Static Branch Frequency and Program Profile Analysis

Youfeng Wu
Sequent Computer Systems Inc.

James R. Larus
University of Wisconsin at Madison

Micro-27
San Jose
Nov. 30 - Dec. 2, 1994

Outline

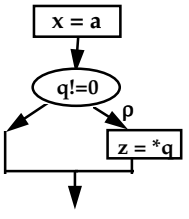
- **Motivation**
- **Branch Probability**
- **Branch Frequency**
- **Function Call Frequency**
- **Global Branch Frequencies**
- **Experiment and Results**
- **Summary and Future Work**

Applications of Program Profiles

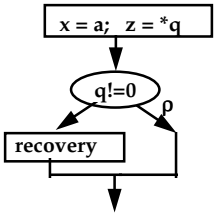
- Branch probability: speculative scheduling
- Branch frequency: branch optimization, instruction cache locality
- Function call frequency: inter-procedural optimization, memory locality
- Block and function execution frequencies: focus optimization, performance analysis

Example: Speculative Scheduling

- Objective: moves instructions across branches to reduce critical path length on wide-issue processors



cost = 2 + ρ



cost = 2 + (1- ρ)*recovery

- If the speculation is wrong, major penalty may occur to recover from the mistake.
- Accurate branch probability is crucial

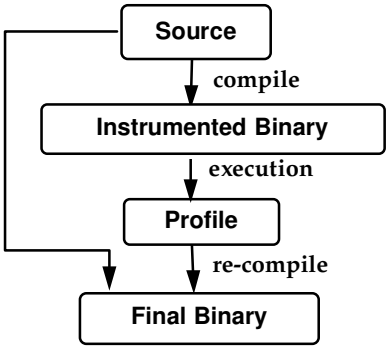
Example: Branch Optimization

- Objective: minimize branch mis-prediction.
- Hardware predicts fall-through and backward branches taken. Mis-prediction costs 20+ cycles.
- Branch frequency information is essential to maximize fall-through and backward branches.

Example: Ordering for Locality

- Objective: Improve cache and memory reference locality
- Placing basic blocks that frequently branch to each other in the same cache line reduces cache misses
- Placing functions that frequently call each other in the same page reduces page fault rate and bus traffic

Dynamic Profiling



- Accurate
- User intervention complicates compilation process
- Affects real time and reactive systems

Static Profiling

Analyze source, estimate profile, and use the information in compiler at the same time.

- Branch directions are relatively independent of input (Fisher-92)
- Relative ordering of profile information, not accurate counts, is important
- Relative ordering is independent of input data
- Powerful heuristics accurately predict conditional branches (Ball-93)
- Evaluate static profiles by matching the relative ordering against dynamic profiles.

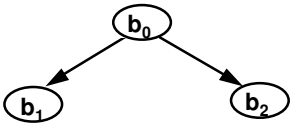
Heuristics for Predicting Branch Directions

- Loop branch: predict back edge taken
- Pointer: predict that a pointer never equals null or two pointers are never equal
- Opcode: predict that the comparison of an integer variable for less than zero fails
- Return: predict that successor containing a return statement is not taken
- etc.

With eight heuristics:

80% of the dynamic branch directions are predicted correctly [Ball-93]

Branch Prediction and Branch Probability



- Assume a heuristic predicts b_1 taken
- What is probability of b_0—>b_1 taken at runtime?
- 80-20 rule [Wagner-94]: the predicted taken branch has a 80% taken probability

  Toss a biased coin, seeing head more often does not mean the probability of head is 80%

Branch Probability

Toss a coin, what's probability of head?

$$\lim_{n \to \infty} \frac{\text{number of heads}}{n \text{ (number of tosses)}}$$

For a heuristic, what is probability that the predicted taken branch will be taken at runtime?

$$\lim_{n \to \infty} \boxed{\frac{\text{number of times that prediction is correct at runtime}}{n \text{ (total number of executions of the predicted branch)}}}$$

⇩

Hit Rate (for reasonably large n)

Branch Probability of Predicted Taken Branch = Hit Rate

Heuristics	Hit Rates
Loop branch	88%
Pointer	60%
Opcode	84%
Guard	62%
Loop exit	80%
Loop header	75%
Call	78%
Store	55%
Return	72%

Resolve Conflicting Predictions

```
if ( i < 0 ) then x = y;
else { x = 1; return; }
```

Opcode Heuristic predicts "else" branch: 84%
Return Heuristic predicts "else" branch: 28%

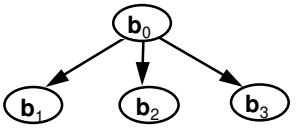
What should be the combined probability of the if-branch or else-branch?

