Regular paper
Journal of the KIECS. pp. 367-374, vol. 12, no. 2, Apr. 30 2017, t. 82, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2017.12.2.367

하둡 분산 파일시스템에서의 유연한 노드 관리를 위한 지연된 블록 복제 기법

류우석*

Delayed Block Replication Scheme of Hadoop Distributed File System for Flexible Management of Distributed Nodes

Woo-Seok Ryu*

요 약

본 논문에서는 빅 데이터 처리를 위한 플랫폼인 하둡이 가지고 있는 분산 노드 관리 기법의 문제점을 분석하고 하둡 분산 파일시스템에서 노드 관리를 유연하게 처리하기 위한 기법을 제안한다. 기존의 방법은 클러스터에 포함된 노드가 일시적으로 연결되지 않는 경우 이를 즉시 고장으로 판정함으로 인해 클러스터를 동적으로 구성하지는 못하는 문제가 있다. 본 논문에서 제안하는 지연된 블록 복제 기법은 연결이 끊어진 노드가 추후 클러스터에 쉽게 편입될 수 있도록 노드의 제거를 최대한 지연함으로써 노드 관리의 유연성을 제공한다. 실험을 통해 제안하는 기법이 클러스터의 규모가 변화하는 환경에서 분산 처리 성능에 영향을 거의 미치지 않으면서도 노드 관리의 유연성을 증대시키는 것을 입증한다.

ABSTRACT

This paper discusses management problems of Hadoop distributed node, which is a platform for big data processing, and proposes a novel technique for enabling flexible node management of Hadoop Distributed File System. Hadoop cannot configure Hadoop cluster dynamically because it judges temporarily unavailable nodes as a failure. Delayed block replication scheme proposed in this paper delays the removal of unavailable node as much as possible so as to be easily rejoined. Experimental results show that the proposed scheme increases flexibility of node management with little impact on distributed processing performance when the cluster size changes.

키워드

Hadoop, HDFS, Block Replication, Fault Tolerance
하둡, 하둡 분산 파일 시스템, 블록 복제, 장애 처리

Ⅰ. 서 론

최근 들어 소셜 미디어, 모바일, IoT(: Internet of Things) 등의 기술 발전으로 인해 데이터의 양 또한 급속도로 증가하고 있으며, 이는 고객의 개별 성향 분석, 맞춤형 광고 등 다양한 요구에 따라 빅 데이터 시스템의 활용을 증대시키고 있다[1-3]. 의료 분야는 대표적인 빅 데이터 활용 분야로서 개인화된 의료, 유전

* 교신저자 : 부산가톨릭대학교 병원경영학과
• 접 수 일 : 2017. 04. 10
• 수정완료일 : 2017. 04. 13
• 게재확정일 : 2017. 04. 24
• Received : Apr. 10, 2017, Revised : Apr. 13, 2017, Accepted : Apr. 24, 2017
• Corresponding Author : Woo-Seok Ryu
Dept. of Health Care Management, Catholic University of Pusan,
Email : wsryu@cup.ac.kr

체 분석 등 그 활용 범위를 넓혀가고 있으며[4], 또한 의료기관의 경우에도 병원 경영 전략의 혁신 및 의료 서비스의 질적 향상을 위해서는 접수・진료・입원・수납의 제 과정에서 생성되는 대량의 진료 기록 빅 데이터의 분석이 필수적이다[5].

빅 데이터를 처리하고 분석하는 시스템 중 아파치 하둡(Apache Hadoop)은 대표적인 빅 데이터 분산 처리 플랫폼 중 하나로서 대량으로 생성된 데이터를 분산 노드들에 저장하고 이를 배치 형식으로 처리하는 오픈소스 기반 플랫폼이다. 하둡은 주요 구성 요소로서 첫째는 빅 데이터를 다수의 분산 노드들에 저장하기 위한 파일 시스템인 하둡 분산 파일시스템(Hadoop Distributed File System, 이하 HDFS)[6]이고 둘째는 HDFS에 저장된 빅 데이터를 병렬 처리하기 위한 맵리듀스 프레임워크 (MapReduce Framework)이다. 그 중 HDFS는 동일한 데이터를 여러 노드들에 분산하여 저장하는 형식으로 1000개 이상의 노드에서도 안정적으로 동작할 수 있도록 고가용성을 제공한다[7].

