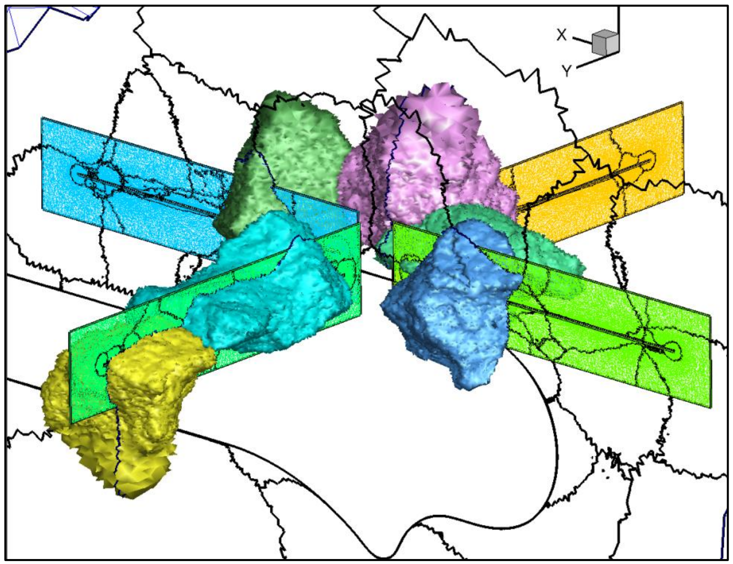
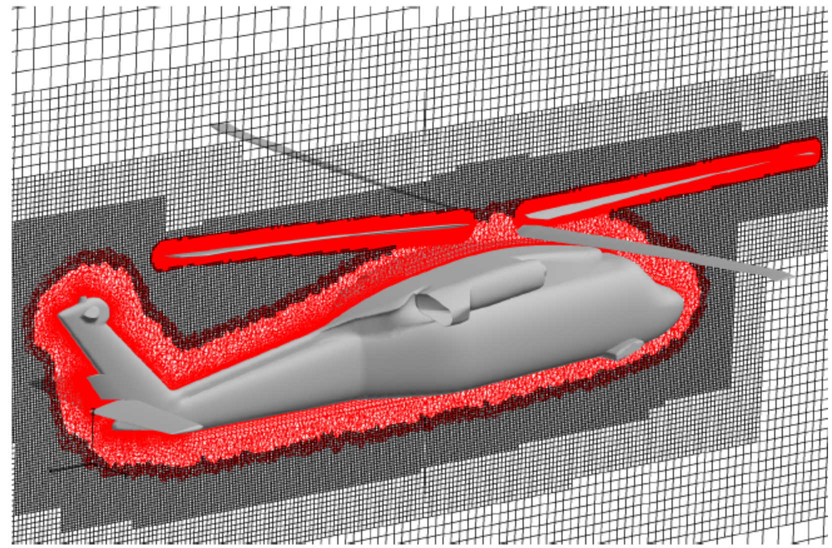
Active Load Balancing for Overset Grid Assembly Procedures





Jay Sitaraman

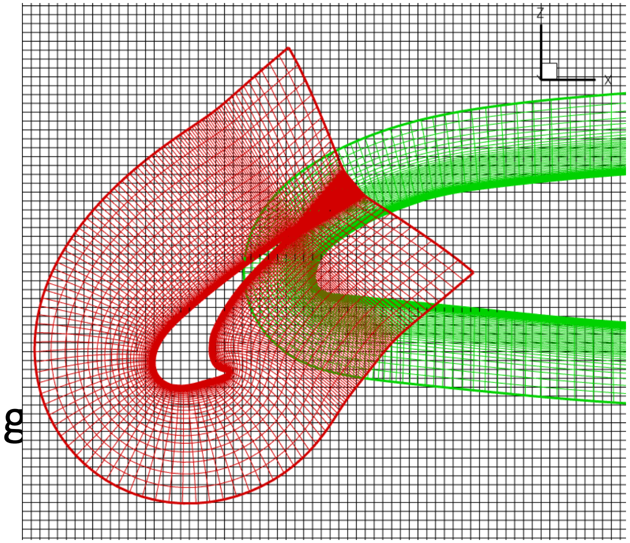
Parallel Geometric Algorithms LLC
Associate Professor University of Wyoming

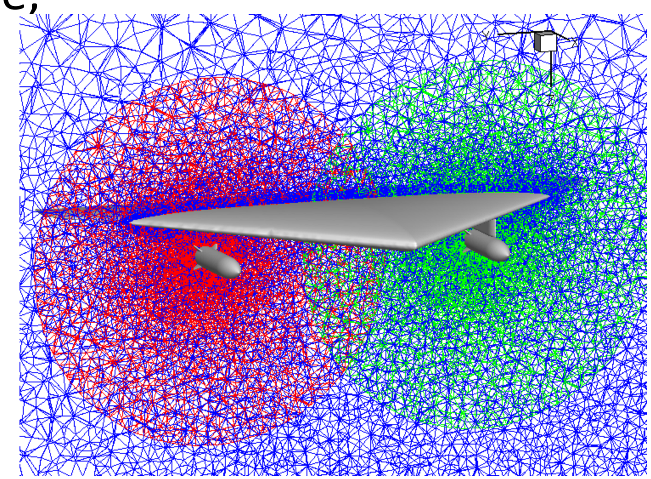
Beatrice Roget

Science and Technology Corporation

Problem Definition

- CFD simulation of complex problems (moving and deforming multiple bodies) require overset meshes
- Overset Grid Assembly method required to identify point types (solver, receptor, hole)
- Many existing OGA codes: PEGASUS5, SUGGAR++/DiRTlib, CHIMPS, OVERFLOW with varying capabilities
- OGA method should be accurate, efficient and scalable, and fully automated.
- Two main challenges for partitioned unstructured meshes and unstructured dual-mesh systems
  - ➢ complex geometry of partition boundaries
    robustness problems for the point-localization
  - ➢ Inherent load imbalance (large variation in the types of mesh-block overlap)
    poor efficiency and scalability





PUNDIT (product of CREATE A/V)

Development history:

- Begin development in early 2008 as part of the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HIARMS)
- First production version in Q4 2008
- Integral part of CREATE A/V Helios (rotary-wing tool) from 2009
- Integral part of CREATE A/V Kestrel (fixed-wing tool) from 2010

Capabilities:

- Based on implicit hole cutting
- Fully parallel and highly automated (no user input)
- Support for node-centered/cell-centered interpolation
- Support for adaptive Cartesian grids
- In production for last 5 years (1000+ different large scale simulations)
- Robust search algorithms
- Improved efficiency and scalability

Primary Developers:
Jay Sitaraman (2008-)
Beatrice Roget (2010-)

Contributors:
Robert Meakin (CREATE A/V)
Mark Potsdam (AFDD)
Rohit Jain (AFDD)
Andy Wissink (AFDD)
Stephen Adamec (CREATE A/V)
Todd Tuckey (Air force)
Dave McDaniel (Air force)
Matt Floros (ARL)

Documentation

<u>Three journal articles and 8 conference papers</u>

- J. Sitaraman, M.Floros, A.Wissink, M.Potsdam, “Parallel Domain Connectivity Algorithm For Unsteady Flow Computations Using Overlapping and Adaptive Grids,” ***Journal of Computational Physics 229(12)(2010) 4703–4723.***

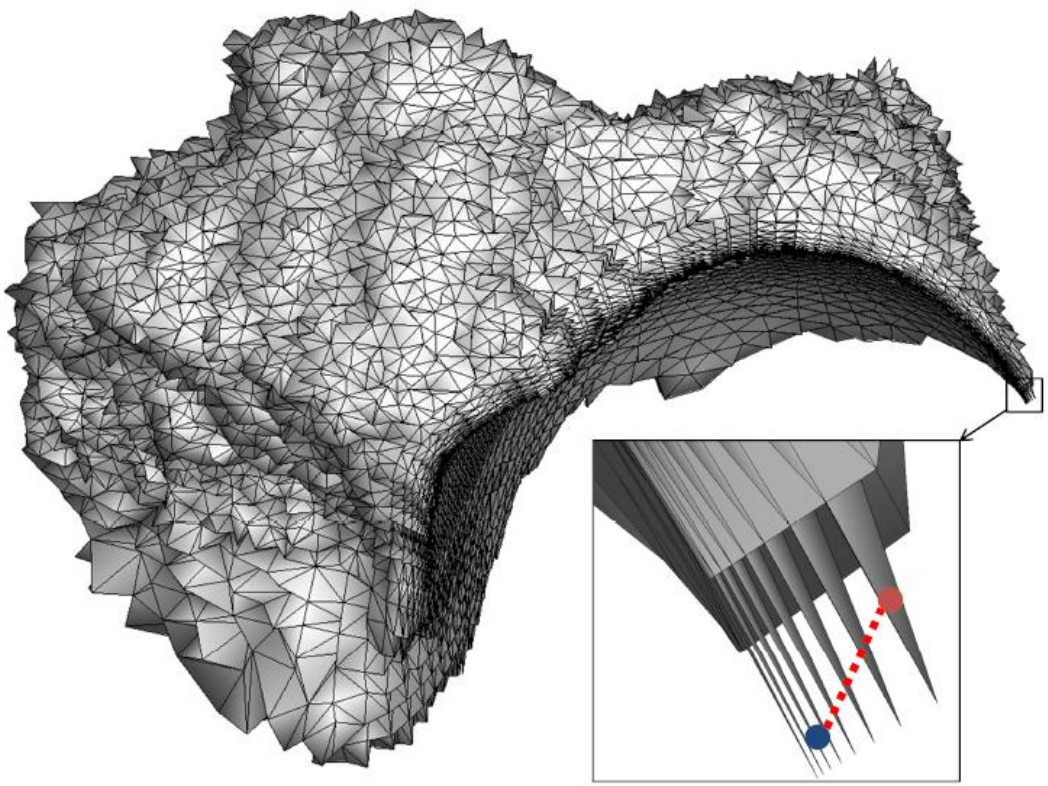
- B.Roget and J. Sitaraman, “Wall Distance Search Algorithm Using Rasterized Marching Spheres,” ***Journal Computational Physics 241 (2013) 76-94.***

- B. Roget and J. Sitaraman, “Robust and Efficient Overset Grid Assembly For Partitioned Unstructured Meshes,” ***Journal of Computational Physics 260 (2014) 1-24***

This presentation is a synopsis of all of the above with focus on the last journal article.

Partition boundary problem

Unstructured Mesh-Block Partition



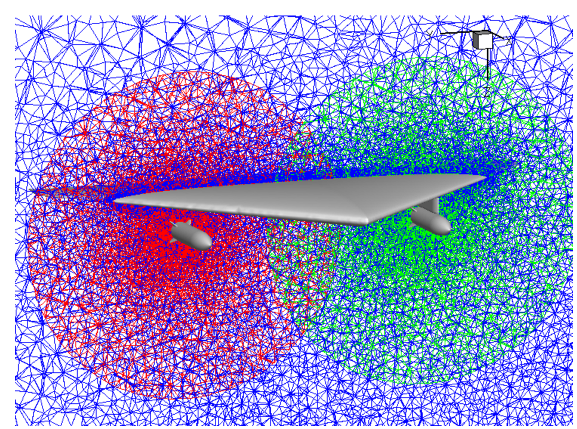
Robustness issue

OGA core task = DONOR SEARCH: find cell(s) containing a point

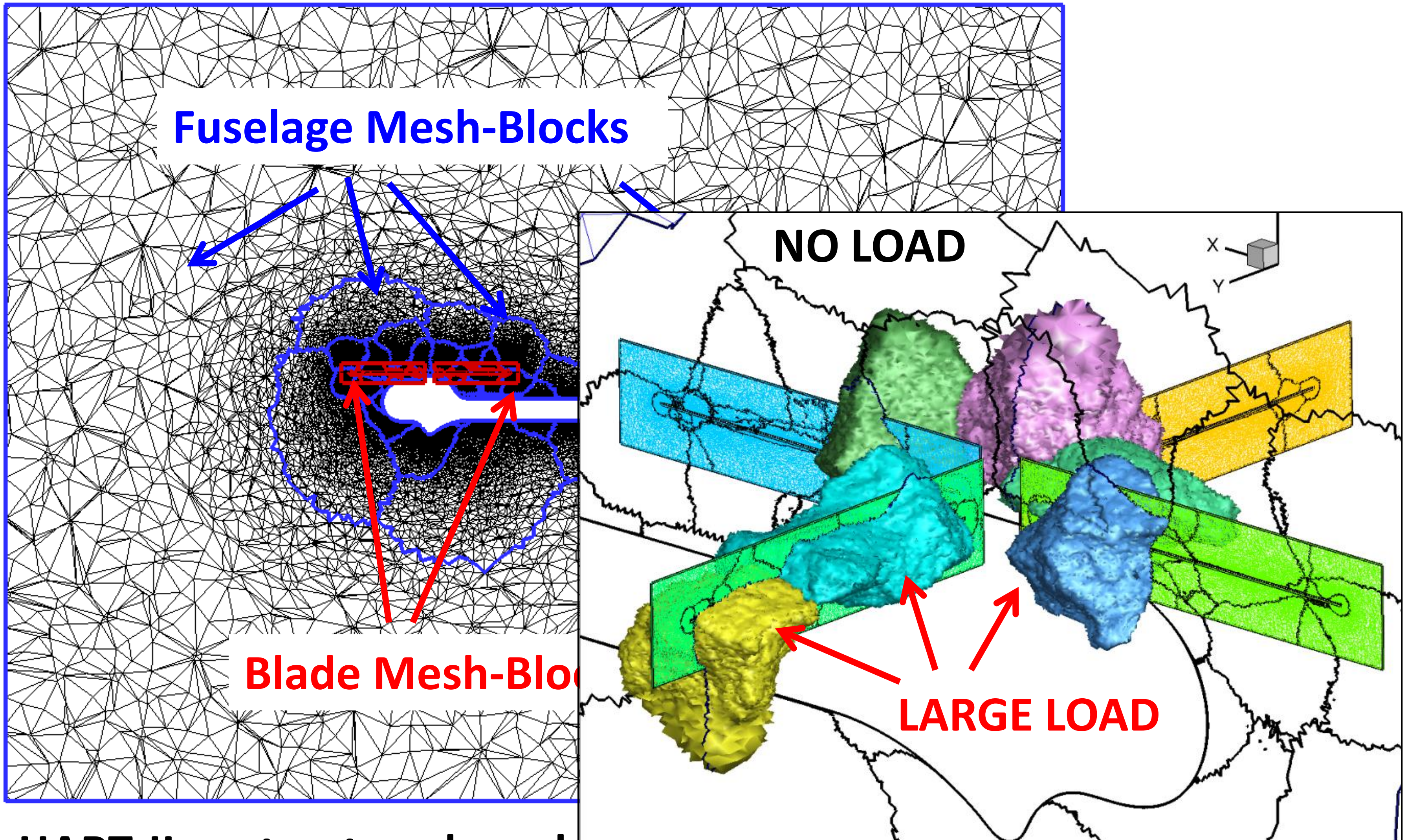
Line-walk search algo:
Move from cell to cell along a line using cell connectivity

