KIPS Tr. Software and Data Eng.
Vol.3, No.11 pp.471~478 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2014.3.11.471

Database Transaction Routing Algorithm Using AOP

Kang Hyun Sik† · Sukhoon Lee†† · Doo-Kwon Baik†††

ABSTRACT

Database replication is utilized to increase credibility, availability and prevent overload of distributed databases. Two models currently exist for replication - Master/Slave and Multi-Master. Since the Multi-Master model has problems of increasing complexity and costs to interface among multiple databases for updates and inserts, the Master/Slave model is more appropriate when frequent data inserts and updates are required. However, Master/Slave model also has a problem of not having exact criteria when systems choose to connect between Master and Slave for transactions. Therefore, this research suggests a routing algorithm based on AOP (Aspect Oriented Programming) in the Master/Slave database model. The algorithm classifies applications as cross-cutting concerns based on AOP, modularizes each concern, and routes transactions among Master and Slave databases. This paper evaluates stability and performance of the suggested algorithm through integration tests based on scenarios.

Keywords : AOP, Transaction, Routing, Transaction Routing, Database

AOP를 사용한 데이터베이스 트랜잭션 라우팅 알고리즘

강 현 식† · 이 석 훈†† · 백 두 권†††

요 약

데이터베이스 복제(Replication)는 분산 데이터베이스 환경에서 신뢰성, 가용성, 과부하 방지 등을 위하여 이용되며, 마스터/슬레이브(Master/Slave), 멀티마스터(Multi-Master)와 같이 두 가지 모델이 존재한다. 멀티마스터 데이터베이스는 다중 데이터베이스에 삽입 및 갱신을 위한 동기화에 따른 복잡도 증가와 비용 증가와 같은 문제를 지닌다. 이러한 이유로 데이터의 삽입과 수정이 빈번히 일어나는 환경에는 마스터/슬레이브 모델을 이용한 데이터베이스 복제가 적합하다. 하지만 마스터/슬레이브 데이터베이스 역시 시스템에서 각 트랜잭션이 마스터로 접속해야 할지, 슬레이브로 접속해야 할지를 선택하기 위한 기준이 명확히 존재하지 않는 문제를 지닌다. 따라서 이 연구에서는 마스터/슬레이브 데이터베이스 모델에서 AOP(Aspect Oriented Programming) 기반의 데이터베이스 트랜잭션 라우팅 알고리즘을 제안한다. 이를 위하여 AOP에 기반하여 애플리케이션을 횡단 관심사로 분리하고 각 관심사들을 모듈화 하여 트랜잭션을 마스터 데이터베이스 및 슬레이브 데이터베이스로 라우팅한다. 이 논문은 시나리오 기반의 기능 통합 테스트를 통하여 제안 알고리즘의 안정성(Stability) 및 성능이 우수함을 평가한다.

키워드 : AOP, 트랜잭션, 라우팅, 트랜잭션 라우팅, 데이터베이스

1. 서 론

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 동시다발적으로 데이터의 삽입 및 갱신이 일어날 때 동기화 문제로 인하여 성능 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 선택하는 것에 대한 기준은 따로 존재하지 않는다. 이

※ 이 논문은 2014년도 한국정보처리학회 춘계학술발표대회에서 'AOP 기반의 트랜잭션 라우팅 알고리즘'의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 고려대학교 소프트웨어공학과 석사과정
†† 준 회 원 : 고려대학교 컴퓨터·전파통신공학과 박사과정
††† 종신회원 : 고려대학교 융합소프트웨어전문대학원 교수
Manuscript Received : June 30, 2014
First Revision : September 30, 2014
Accepted : October 1, 2014
* Corresponding Author : Doo-Kwon Baik(baikdk@korea.ac.kr)

경우에는 일반적으로 프론트엔드 애플리케이션이나 중간의 미들웨어 단계에서 트랜잭션 라우팅(Transaction Routing)이 수행된다.

이 논문에서는 마스터/슬레이브 데이터베이스 모델에서 AOP (Aspect Oriented Programming) 기반의 데이터베이스 트랜잭션 라우팅 알고리즘을 제안한다. AOP에 기반하여 마스터/슬레이브 데이터베이스에 트랜잭션 라우팅을 할 경우, 트랜잭션 라우팅에 대한 관심사를 모듈화 하여 각각의 애플리케이션에서 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. 제2절에서는 관련 연구를 기술하고, 제3절에서는 제안하는 웹 애플리케이션의 구조와 제안 알고리즘을 기술하며, 제4절에서는 이를 통한 실험 결과를 평가 및 검증하며, 제5절에서는 결론 및 향후 연구에 대해 제시한다.

