A WebSocket-based Approach to Transporting Web Application Data

Ross Andrew Hadden

March 26, 2015

Dr. Paul Talaga, committee chair
Dr. Fred Annexstein, committee member
Dr. John Franco, committee member

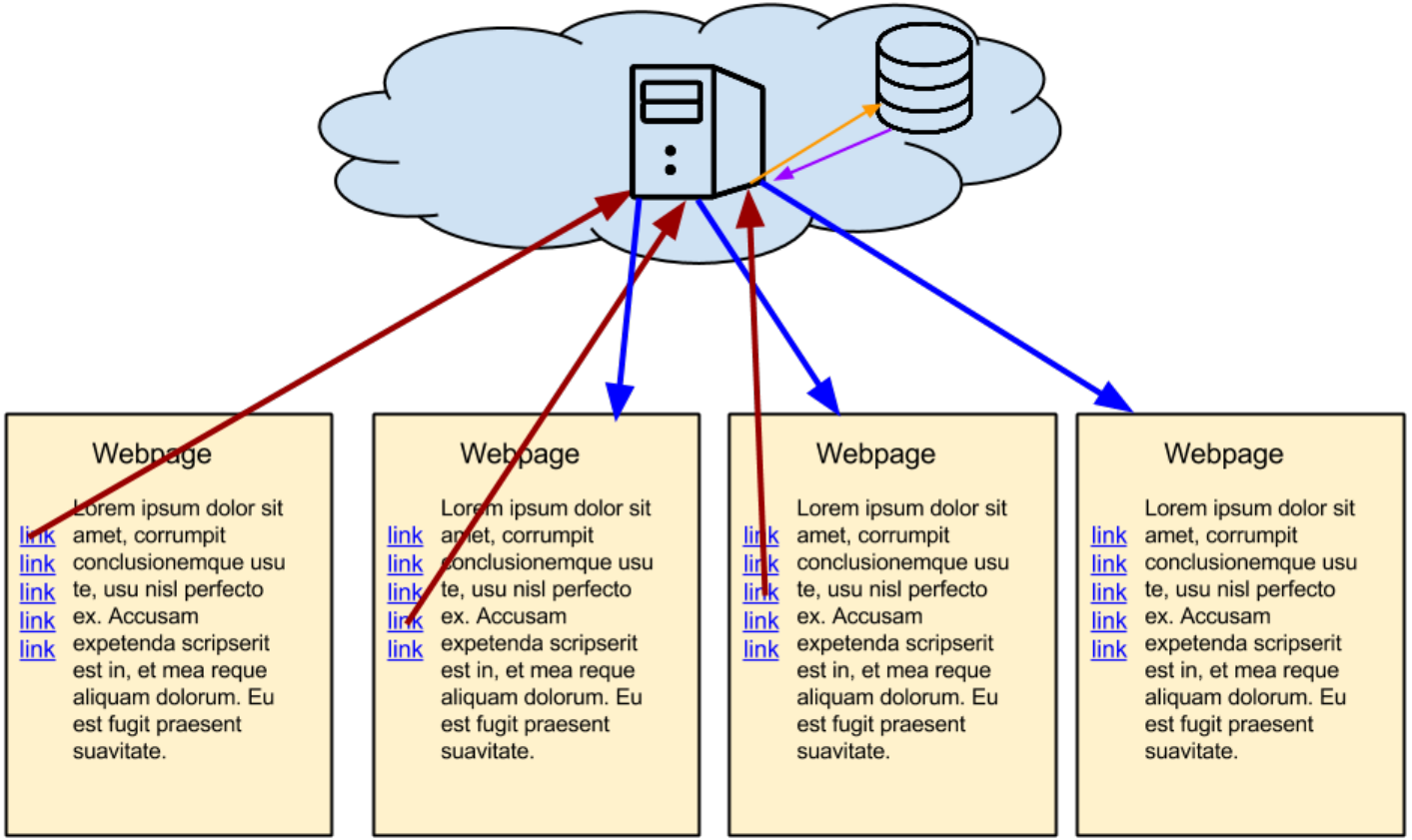
UNIVERSITY OF Cincinnati

Background

- Transporting data to clients
  - Server rendering

UNIVERSITY OF Cincinnati

Typical Server Interaction, 1990's

