Index Construction

Introduction to Information Retrieval
INF 141/ CS 121
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org

Overview

- Introduction
- Hardware
- BSBI - Block sort-based indexing
- SPIMI - Single Pass in-memory indexing
- Distributed indexing
- Dynamic indexing
- Miscellaneous topics

Indices

The index has a list of vector space models

Letter from dead sister haunts brothers

Every time Julie Jensen's brothers hear the letter read, it brings everything back. Most of all, they wonder if they could have saved her. Her husband now stands trial for allegedly killing her. "I pray I'm wrong + nothing happens," Julie wrote days before her 1998 death. full story

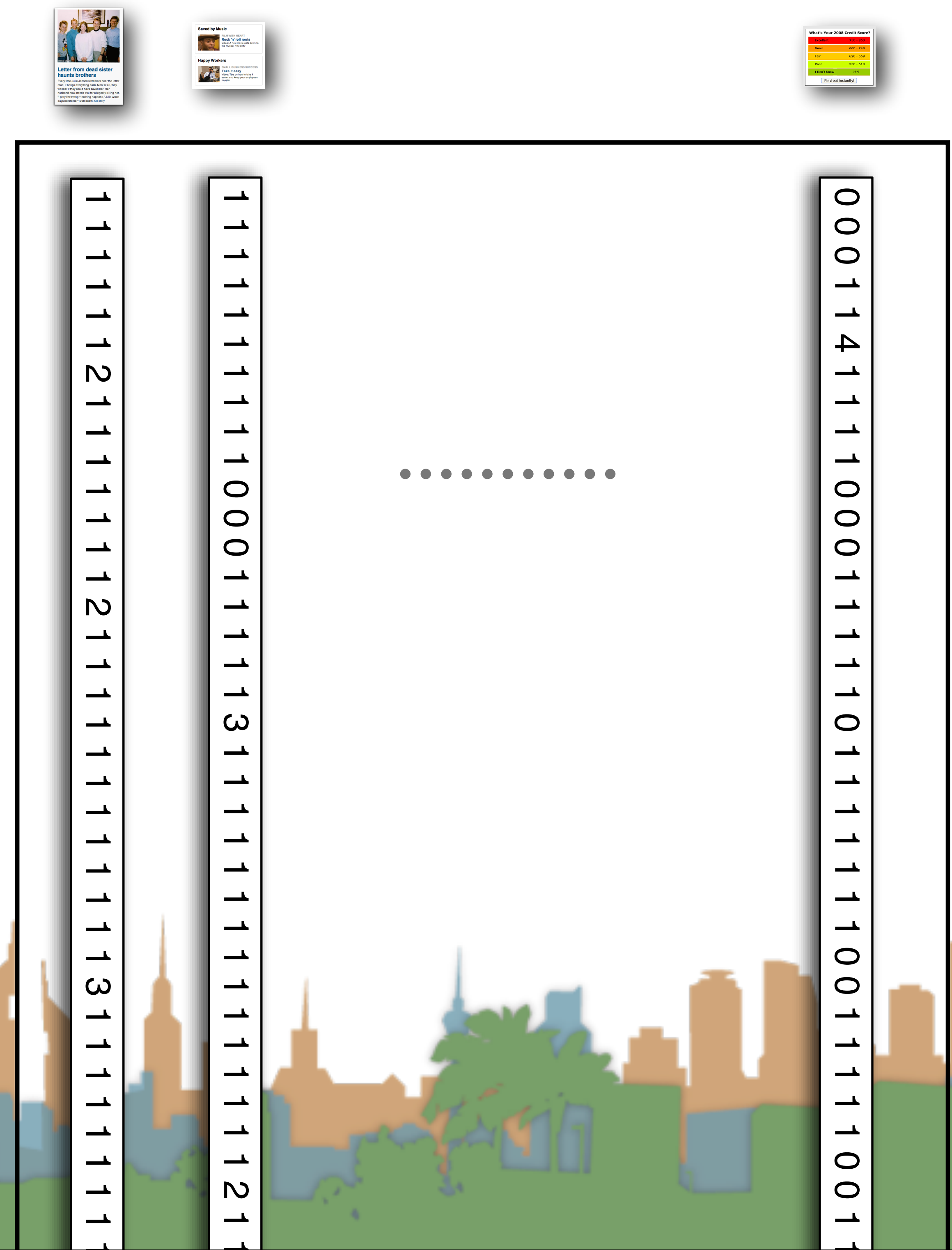
1 1998	
1 Every	1 have
1 Her	1 hear
1 I	3 her
1 I'm	1 husband
1 Jensen's	1 if
2 Julie	1 it
1 Letter	1 killing
1 Most	1 letter
1 all	1 nothing
1 allegedly	1 now
1 back	1 of
1 before	1 pray
1 brings	1 read,
2 brothers	1 saved
1 could	1 sister
1 days	1 stands
1 dead	1 story
1 death	1 the
1 everything	2 they
1 for	1 time
1 from	1 trial
1 full	1 wonder
1 happens	1 wrong
1 haunts	1 wrote

