UNIVERSITY OF TWENTE.

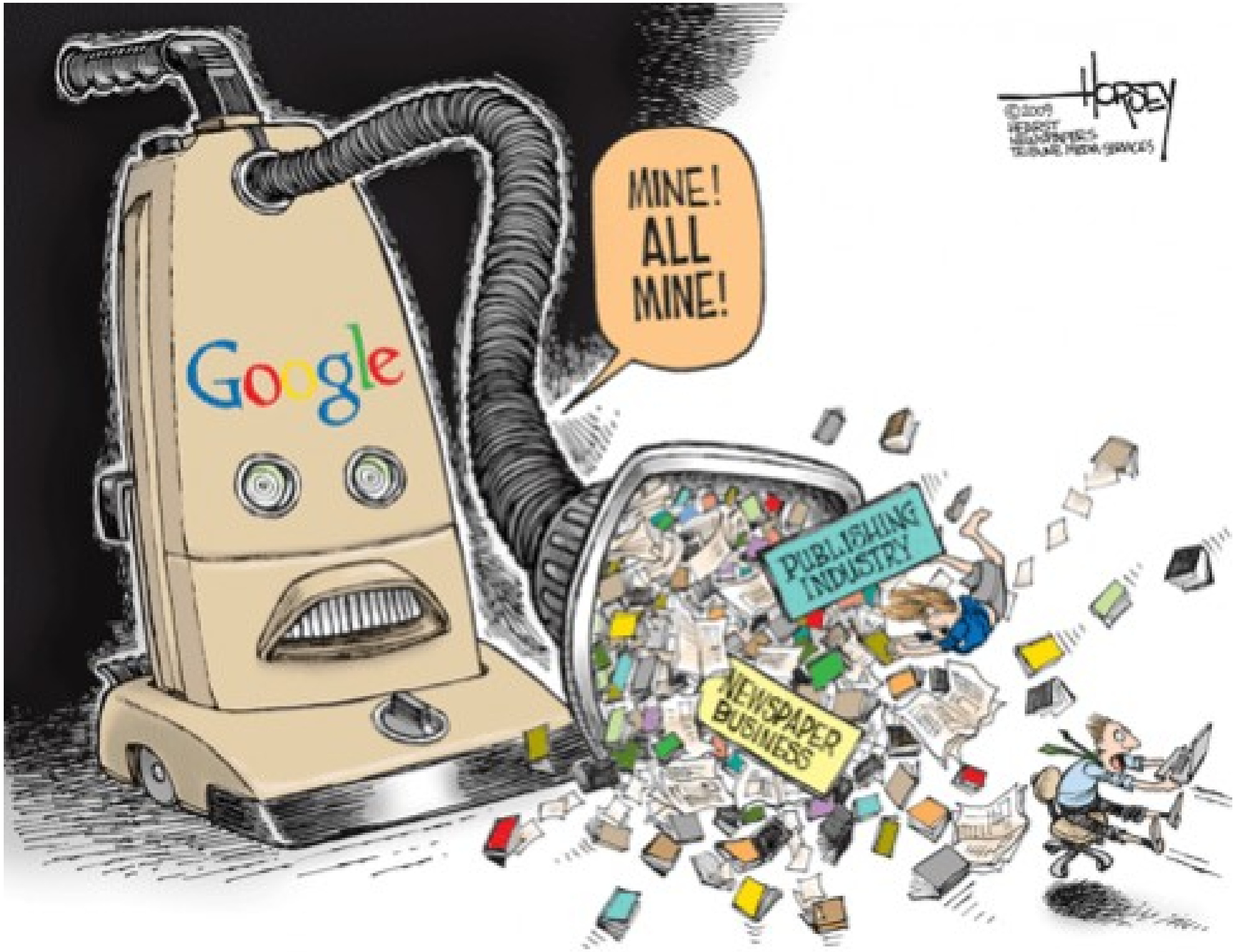
MAPREDUCE INFORMATION RETRIEVAL EXPERIMENTS

CLEF 2010, Tuesday 21 September 2010

Djoerd Hiemstra & Claudia Hauff

University of Twente

INSPIRED BY GOOGLE...



