Yahoo! Research Labs

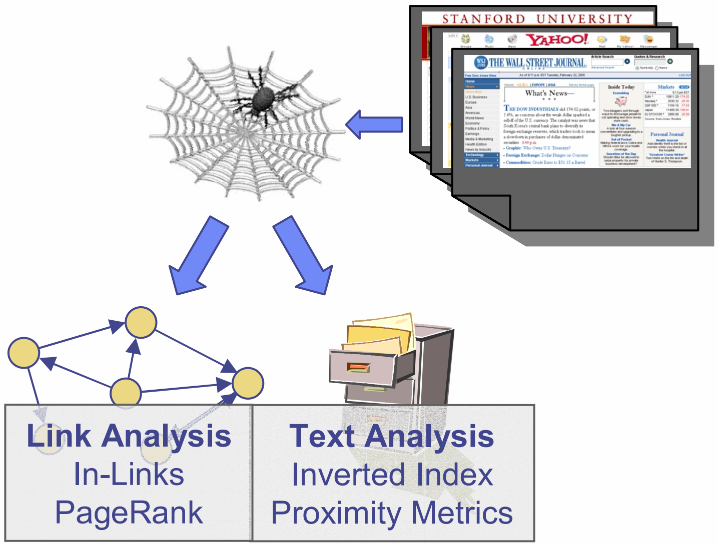
Fast Parallel PageRank

David Gleich

Leonid Zhukov

Pavel Berkhin

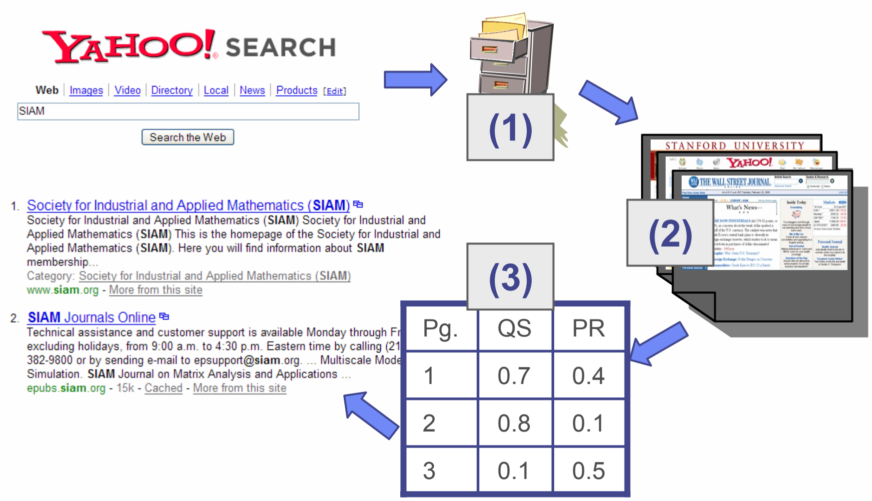
Websearch Engines



At the indexing stage, a web-crawler traverses links between web pages and builds a text database and link database for all pages on the web.

We can do off-line analysis of these databases to build an inverted index, which returns pages that contain a a word, and global link scores like Pagerank.

At search time, (1) we first look up all pages that contain the query word in the inverted index. Then (2) compute a query similarity score for each page and lookup the PageRank score (as well as other features). Finally, we (3) sort the pages and return the results.



Pg.	QS	PR
1	0.7	0.4
2	0.8	0.1
3	0.1	0.5

Websearch

Parallel Motivation

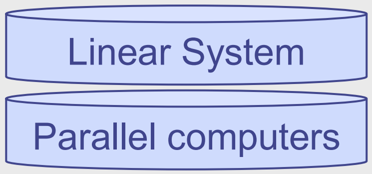
The datasets we have are huge and span much more storage than is possible on a single machine.

We hope to store the matrices in parallel to accelerate the computations.

Name	# Nodes	# Links	Storage
edu	2M	14M	176MB
yahoo-r2	14M	266M	3.25GB
uk	18.5M	300M	3.67GB
yahoo-r3	60M	850M	10.4GB
db	70M	1B	12.3GB
av	1.4B	6.6B	80GB

Our Approach

- Graph in memory.
- Vast computational power.
- Efficient numerical methods.

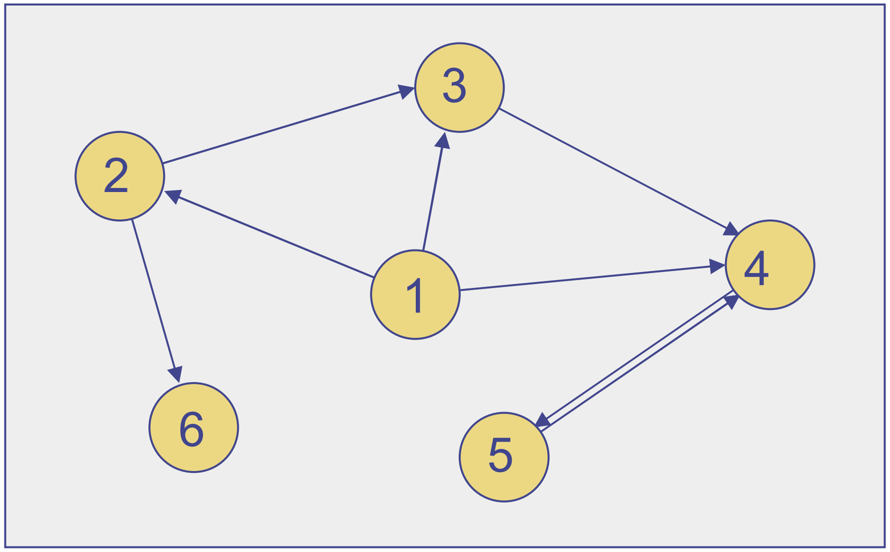


The PageRank Vector

Question: If someone is randomly surfing the web, what is the probability that they will be on a certain page?

