Performance Cloning: A Technique for Disseminating Proprietary Applications as Benchmarks

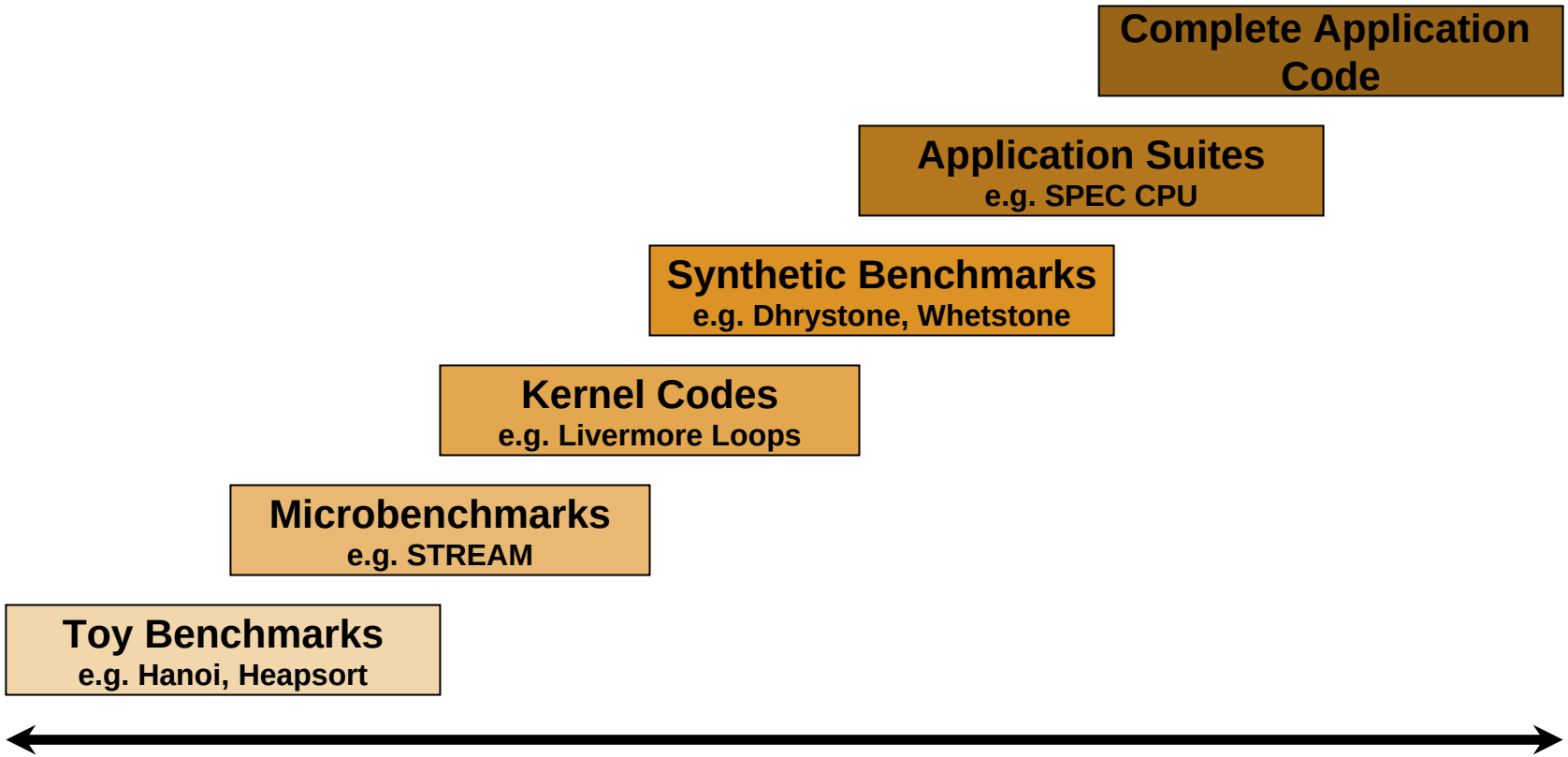
Ajay Joshi (University of Texas)
Lieven Eeckhout (Ghent University, Belgium)
Robert H. Bell Jr. (IBM Corp.)
Lizy John (University of Texas)

IEEE International Symposium on Workload Characterization
October 26, 2006

Outline

- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

Benchmark Spectrum



Less Development Effort	More Development Effort
More Scalable	Less Scalable
More Maintainable	Less Maintainable
Less Representative	More Representative

Real World Applications as Benchmarks

- Increases confidence in making design tradeoffs
- Customize microprocessor design to specific applications
- Best way to understand processor's use
- Perhaps the only way to understand emerging workload characteristics …
- Simplifies purchasing decisions for customers

Challenges With Using Real World Applications

- Real world applications tend to be proprietary
- Using real world applications for performance studies can be tedious

  - Difficult to duplicate user environment

  - Modifying application to research environment

  - Duplicating real input data set

- Real world workloads are a moving target ..

The Problem ….

- Need a methodology to create benchmarks that capture the main performance of real world applications
- Resulting benchmarks should hide functional meaning of code
- Ability to study “what-if” scenarios by varying program characteristics

Outline

- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

Performance Cloning – Central Idea

Real World Application

```
1010101010100101 01
1010101010101010 01
1010101010010101 01
1010101010101001 010
1010101010100101 010
1010101010010010 101 01
1010101010010101 101 01
1010101010101001 01
1010101010101010 01
1010101010101010 01
1010101010100101 010
1010101010101010 101
1010101010101010 01
1010101010101010 11
1010101010010101 010
1010101010101010 101
1010101010101010 11
101010110
```

