

HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

Chương 2

Hệ sinh thái Hadoop

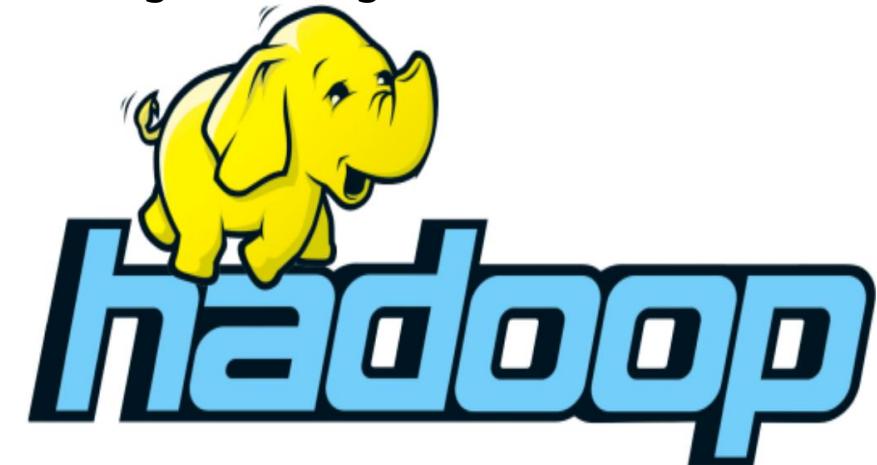
ONE LOVE. ONE FUTURE.

Nội dung

- Apache Hadoop •
Hệ thống tệp tin Hadoop (HDFS) •
MapReduce xử lý dữ liệu mô hình •
Các thành phần khác trong hệ thống sinh thái Hadoop

Mục tiêu của Hadoop

- Mục tiêu chính •
 - Lưu trữ khả năng mở dữ liệu, tin cậy • Xử lý dữ liệu mạnh mẽ
 - Trực quan hóa hiệu quả
- Với thử thách thức
 - Thiết bị lưu trữ tốc độ chậm, máy tính thiếu tin cậy, lập trình phân tích bài hát không dễ dàng



Giới thiệu về Apache Hadoop

- Lưu trữ và xử lý dữ liệu mở, tiết kiệm chi phí
 - Xử lý phân tích dữ liệu với mô hình trình đơn giản, thân thiện hơn MapReduce
 - Thiết kế Hadoop để mở rộng quy mô kỹ thuật thông tin, tăng số lượng máy chủ
 - Thiết kế để vận hành hành động trên phần cứng phổ quát thông tin, có khả năng chịu đựng phần cứng lỗi
- Lấy cảm hứng từ dữ liệu kiến trúc của Google

Các thành phần chính của Hadoop

- Lưu trữ dữ liệu: Hệ thống tệp phân phối Hadoop (HDFS)
- Xử lý dữ liệu: MapReduce framework
- Hệ thống tiện ích:
 - 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 mở

- Thiết kế định hướng “phân tán” ngay từ đầu
 - Mặc định thiết kế của Hadoop để phát triển khai báo trên Cụm máy chủ
- Máy chủ tham gia cụm được gọi là các Các nút
 - Mỗi nút tham gia vào cả 2 vai trò lưu trữ và tính toán toán
- **Mở rộng Hadoop bằng kỹ thuật mở rộng quy mô**
 - Có thể tăng tốc Hadoop lên hàng ngàn nút

Hadoop giải quyết bài toán chịu lỗi

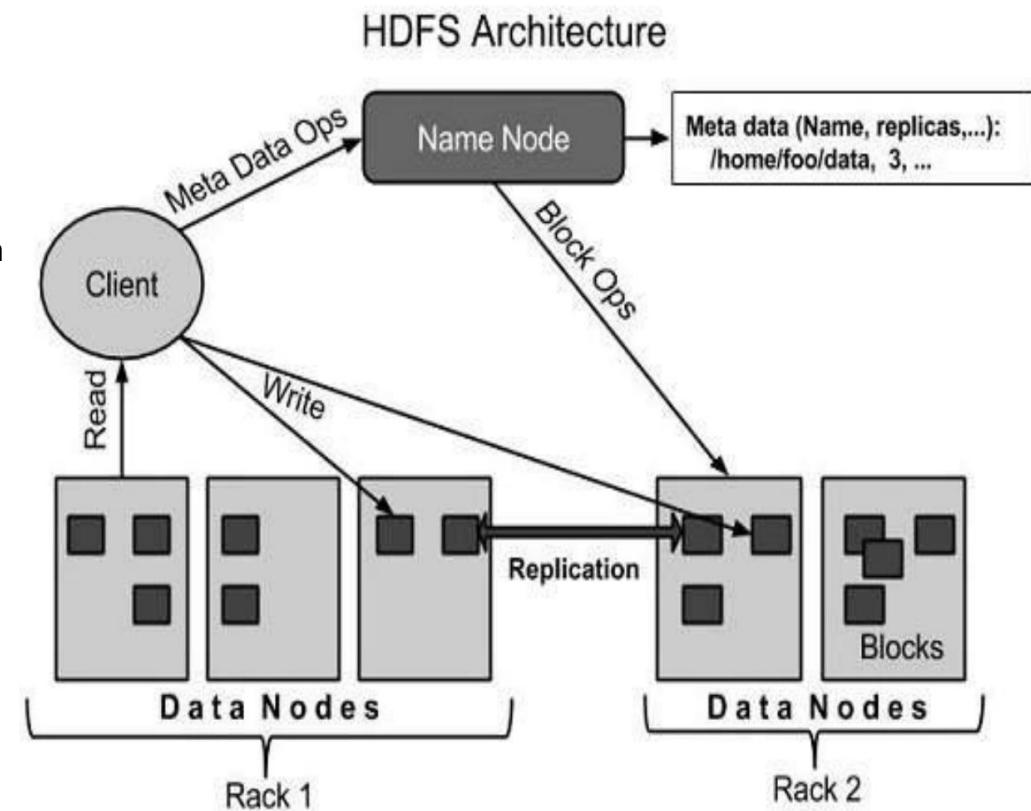
- Với việc phát triển khai trên cụm máy chủ phổ thông
 - Hỗn phần cứng là chuyện thường ngày, không phải là ngoại lệ
 - **Hadoop chịu lỗi thông qua kỹ thuật “dư thừa”**
- Các tệp tệp trong HDFS được phân mảnh, nhân bản ra các nút trong cụm
 - If an node error, data match node which bererenhân bản qua các nút khác
- Công việc xử lý dữ liệu được phân mảnh thành các tác vụ độc lập
 - Mỗi tác vụ xử lý một phần đầu dữ liệu
 - Các tác vụ được thực thi song song với các tác vụ khác
 - Tác vụ lỗi sẽ được tái lập lịch thực thi trên nút khác
 - **Thiết kế hệ thống Hadoop 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 hợp lý cho khối lượng dữ liệu lớn
- Ưu tiên tối đa cho kích thước tệp lớn (từ một số MB tới nhiều TB)
- HDFS không có phân cấp thư mục cây như UNIX
(ví dụ: /hust/soict/hello.txt)
 - Hỗ trợ cơ chế phân quyền và kiểm soát người dùng như của UNIX
- Khác biệt với tập tin hệ thống trên UNIX
 - Hoạt động hỗ trợ duy nhất ghi thêm dữ liệu vào tệp cuối cùng (APPEND)
 - Ghi một lần và đọc nhiều lần

