http://dx.doi.org/10.5392/JKCA.2013.13.09.020

비-전용 분산 컴퓨팅 환경에서 맵-리듀스 처리 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘

An Efficient Data Replacement Algorithm for Performance Optimization of MapReduce in Non-dedicated Distributed Computing Environments

류은경, 손인국, 박준호, 복경수, 유재수
충북대학교 정보통신공학부

Eunkyung Ryu(lyk1728@hanmail.net), Ingook Son(dlsrnr94422@nate.com),
Junho Park(junhopark@chungbuk.ac.kr), Kyoungsoo Bok(ksbok@chungbuk.ac.kr),
Jaesoo Yoo(yjs@chungbuk.ac.kr)

요약

최근 소셜 미디어의 성장과 모바일 장치와 같은 디지털 기기의 활용이 증가함에 따라 데이터가 기하급수적으로 증가하였다. 이러한 대용량의 데이터를 처리하기 위한 대표적인 프레임워크로 맵-리듀스가 등장하였다. 하지만 전용 분산 컴퓨팅 환경에서 균등한 데이터 배치를 기반으로 수행되는 기존 맵-리듀스는 가용성이 다른 비-전용 분산 컴퓨팅 환경에서는 적합하지 않다. 이러한 비-전용 분산 컴퓨팅 환경을 고려한 데이터 재배치 알고리즘이 제안되었지만, 재배치에 많은 시간을 필요로 하고, 불필요한 데이터 전송에 의한 네트워크 부하가 발생한다. 본 논문에서는 비-전용 분산 컴퓨팅 환경에서 맵-리듀스의 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘을 제안한다. 제안하는 기법에서는 노드의 가용성 분석 모델을 기반으로 노드의 데이터 블록 비율을 연산하고, 기존의 데이터 배치를 고려하여 전송함으로써 네트워크 부하를 감소시킨다. 성능평가 결과, 제안하는 기법이 기존 기법에 비해 성능이 우수함을 확인하였다.

■ 중심어 : | 비-전용 분산 컴퓨팅 | 맵-리듀스 | 하둡 |

Abstract

In recently years, with the growth of social media and the development of mobile devices, the data have been significantly increased. MapReduce is an emerging programming model that processes large amount of data. However, since MapReduce evenly places the data in the dedicated distributed computing environment, it is not suitable to the non-dedicated distributed computing environment. The data replacement algorithms were proposed for performance optimization of MapReduce in the non-dedicated distributed computing environments. However, they spend much time for date replacement and cause the network load for unnecessary data transmission. In this paper, we propose an efficient data replacement algorithm for the performance optimization of MapReduce in the non-dedicated distributed computing environments. The proposed scheme computes the ratio of data blocks in the nodes based on the node availability model and reduces the network load by transmitting the data blocks considering the data placement. Our experimental results show that the proposed scheme outperforms the existing scheme.

■ keyword : | Non-dedicated Distributed Computing | MapReduce | Hadoop |

* 본 논문은 한국콘텐츠학회 2013 춘계 종합학술대회 우수논문입니다.
* 본 연구는 KISTI의 국가슈퍼컴퓨팅서비스 개발 및 기술 연구 과제(K-13-L01-C02)의 지원 및 교육부와 한국연구재단의 지역혁신인력양성사업 지원을 받아 수행된 것임(No.2013H1B8A2032298).
접수일자 : 2013년 07월 03일 심사완료일 : 2013년 08월 19일
수정일자 : 2013년 08월 12일 교신저자 : 유재수, e-mail : yjs@chungbuk.ac.kr

