Parallellization of a semantic discovery tool for a search engine

Kalle Happonen
Wray Buntine

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

HELSINKI INSTITUTE OF PHYSICS

MPCA

MPCA is a discrete version of PCA (principle components, top K eigenvectors of a matrix) for sparse integer data.

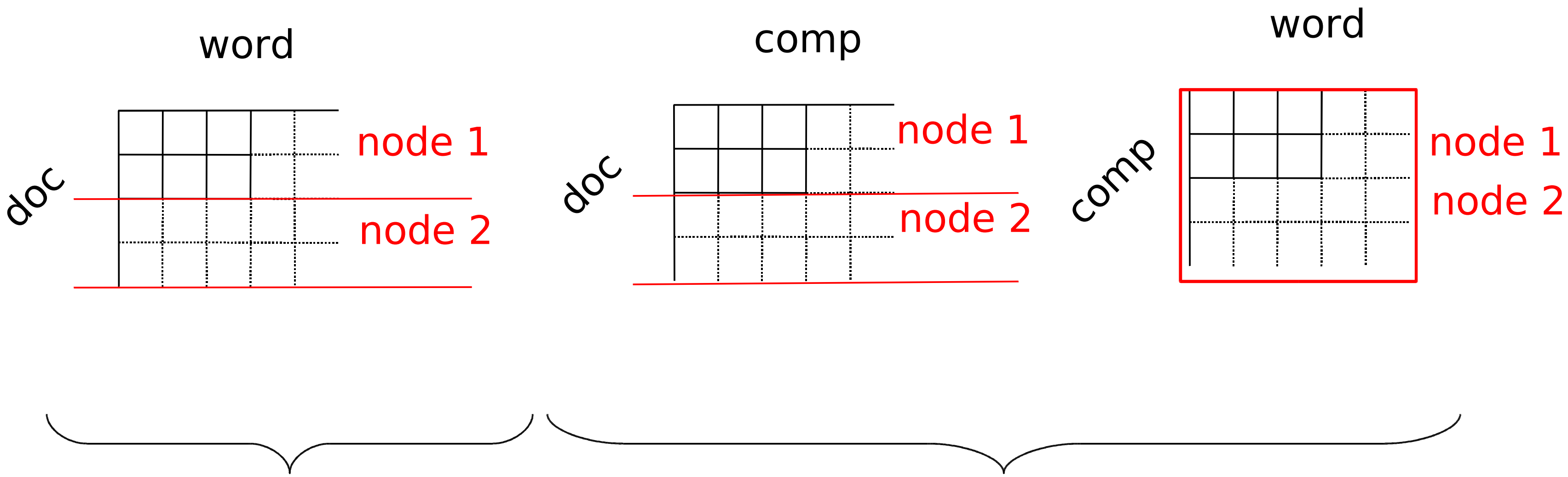
To make it simple

- goes through documents to build a sparse document-word matrix
- creates different categories based on word occurrence
- assigns membership level for documents based on relevance

