groonga
Fulltext Searching
with
Groonga Storage Engine
Brazil Inc.
groonga is...

- a Fulltext Search Library
- for Cloud and Web
Characteristics

• Easy to Embed and Scalable
• Highly Precise Search for Any Language
• Fast Searching and Indexing in Realtime
1. groonga core
Easy to Embed and Scalable

- a Small Library Written in C
- Highly CPU Scalable
- Applicable to Various Process/Thread Model Systems
The diagram illustrates the relationship between various components in a client-server architecture. Starting from the top, there is a client component that connects to two databases: PostgreSQL and MySQL. MySQL is further connected to two more components: InnoDB and Groonga Column Store. Groonga is also connected to Ruby, which contains a Groonga Column Store and Groonga Core. PostgreSQL is also connected to Groonga Core directly. The connections are represented by arrows indicating the direction of data flow.
How to scale it

• Sharding and Replication is the best and only solution
• for Large Scale Search Engines
• since B.G. era
groonga supports...

- No Built-in Scale-Out Solution
- Suppose to be Embedded Into Various Solutions, like Spider
Highly Precise Search for Any Language

• Importance of Precision
• As for Speed, Money talks
• As for Precision, it can't
Highly Precise Search for Any Language

• Unsegmented Languages need
• not only Pluggable Tokenizers
• but also Solutions for Vague Boundaries Problem
Vague Boundaries Problem

ここではきものをぬいてください

ここで / はきものを / ぬいで / ください。Put off your shoes here.

ここでは / きものを / ぬいで / ください。Put off your clothes here.
groonga supports...

Full Inverted Index
✓ inevitable for unsegmented languages

Highly Compressed Index
✓ no stop words needed

Patricia TRIE Lexicon
✓ Partial string match on lexicon
Fast Searching and Indexing in Realtime

• Static Indexing was Mainstream in Full Inverted Index

• Dynamic(Realtime) Indexing is in Great Demand for Web
Fast Searching and Indexing in Realtime

• Web is Growing Exponentially Every Moment

• And Millions of Users Need to Search from Among Them

• Searching and Indexing must be Performed Simultaneously
groonga supports...

- Lock Free Index Update
- Indexing Never Block Searching
Performance of Simultaneous Indexing and Searching
2. groonga column store
groonga column store

• Why Own Data Store?
• No Storage Engines were available
• Suitable for Typical Search Engine Queries
Typical Search Engine Query

- hits large number of records
- filtered by multiple conditions
- group by specific conditions
- order by a dynamic condition
- output limited number of records
Groonga Column Store supports...

- Reduce I/O Cost of Performing Filter/Sort/Group Operations

- Especially the Number of Hits Become Very Large

- Fast Range Search and Geo Search are also Available
groonga
storage engine

Brazil, Inc.
Tasuku SUENAGA
a.k.a. gunyarakun
What I talk about

MySQL

Spider storage engine

MySQL

groonga storage engine

MYISAM
InnoDB

groonga column store

groonga core
MySQL fulltext index

- Phrase search is slow.
- Updating index is slow.
- Cannot combine full text search index with other indexes (like B-Tree).

Our prior product Tritonn solves.
Tritonn

- Tritonn = groonga + patches for MyISAM
## MySQL v.s. Tritonn

Target dataset: Wikipedia English 458,713 record 1088MB

<table>
<thead>
<tr>
<th></th>
<th>MySQL(5.0)’s Fulltext Index</th>
<th>Tritonn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index size</td>
<td>109 MB</td>
<td>1028 MB</td>
</tr>
<tr>
<td>Phrase search for ‘united states’</td>
<td>44.91 sec</td>
<td>0.40 sec</td>
</tr>
<tr>
<td>Indexing after inserting recs</td>
<td>1,474 sec</td>
<td>1,808 sec</td>
</tr>
<tr>
<td>Inserting recs after idx. creation</td>
<td>28,182 sec</td>
<td>1,839 sec</td>
</tr>
<tr>
<td>Where MATCH AGAINST and order by primary key</td>
<td>20.33 sec</td>
<td>0.89 sec</td>
</tr>
<tr>
<td>Where MATCH AGAINST and primary key &gt; 200000</td>
<td>6.55 sec</td>
<td>0.32 sec</td>
</tr>
</tbody>
</table>
So Tritonn provides ...  

