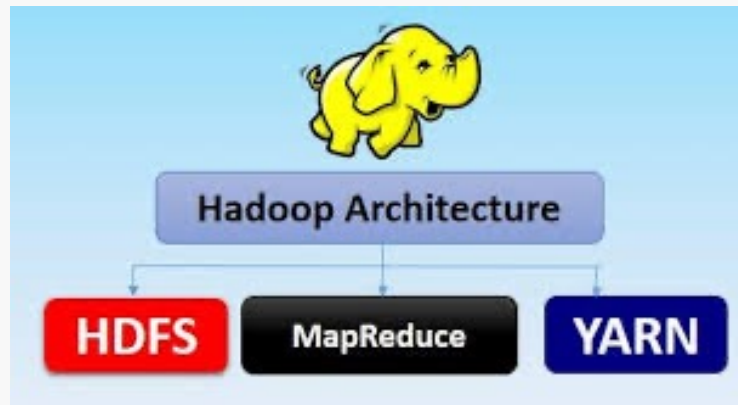




DỮ LIỆU LỚN VÀ ỨNG DỤNG



CHƯƠNG 3: HADOOP YARN & MAPREDUCE



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Mô hình xử lý truyền thống

Bài toán: có một nhật ký thời tiết chứa nhiệt độ trung bình hàng ngày của các năm từ 2000 đến 2015.

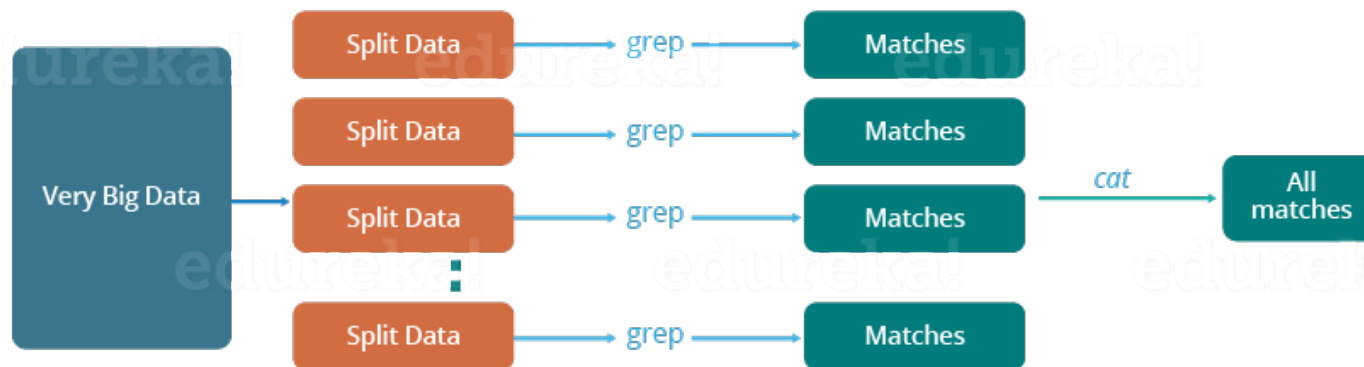
Nhiệm vụ là tính ngày có nhiệt độ cao nhất trong mỗi năm.

Giải quyết:

- Theo cách truyền thống, sẽ chia dữ liệu thành các phần hoặc khối nhỏ hơn và lưu trữ trong các máy khác nhau.
- Sau đó, sẽ tìm nhiệt độ cao nhất ở từng bộ phận được lưu trữ trong máy tương ứng. Cuối cùng, là kết hợp các kết quả nhận được từ mỗi máy để có kết quả cuối cùng.



Mô hình xử lý truyền thống



The Traditional Way



Mô hình xử lý truyền thống

Những thách thức liên quan đến cách tiếp cận truyền thống này:

- **Vấn đề về độ tin cậy:** Điều gì sẽ xảy ra nếu, bất kỳ máy nào đang làm việc với một phần dữ liệu bị lỗi? Việc quản lý chuyển đổi dự phòng này trở thành một thách thức.
- **Vấn đề chia đều:** Làm thế nào để chia dữ liệu thành các phần nhỏ hơn sao cho mỗi máy nhận được một phần dữ liệu đồng đều để làm việc.
- **Tổng hợp kết quả:** Cần có cơ chế tổng hợp kết quả do từng máy tạo ra để tạo ra đầu ra cuối cùng.



Giải thuật MapReduce

- Là giải thuật xử lý dữ liệu phân tán do Google phát triển 2004, xuất phát trên mô hình lập trình hàm.
- Sử dụng xử lý dữ liệu lớn song song một cách tin cậy và hiệu quả trên môi trường cluster
- Chia input thành các task nhỏ hơn và dễ quản lý hơn để thực hiện song song

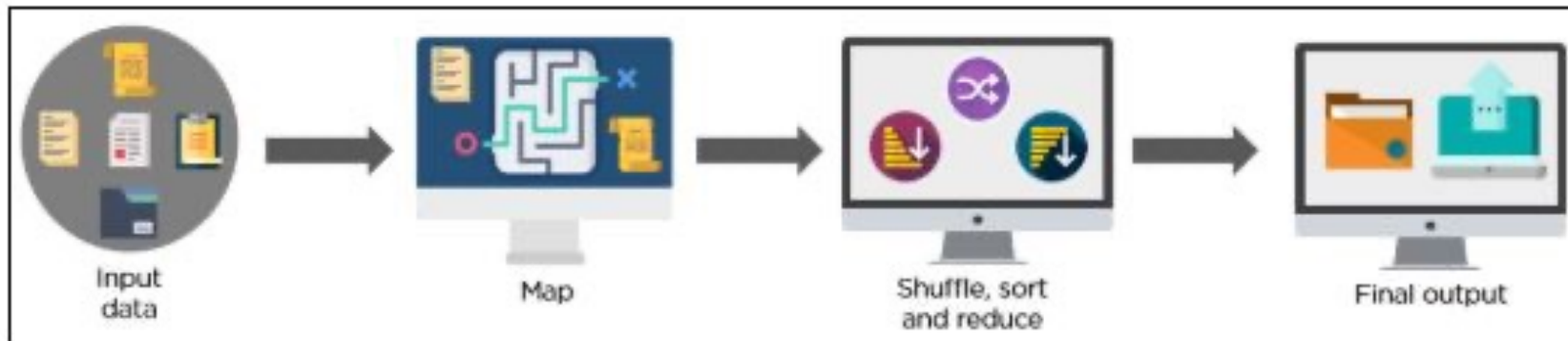
MR cung cấp:

