Data Intensive Computing – Handout 2

Training Cluster

You can login to the training cluster *dlrc* via machine `ufallab.ms.mff.cuni.cz` and port 11422, i.e., using `ssh -p 11422 ufallab.ms.mff.cuni.cz`.

Exercises for using Sun Grid Engine

Wikipedia Data

In the `/dlrc_share/data/wiki/` there are following data:

- `cs-text/cswiki.txt`: Czech Wikipedia data (Sep 2009), file size 195MB, 124k articles.
- `en-text/enwiki.txt`: English Wikipedia data (Sep 2009), File size 4.9BG, 1.7M articles.

Both files are encoded in UTF-8 and contain one particle per line. Article name is separated by `\t` character from the article content.

Tasks

Solve the following tasks. Solution for each task is a source code processing the Wikipedia source data and producing required results, while utilizing distributed computation. The solution does not need to recover when one of the computation fails, but it should detect that and fail as a whole.

Task	*Points*	*Description*
`unique_words`	2	Create a list of unique words used in the articles. Convert them to lowercase to ignore case. Because the article data is not tokenized, use provided `/dlrc_share/data/wiki/tokenizer/{cs,en}_tokenizer`, which reads untokenized UTF-8 text from standard input and produces tokenized UTF-8 text on standard output. It preserves line breaks and separates tokens on each line by exactly one space.
`inverted_index`	3	Compute inverted index – for every lowercased word from the articles, compute ascending *(article id, ascending positions of occurrences as word indices)* pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: `word \t article_id \t space separated occurrences...` Once again use provided tokenizer.

Task	*Points*	*Description*
`wordsim_index`	3	In order to implement word similarity search, compute for each lemma with at least three occurrences all *contexts* in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the *context* of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached). To compute the lemmas for a given article, use provided `/dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer`, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: `lemma \t context \t counts...`
`wordsim_find`	2	Let S be given natural number. Using the index created in `wordsim_index`, find for each lemma S most similar lemmas. The similarity of two lemmas is computed using *cosine similarity* as $\frac{C_A \cdot C_B}{\lvert C_A \rvert \cdot \lvert C_B \rvert}$, where C_L is a vector of occurrences of lemma L contexts. The output shold be a file with one lemma on a line in the following format: `lemma \t most similar lemma \t cosine similarity...`