자바 쓰레드 풀을 이용한 웹 서버의 구현 및 성능 분석

전상현, 이광모, 엄상용,정연진,구태완
한림대학교 정보전자 공과대학 컴퓨터공학과

e-mail : shcheon@ns.ce.hallym.ac.kr

Implementation and Performance Analysis of Web Server Using Java ThreadPool

Sang-Hyun Jeon, Kwang-Mo Lee, Saang-Yong Uhmn,
Yeon-Jin Jung, Tae-Wan Gu
Dept. of Computer Engineering, Hallyum University

요 약

자바는 설계 때부터 네트워크 상의 운영을 고려하여 설계된 언어이기 때문에 소켓 바인딩과 같은 기능을 제공하고 있고, 또한 TCP/IP 프로토콜과 URL 처리 기능을 제공하고 있는 HTTP(HyperText Transfer Protocol)프로토콜을 동시에 처리할 수 있기 때문에 WWW 서비스를 연동하는 웹서버를 구축하는데 적합한 언어이다. 현재의 인터넷은 점차적으로 발전하여 수요는 급격히 증가하고 있으나, 많은 사용자로 인해 네트워크의 성능저하와 서버의 처리 능력 한계로 인하여 사용자의 수요를 충족시키지 못하고 있다. 본 논문에서는 인터넷을 이용한 네트워킹 프로그램에 있어 멀티쓰레드를 이용하여 응용프로그램을 작성하기 위한 쓰레드의 개요를 설명하고, 쓰레드를 활용한 풀을 구성하기 위한 제반 사항을 기술한다. 또한, 응용프로그램으로 동적 쓰레드, 서버 복제,쓰레드 풀 웹 서버를 구현하여 성능을 분석한다.

1. 서론

자바는 초창기부터 네트워크 처리를 효율적으로 제공하기 위해 만들어진 언어이다. Net 패키지는 IP 주소를 표현하기 위한 InetAddress 클래스와 URL 을 표현하기 위한 URL 클래스등이 제공되는 인터넷 주소 관련 API 와 클라이언트 소켓을 위한 Socket 클래스, 서버 소켓을 위한 ServerSocket 클래스등 다양한 소켓클래스가 제공되는 TCP/UDP/IP 관련 API 를 제공한다. 또한 추상 클래스 URLConnection 은 어플리케이션과 URL 사이의 통신을 위한 연결 고리를 나타내는 모든 클래스의 슈퍼 클래스로 URLConnection 의 인스턴스는 지정된 URL 을 통해 참조되는 객체로부터 읽거나 쓰는 용도로 사용된다.[1]

기존의 클라이언트-서버환경에서 하나의 서버에 많은 클라이언트의 요청이 있을 경우, 각 연결에 대한 프로세스의 생성으로 서버에 과부하가 발생하게 되고 요청에 대한 처리시간의 증대에 따른 많은 양의 정보를 처리하는데 어려움이 있다.[2] 이러한 문제를 해결하기 위해 자바의 쓰레드로 3 가지 모델을 구현하여 성능을 측정한다. 첫 번째로, 동적 쓰레드[6]를 이용한 웹 서버를 구축하고, 두 번째로, 서버복제를 통한 동적 쓰레드를 이용한 웹 서버를, 세 번째로 오브젝트 풀 모델을 응용한 오브젝트 풀 모델의 특수한 형태인 쓰레드 풀을 사용하여 서버를 구성하여 각각의 성능을 측정하고, 향후 성능향상을 위한 방법을 제시한다.[6]

본 논문의 구성은 다음과 같다. 2 장에서는 쓰레드에 대하여 살펴보고, 3 장은 동적 쓰레드와 서버 복제와 풀의 구조에 대해서, 4 장은 구현 핵심코드의 내용을, 5 장은 성능 분석을 설명하며, 끝으로 6 장은 결론 및 향후 연구방향을 제시한다.

