Sergio Bossa

@sbtourist

Terrastore

A document database for developers

terrastore

About Me

- **Software architect and engineer**
  - Bwin Italy (online gambling and casinos).
- **Long time open source enthusiast and contributor**
  - Spring.
  - Taconite.
  - Terracotta.
  - Terrastore.
- **(Micro)-Blogger**
  - http://twitter.com/sbtourist
  - http://sbtourist.blogspot.com

NOSQL ... why?

When you're in troubles with ...

- **Data Model.**
  - Relational mismatch.
  - Variable schema.
- **Data Access Patterns.**
  - Expensive joins.
  - Denormalized data.
- **Scalability.**
  - More data.
  - More processing.

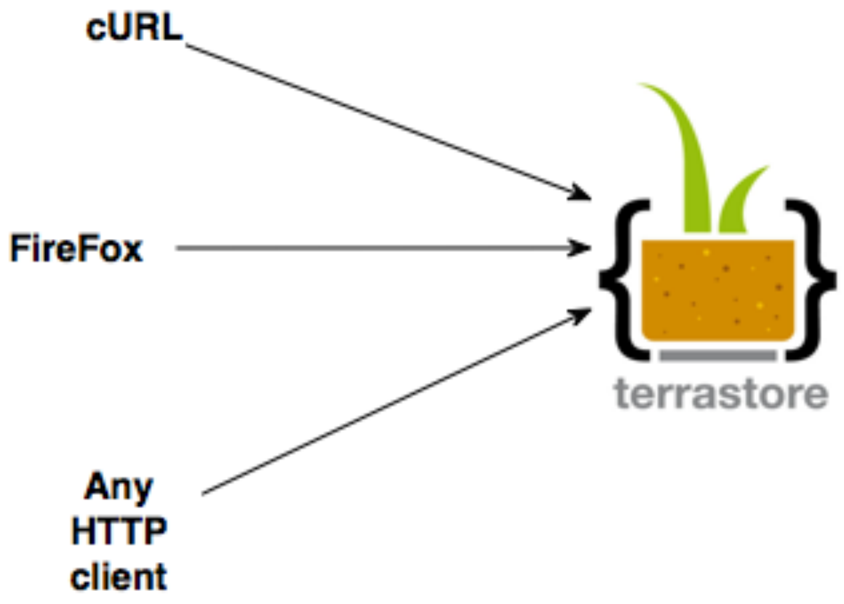
Terrastore!

- Document Store.
  - Ubiquitous.
  - Consistent.
  - Distributed.
  - Scalable.
- Written in Java.
- Based on Terracotta.
- Open Source.
- Apache-licensed.

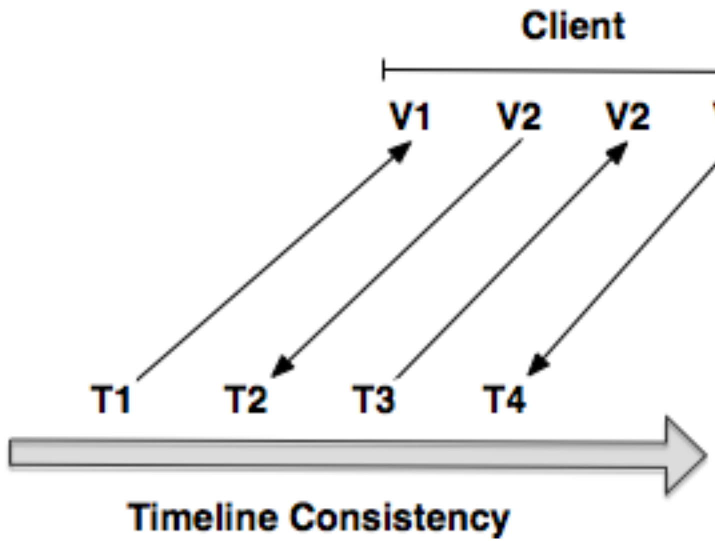
Your data, your documents

```
{
    "name" : "Sergio",
    "surname" : "Bossa",
    "twitter" : "@sbtourist"
}
```

