논문 2014-09-14

MapReduce 시스템을 위한 에너지 관리 알고리즘의 성능평가

(Performance Evaluation of Energy Management Algorithms for MapReduce System)

김 민 기, 조 행 래*
(Min-Ki Kim, Haengrae Cho)

Abstract : Analyzing large scale data has become an important activity for many organizations. Since MapReduce is a promising tool for processing the massive data sets, there are increasing studies to evaluate the performance of various algorithms related to MapReduce. In this paper, we first develop a simulation framework that includes MapReduce workload model, data center model, and the model of data access pattern. Then we propose two algorithms that can reduce the energy consumption of MapReduce systems. Using the simulation framework, we evaluate the performance of the proposed algorithms under different application characteristics and configurations of data centers.

Keywords : Performance evaluation, Simulation, Large-scale data processing, MapReduce

Ⅰ. 서 론

최근 대용량 데이터 처리에 대한 수요가 급증하고 있다. 그에 따라 대용량 데이터를 효율적으로 처리할 수 있는 MapReduce 시스템에 대한 연구들도 활발하게 이루어지고 있다. 기존의 MapReduce 연구들은 실제 시스템이 구축된 환경에서 수집된 데이터를 이용하여 벤치마크를 통한 성능 평가를 하였다[1-4]. 그러나 데이터 수집이 어려운 경우나 벤치마크 환경에 필요한 다수의 서버를 구축하기 어려운 경우에는 개발한 알고리즘의 성능을 평가하기가 매우 어렵다는 문제점이 존재한다. 본 논문에서는 이러한 문제점을 해결하기 위해 다양한 MapReduce 응용분야의 특성과 데이터 센터의 구성을 시뮬레이션 방식으로 쉽게 표현할 수 있는 성능평가 모형을 개발 한다. 개발한 성능평가 모형은 부하 모형과 시스템 모형, 그리고 데이터 모형으로 구성된다. 부하 모형은 다양한 MapReduce 작업 특성을 지원하고, 시스템 모형은 데이터 센터를 구성하는 랙과 서버의 특성을 표현한다. 데이터 모형은 데이터들이 서버에 배치되는 방식 및 각 데이터에 대한 액세스 패턴을 모델링 한다. 시뮬레이션 방식을 사용함으로써 다양한 시스템 구성과 부하 환경에서 알고리즘의 성능을 쉽게 파악할 수 있고 이를 개선한 후 실제 환경에 적용함으로써 비용 및 시간을 절약할 수 있다.

성능평가 모형의 유효성과 적용가능성을 검증하기 위하여 본 논문에서는 MapReduce 시스템의 소비전력을 절감할 수 있는 에너지 관리 알고리즘을 제안하고 성능평가를 수행한다. 클러스터 환경에서 서버 노드는 일반적으로 20-30%만 활용된다. 이러한 환경에서 모든 서버를 항상 활성화 상태로 두었을 때의 에너지 효율은 50% 이하이다[5]. 본 논문에서는 에너지 절감을 위해 서버 활용도가 낮은 기간 동안 선택적으로 서버나 랙의 전원을 차단하는 두 가지 알고리즘을 제안하고, 개발한 성능평가 모형을 이용하여 기존 알고리즘들과 성능을 비교한다.

논문의 구성은 다음과 같다. 2절에서는 관련 연구에 관하여 소개하며, 3절에서는 본 논문에서 개발한 성능평가 모형을 설명한다. 4절에서는 본 논문에서 제안하는 에너지 절감 알고리즘을 설명하고, 5절에서는 실험 모형을 이용하여 수행한 실험 결과를 분석한다. 끝으로 6절에서 결론을 맺는다.

*Corresponding Author (hrcho@yu.ac.kr)
Received: 10 Jan. 2014, Revised: 10 Feb. 2014.
Accepted: 24 Feb. 2014.
M.K. Kim, H. Cho: Yeungnam University
※ 본 논문은 한국연구재단의 BK21+ 프로그램에서 지원하여 연구하였음.

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2014.9.2.109

Ⅱ. 관련 연구

MapReduce 시스템의 성능 평가를 다룬 연구는 실제 환경에서 벤치마크를 하는 방식과 시뮬레이션을 이용한 방식으로 나눌 수 있다.

