# Streamlined On-Chip Temporal Prefetching

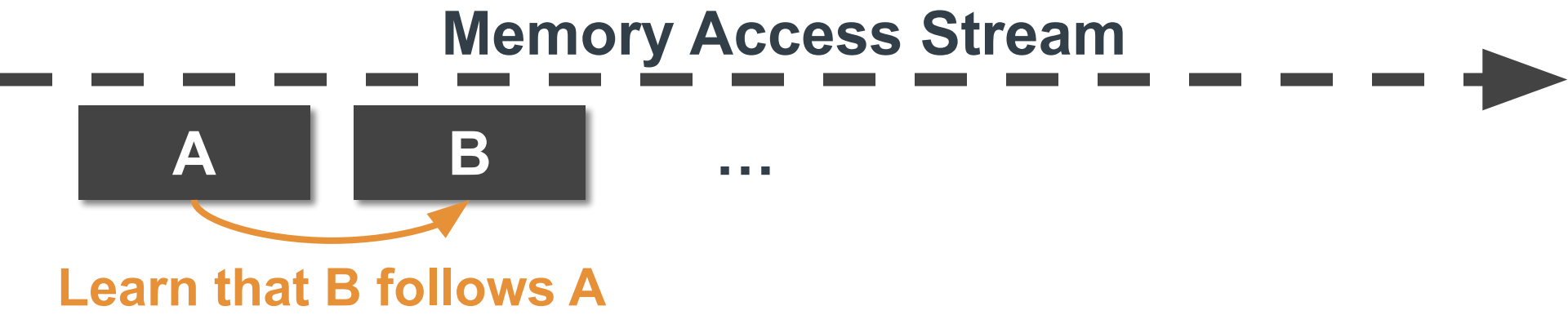
**Quang Duong** **Calvin Lin**

The University of Texas at Austin
Computer Science
*College of Natural Sciences*

# Background: Temporal Prefetching

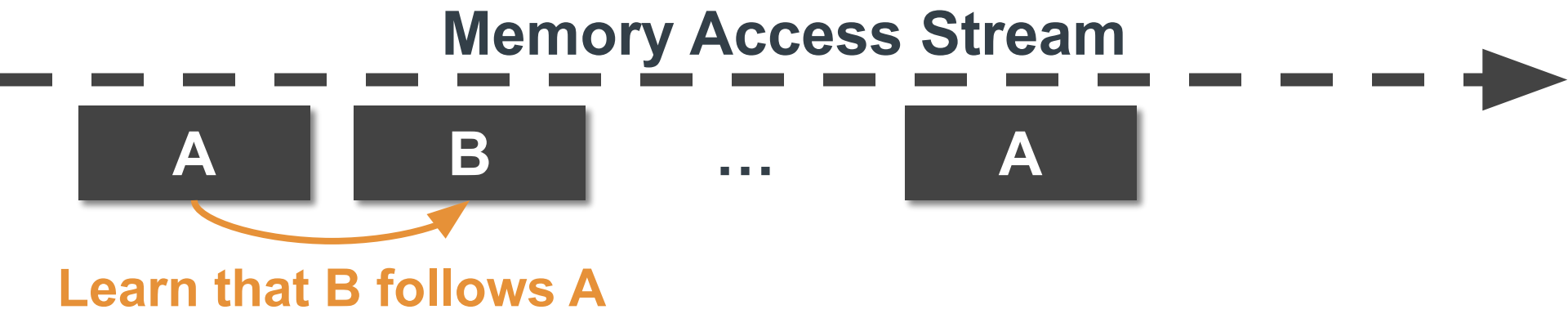

**Capable of covering ANY repeated stream**

# Background: Temporal Prefetching



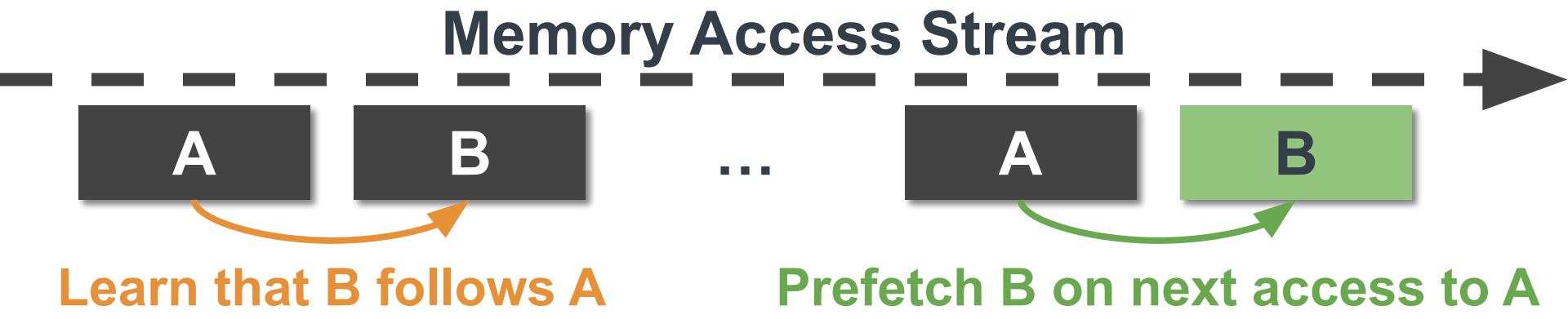

**Capable of covering ANY repeated stream**

# Background: Temporal Prefetching



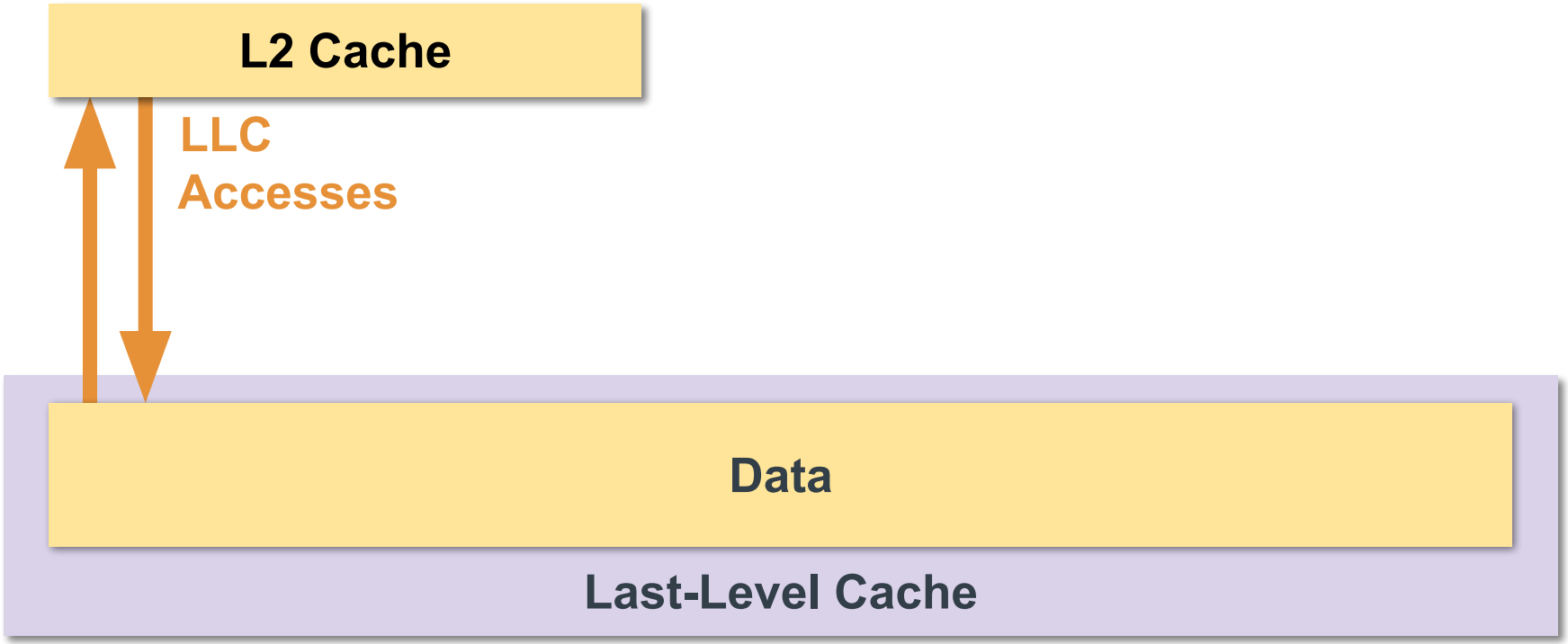

**Capable of covering ANY repeated stream**

# Background: Temporal Prefetching

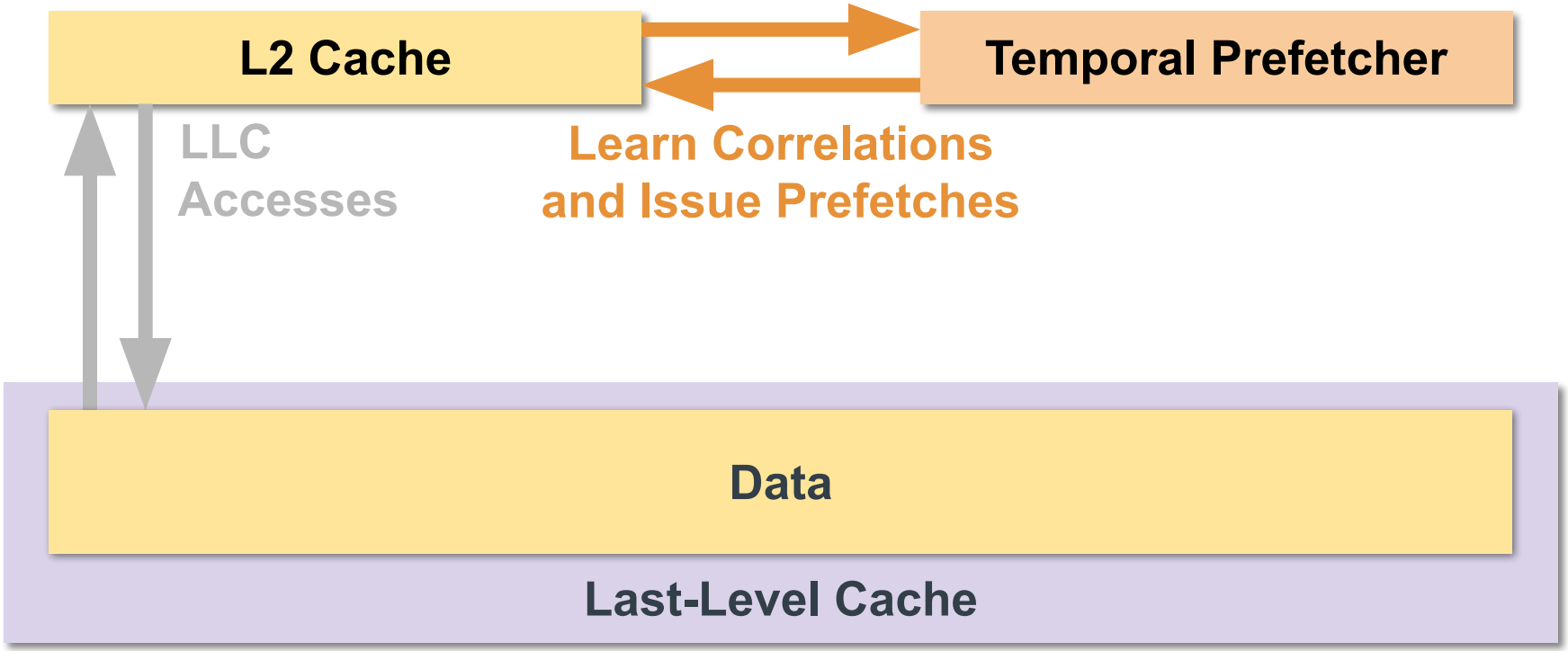



**Capable of covering ANY repeated stream**

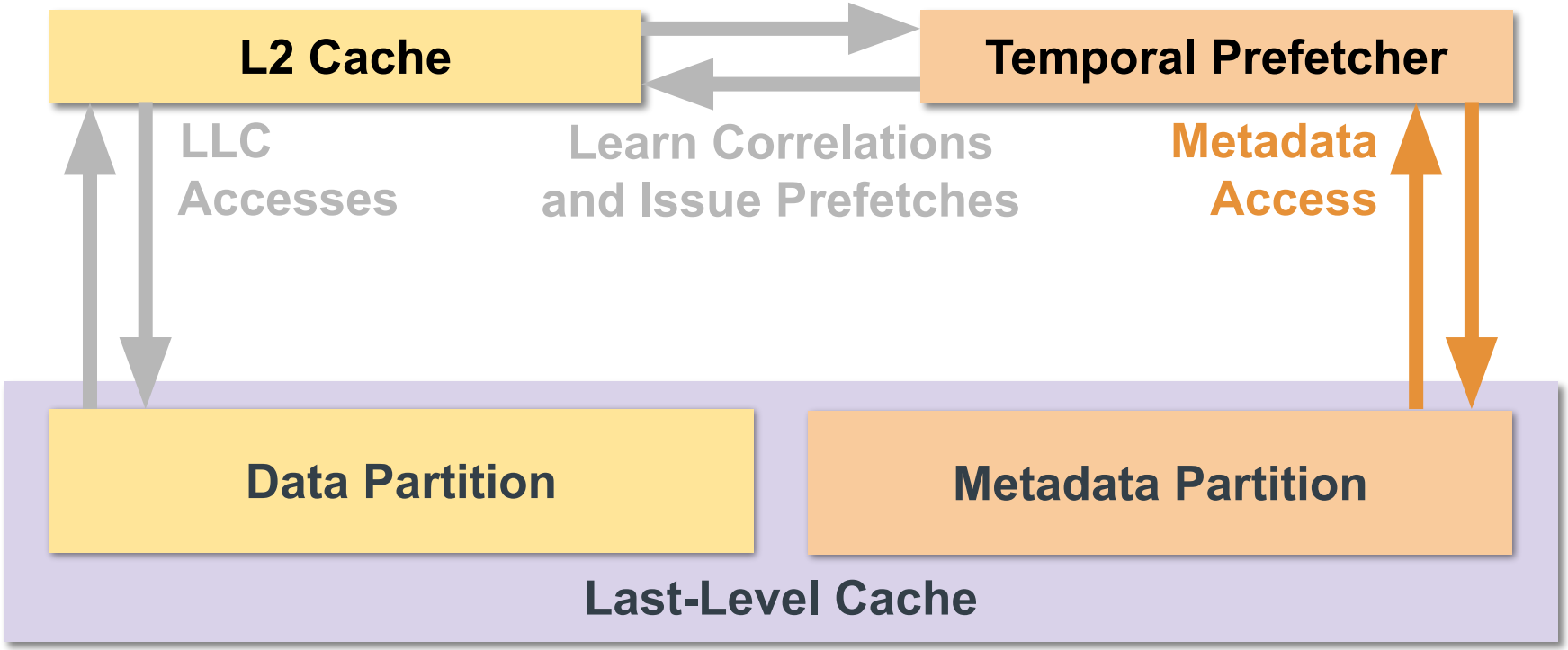
# Background: On-Chip Prefetchers



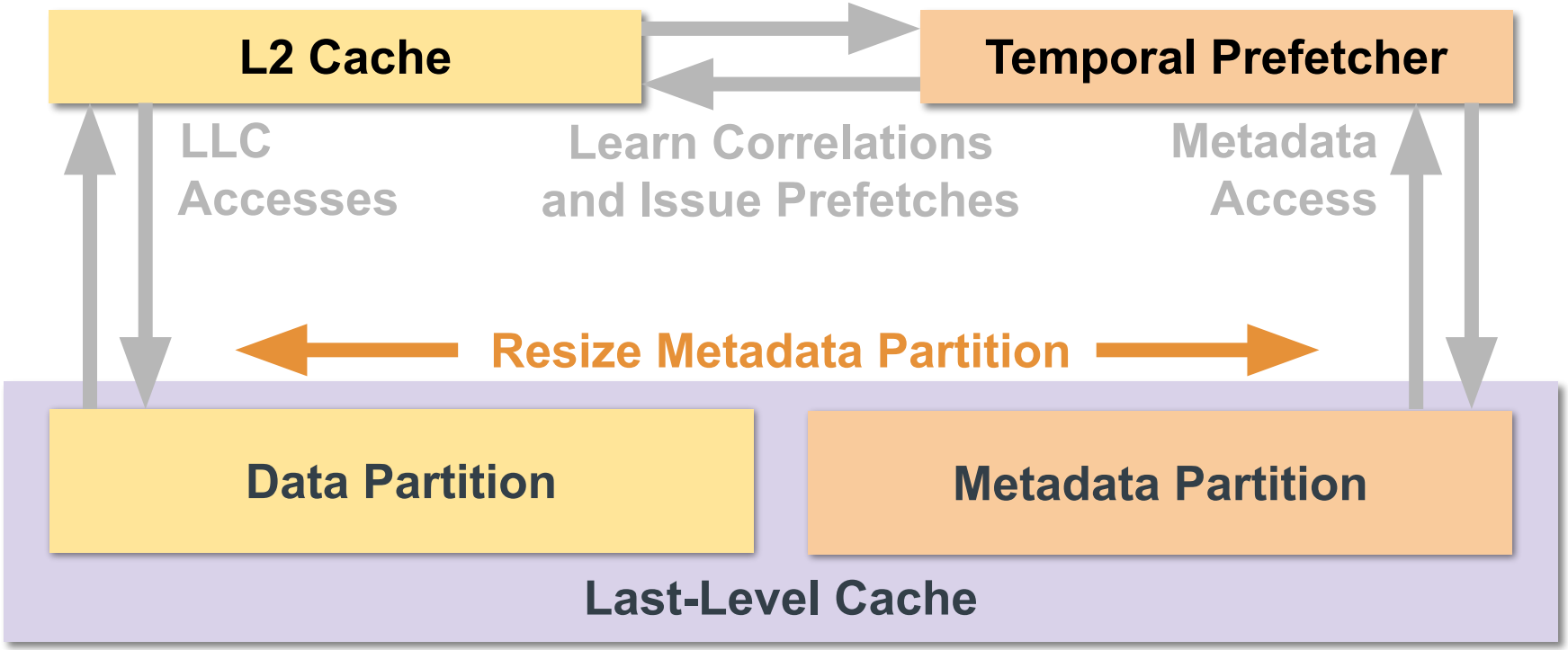
# Background: On-Chip Prefetchers



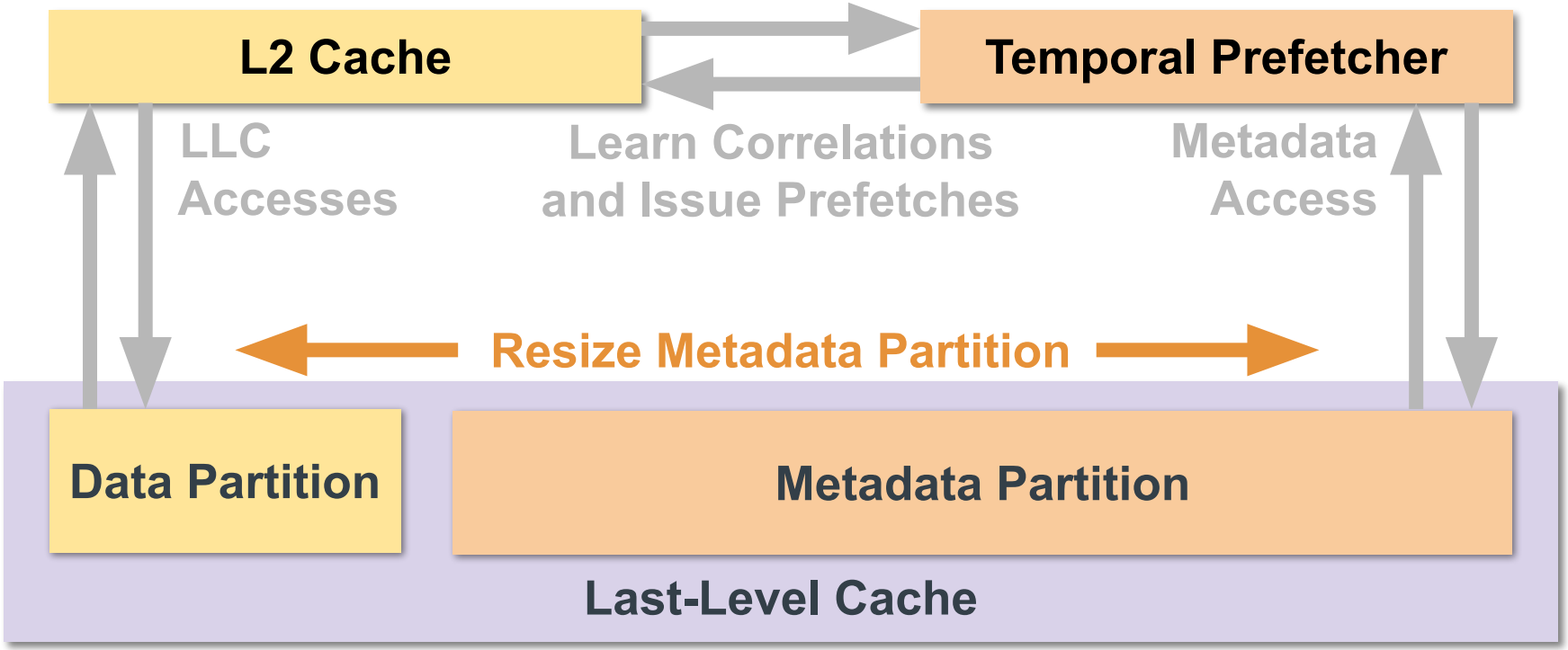
# Background: On-Chip Prefetchers



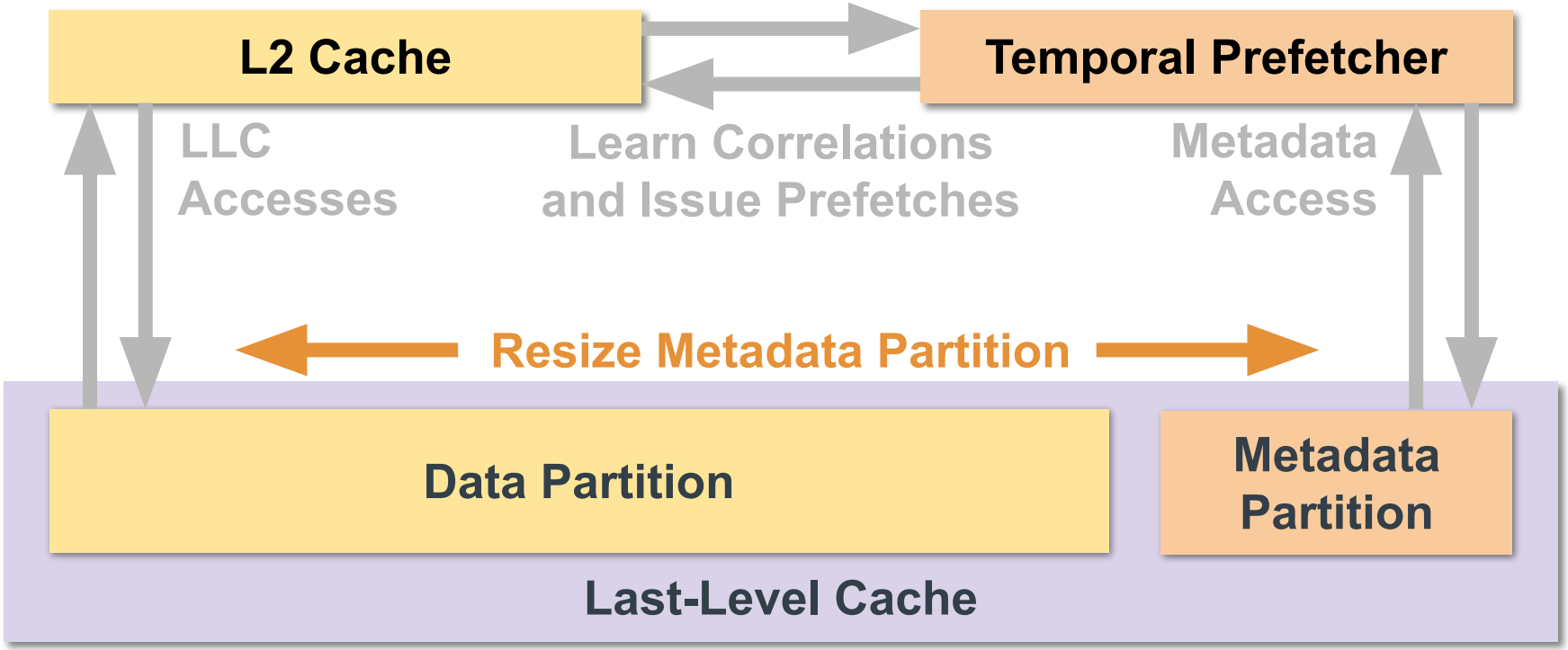
# Background: On-Chip Prefetchers



# Background: On-Chip Prefetchers



# Background: On-Chip Prefetchers



# Motivation: Two Opposing Objectives

(O1) To maximize prefetch coverage, prefetchers need larger metadata partitions

(O2) To minimize the impact on data hit rates, prefetchers need smaller metadata partitions

