2D-Profiling

Detecting Input-Dependent Branches with a Single Input Data Set

Hyesoon Kim
M. Aater Suleman
Onur Mutlu
Yale N. Patt

HPS Research Group
The University of Texas at Austin

Motivation

- Profile-guided code optimization has become essential for achieving good performance.
  - Run-time behavior ≅ profile-time behavior: Good!
  - Run-time behavior ≠ profile-time behavior: Bad!

Motivation

- Profiling with one input set is not enough!
  - Because a program can show different behavior with different input data sets
  - Example: Performance of predicated execution is highly dependent on the input data set
  - Because some branches behave differently with different input sets

Input-dependent Branches

- Definition
  - A branch is input-dependent if its misprediction rate differs by more than some Δ over different input data sets.

	Inp. 1	Inp. 2	Inp.1 – Inp. 2
Misprediction rate of Br. X	30%	29%	1%
Misprediction rate of Br. Y	30%	5%	25%

Input-dependent branch

- Input-dependent br. ≠ hard-to-predict br.

An Example Input-dependent Branch

- Example from Gap (SPEC2K):
  Type checking branch

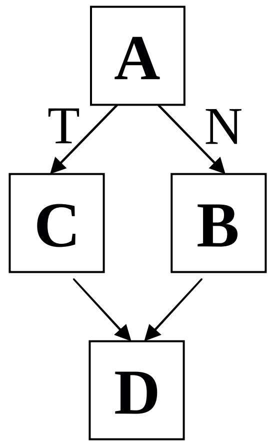
```
TypHandle Sum (A,B)

  If ((type(A) == INT) && (type(B) == INT)) {        //input-dependent br
        Result = A + B;
        Return Result;
  }
  Return SUM(A, B);
```

- Train input set: A&B are integers 90% of the time
  - misprediction rate: 10%
- Reference input set: A&B are integers 42% of the time
  - misprediction rate: 30% (30%-10%)>Δ

Predicated Execution

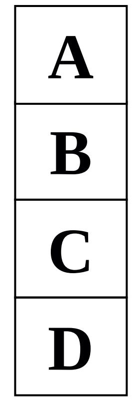
```
if (cond) {
    b = 0;
}
else {
    b = 1;
}
```

(normal branch code)



A
```
p1 = (cond)
branch p1, TARGET
```
B
```
mov b, 1
jmp JOIN
```
C
```
TARGET:
    mov b, 0
```

(predicated code)



