Journal of Korea Spatial Information Society Vol.21, No.3 : 45-53, June 2013
http://dx.doi.org/10.12672/ksis.2013.21.3.045
ISSN 2287-9242(Print)
ISSN 2287-9250(Online)

Node.js를 활용한 웹GIS 서버의 설계와 구현

Design and Implementation of Web GIS Server Using Node.js

전 상 환* 도 경 태**
Sang Hwan Jun Kyoung Tae Doh

요 약 웹GIS는 수년 동안 사용자들에게 효율적이고 정확한 공간정보를 제공하기 위해 최신 웹기술을 기반으로 발전해왔다. 또한 웹GIS 서버는 클라이언트의 요청을 빠르게 연산 처리하고 공간정보 서비스를 제공하기 위해 성능개선을 지속해왔다. 본 연구에서는 서버 개발에 자바스크립트(JavaScript)를 사용하는 이벤트 기반의 비동기식 I/O 처리가 가능한 프레임웍 기술인 Node.js를 활용하여 NodeMap이라고 이름붙인 웹GIS 서버를 설계하고 구현하였다. NodeMap은 기본적으로 OGC 표준 인터페이스를 지원하는 웹GIS 서버이다. 이를 위해 공간 인덱스 및 표준 공간쿼리 함수를 지원하는 DBMS를 활용하여 GIS 데이터를 처리하도록 하였다. 그리고 공간 정보를 타일 맵 위에 렌더링 하기 위해 HTML5 Canvas를 지원하는 Node-Canvas 모듈을 활용하였다. 마지막으로 Node.js의 가장 많이 쓰이는 커넥트 모듈 기반의 프레임웍인 Express 모듈을 활용하였다. 구현된 NodeMap은 성능테스트를 통해 향 후 웹GIS 서버 개발기술로서 Node.js의 활용 가능성을 확인하였다. 본 연구를 통해 기존 서버 개발 기술과 차별화된 기술인 Node.js를 웹GIS 서버 구현에 우선적용 함으로서 향 후 인터넷 GIS 서비스에서의 활용 가능성을 제시하였다.

키워드 : 웹GIS 서버, Node.js, OGC, 이벤트 기반, 비동기식 입출력 모델

Abstract Web GIS, based on the latest web-technology, has evolved to provide efficient and accurate spatial information to users. Furthermore, Web GIS Server has improved the performance constantly to respond user web requests and to offer spatial information service. This research aims to create a designed and implemented Web GIS Server that is named as Nodemap which uses the emergent technology, Node.js, which has been issued for an event-oriented, non-blocking I/O model framework for coding JavaScript on the server development. Basically, NodeMap is Web GIS Server that supports OGC implementation specification. It is designed to process GIS data by using DBMS, which supports spatial index and standard spatial query function. And NodeMap uses Node-Canvas module supported HTML5 canvas to render spatial information on tile map. Lastly, NodeMap uses Express module based connect module framework. NodaMap performance demonstration confirmed a possibility of applying Node.js as a (next/future) Web GIS Server development technology through the benchmarking. Having completed its quality test of NodeMap, this study has shown the compatibility and potential for Node.js as a Web GIS server development technology, and has shown the bright future of internet GIS service.

Keywords : Web GIS Server, Node.js, OGC, event-driven, non-blocking I/O model

1. 서 론

GIS의 활용분야는 최근까지 패러다임의 변화를 겪어 왔다. 과거 공공기관 또는 연구기관에서 국토관리를 위한 지도 제작이나 공간을 분석하고 연구하는 목적으로 주로 활용되었던 GIS는 최근 웹 기술을 통해 누구나 쉽게 접근할 수 있고 활용성 높은 공간정보를 공유하는 서비스가 보편화 되었다. 1990년대 후반에서 2000년대 초반에 등장한 인터넷 지도서비스는 당시 열악했던 플랫폼 성능 때문에

* Sang Hwan Jun, Doctoral Student. Dept. of GeoInformatic Engineering, Pusan National University. junsh.rock@gmail.com
** Kyoung Tae Doh, Senior engineer, CSP Lap, Samsung SDS. ktdo@samsung.com

일반 PC에서 조차 기능들이 제한적이었다. 이는 기본적으로 공간정보 원시데이터의 용량이 일반적인 정보 데이터에 비해 매우 크고 복잡하기 때문이다. 그러나 최근 초고속 통신망의 등장과 하드웨어 및 소프트웨어의 성능이 눈부시게 발전하면서 워크스테이션이나 PC에서뿐 아니라 스마트폰과 같은 모바일 기기에서도 쉽게 공간정보를 활용하고 있다. 또한, 1994년에 조직된 OGC는 지리공간 콘텐츠와 GIS의 웹 서비스에 관한 기술 표준화를 주도하면서 개방형 웹GIS 개발환경이 확산되고 있다.

웹에서 공간데이터를 처리하고 클라이언트에게 전송하는 것은 웹GIS서버에서 수행된다. 이처럼 일반적인 웹 기반 시스템에서 GIS는 별도의 전용 서버를 통해 서비스 되는데 만약 웹GIS서버의 성능이 떨어지면 시스템 전체의 성능이 떨어지는 것처럼 보일 수 있다. 때문에 여러 기업과 연구기관에서는 웹GIS 서버의 개발과 개발된 서버의 성능 향상을 위한 연구 노력이 지속되고 있다.

