Consulting for HOTHOUSE / SCOPUS POC

Trip Report – March 17, 2014 – Paul Nelson

This document is a record of the conversations, discussions, decisions, and whiteboard drawings generated during a consulting trip by Search Technologies to the Elsevier Dayton office on 3/17/2014.

Participants in the meetings included: Darin McBeath (Elsevier), Curt Kohler (Elsevier), Jim Slaton (Elsevier), and Paul Nelson (Search Technologies).

The discussions covered much of SCOPUS features, index structure, different cores, query ratios, etc.

The following sections cover individual topics discussed and action items / recommendations.

# Facets

Current tests on Core-Abstracts are showing very slow results (around 2+ seconds per query) with Solr Facets turned on.

Elsevier mentioned a general question on “how do we handle facets with very large cardinalities”.

Discussed the use of “docValues” for improved faceting with Jose Aguilar:

* Much more efficient storage of facet values 🡪 Less RAM 🡪 Better performance
* Able to handle facets which don’t fit into RAM
  + Uses Solr index structure to store and access facet values
* Requires schema.xml changes:
  + “Codec” to be installed for the field type in
  + docValues=true for the field
* Discussed “Disk” versus “Lucene42” variations

Jose discussed improvements to Solr facet values which were implemented for GPO (previously GPO had “out of memory” errors, but DocValue facets solved the problem. The cardinality of the GPO facets are much smaller than Elsevier facets, however.

Jose did caution that “turning on docValues for too many facets slows indexing time, by as much as 3x”.

RESULT: Elsevier will try turning on docValues to evaluate if facet performance is better.

NOTE: Solr 4.7 has additional performance improvements for DocValues.

# Testing

There was a general discussion about testing methods and testing results.

* For Core-Abstracts (55m records, 600gb index, 1 machine, 60gb RAM, 16 cores):
  + Queries with facets: 2+ second response time
  + Queries without facets: 500-800ms response time
* ~550ms for Solr search time / 750ms including network transfer time for results
* Slow queries appear to be those which return 2000 results
  + Some of these are oddly slow (i.e. 30seconds)

There was a general concern and question about “why can we only get 2-3 QPS?”

Possible causes:

* The test is not testing QPS
  + The tester sends the query, waits for a response before sending the next query.
  + An asynchronous testing environment should be implemented, see below.
* Each query is read from DynamoDB
  + Perhaps this is a bottleneck
  + Recommend creating the entire test and downloading the queries ahead of time, rather than doing this query-by-query during the test
* Network transfer appears to be a bottleneck
  + Modify the test to use SolrJ instead of JSON for query submission and results
  + Look for / test network issues between the test machine and the search server
* OR, the system could just be slow
  + Recommend using 5 partitions (11m abstracts/partition) instead of 1
* Perhaps the system is garbage collecting a lot
  + Does not appear to be running the GC very often (according to Sematext monitoring)
  + Need to verify this with Visual VM
  + Recommend tuning the JVM garbage collector
* Perhaps other Linux parameters are not being set properly
  + Look at how much swapping there is in the Operating System
  + Tune the Linux Virtual Memory values (swappiness, etc.)

Other oddities:

* JVM appears to be configured for 25gb of RAM but is only using up 12.5
  + Does it only need 12.5?
  + Or is Sematex not showing the right number?
  + Or is the configuration wrong?
  + Recommend looking at memory/garbage collection usage with Visual VM instead.

## Asynchronous Testing Environment

Recommend creating “QPS” ramp tests:

* Submit queries at a known, “target” QPS.
  + This is done on a separate “controller” thread.
  + Queries are submitted to a threaded execution queue.
  + Dozens of threads are used to submit each query.
* Ramp up the “target QPS”
* Measure the QPS of the results coming back
* Plot the target QPS against the return results QPS:



