이질 데이터간의 트랜잭션 처리를 위한 AEM의 설계 및 구현

박성은[0] 이정태
아 주 대 학 교 컴 퓨 터 공 학 과
{aceoface, jungtae}@madang.ajou.ac.kr

Design and Implementation of AEM for supporting transaction between DBS data and Multimedia data

Sung-eun Park[0] Jung-tae Lee
College of Computer & Information, Ajou University

요 약

현재의 컴퓨팅 환경은 멀티미디어와 인터넷, 즉 웹 환경으로 요약된다. 이러한 환경에서 운영되는 복잡한 정보 시스템-인터넷 TV 방송, 문서 관리 시스템, 가상 대학, 그룹 웨어, 지식 경영 시스템 등-은 대용량의 데이터베이스 데이터와 멀티미디어 데이터를 기반으로 해서 동작한다. 이러한 시스템들에서의 데이터는 데이터 베이스 내에 존재하는 메타 데이터와 운영체제의 파일 시스템 내에 존재하는 파일 데이터로 나뉜다.

이러한 데이터의 분리로 인해서 나타나는 가장 큰 문제점은 이 둘 데이터 사이의 불일치성(inconsistency)이다. 이는 멀티미디어 데이터와 데이터베이스 데이터를 접근해서 사용하는 기능(operation)의 차이로 인해서 발생하게 된다.

본 논문에서는 복잡한 정보 시스템에서 사용되어 지는 데이터들 사이에 발생되는 불일치성을 해결하며, 쉬운 프로그래밍을 위한 응용 프로그래밍 인터페이스(API)를 제공해주는 AEM(Activity Execution Manager) 프레임워크를 설계 및 구현한다.

1. 서론

현재의 컴퓨팅 환경은 멀티미디어와 웹으로 요약된다. 이러한 환경에서 운영되는 복잡한 정보 시스템들은 데이터의 구조적 측면에서는 대용량의 데이터베이스와 멀티미디어 데이터를 기반으로, 시스템 자체의 구조적 측면에서는 보통 3 계층(3-tier)구조[1]로 구성된다. 이러한 시스템에서 데이터간의 특징은 크게 물리적 분리와 논리적 분리로 나뉜다. 물리적 분리는 복잡한 정보 시스템들이 대부분 메타 데이터를 저장하기 위해서 데이터베이스를 사용하고 실제 서비스에 필요한 주요 내용들은 디스크상에 파일의 형태로 가지게 됨으로 해서 발생하게 된다. BLOB형식의 레코드를 사용해서 물리적 분리를 허용하지 않을 수도 있지만 이는 전체적인 성능에 악영향을 끼치기 때문에 거의 사용하지 않는다. 따라서 각 데이터를 접근해서 사용하는 기능(operation)들의 차이에 의한 논리적 분리도 나타나게 된다. 데이터 베이스 데이터(DBD)와 멀티미디어 데이터(MMD)에 대한 기능과 제공 환경의 측면에서 논리적 분리의 특성은 다음 표1과 같다.

각 데이터의 속성의 차이로 인해 나타나는 가장 큰 문제점은 무결성이다. 서버 시스템에서 사용자의 요구를 처리하기 위해서 DBD와 MMD를 모두 사용한다고 할 때, DBD를 처리하고 MMD처리에서 오류가 발생하면 DBD를 처리하는 기능의 속성으로 인해서 롤백(rollback)을 할 수 없다. 또한 MMD를 처리하고 DBD에서 오류가 생기면 MMD처리 기능의 한계로 인해서 마찬가지로 롤백을 할 수 없기 때문에 어느 경우든지 오류가 발생했을 때 기능의 속성 또는 한계로 인해서 무결성을 지킬 수 없게 된다.

