Distributed PageRank

Aaron Myers, Megan Ruthven

October 18, 2014

1 Abstract

1.1 Problem Description

We will focus on developing and testing iterative methods for distributed page rank. Traditionally, PageRank has been computed as the principal eigenvector of a Markov chain probability transition matrix using a simple power iteration algorithm. We can think of computing PageRank as solving a system of linear equations, and apply iterative solvers to obtain the solution. Parallelization is an important factor in choosing a particular iterative linear solver. We will attempt to develop a distributed algorithm for computing PageRank (both global and personalized) of very large web graphs using MPI. Load balancing is important factor in designing our distributed algorithm. We will consider many strategies, for example, distributing load based on the topology of the graph.

1.2 Proposed Initial Strategies

1. **Krylov Subspace Methods**
   In the typical power iteration, we compute Pb, $P^2b, P^3b,...$ where b is a random vector, ultimately converging on the principle eigenvector. However, much computation is wasted by using only the final result $A^{n-1}b$. We will attempt to use the Kyrlov Matrix:

$$K_n = [b \ \ Ab \ \ A^2b \ \ A^3b...A^{n-1}b] \tag{1}$$

   The columns are not orthogonal, but we can extract and orthogonal basis and use this to find an approximation to the eigenvectors and understand how 'far apart' the eigvectors are so we can estimate the iterations needed to converge.

2. **ADMM**
   Here we proposed applying ADMM as a way to parallelize some iterative method (possible the power iteration) to converge onto a solution faster than simply parallelizing the matrix multiply.

3. **Low Rank Approximation to P**
   To simplify the matrix multiply, we will attempt some form of low rank approximation for P. There are many methods proposed here, Frequent Directions, Sampling, Random SVD, Hashing, Random projection, etc. If we could simplify the matrix vector multiply in the power iteration and combine with some form of ADMM, we may be able to significantly reduce computation time.

4. **Other Linear sovlers**
   We will also apply other linear methods to test performance, namely, GM-RES, Jacobi, Block Jacobi. Along with this, we will attempt to distribute computing load based on some measure of 'distance' between nodes; we may be able to apply some type of clustering here or create a tree out of the data.