Kiến trúc HDFS

- Kiến trúc Master/Slave
- HDFS master: tên nút
 - Quản lý không có tên và tệp lập trình siêu dữ liệu để định vị các khôi
 - Giám sát các nút dữ liệu
- HDFS slave: nút dữ liệu
 - Hoạt động I/O các chunk trực tiếp



Nguyên lý thiết kế cốt lõi của HDFS

- Mẫu I/O
 - Chỉ ghi thêm (Append) giảm chi phí điều khiển tương tranh
- Phân tán dữ liệu
 - Tệp được chia thành các khối lớn (64 MB)
Giảm siêu dữ liệu kích thước
Giảm chi phí truyền dữ liệu •

Nhân bản dữ liệu

- Mỗi chunk thông thường được sao chép thành 3 nhân
- Cơ chế chịu lỗi • Nút
 - Sử dụng cơ chế tái sinh bản sao • Nút tên
 - Sử dụng Nút tên phụ • SNN hỏi
- các nút dữ liệu khi khởi động thay vì phải thực thi cơ chế đồng bộ phức tạp với NN chính

MapReduce xử lý dữ liệu mô tả

- MapReduce là mặc định xử lý dữ liệu mô-đun trong Hadoop
- MapReduce không phải là trình lập ngôn ngữ, được xuất bản bởi Google
- Đặc điểm

của MapReduce

- Đơn giản (Đơn giản)

- Linh hoạt (Flexibility)
- Khả năng mở rộng (Scalability)

Một công việc MR = {Nhiệm vụ riêng lẻ}n

- Mỗi chương trình MapReduce là một công việc (công việc) được phân chia làm nhiều tác vụ độc lập (nhiệm vụ) và các tác vụ này được phân tán trên các nút khác nhau của cụm để thực thi •

Mỗi tác vụ được thực thi độc lập thiết lập với các tác vụ khác để đạt được khả năng

mở rộng • Giảm thiểu truyền thông giữa các nút máy chủ • Tránh phải thực hiện cơ chế đồng bộ giữa các tác vụ

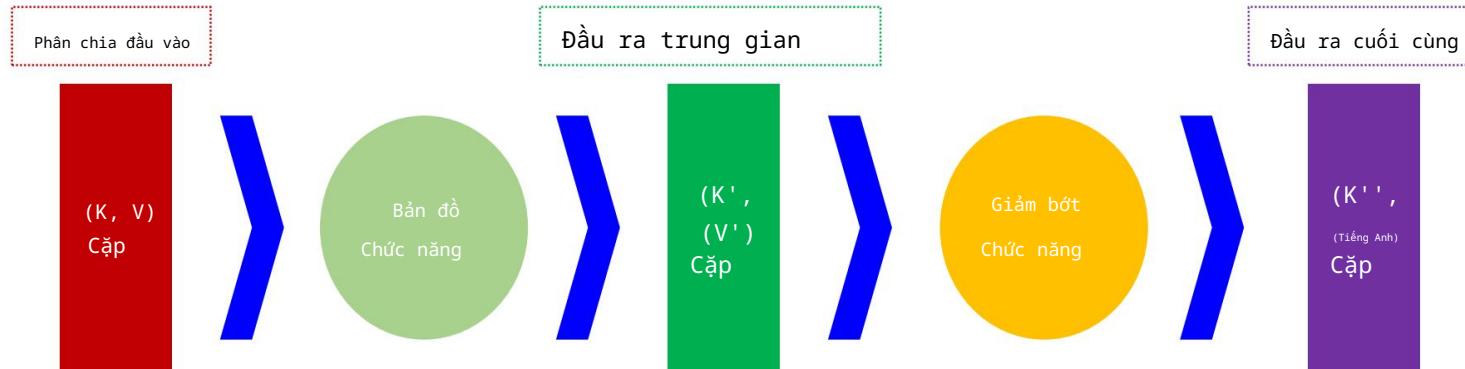
Data cho MapReduce

- MapReduce trong môi trường Hadoop thường xuyên hoạt động với lượng dữ liệu sẵn có trên HDFS
- Khi thực thi, mã chương trình MapReduce được gửi tới các nút có dữ liệu tương ứng



