APT-GET: Profile-Guided Timely Software Prefetching

Presented by Group 1:
Amirali Ebrahimzadeh, Kalab Assefa, Kidus Simegne, Sitota Mersha

APT-GET

- Published at EuroSys ’22
- By Saba Jamilan, Tanvir Ahmed, Grant Ayers, Baris Kasikci, Heiner Litz

Prefetching

- Over 60% of all processor cycles stall due to cache misses
- Prefetching reduces stalls
- Software prefetch = compiler-inserted hint (e.g., llvm.prefetch) to fetch a cache line early
- Irregular access patterns pose challenges
  - E.g. indirect array access of the form $A[B[i]]$

Challenging Benchmark

```
for(e=0; e<OUTER; e++)
 for(i=0; i<INNER; i++)
   // prefetch(&T[BO[e]+BI[i+7]]);
   val = T[BO[e]+BI[i]];
   do_work(val);
```

Problem Statement

- Static software prefetching before APT-GET either:
  - Too conservative ⇒ miss opportunities
  - Too aggressive ⇒ pollute caches, increase memory traffic
- Objective: Prefetch timely, enough ahead to hide latency, but not too early
- Dynamic prefetching via profile data!

APT-GET

- Fully automated approach for inserting software prefetches
- Uses runtime profiles based on hardware performance counters to determine
  - Which loads to prefetch
  - Optimal prefetch distance: How far ahead to prefetch
  - Optimal prefetch injection site: Where to insert prefetch

Hardware Performance Counters

- Perf
  - sample loads that cause frequent LLC misses
- Last Branch Record (LBR)
  - control flow timing
- Precise Event-Based Sampling (PEBS)
  - data access timing

Profiling Delinquent Loads

- Measure performance
  - Trip count
  - Execution time

$$IC\_latency \times D = MC\_latency$$

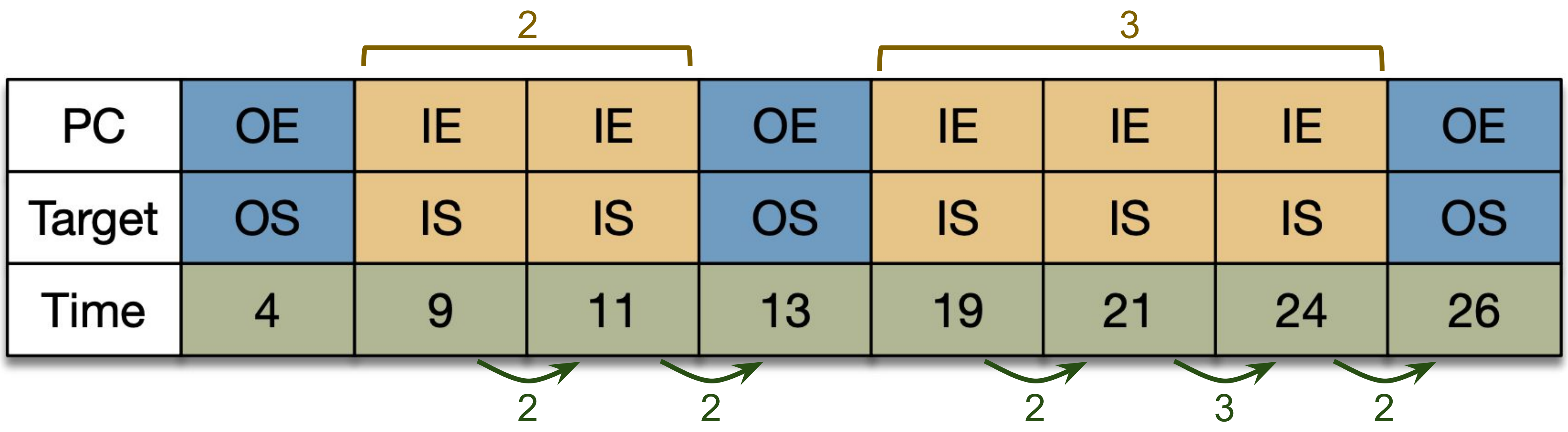
IC = Instruction Component, all non-load instructions

MC = Memory Component, loads causing frequent cache misses

D = prefetch distance (D)

