3차 보고서

개발 환경

**9 조**

**주제** 지도 기반 채팅 커뮤니티 시스템

**팀명** CEO

**팀원** 박은영(32151839)

**보고서 주제** 개발 환경

앞서 1차 보고서에서 주제에 관하여 소개를 하였다. 1차 보고서의 아이디어를 바탕으로 구현계획과 필요 기술들을 3차 보고서에서 다루고자 한다.

1. 서비스 범위

서비스를 하기 위해서는 아래 4가지 크로스 플랫폼을 구현하여야 한다. 하지만 시간 비용상 이번 프로젝트에서는 크롬 웹 브라우저 환경, 특히 요즘 상용화된 1920 \* 1280 해상도의 서비스만 구현할 것이고, 추후 설계 과목 종료 후 2. ~4. 를 구현 할 계획이다.

모바일 환경은 데스크톱 환경보다 더 구현할 수 있는 범위가 크다. 예를 들어 블루투스로 친구등록을 할 수 있는 기능은 모바일 환경에서만 구현을 할 수 있기 때문에 이 번 설계에서는 다루지 않는다.

1. 웹 브라우저 (크롬)
2. 안드로이드
3. 데스크톱 어플리케이션
4. IOS
5. 설계 준비물 및 사용방법

Raspberry Pi, 128GB SD카드

1. Raspberry Pi에 **Raspbian** OS 를 설치 후 MariaDB server 버전 Database를 구축한다.
2. 개발할 PC에서 원격으로 Raspberry Pi의 Database 서버에 접근 가능하며, node.js express Language를 사용하여 정보를 교환한다.
3. 개발 완료 후 소스 코드를 Raspberry Pi에 업로드 한 후 Raspberry Pi에서 node.js 와 vue.js의 서버를 실행한다.

128GB SD 카드, 라즈베리 파이의 kit에 16GB SD카드가 있지만 채팅서비스는 큰 서버가 필요할 것으로 예상하여 128GB SD카드를 사용할 것이다.

3. 웹 브라우저 (크롬)

1. Vue.js front-end framework for Single Page App
   1. 사용 이유
      1. 최신 트렌드의 front end 언어 중에서 가장 학습하기 쉽다.
      2. 컴포넌트

유지보수와 재 사용성을 높이기 위해 코드를 작은 단위로 쪼개어 관리한다. Angular, React도 컴포넌트 방식이지만 그 중 Vue는 단일 파일 컴포넌트 방식 으로 .vue 하나의 파일에 html, css, js을 작성할 수 있어 캡슐화를 제공하게 된다.

* + 1. 성능

Native App(어플리케이션: 안드로이드, ios)과 유사한 퍼포먼스로 UX를 확보(사용자 편리성) + 생산성이 뛰어난 개발 환경 구축

Single Page Application

index.html에 모든 내용을 구현

사용자의 요청에 따라 필요한 부분만 java-script를 통하여 렌더링 한다

각 tag(component)들과 필요에 따라 첨삭할 수 있다.

전체 트래픽 감소 효과

새로 고침을 하여 페이지를 로드 하면 트래픽이 발생하지만 새로 고침을 하지 않고 component들과 데이터들을 첨삭하는 방법을 사용하기 때문에 트래픽이 감소한다.

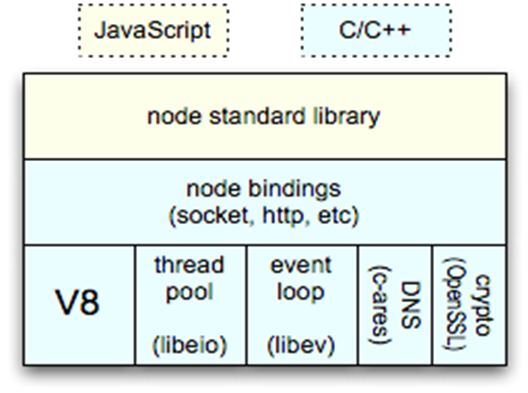
2. back-end - Node.js

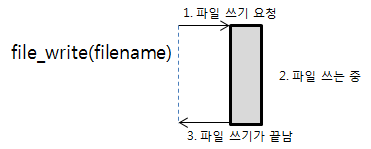
1. Node.js

개발 관점에서는 빠르고 쉬운 장점이 있지만, 반대로 운영 관점에서는 테스트, 장애 대응, 디버깅 등에 대해서는 신경 써야 할 부분이 훨씬 더 많다.

