Blended Program Analysis for Improving Reliability of Real-world Applications

Dr. Barbara G. Ryder
J. Byron Maupin Professor of Engineering
Virginia Tech

Collaborators: Bruno Dufour (Rutgers), Gary Sevitsky (IBM Research), Marc Fisher II (VT), Ben Wiedermann (VT), Shiyi Wei (VT), S. Basu (ug-Lafayette), Luke Marrs (ug-VT); Funded by IBM Open Collaborative Research Program and NSF 08-0811518

Outline

- What is program analysis?
- Challenge of modern software applications
- What is blended program analysis?
- Examples of blended analysis
  - Performance diagnosis for framework-based applications (Java)
  - Data integrity & program understanding for webpage codes (JavaScript)
- Summary

What is Program Analysis?

- Historically,
  - Static program analysis was used for code optimization during compilation
    - Gave a safe approximation of all possible program behaviors, without running the program
    - Allows checking of specific program properties
  - Dynamic program analysis was used for tracking program behavior at runtime
    - Gathered information about program on one execution

What is Program Analysis?

- Static Analysis
  - Input: code
  - Output: model of semantics of program
  - When: At compile time
  - Cost: High
  - Goal: code optimization, property validation, program understanding, test case generation...
- Dynamic Analysis
  - Input: trace of execution(s)
  - Output: an observed property
  - When? At runtime
  - Cost: Low
  - Goal: Debugging, uncovering code dependences,test coverage...

Challenge: Tools for Modern SW Applications

- Framework-based software systems (Java)
  - Transactional, complex, many libraries/ components
    - E.g., personal financial managers, e-commerce applications, information records managers
  - Performance problems difficult to isolate
  - Need understanding of business logic as well as run-time behavior

Dynamic Language Constructs

- Java
  - Reflection – values from run-time environment influence program behavior (e.g., in dynamic class loading)

```
//read name of class to be dynamically loaded
//from command line*
Class a = Class.forName(args[0]); …
Method mainMethod = findMain(a);
mainMethod.invoke( … );
```

*http://media.techtarget.com/tss/static/articles/content/dm_classForname/DynLoad.pdf

Challenge: Tools for Modern SW Applications

- Websites (JavaScript)
  - Constructed from combination of static and dynamically loaded/generated code
  - Code can be constructed at runtime and then executed
  - Program understanding (for maintenance), testing, and validating data integrity are hard

Dynamic language constructs

- JavaScript
  - Ability to construct a URL at runtime and then load that webpage
  - Functions with variable numbers of arguments
  - Ability to write code at runtime and execute it with *eval* statement

```
//xmlhttp is an instance of XMLHttpRequest
eval(xmlhttp.response());
```

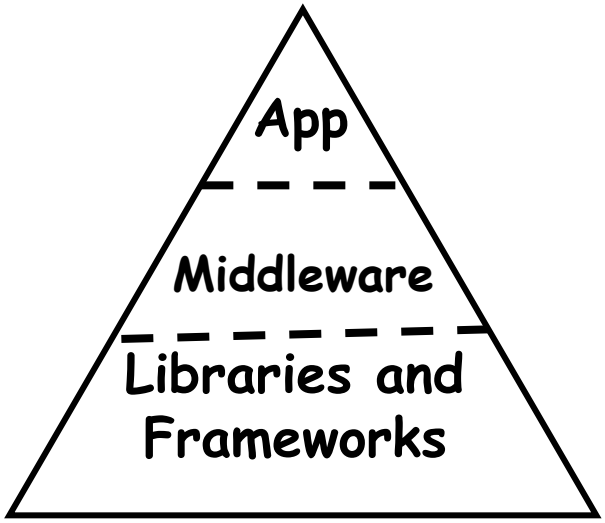
What is Blended Program Analysis?
ISSTA07, FSE08, ICSM10, CS@VT TR-12-18(2012)

- A way to integrate dynamic and static analyses to obtain scalability and precision for specific problems
  - Use of a dynamic program representation
    - Reflects call structure of execution(s)
  - Use of light-weight dynamic information
    - To prune (some) infeasible paths
    - To tie static analysis information to actual run-time objects
    - To deal with dynamic program constructs

Outline

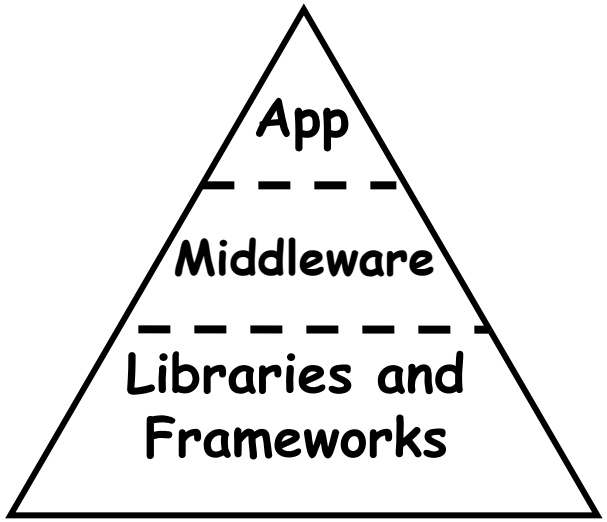
- Blended analysis for performance diagnosis in framework-intensive applications (Java)

Framework-based Applications



- Application is an iceberg
  - Bulk of the code in libraries and frameworks
  - Genre not commonly addressed by research community
  - E.g., financial planning services, e-commerce sites, online reservation systems, Tomcat-based systems software
- Programs are not just large, but are more complex in interactions between frameworks
- Performance problems span multiple layers