1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1

"Term-Document Matrix" Capture Keywords

A Column for Each Web Page (or "Document")

A Row For Each Word (or "Term")



- This picture is deceptive it is really very sparse
- Our queries are terms - not documents
- We need to "invert" the vector space model
- To make "postings"

Terms

- Inverted index
  - (Term, Document) pairs
  - building blocks for working with Term-Document Matrices
- Index construction (or indexing)
  - The process of building an inverted index from a corpus
- Indexer
  - The system architecture and algorithm that constructs the index

Indices

The index is built from term-document pairs

Letter from dead sister haunts brothers

Every time Julie Jensen's brothers hear the letter read, it brings everything back. Most of all, they wonder if they could have saved her. Her husband now stands trial for allegedly killing her. "I pray I'm wrong + nothing happens," Julie wrote days before her 1998 death. full story

(TERM,DOCUMENT)

(1998,www.cnn.com)
(Every,www.cnn.com)
(Her,www.cnn.com)
(I,www.cnn.com)
(I'm,www.cnn.com)
(Jensen's,www.cnn.com)
(Julie,www.cnn.com)
(Letter,www.cnn.com)
(Most,www.cnn.com)
(all,www.cnn.com)
(allegedly,www.cnn.com)
(back,www.cnn.com)
(before,www.cnn.com)
(brings,www.cnn.com)
(brothers,www.cnn.com)
(could,www.cnn.com)
(days,www.cnn.com)
(dead,www.cnn.com)
(death,www.cnn.com)
(everything,www.cnn.com)
(for,www.cnn.com)
(from,www.cnn.com)
(full,www.cnn.com)
(happens,www.cnn.com)
(haunts,www.cnn.com)
(have,www.cnn.com)
(hear,www.cnn.com)
(her,www.cnn.com)
(husband,www.cnn.com)
(if,www.cnn.com)
(it,www.cnn.com)
(killing,www.cnn.com)
(letter,www.cnn.com)
(nothing,www.cnn.com)
(now,www.cnn.com)
(of,www.cnn.com)
(pray,www.cnn.com)
(read,,www.cnn.com)
(saved,www.cnn.com)
(sister,www.cnn.com)
(stands,www.cnn.com)
(story,www.cnn.com)
(the,www.cnn.com)
(they,www.cnn.com)
(time,www.cnn.com)
(trial,www.cnn.com)
(wonder,www.cnn.com)
(wrong,www.cnn.com)
(wrote,www.cnn.com)

Indices

The index is built from term-document pairs

(TERM,DOCUMENT)
(1998,www.cnn.com)
(Every,www.cnn.com)
(Her,www.cnn.com)
(I,www.cnn.com)
(I'm,www.cnn.com)
(Jensen's,www.cnn.com)
(Julie,www.cnn.com)
(Letter,www.cnn.com)
(Most,www.cnn.com)
(all,www.cnn.com)
(allegedly,www.cnn.com)
(back,www.cnn.com)
(before,www.cnn.com)
(brings,www.cnn.com)
(brothers,www.cnn.com)
(could,www.cnn.com)
(days,www.cnn.com)
(dead,www.cnn.com)
(death,www.cnn.com)
(everything,www.cnn.com)
(for,www.cnn.com)
(from,www.cnn.com)
(full,www.cnn.com)
(happens,www.cnn.com)
(haunts,www.cnn.com)
(have,www.cnn.com)
(hear,www.cnn.com)
(her,www.cnn.com)
(husband,www.cnn.com)
(if,www.cnn.com)
(it,www.cnn.com)
(killing,www.cnn.com)
(letter,www.cnn.com)
(nothing,www.cnn.com)
(now,www.cnn.com)
(of,www.cnn.com)
(pray,www.cnn.com)
(read,,www.cnn.com)
(saved,www.cnn.com)
(sister,www.cnn.com)
(stands,www.cnn.com)
(story,www.cnn.com)
(the,www.cnn.com)
(they,www.cnn.com)
(time,www.cnn.com)
(trial,www.cnn.com)
(wonder,www.cnn.com)
(wrong,www.cnn.com)
(wrote,www.cnn.com)

