Trace Based Switching For A Tightly Coupled Heterogeneous Core

Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, Scott Mahlke

Micro-46
December 2013

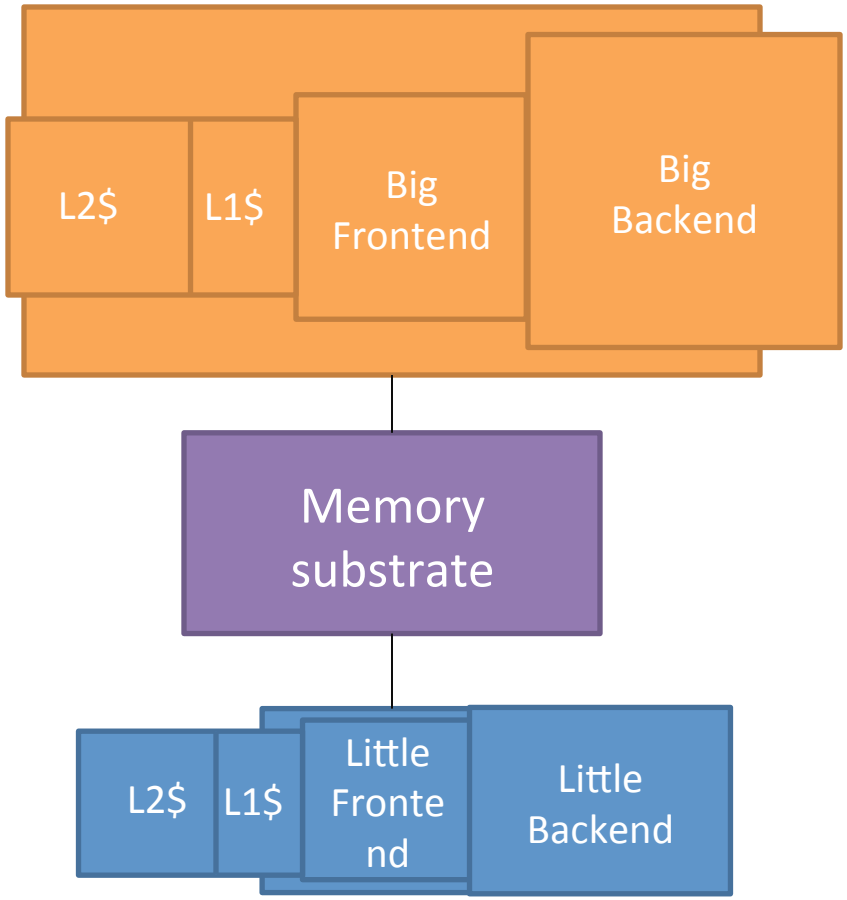
CCC compilers creating custom processors

University of Michigan
Electrical Engineering and Computer Science

Outline

- Fine-grained heterogeneity
- Don’t React – Predict!
- Trace-Based Switching Controller
- Results
- Conclusion

Single-ISA Heterogeneity



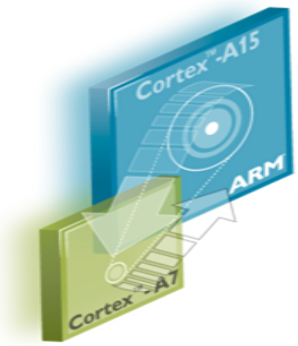
Out-of-order

- ✓ Fast
  - ✓ Good for high performance phases
- ✗ Power hungry structures
  - ✗ Wastes energy on low performance phases

Energy savings with minimal performance loss

In-order

- ✗ Smaller/Slower
  - ✗ 1/2 the performance of big
- ✓ Simpler, low power structures
  - ✓ 3X more energy efficient



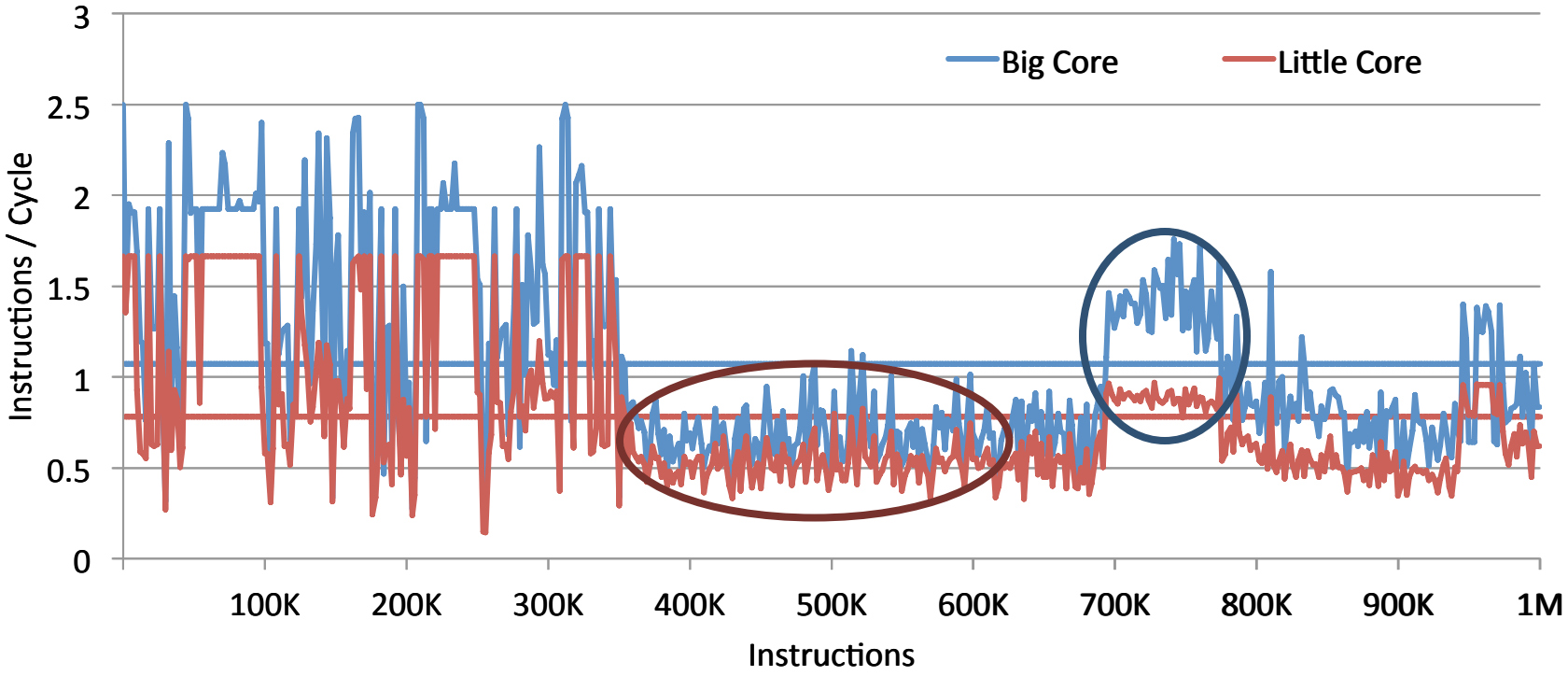
Traditional Heterogeneous Architectures

ARM's big.LITTLE

Transfer overhead = ~20K Cycles
Minimum switching interval = ~1M instructions

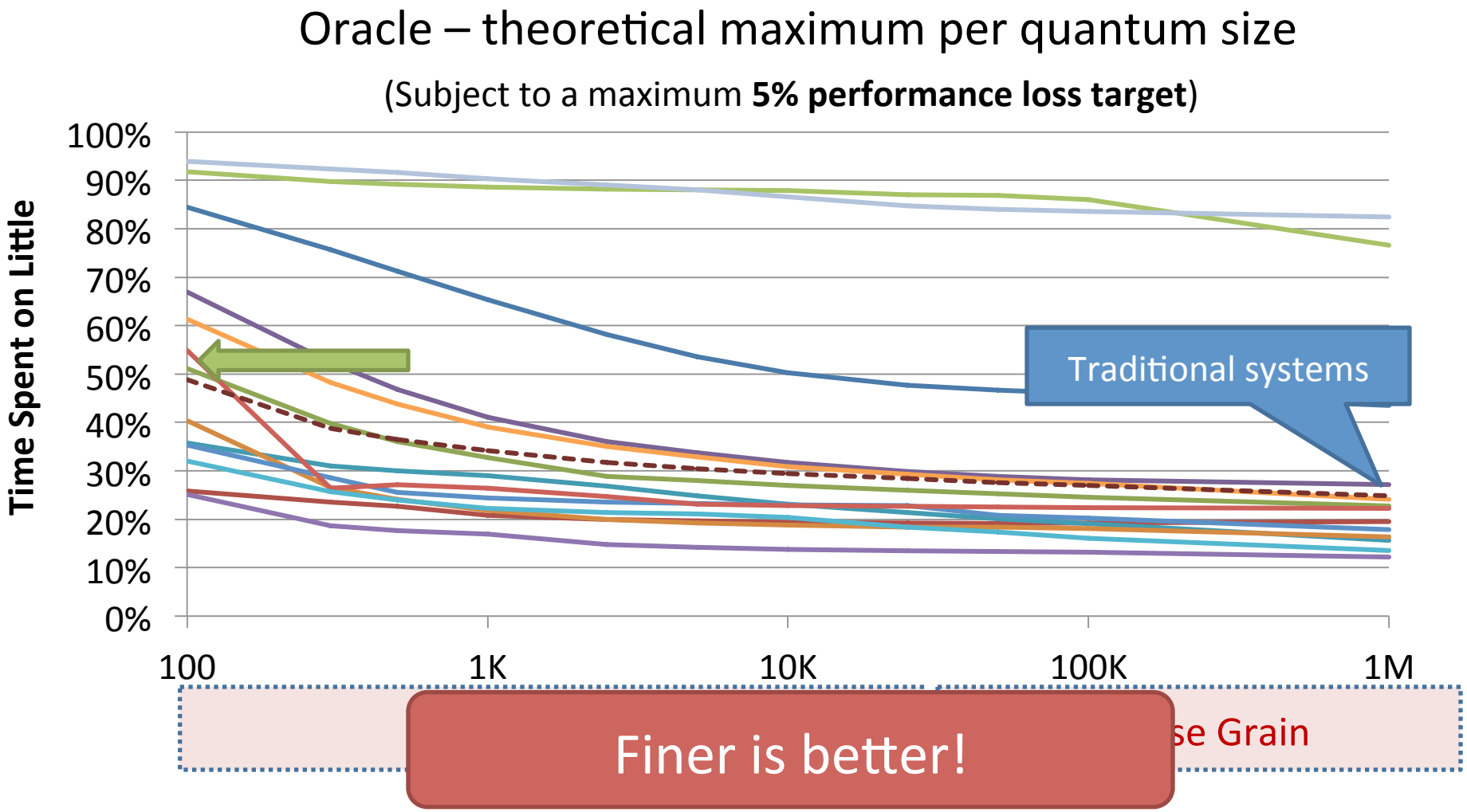
What about low performance phases at finer-granularity?

Performance Change In GCC

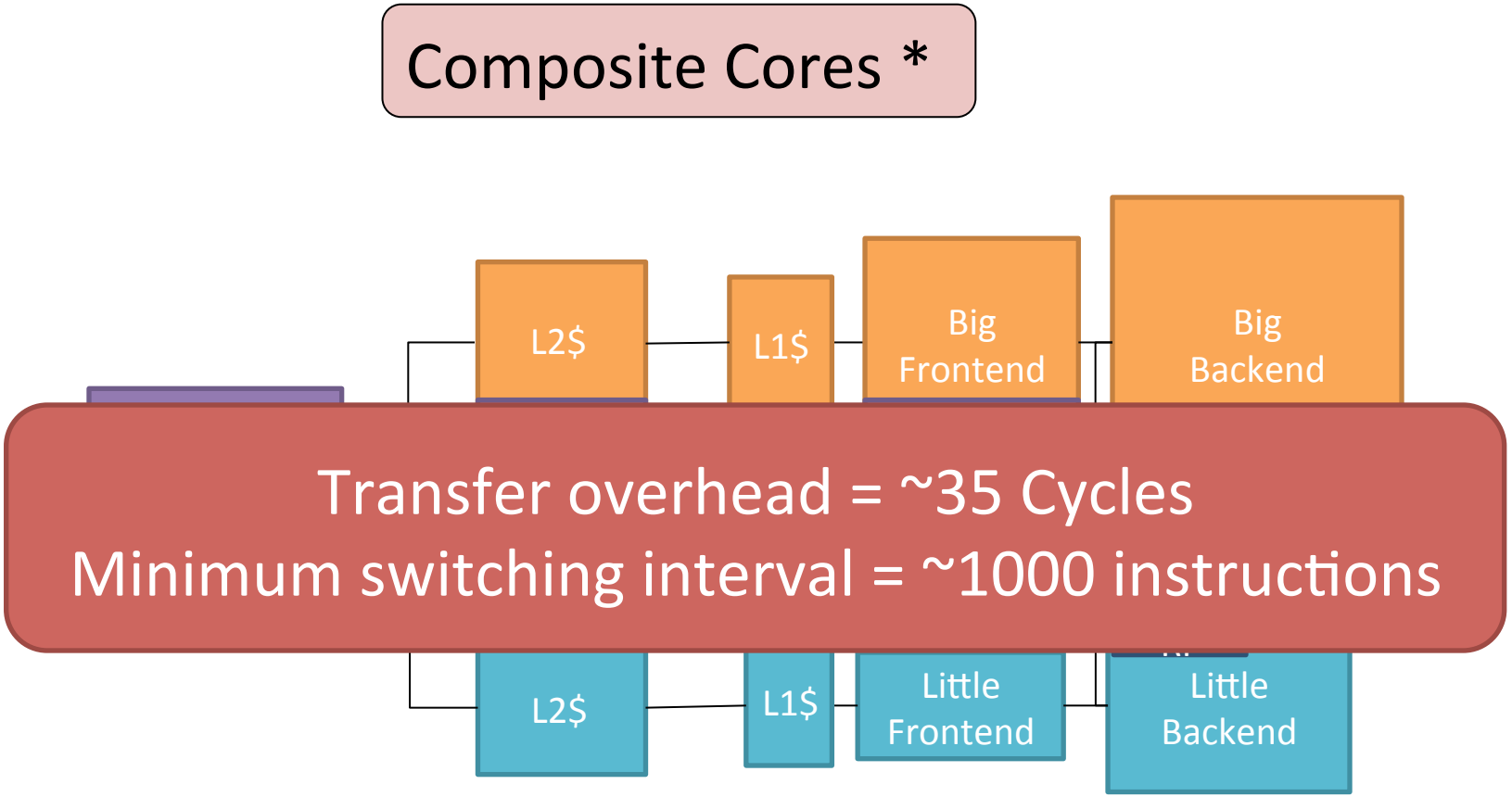


Huge performance changes within a coarse quantum!