Framework-based Applications

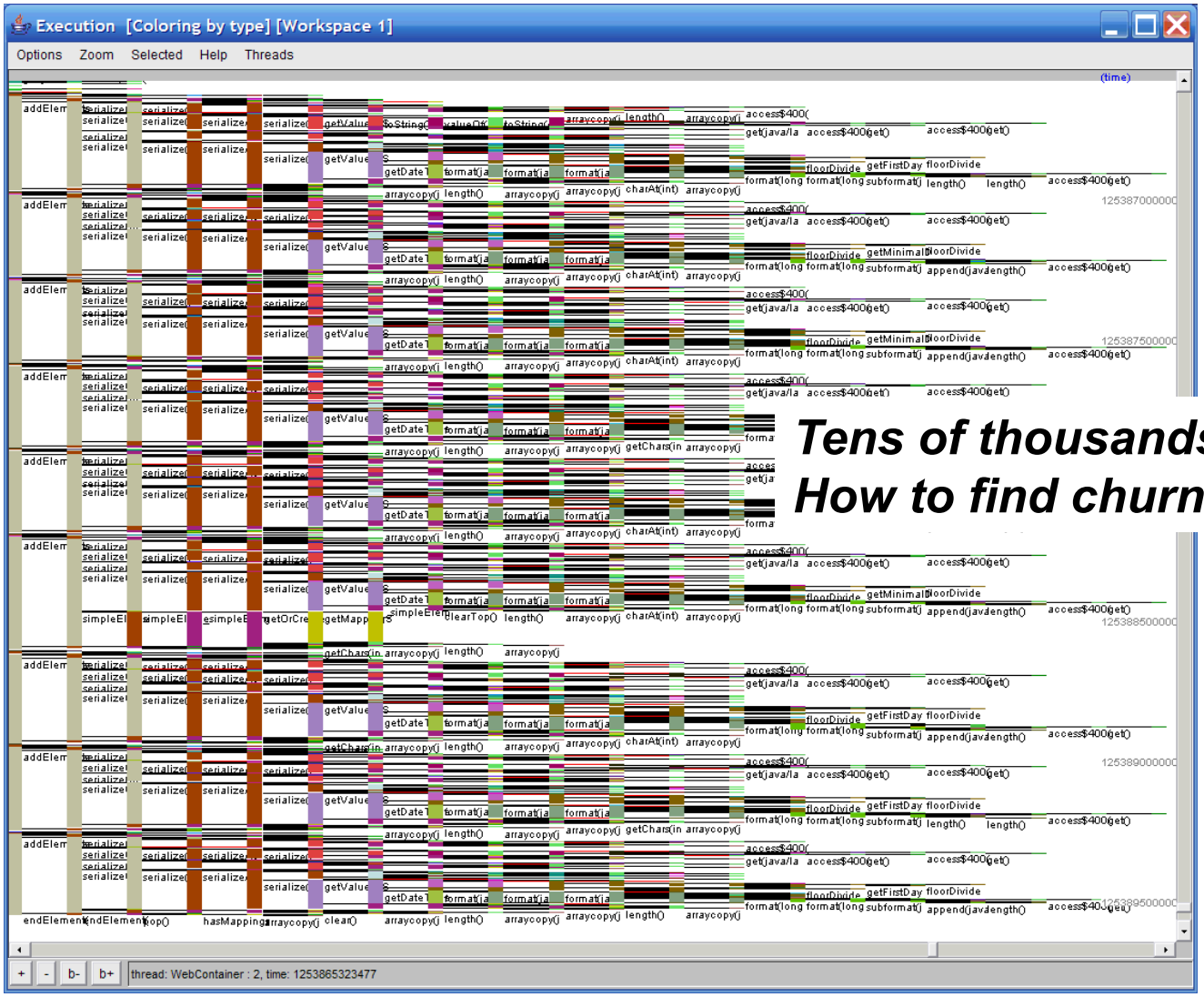


- Software characteristics
  - Not amenable to static analyses
    - Not scalable -- too complex
  - Not amenable to dynamic analysis
    - Too intrusive into execution for production codes
  - Application's main function often is data transformation
- Goal: design analyses for performance diagnosis of these systems

Goal: Find Object Churn

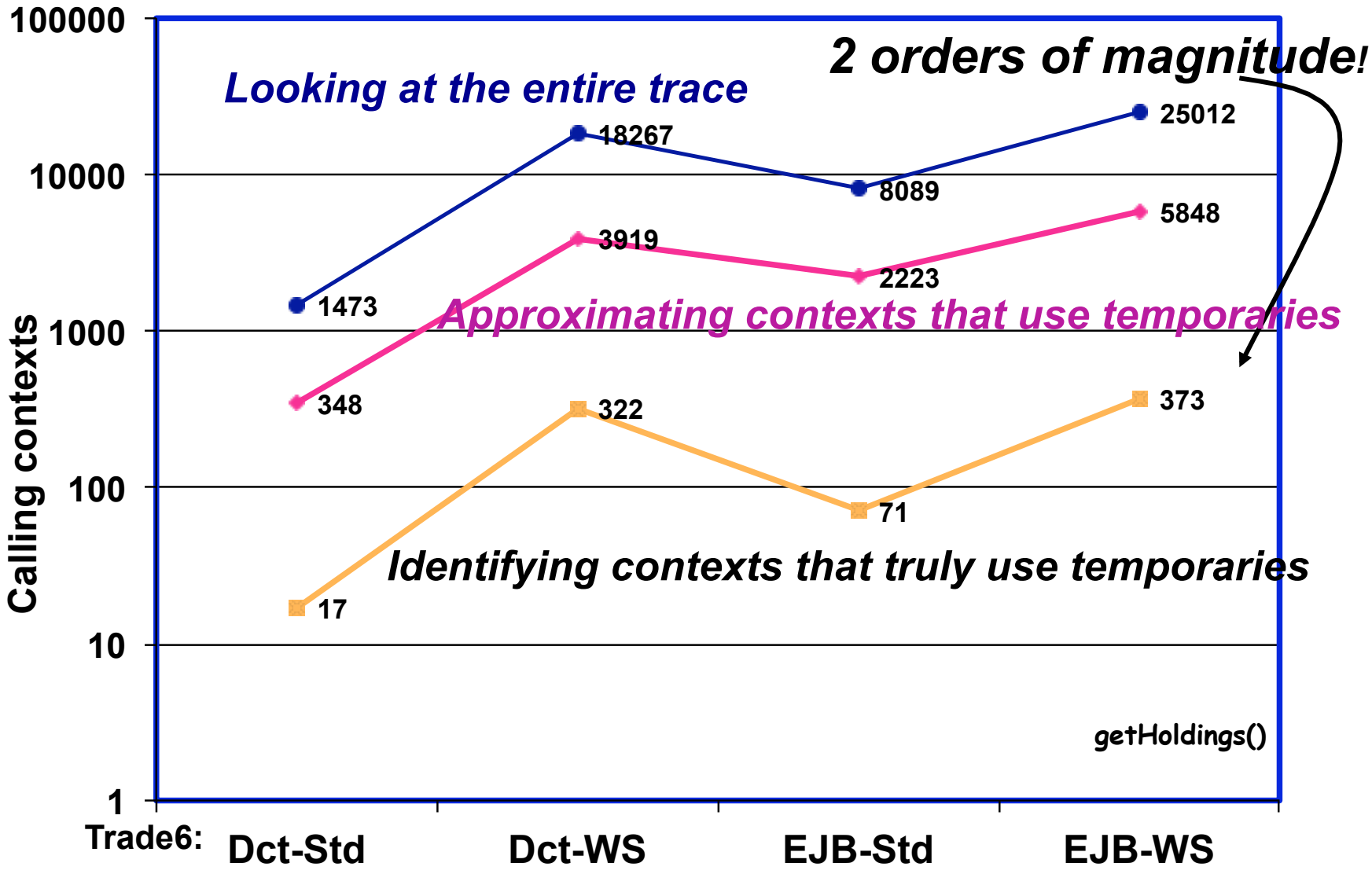
- **Identify execution contexts with excessive use of temporaries**
  - Based on total number of *instances*
  - Not the same as finding often-executed allocation sites
  - Need to identify temporary objects and to approximate "object lifetime"
- **Elimination strategies**
  - Optimize the frequent use of frameworks and libraries together
  - Introduce caching for temporary data structures
  - Code specialization

