Hadoop 및 Mongo DB기반 대용량 로그 처리 시스템 설계 및 구현 연구 결과 발표

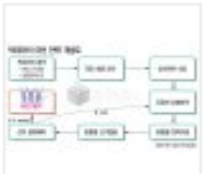
200901435 박헌일

INDEX

연구 배경

1.기존 관계형 데이터베이스 시스템(RDBMS)기반 로그 처리 시스템이
Hadoop 기반 BigData(대용량)플랫폼인 시스템으로 넘어가고 있음

2. 빠르게 대용량의 웹 로그를 처리/분석해야 되는 이유는 선거기간이나 짧은 시간 내에
통계를 내야 되는 마케팅처럼 대량으로 들어오는 유저의 성향을 파악하기 위해서
속도가 빠른 BigData(대용량)시스템이 필요
(미래의 선거는 빅데이터가 당락이 좌우- 오바마 캠프 빅데이터 선거 전략)



6.4 지방선거 **빅데이터** 활용 '주목' 2014.03.17 | 디지털타임스 | 미디어다음

데이터 분석에 기초한 '**빅데이터** 선거'는 2012년 미국 대통령 선거에서 관심을 받기 시작했다. 당시 미국 언론과 전세계는 버락 **오바마** 미국 대통령이 재...

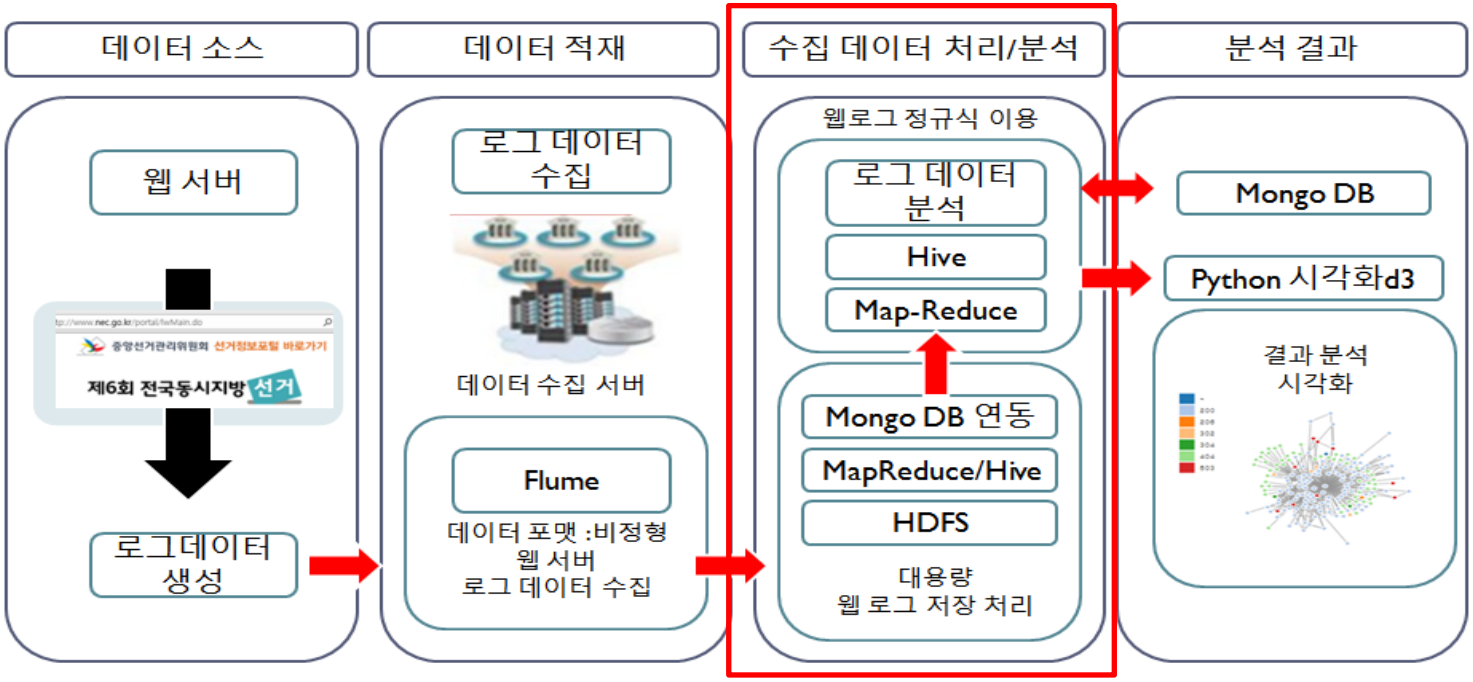
1. 서론 : 연구 설명

연구 주제

- Mongo DB 및 Hadoop 기반 대용량 웹 로그 처리 시스템 설계 및 구현

연구 내용

- 웹 로그처리 /분석 중심 시스템 설계 및 구현
- Hive, MySQL, MongoDB 기반 시스템 성능 평가(Insert , 질의문)



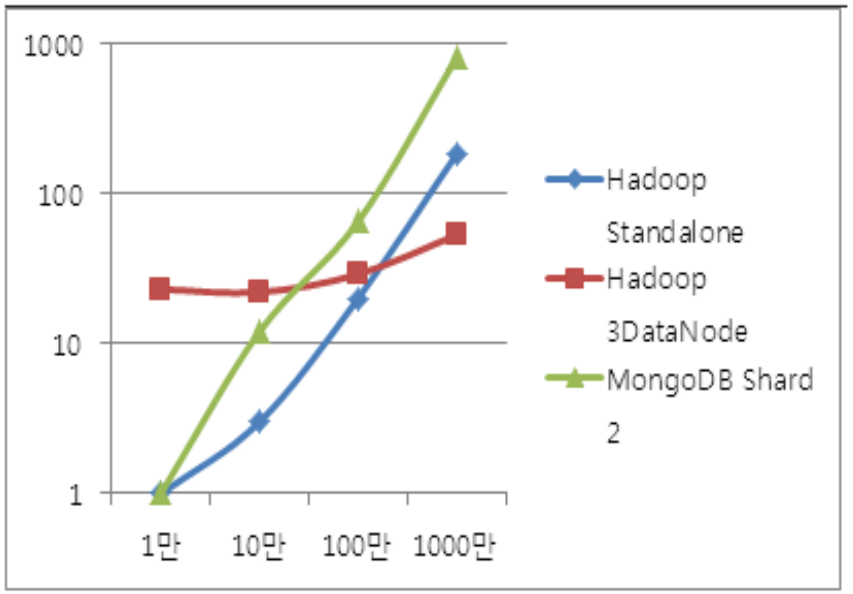
1. 서론 : 연구 배경 및 목표


시스템 구축
시스템 성능 비교
삽입 속도
쿼리 속도
Hadoop 및 Mongo DB 웹 로그 처리시스템 구현
데이터베이스 비교
결과 분석
Python
D3 이용
Fisheye Distortion
MySQL 연동
SQL 테이블생성
Sqoop으로 연동
Mongo DB연동
MapReduce
프로그래밍 작성
샘플링된 데이터 분석
웹 로그 정보 저장
웹 로그 처리/ 분석1
MapReduce
웹 로그 정규식
설계 및 구현
웹 로그 처리/ 분석2
Hive 테이블생성
웹 로그 정규식
설계 및 구현
프로그래밍 비교
Hadoop과 Python
웹 로그 정보 카운트
웹로그 이동
웹 로그 정제
로그 데이터 저장
HDFS
Hive WareHouse
로그 수집 및 적재
적재도구 비교
Flume
Flume으로 웹 로그 적재
HDFS으로 웹로그 이동
웹 로그 데이터 생성
데이터 소스
웹 서버에서
로그데이터 생성
웹 데이터 소스 발생
Web

