DOI : 10.5392/JKCA.10.10.078

데이터 분배 및 태스크 진행 스케쥴링을 통한 맵/리듀스 모델의 성능 향상

Improving the Map/Reduce Model through Data Distribution and Task Progress Scheduling

황인성*, 정경용**, 임기욱***, 이정현****
인하대학교 정보공학과*, 상지대학교 컴퓨터정보공학부**, 선문대학교 컴퓨터정보공학부***,
인하대학교 컴퓨터정보공학부****

In-Sung Hwang(clauzevitz@nate.com)*, Kyung-Yong Chung(kyjung@sangji.ac.kr)**,
Kee-Wook Rim(rim@sunmoon.ac.kr)***, Jung-Hyun Lee(jhlee@inha.ac.kr)****

요약

Map/Reduce 는 최근에 많은 주목을 받고 있는 클라우드 컴퓨팅을 구현하는 프로그래밍 모델이다. 이 모델은 여러 대의 컴퓨터를 이용해서 규모가 큰 데이터를 처리하는 어플리케이션에서 사용된다. 따라서 구성된 컴퓨터들을 효율적으로 사용하기 위해서 데이터를 적당한 크기로 나눈 다음 각각의 컴퓨터에 효율적으로 분배시키는 과정을 결정하는 것이 중요하다. 또한 모델을 구성하고 있는 Map 단계와 Reduce 단계를 실행하는 계획도 성능에 많은 영향을 줄 수 있다. 본 논문에서는 대용량의 데이터를 분리해서 Map 태스크를 실행하는 클라우드 컴퓨팅 노드의 성능과 네트워크의 상태를 고려한 후 각각의 컴퓨팅 노드에게 효율적으로 분배하는 방법을 제안한다. 그리고 Map 단계와 Reduce 단계에서 진행하는 방식을 튜닝하여 Reduce 작업의 처리속도를 향상시켰다. 제안된 방법은 대표적인 두 개의 Map/Reduce 어플리케이션을 이용하여 실험하고 조건에 따라 성능에 어떠한 결과를 미치는지 평가했다.

■ 중심어 : | 맵/리듀스 | 클라우드 컴퓨팅 | 성능 예측 | 하둡 |

Abstract

Map/Reduce is the programing model which can implement the Cloud Computing recently has been noticed. The model operates an application program processing amount of data using a lot of computers. It is important to plan the mechanism of separating the data in proper size and distributing that to a cluster consisted of computing node in efficient for using the computing nodes very well. Besides that, planning a process of Map phases and Reduce phases also influences the performance of Map/Reduce. This paper suggests the effectively distributing scheme that separates a huge data and operates Map task in the considering the performance of computing node and network status. And we make the Reduce task can be processed quickly through the tuning the mechanism of Map and Reduce task operation. Using the two Map/Reduce sample application, we experimented the suggestion and we evaluate suggestion considered it in how impact the Map/Reduce performance.

■ keyword : | Map/Reduce | Cloud Computing | Predict Performance | Hadoop |

* "본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2010-C1090-1031-0004)

접수번호 : #100928-002
접수일자 : 2010년 09월 28일
심사완료일 : 2010년 10월 01일
교신저자 : 황인성, e-mail : clauzevitz@nate.com

I. 서론

클라우드 컴퓨팅은 자원을 공유하여 성능을 향상시킬 수 있다는 점에서 정부와 기업에서 비용절감을 하기 위한 솔루션으로 많은 연구가 진행되고 있다.

대표적으로 Map/Reduce 는 구글에서 제안한 클라우드 컴퓨팅을 구현할 수 있는 프로그래밍 모델로 가장 많이 사용되고 있으며 거대한 규모의 데이터를 처리하는데 있어서 클러스터 환경에서 병렬 처리를 지원한다. 또한 Map 클래스와 Reduce 클래스를 설계하고 Map 메서드와 Reduce 메서드에 데이터를 처리하는 메커니즘을 구현하기가 간단하다[1]. Map/Reduce 를 구현한 Hadoop[9] 은 야후에서 개발된 오픈소스의 프레임워크로 Java나 C++ 등 여러 프로그래밍 언어와 결합해서 어플리케이션을 구현할 수 있고 잘 정리된 API를 제공한다.