I. 서 론

최근 소셜 미디어의 성장과 모바일 장치와 같은 디지털 기기의 활용에 따라 디지털 정보량이 기하급수적으로 증가하여 기존 데이터 저장 및 분석 시스템의 처리 한계를 넘어서게 되었다[1][2]. 이러한 대용량 데이터를 처리하기 위한 대표적인 프레임워크로써, 하둡(Hadoop)[3]은 대규모 자료의 저장 및 처리를 위한 분산 응용 프로그램을 지원하는 대표적인 오픈소스 소프트웨어이다. 하둡은 페타바이트 이상의 대규모 데이터를 클러스터 환경에서 저장하기 위한 하둡 분산 파일 시스템(HDFS: Hadoop Distributed File System)[4]과 이를 기반으로 병렬 처리를 지원하기 위한 맵-리듀스(MapReduce)[5] 프레임워크로 구성되며, 이러한 맵-리듀스 프레임워크는 전용 분산 컴퓨팅 환경(Dedicated Distributed Computing)[6][7]을 기반으로 수행된다. 전용 분산 컴퓨팅 환경의 경우, 특정 작업을 목적으로 일정한 부분에 한해 활용이 가능한 전용 컴퓨팅 환경이 구축되어야 하므로, 이를 위한 구축비용이 증가하는 문제점이 있다. 이러한 문제점을 고려하여, 특정 목적을 위해 활용되는 전용 컴퓨팅 환경이 아닌, 일반적으로 사용되어지는 비-전용 분산 컴퓨팅(Non-dedicated Computing)[8] 기반의 새로운 데이터 분산 처리 환경이 주목받고 있다. 비-전용 분산 컴퓨팅 환경은 SETI@Home 프로젝트[9]에서와 같이 사용자에 의해 활용 중인 네트워크에 연결된 컴퓨터가 작업하지 않는 시간 동안에 발생하는 유휴 자원이나 작업 중에 발생하는 여유 자원을 활용하여 대규모의 분산 컴퓨팅 환경을 구축하여, 이를 기반으로 분석 작업을 수행하는 것을 말한다. 비-전용 분산 컴퓨팅 환경을 기반으로 데이터의 분석 및 처리를 수행할 경우, 큰 구축비용 없이도 대규모 데이터에 대한 고속 분산 처리 수행이 가능하다는 장점을 가진다.

기본적으로 기존의 전용 분산 컴퓨팅 환경을 기반으로 한 맵-리듀스 프레임워크는 작업의 효율성을 고려하여 저장된 데이터의 편차가 발생하지 않도록 각 노드에 데이터 블록을 균등하게 배치한다. 하지만, 비-전용 분산 컴퓨팅 기반의 환경에서는 클러스터를 구성하는 노드의 다른 작업의 활용 및 결함 발생으로 인한 가용성이 모두 다르므로 기존 맵-리듀스 프레임워크와 같은 균등한 데이터 배치는 적합하지 않다. 이러한 문제점을 고려하여, [10]에서는 노드의 가용성 분석을 통해 비-전용 분산 컴퓨팅 환경에 최적화된 데이터 재배치 알고리즘을 제안함으로써 맵-리듀스의 처리 시간을 감소시켰다. 그러나 [10]은 초기 데이터 배치 상태를 전혀 고려하지 않고 전체 데이터를 재배치함으로써 데이터 재배치 과정에서 불필요한 데이터 전송에 의한 네트워크 부하가 발생하고, 재배치에 많은 시간을 필요로 한다. 이러한 문제점을 해결하기 위해, 본 논문에서는 비-전용 분산 컴퓨팅 환경에서 맵-리듀스의 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘을 제안한다. 제안하는 기법에서는 노드의 가용성 분석 모델을 기반으로 데이터 블록 비율을 연산하고, 기존의 데이터 배치를 고려하여 데이터를 재배치함으로써 네트워크의 부하를 감소시키고 처리 성능을 최적화 하는 것이 가능하다.

본 논문의 구성은 다음과 같다. 제 2장에서는 기존에 제안된 대용량의 데이터 처리기법의 문제점과 연구 목적을 설명한다. 제 3장에서는 제안하는 비-전용 분산 컴퓨팅 환경에서 맵-리듀스 처리 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘을 기술한다. 제 4장에서는 기존 기법과의 성능 평가를 통해 제안하는 기법의 우수성을 보이며, 마지막으로 제 5장에서는 결론과 향후 연구 방향을 제시한다.

II. 관련 연구

대용량 데이터를 처리하기 위한 기법들이 활발하게 연구되고 있다. 이러한 대용량 데이터를 처리하기 위한 대표적인 프레임워크인 하둡은 대량의 자료를 처리할 수 있는 분산 응용 프로그램을 지원한다. 하둡은 구글에서 활용되는 기반 시스템인 구글 파일 시스템[11]을 대체하기 위한 하둡 분산 파일 시스템과 이를 기반으로 데이터를 분산시켜 처리한 뒤 하나로 합치는 기술인 하둡 맵-리듀스를 구현한 기술이다.

