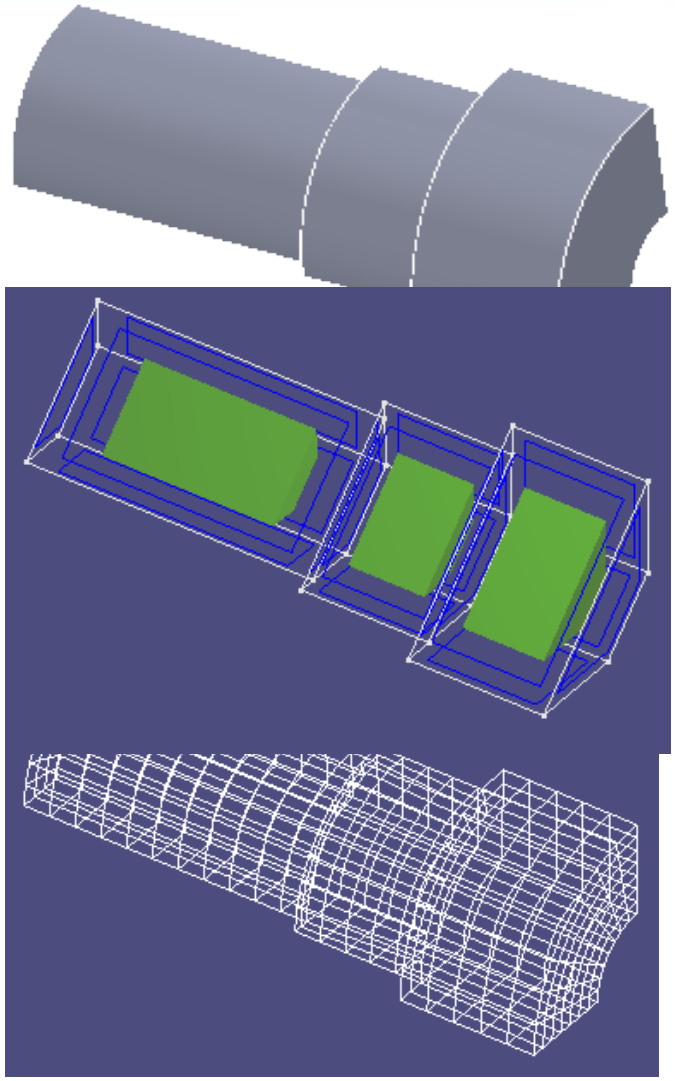
ГОСУДАРСТВЕННАЯ КОРПОРАЦИЯ ПО АТОМНОЙ ЭНЕРГИИ «РОСАТОМ»

Parallelization and Optimization of Multiblock Grid Generation in LOGOS Preprocessor

Speaker: Lazarev V.V.

