Optimistic Hybrid Analysis

David Devecsery,

Peter M. Chen, Satish Narayanasamy, Jason Flinn

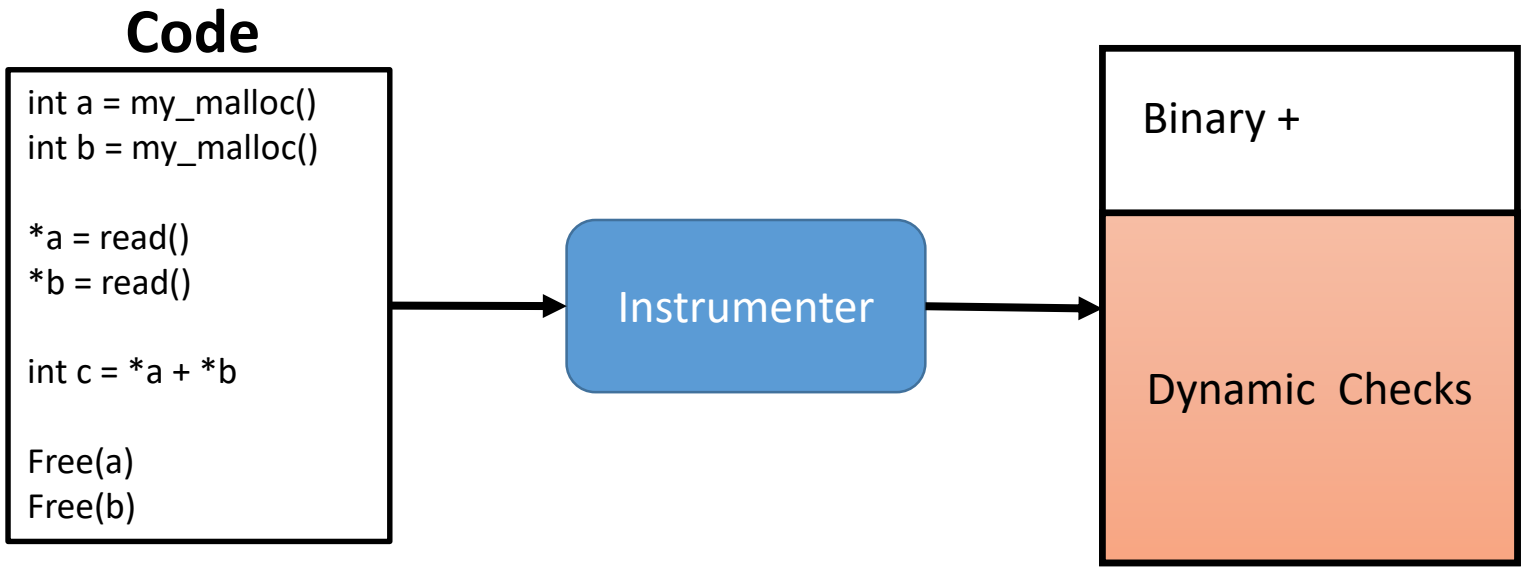
University of Michigan

Motivation – Problem

- Dynamic analysis is essential
  - Memory errors
    - Buffer overflow, use-after-free, double free, memory leak…
  - Information flow
    - Taint tracking, information providence, program slicing…
  - Race freedom
  - Type safety
  - Many more
- We want to enforce these properties in programs
  - Need precise (no false positives) and sound (no false negatives) program analysis

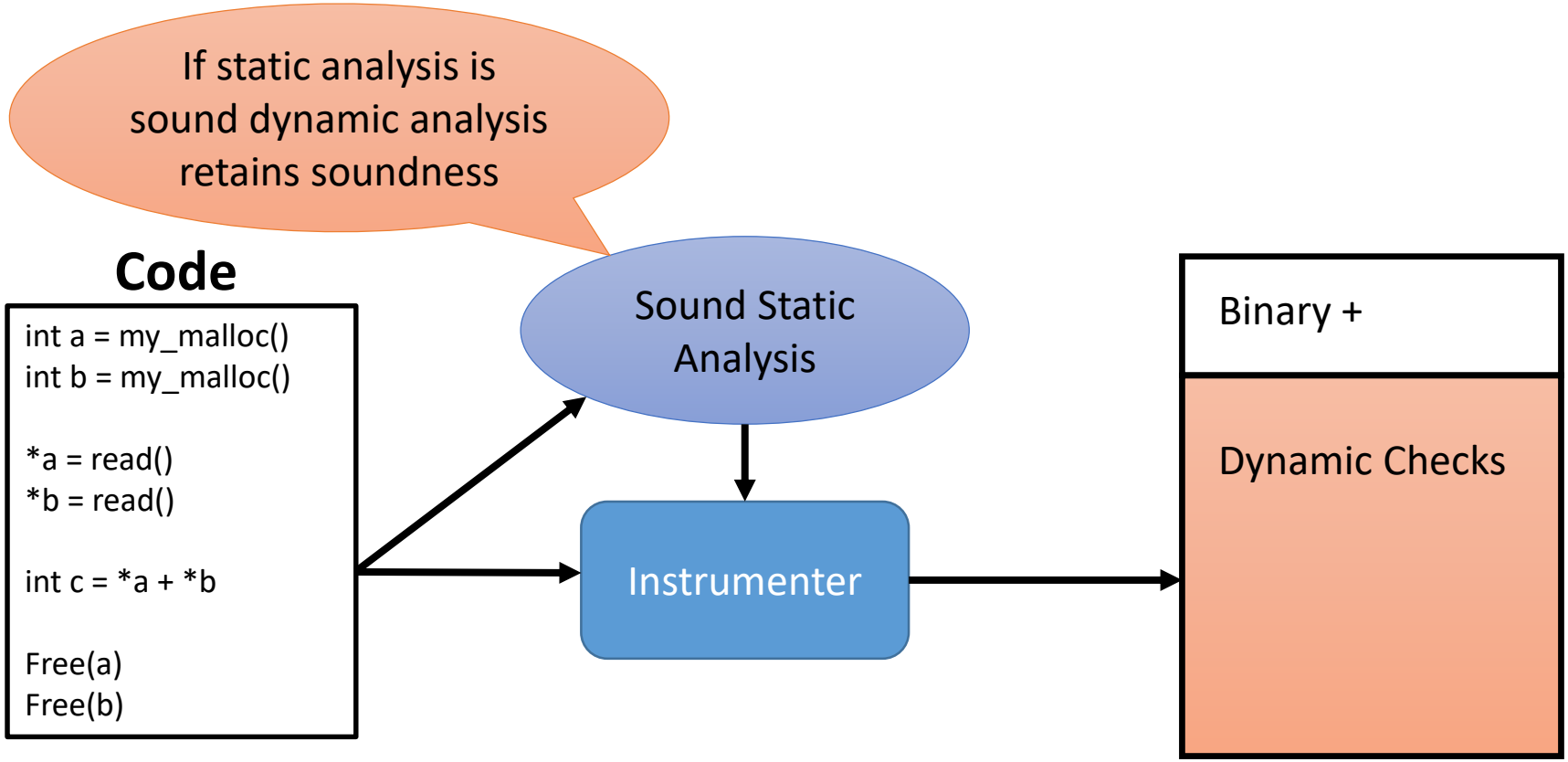
Motivation – Problem

- Problem: Dynamic analysis too expensive
  - Information flow 1-2 orders of magnitude overhead
  - Race detection order of magnitude slowdown
- Question: Can we create a faster dynamic analysis?



