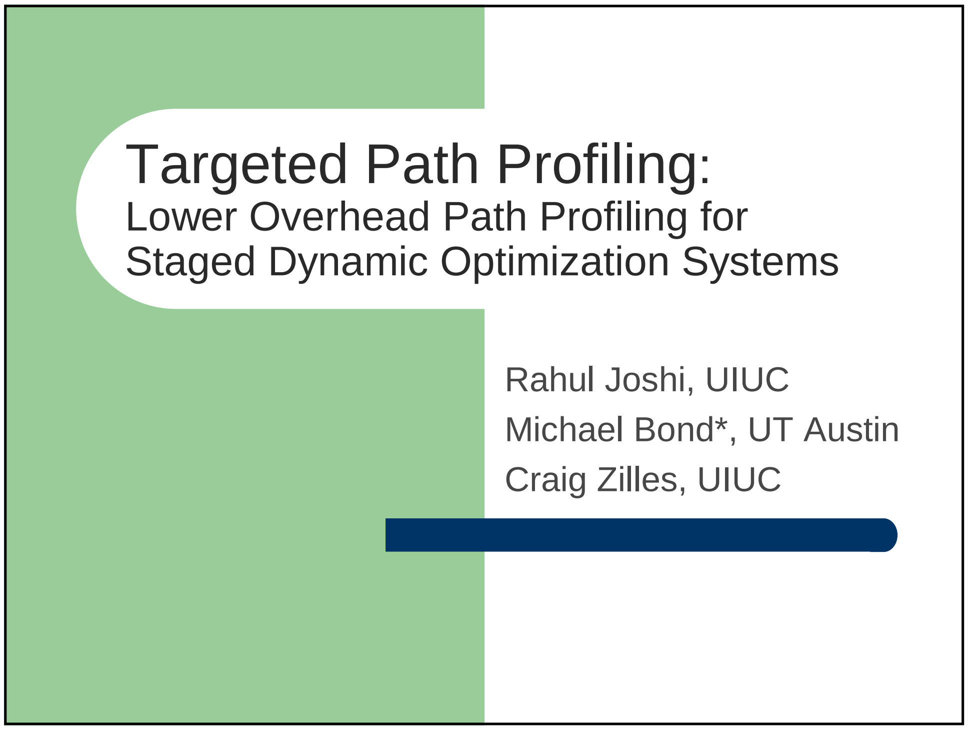
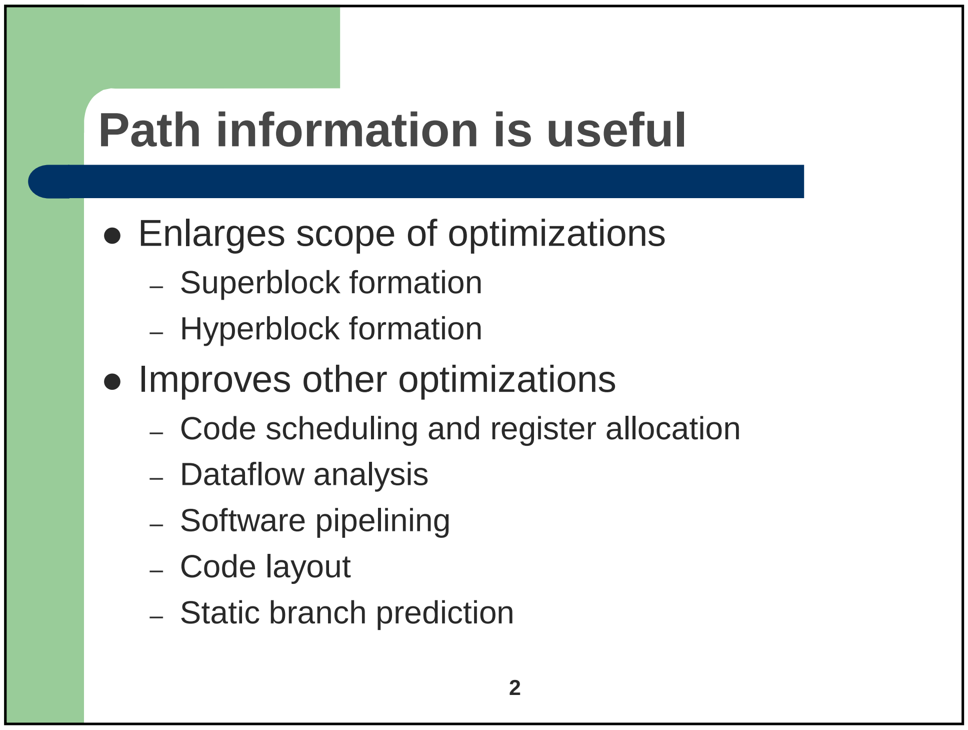


Targeted Path Profiling:
Lower Overhead Path Profiling for Staged Dynamic Optimization Systems

Rahul Joshi, UIUC
Michael Bond*, UT Austin
Craig Zilles, UIUC

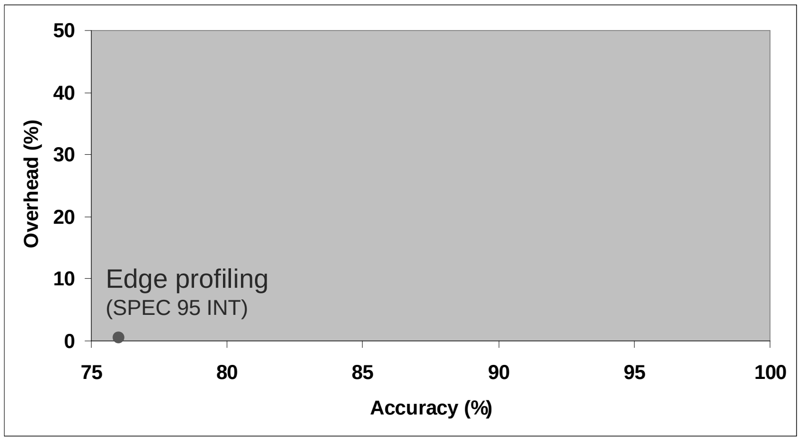


Path information is useful

- Enlarges scope of optimizations
  - Superblock formation
  - Hyperblock formation
- Improves other optimizations
  - Code scheduling and register allocation
  - Dataflow analysis
  - Software pipelining
  - Code layout
  - Static branch prediction