A
```
p1 = (cond)
```
B
```
(!p1) mov  b, 1
```
C
```
(p1) mov  b, 0
```

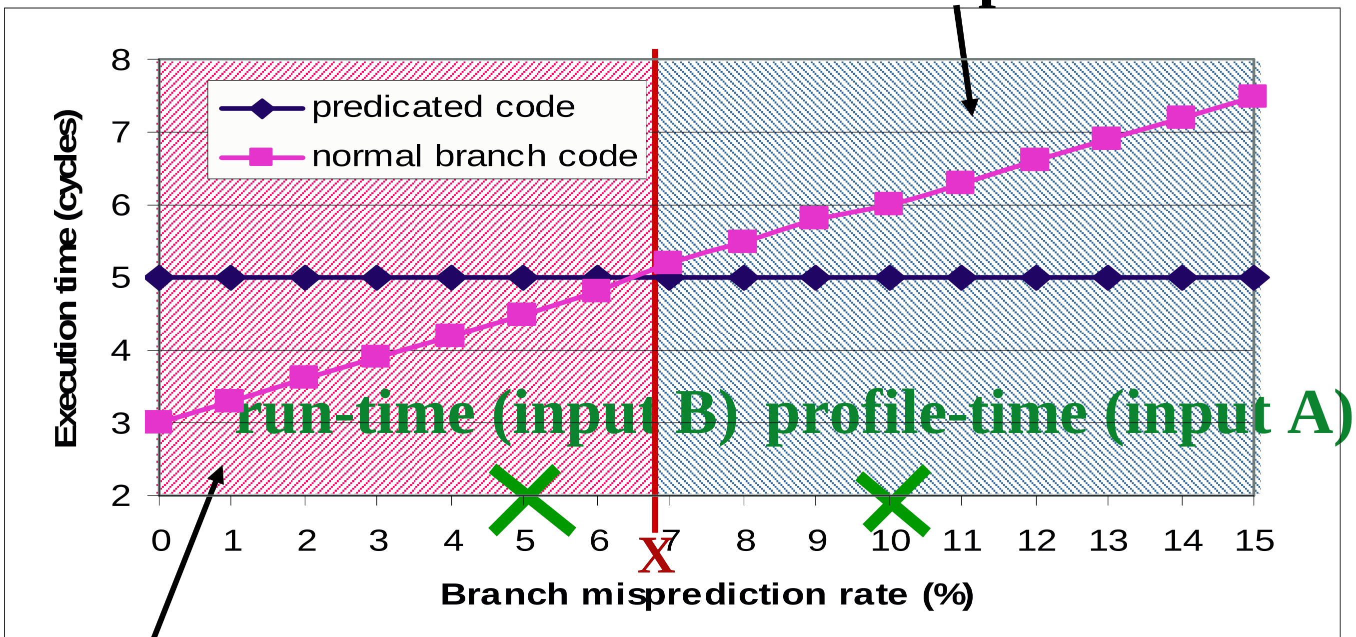
Eliminate hard-to-predict branches

but fetch blocks B and C all the time

UT ECE

Predicated Code Performance vs. Branch Misprediction Rate

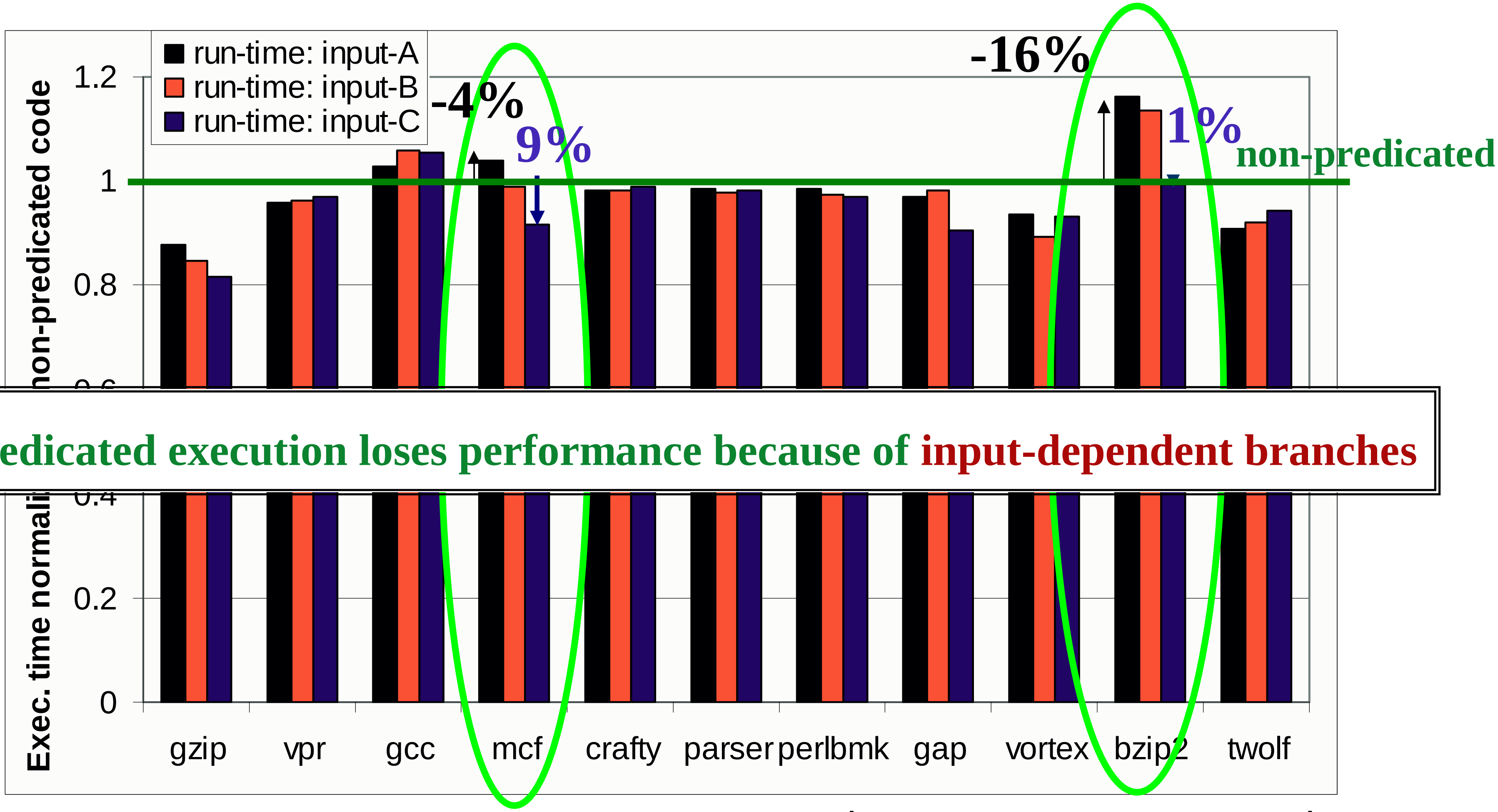
Predicated code performs better



Normal branch code performs better

- Converting a branch to predicated code could hurt performance if run-time misprediction rate is lower than profile-time misprediction rate

Predicated Code Performance vs. Input Set



Predicated execution loses performance because of input-dependent branches

Measured on an Itanium-II machine

If We Know a Branch is Input-Dependent

- May not convert it to predicated code.
- May convert it to a wish branch.

