Profiling The Profiler:

New Metrics to Evaluate and Improve Profile Guided Optimization

Micah Weston

Stan Kvasov, Vince Del Vecchio

Profile Guide Optimization (PGO) Overview

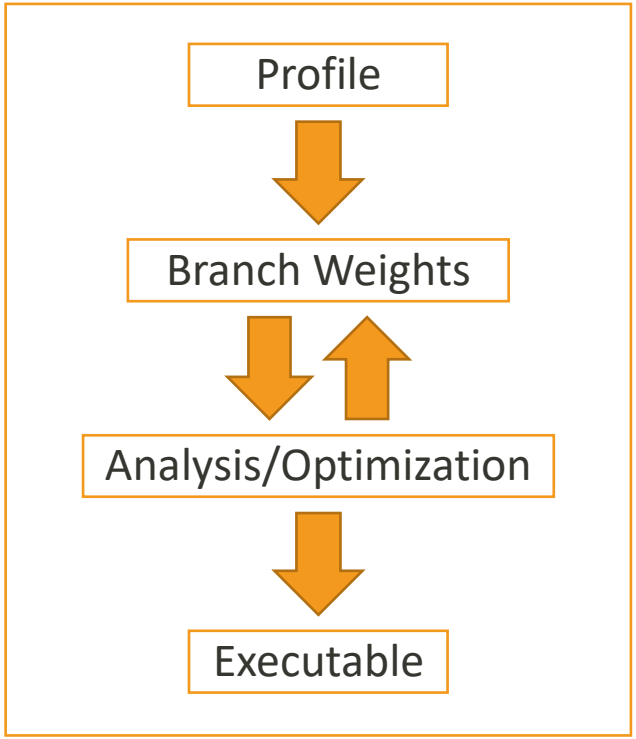
PGO: optimization decisions based on runtime information

- Improved accuracy leads to better optimization decisions

Goal: create metrics to judge PGO information accuracy

- Help track regressions and evaluate improvements
- Help direct bug finding and feature creation

Profile Flow in the Compiler



PGO Metadata

LLVM IR

```
define i32 @dispatch(ptr %f) !prof !1 {
entry:
  %cmp = icmp eq ptr %f, null
  br i1 %cmp, label %return, label %if.else, !prof !2

if.else:
  %call = tail call noundef i32 %f()
  br label %return

return:
  %retval.0 = phi i32 [ %call, %if.else ], [ 0, %entry ]
  ret i32 %retval.0
}

!1 = !{!"function_entry_count", i64 2590}
!2 = !{!"branch_weights", i32 2144, i32 446 }
```

LLVM MIR

```
bb.0.entry:
  successors: %bb.2(0x69f5533a), %bb.1(0x160aacc6)
            ; %bb.2(82.78%),     %bb.1(17.22%)
  liveins: $rdi
  %0:gr64_tc = COPY $rdi
  TEST64rr %0:gr64_tc, %0:gr64_tc, implicit-def $eflags
  JCC_1 %bb.2, 4, implicit $eflags
  JMP_1 %bb.1

bb.1.if.else:  ; predecessors: %bb.0
  TCRETURNri64 %0:gr64_tc, 0, <regmask ...>, implicit $rsp,
implicit $ssp

bb.2.return:   ; predecessors: %bb.0
  %1:gr32 = MOV32r0 implicit-def dead $eflags
  $eax = COPY %1:gr32
  RET 0, $eax
```

Existing PGO Metrics

Metrics	Qualities: Full Realistic Compilation	Avoids Noise*	Indicates PGO Accuracy
Performance – Aggregate Performance Measurement	✓	✗ Sensitive to binary layout or OS scheduling	✗ Not always correlated
Import Quality – Distance from a ground truth during profile loading [1][2]	✗ Must disable inlining to align comparison	✓	~ Only at import time
Unit Tests – Case by case testing of specific changes	✗ Inherently abstracted cases	✓	~ Only for that case

Can we do better?

