MongoDB

Dessì Massimiliano

CONFSL Jug Meeting Cagliari
12 giugno 2010

Author

Software Architect and Engineer
ProNetics / Sourcesense

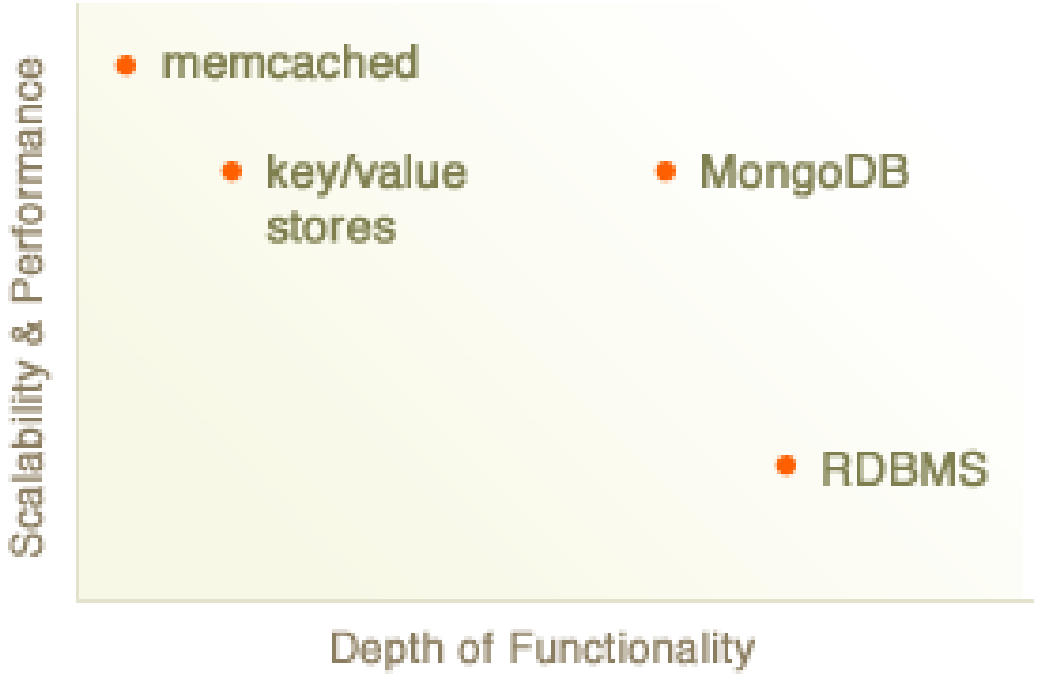
Presidente
JugSardegna Onlus

Fondatore e coordinatore
SpringFramework Italian User Group

Committer - Contributor
OpenNMS - MongoDB - MagicBox

Autore
Spring 2.5 Aspect Oriented Programming

Caratteristiche da 10000 piedi



Context

Graph databases

BigTable

MongoDB CouchDB

Document databases ThruDB

JackRabbit Voldemort

Distributed key-value stores

MapReduce

BigTable, Google 2006

http://labs.google.com/papers/bigtable.html

Le colonne non sono predefinite

Le righe possono esser aggiunte con qualsiasi numero di colonne

Le colonne vuote non vengono salvate

Adatto alle applicazioni dove gli attributi degli oggetti non sono conosciute o cambiano frequentemente

Graph database

Le relazioni tra gli item sono la caratteristica principale

Modello dei dati molto interconnesso

Adatto se nelle applicazioni avete delle lunghe catene di join, molte relazioni many to many e i dati sono già sotto una forma di grafo (social net..).
I graph database sono spesso associati al web semantico e ai datastore RDF.

Document database

Gli item sono dei Documenti
Non sono permesse joins e transactions spalmate su più righe o documenti.
Un documento è formato da valori o liste in formato JSON o XML e viene lavorato sulla struttura di un documento.
Estrazione, indicizzazione aggregazione e filtraggio sono basati sugli attributi contenuti nel documento

Adatto alle applicazioni dove i dati sono indipendenti tra loro e non sono richieste join.

MapReduce, Google 2004

http://labs.google.com/papers/mapreduce.html

Lavori batch di grandi moli di dati senza preoccuparsi della infrastruttura.

- Automatic parallelization and distribution
- Fault-tolerance
- I/O scheduling
- Status and monitoring

Adatto alle applicazioni dove si processando molti dati in processi batch.

