trbmeetup

Fast fulltext search in Ruby, without Java
-Groonga, Rroonga and Droonga-

YUKI Hiroshi

ClearCode Inc.

Abstract

- Fulltext search?
- Groonga and Rroonga
    - easy fulltext search in Ruby
- Droonga
    - scalable fulltext search

Introduction

What's
fulltext search?

Searching without index

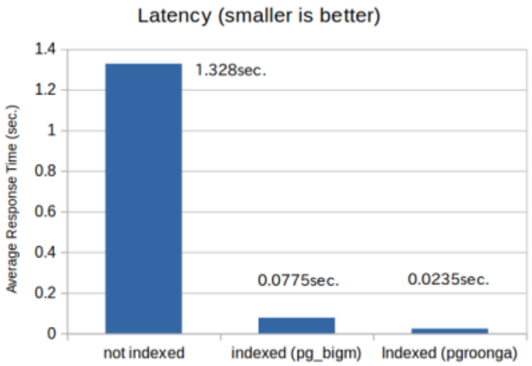
ex. Array#grep
ex. LIKE operator in SQL

```
SELECT name, location
  FROM Store
 WHERE name LIKE '%Tokyo%';
```

- easy, simple, but **slow**

Fulltext search w/ index

- Fast!!



Demonstration

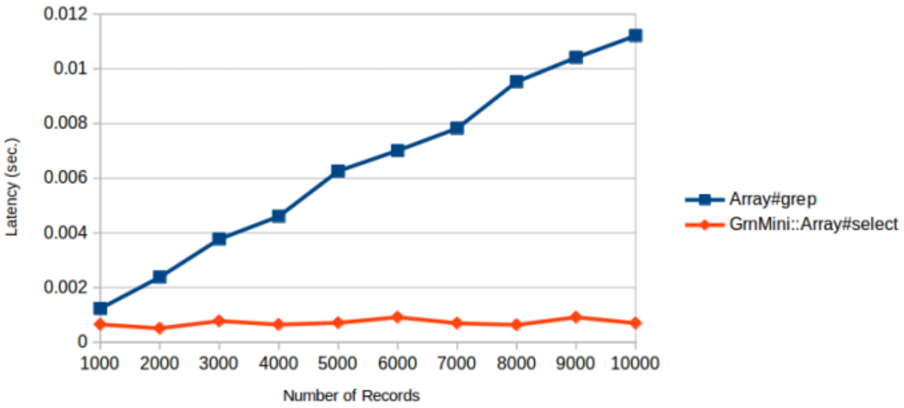
Methods

- ✓ `Array#grep` (not indexed)
- ✓ `GrnMini::Array#select` (indexed)

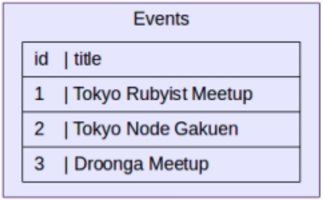
Data

- ✓ Wikipedia(ja) pages

Demonstration: Result

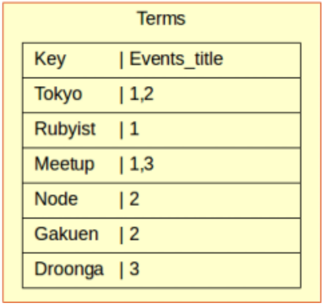


Off topic: why fast?



Events

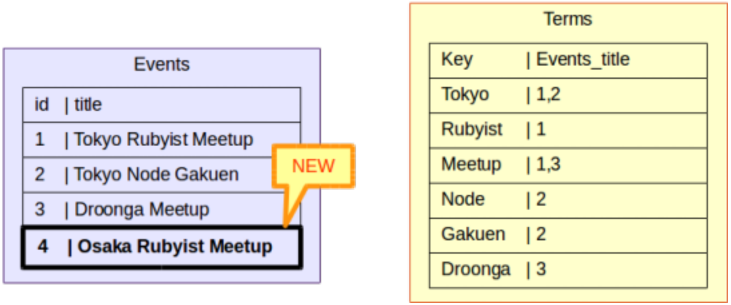
id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup



Terms

Key	Events_title
Tokyo	1,2
Rubyist	1
Meetup	1,3
Node	2
Gakuen	2
Droonga	3

Off topic: why fast?