- Fast phrase search  
- Fast index update (realtime)  
- Works well with other indexes.

But some problems remain.
Remaining problems

- MyISAM based
  - Table lock
    - when updating table, read accesses are blocked.

- Patch based
  - Patch maintainance and building patched MySQL is too messy.

Need for a new solution.
New solution is

- groonga storage engine
  - Use column store of groonga instead of MyISAM.
  - Not patch but storage engine.

Tritonn (old)  
groonga storage engine(new)
Advantage

- Table lock free
  - Column store of groonga is lock-free.
- Only access columns required
  - Not row-based.
- Easy to build and develop

And some optimization for typical queries
Optimization(1)

- COUNT(*) optimization.
  - For queries like below.

```
SELECT COUNT(*) FROM table
WHERE MATCH(col) AGAINST ('query');
```
Optimization(2)

- ORDER BY score and LIMIT optimization.
  - For queries like below.

```sql
SELECT * FROM table
WHERE MATCH(col) AGAINST ('query')
ORDER BY MATCH(col) AGAINST ('query')
LIMIT 10;
```
Conclusion of my part

- groonga storage engine provides
  - Fast phrase search
  - Fast index update (realtime)
  - Inserting records doesn’t block reading records
The combination of Groonga and Spider

Kentoku SHIBA
kentokushiba at gmail dot com
The combination of Groonga and Spider

In this time ...

MySQL

Spider storage engine

MySQL

groonga storage engine

groonga column store

groonga core
Spider Storage Engine is a storage engine for database sharding transparently.

1. Request
2. Just connect to spider
3. Response
The combination of Groonga and Spider

You can get following power by combination of Groonga and Spider.

- The optimization for the fulltext searching with sorting by score.

- The optimization for the sorting by range partition key column.

- The optimization for the fulltext searching with filtering by partition key column.
The optimization for the fulltext searching with sorting by score
(The case of scanning all partitions)
Sorting by score

- Parallel searching
- 2 step limitation

SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
ORDER BY _score LIMIT 100;

SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
ORDER BY _score LIMIT 100;

Parallel searching is coming soon.
The optimization for the sorting by range partition key column
(coming soon)
The sorting by range partition key column

```
SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
ORDER BY c1 LIMIT 100;
```

- Sort optimization with range partition
The optimization for the fulltext searching with filtering by partition key column
The filtering by partition key column

- Partition pruning

```
SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
AND c1 = 60
ORDER BY _score LIMIT 100;
```

```
SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
AND c1 = 60
ORDER BY _score LIMIT 100;
```

```
SELECT * FROM t1
WHERE MATCH(c2) AGAINST('hoge')
AND c1 = 60
ORDER BY _score LIMIT 100;
```
End of the session
If you want to try introduced Spider features, you can download from here and try.

source code
http://groonga.org/pkg/mysql-5.5.8-spider-2.24h-vp-0.13-hs-1.0.src.tgz

binary (Linux x86_64 glibc2.3)
http://groonga.org/pkg/mysql-5.5.8-spider-2.24h-vp-0.13-hs-1.0.bin.tgz

initialize SQL
http://groonga.org/pkg/spider-init-2.24-for-5.5.8.tgz
Contact us

If you have some questions, comments or suggestions, please contact us from here.

Thank you for taking your time!!

Daijiro MORI (morita at razil dot jp)
Tasuku SUENAGA (a at razil dot jp)
Kentoku SHIBA (kentokushiba at gmail dot com)