AXCIS: Accelerating Architectural Exploration using Canonical Instruction Segments

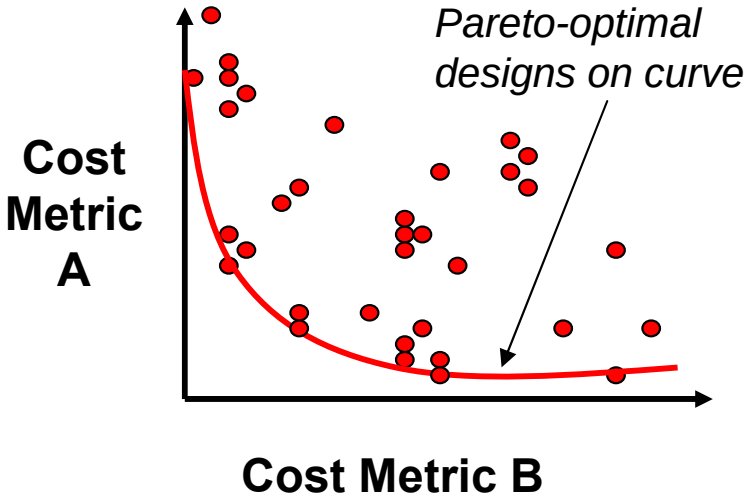
Rose Liu & Krste Asanović

Computer Architecture Group

MIT CSAIL

Simulation for Large Design Space Exploration

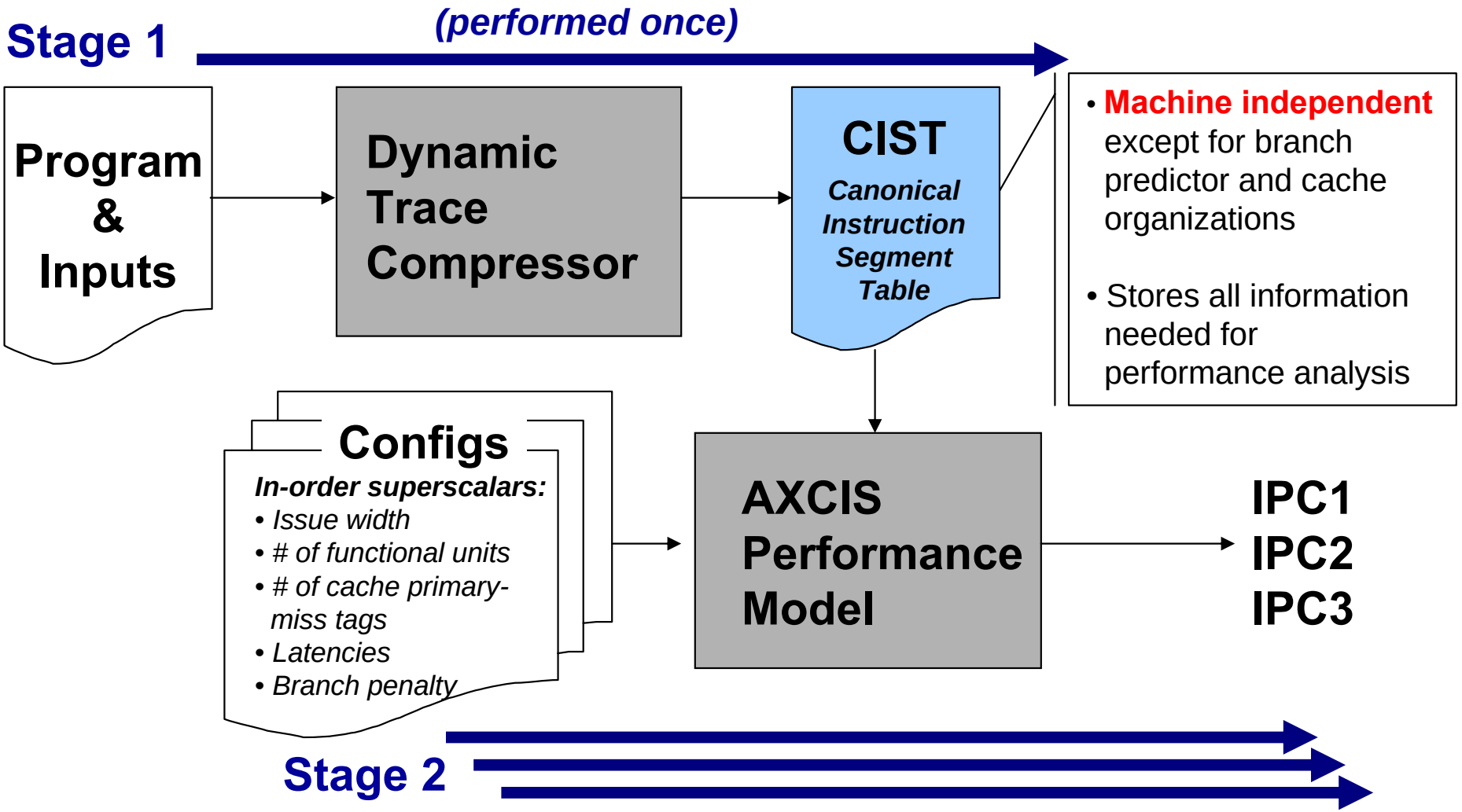
- **Large design space studies explore thousands of processor designs**
  - Identify those that minimize costs and maximize performance



