산업기술연구(강원대학교 산업기술연구소 논문집), 제27권 B호, 2007.
Journal of Industrial Technology, Kangwon Natl. Univ., Korea, No. 27 B, 2007.

웹 서버 구현 방안들에 대한 성능 비교

A Performance Comparison of Web-Server Implementation Schemes

임 동 관* 선 주 호* 김 종 욱* 김 용 석**
Lim, Dong-gwan Seon, Ju-ho Kim, Jong-wook Kim, Yong-Seok

Abstract

For Web server implementations, there are 4 main schemes: process-per-request (PPR), thread-per-request (TPR), worker thread pool (WTP), and worker thread pool with buffers (WTPB). This paper compares performance of the schemes in response time point of view. WTPB shows the best performance. The appropriate number of worker threads for WTPB depends on the request service time. For short requests, the number can be very small. But for longer requests, it is about 1/6 of the number of simultaneous connections.

키워드 : *웹 서버, 작업자 스레드 풀, 성능 비교*
Keywords : *Web server, thread worker pool, performance comparison*

1. 서론

인터넷하면 웹서비스를 연상할 정도로 WWW(World Wide Web)은 인간이 정보를 찾는 주된 수단의 하나로서 자리 잡았다. WWW 서비스, 즉 웹 페이지를 제공하는 주체는 웹 서버인데, 이를 안정적이고 효율적으로 구성하고 운영하는 것은 기업이나 공공 기관들에게 있어서 매우 중요한 문제이다. 대부분의 기업 또는 공공기관들의 웹 서버 운영 목적은 고객들에게 서비스를 제공하는 것이다. 따라서 원활한 서비스 제공을 위해 웹 서버 관리자는 고객들의 요청에 의한 부하나 결함을 처리할 수 있어야 한다.

서버의 성능에 영향을 미치는 요소는 여러 가지가 있다. "얼마나 고성능의 하드웨어로 구성되어 있는가", "네트워크의 대역폭은 얼마나 확보 되어 있는가", "운영체제는 무엇인가", "서버 프로그램이 어떤 형태로 구현되는가", "디스크나 데이터베이스에 얼마나 많은 접근을 요구 하는가" 등을 생각해 볼 수 있다. 서버 프로그램의 구현 방법에는 여러 가지 방안들이 알려져 있다[1,2,3]. 본 논문에서는 서버 프로그램의 구현 방법에 초점을 맞추고, 여러 가지 구현방안들에 대하여 성능을 비교 평가하였다.

초기 웹 서버들은 사용자 요청을 처리하기 위해 새로운 프로세스를 생성하였다. 이후 웹 서버들은 프로세스에 비해 비교적 하드웨어 자원을 적게 필요로 하는 스레드를 이용하기 시작하였다. 스레드는 프로세스보다 생성 및 문맥교환 시 발생하는 오버헤드가 적고 자원 점유량이 적기 때문에 더 효율적이다. 그러나 스레드 또한 생성 및 소멸 과정에 어느 정도의 오버헤드가 필요하므로, 좀 더 비용을 최소화하기 위해 스레드를 미리 생성한 후 요청이 들어올 때마다 특정 스레드를 지정하여 처리하도록 하면 스레드 생성 시 필요한 비용을 줄일 수 있다. 물론 스레드들 사이에 동기화가 필요하며 약간의 비용이 필요하다.

위 고려 사항들을 바탕으로 하여 서버 프로그램을 적절히 구현하면 효율적인 서버 운영이 가능하다. 이러한 사항을 적용한 것의 예로서 대표적인

* 강원대학교 컴퓨터학부 학사과정
** 강원대학교 컴퓨터학부 교수

것으로서 아파치 웹서버로서 전 세계 50% 이상의 사람들이 사용하고 있다[4,5]. 아파치 웹 서버는 꾸준한 성능향상 과정을 거쳐서 오늘날의 다중 스레드 기반의 프로그램이 되었다[2,7]. 그 결과 반응시간 면에서는 약간의 성능 감소가 나타났지만 처리량이 증가하고 시스템 부하가 감소하였다[6]. 웹 서버의 과부하시에 대한 성능평가결과로는 일정 수준 이상의 과부하가 발생시에는 오히려 서버의 처리량이 줄어드는 연구결과도 제시되고 있다 [8].

본 논문에서는 대표적인 서버 프로그램 구현 방안들에 대한 직접적인 비교평가를 실시하였다. 동일한 실행환경에 각 방안들에 대한 성능평가용 프로그램을 직접 구현하여 실행하고 실행 시간을 측정하여 비교하였다.