Map/Reduce 에 관련하여 대부분의 프로젝트들이 여러 컴퓨팅 노드에게 대용량의 데이터를 효율적으로 분배하는 스케줄링을 연구하고 있다. 이 때 거대한 규모의 입력 데이터를 Map 작업을 하는 컴퓨팅 노드에 분배를 해주는 스케줄링 방법에 따라서 전체 성능에 큰 영향을 준다.

본 논문에서는 입력 데이터의 분배 방식과 Map 과 Reduce 의 태스크를 진행하는 과정의 새로운 스케줄링 방법을 제안한다. 입력 데이터는 Map 작업을 실행하는 TaskTracker 들에게 TaskTracker 의 성능과 네트워크의 상태를 고려해서 데이터를 TaskTracker 에게 분배할 수 있다. 또한 Reduce 의 작업속도를 향상시킬 수 있는 방식도 제안한다. 그리고 실험을 통해서 제안한 방법에 대한 성능을 평가한다.

II. 관련 연구

Map/Reduce 프로그래밍 모델의 성능 향상을 위해서 TaskTracker 에 입력 데이터를 균형적으로 분배하는 방법과 이를 위한 TaskTracker 의 성능 예측에 관련된 연구가 가장 많이 진행되고 있다[2-4]. Map/Reduce 모델에서 처리되는 job 의 종류를 CPU 와 I/O 의 사용자원에 따라서 분류하고 작업의 경과를 예측한 후 CPU 와 I/O 를 효율적으로 사용해서 성능의 향상을 얻는 방법[2]과 Map/Reduce 에서 실행되는 job 의 성능을 예측해서 job 에 자원을 할당하는 것을 스케줄링 하는 방법[3]이 그에 해당한다.

그리고 Map/Reduce 모델의 성능을 향상시키기 위해서 네트워크의 자원적인 측면도 고려되어야 한다. 이를 위해서 Map/Reduce 프로그램을 실행하는 도중에 각각의 TaskTracker 에 있는 처리될 데이터 크기를 고려한 후 데이터를 이동해서 전체 TaskTracker 의 작업량의 균형을 맞추는 방법도 있다[5]. 또한 데이터를 분배하는 과정에서 네트워크 대역폭을 균형적으로 사용하는 방법[6]과 프로세스 중에 나타나는 병목 현상의 원인을 실험을 통해서 분석하는 연구[7]가 진행되고 있다. 멀티 스레딩 시스템과 결합해서 규모가 큰 클러스터에서 Map/Reduce 프로그램을 실행시킬 때 발생하는 네트워크 트래픽의 혼잡을 해결하려는 연구도 진행되고 있다[8].

본 논문에서는 TaskTracker 의 성능과 데이터 소스와 TaskTracker 간의 네트워크 상황을 고려해서 입력 데이터를 TaskTracker 에게 균형적으로 나누어 주는 방법을 연구했다. 그리고 TaskTracker 의 성능과 네트워크 상황을 나타내는 상수 값을 효율적으로 계산하는 방법을 제안했다.

III. Map/Reduce와 Hadoop

[그림 1]은 Map/Reduce 의 데이터 처리 진행과정을 보여주는 그림이다. Map/Reduce 의 작업은 Map 과 Reduce 의 두 가지 단계로 분류된다. Map 단계는 어플리케이션에 사용되는 데이터를 Reduce 단계에서 처리하기 전에 미리 데이터를 제련하는 단계이다.

Map 단계에서는 입력된 데이터를 <key, value> 의 쌍으로 데이터를 정리하고 비정상적인 데이터는 배제시킨다. Reduce 단계에서는 Map 단계에서 생성된 <key, value> 데이터를 이용해서 처리하는 단계이다. Map 단계에서 많은 양의 <key, value> 조합의 데이터가 생성되는데 그 데이터들 중에서 똑 같은 key 를 갖

는 데이터들이 하나의 Reduce 를 실행하는 노드로 Map 단계에서 생성된 데이터를 전송한다. 이 Reduce 단계를 실행하는 노드는 전송 받은 데이터로 사용자가 정의한 작업을 실행한다.

