Building the News Search Engine

Ramkumar Aiyengar

Team Leader, R&D News Search, Bloomberg L.P.

andyetitmoves@apache.org

- A technology company
- Our strength and focus is data
- “The Terminal”, vertical portals
- Customers: Primarily finance
- Also government, lawyers etc.

Bloomberg

- Started 9 years ago
- Now team lead for R&D News Search
- Search, alerting, ingest infrastructure
- Started with Solr/Lucene 3 years ago
- Now a committer with the project

The News Search Ecosystem

- Suggest queries as the user is typing
- Understand a query to figure out what’s being requested
    - NLP, entity recognition/disambiguation, spellcheck
- Search for keywords and metadata of documents
- Sort the documents as the usage demands
    - If sorting by relevance, what’s actually relevant for the user?
    - Should some results be promoted ahead of others?
- Alert users when new stories match the active search
- Expose facets for refining and discovery
- Recommend searches and search results

327K+ Subscribers

10 Million Searches PER DAY

1 Million Stories

PUBLISHED EACH DAY

INDEX OF 500 MILLION STORIES

500 Stories
PER SECOND

Available for Search
in ~100ms

180ms
RESPONSE TIME

More. Better. Faster.

Alerts in 100ms

1.5 MILLION
SAVED SEARCHES

What we did in the last few years...

- Existing system based on a proprietary system
  - Proprietary product, past its end of life
  - Inflexible, no scalable relevance sorting
- Enter Solr/Lucene!
  - Rich in features, extensible and actively maintained
  - Free software, we are involved and contribute back!
  - From-scratch alerting backend based on Lucene and Luwak
- Architectural revamp of the News Search backend
  - Scalable with load and data: Just add machines!
  - Maintainable: Easy to add metadata, re-index all of the data

How we did it in four easy steps…

- Make it work
- Make it fast
- Make it stable
- Make it better

What goes in…

- Document
    - News stories, research documents, tweets…
    - Story body, headline, time of arrival, source…
    - Tags (companies, topics, people etc.) associated with the story
- Query

    KEYWORDS:("Donald Trump" N/5 "great" IN STORYTOP/75) AND (REGION:MEX OR REGION:NKOREA) AND NOT TOPIC:ODD AND (WIRE:BLOOMBERG OR WIRE:TWT)*

    - Multiple fields (keywords, topics, regions, sources)
    - Boolean (and/or/not), proximity (n/5), zoning (storytop/75) operators
    - Phrase search ("Donald Trump"), wildcard searches ("great*")
    - Range queries (time of story), search filters (relevance, language)

The madness we deal with…

- Arbitrarily complex Boolean queries
    - for both search and alerting
    - users create queries as large as 20K characters (or more!)
- Lists of metadata can be specified in short hand
    - A “ticker list” could have 1000s of companies interesting to the user
- Stories from 125K+ sources
    - users privileged for a subset of these sources
    - can be turned on/off per user
    - ACLs can have a few, many or all of the users
- Searches and stories in 40 languages
    - any user can have a subset of these selected

The News Search cloud

- Lots of Linux machines on Solr clouds housing:
    - Hundreds of shards, and thousands of Solr cores
    - Multiple tiers: ‘recent’ collection to optimize chronological results
    - Cross data-centre redundancy
- Stories available for search in 125ms, minimal caching
- Custom components for:
    - Parsing: XML query parser, with additions
    - Indexing: For handling tags, more on that later…
    - Searching: Custom Lucene queries for some cases
    - Post Filtering: For privileging of news stories

Parsing search queries

- We need to…
  - Understand In-house search syntax
  - Validate/Privilege tags based on DB
  - Present part of search query in UI
  - Understand query to suggest sources
- Parsed outside Solr to XML queries (SOLR-839)
  - Future: Compact transfer: JSON, Binary? (SOLR-4351)
- Will *state* NP/10 ((“tax*” N/5 “incentive”) OR “sales and use”)* work?
  - Making spans interoperate with any query
  - Originally used flaxsearch/lucene-solr-intervals, now upstream

Searching for news tags

- Each story has multiple tags associated
  - Topics, companies, regions, people…
  - Each tag has a ‘relevance’ provided by a classifier
  - Up to a few hundred tags per story, millions overall
- Tag relevance to be considered for scoring and filtering
  - How do you normalize relevance with keywords?
- One solution: repurpose keyword ranking for tags
  - Use TF/IDF for tags like with keywords
  - Modify searches to be filtered by ranges of TF values

Optimizing searches

- Running “ticker list” searches fast is hard
  - Boolean OR of thousands of terms with “relevance” filters
  - Naïve: BooleanQuery(Filter(TermQuery, FRange)…)
  - Better: BooleanQuery(TermFreqQuery…)
  - Even more: TermsFreqQuery
- Optimizing searches for sorting by time (SOLR-5730)
  - Pluggable merge policy factory in Solr (SOLR-8621)
  - Solr support (use the schema) for EarlyTerminatingSortingCollector
- How aggressive is your merge policy?
  - aka how much can you squeeze out of your SSDs?