이에 본 연구는 급변하는 웹 환경에 빠르게 적응하고 끊임없이 출현하고 있는 최신 기술들을 선적용하는 것으로 기존 시스템보다 향상된 성능과 기술을 제공하기 위한 웹GIS 서버를 설계 하고 구현하고자 한다. 우선 구현하고자 하는 서버는 기존의 서버 구현기술과 차별화된 기술을 적용하여 OGC기반 인터페이스 표준 서비스를 지원하고 개방형 클라이언트 개발 환경을 제공하는 웹GIS 서버 엔진 구현을 목적으로 한다. 본 연구에서 구현할 서버는 2009년 Ryan Dhal에 의해 개발된 자바스크립트 기반의 차세대 웹 플랫폼 개발기술인 Node.js를 활용하여 구현하였다. 또한 구현된 웹GIS 서버의 이름을 NodeMap이라 정하였고 구체적으로 다음과 같은 특징을 가진다.

첫째, 구현된 NodeMap은 개방형 서버로 공간정보 처리에 OGC 표준 기반의 오픈 소스 클라이언트 개발환경을 제공해 준다. 또한, NodeMap은 스스로 웹서버 역할을 하기도 하면서 프록시서버 역할을 수행하면서 스스로 요청을 보내고 응답받아 처리할 수 있도록 하였다.

둘째, NodeMap은 Node.js 기반 기술을 사용하였기 때문에 개발언어로 자바스크립트(JavaScript)를 사용한다. 자바스크립트(JavaScript)는 클라이언트 개발에 주로 사용하는 언어로 대부분의 웹 개발자들이 익숙하게 사용하는 프로그래밍 언어이다. 이는 웹 시스템 설계의 유연함과 구현의 효율성을 제공한다.

셋째, NodeMap은 공간 쿼리와 공간 인덱스를 제공하는 DBMS에 공간 데이터를 로딩하기 위해 쉐이프 파일을 WKT, WKB 형태로 변환하고 DBMS를 통해 저장된 GIS데이터를 처리할 수 있다.

넷째, NodeMap은 이미지 처리에 HTML5 canvas를 지원하는 node-canvas모듈을 사용하여 GIS 데이터를 렌더링 할 수 있게 하였다.

2. 관련기술

2.1 Node.js의 개념 및 특징

2.1.1 Node.js의 개념

Node.js는 2009년 Ryan Dahl에 의해 개발된 자바스크립트 기반의 서버 개발을 위한 소프트웨어 이다. 자바스크립트는 주로 서버 사이드 개발에 활용되기보다는 클라이언트 사이드 개발언어로 웹 브라우저 애플리케이션 콘텐츠 개발에 주로 활용됐다. 왜냐하면, 자바스크립트가 실행속도에 있어 C나 C++과 같은 언어에 비해 느리다는 단점 때문이다[15]. 그러나 2008년 9월 구글사에서 크롬 웹 브라우저에 ECMAScript (ECMA - 262) 3rd Edition 규격의 C++기반 오픈 소스 JIT 가상 머신형식의 자바스크립트 엔진인 속칭 V8(V8 JavaScript Engine)을 탑재하면서 자바스크립트의 속도문제를 극복할 수 있는 전환기를 마련하였다. 그동안 웹 개발에 있어 자바스크립트를 웹 브라우저를 벗어난 곳에서 활용하려는 시도가 계속되어 왔는데 이런 점을 들어 Ryan Dahl은 V8 자바스크립트 엔진을 기반으로 구동하는 Node.js를 개발했다.

2.1.2 Node.js의 기술적 특징

Node.js의 가장 큰 특징은 이벤트 기반 비동기 입출력 방식을 제공한다는 것이다. 기존의 서버 엔진들은 동시 다발적인 요청들을 처리하기 위해 스레드를 생성한다. 서버 엔진에서 네트워크상 I/O 메소드 처리에서 스레딩은 웹 애플리케이션을 관리하는 가장 확실한 방법으로 간주하여 왔다. 특히 동기 방식의 스레드를 동시에 멀티로 생성해 메소드를 처리하는 멀티스레딩을 주로 활용하고 있다. 그렇지만 스레드를 개발하는 것은 과도하게 어려운 단점이 있다[8]. 그리고 서버 엔진들은 요청이 계속 증가할

수록 새로운 스레드를 추가하기 위해 메모리를 지속해서 할당해야 하므로 메모리 소모량이 폭증하게 되고 서버 엔진의 성능을 저하시키는 단점이 있다. 반면에 이벤트 기반의 비동기 방식은 네트워크 입출력 이벤트를 콜백함수를 통해 비동기 I/O 방식으로 처리하기 때문에 요청이 증가하더라도 서버 엔진의 성능이 저하되는 현상을 방지하는 효과가 있다. 또한, 메소드 생성과 이벤트 연결을 구현하는데 멀티스레딩 구현보다 비교적 쉬운 점이 큰 특징이다.

Node.js는 다양한 모듈과 객체들을 제공하고 있다. 기본적으로 Node.js는 내장 모듈을 제공하고 있고 외부 모듈을 설치해 사용하는 것도 가능하다. 다양한 모듈을 제공한다는 점은 프로그래밍 개발에 있어 효율성과 확장성을 제공한다는 점에서 매우 큰 장점이 될 수 있다.

이런 점에서 Node.js는 확장성 있는 네트워크 환경을 간편하게 구축할 수 있는 매우 유용한 개발 환경을 제공한다.

