Apache Lucene and Solr 8: What's coming next?

Uwe Schindler
SD DataSolutions GmbH / Apache Software Foundation

thetaph1 – https://www.thetaphi.de
My Background

• **Committer** and **PMC member** of **Apache Lucene and Solr** - main focus is on development of Lucene Core.
• Implemented fast numerical search and maintaining the new attribute-based text analysis API. Well known as *Generics and Sophisticated Backwards Compatibility* 🤖.
• **Elasticsearch** lover.
• Working as consultant and software architect at **SD DataSolutions GmbH** in Bremen, Germany.
• Maintaining **PANGAEA** (Data Publisher for Earth & Environmental Science) where I implemented the portal's geo-spatial retrieval functions with Apache Lucene Core and Elasticsearch.
Lucene 8: When?

• Expected release date:

  As always: no comment! (but few weeks is likely)

• Release branch (branch_8x) was cut mid-January
10 times faster queries...

New features and changes in Apache Lucene 8
“The” Change

• **New result collection engine**
  – Allows short circuit if total count is not needed

• **Works for combinations of many query types:**
  – TermQuery
  – BooleanQuery: disjunctions
  – PhraseQuery
  – ConstantScoreQuery
How does it work?

• Add some information about maximum TF and norm to posting list blocks (e.g., 64 postings or larger)
• Multi-Level: same stats for block of blocks!
• Stored in already existing “Skip List”
How does it work?

Faster top-k document retrieval using block-max indexes. SIGIR '11 Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, Pages 993-1002, https://doi.org/10.1145/2009916.2010048
How does it work?

• Add some information about maximum TF and norm to posting list blocks (e.g., 64 postings or larger)
• **Multi-Level**: same stats for block of blocks!
• Stored in already existing “Skip List”
What’s a skip list?
What’s a skip list?

Lucene
3  7  8  15  16  19
33 49 51 56

Search
4  5  7  12  15  16
46 47 49
What’s a skip list?

Lucene:
- \(33\) → \(15\) (\(TF_{max}=3\)) → \(33\)
- \(3\) → \(7\) → \(8\) → \(15\) → \(16\) → \(19\)
- \(33\) → \(49\) → \(51\) → \(56\)

Search:
- \(46\) → \(12\) (\(TF_{max}=1\)) → \(46\)
- \(4\) → \(5\) → \(7\) → \(12\) → \(15\) → \(16\)
- \(46\) → \(47\) → \(49\)
“Super-speedy scoring in Lucene 8”

Talk by “@romseygeek” (Alan Woodward) after this one!
New Field and Query Types

• **FeatureField**
  – Encodes scoring value in TF
  – Allows to use **BlockMax** algorithms!

• **LongPoint**#newDistanceFeatureQuery

• **LatLonPoint**#newDistanceFeatureQuery
New Field and Query Types

• **FeatureField**
  - Encodes scoring value in TF
  - Allows to use **BlockMax** algorithms!

• **LongPoint#**

• **LatLonPoint#**
New IntervalQuery aka “Spans”

• Complete reimplementation of SpanQuery hierarchy of classes
• Single Query: An IntervalQuery takes a field name and an IntervalsSource, and matches all documents that contain intervals defined by the source in that field.
Possible IntervalSources
provided by Intervals factory

- **term** — Represents a single term
- **phrase** — Represents a phrase
- **ordered** — Represents an interval over an ordered set of terms or intervals
- **unordered** — Represents an interval over an unordered set of terms or intervals
- **or** — Represents the disjunction of a set of terms or intervals
- **maxwidth** — Filters out intervals that are larger than a set width
- **containedBy** — Returns intervals that are contained by another interval
- **notContainedBy** — Returns intervals that are *not* contained by another interval
- **containing** — Returns intervals that contain another interval
- **notContaining** — Returns intervals that do *not* contain another interval
- **nonOverlapping** — Returns intervals that do *not* overlap with another interval
- **notWithin** — Returns intervals that do *not* appear within a set number of positions of another interval
Possible IntervalSources provided by Intervals factory