Answer: It's PageRank!

How: Convert the web-graph into a Markov chain modeling a random surfer.



YAHOO! Websearch

Deriving the PageRank Equation

1. Normalize out links.

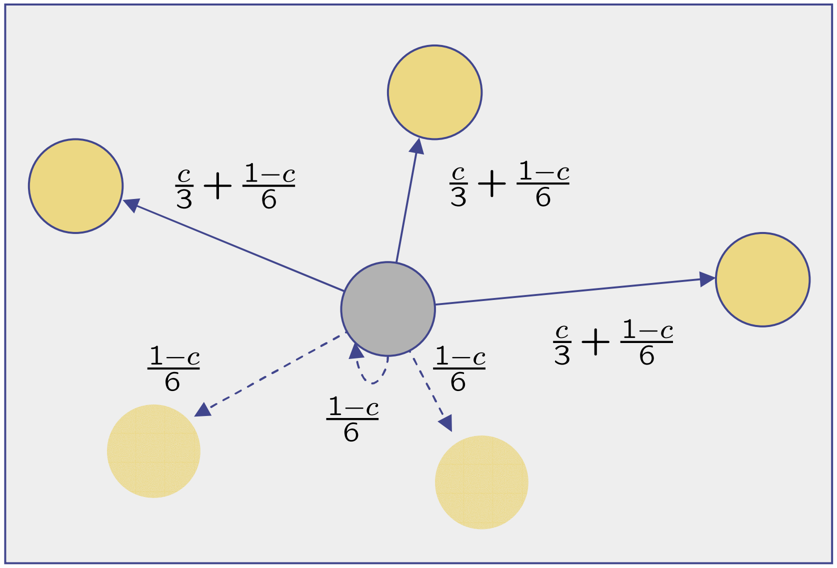
$$P = D^{-1}A$$

2. Fix dangling nodes.

$$P' = P + dv^T$$

3. Add random moves.

$$P'' = cP' + (1-c)ev^T$$



A – adjacency matrix
D – out-degree matrix
d – dangling node indicator
v – personalization vector
c – teleportation coefficient

After these changes the matrix is row-stochastic and irreducible ⇒

a. A unique stationary distribution exists.
b. Power iterations will converge to it.

PageRank Formulations

PageRank is a stationary distribution of a Markov Chain.

Eigensystem

$$P''^T p = \lambda p$$
$$\lambda = 1$$

$$P'' = cP + c(dv^T) + (1-c)(ev^T)$$

Linear system

$$(I - cP^T)x = kv$$
$$p = \frac{x}{||x||}$$

$$k = k(x) = ||x|| - c||P^T x||$$

Simple Stationary Iterations

- PageRank iterations

$$p^{(k+1)} = cP^T p^{(k)} + (1 - c||P^T p^{(k)}||_1)v$$

- Linear system – Jacobi iterations

$$p^{(k+1)} = cP^T p^{(k)} + kv$$

- Iteration Error

$$e^{(k)} = ||x^{(k)} - x^{(k-1)}||_1$$

$$r^{(k)} = ||b - Ax^{(k)}||_1$$

- Converges in k steps

$$k \sim \log(e^{(k)}) / \log c$$

Krylov Subspace Methods (KSP)

Consider a linear system

$$Ax = b$$

and residual

$$r = b - Ax$$

then the Krylov Subspace is

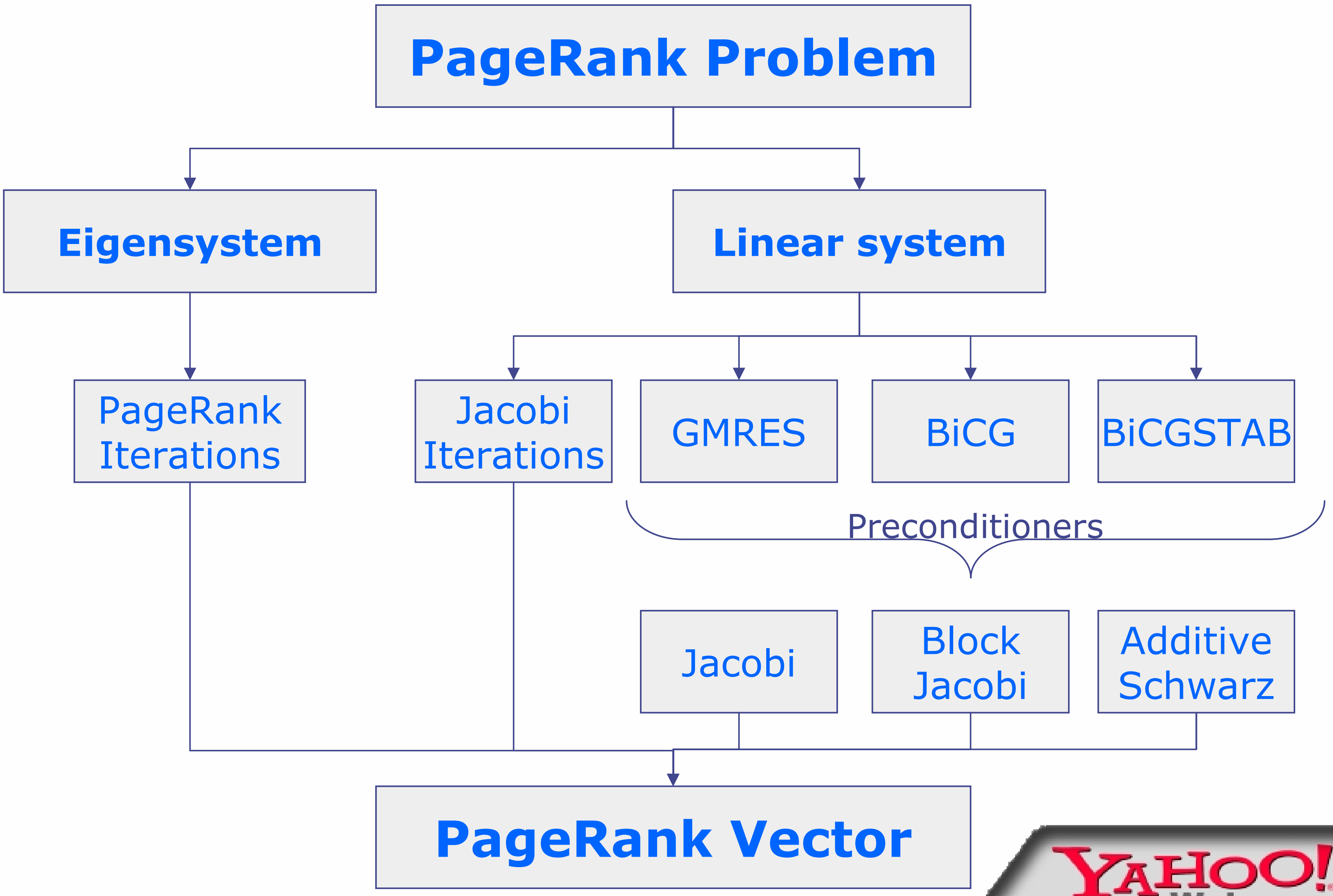
$$K_m = span\{r, Ar, A^2r, .., A^m r\}$$

