MongoDB and Hadoop

Siyuan Zhou

Software Engineer, MongoDB

Agenda

- Complementary Approaches to Data
- MongoDB & Hadoop Use Cases
- MongoDB Connector Overview and Features
- Examples

Complementary Approaches to Data

Operational: MongoDB

Real-Time Analytics	Product/Asset Catalogs	Churn Analysis	Recommender
Security & Fraud	Internet of Things	Warehouse & ETL	Risk Modeling
Mobile Apps	Customer Data Mgmt	Trade Surveillance	Predictive Analytics
Single View	Social	Ad Targeting	Sentiment Analysis

MongoDB

- Fast storage and retrieval
- Easy administration
- Built-in analytical tools
  - Aggregation framework
  - JavaScript MapReduce
  - Geo/text indexes

Analytical: Hadoop

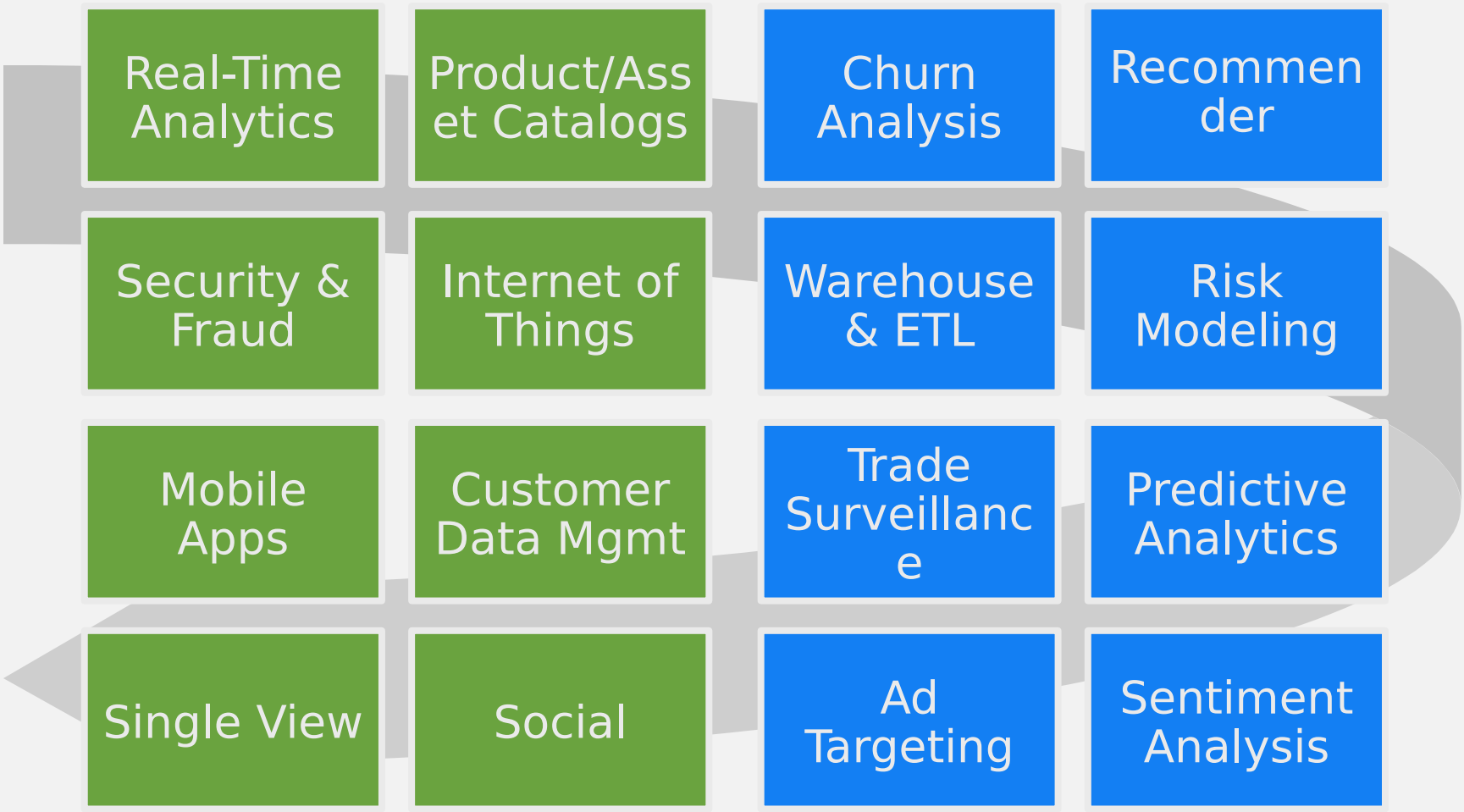
Real-Time Analytics	Product/Asset Catalogs	Churn Analysis	Recommender
Security & Fraud	Internet of Things	Warehouse & ETL	Risk Modeling
Mobile Apps	Customer Data Mgmt	Trade Surveillance	Predictive Analytics
Single View	Social	Ad Targeting	Sentiment Analysis