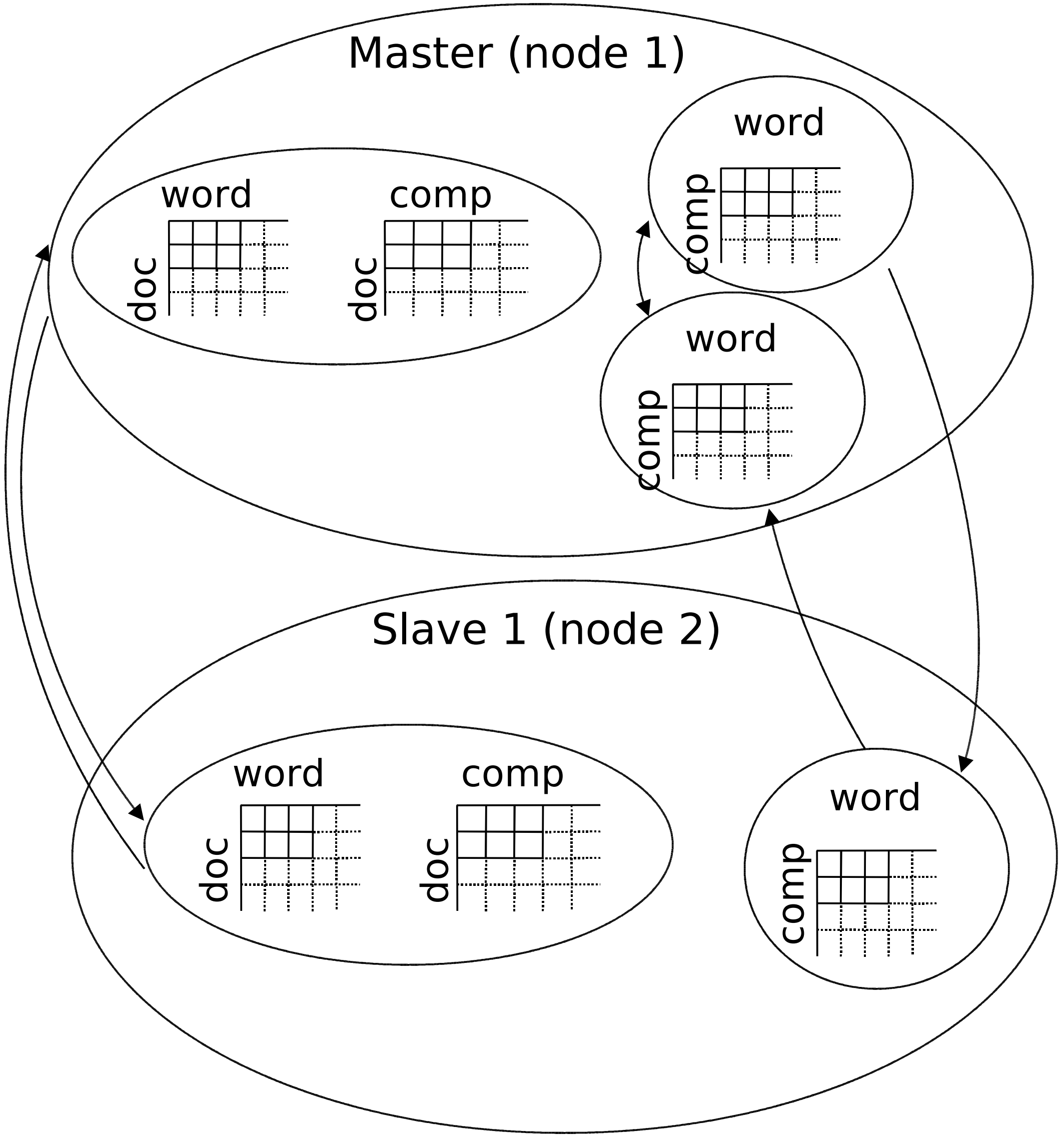
Uses are

- building categories of documents, e.g., Open Directory Project or DMOZ has hierarchical categories
- partitioning a subset of the web into strongly connected regions and identifying hubs (good outgoing links) and authorities (commonly linked to)
- also used in the genome project for genotype discovery, although not this particular software

The matrices



How it works


Master (node 1)
word
comp
doc
doc
word
comp
word
comp
Slave 1 (node 2)
word
comp
doc
doc
word
comp

