دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش کارآموزی (هفته سوم)

محل کارآموزی: شرکت سامانه گستر سحاب‌پرداز

نام استاد کارآموزی

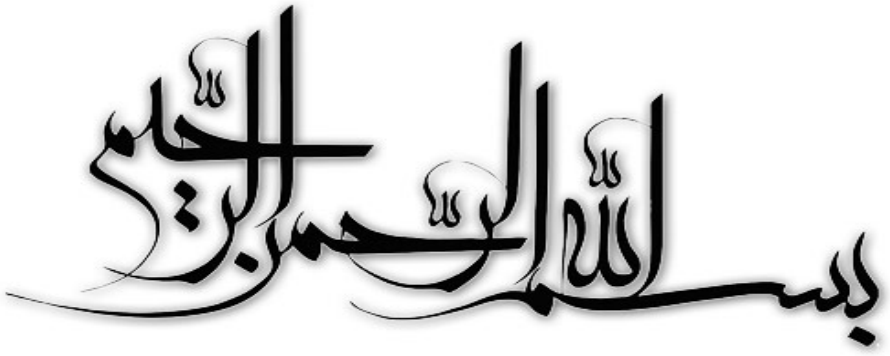
دکتر مسعود صبائی

نام دانشجو

امیرمحمد پیرحسینلو

۹۵۳۱۰۱۴

تابستان ۱۳۹۸


بسم الله الرحمن الرحيم

فهرست مطالب

فصل اول
پروژه دوم - آشنایی با ابزارها و مفاهیم جدید در حوزه‌ی داده‌های حجیم

۱-۱ مقدمه

در هفته سوم، تعریف پروژه دوم در اختیار ما قرار گرفت که هدف آن طراحی یک موتور جستجو است. برای طراحی چنین نرم‌افزاری، نیاز است با تعدادی از ابزارها و همچنین با مفاهیمی در حوزه‌ی داده‌های حجیم **(big data)** آشنا شویم. این ابزارها شامل موارد زیر می‌شوند:

- فریم ورک **Hadoop** [۱]
- پایگاه داده‌ی **HBase** [۲]
- موتور جستجوی **ElasticSearch** [۳]
- فریم ورک **Spark** [۴]
- پلتفرم **Kafka** [۵]

در این هفته با پایگاه داده‌های **ACID** و تئوری **CAP** آشنا شدیم. همچنین نحوه نصب و کار با ابزارها و فریم‌ورک‌های یاد شده را در سطح مقدماتی فراگرفتیم.
طراحی معماری نرم‌افزار و برنامه نویسی در هفته‌های آتی انجام خواهد شد.

۱-۲ تعریف پروژه دوم

هدف طراحی یک موتور جستجو است که باید حدود ۱۰۰ میلیون صفحه **(web page)** را بازدید و محتوای آن شامل لینک‌های خروجی و متن نوشته شده در صفحه را **index** کند. برای ذخیره‌سازی و پردازش داده‌ها دو سرور در اختیار ما قرار گرفته است. هر سرور ۸ ترابایت فضای ذخیره‌سازی از نوع دیسک سخت **(Hard Disk - HDD)** ، ۱۶ گیگابایت حافظه **RAM** و یک **CPU** ۸ هسته دارد.
فایل تعریف پروژه در کنار فایل گزارش قرار دارد.

۱-۳ معرفی ابزارها

در این بخش به معرفی جزئی چند تا از مهم‌ترین ابزارهای مورد استفاده در این پروژه می‌پردازیم.

۱-۳-۱ فریم‌ورک Hadoop

Hadoop یک فریم‌ورک برای ذخیره‌سازی داده‌های انبوه و پردازش آن‌ها است که بر بستر کامپیوترهای معمولی نصب و اجرا می‌شود. **Hadoop** با این فرض ساخته شده است که احتمال خراب شدن **(failure)** حافظه ذخیره‌سازی جانبی [۶] (معمولا از نوع **HDD**) کم نیست، به همین دلیل فریم‌ورک طوری ساخته شده است که این خرابی را به صورت اتوماتیک تشخیص داده و با انجام عملیات‌هایی نظیر ایجاد افزونگی **(replication)** مانع از دست‌رفتن داده **(data loss)** می‌شود.
مدل پردازش داده‌ها در این فریم‌ورک، **Map Reduce** است و یک موتور مخصوص این کار در دل **Hadoop** قرار دارد. از نسخه ۲ به

