Journal of the Korea Industrial Information Systems Research Vol. 21 No.3, June. 2016 :13-19
http://dx.doi.org./10.9723/jksiis.2016.21.3.013 ISSN:1229-3741

메모리 기반 빅데이터 처리 프레임워크의 성능개선 연구

(An Empirical Evaluation Analysis of the Performance of In-memory Bigdata Processing Platform)

이 재 환[1)*], 최 준[2)], 구 동 훈[3)]
(Jae hwan Lee, Jun Choi, and Dong hun Koo)

요 약 최근에 실시간 처리를 위해 메모리 기반의 빅데이터 처리 프레임 워크인 스파크가 널리 사용되고 있다. 스파크는 프로그램이 필요로 하는 중간 데이터를 모두 메모리에 올려놓아, I/O 수행을 최소화함으로써 빠른 응답을 가져올 수 있다. 그러나 응용프로그램의 메모리 사용량이 클러스터의 실제 메모리의 량보다 많을 경우, 최적의 성능을 기대하기 어렵다. 본 논문에서는 메모리 사용량이 많은 페이지랭크 응용 프로그램에서 병목이 되는 현상을 실험을 통해 그 요인에 대해 분석하고, 스파크와 함께 타키온을 구성해서 메모리의 효율적 사용을 통해 병목의 요인을 해결하여 18%의 성능향상을 하였다.

핵심주제어 : 빅데이터, 메모리기반 프레임워크, 스파크, 타키온 파일 시스템

Abstract Spark, an in-memory big-data processing framework is popular to use for real-time processing workload. Spark can store all intermediate data in the cluster memory so that Spark can minimize I/O access. However, when the resident memory of workload is larger that the physical memory amount of the cluster, the total performance can drop dramatically. In this paper, we analyse the factors of bottleneck on PageRank Application that needs many memory through experiment, and cluster the Spark with Tachyon File System for using memory to solve the factor of bottleneck and then we improve the performance about 18%.

Key Words : Bigdata, In-memory Platform, Spark, Tachyon File System

1. 서 론

빅데이터 시대에 도래함에 따라, 다방면에서 데이터가 폭발적으로 생성되고 처리되고 있다. 그러나 실제 사회에서 매일 생성되는 대용량의 데이터의 양을 처리하기 위해서는 현재 인 메모리 기술로는 한계가 있다. 따라서 정보산업업계에서 빅데이터 클러스터구축과 그와 관련된 개발

* Corresponding Author : jlee@kau.ac.kr
† 이 논문은 2013년도 정부(산업통상자원부)의 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행된 기초연구 사업임 (No. 20132010101800)
Manuscript received Apr 19, 2016 / revised May 23 ,2016 / accepted June 20, 2016
1) 한국항공대학교 항공전자정보공학부, 제1저자
2) 한국항공대학교 항공전자정보공학부, 제2저자
3) 한국항공대학교 항공전자정보공학부, 제3저자

이 한창이다[1,2,3]. 정부에서도 빅데이터 클러스터 연구에 많은 예산을 투자하고 있다[4]. 그 중, 빅데이터 클러스터를 구축하기위한 기술 중에서, 빅데이터를 저장하고 처리 할 수 있는 프레임워크 기술 개발이 가장 중요하다.

빅데이터 처리 프레임워크 연구의 중심에 하둡[5]이 있으며, 최근에는 스파크[6]가 널리 사용되고 있는 추세이다. 하둡은 디스크기반 스토리지 시스템인 HDFS[7]와 데이터 처리 엔진인 맵리듀스[8]로 구성되어 있으며, 최근 버전에서는 얀[9]을 통하여 분산처리 스케줄링을 할 수 있게 되었다. 하둡과 스파크에 관련해 연구 과정을 살펴보면 다음과 같다. 하둡과 관련해 성능개선을 위한 연구가 활발했지만, 디스크기반이라는 점에 있어서 한계가 뚜렷이 나타났다. 그 결과 디스크 기반의 데이터 처리에서 메모리 기반의 데이터 처리를 위한 플랫폼인 스파크가 현재 빅데이터 프레임워크에서 중요한 연구과제가 되었다. 그 중 하나가 스파크이다. 하지만, 스파크도 메모리 기반 이라는 점에 있어서 빠른 성능을 기대 할 수 있겠지만, 적은 메모리의 용량이 실제 메모리 속도만큼의 성능을 이용 해 데이터를 처리하는데 병목이 된다.

