[ICWE'19]

Catch & Release:
An Approach to Debugging Distributed Full-Stack JavaScript Applications

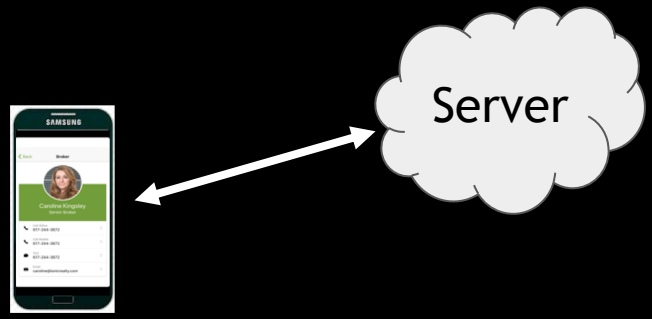
Software Innovation Lab
Department of Computer Science
Virginia Tech

Kijin An and Eli Tilevich

VIRGINIA TECH

Debugging is hard...

Debugging
distributed programs
is harder than
centralized programs...



volatility,
misconfigurations,
partial failure,
middleware problems,
reproducibility...

Key Idea

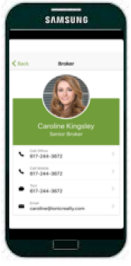
- *Catch & Release(**CandoR**)*

1. **Transform** a distributed application into its semantically equivalent ***centralized variant***
2. **Isolate** and **fix bugs** in the centralized variant

   : separating the **programmer-written code** from configuration/environmental issues as suspected bug causes
3. **Replay** the fixes on the distributed version

VIRGINIA TECH

Full-stack JS Apps: JavaScript everywhere



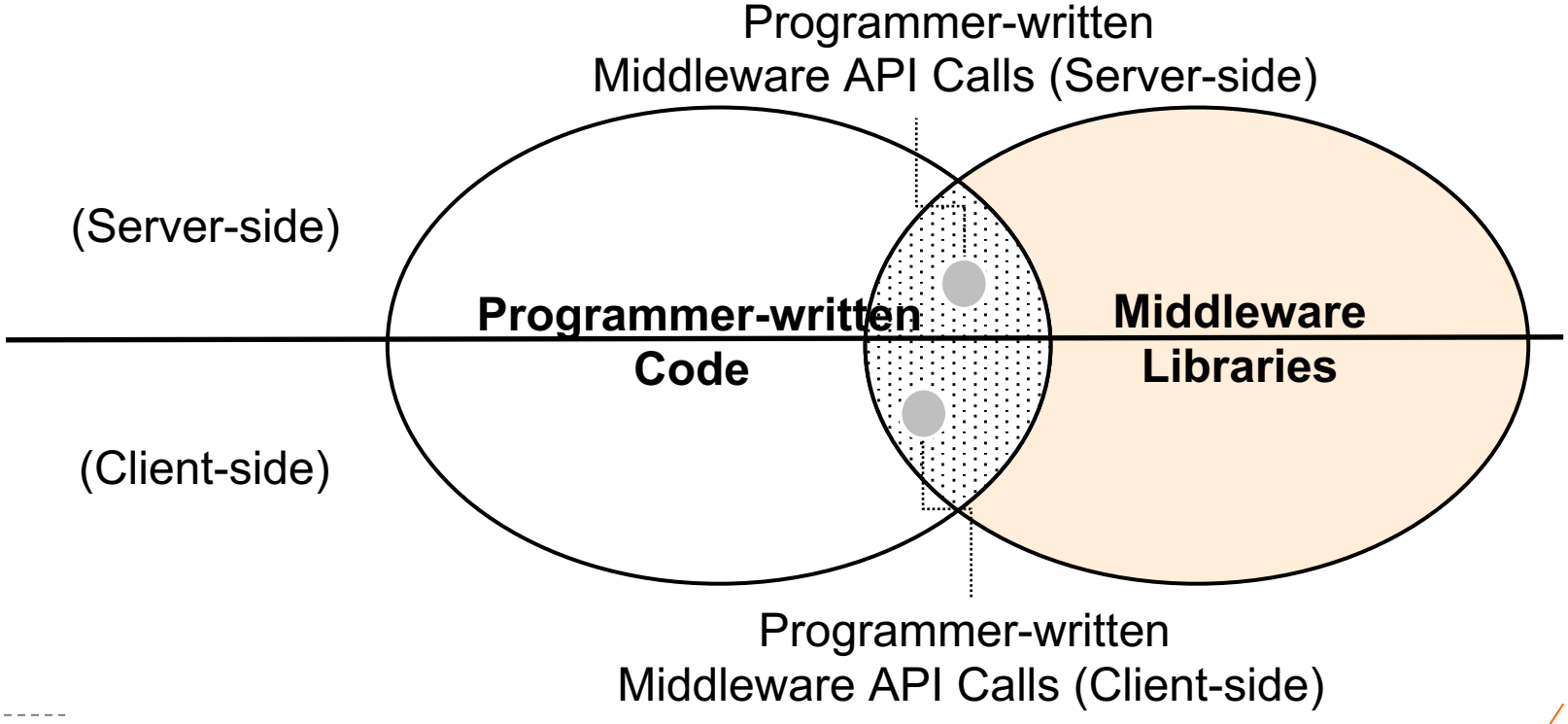
Remote Execution

Server

web client
--angular, react-native, Etc.

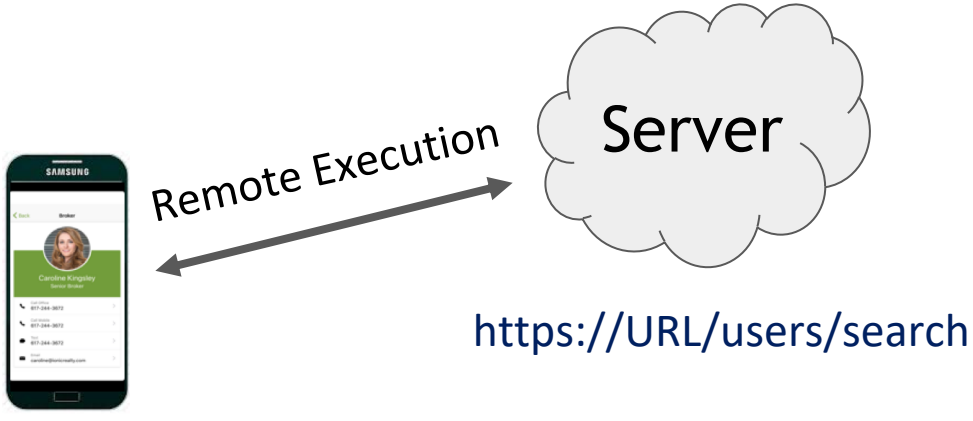
web server
(with Node.js)

Distributed Apps:
Middleware and Programmer-written Code



VIRGINIA TECH

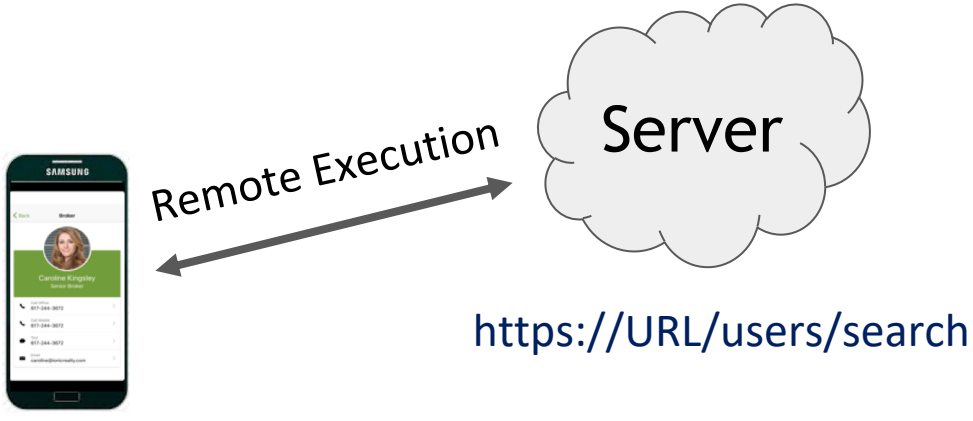
Full-stack JS Apps: BrownNode



(Programmer-written Code)

