빅데이터 활용을 위한 오픈소스 기반 분산데이터 처리 프레임워크 동향

강윤희
백석대학교 정보통신학부부교수
yhkang@bu.ac.kr

1. 서론

인간은 우주로부터 물리 실험, 기후변화 시뮬레이션 및 생물학적 자료 분석을 통해 대용량의 디지털 데이터를 생성하고, 센싱하고 배포한다. 최근 IDC 의 디지털 유니버스(Digital Universe) 연구에 따르면, 전 세계적으로 저장되는 데이터의 총 용량이 2020 년까지 44 제타바이트(zettabyte)에 이르며, 매 2 년마다 데이터 크기는 2 배씩 증가할 것으로 예상한다. 2012 년 Gartner 는 빅데이터(big data)를 향상된 인사이트(insight)와 효과적인 의사결정을 위한 대용량, 고속 및 다양한 특성을 가진 혁신적인 처리가 요구되는 정보자산(information asset)으로 정의하였다.

현재 구글, 페이스북, 아마존 등의 인터넷 기업은 클라우드 컴퓨팅(cloud computing) 환경의 데이터센터에 구축된 계산자원을 활용하여 개인 또는 조직에 필요한 인사이트를 제공하기 위한 빅데이터 분석을 제공하고 있다. 국내에서도 빅데이터 활용을 위한 연구가 인터넷 포털 업체와 정부 부처를 중심으로 준비하고 있거나 진행중이다. 과학연구 분야에서도 시뮬레이션 중심의 연구방법은 계측 및 센싱 장비의 도입으로 페타바이트(peta byte)

* 본 내용과 관련된 사항은 백석대학교 정보통신학부 강윤희 부교수(☎ 041-550-0504)에게 문의하시기 바랍니다.
** 본 내용은 필자의 주관적인 의견이며 IITP 의 공식적인 입장이 아님을 밝힙니다.

수준으로 데이터 폭발이 진행되고 있으며, 이 과정에서 데이터 중심으로 연구방법이 진화하고 있다[1],[2]. 연구방법의 진화에 따라 과학문제 해결은 이론기반의 실험에서 실데이터 분석 연구로 전환되고 있으며, 이들 빅데이터 분석에 과학 클라우드(science cloud) 기반의 그리드 컴퓨팅(grid computing) 환경이 고려되고 있다[7]-[9],[12],[14].

한편 국내에서는 빅데이터가 IT 업계의 화두가 되고 있으나 이를 처리하고 분석하기 위한 분산데이터 처리 기술은 제한적으로 사용하고 있다. 이러한 분산데이터 처리는 조직 내부 및 외부 데이터를 사용하며, 현재 및 과거 자료를 활용하여 예측 및 연관성 분석을 수행한다. 신축적인 컴퓨팅 자원 할당의 특성을 갖는 클라우드 환경에서 빅데이터 활용을 위한 애플리케이션과 소프트웨어 패키지는 MapReduce 기술에 특화된 분산데이터 처리 프레임워크 기술과 결합되고 있다. 과학기술 분야에서도 FutureGrid[6]와 같은 클라우드 테스트베드에서 Hadoop 및 Twister 와 같은 오픈소스 기반의 MapReduce 분산데이터 처리 프레임워크에 대한 성능평가가 이루어지고 있다[3]-[5].

본 고에서는 최근 ICT 환경 및 기술의 변화를 살펴보고 빅데이터 처리를 위한 Map Reduce 프로그래밍 모델의 구조 및 특징을 기술한다. 또한 오픈소스 기반의 빅데이터 처리를 위한 MapReduce 플랫폼을 설명하고 이들을 비교함으로써 문제점 및 해결방안을 기술한다. 본 고의 2 장에서는 주요 ICT 환경 및 기술 변화 동향, 3 장에서는 MapReduce 프로그래밍 모델, 4 장에서는 Hadoop 을, 5 장에서는 Twister 를 소개하고 이들 프레임워크를 비교한다. 6 장에서는 결론을 맺는다.

2. ICT 환경 및 기술 변화

이 장에서는 최근 ICT 환경 및 기술변화를 다음의 3 가지 주제를 중심으로 설명하고 이들 간의 주요한 상관관계를 살펴본다.

- 데이터 홍수(Data deluge)
- 멀티코어 및 GPGPU(General Purpose GPU)
- 클라우드 컴퓨팅

데이터 홍수는 빅데이터란 기술용어로서 표현되고 있으며, 이는 조직 내부에서 직접적으로 생성되는 데이터 및 M2M 기반의 센서 데이터, Web 2.0 시대에 따른 모바일 장치와 SNS(Social Networking Service) 이용 증가로 데이터 양이 급증하면서 이슈화되고 있다.

위키피디아의 정의에 따르면, 빅데이터는 기존의 데이터베이스 도구의 데이터 수집, 저장, 관리 및 분석 처리 능력을 넘어서는 데이터 셋(data set)으로 데이터의 처리 능력에 따라 빅데이터의 한계 차이를 가질 수 있다. 빅데이터의 주요사례는 다음과 같다.