  [Kim et al. Micro'05]
- May not perform other compiler optimizations or may perform them less aggressively.
  - Hot-path/trace/superblock-based optimizations

    [Fisher'81, Pettis'90, Hwu'93, Merten'99]

Our Goal

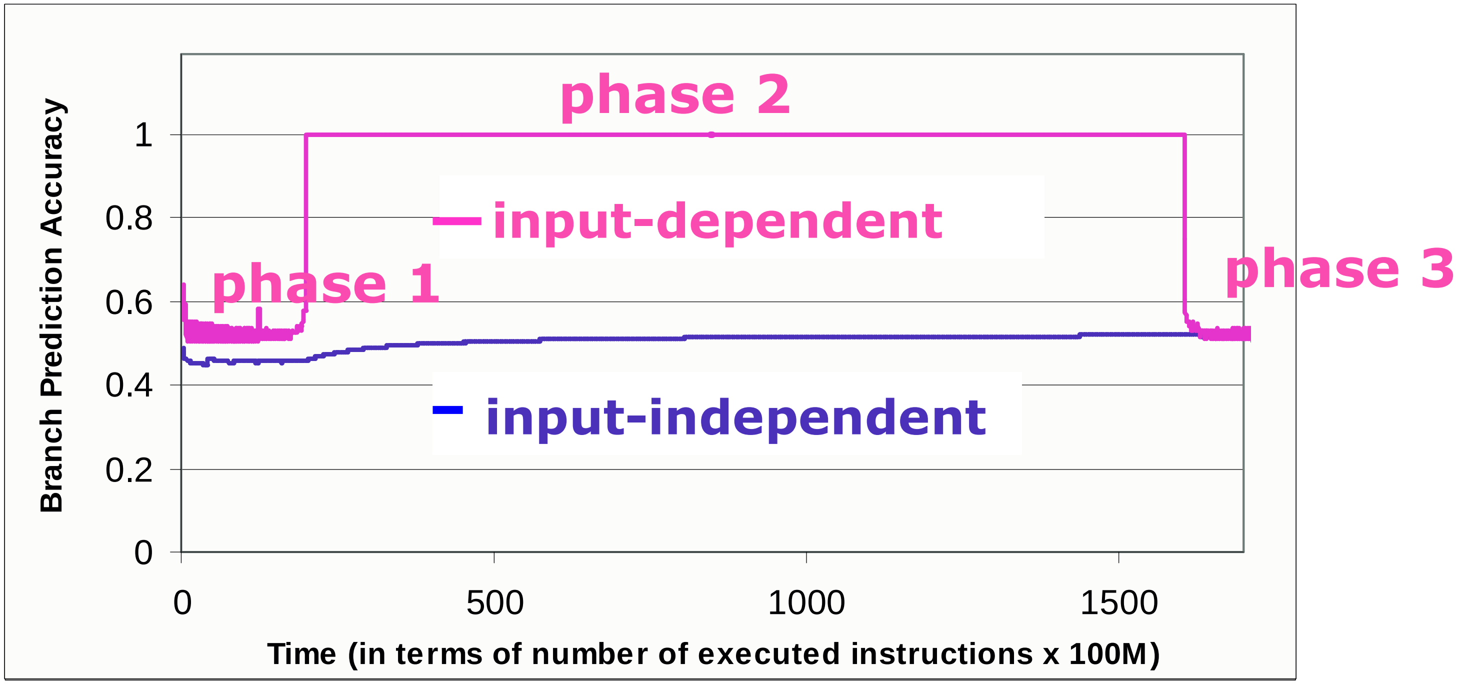
Identify input-dependent branches by using a single input set for profiling