본 논문은 메모리의 부족으로 발생하는 성능제약을 해결하기 위한 방안으로 타키온 파일시스템을 도입하여 효과적인 메모리 사용방안을 제시한다. 메모리 기반의 파일시스템인 타키온[10]을 통해 적은 메모리 용량을 가진 서버환경에서 인메모리기반 프레임워크인 스파크를 활용할 수 있게 한다. 본 논문에서는 스파크만을 이용해서 클러스터를 구성할 때와 스파크와 타키온파일시스템을 함께 이용해 클러스터를 구성할 때 두 가지 경우의 성능 차이에 대해 비교분석하였다. 실험에 사용된 워크로드는 페이지랭크 알고리즘이다. 본 논문의 주요 기여사항은 다음과 같다.

1. 메모리가 적은 서버환경에서의 스파크를 구성할 때, 성능하락과 잡이 실패했던 이유는 Java Garbage Collection (GC) 오버헤드 때문이라는 사실을 실험을 통해 분석.
2. 페이지랭크 워크로드 중, 반복되는 스테이지에서 전체성능에 미치는 요인들에 대해 자세히 분석.
3. 타키온 파일시스템을 스파크와 함께 클러스터로 구성했을 때, 스파크만을 이용해 클러스터를 구성했을 때보다 18%의 성능 향상.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 성능개선 연구대상 프레임워크인 스파크와 타키온에 대해서 설명하고, 3장에서는 실험 워크로드와 실험 환경 세팅에 대해서 설명한다. 4장에서는 실험을 통한 결과 및 분석을 제시하고, 5장에서는 결론과 함께 향후 연구 과제에 대해 논의한다.

2. 이론적 배경

2.1 스파크

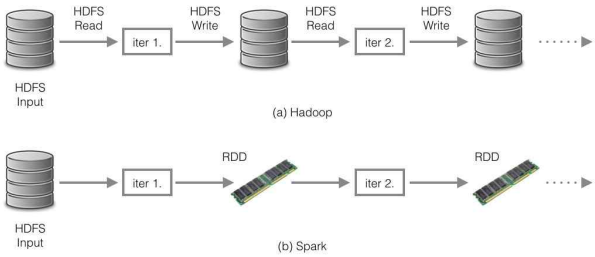

Fig. 1 Comparison between Hadoop and Spark with iteration job workflow

하둡/맵리듀스가 빅데이터에 대한 분석을 용이하게 만들기 시작하였지만, 맵리듀스로는 처리하지 못 하는 어플리케이션들이 여러 가지가 있다. 머신러닝 및 그래프 프로세싱 같은 복잡하고 여러 단계에 걸쳐서 반복적으로 처리되는 어플리케이션과, 프로세싱 중간에 다량의 데이터를 생성하는 특징을 가진 어플리케이션에 대해서는 성능저하가 발생하였다. 그 이유는 데이터 처리가 반복될 때 마다 HDFS, 즉 디스크를 거쳐서 디스크의 읽기/쓰기 양이 많기 때문이다(Fig. 1(a)).

이 문제를 스파크에서는 RDD(Resilient Distributed Datasets)[11]라는 탄력적인 분산 데이터 셋을 통해서 해결했다(Fig. 1(b)). RDD는

한 번 만들어진 후에는 변하지 않는 데이터 셋이다. 스파크는 HDFS와 같은 저장소에서 처리 할 데이터를 읽어 들여 메모리에 Read-Only 방식으로 데이터를 캐싱한다. 캐싱된 데이터를 통해 반복되는 잡에서 디스크까지 읽고/쓰기 할 필요 없이 메모리에 캐싱되어있는 RDD를 통해 연산을 한다. 만들어진 RDD는 계보(Lineage) 라는 방식으로 기록이 되는데, 한번 만들어지고 수정되지 않기 때문에 적은 데이터양을 가지는 계보를 통해서도 잡이 실패했을 때 다시 복구가 되는 장애허용시스템도 쉽게 구축이 가능하다[11].

