CS 395T	Scalable Machine Learning	Fall 2014

Homework 5

Lecturer: Inderjit Dhillon — Date Due: Oct 13, 2014

Keywords: *Galois, PageRank*

Note:

1. The intent of this homework is to familiarize yourselves with the Galois programming model.
2. You will work with the same dataset as the previous homework, `hw4-data.mat`.
3. You can download Galois source from `http://iss.ices.utexas.edu/?p=projects/galois/download`.
4. Submit your code for Problems 2 and 3 through Canvas. Submit a hard copy of your answers in class.

Problem 0: Page Count

$(-\max(0, c-2)$ pt) where c is the total number of pages in your turned-in solutions. Your solutions to this homework cannot exceed 2 pages (assuming your solutions use similar font size as this pdf) unless you want negative scores for this problem. Note that you will turn in code directly on Canvas, and therefore code is excluded from the page count.

Problem 1: Galois Installation

Download the Galois package from `http://iss.ices.utexas.edu/?p=projects/galois` and install it on Stampede.

(a) (2 pt) Follow the instructions (build/readme.txt and build/installing.txt) in the tar-ball or the documentation in the website to compile Galois. Write down all the commands you need to compile Galois. (Hint: You might need to load some modules in stampede to compile successfully). You first few commands will be

```
$ tar xvzf Galois-2.2.1.tar.gz;
$ cd Galois-2.2.1/build;
$ mkdir default; cd default
```

(b) (0 pt) Convert the graph in `hw4-data.mat` into the Galois graph format (.gr) by the command-line graph converter provided by Galois `build/default/tools/graph-convert/graph-convert`.

Problem 2: Sparse Matrix-Vector Multiplication

In this problem, you will implement sparse matrix-vector multiplication and parallelize it on Galois. You will use the Graph data structure in Galois to store the sparse matrix.

(a) (6 pt) Implement the function `y = multiply(A, x)` which performs parallel sparse matrix-vector multiplication.

(b) (4 pt) Using the sparse matrix A in `hw4-data.mat` and random vectors $\mathbf{x}$, report (averaged) running times for 1, 4, 8 and 16 threads.

Problem 3: PageRank

In this problem, you will implement the PageRank algorithm and parallelize it on Galois. The PageRank algorithm is widely-used method for computing the relative rank of web pages based on the Web link structure. The PageRank model involves a directed graph $A \in \mathbb{R}^{n \times n}$ of hyperlinks (edges) between web pages (nodes), a teleportation coefficient α, and a *personalization* vector $\boldsymbol{v}$ (in this problem we use $\boldsymbol{v} = \mathbf{1}/n$). PageRank is computed as the principal (left) eigenvector of a Markov chain probability transition matrix $P = (1-\alpha)D^{-1}A + \alpha \mathbf{1}\boldsymbol{v}^T$ (where D is the diagonal matrix such that $D_{ii} = \sum_j A_{ij}$ and $\mathbf{1}$ denotes the vector of all ones) using a simple power iteration algorithm (like you did in Homework 4).

1. Init $r_i^{(0)} \leftarrow 1/n$, for $i = 1, 2, \ldots, n$.
2. For $t = 1, 2, \ldots, T$
   (a) Update $\boldsymbol{r}^{(t+1)} \leftarrow P^T \boldsymbol{r}^{(t)}$.

(a) (1 pt) Show that $P^T \boldsymbol{r}$ can be computed in $O(nnz(A) + n)$ even though P is dense.

(b) (5 pt) Use the sparse matrix-vector multiplication function developed in Problem 2 and implement the above algorithm in Galois. Using $\alpha = 0.15$ and $T = 50$, run your code for the (undirected) network A in `hw4-data.mat` and report the running time for 16 threads. Compare the time with the corresponding Matlab implementation (with the same T and α).

(c) (2 pt) List the top 10 nodes of A with the highest PageRank values.