표1. 논리적 분리의 특성

	기능		제공 환경	
	종류	속성	직렬성	지원 시스템
DBD	트랜잭션	ACID[2]	예	DBMS
MMD	FSC[3]	D	아니오	OS kernel

2. 관련연구

데이터의 물리적, 논리적 분리[4]의 측면에서 봤을 때, 멀

[1] 클라이언트-서버-[DB서버, 파일 서버]

[2] Atomicity, Consistency, Isolation, Durability

[3] File System Call

[4] 기능 속성의 차이점뿐만 아니라 같은 기능을 이용하더라도 서로 독립적으로 실행되는 것까지 포함

티데이터베이스 시스템(MDBS)의 전역 트랜잭션 매니저(GTM, Global Transaction Manager) 에서 발생하는 문제점과 본 논문에서 지적하고 있는 문제점은 일치한다. 무결성을 보장하기 위해서는 atomicity 와 직렬성도 같이 보장이 되어야 한다.

전역 atomicity 를 위해서는 전역 commit 프로토콜이 필요하며 일반적으로 two -phase commit 프로토콜(2PC)을 사용한다. 전역 트랜잭션과 하위 트랜잭션의 동기 문제를 해결하기 위한 semantic recovery technique 과 redo technique[1] 이 있다. 전역 직렬성에 대해서는 optimistic ticket method[2] 가 가장 많이 사용된다. 이는 전역 직렬성 그래프(GSG, Global Serialization Graph)를 이용한다. 전역 무결성은 GTM 의 동시성 제어(Concurrency control) 전략에 많이 좌우 되며 global serializability 와 two -level serializability[3] 방법이 있다.

3. AEM의 설계

Activity Execution Manager(AEM) 은 웹 기반의 복잡한 정보 시스템을 위한 프레임 워크이다. AEM 에서는 클라이언트에서 요청되는 일련의 작업들을 하나의 activity라는 단위로 묶어서 처리한다.

3.1 Activity

Activity란 AEM 에서 처리되는 일의 단위이며, 분산 트랜잭션과 비슷한 모습을 가진다. Activity를 클라이언트/서버(C/S) 프로그래밍 모델에 입각해서 정의하면 하나의 트랜잭션으로 처리되어져야 하는, 클라이언트에서 서버로 보내는 요청들의 모음으로 정의할 수 있다.

3.2 Partial abort operation

C/S 프로그래밍과 트랜잭션 프로그래밍 유형을 살펴보면 그림 1과 같은 경우가 많을 것이다.

```
1: begin()              8:  else
2:  req1                9:   req3
3:  res1                10:  res3
4:  if(cond1)           11: end if
5:     Abort req1       12: commit()
6:   req2
7:   res2
```

그림 1. 프로그래밍 유형

그래서 AEM 에서는 Activity 에 대한 partial abort 를 지원하기 위해서 *bulb*와 *turn on*, *turnoff*개념을 사용한다. **오류! 참조 원본을 찾을 수 없습니다.**의 (a) 와 같이 *turn on*은 전구를 켜는 것으로 *turn on* 이후의 activity 들은 새로운 전구가 나타날 때 까지 계속해서 같은 전구에 연결이 된다. (b) 는 turnoff 에 대한 그림이다. *Turnoff*를 하게 되면 *turn on* 으로 인해서 연결된 모든 activity 들을 abort 시킨다. (c) 는 트랜잭션 자체를 abort 시킨 것으로 모든 activity 들이 abort 된다.

3.3 AEM 구조(Architecture)

AEM은 기본적으로 4 계층(4-tier)의 구조를 가진다.

