한국정보전자통신기술학회논문지(jkiiect)'19-4 Vol.12 No.2
논문 19-12-02-127 http://dx.doi.org/10.17661/jkiiect.2019.12.2.127

빅데이터 처리율 향상을 위한 인-메모리 기반 하이브리드 빅데이터 처리 기법 연구

이협건**, 김영운*, 김기영

Study of In-Memory based Hybrid Big Data Processing Scheme for Improve the Big Data Processing Rate

Hyeopgeon Lee*, Young-Woon Kim, Ki-Young Kim

요 약 IT기술의 발달로 인해 생성되는 데이터의 양은 매년 기하급수적으로 증가하고 있으며, 이에 대한 대안으로 분산시스템과 인-메모리 기반 빅데이터 처리 기법의 연구가 활발히 이루어지고 있다. 기존 빅데이터 처리 기법들의 처리 성능은 노드의 수와 메모리 용량이 증가될수록 보다 빠르게 빅데이터 처리한다. 그러나 노드의 수의 증가는 빅데이터 인프라 환경에서 장애발생 빈도가 높아지며, 인프라 관리 포인트 및 인프라 운영비용도 증가된다. 또한 메모리 용량의 증가는 노드 구성에 대한 인프라 비용이 증가된다. 이에 본 논문에서는 빅데이터 처리율 향상을 위한 인-메모리 기반 하이브리드 빅데이터 처리 기법을 제안한다. 제안하는 기법은 분산시스템 처리기법에 Combiner 단계를 추가하고, 그 단계에서 인-메모리 기반 처리 기술을 적용하여 기존 분산시스템 기반 빅데이터 처리기법에 비해 빅데이터 처리시간을 약 22% 감소시켰다. 향후, 제안하는 기법의 실질적인 검증을 위해 더 많은 노드로 구성된 빅데이터 인프라 환경에서의 현실적 성능평가가 필요하다.

Abstract With the advancement of IT technology, the amount of data generated has been growing exponentially every year. As an alternative to this, research on distributed systems and in-memory based big data processing schemes has been actively underway. The processing power of traditional big data processing schemes enables big data to be processed as fast as the number of nodes and memory capacity increases. However, the increase in the number of nodes inevitably raises the frequency of failures in a big data infrastructure environment, and infrastructure management points and infrastructure operating costs also increase accordingly. In addition, the increase in memory capacity raises infrastructure costs for a node configuration. Therefore, this paper proposes an in-memory-based hybrid big data processing scheme for improve the big data processing rate. The proposed scheme reduces the number of nodes compared to traditional big data processing schemes based on distributed systems by adding a combiner step to a distributed system processing scheme and applying an in-memory based processing technology at that step. It decreases the big data processing time by approximately 22%. In the future, realistic performance evaluation in a big data infrastructure environment consisting of more nodes will be required for practical verification of the proposed scheme.

Key Words : Big Data, Hadoop, Distributed File System, In-Memory Big Data Processing Scheme, MapReduce

1. 서론

빅데이터 처리 기법은 방대한 양의 데이터를 보다 빠르고 정확하게 처리해야만 한다. 이러한 요구사항을 반영하기 위한 초기 방안은 구성되는 시스템의 성능을 높이는 Scale-Up 방식을 선택하여 운영되었다. 그러

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning(NRF-2017R1D1A1B03034729)
**Corresponding Author : Dept. of Data Analysis, Seoul Gangseo Campus of Korea Polytechnic (hglee67@kopo.ac.kr)
*Dept. of Data Analysis, Seoul Gangseo Campus of Korea Polytechnic
Dept. of Computer Software, Seoil University
Received April 09, 2019 Revised April 15, 2019 Accepted April 15, 2019

나 매년 새롭게 생성되는 데이터의 양은 기하급수적으로 증가하게 되었고, 이에 대한 대안으로 분산시스템 기반 빅데이터 처리기법과 인-메모리 기반 빅데이터 처리기법의 연구가 활발히 이루어지고 있다[1].