- Core indexing step is to sort by terms

Term-document pairs make lists of postings

<u>(TERM,DOCUMENT, DOCUMENT, DOCUMENT, ...)</u>
(**1998**,www.cnn.com,news.google.com,news.bbc.co.uk)
(**Every**,www.cnn.com, news.bbc.co.uk)
(**Her**,www.cnn.com,news.google.com)
(**I**,www.cnn.com,www.weather.com,)
(**I'm**,www.cnn.com,www.wallstreetjournal.com)
(**Jensen's**,www.cnn.com)
(**Julie**,www.cnn.com)
(**Letter**,www.cnn.com)
(**Most**,www.cnn.com)
(**all**,www.cnn.com)
(**allegedly**,www.cnn.com)

- A posting is a list of all documents in which a term occurs.
- This is "inverted" from how documents naturally occur

Terms

- How do we construct an index?

Interactions

- An indexer needs raw text
  - We need crawlers to get the documents
  - We need APIs to get the documents from data stores
  - We need parsers (HTML, PDF, PowerPoint, etc.) to convert the documents
- Indexing the web means this has to be done web-scale

Construction

- Index construction in main memory is simple and fast.
- But:
  - As we build the index we parse docs one at a time
  - Final postings for a term are incomplete until the end.
  - At 10-12 postings per term, large collections demand a lot of space
  - Intermediate results must be stored on disk

Overview

- Introduction
- Hardware
- BSBI - Block sort-based indexing
- SPIMI - Single Pass in-memory indexing
- Distributed indexing
- Dynamic indexing
- Miscellaneous topics

System Parameters

- Disk seek time = 0.005 sec (2014: 0.004 hp - 0.015 mobile)
- Transfer time per byte = 0.00000002 sec
- Processor clock rate = 0.00000001 sec
- Size of main memory = several GB
- Size of disk space = several TB