로그 처리/ 분석 hadoop과 Mongo DB 성능 비교

Hadoop과 MongoDB Map-Reduce의 성능 비교

데이터 건 수	파일 크기	Hadoop		MongoDB Shard 2
		Standalone	Distributed	
1만 건	2MB	1초	23초	1초
10만 건	25MB	3초	22초	12초
100만 건	248MB	20초	29초	65초
1000만 건	2,480MB	183초	53초	805초
장비	-	1대	4대	4대



출처 : NHN 웹 서비스 개발실 자료

Hadoop , Hive와 Mongo DB + Map-Reduce를 같이 쓴다면 성능이 뛰어날 것으로 예상

현재 진행 사항

1. 웹 로그를 HDFS에 적재 및 저장
2. Map-Reduce 웹 로그 패턴 정규 식으로 정보 분할 및 카운트 프로그램 구현
3. Hive 웹 로그 웨어하우스 테이블 생성을 위한 Map-Reduce설계 및 구현
4. Hive로 샘플링, Map-Reduce 프로그램으로 Mongo DB와 연동

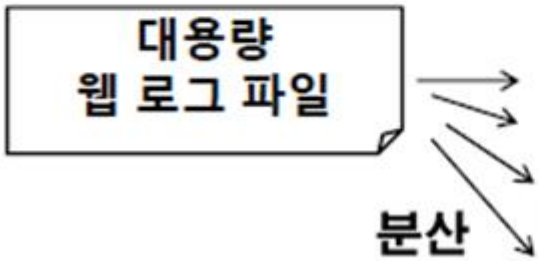
개선 연구 방향

1.Mongo DB에 대해서 연구 및 분석
2.MySQL +Hadoop System 과 Mongo DB +Hadoop System 성능 분석
대표적인 관계형 데이터베이스 MySQL 5 버전을 적용한 웹 로그 처리 시스템과
MongoDB 2버전을 적용한 웹 로그 처리 시스템의 성능을 비교, 평가
500MB, 1GB, 2GB 1.Hadoop 연동으로 삽입, 2.쿼리 속도 비교

3. 연구 구현 결과 : 기존 방식과 Hadoop 방식

- 다른 프로그래밍 말고 Hadoop 기반으로 로그 처리을 하는 가?

  1. MapReduce는 데이터셋을 나누고, 다수의 노드들
  대해 병렬적으로 실행하는 방식으로 통해 질의할 수 있는 능력
  2. MapReduce가 데이터를 DB에 Insert 할 때 기존 방식(c, python등)보다 빠름
  3. 연산을 나누는 것은 하나의 머신에 대하여 지나치게 큰 데이터를
  처리하는 것에 대한 문제를 해결



Name	Type	Size	Replication	Block Size	Modification Time
_SUCCESS	file	0 KB	1	64 MB	2014-05-06 22:14
_logs	dir				2014-05-06 22:12
part-m-00000	file	56.31 MB	1	64 MB	2014-05-06 22:12
part-m-00001	file	56.12 MB	1	64 MB	2014-05-06 22:12
part-m-00002	file	55.4 MB	1	64 MB	2014-05-06 22:13
part-m-00003	file	26.76 MB	1	64 MB	2014-05-06 22:13

3. 연구 구현 결과 : 기존 방식과 Hadoop 방식

다른 프로그래밍 (Python) - 로그 파일 입출력 형식

- 장점 : 스크립트 복잡성이 낮음
- 단점 : 대용량 측면에서 속도, 쿼리속도 데이터베이스에 삽입속도가 느림
  연계된 로그 적재 도구가 거의 없고 분석 관련 도구 지원이 미흡
  일괄적으로 작업을 하는 번거로움

Table: t1

	id	ip	time	method	url
1	0	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/edit/Main/Double_bounce_sender?topic
2	1	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1,3
3	2	64,242,88,10	[07/Mar/2004:16:	GET	/mailman/listinfo/hsdivision HTTP/1.1
4	3	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/view/TWiki/WikiSyntax HTTP/1.1
5	4	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/view/Main/DCCAndPostFix HTTP/1.1
6	5	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/oops/TWiki/AppendixFileSystem?templ
7	6	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/view/Main/PeterThoeny HTTP/1.1
8	7	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/edit/Main/Header_checks?topicparent=
9	8	64,242,88,10	[08/Mar/2004:05:	GET	/twiki/bin/edit/TWiki/UnchangeableTopicBug?topi
10	9	64,242,88,10	[08/Mar/2004:05:	GET	/twiki/bin/view/Main/TWikiGuest?skin=print HTTP
11	10	64,242,88,10	[08/Mar/2004:05:	GET	/twiki/bin/edit/Main/Maps_rbl_reject_code?topicp
12	11	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/attach/Main/OfficeLocations HTTP/1.1
13	12	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/view/TWiki/WebTopicEditTemplate HT
14	13	64,242,88,10	[07/Mar/2004:16:	GET	/twiki/bin/view/Main/WebChanges HTTP/1.1

