Parallel Solution of PageRank Problem

eero.vainikko@ut.ee

Teooriapäevad

Rõuge, 26th January 2007

Overview of the talk

1. Introduction (Problem description, Markov Chain)
2. Mathematical formulation of the PageRank Problem
3. Power iterations method
4. Linear system approach for solving PageRank Problem
5. General parallel solution techniques
6. DOUG package
7. DOUG & PageRank problem

1 Introduction

WWW is a huge collection of data distributed around the globe, in constant chane and growth

# pages indexed by Google	
May-June 2000	1 billion
November-December 2000	1.3 billion
July - August 2002	2.5 billion
November - December 2002	4 billion
January - February 2004	4.28 billion
November - December 2004	8 billion
August 2005	8.2 billion
January 2007 (an estimate)	≈14 billion

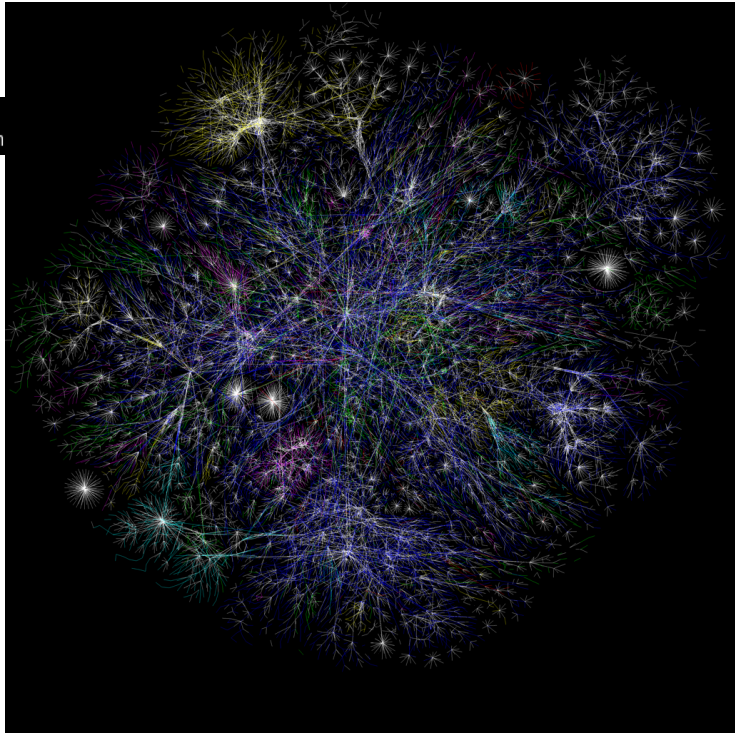
Roughly, doubling every 16 months

- Need really good tools for navigating, searching, indexing the information

How does Internet look like?

net, ca, us com, org mil, gov, edu
jp, cn, tw, au de, uk, it, pl, fr br, kr, nl unknown

Maps of the Internet (http://www.opte.org/maps/)



OK, these are just servers. Imagine, **how would the WWW look like?**

1.1 Description

Original proposal of the PageRank algorithm by L. Page, S. Brin, R. Motwani and T. Winograd, 1998

- one of the reasons why Google is so effective
- a method for computing the relative rank of web pages
- based on web link structure
- has become a natural part of modern search engines
- Also, a useful tool applied in many other search technologies, for example
  - Web spam detection [Z.Gyöngyi et al 2004]
  - crawler configuration
  - P2P trust networks [S.D.Kamvar et al 2003]

1.2 Markov process

Surfing the web, going from page to page by randomly choosing an outgoing link

- can lead to dead ends (*dangling nodes*)
- cycles

Sometimes choosing simply a random page from the Web.