Example LBR Profiling

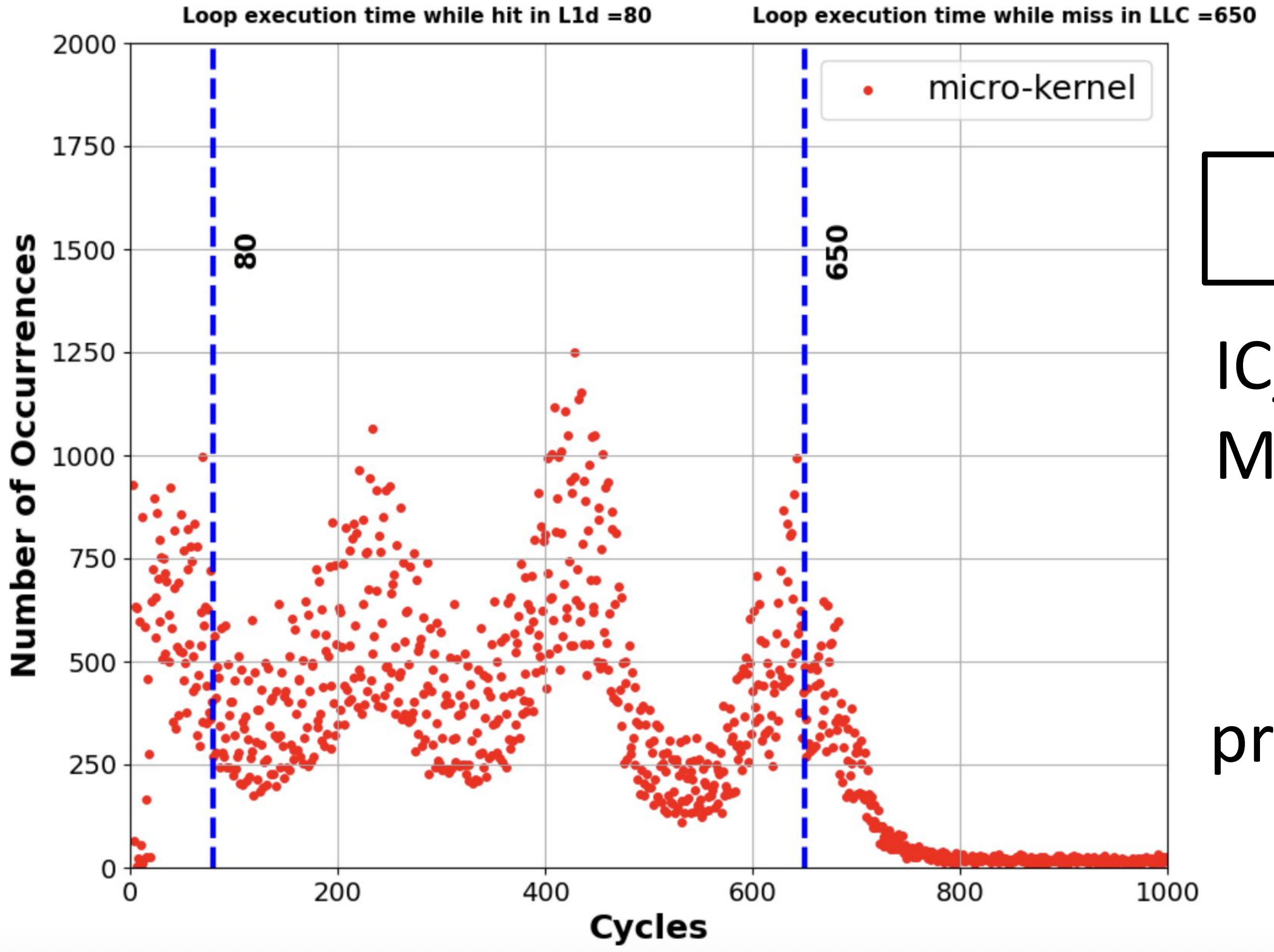
PC	OE	IE	IE	OE	IE	IE	IE	OE
Target	OS	IS	IS	OS	IS	IS	IS	OS
Time	4	9	11	13	19	21	24	26



$$\text{Average Execution time} = \frac{(2+2+2+3+2)}{5} = 2.2$$

$$\text{Average Trip count} = \frac{(2+3)}{2} = 2.5$$

Optimal Prefetch Distance



$$IC\_latency \times D = MC\_latency$$

IC_latency = 80 cycles

MC_latency = 650 cycles - IC_latency

= 570 cycles

prefetch_distance(D) = 570/80 ≈ 7

Optimal Prefetch Injection Site

- Each prefetch should run k iterations ahead of the demand load
- Prefetch should complete before the demand load executes (timeliness).
- Prefetch should not be issued so early that prefetched lines are evicted before use (usefulness).
- Need to balance timeliness vs. usefulness

$$loop\_trip\_count \times k < D$$

Example: Nested Loop Prefetch

```
for(e=0; e<OUTER; e++)
 for(i=0; i<INNER; i++)
   val = T[BO[e]+BI[i]];
```