http://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics

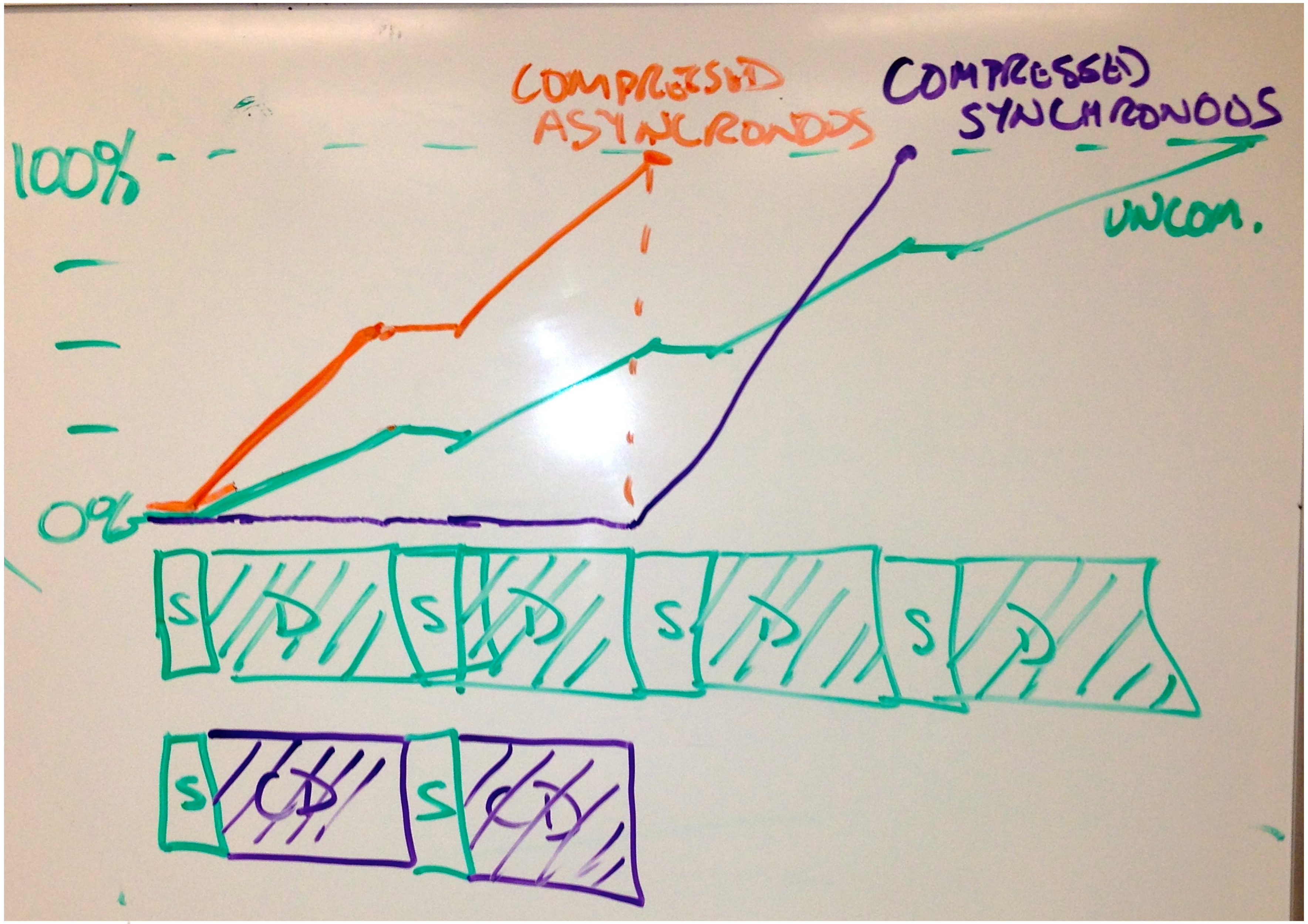
System Parameters

- Data is transferred from disk in blocks
  - Operating Systems read data in blocks, so
  - Reading one byte and reading one block take the same amount of time

