Gulfstream

Staged Static Analysis for Streaming JavaScript Applications

Salvatore Guarnieri
University of Washington

Ben Livshits
Microsoft Research
Safe Code Inclusion In JavaScript

Runtime Enforcement
- Conscript [Oakland 10]
- BrowserShield [OSDI 06]
- Caja

Static Analysis
- Gatekeeper [USENIX Sec 09]
- Staged Information flow for JavaScript [PLDI 09]

Whole program analysis approaches require the entire program
Script Creation

What does \( f \) refer to?
Incremental Loading in Facebook

- Profile
- Inbox
- Friends
- Home

- KB
- 157
- 71%
Outline

• Motivation
• Implementation
• Evaluation
• Conclusions
Queries

• We want to determine something about the program

• Example
  – What does f() refer to
  – Detect alert calls
  – Does this program use setTimeout
Points-To Analysis

• Provides deep program understanding

• Can be used to construct call graphs

• Is the foundation of further analyses

• Answers a simple question: What heap locations does variable x point to
Points-To Example

1. var A = new Object();
2. var B = new Object();
3. x = new Object();
4. x.foo = new Object();
5. y = new Object();
6. y.bar = x;
7. y.add = function(a, b) {};
8. y.add(A, B)
Implementation Strategies

**Datalog with bddbddb**
- Fast for large programs
- Highly tuned
- Large startup cost
- Difficult to implement in the browser

• Used in Gatekeeper [USENIX Sec 09]

**Graph-based flow analysis**
- Very small startup cost
- Customized to work with Gulfstream
- Does not scale well
Implementation

• Normalize JavaScript
  – Turn program into a series of simple statements
  – Introduce temporaries as necessary

• Create flow graph – Use normalized program to generate flow constraints

• Serialize flow graph – Encode the flow-graph so online analysis can use it to update results
Implementation Continued

• Perform points-to analysis
  – Traverse flow graph to find all aliases
  – Follow flow through method boundaries
  – Generate points-to map for queries to use

• Queries – Use points-to data and flow graph to answer queries
Evaluation

• Question – Is Gulfstream faster than non-staged analysis

• Benchmarks
  – Synthetically generated
  – Scraped from Google code
  – Scraped from Facebook

• Simulate diverse environments
  – CPU speed and network properties
  – Cell phone, laptop, desktop, etc.
Laptop Running Time Comparison

After 30KB of updates, Gulfstream is no longer faster
## Simulated Devices

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>CPU coeff. c</th>
<th>Link type</th>
<th>Latency $L$ in ms</th>
<th>Bandwidth $E$ in kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>67.0</td>
<td>EDGE</td>
<td>500</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>Palm Pre</td>
<td>36.0</td>
<td>Slow 3G</td>
<td>500</td>
<td>3.75</td>
</tr>
<tr>
<td>3</td>
<td>iPhone 3G</td>
<td>36.0</td>
<td>Fast 3G</td>
<td>300</td>
<td>12.5</td>
</tr>
<tr>
<td>4</td>
<td>iPhone 3GS 3G</td>
<td>15.0</td>
<td>Slow 3G</td>
<td>500</td>
<td>3.75</td>
</tr>
<tr>
<td>5</td>
<td>iPhone 3GS WiFi</td>
<td>15.0</td>
<td>Fast WiFi</td>
<td>10</td>
<td>75.0</td>
</tr>
<tr>
<td>6</td>
<td>MacBook Pro 3G</td>
<td>1.0</td>
<td>Slow 3G</td>
<td>500</td>
<td>3.75</td>
</tr>
<tr>
<td>7</td>
<td>MacBook Pro WiFi</td>
<td>1.0</td>
<td>Slow WiFi</td>
<td>100</td>
<td>12.5</td>
</tr>
<tr>
<td>8</td>
<td>Netbook</td>
<td>2.0</td>
<td>Fast 3G</td>
<td>300</td>
<td>12.5</td>
</tr>
<tr>
<td>9</td>
<td>Desktop WiFi</td>
<td>0.8</td>
<td>Slow WiFi</td>
<td>100</td>
<td>12.5</td>
</tr>
<tr>
<td>10</td>
<td>Desktop T1</td>
<td>0.8</td>
<td>T1</td>
<td>5</td>
<td>1,250.0</td>
</tr>
</tbody>
</table>
Lessons Learned