- Facebook 의 사용자는 2008 년 8 월 1 억 명의 사용자에서 2013 년 3 월에는 11 억 명으로 급속히 증가하고 있음
- Twitter 의 경우 2013 년 일일 약 4 억 개의 트윗(twit) 메시지가 생성되고 있으며, 이는 2008 년 트윗 메시지가 1 억 개 생성된 것과 비교할 때 매우 빠른 데이터 증가를 보여주고 있음
- 천문학, 환경공학, 분자물리학 등에서도 바이오 정보와 유사한 다양한 종류의 방대한 데이터가 생성되어 제공하고 있으며, LHC(Large Hadron Collider)에서 생성된 데이터는 매년 15 페타바이트로 전 세계 약 8,000 명의 연구자에 의해 분석되고 있음

CPU 성능은 고속의 내부 버스 및 멀티코어 프로세서의 등장으로 처리속도가 빨라지고 있다. 멀티코어는 다수의 코어를 사용하여 CPU 성능을 개선하는 방식으로, CPU 는 클럭 속도를 높이는 방법 대신 코어 수를 증가시키는 구조로 변경되고 있다. 멀티코어를 사용한 서버는 가상화(virtualization) 기술로 인해 하나의 물리 서버에 하나 이상의 가상 머신(virtual machine)을 생성할 수 있으며, 이로 인해 비용 측면에서 컴퓨팅 자원을 효율적으로 관리할 수 있다. GPU(Graphics Processing Unit)는 대규모 계산처리를 위해 특화된 멀티 쓰레드 및 매니코어(many-core) 기반의 병렬처리 프로세서이다. 최근 GPU 는 수십에서 수백 개의 코어를 내장하고 있고 CPU 보다 저렴하여 다양한 계산응용 연구에 활용되고 있다. 이러한 범용 응용 처리를 위한 GPU 활용을 GPGPU 라고 한다. GPGPU 는 병렬처리 능력과 속도 개선을 목적으로 동일한 계산 패턴을 다루는 많은 분야에서 사용되고 있다[15]-[17]. 빅데이터 처리를 위해서는 원시 데이터의 지식화 과정이 필요하며, 이 과정에는 대용량 데이터 처리를 위한 계산자원이 요구되고, 계산중심의 데이터 처리를 위해 GPGPU 을 활용한다.

멀티코어 및 GPGPU 기반시스템의 보급에 따라 데이터센터에 기반한 클라우드 컴퓨팅 환경은 인터넷 환경에서 실시간 대용량 데이터 처리를 가능하게 하고 있으며, 또한 과학 데이터 실험 분야에서 빅데이터 처리연구에도 적용되고 있다. 클라우드 컴퓨팅 기술은 하드웨어를 통해서가 아닌 네트워크를 통해 컴퓨팅 환경을 서비스로서 제공하는 환경으로

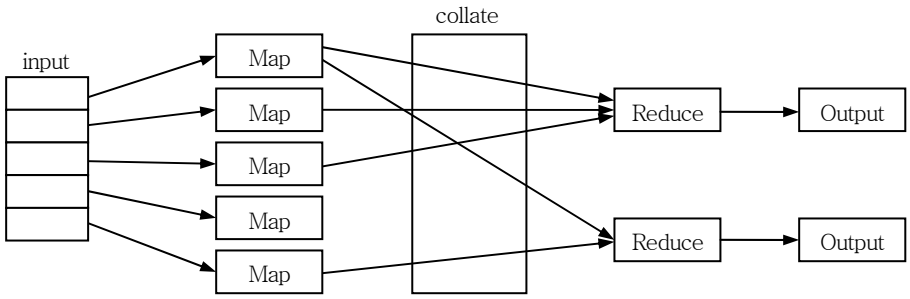
사용자가 어떠한 시스템이나 소프트웨어를 직접 사용할 때 네트워크를 통해 자원을 제공받을 수 있기 때문에 비용 절감 효과 및 IT 인프라의 유연성을 제공한다.

결론적으로 앞서 살펴본 빅데이터, 멀티코어와 GPGPU 의 활용 및 클라우드 컴퓨팅은 서로 밀접한 상관관계를 갖고 있다. 인터넷에 연결된 다양한 센서의 보급으로 IoT (Internet of Things) 환경이 빠르게 구축됨에 따라 데이터는 인터넷에 연결된 장치로부터 수집되고 있다. 현재 50 억 여개의 장치가 인터넷에 연결되어 있는 것으로 추정되고 있으며, 2020 년까지 200 억 개의 장치가 연결될 것으로 예상된다. ICT 기술의 발전에 따라 컴퓨팅 환경의 고성능 및 모바일화는 데이터의 양적 확대를 발생시키고 있으며, 스마트 단말 및 소셜 미디어 등으로 대표되는 다양한 정보채널을 등장시켰다. 클라우드 컴퓨팅 사용자는 경량의 클라이언트 단말을 통해 응용, 스토리지 및 운영체제 등을 서비스로 제공받아 빅데이터 처리를 수행한다. 빅데이터는 ICT 의 주요한 기업의 정책, 생산 및 가치 창출을 위한 동력으로 주요한 키워드가 되고 있으며, 빅데이터를 분산처리하기 위한 프레임워크의 활용은 필수적이다.

