# Fetch Me If You Can

## Evaluating CPU Cache Prefetching and Its Reliability on High Latency Memory

*Fabian Mahling*
*Marcel Weisgut*
*Tilmann Rabl*

*@DaMoN 2025*

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Design IT.
Create Knowledge.

www.hpi.de

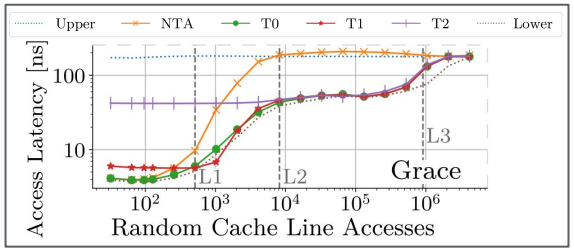
# Contributions

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics

# Contributions

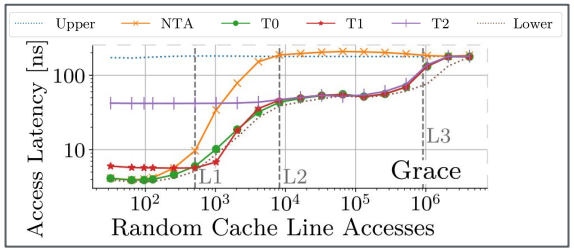
1. Microbenchmarks identifying CPU prefetching characteristics



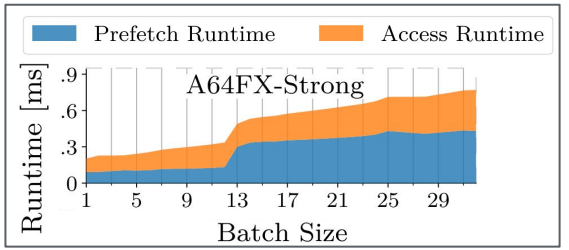

Localities

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics



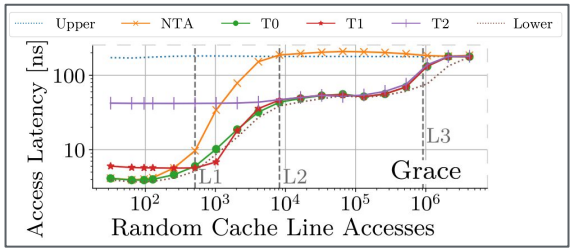

Localities



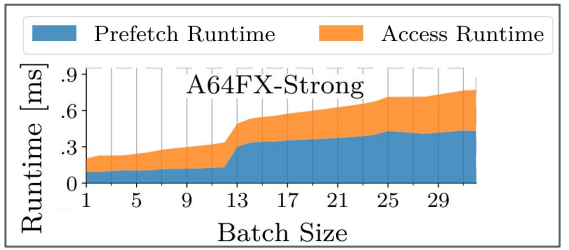

Reliability

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics




Localities

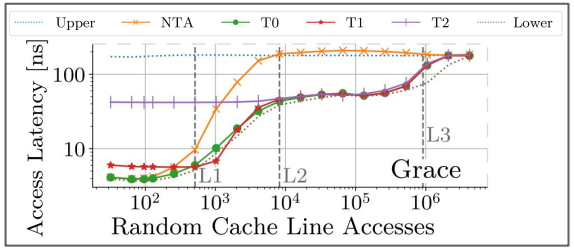



Reliability

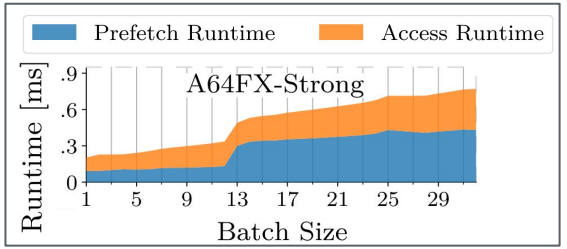
2. Prefetching performance evaluation on heterogeneous memory

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics

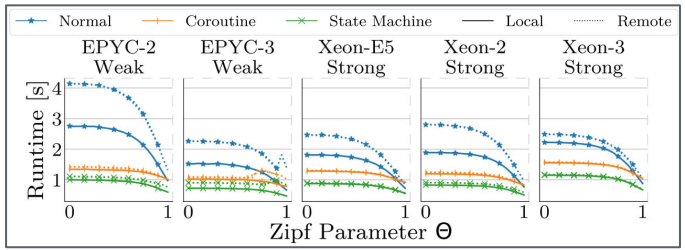



Localities




Reliability

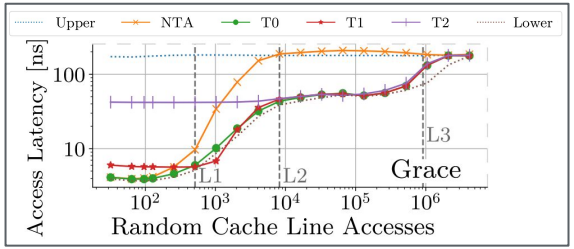
2. Prefetching performance evaluation on heterogeneous memory



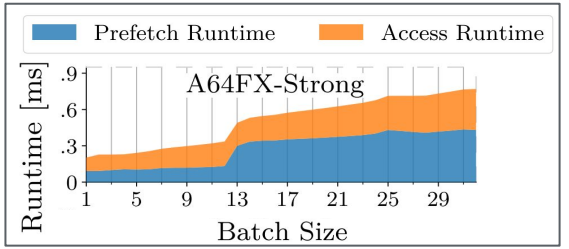

Binary Search

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics

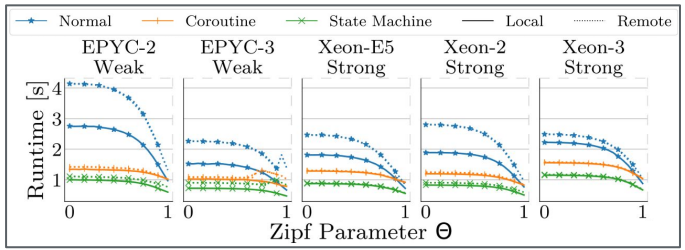



Localities

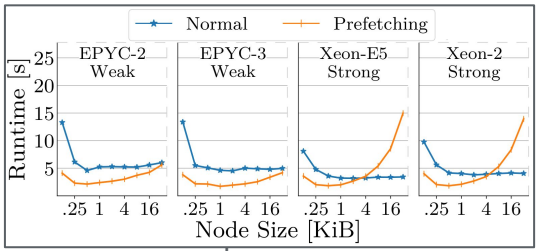



Reliability

2. Prefetching performance evaluation on heterogeneous memory



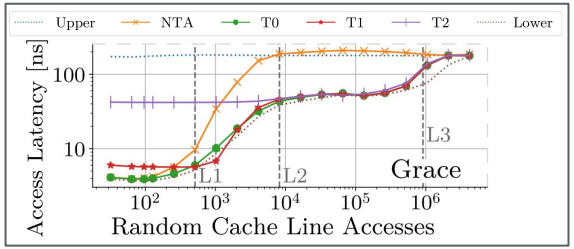

Binary Search



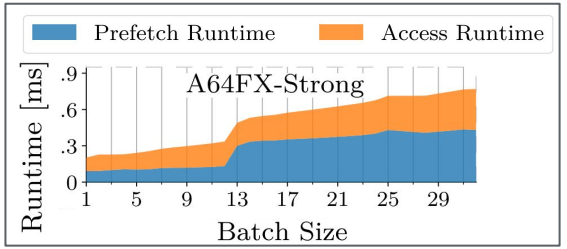

B$^+$-Tree

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics

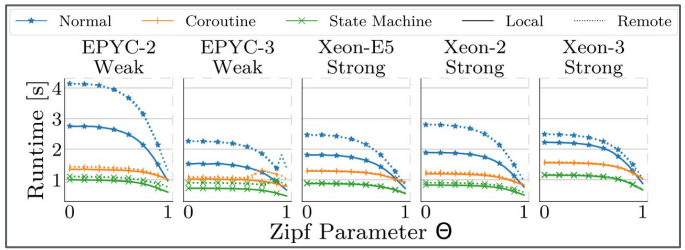



Localities

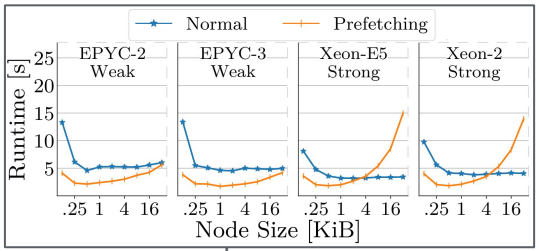



Reliability

2. Prefetching performance evaluation on heterogeneous memory




Binary Search

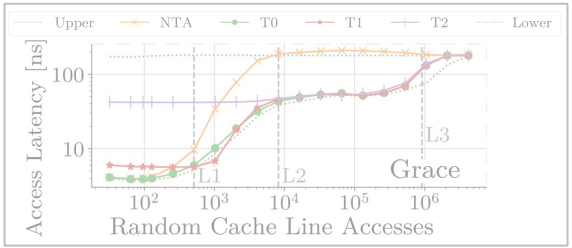



$B^+$-Tree

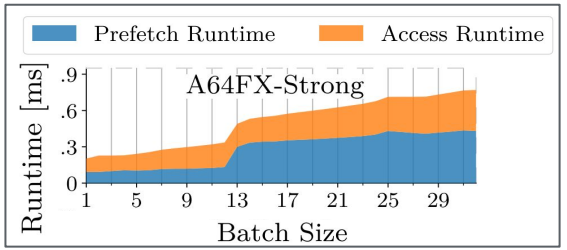
3. Guidelines on when and how to use prefetching