2. 쓰레드의 개요

쓰레드라는 용어는 제어의 흐름(thread of control)을 의미한다. 제어의 흐름이란, 동일한 프로그램 내에서 다른 제어의 흐름과 독립적으로 실행될 수 있는 코드를 뜻한다.[4] 쓰레드는 프로세스에 비해 하나의 프로세스가 제고하는 문맥 정보를 공유하므로 문맥의 전환이 필요할 때 유지할 정보의 크기가 상대적으로 적으며, 보다 세분화된 연산에서 병행성을 활용할 수 있으므로 자원 활용 및 연산의 효율을 높이기에는 유리한 자원이다. 자바는 처음부터 멀티쓰레드를 염려하여 설계 되었으므로, 기본적으로 지원이 되며, 쉬운 환경을 제공한다. 멀티쓰레드를 사용하는 이유는 다음과 같다.[5]

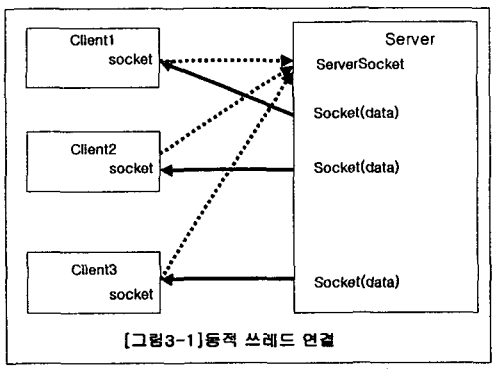
첫째, 사용자와의 향상된 상호작용. 단지 하나의 쓰레드만 사용 가능하다면, 프로그램은 한 번에 한 가지 일만 처리 할 수 있을 것이다. 하나의 프로세스를 가진 시스템에서는 운영체제가 여러 개의 일을 빠르게 스위칭하면서 동시적으로 처리하기 때문에 좀더 나은 사용자의 환경을 제시한다.

둘째, JavaVM 과 운영체제에 지원에 의하여 멀티프로세서 환경하에서 병렬처리가 가능하다.

3. 동적쓰레드와 서버복제 와 풀

3.1 동적쓰레드

동적쓰레드는 각 클라이언트의 연결시 서버의 각각 별도의 쓰레드에서 연결을 함으로써, 연산을 수행하는 방식이다. 그림 3-1 은 연결 상태를 보여준다.



[그림3-1]동적 쓰레드 연결

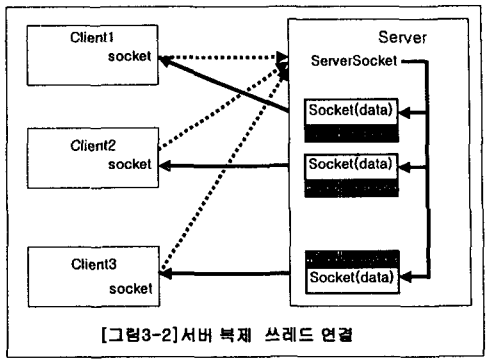
클라이언트 1 이 서버에 접속을 하면, 서버는 새로운 소켓을 만든 후, 1:1 대응형식으로 통신을 수행한다.

3.2 서버복제 쓰레드

서버 복제는 동적쓰레드와 비슷하다. 단지 각 클라이언트의 연결 요청이 있을 때마다 서버를 복제하고 연결을 한다는 것이다. 그림 3-2 는 그 구성을 보여 준다. 클라이언트는 가상적으로 각각의 서버에 접속하는 것처럼 인식되어진다.

3.3 쓰레드 풀

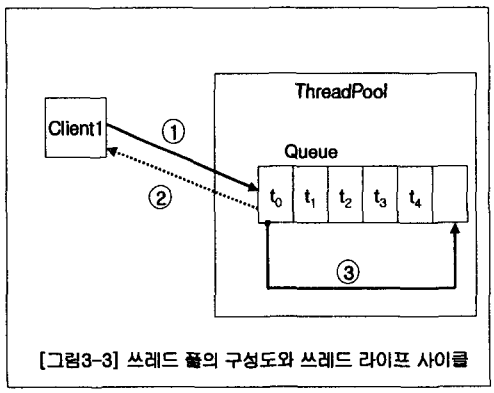
쓰레드 풀은 다수의 객체를 동시에 실행할 수 있는 제한된 개수의 쓰래드의 집합체를 의미한다. 쓰레드 풀은 특정 가상 머신에서 성능에 별 영향을 주지 않고, 프로그램에서 처리할 수 있는 쓰레드의



[그림3-2]서버 복제 쓰레드 연결

개수보다 더 많은 쓰레드를 요구할 때에 사용할 수 있다. 몇 개의 쓰레드를 미리 셋업하여 휴지 상태에 두고 수행해야 할 작업이 있을 때 휴지 쓰레드 중 하나로 수행 시키는 것이다.[4]

쓰레드 풀링은 가상 머신이 매번 적은 계산량을 요구하는 새로운 쓰레드를 만드는 시간을 절약할 수 있다. 추가적으로, 쓰레드의 시작과 가비지 컬렉션에 관계된 오버헤드를 최소화 할 수 있다. 쓰레드 풀에 만들어진 쓰레드는 계속적으로 다른 일에 할당 된다.



