Readme

This Readme contains details about 2 scripts. Both the scripts are written in Ruby. They are explained below:

1. Execution and Deployment

a) Installing Ruby

Ruby can be installed in the following ways depending on your OS.

http://www.ruby-lang.org/en/downloads/

1. **Build from Source (for Unix-like machines)**

   This is the usual way where Ruby can be installed on Unix-like machines, by running `make` and `make install`.

2. **Windows 1-click installer**

   There is a 1-click installer available for Windows, which installs Ruby and the documentations and a couple of IDE's for it.

3. **Debian-based Linux**

   Ruby can also be installed using apt-get as

   ```
   sudo apt-get install ruby
   ```

b) Executing the Scripts

The syntax for executing Ruby Scripts is as follows:

```
ruby <filename>
```

Thus, for running the 2 scripts, we can type the following commands

```
ruby log_predict_protein.rb
ruby parse.rb
```

2. Scripts

The Details of the 2 scripts are mentioned below:

a) log_predict_protein.rb

This is the logger script. It periodically(every 10 minutes) checks the status page of the Predict Protein Server and logs the results to a file called "log.txt". A summary of each line of the script follows:

Line No	Summary
1	Imports the net/http library
2	Start of Function log
3	Gets the HTML status page at www.predictprotein.org/status.php
4	Gets the body of the HTML page retrieved
5	Opens the file "log.txt" in append mode
6	Pretty Printing – Writes 2 newlines to the file
7	Uses Regular Expressions to parse the body of the HTML page looking for the term "Predict Protein". This results in finding the line which contains the current timestamp and writing it to the log file.
8	Pretty Printing – Writes a newline to the file
9	Uses Regular Expressions to parse the body of the HTML page looking for the term "Processes". This results in an array containing the status of 3 kinds – Processes running, Processes waiting in the queue and Processes waiting in error mode. This array is output to the log file – each element on a new line.
10	Closes the file
11	End of Function log
12	Start of while-true loop
13	Start of "try" block (called "begin" in Ruby)
14	Calls function log
15	Sleeps for 10 minutes
16	Start of "catch" block (called "rescue" in Ruby)
17	Opens the file "log.txt" in append mode
18	Writes "Timeout" along with current timestamp to the file
19	Closes the file
20	Sleeps for 1 minute
21	Calls retry which retries the execution of the entire while-true loop
22	End of "catch" block
23	End of while-true loop

Table 1: Code Summary - log_predict_protein.rb

b) parse.rb

This is the parsing script. It goes through the log file and creates 2 CSV files called “running.txt” and “queue.txt”. The “running.txt” file will contain entries consisting of <timestamp, number of jobs running> separated by a newline. The timestamp will always start at 1, and increment. Similarly, “queue.txt” will contain entries consisting of <timestamp, number of jobs on queue>. A summary of each line of the script follows:

Line No	Summary
1	Defines a variable rcount and sets it to 1
2	Defines a variable qcount and sets it to 1
3	Opens the file “running.txt” in write mode
4	Opens the file “queue.txt” in write mode
5	Opens the file “log.txt” in read mode
6	Iterates over the file line-by-line
7	Uses Regular Expressions to check if the line contains the term “running”
8	Writes the value of rcount followed by a comma and followed by the actual number of jobs running and a newline to “running.txt”
9	Increments rcount
10	Uses Regular Expressions to check if the line contains the term “queue”
11	Writes the value of qcount followed by a comma and followed by the actual number of jobs on queue and a newline to “queue.txt”
12	Increments qcount
13	End of “if”
14	End of Iterator
15	Closes the file “log.txt”
16	Closes the file “running.txt”
17	Closes the file “queue.txt”

Table 2: Code Summary - parse.rb