Complex geometry of partition boundary

Multiple exit/re-entry possible

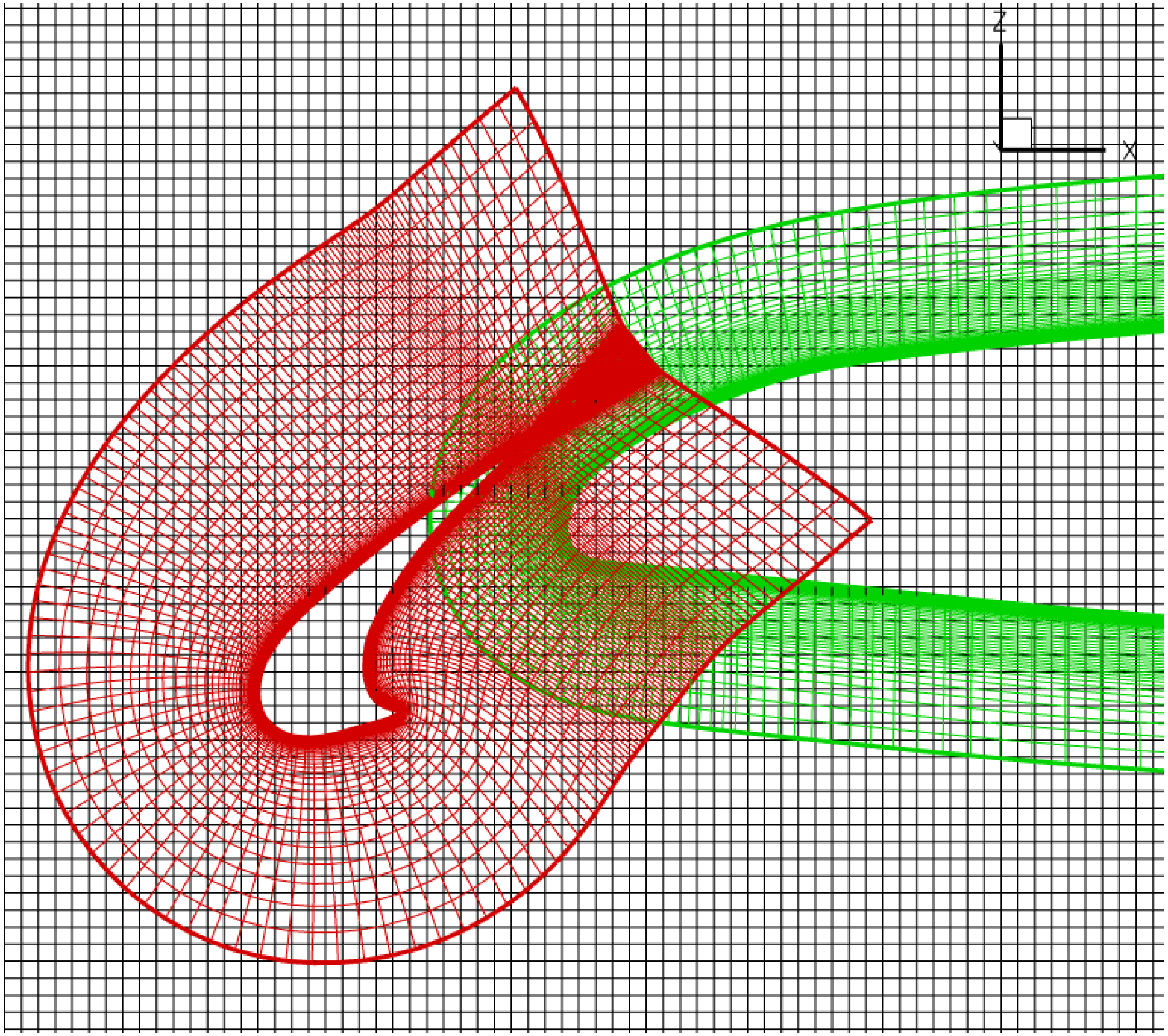


Load imbalance problem



HART-II unstructured mesh system :
1 fuselage, 4 blades, 260 mesh-blocks

Point Types Definition



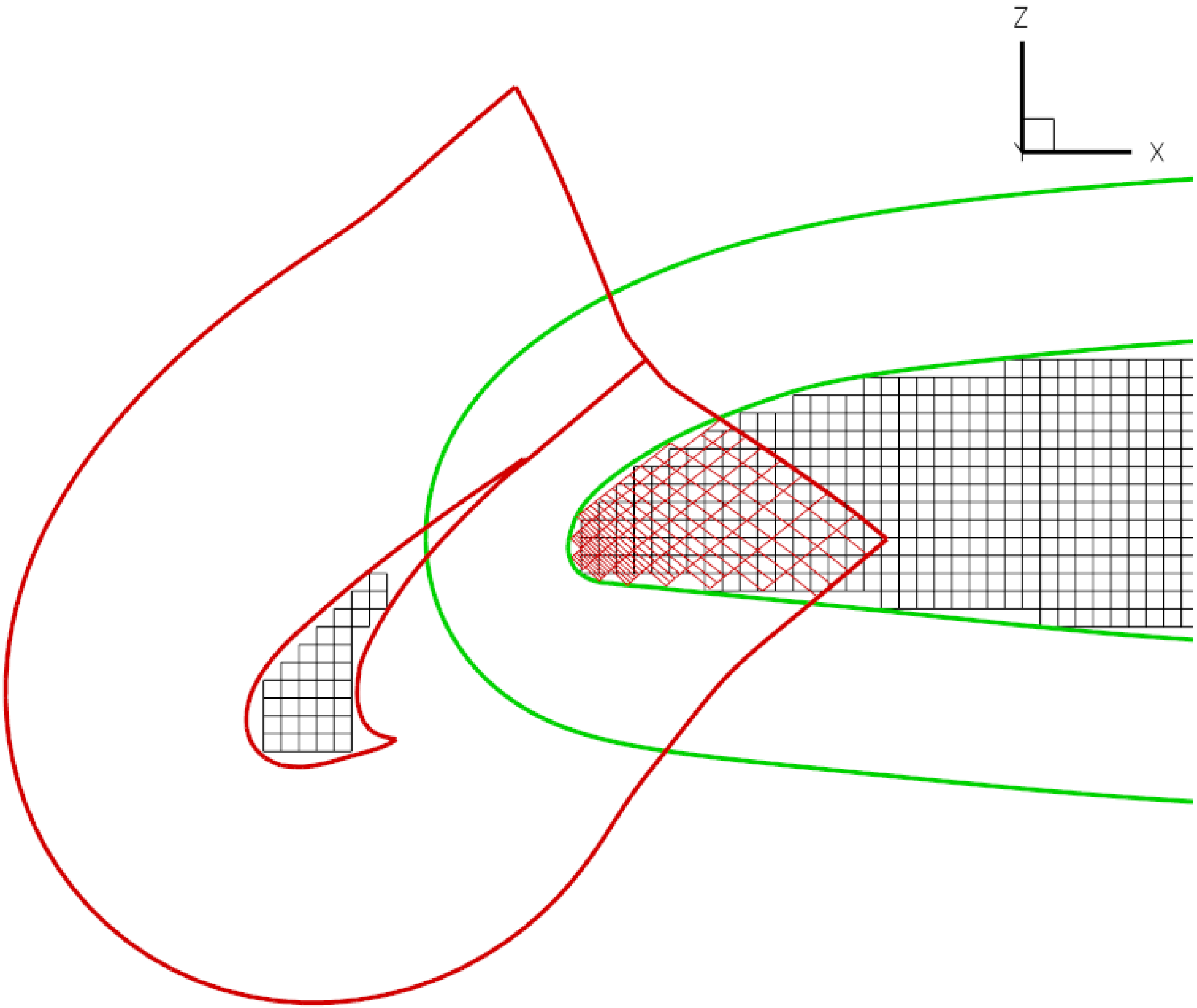
Overlapping mesh system:

OGA procedure attempts to find donor cells for all mesh points (query points)

Donors are selected if they have better resolution capacity

Resolution capacity : Heuristic parameter that quantifies solution quality (Cell volume is used now for donor cells and averaged cell volume for grid nodes)

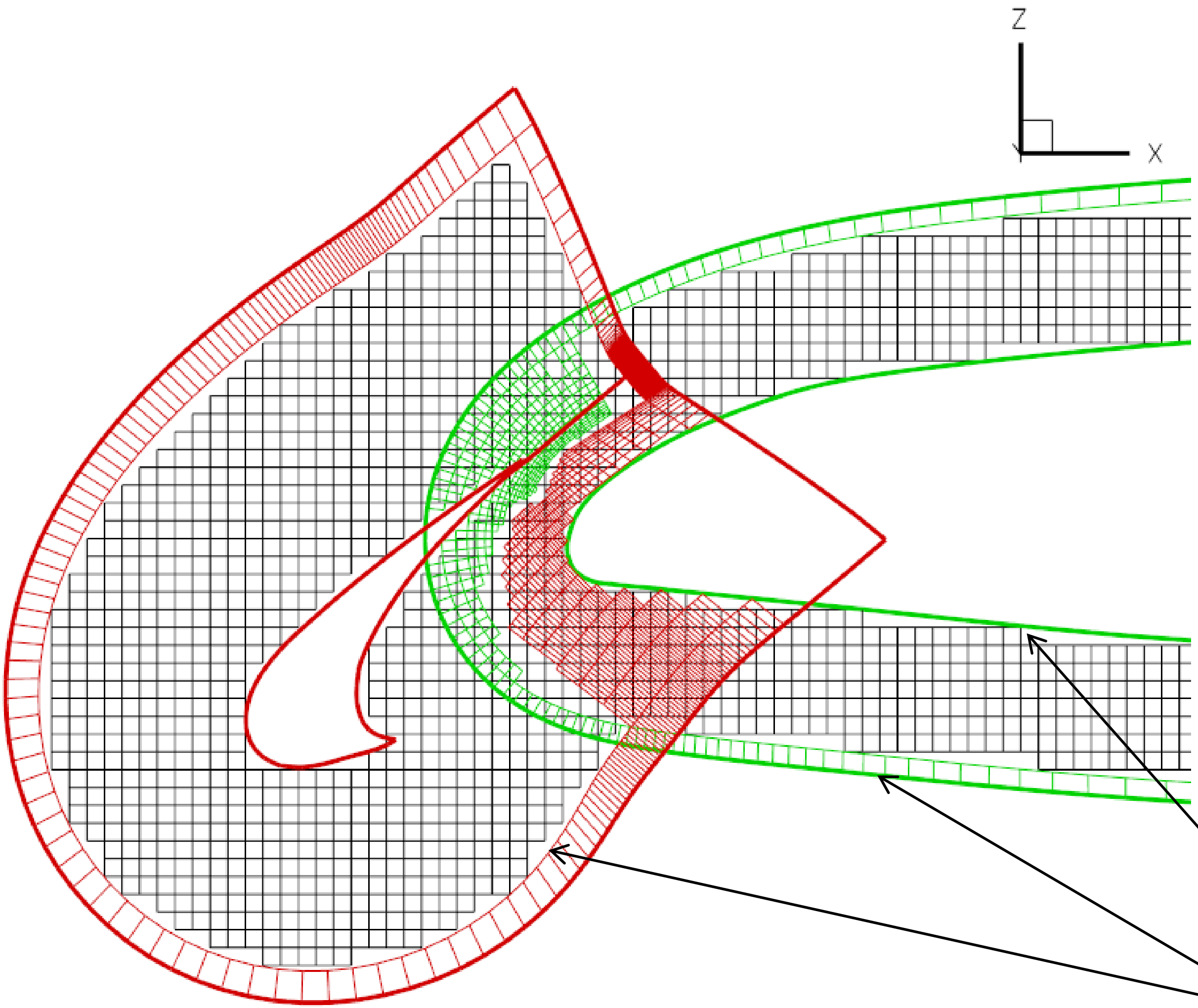
Point Types: hole points



Hole points:

Mesh points that are inside a solid wall

Point Types: receptor points



Receptor Points:

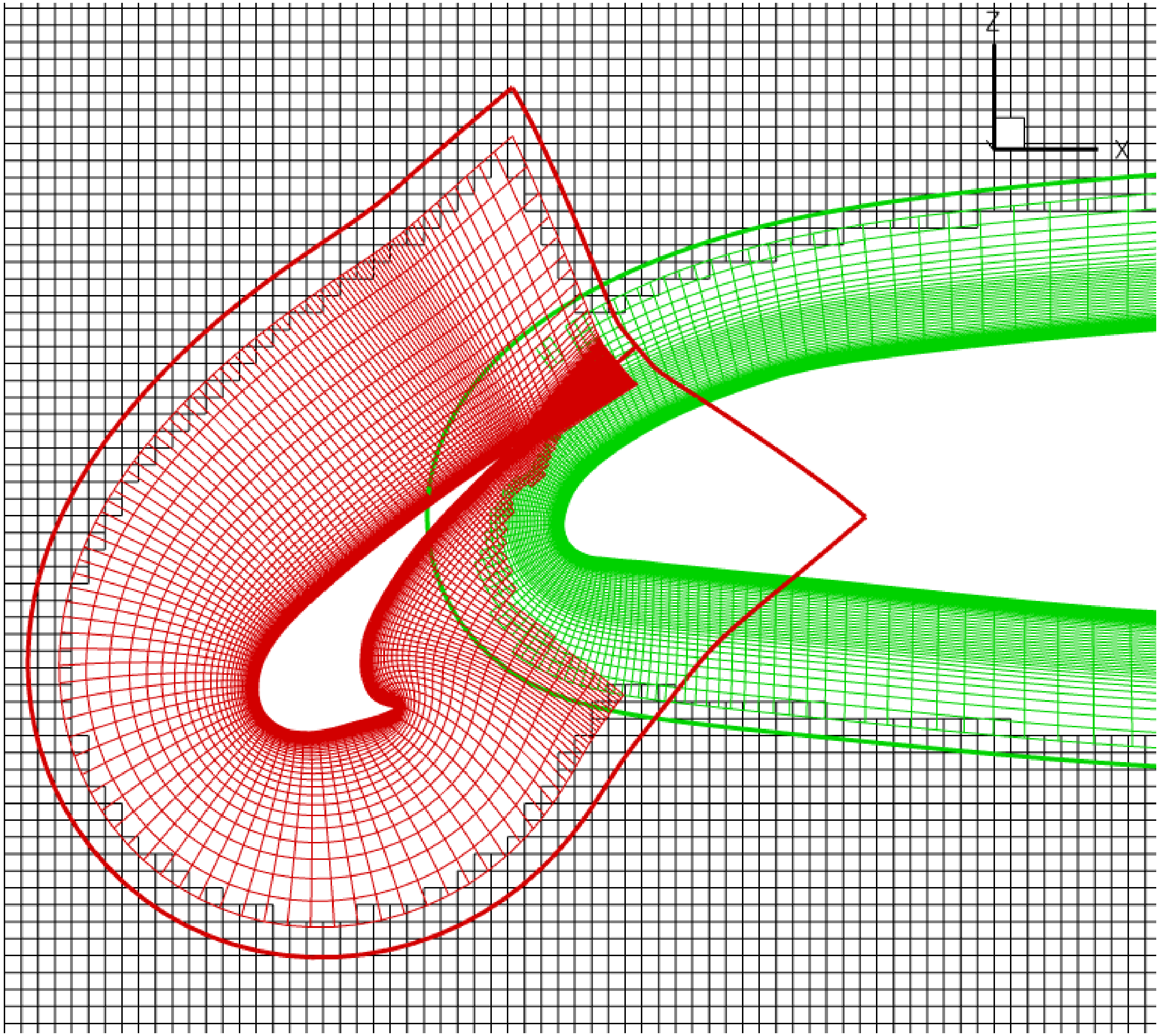
Mesh points that could find donor cells of better resolution capacity

flow solution will be interpolated to these points

Some points are mandatory receptors:

- Neighbors of hole points
- Neighbors of outer boundary

Point Types: Field Point



<u>Field points:</u>

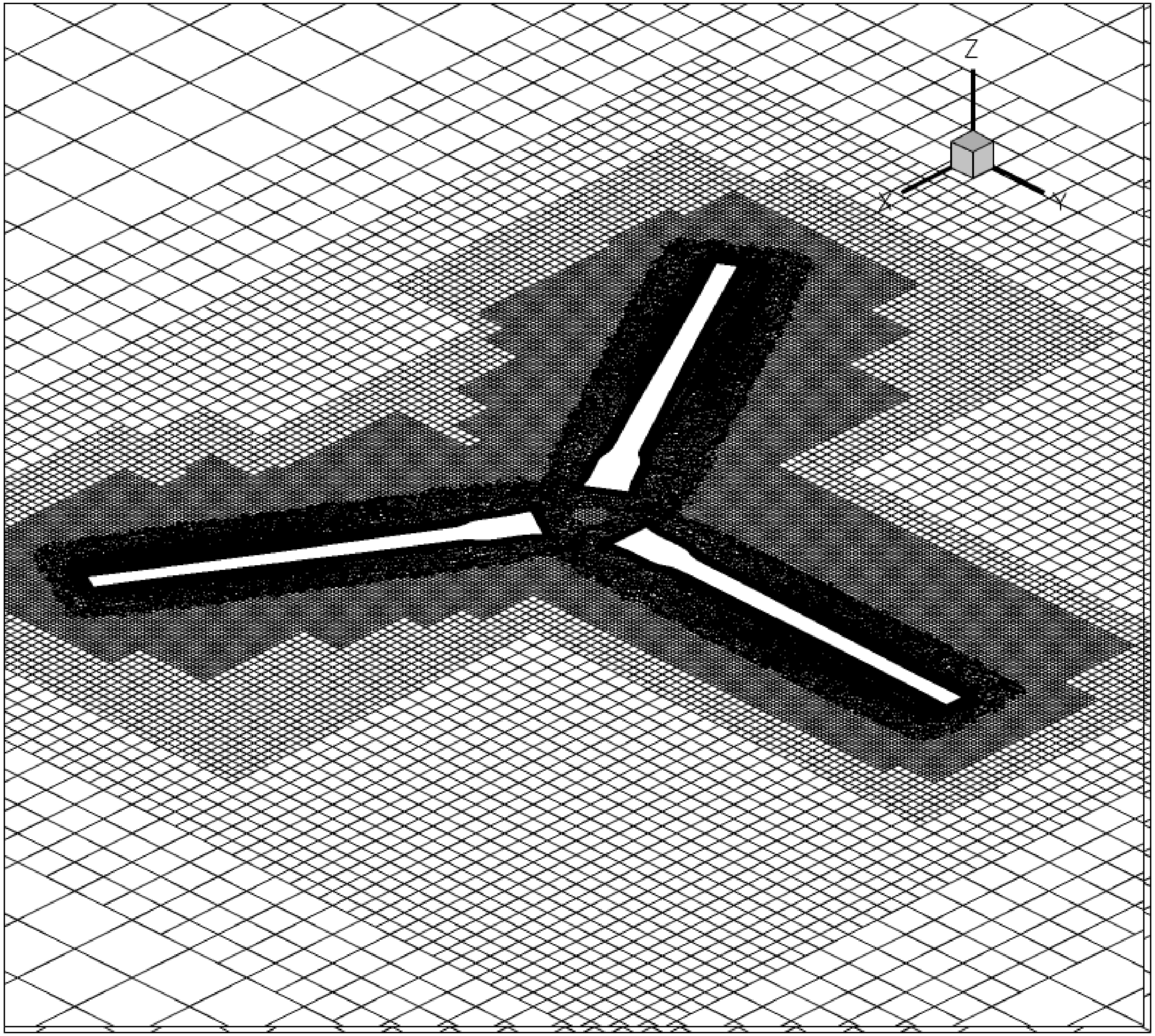
Mesh points were flow variables are being solved

Point where resolution is best, and which is neither hole point nor mandatory receptor.

Automated procedure to identify point type

→ minimal mesh overlap

Staged execution



Automated off-body mesh

Off
ge
an
(af
connectivity)
connectivity)

Near-body and
off-body grids
(fringes and holes
blanked out)

Overview of Presentation

1. Point Localization methods
   - EIM (Exact Inverse Maps): uses Cartesian auxiliary grids and inverse maps
   - ADT (Alternating Digital Tree): uses binary tree

2. Load re-balance Algorithm

3. Results
   - Timing and accuracy comparison between EIM and ADT (HART-II)
   - Scalability comparison with and without load re-balance (HART-II and WPS)
   - UH-60 forward flight CFD/CSD coupling

Overview of OGA method

Hole profiling: Create approximate representation of each solid body

Query Point Identification: For each mesh-block, list of points (from other MB) with potential overlap = QP

Mesh-block profiling: Pre-process mesh-block

Donor search: Identify donor cells for each QP

EIM:
- Create Auxiliary grids and exact inverse maps
- Line-walk search algo, constrained to small volume.