[۱] https://en.wikipedia.org/wiki/Apache_Hadoop
[۲] https://en.wikipedia.org/wiki/Apache_HBase
[۳] https://en.wikipedia.org/wiki/Elasticsearch
[۴] https://en.wikipedia.org/wiki/Apache_Spark
[۵] https://en.wikipedia.org/wiki/Apache_Kafka
[۶] https://www.komprise.com/glossary_terms/secondary-storage

بعد این موتور جای خود را به فریم‌ورک **YARN**[۷] داده است، البته خود **YARN** از این موتور استفاده می‌کند.
این فریم‌ورک برای ذخیره‌سازی داده‌ها از یک سیستم مدیریت فایل به‌نام **HDFS**[۸] استفاده می‌کند. داده‌ها به صورت بلوکی همراه با افزونگی توسط این **file system** در سرورها قرار می‌گیرند. این فریم‌ورک از پروسه‌های زیر تشکیل شده است:

NameNode

مدیریت فایل سیستم **Hadoop** یا همان **HDFS** برعهده‌ی این پروسه است. متادیتا **(metadata)** ی فایل‌های موجود در خوشه **(cluster)** در اختیار این پروسه است. مثلا اطلاعاتی مانند این‌که هر فایل به چند بلوک تقسیم شده‌است، هر بلوک در کدام سرور قرار دارد و برای هر بلوک چند افزونگی ایجاد شده و آن‌ها در کجا قرار دارند. تنها یک پروسه **NameNode** برای کل خوشه اجرا می‌شود. برای جلوگیری از **SPOF**[۹] می‌توان یک **Secondary NameNode** اجرا کرد که از **NameNode** اصلی به صورت دوره‌ای **snapshot** می‌گیرد و در صورت از کار افتادن **NameNode** اصلی، مانع از دست رفتن اطلاعات حیاتی و متادیتاها می‌شود.

DataNode

داده‌های اصلی توسط پروسه‌های **DataNode** در سرورهای مختلف که در خوشه هستند نگه‌داری می‌شود. معمولا به ازای هر سرور، یک پروسه **DataNode** اجرا می‌شود. این پروسه‌ها به صورت تناوبی در هر ۳ ثانیه صحت عملکرد خود را به **NameNode** گزارش می‌دهند.

TaskTracker

داده‌ی مورد نظر برای انجام عملیات‌های **Map**[۱۰] و **Reduce**[۱۱] را از **DataNode** گرفته و پس از انجام عملیات، نتیجه را به **JobTracker** باز می‌گرداند. معمولا در کنار هر **DataNode** ، یک **TaskTracker** حضور دارد.

JobTracker

برنامه‌ی **MapReduce** که توسط کاربر نوشته می‌شود به این پروسه تحویل داده می‌شود تا اجرا شود و نتایج توسط این پروسه به کاربر برگردانده می‌شود. برای انجام عملیات، این پروسه کارها را بین **TaskTracker** ها پخش می‌کند، نتیجه کار را از آن‌ها گرفته، ادغام کرده و نتیجه نهایی را به کاربر برمی‌گرداند. این پروسه در صورتی که یک **TaskTracker** با مشکل روبه‌رو شود یا نتواند وظایفش را انجام دهد، وظیفه را به یک **TaskTracker** دیگر سپرده و خطاهای پیش‌آمده را تا جایی که بتواند رفع می‌کند. اطلاعات مربوط به محل فایل‌ها در خوشه را از **NameNode** دریافت کرده و در اختیار **TaskTracker** ها قرار می‌دهد.

این‌طور می‌توان گفت که در این معماری **Master-Slave**[۱۲] ، **NameNode** و **JobTracker** نقش ارباب **(master)** و **DataNode** ها و **TaskTracker** ها نقش برده **(slave)** را ایفا می‌کنند.
برای ارتباط با **Hadoop** می‌توان از **Java API** استفاده کرد. یک برنامه ساده که عملیات **MapReduce** را بر روی یک فایل کوچک انجام می‌هد در کنار فایل گزارش قرار دارد. وظیفه این برنامه پیدا کردن فرکانس کلمات در فایل ورودی است. نحوه نصب فریم‌ورک و اجرای برنامه در ادامه آمده است.

