멀티 프로세스와 멀티 쓰레드 기법의 효율적 활용을 위한 아파치 2.0 웹 서버의 성능 분석

전흥석, 이승원°
건국대학교 컴퓨터 응용 과학부
hsjeon@kku.ac.kr , lswhh@rainbow.kku.ac.kr

Performance Analysis of Apache 2.0 Web Server for Efficient Utilization of Multi-Process and Multi-Thread Scheme

Heung Seok Jeon, Seung Won Lee
Dept. of Computer Science, Konkuk University

요 약

아파치 웹 서버는 사용자들에게 좀더 빠른 서비스를 제공하기 위해 멀티 프로세스 모델을 사용해 왔다. 그러나 최근 들어 웹 서비스를 사용하는 사람들이 급격히 증가함에 따라 멀티 프로세스 구조보다 더 빠르고 더 많은 요청을 처리할 수 있는 구조가 요구 되어진다. 이러한 문제를 해결하기 위해서 생성 및 문맥전환 등에 있어서 오버헤드가 프로세스 보다 더 적다고 알려진 멀티 쓰레드 모델을 도입하게 된다. 하지만 멀티 쓰레드를 사용하는 것이 항상 멀티 프로세스를 사용한 것보다 더 좋은 성능을 보여주진 않는다. 본 논문에서는 멀티 프로세스와 멀티 쓰레드를 주어진 상황에 따라 어떻게 활용하는 것이 응답시간과 처리율 면에서 더 효율적인지를 알아보기 위하여 시뮬레이터를 제작하여 아파치 웹 서버의 성능을 측정한다. 실험을 통해 아파치 웹 서버의 다양한 지시자에 대한 설정 값에 따라 멀티 프로세스와 멀티 쓰레드 구조의 성능이 달라 질 수 있음을 확인하고 이의 결과를 제시한다.

1. 서 론

아파치 1.3 버전에서는 멀티 프로세스를 기반으로 사용자들의 요청에 대해 서비스한다. 그러나 웹 서비스를 사용하는 사람들이 증가함에 따라 프로세스 보다 더 우수한 성능이 요구되었고, 그에 따라 아파치 2.0 버전에서는 멀티 쓰레드를 기반으로하는 서비스 구조로 개선되었다 [2,4]. 이러한 변화는 쓰레드가 프로세스보다 문맥 전환 (context switch) 등 에서의 오버헤드가 더 적고 자원 점유율이 낮기 때문에 웹 서버가 사용자들의 요청에 더 유연하고 빠르게 대처할 수 있게 하려는 것이다.

그러나 쓰레드를 사용하는 것이 항상 더 나은 성능을 가져오는 것은 아닐 수 있다는 한 연구 결과가 최근에 발표되었다 [1]. 그림 1은 해당 논문에서 제시한 아파치 1.3과 2.0이 다양한 종류의 문서에 대한 요구(request)를 처리한 응답 시간 (response time)을 각각 나타내 주는 그래프이다. 이 그래프에서 볼 때 쓰레드를 기반으로 한 아파치 2.0의 응답시간이 어떤 경우에 있어서는 아파치 1.3보다 더 좋지 못한 결과를 가져온 경우가 발생하였다. 이것은 쓰레드를 사용한다고 해서 항상 더 좋은 성능을 가져오는 것은 아니고, 다른 요인이 작용할 수 있음을 보여주는 것이다.

따라서 본 논문에서는 이러한 결과의 원인을 분석하고 웹 서버의 성능을 향상시킬 수 있는 방안을 제시하고자 한다.

2. 아파치 2.0 웹 서버

아파치 웹 서버는 NCSA httpd 1.3을 기반으로 개발된 웹 서버이다. 아파치는 Apache HTTP Server Project에서 지속적으로 업그레이드 버전을 발표해오고 있다 [2,4]. 현재 발표된 아파치의 최신 버전은 2.0.46 이며 본 논문에서 사용한 것은 2.0.45 버전이다.