Key Idea: Use the extra information in the Krylov subspace to get a better approximation solution at the next step by explicitly minimizing within this subspace.

Important Note: KSP methods only use matrix-vector products.

YAHOO! Websearch

Computational Methods



YAHOO! Websearch

Computational Methods

PageRank Iterations: Convergence $\sim \lambda_2/\lambda_1$ = c.
Jacobi Iterations: Convergence similar to PR iterations.

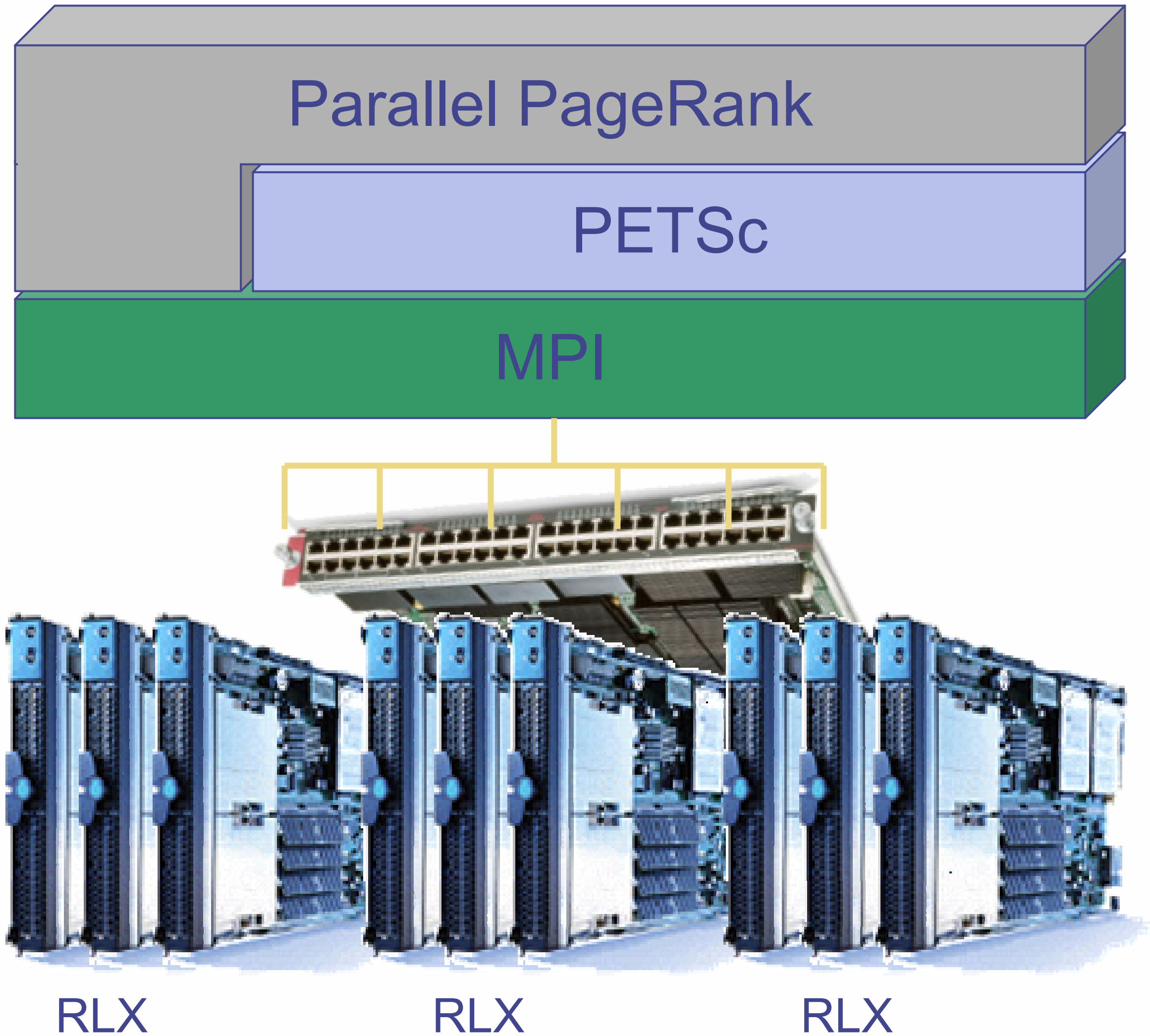
} Stationary Methods

GMRES: Most stable method, iterations can be expensive.
BiCG: Less stable but possibly faster than GMRES.
BiCGSTAB: “Combo” of BiCG and GMRES
Chebyshev, QMR, CGS,….