VIRGINIA TECH

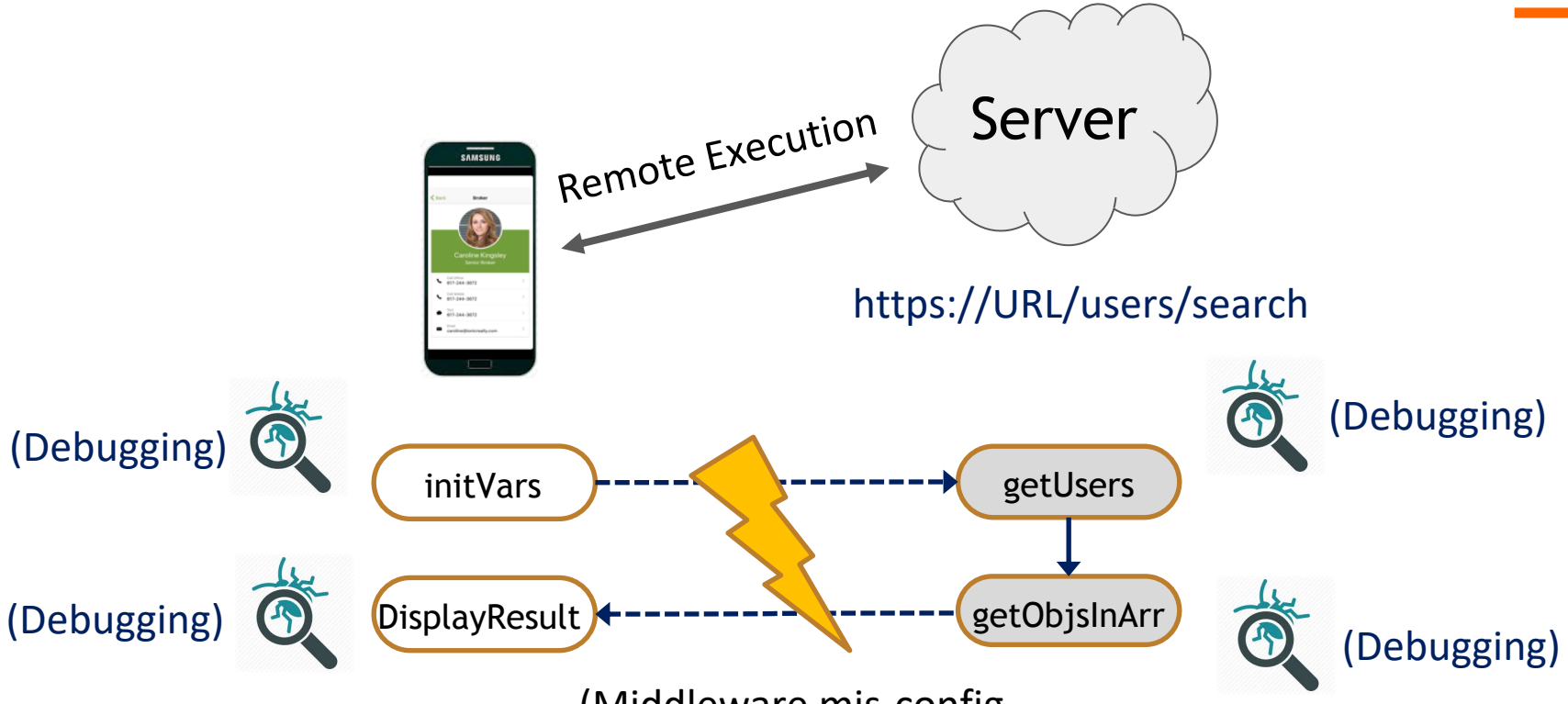
Full-stack JS Apps: BrownNode



(Middleware mis-config.,
Intermittent network)

