High Performance Full Text Search for Database Content

By Peter Zaitsev, Vadim Tkachenko
Percona Ltd

Presented 19 September 2006 at EuroOSCON 2006

About Peter Zaitsev

- Co-Founder Percona ltd
  - Service company specializing in MySQL and LAMP Performance
- 2002-2006 MySQL Inc
  - Head of High Performance Group
  - MySQL Support, Consulting, Benchmarks, Performance analyses
- 1999-2002
  - Co-Founder, CTO of SpyLOG.RU
    - Leading web statistics service in Russia

How do you know about FullText ?

- Personal Interest for years
- Consulting for Top Web Sites, while working in MySQL
- LjSEEK.COM – Hobby project
  - Indexing all public LiveJournal.COM posts for last years
  - Over 100.000.000 posts made available for search
- NNSEEK.COM – NewsGroup Search Engine
- Building system which indexes 10 times more information

Why FullText Search is Important?

- Is not it silly question to ask ?
- Natural way for humans to search for information
  - Search engines took over catalogs as traffic generators in a web long ago
- Quickly growing amount of information stored
  - Personal information, Business information, community information etc.

Why Full Text for DataBase ?

- Database – typical storage for Web Site data
- Contains original data
  - No design elements
  - Typically no data duplication
  - Best data granularity
  - Easy to cover all data
- Performance efficient
  - Web Server, Application Server are not involved.

Types of FullText Search Solutions

- Special Database Features
  - MySQL Full Text Search, Sienna
  - Solutions exists for PostgreSQL, Oracle and many others
- Home baked database based solutions
  - Using stored procedures or set of queries and keyword tables for search
- External Full Text Search Solutions
  - Lucene, Sphinx, Mnogosearch etc.

Full Text Search Features

- Not all FullText search solutions are equal.
- Search Query features
  - Boolean operators, field search, word parts search, phrase search etc
- Search Indexing control
  - Stemming, Stop words, Synonyms, Word separation control
- Support for different content types
  - Text, HTML, DOC, PDF, XML etc
- Index update abilities and speed

More Full Text Search Features

- Document Structure support
  - Indexed and non indexed attributes
- Character set and language support
  - Especially Eastern languages need extra care
- Relevance Quality and control
- Sort modes
- Special Queries
  - Top matching groups, trends
- Clustering

Database used for Demonstration

- Real data set of moderate size
  - Dump of English Wikipedia database
  - 3400617 articles
  - About 5GB in size
- Hardware
  - AMD Sempron with 1GB RAM
  - SATA RAID

Table Structure

```
CREATE TABLE `cont` (
  `old_id` int(8) unsigned NOT NULL default '0',
  `date_added` int(10) NOT NULL default '0',
  `title` varchar(255) character set utf8 NOT NULL
default '',
  `content` mediumtext character set utf8 NOT NULL,
  KEY `wi1` (`old_id`),
  FULLTEXT KEY `search` (`title`,`content`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

MySQL Full Text Search

- Available only for MyISAM tables
- Natural Language Search and boolean search
- Query Expansion support
- ft_min_word_len – minimum 3 char per word by default
- Extensive stop word list by default
  - Containing «zero», «talk» etc
- Frequency based ranking
  - Distance between words is not counted

MySQL Full Text Search Example