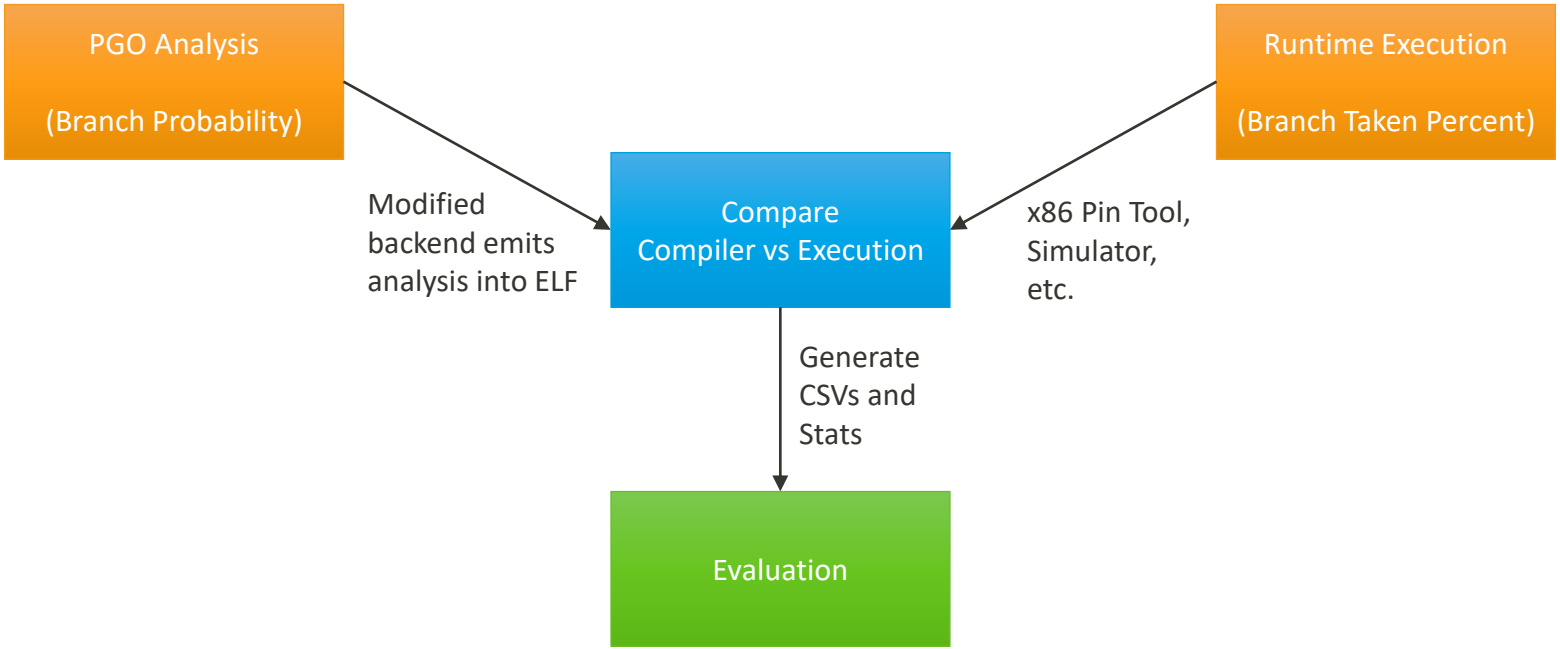
[1] E. Raman and X. D. Li, "Learning branch probabilities in compiler from Datacenter workloads," arXiv.org, https://arxiv.org/abs/2202.06728 (accessed Aug. 22, 2023).
- Compared ML generated weights to the ground truth of Sample PGO

[2] W. He, J. Mestre, S. Pupyrev, L. Wang, and H. Yu, "Profile Inference Revisited," Proceedings of the ACM on Programming Languages, https://dl.acm.org/doi/10.1145/3498714 (accessed Aug. 22, 2023).
- Compared Profi generated Sample PGO weights to the ground truth of Instrumentation PGO

* Assumes deterministic programs

Proposed PGO Accuracy Metrics

- Cross-checking PGO Analysis against Runtime Traces



Metric 1 - Branch Direction Match

```
   (probability >  0.50 and taken_percent >  0.50)   # biased taken
or (probability <  0.50 and taken_percent <  0.50)   # biased non-taken
or (probability == 0.50 and taken_percent == 0.50)   # unbiased
```

Simple Boolean: true/false corresponds to match/mismatch

- Variables derived from the jump/taken edge of a conditional branch
- Special case 50% is treated as neither taken or non-taken to avoid skewing over percent match

Overall Metric

% Direction Match	0.944
Total Matching Jumps	8103
Total Jumps	8582

Individual Branches

Jump Index	Execution Count	Probability	Taken Percent	Direction Match
0	320	0.951	0.855	Yes
1	2100	0.499	0.222	Yes
2	5623	0.040	0.111	Yes
3	60	0.500	0.500	Yes
4	250	0.333	0.667	No
5	89	0.500	0.501	No
6	140	0.003	0.734	No