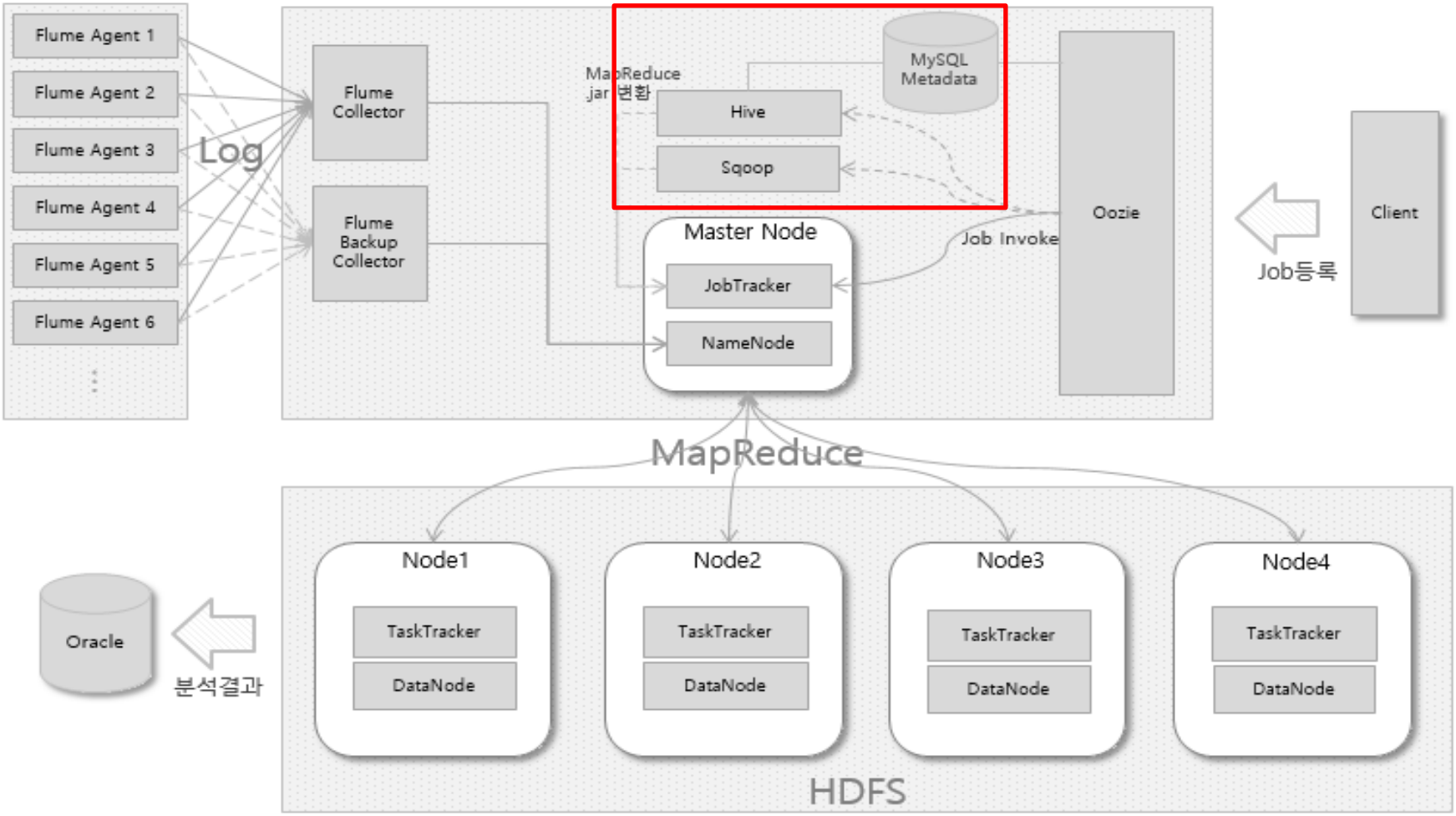
1 - 80 of 80

*Python 3.4.0: access.py -

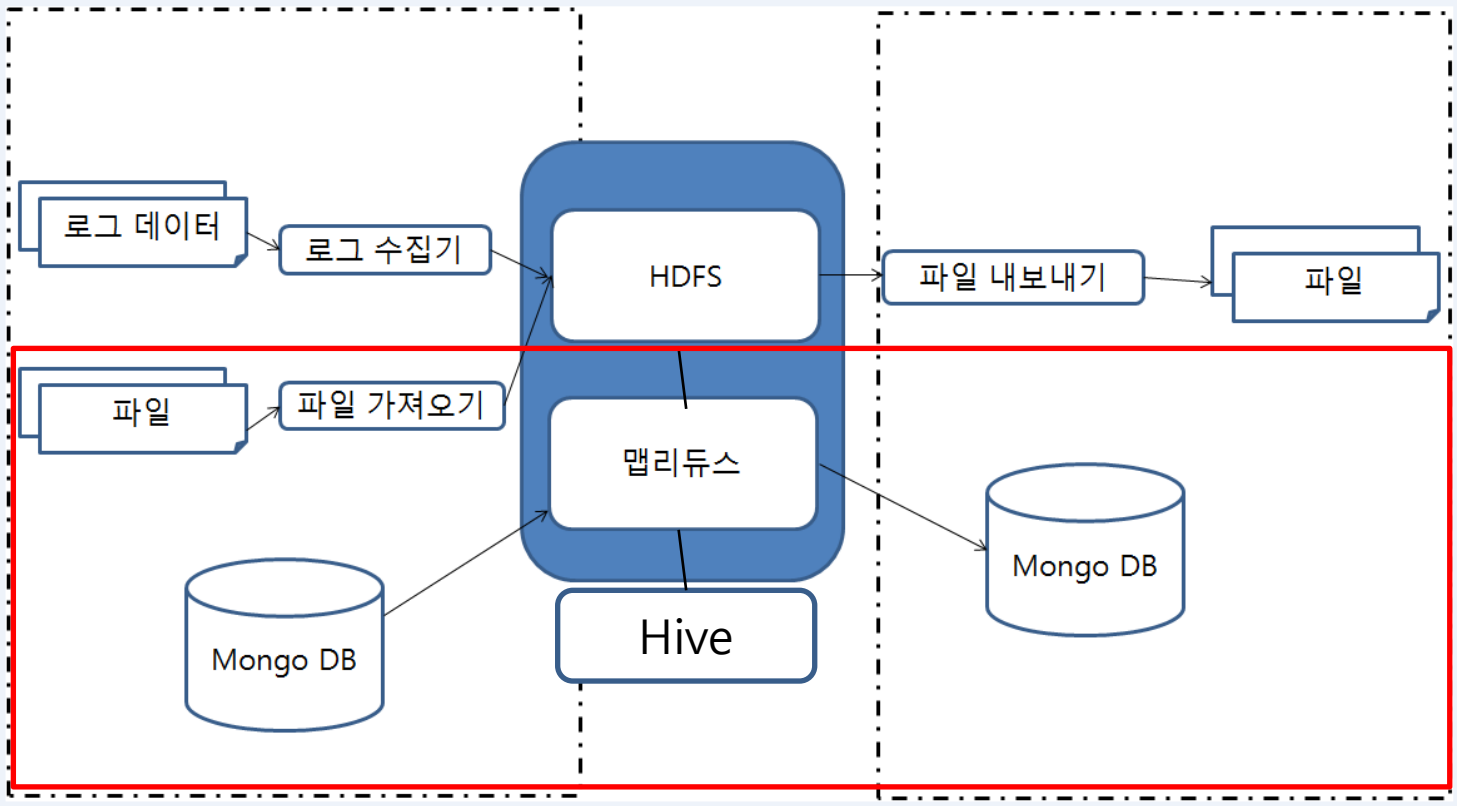
File Edit Format Run Options Windows Help

```
encoding = sys.stdout.encoding
logfile = open('access_log.txt','r',encoding="utf8")
for line in logfile:
        sys.stdout.buffer.raw.write(line.encode(encoding, 'ignore'))
for source in logfile.readlines():

        ip = re.search('"^(.+) ', source)
        print ("IP : ", ip.group(0))

        remo = re.search('(\\S+)(\\S+)', source)
        print ("Info : ", remo.group(0))

        time = re.search('\\[([\\w:/]+\\s[+\\-]\\d{4})\\]', source)
        print ("Access Time : ", time.group(0))

        method = re.search('\"([^\\s]+)', source)
        print ("Method : ", method.group(0))

        page = re.search('(\s/.+\HTTP\/\d\.\d)', source)
        print ("Page : ", url.group(0))

        code = re.search('([^\\s]+)(\\s*[^\"]*)\"', source)
        print ("code : ", code.group(0))

        byte =re.search('(\\d{3}) (\\d+)', source)
        print ("Size : ", byte.group(0))
```

