Node.js에서 멀티쓰레드 기능의 필요성

김가연 ○ 이욱진

한양대학교 컴퓨터공학과

ruredi@hanyang.ac.kr, scottlee@hanyang.ac.kr

The Necessity of Multithread Feature in Node.js

Gayeon Kim ○ Scott Uk-Jin Lee

Department of Computer Science and Engineering, Hanyang University

요 약

스마트 시대가 도래하면서 장소에 구애 받지 않고 인터넷에 접속하여 웹을 사용할 수 있는 시대가 되었다. 인터넷 접근성의 향상으로 웹 어플리케이션의 수가 급증했고 또한 단순한 스크립트 언어였던 JavaScript는 다방면으로 널리 사용되기 시작했다. Node.js는 JavaScript 기반의 서버측 언어로 가장 빠르게 성장하고 있지만 쓰레드를 사용하지 않기 때문에 서버의 리소스를 충분히 활용하지 못하는 문제가 발생했다. 본 논문에서는 멀티쓰레드를 사용하는 다른 서버측 언어들과 Node.js의 성능을 벤치마크를 통해 서로 비교하고, 이를 통해 Node.js에서 멀티쓰레드 기능의 필요성을 강조한다.

1. 서 론

바야흐로 스마트 시대가 도래하면서 월드 와이드 웹의 위상도 달라졌다. 이전에는 웹과 응용프로그램의 경계가 확연했다면 지금은 장소에 구애 받지 않고 인터넷에 접속 가능한 시대가 되었다.

인터넷 접근성이 높아지면서 웹 어플리케이션의 수와 질이 빠르게 높아졌다. 또한 웹 어플리케이션이 발달하면서 자연히 JavaScript와 같은 웹 기반 언어의 인기도 높아졌다. 실제로 JavaScript는 프로그래밍 언어 인기 조사에서 2000년대 이후 꾸준히 10위 이내에 들고 있으며[1] 2013년 이후의 GitHub에 업로드 된 프로젝트 중 JavaScript로 된 프로젝트가 가장 많은 수를 차지하고 있다[2].

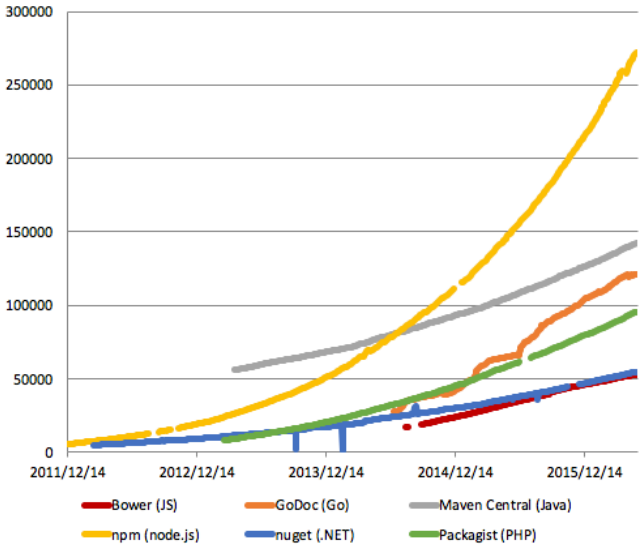
뿐만 아니라 Node.js가 개발됨으로써 JavaScript는 서버측 네트워크 언어로서도 널리 사용되고 있다. Node.js는 2009년에 처음 공개되었고 매년 100% 가량의 성장률을 보이며 현재는 350만명의 사용자를 보유하고 있다[3].

간단한 UI나 이벤트를 처리했던 JavaScript를 서버사이드 언어로 사용하게 되면서 계산량이 급격하게 늘어나게 되었지만 Node.js에서는 멀티쓰레드를 공식적으로 지원하고 있지 않다. 본 논문에서는 Node.js에서 멀티쓰레드를 지원하지 않기 때문에 생기는 단점에 대해 지적하고, 다른 멀티쓰레드를 지원하는 언어들과의 성능을 비교 및 분석하고, Node.js에서 다수의 코어를 사용할 수 있게 하는 기능인 Cluster 모듈과 그 한계에 대해 소개한다.

