프로토콜 버퍼를 사용한 웹 기반 시스템의 성능 향상에 관한 연구

김선종, 김희천
한국방송통신대학교 대학원 정보과학과
e-mail:kkokey@gmail.com

A Study on Performance Improvement of Web-Based Systems Using Protocol Buffers

Sun-Jong Kim*, Hee-Chern Kim*
*Dept. of Computer Science, Graduate School, Korea National Open University

요 약

웹 시스템의 구축 방법에는 하나의 서버에 웹 서버와 DB 서버를 모두 설치하여 운영하는 방법과 여러 개의 서버에 나누어 설치하는 경우가 있는데 두 가지 방법 모두 초기 구축 후 사용자가 많아지면 서버에 과부하가 생겨 해당 웹 사이트의 응답 속도가 느려지는 등 정상적으로 동작 하지 않는 문제가 생긴다. 이러한 문제의 해결책으로 서버를 증설하여 부하를 줄이는 방법을 고려할 수 있지만 해당 방법은 비용적인 측면에서 효율적이지 못하다. 그러므로 본 논문에서는 서버에서 동작하는 프레젠테이션 로직을 기존의 전형적인 톰캣 기반의 웹 서버를 노드제이에스 서버로 교체하여 클라이언트 PC의 브라우저에서 동작하도록 하여 서버의 부하를 줄이고, 또 연계되어 설계될 타 서버와의 확장성과 데이터 처리량의 감소를 위해 데이터 전송 방법으로 프로토콜 버퍼를 사용하여 통신하는 자바스크립트 프레임워크를 제시한다.

1. 서론

기업이 웹기반 시스템을 구축한 후 가장 염려되는 점은 사용자수가 증가하면 서버 과부하로 인해 시스템이 급격히 느려진다는 점이다. 이러한 문제의 대응책으로 서버 증설을 고려할 수 있다. 하지만 서버를 증설하면 서버 내부의 아키텍처를 다시 구성해야 하며 추가된 서버에 대해서는 라이선스 비용이 추가로 발생한다. 이러한 부담을 줄이면서 성능을 유지하기 위한 방법의 연구가 필요하다.

본 논문에서는 서버에서 동작하는 프레젠테이션 로직을 자바스크립트 프레임워크를 사용하여 클라이언트의 브라우저에서 동작하게 하여 부하를 감소시키고, 프로토콜 버퍼를 사용하여 전체 데이터 처리량을 감소시키며, 서버증설 비용과 추가적인 라이선스 비용이 발생하지 않도록 노드제이에스를 사용하는 방법을 제안한다.

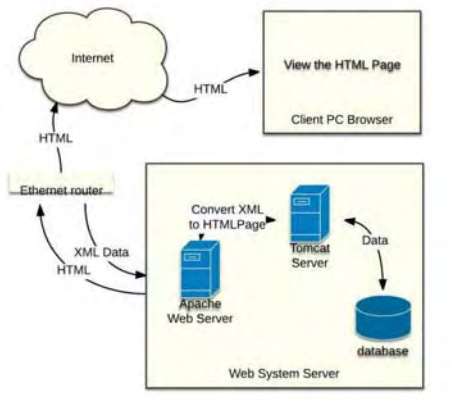
2. 관련연구

2.1 프로토콜 버퍼

프로토콜 버퍼(protocol buffer)는 구글에서 개발한 데이터 구조 형식이다. 프로토콜 버퍼는 유연하고 효율적인 자동화 메커니즘을 통해 데이터를 쉽게 전송할 수 있게 한다. XML과 비슷하다고 생각 할 수 있지만, 데이터양이 적으며 빠르고 간단하게 사용할 수 있게 한다. .proto라는 파일 형식에 한번만 구조를 정의하면 C++, Java, JavaScript, Python에서 구조를 정의하는 코드를 다시 작성하지 않고도 동일하게 사용이 가능하다[1].

2.2 기존 아키텍처 구성

기존 웹기반 시스템의 아키텍처에서 사용자가 적은 소규모 웹 시스템인 경우 하나의 서버에 웹 서버와 WAS 서버 그리고 데이터베이스 시스템을 한꺼번에 설치한다. 하지만 이러한 경우 한 개의 서버가 모든 것을 동작시키기 때문에 웹 시스템의 사용자가 조금만 늘어나도 서버에 강한 부하를 주게 된다. 이러한 경우 데이터베이스 시스템을 다른 서버에 구성하게 된다.