ADT:
- Cells organized in binary tree
- Exhaustive search on reduced set of potential donor cells

Point type assignment: Determine type of each query point (hole, receptor, of field point)

Interpolation: For receptor points, compute interpolation weights

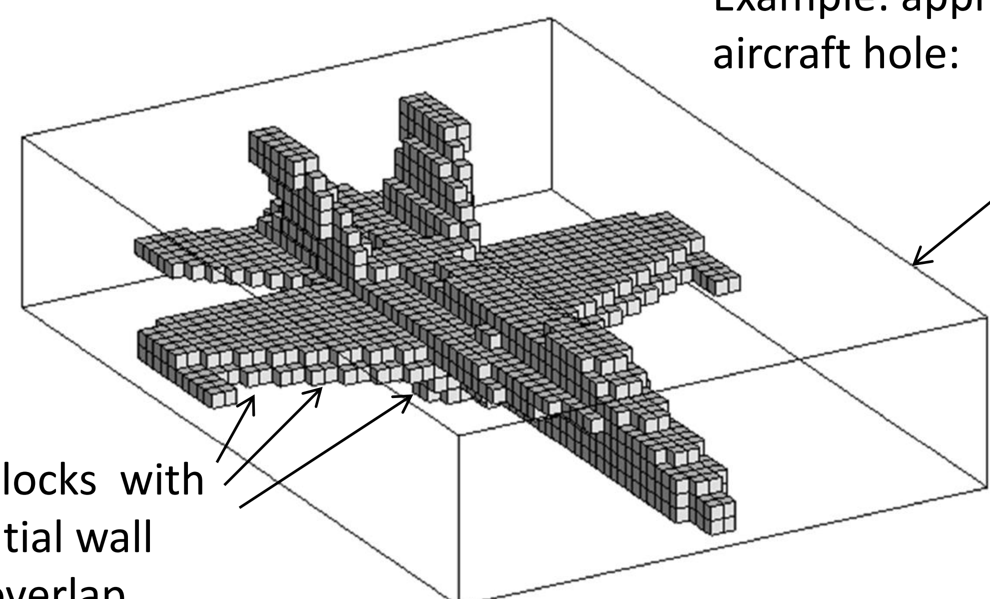
Hole Profiling

Goal: Create approximate representation of each solid body using a Cartesian auxiliary grid to facilitate identification of hole points after donor search step:

Hole points are points in approximate hole representation **AND** with no donor from hole mesh

(true only if approximate hole representation is close enough to the actual body wall: does not include any face of the outer mesh boundary).

Example: approximate representation of aircraft hole:



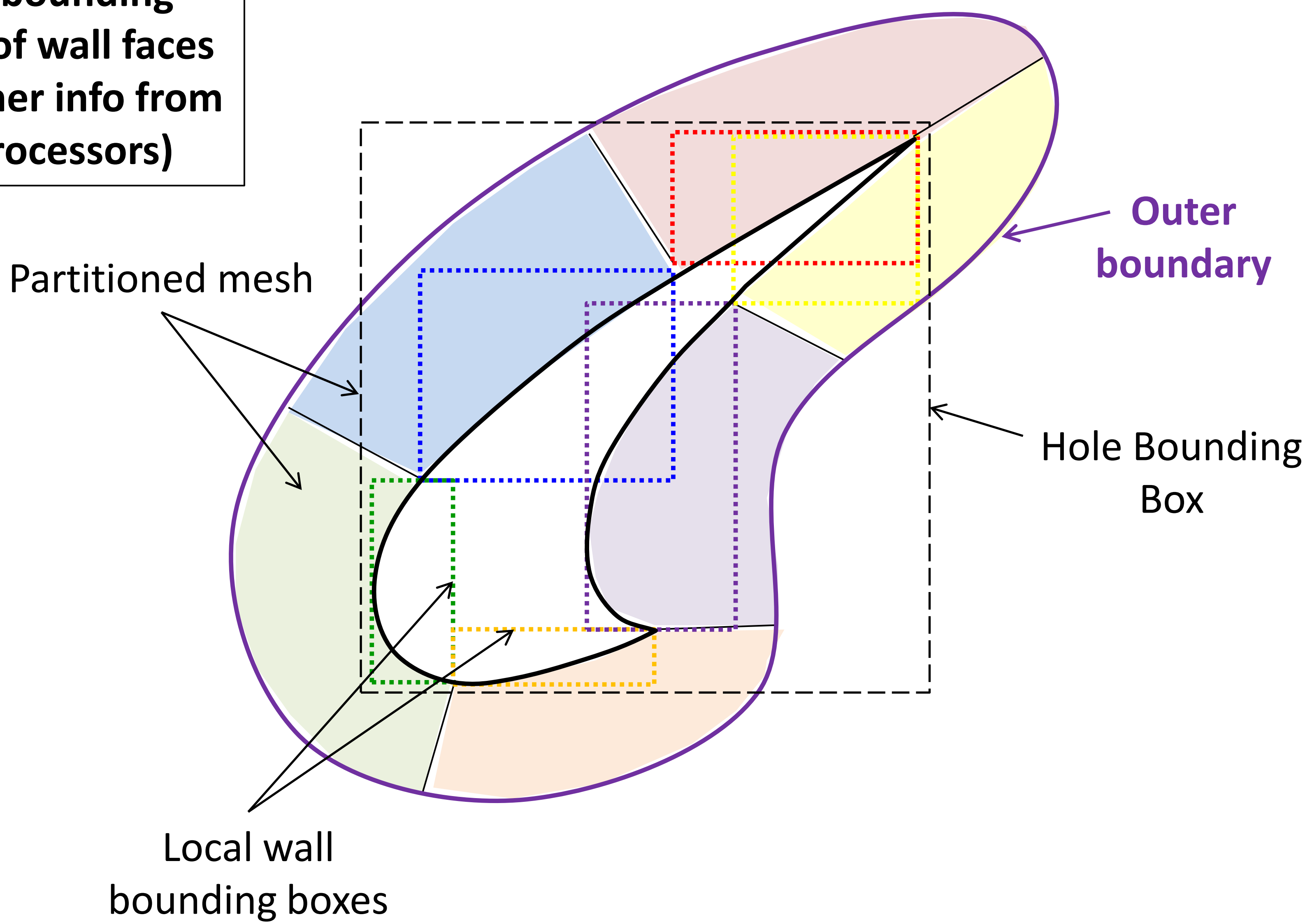
Cartesian auxiliary grid (AG):
- Bounding box of aircraft
- Equal size cells (sub-blocks)

Sub-blocks with potential wall face overlap

Advantage of Cartesian AG:
- Compact representation
- Very efficient identification of containing sub-block

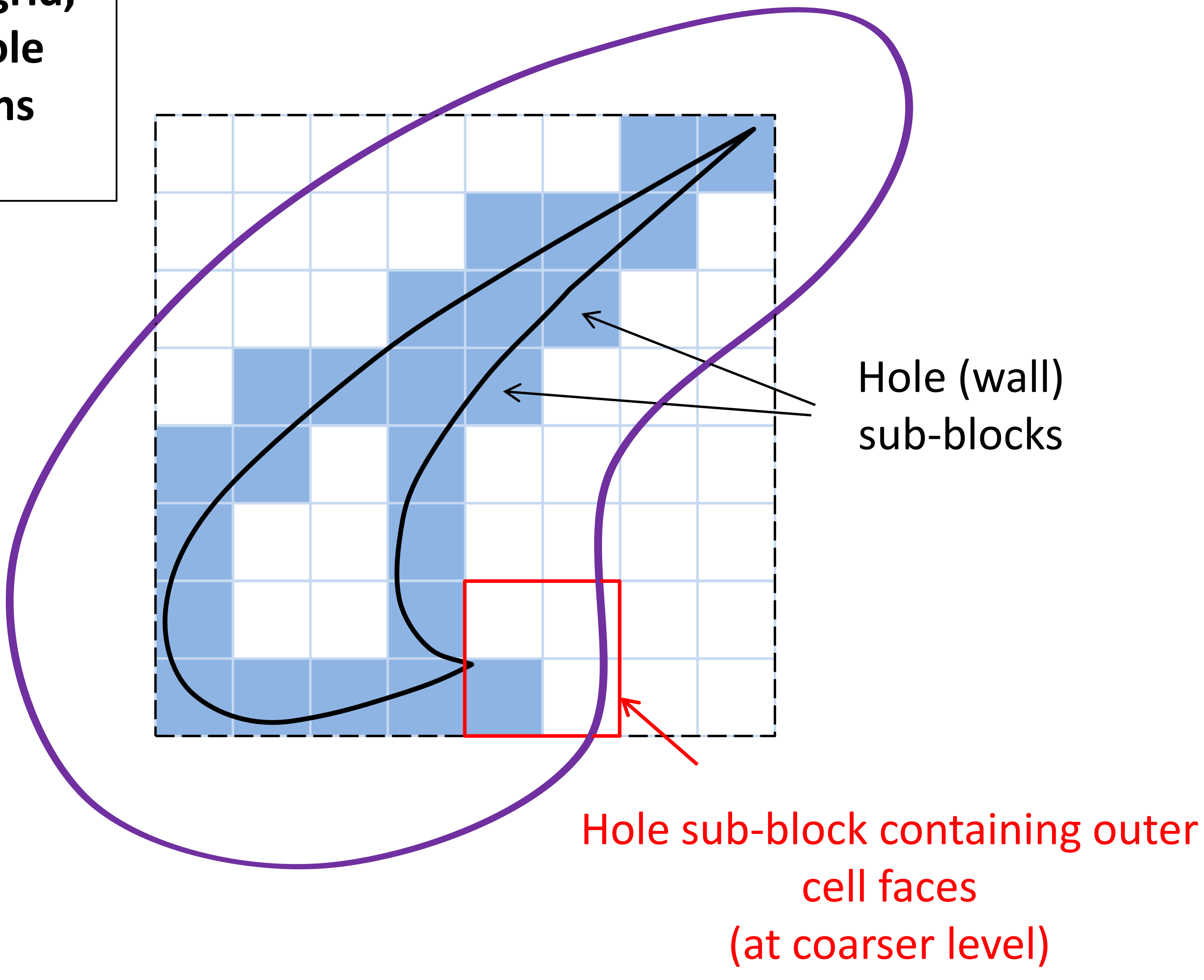
Hole Profiling : step 1

Find bounding box of wall faces (gather info from all processors)



Hole Profiling : step 2

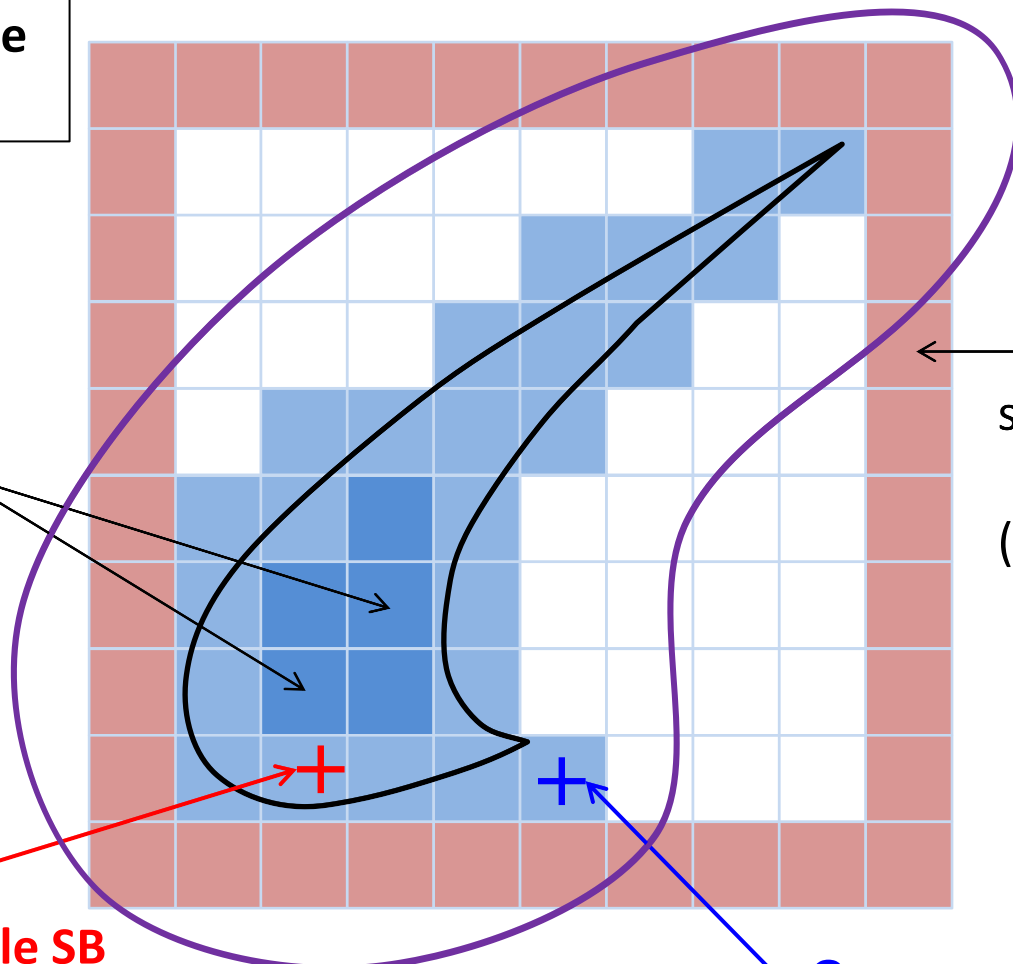
Create auxiliary grid, refine until no hole sub-block contains any outer face



Hole Profiling : step 3

Perform flood-fill to identify all hole sub-blocks

Hole (inner) sub-blocks



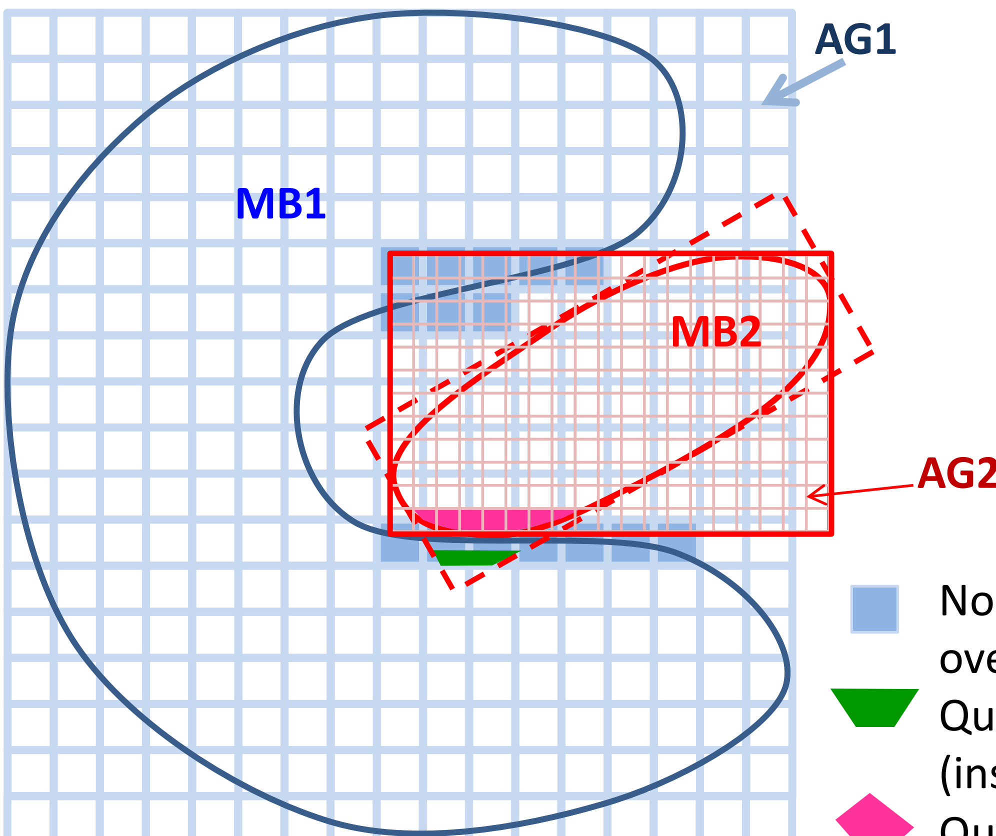
Outer layer of sub-blocks tagged as OUT (seeds of flood-fill algorithm)

Query point in hole SB AND no donor in hole mesh → HOLE POINT

Query point in hole SB, but donor cell exists in hole mesh → Not a Hole Point

Query Point Identification

For each mesh-block, find query points:
points in region of potential overlap
(for which donor cells need to be searched)
→ important to minimize number of QP



Use a combination of :