맵-리듀스는 맵과 리듀스의 2가지 함수를 조합하여 데이터를 처리한다. 맵은 데이터의 집합을 받아들여 사전에 정해진 처리를 통해 새로운 데이터를 생성하는 프로세스이며, 리듀스는 맵에 의해 만들어진 데이터를 모아 최종적으로 원하는 결과로 만들어 내는 프로세스이다. 일반적으로 맵-리듀스 프레임워크는 전용 분산 컴퓨팅 환경을 기반으로 제안되었다. 따라서 작업의 효율성을 고려해 저장된 데이터의 편차가 발생하지 않도록 각 노드에 데이터 블록을 균등하게 배치한다. 하지만, 최근 이러한 전용 분산 컴퓨팅 환경뿐만 아니라, 비-전용 분산 컴퓨팅 환경에서의 활용이 크게 증가하는 추세이다. 비-전용 분산 컴퓨팅 환경에서는 기존의 전용 분산 컴퓨팅 환경과 달리 클러스터를 구성하는 노드의 다른 작업의 활용 및 결함 발생으로 인한 가용성이 모두 다르므로 기존 맵-리듀스 프레임워크와 같은 균등한 데이터 배치는 적합하지 않다.

비-전용 분산 컴퓨팅 환경에서의 맵-리듀스 최적화를 위한 대표적인 연구로써 [10]은 데이터 가용성 분석 모델을 기반으로 데이터를 재배치하는 기법을 제안하였다. [10]은 각 노드들의 가용성을 분석하여 모델화하고 이를 기반으로 데이터를 재배치함으로써 맵-리듀스의 처리 시간을 단축하였다. 그러나 [10]은 초기 데이터를 고려하지 않고 전체 데이터를 재배치함으로써 데이터 재배치 과정에서 불필요한 데이터 전송에 의한 네트워크 부하가 발생하고, 재배치에 많은 시간을 필요로 한다. 예를 들어, [그림 1]과 같이 초기에 4개의 노드에 데이터 블록이 5개로 균등하게 배치되어 있을 경우, 가용성 분석 모델을 기반으로 각 노드마다 최적 수용 데이터 블록 수가 1번 노드에 3개, 2번 노드에 7개, 3번 노드에 8개, 4번 노드에 2개로 조정된다. 최종적으로 최적 수용 데이터 블록 수만큼 전체 데이터 블록들이 이동한다.

[10]은 가용성 분석모델을 기반으로 데이터 블록 비율을 조정함으로써 비-전용 분산 컴퓨팅환경에서 맵-리듀스의 처리 성능을 최적화하였지만 전체 데이터 블록을 재배치함으로써 불필요한 데이터 전송에 의한 네트워크 부하가 발생하고 재배치에 많은 시간을 소비한다. 그러므로 비-전용 분산 컴퓨팅 환경에서 맵-리듀스의 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘의 연구가 필요하다.

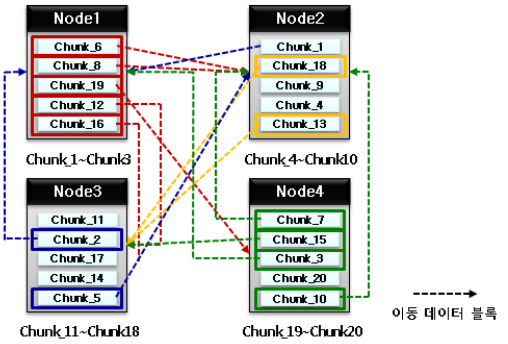

그림 1. 가용성 분석 모델 기반 블록 이동 과정

III. 제안하는 데이터 재배치 알고리즘

3.1 제안하는 시스템의 구조

본 장에서는 비-전용 분산 컴퓨팅 환경에서 맵-리듀스의 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘을 제안한다. 제안하는 기법에서는 노드의 가용성 분석 모델을 기반으로 데이터 블록 비율을 연산하고, 기존의 데이터 배치를 고려하여 데이터를 전송함으로써 네트워크 부하를 감소시킨다. [그림 2]는 제안하는 데이터 재배치 알고리즘을 위한 시스템 구조를 나타낸다.