Hadoop

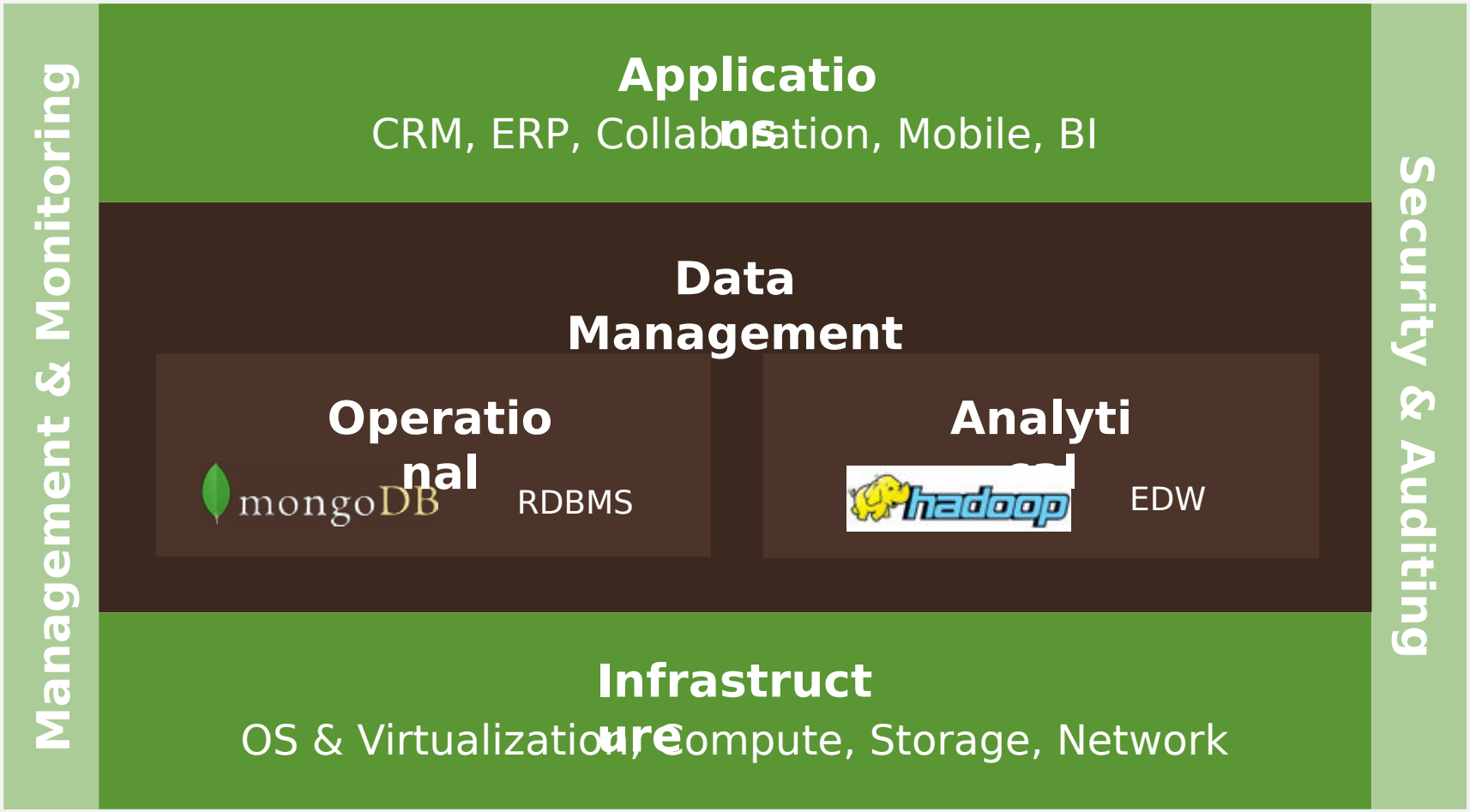
The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

- Terabyte and Petabyte datasets
- Data warehousing
- Advanced analytics
- Ecosystem