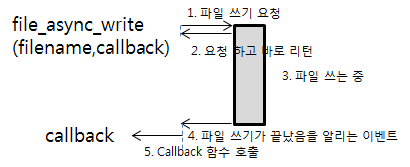
* 1. java-script의 생산성의 향상

HTML5 등으로 인해서 웹 개발에서 자바스크립트의 비중이 높아지게 되었고, 단순한 UI 수준만 필요하던 프론트엔드 개발자들에게도 수준 높은 개발 능력이 요구 되면서, 자바스크립트 기반의 프로그래밍 능력이 많이 향상되어 있던 차에, 백엔드에 대한 높은 접근 장벽이 있는 상태에서 node.js의 등장은 이러한 백엔드에 대한 진입 장벽을 깨버린 계기가 되었다. 자바스크립트 기술을 가지고 서버 백엔드를 개발할 수 있게 되었다.

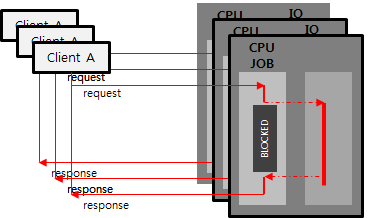
* 1. 사용이유 – 성능
     1. Node.js는 구글의 크롬 V8 자바스크립트 엔진을 기반으로 한, 고성능 네트워크 서버이며 수천 개 명령을 1초에 처리할 수 있다.
     2. node.js의 기본적인 구조인 Single Thread기반의 비동기 IO 처리에서 온다. 하나의 쓰레드가 request를 받으면, 처리를 하고, File IO나 Network 처리 (데이타 베이스 접근)등이 있을 경우에는 IO 요청을 보내 놓고, 작업을 처리하다가, IO 요청이 끝나면 이벤트를 받아서 처리하는 이벤트 방식을 사용한다. 이로 인해서, CPU가 IO 응답을 기다리는 시간이 필요 없고, 대부분의 연산 작업에 사용되기 때문에 높은 효용성을 가질 수 있으며, 특히 하나의 Thread로 여러개의 요청을 처리하는 구조로 되어 있기 때문에, 다수의 클라이언트를 동시에 처리할 수 있는가에 대한 의문인 C10K 문제를 처리할 수 있는데 아주 최적화 되어 있다.
     3. 멀티 쓰레드의 개념이 없다. 서버 프로그램에서 쓰레드 간의 동기화 처리 등이 중요하고 또한 복잡한점 중의 하나인데, 이러한 문제 자체를 제거함으로써 서버 프로그래밍 자체를 매우 단순하게 만들어 버린 것이다
     4. socket.io를 이용한 웹 push 구현이 매우 쉽게 구현이 가능하다.
  2. 단점
     1. Single thread 모델이기 때문에, 하나의 작업 자체가 시간이 많이 걸리면, 전체 시스템의 성능이 아주 급격하게 떨어진다. 그래서, 가벼운 (CPU를 많이 사용하지 않는) 작업 위주로 개발이 되어야 한다.
     2. 자바스크립트에서 오는 문제점은 자바나 다른 언어에 비해서 명시성이 떨어지기 때문에, 코드의 가독성이 자바언어에 비해서 상대적으로 낮기 때문에 유지 보수가 어려워질 수 있다.
     3. 자바스크립트와 같은 스크립트 언어의 특성상 해당 코드가 수행이 되어야 코드에서 에러가 나는지를 확인할 수 있고, 에러가 날 경우 프로세스 자체가 죽는다.
     4. single thread 모델이기 때문에, 멀티 코어 머신 에서 CPU 사용을 최적화할 수 없다는 문제가 있다. 그래서 설계 시, Cluster 모듈 등을 이용하여, 하나의 서버에서 여러 개의 노드 프로세스를 사용하는 모델을 가지고 가야 하며, 또한, 세션 등을 공유할 경우, 세션 공유용 redis 와 같은 부가적인 인프라가 필요하다.
     5. V8 엔진을 기반으로 하는데, 이 V8 엔진은 Garbage collection 기반의 메모리 관리를 하기 때문에, GC시 CPU 사용률이 Spike를 치면서 순간적으로 서버를 멈추게 할 수 있다는 문제점이 있다.
  3. node.js의 내부 작동 원리 구조
     1. 
        + Node.js는 Google의 Chrome V8 자바스크립트 엔진을 기본으로 동작한다. 이를 기반으로 Single Thread 기반의 Event Loop (libuv)가 돌면서 요청을 처리하며, 시스템적으로 non-blocking io를 지원하지 않는 io 호출이 있는 경우, 이를 비동기 처리 하기 위해서 내부의 Thread pool (libio)을 별도 이용하여 처리한다. 그 위에 네트워크 프로토콜을 처리하는 socket, http 바인딩 모듈이 로드 되고, 맨 윗단에, node.js에서 제공하는 standard library (파일 핸들링, console등)이 로드 된다.
     2. Async / Non blocking IO
        + 동기식 IO
          1. file write io를 예를 들어보면, file\_write를 호출하면, 디스크에 파일 쓰기 요청을 하고, 디스크가 파일을 쓰는 동안 프로그램은 file\_write 부분에 멈춰서 대기하게 된다. (블록킹상태). 파일을 쓰는 동안에는 CPU가 사용되지 않기 때문에, CPU는 놀고,파일이 다 써지만 디스크에서 리턴해서 file\_write 함수 다음 코드로 진행을 하게 된다