벤치마크 방식의 예를 들면 36대 노드로 실험 환경을 구축한 후 Hadoop 클러스터의 구성 변경에 따른 성능 평가를 하였다[1]. 그러나 페이스북이나 야후가 보유한 실제 데이터 센터들과 비교하면 그 규모가 매우 적어 벤치마크의 결과와 실제 시스템 간의 성능 차이가 발생할 수 있는 문제점이 있다.

이와는 달리 시뮬레이션 방식은 그 규모의 제한이 없으며 다양한 환경 하에서 MapReduce 알고리즘들의 성능 평가를 할 수 있는 이점이 있다. 예를 들어 Java로 구현된 GreenHDFS 시뮬레이터를 가지고 5페타바이트 규모의 데이터세트를 2600개의 서버 환경에서 알고리즘을 적용하고 성능 평가를 한 연구도 있다[6]. 그러나 기존의 시뮬레이션 방식은 데이터 센터에서 생성된 실제 데이터들을 실험에 이용하는 경우가 대부분이므로 이를 구하기 어려운 경우에는 실험이 곤란하다는 문제점이 있다.

시뮬레이션 방식과 벤치마크 방식을 병행한 연구 방식도 있다[2]. 이 연구에서는 Amazon Web 서비스인 Amazon EC2를 활용하여 수백대 규모의 실험 환경을 가지고 성능 평가를 하였다. 이와 같이 두 가지 연구 방식을 병행하면 시뮬레이션을 통해 실험 규모 및 속도 측면에서의 이점과 함께 실제 환경에서의 정확한 결과 또한 얻을 수 있지만 비용이 많이 든다는 단점이 있다.

Ⅲ. 실험 모형

본 절에서는 MapReduce 알고리즘의 성능을 쉽게 평가할 수 있는 시뮬레이션 기반의 성능평가 모형을 제안한다. 제안하는 성능평가 모형은 부하, 시스템, 데이터 모형으로 구성된다.

1. 부하 모형

MapReduce 시스템의 성능을 평가하기 위해서는 우선 부하 모형을 설계해야 한다. 본 논문에서는 페이스북에서 2010년 10월 1일부터 11월 15일까지 한 달 반 동안 백만 개 이상의 작업을 분석하여 열 가지 유형으로 작업을 분류한 것 중에서 유사한 성질의 작업들을 제외한 다섯 가지 작업 유형을 아래 표 1과 같이 정의하고 이를 활용하였다[4].

표 1. 부하 모형에 사용되는 작업 유형
Table 1. Job types in workload model

	발생 확률	실행 시간	분류
(1)	98.12 %	1분	Small jobs
(2)	0.68 %	8시간	Map only transform
(3)	0.06 %	1시간 20분	Map only aggregate
(4)	0.98 %	30분	Aggregate
(5)	0.16 %	2시간	Transform

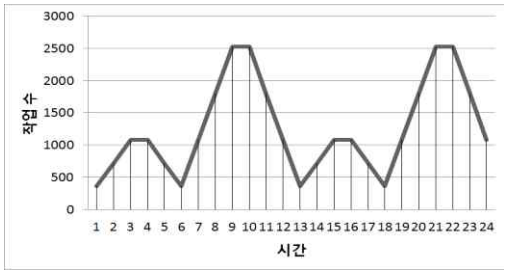

그림 1. 부하 모형 시나리오
Fig. 1 Workload model scenario

표 1에서 (1)은 가장 짧은 1분의 실행 시간을 가지고 98.12%의 가장 높은 발생 확률을 가진 작업이다. (2)는 작업 유형중 가장 긴 8시간의 실행 시간을 가지는 작업이다. (3)은 가장 낮은 0.06%의 발생 확률과 함께 약 1시간 정도의 실행 시간을 요구하는 작업이다. (4)는 (1) 다음으로 가장 많이 발생하는 작업으로 30분의 실행 시간을 가진다. (5)는 2시간의 실행 시간을 요구하는 작업이다.

