Fall 2016 CSCI B534/E599

Project #2-A: MPJ PageRank

1. Deliverables

You are required to turn in the following items in a zip file (**groupid_Project2A.zip**) in this assignment:

1) The source code of parallel PageRank you implemented.
2) The executable class file and the README file that describe its usage.
3) The output file (**groupid_pagerank_output.txt**) which contains the top 10 ranking URL numbers.

Points will be reduced (maximum 0.5 points) if the filename or directory structure is different from instructed above.

2. Evaluation

The total points for Project #2A is 10, where the distribution is as follows:

a. Completeness of your code and output (9 points)
b. Readability and clarity of README.txt (1 points)

3. Introduction

As a follow-up of Project #1, you will implement a parallel version of PageRank using MPI programming interface. We provide a pseudocode for MPI PageRank, and you will complete the implementation based on the pseudocode to make a working version. You will also receive functions for file I/O so that you can focus on main implementation only. For more details on the PageRank algorithm, refer to the previous descriptions for Project #1.

We expect you to implement the program using Java. There is a Java library called MPJExpress that offers the MPI functionalities. You should use this library for the implementation.

Parallel PageRank

Developing parallel PageRank is an active research area for both industry and academia, and numerous algorithms have been proposed. The key idea is to partition PageRank problems into N sub-problems so that N processes solve each sub-problem concurrently. One of the simplest approaches in partitioning is a vertex-centric approach: the graph of PageRank can be divided into groups of vertices and each group will be handled by a process. We take this approach for our MPI PageRank implementation.

4. Compile guide

Fall 2016 CSCI B534/E599

Set up environment

Before compiling, you need to download and set up MPJExpress in your machine. You will use silo (silo.soic.indiana.edu) with your IU account to run your code.
Login to silo using SSH command:

```
ssh username@silo.soic.indiana.edu
```

Download MPJExpress from http://mpj-express.org/ or you can follow the following command:

```
wget https://sourceforge.net/projects/mpjexpress/files/releases/mpj-v0_38.tar.gz/download
mv download mpj-v0_38.tar.gz
tar -zxvf mpj-v0_38.tar.gz
```

You will get a folder named mpj-v0_38. Open the ".bashrc" file:

```
vim ~/.bashrc
```

Then type the following commands at the top of the ".bashrc" file.

```
export MPJ_HOME=/path-to-mpj-unzipped-folder/
export PATH=$PATH:$MPJ_HOME/bin
```

From here type:

```
source ~/.bashrc
```

Compile

After the environment variables are set you can compile the Java program by using the following command:

```
javac -cp .:$MPJ_HOME/lib/mpj.jar MPIPageRank.java
```

Execution

Single Machine

To execute the program with MPI, use the following command in the homework directory:

```
mpjrun.sh -np 8 MPIPageRank <input_file_name> <output_file_name>    <damping factor> <iterations>
```

For example:

```
mpjrun.sh -np 8 MPIPageRank pagerank.input.1000.0 output.txt .85 10
```

-np means the number of processes. In the above case you are creating 8 processes and executing the algorithm in parallel, but this all happens in a single machine.

Fall 2016 CSCI B534/E599

Run MPJ with multi-node mode

There are 24 Linux machines you may use. They are lh008linux-01 to lh008linux-24. You can access lh008linux-01 by lh008linux-01.soic.indiana.edu, lh008linux-02 by lh008linux-02.soic.indiana.edu, etc. If you configured silo as above, then you don't need to do it again in these machines.

Login to lh008linux-01:

```
ssh username@lh008linux-01.soic.indiana.edu
```

Before running on multi-node mode, you will have to create a "machines" file, which has the worker hostname written line by line, e.g. lh008linux-01.soic.indiana.edu and lh008linux-02.soic.indiana.edu. The file name must be "machines".

```
[username@lh008linux-01~]$ cat machines
lh008linux-01
lh008linux-02
```

Once you have set up the "machines" file, you can use mpjboot to start MPJ daemons on worker nodes. Here, an IU passphrase may be required as you are starting the MPJ daemons remotely.

```
mpjboot machines
```

Then you can run MPJ PageRank by adding "-dev niodev" parameters to start your application across worker nodes.

```
mpjrun.sh -dev niodev -np 8 MPIPageRank pagerank.input.1000.0 pagerank.out .85 30
```

After you are done experimenting, execute the following command to clean up the resources.

```
mpjhalt machines
```