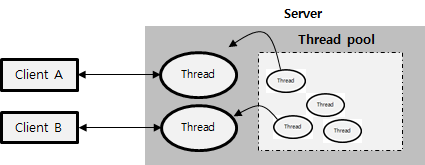


* + - * 비동기식 IO
        1. 파일 쓰기 요청을 할 때, 파일 요청이 끝나면 호출될 함수(callback)를 같이 넘긴다. 파일 쓰기 요청이 접수되면 프로그램은 파일이 다 써지는 것을 기다리지 않고, 요청만 던지고 다음 코드로 진행을 계속한다. 파일을 다 쓰고 나면 앞에서 등록했던 callback 함수를 호출하여 파일 쓰기가 다 끝났음을 알리고 다음 처리를 한다.

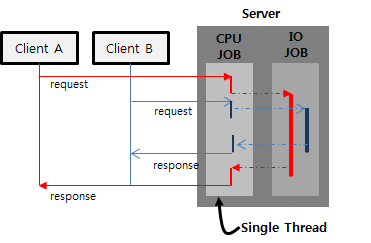


* + 1. Single Thread Model
       - Multi Process(Multi Thread) Tomcat, JBoss, Apache.
         1. Client에서부터 요청이 오면, Thread를 미리 만들어 놓은 Thread Pool에서 Thread를 꺼내서 Client의 요청을 처리하게 하고, 요청이 끝나면 Thread Pool로 돌려보낸 후, 다른 요청이 오면 다시 꺼내서 요청을 처리하게 하는 구조이다. 동시에 서비스 할 수 있는 Client의 수는 Thread Pool의 Thread 수와 같은데, 물리적으로 생성할 수 있는 Thread의 수는 한계가 있다. 예를 들어 Tomcat의 경우 500개 정도의 쓰레드를 생성할 수 있다. (물론 2,000개 정도까지도 생성할 수 있지만, 한계가 있다.) 즉 동시에 처리할 수 있는 Client 수에 한계가 있다. 또한 IO 효율면에서도 보면, 아래 그림과 같이 Client에 할당된 Thread는 IO 작업 (DB,Network,File)이 있을 경우 IO 호출을 해놓고, Thread는 CPU를 사용하지 않는 Wait상태로 빠져 버리게 된다.