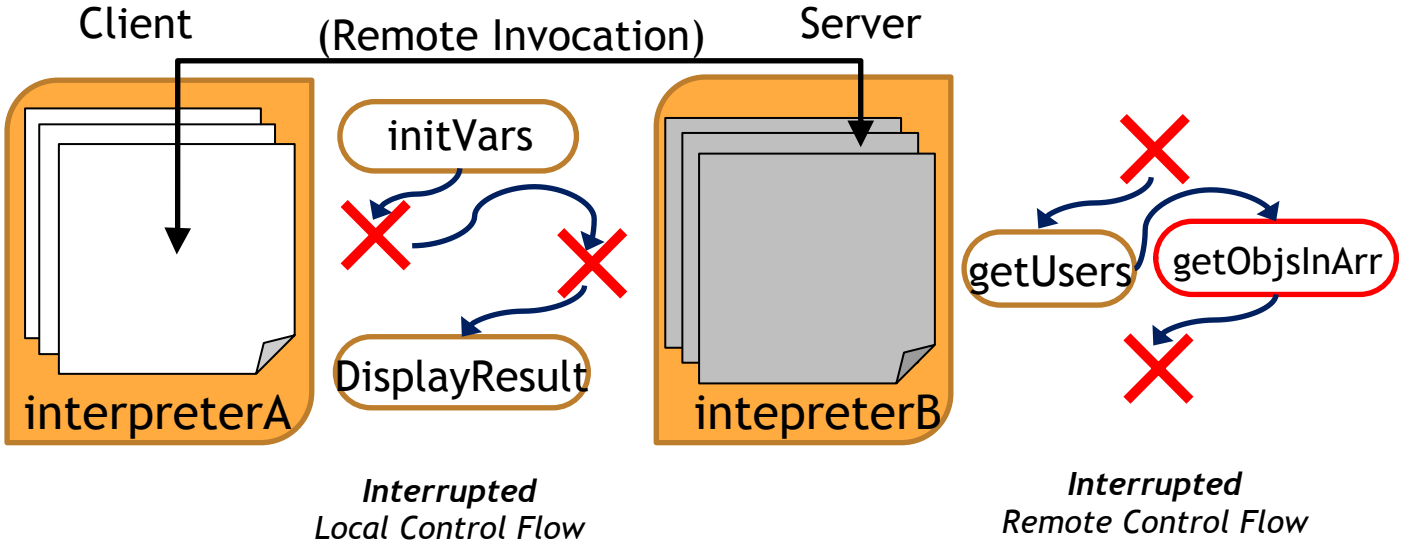
VIRGINIA TECH

Full-stack JS Apps: BrownNode



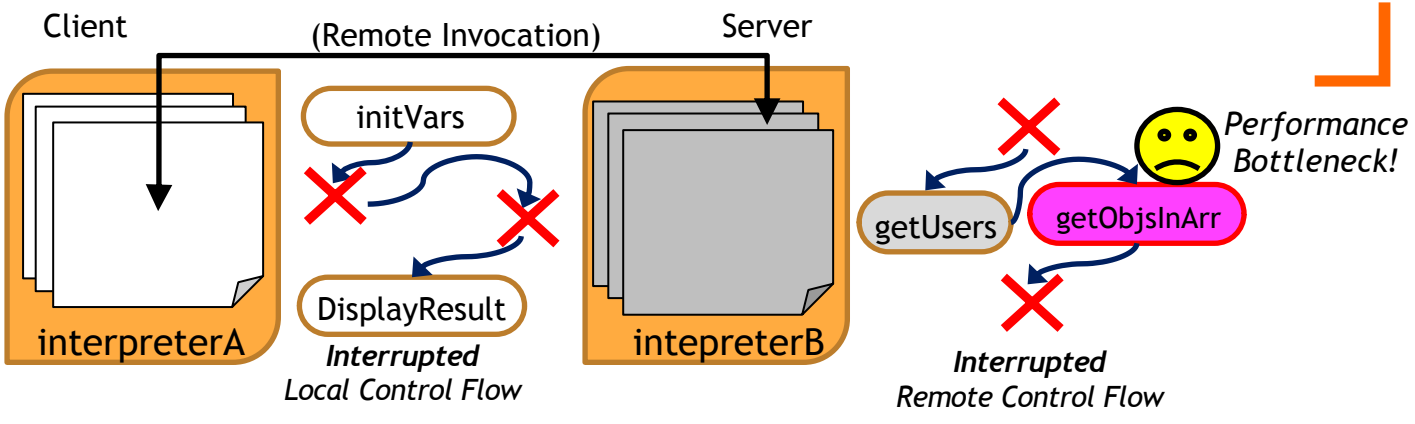
VIRGINIA TECH

Full-stack JS Apps: BrownNode



VIRGINIA TECH

Full-stack JS Apps: BrownNode



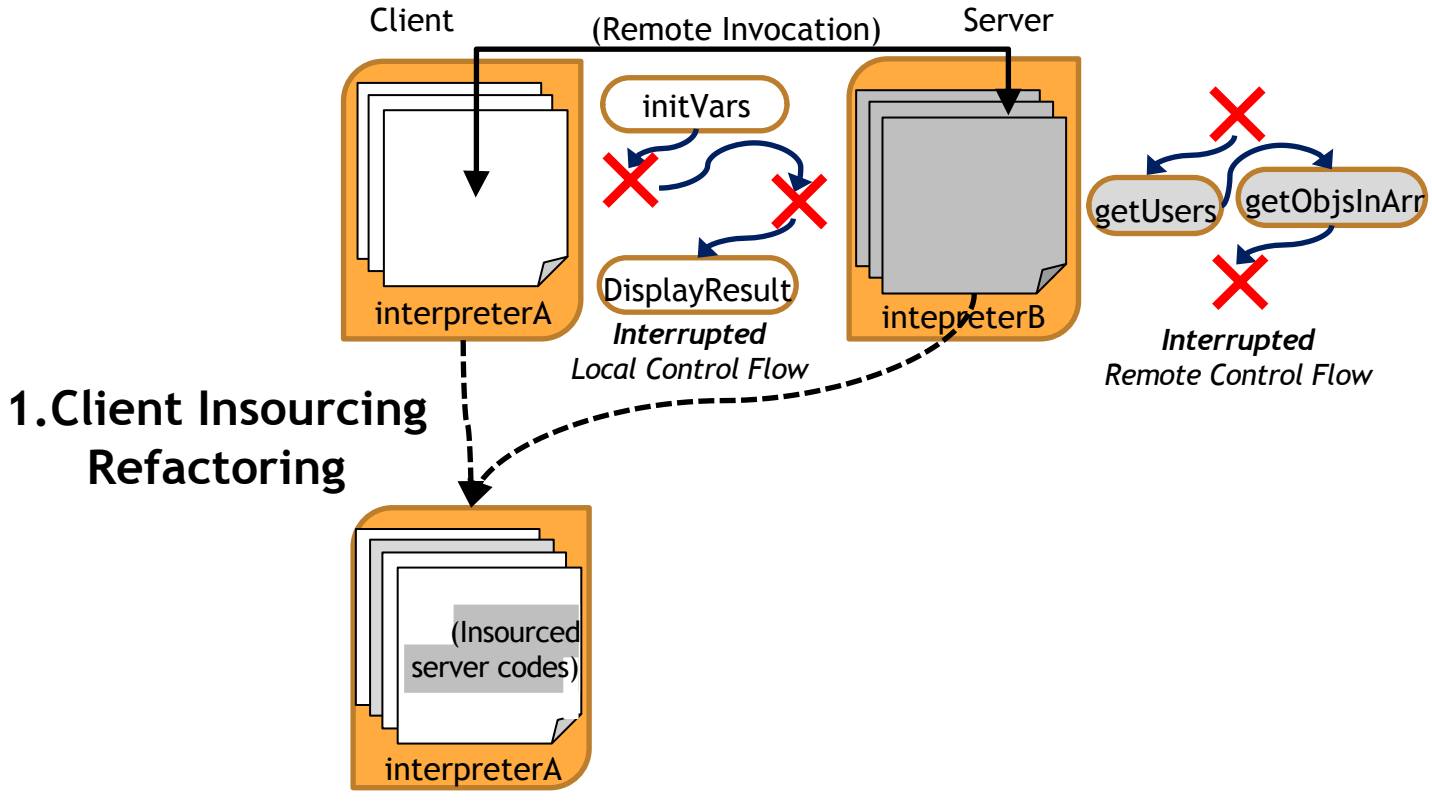

```
$.ajax({
 url: '/users/search',
 data:{
   fName:
$('#fName').val(),
 ...},
 type: 'POST',
 success:
function(data){
  $('#results').text(
   JSON.stringify(data)
);}});
```

```
var express = require('express'),
app = express.createServer(..);
var users=[]; ...
app.post('/users/search',
function(req, res) {
  var data = req.body;
  var result=getUsers(data);
  res.send(result);});

function getUsers(sUser) {
 return getObjsInArray(sUser,users);
}
```

```
function getObjsInArray(obj, array) {
 var foundObjs = [];
 for (var i=0; i<array.length;i++){
  for(var prop in obj) {
   if(obj.hasOwnProperty(prop)) {
    if(obj[prop]===array[i][prop])
    {
      foundObjs.push(array[i]);
      break;
   }}}}
 return foundObjs;
}
```

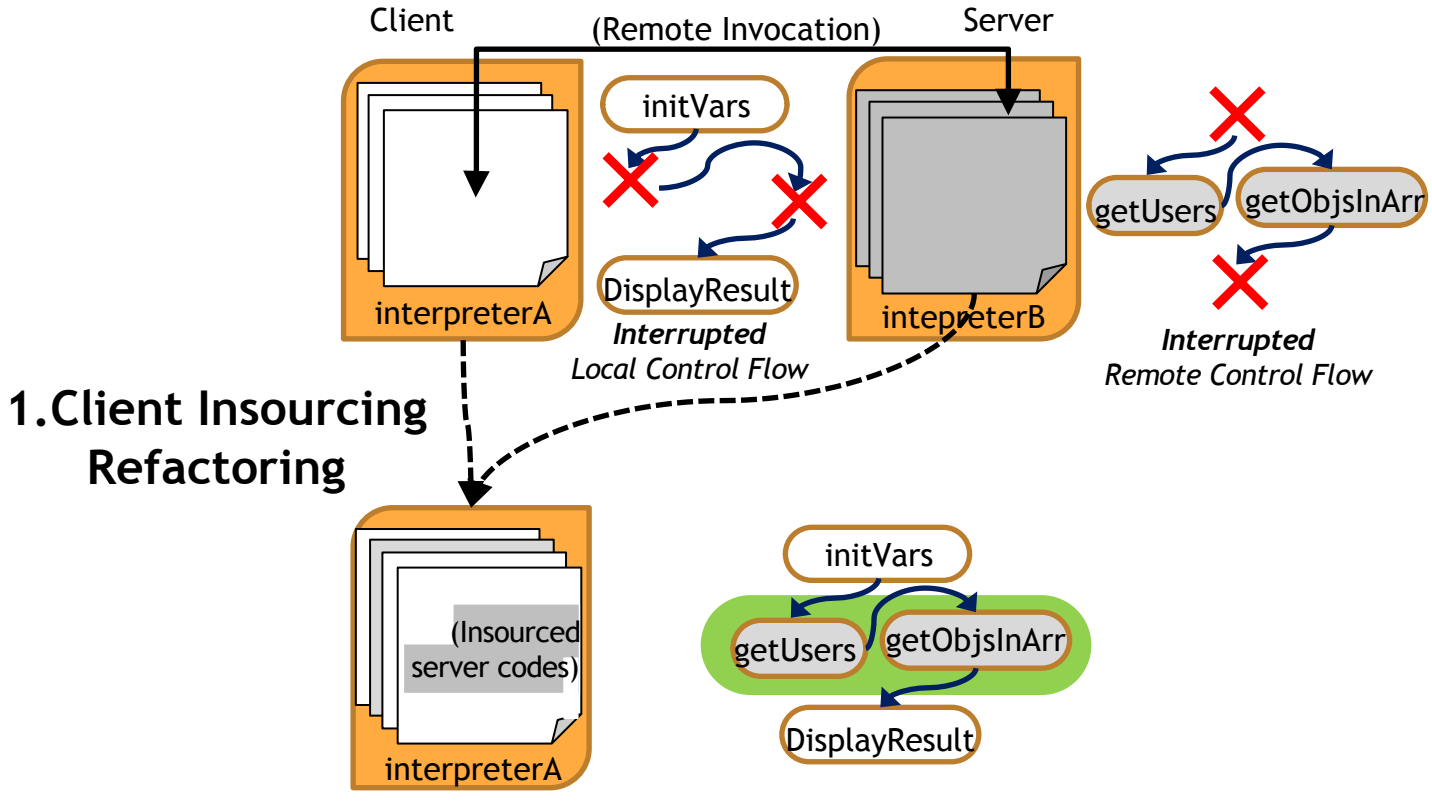
Client Insourcing Refactoring:
Undoing Distribution



