KISM Smart Media Journal
Vol. 4, No. 3 pp. 16~23 ISSN:2287-1322

분산메모리시스템에서의 핫콜드 데이터 분류를 이용한 복합 백업 기법

(Compound Backup Technique using Hot-Cold Data Classification in the Distributed Memory System)

김우철*, 민동희*, 홍지만**

(Woo Chur Kim, Dong Hee Min, Ji Man Hong)

요 약

IT 기술의 발전으로 인해 데이터 처리 시스템은 많은 양의 데이터를 처리 및 가공해야 한다. 하지만, 기존에 사용되던 온디스크(On-Disk) 시스템으로는 급증하는 데이터를 빠르게 처리하는 데 한계점을 가졌다. 이로 인해 많은 분야에서 하드디스크에 데이터를 저장하는 것이 아닌 속도가 빠른 메모리에 데이터를 저장 및 관리하는 인메모리(In-Memory) 시스템이 도입되고 있다. 하지만, 메모리에 데이터를 관리하는 것은 메모리의 특성 중 하나인 휘발성으로 인해 데이터 손실이라는 위험을 갖기 때문에 항상 결함 허용 기법이 뒤따라야 한다. 결함 허용 기법은 인메모리 시스템의 처리 속도를 낮추는 성능 저하 원인이 된다. 따라서 본 논문에서는 인메모리 시스템의 데이터 사용 특성을 고려하여 핫콜드 데이터로 분류하고, 데이터 영속성 보장을 위한 복합 백업 기법을 제안한다. 제안하는 기법은 영속성을 높이고, 성능 저하 원인을 보완한다.

■ 중심어 : 분산메모리시스템 ; 백업 ; 핫콜드 ; 로깅 ; 체크포인팅

Abstract

As the IT technology advances, data processing system is required to handle and process large amounts of data. However, the existing On-Disk system has limit to process data which increase rapidly. For that reason, the In-Memory system is being used which saves and manages data on the fast memory not saving data into hard disk. Although it has fast processing capability, it is necessary to use the fault tolerance techniques in the In-Memory system because it has a risk of data loss due to volatility which is one of the memory characteristics. These fault tolerance techniques lead to performance degradation of In-Memory system. In this paper, we classify the data into Hot and Cold data in consideration of the data usage characteristics in the In-Memory system and propose compound backup technique to ensure data persistence. The proposed technique increases the persistence and improves performance degradation.

■ keywords : Distributed Memory System ; Backup ; Hot-Cold ; Logging ; Checkpointing

I. 서 론

과저 많은 데이터 처리 시스템은 모든 데이터를 주로 하드디스크(HDD, Hard Disk Drive)에서 저장 및 관리 되었다. 메모리(RAM, Random Access Memory)에 비해 상대적으로 느린 속도를 갖는 하드디스크에 데이터를 저장 및 관리하는 것은 전체 컴퓨팅 성능 저하의 큰 원인이 된다. IT 기술의 지속적인 발전은 하드디스크뿐만 아니라 메모리의 단위 용량 당 가격을 하락시켰다. 단위 용량 당 가격의 하락은 느린 처리속도를 갖게 하는 하드디스크 대신 메모리에 데이터를 저장하는 인메모리 시스템(In-Memory System)의 등장 배경의 주원인이 되었다 [1-2]. 메모리에 데이터를 저장 및 관리하는 것은 메모리의 빠른 속도로 인해 빅 데이터(Big Data) 시대에 발생되는 대규모 데이터를 고속의 실시간 처리를 가능하게 해준다. 하지만, 메모리는 전원 인가가 없을 경우 휘발성(Volatility)으로 저장된 데이터가 손실되는 특징을 갖는다. 이로 인해 인메모리 시스템의 예기치 않은 결함 발생 시 모든 데이터가 손실될 수 있다는 위험을

* 학생회원, 숭실대학교 컴퓨터학과
** 정회원, 숭실대학교 컴퓨터학부
이 논문은 2013년도 정부(교육부)의 제원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2013R1A1A2058154)

접수일자 : 2014년 12월 22일
수정일자 : 2015년 09월 16일
게재확정일 : 2015년 09월 21일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

갖게 된다. 따라서 인메모리 시스템은 예기치 않는 위험으로부터 데이터 손실을 방지하고, 지속적인 서비스 제공을 위해서는 결함 허용(Fault Tolerance) 기법을 필요로 한다[3]. 결함 허용 기법은 주로 데이터를 동일한 시스템에 중복 저장하는 로깅 및 체크포인팅 기법과 다른 시스템에 중복 저장하는 이중화 기법 등이 있다. 동일한 시스템에 중복 저장하는 로깅 및 체크포인팅 기법은 시스템의 많은 부하를 발생시키기 때문에 시스템의 데이터 처리 성능을 저하 시킨다. 반면, 다른 시스템에 중복 저장하는 이중화 기법은 추가적인 시스템이 필요하기 때문에 시스템 구축비용이 많이 든다는 단점이 있다. 본 논문에서는 인메모리 시스템의 데이터의 사용 특성에 따라 핫콜드 분류 기법을 이용하여 핫 데이터 및 콜드 데이터로 분류한다. 이후 각 데이터의 특징을 고려하여 로깅 및 체크포인팅 기법을 적용함으로써 데이터의 영속성 보장 및 시스템 부하 문제를 보완한다. 본 논문에서 제안하는 기법을 실제 분산 메모리 시스템에 적용하며, 실험을 통해 복합 백업 기법을 적용한 인메모리 시스템의 데이터 영속성 및 데이터 처리 성능을 비교한다.