# Contributions

1. Microbenchmarks identifying CPU prefetching characteristics

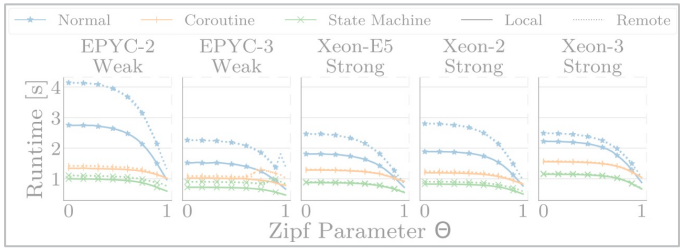



Localities

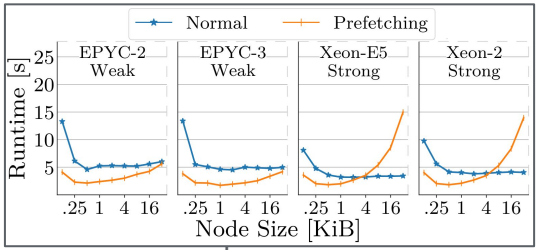



Reliability

2. Prefetching performance evaluation on heterogeneous memory



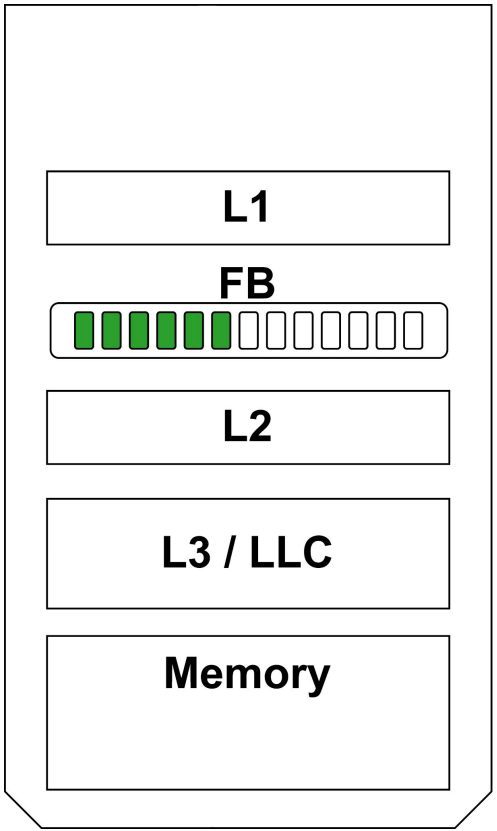

Binary Search




$B^+$-Tree

3. Guidelines on when and how to use prefetching

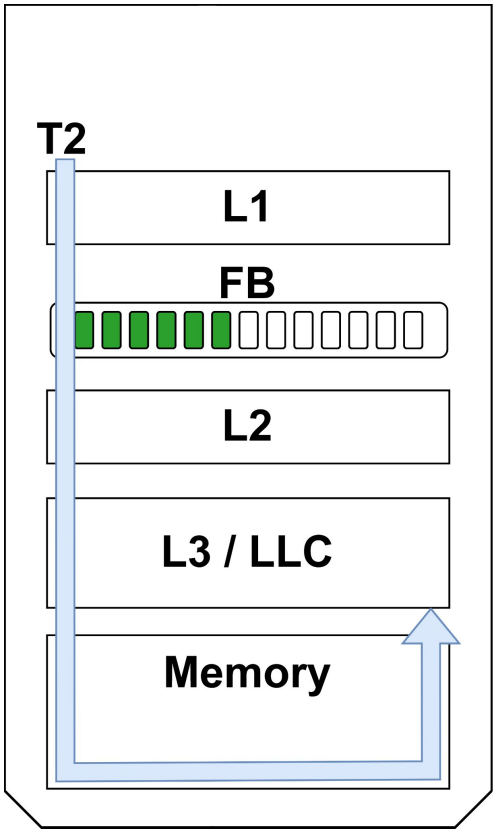
# Software Prefetching




Based on [*Kühn et al. 2024*]

- Asynchronous data movement to cache
- Locality specifies target cache level:

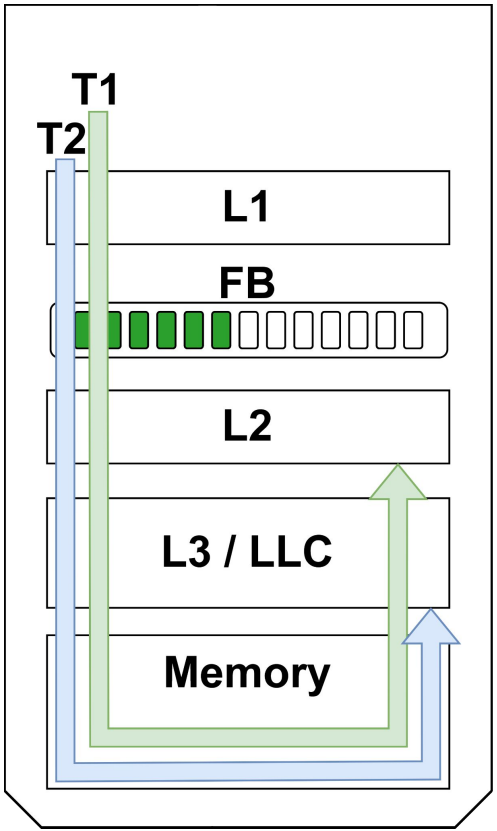
# Software Prefetching




Based on [*Kühn et al. 2024*]

- Asynchronous data movement to cache
- Locality specifies target cache level:
  - *T2* → *L3*

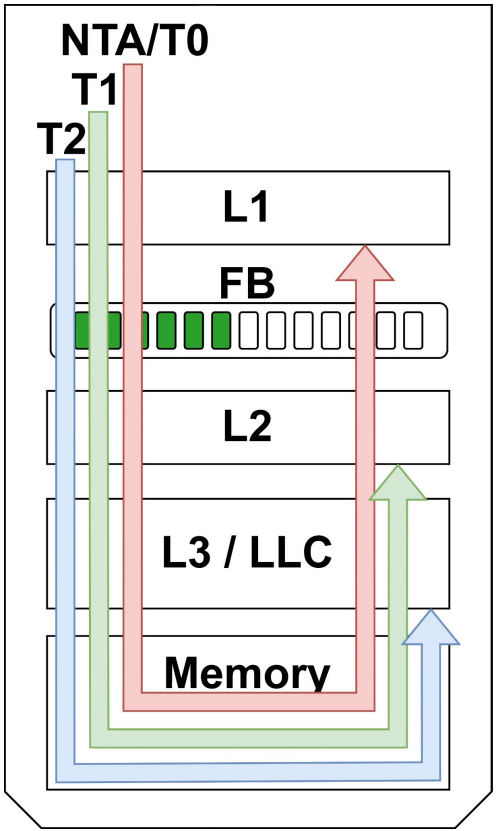
# Software Prefetching




Based on [*Kühn et al. 2024*]

- Asynchronous data movement to cache
- Locality specifies target cache level:
  - *T2* → *L3*
  - *T1* → *L2*

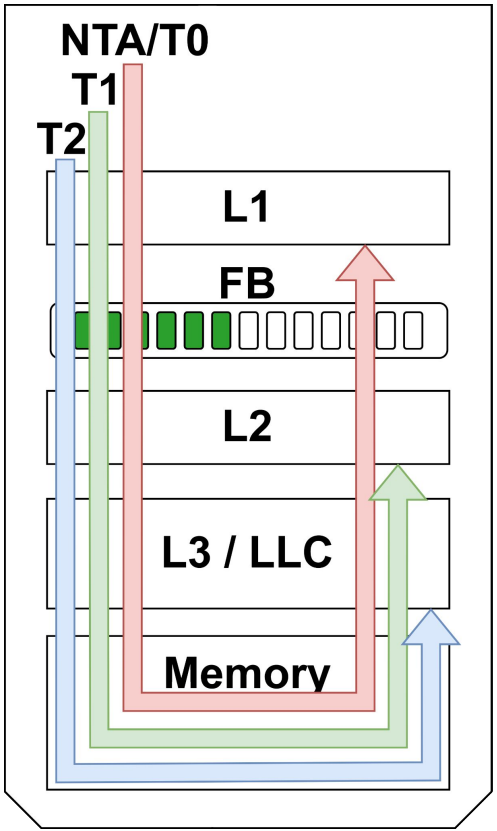
# Software Prefetching




Based on [*Kühn et al. 2024*]

- Asynchronous data movement to cache
- Locality specifies target cache level:
  - *T2* → *L3*
  - *T1* → *L2*
  - *T0 & NTA* → *L1*

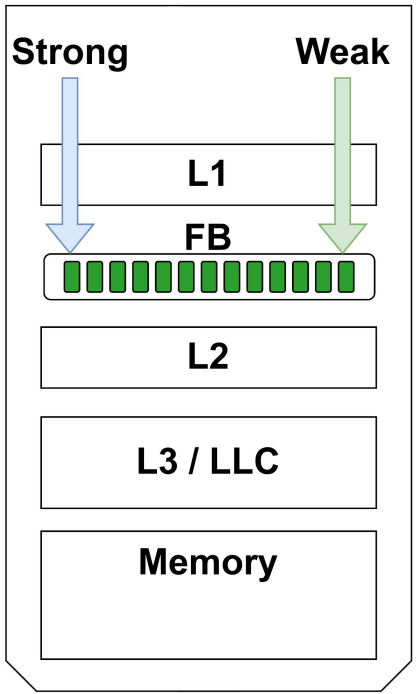
# Software Prefetching




Based on [*Kühn et al. 2024*]