Centralized App

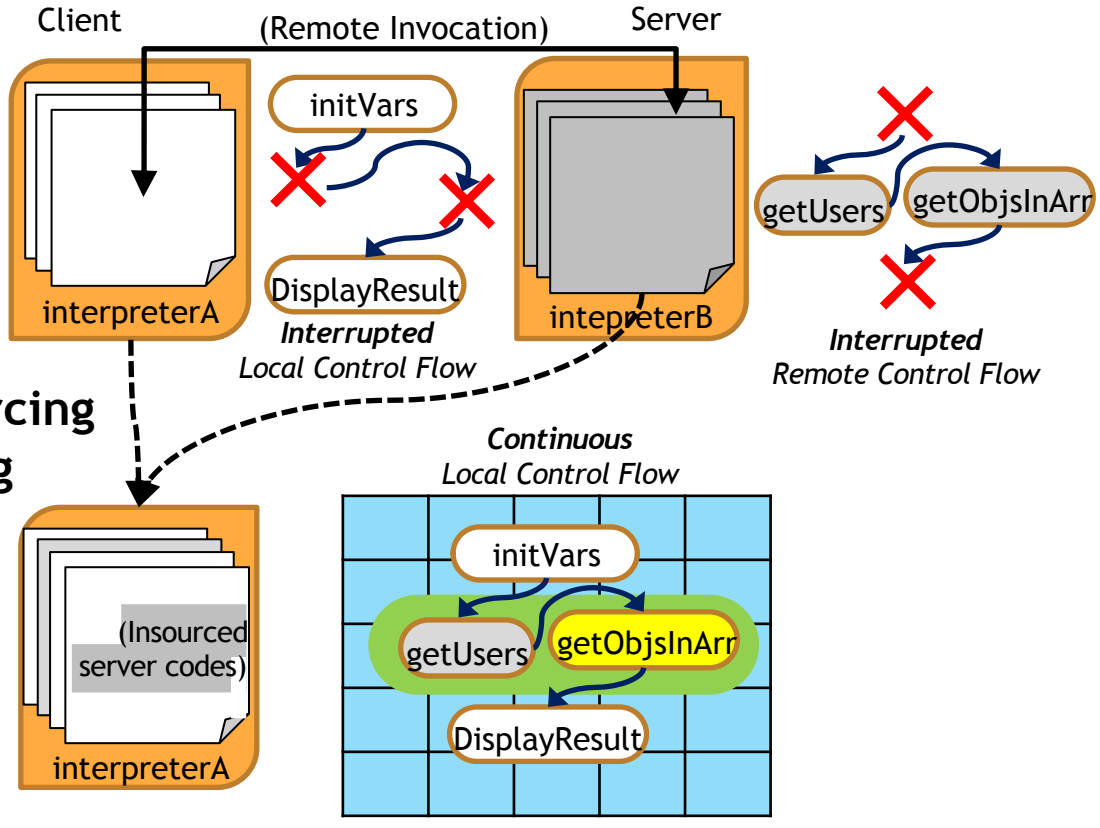
VIRGINIA TECH

Client Insourcing Refactoring:
Undoing Distribution



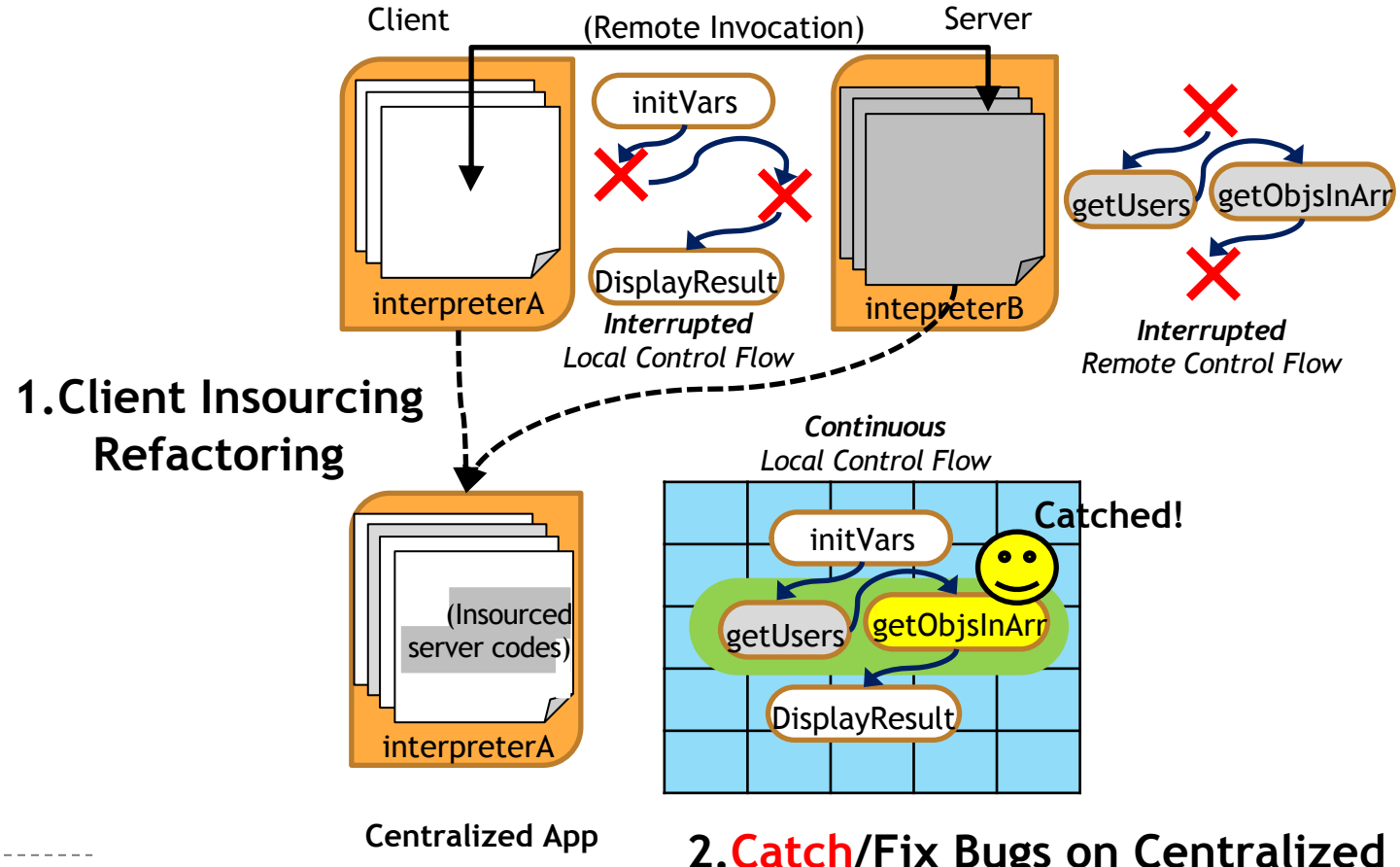
VIRGINIA TECH

Catch & Release (CandOR)



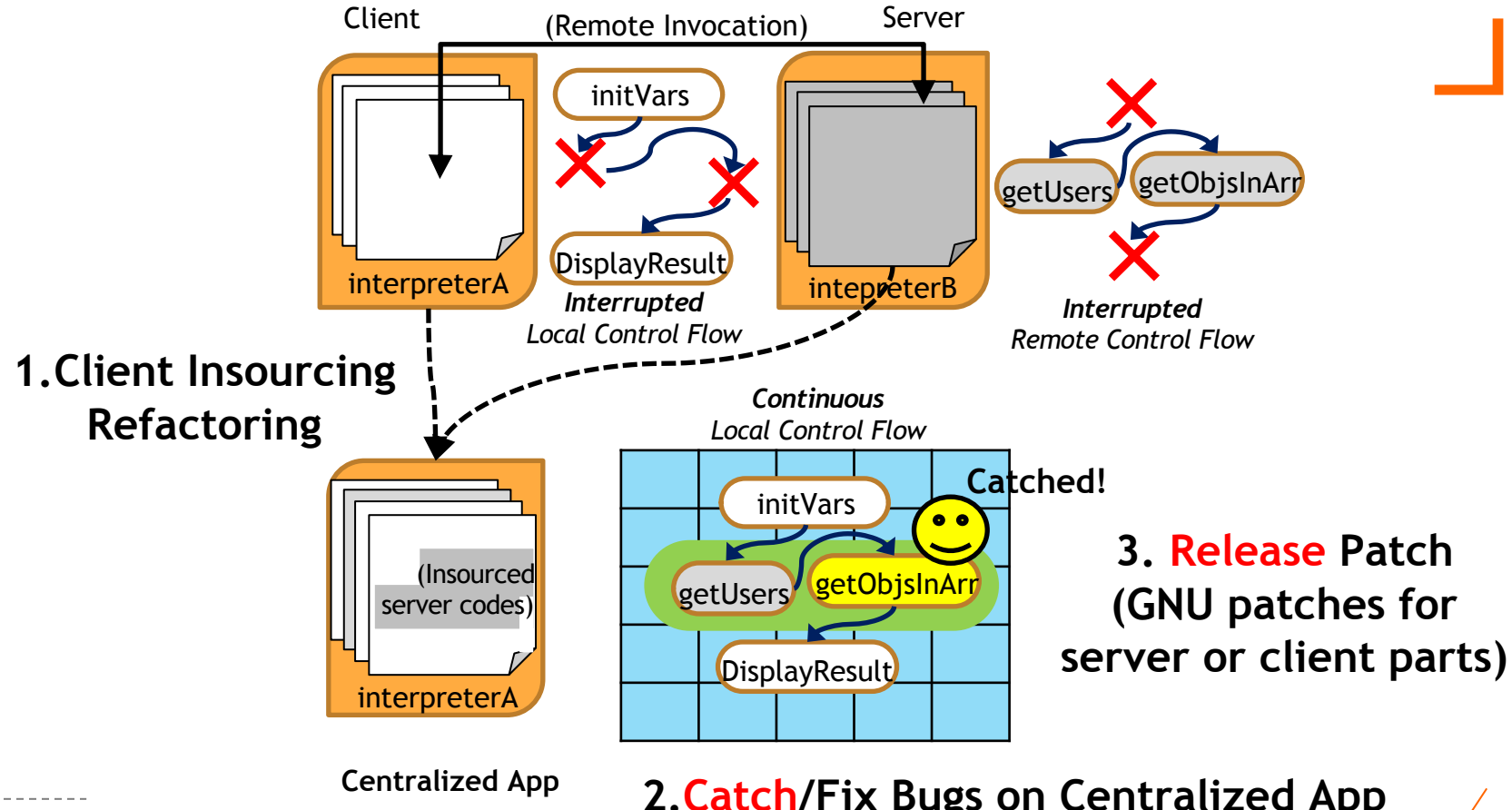
VIRGINIA TECH

Catch & Release (CandOR)



