Optimising Xapian

Richard Boulton

UKUUG, Birmingham
9th August 2009

Xapian is?

- Search engine library
- 10 years old
- GPL
- http://xapian.org/

Not going to talk about...

- Scaling across multiple machines
  - Big topic – go to a cloud talk.
- Optimising ranking of results
  - Huge topic – go to an IR conference!
- Optimising specific installations
  - Filesystems, hardware specification, SSDs etc

Making the most of hardware

- Single machine
- Limited memory
- Database on a slow disk
- Back in the 90s?
- … or just dealing with modern data volumes?

Two types of optimisation

- Picking the right algorithms
- Implementing them well

Requirements

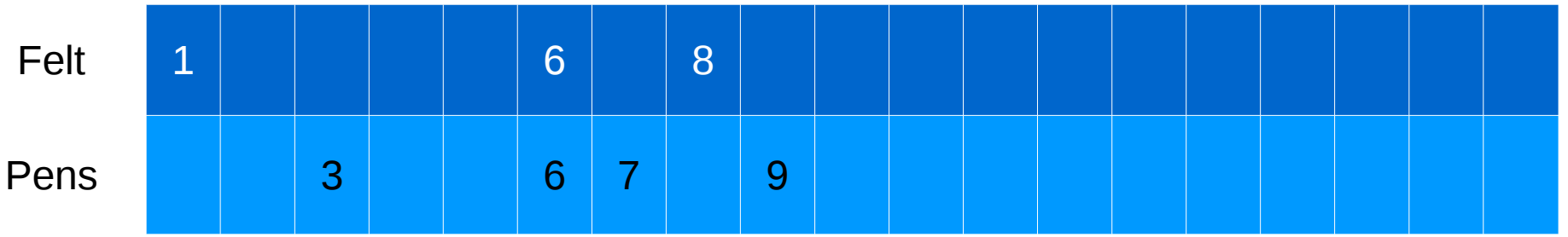
- Given a set of documents
  - Described by a set of terms and frequencies
- And a set of queries
  - Described by terms, frequencies and operators
- Find the documents which best match the query

Analysing the problem

- Do as much work at indexing time as possible
- Precalculated searches?
- Can't precalculate everything...
- Calculate all single-term queries

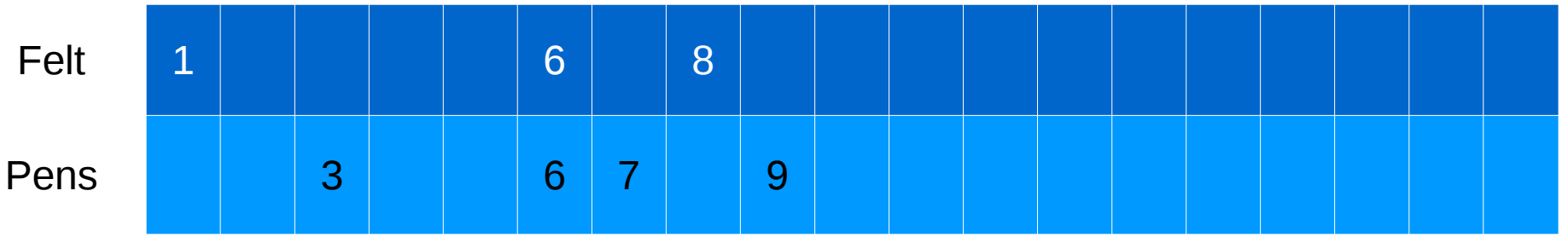
Stored data

- Posting lists:



Single term search

- Read a posting list
- Remember the best



AND search

- Naive approach:
  - Read first list.
  - Hold it in memory:

1	
6	
8	

  - Read next list
  - Merge it in:
  - Select the best

1	
3	
6	
7	
8	
9	

AND search

- Problem – limited by amount of memory
- Problem – no way to avoid reading all of the list

Better AND search

- Read lists in parallel
- Start with the shortest
- Jump forward in second list to keep up with first
- Keep only the best N items

OR search

- Read lists in parallel
- But can't skip items like with AND

- So … make it into an AND
- How?

OR search

- ASSUMPTION: we only want the top few results
- So, keep track of only those
- And, keep track of the lowest weight of those
- Also, calculate upper bound on weight of each term
- When both upper bounds < lowest weight, we need both, so become an AND

Taking it further

- Can apply this idea across whole query tree
- Can introduce other operators – AND_MAYBE
- Phrase queries
  - AND, followed by checking positions
  - Or, store pairs of adjacent terms, and then check positions
  - Or, store certain pairs...

Implementation

- … not a small job
- Datastructures
- Compression techniques
- Micro-optimisations

Datastructures

- Assumption – will have too much data to fit it all in memory
- Disks are slow
- But faster when reading in chunks
- B+-trees – traditional but good
  - Block structured, massively branching tree – very shallow

Posting list chunks

- Store posting lists in chunks, so we can get straight to right chunk for docid we're interested in
- Work out what statistics to store, where
- Get tighter bounds on possible weights, so we can skip better

Document length

- Needed for weight calculation
- Store it in each posting list – duplicated, but no side lookup
- Or store it only once?
- Currently, we store it in all posting lists
- New backend stores it only once → 40% smaller!
- But, currently 10 times slower :(

New problems

- We often have enough memory these days!
- 500M = A huge collection 10 years ago, now only medium
- 10M = A large collection 10 years ago, now small – will often fit fully in memory

=> IO less of a bottleneck – optimise CPU

New problems

- Faceted search
  - Display information about all the items in the result set
  - => Have to calculate all the result set!
  - Or – approximate
  - Or – precalculate the facet values somehow

New problems

- Bias results with external weights
- Page rank / product rank
- Fixed weights – so store documents in decreasing weight order – lets us finish early
- But – harder to update dynamically

Geolocation

- Bias results by distance from a location
- Generate hierarchies of terms
  - HTM easiest way to implement
- Use to restrict candidates
- Combine candidates with dynamically calculated weight

Image similarity

- Terms representing features
- Queries with hundreds of terms
- Current optimisations help
- … but distribution of frequencies and weights is less amenable to early termination.

Variety

- Strict relevance order leads to duplication
  - Similar items get similar scores
- Usually want to present a selection of results
- Order based on combination of novelty and relevance
- Score depends on earlier documents

  => our early termination doesn't work

Questions

Xapian: http://xapian.org/
Me: richard@flax.co.uk