Partitioning and Dynamic Load Balancing for Petascale Applications

Karen Devine, Sandia National Laboratories
Erik Boman, Sandia National Laboratories
Umit Çatalyürek, Ohio State University
Lee Ann Riesen, Sandia National Laboratories
Partitioning and Load Balancing

- Assignment of application data to processors for parallel computation.
- Applied to grid points, elements, matrix rows, particles, ....
Static Partitioning

- Static partitioning in an application:
  - Data partition is computed.
  - Data are distributed according to partition map.
  - Application computes.

- Ideal partition:
  - Processor idle time is minimized.
  - Inter-processor communication costs are kept low.
Dynamic Repartitioning (a.k.a. Dynamic Load Balancing)

- Dynamic repartitioning (load balancing) in an application:
  - Data partition is computed.
  - Data are distributed according to partition map.
  - Application computes and, perhaps, adapts.
  - Process repeats until the application is done.

- Ideal partition:
  - Processor idle time is minimized.
  - Inter-processor communication costs are kept low.
  - Cost to redistribute data is also kept low.
What makes a partition “good,” especially at petascale?

• Balanced work loads.
  – Even small imbalances result in many wasted processors!
    • 50,000 processors with one processor 5% over average workload is equivalent to
      2380 idle processors and 47,620 perfectly balanced processors.

• Low interprocessor communication costs.
  – Processor speeds increasing faster than network speeds.
  – Partitions with minimal communication costs are critical.

• Scalable partitioning time and memory use.
  – Scalability is especially important for dynamic partitioning.

• Low data redistribution costs (for dynamic partitioning).
  – Redistribution costs must be recouped through reduced total execution time.
Partitioning Algorithms in the Zoltan Toolkit

Geometric (coordinate-based) methods
- Recursive Coordinate Bisection (Berger, Bokhari)
- Recursive Inertial Bisection (Taylor, Nour-Omid)
- Space Filling Curve Partitioning (Warren&Salmon, et al.)
- Refinement-tree Partitioning (Mitchell)

Hypergraph and graph (connectivity-based) methods
- Hypergraph Partitioning
- Hypergraph Repartitioning
- PaToH (Catalyurek & Aykanat)
- Zoltan Graph Partitioning
- ParMETIS (U. Minnesota)
- Jostle (U. Greenwich)

• Idea:
  – Divide work into two equal parts using a cutting plane orthogonal to a coordinate axis.
  – Recursively cut the resulting subdomains.
Geometric Repartitioning

- Implicitly achieves low data redistribution costs.
- For small changes in data, cuts move only slightly, resulting in little data redistribution.
Applications of Geometric Methods

- Adaptive Mesh Refinement
- Parallel Volume Rendering
- Crash Simulations and Contact Detection
- Particle Simulations
RCB Advantages and Disadvantages

• Advantages:
  – Conceptually simple; fast and inexpensive.
  – All processors can inexpensively know entire partition (e.g., for global search in contact detection).
  – No connectivity info needed (e.g., particle methods).
  – Good on specialized geometries.

• Disadvantages:
  – No explicit control of communication costs.
  – Mediocre partition quality.
  – Can generate disconnected subdomains for complex geometries.
  – Need coordinate information.

SLAC’s 55-cell Linear Accelerator with couplers: One-dimensional RCB partition reduced runtime up to 68% on 512 processor IBM SP3. (Wolf, Ko)
Graph Partitioning

• Represent problem as a weighted graph.
  – Vertices = objects to be partitioned.
  – Edges = dependencies between two objects.
  – Weights = work load or amount of dependency.
• Partition graph so that …
  – Parts have equal vertex weight.
  – Weight of edges cut by part boundaries is small.

• Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Simon, Hendrickson, Leland, Kumar, Karypis, et al.
Graph Repartitioning

• Diffusive strategies (Cybenko, Hu, Blake, Walshaw, Schloegel, et al.)
  – Shift work from highly loaded processors to less loaded neighbors.
  – Local communication keeps data redistribution costs low.

