AOP 기반의 트랜잭션 라우팅 알고리즘

강현식*, 이석훈**, 백두권***†
*고려대학교 컴퓨터 정보통신대학원 컴퓨터 소프트웨어공학과
**고려대학교 정보통신대학 컴퓨터·전파통신공학과
***고려대학교 융합소프트웨어전문대학원
email : {cj848, leha82, baikdk}@korea.ac.kr

AOP-based Transaction Routing Algorithm

Hyun-Sik Kang*, Sukhoon Lee**, Doo-Kwon Baik***†
* Dept. of Software Engineering, Korea University
** Dept. of Computer and Radio Communications Engineering, Korea University
*** Graduate School of Convergence IT, Korea University

요 약

데이터베이스 복제(Replication)는 데이터의 저장과 백업하는 방법과 관련이 있는 데이터를 호스트 컴퓨터에서 다른 컴퓨터로 복사하는 것이다. 데이터베이스 복제의 종류로는 마스터/슬레이브(Master/Slave), 멀티마스터(Multi-Master)형태가 존재한다. 멀티마스터 데이터베이스는 상당한 비용증가와 복잡도 때문에 어떤 상황에서는 실용적이지 않다. 그러한 이유로 마스터/슬레이브 데이터베이스를 많이 사용한다. 마스터/슬레이브 데이터베이스에서 마스터 데이터베이스는 데이터베이스의 변경을 담당하고 그 결과는 슬레이브로 복제된다. 슬레이브 데이터베이스에서는 읽기 전용 질의만 처리하는 것을 목적으로 한다. 데이터베이스 트랜잭션 처리는 대표적인 횡단 관심사이다. 연구는 마스터/슬레이브 데이터베이스의 트랜잭션을 AOP 를 사용하여 횡단 관심사로 분리하고, 마스터/슬레이브 데이터베이스를 라우팅하는 알고리즘을 제안한다.

1. 서론

데이터베이스 복제(Replication)는 분산 데이터베이스 환경에서 데이터를 각 데이터베이스에 복제하여 신뢰성, 가용성, 과부하 방지 등을 위하여 이용된다. 이를 위하여 마스터/슬레이브(Master/Slave), 멀티마스터(Multi-Master) 데이터베이스와 같은 다양한 모델이 존재한다[1]. 멀티마스터 데이터베이스의 경우 다중 마스터에 동시 다발적으로 데이터의 삽입 및 갱신이 일어날 때 동기화 문제로 인하여 성능에 저하에 대한 부담이 존재한다. 그러나 마스터/슬레이브 데이터베이스의 경우 마스터에서만 삽입/갱신/삭제가 일어나므로 슬레이브는 상대적으로 동기화의 부담은 적다[2].

이러한 이유로 데이터의 삽입 및 수정이 빈번히 일어나는 환경에서는 주로 마스터/슬레이브 데이터베이스 모델이 이용된다. 하지만, 마스터/슬레이브 데이터베이스 모델을 사용하는 시스템에서 마스터에 접속할지 슬레이브에 접속할지를 선택 하는 것에 대한 기준은 따로 존재하지 않는다. 이 경우에는 일반적으로 프론트엔드 어플리케이션이나 중간의 미들웨어 단계에서 트랜잭션 라우팅(Transaction Routing)이 수행된다.

한 편, 관점 지향 프로그래밍(AOP, Aspect Oriented Programming)은 소프트웨어를 핵심 관심사와 횡단 관심사로 분리하여 모듈화시키는 프로그래밍 기법이다[3]. 기존의 객체 지향 프로그래밍은 다수의 객체들에 분산적으로 중복되어 있는 공통 관심사가 발생하게 되어 있어 프로그래밍 유지보수 및 가독성이 낮다는 단점을 지닌다. AOP 는 공통 관심사를 횡단 관심사라고 정의하여 모듈 형태로 분리하여 핵심 관심사에 직조(Weaving)하는 형태로 프로그래밍하는 기법이다.