Current Practice: Jinsight Trace of HoldingDataBean_Ser.serialize()

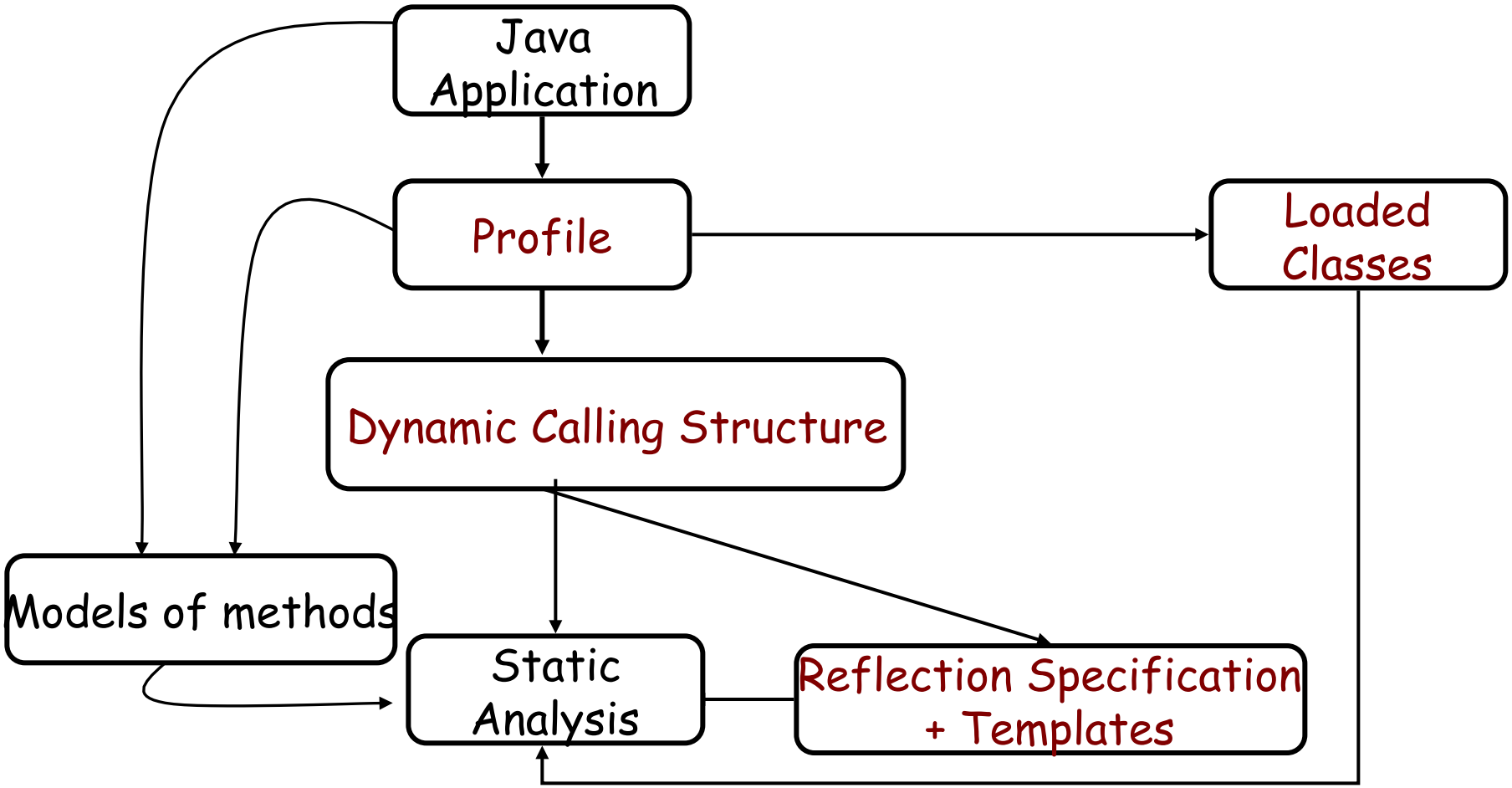


Tens of thousands of calls
How to find churn locality?

