Journal of The Korea Society of Computer and Information
Vol. 19, No. 11, November 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.11.009

하둡 클러스터의 네트워크 사용량 감소를 위한 블록 재배치 알고리즘

김 준 상*, 김 창 현*, 이 원 주**, 전 창 호*

A Block Relocation Algorithm for Reducing Network Consumption in Hadoop Cluster

Jun-Sang Kim*, Chang-Hyeon Kim*, Won-Joo Lee**, Chang-Ho Jeon*

요 약

본 논문에서는 하둡 클러스터의 네트워크 사용량 감소를 위한 블록 재배치 알고리즘을 제안한다. 하둡 클러스터의 스케줄러는 사용자들에게 작업을 받아 다중 태스크로 작업을 나누어서 각 노드들에게 할당한다. 이 때 스케줄러는 데이터 지역성(Data locality)을 만족시키는 노드에 우선적으로 태스크를 할당한다. 만약 처리할 데이터(블록)가 없는 노드에 태스크가 할당되면 다른 노드로부터 전송받아 처리한다. 클러스터의 블록들은 사용 빈도가 각각 다르기 때문에 노드 간 작업 부하의 차이가 발생하며 이로 인해 노드 간 데이터 전송이 빈번해진다. 그래서 제안하는 블록 재배치 알고리즘은 하둡 스케줄러의 작업 할당 패턴에 따라 블록들을 균등하게 재배치한다. 결국 노드들의 작업 부하는 평준화 되고 처리할 블록이 없는 노드에서 태스크를 처리하는 경우가 감소하기 때문에 클러스터의 네트워크 트래픽이 감소한다. 시뮬레이션으로 제안하는 블록 재배치 정책의 성능평가를 진행했으며 기본 지연 스케줄링으로 작업을 처리한 경우와 비교하여 최대 23.3%의 네트워크 사용량 감소를 보였다.

▸ Keywords : 하둡 클러스터, 하둡 스케줄러, 블록 재배치

Abstract

In this paper, We propose a block reallocation algorithm for reducing network traffic in Hadoop cluster. The scheduler of Hadoop cluster receives a job from users. And the job is divided into multiple tasks assigned to nodes. At this time, the scheduler allocates the task to the node that satisfied data locality. If a task is assigned to the node that does not have the data(block) to be processed, the task is processed after the data transmission from another node. There is difference of workload among nodes because blocks in cluster have different access frequency. Therefore, the proposed algorithm relocates blocks

•제1저자 : 김준상 •교신저자 : 전창호
•투고일 : 2014. 8. 21, 심사일 : 2014. 9. 16, 게재확정일 : 2014. 10. 13.
* 한양대학교 컴퓨터공학과(Dept. of Computer Science & Engineering, Hanyang University ERICA Campus)
** 인하공업전문대학 컴퓨터정보과(Dept. of Computer Science, Inha Technical College)
※ 이 논문은 2012년 한양대학교 교내연구비 지원으로 연구되었음(HY-2012-G)

according to the task allocation pattern of Hadoop scheduler. Eventually, workload of nodes are leveled, and the case of the task processing in a node that does not have the block to be processing is reduced. Thus, the network traffic of the cluster is also reduced. We evaluate the proposed block reallocation algorithm by a simulation. The simulation result shows maximum 23.3% reduction of network consumption than default delay scheduling for jobs processing.

▸ Keywords : Hadoop Cluster, Hadoop Scheduler, Block Relocation

I. 서 론

정보기술의 발전과 컴퓨팅 및 네트워크 환경의 확대로 도처에서 생성되는 디지털 데이터의 양이 급격하게 증가하고 있다. 이러한 디지털 데이터들을 빅데이터라 하며 비정형적인 특징을 가지고 있다[1]. 최근 업계 및 사회 분야에서 이러한 빅데이터의 활용도가 급증하고 있다. 그래서 많은 기업과 기관에서 빅데이터의 저장과 분석을 위해 하둡 기반의 클라우드 클러스터를 구축하고 있다. 하둡은 클러스터 환경을 기반으로 동작하는 응용 소프트웨어가 대용량의 자료를 처리할 수 있도록 지원하는 분산 처리 플랫폼이다[2-3].

