ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Nhập môn

Khoa học dữ liệu

(IT4142)

Nội dung môn học

- Lecture 1: Tổng quan về Khoa học dữ liệu
- Lecture 2: Thu thập và tiền xử lý dữ liệu
- Lecture 3: Làm sạch và tích hợp dữ liệu
- Lecture 4: Phân tích và khám phá dữ liệu
- Lecture 5: Trực quan hoá dữ liệu
- Lecture 6: Trực quan hoá dữ liệu đa biến
- Lecture 7: Học máy
- **Lecture 8: Phân tích dữ liệu lớn**
- Lecture 9: Báo cáo tiến độ bài tập lớn và hướng dẫn
- Lecture 10+11: Phân tích một số kiểu dữ liệu
- Lecture 12: Đánh giá kết quả phân tích

Nội dung

- Tổng quan về quản trị dữ liệu lớn
- Hệ sinh thái Hadoop

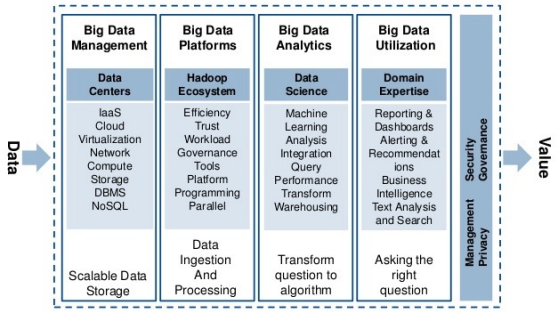
Tổng quan

Dữ liệu được ví như nguồn tài nguyên dầu mỏ mới

Đặc điểm 5'V của dữ liệu lớn

Dữ liệu lớn là tập dữ liệu quá lợn hoặc là quá phức tạp mà các nền tảng lưu trữ và xử lý dữ liệu truyền thống không đáp ứng được.

Các tầng công nghệ cho dữ liệu lớn



Quản lý dữ liệu phải khả mở

- Scalability
  - Khả năng quản lý lượng dữ liệu lớn không ngừng tăng lên theo thời gian.
- Accessibility
  - Cho phép đọc ghi I/O dữ liệu hiệu quả.
- Transparency
  - Truy cập dữ liệu dễ dàng, vị trí lưu trữ dữ liệu trên hệ thống là trong suốt với người dùng cuối.
- Availability
  - Khả năng chống chịu lỗi, khi tăng số lượng người dùng, khi hỏng hóc.

Xử lý và tích hợp dữ liệu phải khả mở

- Tích hợp dữ liệu
  - Dữ liệu có định dạng khác nhau
  - Dữ liệu tồn tại ở các mô hình và lược đồ dữ liệu khác nhau
  - Các vấn đề liên quan đến an toàn an ninh thông tin, quyền riêng tư
- Xử lý dữ liệu
  - Xử lý khối lượng dữ liệu rất lớn
  - Xử lý luồng dữ liệu lớn
  - Xử lý dữ liệu song song, phân tán truyền thống (OpenMP, MPI)
    - Phức tạp, khó học
    - Khả năng khả mở có giới hạn
    - Cơ chế chịu lỗi kém
    - Chi phí hạ tầng đắt đỏ
  - Kiến trúc xử lý dữ liệu luồng dữ liệu lớn
    - Spark mini-batch
    - Apache Flink

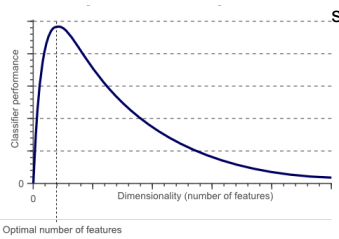
Các giải thuật phân tích dữ liệu khả mở

- Làm nhỏ lại dữ liệu cho phù hợp với các giải thuật truyền thống
  - Eg. Sub-sampling
  - Eg. Principal component analysis
  - Eg. Feature extraction and feature selection
- Song song hoá các giải thuật học máy
  - Eg. k-nn classification based on MapReduce
  - Eg. scaling-up support vector machines (SVM) by a divide and-conquer approach

Eg. Sự bùng nổ số chiều trong dữ liệu (Curse of dimensionality)

- Số lượng mẫu cần cho mô hình học tăng lên khi số chiều dữ liệu tăng
- Trong thực tiễn: Số lượng mẫu để học thường cố định

