제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

DB Connection Pool 관리를 위한 성능 진단도구의 설계 및 구현

이재환, 정인환
한성대학교 컴퓨터신기술대학원
e-mail : leejh@onnuri.or.kr, ihjung@hansung.ac.kr

Design and Implementation of Performance Diagnosis Tool for DB Connection Pool Management

JaeHwan Lee , Inhwan Jung
Graduate School of Computer New-Technology, Hansung University

요 약

웹 어플리케이션 개발 시 데이터베이스 시스템의 사용이 증가함에 따라 데이터베이스 시스템에 접속하는 커넥션 리소스 관리에 대한 중요성이 부각되고 있다. 본 논문은 웹 어플리케이션 구축 시 사용하는 데이터베이스 접속 풀(Database connection Pool)의 성능을 평가하고 진단하는 도구를 제안한다. 본 도구는 성능 및 진단을 통하여 웹 어플리케이션에 가장 적합한 최적화된 DB 커넥션 풀을 최적화 하는 방법을 제시한다. 아울러 제안된 도구를 사용한 효과적인 데이터베이스 접속 풀(Database Connection Pool)의 관리 결과에 대하여 기술한다.

1. 서론

현대 사회가 정보화 되고 IT 산업이 발전하면서 사용하는 데이터의 양이 급격히 증대되고 다양화 되면서 기업체들은 데이터베이스를 연동한 웹/응용 어플리케이션 구축의 필요성을 느끼게 되었다. 어플리케이션 구축 시 DB 자원을 Access 하지 않는 것은 없을 것이다. 최근 들어 가장 많이 활용되는 JSP/SERVLET, EJB[1], 또는 CORBA[2] 와 같은 JAVA 기반의 어플리케이션들은 데이터베이스를 참조하기위해 JDBC(Java Database Connectivity)를 사용하게 된다[3].

JDBC 드라이버는 DB 에 커넥션을 맺을 때 사용되는데, 한번 맺은 커넥션은 반드시 닫아줘야 한다. 왜냐하면 동시에 맺을 수 있는 DB 커넥션의 수는 항상 DB 에서 제한을 받는데, 미리 커넥션을 맺어놓는 구조는 자원의 낭비 뿐만 아니라 요청한 여러 클라이언트들이 동일한 커넥션을 사용함으로써 발생되는 데이터베이스 잠금(DB Locking)과 거래(Transaction)의 중첩 현상이 일어날 수 있기 때문이다. 이러한 이유로 DB 커넥션을 관리하는 DB 커넥션 Pool 관리기법이 중요하게 대두되고 있다[4].

DB 커넥션 Pool 이란 한번 맺은 DB 커넥션을 바로 끊지 않고 DB 커넥션 Pool 에 저장해 놓고 다음에 동일한 요청을 해오면 바로 Pool 에서 저장해둔 커넥션을 꺼내서 줌으로써 빠른 DB 커넥션 Time 을 보장해 주는 것이다[5].

대부분의 상용 응용서버(Application Server) 제품들은 대부분 DB 커넥션 Pool 을 제공하고 있다. 중소업체나 소규모업체에서는 비싼 응용서버(Application Server)를 구입하지 못하므로 DB 커넥션 Pool 을 자체 제작하는 경우나 무료로 제공되는 DB 커넥션 Pool 소프트웨어를 사용하는 사례가 많다. 그러나, 이러한 경우 성능 및 안정성이 보장되지 않는다. 따라서, 이러한 DB 커넥션 Pool 을 효과적으로 관리하기 위한 성능 평가와 검증을 위한 별도의 진단 도구가 요구된다.

본 논문에서는 DB 커넥션 Pool 의 성능평가와 분석을 위한 진단도구를 설계하고 구현한다. 이 진단도구는 편리한 사용자 인터페이스(User Interface)를 제공하며, 다양한 조건으로 부하를 발생 함으로서 DB 커넥션 Pool 의 성능 진단 및 최적화를 가능하게 해준다.