이 논문은 마스터/슬레이브 데이터베이스 모델에서 AOP 기반의 트랜잭션 라우팅 알고리즘을 제안한다. AOP 에 기반하여 마스터/슬레이브 데이터베이스의 트랜잭션 라우팅을 할 경우, 트랜잭션 라우팅에 대한 관심사를 모듈화 하여 각각의 어플리케이션에서 그대로 적용하는 것이 가능하므로 투명한 데이터 접근이 가능해진다. 제안하는 알고리즘은 로그인, 업로드, 댓글 목록과 같은 핵심 관심사에 트랜잭션 라우팅을 위한 횡단 관심사를 적용하여 접근하기 위한 데이터베이스가 마스터인지 슬레이브인지를 선택한다.

이 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구를 기술하고, 제 3 장에서는 제안하는 웹 어플리케이션의 구조와 제안 알고리즘을 기술하며, 제 4 장에서는 이를 통한 실험 결과를 평가 및 검증하며, 제 5 장에서는 결론 및 향후 연구에 대해 제시한다.

이 논문은 2014 년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No.2012M3C4A7033346).
† 교신저자

2. 관련 연구

Leg@Net [4]는 데이터베이스 클러스터 신선도인지(Freshness-Aware) 트랜잭션 라우팅 기법을 사용한다. 이 연구는 멀티마스터 데이터베이스에서 트랜잭션을 라우팅하여 데이터베이스에 복제하며, 데이터베이스의 일관성 및 성능을 유지하는 미들웨어로 동작한다.

Takshkent+ [5]는 복제된 데이터베이스에서의 메모리 인지 로드 밸런싱 및 업데이트 필터링 기법이다. 이는 멀티마스터 데이터베이스에서 트랜잭션에 사용되는 작업량을 측정하여 부하가 가장 적은 데이터베이스에서 실행되게 한다.

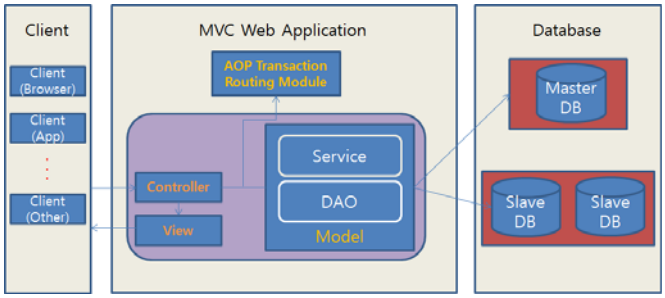
하지만 앞서 언급한 트랜잭션 라우팅에 관련된 연구들은 주로 멀티마스터 데이터베이스에서 어떤 마스터에 연결할 것인가에 대한 기법으로 마스터/슬레이브 환경을 고려하지 않는다.

Django [6]는 마스터/슬레이브 데이터베이스 환경에서 웹 어플리케이션의 개발을 위하여 파이썬 기반의 웹 프레임워크를 제공한다. Django 는 구체적인 마스터/슬레이브에 대한 라우팅 전략을 가지고 있지 않으므로, 어플리케이션을 개발할 때 마다 각각의 트랜잭션 라우팅을 가능하게 코드를 수정해야 한다.

3. AOP 기반의 트랜잭션 라우팅 알고리즘

3.1 제안 알고리즘을 적용한 어플리케이션

이 연구에서 제안하는 알고리즘을 적용한 어플리케이션 구조는 (그림 1)와 같다.

