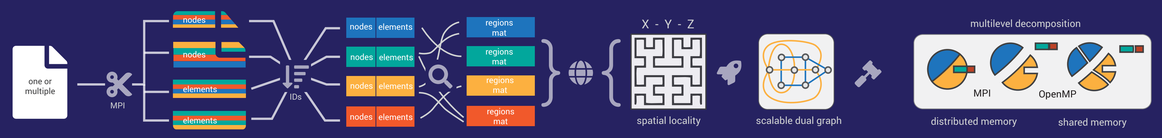
WORKFLOW FOR PARALLEL PROCESSING OF SEQUENTIAL MESH DATABASES



IT4Innovations national supercomputing center Czech Republic

Ondrej Meca ondrej.meca@vsb.cz | Lubomir Riha lubomir.riha@vsb.cz
Tomas Brzobohaty tomas.brzobohaty@vsb.cz

LOADER ARM sequential processing 1270 sec.

169M Nodes
85M Elements

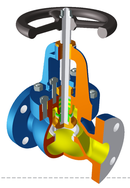
parallel processing
10 sec

VALVE sequential processing 443 sec.

32M Nodes
23M Elements

almost all external faces in regions

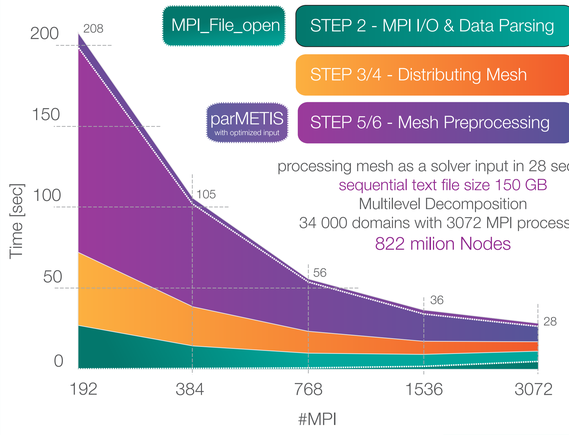
parallel processing
5 sec



JET ENGINE - Parallel Scalability

MPI_File_open
STEP 2 - MPI I/O & Data Parsing
STEP 3/4 - Distributing Mesh
parMETIS with optimized input
STEP 5/6 - Mesh Preprocessing

processing mesh as a solver input in 28 seconds
sequential text file size 150 GB
Multilevel Decomposition
34 000 domains with 3072 MPI processes
822 milion Nodes



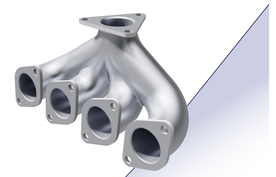
The number of MPIs used for mesh processing is enough for the in-house hybrid solver ESPRESO, which is able to fully utilize modern supercomputers with hybrid architectures.

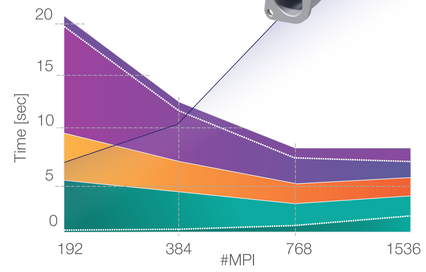
MANIFOLD sequential processing 1359 sec.

100M Nodes
73M Elements

almost all external faces in regions

parallel processing
7 sec



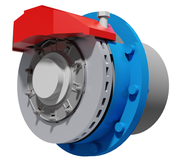


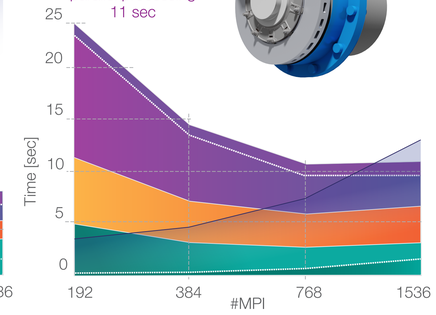
DISC BRAKE sequential processing 938 sec.