} Krylov Subspace Methods

Method	Inner Products	SAXPY	Matrix-Vector	Storage
PAGERANK		1	1	$M+3v$
JACOBI		1	1	$M+3v$
GMRES	$i+1$	$i+1$	1	$M+(i+5)v$
BiCG	2	5	2	$M+10v$
BiCGSTAB	4	6	2	$M+10v$

Building blocks of our system



Custom

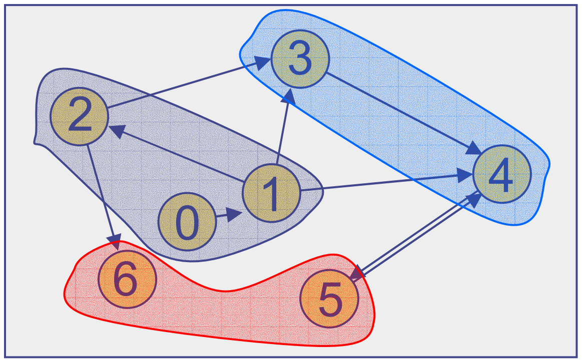
Off the shelf

Gigabit Switch

RLX Blades
Dual 2.8 GHz Xeon
4 GB RAM
Gigabit Ethernet
120 Total

Parallel Graphs

7 nodes, 9 edges



Goal: 3 nodes/proc

0	1	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	0	0	0

Balance Nodes Between Processors

The ideal graph distribution is given by an NP-hard problem. The standard approximate algorithms (ParMeTIS, pjostle, spectral) all fail when applied to webgraphs.

Practical solution

Fill up processors consecutively by row and keep adding rows until

$$w_{rows} n_p + w_{nnz} nnz_p > (w_{rows} n + w_{nnz} nnz)/p$$
$$w_{rows} : w_{nnz} = 1:1, 2:1, 4:1$$

