running jobs

Detailed information about a job can be obtained using `qstat -j job_id`.

`qdel`: Stops jobs with given ids

`qrsh`: Starts interactive shell – think `ssh`

Dumbo and Hadoop::Streaming Examples

```
def mapper(key, value):
    for word in value.split():
        yield word, 1

def reducer(key, values):
    yield key, sum(values)

if __name__ == "__main__":
    import dumbo
    dumbo.run(mapper, reducer)
```

```
sub map {
  my ($self, $line) = @_;
  my ($key, $value) = split /\t/, $line, 2;
  foreach my $word (split /\s+/, $value) {
    $self->emit($word => 1);
  }
}

sub reduce {
  my ($self, $key, $values) = @_;
  my $count;
  for ($count = 0; $values->has_next(); $count++) {
      $values->next;
  }
  $self->emit( $key => $count );
}
```

Tasks

Assume we have Wikipedia content – pairs *(article name, article content)*. Tasks:

- Find unique article names.
- Find unique words used in articles.
- Count all unique words in the articles.
- Compute inverted index – for every word, compute sorted *(article of occurrence, position of occurrence)* pairs of its occurrences.
  - Ideally the articles would be identified using numeric id.
- Create simple N-gram language model – count number of occurrences of unigrams, bigrams, ..., N-grams. The N-gram language model should be reasonably efficient. One possible algorithm:
  - Compute the unique words of the corpus, filter out the words that have only one occurrence, sort them according to the number of their occurrences and number them from 1.
  - In order to represent N-gram, use the N numbers identifying the words, followed by a 0. Store the numbers using variable-length encoding (smaller numbers take less bytes, e.g. `pack 'w*', @word_numbers, 0` in Perl).
  - One file of the resulting index should contain a sorted list of *(N-gram representation, occurrences)* pairs, where N-gram representation is described above and occurrence is a variable-length encoded number of occurrences. No separators are necessary.
  - Every data file should also be accompanied by an index file, which contains every 1000[1]-th N-gram representation of the data file, together with the byte offset of that N-gram representation in the data file. (The motivation behind the index file is that it will be read into memory and if an N-gram is searched for, it will point to the possible position in the data file.)
  - The N-gram representation in one data file should be all smaller or larger than in another data file.

Design suitable representation and solutions when using both `SGE` and `MapReduce` framework.

[1] You are free to choose better Pivejc constant