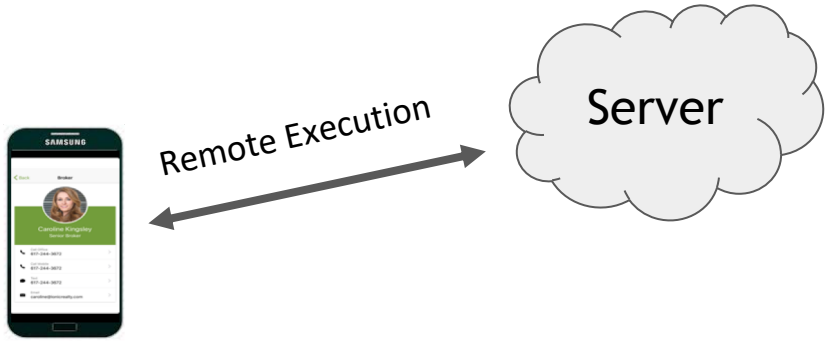
VIRGINIA TECH

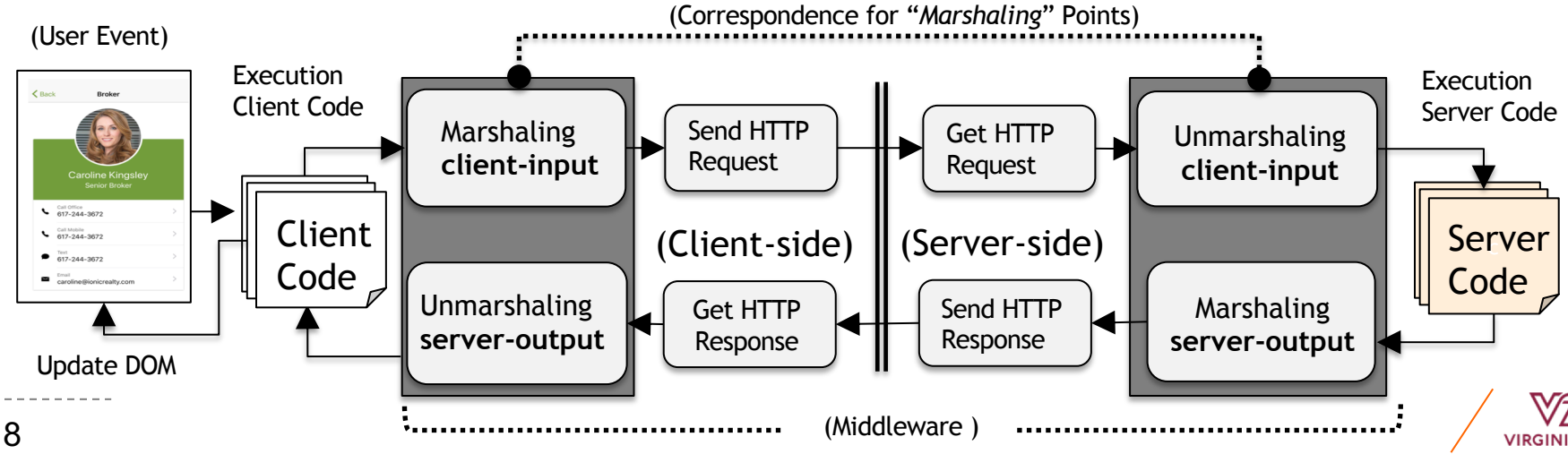
Catch & Release (CandOR)