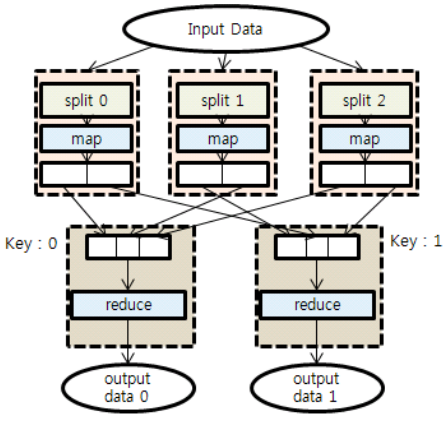

그림 1. Map/Reduce 프로그래밍 모델 구조

Hadoop 에서 작업의 단위는 job 이다. job 은 입력 데이터와 Map/Reduce 프로그램 그리고 환경 설정 정보로 이루어진다. job은 다시 Map task 와 Reduce task 의 task 들로 분류된다. Map/Reduce 프로그래밍 모델은 노드 역할을 하는 컴퓨터들이 모여서 하나의 클러스터를 형성한다. 노드는 JobTrakcer 와 TaskTracker 노드의 두 종류로 구성되고 하나의 JobTrakcer 와 여러 TaskTracker 노드가 뭉쳐서 하나의 클러스터를 형성한다. TaskTracker 는 task 를 실행하는 역할을 하고 JobTrakcer 는 여러 TaskTracker 에 task 할당을 스케쥴링하여 job 을 실행한다.

IV. 데이터 분배 및 진행 디자인

이번 장에서는 TaskTracker 들의 성능을 예측하고 데이터를 분배하는 방법과 Map 작업과 Reduce 작업이 거의 동시에 안정적으로 실행해서 전체적인 어플리케이션의 실행 시간을 단축시키는 방법을 설명한다.

1. 입력 데이터 분배 디자인

데이터를 분배하기 위해서 TaskTracker 에 다시 전달하여 TaskTracker 들 사이의 실행 시간의 균형을 맞추어 주는 방법이 있다. 하지만 이런 방법은 실행 중간에 한번 전송된 데이터를 다시 전송하는 데이터가 많아지면 네트워크에 혼잡이 나타나서 Hadoop 의 전체적인 성능이 떨어질 수 있다. 그러므로 데이터를 분배하기 전에 각 노드의 성능과 상태를 파악해서 적절한 크기의 데이터를 분배하는 방법을 사용해야 한다.

Hadoop 에서는 TaskTracker 들이 입력 데이터를 갖고 있는 소스 컴퓨터들이 네트워크 상으로 가장 근접한 것을 기준으로 정렬된다. 그리고 입력 데이터를 전송할 때 가장 가까운 TaskTracke r에게 먼저 데이터를 전송한다. 본 논문에서는 입력 데이터를 TaskTracker에 전송하기 전에 다음의 세 가지를 고려해서 데이터를 분배한다.

- 데이터 소스 컴퓨터에서 TaskTracker 사이에서 데이터를 전송하는 시간
- TaskTracker 의 큐에 있는 처리할 데이터의 양
- TaskTracker 가 단위 데이터를 처리하는데 필요로 하는 시간

이는 데이터를 전송할 때 데이터 블록들로 이루어진 데이터 그룹마다 id 를 할당하여 그룹을 구분하게 해주며 데이터 소스 컴퓨터에서 TaskTracker로 전송할 때의 시간을 기록한다. 그 후에 TaskTracker가 Map 작업을 다 수행하면 작업 기록과 함께 JobTracker에게 신

표 1. Algorithm 1 MDPR-Sorting algorithm

INPUT:
w /* weight for performance value*/
Ts /*process time of TaskTracker processed a inputdata chunck*/
PList /* List of performace value sordered by ascending*/
$P_s \leftarrow PList.At(S)$
$PList.RemoveAt(S)$
$P_s \leftarrow P_s \times \omega + T_s \times (1-\omega)$
$i \leftarrow 1$
while $i < n$ *do*
  if $P_s \leq PList.Order(i)$ *then*
    $PList.AddOrder(i,\ P_s)$
  end if
  $i \leftarrow i+1$
end while

호를 보내 작업이 종료된 시간을 얻어서 단위 데이터에서 처리량을 계산한다. 이 방법을 통해서 제안한 세 가지의 고려사항을 통합하여 계산하므로 처리량이 줄어든다. 계산하는 방법은 [표 1]의 Algorithm 1을 이용해서 진행한다.