하지만, RDD를 메모리에 그대로 캐싱하는 것에 있어서 한계가 있다. 많은 상용서버에서는 메모리의 용량을 크게 구성하기 쉽지 않다. 메모리는 디스크장치보다 용량대비 비용이 높기 때문이다. 따라서 본 논문에서는 타키온파일시스템을 활용 해 RDD를 보다 효율적으로 캐싱함으로써 스파크의 단점을 보완해 반복되는 워크로드에서의 성능향상을 하였다.

2.2 타키온파일시스템

타키온파일시스템은 UC Berkley의 Amplab에서 오픈소스로 개발된 빅데이터 분석을 위한 메모리기반 스토리지 시스템이다. 타키온파일시스템은 빅데이터 클러스터 프레임워크 또는 잡들 사이에서 메모리의 속도로 데이터를 공유할 수 있도록 돕는 역할을 한다. 따라서 타키온파일시스템은 디스크 기반의 스토리지 시스템과 빅데이터 분석을 위한 엔진 사이에서 다리역할을 할 수 있게 된다. 또한 타키온파일시스템은 HDFS, GlusterFS, 맵리듀스, 스파크 등의 빅데이터 에코시스템과 함께 호환이 가능하다. 타키온파일시스템은 또한 여타 다른 빅데이터 기반 스토리지 시스템과 비슷하게, 마스터노드와 워커노드들로 구성되며 워커노드들에 데이터가 분산 저장되는 방식으로 처리된다.

본 논문에서는 타키온파일시스템을 스파크에 잡을 처리할 때 생성되는 RDD를 캐싱하는 용도로 사용하였다. 본래 스파크는 RDD를 저장 할 때 JVM Heap 에 RDD를 캐싱한다. 스파크에서 잡이 제출되면 잡 수행에 필요한 메모리는 JVM Heap 형태로 메모리상에 올라온다. 즉, RDD를 저장하는 메모리 공간이 JVM Heap이 되는 것이다. 하지만, 이 JVM Heap에 RDD가 저장될 때, 메모리가 잡에서 요구하는 메모리의 양보다 적으면 Java GC가 발생해서 성능저하에 요인이 된다. 따라서 RDD를 스파크의 JVM Heap에 저장하는 것이 아닌, OFFHEAP 이라고 불리는 타키온파일시스템에 RDD를 직렬화해서 보다 작은 크기로 캐싱한다. OFFHEAP은 타키온파일시스템에 RDD를 저장하는 Spark RDD 캐싱옵션의 이름이고, 또한 JVM Heap이 아니라는 점에서도 OFFHEAP 이라는 이름에 의미가 있다.

RDD를 OFFHEAP 공간에 저장하는 것을 통해 예상되는 결과는 자바가상머신 기반으로 돌아가는 스파크에서 JVM Heap이 부족해서 발생하는 Java Garbage Collection(GC)이 줄어들 것이다. 따라서 메모리가 부족해서 실패하거나 성능저하의 원인이 되는 메모리가 부족한 환경에서 타키온은 성능개선의 해결점으로 적용되어질 수 있다. 실험분석에서 실험을 통해 실제로 Java GC가 줄어드는 것을 확인하고, 성능개선에 효과가 있는지 보다 더 자세한 분석을 통해 확인한다.