# Overview

## Our work answers two design questions:

**(Q1) How should on-chip metadata *be represented?***

**(Q2) How should on-chip metadata *be managed?***

# Overview

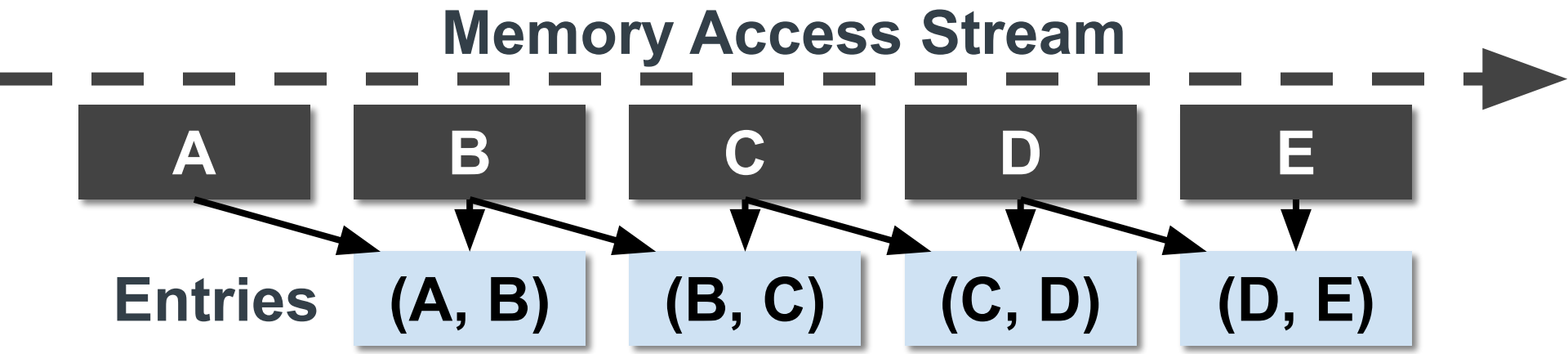
## Our work answers two design questions:

**(Q1) How should on-chip metadata *be represented?***

**(Q2) How should on-chip metadata *be managed?***

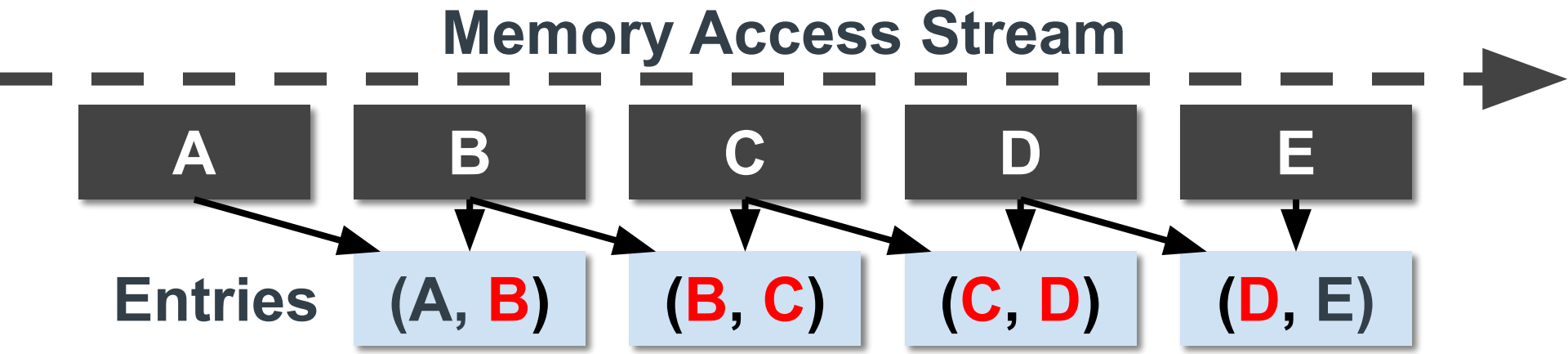
# SOTA Metadata Representation: Pairs

**Entries pair one trigger with one prefetch target**



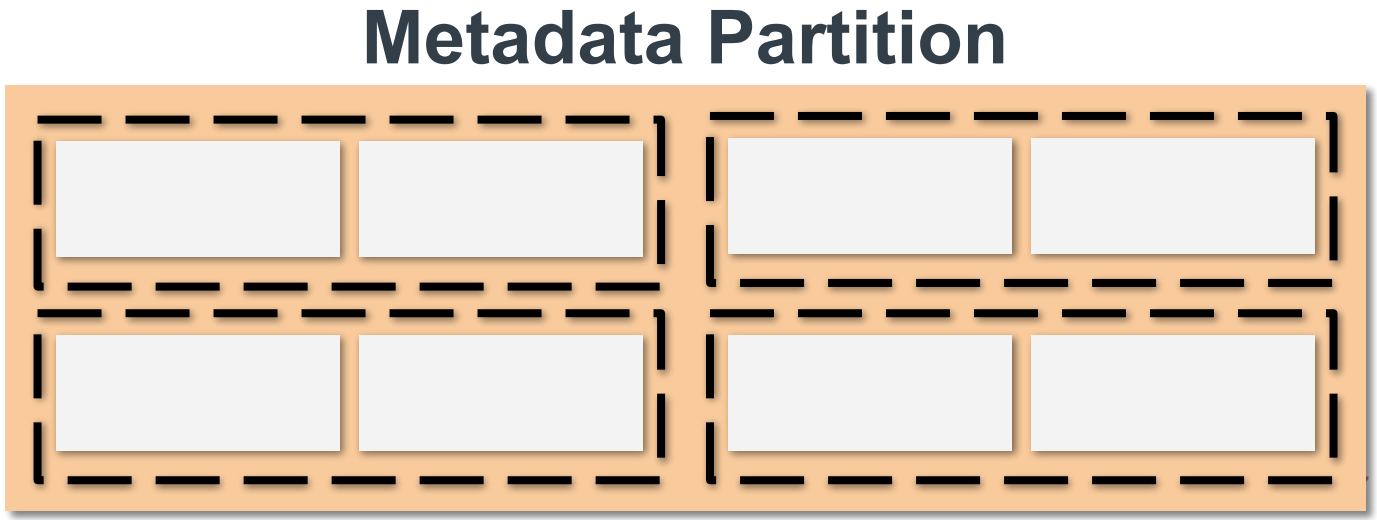
# Pairs are Inherently Redundant

**Pairs inherently store each address twice**



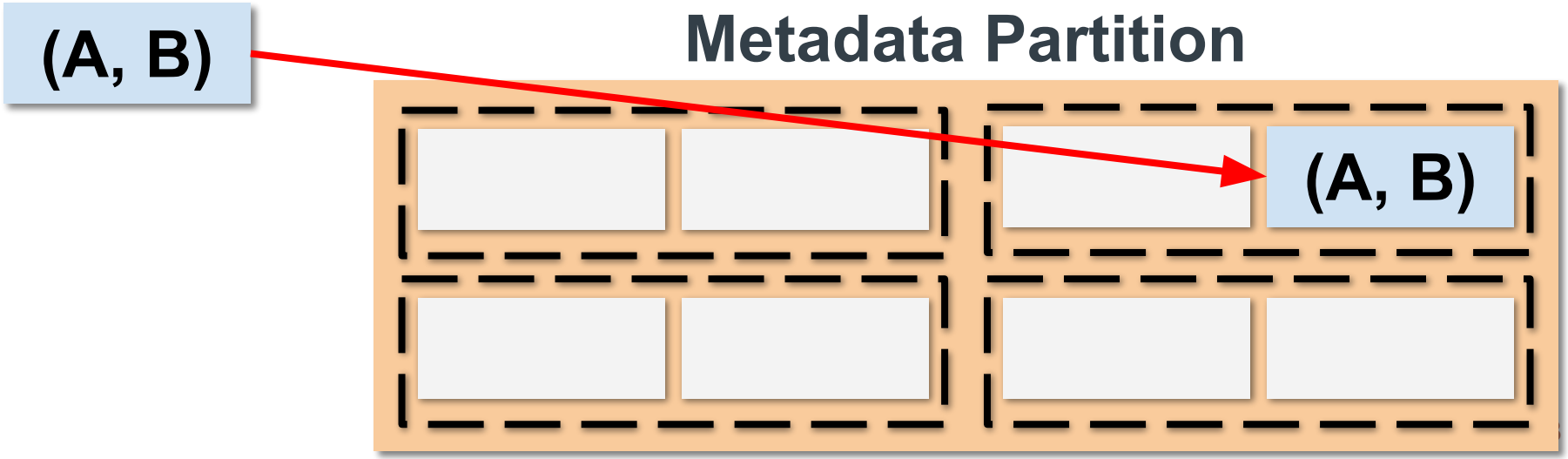
# Pairs lack Spatial Locality

**Temporal locality doesn't translate to spatial locality**



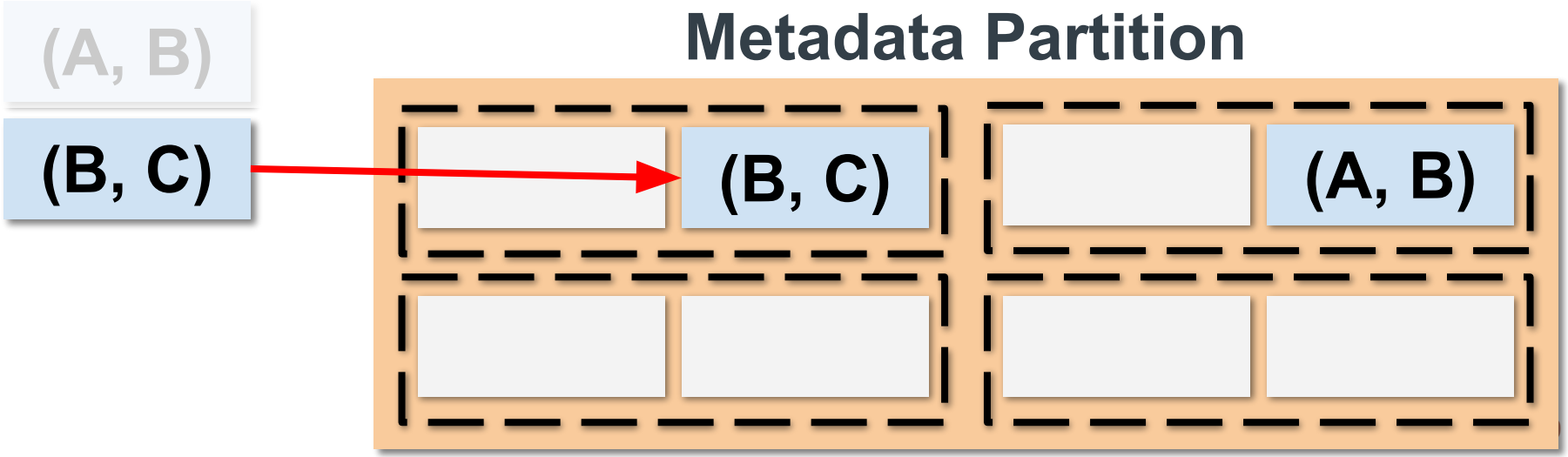
# Pairs lack Spatial Locality

**Temporal locality doesn't translate to spatial locality**



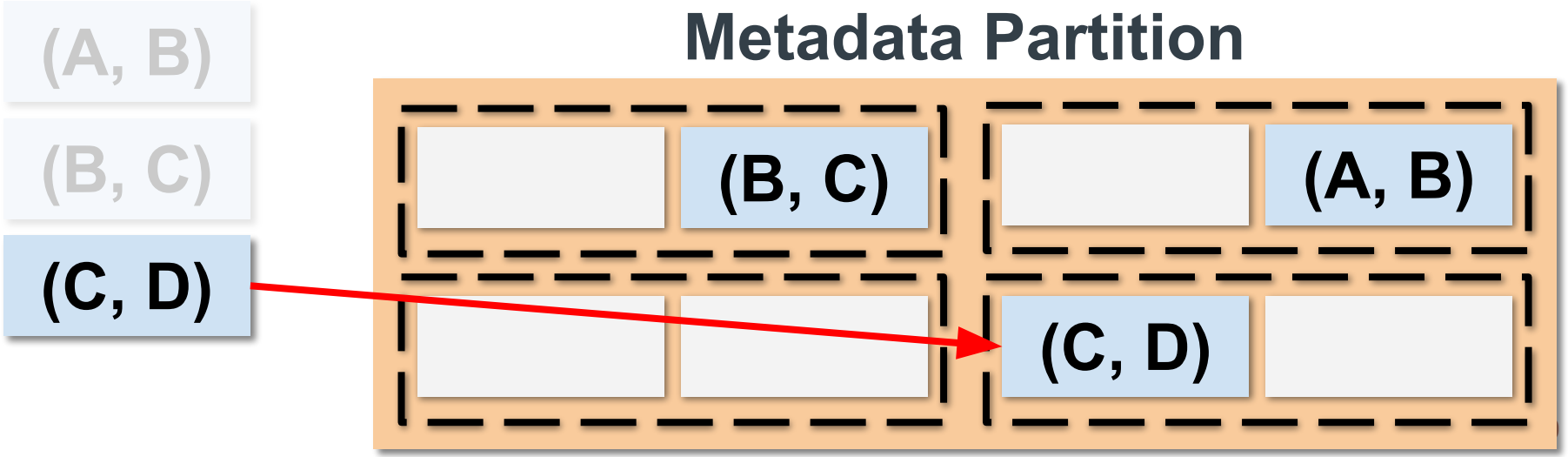
# Pairs lack Spatial Locality

**Temporal locality doesn't translate to spatial locality**



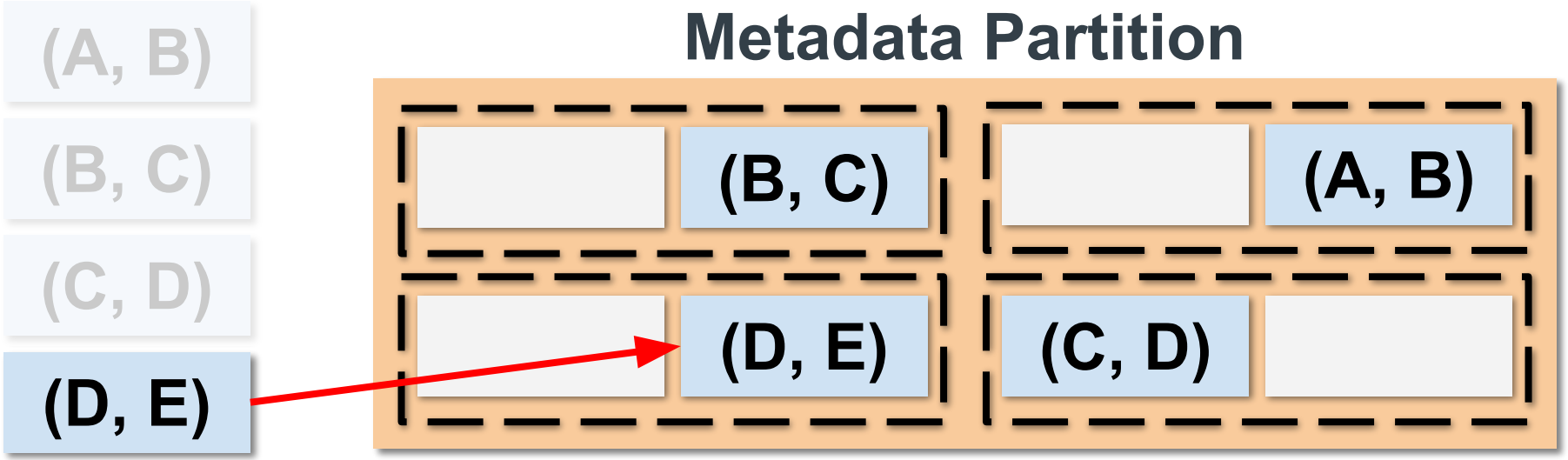
