April 2018

An Application-Specific Microbenchmark for Memory Access

Mahesh Lakshminarasimhan

An Application-Specific Microbenchmark for Memory Access

Abstract

Benchmarking high performance computing systems is crucial to optimize memory consumption and maximize the performance of scientific application codes. We propose a configurable microbenchmark that explores the variations in memory bandwidth for a range of working set sizes, access patterns, and thread configurations. This framework is validated with the comparison of results from STREAM benchmark for repeated execution of the DAXPY triad kernel for both static and dynamic memory allocation. The access patterns emulate the common patterns found in simulation and modeling applications. Using application-specific access patterns we are able to refine the general roofline model for the target application.

BOISE STATE UNIVERSITY

An Application-Specific Microbenchmark for Memory Access

Mahesh Lakshminarasimhan, Dr. Catherine Olschanowsky
Department of Computer Science

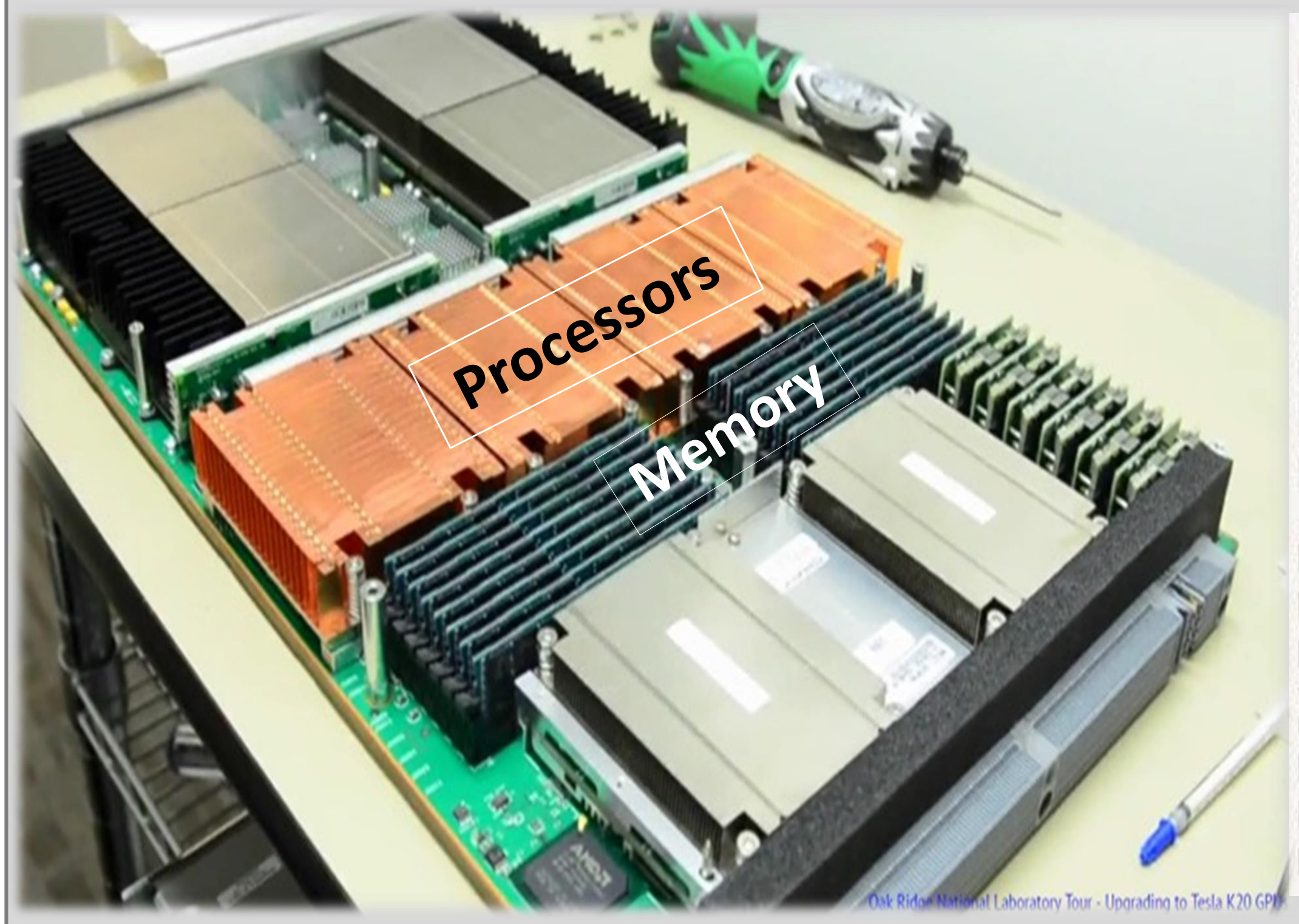
MOTIVATION

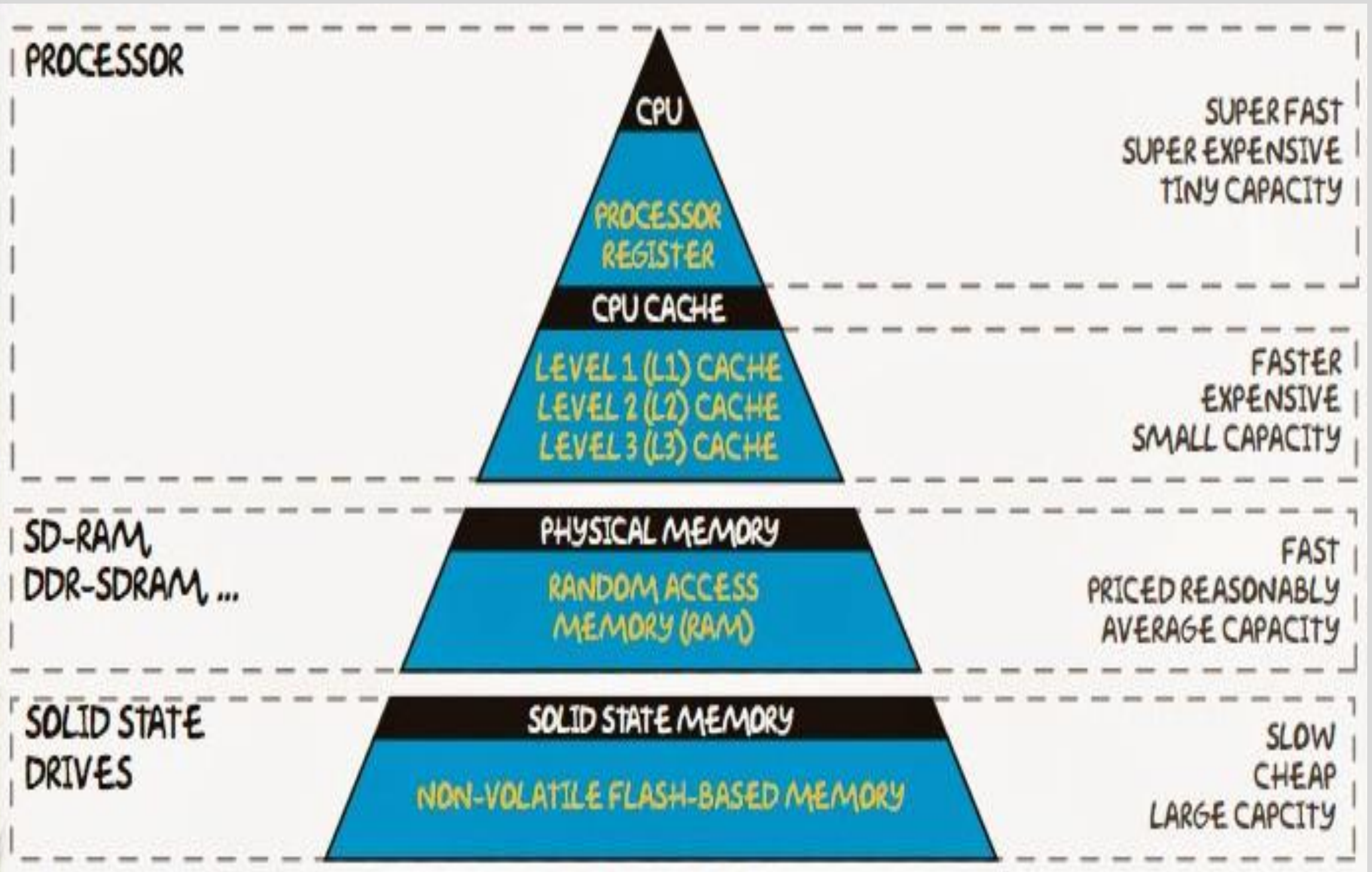
Benchmarking High Performance Computing Systems is essential to maximize scientific application performance. We present a configurable microbenchmark to advise memory optimization strategies for domain scientists.