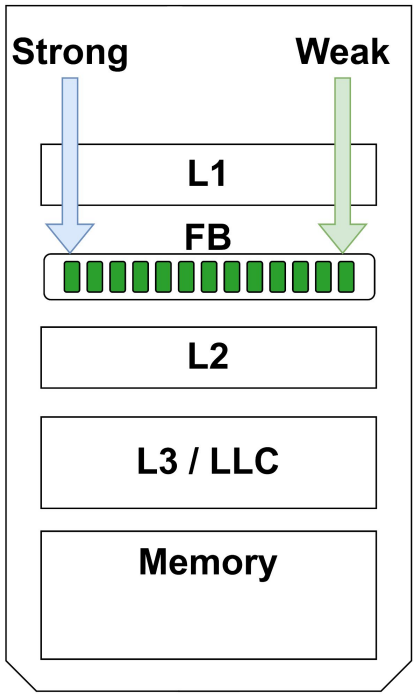
- Asynchronous data movement to cache
- Locality specifies target cache level:
  - *T2* → *L3*
  - *T1* → *L2*
  - *T0 & NTA* → *L1*
- Hides memory access latencies through data movement ahead of the actual access

# Prefetching Reliability



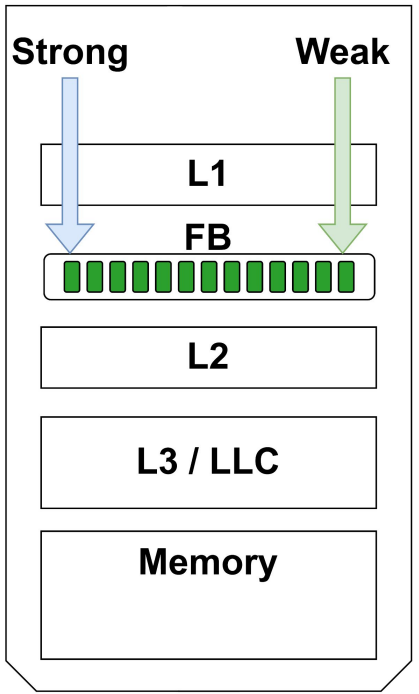
# Prefetching Reliability

- **Strong** or **Weak** prefetching reliability



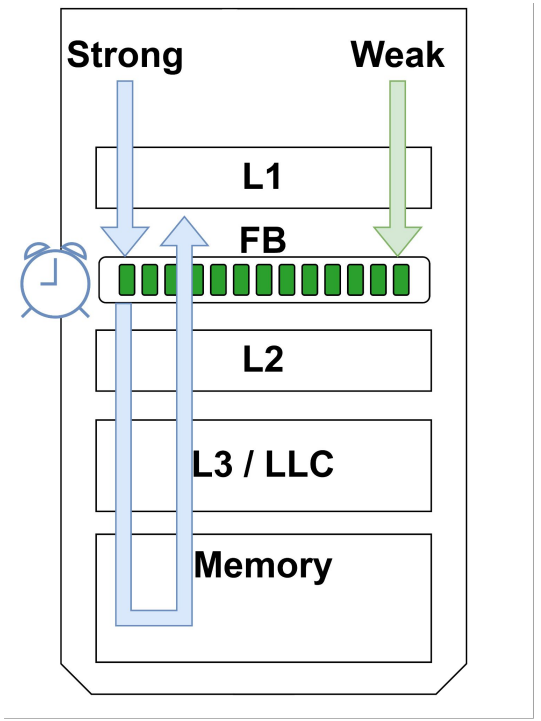
# Prefetching Reliability

- **Strong** or **Weak** prefetching reliability
- Different behavior for sparse hardware resources:



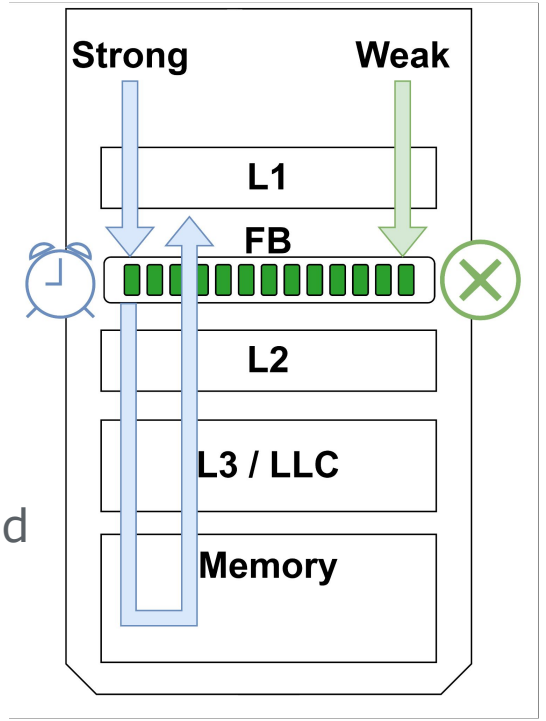
# Prefetching Reliability

- **Strong** or **Weak** prefetching reliability
- Different behavior for sparse hardware resources:
  - **Strong** *"Fetch Me!":*
    - Wait for resource to be freed up

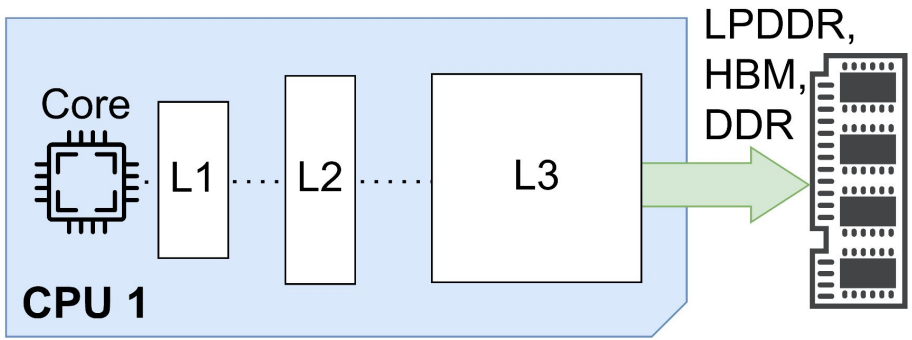


# Prefetching Reliability

- **Strong** or **Weak** prefetching reliability
- Different behavior for sparse hardware resources:
  - **Strong** *"Fetch Me!":*
    - Wait for resource to be freed up
  - **Weak** *"Fetch Me, If You Can!":*
    - Drop prefetches when resources are occupied

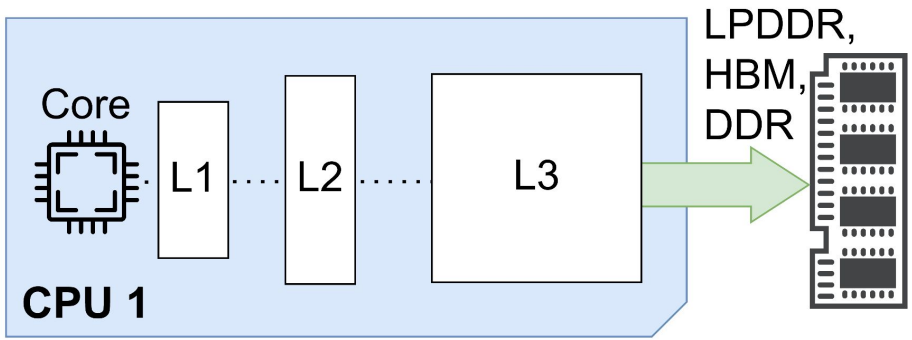


# Used Systems




Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

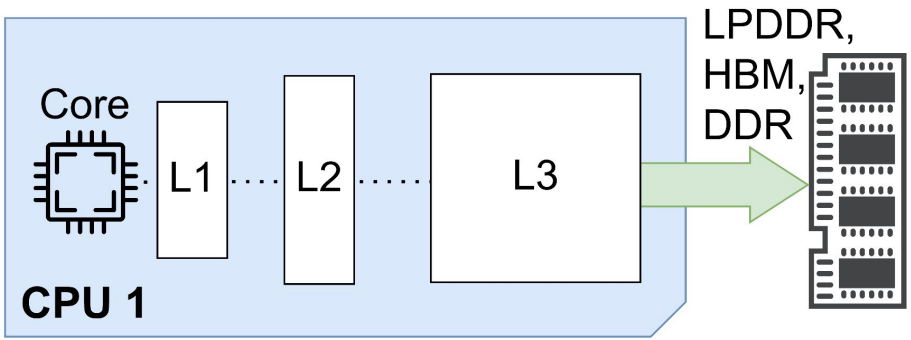
# Used Systems




- Data Placement:

Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

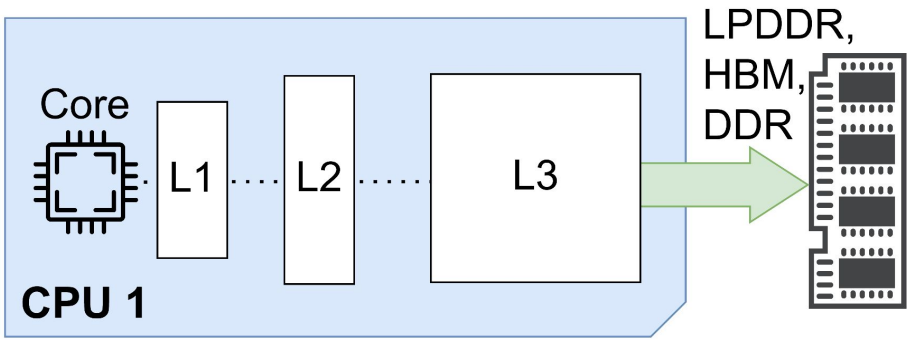
# Used Systems




- Data Placement:
  - **Local**: lowest latency

Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

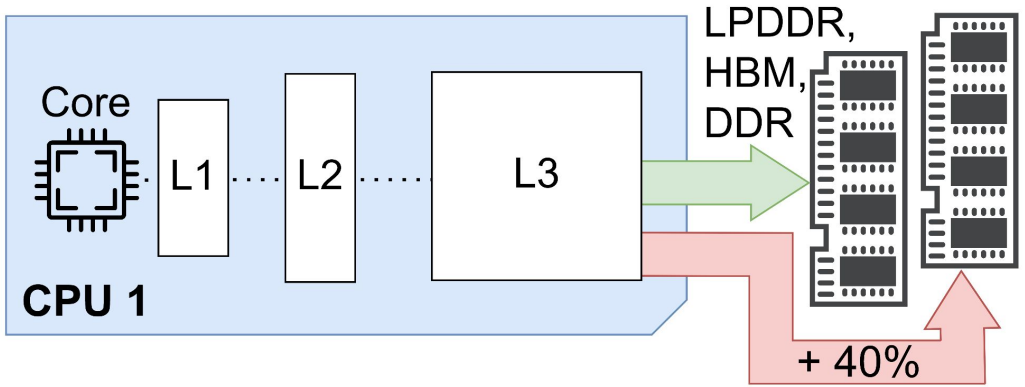
# Used Systems




- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency

Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

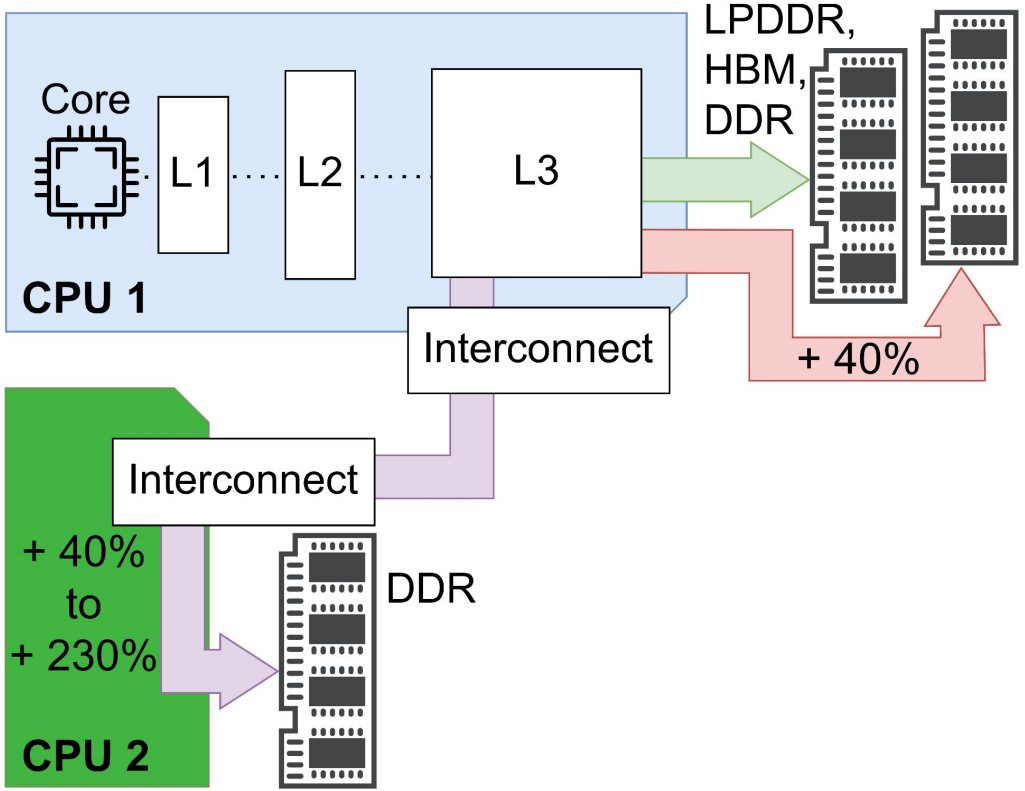
# Used Systems




- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency

Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

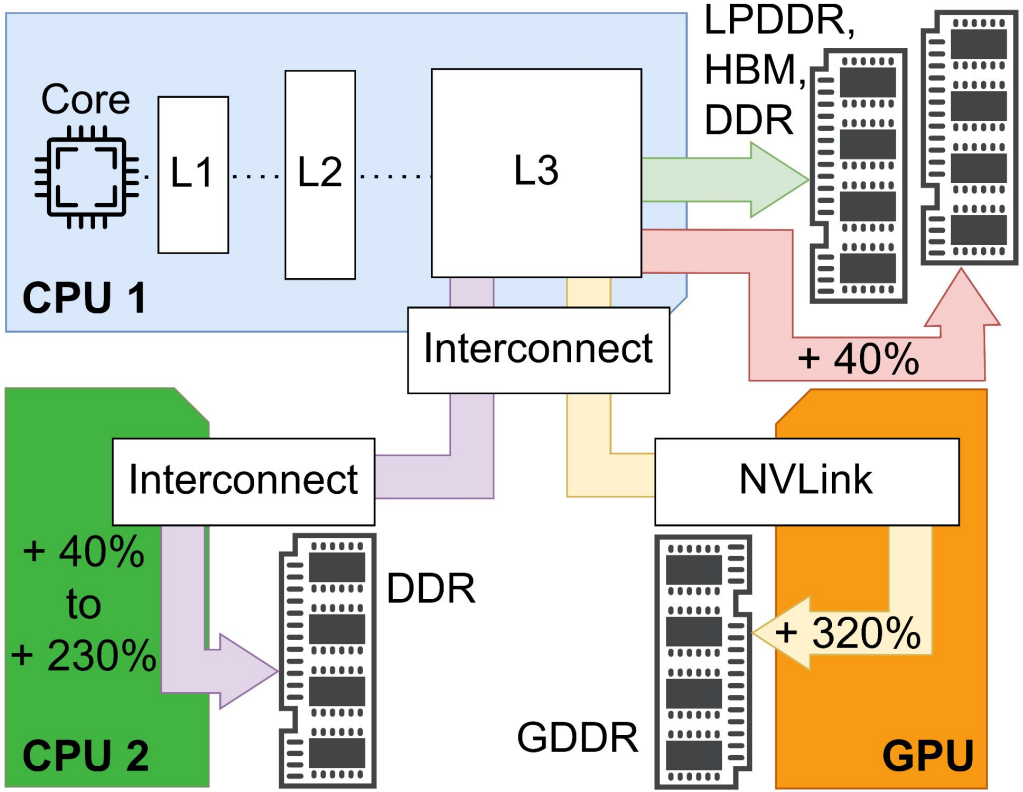
# Used Systems




- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency

Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

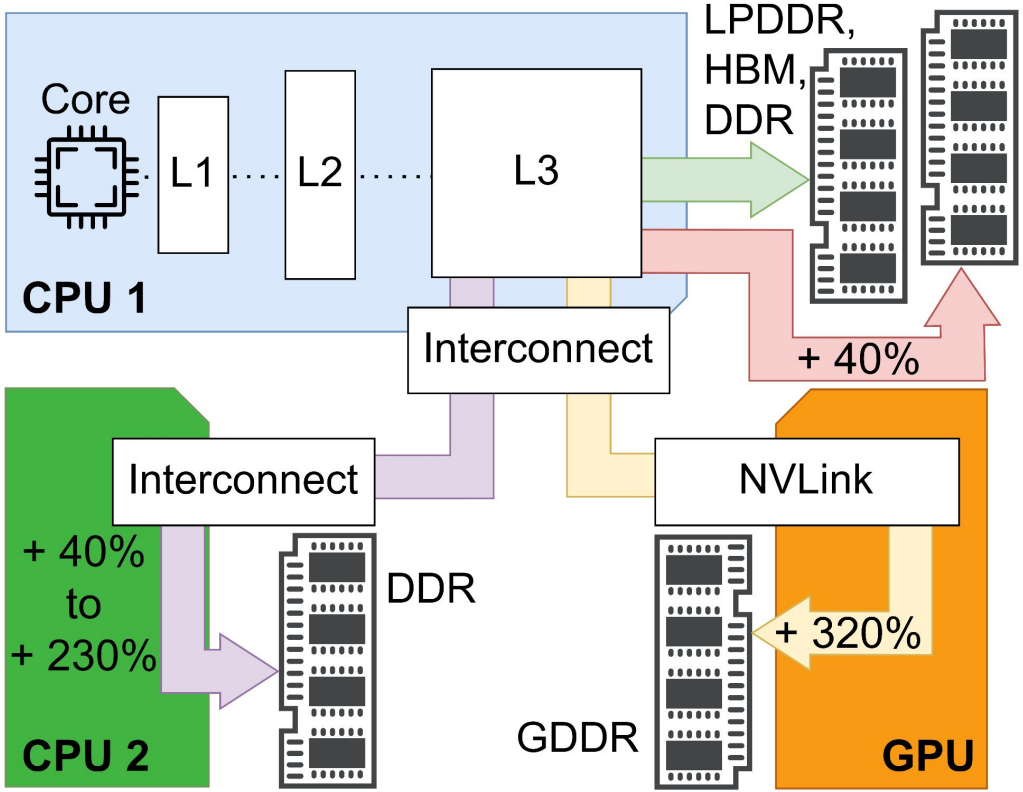
# Used Systems




Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency

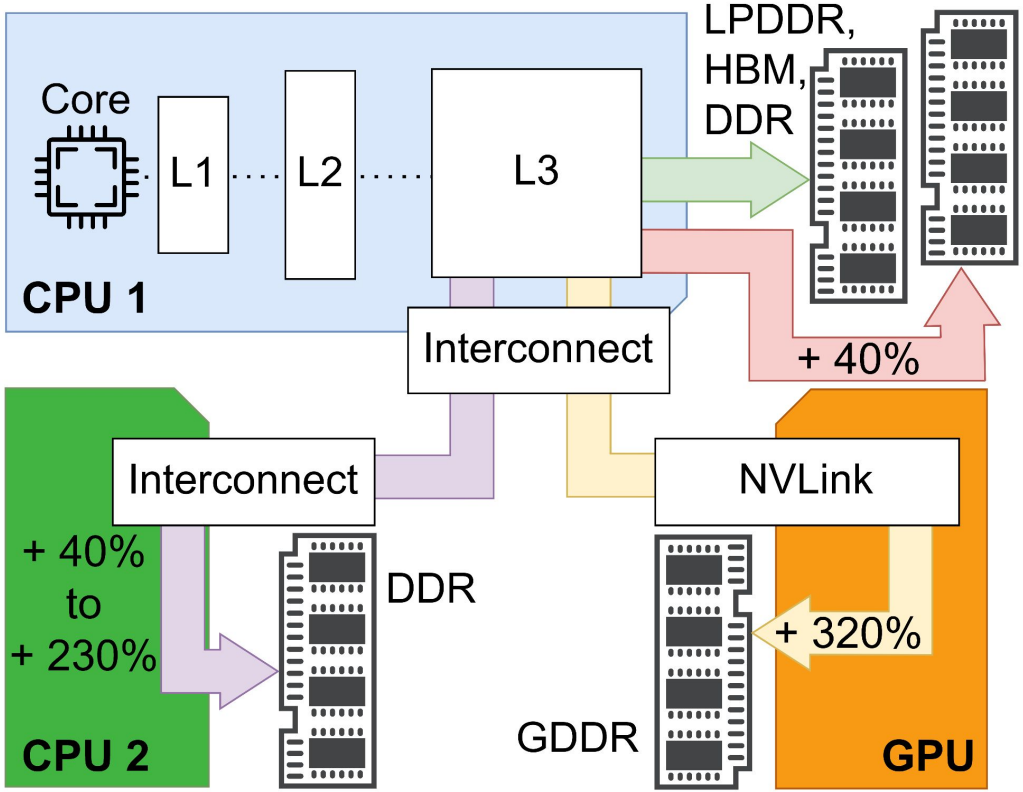
