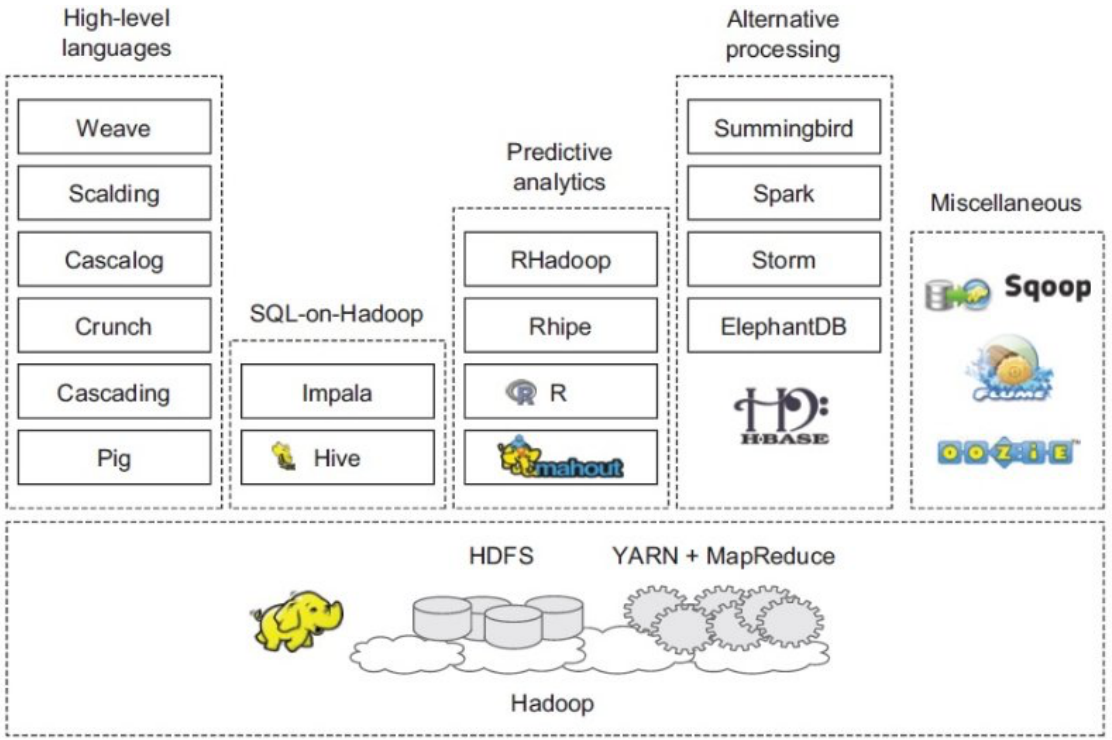
A Glimpse of the Hadoop Echosystem

Hadoop Echosystem

- A cluster is shared among several users in an organization
  - Different services
- HDFS and MapReduce provide the lower layers of the infrastructures
  - Other systems “plug” on top of these
  - Easier way to program applications
  - MapReduce and HDFS are “low level”



HBase

- Hadoop database for random read/write access
  - HBase is an open source, non-relational, distributed “database”
    - modeled after Google's **BigTable**.
  - It runs on top of Hadoop and HDFS, providing BigTable-like capabilities for Hadoop.
  - Eric Brewer’s CAP theorem, HBase is a CP type system.
    - Consistency, availability, partition tolerance.

When to use HBase

- Real big data: billions of rows X millions of columns
  - Data can not store in a single node.
- Random read/write access
- Thousands of operations on big data
- No need of extra features of RDMS like typed columns, secondary indexes, transactions, advanced query languages, etc.

