Partitioning and Task Placement

FASTMath Team Members: K. Devine, G. Diamond, D. Ibanez, V. Leung, A. Prokopenko, S. Rajamanickam, M. Shephard, C. Smith

By balancing complex constraints in assigning data to tasks and accounting for the underlying architecture in assigning tasks to cores, we reduce applications' computation and communication costs for greater parallel performance in leadership-class computers.

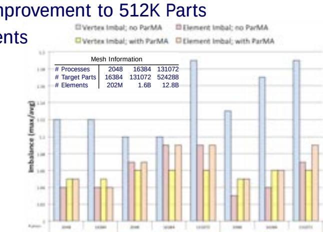
ParMA: Partitioning Using Mesh Adjacencies

Mesh adjacencies provide a more complete problem representation than a standard partitioning graph (e.g., dual graph)

- A complete mesh adjacency structure supports queries in constant time (see the Unstructured Mesh poster for details)
- Partitions can be improved with respect to additional metrics

Example: ParMA Partition Improvement to 512K Parts

- *Goal*: Balance both elements and vertices (including copies at boundaries)
- *Approach*: Given a partition with elements balanced, ParMA's diffusive procedure balances vertices, then re-balances elements



For PHASTA CFD code with 1.44B element mesh and 3.1M parts

- ParMA reduces original 35% vertex imbalance to 27%

ParMA Ghost-Element Balancing in MPAS-Ocean

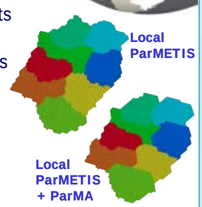
Per-process compute time depends on number of owned elements PLUS number of ghost elements (three layers)

Cost function:

- Needs mesh topology to account for ghosts
- Changes as part boundaries change
- Not easily handled by standard partitioners

ParMA approach: Leverage diffusive-improvement element selection with ghost-based diffusion-targets computation

- *For 3.6M-element mesh, achieved 5% imbalance in 1 second on BG/Q*



Demonstrating the benefit of using ParMA with local ParMETIS in MPAS-Ocean; for little additional cost, balance of owned+ghost elements is achieved.

# Init Parts	Target # Parts	Imbalance: ParMETIS only	Imbalance: ParMETIS+ParMA	# ParMA Iters	ParMETIS time (s)	ParMA time (s)
64	512	7%	5%	6	18.8	0.6
512	1024	12%	5%	11	1.2	0.7
512	2048	17%	5%	14	1.3	0.8
512	4096	21%	5%	21	1.6	1.0

Architecture-Aware Geometric Task Placement

Given an allocation of nodes in a parallel system, accounting for the underlying architecture when assigning MPI tasks to cores can reduce network congestion and communication costs

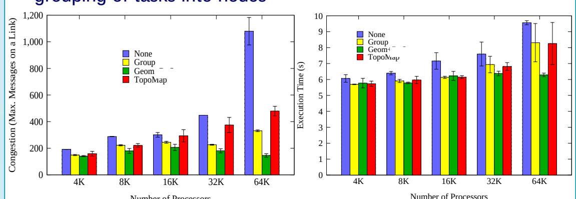
- Especially important for large-scale simulations on systems with non-contiguous node allocations (e.g., Hopper, Cielo)

Approach: Exploit geometric partitioners (e.g., MultiJagged in Zoltan2) to assign interdependent tasks to "nearby" cores

- Use geometric locality as a proxy for task dependence and network connectivity

MiniGhost stencil-based mini-app experiments

- Geometric mapping reduces congestion, communication time, and total execution time relative to default placement (None), LipTopoMap graph-based placement (Toefler, Snir), and 2x2x4 grouping of tasks into nodes



Geometric Mapping reduces Congestion (left) and Execution Time (right)

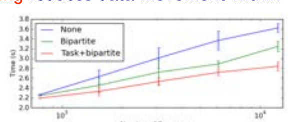
Task Placement in Multigrid Solver MueLu

MueLu multigrid solver (in Trilinos) uses Zoltan2's geometric task placement along with bipartite graph matching to reduce data movement between and within multigrid levels

- Bipartite matching reduces data movement between fine operator (on all cores) and coarse operator (on a subset of cores)
- Zoltan2's geometric task mapping reduces data movement within fine operator

Weak scaling experiments with MueLu on NERSC Hopper

Time for one multigrid solve



Ongoing Research Efforts

- New multi-threaded, multi-constraint and multi-objective graph partitioner using label-propagation schemes
  - Compute partitions more quickly and use memory more efficiently than traditional multilevel partitioners
  - Preliminary results: 8-30x less memory and up to 14x faster than state-of-the-art partitioners
- ParMA predictive load balancing for general mesh adaptation using light-part merging followed by heavy-part splitting
- ParMA testing and optimization on Intel Xeon Phi
- MPI+OpenMP MultiJagged geometric partitioning that provides greater scalability than Recursive Coordinate Bisection
- Assessment of Trilinos' Kokkos manycore performance and portability package for use in partitioning algorithms

For More Information

Software Downloads:

- ParMA: https://github.com/SCOREC/core
- Zoltan and Zoltan2 (in Trilinos): http://trilinos.org

Related publications:

- ParMA: Seol et al., 2012 SC Companion; Smith et al., SISC (sub)
- Task Mapping: Deveci et al., IPDPS14; Leung et al., PPoPP14
- MultiJagged: Deveci et al., SAND2012-10318C, TPDS (sub)
- Label-propagation partitioning: Slota et al., BigData2014 (sub)

Collaborators

K. Jansen, B. Matthews, M. Rasquin (U. Colorado-Boulder)
D. Bunde (Knox College)
M. Deveci, U. Çatalyürek (Ohio State University)
G. Slota, K. Madduri (Penn State University)
E. Boman, J. Brandt, A. Gentile, S. Olivier, K. Pedretti (Sandia)

More Information: http://www.fastmath-scidac.org or contact Lori Diachin, LLNL, diachin2@llnl.gov, 925-422-7130

SciDAC Scientific Discovery through Advanced Computing