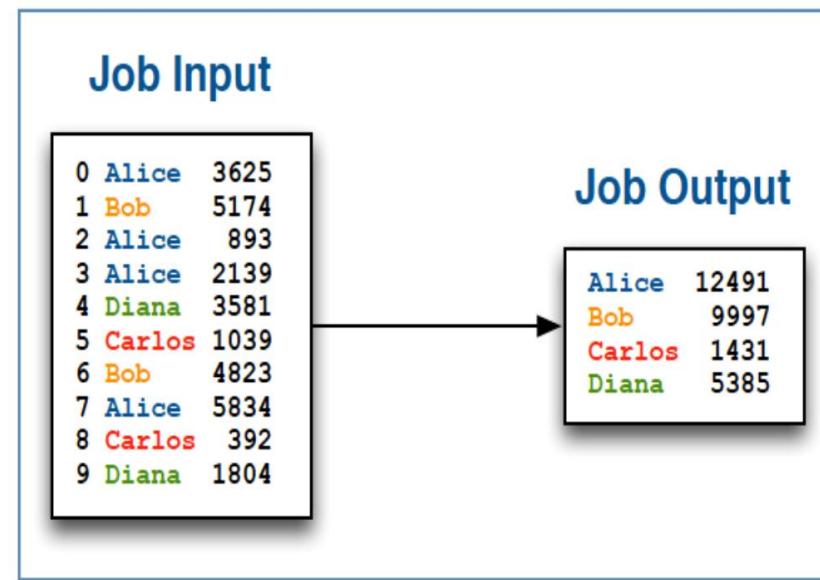
Chương trình MapReduce

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



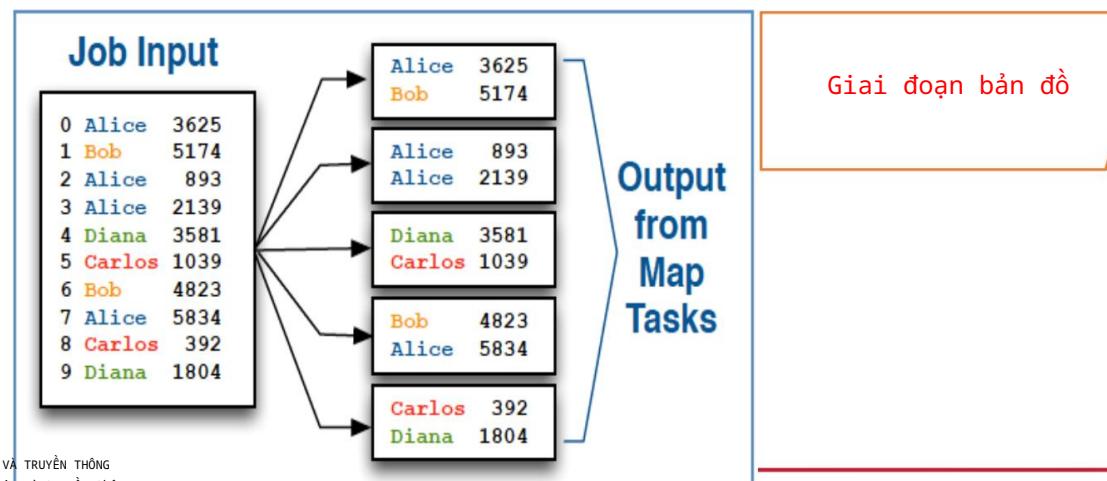
Ví dụ về MapReduce

- Đầu vào: tệp văn bản chứa thông tin về ID đơn hàng, tên nhân viên và số tiền bán ra
- Đầu ra : Doanh số bán (bán hàng) theo từng nhân viên (nhân viên)



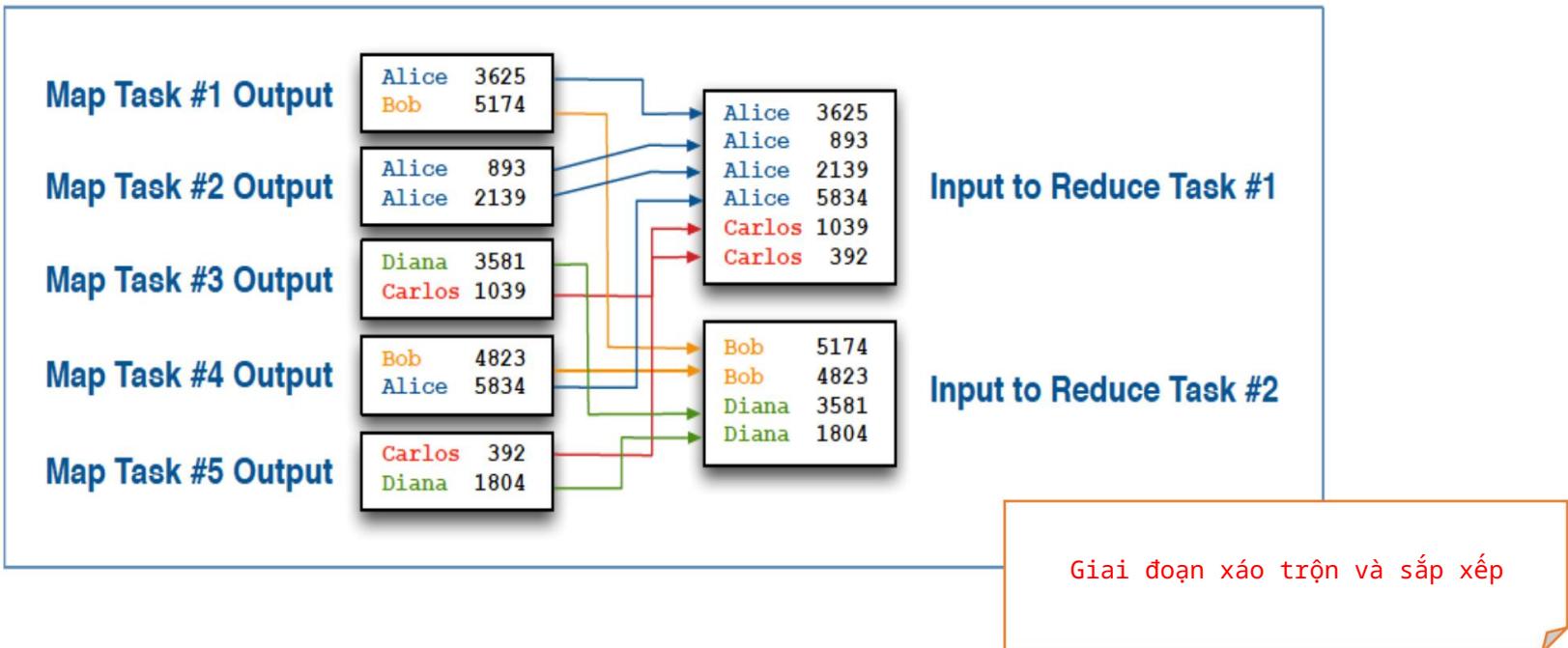
Bản đồ Bước

- Dữ liệu đầu vào được xử lý bởi nhiều tác vụ Lập bản đồ độc lập
 - Số lượng tác vụ Ánh xạ được xác định theo lượng dữ liệu đầu vào (~ số những miếng nhỏ)
 - 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 một lần như từng bản ghi đầu vào
 - Với mỗi bản ghi đầu vào (khóa-giá trị), Mapper đưa ra 0 hoặc nhiều bản ghi đầu ra (khóa - giá trị 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



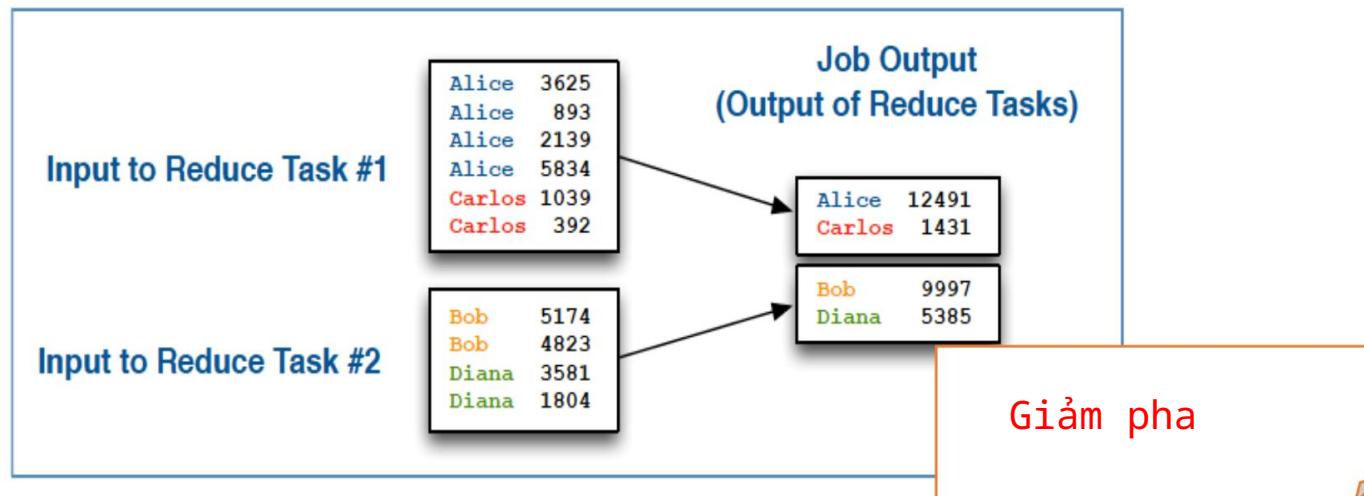
Bước xáo trộn & sắp xếp

- Hadoop tự động sắp xếp và đầu ra của các loại
Người lập bản đồ theo các phân
vùng • Mỗi phân vùng là đầu vào cho một Giảm tốc