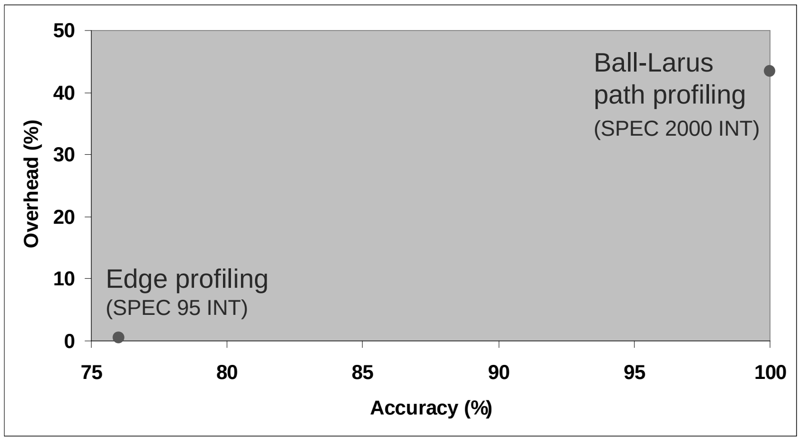
2

Overhead vs. accuracy



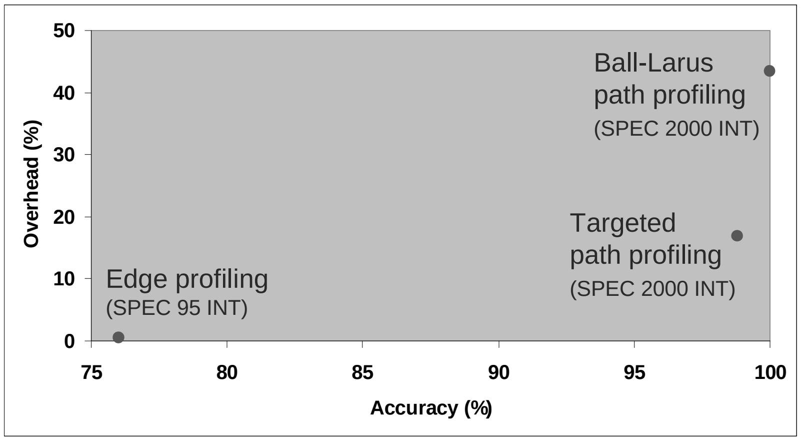
3

Overhead vs. accuracy



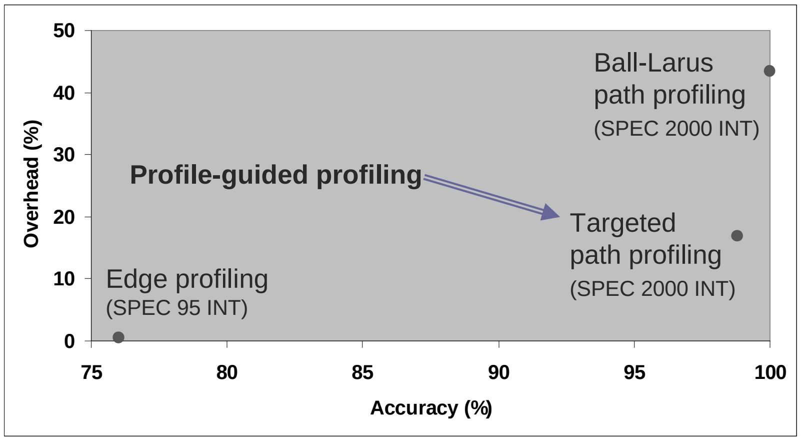
4

Overhead vs. accuracy

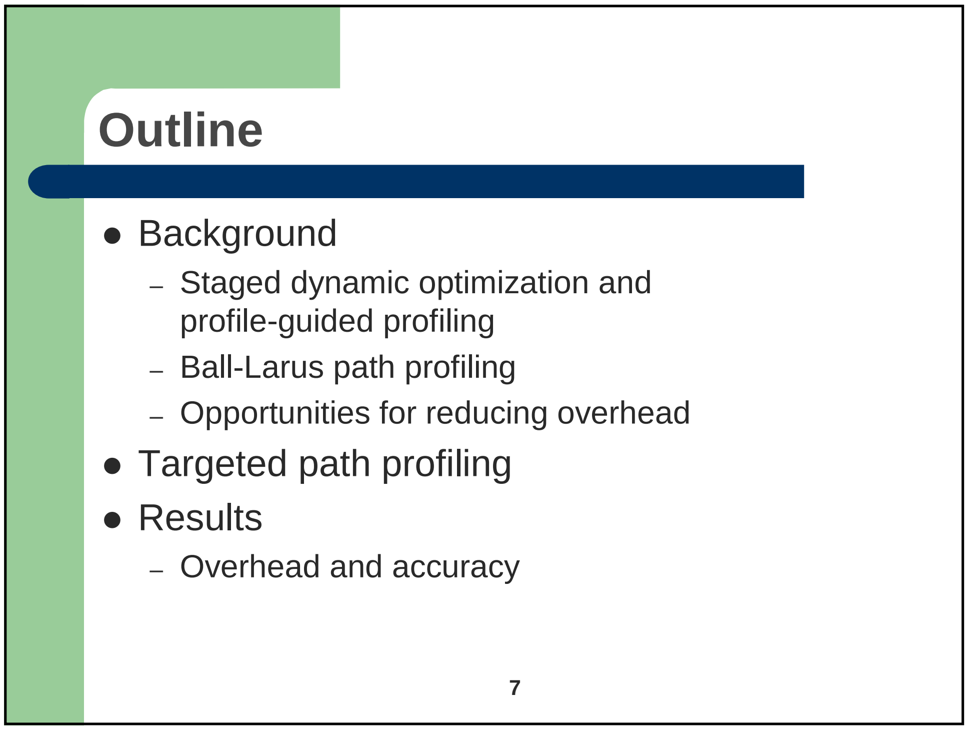


5

Overhead vs. accuracy



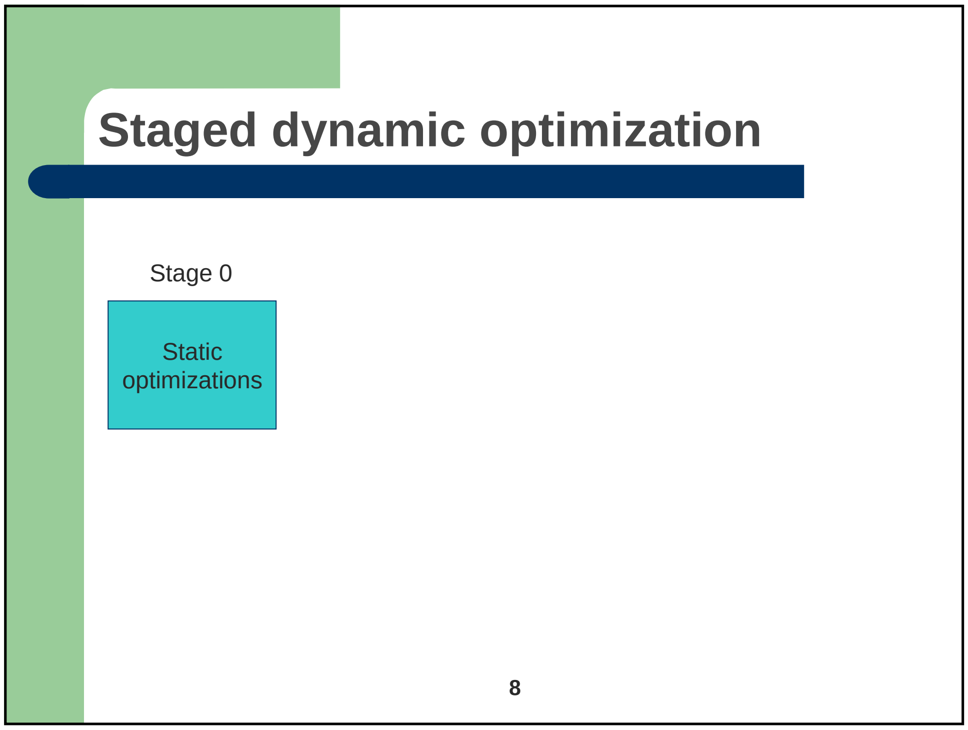
6



Outline

- Background
  - Staged dynamic optimization and profile-guided profiling
  - Ball-Larus path profiling
  - Opportunities for reducing overhead
- Targeted path profiling
- Results
  - Overhead and accuracy

