Enabling Petascale Science through Combinatorial Algorithms

Alex Pothen
CSCAPES Institute
www.cscapes.org

Erik Boman, Karen Devine, Bruce Hendrickson; Paul Hovland, Boyana Norris, Jean Utke; Assefaw Gebremedhin, Florin Dobrian; Umit Catalyurek; Michelle Mills Strout
Parallelization, Load Balancing

Graph Coloring

Performance

Automatic Differentiation

Graph Matching

Combinatorial problems?
Load Balancing

...enabling parallelization and fast run-times for irregular applications
Partitioning and Load Balancing

- **Goal**: assign data (and tasks) to processors to
  - minimize application runtime
  - maximize utilization of computing resources
- **Metrics**:
  - minimize processor idle time (balance workloads)
  - keep inter-processor communication costs low
- **Impacts performance of a wide range of simulations**
  - Accelerator code speeded up 3X with a geometric partitioner
- **Several partitioning and load balancing algorithms**
  - Contact detection
  - Particle simulations
  - Linear solvers & preconditioners
  - Adaptive mesh refinement
Dynamic Load Balancing

• Applications where workload or locality changes during simulation
  – Adaptive mesh refinement
  – Particle methods
• Repartitioning has additional cost: Moving data from old to new decomposition
• IPDPS 2007 Best Paper Award (Boman, Bozdag, Catalyurek, Devine, …)

• Talk: Wed 9:30 A.M. Tutorial: Fri 1:30 P.M.
Repartitioning Model

• Dynamic (adaptive) applications need to load-balance periodically since data and dependencies change.

• Problem: Repartition to accurately trade-off data migration cost against future savings from a data decomposition with lower communication.

• We developed algorithms for repartitioning model:
  \[ \text{executionT} = \#\text{iter} \times (\text{computationT} + \text{communicationT}) + \text{repartT} + \text{migrationT} \]

• Implementation in Zoltan based on hypergraph partitioning.

• Communication volume reduced by 20-30% vs. earlier methods.

• Best paper award at IPDPS’07.
Zoltan Toolkit: Data Services for Dynamic Applications

Dynamic Load Balancing

Graph Coloring

Data Migration

Matrix Ordering

Unstructured Communication

Distributed Data Directories

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Zoltan 3.0

Zoltan 3.0 is now available (www.cs.sandia.gov/Zoltan). New features use hypergraphs for modeling communications accurately:

• Hypergraph repartitioning
  – Reduces total communication in dynamic applications.

• Hypergraph refinement
  – Quickly improves an existing parallel distribution (partitioning)

• Hypergraph partitioning with fixed vertices
  – Allows application to fix certain data to specific processors.

• Hierarchical partitioning
  – 2-level partitioning, possibly using different algorithms, cost metrics
  – Useful for complex computer architectures (e.g., multi-core)
Automatic Differentiation

...enabling the solution of nonlinear differential equations, optimization, sensitivity analysis, uncertainty quantification, etc.
AD: Introduction

• Transforms code for computing a function into code for differentiating it
• Function computed from intrinsic operations, and modeled by a directed acyclic graph (DAG)
• Compute derivatives by composing partial derivatives for each operation, using the chain rule on the DAG
• Efficiency of generated code depends on sophistication of compiler analysis and combinatorial algorithms
AD: Combinatorial Problems

- Paul Hovland, Poster Tues 7:30 P.M.
- Parallel algorithms for differentiating reduction operations
- Reduce operations and storage needed to compute the derivatives by evaluating the DAG in suitable orders
  - Two extreme modes: Forward and Reverse
  - Modeled as vertex and edge elimination in DAG
  - Stop at some intermediate stage to find minimum storage
- Location of checkpoints in reverse mode
- Graph coloring for computing many derivatives in one AD pass through the DAG
- Integration with PETSc and Zoltan toolkits
Sensitivity analysis in climate model

• Sensitivity of flow through Drake Passage to ocean bottom topography (P. Heimbach, MIT)
  – Finite difference approximations: 23 days
  – Naïve automatic differentiation: 2 hours 23 minutes
  – Smart automatic differentiation: 22 minutes
Motivation for Reduction Derivatives

• Parallel applications use reduction operations such as sum, product, max, and min.

