Performance optimization with Lucene 4

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Who am I?

- Lucene Core Committer & PMC Member
- Co-Founder ElasticSearch Inc.
- Co-Founder BerlinBuzzwords
- Twitter: @s1m0nw
- simonw@apache.org
- simon.willnauer@elasticsearch.com

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Why are you here?

- You are Lucene Expert and curious what you can do timorrow? - **Check!**
- You are curious how Lucene can even better that what we already have? - **Check!**
- You are an IR - Researcher and need more ways to do crazy shit? - Check!
- Every CPU cycle counts, ah one of those? - **Check!**
- You are curious how to gain a better user experience? - **Check!**

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

What is performance?

- Better search quality? - Precision / Recall etc.?
- Faster query times?
- Less RAM usage?
- Less Disk usage?
- Higher concurrency?
- Less Garbage to collect?
- An excuse to justify to work on cool things? ;)

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Here is the answer...

- As usual, it depends!
  - Figure out what are your bottlenecks!
  - Benchmark and make your results repeatable!
  - 10x faster than crazy fast is still crazy fast!
- If you are in doubt:
  - Reduce the variables in you benchmark!
  - You can still tune just for the sake of it!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Lucene 4.0

Flexibility, Speed & Efficiency

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Release Notes snapshot...

- Pluggable Codecs
- Per Document Values (DocValues)
- Concurrent Flushing
- Multiple Scoring Models - flexible ranking
- New Term Dictionary
- From UTF-16 to UTF-8
  - no string objects anymore!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Concurrent Flushing

aka. DocumentsWriterPerThread (DWPT)

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Writing Documents in Lucene 3.x