본 논문은 다음과 같이 구성된다. 2장에서는 인메모리 시스템 및 백업 기법 그리고 기존 백업 기법의 문제점을 관련 연구로 다룬다. 3장에서는 인메모리 시스템에 핫콜드 분류 기법을 적용하고, 각 데이터의 사용 특성을 고려한 복합 백업 기법 적용에 대하여 논한다. 4장에서는 3장에서 제안하는 기법의 적합성을 판단하기 위해 실험을 통해 데이터의 영속성 분석 및 기존 기법과의 처리 성능을 비교하여 성능을 평가한다. 5장에서는 본 논문의 결론을 맺으며, 향후 연구 방향에 대해서 제시한다.

Ⅱ. 관련 연구

1. 인메모리 시스템(In-Memory System)

인메모리 시스템(IMC, In-Memory 시스템)의 개념은 일찍 등장하였으나 메모리의 다소 비싼 가격으로 인하여 수요가 많지 않았다. 최근 IT 기술 발전으로 인해 데이터양이 증가하였고, 이렇게 급증하는 데이터를 빠르게 처리하기에는 기존 온디스크 시스템(On-Disk System)으로는 한계가 있다는 문제점에 직면하게 되었다. 이러한 문제점과 함께 메모리의 단위 용량 당 가격의 하락은 과거 등장한 인메모리 시스템의 재등장의 주원인이 되었다. 인메모리 시스템은 기존에 어플리케이션을 구동하기 위해 사용된 메인메모리를 주요 데이터 저장 공간으로 사용하는 시스템이다. [그림 1]과 같이 데이터의 주 저장 공간의 변화는 과거보다 더욱 더 빠른 데이터 처리가 가능해지게 하였다. 온디스크 시스템은 저장 공간은 크지만 많은 양의 데이터를 고속으로 처리하기에는 적합하지 않다. 반면 인메모리 시스템은 저장 공간은 온디스크 시스템에 비해 제한되지만 데이터를 고속으로 처리하기에 적합하다. 인메모리 시스템은 디스크 작업으로 인한 지연시간 없이 데이터를 처리하기 때문에 온디스크 시스템보다 빠른 데이터 처리가 가능하다. 특히 많은 데이터가 발생하는 증권사의 실시간 금융 트레이딩, 온라인 게임, 모바일 어플리케이션, 통신사의 세션관리, 소셜 미디어 등 다양한 분야에서 인메모리 시스템은 효과적이다.

제한된 저장 공간을 갖는 인메모리 시스템의 많은 양의 데이터를 효율적으로 저장하기 위해서는 기존 복잡한 형태의 관계형 데이터베이스 시스템 보다는 간단한 형태의 비정형 데이터베이스 시스템이 적합하다. 비정형 데이터 모델인 NoSQL은 “Not Only SQL” 혹은 “Not Relational”을 의미하며, 기존 관계형 데이터베이스 시스템보다 좋은 수평 확장성을 갖기 때문에 대규모 서버 구축에 적합하다. NoSQL 모델은 데이터 모델에 따라 Key-Value Store, Document Store, Extensible Record Store, Relational Database와 같이 네 가지 형태로 분류될 수 있다 [4]. 본 논문에서는 모든 데이터를 메모리에 관리하며, 대규모 데이터 처리에 적합한 Key-Value Store를 다룬다. Key-Value Store 중 가장 많이 사용되는 Redis Open Source에 대해 다룬다.