• Differentiating sum is trivial; max/min is complicated when the value is on more than 1 proc. (pt of nondifferentiability)

• Differentiating product can be accomplished via pair of parallel prefix operations:

\[ P_k = \prod_{i=1}^{k} x_i, \quad S_k = \prod_{i=k}^{nprocs} x_i \quad \frac{\partial f}{\partial x_k} = P_{k-1}S_{k+1} \]

• New algorithm requires 2log₂P communication phases (half as much as old).
Product Derivative on a Binary (Binomial) Tree

Leaves:
- Pass value to parent
- Set current value to 1
- Combine value from parent with own value

Non-leaves:
- Combine values from left and right children and pass to parent
- Pass value from left child to right child
- Pass value from right child to left child
- Pass value from parent to left and right children
AD: Current Capabilities

- **Fortran 77: ADIFOR 2.0/3.0**
  - Robust, mature tool with excellent language coverage
  - Excellent compiler analysis
  - Efficient forward mode; adequate reverse mode

- **C/C++: ADIC 2.0**
  - Semi-mature tool with full C language coverage
  - Sophisticated differentiation algorithms
  - Efficient forward mode

- **Fortran 90: OpenAD/F**
  - New tool with partial language coverage
  - Sophisticated differentiation algorithms
  - Accurate and novel compiler analysis
  - Innovative templating mechanism
  - Efficient forward and reverse modes
Graph Coloring

...reducing work in Automatic Differentiation; and discovering parallelism in computations
Coloring and Jacobian Computation

**Original Jacobian**

**Compressed** representation
*(Structurally orthogonal columns packed together)*

---

**D1 coloring**
formulation on column inter. graph

**D2 coloring**
bipartite graph
Coloring and H’’essian Computation

• **Symmetrically orthogonal** partition and its representation as a *star coloring*.

• Original Hessian entries *directly* recovered from compressed representation.

• **Substitutable partition** and its representation as an *acyclic coloring*.

• Original Hessian entries *indirectly* recovered from compressed representation, by separately “solving” *two-colored* trees.
Coloring and Derivatives: The Big Picture

• **Scenarios and coloring models:**
  – unsymmetric vs symmetric matrix
  – direct vs substitution method
  – uni- vs bi-directional partitioning

• **Developed novel sequential algorithms**

• **Future plans**
  – Develop parallel versions
  – integrate with AD tools

<table>
<thead>
<tr>
<th></th>
<th>1d partition</th>
<th>2d partition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jacobian</strong></td>
<td>Distance-2 coloring</td>
<td>Star bicoloring</td>
</tr>
<tr>
<td><strong>Hessian</strong></td>
<td>Star coloring</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Jacobian</strong></td>
<td>NA</td>
<td>Acyclic bicoloring</td>
</tr>
<tr>
<td><strong>Hessian</strong></td>
<td>Acyclic coloring</td>
<td>NA</td>
</tr>
</tbody>
</table>

![D2 coloring](image1)

![Star coloring](image2)

![Acyclic coloring](image3)
Coloring and parallel computation

- Coloring useful in scheduling dependent tasks. Examples in CSE:
  - Iterative solvers
  - Preconditioning
  - Scheduling iterations for cache reuse

- Computational graph distributed across processors → Coloring needs to be computed in parallel

- Greedy coloring heuristics effective in practice, but hard to parallelize

- Vertices ∼ computational subtasks
- Edges ∼ dependencies
- Model: Distance-1 coloring (same-color vertices ∼ concurrently executed tasks)
- Number of colors ∼ computational steps
Framework for parallel coloring