Motivation – Hybrid Analysis

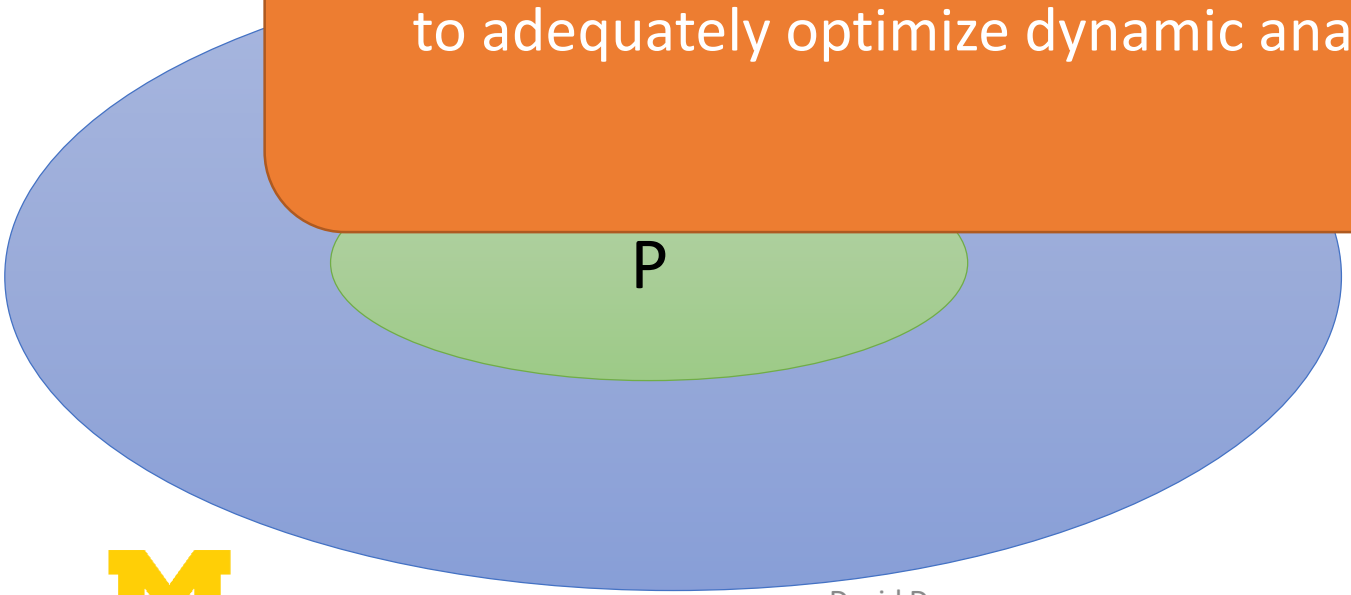
- Use static analysis to optimize dynamic analysis



Sound Static Analysis is Inaccurate

- Makes conservative approximations for soundness
- Sou

m States
tion used



Sound static analysis considers too many states to adequately optimize dynamic analysis!

Optimistic Hybrid Analysis

- Novel dynamic analysis optimization methodology
  - Combines profiling, **unsound static analysis**, and **speculative dynamic** execution
  - Unsound static analysis – better optimize analysis
  - Speculative execution – handles analysis unsoundness
  - No loss of soundness or precision
- Optimistic Hybrid Backward Slicer: **OptSlice**
  - Retains accuracy of traditional dynamic slicer
  - On average: **7.7x faster** than hybrid dynamic slicer

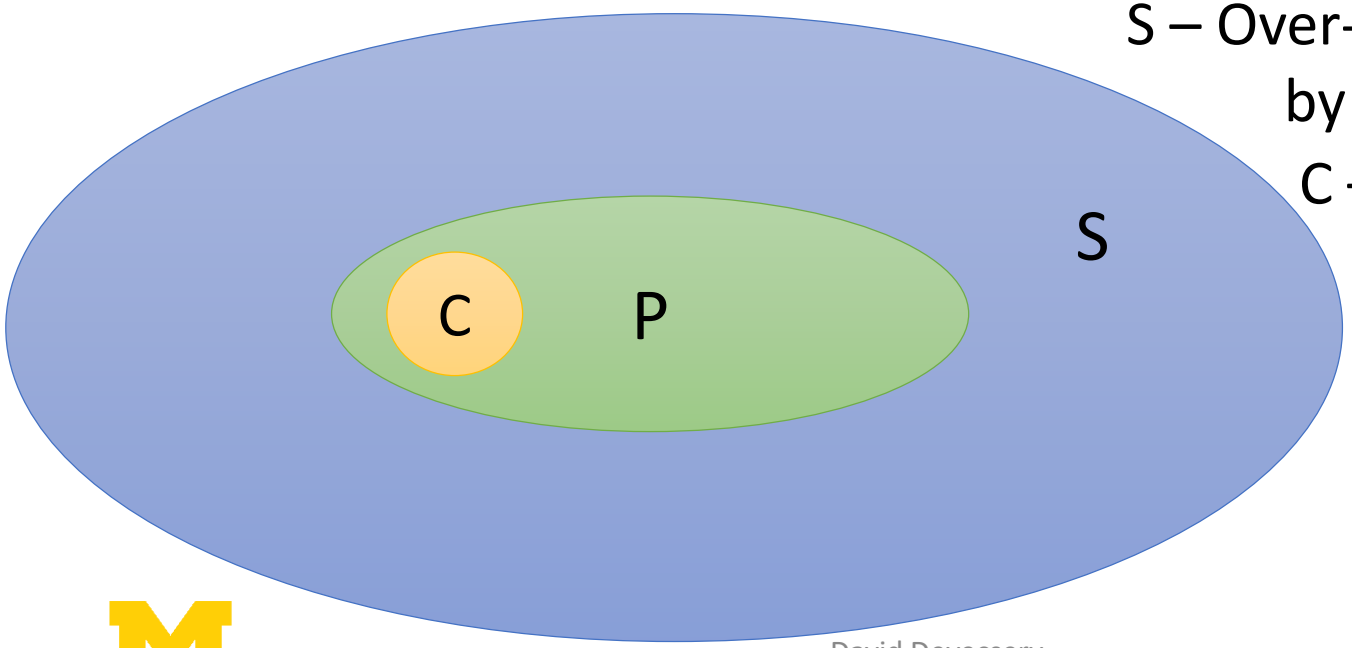
Overview

- ~~Motivation~~
- Optimistic Hybrid Analysis Overview
  - Likely Invariant Gathering
  - Optimistic Static Analysis
  - Speculative Dynamic Analysis
- OptSlice – Optimistic Dynamic Backward Slicer
- Evaluation
- Conclusion

Idea: Optimistic Analysis

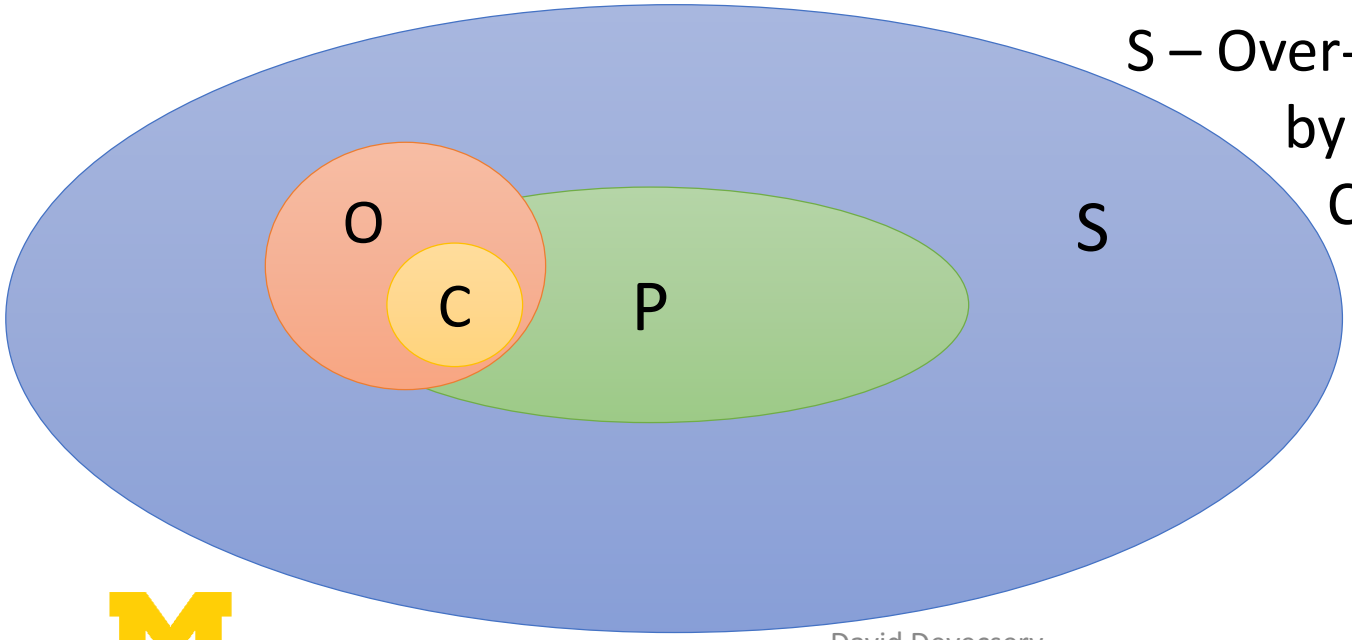
- **Observation**: Hybrid analysis need static analysis to explore **only** states executed by analyzed executions