# Used Systems




Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency
- Different memory access paths have varying access latencies.

# Used Systems




Based on [*Vuppalapati et al. 2024, Werner et al. 2025*]

- Data Placement:
  - **Local**: lowest latency
  - **Remote**: highest latency
- Different memory access paths have varying access latencies.
- Higher access latencies cause performance loss.

*Research Question*

**Can Software Prefetching hide increased memory access latencies, independently of the given prefetching reliability?**

# Prefetching Reliability - Microbenchmark

# Prefetching Reliability - Microbenchmark

- Two phase microbenchmark:

# Prefetching Reliability - Microbenchmark

- Two phase microbenchmark:

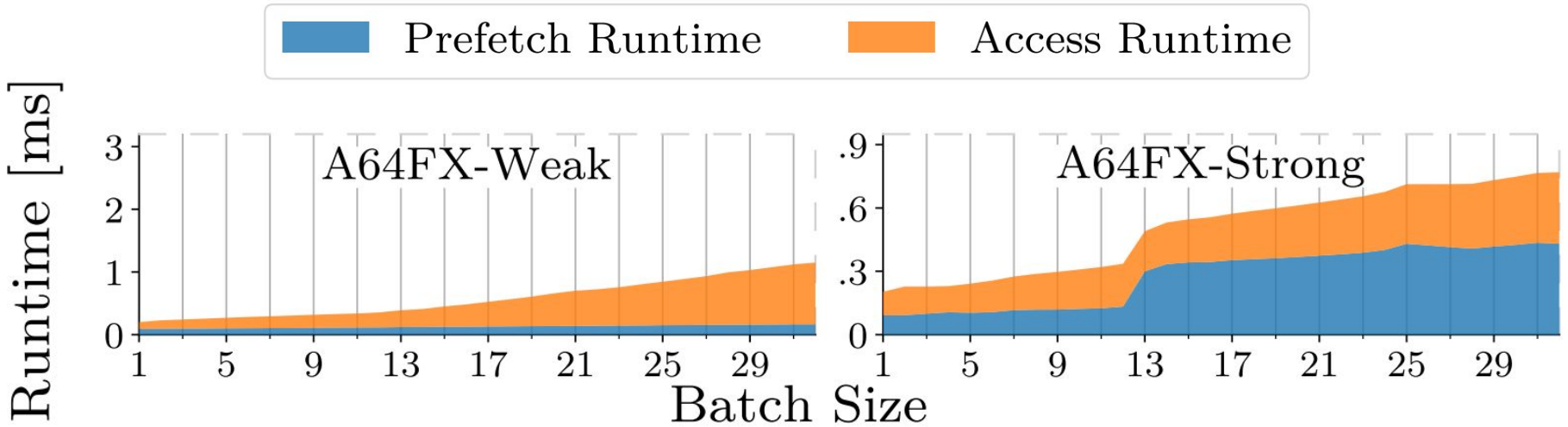
1. Prefetch batches of random addresses

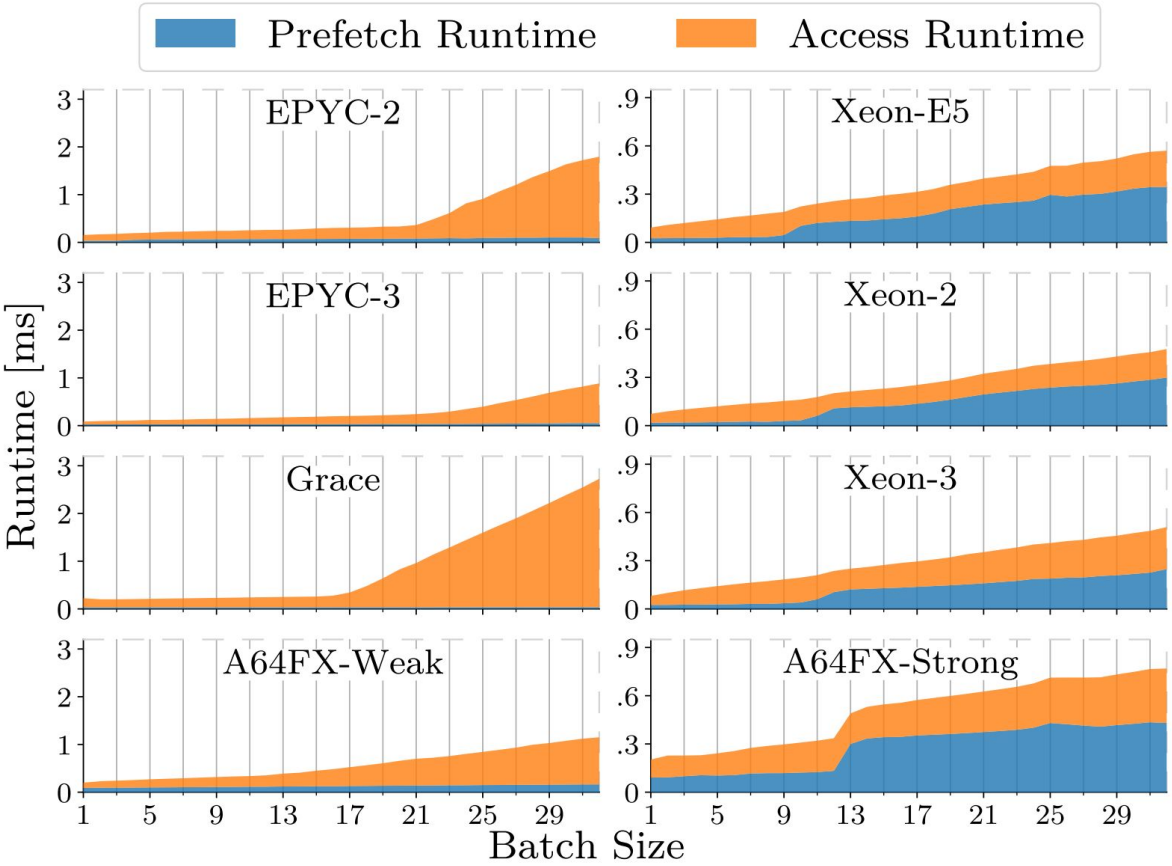
# Prefetching Reliability - Microbenchmark