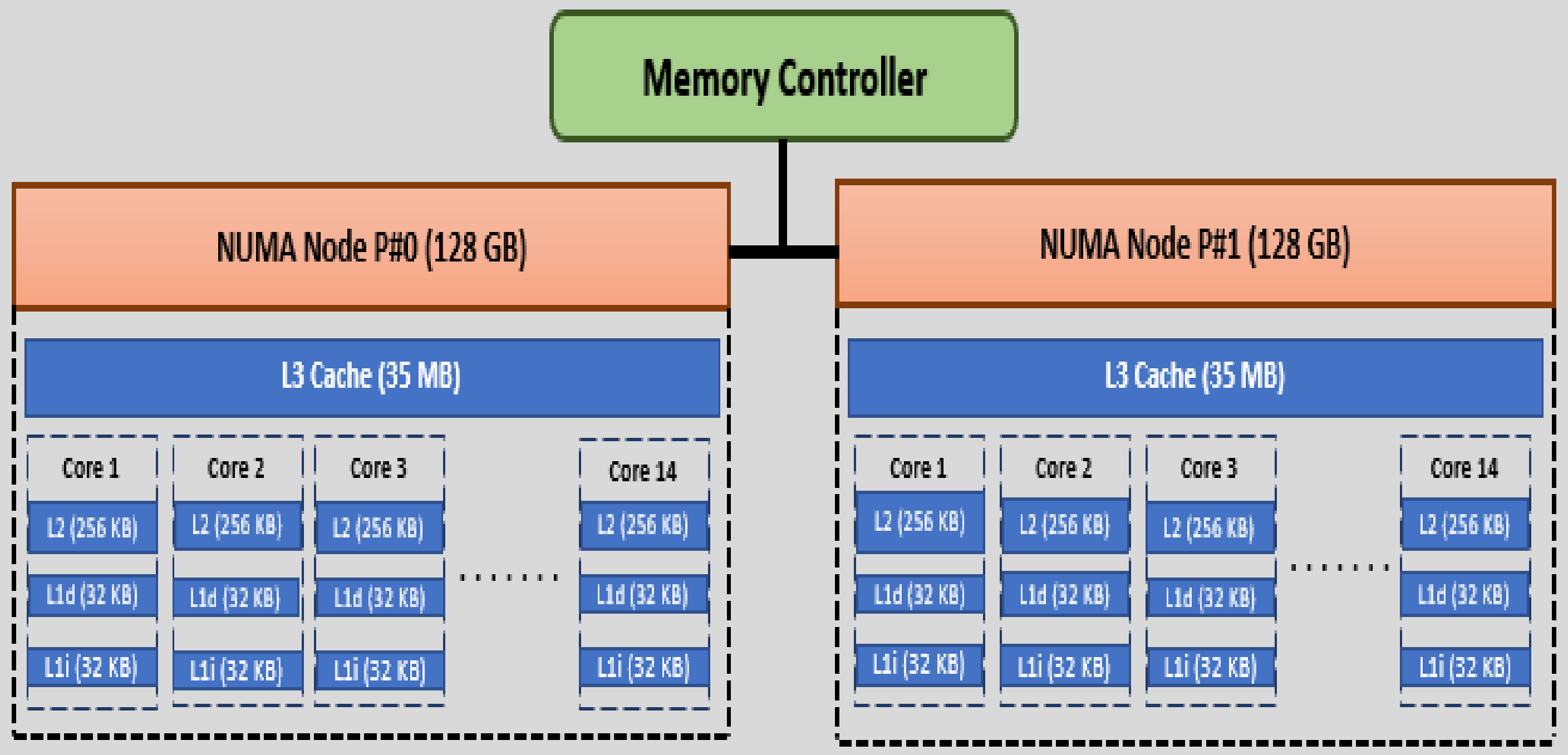
APPROACH

We explore the variation in memory bandwidth with working data set sizes across different levels of memory hierarchy for synthetic and realistic memory access patterns with different thread configurations.