Workload Characteristics

Instruction Mix
Basic Block Size
ILP
Data Locality
……

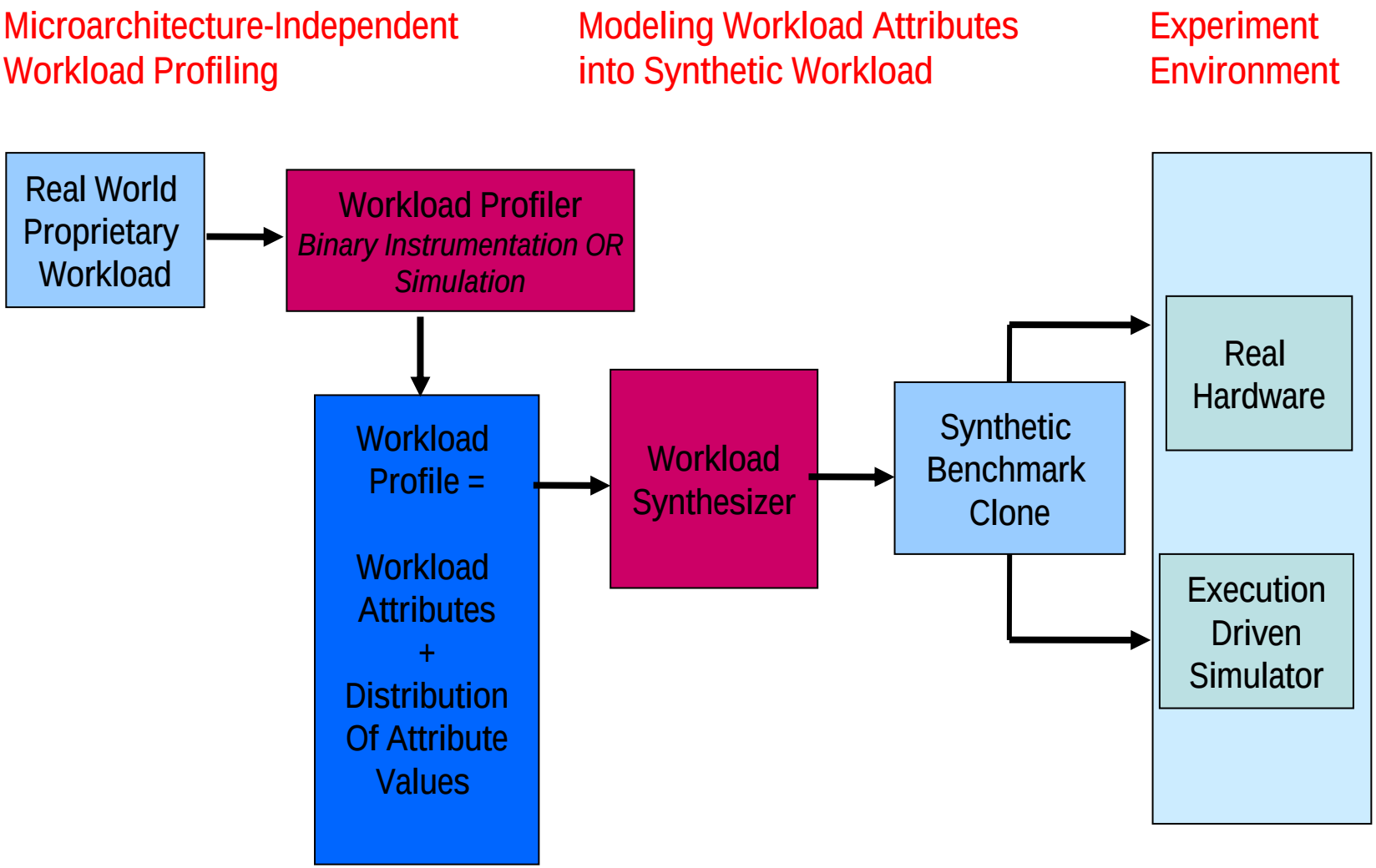
Performance Clone

```
ADD R1, R2,R3
LD R4, R1, R6
MUL R3, R6, R7
ADD R3, R2, R5
DIV R10, R2, R1
SUB R3, R5, R6
STORE R3, R10, R20
ADD R1, R2,R3
LD R4, R1, R6
MUL R3, R6, R7
ADD R3, R2, R5
DIV R10, R2, R1
SUB R3, R5, R1
BEQ R3, R6, LOOP
SUB R3, R5, R6
STORE R3, R10, R20
DIV R10, R2, R1
……..
```

Measure Inherent Workload Characteristics

Generate Clone with Similar Characteristics

Performance Cloning Framework



Outline

- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

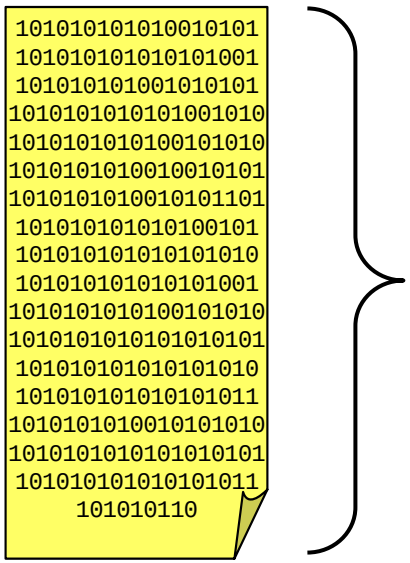
Microarchitecture-Independent Profile

- Control Flow Behavior
- Data Locality
- Control Flow Predictability
- Instruction Mix
- Instruction Level Parallelism

Control Flow Behavior (1)

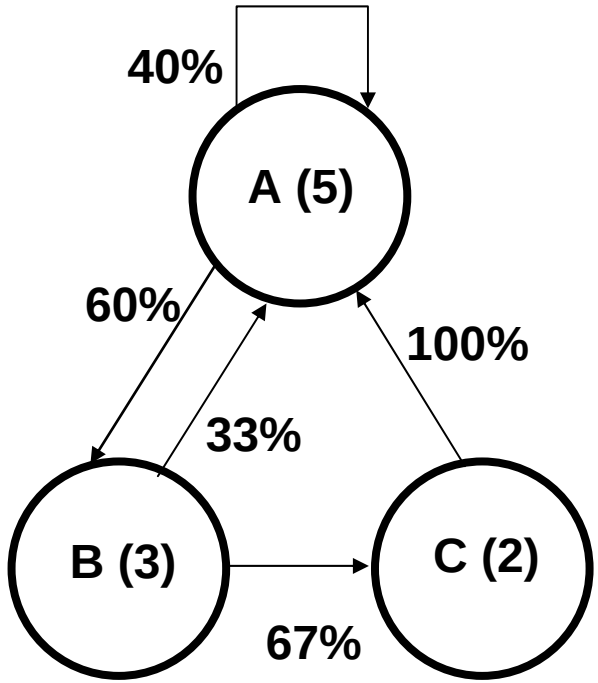
Statistical Flow Graph (K=0)

[Eeckhout *et al.*, ISCA 2004]



Application Binary

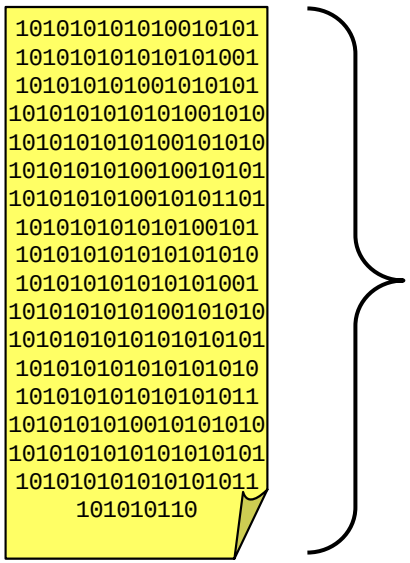
Profiling



Control Flow Behavior (2)

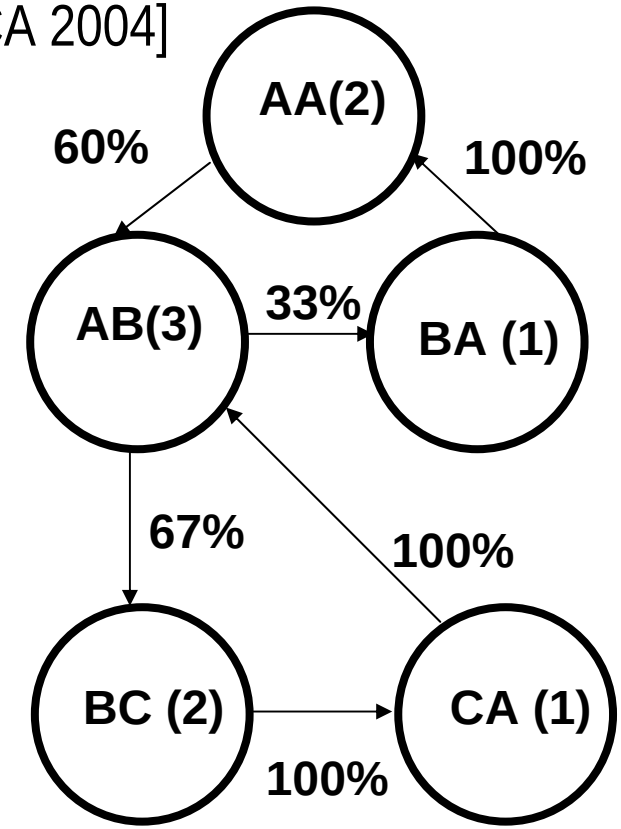
Statistical Flow Graph (K=1)

[Eeckhout *et al.*, ISCA 2004]



Application Binary

Profiling



Modeling Memory Data Access Pattern

- Identify streams of data references
- A Stream?
  - Sequence of memory addresses in an arithmetic progression
  - Elements of arrays A, B, and C form 3 streams

```
for( ii = 0; ii < N; ii ++)
      A [ii]        =        B [ii]        +        C [ii]

200, 204, 208 ..        320, 324, 328 ..        404, 408, 412 ...
Issuing Sequence :  320, 404, 200, 324, 408, 204 ....
```