7

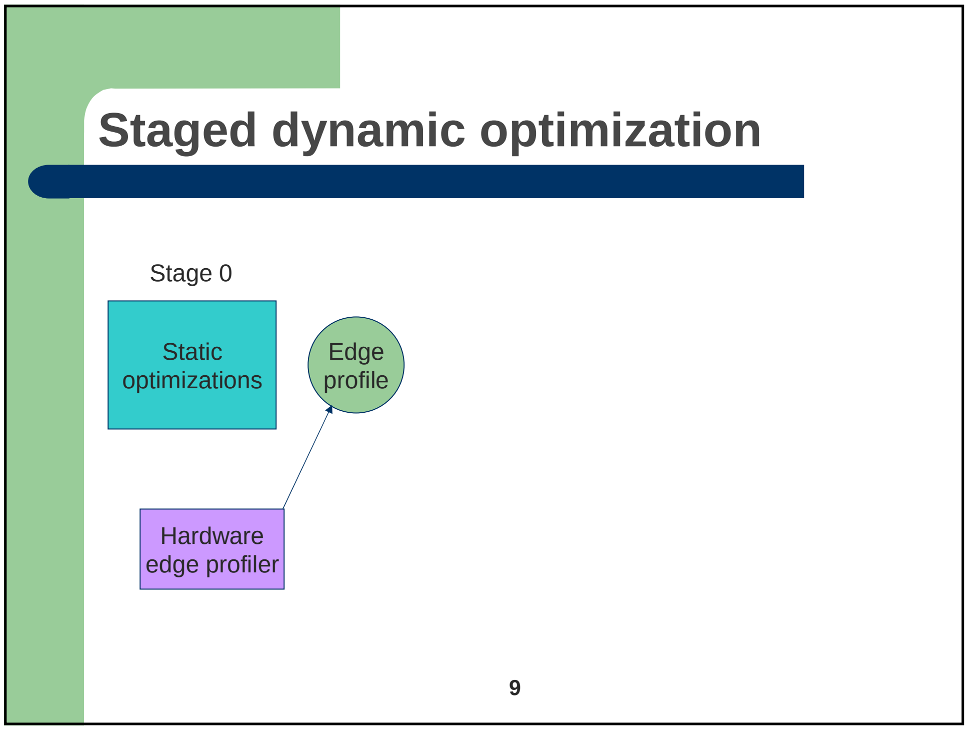


Staged dynamic optimization

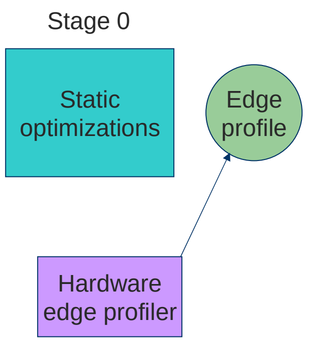
Stage 0

Static optimizations

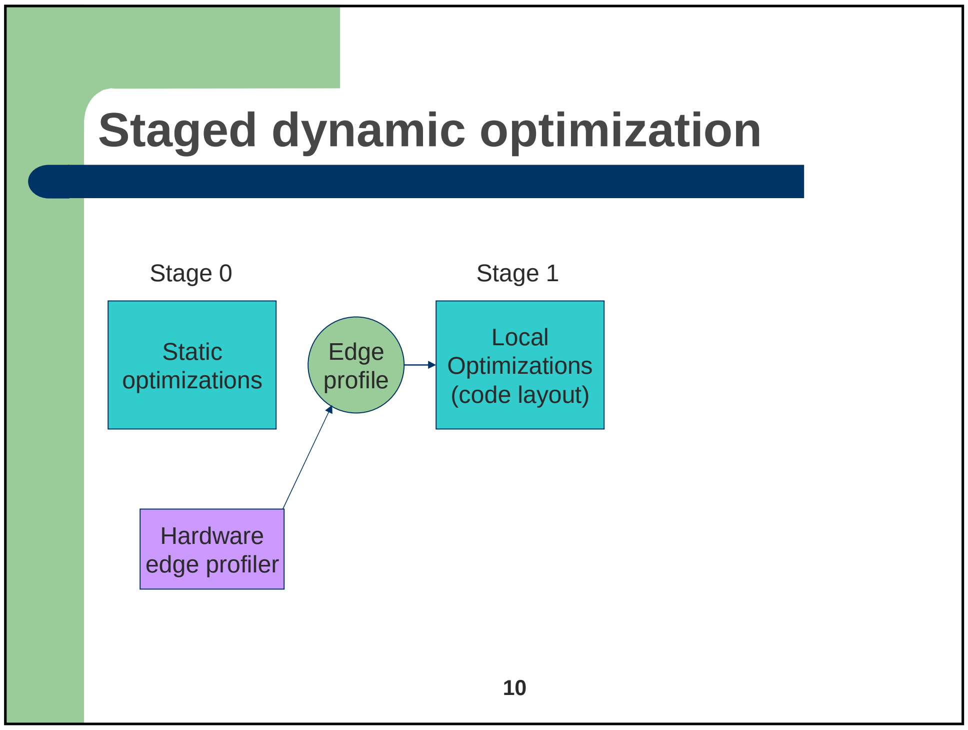
8



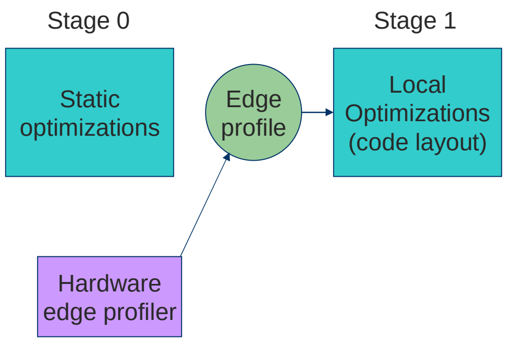
Staged dynamic optimization



9

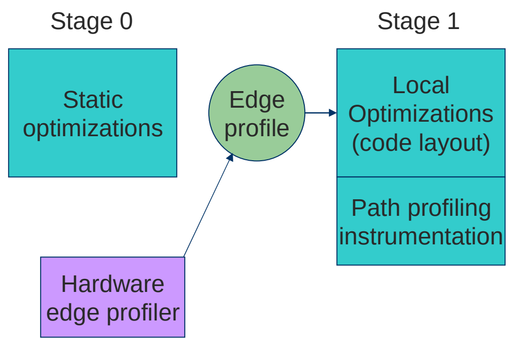


Staged dynamic optimization



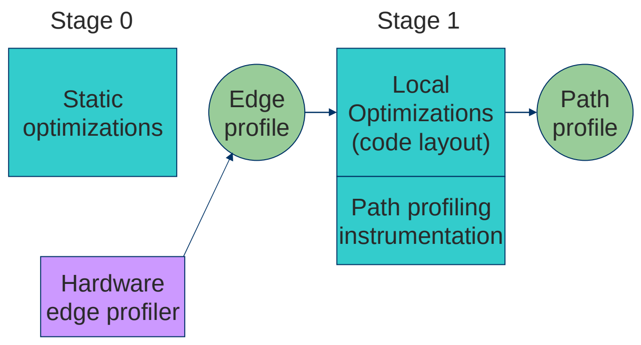
10

Staged dynamic optimization



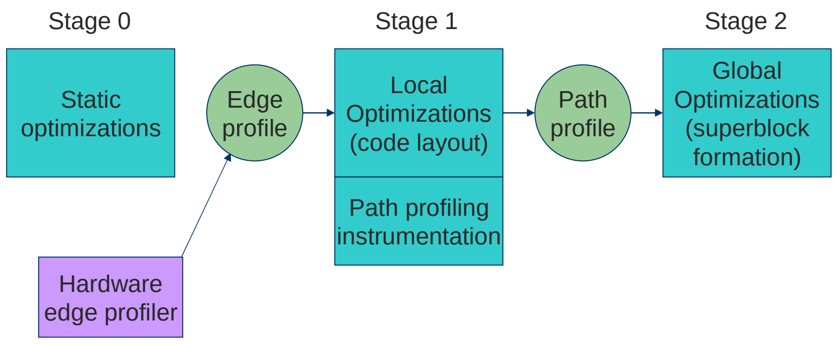
11

Staged dynamic optimization



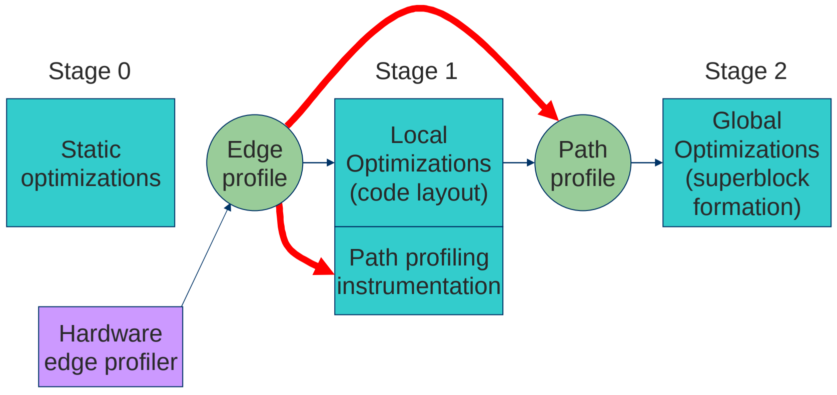
12

Staged dynamic optimization

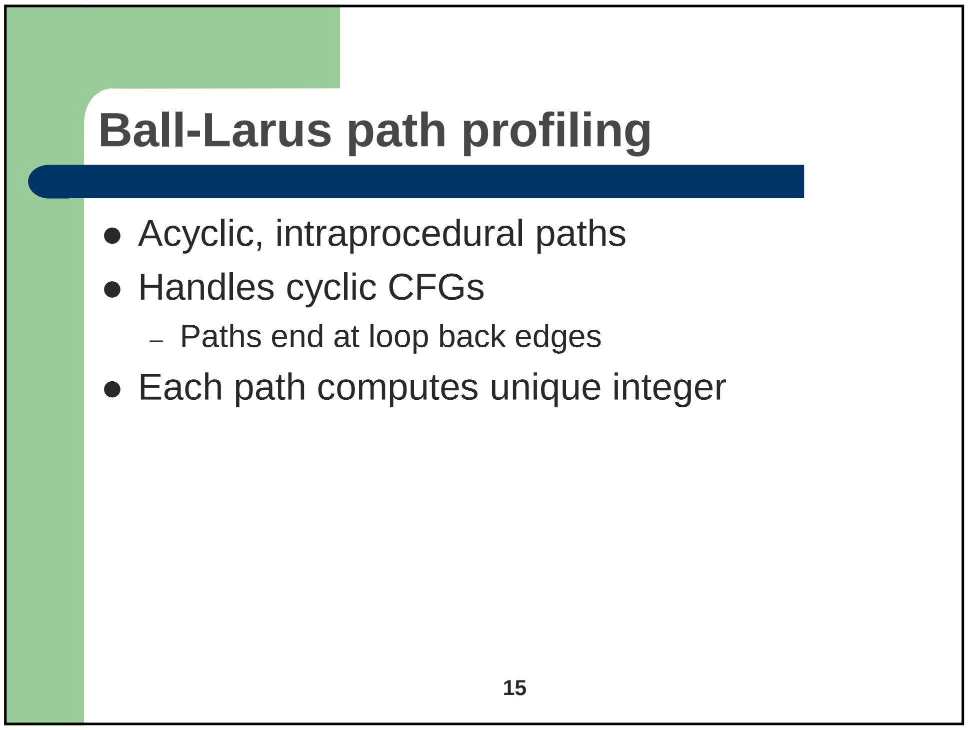

Stage 0
Stage 1
Stage 2
Static optimizations
Edge profile
Local Optimizations (code layout)
Path profile
Global Optimizations (superblock formation)
Path profiling instrumentation
Hardware edge profiler

13

Profile-guided profiling


Stage 0
Stage 1
Stage 2
Static optimizations
Edge profile
Local Optimizations (code layout)
Path profile
Global Optimizations (superblock formation)
Path profiling instrumentation
Hardware edge profiler