HDFS	Hbase
Good for storing large file	Built on top of HDFS. Good for hosting very large tables like billions of rows X millions of column
Write once. Append to files in some of recent versions but not commonly used	Read/write many
No random read/write	Random read/write
No individual record lookup rather read all data	Fast records lookup(update)

HBase

- Type of NoSql database
  - HBase is really more a "Data Store" than "Data Base". It lacks many of the features you find in an RDBMS, such as typed columns, secondary indexes, triggers, and advanced query languages, …
- Strongly consistent read and write
- Automatic sharding (i.e., "horizontal partitioning")
  - HBase tables are distributed on the cluster via regions, and regions are automatically split and re-distributed as data grows
- Automatic RegionServer failover
- Hadoop/HDFS Integration
- Massively parallelized processing via MapReduce for using HBase as both source and sink.
- Java API for programmatic access, REST for non-Java front-ends.

Gets all the data for the row
hbase> get '/user/user01/customer', 'jsmith'

Limit this to only one column family
hbase> get '/user/user01/customer', 'jsmith', {COLUMNS=>['addr']}

Limit this to a specific column
hbase> get '/user/user01/customer', 'jsmith', {COLUMNS=>['order:numb']}

Scan all rows of table 't1'
hbase> scan 't1'

Specify a timerange
hbase> scan 't1', {TIMERANGE => [1303668804, 1303668904]}

Specify a startrow, limit the result to 10 rows, and only return selected columns
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}

Hive

“The Apache Hive ™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.”

Hive

- An SQL like interface to Hadoop.
- Data warehouse infrastructure built on top of Hadoop
- Provide data summarization, query and analysis
- Query execution via MapReduce
- Hive interpreter transparently converts queries to MapReduce.
  - But other backends are also supported, e.g., Spark
- Open source, developed by Facebook
- Also used by Netflix, Cnet, Digg, eHarmony etc.

```
SELECT customerId, max(total_cost)
FROM hive_purchases
GROUP BY customerId
HAVING count(*) > 3;
```

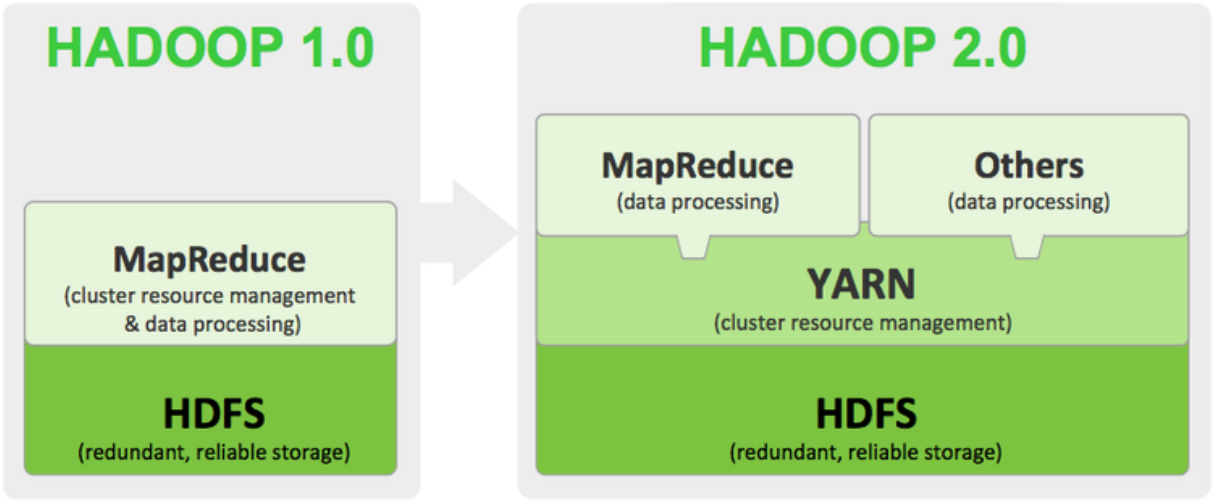
```
1  DROP TABLE IF EXISTS docs;
2  CREATE TABLE docs (line STRING);
3  LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
4  CREATE TABLE word_counts AS 5 SELECT word, count(1) AS count FROM
6  (SELECT explode(split(line, '\s')) AS word FROM docs) temp
7  GROUP BY word
8  ORDER BY word;
```