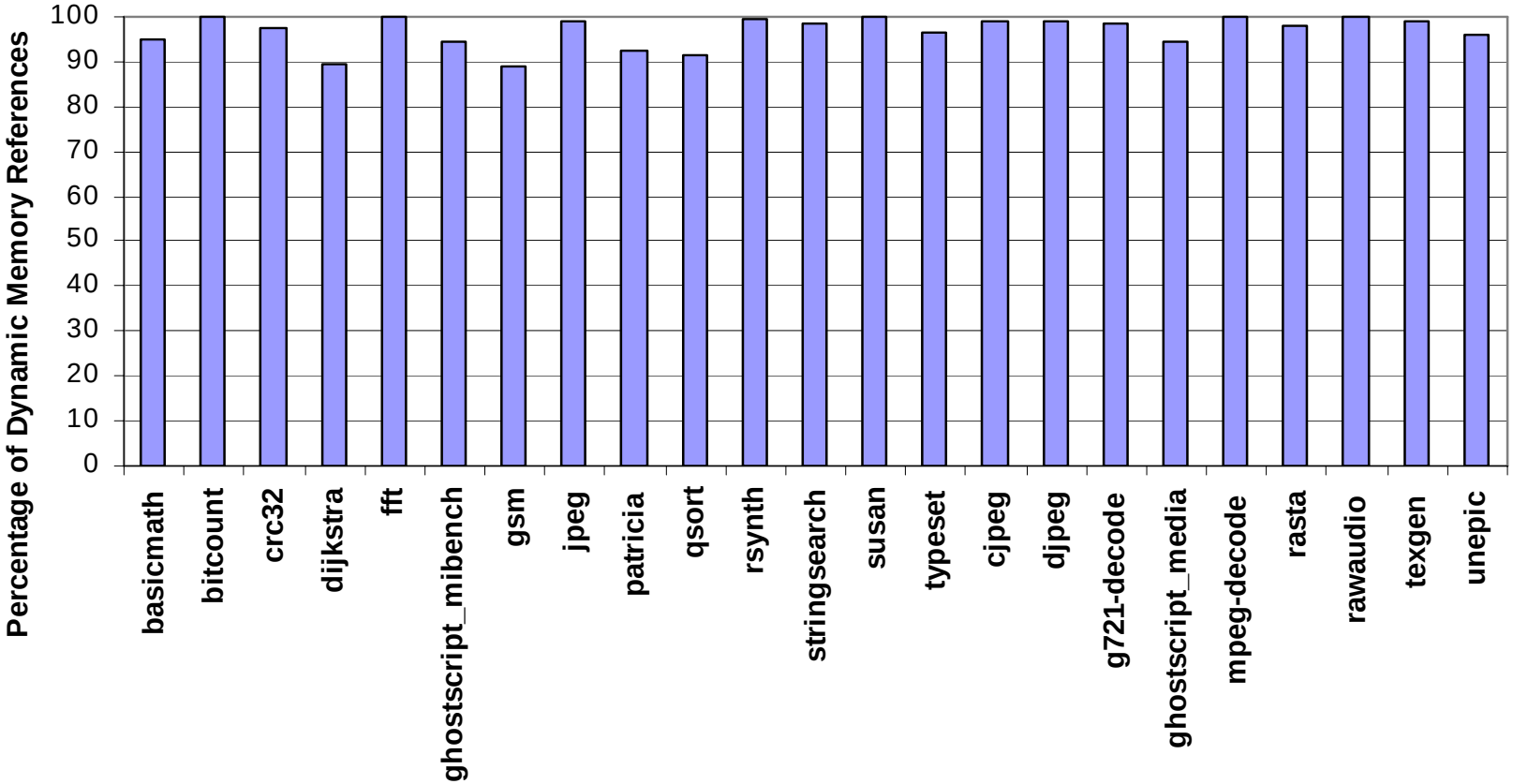
- Streams are interleaved and may contain noise

  4, 8, 12, 16, 1, 3, 20, 24, 5, 7, 2, 9, 11, 28 ...

Extracting Streams

- Reference pattern of static Load / Store Instructions
  - PC-correlated spatial locality
    - Dependence on address referenced by nearby Ld / St
    - Programs with pointer chasing codes
  - PC-correlated temporal locality
    - Dependence on previous address generated by same Ld / St
    - Programs with multidimensional arrays
- Could static Load / Store instructions be natural sources of streams ?
- Profile every static Load / Store instruction
  - Number of different strides with which it accesses data

Behavior of Static Load/Store Instructions



As a First-Order Model, Static Load/Stores can be modeled as single stream

Modeling Control Flow Predictability

- Capture behavior of easy and difficult to predict branches
- Inherent program feature that captures branch behavior
- Transition Rate [Haungs et al. HPCA'00]

  # of Taken-Not Taken transitions / # of times executed
- Branches with low transition-rate (easier to predict)

  TTTTTTTTTN, NNNNNNNNNT
- Branches with high transition-rate (easier to predict)

  TNTNTNTNTN
- Branches with moderate transition-rate (tougher to predict)

Modeling Instruction Level Parallelism

Dependency Distance

ADD R1, R3,R4

MUL R5,R3,R2

ADD R5,R3,R6

LD R4, (R8)

SUB R8,R2, R1

Read After Write Dependency Distance = 3

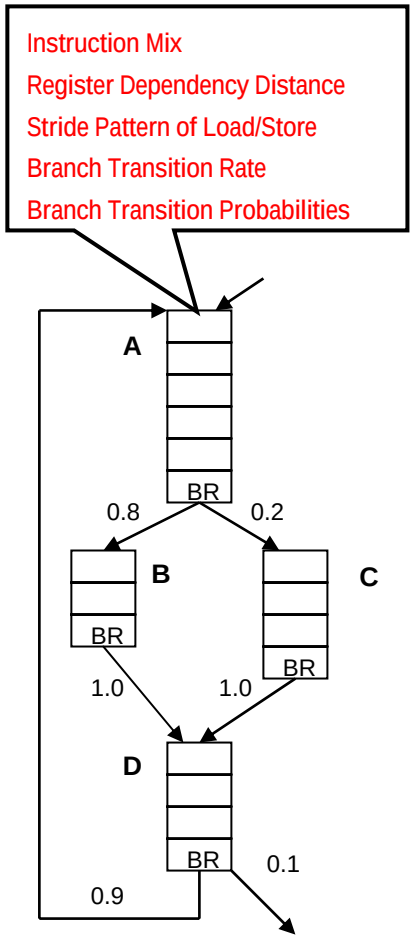
Measure Distribution of Dependency Distances

Upto 1, Upto 2, Upto 4, Upto 8, Upto 16, Upto 32, >32

Outline

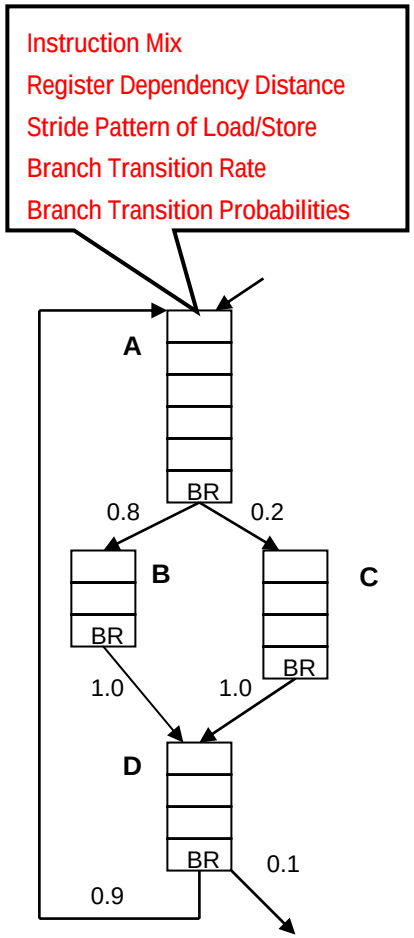
- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

