SAND2018-9078C

The Case for Semi-Permanent Cache Occupancy

PRESENTED BY

Matthew G. F. Dosanjh

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Authors

Matthew G. F. Dosanjh, Sandia National Laboratories

S. Mahdieh Ghazimirsaeed, Queen's University

Ryan E. Grant, Sandia National Laboratories

Whit Schonbein, Sandia National Laboratories

Michael J. Levenhagen, Sandia National Laboratores

Patrick G. Bridges, University of New Mexico

Ahmad Afsahi, Queen's University

Introduction

- Many communication libraries were designed and optimized for single threaded environments
  - The Message Passing Interface or MPI
  - Protocol processing
- These environments continue to see a radical shift
  - Multi Core
  - Many Core
  - Hardware Threads
- Usage of communication libraries is adapting
- The performance issues we have observed are bound by memory latency
- Thesis – By manipulating cache behavior, we can reduce these latencies increasing performance

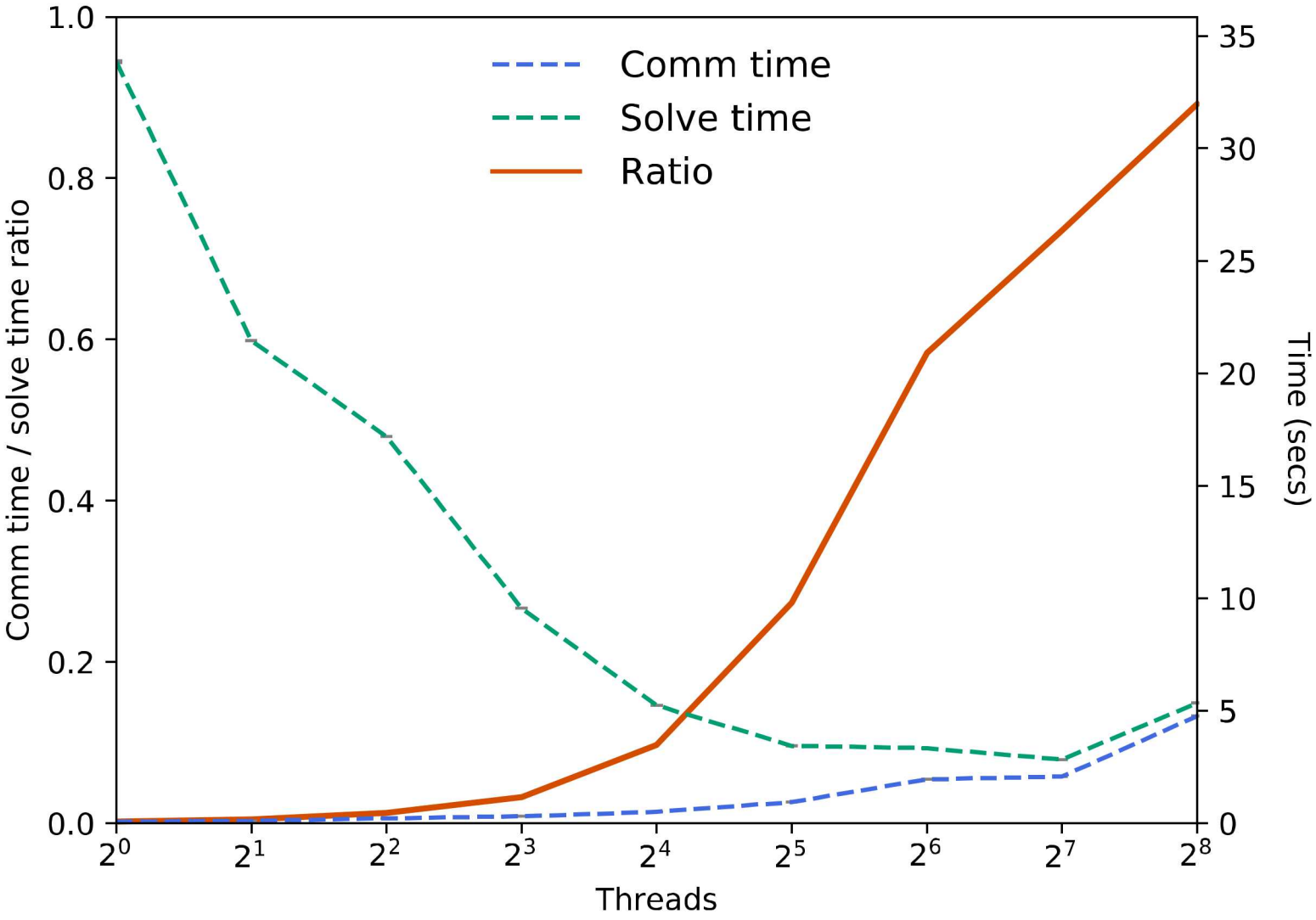
Background - MPI

- MPI is the dominant HPC communication model
- Point-to-point operations
  - Send/Recv
  - Wild-cards
  - Ordering
- Four Thread Models
  - MPI_THREAD_SINGLE
  - MPI_THREAD_FUNNLED
  - MPI_THREAD_SERIALIZED
  - MPI_THREAD_MULTIPLE
- Exascale application developers want to use MPI in a multithreaded manner
  - D. E. Bernholdt, S. Boehm, G. Bosilca, M. Venkata, R. E. Grant, T. Naughton, H. Pritchard, and G. Vallee. A survey of MPI usage in the U.S. exascale computing project. Concurrency and Computation: Practice and Experience, 2017. in press.

The New Computing Environment

- Core counts continue to increase
  - Cori/Trinity – 68 cores or 272 threads
  - Astra – 56 cores or 224 threads
- Applications leverage multithreaded runtimes to use these resources
  - Mahesh Rajan, Douglas W. Doerfler, and Simon D. Hammond. 2015. Trinity Benchmarks on Intel Xeon Phi. Technical Report. Sandia National Laboratories.
- Multithreaded system software access
  - Mechanics – Synchronization, locks, etc.
  - Behaviors – Non-determinism, ordering, etc.
- The mechanics are well studied, but behaviors are not
- What happens when we use multithread communication in a highly optimized application?

