Virtual Whiteboard

Calvin Hamus
Justin Tolman
Milson Munakami
Patrick Lee

Agenda

- Objective
- Introduction / Key Concept
- Technologies Used
- Libraries
- API Descriptions
- Future Improvements
- Demo
- Q & A

What is Virtual Whiteboard ?

Online Interactive White Boards

Collaborating Tools

Real-time virtual rooms,
Text-chat and more.

Problem

- Geographical location and timing
  - Sharing knowledge
  - Interactive Teaching
  - Training and tutoring
  - Creative brainstorming
  - Sales and product demos
  - Interviews and tests

Solution

- Web-based online virtual whiteboard
- Real-time multi-user collaboration
- Text chat with users
- Run unlimited sessions

Technologies Used

- Dev environment setup using Vagrant and Puppet
- MySQL database
- Java REST API
- NodeJS server and load balancer
- Nginx load balancer
- Redis
- JavaScript client app
- SocketIO

Setup

- 1 load balancer
- 3 back-end Tomcat servers (Java API)
- 3 NodeJS servers
- 1 MySQL database (DB) server
- Tomcat servers

Client Side

Out client side implementation utilizes some of the latest web technologies.

HTML5, CSS3, and the latest JavaScript methods allow for much greater flexibility in what can be done within a browser. With canvas and SVG this includes image manipulation. The w3c is planning increased future integration of between them.

We use web sockets for real time interaction and AJAX style calls for database interactions.

Client Side Third Party Resources

jQuery
JavaScript's prototypical inheritance with the convenience of CSS selectors

jQuery-ui
Store state information for useful graphical tools.

normalize.css
Sets up a common starting place so your stylesheets will work the same in all browsers.

Spectrum
A color picker plug-in for jQuery

jQuery.contextMenu
A context based right click menu plug-in for jQuery. (We haven't actually implemented it yet,)

Graphics

Raster vs. Vector

Extensibility & Potential Improvements

We designed our application to be easily extended and added upon. Some additional potential uses are:

Additional drawing tools. The sky's the limit on this one, and with jQuery's plug-in architecture they can be added without modification to existing code in most cases.

File collaboration. With the addition of a file server we could easily include the ability to share files.

Slide Presentations. It would be fairly easy to save specific drawings as slides, and play them back

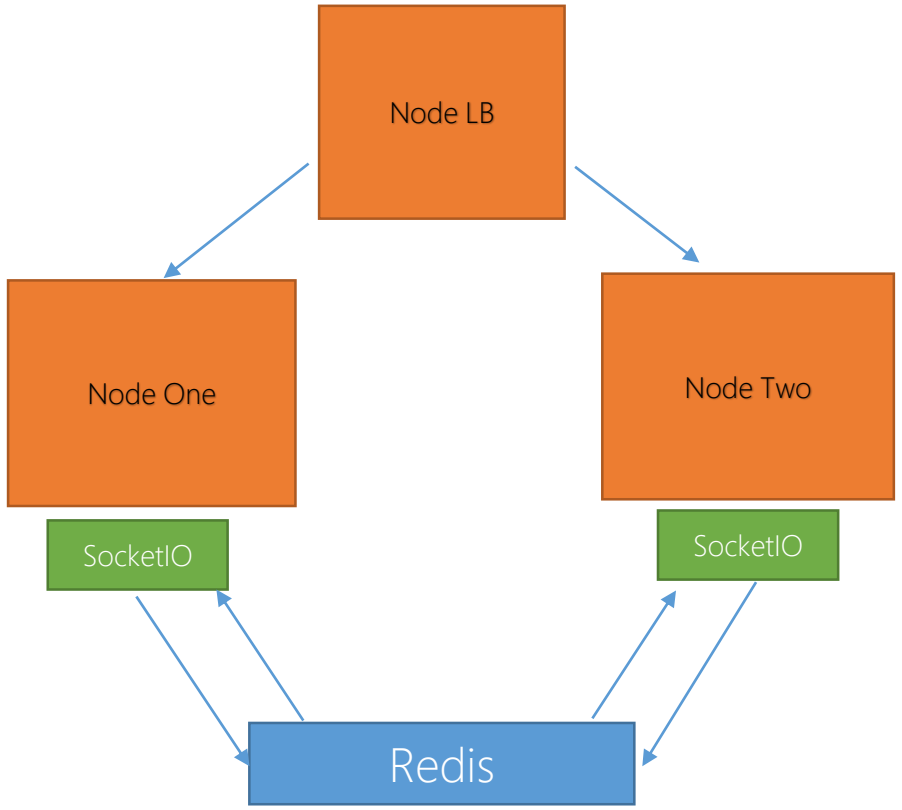
3D. WebGL and three.js are compatible with the technologies we're using, with some work collaboration on 3D projects like architecture, manufacturing, and 3D printing could be integrated.