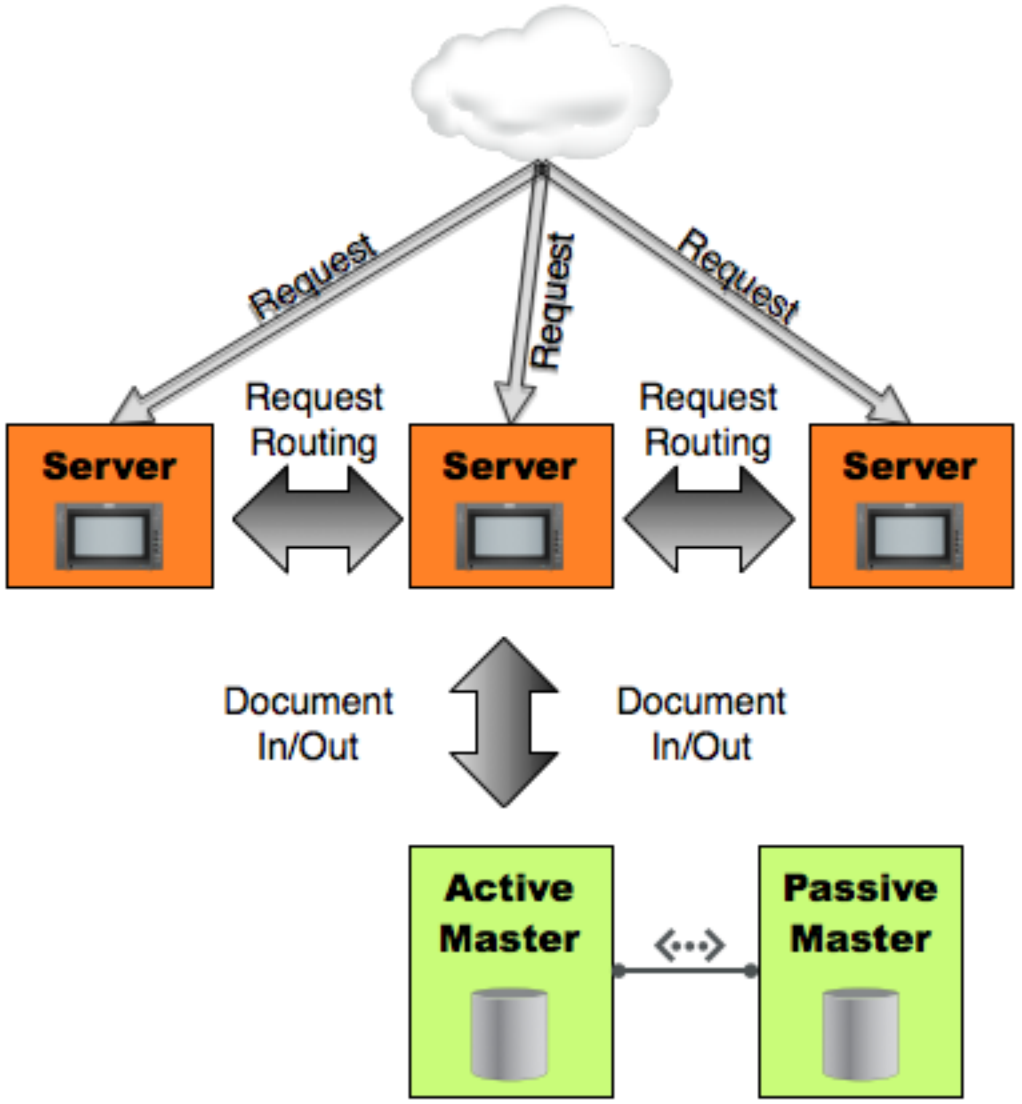
Access everywhere



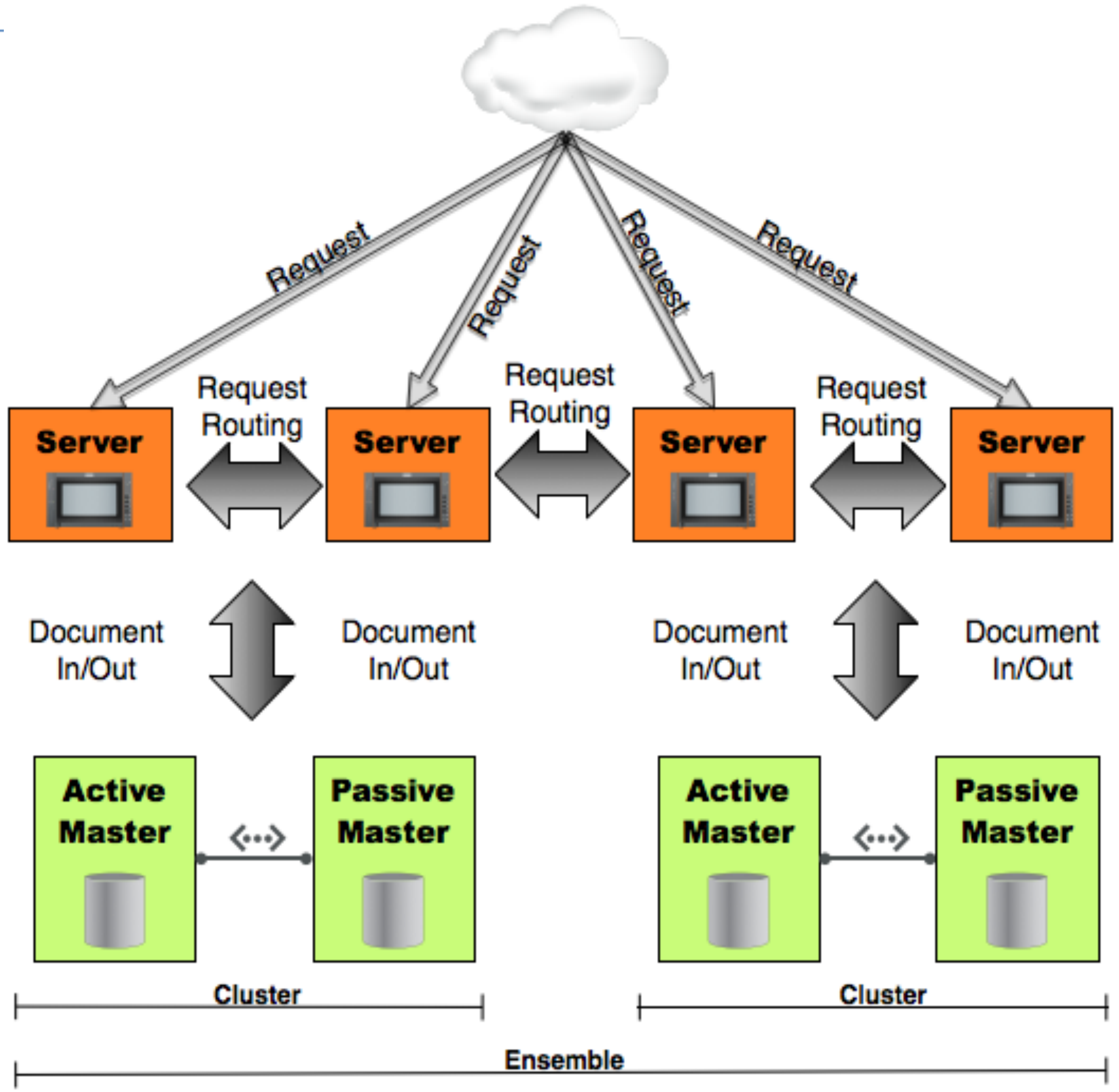
Keep consistent



Distribute as a single cluster



Distribute as multiple clusters



But ... Why Terrastore?!?!

Get started in seconds

- One command installation and startup ...

```
$> ant -f terrastore-install.xml \
quickstart -Dquickstart.dir=...
```

Easy installation

- Master:

```
$> ant -f terrastore-install.xml \
single-master \
-Dmaster.server.port 9510 \
-Dinstall.dir=...
```

- Server:

```
$> ant -f terrastore-install.xml \
server \
-Dinstall.dir=...
```

No complex configuration

- Master:

```
$master>./start.sh
```

- Server:

```
$server>./start.sh \
--master \
--httpHost \
--httpPort \
--nodeHost \
--nodePort \
...
```

Easy scale-out

- A command line parameter:

```
$server>./start.sh \
--master \
--ensemble
```

- And a json configuration file:

```
{
"localCluster" : "apple",
"discoveryInterval" : 5000,
"clusters" : ["apple", "orange"],
"seeds" : {"orange" : "192.168.1.2:6001"}
}
```

No impedence mismatch

- Java:

```
public class Character {
    private String name;
    private List<Character> friends;
    private List<Character> foes;
    // ...
}
```

- Json:

```
{"name" : "Spider-man",
"friends" : [{"name" : "Iceman"}]
"foes" : [{"name" : "Kingpin"}]}
```

No distracting (meta)data

- No:

```
{
"version" : "1", "timestamp" : "12345",
"data" :
{"name" : "Spider-man",
"friends" : [{"name" : "Iceman"}]
"foes" : [{"name" : "Kingpin"}]}
}
```

- Just:

```
{"name" : "Spider-man",
"friends" : [{"name" : "Iceman"}]
"foes" : [{"name" : "Kingpin"}]}
```