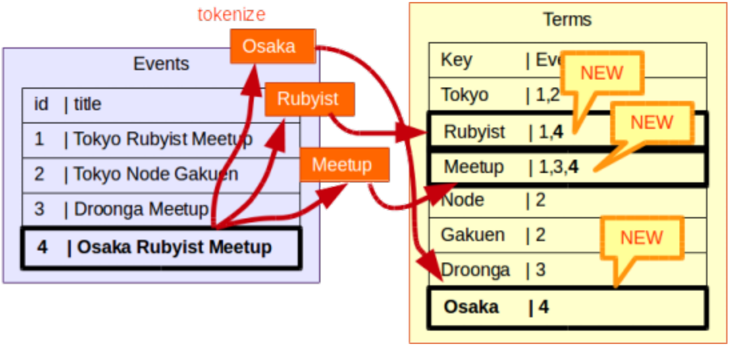
Events

id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup
4	**Osaka Rubyist Meetup**

Terms

Key	Events_title
Tokyo	1,2
Rubyist	1
Meetup	1,3
Node	2
Gakuen	2
Droonga	3

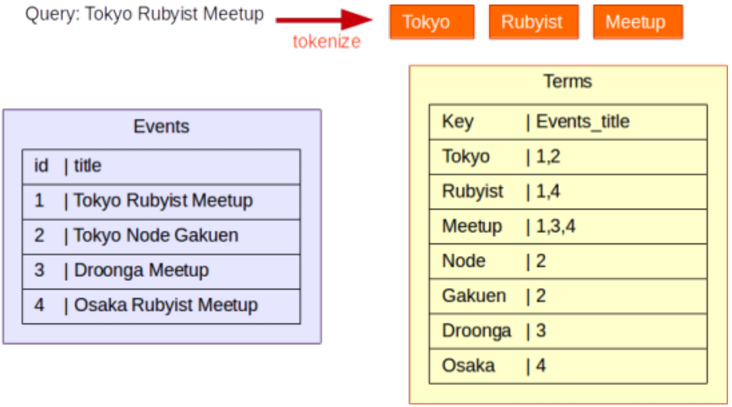
Off topic: why fast?



id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup
4	**Osaka Rubyist Meetup**

Key	Eve
Tokyo	1,2
Rubyist	1,4
Meetup	1,3,4
Node	2
Gakuen	2
Droonga	3
Osaka	**4**

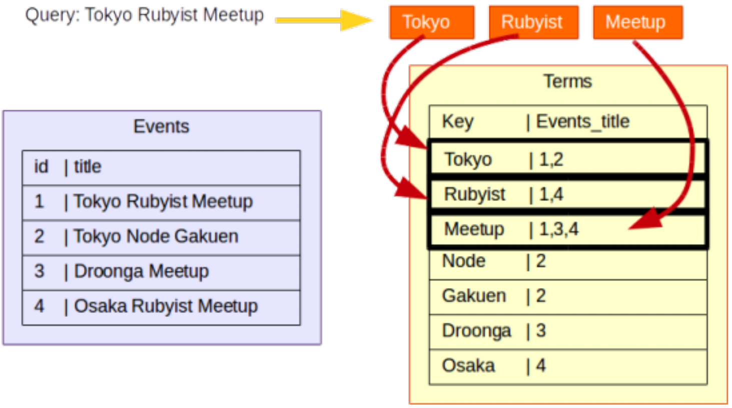
Off topic: why fast?



id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup
4	Osaka Rubyist Meetup

Key	Events_title
Tokyo	1,2
Rubyist	1,4
Meetup	1,3,4
Node	2
Gakuen	2
Droonga	3
Osaka	4

Off topic: why fast?


Query: Tokyo Rubyist Meetup

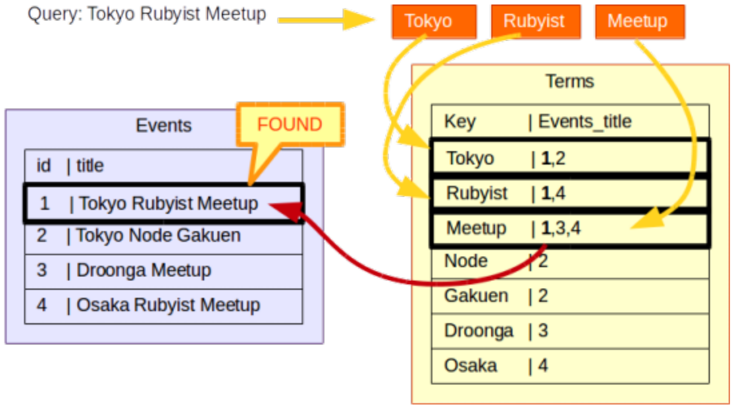
Events

id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup
4	Osaka Rubyist Meetup

Terms

Key	Events_title
Tokyo	1,2
Rubyist	1,4
Meetup	1,3,4
Node	2
Gakuen	2
Droonga	3
Osaka	4

Off topic: why fast?