IndexWriter

DocumentsWriter

Thread State | Thread State | Thread State | Thread State | Thread State

merge segments in memory

Merge on flush

Flush to Disk

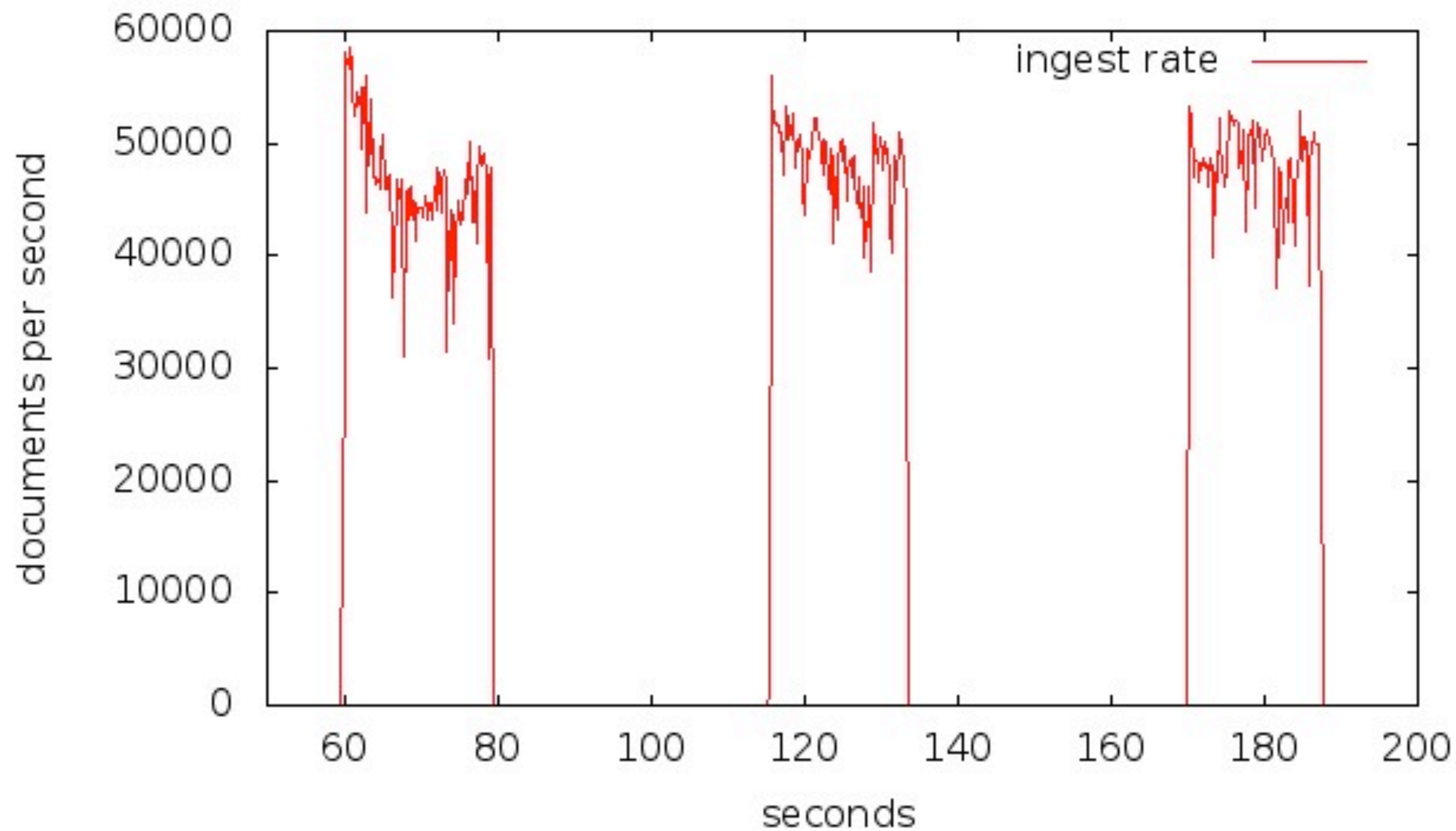
Directory

Multi-Threaded

Single-Threaded

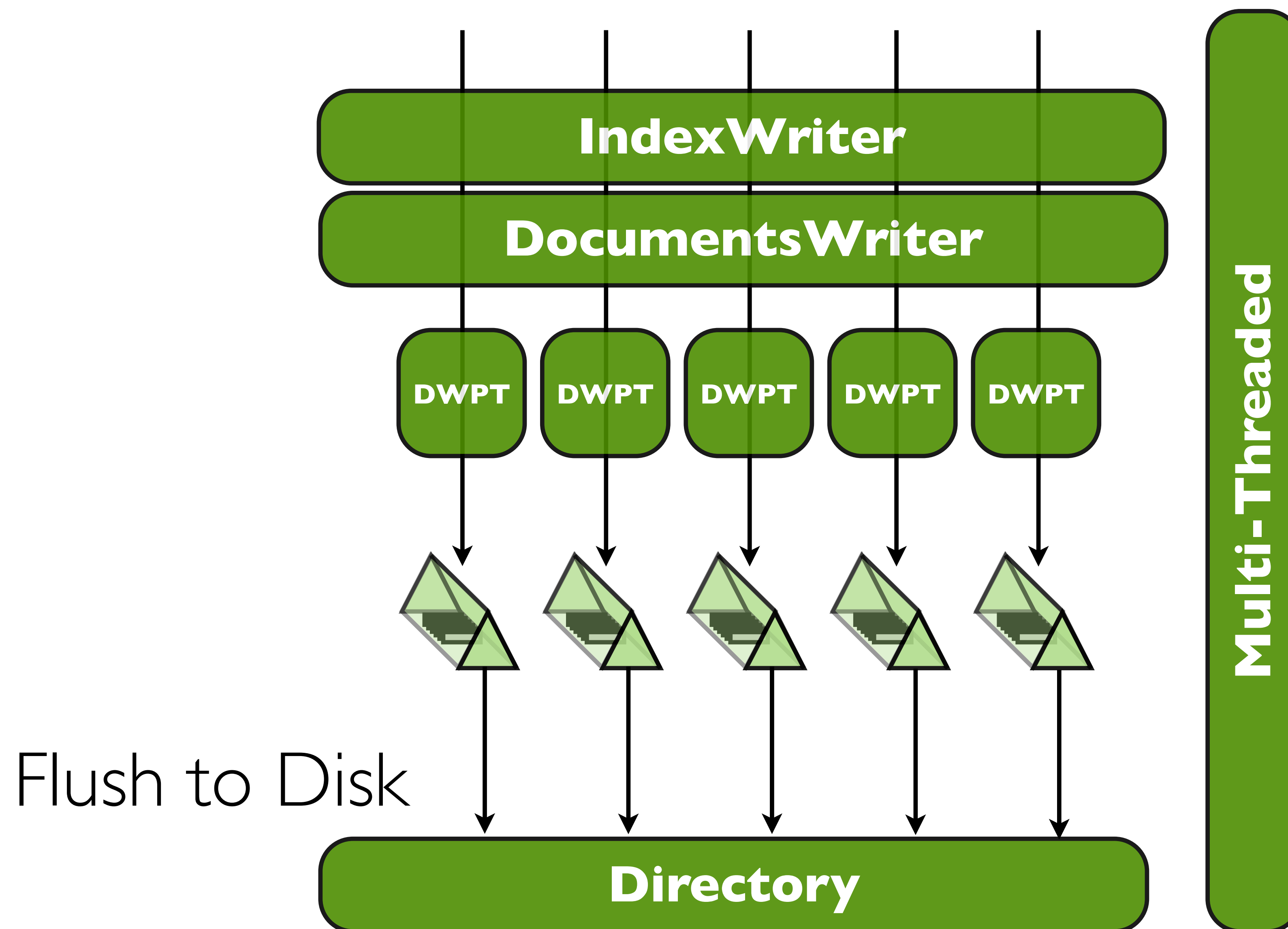
Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