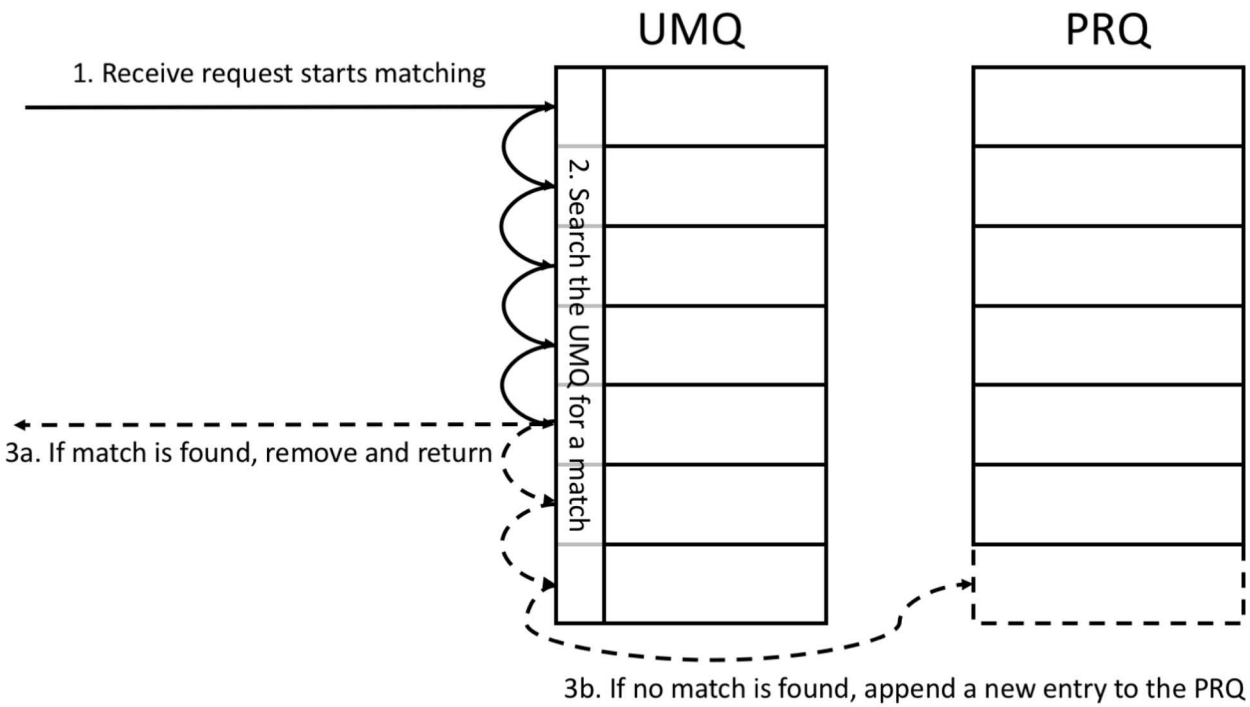
The Effect of Multithreaded Communication on MiniFE



Potential Effects on MPI message processing

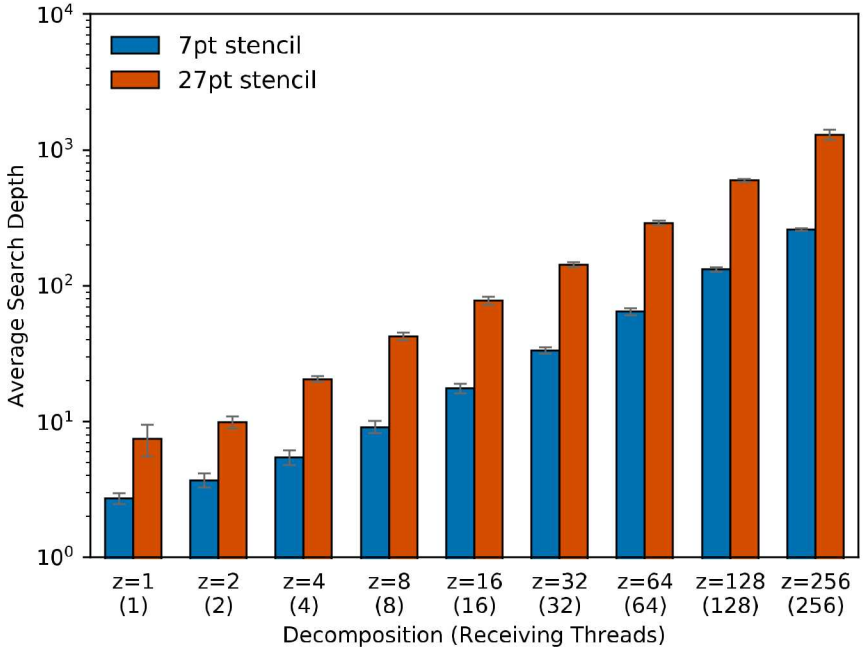
- Message Matching is a processing step for MPI Communication
- It ensures MPI's Send-Receive ordering requirements
- Matches incoming data to a buffer based on three identifiers
  - ContextID – Identifies the communicator
  - Rank – Address of the sending process
  - Tag – A user specified identifier
- Increasing threads
  - More messages to process
  - Increased ordering variance