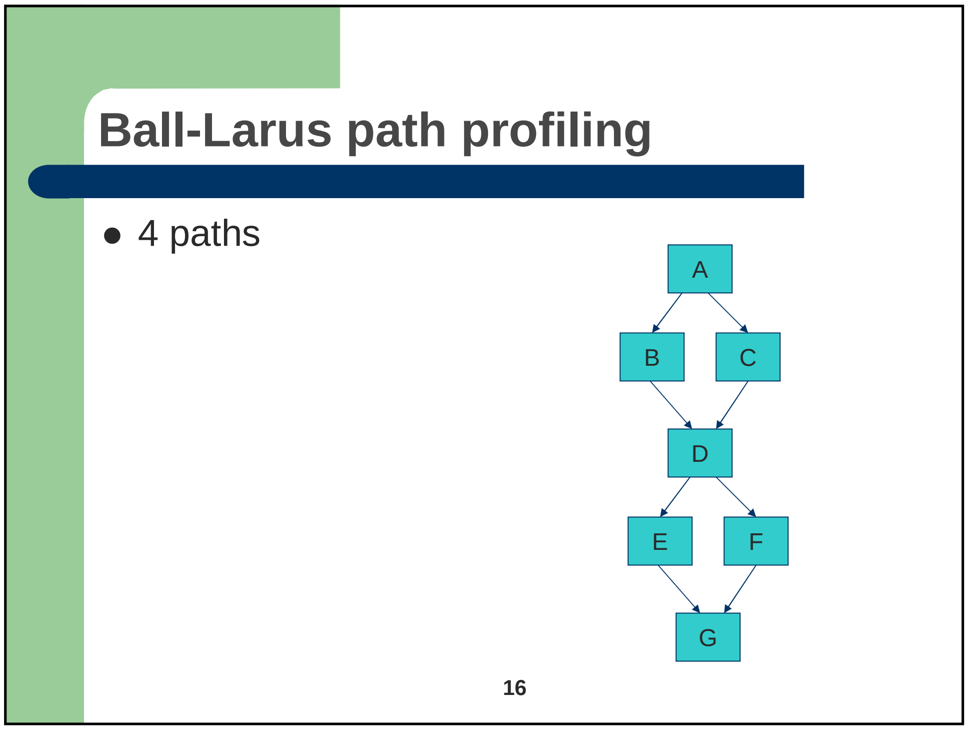
14



Ball-Larus path profiling

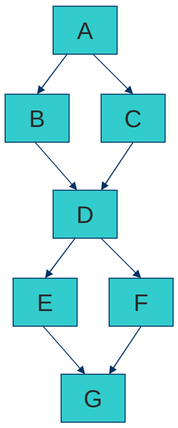
- Acyclic, intraprocedural paths
- Handles cyclic CFGs
  - Paths end at loop back edges
- Each path computes unique integer

15



Ball-Larus path profiling

- 4 paths

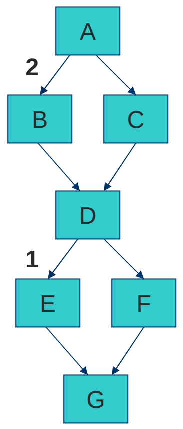


$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[dr] & & C \arrow[dl] \\ & D \arrow[dl] \arrow[dr] & \\ E \arrow[dr] & & F \arrow[dl] \\ & G & \end{tikzcd}$$

