The Zoltan2 Toolkit: Partitioning, Task Placement, Coloring and Ordering

FASTMath Team Members: K. Devine, M. Deveci, V. Leung, S. Rajamanickam, M. Wolf (Sandia); G. Diamond (Rensselaer)
In collaboration with E. Boman, J. Brandt, A. Gentle, S. Olivier, K. Pedretti, L.A. Riesen (Sandia); Ümit Çatalyürek (Ohio State)

Zoltan2: A new toolkit of combinatorial algorithms addressing the needs of parallel applications on emerging architectures

### Zoltan2 Overview and Goals
- Algorithms needed by applications on emerging architectures
  - Partitioning and task placement for hierarchical memory systems
  - Node-level coloring for multi-threaded parallelism
- Multi-threaded implementations that run on emerging architectures
- Support for very large application problem sizes
  - Templated data types for local and global indices
- Application-focused interface supporting meshes, matrices, vectors, particles, coordinates, graphs
- Open-source software in Trilinos’ next-generation solver stack

<table>
<thead>
<tr>
<th>Successor to the Widely Used Zoltan Toolkit</th>
<th>Zoltan</th>
<th>Zoltan2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelism</td>
<td>MPI-only</td>
<td>MPI+X</td>
</tr>
<tr>
<td>API</td>
<td>Application builds model (e.g., graph, hypergraph) for algorithm</td>
<td>Application describes its data (matrix, mesh); algorithm builds model</td>
</tr>
<tr>
<td>Capabilities</td>
<td>Parallel partitioning</td>
<td>Parallel partitioning</td>
</tr>
<tr>
<td></td>
<td>Parallel coloring</td>
<td>Architecture-aware task placement</td>
</tr>
<tr>
<td></td>
<td>Global and local ordering</td>
<td>On-node coloring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>On-node ordering</td>
</tr>
<tr>
<td>Optional TPLs</td>
<td>PT-Scotch (INRIA/LaBRI) ParMETIS (U. Minnesota) PaToH (Ohio St. U.)</td>
<td>PT-Scotch (INRIA/LaBRI) ParMETIS (U.Minnesota) ParMA (Rensselaer) AMD (U.Florida) LDMS: Lightweight Distributed Metric Service (Sandia)</td>
</tr>
<tr>
<td>Maturity</td>
<td>Highly mature; maintenance only</td>
<td>Research platform for emerging architectures</td>
</tr>
<tr>
<td>Integration</td>
<td>No dependence on Trilinos</td>
<td>Integrated with Trilinos next-generation software stack</td>
</tr>
<tr>
<td>Language</td>
<td>C (with F90 &amp; C++ APIs)</td>
<td>Templated C++11</td>
</tr>
<tr>
<td>Distribution</td>
<td>Stand-alone or in Trilinos</td>
<td>In Trilinos</td>
</tr>
</tbody>
</table>

### Scalable Partitioning
- Assign data/work to processors so that processor idle time and interprocessor communication are minimized
- **MultiJagged**: Multi-threaded geometric (coordinate) partitioning
  - MPI+OpenMP implementation
  - Multisection has less data movement, greater scalability than Recursive Coordinate Bisection
  - Fast; scalable; enforces geometric locality
  - Good for adaptivity, particles, contact detection
- **Topology-based (graph, hypergraph, mesh) partitioning**
  - Explicitly models communication costs through data dependencies
  - Good for mesh-, matrix- and network-based applications
  - Integrated with:
    - Graph partitioning: PT-Scotch; ParMETIS
    - Hypergraph partitioning: Zoltan
    - Mesh partitioning & partition improvement: ParMA (Rensselaer)

### Architecture-Aware Task Placement
- Given a (possibly non-contiguous) node allocation in a parallel computer, reduce application communication costs and runtime by placing interdependent MPI tasks on “nearby” cores
- Important in extreme-scale systems:
  - Allocations can be sparse and spread far across the network
  - Messages can travel long routes
  - Increasing locality reduces congestion and communication time
- **Approach**:
  - Use geometric proximity of tasks’ data as a proxy for communication costs between tasks
  - Apply MultiJagged partitioner to order both tasks’ data and nodes’ coordinates
  - Map a task to the core with the same part number

### On-Node Balanced Graph Coloring
- Coloring: Label graph vertices so that adjacent vertices have different labels and the number of labels is small
  - Good for on-node parallelism: Each label is an independent set that can be computed in parallel
- **Balanced coloring**: Label roughly the same number of vertices with each label, at the possible expense of using slightly more labels
  - Important for GPUs: labels with too few vertices cause idle time

### On-Node Matrix Ordering
- Find permutation of local matrix that reduces fill during factorization
  - Reverse Cuthill-McKee
  - Sorted Degree
  - Approximate Minimum Degree, via AMD
- Used, e.g., in Trilinos’ IFPACK2 sparse-matrix preconditions

### Ongoing and Future Work
- Integration of Kokkos performance-portable programming model (Edwards, Trott; Sandia) into Zoltan2 interface and algorithms
- KokkosKernels: New toolkit of on-node Kokkos-based graph algorithms
- Task placement for new network topologies (e.g., Dragonfly)
- Interface to PULP (Slota, Madduri, Rajamanickam; PSU, Sandia) for partitions that minimize multiple constraints & objectives

**More Information**: [http://www.fastmath-scidac.org](http://www.fastmath-scidac.org) or contact Karen Devine, kddevin@sandia.gov