Performance Clone Generation

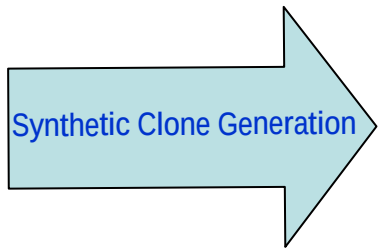


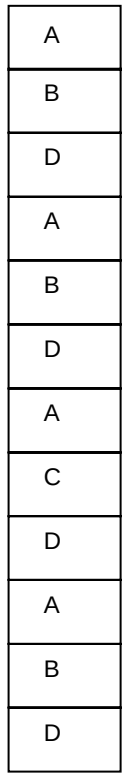
Workload Profile

Performance Clone Generation

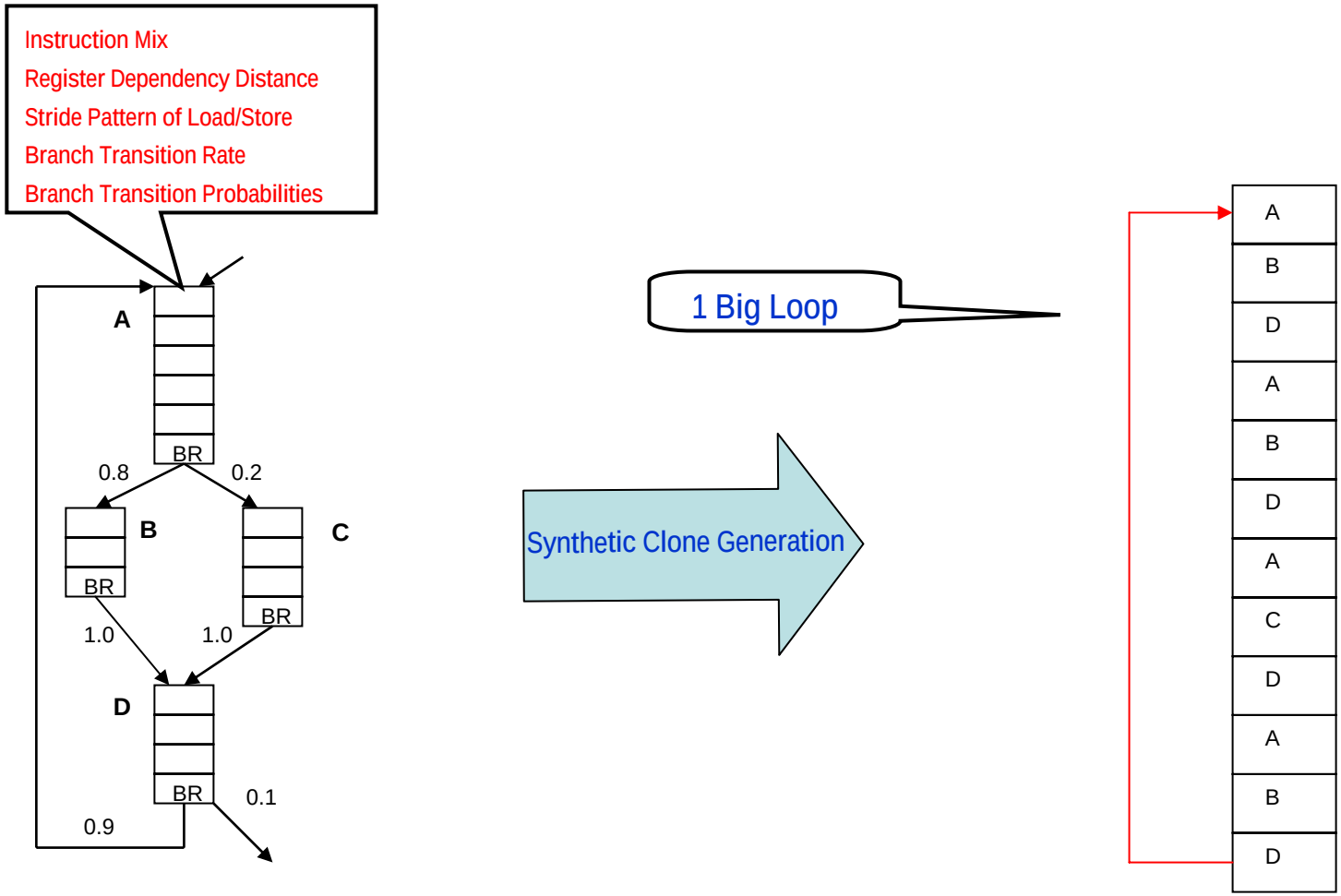


Workload Profile

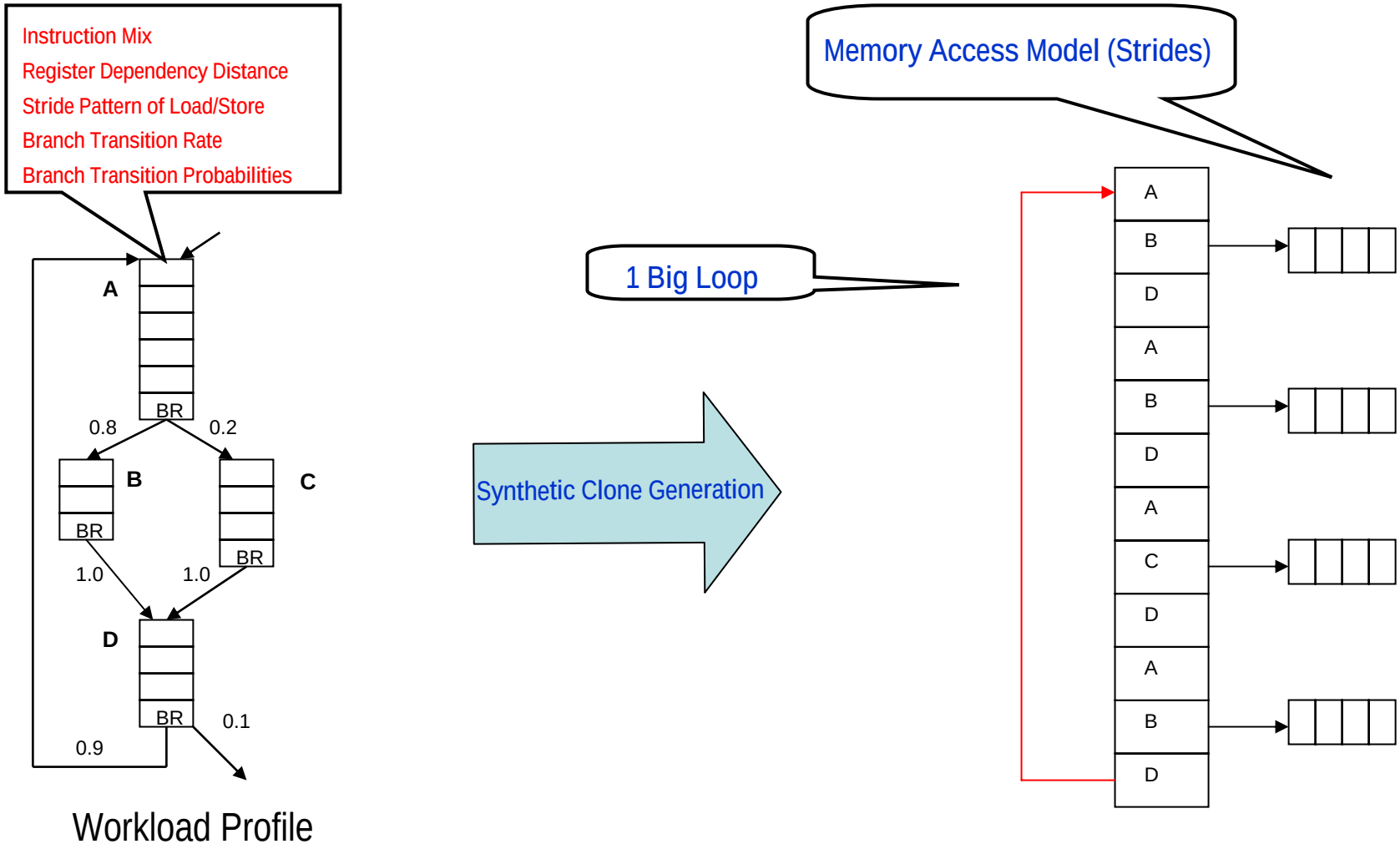




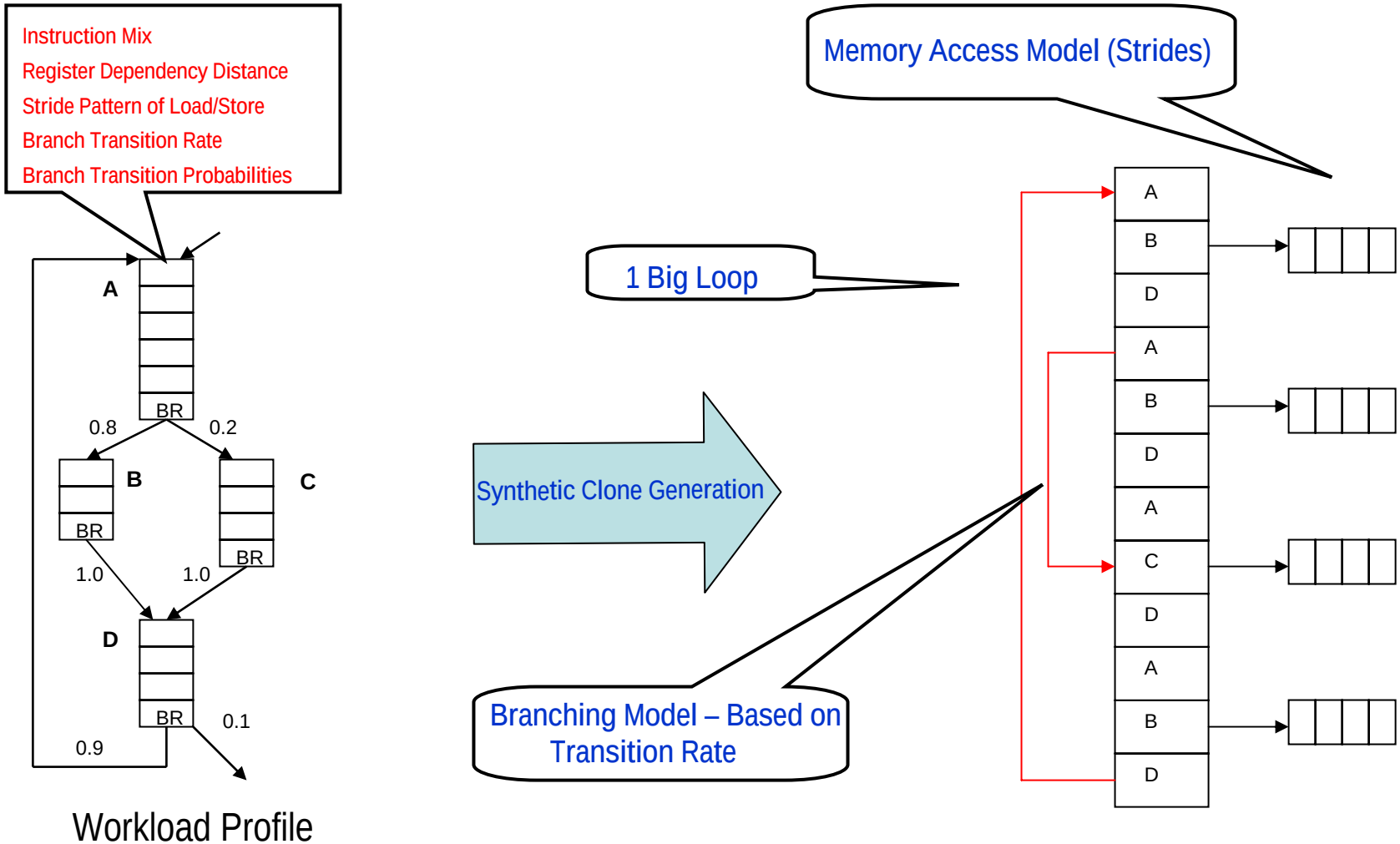
Performance Clone Generation



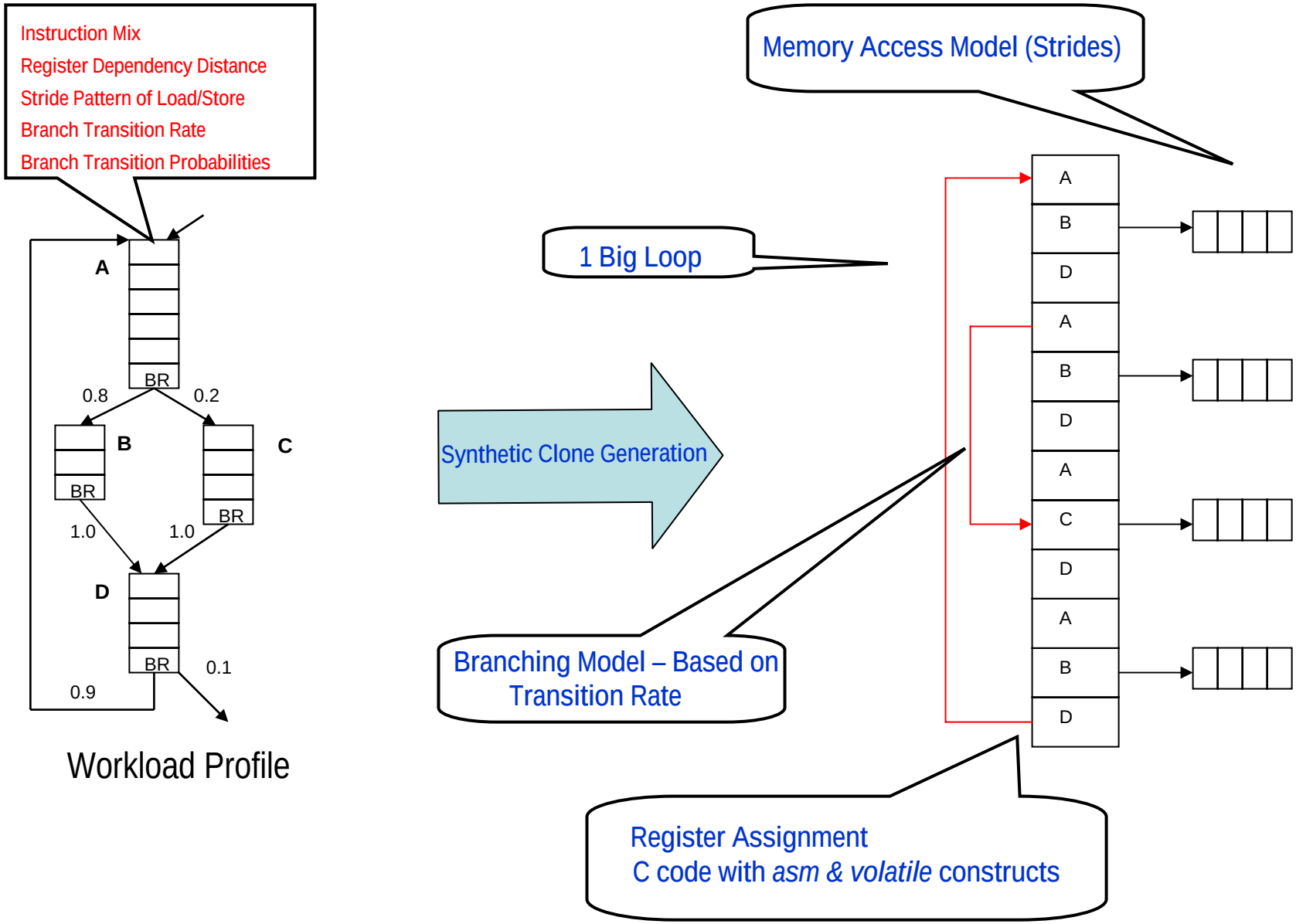
Performance Clone Generation



Performance Clone Generation



Performance Clone Generation