• Multilevel partitioners that account for data redistribution costs in refining partitions (Schloegel, Karypis)
  – Parameter weights application communication vs. redistribution communication.
Applications using Graph Partitioning

Finite Element Analysis

Multiphysics and multiphase simulations

Linear solvers & preconditioners
(square, structurally symmetric systems)
Graph Partitioning: Advantages and Disadvantages

- **Advantages:**
  - Highly successful model for mesh-based PDE problems.
  - Explicit control of communication volume gives higher partition quality than geometric methods.
  - Excellent software available.
    - **Serial:**
      - Chaco (SNL)
      - Jostle (U. Greenwich)
      - METIS (U. Minn.)
      - Party (U. Paderborn)
      - Scotch (U. Bordeaux)
    - **Parallel:**
      - Zoltan (SNL)
      - ParMETIS (U. Minn.)
      - PJostle (U. Greenwich)

- **Disadvantages:**
  - More expensive than geometric methods.
  - Edge-cut model only approximates communication volume.
Hypergraph Partitioning


- Hypergraph model:
  - Vertices = objects to be partitioned.
  - Hyperedges = dependencies between two or more objects.

- Partitioning goal: Assign equal vertex weight while minimizing hyperedge cut weight.
Hypergraph Repartitioning

• Augment hypergraph with data redistribution costs.
  – Account for data’s current processor assignments.
  – Weight dependencies by their size and frequency of use.
• Hypergraph partitioning then attempts to minimize total communication volume:
  Data redistribution volume
  + Application communication volume
  Total communication volume

• Lower total communication volume than geometric and graph repartitioning.

Best Algorithms Paper Award at IPDPS07
“Hypergraph-based Dynamic Load Balancing for Adaptive Scientific Computations”
Catalyurek, Boman, Devine, Bozdag, Heaphy, & Riesen
Hypergraph Applications

Finite Element Analysis

Circuit Simulations

Linear programming for sensor placement

Multiphysics and multiphase simulations

Linear solvers & preconditioners (no restrictions on matrix structure)

Data Mining

Linear programming for sensor placement

Linear solvers & preconditioners (no restrictions on matrix structure)
Hypergraph Partitioning: Advantages and Disadvantages

• Advantages:
  – Communication volume reduced 30-38% on average over graph partitioning (Catalyurek & Aykanat).
    • 5-15% reduction for mesh-based applications.
  – More accurate communication model than graph partitioning.
    • Better representation of highly connected and/or non-homogeneous systems.
  – Greater applicability than graph model.
    • Can represent rectangular systems and non-symmetric dependencies.

• Disadvantages:
  – More expensive than graph partitioning.
Performance Results

• Experiments on Sandia’s Thunderbird cluster.
  – Dual 3.6 GHz Intel EM64T processors with 6 GB RAM.
  – Infiniband network.

• Compare RCB, graph and hypergraph methods.

• Measure …
  – Amount of communication induced by the partition.
  – Partitioning time.
Test Data

SLAC *LCLS
Radio Frequency Gun
6.0M x 6.0M
23.4M nonzeros

Xyce 680K ASIC Stripped Circuit Simulation
680K x 680K
2.3M nonzeros

Cage15 DNA Electrophoresis
5.1M x 5.1M
99M nonzeros

SLAC Linear Accelerator
2.9M x 2.9M
11.4M nonzeros
Communication Volume:
Lower is Better

- **SLAC 6.0M LCLS**
  - Communication Volume
  - Number of parts = number of processors.
  - Graphs showing communication volume vs. number of processors for various numbers of parts.

- **SLAC 2.9M Linear Accelerator**
  - Similar graphs as SLAC 6.0M LCLS.

- **Xyce 680K circuit**
  - Graphs showing communication volume vs. number of processors for various numbers of parts.

- **Cage15 5.1M electrophoresis**
  - Graphs showing communication volume vs. number of processors for various numbers of parts.

- **RCB**, **Graph**, **Hypergraph**
  - Color-coded graphs indicating different types of parts.
Partitioning Time: Lower is better

SLAC 6.0M LCLS

1024 parts. Varying number of processors.