- Tự động phân bổ và song song hóa.
- Chịu lỗi.
- Lập lịch I/O
- Giám sát trạng thái

<https://research.google.com/archive/mapreduce-osdi04-slides/index.html>



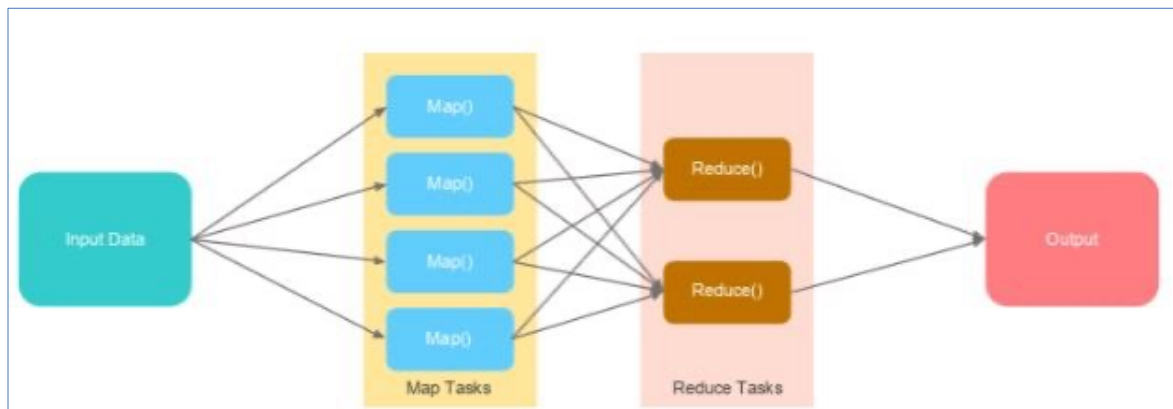
Mô hình MapReduce



Nó có hai thành phần hoặc giai đoạn chính là: **pha Map** và **pha Reduce**



Mô hình MapReduce



- **input data:** Hadoop chấp nhận dữ liệu ở nhiều định dạng khác nhau và lưu trữ nó trong HDFS . Dữ liệu đầu vào này được thực hiện bởi nhiều tác vụ bản đồ.
- **Map Tasks:** Map đọc dữ liệu, xử lý và tạo các cặp khóa-giá trị. Số lượng Map tasks phụ thuộc vào tệp đầu vào và định dạng của nó.
- **Reduce Tasks:** Nó xáo trộn, sắp xếp và tổng hợp các cặp khóa-giá trị trung gian (bộ giá trị) thành một tập hợp các bộ giá trị nhỏ hơn
- **Output:** Tập hợp các bộ giá trị nhỏ hơn là đầu ra cuối cùng và được lưu trữ trong HDFS.



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Cách thức MapReduce hoạt động

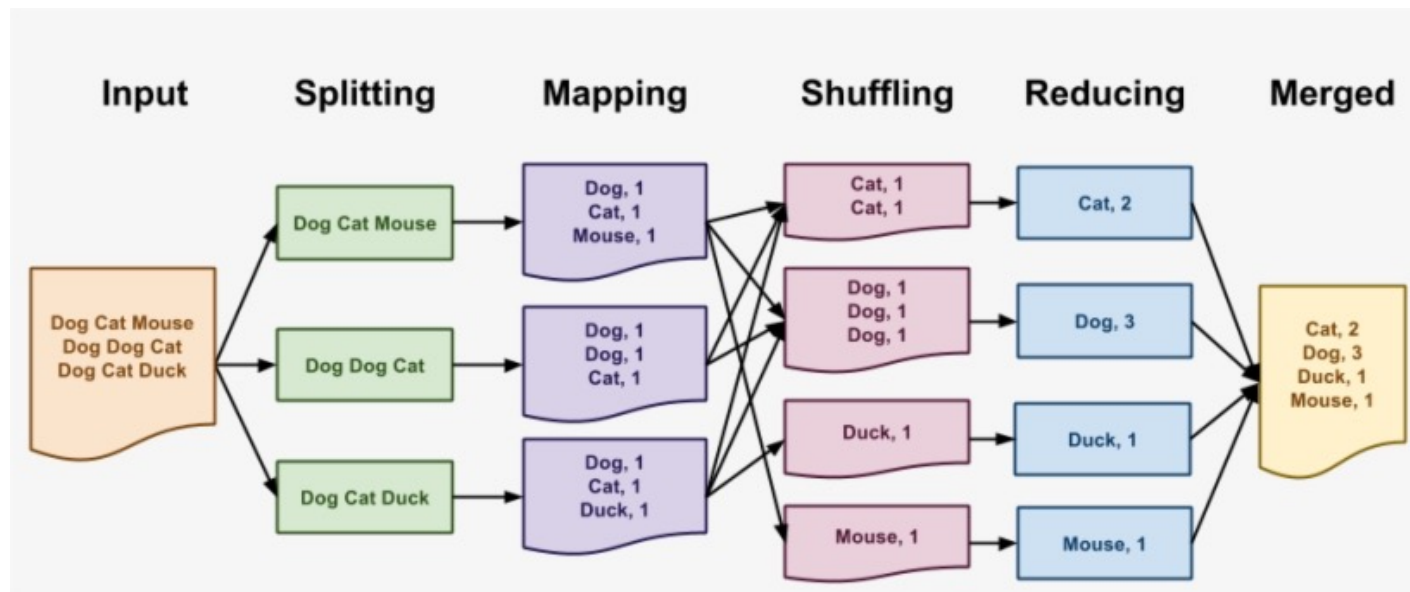
Ví dụ: có một tệp văn bản được gọi là example.txt có nội dung như sau:

Dog, Cat, Mouse, Dog, Dog, Cat, Dog, Cat and Duck

Nhiệm vụ: chúng ta phải thực hiện đếm từ trên sample.txt bằng MapReduce.