제안하는 기법의 시스템 구조는 기존 기법의 시스템 구조와 유사하게 하둡 프레임워크의 NameNode와 Client를 수정하였다. NameNode는 Heart-Beat Collector를 통해 노드의 장애시간을 기록하고, *Performance Predictor*를 추가하여 노드의 평균 장애시간을 분석한다. 분석 모델을 기반으로 *Data Block Distributor*는 데이터 블록을 배치한다. 제안하는 알고리즘에서 데이터 재배치의 효율적인 수행을 위해, Client는 기존 데이터의 위치 변경 사항을 파악하는 것이 필요하다. 이를 위해, 제안하는 시스템에서는 하둡 분산 파일 시스템 상에서 새롭게 이동되는 데이터의 위치를 파악하기 위한 쉘 명령어로써 *existingData*를 추

가한다.

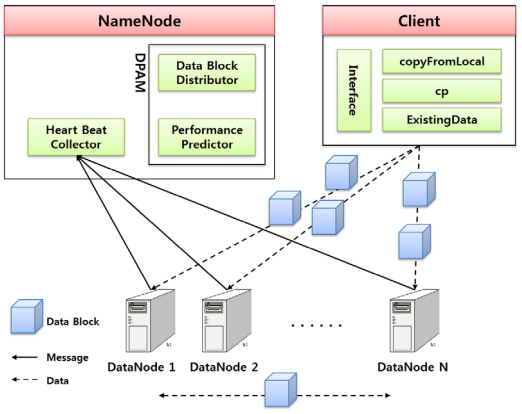

그림 2. 제안하는 시스템 구조

3.2 노드의 가용성 분석 모델

본 논문에서는 기존의 비-전용 분산 컴퓨팅 환경에서의 노드 가용성 분석 모델[10] 기반의 새로운 데이터 재배치 알고리즘을 제안한다. 비-전용 분산 컴퓨팅 환경에서의 가용성 분석 모델은 노드에 발생되는 평균 장애시간을 포함한 태스크의 실행시간을 기반으로 노드의 평균 맵-리듀스의 처리시간을 예측하여 노드의 가용성을 판단한다. 일반적으로 노드에서의 장애 형태는 [그림 3]과 같이 표현된다. 태스크 수행 시간을 기준으로 태스크를 수행하는 도중 결함이 발생하거나 외부의 영향으로 인해 태스크가 중단되는 시간을 장애시간으로 간주한다. MTBI (Meab - Time - between - Interruptions)는 장애시간과 장애시간 사이의 태스크 수행 시간을 의미한다.

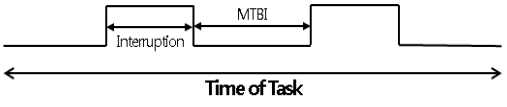

그림 3. 노드에서 발생하는 장애 형태

장애시간을 포함한 단일 태스크 실행시간은 [그림 4]에서와 같이 단일 태스크가 수행되는 동안 장애가 발생하지 않고 완벽하게 태스크가 수행되는 시간이다. 예를 들어, 태스크가 완벽하게 실행되기 위해 필요로 하는 X시간 내에 장애가 발생할 경우, 처음부터 다시 태스크를 수행하며 단일 태스크가 완전히 완료될 때까지 반복한다. 이와 같은 과정을 통해 단일 태스크가 수행되는 시간은 *장애 발생 시간 X, 장애로 인해 재 시작하는 시간 Y, 단일 태스크를 완전히 수행한 시간 γ*의 합과 같다.