- Wordcount in Hive
- Just a curiosity – **probably not the typical kind of query**

  https://en.wikipedia.org/wiki/Apache_Hive

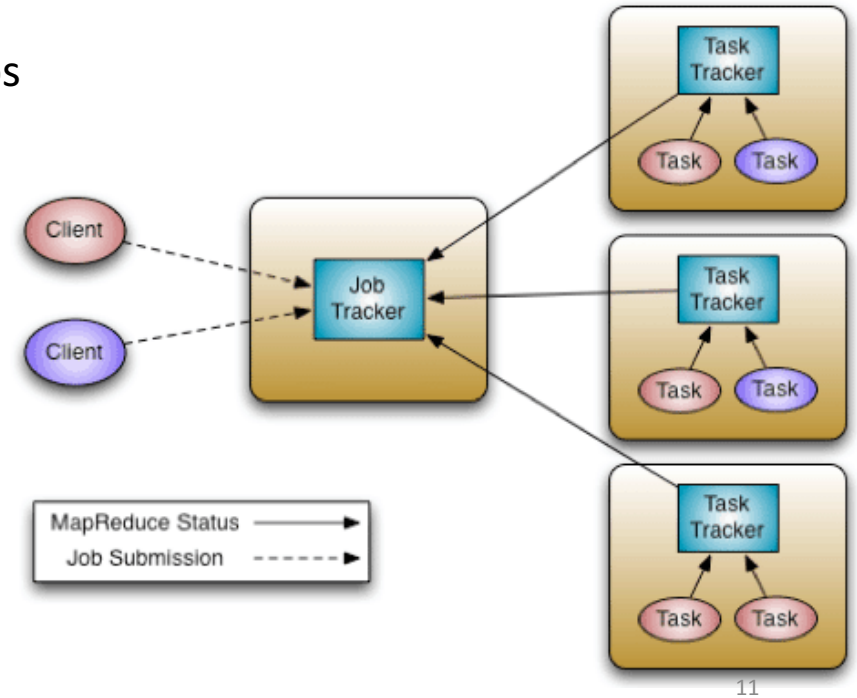
YARN

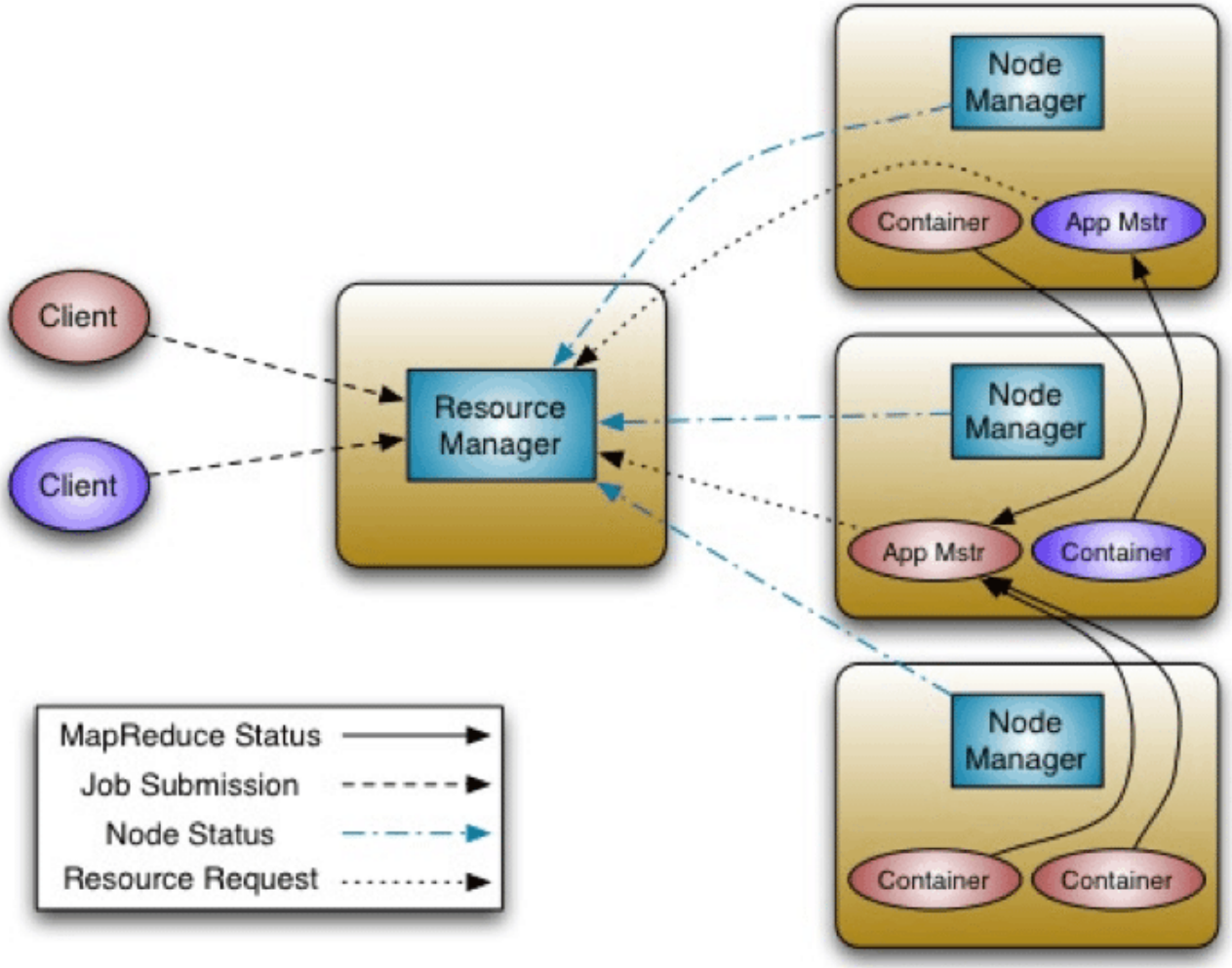
- Yet Another Resource Negotiator
- YARN Application Resource Negotiator(Recursive Acronym)
- Remedies the scalability shortcomings of “classic” MapReduce
- A general purpose framework. MapReduce is one application.