- **Essential ingredients of framework:**
  - Partition graph on processors, and speculate color subgraphs in rounds
  - Exchange color info after a superstep (coloring a specified no. of vertices)
  - Detect conflicts after each round, resolve using randomization, recolor when needed

- **Applied to D-1 and D-2 coloring, implemented in MPI; available in Zoltan**

- **Extending the framework to**
  - Tera- and peta-scale machines
  - Other graph problems

Weak scalability on two families of graphs: random (unstructured); planar (structured).
Matchings in Graphs

...enabling load balancing and linear solvers
Matchings in Graphs

- **Pair vertices joined by an edge**
  - Each vertex paired exclusively with one other, or none

- **Maximum matchings**
  - Cardinality: Number of matched edges
  - Edge weighted: sum of weights of matched edges
  - Vertex weighted
Matchings in Graphs

- **Matching is a pairing of vertices; a vertex is paired with one neighboring vertex or none**

- **Applications**
  - Place large elements on diagonals of matrices for solvers
  - Block triangular form to reduce work in solvers, improve condition number
  - Coarsening step in multilevel graph and hypergraph partitioners
Block Upper Triangular Form (BTF)

Circuit model from Xyce (Hoekstra, Day; Sandia) 683K rows, 2M nnz, 584K diag blocks
Solved 200 times faster! 100M problem waits.
Challenges for Petascale Computing

- Need new parallel CSC algorithms to be designed. Go boldly where no algorithms have gone before!
- Important to run faster than applications they are used in, but scalability of CSC algorithms is a misplaced concern.
- Multiple cores, complex network topologies, deeper memory hierarchies make CSC issues even more critical for performance.
Outreach and Training

• Organized the SIAM Workshop on CSC in Feb. 2007. 100 attendees, 12 early career researchers supported. SIAM News article in May 2007. URL: [www.cscapes.org](http://www.cscapes.org), click on CSC07

• International collaborations with CERFACS, AD groups in Germany, CSC groups in Norway and other countries.

• 3 Postdoctoral researchers, 4 PhD students, and an undergraduate are involved in CSCAPES research, and are co-mentored by Lab scientists.

• Working with several enabling technology and applications groups to integrate CSC software and solve their combinatorial problems.

• We welcome application kernels where CSC issues are significant; tell us about your combinatorial problems!
Mapping a Binary Tree to a Binomial Tree

Diagram:

- Binary Tree:
  - Root: 0
  - Left child: 4
  - Left child of 4: 0
  - Left child of 0: 2
  - Left child of 2: 1
  - Left child of 1: 0
  - Left child of 0: 3
  - Left child of 3: 2
  - Left child of 2: 4
  - Left child of 4: 5
  - Left child of 5: 6
  - Left child of 6: 7

- Binomial Tree:
  - Root: 0
  - Left child: 4
  - Left child of 4: 1
  - Left child of 1: 3
  - Left child of 3: 5
  - Left child of 5: 6
  - Left child of 6: 7
Parallel Prefix on a Binary (Binomial) Tree

Leaves:
• Pass initial value to parent
• Combine value from parent with own value

Non-leaves:
• Combine values from left and right children and pass to parent
• Pass value from left child to right child
• Pass value from parent to left and right children
Combinatorial Scientific Computing and Petascale Simulations

- A SciDAC Institute Funded by DOE’s Office of Science

Alex Pothen, Florin Dobrian, Assefaw Gebremedhin

Erik Boman, Karen Devine, Bruce Hendrickson

Umit Catalyurek

Paul Hovland, Boyana Norris, Jean Utke

Michelle Strout

www.cscapes.org
Enabling Petascale Computational Science through Combinatorial Algorithms

Alex Pothen, Assefaw Gebremedhin, Florin Dobrian

Umit Catalyurek

Erik Boman, Karen Devine, Bruce Hendrickson

Paul Hovland, Boyana Norris, Jean Utke

Michelle Mills Strout