Algorithm 1은 TaskTracker 들 사이에서 성능의 순위를 매기는 알고리즘이다. T_s 는 소스에서 데이터 덩어리를 Map 작업을 하는 TaskTracker 에 전송하는 시간과 TaskTracker 에서 처리하는 시간 그리고 그 결과를 JobTracker 에 보고하는 시간까지 고려하여 만들어진다.

이 때, 단 한번의 TaskTracker 에 대한 기록으로 TaskTracker 의 상태와 성능을 파악하는 것은 정확하지 않은 정보를 바탕으로 데이터를 분배할 확률이 높다. 그래서 TaskTracker 의 이전 성능 측정값을 지우지 않고 현재 측정한 값과 함께 고려하는 가중치 적용 방식을 사용했다. 그리고 ω 값은 실험을 통해 최적의 성능을 보여주는 값을 찾았다.

PList 는 이전까지 Algorithm 1 방식을 이용해서 계산하고 저장된 성능을 나타내는 값들의 리스트 자료구조이다. 이 리스트는 Performance value 가 낮은 숫자부터 오름차순으로 저장되어 있고 TaskTracker 식별자로 값에 접근할 수 있다. 성능 값을 정렬하는데 사용하는 알고리즘은 삽입 정렬 방식을 사용했다. 새로 입력된 값 이외에 리스트에 있는 나머지 값들은 정렬되어 있는 상태이기 때문에 삽입 정렬을 사용하면 이 알고리즘의 수행시간은 HDFS 를 이루는 노드 개수 n 에 대해서 O(n) 이 되어 빠르게 정렬을 할 수 있다.

2. QuickStart Reduce Task 디자인

프로그래밍의 성능의 향상을 위해서 job 스케쥴링의 방식에 변화를 주었다. HDFS 에서 TaskTracker 들이 Map task 단계에서 입력된 데이터를 처리할 때 키 값과 함께 Reduce 에 입력될 데이터가 나타난다. Map 단계가 종료되면 Reduce 단계를 실행하게 되고 Reduce task 를 실행하는 하나의 TaskTracker 가 하나의 key 값을 갖는 데이터를 처리하게 된다. 이 부분에서 Map task 를 진행하는 동안에 Reduce task 도 같이 진행되는 방식을 [그림 2]와 같이 디자인 했다.