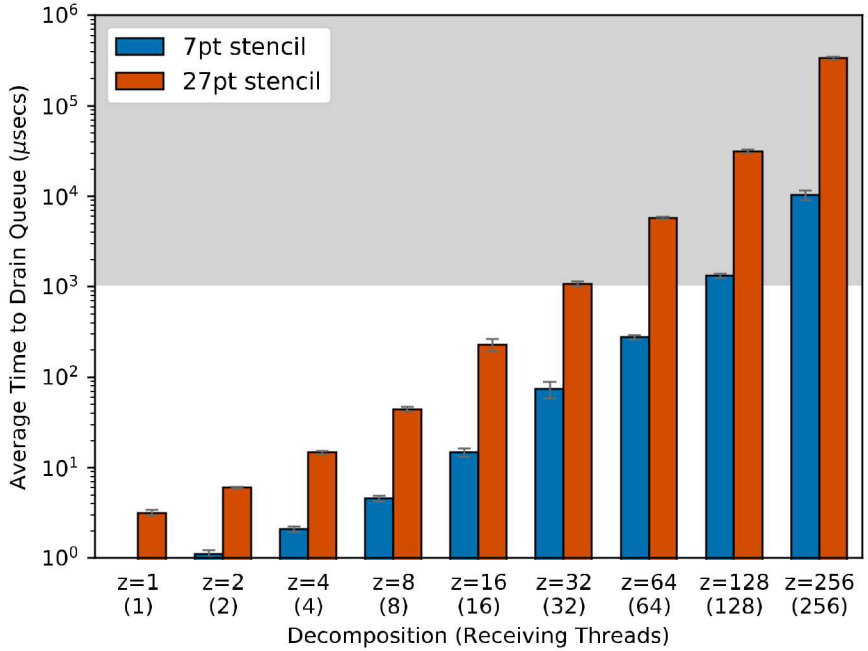
Message Matching Traditional Model



The Impact of Multithreaded Non-determinism

- Modeling multithreaded BPS applications
- Average Search depths exceed 1000 elements
- Matching drain times can exceed iteration targets (grey region)
  - Molecular Dynamics Codes such as LAMMPS and GROMACS





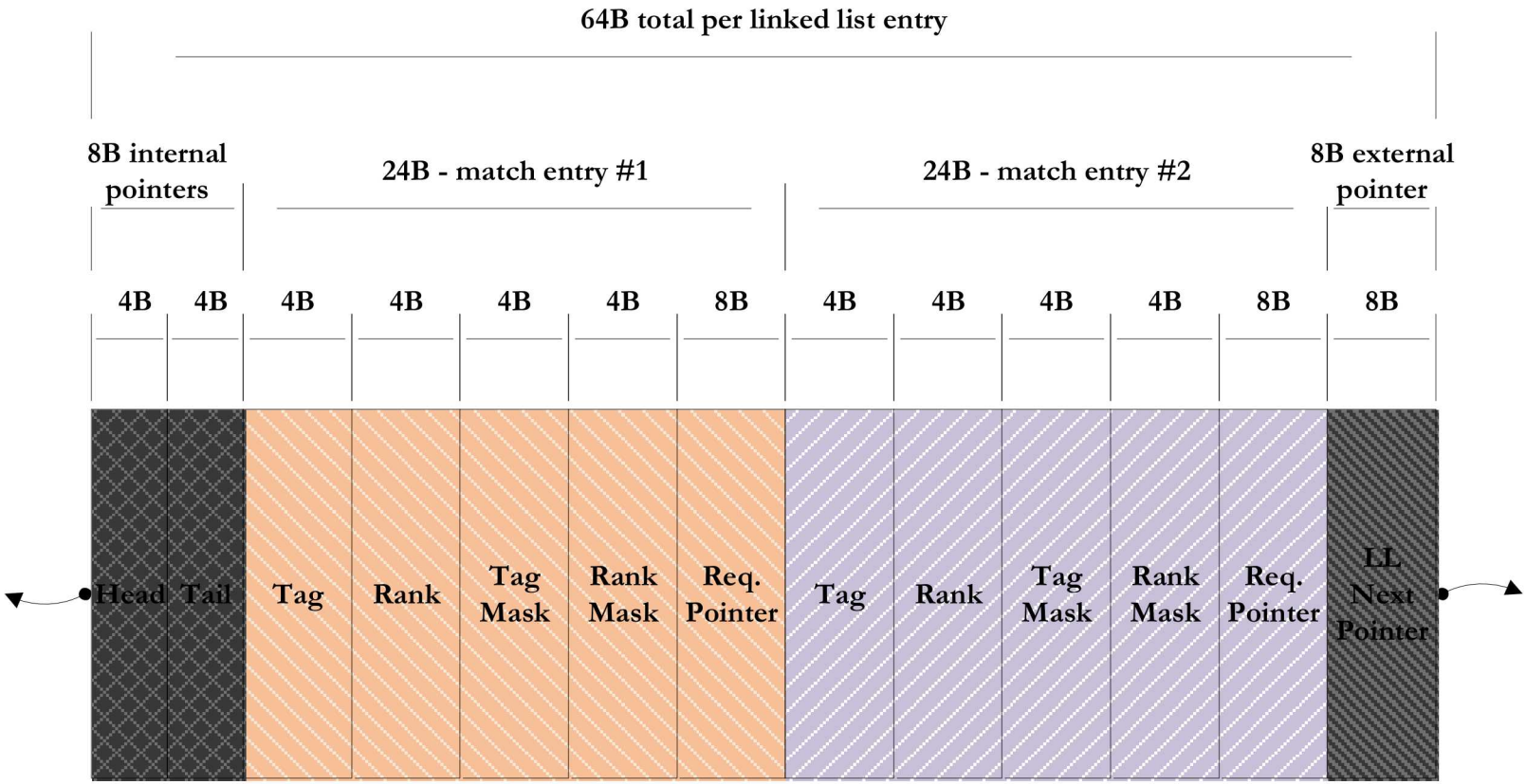
W. Schonbein, M. G. F. Dosanjh, et al. Measuring multithreaded message matching misery. In *Proceedings of the International European Conference on Parallel and Distributed Computing*. 2018.