122M Nodes
45M Elements

12 element regions
30 face regions

parallel processing
11 sec



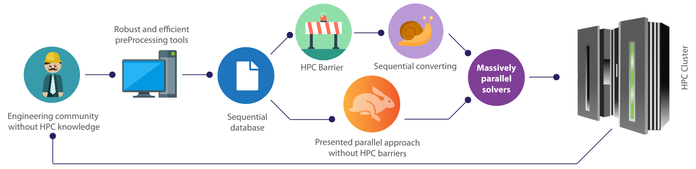


Graphs are composed from areas that correspond with the steps outlined at the bottom. The first (green) area shows times for reading and parsing a file. The highlighted part shows the increasing time of MPI_File_open. It introduces an overhead for the large number of MPI processes (the time is not dependent on the size of a read file). We include this function in the measurements because we present the total time needed to prepare a mesh for a solver. On the other hand, MPI I/O is out of the scope of our approach. Hence, we have not implemented any kind of optimizations. The second (orange) area shows times for parallel sorting according to IDs, assigning regions, and clusterization of a mesh via SFC. The last (purple) area shows times for improving the quality of the decomposition. It includes computation of the dual graph and calling ParMETIS (its computation time is also highlighted). The last area also contains the second level decomposition (using METIS), a synchronization of regions, rearranging nodes and elements, etc. These steps are required for the effective functioning of our parallel solver.

parMETIS time without clusterization

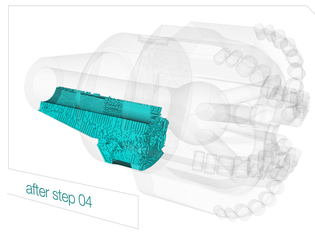
For comparison, graphs include a reprezentation of parMETIS when clusterization is excluded from the processing workflow. If the finite element model contains lots of element regions, the running time of parMETIS does not rise so quickly (natural clusterization). The parMETIS running time in the optimized workflow is in the graphs included in the mesh processing stage.

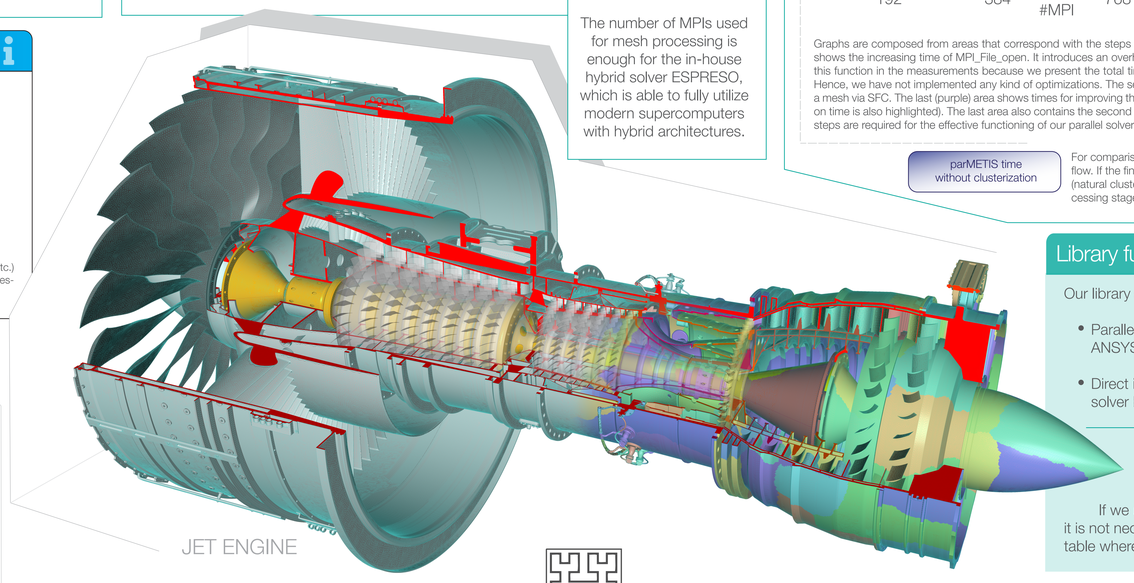
MOTIVATION



Our motivation is to create a library connecting between tools for the creation of complex engineering models (such as ANSYS, HyperMesh, ANSA, ABAQUS, etc.) along with open source parallel solvers enabling the broader usage of HPC by the engineering community. The result of this direct connection is robust preprocessing together with possibility to connect various highly parallel solvers that are able to solve non-standard problems.

Spatial Locality - elements on MPI RANK 0





Library functionality

Our library can be used in the two following ways:

- Parallel database files converter - external sequential files to binary parallel format ANSYS, ABAQUS, etc. to ESPRESO, OpenFOAM, etc.
- Direct input to the parallel solvers - the proposed library is part of our scalable ESPRESO solver based on hybrid FETI methods for solution of multiphysical problems.