2.1.3 Node.js의 활용 사례 및 장단점

Node.js는 신생 개발환경으로 세상에 소개 된 지 오래되지 않았지만, 점점 많은 인터넷 서비스 분야에서 활용하려는 시도가 늘어나 새로운 서버개발 대안 기술로 주목받고 있다. 실제로 여러 인터넷 서비스 관련 기업들이 Node.js를 사용하고 있다. 대표적으로 세계적인 소프트웨어 업체 마이크로소프트사는 자사 클라우드 서비스인 Windows Azure서비스에 Node.js를 사용하고 2011년 12월에는 SDK를 배포했다[15]. 또한 eBay, LinkedIn, Yahoo와 같은 대기업에서도 일부 또는 모든 서버구성에 Node.js를 사용하였다. 그리고 UBER, Transloadit사 역시 Node.js를 사용하고 있는 대표적인 기업이다[11]. 이처럼 주요기업들이 인터넷 서비스를 위한 서버구성에 Node.js를 활용하여 구축하였고 대규모 네트워크 통신을 원활하게 처리하고 있다는 점, 또한 구글이 Node.js에서 활용되는 V8엔진을 향 후 지속적인 성능 향상을 시도하고 있다는 점에서 점점 많은 분야와 기업들이 사용하려는 요구가 늘어날 것으로 예측된다.

Node.js의 가장 대표적인 장점은 비동기식 I/O 방식을 추구한다는 점이다. 실제로 Node.js에서는 요청과 응답 사이에 대기시간이 없다. 요청이 발생하면 Node.js는 응답이 도착할 때까지 대기하지 않고 다른 요청들을 계속 실행하는데 해당 요청에 대한 응답이 도착하면 곧바로 이벤트를 발생시켜 정보를 처리하기 때문에 시간적으로나 메모리 효율에서 큰 장점을 있다.

Node.js의 두 번째 장점은 자바스크립트를 활용한 개발이 가능하다는 점이다. 자바스크립트는 그동안 주로 클라이언트 사이드 개발언어로 활용되는 것이 당연시 여겨져 왔고 자바스크립트를 소개하는 책자에서나 개발자들 사이에 클라이언트 사이드 개발언어로 소개되고 인식됐다. 그러나 Node.js를 사용해 서버개발에 자바스크립트를 활용한다는 것은 개발자가 클라이언트, 서버 사이드 양쪽을 프로그래밍 언어에 대한 제약 없이 익숙한 언어와 기술로 자유롭게 컨트롤 할 수 있다는 점에서 상당히 큰 장점이라고 할 있다.

Node.js의 활용에 장점만 있는 것은 아니다. V8 엔진을 사용했다고 하지만 C나 C++로 구현된 서버에 비해 절대적으로 빠르다고 할 수는 없다. 속도의 약점을 극복하기 위해 개발자는 설계단계와 코딩단계에서 이런 부분을 감안하여 개발할 필요가 있다. 그리고 Node.js는 공식 공개된 최신 버전이 v0.10.1로 아직 Node.js를 활용하는데 참고할 레퍼런스나 모듈들이 부족하기 때문에 실제 프로젝트에서 개발에 어려움이 있을 수 있다[13]. 신생개발환경이라는 단점에도 Node.js가 업계에서 폭발적으로 관심을 받고 있고 향 후 많이 활용된다면 자연스럽게 다양한 모듈들이 개발될 것이고 기술들이 축적되면서 레퍼런스도 부족하지 않게 될 것이다. 실제로 Node.js에 관한 정보들이 빠르게 확산하고 있고 모듈들도 증가하는 추세로 여러 저서와 블로그, Node.js 공식 홈페이지(http://nodejs.org)에 정보들이 지속적으로 업데이트 되고 있다.

3. 최근 연구 동향

최근 GIS 관련 연구는 공간정보의 인터넷 서비스가 활성화되면서 관련 연구도 점점 증가 추세에 있다. 그 중에서도 OGC표준 적용에 관한 연구와 이를 기반으로 시스템을 설계하거나 구현하고자 하는 연구들이 최근 많이 이루어지고 있다. 그러나 본 연구에서 활용된 기술인 Node.js를 웹GIS분야에 활용하려는 시도는 거의 없을 뿐 아니라 관련 연구도 미미하다. 하지만 Node.js가 최신기술이지만 유수의

기업들이 이미 서버개발에 활용하고 있고 국내외에서 연구와 관련 보고서, IT 관련 블로그 등을 통해 소개되면서 많은 관심을 불러일으키고 있다.

기본적으로 서버를 설계하고 구현하는 데 있어 가장 먼저 고려해야 할 점은 서버의 성능이다. Node.js는 성능이 완전하게 검증된 기술은 아니다. 이 때문에 최근에 수행된 관련 연구들은 주로 기존 서버들과 성능비교를 위한 테스트를 실험하거나 활용을 위한 설계 구현에 관한 연구들이 주를 이룬다. 우선 연구[8]에서는 Node.js와 Aphache Server, EventMachine에 동시다발적인 요청을 낮은 레벨에서 점차 늘려가는 테스트를 통해 기존 서버와 Node.js로 개발된 서버의 성능을 비교 분석하였다. 그리고 Node.js를 실제 프로젝트에 적용하여 실제로 시스템을 구현하는 것으로 높은 효용성 또는 활용성을 갖춘 시스템을 설계 구축[2,8,13,14]하기 위한 방안을 제시하는 연구 사례도 있다. 그러나 대부분 서버들이 C++ 또는 자바(Java)로 개발되었고 관련 연구[4]에서도 웹 맵 서버 구현에 이런 프로그래밍 언어들을 활용했다.

최근 연구에서 웹GIS 서버들은 개방형 구조를 추구하고 있다[3,6,7,10]. 개방형 구조의 서버들은 주로 OGC 표준 인터페이스 채택하는 것으로 시스템에 구애받지 않고 클라이언트에 개방형 개발환경을 제공해준다. 이를 통해 확장성도 보장할 수 있기 때문이다. 개방성과 확장성은 공간정보 서비스 개발에 비용절감과 유연함을 제공하는데 비해 상업적 목적으로 설계 구현된 GIS 시스템은 너무 크거나 사용하기 어렵고 비용이 매우 비싸질 수 있다는 단점이 있다[4].