2. 관련 연구

2.1 관점 지향 프로그래밍(AOP)

AOP는 소프트웨어를 핵심 관심사와 횡단 관심사로 분리하여 모듈화 시키는 프로그래밍 기법이다[3]. 기존의 객체 지향 프로그래밍은 다수의 객체들에 분산적으로 중복되어있는 공통 관심사가 발생하게 되어있어 프로그래밍 유지보수 및 가독성이 낮다는 단점을 지닌다. AOP는 공통 관심사를 횡단 관심사라고 정의하여 모듈 형태로 분리하여 핵심 관심사에 직조(Weaving)하는 형태로 프로그래밍 하는 기법이다.

Fig. 1은 로그인, 업로드, 댓글 목록과 같은 핵심 관심사 예와 핵심 관심사에 중복으로 들어있는 공통 관심사인 횡단 관심사를 로깅, 보안, 트랜잭션 등의 예로 분류하였다.

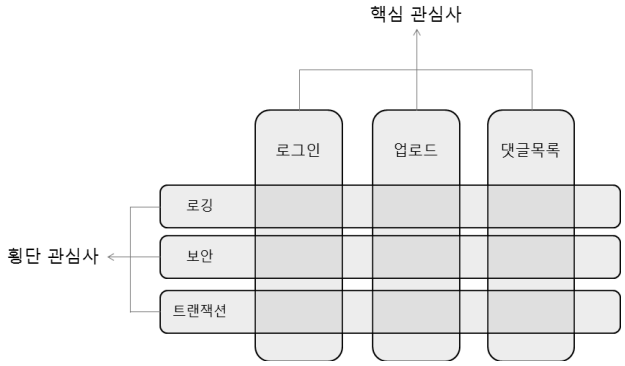

Fig. 1. Example for Core Concerns and Cross Concerns

AspectJ에서는 아래와 같은 AOP에 대한 명세를 정의하였다.

a) 관점(Aspect): 관점은 구현하고자 하는 횡단 관심사의 기능을 의미한다. 이 논문에서의 관점은 트랜잭션 라우팅을 의미한다.
b) 결합점(Join Point): 결합점은 관점을 삽입하여 실행 가능한 애플리케이션의 특정 지점을 의미한다.
c) 포인트컷(Pointcut): 결합점의 집합으로 관점 표현식(expression)으로 표현된다.
d) 직조(Weaving): 직조는 관심을 대상 객체에 적용하여 기존의 객체와 다른 새로운 객체를 생성하는 것을 의미하며, 직조는 크게 두 가지 방식으로 수행될 수 있는데 컴파일 되는 도중에 객체의 소스를 변경하여 직조되는 컴파일 시간(Compile time), 직조될 대상이 실행되는 시점에 직조되는 실행 시간(Runtime)이 있다. AspectJ는 위의 두 가지 외에 추가로 JVM(Java Virtual Machine)에서 클래스가 로딩 될 때인 클래스 로딩 시간(Class Load Time)에 대상 클래스의 바이트코드(Byte Code)를 변경하여 수행한다.
e) 충고(Advice): 충고는 관점의 실제 구현체로 결합점에 삽입되어 동작하는 코드를 말한다. 충고는 포인트컷과 함께 동작하게 되는데 충고의 종류는 시점에 따라 Before, Around, After, After Returning, After Throwing 등으로 나뉜다. Before는 해당 결합점이 실행되기 직전에 직조되는 것을 의미한다. Around는 해당 결합점의 실행 직전에 직조되어 해당 메소드를 수행하거나 수행하지 않게 하는 것까지 책임을 진다. After는 해당 결합점의 실행 이후에 직조되는 것을 의미한다. After Returning은 해당 결합점의 실행 이후 리턴하는 시점에 직조되는 것을 의미한다. After Throwing은 해당 포인트컷의 실행 도중 예외(Exception)를 던질 경우 직조된다.

목표-시나리오 모델링 기반 횡단 관심사 식별 및 명세화 방법은 목표-시나리오 기반의 요구사항 분석 방법을 기반으로 횡단 관심사를 식별하는 방법이다[4]. 이 연구에서는 비즈니스 수준, 서비스 수준, 상호작용 수준, 내부 수준을 목표와 시나리오 추상수준의 4가지 항목으로 정의하여 시스템 전체 관심사를 분석하고 분석된 결과 중 중복을 식별하여 횡단 관심사를 분석한다. 하지만 위 연구에서는 전반적인 시스템 관심사에 대한 포괄적인 분석에 대한 방법론만을 제시한다.