MapReduce Limitations

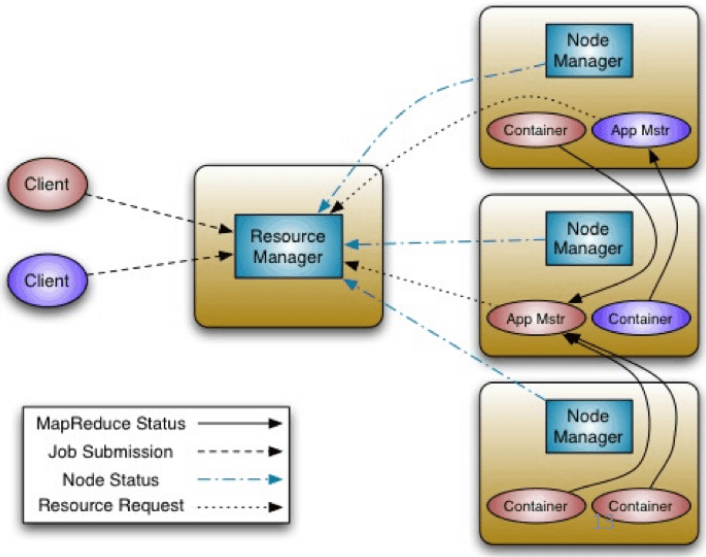
- Scalability
  - Maximum Cluster Size – 4000 Nodes
  - Maximum Concurrent Tasks – 40000
  - Coarse synchronization in Job Tracker
- Single point of failure
  - Failure kills all queued and running jobs
  - Jobs need to be resubmitted by users
  - Restart is tricky due to complex state





For a (short) introduction:
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

