Mesh Partitioning and Load Balancing

Sabina Rips

Contents:
- Introduction / Motivation
- Goals of Load Balancing
- Structures
- Tools

Flow Chart of a Parallel (Dynamic) Application

Partitioning of the initial mesh

Computation

Iteration step
Data exchange

+ Meshrefinement and/or - movement

Imbalance

Efficiency loss

Loadredistribution
Mesh Partitioning and Load Balancing

**Structure of a partitioned mesh**

![Diagram showing global mesh and partitioned mesh with halo-cells](image)

**Application Example from HPS-ICE Project**

Example: Model DBAG, 360° angle of crankshaft, 148,342 mesh cells
Application Example from HPS-ICE Project

At 540° angle of crankshaft: 273,738 mesh cells

Load Imbalance

Between 360° and 540°: Increase of meshsize by a factor of 2
Without Loadbalancing:
  2 Processors are sharing the additional load
  Significant loss of efficiency
Goals of Load Balancing

1. Load Balance

2. Minimal communication costs
   - minimize the number of boundary cells, created by the mesh partitioner
   - minimize the number of processors, one has to communicate with

3. Minimal migration costs

Steps of a Load Balancing Procedure

2 steps of load distribution:

1. Decision:
   - how many and
   - which cells should be moved
   - where
   † Load balancing algorithms (→ Tools)

2. Migration:
   move the mesh cells to the selected processors
   † has to be done by the application program
How Do Load Balancer Operate?

How can I handle the problem, using [known] methods/algorithms?

Mapping of the problem to known structures:

**Graphs**

Nodes of the graph $\leftrightarrow$ computation costs
Edges of the graph $\leftrightarrow$ communication costs

Mapping Mesh $\Rightarrow$ Graph

Mesh

Graph
Further information represented in a graph:

- Computation costs ↔ Weighting of nodes
- Communication costs ↔ Weighting of edges

Example:

![Graph example diagram](image)

Goals of Graph Algorithms

1. Equal number of nodes within each partition  
   (weighted: equal sum of node weights within each partition)

Partition 1

Partition 2

load balance: 100 %  
(weight of cut edges: 4)
Goals of Graph Algorithms

2. Minimal number of cut edges
   *(weighted: minimal sum of cut edges weight)*

Partition 1

Partition 2

Load balance: 100%
Weight of cut edges: 3

Max. processor degree: 3
Max. processor degree: 2
Mesh Partitioning and Load Balancing

Sabina Rips

Slide 15
HPC Center Stuttgart

Goals of Graph Algorithms

Task for non weighted graphs:
find a decomposition of the graph,

• with preferably the same number of nodes within each partition
• with a minimal number of cut edges between the partitions

NP-complete problem
(it cannot be solved in reasonable time in case of bigger problems)

use of heuristics,
which give a solution near the optimal one

Tools

• Jostle: (Chris Walshaw, University of Greenwich)
  • initial graph partitioning
  • dynamic load distribution
  • serial and parallel
  • http://www.gre.ac.uk/jostle

• Metis: (George Karypis, University of Minnesota)
  • multilevel partitioning algorithms
  • initial partitioning
  • serial
  • (parMetis: parallel version)
  • http://www.cs.umn.edu/~karypis/metis/metis.html
  • HLRS: http://www.hlrs.de/organization/par/services/tools/loadbalancer/metis.html

• Jostle: (Chris Walshaw, University of Greenwich)
  • initial graph partitioning
  • dynamic load distribution
  • serial and parallel
  • http://www.gre.ac.uk/jostle

• Metis: (George Karypis, University of Minnesota)
  • multilevel partitioning algorithms
  • initial partitioning
  • serial
  • (parMetis: parallel version)
  • http://www.cs.umn.edu/~karypis/metis/metis.html
  • HLRS: http://www.hlrs.de/organization/par/services/tools/loadbalancer/metis.html

Tools

• Jostle: (Chris Walshaw, University of Greenwich)
  • initial graph partitioning
  • dynamic load distribution
  • serial and parallel
  • http://www.gre.ac.uk/jostle