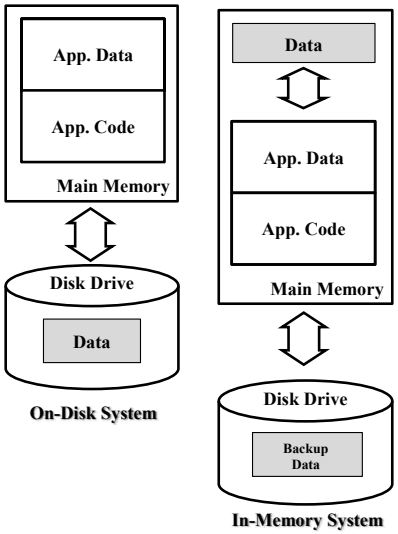

그림 1. 데이터 처리 시스템 비교

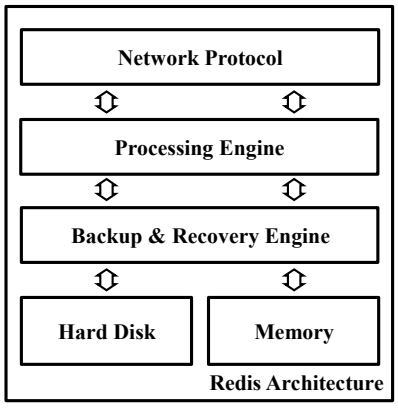

그림 2. Redis Architecture

2. 백업 기법

백업 기법은 동일한 시스템에 복제하는 방법과 다른 시스템에 복제하는 방법이 있다. 동일한 시스템에 복제하는 방법은 로깅 및 체크포인팅 기법이 있다. 시스템 구축비용은 낮지만 시스템에 부하를 일으키는 단점이 있다. 다른 시스템에 복제하는 방법은 시스템 부하는 적지만 시스템 구축비용이 크다는 단점이 있다. 본 논문에서는 시스템 부하를 줄이기 위해 동일한 시스템에 복제하는 방법인 로깅 기법 및 체크포인팅 기법을 다룬다.

가. 로깅 기법

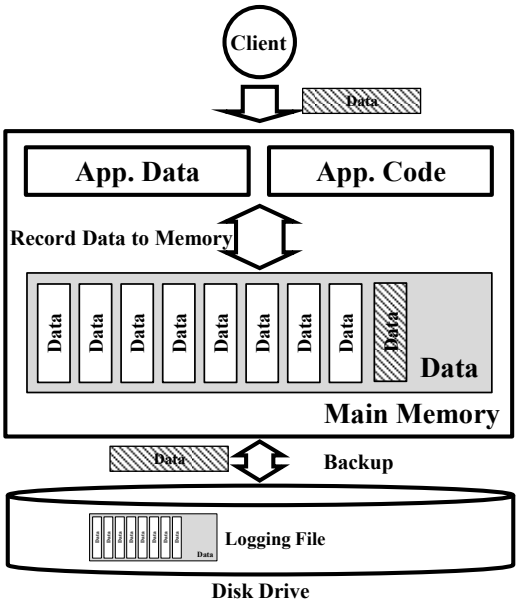

그림 3. 로깅 기법

로깅(Logging) 기법은 인메모리 시스템에 저장되어 있거나 혹은 저장될 데이터 변화를 하드디스크에 기록하는 백업 기법이다. 따라서 시스템의 예기치 않는 결함 발생 시 하드디스크에 기록된 로깅 데이터를 통해 인메모리 시스템의 데이터 복구가 가능하다. 로깅 기법은 [그림 3]과 같이 데이터의 변화를 하드디스크에 기록하기 때문에 데이터 손실을 최소화 할 수 있다는 장점을 갖는다. 하지만, 데이터 변화 발생 시 하드디스크 입출력 작업으로 인해 발생되는 시스템 부하로 인해 데이터 처리 속도가 떨어진다는 단점을 갖는다. 또한, 데이터의 모든 변화를 기록하기 때문에 하드디스크에 불필요한 쓰기 작업이 발생한다. 이는 시스템의 불필요한 부하 발생 및 하드디스크의 공간 낭비라는 문제점을 갖게 한다. 로깅 백업 파일의 불필요한 공간 낭비를 방지하기 위해 많은 시스템에서는 로깅 백업 파일의 재구성하는 작업을 별도로 진행한다. 로깅 데이터 파일 재구성 시 시스템 부하가 증가하게 된다.

나. 체크포인팅 기법

체크포인팅 기법은 특정 주기마다 인메모리 시스템에 저장된 모든 데이터를 하드디스크에 기록하는 백업 기법이다 [5]. 로깅 기법과 달리 데이터 처리 시 디스크 입출력이 발생하는 것이 아닌 [그림 4]와 같이 특정 주기마다 데이터를 기록하기 때문에 백업 수행 시 많은 디스크 입출력 작업이 발생한다. 따라서 시스템 결함 발생 시 특정 주기에 기록된 백업 파일을 통해 인메모리 시스템의 데이터 복구가 가능하다. 체크포인팅 작업으로 인한 시스템 부하의 크기는 체크포인팅 주기에 의해 결정된다. 체크포인팅 주기가 짧을 경우 하드디스크에 데이터를 기록하는 횟수가 많아진다. 그 결과 시스템의 부하는 커지며, 이는 저장된 데이터양이 많을수록 커진다. 반면, 체크포인팅 주기가 길 경우 하드디스크에 데이터를 기록하는 횟수는 줄어든다. 그 결과 시스템의 부하는 줄어들지만 체크포인팅 주기 사이에 발생된 데이터 변화에 대한 기록을 할 수 없기 때문에 데이터 손실 확률이 증가한다는 단점이 있다. 결과적으로 체크포인팅 기법은 로깅 기법에 비해 데이터 공간 낭비가 적으며 시스템 부하를 적게 발생시키는 장점이 있지만 데이터 손실 위험이 높다는 치명적인 단점을 갖는다.