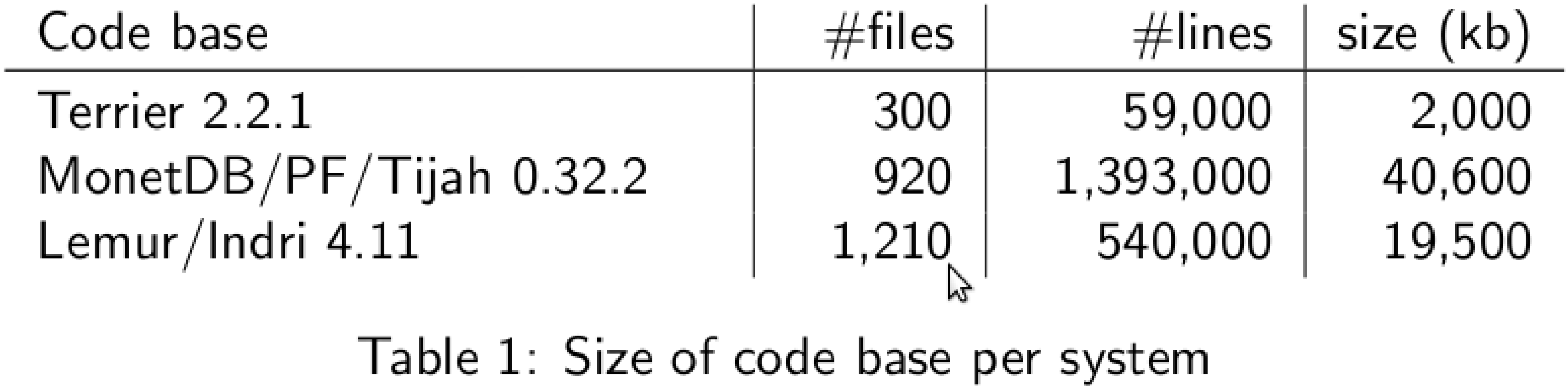
… A NEW COURSE ON "BIG DATA"

- Distributed Data Processing using MapReduce
  - M.Sc. Course Computer Science
  - with Maarten Fokkinga
  - Nov. 2009 – Feb. 2010

FAQ: HOW TO DO CLEF?

1. Have a really cool new idea :-)
2. Code the new approach in PF/Tijah, or Lemur, or Terrier, or Lucene... :-(
3. Index documents from a test collection :-|
4. Put the test queries to the experimental search engine and gather the top X results :-|
5. Compare the top X to a golden standard :-)
6. Done! :-P

CODE THE NEW APPROACH?



Code base	#files	#lines	size (kb)
Terrier 2.2.1	300	59,000	2,000
MonetDB/PF/Tijah 0.32.2	920	1,393,000	40,600
Lemur/Indri 4.11	1,210	540,000	19,500

Table 1: Size of code base per system

MAP/REDUCE

“A simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.”

Dean and Ghermawat, “MapReduce: Simplified Data Processing on Large Clusters”, 2004

MAP/REDUCE

- More simply, MapReduce is:

  A parallel programming model
  (and implementation)

MAP/REDUCE PROGRAMMING MODEL

- Process data using map() and reduce() functions
  - □ The map() function is called on every item in the input and emits intermediate key/value pairs
  - □ All values associated with a given key are grouped together
  - □ The reduce() function is called on every unique key, and its value list, and emits output values

MAP/REDUCE: PROGRAMMING MODEL

- More formally,
  - map(k1,v1) --> list(k2,v2)
  - reduce(k2, list(v2)) --> list(v2)

MAP/REDUCE: WORD COUNT EXAMPLE

```
mapper (DocId, DocText) =
  FOREACH Word IN DocText
    OUTPUT(Word, 1)


reducer (Word, Counts) =
  Sum = 0
  FOREACH Count IN Counts
    Sum = Sum + Count
  OUTPUT(Word, Count)
```