Talk Outline

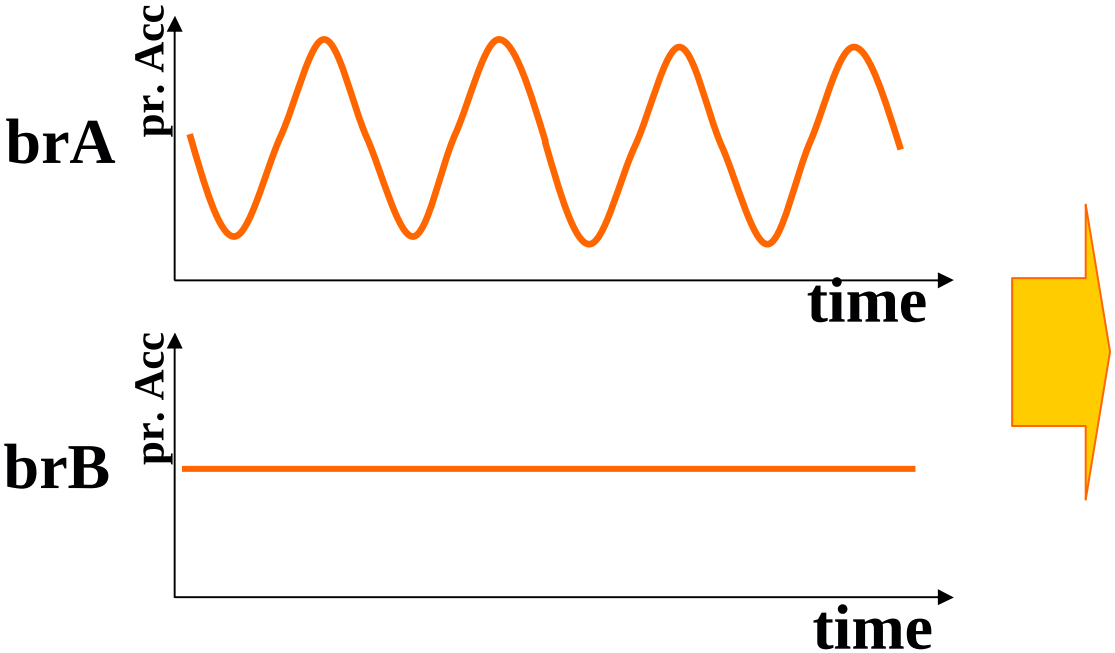
- Motivation
- 2D-profiling Mechanism
- Experimental Results
- Conclusion

Key Insight of 2D-profiling

Phase behavior in prediction accuracy is a good indicator of input dependence



Traditional Profiling



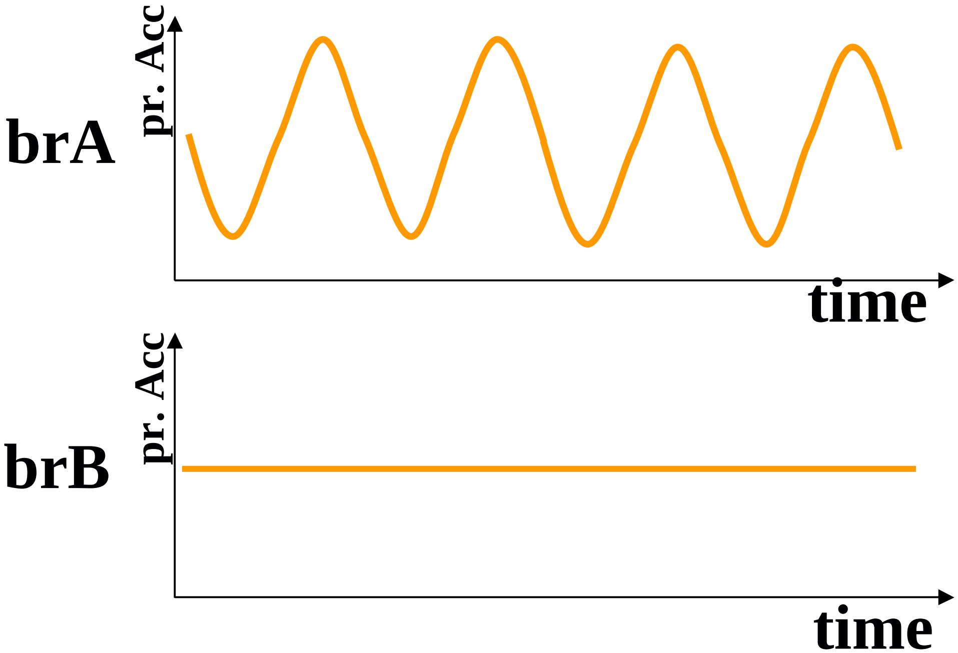
MEAN pr.Acc(brA)

MEAN pr.Acc(brB)

MEAN pr.Acc(brA) ≅ MEAN pr.Acc(brB)

behavior of brA ≅ behavior of brB

2D-profiling



MEAN pr.Acc(brA)

STD pr.Acc(brA)

MEAN pr.Acc(brB)

STD pr.Acc(brB)