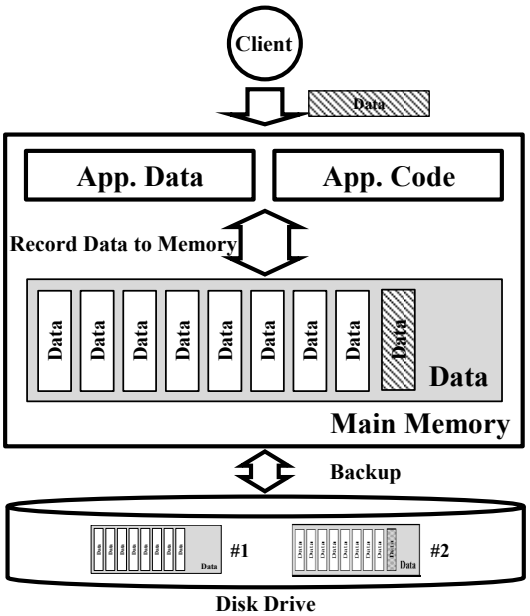

그림 4. 체크포인팅 기법

3. 기존 백업 기법의 문제점

인메모리 시스템은 예기치 않은 시스템 결함으로 인한 데이터 손실 방지를 위해 다양한 결함 허용 기법들을 사용한다. 동일한 시스템에 데이터를 기록하는 체크포인팅 기법 및 로깅 기법, 다른 시스템에 기록하는 이중화 기법 등이 있다. 동일한 시스템에 데이터를 기록할 경우 시스템 부하로 인하여 데이터 처리 속도에 영향을 미치는 큰 단점이 있다.

가. 체크포인팅 주기에 따른 데이터 손실

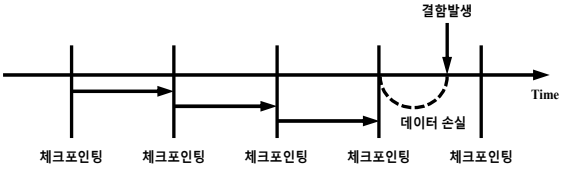

그림 5. 체크포인팅 주기로 인한 문제점

체크포인팅 주기는 인메모리 시스템의 처리 성능 및 데이터 손실량에 큰 영향을 미친다. 체크포인팅 주기에 따라 인메모리 시스템에 저장된 모든 데이터를 하드디스크에 기록한다. [그림 5]와 같이 체크포인팅이 이루어진 이후 다음 체크포인팅 주기가 올 때까지 발생되는 데이터는 결함 발생 시 손실된다. 이 주기를 짧게 할 경우 데이터의 손실되는 양을 줄일 수 있다. 하지만, 잦은 하드디스크 입출력 작업으로 인해 시스템 부하가 많이 발생하게 된다.

나. 불필요한 데이터의 하드디스크 쓰기 작업

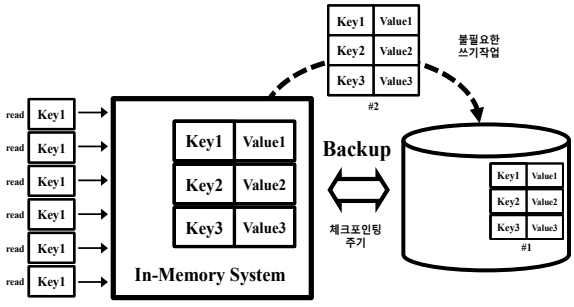

그림 6. 읽기 특성에 부적합한 체크포인팅 기법

체크포인팅 기법은 특정 주기마다 발생되기 때문에 읽기 작업이 빈번히 발생되는 데이터가 저장된 시스템에 부적합하다. 실제 변경된 데이터가 존재하지 않더라도 체크포인팅 주기에 따라 하드디스크 쓰기 작업이 발생하며, 시스템의 부하로 인해 읽기 지연 시간이 발생된다. 즉, 이미 기록된 파일과 동일한 데이터를 기록하기 때문에 불필요한 하드디스크 쓰기 작업이 발생된다.

다. 데이터 특성에 따른 로깅 데이터의 크기 증가

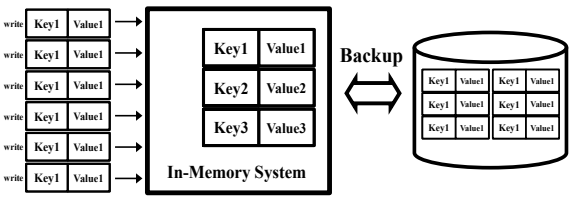

그림 7. 로깅 파일의 불필요한 공간 낭비

로깅 기법은 데이터 변화를 기록하기 때문에 자주 변화되는 데이터의 변화를 기록하기에 적합하다. 하지만, 동일한 데이터의 변화를 기록하게 될 경우 백업 데이터 파일의 불필요한 공간 낭비라는 문제점을 갖게 한다. 불필요한 공간 낭비 발생을 방지하기 위해 많은 로깅 기법은 로깅 백업 파일을 재구성 하는 기법을 사용한다. 로깅 백업 파일의 재구성 작업은 체크포인팅 기법과 동일하게 시스템의 부하를 발생하게 된다. 즉, 데이터의 쓰기 작업이 증가할수록 로깅 백업 기법을 갖는 시스템의 데이터 처리 성능이 저하되게 된다.