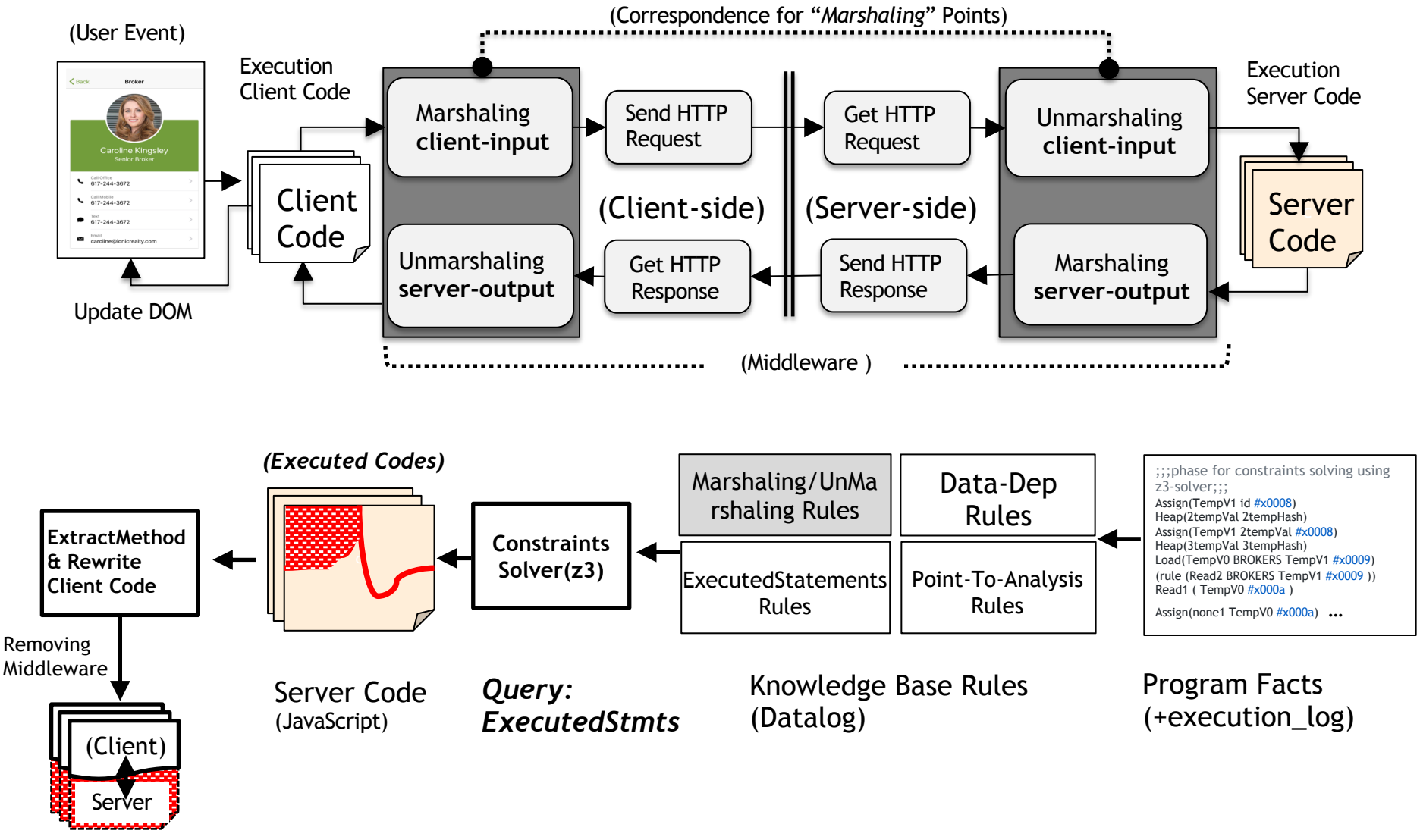
VIRGINIA TECH

Details for Client Insourcing Refactoring





VIRGINIA TECH

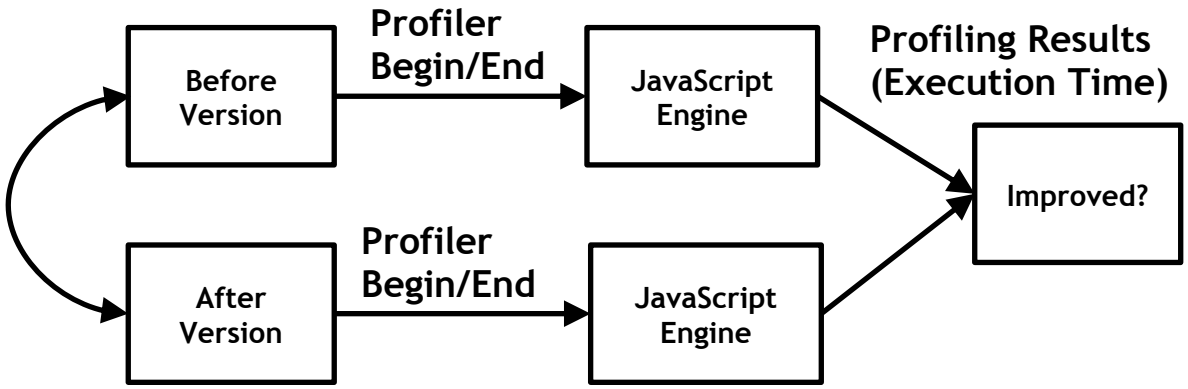

(User Event)
(Correspondence for "Marshaling" Points)
Execution Client Code
Execution Server Code
Broker
Caroline Kingsley
Senior Broker
Client Code
Marshaling client-input
Send HTTP Request
Get HTTP Request
Unmarshaling client-input
Server Code
(Client-side)
(Server-side)
Unmarshaling server-output
Get HTTP Response
Send HTTP Response
Marshaling server-output
Update DOM
(Middleware)
(Executed Codes)
ExtractMethod & Rewrite Client Code
Constraints Solver(z3)
Marshaling/UnMarshaling Rules
Data-Dep Rules
ExecutedStatements Rules
Point-To-Analysis Rules
;;;phase for constraints solving using z3-solver;;;
Assign(TempV1 id #x0008)
Heap(2tempVal 2tempHash)
Assign(TempV1 2tempVal #x0008)
Heap(3tempVal 3tempHash)
Load(TempV0 BROKERS TempV1 #x0009)
(rule (Read2 BROKERS TempV1 #x0009))
Read1 (TempV0 #x000a)
Assign(none1 TempV0 #x000a) ...
Removing Middleware
(Client)
Server
Server Code (JavaScript)
Query: ExecutedStmts
Knowledge Base Rules (Datalog)
Program Facts (+execution_log)

Performance Bottleneck Patterns for (centralized) JavaScript [17] (Selakovic, M. et al., ICSE'15)

```
function getObjsInArray(obj, array) {
 var foundObjs = [];
 for (var i=0; i<array.length;i++){
  for(var prop in obj) {
   if(obj.hasOwnProperty(prop)) {
    if(obj[prop]===array[i][prop])
    {
      foundObjs.push(array[i]);
      break;
   }}}}
 return foundObjs;
}
```

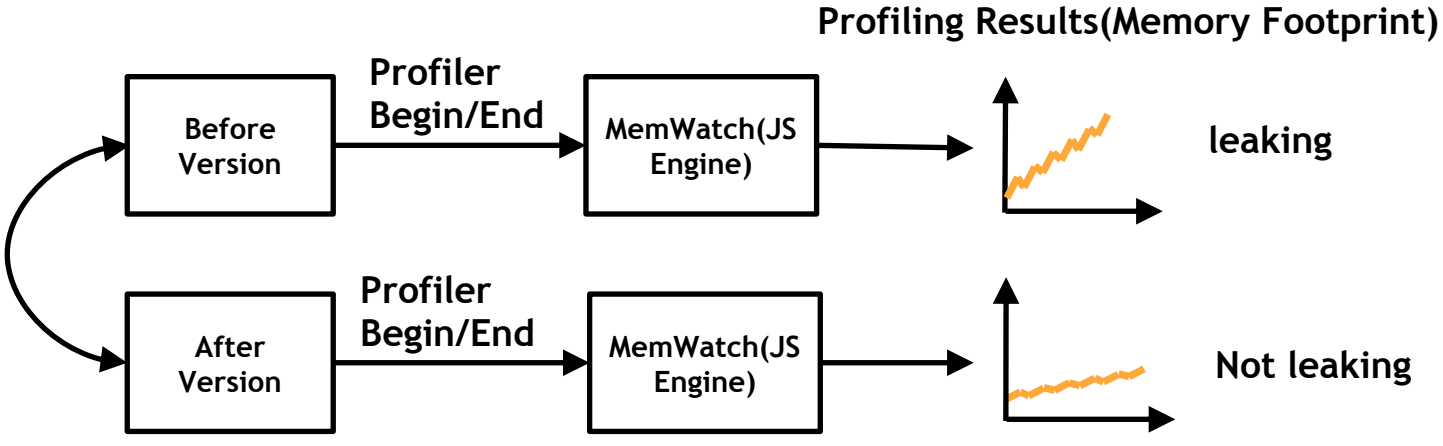
GNU Patch

```
5,18c1,16
<(original code for getObjsInArray)
---
> function getObjsInArray(obj, array) {
>  var foundObjs = [];
>  var keys = Object.keys(obj);
>  for (var i=0; i < array.length; i++){
>   for (var j = 0, l = keys.length; j <
  l; j++) {
>   var key = keys[j];
>   if (obj[key] === array[i][key]) {
>     foundObjs.push(array[i]);
>     break;
> }}}
> return foundObjs;
> }
```

