Efficiency and Parallelism on the Lucata Pathfinder with the HPCG Benchmark[i]

Yongnuo Yang[1^], Jeremy Wang[2^], Alexander Contratti[2]
Faculty Advisor : Dr. Jeffrey Young[3]

[1] Student in School of Electrical and Computer Engineering [2] Student in School of Computer Science [3] Senior Research Scientist in School of Computer Science
Georgia Institute of Technology, Atlanta, Georgia 30332

Georgia Tech College of Computing
Center for Research into Novel Computing Hierarchies

Near-Memory Architecture and the Lucata Pathfinder

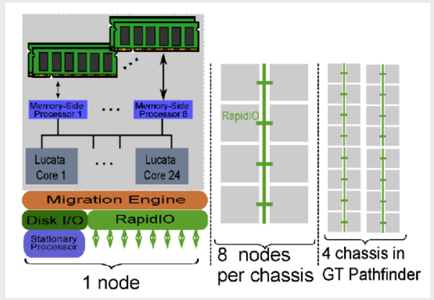
Near-memory computing (NMC) is a type of computer architecture that abandons a traditional cache hierarchy in favor of placing processing elements close to memory to address the performance bottleneck caused by the memory wall problem.

The Lucata Pathfinder is 32-node near-memory system that uses lightweight migratory threads in a PGAS (Partitioned Global Address Space). Each node contains:

- 24 Lucata Cores (LC, the main computing unit)
- 8 memory-side processors (MSP)
- 1 stationary core (SC, for system services)

Cores are connected by a Serial RapidIO network.

[1] Organization of GT Pathfinder System

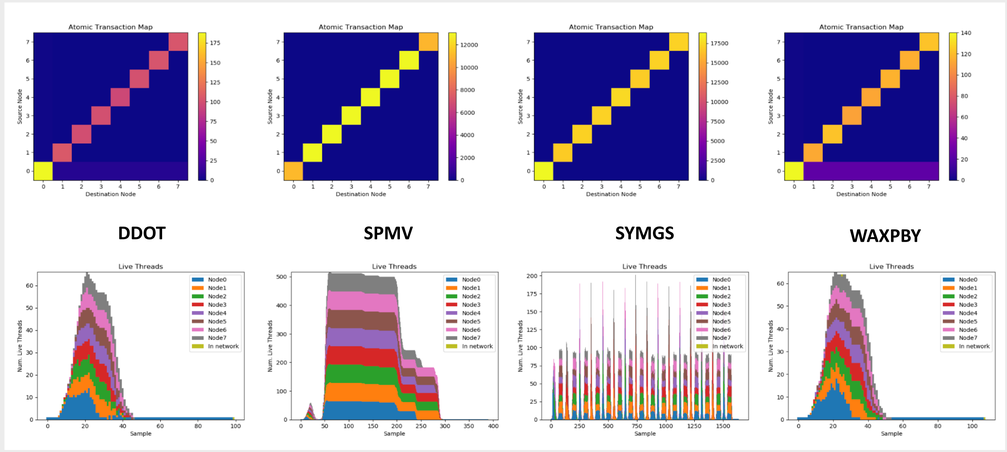


Pathfinder Workflow

- Programs are first tested in *emusim*, a simulation environment for the Pathfinder
- The simulation is then profiled for optimization
- The program is then run on the Pathfinder itself, scaling up in hardware resource usage:
  - Single-node execution
  - Single-chassis (8-node) execution
  - Four-chassis (32-node) execution
- Hardware performance counters gather profiling data for runs on the Pathfinder
- All profiling data is analyzed to ensure equal work distribution and assess hardware scalability and efficiency

