클라우드 환경에서 확장성을 지원하는 트랜잭션 처리 방법

김치연*

A Study for Transaction Processing Supporting Scalability in the Cloud

Chi-Yeon Kim*

요 약

최근에 클라우드 컴퓨팅 패러다임은 다양한 응용에서 받아들여지고 있다. 클라우드 환경의 데이터 관리 시스템은 대용량의 데이터와 확장성을 지원하는 능력이 요구된다. 대용량의 데이터를 다루기 위해 일관성과 트랜잭션의 제약을 완화하는 것이 필요하고, 확장성을 지원하기 위해 구성 요소의 가감을 허용해야 한다. 이 논문에서는 클라우드 환경에서 트랜잭션을 처리할 때 필요한 시스템 모델과 확장가능한 모듈 관리 알고리즘에 대하여 제안한다. 시스템 모델은 트랜잭션 관리 모듈과 데이터 관리 모듈로 구성된다. 모듈 관리 알고리즘은 불필요한 재분배가 발생하지 않고, 기존 모듈의 부하를 덜어줄 수 있다. 성능 분석 결과, 모듈의 확장으로 응답 시간을 향상시키고 트랜잭션 철회율을 감소시킬 수 있음을 알 수 있었다.

ABSTRACT

Recently the cloud computing paradigm has been widely accepting in various applications. Data management system of cloud computing requires ability to manage tremendous data and supporting scalability. The former can be achieved by weaken consistency and limitation of transactions, and the latter needs expand or shrink of components. In this paper, we propose a transaction processing model and a scalable module management algorithm when transaction is executed in the cloud. Transaction processing model consists of a transaction management module and a data management module. Scalable module management algorithm has no redistribution of components and may alleviates loads of existed modules. With simulation results, we can see the improvement of response time and decrease abort ratio of transactions.

키워드

ACID property, Scalability, Cloud, Transaction Management
ACID 특성, 확장성, 클라우드, 트랜잭션 관리

Ⅰ. 서 론

클라우드 컴퓨팅은 클라이언트가 소유하지 않은 컴퓨팅 자원을 인터넷을 통해 서비스 형태로 이용하는 방식이라고 정의할 수 있다[1]. 최근의 서비스 지향적인 컴퓨팅 환경에 잘 어울리는 클라우드 환경은 소프트웨어나 하드웨어 자원을 필요한 만큼 빌려쓰고 비용을 지불할 수 있기 때문에 규모의 유동성이 필요한

* 목포해양대학교 해양컴퓨터공학과(gegujang2@mmu.ac.kr)
접수일자 : 2012. 07. 06 심사(수정)일자 : 2012. 07. 26 게재확정일자 : 2012. 08. 09

응용에 효율적이다. 클라우드에서 제공하는 서비스는 전형적으로 IaaS, PaaS, SaaS로 분류되며, 최근에는 서비스가 더욱 다양해지고 있는데 그 중 하나가 DaaS로 언급되는 Database as a Service이다[2]. 최근 스마트 기기의 보급과 더불어 확산되고 있는 Facebook과 같은 SNS도 클라우드 패러다임을 사용하고 있다.

클라우드 컴퓨팅 안에는 여러 노드를 가진 분산 컴퓨팅 방식과 가상화의 개념을 사용한 그리드 컴퓨팅, 과금 방식에서 동일한 유틸리티 컴퓨팅, 데이터 및 응용의 운영 방식에서는 서버 기반의 컴퓨팅, 그리고 네트워크 컴퓨팅 기술 등이 복합되어 있어[1], 다양한 분야에서 연구가 이루어지고 있다.

분산 데이터 관리의 관점에서 클라우드 환경의 특징은 클라우드에서 관리되는 데이터의 용량과 확장가능한 구조를 꼽을 수 있다. Facebook의 경우, 하루에 생성되는 원시 데이터의 양이 60~90TB 범위이다[3]. 이러한 대량의 데이터를 저장, 관리하기 위해서는 스토리지 뿐 아니라 질의 처리를 위한 새로운 서버 기술이 필요하다[4]. 서버 능력을 향상시키지 않은 상태에서 스토리지 용량만 늘어나면 처리 속도가 저하되기 때문이다. 이전의 환경과 비교될 수 없을 만큼의 대용량의 데이터를 관리하면서 발생하는 문제 중 하나는 기존의 제약 조건을 완화해야 한다는 점이다. 그 중 하나가 CAP 이론에서 지적한 데이터 일관성의 완화이다[5]. 클라우드 환경에서 중요시되는 데이터 가용성을 위해 일관성은 완화되었고, 그 결과로 궁극적 일관성과 같은 약한 일관성이 제안되었다[6]. 데이터 일관성의 완화에 추가하여 트랜잭션 자체의 제약도 완화되었다. SimpleDB나 구글의 BigTable과 같은 응용에서 수행되는 트랜잭션은 일관성 기준을 완화하거나 연산을 제한하는 등, 제약을 완화한 트랜잭션이다[7]. 하지만, 여전히 웹 2.0 응용, 온라인 경매, 협업 편집 등의 응용에서는 강한 제약을 필요로 한다[8].