- **Speed vs. Accuracy tradeoff**
  - Maximize simulation speedup while maintaining sufficient accuracy to identify interesting design points for later detailed simulation

CSAIL

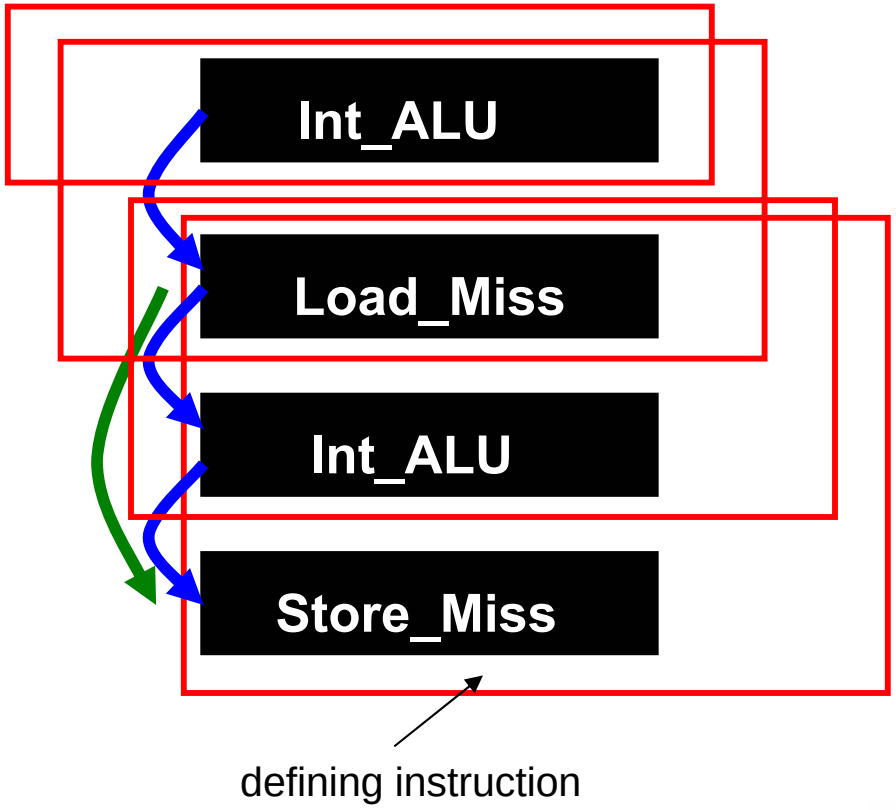
AXCIS Framework



Instruction Segments

- **An instruction segment captures all performance-critical information associated with a dynamic instruction**

```
addq
ldq   (cache miss)
subq
stq   (cache miss)
```