MAP/REDUCE RUNTIME SYSTEM

1. Partitions input data
2. Schedules execution across a set of machines
3. Handles machine failure
4. Manages interprocess communication

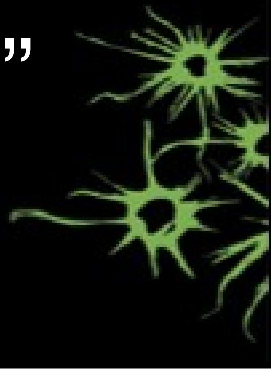
MAP/REDUCE: ANCHOR TEXTS

```
mapper (DocId, DocText) =
  FOREACH (AnchorText, Url) IN DocText
    OUTPUT(Url, AnchorText)

reducer (Url, AnchorTexts) =
  OutText = ''
  FOREACH AnchorText IN AnchorTexts
    OutText = OutText + AnchorText
  OUTPUT(Url, OutText)
```

MAP/REDUCE: SEQUENTIAL IR

```
mapper (DocId, DocText) =
  FOREACH (QueryID, QueryText) IN Queries
    Score = cool_score(QueryText, DocText)
    IF (Score > 0)
    THEN OUTPUT(QueryId, (DocId, Score))

reducer (QueryId, DocIdScorePairs) =
  RankedList = ARRAY[1000]
  FOREACH (DocId, Score) IN DocIdScorePairs
    IF (NOT filled(RankedList) OR
      Score > smallest score(RankedList))
    THEN ranked_ins(RankedList, (DocId, Score))
  FOREACH (DocId, Score) IN RankedList
    OUTPUT(QueryId, DocId, Score)
```

“LET’S QUICKLY TEST THIS ON 12 TB OF DATA”



CASE STUDY: CLUEWEB09

- Web crawl of 1 billion pages (25 TB)
  - □ crawled in Jan. – Feb. 2009
  - □ using only the English pages (0.5 billion)
- Cluster of 15 commodity machines
  - □ running Hadoop 0.19.2

YAHOO!