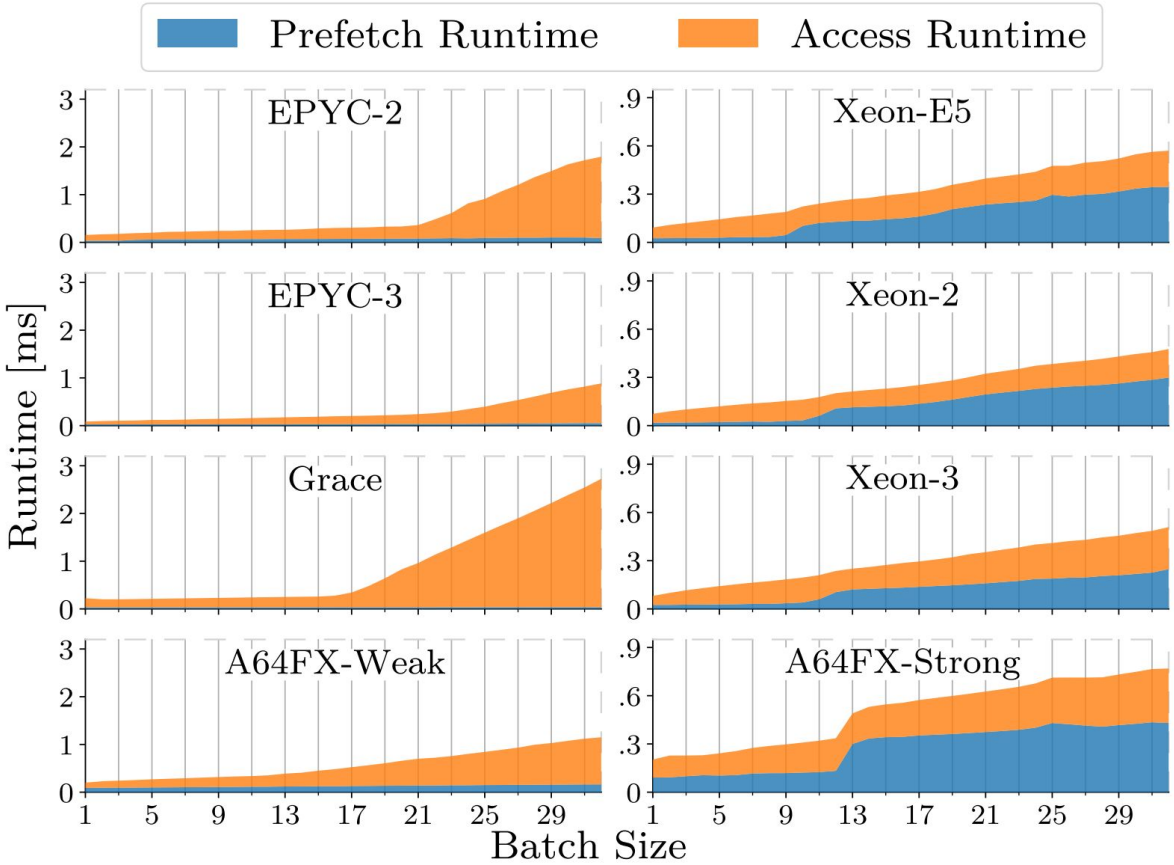
- Two phase microbenchmark:
  1. Prefetch batches of random addresses
  2. Access prefetched addresses

# Prefetching Reliability - Microbenchmark

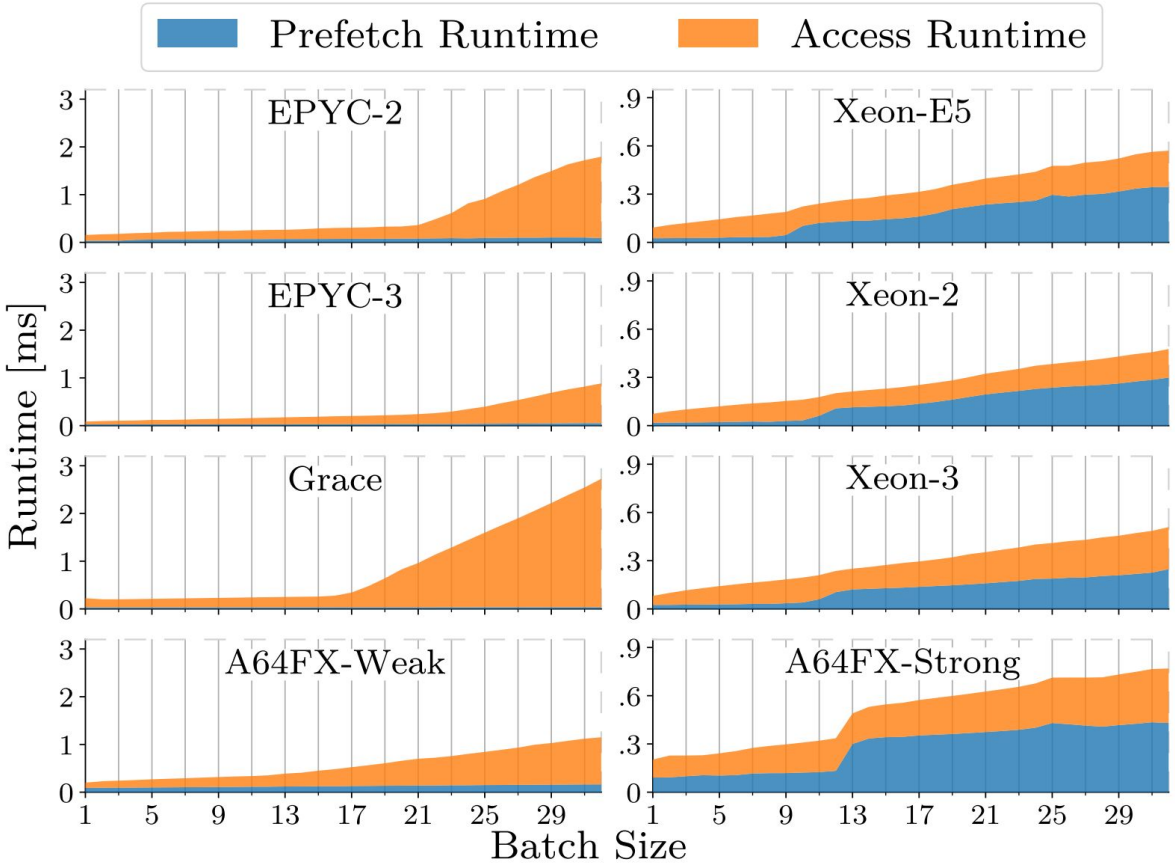
- Two phase microbenchmark:
  1. Prefetch batches of random addresses
  2. Access prefetched addresses



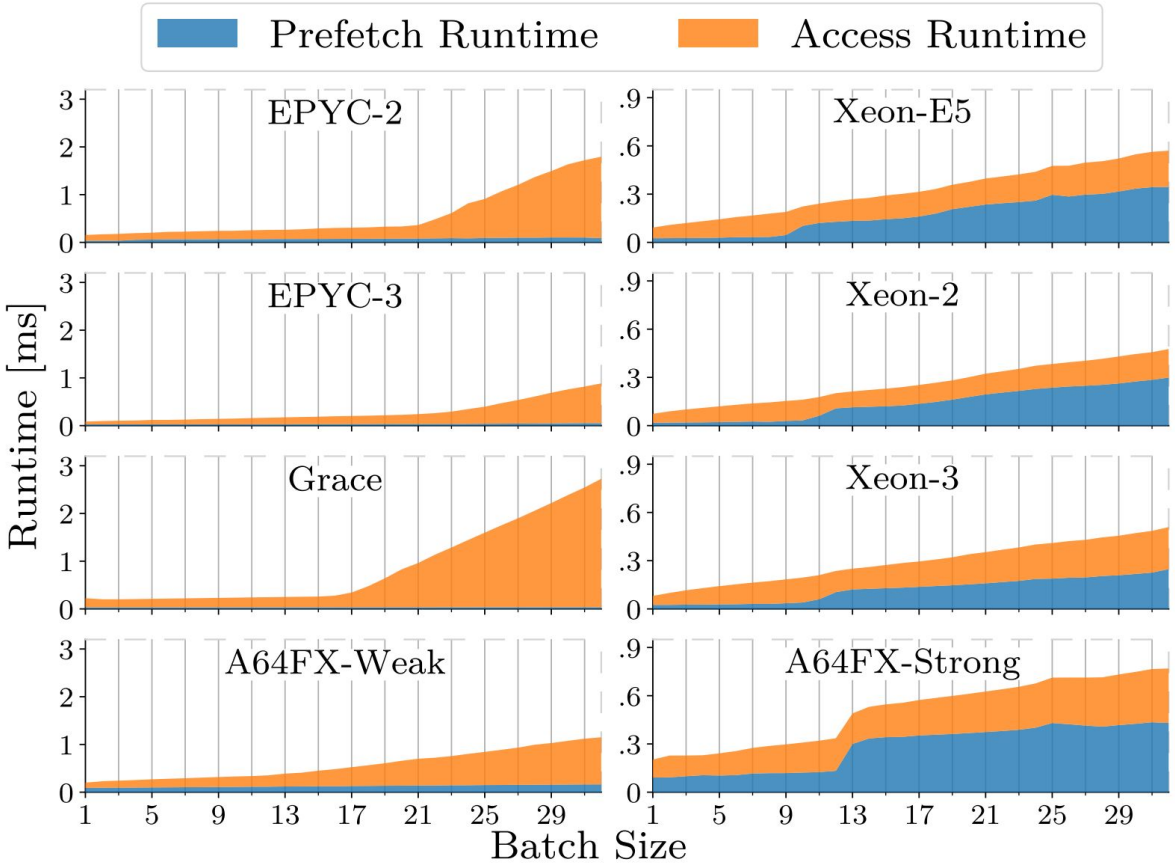

Prefetch Runtime
Access Runtime
EPYC-2
Xeon-E5
EPYC-3
Xeon-2
Grace
Xeon-3
A64FX-Weak
A64FX-Strong
Runtime [ms]
Batch Size
0
1
2
3
.3
.6
.9
1
5
9
13
17
21
25
29




- Classification of Prefetching reliability




- Classification of Prefetching reliability
- Spike in (prefetching) runtime after a batch size of *X*




- Classification of Prefetching reliability
- Spike in (prefetching) runtime after a batch size of *X*
  - *X* = FB size