3. 연구 구현 결과 - 기존 Hadoop 및 MySQL 기반 로그 처리 시스템

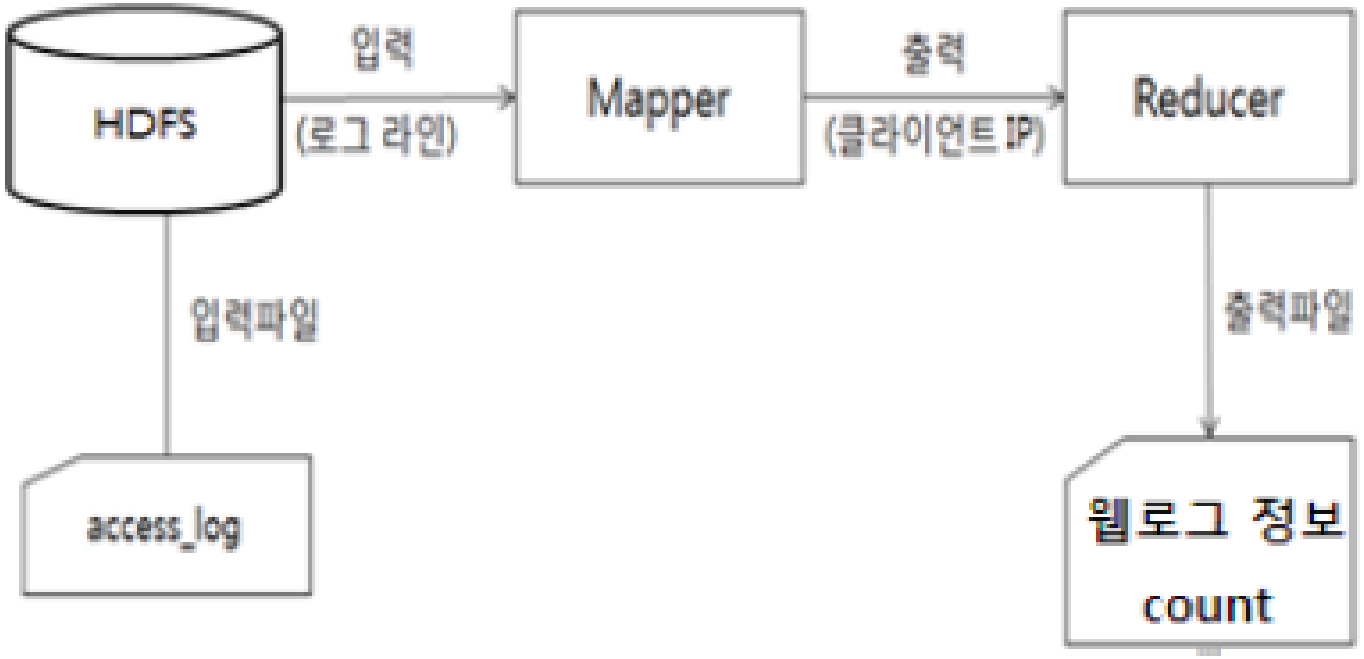


3. 연구 구현 결과 - Hadoop 및 Mongo DB 기반 로그 처리 시스템



로그 분석에서 Mongo DB를 쓰는 이유는 Hadoop의 단점을 극복하기 위해서 씀
Hadoop은 대용량으로 로그 데이터 건수가 많으면 분석 속도가 빠르나 데이터 건수가 작으면 분석 속도가 느리기 때문에 해결책으로 Mongo DB + Map-Reduce를 씀
또한, MongoDB가 Hive나 MySQL보다 빠른 쿼리 속도 때문에 사용

Map-Reduce를 이용한 웹 로그 정보 및 분할 설계



3. 연구 구현 결과 : Map-Reduce 설계

Map-Reduce를 이용한 웹 로그 정보 분할 및 카운트 방법

웹 로그 포맷을 파싱하기 위해 정규식 표현 식으로 패턴을 구현

"^(.+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"([^\\s]+)

(1) client Ip 또는 도메인 (2)- (3) - (4)accessTime (5)requestMethod

([^\\s]+)(\\s*[^\"]*)\" (\\d{3}) (\\d+) ";

(6)requsetPage (7)responsecode (8)bytesSent

```
yanafal.tele.nokia.fi - - [01/Jun/1995:00:26:31 -0600] "GET /~scottp/publish.html" 200 271
130.54.25.198 - - [01/Jun/1995:00:28:36 -0600] "GET /~macphed/finite/fe_resources/node1.html" 200 9651
```

→ ^(.+)분석 : ^ :줄의 시작 , .+ . 1회 이상 반복 ( )는 group를 지칭 , group(1) : client Ip

yanafal.tele.nokia.fi 130.54.25.198

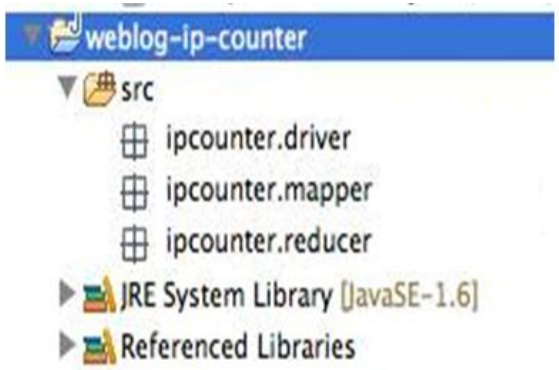
```
if ( matcher.matches() ) {
        clientIp = matcher.group(1);
        accessTime = matcher.group(4);
        requestMethod = matcher.group(5);
        requestPage = matcher.group(6);
        responseCode = matcher.group(7);
        bytesSent = matcher.group(8);
```

결론 : ip 뿐만 아니라 웹로그에서 다른 정보도 group으로 묶기 때문에 카운트 할 수 있음

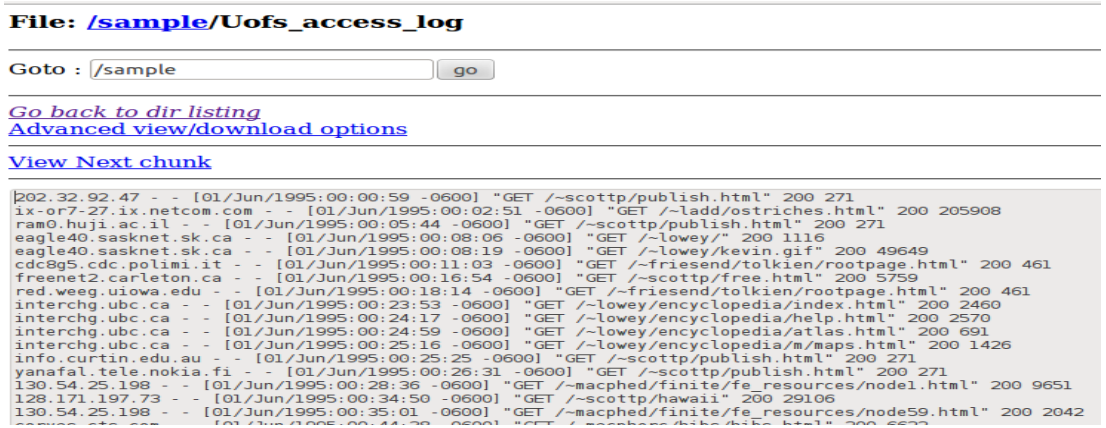
3. 연구 구현 결과 : Map-Reduce 구현

Map-Reduce를 이용한 웹 로그 IP카운트 및 분할

1.Eclipse로 Java 작업 후 Jar파일 생성



2. HDFS 웹 로그 저장



3.MapReduce 웹 로그 IP Count 프로그램 실행

```
heonil89@ubun:/opt/hadoop-1.0.3$ ./bin/hadoop jar weblog-ip-loader.jar
ipcounter.driver.IpCounterDriver /sample UofS_access_log/result/out1
14/05/06 23:05:10 INFO input.FileInputFormat: Total input paths to process : 1
14/05/06 23:05:10 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/05/06 23:05:10 WARN snappy.LoadSnappy: Snappy native library not loaded
14/05/06 23:05:11 INFO mapred.JobClient: Running job: job_201405062144_0019
14/05/06 23:05:12 INFO mapred.JobClient:  map 0% reduce 0%
14/05/06 23:05:41 INFO mapred.JobClient:  map 2% reduce 0%
14/05/06 23:05:45 INFO mapred.JobClient:  map 5% reduce 0%
14/05/06 23:05:48 INFO mapred.JobClient:  map 15% reduce 0%
14/05/06 23:05:51 INFO mapred.JobClient:  map 21% reduce 0%
```