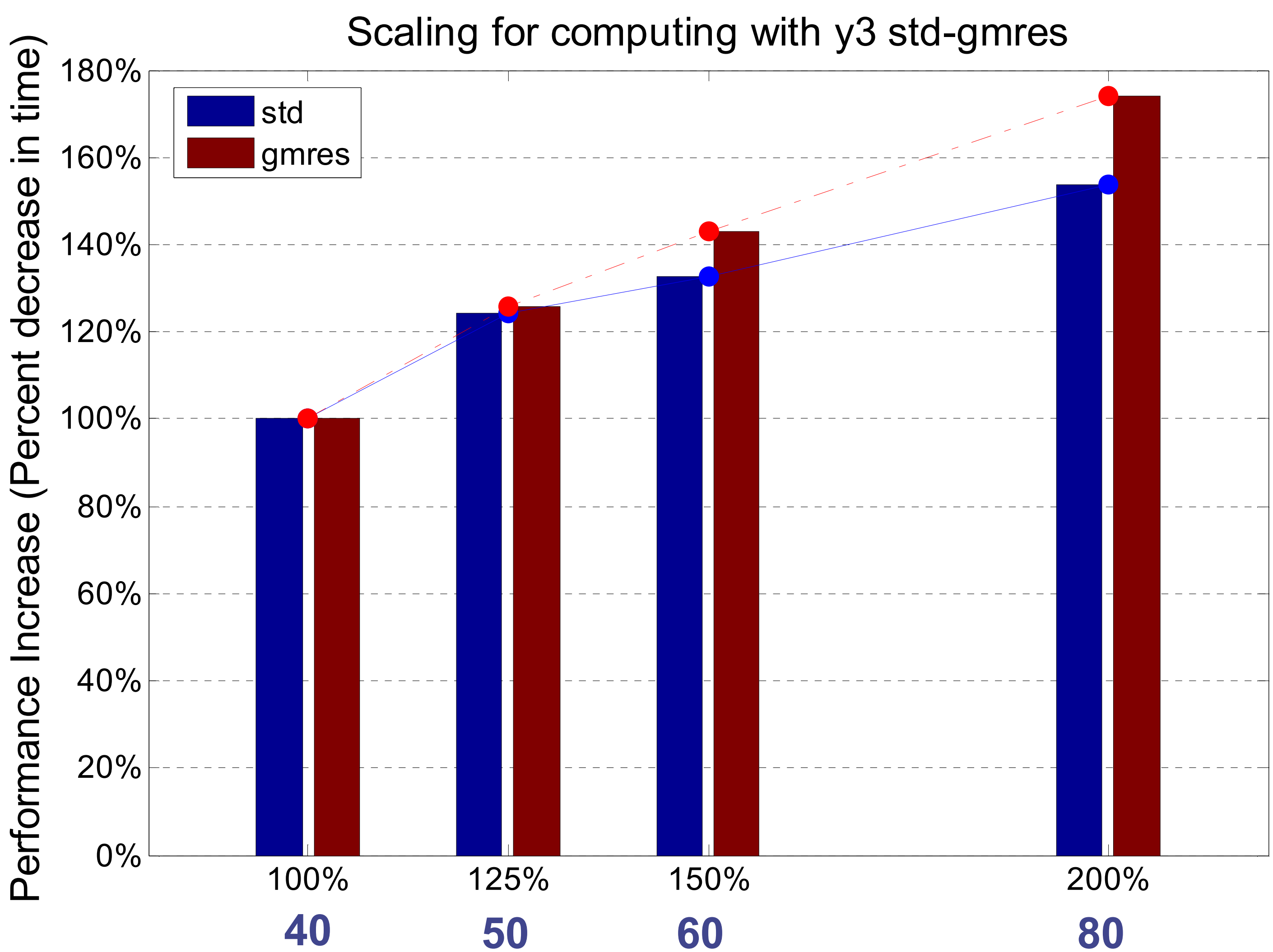
Experimental results

Name	Size	Power	Jacobi	GMRES	BiCG	BCGS
edu 20 procs	2M 14M	84 0.09/7.5s	84 0.07/6.5s	21* 0.6/13.2s	44* 0.4/17.7s	21* 0.4/8.7s
yahoo-r2 20 procs	14M 266M	71 1.8/129s	65 1.9/126s	12* 16/194s	35* 8.6/300s	17* 9.9/168s
uk 60 procs	18.5M 300M	73 0.09/7s	71 0.1/10s	22* 0.8/17.6s	25* 0.8/19.4s	11* 1.0/10.8s
yahoo-r3 60 procs	60M 850M	76 1.6/119s	75 1.5/112s			
db 60 procs	70M 1B	62 9.0/557s	58 8.7/506s	29 15/432s	45 15/676s	15* 15/220s
av 140 procs	1.4B 6.6B	72 4.6/333s				26 15/391s

The size is the number of nodes (pages) and number of edges (links).

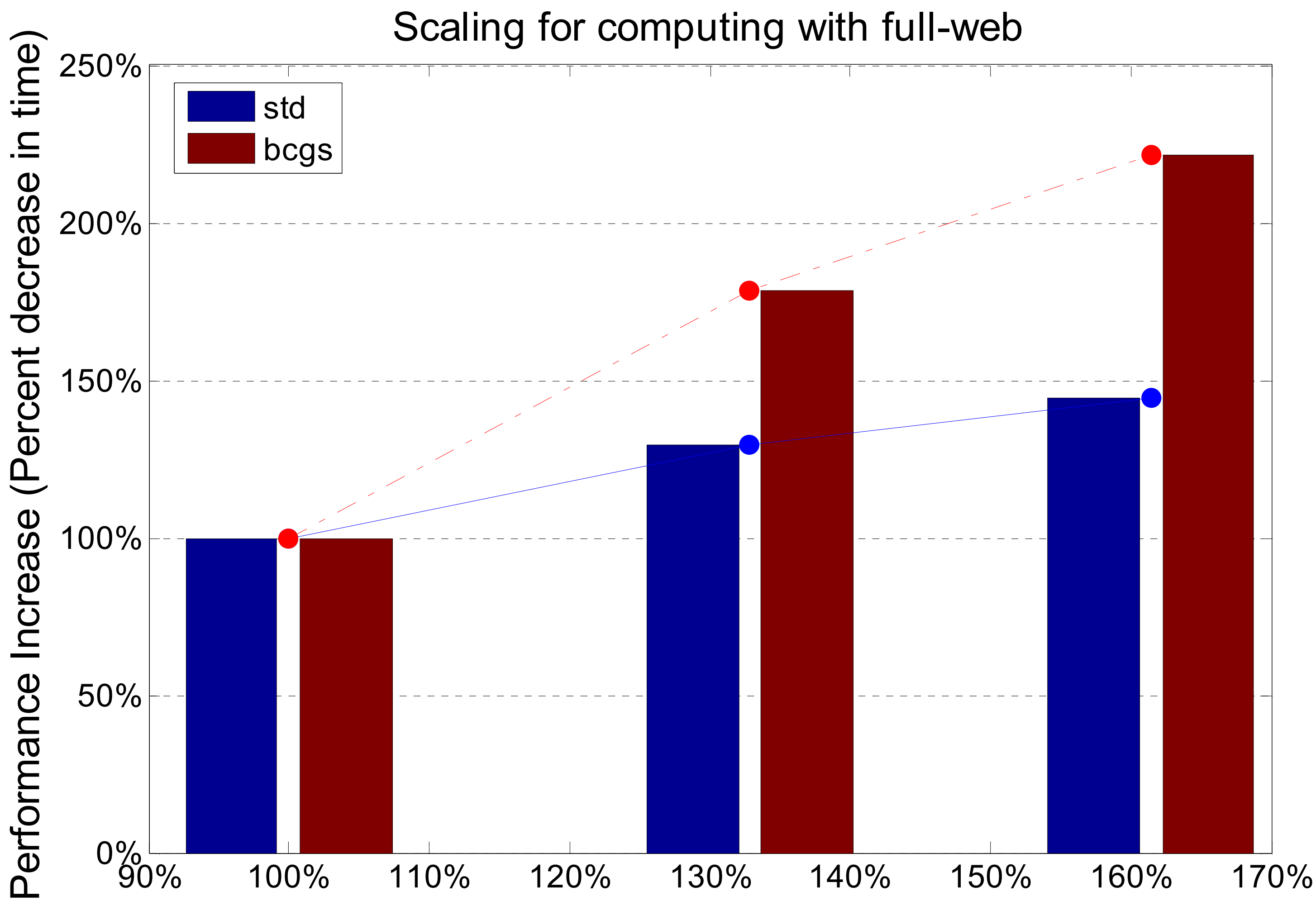
Each entry is the number of iterations, time per iteration, and total time. * denotes a preconditioner. Residual is 10^{-7}.