대용량의 데이터 관리와 더불어 확장성은 클라우드 환경의 핵심 키워드이다. 확장성은 규모의 유동성을 제공하며, 클라우드 환경의 고유한 특징이다. 클라우드 환경에서 트랜잭션 처리를 다룬 연구들에서는 트랜잭션 처리 방법과 더불어 확장성의 제공에 대해 언급하고 있지만 구체적인 확장 알고리즘에 대한 명세가 없다. 클라우드 환경에서 확장성을 제공하기 위해서는 트랜잭션 관리 모듈과 데이터 관리 모듈의 확장이 가능해야 한다. [7][8][9]에서 제안한 트랜잭션 처리 방법들 또한 확장성 부분은 상대적으로 명확하지 않거나 구체적인 알고리즘이 없다. 따라서 이 논문에서는 클라우드 환경에서 트랜잭션을 원활하게 수행하기 위한 확장성 제공 방법에 대하여 논의하고자 한다.

논문의 구성은 다음과 같다. 2장에서는 클라우드 환경에서 트랜잭션 처리를 다룬 연구들의 트랜잭션 처리 구조와 확장성 측면을 살펴보고, 3장에서는 확장성을 제공하기 위한 알고리즘을 제안한다. 4장에서는 제안하는 방법에 대한 성능 분석을 다루고, 5장에서는 결론을 기술한다.

II. 관련 연구

이 장에서는 클라우드 환경에서 트랜잭션 관리와 확장성을 다룬 세 개의 연구[7][8][9]에 대하여, 트랜잭션 처리 구조와 확장성 지원 방법, 적용가능한 응용 분야 및 허용하는 연산의 측면에서 분석한 내용을 기술한다.

2.1 트랜잭션 처리 구조

전통적인 데이터베이스 시스템에서 트랜잭션 처리 구조는 트랜잭션 관리자, 스케줄러, 데이터 관리자로 구성된다[10]. 트랜잭션 관리자는 트랜잭션을 분배하고 분산 트랜잭션에 대한 조정자의 역할을 수행하며, 스케줄러는 잠금이나 타임스탬프 기법을 이용하여 트랜잭션을 스케줄링한다. 데이터 관리자는 데이터의 캐쉬를 관리하며 고장이 발생하면 회복을 담당한다. 클라우드 환경에서 제안된 트랜잭션 처리 구조 또한 기본 구조는 크게 다르지 않으나 클라우드 환경에 맞는 독특한 모듈을 사용하거나 기존 모듈을 통폐합한 처리 구조를 사용하고 있다. 그림 1~3은 세 연구에서 제안한 트랜잭션 처리 구조이다.

트랜잭션 처리 구조의 측면에서 [7]의 특징은 HTM(Higher Level Transaction Manager)과 OTM(Owning Transaction Manager)으로 명명된 두 레벨의 트랜잭션 관리자를 두었다. HTM의 기능은 모든 판독 전용 트랜잭션을 캐쉬된 데이터를 이용하여 수행하고, 미니 트랜잭션[11]에 대해서는 조정자의 역할을 수행한다. 이를 위해 HTM은 실제 배타적으로 데

이터를 소유하고 있는 OTM에 대한 매핑을 유지해야 하며, HTM은 확장 가능하다. 분산 잠금 관리를 위해 Chubby[12]와 유사한 시스템을 사용하였고, 중복 데이터의 일관성을 위해 Paxos[13] 합의 알고리즘을 사용하였다.