분산시스템 기반 빅데이터 처리 기법과 인-메모리 기반 빅데이터 처리 성능은 노드의 수와 노드의 성능과 밀접한 관계가 있다. 빅데이터 처리 기법에 적용되는 노드의 수와 메모리 용량이 증가될수록 빅데이터 처리 성능은 증가한다[1, 2].

그러나 노드의 수의 증가는 빅데이터 인프라 환경에서 발생 가능한 장애발생 빈도가 높아지고, 이로 인해 인프라 관리 포인트 및 인프라 운영에 필요한 비용도 증가된다. 또한 메모리 용량의 증가는 빅데이터 처리에 필요한 노드 구성에 대한 인프라 비용이 증가된다[3].

이에 본 논문에서는 빅데이터 처리율 향상을 위한 인-메모리 기반 하이브리드 빅데이터 처리 기법을 제안한다. 제안하는 기법은 맵리듀스의 주요 단계에 Combiner 단계를 추가하고, 그 단계에서 인-메모리 기반 처리 기술을 적용하여 기존 분산시스템 기반 빅데이터 처리기법에 비해 노드의 수를 감소시키고, 빅데이터 처리율을 약 22% 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 사용 중인 분산시스템 기반 빅데이터 처리 기법과 인-메모리 기반 빅데이터 처리 기법을 살펴보고, 요구사항을 분석한다. 3장에서는 본 논문에서 제안하는 빅데이터 처리율 향상을 위한 인-메모리 기반 하이브리드 빅데이터 처리 기법을 제시한다. 4장에서는 제안한 기법의 성능을 분석하고, 마지막 5장에서는 결론을 제시한다.

2. 관련연구

본 장에서는 빅데이터 처리 방식에 따라 구분하여 가장 보편적으로 많이 사용되는 분산시스템 기반 빅데이터 처리 기법과 인-메모리 기반 빅데이터 처리 기법에 대해 살펴본다. 또한 연구된 결과를 기반으로 제안하는 빅데이터 처리 기법의 요구사항을 도출한다.

2.1 분산시스템 기반 빅데이터 처리 기법

분산시스템 기반 빅데이터 처리 기법[4, 5]은 방대한 양의 데이터를 처리하기 위해 기존의 분산시스템을 활용한다. 빅데이터 처리 기법에 많이 사용되는 분산시스템으로 크게 하둡 분산 파일 시스템을 활용한 빅데이터 처리 기법이 GlusterFS를 활용한 빅데이터 처리 기법 보편적으로 많이 사용되고 있다. 하둡 분산 파일 시스템을 활용한 빅데이터 처리 기법은 맵리듀스 프레임워크를 적용하여 빅데이터를 처리한다. 이 맵리듀스 프레임워크는 분산 파일 시스템에 저장된 방대한 데이터를 정제 및 처리, 분석하는 역할을 수행한다. [그림 1]은 맵리듀스 프레임워크 기반 빅데이터 처리 기법을 나타낸다.

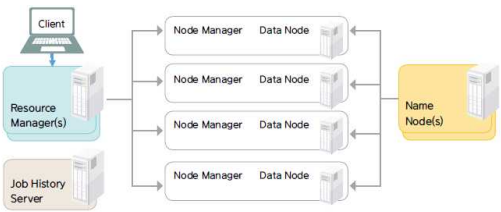

그림 1. 맵리듀스 프레임워크 기반 빅데이터 처리 기법
Fig. 1. Mapreduce frameworks based big data processing scheme

맵리듀스 프레임워크 기반 빅데이터 처리 기법은 하둡에서 제공하는 Resource Manager, Job History Server, Node Manager를 통해 방대한 양의 데이터가 저장되는 데이터 노드들과 그 데이터 노드들의 메타 정보가 저장되는 네임 노드를 관리한다. 이 맵리듀스 프레임워크의 주요 특징은 데이터 노드의 수가 증가될수록 빅데이터 처리 시간도 감소한다. 그러나 인프라 환경은 데이터 노드의 수가 증가될수록 장애가 발생할 확률이 증가되기 때문에 적절하게 구성해야 한다.