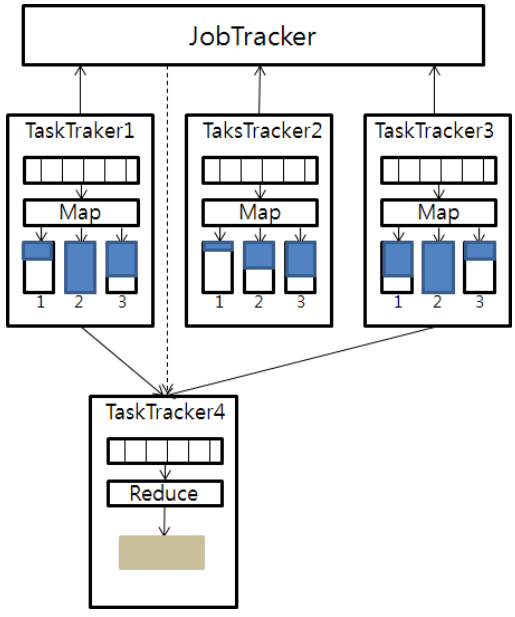

그림 2. QuickStartReduce 디자인 구조

Map task 를 실행하는 TaskTracker 에서 입력 데이터를 처리하는 동안에 키 값들이 나온다. TaskTracker 는 JobTracker 에 정기적으로 HeartBeat 신호를 전송하는데 여기에 TaskTracker에서 나타난 키 들과 해당 키 값에 얼마나 많은 데이터가 처리되었는지 등의 정보를 전송한다. JobTracker 는 이런 HeartBeat 신호를 받고 첨부된 정보를 본다. 이 과정에서 특정 키 값의 데이터가 일정량이 모였다는 정보를 얻으면 해당 키 값의 데이터를 입력 받아서 Reduce task 를 실행 할 수 있는 TaskTracker 를 설정하고 해당 job 의 Map task 를 진행하는 모든 TaskTracker 에게 이 정보를 전송한다. 이 정보를 받은 TaskTracker 는 해당되는 키 값의 데이터가 일정량이 모이면 JobTracker 에게서 받은 메시지의 정보를 확인해서 그 키 값의 데이터를 입력 받고 Reduce 작업을 실행하는 TaskTracker 에게 전달한다. 그 결과 Reduce 의 작업속도를 향상시킬 수 있다.

V. 구현

본 논문에서 제안한 방법을 위해 구현한 모듈은 Measure TaskTracker, Quick Start Reduce 이다. 이 모듈은 Hadoop-0.20.0 에 Plug-in 으로 결합해서 어플리케이션의 성능을 향상시킨다.

1. Estimate Datanode Performance 모듈

Measure TaskTracker 모듈은 HDFS 를 구성하는 TaskTracker 들의 성능을 측정하는 Algorithm 1을 구현한 모듈이다. 데이터 그룹을 분배할 때 네트워크 전송 상태가 양호하고 TaskTracker 자체의 성능도 좋은 노드를 가장 우선 이용해서 어플리케이션 전체의 성능을 향상시키게 만들 수 있다.

2. QuickStart Reduce Task 모듈

HDFS 에서 Map task 를 실행하는 TaskTracker 가 task 를 진행하면서 키 값과 함께 처리된 결과 데이터를 생성시킨다. Quick Start Reduce Task 모듈은 하나의 TaskTracker 가 Map task 를 진행하면서 생성된 어떤 키 값을 갖는 데이터가 일정량 이상 나타나게 되면 JobTracker 에 해당 키 값의 데이터를 이용해서 Reduce task 를 실행하는 TaskTracker 의 생성을 요구하는 메시지 패킷을 전송한다. 메시지 전송하는 방법을 하나의 키 값의 데이터들이 일정량 이상 생기면 바로 JobTracker 에 메시지 패킷을 전송해야 하는지 아니면 정기적으로 JobTracker 에 전송하는 HeartBeat 패킷에 이 정보를 포함해서 전송해야 하는 지는 실험을 통해서 결과를 알아본다.

VI. 실험 및 평가

실험에 사용한 어플리케이션은 WordCount[11] 와 MaxTemperature[10] 이다.

WordCount 어플리케이션은 입력 데이터로 문서 파일들을 받는다. 이 문서 파일들을 TaskTracker 에 분배를 해주면 Map task 를 실행하는 TaskTracker 는 입력 받은 문서 파일에 있는 모든 문자들을 공백 문자를 기준으로 단어를 만드는데 이 단어가 프로그램에서 키 값이 된다. Reduce task 는 이 특정 키 값을 갖는 데이터들만 받고 그 데이터의 개수를 모두 더해서 단어의 개수를 계산한다.

MaxTemperature 어플리케이션은 년도를 키 값으로 갖는 특정한 형식의 레코드로 이루어진 파일을 입력 데이터로 사용한다. Map task 는 받은 파일에 있는 레코드에서 온도 값만 가져온다. 그리고 그 온도 값이 너무 심각한 이상치 값이면 Reduce task 에 넘기기 전에 그 값을 삭제한다. Reduce task 에서는 년도를 키 값으로 갖는 온도 데이터들을 받고 온도 값들을 비교해서 최고의 그 년도에 최고의 온도 값을 찾는다.