Multiblock grid preparation method in LOGOS preprocessor

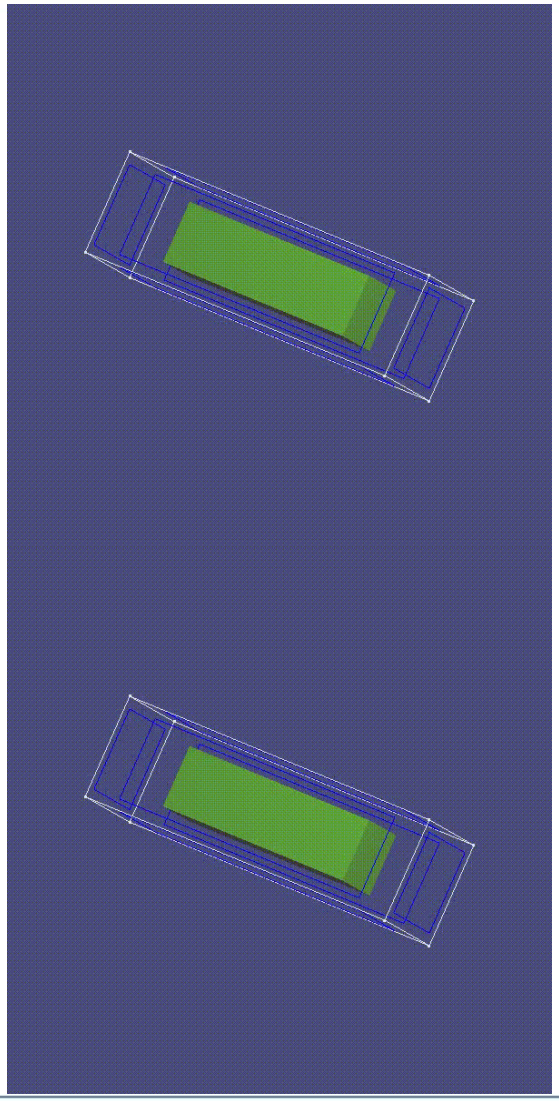


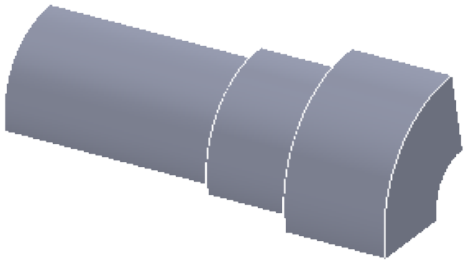
Original geometry is a geometric model in analytic representation.

Blocking is decomposition of original geometry into blocks.

Multiblock grid is a composite grid where structured subgrids share common interfaces.

Multiblock grid generation process





Additional graphical representations of blocks are superimposed on geometry

It took the user 33 operations in 15 minutes to prepare the blocking shown here.

Decomposition of complex geometries takes several weeks and months. User re-generates the multiblock grid many times.

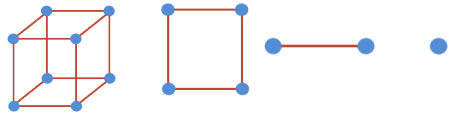
Grid generation time is bottleneck.

Internal representation of blocking

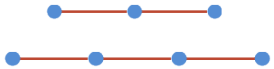
Blocking consists of block topology and geometry:

- block topology is description of block connections;
- block geometry is shape and arrangement of blocks in space.

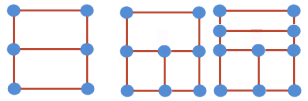
Block is defined by 6 faces; face, by 4 edges; and edge, by 2 vertices.



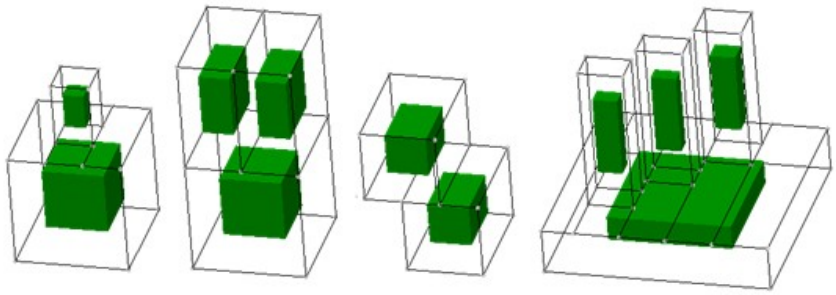
Complex edge is an edge composed of two edges having a common vertex, each of which can be complex, too.



Complex face is a face composed of two faces having a common edge, each of which can be complex, too.



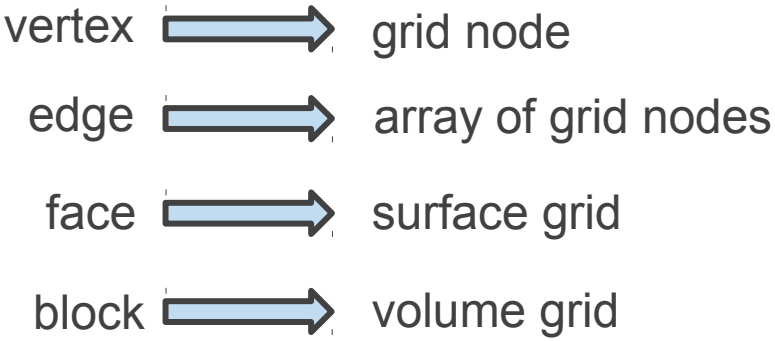
Examples of blocking, where interface is represented by complex faces and edges.



Internal representation of multiblock grid

Multiblock grid is a composite conformal grid consisting of structured subgrids.

Multiblock grid has fragment representation. Each element in blocking is associated with a self-contained grid fragment.



Adjacent blocks have conforming interfaces (common grid nodes).

Grid conversion from fragment representation into other formats (e.g. EFR, NGEOM) is performed by special fragment merging algorithms (not discussed here).

Parallelization and optimization of grid generation

- Parallel grid generation by fragments
- Parallel grid generation in a fragment
- Grid re-generation only on modified blocks
- Grid writing to file by fragments

Parallel grid generation by fragments

Program code paralleled using OpenMP.