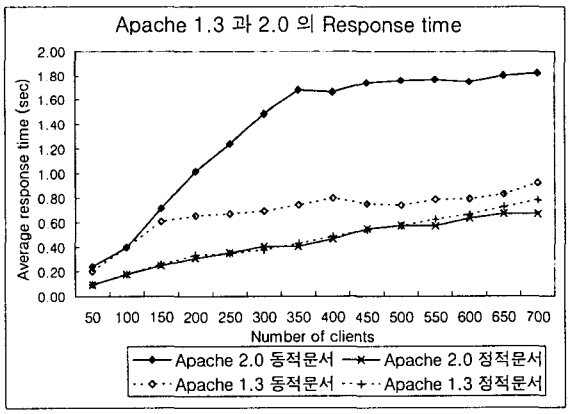

그림1. Apache1.3과 2.0의 응답 시간 비교

아파치 2.0은 특정 사이트의 요구 조건에 더 특화 하기 위해서 다양한 선택적 실행 방법을 가지고 있다. 그 중 유닉스 플랫폼에서 사용하는 대표적인 것으로 worker, prefork, 그리고 perchild 모듈이 있다. 이러한 모듈들은 컴파일 시 결정해야 하며 한번에 하나의 모듈만 컴파일 될 수 있다. 컴파일한 후 실행에 관련된 프로세스와 쓰레드의 수 등 여러 가지 설정들은 httpd.conf 파일의 지시자(directive)를 변경함으로써 가능하다[2,4].

Worker와 prefork 모듈들은 각각 확장성 (많은 요청에 대해 유연하게 대처할 수 있는 능력)이 요구되는 사이트, 안정성과 오래된 소프트웨어와의 호환성이 필요한 사이트에 사용되고, perchild 모듈은 다른 사용자 아이디로 여러 호스트를 서비스하는 것에 사용된다 [2]. 본 논문은 사용자들의 급격한 증가에 대처 할 수 있는 능력을 실험하기 위하여 worker 모듈을 사용한다.

3. Worker 모듈의 지시자 연구

Worker 모듈은 멀티 프로세스와 멀티 쓰레드의 혼합형 동작방식을 사용한다. Worker 모듈과 관련된 지시자들은 StartServers, ThreadsPerChild, MinSpareThreads, MaxSpareThreads, ServerLimit 그리고 MaxClients 가 있다. 프로그램이 실행되면 주 프로세스는 StartServers에 설정되어 있는 수 만큼의 자식 프로세스를 생성한다. 자식 프로세스들은 ThreadsPerChild에 설정에 따라 쓰레드들을 각각 생성한다. 이 쓰레드들이 실제 요청에 대한 서비스를 수행 한다. 현재의 쓰레드들 중 요청에 대한 서비스 중에 있지 않은 쓰레드들이 스페어(spare) 쓰레드이다. 어떠한 요청도 없다면, StartServers * ThreadsPerChild가 스페어 쓰레드 수가 된다.

만약 이 때 요청이 들어오면 스페어 쓰레드의 수가 줄게 될 것이고, MinSpareThreads보다 적은 수의 스페어 쓰레드가 남게 된다면, 주 프로세스는 자식 프로세스를 추가로 생성한다. 그리고 다시 그 자식 프로세스는 ThreadPerChild에 설정되어 있는 수의 쓰레드들을 생성한다. 이런 방식으로 생성될 수 있는 자식 프로세스의 총 수는 ServerLimit 까지 가능하다. 요청에 대한 응답을 마쳐서 스페어 쓰레드가 MaxSpareThreads 보다 많게 된다면 자식 프로세스 중 하나를 제거한다. 이렇게 해서 MinSpareThreads와 MaxSpareThreads사이의 수로 스페어 쓰레드들을 유지한다. 그리고 만약 어떤 자식 프로세스가 처리한 요청의 수가 MaxRequestsPerChild 만큼이 되면 그 자식 프로세스를 제거 한다. 이 지시자가 0으로 설정되어있으면 처리한 요청의 수를 제한하지 않는다. MaxClients는 동시에 서비스 가능한 클라이언트의 수를 설정하는데 쓰이는 지시자 인데, 이것은 ServerLimit * ThreadsPerChild까지 제한된다. 만약 연결 요청이 MaxClients 보다 많이 들어온다면 ListenBacklog (대기하는 queue의 크기를 정하는 지시자)의 수까지 대기하게 된다.