# Pairs lack Spatial Locality

**Temporal locality doesn't translate to spatial locality**



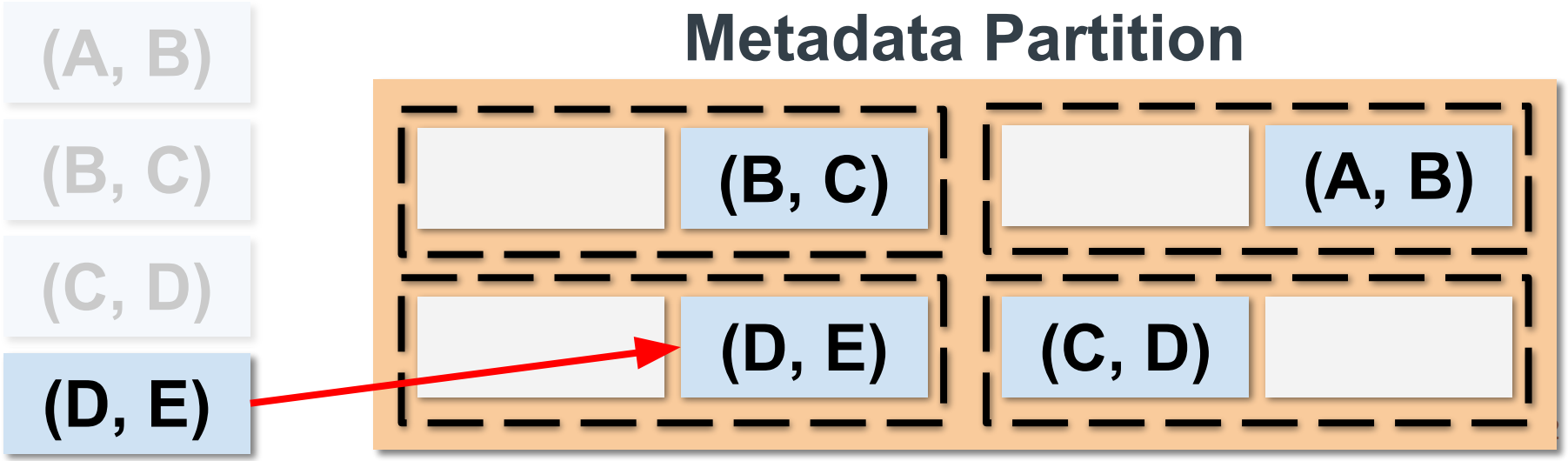
# Pairs lack Spatial Locality

**Temporal locality doesn't translate to spatial locality**



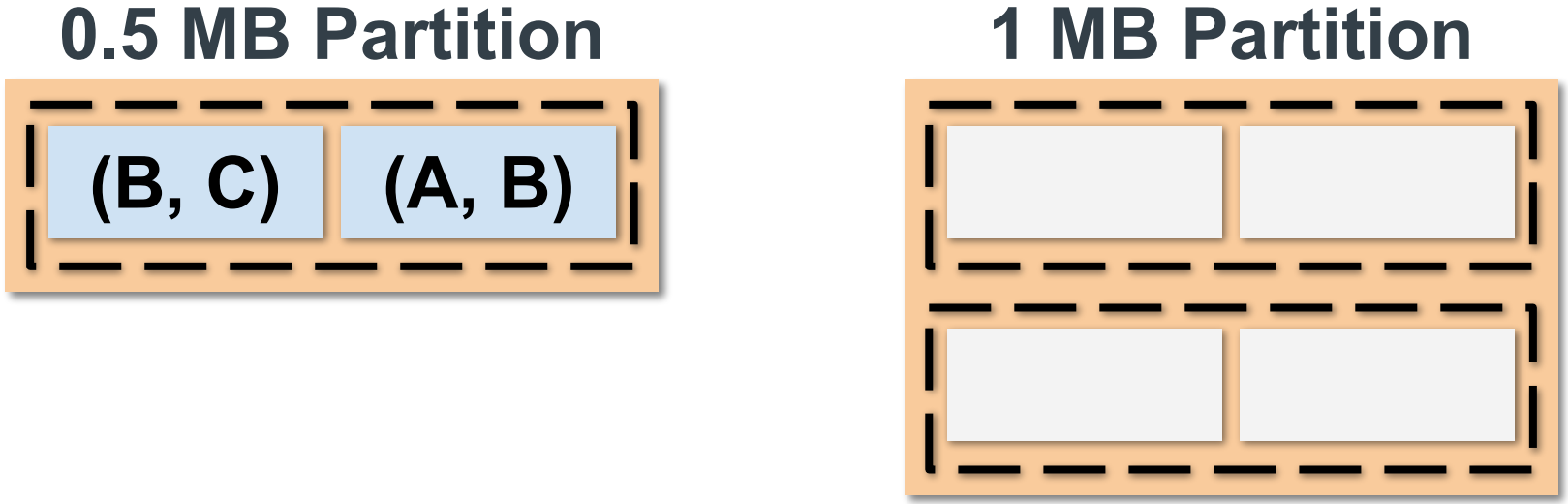
# Pairs lack Spatial Locality

**Pairs incur one LLC read per prefetch**



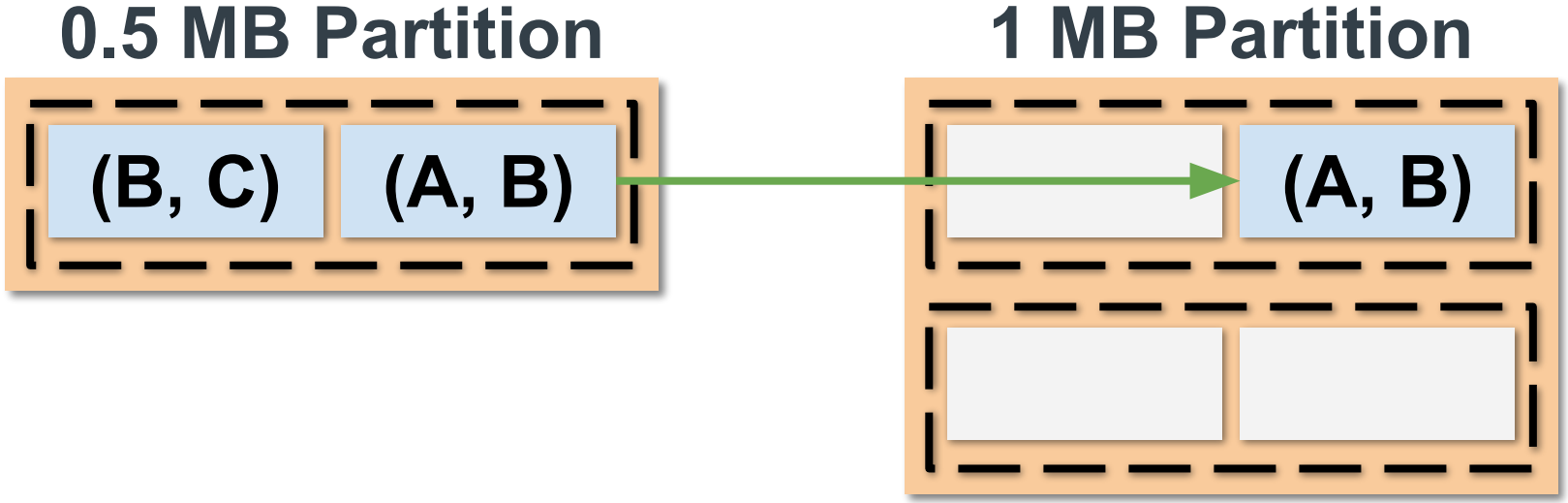
# Pairs incur Traffic on Resize

**Pair indexing function depends on partition size**



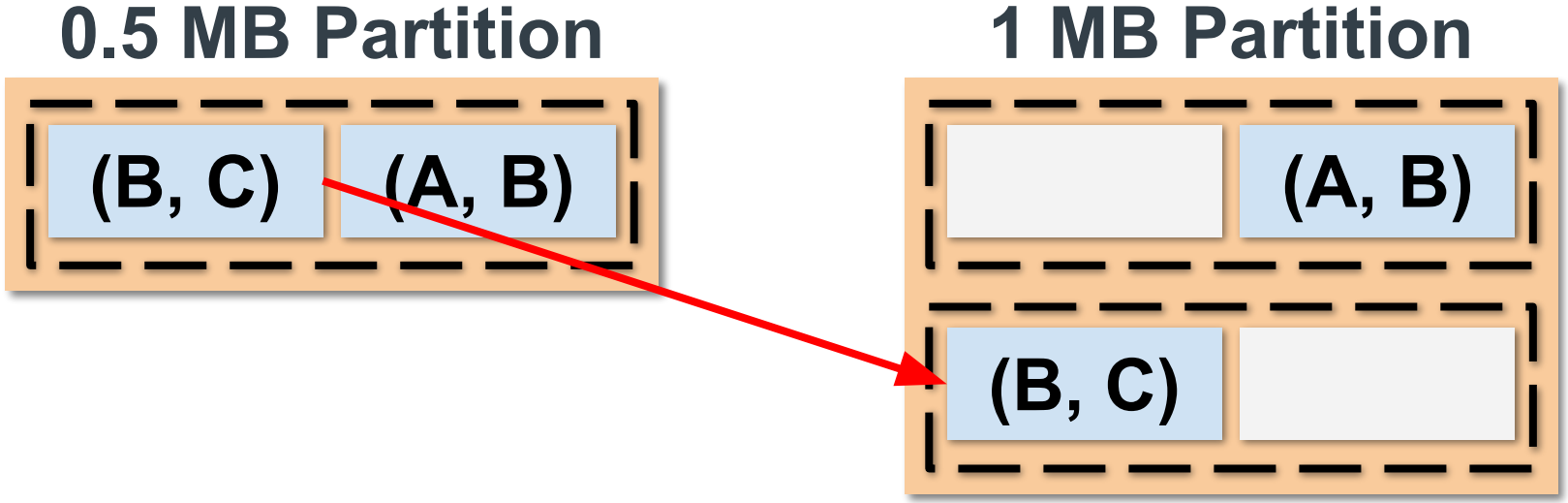
# Pairs incur Traffic on Resize

**Pair indexing function depends on partition size**



# Pairs incur Traffic on Resize

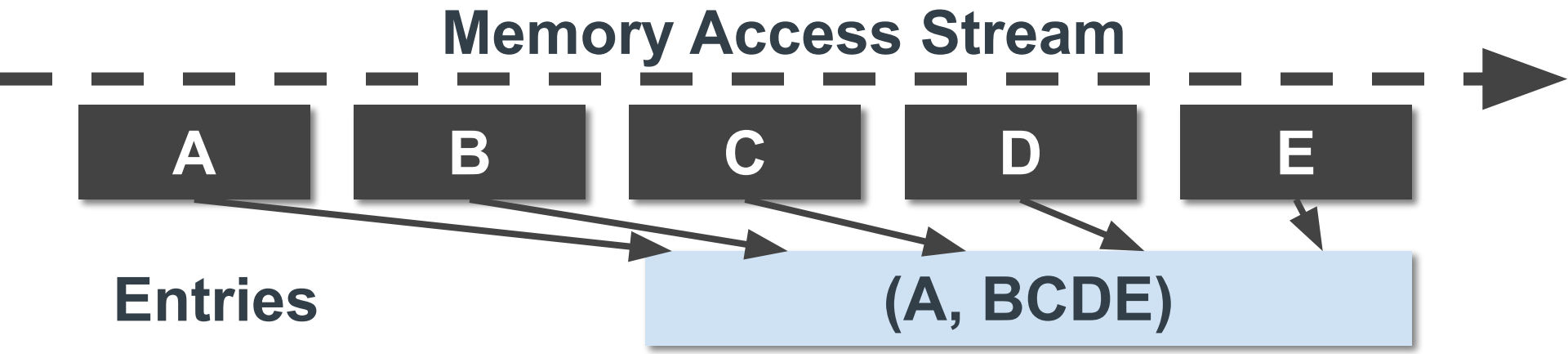
**Pair indexing function depends on partition size**