본 연구는 기존의 연구에서 제시된 개방형 서버 구조와 Node.js의 장점을 극대화 할 수 있는 최적화된 설계방안을 적용하여 NodeMap을 구현할 것 이다.

4. NodeMap의 설계와 구현

NodeMap은 GIS 데이터처리, 이미지 처리, 클라이언트 요청에 대한 응답 처리의 3부분으로 나누어 설계하고 구현되었다. 사용자에게 GIS 데이터를 보여 줄 때 NodeMap은 DBMS에 저장된 공간 데이터를 추출한 후에 모듈을 통해 이미지 처리를 수행한다. 또한 클라이언트의 요청에 응답 처리 하여 최종 서비스를 제공해 주는 절차로 서버를 구현했다. 구현절차는 아래와 같다.

4.1 GIS 데이터 처리

GIS 데이터는 여러 가지 방법으로 저장되고, 저장되는 포맷도 다양하다. 기본적으로 NodeMap은 GIS 데이터를 DBMS에 저장하기 위한 데이터 변환 모듈을 구현하여 다양한 DBMS를 지원할 수 있도록 설계하였다. 본 연구에서는 기능 단위 테스트를 위해 먼저 오픈소스 기반 DBMS인 Postgres의 PostGIS를 선택했다. PostGIS는 점형, 선형, 면형에 대한 공간 인덱스는 물론 표준 공간함수를 지원해주기 때문이다. 게다가 GeoJSON, WKT, WKB, GML 등의 다양한 형태의 데이터 아웃풋을 지원해준다.

Node.js의 모듈 중 PostGIS의 메소드에 쉽게 접근할 수 있는 pg 모듈을 npm 레지스트리에서 다운받아 설치하고 NodeMap에 삽입하여 활용하였다. NodeMap은 pg모듈을 통해 Node.js 와 PostGIS 간 쿼리를 받아서 데이터를 가져올 수 있도록 구현했다. 명령어는 다음과 같다.

```
> npm install pg
```

모듈을 설치한 후 필요한 데이터 입력을 위해 쉐이프 데이터를 입력하는 모듈을 개발해서 레이어에 데이터를 입력하고, GeoJSon으로 레이어에서 데이터를 얻어오도록 구현 했다. Figure 1은 pg모듈을 통해 DBMS와 통신해 공간쿼리를 수행하는 것을 나타낸다. 아래 소스는 레이어에서 GeoJSON형태로 데이터를 처리하는 부분의 소스이다.

```
var select = exports.select = function select( layer ,bounds, callback ){
  var query = "SELECT ST_AsGeoJSON("+geometryColumnName+")";
  query = query + " AS geometry FROM "+layer.name;
  query = query + " WHERE "+geometryColumnName+" && ";
  query = query + "ST_MakeEnvelope("+ bounds.toBboxString() + "
                ,3785)";
  client.query(query, function(err, result) {
  if(err){
    console.log( err );
  }else if(typeof result == "object"){
    callback(result.rows);
  }else{
    console.log("unexpected!",err);
  }
 });
};
```

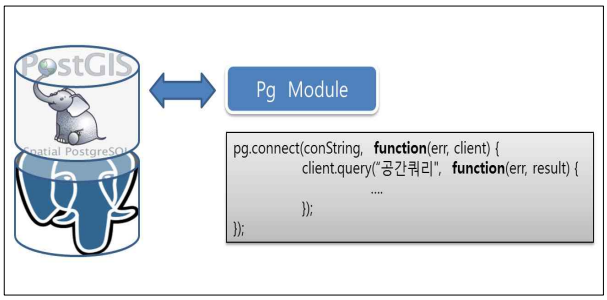

Figure 1. Pg module for GIS data processing

4.2 이미지 처리

이미지 맵 서버를 구축하기 위해서는 가져온 GIS 데이터를 이미지 위에 렌더링해야 하는데, 이를 위해 Node-Canvas모듈을 사용했다.

Node-Canvas 모듈은 HTML5의 canvas 스펙[5]을 준수하고 자바스크립트(JavaScript)로 작성된 명령어를 인식하여, 이미지 객체를 렌더링해 주는 역할을 한다. Node-Canvas 모듈은 오픈소스 그래픽 렌더링 패키지인 cairo[1]를 사용해 만들어져 있다. 아래의 소스는 레이어 렌더링을 구현한 소스이다.

```
Layer.prototype.draw = function( rows, bounds, callback ){
  map = this.map;
  map.context.scale( 1/map.resolution, -1/map.resolution );
  map.context.translate( - bounds.minx,
                - bounds.miny - map.canvas.height*map.resolution);
  this.conditionSize--;
  if(!this.styler){
    console.log("Styler has not initialized!");
    return false;6
    var ri = rows.length
    while(ri--){
      var row = rows[ri];
      var conditionId = row["conditionid"];
      var brush = this.styler.getBrushByCondition( conditionId );
      brush.draw( map, row["geometry"] );
    }
  if(!this.conditionSize){
    this.emit("drawEnd", rows, callback );
  }
};
```

Node-Canvas를 사용한 이유는 앞으로 자바스크립트(JavaScript) 표준 스펙으로 언급이 되고 있는 HTML5를 따르고 있기 때문이며, HTML5 canvas를 사용하는 것은 표준 행렬 이미지 렌더링 방식을 지원하고 있어 좌표변환이 쉽기 때문이다. DBMS에서 추출해온 GIS 데이터는 행과 열의 값을 분석해서 각 행의 데이터의 좌표값을 Node-Canvas 모듈을 통해 이미지 객체로 저장한다. 이렇게 저장된 이미지 객체는 HTML5의 PNG 스트림 형태를 제공하기 때문에 바로 웹으로 전송 가능한 형태가 된다. 이것이 Node-Canvas를 채택하게 된 또 하나의 이유이다.