Fragment representation of multiblock grid simplifies implementation of parallel algorithm.

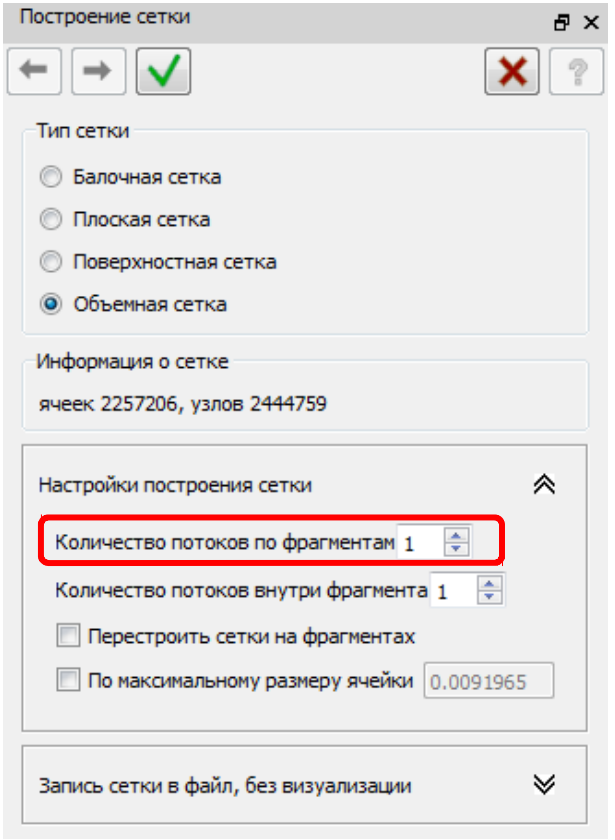
Dynamic balancing by OpenMP tools.

```
#pragma omp parallel for schedule(dynamic)
for( int iBlock = 0; iBlock < nBlocks; ++iBlock )
```

Grids are first generated at the boundary (vertices, edges, faces) and then on fragments.

Progress bar and process canceling implemented.

Graphical user interface for grid generation settings



Parallel grid generation by fragments

Testing of parallelization by fragments

Computer specification:

- four 6-core Intel Xeon X7542 processors, 2.67 GHz frequency;
- 256 GB RAM.

Original geometry is a quarter of a nuclear reactor melt trap.

Blocking consists of 1303 blocks.

Resulting multiblock grid:

- 2.25 million cells;
- 4.58 million cells;
- 8.80 million cells.

Parallel grid generation by fragments

Grid generation time as a function of the number of threads by fragments, s

Problem size, million cells	Number of threads by fragments									
	1	2	3	4	5	6	7	8	9	10
2.25	154.2	84.1	57.4	40.3	36.4	33.6	30.7	27.0	25.5	25.5
4.58	318.0	171.3	127.7	82.3	74.7	70.6	62.7	58.3	54.7	53.1
8.80	499.6	266.3	193.0	120.2	112.6	103.3	91.3	88.4	83.0	79.0