GlusterFS[2, 6]를 활용한 빅데이터 처리 기법은 앞서 설명한 하둡 분산 파일 시스템을 활용한 빅데이터 처리 기법과 유사한 분산 파일 시스템으로 구성된다. GlusterFS는 하둡 분산 파일 시스템과 달리 데이터가 저장되는 노드들에 대한 메타데이터를 별도로 생성하지 않으며, 별도의 시스템이나 기술을 통해 구축하길 권고한다. 이로 인해 네임노드의 장애로 인한 노드정보 유실로 인한 데이터 소실과 같은 문제는 하둡 분산 파일 시스템에 발생하기 쉽지만, GlusterFS는 잘 발생하

지 않는다. [그림 2]는 GlusterFS 기반 빅데이터 처리 기법을 나타낸다.

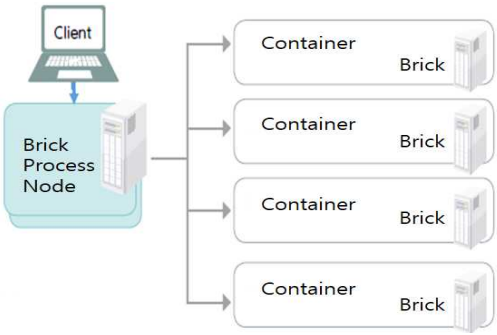

그림 2. GlusterFS 기반 빅데이터 처리 기법
Fig. 2. GlusterFS based big data processing scheme

[그림 2]에서 GlusterFS 기반 빅데이터 처리 기법은 별도로 구성하는 Brick Process Node를 통해 저장되는 데이터들을 하둡의 데이터 노드와 같은 역할을 수행하는 Container들을 통해 관리한다. 저장되는 데이터는 GlusterFS에서 정의한 Brick 단위로 저장된다.

2.2 인-메모리 기반 빅데이터 처리 기법

인-메모리 기반 빅데이터 처리 기법[2, 7]은 분산 저장된 노드들의 메모리를 활용하여 방대한 양의 데이터를 처리하는 기법이다. 인-메모리 기반 빅데이터 처리 기법에서 보편적으로 많이 사용되는 방법은 스파크를 이용하여 방대한 양의 데이터를 처리하는 것이다.

스파크는 맵리듀스의 다음 단계로 개발된 기술이며, 맵리듀스보다 높은 수준의 API를 제공한다. API 제공으로 스파크는 보다 로직 중점적인 빅데이터 프로그래밍이 가능하다. 스파크의 빅데이터 처리 성능은 데이터가 저장된 노드들의 메모리를 기반으로 동작한다. 이로 인해 앞서 설명한 분산시스템 기반 빅데이터 처리 기법에 비해 약 100배 이상 빠르다. 이러한 처리 성능으로 인해 스파크는 기존의 분산시스템 기반 빅데이터 처리 기법에 비해 빅데이터 처리를 위한 대기 시간이 낮아 방대한 양의 데이터를 거의 실시간 처리와 가깝게 수행한다. [그림 3]은 인-메모리 기반 빅데이터 처리 기법을 나타낸다.