4.3 웹 응답 처리

그냥 함수로써, 혹은 패키지로써 DBMS를 처리하는 데이터 처리 방법과 이미지 프로세싱만으로도 충분히 활용가치가 있다고 볼 수 있으나, Node.js 큰 장점 중 하나인 웹 환경에서 NodeMap이 구현될 수 있도록 웹 응답처리를 구현했다.

Npm 레지스트리에 등록된 모듈 중 가장 잘알려진 웹 서비스 모듈인 expressJS를 사용했다. expressJS는 웹 요청을 받아서 응답을 해주는 역할을 하고 있는데 Node.js 자체의 웹서버 기능을 제공하고 있고 구현하는 것도 가능하지만, RestFull 서비스라든지, 간단하게 웹 모듈을 구성하기 위해 expressJS가 가장 적합한 모듈이다.

4.4 전체 절차

기본적인 웹 상태에서 클라이언트는 지도 바운더리를 찾아서 NodeMap에 넘겨주게 되고 NodeMap은 그 요청을 다음과 같은 순서로 처리한다.

첫째, 지도바운더리와 지도 바운더리를 이용해 계산한 값을 이용해 각 행렬의 값들을 추출한다.

둘째, 요청에 해당하는 공간 쿼리를 DBMS(PostGIS)에 요청하고 쿼리에 대한 결과 값은 GeoJSON 형태로 데이터를 받는다.

셋째, GeoJSON형태로 받은 데이터값을 각 행별로 혹은 각 피쳐별로 분석한 후에 추출된 값과 좌표 차이 값을 이용하여 렌더링한다.

넷째, 렌더링이 끝나면 Node-Canvas의 canvas 객체에서 PNG 스트림을 추출한다.

마지막으로 웹 응답에 추출된 이미지 스트림을 제공하고 브라우저를 통해 GIS 데이터가 렌더링 된 것을 확인할 수 있다.

Figure 3은 구현된 NodeMap의 전체 시스템 개요 구성도이다. 구성도에는 구현된 NodeMap의 부분별 기능과 해당 모듈을 나타내고 있다. Figure2는 구

현된 서버에 실제로 웹브라우저를 통해 클라이언트가 NodeMap에서 DBMS에 저장된 지도 데이터를 요청하여 타일지도 위해 디스플레이 한 것이다.

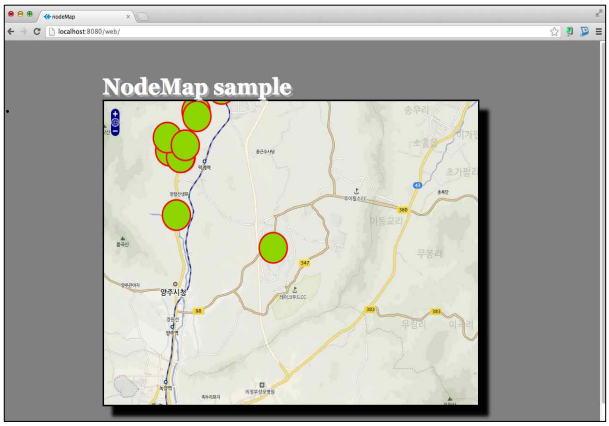

Figure 2. NodeMap map view

5. 성능 테스트

본 연구에서 구현한 NodeMap은 기존의 서버 엔진들과 차별화된 서버 엔진이다. 기존의 서버들은 스레딩을 통해 I/O를 처리하는 반면 NodeMap은 Node.js의 가장 큰 기술적 특징 비동기식으로 I/O를 처리하기 때문에 대규모 리퀘스트 처리에 적합하다. NodeMap의 향 후 활용 가능성을 검증하기 위해 성능테스트 벤치마킹 도구인 NodeLoad를 활용해 현재 국내의 웹GIS 관련 공공 프로젝트에서 가장 보편적으로 사용되고 있는 상용 웹GIS 엔진 후보군 4개 중 본 연구에서 비교테스트 할 수 있는 환경이 구축된 A서버와 B서버를 NodeMap과 성능 비교 테스트를 수행했다. A서버는 C++, B서버는 자바(Java)로 구현된 서버 엔진이다. 테스트 방식은 가상 유저를 20, 50, 100으로 생성하여 20분씩 동일한 공간 데이터 조건의 WMS의 GetMap리퀘스트를 테스트하였다. 아래의 그림과 표는 NodeLoad 테스트 결과로 추출된 결과 리포트이다. 그래프의 x축은 응답시간을 나타내고 최하위 그래프는 최소 응답시간 최상위 그래프는 최대 응답시간을 나타낸다.

우선 NodeMap의 테스트 결과를 살펴보겠다. Figure 4는 NodeMap의 20유저 테스트 결과 그래프로 최하위 최소 응답시간 그래프와 최상위의 최대 응답시간 그래프가 거의 일정한 그래프를 나타내고 있다. 이는 응답처리 속도가 일정하다는 것을 보여주는 것이다.

