모바일게임에 적용 가능한 비정형 Big Data 처리를 위한 Incremental MapReduce

박성준*, 김정웅**
*한국교육개발원, **서울게임전문학교 게임학부
e-mail:pcard8@gmail.com

Incremental MapReduce of atypical Big Data Processing in Mobile Game

Sung-Joon Park*, Jung-Woong Kim**
*Korean Educational Development Institute
**Seoul Game College

요 약

비정형 게임 Big Data에서 고효율 정보를 추출하고, 신뢰 할 수 있는 클러스터 게임서버 환경을 위한 병렬 처리를 위해 MapReduce를 사용한다. 본 논문에서는 빈번하게 입력되는 신규 게임데이터 처리를 위해 함수 Demap을 사용하는 Incremental MapReduce를 적용하여 불필요한 중간 값 저장과 재계산 없이 점차적으로 MapReduce 함수를 실행한다.

1. 서론

스마트 시대에는 기존의 정보가 아니었던 많은 것들을 시작으로 하여 우리가 하는 말, 행동, 접하는 환경, 우리의 위치 등 모든 것이 정보가 된다. 모든 상황들이 정보가 되어 기록될 수 있는 환경에서 엄청나게 다양한 형태의 데이터가 Big으로, 빠른 속도로 쏟아지고 있다.

한편 모바일게임은 특성 상 수십만 사용자가 반복적인 플레이 세션을 갖고, 각 게임 화면, 아이템 획득, 소셜 강화, 이벤트 데이터 그리고 플레이 하는 시각이 비슷하여 정형, 비정형 데이터의 처리가 가장 중요한 문제가 되고 있다.

HDFS, HBase, MapReduce 구조를 가진 Hadoop은 오픈소스 분산 처리기술 프로젝트로 정형, 비정형의 Big Data 분석에 가장 선호되는 솔루션이다.

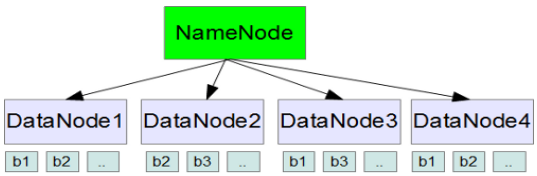
본고에서는 페타 바이트(peta byte) 이상의 대용량 게임 데이터를 신뢰할 수 없는 서버로 구성된 모바일게임 클러스터 환경에서 병렬 처리를 지원하기 위한 MapReduce의 최적화 방안을 모색한다.

2. Hadoop

Hadoop은 Big Data 분야의 리눅스라고 할 수 있을 만큼 광범위하게 사용되고 있는 오픈소스 기술이다. 값비싼 상용 제품과 달리 기존 서버 인프라 위에서 혁신적으로 저렴한 가격으로 Big Data 분석 시스템을 구축할 수 있다는 장점으로 각광을 받고 있다.

그러나 오픈소스 특유의 불안정성이나 시스템 구축에 따르는 난이도 등의 문제가 있어, Hadoop을 기반으로 하되 좀 더 안정적인 Big Data 분석을 위한 솔루션이 요구되었고, 특히 Cloud 컴퓨팅과 결합하여 서비스로 제공하는 움직임도 있다[1].

Hadoop은 신뢰성과 확장성의 구조를 사용해서 Big Data 셋에 Query를 사용하고 빠른 결과를 받을 수 있도록 하는 분산 컴퓨팅을 사용하는 오픈 소스 소프트웨어이다. Google이 GFS와 MapReduce에 기초한 분산 컴퓨팅 모델을 사용함으로 시작된 Hadoop은 자바로 구성되어 모든 플랫폼에서 작동하며, RDBMS가 아닌 비구조적 데이터에 사용된다[2]. 또한 HDFS는 Hadoop이 사용하는 storage인 Hadoop Distributed File System이다.



(그림 1) HDFS의 high-level 아키텍처

MapReduce는 Google에서 분산 컴퓨팅을 지원하기 위한 목적으로 제작하여 2004년 발표한 소프트웨어 프레임워크로 페타 바이트(peta byte) 이상의 대용량 데이터를 신뢰할 수 없는 컴퓨터로 구성된 클러스터 환경에서 병렬 처리를 지원하기 위해서 개발되었다[3]. 이 프레임워크는 함수형 프로그래밍에서 일반적으로 사용되는 Map과

1) Google의 Big query 서비스가 여기에 해당 된다.
2) http://www.thegeekstuff.com/2012/01/hadoop-hdfs-mapreduce-intro/
3) http://ko.wikipedia.org/wiki/MapReduce

Reduce라는 함수 기반으로 주로 구성된다.