- oriented bounding box (OBB) overlap check and
- Cartesian auxiliary grids superimposed on each Mesh-block (non-empty sub-blocks tagged)

to obtain set of query points as small as possible

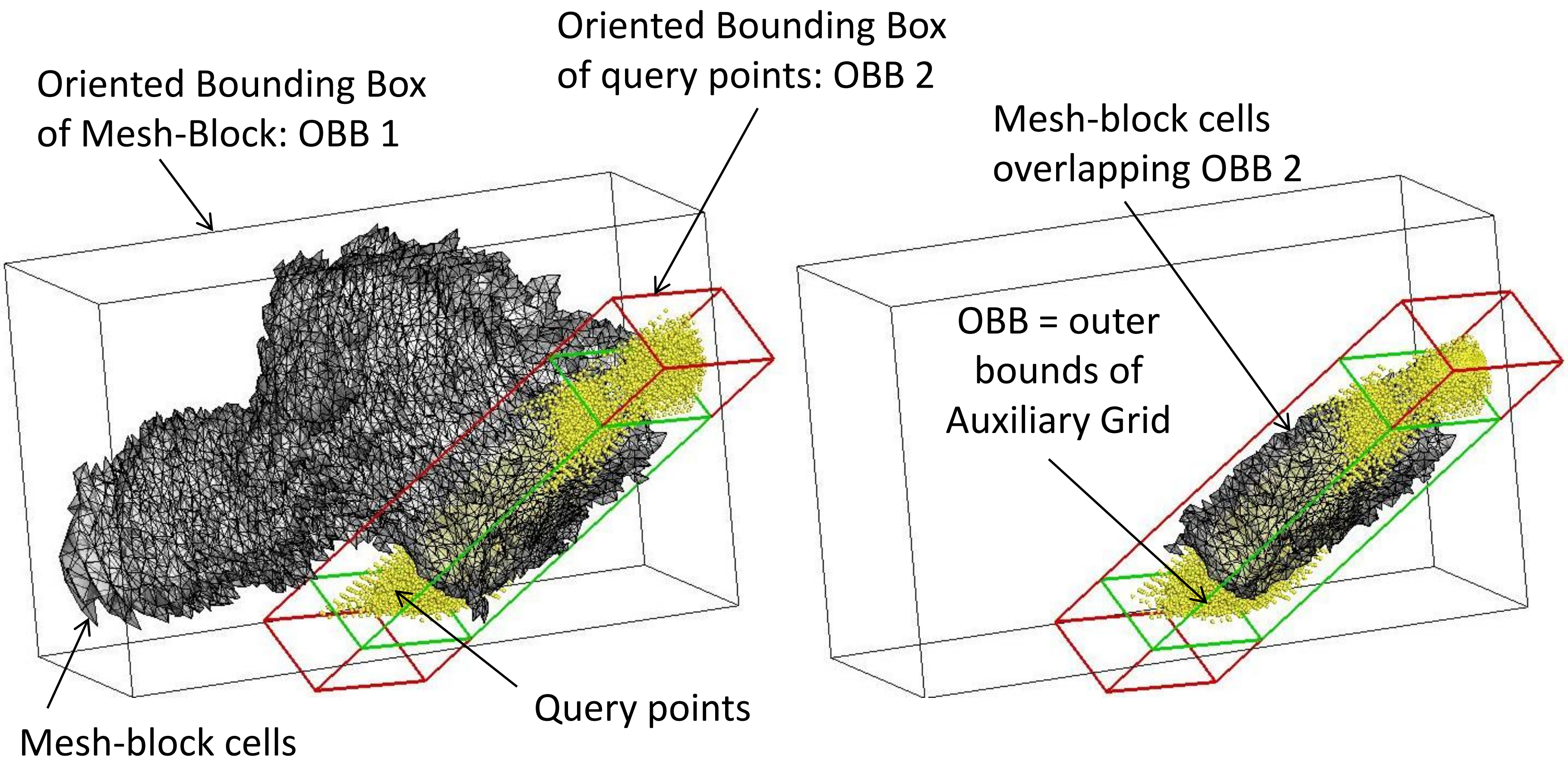
Non-empty sub-blocks of AG1 overlapping AABB of MB2

Query points for MB2 (inside OBB of MB2)

Query points for MB1

Query Point Identification

Only cells overlapping Query Points need to be pre-processed in next step (Mesh-Block profiling):



Mesh-Block Profiling (EIM)

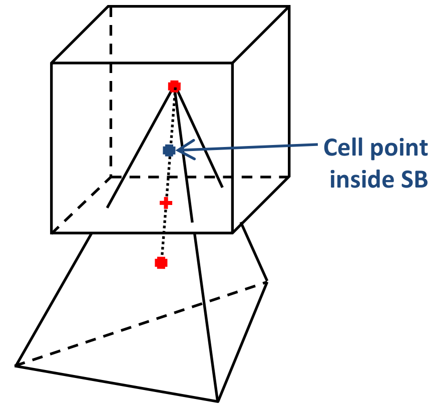
Create Cartesian Auxiliary Grid around cells and identify, for each sub-block, at least a cell point:

- **cell centers whenever possible**
- **any cell point otherwise**

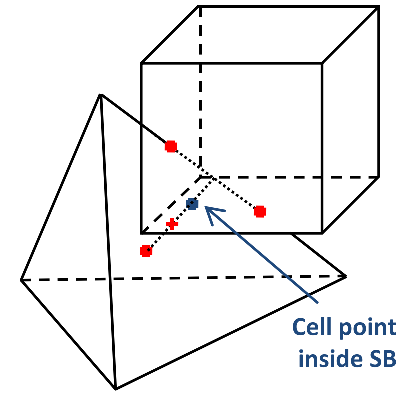
This point will serve as the starting point of the line search during donor search

Exact Inverse Map:
only sub-blocks with no overlap with mesh-block cells do not store any point.

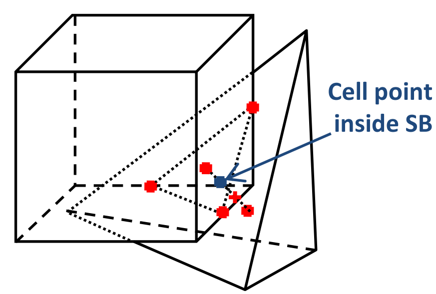
Another map is also created to store, for each sub-block, all boundary faces contained (based on BB overlap)



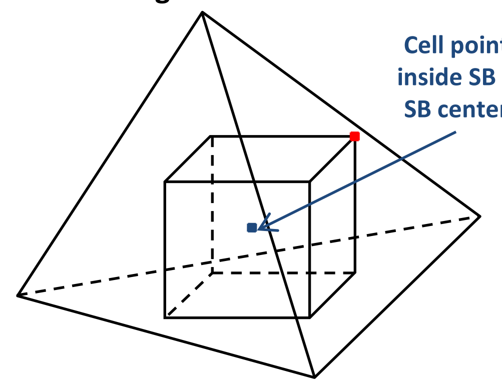
Cell Vertex inside Sub-Block



Cell Edge intersects Sub-Block face



Sub-Block edge intersects cell face



Sub-block is entirely inside cell

Sub-block size determines efficiency of algorithm.
From empirical order analysis, near-optimal rule is:

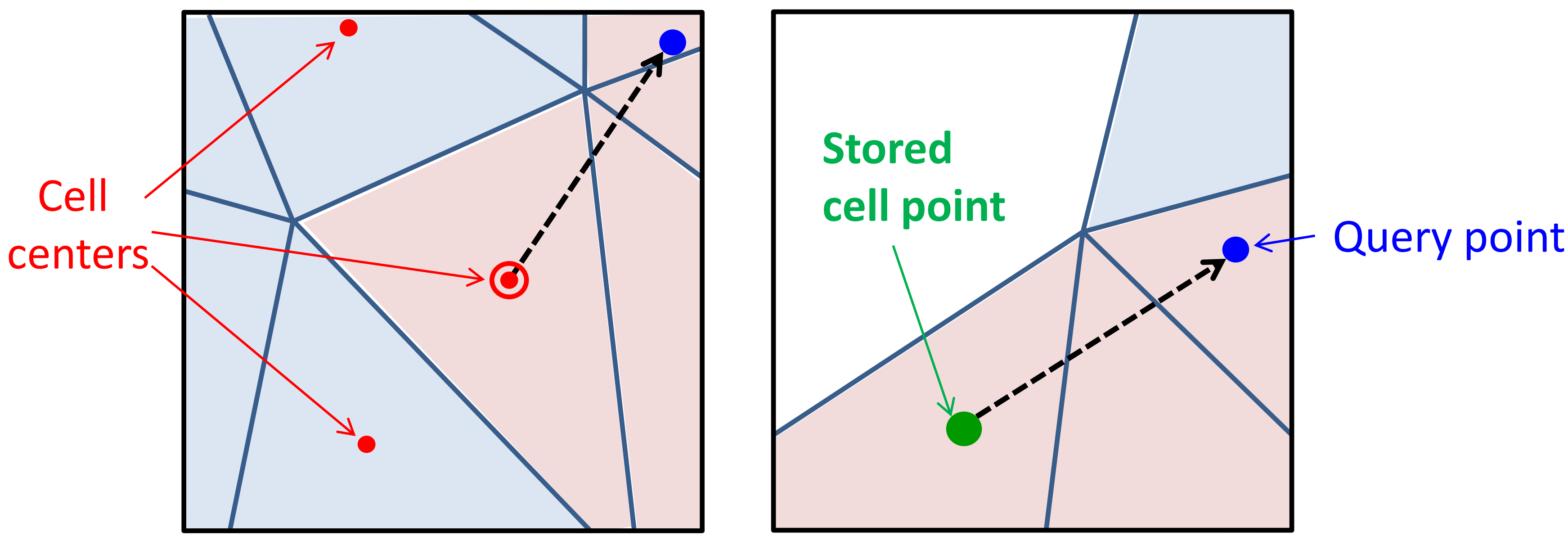
$$N_{SB} = 0.1\ {N_p}^{0.4}\ {N_c}^{0.6}$$

Donor Search (EIM)

Problem : identify containing mesh-block cell for each Query Point

Line-walk search algorithm:
Form a line from starting point to query point (inside known cell), walk from cell to cell along that line until line does not intersect any cell face (donor found) or a boundary face is crossed (QP possibly out, but must check for re-entry)

Cell centers in sub-block of QP | No cell centers in sub-block of QP



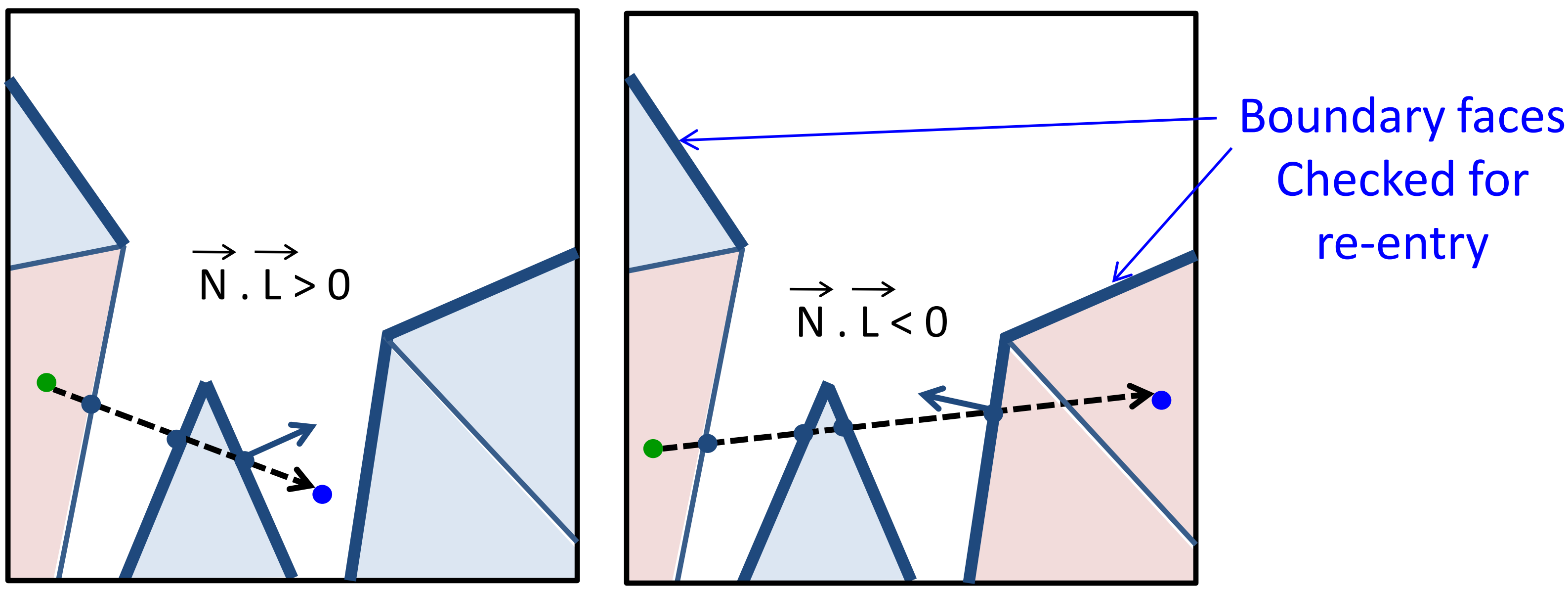
Both start point and query point are in a single sub-block of the AG
→ Entire line-search constrained to sub-block: easier to check for re-entry

Donor Search (EIM)

If the search-line crosses a boundary face, check other boundary faces in the sub-block for possible re-entry:
New intersection closest to QP: face normal points in same or different direction as search vector ?

Same → no donor exists

Different → a donor cell exists, search can resume from this cell.

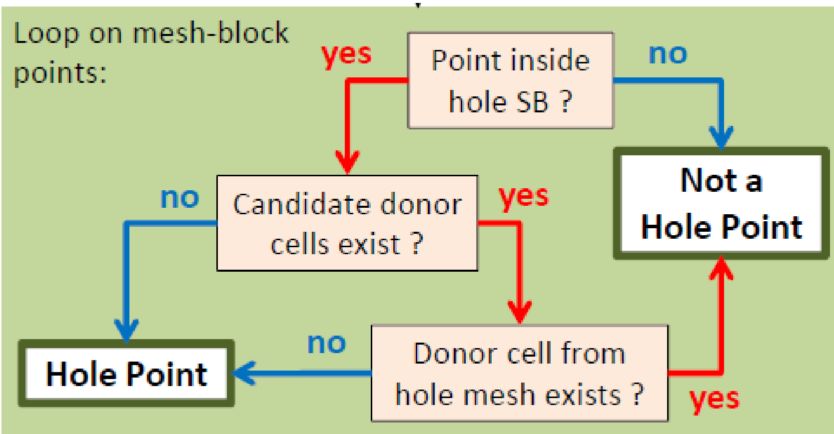


Robustness issues:

- Tolerance for determining face crossing
- Interpolation weight check
- Moving search-line if too close to vertex/edge /face