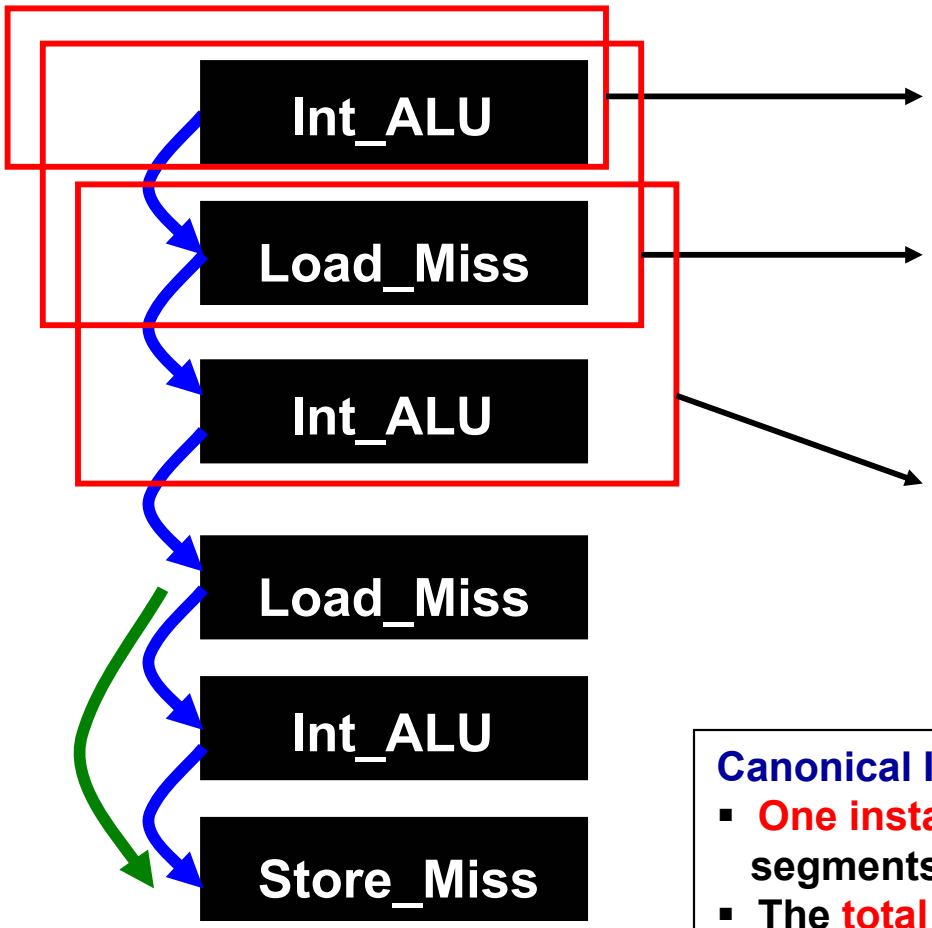


Dynamic Trace Compression

- **Repetition in program behavior such as loops, and code reuse cause instruction segments of different dynamic instructions to be canonically equivalent**
- **Ideal Compression Scheme:** (no loss in accuracy)
  - Compress two segments if they always experience the same stall cycles regardless of the machine configuration
  - Impractical to implement within the Dynamic Trace Compressor
- **Three compression schemes that approximate this ideal scheme**
  - Each selects a different tradeoff between accuracy and speedup
  - Our simplest scheme compresses segments that look the same (i.e. have the same length, instruction types, dependence distances, branch and cache behaviors)