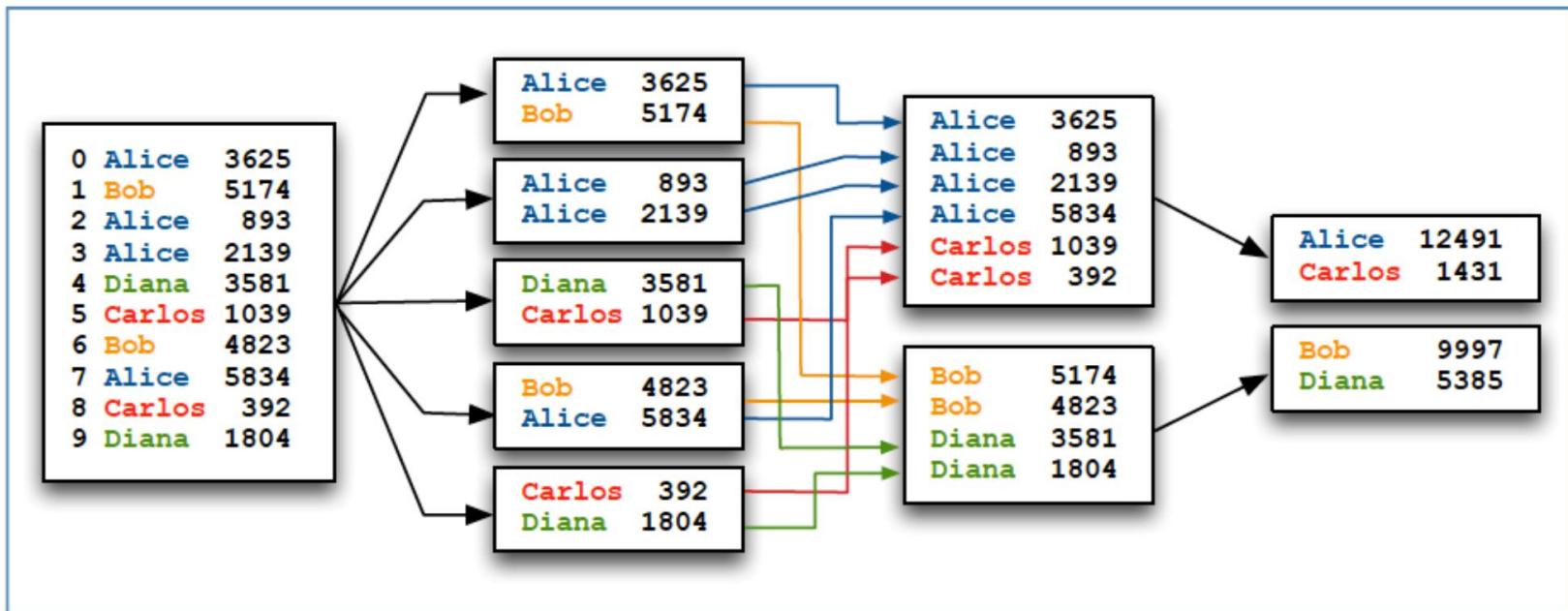


Bước Giảm

- Bộ giảm tốc nhận đầu vào ngẫu nhiên & sắp xếp dữ liệu từ bước
 - Tắt cả các khóa ghi - giá trị tương ứng với một khóa được xử lý bởi một Bộ giảm tốc duy nhất
 - Giống bước Bảng đồ, Bộ xử lý giảm tốc từng lần một, mỗi lần có toàn bộ các giá trị tương ứng
- Trong ví dụ, hàm less đơn giản là tính tổng doanh số cho từng nhân viên, đầu ra là các cặp key - giá trị tương ứng với tên nhân viên - doanh số tổng



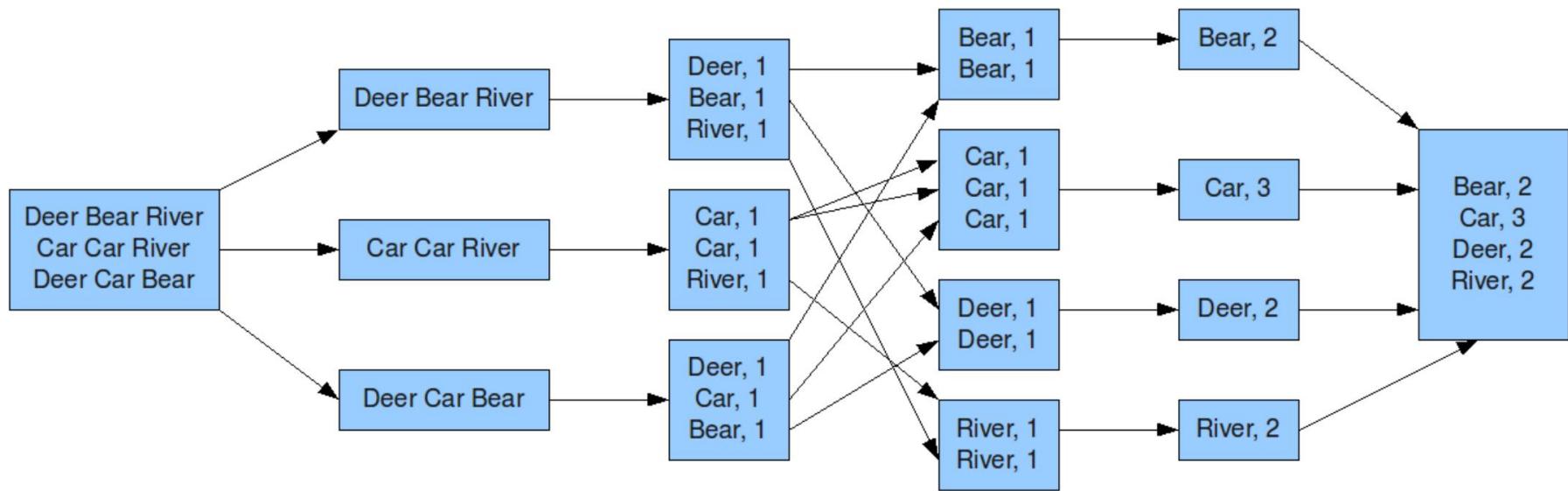
Luồng dữ liệu



Luồng Đánh Giá với bài toán Đếm từ

The overall MapReduce word count process

Input Splitting Mapping Shuffling Reducing Final result



Chương trình Thực tế Word Count (1)

