Проектирование и эксплуатация информационных систем в медиаиндустрии

Выломова Екатерина Алексеевна

e-mail: evylomova@gmail.com

0. Предыдущие лекции

- Открытые системы
- Веб-сервисы
- SOAP, RPC, WSDL, REST
- SOA, WEB 2.0
- Бизнес-процессы
- Примеры

Повторение

0. Принцип Парето

20% усилий дают

80% результата

I. Лекция 9. MapReduce & Hadoop

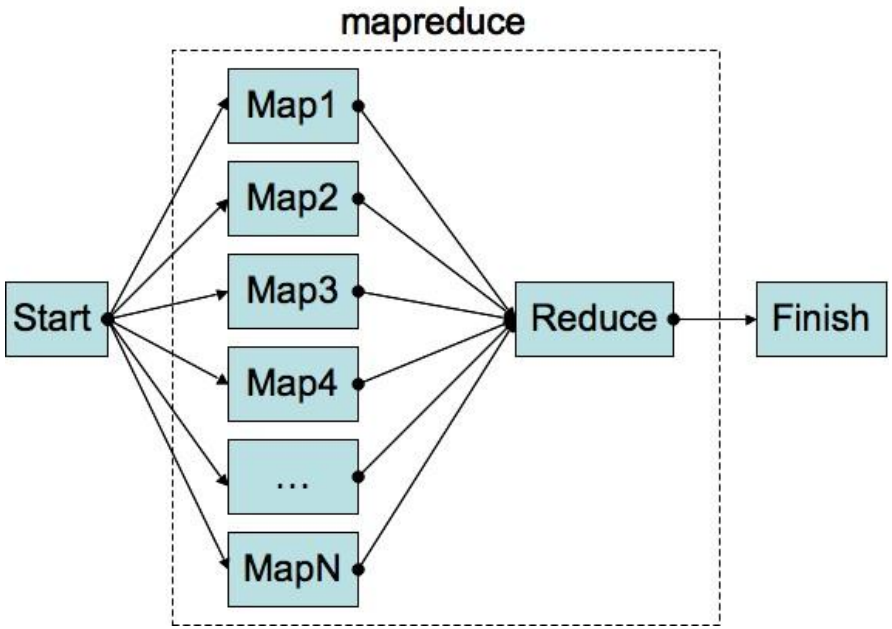
- MapReduce
- Hadoop
- MongoDB
- CouchDB

Основные понятия

I. MapReduce

MapReduce— программный фреймворк, представленный компанией Google, используемый для параллельных вычислений над очень большими, несколько петабайт,наборами данных в компьютерных кластерах..

Фреймворк для вычисления некоторых наборов распределенных задач с использованием большого количества компьютеров (называемых «нодами»), образующих кластер.



MapReduce

I. MapReduce

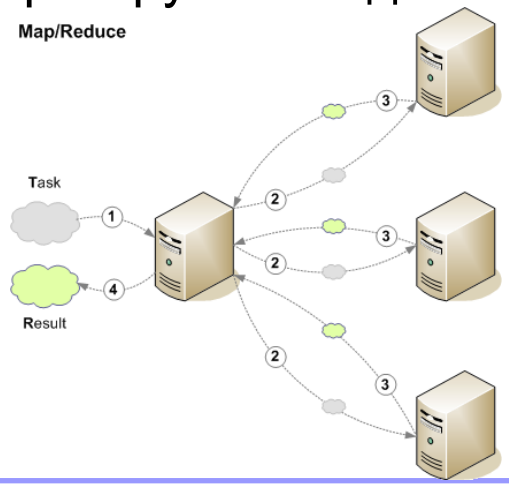
MapReduce разделяется на два этапа:

- ***Map:*** предварительная обработка данных.

Мастер-нода получает входные данные, разделяет их и передает другим для обработки

- ***Reduce:*** свертка предварительно обработанных данных.

Главный узел получает ответы от других, агрегирует и выдает результат



I. MapReduce. Пример

```
// Функция, используемая рабочими нодами на Map-шаге
// для обработки пар ключ-значение из входного потока
void map(String name, String document):
        // Входные данные:
        // name - название документа
        // document - содержимое документа
        for each word w in document:
                EmitIntermediate(w, "1");
```

I. MapReduce. Пример

```
// Функция, используемая рабочими нодами на Reduce-шаге
// для обрабоки пар ключ-значение, полученных на Map-шаге
reduce(String word, Iterator partialCounts):
        // Входные данные:
        // word - слово
        // partialCounts - список группированных промежуточных результатов.
        // Количество записей в partialCounts и есть
        // требуемое значение
        int result = 0;
        for each v in partialCounts:
                result += parseInt(v);
        Emit(AsString(result));
```