Markov chain or *Markov process*

The limiting probability that an infinitely dedicated random surfer visits any particular page is its *PageRank*

2 Mathematical formulation of PageRank problem

2.1 Problem setup

W - set of web pages reachable in a chain following hyperlinks from a root page

G - corresponding $n \times n$ ***connectivity matrix:***

$$g_{ij} = \begin{cases} 1 & \text{if} \quad \exists \, \text{hyperlink } i \leftarrow j \\ 0 & \text{otherwise.} \end{cases}$$

- G can be huge, is sparse, column j shows the links on jth page
- # nonzeros in G - the total number of hyperlinks in W

Let r_i and c_j be the row and column sums of G:

$$r_i = \sum_j g_{ij}, \; c_j = \sum_i g_{ij}.$$

- r_i - *in-degree* of the ith page
- c_j - *out-degree* of the jth page.

Let p - the probability that the random walk follows a link.

- A typical value is $p = 0.85$
- $1 - p$ is the probability that some arbitrary page is chosen
- $\delta = (1-p)/n$ - probability that a particular random page is chosen.

Let B be the $n \times n$ matrix with elements b_{ij}:

$$b_{ij} = \begin{cases} pg_{ij}/c_j + \delta & : \quad c_j \neq 0 \\ 1/n & : \quad c_j = 0 \end{cases}$$

Notice that:

- B is not sparse
- most of the values $= \delta$ (the probability of jumping from one page to another without following link)
- If $n = 4 \cdot 10^9$ and $p = 0.85$, then $\delta = 3.75 \cdot 10^{-11}$
- B - the transition probability matrix of the Markov chain
- $0 < b_{ij} < 1$
- $\sum_{i=1}^{n} b_{ij} = 1, \forall i$

Matrix theory: ***Perron-Frobenius theorem*** applies:

$\exists!$ (within a scaling factor) solution $x \neq 0$ of the equation

$$x = Bx.$$

If the scaling factor is chosen such that $\sum_i x_i = 1$ then x is is the state vector of the Markov chain and is Google's PageRank; $0 < x_i < 1$.

2.2 Power method

Algorithm Power method
Input: Matrix B, initial vector x, threashold ε
Output: PageRank vector y
repeat
  $x \leftarrow Bx$
until $\|x - Bx\| < \varepsilon$
$y \leftarrow x / \|x\|$

In practice, matrix B (or G) is never formed.

2.3 Transfer to a linear system solution

the first idea: the solution of the problem

$$x = Bx$$

being equivalent to

$$(I - B)x = 0$$

But, the non-sparsity of $I - B$!

Is there a better way?

Yes: Note that

$$B = pGD + ez^T, \tag{1}$$

where D - diagonal matrix

$$d_{jj} = \begin{cases} 1/c_j & : \quad c_j \neq 0 \\ 0 & : \quad c_j = 0 \end{cases}, \; e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \; z = \begin{cases} \delta & : \quad c_j \neq 0 \\ 1/n & : \quad c_j = 0 \end{cases}$$

- ez^T - rank-one matrix - the random choices of Web pages that do not follow links.

The equation

$$x = Bx$$

is becoming thus due to (1):

$$x = (pGD + ez^T)x$$

$$x - pGDx = e\underbrace{z^T x}_{\gamma}$$

$$\underbrace{(I - pGD)}_{A} = \gamma e\,,$$

we get the system of linear equations to solve:

$$Ax = e \tag{2}$$

(We temporarily take $\gamma = 1$.) After solution of (2), the resulting x can be scaled so that $\sum_i x_i = 1$ to obtain PageRank.

Note that the matrix $A = I - pGD$ is

- sparse
- nonsinguar, if $p < 1$
- nonsymmetric
- huge in size

3 Solution methods for (2)

Solve the system of linear equations

$$A\mathbf{x} = \mathbf{b}$$

where the matrix A is:

- sparse,
- large,
- may have highly varying coefficients (for example, $|a_{ij}| \in [10^{-6}, 10^{6}]$)

3.1 Available methods

Direct methods

UMFPACK, SuperLU, MUMPS

- Analysing step
- factorisation step
- solving step

Roughly 100-10-1 time factor. 2D - OK, 3D - ?.