Data Locality

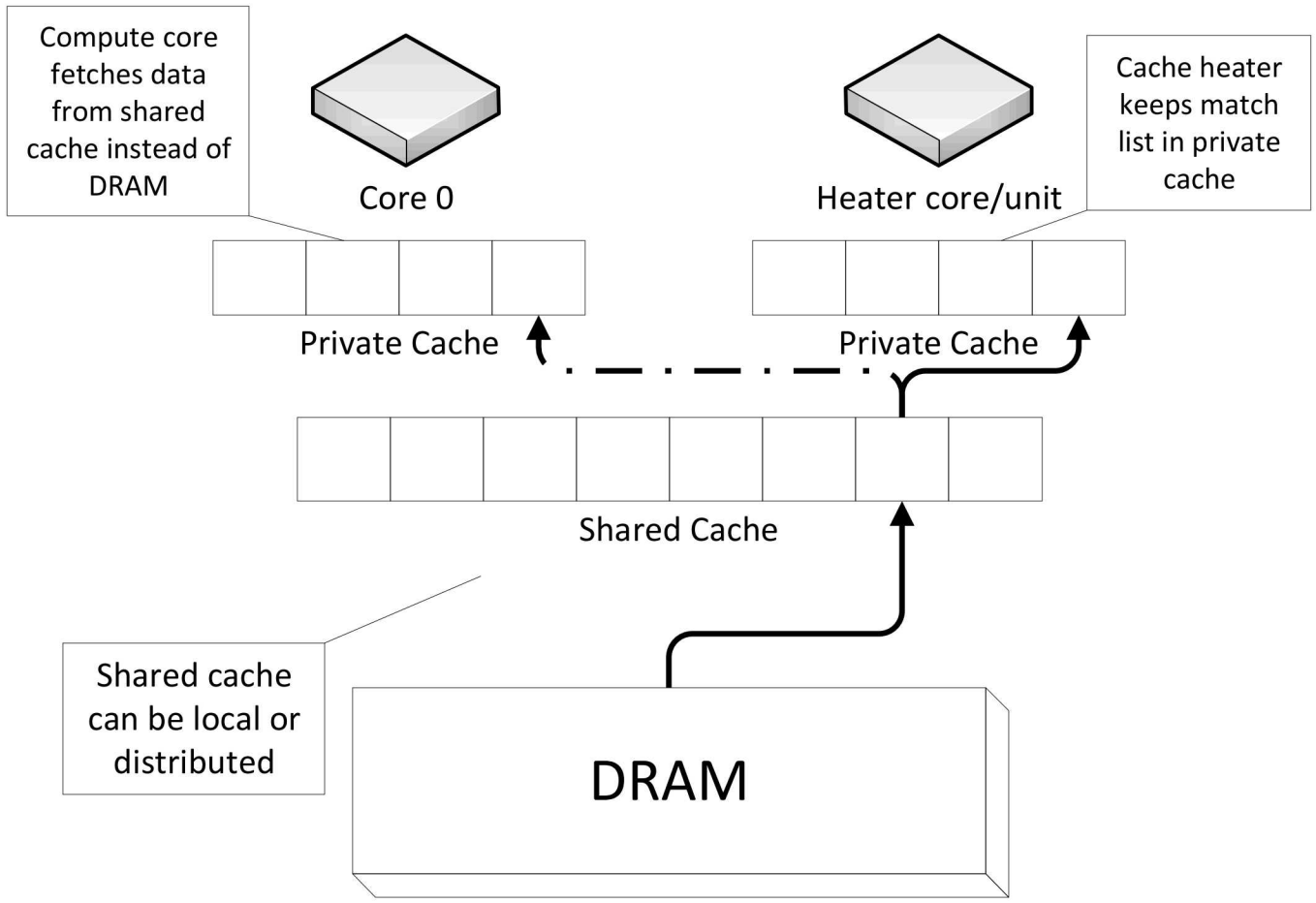
- MPI Message Matching performance is limited by the performance of data access
  - Linked lists elements are non contiguous in memory and span a number of cache lines
  - Linked list searching is difficult for the prefetcher
- How much of this can we improve by eliminating cache misses?
- Most architectures don't provide a cache control interface
- Increasing data locality will reduce the cost of data access
  - Spatial – combining multiple element into contiguous memory
  - Temporal – accessing the list periodically to ensure that the list isn't evicted from cache
- Tools to control locality
  - Spatial Locality - Linked List of Arrays
  - Temporal Locality -Hot Caching

Linked List of Arrays

- A new matching data structure
- Increases spatial locality by leveraging contiguous memory
- Fits multiple elements per cache line



Hot Caching


Compute core fetches data from shared cache instead of DRAM
Core 0
Heater core/unit
Cache heater keeps match list in private cache
Private Cache
Private Cache
Shared Cache
Shared cache can be local or distributed
DRAM

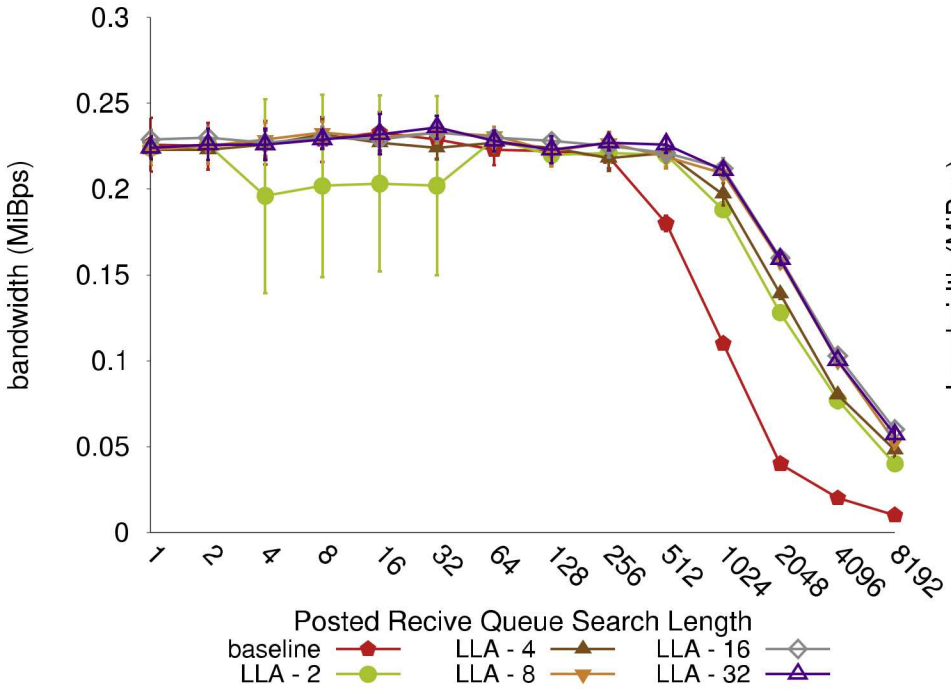
Implementation Notes

- Linked list of arrays has modifiable length of entries per array
- Hot caching and linked list of arrays can be combined
  - Increases both spatial and temporal locality
  - Better control of memory allows from linked list of arrays implementation allows for reduced registration overheads in hot caching