3. 실험방법

3.1 워크로드

인메모리 기반 프레임워크인 스파크에서 자주 사용하는 워크로드인 페이지랭크 알고리즘을[12] 이용해서 실험을 진행하였다. 페이지랭크 알고리즘은 구글에서 개발된 알고리즘으로 노드와 엣지로 이루어져 있는 그래프에 대해 반복연산을 통해 각 노드의 랭크를 부여하는 알고리즘이다. 페이지랭크 알고리즘의 특성상 연산에 필요한 데이터양과 워커노드간의 주고받는 데이터량이(셔플에 필요한 데이터량) 많아서 메모리의 부족한 환경에서의 성능개선 연구에 적합한 알고리즘이다. 실험에 사용 될 데이터 셋은 Stanford 대학의 SNAP(Stanford Network Analysis Platform)에서 제공하는 네트워크관계에 대해 기록되어있는 데이터 셋이다[13]. 본 논문에서 페이지랭크 워크

로드를 사용해서 반복되는 스테이지에서 RDD를 캐싱하지 않을 때(NO caching), 메모리에 캐싱할 때(Memory), 타키온파일시스템에 캐싱할 때(TFS: OFFHEAP) 총 3가지로 나누어서 실험한다. 따라서 RDD를 캐싱하는 위치를 통해 전체 잡의 수행시간에 끼치는 영향을 보고, 그 이유에 대해서 분석한다.

3.2 클러스터 환경

클러스터의 서버 환경은 네임노드(마스터노드) 1개, 데이터노드(워커노드) 4개로 구성하였다. HDFS를 구성하는 서버에는 네임노드와 데이터노드가 있는데, 스파크에서 스케줄링을 담당하는 마스터노드를 네임노드에 구성하고, 일을 받아서 태스크를 수행하는 워커노드를 데이터노드들에 구성하였다. 네임노드의 하드웨어 구성은 CPU 3.4GHz Xeon E3-1240V3 Quad-Core Processor, 메모리 8GB, 운영체제는 삼성 SATA3 SSD 120GB, HDFS는 삼성 SATA3 512GB 에 구성했다. 데이터노드는 앞서 설명한 네임노드에서 메모리의 부족한 환경을 구성하기 위해서 메모리를 4GB로 구성하였다.

다음으로는 소프트웨어 구축 환경에 대해 설명한다. 운영체제는 CentOS 6.6 버전이다. 빅데이터 플랫폼 구성은 Fig. 2 와 같이 구성하였다. 실험에 사용될 페이지랭크 워크로드를 위해 필요한 데이터 셋인 텍스트 파일을 하둡 HDFS에 분산 저장 하였다. 이 때 HDFS에 분산 저장되는 복사본의 수는 3이다. 분산처리플랫폼은 스파크를 사용하였다. 또한, 실험 비교를 위해 스파크를 통해 잡을 수행하는 도중에 생성되는 데이터 셋인 RDD를 타키온파일시스템에 저장하기 위해서 메모리에 타키온파일시스템을 구성하였다. 각 프레임워크의 소프트웨어 버전은 Hadoop 2.6.2, Spark 1.5.2, Tachyon 0.7.1 이다.