• Metis: (George Karypis, University of Minnesota)
  • multilevel partitioning algorithms
  • initial partitioning
  • serial
  • (parMetis: parallel version)
  • http://www.cs.umn.edu/~karypis/metis/metis.html
  • HLRS: http://www.hlrs.de/organization/par/services/tools/loadbalancer/metis.html

Slide 16
HPC Center Stuttgart

32. — Mesh Partitioning and Load Balancing —

32-8
Jostle

- operates with an already **partitioned and distributed mesh**
- in the initial case: at first partitioning of the mesh using simple methods
- operates in general **parallel**
  - no need of gathering the data on one processor
  - partitioning effort is spread to the processors
  - migrates boundary layers to the neighbouring processors

  ↑ **minimization of cell migration**

Jostle

- available as stand-alone program (only serial version) and as procedure call (C und FORTRAN)
- different graph formats possible
- controlling the behaviour is possible by using parameters (e.g. load imbalance)

- Call (C):

  ```c
  void pjostle(int *nnodes, int *offset, int *core, int *halo,
  int *index, int *degree, int *node_wt, int *partition,
  int *local_nedges, int *edges, int *edge_wt,
  int *network, int *processor_wt, int *output_level,
  int *dimension, double *coords);
  ```
Jostle (serial)

Balance: ≈100 %
Cut edges: 10,509
Run time (SGI R4600): 58,9 sec

8 Partitions, 273,738 Cells

Jostle (parallel)

Balance: ≈ 100 %
Cut edges: 10,638
Run time (SGI R4600): 532,2 sec

(in case of running 8 MPI-Processes)

8 Partitions, 273,738 Cells
Metis

- serial graph partitioning
- available as standalone program and as procedure call (C und FORTRAN)
- fixed data format
- consecutive cell numbering required

- Call(C):
  \[
  \text{METIS\_PartGraphRecursive}(\text{n}, \text{xadj}, \text{adjncy}, \\
  \text{vwgt}, \text{adjwgt}, \text{wgtflag}, \text{numflag}, \\
  \text{nparts}, \text{options}, \text{edgecut}, \text{part});
  \]

ParMetis

- static & dynamic partitioning
- consecutive cell numbering required
- no settings possible

- Call for graph repartitioning (C):
  \[
  \text{ParMETIS\_RepartLDiffusion}(\text{vtxdist}, \text{xadj}, \\
  \text{adjncy}, \text{vwgt}, \text{adjwgt}, \text{wgtflag}, \text{numflag}, \text{options}, \text{edgecut}, \text{part}, \\
  \text{MPI\_Comm}, \text{comm});
  \]
Mesh Partitioning and Load Balancing

8 Partitions, 273,738 Cells

Balance: 100%
Cut edges: 10,007
Run time (SGI R4600): 34.8 sec

Tools: Comparison

<table>
<thead>
<tr>
<th>Tool</th>
<th>Cut edges</th>
<th>Run time (sec)</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jostle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ser.</td>
<td>10.009</td>
<td>58.9</td>
<td></td>
<td>worse results</td>
</tr>
<tr>
<td>Par.</td>
<td>10.638</td>
<td>532.2</td>
<td></td>
<td>arbitrary cell numbering, various user settings</td>
</tr>
<tr>
<td>Metis</td>
<td>10.007</td>
<td>34.8</td>
<td>fast, good results</td>
<td>serial</td>
</tr>
</tbody>
</table>
Summary

Strategy:
1. At first:
   - **initial partitioning:**
     a. choose appropriate serial partitioner
        (Jostle, Metis)
     b. distribute the cells to the processors
2. In case of an existing load imbalance:
   - **dynamic load balancing:**
     a. cost-benefit-calculation
     b. in case of benefit:
        - when migration tool available:
          - parallel LB ↑ Jostle
        - else:
          - gather cell info on a particular processor
          - proceed like in case of initial partitioning

Overview

Initial Partitioning

Chaco → Metis → Jostle

Central processor

Processor 1

Computation

Load imbalance

Cost - Benefit - Calculation

Migration tool

No

Yes

Jostle

Cell exchange

Processor 2

Computation

Load imbalance

Cost - Benefit - Calculation

Migration tool

No

Yes

Jostle

Cell exchange