Map-Reduce를 이용한 웹 로그 IP 카운트 및 분할 HDFS에 저장

File: /user/heonil89/UofS_access_log/result/out1/part-r-00000

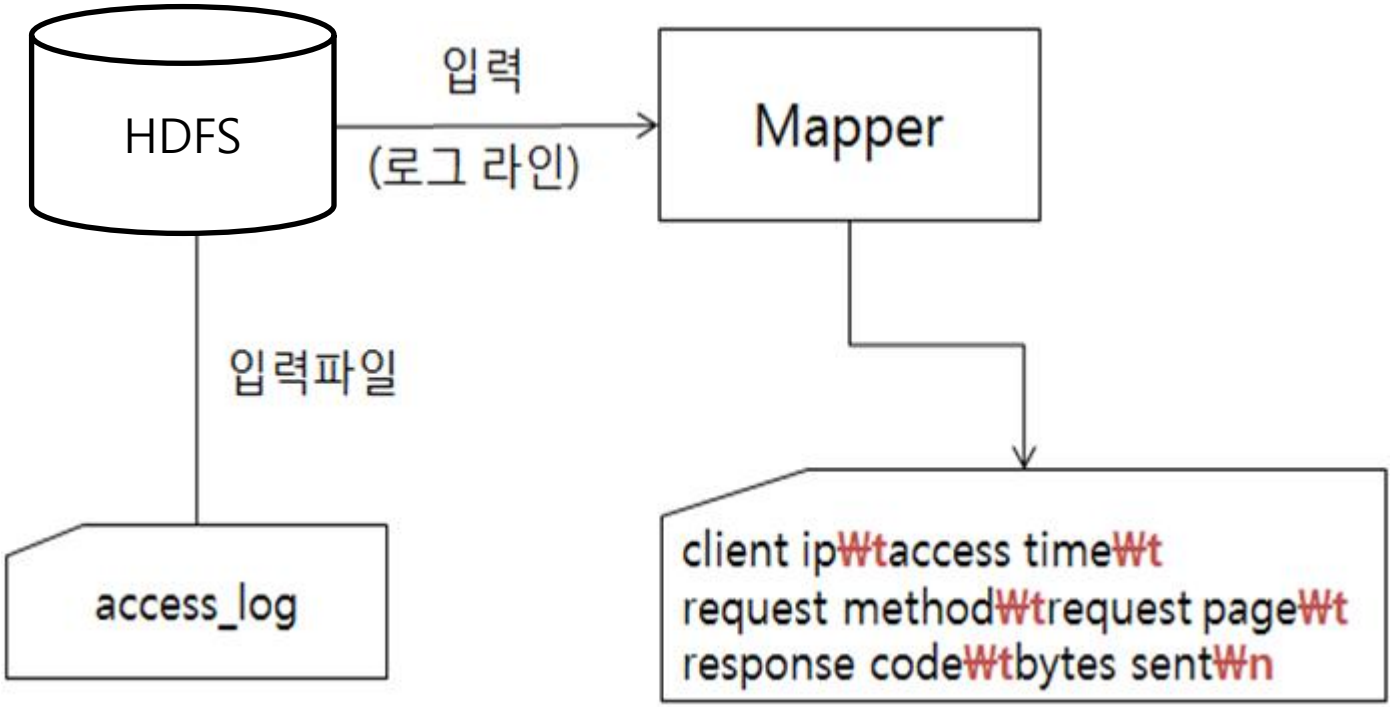
Goto : /user/heonil89/UofS_access_log/r go

Go back to dir listing
Advanced view/download options

View Next chunk

```
128.187.54.97   8
128.187.54.98   6
128.187.56.2    6
128.187.6.195   1
128.187.62.78   2
128.187.65.109  12
128.187.65.130  15
128.187.69.26   1
128.187.69.32   4
128.187.70.239  29
128.187.73.166  21
128.187.73.192  2
128.187.73.230  3
128.187.8.147   10
128.187.96.130  1
128.19.20.10    4
128.191.136.85  6
128.191.172.68  2
128.191.4.131   1
128.191.4.154   2
128.191.50.6    1
```

Hive 테이블 데이터 생성을 위한 Map-Reduce설계



Hive를 쓰는 이유 : 간편한 쿼리로 Map-Reduce 코딩을 안 짜도 되기 때문(단점 극복)
Reducer가 없는 이유 : 이 경우 웹 로그 입력 라인 하나당 하나의 라인을 출력하기 때문에, 모든 과정을 Mapper에서 처리할 수 있기 때문

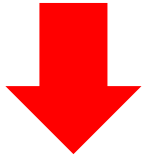
Hive weblog테이블 - 웹 로그 나누기

웹 로그 포맷을 파싱하기 위해 정규식 표현 식으로 패턴을 구현

```
"^(.+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"([^\\s]+)
```
(1) client Ip 또는 도메인 (2)- (3) - (4)accessTime (5)requestMethod

```
([^\\s]+)(\\s*[^\"]*)\" (\\d{3}) (\\d+) ";
```
(6)requsetPage (7)responsecode (8)bytesSent

```
130.54.25.198 - - [01/Jun/1995:00:28:36 - 0600]  "GET/~macphed/finite/fe_resources/node1.html 200 9651
```



```
130.54.25.198\t                                  --client ip
01/Jun/1995:00:28:36 -0600\t                     --access time
GET\t                                            --request method
/~macphed/finite/fe_resources/node1.html\t       --request page
200\t                                            --response code
9651\n                                           --bytes sent
```

Hive 테이블 데이터 생성 및 HDFS안에 Hive warehouse 생성

```
hive> CREATE TABLE weblog(
    > client_ip string,
    > access_time string,
    > request_method string,
    > request_page string,
    > response_code string,
    > bytes_sent string)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > ;
OK
Time taken: 0.41 seconds
```

Contents of directory /user/hive/warehouse

Goto : /user/hive/warehouse go

Go to parent directory

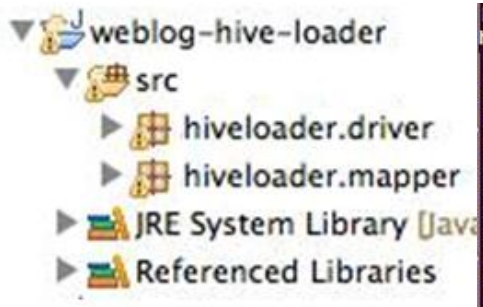
Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
weblog	dir				2014-05-06 23:17	rwxr-xr-x	heonil89	supergroup