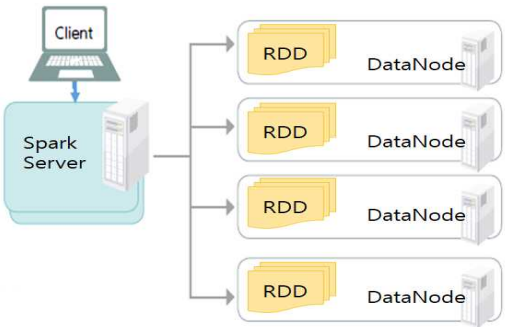

그림 3. 인-메모리 기반 빅데이터 처리 기법
Fig. 3. In-Memory based big data processing scheme

[그림 3]에서 인-메모리 기반 빅데이터 처리 기법은 데이터 노드들의 메모리를 통해 데이터를 처리한다. 메모리를 통한 처리 방법은 저장된 데이터가 변환 및 처리될 때마다 스파크에서 정의한 RDD(Resilient Disributed Data) 데이터 구조를 통해 메모리에 저장하며 데이터를 처리한다.

2.3 요구사항 분석

본 절에서는 앞서 설명한 분산시스템 기반 빅데이터 처리 기법과 인-메모리 기반 빅데이터 처리 기법에 대한 문제점을 도출하고, 요구사항을 분석한다.

첫째, 분산시스템 기반 빅데이터 처리 기법의 빅데이터 처리 성능은 노드가 많지 않으면 처리율이 감소한다. 분산시스템 기반 빅데이터 처리 기법의 노드의 수는 빅데이터 처리 속도와 밀접한 관계가 있으며, 노드의 수가 많을수록 처리 성능은 증가한다. 그러나 노드의 수의 증가는 빅데이터 인프라 환경에서 발생 가능한 장애발생 빈도가 높아지고 인프라 관리 포인트가 증가되어 인프라 운영에 필요한 비용이 증가된다.

장애가 발생할 확률(P), 노드 전체의 수를 40대, 3중화로 분산 저장되며, 장애가 발생한 노드의 수($ErrNode$)로 정의하면, 장애가 발생한 확률은 수식(1)과 같다.

$$P = ErrNode\left(\frac{e_1}{n}\right)\left(\frac{e_1-1}{n-1}\right)\left(\frac{e_1-2}{n-2}\right) \quad (1)$$

수식(1)에서 장애가 발생한 노드의 수를 10대로 정

의하면, 약 0.0025% 확률로 장애가 발생할 수 있음을 예측할 수 있다.

수식(2)는 손실되는 블록의 수(*ErrBlock*)를 나타낸다.

$$ErrBlock = P\frac{DefaultBlockSize}{DataStroreSize} \quad (2)$$

수식(2)에서 저장된 데이터 크기를 100TB, 데이터 노드에 저장되는 기본 블록 크기를 64MB로 정의하면, 손실되는 블록의 수는 약 3,906개가 발생된다. 이러한 저장된 블록의 손실은 데이터의 신뢰성이 중요한 시스템에서 매우 치명적이다.

따라서 빅데이터 처리 기법은 장애발생 빈도를 감소시키며, 보다 빠른 빅데이터 처리가 가능해야 한다.

둘째, 인-메모리 기반 빅데이터 처리 기법의 인프라 구축비용은 높다. 인-메모리 기반 빅데이터 처리 기법은 데이터가 저장되는 노드들의 메모리를 사용하여 데이터를 처리한다. 이러한 특성은 처리해야할 데이터의 용량보다 큰 메모리의 용량으로 노드를 구성해야만 정상적인 빅데이터 처리가 가능하다. 용량이 큰 파일에 대한 빅데이터 처리는 인-메모리 기반 빅데이터 처리 기법을 적용하기에는 많은 비용이 발생한다.

따라서 빅데이터 처리 기법은 인-메모리 기반 빅데이터 처리 기법과 분산시스템 기반 빅데이터 처리 기법이 융합된 하이브리드 방식의 빅데이터 처리 기법이 요구된다.

3. 제안하는 빅데이터 처리 기법

제안하는 인-메모리 기반 하이브리드 빅데이터 처리 기법은 앞서 분석된 요구사항에 맞춰 인-메모리 기반 빅데이터 처리 기법과 분산시스템 기반 빅데이터 처리 기법이 융합된 하이브리드 방식으로 빅데이터를 처리한다. [그림 4]는 제안하는 빅데이터 처리 기법의 처리 프로세스를 나타낸다.