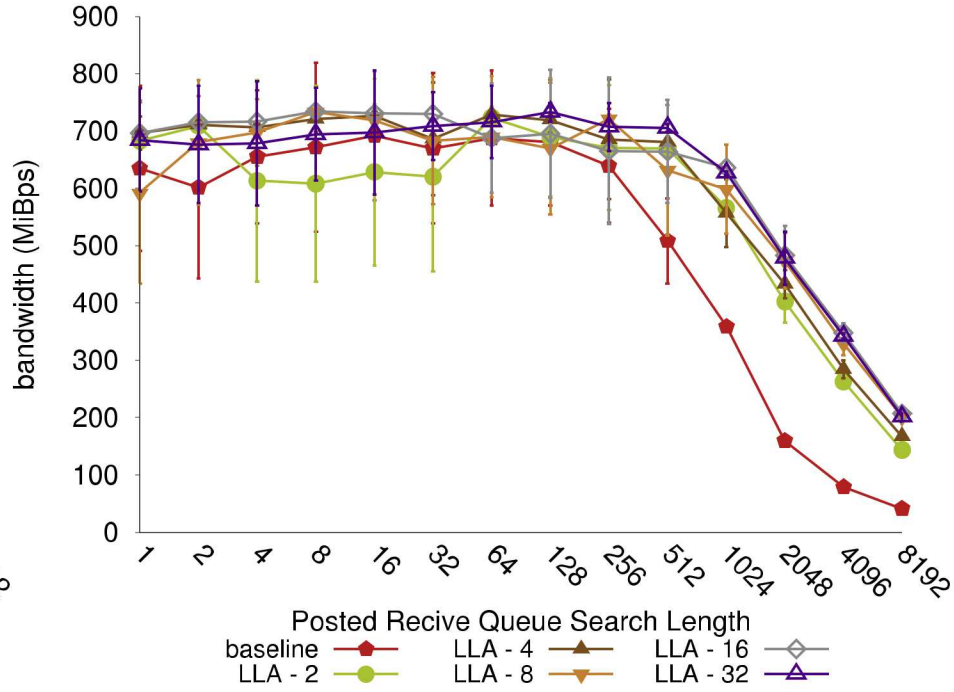
Experimental Setup

- Modified MVAPICH with new matching engines
  - Linked List of Arrays
  - Hot Caching
  - Combination
- Modified OSU Bandwidth Microbenchmark
  - Cache clearing
  - Pre-posting recvs
  - Control of search depths
- Two systems
  - Sandy Bridge
  - Broadwell

Spatial Locality – Sandy Bridge

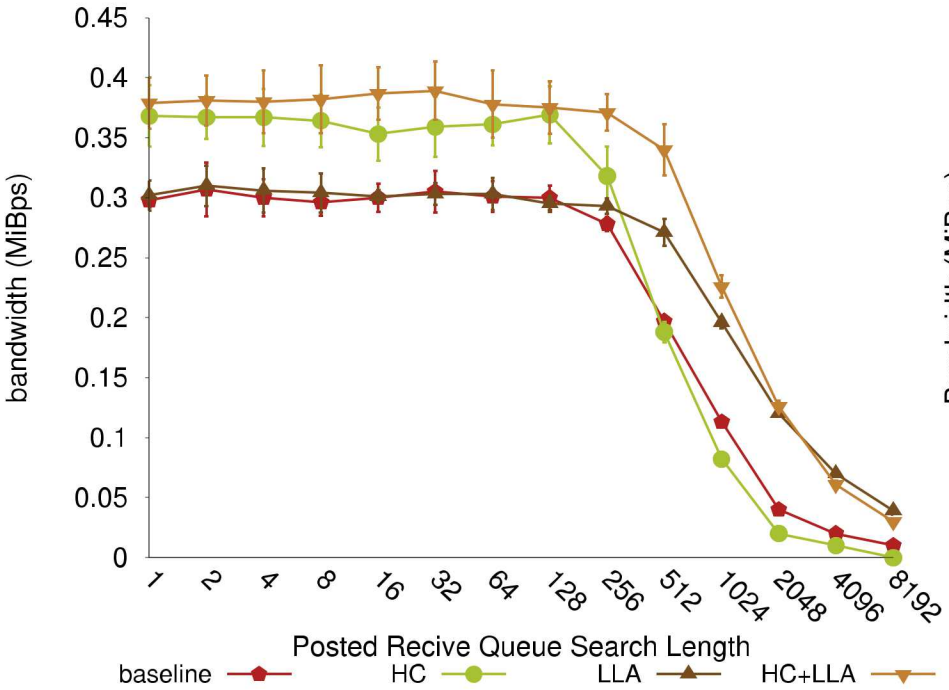


Small Messages (1B)

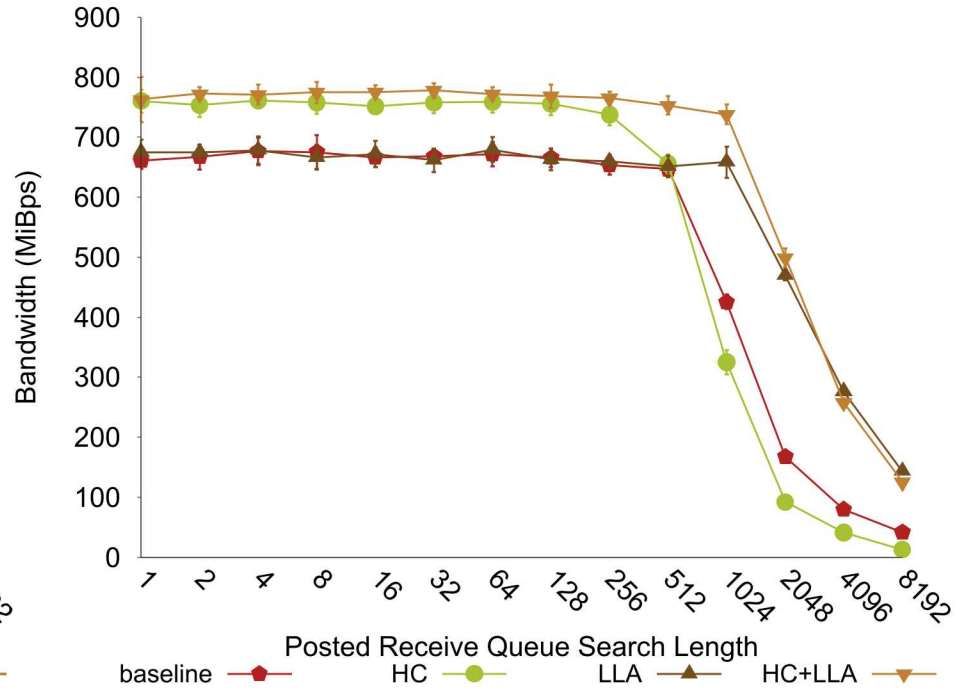


Medium Messages (4KiB)