라. 기존 백업 기법의 성능 변화

데이터 특성을 고려하지 않는 체크포인팅 및 로깅 기법의 사용의 문제점을 분석하기 위해 읽기 및 쓰기 비율에 따른 인메모리 시스템의 데이터 처리 성능 변화를 실험하였다. 실험은 체크포인팅 기법, 로깅 기법 및 모두 사용한 경우와 모두 사용하지 않는 경우로 구분하였다. 데이터의 처리 성능 비교를 위해 총 2,000,000 개의 트랜잭션에 대하여 읽기 및 쓰기 작업을 2:8, 5:5, 8:2 로 구분하였으며, 처리 성능 변화를 분석하였다. 체크포인팅 기법은 로깅 기법을 사용하는 경우보다 성능이 좋았지만 데이터 손실의 위험이 있었다. 하지만, 로깅 기법은 데이터 손실의 위험은 적지만 성능이 좋지 않았다. 따라서 본 논문에서는 각 기법을 별도로 적용하는 것이 아닌 각 데이터 사용 특성을 고려하여 데이터 백업 기법을 적용한다. 체크포인팅 기법의 특징인 적은 시스템 부하를 갖으며 로깅 기법의 특징인 적은 데이터 손실 위험도를 갖게 한다. [그림 8]은 제안하는 복합 백업 기법의 목표 성능 구간을 표시한다 [6]. 제안하는 기법은 낮은 성능 저하를 가질 뿐만 아니라 데이터의 손실 위험을 줄일 수 있다.

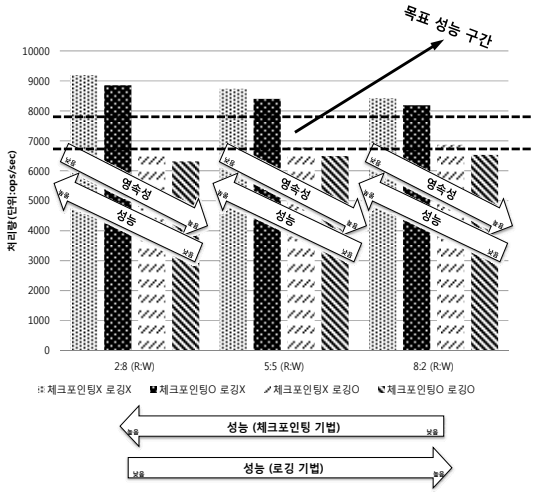

그림 8. 백업 기법 선택에 따른 성능 변화

Ⅲ. 복합 백업 기법

이 절에서는 기존 백업 기법의 문제점인 성능 저하 문제 및 데이터 손실 문제점을 보완한 복합 백업 기법에 대하여 설명한다. 데이터의 사용 특성을 고려한 핫콜드 데이터 분류 기법에 대해 설명하며, 이를 활용한 복합 백업 기법에 대해 설명한다. [그림 9]는 핫콜드 데이터 분류 기법을 적용한 인메모리 시스템의 복합 백업 기법을 나타낸다. 핫 데이터는 콜드 데이터는 별도 비트맵을 통해 관리하며, 각 데이터에 별도의 백업 기법을 적용함으로써 데이터 손실 및 시스템 부하를 줄인다.

