Scalable Computing for Power Law Graphs: Experience with Parallel PageRank

David Gleich*
Stanford University, ICME

Leonid Zhukov
Yahoo!

Abstract

In this paper we report on the behavior of parallel numerical algorithms applied to computing the PageRank vector for a 1.5 billion node directed webgraph. We also describe the details of our parallel implementation that was able to compute the PageRank vector for this webgraph on a distributed memory 140 processor RLX cluster in under 6 minutes.

Historically, much of the work in parallel computing has been focused on the numerical solutions of problems arising from the engineering and physical sciences. The graphs used in these problems are either regular, e.g. a grid, or have slight variance in the number of nearest neighbors. Most of the widely used parallel computing toolkits are devoted to solving such problems and follow this paradigm.

Numerical problems arising in data mining and information retrieval have quite different properties. First, the data is already discrete and there is no continuous representation possible. Second, there is no low-dimensional space easily associated with the data. The third and most important difference from standard scientific computing datasets is the presence of a power law distribution in the graph. A power law graph is defined by the property that the number of vertices with degree k is proportional to $k^{-\beta}$ for the power law exponent β. In such a graph there are a few vertices with high degrees and many vertices with low degrees. These properties yield unusual graphs compared to the regularity of finite difference or finite element graphs.

We will focus on webgraphs and a parallel PageRank computation. The PageRank algorithm is a method for computing the relative rank of web pages based on the Web link structure. The model involves a directed graph of hyperlinks (edges) between web pages (nodes), a teleportation coefficient c, and a teleportation vector v. One interpretation of a page's PageRank is the probability of finding a random surfer on that page when the surfer randomly follows links between pages and restarts at a page in v with probability $(1-c)v_i$. PageRank computations are a key component of modern Web search ranking systems.

Traditionally, PageRank has been computed as the principle eigenvector of a Markov chain probability transition matrix using a simple power iteration algorithm. We consider the PageRank linear system formulation and iterative methods for its solution. There are two requirements for the iterative linear solver: i) it should work with nonsymmetric matrices and ii) it should be parallelizable. Because the matrix is strongly diagonally dominant, we consider stationary Jacobi iterations as well as several Krylov subspace methods.

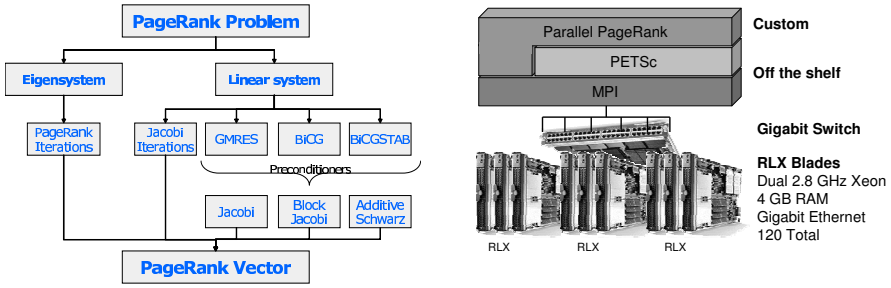
Our parallel computer was a cluster of RLX blades connected in a fully connected topology with gigabit ethernet. We had twelve chassis composed of 10 dual processor Intel Xeon blades with 4 GB of memory each (240 processors, and 480 GB memory total). The parallel PageRank codes use the Portable, Extensible Toolkit for Scientific Computation (PETSc) to implement basic linear algebra operations and basic iterative procedures on parallel sparse matrices.

In our experiments we used seven Web related directed graphs. The av graph, the Alta Vista 2003 web crawl, has 1.4B nodes and 6.6B edges. We also constructed and used three subsets of the av graph (edu, yahoo-2, yahoo-3) and a Host graph (70M nodes, 1B edges), obtained from a Yahoo! crawl by linking hosts through agglomerated links between pages on separate hosts.