하둡은 클라우드 클러스터가 저장하고 있는 대형 데이터셋의 처리/생성을 구현하기 위해 맵리듀스(MapReduce)[4] 프로그래밍 모델을 사용한다. 맵리듀스의 작업 스케줄러는 기본적으로 FIFO이기 때문에 순차적으로 태스크를 노드들에게 할당한다. 만약 노드에 태스크가 할당 되었을 때 필요한 데이터(블록)가 없으면 다른 노드에서 가져와서 처리한다. 그러므로 클러스터가 처리할 작업의 분량이 많을 때 클러스터 내부의 데이터를 호출하는 경우와 외부에서 호출하는 경우의 비율이 나빠진다. 이런 경우를 데이터 지역성(Data locality)이 저하라고 한다. 원격 데이터 호출이 많아지면 네트워크 사용량이 증가한다. 만약 네트워크가 과부하 상태가 되면 네트워크를 사용하는 다른 서비스와 작업의 처리가 지연된다.

하둡 스케줄러는 초기 배치된 블록들이 시간이 지나도 재배치되지 않는 환경에서 동작한다. 그러므로 작업 처리 시 노드 간 작업 처리 부하의 차이가 발생한다. 사용 빈도가 높은 데이터를 많이 가지고 있는 노드는 항상 동작 상태이고 그렇지 않는 노드는 유휴 노드가 되어 다른 노드에서 데이터를 전송받아 처리한다. 결국 노드들은 태스크들을 적절히 분배하여 작업 처리를 하지만 데이터 전송으로 인한 네트워크 사용량이 증가한다.

본 논문에서는 하둡 클러스터의 네트워크 사용량 감소를 위한 블록 재배치 알고리즘을 제안한다. 제안하는 알고리즘은 하둡 클러스터의 작업 패턴에 따라 노드의 블록들을 재분배한다. 원격 노드의 작업 부하가 낮고 외부에서 받아온 블록의 사용 빈도가 높은 경우 기존 보유하고 있는 블록 중 사용 빈도가 낮은 블록과 교환한다. 그러므로 각 노드들의 작업 부하는 평준화 되고 해당 블록이 없는 유휴 노드에서 태스크를 처리하는 경우가 감소하기 때문에 클러스터의 네트워크 사용량이 감소한다. 본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고 3장에서는 본 논문에서 제안하는 블록 재배치 알고리즘에 대해서 기술한다. 4장에서는 성능평가와 그 결과를 제시하고 5장에서 결론을 내린다.

II. 관련 연구

1. Hadoop Distributed File System(HDFS)

HDFS[5]는 하둡 클러스터에 사용되는 분산 파일 시스템으로 구글의 Google File System(GFS)[6]와 유사한 구조이다. 하둡 클러스터의 서버들은 네임 노드(Name node)와 데이터 노드(Data node)로 구성되어 있다. 데이터 노드는 하둡 클러스터에서 사용/생성되는 데이터들을 저장하고 네임 노드는 데이터 노드들의 메타데이터와 상태를 실시간으로 관리한다. 데이터는 64MB(기본 값) 단위의 블록들로 나누어져 각 데이터 노드들에게 분산 배치된다.

하둡 클러스터는 기본적으로 저가의 서버들을 이용해서 구축된다. 그러므로 HDFS는 노드들의 결함 허용(Fault tolerance)을 위해 기본적으로 3개의 복사본을 생성해서 운

용한다. 서버의 결함 허용을 위해 같은 랙(Rack)에 1개, 네트워크의 결함 허용을 위해 다른 랙에 1개를 생성해서 배치한다.

2. 하둡 내장 스케줄러

기본 하둡 스케줄링 알고리즘[7]은 FIFO 큐를 사용하여 동작한다. 우선 작업들을 요청된 순서대로 개별 태스크들로 분리한 후 노드들의 빈 슬롯들에게 할당한다. 그러므로 나중에 할당된 작업은 나중에 처리되기 때문에 클러스터 사용자들 간 자원 할당이 공정하게 이루어지지는 않는다. 옵션으로 우선 순위 할당도 가능하지만 따로 설정해주어야 한다.

Fair 스케줄러[7]는 페이스북에서 자사의 하둡 클러스터를 관리하기 위해 개발했다. Fair 스케줄러는 모든 사용자들에게 클러스터의 자원을 공정하게 사용하도록 하는 것을 목표로 한다. 사용자들은 작업들을 풀(Pool)들에게 할당하고 각 풀은 작업들이 필요로 하는 자원들이 가용량을 초과했을 때 최소한의 맵과 리듀스 슬롯들을 보장한다. Fair 스케줄러는 선점을 지원해서 풀이 특정 기간 동안 공정한 자원 분배를 받지 못했을 경우 가용 용량 이상의 태스크들을 종료(Kill)시키고 슬롯을 할당한다.