P – Possible Program States
S – Over-approximation used by analysis
C – Common Executed States



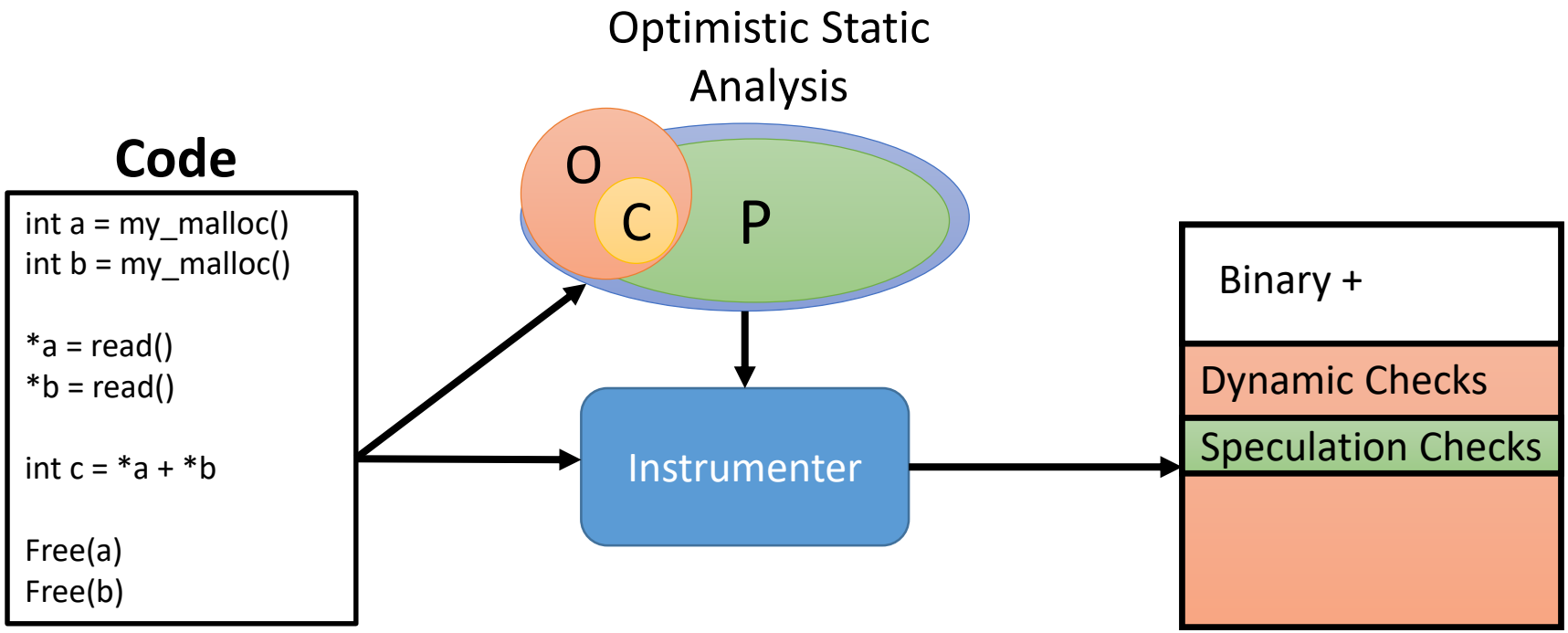
Idea: Optimistic Analysis

- Idea: Instead use unsound static analysis
  - We call: **Optimistic Static Analysis**
- Have dynamic analysis detect & handle unsoundness



P – Possible Program States
S – Over-approximation used by analysis
C – Common states
O - Optimistic Static Analysis