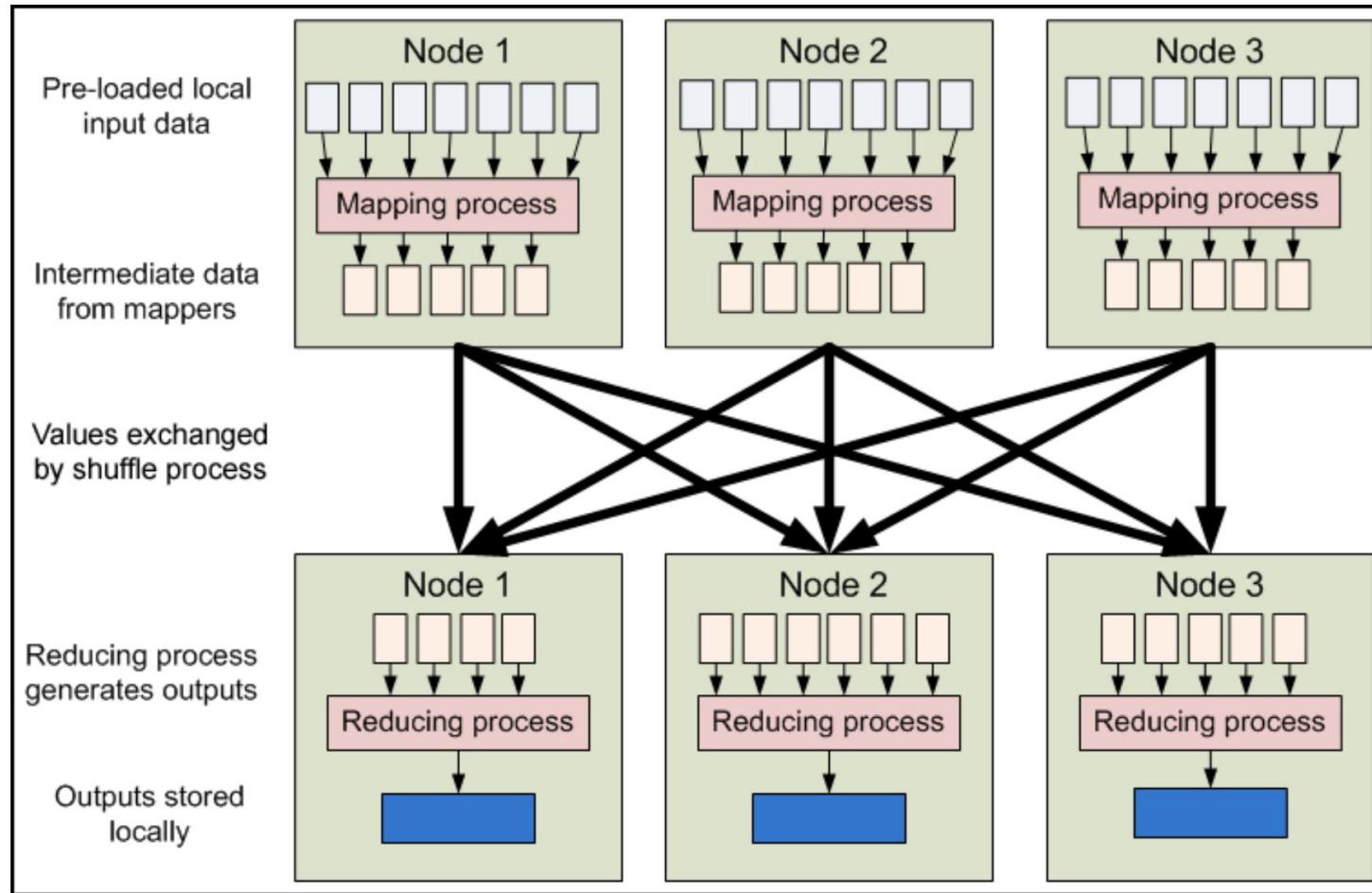
```
9 import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import org.apache.hadoop.util.GenericOptionsParser;
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     }
}
```



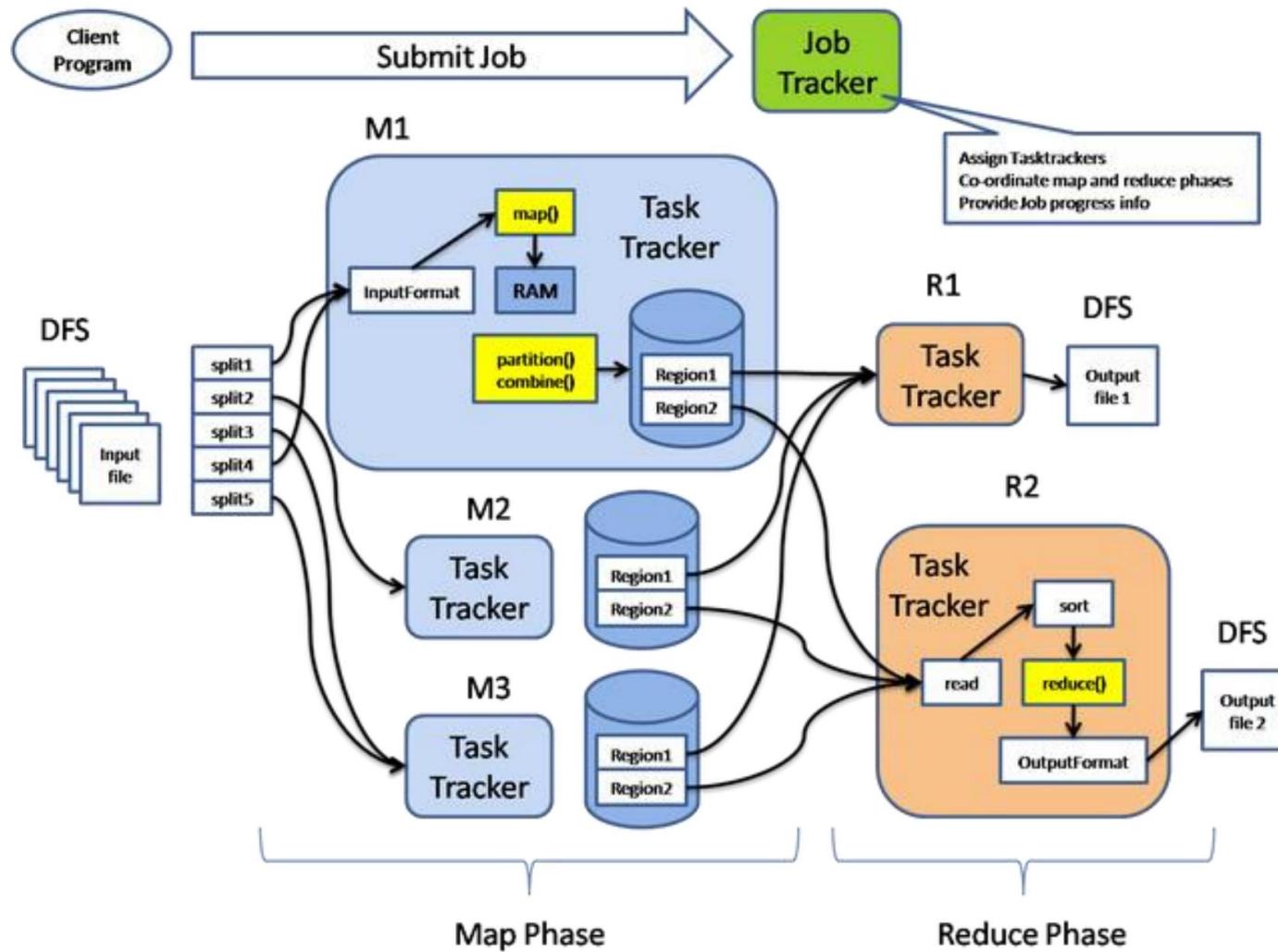
Chương trình Thực tế Word Count (2)

```
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, InterruptedException
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 phân tán môi trường



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, các 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' mà là 1 phần của hệ thống sinh thái Hadoop
 - Hầu hết là nguồn mở trên Apache

Lợn Apache

- Apache Pig cung cấp khả năng xử lý dữ liệu cao
 - Pig đặc biệt tốt cho các phép toán Join and Transformation
- Trình biên dịch Pig run trên máy khách
 - Biến đổi tập lệnh PigLatin thành các công việc của MapReduce
 - Thực hiện 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;
```

