Evaluating Best and Worst Case Scenarios on Two-Level Memory Systems

Ryan J. Huber
University of Minnesota
huber395@umn.edu

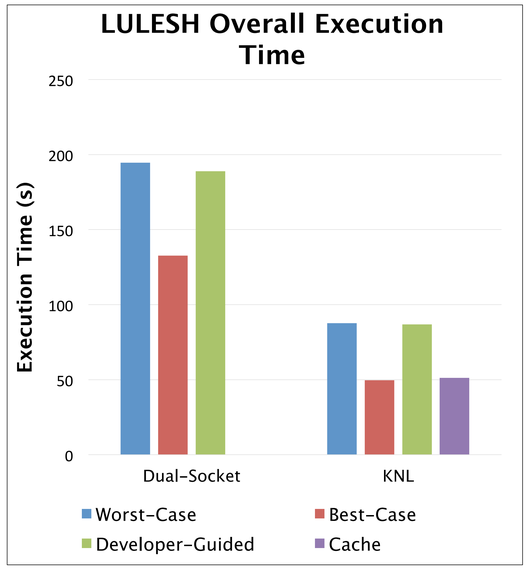
Edgar A. León
Lawrence Livermore National Laboratory
leon@llnl.gov

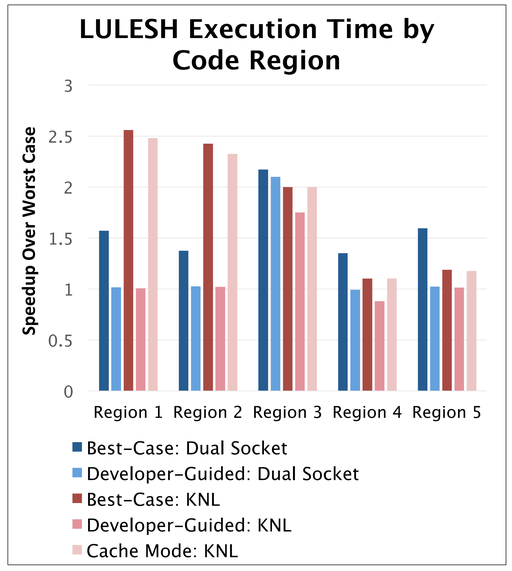
Abstract

To achieve the capacity and bandwidth requirements of an exascale memory system, vendors are employing multiple levels of memory: a small high-bandwidth memory close to the processor and a large but low-bandwidth memory. To leverage the high-bandwidth memory, application developers could explicitly place data structures in fast memory, but this becomes impractical for large HPC codes. Another approach is to develop metrics to semi-automatically identify candidate data structures to place in fast memory.

In this poster, we evaluate the performance of an LLNL proxy application under a variety of data allocation policies. Our results highlight the importance of properly utilizing fast memory by establishing empirical lower and upper bounds on the performance improvement of the application. We also raise the question whether a developer-guided approach may or may not be enough to significantly impact performance. Finally, we show that a traditional, one-level memory system can be used to gain insight into the performance of data placement strategies before porting an application to a two-level memory architecture.

Evaluating Data Placement Policies with a Hydrodynamics Mini-App





LULESH: An explicit shock hydrodynamics proxy application that contains data access patterns and computational characteristics of larger hydrodynamics codes at LLNL.

Policies:

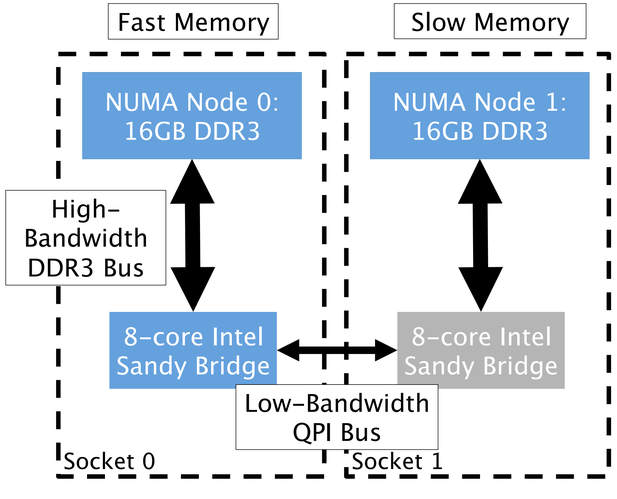
- **Best-case:** place data in fast memory
- **Worst-case:** place data in slow memory
- **Developer-guided:** place selected data in fast memory using developer guidance

Findings:

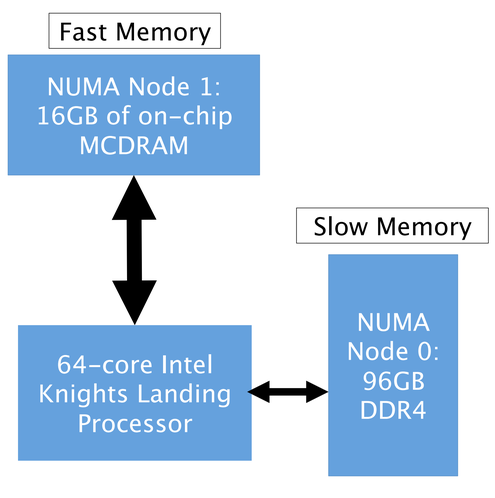
- Best-case policy improves performance by 32% for emulation and 43% for KNL.
- Greater improvement on KNL due to its wider gap in memory bandwidth.
- KNL in cache mode is almost as fast as the best-case policy—caused by LULESH's small working set.
- Effectiveness of developer-guided policy is region dependent.

Machine Environment and Memory Bandwidth Measurements

Emulation with Dual-Socket Hardware



Intel Knights Landing (KNL)



Bandwidth Measurements with the STREAM Benchmark

Dual-Socket Hardware

Test	Far Socket	Local Socket
Copy (GB/s)	10.75	28.02
Scale (GB/s)	10.74	28.13
Add (GB/s)	12.21	31.61
Triad (GB/s)	12.23	31.76
Normalized	**1X**	**2.60X**

Knights Landing (KNL)

Test	DDR4	MCDRAM
Copy (GB/s)	50.78	206.39
Scale (GB/s)	50.75	256.63
Add (GB/s)	57.04	279.09
Triad (GB/s)	57.09	295.17
Normalized	**1X**	**4.81X**

Conclusions

- Our best-case and worst-case policies illustrate the importance of using fast memory efficiently.
- More research is needed to evaluate the effectiveness of developer-guided allocation policies on performance.
- Single-level memory systems can provide useful insights to understand data placement tradeoffs before porting to a two-level memory machine.

Future work

- Evaluate CPU+GPU architectures.
- Extend analysis to other memory intensive codes including TeaLeaf and miniFE.
- Develop metrics to evaluate the potential of data objects for fast memory placement.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-698472.