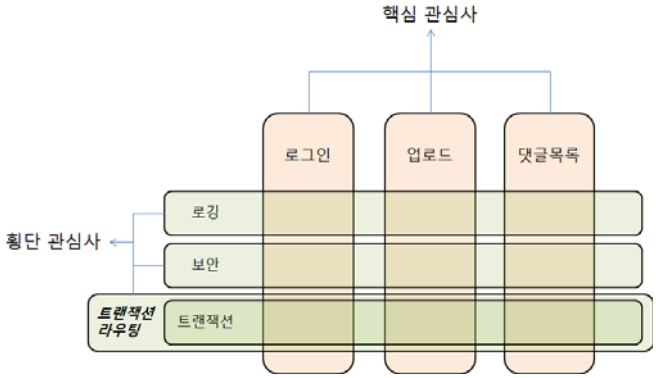


(그림 1) 제안 알고리즘을 적용한 어플리케이션 구조

클라이언트(Client)는 웹 어플리케이션에 비즈니스 로직(Business Logic)을 요청하고 어플리케이션에서는 컨트롤러(Controller)에서 클라이언트의 요청을 처리하여 최종적으로 뷰(View)를 통해서 결과를 반환한다. 이 과정에서 클라이언트의 요청이 컨트롤러에서 모델의 일부분에 해당하는 서비스(Service)의 메소드를 호출하게 되는데 컨트롤러에서 서비스의 메소드로 진입하기 전에 AOP 트랜잭션 라우팅 모듈(AOP Transaction Routing Module)로 직조 되어 제안하는 알고리즘에서 일치되는 조건에 만족할 경우 슬레이브 데이터베이스(Slave DB)로 선택하고 그렇지 않을 경우 마스터 데이터베이스(Master DB)로 선택한다. 서비스에서는 비즈니스 로직을 수행하며 DAO(Data Access Object)에서는 서비스에서 수행하는데 필요한 데이터베이스의 자료를 접근 및 수정을 한다. 이 과정에서 DAO 에서는 AOP 트랜잭션 라우팅 모듈에서 선택된 데이터베이스에 접근하여 질의를 수행한다. 제안하는 알고리즘이 사용되는 시스템의 구조를 보면 클라이언트의 모든 요청은 컨트롤러를 거치게 된다. 컨트롤러에서도 AOP 트랜잭션 라우팅 모듈 적용이 가능하나 컨트롤러에서만 AOP 트랜잭션 라우팅 모듈을 적용할 경우 웹 어플리케이션뿐만 아니라 같은 모델을 사용하는 다른 어플리케이션이 존재할 경우 정상적으로 트랜잭션 라우팅이 불가능하게 된다.

3.2 제안 알고리즘

이 절에서는 연구에서 제안하는 트랜잭션 라우팅에 대한 설명과 AOP 명세, 라우팅 전략 및 예외사항 처리를 설명한다. (그림 2)는 로그인, 업로드, 댓글목록과 같은 핵심 관심사 예와 핵심 관심사에 중복으로 들어 있는 공통 관심사인 횡단 관심사를 로깅, 보안, 트랜잭션 및 트랜잭션 라우팅 등의 예로 분류 하였다.



(그림 2) 핵심 관심사 및 횡단 관심사 적용 예

AOP 에서는 아래와 같은 새로운 개념을 정의하였다[3].

1. 관점(Aspect) : 관점은 구현하고자 하는 횡단 관심사의 기능.
2. 결합점(Join Point) : 결합점은 관점을 삽입하여 실행 가능한 어플리케이션의 특정 지점.
3. 포인트컷(Pointcut) : 결합점의 집합.
4. 충고(Advice) : 충고는 관점의 실제 구현체로 결합점에 삽입되어 동작하는 코드.
5. 직조(Weaving) : 직조는 관심을 대상 객체에 적용하여 기존의 객체와 다른 새로운 객체를 생성.

<표 1> AOP 트랜잭션 라우팅 알고리즘 AOP 명세

AOP 구성요소	설명
관점	트랜잭션 라우팅
결합점	모든 서비스의 메소드가 시작되기 전
포인트컷	execution(* package.name..*Service*.*(..))
충고	Around

<표 1>은 위에서 정의된 개념을 통해 이 논문에서 제안하는 트랜잭션 라우팅의 AOP 명세를 나타낸다. <표 1>에서 충고가 Around 이기 때문에 서비스의 메소드가 실행되기 전에 직조되어 마스터/슬레이브 데이터베이스를 선택을 하게 된 후 원래 실행될 예정이었던 서비스 메소드를 실행하게 된다. 제안하는 알고리즘은 <표 2>과 같다.

