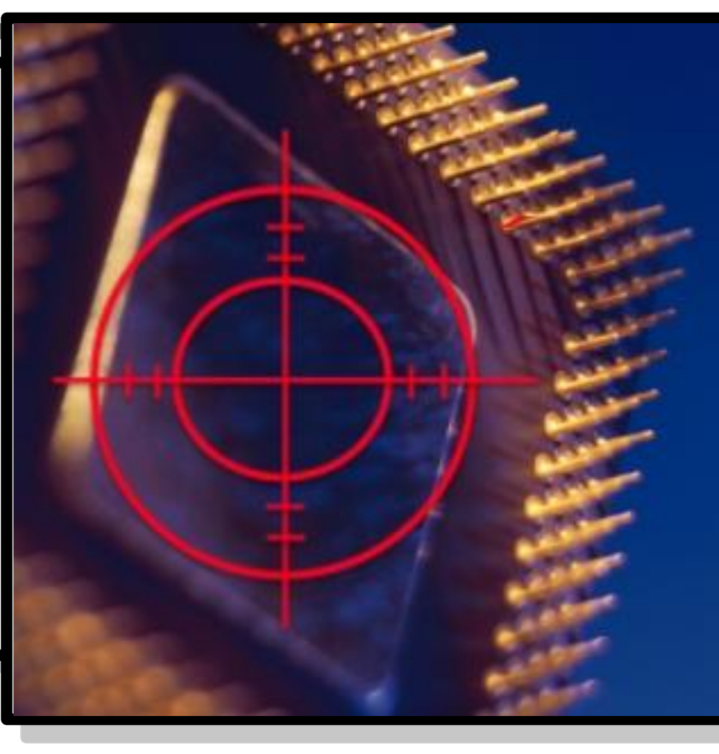
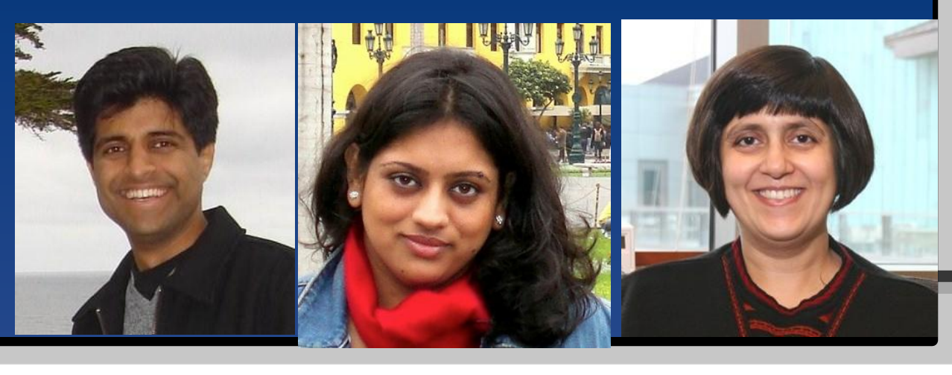
GSRC FCRP



Look Ma, No SDCs!

Siva Hari†, Radha Venkatagiri†, Sarita Adve†, and Helia Naeimi‡
†University of Illinois at Urbana-Champaign, ‡Intel Corporation