위와 같은 다양한 지시자들 중에서 본 논문에서 관심이 있는 것은 스페어 쓰레드를 유지하기 위한 MinSpareThreads와 MaxSpare Threads 들이다. 본 연구의 진행과정에서 다양한 지시자의 변화를 실험하였고, 그 결과에 의하면 위의 스페어 쓰레드를 유지하기 위한 지시자들이 전체 성능에 영향을 미칠 수 있음을 발견하였다. 자세한 내용은 다음절에서 설명한다.

4. 성능 평가

4.1 실험 내용 및 환경

이 절에서는 앞절에서 언급한것처럼 아파치의 지시자 설정이 성능에 미치는 영향을 분석하기 위한 실험을 진행한다. 본 연구를 위한 실험은 256 MB의 메모리와 Pentium III 1.0 GHz의 CPU를 탑재한 레드헷 8.0 리눅스에서 진행한다. 클라이언트 부분의 시뮬레이터는 자체 제작한 쓰레드 기반의 시뮬레이터를 사용한다. 이 시뮬레이터는 동시접속 클라이언트를 묘사하기 위해서 하나의 쓰레드가 한 클라이언트를 대신 한다. 그림 2에서 시뮬레이터의 전체적인 구성을 보여준다.

4.2 실험 및 분석

본 논문에서는 실험을 위하여 아파치의 지시자들을 다음과 같은 두 가지 유형으로 설정한다. 즉, 하나의 설정은 계속되는

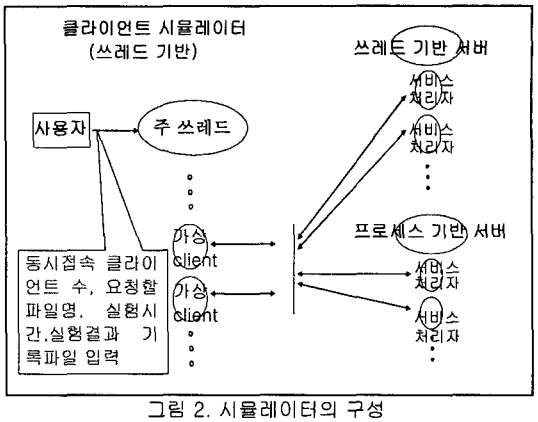

그림 2. 시뮬레이터의 구성

사용자의 요구를 처리하기 위해서 멀티 프로세스를 주로 활용하도록 하고, 다른 하나는 멀티 쓰레드를 주로 활용하도록 하는 것이다. 구체적으로 설명하자면 다른 지시자는 기본 설정에서 변화하지 않은 상태에서 시작하는 프로세스 개수인 StartServers와 서버 프로세스의 개수를 제한하는 ServerLimit 그리고 ThreadPerChild를 변화하였다. 쓰레드를 주로 사용하는 아파치와 프로세스를 주로 사용하는 아파치의 처음 시작하는 쓰레드의 개수(StartServers*ThreadPerChild)는 같으며, 동시에 처리할 수 있는 최대 클라이언트 수(MaxClients)도 같게 설정하였다. 표1에서 그 두 가지 설정을 자세히 보여준다.