Capacity 스케줄러[7]는 야후에서 개발한 하둡 스케줄러로 사용자들의 수가 많고 사용자들 간 연산 자원의 공정한 배치가 필요한 환경을 염두에 두고 개발했다. Capacity 스케줄러는 태스크들을 배치할 때 사용자를 맵과 리듀스 슬롯들의 수를 조정할 수 있는 큐에 할당한다. 큐의 남는 자원들이 발생 했을 경우 다른 큐에 잠시 빌려줄 수도 있다.

지연 스케줄링[8]은 Fair 스케줄러를 기반으로 데이터 지역성을 높여 하둡 클러스터의 성능을 높이는 스케줄링 기법이다. 기존 스케줄러는 태스크를 노드에 할당할 때 데이터 지역성을 만족시키는 노드에 우선적으로 할당한다. 그 외의 태스크는 유휴 노드에 다른 노드의 데이터를 전송받아 태스크를 처리한다. 그러므로 데이터 전송에 따른 지연과 네트워크 사용량이 증가한다. 지연 스케줄링은 데이터의 전송 지연과 데이터 지역성을 만족하는 노드의 남은 태스크 처리 시간을 비교해서 태스크를 대기시킬지 전송 후 다른 노드에서 처리할지 여부를 결정한다. 그러므로 전반적인 전송 지연이 감소하고 불필요한 데이터 전송이 줄어 네트워크 사용량의 증가폭을 줄인다.

하둡 스케줄러들은 기본적으로 많은 작업을 라운드 로빈 방식으로 분산처리 하기 때문에 네트워크 용량 초과나 Straggler 문제[9]가 발생하지 않는다면 전체 작업 처리 속도는 큰 차이가 없다. 그러므로 스케줄러 성능 향상 관련 연구들은 주로 이 두 문제를 해결하는데 초점을 두고 있다.

3. 스케줄러 성능향상 관련 연구

LATE(Longest Approximate Time to End) 스케줄러[9]는 지연 스케줄링에서 Straggler를 선정 과정을 개선하여 클라우드 클라스터의 성능을 향상시켰다. 기존의 하둡 스케줄러는 Straggler를 선택해서 유휴 노드에 재할당하는데 이를 Speculative task라고 한다. 현재의 하둡은 휴리스틱에 의해 정한 임계값을 넘으면 Speculative task를 실행한다. 그러므로 Speculative task가 너무 과도하게 실행되어 전체 클러스터의 성능을 저하시킨다. LATE 스케줄러는 이 문제를 해결하기 위해서 단순한 임계값 대신 태스크의 남은 시간을 구해서 Speculative task 실행 여부를 결정한다.

LATE는 지연 스케줄링의 Straggler 문제를 개선하였지만 기본적인 스케줄링은 지연 스케줄링을 그대로 사용한다. Speculative task의 감소로 인한 약간의 전송량이 감소하지만 전체적인 네트워크 사용량은 큰 차이가 없다. 그러므로 본 논문에서는 전체 하둡 클러스터의 네트워크 사용량을 감소시키기 위해 하둡 스케줄러와 함께 사용할 수 있는 블록 재배치 알고리즘을 제안한다.

III. 블록 재배치 알고리즘

1. 개요

하둡 클러스터가 보유하고 있는 블록들의 사용빈도는 일정하지 않다. 그러므로 사용 빈도가 높은 블록들을 많이 보유하고 있는 노드는 항상 태스크 점유상태가 되고 그렇지 않는 노드는 유휴상태가 되어 다른 노드의 블록을 가져와서 데이터를 처리하게 되어 결국 네트워크 사용량이 높아지게 된다. 만약 많이 사용되는 블록들을 각 노드들에게 균일하게 배치하면 다른 노드의 블록을 가져오는 빈도가 감소하게 되어 네트워크 사용량을 감소시킬 수 있다.

제안하는 블록 재배치 알고리즘은 데이터 접근 패턴을 반영하여 실시간으로 블록을 재배치한다. 그러므로 데이터 접근 패턴이 변하더라도 각 노드의 부하를 균일하게 유지시켜 블록 이동으로 인한 네트워크 사용량을 줄인다.