또한 다섯 개의 컴퓨터 노드를 이용하고 WordCount 와 MaxTemperature 어플리케이션을 제안한 방법을 포함한 각각의 연구마다 5번씩 실행해서 평균 성능을 확인한다. 그리고 입력하는 데이터의 크기를 변경하면서 실험을 수행하며 이 때 데이터 블록의 크기는 기본인 64MB로 사용한다.

첫 번째 실험은 Quick Start Reduce Task 모듈 방식 실험이다. 이 실험은 하나의 키 값의 데이터가 일정량 이상 모이면 JobTracker 에 바로 패킷을 전달해서 Reduce task 를 수행하는 TaskTracker 를 만드는 방식과 주기적인 HeartBeat 전송에 키 값의 데이터를 포함한 정보를 전달하여 TaskTracker 를 만드는 방식의 성능을 비교했다. 이 실험은 MaxTemperature 어플리케이션을 이용해서 진행되었다. 데이터량을 변화시키면서 제안한 두 가지 방식의 성능 차이를 평가했다.

[그림 3]에 나온 결과에 의하면 특정 키 값의 데이터가 일정량 이상 모였을 때 바로 JobTracker 에 Reduce task 를 실행할 수 있는 TaskTracker 들을 생성하는 방식의 Direct Scheme 가 주기적으로 JobTracker 에 HeartBeat 패킷을 전송하는 Period Scheme 보다 낮은 성능을 보여준다. 이는 특정 키의 데이터가 일정량 모이는 시간이 HeartBeat 신호를 보내는 주기와 거의 비슷하게 진행되고 오히려 Direct Scheme 에서 보내는 패킷이 오히려 중복되는 것이 많이 나타나게 되어 네트워크 자원을 많이 소비해서 나타난 결과이다.

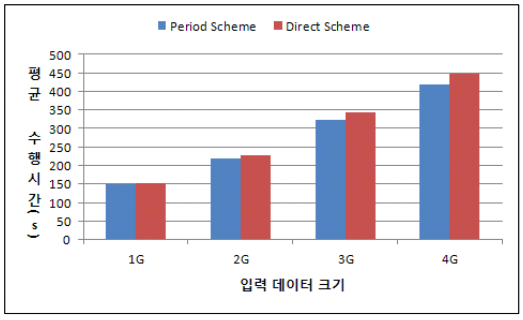

그림 3. Period 방식과 Direct 방식의 성능 비교

두 번째 실험은 Measure TaskTracker 모듈에서 설명한 Algorithm1에서 사용한 가중치 w 값을 변경하면서 어떤 값이 최적의 값인지 실험을 통해서 확인했다.

실험은 가중치 값을 0.2, 0.4, 0.6, 0.8 로 두어서 WordCount 와 MaxTemperature 어플리케이션으로 실험했다. 각 어플리케이션은 입력 데이터의 양은 4GB이다.

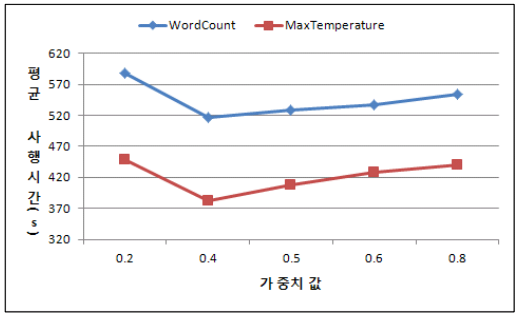

그림 4. QuickStart ReduceTask의 가중치의 변화

[그림 4]의 결과처럼 가중치가 0.4 일 때 전체적으로 가장 빠르게 어플리케이션을 종료할 수 있었다. 가중치가 0.2 인 경우에 실험 할 때마다 성능에서 상대적으로 약간의 차이를 보였다. 그 이유는 가중치가 상대적으로 낮으면 새로 측정된 값만으로 TaskTracker 의 성능을 예측하는데 사용하게 된다. 그 때문에 어플리케이션이 실행하는 도중에 발생하는 네트워크의 오류나 다른 문제로 TaskTracker 의 성능이 큰 차이가 나타나서 안정적이지 못한 상황이 되는 것이다. 실험 결과에 의하면 가중치가 0.4 인 부분에서 가장 좋은 성능이 나타났다.