3. MapReduce 프로그래밍 모델

분할정복(divide-conquer)은 대용량 데이터 문제를 다루기 위한 접근방법으로 큰 문제를 작은 하위문제로 분할하여 해결한다. 이때 작은 하위 문제들은 독립적으로 수행이 가능하기 때문에 단일 프로세스, 다중 코어 프로세스, 다중 프로세서 또는 클러스터 내의 프로세서의 쓰레드를 통해 병렬 수행할 수 있다[3]. HPC(High Performance Computing) 분야의 응용 개발을 위한 라이브러리인 MPI(Message Passing Interface) 및 PVM (Parallel Virtual Machine)은 분할정복을 지원하며, 이를 위해 낮은 수준의 API 을 제공한다. 그러나 응용 개발자가 이들 API 를 사용할 때 계산자원을 관리해야 하는 제약점을 갖는다. 즉, 응용개발 과정에서 프로그램 개발자는 프로그램 수준에서 메시지 송수신 관련 통신 프리미티브, 네트워크 토폴로지, 상호배제 등의 세부사항을 고려해야 한다.

MapReduce 프로그래밍 모델은 프로그래머로부터 시스템 수준의 세부사항을 숨길 수 있도록 추상화 한다. 이는 주어진 문제를 독립적인 작업으로 병렬 및 분산처리 한다[3], [4]. MapReduce 프로그래밍 모델은 Lisp 및 ML 과 같은 함수 프로그래밍에 기반을 두고 있다. key-value 의 쌍은 MapReduce 의 기본 자료구조를 구성하며, key 와 value 는



<자료>: http://blog.werneckpaiva.com.br/2011/08/como-funciona-o-map-reduce-usado-pelo-google/

(그림 1) MapReduce 처리 흐름

정수, 실수, 문자열, 바이트열 또는 임의의 복잡한 자료구조로서 정의될 수 있다. Map Reduce 프로그램은 Map 과 Reduce 의 함수로 이루어진다.

Map 함수는 주어진 블록의 데이터인 key1, value1 를 입력 받아 이를 가공하고 분류한 후, 새로운 키인 key2 와 새로운 값인 value2 의 쌍인 key2, value2 의 리스트를 생성한다. Reduce 함수는 key2 로 그룹핑(collate)된 값 목록인 list(value2)의 쌍인 key2, list(value2)으로 입력 받아 새로운 key-value 쌍인 key3, value3 의 리스트를 출력한다. 이 과정에서는 중간 값인 key2 에 대한 shuffle 과 sort 를 통해 그룹핑한다. (그림 1)은 MapReduce 처리흐름을 보인 것으로 입력 데이터는 MapReduce 작업을 위해 특정한 크기의 작은 블록으로 파티션한 후 특정한 크기로 나누어진 데이터 블록을 배포한다. 이후 Map 함수와 Reduce 함수의 수행을 위한 Mapper 태스크와 Reducer 태스크가 독립적으로 시작하며, Mapper 태스크와 Reducer 태스크는 데이터를 병렬적으로 처리한다.

여기서는 미국의 기상 데이터센터인 NCDC(http://www.ncdc.noaa.gov/)에서 제공하는 데이터 셋을 활용하여 MapReduce 응용을 구성한 예를 간략히 설명한다. 사용한 데이터 셋은 1950 년에서 1990 년까지 미국 내에 발생된 토네이도에 대한 기상 데이터로서 라인단위로 레코드가 구성되며 ASCII 포맷의 텍스트 자료형식으로 표현된다.

Algorithm 1 은 Map 함수에 대한 기능을 기술한 것으로 라인 단위로 입력파일을 읽는다. Map 함수의 입력 파일은 실행 전에 저장되어야 한다. Map 함수는 filter 함수를 호출하여 입력파일의 오프셋을 key1 의 값으로, 입력된 라인을 value1 으로 입력 받으며, value1 으로부터 연도와 발생빈도(occurrence)를 추출하여 연도를 새로운 키인 key2 로, 발생빈도를 value2 로 지정한 후 새로운 key-value 쌍의 리스트를 생성한다. key2, value2 의

MapReduce 프레임워크 내부에서 연도 값을 갖는 key2 를 사용하여 shuffle 과 sort 를 통해 그룹핑한다. 이 Map 함수를 수행하는 Map 태스크 수는 독립적으로 병렬 수행한다.

```
Algorithm 1: Map()
<key> : line offset within the input file
<value> : line itself from the input file
// A line represents a record
// Extract year and a subset of variables in the record
 (year, occurrence)     filter(line)
Emit(year, occurrence)
```