Outline

- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

Tools & Benchmarks

- SimpleScalar/Wattch Simulators for profiling and cycle-accurate simulation
- Alpha ISA – Programs compiled with Compaq cc v6.3-025 –O3 level
- Benchmarks from MiBench and MediaBench benchmark suites as representatives of characteristics of Embedded Applications

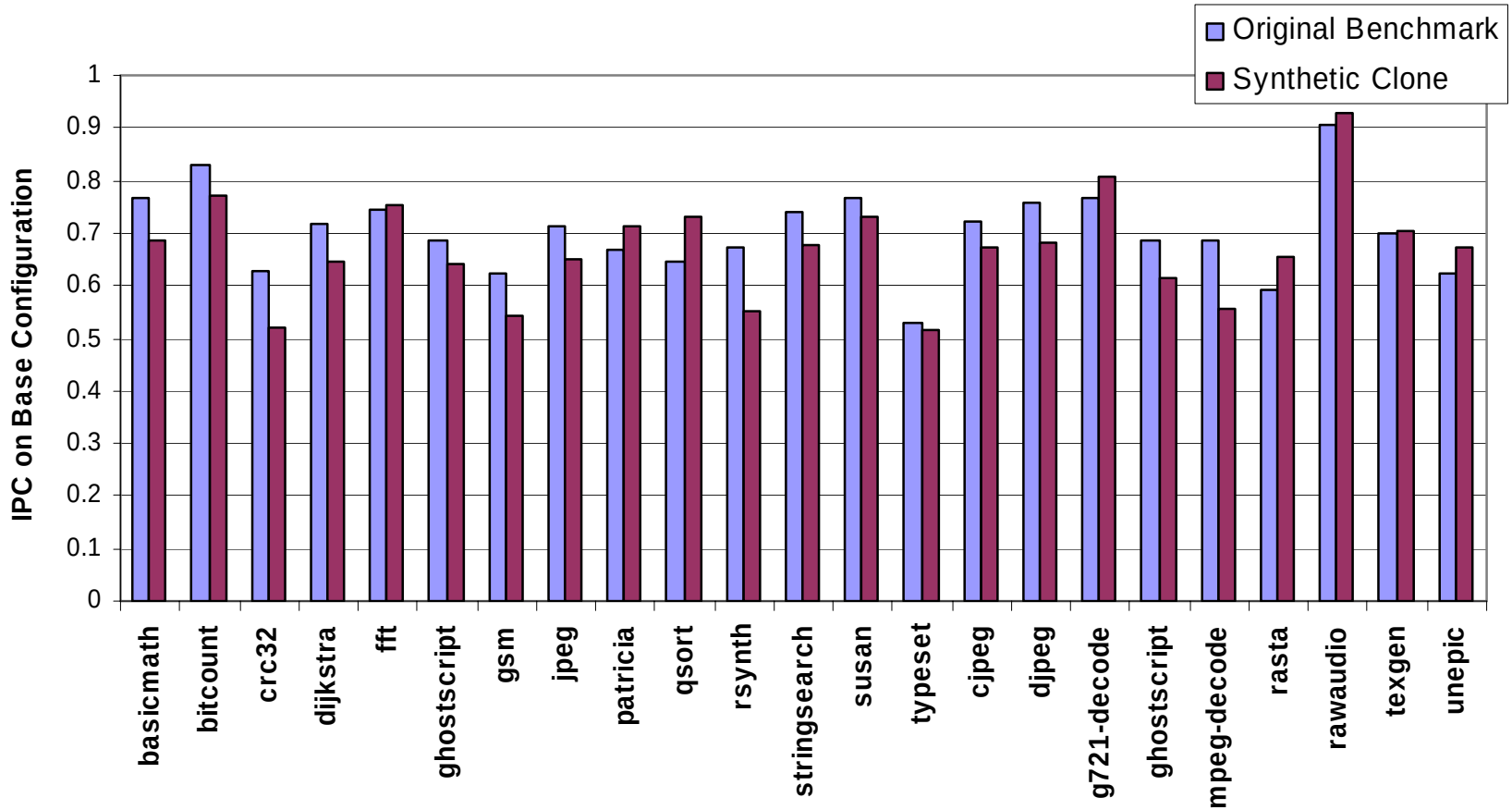
Program	Application Domain
basicmath, qsort, bitcount, susan	Automotive
crc32, dijkstra, patricia	Networking
fft, gsm	Telecommunication
ghostscript, rsynth, stringsearch	Office
jpeg, typeset	Consumer
cjpeg, djpeg, g721-decode, ghostscript, mpeg, rasta, rawaudio, texgen, unepic	Media

Evaluation

- Absolute accuracy
  - Ability of performance clone to estimate absolute IPC and Power
- Relative accuracy
  - Sensitivity (IPC and Power) of performance clone to cache & microarchitecture design changes
- Base Configuration