A benchmark (10M English Wikipedia)



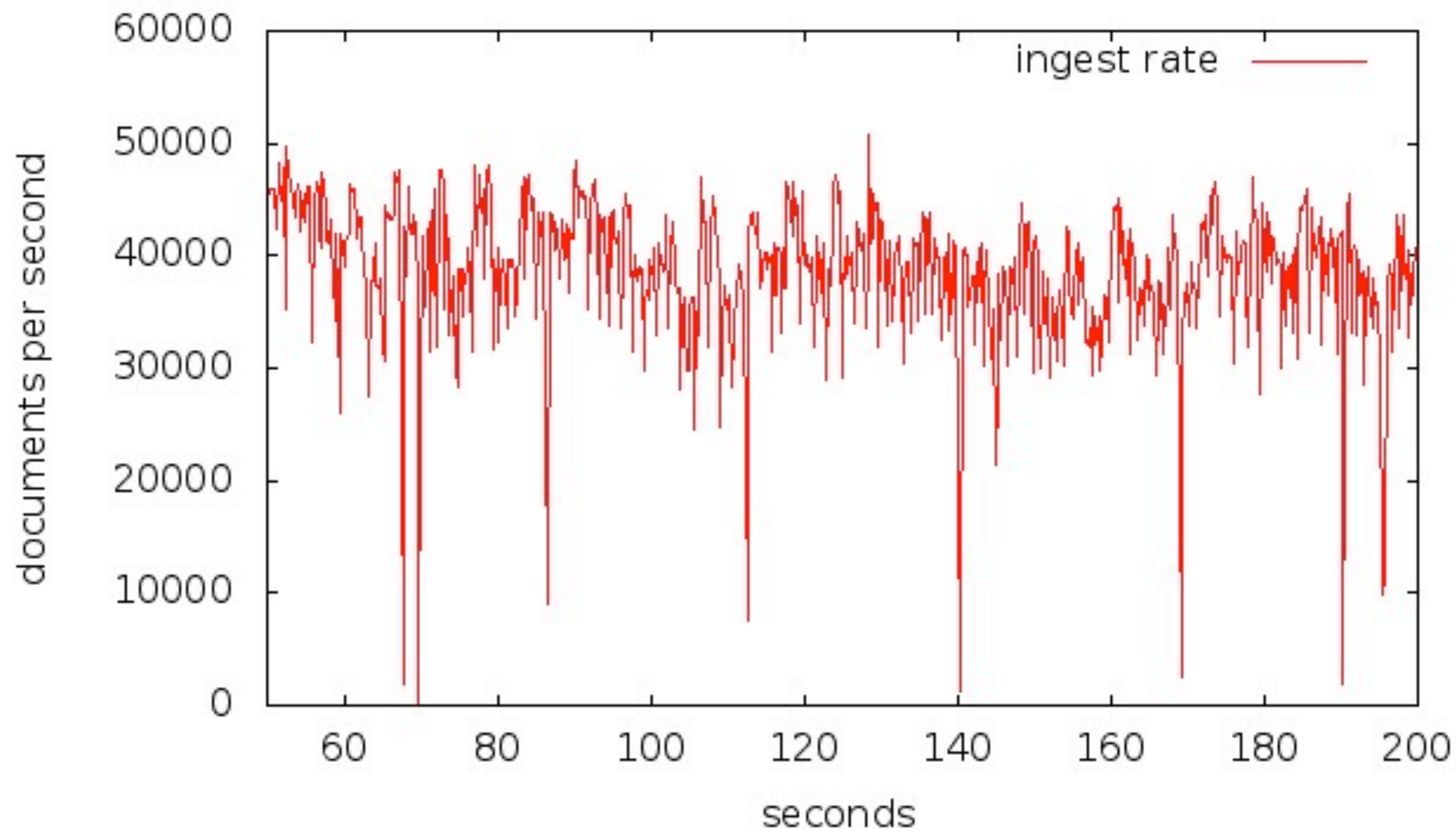
Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Concurrent Flushing in 4.0



Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

The same Benchmark...



Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

The improvement...

~4 KB Wikipedia English docs

Plain text GB/hour

280
260
240
220
200
180
160
140
120
100
80

Jul 11 Oct 11 Jan 12 Apr 12 Jul 12 Oct 12

Date

Reduced RAM buffer (ramBufferSizeMB) from 512MB to 320MB
Increased the # of threads from 6 to 20

Committed Concurrent Flushing

http://people.apache.org/~mikemccand/lucenebench/indexing.html

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Concurrent Flushing

- Indexing can gain a lot if hardware is concurrent
  - wait free flushing and indexing
- less RAM might increase your throughput
  - maximizing the IO utilization
- Concurrent Flushing can “hammer” your machine
  - if ssh doesn’t respond - it’s DWPT
- More segments are created ie. more merging
- Tune carefully if you index in to search machines
  - you can easily kill you IO cache - 1 indexing thread might be enough!
  - adjust # thread states and the RAM buffer

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

DocValues

aka. Column Stride Fields

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

You Know FieldCache?

Lucene can un-invert a field into FieldCache

weight
5.8
1.0
2.7
2.7
4.3
7.9
1.0
3.2
4.7
7.9
9.0

parse

convert to datatype

un-invert

array per field / segment

float 32

term	freq	Posting list
1.0	1	1 6
2.7	**1**	**2 3**
3.2	1	7
4.3	1	4
4.7	1	8
5.8	1	0
7.9	1	5 9
9.0	1	10

UTF-8 bytes

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

The problem...

- Uninverting is heavy (CPU & IO)
- Creates potentially lots of garbage
- Required to be in JVM Memory
- NRT suffers on Re-Open
- Warming Queries take forever
- Unnecessary type conversion
- All fields are always sorted!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

The solution...

Once column per field and segment

One value per document

field: time	field: id	field: page_rank
1288271631431	1	3.2
1288271631531	5	4.5
1288271631631	3	2.3
1288271631732	4	4.44
1288271631832	6	6.7
1288271631932	9	7.8
1288271632032	8	9.9
1288271632132	7	10.1
1288271632233	12	11.0
1288271632333	14	33.1
1288271632433	22	0.2
1288271632533	32	1.4
1288271632637	100	55.6
1288271632737	33	2.2
1288271632838	34	7.5
1288271632938	35	3.2
1288271633038	36	3.4
1288271633138	37	5.6
1288271632333	38	45.0

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

DocValues

- No Uninverting
- Compact In-Memory representation
- Fast Loading (~10x faster than FC for a float field)
- Strong typed (int, long, float, double, bytes)
- Sorted if necessary
- On-Disc access via same interface
- Possible on any field
- One Value per Document & Field

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Usecases

- Sorting
- Grouping
- Faceting
- Scoring (Norms & Document Boosting)
- Key / Value Lookups
- Persisted Filters
- Geo-Search

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Flexible Scoring

Similarity & Friends

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Lucene 3.x

- Vector-Space Model (TF-IDF) and that's it
- Hard to extend
- Insufficient index statistics (avg. field length)
- Global model and not per-field

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Lucene 4.0

- Added Per-Field Similarity
- Score-calculation is private to the similarity
- Lots of new index statistics
  - total term frequency
  - sum document frequency
  - sum total term frequency
  - doc count per field
- Norms are DocValues ie. not bound to single byte!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

New Scoring models

- Okapi BM-25 Model
- Language Models
- Information Based Models
- Divergence from Randomness
- Yours goes here....

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Codecs

aka. Pluggable Index Formats

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Lucene 3.6

- One index format
- Impossible to extend without forking Lucene
- Improvements hardly possible
  - Backwards Compatibility
  - Tight coupled Reader and Writer
- Even experiments required massive internal Lucene knowledge

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Lucene 4.0

- Introduced a Codec Layer
  - a common interface providing access to low-level data-structures
  - all read and write operations & format are private to the codec
  - fully customizable
  - Postings, Term-Dictionary, DocValues, Norms are per field

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

What does this buy us?

- Data-Structures tailored to a specific usecase
  - Wanna read you document backwards - do it!
  - Wanna keep every term in memory - do it!
  - Wanna use a B-Tree instead of a FST - do it!
  - Wanna use a Bloom Filter on top - do it!
- Lucene gave up control over all low level data-structures
- Lots of different implementations shipped with Lucene 4

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Available codecs / formats?