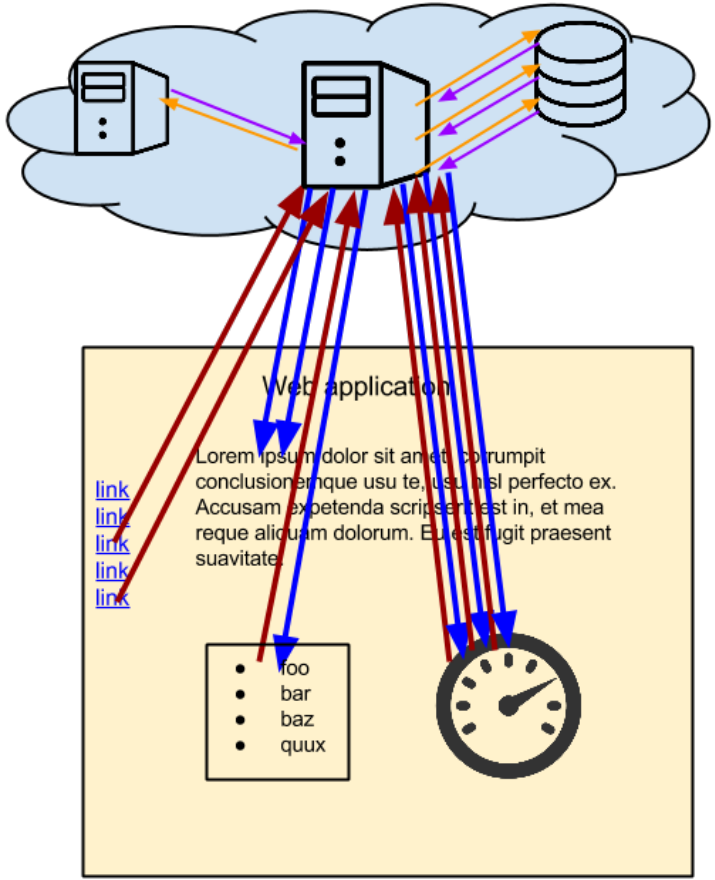


UNIVERSITY OF Cincinnati

Background

- Transporting data to clients
  - Server rendering
  - AJAX

Typical Server Interaction, 2010

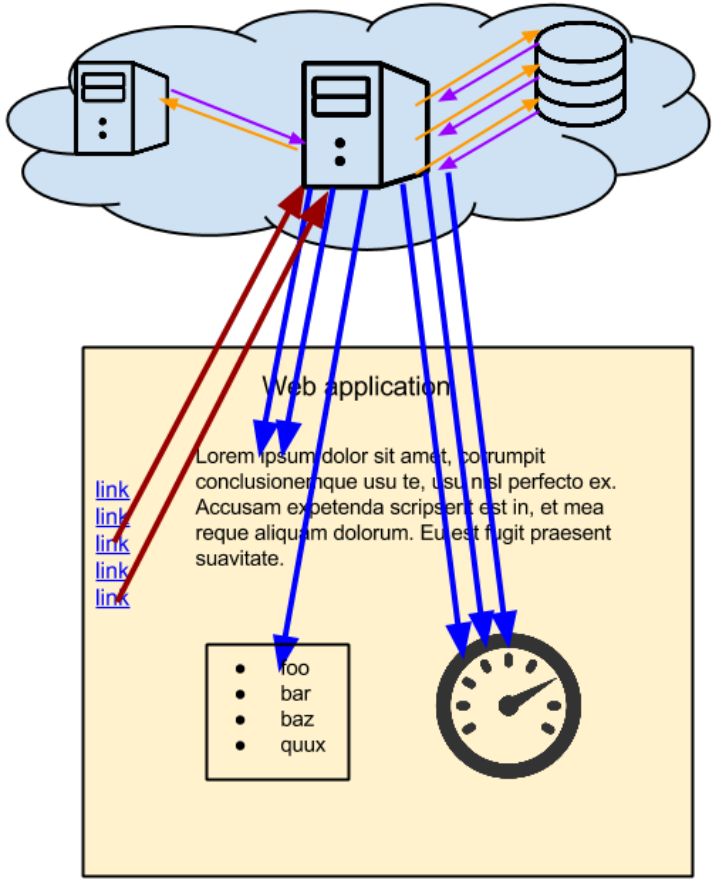


UNIVERSITY OF Cincinnati

Background

- Transporting data to clients
  - Server rendering
  - AJAX
  - WebSockets

Typical Server Interaction, 2015



UNIVERSITY OF Cincinnati

Motivation

Maybe we can improve on AJAX!