하둡은 다수의 분산 노드들로 구성된 클러스터에서 구동되며, 클러스터에 포함된 노드들은 데이터의 저장 및 분석을 위해 하둡에 종속되어 있는 특징이 있다. 이 노드들은 빅 데이터 처리를 위한 목적으로만 사용되는 하둡 전용 시스템들이며 일반적으로 고장이 나거나 성능이 매우 노후하지 않는 한 클러스터로부터 제거되지는 않는다. 특정 노드로의 접근이 불가한 상태가 발생하면 하둡은 이를 고장이 발생한 노드로 판단하여 즉시 클러스터에서 제거하고 해당 노드가 저장하고 있던 데이터들을 다른 노드들에게 다시 재배치함으로써 데이터의 고가용성을 유지한다. 이와 같이 하둡 노드들은 클러스터 내에서 강 결합되어 있는 특징이 있다.

강 결합된 클러스터의 문제점은 필요시 노드를 선택적으로 클러스터에 편입시키거나 제거하기가 어렵다는 점이다. 즉, 유휴 자원이 일시적으로 확보된 경우 선택적으로 클러스터에 포함하여 분석 성능을 높이거나 클러스터 내의 노드들을 필요시 일시적으로 다른 용도로 전환하여 사용할 수 없으므로 전반적인 시스템의 활용성이 떨어지는 문제가 있다. 본 논문에서는 이를 해결하기 위해 먼저 하둡 노드 관리 기법의 문제점을 분석하고, 다수의 노드들을 필요에 따라 유연하게 클러스터에 편입시키거나 제거함으로써 시스템의 활용성 및 재사용성을 높이기 위한 기법을 제시한다.

이 문제를 먼저 제시한 기존의 연구[8]에서는 유연성 확보를 위해 노드의 상태를 명시적으로 변경하도록 하는 명령 인터페이스를 제안하였으나 본 논문에서는 별도의 명령 인터페이스를 사용하는 대신 노드의 접근 제한이 발생하는 경우 이를 즉시 고장으로 판정하지 않고 재 편입 가능성을 허용함으로써 클러스터의 유연성 및 노드의 활용성을 더욱 증대시키는 기법을 제안한다. 접근 제한 상황이 발생할 때 데이터의 가용성을 유지하는 범위 내에서 데이터의 재배치를 최대한으로 지연함으로써 해당 노드가 추후 클러스터에 다시 포함될 수 있도록 하고 이를 통해 클러스터를 유연하게 관리하는 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 하둡 HDFS의 고가용성을 위한 노드 관리 메커니즘을 분석한다. 3장에서는 노드의 접근 제한 상태를 포함하는 새로운 노드 상태 모델을 제안하고, 4장에서는 접근 제한 상태에서 데이터 재배치를 지연하기 위한 지연된 블록 복제 기법을 제안한다. 5장에서는 제안한 기법을 통해 변경된 클러스터 규모에 따른 하둡의 성능을 비교 분석하고 6장에서 결론을 도출한다.

II. HDFS의 노드 관리 메커니즘 분석

하둡 클러스터는 하나의 네임노드(Name Node)와 다수의 데이터노드(Data Node)로 구성되어 있다. 네임노드는 마스터 노드로서 파일시스템의 네임스페이스를 관리하며, 파일시스템의 디렉토리 및 파일에 대한 메타데이터를 유지한다. 파일은 데이터노드들에 저장되는데, HDFS는 분산 노드 환경에서 데이터의 유실을 방지하기 위해 파일을 고정 사이즈의 블록으로 분할하고 하나의 블록을 여러 데이터노드에 복제하여 저장한다. 이때의 블록 크기의 기본 값은 128MB이며 복제 계수(Replication Factor)의 기본 값은 3으로 설정되어 있다.