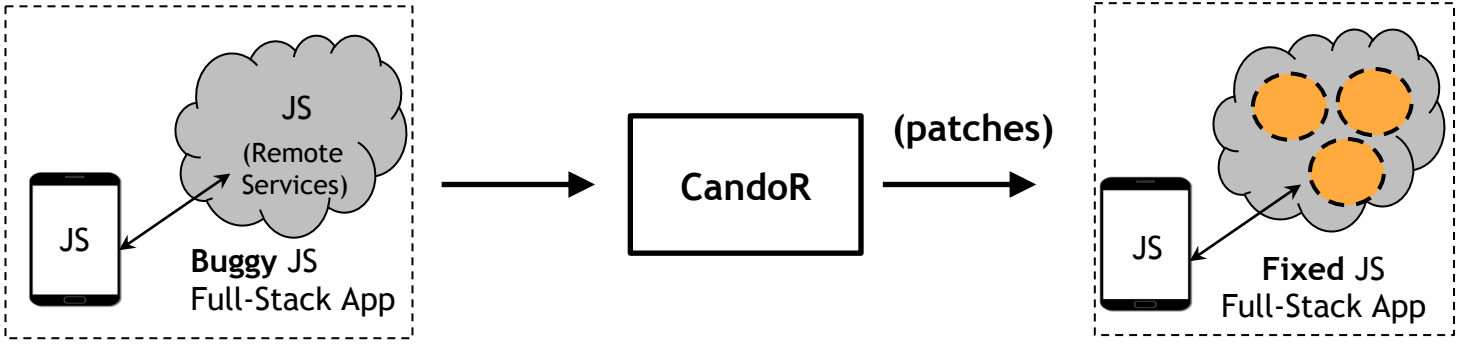


VIRGINIA TECH

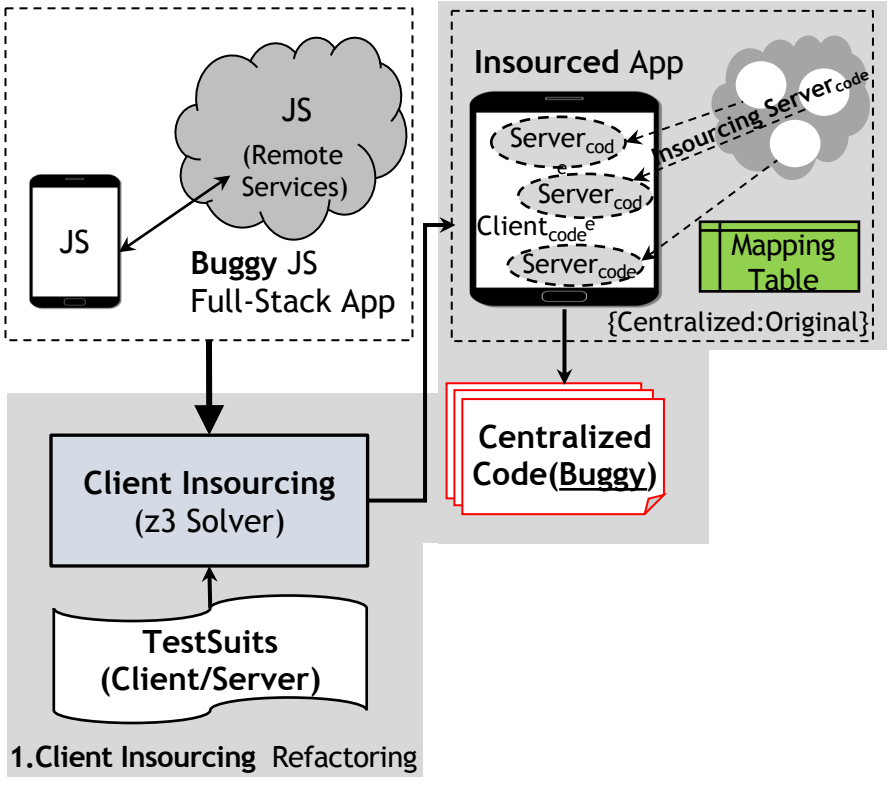
Memory Leak (memwatch[11]*)*

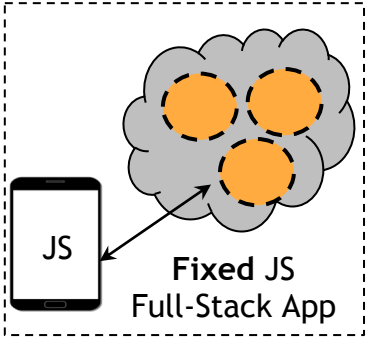


VIRGINIA TECH

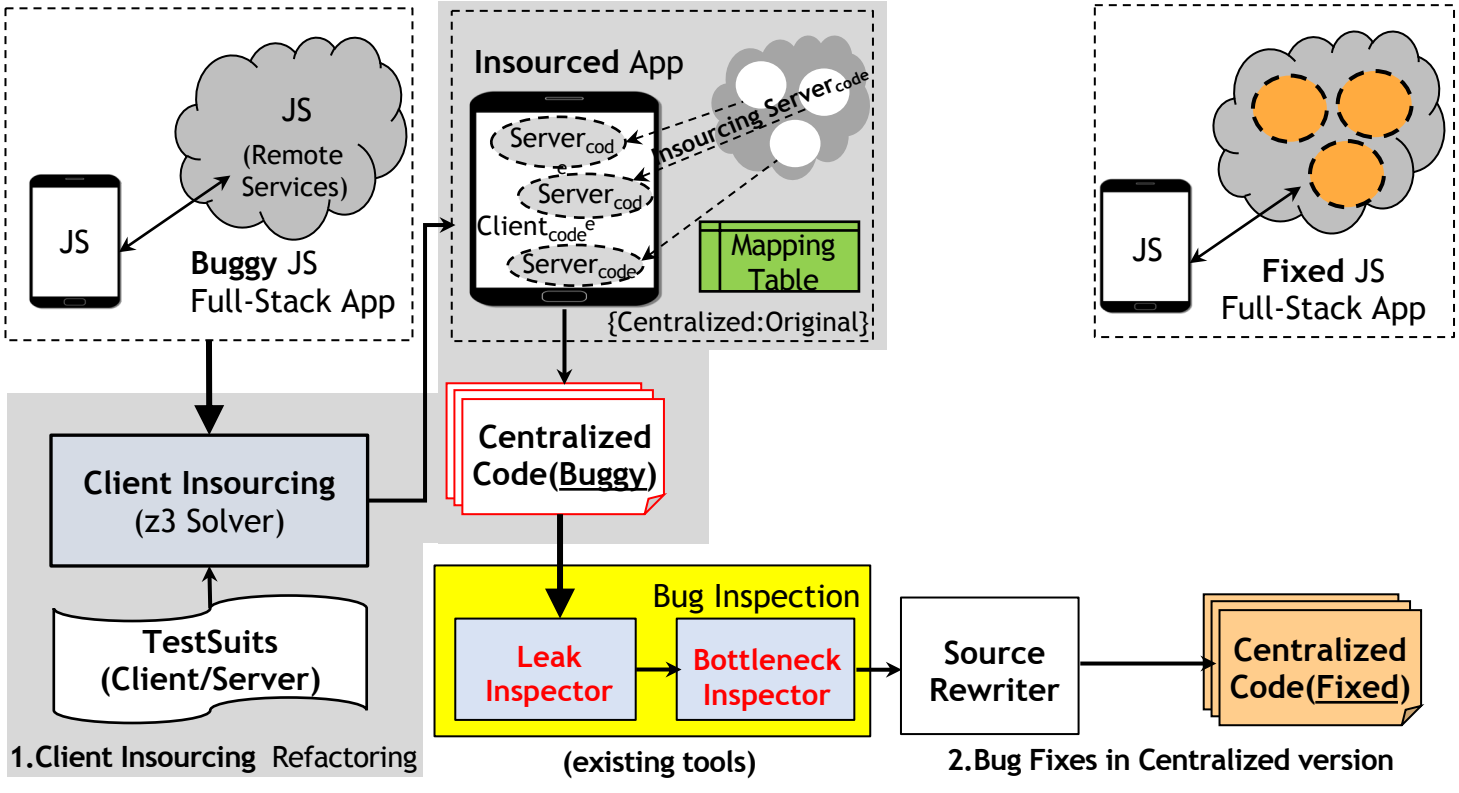


VIRGINIA TECH

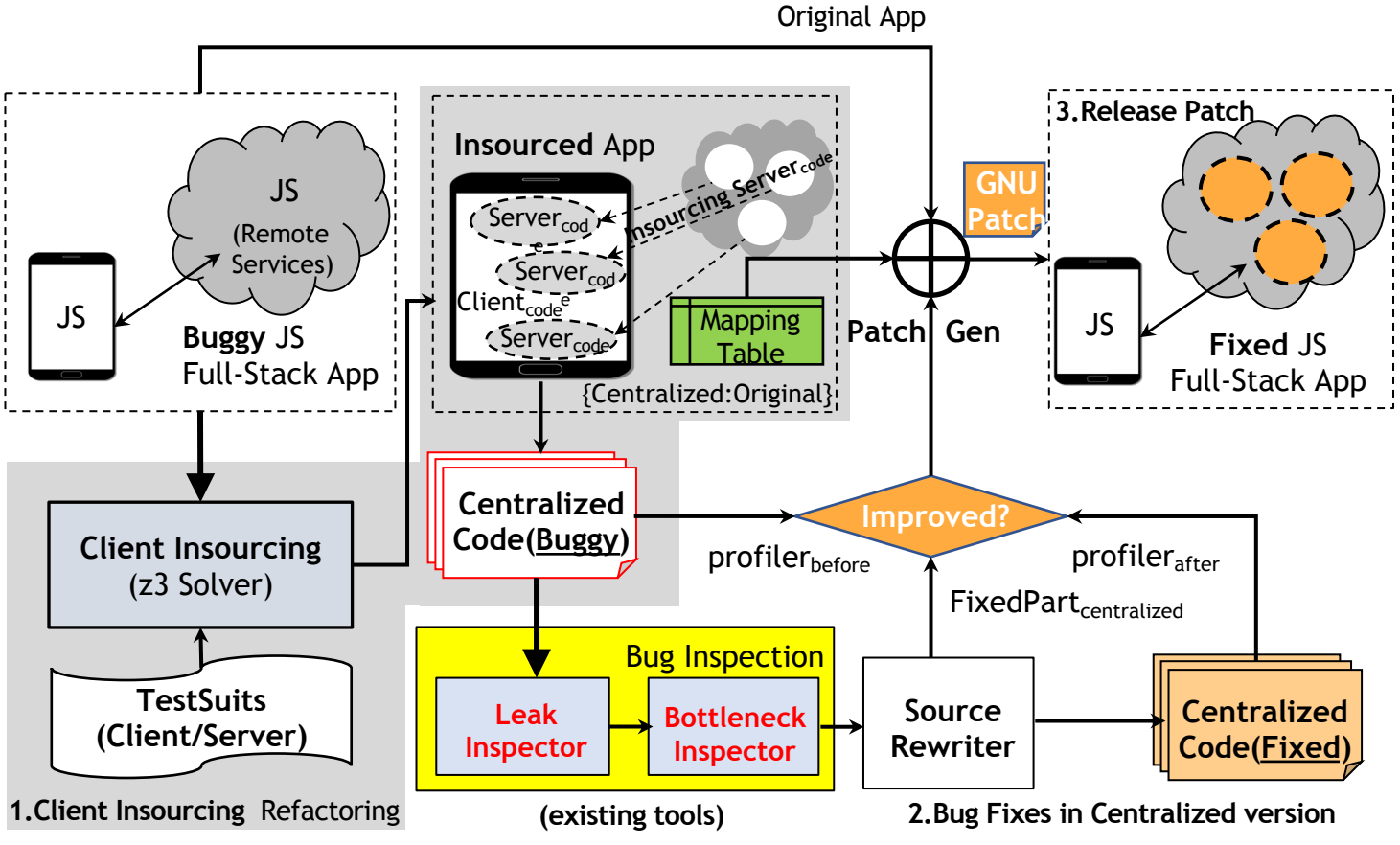




VIRGINIA TECH



VIRGINIA TECH



VIRGINIA TECH

Evaluation: Subject Applications

Subject Apps	Remote Services	Bug Types
theBrownNode [19]	/users/search /users/search/id	Inefficient iteration (Performance Bottleneck)
Bookworm [3]	/api/ladywithpet /api/thedea /api/theredroom /api/thegift	Misused APIs (Performance Bottleneck)
Search_leak [16]	/search_leak	Memory leak
ionic2_realty_rest [15]	/properties/favorites	Memory leak

VIRGINIA TECH

Evaluation: Client Insourcing

Subject Apps	S_{ULOC}	C_{ULOC}	Remote Services	CI_{ULOC}
theBrownNode [19]	147	43	/users/search /users/search/id	37 36
Bookworm [3]	371	1814	/api/ladywithpet /api/thedea /api/theredroom /api/thegift	394 394 394 394
Search_leak [16]	34	13	/search_leak	17
ionic2_realty_rest [15]	453	387	/properties/favorites	24

