Distributed PageRank - First Report

Aaron Myers, Megan Ruthven

November 17, 2014

Contents

1 Introduction

This purpose of this project is to investigate distributed PageRank and attempt to find alternate approaches to PageRank that might offer some additional value in some form (faster, easier, fewer iterations, etc). The goal of this project for us is two-fold.

1. Apply ADMM [3] to the linear problem and determine if there is value in taking an easy-to-parallelize approach rather than complex methods.
2. Approach the problem with the typical power iteration, treating the matrix as a graph and attempting to do the appropriate load balancing and work list updates to allow for faster convergence or fewer iterations. Our idea is maintain a worklist that contains only nodes (and connected nodes) whose updated value is above a set threshold, we will refer to this as "delta updating".

2 Linear System Approach

This approach requires that we form the PageRank problem into a linear system (Ax=b) in which case solving for x would provide the PageRank. Below is a simple derivation taken from [2].

$$P' = P + dv^T \tag{1}$$
$$P'' = cP' + (1-c)ev^T \tag{2}$$
$$x^{k+1} = P''^T x^k \tag{3}$$

Where P$'$ and P$''$ modified PageRank matrix to create a connected graph and add a personalization factor and equation 3 is the simple Power Iteration. (e is the vector of all 1).
Given the additional information below, we can derive the linear system for finding the principal eigenvector.

$$d^T x = \|x\| - \|P^T x\| \tag{4}$$
$$x = [cP^T + c(vd^T) + (1-c)ve^T]x \tag{5}$$

We then have the resulting equation:

$$(I - cP^T)x = kv \tag{6}$$

We now have the principle eigenvector solver in the from of Ax = b, a linear system; where A = I-cP^T and kv = b

2.1 ADMM

Most of the articles we encountered for parallel pagerank used Jacobi iteration or some Krylov Subspace method (GMRES, BiCGSTAB, etc), but we attempted to implement something we were introduce to in this course, namely ADMM [3]. This is an extremely simple way to parallelize a linear solve and we will compare these results to GMRES and BiCGSTAB for the same problem parallelizing using PETSc. We expect ADMM to have worse performance, but we would like to quantify the loss in accuracy/time relatvie to the ease of implementation.

Below is a brief description of the ADMM idea and algorithm.
We take the linear problem and split up the data accordingly:

$$A = [A_1 ... A_n]' \tag{7}$$

$$b = [b_1 ... b_n]' \tag{8}$$

Our origninal minimization of Ax-b with a certain norm and regulariztion on x now becomes:

$$minimize \quad \sum_{i=1}^{N} l_i(A_i x_i - b_i) + r(z) \tag{9}$$

$$subject \;\; to \;\; x_i - z = 0 \;\; \forall i \tag{10}$$

Where x_i are local variables that we force to match the global solution z at each step.
The resulting algorithm, using the augmented lagrangian presented in the ADMM method [3], is as follows:

Algorithm 1 ADMM Iteration

1: $x_i^{k+1} = argmin_x \;\; l_i(A_i x_i - b_i) + \frac{\rho}{2}\|x_i^k - z^k - u_i^k\|_2^2$
2: $z^{k+1} = argmin_z \;\; r(z) + \frac{N\rho}{x}\|z^k - \bar{x}^{k+1} - \bar{u}^k\|_2^2$
3: $u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1}$

Where $u_i^k = \frac{1}{\rho} y_i^k$

2.2 ADMM Results Compared to other Linear methods

Below are the inital results for ADMM programmed in Matlab for a simple data set (a disconnected synthetic 11 node graph).

Table 1: Linear Method Table

5 iter results	
Method	$\|\hat{x} - x\|$
GMRES	0.0047
BiCGSTAB	0.0056
ADMM	$3.4e^{-17}$

3 Power Iteration Approach

In addition to the Linear system, we will approach Distributed PageRank as a Graph with power iteration. We have slightly modified the approach (inspired by [4]) by first computing all updated pagerank values and for every subsequent update, we only modify the pagerank of the nodes pointed to by an update change in value above some certain threshold. We will also use the magnitude of these changes to prioritize a worklist for the algorithm to execute.

Algorithm 2 Power Iteration with Worklist

```
Initialize x, δ (threshold)
Compute Px for all nodes
while Worklist is not empty do
  if x_i in worklist then
    take x_i off the worklist
    x_i^new = (1 − α) ∗ P_i ∗ x + α/#[x]
    if |x_i^new − x_i| > δ then
      x_i = x_i^new
      add x_j onto worklist : ∀x_i → x_j
    end if
  end if
end while
```

3.1 Power Graph Results

We implemented the two algorithms of pagerank with openMP, and compared the time to convergence. On the A dataset from HW4, the pagerank values converged in fewer iterations (25 to 20) and on average, takes less time to run one iteration (0.189 to 0.095 seconds) running on 16 threads. This resulted in a total calculation time of 4.72 seconds for the baseline power iterations and 1.90 seconds for the delta method. A summary table is below.

Table 2: Power Iteration - A

Method Comparison		
Method	Iteration Count	Time(s)
Power Iteration	25	4.72
Delta Update	20	1.90

Table 3: Power Iteration - Friendster

Method Comparison		
Method	Iteration Count	Time(s)
Power Iteration	23	49.6
Delta Update	30	19.5

4 Next Steps

Now that we have seen the value of both ADMM and data-driven PageRank, our next steps will involve parallelizing using MPI

1. Parallelize the ADMM method using MPI and compare the results to running GMRES and BiCGSTAB with PETSc
2. Parallelize the data-driven PageRank problem using MPI and compare these results to other parallel Power Iteration approaches
3. Collect all results and provide conclusions about all methods and the value that each provides.

4.1 Load Balancing

We will also implement load balacing with both ADMM and the delta updating as described below.

1. ADMM: each iteration, we will determine if the new local variable value has changed significantly. If so, it will be pushed to the master worklist to be operated on for the next iteration. If not, it will be removed from the worklist.
2. Delta Updating: similarly, each iteration will check the updated value of the node and push it and its out-neighbors to the master worklist if above a certain threshold. This master worklist will have duplicated removed and will divide the work evenly across all computing nodes. Ideally, the more connected nodes would be sent to the same compute node so some type of clustering may be beneficial for this operation.

References

[1] David Gleich, et al. *Scalable Computing for Power Law Graphs: Experience with Parallel PageRank*

[2] David Gleich, et al. *Fast Parallel PageRank: A Linear System Approach*

[3] Stephen Boyd, et al. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*

[4] Joyce *Presentation on Parallel PageRank*