Combining Probabilities

- What will you do?
  Doctor #1 says you need a surgery
  Doctor #2 says you have no problem
  Doctor #3 ...

- Total ordering [Ball-93]: order the doctors and always pick up the first one.
  Should we seek second opinion?
- Dempster-Shafer algorithm: assign credibility to the doctors and calculate an overall decision.

Assigning Probabilities to Sets of Outcomes



Heuristic 1: b_1 taken 80%

$p(\{b_0 \rightarrow b_1\}) = .8$

$p(\{b_0 \rightarrow b_2, b_0 \rightarrow b_3\}) = .2$

Heuristic 2: b_2 taken 60%

$p(\{b_0 \rightarrow b_2\}) = .6$

$p(\{b_0 \rightarrow b_1, b_0 \rightarrow b_3\}) = .4$

Model "unknown":

$p(\{b_0 \rightarrow b_2, b_0 \rightarrow b_3\}) = .2$ $\not\equiv$ $p(\{b_0 \rightarrow b_2\}) = .1$, $p(\{b_0 \rightarrow b_3\}) = .1$

Baye's

Dempster-Shafer Probability Combination

Heuristic 1 / Heuristic 2	$\{b_0 \rightarrow b_1\}$ (.8)	$\{b_0 \rightarrow b_2, b_0 \rightarrow b_3\}$ (.2)
$\{b_0 \rightarrow b_2\}$ (.6)	empty (.48)	$\{b_0 \rightarrow b_2\}$ (.12)
$\{b_0 \rightarrow b_1, b_0 \rightarrow b_3\}$ (.4)	$\{b_0 \rightarrow b_1\}$ (.32)	$\{b_0 \rightarrow b_3\}$ (.08)

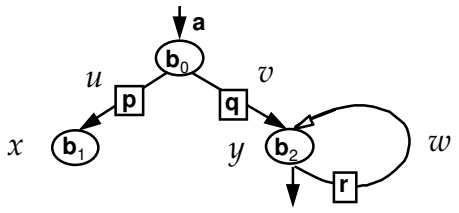
Heuristic 1⊕ Heuristic 2:

$p(\{b_0 \rightarrow b_1\}) = .32/.52 = .62$

$p(\{b_0 \rightarrow b_2\}) = .12/.52 = .23$

$p(\{b_0 \rightarrow b_3\}) = .08/.52 = .15$

Block and Branch Frequency



Given:

a: block_freq(b_0)

p: branch_prob(b_0—>b_1)

q: branch_prob(b_0—>b_2)

r: branch_prob(b_2—>b_2)

Find:

***u*: branch_freq(b_0—>b_1)**

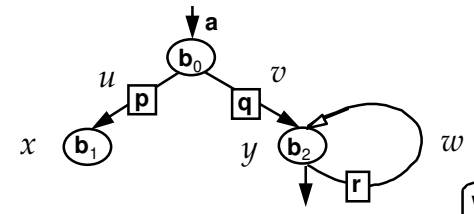
***v*: branch_freq(b_0—>b_2)**

***w*: branch_freq(b_2—>b_2)**

***x*: block_freq(b_1)**

***y*: block_freq(b_2)**

Straightforward Solution



O((n+e)^3).

System of equations

***u* = a * p**

***v* = a * q**

***w* = *y* * r**

x* = *u

***y* = *v* + w**

What if r = 1?

If r = 1, no solution

if r < 1, *u* = a * p

***v* = a * q**

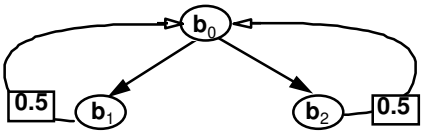
***w* = a * q * r / (1 - r)**

***x* = a * p**

***y* = a * q / (1 - r)**

Could r = 1?