(그림 1) 기존 웹기반 시스템의 아키텍처

그림 1은 기존 시스템의 아키텍처가 어떤 방식으로 구성되어 있는지 보여준다. 프레젠테이션 로직을 서버에서 처리하는 그림 1은 사용자가 인터넷을 통해 해당 서버의 웹사이트에 접속하고 요청하게 되면, 서버에서는 사용자가 요청한 데이터를 DB에서 가져와 변환하며, 사용자의 브라우저에서 볼 수 있게끔 일반적인 데이터에서 HTML페이지로 렌더링하여 사용자로 하여금 웹 페이지를 볼 수 있게 한다. 이러한 서버 구성에서 사용자가 늘어나면 서버에 부하를 주게 되고 서버 리소스를 접속자 처리에 사용하게 되면서 서버 전체의 리소스가 부족하게 된다. 결국 클라이언트의 접속처리를 하는데 지연현상이 발생하게 된다. 이것의 해결책으로 서버의 증설이나 분산 시스템을 구성하여 서버에 가해지는 부하를 줄이는 방법을 고려할 수 있다[2].

2.3 톰캣

톰캣(Tomcat)은 Apache에서 오픈소스로 개발된 서블릿(Servlet) 컨테이너이며 JSP를 지원하는 웹 애플리케이션 서버이다. 사용자와 자바 응용 사이를 연결 시켜주는 서블릿을 사용하여 사용자가 보게 되는 웹 브라우저에 서버의 데이터를 HTML페이지로 변환하여 표시해준다.[3]

2.4 노드제이에스

노드제이에스(Node.js)는 서버 측에서 동작하는 소프트웨어 플랫폼이다. 자바스크립트 엔진인 V8 위에서 동작하며 코드 작성을 위해 자바스크립트를 이용한다. 내장 HTTP 서버 라이브러리를 포함하고 있어 아파치 서버 등의 별도 소프트웨어가 없어도 웹 서버로 동작이 가능하다. 저사양의 임베디드 시스템에서도 약간의 커스터마이징을 거치면 무리없이 서버 역할을 수행 할 수 있다[3]. 본 논문에서는 라이선스 문제를 해결하기 위해 기존의 전형적인 웹 서버 톰캣 대신에 Node.js를 사용하였다. 톰캣을 웹 서버로 사용하게 되면 JVM을 사용하게 되는데 이 경우 GPL 라이선스를 채택하고 있는 JVM 때문에 기업에서 톰캣을 사용한 웹 서비스를 할 경우 해당 자바 소스코드를 모두 공개해야 하는 의무가 생긴다. 기업 기밀인 소스코드를 공개해야 됨으로써 기업 입장에선 큰 문제가 생긴다. 반면 Node.js의 경우 BSD계열의 MIT 라이선스로 소스코드를 무조건 공개해야 하는 의무는 없기 때문에 기업 기밀인 소스코드를 공개하지 않아도 된다[4].

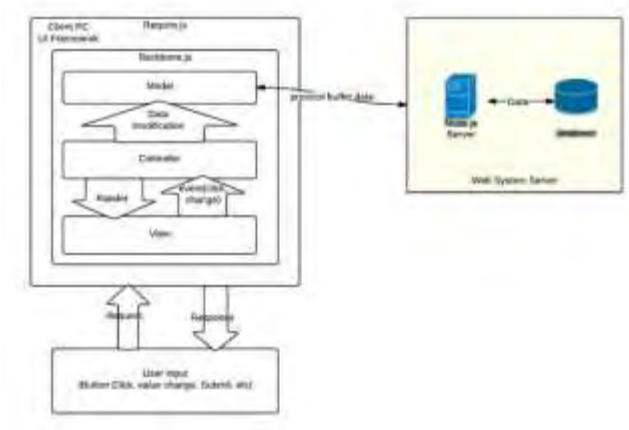
3. 웹 기반 시스템의 아키텍처 제안

3.1 클라이언트 기반의 아키텍처

클라이언트 기반의 아키텍처에서 가장 핵심이 되는 부분은 서버에서 동작하는 프레젠테이션 로직을 클라이언트에서 동작하게 하여 시스템에 부하를 주지 않음과 동시에 서버의 라이선스 문제에서 자유로워지는 것이다. Node.js를 사용하여 라이선스 비용을 지불하지 않고 개발 할 수 있게 하였으며 Node.js로 만든 웹서버를 통해 데이터베이스에 접속하여 데이터를 가져올 수 있게 하였다. 대신 시스템에 부하를 최대한 줄이기 위해 HTML 데이터가 아닌 프로토콜 버퍼 데이터를 생성하여 클라이언트에 보내주고 해당 데이터는 클라이언트에서 자바스크립트 프레임워크를 통해 사용자가 볼 수 있는 페이지를 만들도록 구성하였다.