NodeJs and SocketIO

- My Objective
  - Use a scripting language server (Node) to scale to a cloud based architecture.

- Challenge
  - Scale a system that does not want to scale nicely

Node Architecture



Node LB

```
//http://goldfirestudios.com/blog/136/Horizontally-Scaling-Node.js-and-WebSockets-with-Redis
var http = require('http');
var proxy = require('http-proxy');
//var request = require('request');

http.globalAgent.maxSockets = 10240;

// Define the servers to load balance.
var servers = [
{host: 'cs597-VirtualWhiteboardNodeOne', port: 3000},
{host: 'cs597-VirtualWhiteboardNodeTwo', port: 3000}
];
var failoverTimer = [];


// Create a proxy object for each target.
var proxies = servers.map(function (target)
{
        return new proxy.createProxyServer({
        target: target//,
        //ws: true
        });
});

/**
* Select a random server to proxy to. If a 'server' cookie is set, use that
* as the sticky session so the user stays on the same server (good for ws fallbacks).
* @param {Object} req HTTP request data
* @param {Object} res HTTP response
* @return {Number} Index of the proxy to use.
*/
var selectServer = function(req, res) {
var index = -1;
var i = 0;

// Check if there are any cookies.
if (req.headers && req.headers.cookie && req.headers.cookie.length > 1)
{
        var cookies = req.headers.cookie.split('; ');

        for (i=0; i<cookies.length; i++)
        {
                if (cookies[i].indexOf('server=') === 0)
                {
                        var value = cookies[i].substring(7, cookies[i].length);
                        if (value && value !== '') {
                        index = value;
                        break;
                        }
                }
        }
`
```

```
if (proxies[index].options.down)
{
        index = -1;

        var tries = 0;
        while (tries < 5 && index < 0) {
        var randIndex = Math.floor(Math.random() * proxies.length);
        if (!proxies[randIndex].options.down) {
        index = randIndex;
        }

        tries++;
        }
}

index = index >= 0 ? index : 0;

// Store the server index as a sticky session.
if (res) {
res.setHeader('Set-Cookie', 'server=' + index + '; path=/');
}

return index;
};

var server = http.createServer(function(req,res)
{
        var proxyIndex = selectServer(req, res);
        var proxy = proxies[proxyIndex];
        //console.log('blancing request to: ',proxy);
        proxy.web(req, res);
        proxy.on('error', function(err)
        {
                Console.log('proxy error',err);
        });
server.on('error',function(err){
        Console.log('server error');
})
});

// Get the next server and send the upgrade request.
server.on('upgrade', function(req, socket, head)
{
        var proxyIndex = selectServer(req);
        var proxy = proxies[proxyIndex];
        proxy.ws(req, socket, head);

        proxy.on('error', function(err, req, socket)
        {
        console.log(err);
                socket.end();
        //startFailoverTimer(proxyIndex);
        });
```

Pre socket-io-redis

```
-    //, redis = require('redis')
     , redis = require('socket.io-redis')
     , session = require('express-session')
     , cookieParser = require('cookie-parser');
@@ -12,73 +11,45 @@ var drawinginstructions = {};
 var clients = {};
 var messages = {};

-//var express = require('express');
 var app = express();
 var server = http.createServer(app);
 var io = require('socket.io').listen(server);
-//var http = require('http').Server(app);
-//var io = require('socket.io')(http);
 var log = {};
 log.users = [];
 log.messages = [];
 log.drawings = [];
 log.links = [];

-/*var RedisStore = require('connect-redis')(session),
-       rClient = redis.createClient(6379,'cs597-VirtualWhiteboardDB'),
-       sessionStore = new RedisStore({client:rClient});*/
-//app.set('port',process.env.PORT || 3000);
-//app.set('views',__dirname + '/www');
-//app.set('view engine','ejs');
-app.use(cookieParser('your secret here'));
-//app.use(session({store:sessionStore, key:'server', secret:'your secret here',resave:true,saveUninitialized:true}));
-
-
 app.use(express.static(__dirname + '/www'));
-//var SessionSockets = require('session.socket.io');
-//var sessionSockets = new SessionSockets(io, sessionStore, cookieParser, 'server');

-/**NEW REDIS CLIENTS**/
-//var sub = redis.createClient(6379,'cs597-VirtualWhiteboardDB');
-//var pub = redis.createClient(6379,'cs597-VirtualWhiteboardDB');
-var channel = "";
-//sub.subscribe('msg');
-//sub.subscribe('link');
-//sub.subscribe('draw');
```