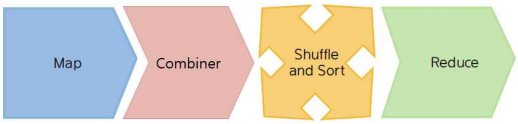

그림 4. 제안하는 빅데이터 처리 기법의 처리 프로세스
Fig. 4. The process of big data processing scheme

제안하는 빅데이터 처리 기법은 하둡을 활용하며, 분산시스템은 하둡 분산 파일 시스템을 사용하며 맵리듀스를 통해 빅데이터를 처리한다. 제안하는 빅데이터 처리 기법의 프로세스는 맵리듀스 프레임워크의 주요 단계를 진행한다. 맵리듀스의 주요 단계는 Map, Combiner, Shuffle and Sort와 Reduce이다. Map 단계는 분석 및 처리될 데이터들을 용도에 맞게 1차 정제 및 가공하는 역할을 수행한다. Combiner 단계는 본 논문에서 제안하는 단계로 앞서 실행되는 맵리듀스의 Map 단계에서 처리되는 방대한 양의 데이터들을 Shuffle and Sort 단계로 전송하기 전에 Reduce 단계와 유사하게 데이터를 처리한다. Shuffle and Sort 단계는 맵리듀스 프레임워크에서 자동적으로 수행되는 단계로 Map 단계나 Combiner 단계에서 전송된 데이터를 취합·그룹으로 묶은 뒤 정렬한다.

Reduce 단계는 앞서 실행된 Shuffle and Sort 단계에서 전송받은 데이터를 기반으로 처리 및 분석을

```
1   //Mapper로부터 전달된 분석대상 가져오기
2   Load Context
3
4   //메모리에 올릴 데이터 모델 생성
5   DataModel dm = new DataModel();
6
7   //Mapper로부터 전달된 데이터 전체 읽기
8   while(Context.nextKey){
9     //레코드 단위 데이터 읽기
10    Object Key = Context.currentKey;
11    Object Value = Context.currentValue;
12
13    //데이터 모델에 데이터 저장하기
14    ds.add(Key, Value);
15  }
16
17  //메모리에 데이터 올리기
18  boolean success = Store(ds, Contex);
19
18  if (success){
19    //Combiner 실행
18    ExecuteCombiner();
19  }
20
```

그림 5. Combiner 단계의 데이터 처리를 위한 의사코드
Fig. 5. The pseudo code of combiner step for data processing

수행하고, 그 결과를 생성한다.

제안하는 빅데이터 처리 기법은 주요 단계 중 Combiner 단계에 인-메모리 기반 처리 기술을 적용하여 보다 빠른 빅데이터 처리를 수행하도록 한다. [그림 5]는 Combiner 단계의 데이터 처리를 위한 의사코드를 나타낸다.

Combiner 단계는 Map 단계에서 전달받은 데이터를 메모리를 올리기 위해 가장 먼저 맵리듀스의 공용 데이터 저장소와 같은 역할을 수행하는 Context 객체를 로딩한다. Context 객체는 Map 단계에서 1차 처리된 데이터들이 모두 저장되는 객체이며, 메모리에 저장하기 위한 데이터 모델은 키와 값 형태로 구성된다. 키와 값 형태로 데이터 모델을 구성되는 이유는 맵리듀스에서 데이터 전달하는 기본 데이터 모델이 키와 값 형태로 저장되기 때문이다. [그림 6]은 메모리에 저자장되는 키와 값 형태의 데이터 모델을 나타낸다.

```
{Mapper Thread ID : [
 {Key : Value},
 {Key : Value},
 {Key : Value},
]}
```

그림 6. 메모리에 저장되는 키·값 형태 데이터 모델
Fig. 6. The data model of a pair of Key·Value type

[그림 6]에서 데이터 모델은 Map 함수가 실행되는 Mapper 객체의 쓰레드마다 키와 값 형태의 배열로 저장되며, 데이터 구조는 데이터 처리 속도에 적합한 JSON 사용한다.