=> Độ số chiều trong d



Hệ sinh thái Hadoop

Apache Hadoop
Hệ thống tệp tin Hadoop (HDFS)
Mô thức xử lý dữ liệu MapReduce
Các thành phần khác trong hệ sinh thái Hadoop

Mục tiêu của Hadoop

- Mục tiêu chính
  - **Lưu trữ dữ liệu khả mở, tin cậy**
  - **Powerful data processing**
  - Efficient visualization
- Với thách thức
  - Thiết bị lưu trữ tốc độ chậm, máy tính thiếu tin cậy, lập trình song song phân tán không dễ dàng

Giới thiệu về Apache Hadoop

- **Lưu trữ và xử lý dữ liệu khả mở, tiết kiệm chi phí**
  - Xử lý dữ liệu phân tán với mô hình lập trình đơn giản, thân thiện hơn như MapReduce
  - Hadoop thiết kế để mở rộng thông qua kỹ thuật scale-out, tăng số lượng máy chủ
  - Thiết kế để vận hành trên phần cứng phổ thông, có khả năng chống chịu lỗi phần cứng
- Lấy cảm hứng từ kiến trúc dữ liệu của Google

Các thành phần chính của Hadoop

- Lưu trữ dữ liệu: Hệ thống tệp tin phân tán Hadoop (HDFS)
- Xử lý dữ liệu: MapReduce framework
- Các tiện ích hệ thống:
  - Hadoop Common: Các tiện ích chung hỗ trợ các thành phần của Hadoop.
  - Hadoop YARN: Một framework quản lý tài nguyên và lập lịch trong cụm Hadoop.

Hadoop giải quyết bài toán khả mở

- Thiết kế hướng "phân tán" ngay từ đầu
  - Hadoop mặc định thiết kế để triển khai trên cụm máy chủ
- Các máy chủ tham gia vào cụm được gọi là các Nodes
  - Mỗi node tham gia vào cả 2 vai trò lưu trữ và tính toán
- Hadoop mở rộng bằng kỹ thuật scale-out
  - Có thể tăng cụm Hadoop lên hàng chục ngàn nodes

Hadoop giải quyết bài toán chịu lỗi

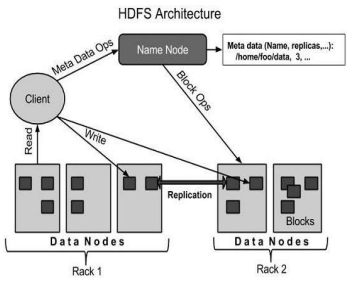
- Với việc triển khai trên cụm máy chủ phổ thông
  - Hỏng hóc phần cứng là chuyện thường ngày, không phải là ngoại lệ
  - Hadoop chịu lỗi thông qua kỹ thuật "dư thừa"
- Các tệp tin trong HDFS được phân mảnh, nhân bản ra các nodes trong cụm
  - Nếu một node gặp lỗi, dữ liệu ứng với nodes đó được tái nhân bản qua các nodes khác
- Công việc xử lý dữ liệu được phân mảnh thành các tác vụ độc lập
  - Mỗi tác vụ xử lý một phần dữ liệu đầu vào
  - Các tác vụ được thực thi song song với các tác vụ khác
  - Tác vụ lỗi sẽ được tái lập lịch thực thi trên node khác
- Hệ thống Hadoop thiết kế sao cho các lỗi xảy ra trong hệ thống được xử lý tự động, không ảnh hưởng tới các ứng dụng phía trên

Tổng quan về HDFS

- HDFS cung cấp khả năng lưu trữ tin cậy và chi phí hợp lý cho khối lượng dữ liệu lớn
- Tối ưu cho các tập tin kích thước lớn (từ vài trăm MB tới vài TB)
- HDFS có không gian cây thư mục phân cấp như UNIX (vd., /hust/soict/hello.txt)
  - Hỗ trợ cơ chế phân quyền và kiểm soát người dùng như của UNIX
- Khác biệt so với hệ thống tập tin trên UNIX
  - Chỉ hỗ trợ thao tác ghi thêm dữ liệu vào cuối tệp (APPEND)
  - Ghi một lần và đọc nhiều lần

Kiến trúc của HDFS

- Kiến trúc Master/Slave
- HDFS master: name node
  - Quản lý không gian tên và siêu dữ liệu ánh xạ tệp tin tới vị trí các chunks
  - Giám sát các data node
- HDFS slave: data node
  - Trực tiếp thao tác I/O các chunks