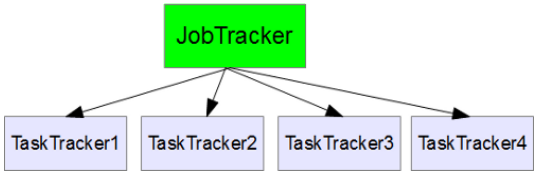
데이터베이스 CouchDB는 자연스러운 양방향 복제를 밑받침 하는 버전 컨트롤 모델과 MapReduce Query 지원을 통해 새로운 대용량 웹 문서 처리의 가능성을 보여주고 있다. 관계형 데이터베이스가 아닌 문서기반 데이터베이스이기 때문에, SQL대신 데이터 표현에 JSON을 사용하고 자바스크립트는 Query에 사용한다는 것이 특이하다[4].

CouchDB는 View라는 MapReduce함수를 이용하여 비정형 데이터를 가공 처리한다. View는 CouchDB 문서를 인수로 받아 계산을 수행하면서 키-값 쌍을 추가한다. Reduce함수가 정의되어 있다면 Map함수에서 발생시킨 키-값 쌍을 받아서 키를 기준으로 데이터를 가공한다. 조회 결과는 Map함수에서 넘겨준 키를 기준으로 정렬된다. 키 값이 null이면 DocID를 기준으로 정렬한다.

대용량 데이터의 처리를 위해서 CouchDB는 View인덱스를 지원한다. 같은 설계 문서에 포함된 View는 같은 그룹으로 인덱스 되는데 DocID와 View함수의 결과 값으로 구성된다.

3. MapReduce

CouchDB로부터 넘겨받은 Key-Value쌍으로 구성된 아래와 같은 입력 데이터는 두 가지 함수 Map과 Reduce로 정의된 MapReduce를 호출하여 실행한다.



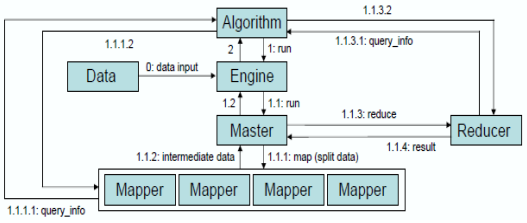
(그림 2) MapReduce Architecture

(그림 2)[5]는 MapReduce의 기본 구조를 나태고 있고, (그림 3)[6]는 분산 컴퓨팅 환경이 아닌 단일 컴퓨터 내에서 여러 개의 CPU에 의해 처리되는 멀티코어 Map Reduce의 프레임워크이다.

4) CouchDB는 Cluster Of Unreliable Commodity Hardware의 약어로, 2008년 2월에 아파치 인큐베이팅 프로젝트(http://incubator.apache.org/)에 편입된 문서 기반 데이터베이스이다. 현재 아파치 프로젝트 중에서는 유일하게 Erlang의 분산 처리 능력을 데이터베이스로 옮겨와 사용한다.

5) http://www.thegeekstuff.com/2012/01/hadoop-hdfs-mapreduce-intro/

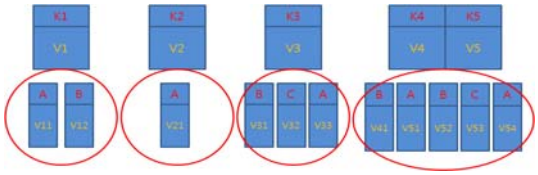
6) Cheng-Tao Chu외 6인, Map-Reduce for Machine Learning on Multicore, CS. Department, Stanford University.