Temporal Locality – Sandy Bridge

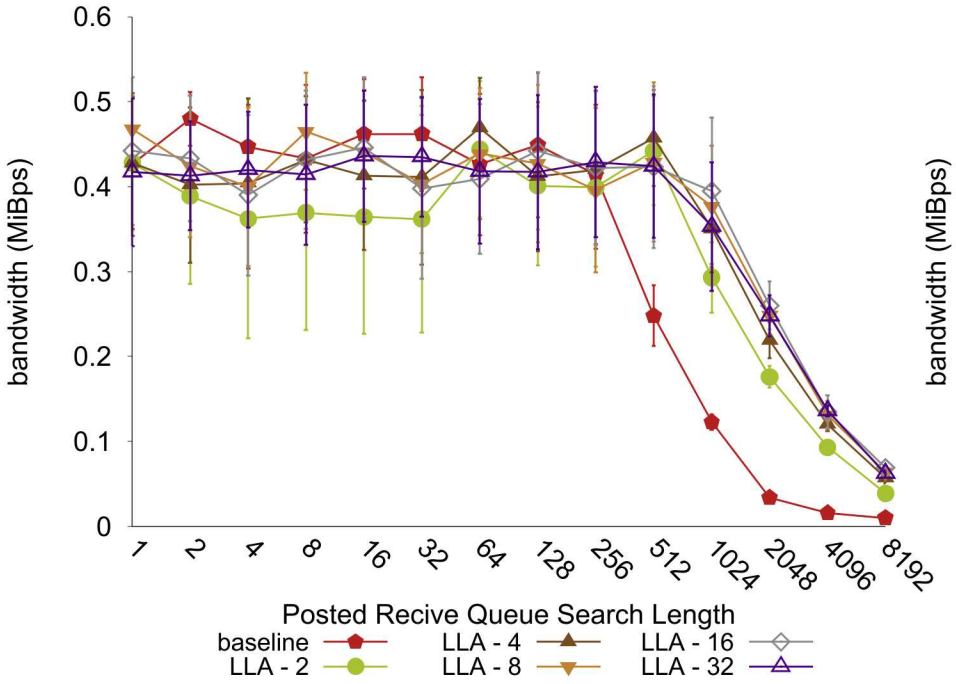


Small Messages (1B)

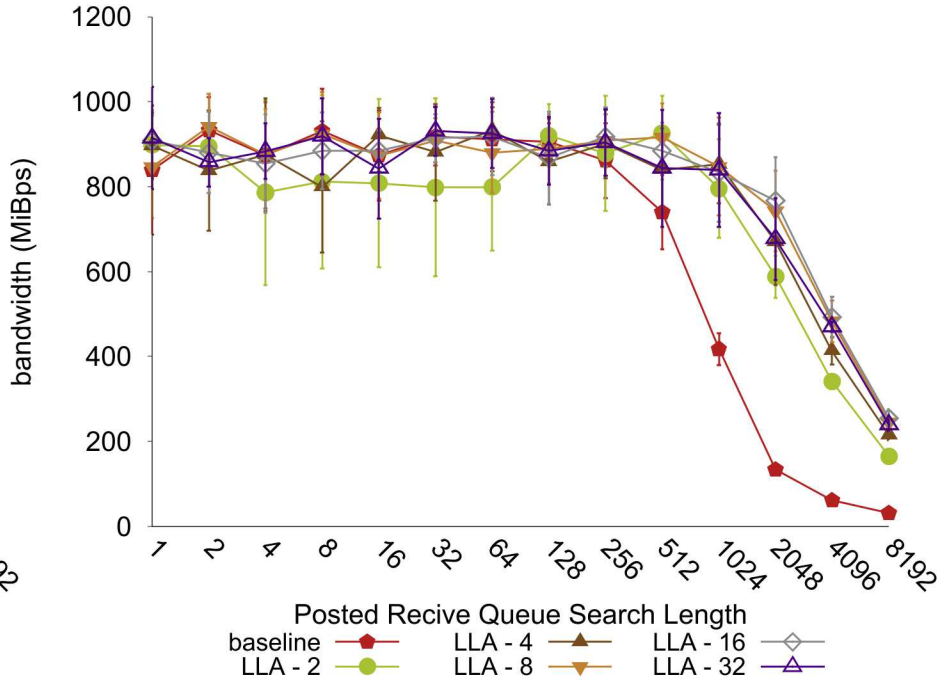


Medium Messages (4KiB)