MEMORY HIERARCHY IN MODERN COMPUTERS

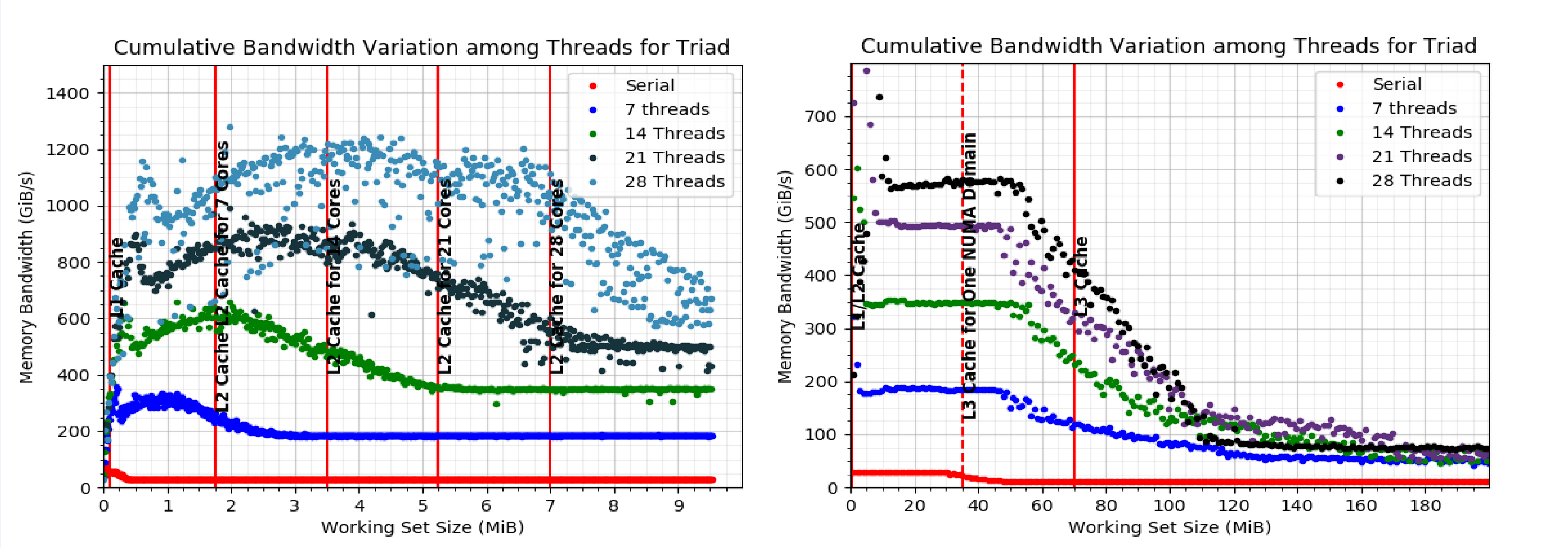






VARIATION IN CUMULATIVE MEMORY BANDWIDTH DUE TO MEMORY ACCESS PATTERNS

1. THE SYNTHETIC TRIAD KERNEL



(a) In L1/L2 Cache

(b) In Main Memory

CUSTOM BENCHMARK

CONFIGURATION <triad_run.c>

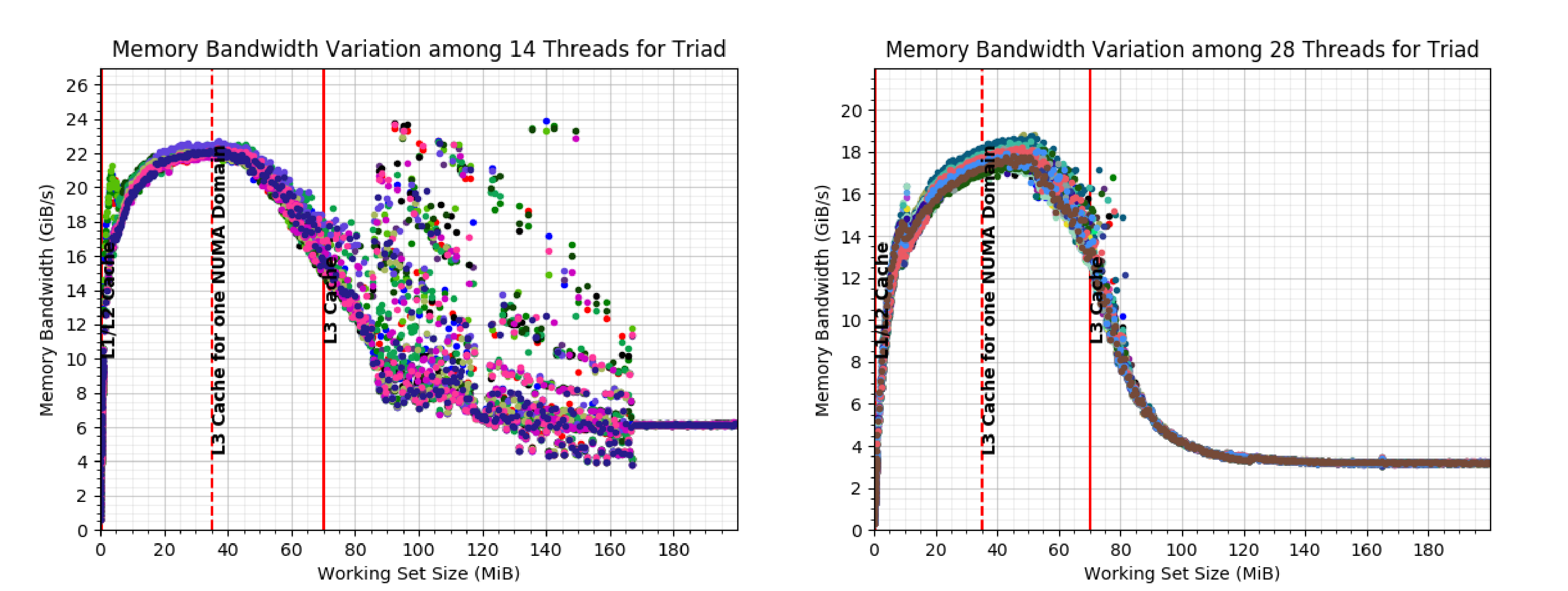
```
for(j=0; j<size; j++){
    a[j] = b[j] + d*c[j];
}
```