2. Load Value

Load Value(LV)는 하둡 클러스터를 구성하는 각 요소들의 작업 부하를 의미하는 수치이다. LV는 클러스터의 각 노드에서 블록의 재배치 여부를 결정하기 위해 사용되며

Chunk Load Value(CLV), Node Load Value(NLV), Rack Load Value(RLV)로 분류된다.

CLV는 블록의 사용 정도를 나타내는 수치이다. CLV의 초기 값은 0이며 해당 블록이 사용되었을 때 기존 CLV에 당시 태스크 처리 횟수를 합산하여 갱신된 CLV를 산출한다. 태스크 처리 횟수는 점점 커지기 때문에 최근에 사용된 블록이 비교적 높은 CLV를 가지게 된다.

NLV는 노드 전체 블록의 사용 정도를 나타내는 수치이다. 노드 내 모든 CLV의 합으로 산출한다. 해당 노드에서 보유한 블록에 대한 요청이 많을수록 NLV가 높아진다. NLV는 노드의 작업 부하를 측정하는 척도이다.

RLV는 랙 전체의 사용 정도를 나타내는 수치이다. 랙 내 모든 NLV의 합으로 산출한다. RLV는 랙 전체의 작업 부하를 측정하는 척도이다.

각 LV들은 맵리듀스 작업이 수행되면서 실시간으로 갱신된다. 그러므로 블록 재배치 알고리즘은 LV를 사용해서 맵리듀스 작업의 블록 접근 패턴에 적합하게 블록을 재배치한다.

3. 블록 재배치 알고리즘

블록 재배치 알고리즘은 하둡 클러스터를 구성하는 노드들의 부하를 평준화시켜 블록 이동을 최소화하여 네트워크 대역폭 소모를 줄인다. 블록의 재배치는 그림 1과 같은 순서대로 동작한다.

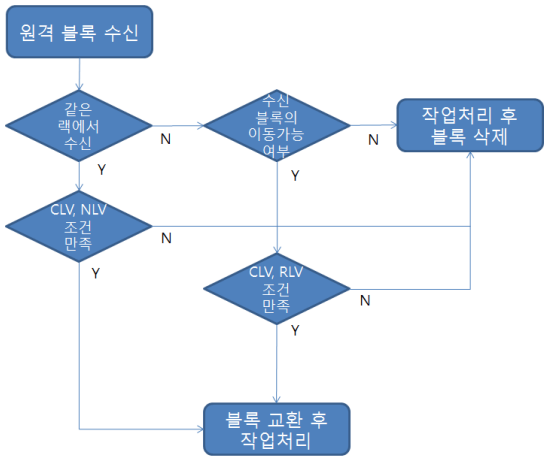

그림 1. 블록 재배치 알고리즘
Fig. 1. Block Relocation Algorithm

블록 재배치 알고리즘은 노드가 태스크 처리를 위해 외부 노드에 위치한 블록을 수신했을 때부터 시작된다. 노드는 수신한 블록이 원래 위치가 어디인지 먼저 확인한다. 만약 같은 랙에 위치한 노드에서 전송된 블록이면 이동이 가능하지만 다른 랙인 경우 이동이 불가능 할 수도 있다. HDFS는 결함허용을 위해 기본적으로 2개의 사본을 동일 랙에, 1개의 사본을 다른 랙에 저장하기 때문이다. 수신한 블록이 이동 가능한 블록으로 결정되면 몇 가지 조건을 만족시키는지 확인한 후에 노드에 재배치를 결정한다. 블록의 재배치는 수신한 블록을 노드에 저장하고 이동 가능한 블록 중 가장 낮은 CLV를 가진 블록을 송신 노드에 보내서 맞교환 하는 방식으로 동작한다. 만약 해당 조건들을 만족시키지 못한다면 지연 스케줄러와 같이 태스크 처리 후 삭제된다.

동일한 랙에서 수신된 블록인 경우 CLV와 NLV 조건을 만족하면 재배치가 실행된다. CLV 조건은 교환 대상인 두 블록의 CLV가 정해진 임계값 이상의 차이가 있어야 한다는 것이다. CLV가 비슷한 두 블록의 교환으로 발생되는 불필요한 대역폭 소모를 막기 위한 조건이다. NLV 조건은 만약 두 블록을 교환했을 때 재계산된 송신 노드의 NLV와 수신 노드의 NLV의 차가 적어야 한다는 것이다. CLV 차이가 큰 두 블록의 교환으로 인해 두 노드의 NLV차이가 더 커지는 것을 방지하기 위한 조건이다.