Tổng quan Apache

- Cũng là một lớp vật thể cao của MapReduce • Giảm thời gian phát triển • Cung cấp ngôn ngữ HiveQL: ngôn ngữ giống SQL
- Hive trình biên dịch trên máy khách
 - Chuyển tập lệnh HiveQL thành công việc MapReduce
 - Thực hiện 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

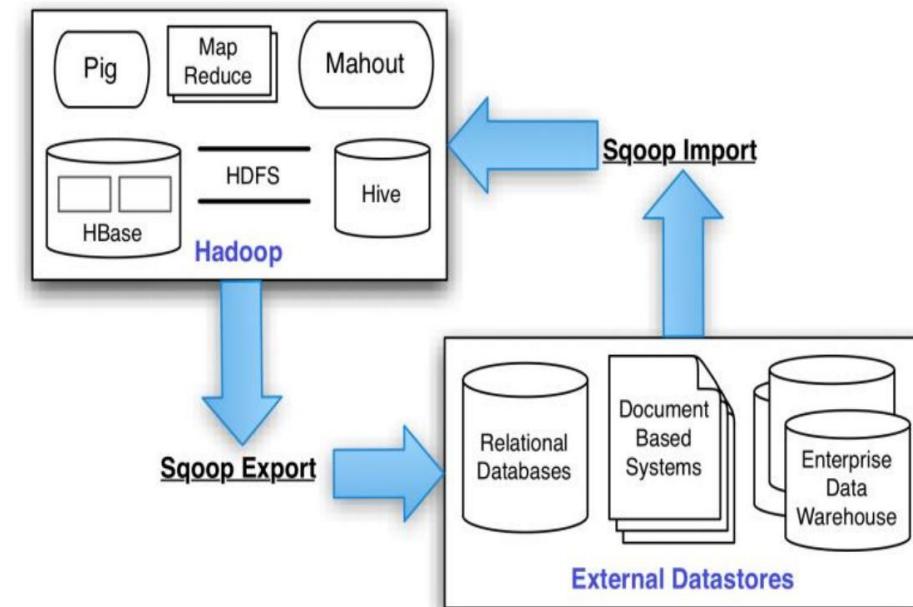


- HBase là một phân tán mở rộng cột cơ sở dữ liệu, lưu trữ dữ liệu trên HDFS
 - Được xem như hệ quản trị cơ sở dữ liệu của Hadoop
- Dữ liệu được tổ chức về mặt logic là các bảng, bao gồm rất nhiều dòng và cột
 - Bảng kích thước có thể tăng lên hàng Terabyte, Petabyte
 - Bảng có thể có hàng nghìn cột
- Có tính khả năng mở cao, đáp ứng băng thông ghi tốc độ dữ liệu 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 Truyền tải QTCSQL • Là
NoSQL : không có ngôn ngữ truy vấn cao như SQL • Phải sử dụng API để quét/ đặt/ lấy/ theo khóa dữ liệu

Apache Sqoop

- Sqoop là một công cụ cho phép trung chuyển dữ liệu theo khối từ Apache Hadoop và các cơ sở dữ liệu có cấu trúc như vậy
- Hệ thống cơ sở dữ liệu
- Hỗ trợ nhập tất cả các bảng, một bảng hoặc 1 phần của bảng vào HDFS
- Thông qua
 - Map only hoặc
 - Công việc MapReduce
- Kết quả là 1 thư mục trong HDFS chứa các phân tách văn bản tập tin theo phân tách ký tự (vd. , hoặc \t)
- Hỗ trợ xuất dữ liệu ngược

return from Hadoop ra external

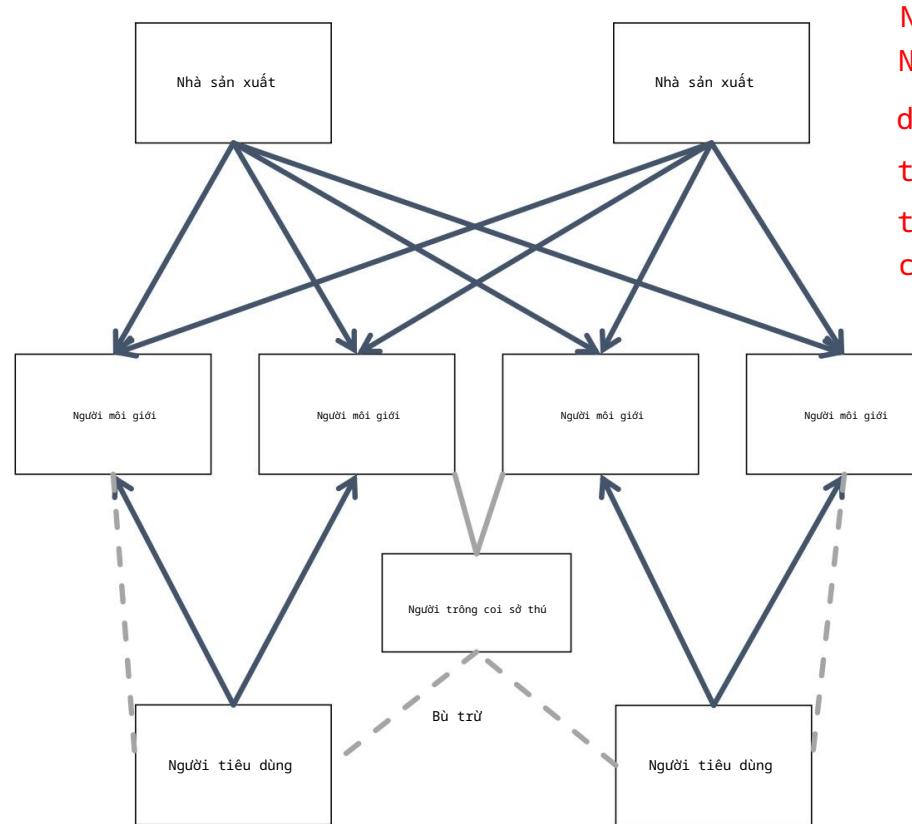


Apache Kafka

Nhà sản xuất

Kafka
Cụm

Người tiêu dùng



Nhà sản xuất không cần biết
Người tiêu
dùng Đảm bảo hoạt động và
tin cậy trong quá trình
trung chuyển dữ liệu giữa
các bên