Distributed key-value store

I database distribuiti partizionano e replicano i dati in maniera trasparente i dati su molte macchine in un cluster.
I dati possono non essere immediatamente consistenti.

La scelta è tra:

- Bassa latenza (velocità per req-res)
- Alto throughput (processi batch)

Adatti ad applicazioni dove i dati sono indipendenti e la disponibilità e performance dei dati sono più importanti delle caratteristiche ACID.

Document Database

Gli item sono dei Documenti

Non esistono join e transazioni su più documenti.

Un documento è formato da valori o liste in formato JSON o XML e viene lavorato sulla struttura di un documento.

Estrazione, indicizzazione aggregazione e filtraggio sono basati sugli attributi contenuti nel documento

Adatto alle applicazioni dove i dati sono indipendenti tra loro e non sono richieste join, i riferimenti ad altri documenti sono comunque possibili.

MongoDB

Built For Speed

Schema Free

Collection oriented storage: easy storage of object/JSON -style data

Dynamic queries and Indexes

Full index support, including on inner objects and embedded arrays

Query profiling

Replication and fail-over support

Efficient storage of binary data including large objects (e.g. photos and videos)

Auto-sharding for cloud-level scalability

Map/Reduce

Commercial Support, Hosting and Consulting Available

Schema Free

In un RDBMS la struttura dei dati è vincolata allo schema in cui definiamo tabelle con le colonne per meglio relazionare i dati.

In MongoDB lo “schema” viene definito dai dati quando vengono salvati dentro le collection

Raggruppiamo le entità della nostra applicazione in collections (Users, Centers...).

Contenitori

RDBMS

- Dati raggruppati in tabelle
- Schema Pre Definito
-Tabelle Pre definite
- Campi indicizzabili predefiniti
- Relazioni predefinite

MongoDB

- Dati raggruppati in collections
- Collection create al momento del primo salvataggio
- Schema definito dai tipi dei campi contenuti dai singoli documenti al momento del salvataggio
- Campi indicizzabili
- Relazioni non predefinite

Contenitori dati

RDBMS

- Dati salvati in righe
- Campi e tipi predefiniti
- Chiavi esterne predefinite
- Chiave primaria

MongoDB

- Dati salvati in Document
- Campi e tipi definiti al momento del salvataggio
- Sottodocumenti o reference non predefiniti
- _id field

Document JSON/BSON

Rappresentazione JSON salvataggio Binary-JSON


```
{ "_id" : "027b6e279578a64aa0684700" , "address-city" : "Ca" ,
"address-zipCode" : "09100" , "address-province" : "CA" ,
"address-region" : "Sardegna" , "address-country" : "Campidano" ,
"address-streetName" : "V.le Europe" , "address-streetNumber" : "4" ,
"telephone-contactMobile" : "3391234556" ,
"telephone-contactTelephoneHome" : "070123456" ,
"telephone-contactTelephoneWork" : "070987654" ,
"telephone-contactAcceptSms" : true ,
"userInfo-dateOfBirth" : "2009-09-08T15:30:30Z" ,
"userInfo-email" : "max@gmail.com" , "userInfo-name" : "Paperino" ,
"userInfo-surname" : "Paolino" , "userInfo-sensibleData" : "no sensible data" ,
"id" : "d37m3051128" , "description" : "descr" , "groupId" : "15" ,
"centerId" : "centerThree" , "_ns" : "centerUser"}
```