2. 대표적인 웹 서버 구현 방안들

2.1 요청별 프로세스 생성

대표적인 웹 서버 구현 방식으로는 요청별 프로세스 생성 서버, 요청별 스레드 생성 서버, 스레드 작업자 풀 서버, 및 버퍼 이용 스레드 작업자 풀 서버의 4가지가 있다[1][2].

요청별 프로세스 생성 (Process-per-Request: PPR) 서버는 그림 1에서 보는 것과 같이 각 클라이언트의 요청을 처리하기 위해 매번 새로운 작업자 프로세스를 생성하는 방식이다. 먼저 클라이언트가 서버로 연결을 요청하면 접속을 관리하는 프로세스가 요청을 받아들이고 소켓을 생성한다. 그리고 해당 클라이언트와 통신할 작업자 프로세스를 생성하고 이미 생성한 소켓을 전달한다. 생성된 작업자 프로세스는 전달 받은 소켓을 통하여 해당 클라이언트가 종료할 때까지 요청들을 처리하게 된다.

2.2 요청별 스레드 생성

요청별 스레드 생성 (Thread-per-Request: TPR) 서버는 PPR 서버와 비슷하다. 단지, 각 클라이언트 요청을 처리하기 위해 프로세스 대신 스레드를 생성한다는 점만 다르다. 프로세스의 생성은 부모프로세스와 독립적인 주소 공간을 확보해야 하므로 별도의 메모리 및 스왑 공간 할당 작업이 수반되며, 문맥교환 과정에서 페이지 테이블을 변경하는 절차를 거치게 된다. 이에 비해서 스레드는 부모 프로세스와 동일한 주소공간을 가지므로 생성 작업에 소요되는 시간이 절약되고 문맥교환 과정도 단순해지므로 TPR 서버가 PPR 서버에 비해서 비교적 효율적일 것이다.

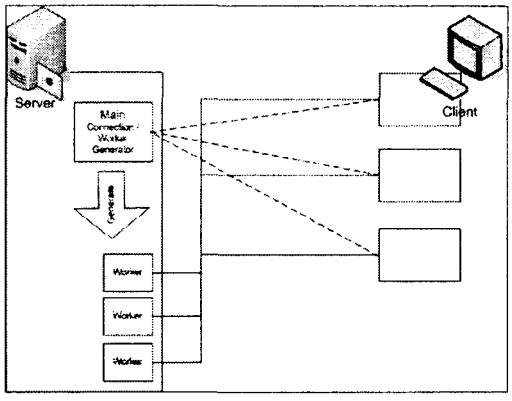

그림 1. 요청별 프로세스/스레드 생성 서버

2.3 스레드 작업자 풀

스레드 작업자 풀 (Thread Worker Pool: TWP) 서버는 일정한 개수만큼의 작업자 스레드들을 생성해 둔다. 작업자 스레드들은 대기 상태로 존재하며 동기화 과정을 통해서 경쟁적으로 클라이언트의 연결 요청을 받아들이고 서비스를 제공한다. 스레드는 클라이언트에 대한 서비스가 끝나면 다시 연결을 받아들이기 위해 대기 상태로 돌아간다. 그림 2는 TWPI 서버의 동작을 보여준다.

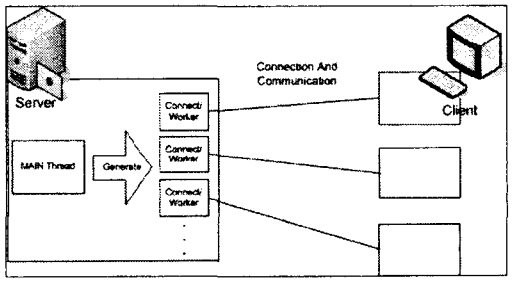

그림 2. 스레드 작업자 풀 서버

2.4 버퍼 이용 스레드 작업자 풀

버퍼 이용 스레드 작업자 풀 (Thread Worker Pool with Buffers: TWPB) 서버는 TWP 서버와 거의 비슷하지만 클라이언트의 최초 접속 요청을 받아들이는 스레드가 따로 존재하고, 작업자 스레드들이 버퍼를 통해 접속 정보를 받아들이는 점이 다르다. 최초 접속 요청을 받아들이는 스레드는 매 접속요청마다 소켓을 생성하고 그 번호를 순환 버퍼에 저장한다. 그러면 대기 중인 작업자 스레드들이 가져가서 해당 클라이언트의 요청을 처리한다. 서비스를 완료한 스레드는 다음 요청을 처리하기 위해서 버퍼로부터 다음 요청을 위한 소켓정보를 받기위해 대기한다.