Parallelization



yahoo-r3: 60 M pages, 850 M links.

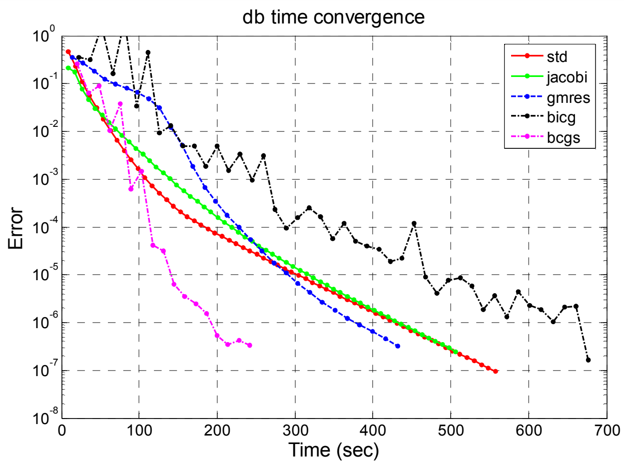
Full web parallelization

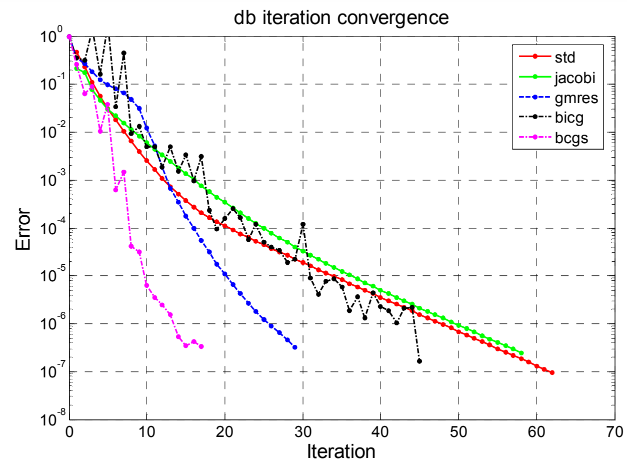


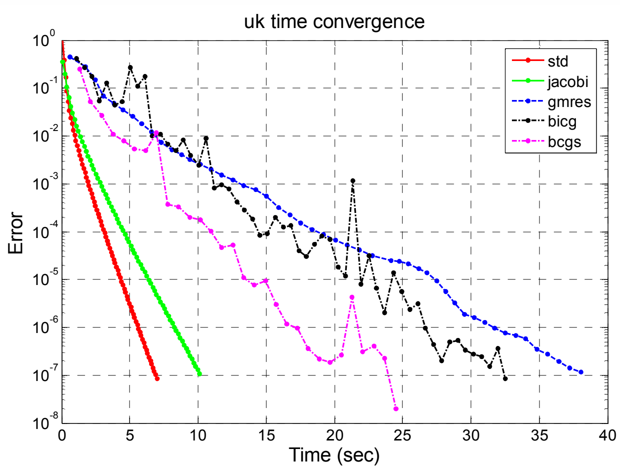
140 **186** **226**

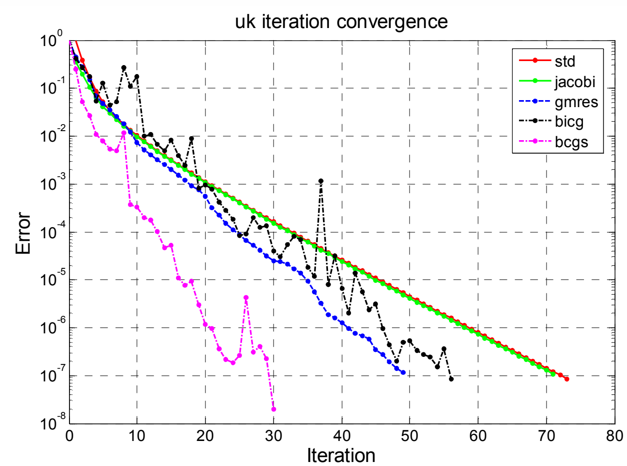
av: 1.4 B pages, 6.6 B links.