Operational vs. Analytical: Lifecycle



Enterprise IT Stack



MongoDB & Hadoop Use Cases

Commerce

Applications powered by mongoDB

Hadoop Connector

Analysis powered by hadoop

- Products & Inventory
- Recommended products
- Customer profile
- Session management

- Elastic pricing
- Recommendation models
- Predictive analytics
- Clickstream history

Insurance

Applications powered by mongoDB

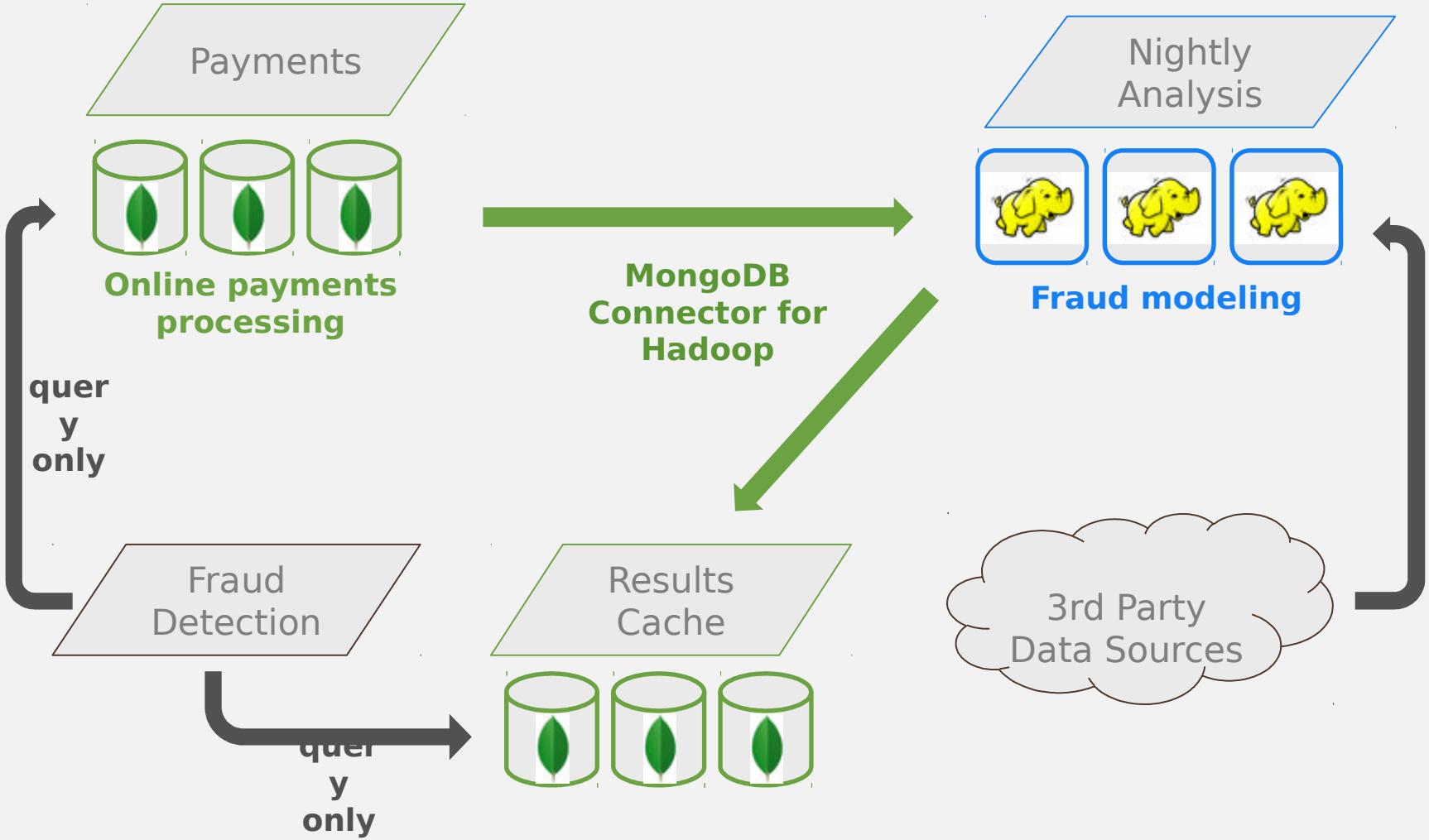
Hadoop Connector

Analysis powered by hadoop

- Customer profiles
- Insurance policies
- Session data
- Call center data

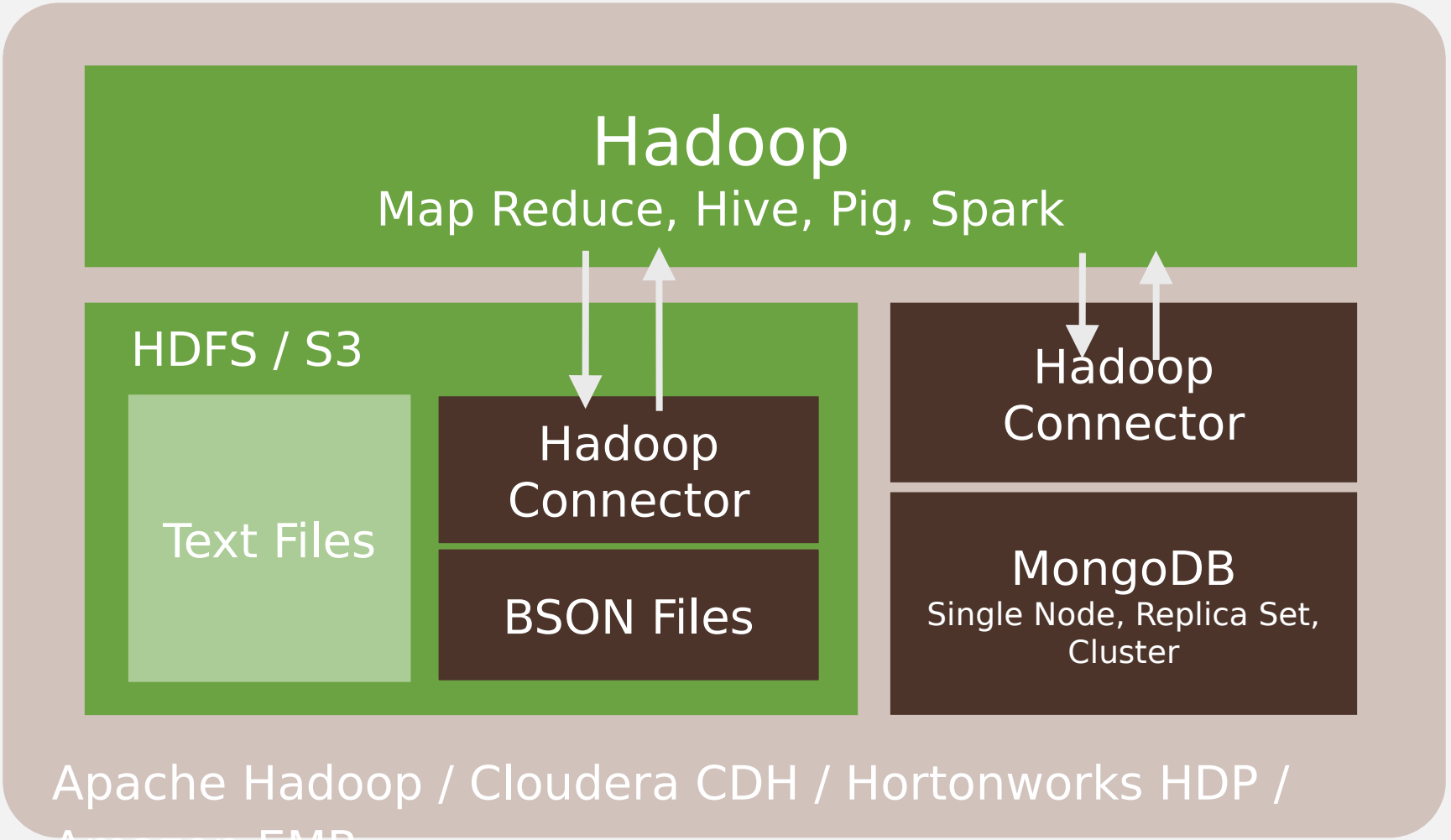
- Customer action analysis
- Churn analysis
- Churn prediction
- Policy rates

