Lecture 15A (23 April)

Scaling

So far we have just worked on a single machine. But there is a lot more!

We have been using the **n-tier** model:

- Front-end
- Application layer
- Storage layer

You can add additional business logic, etc. layers.

Can scale:

- Add more servers - SOA
- Replication - primary/replica (master/slave); one writer, many readers
- Multi-master replication
- Distributed storage - not everything is on the same place!
    - Distributed hash tables
    - Sharding

Trade-offs

- Multi-master and distributed applications have reliability tradeoffs
- **Consistency** (not in the ACID sense)
- **Availability**
- **Partition tolerance**
- CAP ‘theorem’: choose 2

Models

- Eventual consistency
- Majority systems

Many relational database systems support primary/replica; some support multi-master (Postgres recently added support).

Beyond SQL

- We've focused on relational models

- Remaining models are largely *schema-free*

Key-Value Stores

- Dictionaries, but on disk.
- Long history: dbm, gdbm, ndbm, Berkley DB
- Modern embedded: LevelDB, Kyoto Cabinet
- Basis for many distributed storage architectures (memcached, Cassandra)
    - map well to *distributed hash tables*
- using the Unix file system

Key-Value ++

- Redis: rich values
- Table stores (hierarchical key-value stores, cell stores)

Triple Stores, Graph Stores

- RDF triple stores
- Neo4J
- SPARQL

Document Stores

- Shove JSON documents in the database
    - CouchDB
    - MongoDB
- Historically: shove XML in the database, query with XPath/XQuery

Example: MongoDB

Insert some documents.

Review some documents.

Map-Reduce to summarize categories:

```
db.books.mapReduce(
    function() {
```

```
        this.subjects.forEach(function(s) {
            emit(s, 1)
        })
    },
    function(key, values) {
        return Array.sum(values)
    },
    { query: { subjects: {$exists: true}},
      out: 'author_books' }
)
```

And then get the results:

```
db.author_books.find().sort({value: -1})
```

Processing Models

- We've seen SQL queries
- Similar model with other query languages
    - SPARQL (RDF stores)
    - XQuery (XML stores)
- ‘query documents’ (Mongo)

Map/Reduce

- Distributed/parallel computing architecture
- Write code (JavaScript, Erlang, Java, etc.)
- All computation in terms of 2 functions:
    - map: takes object, emits key-value pairs about it
    - reduce: takes key & associated values, computes summarized value

Example: counting tags

Map : Emit tags, with value '1'

Reduce : Sum tag values

Used as large-scale parallel computing infrastructure, developed by Google.

Basis for query computations in some NoSQL databases (MongoDB, CouchDB)

Only indexing mechanism for Couch.