- Splits up the major functions of JobTracker:
- The ResourceManager has two components: Scheduler and ApplicationsManager.
- Scheduler: performs no monitoring or tracking of status for the application.
  - No guarantees about restarting failed tasks either due to application failure or hardware failures.
  - Performs its scheduling function based on the resource requirements of the applications;
  - Abstract notion of a resource *Container* (memory, cpu, disk, network etc.)
- The ApplicationsManager is responsible for accepting job-submissions, negotiating the first container for executing the application specific ApplicationMaster
  - Provides the service for restarting the ApplicationMaster container on failure.
- ApplicationMaster (one per application)
  - Negotiate appropriate resource containers from the Scheduler
  - Tracks their status and monitoring for progress.
  - Runs as a normal container.
  - Framework specific library
  - Works with the NodeManager(s) to execute and monitor the tasks.
- NodeManager (NM)
  - A new per-node slave is responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting to the Resource Manager.



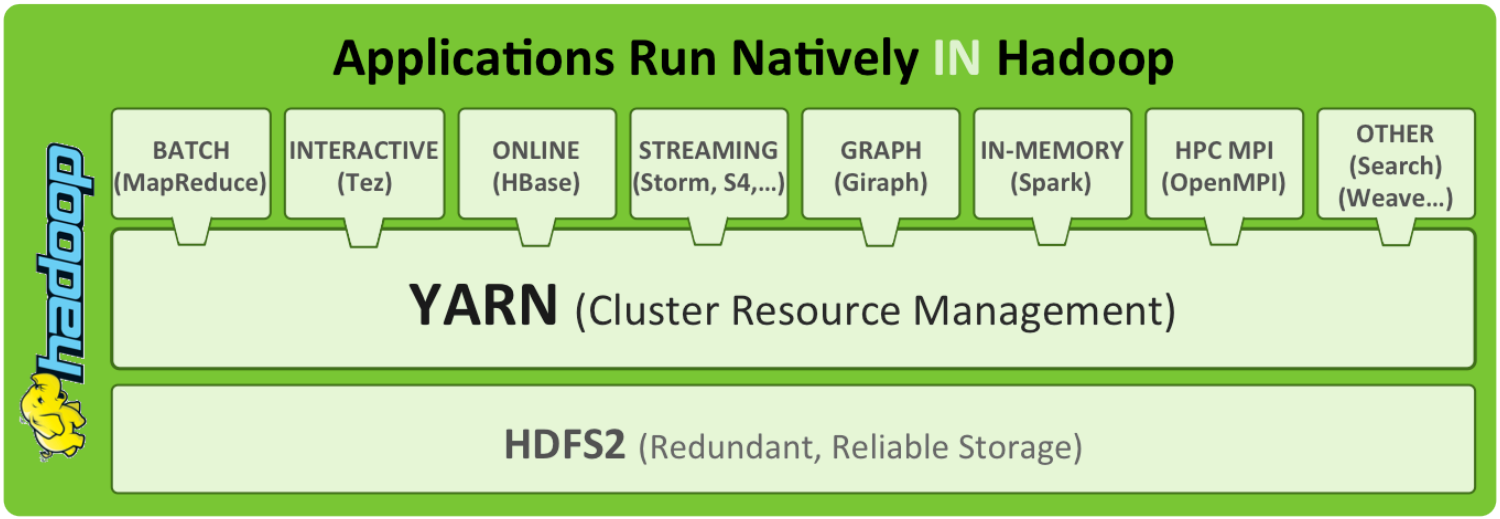
YARN

- Fault Tolerance and Availability
  - Resource Manager
    - No single point of failure – state saved in ZooKeeper
    - Application Masters are restarted automatically
  - Optional failover via application-specific checkpoint
    - MapReduce applications pick up where they left off via state saved in HDFS

- Scalability
  - 6000 - 10000 Nodes
  - 100 000+ Concurrent Tasks
  - 10 000+ Jobs

YARN

- Support for paradigms other than MapReduce (**Multi tenancy**)
  - HBase on YARN (HOYA), Machine Learning: Spark, Graph processing: Giraph, Real-time processing: Storm
- Enabled by allowing the use of paradigm-specific application master
- Run all on the same Hadoop cluster!



Sources

- Hadoop 2.0 and YARN - Subash D’Souza
- https://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/
- https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
- http://hbase.apache.org/book.html#arch.overview