4. 실험결과 및 분석

Fig. 3 은 페이지랭크 알고리즘을 1GB의 데이터에 대해서 실험한 결과이다. 스파크의 수행메

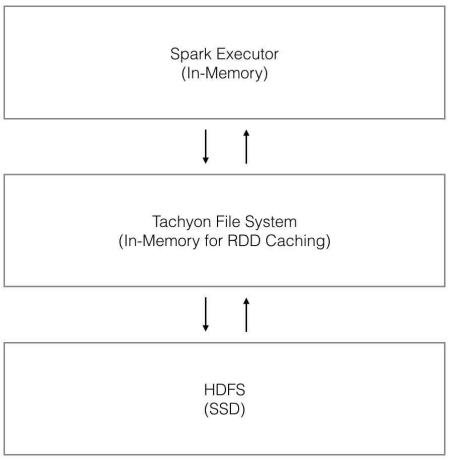

Fig. 2 Cluster Framework with Tachyon for Experiment

모리로는 워커노드당 최대로 설정 할 수 있는 2.7GB로 설정했고, 타키온파일시스템은 300MB로 메모리에 구성하였다. 실험은 세 가지 옵션으로 진행했다. 첫째로, No caching 옵션은 페이지랭크 알고리즘을 반복할 때 발생하는 RDD를 캐싱하지 않을 때이다. 둘째로, Memory 옵션은 JVM Heap 에 캐싱한다. 마지막으로, TFS 옵션은 타키온 파일시스템에(OFFHEAP) RDD를 캐싱하는 옵션이다. Fig. 3 은 잡이 스파크에 제출되고 총 걸린 시간을 No caching 옵션으로 정규화한 그래프이다. Fig. 3에서 메모리에 RDD를 캐싱하는 경우에 오히려 잡이 실패하는 것을 볼 수 있다.

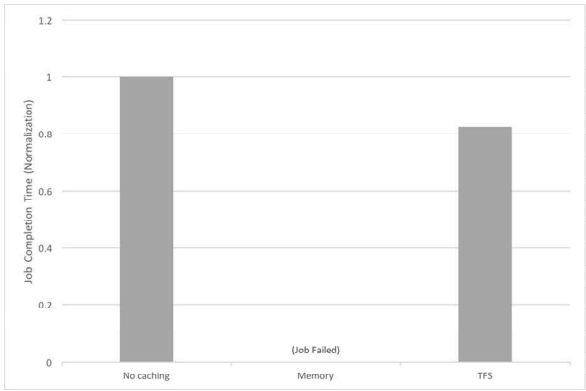

Fig. 3 PageRank Workload Experiments in Spark from each caching option

이 실험결과를 통해 알 수 있는 것은 메모리가 부족한 환경에서 스파크의 잡이 실패한다는 것이다. RDD를 메모리에 캐싱하는 것은 중간데이터를 재사용하므로 잡의 수행시간이 빨라 질 수 있다. 하지만, 스파크의 수행메모리의 총량이 잡이 필요한 메모리보다 적을때, JVM의 Old 영역에 자리 잡고 있는 RDD 때문에 초과로 RDD를 캐싱하거나 실제로 Job을 수행 할 수 있는 메모리가 부족하다. 이 때 JVM은 Java GC를 수행한다. Java GC가 수행시간에 영향을 끼쳐 태스크가 느려지거나 오버헤드가 발생한다. 그로 인해 스테이지가 실패하고 전체적인 잡 수행이 실패하게 된다. Fig. 3 에서 메모리에 RDD를 캐싱한 실험이 이에 해당한다(Memory옵션). 이 때 위해서 발생한 Java GC 시간은 120초였고, GC 오버헤드 제한시간이 지나서 실패하게 되었다.

이를 보완하기 위해서 본 논문에서는 타키온파일시스템에 RDD를 캐싱했다. Fig. 3에서 TFS 옵션이 이에 해당한다. 이 실험에서는 전체 태스크중 가장 긴 Java GC 시간이 8초였다. RDD를 타키온파일시스템에 캐싱함으로 RDD가 직렬화되어서 RDD 크기가 줄어들었기 때문이다. 따라서 TFS 옵션을 통해 캐싱을 하지 않았을 때보다 18%의 성능향상을 얻을 수 있었고, 메모리에 캐싱했을 때 실패했던 잡이 성공한다.

Fig. 4 은 실험결과를 더 자세하게 분석하기 위해서 잡의 스테이지별로 수행시간을 그린 그래프이다. Distinct 스테이지는 텍스트파일을 HDFS에서 읽어서 새로운 RDD를 만들어내는 스테이지이므로 캐싱옵션에 관계없이 수행시간이 비슷한 것을 확인 할 수 있다. flatMap2 스테이지에서는 Distinct 스테이지를 통해서 만들어진 RDD를 셔플읽기를 한 후에 groupByKey, mapValue, join, map, flatMap 과 같은 함수들을 처리한다. 그 후에, 만들어진 RDD를 캐싱한다. 이후에 반복되는 flatMap 스테이지들 에서는 캐싱된 RDD를 읽어서 앞서 언급한 함수들을 처리한다.