I. MapReduce. Пример

ВХОД: строка «foo bar baz bar»

НУЖНО НА ВЫХОДЕ: { foo: 1, bar: 2, baz: 1 }

1. Разбиение строки:
   ['foo', 1]
   ['bar', 1]
   ['baz', 1]
   ['bar', 1]
2. Сортировка:
   bar, 1
   bar, 1
   baz, 1
   foo, 1
3. Объединяем:
   bar, (1,1)
   baz, (1)
   foo, (1)
4. Складываем:
   bar, 2
   baz, 1
   foo, 1

I. MapReduce. Пример

На Python:

```
words = ["foo", "bar", "baz"]
def map1(word):
        return [word, 1]

arr = ["foo", [1,1]]
def reduce1(arr):
        return [ arr[0], sum(arr[1]) ]
```

I. MapReduce. Пример

Сортировка по частоте:

1. По возрастанию=> умножаем на -1

[слово, 15] -> map() возвращает -> [-15, слово]

[слово2, 15] -> map() возвращает -> [-15, слово2]

[слово3, 120] -> map() возвращает -> [-120, слово3]

[слово4, 1] -> map() возвращает -> [-1, слово4]

2. Группировка

-120, (слово3)

-15, (слово, слово2) <-- два слова на строке - сгруппировал все по первому ключу!

-1, (слово4)