Motivation – Hybrid Analysis

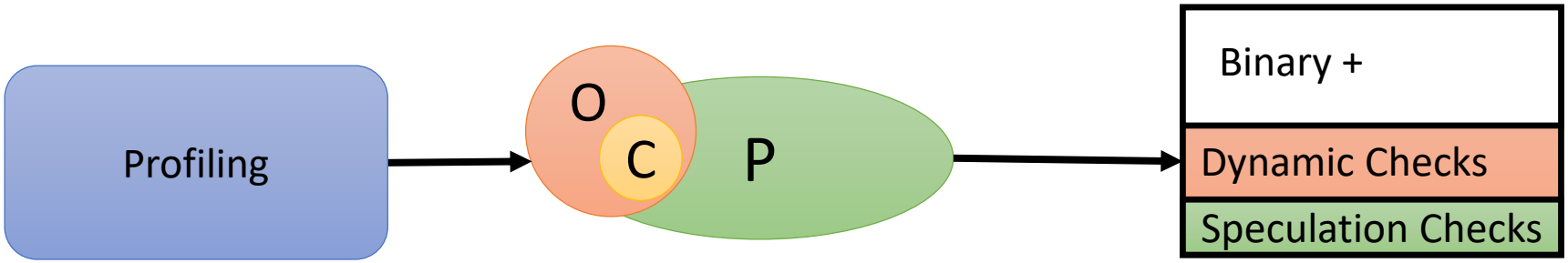


Optimistic Hybrid Analysis Phases

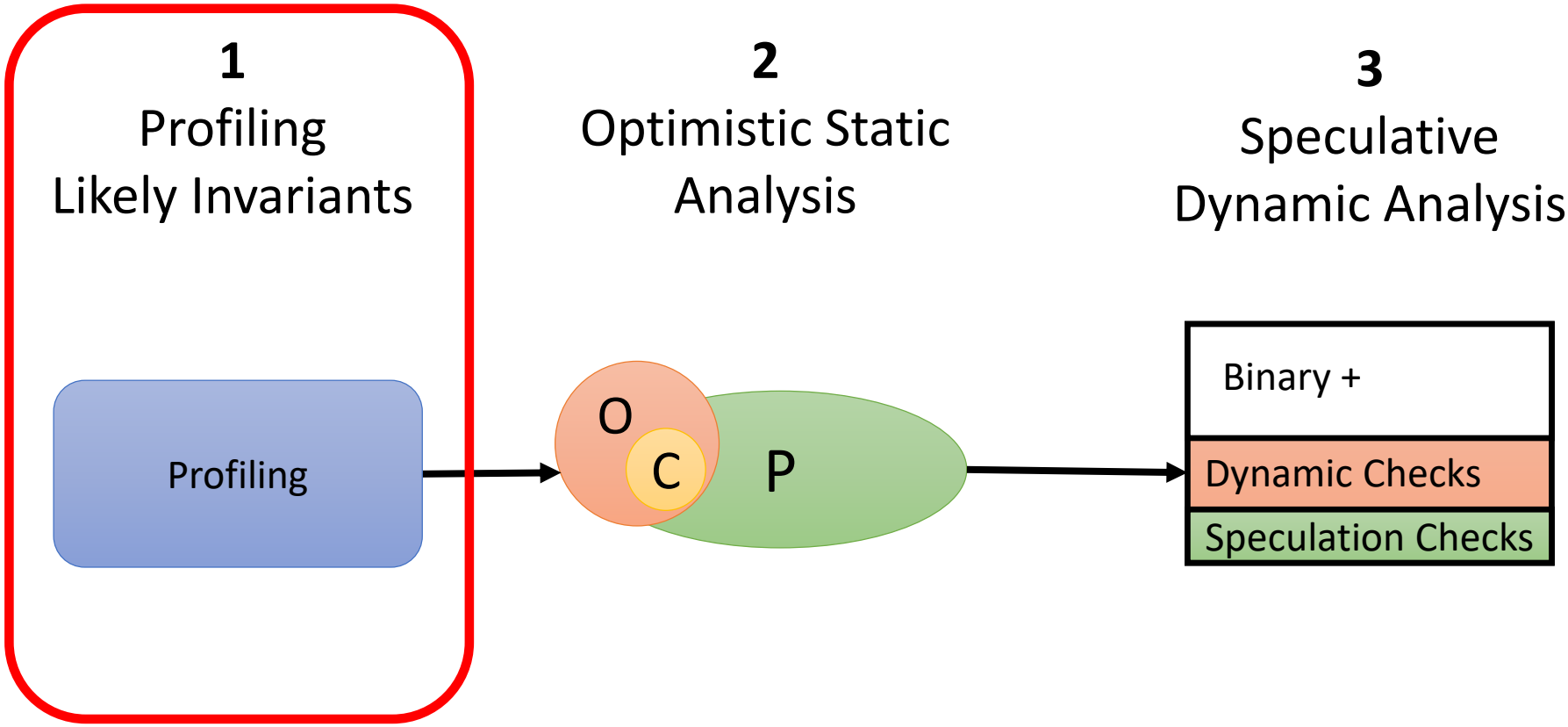
1
Profiling
Likely Invariants

2
Optimistic Static
Analysis

3
Speculative
Dynamic Analysis



Optimistic Hybrid Analysis



Likely Invariants

- **Goal**: Guide optimistic static analysis towards C
- Static analysis assumes likely invariants true
  - Reduce state space searched
  - Add **bounded** unsoundness – if invariant false
- Likely Invariants should be:
  - **Strong** - Help prune states considered by static analysis
  - **Cheap** - Be inexpensive to check dynamically
  - **Stable** - Hold true for most analyzed executions
- Gathered by profiling pass over program

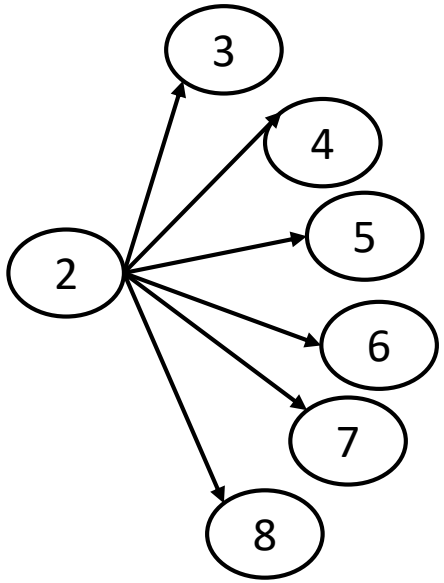
Our Invariants

- Likely Unreachable Code
  - Identifies regions of code which are not dynamically run
  - Stops static analysis from considering code
- Likely Callee Sets
  - Identifies targets of indirect function calls
  - Removes indirect function calls from analysis
- Likely Unreachable Call Contexts
  - Identifies call-stacks reached dynamically
  - Prunes call-graph in context-sensitive analysis

Likely Callee Sets

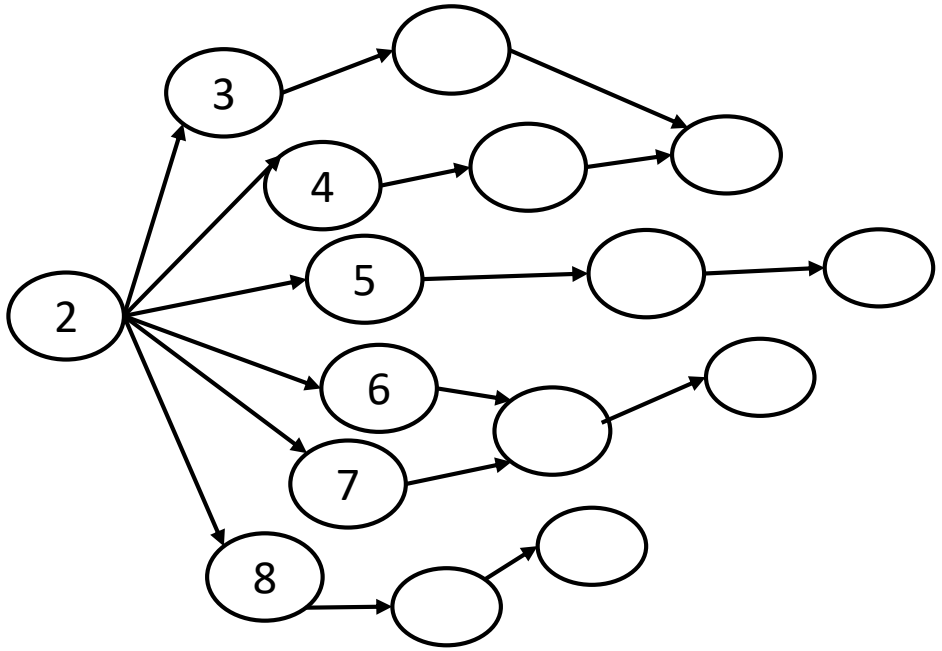
- Assume indirect calls can only call dynamically observed functions

Source Code	Static Analysis Graph

fcn() {
1: hash_fcn_t *fcn =
get_hash_fcn();
2: hash = *fcn(arg1, arg2);
}
 	
3: sha1(arg1, arg2) { … }
4: sha256(arg1, arg2) {…}
5: sha3(arg1, arg2) {…}
6: md5 (arg1, arg2) {…}
7: ripemd(arg1, arg2) {…}
8: whirlpool(arg1, arg2) {…}
	

Likely Callee Sets

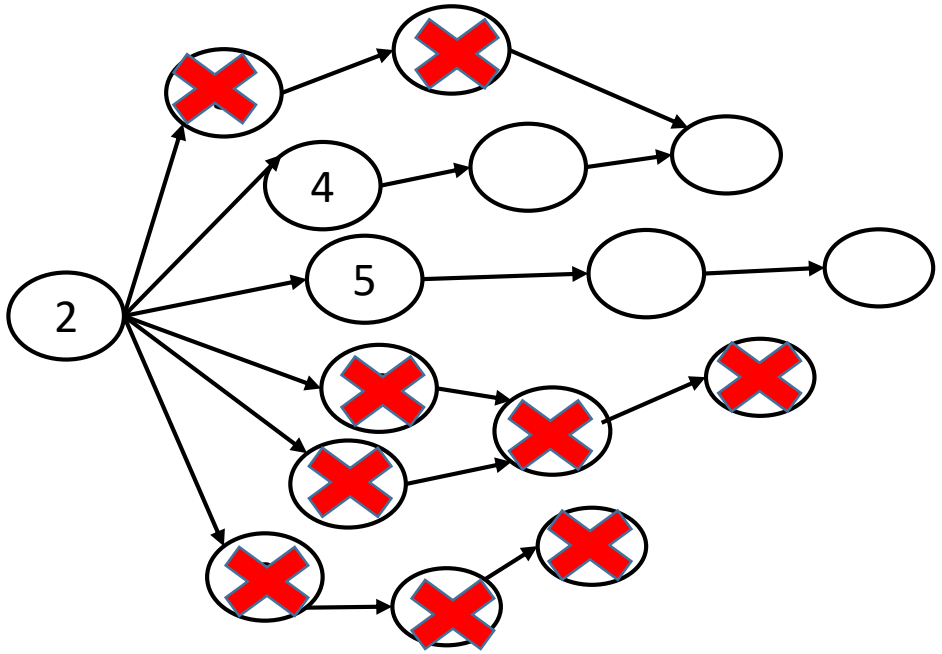
Source Code	Static Analysis Graph

fcn() {
1: hash_fcn_t *fcn =
get_hash_fcn();
2: hash = *fcn(arg1, arg2);
}
 	
3: sha1(arg1, arg2) { … }
4: sha256(arg1, arg2) {…}
5: sha3(arg1, arg2) {…}
6: md5 (arg1, arg2) {…}
7: ripemd(arg1, arg2) {…}
8: whirlpool(arg1, arg2) {…}
	

Likely Callee Sets

- Indirect Functions typically only call a small set of functions

Source Code	Static Analysis Graph

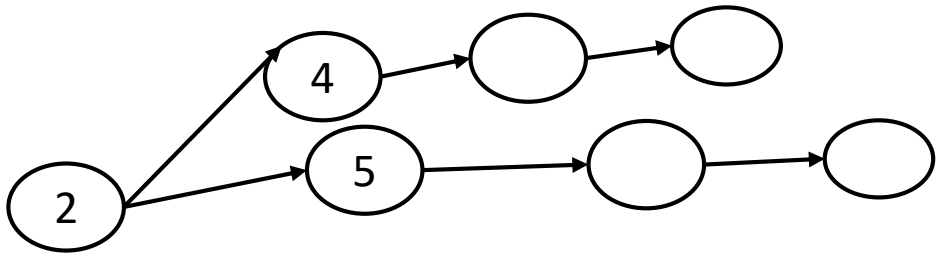
fcn() {
1: hash_fcn_t *fcn =
get_hash_fcn();
2: hash = *fcn(arg1, arg2);
6: md5 (arg1, arg2) {…}
7: ripemd(arg1, arg2) {…}
8: whirlpool(arg1, arg2) {…}
	