부하 모형은 각 분류에 해당하는 작업들의 발생 확률을 정의하고, 시간별 작업수를 변경함으로써 다양한 부하 환경을 생성 할 수 있다. 본 논문에서는 (1)이 대부분인 부하 환경을 고려하였고, 시간별로는 작업 부하가 여섯 시간의 주기를 가진다. 그리고 시간당 최소 500개에서 최대 2,500개까지의 작업이 발생하는 환경을 가정하였다. 그림 1은 본 연구에서 활용한 부하 모형 시나리오를 나타낸다.

2. 시스템 모형

시스템 모형은 데이터 센터의 구성을 나타내고 있으며, 랙과 서버 모형으로 나누어진다.

랙은 여러 개의 서버를 포함하며, 랙 내부/외부의 통신을 처리한다. 본 논문에서는 [2]와 동일하게 랙 내부 통신의 대역폭은 128Gbps, 랙 외부의 통신의 대역폭은 64Gbps로 설정하였다. 랙은 고정적인 에너지 소비량(냉각 시스템, 통신)과 활성화된

표 2. 실험 모형 매개 변수
Table 2. Model parameters

매개 변수	변수 값
Number of Racks	75, 300
Number of Nodes	3000
Rack Transition Time (s)	300
Node Transition Time (s)	30
Number of M / R Slots per Server	10 / 5
Idle Server Power (w/h)	100
Run Server Power (w/h)	500
Internal / External Network (Gbps)	128 / 64
Hot Data Layout	0, 0.9
Hot Data Ratio (%)	40
Hot Data Access Ratio (%)	80

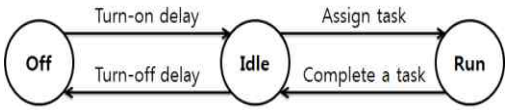

그림 2. 서버 모형의 상태 변화도
Fig. 2 State transition diagram for a server

서버에 비례한 에너지 소비량을 가진다. 에너지 소비량에 대한 구체적인 값들은 [7]를 참조하였으며 표 2에 이를 정리하였다.

MapReduce 작업을 수행하기 위해서 각 서버는 Map과 Reduce 슬롯을 가지는데 [8]를 참조하여 Map 슬롯 수는 10으로, Reduce 슬롯 수는 5로 설정하였다. 각 서버의 상태는 그림 2에 나타나는 바와 같은데, Off는 서버가 꺼져 있는 상태이고, Idle은 서버가 켜져 있으나 할당 된 작업이 없는 대기 상태를 나타낸다. 그리고 Run은 작업이 할당되어 처리중인 상태를 나타낸다. 각 상태에 따른 에너지 소비량(w/h)은 Idle일 때 100, Run일 때 500이다.

3. 데이터 모형

데이터 모형은 데이터들을 서버에 배치하는 방식과 각 데이터에 대한 액세스 패턴을 나타낸다.

데이터 액세스에 대한 skew를 표현하기 위하여 자주 사용되어지는 데이터와 그렇지 않은 데이터를 Hot/Cold 데이터로 분류하였다. 전체 데이터 중에 Hot 데이터의 비율을 다양하게 변화시키면서 실험을 수행할 수 있다. 여기서는 [3]을 참조하여 Hot 데이터 비율은 40%, Hot 데이터에 대한 액세스 비율은 80%로 설정하였다.

다양한 시뮬레이션 환경을 구성하기 위해서 그림 3과 같이 Hot 데이터 배치 방식을 균등 분포와

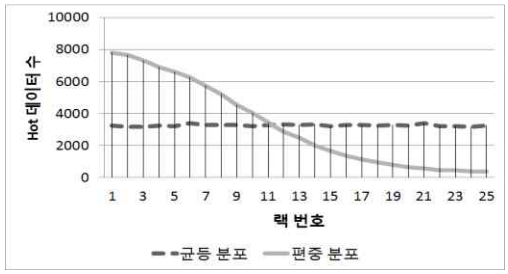

그림 3. Hot 데이터 배치
Fig. 3 Hot data layout

편중 분포로 나누어 실험을 수행하였다. 편중 분포는 특정 랙에 더 많은 Hot 데이터를 배치한 것이고 균등 분포는 모든 랙에 고르게 Hot 데이터를 분산 배치한 것이다. 표 2의 Hot Data Layout에서 0과 0.9는 그림 3에서의 균등 분포와 편중 분포에 각각 대응된다.