본 논문의 구성은 다음과 같다. 2 장 관련연구에서

는 일반적으로 사용하는 커넥션 모델에 대하여 설명한다. 3 장에서는 설계 및 구현에 대하여 설명한다. 4 장에서는 실험 및 평가에 대하여 설명한다. 마지막으로 5 장에서는 결론 및 향후 연구에 대하여 기술한다.

2. 관련연구

DB 커넥션 Pool 에 대한 최적화 기법은 개발자들에 의해 지속적으로 연구 및 개발되어 오고 있다. 그러나, DB 커넥션 Pool 이 어떤 프로그램에서도 최적화된 성능을 내지는 못한다. 무료 DB 커넥션 Pool 관리 프로그램인 Poolman[6] 경우도 좋은 기능들이 있지만 모든 프로그램에 최적화된 것은 아니다. 즉, 사용하기 전에 해당 프로그램에 사용이 적합한지 먼저 성능 테스트와 기능 분석을 해야한다. 이러한 문제점을 보완하고 최적화하기 위해 본 논문에서는 실제 웹사이트 개발 시에 사용한 DB 커넥션 Pool 을 대상으로 다양한 형태의 커넥션조건으로 DB 커넥션을 발생시켜 DB 커넥션 Pool 의 성능과 문제점을 파악하여 응용 프로그램(Application Program)이 최적화된 성능을 보일 수 있게 할 수 있는 도구를 제공하고자 한다.

다음에 제시한 DB 커넥션 Pool 은 일반적으로 중소규모의 자바 응용 프로그램(Java Application Program)에서 사용하는 DB 커넥션 Pool 의 모델이다.

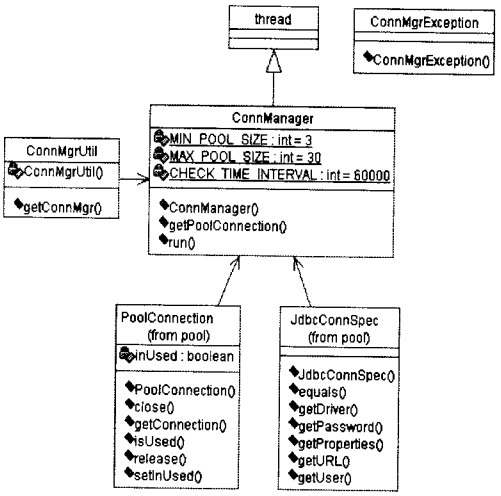

그림 1. DB Connection Pool Model

이 모델에서는 ConnManager 클래스가 스레드(Thread)로 돌면서 DB 커넥션 Pool 을 관리하고 있으며, 초기 커넥션은 3 개인데 이것은 Pool 에 항상 남아 있어야 할 여유분의 커넥션 수이다. 즉, 동시에 여러 Client 가 Pool 커넥션을 요청하면 물리적인 DB 커넥션의 수가 올라간다. 그러나,Clinet 들이 반환을 하면 대기상태로 되고 다음 요청시에 재사용할 수 있다. 이것은 파라미터로 조정할 수 있다.

동시에 가져갈 수 있는 최대 커넥션 수는 30 개이며, 이것은 여러 클라이언트들이 커넥션을 가져갔으나 반환하지 않았고, 이미 최대치에 도달해서 더 이상 커넥션이 없을 때 더 이상 커넥션을 맺지 않고 예외(ConnMgrException)를 발생시킨다.

ConnManager 가 Pool 을 체크하는 주기는(Check Time)은 1 분으로 설정해 놓았다. 즉, 가용한 커넥션 수를 최소 개수로 낮춰주고, 사용하지 않는(반환되지 않은) 커넥션은 강제로 close() 시킨다.