Optimizing searches

- You really need that ShardHandlerFactory? (and other tales of GC)
  - Even small inefficiencies multiply at scale (e.g. SOLR-6603)
  - Routing smartly to reduce the probability of GC (SOLR-6730)
- Looking out for what the kernel is doing
  - “swappiness”, I/O scheduler fit for SSDs, huge pages
- Watch where the time’s spent (may not be where you expect…)
  - No point with fast searches if max connections is too low
  - There may be that odd hardcoded number (e.g. SOLR-6605)
  - Even Jetty could have bugs which cause requests to stall and timeout

Scaling Solr Cloud

- Distributed coordination (good ol’ Overseer!)
    - Hundreds of cores restarted at a time during weekends
    - Scaling cluster state (SOLR-5381, SOLR-5872)
- Leadership mechanisms have to scale
    - Transitions have to happen quickly (SOLR-6261)
    - Leaders shouldn’t gang up on some machines (SOLR-6491)
- Replica recovery should not affect live traffic
    - Worse, shouldn’t affect cloud stability with network saturation!
    - Throttling (SOLR-6485), using a different network (SOLR-9044)
    - Use transaction log recovery where possible (SOLR-6359)

There will be storms…

- Thousands of cores in a cloud is a lot of fun □
  - Started with 4.3.1 with many stability concerns, a lot better now
- If there’s a race condition, we will hit it!
  - Is it safe to stop multiple replicas of a shard simultaneously?
  - What happens when you shutdown in the middle of a merge?
  - Can a delete-by-query around a leader switch stall it? (SOLR-8760)
- If you have to screw up, be controlled about it!
  - Zombie checks should be light (SOLR-5718)
  - Will the cloud always heal after a network partition?

Storms in teacups can blow over…

- With infinite query flexibility come poisonous queries
  - No good can come out of phrases, wildcards and spans in excess
  - “Why don’t I copy paste the entire text to find the article?”
  - “My keyboard has a key stuck, time for lunch!”
- Solr now has better circuit breakers for queries (SOLR-5986)
  - Long queries can take down replicas with GC pressure!
  - We can do better, statistical “query plans” anyone?
- User replica affinity (SOLR-6730)
  - People can be persistent with their failing queries!
  - Protects the system against one user taking down the cloud

Containing systemic failure

- Protecting one part from the system from the other
    - Isolating thread-pools for searching and indexing (SOLR-7344)
    - Isolating query federation from query execution
    - Isolating critical roles like the Overseer (SOLR-5476)
    - Future: Isolating costly queries from cheap ones (what’s costly?)
- It’s all one happy cloud, until garbage gets into the input…
    - Loosely coupled replicas to mitigate issues with input pipeline
    - CDCR to soon help synchronisation! (SOLR-6465, SOLR-6466)

Improving the search experience

- Grouping is great, as long as it's pragmatic
  - People, not bots, sometimes get to hundreds of pages down!
  - Considering a window of N results for grouping and deep paging
- Implementing a Learning-to-Rank framework in Solr (SOLR-8542)
  - Define features, models to rank results
  - Get back feature values with responses to train models offline
  - Talk at Lucene Revolution: Learning to Rank in Solr on YouTube
- Showing what's trending in news, and intelligent faceting

The road ahead…

- Never-ending quest of relevance: better user models, connecting data
- Searching across languages with language detection and translation
- Better searching across news and social media
- Searching and scoring effectively in bulk
    - “Get me the most important story for each company in my portfolio”
- Readership sorted views for any search
    - Tens of millions of story hits per day, how best to index? (SOLR-5944?)
- Blending chronological and relevance ranked searching

Committers: 3, Patches: 100s, Challenges: Countless!

Q&

Alerting: Prospective search

- Searching turned upside down
    - Find which of millions of searches match one document
    - Alert users who are interested in these searches
    - Use tailored searches to tag documents with topics
    - No out of the box support for Solr
- Initial two-week prototype
    - MemoryIndex, loop over all searches registered
    - Works, but too slow for any production use
    - In theory, you can “throw more hardware”, but we can do better…

Alerting: Prospective search

- Baleene: A standalone application for prospective search
    - Based on and improving Luwak, in turn based on Lucene
    - Understands document schema like Solr does
    - Initially a Lucene fork needed, then merged with 5.3
    - Indexes queries, and “pre-searches” queries of documents
    - “Turning search upside down” – Alan Woodward at Buzz 2014
    - Planning to release application as open source
- Future: Alerting based on relevance
    - Feed document frequencies from Solr to Baleene for scoring
    - Top ranked result screens updating in real time