HDFS에서의 복제본 배치는 랙 인지(Rack-aware) 기반 복제 배치 전략을 기본으로 하며 그림 1과 같이 클러스터의 네트워크 위상 구조(Network Topology)

에 따라 서로 다른 랙의 다른 노드에 분산 저장하는 방식을 따른다[9]. 이는 기존의 분산 파일 시스템과 차별화되는 특성이며, 파일시스템의 신뢰성, 가용성 및 데이터 접근 성능을 높이는데 매우 중요한 역할을 수행한다[3].

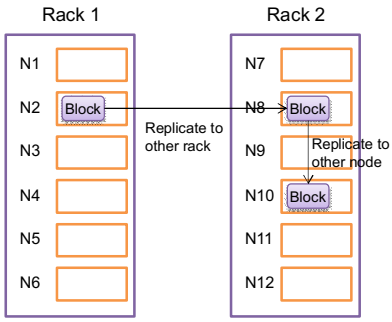

그림 1. HDFS의 블록 복제본 배치 기법
Fig. 1 Block replica placement scheme of HDFS

클러스터에서 데이터노드가 제거되는 경우는 크게 두 가지로 구분할 수 있다. 첫 번째는 예기치 못한 시스템의 고장 등으로 인해 해당 데이터노드가 더 이상 정상 동작하지 않을 때이다. 네임노드는 데이터노드의 상태를 모니터링하기 위해서 각 데이터노드로부터 노드의 상태 및 블록 복제본 정보를 포함한 하트비트(Heartbeat) 신호를 주기적으로 전송받는다. 이때의 주기는 HDFS의 속성으로 설정이 가능하며 기본 값은 3초이다. 만일 하트비트 신호를 특정 데이터노드로부터 10분(기본 값)동안 수신하지 못하면 네임노드는 해당 데이터노드가 고장나서 더 이상 사용하지 못하는 데드 노드(Dead Node)로 판정한다.

데드 노드가 발생하면 블록 관리자(Block Manager)에서 상시 쓰레드로 동작하는 복제 모니터(ReplicationMonitor)가 이를 즉시 감지하고 해당 노드가 저장하고 있던 블록들의 복제 계수를 유지시키기 위하여 블록 복제(Block Replication)를 다시 수행한다. 즉, 이를 통해 데드 노드가 가지고 있었던 블록들을 다시 다른 노드들에게 바로 재배치함으로써 데이터의 가용성을 유지한다.

두 번째 경우는 예상 가능한 장애로서 하드웨어의 교체를 위해 데이터노드를 가동 중단하는 것이다. 이 경우 HDFS의 속성 중 하나인 *dfs.hosts.exclude*에 데이터노드의 이름을 기록하고 하둡의 dfsadmin 프로그램을 통해 노드 재설정을 요청하면 HDFS는 즉시 위임해제 관리자(DecommissionManager)를 통해 해당 데이터노드를 위임 해제(Decommissioning) 상태로 전환한다. 노드가 위임 해제 상태로 전환되면 해당 데이터노드가 보유하고 있던 블록들을 클러스터 내의 다른 데이터노드로 복제하게 된다. 이 작업이 끝나면 노드의 상태는 위임 해제 완료(Decommissioned) 상태가 되며 안전하게 노드를 제거할 수 있다.

앞서 언급한 두 가지 장애 상황 모두 HDFS에서는 유실될 데이터 블록의 복제 계수를 유지 위해 즉시 블록 복제를 수행함으로써 데이터의 가용성을 유지시키는 공통점이 있다. 이 절차는 해당 노드가 더 이상 사용되지 않으며 클러스터에서 영구히 제거됨을 의미하므로, 이 정책을 유지하는 한 기존의 유휴 시스템들을 선택적으로 클러스터에 편입시키거나 일시적으로 제거하기가 어려운 문제가 있다.