그림 2 는 실제 커넥션을 사용하는 코드이다. 이러한 DB 커넥션 Pool 의 구현 및 사용 시 커넥션 자원(Connection Resource)이 반환이 제대로 되지 않거나, 설계의 잘못으로 인해 서버가 다운되는 현상이 초래되는 경우가 종종 있다. 즉, DB 커넥션 Pool 사용 시 커넥션을 맺고 사용 후 반드시 close()나 release()로 Pool 에 반환이 되어야 하는데 반환을 하지 않을 때나, 커넥션 타임을 너무 길게 설정해 다른 접속요구자가 커넥션을 얻지 못하는 등의 문제로 심각한 결과를 야기 시킨다.

```
Connection 의 max 값을 담는 변수의 초기값을 설정;
Properties 클래스 변수선언 및 객체를 생성;
DB Connection 을 위한 driver, url, user, password 를 설정;

JDBC Connection Spec 을  담는 변수선언 및 객체를 생성;

Connection Pool 클래스 배열의 변수선언;

try {
  Connection Manager 클래스 변수선언 및 객체생성;
  Connection Pool 클래스 배열의 변수에 Connection 의 max 값만큼
  Connection Pool 객체를 생성;

      for ( 0 부터 Connection 의 max-1 값까지 1 씩증가) {
          해당 DB Spec 정보를 통해 Connection 을 하나씩 얻어
          Connection Pool 클래스 배열에 담음;
      }
} catch (Exception e) {
      에러시 에러정보 표시;
} finally {
     for ( 0 부터 Connection 이 null 이 아니고 Connection 의
          max-1 값까지 1 씩증가){
            Connection Pool 을 release 함;
      }
}
```

그림 2. DB Connection Pool 사용 예

본 논문에서 구현된 진단 도구에서는 이 모델을 이용하여 임의적으로 커넥션수, 지연시간, 동시접속수 등을 포함한 여러가지 조건들을 입력하여 커넥션을 발생시키게 하였다.

3. 설계 및 구현

본 연구에서 구현한 진단 도구는 커넥션 수, 커넥션 타임, 동시 접속수, 지연시간, 미반환 커넥션수, 커넥션 인터벌 등 다양한 조건으로 커넥션을 발생 시킬 수 있고 실시간으로 커넥션풀의 상황을 모니터링 할

수 있도록 고려하였다.

진단 도구를 설계하면서 고려된 사항은 다음과 같다.

- ◆ 커넥션 Pool 설정 : 동적으로 커넥션 Pool Property 를 설정할 수 있다.
- ◆ 실시간 모니터링 : 커넥션 발생시 현재 초기 생성한 커넥션 수, 현재 사용가능한 커넥션 수, 현재 사용중인 커넥션 수를 표시 할 수 있도록 했다.
- ◆ JDBC 드라이버 설정 : 여기선 오라클사에서 제공하는 JDBC 씬(thin)드라이버를 사용하였다.

본 연구에서 구현한 진단 도구의 부하 조건에 따른 커넥션을 발생시키는 알고리즘이다.

```
//커넥션 풀을 초기화한다.
pool = new ConnPool(TrafficData.getInit 커넥션());

//접속 개수만큼 루프를 돌리며 커넥션을 생성한다.
for( int count=0 ;
        count < TrafficData.getMax 커넥션() ; count++){

   // 미반환할 커넥션수
   int losecount = 0;
   //커넥션 풀에서 커넥션을 얻는다.
   pool.get 커넥션();

   try{
    // 접속 간격
      Thread.sleep(TrafficData.getWaitTime());
      }
      catch(InteeruptedException e){
          System.err.println(e.toString());
      }

  // 미반환 접속이 없으면 풀에 커넥션을 반환
   if(TrafficData.getLose 커넥션() == 0){
      pool.release();

  // 미반환 접속이 있으면 반환수가 될때까지
  // 카운트를 증가하며 커넥션을 반환하지 않는다.
    }else if(losecount <=  TrafficData.getLose 커넥션()){
          if(((int)Math.random()*TrafficData.getInit   커넥션())
              == count){
                  losecount++;
              }
          }
      }
```

그림 3. 부하 발생 알고리즘

이 진단 도구는 Windows 환경하에서 Java 와 JDK1.4 API 를 스윙을 이용하여 구현되었다. 컴파일러와 디버거는 JDK 에서 제공하는 javac 와 jdb 를 이용하였다[7]. 그리고, 화면은 크게 데이터베이스 연결정보, 커넥션 조건항목, 모니터링 화면으로 구성되어있다.

그림 4 는 본 논문에서 구현한 진단 도구의 실행 화면을 보여주고 있다.