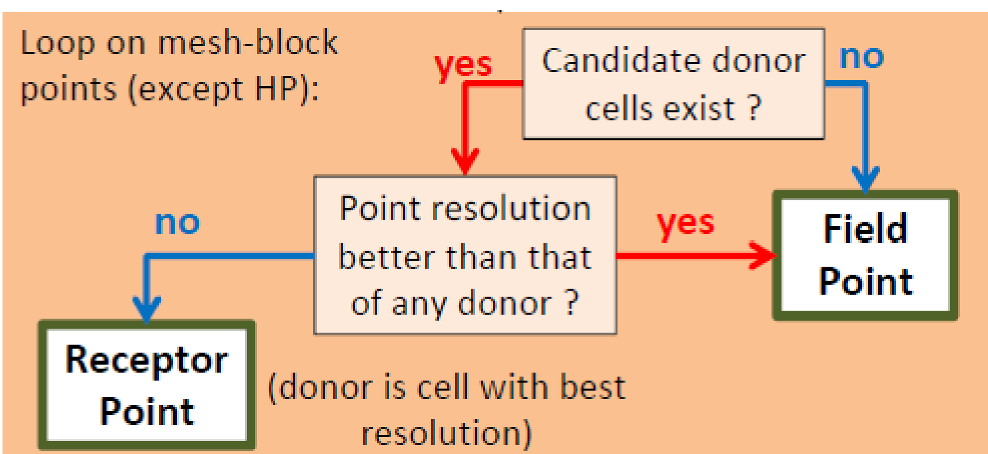
Point Type Assignment

1. IDENTIFY HOLE POINTS

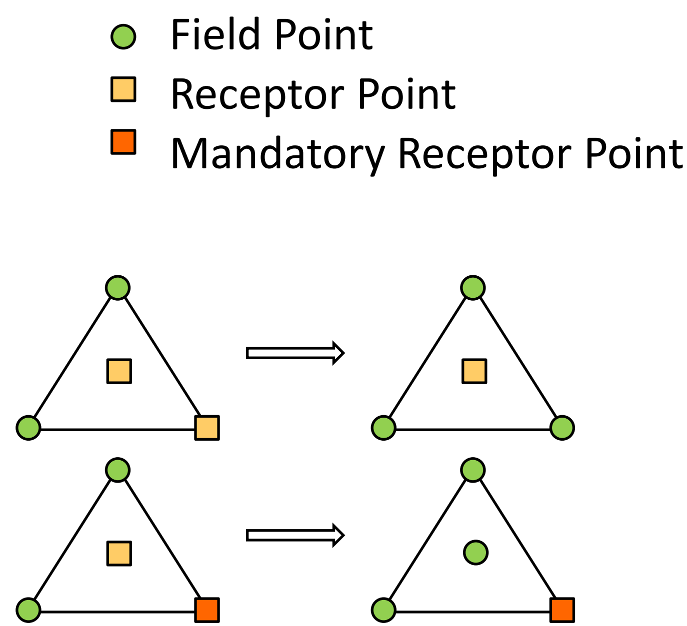


2. IDENTIFY FIELD/RECEPTOR POINTS

Identify first mandatory receptors



3. RESOLVE POINT TYPE CONFLICTS

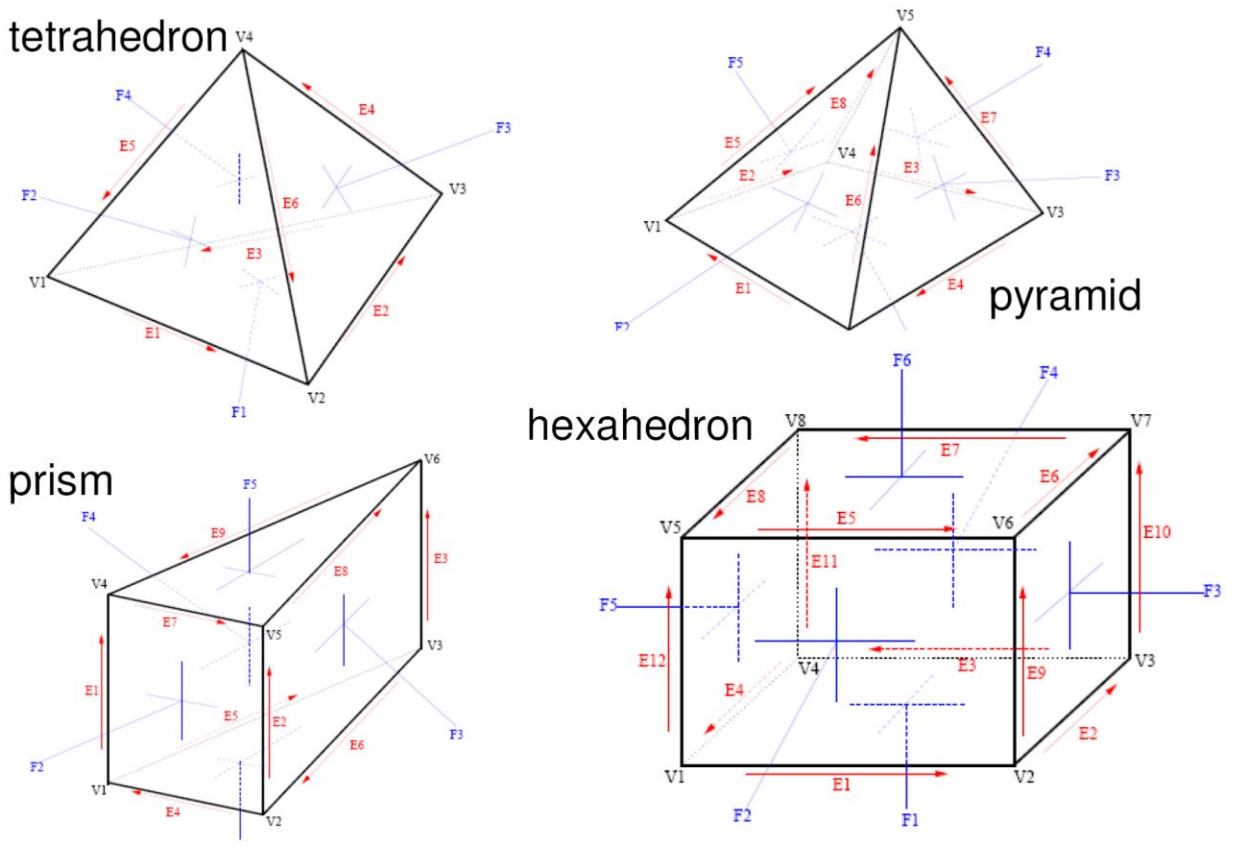


Donor cells should not have any node of “Receptor Point” type

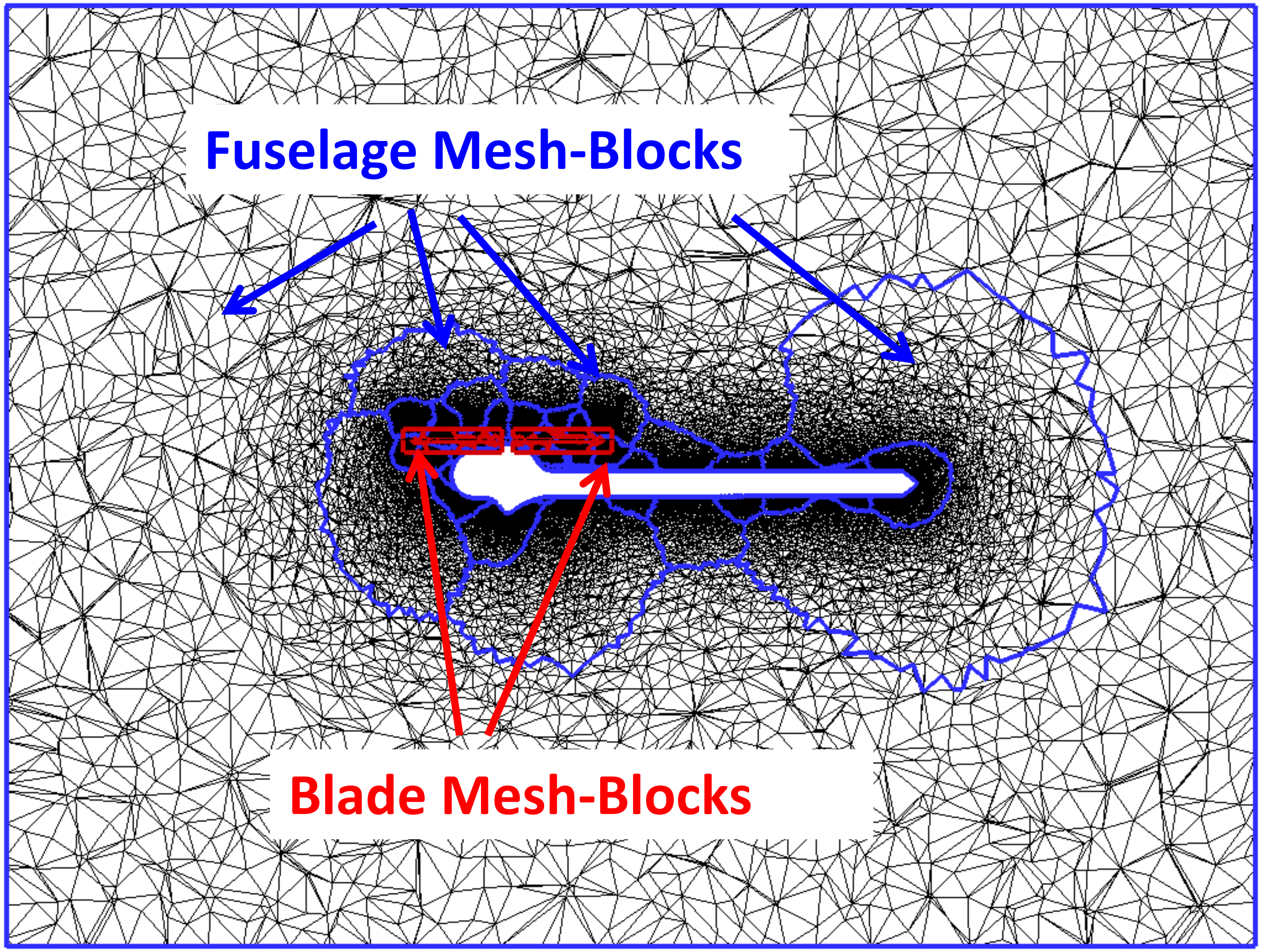
Interpolation

Receptor points: interpolation weights computed using Newton-Raphson procedure.

Supported cell types:

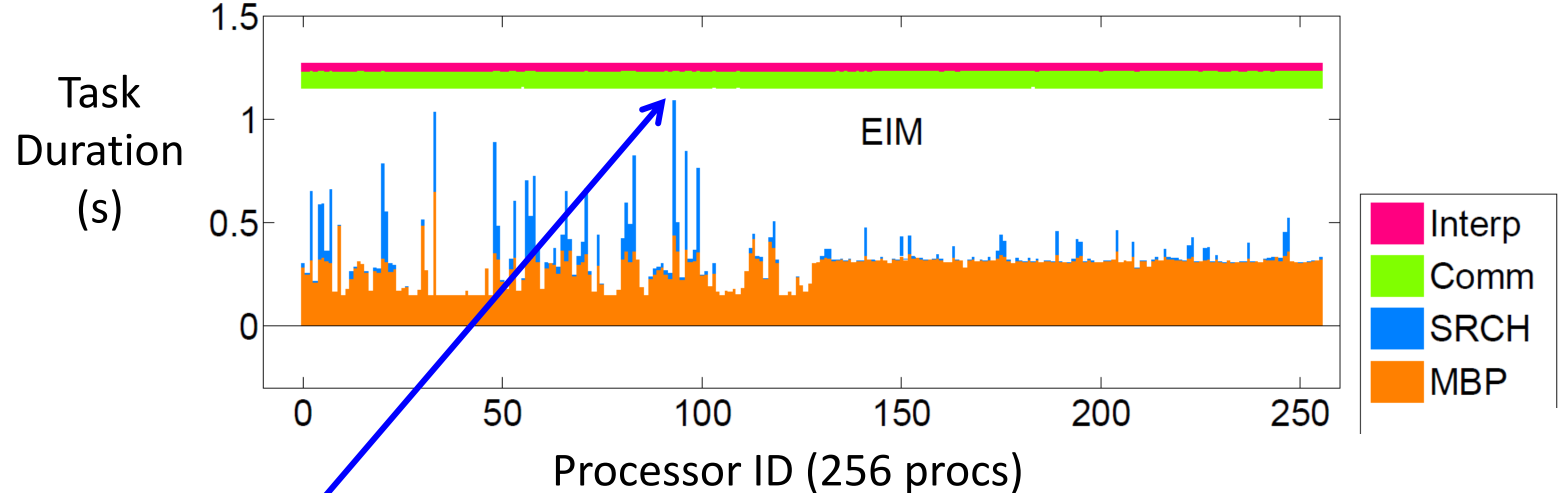


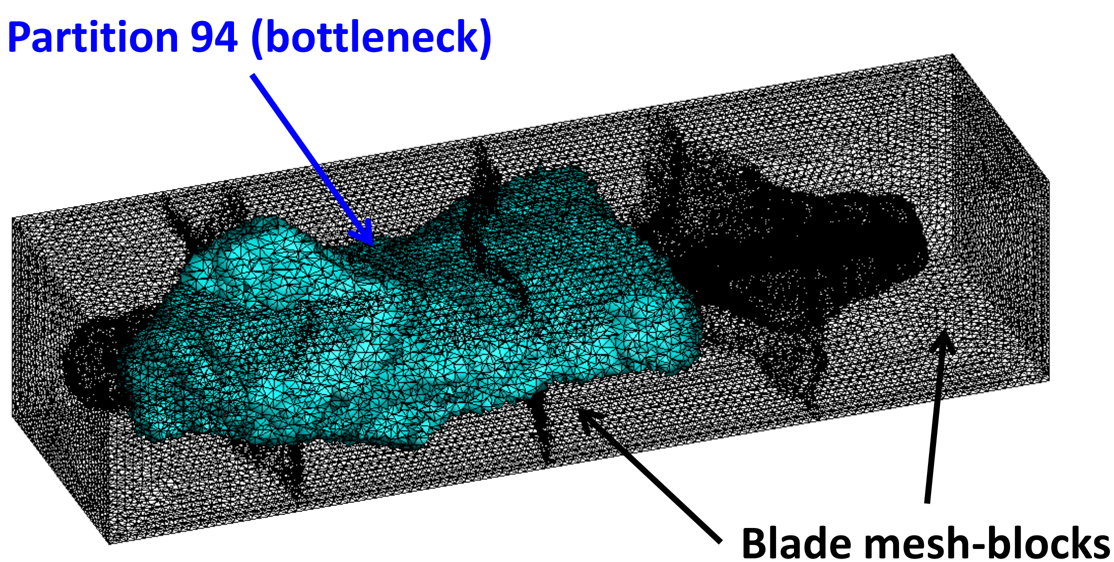
Load imbalance problem

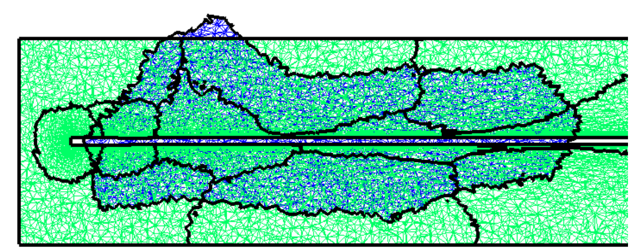


HART-II unstructured mesh system :
1 fuselage, 4 blades, 260 mesh-blocks

Load imbalance problem





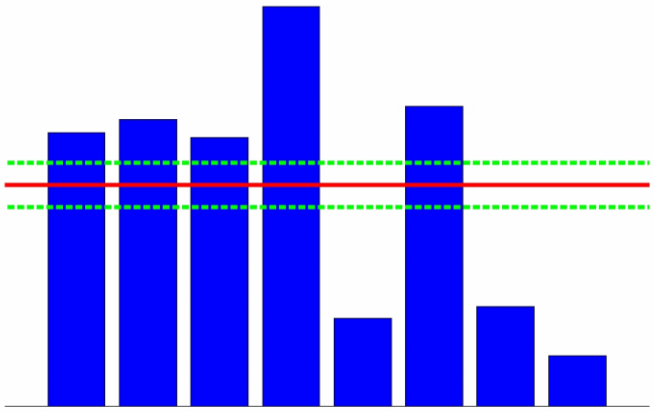


Sectional view

Load Re-balance Algorithm

Simple load re-balancing:

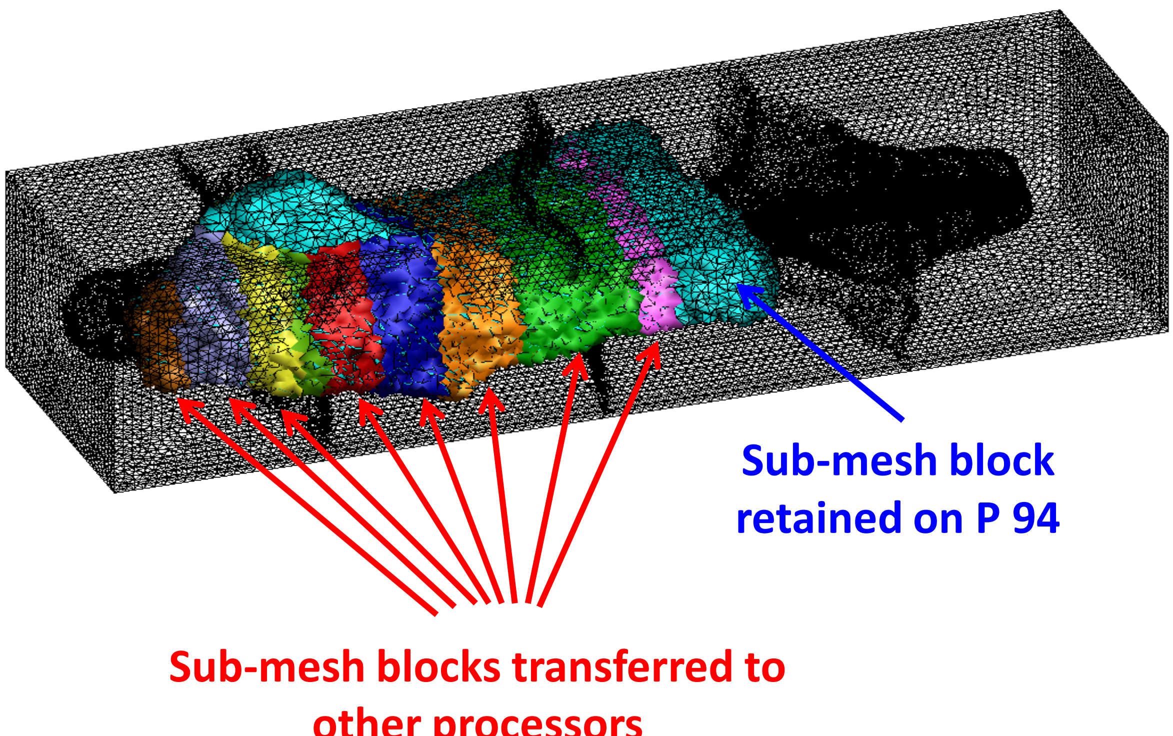
- ✓ Load per processor estimated: total OGA time
  target load = load average
- ✓ Most loaded processor donates to least loaded processor, until all are within 20% of target load
- ✓ Load assumed prop. to number of QP: if P1 needs to transfer x% of its load, it transfers x% of its Query Points.