Fine-grain Heterogeneity Has Potential



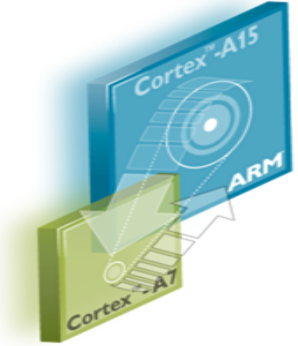
Fine-grained Heterogeneous Architectures



**Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012*

Outline

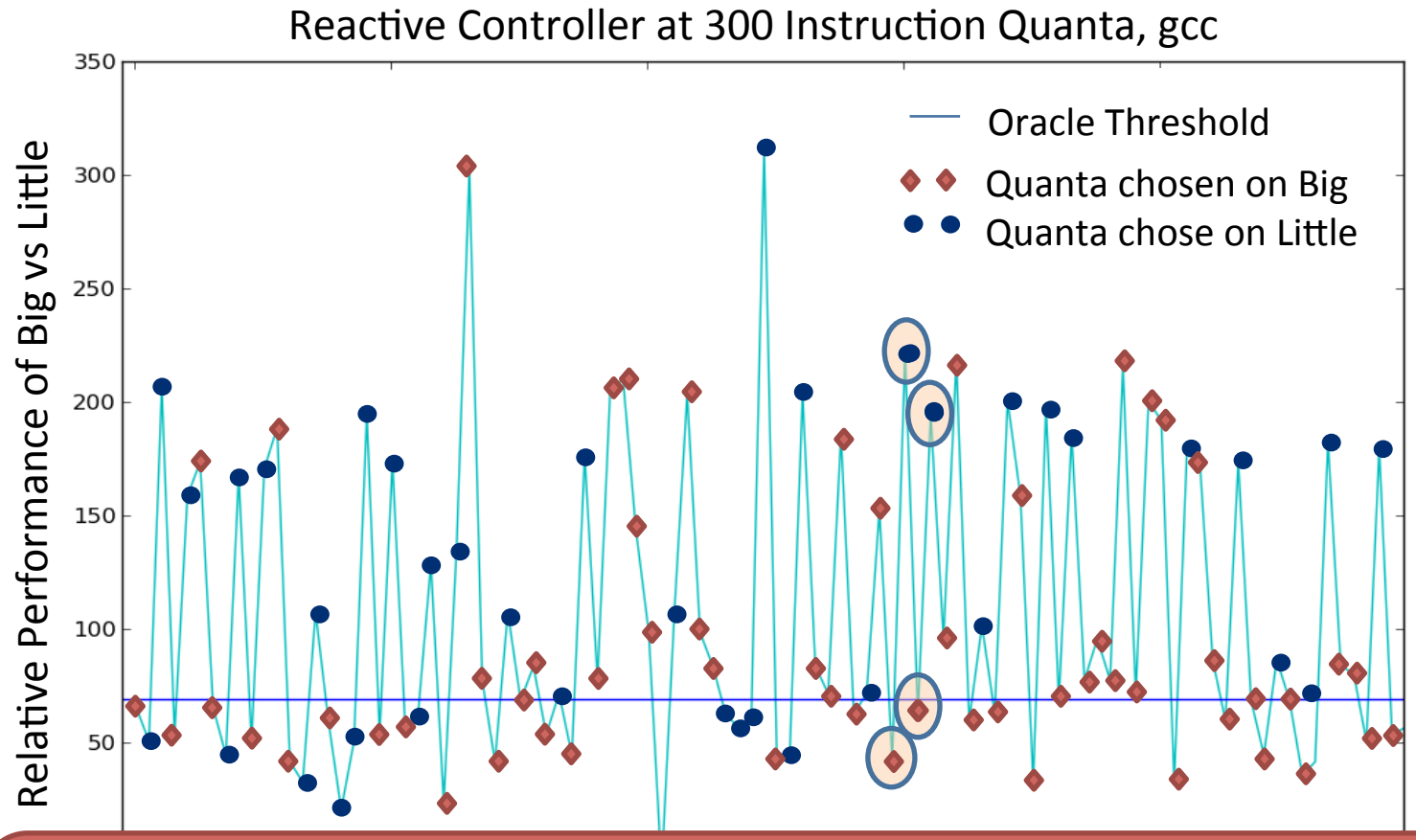
- Fine-grained heterogeneity
- Don’t React – Predict!
- Trace-Based Switching Controller
- Results
- Conclusion



Traditional Switching Controller

Assumption:
A quantum's behavior is indicated by the recent past

Reactive Follies at Fine-Granularity



Pick Big

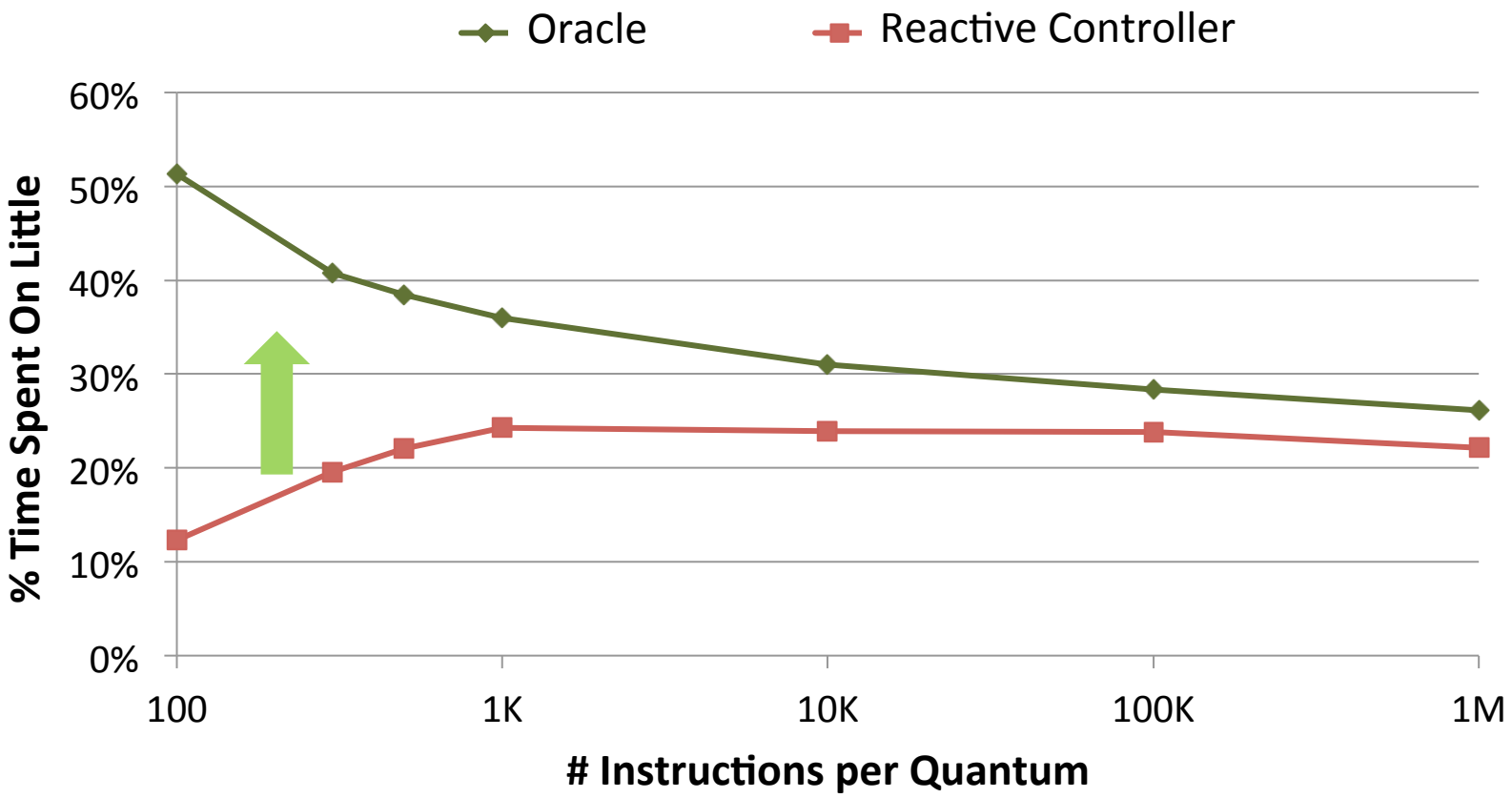
Pick Little

Incorrectly picked little → lost performance, on 29% of quanta

Incorrectly picked big → lost opportunities for energy savings on 26% of quanta

Fine-granularity “Reacts” Badly

Don’t React – Predict!



Outline

- Fine-grained heterogeneity
- Don’t React – Predict!
- Trace-Based Switching Controller
- Results
- Conclusion

History Based Prediction

Observation

Super-Trace

Code repeats (loops, functions)

Exploit this inherent repeatable nature of code

Behavior repeats if placed in program context

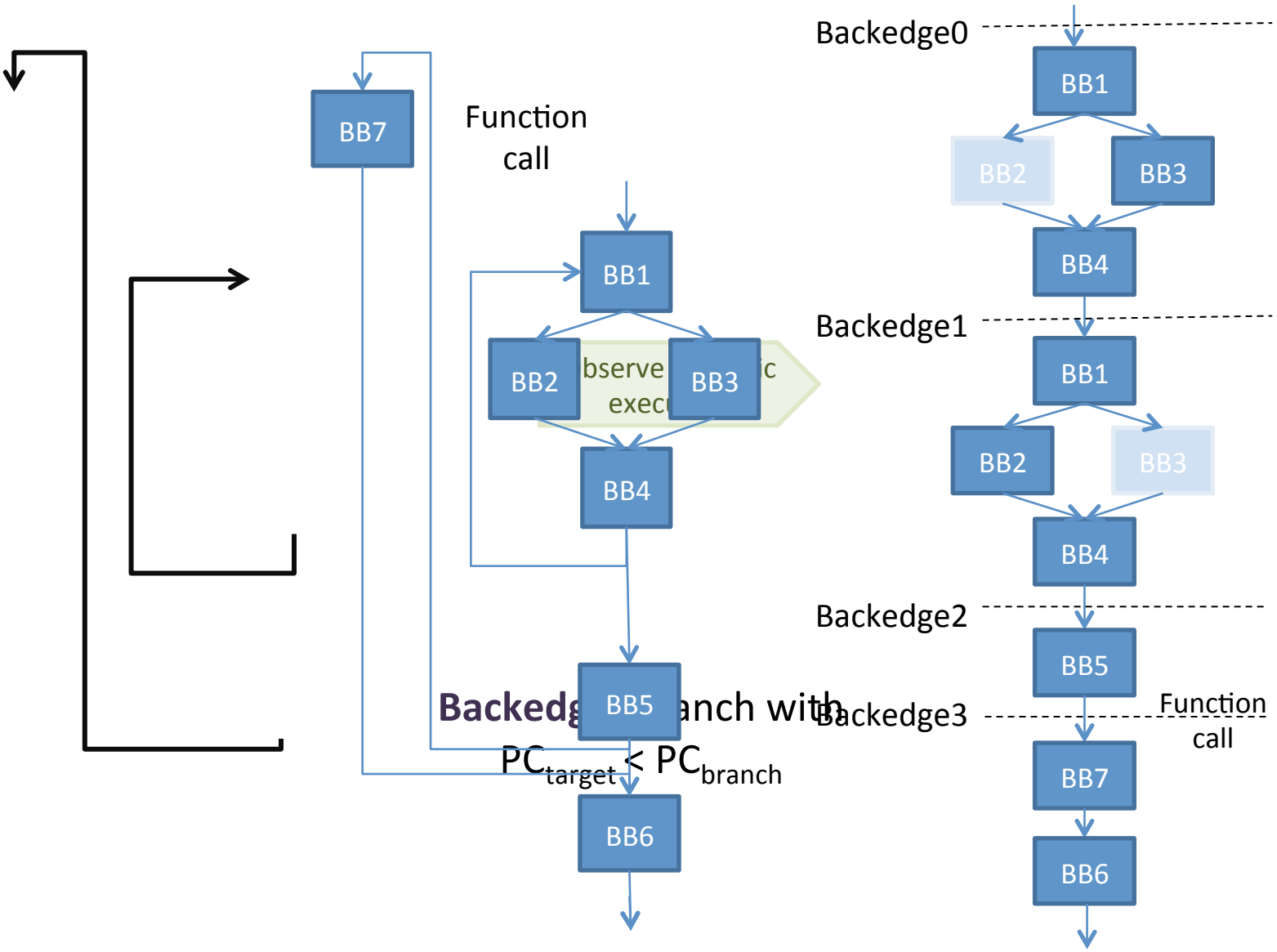
Objective

1. Predict oncoming Super-traces
2. Predict its backend preference

Our Assumption

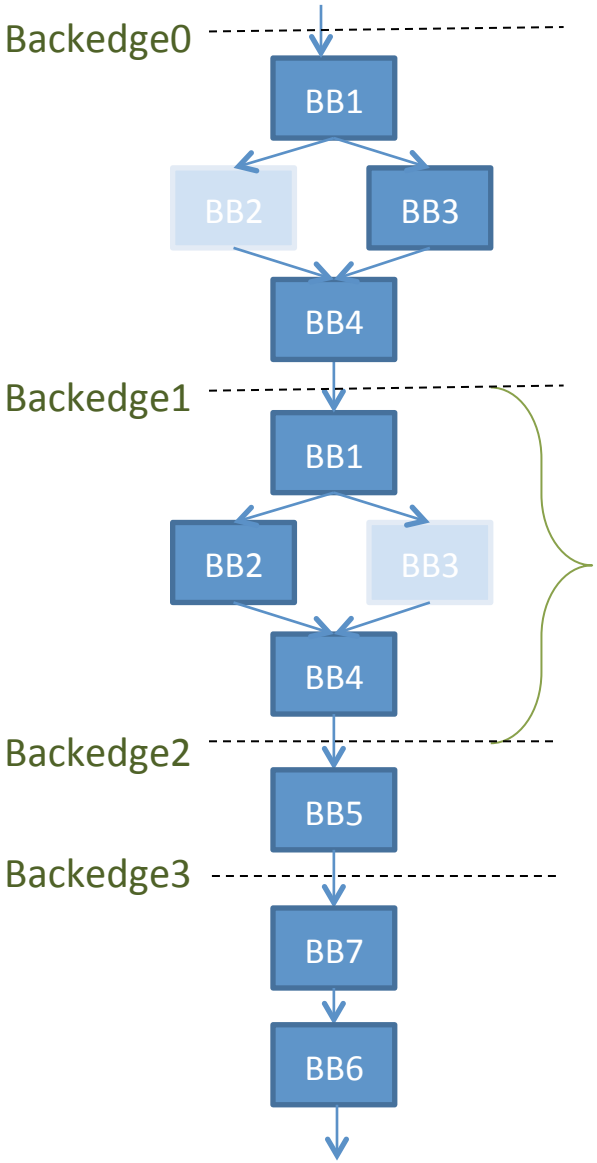
A Super-Trace's behavior history follows a pattern