- Pulsing Postings Format
  - Inlines postings into the term dictionary
- Bloom Postings Format
  - Uses a bloom filter to speed up term lookups
  - Helps with NRT on ID fields to speed up deleting docs
- Block Postings Format
  - uses state of the art block compression
  - new default in Lucene 4.1
  - speeds up queries if positions are present but not used

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Available codecs / formats?

- Block Tree Term Index (default Lucene 4.0)
  - reduces memory footprint 30x less
  - massive lookup speed improvements
- Simple Text Postings Format
  - helpful for debugging
  - writes everything as plain text
- Memory Postings Format
  - holds everything in memory
  - 1Million Key-Value lookups / second

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Not just postings

- Compressed Stored Fields
  - Will come with Lucene 4.1
  - Uses LZ4 Compression
- Everything we write is exposed via Codec
  - DocValues - have your own format
  - Norms (Essentially DocValues)
  - Delete Documents
  - Term Vectors
  - Segment Level information

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Encourage Researchers

- Good idea for postings compression?
  - write a postings format!
- Lucene offers a lot now on the lowest level!
  - you like bits and bytes - help us to improve!
- Try - Measure - Improve!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Wrapping up

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

What is left?

- ...if I had more time...
- Improved Filter execution up to 500% faster
- Automaton Queries
  - Fast Regular Expression Query
  - FuzzyQuery is 100x to 200x faster than in 3.x
- Term offsets in the index
- New Spellcheckers and Query Suggesters
- Many more... talk to me if you are curious!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

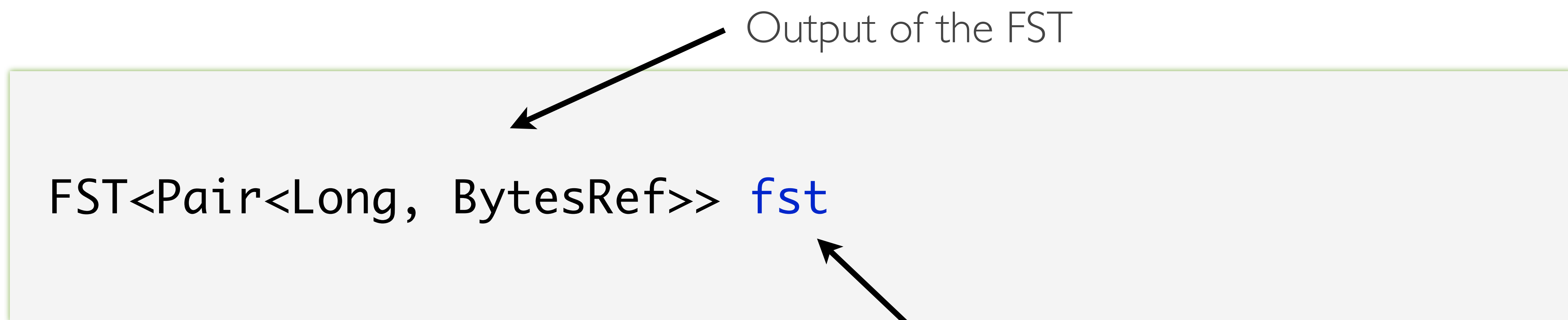
The end...

Thank You!

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

Backup Slides... Finite State Tranducers

- Check out our FST Package
  - Highly memory efficient and Fast Finite State Transducer
  - Excellent for fast key / value lookups
  - Suggesters / TermDictionaries / Analyzers use it



Input is a Int 32 sequence (UTF-32) and output a Long / Bytes pair

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.

elasticsearch.

Backup Slides...Automatons

- Check out the Automaton Package
  - Flexible query creation
  - Combine Levenshtein Automaton other Automatons

```
// a term representative of the query, containing the field.
// term text is not important and only used for toString() and such
Term term = new Term("body", "dogs~1");

// builds a DFA for all strings within an edit distance of 2 from "bla"
Automaton fuzzy = new LevenshteinAutomata("dogs").toAutomaton(1);

// concatenate this with another DFA equivalent to the "*" operator
Automaton fuzzyPrefix = BasicOperations.concatenate(fuzzy, BasicAutomata
    .makeAnyString());

// build a query, search with it to get results.
AutomatonQuery query = new AutomatonQuery(term, fuzzyPrefix);
```

Copyright Elasticsearch 2012. Copying, publishing and distributing without written permission is strictly prohibited.