NearRealtimeSearch

Near Realtime Search

Near realtime search in Lucene refers to features added to IndexWriter in Lucene version 2.9 that enable updates to be efficiently searched hopefully within milliseconds after an update is completed.

- Minimize IO overhead
- Transparent to the user
- Efficient RAM management
- Load field caches at the segment level

One goal of the near realtime search design is to make NRT as transparent as possible to the user. Another is minimize the latency after an update is made to perform a search that includes the update. Now Lucene offers a unified API where one calls getReader and any updates are immediately searchable.

At this point NRT is a workaround for the latency of fsync on most operating systems. Meaning for updates, instead of syncing what's in RAM to disk, we perform merges and keep deletes in RAM until a maximum bytes size is reached or commit is called by the user. In the future NRT may involve searching directly on the ram buffer without first encoding a full Lucene segment. Prior to 2.9 loading field caches was an IO latency issue on the search side.

Index Writer manages the subreaders internally so there is no need to call reopen, instead getReader may be used. A benefit of this design is the efficiency of deletes where they are not written to disk until commit is called. Deletes are carried over in the RAM segment reader held by index writer. Here we're leveraging the index reader clone method which when used, keeps references to deletes and norms via a copy by reference mechanism. Otherwise we could be calling fsync just to update one document.

NRT adds an internal ram directory (LUCENE-1313) to index writer where documents are flushed to before being merged to disk. This technique decreases the turnaround time required for updating the index when calling getReader.

Sample code:

```
IndexWriter writer; // create an IndexWriter here
Document doc = null; // create a document here
writer.addDocument(doc); // update a document
IndexReader reader = writer.getReader(); // get a reader with the new doc
Document addedDoc = reader.document(0);
```

Internals

- Index Writer pools Segment Readers
- Field caches are searched at the segment level (LUCENE-1483). They only need to be loaded per segment rather than for all segments (which was the functionality pre-2.9)
- Index Writer.getReader (LUCENE-1516) flushes updates without calling commit or flushing deletes to disk (i.e. doesn't call fsync)
- Speedup in indexing because instead of waiting for the RAM buffer to be written to disk, the RAM buffer is more quickly written to the Index Writer internal RAM Directory
- File Switch Directory (LUCENE-1618) is used by NRT to write potentially large docstores and term vectors to disk rather than to the RAM Directory. This makes more RAM available for NRT.
- Index Reader.clone (LUCENE-1314) is used in Index Writer to carry deletes over within segment readers. It is also used to freeze a version so that a merge may complete and deletes may be safely applied and searched on concurrently.
- Cloning bitvectors could rapidly consume heap space if updates are frequent, so LUCENE-1526 divides the bitvector into chunks.

IO Cache

Large merges potentially bump existing segments out of the IO cache. A query that was fast may suddenly be slow due to the latency of accessing the hard drive. One way to address this is to implement a JNI based Directory that implements fadvise or madvise. The advise calls would allow segment merger to tell the OS not to load the segments being merged into the IO cache.