Nguyên lý thiết kế cốt lõi của HDFS

- I/O pattern
  - Chỉ ghi thêm (Append)→ giảm chi phí điều khiển tương tranh
- Phân tán dữ liệu
  - Tệp được chia thành các chunks lớn (64 MB)

  → Giảm kích thước metadata

  → Giảm chi phí truyền dữ liệu
- Nhân bản dữ liệu
  - Mỗi chunk thông thường được sao làm 3 nhân bản
- Cơ chế chịu lỗi
  - Data node: sử dụng cơ chế tái nhân bản
  - Name node
    - Sử dụng Secondary Name Node
    - SNN hỏi data nodes khi khởi động thay vì phải thực hiện cơ chế đồng bộ phức tạp với primary NN

Mô thức xử lý dữ liệu MapReduce

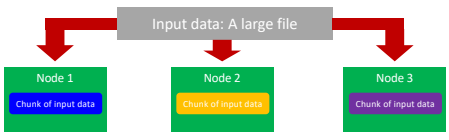
- MapReduce là mô thức xử lý dữ liệu mặc định trong Hadoop
- MapReduce không phải là ngôn ngữ lập trình, được đề xuất bởi Google
- Đặc điểm của MapReduce
  - Đơn giản (Simplicity)
  - Linh hoạt (Flexibility)
  - Khả mở (Scalability)

A MR job = {Isolated Tasks}n

- Mỗi chương trình MapReduce là một công việc (job) được phân rã làm nhiều tác vụ độc lập (task) và các tác vụ này được phân tán trên các nodes khác nhau của cụm để thực thi
- Mỗi tác vụ được thực thi độc lập với các tác vụ khác để đạt được tính khả mở
  - Giảm truyền thông giữa các node máy chủ
  - Tránh phải thực hiện cơ chế đồng bộ giữa các tác vụ

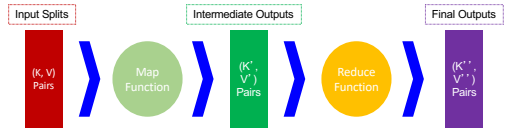
Dữ liệu cho MapReduce

- MapReduce trong môi trường Hadoop thường làm việc với dữ liệu đa có sẵn trên HDFS
- Khi thực thi, mã chương trình MapReduce được gửi tới các node đã có dữ liệu tương ứng



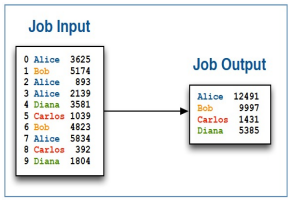
Chương trình MapReduce

- Lập trình với MapReduce cần cài đặt 2 hàm Map và Reduce
  - 2 hàm này được thực thi bởi các tiến trình Mapper và Reducer tương ứng.
- Trong chương trình MapReduce, dữ liệu được nhìn nhận như là các cặp khóa – giá trị (key – value)
- Các hàm Map và Reduce nhận đầu vào và trả về đầu ra các cặp (key – value)



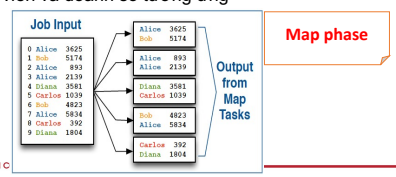
Ví dụ về MapReduce

- Đầu vào: tệp văn bản chứa thông tin về order ID, employee name, and sale amount
- Đầu ra : Doanh số bán (sales) theo từng nhân viên (employee)



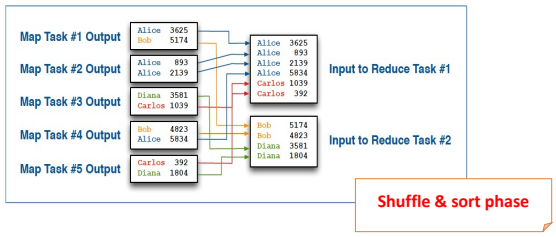
Bước Map

- Dữ liệu đầu vào được xử lý bởi nhiều tác vụ Mapping độc lập
  - Số lượng các tác vụ Mapping được xác định theo lượng dữ liệu đầu vào (~ số chunks)
  - Mỗi tác vụ Mapping xử lý một phần dữ liệu (chunk) của khối dữ liệu ban đầu
- Với mỗi tác vụ Mapping, Mapper xử lý lần lượt từng bản ghi đầu vào
  - Với mỗi bản ghi đầu vào (key-value), Mapper đưa ra 0 hoặc nhiều bản ghi đầu ra (key – value trung gian)
- Trong ví dụ này, tác vụ Mapping đơn giản đọc từng dòng văn bản và đưa ra tên nhân viên và doanh số tương ứng