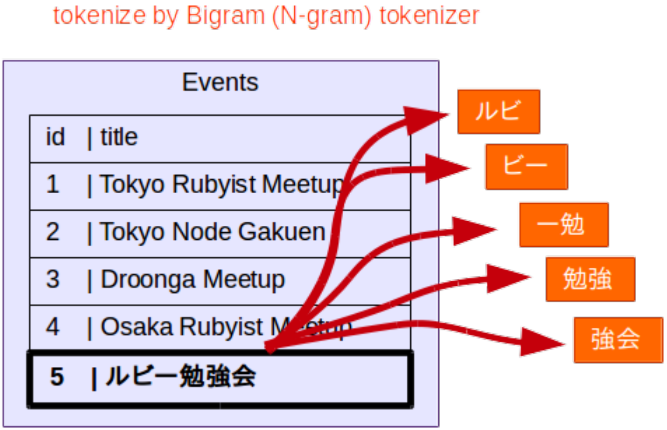
Events

id	title
1	Tokyo Rubyist Meetup
2	Tokyo Node Gakuen
3	Droonga Meetup
4	Osaka Rubyist Meetup

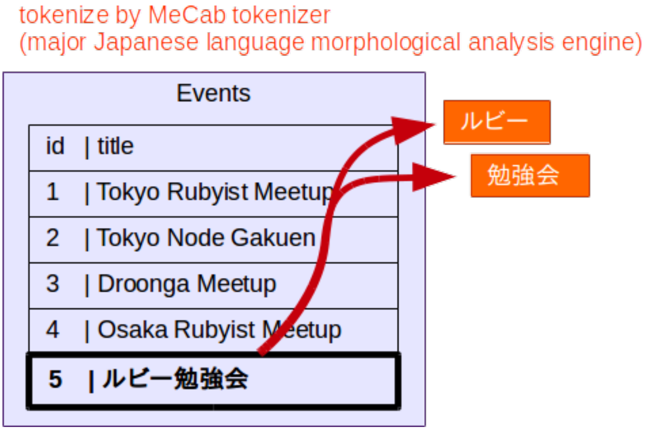
Terms

Key	Events_title
Tokyo	**1**,2
Rubyist	**1**,4
Meetup	**1**,3,4
Node	2
Gakuen	2
Droonga	3
Osaka	4

Off topic: why fast?



Off topic: why fast?



How introduce?

Major ways

- Sunspot
- elasticsearch-ruby

Sunspot?

A client library of
Solr
for Ruby and Rails
(ActiveRecord)

Sunspot: Usage

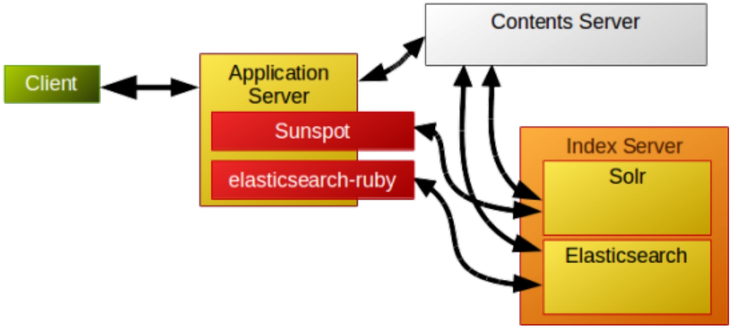
```
class Post < ActiveRecord::Base
  searchable do
    # ...
  end
end

result = Post.search do
  fulltext 'best pizza'
  # ...
end
```

elasticsearch-ruby?

A client library of **Elasticsearch** for Ruby

```
client = Elasticsearch::Client.new(log: true)
client.transport.reload_connections!
client.cluster.health
client.search(q: "test")
```

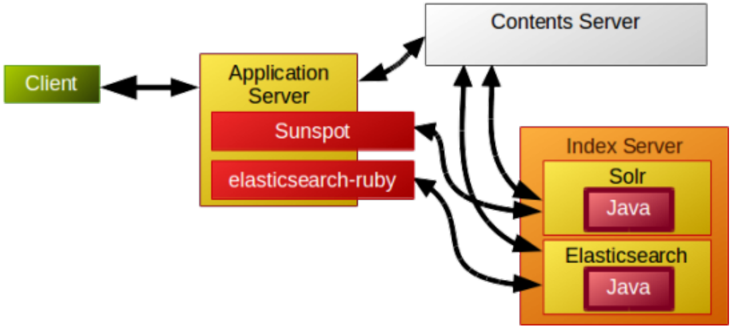
Relations of services



But…

- Apache Solr: “built on Apache Lucene™.”
- Elasticsearch: “Build on top of Apache Lucene™”
- Apache Lucene: “written entirely **in Java**.”