[그림3-3] 쓰레드 풀의 구성도와 쓰레드 라이프 사이클

그림 3-3 는 쓰레드 풀의 구성과 그 라이프 싸이클을 보여준다. ①은 클라이언트가 서버에게 연결을 요청한것이고, ②는 서버에서 T_0라고 명명되어진 쓰레드를 할당한 것을 보여준다. ③은 연결이 끝난 쓰레드가 큐의 맨 마지막으로 재진입 되어짐을 보여준다. 계산 집중적인 경우를 처리하는 서버는, 쓰레드가 생성되는 시간보다는 처리하는 시간에 더 많은 비중을 차지하므로, 처리율의 큰 향상은 기대하기 어렵다.[4] 이와 반대의 경우는 한 번 생성된 쓰레드가 계속적으로 재사용되어 짐으로서 계산 집중적인 서버의 경우 보다 높은 처리율을 보여준다.

4. 서버 구현

4.1 동적쓰레드 서버 구현

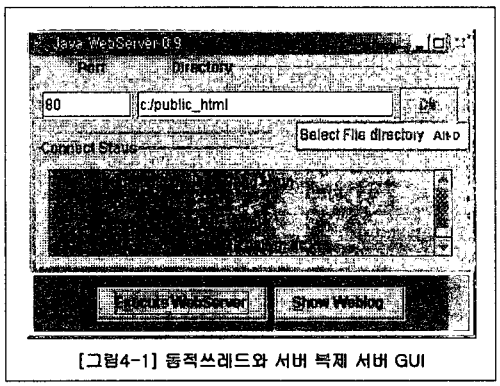
동적쓰레 서버의 구조는 일반적이 멀티쓰레드의 응용프로그램의 구조와 같다.

```
public WebService(String webRoot, int portNo)  {
```

```
  documentbase=new File(webRoot);
  try {
    ss = new ServerSocket(portNo);
    while(true) {
      WebService webService = new
          WebService(ss.accept());
      webService.start();
    }
  }
  catch(IOException e) {
  }
}
```

4.2 서버 복제 서버 구현

복제서버는 ServerSocket 을 만들고, 클라이언트로부터 연결 요청이 오면, 자신을 복제한 쓰레드를 생성한 후 새로운 쓰레드에 의하여 클라이언트로의 연결이 이루어진다. 그림 3-2 참조.



[그림4-1] 동적쓰레드와 서버 복제 서버 GUI

그림 4-1 은 웹 서버를 관리할 수 있는 GUI 를 보여주고 있다. 이 GUI 는 동적으로 홈을 관리할 수 있으며, Show Weblog 는 접속한 곳의 IP 와 시간을 알 수 있다.

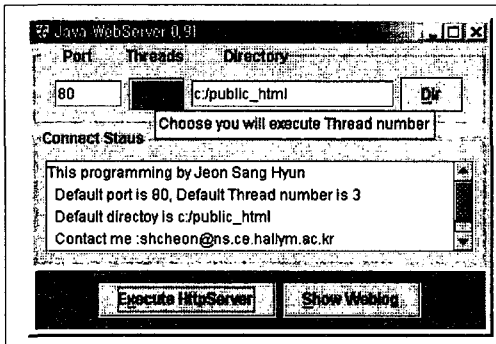
다음의 코드는 구현 부분 중 연결 부분을 보여준다.

```
public class WebServer
        implements Runnable, Cloneable {
  ServerSocket ss = null;
  Socket socket = null;
  Thread runner = null;
    .......
  public void run() {
    .....
  Socket clientSocket = ss.accpet();
  Webserver connectSocket = (WebServer) clone();
  connectSocket.server =null;
  connectSocket.data = clientSocket;
  connectSocket.runner = new Thread(connectSocket);
  connectSocket.runner.start();
  ....
}
```

4.3 쓰레드 풀 서버 구현

쓰레드 풀 서버의 GUI 는 그림 4-2 와 같으며, 동적 쓰레드 서버와 다른점은 풀 안에 유지할 쓰레드(workerThread)의 개수이다. 큐는 FIFO 구조이므로 사용하고 난 후의 쓰레드는 큐에 삽입되어 다음 연결을 기다린다. 서버의 우선순위를 workerThread 보다 낮춤으로서 현재 진행중인 요청보다는 새로 들어오는 연결을 처리하도록 하였고, 큐는 workerThread 의 개수만큼만 만들도록 하였다.

```
private WebService[] workerQueue;
private QueueObject qo;
```