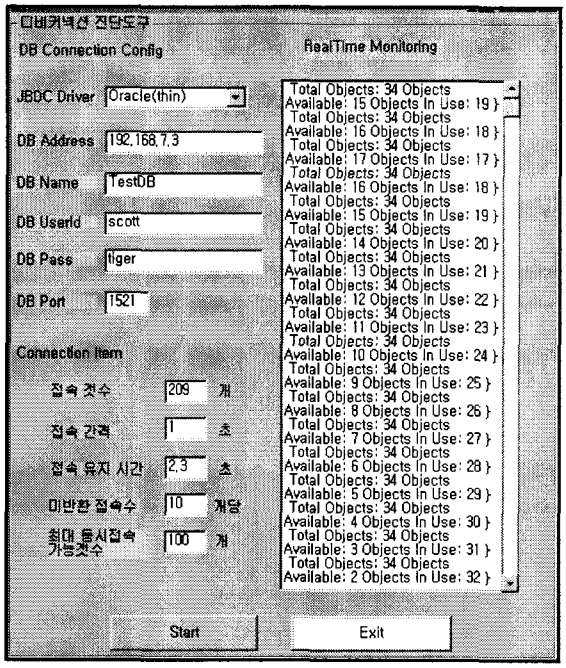

그림 4. 진단도구의 실행 화면

데이터베이스 정보와 접속(커넥션) 발생 항목 등 모든 조건을 입력한 뒤 Start 버튼을 누르면 접속(커넥션)이 발생하게 되며, 오른쪽 프레임에선 실시간으로 현재 DB 커넥션 Pool 의 상황을 모니터링 할 수 있다.

RealTime Monitoring 에서 Total Object 는 현재 DB 커넥션 Pool 에 만들어져 있는 커넥션 수이며, Object Available 는 현재 사용할 수 있는 대기중인 커넥션이고, Object In Use 는 현재 사용중인 커넥션 수이다. 이때 반환되지 않는 커넥션은 Object In Use 에 속한다.

4. 실험 및 평가

실험은 실제 특정 웹사이트 개발시에 사용한 DB Connention Pool 을 대상으로 하였다.

1 일평균 접속 수	209
최대 동시접속 수	21
평균 쿼리 응답속도	2.3 초
최대 커넥션 수	100
문제점	3-5 일마다 접속불능

이 웹사이트는 3-5 일에 한번 주기로 데이터베이스에 접속이 되질 않는 것으로 보아 커넥션이 누수가 되는 곳이 있는 것으로 의심되었었다.

그림 5, 6 은 본 논문에서 구현한 DB Connection Pool 도구를 이용하여 실제 DB Connection Pool 이 발생하는 과정을 보여준다.

다음 그림 5 는 정상적으로 커넥션을 DB 커넥션 Pool 에 반환했을 때 결과이다. 부하는 접속 간격 1 초, 평균 커넥션 타임 2.3 초, 동시 커넥션 수는 16 개로 하였다.

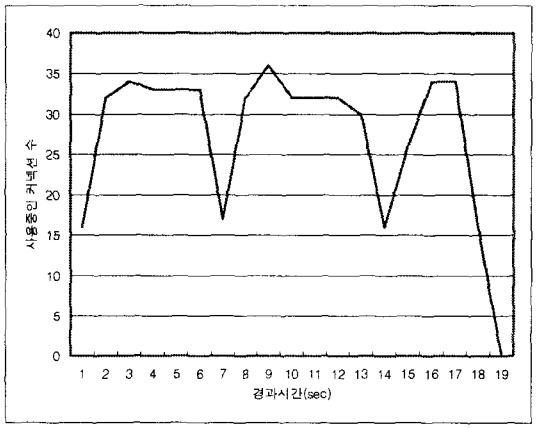

그림 5. 정상적인 커넥션 반환 결과

그림 5 에서 보듯이 커넥션이 증가 했다가 풀에 반환되는 원활한 커넥션 수급을 보여주고 있다.