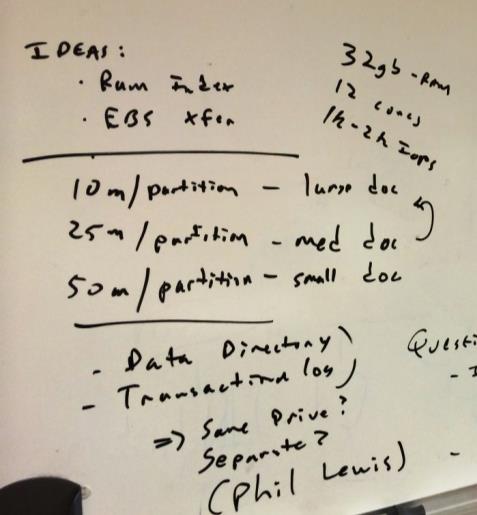
Graphs such as the above clearly identify when the system goes into saturation. Measuring average QPS in saturation identifies the maximum QPS that the system can support.

The above was done with the FAST “SBench” tool.

ACTION ITEM (Paul): Provide “SBench” to Elsevier.

# Number of Documents per Partition

The general structure of the index was discussed. The number of documents per Solr partition:

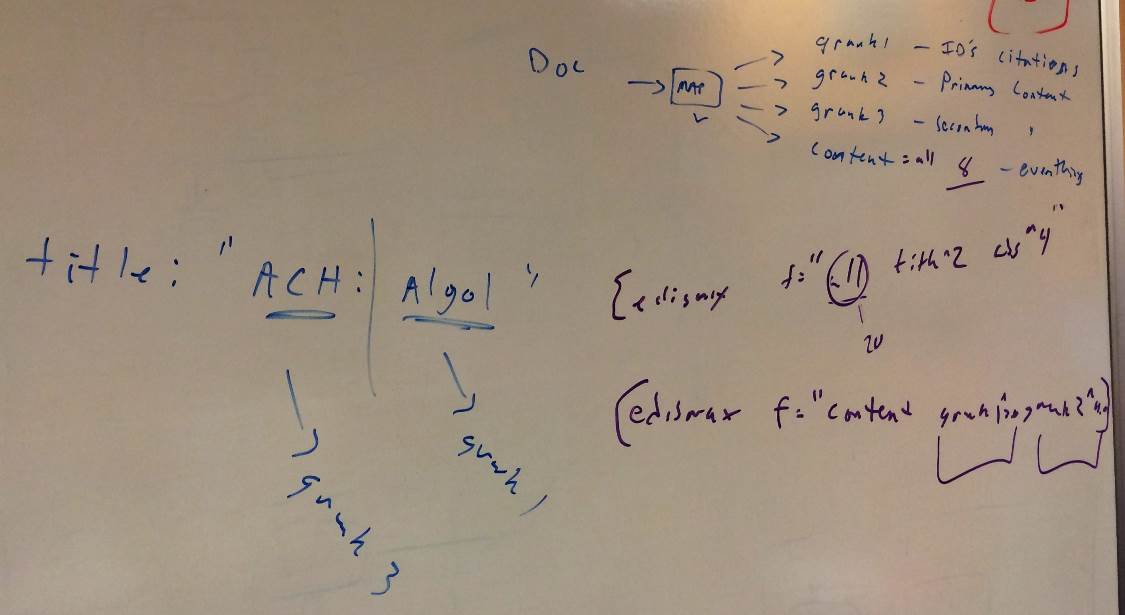


Discussed the general size of a Core-Abstract and decided they were somewhere in the “Medium-Large” vicinity.

Recommended dividing up the 55m abstracts into 5 shards for testing.

# Composite Fields

Discussed how fields should be indexed for relevancy ranked keyword search:



Recommended separating the fields for the relevancy model away from other fields.

So, do \*not\* use “title”, “author”, etc for relevancy ranked keyword search. Instead use “grank1”, “grank2”, “grank3” & “content” (see picture). All content in any field should be grouped and weighted the same.

This allows for more complex business rules to determine the priority of content search, and will simplify and regularize the search expressions.

Recommend having no more than 4 fields in the query (the “edismax” ‘f’ parameter). Content should be mapped to one of the four fields rather than adding a large number of fields to the query (which will make queries slow).