Java!!


Contents Server
Client
Application Server
Sunspot
elasticsearch-ruby
Index Server
Solr
Java
Elasticsearch
Java

In short

- They require **Java**.
- My Ruby product have to be combined with **Java**, just for fulltext search.

Alternative choice

Groonga
and
Rroonga

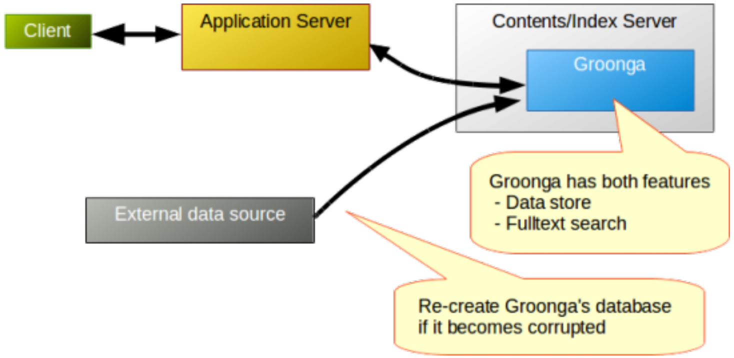
Groonga

- Fast fulltext search engine written in **C**
- Originally designed to search increasing huge numbers of comments in “2ch” (like Twitter)

Groonga

- Realtime indexing
  - Read/write lock-free
  - Parallel updating and searching, without penalty
  - Returns latest result ASAP
- No transaction
  - No warranty for data consistency

Relations of services



Groonga's interfaces

via command line interface

```
$ groonga="groonga /path/to/database/db"
$ $groonga table_create --name Entries
    --flags TABLE_PAT_KEY --key_type ShortText
$ $groonga select --table Entries
                  --query "title:@Ruby"
```

Groonga's interfaces

via HTTP

```
$ groonga -d --protocol http --port 10041
                                 /path/to/database/db

$ endpoint="http://groonga:10041"
$ curl "${endpoint}/d/table_create?name=Entries&
            flags=TABLE_PAT_KEY&key_type=ShortText"
$ curl "${endpoint}/d/select?table=Entries&
                               query=title:@Ruby"
```

Groonga's interfaces

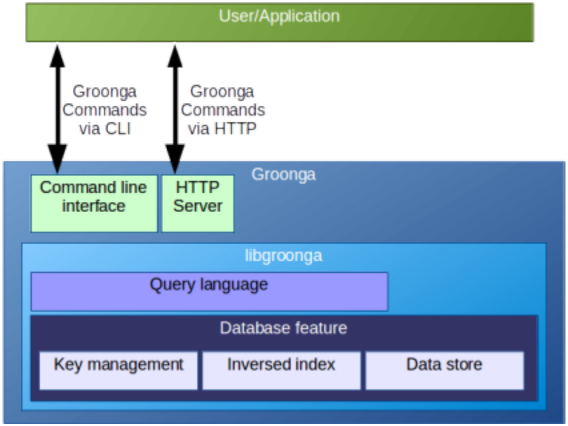
Narrowly-defined "Groonga"

- ✓ CLI or server

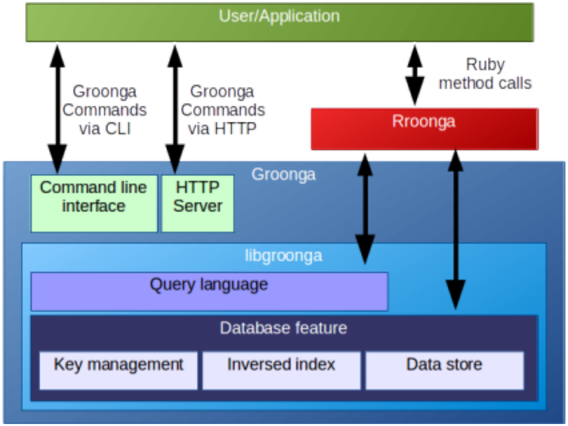
libgroonga

- ✓ In-process library
- ✓ Like as "better SQLite"