다음은 그림 6 에서는 똑 같은 조건에 추가조건으로 접속 10 번 마다 하나씩 커넥션을 미 반환 하도록 부하를 주었다.
그림에서 보듯이 실험에서 사용중인 커넥션수의 수는 계속증가하고, 포화 상태일 때 다시 커넥션 요청이 오면 데이터베이스 에러가 발생하고 데이터베이스 접속이 되지 않는다. 이것은 소수의 커넥션이 계속 Pool 에 반환되지 않고 누적되어 포화상태에 이르렀기 때문이다.

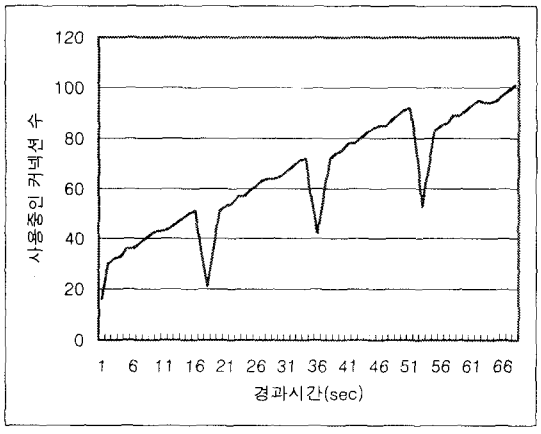

그림 6. 비정상적인 커넥션 반환 결과

위 결과를 토대로 근본적인 문제는 프로그램의 결함에 있으나, DB Connention Pool 을 최적화 함으로써 문제를 어느 정도 해소 할 수 있다. 이 실험에서 문제점으로 나온 Pool 에 반환되지 않는 커넥션을 주기적으로 체크 하는 모듈과, 커넥션수가 최대값에 도달 했을 시 새로운 요청자에게 대기시간(wait())주고, 다른 쓰지 않는 커넥션을 강제로 종료시켜 커넥션을 맺어주는 모듈을 추가하여 커넥션 수의 증가가 둔화되는 결과를 얻을 수 있었다.

그림 6 에서와 보여지는 것과 같이 본 논문에서 구현한 도구를 이용하여, 원하는 부하조건으로 커넥션 Pool 의 상태를 검사할 수 있다.

5. 결론 및 향후 연구

일반 중소규모업체에서 개발자가 자체적으로 DB 커넥션 Pool 을 개발해서 응용프로그램(Application Program)의 높은 성능을 기대하기는 쉽지 않으며, DB 커넥션 Pool 을 최적화하기 위해서는 많은 노력과 지식이 필요하다.

본 논문에서는 DB 커넥션 Pool 의 성능을 진단하여 최적화된 성능과 안정성을 기대할 수 있는 성능분석 도구를 설계하고 구현하였다.
실험을 통하여 효과적으로 DB Connection 의 부하 검증을 할 수 있는 것을 보여주었다.

본 논문에 이은 향후 연구로는, 다양한 조건의 DB Connection 부하가 생성될 수 있도록 부하 생성 알고리즘을 보완하고 실제 데이터베이스 프로젝트에 적용할 수 있도록 사용자 인터페이스를 보완하는 것이다.

참고문헌

[1] ED Roman, "Mastering Enterprise JavaBeans & the Java 2 Platform, Enterprise ED", Wiley & Sons, 1999.
[2] *Object Management Group, Realtime CORBA Joint Revised Submission, OMG Document orhos/99-02-12. ed. Mar. 1999*
[3] M. Sood, Examming JDBC Drivers, *Dr. Dobbs Journal*, pp.82-87, Jan 1998.
[4] D.S. Brahme, et, al, "Transaction – Based Verification Methology," Cadence Berkely Labs, Technical Report, Aug. 2000.
[5] "Java Development Framework Document - DB Connection Pool", http://www.javaservice.net, 1999.
[6] Poolman, http://sourceforge.net
[7] Mike phi, Using JDB in a multithreaded qui environment, http://forum.java.sun.com, 2003.
[8] *P. Oneil, Database Principles Programming Performance*, Morghan Kaufmann Publishers, 1994
[9] 볼랜드 자바팀,JbuilderStudyNet 공저, Jbuilder7, 가남사, 2003.