<표 2> 제안하는 AOP 트랜잭션 라우팅 알고리즘

```
procedure ROUTING(M, D, A, J):
  if D is null then
    if A is null then
      if M contains find or M contains get then
        set D to SLAVE;
      else set D to MASTER;
    else set D to A.VALUE;
    try
      J.proceed();
    finally
      set D to null;
  else J.proceed();
end
```

각 변수는 아래와 같다.

· M = 메소드 명칭
· D = 선택될 데이터베이스
· A = @RoutingDataSource 어노테이션 값
· J = 결합점 정보를 담고 있는 객체

제안하는 알고리즘은 메소드의 이름을 기반으로 라우팅한다. 슬레이브 데이터베이스만 사용해야 할 경우는 트랜잭션에서 읽기 전용 에 해당하는 트랜잭션만 포함하며 해당 메소드들은 보통의 어플리케이션에서 get, find 와 같은 형태의 이름을 띄게 된다. 그러나 그러한 메소드의 경우라 할지라도 마스터 데이터베이스를 사용해야 할 경우가 존재한다. 다음과 같은 상황에서는 get, find 와 같은 형태의 메소드 일지라도 마스터 데이터베이스를 선택할 수 있게 하였다.

1. 대상 메소드에 @RoutingDataSource 어노테이션이 존재하고 어노테이션의 value 가 DataSourceType.마스터 값을 가질 경우
2. 대상 메소드가 호출되기 전에 이미 다른 서비스 레이어에서 마스터 DB 가 선택된 경우

@RoutingDataSource 어노테이션은 연구를 위해 사용된 자바의 커스텀 어노테이션으로 MASTER 와 SLAVE 값을 가질 수 있으며 각 데이터베이스에 매핑된다. 위 설명에서 예외 적인 경우가 발생하는 이유는 해당 어플리케이션이 이름을 기반으로 데이터베이스를 선택한다는 보장이 없을 수 있으며, 이미 해당 서비스가 호출되기 전에 다른 서비스에서 마스터/슬레이브 데이터베이스를 선택 했을 경우에는 선택할 필요가 없기 때문이다. 그렇기 때문에 서비스에서 실행되는 모든 메소드에 대해서 직조한 뒤에 메소드 이름을 검사하는 등의 알고리즘이 실행된다.

4. 실험 평가

4.1 실험 환경 및 방법

이 연구에서는 자바의 AspectJ[3] AOP 의 구현체로 사용 하였다. 실험은 이미 구현된 A 사의 문서 편집 클라우드 스토리지 서비스의 테스트 베드 환경으로 테스트를 진행 하였다. 테스트에 사용된 A 사의 서비스는 RESTful API(Application Programming Interface)를 통한 테스트의 자동화가 가능하다. 해당 서비스는 JDK 1.7.0.40, Spring Framework 4.0.0, Hibernate ORM Framework 4.3.1, 마스터/슬레이브 구성은 MySQL 5.5.31 을 사용 하였고, 멀티마스터의 경우 동일 MySQL 버전과 Galera Cluster 를 사용 하였다. WAS(Web Application Server)의 경우 Apache Tomcat 7.0.39 버전을 사용하였다. 테스트 클라이언트로 NHN 의 nGrinder 의 사전에 정해진 트랜잭션 시나리오를 Python 스크립트로 구현하였다.

실험을 위하여 가상의 50 명 유저가 12 시간동안 서비스를 사용한다는 가정으로 테스트를 진행하였다. 50 명의 유저는 테스트베드 환경에서 순수하게 트랜잭션 라우팅 성능을 확인할 수 있을 정도의 유저수/부하이며 그 이상이 될 경우 데이터베이스 성능 한계로 인해 제대로 된 결과를 도출할 수 없다.