버퍼는 공유메모리와 적절한 동기화 기능을 활

용하여 구현하거나 운영체제에서 제공하는 메시지 큐 또는 메일박스 기능을 직접 활용하여 구현할 수 있다. 그림 3은 유한 버퍼를 사용하는 TWPB 서버에서 각 스레드들이 소켓을 획득하고 통신하는 과정을 보여준다.

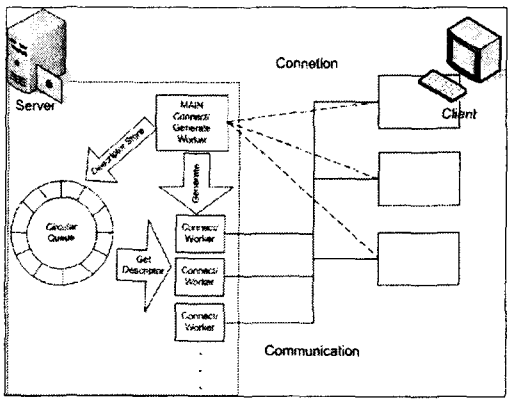

그림 3. 버퍼 사용 스레드 작업자 풀 서버

3. 실험 환경 및 결과 측정 방법

네 가지의 웹서버 구현 방안들을 비교하기 위해서 동일한 컴퓨터 환경에 클라이언트와 서버의 기본 기능을 직접 구현하고 실행 속도를 측정하였다. 표 1은 클라이언트와 서버를 실행시킨 컴퓨터의 하드웨어 사양 및 운영체제를 보여준다.

표 1. 성능평가를 위한 컴퓨터 사양

항목	클라이언트	서 버
CPU	Intel P4 2.4GHz	Via 1.0GHz
MEM	512MB	512MB
OS	Asianux 2.6.9	Asianux 2.6.11

각 서버의 성능을 측정하기 위해서는 동일한 환경에서 실행해야하고, 각 방안별 핵심 구현 방법 외에는 성능에 영향을 줄 수 있는 요인들을 제거해야 한다. 따라서 디스크 접근이나 네트워크 트래픽 등 성능에 영향을 주는 다른 요인들을 최소화하여 실험 하였다. 각 방안별 성능의 차이는 주로 프로세스 및 스레드의 생성 비용과 스레드간의 동기화 비용의 차이에서 기인할 것이다.

네트워크상의 트래픽에 의한 영향을 줄이기 위해 실험 환경의 서버와 클라이언트는 크로스 케이블을 이용하여 1:1로 직접 연결하였다. 그리고 성능평가를 위해 구현된 서버는 내부적으로 디스크 접근 동작을 하지 않도록 하였다.

성능 측정을 위해서 서버에 부하를 발생시키고 그 결과를 측정하기 위한 클라이언트 프로그램을 직접 작성하여 실행하였다. 이 클라이언트 프로그램은 지정된 수의 클라이언트 프로세스들을 생성하고, 각 프로세스는 지정된 횟수만큼 서버로 연결을 시도한다. 서버와의 연결이 이루어지면 일정 횟수의 데이터 교환을 한 후에 연결을 종료한다. 데이터 교환은 서버에게 가상의 요청 메시지를 보낸 후에 응답 메시지를 수신하는 과정을 거친다.

여기서 클라이언트 프로세스의 개수는 동시 연결 개수를 의미하며 연결 당 메시지 교환 횟수는 연결 시간을 의미한다. 클라이언트 프로그램은 시작된 후 모든 프로세스가 종료할 때까지 걸린 시간을 출력한다.

실제 성능 측정에 적용한 내용으로는 클라이언트 프로세스 개수를 50개, 100개 및 150개로 변화시키고, 프로세스 당 서버와의 연결회수를 100회씩 처리했으며, 한 번의 연결 당 메시지 교환 회수는 2회와 20회를 적용하였다. 하나의 메시지는 1204 바이트의 크기를 적용하였다. TWP 서버와 TWPB 서버의 경우에는 작업자 스레드 개수와 버퍼 크기를 변경하면서 측정하였다. 성능 측정은 서버 유형별로 5회 또는 10회 실시 후 그 평균값을 사용하였다.