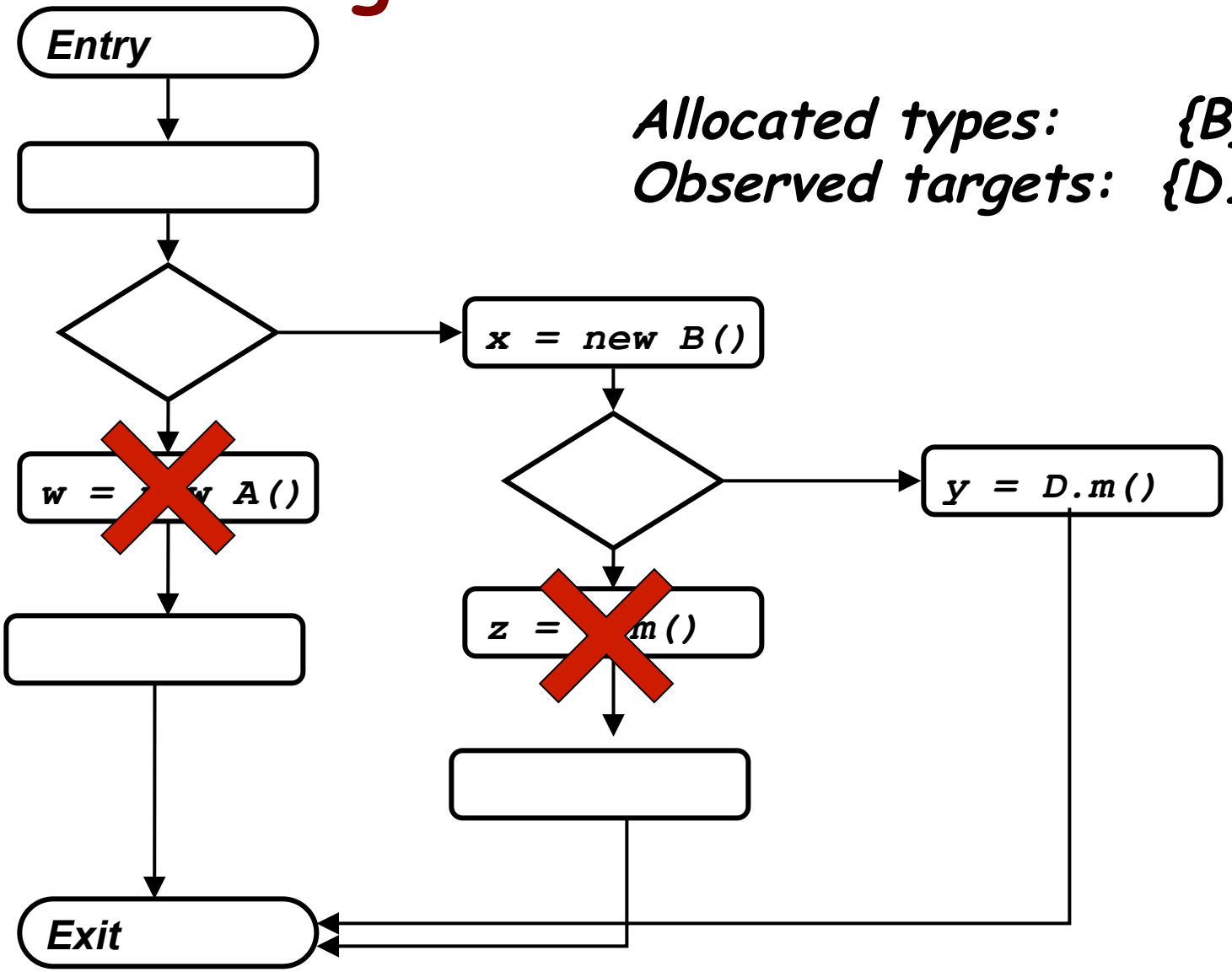
Blended Analysis - Scalability



Blended Analysis Paradigm

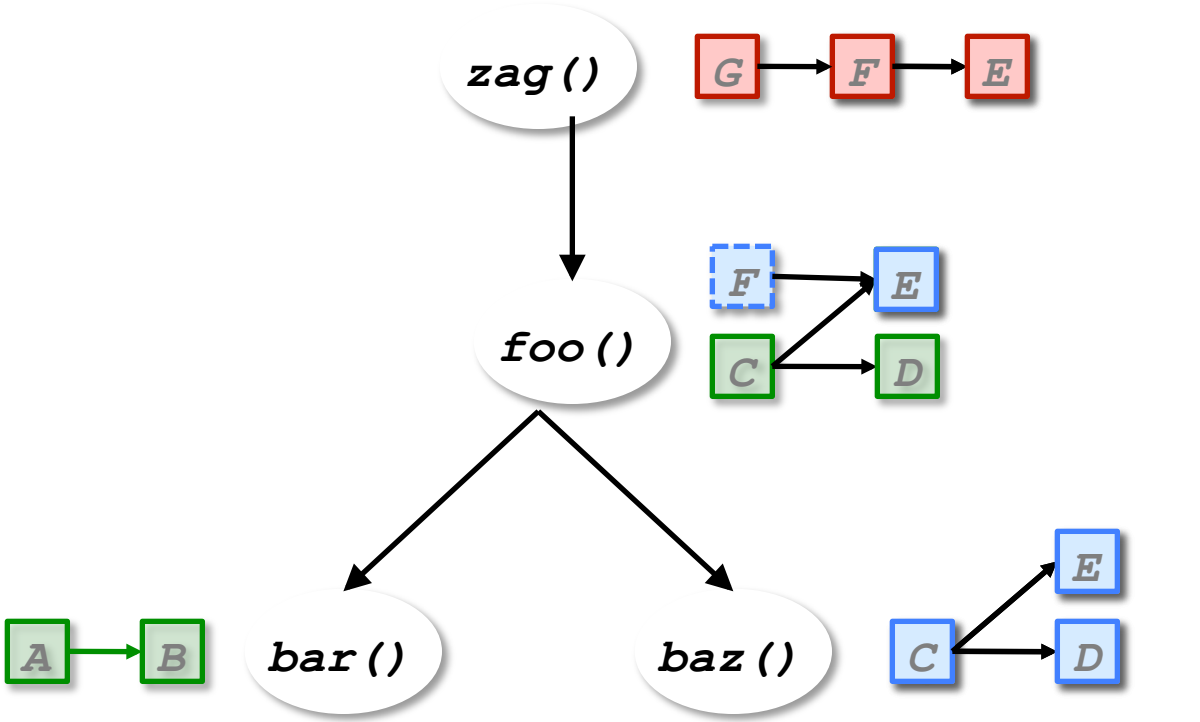


Pruning Code in Methods



Allocated types: {B}
Observed targets: {D.m}

Escape Analysis, by Example




```
void bar() {
    a = new A();
    a.x = new B();
}
C baz() {
    c = new C();
    c.y = new D();
    c.z = new E();
    return c;
}
void foo(F f) {
    c = f.w;
    f.w = c.z;
}
void zag() {
    F f = new F();
    foo(f);
    G.global = f;
}
```

Disposition