맵리듀스 작업이 진행되면서 각 노드들의 NLV는 평준화된다. 맵리듀스 작업 처리 도중 접근 패턴의 변화가 발생해도 LV들이 실시간으로 업데이트되기 때문에 빠르게 평준화에 도달할 수 있다. 각 노드들의 작업 부하가 평준화 되면 외부 노드에서 블록을 전송받아 처리하는 태스크가 감소하므로 하둡 클러스터 네트워크의 대역폭 소모를 줄일 수 있다.

IV. 성능평가

1. 개요

본 논문에서는 직접 구현 대신 JAVA로 구현된 시뮬레이터를 이용해 블록 재배치 알고리즘의 성능평가를 수행한다. 성능평가는 클러스터의 네트워크 사용량을 측정하는 것이기 때문에 하둡을 수정해서 직접 구현한다면 다수의 랙으로 구성된 대규모 클러스터 환경이 필요하기 때문이다.

시뮬레이터는 기존 연구의 성능 평가와 달리 작업 처리 시간이 아닌 네트워크 부하를 측정하는 것에 초점을 두고 구현했다. 제안하는 블록 재배치 알고리즘의 목적이 하둡 클러스터의 네트워크 전송량을 감소시키는 것이기 때문이다. 지연 스케줄링은 현재 태스크의 종료 시점을 예측해서 추가 태스크

를 할당하기 때문에 모든 작업들이 유휴 시간 없이 연속적으로 처리된다. 그러므로 Straggler 문제가 없다고 가정하면 최선(Optimal)에 가까운 작업 처리 시간을 보여준다. 하지만 작업의 연속적 처리를 위해 원격 블록의 호출이 빈번해지기 때문에 블록 배치에 따라 전송량의 차이가 존재한다. 그래서 블록 재배치 알고리즘의 성능 평가는 지연 스케줄링에 블록 재배치 알고리즘을 적용 전후의 네트워크 전송량을 비교하는 방법으로 이루어진다.

2. 시뮬레이션 환경

시뮬레이션 토폴로지는 4개의 랙에 각 10대의 서버가 연결된 간단한 구조이다. 랙과 랙 사이는 기가비트 이더넷(1GBit/s)로 연결되어 있다. 제안하는 알고리즘은 원격 데이터 호출을 줄여 네트워크 사용량을 줄이기 때문에 랙이 많을수록 더 큰 성능향상을 기대할 수 있다. 그리고 시뮬레이션의 목적이 작업 처리 산출이 아닌 네트워크 부하를 측정하는 것이기 때문에 노드의 성능에 관련된 시스템 파라미터들을 단순화 했다. 각 노드들은 동일한 성능을 가지며 동일한 용량의 블록들을 가진다. 본 시뮬레이션에서 시스템 파라미터는 표 1과 같고 작업 파라미터는 표 2와 같다. 시뮬레이션에 사용되는 블록 개수는 총 70,000개인데 하둡은 기본적으로 2개의 복사본을 추가로 생성하기 때문에 총 블록의 개수는 210,000개이다. 블록들은 하둡의 기본적인 복사본 배치 규칙을 만족하는 조건 하에서 5,250개씩 무작위로 각 노드들에게 배치한다.

표 1. 시스템 파라미터
Table 1. System Parameters

항목	값
총 노드 수	40개
블록 용량	64MByte
클러스터의 블록 개수	70,000개
복사본 포함 블록 개수	210,000개

표 2. 작업 파라미터
Table 2. Job Parameters

항목	값
작업 당 맵/리듀스 비율	9:1
전체 작업 수	630,000개
전체 태스크 수	6,300,000개

태스크 하나의 처리 시간은 6.27초로 가정한다. 태스크 처리 시간은 금[10]의 연구에서 산출된 WordCount 수행시간(Non-모니터) 값의 평균이다.

210,000개의 블록은 실제 클라우드 환경과 마찬가지로 각각 사용빈도가 다르다. 블록 사용 빈도는 Zipf 유사 분포로 생성한다. Zipf 유사 분포는 이미 많은 연구에서 분산 컴퓨팅 환경에 적합한 파일 인기도 분포라는 것을 증명했다[11-14]. 그래서 본 연구에서는 Lee[14]의 연구에서 사용한 ZipfL-0.8(Zipf 유사 분포, $\alpha = 0.8$)을 사용해서 블록 사용 빈도를 산출했다. 이렇게 산출한 전체 블록 사용빈도는 총 6,300,000번이며 630,000개의 작업에 랜덤으로 할당된다.