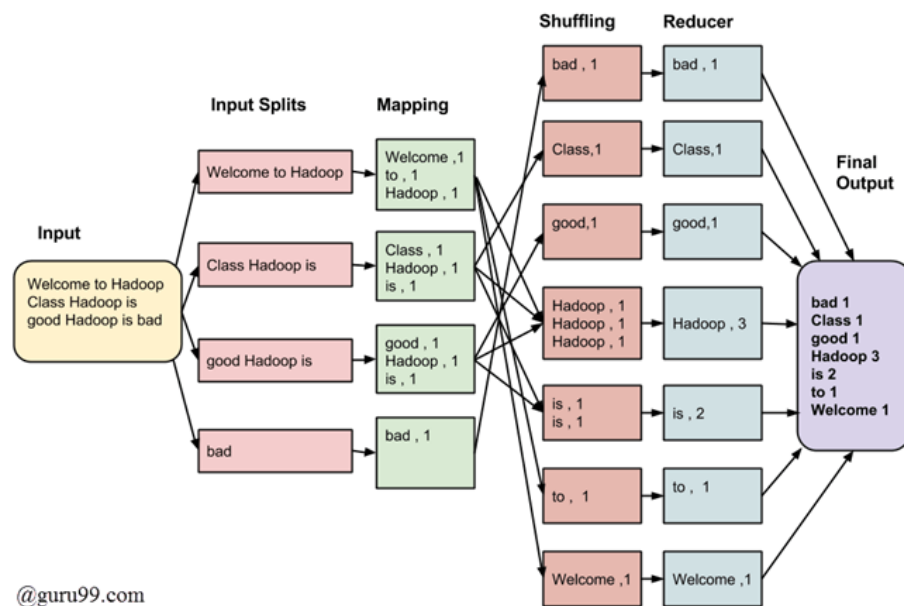


Cách thức MapReduce hoạt động





Cách thức MapReduce hoạt động





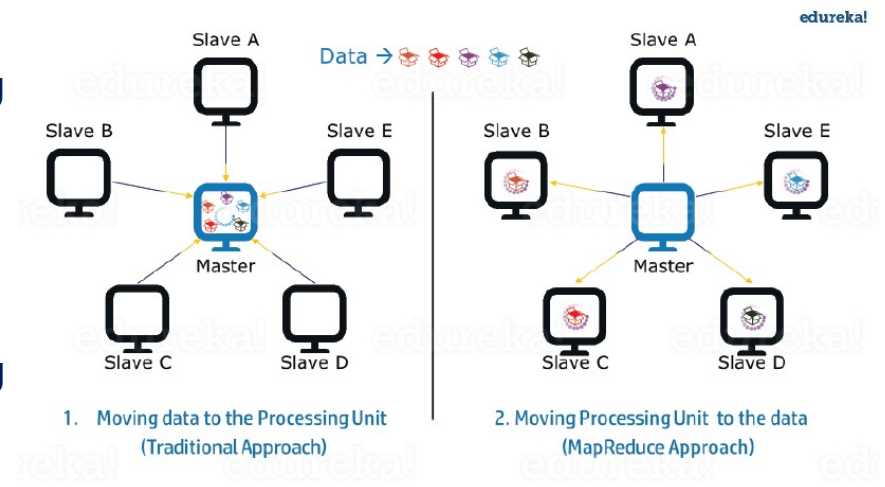
Hai ưu điểm lớn nhất của MapReduce

1. Xử lý song song:

Vì dữ liệu được xử lý bởi nhiều máy thay vì một máy song song, thời gian thực hiện để xử lý dữ liệu sẽ giảm đi rất nhiều

2. Vị trí dữ liệu:

Thay vì di chuyển dữ liệu đến đơn vị xử lý, chúng tôi đang chuyển đơn vị xử lý sang dữ liệu trong MapReduce Framework.





DỮ LIỆU LỚN VÀ ỨNG DỤNG

BÀI THỰC HÀNH

HƯỚNG DẪN SỬ DỤNG MAPREDUCE



Nhắc lại lý thuyết

MapReduce là một programming framework cho phép chúng ta thực hiện xử lý phân tán và song song trên các tập dữ liệu lớn trong môi trường phân tán.

- MapReduce gồm 2 nhiệm vụ tách biệt là Map và Reduce.
 - Giống như cái tên của nó giai đoạn Reduce sẽ diễn ra sau khi giai đoạn Mapper hoàn thành.
 - Giai đoạn Map() bao gồm lọc (filter) và phân loại (sort) trên dữ liệu trong khi giai đoạn Reduce() thực hiện quá trình tổng hợp dữ liệu.



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Bài toán

Thực hành ví dụ WordCount sử dụng MapReduce

Giả sử có một tệp văn bản được gọi là example.txt có nội dung như sau :

~~— Dog, Cat, Mouse, Dog, Dog, Cat, Dog, Cat and Duck~~

Yêu cầu: thực hiện đếm từ trên sample.txt bằng MapReduce.