MEAN pr.Acc(brA) ≅ MEAN pr.Acc(brB)

STD pr.Acc(brA) ≠ STD pr.Acc(brB)

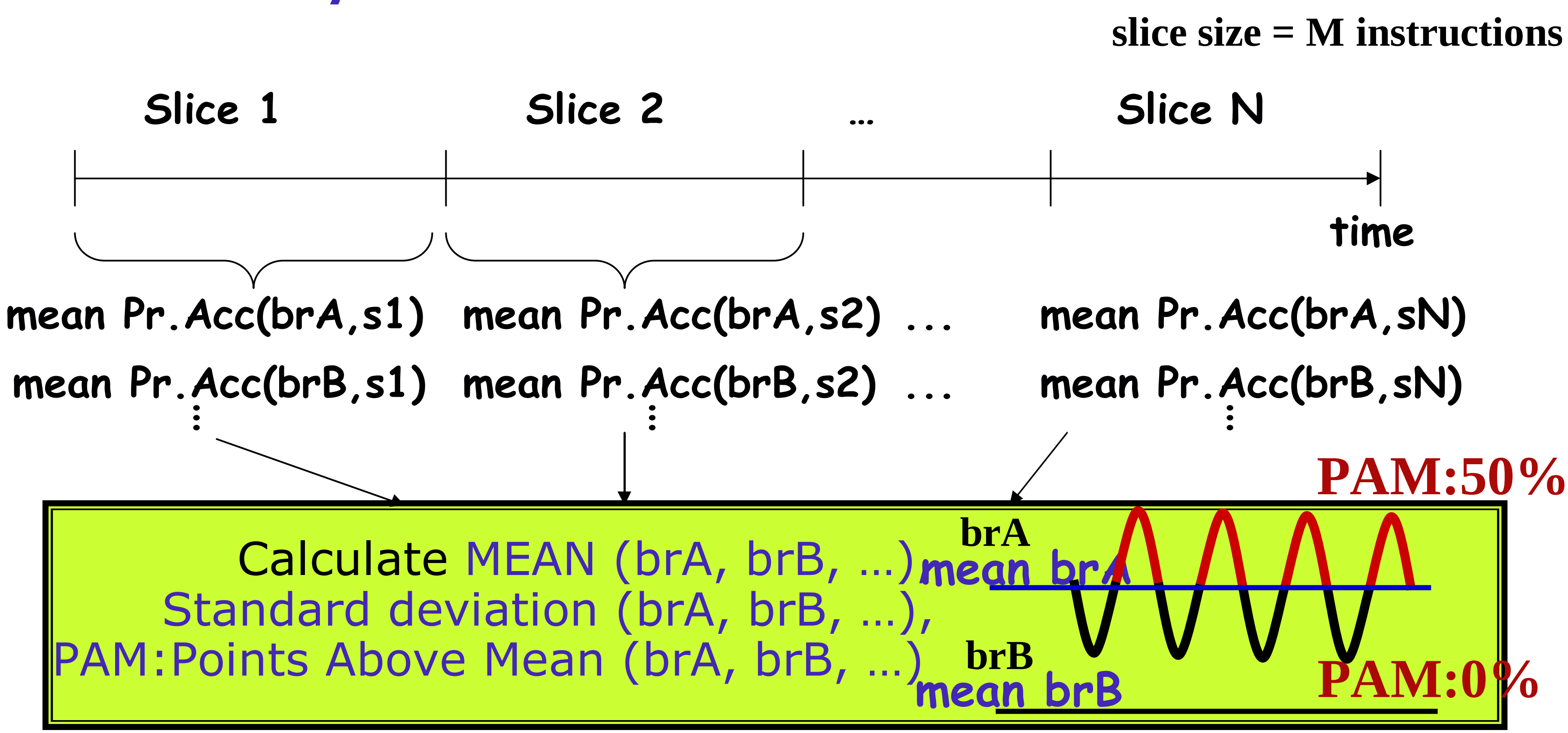
behavior of brA ≠ behavior of brB

A: input-dependent br, B: input-independent br

UT ECE

2D-profiling Mechanism

- The profiler collects branch prediction accuracy information for **every static branch** **over time**



Input-dependence Tests

- STD&PAM-test: Identify branches that have **large variations** in accuracy over time (phase behavior)
  - STD-test (STD > threshold): Identify branches that have large variations in the prediction accuracy over time
  - PAM-test (PAM > threshold): Filter out branches that pass STD-test due to a few outlier samples
- MEAN&PAM-test: Identify branches that have **low prediction accuracy** and **some time-variation** in accuracy
  - MEAN-test (MEAN < threshold): Identify branches that have low prediction accuracy
  - PAM-test (PAM > threshold): Identify branches that have some variation in the prediction accuracy over time
- A branch is classified as input-dependent if it passes either STD&PAM-test or MEAN&PAM-test