Likely Invariant:
*fcn is sha2 or sha3

Likely Callee Sets

- Indirect Functions typically only call a small set of functions

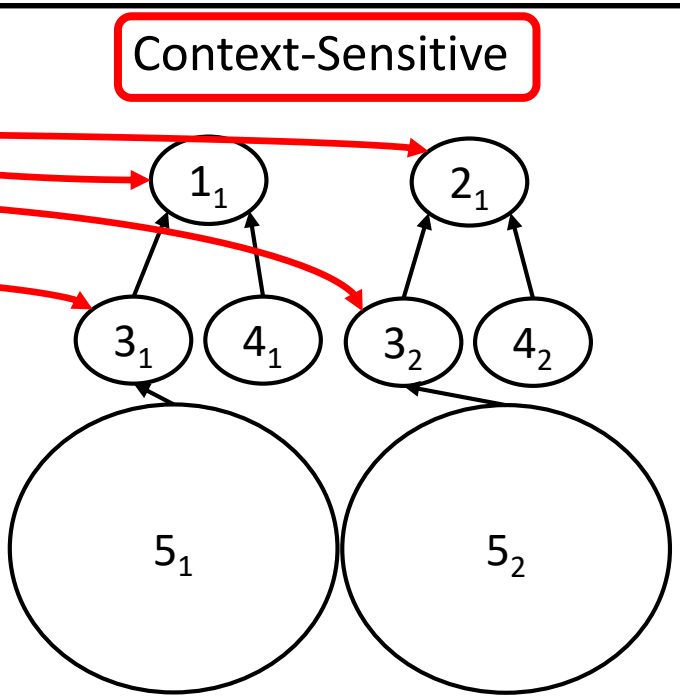
Source Code	Static Analysis Graph

fcn() {
1: hash_fcn_t *fcn =
get_hash_fcn();
2: hash = *fcn(arg1, arg2);
6: md5 (arg1, arg2) {…}
7: ripemd(arg1, arg2) {…}
8: whirlpool(arg1, arg2) {…}
	

Likely Invariant:
*fcn is sha2 or sha3

Likely Unused Call Contexts

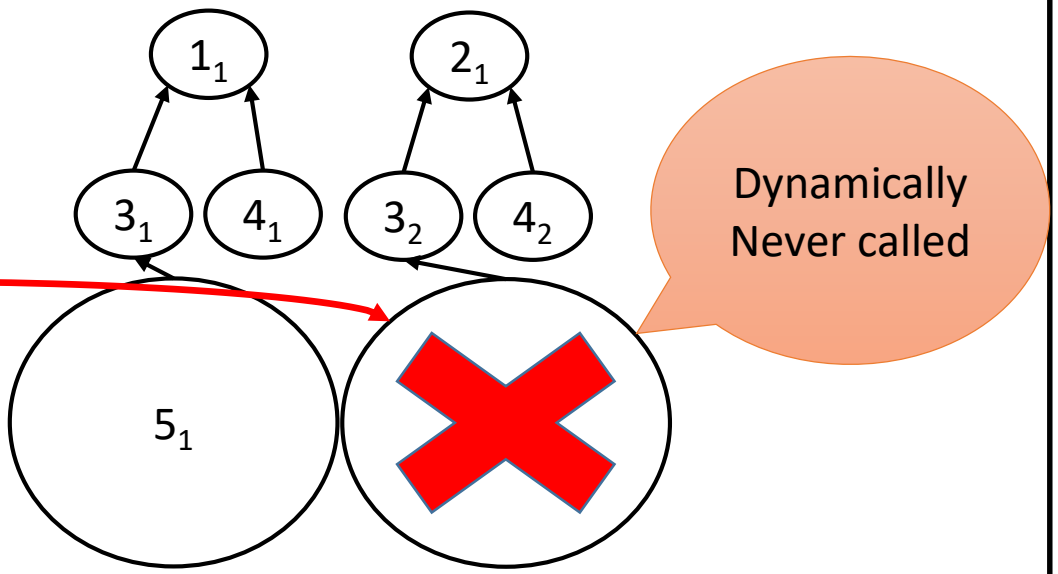
Source Code	Analysis Graph

main() {
1: a = my_malloc();
2: b = my_malloc();}
 	
my_malloc() {
if (!g_init)
3: return do_init();
4: return malloc(…);}
 	
do_init() {
g_init = true;
5: // Long complex
initialization code}
	

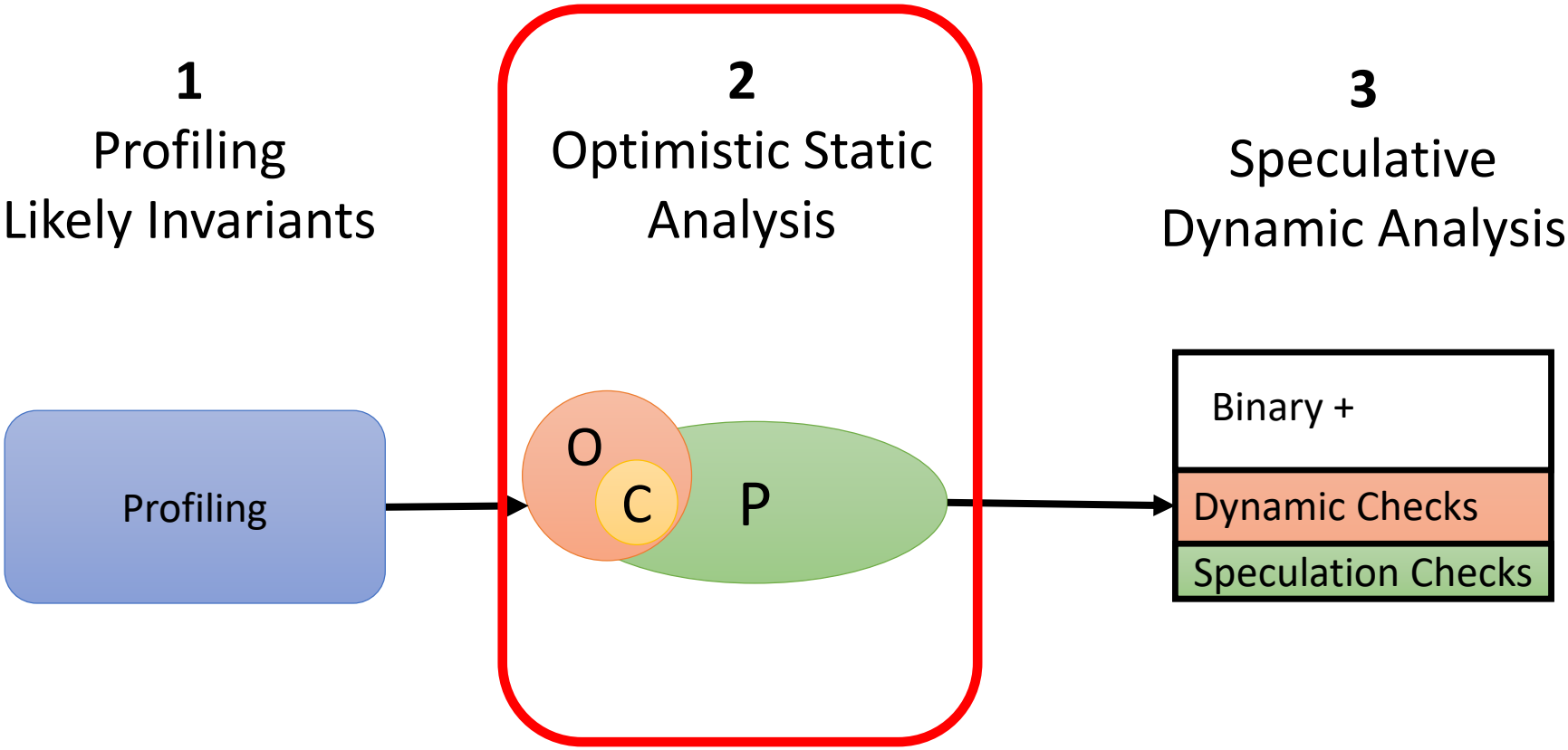
Likely Unused Call Contexts

- Remove call-stacks which never dynamically happen

Source Code	Analysis Graph

main() {
1: a = my_malloc();
2: b = my_malloc();}
 	
my_malloc() {
if (!g_init)
3: return do_init();
4: return malloc(…);}
 	
do_init() {
g_init = true;
5: // Long complex
initialization code}
	Context-Sensitive
	 