CSAIL

Instruction Segments & CIST Example



Freq	Segment
1	Int_ALU
1	Int_ALU Load_Miss
1	Load_Miss Int_ALU

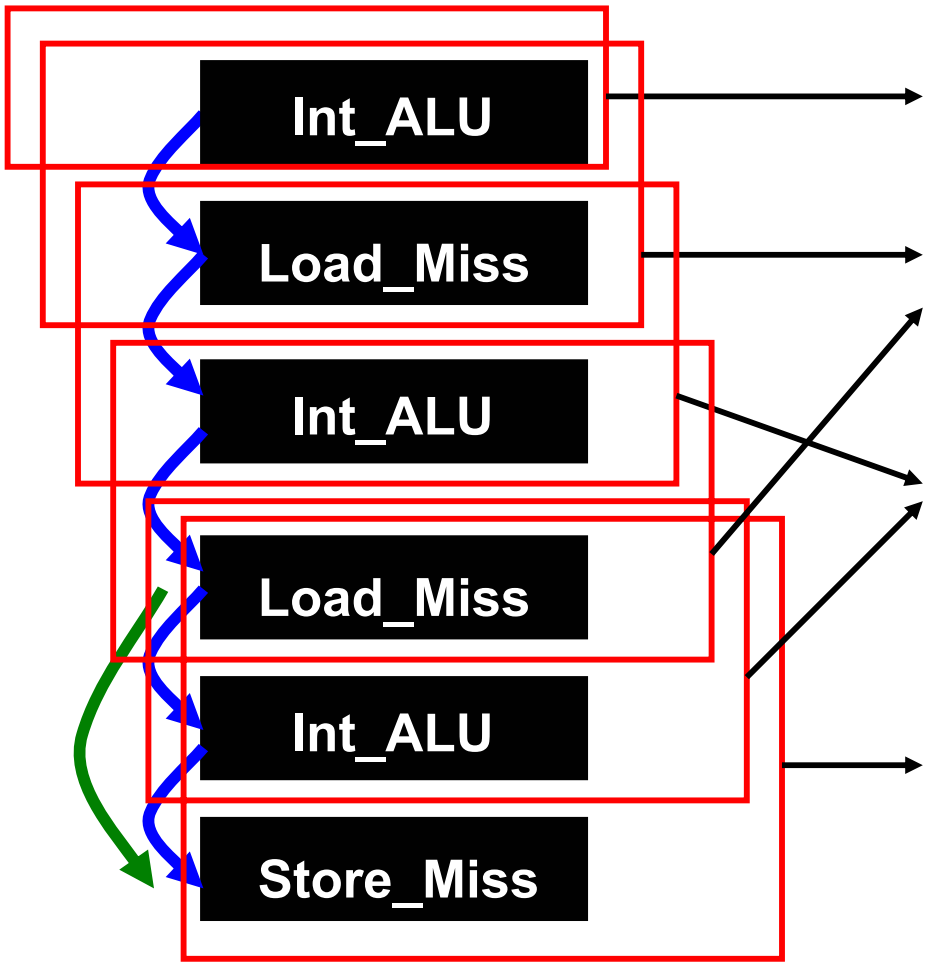
Total ins: 3

Canonical Instruction Segment Table (CIST) records:

- **One instance of each set of canonically equivalent segments and its frequency count**
- **The total dynamic instructions analyzed during trace compression**

CSAIL

Instruction Segments & CIST Example



Freq	Segment
1	Int_ALU
~~1~~ 2	Int_ALU Load_Miss
~~1~~ 2	Load_Miss Int_ALU
1	Load_Miss Int_ALU Store_Miss