Go back to DFS home

3. 연구 구현 결과 : Map-Reduce, Hive 구현

MapReduce로 H ive 웹 로그 테이블에 맞게 정제된 웹 로그 데이터

1.Eclipse로 Java 작업 후 Jar파일 생성



2. MapReduce웹 로그 정제 프로그램 실행

```
heonil89@ubun:/opt/hadoop-1.0.3$ ./bin/hadoop jar weblog-hive-loader.jar
hiveloader.driver.HiveLoaderDriver /sample UofS_access_log/result/out7
14/05/06 22:12:27 INFO input.FileInputFormat: Total input paths to process : 1
14/05/06 22:12:27 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/05/06 22:12:27 WARN snappy.LoadSnappy: Snappy native library not loaded
14/05/06 22:12:30 INFO mapred.JobClient: Running job: job_201405062144_0007
14/05/06 22:12:31 INFO mapred.JobClient:  map 0% reduce 0%
14/05/06 22:12:58 INFO mapred.JobClient:  map 2% reduce 0%
14/05/06 22:13:01 INFO mapred.JobClient:  map 5% reduce 0%
14/05/06 22:13:04 INFO mapred.JobClient:  map 8% reduce 0%
14/05/06 22:13:10 INFO mapred.JobClient:  map 13% reduce 0%
14/05/06 22:13:13 INFO mapred.JobClient:  map 22% reduce 0%
14/05/06 22:13:16 INFO mapred.JobClient:  map 25% reduce 0%
14/05/06 22:13:19 INFO mapred.JobClient:  map 29% reduce 0%
14/05/06 22:13:22 INFO mapred.JobClient:  map 36% reduce 0%
```

3.정제된 로그 파일 HDFS 에저장

File: /result/out7/part-m-00000

Goto : /result/out7 go

Go back to dir listing
Advanced view/download options

View Next chunk

```
202.32.92.47  01/Jun/1995:00:00:59 -0600 GET /~scottp/publish.html    200 271
ix-or7-27.ix.netcom.com 01/Jun/1995:00:02:51 -0600 GET /~ladd/ostriches.html    200 205908
ram0.huji.ac.il  01/Jun/1995:00:05:44 -0600 GET /~scottp/publish.html    200 271
eagle40.sasknet.sk.ca    01/Jun/1995:00:08:06 -0600 GET /~lowey/ 200 1116
eagle40.sasknet.sk.ca    01/Jun/1995:00:08:19 -0600 GET /~lowey/kevin.gif    200 49649
cdc8g5.cdc.polimi.it 01/Jun/1995:00:11:03 -0600 GET /~friesend/tolkien/rootpage.html    200 461
freenet2.carleton.ca 01/Jun/1995:00:16:54 -0600 GET /~scottp/free.html  200 5759
red.weeg.uiowa.edu 01/Jun/1995:00:18:14 -0600 GET /~friesend/tolkien/rootpage.html    200 461
interchg.ubc.ca 01/Jun/1995:00:23:53 -0600 GET /~lowey/encyclopedia/index.html    200 2460
interchg.ubc.ca 01/Jun/1995:00:24:17 -0600 GET /~lowey/encyclopedia/help.html    200 2570
interchg.ubc.ca 01/Jun/1995:00:24:59 -0600 GET /~lowey/encyclopedia/atlas.html    200 691
```

3. 연구 구현 결과 : Map-Reduce, Hive 구현

Hive – weblog 테이블에서 Post으로 접근한 client ip와 도메인 조회 및 카운트 (원하는 로그 정보 샘플링)

```
hive> select client_ip, count(client_ip) from weblog where request_method = 'POS
T' group by client_ip;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201405070011_0016, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201405070011_0016
Kill Command = /opt/hadoop-1.0.3/libexec/../bin/hadoop job  -kill job_2014050700
11_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-05-07 01:42:40,439 Stage-1 map = 0%,  reduce = 0%
2014-05-07 01:42:52,493 Stage-1 map = 29%,  reduce = 0%
```

3. 연구 구현 결과 : Map-Reduce, Hive 구현

heonil89@ubun: /opt/hadoop-1.0.3/hive

```
x245-50.cvm.umn.edu      1
x43glen.glen-net.ca      1
x83glen.glen-net.ca      1
xray.env.uea.ac.uk       1
xslip04.csrv.uidaho.edu  1
xslip46.csrv.uidaho.edu  2
xy01.usi.edu     5
xyplex3-4-12.ucs.indiana.edu     2
y1i.kootenay.net         1
yertle.dgp.toronto.edu   1
yorke.islandnet.com      1
yukon.sask.aecl.ca       1
yvr-ppp-29.cyberstore.ca         4
yyj-ppp-18.cyberstore.ca         1
yyz4.tor.hookup.net      2
zac01p02.zone.ca         2
zeta.fhl.washington.edu  1
zeus.usask.ca    12
zubra.cc.umanitoba.ca    2
zwicke.tn.cornell.edu    1
Time taken: 66.985 seconds, Fetched: 2817 row(s)
```

3. 연구 구현 결과 : Map-Reduce, Hive 구현 - 쿼리 문으로 샘플링

heonil89@ubun: /opt/hadoop-1.0.3/hive

```
hive> select request_method, count(request_method) from weblog1 group by request
_method
    > ;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201405122123_0032, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201405122123_0032
Kill Command = /opt/hadoop-1.0.3/libexec/../bin/hadoop job  -kill job_2014051221
23_0032
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-05-13 00:02:07,886 Stage-1 map = 0%,  reduce = 0%
2014-05-13 00:02:27,018 Stage-1 map = 42%,  reduce = 0%
2014-05-13 00:02:33,060 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.45 se
c
2014-05-13 00:02:34,067 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.45 se
```

heonil89@ubun: /opt/hadoop-1.0.3/hive

```
2014-05-13 00:02:58,959 Stage-1 map = 100%,  reduce
sec
2014-05-13 00:02:59,968 Stage-1 map = 100%,  reduce
sec
2014-05-13 00:03:00,974 Stage-1 map = 100%,  reduce
sec
2014-05-13 00:03:01,982 Stage-1 map = 100%,  reduce
sec
2014-05-13 00:03:02,988 Stage-1 map = 100%,  reduce
sec
2014-05-13 00:03:03,996 Stage-1 map = 100%,  reduce
sec
MapReduce Total cumulative CPU time: 9 seconds 460
Ended Job = job_201405122123_0032
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 9.46 sec
Write: 32 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 460 msec
OK
GET     1631742
HEAD    1845
POST    3371
Time taken: 98.058 seconds, Fetched: 3 row(s)
hive>
```

heonil89@ubun: /opt/hadoop-1.0.3/hive

```
FAILED: ParseException line 2:0 mismatched input '<EOF>' expecting FROM near 'f'
 in from clause

hive> select response_code, count(response_code) from weblog1 group by response_
code;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201405122123_0031, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201405122123_0031
Kill Command = /opt/hadoop-1.0.3/libexec/../bin/hadoop job  -kill job_2014051221
23_0031
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-05-12 23:27:42,962 Stage-1 map = 0%,  reduce = 0%
2014-05-12 23:27:55,144 Stage-1 map = 42%,  reduce = 0%
2014-05-12 23:27:58,478 Stage-1 map = 83%,  reduce = 0%, Cumulative CPU 5.1 sec
2014-05-12 23:27:59,484 Stage-1 map = 83%,  reduce = 0%, Cumulative CPU 5.1 sec
2014-05-12 23:28:00,489 Stage-1 map = 83%,  reduce = 0%, Cumulative CPU 5.1 sec
```

heonil89@ubun: /opt/hadoop-1.0.3/hive

```
ec
2014-05-13 00:16:28,812 Stage-1 map = 100%,  reduce
ec
2014-05-13 00:16:29,818 Stage-1 map = 100%,  reduce
ec
2014-05-13 00:16:30,828 Stage-1 map = 100%,  reduce
ec
2014-05-13 00:16:31,839 Stage-1 map = 100%,  reduce
ec
MapReduce Total cumulative CPU time: 9 seconds 400
Ended Job = job_201405122123_0033
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 9.4 sec
rite: 3770 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 400 msec
OK
200   813039
206   22
302   6
```