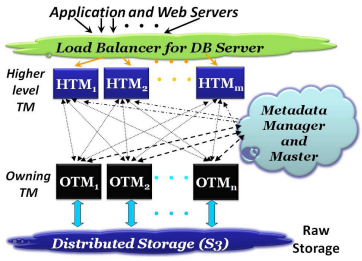

그림 1. [7]의 시스템 모델
Fig. 1 System model of [7]

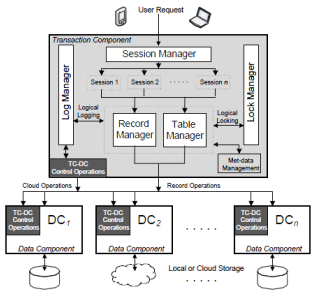

그림 2. [8]의 시스템 모델
Fig. 2 System model of [8]

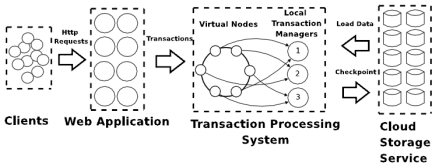

그림 3. [9]의 시스템 모델
Fig. 3 System model of [9]

[8]에서는 트랜잭션 관리자 역할을 하는 TC(Transaction Component) 모듈에 세션 관리자, 로그 관리자, 잠금 관리자, 레코드 관리자와 테이블 관리자를 두고, DC(Data Component)모듈과 교환하기 위한 제어 정보를 저장한다. 또한 연산을 접근할 데이터가 있는 DC로 보내기 위한 메타 데이터도 저장하고 있다. DC 모듈은 물리적인 데이터를 저장하며, 레코드 접근을 위한 인터페이스를 제공한다. 동시성 제어를 위해 잠금을 사용한다. [9]에서는 TPS(Transaction Processing System) 안에 LTM(Local Transaction Manager)이 존재하는데, 트랜잭션은 임의의 LTM에 제출되며 트랜잭션이 접근하는 모든 LTM에 대한 조정자의 역할을 수행한다. 수정된 데이터는 LTM에 저장되었다가 정기적인 검사점을 통하여 클라우드 스토리지에 반영된다. 타임스탬프 기법으로 스케줄되며 분산 수행된 트랜잭션의 완료를 위해 2PC 완료 프로토콜을 사용하였다. LTM에 대한 데이터 할당은 해싱을 이용하며, LTM 장애에 대비하여 가상 노드와 트랜잭션 상태를 하나 이상의 LTM에 중복 저장하였다.

2.2 확장성의 지원

클라우드 환경이 전통적인 환경과 구별되는 가장 큰 특징 중 하나가 확장성이다. 트랜잭션 처리 구조와 관련하여 확장성은 트랜잭션 처리를 담당하는 트랜잭션 관리자 모듈의 확장과 데이터를 저장하는 데이터 모듈의 확장으로 분류할 수 있다. [7]에서는 동적 파티션을 통해 데이터 재할당이 가능하도록 하여 확장성을 지원한다. 시스템에 부하가 증가하면 파티션들은 상대적으로 부하가 적은 OTM에 재할당되거나 새로운 OTM을 생성하여 부하를 조정한다. HTM 또한 새로 생성되거나 제거되는 것이 가능하다. [8]에서 확장성 지원은 (1) 사용자의 관점에서 좀 더 많은 응용의 실행 능력이 필요할 때 응용 서버를 추가하고, (2) 데이터 볼륨이 늘어날 때 더 많은 DC 서버를 추가한다. 마지막으로 (3) 하나의 TC가 처리할 수 있는 트랜잭션보다 많은 트랜잭션이 도착하여 포화 상태가 되면 새로운 TC가 초기화될 수 있는 경우로 분리하여 확장성을 언급하였다. [9]에서는 실험 결과를 통하여 LTM과 클라우드 스토리지인 HBase 서버를 확장할 수 있음을 보였다.

2.3 응용 분야 및 연산

[7]에서 트랜잭션은 하나의 파티션 안에서 수행된다는 제한을 갖는다. 따라서 정적으로 파티션된 데이터를 접근하는 기업 응용에 적용가능하다. 파티션은 신축성과 확장성을 위해 동적으로 재구성되는 것이 가능하며, 판독 전용 트랜잭션과 갱신 트랜잭션을 모