Super-trace Construction



Super-trace Construction

Trace: Block of code defined at backedge boundaries



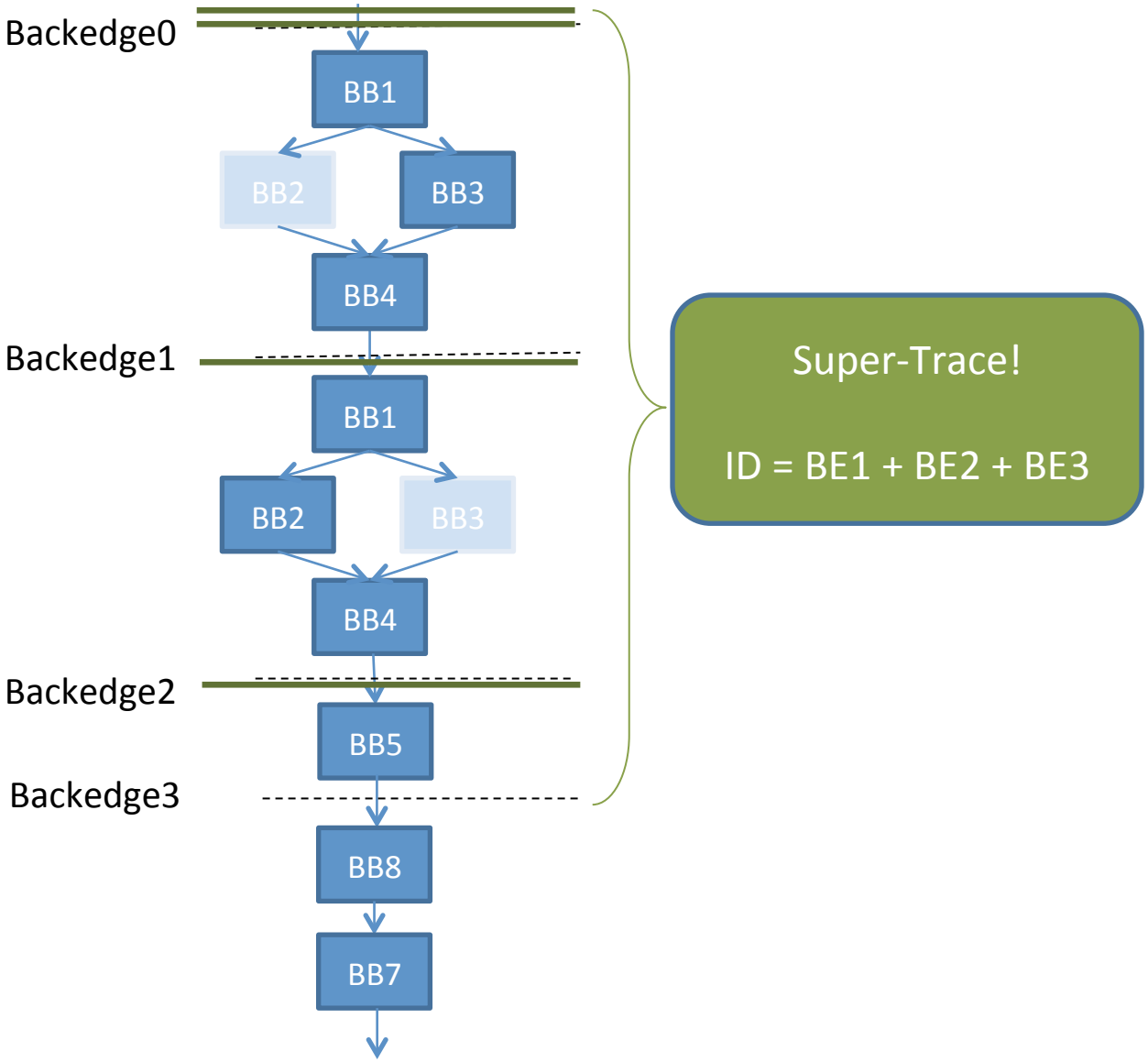
~ 50 instructions…

Performance highly dependent on context

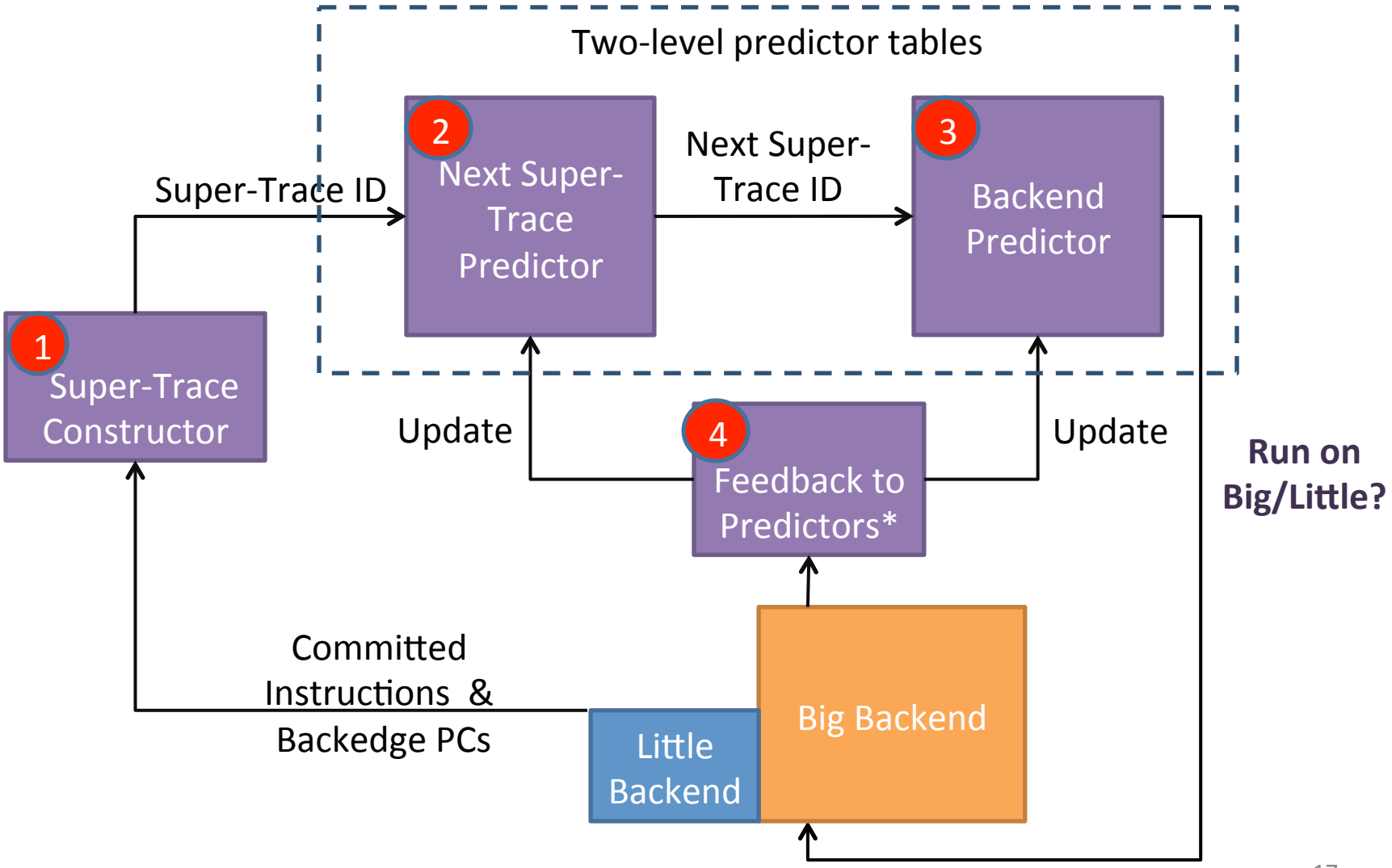
Combine traces to meet minimum length requirement (300 instructions)

Super-trace Construction

Count instructions between backedges



Trace-Based Controller Overview

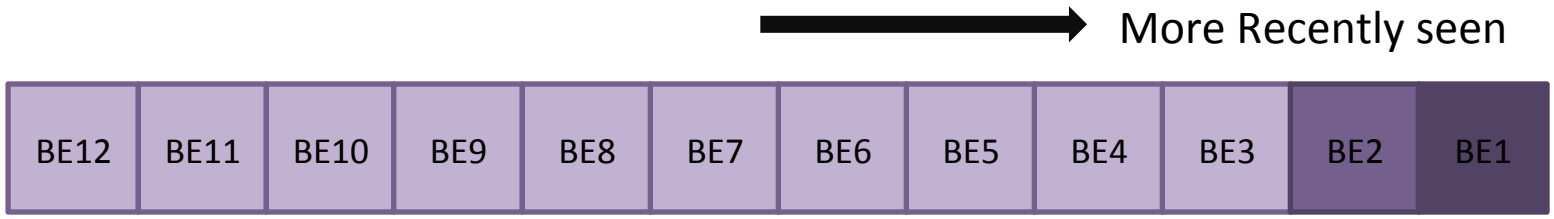


**Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012*

1 Super-Trace Constructor

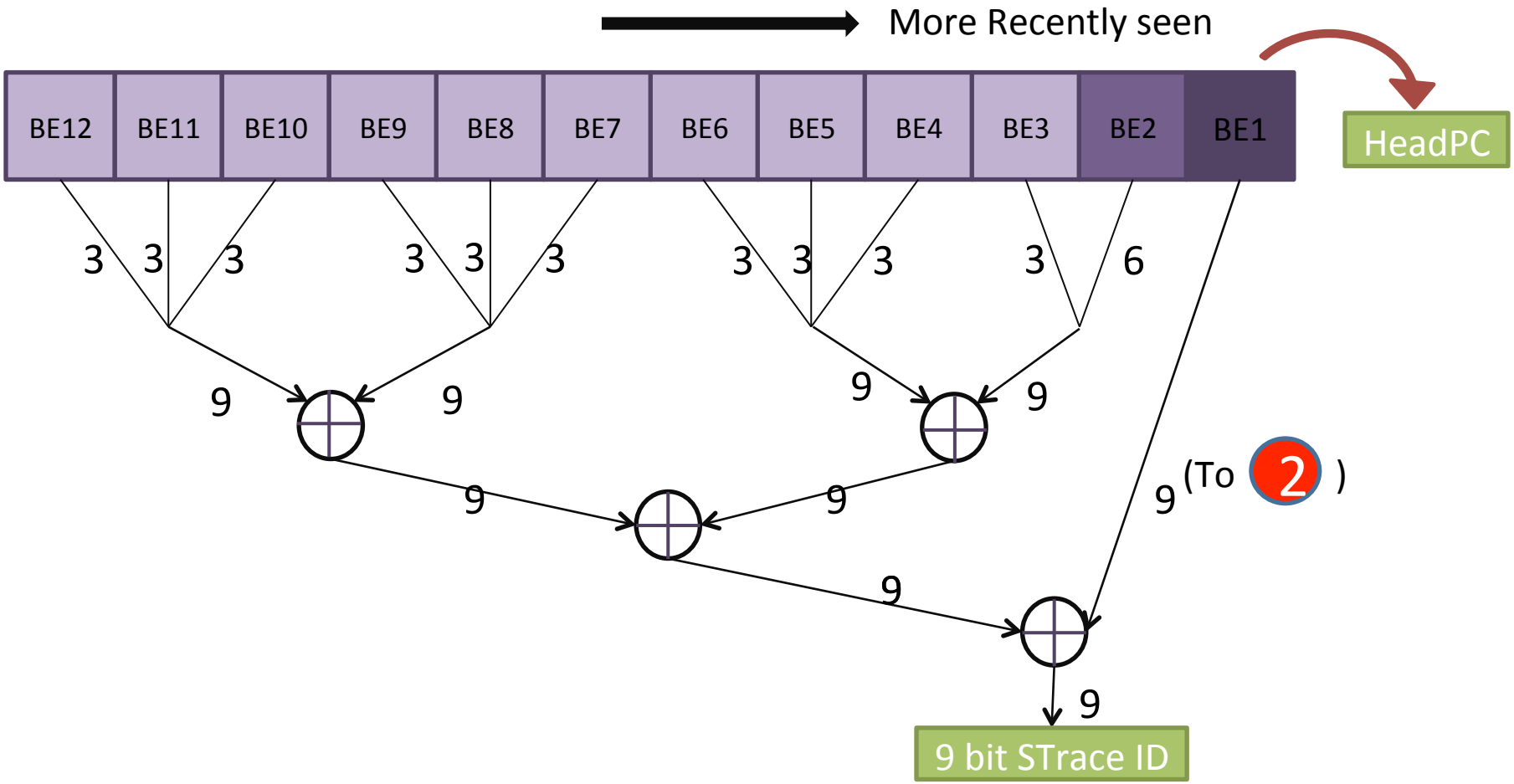
1. Check for minimum instruction length constraint