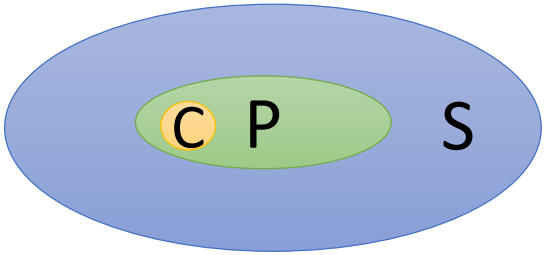
Optimistic Hybrid Analysis



Optimistic Static Analysis

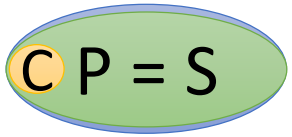
- Assumes likely invariants from profiling are true
- Likely Invariants allow analysis to approximate C
- If likely invariants hold, analysis is sound for execution

Sound Static Analysis



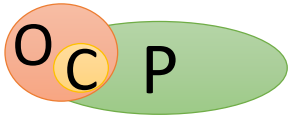
S > P

Perfect Static Analysis



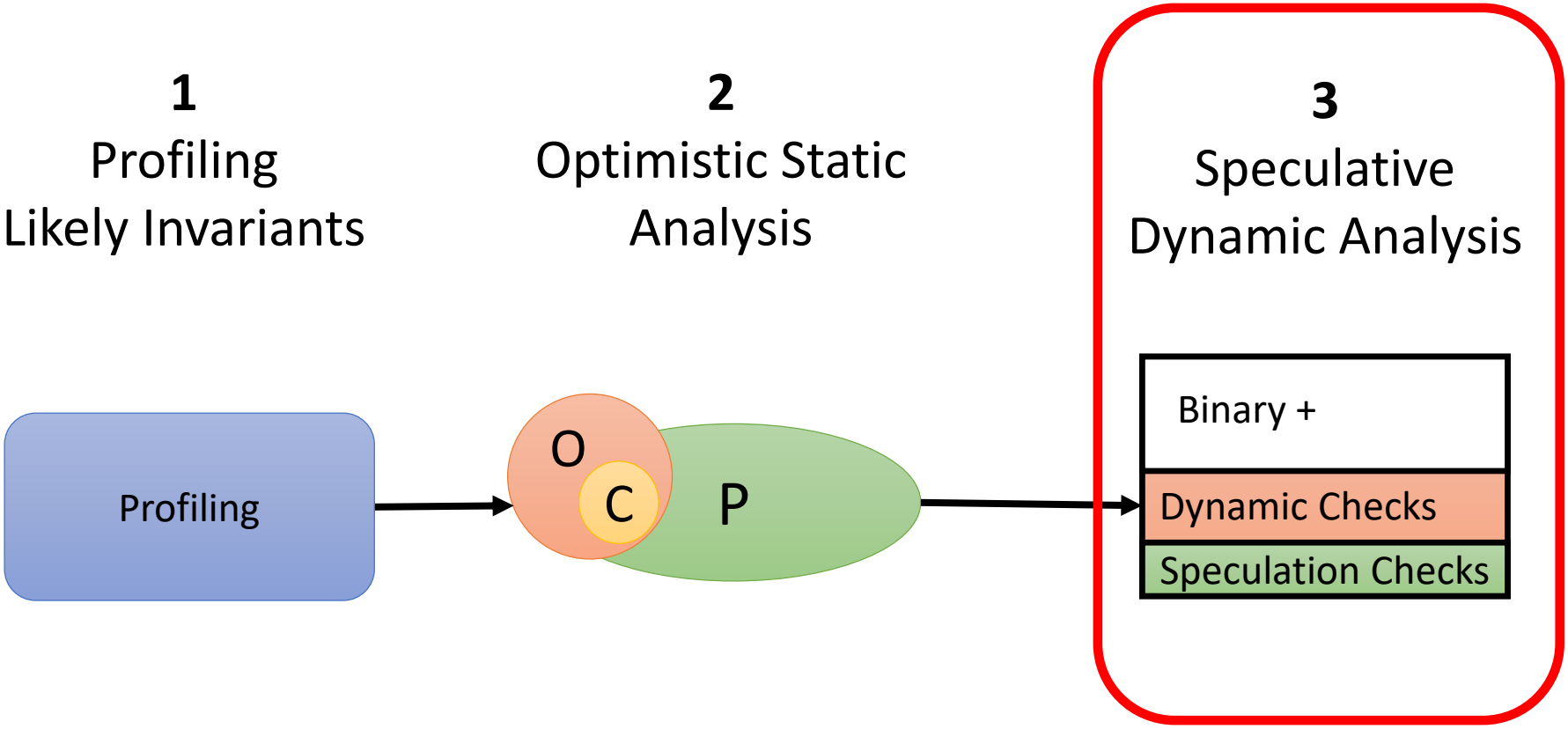
S = P

Optimistic Static Analysis



O < P

Optimistic Hybrid Analysis



Speculative Dynamic Analysis

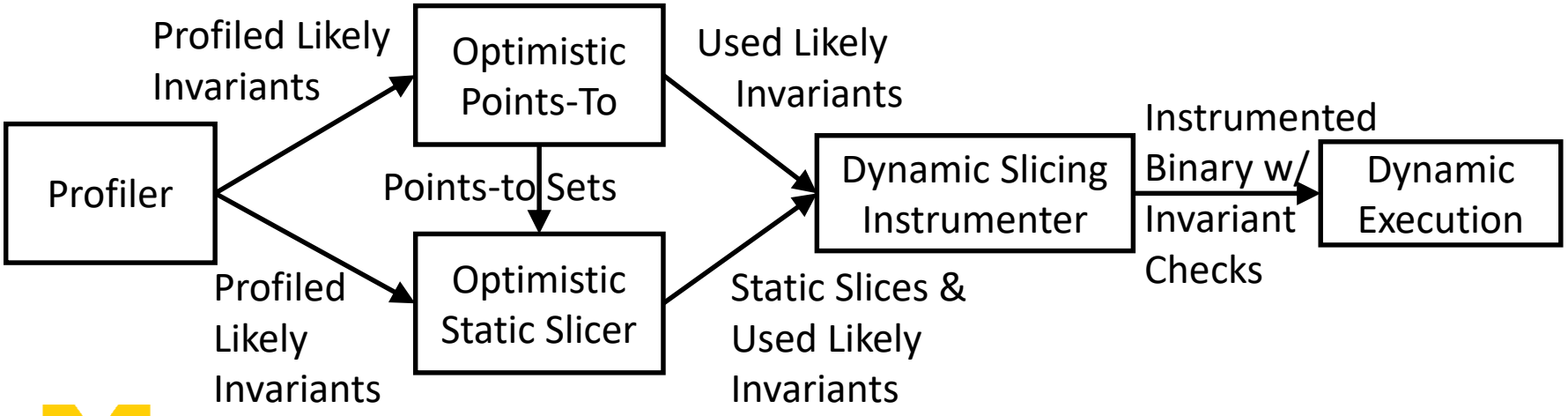
- Runs optimized analysis
  - Fraction of original checks
  - Added likely-invariant speculation checks
    - Lightweight by design (by likely-invariant choices)
- May have to roll-back on mis-speculation
  - Mis-speculations are rare
  - Worst case: full-execution roll-back re-execute
  - On re-execute – use conservative dynamic analysis
    - Eg. A traditional hybrid analysis

Overview

- ~~Motivation~~
- ~~Optimistic Hybrid Analysis Overview~~
  - ~~Likely Invariants~~
  - ~~Optimistic Static Analysis~~
  - ~~Speculative Dynamic Analysis~~
- OptSlice – Optimistic Dynamic Backward Slicer
- Evaluation
- Conclusion