Trip Count
Inner = 100
Outer = 5

```
for(e=0; e<OUTER; e++)
 for(i=0; i<INNER; i++)
  prefetch(&T[BO[e]+BI[i+7]]);
  val = T[BO[e]+BI[i]];
```

Trip Count
Inner = 2
Outer = 100

```
for(e=0; e<OUTER; e++)
 prefetch(&T[BO[e+1]+BI[0]]);
 prefetch(&T[BO[e+1]+BI[1]]);
 for(i=0; i<INNER; i++)
   val = T[BO[e]+BI[i]];
```

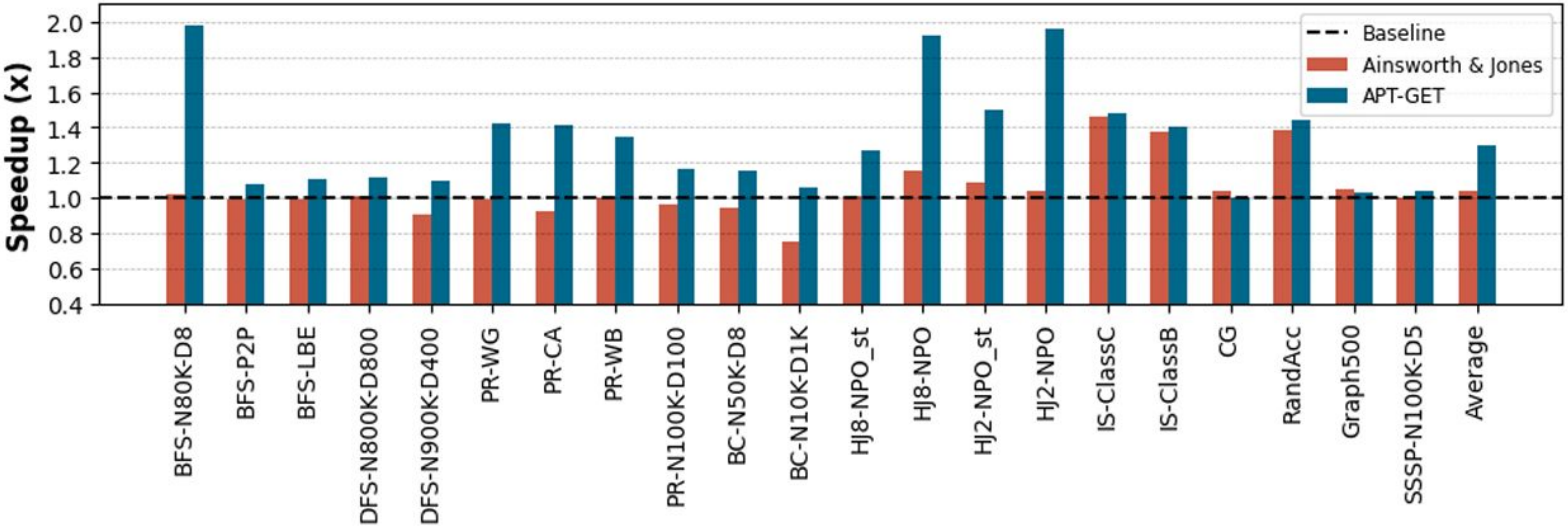
Compiler Prefetch Injection (LLVM Pass)

1. Map sampled PCs to IR instructions
2. Backward Dependency analysis
   - Track instructions required to compute load address
   - Include loop induction variables (PHINodes)
3. Insert prefetch instructions
   - Respect computed prefetch distance
   - Preserve program correctness