Convergence Results







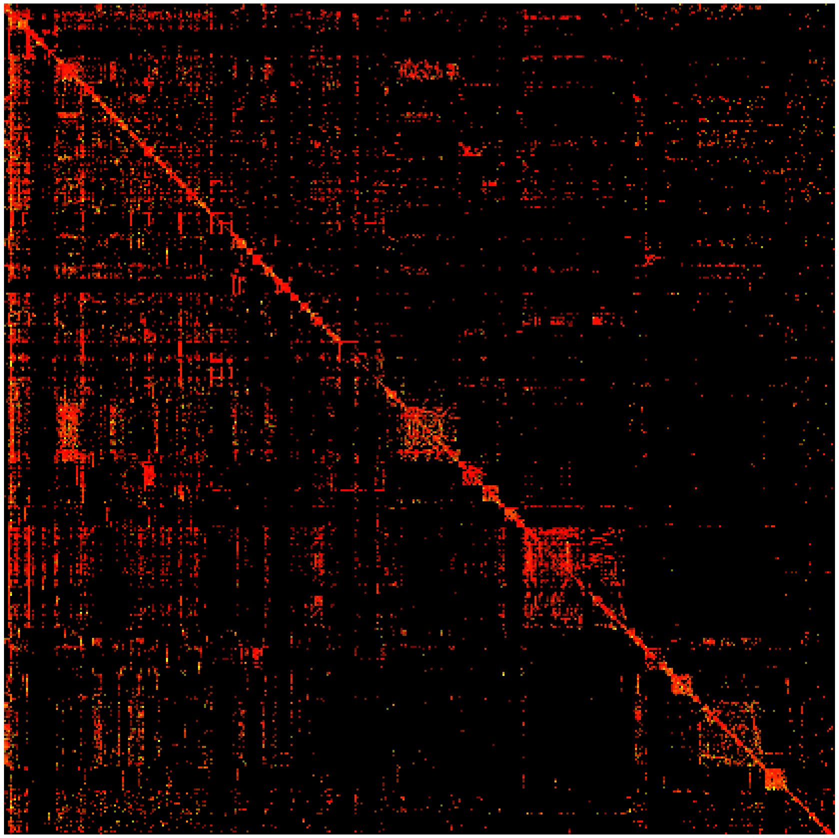


YAHOO! Websearch

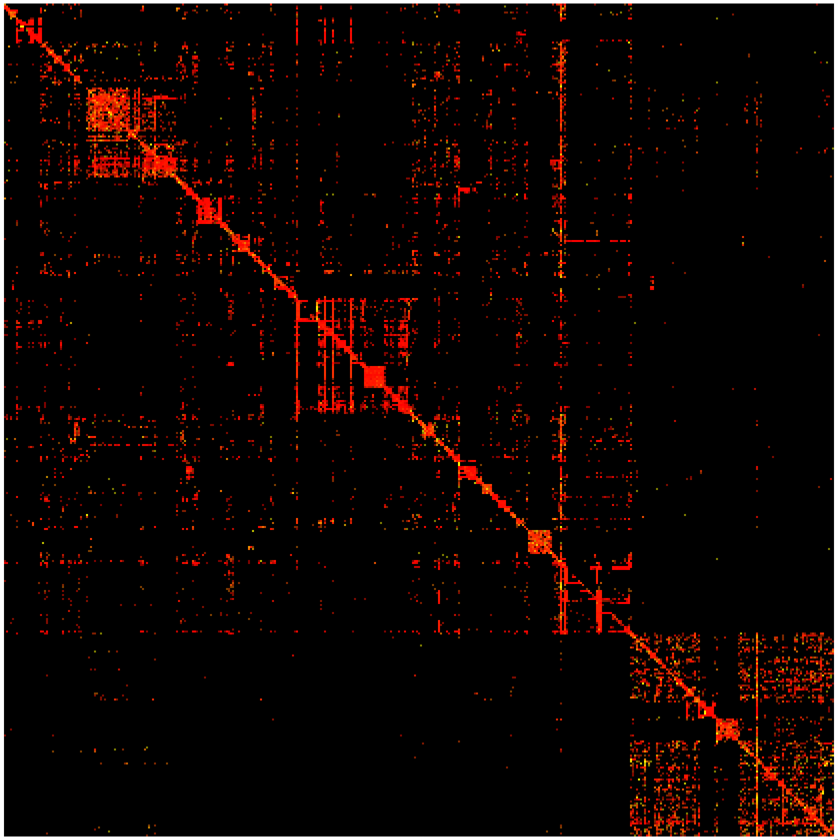
PageRank Acceleration Permutation

Domain lexicographic sorting reveals block structure in the webgraph.