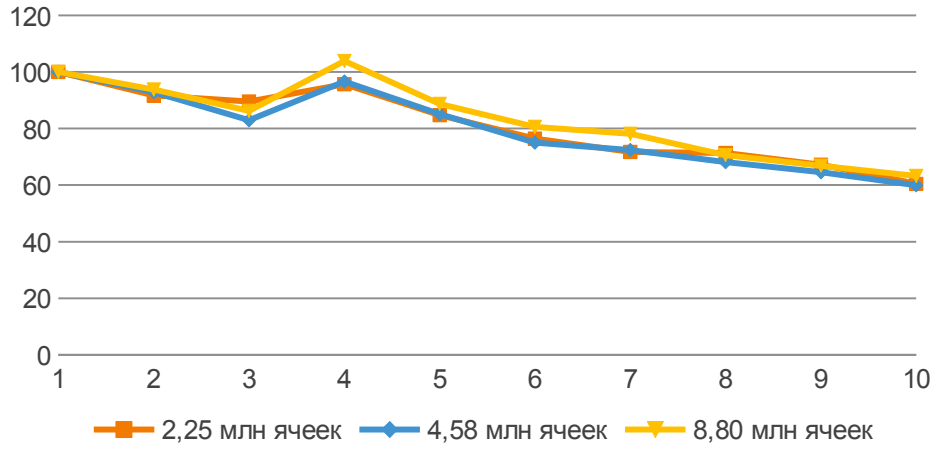
Parallelization efficiency

$$E_n = \frac{T_1}{T_N * N} * 100\%$$

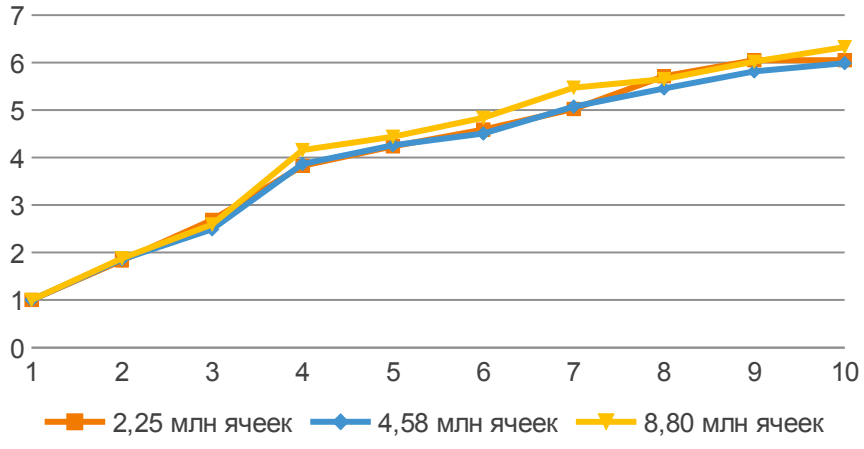
Speedup

$$Sp_N = \frac{T_1}{T_N}$$

Parallelization efficiency, %



Speedup



Parallel grid generation in a fragment

Structured grids are represented by two- and three-dimensional node arrays.

Steps of structured grid generation on fragment:

- boundary assembly from down fragments;
- decomposition of inner volume by the number of threads;
- calculation of inner nodes.

Consider decomposition: $(Array, N_{max}, V_{min}) \rightarrow [frag_1, \ldots, frag_N]$

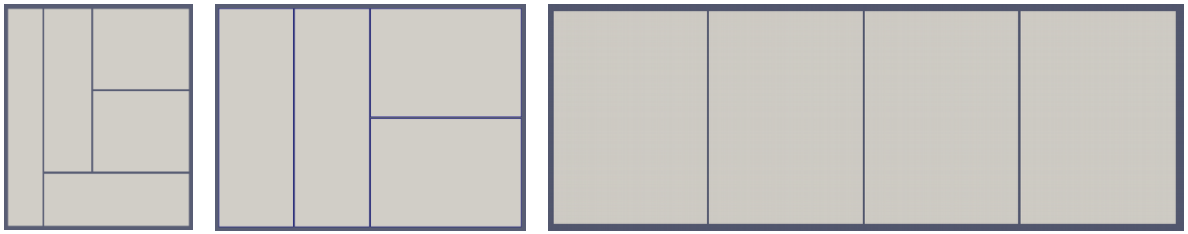
N_{max} Max number of fragments.

V_{min} Min volume of fragment.

$$N_{max} \geq N, |frag_i| \geq V_{min}$$

Decomposition is performed by clipping the required volume from the array on the larger fragment side.

Examples of two-dimensional array decompositions



Parallel grid generation in a fragment

Grid generation time as a function of the number of threads, s.

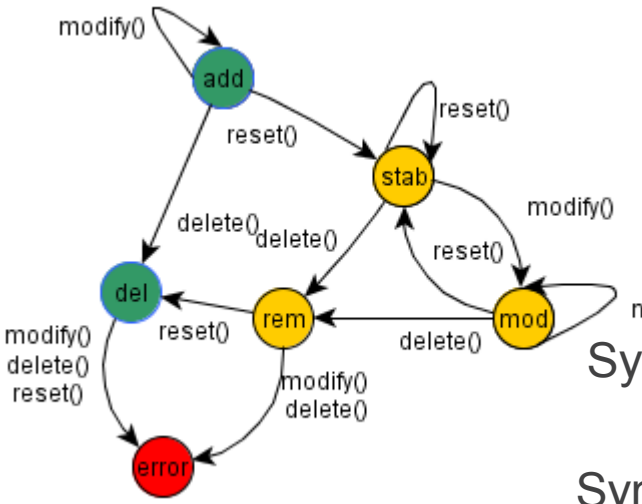
No. of threads by fragments	Total number of threads							
	1	4	8	12	16	20	24	28
1	544.7	148.9	97.8	84.5	71.5	70.5	75.3	74.6
2	–	142.8	81.0	66.7	55.5	51.7	48.2	47.8
4	–	123.4	74.8	64.4	50.3	41.2	40.6	38.6
8	–	–	89.7	–	53.8	–	41.1	–

Rows represent the number of threads by fragments.
Columns represent the total number of threads.
Number of threads in fragment = column value / row value.
Red cells represent drop in speedup ratio.