IV. 알고리즘

본 절에서는 MapReduce 시스템의 소비 전력을 절감하기 위하여 활성화된 서버 수를 동적으로 제어하는 두 가지 기존 알고리즘과 본 논문에서 제안하는 두 가지 알고리즘을 설명한다. 각 알고리즘은 서버 활성화 규모에 따른 차이점을 가진다.

1. 기존 알고리즘

A. Covering Set Strategy (CS)

CS는 분산파일시스템(DFS: Distributed File System)에서 데이터 배치를 이용하여 에너지 소비를 줄이는 전략이다[1]. 기본 개념은 모든 데이터 블록에 대해서 세 개의 사본을 유지하는 것이다. ‘CS 노드’는 모든 데이터 블록에 대해 하나의 사본을 가지는 노드들로 이루어지며 항상 활성화 상태로 둔다. 이와는 달리 나머지 노드들은 부하 상태에 따라 소비 전력을 절감하기 위하여 비활성화 될 수 있다. 비록 특정 노드가 비활성화 되더라도 CS 노드를 통하여 모든 데이터를 액세스할 수 있으므로 데이터의 가용성은 유지될 수 있다. 만약 CS 노드가 전체 노드들의 25%라면, 작업 처리 중에 나머지 75%의 노드들은 전원을 차단하여 에너지 효율을 높일 수 있다.

B. All-In Strategy (AI)

AI는 CS 전략이 가지는 몇 가지 약점을 보완한 전략이다[6]. CS가 가지는 약점은 크게 세 가지로

그림 4. 활성화 서버 선택의 예
Fig. 4. Example of a selection of activation server

첫째 DFS 데이터 배치 전략을 수정해야 하며, 둘째로 CS 노드들로만 운영될 경우 응답 시간이 저하될 수 있다. 마지막으로 CS는 뛰어난 작업 부하 예측 시스템이 필요하다는 점이다. AI는 시스템의 모든 노드들로 작업을 처리하고, 작업이 없을 때는 전체 시스템의 전원을 차단하는 전략이다. 그 결과 AI는 데이터 배치 전략을 수정할 필요가 없고, 세밀하게 노드 전원 관리를 할 필요가 없다. 뿐만 아니라, 작업부하의 응답 시간 저하를 쉽게 계산 할 수 있다는 장점이 있다. 본 논문에서는 작업 부하 시나리오가 가지는 특성에 따라 서버의 일정 부분을 항상 활성화해두고 나머지 부분에 대해서 기존 AI 특성을 가지는 수정된 알고리즘을 구현하여 실험을 수행 하였다.

2. 제안한 알고리즘

A. 랙 기반 활성화 알고리즘 (RA)

RA(Rack-based Activation)는 추가적인 서버 활성화가 필요할 때 현재 작업이 액세스 할 데이터를 수집한 후 가장 많은 데이터가 포함된 랙에 있는 서버를 우선적으로 활성화 한다. 이 알고리즘은 랙의 모든 서버가 활성화가 되기 전까지 꺼져 있는 다른 랙의 서버는 활성화를 하지 않는다. 예를 들어 그림 4는 현재 작업에 대한 데이터 보유량을 서버 1 > 서버 4 > 서버 7 > 서버 2 순으로 나타내었다. 여기서 추가적으로 3대의 서버 활성화가 필요할 때 RA는 서버 1, 서버 2, 서버 3을 추가적으로 활성화하고 나머지 랙과 서버는 비활성화 상태를 유지한다. 그 결과 랙이 사용하는 에너지 소비량이 많을 경우에 에너지 효율성을 높일 수 있다.