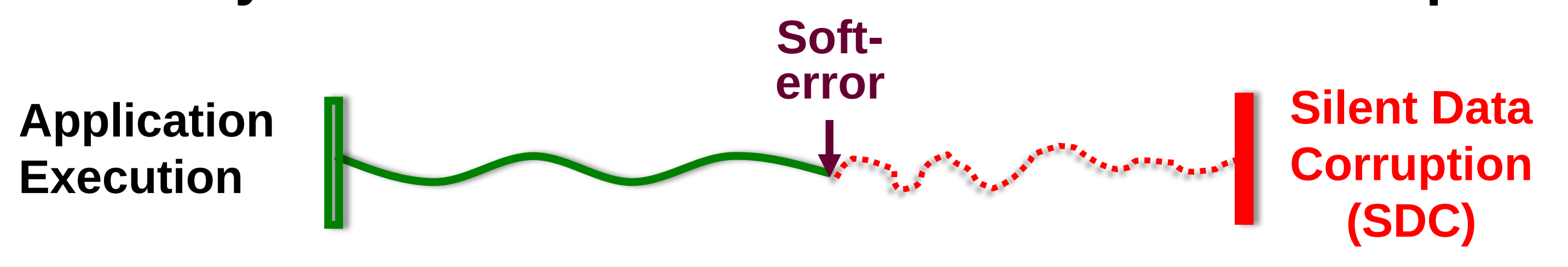
Resilient System Task # 5.5.3



GSRC Annual Symposium
Oct 17, 2012
PUB ID 2926

Motivation

- Resiliency solution should handle Silent Data Corruptions

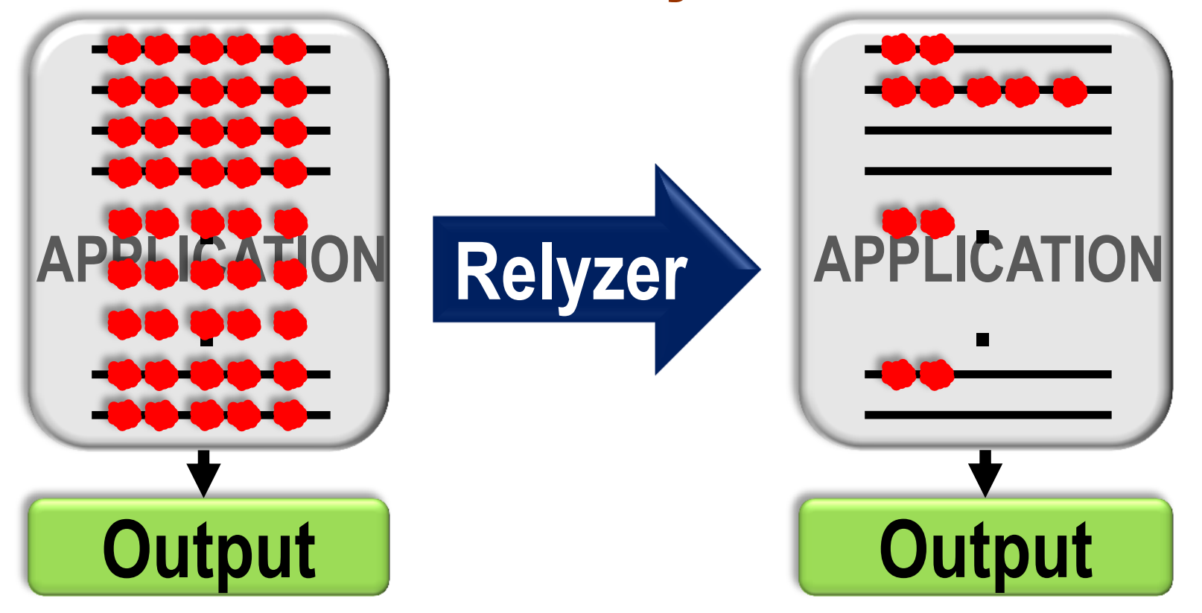


- Can we find *all* SDC-prone application locations?
- How to cost-effectively convert SDCs to detections?
- How to tune resiliency vs. performance?

Finding SDC-Vulnerable App Sites [ASPLOS 2012]

Traditional Statistical Fault Injections	Ideal Approach
Injections in few app-sites	Injections in all app sites
Cannot find all SDC sites	Find *all* SDC sites

Analyze *all* app fault-sites with *few* injections



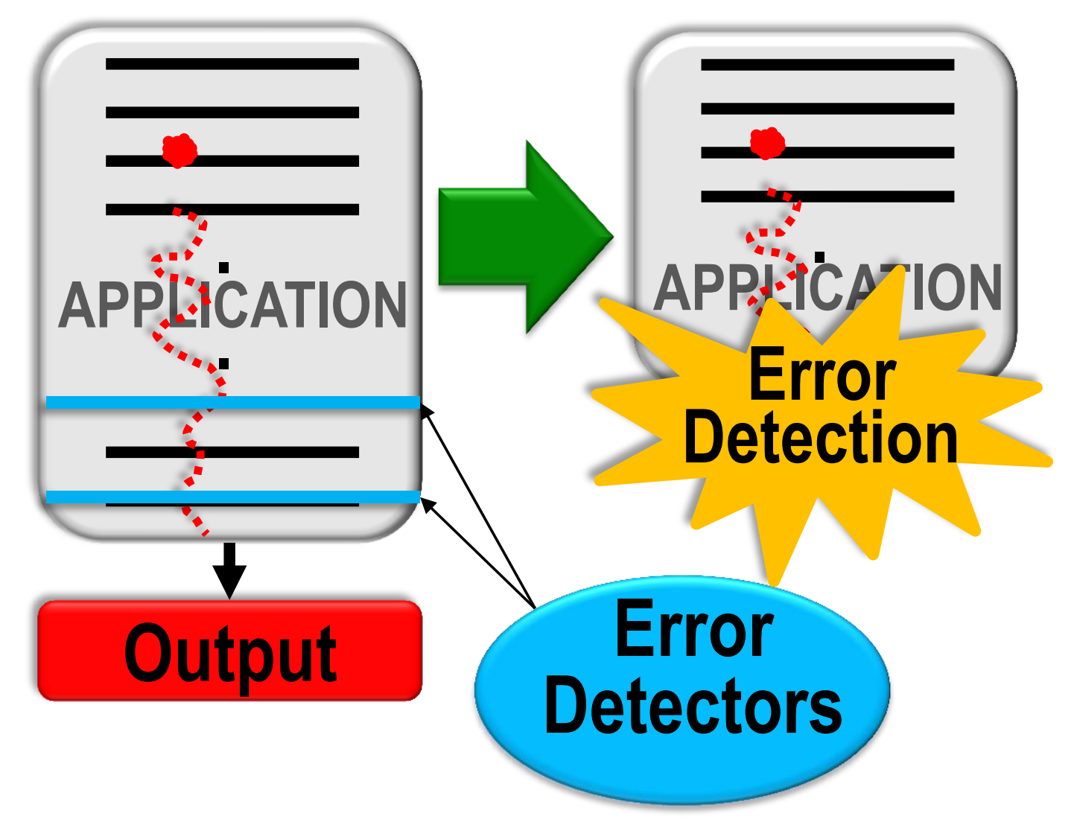
Relyzer can identify *all* SDC producing application locations