Adatto

Siti web

High volume, low data

Alta scalabilita

Dati in formato JSON

Caching

Logging

Analisi real-time

Non adatto

Sistemi altamente transazionali

Traditional Business Intelligence

Problemi che richiedono il SQL

Mongo Production Deployments

http://www.mongodb.org/display/DOCS/Production+Deployments

and many others

CERN Large Hadron Collider

http://blog.mongodb.org/post/660037122/holy-large-hadron-collider-batman

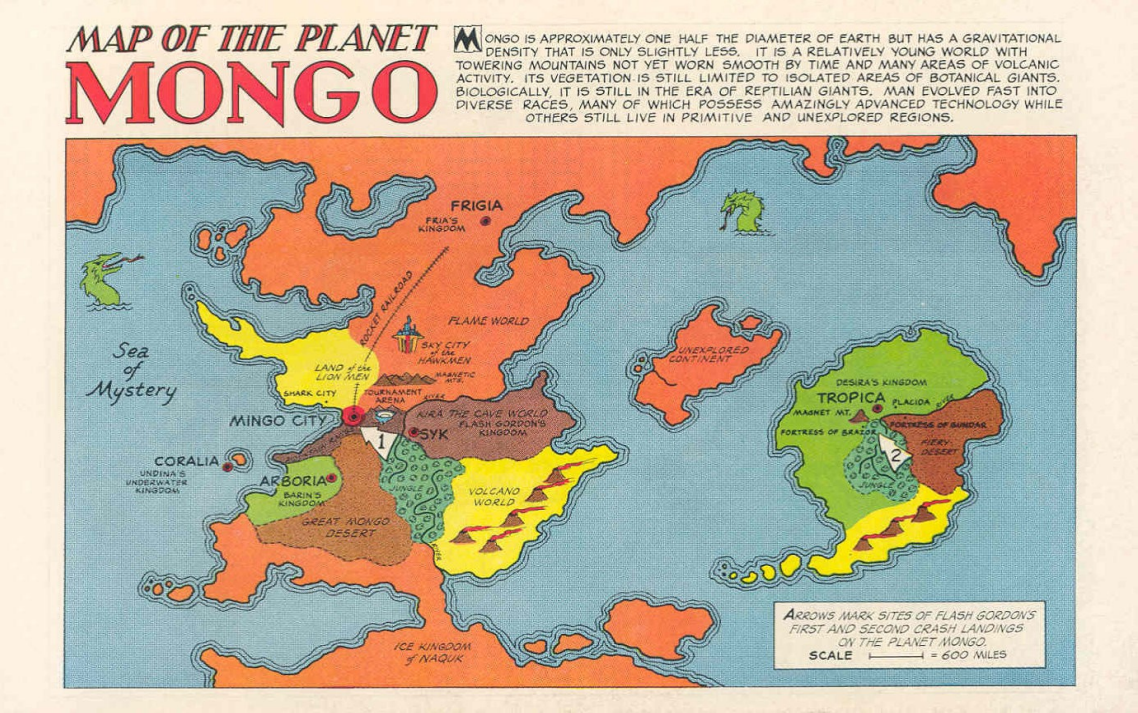
The Compact Muon Solenoid (CMS)

Data Aggregation System

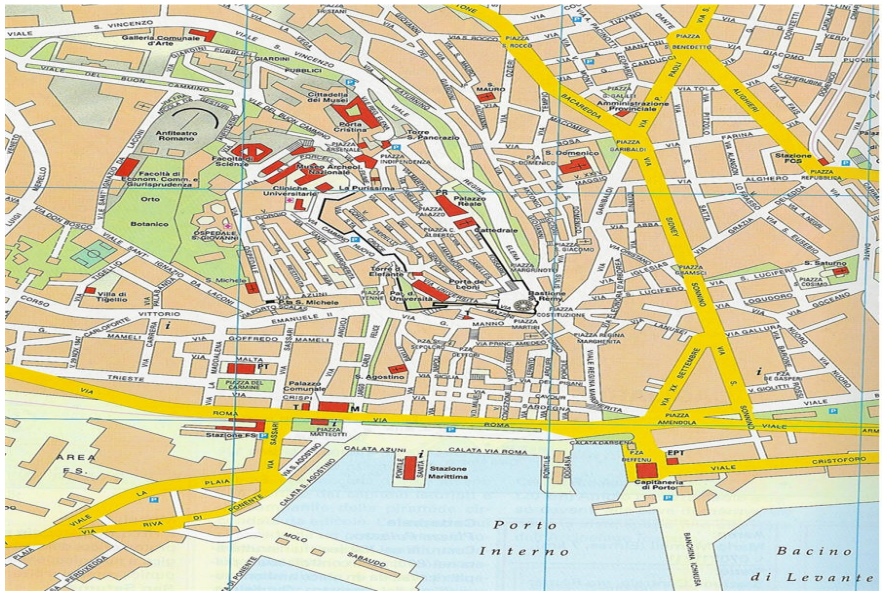
The aggregation system uses MongoDB as a cache.

It checks if Mongo has the aggregation the user is asking for and, if it does, returns it, otherwise the system does the aggregation and saves it to Mongo.