Groonga


User/Application
Groonga Commands via CLI
Groonga Commands via HTTP
Groonga
Command line interface
HTTP Server
libgroonga
Query language
Database feature
Key management
Inversed index
Data store

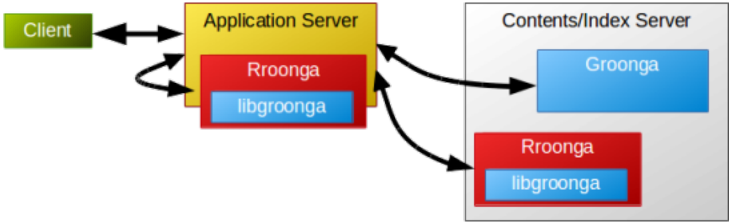
Rroonga


User/Application
Groonga Commands via CLI
Groonga Commands via HTTP
Ruby method calls
Rroonga
Groonga
Command line interface
HTTP Server
libgroonga
Query language
Database feature
Key management
Inversed index
Data store

Rroonga

- Based on libgroonga
- Low-level binding of Groonga for **Ruby**

Relations of services



Usage: Install

```
% sudo gem install rroonga
```

Groonga (libgroonga) is also installed as a part of the package.

Usage: Prepare

```
require "groonga"

Groonga::Database.create(path: "/tmp/bookmark.db")
# Or
Groonga::Database.open("/tmp/bookmark.db")
```

Usage: Schema

```
Groonga::Schema.define do |schema|
  schema.create_table("Items",
                      type:     :hash,
                      key_type: "ShortText") do |table|
    table.text("title")
  end
  schema.create_table("Terms",
                      type:              :patricia_trie,
                      normalizer:        "NormalizerAuto",
                      default_tokenizer: "TokenBigram") do |table|
   table.index("Items.title")
  end
end
```

Usage: Data loading

```
items = Groonga["Items"]
items.add("http://en.wikipedia.org/wiki/Ruby",
          title: "Wikipedia")
items.add("http://www.ruby-lang.org/",
          title: "Ruby")
```

Usage: Fulltext search

```
items = Groonga["Items"]
ruby_items = items.select do |record|
  record.title =~ "Ruby"
end
```

FYI: GrnMini

- Lightweight wrapper for Rroonga
- Limited features, but easy to use

FYI: GrnMini: Code

```ruby
require "grn_mini"

GrnMini::create_or_open("/tmp/bookmarks.db")

items = GrnMini::Array.new("Items")
items << { url:   "http://en.wikipedia.org/wiki/Ruby",
           title: "Ruby - Wikipedia" }
items << { url:   "http://www.ruby-lang.org/",
           title: "Ruby Language" }

ruby_items = items.select("title:@Ruby")
```

Good first step to try fulltext search in your Ruby product.