2. Hash backedge PCs to provide a super-trace ID

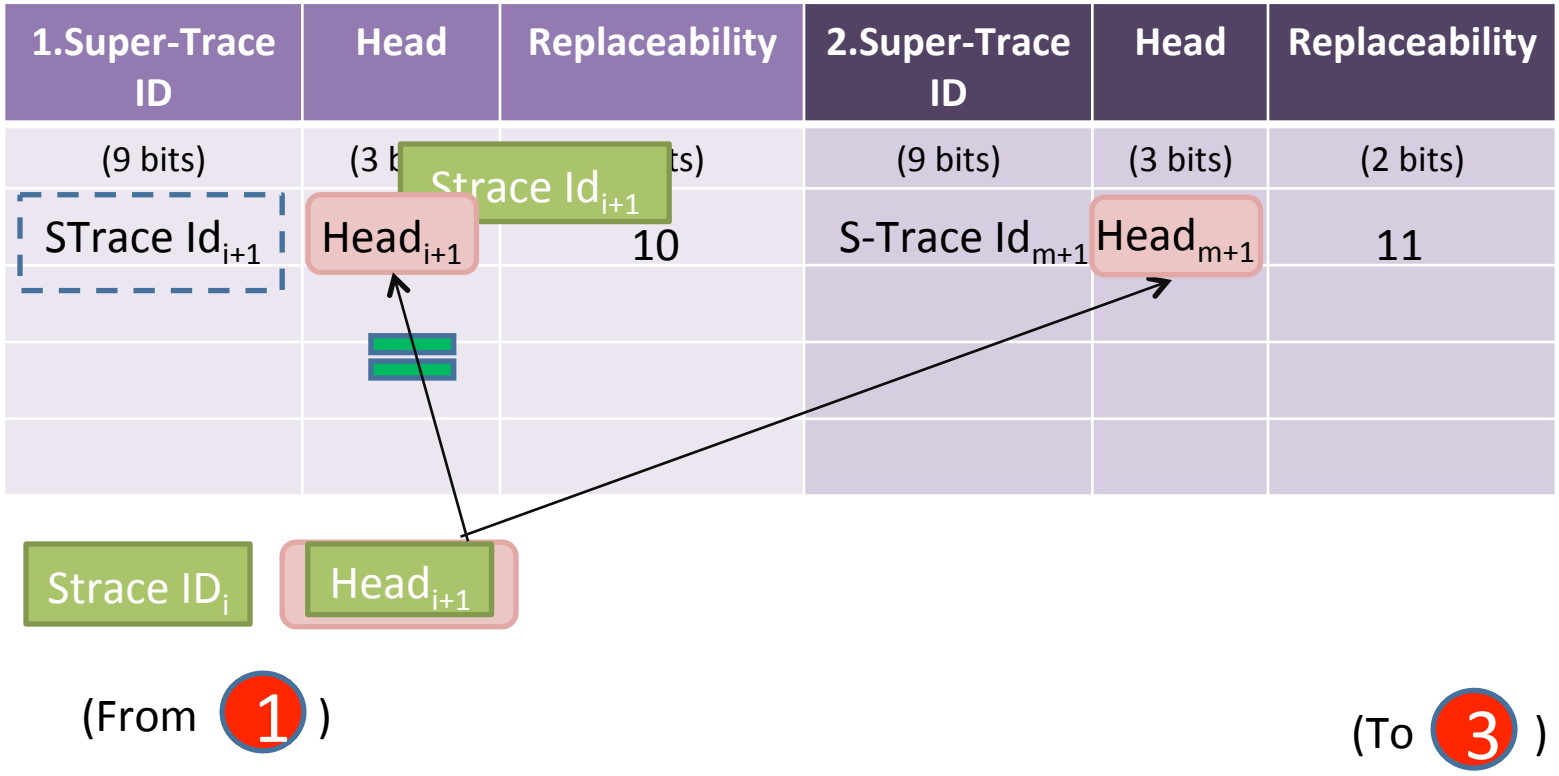


1 Super-Trace Constructor

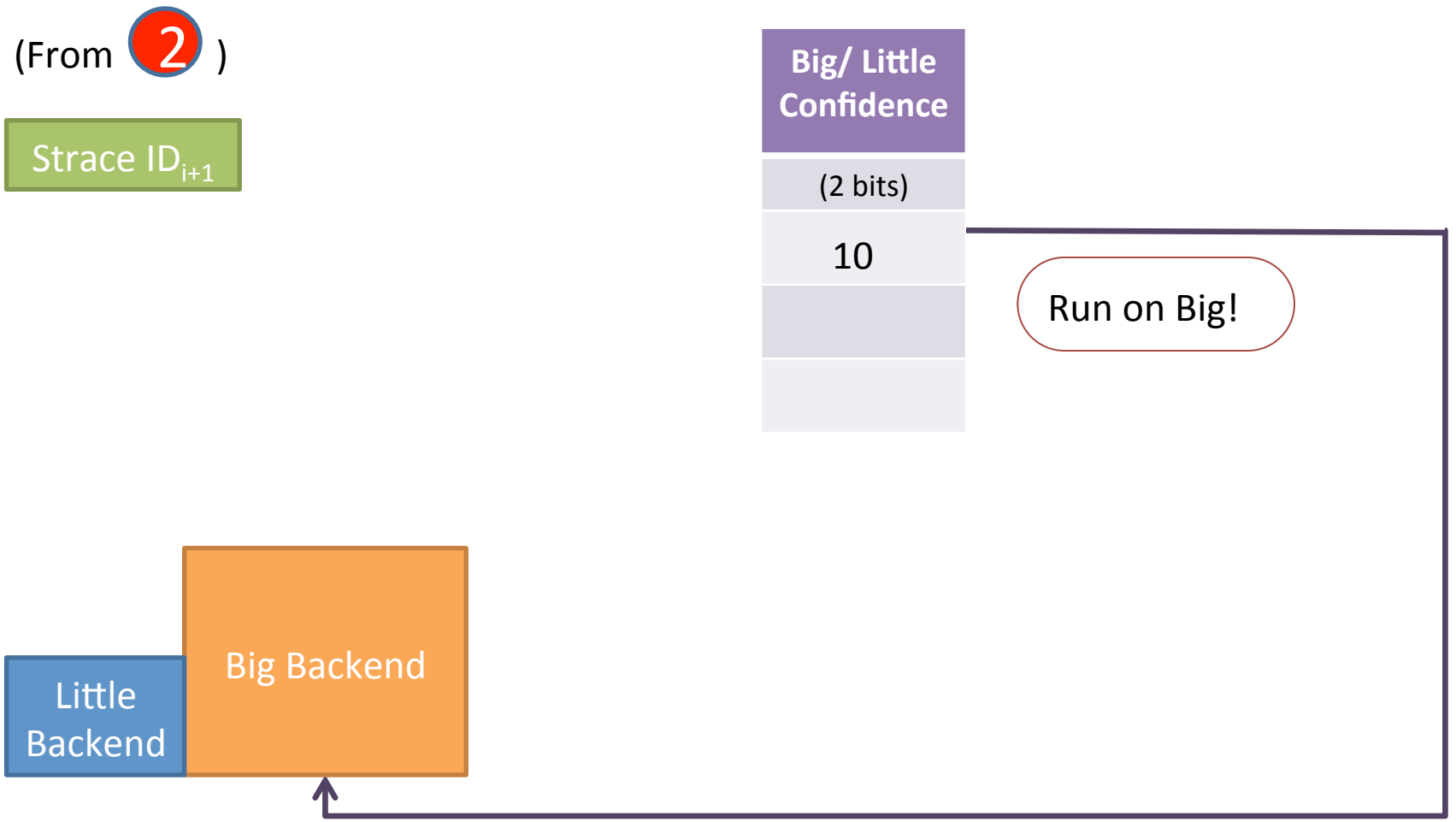
Hash backedge PCs to provide a super-trace ID



2 Next Super-Trace Predictor



3 Next Backend Predictor



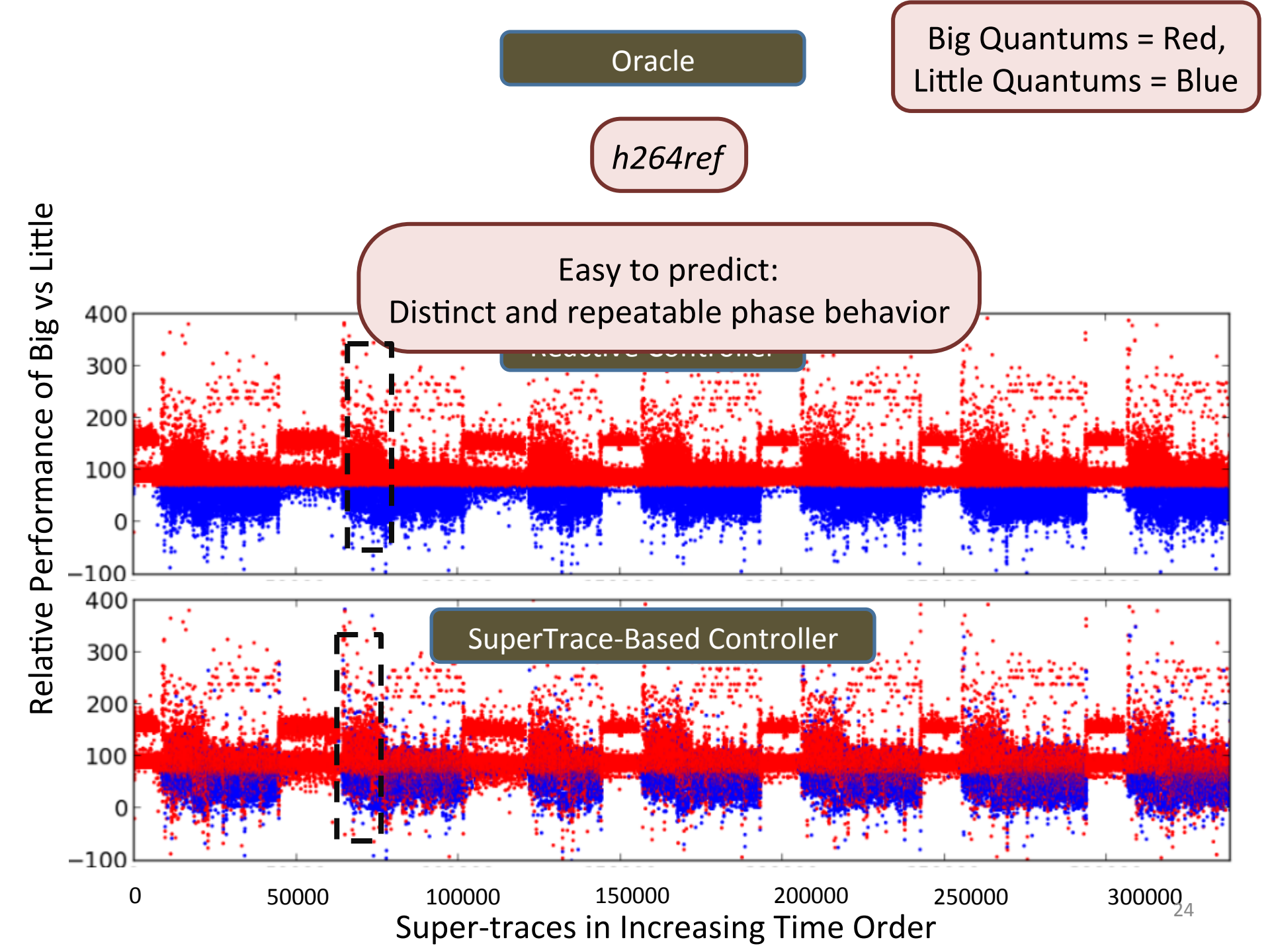
Evaluation

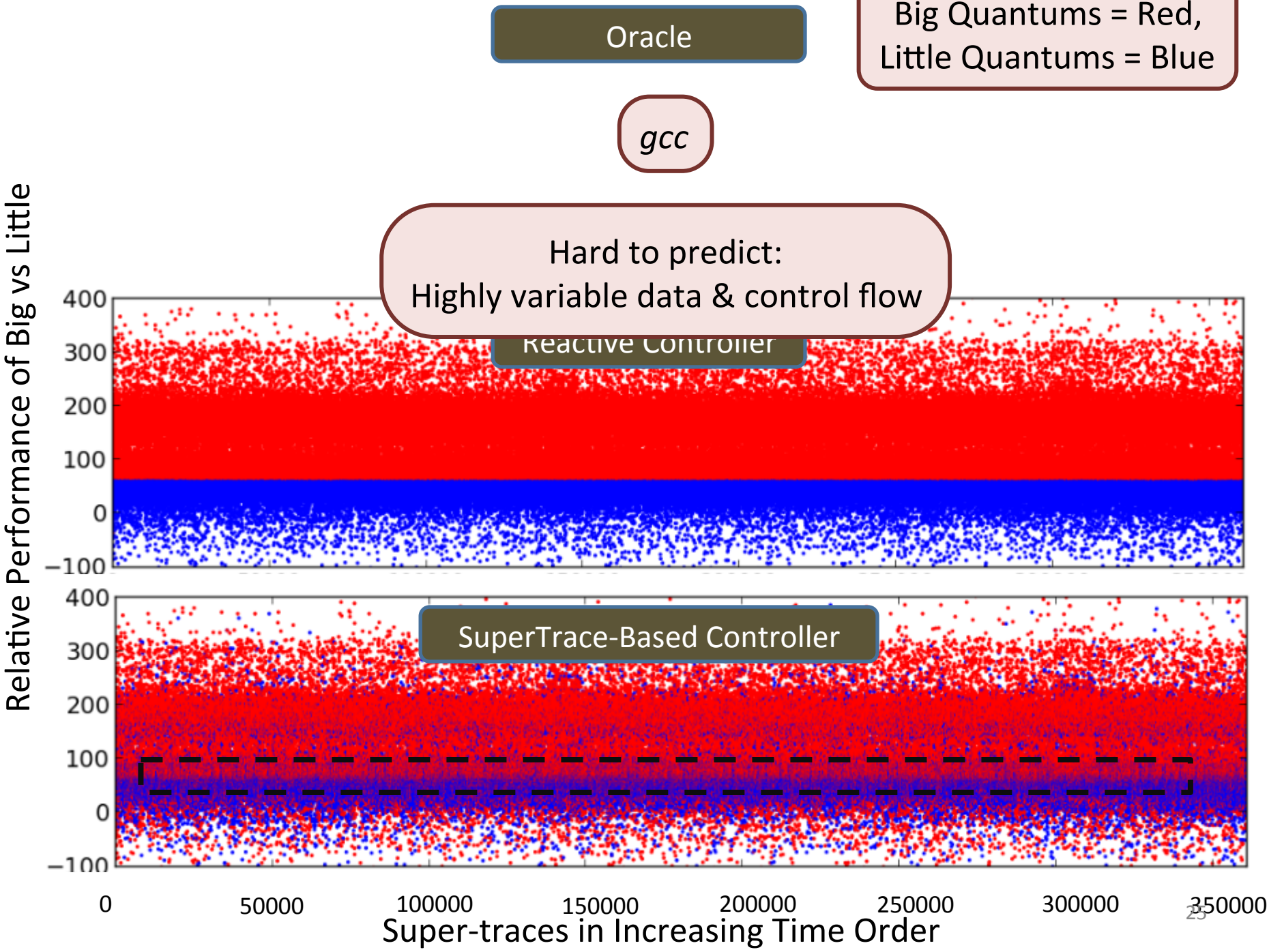
Architectural Feature	Parameters
Big Core	3 wide O3 @ 1.2GHz 12 stage pipeline 128 ROB Entries 128 entry register file
Little Core	2 wide InOrder @ 1.2GHz 8 stage pipeline 32 entry register file
Memory System	32 KB L1 i/d cache, 2 cycle access 1MB L2 cache, 15 cycle access 1GB Main Mem, 80 cycle access