Kafka cho phép phân tích các tham số mạch
thành phần vào dữ liệu luồng

Người Apache Oozie

- Oozie là một công việc lập lịch hệ thống để quản lý các công việc thực thi trên cụm Hadoop
- Luồng công việc của Oozie là sơ đồ có vòng hướng (Biểu đồ chu kỳ có hướng (DAG)) của các khối công việc
- Oozie hỗ trợ nhiều loại công việc khác nhau
 - 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 email nhận

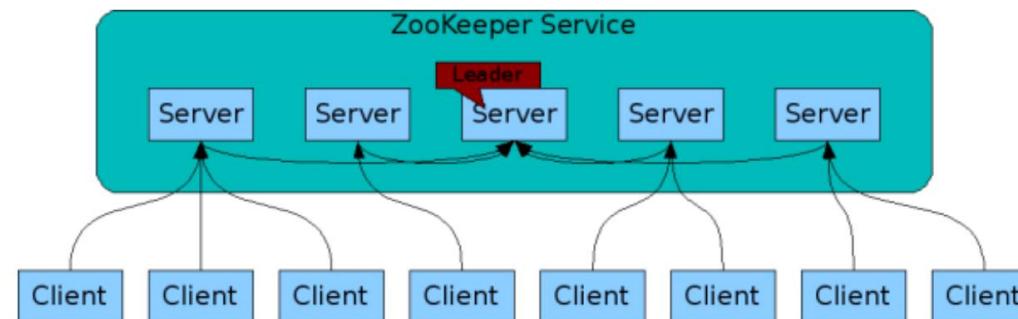


Người trông coi sở thú Apache

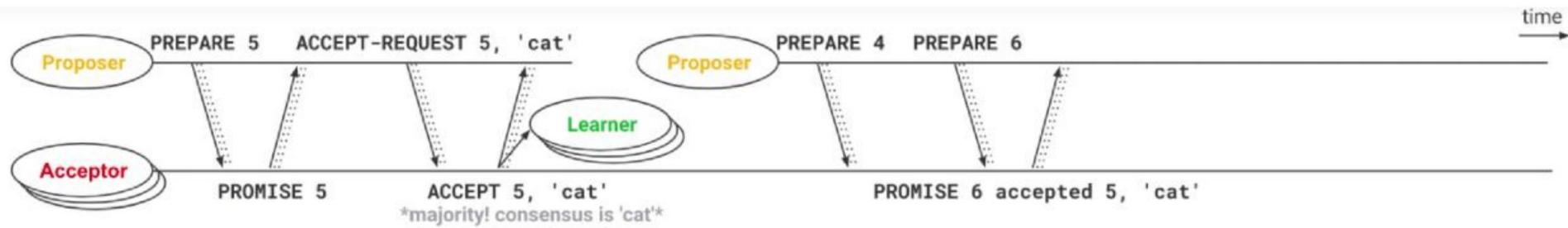
- Apache ZooKeeper là một dịch vụ cung cấp các chức năng phân phối hợp lý độ tin cậy cao
- Quản lý thành viên trong nhóm máy chủ
- Bầu cử lãnh đạo
- Quản lý cấu hình thông tin

Giám sát hệ thống trạng thái

- Đây là cốt lõi dịch vụ, tối quan trọng trong phân tán hệ thống



Thuật toán PAXOS



⇒ **Proposer** wants to propose a certain value:

It sends **PREPARE ID_p** to a majority (or all) of **Acceptors**.

ID_p must be unique, e.g. slotted timestamp in nanoseconds.

e.g. **Proposer** 1 chooses IDs 1, 3, 5...

Proposer 2 chooses IDs 2, 4, 6..., etc.

Timeout? retry with a new (higher) ID_p.

⇒ **Acceptor** receives a **PREPARE** message for ID_p:

Did it promise to ignore requests with this ID_p?

Yes -> then ignore

No -> Will promise to ignore any request lower than ID_p.

Has it ever accepted anything? (assume accepted ID=ID_a)

Yes -> Reply with **PROMISE ID_p accepted ID_a, value**.

No -> Reply with **PROMISE ID_p**.

★ If a majority of acceptors promise, no ID<ID_p can make it through.

⇒ **Proposer** gets majority of **PROMISE** messages for a specific ID_p: It sends **ACCEPT-REQUEST ID_p, VALUE** to a majority (or all) of **Acceptors**. Has it got any already accepted value from promises?
Yes -> It picks the value with the highest ID_a that it got.
No -> It picks any value it wants.

⇒ **Acceptor** receives an **ACCEPT-REQUEST** message for ID_p, value:

Did it promise to ignore requests with this ID_p?

Yes -> then ignore

No -> Reply with **ACCEPT ID_p, value**. Also send it to all **Learners**.

★ If a majority of acceptors accept ID_p, value, consensus is reached. Consensus is and will always be on value (not necessarily ID_p).

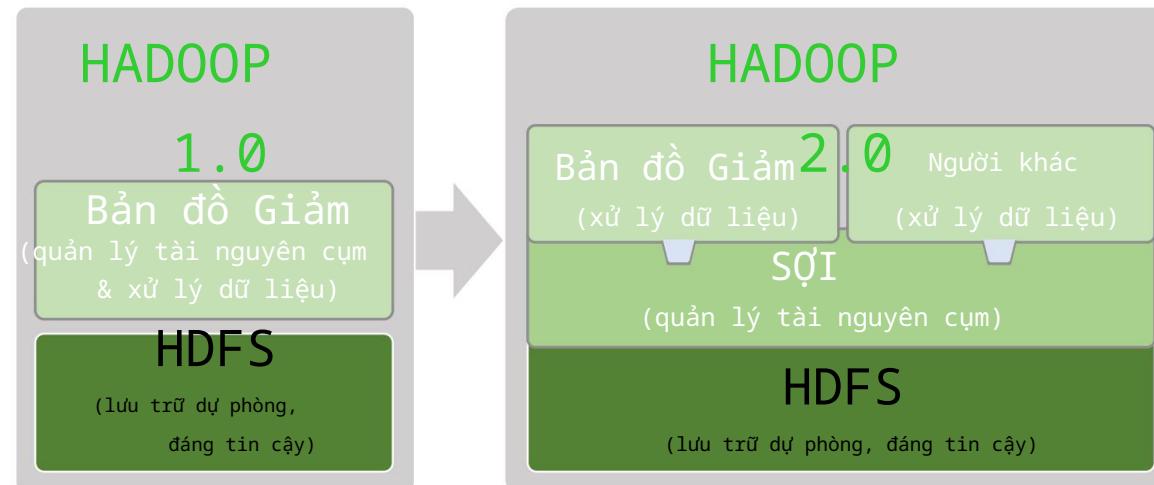
⇒ **Proposer** or **Learner** get **ACCEPT** messages for ID_p, value:

★ If a proposer/learner gets majority of accept for a specific ID_p, they know that consensus has been reached on value (not ID_p).