MapReduce 프레임워크는 동일한 키를 갖는 모든 값을 Reduce 함수에 전달하도록 한다. Reduce 함수는 특정 키를 갖는 값들의 리스트인 key2, list(value2)로서 입력 받는다. 본 예에서는 특정 연도가 key2 로 표현되므로 발생빈도 리스트인 list(value2) 값의 합계를 구한다. 이후 새로운 key3 인 연도와 토네이도 발생빈도의 합계인 value3 가 생성한다. Algorithm 2 는 Reduce 함수에 대한 기능을 기술한 것이다.

```
Algorithm 2: Reduce()
<key> : year
<value> : iterator over the list of tornado occurrences
// Extract year and a subset of variables in the record
Emit(key, sum([list of occurrences])
```

4. Hadoop

이 장에서는 하둡(Hadoop)의 아키텍처 상의 특징을 MapRedue 처리 계층과 분산 파일시스템인 HDFS(Hadoop Distributed File System) 계층으로 나누어 설명하고 하둡 프레임워크의 문제점과 하둡 응용의 제약점을 기술한다.

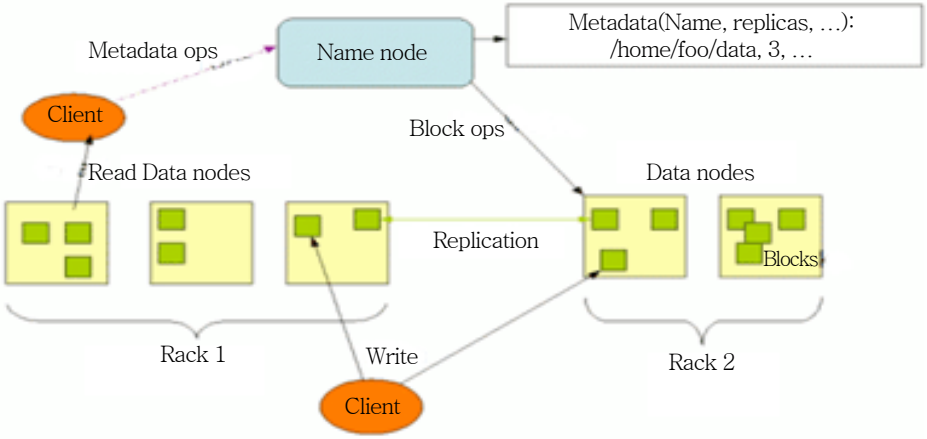
하둡[5]은 Apache 그룹의 MapReduce 기반 오픈소스 소프트웨어 기반의 미들웨어로서 Yahoo, Facebook, Amazon, IBM, NexR 등 많은 기업들에서 클라우드 컴퓨팅 플랫폼으로 활용되고 있다. 하둡은 크게 분산 프로그래밍 모델인 MapReduce 와 분산 파일 시스템인 HDFS 의 두 가지 구성요소로 이루어진다.

하둡의 MapReduce 응용에서 클라이언트가 관리하는 작업단위는 잡(Job)으로, 잡은 입력 데이터, MapReduce 프로그램 및 수행을 위한 설정 정보로 구성된다. 잡의 5 pt MapReduce 프로그램은 Map 태스크와 Reduce 태스크로 나누어 실행된다. 잡 실행 과정

의 제어를 위해 하나의 잡트래커(job tracker)와 Map 과 Reduce 태스크를 위해 태스크트래커(task tracker)가 사용된다. 잡트래커는 태스크트래커들이 수행할 태스크를 스케줄링 함으로써 시스템 전체에서 모든 잡이 수행되도록 조정한다. 태스크트래커는 잡트래커의 요청에 따라 Map 태스크와 Reduce 태스크를 생성하여 태스크를 수행하며, 처리결과를 잡트래커에 전달한다. 이때 태스크가 실패하면, 잡트래커는 해당 태스크를 다른 태스크 트래커에 생성한 후 처리하도록 재스케줄링 한다.