VIRGINIA TECH

Evaluation: Performance Bottleneck

Remote Services	Bug Types	F_{ULOC}	P^{D}_{before} (P^{CI}_{before})	P^{D}_{after} (P^{CI}_{after})	$P^{d}_{improved}$ ($P^{CI}_{improved}$)
/users/search	Inefficient iteration	31	0.36ms	0.26ms	27.8%
			(0.19ms)	(0.13ms)	(31.58%)
/users/search/id	Inefficient iteration	31	1.7ns	1.19ns	29.53%
			(2.5ns)	(1.63ns)	(34.8%)
/api/ladywithpet	Misused APIs	18	5.89ms	4.99ms	15.28%
			(2.74ms)	(2.24ms)	(18.13%)
/api/thedea	Misused APIs	18	5.63ms	4.82ms	14.39%
			(2.71ms)	(2.25ms)	(16.97%)
/api/theredroom	Misused APIs	18	0.65ms	0.53ms	18.06%
			(1.87ms)	(1.56ms)	(16.58%)
/api/thegift	Misused APIs	18	1.17ms	1.04ms	11.11%
			(0.36ms)	(0.31ms)	(13.89%)

VIRGINIA TECH

Evaluation: Performance Bottleneck

Remote Services	Bug Types	F_{ULOC}	P^{D}_{before} (P^{CI}_{before})	P^{D}_{after} (P^{CI}_{after})	$P^{d}_{improved}$ ($P^{CI}_{improved}$)
/users/search	Inefficient Iteration	18	0.36ms	0.26ms	27.8%
			(0.19ms)	(0.13ms)	(31.58%)
/users/search/id	Inefficient Iteration	18	1.7ns	1.18ns	29.53%
			(2.5ns)	(1.63ns)	(34.8%)
/api/ladywithpet	Misused APIs	18	5.89ms	4.99ms	15.28%
			(2.74ms)	(2.24ms)	(18.13%)
/api/thedea	Misused APIs	18	5.63ms	4.82ms	14.39%
			(2.71ms)	(2.25ms)	(16.97%)
/api/theredroom	Misused APIs	18	0.65ms	0.53ms	18.06%
			(1.87ms)	(1.56ms)	(16.58%)
/api/thegift	Misused APIs	18	1.17ms	1.04ms	11.11%
			(0.36ms)	(0.31ms)	(13.89%)

Distributed Version(Original)

Centralized Version(Proxy)

$$\frac{P_{before} - P_{after}}{P_{before}}$$

VIRGINIA TECH

Evaluation: Performance Bottleneck $\frac{P_{before} - P_{after}}{P_{before}}$

Remote Services	Bug Types	F_{ULOC}	P^{D}_{before} (P^{CI}_{before})	P^{D}_{after} (P^{CI}_{after})	$P^{D}_{improved}$ ($P^{CI}_{improved}$)
/users/search	Inefficient iteration	31	0.36ms (0.19ms)	0.26ms (0.13ms)	27.8% (31.58%)
/users/search/id	Inefficient iteration	31	1.7ns (2.5ns)	1.19ns (1.63ns)	29.53% (34.8%)
/api/ladywithpet	Misused APIs	18	5.89ms (2.74ms)	4.99ms (2.24ms)	15.28% (18.13%)
/api/thedea	Misused APIs	18	5.63ms (2.71ms)	4.82ms (2.25ms)	14.39% (16.97%)
/api/theredroom	Misused APIs	18	0.65ms (1.87ms)	0.53ms (1.56ms)	18.06% (16.58%)
/api/thegift	Misused APIs	18	1.17ms (0.36ms)	1.04ms (0.31ms)	11.11% (13.89%)

VIRGINIA TECH

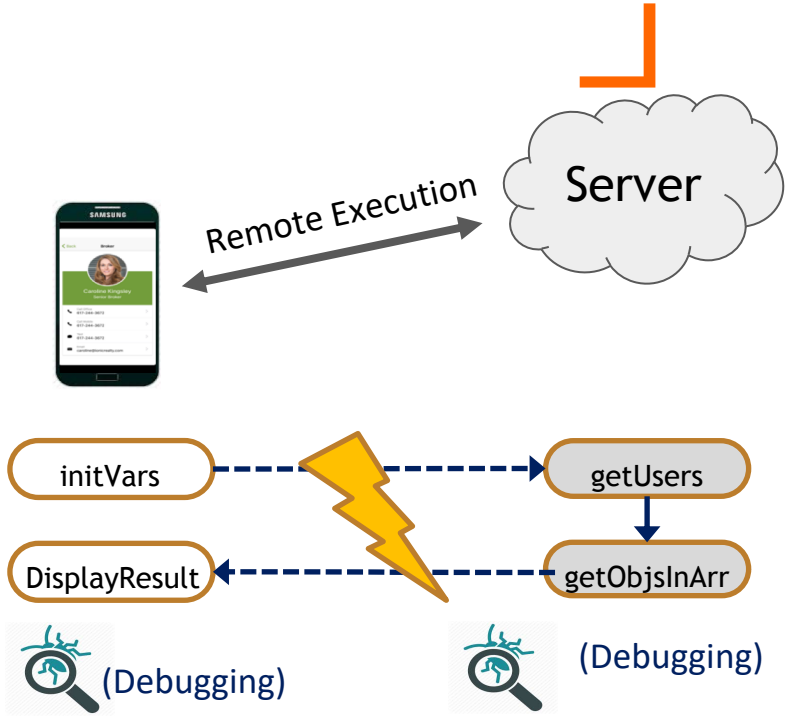
Evaluation: Memory Leak

Remote Services	Bug Types	F_{ULOC}	P^{D}_{before} (P^{CI}_{before})	P^{D}_{after} (P^{CI}_{after})	$P^{D}_{improved}$ ($P^{CI}_{improved}$)
/search_leak	Memory leak	24	619.10kb	519.13kb	16.15%
			(476.16kb)	(409.10kb)	(14.08%)
/properties/favorites	Memory leak	42	824.62kb	511.37kb	37.99%
			(1431.28kb)	(922.51kb)	(35.54%)

VIRGINIA TECH

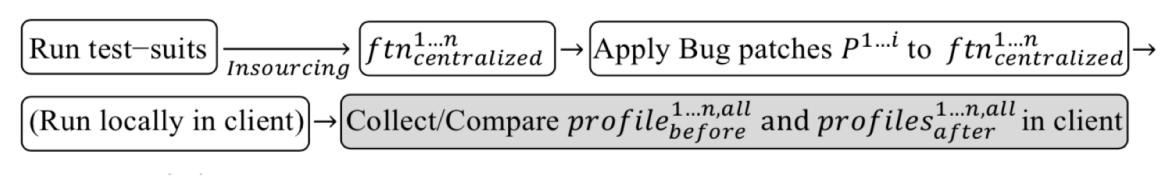
Debugging (State of the Practice)

Locate $ftn_{remote}^{n^{th}}$'s entry/exit points in server → Insert Start/End Profiler for $profile_{before}^{n^{th},server}$ → Insert Start/End Profiler in client $ftn_{client}^{n^{th}}$ for $profile_{before}^{n^{th},client}$ → (Run HTTP methods in client) → Collect $profile_{before}^{n^{th},server}$ / $profile_{before}^{n^{th},client}$ for $profile_{before}^{n^{th},all}$ → Apply Bug patches $P^{1...i}$ to $ftn_{remote}^{n^{th}}/ftn_{local}^{n^{th}}$ → (Run HTTP methods in client) → Collect $profile_{after}^{n^{th},server}$ / $profile_{after}^{n^{th},client}$ for $profile_{after}^{n^{th},all}$ $\xrightarrow[repeat\ (n-1...1)]{}$*... → Compare $profile_{before}^{1...n,all}$ and $profiles_{after}^{1...n,all}$

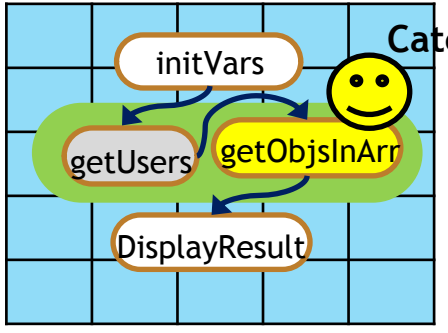


VIRGINIA TECH

Debugging (with CandoR)



(Debugging)



VIRGINIA TECH

Related Work: Debugging Distributed Web Applications

- Client-side scripting to automatically exercise UI elements and to conduct state-based testing [8,10,12,13]
  - Approximate server-side app logic as simple states
- Record and Replay(R&R) is an execution framework that efficiently captures distributed execution traces [1,14]
- Performance & Memory debugging Tools for JavaScript
  - MemInsight, JITProf, DLint, JSweeter, BLeak, MemWatch
  - Currently, all these approaches need to be applied separately to the server or client parts of full-stack JavaScript applications

Conclusion, QnA

- We have presented a new debugging approach—**CandoR**—that facilitates the debugging of full-stack JavaScript applications.
- We plan to generalize our approach to **multi-lingual applications**, server and client are written by different programming languages.