Experimental Setup

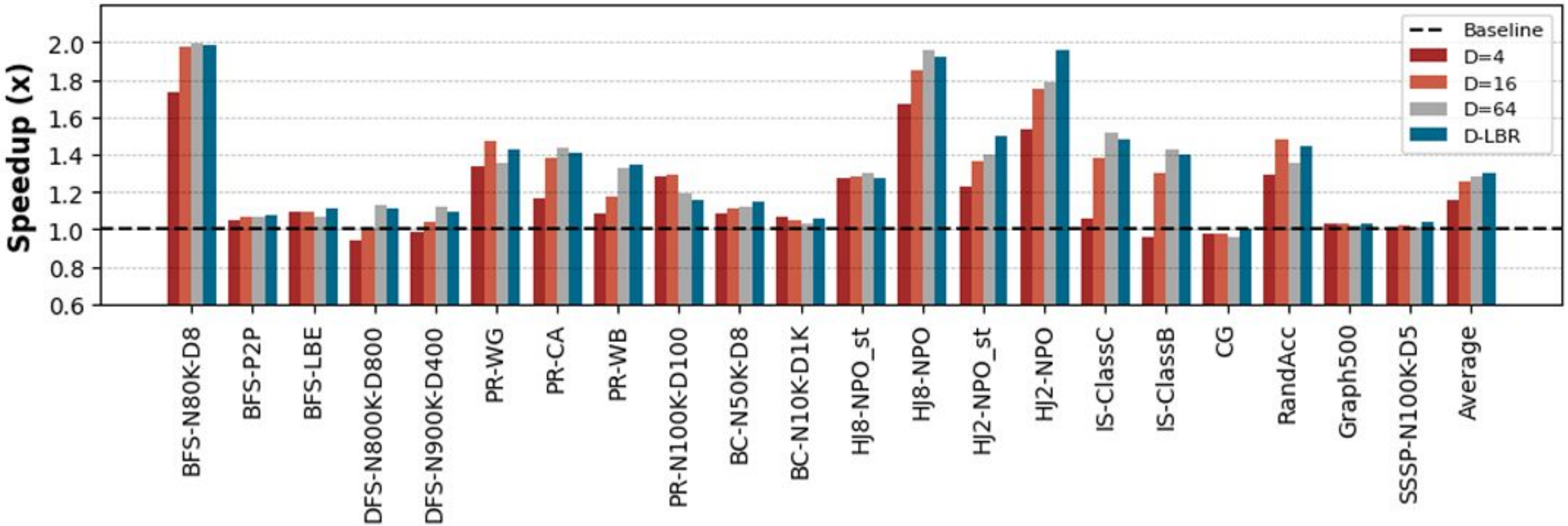
- The benchmark is *-o3* pass and Ainsworth & Jones (state of the art static prefetcher)
- 10 real world applications
- They collected the data from real internet datasets.

Performance Improvement



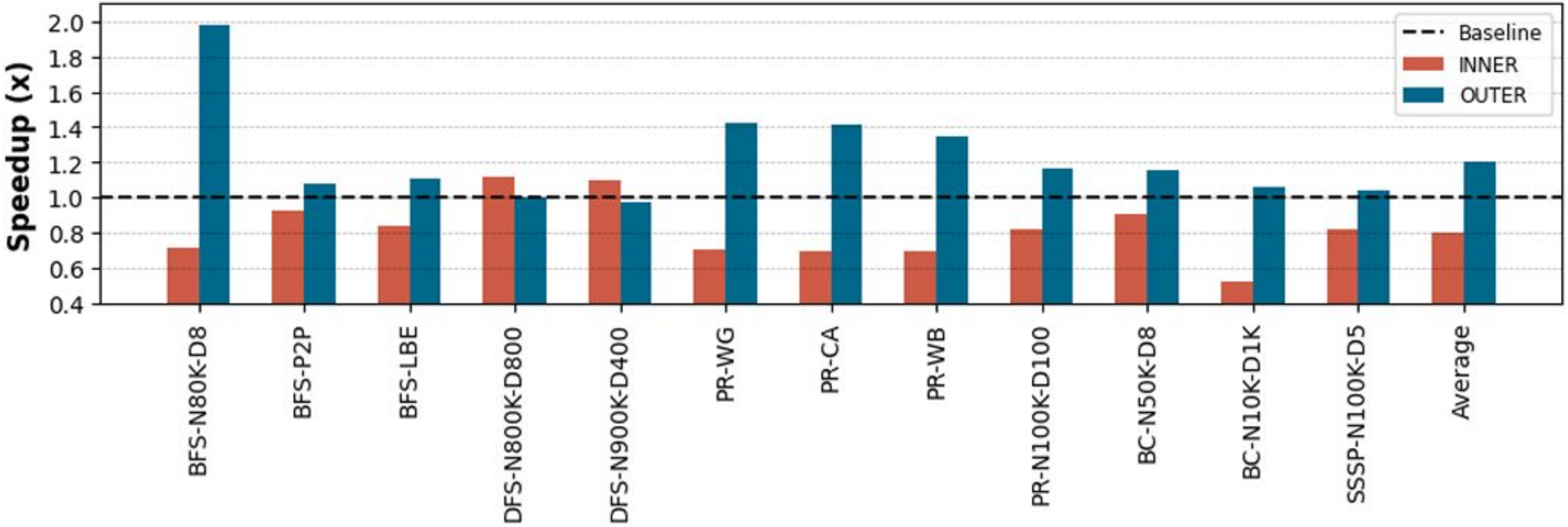
- APT-GET achieves **1.30× average speedup**
- The state of the art (Ainsworth & Jones) is 1.04×