BENCHMARK TEMPLATE

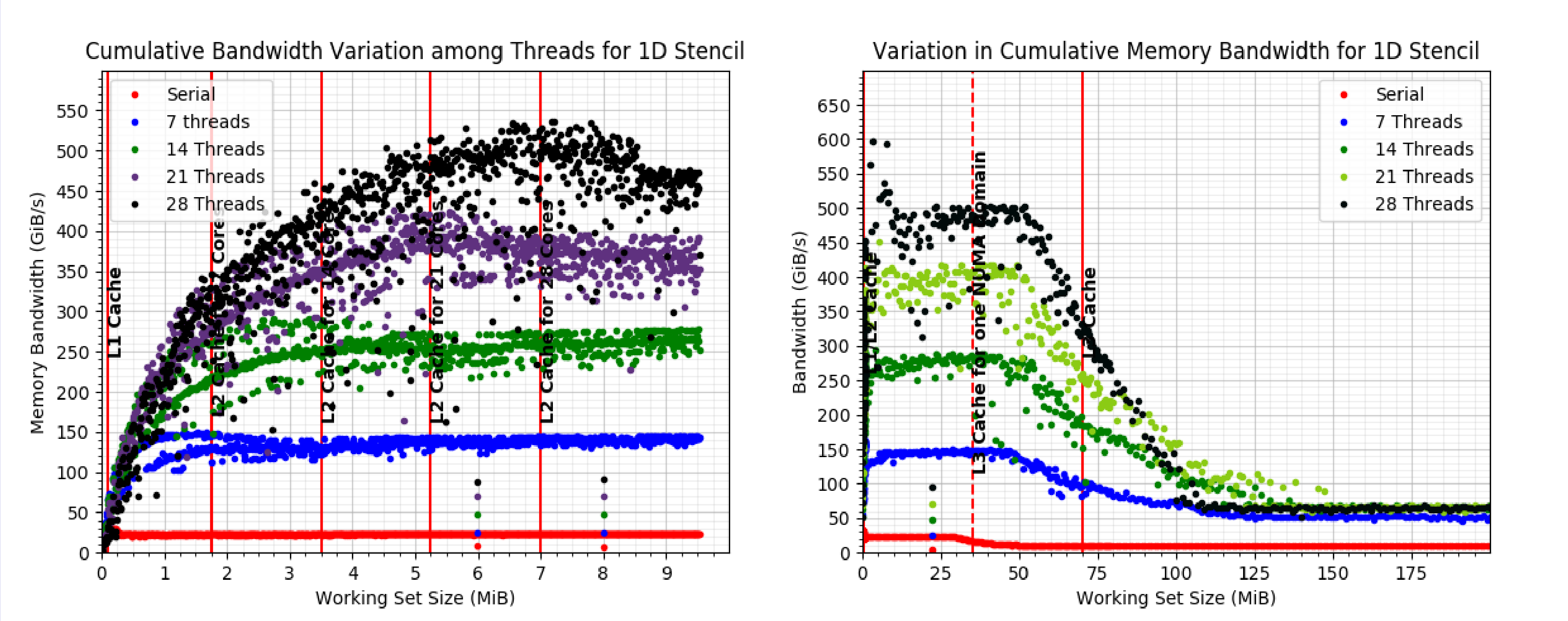
```
for(i=0; i<ntimes; i++){
    #include<triad_run.c>
}
```

PERFORMANCE VARIATION AMONG THREADS – TRIAD

We examine the variation in the memory bandwidth among each thread for 14 and 28 cores on executing the synthetic triad operation.