Fraud Detection



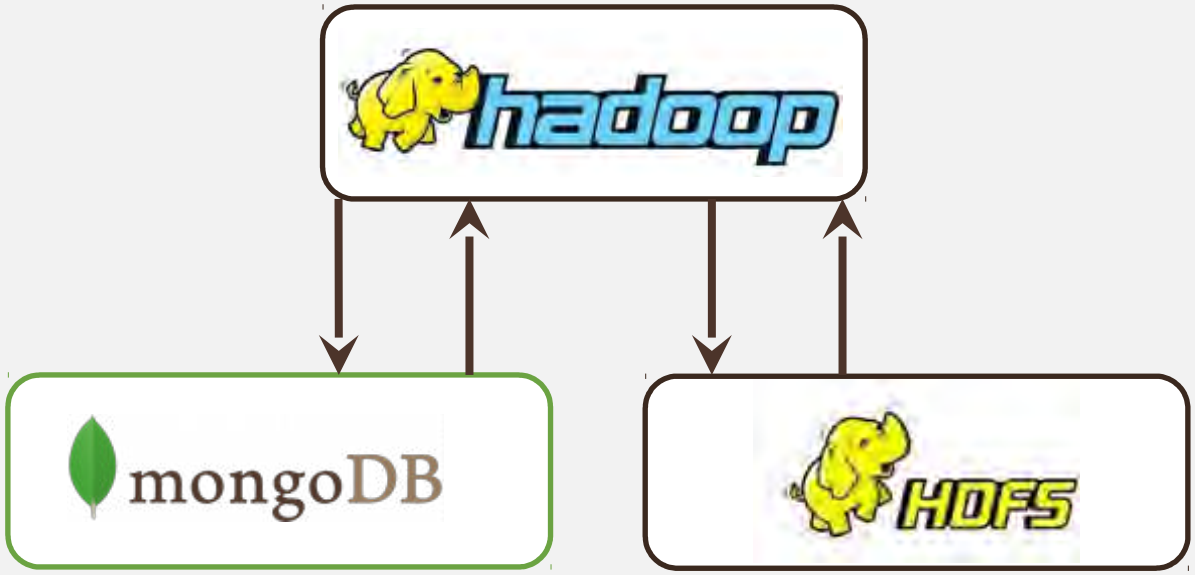
MongoDB Connector for Hadoop

Connector Overview



Data Movement

Dynamic queries to MongoDB vs. BSON snapshots in HDFS



- Dynamic queries with latest data
- Puts load on operational database

- Snapshots move load to Hadoop
- Add predictable load to MongoDB

Connector Features and Functionality

- Computes splits to read data
  - Single Node, Replica Sets, Sharded Clusters
- Mappings for Pig and Hive
  - MongoDB as a standard data source/destination
- Support for
  - Filtering data with MongoDB queries
  - Authentication
  - Reading from Replica Set tags
  - Appending to existing collections

Data Split

- Standalone and replica set
  - Split data into chunks
- Cluster
  - Unsharded, same as standalone
  - Sharded, split per chunk
  - Sharded, split per shard
- BSON Files
  - *.split* file stores metadata

MapReduce Configuration

- MongoDB input
  - mongo.job.input.format = com.hadoop.MongoInputFormat
  - mongo.input.uri = mongodb://mydb:27017/db1.collection1
- MongoDB output
  - mongo.job.output.format = com.hadoop.MongoOutputFormat
  - mongo.output.uri = mongodb://mydb:27017/db1.collection2
- BSON input/output
  - mongo.job.input.format = com.hadoop.BSONFileInputFormat
  - mapred.input.dir = hdfs:///tmp/database.bson
  - mongo.job.output.format = com.hadoop.BSONFileOutputFormat
  - mapred.output.dir = hdfs:///tmp/output.bson

Pig Mappings

- Input: BSONLoader and MongoLoader

```
data = LOAD 'mongodb://mydb:27017/db.collection'
    using com.mongodb.hadoop.pig.MongoLoader
```

- Output: BSONStorage and MongoInsertStorage

```
STORE records INTO 'hdfs:///output.bson'
    using com.mongodb.hadoop.pig.BSONStorage
```

Pig Mappings

- Primitive type (Integer, String) => Primitive type in Pig
- Document => Tuple defined by the schema
- Document => Tuple of only one Map element (dynamic schema mode)
- Subdocument => Map
- Array => Tuple or Bag

Hive Support

```
CREATE TABLE mongo_users (id int, name string, age int)
STORED BY
"com.mongodb.hadoop.hive.MongoStorageHandler"
WITH SERDEPROPERTIES("mongo.columns.mapping" =
"_id,name,age")TBLPROPERTIES("mongo.uri" =
"mongodb://host:27017/test.users")
```