Spatial Locality – Broadwell

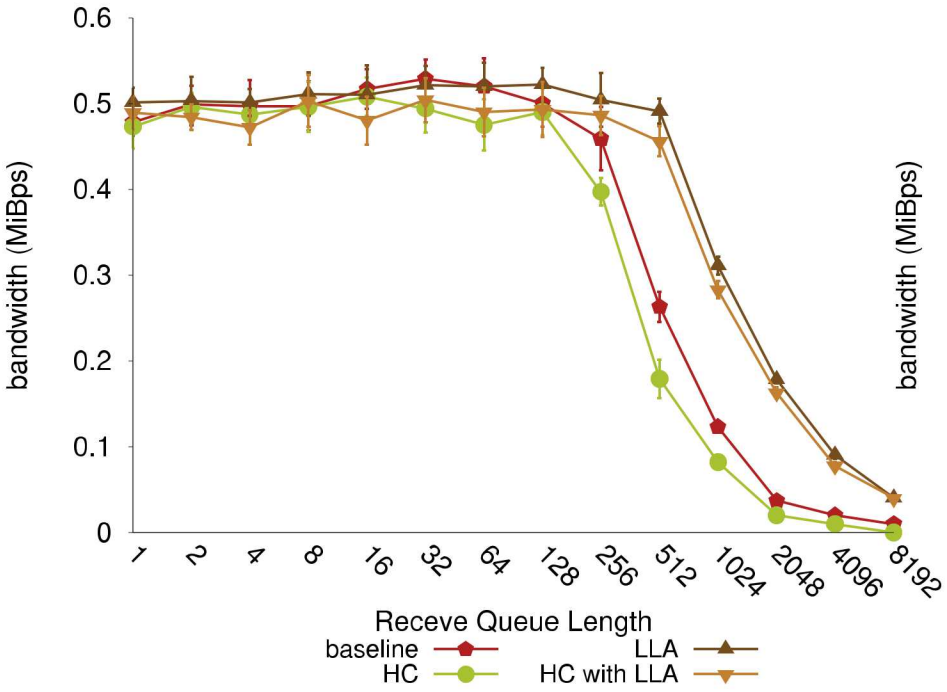


Small Messages (1B)

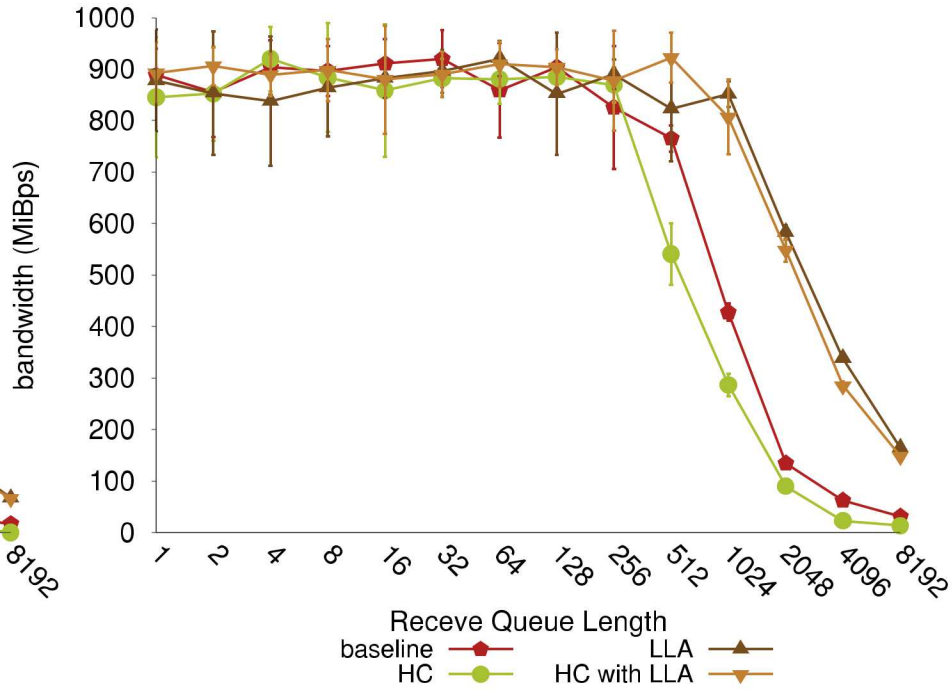


Medium Messages (4KiB)

Temporal Locality – Broadwell



Small Messages (1B)



Medium Messages (4KiB)