III. 하둡 데이터노드의 상태 모델

이 장에서는 하둡 클러스터에 포함된 노드들의 유연성(Flexibility)을 제공하기 위한 기법을 제시한다. 본 논문에서 제시하는 노드의 유연성 확보 기법은 지정된 시간 이상 하트비트 신호를 수신하지 못할 때 해당 노드를 데드 노드로 즉시 판정하는 대신 다른 목적으로 해당 노드를 당분간 중지시킨 것으로 가정하는 것이다. 이를 표현하기 위해 본 논문에서는 하둡 데이터노드의 상태 모델을 그림 2와 같이 제안한다. 그림 2에서 점선으로 표시된 데드(Dead) 상태는 기존의 데이터 노드 상태 모델에서 제거된 것이며 정상(Normal), 위임 해제(Decommissioning) 및 위임 해제 완료(Decommissioned) 상태는 기존에 하둡에 존재하는 상태이다.

본 논문에서 새로 추가한 접근 제한(Inaccessible) 상태는 하트비트 신호의 미수신이 발생한 경우 전이되는 상태로서 기존의 데드(Dead) 상태를 대체하는 상태이다. 접근 제한 상태에서는 즉시 블록의 복제를 수행하는 대신 최대한 시간을 지연하면서 점진적으로 블록 복제를 수행한다. 이 상태에 있는 데이터노드가 다시 가동 가능한 상태가 되어서 하드비트 신호를 네

임노드에 보내게 되면 즉시 정상 상태로 전이된다. 추후 점진적인 블록 복제에 따라 블록들이 모두 다른 노드들로 복제가 되면 그때 데드 노드와 같이 클러스터에서 제거가 된다. 시간을 지연하면서 점진적으로 블록을 복제하기 위한 기법의 상세한 메커니즘은 4장에서 설명한다.

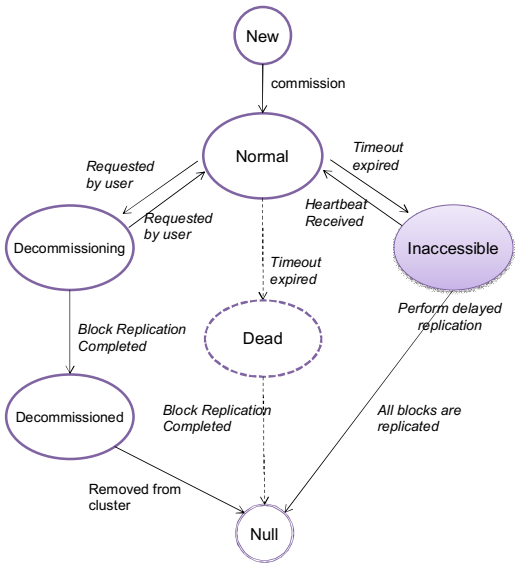

그림 2. 데이터노드의 상태 전이 다이어그램
Fig. 2 State transition diagram of a data node

Ⅳ. 지연된 블록 복제 기법

접근 제한 상태에 있는 데이터노드에 대한 블록 복제를 점진적으로 처리할 때 본 논문의 접근 방법은 블록의 가용성을 유지하는 전제 하에 복제를 최대한 지연하는 것이다. 만일 몇 개의 노드가 연쇄적으로 접근 제한이 되면 특정 블록의 복제본이 모두 접근 가능하지 않을 수 있으며 이 경우 블록의 가용성이 보장받게 되지 못하는 문제가 발생한다. 이에 따라 이 장에서 제안하는 지연된 블록 복제 기법(Delayed Block Replication Scheme)은 블록의 가용성을 모니터링하면서 최소 가용성이 보장받지 못하는 상황이 발생할 때에만 복제를 수행함으로써 점진적인 복제를 가능하게 하는 기법이다.

블록에 대한 최소 가용성을 측정하기 위해 HDFS의 추가적인 속성인 최소 복제 계수를 정의한다. 최소 복제 계수는 일부 노드가 접근 제한이 되는 상황에서도 접근 가능한 블록의 최소값을 의미하며, 그 값은 2 이상 복제 계수 이하로 설정할 수 있다. 값을 2 이상으로 설정한 이유는 만일 최소 복제 계수를 1로 설정하면 해당 노드가 접근 제한이 될 때 더 이상 복제본을 접근할 수 없기 때문이다.

지연된 블록 복제 기법은 접근 제한 노드에 저장되어 있던 블록들을 모니터링 하면서 복제본 수가 최소 복제 계수에 미치지 못하는 블록에 대해서만 블록 복제를 수행하는 기법으로 이 기법의 상세 알고리즘은 다음과 같다.

