Apache Lucene - Overview

Table of contents

1 Apache Lucene

Apache Lucene(TM) is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

Apache Lucene is an open source project available for free download. Please use the links on the left to access Lucene.

2 The Apache Software Foundation

The Apache Software Foundation provides support for the Apache community of open-source software projects. The Apache projects are defined by collaborative consensus based processes, an open, pragmatic software license and a desire to create high quality software that leads the way in its field. Apache Lucene, Apache Solr, Apache PyLucene, Apache Open Relevance Project and their respective logos are trademarks of The Apache Software Foundation. All other marks mentioned may be trademarks or registered trademarks of their respective owners.

3 Lucene News

3.1 27 November 2011 - Lucene Core 3.5.0

The Lucene PMC is pleased to announce the availability of Apache Lucene 3.5.0.

Lucene can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/. Maven artifacts are available here.

See CHANGES and contrib CHANGES for details.

Highlights of the Lucene release include:

- Added a very substantial (3-5X) RAM reduction required to hold the terms index on opening an IndexReader. (LUCENE-2205)
- Added IndexSearcher.searchAfter which returns results after a specified ScoreDoc (e.g. last document on the previous page) to support deep paging use cases. (LUCENE-2215)
- Added SearcherManager to manage sharing and reopening IndexSearchers across multiple search threads. Underlying IndexReader instances are safely closed if not referenced anymore. (LUCENE-3445, LUCENE-3558)
- Added SearcherLifetimeManager which safely provides a consistent view of the index across multiple requests (e.g. paging/drilldown). (LUCENE-3558, LUCENE-3486)
- Renamed IndexWriter.optimize to forceMerge to discourage use of this method since it is horribly costly and rarely justified anymore. (LUCENE-3454)

- Added NGramPhraseQuery that speeds up phrase queries 30-50% when n-gram analysis is used. (LUCENE-3426)
- Added a new reopen API (IndexReader.openIfChanged) that returns null instead of the old reader if there are no changes in the index. (LUCENE-3464)
- Improvements to vector highlighting: support for more queries such as wildcards and boundary analysis for generated snippets. (LUCENE-1824, LUCENE-1889)
- IndexSearcher and IndexReader now perform additional checks to throw AlreadyClosedExceptions if searches are performed on a closed IndexReader. Performing searches on already closed reader can cause JVM crashes when invalid memory mapped files are referenced.
- Several bugfixes, including a bug where closing an NRT reader after the writer was closed was incorrectly invoking the DeletionPolicy. See CHANGES.txt entries for full details.

3.2 26 October 2011 - Java 7u1 fixes index corruption and crash bugs in Apache Lucene Core and Apache Solr

Oracle released Java 7u1 on October 19. According to the release notes and tests done by the Lucene committers, all bugs reported on July 28 are fixed in this release, so code using Porter stemmer no longer crashes with `SIGSEGV`. We were not able to experience any index corruption anymore, so it is safe to use Java 7u1 with Lucene Core and Solr.

On the same day, Oracle released Java 6u29 fixing the same problems occurring with Java 6, if the JVM switches `-XX:+AggressiveOpts` or `-XX:+OptimizeStringConcat` were used. Of course, you should **not** use experimental JVM options like `-XX:+AggressiveOpts` in production environments! We recommend everybody to upgrade to this latest version 6u29.

In case you upgrade to Java 7, remember that you may have to reindex, as the unicode version shipped with Java 7 changed and tokenization behaves differently (e.g. lowercasing). For more information, read `JRE_VERSION_MIGRATION.txt` in your distribution package!

3.3 14 September 2011 - Lucene Core 3.4.0

The Lucene PMC is pleased to announce the availability of Apache Lucene 3.4.0.

Lucene can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/. Maven artifacts are available here.

If you are already using Apache Lucene 3.1, 3.2 or 3.3, we strongly recommend you upgrade to 3.4.0 because of the index corruption bug on OS or computer crash or power loss (LUCENE-3418), now fixed in 3.4.0.

See CHANGES and contrib CHANGES for details.

Highlights of the Lucene release include:

- Fixed a major bug (LUCENE-3418) whereby a Lucene index could easily become corrupted if the OS or computer crashed or lost power.
- Added a new faceting module (`contrib/facet`) for computing facet counts (both hierarchical and non-hierarchical) at search time (LUCENE-3079).
- Added a new join module (`contrib/join`), enabling indexing and searching of nested (parent/child) documents using `BlockJoinQuery`/`Collector` (LUCENE-3171).
- It is now possible to index documents with term frequencies included but without positions (LUCENE-2048); previously `omitTermFreqAndPositions` always omitted both.
- The modular `QueryParser` (`contrib/queryparser`) can now create `NumericRangeQuery`.
- Added SynonymFilter, in `contrib/analyzers`, to apply multi-word synonyms during indexing or querying, including parsers to read the wordnet and solr synonym formats (LUCENE-3233).
- You can now control how documents that don't have a value on the sort field should sort (LUCENE-3390), using `SortField.setMissingValue`.
- Fixed a case where term vectors could be silently deleted from the index after `addIndexes` (LUCENE-3402).