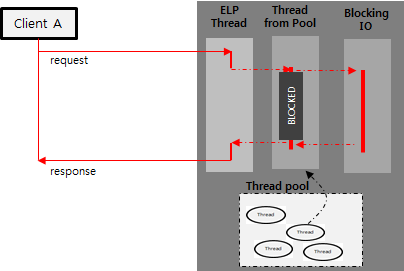




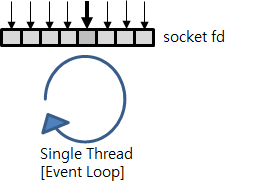
* + - * 이런 문제를 해결 하기 위한 것이 Single Thread 기반의 비동기 서버인데, 하나의 Thread만을 사용해서 여러 Client로부터 오는 Request를 처리한다. 단, IO 작업이 있을 경우 앞에서 설명한 비동기 IO방식으로 IO 요청을 던져 놓고, 다시 돌아와서 다른 작업을 하다가 IO 작업이 끝나면 이벤트를 받아서 처리하는 구조이다. 아래 그림 에서처럼, Client A가 요청을 받으면, CPU 작업을 먼저 하다가 IO작업을 던져 놓고, Client B에서 요청이 오면, CPU작업을 하다가 IO작업을 던져 놓고, Client A의 IO작업이 끝나면 이를 받아서 Client A에 리턴 하는 식의 구조이다. IO작업시 기다리지 않기 때문에(Block 되지 않기 때문에), 하나의 Thread가 다른 요청을 받아서 작업을 처리할 수 있는 구조가 된다. 이 요청을 받아서 처리하는 Thread를 ELP (Event Loop Thread)라고 한다.



* + 1. Thread pool
       - Node.js도 single thread만 사용하는 것이 아니라 내부적으로 multi thread pool을 사용하기는 한다. 예를 들어 file open등과 같은 일부 IO는 OS에 따라서 nonblocking function을 지원하지 않는 경우가 있기 때문에, 이러한 blocking io function을 호출할 경우에는 어쩔 수 없이 blocking이 발생하는 데, 이 경우 single thread로 구현된 event loop thread가 정지되기 때문에 이러한 문제를 해결 하기 위해서 내부적으로 thread pool을 별도로 운영하면서 blocking function call의 경우에는 thread pool의 thread를 이용하여 IO 처리를 하여 event loop thread가 io에 의해서 block되지 않게 한다.



* + 1. Event Loop
       - 하나의 Thread로 여러 클라이언트의 요청, 즉 여러 개의 socket connection을 어떻게 처리할까? 방법은 Multiplexing에 있다. 여러 개의 socket이 동시에 연결되어 있는 상태에서 하나의 Thread는 어느 socket으로부터 메시지가 들어오는 지 보다가, socket에서 메시지가 들어오면, 그 메시지를 꺼내 받아서 처리를 하는 방식이다.



* + - * single thread 모델에서 주의해야 하는 점은 CPU 작업이 길어질 경우에는 다음 request를 처리하지 못하기 때문에, 다음 request처리가 줄줄이 밀려버릴 수 있다는 것이다.
    1. node.js의 사용 용도
       - Async IO를 사용하기 때문에, file upload/download와 같은 network streaming 서비스에 유리하다.
       - real time web application, 채팅 서비스 같은 곳에 socket.io를 이용하면 쉽게 만들 수 있다.
       - Single page app 개발에 좋다. 가볍고 생산성이 높은 웹 개발 프레임웍을 가지고 있고, 간단한 로직을 가지면서 대용량 그리고 빠른 응답 시간을 요구로 하는 애플리케이션에 적절하다.

3. 사용 라이브러리 및 API

1. express.js

Express는 웹 페이지 개발 및 REST API 개발에 최적화된 프레임웍으로 매우 사용하기가 쉽다.

* app.get('/', function function(req,res){ });

Router는 특정 URL로 들어오는 HTTP Request를 처리하는 패턴이다.

1. express-mysql-session (Ver.2.0 추가예정)

A MySQL session store for express.js.

세션 id와 value를 database에 저장할 수 있게 해주는 라이브러리

1. Socket.io
2. Google maps