Node Issues

- Load balance socketIO?
  - Redis

Post

```
/*SOCKET DATA*/
io.adapter(redis({host:'cs597-VirtualWhiteboardDB',port:6379}));
//io.adapter(redis({host:'localhost',port:6379}));
```

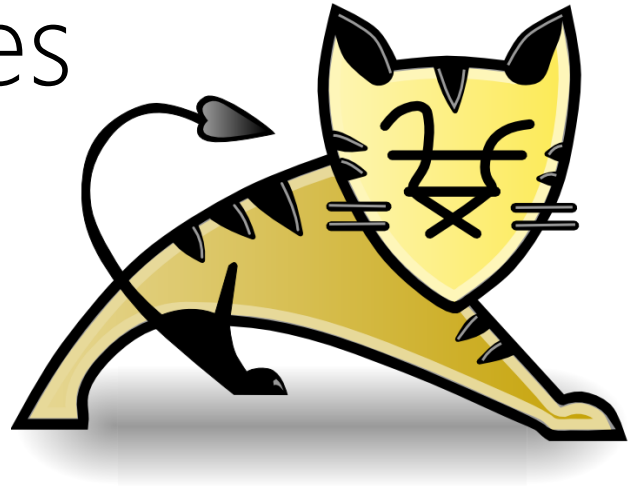
Node Issues Part 2

- SocketIo Rooms?
  - Solution – keep all changes in data base and populate room when new user joins room.

JAVA API: Requirements

- Save whiteboard changes and chat messages in real-time
- Send history of rooms to users upon entry
- Use flexible data format to support new features
- Handle write-heavy workload

Java API: Technologies



- Nginx load balancer
- MySQL database
- Tomcat app server
- Maven, Jersey, Jackson, Hibernate, C3P0

Primary challenge…

- connection pooling and performance under load

Java API: Examples

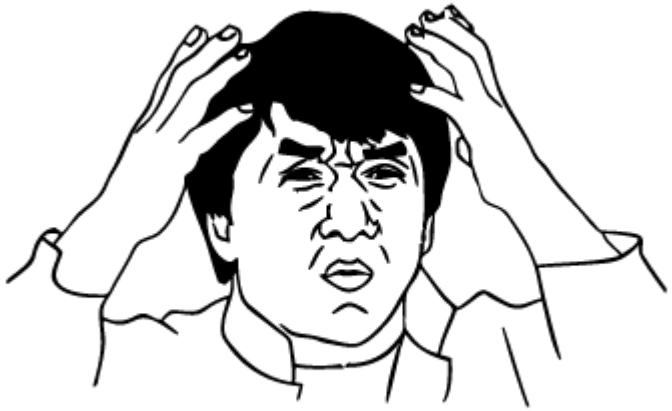
```
curl -X POST -H "Content-type: application/json" -d '{"action": "draw", "user": "Guest",
"data": "canvas|l1|line|523|342|523|341|black|1"}'
http://cs597-VirtualWhiteboardLB/whiteboard-api/room/updateboard/cs597/user/Guest

curl -X GET http://cs597-VirtualWhiteboardLB/whiteboard-api/room/cs597

{
   "id": 72,
   "chat": [ { "action": "message", "user": "Ronnie", "msg" :"Yeah buddy!" } ],
   “whiteboard”: [
      { "action": "draw", "user": "Guest", "data": "canvas|l1|line|523|342|523|341|black|1" },
      { "action": "draw", "user": "Ronnie", "data": "canvas|l1|point|480|312|red|1" }
   ]
}
```

Dev Setup: The problem?

- Applications can be complex and hard to set up manually
- Differences in base OS, software versions, config files…
- End up with “snowflake setups” (no two are alike)
- The dreaded “Works On My Box”



Dev Setup: The solution?

```
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "ubuntu/trusty64"

  dev.vm.provider "virtualbox" do |vb|
    vb.memory = 1024
    vb.cpus = 2
  end

  dev.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file = "site.pp"
    puppet.module_path = "puppet/modules"
  end
end
```

Dev Setup: The solution?

```
class nginx {
  package { 'nginx': ensure => present }

  service { 'nginx':
    ensure  => running,
    require => Package['nginx'];
  }

  file { '/etc/nginx/nginx.conf':
    source  => 'puppet:///modules/nginx/nginx.conf',
    require => Package['nginx'],
    notify  => Service['nginx'];
  }
}
```

Dev Setup: The nickel tour

Let's have a quick look before the real demo

Demo

Q&A

Thank you!