4. 결과 및 분석

현재 실험 대상 서버들의 성능에 영향을 미치는 요소는 크게 프로세스와 스레드의 생성 비용, 스레드간 동기화, 그리고 생성 스레드의 개수, 버퍼의 개수 등으로 나누어 볼 수 있다. 일반적으로 프로세스의 생성 비용은 스레드 생성 비용 보다 크다. 또한 스레드 생성 비용은 스레드간 동기화 비용보다 클 것이다. 따라서 서버의 성능은 PPR 서버가 가장 낮고, 그 다음이 TPR, 및 TWP/TWPB의 순서일 것이다. TWP 서버와 TWPB 서버 간에는 동기화의 위치 및 방법이 결과에 영향을 미칠 것이다.

그림 4는 클라이언트 프로그램에서 생성하는 프로세스의 개수, 즉 서버와의 동시 연결 개수를 증가 시키면서 각 서버의 성능을 측정한 결과를 보여준다. 연결 당 메시지 송수신 횟수를 2회로 했고, TWP 서버와 TWPB 서버의 스레드 개수와 버퍼 크기는 충분히 크게 하여 가장 좋은 성능을 보이는 값을 측정한 것이다.

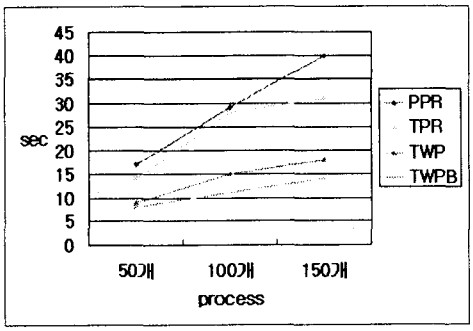

그림 4. 동시 연결 개수에 따른 소요시간

예상한 것과 같이 프로세스와 스레드 생성 비용은 서버 성능에 영향을 미쳤다. 그러나 그 영향은 TPR이 PPR에 비해 10%정도의 성능 개선을 보여주었다. 이에 비해서 TWP 및 TWPB는 PPR에 비해서 반 이하의 시간만 소모하는 것을 볼 수 있다. 이들에서는 스레드를 생성하는 시간이 서버의 시작 시에만 발생하고 클라이언트들에게 서비스하는 시간에는 동기화를 위한 시간만 소모하면 된다. 측정 결과를 보면 스레드 간 동기화 비용이 프로세스나 스레드의 생성 비용보다 훨씬 작다는 것을 보여준다. TWP에 비해서 TWPB이 우수한 것을 확인할 수 있는데, 각 작업자 스레드가 직접 소켓 연결을 시도하는 것에 비해서 주 스레드가 모든 접속요청을 받은 다음 이를 작업자 스레드들에게 배분하는 것이 효율적임을 보여주고 있다.

작업자 스레드들을 미리 생성해 놓고 처리하는 TWP 서버에서는 작업자 스레드들의 개수가 늘어날수록 성능이 개선될 수 있다. 그림 5는 서버에서 사용하는 작업자 스레드의 개수에 따른 응답시간의 변화를 보여준다. 여기서 클라이언트 프로그램에서 동시 연결 개수를 50개, 프로세스 당 연결 횟수가 100회, 그리고 연결 당 메시지 송수신 횟수는 20회일 때를 적용하였다.

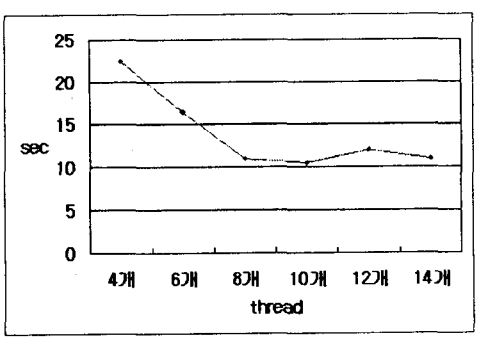

그림 5. 작업자 스레드 개수에 따른 소요시간
(동시연결 개수 50개)

성능 개선은 작업자 스레드가 일정 개수 이상이 되면 더 이상 성능이 향상되지 않는 것을 그래프를 통해 볼 수 있다. 이것은 CPU 등의 하드웨어 성능이나 네트워크 대역폭 등의 다른 요인 때문에 전체 성능이 제한되는 것으로 보인다.

그림 6은 동시연결 개수를 100개로 늘려서 측정한 것이다. 그림 5와 비교해보면, 동시연결 개수가 많은 경우에는 작업자 스레드 개수까지 성능 개선이 이루어지는 것을 확인할 수 있다. 동시연결 개수와 성능을 개선할 수 있는 작업자 스레드 개수 간에는 상관관계가 있는데, 대략 동시연결 개수의 1/6 정도의 작업자 스레드들이 필요한 것으로 판단할 수 있다. 그러나 서버의 하드웨어 사양에 따라 스레드 개수를 늘리는 데에는 현실적인 한계가 있을 것이다.