- Reduce request gap
- Reduce request latency
- Utilize Cornerstone for convenient experimentation

Overview

AJAX method

1. Client makes endpoint request
2. Server responds immediately
3. Client loads
4. Client fetches data
5. Client receives data when ready

Thesis method

1. Client makes endpoint request
2. Server responds immediately
3. Server fetches data
4. Client loads
5. Client receives data when ready

UNIVERSITY OF Cincinnati

Theoretical Advantages

- No request gap
- Fewer requests; same responses

UNIVERSITY OF Cincinnati

Implementation

```
// someController.js

let someController = {

    someRoute(req, res) {

        res.view({

            foo: "hello",

            bar: "world"

        });

    }

};
```

```
{{! someController/someRoute.hbs }}

<p>{{foo}}</p>

<p>{{bar}}</p>
```



```
<p>hello</p>

<p>world</p>
```

UNIVERSITY OF Cincinnati

Implementation, continued

- `Promise`
- WebSocket connection listener
- `{{{stream}}}` helper

```
// someController.js

let delayedData = new Promise(function(resolve, reject) {

    // resolves the promise after 5000 ms

    setTimeout(function() {

        resolve("delayed data!");

    }, 5000);

});


let someController = {

    someRoute(req, res) {

        res.view({

            foo: "hello",

            bar: delayedData

        });

    }

};
```

```
<p>{{{stream "bar"}}}</p>
```



```
<p><var data-promise="bar"></var></p>
```