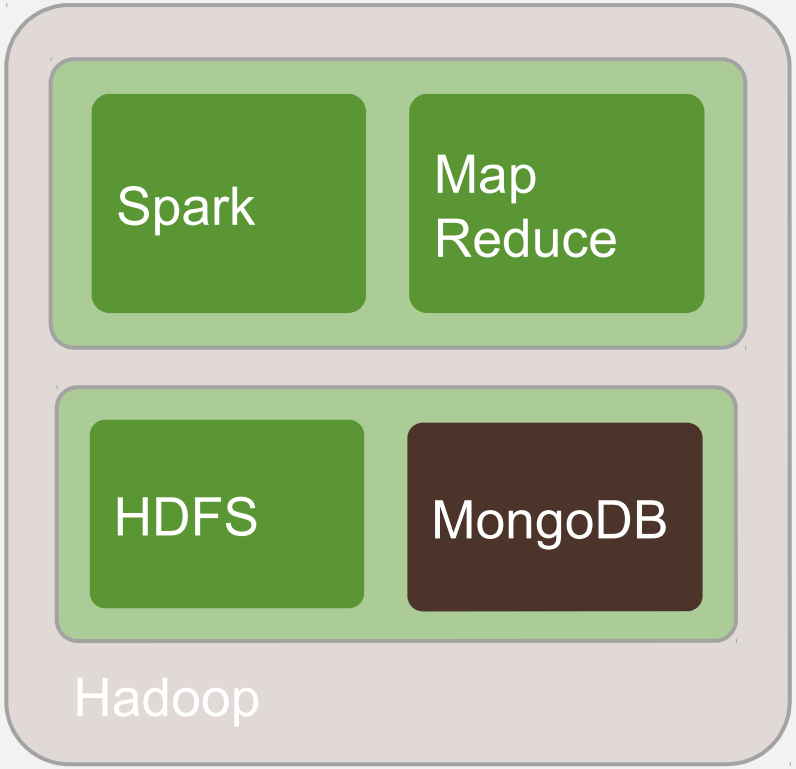
- Access collections as Hive tables
- Use with MongoStorageHandler or BSONSerDe

Hive Support

- Primitive type (Integer, String) => Primitive type in Hive
- Document => A row in Hive
- Field (even if in sub-document) => a field of the row (with schema)
- Array => List
- Subdocument => Struct / Map

Spark Usage

- Use with MapReduce input/output formats
- Create Configuration objects with input/output formats and data URI
- Load/save data using SparkContext Hadoop file API



Examples

https://github.com/lovett89/mongodb-hadoop-workshop

Data Schema - Recommendation System

```
> db.ratings.findOne()
{
    "_id" : ObjectId("5388e41c12569b70c376e9fb"),
    "userid" : 1,
    "movieid" : 122,
    "ts" : ISODate("1996-08-02T11:24:06Z"),
    "rating" : 5
}
```

Read BSON

```
// create base BSON Configuration object
Configuration bsonConfig = new Configuration();
bsonConfig.set("mongo.job.input.format”,
  "com.mongodb.hadoop.BSONFileInputFormat");

JavaRDD<Object> movies = sc.newAPIHadoopFile(HDFS +
  "/movielens/movies.bson",
  BSONFileInputFormat.class,
  Object.class,
  BSONObject.class,
  bsonConfig).map(
    new Function<Tuple2<Object, BSONObject>, Object>() {
        @Override
        public Object call(Tuple2<Object, BSONObject> doc)
                throws Exception {
            return doc._2.get("movieid");
        }
    } );
```

Write BSON

```
// create BSON output RDD from predictions
JavaPairRDD<Object,BSONObject> predictions =
predictedRatings.mapToPair(
  new PairFunction<Rating, Object, BSONObject>() {
    @Override
    public Tuple2<Object, BSONObject> call(Rating rating)
throws Exception {
      DBObject doc = BasicDBObjectBuilder.start()
          .add("userid", rating.user())
          .add("movieid", rating.product())
          .add("rating", rating.rating())
          .add("timestamp", new Date())
          .get();
     // null key means an ObjectId will be generated on
insert
      return new Tuple2<Object, BSONObject>(null, doc);
    }
  }
);
```

Write BSON (cont.)

```
// Create MongoDB output Configuration
Configuration outputConfig = new Configuration();
outputConfig.set("mongo.output.format”,
    "com.mongodb.hadoop.MongoOutputFormat");
outputConfig.set("mongo.output.uri", MONGODB + "." +
    OUTPUT);

predictions.saveAsNewAPIHadoopFile(
     "file:///not-applicable",
     Object.class,
     Object.class,
     MongoOutputFormat.class,
     outputConfig);
```

Questions?