# CPU Prefetching Reliability - $B^+$-Tree

# CPU Prefetching Reliability - $B^+$-Tree

- Interleave multiple lookups [*Kühn et al. 2024*]:

# CPU Prefetching Reliability - $B^+$-Tree

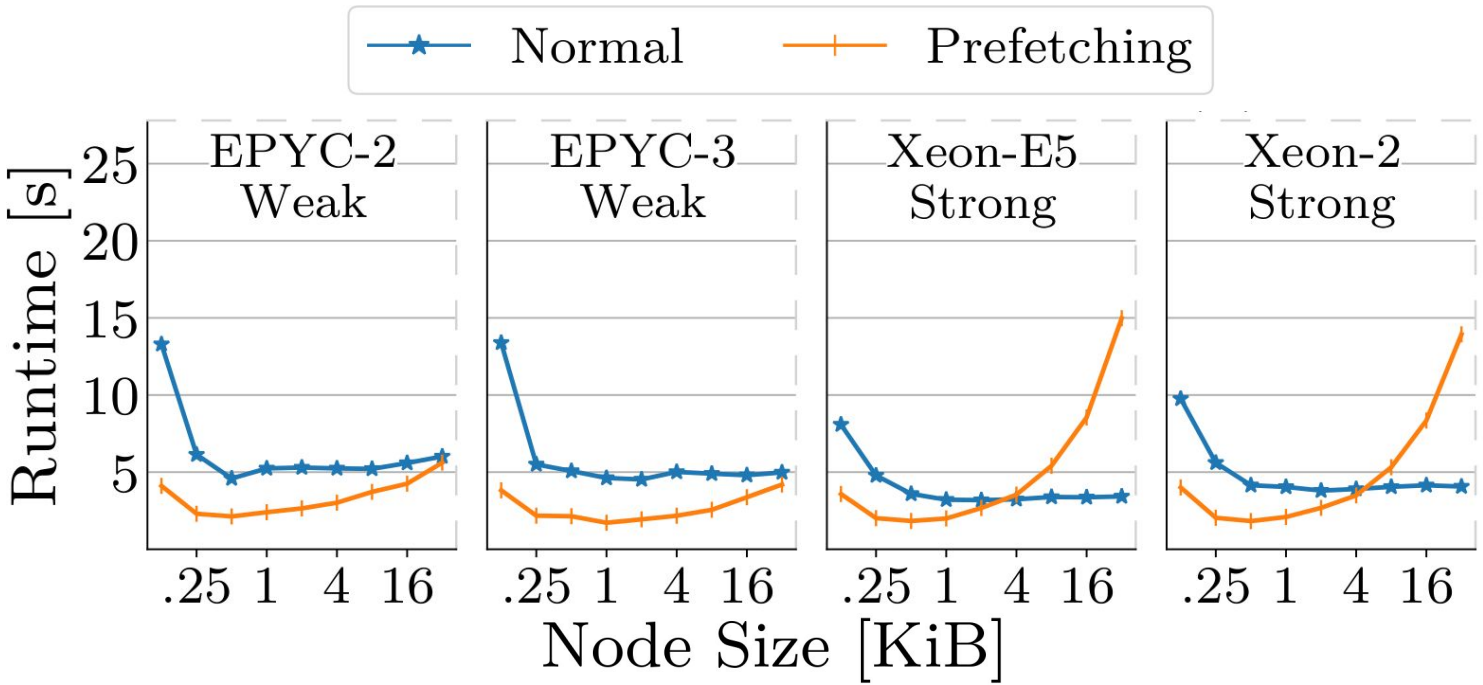
- Interleave multiple lookups [*Kühn et al. 2024*]:

  > Prefetch node > Continue with other lookups > Access node > Repeat

# CPU Prefetching Reliability - B$^+$-Tree

- Interleave multiple lookups [*Kühn et al. 2024*]:

  > Prefetch node > Continue with other lookups > Access node > Repeat

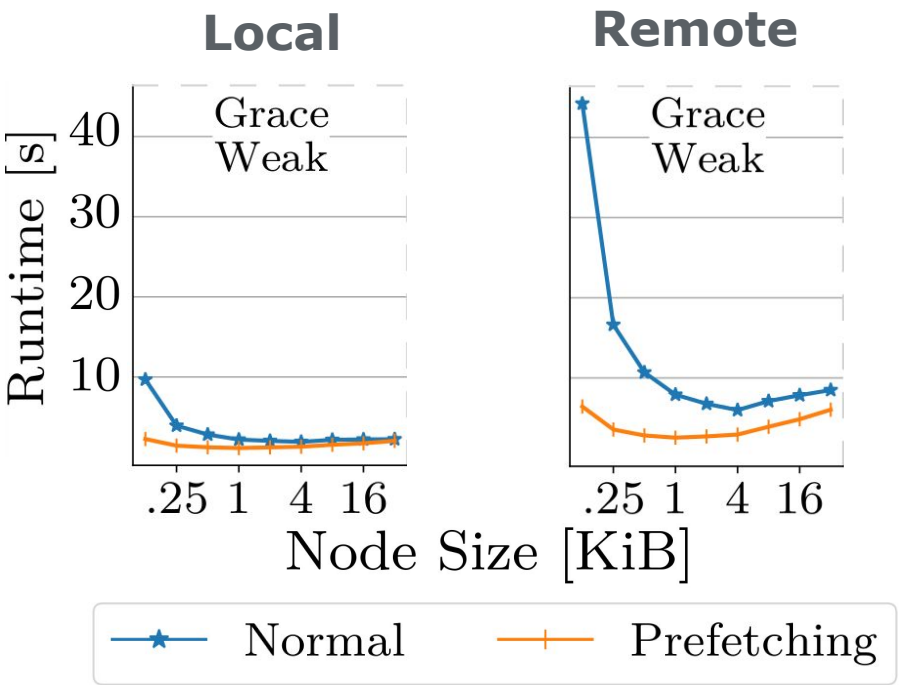


# CPU Prefetching Reliability - $B^+$-Tree

- Benchmarks with low and high memory access latencies

# CPU Prefetching Reliability - B$^+$-Tree

- Benchmarks with low and high memory access latencies



# Conclusion & Future Work

Conclusion:

# Conclusion & Future Work

Conclusion:

- Prefetching implementations vary across CPUs

  (prefetching reliability, target cache, eviction behavior)

# Conclusion & Future Work

Conclusion:

- Prefetching implementations vary across CPUs

  (prefetching reliability, target cache, eviction behavior)
- Prefetching reliability important when prefetching speculatively

# Conclusion & Future Work

Conclusion:

- Prefetching implementations vary across CPUs

  (prefetching reliability, target cache, eviction behavior)
- Prefetching reliability important when prefetching speculatively
- Prefetching can hide increased memory access latencies

# Conclusion & Future Work

Conclusion:

- Prefetching implementations vary across CPUs

  (prefetching reliability, target cache, eviction behavior)
- Prefetching reliability important when prefetching speculatively
- Prefetching can hide increased memory access latencies

Future Work:

- Multithreaded experiments

# Conclusion & Future Work

Conclusion:

- Prefetching implementations vary across CPUs
  (prefetching reliability, target cache, eviction behavior)
- Prefetching reliability important when prefetching speculatively
- Prefetching can hide increased memory access latencies

Future Work:

- Multithreaded experiments

Paper

Code

# Backup-Slides

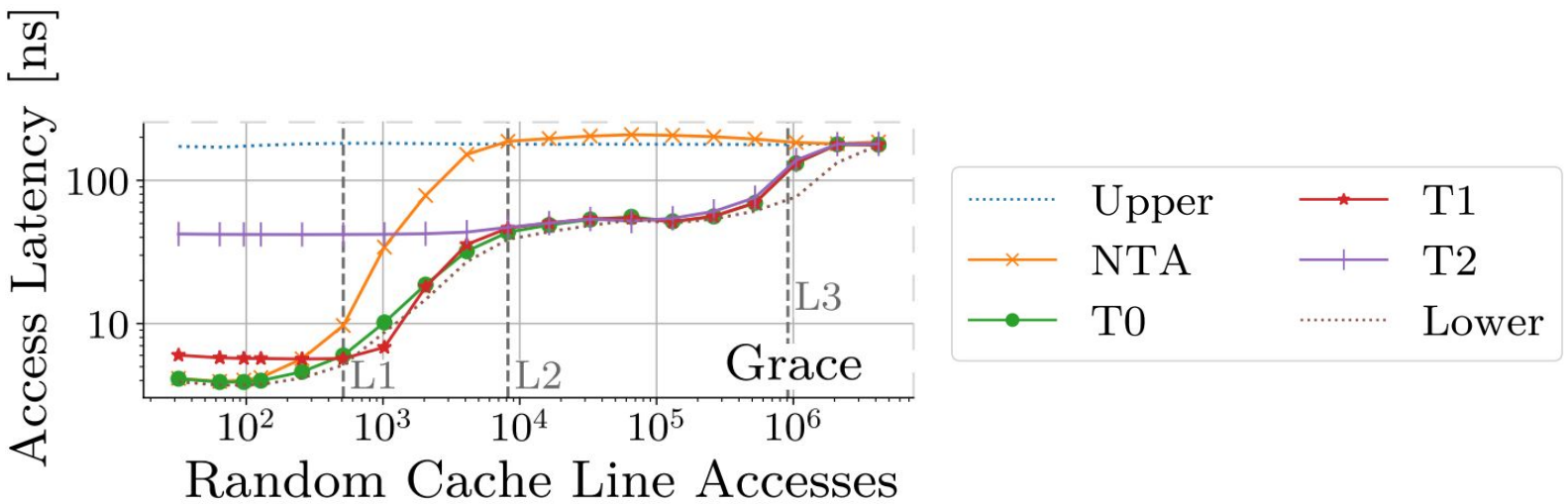
# High Latency Memory

- Heterogeneous memory causes varying access latencies
- 1.4x – 4.2x for used Systems
- **Problem:** Performance loss in latency bound workloads