(그림 3) Multicore MapReduce Framework

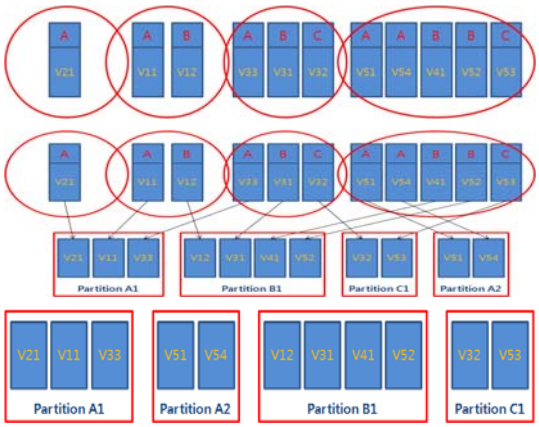
5개의 데이터 v1, v2, v3, v4, v5가 key k1, k2, k3, k4, k5를 가지고 있을 경우 MapReduce에 의해 다음 과정을 거친다.

map(in_key, in_val)
-> list(out_key, intermediate_val)
reduce(out_key, list(intermediate_val))
-> list(out_value)



(그림 4) Map()

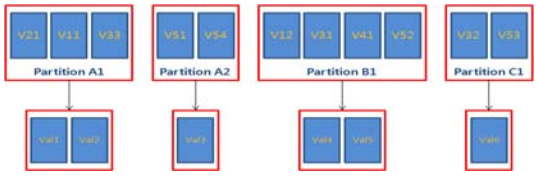
map(K1, V1) -> [(A, V11), (B, V12)]와 같이 Map함수가 복합 항목(List)을 반환 할 수도 있고 빈 항목(List)을 반환할 수 있다. 그 결과는 Map의 알고리즘을 따른다.



(그림 5-7) Rotate()

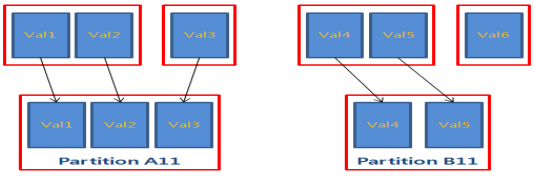
Key에 의한 정렬을 하고, Map함수에 의해 호출된 Rotate함수는 Map함수의 결과를 재편성하기 위하여 키의 항목을 재배열한다. Rotate함수는 프레임워크에 의해 가능하며 반복적으로 수행 될 수 있다. 하나의 키가 여러 개의 파티션을 출력할 수 있다[7]. 다시 Partition Key에 의한 정렬을 한다.

7) byte 크기의 덩어리로 분할하는 작업



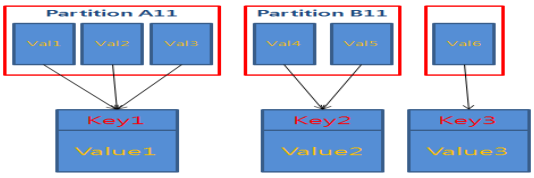
(그림 8) Reduce()

각 Partition들이 Reduce함수를 호출하여 실행되면 reduce(A1, [V21, V11, V33]) -> [Val1, Val2]와 같이 0개 또는 여러 value들을 반환 할 수 있다. Reduce의 결과는 알고리즘에 따라 다르게 나타난다.



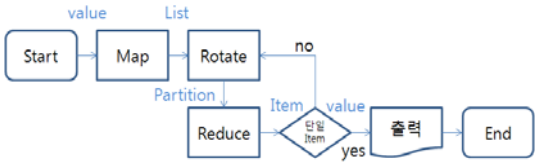
(그림 9) Rotate() again

최종 Reduce함수가 각 Key별 하나의 결과 값을 위하여 Rotate함수를 반복적으로 수행할 수 있다. Reduce함수가 복합 item을 반환하면 간단화를 계속 진행한다.



(그림 10) Reduce() again

최종적으로 MapReduce에 의해 {Key1:Value1, Key2: Value2, Key3:Value3}의 형태로 Big Data는 다른 데이터로 정보화 될 것이다.



(그림 11) Incremental MapReduce 전 단계

4. Incremental MapReduce

함수 Map과 Reduce의 절차적 실행으로 의도하는 Map Reduce 처리를 기대하는 것은 무리가 있다. 이를 해결하기 위하여 MapReduce 작업 최적화에 대한 연구가 다양하게 진행되고 있다.