본 논문의 구성은 다음과 같다. 2장에서는 Node.js의 성장세와 동작 방식, 그리고 그 단점에 대해 서술한다. 3장에서는 다른 서버측 프로그램과 Node.js의 쓰레드 사용에 따른 성능을 비교하고, 4장에서는 Node.js의 멀티 프로세스 모듈 Cluster의 장단점을 소개한다. 마지막으로 5장에서는 본 논문의 내용을 정리하고 향후 연구 방향에 관해 논의한다.

2. Node.js와 멀티쓰레드

Node.js는 최근 크게 인기를 끌고 있다. 특히 npm(node package modules)로 대표되는 Node.js 모듈 생태계가 빠르게 성장하면서 사용자가 많이 늘어났다.



[그림 1] 업로드 된 모듈 개수 추이

[그림 1]은 웹에서 사용하는 여러가지 모듈의 개수를 조사한 그래프이다. npm은 이들 가운데서도 단연코 가

장 빠른 성장세를 보였으며 현재도 하루에 360개 가량의 모듈이 업로드 되고 있는 것으로 나타났다[4].

Node.js는 이벤트 주도(event-driven), Non-blocking I/O 모델을 사용한다. 그러나 다른 서버측 프로그래밍 언어에서 흔히 채용하는 쓰레드 개념을 사용하지 않는다.

Node.js 공식 홈페이지에서는 쓰레드 기반의 네트워크는 비효율적이고 사용하기 어렵기 때문에 지원하지 않으며, 데드락(Dead-lock)이 발생하지 않는다는 장점도 소개하고 있다[3].

그러나 많은 경우에 Node.js를 사용하는 웹 어플리케이션에서 쓰레드를 사용하지 않지만 Blocking이 발생한다. 현대 웹 어플리케이션에서 데이터베이스 서버를 사용하는 것은 일반적이고 데이터베이스는 다른 프로세스에서 실행되기 때문이다.

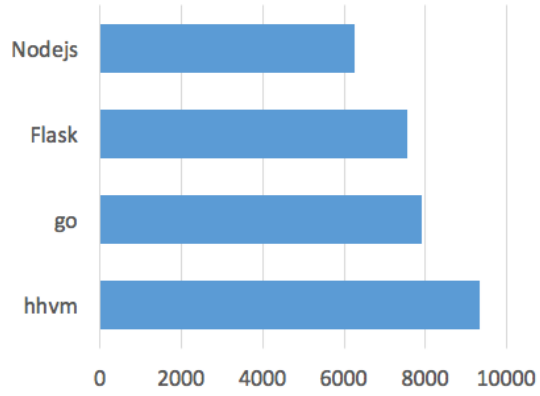
또한 Node.js는 구글 Chrome에서 사용하는 V8 JavaScript 엔진을 사용하고 있는데 이 엔진의 한계 때문에 한 프로세스에 최대 1.5GB의 메모리만 사용할 수 있다. 대부분의 서버 컴퓨터가 듀얼코어 이상의 CPU를 제공하고 1.5GB 이상의 메모리를 제공하지만 Node.js는 단일 프로세스, 단일 쓰레드로 되어있기 때문에 서버의 리소스를 충분히 활용할 수 없다.

3. 여러가지 서버측 프로그래밍 언어들과의 성능 비교

이 장에서는 TechEmpower에서 진행한 Framework Benchmark 12라운드의 결과를 분석하여 실제로 멀티쓰레드를 사용하는 다른 서버 측 프로그래밍 언어들과 Node.js가 다수의 쿼리를 처리하는 속도를 비교해 본다. 이 실험은 Dell R720xd dual Xeon E5-2660 v2 CPU와 32GB 메모리, 10GB 이더넷 환경에서 진행되었다. 여러 개의 쿼리 요청을 동시에 보냈을 때의 성능을 비교하기 위하여 한 요청 당 1, 5, 10, 15, 20개의 쿼리를 동시에 256번 보냈을 때의 초당 쿼리 처리 개수를 측정했다.

