Understanding Asynchronous Interactions In Full-Stack JavaScript

Saba Alimadadi, Ali Mesbah and Karthik Pattabiraman

ICSE 2016
saba@ece.ubc.ca

Sahand: http://github.com/saltlab/sahand
JavaScript: Most popular language

JavaScript: Top languages on GitHub
Understanding JavaScript Apps

Whole Picture
Challenge 1. **Server-Side Callbacks**

- Asynchronous execution
- Callback **hell**

```javascript
fs.readdir(source, function(err, files) {
    files.forEach(function(filename, fileIndex) {
        gm(source + filename).size(function(err, values) {
            widths.forEach(function(width, widthIndex) {
                this.resize(w, h).write(newName, function(err) {
                })
            })
        })
    })
})
```

// example from callbackhell.com
Challenge 2. **Network Communications**

- **Client**
  - Request 1
  - Request 2

- **Server**
  - Processing time 1
  - Processing time 2
  - Response 1
  - Response 2

**Time**
Challenge 3. Asynchronous Client Side
Summary of Challenges

• Server-side callbacks
• Network communication
• Asynchronous client side

Related work:

EMSE’13 UIST’14 ICSE’14, ECOOP’15
Our Approach: Sahand

1. Instrument automatically
2. Trace full-stack execution
3. Infer a behavioural model
4. Visualize the model
Behavioral Model

Nodes — Lifelines of function executions

(A)Synchronous client/server events

foo() — bar() — baz()

Links — Time, Type, Direction

event
Real Behavioural Models Are Complex
Visualization

**Client-Side Analysis**

<table>
<thead>
<tr>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

**Server-Side Analysis**

<table>
<thead>
<tr>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>

Connecting client and server
Visualization

Client-Side Analysis

<table>
<thead>
<tr>
<th>foo()</th>
<th>active</th>
<th>active</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar()</td>
<td>active</td>
<td></td>
</tr>
</tbody>
</table>

Events and DOM interactions

Timeouts

XHRs

Time — Temporal primitives — Time points

Server-Side Analysis

<table>
<thead>
<tr>
<th>baz()</th>
<th>active</th>
<th>act</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.js:45</td>
<td>scheduled</td>
<td>active</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
<td></td>
</tr>
</tbody>
</table>

Event loop
## Visualization

### Client-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

### Server-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
</tr>
<tr>
<td>app.js:45</td>
<td>active</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>

### Function executions

Time — Temporal primitives — Time intervals

### Callbacks

Visualization of client-side and server-side analysis, focusing on function executions and scheduling.
Visualization

**Client-Side Analysis**

- `foo()` active active
- `bar()` active

**Server-Side Analysis**

- `baz()` active active act
- `app.js:45` scheduled active
- `qux()` active

Time — Structure of time → Linear & Branching
Visualization

Client-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

Time — Structure of time — Linear & Branching

Server-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active act</td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled active</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>
Implementation: **Sahand**

- Express.js application
- Proxy -> dynamic instrumentation
- Esprima, Estraverse, Escodegen

https://github.com/saltlab/sahand
Evaluation

Does using Sahand improve developers’ performance in program comprehension tasks?
Controlled Experiment

- **Sahand**’s effect on developers’ performance
- 12 Participants
- Object: full-stack JavaScript application
Controlled Experiment

• Design
  – Control: tool and expertise level
  – Measure: performance

• Procedure
  – Pre-questionnaire
  – Tutorial
  – Tasks
  – Post-questionnaire
Results Highlight

Using **Sahand**

3 times more accuracy

In the same time
Summary of Challenges

- Server-side callbacks
- Network communication
- Asynchronous client side

Related work:
- Zaidman et al. EMSE13
- Hibschman et al. UBT14
- Alimadadi et al. ICSE14, ECOOP15

Behavioral Model: Example

Results Highlight

Using Sahand
3 times more accuracy
In the same time

Saba Alimadadi
Hire Me!