Prefetch Distance Optimization



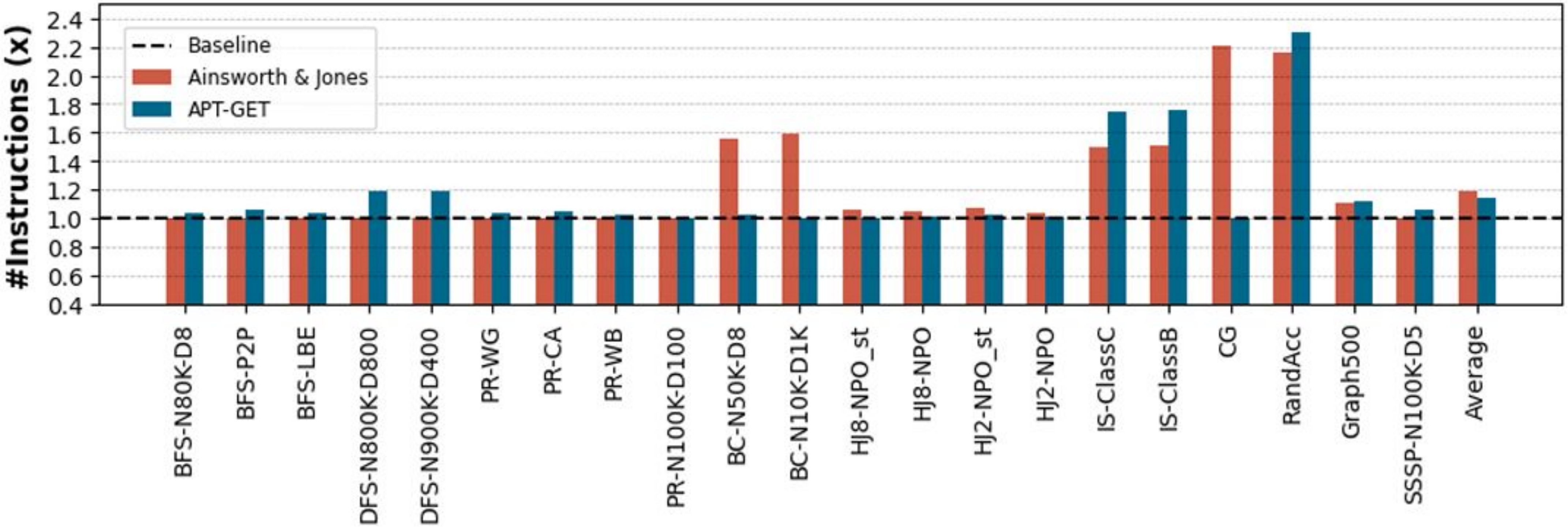
- **Dynamic prefetch distance** is better than static prefetch distances.

Prefetch Injection Site Optimization



- Inner loop injection is rarely optimal
- We need **dynamic prefetch injection site**

The Cost



- APT-GET executes **0.05× fewer** additional prefetch instructions than Ainsworth & Jones (1.14× vs. 1.19×)

Limitations & Our Thoughts

- The benchmarks “IS” and “RandAcc” the instruction overhead is too much.
  - Maybe they could use conditional prefetch slice injection.
- LBR size limits for high Inner Loop Iterations and Complex Inner Loops.
  - Maybe they could use a different perf counter.

Conclusion

- **Problem**: Static software prefetching techniques fail to generate timely prefetches.
- **Solution**: using hardware Last Branch Record to dynamically characterize loops and find the optimal prefetch-distance and injection site.
- **Result:** speedups up to 1.98x over static methods.

UNIVERSITY OF MICHIGAN

Questions?