2.2 분산 복제 데이터베이스

[1]은 미들웨어 기반 분산 데이터베이스 복제에 대한 이론과 실제에 대한 차이점에 대하여 연구한다. 또한, 다양한 사례를 들어 분산 데이터베이스에서의 복제에 관한 설명을 하며 분산 데이터베이스에서의 데이터 복제에 대해 설명한다.

[5]는 복제 방법을 사용한 분산 데이터베이스들의 관리에 대한 연구로 기본적인 마스터/슬레이브 방식들인 주기적인

마스터 데이터베이스의 데이터를 슬레이브로 복제하는 방식이나 DBMS 벤더에서 제공하는 테이블 추출 유틸리티를 통한 테이블 복제 방식 등을 설명한다.

그러나 이러한 연구들은 마스터/슬레이브를 설명할 때 어떤 마스터/슬레이브 데이터베이스를 선택해야 하는지에 대한 방법에 대해서는 기술하지 않는다.

2.3 로드 밸런싱

로드 밸런싱 알고리즘은 트래픽이 많을 때 각 데이터베이스로 그 처리를 분산시키기 위한 알고리즘으로 여러 가지 형태가 존재한다. 데이터베이스에 적용할 수 있는 기본적인 알고리즘으로는 데이터베이스별로 순차적으로 선택하는 형태인 Round Robin, 현재 접속자가 가장 적은 데이터베이스를 선택하는 Least Connection, 각 데이터베이스별 가중치를 주어 선택하는 Weighted Round Robin, 각 클라이언트의 원격 주소값을 통한 해시값을 생성하여 데이터베이스를 선택하는 Persistent Hash 방식, 응답 시간이 가장 빠른 데이터베이스를 선택하는 Best Response Time 등이 있다[6].

2.4 트랜잭션 라우팅

Leg@Net는 데이터베이스 클러스터 신선도인지(Freshness-Aware) 트랜잭션 라우팅 기법을 사용한다[7]. 이는 멀티마스터 데이터베이스에서 트랜잭션을 라우팅하여 데이터베이스에 복제하며, 데이터베이스의 일관성 및 성능을 유지하는 미들웨어로 동작한다.

Takshkent+는 복제된 데이터베이스에서의 메모리 인지 로드 밸런싱 및 업데이트 필터링 기법이다[8]. 이는 멀티마스터 데이터베이스에서 트랜잭션에 사용되는 작업량을 측정하여 부하가 가장 적은 데이터베이스에서 실행되게 한다. 또한 이러한 작업량에 대한 데이터를 지속적으로 누적한 후 업데이트하여 각 복제 대상 데이터베이스에 업데이트 전파의 효율을 높이게 한다.

하지만 이러한 트랜잭션 라우팅 연구들은 주로 멀티마스터 데이터베이스에서 어떤 마스터에 연결할 것인가에 대한 기법으로 마스터/슬레이브 데이터베이스 환경을 고려하지 않는다.

Django는 마스터/슬레이브 데이터베이스 환경에서 웹 애플리케이션의 개발을 위하여 파이썬 기반의 웹 프레임워크를 제공한다[9]. Django는 구체적인 마스터/슬레이브에 대한 라우팅 전략을 가지고 있지 않으므로, 애플리케이션을 개발할 때마다 각각의 트랜잭션 라우팅을 가능하게 코드를 수정해야 한다. Django는 Database Router라는 클래스를 제공하며 이 클래스에서는 db_for_read, db_for_write와 같이 데이터베이스에서 읽거나 쓰거나 하는 등의 트랜잭션이 일어날 때 처리하는 방안을 직접 코딩해야 한다. 그러나 이러한 방법은 복합 트랜잭션에 대해서는 대응할 수 없으며 동기화에 대한 문제가 발생할 수 있다.

3. AOP 기반의 트랜잭션 라우팅 알고리즘

3.1 제안 알고리즘을 적용한 애플리케이션

이 연구에서 제안하는 알고리즘을 적용한 애플리케이션 구조는 Fig. 2와 같다.