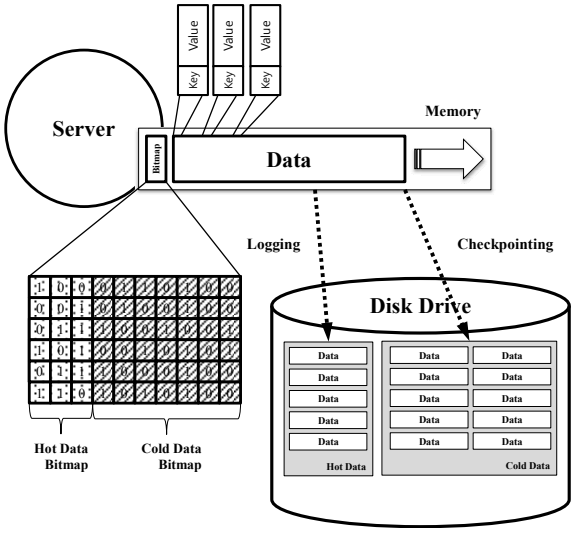

그림 9. 핫콜드 데이터 분류를 사용한 복합 백업 기법

1. 핫콜드 데이터 분류

핫콜드 데이터 분류 기법은 데이터의 사용 특성에 따라 핫 데이터 및 콜드 데이터로 분류한다 [7]. 데이터의 사용 특성은 읽기 작업이 주로 이루어지는지 혹은 쓰기 작업이 주로 이루어지는지로 구분한다. 즉, 핫 데이터는 주로 쓰기 작업이 이루어지는 데이터로 추정될 경우로 정의한다. 콜드 데이터는 주로 읽기 작업이 이루어지는 데이터로 추정될 경우로 정의한다. 읽기 작업 및 쓰기 작업이 번갈아 이루어질 경우 명확히 정의할 수 없기 때문에 웜 데이터로 정의한다. 웜 데이터는 분류 기준에 따라 핫 데이터 및 콜드 데이터로 분류될 수 있다. 웜 데이터 관리는 서버 용량, 데이터 접근 패턴 등 시스템 특성에 따라 비율을 조절함으로써 적응적인 설정이 가능하다. 논문에서 제안하는 핫콜드 데이터 분류 기법은 특정 주기에 따라 핫콜드 데이터 분류 알고리즘을 사용한다. 본 논문에서는 데이터의 사용 빈도는 고려하지 않으며, 사용 특성만을 고려한다.

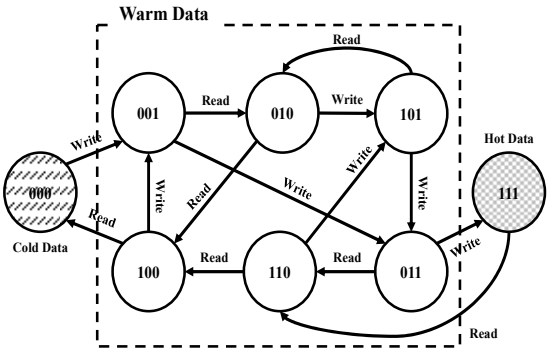

그림 10. 3비트 핫콜드 데이터 분류

각 데이터 별 기록된 사용 특성 정보를 통해 핫콜드 데이터 분류 알고리즘은 핫 데이터 및 콜드 데이터를 분류할 수 있다. 각 비트에 시간의 흐름에 따라 가중치를 부여함으로써 핫 데이터 및 콜드 데이터를 분류할 수 있다. [그림 10]은 읽기 및 쓰기 작업에 따른 비트변화를 나타낸다. 기록된 비트 변화 정보를 가지고 핫 데이터 및 콜드 데이터를 분류 할 수 있다.

2. 복합 백업 기법

기존 백업 기법은 백업 기법 선택에 따라 데이터 손실과 성능 사이의 Trade-off 관계에 있다. 제안하는 복합 백업 기법은 데이터의 손실을 줄이고, 인메모리 시스템의 성능 향상을 주목적으로 한다. 핫콜드 데이터 분류

기법을 통해 분류된 핫 데이터와 콜드 데이터는 각 특성을 가지고 있다. 핫 데이터는 자주 쓰이기 때문에 현재 저장된 값이 시간이 지날수록 변화할 확률이 크다. 데이터 변화가 발생할 경우 백업 파일은 유효하지 않게 된다. 따라서 로깅 기법을 적용함으로써 데이터의 변화에 따른 백업 파일 갱신은 데이터 손실을 최소화할 수 있다. 콜드 데이터는 자주 읽히기 때문에 현재 저장된 값이 시간이 지나더라도 변화하지 않을 확률이 크다. 데이터 변화가 빈번히 발생하기 때문에 백업 파일의 유효 시간이 증가하게 된다. 따라서 체크포인팅 기법을 적용함으로써 불필요한 백업 파일 갱신 작업을 최소화할 수 있다.

핫콜드 데이터 분류 기법에 의해 분류된 핫 데이터 및 콜드 데이터는 각 특성을 고려한 백업 기법을 적용한다. 복합 백업 알고리즘은 [표 1]과 [표 2]와 같다.

표 1. 복합 백업 알고리즘(체크포인팅)

```
1: Function backupCheckPointing() {
2:   if (is Checkpoint Cycle) {
3:     while ((key = NextKey()) != NULL) {
4:       if (key.bitmap == COLD_DATA)
5:         write(checkpoint file, key, value);
6:     }
7:   }
8: }
```

표 2. 복합 백업 알고리즘(로깅 재구성)

```
1: Function backupLoggingRewrite() {
2:   while ((key = NextKey()) != NULL) {
3:     if (key.bitmap == HOT_DATA)
4:       write(logging file, key, value);
5:     }
6:   }
7: }
```

Ⅳ. 실험 및 평가

1. 실험 환경

구분		설명
H/W	CPU	Intel Xeon E5-2665 (16 core, 2.4Ghz)
	Memory	32 GB (4*8GB)
	Network	10G Ethernet
S/W	Operating System	Redhat Enterprise Linux 6.5
	Kernel	2.6.32-358.el6.x86_64
	In-memory System	Redis Open Source 2.8.17
	Benchmark Tool	YCSB (Yahoo! Cloud Serving Benchmark)

