Optimizing the Barnes-Hut Algorithm for Multicore Clusters
Junhao Zhang, Babak Behzad, Marc Snir
University of Illinois at Urbana-Champaign, {jczhang, bbehzad2, snir}@illinois.edu
SC’13, November 17-22, 2013, Denver, Colorado, USA

Motivation
• Important trends of supercomputer evolution:
  • Number of cores per node keeps increasing;
  • Amount of memory per core is decreasing;
  • One-sided communication (RDMA) is increasingly well supported on the interconnection networks.
• It is important to explore new programming models to cope with or leverage these trends.
• Through the Barnes-Hut (BH) algorithm, we study a promising computing paradigm, i.e., partitioned global address space (PGAS) + X, which:
  • Uses a global address space for internode communication;
  • Hides latency through intranode multithreading;
  • Achieves data sharing and message reduction through multithreading.
• We show many advantages gained from this design.

What is Barnes-Hut?
• An O(N log N) algorithm for the n-body problem, which needs to compute forces on every body (particle).
• Use an octree as the main data structure.
• Has massive parallelism:
  • For all bodies, can compute their forces in parallel.
  • For one body, can interact with different tree nodes (aka cells) in parallel, provided the results are summed together finally.
• Irregular, whether to traverse children of a cell (i.e., open a cell) is computation dependent.

Hybrid BH Design
• Implemented in PPL, a C++ template library having a PGAS memory model.
• Octree is distributed among compute nodes and cells are linked by global pointers provided by PPL.
• Each cell has a flag localized to indicate whether its children have been cached locally.

Experimental Setup
• Tested on a Linux cluster with each node having two hex-core Xeon® X5650 CPU@2.66GHz, with up to 64 nodes.
• Generated input bodies using the Planner model.
• Compared the current code (named PPL BH) with UPC BH [1] (a BH code without multithreading), SPLASH-2 BH [5] (on single node) and other BH codes.

Results
• Less memory by sharing the cached octree through multithreading in stead of having each core build its own cached octree as in UPC BH.
• With n=1M, 0.05 on 64 nodes, saved 59% of memory.
• Less communication, since data sharing also helps to filter duplicated off-node data requests.
• With the same setting, saved 57% internode message.
• Better load balancing by applying OpenMP guided-like scheduling, to adapt to runtime fluctuations.
• Execution time variation of the 12 threads is < 2.5%, which was up to 71% in UPC BH.

Conclusions and Future Work
• We optimized the Barnes-Hut algorithm in a PGAS + X programming style, which brings many benefits as shown and thus is well suited to multicore clusters.
• We integrated multithreading and one-sided communication, with multitasking as the primary mechanism for latency hiding.
• In future, we like to abstract common runtime services from PPL BH and try to benefit other irregular applications.

References

Acknowledgments
This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357, and by the U.S. Department of Energy Sandia National Lab grant 1205852.