Search Engine Rodeo

Anthony Arnone and Neal Richter – Summer 2007

Our goal was a quick-and-dirty comparison of the relative performance of three open source search engines. The candidate engines are MySQL Fulltext search, Sphinx, and Solr. This was part of an ongoing effort to evaluate the “enterprise scalability”, namely speed and stability, of these search engines.

The Experiment

Test Computer

The machine used for testing had the following properties:

2 dual-core Opteron 2.2GHz processors

4GB DDR 333 MHz RAM

7200 SATA2 Raid (running at SATA1 speed)

Document Types

There are multiple document types, with each type residing in its own search index. The purpose of these document types was to test the multi-field search capabilities of the candidates.

Type	Fields
Plain	one text field
Simple	Title, URL, keywords, body
Structured	Title, URL, keywords, plus multiple body text fields

Documents

For each document type {plain, simple, structured} and each search engine {MySQL Fulltext, Sphinx, Solr}, there were 100,000 documents, for a total of 900,000 documents. Each document is randomly generated from a large (23 megabytes) corpus of text. The documents were created to be the same size, regardless of their type. The average size of the total 100,000 documents on disk was 290 megabytes, making for an average of 2.9 kilobytes per document.

Disk Usage

Disk usage by the search engines is of concern, as the scalability can become hampered not just by the physical disk that is used for searching, but for how much data must be loaded from disk to do searches. Each candidate has its own method of storing the searchable index, and while each search engine requires space for the search index itself, it will not necessarily store the original text. MySQL Fulltext *has* to store the original text, while Solr can optionally store the data. Sphinx has limited text storage capabilities, but can connect to its MySQL database source for information about the documents if required. For the test results, only the disk usage of the searchable index is considered.

Query Testing

The utility used to issue queries and collect statistics was Jmeter, an Apache project written in Java. The queries were issued from a secondary machine (100 Mbit Ethernet connection). Queries are randomly selected from a corpus of approximately 11,000 words that are known to exist in the documents. Search words are at least four characters long, and consist of only alphanumeric (A-Z a-z 0-9) characters. Each index was queried a total of 5000 times for each document type, with Jmeter running 100 threads with 50 requests each. The different document/index query sets were run independently (nothing else was being done at the time).

For MySQL Fulltext and Sphinx we wrote a PHP script to use ODBC or SphinxAPI to contact the engines, issue queries and format the results into HTML roughly equivalent to SOLR's XML output.

Versions

- MySQL Fulltext via MySQL 5.0
- Sphinx 0.9.7
- Solr 1.2

  Default configurations of all engines were used.

Criteria

Indexing 100K documents must be faster than 10 minutes to bother with query per second testing.

Results

The results are organized by document type. Note that not all tests were performed.

Plain	Insert/Index Time	HDD Size	Queries
MySQL Fulltext	16 min 10 sec	260 MB (.MYI file)	skipped
Sphinx	30 sec	137 MB (.spd file)	138 / sec*
Solr	5 min 23 sec	106 MB	465 / sec

Simple	Insert/Index Time	HDD Size	Queries
MySQL Fulltext	24 min 2 sec	344 MB (.MYI file)	skipped
Sphinx	42 sec	235 MB (.spd file)	90 / sec*
Solr	5 min 47 sec	106 MB	478 / sec

Structured	Insert/Index Time	HDD Size	Queries
MySQL Fulltext	26 min 18 sec	389 MB (.MYI file)	skipped
Sphinx	49 sec	272 MB (.spd file)	84 / sec*
Solr	6 min 3 sec	103 MB	466 / sec

* PHP script used to issue each query and format to HTML

Conclusions

MYSQL Fulltext

MYSQL Fulltext search was the worst performing engine. Index time was quite bad and size on disk of the index structures was also the worst of the lot. Query per second tests were skipped as the slow indexing speed eliminated it from further consideration. Anecdotally, the search query latency was relatively poor, implying a poor searches per second metric.

MYSQL Fulltext is not recommended for anything where performance is desired.

Sphinx

Sphinx search was lightning fast to index documents. It's size on disk was reasonable, similar to the size of the raw documents repository. Search speed as compared to Solr was quite poor in our tests.

It is nearly certain that a Java Servlet communicating with Sphinx's searchd would double the searches per second as starting the php executable upon every hit is expensive. At a minimum Apache's mod_php would speed up searches per second over our tests.

~~Sphinx is recommended for any situation where a closely integrated engine with MySQL is desired and especially for high write-load situations where indexing speed is more important than queries per second performance.~~

However it should be noted that it's ability to index homogeneous document types is not great. Also, at first glance it's capabilities to do sophisticated query types, process chaining and result ranking algorithms was not as good as Solr.

Solr

Solr has the best index on disk size by a wide margin and was the clear winner in searches per second in our limited test. It did suffer a distant second place versus Sphinx for indexing speed.

~~Solr is recommended for high query-load situations.~~ Solr also is the clear winner in the raw capabilities of homogeneous document structure indexing and the wide variety of powerful query types & filter chains. The ability to manipulate result ranking algorithms via either configurations or custom Java plug-ins is hard to replicate in any other engine not based upon Lucene.

These conclusions are retracted in light of the Sphinx team's initial feedback on the test. More information TBA. We may do a joint test with them.