[۷] **Yet Another Resource Negotiator**
[۸] **Hadoop Distributed File System**
[۹] **Single Point Of Failure**
[۱۰] نگاشت
[۱۱] کاهش
[۱۲] https://en.wikipedia.org/wiki/Master/slave_(technology)

۱-۴ پایگاه داده HBase

یک پایگاه‌داده‌ی توزیع‌شده [۱۳]، غیررابطه‌ای [۱۴]، ستون‌گرا [۱۵] و کلید-مقدار [۱۶] است که بر بستر **HDFS** اجرا می‌شود. برای ذخیره‌سازی مقدار بسیار زیادی از داده‌های تنک **(sparse)** همراه با ویژگی **fault tolerant** ساخته شده است. برای عملیات‌های خواندن و نوشتن سریع بر روی دیتاست **(dataset)** های بزرگ همراه با گذردهی **(throughput)** بالا و تاخیر **(latency)** کم بسیار مناسب است. به‌طور خلاصه از ویژگی‌های **HBase** می‌توان به موارد زیر اشاره کرد:

- فشرده‌سازی **(compression)**
- عملیات‌های درون حافظه‌ای [۱۷]
- سازگاری **(consistency)**
- تا حدودی قابلیت تحمل خطا [۱۸]
- و...

راه‌های دسترسی و فرمان‌دادن به **HBase** عبارتند از:

- **Java API**
- **REST API** [۱۹]
- **Apache Avro** [۲۰]
- **Apache Thrift** [۲۱]

در این پروژه از **Java API** استفاده می‌کنیم به همین دلیل از توضیح موارد دیگری که در بالا ذکر شد اجتناب می‌کنیم.
یک نمونه کد ساده برای اتصال به **HBase** همراه گزارش ضمیمه شده است. در این کد چند جدول در داخل پایگاه‌داده ساخته شده و در آن داده قرار می‌دهیم و داده‌ها را می‌خوانیم و به روز رسانی می‌کنیم. نحوه نصب **HBase** و اجرای کد در ادامه آمده است.

۱-۵ ElasticSearch

یک موتور جستجوی متن توزیع‌شده، بلادرنگ **(real time)**، مقیاس‌پذیر، مستقل از سکو [۲۲] و کاراست که برپایه **Apache Lucene**[۲۳] ساخته شده است. امکان جستجوی بلادرنگ را فراهم می‌کند و مانند پایگاه داده‌ی **Mongodb**، سندگرا **(document-based)** است. انواع داده‌های ساخت‌یافته **(structured)** و بدون ساختار **(unstructured)** را پشتیبانی می‌کند.
ElasticSearch موارد زیر را پشتیبانی می‌کند:

[۱۳] **distributed**
[۱۴] **non relational**
[۱۵] **column oriented**
[۱۶] **key-value store**
[۱۷] **in-memory operations**
[۱۸] **partially tolerable against failure**
[۱۹] https://en.wikipedia.org/wiki/Representational_state_transfer
[۲۰] https://en.wikipedia.org/wiki/Apache_Avro
[۲۱] https://en.wikipedia.org/wiki/Apache_Thrift
[۲۲] **cross-platform**
[۲۳] https://en.wikipedia.org/wiki/Apache_Lucene

- توزیع‌پذیری **(distribution)**
- **sharding** [۲۴]
- تکثیر **(replication)**
- خوشه بندی **(clustering)**
- معماری چندگره‌ای [۲۵]
- عملیات‌های فله‌ای [۲۶]
- و...

اما موارد زیر را پشتیبانی نمی‌کند:

- عملیات‌های نگاشت-کاهش [۲۷]
- معاملات توزیع‌شده [۲۸]

برای برقراری ارتباط با **ElasticSearch** باید از پروتکل **HTTP** استفاده کرد. داده‌های ارسالی یا دریافتی در قالب ساختارداده‌ی **JSON** در قسمت **body** درخواست **(request)** یا پاسخ **(response)** قرار می‌گیرند.

برای مثال دستور زیر

```
curl --header "Content-Type: application/json" --request POST

--data '{"title": "Java 8 In Depth", "category":"Java"}'

http://localhost:9200/bookstore/books/1001
```

داده‌ی موجود در قالب **JSON** را در ایندکس **bookstore** با تایپ **book** و ایدی **1001** قرار می‌دهد.

ساختار **URL** ارسالی به **ElasticSearch** به صورت زیر است:

http://server:port/index(lowercase)/Type/

یک برنامه ساده برای ارسال داده به **ElasticSearch** و دریافت داده از آن همراه گزارش ضمیمه شده است که شامل کد جاوا و کد **curl** می‌شود. نحوه نصب **ElasticSearch** و اجرای برنامه در ادامه آمده است.