The parallel version

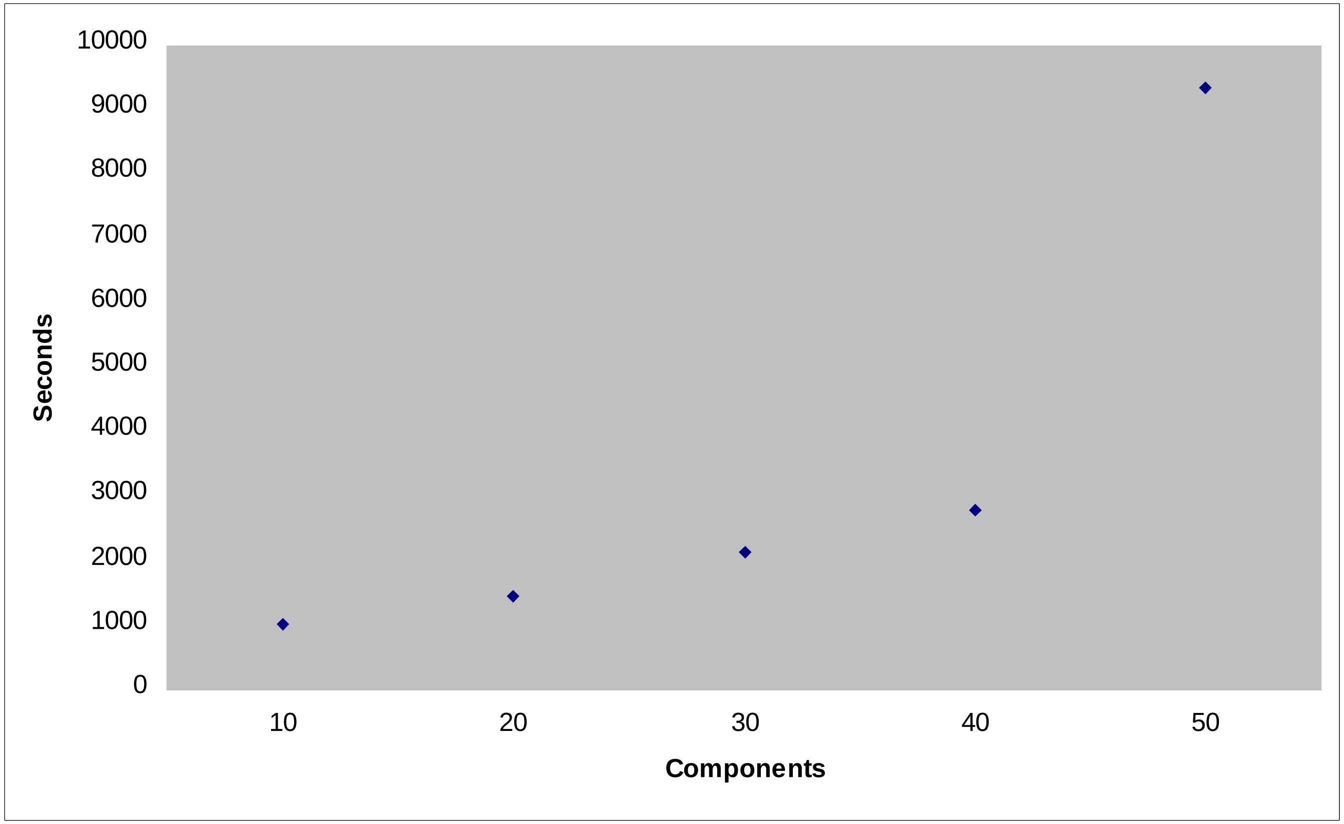
Relevant facts

- The task is divided by splitting up the documents
- The component - word matrix is large – a lot of data to transfer
- Total amount of data to transfer between loops is linearilly dependent on node amount
- The component - word matrix is always in memory – the available memory limits the component amount for a problem