```
mysql> CREATE TABLE articles (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   title VARCHAR(200),
    ->   body TEXT,
    ->   FULLTEXT (title,body)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO articles (title,body) VALUES
    -> ('MySQL Tutorial','DBMS stands for DataBase ...'),
    -> ('How To Use MySQL Well','After you went through a ...'),
    -> ('Optimizing MySQL','In this tutorial we will show ...'),
    -> ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> ('MySQL vs. YourSQL','In the following database comparison ...'),
    -> ('MySQL Security','When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----+-----------------+---------------------------------------+
| id | title           | body                                  |
+----+-----------------+---------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...          |
+----+-----------------+---------------------------------------+
2 rows in set (0.00 sec)
```

MySQL Full Text Search Performance

- Index (and better data) should fit in memory
  - And key_buffer_size sized appropriately
- Stop words list should contain most frequent keywords
  - But make sure you do not need to search by them
- Lowering ft_min_word_len decreases performance
  - Low length keywords are often frequent
- OPTIMIZE TABLE
  - updates/deletes fragment index

Avoid Count and Limit

- Avoid counting number of matches
  - select title from cont where match (title,content) against ("democratic president candidate") limit 10;
    - Takes 0.25 sec
  - select count(*) from cont where match (title,content) against ("democratic president candidate")
    - Takes 16 min 13.41 sec
- Avoid Sorting
  - By separate field as «date» or by self computed relevance in boolean search
  - select title,match(title) against ("post office" in boolean mode)*10 + match(content) against("post office" in boolean mode) relevance from cont where match (title,content) against ("post office" in boolean mode) order by relevance desc limit 10;
    - Takes 3 min 38.35 sec

Be careful with WHERE and LIMIT

- Be care careful with extra where clause if it filters out a lot of matches.
  - select title from cont where match (title,content) against ("global service company") and date_added>unix_timestamp("2006-07-18 18:00:00") limit 10;
    - Takes 26 min 43.59 sec
- Large LIMIT offset slow down dramatically
  - select title from cont where match (title,content) against ("computer game industry") limit 1000,10;
    - Takes 9.99 sec

GROUP BY and Phrase Search

- GROUP BY requires all results so it is slow as well