Metric 2 - Branch Probability Error

$$\frac{\sum_b \left[\left(P_c(b) - P_e(b) \right)^2 \times T(b) \right]}{\sum_b [T(b)]}$$

Weighted average of squared error:

Per condition jump b

- $P_c(b)$ = *Predicted* **c**ompiler probability
- $P_e(b)$ = *Actual* **e**xecution taken percent
- $T(b)$ = Total jump occurrences

Overall Metric

Probability Error	0.034
Sum of Weighted Error	295.125
Total Jumps	8582

Jump Index	Execution Count	Probability	Taken Percent
0	320	0.951	0.855
1	2100	0.499	0.222
2	5623	0.040	0.111
3	60	0.500	0.500
4	250	0.333	0.667
5	89	0.500	0.501
6	140	0.003	0.734

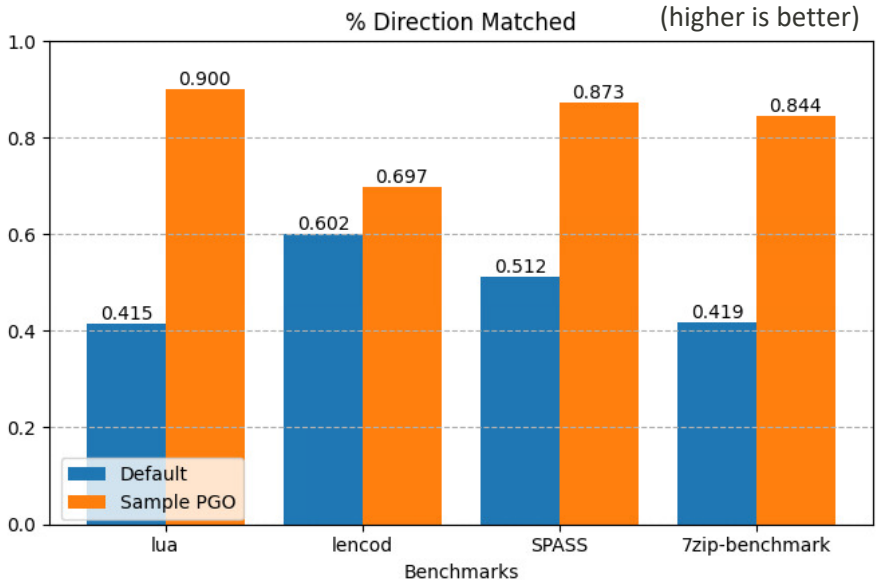
Metrics Revisited

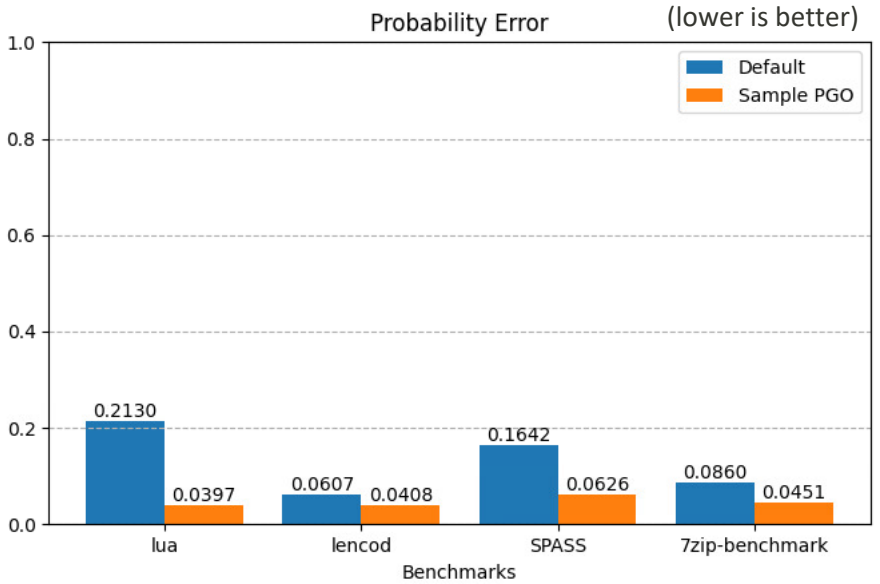
Metrics	Qualities: Full Realistic Compilation	Avoids Noise*	Indicates PGO Accuracy
Performance – Aggregate Performance Measurement	✓	✗ Sensitive to layout changes or OS scheduling	✗ Not always correlated
Import Quality – Distance from a ground truth during profile loading	✗ Must disable inlining to align comparison	✓	~ Only at import time
Unit Tests – Case by case testing of specific changes	✗ Inherently abstracted cases	✓	~ Only for that case
New PGO Accuracy Metrics – Branch direction match and probability error	✓	✓	✓