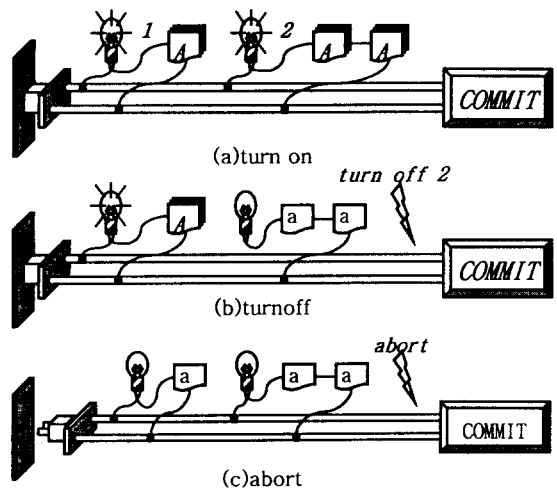

그림 2. Partial abort operations

계층1인 클라이언트 부분은 사용자의 인터페이스를 담당하는 애플릿으로 구성된다. 계층2는 단일 AEM 서버로 구성되며 이곳에서 activity 의 분해(decomposition)와 전역 무결성을 책임지게 된다. 계층3은 각 지역 데이터베이스 시스템 또는 지역 파일 시스템 위에서 운영되는 서버 프로세스인 래퍼(wrapper)서버가 위치한다. 파일 시스템인 경우 FSC 만으로는 원하는 기능을 얻을 수 없기 때문에 트랜잭션을 지원하도록 파일 래퍼서버가 필요하다. 각 래퍼서버에서는 AEM 에 의해서 분해된 activity들의 실행을 책임지며, 필요하다면 simulate prepare-to-commit상태를 지원한다. 마지막으로 계층4에는 실제적인 데이터가 저장되는 데이터베이스 시스템과 파일 시스템이 위치하게 된다.

3.4 AEM설계 전략

AEM에서 전역 atomicity, 전역 직렬성, 그리고 전역 무결성을 보장하기 위해서 관련연구에서 제안된 다음의 방법들을 사용한다.

- 전역 atomicity
  - two-phase commit(2PC) 프로토콜
  - prepare-to-commit 또는 simulated prepare-to-commit 상태 지원
- 전역 직렬성
  - optimistic ticket method
  - global serialization graph(GSG)
- 전역 무결성
  - 값 의존성을 이용한 flow graph(FG)
  - global integrity constraints(GIC)
- 복구(recovery) 전략
  - semantic recovery technique(undo technique)

3.5 AEM의 commit과정

AEM에서 activity를 commit할 때는 전역 atomicity, 전역 직렬성, 그리고 전역 무결성을 보장하기 위해서 여

러 단계를 거치게 된다. 각 단계에 대한 설명은 다음과 같다.

① Flow graph를 생성해서 전역 무결성을 깨뜨리지 않는지를 검사한다.
② AEM은 activity에서 접근한 각 래퍼(wrapper) 서버에게 prepare-to-commit을 보낸다.
③ 래퍼서버는 필요에 따라서 simulated prepare-to-commit상태로 된다. Ticket 값을 AEM에서 보낸다.
④ AEM서버는 돌려 받은 ticket 값을 이용해서 GSG를 만든다.
⑤ GSG의 사이클(cycle)을 검사한다. GSG가 사이클을 이루게 되면 abort 시키고 아니면 activity를 완전히 commit시킨다.

3.6 AEM 인터페이스

AEM에서는 전통적인 트랜잭션 프로그래밍 모델인 begin, commit, abort의 형식을 지원한다. 이는 프로그래머가 AEM을 이용해서 쉽게 프로그래밍을 할 수 있도록 하기 위함이다.

```
1: string begin () // return activity id, aid
2: string commit (string aid)
3: string abort (string aid)
4: string turn_on (string) // return bulb id, bid
5: string turn_off (string aid, string bid)
```

그림 3. AEM 인터페이스

위의 인터페이스는 activity를 제어하기 위한 가장 기본적인 함수들만을 가진 것이다. 실제 서비스를 위해서는 이 인터페이스를 상속 받아서 적절한 함수를 추가해야 한다.