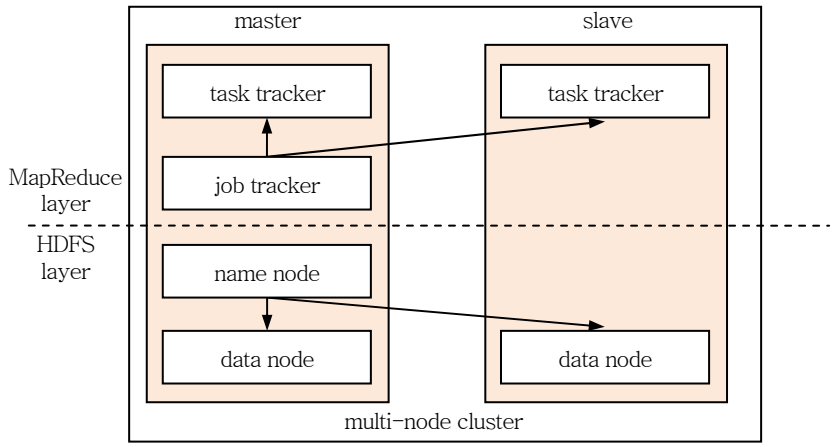
HDFS 는 하둡 기반 MapReduce 응용처리에 필요한 대용량 데이터를 저장할 수 있는 분산 파일 시스템이다. 수천 대 규모의 저가 서버 클러스터를 묶어 단일 파일 시스템 이미지를 제공하는 높은 확장성과 데이터 안정성을 보장하기 위해 최소 세 개의 복제본(replica)을 유지한다. 클라이언트는 데이터 블록을 네임노드(Namenode)에 요청한다. 네임노드는 파일을 구성하는 데이터 블록의 메타데이터를 유지하며 클라이언트에 블록의 위치정보를 제공한다. 이후 클라이언트는 파일에 대한 실제 연산을 수행하기 위해 특정 데이터노드(Datanode)에 접근하여 처리한다. (그림 2)은 HDFS 에서 네임노드와 데이터노드 간의 상호작용을 보인 것이다.

하둡 클러스터 구성은 네임노드와 잡트래커가 통합된 마스터 노드와 태스크 트래커와 데이터노드의 슬레이브노드로 나누어진다. 하둡에서 마스터와 슬레이브노드 사이의 제어정보는 RPC(Remote Procedure Call) 프로토콜이 전송을 위해 사용되며, 마스터와 클라이언트 사이의 통신 역시 RPC 가 사용된다. 그리고 데이터노드와 클라이언트는 TCP 소



<자료>: http://hadoop.apache.org/docs/r2.3.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

(그림 2) HDFS 아키텍처



<자료>: http://en.wikipedia.org/wiki/Apache_Hadoop

(그림 3) Hadoop 아키텍처 구조

켓을 통해 데이터가 전달된다. 마지막으로 MapReduce 과정에서 태스크트래커는 Map 태스크의 결과를 Reduce 태스크로 전달한다. (그림 3)은 하둡의 전체 아키텍처를 보인 것이다.

하둡은 단일 입력과 단일 출력을 갖는 알고리즘을 병렬 처리하기 위한 목적으로 고안되었다. 그러나 복잡한 알고리즘은 MapReduce 작업으로 구성하기 용이하지 않으며 반복적인 데이터 처리에 있어서 성능 상의 제약사항을 갖는다. MapReduce 모델의 특성상 MapReduce 작업은 모든 reduce 태스크와 전역적으로 동기화 되어야 하며, 이를 위해 단일 Reduce 태스크 단계의 수행을 필요로 한다. 또한 여러 개의 MapReduce 작업으로 구성된 경우 Map 과 Reduce 는 이전 단계가 종료되기 전에는 다음단계의 작업을 처리할 수 없는 제약점을 갖는다.

하둡의 HDFS 는 결함 포용 및 확장성을 지원하기 위해 네임노드와 데이터노드 간에 주기적인 메시지 전달이 발생하며, 네임노드는 전달된 메시지를 통해 정상적으로 동작함을 확인한다. 네임노드는 데이터노드로부터 전달된 메시지 내용 중 블록에 대한 정보를 통해 데이터노드의 블록을 확인하고 복제본의 위치를 결정한다. 이는 빈번한 디스크의 데이터 저장에 따른 I/O 및 네트워크 대역폭의 사용을 발생시킨다.

대부분의 과학 데이터 실험에 사용되는 알고리즘은 반복 연산의 수행을 필요로 한다. 하둡 기반의 MapReduce 는 이러한 반복을 필요로 하는 루프 연산에 적합하지 못하며, MapReduce 작업에서 작업의 실행과 종료를 위한 상태 정보를 유지하지 못함으로써 중간

결과 데이터의 처리를 위한 I/O 의 제약 요소로 작용한다.