Converting SDCs to Detections [DSN 2012]

Traditional approach: Instruction duplication

Our approach:

Low cost error detectors
+
Selective duplication
⬇
Tunable resiliency at low cost



Relyzer: Exploiting App-level Fault Equivalence

Fault model: Transient bit-flips in register operands of *every executing instruction*

Relyzer prunes app fault-sites that need detailed injections

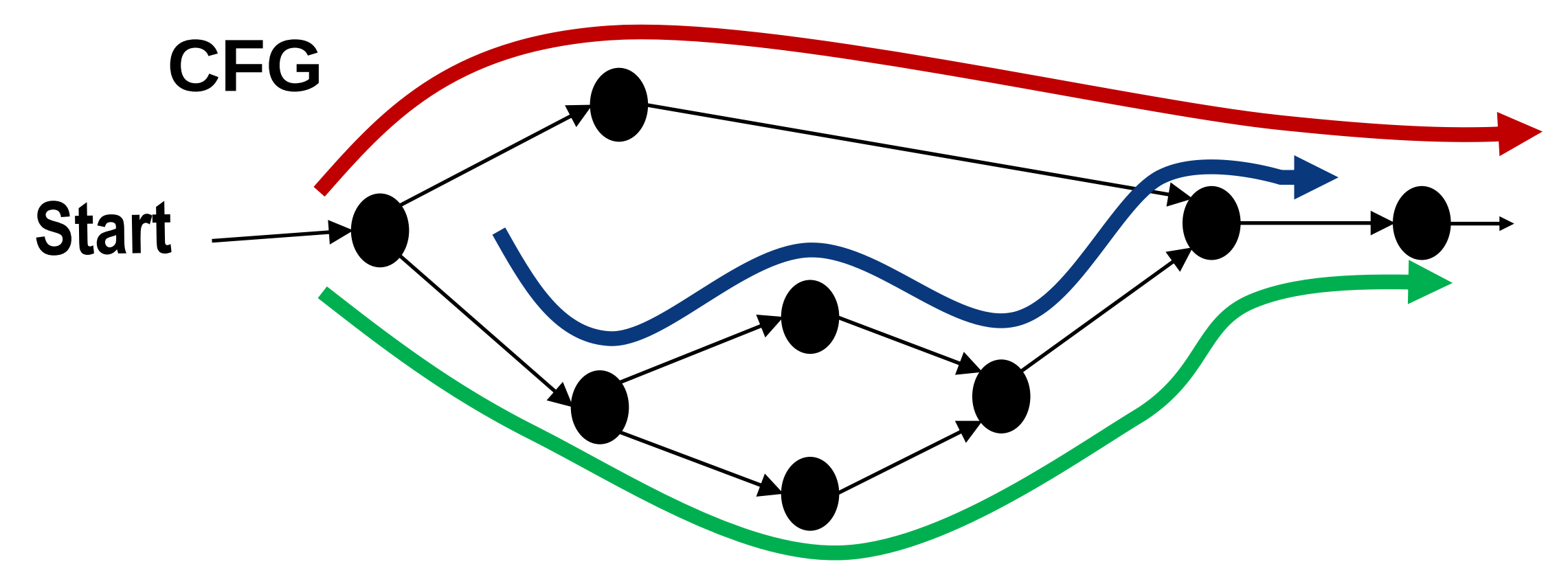
Application level fault equivalence

E.g., Precise Def-Use Analysis

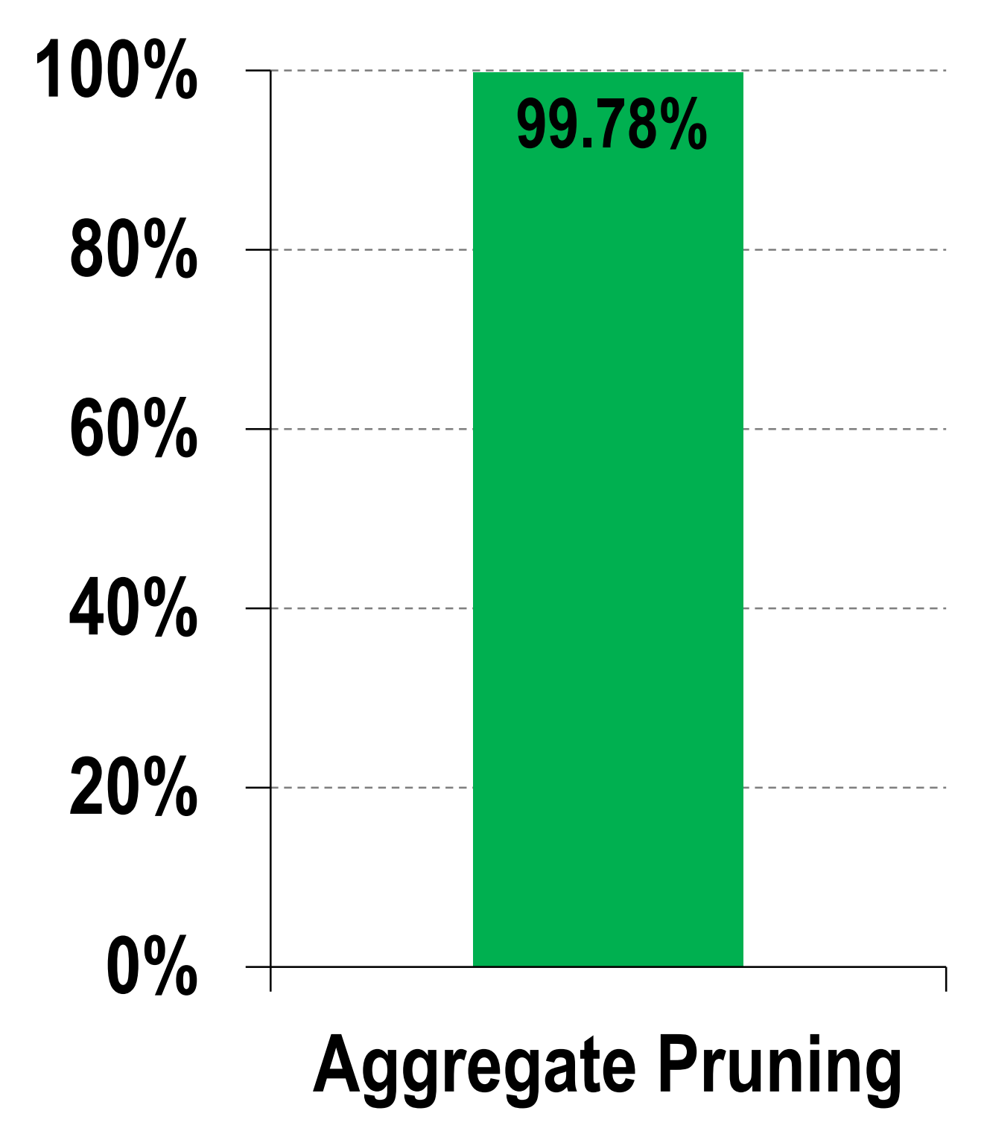
- Faults in definition are equivalent to faults in first use