• **Slow devices** benefit from Gulfstream

• A **slow network** can negate the benefits of the staged analysis

• **Large page updates** don’t benefit from Gulfstream
Facebook Experiment

• Visit 4 pages
  – Home
  – Friends
  – Inbox
  – Profile

• Each page loads additional JavaScript
Gulfstream Savings: Fast Devices

10 seconds saved

Seconds

profile
inbox
friends
home
Conclusion

• Gulfstream, staged analysis for JavaScript

• Staged analysis
  – Offline on the server
  – Online in the browser

• Wide range of experiments
  – For small updates, Gulfstream is faster
  – Devices with slow CPU benefit most
The End

• Contact: salvatore.guarnieri@gmail.com
BONUS
BONUS
With Static Analysis

- Adds `widget.js` to web page
- All `String.substring` uses on page will send their data to the attacker

Content provider runs analysis and notices a global overwrite and doesn’t send the page to the client

Web Application

Third Party Server

Sets `String.substring` to send data to attacker

Web page

widget.js
Web Sites Are Dynamic

• Web sites update data without reloading the page
• The same is done with code
• Typical analysis results would be invalid after the first code update
Program has more new code and potentially vastly different meaning than previous update

Analysis results are invalid
Analysis results are invalid
Analysis results are invalid
Incremental Points-to Information

• Not every reference is updated

• Only update points-to information for updated references

• Updates to websites are generally small
Soundness

• Gulfstream is *not* sound

• Doesn’t disallow any language features though

• Useful for bug finding
Perform staged analysis

page.html
a.js
b.js
c.js

Updated analysis results
Updated staged state
page2_update.js
User interaction

Your homepage

Cool link
New Content

Next page

page.html
New data can come from other servers

First Party Server

Get cool.js
New data can come from other servers
Perform staged analysis

First Party Server

- page.html
- a.js
- b.js
- c.js
- page2_update.js

Updated analysis results
Updated staged state
cool.js
Client

Server

- Page is created
- Generate points-to for static page
- Page is requested

Interact with page

Code update

- Send results
- Update points-to results

Change this to nicer animation, but does this belong here? It could go into more details that previous animation but will likely be pretty similar.
Incremental Algorithm
Transferring Data

• Part of the time to do staged analysis is network transfer time

• Efficient representation of points-to data minimizes transfer time

I think this should be mentioned in the previous slide when talking about what it is Sending over the network
Different platforms

• Two independent measures
  – Analysis speed improvement on local machine
  – Transfer time for intermediate data
• Full analysis may be faster if network link is slow compared to CPU speed
• Different devices will have different points where full analysis is better than staged
Non-Staged Outline

1. Load Page
2. Concatenate Scripts
3. Normalize JavaScript
4. Generate Constraints
5. Solve Constraints
6. Run Queries
7. Code Update
Staged Data Transfer

This data must now be sent to the client
Staged Queries

• Only recompute query for updated points-to-data
Staged is always right for Facebook

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>541</td>
<td>290</td>
<td>291</td>
<td>119</td>
<td>121</td>
<td>5</td>
<td>7</td>
<td>15</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>friends</td>
<td>3808 3</td>
<td>2046 0</td>
<td>2046 9</td>
<td>8516</td>
<td>8530</td>
<td>554</td>
<td>564</td>
<td>1133</td>
<td>450</td>
<td>454</td>
</tr>
<tr>
<td>inbox</td>
<td>6968 5</td>
<td>3743 9</td>
<td>3745 1</td>
<td>1558 9</td>
<td>1560 6</td>
<td>1022</td>
<td>1035</td>
<td>2075</td>
<td>827</td>
<td>832</td>
</tr>
<tr>
<td>profile</td>
<td>2230 29</td>
<td>1198 33</td>
<td>1198 45</td>
<td>4992 0</td>
<td>4993 7</td>
<td>3311</td>
<td>3323</td>
<td>6652</td>
<td>2658</td>
<td>2663</td>
</tr>
</tbody>
</table>
Future Directions

• Apply to eval in browser
Conclusions

• Staged points-to analysis for JavaScript
• Allows analysis as program loads more code
• Decreases time for analysis for small code updates
• Benefits low powered devices most
User interaction

Your homepage

Cool link

Next page
User interaction causes delayed code load

Get page2 data
User interaction causes delayed code load

page2_update.js
Staged analysis is always faster for Facebook

<table>
<thead>
<tr>
<th>Page</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>home</td>
<td>541</td>
</tr>
<tr>
<td>friends</td>
<td>38083</td>
</tr>
<tr>
<td>inbox</td>
<td>69685</td>
</tr>
<tr>
<td>profile</td>
<td>22302</td>
</tr>
</tbody>
</table>