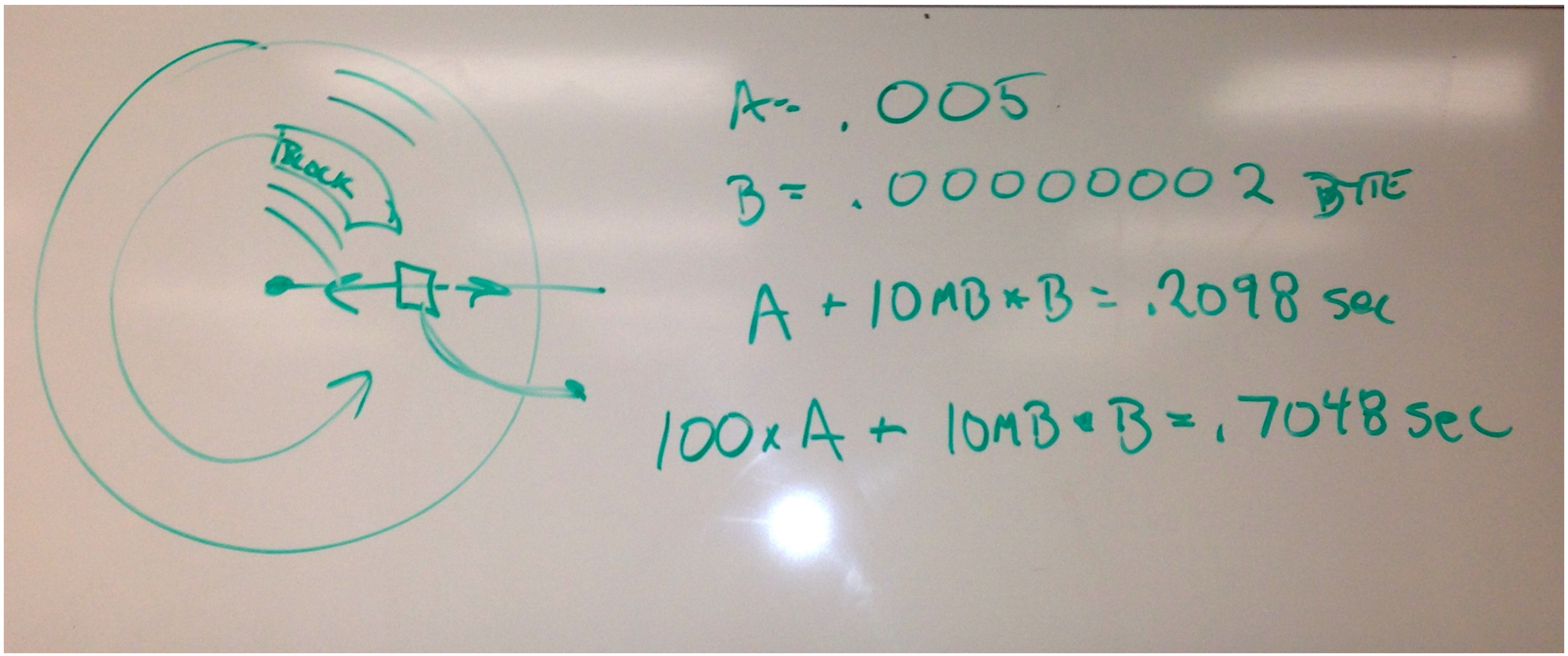
System Parameters

- Disk Seek Time
  - The amount of time to get the disk head to the data
  - About 10 times slower than memory access
  - We must utilize caching
  - No data is transferred during seek
- Data is transferred from disk in blocks
  - There is no additional overhead to read in an entire block
    - 0.2098 seconds to get 10 MB if it is one block
    - 0.7048 seconds to get 10 MB if it is stored in 100 blocks

System Parameters



System Parameters



System Parameters

- Data transfers are done on the system bus, not by the processor
  - The processor is not used during disk I/O
  - Assuming an efficient decompression algorithm
    - The total time of reading and then decompressing compressed data is usually less than reading uncompressed data.

Overview

- Introduction
- Hardware
- BSBI - Block sort-based indexing
- SPIMI - Single Pass in-memory indexing
- Distributed indexing
- Dynamic indexing
- Miscellaneous topics

Reuters collection example (approximate #'s)

- 800,000 documents from the Reuters news feed
- 200 terms per document
- 400,000 unique terms
- number of postings 100,000,000


REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enoug

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Reuters collection example (approximate #'s)

- Sorting 100,000,000 records on disk is too slow because of disk seek time.
  - Parse and build posting entries one at a time
  - Sort posting entries by term
    - Then by document in each term
  - Doing this with random disk seeks is too slow
  - e.g. If every comparison takes 2 disk seeks and N items need to be sorted with N log2(N) comparisons?
    - How long is that going to take?

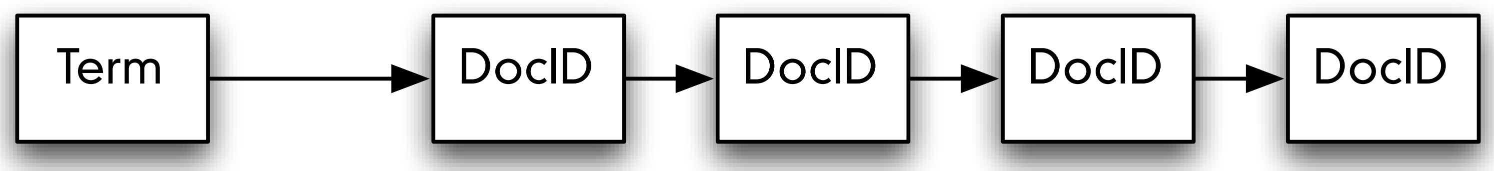
Reuters collection example (approximate #'s)

- 100,000,000 records
- Nlog2(N) is = 2,657,542,475.91 comparisons
- 2 disk seeks per comparison = 13,287,712.38 seconds x 2
- = 26,575,424.76 seconds
- = 442,923.75 minutes
- = 7,382.06 hours
- = 307.59 days
- = 84% of a year
- = 1% of your life

Index Construction

Review

- termID is an index given to a vocabulary word
  - e.g., “house” = 57820
- docID is an index given to a document
  - e.g., “news.bbc.co.uk” = 74291
- posting list is a data structure for the term-document matrix