UNIVERSITY OF Cincinnati

Testing Methods

- Server
  - relatable benchmark
- AJAX
  - modern practice
- Stream
  - thesis

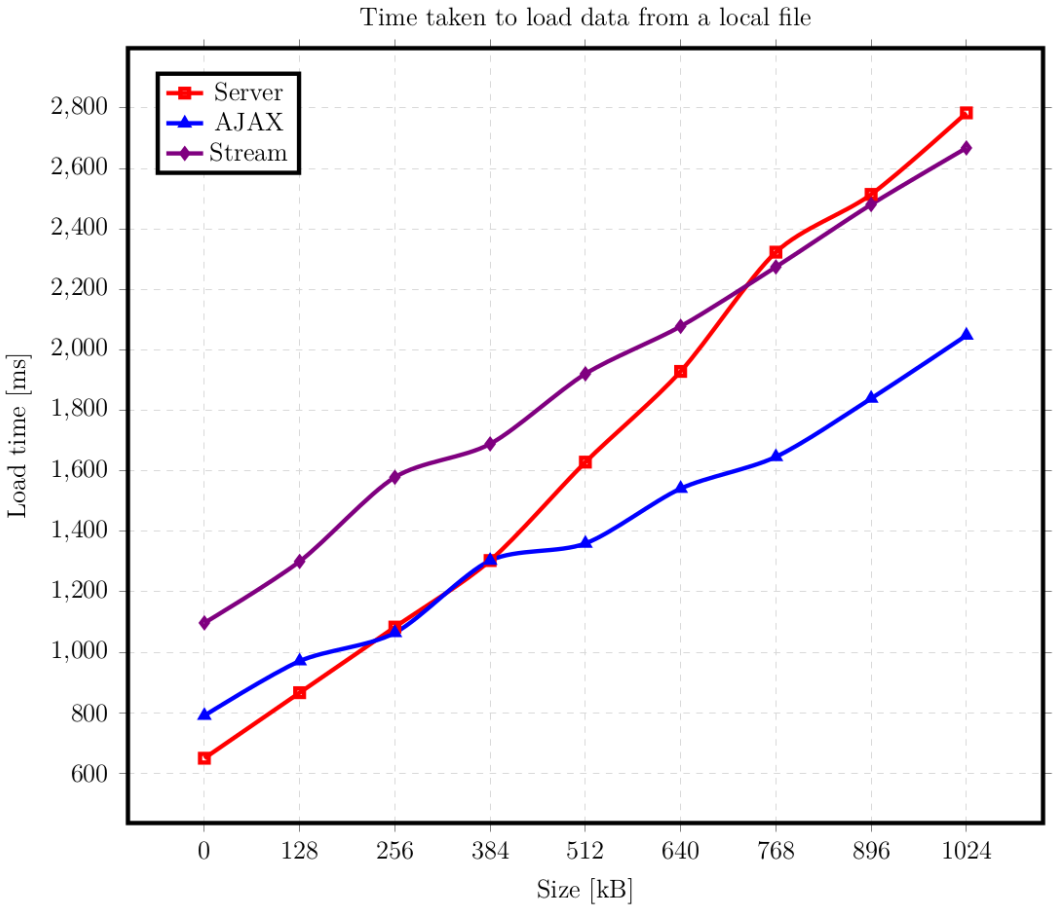
UNIVERSITY OF Cincinnati

Testing Scenarios

- Local file
- Local database
- Timeout
- Series
- Parallel

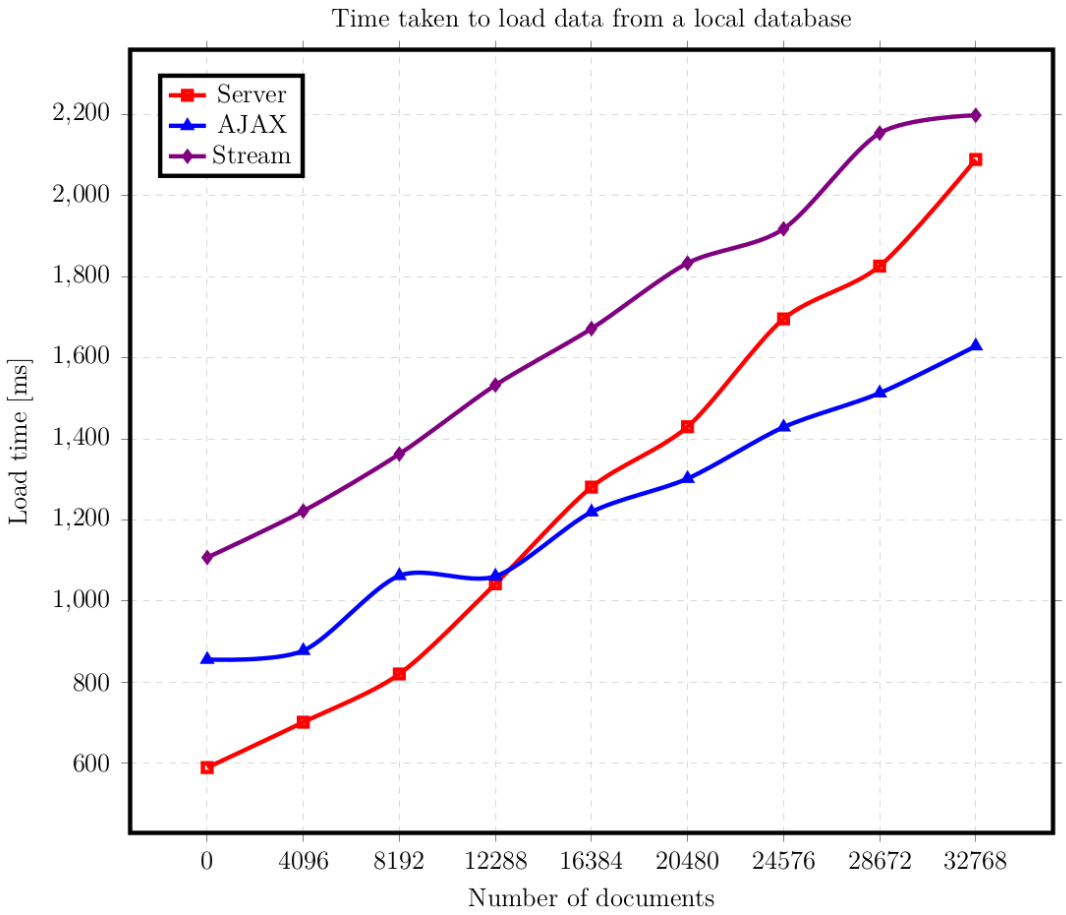
UNIVERSITY OF Cincinnati

Data - Local File



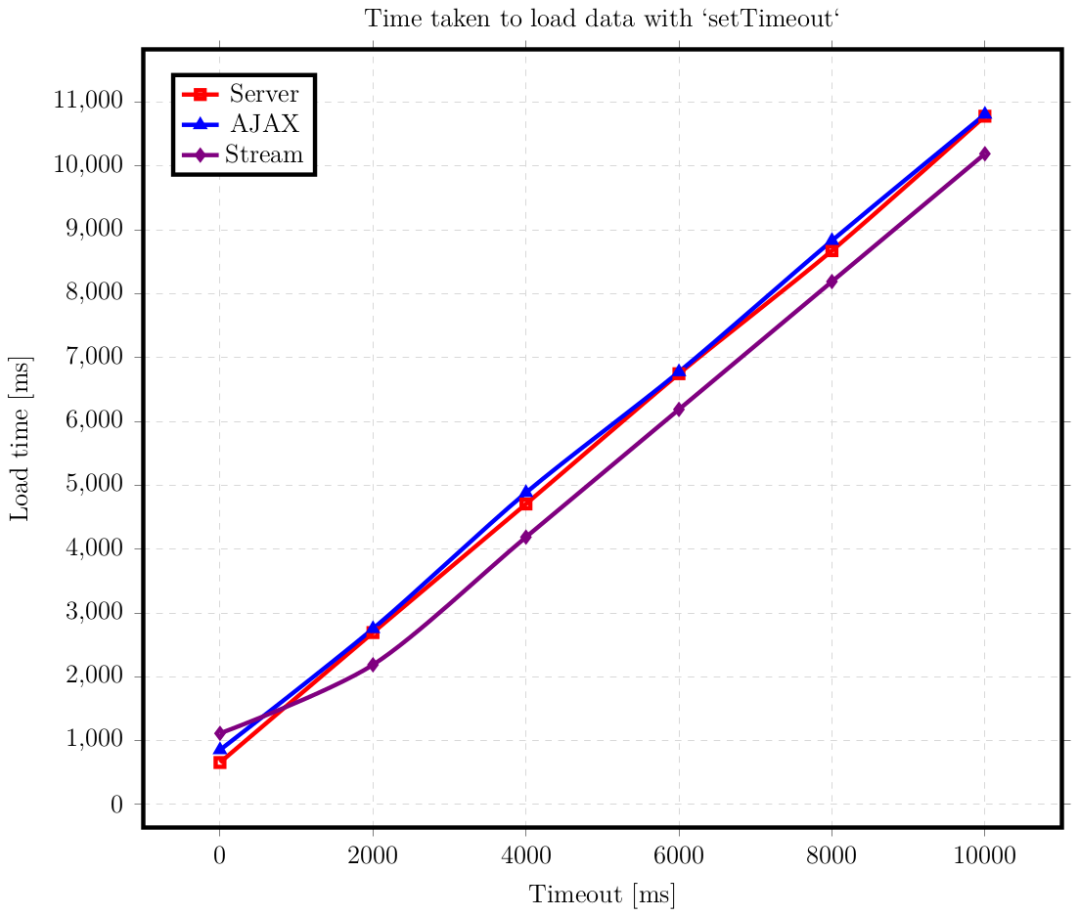
UNIVERSITY OF Cincinnati

Data - Local Database



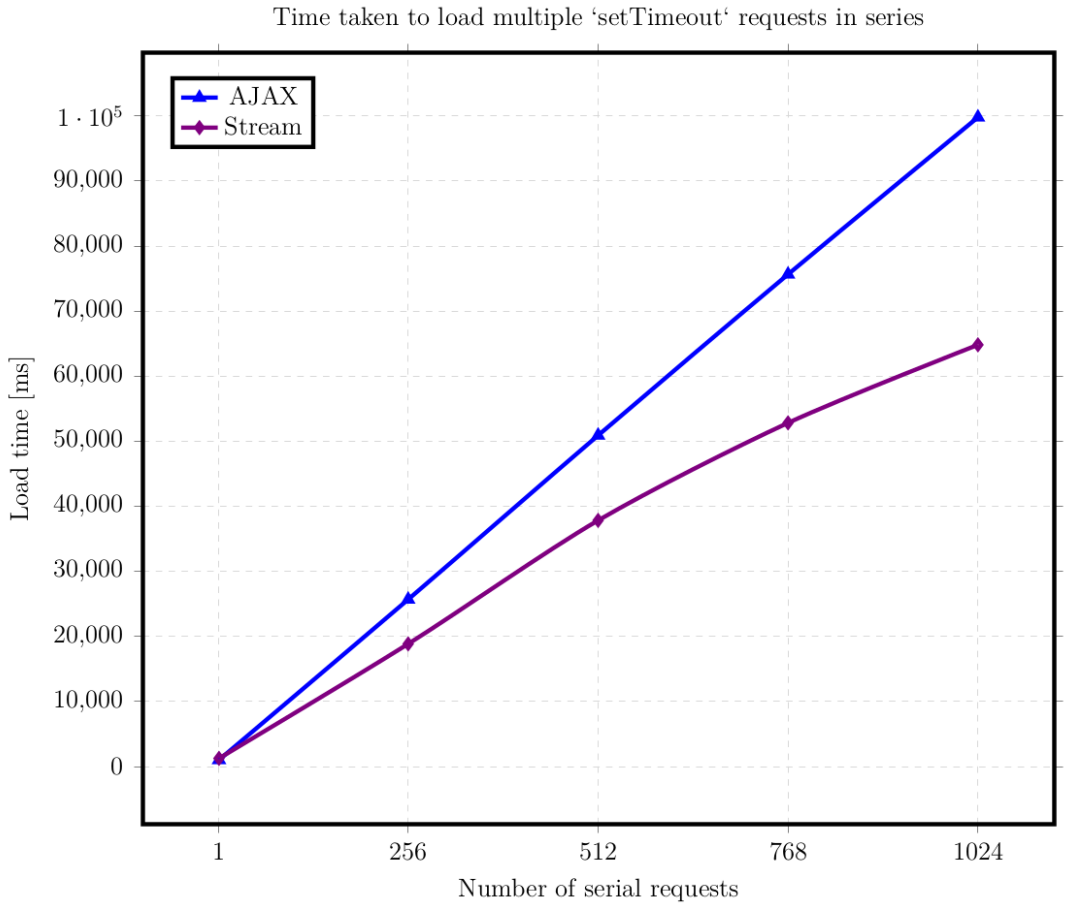
UNIVERSITY OF Cincinnati

Data - Timeout



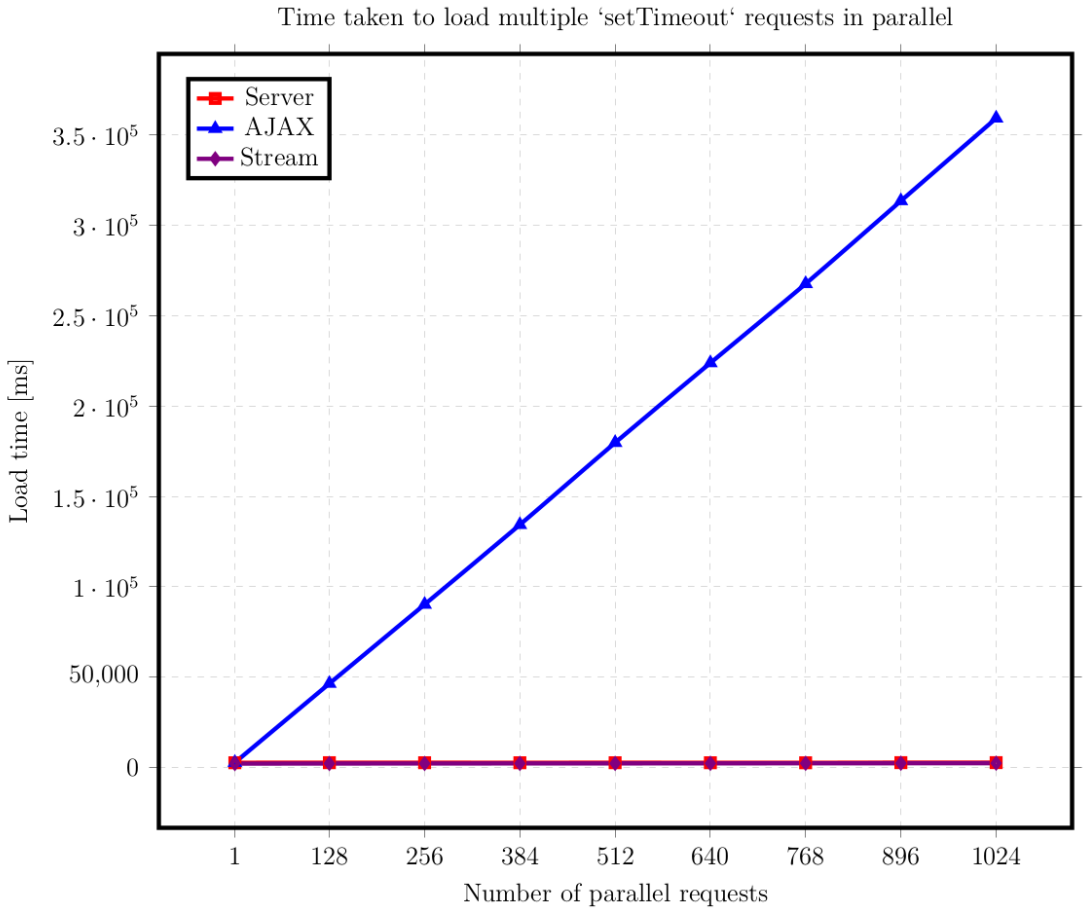
UNIVERSITY OF Cincinnati

Data - Series



UNIVERSITY OF Cincinnati

Scenarios - Parallel



UNIVERSITY OF Cincinnati

Analysis

- Number of HTTP requests, responses