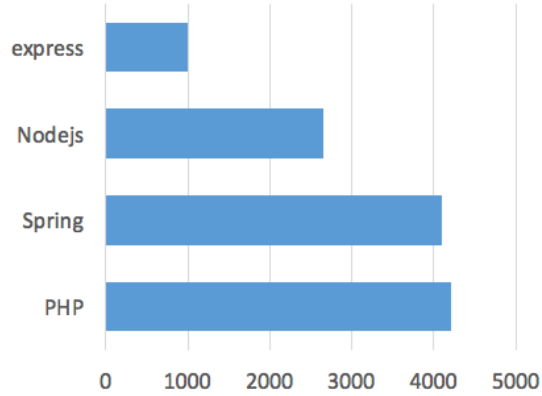
데이터베이스를 처리하는 방법에 따라 처리 속도가 크게 달라지기 때문에 벤치마크에서 제공하는 결과들 중에 Object-Relational Mapping(ORM)을 이용한 결과와 이용하지 않은 결과를 따로 비교하였다. ORM은 관계형 데이터베이스를 객체 지향 프로그래밍에서 사용할 때 객체와 테이블 간의 관계를 자동으로 설정하고 처리해주는 기능이다.

아래의 [그림 2]는 벤치마크에서 ORM을 사용하지 않은 결과이다. hhvm은 페이스북에서 개발한 PHP를 JIT (just-in-time) 컴파일으로 실행하는 엔진이다. Go는 구글에서 개발한 프로그래밍 언어이고 Flask는 파이썬 언어 기반의 웹 프레임워크이다. 그리고 이들은 Node.js를 제외하고는 모두 멀티쓰레드를 제공한다.



[그림 2] ORM을 사용하지 않은 서버의 쿼리 응답 개수[5]

결과에서 go는 컴파일 언어이고, hhvm은 스크립트 언어인 PHP를 JIT 컴파일 방식으로 일종의 컴파일을 한 결과이기 때문에 비교적 처리 속도가 빠르다는 것을 알 수 있다. 특히 hhvm은 초당 9322개의 쿼리를 처리했다. 그러나 Flask는 스크립트 언어인 파이썬을 이용한 프레임워크이지만 go에 비해 초당 쿼리 처리 개수가 381개 밖에 차이가 나지 않는 비등한 결과를 보였다. 반면에 Node.js와 go의 초당 쿼리 처리 개수는 1666개 차이가 났다.



[그림 3] ORM을 사용한 서버의 쿼리 응답 개수[5]

[그림 3]은 ORM을 사용한 경우의 결과이다. 데이터베이스를 객체로 만드는 과정이 중간에 들어가기 때문에 [그림 2]의 결과보다 쿼리 처리 개수가 거의 50% 정도 줄어드는 등 성능이 많이 떨어진다. 그러나 많은 웹 프레임워크에서 데이터를 객체로 사용할 수 있다는 구조적 장점 때문에 ORM을 사용하고 있다.

PHP는 대표적인 서버측 개발 언어이고 Spring은 Java를 기반으로 한 프레임워크이다. 벤치마크 결과에 따르면 이들은 Node.js에 비해 초당 약 2000여개의 쿼리를 더 처리할 수 있었다. express는 Node.js에서 가장 흔히 쓰이는 프레임워크이다. 많은 Node.js 기반 웹 어플리케이션이 express로 되어있지만 쿼리 처리

속도는 프레임워크를 사용하지 않았을 때 보다 느리다.

이러한 벤치마크 결과들을 보았을 때 Node.js가 다수의 쿼리를 처리하는데 다른 서버측 개발 언어보다 느리다는 것을 알 수 있다. 그리고 Node.js 웹 어플리케이션에서 가장 많이 사용되는 express는 특히 이 능력이 떨어진다는 것을 알 수 있다.