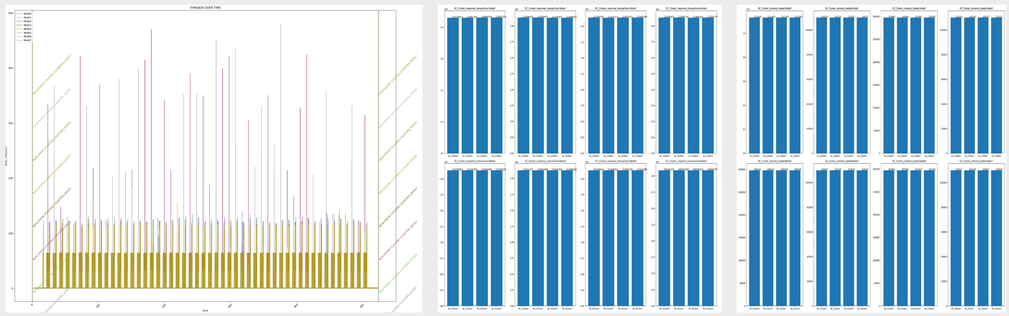
Profiling, HW Performance Counters, and Register Polling in HPCG Benchmark Execution

The Lucata Pathfinder runs an optimized HPCG derived from G. P. Krawezik et al[2], which employs a coloring algorithm to enhance data stripping and storage, optimizing data access in the NMC structure. The benchmark executables were compiled by the authors using the latest toolchain and underwent several adaptations for optimizations. Parameters were tuned to achieve evaluation results.



Simulation profiling results (size=16, nodes=8) are organized into four math operation kernels: **DDOT**(DotProduct), **SPMV**(Sparse Matrix Multiplication), **SYMGS**(Symmetric Gauss-Seidel), **WAXPBY**(Addition of Two Scaled Vectors)

Atomic Transaction Map indicates higher, favorable data locality within nodes.
Live Thread Map demonstrates healthy concurrency patterns within the simulation.



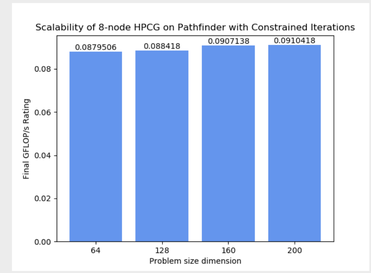
Hardware execution profilings are done on an 8-node configuration with a problem size of 128.

- **Live threads** generation data indicates the presence of parallel computation. The observed pattern closely resembles the SYMGS pattern above as a compute-intensive preconditioner. Spikes characterized by instantaneous high activity are potentially associated with SPMV operations.
- **Hardware performance counters** offer valuable insights into workload distribution, showcasing a well-balanced load across the various computing elements. Specifically, selected counters related to transaction responses and context loading demonstrate uniformity in workload allocation.

Discussions & Limitations

- Profiling data demonstrates an effective distribution of data and dynamic thread behavior.
- Performance counters indicate that workloads are evenly distributed among the LC clusters across different nodes of the system.
- Results from polling registers provide strong evidence of concurrent execution during the actual program run.
- To effectively stress both memory and parallelization in HPCG, a problem size of approx. 829 is required for 512 GB RAM; however, this is unacceptable due to long runtimes with problem sizes beyond 256.
- Software-based power sampling lacks granularity, making it challenging to assess the testbed's efficiency accurately.

Ongoing Efforts



- Running larger problem sizes within a reasonable runtime to support and finalize preliminary scalability results
- Refine methods for gathering and interpreting power sampling data for the Pathfinder
- Report final scalability and power results and compare a scaled-up Pathfinder with traditional x86 architectures

References & Acknowledgments

[1] Jeffrey Young, “gt-crnch-rg/lucata-pathfinder-tutorial: HPEC 2022 Tutorial”. Zenodo, 09/27/2022. doi: 10.5281/zenodo.7117251.
[2] G. P. Krawezik, S. K. Kuntz and P. M. Kogge, "Implementing Sparse Linear Algebra Kernels on the Lucata Pathfinder-A Computer," 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2020, pp. 1-6, doi: 10.1109/HPEC43674.2020.9286207.
i This work was supported by the Georgia Tech Vertically Integrated Project (VIP): Future Computing with Rogues Gallery. This research was supported in part through research infrastructure and services provided by the Rogues Gallery testbed hosted by the Center for Research into Novel Computing Hierarchies (CRNCH) at Georgia Tech. The Rogues Gallery testbed is primarily supported by the National Science Foundation (NSF) under NSF Award Number #2016701. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s), and do not necessarily reflect those of the NSF.
^ These authors contributed equally to this work.