Iterative methods

- Richardson's type iterations (Gauss-Seidel, SSOR,...)
- Krylov subspace methods

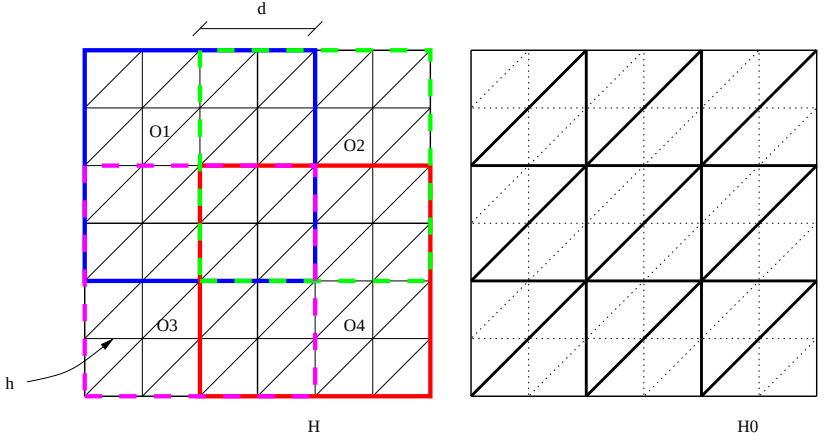
Domain Decomposition (DD)

- non-overlapping methods

substructuring methods, additive average methods and others.

- overlapping methods

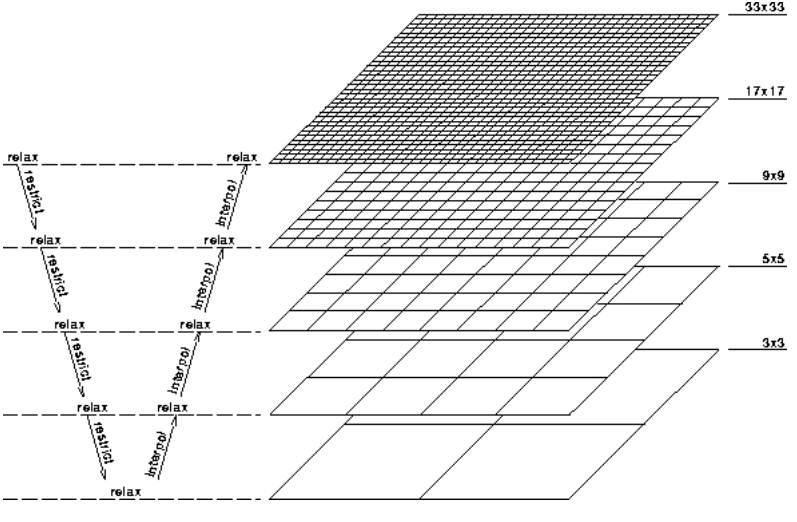
Additive Schwarz methods



MultiGrid

Generalisation of DD to multiple levels, but:
moderate coarsening from finer to coarser levels

- Geometric multigrid



- Algebraic multigrid
  - f-c colouring
  - aggregation-based

4 DOUG

4.1 DOUG – fast "*black box*" solver

Domain Decomposition on Unstructured Grids

DOUG (*University of Bath, University of Tartu*)

I.G.Graham, M.Haggers, R. Scheichl, L.Stals, E.Vainikko, K.Skaburskas, M.Tehver, O.Batrašev, C.Pöcher, M.Niitsoo 1997 - 2007

DOUG developent site (`http://dougdevel.org`)

Parallel implementation based on:

- **MPI**
- **UMFPACK**
- **(METIS)**
- **BLAS**

4.2 DOUG (vers. 2) overview

- Large linear system solver
- automatic parallelisation and load-balancing
- Block-structured matrices (systems of PDEs)
- 2D & 3D problems
- 2-level Additiivne Schwarz method
- 2-level partitioning of the domain
- Automatic Coarse Grid generation
- Adaptive refinement of the coarse grid
- Different input-types for linear systems
- GRID-enabled WWW-interface

4.3 Overview of DOUG strategies

- Iterative solver based on Krylov subspace methods
  PCG, MINRES, BICGSTAB, 2-layered **FPGMRES** with left or right preconditioning.