Simulator	Gem5, Full system
Energy Model	McPAT
Benchmarks	SPEC 2k6, compiled for Alpha ISA Fast Forward 2 billion instructions, simulate 100 million instructions

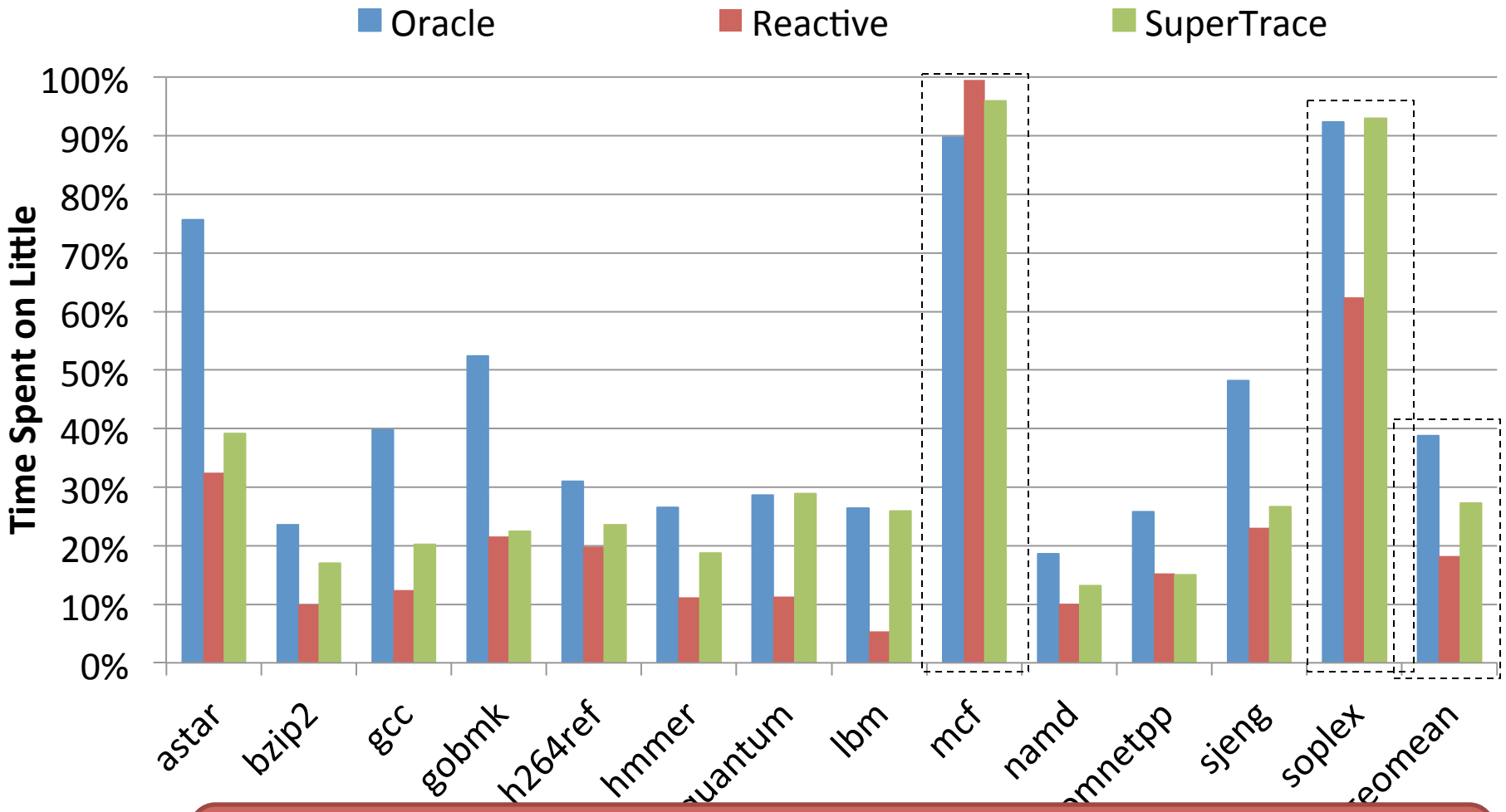
Outline

- Fine-grained heterogeneity
- Don’t React – Predict!
- Trace-Based Switching Controller
- Results
- Conclusion


Oracle
Big Quantums = Red,
Little Quantums = Blue
h264ref
Easy to predict:
Distinct and repeatable phase behavior
SuperTrace-Based Controller
Relative Performance of Big vs Little
400
300
200
100
0
−100
0
50000
100000
150000
200000
250000
300000
Super-traces in Increasing Time Order

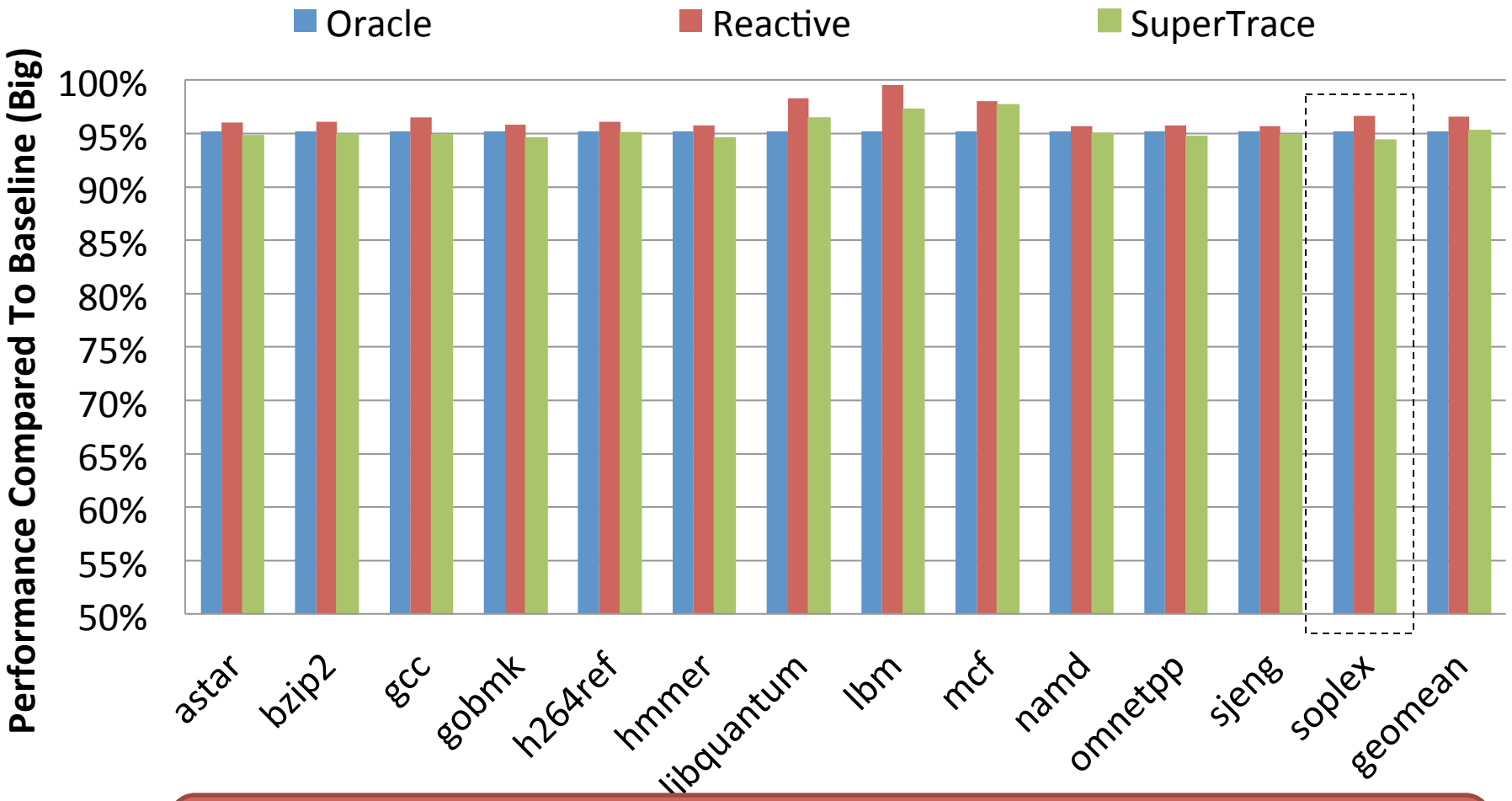

Oracle
Big Quantums = Red,
Little Quantums = Blue
gcc
Hard to predict:
Highly variable data & control flow
Reactive Controller
SuperTrace-Based Controller
Relative Performance of Big vs Little
400
300
200
100
0
−100
0
50000
100000
150000
200000
250000
300000
350000
Super-traces in Increasing Time Order

Time Spent on Little



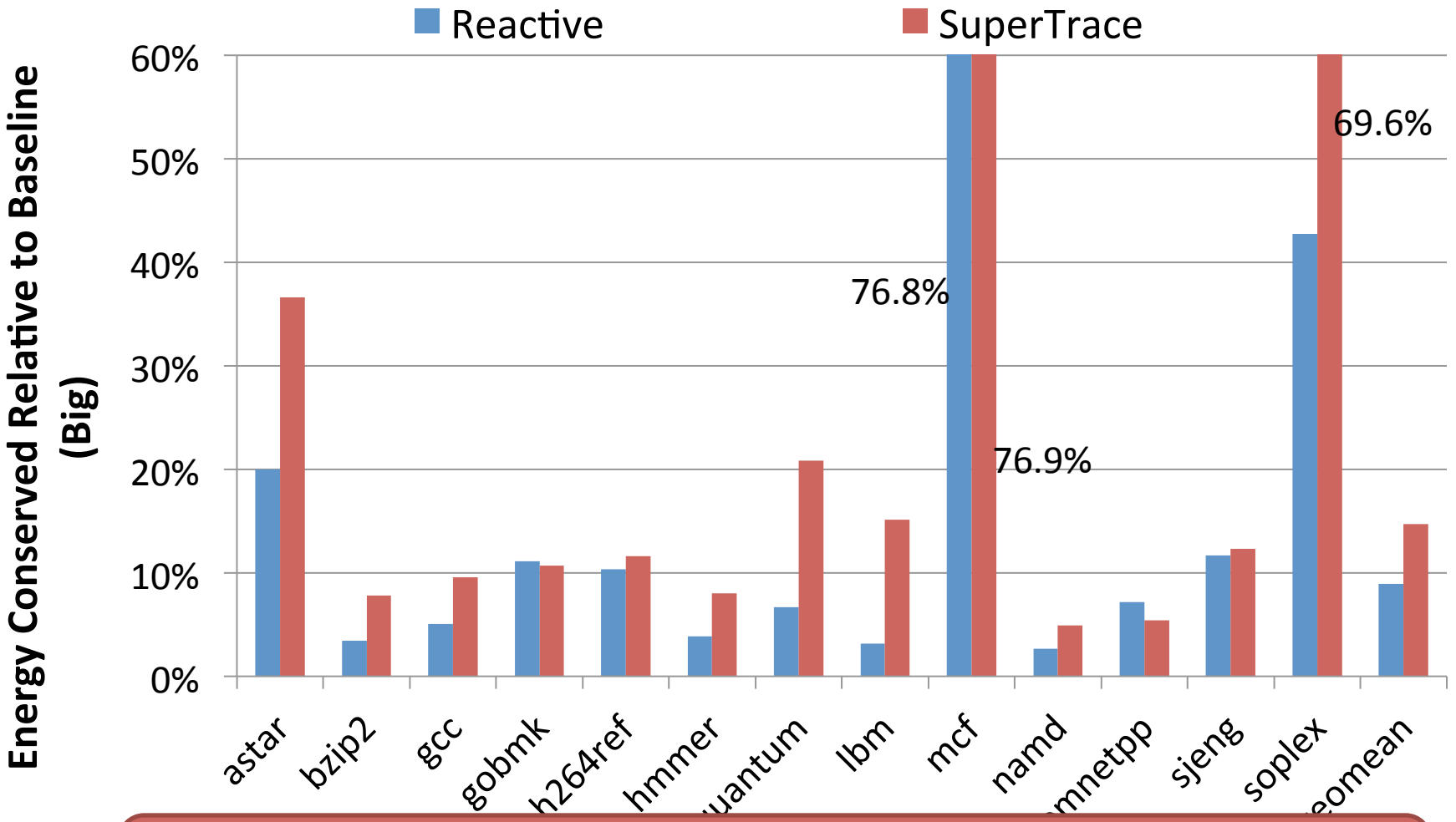
Time spent on the Little backend increases by 46% on average

Performance Relative To Standalone OoO



The controllers all honor the 95% performance target

Energy Conservation



On average, we conserve 43% more energy than a reactive controller

Conclusion

- Don’t React – Predict!
  - Utilize Little 46% more than current work
- 15% energy savings over the baseline (43% more than existing work)
  - small hardware overheads (1.9kB)