Domain decompositon can follow element regions or element types. This functionality enhances the scalable properties of our parallel algorithms.

If we use direct input to the parallel solvers for multiple solutions on the same database, it is not necessary to have a fixed number of MPIs (domains) in a parallel binary file. This is suitable where there is variability in the availability of computational resources at the desired time.

Six steps for scalable data processing

STEP 01

INPUT EXTERNAL DATA

Create data in external tools

As an example, we present results based on the input data file generated by the commercial ANSYS software. The input file is a text file that contains a full description of a finite element model, and the data are organized in blocks, the properties of which are defined by commands.
The major part of the file contains a description of nodes and elements. These blocks are followed by another problem description such as definition of regions, materials, boundary and initial conditions, etc.

Many commercial software packages, such as ABAQUS, NASTRAN etc. have very similar command structures for the database files.

This type of files do not suppose parallel processing.

STEP 02

MPI I/O & DATA PARSING

Loading and understanding data

The first step is to load data from one or multiple files to the main memory. This is performed by a collective MPI file reader. Each process reads approximately the same number of bytes that are aligned, to allow parsing by each process separately.

After loading is done, interpretation of the input language is performed. This is based on the processing of the data and commands in the ANSYS text file structure. Detected blocks, their positions, and parameters are exchanged among all MPI processes.

A similar approach can easily be used for other command based database files (ABAQUS, NASTRAN)

STEP 03

PARALLEL SORTING

Balance distributed data

Since elements and coordinates IDs are randomly distributed, processes with elements do not known where to ask for the coordinates. This fact also complicates the process of assigning the regions that are usually given by a list of nodes or elements IDs. In order to ensure approximately the same amount of work for all processes, we first balance the nodes and elements, and then parallel sorting is applied according to given sets of IDs.

After the sorting step, distribution of the nodes can easielly be described by a vector. Due to this, elements know where to ask for coordinates.

This step allows assigning elements, nodes, and faces regions.

STEP 04

SPATIAL LOCALITY - CLUSTERIZATION

Space Filling Curve decomposition

The main problem of loading a mesh in parallel is the non-contiguous distribution of elements among processes. To be able to achieve a good scalability of the mesh processing, the following properties of a mesh are required:

- the mesh elements have to be uniformly distributed to spatially close clusters,
- the number of neighbouring processes must be as small as possible.

External tools for mesh generators do not take this into account, because the spatial position of elements is irrelevant. To keep the communication overhead relatively small, elements have to be re-arranged.
In this step, elements that are distributed according to their IDs are redistributed into clusters - a set of elements that is assigned to one particular process and fulfill the spatial locality requirements.

The clusterization is based on the Hilbert Space Filling Curve, which has minimal input requirements (coordinates only) and allows effective parallelization. The speed and scalability of the algorithm is enhanced at the expense of the quality of the final distribution.

Geometric decomposition is crucial for scalability of the next step.

STEP 05

DUAL GRAPH

Scalable solution

Decompositions based on spatial locality are generally worse, in terms of finite element solver requirements, than dual graph based approaches, especially for problems with complicated geometries.

Geometric decomposition reduces the number of neighbors and allows the distributed dual graph to be computed efficiently in parallel. For improving solver scalability, the dual graph also captures aditional mesh informations like bodies, regions, and element types.

The dual graph is suitable for decomposition of complex geometries with various parameters.

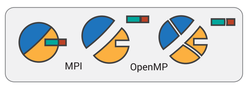
STEP 06

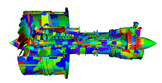
DOMAIN DECOMPOSITION

Multilevel approach

A well balanced decomposition of a mesh directly impacts the scalability of the applied solvers. For the final domain decomposition, we use libraries that provide "high quality" decomposition and support the MPI parallelization needed for HPC clusters. (ParMETIS or ptSCOTCH). These libraries have neighbor dependent scalability.

The described workflow prepares multilevel decomposition suitable for massively parallel solvers.





Would you like to know more?

Acknowledgement
This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPS II) project "IT4Innovations excellence in science - LQ1602" and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center - LM2015070".
This work is partially supported by project EXPERTISE - models, EXperiments and high PERformance computing for Turbine mechanical integrity and Structural dynamics in Europe, http://www.msca-expertise.eu, and also partially supported by the SGC grant No. SP2018/159 „Hardware acceleration of matrix assembler and GUI development of ESPRESO library", VSB-TU Ostrava.