3. 시뮬레이션 결과

표 3은 앞서 설명한 방법으로 생성한 작업을 10번 수행해서 평균값을 산출한 결과이다. 성능 비교를 위해 하둡 스케줄러에 기본 지연 스케줄링만 적용했을 때와 블록 재배치 알고리즘을 함께 적용했을 때를 구분해서 산출했다. 그리고 블록 재배치 알고리즘은 CLV 조건의 임계값에 따라 블록의 교체 빈도를 조절한다. 그러므로 임계값을 작게 두면 불필요한 블록 맞교환이 증가하며 크게 두면 내부 처리 블록과 원격 처리 블록의 비율이 나빠진다. 본 성능평가에서는 CLV의 임계치를 10%(TH-10%), 20%(TH-20%), 30%(TH-30%)로 구분하여 시뮬레이션 하였다. 시뮬레이션 결과는 모든 CLV 조건의 임계값에서 기본 지연 스케줄링에 비해 향상된 내부/원격 처리 블록 비율을 보였다. CLV 조건의 임계값을 낮출수록 블록 교체율이 높아지면서 오버헤드가 더 발생하였으나 원격 처리 블록이 감소한 것보다는 현저히 낮은 것을 확인할 수 있다.

표 3. 시뮬레이션 결과
Table 3. Simulation Result

조건	내부 처리 블록 (TByte)	원격 처리 블록 (TByte)	블록 교체 오버헤드 (TByte)
Default	12.30	26.15	0
TH-10%	19.99	18.46	1.43
TH-20%	18.84	19.61	0.92
TH-30%	16.53	21.92	0.72

그림 2는 위의 시뮬레이션 결과를 바탕으로 계산한 네트워크 전송량이다. 네트워크 전송량은 원격 처리 블록과 블록 교체 오버헤드를 합산한 값이다. 모든 CLV 조건의 임계값 하에서도 기본 지연 스케줄링보다 약 13.4~23.3%의 네트워크

전송량이 감소했다. 다만 CLV 조건 임계값 10%와 20%의 차이는 크지 않았는데 내부 블록 처리의 용량과 블록 교체 오버헤드가 반비례하기 때문이다. CLV 조건 임계값 30%는 블록 교체 오버헤드는 줄지만 내부 처리 블록이 크게 늘어났기 때문에 네트워크 전송량 차이가 발생하였다. 최적의 CLV 조건의 임계값은 적용하는 하둡 클러스터의 데이터 사용 패턴에 따라 달라지기 때문에 클러스터 관리자가 상황에 따라 CLV 조건의 임계값을 조정할 필요가 있다.

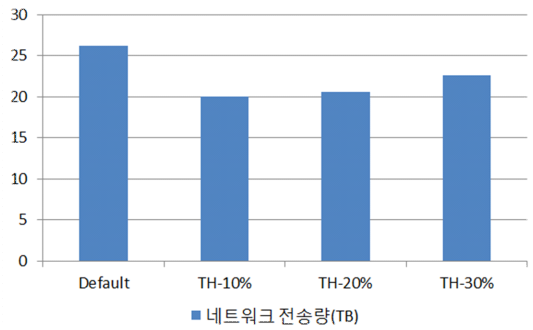

그림 2. 네트워크 전송량 비교

V. 결 론

본 논문에서는 하둡 클러스터의 네트워크 사용량 감소를 위한 블록 재배치 알고리즘을 제안했다.

클러스터의 데이터 블록들은 사용 빈도가 각각 다르기 때문에 노드 간의 작업 부하 차이가 발생된다. 노드 간 부하 불균형은 원격 태스크 처리의 증가의 원인이 되며 이로 인해 노드 간 데이터 전송이 빈번해진다. 제안한 블록 재배치 알고리즘은 하둡 스케줄러의 태스크 할당 패턴에 따라 블록들을 재배치하여 노드들의 작업 부하를 평준화 시킨다. 노드들의 작업 부하가 평준화 되면 원격 태스크 처리가 감소하기 때문에 하둡 클러스터 전체의 네트워크 사용량이 감소한다.