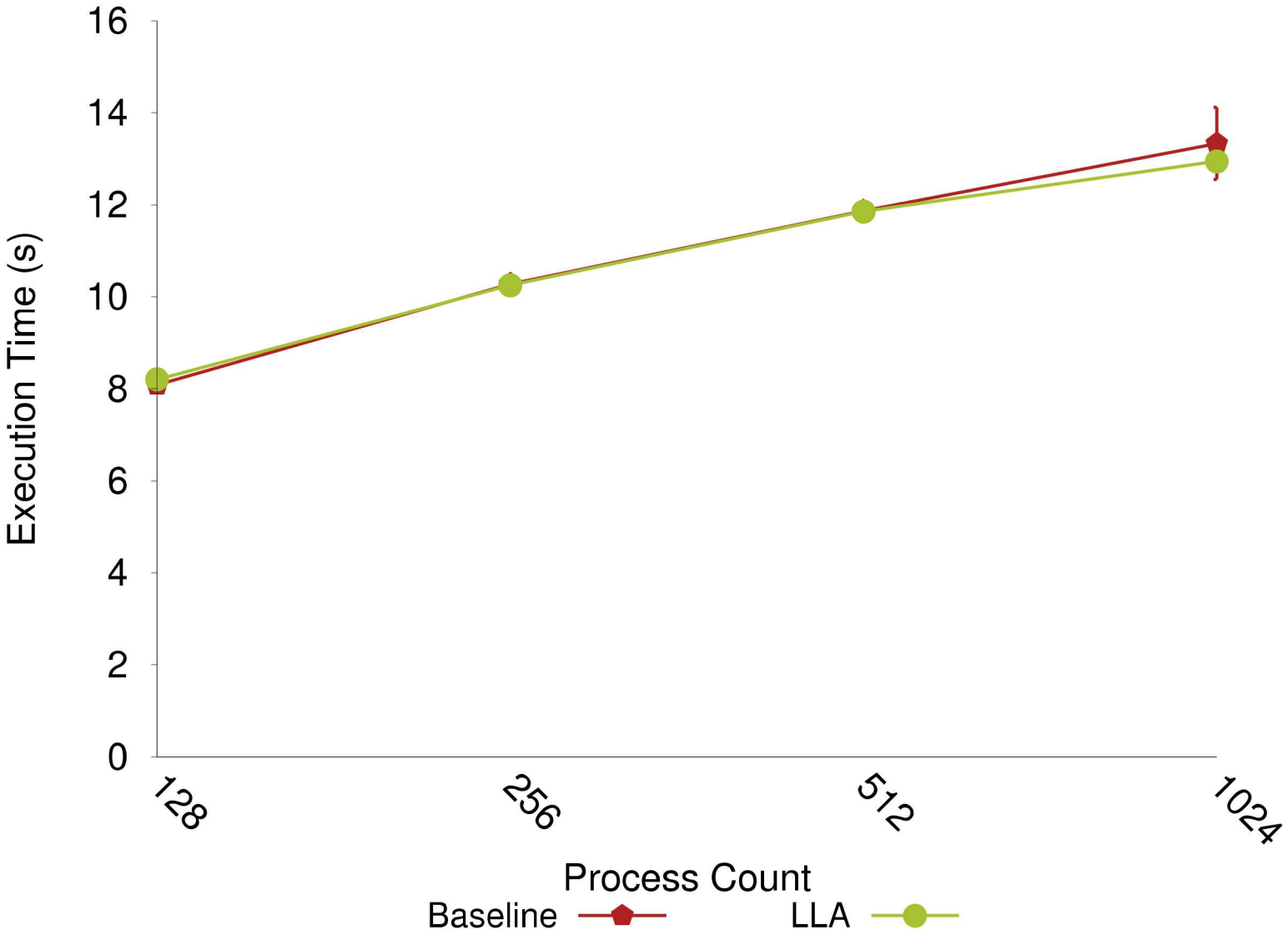
Microbenchmarks Conclusions

- Both architectures show an increase from increased spatial locality
  - Both show a significant impact at 2 elements per cache line
  - We observe diminishing returns from spatial locality between multiple cache-lines
- Sandy Bridge shows a significant speedup from temporal locality
- Broadwell does not, which could be due to a number of reasons
  - Hot-caching proof-of-concept shows an impact
  - Possible causes
    - Tuning mismatch
    - Network speed bottleneck
    - Architectural changes

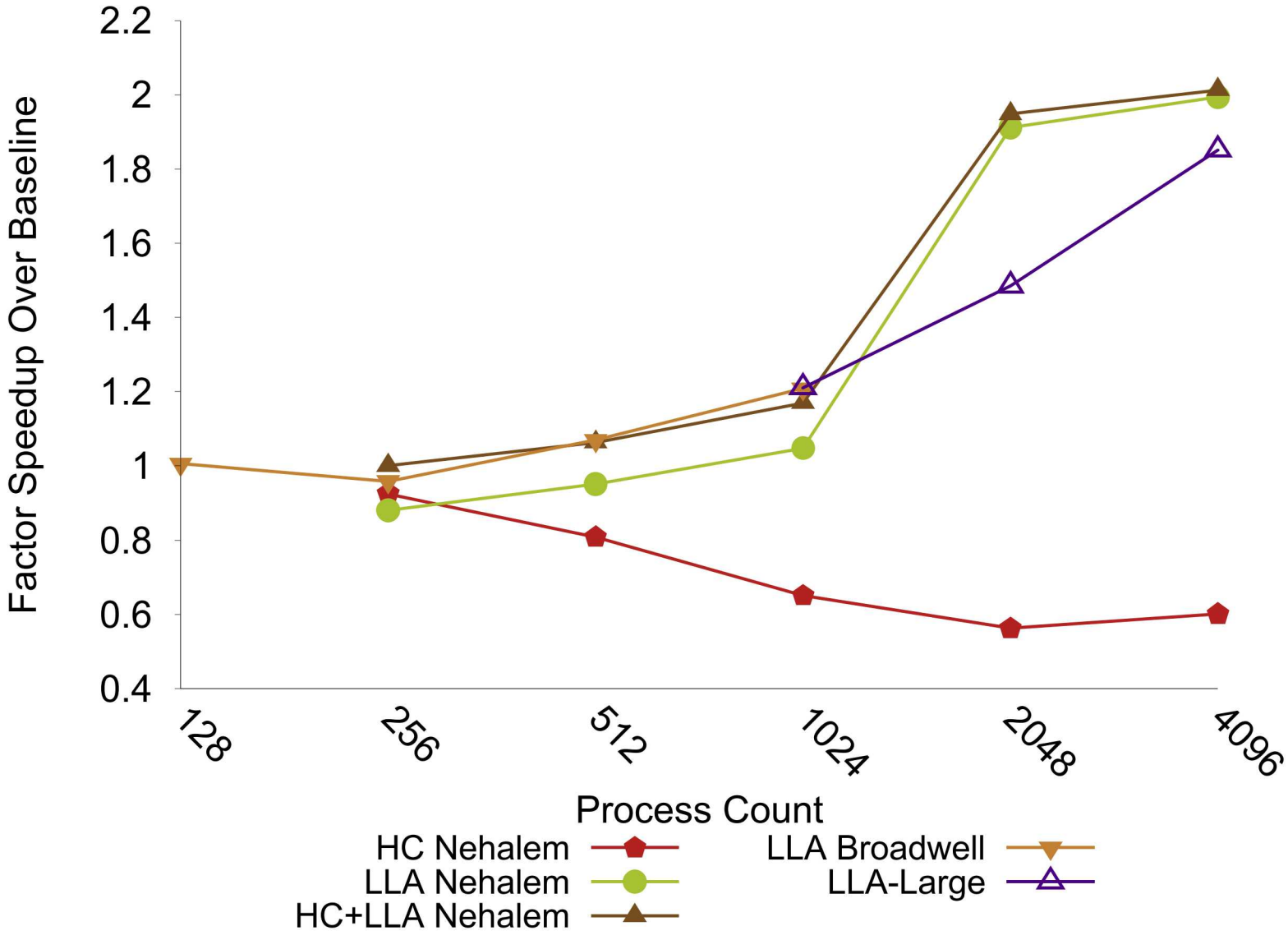
Application Experiments

- To evaluate the application impact of locality we ran multiple proxy and real applications
- AMG2013 is representative on many current applications
  - Adaptive multi grid proxy application
  - Leverages communication determinism to reduce search depths
- Fire Dynamics Simulator (FDS)
  - Full application
  - Currently the defacto message matching benchmark
  - Several communication inefficiencies but closest representation to multithreaded communication behavior from a matching standpoint
- Given the determinism, we expect minimal impact on current applications and significant impact on FDS

Application Impact – AMG



Application Impact – FDS



Pseudo-Permanent Cache Occupancy

- Locality significantly impacts matching performance
- There are a number of ways this can be accomplished
  - Hot caching
  - Network cache
  - Pining memory to cache

Future Work

- Expanding the study to include other parts of message processing
- Leveraging these techniques in other scalable system software
- Exploring the potential for cache pinning within an application
- Simulation of the impact of an on-chip network cache

Questions?