3.4 28 July 2011 - WARNING: Index corruption and crashes in Apache Lucene Core / Apache Solr with Java 7

Oracle released Java 7 today. Unfortunately it contains hotspot compiler optimizations, which miscompile some loops. This can affect code of several Apache projects. Sometimes JVMs only crash, but in several cases, results calculated can be incorrect, leading to bugs in applications (see Hotspot bugs 7070134, 7044738, 7068051).

Apache Lucene Core and **Apache Solr** are two Apache projects, which are affected by these bugs, namely all versions released until today. Solr users with the default configuration will have Java crashing with `SIGSEGV` as soon as they start to index documents, as one affected part is the well-known Porter stemmer (see LUCENE-3335). Other loops in Lucene may be miscompiled, too, leading to index corruption (especially on Lucene trunk with pulsing codec; other loops may be affected, too - LUCENE-3346).

These problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs, affecting also many more applications. In response to our questions, they proposed to include the fixes into service release u2 (eventually into service release u1, see this mail). **This means you cannot use Apache Lucene/Solr with Java 7 releases before Update 2!** If you do, please don't open bug reports, it is not the committers' fault! At

least disable loop optimizations using the `-XX:-UseLoopPredicate` JVM option to not risk index corruptions.

Please note: Also Java 6 users are affected, if they use one of those JVM options, which are **not** enabled by default: `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts`.

It is strongly recommended not to use any hotspot optimization switches in any Java version without extensive testing!

In case you upgrade to Java 7, remember that you may have to reindex, as the unicode version shipped with Java 7 changed and tokenization behaves differently (e.g. lowercasing). For more information, read `JRE_VERSION_MIGRATION.txt` in your distribution package!

3.5 1 July 2011 - Lucene Core 3.3

The Lucene PMC is pleased to announce the availability of Apache Lucene 3.3.

Lucene can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/

Highlights of the Lucene release include:

- The spellchecker module now includes suggest/auto-complete functionality, with three implementations: Jaspell, Ternary Trie, and Finite State.
- Support for merging results from multiple shards, for both "normal" search results (TopDocs.merge) as well as grouped results using the grouping module (SearchGroup.merge, TopGroups.merge).
- An optimized implementation of KStem, a less aggressive stemmer for English
- Single-pass grouping implementation based on block document indexing.
- Improvements to MMapDirectory (now also the default implementation returned by FSDirectory.open on 64-bit Linux).
- NRTManager simplifies handling near-real-time search with multiple search threads, allowing the application to control which indexing changes must be visible to which search requests.
- TwoPhaseCommitTool facilitates performing a multi-resource two-phased commit, including IndexWriter.
- The default merge policy, TieredMergePolicy, has a new method (set/getReclaimDeletesWeight) to control how aggressively it targets segments with deletions, and is now more aggressive than before by default.
- PKIndexSplitter tool splits an index by a mid-point term.

See CHANGES and contrib CHANGES for details. **Binary and source distributions are available here.** Maven artifacts are available here.

3.6 4 June 2011 - Lucene Core 3.2

The Lucene PMC is pleased to announce the availability of Apache Lucene 3.2.

Lucene can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/

Highlights of the Lucene release include:

- A new grouping module, under lucene/contrib/grouping, enables search results to be grouped by a single-valued indexed field
- A new IndexUpgrader tool fully converts an old index to the current format.
- A new Directory implementation, NRTCachingDirectory, caches small segments in RAM, to reduce the I/O load for applications with fast NRT reopen rates.
- A new Collector implementation, CachingCollector, is able to gather search hits (document IDs and optionally also scores) and then replay them. This is useful for Collectors that require two or more passes to produce results.
- Index a document block using IndexWriter's new addDocuments or updateDocuments methods. These experimental APIs ensure that the block of documents will forever remain contiguous in the index, enabling interesting future features like grouping and joins.
- A new default merge policy, TieredMergePolicy, which is more efficient due to being able to merge non-contiguous segments. See http://s.apache.org/merging for details.
- NumericField is now returned correctly when you load a stored document (previously you received a normal Field back, with the numeric value converted string).
- Deleted terms are now applied during flushing to the newly flushed segment, which is more efficient than having to later initialize a reader for that segment.