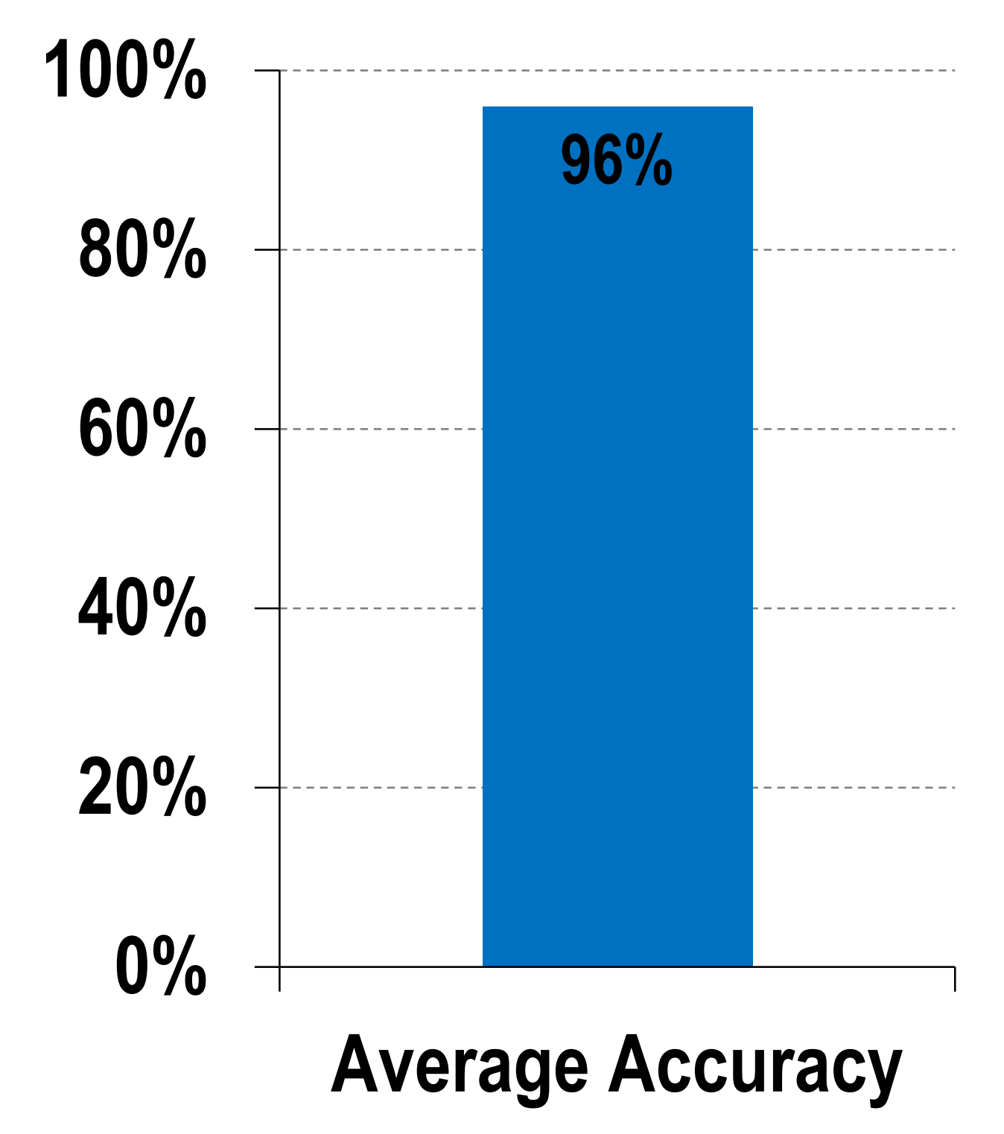
E.g., Heuristics-based Control Analysis

- Idea: Faults flowing through same instrns. behave similarly



Results: 2 to 6 orders of magnitude pruning at 96% accuracy