# What is a better metadata representation?

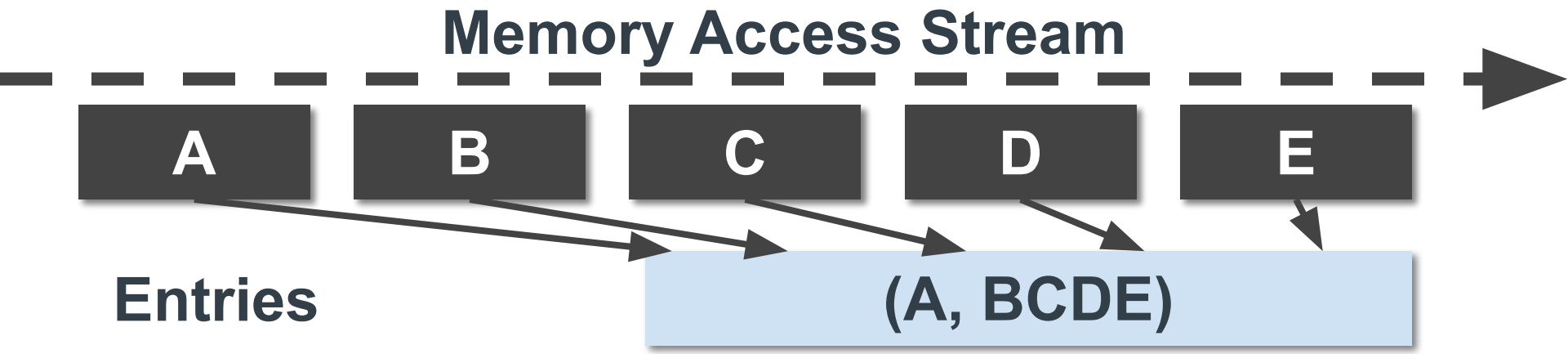
# Our Metadata Representation: Streams

**Entries map a trigger to multiple prefetch targets**



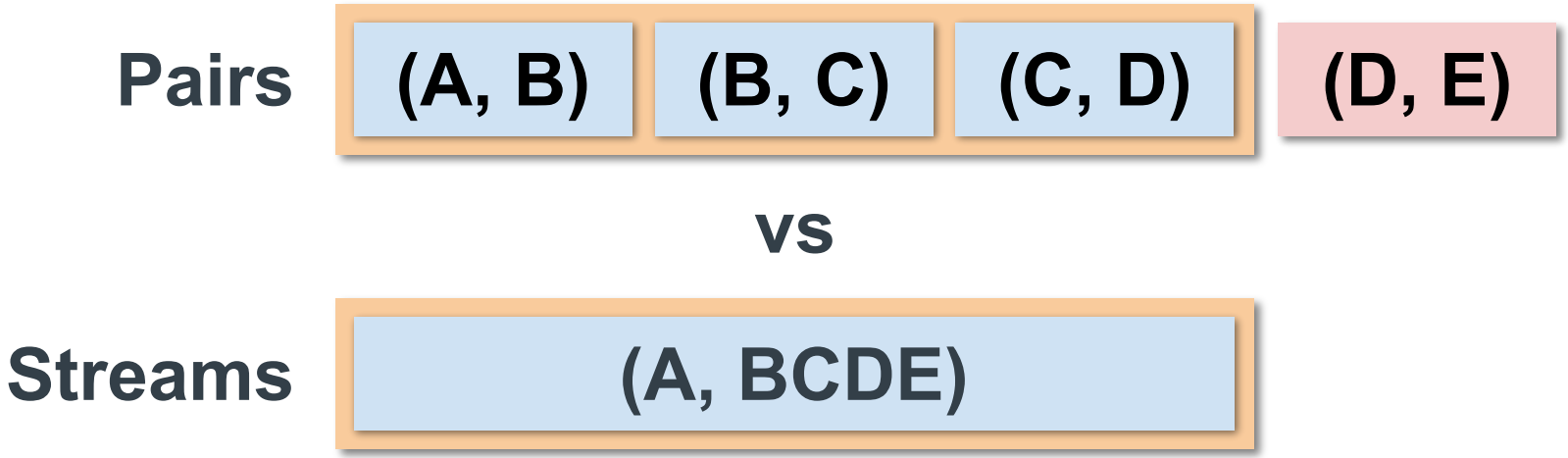
# Our Metadata Representation: Streams

**Entries map a trigger to 4 prefetch targets**



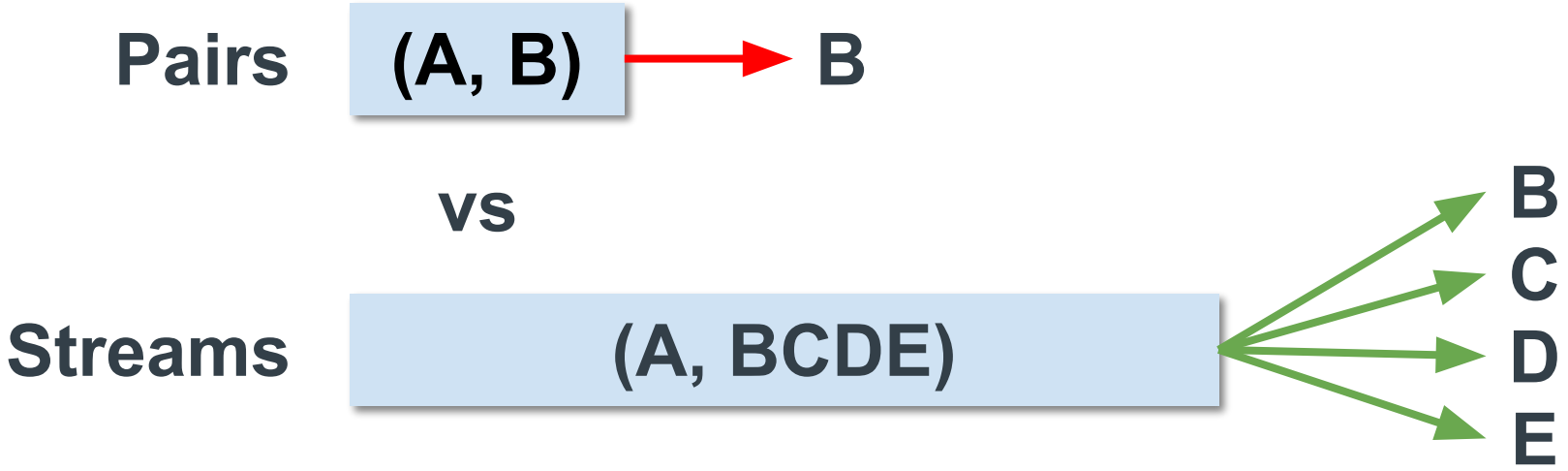
# Streams reduce Redundancy

**Streams enable the store of 33% more correlations**



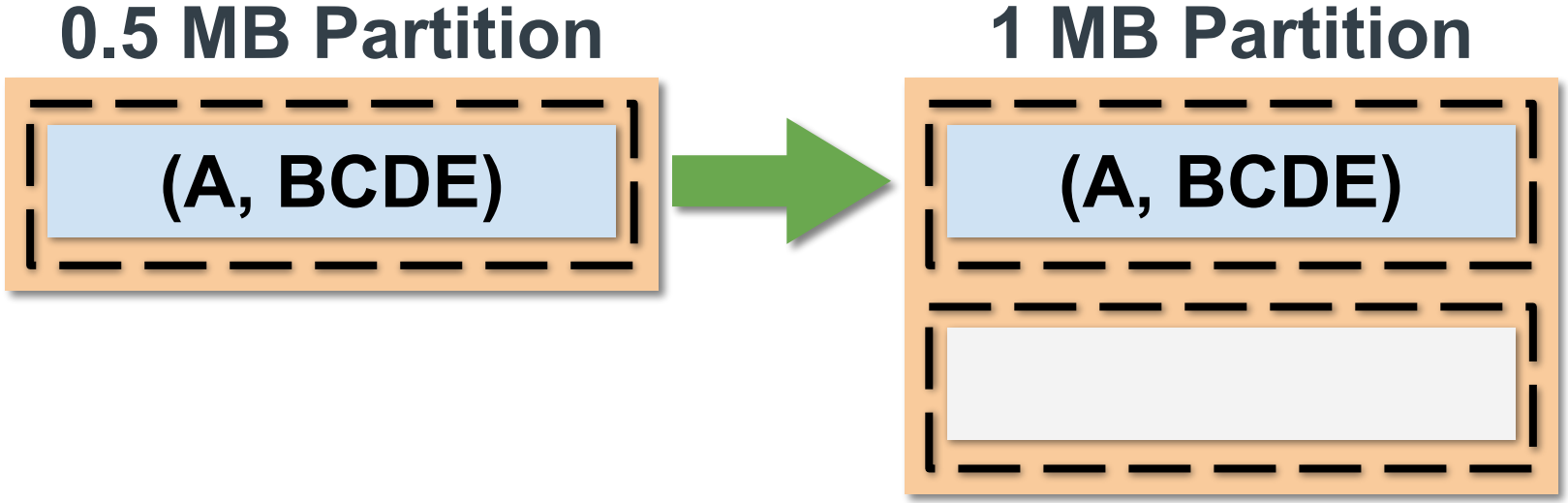
# Streams have Inherent Spatial Locality

**Streams reduce metadata traffic by up to 4x**



# Streams Resize Migration-Free

**Streams enable a simpler, fixed indexing function**



# Streams introduce New Problems

# Streams introduce New Problems

**Streams introduce a new form of redundancy**

(A, BCDE)

(B, CDEF)

# Streams introduce New Problems

**Streams introduce a new form of redundancy**

**Streams inherently reduce the number of triggers**

**Streams lead to more conflict misses**

# Our Streamline prefetcher resolves these problems

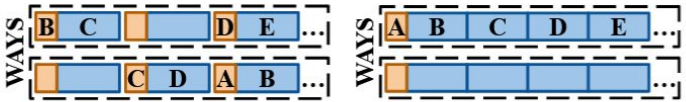
**See paper for more details**

m on High-Performance Computer Architecture (HPCA)

## Streamlined On-Chip Temporal Prefetching

Quang Duong and Calvin Lin
Department of Computer Science, The University of Texas at Austin
{qduong,lin}@cs.utexas.edu

***Abstract*—In this paper, we present the Streamline temporal prefetcher, which introduces a stream-based metadata representation that produces three significant benefits over Triangel, the previous state-of-the-art in temporal prefetching. First, it removes redundancy present in Triangel's metadata. Second, it prioritizes the storage of those metadata entries that have higher prefetch utility. Third, it eliminates the need for the untenable LLC traffic that is induced when Triangel dynamically adjusts the size of its metadata store. The end result is that for memory-intensive SPEC 2006, SPEC 2017, and GAP benchmarks, Streamline outperforms Triangel by 6.7 percentage points on an 8-core**