Combiner 단계는 Mapper 객체로부터 전달받은 데이터들을 저장할 데이터 모델 객체를 생성한 뒤, Context 객체에 저장된 데이터를 읽기 시작한다. 읽은 데이터는 생성된 데이터 모델에 저장된다. 생성한 데이터 모델은 Mapper 객체로부터 전달받은 데이터들이 모두 저장이 완료되면 메모리에 저장된다. Combiner 단계는 메모리 저장이 완료되면, 미니 리듀서와 같은 역할을 수행하는 Combiner를 실행하며 종료한다.

4. 성능평가

본 장에서는 제안하는 인-메모리 기반 하이브리드 빅데이터 처리 기법에 대한 성능평가를 수행한다. 성능평가 항목은 앞서 관련 연구에서 분석한 요구사항을 기반으로 분산시스템 기반 빅데이터 처리 기법 (*Distributed Big Data Processing Scheme, DBPS*)과 제안하는 빅데이터 처리 기법(*Hybrid Big Data Processing Scheme, HBPS*)의 노드 수 증가에 따른 빅데이터 처리시간과 장애 발생 확률에 따른 소실되는 데이터의 크기를 비교 분석한다.

4.1 실험 환경

〈표 1〉은 성능평가를 위한 주요 환경 구성을 나타낸다.

표 1. 성능평가를 위한 주요 환경 구성
Table. 1. Environment for performance analysis

항목	내용
빅데이터 인프라 환경	• 실행환경 : 하둡 완전 분산모드 • 설치버전 : 하둡 2.82 • 네트워크 : 10G
서버(노드) 스펙	• Dell PowerEdge R730 - Intel Xeon 3.4Ghz 8Core 64G RAM
노드 구성	• 네임노드 : 1대 • 2차 네임노드 : 1대 • 데이터노드 : 3대-10대
분석 대상	• 아파치 웹 로그 • 용량 : 3G
분석 기간	• 최근 31일
맵리듀스 잡	• 비정상적 URL 접속 횟수 분석

성능평가를 위한 빅데이터 인프라 환경은 하둡을 기반으로 완전 분산 모드를 구축하여 검증한다. 성능평가에 사용된 하둡의 버전으로 정식 배포된 2.82 버전을 사용한다. 하둡 완전 분산 모드로 구성되는 노드는 최소 5대부터 12대까지 구성된다. 분석되는 데이터들의 메타 정보가 저장되는 네임노드와 2차 네임노드는 각 1대씩 구성하여 총 2대를 사용한다. 데이터 노드는 성능평가를 위해 최소 3개의 데이터 노드부터 최대 10대의 데이터 노드까지 증가시키며 성능 평가한다. 빅데이터 처리에 대한 분석 대상은 약 3G 용량의 아파치 웹

서버 로그이며, 2018년 1월부터 12월까지의 데이터이다. 빅데이터 처리에 대한 분석 기간은 12개월 데이터 중 최근 31일 데이터를 대상으로 추출 및 분석을 수행하며, 맵리듀스 잡의 수행 내용은 비정상적인 URL 접속 횟수를 분석하는 잡이다.

4.2 빅데이터 처리시간 분석

빅데이터 분석 처리 시간 분석은 데이터 노드를 최소 3개부터 10까지 증가시키며, 맵리듀스 잡의 실행시간을 측정하고, 그 결과를 비교 분석한다. 분석 대상은 아파치 웹 로그 중 최근 31일의 데이터로 정의한다. 〈그림 7〉과 〈표 2〉는 제안하는 기법과 비교 대상과의 빅데이터 처리시간 비교 분석한 결과를 나타낸다.