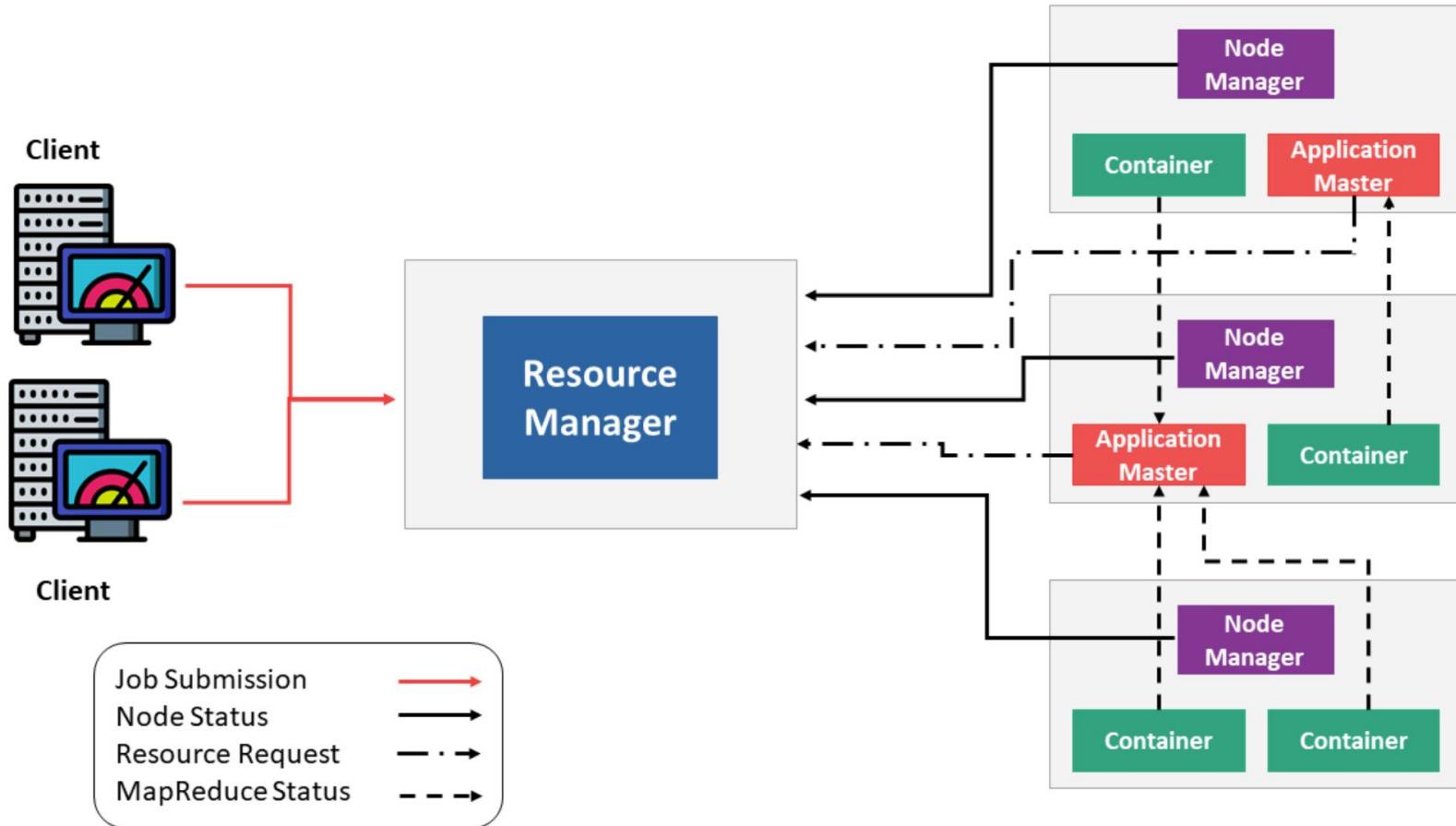
https://www.youtube.com/watch?v=d7nAGI_NZPk

YARN - Một nhà đàm phán tài nguyên khác

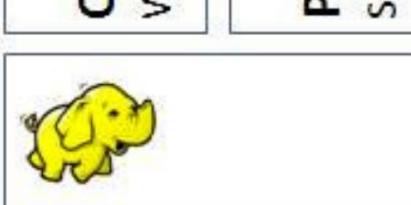
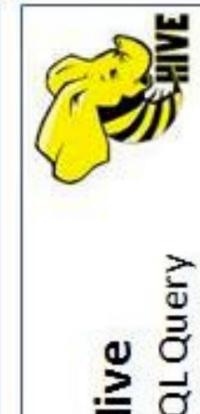
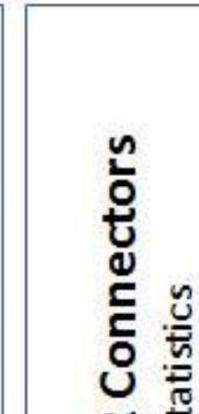
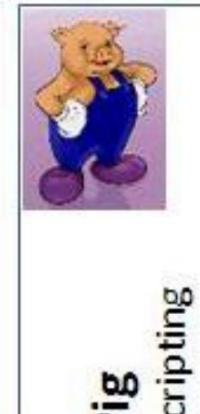
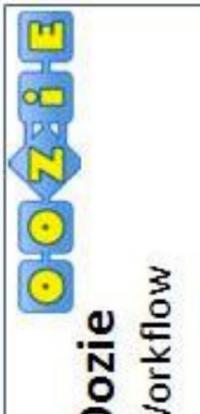
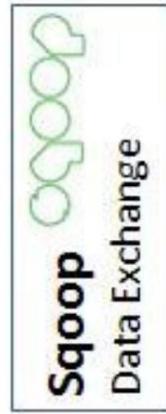
- Các nút có tài nguyên là bộ nhớ và lõi CPU
- YARN đóng vai trò cung cấp tài nguyên phù hợp cho các ứng dụng khi có yêu cầu
- YARN được đưa ra từ Hadoop 2.0
 - Cho phép MapReduce và non MapReduce cùng chạy trên 1 cụm Hadoop
 - Với công việc MapReduce, vai trò của trình theo dõi công việc được thực hiện bởi trình theo dõi ứng dụng



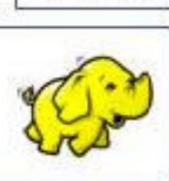
Ví dụ về cấp phát trên YARN



Bức tranh tổng hợp hệ sinh thái Hadoop



YARN Map Reduce v2
Distributed Processing Framework



Nền tảng quản lý dữ liệu lớn

AEROSPIKE



Clustrix

Couchbase

amazon
DynamoDB

APACHE
HBASE

IBM
Cloudant
The Data Storage Cloud Layer

MarkLogic

memsql
Speed. Scale. Simplicity.

mongoDB

NUODB

ORACLE
NOSQL
DATABASE

riak

splice
MACHINE

TRANSATTICE

VOLTDB