CODE THE NEW APPROACH

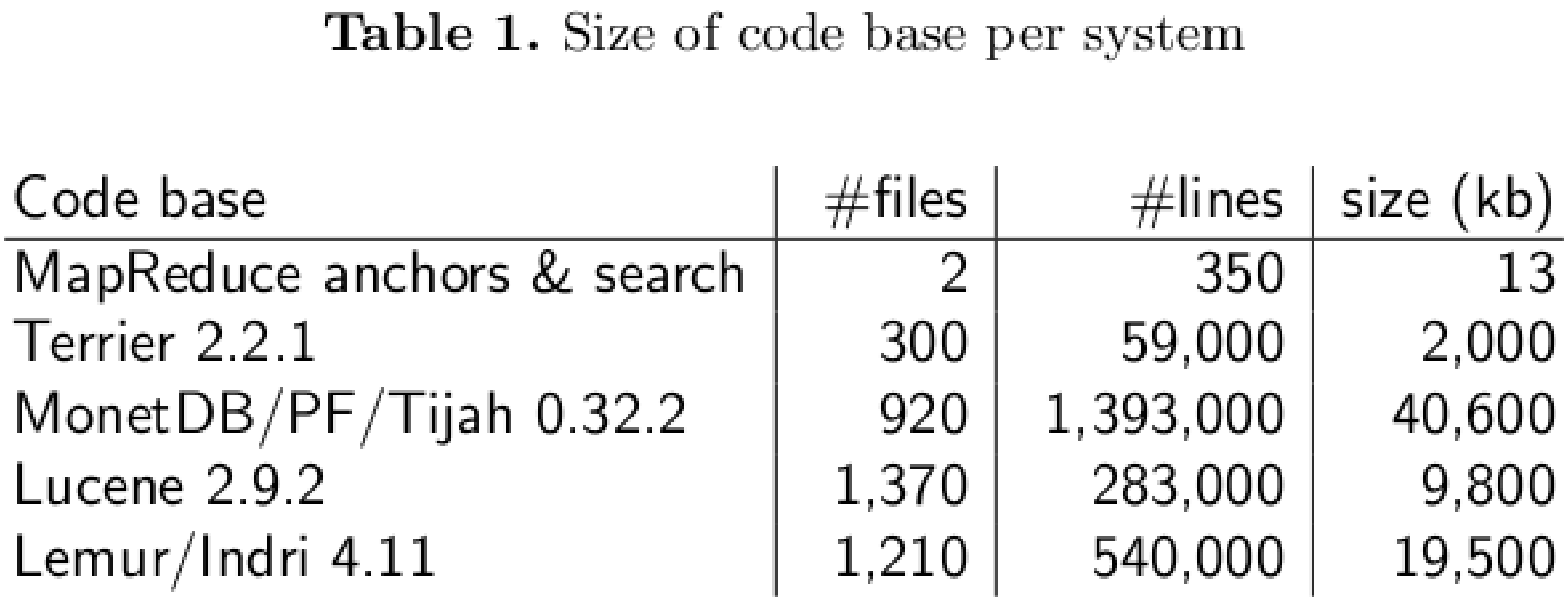


Table 1. Size of code base per system

Code base	#files	#lines	size (kb)
MapReduce anchors & search	2	350	13
Terrier 2.2.1	300	59,000	2,000
MonetDB/PF/Tijah 0.32.2	920	1,393,000	40,600
Lucene 2.9.2	1,370	283,000	9,800
Lemur/Indri 4.11	1,210	540,000	19,500

ANCHOR TEXTS

- Takes about 11 hours
- Anchor texts available from: http://mirex.sourceforge.net

SEQUENTIAL SEARCH

- 50 test queries take less than 30 minutes on Anchor Text representation
- Language model, no smoothing, length prior
- Expected Precision at 5, 10 and 20 documents (MTC method):

0.42 0.39 0.35

(0.44 0.42 0.38 U. Amsterdam)
(0.43 0.38 0.38 Microsoft Asia)
(0.42 0.40 0.39 Microsoft UK)