Figure 5는 NodeMap의 50유저 테스트 그래프이고 Figure 6은 100유저 테스트 그래프이다. 50유저 테스트에서 비교적 안정적인 그래프 곡선을 나타냈지만, 간헐적으로 그래프가 급상승하면서 평균 응답시간이 증가하는 구간이 나타나기도 했다. 반면에 100

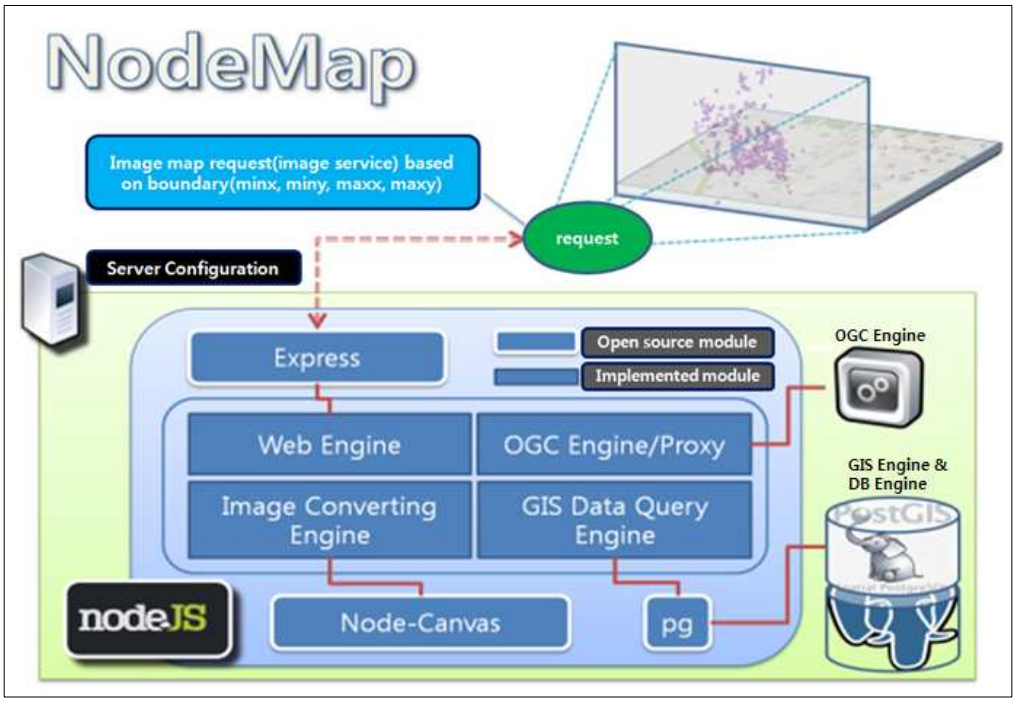

Figure 3. NodeMap Overview

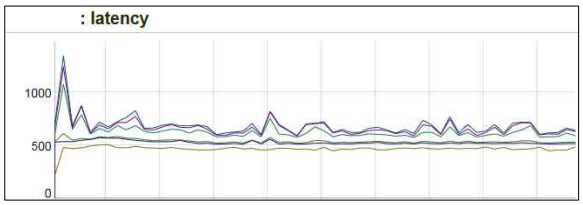

Figure 4. 20user latency of NodeMap

유저 테스트에서 이와 정반대로 그래프의 큰 변화 없이 테스트가 완료 될 때까지 일정한 응답시간을 보여줬다.

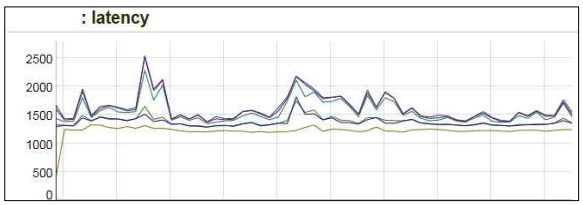

Figure 5. 50user latency of NodeMap

100유저 테스트 그래프가 20유저와 50유저 테스트 그래프에 비해 평균 응답시간은 늘어났지만 그래프에서 보듯 응답처리에 있어 요청의 증가에도 안정적인 성능을 확인할 수 있다.

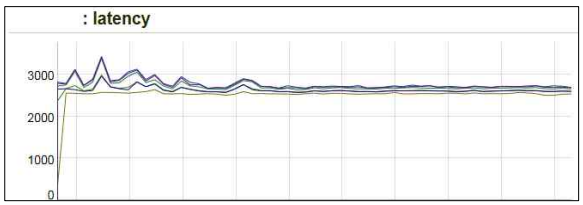

Figure 6. 100user latency of NodeMap

Table 1은 테스트에서 도출된 리포트의 응답시간 결과를 추출한 것이다. 20유저에서 50유저로 요청이 증가될 때 평균응답시간이 157.2% 늘어났고 100유저로 요청이 증가했을 때 83.8%의 증가율을 나타냈다. 이는 Node.js의 비동기식 I/O처리 방식이 동시다발적인 요청들에도 큰 성능저하 없는 정상적인 작동을 확인 할 수 있다.

Table 1. Surveying result of NodeMap's latency

	20user	50user	100user
LATENCY MIN	0.208sec	0.377sec	0.255sec
LATENCY MAX	1.337sec	2.539sec	3.439sec
LATENCY AVG	0.536sec	1.379sec	2.635sec
LATENCY MEDIAN	0.521sec	1.351sec	2.616sec

Figure 7, 8, 9는 C++기반 엔진 A서버의 테스트 결과 그래프이다. A서버는 NodeMap과 마찬가지로 요청이 증가하더라도 안정적인 그래프 곡선을 보여줬다. 하지만 Table2의 응답시간 결과표에서 NodeMap에 비해 평균 응답시간이 조금 높게 나타났고 20유저에서 50유저와 100유저로 늘려가는 테스트의 누적된 결과 응답시간 증가율이 153.8%와 96.4%로 NodeMap과의 비교해 100유저 테스트에서만 다소 응답시간이 크게 늘어난 것을 확인 할 수 있다.