Table 1 은 반복되는 스테이지 중에서 RDD를 첫 번째로 재사용하는 flatMap3 스테이지를 캐싱 옵션에 따라서 분류하였다. 하나의 워커노드에서 셔플읽기의 양, 캐싱된 RDD의 양, 셔플스필 된 셔플데이터의 양을 나타낸다. No caching 옵션에서는 RDD를 캐싱하지 않으므로 RDD의 크기가 0이고, 셔플읽기 된 데이터의 크기가 크다. 눈여겨 볼만한 점은 TFS 옵션에서 RDD의 크기가 204MB로 Memory 옵션에 비해 4배 이상 작은 크기라는 것이다. 이는 RDD가 TFS에 캐싱될 때 직렬화되어서 캐싱되기 때문이다. 따라서, 잡을 수행할 수 있는 JVM Heap에 여유 공간이 생긴다.

Memory 옵션에서는 flatMap3 스테이지에서 RDD의 크기가 963MB로 크기 때문에 GC Overhead가 발생해서 워커노드가 죽는 ExecutorLostFailure 에러가 발생하게 된다. 이는 각 스테이지가 실패하는 원인이 되고, 결과적으로 잡이 실패하게 된다. 따라서 그래프에서 flatMap3 스테이지 이후로는 수행시간이 0이다. 반면에 TFS는 잡도 성공하고, 세 옵션 중에서 가장 빠른 수행시간을 보인다. 그 이유는, Memory 옵션에 비해 GC 오버헤드도 줄어들고, No caching 옵션에 비해 셔플읽기, 메모리가 부족해서 셔플데이터가 디스크에 직렬화 되어서 쓰이는 셔플스필 되는 데이터가 적기 때문이다.

본 실험에서 타키온파일시스템에 RDD를 직렬화 해서 저장해서 RDD의 크기를 줄여 메모리의 사용량을 줄여 병목현상들을 해결하였다. 실험의 환경은 메모리가 부족할 경우의 환경이었지만, 메모리가 부족하지 않을 경우에 RDD를 직렬화해서 캐싱하면 메모리 사용량은 낮아지지만 되레 직렬화 하는데 사용되는 CPU 자원 때문에 병목현상이 될 수 있다[14].

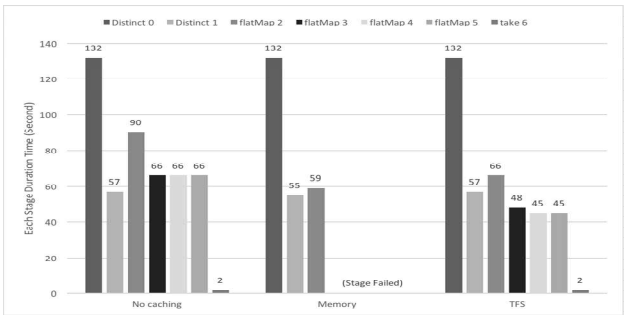

Fig. 4 PageRank Workload Experiments results that indicate each stage duration time from each caching option.

Table 1 Information of flatMap Stage3 in Worker node

Caching Option	Shuffle Read (MB)	RDD Size (MB)	Shuffle Spill Memory (MB)	Shuffle Spill Disk (MB)
No Caching	277.9	0	3600	421.8
Memory	54.8	963.6	2500	276.4
TFS	54.8	204	2400	257.1

5. 결 론

4장에서 PageRank 알고리즘의 실험을 통해서 반복되는 알고리즘, 워커노드들 끼리의 셔플의 양이 많고, RDD를 재사용하는 워크로드에서 메모리의 사용방안이 중요함을 알 수 있다. 스파크에서 잡을 처리 할 때, RDD를 캐싱하는 것이 캐싱하지 않을 때보다 빠르지만, 메모리가 부족한 경우에 RDD를 캐싱하면 Java GC 오버헤드로 인해서 잡이 실패 한다. 따라서 성능향상과 동시에 안정된 인 메모리 기반 프레임워크를 구성하기 위해서 타키온 파일시스템이 대안이 된다. 같은 메모리를 사용하지만 직렬화된 RDD를 직렬화하지 않았을 때보다 적은 크기로 캐싱할 수 있고, 메모리의 성능으로 RDD를 읽고, 쓰기를 할 수 있다. 실험의 결과로 타키온파일시스템에 RDD를 캐싱할 때, 캐싱하지 않을 때보다 18%의 성능향상을 보였고, 메모리에 캐싱할 때 실패했던 잡이 성공하였다.