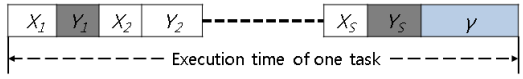

그림 4. 단일 태스크 실행 시간

*단일 태스크의 평균 수행시간(E(T))*은 고장확률 밀도함수와 푸아송 분포를 통해 *평균 장애 시간(E(X))*과 *평균 재시작 시간(E(Y))*, 그리고 *장애가 일어날 평균 횟수(E(S))*를 예측한다. 그러므로 단일 태스크의 평균 수행 시간은 식 (1)과 같이 나타낸다.

$$
\begin{aligned}
E(X) &= \int_{t=0}^{\gamma} t f_x(X=t) = \frac{1}{\lambda} + \frac{\gamma}{1-e^{-\lambda t}} \\
E(Y) &= \frac{\mu}{1-\lambda\mu} \\
E(S) &= \frac{1-e^{-\gamma\lambda}}{e^{-\gamma\lambda}} = e^{\gamma\lambda} - 1 \\
E(T) &= E(E(T|S)) = E(\gamma + SE(Y) + SE(Y)) \\
&= (e^{\gamma\lambda} - 1)(\frac{1}{\lambda} + \frac{\mu}{1-\lambda\mu})
\end{aligned}
\tag{1}
$$

3.3 데이터 재배치 알고리즘

분석 모델을 기반으로 각 노드에 데이터 블록 재배치를 위한 알고리즘을 제안한다. [그림 5]와 같이 첫 번째로 노드별 데이터 블록 수를 결정하기 위해 네임노드의 *Data Block Distributor*는 *BuildHashTable*을 구축한 뒤 *BuildHashTable*을 기반으로 데이터를 재배치한다.

맵-리듀스 프레임워크는 *n*개의 노드에서 *m*개의 입력 데이터 블록을 처리하므로, 성능 최적화를 위한 가용성 분석 모델을 기반으로 노드에서 처리 가능한 데이

터 블록 비율을 연산한다. 식 (2)와 같이 각 노드 i의 *데이터 블록 비율*($Rate_i$)은 노드 i의 *태스크 실행시간* ($E_i(T)$)과 *전체 실행시간*(Φ)을 기반으로 연산되고, 최종적으로 전체 n개의 노드에 m개의 데이터 블록을 분산하기 위해, 식 (3)과 같이 *데이터 블록 비율*($Rate_i$)에 맞게 각 노드 i의 *데이터 블록 수*(w_i)를 연산한다.

$$Rate_i = \frac{1/E_i(T)}{\Phi} \quad (2)$$

$$w_i = m \times Rate_i \quad (3)$$

데이터 블록의 전송을 최소화하기 위해, 기존의 배치된 데이터를 고려하여 노드의 데이터 비율에 적합하도록 최소한의 데이터 블록을 전송한다. 식 (4)와 같이 기존의 데이터 블록수와 노드에 적합한 데이터 블록 수를 비교하여 데이터 블록을 전송해야할 노드와 데이터 블록 수를 연산한다. 식 (4)에서 $Deviation_i$는 기존의 데이터 블록 수와 새롭게 연산된 데이터 블록 수의 편차를 나타낸다. 이 때, pre_w_i는 노드 i에 기존에 배치되어있는 블록 수, w_i는 가용성 분석을 기반으로 새롭게 연산된 블록수를 나타낸다.

$$Deviation_i = pre\_w_i - w_i \quad (4)$$

식 (4)에서 계산된 데이터 블록수의 편차가 $Deviation_i > 0$ 이면, 노드 i에 저장된 데이터 블록 수가 재편성한 데이터 블록수보다 많으므로 다른 노드에 전송해야 한다.