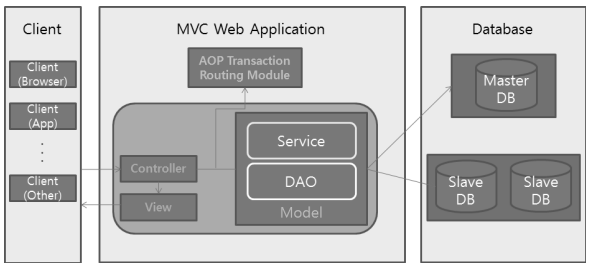

Fig. 2. Structure of The Application that The Proposed Algorithm

클라이언트(Client)는 웹 애플리케이션에 비즈니스 로직(Business Logic)을 요청하고 애플리케이션에서는 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 된다. 컨트롤러에서 서비스의 메소드로 진입하기 전에 AOP 트랜잭션 라우팅 모듈(AOP Transaction Routing Module)로 직조되어 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 수행하는 데 필요한 데이터베이스의 자료에 접근 및 수정을 한다. 이 과정에서 DAO에서는 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 애플리케이션뿐만 아니라 같은 모델을 사용하는 다른 애플리케이션이 존재할 경우 정상적으로 트랜잭션 라우팅이 불가능하게 된다.

3.2 제안 알고리즘

이 절에서는 연구에서 제안하는 트랜잭션 라우팅에 대한 설명과 라우팅 전략 및 예외사항 처리를 설명한다.

이 연구에서 제안하는 트랜잭션 라우팅 알고리즘은 AOP의 횡단 관심사로 분류가 되며 Fig. 3과 같이 트랜잭션 AOP 모듈보다 먼저 수행되어야 한다.

Table 1은 위에서 정의된 개념을 통해 이 논문에서 제안하는 트랜잭션 라우팅의 AOP 명세를 나타낸다. Table 1에서 충고가 Around이기 때문에 서비스의 메소드가 실행되기 전에 직조되어 마스터/슬레이브 데이터베이스를 선택을 하

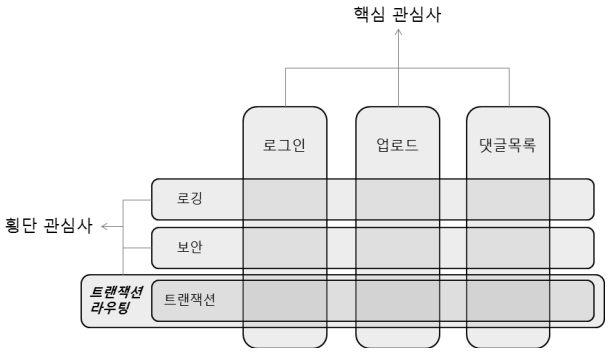

Fig. 3. Transaction Routing Cross Concern

Table 1. AOP Transaction Algorithm AOP Description

AOP Component	Description
Aspect	Transaction Routing
Join Point	Before Service Method Execution
Pointcut	execution(* package.name..*Service*.*(..))
Advice	Around

게 된 후 원래 실행될 예정이었던 서비스 메소드를 실행하게 된다. 제안하는 알고리즘은 Table 2와 같다.

Table 2. AOP Transaction Routing Algorithm

```
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

이 논문에서 제안하는 알고리즘의 목표는 마스터/슬레이브 데이터베이스를 선택하는 것으로 알고리즘의 목적 자체만으로는 Before 충고를 사용하는 것이 간편할 수 있으나 Around 충고를 사용한 이유는 각 트랜잭션이 종료되고 나면 새롭게 트랜잭션을 라우팅 해야 하기 때문이다. 즉, 위 알고리즘에서 명세한 D값을 초기화하는 작업이 직조된 메소드를 수행하고 난 다음에 반드시 일어나야 하기 때문이다. 제안하는 알고리즘은 메소드의 이름을 기반으로 라우팅한다. 슬레이브 데이터베이스만 사용해야 할 경우는 트랜잭션에서 읽기 전용에 해당하는 트랜잭션만 포함하며 해당 메소드들은 보통의 애플리케이션에서 get, find와 같은 형태의 이름을 띄게 된다. 그러나 그러한 메소드의 경우라 할지라도 마스터 데이터베이스를 사용해야 할 경우가 존재한다. 다음과 같은 상황에서는 get, find와 같은 형태의 메소드일지라도 마스터 데이터베이스를 선택할 수 있게 하였다.

a) 상황 1: 대상 메소드에 @RoutingDataSource 어노테이션이 존재하고 어노테이션의 value가 DataSourceType.MASTER 값을 가질 경우
b) 상황 2: 대상 메소드가 호출되기 전에 이미 다른 서비스 레이어에서 마스터 DB가 선택된 경우

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

3.3 AOP 우선순위

AOP는 각 AOP 모듈 간의 우선순위가 매우 중요하다. AOP에서는 이러한 것을 관점 우선순위(Aspect Precedence)라고 정의한다[3]. 트랜잭션 처리는 대표적인 횡단 관심사 중 하나이며 트랜잭션이 적용될 때에는 데이터베이스 연결 작업이 선행된다. 그렇기 때문에 애플리케이션에서는 AOP 모듈로 트랜잭션을 적용할 경우 트랜잭션이 수행되기 전에 이 논문에서 제안한 알고리즘이 수행되어야 한다. Table 3은 AspectJ로 구현한 트랜잭션 및 트랜잭션 라우팅에 대한 관점 우선순위의 예를 나타낸다.

Table 3. Transaction, Transaction Routing Aspect Precedence