Lưu ý: Sinh viên sẽ tự tạo bộ dữ liệu bao gồm mã sinh viên của mình và các bạn trong lớp. Trong đó MSV của chính mình sẽ xuất hiện nhiều nhất (5 lần).



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình

Về cơ bản, toàn bộ chương trình MapReduce có thể được chia thành ba phần:

- Mapper Code
- Reducer Code
- Driver Code

Toàn bộ mã nguồn kèm giải thích xem tại: <https://hadoop.apache.org/docs/hadoop-mapreduce>

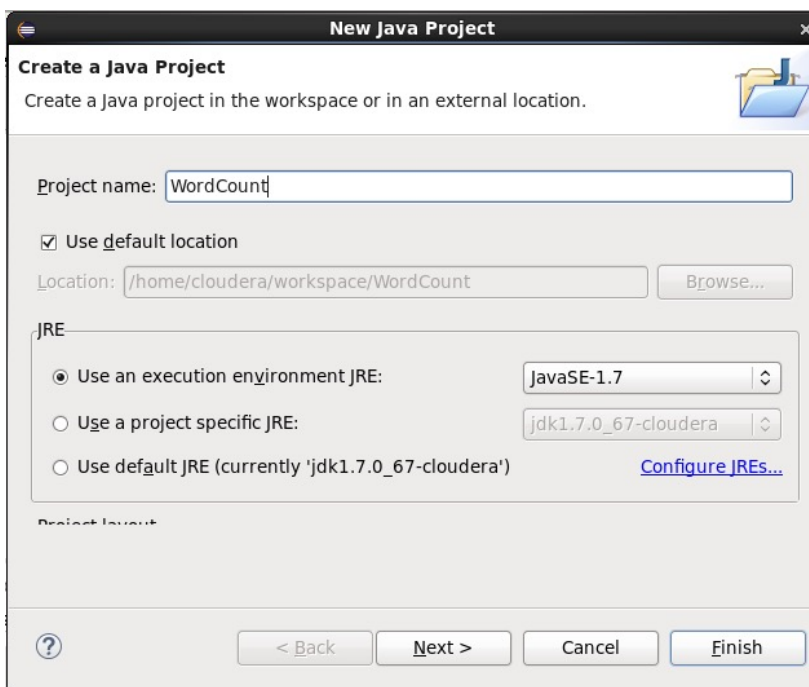


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

Tạo Project

- Chuột phải vào project sau đó chọn New Java Project.
- Đặt tên cho project là WordCount.
- Sau đó ấn next để tiếp tục.

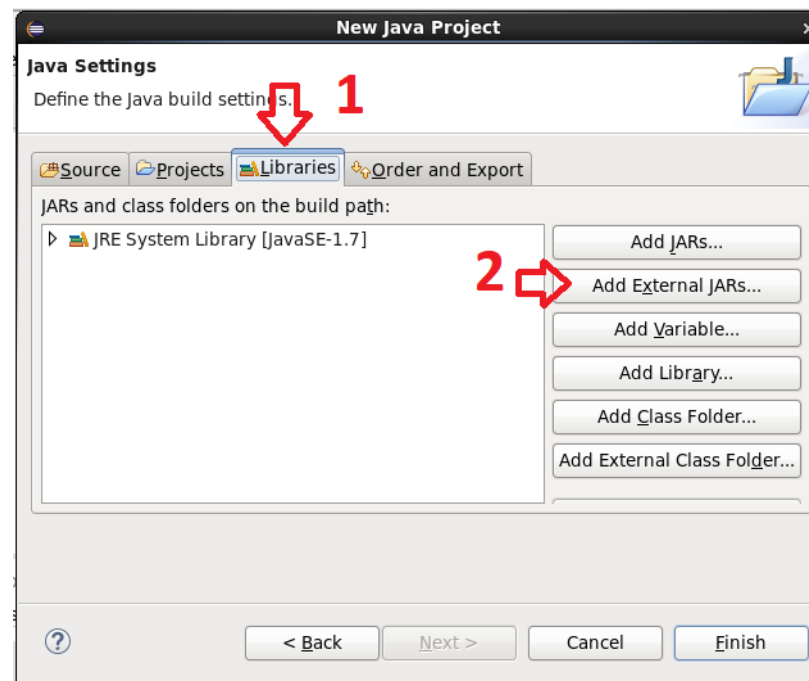




DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

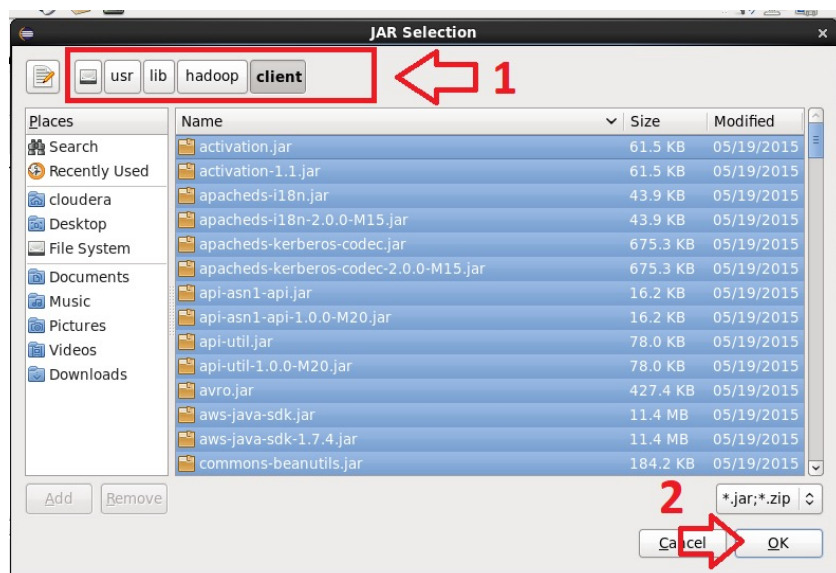
- Add Libraries cho Project
- Thực hiện 2 lần, cụ thể xem các bước tiếp theo



