MongoDB

NoSQL なドキュメント指向データベース MongoDB (v2.x 〜) についてのメモです。
ゆるふわと言わたりしますが、サッと使うのには便利。まともに実用するには他と同様、相応の知見と工夫が必要な印象です。(良い意味で) 少し枯れ始めてきたかもしれません。
開発元は "10gen" でしたが、わりと最近、"MongoDB" という社名になりました。

特徴

- NoSQL
- スキーマレス＆ドキュメント型 (JSON(BSON))
  - なんといっても、スキーマレス。
  - フィールドをネスト可能
  - join は無い。RDB のような正規化はほとんど必要ない。(スキーマデザイン)
- index キー検索だけでなく、いかなるフィールドに対しても (ネストさえも) 検索が可能。like 検索、location(位置情報) 検索もサポート。
  - →但し、index を適切に張っておかないと全スキャンを行うことになる。
- CPU リソースはともかく (single-core サポート) だが、I/O、メモリ、HDD は贅沢に消費する。
  - →メモリに載る範囲であれば、とても高速。(載りきらなくなった場合は ... 場合は ...orz)
- GridFS による大容量ファイル格納をサポート
- Journaling をサポート
- JavaScript ベースのインタラクティブシェル
- 総じて、RDBMSっぽい機能が備わっており、MySQL などからも移行しやすい。(但し、RDBMS から見ると、相当割り切ったといえる仕様が多いため、理解が必要。)

RDB 用語との対応

MongoDB	RDB
コレクション	テーブル
行	ドキュメント

Query

- http://www.mongodb.org/pages/viewpage.action?pageId=6029355

Index

http://www.mongodb.org/pages/viewpage.action?pageId=5800049
Index を張らない場合、データ増加に伴う find の性能劣化が激しくなる。
なるべく Index を張っておくこと。また、複合 Index も生成可能。
但し、読み込みよりも書き込みのほうが多いコレクションには、Index を張らないほうがいい。

シェル : Index の生成

```
db.foo.ensureIndex({hoge:1});
```

シェル : クエリの実行計画

db.foo.find(bar).explain() で確認可能。

```
db.foo.find({ hoge:"piyo"}).explain();
```

http://www.slideshare.net/matsuou1/20110514-mongo-db

capped コレクション

固定されたサイズのコレクション。ロギングなどに最適。
http://www.mongodb.org/pages/viewpage.action?pageId=5079212
capped コレクションには、Index を張らないほうがいい。

Journal (ジャーナリング、先行書き込みログ)

v1.8 から導入された機能。v1.9.2 からデフォルトで有効となった。
オペレーションを行う前に、ジャーナルファイルにオペレーションログを先行書き込みする。

- メリット :
  - mongod の異常終了などからの復帰時に、整合性確認のための Repair が不要となり、迅速かつ安全に復帰できる。
- デメリット :
  - 3GB 程度のディスク領域が必要となる。
  - パフォーマンスにわずかながら影響する。

単体で開発に使っている場合などは、無効化しても構わないであろう。

```
$ mongod --nojournal
```

(異常発生した場合は、今までどおり repair するか、もしくは、不要ならばデータを消去すれば良い。)

参照 : MongoDB journal casebook
参照 : MongoDB の新機能：ジャーナリングについて詳しく

HTTP Admin UI

Web ブラウザから MongoDB の状態を確認できる。
mongod の起動時に --rest オプション付きで実行する必要がある。

```
mongod --port 27017 --rest
```

ポートは、mongod のポート + 1000 (即ちデフォルトでは 28017) となる。

また、yum などでインストールした場合は、
/etc/mongodb.conf に 以下を追加する

```
rest = true
```

(参照)

尚、SELinux により実行がブロックされる場合は

```
semanage port -a -t http_port_t -p tcp 28017
grep mongod /var/log/audit/audit.log | audit2allow -M mypol
semodule -i mypol.pp
```

のようにして許可する。

Mongos router

複数の mongod プロセスをまとめるためのルータ
http://www.mongodb.org/pages/viewpage.action?pageId=5800002
http://d.hatena.ne.jp/matsukaz/20110417/1303057728

Replica set (**冗長構成**)

Replica set は、Replica pairs よりさらに柔軟に構成が可能。

- master (primary)
- slave (secondary)
- [arbiter]

の 3 台以上で構成できる。(slave と arbiter を物理的な 1 台で兼用も一応可能)
arbiter は、障害発生時の master 昇格決定をサポートするのみの役割。
master/slave のみの 2 台構成となると正常に動作しない。(参照)

http://d.hatena.ne.jp/ryopeko/20101005/1286262922
http://sssslide.com/www.slideshare.net/naverjapan/mongodb-9422893
http://doryokujin.hatenablog.jp/entry/20101102/1288651712

SlaveOK

slave に対する参照を可能にする。(master に対しては insert/update/delete のみを行う運用)

Replica pairs (**冗長構成** (**旧**))

- master
- slave
- [arbiter]

の 2 台または 3 台で構成できる。
http://www.mongodb.org/display/DOCSJP/Replica+Pairs

arbiter は、障害発生時の master 昇格決定をサポートするのみの役割。
--arbiter オプションにより、arbiter なしでの運用も可能。(即ち、ネットワーク分断時には両方が master となる。)

Sharding (**負荷分散**)

http://sssslide.com/www.slideshare.net/doryokujin/mongodb-9208855

Tips

・like 検索 : http://d.hatena.ne.jp/hikaruright/20121022/1350874420

事例

・NAVER Photo Albumhttp://sssslide.com/www.slideshare.net/naverjapan/mongodb-9422893