B. 서버 기반 활성화 알고리즘 (SA)

앞의 RA가 에너지 효율성을 추구한 것이라면 SA(Server-based Activation)는 작업 처리 속도에

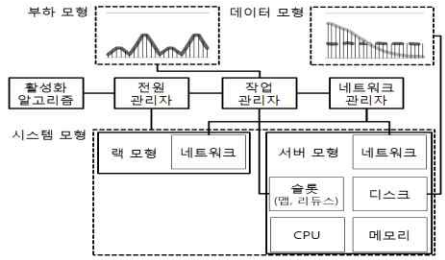

그림 5. 시뮬레이션 모형
Fig. 5. Simulation model

서 성능을 향상시키기 위한 알고리즘이다. 작업에 필요한 데이터가 가장 많은 서버를 우선적으로 활성화함으로써 처리 속도는 향상되나 데이터가 여러 랙에 분산되어 있으면 에너지 소비량이 증가하는 단점이 존재한다. 그림 4와 같이 각 서버가 현재 작업에 대한 데이터 보유량을 나타내고 있을 때, 추가적으로 3대의 서버 활성화가 필요한 경우 SA는 서버 1, 서버 4, 서버 7을 추가적으로 활성화하게 된다. 이 때 SA는 RA에 비해 2대의 추가 랙 활성화로 인해 에너지 효율성이 낮아진다.

V. 실험 결과

본 절에서는 제안한 성능평가 모형의 동작여부 및 정확성을 검증하기 위하여 네 가지 서버 활성화 알고리즘(CS, AI, RA, SA)들의 정성적인 평가와 실제 실험 결과를 비교한다. 실험에 사용된 주요 매개변수는 3절의 표 2에 나타내었다.

주요 성능지수는 소비 전력과 평균 응답 시간이다. 소비 전력은 활성화 상태인 서버와 랙에서 소비되는 전력의 합으로 정의된다. 평균 응답 시간은 각 작업 분류별로 작업 요청부터 완료까지의 소요시간에 대한 평균으로 정의된다. 보조 성능지수로 최대 응답시간과 데이터 전송량을 사용한다. 최대 응답시간은 작업 요청부터 완료까지의 소요시간에 대한 최대 시간으로 정의된다. 데이터 전송량은 랙 내부/외부의 전체 데이터 전송량으로 정의된다.

본 실험에서는 이산-이벤트 시뮬레이션 패키지인 CSIM을 활용하여 시뮬레이터를 개발했다. 실험에 활용된 시뮬레이션 모형은 그림 5와 같다[9]. 그림 5의 시뮬레이션 모형은 3절에서 설계한 부하, 시스템, 데이터 모형을 포함하고 있다. 여기서 작업 관리자는 작업을 각 서버에 할당하며, 전원 관리자

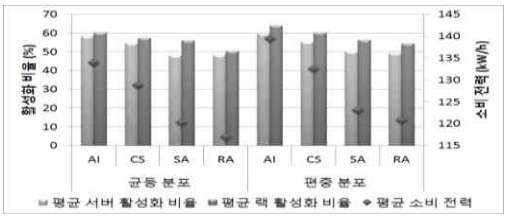

그림 6. 성능 평가 결과 – 에너지 소비
Fig. 6 Energy efficiency evaluation

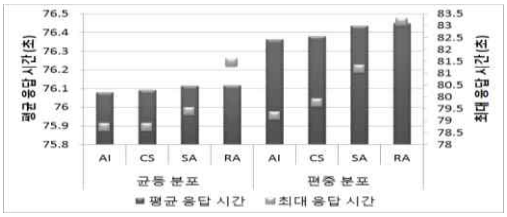

그림 7. 성능 평가 결과 – 응답 시간(Small jobs)
Fig. 7 Performance evaluation for small jobs

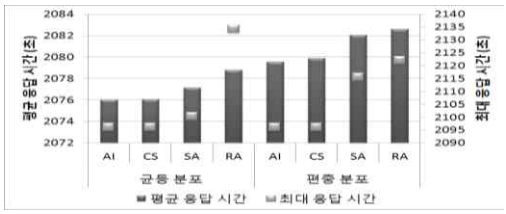

그림 8. 성능 평가 결과 – 응답 시간(Aggregate)
Fig. 8 Performance evaluation for aggregate

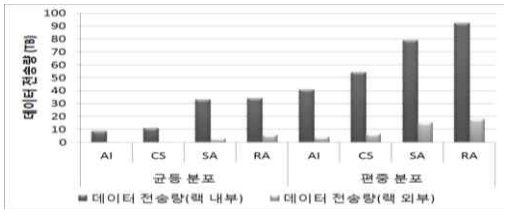

그림 9. 데이터 전송량
Fig. 9 Data transfer

는 랙과 서버의 활성화 상태를 관리한다.