두 허용한다. [8]에서 제안한 방법은 클라우드 환경에서 데이터가 어떻게 분산되었는지에 상관없이 트랜잭션이 수행되는 응용에 적용가능하다. 응용 분야에 대한 구체적인 예로, 클라우드에 기반한 SNS의 한 예로 웹과 모바일에서 수행가능한 MSRBooK을 구현하였다. MSRBook은 윈도우즈 애저를 사용하여 데이터를 관리하며 판독 전용 트랜잭션과 갱신 트랜잭션을 수행할 수 있다. [9]의 방법은 웹에서 수행되는 단기(short-lived) 트랜잭션이 주요 대상이다. 판독 전용 트랜잭션과 갱신 트랜잭션을 모두 허용하는데, 판독 전용 트랜잭션의 경우 스냅샷 상에서 수행되는 범위 질의를 허용한다. 고장에 대비하기 위해 트랜잭션 상태와 데이터는 중복되어 있으며, 완료를 위해 2단계 완료 프로토콜을 사용한다.

III. 제안하는 방법

이 장에서는 클라우드 환경에서 트랜잭션 처리를 위한 처리 구조와 모듈의 확장 알고리즘에 대하여 기술한다. 클라우드 환경에서 트랜잭션은 고유의 제약을 고려해야 하고, 작업 부하에 따라 모듈의 확장이 가능해야 한다. 2장에서 언급한 연구들에서는 각기 확장성의 지원을 언급하고는 있지만 구체적인 방법에 대한 논의가 없다. 따라서 이 논문에서는 클라우드 환경에서 트랜잭션 수행을 위한 처리 구조를 보이고, 그 모듈의 확장 알고리즘을 다룸으로써 확장시 고려해야 할 사항들에 대하여 살펴본다.

3.1 트랜잭션 처리 구조

이 논문에서 제안하는 시스템 모델은 트랜잭션 관리 모듈(TMM : Trasaction Management Module)과 데이터 관리 모듈(DMM : Data Management Module)로 구성된다. 트랜잭션 관리 모듈 안에는 트랜잭션 분배 기능을 수행하는 지역 트랜잭션 관리자 LTM (Local Transaction Manager), 잠금 관리를 하는 잠금 관리자 LM(Lock Manager), 그리고 클라우드 구성을 관리하는 구성 매니저 CM(Configuration Manager)이 존재한다. TMM은 클라이언트로부터 제출된 트랜잭션의 분배와 스케줄링을 담당하고, DMM은 스케줄된 연산을 수행하여 결과를 반환한다. 트랜잭션의 동시성 제어를 위해 잠금을 사용하며, 확장 알고리즘에 집중하기 위하여 데이터 중복과 고장은 고려하지 않는다. LTM은 트랜잭션이 제출되면 트랜잭션 수행 여부를 판단한다. 현재 LTM의 부하가 너무 크거나 제출된 트랜잭션이 접근하는 데이터가 현재의 노드에 없다면 다른 TMM으로 트랜잭션을 전송하고, 현재의 LTM이 대리자가 되어 트랜잭션의 수행을 감독한다. LM은 트랜잭션이 수행되기 전, 잠금 설정 여부를 결정하고, 성공적으로 잠금이 수여된 연산은 DMM으로 보내진다. 이 때 LTM은 DMM의 정보를 필요로 한다. 이 정보는 메타 데이터의 일부로 저장되며, 메타 데이터에는 현 클라우드 환경의 구성 정보도 포함되며, CM은 구성이 갱신될 때마다 메타 데이터를 갱신한다. DMM에서 성공적으로 수행된 연산은 LTM으로 반환되며 LTM은 분산 수행된 연산에 대해 조정자가 되어 완료 여부를 결정한다. 완료 프로토콜로는 2PC 프로토콜을 사용한다. 클라우드 환경에서 수행되는 트랜잭션의 예로는, 페이스 북과 같은 응용에서 친구 리스트를 갱신하거나 자신의 블로그에 사진을 업로드하는 등의 예를 생각할 수 있다. 그림 4는 논문에서 사용하는 시스템 모델이다.

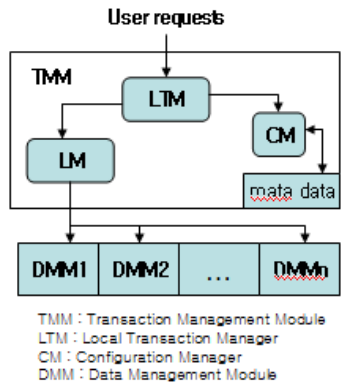

그림 4. 제안하는 시스템 모델
Fig. 4 System model

3.2 확장성의 제공

확장성의 제공은 두 가지 측면에서 고려되어야 한다. 클라이언트로부터 요구가 증가하여 TMM이 확장되는 경우와 데이터의 볼륨이 증가하여 DMM이 확장되는 경우이다.

