ReleaseNote40

```
MMM???? 2012, Apache Lucene, 4.0 available
The Lucene PMC is pleased to announce the release of Apache Lucene 4.0

Apache Lucene 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.

This release contains numerous bug fixes, optimizations, and
improvements, some of which are highlighted below.  The release
is available for immediate download at:
   http://lucene.apache.org/core/mirrors-core-latest-redir.html

See the CHANGES.txt file included with the release for a full list of
details.

Lucene 4.0 Release Highlights:

 * The index formats for terms, postings lists, stored fields, term vectors, etc
   are pluggable via the Codec api. You can select from the provided
   implementations or customize the index format with your own Codec to meet your needs.

 * Similarity has been decoupled from the vector space model (TF/IDF). Additional models
   such as BM25, Divergence from Randomness, Language Models, and Information-based models
   are provided (see http://www.lucidimagination.com/blog/2011/09/12/flexible-ranking-in-lucene-4).

 * The new doc values feature stores typed values per-document.  It
   can be used for custom scoring factors (accessible via
   Similarity), for pre-sorted Sort values, and more.

 * IndexWriter now flushes segments to disk concurrently, when the
   application uses multiple threads for indexing, resulting in substantial performance improvements
   (see http://blog.mikemccandless.com/2011/05/265-indexing-speedup-with-lucenes.html).

 * Per-document normalization factors ("norms") are no longer limited to a single byte.
   Similarity implementations can use any DocValues type to store norms.

 * New index statistics have been added, including the number of tokens for a term or field, number of postings
   for a field, and number of documents with a posting for a field.  These support additional
   scoring models (see
   http://blog.mikemccandless.com/2012/03/new-index-statistics-in-lucene-40.html).

 * A new default term dictionary/index (BlockTree) indexes shared prefixes
   instead of every n'th term. This is not only more time- and space- efficient, but can
   avoid going to disk at all for terms that do not exist in certain cases. Alternative term
   dictionary implementions are provided and pluggable via the Codec api.

 * Indexed terms are no longer limited to UTF-16 char sequences; they can now be any binary
   value encoded as byte arrays. By default, text terms are encoded as UTF-8
   bytes. Sort order of terms is defined by their binary value, which is identical
   to UTF-8 (Unicode code point) sort order.

 * Substantially faster performance when using a Filter during searching.

 * File-system based directories can rate-limit the IO (MB/sec) of merge
   threads, to reduce IO contention between merging and searching threads.

 * A number of alternative Codecs and components have been added: "Appending"
   works with append-only filesystems (such as Hadoop DFS), "Memory" writes the entire
   terms+postings as an FST read into RAM (see
   http://blog.mikemccandless.com/2011/06/primary-key-lookups-are-28x-faster-with.html),
   "Pulsing" inlines the postings for low-frequency terms into the term dictionary (see
   http://blog.mikemccandless.com/2010/06/lucenes-pulsingcodec-on-primary-key.html),
   "SimpleText" writes all files in plain-text for easy debugging/transparency (see
   http://blog.mikemccandless.com/2010/10/lucenes-simpletext-codec.html),
   "Bloom" uses a bloom filter to sometimes avoid disk seeks when looking up terms,
   "Direct" holds all postings as simple byte[] and int[] for very fast performance at the
   cost of very high RAM consumption, "Block" use a new index layout and compression scheme for
```

improved performance, among others.

* Term offsets can be optionally encoded into the postings lists and retrieved per-position.

* A new AutomatonQuery returns all documents containing any term matching a provided finite-state automaton (see http://www.slideshare.net/otisg/finite-state-queries-in-lucene).

* FuzzyQuery is 100-200 times faster than in past releases (see http://blog.mikemccandless.com/2011/03/lucenes-fuzzyquery-is-100-times-faster.html).

* A new spell checker, DirectSpellChecker, finds possible corrections directly against the main search index without requiring a separate index.

* Various in-memory data structures such as the term dictionary and FieldCache are represented more efficiently with less object overhead (see http://blog.mikemccandless.com/2010/07/lucenes-ram-usage-for-searching.html).

* All search logic is now required to work per segment, IndexReader was therefore refactored to differentiate between atomic and composite readers (see http://blog.thetaphi.de/2012/02/is-your-indexreader-atomic-major.html).

* Lucene 4.0 provides a modular API, consolidating components such as Analyzers and Queries that were previously scattered across Lucene core, contrib, and Solr. These modules also include additional functionality such as UIMA analyzer integration and a completely reworked spatial search implementation.

Noteworthy changes since 4.0-BETA:

* A new "Block" PostingsFormat offering improved search performance and index compression. This will likely become the default format in a future release. (see http://blog.mikemccandless.com/2012/08/lucenes-new-blockpostingsformat-thanks.html).

* All non-default codec implementations were moved to a separated codecs module. Just add lucene-codecs-4.0.0.jar to your classpath to test these out.

* Payloads can be optionally stored on the term vectors.

* Many bugfixes and optimizations.

Please read CHANGES.txt and MIGRATE.txt for a full list of new features and notes on upgrading. Particularly, the new apis are not compatible with previous versions of Lucene, however, file format backwards compatibility is provided for indexes from the 3.0 series and the 4.0-alpha and -beta releases.

Please report any feedback to the mailing lists (http://lucene.apache.org/core/discussion.html)

Note: The Apache Software Foundation uses an extensive mirroring network for distributing releases. It is possible that the mirror you are using may not have replicated the release yet. If that is the case, please try another mirror. This also goes for Maven access.