Load Re-balance Algorithm

- ✓ QP to transfer are chosen by dividing overloaded mesh-block in the longest direction and using a Cartesian auxiliary grid to efficiently identify the required number of QP

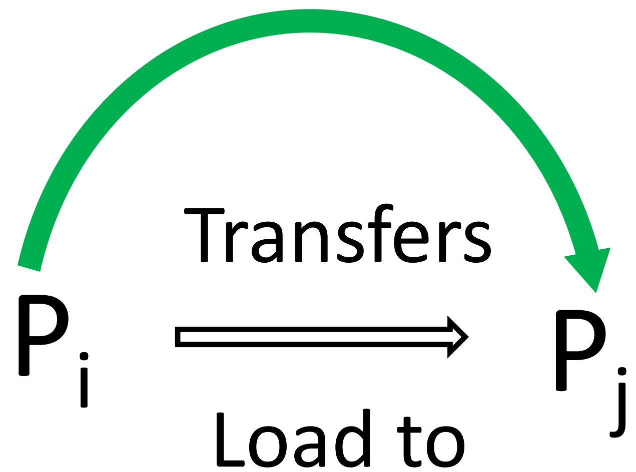
- ✓ Along with QP, overlapping cells information is also transferred. Currently, ADT method is used to perform load rebalance (less data required)



Adaptive Load Re-balance

- ✓ After initial load re-balance, load distribution still inadequate:
  - Duration of new communication tasks not accounted for
  - Assumption of load prop to nQP inaccurate

- ✓ Adaptive load re-balance: use current load measurements to correct previous load transfer matrices



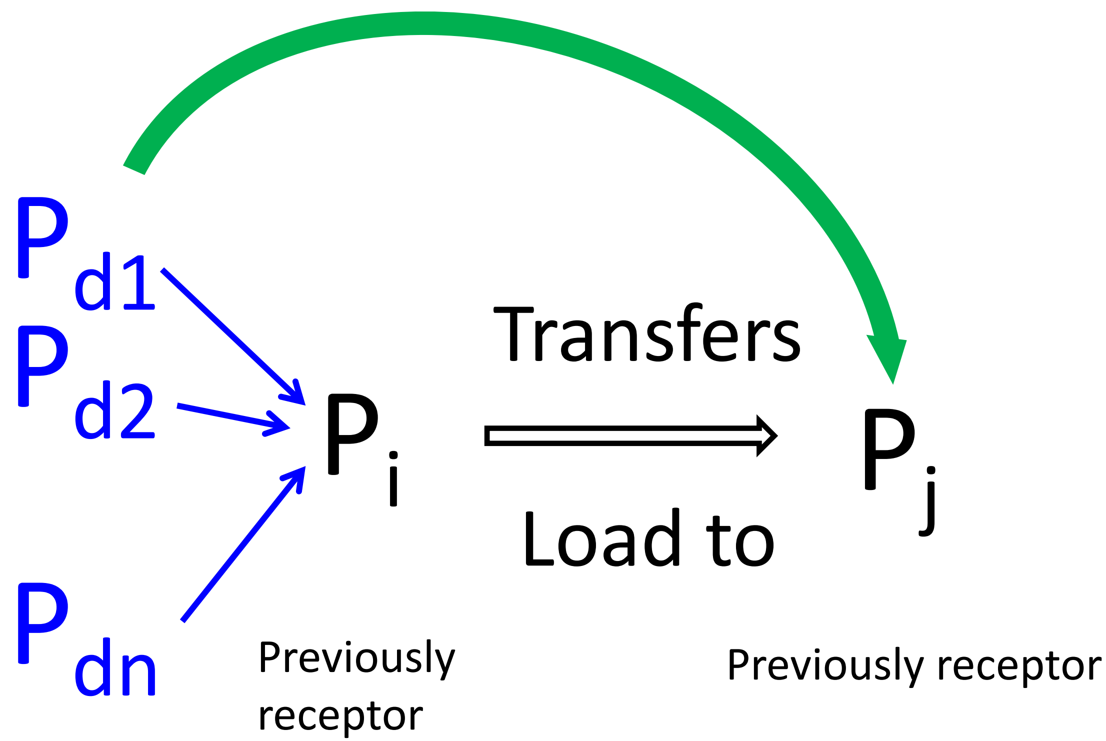
Previously donor

Previously receptor

Adaptive Load Re-balance

✓ After initial load re-balance, load distribution still inadequate:
- Duration of new communication and partitioning tasks not accounted
- Assumption of load prop to nQP inaccurate

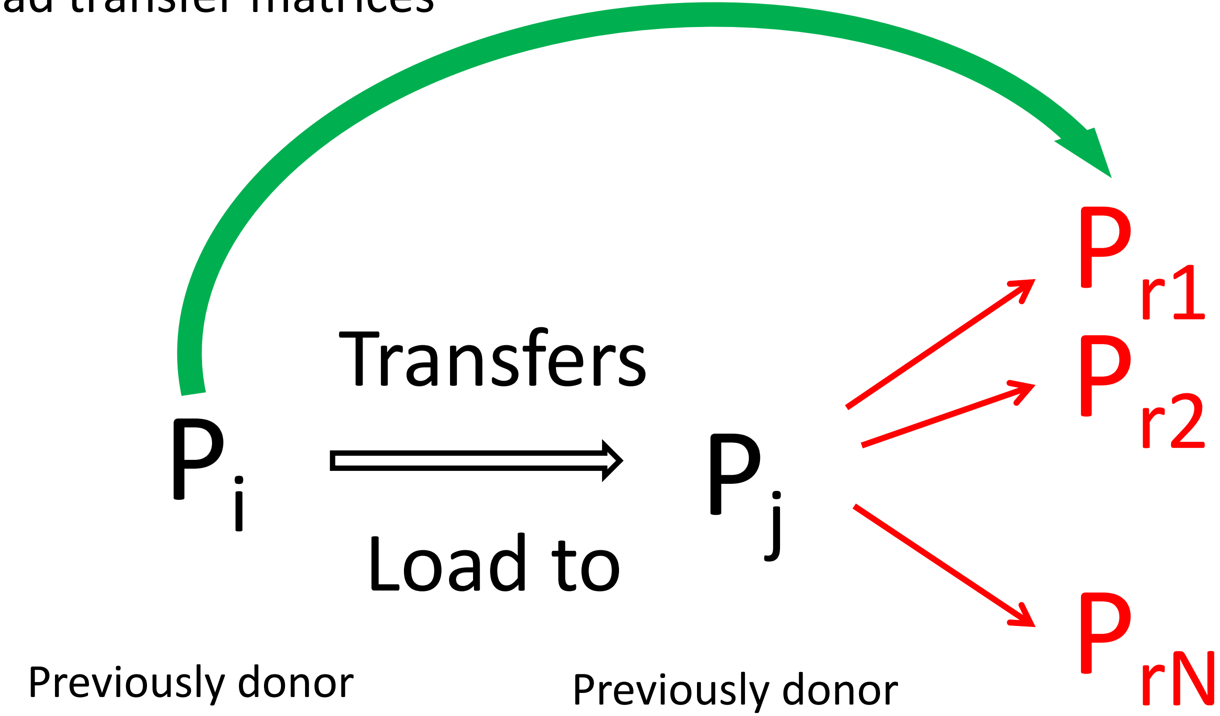
✓ Adaptive load re-balance: use current load measurements to correct previous load transfer matrices, include cost of new operations



Adaptive Load Re-balance

- ✓ After initial load re-balance, load distribution still inadequate:
  - Duration of new communication tasks not accounted for
  - Assumption of load prop to nQP inaccurate

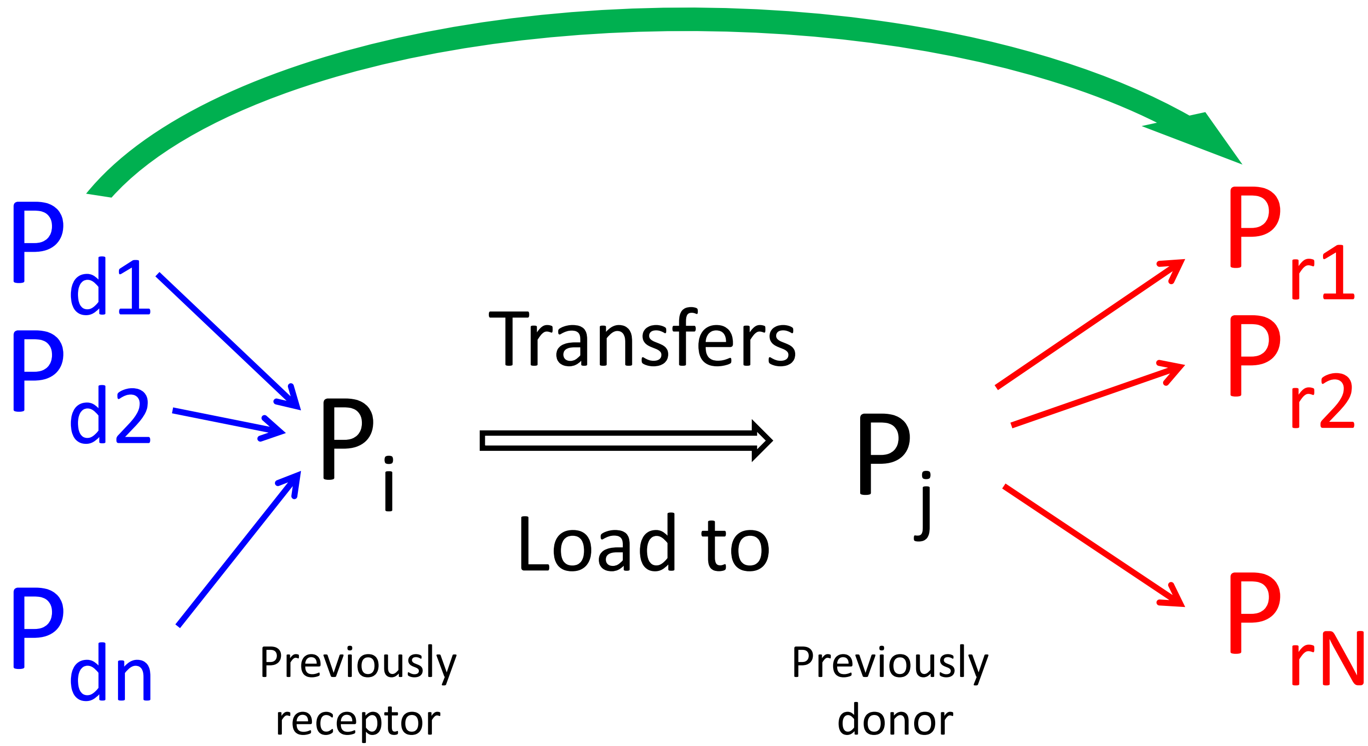
- ✓ Adaptive load re-balance: use current load measurements to correct previous load transfer matrices



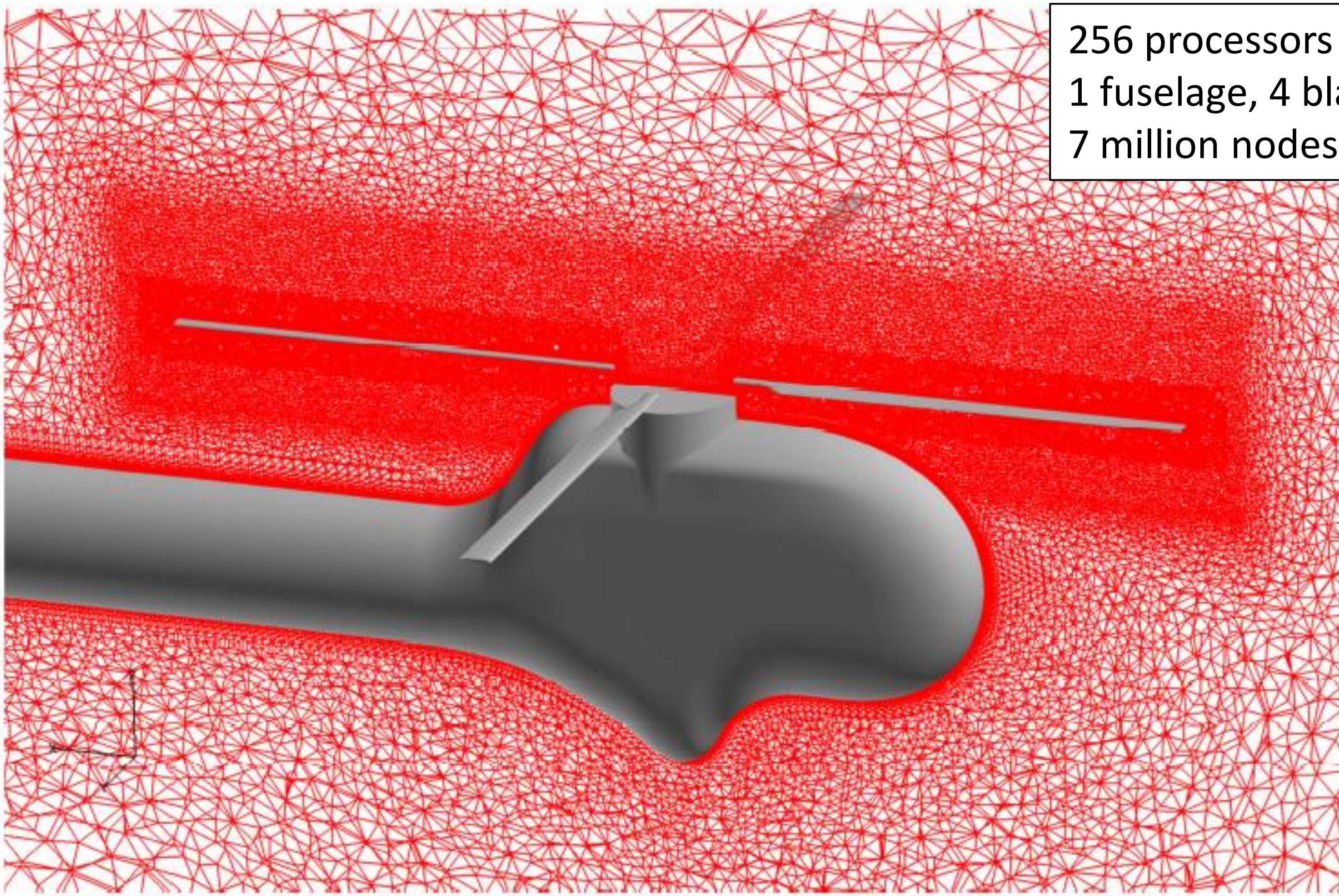
Adaptive Load Re-balance

✓ After initial load re-balance, load distribution still inadequate:
- Duration of new communication tasks not accounted for
- Assumption of load prop to nQP inaccurate

✓ Adaptive load re-balance: use current load measurements to correct previous load transfer matrices