A

B

C

D

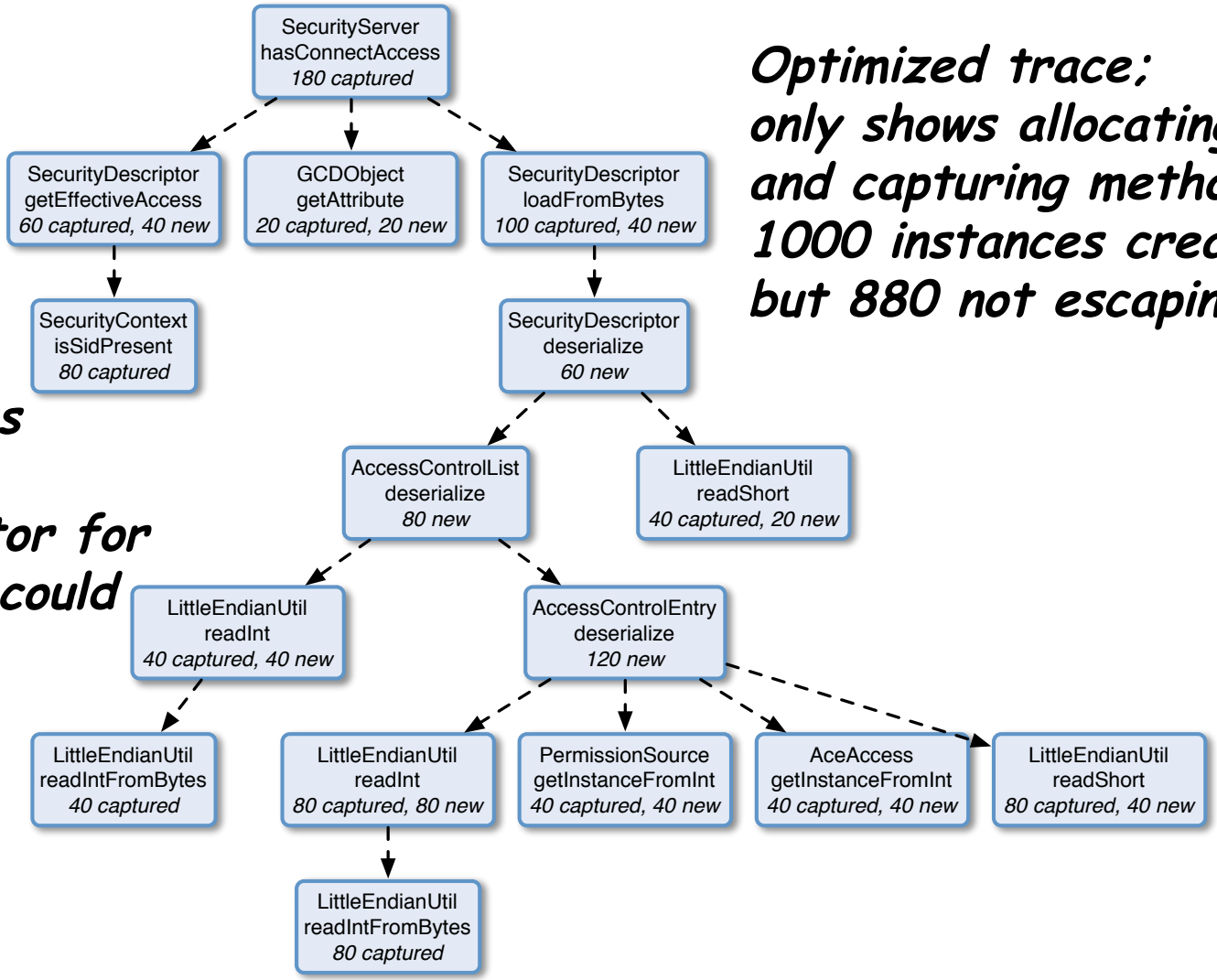
E

F

G

Captured Arg-escaping Globally escaping

Invocation tree for hasConnectAccess



Optimized trace; only shows allocating and capturing methods 1000 instances created but 880 not escaping

Each invoke of hasConnectaccess has a unique SecurityDescriptor for an ID instance; could cache w/I ID;

Metrics

Designed new metrics for blended escape analysis

- Measure effectiveness of pruning
  - Scalability of analysis – % of blocks in methods pruned