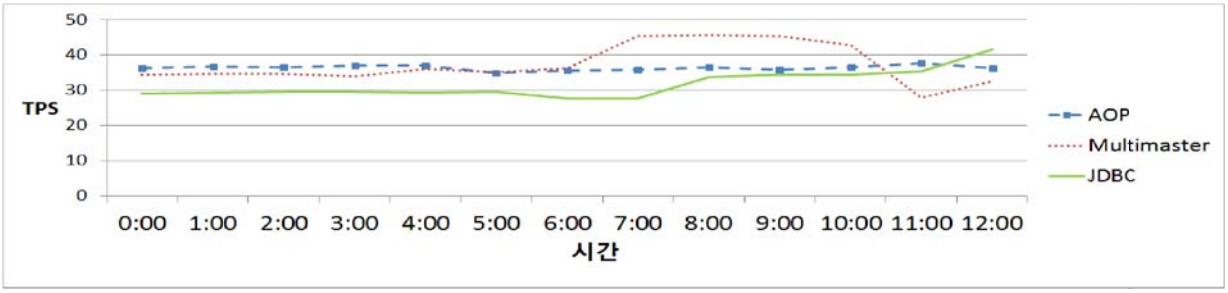
평가 항목으로는 테스트는 안정성과 속도 그리고 정확한 라우팅 여부를 선정하였다. 평가 대상으로 <표 3>과 같이 제안 알고리즘인 AOP 기반 알고리즘과 AOP 기반 알고리즘을 테스트한 환경과 유사하면서도 멀티마스터를 구현한 MySQL 의 Galaera Cluster 를 선정 하였다. 또한 Django 에서 제공하는 데이터베이스 라우팅 기능과 유사한 어플리케이션의 수정이 필요한 MySQL 의 JDBC 드라이버(Driver)를 이용한 트랜잭션 라우팅을 선정 하였다.

<표 3> 비교대상 특징 비교

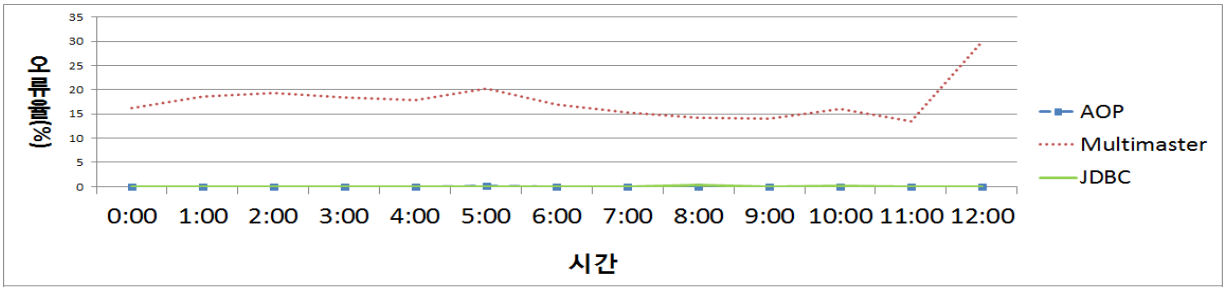
명칭	설명	코드수정
AOP	AOP 트랜잭션 라우팅 모듈을 사용한 마스터/슬레이브 트랜잭션 라우팅	X
Multimaster	MySQL 멀티마스터 DB Galera Cluster 구성을 통한 라운드 로빈 트랜잭션 라우팅	X
JDBC	JDBC Driver 를 통한 마스터/슬레이브 트랜잭션 라우팅	O

4.2 평가

이 논문에서 제안한 AOP 기반 트랜잭션 라우팅 알고리즘을 통해 12 시간동안 50 명의 가상 유저로 테스트 진행하여 AOP 트랜잭션 라우팅 방식과 JDBC 드라이버를 통한 라우팅 방식 모두 정확한 라우팅 결과를 도출 하였으며 평균 TPS(Transaction Per Second)의



(그림 3) 평균 TPS



(그림 4) 평균 오류율

경우 (그림 3)과 같은 결과가 도출 되었다.