  - Server: `1, 1`
  - AJAX: `n+1, n+1`
  - WebSockets: `2, n+2`

UNIVERSITY OF Cincinnati

Analysis, continued

- File and database tests
- Timeout test
- Series test
- Parallel test

UNIVERSITY OF Cincinnati

Conclusions

- Parallel requests → WebSockets
- Sequential requests → WebSockets
- Large data → AJAX
- Both?

Related Work

- MeteorJS
- HTTP pipelining (in `HTTP/1.1`)
  - bizarre vendor support
- `HTTP/2` (formerly Google SPDY)
  - bizarre vendor support

Future Work

- stream data in chunks
- stream data updates

UNIVERSITY OF Cincinnati

Questions?

Feel free to ask me anything

UNIVERSITY OF Cincinnati

References

"Browserscope." 2015. http://www.browserscope.org.

"Can I Use." 2015. http://caniuse.com.

"Cornerstone." 2015. https://github.com/ConnectAi/cs.

ECMA International. 2011. *Standard ECMA-262 - ECMAScript Language Specification.* 5.1. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

"Front-end Code Standards & Best Practices." 2012. Isobar. http://isobar-idev.github.io/code-standards.

"Hello HTTP/2, Goodbye SPDY." 2015. http://blog.chromium.org/2015/02/hello-http2-goodbye-spdy-http-9.html.

"Meteor." 2015. http://meteor.com.

UNIVERSITY OF Cincinnati

References, continued

"Mozilla Developer Network." 2015. Mozilla. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference.

"Node.js v0.12.0 Manual & Documentation." 2015. Node.js. https://nodejs.org/docs/v0.12.0/api.

"RFC 6455 - The WebSocket Protocol." 2015. Internet Engineering Task Force. http://tools.ietf.org/html/rfc6455.

"RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1)." 2015. Internet Engineering Task Force. http://tools.ietf.org/html/rfc7230.

"Reducing HTTP latency with SPDY." 2009. http://lwn.net/Articles/362473.

"SPDY: An experimental protocol for a faster web." 2009. http://dev.chromium.org/spdy/spdy-whitepaper.

UNIVERSITY OF Cincinnati

References, continued

Souders, Steve. 2007. *High Performance Web Sites - Essential Knowledge for Frontend Engineers: 14 Steps to Faster-Loading Web Sites.* O'Reilly.

"Top 5 Desktop Browsers from Q3 2008 to Q1 2015." 2015. StatCounter. http://gs.statcounter.com/#desktop-browser-ww-quarterly-200803-201501.

"socket.io." 2015. http://socket.io.

UNIVERSITY OF Cincinnati