```
public aspect TransactionRoutingAspectOrdering
{
    declare precedence : TransactionRouting, Transactional;
}
```

Table 3에서 TransactionRouting은 이 연구에서 제안한 AOP 트랜잭션 라우팅 모듈을 의미하며 Transactional은 트랜잭션을 적용하는 트랜잭션 AOP 모듈을 의미한다.

4. 실험 평가

4.1 실험 환경 및 방법

이 연구에서는 자바의 AspectJ AOP를 사용한다. 실험은 이미 구현된 A사의 문서 편집 클라우드 스토리지 서비스의 테스트베드 환경으로 구축한다. 적용 서비스는 RESTful API(Application Programming Interface)를 통한 테스트의 자동화가 가능하다. 구현은 JDK 1.7.0.40, Spring Framework 4.0.0, Hibernate ORM Framework 4.3.1를 이용한다. 마스터/슬레이브 데이터베이스는 MySQL 5.5.31을 사용하고, 멀티마스터의 경우 동일 MySQL 버전과 스토리지 엔진은 InnoDB를 사용한다. 멀티마스터 구성을 위해 Galera Cluster를 사용하였다. WAS(Web Application Server)의 경우 Apache Tomcat 7.0.39 버전을 사용한다. 테스트 클라이언트로 NHN의 nGrinder의 사전에 정해진 트랜잭션 시나리오를 Python 스크립트로 구현한다.

실험을 위하여 가상의 50명 유저가 12시간 동안 서비스를 사용한다는 가정으로 테스트한다. 50명의 유저는 테스트베드 환경에서 순수하게 트랜잭션 라우팅 성능을 확인할 수 있을 정도의 유저 수/부하이며 그 이상이 될 경우 데이터베이스 성능 한계로 인해 제대로 된 결과를 도출할 수 없다. 평가항목으로는 안정성과 속도 그리고 정확한 라우팅 여부를 선정한다. 평가대상으로 Table 4와 같이 제안 알고리즘인 AOP 기반 알고리즘과 AOP 기반 알고리즘을 테스트한 환경과 유사하면서도 멀티마스터를 구현한 MySQL의 Galaera Cluster를 선정한다. 또한 Django에서 제공하는 데이터베이스 라우팅 기능과 유사한 애플리케이션의 수정이 필요한 MySQL의 JDBC 드라이버(Driver)를 이용한 트랜잭션 라우팅을 선정한다.

Table 4. Features of Web Servers

Name	Description	Code Modification
AOP	Proposed AOP Module Transaction Routing	X
Multimaster	MySQL Multi-Master DB Galera Cluster Round Robin Transaction Routing	X
JDBC	JDBC Driver Master/Slave Transaction Routing	O

실험은 비교대상들로 구축된 웹 서버에서 주어진 12개의 시나리오를 기준으로 nGrinder를 통해 12시간씩 진행된다. 테스트의 평가항목으로 12시간 동안 지속적인 테스트를 통해서 안정성과 속도 그리고 정확한 라우팅 여부를 평가항목으로 진행하였다.

Table 5의 각 시나리오는 A사의 서비스의 통계 자료를 기반으로 사용자들의 전체 리퀘스트 숫자별 실제 사용 빈도를 측정하여 작성된다. 각 시나리오에서 데이터베이스 항목은 해당 트랜잭션 안에서 동일한 테이블의 동일한 행에서 INSERT/UPDATE/DELETE와 SELECT가 함께 일어나게 되면 마스터로 설정이 되어야 함을 의미하며, 단순하게 SELECT만 일어난다면 슬레이브로 설정되어야 함을 의미한다.