* Assumes deterministic programs

Baseline Metrics on Sample PGO Benchmarks

- 4 benchmarks taken from llvm-test-suite
- Compiled using ReleaseThinLTO

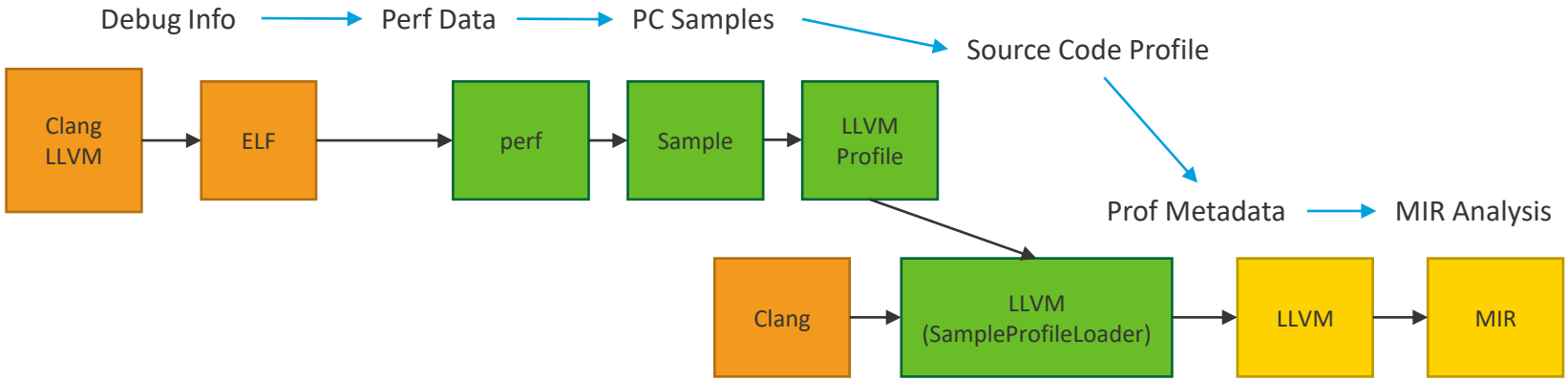




Sample PGO Zones for Improvement

Zones:

- **Sampled Debug Info – compiling the original program**
- **Profile Loading – collecting, converting, and mapping samples**
- **PGO Metadata Usage – optimizing and manipulating IR**



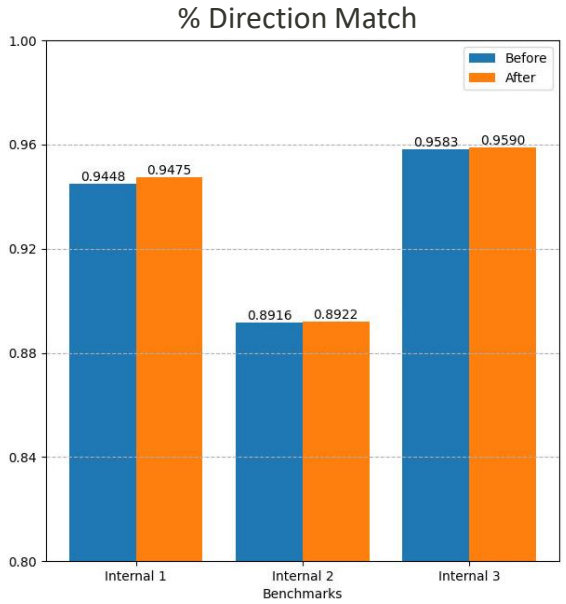
Metrics in Use – Improved Backend Debug Info Handling

- Updated debug info handling in our internal backend
  - Following guidance from LLVM's How to Update Debug Info: A Guide for LLVM Pass Authors
  - Using drop location when performing hoisting
- Performance did not show change, but PGO accuracy showed slight improvement