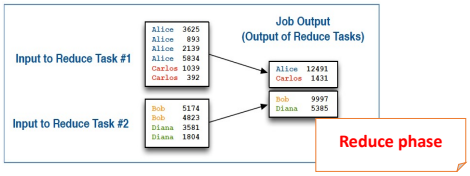
Bước shuffle & sort

- Hadoop tự động sắp xếp và gộp đầu ra của các Mappers theo các partitions
  - Mỗi partitions là đầu vào cho một Reducer

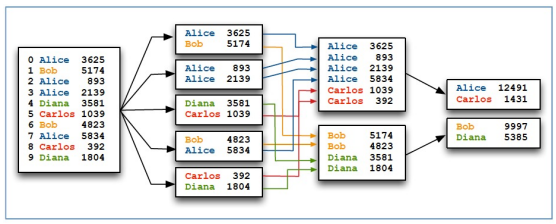


Bước Reduce

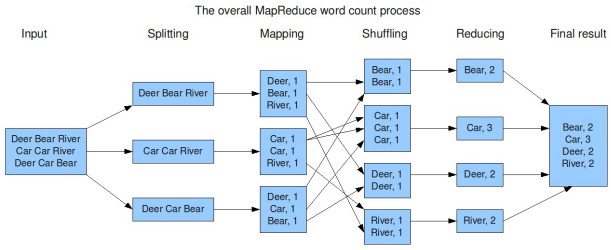
- Reducer nhận dữ liệu đầu vào từ bước shuffle & sort
  - Tất cả các bản ghi key – value tương ứng với một key được xử lý bởi một Reducer duy nhất
  - Giống bước Map, Reducer xử lý lần lượt từng key, mỗi lần với toàn bộ các values tương ứng
- Trong ví dụ, hàm reduce đơn giản là tính tổng doanh số cho từng nhân viên, đầu ra là các cặp key – value tương ứng với tên nhân viên – doanh số tổng



Luồng dữ liệu



Luồng dữ liệu với bài toán Word Count



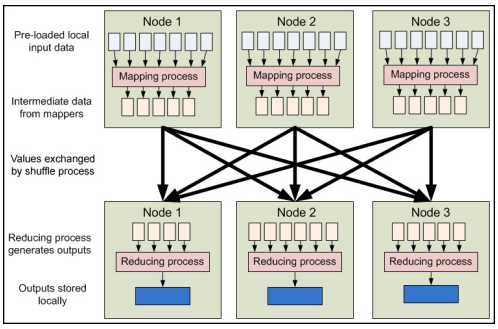
Chương trình Word Count thực tế (1)

```
9  import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import org.apache.hadoop.util.GenericOptionsParser;
15
16
17
18
19 public class WordCount {
20 public static void main(String [] args) throws Exception
21 {
22 Configuration c=new Configuration();
23 String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
24 Path input=new Path(files[0]);
25 Path output=new Path(files[1]);
26 Job j=new Job(c,"wordcount");
27 j.setJarByClass(WordCount.class);
28 j.setMapperClass(MapForWordCount.class);
29 j.setReducerClass(ReduceForWordCount.class);
30 j.setOutputKeyClass(Text.class);
31 j.setOutputValueClass(IntWritable.class);
32 FileInputFormat.addInputPath(j, input);
33 FileOutputFormat.setOutputPath(j, output);
34 System.exit(j.waitForCompletion(true)?0:1);
35 }
```

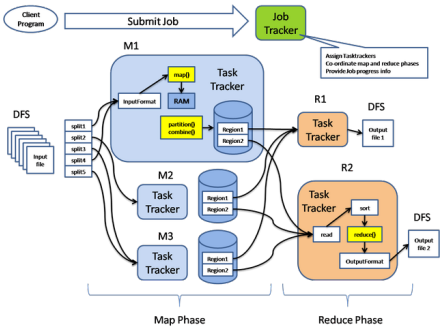

Chương trình Word Count thực tế (2)

```
36 public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
37 public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
38 {
39 String line = value.toString();
40 String[] words=line.split(",");
41 for(String word: words )
42 {
43       Text outputKey = new Text(word.toUpperCase().trim());
44   IntWritable outputValue = new IntWritable(1);
45   con.write(outputKey, outputValue);
46 }
47 }
48 }
49
50 public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
51 {
52 public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedE
53 {
54 int sum = 0;
55    for(IntWritable value : values)
56    {
57    sum += value.get();
58    }
59    con.write(word, new IntWritable(sum));
60 }
```