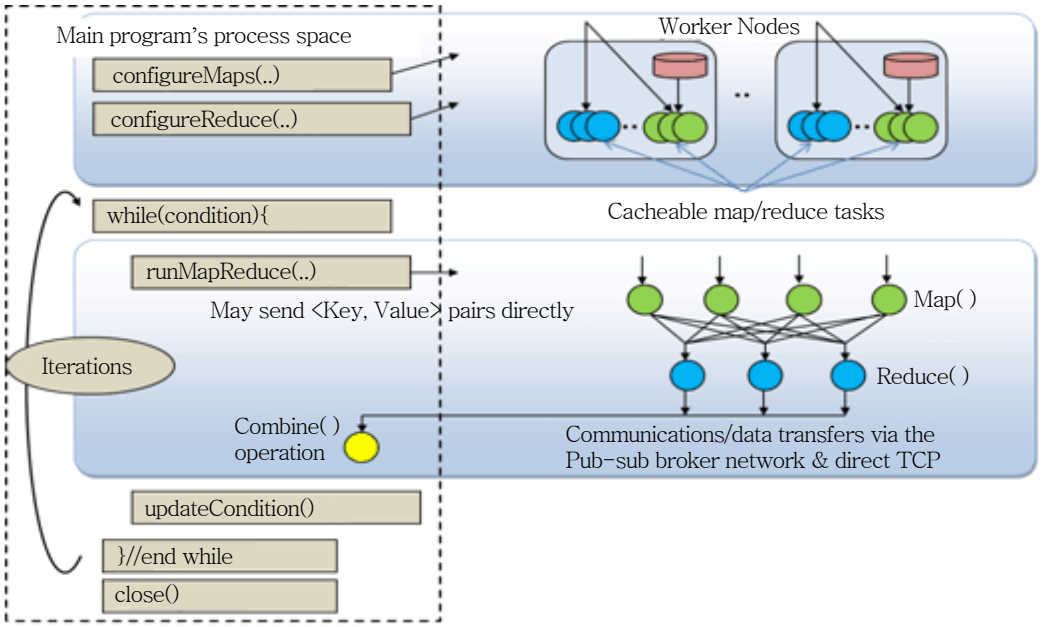
이러한 반복연산의 문제점을 해결하기 위해 개발된 시스템으로는 Twister, HaLoop, Pregel 및 Hama 등이 있다. Twister 와 HaLoop 은 반복연산처리에서 필요한 정적 자료를 캐싱하여 MapReduce 반복에서의 I/O 성능을 개선하였다. Pregel 은 다수의 반복을 필요로 하는 그래프 데이터 처리를 위해 Google 에서 설계하였다. Pregel 은 BSP(Bulk Synchronization Processing) 모델을 기반으로 개발되어 링크분석에 사용하고 있다. 이 과정에서 각 노드는 데이터를 입력 받아 처리한 후 이를 다음 반복에서 데이터가 필요한 노드에 전달한다. Hama 는 Apache 그룹에 의해 진행되고 있는 프로젝트로서 Pregel 의 기능을 오픈소스로 구현하였다.

5. Twister

빅데이터 처리를 위한 기계학습(machine learning) 및 데이터 마이닝(data mining) 분야의 많은 응용에 사용되는 알고리즘은 데이터 처리과정에서 반복적인 계산을 필요로 한다. 그러나 하둡은 반복적인 계산을 위한 과정을 처리하기 위한 응용 수행 시, 태스크 생성을 포함한 초기화, 정적인 데이터의 재로딩 및 Map 과 Reduce 태스크 간의 통신 및 데이터 전송에 오버헤드를 갖는다. 이러한 문제를 해결하기 위해 반복적인 MapReduce 처리를 위한 분산데이터 처리 프레임워크에 대한 필요성이 증가하고 있으며, Twister 는 대표적인 오픈소스 기반 MapReduce 프레임워크이다.

Twister 는 자바로 구현된 MapReduce 프레임워크로서 프로그래밍 모델 측면에서 map 과 reduce 태스크의 구성 및 reduce 출력을 결합할 수 있는 combine 단계 기능을 제공한다[18]. Twister 는 클라이언트가 작업 수행에 명시한 형상정보를 기반으로 단일 또는 여러 횟수의 반복을 수행할 수 있도록 제공한다. 예로 map 과 reduce 작업의 단일 반복만을 요구하는 경우 작업은 일반적인 MapReduce 와 같이 수행을 종료하게 된다. 그러나 MapReduce 를 반복해서 수행하는 경우, reduce 단계의 결과는 각 반복의 마지막 단계인 combine 메소드에 의해 수집되며, Twister 클라이언트는 MapReduce 태스크의 다음 반복에서 key-value 쌍을 계산에 다시 입력한다.

(그림 4)는 Twister 에서 수행되는 응용의 수행 모델을 보인 것으로 응용에서는 while 반복을 통해 MapRduce 호출을 수행하며, 이때 runMapReduce()가 수행된다. RunMap

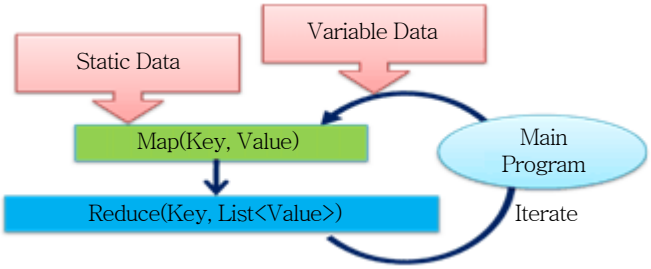


<자료>: http://www.iterativemapreduce.org/

(그림 4) Twister 응용 수행 모델

Reduce 는 수행과정에서 map 태스크인 Map()의 수행결과는 pub-sub 메시징 엔진을 통해 reduce 태스크인 Reduce()에 전달한다. 응용 프로그램은 MapRudce 응용의 작업 흐름을 제어하며, configureMap()과 configureReduce() 메소드는 map 태스크와 reduce 태스크를 작업자 노드에 배치한다.