# Indexing Architecture / Flipping / Publication Counts / Reverse Links

The goal for Elsevier is to have a completely consistent view of the index at any point in time. This includes:

* Accurate “PC” (publication count) numbers
  + The count of items which reference the abstract in question
* Accurate “inbound links”
  + The list of other abstracts which reference the abstract in question

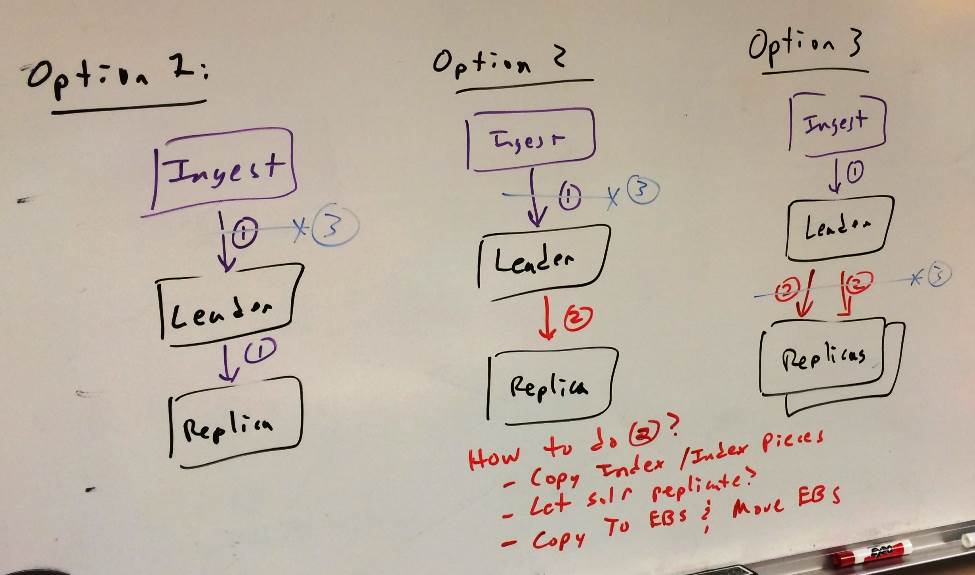
These items are complex because they are generated data. Current Elsevier uses Amazon Redshift to compute the reverse links and the PC counts and then update (e.g. re-index) all of the records which change.

## ARCHITECTURE-A: Creating an Index with Updates and then “flipping it” on-line

This is the more “off-the-shelf” architecture.

### Indexing

Three alternatives for indexing with Architecture A were discussed:



Option 1:

1. Ingest to a leader which is replicated “as you go along”.
2. When done, sever the link from the ingestion server. Move both the leader and replica on-line as the new production system

Option 2:

1. Ingest to a leader.
2. When done, add the replica to the leader. Solr copies the index to the replica.
3. When done, sever the link from the ingestion server. Move both the leader and replica on-line as the new production system

Option 3:

1. Ingest to a leader.
2. When done, add two replicas. Allow Solr to replicate from the leader to both replicas.
3. When done, sever the link to the replicas. Move them both on-line as the new production system.

General agreement that option 3 above is best / most stable. It does not require updating an old-version of the index and appears to be a simpler, more stable architecture.