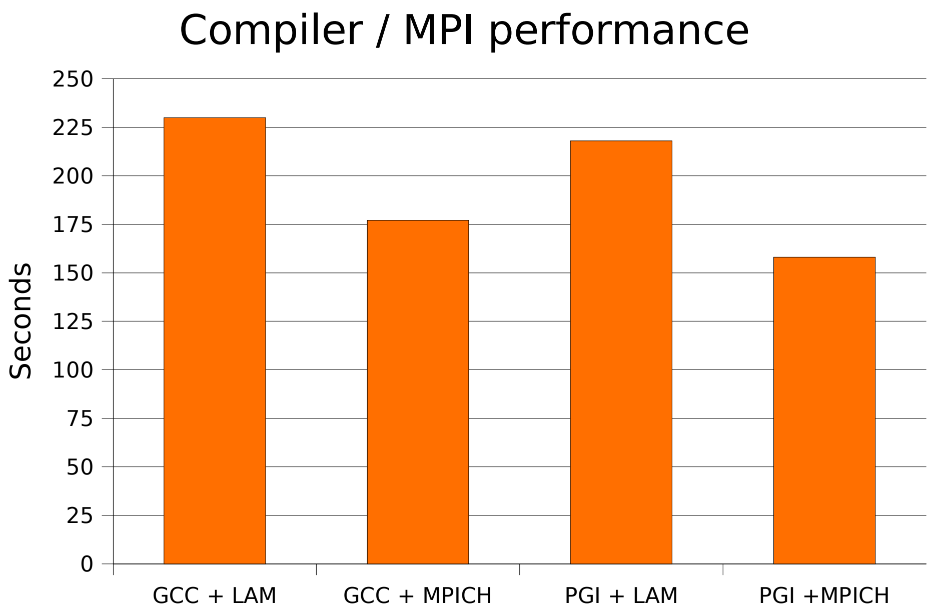
What we have done

- We are using MPI for parallellization
- We have a working tested parallel version – with room for improvement

Complexity


10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Seconds
10
20
30
40
50
Components