표 1. 프로세스와 쓰레드 비교 실험을 위한 아파치 지시자 설정

지시자(directive)	쓰레드를 주로 사용하는 경우	프로세스를 주로 사용하는 경우
ServerLimit	3	150
StartServers	1	50
MaxClients	150	150
MinSpareThreads	25	25
MaxSpareThreads	75	75
ThreadsPerChild	50	1
MaxRequestsPerChild	0	0

쓰레드의 문맥전환의 오버헤드는 프로세스의 70%정도라고 알려져 있기 때문에[3], 쓰레드를 주로 사용한 경우에 응답시간과 처리율 면에서 우수할 것으로 예상된다. 그러나 실제 실험 결과에 의하면 예상과는 다르게 쓰레드를 주로 사용하는 설정을 가진 아파치가 프로세스를 주로 사용하는 아파치 보다 더 좋지 못한 결과를 보여준다.

그림 2와 3에서 볼 수 있듯이 응답시간은 프로세스를 주로 사용하는 아파치가 더 빠르게 나왔고, 처리율 면에서도 프로세스를 주로 사용하는 아파치의 처리율이 더 많은 것으로 나타났다. 이것은 멀티 프로세스와 쓰레드를 사용하는 과정에서 다른 요인이 작용한 것으로 판단된다.

이와 같은 결과에 대한 원인을 분석하기 위하여 다음과 같은 추가 실험을 진행하였다. 표 2는 추가 실험에 사용된 지시자의 내용을 보여준다.

앞선 실험에서는 쓰레드 들을 미리 생성해 놓고 스페어 쓰레드들을 정해진 지시자대로 유지하기 위해서 지속적으로 쓰레드를 가진 프로세스를 생성하고 제거하는 작업을 반복 한다. 그러나

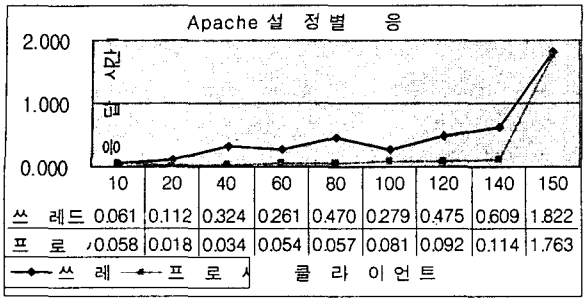

그림 2. 표 1의 경우 쓰레드를 주로 사용하는 아파치와 프로세스를 주로 사용하는 아파치의 응답시간 비교

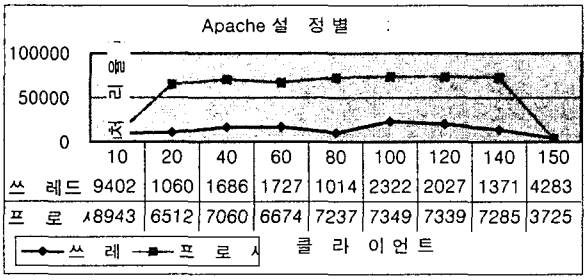

그림 3. 표 1의 경우 쓰레드를 주로 사용하는 아파치와 프로세스를 주로 사용하는 아파치의 처리율 비교

표 2. 오버헤드 분석 실험을 위한 아파치 지시자 설정

지시자(directive)	쓰레드를 주로 사용하는 경우	프로세스를 주로 사용하는 경우
ServerLimit	3	150
StartServers	3	150
MaxClients	150	150
MinSpareThreads	150	150
MaxSpareThreads	150	150
ThreadsPerChild	50	1
MaxRequestsPerChild	0	0

표 2를 통한 실험에서는 스페어 쓰레드들을 유지하기 위해서 자식 프로세스를 제거하거나 생성하는 일이 없다. 이는 처음 시작 시 총 150개의 쓰레드들을 생성해 놓으면 그 쓰레드의 개수는 MinSpareThreads와 MaxSpareThreads 인 150개에 적절하여 더 이상의 조절이 필요없기 때문이다. 그리고 요청이 들어왔을 때 스페어 쓰레드들이 줄어들게 되어도 ServerLimit에서 제한을 받기 때문에 프로세스를 생성하지 않는다.