16

Ball-Larus path profiling

- 4 paths
- Each path computes unique integer

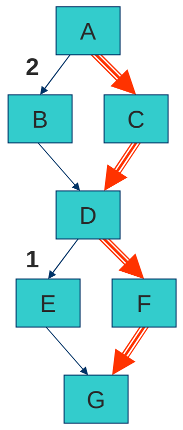


17

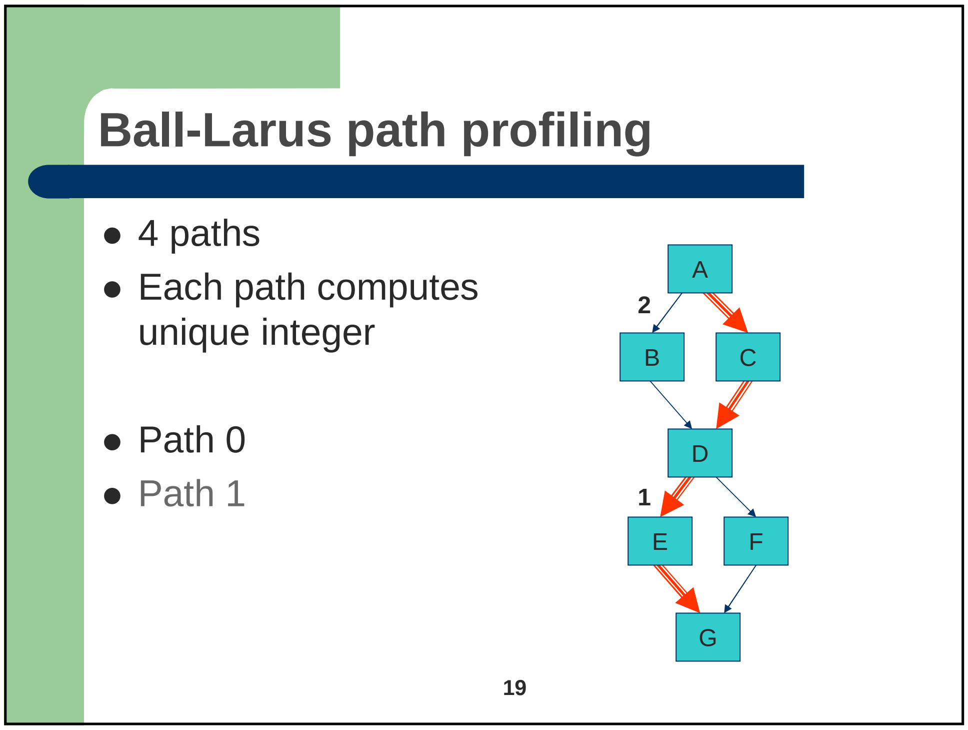
Ball-Larus path profiling

- 4 paths
- Each path computes unique integer

- Path 0



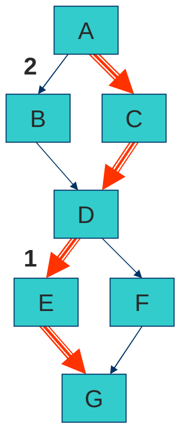
18



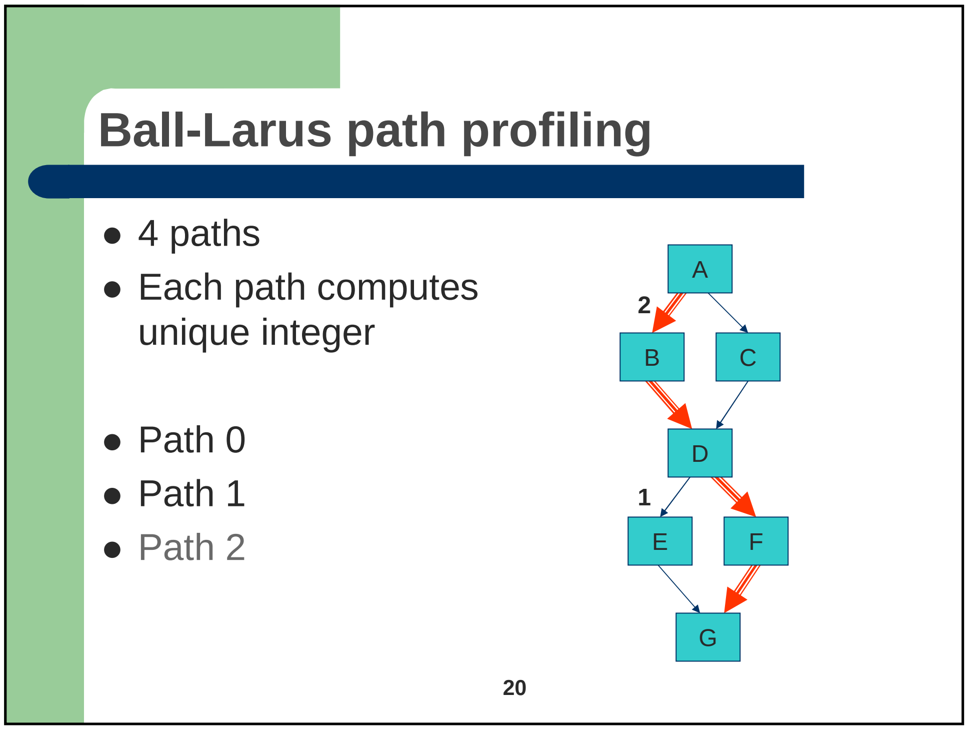
Ball-Larus path profiling

- 4 paths
- Each path computes unique integer

- Path 0
- Path 1



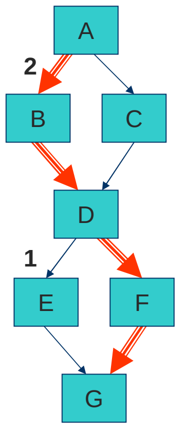
19



Ball-Larus path profiling

- 4 paths
- Each path computes unique integer

- Path 0
- Path 1
- Path 2

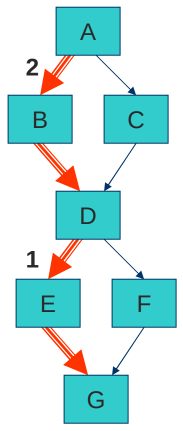


20

Ball-Larus path profiling

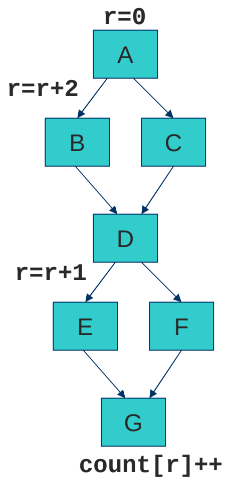
- 4 paths
- Each path computes unique integer

- Path 0
- Path 1
- Path 2
- Path 3



Ball-Larus path profiling

- `r`: path register
- `count`: array of path frequencies



Overhead in Ball-Larus path profiling

	SPEC 95	SPEC 2000
gcc	96%	87%
INT Avg	41%	43%
FP Avg	12%	22%
Overall Avg	**28%**	**37%**

23

Overhead in Ball-Larus path profiling

	SPEC 95	SPEC 2000
gcc	96%	87%
INT Avg	41%	43%
FP Avg	12%	22%
Overall Avg	**28%**	**37%**