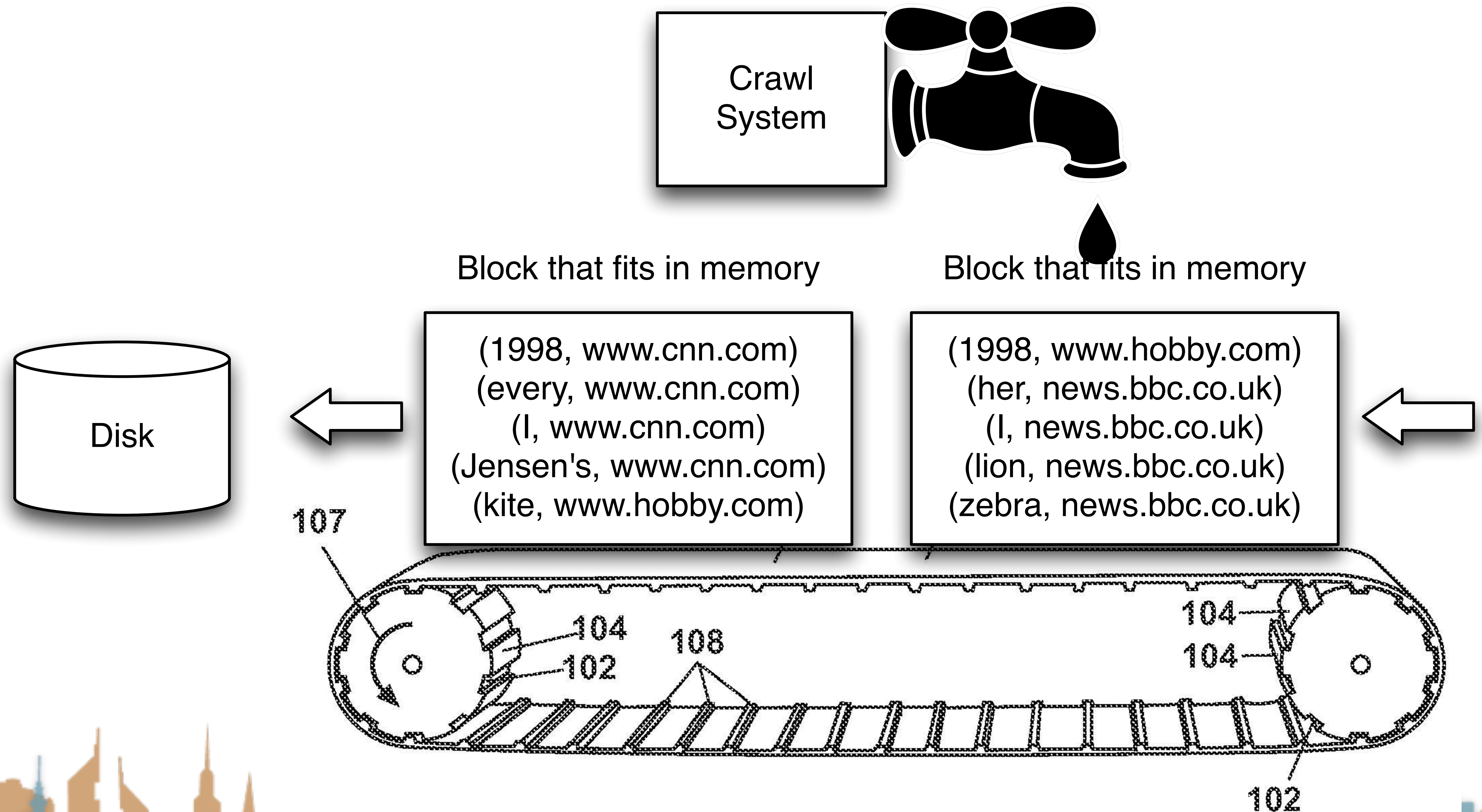
- posting list is an inverted data structure

BSBI - Block sort-based indexing

Different way to sort index

- 12-byte records (term, doc, meta-data)
- Need to sort T= 100,000,000 such 12-byte records by term
- Define a block to have 1,600,000 such records
    - can easily fit a couple blocks in memory
    - we will be working with 64 such blocks
- Accumulate postings for each block (real blocks are bigger)
- Sort each block
- Write to disk

Different way to sort index



Different way to sort index

Merged Postings

(1998, www.cnn.com, www.hobby.com)
(every, www.cnn.com)
(her, news.bbc.co.uk)
(I, www.cnn.com, news.bbc.co.uk
(Jensen's, www.cnn.com)
(kite, www.hobby.com)
(lion, news.bbc.co.uk)
(zebra, news.bbc.co.uk)

(1998, www.cnn.com)
(every, www.cnn.com)
(I, www.cnn.com)
(Jensen's, www.cnn.com)
(kite, www.hobby.com)

(1998, www.hobby.com)
(her, news.bbc.co.uk)
(I, news.bbc.co.uk)
(lion, news.bbc.co.uk)
(zebra, news.bbc.co.uk)

....