Talk Outline

- Motivation
- 2D-profiling Mechanism
- Experimental Results
- Conclusion

Experimental Methodology

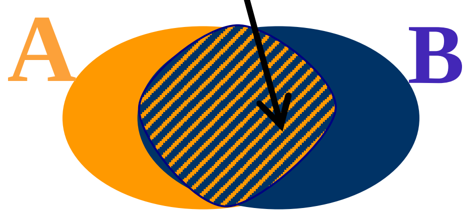
- Profiler: PIN-binary instrumentation tool
- Benchmarks: SPEC 2K INT
- Input sets
  - Profiler: Train input set
  - Input-dependent Branches: Reference input set and train/other extra input sets
- Input-dependent branch: misprediction rate of the branch changes more than Δ = 5% when input data set changes
  - Different Δ are examined in our TechReport [reference 11].
- Branch predictors
  - Profiler: 4KB Gshare, Machine: 4KB Gshare
  - Profiler: 4KB Gshare, Machine: 16KB Perceptron (in paper)

Evaluation Metrics

- Coverage and Accuracy for input-dependent branches

Correctly Predicted Input-dependent br.

Predicted Input-dependent br. (2D-profiler)

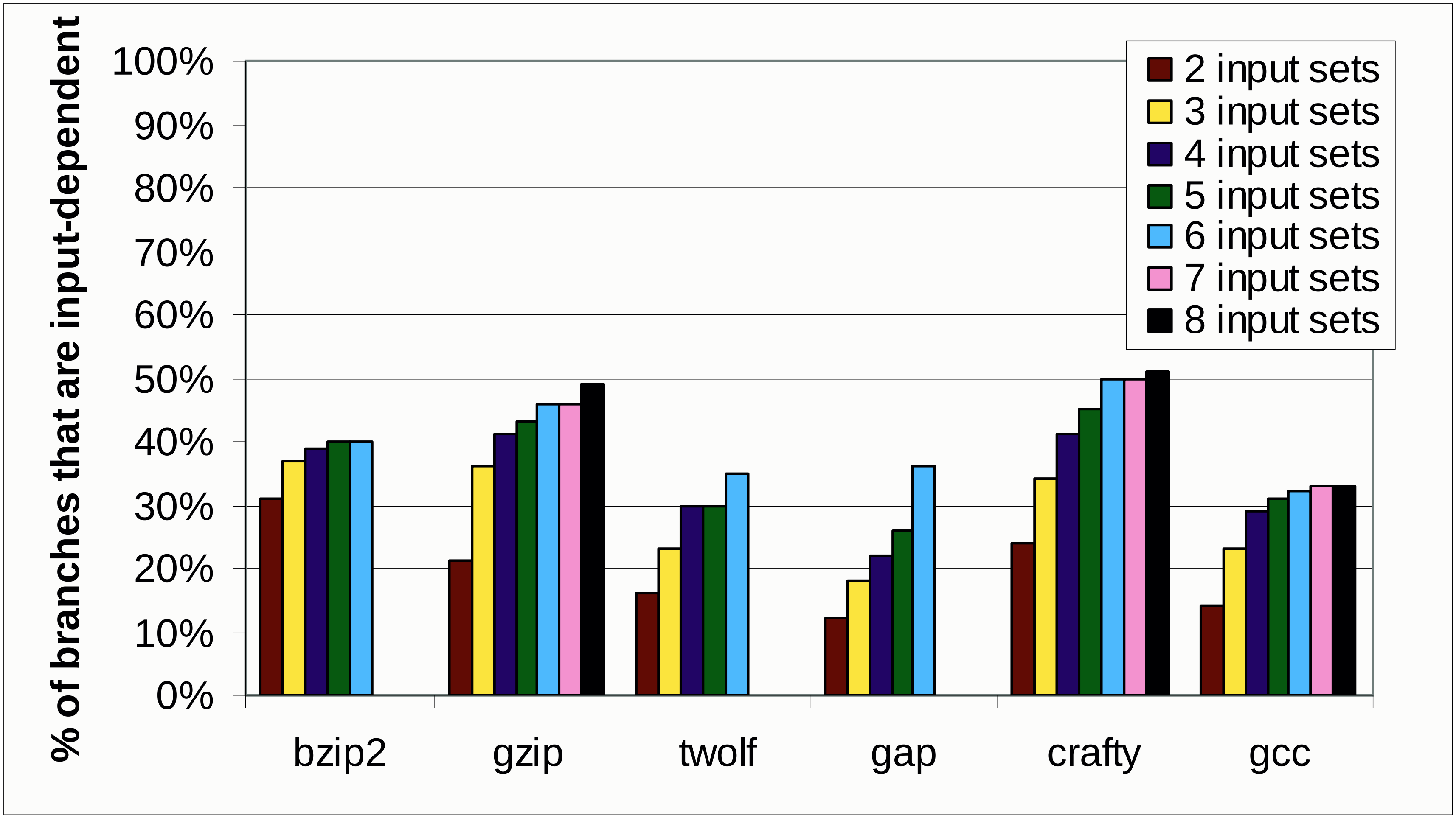


Actual Input-dependent br.