제안하는 알고리즘의 성능평가를 위해 시뮬레이션을 수행하였다. 우선 하둡의 지연 스케줄러에서 블록 재배치 알고리즘을 적용하면 약간의 데이터 교체로 인한 네트워크 사용이 발생하지만 이로 인해 감소하는 네트워크 사용량이 훨씬 더 큰 것을 확인할 수 있었다. 이를 합산하여 결과를 산출 했을 때 지연 스케줄링만 적용했을 경우에 비해 네트워크 사용량이 약 13.4~23.3% 감소하였다.

참고문헌

[1] Jeong-Hyeok Park, Sang-Yeol Lee, Da-Hyun Kang, Joong-Ho Won, "Hadoop and MapReduce," Journal of the Korean Data & Information Science Society, Vol 24, No 5, pp. 1013-1027, 2013.

[2] Apache Hadoop http://hadoop.apache.org

[3] Olson, Mike. "Hadoop: Scalable, flexible data storage and analysis." IQT Quarterly 1.3 (2010): 14-18.

[4] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[5] Borthakur, Dhruba. "The hadoop distributed file system: Architecture and design." Hadoop Project Website 11 (2007): 21.

[6] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." ACM SIGOPS Operating Systems Review. Vol. 37. No. 5. ACM, 2003.

[7] Zaharia, Matei. "Job scheduling with the fair and capacity schedulers." Hadoop Summit 9 (2009).

[8] Zaharia, Matei, et al. "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling." Proceedings of the 5th European conference on Computer systems. ACM, 2010.

[9] Zaharia, Matei, et al. "Improving MapReduce Performance in Heterogeneous Environments." OSDI. Vol. 8. No. 4. 2008.

[10] Tae Hoon Keum, Won Joo Lee, Chang Ho Jeon, "Design and Implementation of a Monitor for Hadoop Cluster," Journal of the Institute of Electronics and Information Engineers, Vol 41, No 1, pp. 8-15, 2012.

[11] Cameron, David G., et al. "Evaluating scheduling and replica optimisation strategies in OptorSim." Proceedings of the 4th International Workshop on

Grid Computing. IEEE Computer Society, 2003.
〔12〕 Ranganathan, Kavitha, and Ian Foster. "Simulation studies of computation and data scheduling algorithms for data grids." Journal of Grid Computing 1.1 (2003): 53-62.
〔13〕 Tang, Ming, et al. "Dynamic replication algorithms for the multi-tier data grid." Future Generation Computer Systems 21.5 (2005): 775-790.
〔14〕 Lee, Ming-Chang, Fang-Yie Leu, and Ying-ping Chen. "PFRF: An adaptive data replication algorithm based on star-topology data grids." Future Generation Computer Systems 28.7 (2012): 1045-1057.

저 자 소 개

김 준 상

2003: 한양대학교
전자컴퓨터공학부 공학사
2005: 한양대학교
컴퓨터공학과 공학석사
2008-2012: 해군사관학교
컴퓨터학과 전임강사
현 재: 한양대학교
컴퓨터공학과 박사과정
관심분야: 성능 분석, 그리드 컴퓨팅, 클라우드 컴퓨팅
Email : kjspbe@hanyang.ac.kr

김 창 현

2008: 경일대학교
컴퓨터공학과 공학사
2010: 한양대학교
컴퓨터공학과 공학석사
현 재: 한양대학교
컴퓨터공학과 박사과정
관심분야: 센서 네트워크, 성능 분석, 그리드 컴퓨팅, 클라우드 컴퓨팅
Email : ctcquatre@hanyang.ac.kr

이 원 주

1989: 한양대학교
전자계산학과 공학사
1991: 한양대학교
컴퓨터공학과 공학석사
2004: 한양대학교
컴퓨터공학과 공학박사
현 재: 인하공업전문대학
컴퓨터정보과 교수
관심분야: 병렬처리시스템, 성능분석, 모바일 컴퓨팅, 그리드 컴퓨팅, 클라우드 컴퓨팅
Email : wonjoo2@inhatc.ac.kr

전 창 호

1977: 한양대학교 전자공학과 학사
1982: Cornell University
컴퓨터공학과 석사
1986: Cornell University
컴퓨터공학과 박사
현 재: 한양대학교 컴퓨터공학과 교수
관심분야: 병렬처리시스템, 성능 분석, 그리드 컴퓨팅, 클라우드 컴퓨팅
Email : chj5193@hanyang.ac.kr