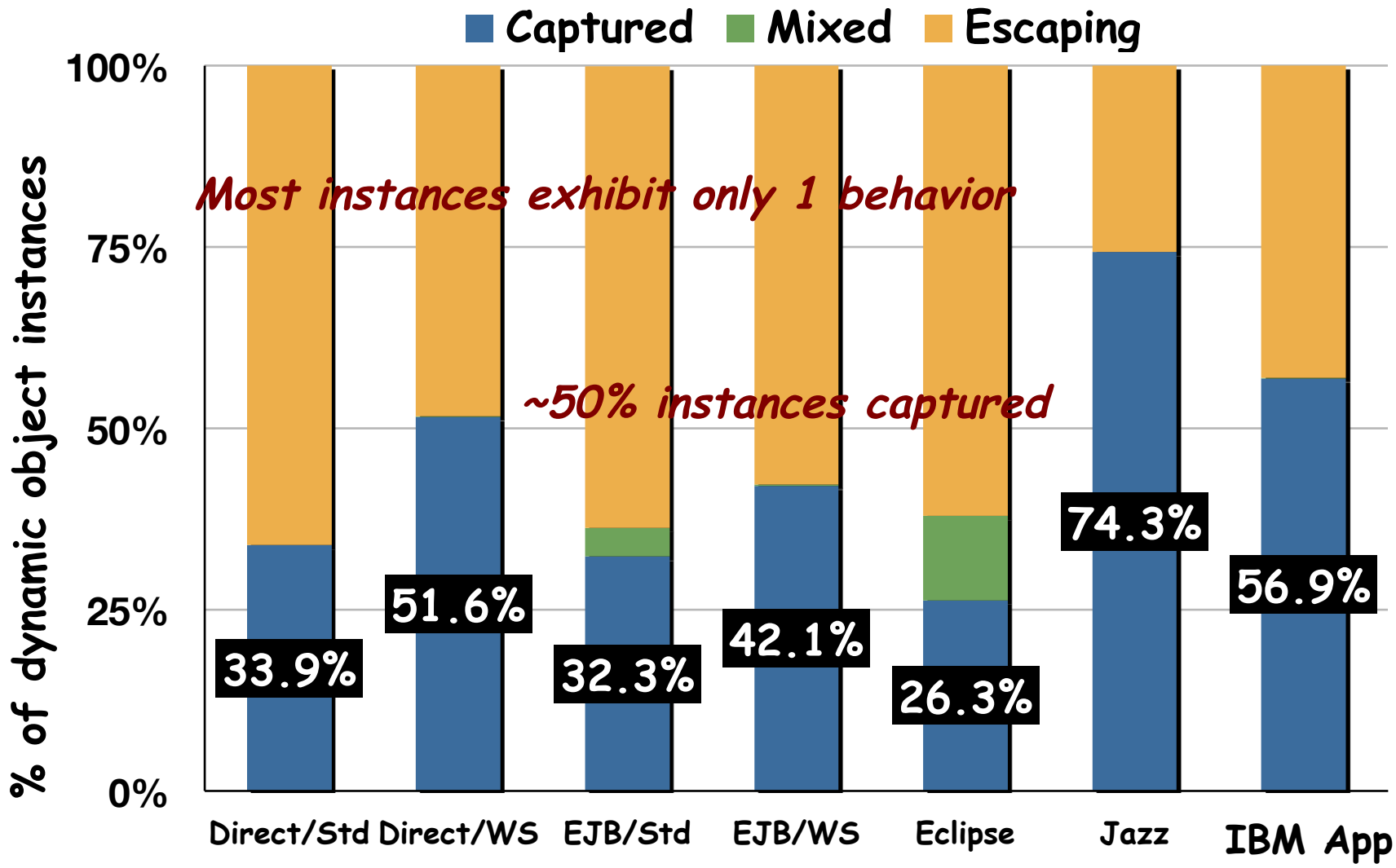
Scalability

Benchmark	Analysis time (h:m:s)		Speedup	%Pruned
	No pruning	Pruned		
Direct-Std	00:00:18	00:00:17	1.1	45%
Direct-WS	01:34:01	00:04:41	20.0	40%
EJB-Std	00:04:24	00:01:46	2.5	43%
EJB-WS	N/A	29:23:16	N/A	43%
Eclipse	24:37:12	06:37:22	3.7	29%
Jazz	02:49:55	00:39:06	4.3	27%
IBM Appln	00:04:35	00:02:05	2.2	39%

Metrics

- Measure usage of temporaries
  - Disposition- categorizes instances as globally: escaping, captured, mixed