$$\mathrm{COV} = \frac{\mathrm{A} \cap \mathrm{B}}{\mathrm{A}} = \frac{\text{Correctly Predicted}}{\text{Actual Input-dependent}}$$

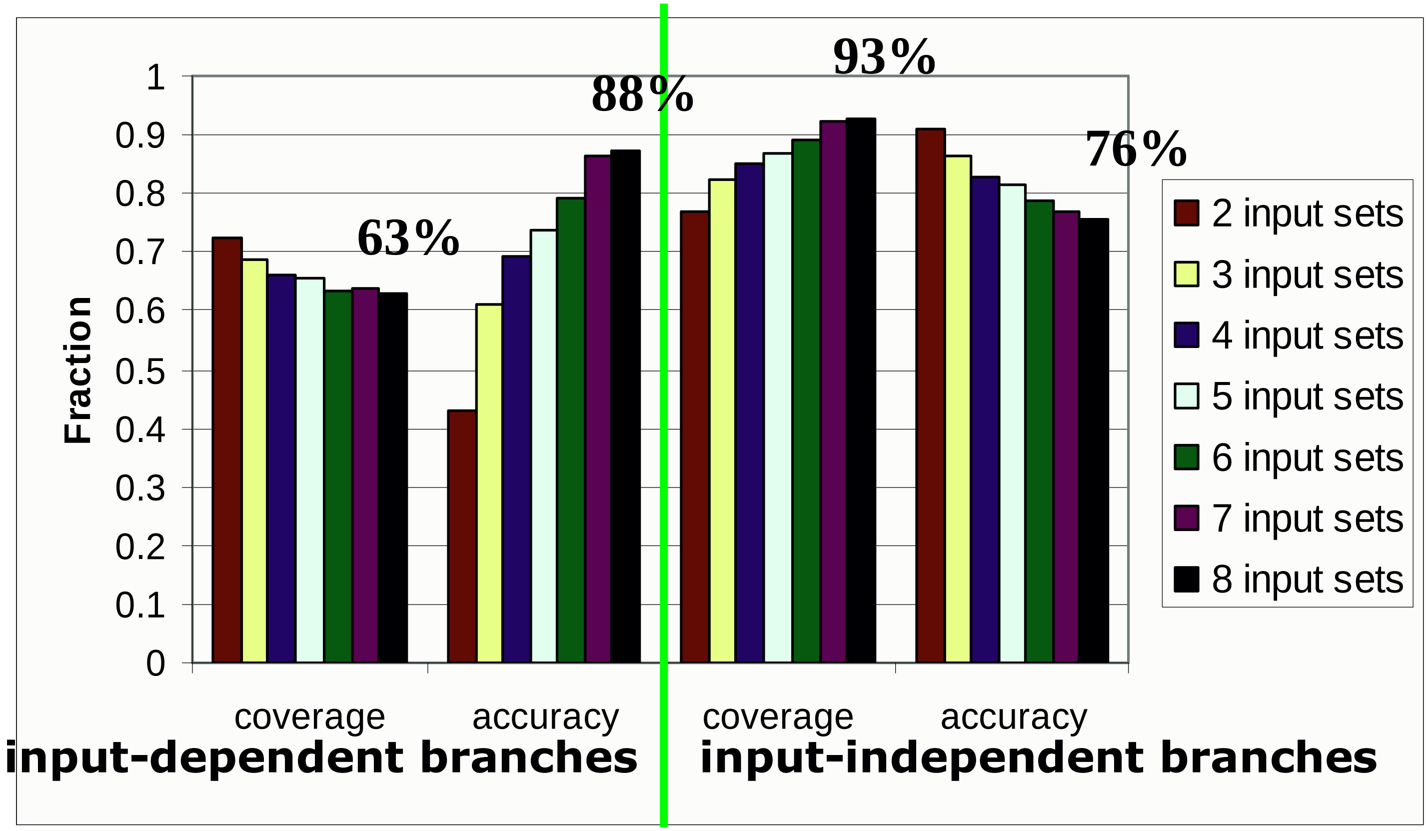
$$ACC = \frac{\mathrm{A} \cap \mathrm{B}}{\mathrm{B}} = \frac{\text{Correctly Predicted}}{\text{Predicted Input-dependent}}$$