- Non-blocking communication where at all possible

  Ax-operation: $\mathbf{y} := A\mathbf{x}$ – :-)

  Dot-product: $(\mathbf{x}, \mathbf{y})$ – :-(

- Preconditioner based on Domain Decomposition with 2-level solvers

  Applying the preconditioner P: solve for $\mathbf{z} : P\mathbf{z} = \mathbf{r}$. :?

- Subproblems are solved with a direct, sparse multifrontal solver (UMFPACK)

5 DOUG95 & *aggregation*

5.1 Aggregation-based DD methods

Have been analysed upto some extent:
- Analysis for multiplicative Schwarz [Vanek & Brezina, 1999]
- Analysis for additive Schwarz [Jenkins et al., 2001] and [Lasser & Tosselli, 2002].
- Sharper bounds [R. Scheichl, E. Vainikko, 2006

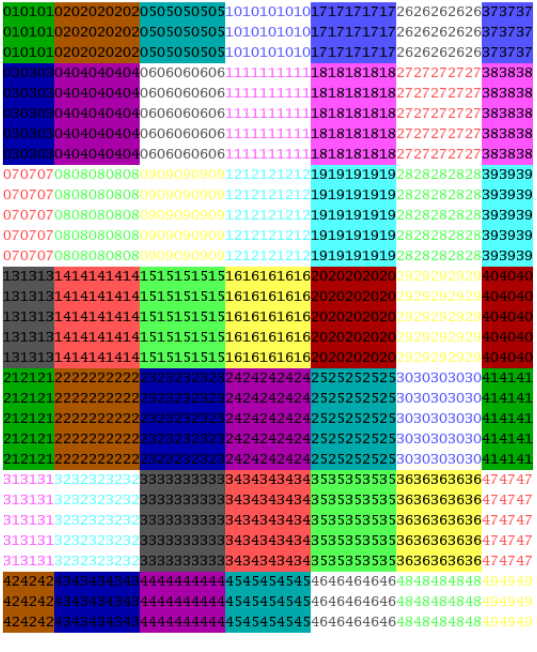
Aggregation:

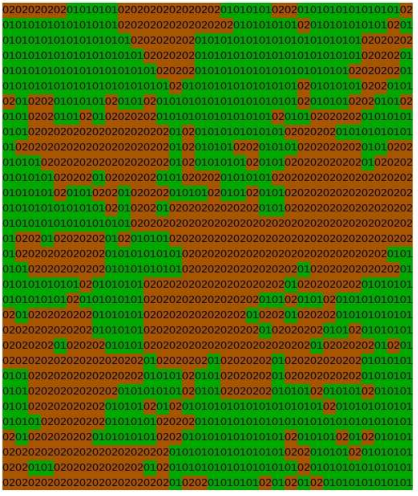
Key issues:

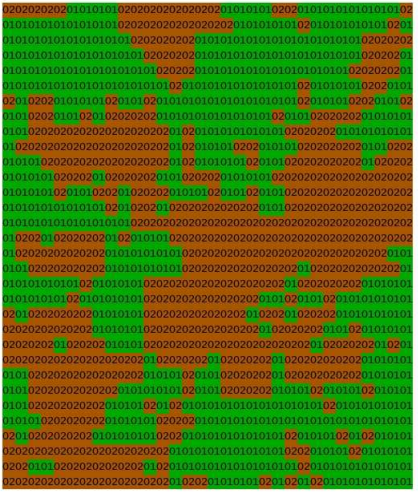
- how to find good aggregates?
- Smoothing step(s) for restriction and interpolation operators