- Step 1 : 노드가 접근 제한이 발생하면 해당 노드가 가지고 있는 블록 각각에 대해 접근 가능한 블록 복제본의 개수를 계산한다.
- Step 2 : Step 1에서 복제본 개수가 최소 복제 계수에 미달하는 블록이 있으면 해당 블록을 가지고 있던 접근 제한 노드 중 접근 제한 시간이 가장 오래된 노드의 블록 정보를 무효화하고 정상 노드를 통해서 해당 블록의 복제를 수행한다.
- Step 3 : 접근 제한 노드에서 하트비트 신호가 다시 수신되면 정상 상태로 전환하고 가지고 있는 블록들에 대한 일관성 체크를 수행한다. 이때, 블록 복제본의 개수가 블록 복제 계수보다 큰 경우 하둡 밸런서(Balancer)를 통해 개수가 자동으로 조정된다.
- Step 4 : 접근 제한 노드가 가지고 있는 블록들이 모두 무효화되면 해당 노드를 클러스터에서 제거한다.

그림 3은 노드의 접근 제한에 따른 지연된 블록 복제 기법의 알고리즘을 상세하게 설명하기 위한 예시이다. 이 그림에서는 복제 계수와 최소 복제 계수를 각각 3과 2로 설정한 것으로 가정한다. 그림 3 (a)는 6개의 데이터노드로 구성된 클러스터의 초기 상태로서 8개의 블록이 3개씩 복제가 된 것을 확인할 수 있다. 이때 노드6이 접근 제한 상태가 되면 해당 노드가 가지고 있는 블록들에 대한 지연된 블록 복제가 그림 3 (b)와 같이 시작되는데 노드6이 가지고 있는 블록 B3, B5, B6, B7은 다른 노드를 통해 최소 복제 계수

를 만족하므로 즉시 블록 복제가 수행되지는 않는다.

그림 3 (c)와 같이 노드2가 추가적으로 접근 제한이 되면 B3과 B7은 접근 가능한 블록이 하나밖에 남지 않으며 이에 최소 복제 계수보다 작아지게 된다. 이때 두 블록에 대해서는 최소 복제 계수를 만족시켜 주기 위해서 복제를 수행하고 먼저 접근 제한이 발생한 노드6이 가지고 있던 해당 블록들을 무효화 처리한다. 추가적으로 그림 3 (d)와 같이 노드5가 접근 제한이 발생하면 B5, B6, B7의 복제본 개수가 최소 복제 계수에 미달하므로 이에 대한 블록 복제를 수행하고 노드2와 노드6이 가지고 있던 해당 블록들을 무효화한다. 이때 노드6은 더 이상 유효한 블록을 가지고 있지 않으므로 클러스터에서 제거된다. 이를 통해 접근 제한된 노드를 즉시 제거하지 않아도 최소한의 데이터 가용성을 유지할 수 있으며 해당 노드들은 언제든지 다시 클러스터에 재 편입되면 가지고 있던 블록들을 다시 사용할 수 있으므로 클러스터에서 노드들을 보다 유연하게 관리할 수 있다.

Ⅴ. 실험 결과

이 장에서는 클러스터의 데이터노드의 접근 제한에 따라 클러스터의 규모가 변경될 때 발생하는 하둡의 성능 변화를 분석하고자 한다. 실험 환경으로 2 코어 펜티엄 프로세서가 장착된 7개의 PC를 1기가비트 스위칭 허브로 연결하였으며, 소프트웨어로는 우분투 14와 하둡 2.7.3을 이용하였다.

클러스터 규모에 따른 성능 분석을 위해 두 가지 하둡 클러스터를 구성하였다. 첫 번째는 그림 3 (a)와 유사하게 1대의 네임노드와 6대의 데이터노드로 구성된 클러스터이고 두 번째는 그림 3 (d)와 유사하게 1대의 네임노드와 3대의 데이터노드로 구성된 클러스터이다. 지연된 복제 블록 기법을 적용하기 위해 첫 번째 클러스터에서는 복제 계수를 3으로 설정하였으며, 두 번째 클러스터는 복제 계수를 2로 설정하였다.