- Spin loops
- Function call inside loop:
  for (;;) { ...; do_a_few_times_then_exit(); }
- More complicated cases:



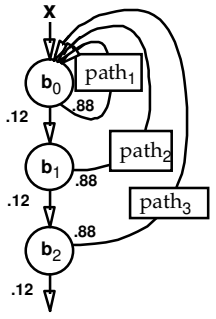
A Fast Algorithm for Reducible Graphs

- Reducible: Loop head dominates all nodes in the loop.
- Single loop head: first find frequency of the loop head, and then propagate to all nodes in the loop, as if a DAG
- Nested loop heads: go through the loop heads from inner-most outward, and propagate as if DAGs.
- For non-reducible graph the algorithm produces an approximate result.

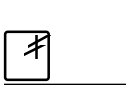
Complexity:

number of loop heads * number of edges: O (n * e)

Example: Single Loop Head

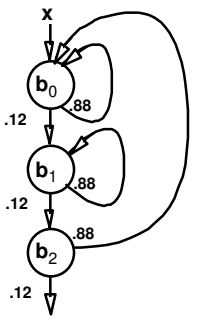


- Cyclic probability (cp): the probability that the loop head comes back to itself.
- $cp(b_0) = p(path_1) + p(path_2) + p(path_3)$
  $= .88 + .12*.88 + .12 * .12 * .88 = .9983$
- $block_freq(b_0) = x + block_freq(b_0) * cp(b_0)$
  $= 578.70 * x$
  $= x / (1 - cp(b_0))$
- If $cp(b_0) = 1$,
  Set $cp(b_0)$ = spin loop heuristic branch probability



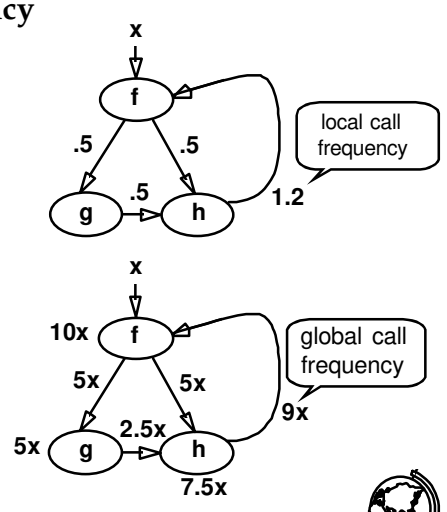
Example: Nested Loop Heads

- Inner loop, $cp(b_1) = .88$.
- Outer loop, $cp(b_0) = .88+.12*.12*.88 = .986$
- $block_freq(b_0) = x / (1 - cp(b_0)) = 71.4 * x$
- $block_freq(b_1)$
  $= branch_freq(b_0 \text{—>} b_1) / (1 - cp(b_1))$
  $= .12 * x / ((1 - cp(b_0)) * (1-cp(b_1))$
  $= 71.4 * x$
- $block_freq(b_2)$ =
  $= block_freq(b_1) * branch_prob(b_1 \text{—>} b_2)$
  $= 71.4 * x * .12 = 8.57 * x$



Call Frequency

- Construct call graph with local call frequencies
- Propagate local call frequencies to determine global call frequencies
- Call graphs usually not reducible:
- Extend propagation algorithm to general graphs



Global Branch Frequency

- Function f is called C times
- Branch b_0->b_1 takes K times inside f
- Global branch frequency of b_0->b_1 is C * K

Evaluate Static Profiles

Static		Dynamic		Top%	Unweighted	Weighted
p1	100	p1	1000	10%	1.00	1.00
p3	95	p2	200	20%	0.50	0.83
p2	70	p3	150	30%	1.00	1.00
p5	60	p4	100	40%	....	
p4	50	p5	40	50%	...	
.....						
p10	15	p10	0	100%	....	

Weighted Top 20% matching score = 1000/1200 = 0.83

Performance Result (Weighted Top 20% Matching Scores)

Profiles	SPECint92 Benchmarks	UNIX Commands
Local block	82%	79%
Local edge	69%	73%
Function invocation	85%	82%
Function call	72%	76%
Global block	77%	78%
Global edge	78%	79%

Summary

- Heuristic branch probability = heuristic hit rate
- Dempster-Shafer algorithm combines multiple heuristic probabilities
- Fast branch frequency propagation algorithm
- Fast function frequency propagation algorithm
- Experiment shows that static profiles are highly accurate

Future Work

- More heuristics
- Heuristics for multi-way branches
- Heuristics use global information
- Focused experiment to tune heuristic hit rates
- Indirect function calls