...

BSBI - Block sort-based indexing

BlockSortBasedIndexConstruction

$\textsc{BlockSortBasedIndexConstruction}()$

$n \leftarrow 0$

while $(\textit{all documents not processed})$

  do $\textit{block} \leftarrow \textsc{ParseNextBlock}()$

    $\text{BSBI-}\textsc{Invert}(\textit{block})$

    $\textsc{WriteBlockToDisk}(\textit{block}, f_n)$

$\textsc{MergeBlocks}(f_1, f_2 \ldots, f_n, f_{merged})$

Block merge indexing

- Parse documents into (TermID, DocID) pairs until “block” is full
- Invert the block
  - Sort the (TermID,DocID) pairs
  - Compile into TermID posting lists
- Write the block to disk
- Then merge all blocks into one large postings file
  - Need 2 copies of the data on disk (input then output)

Analysis of BSBI

- The dominant term is O(TlogT)
    - T is the number of TermID,DocID pairs
- But in practice ParseNextBlock takes the most time
- Then MergingBlocks
- Again, disk seeks times versus memory access times

Analysis of BSBI

- 12-byte records (term, doc, meta-data)
- Need to sort T= 100,000,000 such 12-byte records by term
- Define a block to have 1,600,000 such records
  - can easily fit a couple blocks in memory
  - we will be working with 64 such blocks
- 64 blocks * 1,600,000 records * 12 bytes = 1,228,800,000 bytes
- Nlog2N comparisons is 5,584,577,250.93
- 2 touches per comparison at memory speeds (10e-6 sec) =
  - 55,845.77 seconds = 930.76 min = 15.5 hours

Overview

- Introduction
- Hardware
- BSBI - Block sort-based indexing
- SPIMI - Single Pass in-memory indexing
- Distributed indexing
- Dynamic indexing
- Miscellaneous topics

SPIMI

- BSBI is good but,
  - it needs a data structure for mapping terms to termIDs
  - this won't fit in memory for big corpora
- Straightforward solution
  - dynamically create dictionaries
  - store the dictionaries with the blocks
  - integrate sorting and merging

Single-Pass In-Memory Indexing

```
SPIMI-Invert(tokenStream)
  outputFile ← NewFile()
  dictionary ← NewHash()
  while (free memory available)
      do token ← next(tokenStream)
          if term(token) ∉ dictionary
             then postingsList ← AddToDictionary(dictionary, term(token))
             else  postingsList ← GetPostingsList(dictionary, term(token))
          if full(postingsList)
             then postingsList ← DoublePostingsList(dictionary, term(token))
          AddToPostingsList(postingsList, docID(token))
  sortedTerms ← SortTerms(dictionary)
  WriteBlockToDisk(sortedTerms, dictionary, outputFile)
  return outputFile
```