- Opportunities for reducing overhead?
  - When there are many paths
  - When edge profile gives perfect path profile

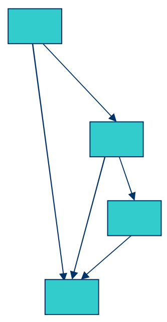
24

Routines with many paths

- Many *possible* paths
  - Exponential in number of edges
  - Can't use array of counters
- Number of *taken paths* small
  - Ball-Larus uses hash table
  - Hash function call expensive
    - Hashed path ~5 times overhead

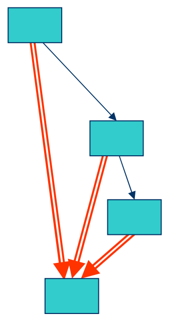
25

Edge profile gives perfect path profile



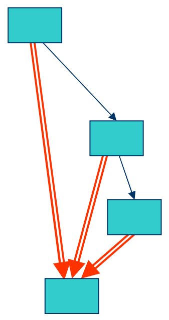
26

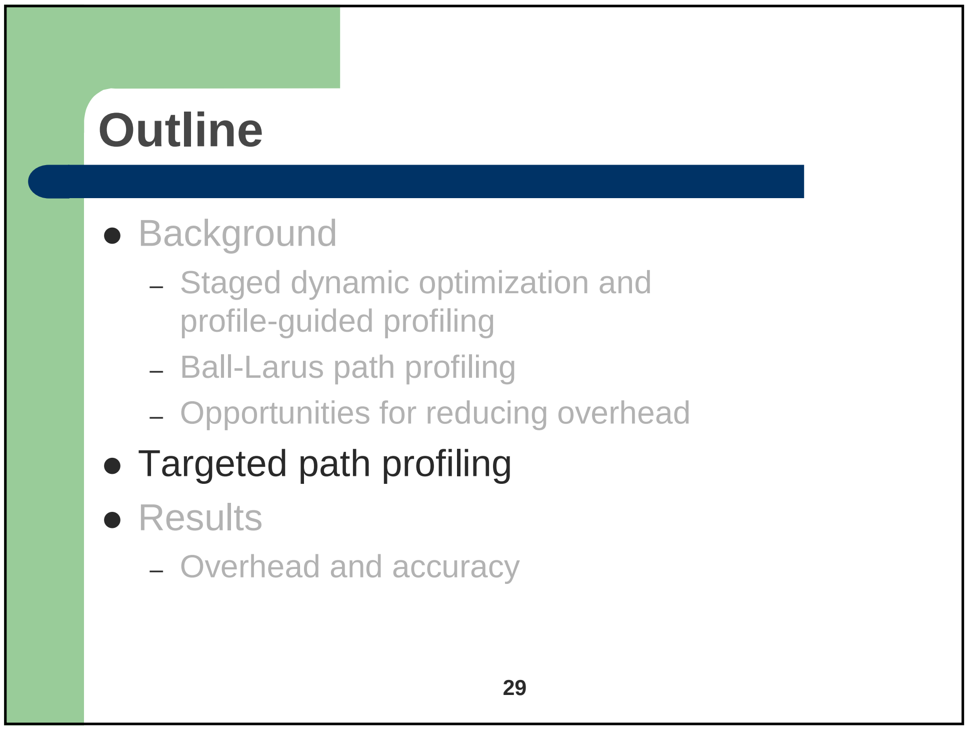
Edge profile gives perfect path profile



Edge profile gives perfect path profile

- An *obvious path* contains an edge that is only on that path
  - Path uniquely identified by edge
  - Path freq = edge freq
- If all paths obvious, edge profile gives perfect path profile

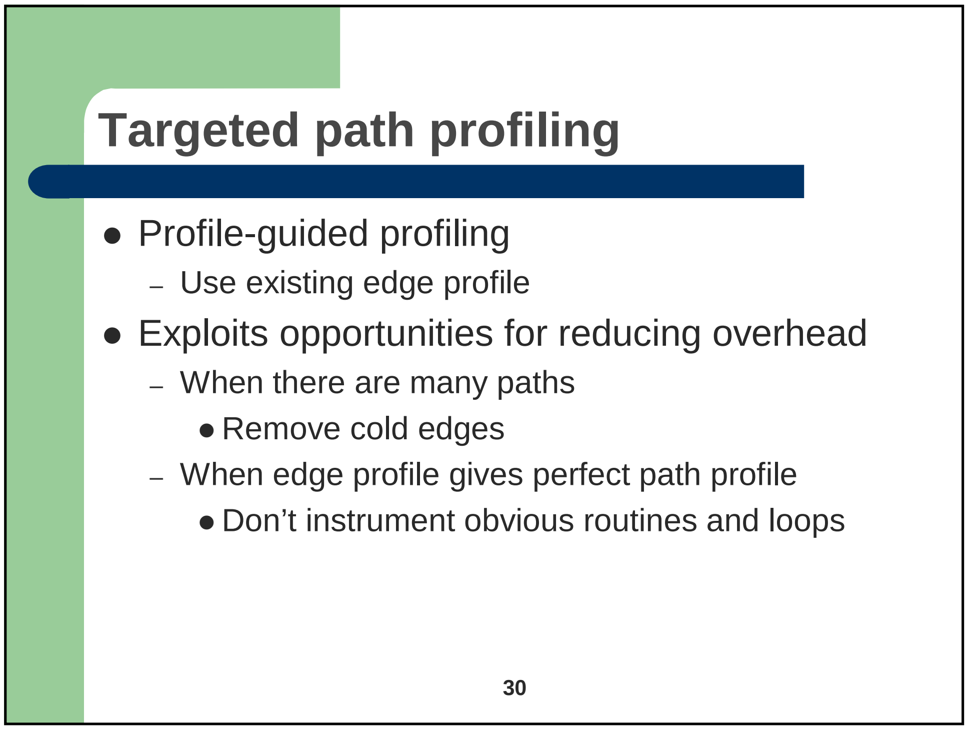




Outline

- Background
  - Staged dynamic optimization and profile-guided profiling
  - Ball-Larus path profiling
  - Opportunities for reducing overhead
- Targeted path profiling
- Results
  - Overhead and accuracy