Total ins: 6

AXCIS Performance Model

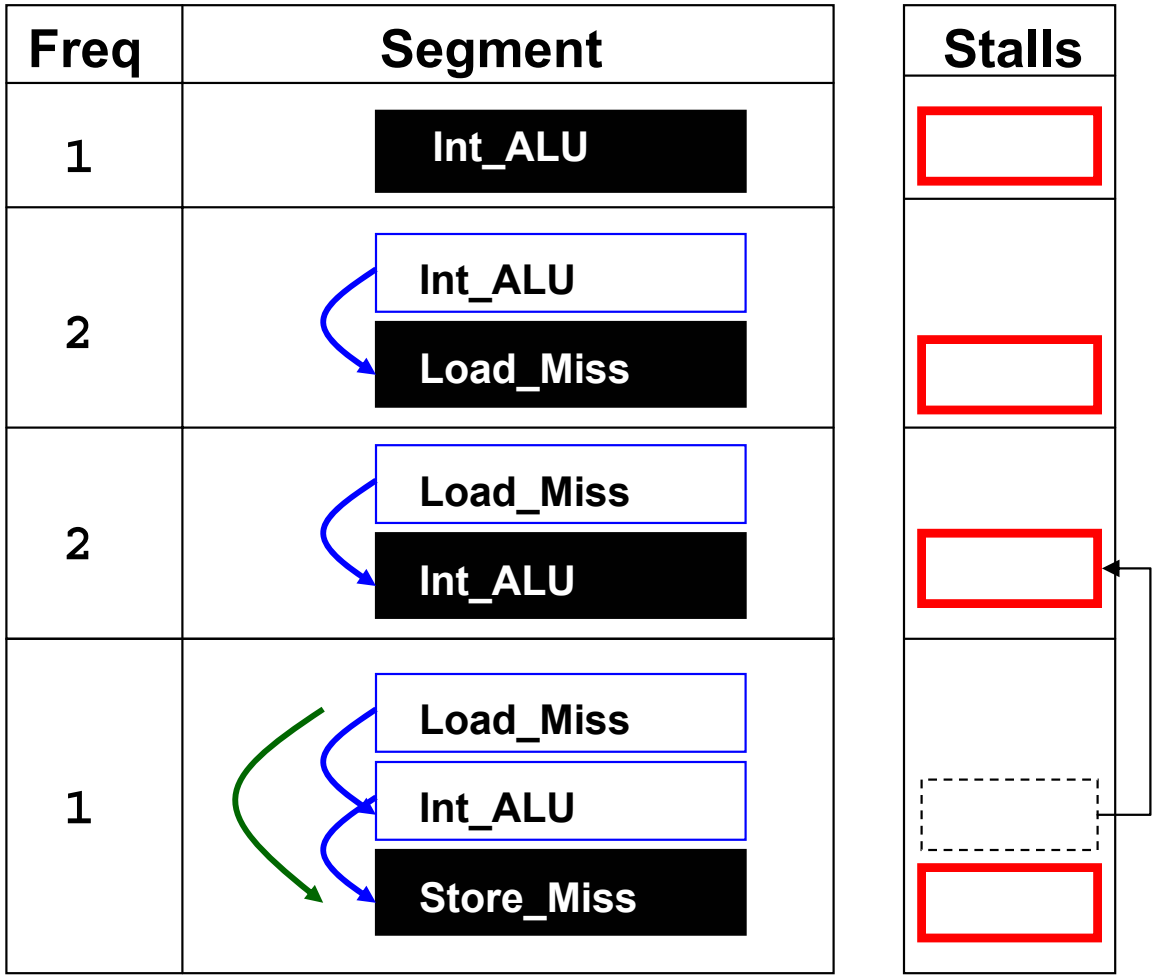
- **Methodology is independent of the compression scheme used to generate the CIST**
- **Calculates IPC using a single linear dynamic programming pass over CIST entries**

$$\mathbf{IPC} = \frac{\text{Total Ins}}{\text{Total Ins} + \mathbf{Total\ Effective\ Stalls}} = \frac{\text{Total Ins}}{\text{Total Cycles}}$$

$$\mathbf{Total\ Effective\ Stalls} = \sum_{i=1}^{\text{CIST Size}} \text{Freq}(i) * \text{EffectiveStalls}(\text{DefiningIns}(i))$$

CSAIL

Dynamic Programming Example



- Total work is proportional to the # of CIST entries
- Calculate the stalls of the defining instruction in each segment
- Look up stalls of other instructions in previous entries

Total ins: 6

Look up in previous segment
Calculate

CSAIL

Experimental Setup

- **Evaluated AXCIS against a baseline cycle accurate simulator on 24 SPEC2K benchmarks using their respective optimal compression schemes**

- **Evaluated AXCIS for:**
  - Accuracy:

$$\text{Absolute IPC Error} = \frac{|\ \text{AXCIS} - \text{Detailed Sim}\ |}{\text{Detailed Sim}} * 100$$