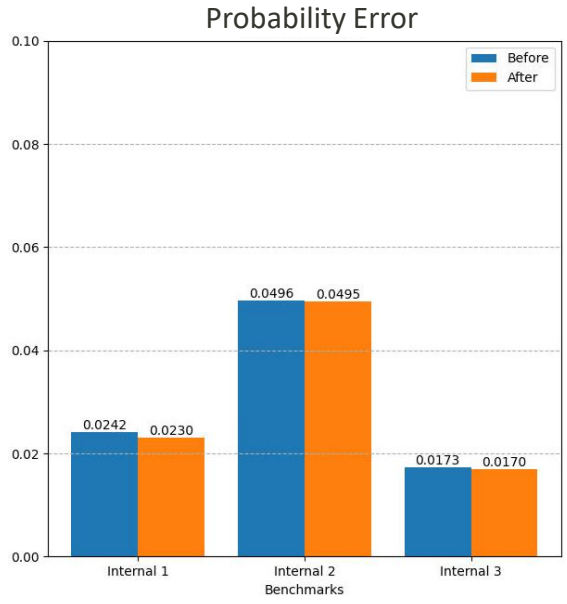
```
while (cond) {
    // ...
    int val = INVARIANT_CODE;
    // ...
}
```

→

```
int val = INVARIANT_CODE;
while (cond) {
    // ...
}
```



(higher is better)



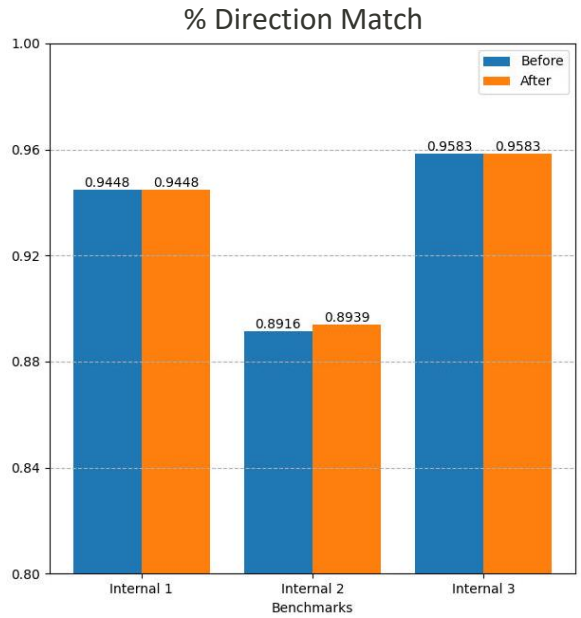
(lower is better)

Metrics in Use – Constant Iteration Loops

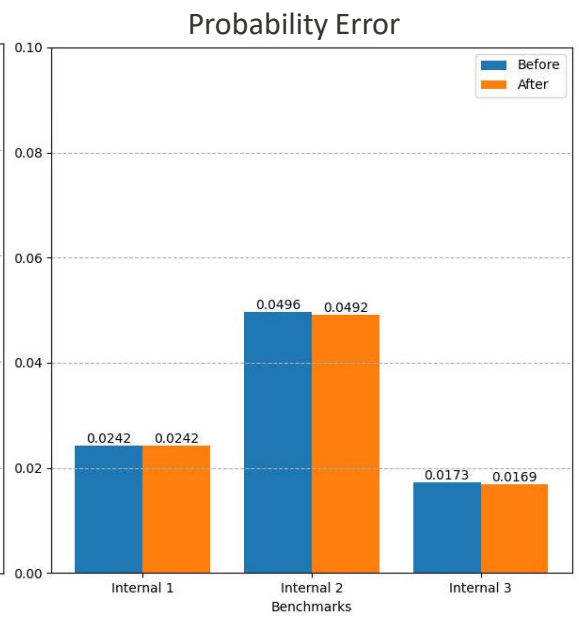
- Sample PGO errors may accidentally bias exiting edge of a loop
  - Compiler can detect and adjust this on **constant iteration loops**
- Detected by **patterns in the individual branch direction match** data when optimizing for size

```
// ... code ...
for (int i = 0; i < 6; i++) {
  // ... code ...
}
// ... code ...
```

Original Sample	Adjusted
- Branch Weights - Loop Back: 2610 Exiting : 9018	- Branch Weights - Loop Back: 9967 Exiting : 1661
- Total Weight - 11628	- Total Weight - 11628
- Loop Back Prob - 22.4%	- Loop Back Prob - 85.7 %



(higher is better)



(lower is better)

Conclusions

New metrics help improve coverage for PGO evaluation and tracking

Next Steps:

- RFC in progress and then PR into upstream
- Incorporate BlockFrequencyInfo into metrics

Contacts

- Micah – micah.weston@mediatek.com
- Stan – stan.kvasov@mediatek.com
- Vince – vince.delvecchio@mediatek.com

Links:

- RFC: Discourse Link
- Patches: D158889 and D158890

MEDIATEK