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

Efficiency loss
Loadredistribution

Imbalance
Structure of a partitioned mesh

Global mesh

Partitioned mesh

Halo-Cells

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 mesh size by a factor of 2
Without Loadbalancing:
  2 Processors are sharing the additional load
  Significant loss of efficiency
Goals of Loadbalancing

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 Loadbalancing 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 ↔ computation costs
Edges of the graph ↔ communication costs
Graph

Further information represented in a graph:

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

Example:

Goals of Graph Algorithms

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

   Partition 1
   2
   1
   1
   3

   Partition 2
   2
   1
   1
   3
   1

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

2a. Minimal processor degree
   (node degree: number of neighbours)

Max. processor degree: 3
Max. processor degree: 2
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

\( \text{NP-complete} \) 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

1. **Jostle:** (Chris Walshaw, University of Greenwich)
   - initial graph partitioning
   - dynamic load distribution
   - serial and parallel
   - http://www.gre.ac.uk/jostle
   - HLRS: http://www.hlrs.de/organization/par/services/tools/loadbalancer/jostle.html

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

• 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 and FORTRAN)
- fixed data format
- consecutive cell numbering required

Call(C):

```c
METIS_PartGraphRecursive(int *n, idxtypen *xadj, idxtypen *adjncy,
                        idxtypen *vwgt, idxtypen *adjwgt, int *wgtflag, int *numflag,
                        int *nparts, int *options, int *edgecut, idxtypen *part);
```

### ParMetis

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

Call for graph repartitioning (C):

```c
ParMETIS_RepartLDiffusion(idxtypen *vtxdist, idxtypen *xadj,
                           idxtypen *adjncy, idxtypen *vwgt, idxtypen *adjwgt, int *wgtflag,
                           int *numflag, int *options, int *edgecut, idxtypen *part,
                           MPI_Comm *comm);
```
### Metis

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

8 Partitions, 273,738 Cells

### 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>10,509</td>
<td>58.9</td>
<td>serial &amp; parallel</td>
<td>worse results</td>
</tr>
<tr>
<td></td>
<td>10,638</td>
<td>53.2</td>
<td>arbitrary cell numbering</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>various user settings</td>
<td></td>
</tr>
<tr>
<td>Metis</td>
<td>10,007</td>
<td>34.8</td>
<td>fast</td>
<td>serial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>good results</td>
<td></td>
</tr>
</tbody>
</table>

- Various user settings
- Worse results
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

Processor n
- Computation
- Load imbalance
- Cost - Benefit - Calculation
- Migration tool
  - Yes
  - No
- Collect cell information
- Cell exchange

Central processor

Migration tool
- Yes
- No