MapReduce trên môi trường phân tán



Vai trò của Job tracker và Task tracker



Các thành phần khác trong hệ sinh thái Hadoop

- Ngoài HDFS và MapReduce, hệ sinh thái Hadoop còn nhiều hệ thống, thành phần khác phục vụ
  - Phân tích dữ liệu
  - Tích hợp dữ liệu
  - Quản lý luồng
  - Vvv
- Các thành phần này không phải là 'core Hadoop' nhưng là 1 phần của hệ sinh thái Hadoop
  - Hầu hết là mã nguồn mở trên Apache

Apache Pig

- Apache Pig cung cấp giao diện xử lý dữ liệu mức cao
  - Pig đặc biệt tốt cho các phép toán Join và Transformation
- Trình biên dịch của Pig chạy trên máy client
  - Biến đổi PigLatin script thành các jobs của MapReduce
  - Đệ trình các công việc này lên cụm tính toán

```
people = LOAD '/user/training/customers' AS (cust_id, name);
orders = LOAD '/user/training/orders' AS (ord_id, cust_id, cost);
groups = GROUP orders BY cust_id;
totals = FOREACH groups GENERATE group, SUM(orders.cost) AS t;
result = JOIN totals BY group, people BY cust_id;
DUMP result;
```


Apache Hive

- Cũng là một lớp trừu tượng mức cao của MapReduce
  - Giảm thời gian phát triển
  - Cung cấp ngôn ngữ HiveQL: SQL-like language
- Trình biên dịch Hive chạy trên máy client
  - Chuyển HiveQL script thành MapReduce jobs
  - Đệ trình các công việc này lên cụm tính toán

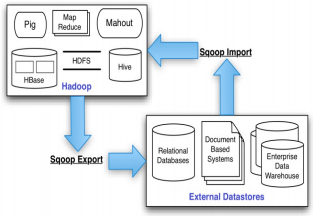

```
SELECT customers.cust_id, SUM(cost) AS total
  FROM customers
  JOIN orders
    ON customers.cust_id = orders.cust_id
GROUP BY customers.cust_id
ORDER BY total DESC;
```


Apache Hbase

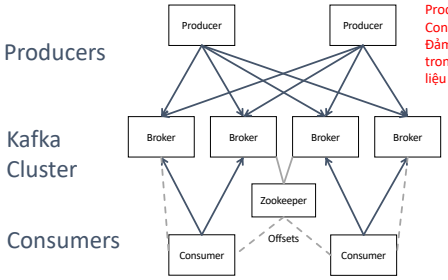
- HBase là một CSDL cột mở rộng phân tán, lưu trữ dữ liệu trên HDFS
  - Được xem như là hệ quản trị CSDL của Hadoop
- Dữ liệu được tổ chức về mặt logic là các bảng, bao gồm rất nhiều dòng và cột
  - Kích thước bảng có thể lên đến hàng Terabyte, Petabyte
  - Bảng có thể có hàng ngàn cột
- Có tính khả mở cao, đáp ứng băng thông ghi dữ liệu tốc độ cao
  - Hỗ trợ hàng trăm ngàn thao tác INSERT mỗi giây (/s)
- Tuy nhiên về các chức năng thì còn rất hạn chế khi so sánh với hệ QTCSDL truyền thống
  - Là NoSQL : không có ngôn ngữ truy vấn mức cao như SQL
  - Phải sự dụng API để scan/ put/ get/ dữ liệu theo khóa

Apache Sqoop

- Sqoop là một công cụ cho phép trung chuyển dữ liệu theo khối từ Apache Hadoop và các CSDL có cấu trúc như CSDL quan hệ
- Hỗ trợ import tất cả các bảng, một bảng hay 1 phần của bảng vào HDFS
  - Thông qua Map only hoặc MapReduce job
  - Kết quả là 1 thư mục trong HDFS chứa các tập tin văn bản phân tách các trường theo ký tự phân tách (vd. , hoặc \t)
- Hỗ trợ export dữ liệu ngược trở lại từ Hadoop ra bên ngoài