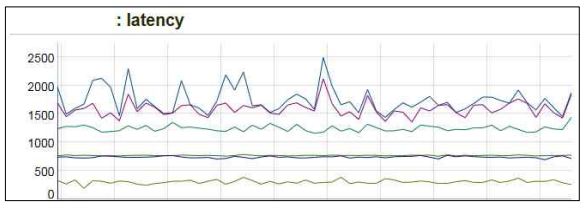

Figure 7. 20user latency of Engine A

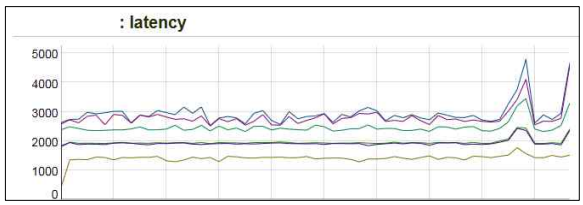

Figure 8. 50user latency of Engine A

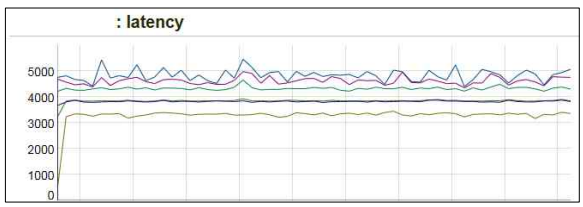

Figure 9. 100user latency of Engine A

Table 2의 응답시간 결과표는 A서버의 누적된 응답시간 결과이다. A서버는 이미 성능이 검증되어 국내 여러 웹GIS관련 시스템에서 활용되고 있는 서버이다.

Table 2. Surveying result of engine A's latency

	20user	50user	100user
LATENCY MIN	0.195sec	0.410sec	0.386sec
LATENCY MAX	2.490sec	4.791sec	5.454sec
LATENCY AVG	0.770sec	1.955sec	3.841sec
LATENCY MEDIAN	0.740sec	1.916sec	3.821sec

Figure 10, 11, 12은 Java기반 엔진인 B서버의 테스트 결과 그래프이다. B서버의 경우 20유저 테스트는 비교적 원활한 응답처리와 안정적인 성능을 보여줬다.

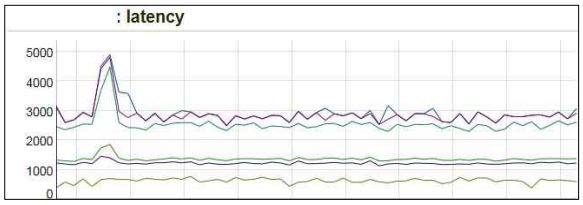

Figure 10. 20user latency of Engine B

하지만 Figure 11의 그래프에서 B서버가 유저의 수가 50으로 증가되자 간헐적인 응답처리 실패와 성능저하로 인한 그래프의 급격한 변화를 보여줬다.

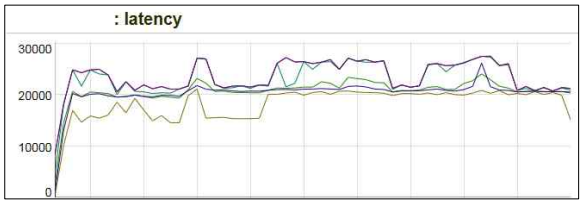

Figure 11. 50user latency of Engine B

그리고 Figure 12, 13의 그래프에서 B서버가 100유저 테스트 대부분의 요청에 응답처리를 하지 못하고 성능저하로 인한 비정상적 서버상태를 그래프를 통해 확인 할 수 있다. Figure12에서 박스에 표시된 그래프 구간은 서버가 다운되지 않았지만 모든 요청들을 처리하지 못하고 서버 대기상태에서 이전에 처리하지 못한 요청들을 넘겨버리고 잠시 후 다시 클라이언트에서 넘겨진 요청들을 처리해버리는 것을 그래프를 통해 확인 할 수 있다. A와 B서버는 기본적으로 스레드 생성을 통해 동시다발적인 요청들을 처리하는 서버들이다.

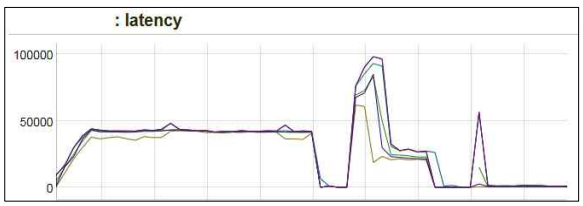

Figure 12. 100user latency of Engine B

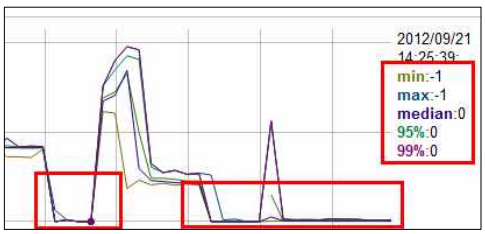

Figure 13. Engine B's 100 user test failed to respond

Table 3은 B서버 응답시간 결과표로 50유저와 100유저 테스트 응답시간 결과가 20유저 테스트에 비해 매우 크게 증가했고 100유저 테스트의 경우 서버의 정상적인 운영이 어려울 정도로 응답처리에 실패했기 때문에 응답시간 결과가 무의미 하다.

Table 3. Surveying result of engine B's latency

	20user	50user	100user
LATENCY MIN	0.378sec	0.219sec	1sec
LATENCY MAX	4.898sec	27.55sec	98.05sec
LATENCY AVG	1.363sec	20.51sec	0.729sec
LATENCY MEDIAN	1.215sec	20.75sec	4sec

3개의 웹GIS서버의 동일한 조건에서의 성능비교 테스트 결과 NodeMap의 성능이 빠른 응답시간 속도와 지속적으로 증가되는 요청을 안정적으로 처리할 수 있는 성능을 보여줬다. 이는 Node.js로 구현된 NodeMap이 동시다발적 요청이 증가할수록 성능의 우수함이 돋보이는 점을 확인할 수 있다.