Trace Based Switching For A Tightly Coupled Heterogeneous Core

Shruti Padmanabha, Andrew Lukefahr,
Reetuparna Das, Scott Mahlke

shrupad@umich.edu

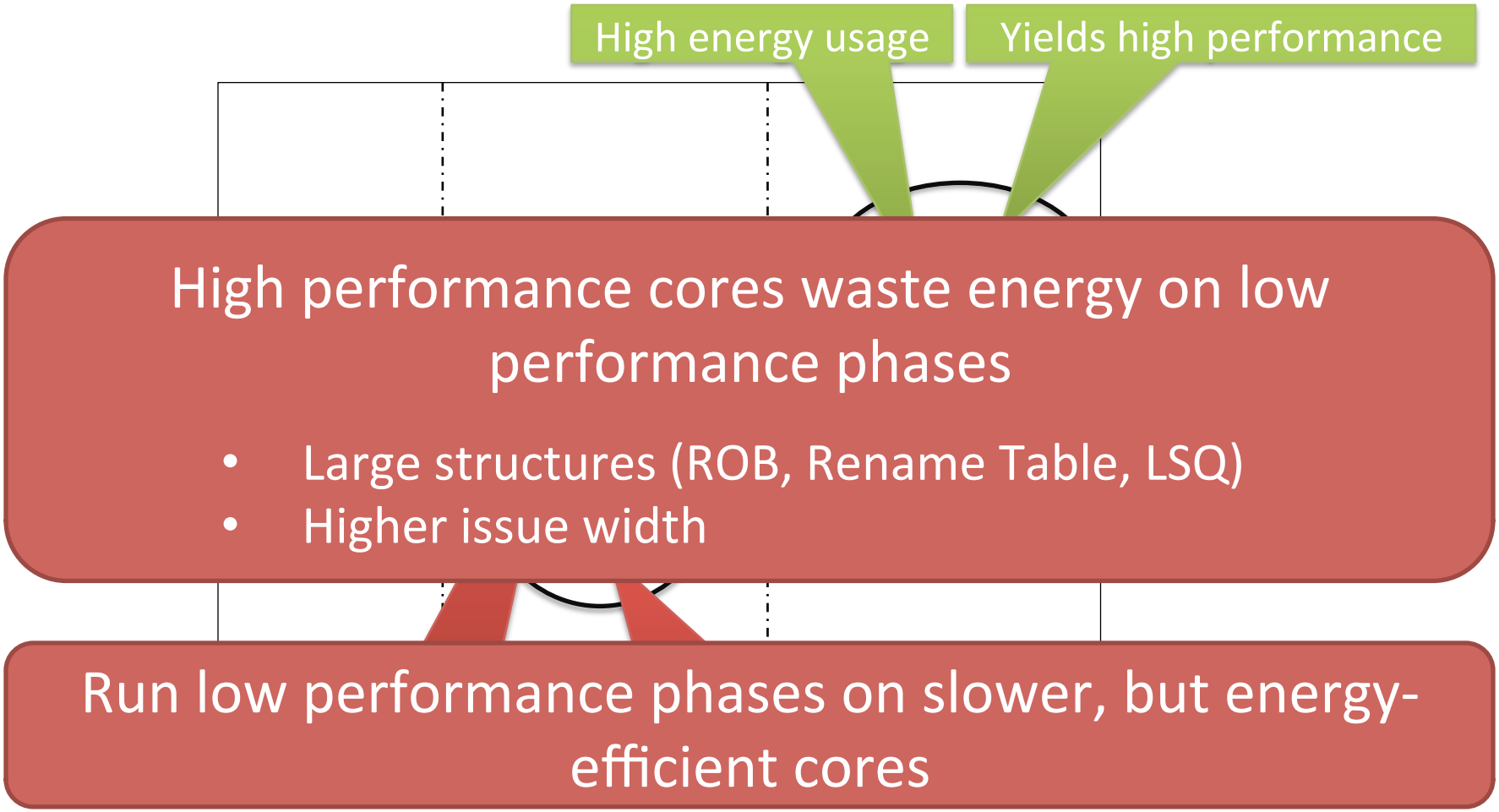
Thank you!

Micro-46

December 2013

Backup

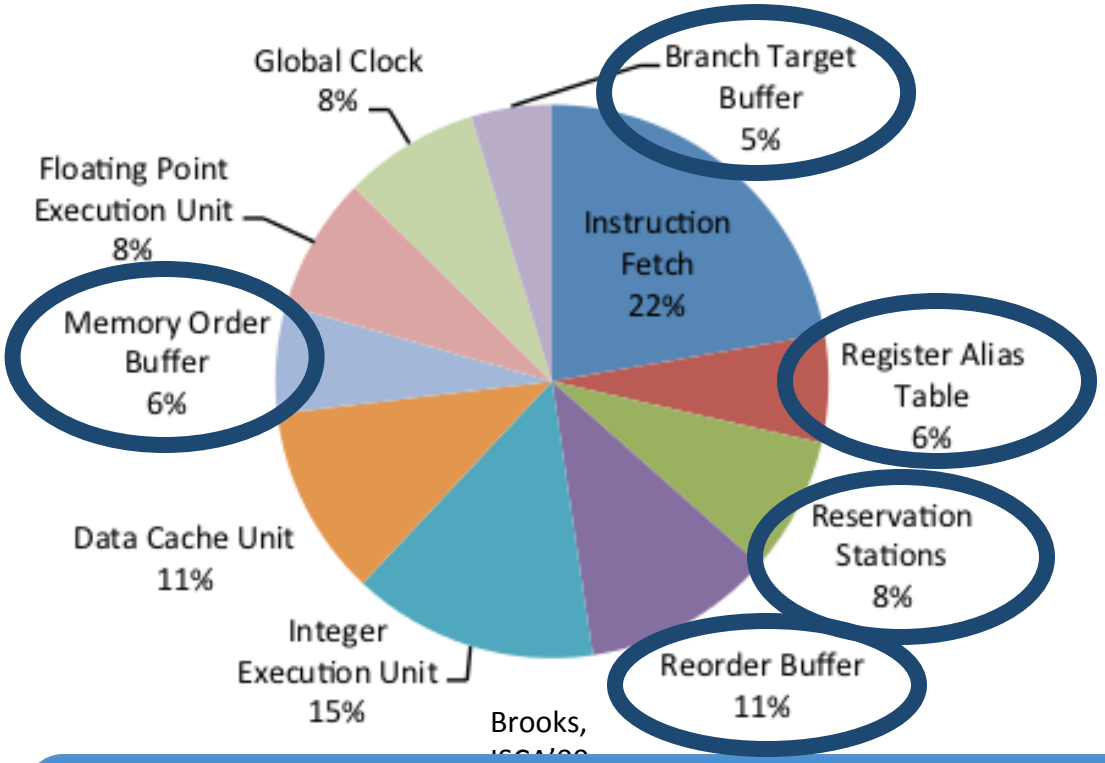
Case for Heterogeneous Cores



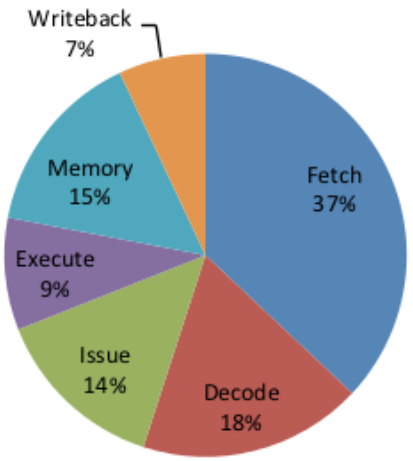
General purpose application executing on a **high performance core**

Core Energy Comparison

Out-of-Order



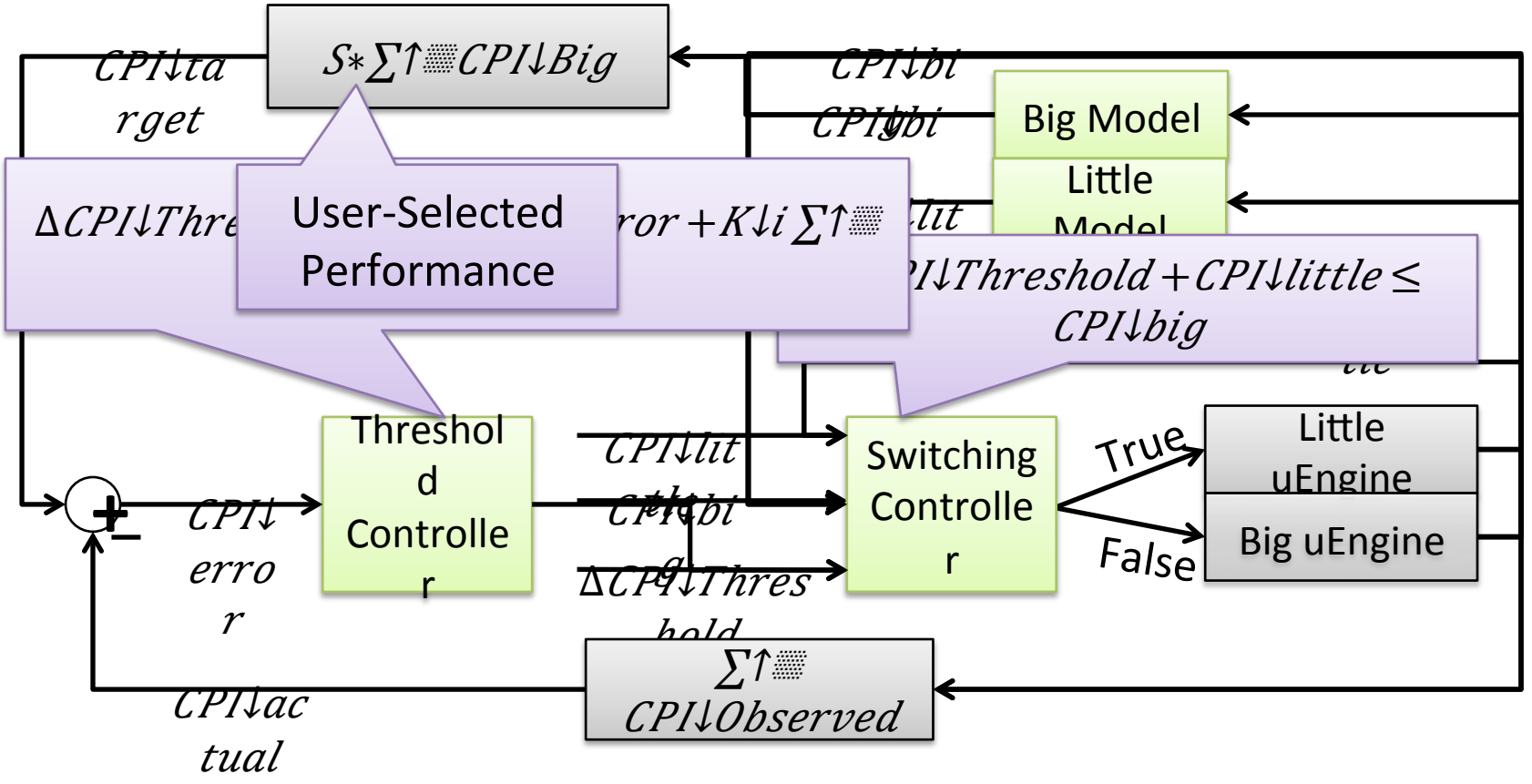
In-Order



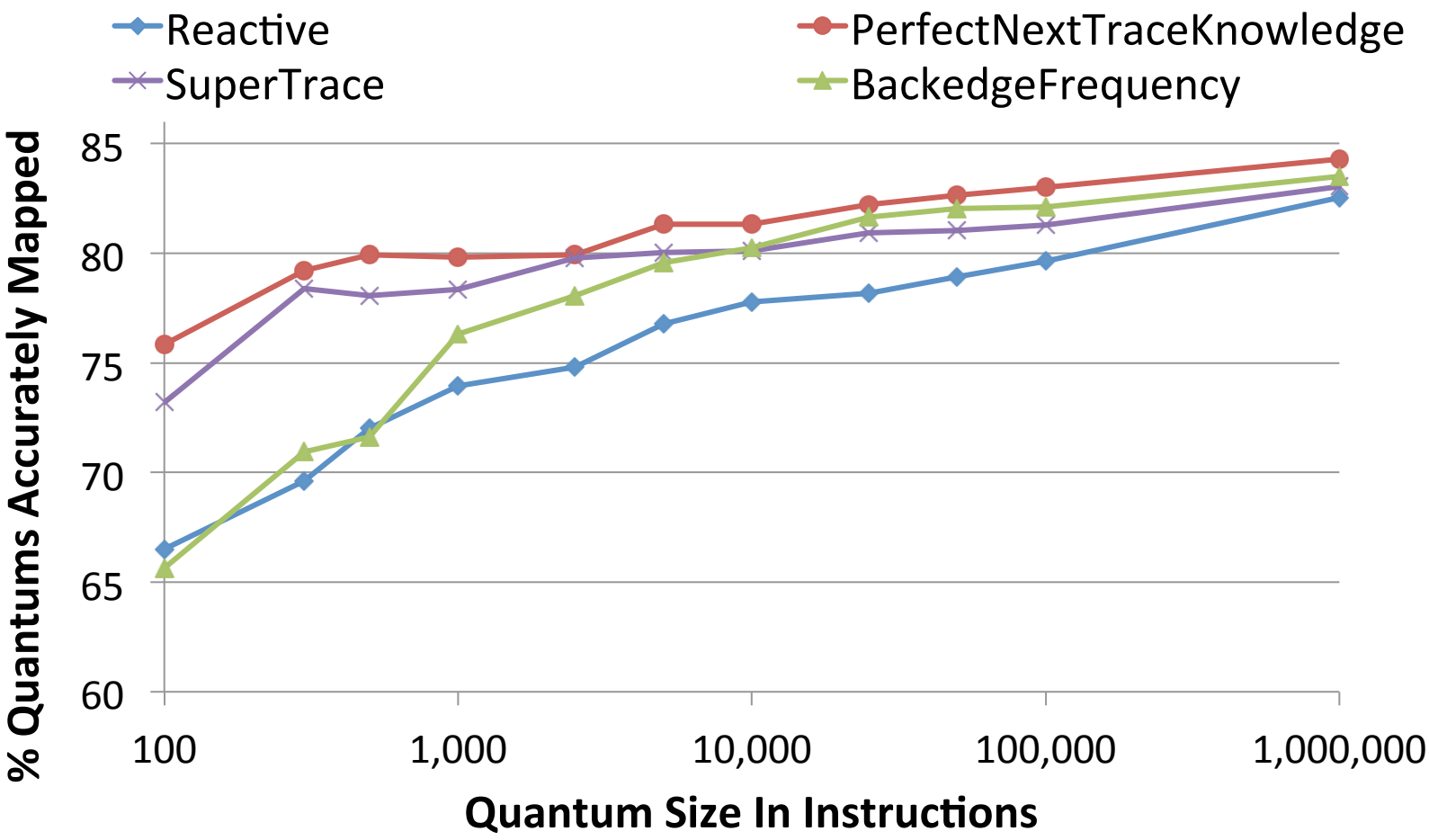
Dally, IEEE Computer'08

Do we always need the extra hardware?