(a) **Pairwise Metadata** (b) **Stream-Based Metadata**

Fig. 1: **Benefits of Stream-Based Designs**: We show two truncated LLC ways of metadata entries for the stream [*A*, *B*, *C*, *D*, *E*]. **(a)** Pairwise entries redundantly store addresses as both **trigger** and **prefetch target**. Moreover, since

# Overview

## Our work answers two design questions:

(Q1) How should on-chip metadata *be represented?*

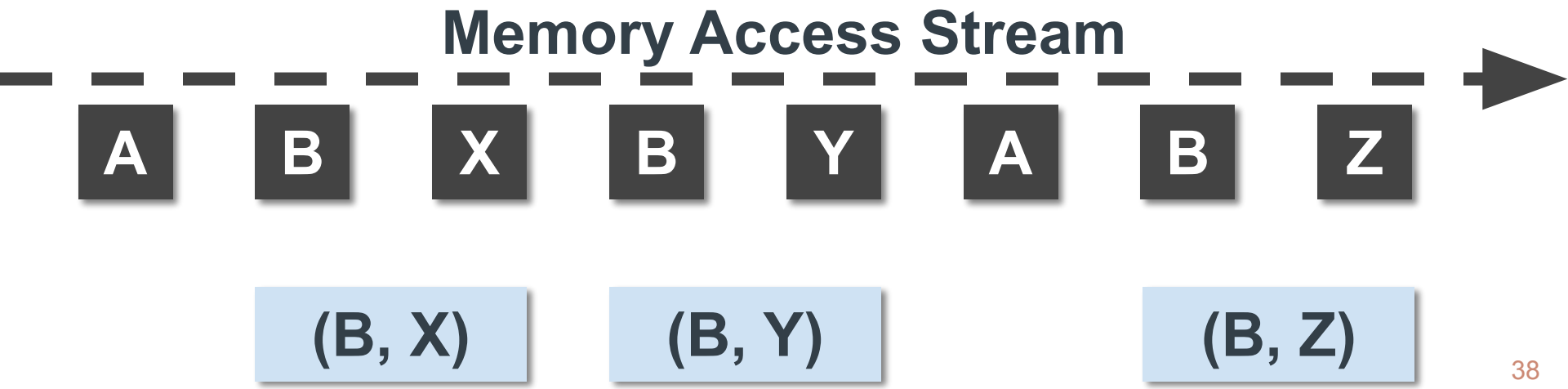
**(Q2) How should on-chip metadata *be managed?***

# Metadata Replacement Policy

**Prior work treats metadata the same as raw data**

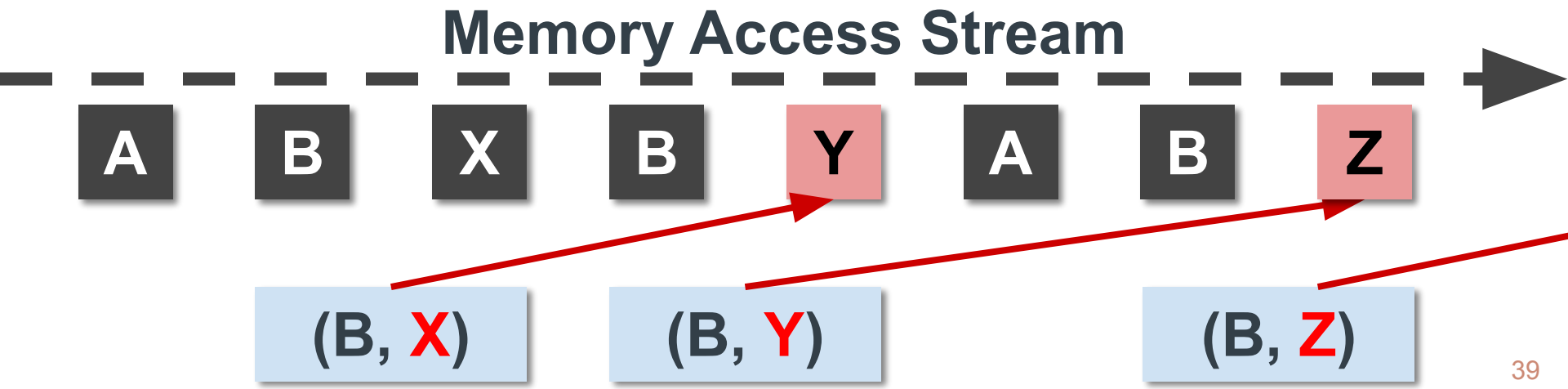
# Metadata Replacement Policy

**Prior work treats metadata the same as raw data**



# Metadata Replacement Policy

**Prior work treats metadata the same as raw data**



# Metadata Replacement Policy

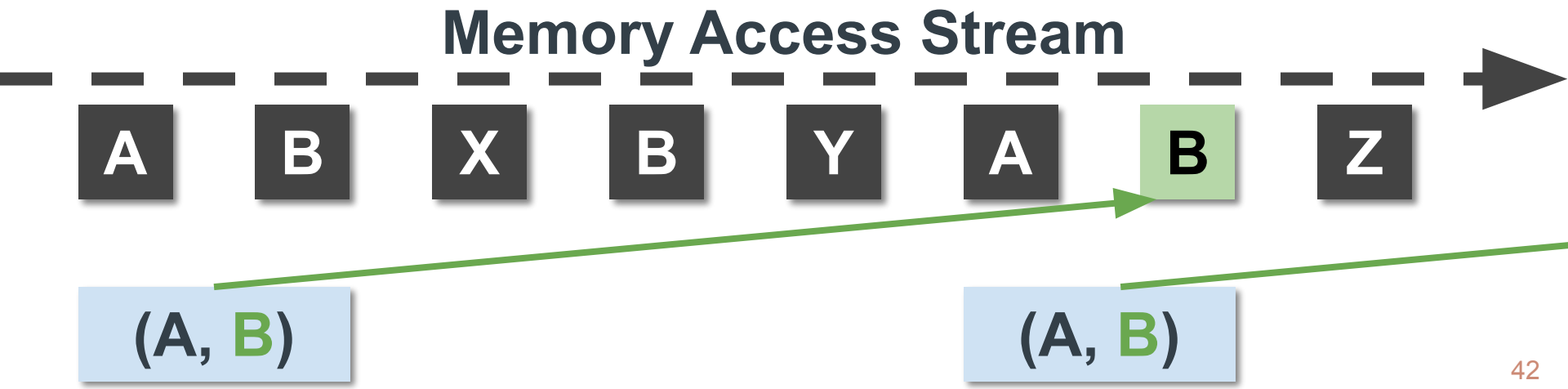
**Our work considers the prefetch utility of metadata**