4. 결 론

본 연구는 Node.js를 활용해 개방형 웹GIS 서버 엔진 NodeMap을 구현하였다. 일반적으로 기존 서버 엔진들은 스레드를 생성하는 것으로 동시다발적인 요청들을 처리한다. 그러나 지속해서 유입되는 다량의 요청들을 처리하기 위해 스레드를 계속 생성하게 되면 메모리의 한계에 도달하게 되고 이는 서버 엔진의 순간적인 성능저하를 초래하게 된다[12]. 이에 반해 Node.js로 구현된 NodeMap은 다량의 요청들이 유입된다 하더라도 스레드 방식과 달리 이벤트 기반의 비동기식 I/O 처리를 통해 서버 엔진의 메모리 할당을 최소화하고 서버의 성능저하를 최소화할 수 있는 장점이 있다.

구현된 NodeMap은 웹GIS 서버 엔진으로 공간데이터를 처리하기 위해 공간 인덱싱 및 공간 쿼리를 지원하는 DBMS와 GeoJson, WKT, WKB, GML과 같은 OGC 표준 형태의 데이터 아웃풋을

지원한다. 그리고 지도를 표현하기 위해 표준 행렬 이미지 렌더링 방식의 cairo모듈을 사용하였다. cairo모듈은 Node-Canvas 모듈로서 오픈소스 그래픽 렌더링 패키지로 최근 주목받고 있는 HTML5의 canvas표준 스펙을 준수하고 있다. 또한, NodeMap의 응답처리를 위해 Node.js에서 가장 많이 활용되고 있는 express모듈을 활용하여 웹 요청을 처리하고 응답하는 역할을 하도록 구현하였다. 구현된 NodeMap은 대표적인 국내 웹GIS 서버 엔진들과 성능 비교를 위한 벤치마킹 테스트를 수행하였다. 이를 통해 NodeMap의 향 후 활용 가능성을 확인할 수 있었다.

Node.js는 비록 최신 기술로 아직 검증되지 않은 부분도 있지만 최근 IT 분야에서 많은 관심을 불러일으키고 있고 여러 기업에서 서버 구축에 도입하려는 시도가 활발해지고 있다. 그럼에도 아직 서버 구현 활용 기술로서 향 후 기존 서버개발 기술을 대체할 확실한 하나의 기술이라고 단정하고 예측하기는 어렵다. 그리고 웹GIS 서버 엔진 개발에 소수 개발자의 개인적 관심을 제외하면 연구사례 또는 활용사례가 없다고 할 수 있다. 그러나 웹GIS가 더 나은 지도 서비스를 제공하기 위해 검증되고 일반적인 웹 기술에만 의존하기보다는 Node.js와 같은 새로운 기술에 대한 적극적인 융합과 지속적인 연구를 통한 선제적 기술 도입으로 급변하는 웹 환경에 더욱 빠르게 적응하고 진보해 나갈 수 있는 발판을 마련할 필요가 있다.

References

[1] Cairo, 2012, downloaded node-canvas module http://www.cairographics.org/.

[2] Contreras, S. A, 2011, An Application Framework for High-Available Systems in Node.js, Master of Science Thesis Stockholm.

[3] Cho, D. S; Park, J. H, 2002, Design and Implementation of Open Web Map Server, Journal of Korea Information Processing Society, 9(6):981-990.

[4] Fernandez, P; Bejar, R; Latre, M.A; Valino, J; Banares J. A; Muro-Medrano P. R, 2000, Web mapping interoperability in practice, a Java Approach Guided Open GIS Web Map Server Interface Specification, Proceedings of the 6th EC-GI&GIS Workshop:The Spatial Information Society-Shaping the Future 2000.

[5] HTML5 Canvas, 2013, Last Updated 27 March 2013, http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html.

[6] Kang, J. H; Baek, I. G; Han, K. J, 2001, Design and Implementation of a Web Map Server based on GML, Paper presented at the autumn conference for Korean Institute of Information Scientists and Engineers, 28(2):88-90.

[7] Lee, H. J; Jun, B. G; Hong, B. H., 2000, Design of Web Map Server for supporting Fusion Service, Journal of Korea Spatial Information System Society, 3(2):109-122.

[8] McCune, R. R , 2011, Node.js Paradigms and Benchmarks, STRIEGEL, GRAD OS F'11, PROJECT DRAFT.

[9] Ministry of Land, Transport and Maritime Affairs, 2010, Design and Implementation of Open API for KOPSS GIS Engine, P.11-16.

[10] Nam, K. W; Ha, S. W, 2009 Awarematics/WMSServer : Design and Implementation of a Open Source Web Map Service Server, Journal of Korean Spatial Infomation System Society, 11(3):70-72.

[11] Node.js, 2013, Homepage of node.js, www.nodejs.org.

[12] Ousterhout, J, 1995, Why Threads are a Bad Idea(For most purpose), talk given at USENIX Annual Conference.

[13] Paudyal, U, 2011, Scalable web application using node.js and CouchDB, Uppsala University,

[14] Tilkov, S; Vinoski, S, 2010, Node.js:Using JavaScript to Build High-Performance techniques, The Sixteenth IEEE symposium on Computers and Communications. http://uu.diva-protal.org/samsh/record.jsf?pid=diva2:443102.

[15] Yoon, I. S, 2012, Node.js Programming for Modern Web, Hanbit Media.

논문접수 : 2012.11.19
수 정 일 : 1차 2013.03.29 / 2차 2013.04.26
심사완료 : 2013.06.20