# PHẦN 1. JAVA CORE

## 1. Tài liệu [1 Tuần ]

OOP core:
http://www.w3resource.com/java-tutorial/java-object-oriented-programming.php

Concurrency:
http://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/
(*cái này nó viết bằng syntax của java 8*)

Collection(s)
(*Overview*) http://cs.lmu.edu/~ray/notes/collections/
(*compare*) http://www.codejava.net/java-core/collections/java-collections-framework-summary-table
(*performance*) http://infotechgems.blogspot.com/2011/11/java-collections-performance-time.html

I/O:
(*phần rút gọn*) http://www.studytonight.com/java/java-io-stream.php
(*phần đầy đủ*) http://www.ntu.edu.sg/home/ehchua/programming/java/j5b_io.html
(phần đầy đủ khá phức tạp, và nhiều phần, em đọc lướt thôi nhé)

Exception
http://www.journaldev.com/1696/java-exception-handling-tutorial-with-examples-and-best-practices

SQL & JDBC
Sử dụng được các lệnh Insert, select, update, delete. *(Tài liệu tìm trên mạng)*
JDBC: http://www.codejava.net/java-se/jdbc/jdbc-tutorial-sql-insert-select-update-and-delete-examples

## 2. Test:

(Điểm đánh giá từ 0-10)

1. Sử dụng Set để tìm tập giao và tập hợp giữa 2 tập hợp (tự tạo ra 2 tập hợp, mỗi tập hợp 200.000 phần tử không trùng nhau và các phần tử là một số Integer ngẫu nhiên, 2 tập hợp phải có phần tử trùng nhau) trong khoảng thời gian ngắn nhất.
2. Cho một văn bản bất kỳ, đặt ở file input.txt. Hãy đọc file và đếm số lần xuất hiện của từng từ (mỗi từ phân biệt nhau bằng dấu cách) rồi ghi vào file output.txt.
3. Lập trình đếm từ đa luồng cho bài 2 với dữ liệu vào là một folder chứa nhiều file text. Hãy xử lý song song các file và tìm top 10 từ xuất hiện nhiều nhất, và top 10 từ xuất hiện ít nhất của toàn bộ dữ liệu có trong folder. Lưu ý, chỉ được chạy tối đa 6 luồng cùng lúc.
4. Sinh ngẫu nhiên 30.000 điểm trên mặt phẳng tọa độ 2 chiều số nguyên. Sao cho 8.000 điểm đầu tiên có cách điểm A(800, 800) một độ dài không quá 400 đơn vị, 10.000 điểm tiếp theo cách điểm B(4000,800) không quá 500 đơn vị, và 12.000 điểm cuối cùng cách điểm C(2400, 2400) không quá 600 đơn vị. Trộn ngẫu nhiên 30.000 điểm này, sau đó ghi ra file output4.txt

# PHẦN 2. TOOLs

## 1. Tài liệu

Làm quen với Linux (shell, file system, permission…):
https://help.ubuntu.com/community/LinuxFilesystemTreeOverview
http://ryanstutorials.net/linuxtutorial/permissions.php
https://www-xray.ast.cam.ac.uk/~jss/lecture/computing/notes/out/commands_basic/

Mã hóa, bảo mật với khóa công khai (SSH)
https://vi.wikipedia.org/wiki/M%E1%BA%ADt_m%C3%A3_h%C3%B3a_kh%C3%B3a_c%C3%B4ng_khai

Mạng và các giao thức cơ bản (TCP/IP, port, SSH, http…)
https://vi.wikipedia.org/wiki/TCP/IP
http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177

Sử dụng VPN (openVPN), IDE (eclipse, intellij), Maven, git
*Tài liệu tìm trên mạng*

## 2. Test: [1 Tuần]

(Điểm đánh giá từ 0-10)

1. Tạo maven project, sau đó add thêm 2 thư viện logback, và json để parse một dữ liệu json từ url http://news.admicro.vn:10002/api/realtime?domain=kenh14.vn.
Tạo một thread lấy dữ liệu từ url mỗi 1s một lần, tách lấy số user và lưu vào biến tĩnh S.
Tạo một thread khác quét biến S mỗi 2s một lần, và ghi lại sử dụng logback theo quy tắc sau:

- Lần đầu sẽ ghi giá trị biến S với level info
- Các lần sau nếu biến S lớn hơn 1.5% so với giá trị của biến S vào thời điểm trước đó thì ghi lại giá trị biến S với level Info
- Nếu quá lâu (12s) mà không ghi data lần nào thì sẽ buộc ghi giá trị biến S với level Debug

| 0s | 2s | 4s | 6s | 8s | 10s | 12s | 14s | 16s | 18s | 20s | 22s |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1032 | 1022 | 1092 | 1091 | 1082 | 1080 | 1075 | 1072 | 1060 | 1059 | 1075 | 1076 |
| Info | | Info | | | | | | Debug | | Info | |

2.
Tạo maven project với project encode là utf-8, sử dụng thêm thư viện jsoup, sau đó build toàn bộ project thành một file jar và 1 thư mục dependency chứa tất cả các thư viện đi kèm (sử dụng plug-in maven-dependency-plugin của maven). Sử dụng thư viện jsoup lấy nội dung của url "http://dantri.com.vn" và ghi nội dung ra một file với tên file là thời gian hiện tại.
Viết 1 file shell số 1 để call file jar vừa build
Sử dụng shell script của linux để chạy file shell thứ 1 mỗi 5 giây một lần.

3.
Cài đặt redis(https://redis.io) trên máy local (thiết lập mật khẩu là abc@123)
**Viết chương trình ghi vào redis** 1 hashmap tên là test với key từ 1 đến 10k value là bình phương của key.

**Tạo một project viết một restful webservice** (có thể sử dụng sparkjava) trả lại giá trị bình phương của số đầu vào vào từ param trên url của request (Ví dụ: http://localhost:8080/prime?n=10000 sẽ trả lại 10000000000).

4. Viết chương trình benmark service vừa tạo (sử dụng 4 luồng chạy song song call vào service).
Sử dụng guava cache để giảm số lần tính cùng một số của bài 3, nếu sau 30s không có request thì cache tự xóa. Tính lại tốc độ của service lần này

5. Upload project bài 4 lên github.com.

# PHẦN 3. Hadoop/Spark/Cassandra

## 1. Tài liệu [2 Tuần]

Tài liệu chung về bigdata:

Hadoop:
http://barbie.uta.edu/~jli/Resources/MapReduce&Hadoop/Hadoop%20The%20Definitive%20Guide.pdf
sách tiếng anh về hadoop đọc chương 2,3, 4(phần serialization,sequence file), 6
có thể tham khảo slide tiếng việt
https://drive.google.com/drive/u/0/folders/0B4fliHYgI6GrWW9jSXlGM2JiMnM
tài liệu nâng cao về mapreduce(không bắt buộc)
http://barbie.uta.edu/~jli/Resources/MapReduce&Hadoop/MapReduce%20Design%20Patterns.pdf
Spark:
https://spark.apache.org
https://github.com/CjTouzi/Learning-RSpark/blob/master/Zaharia%20M.%2C%20et%20al.%20Learning%20Spark%20(O'Reilly%2C%202015)(274s).pdf
Tài liệu tiếng việt
https://docs.google.com/presentation/d/1sk92roXDIEfSUYuEY9_0BsS-QAeNM-tC_9spmMl63d0/edit#slide=id.g266040a5cf_0_11

## 2. Test:

**1. Cài đặt hadoop:**
Cài đặt theo link:
http://pingax.com/install-hadoop2-6-0-on-ubuntu
Dùng jps để kiểm tra lại phần cài đặt.
Note: hệ thống có 2 user: user hduser chứa quyền cao nhất trên hdfs và user của bạn(trong ví dụ là abc)
Dùng shell với user hduser tạo folder /user/abc - chown cho user abc .
Dùng shell với user abc upload file text lên. Dùng lệnh cat của hdfs để lấy nội dung của file text ra.
**2. Map Reduce:**
- Viết ứng dụng wordcount = mapreduce, chạy lệnh từ shell
- Viết ứng dụng wordcount = mapreduce, có combiner, giải thích cách combiner thực hiện. đầu ra ghi ra ghi vào server redis dựng lên ở bài tập trước. (Note: với mỗi reducer chỉ khởi tạo connection đến redis 1 lần)
**3. Spark:**
**3.1 Lý thuyết:**
- Giải thích khái niệm lazy của rdd
- Hiểu các operation cơ bản của rdd: reduce< reduceByKey, groupByKey, collect, saveAsTextFile
join,union. .....
- Hiểu dc cách accumulator hoạt động
-HIểu được cách broadcast hoạt động
- Hiểu được shufffle-sort
- Hiểu được rdd-persistence
- HIểu được cách hoạt động của spark streaming
-Hiểu được các phần về spark sql