하둡 프레임워크에서는 [그림 6]과 같이 노드의 물리적인 거리에 따른 IP 할당 정책을 활용한다. 예를 들어 '10.0.1.1'과 '10.0.1.2'는 매우 근접한 노드이지만, '10.1.1.1'은 상대적으로 원거리 노드라는 것을 파악하는 것이 가능하다. 하둡 분산 파일 시스템은 이를 고려하여, 가능한 근접 서버와의 연결을 통해 데이터를 송수신을 수행함으로써 네트워크 부하를 감소시킨다.

```
//BuildHashTable
Input: 단일 태스크가 수행되는 시간은 장애 발생 시간 X, 장애로 인해 재 시작하
       는 시간 Y,단일 태스크를 완전히 수행한 시간 γ
output: HashTable → HT(BlockID, CorrespondingNodeIndex)
BEGIN
  식(1)을 이용하여 각 노드의 예상 실행시간 연산 Ei(T)
  FOR EACH node i( 0 ≤ i 〈 n )
    rate_i = 1/E_i(T) / Φ
    w_i = m × rate_i // 노드 I에 할당할 블록 수 연산
    Deviation_i = pre_w_i − w_i // 기존의 블록 수와 재 배치 블록
수의 편차 연산
  END FOR EACH
  FOR EACH node i( 0 ≤ i 〈 n )
    if(Deviationi 〉 0)
      FOR EACH node i( 0 ≤ j 〈 n )
        if(Deviationj 〈 0)
          // 다른 노드와의 거리 연산
          Distance_ij = |Node_IP_i − Node_IP_j|
              //가까운 노드 순으로 arr_i[]배열에 저장
          if(Distanceij 〈 arr_i[num])
            arr_i[num+1] = arr_i[num]
            arr_i[num] = Node_IPj
            num++
          else
            arr_i[num+1] = Node_IPi
            num++
        for(int cnt = 0; cnt 〈 arr_i.size; cnt++;)
          if(Deviationarr_i[cnt] 〈 0)
            // 해시테이블 반환
            return HT(r) = (BlockID, arr_i[cnt]
            Deviationarr_i[cnt] −= deviationi
        END for
    else
      // 해시테이블 반환
     return HT(r) = (BlockID, node i)
  END FOR EACH
END
```

그림 5. Build Hash Table 알고리즘

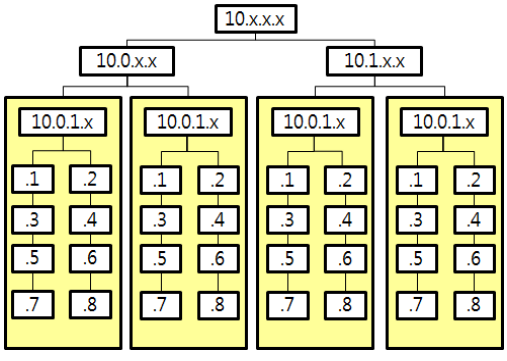

그림 6. 거리에 따른 주소 체계

이러한 주소 할당 기법을 기반으로, 제안하는 기법에서는 노드 i에서 전송할 목적지 노드를 선출하기 위해 식 (5)와 같이 *다른 노드와의 거리*(Distance$_{ij}$)를 연산하여 가까운 노드를 산출한다. 식 (5)에서 Node_IP$_i$와 Node_IP$_j$는 노드 i, j의 실제 IP를 나타낸다.

$$Distance_{ij} = |Node\_IP_i - Node\_IP_j| \quad (5)$$

선출된 노드를 중심으로 $Deviation_j < 0$ 인 노드를 탐색하여 *해시테이블*(HT(r))을 생성한다. 이때, 노드 j에 재배치된 블록의 *해시테이블*(HT(r))이 생성되면 노드 j의 $Deviation_j$는 재배치된 블록 수를 기반으로 갱신된다.

데이터를 재배치하기 위해 *BuildHashTable()*을 기반으로 랜덤 변수 r ($0 \le r < m$)을 생성하여 r의 값을 키 값으로 사용한다. r의 값과 *HT(r)*의 BlockID에 해당하는 블록을 해당하는 노드에 데이터를 재배치함으로써 데이터 블록의 배치를 수행한다.

위와 같이 제안하는 효율적인 데이터 재배치 알고리즘은 노드의 가용성 분석 모델을 기반으로 데이터 블록 비율을 연산하고 기존 데이터 배치를 고려하여 데이터를 재배치함으로써 전송비용을 감소시키고 가까운 노드를 산출하여 데이터 블록을 전송함으로써 네트워크 부하를 감소시키는 것이 가능하다.

Ⅳ. 성능 평가

4.1 실험 환경

본 장에서는 제안하는 기법의 우수성을 보이기 위해 기존 기법[10]과의 시뮬레이션을 통한 성능 비교 평가를 수행하였다. 따라서 본 논문에서는 [10]에서 구축한 시뮬레이션 환경을 기반으로 수행하였다. 본 시뮬레이션은 Java 시뮬레이터를 바탕으로 [표 1]과 같은 성능 평가 환경을 구성하여 수행하였다. 노드의 결함 발생은 임의로 발생시켰으며, 기본적으로 노드에 배치되는 블록 수는 10개로 지정 하였다.

