MapReduce

Databases Three

Otago Polytechnic
Dunedin, New Zealand

Problem: counting splatts

Suppose we want to count the number of splatts in our collection.

- Relational : `select count(*) from splatts;`
- Document : If we had a splatts collection, we could do this - `db.splatts.count()` - but we don't have one.
- We need to tell MongoDB how to find and count our splatts.

The data

```
 {
   "_id" : ObjectId("541671756269625 9b8010000"),
   "email" : "imp@casterlyrock.com",
   "follower_ids" : [ ObjectId("5416717562696259b8000000") ],
   "name" : "Tyrion Lannister",
   "splatts" : [
      {
       "_id" : ObjectId("5416727962696259b8030000"),
       "body" : "user 2 splatt 2",
       "created_at" : ISODate("2014-09-15T05:00:41.376Z")
      }
   ]
}
```

The plan

1. Iterate over the users[1];
2. Count the splatts belonging to each one;
3. Add up the counts.

This is a common pattern, and it has a name: MapReduce.

[1]Our data set is small, but this could be parallelised over a large data set.

MapReduce

1. Map: For each user, count the splatts.
2. Reduce: Sum up the indvidual counts.

MapReduce implementation

We need to supply JavaScript code to MongoDB telling it how to perform the Map and Reduce steps

- We write a map function that is called one time on each user in the collection. It will emit the count of each user's splatts.
- We write a reduce function that sums up the output from the map function.
- We supply both of these functions to MongoDB's `mapReduce` command.

Map

```
var map = function() {
    var length = 0;
    if(this.splatts) {
        length = this.splatts.length
    }
    emit ("count", length);
};
```

Map results

After applying the map to a collection with 3 users, the result set may look like this:

```
results = [
    {key: "count", value: 4 },
    {key: "count"  value: 3 },
    {key: "count", value: 3 }
    ]
```

Reduce

```
var reduce =  function(key, val) {
    var data =  0;
    val.forEach(function(v) {
        data += v;
    });
    return data;
}
```

This reduce function will be called one time for each key value in the map results.

MapReduce

We use our map and reduce functions in a call to MongoDB's mapReduce command.

```
db.users.mapReduce(
  map,
  reduce,
  {
    out: {inline: 1}
  }
}
```

The final result

```
{
  "results" : [
    {
      "_id" : "count",
      "value" : 10
    }
  ],
  "timeMillis" : 0,
  "counts" : {
    "input" : 3,
    "emit" : 3,
    "reduce" : 1,
    "output" : 1
  },
  "ok" : 1,
}
```

In Ruby

The Mongoid library provides access to MongoDB's MapReduce functionality:

```
 map = %Q{ function() {
      var length = 0;
      if(this.splatts) {
          length = this.splatts.length
      }
      emit ("count", length);
  }
}
 reduce =  %Q{ function(key, val) {
    var data =  0;
    val.forEach(function(v) {
        data += v;
    })
    return data;
 }
}
```

In Ruby

```
User.map_reduce(map,reduce).out(inline: true)
```

Today's lab

1. Adapt the test scripts you wrote earlier this semester to enter some sample data into your MongoDB database.
2. Use MapReduce from within the MongoDB shell to get a count of the number of splatts.
3. Add a function to your Rails UsersController to return the total number of splatts.