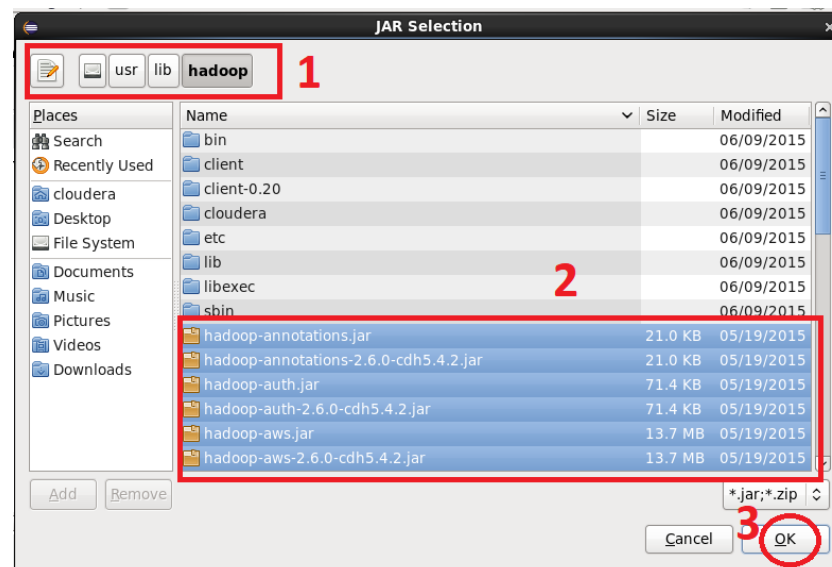


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế



Lần 1- Add tất cả file .jar trong client

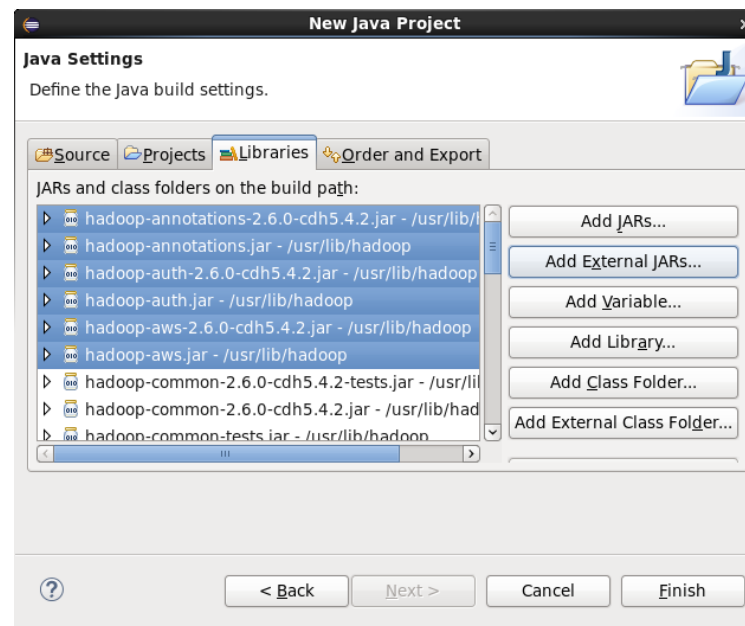


Lần 2- Add tất cả file .jar trong hadoop



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế



Nhấn Finish để hoàn tất quá trình tạo project



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

- Tạo Class mới đặt tên WordCount
- Copy source code của chương trình tại liên kết sau:
<https://hadoop.apache.org/docs/hadoop-mapreduce>
- Lưu lại chương trình

```
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  training
    WordCount
      src
        (default package)
          WordCount.java
      JRE System Library [JavaSE-1.7]
      Referenced Libraries

*WordCount.java
1 import java.io.IOException;
2 import java.util.StringTokenizer;
3
4 import org.apache.hadoop.conf.Configuration;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Reducer;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13
14 public class WordCount {
15
16     public static class TokenizerMapper
17         extends Mapper<Object, Text, Text, IntWritable>{
18
19         private final static IntWritable one = new IntWritable(1);
20         private Text word = new Text();
21
22         public void map(Object key, Text value, Context context
23             throws IOException, InterruptedException {
```

Problems @ Javadoc Declaration Console

No consoles to display at this time.