4.2 평가

이 절에서는 성능 및 안정성을 평가한다. 이를 위하여 이 논문에서 제안한 AOP 기반 트랜잭션 라우팅 알고리즘을 적용한 시스템과 대조군으로 멀티마스터, JDBC를 통해 12

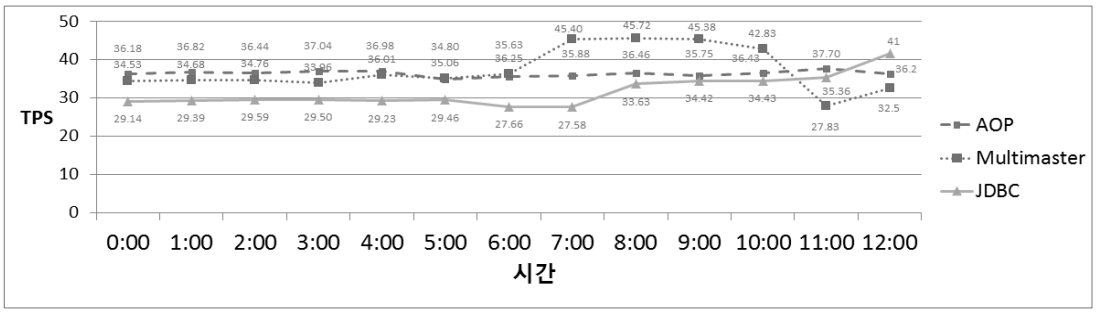

Fig. 4. Average TPS

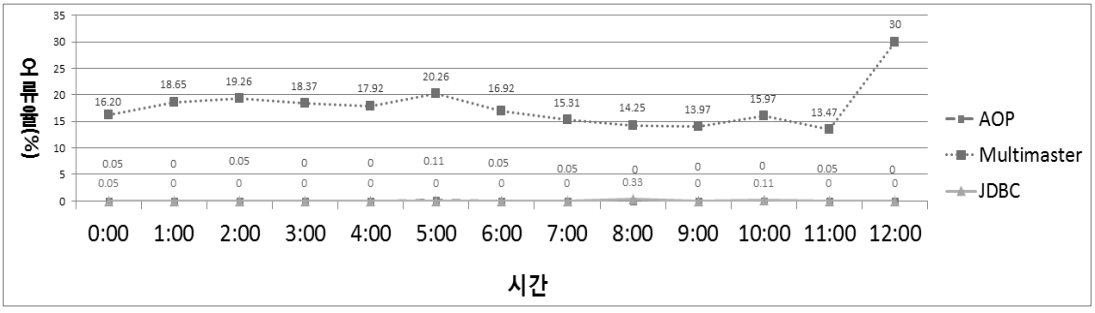

Fig. 5. Average Error Rate

Table 5. Scenario Description

Transaction Subject	Type	Frequency	DB
Login	Complex	5.65	Master
Sync File Add	Complex	33.98	Master
Download File	Complex	26.83	Master
1 Page Convert	SELECT	2.04	Slave
1 Page Convert Page View	SELECT	26.83	Slave
Create Share Link	Complex	0.78	Master
Sync Update	Complex	0.36	Master
Sync Directory Add	Complex	0.70	Master
Sync Move	Complex	0.71	Master
Sync Rename	Complex	0.71	Master
Sync File Copy	Complex	0.71	Master
Search File Name	SELECT	0.70	Slave

시간 동안 50명의 가상 유저로 테스트를 진행하여 결과를 도출한다. Fig. 4는 실험으로 나온 결과로서 평균 TPS (Transaction Per Second)를 보인다. Fig. 4에서 알 수 있는 결과와 같이 AOP 트랜잭션 라우팅 방식과 JDBC 드라이버를 통한 라우팅 방식 모두 정확한 라우팅 결과를 도출하며, 멀티마스터의 경우 다량의 데드락 오류가 발생한다.

TPS는 하나의 트랜잭션이 처리되는 데 걸리는 시간으로 이 연구에서는 하나의 시나리오를 트랜잭션 단위로 가정하여 시나리오 하나가 수행되는 시작부터 종료까지 걸린 시간을 나타낸다. TPS가 높을수록 단위 시간에 더 많은 요청을 처리한 것이며 그래프의 기울기의 변화가 급할수록 안정적이지 못한 것을 의미한다. 즉, Fig. 4에서 보이는 것과 같이, 제안 알고리즘이 적용된 시스템은 12시간 동안 그래프의 기울기 변화가 거의 없이 안정적으로 일정한 속도를 유지한다.

Fig. 5는 각 시스템의 평균 오류율을 보인다. AOP 기반 알고리즘과 JDBC는 데이터베이스와 관련된 오류는 거의 발생하지 않지만 멀티마스터의 경우는 평균 17.73%에 따르는 높은 오류율을 보인다. 또한 Table 6은 에러에 대한 결과로 각 시스템에서 문서 변환 오류와 그에 따른 데드락이 발생한 결과를 보인다. 웹 페이지에서의 문서 변환 오류의 경우

Table 6. Error Result

Name	Document Convert Error	Deadlock
AOP	7	0
Multimaster	7	3,599
JDBC	9	0

세 시스템 모두 비슷한 결과가 나타났으며 이는 A사의 서비스의 별개 변환 프로그램에 많은 부하가 발생하였을 때 타임아웃으로 인한 오류로 데이터베이스와 관련 없이 발생하는 이슈이다.