1. 데이터 배치에 따른 성능 평가

본 실험은 데이터를 모든 랙에 균등하게 분산 시켰을 때와 데이터를 특정 랙에 편중 시켰을 경우 서버 활성화 알고리즘의 성능 차이를 관찰하는 것이 주요 목적이다.

실험 결과를 확인하기에 앞서 각 알고리즘에 대한 성능 예측을 해보면 소비 전력에 있어서는 RA < SA < CS < AI 의 순서로 RA가 가장 소비 전력이 낮고 AI가 가장 높을 것으로 예상된다. 그 이유는, RA와 SA는 필요한 만큼만 서버를 활성화하기 때문에 소비 전력량이 낮고 필요 이상의 서버를 활성화하는 CS와 AI는 많은 전력을 소비하기 때문이다. 그리고 SA가 RA보다 소비 전력이 높을 것으로 예상된다. 그 이유는, RA는 하나의 랙을 활성화 하면 그 랙에 포함된 모든 서버가 활성화 될 때까지 다른 랙을 활성화할 수 없다. 이와는 달리 SA는 필요한 서버를 랙과는 무관하게 활성화한다. 따라서 같은 수의 서버가 활성화 되어 있어도 RA에서 활성화되는 랙의 수가 적다.

다음으로 응답 시간의 경우 AI와 CS가 SA와 RA보다 빠를 것으로 예측 된다. 그 이유는, RA와 SA는 서버 활성화가 필요할 때마다 활성화 시간만큼 지연이 발생한다. 이와는 달리, CS와 AI는 추가적인 서버 활성화에 드는 지연시간이 없으므로 빠른 응답 시간을 보일 것으로 예상 된다. 그리고 부분적인 응답 시간은 SA가 RA보다 빠를 것이다. 그 이유는, RA의 경우는 필요한 데이터를 가진 서버가 활성화 되어 있지 않을 경우가 SA에 비해 더 빈번하므로 데이터 전송량이 증가하기 때문이다.

실험 모형을 이용한 성능 평가 결과가 그림 6에서 9까지 나타난다. 그림 6의 에너지 소비 결과를 보면 앞서 예측한 결과와 동일하게 RA < SA < CS < AI 순으로 소비 전력을 나타내고 있다. 편중 분포의 경우 AI를 기준으로 봤을 때 CS, RA, SA순으로 각각 5%, 13%, 15.5%만큼 소비 전력의 차이를 보인다. 그리고 균등 분포일 때 RA와 SA에서 랙 활성화 비율의 차이가 5.6%로 나타났다. 반면에 편중 분포일 때 랙 활성화 비율의 차이는 1.9%로 나타났다. 이로 인해 Hot 데이터의 배치에 따른 소비 전력의 차이가 균등의 경우 3.3kW/h, 편중의 경우 2.3kW/h만큼 RA의 소비 전력이 낮게 나타난다.

다음으로 평균 응답 시간 결과를 보면 예상대로 AI와 CS가 SA와 RA보다 빠른 것으로 나타났다. AI가 CS보다 더 큰 활성화 규모를 보임에도 응답 시간이 최소 0.01초에서 최대 0.3초 정도의 차이를 보이며 비슷한 성능이 나타났다. 그 이유는, 추가적인 활성화에 따른 지연시간에 의한 응답 시간의 차이가 전체 평균에 큰 영향을 주지 않기 때문이다. 그리고 최대 응답 시간 결과는 SA가 RA보다 빠르게 나타났다. SA는 작업 데이터를 보유한 서버를

우선적으로 추가 활성화한다. 그렇기 때문에 그림 9와 같이 전체적인 데이터 전송량이 SA가 RA보다 10%~16% 더 적게 나타난다. 이로 인해 응답 시간의 차이가 발생하고 그림 7에서와 같이 균등 분포의 경우 SA가 RA보다 최대 응답 시간이 2초 정도 더 빠름을 확인할 수 있다.