3. 연구 구현 결과 : 시각화 도구 d3 사용

웹 로그 샘플링 데이터 - 예)Response code 연관 분석

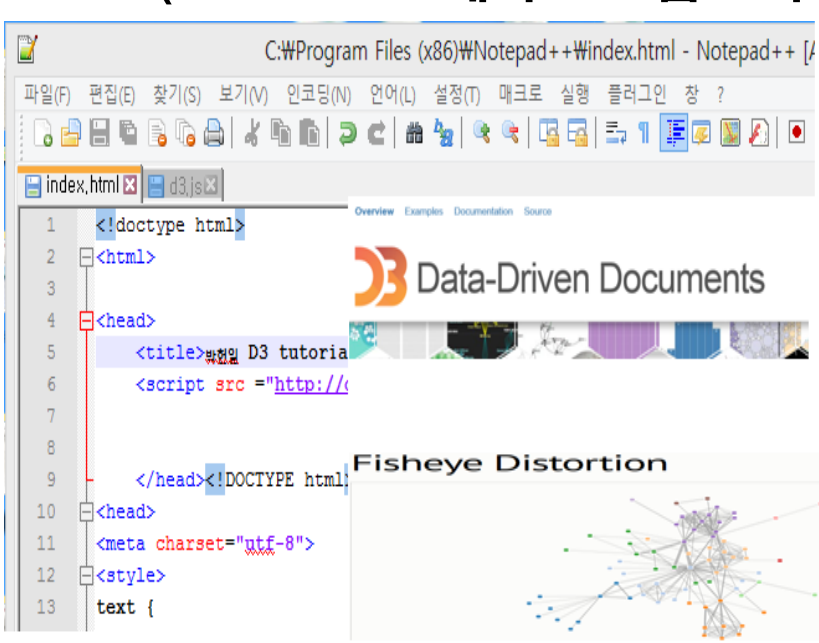
1.하둡 (MapReduce /Hive)

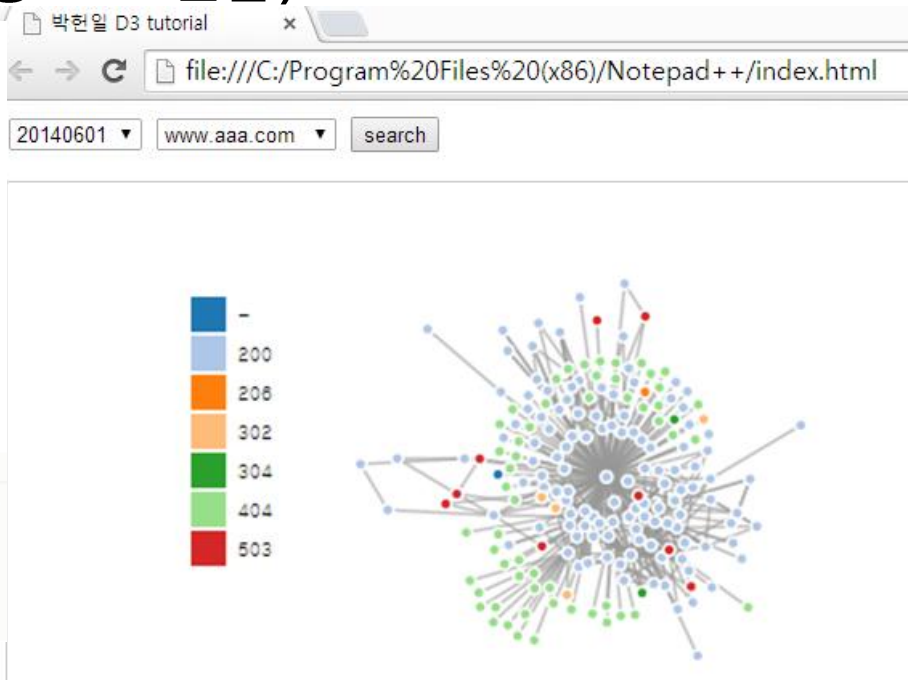
- 웹 로그에서 필요한 정보만 출력한 것으로 하였음

2. 파이썬 - MapReduce, Hive의 결과 output을 d3.js(자바스크립트)에서 사용할 수 있는 json으로 변환해주는 프로그램 사용

3. d3 -fisheye Distortion 예제(오픈소스)를 기초로 수정한 소스 사용

(NotePad++에서 스크립트 작성- html 연결)





3. 연구 구현 결과 : MySQL sqoop으로 연동

MySQL weblog 테이블 생성

```
mysql> CREATE TABLE weblog( clinet_ip VARCHAR(15), access_time VARCHAR(32), requ
est_method VARCHAR(4), request_page VARCHAR(64), response_code VARCHAR(3), bytes
_sent VARCHAR(10));
Query OK, 0 rows affected (0.09 sec)

mysql> show tables;
+------------------+
| Tables_in_weblog |
+------------------+
| weblog           |
+------------------+
1 row in set (0.00 sec)
```

```
+--------------------+---------------+
| weblog             | Size(MB)      |
+--------------------+---------------+
| information_schema |    0.00878906 |
| logs               |    0.01562500 |
| mysql              |    0.64578247 |
| performance_schema |    0.00000000 |
| weblog             | 1957.00000000 |
+--------------------+---------------+
5 rows in set (0.48 sec)
```

sqoop으로 MySQL에 weblog 테이블에 웹 로그 정보가 저장

```
heonil89@ubun:~$ sqoop export -m 1 --connect jdbc:mysql://localhost/logs --userr
ame hdp_usr --passsword test1 --table weblog --export-dir /sample/UofS_access_lo
g --input-fields-terminated-by '\t' --mysql-delimiters
Warning: /usr/lib/hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
```

```
mysql> select * from weblog limit 10;
+-----------------+-------------+----------------+-------------+---------------
+------------+
| clinet_ip       | access_time | request_method | request_page | response_code
| bytes_sent |
+-----------------+-------------+----------------+-------------+---------------
+------------+
| 202.32.92.47 -  |[01/Jun/1995 | "GET           |/~scottp/publi|  200
| 271"       |
```

3. 연구 구현 결과 : Mongo DB Map-Reduce 프로그램으로 연동

Map-Reduce 프로그램으로 HDFS에 있는 웹 로그 파일 Mongo DB 에 삽입

```
heonil89@ubun:/opt/hadoop-1.0.3$ ./bin/hadoop jar ExportToMongoDBFromHDFS.jar Ex
portToMongoDBFromHDFS
Configuration: Configuration: core-default.xml, core-site.xml
14/05/07 03:58:02 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
14/05/07 03:58:02 INFO input.FileInputFormat: Total input paths to process : 1
14/05/07 03:58:03 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/05/07 03:58:03 WARN snappy.LoadSnappy: Snappy native library not loaded
14/05/07 03:58:04 INFO mapred.JobClient: Running job: job_201405070011_0027
14/05/07 03:58:05 INFO mapred.JobClient:  map 0% reduce 0%
14/05/07 03:58:40 INFO mapred.JobClient:  map 100% reduce 0%
14/05/07 03:58:55 INFO mapred.JobClient: Job complete: job_201405070011_0027
```