SLAC 2.9M Linear Accelerator

Xyce 680K circuit

Cage15 5.1M electrophoresis

SLAC 6.0M LCLS

Partitioning time (secs)

Number of Processors

SLAC 2.9M Linear Accelerator

Partitioning time (secs)

Number of Processors

Cage15 5.1M electrophoresis

Partitioning time (secs)

Number of Processors
Repartitioning Experiments

- Experiments with 64 parts on 64 processors.
- Dynamically adjust weights in data to simulate, say, adaptive mesh refinement.
- Repartition.
- Measure repartitioning time and total communication volume:
  
  Data redistribution volume
  + Application communication volume
  
  Total communication volume
Repartitioning Results: Lower is Better

SLAC 6.0M LCLS

Xyce 680K circuit

Repartitioning Time (secs)
Aiming for Petascale

• Reducing communication costs for applications.
  – Reducing communication volume.
    • Two-dimensional sparse matrix partitioning (Catalyurek, Aykanat, Bisseling).
    • Partitioning non-zeros of matrix rather than rows/columns.
  – Reducing message latency.
    • Minimize maximum number of neighboring parts.
    • Balancing both computation and communication (Pinar & Hendrickson); balance criterion is complex function of the partition instead of simple sum of object weights.
  – Reducing communication overhead.
    • Map parts onto processors to take advantage of network topology.
    • Minimize distance messages travel in network.
Aiming for Petascale

- Hierarchical partitioning in Zoltan v3.
  - Partition for multicore/manycore architectures.
    - Partition hierarchically with respect to chips and then cores.
    - Similar to strategies for clusters of SMPs (Teresco, Faik).
    - Treat core-level partitions as separate threads or MPI processes.
  - Support 100Ks processors.
    - Reduce collective communication operations during partitioning.
    - Allow more localized partitioning on subsets of processors.
Aiming for Petascale

• Improving scalability of partitioning algorithms.
  – Hybrid partitioners (particularly for mesh-based apps.):
    • Use inexpensive geometric methods for initial partitioning; refine with hypergraph/graph-based algorithms at boundaries.
    • Use geometric information for fast coarsening in multilevel hypergraph/graph-based partitioners.

  – Refactored partitioners for bigger data sets and processor arrays.
Aiming for Petascale

• Testing and performance evaluation.
  – Examine effectiveness of partitions in applications.

• Wanted: Collaborations with application developers!
For More Information…

  – Download Zoltan v3 (open-source software).
  – Read User’s Guide.

• ITAPS Interface to Zoltan:
  https://svn.scorec.rpi.edu/wsvn/TSTT/Distributions/

• SciDAC Tutorial, June 29, 1:30-4:30pm, MIT
  – Load-balancing and partitioning for scientific computing using Zoltan
  – Erik Boman, Karen Devine
Thanks

SciDAC CSCAPES Institute (A. Pothen, Old Dominion U., PI)
SciDAC ITAPS Center (L. Diachin, LLNL, PI)
NNSA ASC Program

• S. Attaway (SNL)
• C. Aykanat (Bilkent U.)
• A. Bauer (RPI)
• R. Bisseling (Utrecht U.)
• D. Bozdag (Ohio St. U.)
• T. Davis (U. Florida)
• J. Faik (RPI)
• J. Flaherty (RPI)
• R. Heaphy (SNL)
• B. Hendrickson (SNL)
• M. Heroux (SNL)
• D. Keyes (Columbia)
• K. Ko (SLAC)

• G. Kumfert (LLNL)
• L.-Q. Lee (SLAC)
• V. Leung (SNL)
• G. Lonsdale (NEC)
• X. Luo (RPI)
• L. Musson (SNL)
• S. Plimpton (SNL)
• J. Shadid (SNL)
• M. Shephard (RPI)
• C. Silvio (SNL)
• J. Teresco (Mount Holyoke)
• C. Vaughan (SNL)
• M. Wolf (U. Illinois)

Communication Volume:  
Lower is Better

**SLAC 6.0M LCLS**

- 1024 parts.  
- Varying number of processors.

**SLAC 2.9M Linear Accelerator**

**Xyce 680K circuit**

**Cage15 5.1M electrophoresis**

- RCB  
- Graph  
- Hypergraph

Graphs showing communication volume for different systems with varying numbers of processors.
Partitioning Time: Lower is better

**SLAC 6.0M LCLS**

**SLAC 2.9M Linear Accelerator**

**Xyce 680K circuit**

**Cage15 5.1M electrophoresis**

Number of parts = number of processors.