1) TMM의 확장

시스템에 존재하는 TMM이 수행할 수 있는 능력 이상의 트랜잭션이 제출된다면 TMM의 확장이 필요하다. TMM은 쓰레드와 같은 소프트웨어 모듈이지만 TMM의 추가로 새로운 머신을 도입해야 하는 경우가 생길 수 있다. 하나의 머신에서 수행될 수 있는 소프트웨어 모듈이 무한하지 않기 때문이다. TMM이 확장될 때 DMM을 함께 확장할 것인지 여부를 고려해야 한다. TMM이 확장될 때 DMM을 확장하지 않으면 확장된 TMM으로 제출된 모든 트랜잭션의 수행은 결국 다른 TMM으로 전달되어야 하고, 확장된 TMM은 조정자의 역할만 담당하게 된다. 대부분의 TMM의 수행 능력이 한계에 도달한 상황에서 다른 TMM에서 전달된 연산으로 인한 부하가 추가로 발생한다면 TMM을 확장하는 효과를 누리기 어렵다. 따라서 이 논문에서는 효율을 위하여 TMM이 확장될 때는 하나 또는 기본적인 수의 DMM을 갖고 확장되는 방법을 사용한다. 또한, TMM이 확장되면 전체 시스템의 부하 균형을 위하여 기존 DMM의 재분배 여부를 고려할 수 있다. TMM이 확장될 때마다 전체 DMM을 재분배하게 되면 전체 시스템의 부하 균형을 달성할 수는 있지만 잦은 분배 절차로 인한 오버헤드 때문에 효율적이지 못하다. 따라서 TMM이 추가되면 DMM을 재분배하는 과정 없이 현재의 메타 데이터를 복사하여 제출되는 트랜잭션을 스케줄링하고, 추후 DMM이 추가될 때 새로운 TMM에 할당함으로써 재분배로 인한 오버헤드를 줄이고자 한다. 그림 5는 TMM 확장 알고리즘이다.

2) DMM의 확장

DMM이 확장되는 경우는 데이터 볼륨의 증가로 추가적인 스토리지가 필요한 경우이다. 따라서 DMM의 확장은 저장 공간인 스토리지 추가를 동반한다. DMM은 단독으로 존재할 수 없고 적어도 하나의 TMM의 제어를 받아야 하기 때문에 DMM을 확장할 때는 어떤 TMM의 제어를 받을지 결정해야 한다. DMM의 확장이 반드시 TMM의 확장을 의미하지는 않는다. 하나의 TMM이 제어가능한 DMM의 수는 유한하지만, 확장된 DMM을 제어할 TMM을 찾을 수 없는 경우에만 TMM이 확장된다. 그림 6은 DMM 확장 알고리즘이다.

```
Procedure Create_TMM() {
 if (sum of all TMM's capability > user's request) {
    create a TMM_new with a DMM;
    copy TMM's meta data to TMM_new;
    process a transaction;
 }
 update meta data; // 확장된 모듈 반영
}
```

그림 5. TMM 확장 알고리즘
Fig. 5 Expansion algorithm for TMM

```
Procedure Create_DMM() {
 if (any TMM exists !(is_FULL())) {
    DMM_new is attached to TMM;
 }
 else {  // all TMM is_FULL()
    call Create_TMM_new();
 }
 update meta data; // 확장된 모듈 반영
}
```

그림 6. DMM 확장 알고리즘
Fig. 6 Expansion algorithm for DMM

TMM과 DMM의 연결 구조에 대한 정보는 CM을 통하여 메타 데이터에 저장되며, 필요할 때마다 갱신된다. 메타 데이터는 모든 TMM에 전역적으로 유지되며, 한 모듈에서 확장이나 축소가 발생되면 다른 모듈에 갱신을 전파함으로써 정합성을 유지한다. 메타 데이터는 배열이나 벡터를 이용하여 구현가능하다.

Ⅳ. 분 석

이 장에서는 제안한 트랜잭션 처리 알고리즘의 정확성을 보이기 위해 트랜잭션의 ACID 특성 측면과 모의 실험 결과에 대하여 기술한다.