See CHANGES and contrib CHANGES for details. **Binary and source distributions are available here.** Maven artifacts are available here.

3.7 31 March 2011 - Lucene Core 3.1

The Lucene PMC is pleased to announce the availability of Apache Lucene 3.1.

Lucene can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/

Highlights of the Lucene release include:

- Numerous performance improvements: faster exact PhraseQuery;
  merging favors segments with deletions; primary key lookup is faster;
  IndexWriter.addIndexes(Directory[]) uses file copy instead of merging; various Directory performance improvements; compound file is dynamically turned off for large segments; fully deleted segments are dropped on commit; faster snowball analyzers (in contrib); ConcurrentMergeScheduler is more careful about setting priority of merge threads.
- ReusableAnalyzerBase makes it easier to reuse TokenStreams correctly.

- Improved Analysis capabilities: Improved Unicode support, including Unicode 4, more friendly term handling (CharTermAttribute), easier object reuse and better support for protected words in lossy token filters (e.g. stemmers).
- ConstantScoreQuery now allows directly wrapping a Query.
- IndexWriter is now configured with a new separate builder API, IndexWriterConfig. You can now control IndexWriter's previously fixed internal thread limit by calling setMaxThreadStates.
- IndexWriter.getReader is replaced by IndexReader.open(IndexWriter). In addition you can now specify whether deletes should be resolved when you open an NRT reader.
- MultiSearcher is deprecated; ParallelMultiSearcher has been absorbed directly into IndexSearcher.
- On 64bit Windows and Solaris JVMs, MMapDirectory is now the default implementation (returned by FSDirectory.open). MMapDirectory also enables unmapping if the JVM supports it.
- New TotalHitCountCollector just counts total number of hits.
- ReaderFinishedListener API enables external caches to evict entries once a segment is finished.

3.8 3 December 2010 - Lucene Java 3.0.3 and 2.9.4 available

Both releases fix bugs in the previous versions:

- 2.9.4 is a bugfix release for the Lucene Java 2.x series, based on Java 1.4.
- 3.0.3 has the same bug fix level but is for the Lucene Java 3.x series, based on Java 5.

New users of Lucene are advised to use version 3.0.3 for new developments, because it has a clean, type-safe API.

This release contains numerous bug fixes and improvements since 2.9.3 / 3.0.2, including:

- a memory leak in `IndexWriter` exacerbated by frequent commits
- a file handle leak in `IndexWriter` when near-real-time readers are opened with compound file format enabled
- a rare index corruption case on disk full
- `NumericRangeQuery` / `NumericRangeFilter` sometimes returning incorrect results with bounds near `Long.MIN_VALUE` and `Long.MAX_VALUE`
- various thread safety issues
- Lucene 2.9.4 can now also read indexes created by 3.0.x

Both releases are fully compatible with the corresponding previous versions. We strongly recommend upgrading to 2.9.4 if you are using 2.9.x; and to 3.0.3 if you are using 3.0.x.

See 3.0.3 CHANGES and 2.9.4 CHANGES for details. **Binary and source distributions are available here.** Maven artifacts are available here.

3.9 18 June 2010 - Lucene Java 3.0.2 and 2.9.3 available

Both releases fix bugs in the previous versions:

- 2.9.3 is a bugfix release for the Lucene Java 2.x series, based on Java 1.4.
- 3.0.2 has the same bug fix level but is for the Lucene Java 3.x series, based on Java 5.

New users of Lucene are advised to use version 3.0.2 for new developments, because it has a clean, type-safe API.

Important improvements in these releases include:

- Fixed memory leaks in `IndexWriter` when large documents are indexed. It also uses now shared memory pools for term vectors and stored fields. `IndexWriter` now releases `Fieldable`s and `Reader`s on close.
- `NativeFSLockFactory` fixes and improvements. Release write lock if exception occurs in `IndexWriter` ctors.
- Improve concurrency of `IndexReader`, especially in the context of near real-time readers.
- Near real-time readers, opened while `addIndexes*` is running, no longer miss some segments.
- Performance improvements in `ParallelMultiSearcher` (3.0.2 only).
- `IndexSearcher` no longer throws `NegativeArraySizeException` if you pass `Integer.MAX_VALUE` as `nDocs` to search methods.

Both releases are fully compatible with the corresponding previous versions. We strongly recommend upgrading to 2.9.3 if you are using 2.9.x; and to 3.0.2 if you are using 3.0.x.

See 3.0.2 CHANGES and 2.9.3 CHANGES for details. **Binary and source distributions are available here.** Maven artifacts are available here.