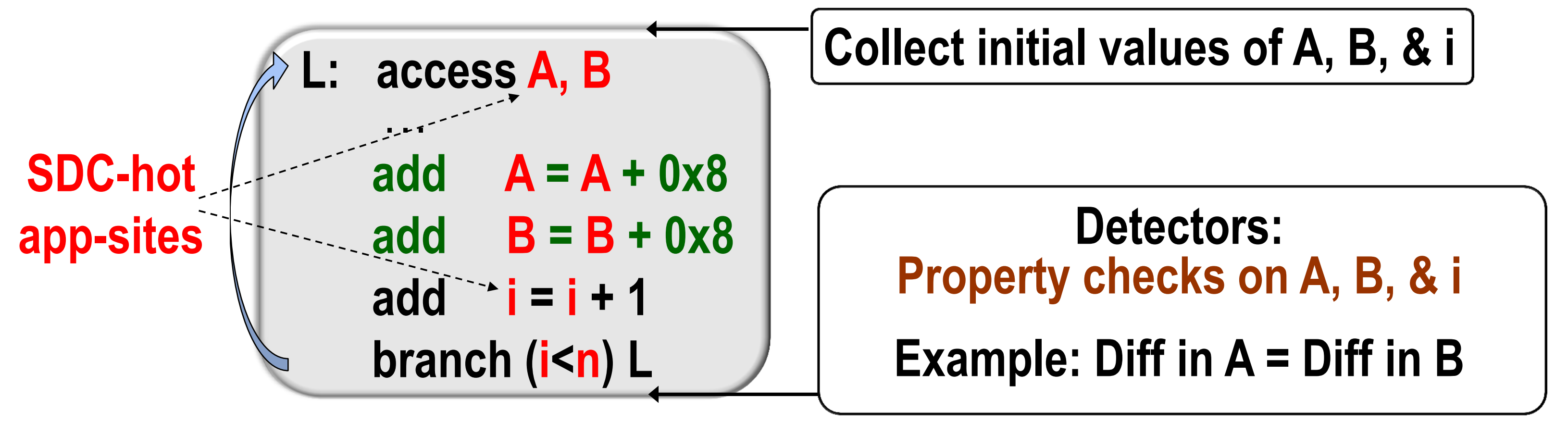




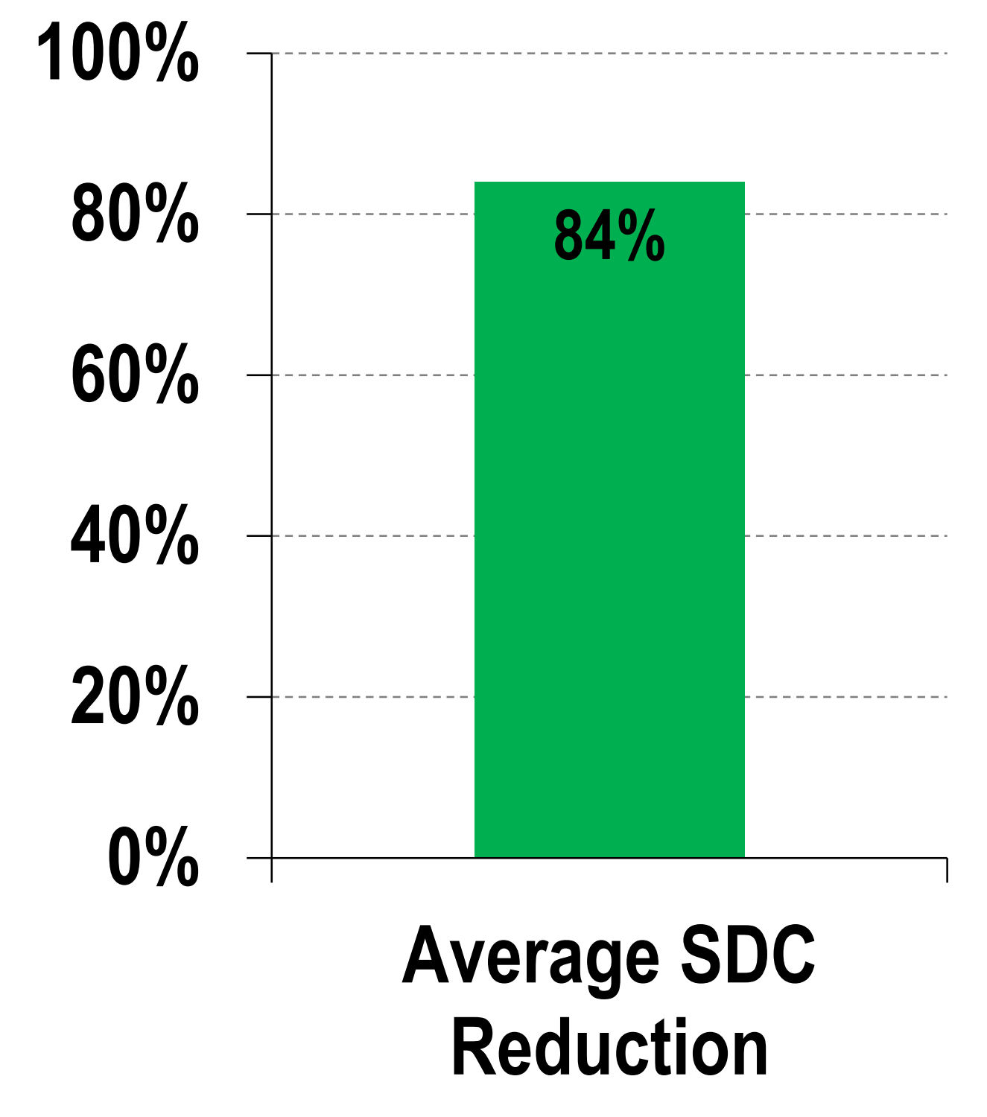
Low cost Program-level Detectors

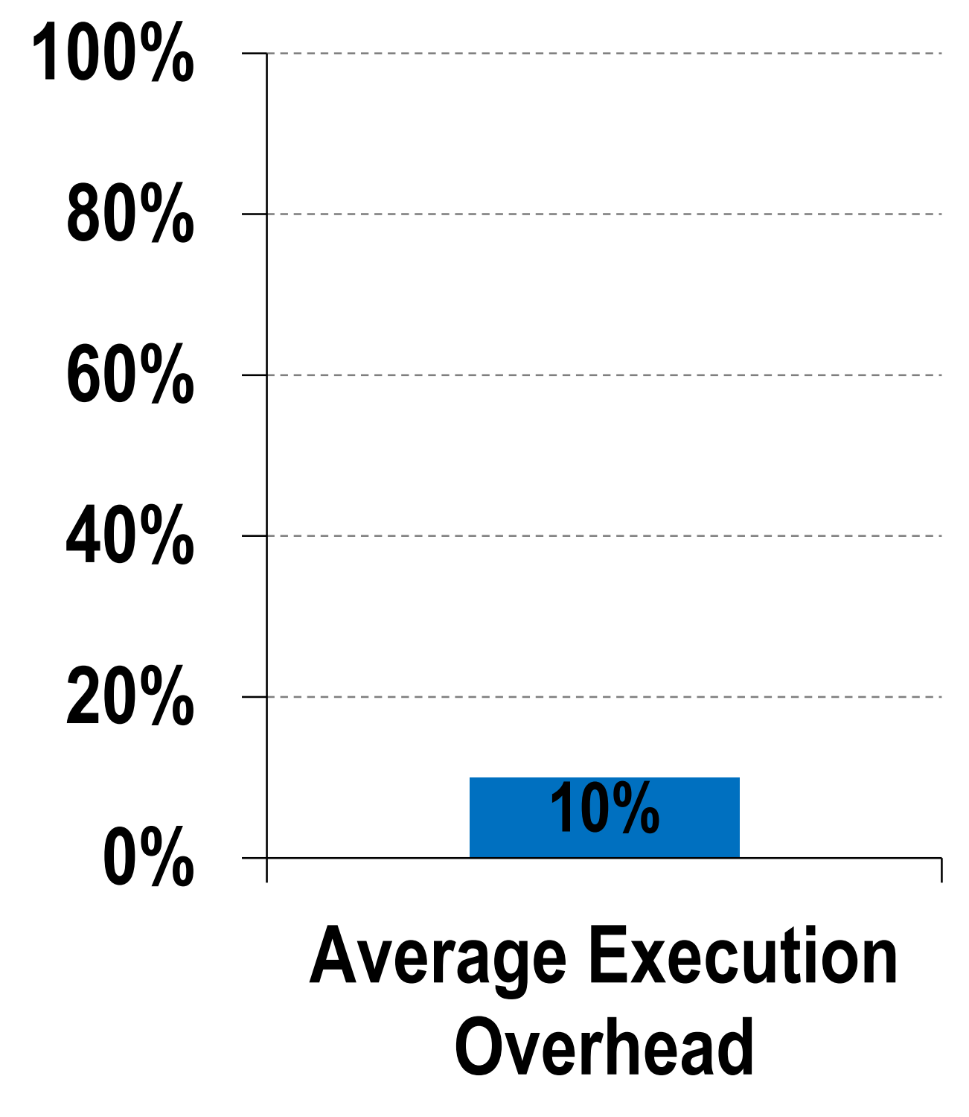
Challenges:	Our approach:
Where to place?	Many errors propagate to few program values • End of loops and function calls
What to use?	Program level properties tests • E.g., value equality, bounds
Uncovered fault-sites?	Selective duplication on few locations