  - select count(*) cnt,date(from_unixtime(date_added)) from cont where match (title,content) against ("computer game industry") group by date(from_unixtime(date_added));
    - Takes 26 min 38.76 sec
  - For example if we want to graph number of matches over time.
- Phrase Search makes Boolean Search pretty slow
  - select title from cont where match (title,content) against ('"bill gates will"' in boolean mode) limit 10;
    - Takes 18.08 sec
  - MySQL has to look at data rows which match words to check if there is phrase match so it is especially bad for frequent words in rare combination

FullText Search and Updates

- MyISAM FullText Search is based on BTREE
  - Special form of BTREE index
- Each word is index entry
- Updating text with 1000 words - 1000 key entries needs to be updated
  - A lot of random IO if index is not in memory
- Index degradation by fragmentation
  - Run **OPTIMIZE TABLE** for best performance

MySQL 5.1 FullText Parser Plugins

- Replacing Default Parser to do following:
  - Apply special rules, such as stemming, different way of splitting words etc
    - Example if you want C++ or TCP/IP to be single word
  - Do pre-parsing – processing PDF or HTML files
    - Can be used together with default parser
- May do same for query string
  - If you build index with stemming search words also need to be stemmed for search to work.

SIENNA

- Set of patches for MySQL
- Replaces MySQL FullText search on low level
  - http://qwik.jp/senna/install_en.html
- Supports MECAB for parsing eastern languages
  - http://mecab.sourceforge.jp/
- Most documentation is in Japanese
- Should have most of native Full Text search issues.
- Did not complete indexing in 24 hours

Home Baked Solutions

- Normally Light duty.
  - Hard to implement efficiently based standard database structures
- Can be helpful with special requirements and small data size.
- This is not about using BLOB to store index parts in binary form. Such solutions are external using MySQL for storage
- Nice one: http://www.firestuff.org/wordpress/

Lucene

- Popular full text search **Library** written in Java
  - http://lucene.apache.org/
  - Clucene – C port exists, but is not current
  - Is not specially tailored for indexing databases
  - Some coding is needed for integration
- Dynamic index changes possible
- Very Advanced query language
  - Wildcard searches:
  - Search in Fields: title:"The Right Way" AND text:go
  - Proximity Searches, Fuzzy Searches etc

Lucene

- Supports attributes (indexed and non indexed)
- Some CJK Support
- Easily integrates with Java
- Presentation by Grant Ingersoll
  - “Advanced Lucene”
  - Wednesday 17:00

Mnogosearch

- Mainly designed as Web search engine