3. Умножение на -1 и разбиение

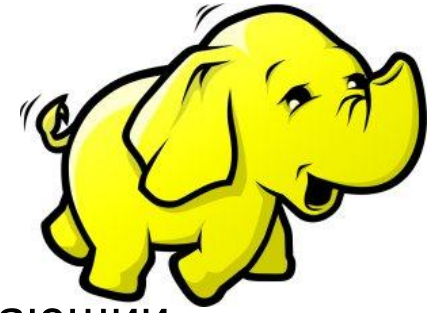
120, слово3

15, слово,

15, слово2

1, слово4

II. Hadoop

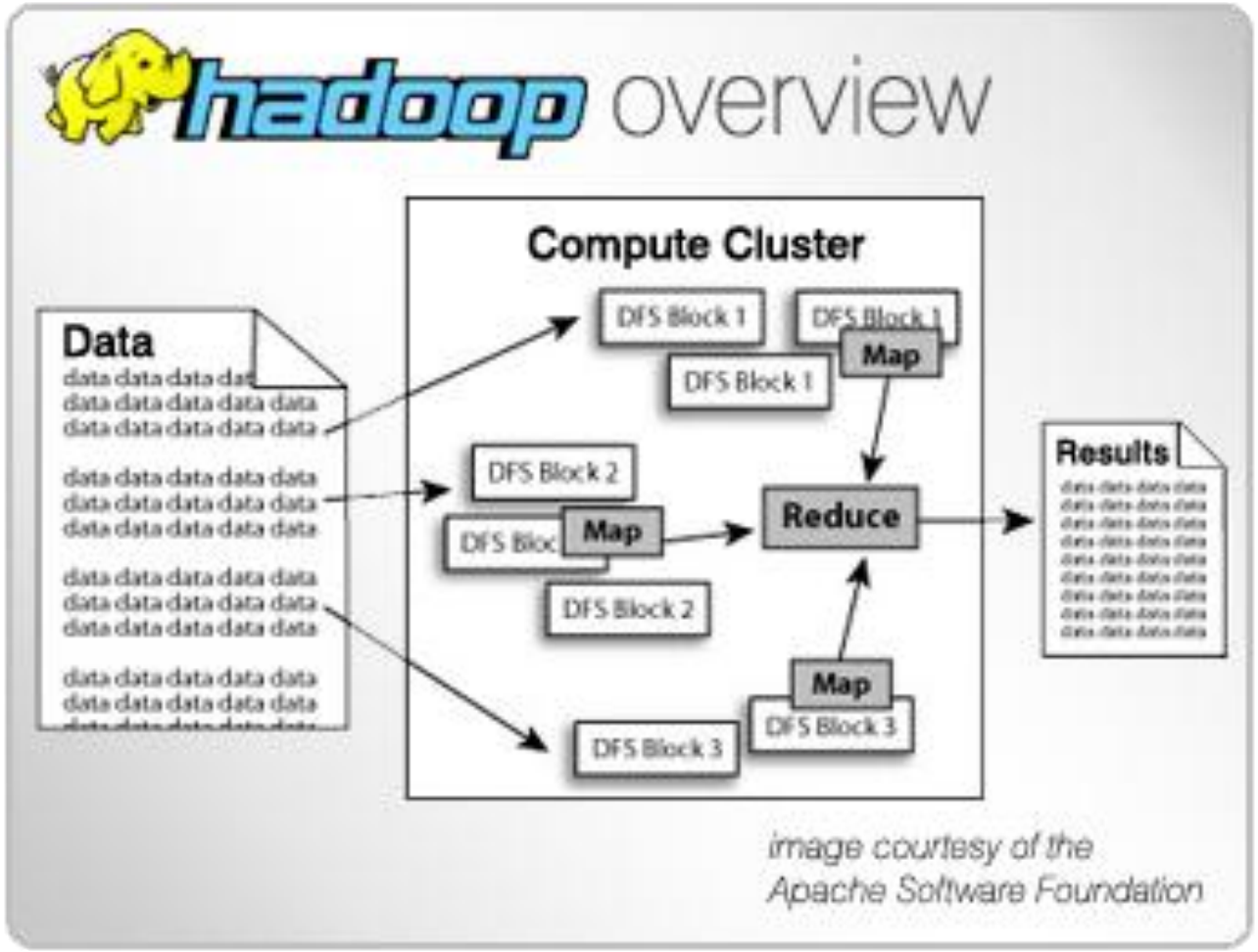


Hadoop - свободный Java-фреймворк, поддерживающий выполнение распределённых приложений, работающих на больших кластерах, построенных на обычном оборудовании.

Поддерживает парадигму MapReduce.

Приложение разделяется на большое количество небольших заданий, каждое из которых может быть выполнено на любом из узлов кластера.

II. Hadoop



Hadoop

III. MongoDB

MongoDB - документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц.

Основные возможности данной СУБД:

- Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)
- Достаточно гибкий язык для формирования запросов
- Динамические запросы
- Полная поддержка индексов
- Профилирование запросов
- Быстрые обновления «на месте»
- Эффективное хранение двоичных данных больших объёмов, напр., фото и видео
- Журналирование операций, модифицирующих данные в БД
- Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг
- Может работать в соответствии с парадигмой MapReduce

III. MongoDB

MongoDB - документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц.

Основные возможности данной СУБД:

- Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)
- Достаточно гибкий язык для формирования запросов
- Динамические запросы
- Полная поддержка индексов
- Профилирование запросов
- Быстрые обновления «на месте»
- Эффективное хранение двоичных данных больших объёмов, напр., фото и видео
- Журналирование операций, модифицирующих данные в БД
- Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг
- Может работать в соответствии с парадигмой MapReduce

III. MongoDB. Примеры

- j = { name : "mongo" };

{"name" : "mongo"}

- t = { x : 3 };

{ "x" : 3 }

- db.things.save(j);
- db.things.save(t);
- db.things.find();

{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
{ "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }

III. MongoDB. Примеры

- for (var i = 1; i <= 20; i++)
- db.things.save({x : 4, j : i});
- db.things.find();

```
{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }
{ "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }
{ "_id" : ObjectId("4c220a42f3924d31102bd856"), "x" : 4, "j" : 1 }
{ "_id" : ObjectId("4c220a42f3924d31102bd857"), "x" : 4, "j" : 2 }
{ "_id" : ObjectId("4c220a42f3924d31102bd858"), "x" : 4, "j" : 3 }
{ "_id" : ObjectId("4c220a42f3924d31102bd859"), "x" : 4, "j" : 4 }
.....
```

III. MongoDB. Примеры

ПОИСК: **SELECT * FROM things WHERE name=«mongo»**

db.things.find({name:"mongo"}).forEach(printjson);

{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }

Или **FindOne**

printjson(db.things.findOne({name:"mongo"}));

{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }

Соответствие SQL :
http://ru.wiki.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart

IV. CouchDB

CouchDB — документо-ориентированная система управления базами данных, не требующая описания схемы данных. Эта программа является свободной, открытой, и написана на языке Erlang.