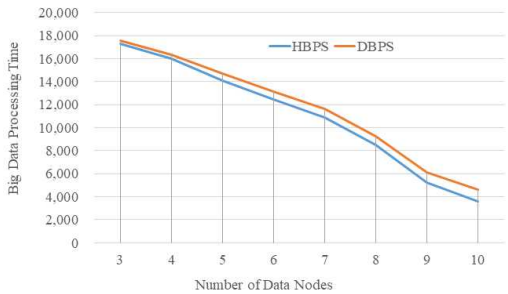

그림 7. 빅데이터 처리시간 비교 분석
Fig. 7. Comparison analysis of big data processing time

표 2. 데이터 노드의 수에 따른 빅데이터 처리 시간 비교 분석 결과
Table 2. Result of comparison analysis of big data processing time by number of data nodes

노드	3-4	5-6	7-8	9-10	합계
HBPS	33,290	26,544	19,404	8,832	88,070
DPBS	33,864	27,858	20,865	10,736	93,323
증감	-574	-1314	-1461	-1904	-5,253

결과에 따르면, *HBPS*는 *DBPS*에 비해 데이터 노드의 수가 증가될수록 더 빠르게 데이터를 처리하였다. 빅데이터 처리속도는 3개의 데이터 노드에서 *HBPS*가 *DBPS*보다 -260*ms*, 10개의 데이터 노드에서 -1,046*ms* 빠르게 데이터를 처리하였다. 특히, *HBPS*의 빅데이터 처리시간은 노드의 수가 증가될수록 약 22%씩 감소하였다.

이러한 결과가 발생한 이유는 *HBPS*에 적용된 Combiner 단계가 Map 단계 이후, 리듀스와 유사하게 리듀스 처리를 메모리 기반으로 수행하기 때문이다.

따라서 *HBPS*은 빅데이터 인프라 환경의 규모가 증가될수록 *DBPS*에 비해 적은 데이터 노드의 수로 운영이 가능해지며, 인프라 운영 및 구축비용은 감소한다.

4.3 노드 수에 따른 데이터 소실량 분석

노드의 수에 따른 데이터 소실량 분석은 요구사항 분석에서 도출된 장애가 발생할 확률을 기반으로 노드의 수에 따른 데이터 소실량을 분석한다. 데이터 노드의 수는 최소 3개부터 100대, 데이터 크기는 10GB, 기본 블록 크기는 64MB로 정의한다. 〈그림 8〉과 〈표 3〉은 노드의 수에 따른 데이터 소실량 분석을 나타낸다.

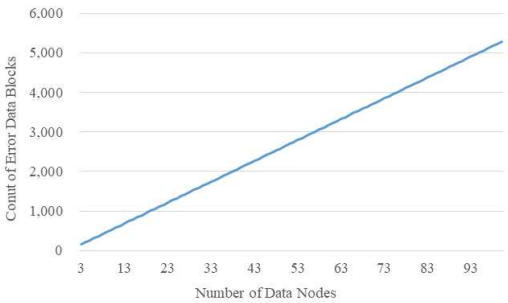

그림 8. 데이터 노드의 수에 따른 데이터 소실량 분석
Fig. 8. Analysis of error data blocks by number of data nodes

표 3. 데이터 노드의 수에 따른 데이터 소실량 비교 분석 결과
Table 3. Result of comparison analysis of error data blocks by number of data nodes

노드	3-20	21-40	41-60	60-80	81-100
소실	11,074	32,168	53,262	74,355	95,449
증감	-	21,094	32,168	42,188	53,262
비율	-	10%	47%	61%	74%

결과에 따르면, 데이터 노드의 수는 증가될수록 장애로 인한 데이터 소실량은 급격하게 증가하였다. 데이

터 노드의 수가 21개부터 40까지의 빅데이터 인프라 환경은 노드의 수가 3개부터 20까지의 빅데이터 인프라 환경에 비해 데이터 소실량이 약 10% 증가하였지만, 데이터 노드의 수가 81개부터 100까지의 빅데이터 인프라 환경은 데이터 소실량이 약 74%로 급격하게 증가되었다. 이때 데이터 소실량은 53,262개의 블록이다.