표 1. 성능평가 환경

환경 변수	설정 값
노드의 수(EA)	10 ~ 50
네트워크 대역폭 (MB)	4 ~ 32
데이터 크기(GB)	6.4 ~ 32
데이터 블록 사이즈(MB)	64
노드 당 청크 수(EA)	10

4.2 실험 결과

[그림 7]은 노드 수에 따른 재배치 데이터 블록 비율을 비교 평가한 결과이다. 기존 재배치 알고리즘[10]의 경우 기존의 데이터 배치를 고려하지 않고 재배치함으로써 데이터 블록의 재전송 할 비율이 높아진다. 제안하는 기법은 기존의 데이터 배치를 고려함으로써 데이터 블록의 재전송 비율을 감소시킨다. 성능 평가 결과, 제안하는 기법의 데이터 재배치 블록의 비율이 평균 약 75% 감소하였다.

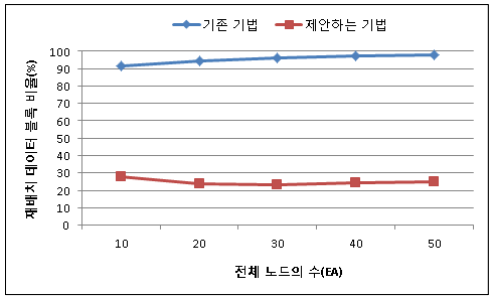

그림 7. 노드 수에 따른 재배치 데이터 블록 비율

[그림 8]은 대역폭에 따른 재배치 데이터 블록의 이동시간을 비교 평가한 결과이다. 노드 10개와 노드 당 청크 수 10개를 배치한 환경에서 실험하였다. 기존 재배치 알고리즘[10]의 경우 기존의 데이터 배치를 고려하지 않고 재배치함으로써 데이터 블록을 재전송 하는데 많은 시간을 필요로 할 뿐 아니라 대부분의 블록들이 이동함으로 인해 네트워크 부하가 높아진다. 제안하는 기법은 기존의 데이터 배치를 고려하여 재배치 데이터 블록을 최소화함으로써 재배치되는 데이터 블록의 이동 시간을 감소시킨다. 성능 평가 결과, 제안하는 기법의 데이터 재배치 블록의 이동시간이 평균 약 49% 감소하였다.

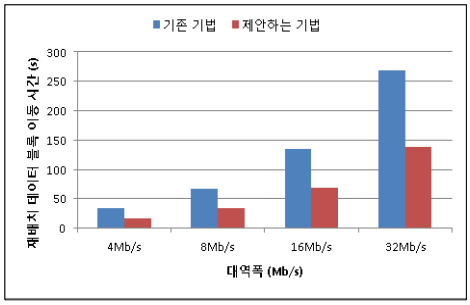

그림 8. 대역폭에 따른 재배치 데이터 블록 이동시간

V. 결론 및 향후 연구

본 논문에서는 비-전용 분산 컴퓨팅 환경에서 맵-리듀스의 성능 최적화를 위한 효율적인 데이터 재배치 알고리즘을 제안하였다. 제안하는 기법에서는 노드의 가용성 분석 모델을 기반으로 데이터 블록 비율을 연산하고, 기존의 데이터 배치를 고려하여 데이터를 전송함으로써 네트워크 부하를 감소시키는 것이 가능하다. 성능 평가 결과, 제안하는 기법의 데이터 재배치 블록 비율이 기존 기법에 비해 약 75%, 데이터 재배치 블록의 이동시간이 약 49% 감소하였다. 이를 통해 제안하는 기법에서 데이터 블록의 재배치를 위한 네트워크의 부하가 감소되는 것을 확인하였다. 향후 연구로는 데이터 재배치 후 맵-리듀스의 처리 속도를 향상시키기 위한 알고리즘을 접목하는 것이다.

참 고 문 헌

[1] J. Dittrich and J. Quiané-Ruiz, *Efficient big data processing in Hadoop MapReduce*, Proc. of the VLDB Endowment, pp.2014-2014, 2012.