Mongo DB에 weblog 테이블에 웹 로그 정보가 저장 및 쿼리 문 걸린 시간(2GB)

```
heonil89@ubun:/opt/hadoop-1.0.3$ mongo
MongoDB shell version: 2.0.4
connecting to: test
> show dbs
local   (empty)
test    1.953125GB
> db.weblogs.find();
{ "_id" : NumberLong(0), "value" : "202.32.92.47 - - [01/Jun/1995:00:00:59 -0600] \"
GET /~scottp/publish.html\" 200 271" }
```

3. 연구 구현 결과 : 성능 평가

건수에 따라서 웹 로그를 토대로 Hadoop Echo System 기반으로 쿼리 문은 select *from weblog 문 과 db.weblog.find 문 비교

Hive

```
Time taken: 0.167 seconds, Fetched: 1 row(s)
hive>
p.html          200
Time taken: 0.236 seconds, Fetched: 10 row(s)
hive>
130.89.250.24   01/Jun/1995:02:45:12 -0600
outh_32.gif             200
Time taken: 0.324 seconds, Fetched: 100 row(s)
hive>
t?file=SITES1   HTTP/1.0        200
Time taken: 0.424 seconds, Fetched: 1000 row(s)
hive>
00
Time taken: 2.022 seconds, Fetched: 10000 row(s)
hive>
gif     HTTP/1.0        200
Time taken: 19.836 seconds, Fetched: 100000 row(s)
hive>
Time taken: 180.335 seconds, Fetched: 1000000 row(s)
hive> select *from weblog limit 1;
OK
Time taken: 353.251 seconds, Fetched: 2000000 row(s)
hive>
```

MySQL

```
1 row in set (0.22 sec)
10 rows in set (0.33 sec)
100 rows in set (0.41 se
1000 rows in set (1.00 s
10000 rows in set (3.38
100000 rows in set (40.
1000000 rows in set (22
2000000 rows in set (432
```

Mongo DB

```
> db.weblogs.find().explain();
{
"nscanned" : 1,
"nscannedObjects" : 1,
"n" : 1,
"millis" : 0,
"nscannedObjects" : 10,
"n" : 10,
"millis" : 0,
"nYields" : 0,
"cursor" : "BasicCursor",
"nscanned" : 100,
"nscannedObjects" : 100,
"n" : 100,
"millis" : 0,
"nYields" : 0,
"cursor" : "BasicCursor",
"nscanned" : 1000,
"nscannedObjects" : 1000,
"n" : 1000,
"millis" : 23,
"nYields" : 0,
"nChunkSkips" : 0,
"nscannedObjects" : 10000,
"n" : 10000,
"millis" : 76,
"nYields" : 0,
blogs.find().limit(100000).e
"cursor" : "BasicCursor",
"nscanned" : 100000,
"nscannedObjects" : 100000,
"n" : 100000,
"millis" : 282,
"nYields" : 0,
"cursor" : "BasicCursor",
"nscanned" : 1000000,
"nscannedObjects" : 1000000,
"n" : 1000000,
"millis" : 1874,
"nYields" : 0,
"nChunkSkips" : 0,
"cursor" : "BasicCursor",
"nscanned" : 2000000,
"nscannedObjects" : 2000000,
"n" : 2000000,
"millis" : 4128,
"nYields" : 0,
"nChunkSkips" : 0,
```

건수에 따라서 웹 로그를 토대로 Hadoop Echo System 기반으로
쿼리 문은 select *from weblog 문 과 db.weblog.find 문 비교

Log Data (단위 sec)	MapReduce Hive	MySQL	MongoDB
	Query	Query	Query
1건	0.167	0.22	-
10건	0.236	0.33	-
100건	0.324	0.41	-
1000건	0.424	1.00	0.023
10000건(만건)	2.022	3.38	0.072
100000건(10만건)	19.836	40.22	0.282
1000000건(100만건)	180.335	220.14	1.874
2000000건(200만건)	353.251	432.52	4.128

500MB, 1GB, 2GB 웹 로그를 토대로 Hadoop Echo System 기반으로 Insert 해서 성능 비교를 한 것이고 쿼리 문은 select *from weblog 문 과 db.weblog.find 문 비교

Log Data Set	MapReduce+Hive		MySQL		Mongo DB	
	Insert	Query	Insert	Query	Insert	Query
500MB	적재 과정	422sec	329sec	531Sec	185sec	4.7sec
1GB		751sec	712sec	1202sec	301sec	6.2sec
2GB		1012sec	1624sec	2512sec	808sec	8.7sec

결론 :MapReduce+Hive는 Insert 과정이 적재 과정이어서 없으나 MapReduce를 거쳐야 한다는 Query에서 느린결과
로그 데이터 용량이 커질수록 MySQL 와 MongoDB 로그 데이터 Insert와 Query 시간이 점점 차이 나는 것을 알 수 있음.
Query문은 저 용량일 때는 거의 차이가 안 나지만 대용량일 때 차이가 심함

결론 도출

1. 기존 관계 형 데이터베이스 시스템 (RDBMS)의 단점을 극복할
   로그 처리 최적화된 방안으로 Hadoop 및 Mongo DB기반으로 한
   대용량 웹 로그 처리 시스템 을 설계 및 구현을 하였음

2. Hadoop 및 MySQL 기반 시스템보다 Hadoop 및 MongoDB 기반 시스템이
   삽입과 질의문의 속도가 저용량 일 때는 차이가 안 나지만 대용량일 때
   성능차이가 많이 나는 것을 알 수 있음
   (이후에 쿼리 문을 여러 번 돌려서 신뢰성 있는 성능평가와
   D3으로 시각화 하는 것에 부족하여 초점을 맞추어 더 연구 예정)

시사점

1. Hadoop 및 MongoDB 기반으로 한 웹 로그 처리 시스템 에서 처리 중심이 아닌
   로그 분석 중심(시각화)으로 발전
   컴퓨터 한대로 연구를 하였는 데 컴퓨터나 서버 여러 대로 병렬 분산 처리

2. MongoDB 뿐만 아니라 Hadoop과 다른 SQL-on-Hadoop과 NoSQL로 연동해서
   시스템 설계 및 구현, 성능 분석 연구

3. 웹 로그 처리/분석 뿐만 아니라 다양한 분야 로그 처리 분석으로 연구
   (시스템, 네트워크,게임 로그 등)

5. 참고 문헌

하둡 완벽 가이드 3판 - 톰 화이트 (2013.6)

하둡 프로그래밍 - 정재화 (2012.10)

하둡 실전 운용 가이드 - 에릭 새머 (2013.7)

따라하며 배우는 하둡과 빅데이터 분석 실무

Mongo DB Master가 해설하는 NoSQL & mongoDB - 주종면 (2012.2)

맵리듀스 디자인 패턴 - 도널드 마이너

실용주의 하둡 red -이송준(2013.6)

THANK YOU