Introduction

This report presents results of processing wikipages using XGrid.

Description of the wikipage processing program

First a list of Ids is prepared on the user machine.

Then the program (processIdPage.py) is run on the XGrid. It first gets list of Ids. Then for each Id the program:

- gets the page with that Id
- extracts categories and the 5 most frequent words
- adds results to the data structure (dictionary of dictionaries) which keeps track of categories and counts how many times particular word has been associated with that category
- outputs result

All of the output is aggregated at the client using program aggreage_output.py.

Example execution

First we need to prepare the files. The following command will create 100 files (with file names Ids_part_1.txt, Ids_part_2.txt and so on) of 100 ids in each file.

```
./prepareIds.py 100 100
```

Then run a program to automatically create a batch file. I modified the makeBatchMulti.py so that it creates batch files without any user input required, except for the command line parameters. The parameters are: batch file name; job identifier; how many Id files to add.

```
makeBatchMulti.py 100by100ids.batch dj 100
```

To submit job to XGrid run the following command:

```
time xgrid -job batch 100by100Ids.batch |getXGridOutput.py |
aggregate_output.py > results_100by100Ids.txt
```

I put all that in a file runExperiment.sh. To run 10 tasks, each processing 100 pages (total of 1000 wikipedia pages) run

```
./runExperiment.sh 10 100
```

Measuring execution time

The reported time is the time spent to run the batch job at XGrid. Experiments were performed on Wed, April 7th, around 8-10pm and Thursday April 8th, 5-6pm. On each pair of parameters the experiment is run only once. I calculate the time to process one page and plot that on a 3D scatterplot in Figure 1.

Results

In order to process 1000 pages it is faster to have 10 tasks processing 100 pages each (19 seconds total) than having 100 tasks processing 10 pages each (46 seconds total). And it is faster to have 50 tasks each processing 200 pages (219 seconds), than 20 tasks to process 500 pages (247 seconds total). Thus there is no definite answer of less tasks is better, or more pages per task is better. So let's try to find a balance.

On the graph, if we look at the slice of the graph where number of tasks is 20, it takes more and more time to process one page when number of pages processed per task is increased. This suggests of some kind of serialization. The explanation could be that the IO part takes relatively longer than the processing part of a single page. And since there is only one server, it can process only that much requests. Another possible explanation is that there is more overhead because more data (a file with list of Ids) is passed along with the tasks.

Now if we look at the slice of the graph where number of pages per task is 100, then 20 tasks and 70 tasks perform the best. It is interesting that 70 tasks perform better than running 60 or 80 tasks. (But also note that this is only one run).

Thus it seems that the best way to process the whole wikipedia on the Smith College XGrid system would be 70 tasks, and it would take at least 25.99 hours ($0.01871429 * 5000000/60/60$). This is the lower bound because generally it seems that the processing time of 1 page increases as the number of pages per task is increased.

Figure 1: 3D scatterplot of time to process 1 wikipedia page