2. 랙 당 서버 수에 따른 성능 평가

다음 실험으로 전체 노드 수는 동일하지만 랙 당 서버 수가 다를 때 RA와 SA의 성능을 비교하였다. RA의 경우는 랙을 기준으로 서버를 활성화하기 때문에 랙과 서버의 활성화 규모가 항상 일정한 비율을 유지하고 SA는 최악의 경우에 모든 랙들이 각각 하나의 서버를 운영하기 위해서 활성화 되는 경우도 발생 할 수 있다. 랙 당 서버가 많은 경우(R_L)와 서버 수가 적은 경우(R_S)로 나누어 실험을 수행하였다. 구체적으로, R_L의 경우 랙 당 서버 수가 40개, R_S의 경우 랙 당 서버 수는 10개이다. 랙의 수는 R_L의 경우 75개, R_S의 경우 300개이다. 그 결과 두 경우 모두 전체 서버 수는 3,000개이다.

실험 결과는 그림 10에서 12까지 나타난다. 그림 10의 에너지 소비 결과를 보면 RS와 SA 모두 R_L에서 소비 전력이 높게 나타났다. 그 이유는, R_L에서 랙의 고정 소비 전력이 R_S보다 크기 때문이다. 특히 균등 분포일 때 가장 많은 소비 전력을 나타낸 경우는 SA이며 R_L일 때이다. 이러한 경우 대부분의 랙이 적은 수의 서버를 위해 활성화되기 때문에 에너지 소비도 크게 나타난다. 반면에 가장 적은 소비 전력을 나타낸 경우는 RA이며 R_S일 때이다. 이 두 경우의 소비 전력 차이는 6.4%로 나타났다. 다음으로 그림 11의 응답 시간 결과를 보면 R_L의 평균 응답시간이 최소 0.5초에서 최대 1.8초 R_S보다 빠르게 나타났다. 그 이유는, 랙간의 데이터 전송량의 차이에 있다. R_L이 R_S보다 랙 당 서버 수가 4배 많기 때문에 각각의 랙이 보유한 데이터 보유량도 많다. 따라서 R_L은 랙 내부에서 작업을 수행 할 수 있는 경우가 더 많아서 R_S보다 랙 외부 데이터 전송량이 적어진다. 실제로 그림 12에서 나타난 것과 같이 R_S가 R_L보다 랙 외부 데이터 전송량이 평균 3TB정도 더 많다. 따라서 대부분의 경우 SA와 RA는 R_L에서 응답 시간이 빠름을 확인 할 수 있다. 단, 유일한 예외로 RA는 균등 분포에서 R_L보다 R_S에서 응답 시간이 조금 빨랐다. 그 이유는, R_L에서 RA는 40개의 서버가 모두 활성화 될 때까지 새로운 랙을 활성화할 수 없지만, R_S에서는 10개의 서버만 활성화하면 새로운 랙을 활성화 할 수 있다. 따라서 R_S일 때 현재 데이터 액세스 패턴에 적합한 랙을 보다 빨리 활성화할 수 있다.

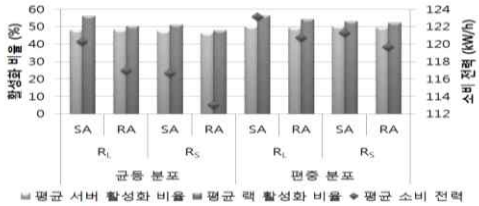

그림 10. 성능 평가 결과 – 에너지 소비
Fig. 10 Energy efficiency evaluation

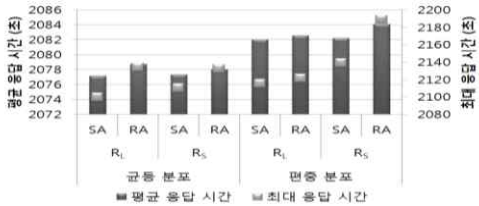

그림 11. 성능 평가 결과 – 응답 시간(Aggregate)
Fig. 11 Performance evaluation for aggregate

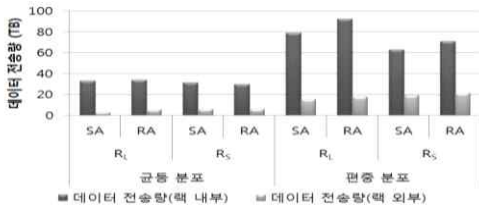

그림 12. 데이터 전송량
Fig. 12 Data transfer

VI. 결론 및 향후 연구 방향

대규모 데이터를 효율적으로 처리할 수 있는 MapReduce 시스템 관련 알고리즘에 대한 연구가 활발히 이루어지고 있다. 마찬가지로 이러한 알고리즘들에 성능 평가를 다룬 연구도 활발히 이루어지고 있다.