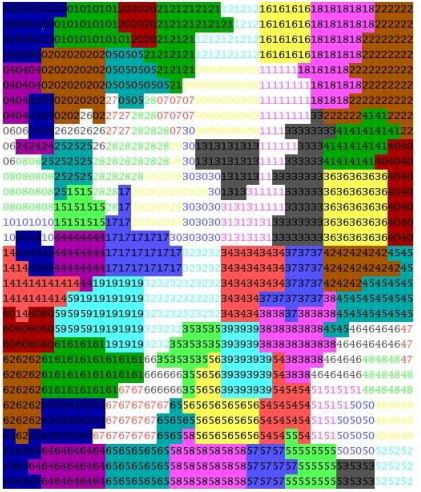
Four (often conflicting) aims:

- follow adequatly underlying physial properties of the domain
- try to retain optimal aggregate size
- keep the shape of aggregates regular
- reduce communication => develop aggregates with smooth boundaries









5.2 Algorithm (Shape-preserving aggregation)

Input: Matrix A, aggregation radius r, strong connection threashold α.

Output: Aggregate number for each node in the domain.

1. Scale A to unit diagonal matrix (all ones on diagonal
2. Find the set S of matrix A strong connectons: $S = \cup_{i=1}^{n} S_i$, where

$$S_i \equiv \{j \neq i : |a_{ij}| \geq \alpha \max_{k \neq i} |a_{ik}|,$$

unscale A; aggr_num:=0;

3. aggr_num:=aggr_num+1;

Choose a seednode x from G or if $G = \emptyset$, choose the first nonaggregated node x;

level:=0

4. If (level<r)

Add recursively all strongly connected non-aggrega

neighbours to the aggregate aggr_num with level+1

and perform smoothing step on each level

elseif (level<$2r$)

Find layer(level+1)...layer($2r$).

endif

5. On the longest layer(i), i=$r+1,...,2r$ add node(s) with shortest distance from x to the set G and goto step 3.

5.3 Parallel implementation

Interpolation ($I = R^T$), restriction (R) operators + smoothing operator – sparse matrix structure.

Coarse matrix $A_0 = RAR^T$, through **sparse matrix multiplication (SMM) operations are key routines** affecting the performance of initialisation stage.

- SMM produce sparse matrices
- SMM in our case easy to parallelise, as all data is available locally (due to overlap)

6 DOUG@GRID

6.1 Motivation

PROBLEM:

- Dynamic nature of GRID *versus*:
  – good parallel solvers need synchronisation steps :-(
  – no fault tolerance in mainstream MPI implementations :-(

=> A) need for methods that do not need regular synchronisation

=> B) Need for fault-tolerant communication libraries

6.2 Possible solutions:

A) Algorithms

- Asynchronous DD methods
  - do not base on Krylov subspace methods (Richardson's type iteration methods)
  - slower convergence
  - not very much studied mathematically (due to stochastic nature)

- Possible asynchronous Krylov subspace methods
  - do such exist at all? (Flexible GMRES)
  - some synchronisation still needed

B) Fault tolerance

- Important for long-running computations

DOUG as a web-service

Using GRID as a development utility

- running DOUG health-checks during the development process
- Automatic profiling system
- Using pre/post-commit scripts of Subversion to achieve it

6.3 DOUG strategies for PageRank problem

Using iterative solvers for the PageRank Linear System solution are reported to be very problem dependant.

- Our main interest: **how our aggregation-based DD Methods will work with PageRank**

An ongoing work

Strategies:

- Krylov subspace methods: PBiCGSTAB, PGMRES
- Asynchronous Domain Decomposition methods based on Gauss-Seidel methods

- work in progress

Questions?

Thank You!