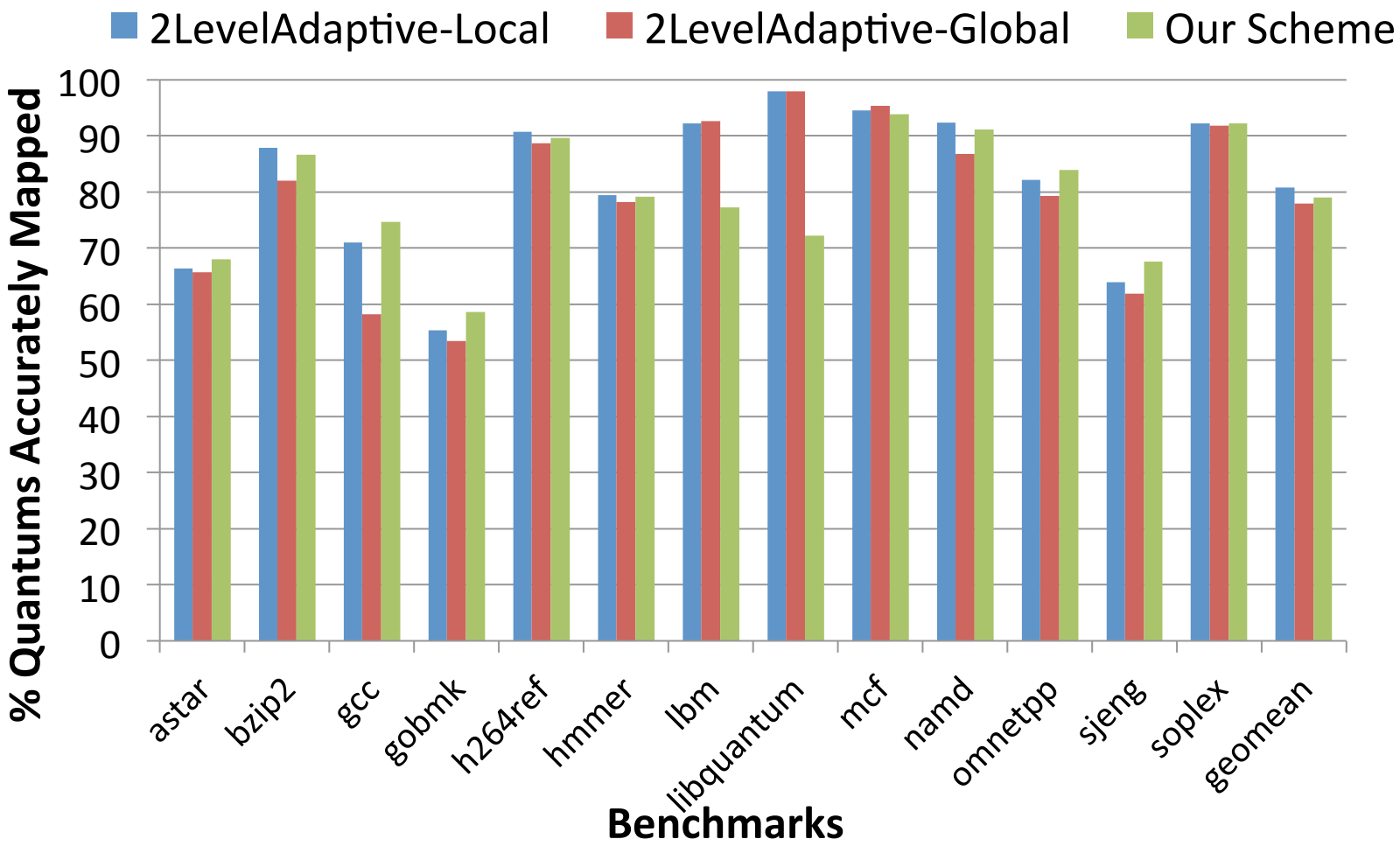
Reactive Online Controller



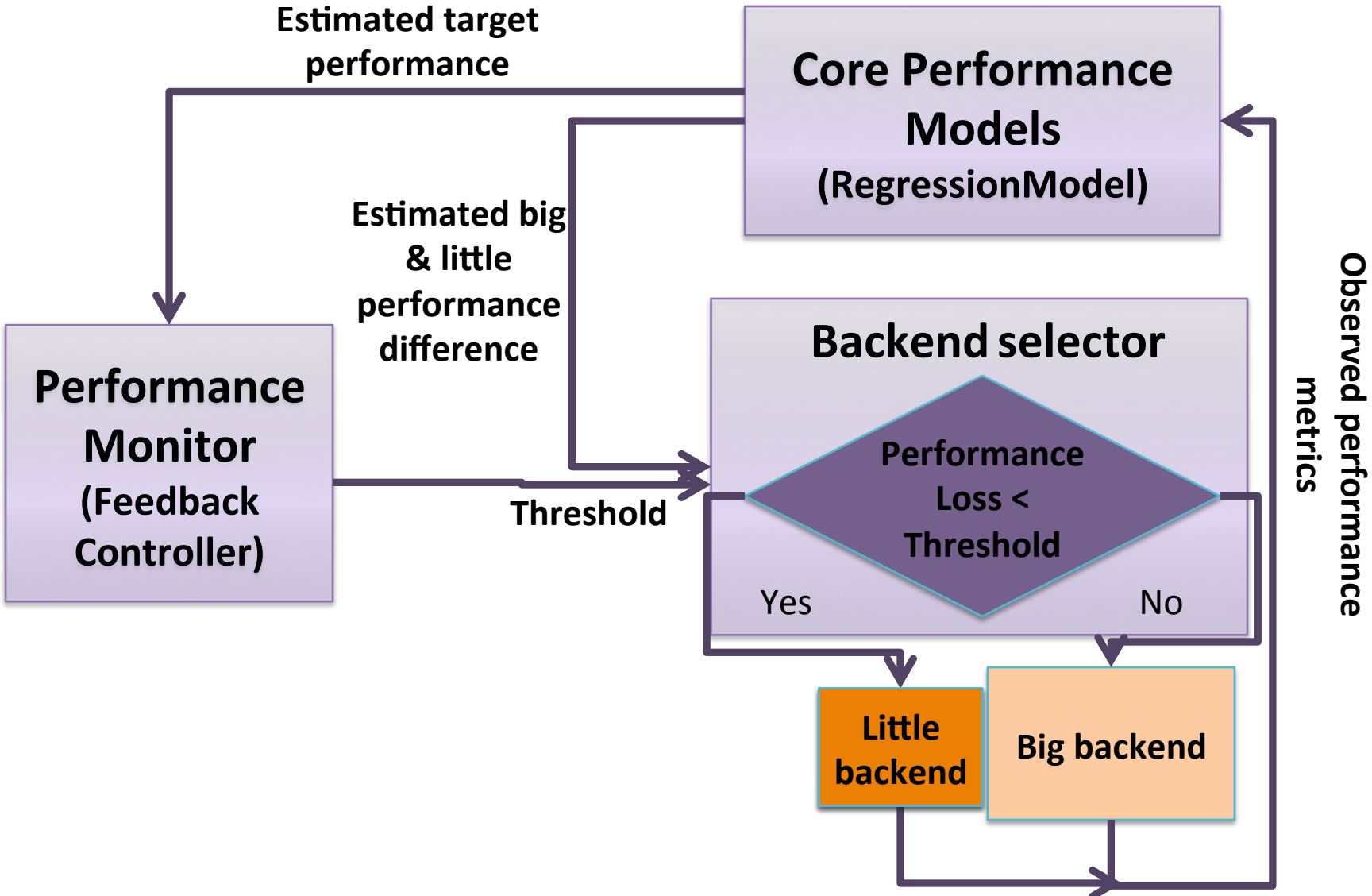
Accuracy


Reactive
PerfectNextTraceKnowledge
SuperTrace
BackedgeFrequency
% Quantums Accurately Mapped
85
80
75
70
65
60
100
1,000
10,000
100,000
1,000,000
Quantum Size In Instructions

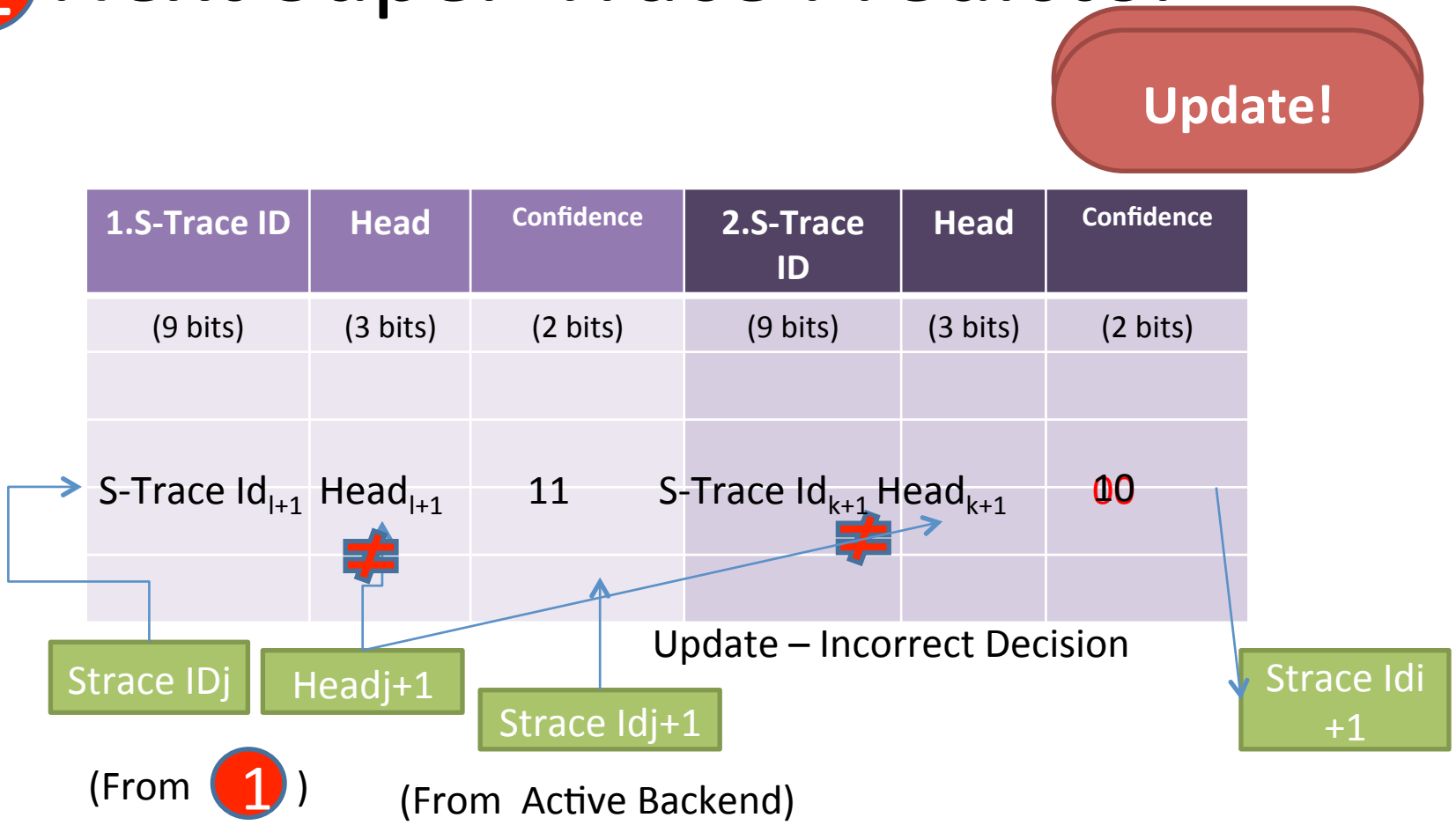
Sensitivity to Other Schemes



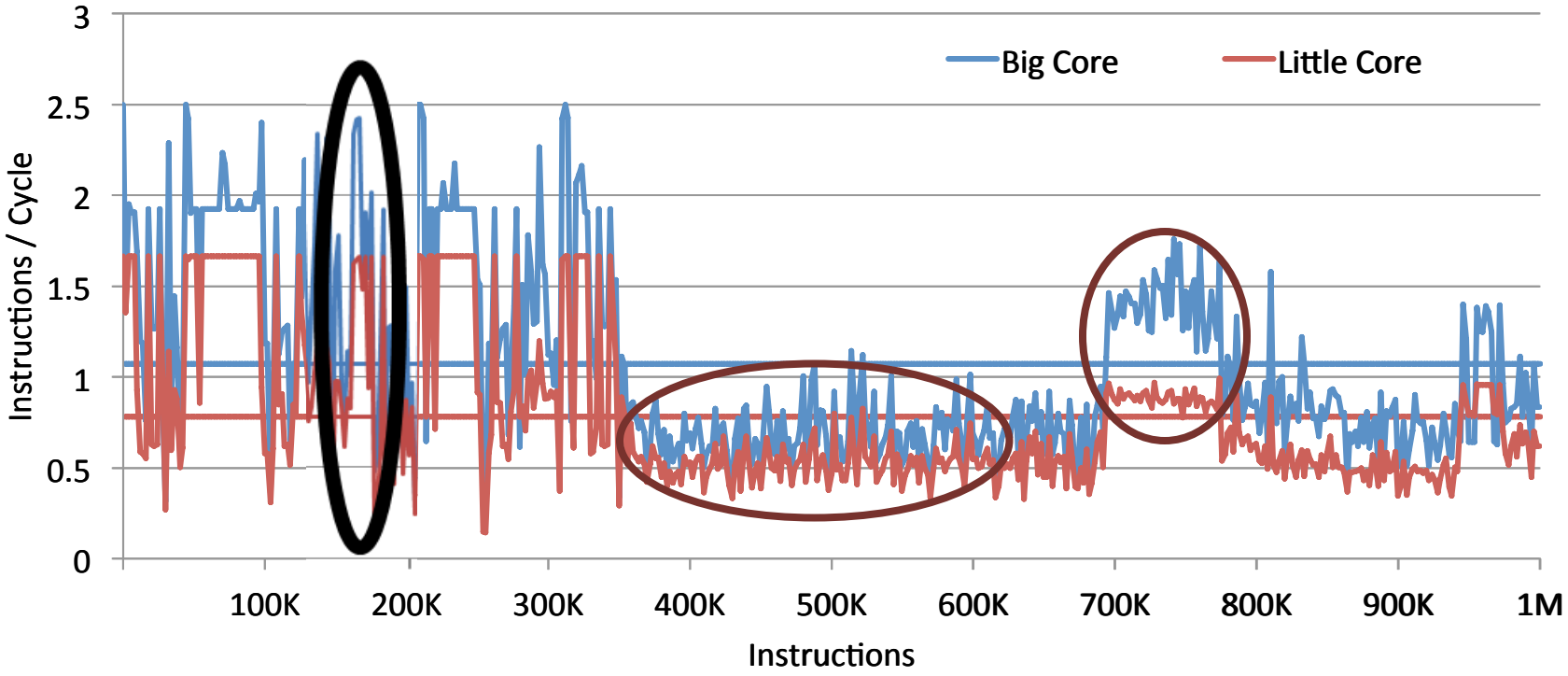
Composite Cores – **Controller** Overview