트랜잭션의 원자성은 연산들이 전부 수행되거나 전혀 수행되지 않아야 유지된다. 분산 수행되는 연산들이 존재할 때 원자성은 완료 프로토콜에 의해 보장된다. 많은 연구들을 비롯하여 이 논문에서도 2PC 프로토콜을 사용하여 트랜잭션의 원자성을 보장한다.

일관성은 트랜잭션의 수행으로 데이터베이스 상태가 하나의 일관된 상태에서 다른 일관된 상태로 전환되어야 보장되며, 일관성은 트랜잭션의 무결성 제약조건으로 달성된다. 이 성질은 무결성 제약을 가진 트랜잭션이 정확하게 수행된다면 보장할 수 있다.

고립성은 트랜잭션들 사이에 간섭 현상이 발생하지 않도록 트랜잭션의 중간 결과를 다른 트랜잭션이 접근하지 못하도록 제어함으로써 달성할 수 있다. 이는 잠금이나 타임스탬프 기법으로 연산의 충돌을 방지함으로써 가능하다. 제안한 방법에서 사용하는 잠금 프로토콜에 의해 충돌 관계의 연산은 동시에 수행되지 않으므로 달성된다고 할 수 있다.

영구성은 한번 데이터베이스에 반영된 결과는 고장이 발생하더라도 영향 받지 않아야 한다는 성질이다. 이는 시스템에 고장이 발생했을 때 회복 기법에 의해 달성되는 특성이다. 이 논문에서는 고장을 가정하지 않았으므로 영구성은 논의하지 않는다.

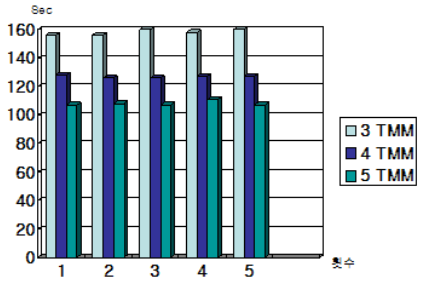

그림 7. 트랜잭션 응답 시간
Fig. 7 Response time

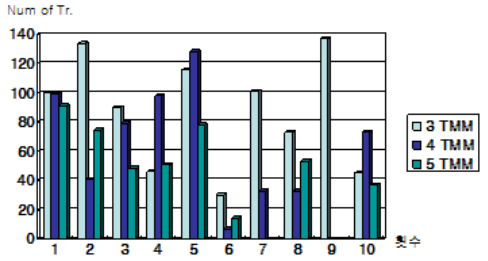

그림 8. 트랜잭션 철회율
Fig. 8 Abort ratio

제안한 방법의 성능 분석을 위하여 트랜잭션 응답 시간과 트랜잭션 철회율에 대하여 모의 실험하였다. 응답 시간은 동일한 개수의 트랜잭션을 TMM의 수를 변화시켜 수행할 때, 트랜잭션이 종료되는 시간을 측정하는 것으로 실험하였는데, TMM의 수가 3~5로 증가할수록 응답 시간이 감소함을 알 수 있었다. 그림 7은 응답 시간에 대한 실험 결과이다.

트랜잭션 철회율을 측정하기 위하여 사용자의 트랜잭션이 기록 연산일 때 DMM 용량을 고정하고 저장 공간의 부족으로 철회되는 트랜잭션 수를 측정하였는데, 그림 8은 1500개의 트랜잭션에 대하여 DMM의 확장이 허락되지 않을 때 철회되는 트랜잭션 수를 보여준다. DMM의 저장 공간 비율을 변동시킨 실험도 실시하였는데, 저장 공간이 늘어날수록 철회율은 줄어들었다. 모의 실험은 32비트 개인용 컴퓨터에서 자바를 사용하여 구현하였고, 판독 연산과 기록 연산의 비율은 2:1로 가정하였다.

V. 결 론

이 논문에서는 최근 SNS를 포함하여 널리 활용되고 있는 클라우드 환경에서 트랜잭션을 처리할 때 필요한 확장성 제공 방법에 대하여 제안하였다. 클라우드 환경에서 트랜잭션 처리 방법을 제안한 많은 연구들은 대부분 확장성보다는 처리 알고리즘에 집중하고 있다. 따라서 이 논문에서는 클라우드 환경에서 트랜잭션 처리를 위한 처리 구조와 모듈의 확장 알고리즘에 대해 다루었다.