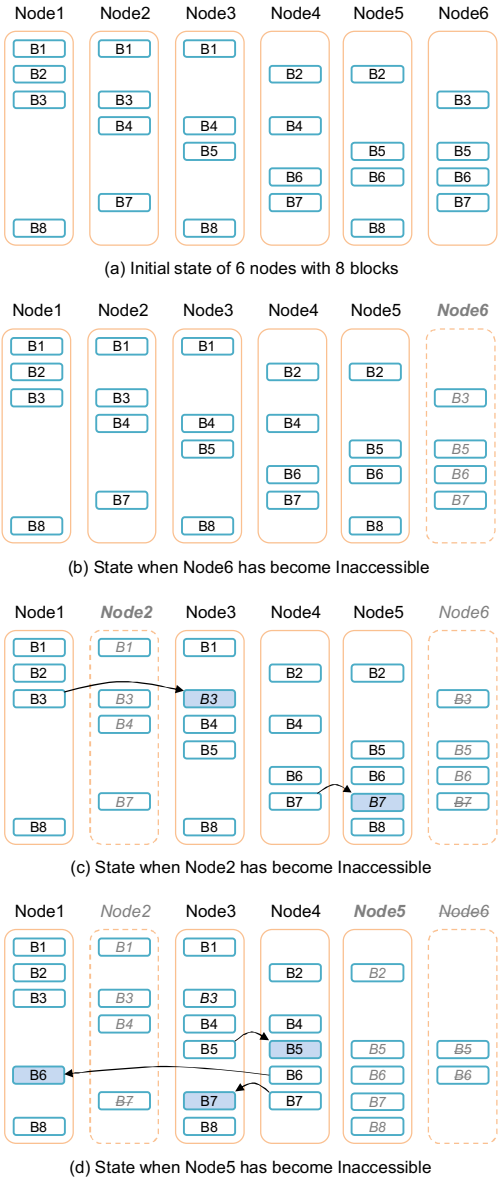

그림 3. 지연된 블록 복제 기법 예시
Fig. 3 An example of delayed block replication

성능 평가를 위한 데이터 셋은 임의 생성된 256MB, 512MB, 1GB, 2GB 네 개의 텍스트 파일을 사용하였으며, 하둡 잡(Job)은 패키지에 기본으로 포함되어 있는 wordcount 예제 프로그램을 이용하였다. 잡의 실행은 단일 잡을 실행하는 경우와 하둡에 충분

한 부하를 발생시킨 후 잡을 실행하는 경우로 구분하였다. 이때의 시스템 부하는 10GB 파일에 대해 wordcount 잡을 먼저 실행시키는 형태로 설정하였다.

그림 4는 각 클러스터에서 네 개의 파일에 대한 단일 잡 실행 성능을 도시한 결과이다. 3 데이트 노드 클러스터가 약간 성능이 높게 나오는데 그 이유는 노드의 개수가 적음으로 인해 리듀스(Reduce) 작업에 따른 추가적인 노드 간 네트워크 전송 비용이 줄어들었기 때문이다.

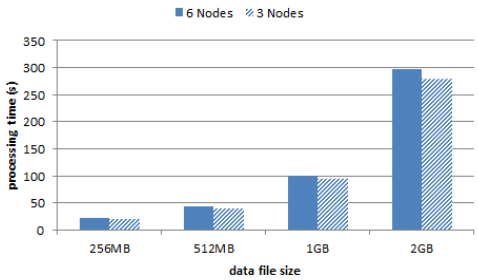

그림 4. 단일 잡의 실행 성능 비교
Fig. 4 Performance comparison of single job

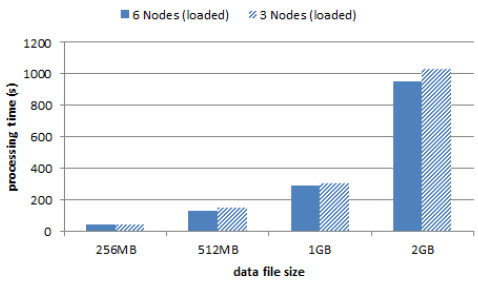

그림 5. 시스템 부하 상태에서 잡의 성능 비교
Fig. 5 Performance comparison of job when heavily loaded