김태경 외 2인은(2012) SPARQL 질의 처리 시에 Map Reduce Job의 개수를 줄이기 위한 두 가지 서로 다른 기법을 혼용 할 것을 제안했다. 서로 관련이 없는 조인 키들을 동시에 하나의 MapReduce Job에서 수행하는 비상층 조인과 중복을 이용해서 여러 개의 조인키를 한 번에 조인하는 멀티웨이 조인 기법이다. 이 두 가지 기법을 혼용함으로써, 기존에 제안된 기법보다 적은 수의 MapReduce Job을 이용해 질의를 처리 할 수 있다. 또한 이로 인해 발생하는 트리플 패턴 그룹화 문제에 대한 Grid 알고리즘을 제안했다.

강민구 외 2인은(2012) 전체 MapReduce 작업의 스케줄링 및 작업 할당을 담당하는 JobTracker의 SPOF (Single Point of Failure) 문제를 지적하고, 위와 같은 문제를 해결하고자 MapReduce 프레임워크의 JobTracker 결함 허용 메커니즘을 설계 구현하였다.

순차 패턴 마이닝(Sequential Pattern Mining)은 웹 접속 패턴, 고객 구매 패턴, 특정 질병의 DNA 시퀀스를 찾는 등 광범위한 분야에서 사용된다. 김진현 외 1인은 (2011) MapReduce 프레임웍 상에서 MapReduce 함수 호출을 최적화하는 순차 패턴 마이닝 알고리즘을 개발하였다. 이를 통해 여러 대의 기계에 데이터들을 분산시켜 병렬적으로 빈번한 순차 패턴을 찾는다.

이명철(2009)은 Incremental 상황을 특허 '스트림 데이터에 대한 점진적인 MapReduce 기반 분산 병렬 처리 시스템 및 방법'에서 논하였다. 이미 수집되어 있는 대용량 저장 데이터는 물론 분산 병렬 처리 작업이 수행되는 동안에도 연속적으로 수집되는 대량의 스트림 데이터에 대해서 점진적인 MapReduce 기반 분산 병렬 처리 기능을 제공하는 방법론이다.

본고에서는 Incremental 상황에서 MapReduce 알고리즘을 함수 Demap를 사용하여 최적화하는 방법을 제안한다.

기존의 MapReduce 알고리즘이 Big Data를 처리하는 동안에도 새롭게 입력되는 대량의 데이터를 결과에 적용시키기 위해서는 전체적으로 재계산을 하거나 모든 중간 결과값을 저장하고 변경된 입력에 대한 계산을 추가적으로 수행하는 방법이 일반적이다. 그러나 이는 직접적이지 못한 데이터를 위한 스토리지 낭비와 분리된 계산을 통합해야 하는 어려움 때문에 비효율적이다.

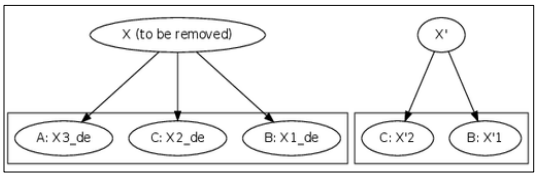
Incremental MapReduce는 이전의 결과에 대한 실행 취소가 가능할 때만 실행 할 수 있다. Incremental map reduce가 가능하다면 직접적인 데이터만을 저장하고, 계산을 분리하여 재계산 없이 효율적으로 운용 할 수 있다.

Incremental MapReduce는 아래와 같이 함수 Demap를 사용하여 MapReduce 실행 위에서 진행한다.

demap(in_key, in_val) -> list(out_key, out_val)

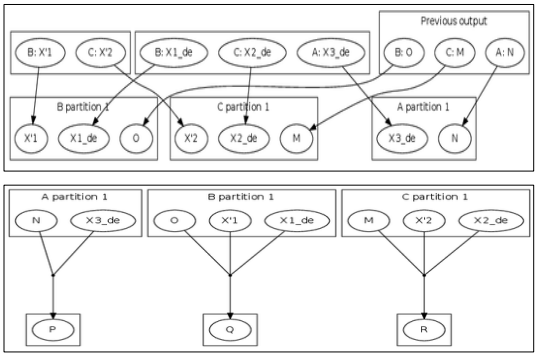
X를 X′로 변형하고, 점차적으로 결과를 변경한다. 새롭게 이전 값과 함수 Map을 가지고 Demap 함수를 실행 한다. 물론 결과는 MapReduce 알고리즘을 이용한다.

demap(X) -> [(B, X1_de), (C, X2_de), (A, X3_de)]
map(X′) -> [(B, X′1), (C, X′2)]



(그림 12) Map and Demap calls

이전 결과에 포함하여 Rotate를 하고, 갱신된 결과를 얻는 마지막 Reduce 작업을 진행한다.