트랜잭션 처리 구조는 트랜잭션 관리 모듈 TMM과 데이터 관리 모듈 DMM으로 구성되고, TMM 안에는 지역 트랜잭션 관리자와 잠금 관리자, 그리고 구성 관리자가 존재하고, 메타 데이터가 저장된다. DMM은 실제 데이터를 소유한 모듈이며, 연산을 수행하여 결과를 반환한다.

확장성을 지원하기 위하여, 확장가능한 모듈은 TMM과 DMM이다. 하나의 TMM이 처리할 수 있는 트랜잭션보다 많은 트랜잭션이 제출될 때 TMM은 확장된다. TMM이 확장될 때는 기존의 TMM과 DMM의 연결 정보가 저장된 메타 데이터를 복사하고, 기존 TMM들의 수행 부담을 줄이기 위해 DMM과 함께 확장하도록 하였다. DMM은 데이터 저장 공간을 추가로 확보하기 위하여 확장된다. DMM이 확장될 때는

특정 TMM의 제어를 받도록 해야 하며, 모든 TMM이 더 이상 DMM을 추가할 수 없을 때 TMM의 확장을 동반한다. 제안하는 알고리즘의 성능 분석을 위해 응답 시간과 트랜잭션 철회율에 대한 모의 실험을 실시하였다.

참고 문헌

[1] 데이코 산업 연구소, 클라우드 컴퓨팅, 차세대 컴퓨팅 기술/시장 동향과 사업 전략, 데이코 산업 연구소, pp. 21-22, 2010.

[2] D. J. Abadi, "Data management in the cloud: Limitations and opportunities", IEEE Data Engineering Bulletin, Vol. 32, No. 1, Mar. 2009.

[3] Ashish Thusoo, Zheng Shao, et al., “"Data warehousing and analytics infrastructure at facebook"", SIGMOD'10 Proceedings of the 2010 international conference on Management of data ACM, pp. 1013-1020, 2010.

[4] 송용주, “대용량, 클라우드 서버 환경에서 DBMS가 고민해야 할 문제”, 정보과학회지, 29권, 5호, pp. 37-44, 2011.

[5] S. Gilbert and N. A. Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services", SIGACT News, Vol. 33, No. 2, pp. 51-59, 2002.

[6] W. Vogels, "Eventually Consistent", Communication ACM, Vol. 52, No. 1, pp. 40-44, 2009.

[7] S. Das, D. Agrawal, and A. El Abbadi, "Elastras: an elastic transactional data store in the cloud," in USENIX HotCloud, Article No. 7, 2009.

[8] Justin J. Levandoski, David Lomet, Mohamed F. Mokbel, and Kevin Keliang Zhao, "Deuteronomy: Transaction support for cloud data", In 5th Biennial Conf. on Innovative Data Systems Research(CIDR), Asilomar, CA, USA, January, pp. 123–133, 2011.

[9] Z. Wei, G. Pierre, and C.-H. Chi. "Scalable Transactions for Web Applications in the Cloud", In Proceedings of the Euro-Par Conference on Parallel Processing, pp. 442-453, 2009.

[10] P. A. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, pp. 17-23, 1987.

[11] M. K. Aguilera, A. Merchant, M. Shah, A. Veitch and C. Karamanolis, "Sinfonia: A New Paradigm for Building Scalable Distributed Systems", In SOSP, pp. 159-174, 2007.

[12] Mike Burrows, “"The Chubby lock service for loosely coupled distributed systems"", OSDI'06 Proceedings of the 7th conference on Symposium on Operating Systems Design and Implementation, pp. 335-350, 2006.

[13] L. Lamport. “The part-time parliament.”, ACM Trans. Comput. Syst., Vol. 16, No. 2, pp. 133-169, 1998.

[14] 박경욱, 김경옥, 반경진, 김응곤, “클라우드 기반 센서 데이터 관리 시스템 설계 및 구현”, 한국전자통신학회논문지, 5권, 6호, pp. 672-677, 2010.

저자 소개

김치연(Chi-Yeon Kim)

1992년 전남대학교 전산통계학과 졸업 (이학사)

1994년 전남대학교 대학원 전산통계학과 졸업(이학석사)

1999년 전남대학교 대학원 전산통계학과 졸업(이학박사)

2002년~현재 목포해양대학교 해양컴퓨터공학과 교수

※ 관심분야 : 트랜잭션 관리, 클라우드 컴퓨팅