It was established by fitting that the best result is provided by configuration of 5 by 5 threads, 37.8 s.
Speedup: 14.4
Parallelization efficiency: 58%

Grid of
8.80 million cells

Grid re-generation only on modified blocks



Primary task is to track and sum up the acts of addition, modification and deletion of blocks.

Each block is associated with a finite-state machine that tracks variations in grid fragments it is related to.

Synchronization is making blocking and grid conformal.

Synchronization involves deletion of fragments from multiblock grid in modified or deleted blocks.

Possible states of blocks:

- stable (stab);
- modified (mod);
- added (add);
- marked as remove (rem);
- in queue to delete (del).

Actions to block:

- modify block (modify);
- delete block (delete);
- reset state (reset).

error – error state of block; should not be present ideally; occurs if the block being modified or deleted has been deleted before.

Grid re-generation only on modified blocks

Grid re-generation time on modified blocks

Parallelization by fragments and in fragments disabled

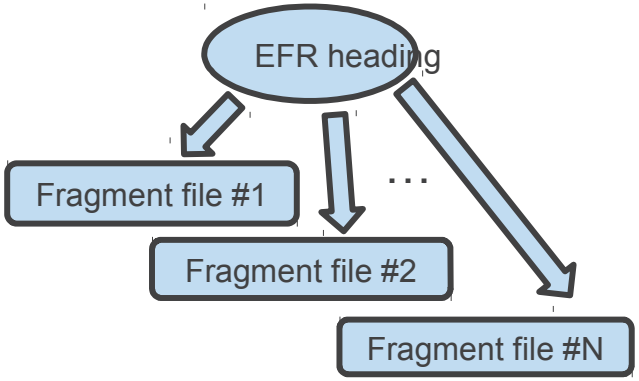
Operation	Time, s.
Generation of volume grid from empty state	153.1
Generation of surface grid from empty state	143.1
Generation of volume grid from existing surface grid	15.7
Repeated generation of volume grid	4.0
Generation of surface grid from existing volume grid	0.7
Repeated generation of surface grid	0.8
Generation of volume grid after varying the number of cells on wall	8.0
Generation of volume grid after excluding 100 fragments	2.1

Grid of
2.25 million cells

Grid writing to file by fragments

Generation of grids not fitting in RAM

Grids are stored in a distributed EFR file (VNIIEF-developed) as grids consisting of self-contained fragments combined by a single file, EFR heading.



Grid-to-file writing algorithm
Choose block from geometry. Generate boundary grid on it. Generate volume grid. Write grid to file. Delete volume grid from memory. Delete boundary grid if there are no adjacent blocks with not-yet-generated grids.

Peak memory consumption does not exceed the size of the largest fragment.