http://host.domain.tld/path → http://tld.domain.host/path

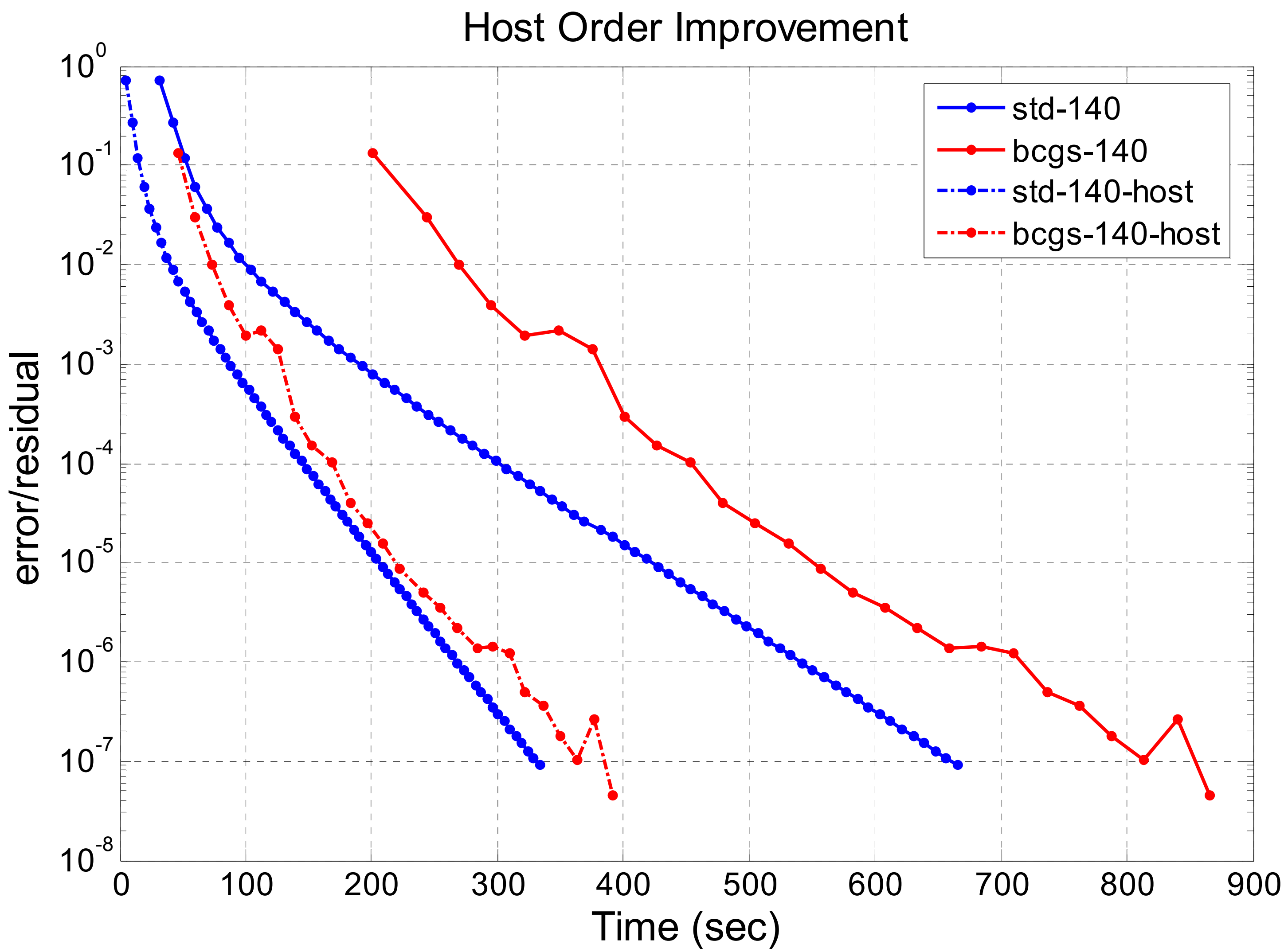


bs-cc: 17 k pages, 133 k links



YAHOO! Websearch

PageRank Acceleration Permutation

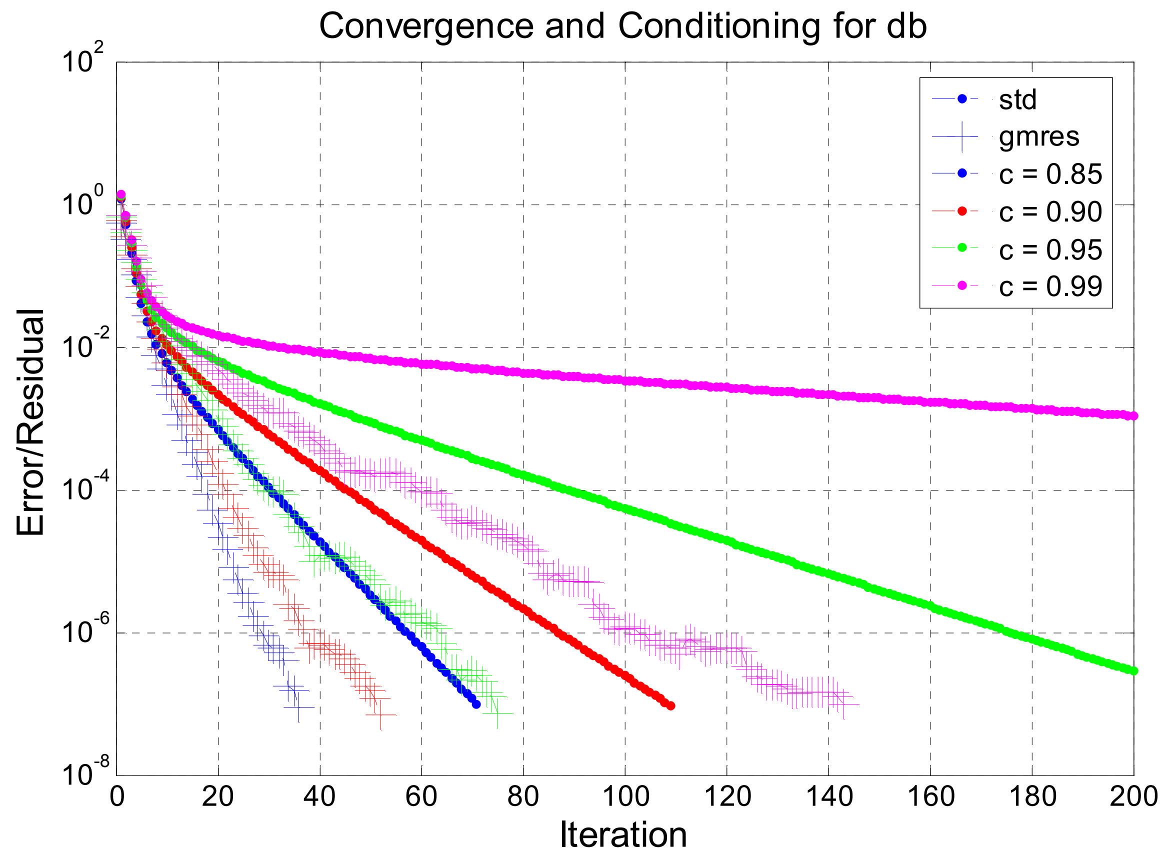


av: 1.4 B pages, 6.6 B links.

YAHOO! Websearch

Applications – High c

If we decrease the probability of random jumps the surfer makes, the problem becomes ill-conditioned. The advanced linear systems can still converge in this case.



db: 70 M pages, 1 B links.

Conclusion

PageRank can efficiently be computed as both an eigenvector and as a solution of a linear system on a distributed memory parallel machine.

The best method to use is graph and computing architecture dependent.

The PageRank problem scales wells on a fully-connected network topology.

The PageRank linear system can converge at high values of c.

David Gleich, Leonid Zhukov, and Pavel Berkhin. "Fast Parallel PageRank: A Linear System Approach." Yahoo! Technical Report, 2004. www.stanford.edu/~dgleich/publications/prlinear-dgleich.pdf