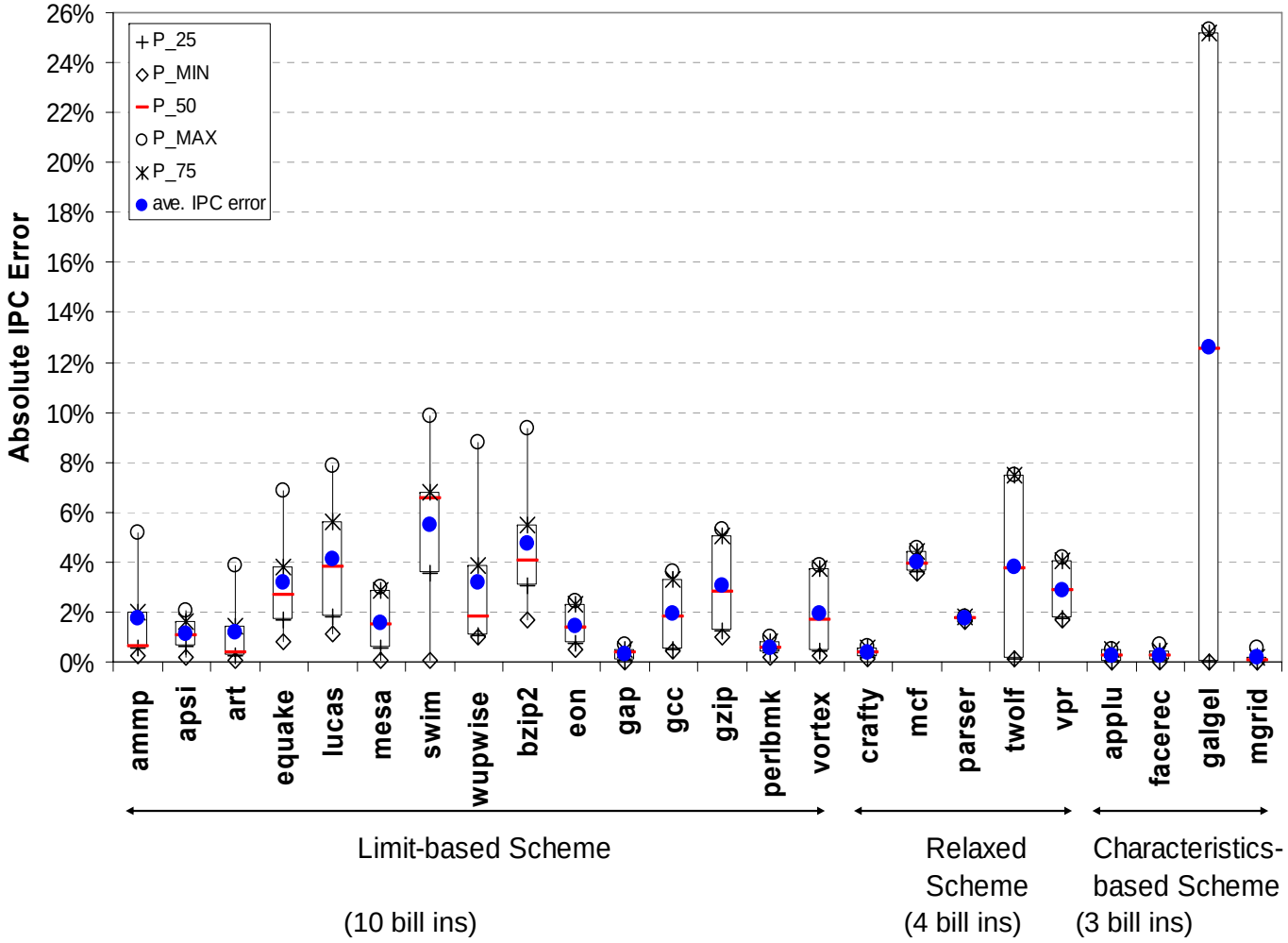
  - Speed: # of CIST entries, time in seconds

- **For each benchmark, simulated many configurations that span a large design space:**
  - Issue width: {1, 4, 8}, # of functional units: {1, 2, 4, 8},
    Memory latency: {10, 200 cycles},
    # of primary miss tags in non-blocking data cache: {1, 8}