# Metadata Replacement Policy

**Our work considers the prefetch utility of metadata**

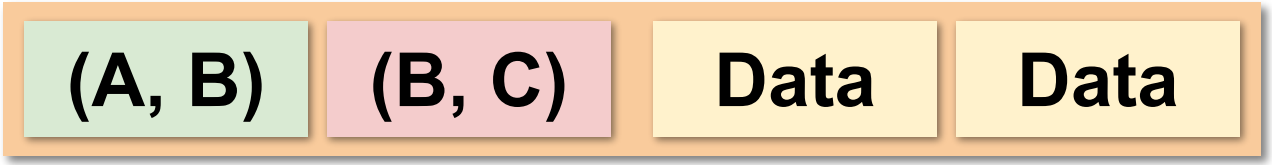
# Metadata Replacement Policy

**Our work considers the prefetch utility of metadata**



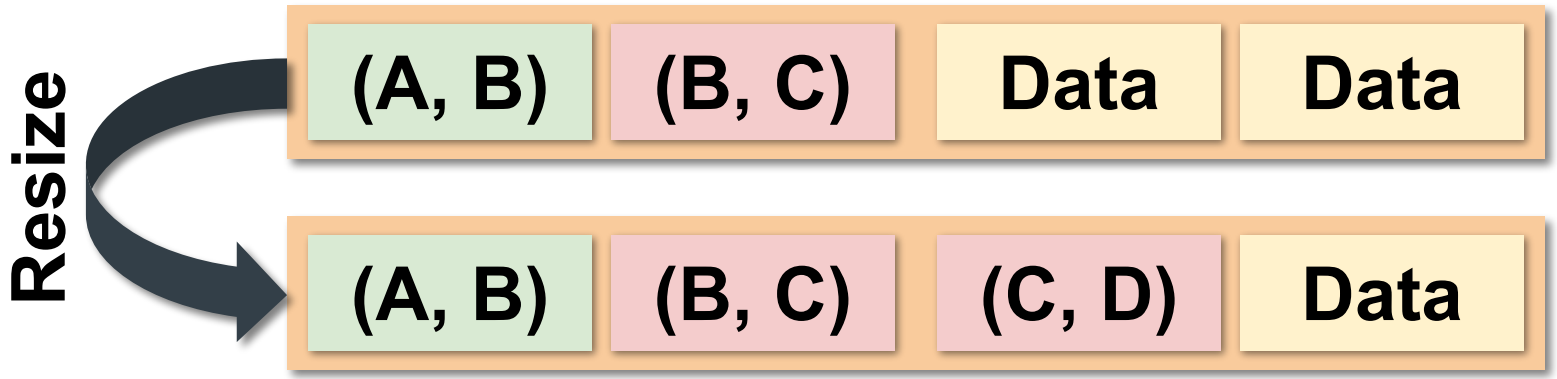
# Metadata Dynamic Partitioning

**Prior work partitions the LLC to maximize the combined data and metadata hit rate**



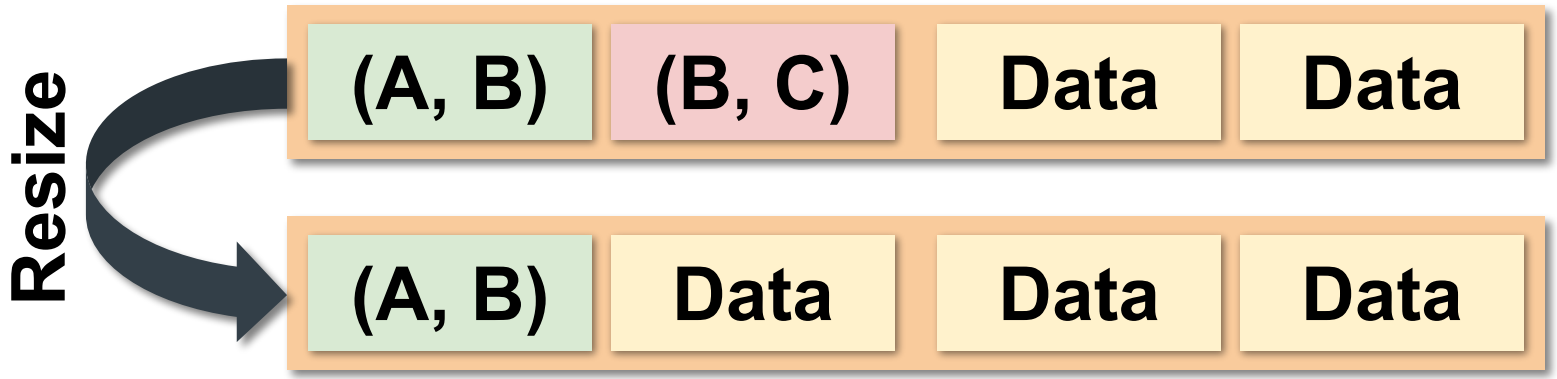
# Metadata Dynamic Partitioning

**Prior work partitions the LLC to maximize the combined data and metadata hit rate**



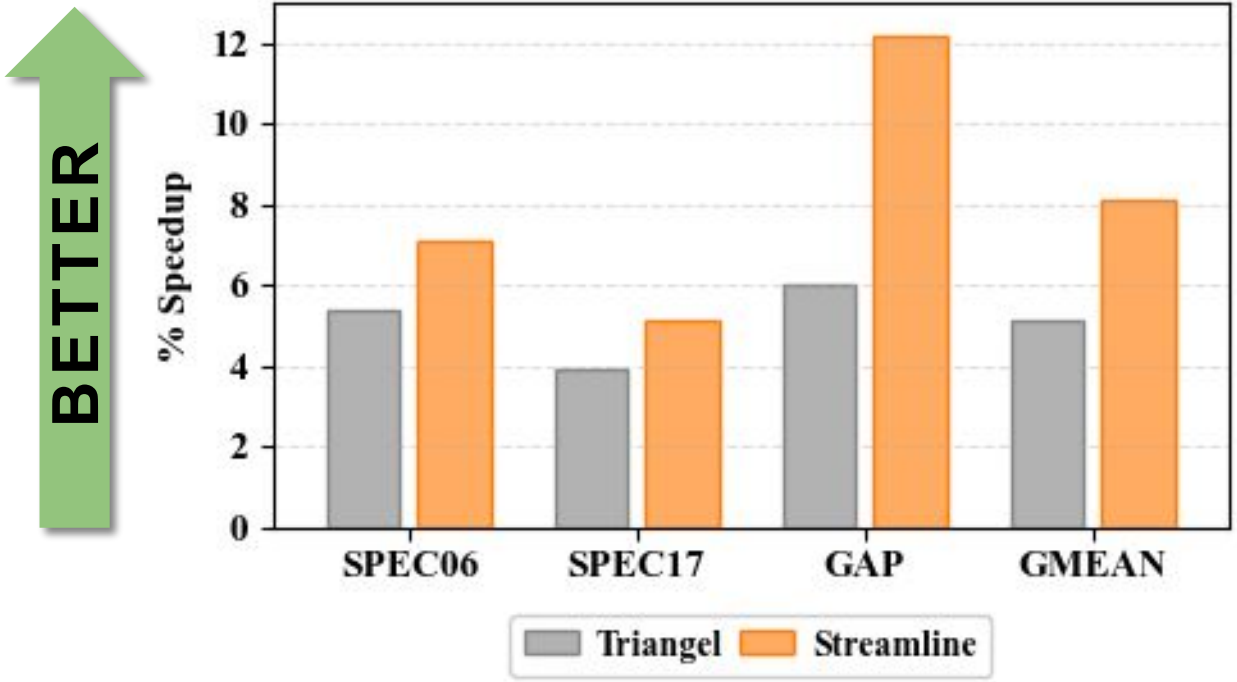
# Metadata Dynamic Partitioning

**Our work partitions the LLC to maximize the combined data and useful metadata hit rate**



# EVALUATION

# Single Core Performance

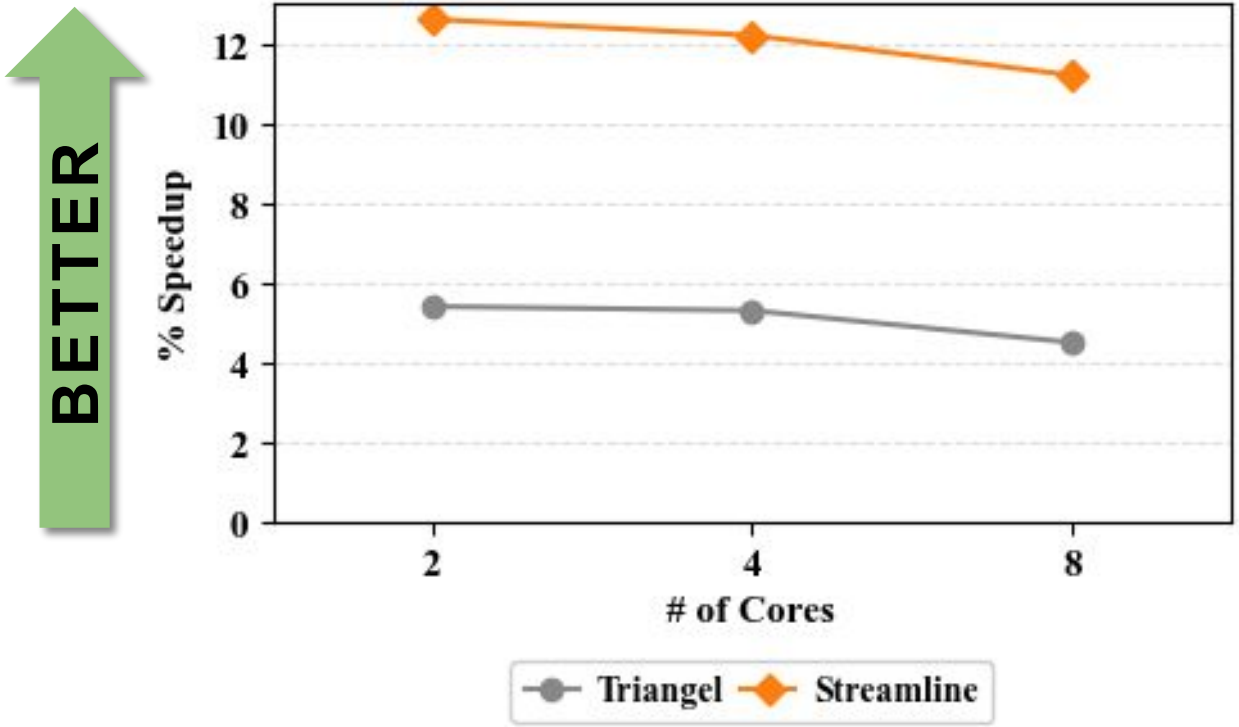



**Speedup: + 3 %pt**

**Coverage: + 12.5 %pt**

**Accuracy: + 3.6 %pt**

# Multi Core Performance

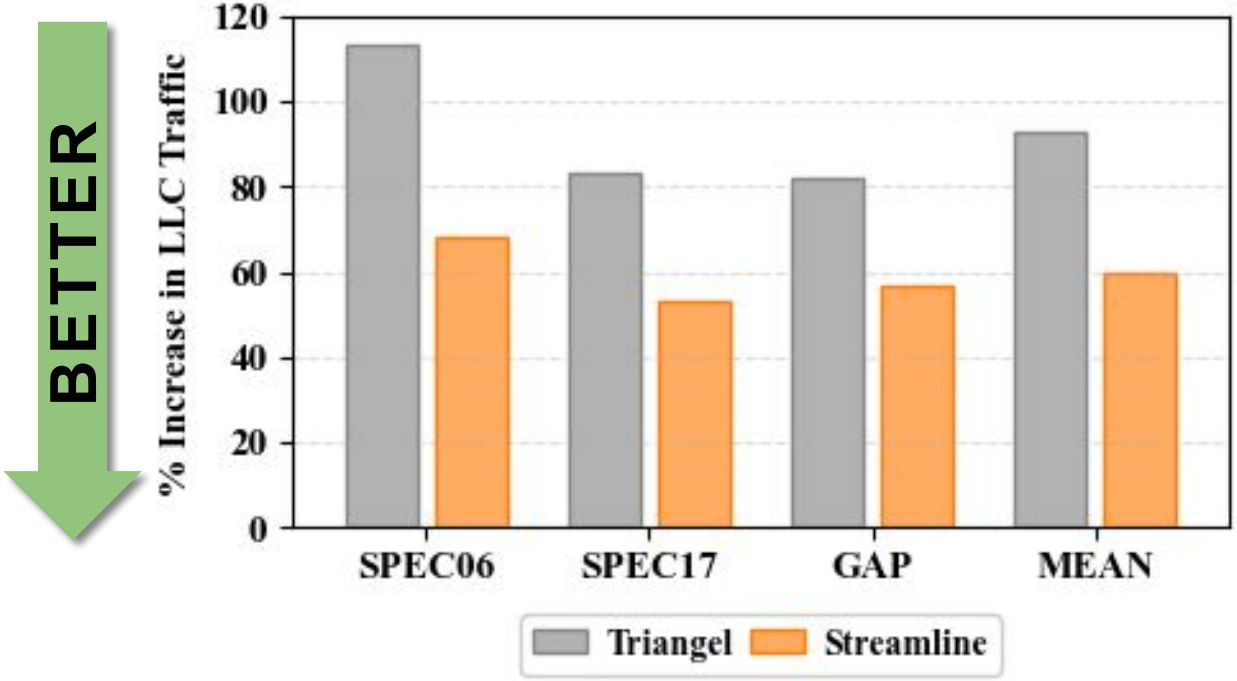



on 2-Core: + 7.2 %pt

on 4-Core: + 6.8 %pt

on 8-Core: + 6.7 %pt

# On-Chip Metadata Traffic

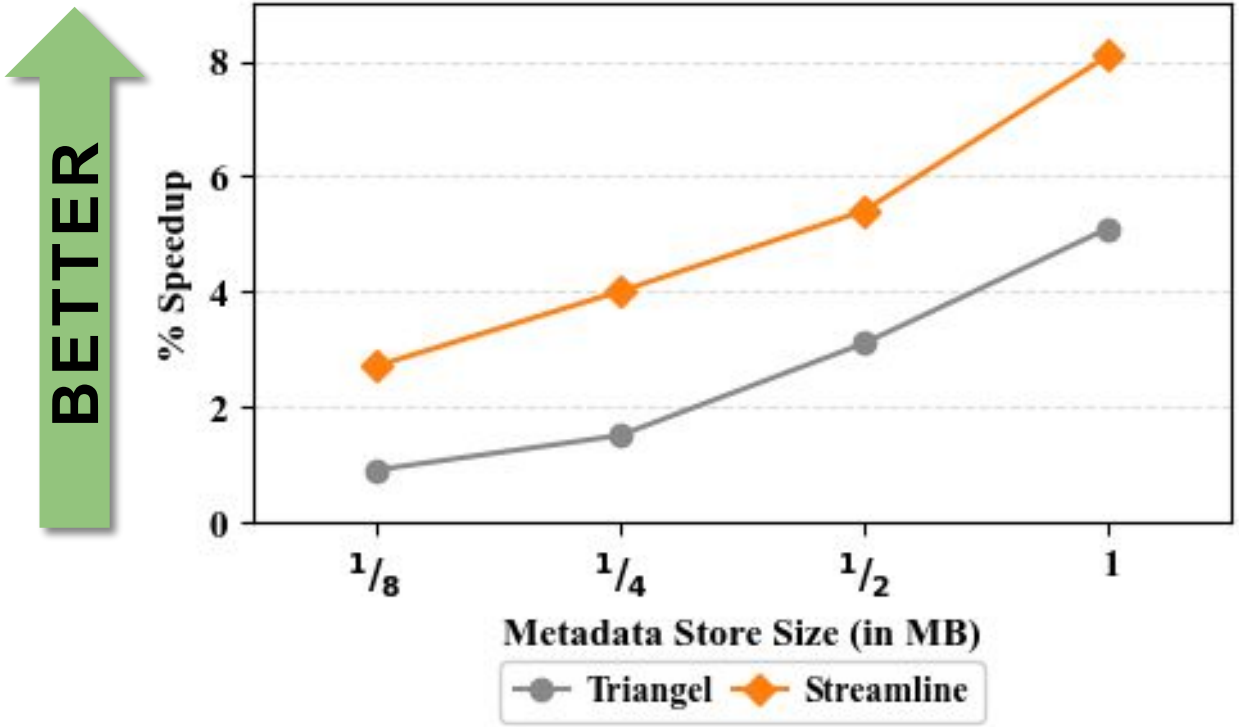



**Read Traffic: - 26 %**

**Write Traffic: - 56 %**

**All Traffic: - 37 %**

# Storage Efficiency

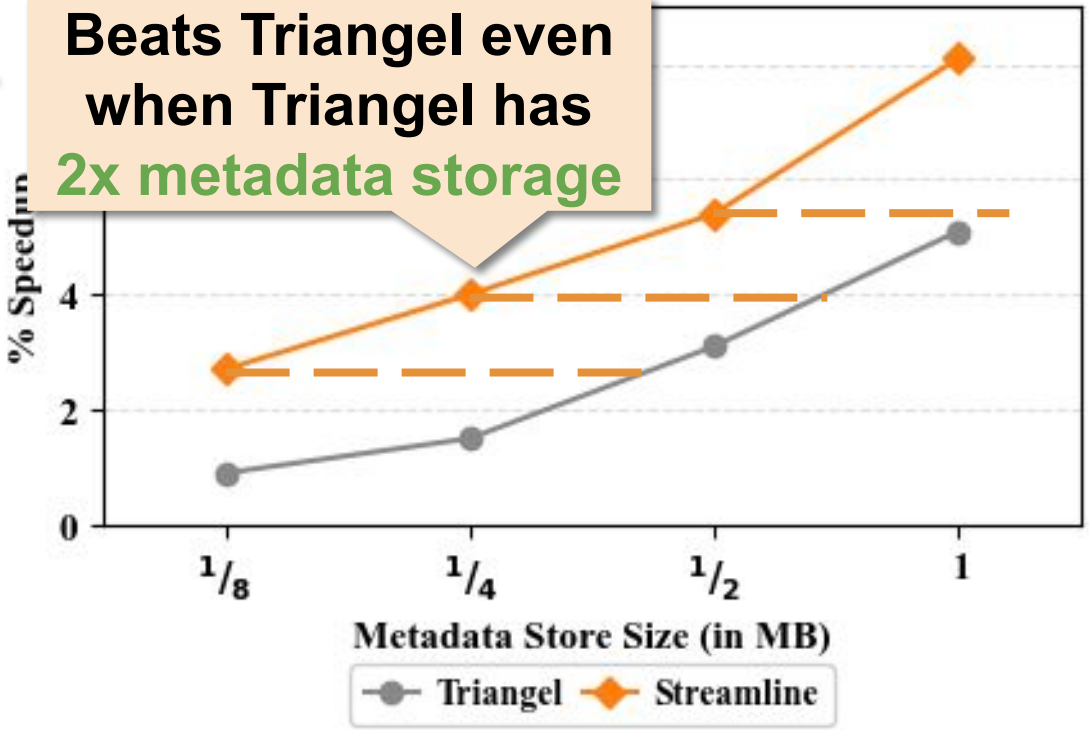



- 1/8 MB: + 1.8 %pt