Apache Kafka



Producers không cần biết Consumers
Đảm bảo sự linh hoạt và tin cậy trong quá trình trung chuyển dữ liệu giữa các bên

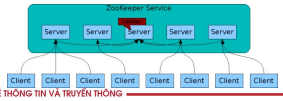
Kafka cho phép phân tách mạch lạc các thành phần tham gia vào luồng dữ liệu

Apache Oozie

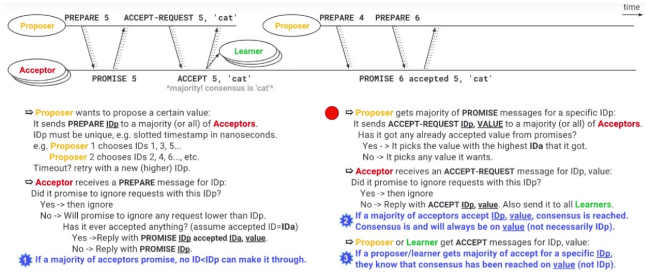
- Oozie là một hệ thống lập lịch luồng công việc để quản lý các công việc thực thi trên cụm Hadoop
- Luồng workflow của Oozie là đồ thị vòng có hướng (Directed Acyclical Graphs (DAGs)) của các khối công việc
- Oozie hỗ trợ đa dạng các loại công việc
  - Thực thi MapReduce jobs
  - Thực thi Pig hay Hive scripts
  - Thực thi các chương trình Java hoặc Shell
  - Tương tác với dữ liệu trên HDFS
  - Chạy chương trình từ xa qua SSH
  - Gửi nhận email

Apache Zookeeper

- Apache ZooKeeper là một dịch vụ cung cấp các chức năng phối hợp phân tán độ tin cậy cao
  - Quản lý thành viên trong nhóm máy chủ
  - Bầu cử leader
  - Quản lý thông tin cấu hình động
  - Giám sát trạng thái hệ thống
- Đây là các service lõi, tối quan trọng trong hệ thống phân tán



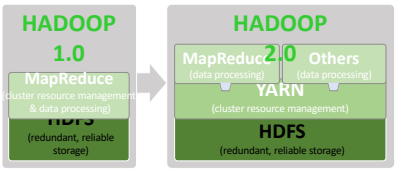
PAXOS algorithm



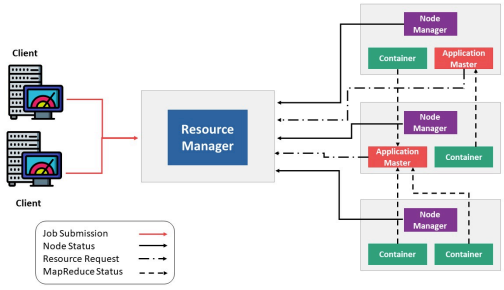
https://www.youtube.com/watch?v=d7nAGI_NZPk

YARN – Yet Another Resource Negotiator

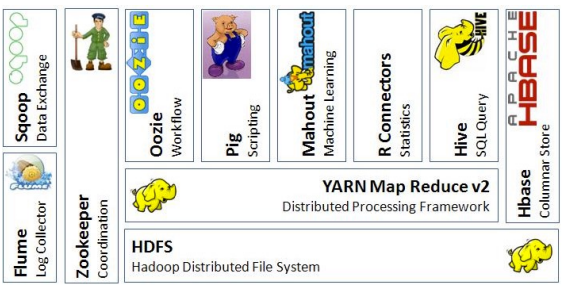
- Nodes có tài nguyên là – bộ nhớ và CPU cores
- YARN đóng vai trò cấp phát lượng tài nguyên phù hợp cho các ứng dụng khi có yêu cầu
- YARN được đưa ra từ Hadoop 2.0
  - Cho phép MapReduce và non MapReduce cùng chạy trên 1 cụm Hadoop
  - Với MapReduce job, vai trò của job tracker được thực hiện bởi application tracker



Ví dụ về cấp phát trên YARN



Bức tranh tổng thể hệ sinh thái Hadoop



Hortonworks Data Platform Sandbox

Demo

Các platform quản lý dữ liệu lớn

SOICT
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Chân thành cảm ơn!

soict.hust.edu.vn/ fb.com/groups/soict