CSAIL

Results: Accuracy

Distribution of IPC Error in quartiles

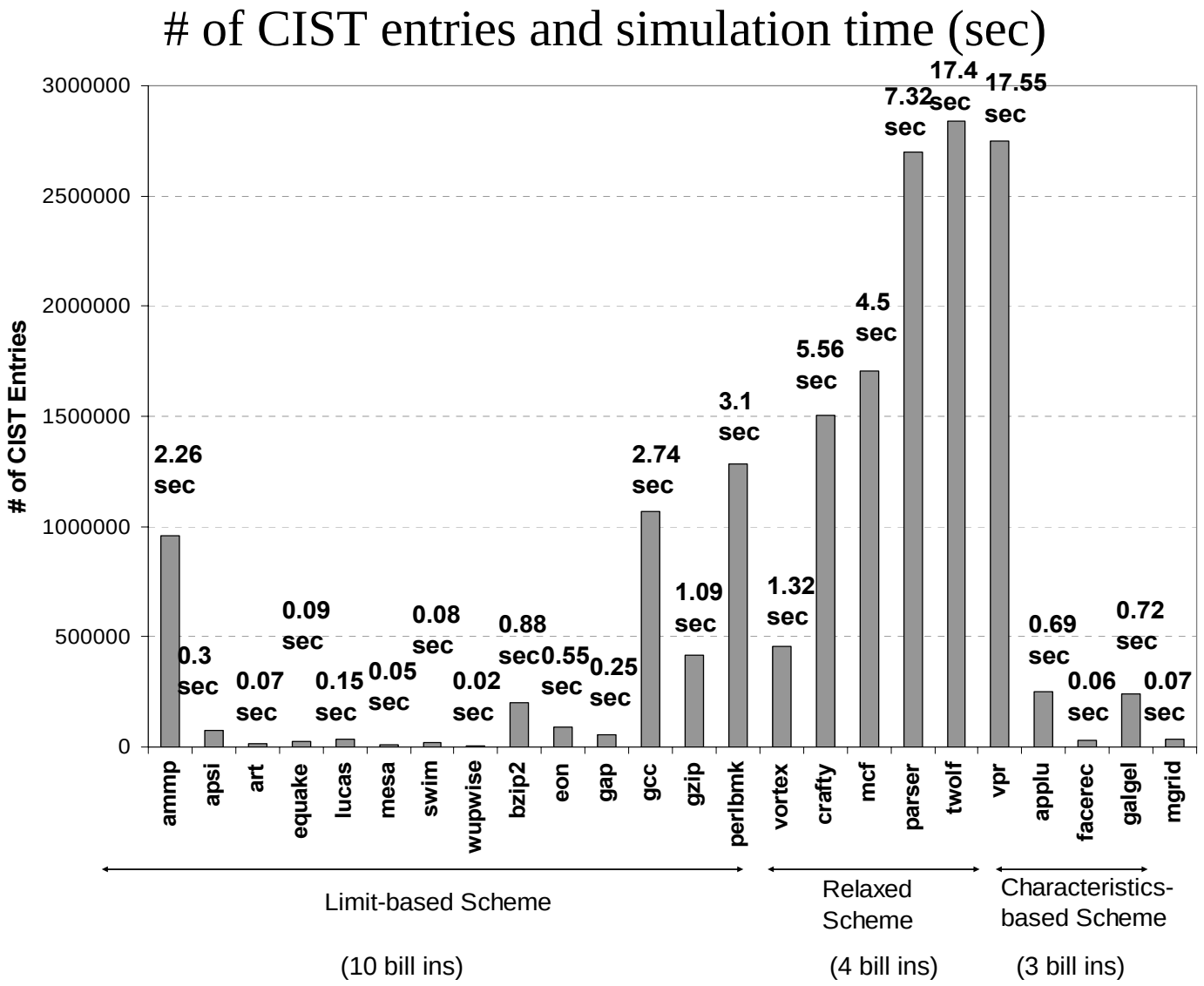


Average Absolute IPC Error = 2.6 %

Average Error Range = 4.4%

CSAIL

Results: Speed



- AXCIS is over **4 orders of magnitude faster** than detailed simulation
- While detailed simulation takes hours to simulate billions of instructions, **AXCIS takes seconds**

CSAIL

Conclusion

- **AXCIS is a fast, accurate, and flexible tool for design space exploration**
- **AXCIS**
  - Over four orders of magnitude faster than detailed simulation
  - Highly accurate across a broad range of designs
  - Predicts performance as well as buffer occupancies
- **Future Work**
  - More general compression schemes
  - Support out-of-order processors

CSAIL