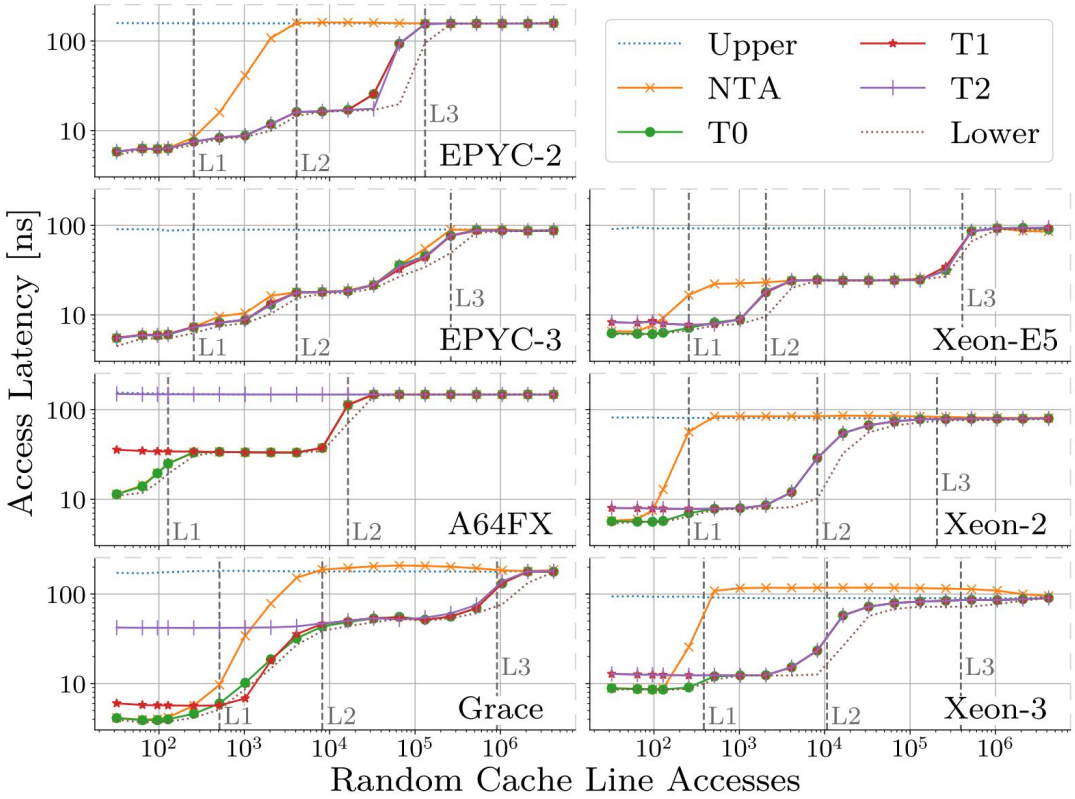
| ID | CPUs | Memory | Latency [ns] | |
|---|---|---|---|---|
| | | | Local | Remote |
| EPYC-2 | 2× 7742 | DDR4 | 153 | 277 |
| EPYC-3 | 2× 7413 | DDR4 | 83 | 272 |
| Xeon-E5 | 2× E5-2689 v4 | DDR4 | 80 | 130 |
| Xeon-2 | 2× 5220S | DDR4 | 76 | 156 |
| Xeon-3 | 2× 8352Y | DDR4 | 83 | 134 |
| A64FX | 1× A64FX | HBM2 | 215 | 307 |
| Grace | 1× Grace CPU | LPDDR5X HBM3 | 181 | 760 |

# CPU Prefetching Localities

- Prefetch batches of random cache lines with different localities
- Measure subsequent access latency
- Compare to accessed cache lines (Lower) and random cache lines (Upper)

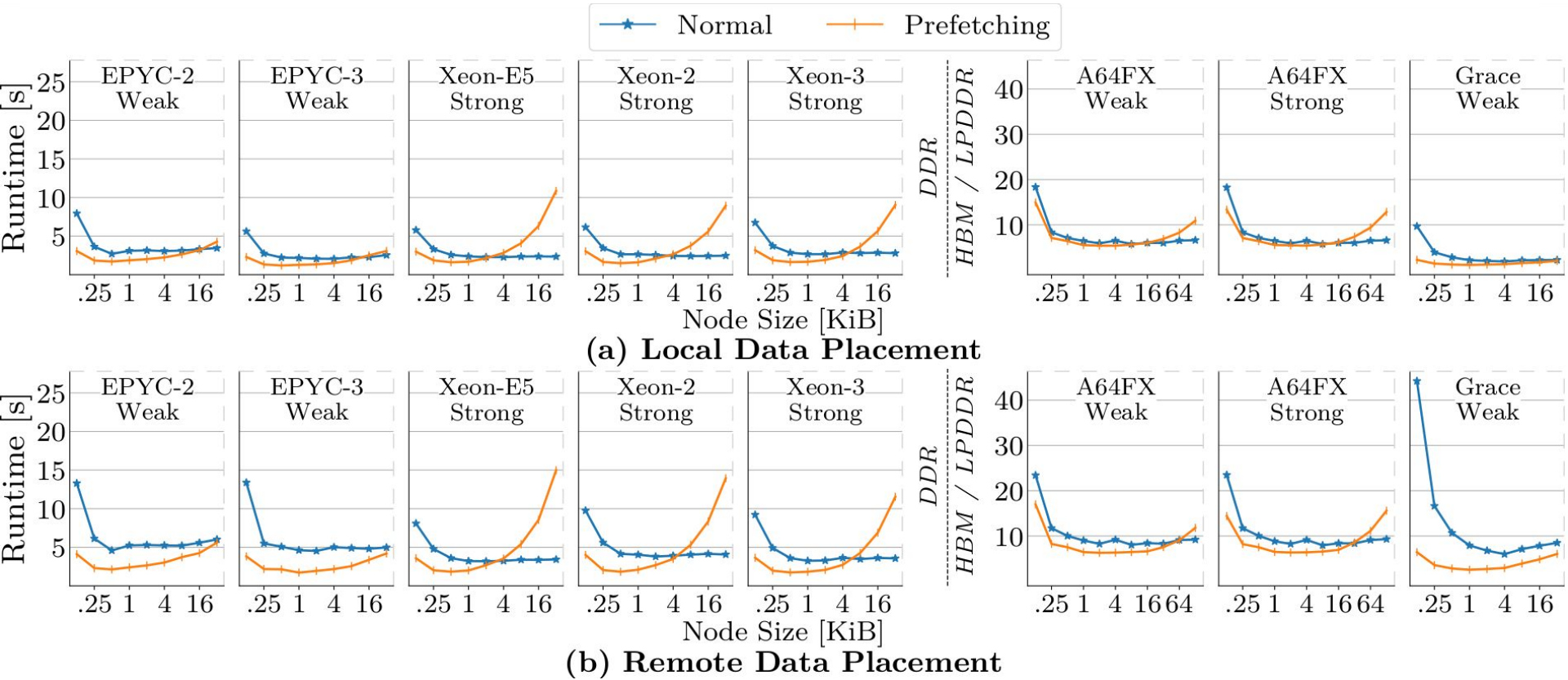


# CPU Prefetching Localities




- *T0* and *NTA* implementations consistent on most CPUs
- Other localities have varying implementations, even for the same CPU manufacturer

# CPU Prefetching Reliability - B$^+$-Tree



# CPU Prefetching High Latency - Binary Search

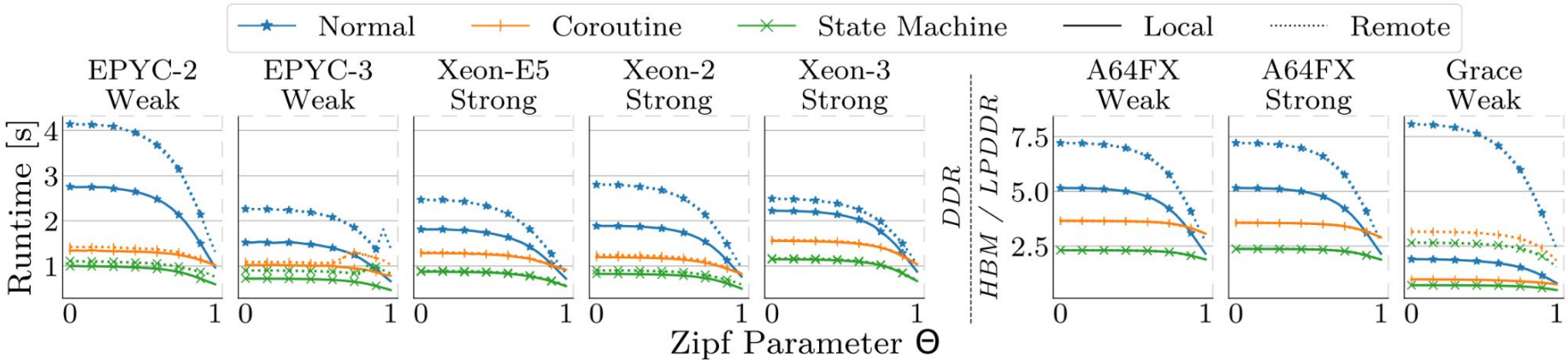



**Figure 6: Binary search runtimes given different data placement, implementations, and Zipf parameters.**