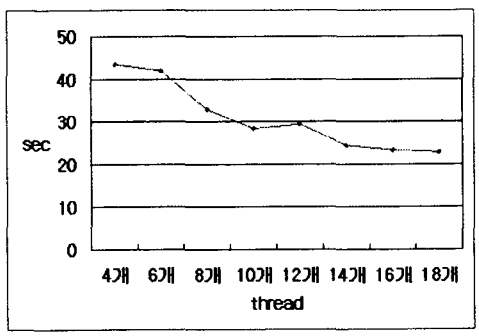

그림 6. 스레드 개수 변화에 따른 시간 변화
(동시연결 개수 100개)

여기서는 클라이언트의 연결 시간이 긴 경우(연결당 20회의 메시지 교환)를 대상으로 하였는데, 연결시간이 짧은 경우 (연결당 메시지 교환 2회)에는 작업자 스레드의 개수에 따른 영향은 거의 나타나지 않았다. 동시에 많은 클라이언트들이 접속을 요청할 때 하나의 연결 당 작업자 스레드의 작업량 (I/O 작업 등)이 많아지면 접속 요청을 하는 클라이언트들의 대기 시간이 늘어나게 되는데 적절히 많은 작업자 스레드들을 적용하면 전체적인 성능향상을 기대할 수 있을 것이다.

5. 결론

본 논문에서는 대표적인 서버 프로그램 구현 방식들에 대한 비교평가를 실시하였다. 대표적인 웹 서버 구현 방식으로는 요청별 프로세스 생성 서버, 요청별 스레드 생성 서버, 스레드 작업자 풀 서버, 및 버퍼 이용 스레드 작업자 풀 서버의 4가지가 있다. 이들에 대해 동일한 실행환경에 각 방안들에 대한 성능평가용 프로그램을 직접 구현하여 실행하고 실행 시간을 측정하여 비교하였다.

클라이언트의 요청마다 프로세스를 생성하여 서비스 하는 것은 스레드를 생성하는 것이 10% 정

도 성능 향상을 얻을 수 있었다. 그러나 스레드를 생성하는 작업을 줄이기 위한 방안으로서 미리 작업자 스레드들을 생성하여 풀을 만들어 놓고 클라이언트의 요청마다 하나의 작업자 스레드가 처리하도록 하는 방법은 50% 정도의 응답시간 감소효과를 얻을 수 있었다. 클라이언트의 요청에 대하여 소켓 연결 작업은 작업자 스레드들이 직접 처리하는 것에 비해서 전담 스레드가 연결작업을 한 후에 그 소켓을 작업자 스레드에게 전달하는 방식(WTPB)이 보다 효율적임을 확인하였다.

WTPB를 적용하는데 있어서 클라이언트들의 동시 연결 개수에 따라 적절한 작업자 스레드 개수를 정해야 한다. 연결당 소비시간이 아주 짧을 경우에는 작은 개수의 작업자 스레드만으로도 충분하지만 소비시간이 길어질수록 필요한 작업자 스레드 개수가 늘어난다. 작업자 스레드가 너무 작으면 클라이언트가 대기하는 시간이 길어지고, 너무 많으면 서버에서 불필요한 오버헤드를 초래할 수가 있을 것이다.

참 고 문 헌

[1] Kay A. Robbins and Steven Robbins, UNIX Systems Programming, *Prentice Hall*, pp.706-744, pp.970-1008, 2004.

[2] Netcraft, The Netcraft WWW Server Survey, http://news.netcraft.com.

[3] Paul S. Hethmon, Illustrated Guide to HTTP, Prentice Hall, 1997.

[4] R.B. Bloom, Apache Server 2.0: The Complete Reference, *McGraw-Hill*, 2002.

[5] Apache, Apache HTTP Server Documentation, http://www.apache.org.

[6] Kihun Chong, Miryoung Yeom and Sam H. Noh, "An Assessment of the Apache Web Server 2.0 Performance on Linux", *Proc. of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication System(SPECTS 200)*, pp. 224-231, July, 2002.

[7] Y. Hu, A. Nanda and Q. Yang, "Measurement, Analysis and Performance Improvement of the Apache Web Server", *Performance, Computer and Communications Conference*, pp. 261-267, 1999.

[8] G. Banga and P. Druschel, "Measuring the Capacity of a Web Server", *Proc. of USENIX Symposium on Internet Technologies and Systems*, Dec. 1997.