웹 어플리케이션 서비스를 제공하는 많은 스타트업에서는 빠르게 서비스를 구축하기 위하여 Node.js를 기반으로 하는 express를 이용한다. 그러나 사용자가 늘어났을 때의 속도 저하가 심각하기 때문에 다른 언어를 이용한 프레임워크로 다시 구현하는 경우가 빈번하다. 그것으로 인하여 많은 시간과 인력의 낭비가 발생하며 무시할 수 없는 수준이다.

4. Node.js에서 멀티 코어를 사용 가능한 기능 소개

이 장에서는 Node.js 환경에서 병렬 프로그래밍을 지원하는 기능인 Cluster 의 특징과 장단점을 살펴본다.

Cluster는 Node.js 내에서 여러 개의 프로세스를 생성하여 실행할 수 있는 모듈이다[3]. Node.js 공식 홈페이지에서도 다수의 코어를 사용하는 환경에서 자원을 충분히 활용하기 위하여 Cluster 모듈을 사용하라고 권장하고 있다. 또한 Cluster 모듈은 로드 밸런싱(load-balancing)을 지원하기 때문에 한쪽 프로세스에 작업량이 몰리는 것을 방지해준다.

그러나 Cluster는 프로세스를 여러 개 생성하는 방식이기 때문에 데이터 동기화 문제가 발생한다. 만약에 세션을 사용하려고 하면 외부에 캐시 서버를 두고 저장하는 방법을 이용해야 한다. 이런 방법을 사용하면 데이터베이스 서버를 하나 더 사용하는 것과 같이 잦은 Blocking이 발생하게 될 것이다. 그리고 여러 개의 프로세스를 사용하는 방법은 여러 개의 쓰레드를 사용하는 방법보다 문맥 교환(Context-switching) 오버헤드가 많이 발생한다[6]. 이런 문제로 인하여 여러 개의 프로세스를 사용한다고 하여도 예상 했던 것 만큼의 성능 향상 효과를 얻을 수 없다.

5. 결론 및 향후 연구

JavaScript 기반 서버측 언어인 Node.js는 Non-blocking I/O 모델을 사용하기 때문에 다른 서버측 언어에서 널리 사용하는 멀티쓰레드 기능이 없다. 현대의 많은 서버용 컴퓨터들이 다수의 코어를 사용하고 있지만 단일 쓰레드, 단일 프로세스로 되어 있는 Node.js 웹 어플리케이션에서는 이와 같은 서버의 자원을 활용하기 어렵다.

본 논문에서 소개한 벤치마크에서 확인 할 수 있듯이 Node.js는 멀티쓰레드를 사용하는 다른 서버측 언어에 비해 여러 개의 쿼리를 처리하는 속도가 떨어진다. 단일 프로세스의 단점을 극복하기 위하여 Cluster라는 모듈이 존재하지만 이 모듈 역시 멀티쓰레드를 지원하지 않기 때문에 데이터 동기화에 문제가 발생한다.

이러한 문제들을 해결하기 위해 Node.js는 쓰레드 개념을 도입하는 것이 적절할 것이며, 멀티쓰레드의 사용을 통하여 성능 향상을 이끌어 낼 수 있을 것이다.

본 논문에서 분석한 결과를 토대로 하여 추후에는 Node.js에서 쓰레드를 도입하여 사용하는 모듈을 개발하고 기존의 단일 쓰레드 모델과 성능을 비교할 예정이다.

사 사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [R0601-16-1063, ICT 장비용 SW 플랫폼 구축]

참 고 문 헌

[1] TIOBE Index for May 2016 http://www.tiobe.com/tiobe_index

[2] A. La, Language Trends on GitHub https://github.com/blog/2047-language-trends-on-github

[3] Node.js 공식홈페이지 http://nodejs.org

[4] Module Counts http://www.modulecounts.com

[5] TechEmpower - Web Framework Benchmarks https://www.techempower.com/benchmarks/#section=data-r12

[6] A. Agarwal, “Performance Tradeoffs in Multithread Processors,” IEEE Trans. Parallel Distrib. Syst., vol. 3, no. 5, pp. 525-539, Sep. 1992.