OptSlice

- Optimistic Dynamic Backwards Slicer
- Uses three likely invariants discussed
- Uses optimistic static points-to and slicing analyses
- Heavyweight (full-execution) roll-back recovery



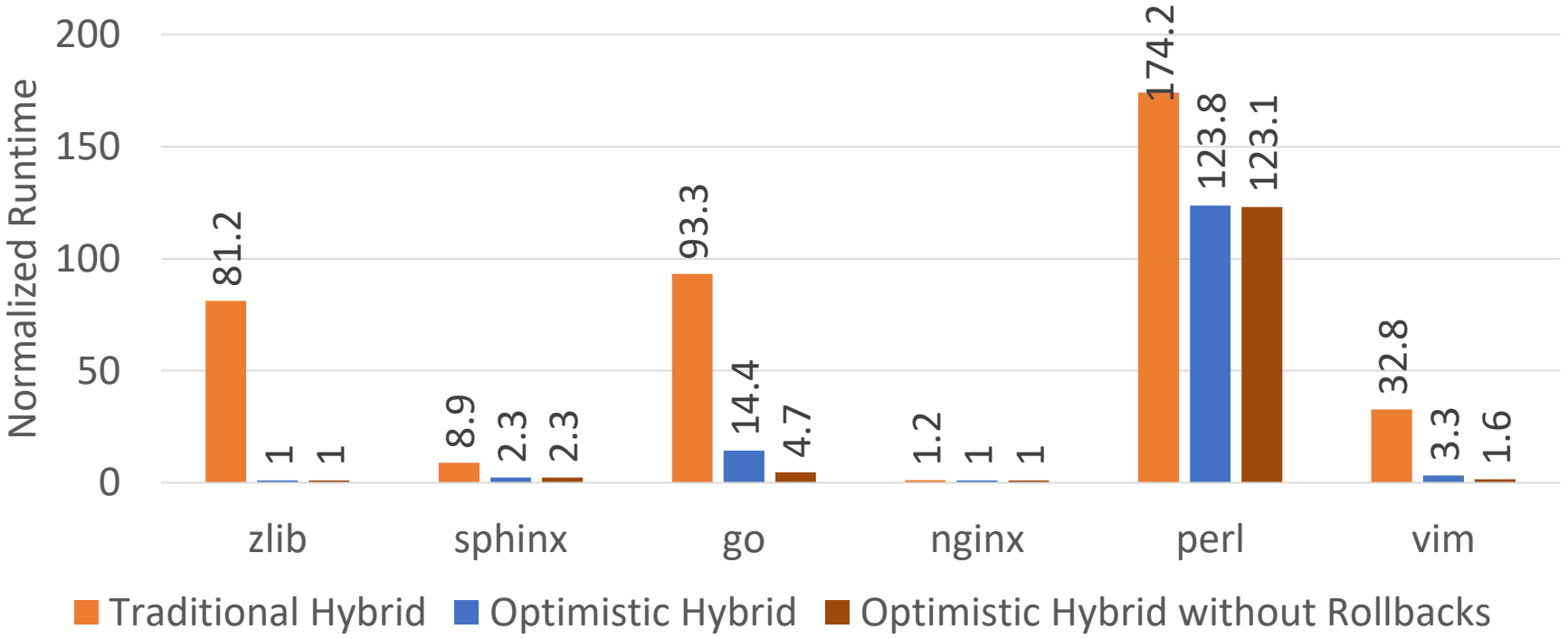
OptSlice – Misc. Details

- Static analyses have:
  - Call context-sensitive and insensitive versions
  - Use the most accurate version that will scale
  - Optimistic Static Analysis processes less states
    - Generally more scalable
- Built on-top of Giri dynamic slicer (ASPLOS 13)

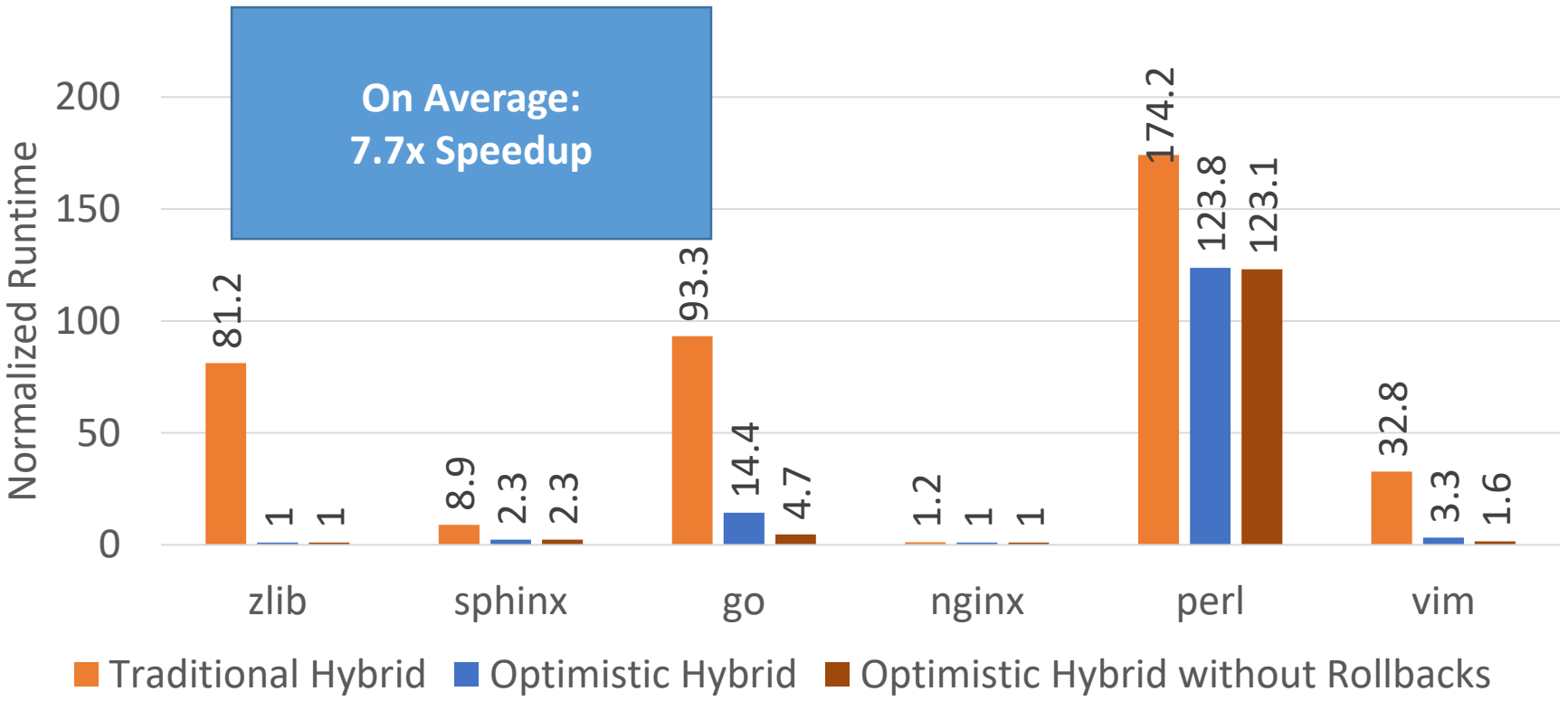
Evaluation

- All experiments bounded to 8hrs, 16GB RAM
  - We use the best static analysis that will complete
- Analysis built in LLVM-3.1
- **Goal**: Show how OHA techniques enable OptSlice to accelerate dynamic analysis
- Evaluate three portions
  - OptSlice speedup
  - Invariant profiling requirements
  - Static analysis improvements