3.2 프레임워크 구성

WAS 서버에서 동작하던 MVC 패턴 방식의 프레젠테이션 로직을 클라이언트로 이동시키기 위해서는 여러 개의 자바스크립트 프레임워크가 필요하다. 프레임워크에서 실질적으로 MVC 패턴을 구현하는 것은 Backbone.js를 사용하여 대신하게 되고 불필요한 로딩을 줄이기 위한 라이브러리 동적 로딩을 위해서는 require.js가 필요하며 Handlebars.js를 사용하여 HTML에 실제 데이터를 매핑시키고 사용자의 브라우저에 데이터가 입혀진 페이지를 보여주게 된다. 그림 2는 해당 프레임워크의 구성을 그림으로 표현한 것이다.



(그림 2) 프레임워크 구성도

4. 평가

데이터베이스에서 데이터를 뽑아서 XML 데이터로 변환한 후 화면에 보여주는 것과 프로토콜 버퍼를 사용하여 인코딩한 데이터를 화면에 보여주는 것을 각각 테스트 하여 비교하였다. 실험을 위해 Microsoft Azure를 사용한 가상환경을 설치하였고, 측정을 위해 크롬의 개발자 도구를 사용하였다.

두 가지 사례를 비교 하였다. 첫째는 톰캣과 XML을 이용하여 기존 방식으로 구축된 서버기반 시스템이고, 둘째는 Node.js와 프로토콜 버퍼를 이용한 클라이언트 기반 시스템이다. 여기서 “서버에서 로딩된 데이터”를 보여주는 페이지의 속도 차이를 측정하였다. 데이터는 총 10만개의 테이블 행을 사용하였고, 측정 항목은 총 페이지 로딩 시간과 브라우저의 메모리 사용량이다.

XML과 프로토콜 버퍼를 각각 세 번 측정하여 데이터

를 비교하였다. 다음 그림 3은 XML 데이터를 사용하였을 때 로딩 시간을 측정한 결과이다.

(그림 3) XML 데이터 페이지 로딩 시간 측정

다음 그림 4는 프로토콜 버퍼를 사용 했을 때 로딩 시간을 측정한 결과이다.

(그림 4) 프로토콜 버퍼 페이지 로딩 시간 측정

위 그림에서 나타난 전체 로딩 시간을 표로 만들면 다음과 같다.

(표 1) XML과 PB의 비교

	1회	2회	3회
XML	45.38s	16.02s	15.13s
프로토콜 버퍼	1.81s	1.49s	1.64s

실험에서 측정된 시간을 보면 XML은 최대 45.38s 프로토콜 버퍼의 경우 최대가 1.81s로 약 25배 정도 XML의 로딩시간이 더 긴 것을 확인 할 수 있었다.

5. 결론

본 논문에서는 사용자수 증가로 인한 웹기반 시스템의 과부하를 해결하기 위해 시스템을 확장할 때, 서버의 추가 증설은 추가 비용이 들고 경우에 따라서는 전체 시스템 아키텍처를 재설계해야 한다는 점과 추가 라이선스 비용이 들어간다는 문제를 해결하고자 하였다. 라이선스 비용 문제를 해결 하기 위해 오픈소스인 Node.js를 사용하여 웹서버를 구축하고, 서버의 리소스 사용을 줄이기 위해 자바스크립트 프레임워크를 사용하여 프레젠테이션 로직을 클라이언트로 이동시켜 동작하게 하고, 타 서버와의 연계를 고려한 데이터 전송 방법인 프로토콜 버퍼를 사용하여 전체 데이터 처리량을 감소시키는 자바스크립트 프레임워크를 제안하였다.

실험을 통해 결과 데이터를 로딩 하는 시간을 최대 25배까지 줄일 수 있는 것을 확인하였다. 모든 상황에서 적용 할 수 있는 것은 아니지만, 본 논문에서 제시한 방법을 사용하여 서비스의 전부 또는 일부를 구축한다면 기업 입장에서는 많은 양의 비용을 줄일 수 있을 것이라고 기대한다.

참고문헌

[1] Protocol Buffers - Google's data interchange format, Copyright 2008 Google Inc. (http://code.google.com/p/protobuf)

[2] 김영찬, 김태간, 이세훈, 임기욱, & 이정현. (2009). 동시접속 사용자 접근을 고려한 데이터베이스 커넥션 풀 아키텍처. 한국콘텐츠학회논문지, 9(1), 89-97.

[3] 이호석. "Java 웹 서비스 아키텍처와 SOA 개념." 한국멀티미디어학회 학술발표논문집 (2006): 615-618.

[4] 김동훈, 홍경환, & 신동군. (2013). 임베디드 환경 Node. js 의 병렬화와 메모리 중복 제거 효과 분석. 한국정보과학회 학술발표논문집, 1290-1292.

[5] 김용훈. (2012). 공정한 오픈소스소프트웨어 활용을 위한 사용자환경 연구. 디지털정책연구, 10(1), 357-364.

[6] 박창근, 김기웅, & 유재형. (2010). 오픈 소스 소프트웨어 기반의 망관리 시스템 개발. KNOM Review, 13(1), 46-54.