기본적인 Hadoop 에서는 입력 데이터를 분배할 때 네트워크의 거리를 기준으로 노드들을 정렬한다. 그리고 가장 가까운 노드부터 입력 데이터의 조각을 분배하고 작업이 없는 노드에 먼저 할당하는 방법을 사용한다. 세 번째 실험에서는 이 방식과 논문에서 제안한 두 가지 방식을 결합한 것과 가중치를 0.4 를 사용한 MDPR-Sorting 알고리즘만 적용한 방식과 주기적으로 HeartBeat 패킷을 전송하면서 빠르게 Reduce task 를 실행하는 TaskTracker 설정 방식만 사용해서 각각의 성능을 비교했다.

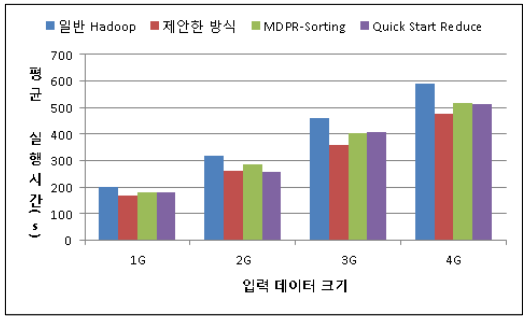

그림 5. WordCount 어플리케이션 평균 수행시간

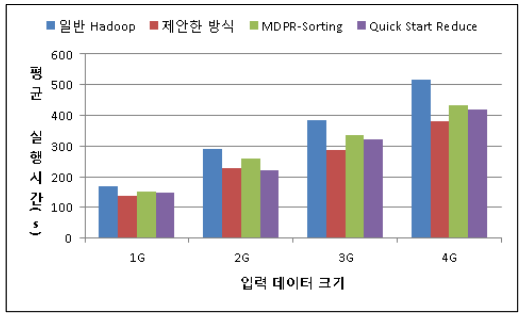

그림 6. MaxTemperature 어플리케이션 성능 측정

[그림 5][그림 6]은 어플리케이션의 실행시간을 측정한 값을 보여준다. 그 결과 많게는 74% 에서 적게는 85% 의 어플리케이션 실행시간을 단축시켰다. 전체적으로 MaxTemperature 어플리케이션에서 상대적으로 더 크게 성능이 향상되었다. 상대적으로 WordCount 에서는 키 값의 종류가 많이 나타났다. 입력한 데이터에 있는 단어들을 포함한 많은 종류의 키 값이 생성되었는

데 키 값의 종류가 너무 많고 그 키 값을 갖는 데이터 숫자가 적어서 Reduce task 를 실행하는 TaskTracker 가 빠르게 생성되지 않은 경우가 많이 나타난 것으로 분석된다. 비록 키 값의 종류가 많고 그 키 값을 갖는 데이터들의 숫자가 작으면 QuickStart Reduce 방식의 효율성은 낮아지지만 하나의 키 값에 대해서 많은 데이터를 갖는 경우라면 좋은 성능을 얻을 수 있다.

VII. 결론 및 향후연구

본 논문에서는 프로그래밍 모델의 성능을 향상시키기 위해서 데이터의 분배와 Reduce task 진행 조건에 따라 변화를 주고 실험을 통해서 제안한 방법에 대한 효율성을 평가했다. 실험결과 제안한 방법이 어플리케이션 동작 과정에서 효율적인 성능을 보였지만 어플리케이션 고유의 특징에 따라서 상대적으로 성능의 차이가 크게 나타났다. 향후 연구에서는 실행하는 어플리케이션이 가질 수 있는 특징들에 대해서 연구하고 각 특징의 어플리케이션 마다 어떤 스케줄링 방식을 적용해야 할 지 결정하는 연구가 필요하다.

참 고 문 헌

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In the Proceedings of the 6th Symposium on Operating Systems Design and Implementation, pp.107-113, 2004.