[2] I. Hwang, K. Jung, K. Im, and J. Lee, "Improving the Map/Reduce Model through Data Distribution and Task Progress Scheduling," *Journal of the Korea Contents Association*, Vol.10, No.10, pp.78-85, 2010.

[3] http://hadoop.apache.org.

[4] K. Shvachko, H. Huang, S. Radia, and R. Chansler, *The Hadoop Distributed File System*, Proc. of the IEEE Symposium on Massive Storage Systems, pp.1-10, 2010.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Magazine Communications of the ACM, Vol.51, Issue1, pp.107-113, 2008.

[6] D. Werthimer, J. Cobb, M. Lebofsky, D. Anderson, and E. Korpela, "SETI@HOME-Massively Distributed Computing for SETI," Journal of Computing Science and Engineering, Vol.3, No.1, pp.78-83, 2001.

[7] D. L. Eager, E.D. Lazowska, and J. Zahor-jan, "Adaptive Load Sharing in Homogeneous Distributed Systems," Journal of Software Engineering, Vol.12, No.5, pp.662-675, 1986.

[8] S. T. Leutenegger, X. H. Sun, *Distributed Computing Feasibility in a Non-dedicated Homogeneous Distributed System*, Proc. of the ACM/IEEE Conference on Supercomputing, pp.143-152, 1993.

[9] "SETI@home", http://setiathome.berkeley.edu

[10] H. Jin, X. Yang, X. H. Sun, and I. Raicu, *ADAPT: Availability-Aware MapReduce Data Placement for Non-Dedicated Distributed Computing*, Proc. of IEEE International Conference on Distributed Computing Systems, pp.516-525, 2012.

[11] S. Ghemawat, H. Gobioff, and S. Leung, *The Google File System*, Proc. of ACM Symposium on Operating Systems Principles, pp.29-43, 2003.

저 자 소 개

류 은 경(Eunkyung Ryu) 준회원

- 2012년 2월 : 충북대학교 정보통신공학(공학사)
- 2012년 3월 ~ 현재 : 충북대학교 정보통신공학 석사과정

<관심분야> : 센서 네트워크, 빅데이터, 맵-리듀스, 데이터베이스 시스템 등

손 인 국(Ingook Son) 정회원

- 2012년 2월 : 충북대학교 정보통신공학과 공학사.
- 2012년 3월 ~ 현재 : 충북대학교 정보통신공학과 석사과정

<관심분야> : 모바일 네트워크, 무선 센서 네트워크, 위치 기반 서비스, 빅 데이터, 클라우드 컴퓨팅 등

박 준 호(Junho Park) 정회원

- 2008년 2월 : 충북대학교 정보통신공학과(공학사)
- 2010년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2010년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 무선 센서 네트워크, 데이터베이스 시스템, RFID, 차세대 웹, LMS/LCMS, 바이오인포매틱스 등

복 경 수(Kyoungsoo Bok) 정회원

- 1998년 2월 : 충북대학교 수학과(이학사)
- 2000년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2005년 2월 : 충북대학교 정보통신공학과(공학박사)
- 2005년 3월 ~ 2008년 2월 : 한국과학기술원 전산학과 Postdoc
- 2008년 3월 ~ 2011년 2월 : ㈜가인정보기술 연구소
- 2011년 3월 ~ 현재 : 충북대학교 정보통신공학과 초빙부교수

<관심분야> : 데이터베이스 시스템, 자료저장 시스템, 위치기반서비스, 모바일 P2P 네트워크, 센서네트워크 및 RFID 등

유 재 수(Jaesoo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : KAIST 전산학과(공학석사)
- 1995년 2월 : KAIST 전산학과(공학박사)
- 1995년 3월 ~ 1996년 8월 : 목포대학교 전산통계학과(전임강사)
- 1996년 8월 ~ 현재 : 충북대학교 정보통신공학부 및 컴퓨터정보통신연구소 교수
- 2009년 3월 ~ 2010년 2월 : 캘리포니아주립대학교 방문교수

<관심분야> : 데이터베이스시스템, 빅데이터, 센서네트워크 및 RFID, 소셜네트워크서비스, 분산객체컴퓨팅, 바이오인포매틱스 등