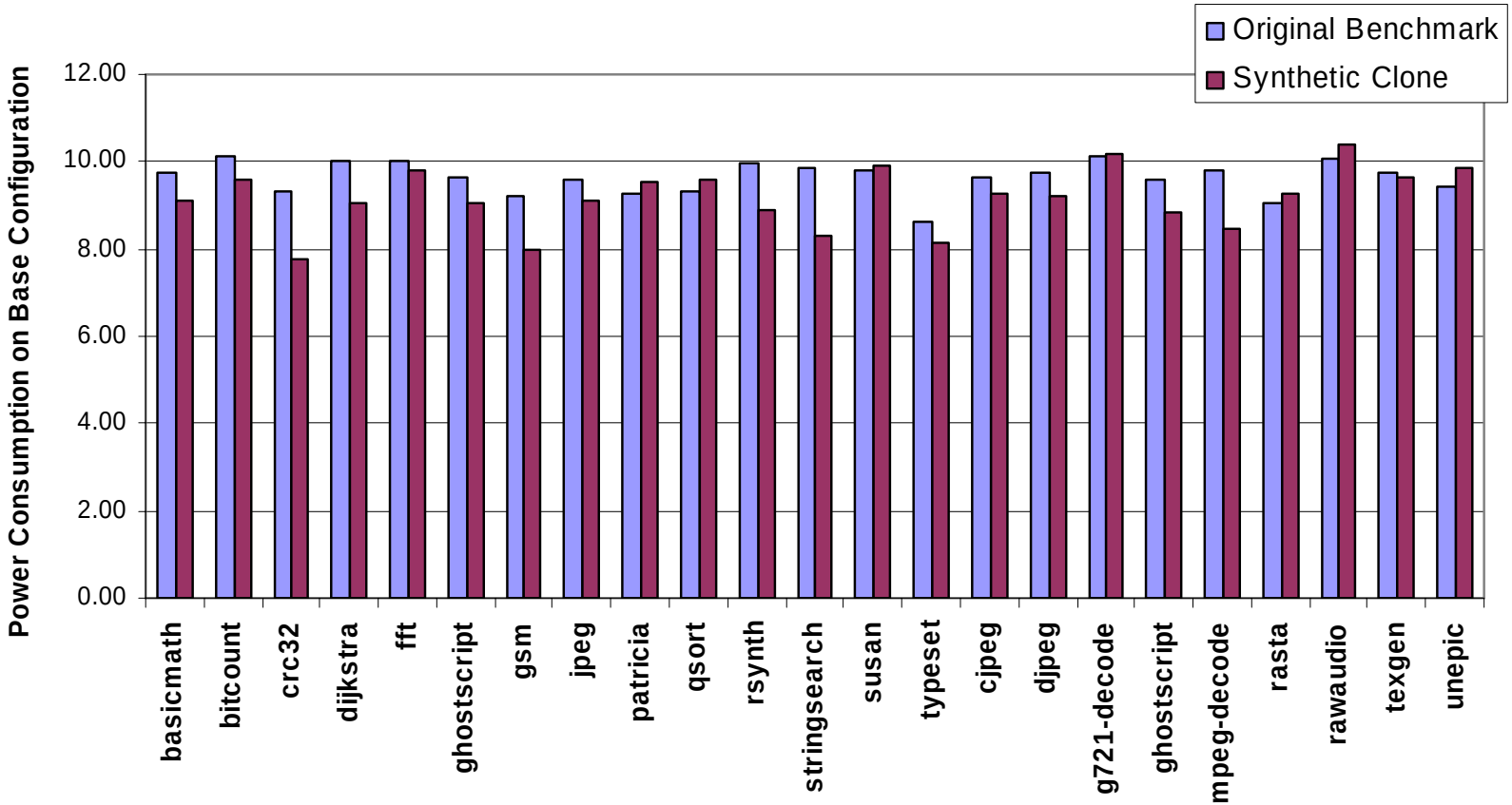
L1 I-cache	16 KB/2-way/32 B
L1 D-cache	16 KB/2-way/32 B
L2 Unified cache	64 KB/4-way/64 B
Fetch, Decode, and Issue Width	1-wide out-of-order
Fetch Queue	8 entry
Branch Predictor	2-level GAp predictor
Functional Units	2 Integer ALU, 1 FP Multiplication Unit, 1 FP ALU
Reorder Buffer	16 entries
Load Store Queue	8 entries
Memory (Bus Width, First Block Latency)	8 B, 40 cycles

Absolute Accuracy in IPC



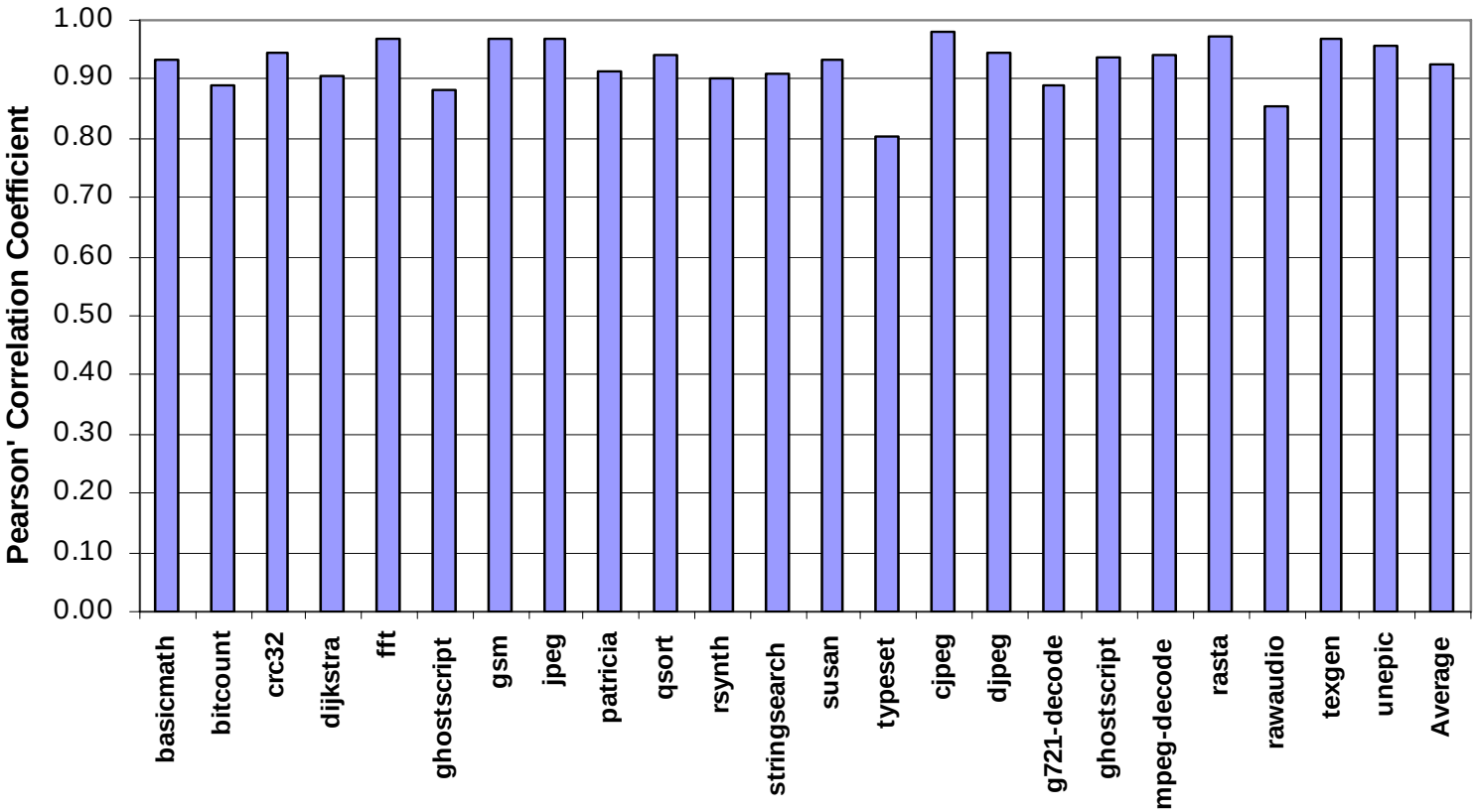
Average absolute error in estimating IPC is 8.7%