그림 11. 실험 환경

제안하는 복합 백업 기법 성능을 측정하기 위해 [그림 11]과 같이 실험 환경을 구축하고, 테스트를 수행한다. 인메모리 시스템의 데이터 처리량 측정을 위해 Yahoo에서 제공되는 벤치마크 툴인 YCSB를 사용한다 [8]. YCSB는 읽기 및 쓰기 작업의 빈도를 변화할 수 있으며, 데이터의 평균 처리량 및 지연되는 구간 등을 분석할 수 있다. 데이터는 총 2,000,000 개의 트랜잭션을 사용한다. 읽기 및 쓰기 작업은 기존 백업 기법의 문제점을 분석했을 때와 동일하게 2:8, 5:5, 8:2로 구분한다. 제안하는 복합 백업 기법과 기존 백업 기법의 데이터 처리량을 비교하고, 복합 백업 기법을 통해 생성된 백업 파일의 크기 및 데이터 복구 여부를 평가한다. [그림 12]와 같이 인메모리 시스템을 구성하였으며, 인메모리 오픈소스는 Redis를 사용하였다.

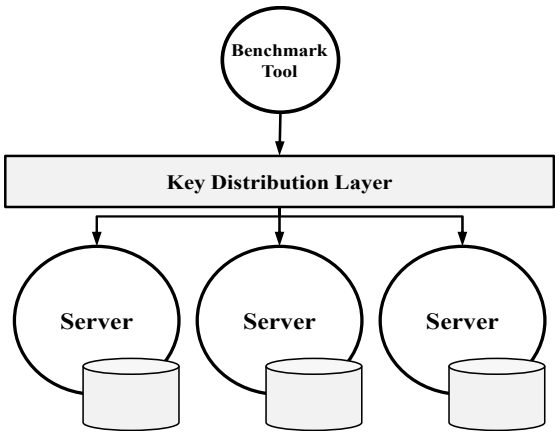

그림 12. 서버 구성

2. 실험 결과

본 논문에서 제안하는 복합 백업 기법과 기존 백업 기법의 성능 비교 결과이다. 성능 비교는 읽기 및 쓰기 빈도를 구분하였고, 데이터 처리량을 측정하였다. [그림 13]은 각 백업 기법에 따른 데이터 처리 성능 비교를 나타낸다.

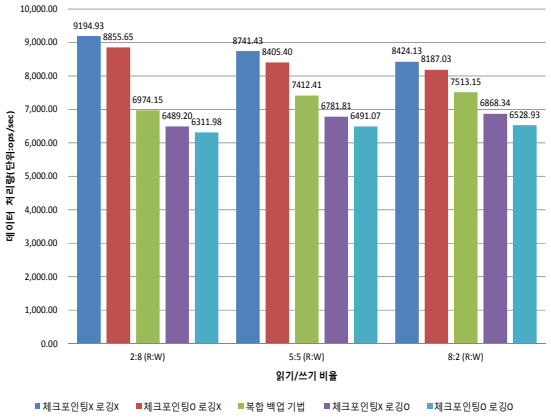

그림 13. 백업 기법 성능 비교

복합 백업 기법은 기존 백업 기법인 로깅 기법에 비해 높은 처리량을 보였다. 반면, 기존 백업 기법인 체크포인팅 기법에 비해 낮은 처리량을 보였다. [그림 14]는 기존 백업 기법 및 복합 백업 기법으로 생성된 백업 파일 크기 비교를 나타낸다. 기존 백업 기법은 로깅 기법 및 체크포인팅 기법이 모든 데이터에 대해 독립적으로 동작하기 때문에 동일한 데이터를 중복 저장하고 있다. 반면, 복합 백업 기법은 동일한 데이터에 별도의 백업 기법을 적용하기 때문에 약 50% 정도 크기를 갖는 백업 파일이 생성된다.

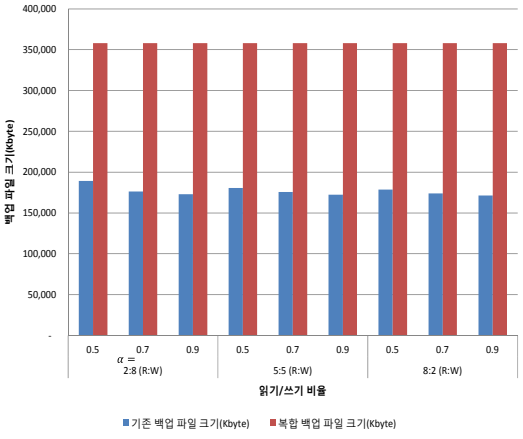

그림 14. 백업 파일 크기 비교

복합 백업 기법은 체크포인팅 및 로깅 기법을 동시 적용하기 때문에 결함 발생 이후 재가동 되더라도 모든 데이터 복구가 가능하다. [표 3]은 백업 파일을 통한 데이터 복구 결과이다. 읽기 및 쓰기 비율에 따라 서로 다른 크기를 갖는 백업 파일이 생성되었지만, 모든 데이터가 복구되었다.