4. 결론 및 향후 계획

본 논문에서는 웹을 기반으로 한 복잡한 정보 시스템에 적합한 Activity Execution Manager 를 설계 및 구현하는 방법을 알아보았다. AEM 에서는 여러 개의 요청들을 하나의 트랜잭션으로 처리하면서 partial abort 를 지원하는 activity 를 사용한다. 이로써 클라이언트 프로그래머는 데이터 베이스 데이터와 멀티미디어 데이터에 대해서 전역 atomicity, 전역 직렬성, 전역 무결성을 고려하지 않아도 손쉽게 프로그램을 작성할 수 있다. 또한 activity프로그래밍 모델이 기존의 전통적인 트랜잭션 프로그래밍 모델과 같기 때문에 쉽게 프로그래밍을 할 수 있다. 따라서, AEM 은 복잡한 정보 시스템의 구축에 필요한 시간과 경비를 줄이는데 많은 도움이 될 것이다. AEM에서 commit 과정은 two -phase commit protocol(2PC)를 사용한다. 2PC 는 그 자체로도 한번의 commit을 위해서 많은 메시지를 주고 받기 때문에 성능이 좋지 않다. 또한 부가적으로 AEM 에서는 무결성과 직렬성을 위해서 별도의 작업을 수행한다. 그래서 AEM의 commit 과정은 많은 단계로 인해서 높은 성능을 보이지 않는다. 2PC 를 비롯해서 무결성과 직렬화의 검사 단계를 줄이는 방안을 연구해서 commit 과정에 걸리는 시간을 단축해야 한다. 현재 AEM 의 구현은 클라이언트와 AEM 서버가 TCP/IP 소켓을 통해서 통신을 하며, 각 래퍼서버는 오브젝트의 형태로 AEM 서버 프로세스안에서 쓰레드로 실행되도록 되어 있다. AEM 이 현재 많이 사용되고 있는 클라이언트/서버 환경을 바탕으로 구현이 되었지만 앞으로는 CORBA 나 EJB 를 이용해서 분산 객체 환경으로 전환되어야 할 것이다.

5. 참고 문헌

[1]. Sharad Mehrotra, Rajeev Rastogi, Yuri Breitbar, Henry F. Korth, and Avi Silberschatz, " Ensuring transaction Atomicity in Multidatabase Systems ", Proceedings of the 11 th ACM SIGACT -SIGMOD-SIGART symposium on Principlies of database systems, 1992

[2]. Thomas Tesch, and J gen WDch, " Global nested transaction management for ODMG -compliant multi - database systems ", Proceedings of the 6 th international conference on Information and knowledge management, 1997, Pages 67-74

[3]. Panayiotis K. Chrysanthis, and Krithi Ramamritham, " ACTA: A Framework for Specifying and Reasoning about transaction Structure and Behavior", Proceedings of the 1990 ACM SIGMOD international conference on Management of data, 1990, Pages 194-203

[4]. Claire Morpain, Mich èle Cart, Jean Ferri é and Jean-François Pons, " Maintaining database consistency in presence of value dependencies in multidatabase systems ", Proceedings of the 1996 ACM SIGMOD international conference on Management of data, 1996, Pages 459-468

[5]. Peter Scheuermann, and Hsiang -Lung Tung, " A recovery s cheme for multidatabase systems ", Proceedings of the second international conference on Information and knowledge management, 1993, Page 665

[6]. Jim Gray, and Andreas Reuter, " Transaction Processing: Concepts and Techniques ", Morgan Kaufmann Publishers, 1992

[7]. Philip A. Bernstein, and Eric Newcomer, " Principles of transaction Processing ", Morgan Kaufmann Publishers, 1997

[8]. Thomas Connolly, Carolyn Begg, and Anne Strachan, " Database Systems: A Practical Approach to Design, Implementation and Managemen t", Addison-Wesley, 1997

[9]. D. Georgakopoulos, M, Rusinkewicz, and A. P. Sheth, " Using tickets to enforce the serializability of multidatabase transaction ", IEEE transactions on Knowledge and Data Engineering, 6(1), February, 1994

[10]. Dan Chang, and Da n Harkey, " Client/Server Data Access with Java and XML ", Wiley Computing Publising, 1998

[11]. McGrath Seam, " XML by Example: Building E - Commerce applications", Prentice Hall, 1998