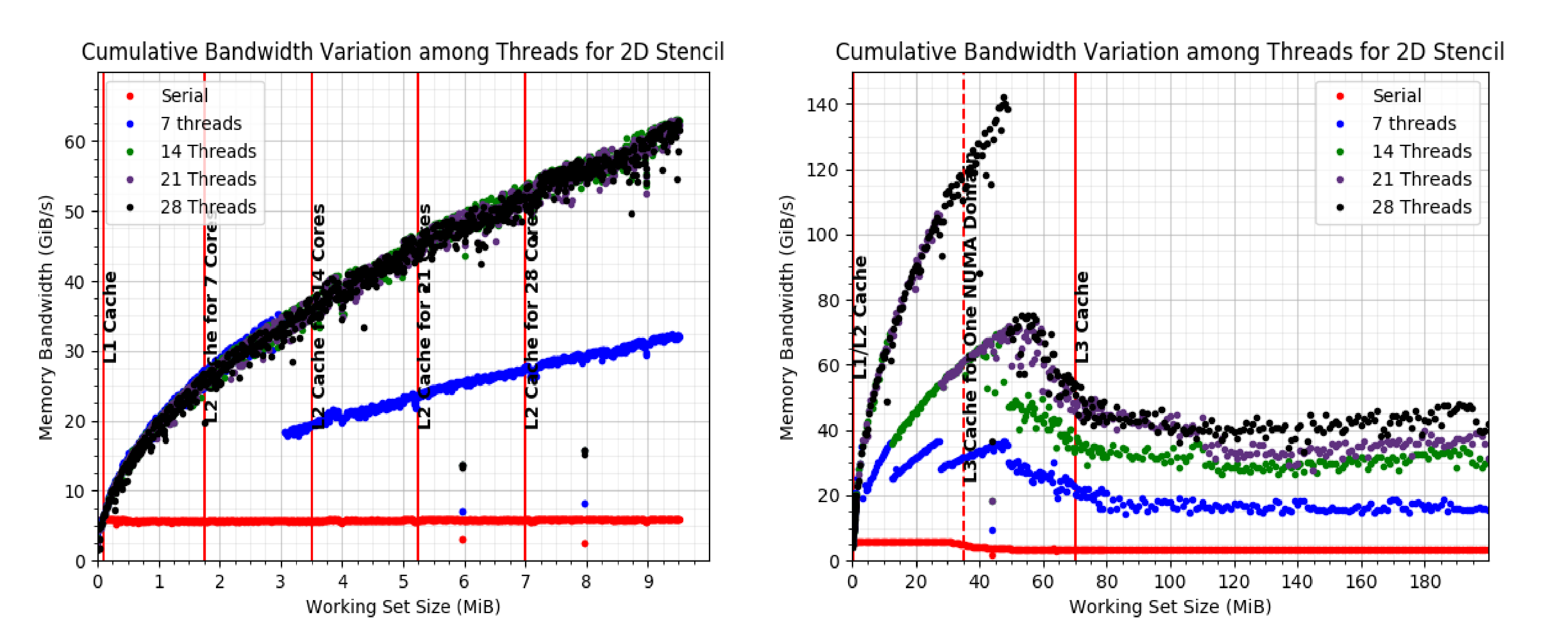
2. ONE-DIMENSIONAL STENCIL OPERATION



(a) In L1/L2 Cache

(b) In Main Memory

3. TWO-DIMENSIONAL STENCIL OPERATION



(a) In L1/L2 Cache

(b) In Main Memory

OBSERVATIONS

- Access patterns matter: The synthetic triad outperforms the more realistic multidimensional stencil patterns.
- Stencil operations, especially two dimensional stencil, struggle to scale across all levels of memory hierarchy.
- Some working set sizes are problematic and consistently yield low performance for different access patterns.
- Variation in performance increases when the working set sizes exit cache .