Absolute Accuracy in Power



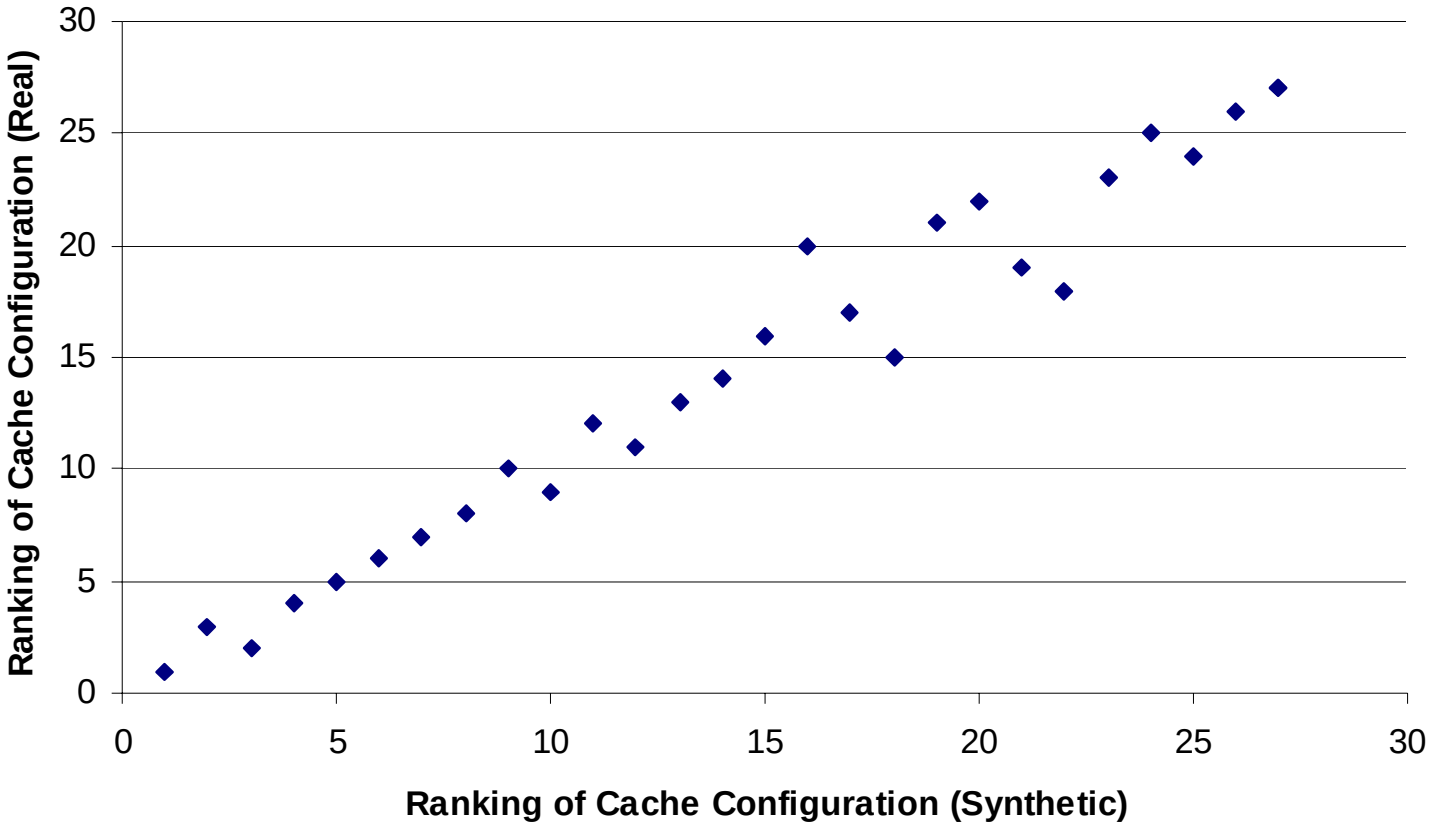
Average absolute error in estimating power is 6.4%

Tracking Design Changes (1)



Across 28 cache configurations

Tracking Design Changes (2)



Across 28 cache configurations

Tracking Design Changes (3)

Design Change	Average Relative Error in IPC	Average Relative Error in Power
Double the number of entries in the reorder buffer and load store Queue	5.81%	3.41%
Reduce the L1 cache size to half	1.48%	0.39%
Double the fetch, decode, and issue Width	5.41%	4.59%
Change the predictor from a 2-level to a not-taken predictor	6.51%	1.80%
Change the instruction issue policy to in-order	3.26%	1.22%

5 Different Microarchitecture Changes

Outline

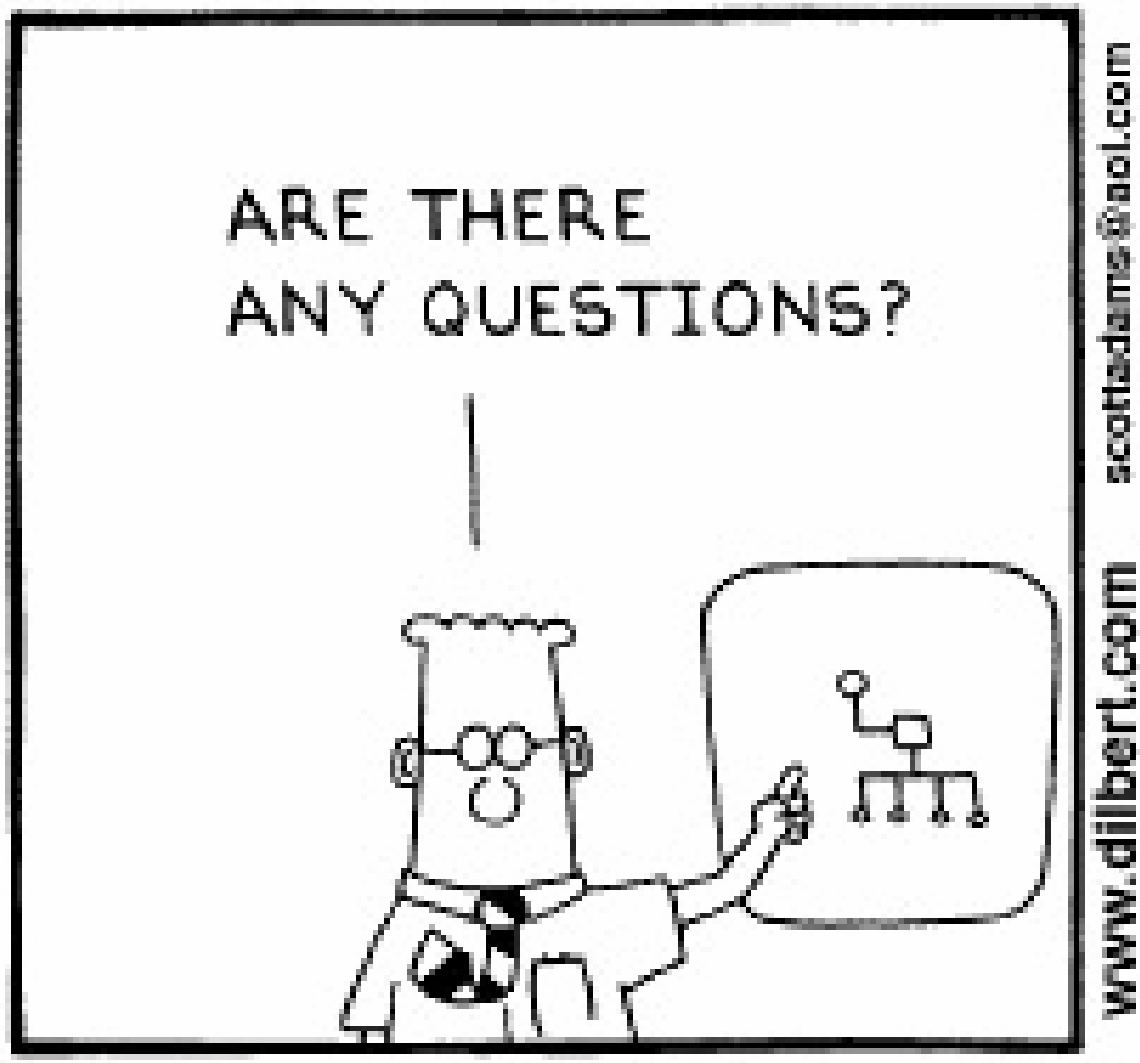
- Background and Motivation
- Performance Cloning – Central Idea
- Performance Cloning Framework
- Workload Profiling
- Algorithm for Clone Generation
- Analysis and Results
- Summary

Conclusions

- Technique that clones performance but hides functional meaning of code
  - Architects & Designers can get access to proprietary workloads
  - Foster benchmark sharing between industry and academia
  - Customers can make informed purchase decisions
- Evaluation of technique on embedded benchmarks is promising
  - Synthetic clone exhibits similar power/performance characteristics
  - Synthetic clone is a good proxy to original application

Challenges & Limitations

- Compiler technology is absorbed into the performance clone
  - Limited use for compiler studies
- Benchmark contains ISA specific embedded asm statements
  - Every embedded microprocessor designer cares about single ISA
  - Possibilities for true portability – virtual ISA, binary translation
- Abstract workload model simple by construction
  - Ability to perform “what-if” performance studies
  - Higher order models to capture complex dataflow


ARE THERE
ANY QUESTIONS?
www.dilbert.com scottadams@aol.com