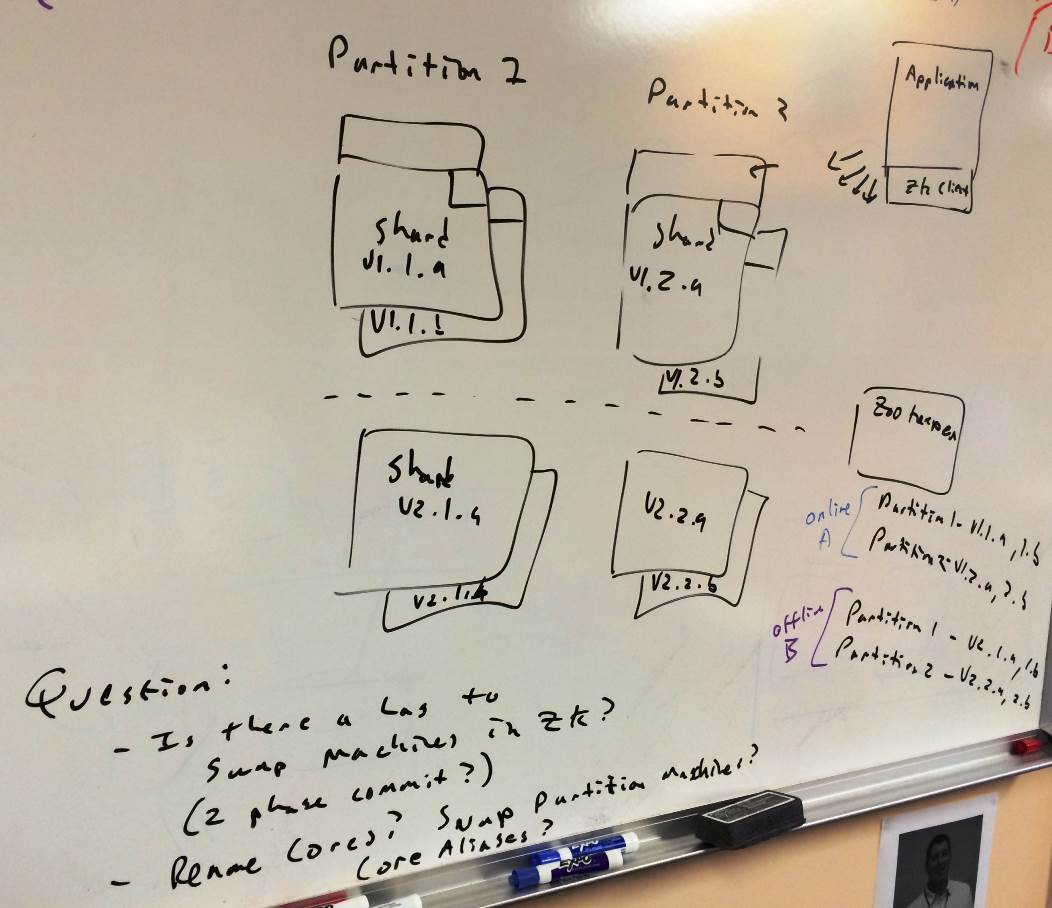
Discussed methods for replicating indexes:

* Using Solr
* Manually copy / Rsync the modified Solr index files
* Copy entire index to an ESB store, then dismount and re-mount the EBS onto another machine

### Query

On the query side, there are two alternatives:

1. Use a hardware load balancer to move queries from one production system to an entirely different production system.
   1. E.g. An A/B ping-pong approach
   2. Use DNS or hardware router to re-route requests when the swap occurs
2. Alternative 2 is to make changes in Zookeeper to change what is the off-line servers vs the on-line servers for a set of cores:



### Action Items for Architecture-A

ACTION ITEM: (Phil Lewis):

* How much lag is there when swapping machines in ZooKeeper?
* Can you rename cores in Zookeeper? Swap partitions? Use Core Aliases to swap an “off-line” index into an “on-line” index?
* Discuss index and replica structure / methods.
* Should SolrCloud transaction log be on the same disk as the index?
* What is a Solr Snapshot?
  + Is it transferable / portable to other machines?
* Any information on number of indexing threads? Other Solr configuration information?
* Can a cluster be brought up “read only” ?

## ARCHITECTURE-B: In-Place Updates

Alternatives were discussed about how to perform “in-place” updates without having to flip between two different systems.