**3.2 Thực hành**
**Các bài tập của spark phải chạy trên yarn thông qua spark-submit, chạy được trên mode local không có ý nghĩa.**

chạy ứng dụng wordcount trên yarn, dùng hàm collect với rdd trả về, in ra màn hình.
dùng accumulator để tính số dòng, số chữ cái(a-z của toàn bộ văn bản)

Tạo 2 file dữ liệu (text). File thứ 1 chứa thông tin sinhvien-id và điểm + môn học + loại kiểm tra ( cách nhau = dấu tab) 100k dòng nằm trên hdfs. File thứ 2 chứa thông tin về loại kiểm tra và hệ số (15' hệ số 1, 1 tiết hệ số 2 , cuối kỳ hệ số 5) nằm trên local. Viết chương trình spark tính điểm trung bình của từng môn của từng sv.

Lưu dữ liệu vừa tạo bằng spark-sql thành dạng parquet.
Dùng spark shell đọc dữ liệu vừa tạo. lấy ra điểm của 1 sinh viên. tính điểm trung bình của toàn bộ các sinh viên của các môn học.

# Lập trình thuật toán kmean trên spark.

# Phần 4: Golang

## Tài liệu

Getting started

https://tour.golang.org/welcome/1

- Build ra file chạy từ main.go

Viết hàm test trong golang

https://golang.org/pkg/testing

- Hiểu rõ cơ chế error handling trong golang

- Hiểu rõ cách viết 1 hàm trong golang:

+ Nhiều đầu ra cho 1 hàm khác nhau

+ Khi nào 1 hàm viết in hoa và khi nào 1 hàm viết thường

- Giải thích được khái niệm về concurrency:

https://blog.golang.org/concurrency-is-not-parallelism

+ Thế nào là go routine, cách tạo go routine

+ Hiểu được khái niệm channel, wait group, read write lock

+ HIểu được cách 1 biến đếm thực thi trong hệ thống đa luồng

https://gobyexample.com/atomic-counters

- Dựng được lên 1 http server trả về chuỗi hello-world

- Tương tác với mysql

- Build 1 chuối json với easyjson:

https://github.com/mailru/easyjson

## Test:

1. Viết 3 server: Service 1 trả về 1 số, server 2 trả về 1 số, 2 service này có độ trễ 100ms, Service 3 call 2 service trên 1 cách song song. trả về kết quả là tổng 2 số

2. Viết được ví dụ đọc các bản ghi từ mysql và in ra màn hình (có thể tự cài mysql hoặc dùng docker để pull image của mysql về)
3. Viết 1 server với 1 counter, sau đó viết 1 client gọi vào server 10k lần. Trả về kết quả số request đã thực hiện.

# Phần 5 : DOCKER

## Tài liệu. https://docs.docker.com/

1. Cài đăt docker , docker-compose, docker-swarms

- Deploy 1 container trên docker theo hướng dẫn https://docs.docker.com/get-started/part2/

- Tìm hiểu cách tạo 1 image docker qua dockerfile.

- Xác định cách quản lý data trong docker. Hiểu khái niện volumes, bind mount, tmpfs mount https://docs.docker.com/storage/

- Các quản lý network trong docker: bridge, overlay, ingress, host, macvlan  network https://docs.docker.com/network/

- Tìm hiểu cách cấp phát tài nguyên trên docker : https://docs.docker.com/config/containers/resource_constraints/

## Test.

0. Tạo image các service phần golang từ dockerfile. Tạo image từ 1 container.

1. Deploy 1 image theo docker-compose

2. Deploy 1 image trên tất cả các node của docker swarms theo docker stack.

3. Deploy các service trên phần golang lên docker và tiến hành chạy lại kết quả bài test.

4. Deploy image mysql sao cho tự động bật lại nếu container chết và không bị mất data trước đó.

5. Deploy 2 container trên 1 host và từ 1 container call đến container còn lại theo bridge network.