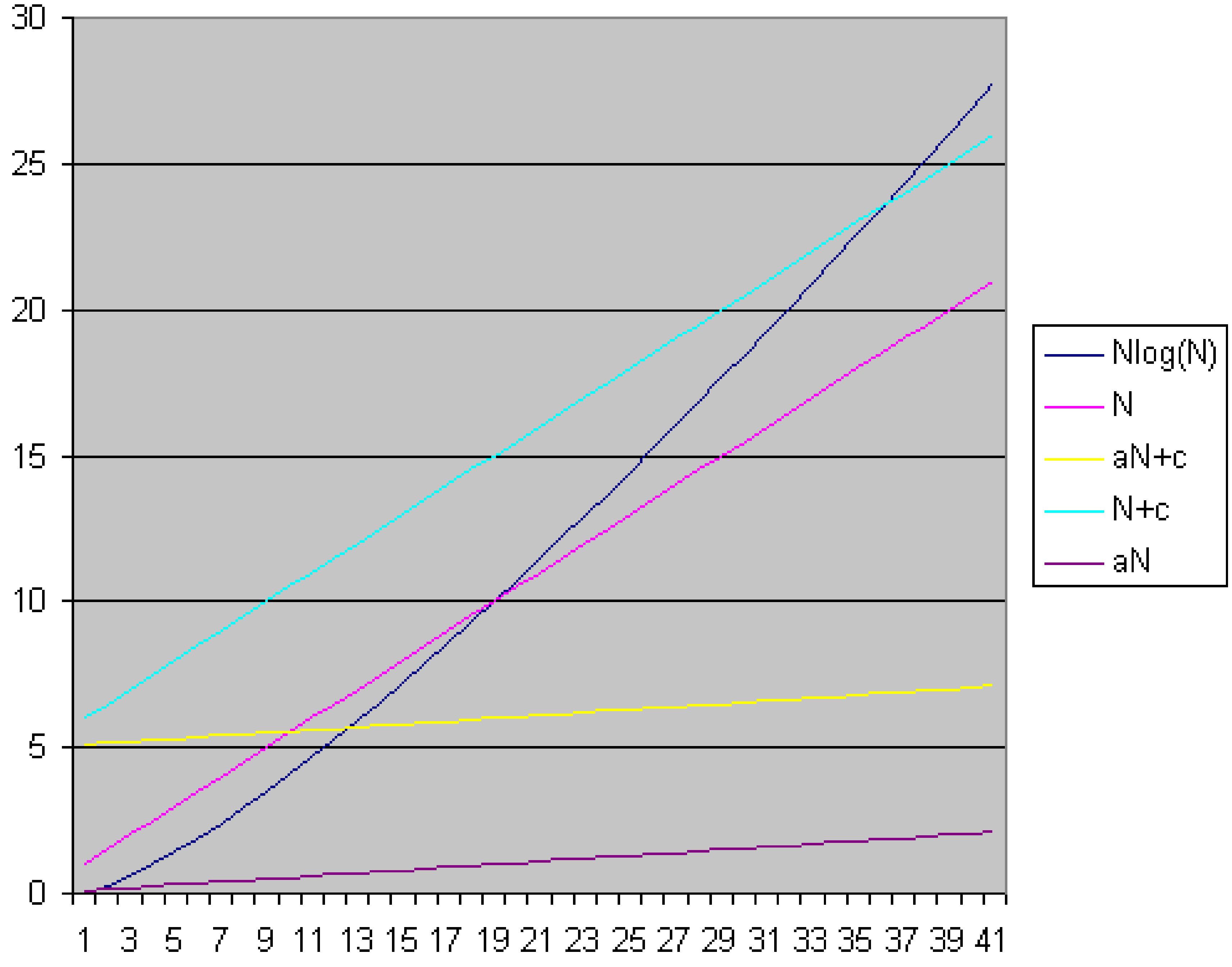
Single-Pass In-Memory Indexing

- So what is different here?
  - SPIMI adds postings directly to a posting list.
    - BSBI first collected (TermID,DocID pairs)
      - then sorted them
      - then aggregated the postings
  - Each posting list is dynamic so there is no term sorting
  - Saves memory because a term is only stored once
  - Complexity is O(T)
  - Compression enables bigger effective blocks

Large Scale Indexing

- Key decision in block merge indexing is block size
- In practice, spidering often interlaced with indexing
- Spidering bottlenecked by WAN speed and other factors

Single-Pass In-Memory Indexing



Overview

- Introduction
- Hardware
- BSBI - Block sort-based indexing
- SPIMI - Single Pass in-memory indexing
- Distributed indexing
- Dynamic indexing
- Miscellaneous topics

Review

- termID is an index given to a vocabulary word
  - e.g., “house” = 57820
- docID is an index given to a document
  - e.g., “news.bbc.co.uk” = 74291
- posting list is a data structure for the term-document matrix

Term → DocID → DocID → DocID → DocID

- posting list is an inverted data structure

Review

- BSBI and SPIMI
  - are single pass indexing algorithms
  - leverage fast memory vs slow disk speeds
  - for data sets that won’t fit in entirely in memory
  - for data sets that will fit on a single disk

Review

- BSBI
  - builds (termID, docID) pairs until a block is filled
  - builds a posting list in the final merge
  - requires a vocabulary mapping word to termID
- SPMI
  - builds posting lists until a block is filled
  - combines posting lists in the final merge
  - uses terms directly (not termIDs)

Index Construction

- What if your documents don’t fit on a single disk?
  - Web-scale indexing
    - Use a distributed computing cluster
    - supported by “Cloud computing” companies

End of Chapter 4

LUCI