본 논문에서는 먼저 시뮬레이션을 이용한 MapReduce 시스템의 성능 평가 모형을 개발하였다. 성능 평가 모형은 부하, 시스템, 데이터 모형으로 구성되어 있다. 다음으로 MapReduce 시스템의 소비 전력을 절감하기 위하여 활성화된 서버수를 동적으로 제어하는 두 가지 알고리즘(RA와 SA)을 제안하였다. 각 알고리즘은 서버 활성화 규모에 따른 차이점을 가진다. 제안한 알고리즘의 특성을 고려한 실험 모형을 설계하고 시뮬레이션을 통한 성능 평가를 하였다. 실험을 통해 얻은 주요 결과는 다음과 같다:

- RA와 SA는 기존의 CS와 AI 알고리즘보다 최대 15.5%만큼 소비 전력을 절감하였다.
- RA는 SA보다 소비 전력을 최대 6.4% 절감할 수 있지만, 응답 시간은 최대 1.6% 정도 증가하였다.
- 데이터가 모든 랙에 고르게 분산된 균등 분포의 경우가 데이터가 일부 랙에 편중 분포된 경우보다 응답 시간이 빠르게 나타났다.

향후 연구로 시뮬레이션 방식을 통해 본 연구에서 다룬 알고리즘을 개선하고 더 다양한 알고리즘을 고안하여 성능 평가를 진행할 예정이다.

References

[1] J. Leverich, C. Kozyrakis, "On the Energy (in) Efficiency of Hadoop Clusters," ACM SIGOPS Operating Systems Review, Vol. 44, No. 1, pp.61-65, 2010.

[2] Y. Chen, S. Alspaugh, D. Borthakur, R. Katz, "Energy Efficiency for Large-Scale Map-Reduce Workloads with Significant Interactive Analysis," Proceedings of ACM European Conference on Computer Systems, pp.43-56, 2012.

[3] Y. Chen, A. Ganapathi, R. Griffith, R. Katz, "The Case for Evaluating MapReduce Performance Using Workload Suites," Proceedings of IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp.390-399, 2011.

[4] C.L. Abad, N. Roberts, Y. Lu, R.H. Campbell, "A Storage-Centric Analysis of MapReduce Workloads: File Popularity, Temporal Locality and Arrival Patterns," Proceedings of IEEE International Symposium on Workload Characterization, pp.100-109, 2012.

[5] W. Lang, J.M. Patel, "Energy Management for MapReduce Clusters," Proceedings of VLDB Endowment, Vol. 3, No. 1-2, pp.129-139, 2010.

[6] R.T. Kaushik, M. Bhandarkar, "GreenHDFS: Towards An Energy-Conserving Storage-Efficient, Hybrid Hadoop Compute Cluster," Proceedings of USENIX Annual Technical Conference, 2010.

[7] V.A. Patil, V. Chaudhary, "Rack Aware Scheduling in HPC Data Centers: An Energy Conservation Strategy," Proceedings of IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, pp.814-821, 2011.

[8] S. Hammoud, M. Li, Y. Liu, N.K. Alham, Z. Liu, "MRSim: A Discrete Event Based MapReduce Simulator," Proceedings of International Conference on Fuzzy Systems and Knowledge Discovery, pp.2993-2997, 2010.

[9] H. Schwetman, "User's Guide: Article Reprints: CSIM 18-The Simulation Engine," World Wide Web electronic publication, 2009.

저 자 소 개

김 민 기

2013년 영남대 컴퓨터공학과 학사.

현재, 영남대 컴퓨터공학과 석·박사통합학위과정.

관심분야: 빅 데이터, 맵리듀스, 데이터마이닝.

Email: kminki@ynu.ac.kr

조 행 래

1988년 서울대 컴퓨터공학과 학사.

1990년 한국과학기술원 전산학과 석사.

1995년 한국과학기술원 전산학과 박사.

현재, 영남대 컴퓨터공학과 교수.

관심분야: 데이터베이스, 분산/병렬 처리, 빅 데이터.

Email: hrcho@yu.ac.kr