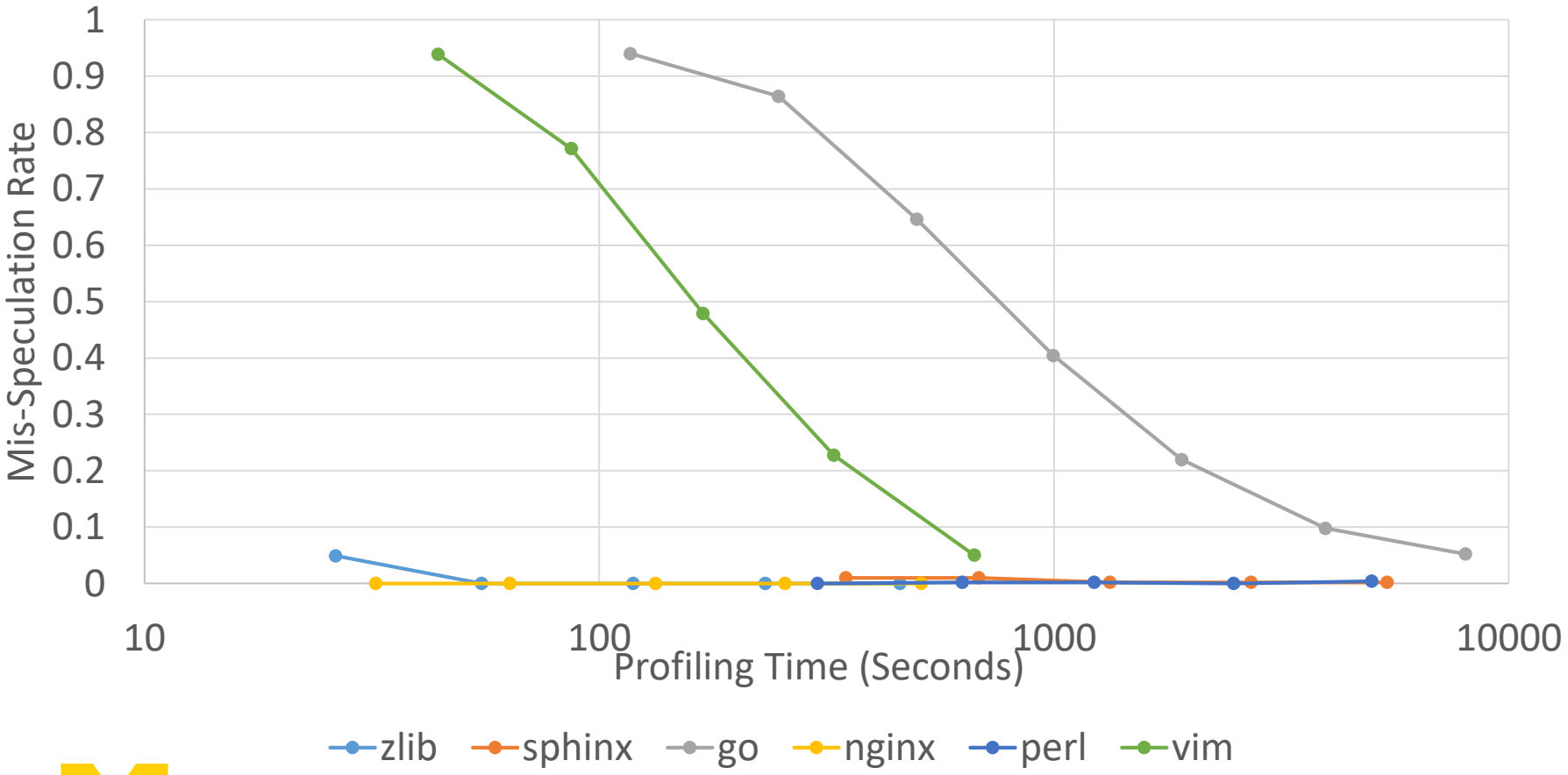
Runtime


Normalized Runtime
200
150
100
50
0
81.2
1
1
8.9
2.3
2.3
93.3
14.4
4.7
1.2
1
1
174.2
123.8
123.1
32.8
3.3
1.6
zlib
sphinx
go
nginx
perl
vim
Traditional Hybrid
Optimistic Hybrid
Optimistic Hybrid without Rollbacks

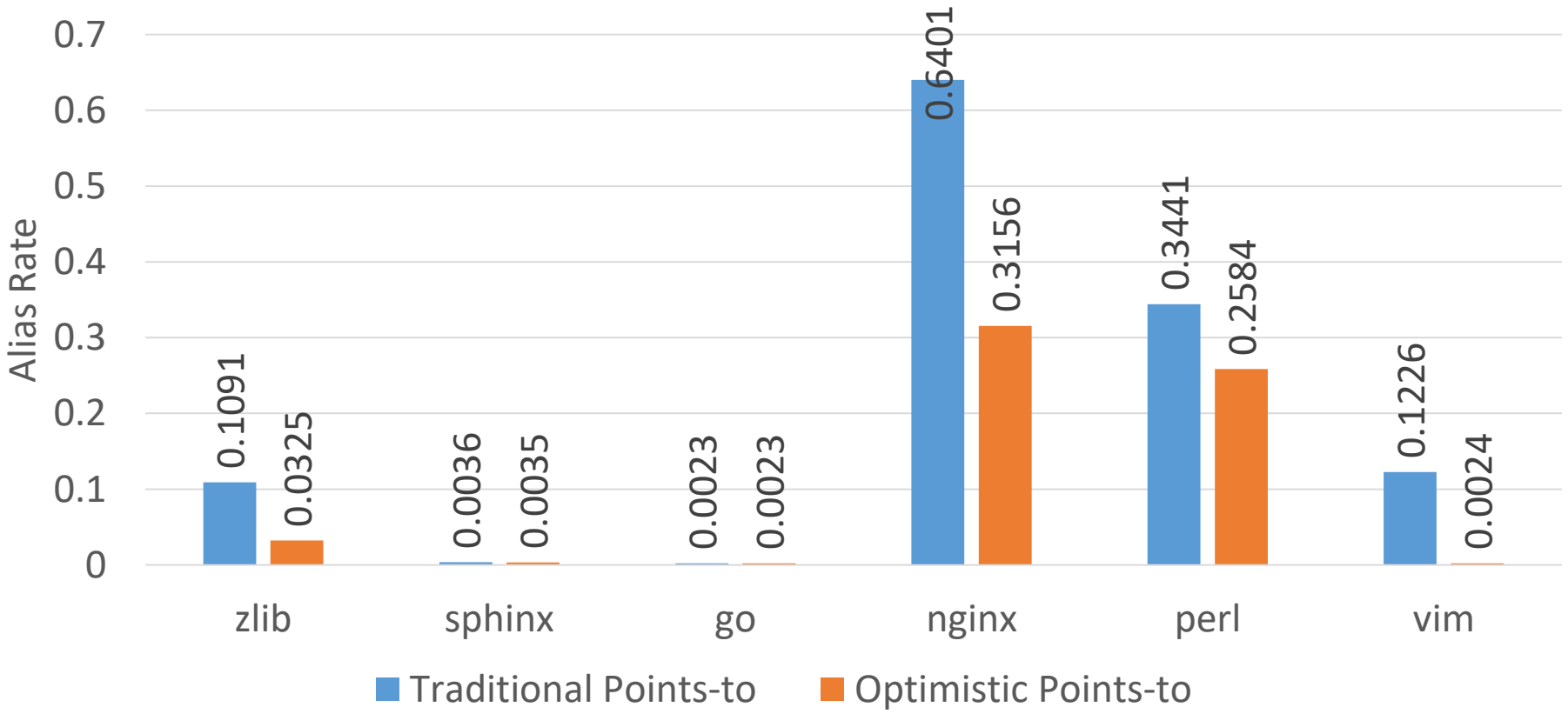
Runtime


On Average:
7.7x Speedup
Normalized Runtime
200
150
100
50
0
81.2
1
1
8.9
2.3
2.3
93.3
14.4
4.7
1.2
1
1
174.2
123.8
123.1
32.8
3.3
1.6
zlib
sphinx
go
nginx
perl
vim
Traditional Hybrid
Optimistic Hybrid
Optimistic Hybrid without Rollbacks

Profiling


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Mis-Speculation Rate
10
100
1000
10000
Profiling Time (Seconds)
zlib
sphinx
go
nginx
perl
vim

Static Points-to Analysis



Conclusion

- Optimistic Hybrid Analysis
  - Novel analysis methodology: combines **unsound** static analysis, and **speculative** dynamic analysis
  - Accelerates dynamic analysis without sacrificing accuracy or soundness
- OptSlice
  - Optimistic Hybrid Backward Slicer
  - 7.7x average speedup versus traditional hybrid slicing

Questions?