(그림 13-14) Demap calls 후 Rotate, Reduce

Incremental MapReduce의 알고리즘

```
function map(in_key, in_val) {
  return [(in_val.job, 1)]
}
function demap(in_key, in_val) {
  return [(in_val.job, -1)]
}
function reduce(out_key, values) {
  return [sum(values)];
```

}을 통해 최종 결과로 {A: P, B: Q, C: R}을 얻을 수 있다.

알고리즘이 Map의 in_key 인수를 적용하지 않는 것으로 이해 될 수 있지만 일련의 MapReduce 작업은 모든 중간 결과에 대한 문서 아이디(DocID)가 필요하고, 다음 작업의 입력으로 키를 사용하기 때문에 키 출력을 제공하고 있는 것이다.

5. 결론

휴대 전화나 스마트폰, PDA, 포터블 미디어 플레이어 등의 휴대용 기기를 통해 즐기는 Mobile Game의 데이터는 정형, 비정형 Big Data 일 수 밖에 없다.

특히 비정형 게임 Big Data에서 고효율 정보를 추출하고, 신뢰 할 수 있는 클러스터 게임서버 환경을 위한 병렬 처리를 지원하는 Hadoop의 MapReduce는 매우 유용한 도구이다.

그러나 MapReduce 알고리즘이 간단한 반면 새롭게 입력되는 대량의 데이터를 결과에 적용시키기는 작업은 부하가 상당하다.

때문에 새로운 데이터에 대한 MapReduce 함수의 최적화 방법이 요구 되었고, 본 논문에서는 함수 Demap를 사용하여 불필요한 중간 값 저장과 재계산 없이 점차적으로 MapReduce를 진행하는 Incremental MapReduce를 제안한다.

참고문헌

[1] 강민구, 박기진, 황병현, MapReduce 프레임워크의 작업 완료 시간 단축을 위한 JobTracker 결함 허용 메커니즘, 정보과학회논문지 18(3) 173-180 1738-6322, 2012.

[2] 김진현, 심규석, 맵리듀스 프레임웍 상에서 맵리듀스 함수 호출을 최적화하는 순차 패턴 마이닝 기법, 정보처리학회지 제18-D권 제2호 통권 제137호 (2011년 4월) pp.81-88 1598-2866, 2011.

[3] 김태경, 김기성, 김형주, 맵리듀스에서 중복기반 조인과 비상충 조인을 이용한 효율적인 SPARQL 질의 처리, 정보과학회논문지 제39권 제4호 246p-254p 1738-6322, 2012.

[4] 머브 에이드리언, 빅데이터 처리 기술 : 오픈소스 하둡·맵리듀스, 빅데이터 문제해결방안 부상, Network times 통권212호(2011년 4월) pp.149-151 1228-9582, 화산미디어, 2011.

[5] 손민선, 문병순, 빅 데이터 시대의 한국, 갈라파고스가 되지 않으려면, LG경제연구원, 2012.

[6] 이명철(발명자), 이미영(고안자), 한양특허법인(대리인), 한국전자통신연구원(출원인), 스트림 데이터에 대한 점진적인 맵리듀스 기반 분산 병렬 처리 시스템 및 방법, (2011.06.23).

[7] 최용권, 백성하, 김경배, 배해영, MapReduce와 시공간 데이터를 이용한 빅 데이터 크기의 이동객체 갱신 횟수 감소 기법(Update Frequency Reducing Method of Spatio-Temporal Big Data based on MapReduce), 한국GIS학회지 20(2) 137-153 1738-737X KCI, 2012.

[8] Cheng-Tao Chu외 6인, Map-Reduce for Machine Learning on Multicore, CS. Department, Stanford University.

[9] http://eagain.net/blog/

[10] http://www.thegeekstuff.com/2012/01/hadoop-hdfs-mapreduce-intro/

[11] http://ko.wikipedia.org/wiki/MapReduce