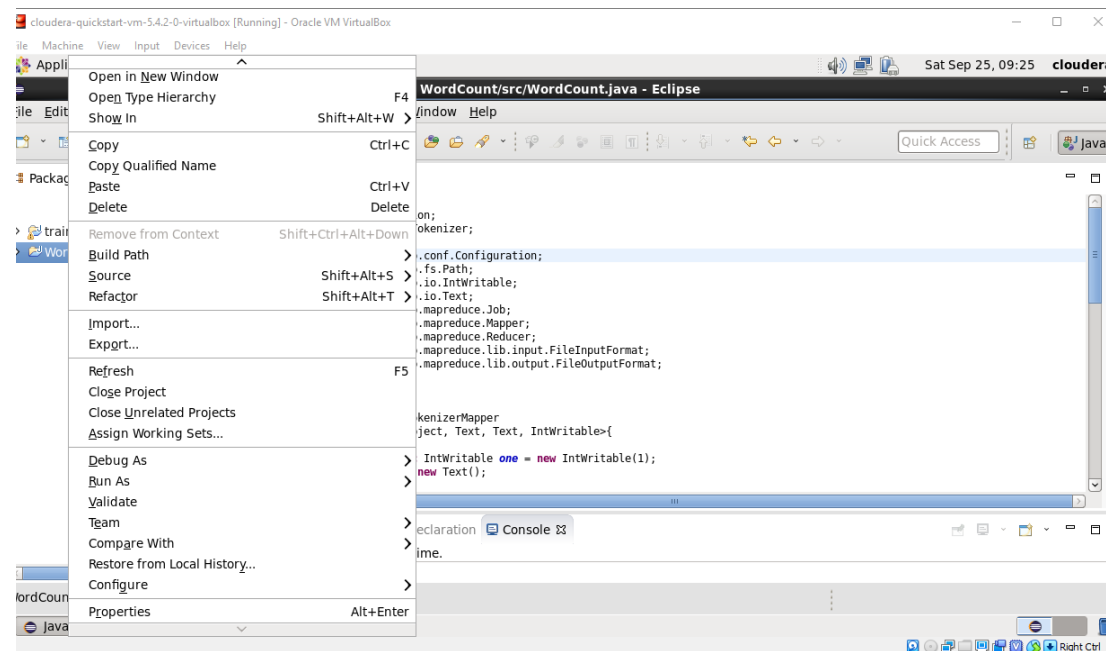


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

Export chương trình

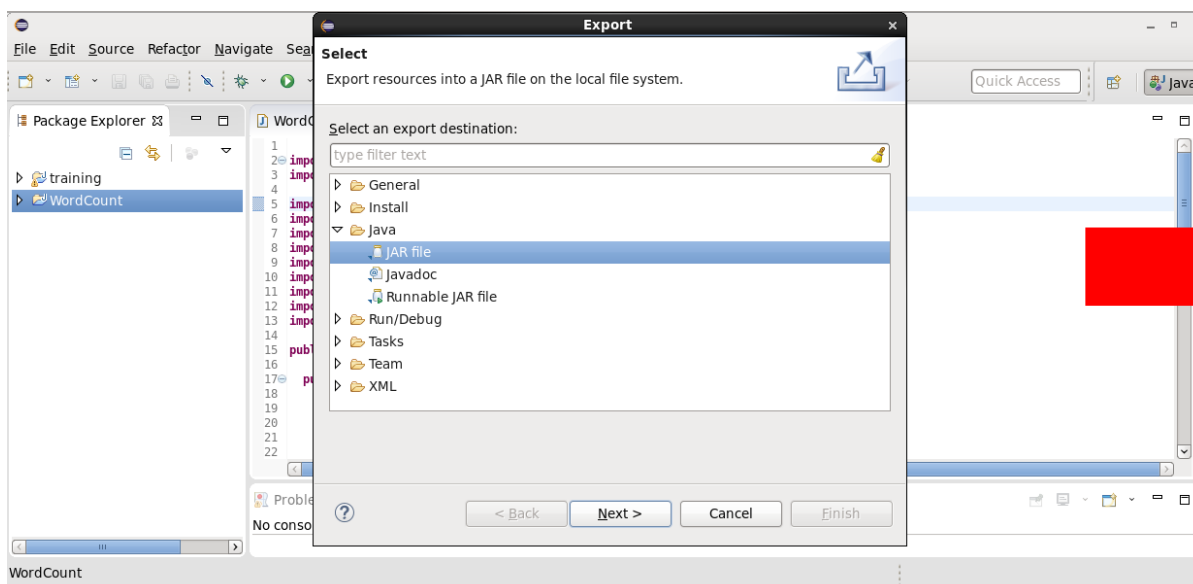
- Chuột phải vào project sau đó chọn **Export**