- 1/4 MB: + 2.5 %pt
- 1/2 MB: + 2.3 %pt
- 1 MB: + 3 %pt

# Storage Efficiency

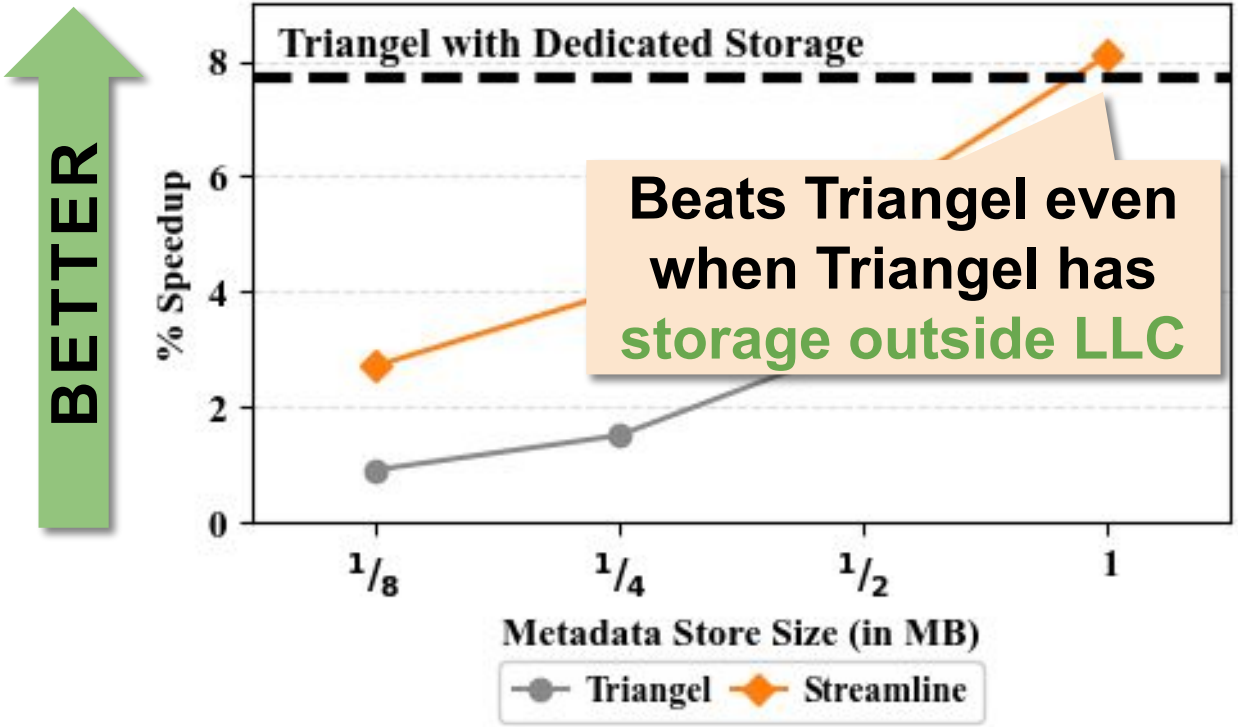
BETTER

**Beats Triangel even when Triangel has 2x metadata storage**



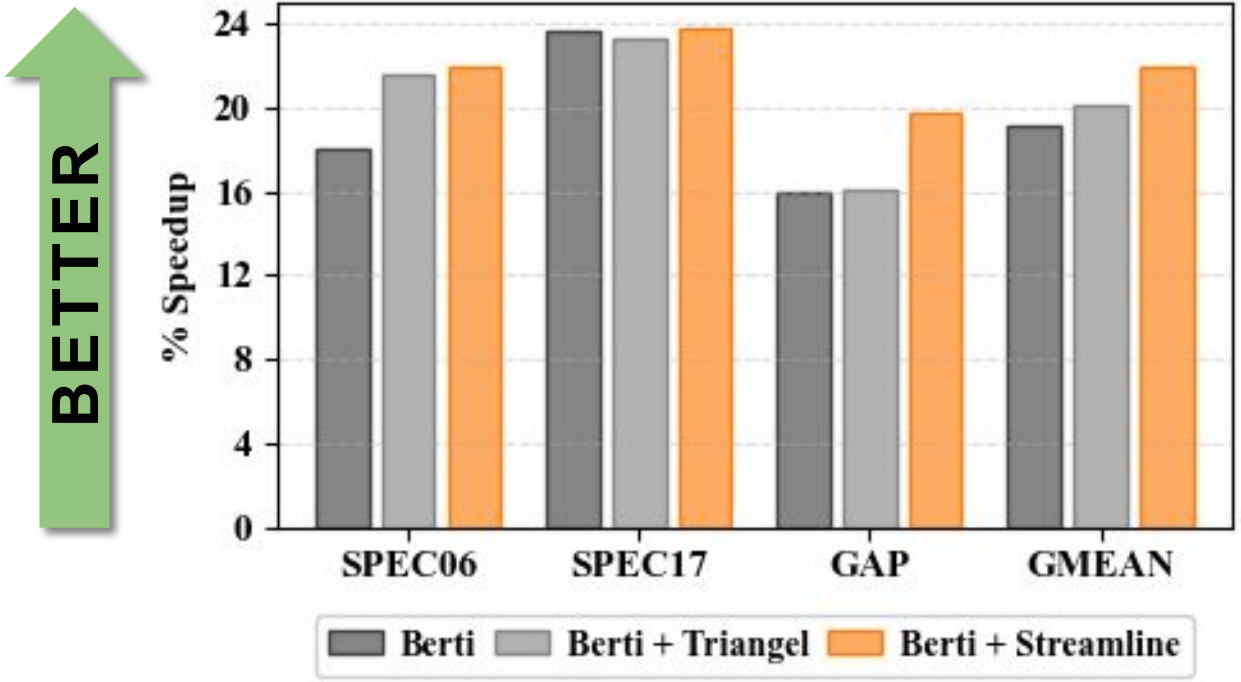

- 1/8 MB: + 1.8 %pt
- 1/4 MB: + 2.5 %pt
- 1/2 MB: + 2.3 %pt
- 1 MB: + 3 %pt

# Storage Efficiency




- 1/8 MB: + 1.8 %pt
- 1/4 MB: + 2.5 %pt
- 1/2 MB: + 2.3 %pt
- 1 MB: + 3 %pt

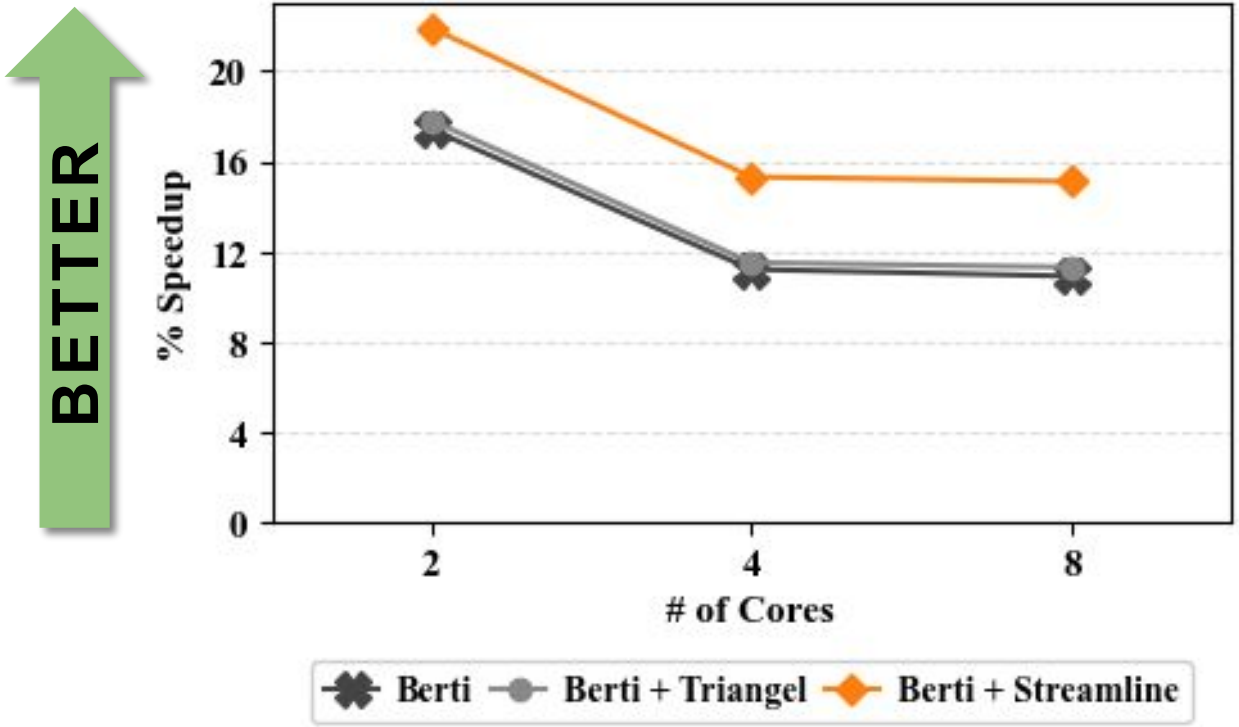
# Single Core w/ SOTA Delta Prefetcher




**vs Berti: + 3.9 %pt**

**vs Triangel: + 2.9 %pt**

# Multi Core w/ SOTA Delta Prefetcher

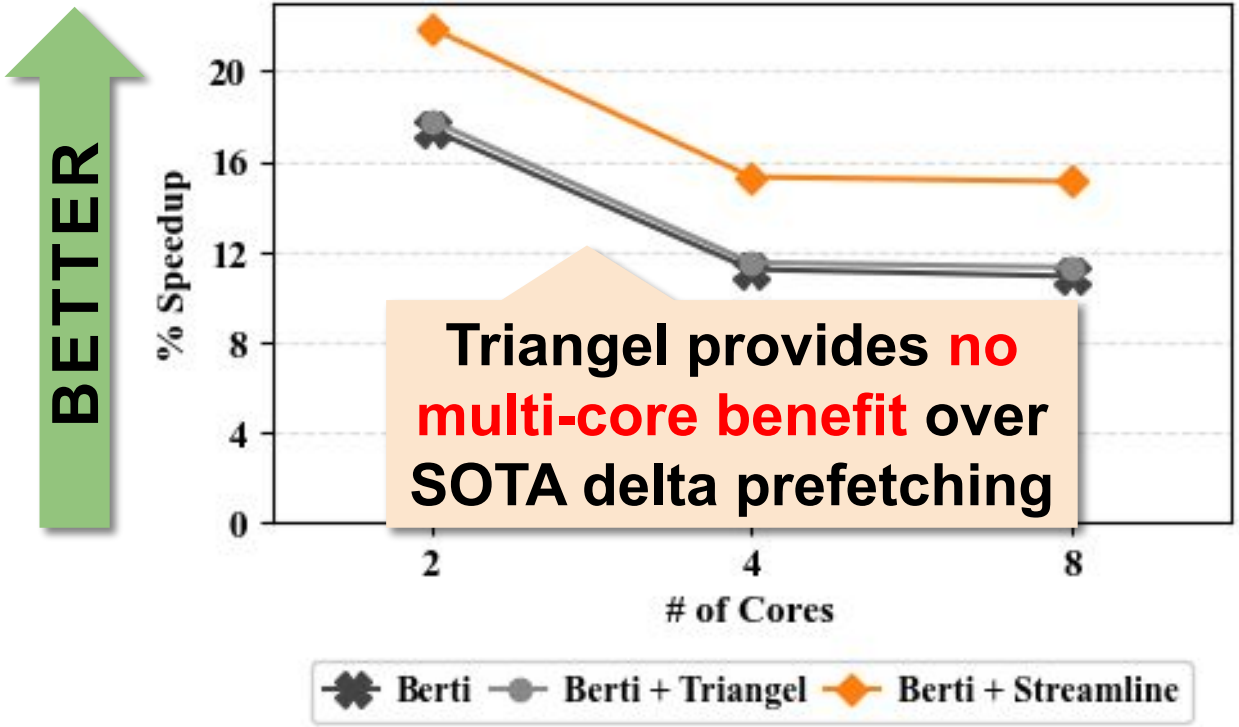



on 2-Core: + 4.1 %pt

on 4-Core: + 3.8 %pt

on 8-Core: + 3.8 %pt

# Multi Core w/ SOTA Delta Prefetcher




Triangel provides no multi-core benefit over SOTA delta prefetching

on 2-Core: + 4.1 %pt

on 4-Core: + 3.8 %pt

on 8-Core: + 3.8 %pt

# Brief Summary

**Our Streamline prefetcher leverages the *structure and the semantics of streams* in:**

**(1) our *representation* of on-chip metadata**

**(2) our *management* of on-chip metadata**

# Streamlined On-Chip Temporal Prefetching

## Thank You

Quang Duong Calvin Lin



The University of Texas at Austin
Computer Science
College of Natural Sciences