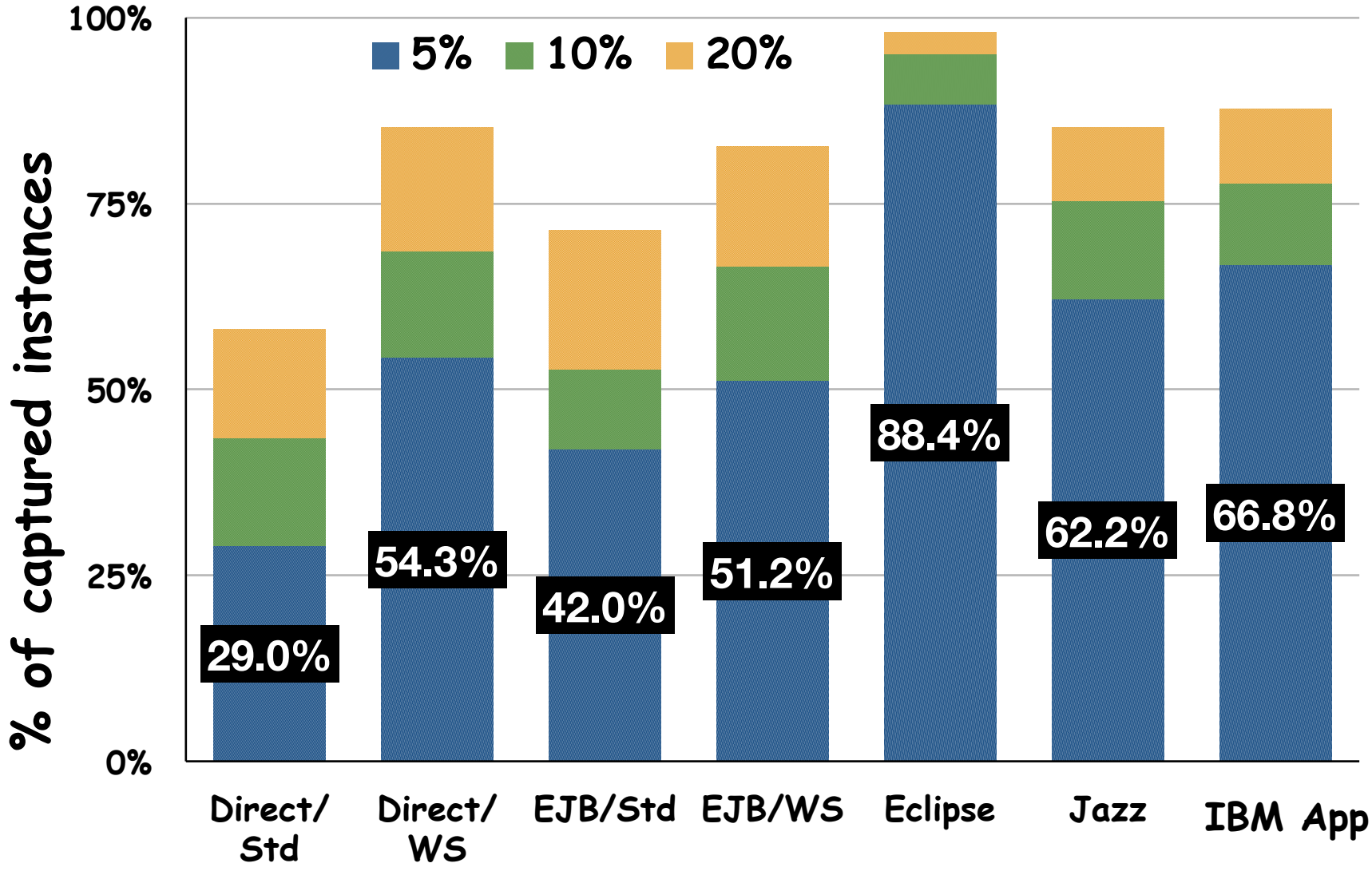
Disposition of Instances



Metrics

- Measure usage of temporaries
  - Concentration- measures locality of temporary usage

Concentration of Captured Instances



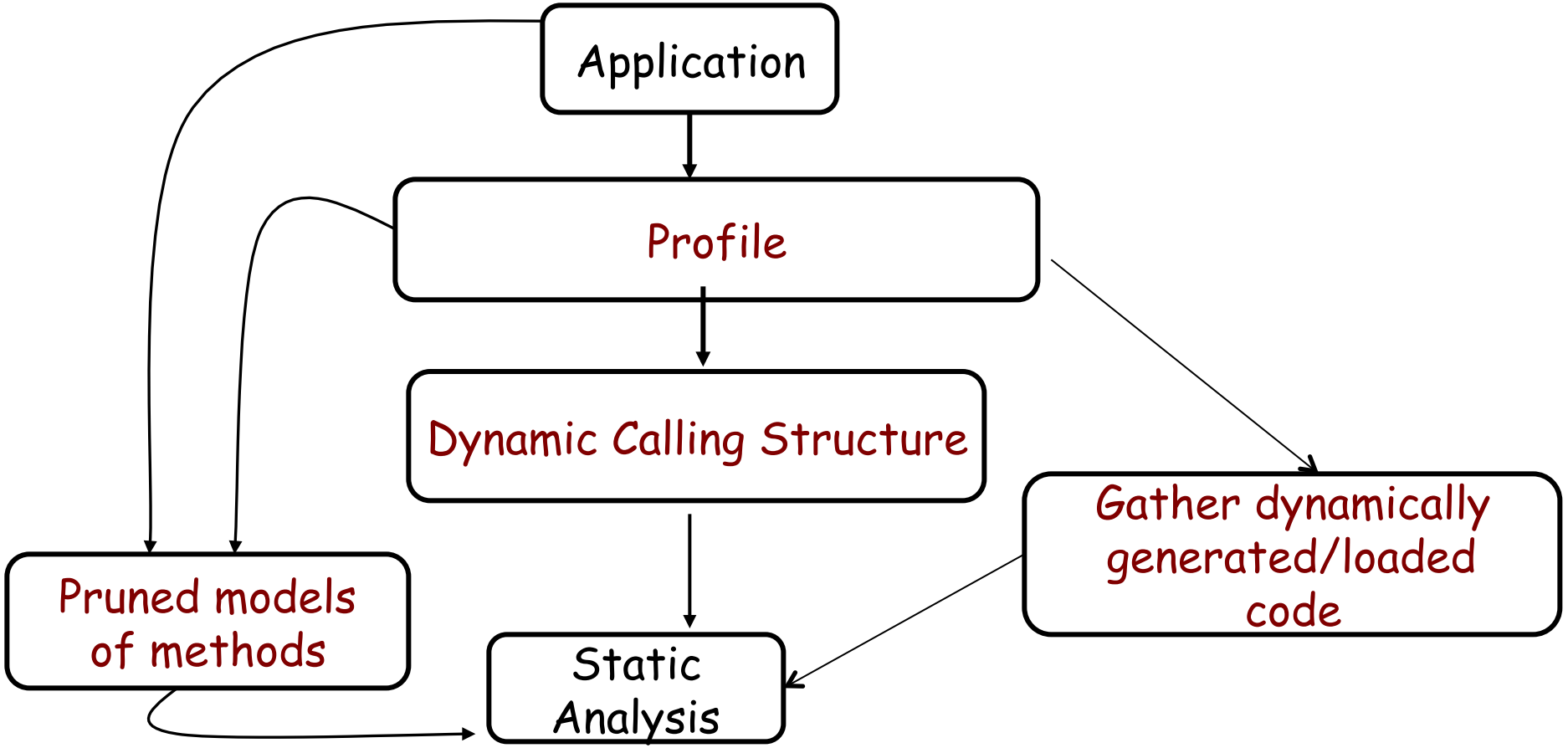
Outline

- Blended analysis for capturing effects of dynamically generated or dynamically loaded code (JavaScript)

JavaScript – Website Glue Code

- JavaScript is *lingua franca* of client-side applications
  - 98 out of 100 most popular websites use JavaScript (Guarnieri et al, ISSTA11)
  - Use of dynamic features is evident in websites (Richards et al, ECOOP11, PLDI10; Zorn et al, WebApps10)

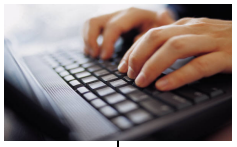
Blended Analysis Paradigm for JavaScript



How to Profile a Website for Analysis?