보다 더 일반적인 경우로서 하둡에 시스템 부하를 발생시켰을 때의 비교는 그림 5와 같이 6 데이터 노드 클러스터가 다소 성능이 높게 측정되었으나 그 차이는 크지 않으며 이는 클러스터의 규모에 따라 발생하는 차이라 볼 수 있다. 즉, 본 논문이 제안하는 지연된 블록 복제 기법은 하둡의 전체적인 성능에 큰 영향을 미치지 않으면서도 노드 제거 시 즉시 블록을 복제하지 않으므로 저장 공간 관리의 효율성을 제공하고 이를 통해 클러스터의 규모의 증대/감소 시 블록 복제 횟수를 줄임으로서 보다 유연하게 클러스터를 관리할 수 있음을 확인 가능하다.

Ⅵ. 결 론

본 논문에서는 하둡 클러스터에 있는 데이터노드의 관리를 위한 새로운 노드의 상태 모델을 제안하고 접근이 제한된 노드에 대한 지연된 블록 복제 기법을 제안하였다. 제안한 기법을 통해 접근이 제한된 노드에 대한 점진적인 복제 처리가 가능하게 됨으로써 클러스터 내에 포함된 노드들을 보다 유연하게 활용할 수 있으며 기존에 사용하는 시스템들 또한 유연하게 클러스터에 편입시키는 것이 가능하다. 또한 실험 평가를 통해 제안한 기법이 시스템의 성능에 큰 영향이 미치지 않음을 입증하였다. 본 논문의 향후 연구로서 제안한 기법의 알고리즘을 최적화하고 보다 다양한 상황에서의 실험 평가를 통해 본 기법을 추가 검증하는 것이 필요하다.

감사의 글

이 연구는 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구 임(No. NRF-2016R1C1B1012364).

References

[1] H. Yoon, “Development of Contents on the Marine Meteorology Service by Meteorology and Climate Big Data,” *J. of The Korea Institute of Electronic Communication Sciences*, vol. 11, no. 2, 2016, pp. 125-138.

[2] H. Chen, R. Chiang, and V. C. Storey, “Business intelligence and analytics: From big data to big impact,” *MIS Quarterly*, vol. 36, no. 4, 2012, pp. 1165-1188.

[3] C. Ryu, “Context Inference and Sensor Data Classification of Big Data Stream Environment,” *J. of The Korea Institute of Electronic Communication Sciences*, vol. 9, no. 10, 2014, pp. 1079-1085.

[4] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," *Health Information Science and Systems*, vol. 2, no. 1, 2014, pp. 1-10.
[5] J. Choi, "Utilization value of medical Big Data created in operation of medical information system," *J. of The Korea Institute of Electronic Communication Sciences*, vol. 10, no. 12, 2015, pp. 1403-1410.
[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," In *Proc. IEEE Symp. on Mass Storage Systems and Technologies (MSST)*, NV, USA, May 2010, pp. 1-10.
[7] D. Borthakur, J. Sarma, and J. Gray, "Apache Hadoop Goes Realtime at Facebook, " In *Proc. the 2011 ACM SIGMOD Int. Conf. on Management of data*, Athens, Greece, 2011, pp. 1071-1080.
[8] W. Ryu, "Flexible management of data nodes for hadoop distributed file system," In *Proc. Int. Conf. on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2017)*, Venice, Italy, 2017.
[9] T. White, "*Hadoop: The definitive guide, 4th Edition*," O'Reilly Media, Inc., 2015.

저자 소개

류우석(Woo-Seok Ryu)

1997년 부산대학교 컴퓨터공학과 졸업 (공학사)
1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)
2012년 부산대학교 대학원 컴퓨터공학과 졸업(공학박사)
2013년 ~현재 부산가톨릭대학교 병원경영학과 조교수
※ 관심분야 : 의료정보, 의학용어, U-Health, 빅데이터, 하둡 플랫폼