Geospatial Indexing



MongoDB supports two-dimensional geospatial indexes. It is designed with location-based queries in mind.



GRIDFS

GridFS is a specification for storing large files such video, photos, blob in MongoDB.

GridFS uses two collections to store data:

- files contains the object metadata
- chunks contains the binary chunks with some additional accounting information

MapReduce

```
db.runCommand(
 { mapreduce : <collection>,
   map : <mapfunction>,
   reduce : <reducefunction>
   [, query : <query filter object>]
   [, sort : <sort the query.  useful for optimization>]
   [, limit : <number of objects to return from collection>]
   [, out : <output-collection name>]
   [, keeptemp: <true|false>]
   [, finalize : <finalizefunction>]
   [, scope : <object where fields go into javascript global scope >]
   [, verbose : true]
 }
);
```

MapReduce

Collection “posts”, ciascun documento formato in questo modo

```
{
    "title" : "Sun", "author" : "max", "content" : "....",
    "tags" : ["MongoDB", "Java", "day", "MapReduce"]
}
```

```
map = function() {
   if (!this.tags) {
      return;
   }
   for (index in this.tags) {
       emit(this.tags[index], 1);
   }
}

reduce = function(previous, current) {
   var count = 0;

   for (index in current) {
       count += current[index];
   }
   return count;
}
```

MapReduce

```
result = db.runCommand({
  ... "mapreduce" : "posts",
  ... "map" : map,
  ... "reduce" : reduce,
  ... "out" : "tags"})
```

```
> db.tags.find()
  {"_id" : "MongoDB", "value" : 3}
  {"_id" : "Map/Reduce", "value" : 1}
  {"_id" : "Java", "value" : 7}
  {"_id" : "day", "value" : 1}
  {"_id" : "Sun", "value" : 2}
```

MongoDB Tools

- mongo – Shell javascript interattiva
- mongod - Server
- mongodump – Dump database
- mongoexport – Export collection in JSON, CSV
- mongofiles – Import GridFS DOCS
- mongoimport – Importa collection da JSON/CSV/TSV
- mongostat – Mostra statistiche in tempo reale
- mongorestore – Restore mongodump output
- mongos – Auto sharding
- mongosniff -Sniffer

Mongo Driver

Mongo può essere utilizzato con i linguaggi più diffusi:

Java (Groovy, Scala, JRuby), PHP, Ruby, C++, Python,
Perl, C#, Erlang, Javascript server side.
Ogni driver predispone i nostri oggetti ad
essere utilizzati in Mongo
ed utilizza le “magic words” attese da mongo
per compiere le operazioni

Atomicità

Il motivo principale della mancanza delle transazioni è la lentezza e il costo dei lock in ambienti distribuiti.

Non essendoci le transazioni che coinvolgono più dati, qui abbiamo atomicità a livello di singolo documento con i seguenti comandi:

```
$set  $inc $push $pushAll $pull $pullAll
```

Collection

I nostri item (Document) sono salvati dentro delle Collection che possiamo vedere come i corrispettivi delle tabelle in un RDBMS.

Una collection viene creata effettivamente quando un document viene salvato al suo interno.

```
public void createCollectionCenters(Mongo mongo) {
     NoCenter center = new NoCenter();
     DBCollection collectionCenters = mongo.getCollection("centers");
     collectionCenters.insert(new BasicDBObject(center.toMap()));
 }
```

Mongo Java Driver

Per essere salvati o letti i nostri oggetti devono essere “traslati” in oggetti `com.mongodb.DBObject`, per fare questo ci sono due alternative:

Implementare l' interfaccia `DBObject`

Utilizzare un `BasicDBObjectBuilder`

Mongo Java Driver

Se la nostra `MyClass` implementa l' interfaccia `DBObject` assegneremo i valori in questo modo.

```
MyClass myObj= new MyClass();
myObj.put("user", userId);
myObj.put("message", msg);
myObj.put("date", new Date());
...
collection.insert(myObj);
```

Mongo Java Driver

Utilizzando BasicDBObjectBuilder

```
collection.insert(
    BasicDBObjectBuilder.start()
    .add("user", myObj.getId())
    .add("user", myObj.getUser())
    .add("date", new Date())
    .....
    .get();
);
```

Mongo Java Driver

Una terza alternativa, più pratica consiste nel passare al Builder la mappa con i valori contenuti nel nostro oggetto.

```
BasicDBObjectBuilder.start(myObj.toMap()).get();
```

Dobbiamo semplicemente aggiungere agli oggetti della nostra applicazione una rappresentazione sotto forma di Map e un costruttore con un Map per ricostruirli da un `DBObject`

```
public class MyClass{

    public MyClass(Map map){...}

    public Map toMap(){..}
}
```

Interrogazioni

Le query di ricerca vengono fatte costruendo i parametri di ricerca tramite un `DBObject`.