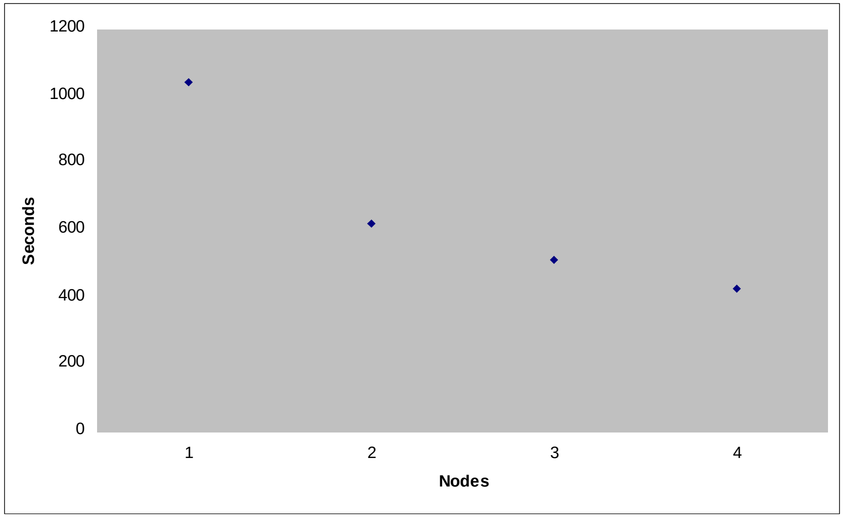
Some performance data

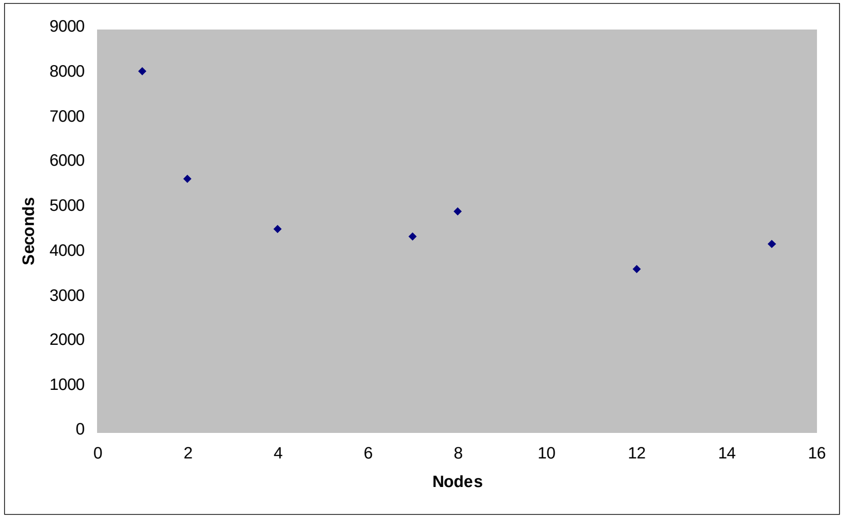


Small testrun on a 4 node cluster

- 2 * Athlon64 cpus / node
- 2 GB memory / node
- gigabit ethernet

Scalability

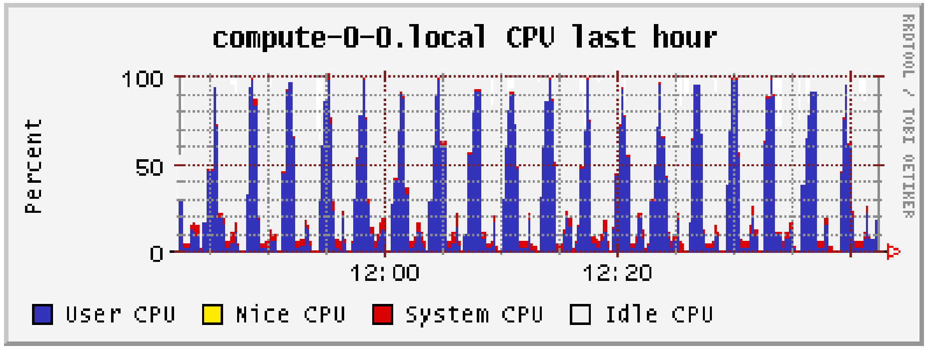


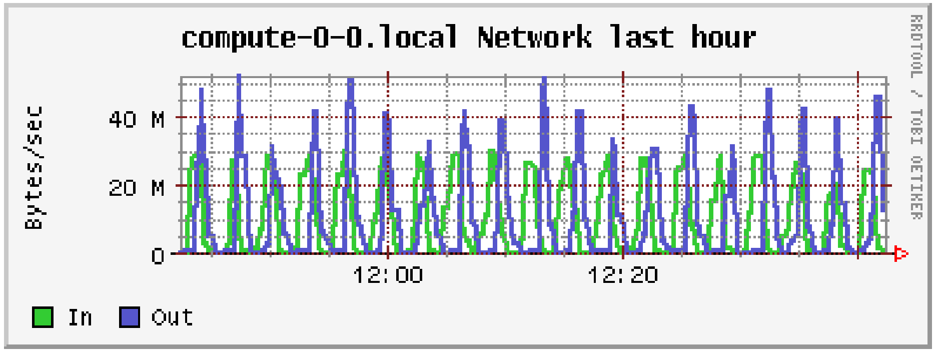


HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

HELSINKI INSTITUTE OF PHYSICS

Problems





Data transfer the limiting factor - solutions

- Use a faster network
- Compress data before transfer
- Change the communication model - possibly need to break the algorithm
- Run on reduced problem to convergence – then run on full problem

Memory requirements

- Larger jobs mean more words -> less components with the same amount of memory
- Run large jobs on nodes with much memory?
- Compress data in memory?

The present and the future

The Present

- Working parallellization with MPI
- Succesful runs on clusters
- Has been used on the whole Finnish web (3 M docs), Wikipedia (600k docs) and DMOZ (6 M docs)
- Problems with memory requirements and communication

The Future

- Run on much larger document sets
- Improve scalability
- Runs on a grid