Example: Loop incrementalization based detector



Results: 84% SDCs detected at 10% cost





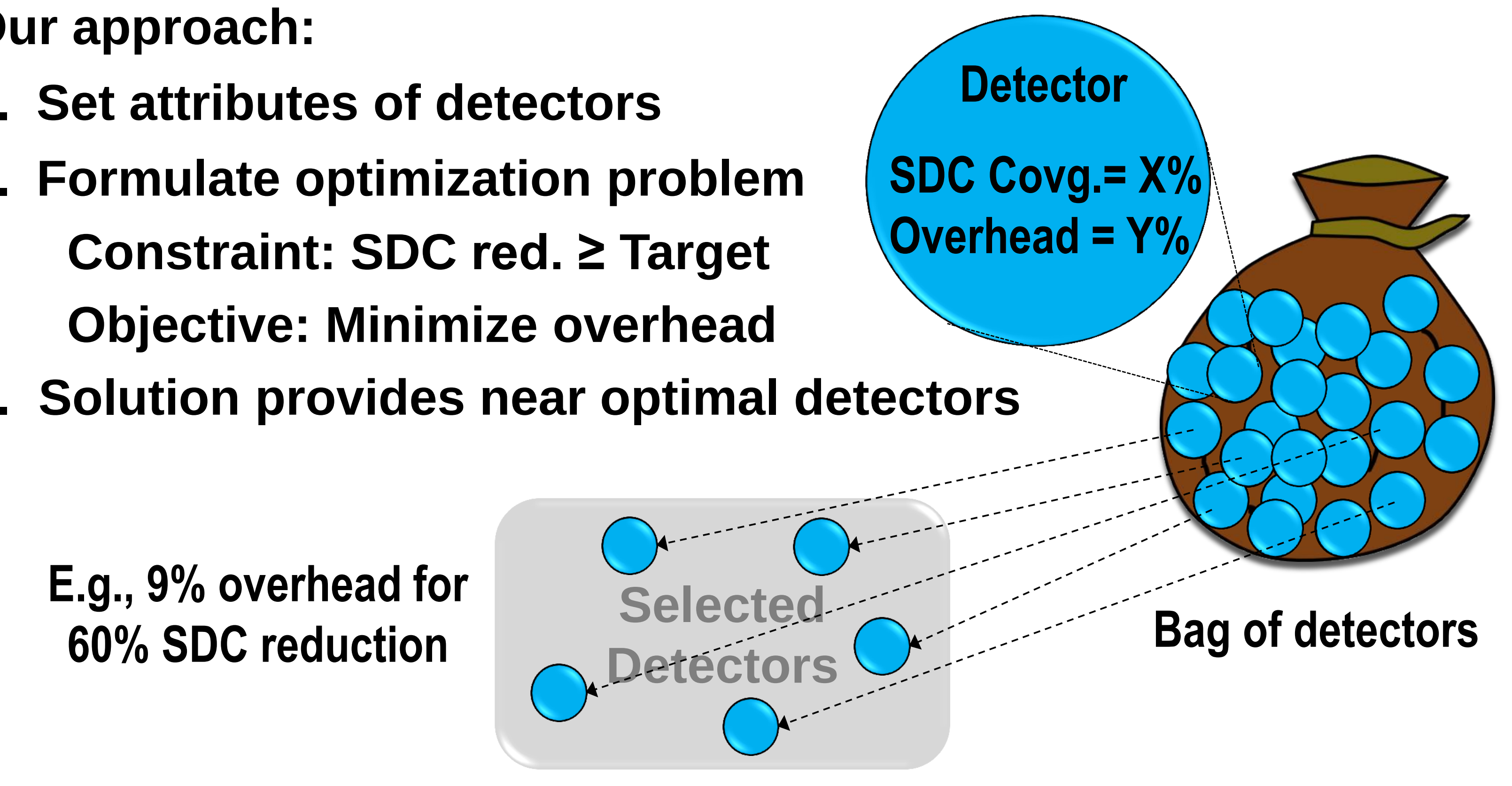
Tuning Resiliency vs. Performance

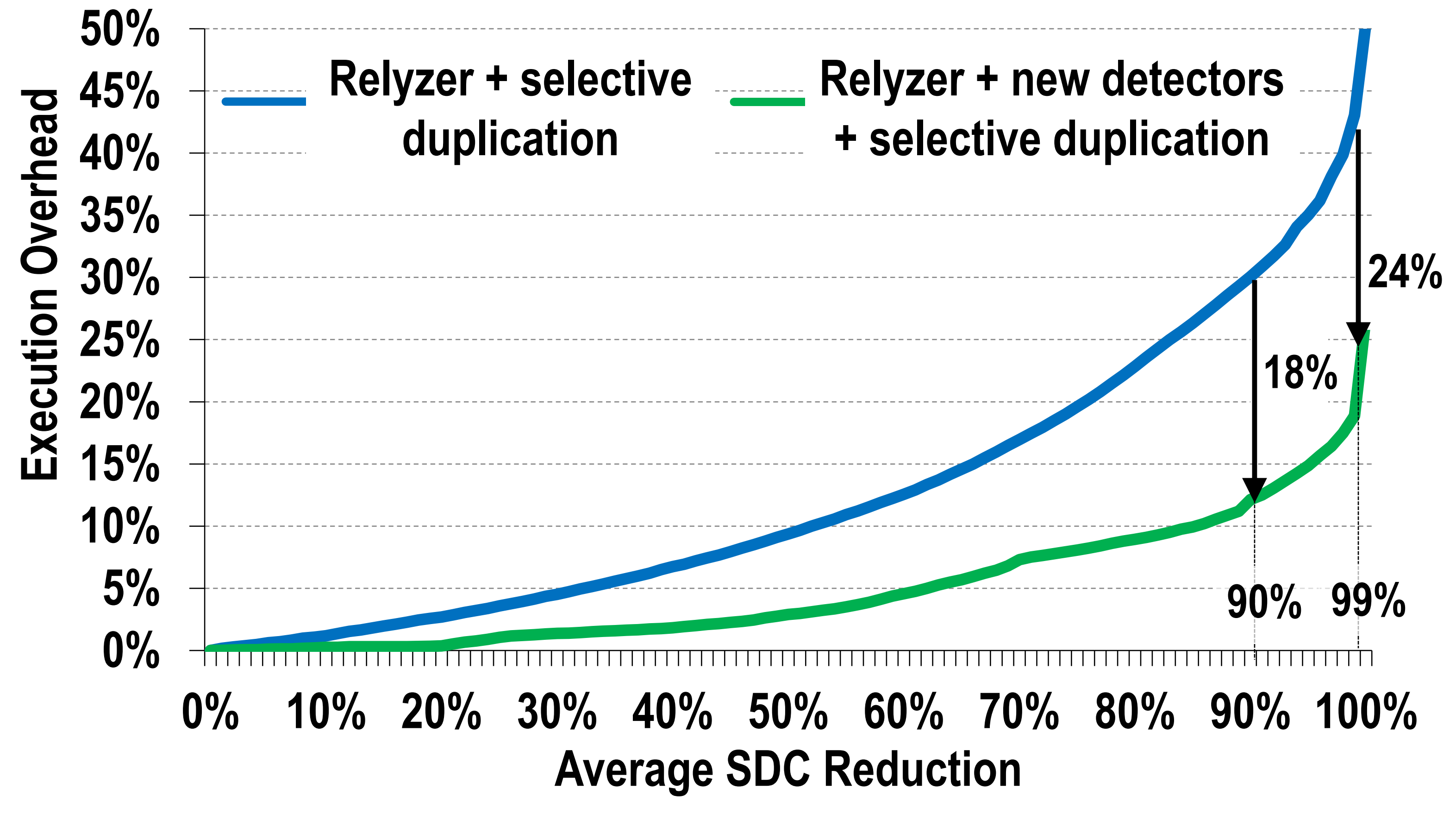
Need to find lowest-cost detectors for a target SDC reduction

For the first time, made possible by Relyzer

Our approach:

1. Set attributes of detectors
2. Formulate optimization problem
   Constraint: SDC red. ≥ Target
   Objective: Minimize overhead
3. Solution provides near optimal detectors





Conclusions

Relyzer finds a comprehensive list of SDC-producing application fault-sites
New program-level detectors cost effectively convert SDCs to detections
Relyzer + new detectors + selective duplication = Tunable resiliency at low cost

Ongoing Work

Developing fast fault simulation framework to speedup Relyzer even further
Modularized resiliency analysis to automatically find and protect SDC-vulnerable app sections
Program-level metrics to identify SDC producing app-sites without *any* injections