Twister 의 응용은 작업 수행에 명시한 형상정보를 기반으로 단일 또는 여러 횟수의 반복을 수행할 수 있도록 제공한다. 예로 map 과 reduce 작업이 1 회 수행하는 작업은 기존의 MapReduce 와 같이 수행한다. 그러나 MapReduce 를 반복 수행하는 경우 reduce



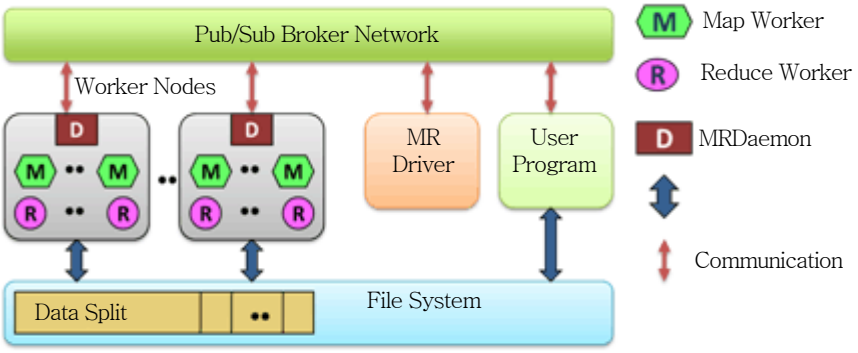
<자료>: http://www.iterativemapreduce.org/

(그림 5) 반복수행에서 정적 정보 캐싱 및 재사용

단계의 결과는 각 반복의 마지막 단계인 combine 메소드에 의해 수집되게 되며, Twister의 사용자 프로그램은 key-value 쌍을 MapReduce 태스크의 다음 반복에서 입력한다. Twister 는 루프 수행 동안 변화되지 않는 정적 데이터(static data)와 변경되는 데이터(variable data)를 분리한 후 정적 정보를 캐싱하는 기능을 제공한다. 이는 동일한 map 과 reduce 태스크가 반복해서 수행하는 작업의 성능을 개선하였다. (그림 5)는 반복수행 과정에서 정적 정보를 캐싱하여 재사용함을 보인 것이다.

Twister 는 pub/sub 브로커 네트워크를 구성하여 데이터를 전송하며, 초기 Twister 버전의 pub/sub 브로커 네트워크는 Narada Brokering 을 사용하여 구현하였다. Narada Brokering 은 인디애나 대학에서 개발된 메시징 미들웨어로서 확장성, 로드밸런싱, 결함처리기능을 제공하며, Twister 기반 MapReduce 응용의 데이터 배포를 위한 미들웨어 구성요소에 독립적이고 협력적인 서비스 제공자의 역할을 담당하는 하부구조이다[19]. 이는 통지(notification)를 통한 비동기적인 통신 메커니즘을 제공함으로써 응용 구성요소 간에 낮은 결합도(loosed coupling)를 제공하는 장점이 있다[13].

Twister 는 논리적으로 마스터 노드와 작업자 노드로 구성되며, 이들 노드에서는 클라이언트인 User Program, 데몬인 MRDaemon 및 작업 태스크인 Map Worker 와 Reduce Worker 의 구성요소가 수행된다. 작업 태스크는 map 태스크와 reduce 태스크로 나누어진다. 클라이언트는 사용자 프로그램과 MapReduce 수행을 위한 드라이버인 MRDriver 로 이루어지며, 데몬은 작업자 노드에서 수행되고, map 태스크와 reduce 태스크의 생명주기를 관리한다. 작업자 노드의 태스크 수행을 위한 제어 메시지와 중간 처리 결과인



<자료>: http://www.iterativemapreduce.org/

(그림 6) Twister 런타임 아키텍처

key-value 쌍의 데이터 메시지는 기 정의된 인터페이스 집합을 사용하여 pub/sub 브로커 네트워크를 통해 데몬과 MapReduce 드라이버인 MRDriver 에 전달된다. (그림 6)은 Twister 의 런타임 아키텍처를 보인 것이다.

6. 결론

본 고에서는 ICT 환경 및 기술 동향을 살펴본 후 빅데이터 문제와 응용에서의 빅데이터 활용을 위해 등장한 기술인 MapReduce 프로그래밍 모델과 MapReduce 기반의 오픈소스 기반 분산 데이터 처리 프레임워크인 Hadoop 과 Twister 를 소개하였다. 데이터 홍수로 인한 빅데이터는 기업 내부 및 외부의 데이터, M2M 기반의 센서 데이터 및 SNS 이용 증가로 인해 모든 분야에서 방대해지고 있으며, FutureGrid 와 같은 과학기술 분야 프로젝트에서도 빅데이터가 주요한 해결과제임을 확인할 수 있다. ICT 환경 및 기술 변화에 따라 CPU 성능 개선과 멀티코어 및 고속 계산처리를 위해 GPGPU 기반 시스템의 보급이 이루어짐을 보았다. 멀티코어 및 GPGPU 기반 시스템의 보급은 데이터 센터 기반의 클라우드 컴퓨팅 환경을 활용한 빅데이터 처리를 위한 서비스 기반의 하부구조로서 비용 측면 및 인프라 유연성을 제공하고 있다.