  - http://www.mnogosearch.org/
- Stores index in MySQL Database
- Can be set up to index database as well
  - Relatively easy to use
- Rather slow with indexing
- Slow searching over large data sets

Sphinx Search

- Designed for indexing Database content
  - http://www.sphinxsearch.com
- Focuses
  - High performance
  - Search Quality
  - Ease of use
- Supports multi-node clustering out of box
- Support Multiple Attributes
- Different sort modes (relevance, data etc)

Sphinx Search

- Supports trend queries
- Support for snippets
- Client available as MySQL Storage Engine plugin
- Number of limitations
  - No CJK yet, no partial word searches, hard to update indexes.
- Disclaimer: I'm acting as advisor on this project and use it in our projects

Tbgsearch

- Vector based Search System
  - http://www.tbg.nu/tbgsearch/
- Written in Perl, Stores data in MySQL Database
- Very fast for medium size data sizes
- Only boolean searches available

Time for benchmarks

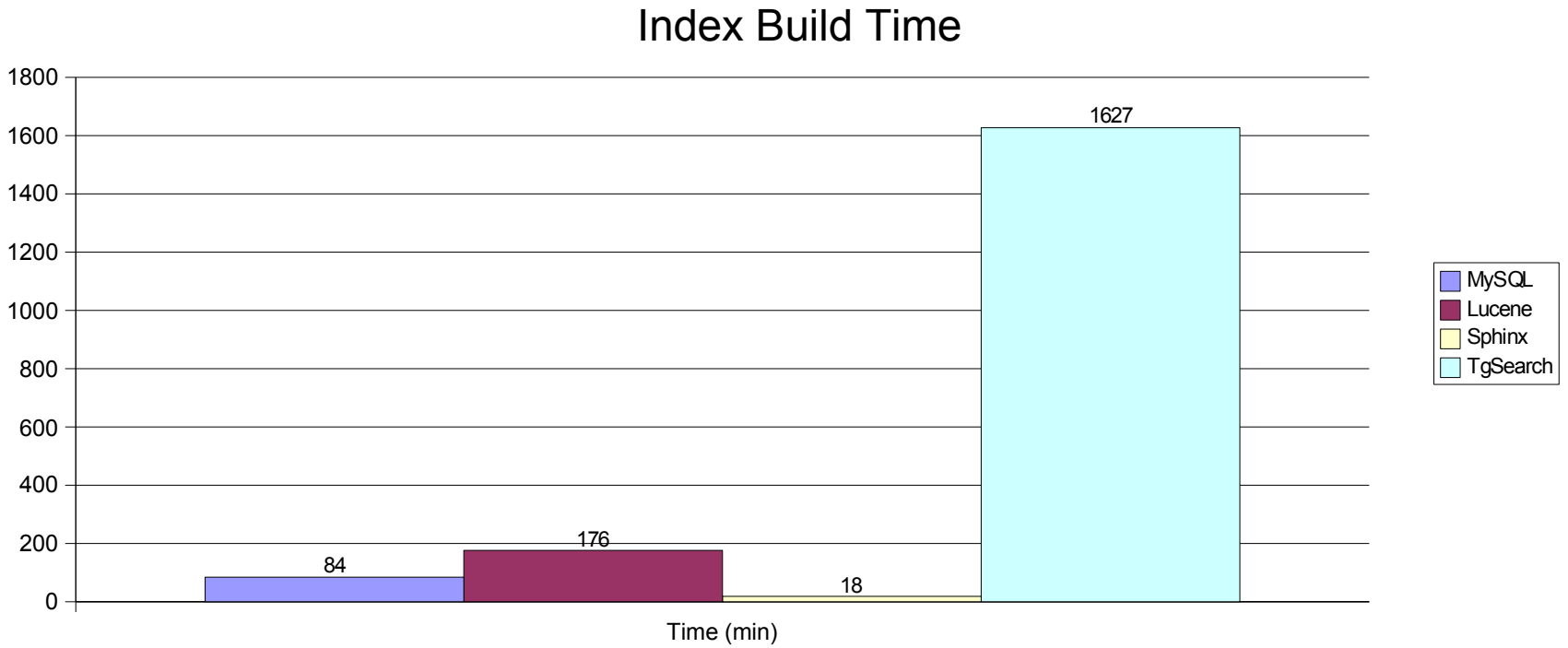
- Using described database with WIKIPEDIA Content
- Using queries from AOL published sample
  - It does not perfectly match wikipedia content but good enough.
- Executing 10.000 random queries from the log, some are more frequent than others
- Measure average query time
  - We also measured other stats but do not have time to report