For much more load…

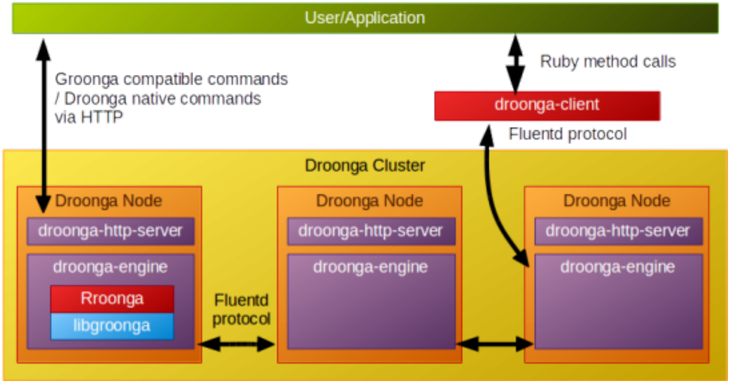
Groonga

works with **single process** on a computer

Droonga

works with **multiple computers** constructiong a Droonga cluster

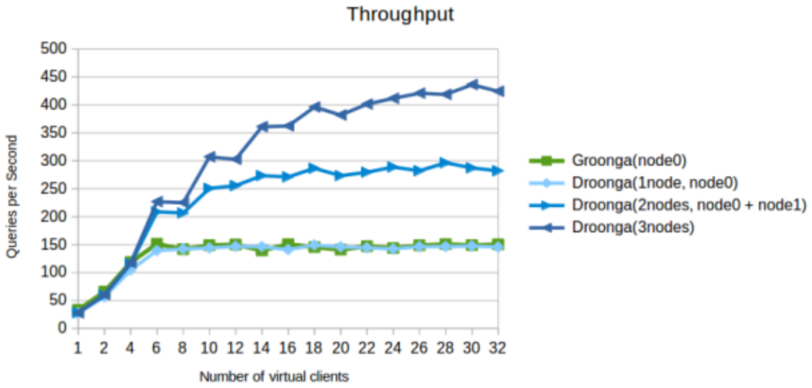
Droonga


User/Application
Ruby method calls
Groonga compatible commands / Droonga native commands via HTTP
droonga-client
Fluentd protocol
Droonga Cluster
Droonga Node
droonga-http-server
droonga-engine
Rroonga
libgroonga
Fluentd protocol
Droonga Node
droonga-http-server
droonga-engine
Droonga Node
droonga-http-server
droonga-engine

Droonga

- Scalable
  (replication + partitioning)
- Groonga compatible
  HTTP interface
- Client library for Ruby
  (`droonga-client`)

Droonga


Throughput
Queries per Second
500
450
400
350
300
250
200
150
100
50
0
1 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
Number of virtual clients
Groonga(node0)
Droonga(1node, node0)
Droonga(2nodes, node0 + node1)
Droonga(3nodes)

Usage of Droonga

Setup a Droonga node

```
# base="https://raw.githubusercontent.com/droonga"
# curl ${base}/droonga-engine/master/install.sh | ¥
    bash
# curl ${base}/droonga-http-server/master/install.sh | ¥
    bash
# droonga-engine-catalog-generate --hosts=node0,node1,node2
# service droonga-engine start
# service droonga-http-server start
```

Usage of Droonga

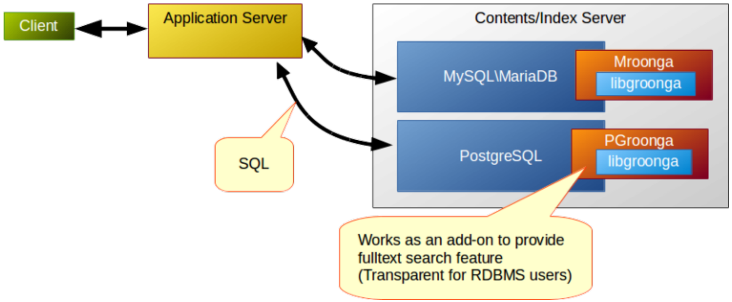
Fulltext search via HTTP
(compatible to Groonga)

```
$ endpoint="http://node0:10041"
$ curl "${endpoint}/d/table_create?name=Store&
        flags=TABLE_PAT_KEY&key_type=ShortText"
```

More chices

- Mroonga
  - Add-on for MySQL/MariaDB (Bundled to MariaDB by default)
- PGroonga
  - Add-on for PostgreSQL

Relations of services



SQL w/ fulltext search

Mroonga

```
SELECT name,location
  FROM Store
 WHERE MATCH(name)
     AGAINST('+東京' IN BOOLEAN MODE);
```

SQL w/ fulltext search

PGroonga

```
SELECT name,location
  FROM Store WHERE name %% '東京';

SELECT name,location
  FROM Store WHERE name @@ '東京 OR 大阪';

SELECT name,location
  FROM Store WHERE name LIKE '%東京%';
/* alias to "name @@ '東京'"*/
```

Conclusion

- **Rroonga** (and **GrnMini**) introduces fast fulltext search into your Ruby product instantly
- **Droonga** for increasing load
- **Mroonga** and **PGroonga** for existing RDBMS

References

Sunspot

http://sunspot.github.io/

elasticsearch-ruby

https://github.com/elasticsearch/elasticsearch-ruby

References

Apache Lucene

http://lucene.apache.org/

Apache Solr

http://lucene.apache.org/solr/

Elasticsearch

http://www.elasticsearch.org/overview/elasticsearch/

References

Groonga

http://groonga.org/

Rroonga

http://ranguba.org/

GrnMini

https://github.com/ongaeshi/grn_mini

References

Droonga

http://droonga.org/

Mroonga

http://mroonga.org/

PGroonga

http://pgroonga.github.io/

References

Comparison of PostgreSQL, pg_bigm and PGroonga

http://blog.createfield.com/entry/2015/02/03/094940

Advertisement

- Serial comic at Nikkei Linux
- 2015.2.18 Release
- ¥1728 (tax-inclusive)
- Paper/Kindle