데드락의 경우 멀티마스터 데이터베이스에서 발생할 수 있는 전형적인 형태의 오류이다. 여러 트랜잭션이 동시에 같은 데이터베이스의 행을 하나의 트랜잭션이 완료되기 전에 삽입이나 삭제 혹은 갱신할 경우 오류가 발생할 수 있다. 멀티마스터에서는 다중 마스터 데이터베이스들과의 동기화 속도 문제로 서로 간 동기화가 전부 일어나지 않은 상태에서 삽입 및 삭제가 동시다발적으로 이루어질 경우 하나의 트랜잭션을 제외한 나머지 트랜잭션이 모두 실패하는 문제가 발생한다. 이 경우 데드락이 특정 시점에 몰리게 되며 이는 TPS의 감소와 오류율 증가로 이어진다. 이는 Fig. 4와 Fig. 5의 10:00~12:00 구간에서 멀티마스터의 결과로 잘 드러난다.

Table 7에서는 각 실험군별 평균 TPS와 구간별 변화량의 합을 통하여 구체적인 수치를 보인다. Table 7에서의 결과와 같이 TPS는 멀티마스터가 가장 높았으나 구간별 변화량의 합이 가장 커서 다른 시스템에 비해 불안정함을 알 수 있다.

Table 7. Average TPS & Variation Summation

Name	Average TPS	Variation Summation
AOP	36.34	9.70
Multimaster	37.30	37.23
JDBC	31.61	16.91

Fig. 6은 TPS의 각 시간 구간별 변화량을 절댓값으로 나타내었으며 값의 변화가 적은 것이 안정적이라는 의미로 해

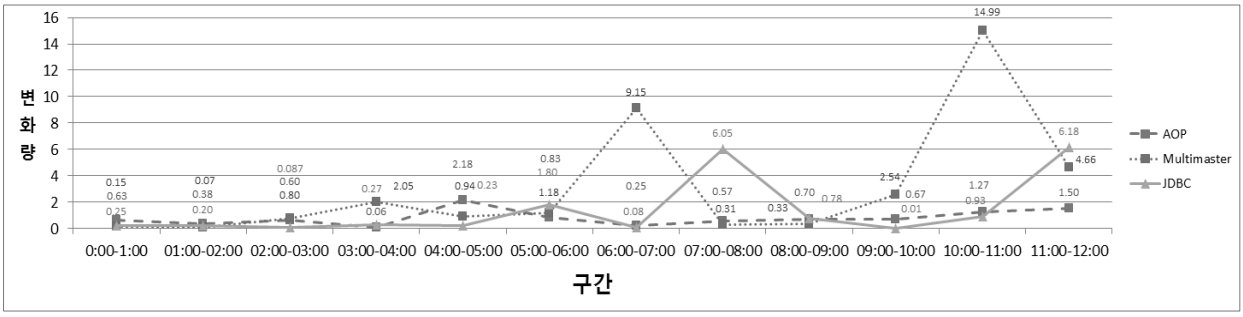

Fig. 6. TPS Variation by Time Range

석할 수 있다. 그래프 변화량이 급격하게 일어나는 멀티마스터나 소폭이지만 변화량 그래프가 유동적인 JDBC에 비해 AOP는 변화량이 거의 없다. 따라서 AOP가 멀티마스터나 JDBC에 비해 더 안정적이라고 볼 수 있다.

5. 결론 및 향후 연구

이 논문은 기존의 마스터/슬레이브 데이터베이스 모델이 지니고 있는 애플리케이션 코드 레벨에서의 수정이 필요한 의존성 문제점을 해결하기 위하여 AOP 기반의 트랜잭션 라우팅 알고리즘을 제안하였다. AOP 모듈을 이용하여 트랜잭션 라우팅을 이용하게 되면 애플리케이션 코드가 횡단 관심사로 분리되면서 기존의 코드상에서 트랜잭션을 라우팅할 필요가 없어지므로 가독성 및 유지보수성이 좋아진다.

실험에 따른 결과로 제안하는 알고리즘을 적용할 경우 안정성, 오류 발생률이 전체적으로 가장 적은 것이 확인되었다. 세부적으로는 멀티마스터보다는 안정성이 더 뛰어난 것이 확인되었다. 또한, JDBC 드라이버를 통한 방법과의 비교에서는 속도 및 안정성은 유사하게 측정되었으나, 사용자의 트랜잭션에 대하여 애플리케이션 코드를 수정하지 않는다는 점에서 우수함을 보였다. 따라서 이 논문에서 제안하는 AOP 기반의 트랜잭션 라우팅 알고리즘은 다른 방법들에 비해 애플리케이션의 수정이 없으며 결과 또한 안정적이며 우수한 성능을 보였다.

향후 연구로는 이름 기반이 아닌 다른 형태로 마스터/슬레이브를 선정할 수 있는 기준을 마련하는 것과 슬레이브 선택 후 슬레이브 그룹 내의 특정 데이터베이스를 선택하는 것에 대한 연구가 요구된다.

References

[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. H. Kim, M. S. Kim, and S. Y. Park, “An Identification and Specification Method of Crosscutting Concerns based on Goal-Scenario Modeling for Aspect-Oriented Software Development,” *Journal of Korea Information Science Society*, Vol.35, No.7, pp.424-430, 2008.