Input-dependent Branches



UT ECE

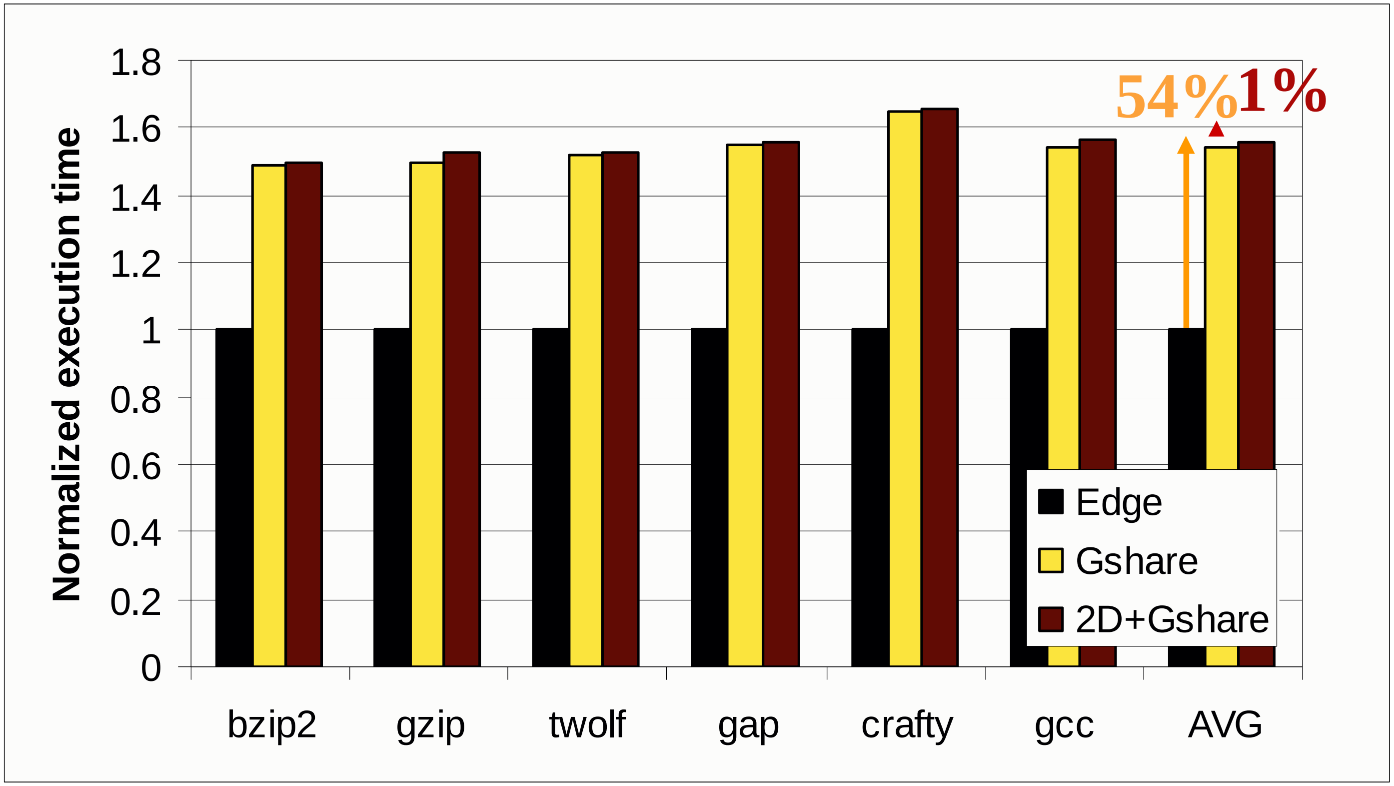
2D-profiling Results



Phase behavior and input-dependence are strongly correlated!

UT ECE

The Cost of 2D-profiling?



- 2D-profiling adds only 1% overhead over modeling the branch predictor in software
  - Using a H/W branch predictor [Conte'96]

Conclusion

- 2D-profiling is a new profiling technique to find input-dependent characteristics by using a single input data set for profiling
- 2D-profiling uses time-varying information instead of just average data
- Phase behavior in prediction accuracy in a profile run → input-dependent
- 2D-profiling accurately identifies input-dependent branches with very little overhead (1% more than modeling the branch predictor in the profiler)
- Applications of 2D-profiling are an open research topic
  - Better predicated code/wish branch generation algorithms
  - Detecting other input-dependent program characteristics