۱-۶ حالات نصب

ابزارهای نام برده شده در قسمت‌های قبل را می‌توان به دو صورت نصب کرد:

[۲۴] https://en.wikipedia.org/wiki/Shard_(database_architecture)
[۲۵] **multinode architecture**
[۲۶] **bulk operations**
[۲۷] **Map-Reduce operations**
[۲۸] **distributed transactions**

۱-۶-۱ Standalone mode

در این حالت، یک نمونه **(instance)** از هر کدام از ابزارها را بر روی تنها یک سرور اجرا می‌کنیم. نکته قابل توجه این است که این ابزارها برای سیستم‌های توزیع‌شده و خوشه‌ای ساخته شده‌اند و محبوبیت و کارایی بالای آن‌ها به خاطر استفاده از معماری توزیع‌شده است. با محدود کردن آن‌ها تنها به یک سرور نمی‌توانیم کارایی مطلوب را به دست آوریم. به همین دلیل شرکت دو سرور در اختیار ما قرار داده است تا آن‌ها را به صورتی که در ادامه توضیح می‌دهیم نصب کنیم.

۲-۶-۱ Distributed mode

در این حالت در هر سرور در خوشه یک پروسه از ابزار مورد نظر اجرا کرده و پیکربندی مربوط به ارباب **(master)** و برده‌ها **(slaves)** را به صورت مجزا انجام می‌دهیم.

۷-۱ نصب ابزارها و اجرای کدهای قسمت‌های قبل

کد کار با ابزارها در پوشه‌های مختلف ضمیمه شده است، منتهی قبل اجرای آن‌ها باید ابزارها نصب باشند. برای نصب آن‌ها به آدرس‌های زیر مراجعه کنید.

- نصب **Hadoop**

https://www.edureka.co/blog/install-hadoop-single-node-hadoop-cluster

- نصب **HBase**

https://computingforgeeks.com/how-to-install-apache-hadoop-hbase-on-ubuntu

- نصب **ElasticSearch**

https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

نحوه اجرای کدها در سیستم عامل لینوکس (باید جاوا از قبل نصب باشد.):

- اجرای نمونه کد کار با **Hadoop**

java -jar test-1.0-SNAPSHOT-jar-with-dependencies.jar input.txt output

برای مشاهده خروجی:

cat output/part-r-00000

- اجرای نمونه کد کار با **HBase**

ابتدا باید **Hadoop** و **HBase** اجرا شده باشند:

start-all.sh

start-hbase.sh

حال کد را اجرا می‌کنیم:

java -jar 1-1.0-SNAPSHOT-jar-with-dependencies.jar

- اجرای نمونه کد کار با **ElasticSearch**

java -jar 1-1.0-SNAPSHOT-jar-with-dependencies.jar 2> output.txt

دستورات **curl** نیز در فایل **commands.txt** قرار دارد.

۱-۸ نتیجه گیری

در این هفته با ابزارها و مفاهیم جدیدی در حوزه داده‌های حجیم آشنا شدیم که از میان آن‌ها می‌توان به ابزارهای **Hadoop** ، **HBase** و **ElasticSearch** اشاره کرد. نصب ابزارها و کار با آن‌ها را به صورت مقدماتی فراگرفتیم و معماری نرم‌افزار را در هفته آتی انجام خواهیم داد.

واژه‌نامه‌ی انگلیسی به فارسی

A

architecture . . . . . معماری

B

big data . . . . . داده‌ی حجیم

body . . . . . بدنه

bulk . . . . . توده، فله

C

cluster . . . . . خوشه

clustring . . . . . خوشه بندی

compression . . . . . فشرده سازی

consistency . . . . . سازگاری

D

data loss . . . . . از دست دادن داده

dataset . . . . . مجموعه‌ای از داده‌ها

distribution . . . . . توزیع

document-based . . . . . سندگرا

F

failure . . . . . نقصان،خطا

fault tolerant . . . . . مقاوم در برابر خطا

H

hard disk . . . . . دیسک سخت

I

index . . . . . فهرست کردن

instance . . . . . نمونه

L

latency . . . . . تاخیر

M

map . . . . . نگاشتن

master . . . . . ارباب

O

operation . . . . . عملیات

P

platform . . . سکو، در اینجا منظور سیستم عامل است.

R

real time . . . . . بلادرنگ

reduce . . . . . کاهش دادن

replication . . . . . تکثیر

request . . . . . درخواست

response . . . . . واکنش

S

slave . . . . . برده

snapshot . . . . . کپی

sparse . . . . . تنک

structured . . . . . ساخت‌یافته

T

throughput . . . . . گذردهی

transaction . . . . . معامله

U

unstructured . . . . . بدون ساختار

W

web page . . . . . صفحه وب