[5] M. Petrini, “Distributed Databases Management Using Replication Method,” The Annals of the University of Petrosani, *Economics*, Vol.9, No.4, pp.135-140, 2009.

[6] D. J. Choi, K. S. Chung, and J. S. Shon, “An Improvement on the Weighted Least-Connection Scheduling Algorithm in Web Cluster Systems,” *Korea Internet Service Promotion Association*, Vol.33, No.1, pp.199-201, 2006.

[7] 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.

[8] 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.

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

강 현 식

e-mail : cj848@korea.ac.kr

2011년 서경대학교 컴퓨터공학과(학사)

2012년~현 재 고려대학교 소프트웨어공학과 석사과정

관심분야 : 데이터베이스, 시맨틱웹

이 석 훈

e-mail : leha82@korea.ac.kr

2009년 고려대학교 전자 및 정보공학부 (학사)

2011년 고려대학교 컴퓨터·전파통신공학과 (공학석사)

2011년~현 재 고려대학교 컴퓨터·전파통신공학과 박사과정

관심분야 : 온톨로지, 데이터마이닝, 메타데이터 레지스트리, 자율컴퓨팅 등

백 두 권

e-mail : baikdk@korea.ac.kr

1974년 고려대학교 수학과(학사)
1977년 고려대학교 산업공학과(석사)
1983년 Wayne State Univ. 전산학과(석사)
1985년 Wayne State Univ. 전산학과(박사)
1986년～2013년 고려대학교 컴퓨터・전파통신공학과 교수
1989년～1991년 고려대학교 전산과학과 학과장
1990년～1991년 미국 Arizona대학교 객원 교수
1991년～2013년 ISO/IEC JTC1/SC32 전문위원회 위원장
1993년～1999년 한국과학기술원 객원책임연구원
1993년～1999년 한국DB진흥센터 표준연구위원
1996년～1997년 고려대학교 컴퓨터과학기술연구소(초대소장)
1997년～1998년 고려대학교 정보전산원 원장
1998년～1999년 한국정보과학회 전산교육연구회 운영위원장
1999년～2001년 정보통신진흥협회 데이터기술위원회 의장
2002년～2004년 고려대학교 정보통신대학(초대학장)
2002년～2003년 한국시뮬레이션학회 회장
2003년～현 재 정보통신부 컴퓨터프로그램보호위원회 위원
2004년～2005년 한국정보처리학회 부회장
2005년～2008년 한국소프트웨어진흥원 이사
2009년～2010년 고려대학교 정보통신대학 학장
2013년～현 재 고려대학교 융합소프트웨어전문대학원 교수
2013년～현 재 ISO/IEC JTC1/SC32 전문위원회 전문위원
관심분야 : 메타데이터, 소프트웨어공학, 데이터공학, 컴포넌트기반 시스템, 메타데이터 레지스트리, 프로젝트 매니지먼트 등