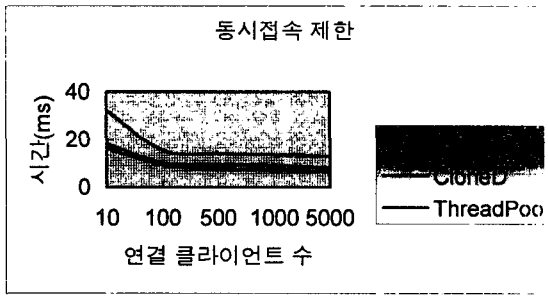


[그림4-2] 쓰레드풀 서버의 GUI

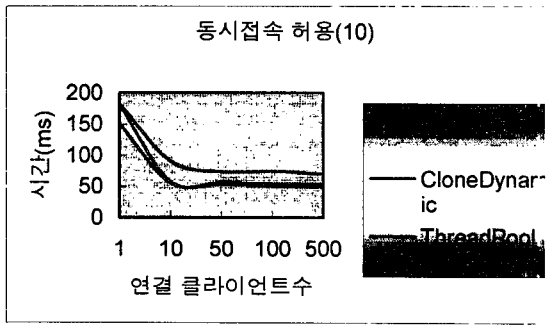
```
int workerPriority = serverPriority –1;
qo = new QueueObject(workersNumber);
workerQueue = new WebService[workersNumber];
```

5. 성능 분석

실험한 결과는 다음과 같다.



[표 1] 동시접속을 제한한 클라이언트의 연결.



[표 2] 동시접속을 허용(10 개의 클라이언트).

위의 표에서는 전체적인 성능면에서는 동적쓰레

드, 서버복제, 쓰레드 풀의 순서로 성능이 개선되어 짐을 보여준다. 그러나, 예상보다도 서버의 복제를 통한 서버의 구성의 성능이 가장 좋았다. [6]의 내용에서는 쓰레드 풀을 구성한 경우에 약 40%정도의 성능향상을 보인 것으로 나와있으나, 코드를 분석한 결과 저자의 쓰레드 풀은 본 논문에서의 동적쓰레드의 소스와 유사함을 발견했다. 그러므로, [6]의 결과와 이 결과는 비교를 할 수 없다.

6. 결론및 연구 방향

본 논문에서는 자바로 웹 서버를 구현하여 성능을 측정하였다. 쓰레드 풀 웹 서버는 동적쓰레드 웹서버보다 30%~50%의 성능이 향상되었고, 서버복제의 경우보다는 약 5%~10%의 성능이 향상되었다. 그러나, 동시접속을 허용하는 실험에서의 에러 율이 10 대의 클라이언트가 100 번의 접속을 시도한 결과 동적 웹서버는 평균 90%의 접속 율을 보였으나, 쓰레드 풀 웹서버는 80%의 접속 율을 보여주는 반면에 서버복제 서버는 안정성있는 결과를 나타내었다. 이는 서버가 workerThread 에게 작업을 전달하려고 했으나, 쓰레드 풀안에 쓰레드가 유휴 쓰레드의 부재로 인하여 발생하는 공백 상태이다.

추후 연구 방향은 동시접속을 수행하였을 경우, 다른 스케줄링 방식을 사용하여 접속율을 올리고, Thread.currentThread().slepp(time)을 이용한 적절한 유휴기간의 설정이 필요할 것 같다. 또한, 자바의 이기종간의 이식성을 고려하여 분산 서버를 구축하여 에러율을 높이는 연구가 계속되어야 할 것이다.

참고문헌

[1]이현우, 김형국, 홍성민, " Java Programming Bible" , 영진출판사, 1999

[2] 최성, 정상교, "JAVA 네트워킹프로그램 설계기법 및 3 차원 채팅프로그램 구현에 관한 연구", '97 한국정보과학회 추계 학술발표논문집(4), pp297~300,1997

[3] Elliotte Rusty Harold, Mike Loukides" Java Network Programming", O'reilly,1197

[4] Scott Oaks, Henry Wong,"Java Threads",O'reilly,1999

[5] Paul Hyde,"Java Thread Programming",SAMS,1999

[6] Reinhard Klemm,"Practical Guidelines for Boosting Java Server Performance",java99