[2] C. Tian, H. Zhou, Y. He, and L. Zha, "A Dynamic Scheduler for Heterogeneous Workloads," The 8th International Conference on Grid and Cooperative Computing, pp.218-224, 2009.

[3] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguadé, M. Steinder, and I. Whalley, "Performance-Driven Task Co-Scheduling for MapReduce Environments," The 12th IEEE/IFIP Network Operations and Management Symposium, pp.373-380, 2010.

[4] K. Morton, A. Friesen, M. Balazinska, and D. Grossman, "Estimating the Progress of MapReduce Pipelines," 26th IEEE International Conference on Data Engineering, pp.681-684, 2010.

[5] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares and X. Qin, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," The 24th IEEE International Symposium on Parallel & Distributed Processing: Workshops and Phd Forum, pp.1-9, 2010.

[6] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop Distributed Filesystem: Balancing Portability and Performance," The 11th IEEE International Symposium on Performance Analysis of Systems and Software, pp.122-133, 2010.

[7] Z. Vrba, P. Halvorsen, C. Griwodz, and P. Beskow, "Kahn Process Networks are a Flexible Alternative to MapReduce," The 11th IEEE International Conference on High Performance Computing and Communications, pp.154-162, 2009.

[8] S. H. Kang and D. A. Bader, "Large Scale Complex Network Analysis using the Hybrid Combination of a MapReduce cluster and a Highly Multithreaded System," The 24th IEEE International Symposium on Parallel & Distributed Processing: Workshops and Phd Forum, pp.1-8, 2010.

[9] http://lucene.apache.org/hadoop

[10] T. White, Hadoop: *The Definitive Guide,* O'Reilly, 2009.

[11] http://hadoop.apache.org/common/docs/

r0.20.2/ mapred_tutorial.html

저 자 소 개

황 인 성(In-Sung Hwang) 준회원

- 2009년 2월 : 인하대학교 컴퓨터정보공학과(공학사)
- 2009년 3월 ~ 현재 : 인하대학교 정보공학과 석사과정

<관심분야> : 클라우드 컴퓨팅, RFID/유비쿼터스 시스템

정 경 용(Kyuung-Yong Chung) 정회원

- 2000년 2월 : 인하대학교 전자계산공학과(공학사)
- 2002년 2월 : 인하대학교 컴퓨터정보공학과(공학석사)
- 2005년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2005년 9월 ~ 2006년 2월 : 한세대학교 IT학부 교수
- 2006년 3월 ~ 현재 : 상지대학교 컴퓨터정보공학부 교수

<관심분야> : 유비쿼터스 컴퓨팅, 인공지능시스템, 데이터마이닝, U-CRM

임 기 욱(Kee-Wook Rim) 정회원

- 1977년 2월 : 인하대학교 전자공학과(공학사)
- 1987년 2월 : 한양대학교 전자계산학(공학석사)
- 1994년 8월 : 인하대학교 전자계산학(공학박사)
- 1977년 ~ 1988년 : 한국전자통신연구소 시스템소프트웨어 연구실장
- 1989년 10월 ~ 1996년 12월 : 한국전자통신연구원 시스템연구부장, 주전산기(타이컴)Ⅲ,Ⅳ 개발사업 책임자
- 2001년 7월 ~ 1999년 12월 : 한국전자통신연구원 컴퓨터소프트웨어 연구소장
- 2000년 ~ 현재 : 선문대학교 컴퓨터정보학부 교수

<관심분야> : RDBMS, 운영체제, 시스템구조

이 정 현(Jung-Hyun Lee) 정회원

- 1977년 2월 : 인하대학교 전자과(공학사)
- 1980년 9월 : 인하대학교 전자공학과(공학석사)
- 1988년 2월 : 인하대학교 전자공학과(공학박사)
- 1979년 ~ 1981년 : 한국전자기술연구소 연구원
- 1984년 ~ 1989년 : 경기대학교 전자계산학과 교수
- 1989년 1월 ~ 현재 : 인하대학교 컴퓨터공학부 교수

<관심분야> : 자연어처리, HCI, 음성인식, 정보검색, 고성능 컴퓨터구조