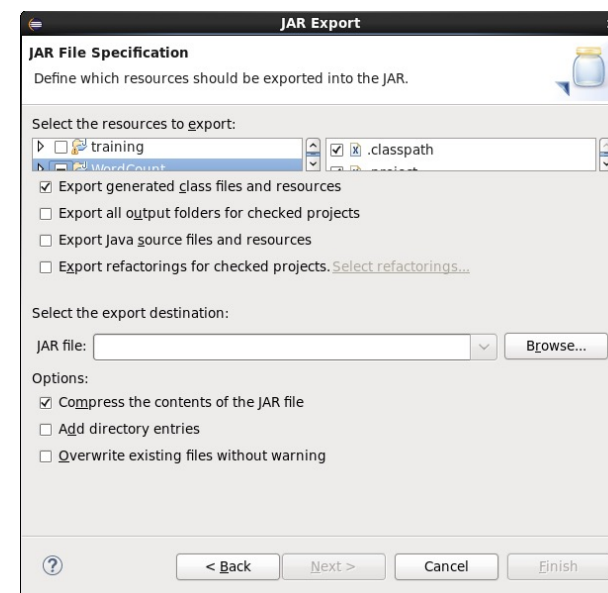


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế



1. Chọn JAR file

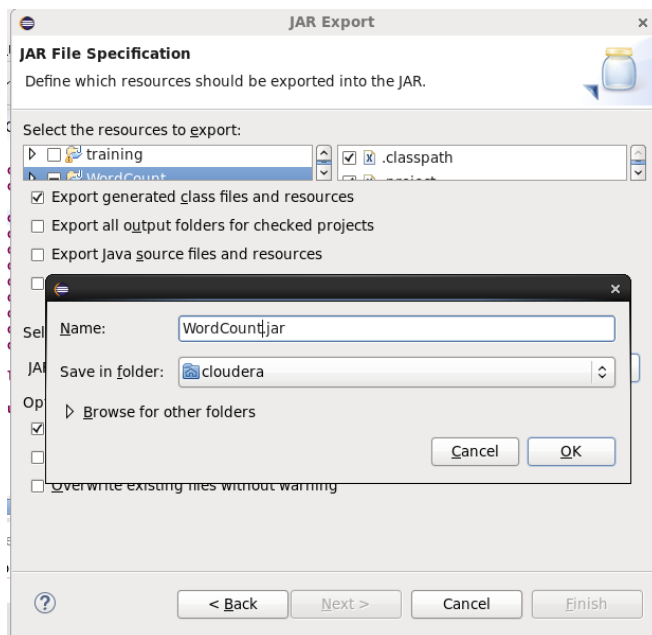


2. Chọn đường dẫn (Browse)

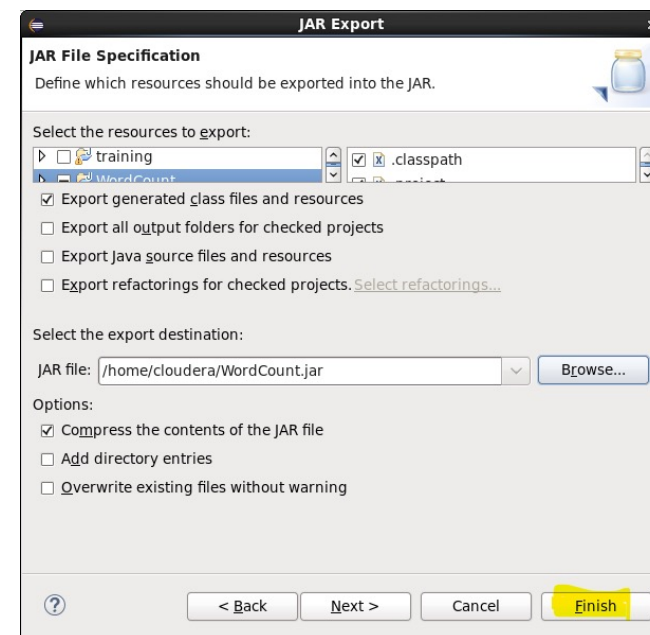
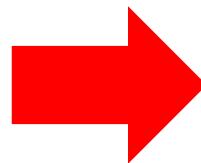


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế



1. Đặt tên cho file là WordCount.jar

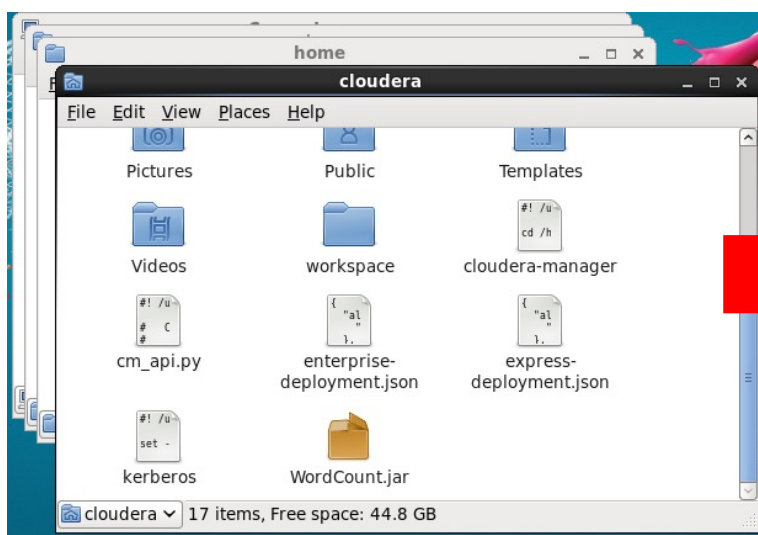


2. Nhấn Finish để hoàn thành

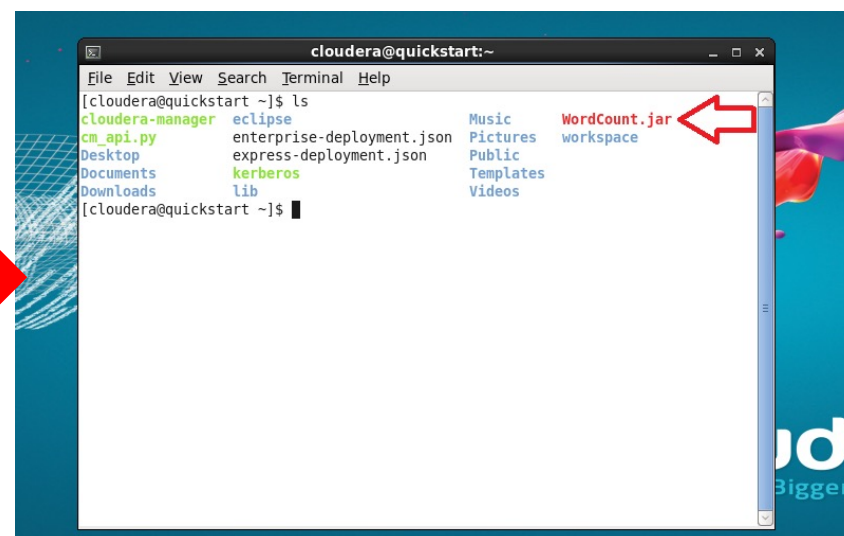


DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế



Kiểm tra file WordCount.jar đã export
(cách 1)



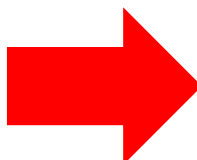
Kiểm tra file WordCount.jar đã export
(cách 2)



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ ls  
cloudera-manager eclipse Music WordCount.jar  
cm_api.py enterprise-deployment.json Pictures workspace  
Desktop express-deployment.json Public  
Documents kerberos Templates  
Downloads lib Videos  
[cloudera@quickstart ~]$ cat >/home/cloudera/Processfile1.txt  
Ha Noi  
Ha Noi  
Ho Chi Minh  
Da Nang  
Da Nang  
Da Nang  
Can Tho  
Can Tho  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ha Noi  
^Z  
[1]+ Stopped cat > /home/cloudera/Processfile1.txt  
[cloudera@quickstart ~]$
```