2 Next Super-Trace Predictor

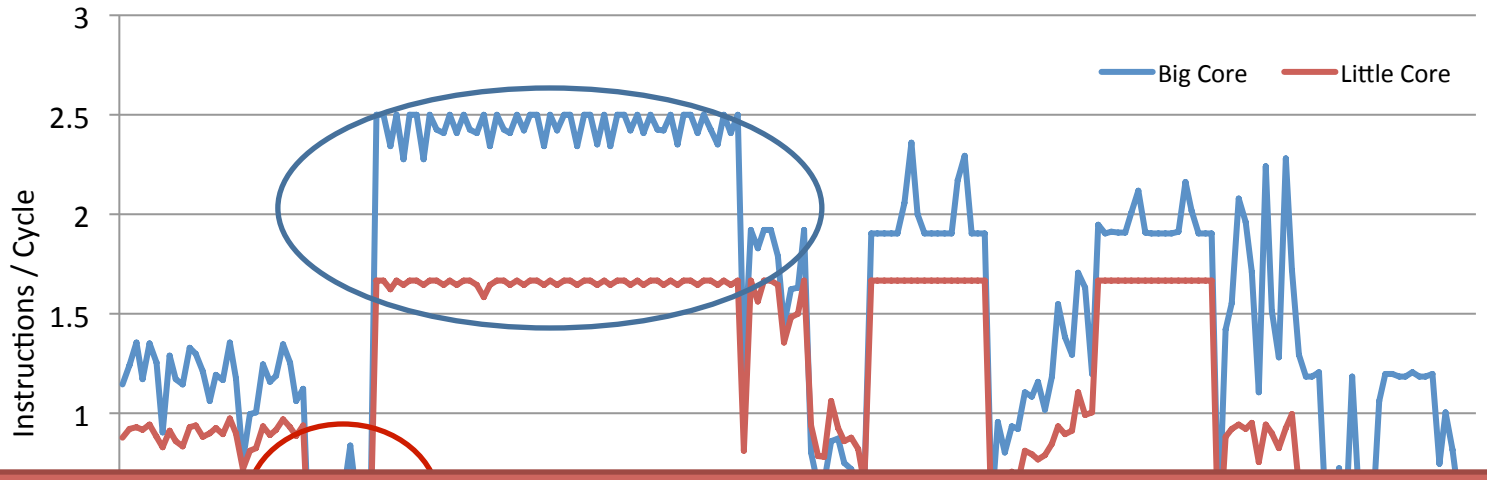


Performance Change In GCC



Huge performance changes within a coarse quantum!

Fine-grain Performance Phases Exist

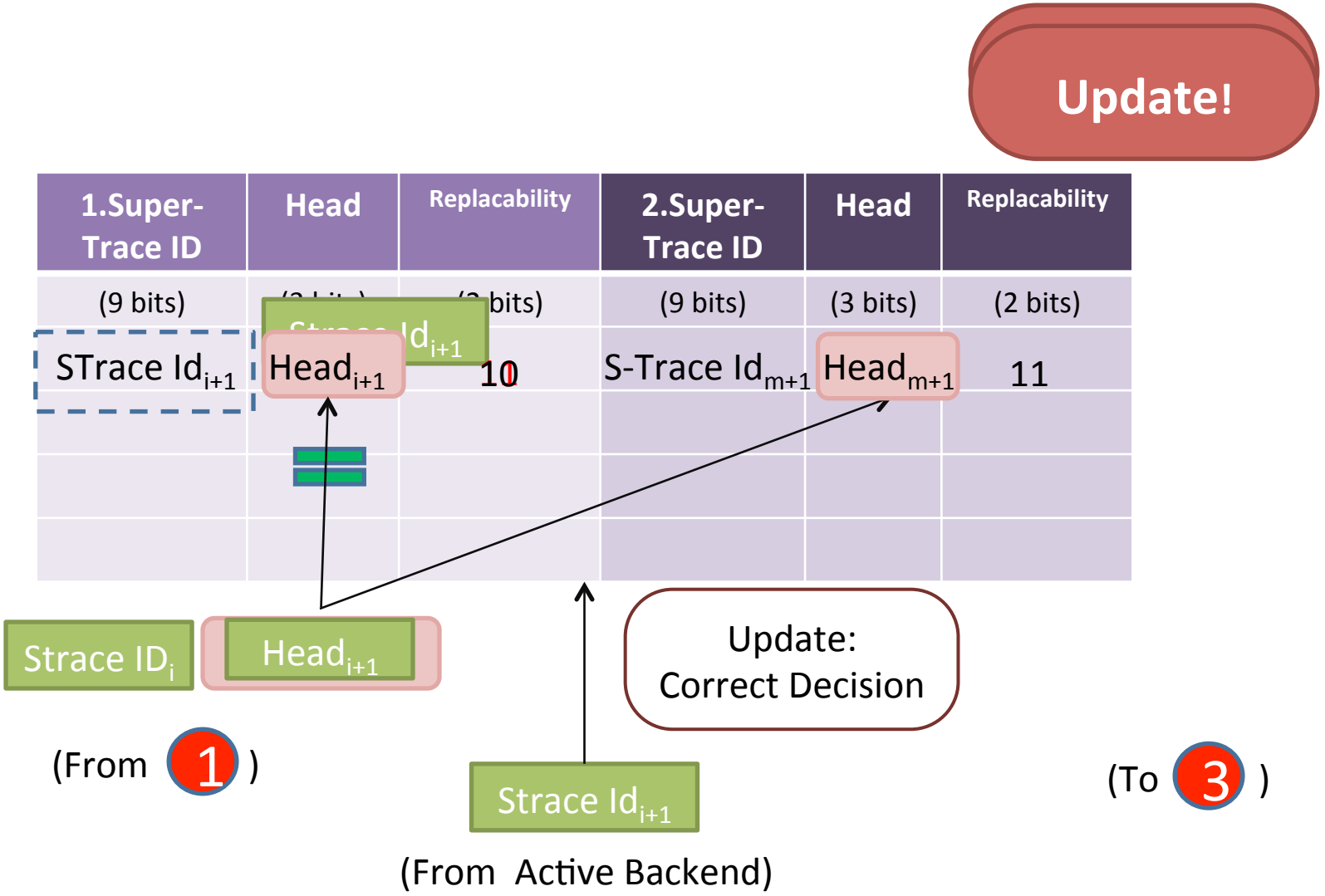


Fine grain offers more opportunity to save energy by exploiting:

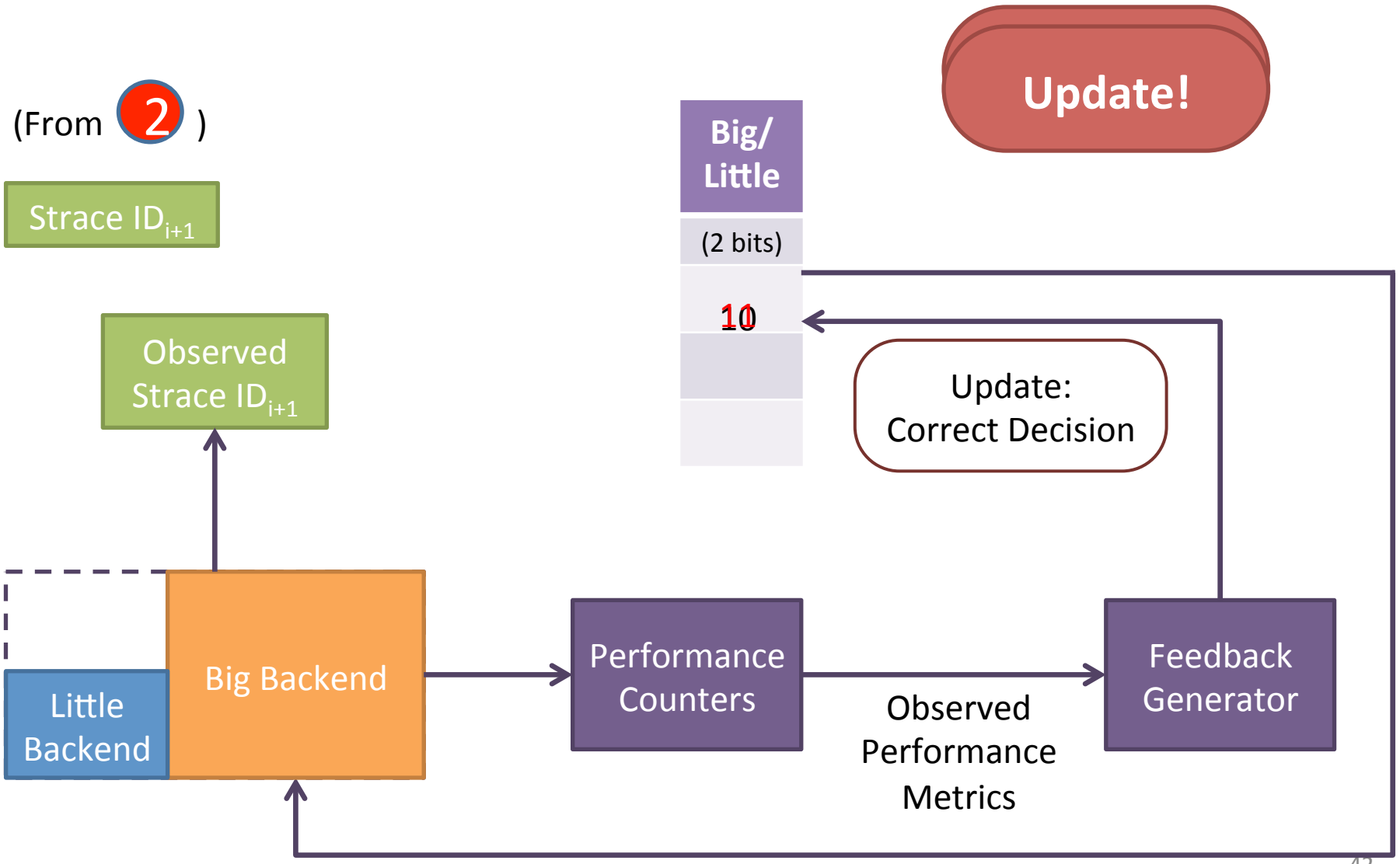
- Dependent compute
- Dependent load misses
- Branch mispredicts

**Composite Cores: Pushing Heterogeneity into a Core, Lukefahr et al, Micro 2012*

2 Next Super-Trace Predictor

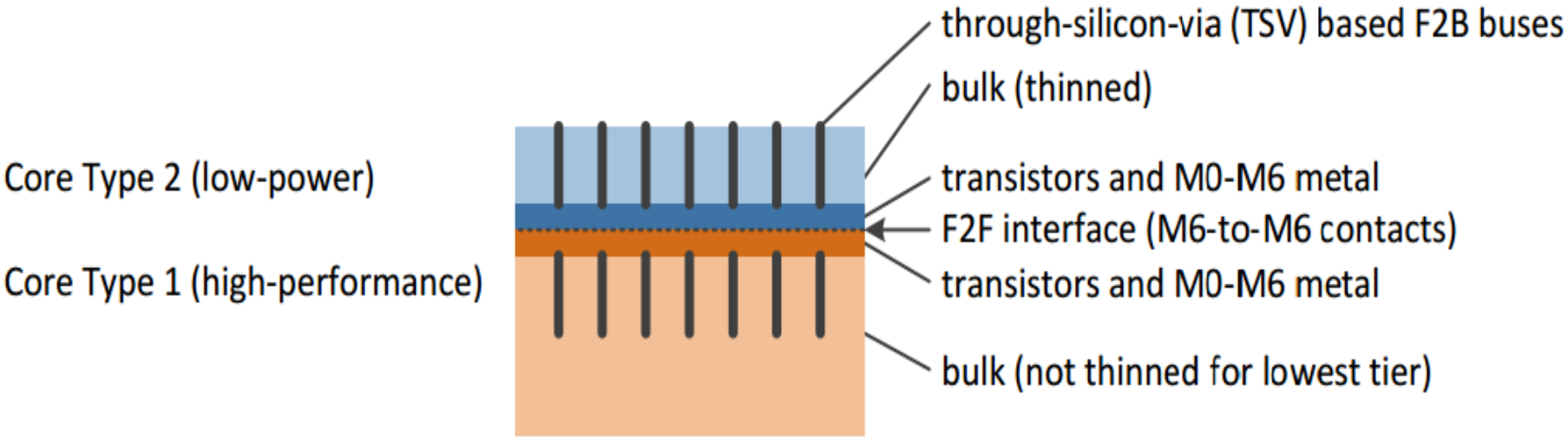


3 Next Backend Predictor



Fine-grained Heterogeneous Architectures

(B) H3 *



**Rationale for a 3D Heterogeneous Multi-core Processor, Rotenberg, ICCD 2013*