- **term** — Represents a single term
- **phrase** — Represents a phrase

```java
Query q = new IntervalQuery(field,
    Intervals.ordered(
        Intervals.term("lucene"),
        Intervals.maxwidth(3, Intervals.ordered(Intervals.term("foo"), Intervals.term("bar"))));
```

- **containedBy** — Returns intervals that are contained by another interval
- **notContainedBy** — Returns intervals that are *not* contained by another interval
- **containing** — Returns intervals that contain another interval
- **notContaining** — Returns intervals that do *not* contain another interval
- **nonOverlapping** — Returns intervals that do *not* overlap with another interval
- **notWithin** — Returns intervals that do *not* appear within a set number of positions of another interval
ByteBufferDirectory

• **Replacement** for non-scaleable RAMDirectory
  – Broken concurrency
  – Millions of small `byte[8192]` arrays

• Shares backing infrastructure with MMapDirectory
  – Allocates ByteBuffer (possibly off-heap!)
Index Format Improvements

• **BlockMax** statistics in Skip Lists
  – Speeds up disjunctions

• **Jump tables** for DocValues
  – DocValues based queries now allow to jump to later doc ids with $O(1)$
HOW TO MIGRATE?
Lucene 7: Index Version Enforcement

Lucene stores version that created index

– Each segment records lowest version that contributed to it during merge
– Preserved during merges or index upgrades
Lucene 7: Index Version Enforcement (2)

- Better detection of no longer supported features
  - Broken offset detection by default enabled for new indexes
- New norms data type!
Lucene 8: "Anti-Feature"

Removal of Lucene 6 index support!

• Get rid of old index segments?!: IndexUpgrader no longer helps!
• Elasticsearch supports reindexing old indexes during migration!
Lucene 8: "Anti-Feature"

If you need a hack when updating ancient indexes:

Contact me!

(there are ways to do this, but you will lose correct scoring)
Going forward...

New features and changes in Apache Solr 8
HTTP/2

- Solr nodes can now listen and serve HTTP/2 requests. Most of internal requests use Http2SolrClient.
- Internal requests are sent by using HTTP/2, Solr 8.0 nodes can't talk to old nodes (7.x).
HTTP/2: How to migrate

• Do rolling updates as normally, but the Solr 8.0 nodes must start with `-Dsolr.http1=true` as startup parameter. By using this parameter, internal requests are sent by using HTTP/1.1.

• When all nodes are upgraded to 8.0, restart them, this time `-Dsolr.http1` parameter should be removed.
Support for HTTP/2 with TLS enabled:

- Requirement: **Java 9+**
- **Solr on Java 8** automatically *disables* HTTP/2 support if TLS is enabled!
BM25 changes

• Lucene 8 has **simplified BM25F** compatible scoring
• Absolute scores are *lower*!
• **Sort order will not change** in normal cases
• **Solr:** If schema match version < 8, legacy scoring is used
Lucene/Solr: Minimum Java Version
Current state

• Requirement: **Java 8 as minimum version**
• **Apache Lucene** works flawless with Java 9, 10, 11 => Faster!
• **Apache Solr** has minor problems:
  – Hadoop integration *(fix coming)*
  – Kerberos Authentication *(fix coming)*
  – **HTTP/2 with TLS** requires Java 9+
Support for Java 9+

• Performance improvements in compression
  – LZ4 (stored fields)
• More bounds checks in API
  – No slowdown with Java 9+ due to intrinsics

Lucene’s JAR files are MR-JARs!
Support for Java 9+

- Performance improvements in compression
  - LZ4 (stored fields)
- More bounds checks
  - No slowdown with Java 9+

Lucene’s JAR files are MR-JARs!
Java 8 / 9 / 10 / 11

• No more Java 9 or 10 releases (EOL)
• Oracle Java 8 had LTS support till 3 days ago, now EOL!
• Ubuntu has LTS support for Java 8 and 11
• AdoptOpenJDK has LTS releases for 8 and 11
Future

• **Lucene Master branch (9.0)** likely to switch to **Java 11** in near future!
• Lucene / Solr **8 stays on** Java 8, but full support for later versions with **MR-JAR** feature!
• **Recommendation:** Use Java 11 LTS (AdoptOpenJDK) in production!
THANK YOU!

Questions?