그림 4와 5는 추가 실험에 대한 실험 결과이다. 실험 결과는 예상대로 쓰레드를 주로 사용한 아파치가 응답시간과 처리율 면에서 우수한 성능을 보여준다. 이상과 같은 실험 결과를 볼 때, 첫 번째 실험에서 쓰레드 위주의 아파치가 나쁜 성능을 보여준 것은 스페어 쓰레드를 유지하는데 드는 비용이 쓰레드를 이용한 장점을 살리지 못한 주요 원인으로 분석된다. 이것은 쓰레드를 웹 서버에 이용하는 데 있어서 성능을 저하시킬 수 있는 여러 요소들을 고려해야만 프로세스를 이용한 웹 서버보다 더 좋은 성능을 가질 수 있다는 것이다. 그 중 응답시간을 단축하기 위해서 고안된 스페어 쓰레드들의 범위에 대한 더 많은 고찰이 필요할 것이다.

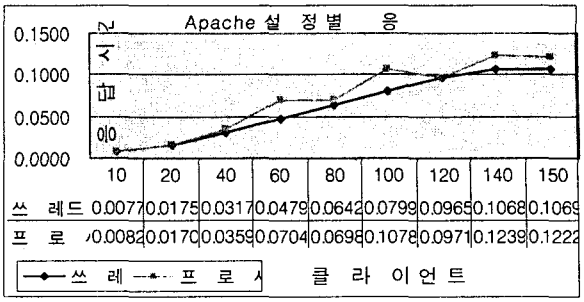

그림 4. 표 2의 경우 쓰레드를 주로 사용하는 아파치와 프로세스를 주로 사용하는 아파치의 응답시간 비교

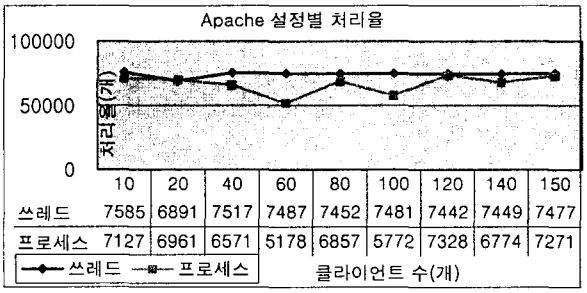

그림 5. 표 2의 경우 쓰레드를 주로 사용하는 아파치와 프로세스를 주로 사용하는 아파치의 처리율 비교

5. 결론 및 향후 연구

본 논문에서는 멀티 프로세스와 쓰레드를 웹 서버에 이용할 경우에 있어서 영향을 끼칠 수 있는 지시자에 대해 실험적인 방법을 통해 분석하였다. 점점 늘어나는 웹 사용자들에 대처하기 위해서 웹 서버는 더욱 강력한 성능을 요구 받는다. 이러한 요구에 대응하기 위해서 웹 서버는 상황에 맞는 효율적인 지시자 설정을 통해 더 높은 확장가능성과 자원이용률을 보여야 할 것이다. 현재 우리는 가장 좋은 성능을 낼 수 있는 웹 서버의 멀티 쓰레드와 멀티 프로세스의 구조를 자동으로 설정하기 위한 연구를 진행하고 있다. 진행되는 연구를 통해 다양한 상황에 맞는 적절한 설정이 어떤 것인지에 대한 방향을 제시하게 될 것이다.

참고문헌

[1] Mi-ryeong Yeom, Kihun Chong, Sam H. Noh, An Assessment of the Apache Web Server 2.0 Performance on Linux, In Proceedings of 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002) July 14-18, 2002.

[2] The Apache Software Foundation http://www.apache.org, 1999-2003.

[3] http://www.atnf.csiro.au/~rgooch/benchmarks/linuxscheduler.html

[4] Ryan B. Bloom, Apache Server 2.0: The Complete Reference, McGraw-Hill Osborne Media, 1st edition June 26, 2002