MapReduce 프로그래밍 모델은 소프트웨어 측면에서 빅데이터의 처리를 위한 병렬 및 분산처리를 위한 이론적 배경으로 도입되었다. MapReduce 프로그래밍 모델은 오픈소스 기반 분산 데이터 처리 프레임워크로 개발되었으며, 대표적인 프레임워크인 하둡의 구성 및 제약사항을 살펴보고, 이를 해결하기 위한 Twister 의 시스템 아키텍처 구성 및 특징을 살펴보았다. 하둡은 빅데이터 처리를 위한 많은 응용의 알고리즘이 반복적인 계산을 요구하지만 아키텍처 측면에서 반복적인 MapReduce 연산에 적합하지 못하며, HDFS 의 유지에 필요한 추가적인 I/O 및 네트워드 오버헤드에 대한 해결이 필요하다. 이를 해결하기 위해 과학기술 응용에 적합하도록 개발된 Twister 는 반복 연산을 지원하기 위해 MapReduce 모델을 확장하였으며, 데이터 전송을 위해 pub/sub 기반의 메시징 미들웨어를 사용한다.

<참 고 문 헌>

[1] Guo, Z.(G.), R. Singh, and M. Pierce, “Building the PolarGrid Portal Using Web 2.0 and OpenSocial,” The International Conference for High Performance Computing, Networking, Storage and Analysis(SC’09), Porland, OR, ACM Press, 2009. 11. 20, p.5.

[2] Hayden, L., G. C. Fox, and A. Adade, “Implementing Cyberinfrastructure in Support of Greenland and Antarctic SAR Data Sets,” 7th International Conference of the African Association of Remote Sensing of the Environment(AARSE)-2008, Accra, Ghana, 2008. 10. 27.
[3] J. Dean and S. Ghemawat, “Mapreduce: Simplified Data Processing on Large Clusters,” Communications of the Acm, Vol.51, 2008.Jan., pp. 107-113.
[4] J. Ekanayake, et al., “MapReduce for Data Intensive Scientific Analyses,” the 2008 Fourth IEEE International Conference on eScience 2008.
[5] Hadoop. .http://hadoop.apache.org/
[6] FutureGrid.,FututrGrid Portal. Available: https://portal.futuregrid.org/
[7] Yunhee Kang, Heeyeoul Choi, An Empirical Study for Handling Scientific Datasets, International Journal of Grid and Distributed computing, Vol.5, No.3, 2012.
[8] Yunhee Kang, Heeyeoul Choi, MapReduce based Scientific Data Experiment Framework for Transforming Data Set, NGCIT 2012,
[9] Yunhee Kang, Geoffrey C. Fox, Performance Analysis of Twister based MapReduce Application on Virtualization System in the FutureGrid, Information, Vol. 7, No. 3, March 2014, pp.951-956.
[10]강윤희, FutureGrid 환경에서의 가상머신 모니터링 도구, 한국정보기술학회 논문지, Vol.9 No.5, 2011, pp.185-191.
[11] 강윤희, FutureGrid 클라우드 컴퓨팅 환경에서의 Twister 수행 MapReduce 응용 구성, 한국정보기술학회 논문지, Vol.9 No.4, 2011, pp.147-154.
[12] 강윤희, 궁상환, 강 경우, 김백민, 장행진, 유성윤, 앙상블 기후 시뮬레이션 지원을 위한 데이터팜 시스템 설계, 한국정보기술학회 논문지, 제 11 권, 제 1 호, 2013, pp.159-167.
[13] 강윤희, u-City 시스템의 통합관리를 위한 메시징기반 소프트웨어 아키텍처, 제 10 권, 제 7 호, 2012, pp.171-177.
[14] 강윤희, 강명주, 고완기, MapReduce 기반 센싱데이터 처리에 관한 연구, 2012 년도 융복합지식학회 하계종합학술발표회 논문집, 2012.
[15] S. Ryoo, C. Rodrigues, et al., Optimization principles and application performance of a multithreaded GPU using CUDA, Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, 2008, pp.73-82.
[16] Kamran Karimi, Neil G. Dickson, and FirasHamze, A Performance Comparison of CUDA and OpenCL, CoRR, May 2010.
[17] Neil G. Dickson, Kamran Karimi, FirasHamze, Importance of Explicit Vectorization for CPU and GPU Software Performance, Journal of Computational Physics, Volume 230, Issue 13, 2011.
[18] J. Ekanayake, et al., “Twister: A Runtime for Iterative MapReduce,” the First International Workshop on MapReduce and its Applications(MAPREDUCE'10)-HPDC2010, 2010.
[19] S. Pallickara and G. Fox, “NaradaBrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids,” the Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Rio de Janeiro, Brazil, 2003.