- Run several executions of the website, to explore its behaviors
- Each execution is comprised by a set of page traces
- Gather all page traces for the same webpage together and consider them a single JavaScript program for analysis

JavaScript Analysis Framework



Execution collector

Trace selector

Dynamic analysis

Static analyzer

Solution integrator

JS Program Solution

Static analysis

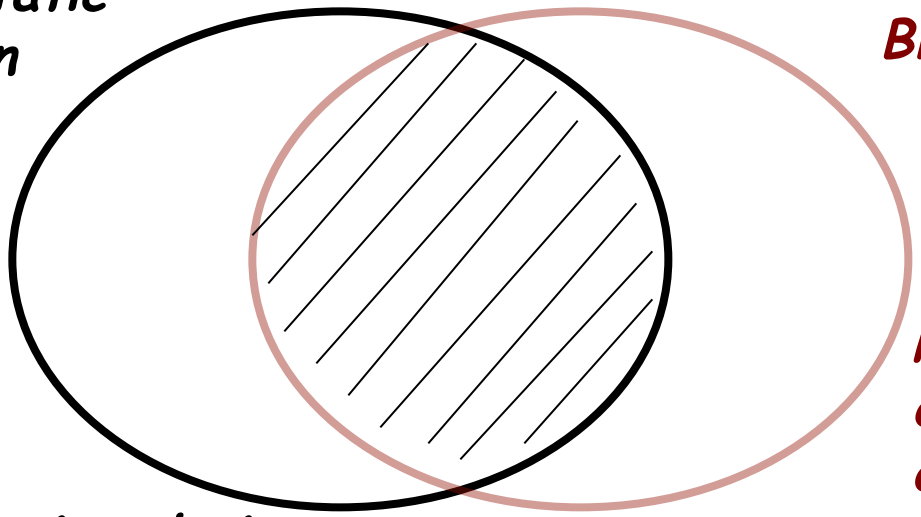
Tainted Input Analysis

- Integrity violation – allowing user input to reach a sensitive operation
  - Tainted Source: user has control of its value
  - Sensitive operation: can affect behavior of website or browser
  - Source-sink pair reported if there is a possible dataflow between them (ignoring sanitizers)
- Hyp: blended tainted input analysis will report fewer false alarms and more true positives than pure static tainted input analysis

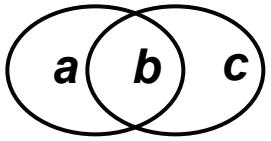
Benchmarks1

Website	Page count	Trace count
bing.com	19	30
twitter.com	14	30
linkedin.com	12	30
qq.com	18	30
wordpress.com	23	30
sina.com.cn	16	30
163.com	22	30
cnn.com	18	30
msn.com	18	30
conduit.com	10	16
imdb.com	10	18
myspace.com	10	24
sohu.com	10	19
xing.com	10	18
xunlei.com	10	22
zedo.com	10	16
washingtonpost.com	10	27
pconline.com.cn	10	21
Average	**14**	**25**

Blended vs Pure Static Analysis



Shaded area is part of solution found by pure static and blended



Tainted Input Results

Website	Pure Static true soln	Pure Static false alarm	Blended true soln	Blended false alarm
live.com			1	
youtube.com	1		1	
myspace.com			1	*false negatives*
sohu.com	2	1	2	
xunlei.com	3	*false positives*	3	
msn.com			1	
bing.com		1		
totals	6	2	9	

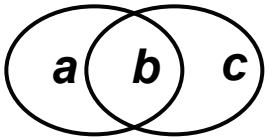
Statement-level Side-effects (ST-MOD) For Program Understanding

- Want to know the number of objects whose f field value may be changed at statement: *x.f=*
  - Helpful in navigating unfamiliar code
- How solve?
  - Find which objects o that x can point to
  - Find which objects o.f can point to

Benchmarks2

Website	Page count	Trace count	*eval* page	variadic function
google.com	203	2104	52	177
facebook.com	138	1098	23	65
youtube.com	122	579	19	29
yahoo.com	52	265	21	13
baidu.com	49	147	6	16
wikipedia.org	67	130	0	3
live.com	54	226	10	44
blogger.com	24	146	6	7
totals	**709**	**4695**	**137**	**354**

Comparison: Pure Static to Blended Solutions
(points-to)



Website	% Coverage of pure static solution	%Additional results
google.com	89.7	5.9
facebook.com	85.3	7.5
youtube.com	89.1	9.9
yahoo.com	78	9.8
baidu.com	93	6.7
wikipedia.org	92.1	none
live.com	81.8	7.5
blogger.com	83.8	1.4
mean	**86.6**	**7.0**

ST-MOD Solutions

Websites	Pure Static	Blended	
	Average number of objects in static code	Average number of objects in static code	Average number of objects in dynamic code
google.com	5.8	2.4	2.1
facebook.com	7.7	4.1	3.6
youtube.com	5.9	3.5	2.4
yahoo.com	5.2	2.5	2.8
baidu.com	2.6	1.4	1.8
live.com	2.9	1.6	2.2
blogger.com	4.5	2.8	2.3
average	4.9	2.6	2.5

Summary

- Modern SW applications require new approaches to program analysis - the engine behind SW tools
- New blended analysis paradigm seems promising for handling the more dynamic programming constructs
  - Performance diagnosis of framework-based apps
  - Understanding of website codes for enhancement
  - More experimentation and investigation needed to select best analysis for specific use

Challenges

- Expect more combinations of static and dynamic analyses for web & mobile apps
- Challenge: analyzing 3rd party apps (executables) in combo with known codes
- Challenge: analyzing event-driven codes
- Challenge: increased use of explicit concurrency will require new tools and supporting analyses

Thank You