이러한 결과가 발생한 이유는 분산시스템의 특성상 데이터 노드의 수가 증가할수록 네트워크 및 각 데이터 노드들의 예기치 않는 오류가 발생할 확률이 높기 때문이다.

따라서 데이터 노도들의 수를 줄이며, 빅데이터 처리시간을 감소하기 위한 연구개발이 필요하다.

5. 결론

IT기술의 발달로 인해 매년 새롭게 생성되는 데이터의 양은 기하급수적으로 증가하고 있다. 이에 대한 대안으로 기존의 빅데이터 처리 기법은 노드의 수와 메모리의 양을 증가시켜 방대한 양의 데이터를 처리하고 있다.

이에 본 논문에서는 빅데이터 처리율 향상을 위한 인-메모리 기반 하이브리드 빅데이터 처리 기법을 제안하였다. 제안하는 기법은 분산시스템 처리기법에 Combiner 단계를 추가하고, 그 단계에서 인-메모리 기반 처리 기술을 적용하였다. 제안하는 기법의 성능평가의 결과에 따르면, 빅데이터 처리시간은 기존 기법에 비해 10개의 데이터 노드에서 1,046ms 빠르게 처리하였으며, 노드의 수에 따른 데이터 소실량은 기존 기법이 제안하는 기법에 비해 81개부터 100개의 데이터 노드에서 53,262개의 블록이 소실되었다.

REFERENCES

[1] H. Lee, Y. Kim, J. Park and J. Lee, "Map Reduce-Based Partitioner Big Data Analysis Scheme for Processing Rate of Log Analysis," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 5, pp. 593~600, 2018

[2] H. Lee, Y. Kim, K. Kim and J. Choi, "Design of GlusterFS Based Big Data Distributed Processing System in Smart Factory," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 1, pp. 70~75, 2018

[3] D. Hwang, K. Ko, S. Park and W. Kim, "Development for establishing Big Data-based alley commercial area," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 6, pp. 784~792, 2018

[4] H. G. Lee, Y. W. Kim and K. Y. Kim, "Implementation of an Efficient Big Data Collect ion Platform for Smart Manufacturing," Journal of Engineering and Applied Sciences, 12(2Si), pp. 6304-6307, 2018

[5] Y. Kwon and I. Kim, "A Study on Anomaly Signal Detection and Management Model using Big Data," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 16, No. 6, pp. 287~294, 2016

[6] J. Kim, J. Park and S. Chung, "Analysis of Network Log based on Hadoop," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 17, No. 5, pp. 125~130, 2017

[7] E. Jeong and B. Lee, "A Design of Hadoop Security Protocol using One Time Key based on Hash-chain," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 10, No. 4, pp. 340~349, 2017

저자약력

이 협 건(Hyeopgeon Lee) **[종신회원]**

•2011.03 - 2015.08,
숭실대학교 일반대학원
컴퓨터학과 공학박사
•2015.12 - 현재,
한국폴리텍대학 서울강서캠퍼스
데이터분석과 교수

〈관심분야〉 빅데이터, 실시간 분석, 데이터분석

김 영 운(Young-Woon Kim) **[종신회원]**

•2004.09 - 2008.08,
숭실대학교 일반대학원
컴퓨터학과 공학박사
•2015.12 - 현재,
한국폴리텍대학 서울강서캠퍼스
데이터분석과 교수

〈관심분야〉 빅데이터, 실시간 분산 처리, 데이터분석

김 기 영(Ki-Young Kim) **[종신회원]**

•2004.03 - 현재,
서일대학교 소프트웨어공학과
부교수

〈관심분야〉 빅데이터, 무선통신, 사물인터넷