Simple basic operations

- Put documents in buckets ...

```
PUT /bucket/key
Content-Type: application/json
Request Body: {...}
```

- Get documents from buckets ...

```
GET /bucket/key
Content-Type: application/json
Response Body: {...}
```

- Delete documents from buckets ...

```
DELETE /bucket/key
Content-Type: application/json
```

Range deletes

- Delete documents in bucket with keys in a given range ...

```
DELETE /bucket/range?comparator=comparator_name&
startKey=start_key&
endKey=end_key&
timeToLive=max_age
Content-Type: application/json
Response Body: [...]
```

Range queries

- Find documents in bucket with keys in a given range

```
GET /bucket/range?comparator=comparator_name&
startKey=start_key&
endKey=end_key&
timeToLive=max_age
Content-Type: application/json
Response Body: {...}
```

Merge updates

- Update documents in place by providing a merge descriptor ...

```
POST /bucket/key/merge
Content-Type: application/json
Request Body: {...}
Response Body: {...}
```

- The merge descriptor provides a syntax to describe update operations:

```
{"+" : {"newField" : "value"}
"-" : ["oldField"]}
```

Predicate queries

- Find documents in bucket satisfying a given predicate condition ...

```
GET /bucket/predicate?
predicate=type:expression
Content-Type: application/json
Response Body: {...}
```

Conditional put/get

- Conditionally put documents in buckets ...

```
PUT /bucket/key?
predicate=type:expression
Content-Type: application/json
Request Body: {...}
```

- Conditionally get documents from buckets ...

```
GET /bucket/key?
predicate=type:expression
Content-Type: application/json
Response Body: {...}
```

Map/Reduce

- Run Map/Reduce queries over a bucket ...

```
POST /bucket/mapReduce
Content-Type: application/json
Request Body: {...}
Response Body: {...}
```

- Queries are described by a simple JSON document:

```
{
"range" :
   {"startKey":\"k1\","endKey":\"k2\",
   "comparator":\"comparator\","timeToLive":10000},
"task":
   {"mapper\":"mapper",
   "combiner":"combiner",
   "reducer":"reducer",
   "timeout":10000}
}
```

Merge updates

- Update documents in place by providing a merge descriptor ...

```
POST /bucket/key/merge
Content-Type: application/json
Request Body: {...}
Response Body: {...}
```

- The merge descriptor provides a syntax to describe update operations:

```
{"+" : {"newField" : "value"}
"-" : ["oldField"]}
```

Update functions

- Atomically update documents via complex functions ...

```
POST /bucket/key/update?function=function_name&
timeout=timeout_value
Content-Type: application/json
Request Body: {...}
Response Body: {...}
```

What if ... custom comparators?

```
@AutoDetect(name="my-comparator")
public class MyComparator implements
   terrastore.store.operators.Comparator {

   public int compare(String key1, String key2) {
      // ...
   }
}
```

What if ... custom conditions?

```
@AutoDetect(name="my-condition")
public class MyCondition implements
    terrastore.store.operators.Condition {

    public boolean isSatisfied(String key,
        Map<String, Object> value, String expression) {
      // ...
    }
}
```

What if ... custom functions?

```
@AutoDetect(name="my-function")
public class MyFunction implements
    terrastore.store.operators.Function {

    public Map<String, Object> apply(String key,
        Map<String, Object> value,
        Map<String, Object> parameters) {
        // ...
    }
}
```

What if ... custom aggregators?

```
@AutoDetect(name="my-aggregator")
public class MyAggregator implements
   terrastore.store.operators.Aggregator {

   public Map<String, Object> apply(
       List<Map<String, Object>> values,
       Map<String, Object> parameters) {
       // ...
   }
}
```

More!

- **Backup**
    - Import/export documents to/from buckets.
- **Events management**
    - Get notified of document updates.
        - Third-party products integration.
        - Write-behind.
    - ActiveMQ integration for higher reliability.
- **Custom data partitioning**
    - Retain control of where your data is placed.
- **Indexing and searching**
    - Terrastore-Search.
        - ElasticSearch integration.
- **Cross-origin resource sharing support**
    - Integrate with browser-based clients.
- **...**

Final words ... engaging.

- **Explore**
  - http://code.google.com/p/terrastore
- **Download**
  - http://code.google.com/p/terrastore/downloads/list
- **Hack**
  - http://code.google.com/p/terrastore/source/checkout
- **Participate**
  - http://groups.google.com/group/terrastore-discussions
- **Enjoy!**

Q & A

Contact me on:
http://twitter.com/sbtourist