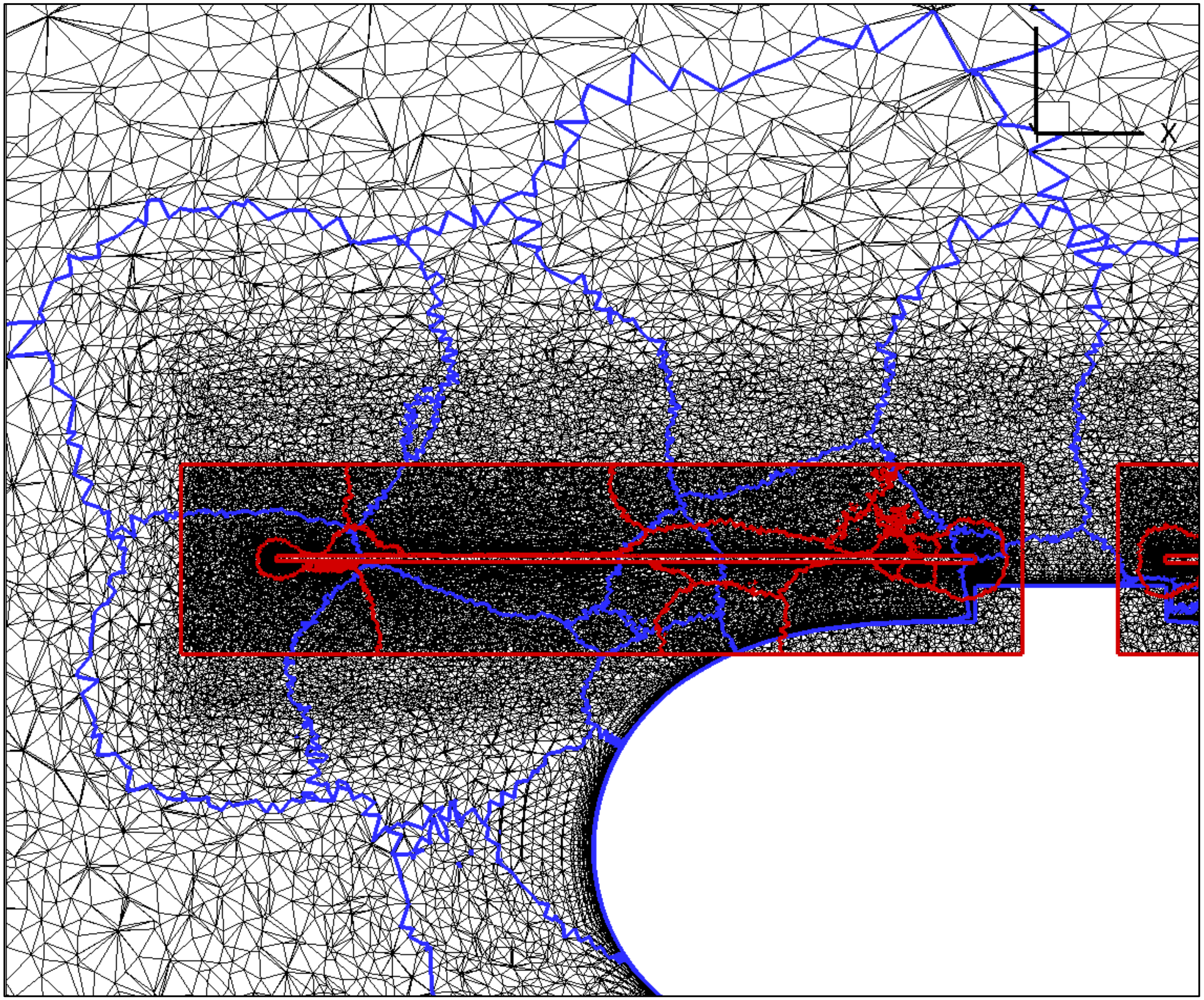


HART-II case

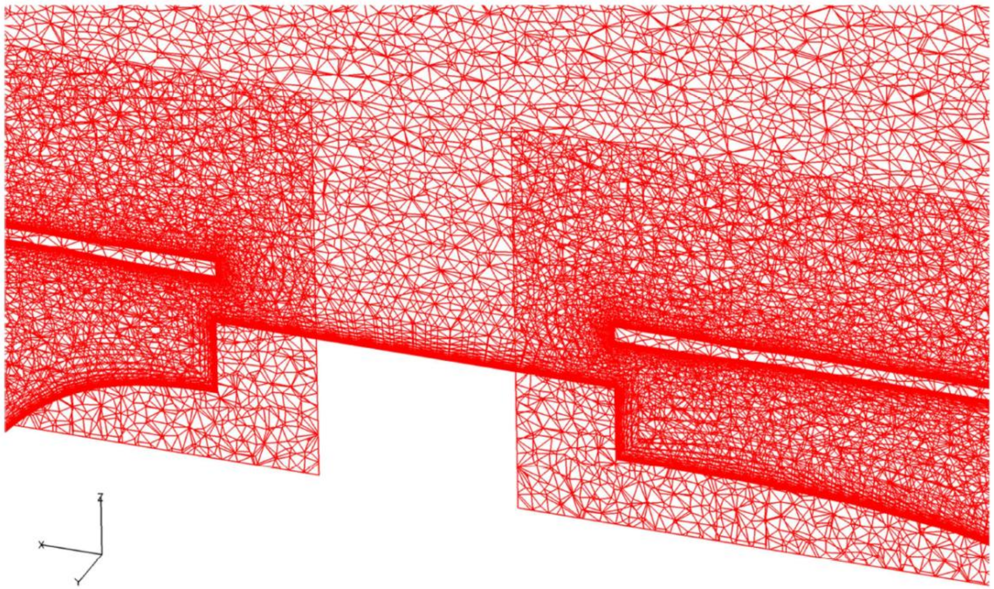


256 processors
1 fuselage, 4 blades
7 million nodes

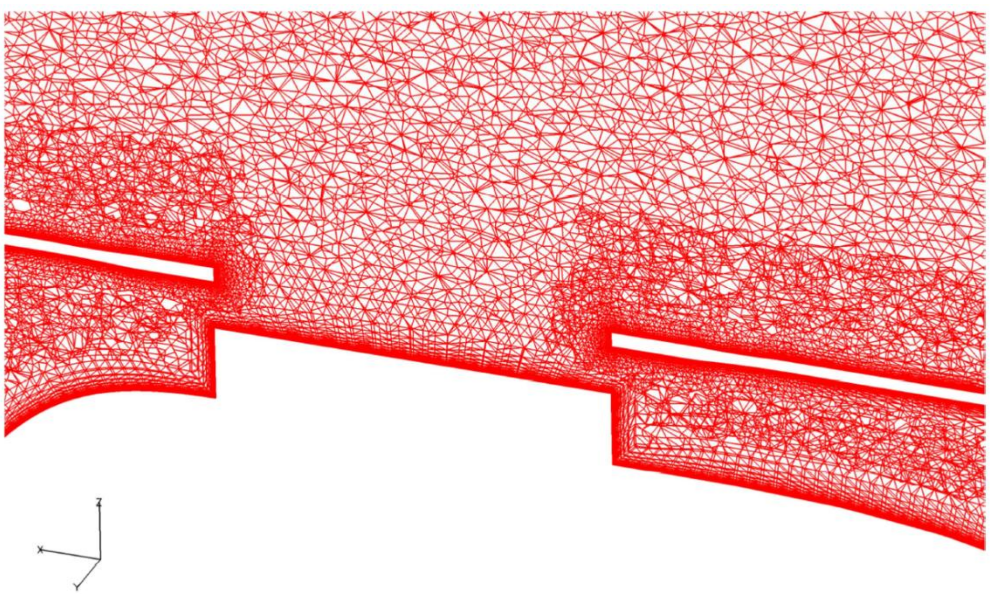
Detail of Blade/Fuselage overlap



HART-II case : OGA results



Before

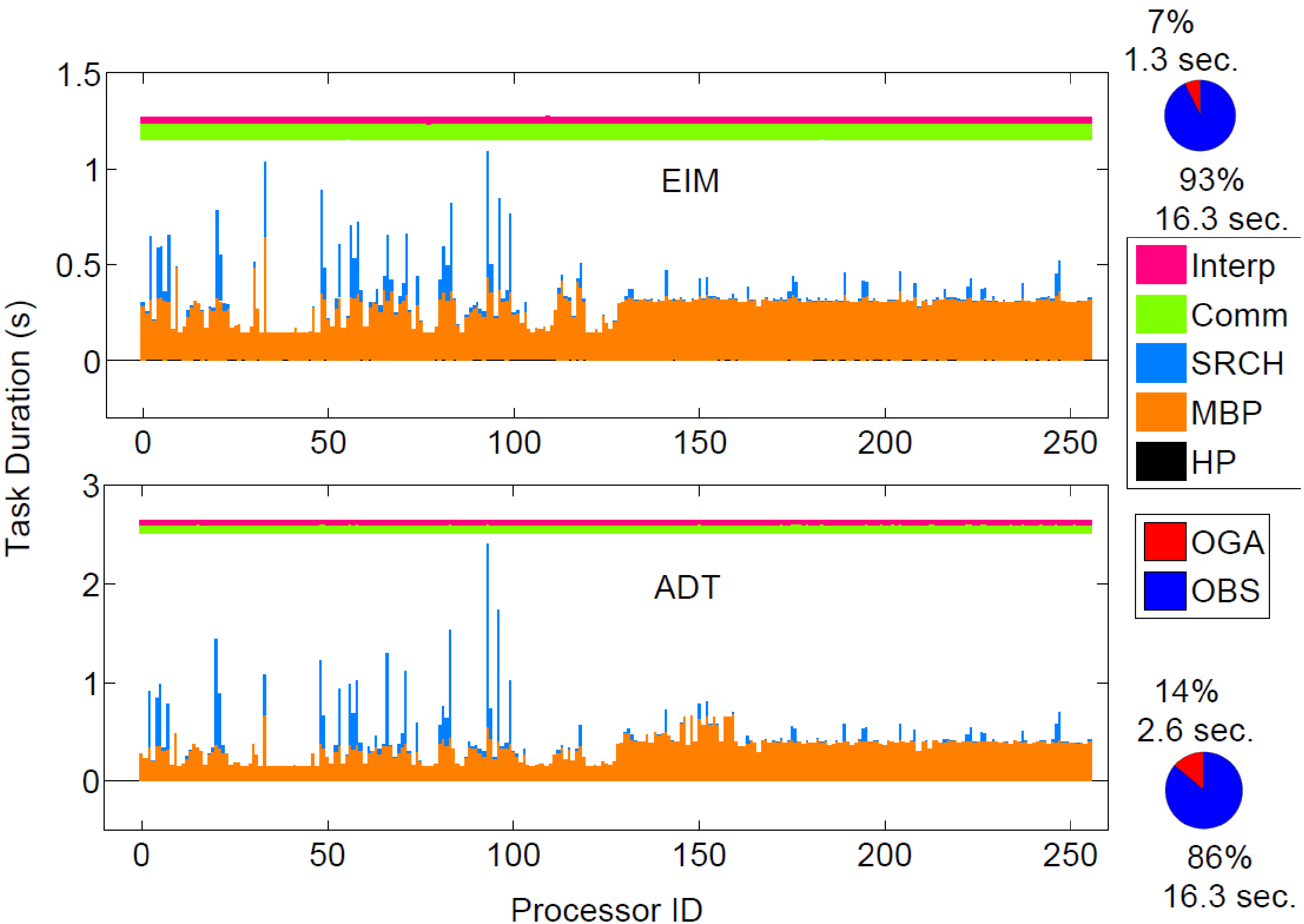


After:

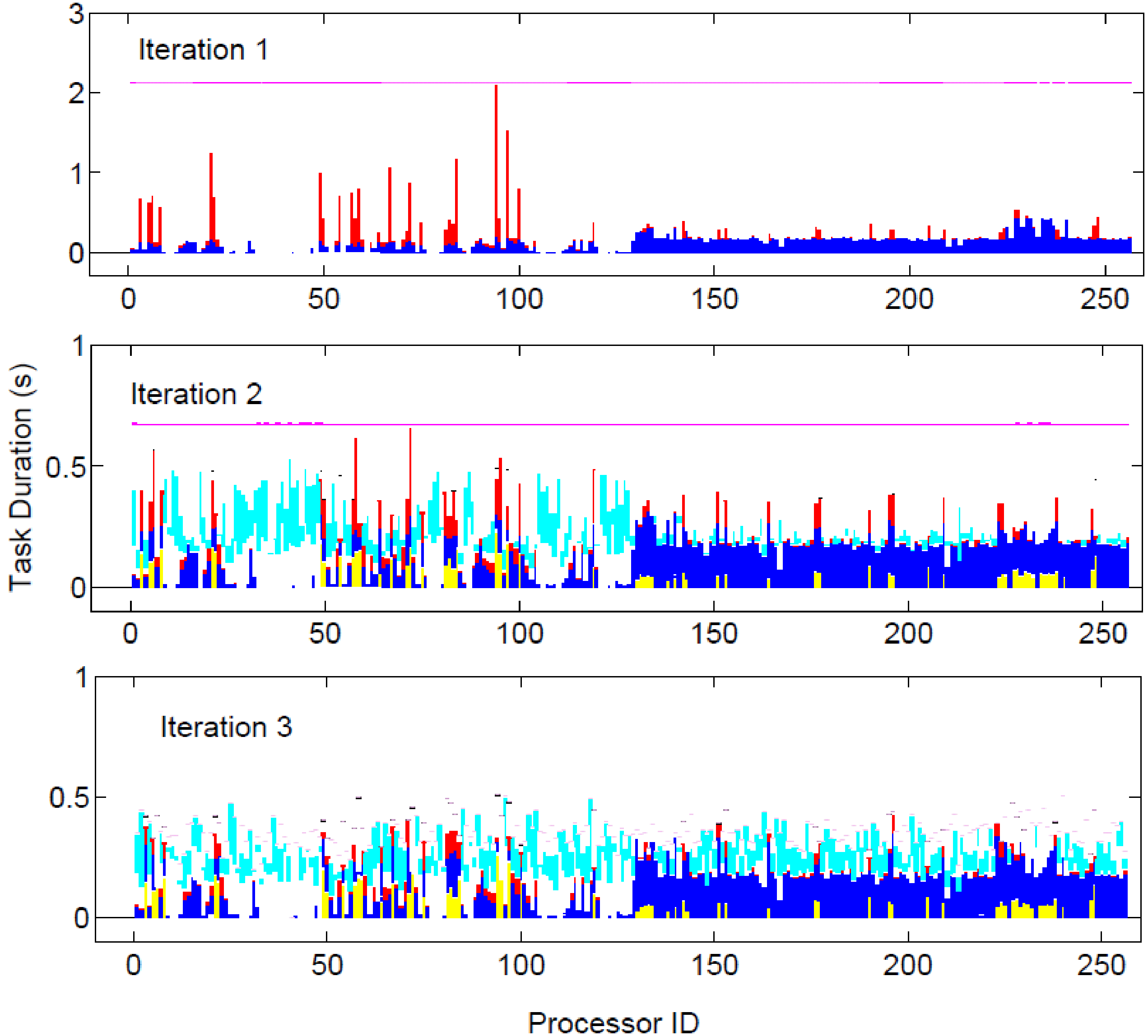
Same point types identified for ADT and EIM methods

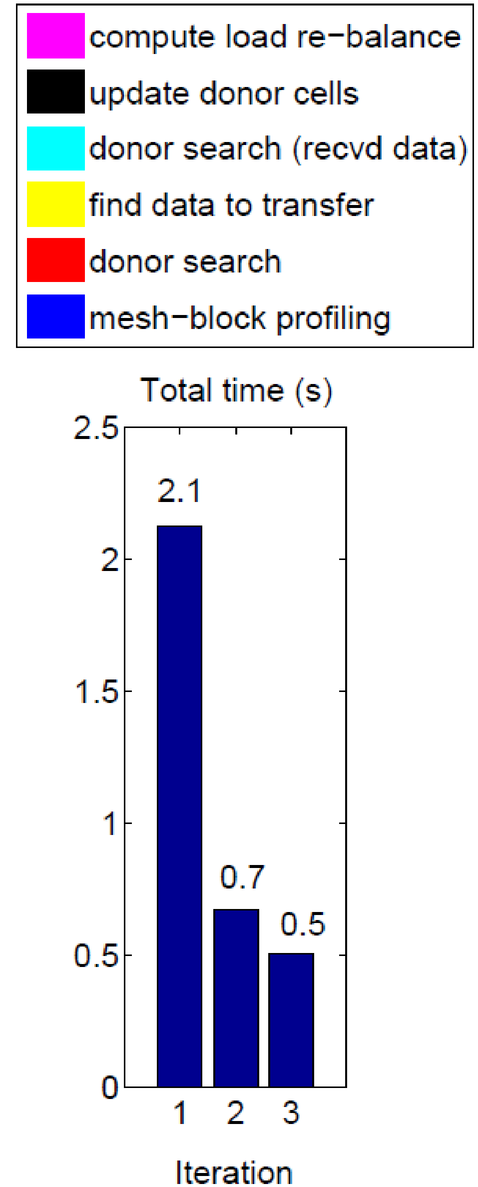
Mesh with best resolution selected automatically → minimal overlap

Timing comparison EIM / ADT

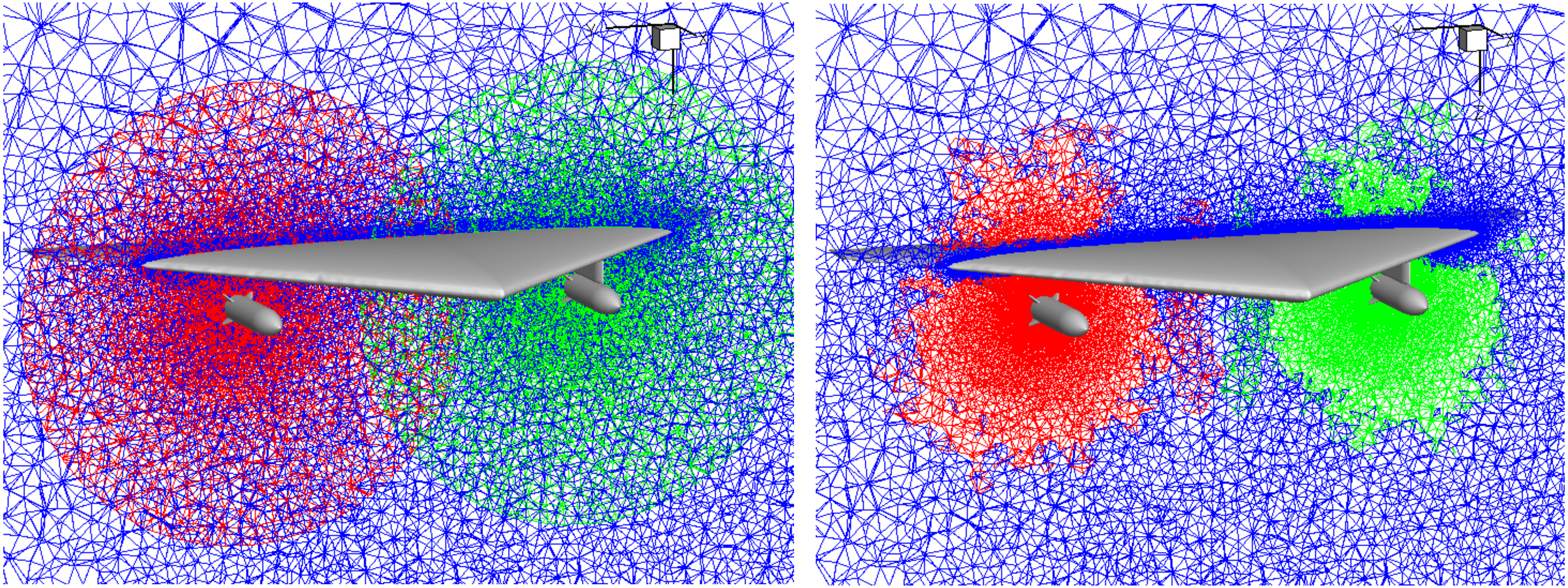


Load re-balance Results





Wing-Pylon-Store case (WPS)

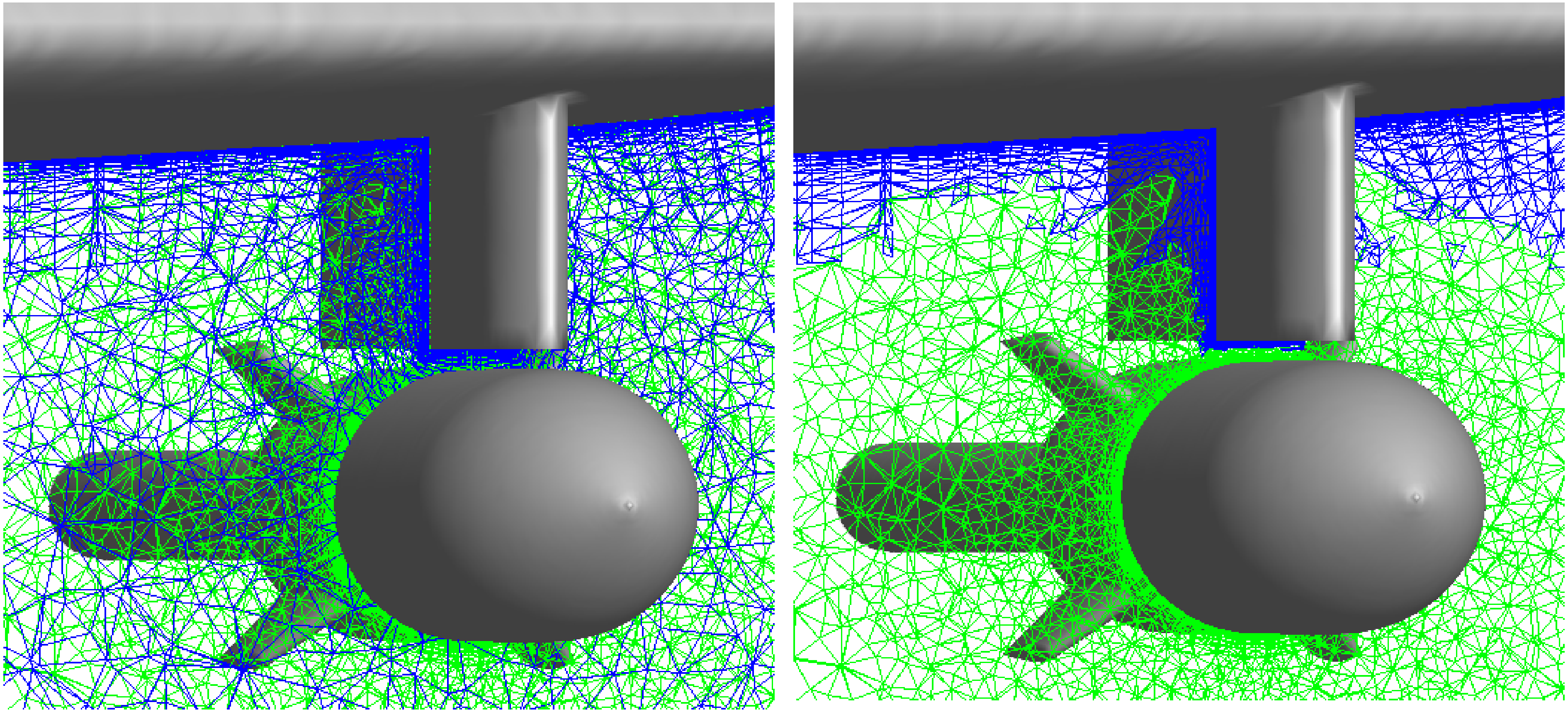


WPS case:

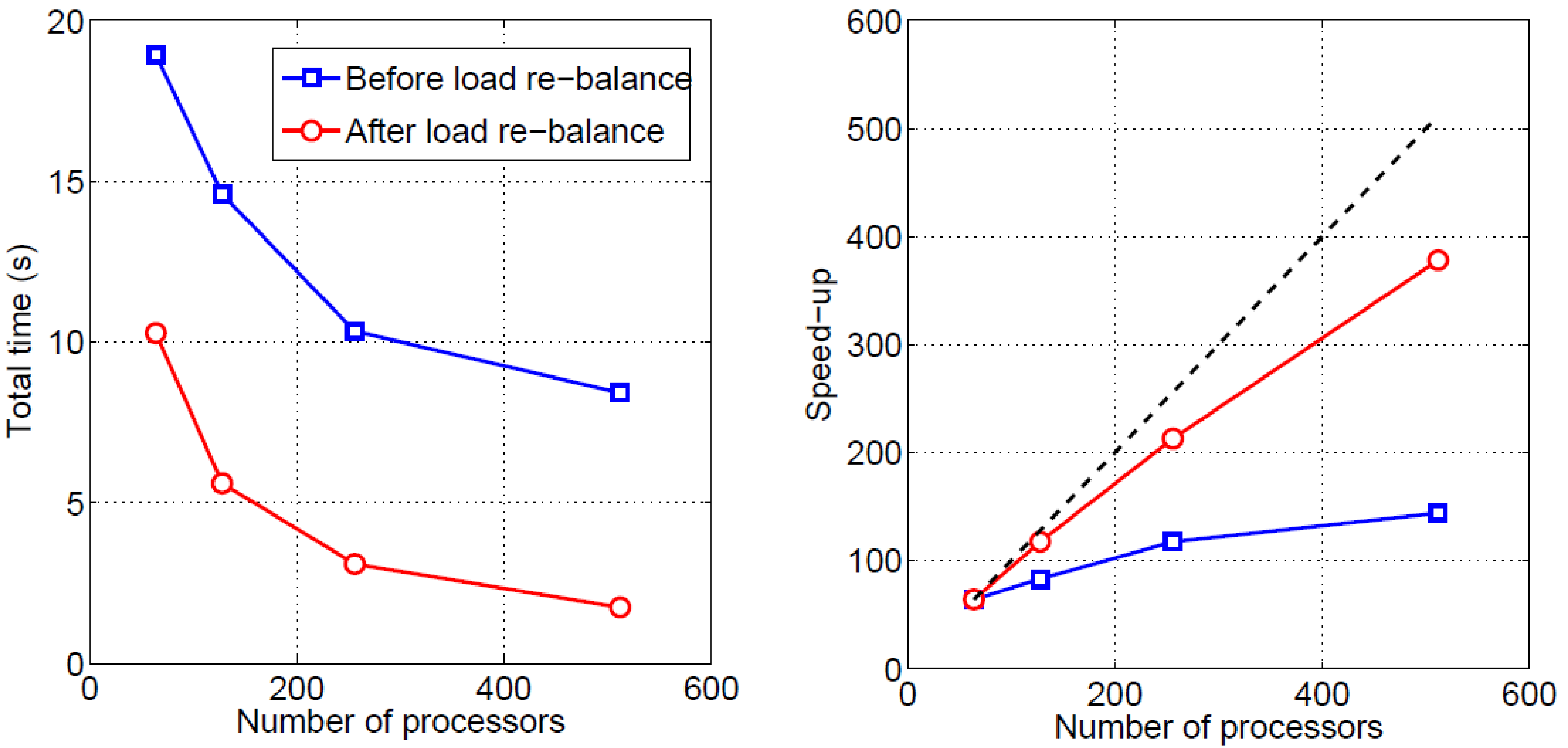
3 unstructured meshes
(1 wing, 2 stores)

15 million cells

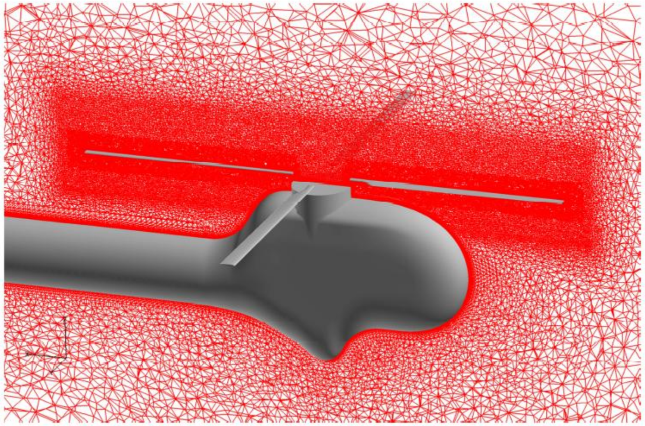
Detail of pylon/store overlap



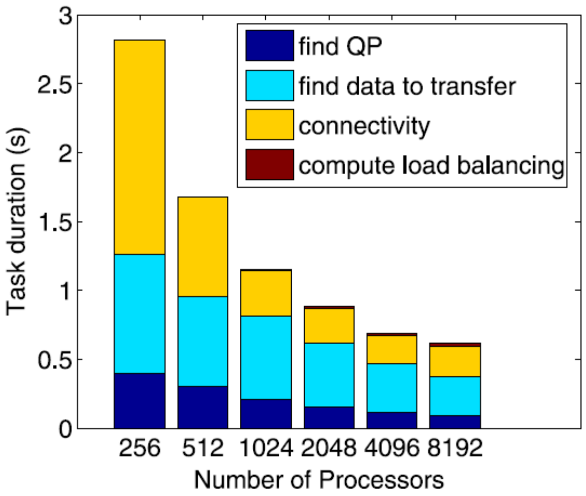
Scalability Results: WPS

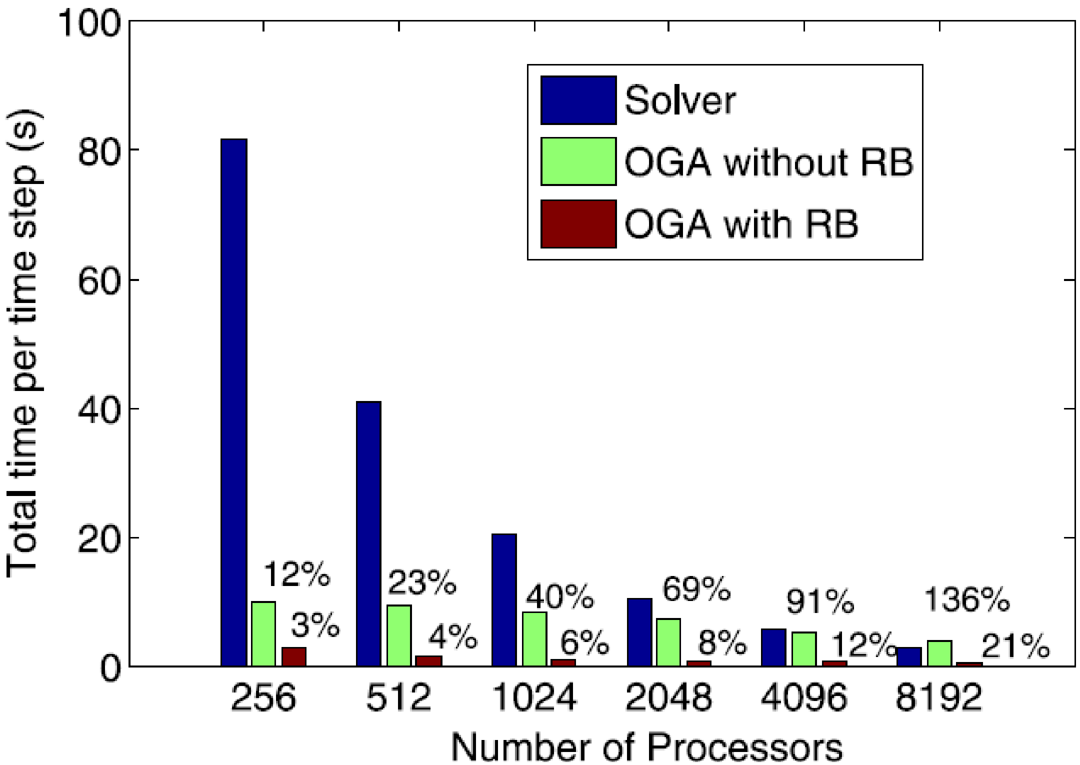


Scaling to large number of cores



HART-II with 80 million nodes and ~ 320 million cells

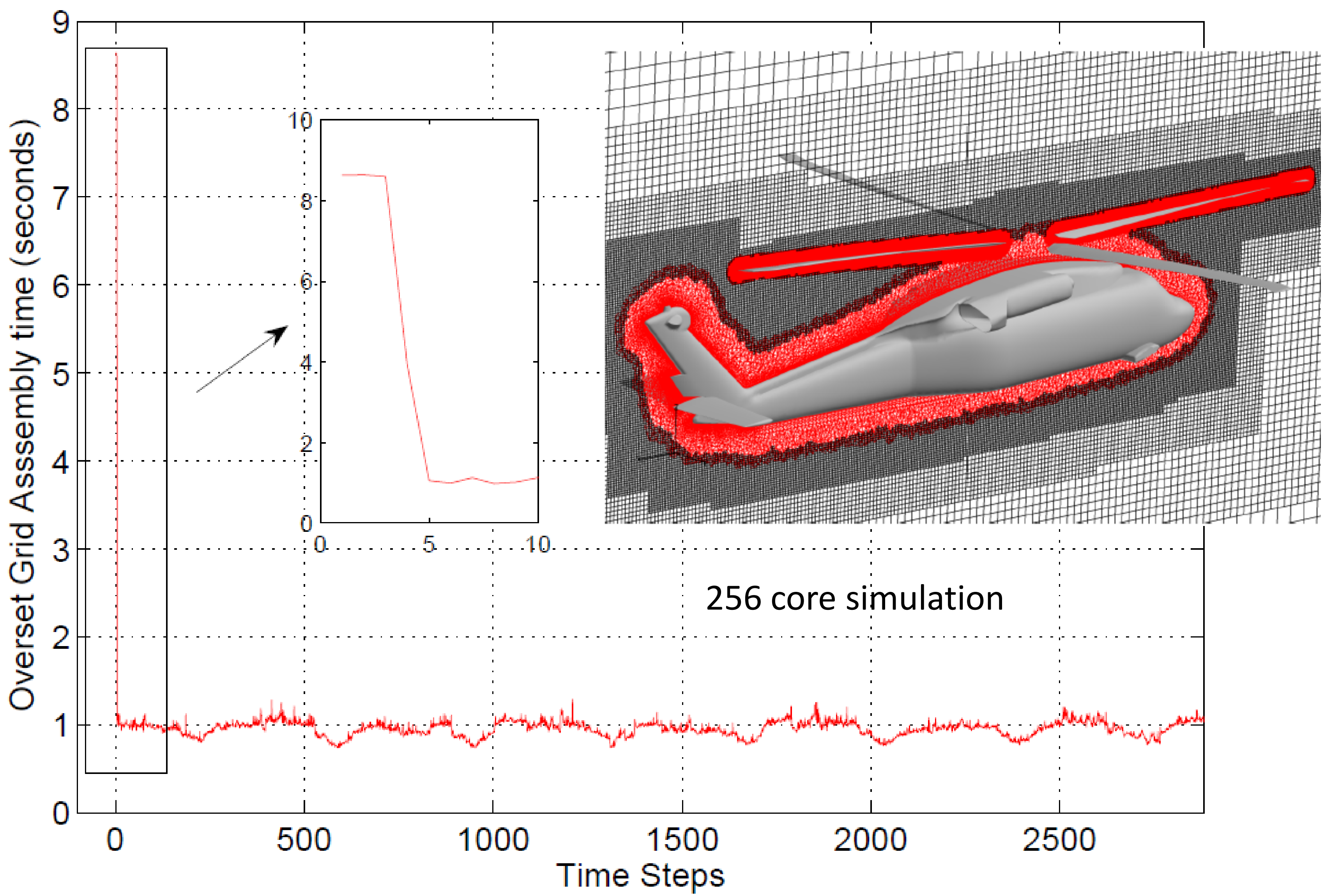




At 8192 cores overset grid assembly takes more time than solver time (136%) without load-balancing. With load-balancing this overhead is reduced to a manageable (20%).

However, OGA is still not linearly scalable

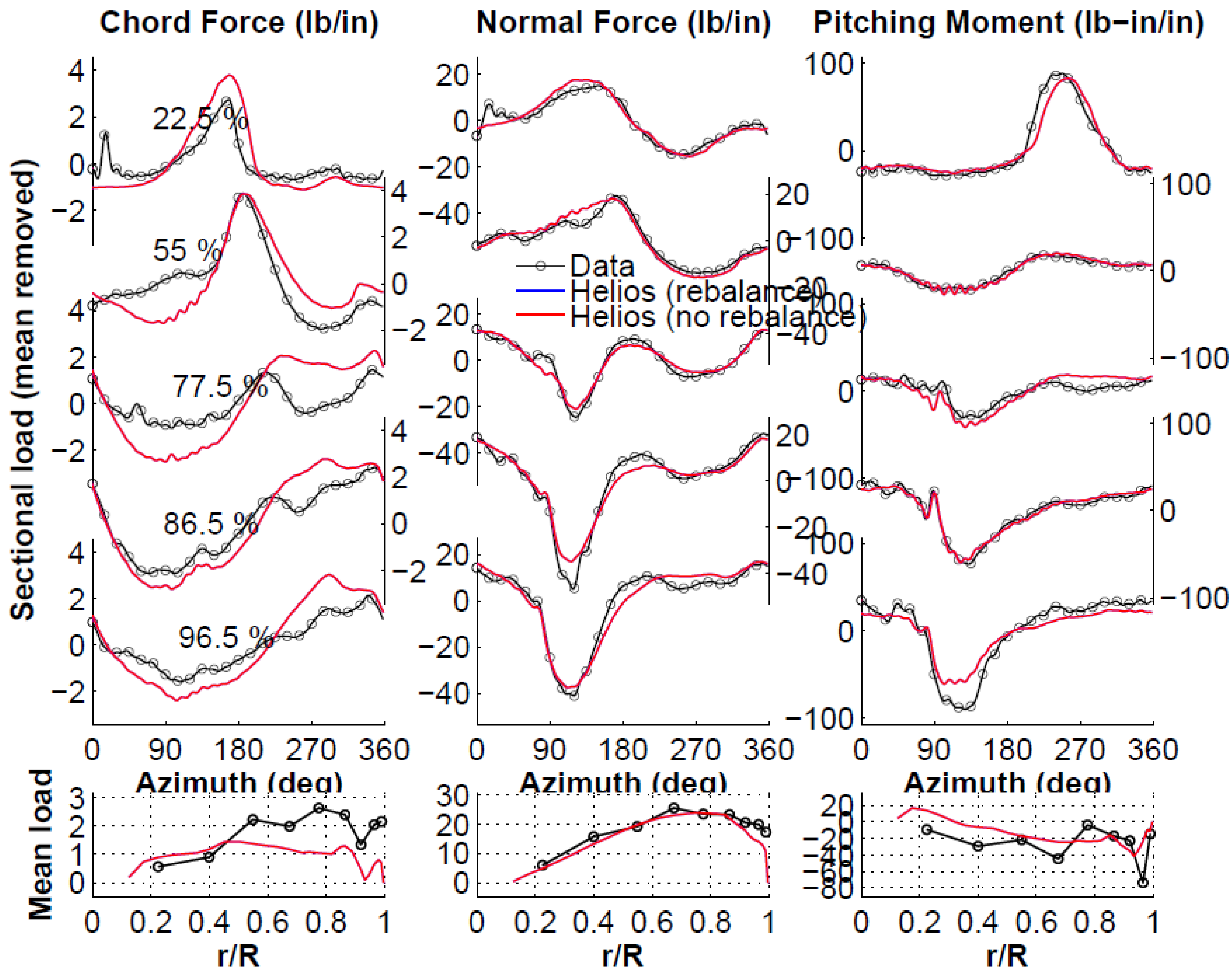
UH-60A CFD/CSD coupling



Predicted Aerodynamic loading

Overset grid assembly time reduced by an order of magnitude with the same end result in prediction

Increased throughput



Conclusions and Outlook

- Exact Inverse Map method to perform OGA on partitioned unstructured meshes in parallel:
  - method uses Cartesian auxiliary grids to build exact inverse maps to speed up donor search (line-walk search)
  - Method shown to be robust and accurate by comparing with ADT method, while at the same time more efficient than ADT (x 2 for HART-II case)
- Designed an adaptive load re-balance algorithm to tackle the large load imbalance:
  - improved efficiency (total time reduced by 76% for HART-II) and scalability (speed-up increased from 117 to 213 using 256 processor for the WPS case)
  - Showed improvement in execution time on up to 8192 cores

Future Work and Acknowledgements

- Explore further improvements in efficiency :

    ✓ Extend load re-balance algorithm for improving scalability further

- High-order and conservative overset grid assembly in parallel

We gratefully acknowledge:

- *Support from U.S. Army Research Office (Dr Roger Strawn)*
- *Contributions from CREATE A/V development team:*

    Dave McDaniel, Stephen Adamec, Todd Tuckey
    Robert Meakin, Mark Potsdam, Andrew Wissink