### Hard Commits / Soft Commits

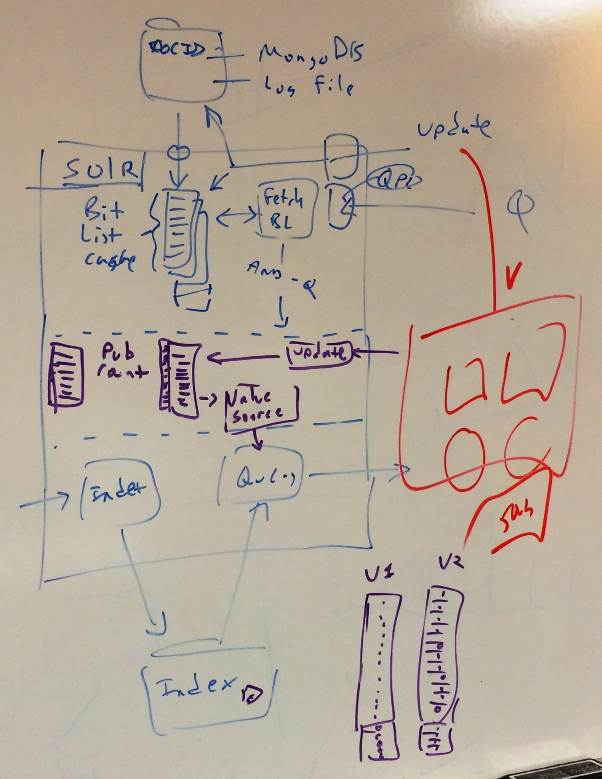
The difference between hard commits and soft commits were discussed.

Document updates and new documents *could* be written to all Solr machines “in-place” and committed – *as long as the search engines continue to search over the old index version*. The idea is that when the “flip” occurs, a message is sent to all search engines to “soft commit” – which causes them to update their view of the index to the latest version.

### In-Memory Updates

The second architecture discussed was maintaining in-memory structures for publication count and reverse links, and updating these “in place”.

Elsevier requested a quote for accomplishing this. This will be provided in a separate document.



# Query Processing Language (QPL) / XML / Query Manipulation

Several aspects of QPL were discussed.

## Full XML Scope Searching

Jose Aguilar demonstrated how QPL could be used for full XML searching, and all of the features which it supports (nested scopes, embedded phrases, etc.).

There was general agreement that the GPO XML solution should be refactored to make it simpler for Elsevier to use.

Elsevier covered their use of simple Spanning queries (SpanNear [ordered] + SpanNot) to implement simple multi-valued record searching. While this appears like it could work for simple cases, Search Technologies expects (and Elsevier agrees) that it is not a general-purpose solution.

ACTION ITEM: Mary Jo Houghton – Provide a price for Elsevier for QPL with XML.

## Other features of QPL

* Wildcards in phrases
  + Not discussed in great detail, but this is a feature of QPL
* Search-server side business rules
  + Essentially “stored procedures” which allows moving business rules from the application to the search engine.
  + Allows modifications for updated relevancy scoring or new search features to be made without having to change how queries are constructed.
* Compatible with Apache Zookeeper
  + Updates to QPL scripts are automatically propagated to all nodes.
* Dictionary-based Lemmatizer
  + Available with QPL
* Wildcard expansion cutoff
  + QPL has special coding for generating wildcard expansions.
* Full “standard – end-user friendly” query parser with NEAR/#, BEFORE/#, ADJ, and “field:” operators.

# Miscellanous

## SolrJ

Discussed that Elsevier could see a 25% improvement in network transfer performance using SolrJ instead of JSON for transferring search results (on the search side).

SolrJ uses Java Binary serialization / deserialization to transfer content over the wire. This is more efficient than JSON or XML methods.

## Use of Hadoop

Generally, Search Technologies believes that Hadoop is a potentially transformational search technology – but only if embraced as more of an end-to-end platform.

Using it simply to create indexes for SCOPUS will likely not find much advantage to doing this in Hadoop.

## Recommended Solr Configurations

Search Technologies does not have recommended Solr Schema or Solr configurations that are appropriate to the SCOPUS use case.

## Tomcat vs Jetty

Generally prefer Tomcat, but the different (to the best of our understanding) is small.

## Apache Tika

Not recommended for text extraction from XML. Only for extraction from PDF, MS-Word, etc.

## Monitoring Tools

Sematext is the preferred tool.