In summary, the main results of our experiments show that:

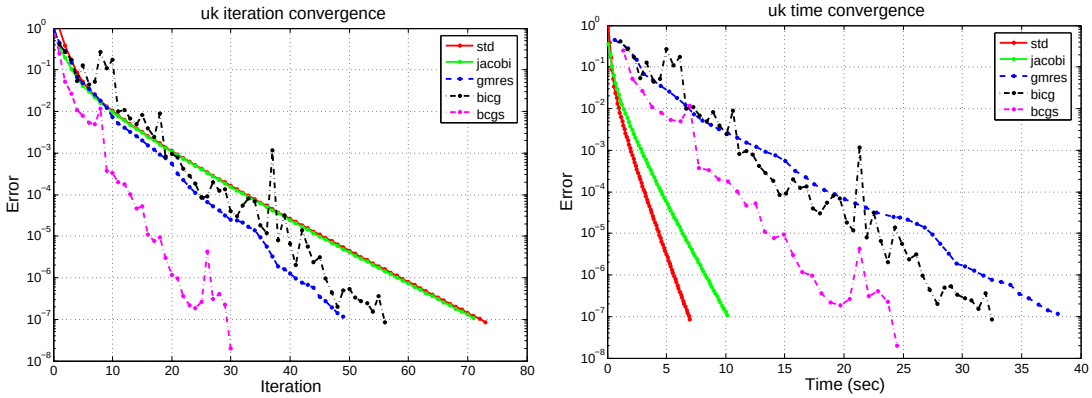
- The power and Jacobi methods have approximately the same behavior.
- The convergence of Krylov methods strongly depends on the graph and is non-monotonic.
- Although the Krylov methods have the highest average convergence rate and fastest convergence by number of iterations, on some graphs, the actual run time is longer than simple power iterations.
- BiCGSTAB and GMRES have the highest rate of convergence and converge in the smallest number of iterations. and GMRES demonstrates more stable behavior.
- The best method to use is either power iterations or BiCGSTAB. The final choice of method is dependent on the time of a parallel matrix-vector multiply compared with the time of the extra work performed in the BiCGSTAB algorithm.
- The BiCGSTAB algorithm scales better than power iterations due to the parallelism in the extra work performed.

Using our parallel implementation, we reduced the time to compute a PageRank vector on a full webgraph from 10 hours to 5.5 minutes.



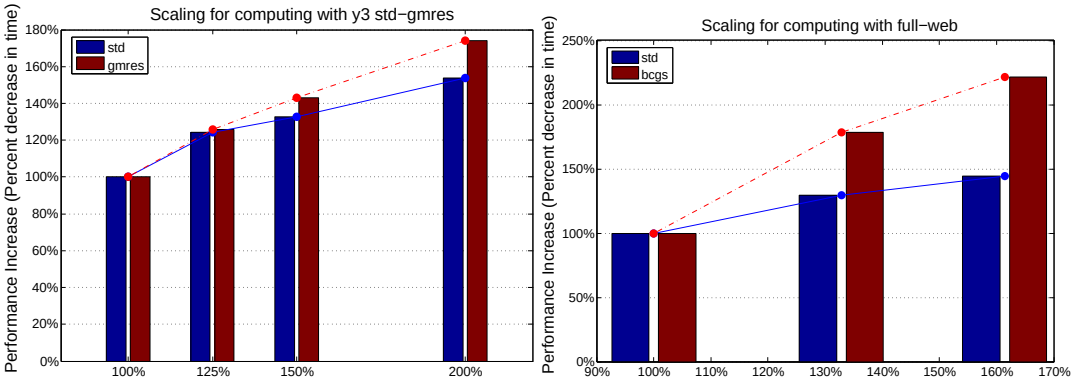
(a) Computational methods (b) Block structure

Figure 1: Our Parallel PageRank System.



(a) Convergence Iterations (b) Convergence Time

Figure 2: Convergence of iterative methods on the uk Web graph.



(a) Power iterations and GMRES on y3. (b) Power iterations and BiCGSTAB on av.

Figure 3: Parallel scaling performance of our algorithms. In both cases, we see that the KSP methods, BiCGSTAB (bcgs) and GMRES, scale better than power iterations (std).