또한, 분석을 통하여서 수행시간에서 이득을 볼 수 있도록 Java GC 오버헤드를 줄이고, 셔플 읽기, 셔플스필의 양을 줄이는 것이 중요하다는 것을 보였다. 따라서 성능에 주는 요인들을 해결하기 위해서 프레임워크의 개선과 함께 데이터 처리 알고리즘의 연구가 향후 연구에 있어서 중요한 과제가 된다.

References

[1] S. Y. Kim, S. H. Lee, and H. S. Hwang, “A Study of Factors Affecting Attitude Towards Using Mobile Cloud Service”, Journal of the Korea Industrial Information System Society, Vol.18, No. 6, pp.83-94, 2013. *(journal)*

[2] J. W. Kim, “A workflow scheduling based on decision table for cloud computing”, Journal of the Korea Industrial Information System Society, Vol.17, No. 5, pp.29-36, 2012. *(journal)*

[3] J. I. Chaos, and J. H. Ching, “A study on finding influential twitter users by clustering and ranking techniques”, Vol.20, NO. 1, pp.19-26, Feb, 2015. *(journal)*

[4] H. S. Han, H. D. Yang, and K. H. Kim, “Research on Cloud Computing-Based SHE Inorganization Platform Policy”, Vol. 19, No. 5, Oct, 2014. *(journal)*

[5] T. White, “Hadoop: The Definitive Guide”, 2015. *(book)*

[6] Zachariah, Malted, eh ad. "Spark: Cluster Computing with Working Sets." Hotblood10 (2010): 10-10.

[7] Hadoop, Konstantin, eh ad. "The Hadoop distributed file system." Mass Storage Systems and Technologies (MUST), 2010 IEEE 26th Symposium on. IEEE, 2010.

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

[9] Hotblood, Veined Kumara, eh ad. "Apache Hadoop yarn: Yet another resource negotiator." Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013.

[10] Lf, Honduran, eh ad. "Tachyon: Reliable, memory speed storage for cluster computing frameworks." Proceedings of

the ACM Symposium on Cloud Computing. ACM, 2014.
[11] Zachariah, Malted, eh ad. "Resilient distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USETI conference on Networked Systems Design and Implementation. USETI Association, 2012.
[12] Page, Lawrence, eh ad. "The PageRank citation ranking: bringing order to the web." (1999).
[13] http://snap.stanford.edu/data/soc-LiveJournal1.html
[14] http://sujee.net/2015/01/22/understanding-spark-caching/ #.V0ad95E6

이 재 환 (Jae-hwan Lee)

- 정회원
- 서울대학교 전기공학부 공학사
- 서울대학교 전기공학부 공학석사
- University of Maryland 전산학박사
- 현재 : 한국항공대학교 항공전자정보공학부 조교수
- 관심분야 : 분산컴퓨팅, 빅데이터 플랫폼, 고성능 컴퓨팅

최 준 (Jun Choi)

- 한국항공대학교 항공전자정보공학부 공학사
- 현재 : 한국항공대학교 항공전자정보공학부 석사과정 중
- 관심분야 : 분산컴퓨팅, 빅데이터 플랫폼, 머신러닝

구 동 훈 (Dong hun Koo)

- 한국항공대학교 항공전자정보공학부 공학사
- 현재 : 한국항공대학교 항공전자정보공학부 석사과정 중
- 관심분야 : 분산컴퓨팅, 빅데이터 플랫폼, 고성능컴퓨팅