TPS 가 높을수록 단위 시간에 더 많은 요청을 처리한 것이며 그래프의 기울기의 변화가 급할수록 안정적이지 못한 것을 의미한다. 연구에서 제안한 AOP 트랜잭션 라우팅 모듈이 적용된 시스템의 경우 12 시간 동안 그래프의 기울기 변화가 거의 없이 안정적으로 일정한 속도를 유지 하였으며 (그림 4)같이 오류율은 JDBC 와 더불어 발생하지 않아 세 가지 실험 중 가장 우수한 결과가 도출 되었다.

오류 종류는 <표 4> 와 같이 웹 페이지에서의 문서 변환 오류의 경우 세 가지 모두 비슷하게 나타났으며 이는 A 사의 서비스 변환 프로그램에 많은 부하가 발생 하였을 때 나오는 이슈로 실험 결과와 큰 관련이 없다. 데드락의 경우 멀티마스터 DB 에서 발생할 수 있는 전형적인 오류 형태로 여러 트랜잭션이 동시에 같은 데이터베이스의 행을 삽입/삭제/갱신할 경우 발생할 수 있는 오류이다. 이 실험 결과로 멀티마스터 DB 의 경우 동시에 데이터를 수정할 경우 매우 불안한 결과가 도출 되었다.

<표 4> 오류 분석 결과

명칭	문서 변환 오류	데드락
AOP	7	0
멀티마스터	7	3,599
JDBC	9	0

5. 결론 및 향후 연구

이 논문에서는 마스터/슬레이브 데이터베이스 모델에서 AOP 기반의 트랜잭션 라우팅 알고리즘을 제안하였다. 실험 결과에서 제안하는 알고리즘을 적용할 경우 안정성, 오류 발생율이 전체적으로 가장 적은 것이 확인 되었다. 세부적으로는 멀티 마스터 형태를 사용한 경우 보다는 안정성이 더 뛰어난 것이 확인되었으며 JDBC 드라이버를 통한 방법과의 비교에서는 속도 및 안정성은 유사하게 측정 되었으나 사용자의 트랜잭션에 대한 어플리케이션의 코드 수정을 필요로 하게 하였다. 위와 같은 결과로 이 논문에서 제안하는 방법을 사용하게 되면 안정적이며 성능이 뛰어나면서도 어플리케이션을 수정하지 않게 되는 결론이 도출 되었다. 향후 연구로는 이름 기반이 아닌 다른 형태로 마스터/슬레이브를 선정할 수 있는 기준을 마련하는 것과 슬레이브 선택후 슬레이브 그룹 내의 특정 데이터베이스를 선택하는 것에 대한 연구가 요구된다.

참고문헌

[1] E. Cecchet, G. Candea and A. Ailamaki, “Middleware-based Database Replication: The Gaps Between Theory and Practice,” In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 739-752, 2008

[2] K. K. Hercule, M. M. Eugine, B. B. Paulin and L. B. Joel, "Study of the Master/Slave replication in a distributed database,” International Journal of Computer Science Issues, vol. 8, no. 5, pp. 319-326, 2011.

[3] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm and W. G. Griswold, "An overview of AspectJ,” ECOOP 2001-Object-Oriented Programming. Springer Berlin Heidelberg, pp. 327-354, 2001.

[4] S. Gançarski, H. Naacke, E. Pacitti and P. Valduriez. "The Leganet system: Freshness-Aware Transaction Routing in a Database Cluster,” Information Systems, vol. 32, no. 2, pp. 320-343, 2005.

[5] S. Elnikety, S. Dropsho and W. Zwaenepoel, “Tashkent+: Memory-aware load balancing and update filtering in replicated databases,” ACM SIGOPS Operating Systems Review, vol. 41, no. 3, pp. 399-412, 2007.

[6] Django, “a high-level Python Web framework that encourages rapid development and clean, pragmatic design,“ https://www.djangoproject.com