Disclaimer

- As with all benchmarks results for your particular case may vary
- Full Test scripts will be published at http://www.mysqlperformanceblog.com
- We applied best effort to get best results from each engine, but experience is different
  - Expert level for Sphinx, MySQL
  - Less experience for Lucene, Mnogosearch
- Versions: Lucene 2.0, Sphinx 0.9.7, MySQL 5.0.24

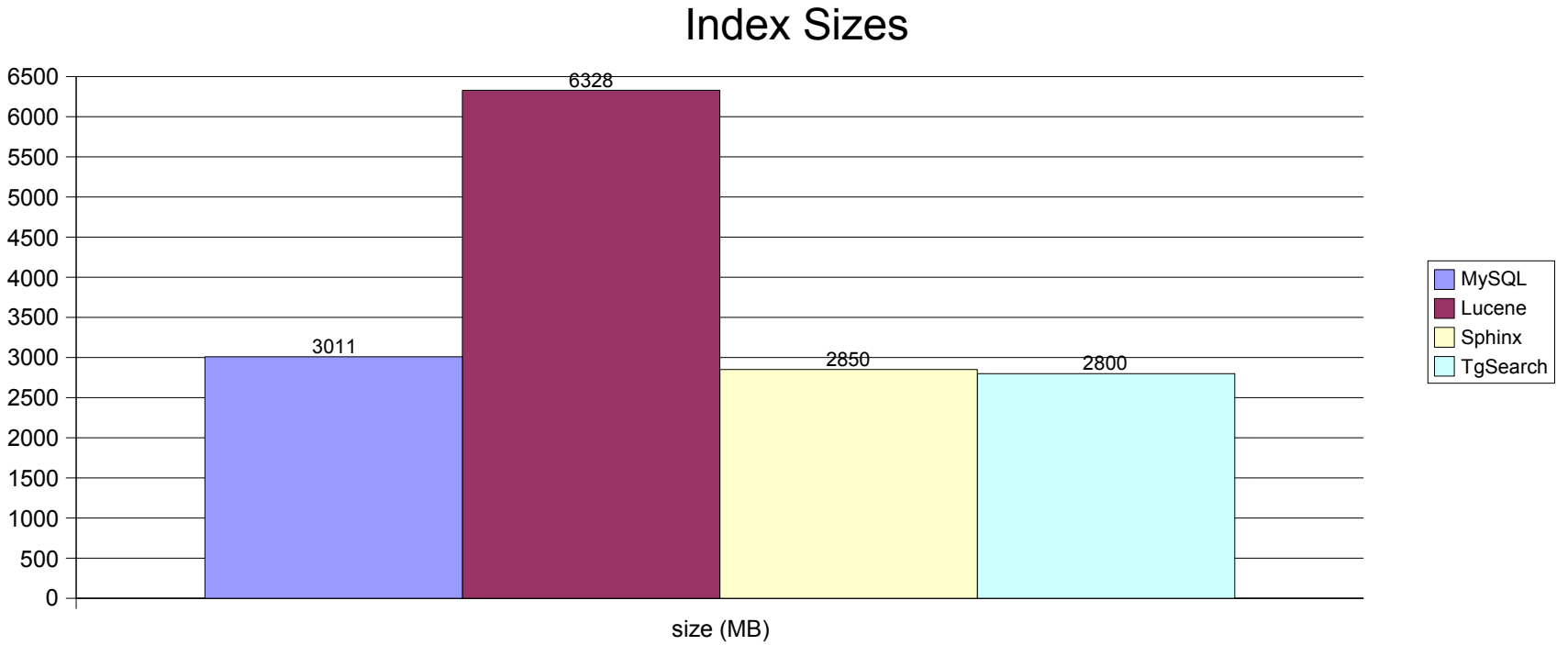
Index Build Speed

- Building index on data already in table
  - MySQL FullText Search does full table rebuild
  - Sienna, Mnogosearch failed to complete in 24h



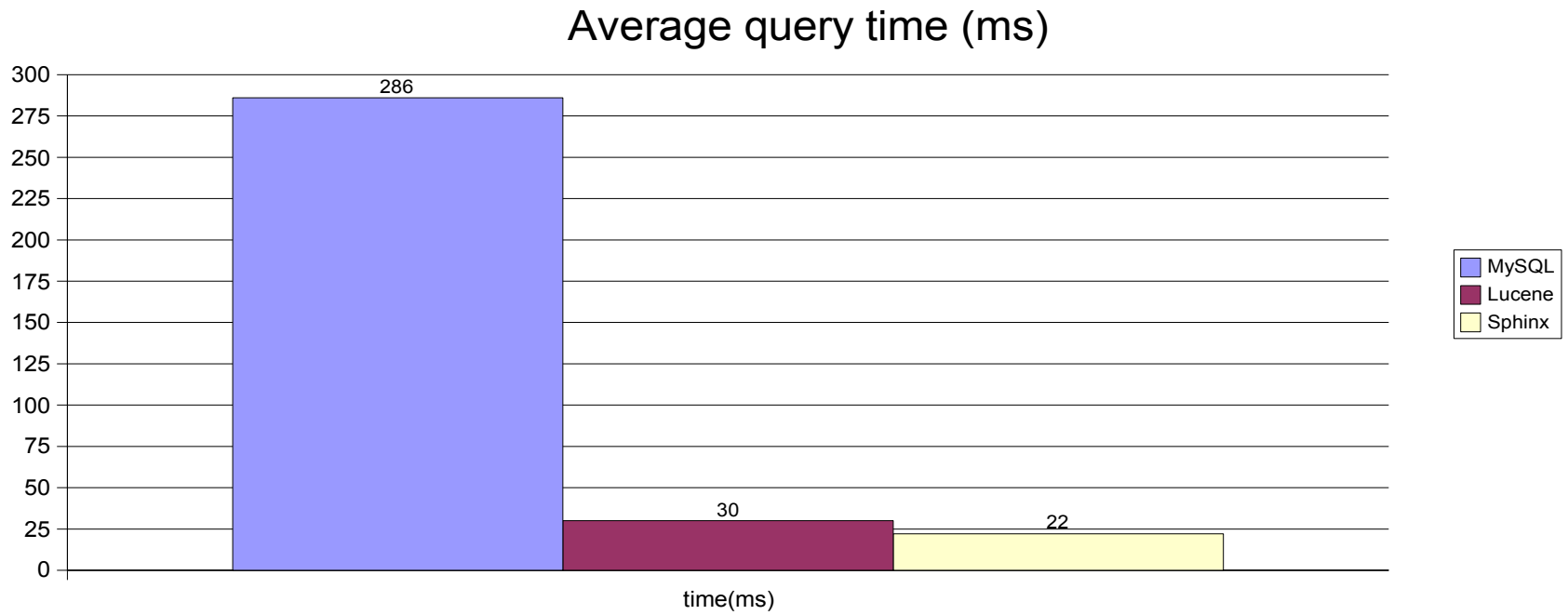
Index Sizes

- IO bound and CPU bound workload.
- Different data in Index
  - MySQL,TgSearch does not have word positions, Lucene and Sphinx have.



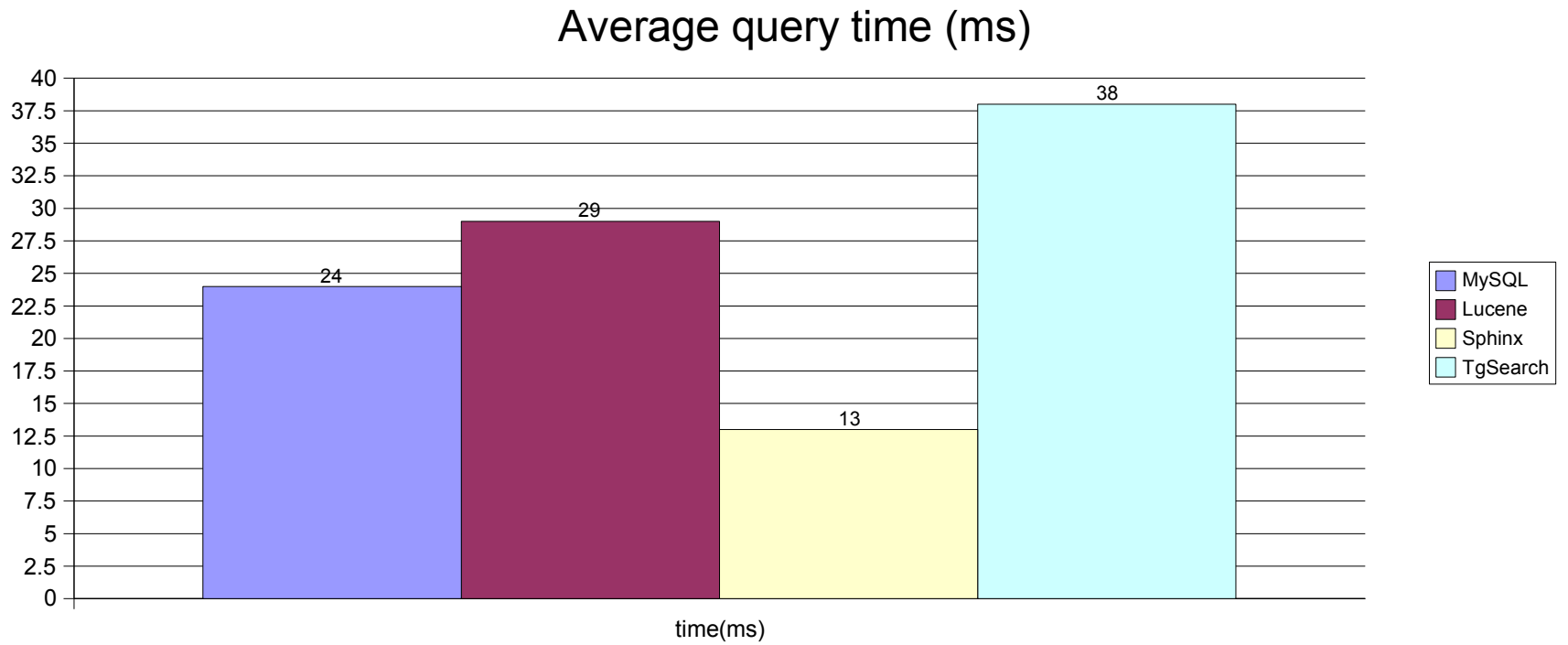
Standard Search Performance

- Search performed with default relevance sorting
  - Which is different for different engines
- First 20 rows requested, no count for MySQL



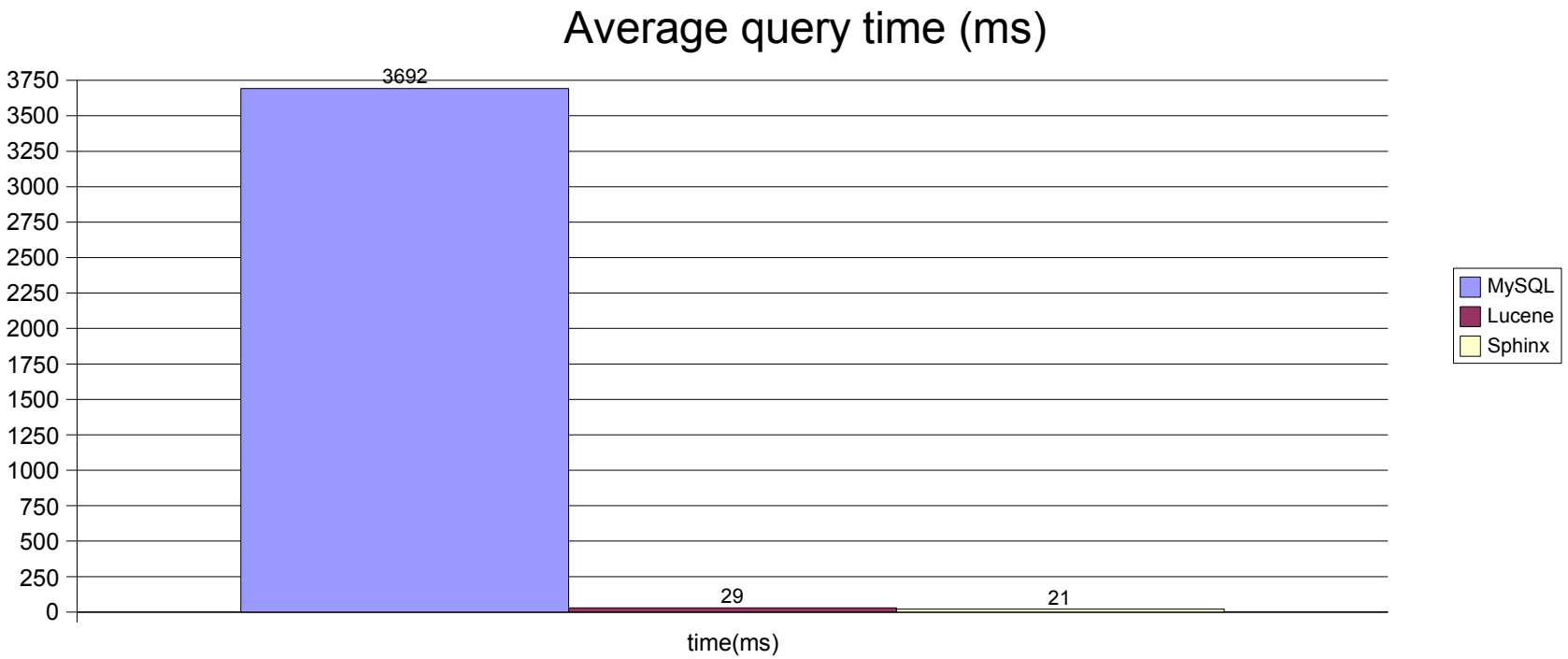
Boolean Search

- 20 Matches, Do not care about relevance
- Good fair way to compare speed
- Smaller working set
- Only results available for TgSearch



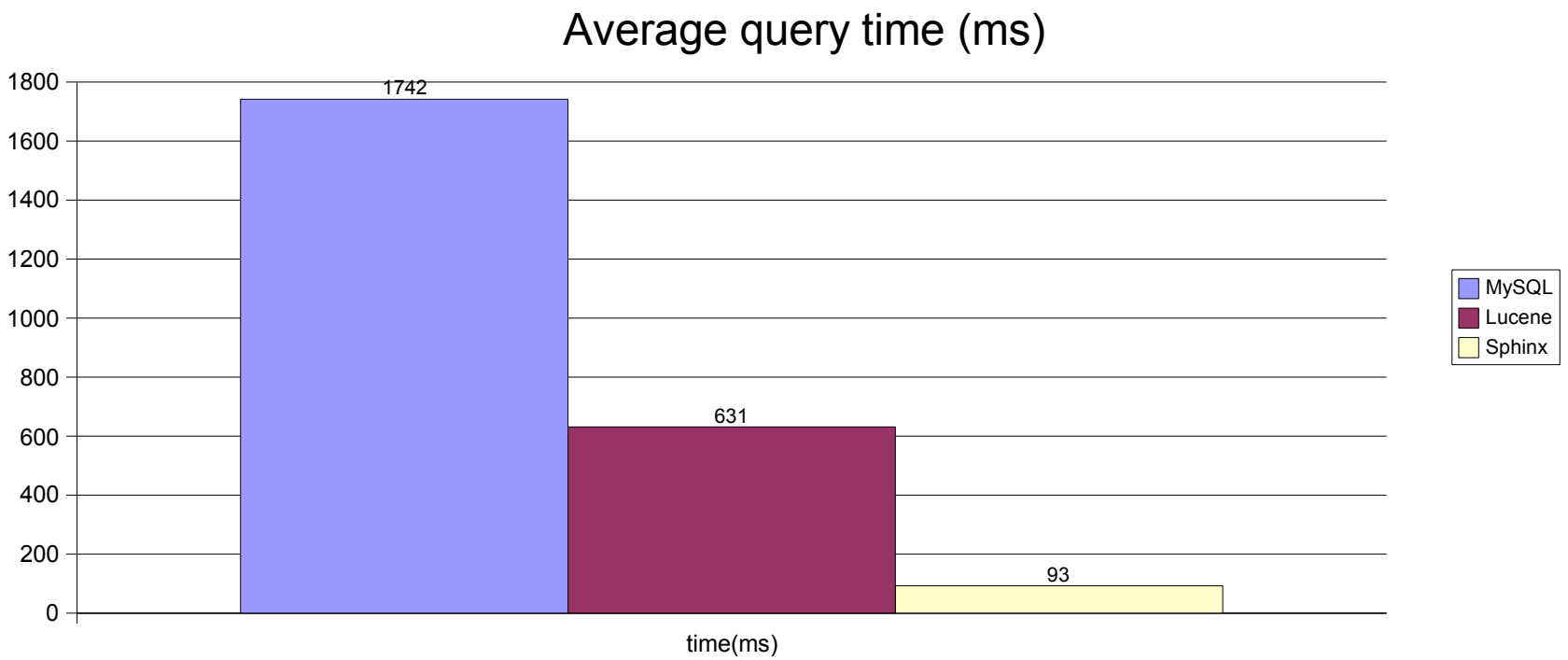
Phrase Search

- More resource intensive for standard index types
- MySQL does with boolean search
  - No relevance sorting applied



Multi User Search

- Running 10 test applications concurrently



My Thoughts on Performance

- MySQL FullText Search is good for
  - Small (relatively) data sizes
  - If you can use a lot of hardware
  - Your application is simple – No complex sorts etc.
- Lucene and Sphinx both can handle large data sizes
  - Sphinx is easier to use, has clustering out of the box, snippet generation
  - Lucene has more features, dynamic updates

General Optimization Notices

- Cache Search results
- Perform search in the parallel on portions of data
  - Each engine has its limit of hits/CPU it can process
- Use stop words if possible
- Keep search engine engine fragmentation low
  - Random IO is always slower
- Try to have your working set in memory

Beyond simple FullText Search

- FullText search indexes allow fast set operation.
  - Use Codes for object properties and **BOOLEAN** search for retrieval
- Codes can be used to filter search result:
  - AGAINST ("+GROUP124 +MySQL +Book")
- Very efficient for handling tags
  - Custom parser, or replace spaces with something
  - Also makes your queries much simple when it comes to tags union and intersections

Thanks For Coming !

- Time for questions !
- Contacts:
  - http://www.mysqlperformanceblog.com
  - pz@mysqlperformanceblog.com