```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
Da Nang  
Da Nang  
Can Tho  
Can Tho  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ha Noi  
^Z  
[1]+ Stopped cat > /home/cloudera/Processfile1.txt  
[cloudera@quickstart ~]$ cat /home/cloudera/Processfile1.txt  
Ha Noi  
Ha Noi  
Ho Chi Minh  
Da Nang  
Da Nang  
Da Nang  
Can Tho  
Can Tho  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ha Noi  
[cloudera@quickstart ~]$
```

1. Tạo và chèn nội dung cho file
Processfile1.txt

2. Kiểm tra file Processfile1.txt vừa tạo



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputfolder1  
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Processfile1.txt /inputfolder1/  
[cloudera@quickstart ~]$
```

1. Put file vừa tạo lên HDFS

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputfolder1  
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Processfile1.txt /inputfolder1/  
[cloudera@quickstart ~]$ hdfs dfs -cat /inputfolder1/Processfile1.txt  
Ha Noi  
Ha Noi  
Ho Chi Minh  
Da Nang  
Da Nang  
Da Nang  
Can Tho  
Can Tho  
Ho Chi Minh  
Ho Chi Minh  
Ho Chi Minh  
Ha Noi  
[cloudera@quickstart ~]$
```

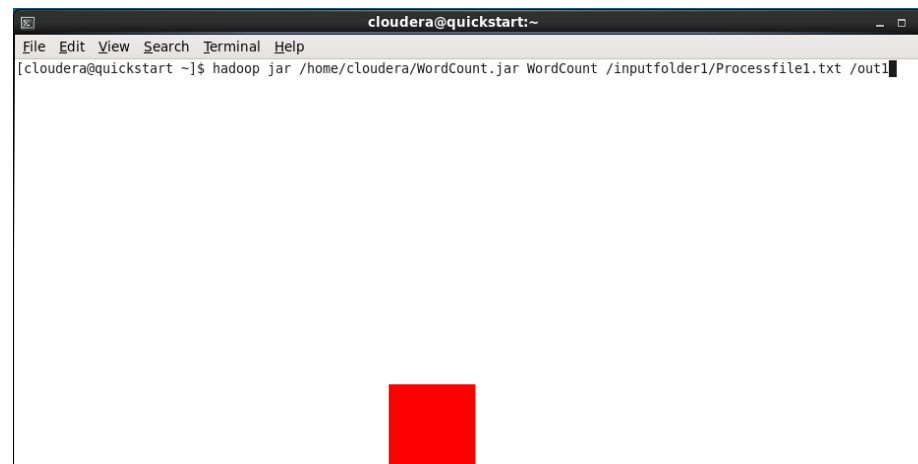
2. Kiểm tra lại nội dung của file trên HDFS



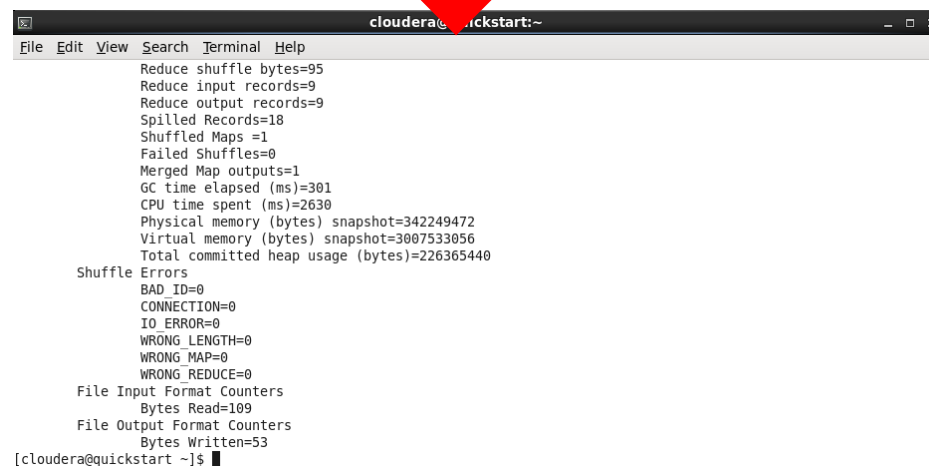
DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình

- Chạy chương trình đếm từ với file Processfile1.txt
- Cần đợi một vài giây để câu lệnh chạy hoàn tất.



```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder1/Processfile1.txt /out1
```



```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
Reduce shuffle bytes=95  
Reduce input records=9  
Reduce output records=9  
Spilled Records=18  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=301  
CPU time spent (ms)=2630  
Physical memory (bytes) snapshot=342249472  
Virtual memory (bytes) snapshot=3007533056  
Total committed heap usage (bytes)=226365440  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=109  
File Output Format Counters  
Bytes Written=53  
[cloudera@quickstart ~]$
```



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Viết chương trình thực tế

Kiểm tra kết quả khi chương trình chạy hoàn tất.

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ hdfs dfs -ls /out1  
Found 2 items  
-rw-r--r-- 1 cloudera supergroup 0 2021-09-25 10:26 /out1/_SUCCESS  
-rw-r--r-- 1 cloudera supergroup 53 2021-09-25 10:26 /out1/part-r-0000  
[cloudera@quickstart ~]$ hdfs dfs -cat /out1/part-r-0000  
bash: hdfs: command not found  
[cloudera@quickstart ~]$ hdfs dfs -cat /out1/part-r-0000  
Can 2  
Chi 4  
Da 3  
Ha 3  
Ho 4  
Minh 4  
Nang 3  
Noi 3  
Tho 2  
[cloudera@quickstart ~]$
```

Kết quả của chương trình



DỮ LIỆU LỚN VÀ ỨNG DỤNG

Ví dụ 2

Dưới đây là dữ liệu liên quan đến mức tiêu thụ điện của một tổ chức. Nó chứa mức tiêu thụ điện hàng tháng và mức trung bình hàng năm trong các năm khác nhau. Dữ liệu trên được lưu dưới dạng tệp **sample.txt** và được cung cấp dưới dạng đầu vào. Tệp đầu vào trông như hình dưới đây.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45