EXPERIMENTAL RESULTS

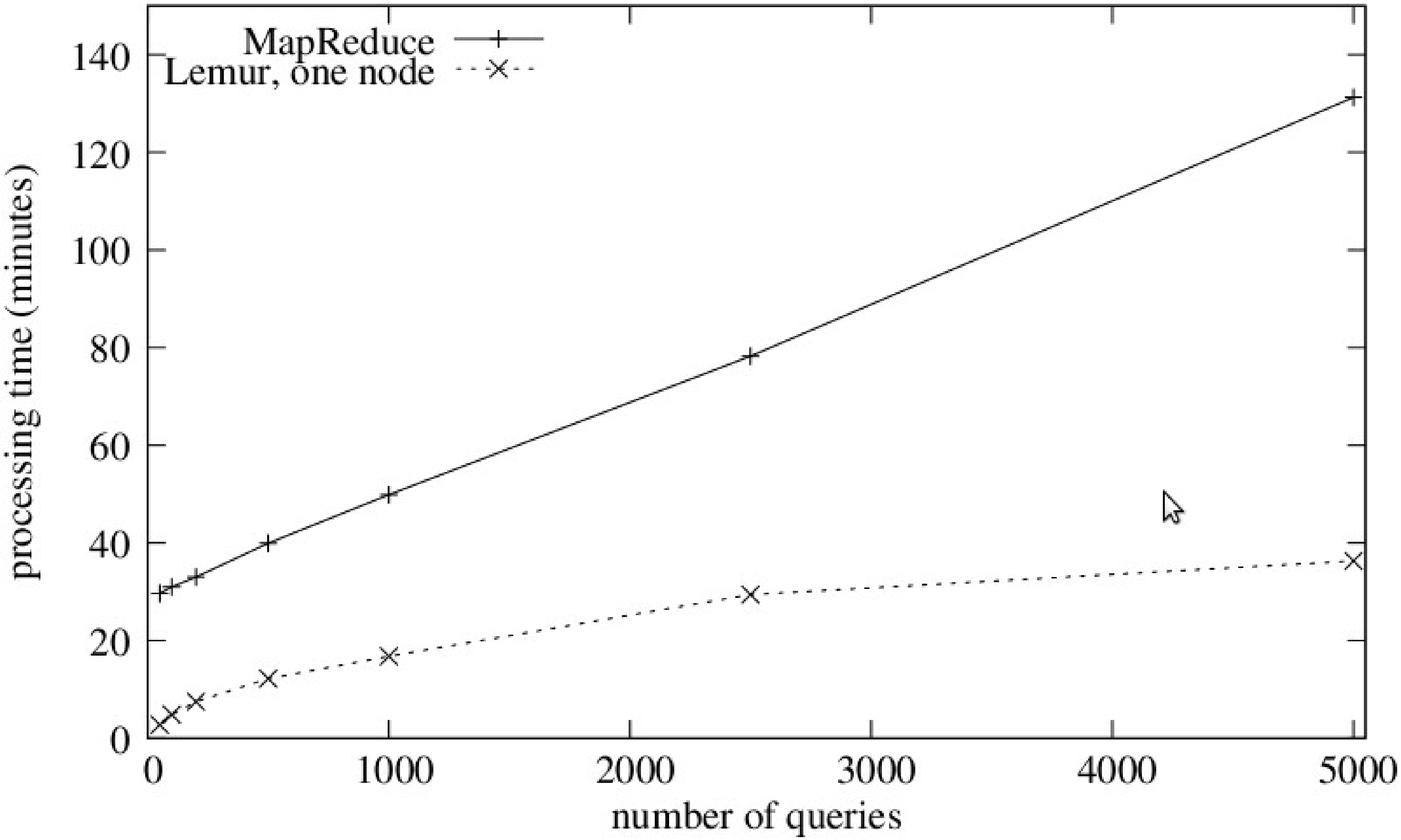


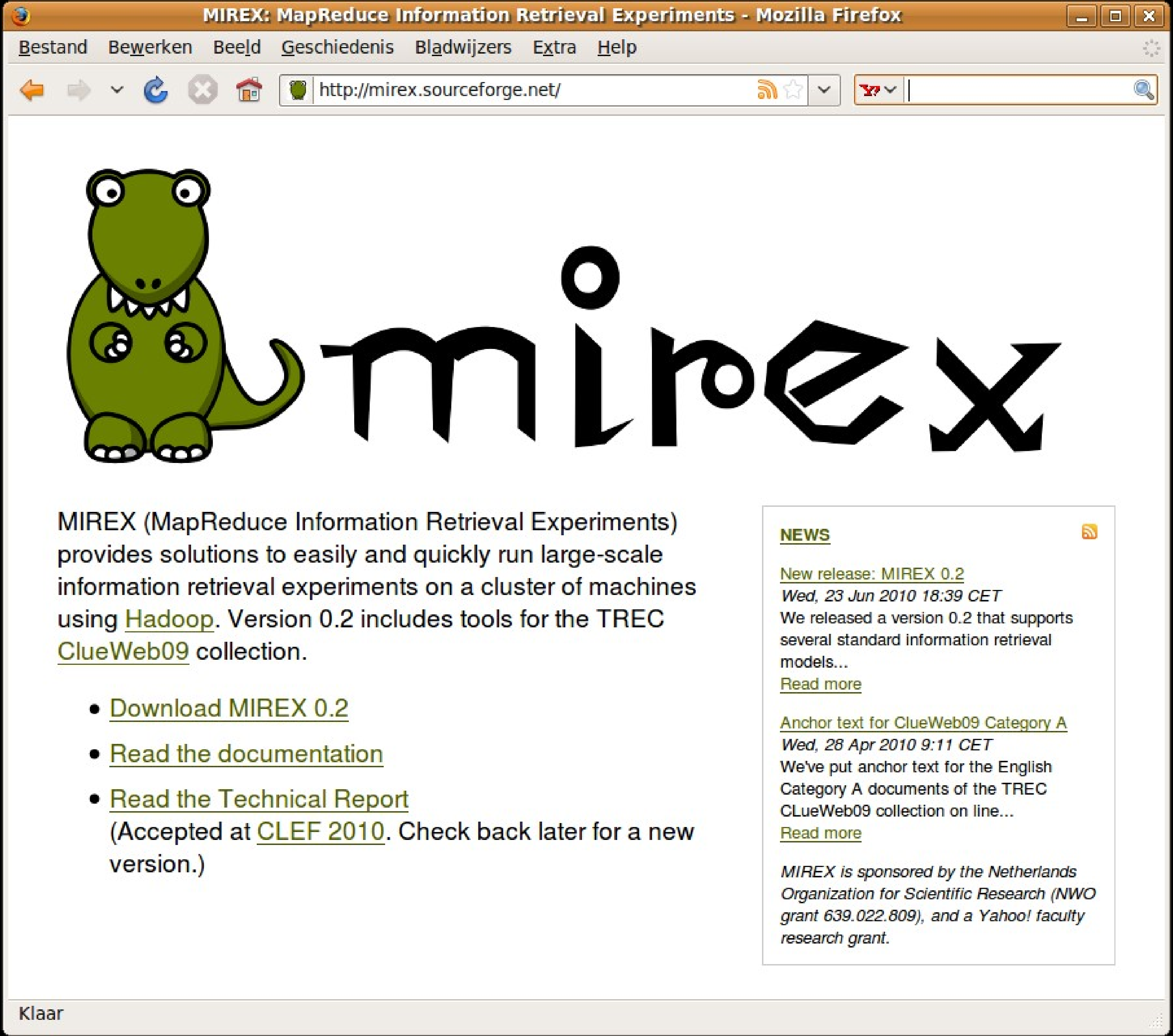
Figure 2: Processing time for query set sizes

BENEFITS FOR RESEARCHERS

1. Spend less time on coding and debugging
2. Easy to include new information that is not in the engine's standard inverted index
3. Oversee all the code used in the experiment
4. Large-scale experiments done in reasonable time

CONCLUSION

- Less than 10 times slower than “Lemur one node” (on same anchor index)
- Faster turnaround of the experimental cycle:
  - Faster coding
  - = more experiments
  - = more improvement of search quality
  - = better system!


MIREX: MapReduce Information Retrieval Experiments - Mozilla Firefox
Bestand
Bewerken
Beeld
Geschiedenis
Bladwijzers
Extra
Help
http://mirex.sourceforge.net/
mirex
Klaar

MIREX (MapReduce Information Retrieval Experiments) provides solutions to easily and quickly run large-scale information retrieval experiments on a cluster of machines using Hadoop. Version 0.2 includes tools for the TREC ClueWeb09 collection.

- Download MIREX 0.2
- Read the documentation
- Read the Technical Report (Accepted at CLEF 2010. Check back later for a new version.)

NEWS

New release: MIREX 0.2
Wed, 23 Jun 2010 18:39 CET
We released a version 0.2 that supports several standard information retrieval models...
Read more

Anchor text for ClueWeb09 Category A
Wed, 28 Apr 2010 9:11 CET
We've put anchor text for the English Category A documents of the TREC CLueWeb09 collection on line...
Read more

MIREX is sponsored by the Netherlands Organization for Scientific Research (NWO grant 639.022.809), and a Yahoo! faculty research grant.

ACKNOWLEDGEMENTS

- Maarten Fokkinga, Sietse ten Hoeve, Guido van der Zanden, and Michael Meijer
- Yahoo Research, Barcelona
- Netherlands Organization for Scientific Research (NWO), grant 639.022.809.