29



Targeted path profiling

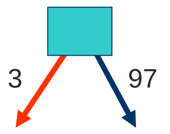
- Profile-guided profiling
  - Use existing edge profile
- Exploits opportunities for reducing overhead
  - When there are many paths
    - Remove cold edges
  - When edge profile gives perfect path profile
    - Don't instrument obvious routines and loops

30

Removing cold edges

- Examine relative execution frequency of each branch

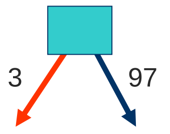
```
if (relFreq < threshold)
  edge is cold
```

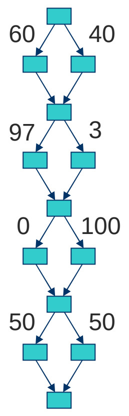


Removing cold edges

- Examine relative execution frequency of each branch

```
if (relFreq < threshold)
  edge is cold
```

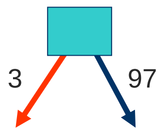


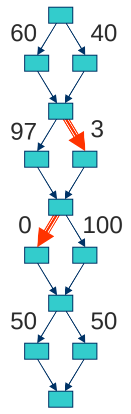


Removing cold edges

- Examine relative execution frequency of each branch

```
if (relFreq < threshold)
  edge is cold
```

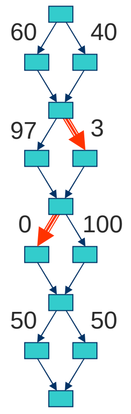




33

Removing cold edges

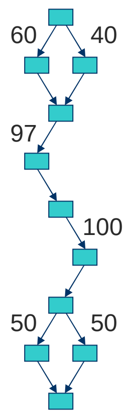
- A path that contains a cold edge is a *cold path*
- Removing an edge may halve number of paths



34

Removing cold edges

- A path that contains a cold edge is a *cold path*
- Removing an edge may halve number of paths
- Number of paths: 16 → 4

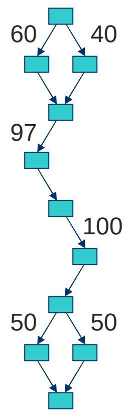


35

Removing cold edges

- A path that contains a cold edge is a *cold path*
- Removing an edge may halve number of paths
- Number of paths: 16 → 4

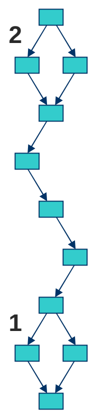
Goal: hashed → non-hashed



36

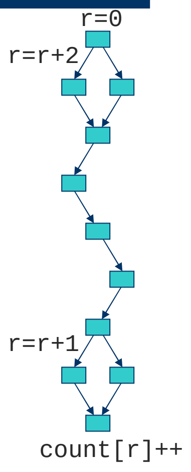
Removing cold edges

- Remaining paths potentially hot
- 4 paths → [0, 3]



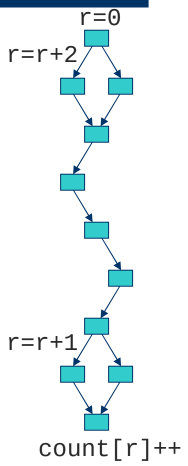
Removing cold edges

- Remaining paths potentially hot
- 4 paths → [0, 3]



Removing cold edges

- What if cold edge taken?

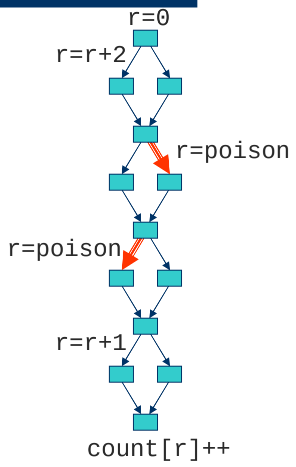


39

Removing cold edges

- What if cold edge taken?

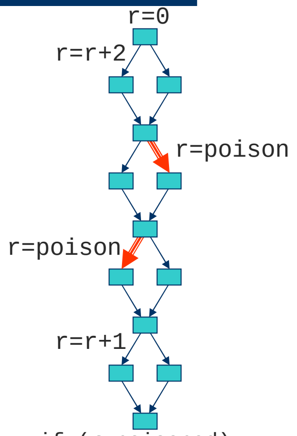
- Cold edges poison path



40

Removing cold edges

- What if cold edge taken?
- Cold edges poison path
- Instrumentation checks for poisoned path



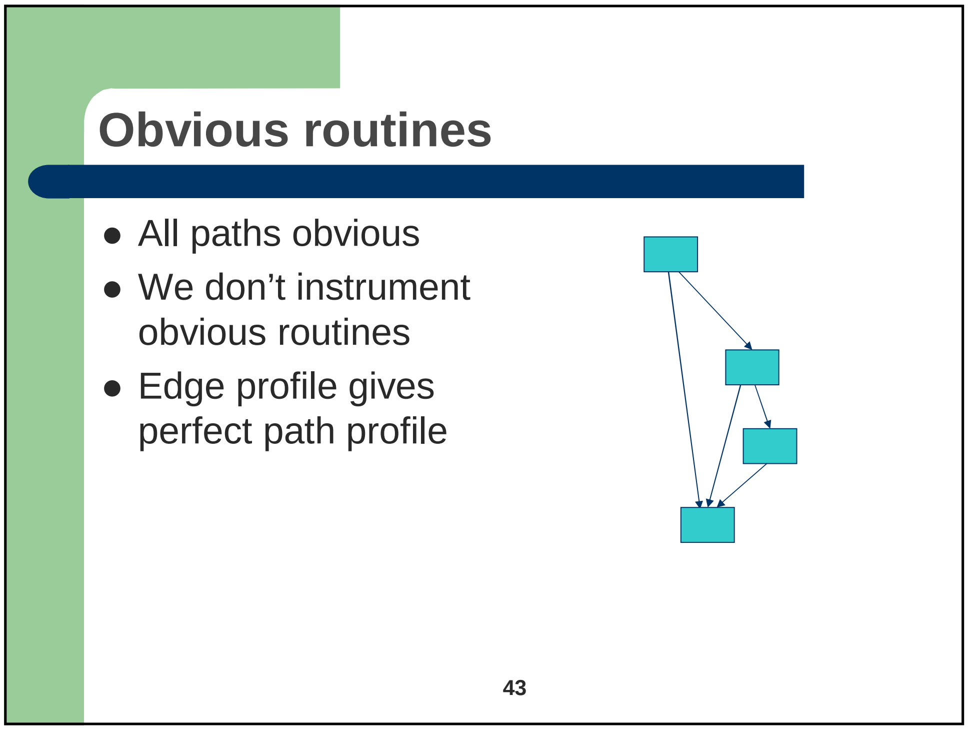
```
if (r poisoned)
  cold_counter++
else
  count[r]++
```