- данные сохраняются не в виде JSON-подобных документов, моделью которых является не таблицы, а деревья;
- типизация элементов не поддерживается — вместо этого пользователь может написать функцию-валидатор;
- целостность базы данных обеспечивается исключительно на уровне отдельных записей
- связи между таблицами или записями принципиально не поддерживаются;
- функции-валидаторы, функции-представления, функции-фильтры сохраняются в текстовом виде в самой базе данных;
- каждой базе данных в системе CouchDB соответствует единственное B-дерево (не путать с двоичным деревом);
- каждое B-дерево хранится в виде отдельного файла на диске;

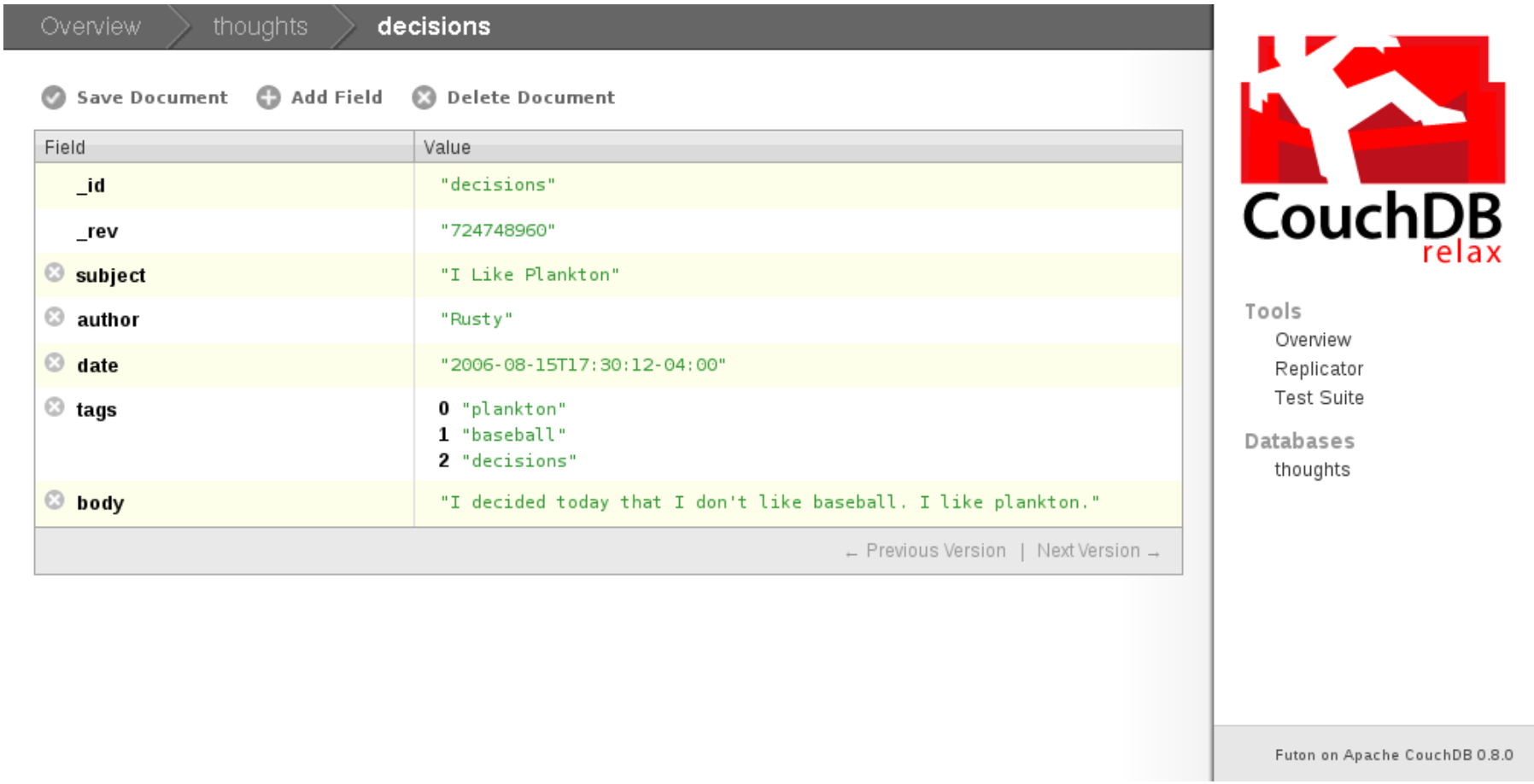
IV. CouchDB. Пример


```
import couchdb
couch = couchdb.Server() # Assuming localhost:5984
# If your CouchDB server is running elsewhere, set it up like this:
couch = couchdb.Server('http://example.com:5984/')
# select database db = couch['mydb']
#create a document and insert it into the db:
doc = {'foo': 'bar'}
db.save(doc)
```

IV. CouchDB. Пример


```
doc = """
    {
        "value":
       {
           "Subject":"I like Planktion",
           "Author":"Rusty",
           "PostedDate":"2006-08-15T17:30:12-04:00",
           "Tags":["plankton", "baseball", "decisions"],
            "Body":"I decided today that I don't like baseball. I like
plankton."
       }
   }
  """
```

IV. CouchDB. Пример



CouchDB