표 3. 데이터 복구 결과

구분	설명			
R/W 비율	체크포인팅 파일 데이터 수	로깅 파일 데이터 수	복구된 데이터 수	복구율 (%)
2:8	113,804	36,197	150,001	100 %
5:5	80,808	69,193	150,001	100 %
8:2	65,608	84,393	150,001	100 %

3. 실험 결과 분석

[표 4]는 실험 결과에 대한 요약을 나타낸다. 동일한 데이터로 기존 백업 기법 및 복합 백업 기법을 테스트 수행할 경우 전체적인 디스크 입출력 시간이 줄어든다. 콜드 데이터는 주로 읽기 작업이 이루어지기 때문에 체크포인팅 주기가 길어지며 체크포인팅 파일의 유효시간이 길어지게 된다. 또한, 체크포인팅 파일 기록 시 콜드 데이터만 기록하기 때문에 체크포인팅 파일 기록 시간이 줄어들게 되며, 로깅 파일의 재구성 시 핫 데이터만 재구성하기 때문에 발생되는 디스크 입출력 시간이 줄어든다.

표 4. 실험 결과 분석 요약

비교 항목	기존 백업 기법	복합 백업 기법
로깅 파일 재구성 횟수	동일함	동일함
체크포인팅 주기	짧다	길다
체크포인팅 파일 기록 시간	길다	짧다
로깅 파일 기록 시간	길다	짧다
체크포인팅 파일 유효 시간	짧다	길다

V. 결 론

본 논문에서는 인메모리 시스템의 데이터 손실을 방지하기 위해 복합 백업 기법을 제안하였다. 복합 백업 기법은 데이터 사용 특성을 고려한 핫콜드 데이터 분류 기법을 사용하여 각 데이터에 서로 다른 백업 기법을 적용하였다. 기존 백업 기법은 데이터 사용 특성을 고려하지 않았기 때문에 동일한 데이터를 중복 백업하거나 혹은 데이터 손실과 처리량의 Trade off 관계를 가졌다. 반면, 제안하는 복합 백업 기법은 데이터의 읽기 및 쓰기 사용 특성을 고려하여 로깅 및 체크포인팅 기법을 별도 적용하였기 때문에 데이터의 손실 없이 성능 저하 문제점을 보완할 수 있었다. 본 논문에서 제안하는 복합 백업 기법의 성능 테스트를 위해 기존 백업 기법과 데이터 처리량을 비교하였다. 또한, 데이터 백업 파일의 크기를 분석하였고 데이터 복구 여부를 실험하였다. 백업 파일의 크기는 기존 기법에 비해 줄었으며, 데이터의 손실도 없앨 수 있었다. 하지만, 본 논문에서는 데이터 사용 특성만을 고려하였으며, 시간 및 사용 빈도를 고려하지 않았다. 향후 연구에서 시간 및 사용 빈도를 고려한 핫 데이터 및 콜드 데이터 분류 알고리즘을 연구할 계획이다.

REFERENCES

[1] 구영수, “In-Memory Computing 기술현황 및 전망”, *Technology Inside G CNS R&D Journal*, 2013. 1

[2] M.K. Gupta, V. Verma, and M.S. Verma, “In-Memory Database Systems - A Paradigm Shift”, *International Journal of Engineering Trends and Technology (IJETT)*, vol. 6, no. 6, pp. 333-336, Dec. 2013.

[3] F. Cristian, “Understanding Fault-Tolerant Distributed Systems”, *Communications of the ACM*, vol. 34, no. 2, pp. 56-78, Feb. 1991.

[4] R. Cattell, “Scalable SQL and NoSQL Data Stores”, *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12-27, Dec. 2010.

[5] R. Koo and S. Toueg, “Checkpointing and Rollback-Recovery for Distributed Systems”, *Software Engineering, IEEE Transactions on*, vol. 13, no. 1, pp. 23-31, Jan. 1987.

[6] J. Zhao, S. Li, D.H. Yoon, Y. Xie, and N.P. Jouppi, “Kiln: Closing the Performance gap between Systems with and without persistence support”, *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 421-432, Canada, Dec. 2013.

[7] J.J. Levandoski, P. Larson, and R. Stoica, “Identifying Hot and Cold Data in Main-Memory Databases”, *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, Brisbane QLD, pp. 26-37, Apr. 2013.

[8] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with YCSB”, *Proceedings of the 1st ACM Symposium on Cloud Computing*, pp. 143-154, USA, Jun, 2010.

저 자 소 개

김우철(학생회원)

2013년 숭실대학교 컴퓨터학부 학사 졸업.
2014년 숭실대학교 컴퓨터 학과 석사 과정.
<주관심분야 : 운영체제, 시스템 소프트웨어, 분산메모리시스템>

민동희(학생회원)

2014년 숭실대학교 컴퓨터학부 학사 졸업.
2014년 숭실대학교 컴퓨터 학과 석사 과정.
<주관심분야 : 운영체제, 시스템 소프트웨어, 분산메모리시스템>

홍지만(정회원)

1994년 고려대학교 컴퓨터 학과 학사 졸업.
1997년 서울대학교 컴퓨터공학과 석사 졸업.
2003년 서울대학교 컴퓨터공학과 박사 졸업.
<주관심분야 : 운영체제, 시스템 소프트웨어, 임베디드 시스템>