41

Checking for poison

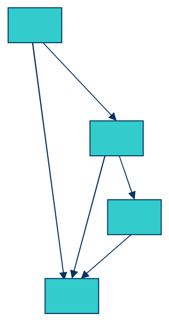
```
if (r poisoned)
  cold_counter++
else
  count[r]++
```

42

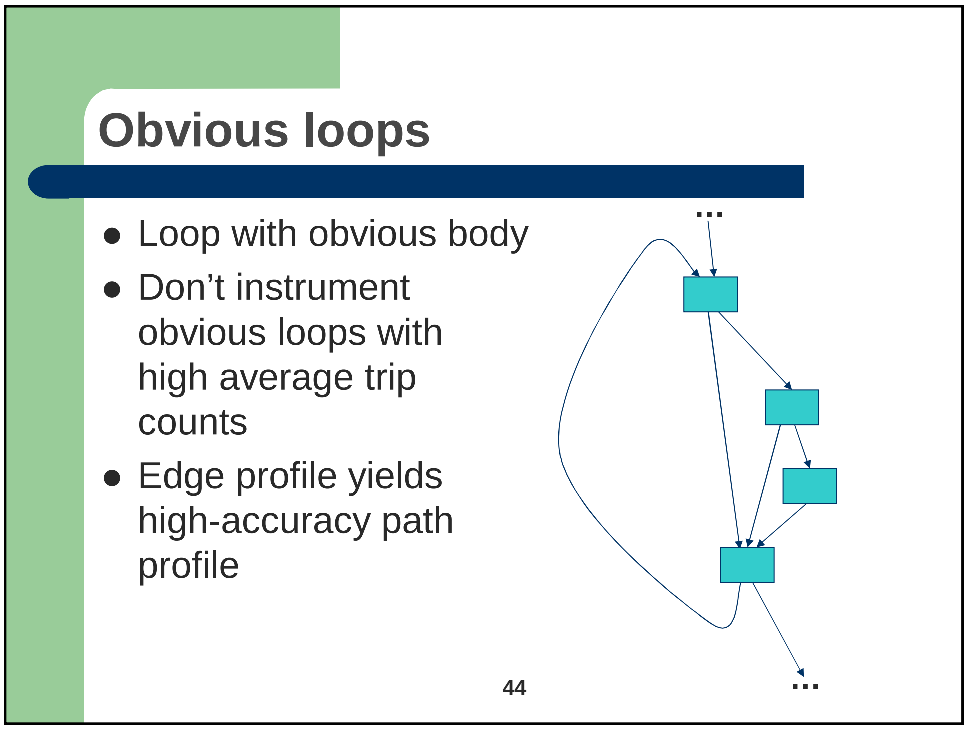


Obvious routines

- All paths obvious
- We don’t instrument obvious routines
- Edge profile gives perfect path profile

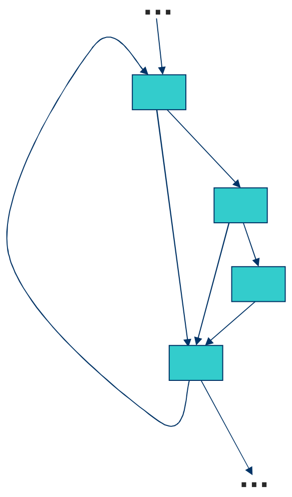


43



Obvious loops

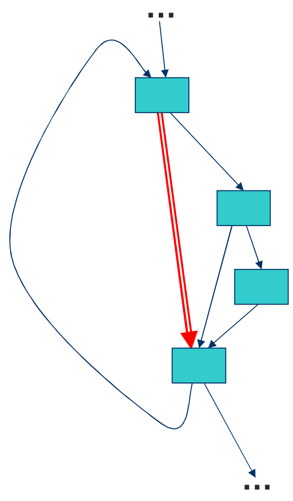
- Loop with obvious body
- Don’t instrument obvious loops with high average trip counts
- Edge profile yields high-accuracy path profile



44

Obvious loops

- Loop with obvious body
- Don’t instrument obvious loops with high average trip counts
- Edge profile yields high-accuracy path profile



45

Summary of our techniques

- Remove cold edges
  - Eliminates many cold paths
  - Count paths with array (instead of hash table)
- Don’t instrument obvious routines and loops
  - Edge profile derives path profile

46

Outline

- Background
  - Staged dynamic optimization and profile-guided profiling
  - Ball-Larus path profiling
  - Opportunities for reducing overhead
- Targeted path profiling
- Results
  - Overhead and accuracy

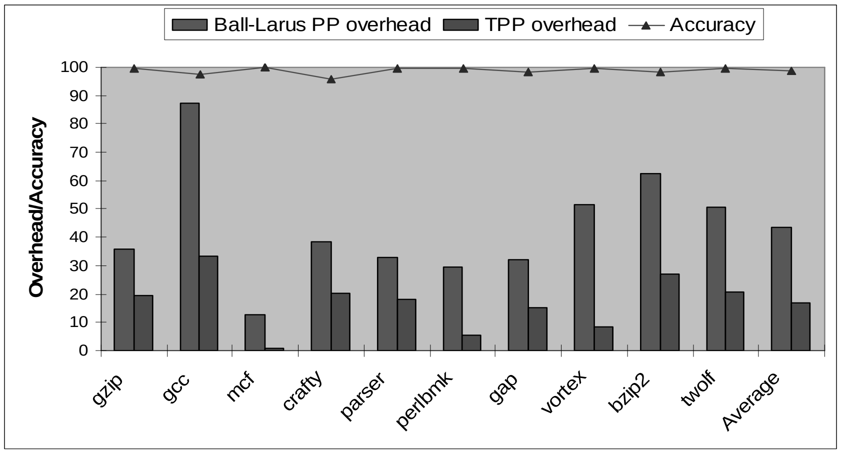
47

Implementation

- Static profiling
- **PP**: tool for path profiling
- **TPP**: tool for targeted path profiling
- Tools instrument native SPARC executables
  - SPEC 95 **`ref`**
  - SPEC 2000 **`ref`**

48

Results: SPEC 2000 INT



49

Where does benefit come from?

- Cold path elimination alone: 60%
- Add obvious path elimination: + 40%

- Little benefit from obvious path elimination alone

50

Related work

- Dynamo [Bala et al. ‘00]
  - Successful online path-guided optimization
  - “Bails out” when no dominant path
- Instrumentation sampling [Arnold & Ryder ‘01]
  - Orthogonal to targeted path profiling
- Selective path profiling [Apiwattanapong & Harrold ’02]
  - Useful when only a few paths of interest

51

Summary

- Profile-guided profiling in a staged dynamic optimization system
- Two synergistic techniques
  - Remove cold paths
  - Don’t instrument obvious routines and loops
- Reduces overhead by half (SPEC 95) to two-thirds (SPEC 2000)
- High accuracy: ~99%

52