Grid writing to file by fragments

Grid generation and writing to RAM and EFR file

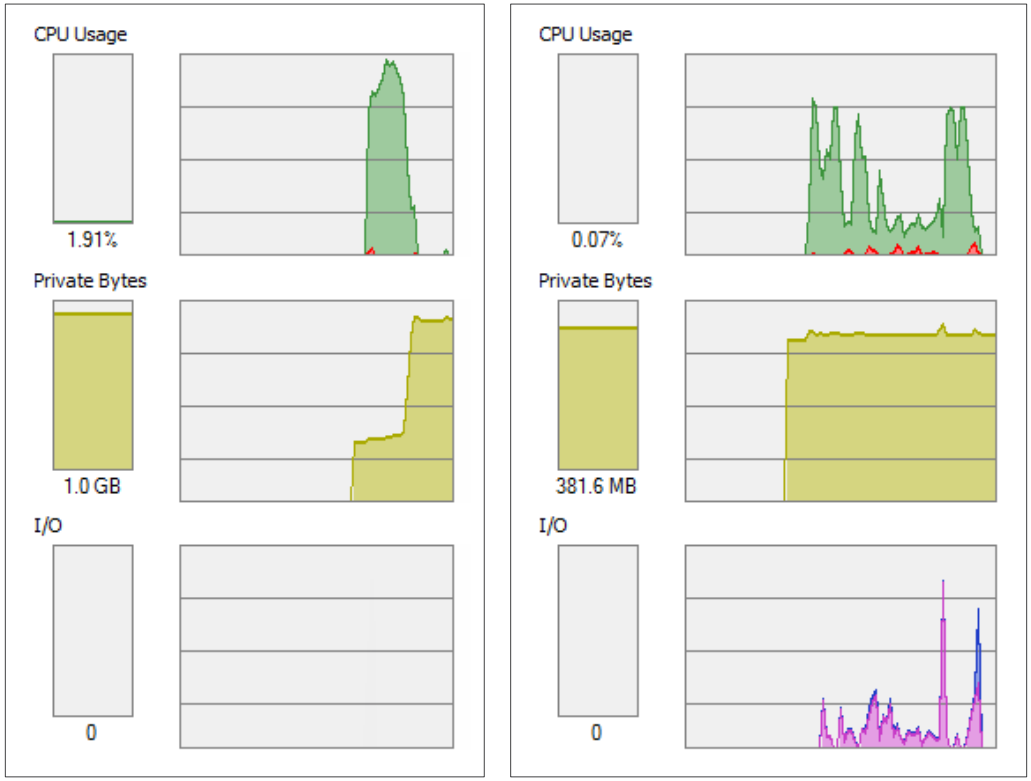
Personal computer specification:
- Intel Core i5-2400, 3.1 GHz;
- 16 GB RAM.

Grid specification:
- 2.25 million cells;
- 2.44 million nodes;
- 6.97 million faces;
- 570 MB on disk.

Initial memory usage by pre-processor: 350 MB

Peak memory usage:
- write to file: 382 MB
- write to RAM: 1024 MB

Write to RAM ***Write to EFR file***



Diagrams of CPU, I/O and memory usage

Grid writing to file by fragments

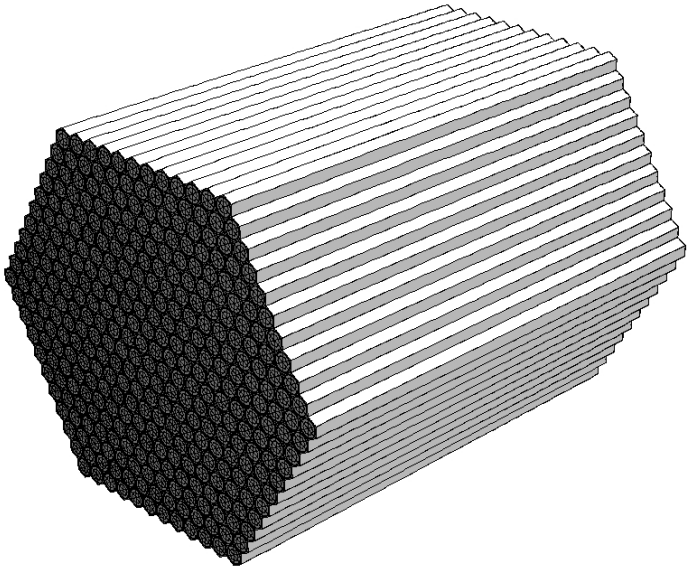
Large-size grid generation and writing to EFR file

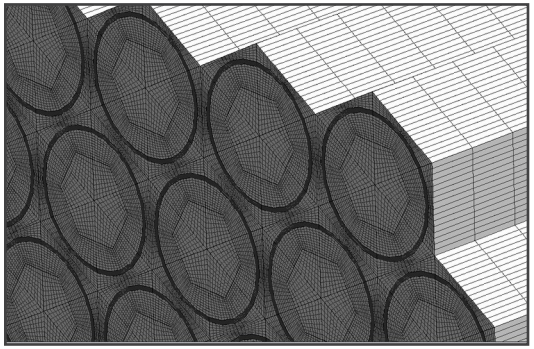
Geometric model of fuel assembly mockup comprised of 352 fuel elements.

Grid specification:

- 1.0 billion cells;
- 1.0 billion nodes;
- 3.2 billion faces;
- 250 GB on disk;
- 8500 fragment files.

Grid generation time: 4 hours
Peak memory usage: 1 GB





Conclusions

- Speedup by a factor of 14.4 and 58-percent parallelization efficiency have been achieved.
- Repeated generation of multiblock grid on modified geometry decomposition keeps track of previous generation.
- Grids of 1 billion cells can be generated on a PC.
- Algorithms have been implemented and integrated in LOGOS ver.5.1 preprocessor.

Thank you