Possiamo anche paginare il risultato della query.

```
List<DBObject> objects =
   collection.find(
      BasicDBObjectBuilder.start().
         add("userInfo-surname", surname).
         add("centerId", centerId).get()
   ).
   skip(offset * page).limit(offset).toArray();
```

Interrogazioni

Numero documenti in una collezione

```
collection.find(
    BasicDBObjectBuilder.start()
    .add("centerId",centerId).get()
).length();
```

Interrogazioni

Se vogliamo recuperare solo alcuni campi del documento (Select SQL), costruiamo un `DBObject` con quei campi e utilizziamo il find che accetta due DBObject (query, campi richiesti).

Il `DBCursor` è il corrispondente del ResultSet jdbc.

```
DBCursor cursor = coll.find(
    BasicDBObjectBuilder.start().add("centerId", centerId).get(),
    BasicDBObjectBuilder.start().add("userInfo-email", "").get()
);
```

Interrogazioni

Con espressioni regolari

```
String pattern = new StringBuilder().append(character).append("*").toString();
List<DBObject> objects = coll.find(BasicDBObjectBuilder.start()
                    .add("userInfo-surname",
                          java.util.regex.Pattern.compile(pattern))
                    .get()).toArray();
```

Interrogazioni Avanzate

Nelle query abbiamo a disposizione anche :

`<, <=, >, >=, $ne, $in, $nin,`

`$mod, $all, $size, $exists`

Valori dentro Array

Valori dentro un oggetto embedded

Full language expressions con `$where`

`limit(), skip(), snapshot(), count(), group()`

Delete

Delete utilizzando il DBcursor

```
public boolean deleteCenterUser(String id, String centerId) {
        DBCursor cursor = coll.find(
                          new BasicDBObject(),
                           BasicDBObjectBuilder.start()
                           .add("id", id)
                           .add("centerId", centerId).get()
                          );
        boolean result = false;
        while (cursor.hasNext()) {
            coll.remove(cursor.next());
            result = true;
        }
        return result;
}
```

Insert

```
public String insertCenterUser(CenterUser user) {
    Map fields = user.toMap();
    fields.put("id", IdGenerator.getUniqueId());//CustomId
    DBObject obj = coll.insert(BasicDBObjectBuilder.start(fields).get());
    return obj != null ? obj.get("id").toString() : Constants.ID_NEW;
}
```

Update

```
public int updateCenterUser(CenterUser user) {

       DBObject  obj = coll.update(
                   BasicDBObjectBuilder.start().
                      add("id",user.getEntity().getId()).get(),
                   BasicDBObjectBuilder.start(user.toMap()).get(),
                   false,
                   false
                );
        return obj != null ? 1 : 0;
}
```

Upsert

Update if present insert is missing

```
public int updateCenterUser(CenterUser user) {

       DBObject obj = coll.update(
                     BasicDBObjectBuilder.start().
                        add("id",user.getEntity().getId()).get(),
                     BasicDBObjectBuilder.start(user.toMap()).get(),
                     true,
                     false
                  );
        return obj != null ? 1 : 0;
}
```

Delete

Delete diretto

```
public void deleteCenterUser(String id, String centerId) {
    DBObject obj = coll.findOne(
          BasicDBObjectBuilder.start().
          add("id", id).
          add("centerId", centerId).get());
    coll.remove(obj);
}
```

Q & A

p.s.

p.p.s. It's a Joke :-)

Grazie per l'attenzione !

Massimiliano Dessì
desmax74 at yahoo.it
massimiliano.dessi at pronetics.it

http://twitter.com/desmax74

http://jroller.com/desmax

http://www.linkedin.com/in/desmax74

http://www.slideshare.net/desmax74

http://wiki.java.net/bin/view/People/MassimilianoDessi

http://www.jugsardegna.org/vqwiki/jsp/Wiki?MassimilianoDessi