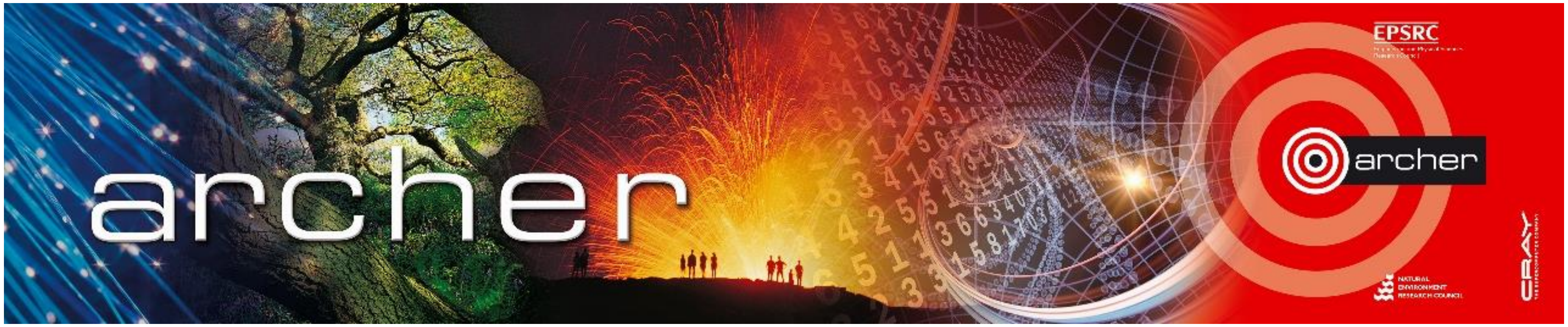


Welcome!

Virtual tutorial starts at 15:00 GMT

Please leave feedback afterwards at:
www.archer.ac.uk/training/feedback/online-course-feedback.php

Parallel supermeshing for multimesh modelling

ARCHER Virtual Tutorial, 13/07/2016

Iakovos Panourgias
<i.panourgias@epcc.ed.ac.uk>

Reusing this material

Outline

- Motivation
- Fluidity
- libsupermesh 2D / 3D
- Results

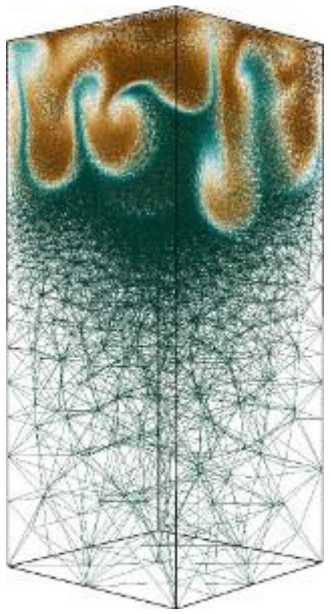
Motivation

- Models which use multiple non-matching unstructured meshes generally need to solve a computational geometry problem, and construct intersection meshes in a process known as supermeshing.

- Parallel supermeshing; for meshes with non-matching domain decompositions.

Fluidity

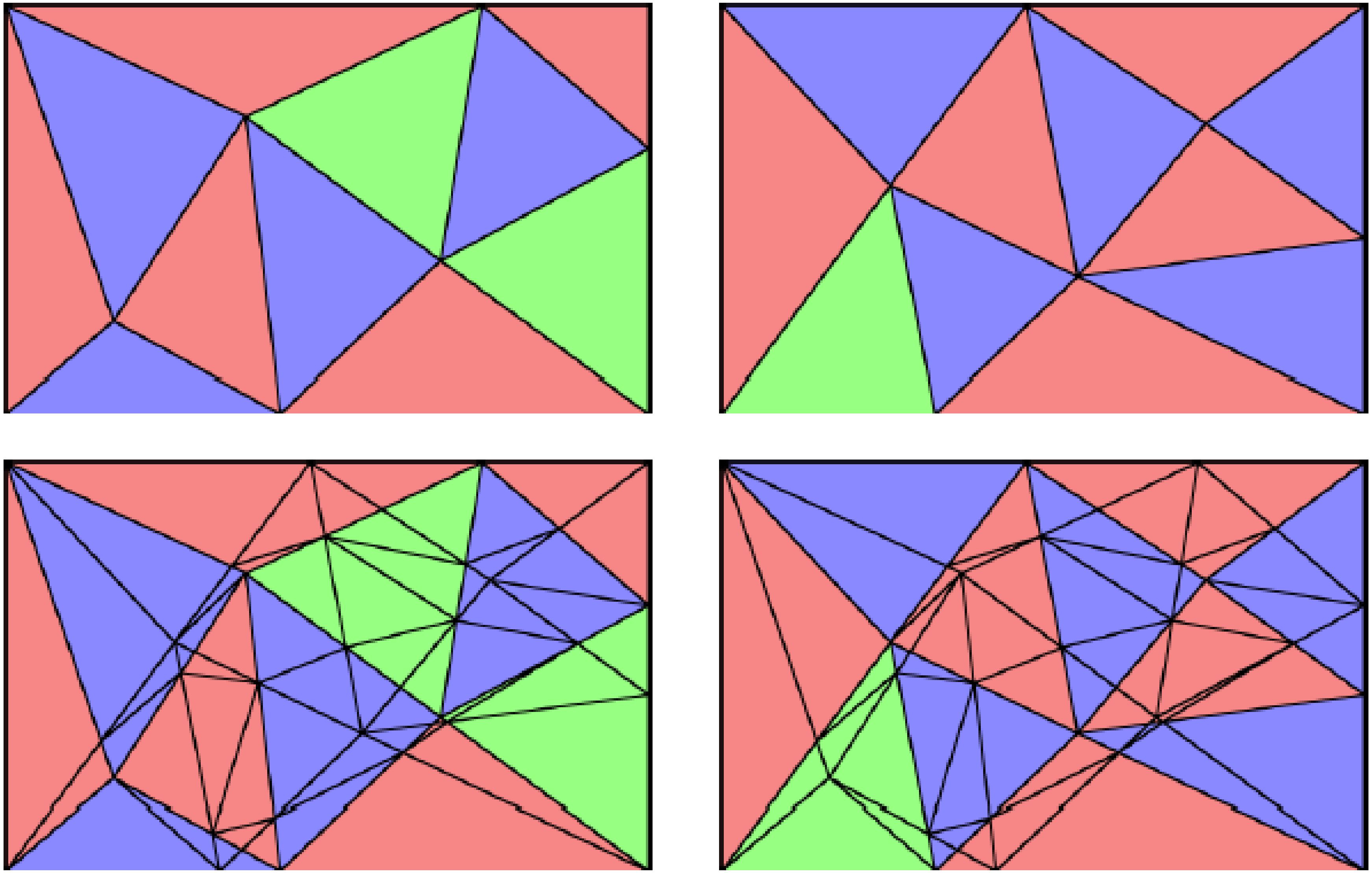
- Unstructured finite element code (one, two and three dimensions)
- Anisotropic mesh adaptivity
- Applications:
- CFD, geophysical fluid dynamics,
  mantle convection,
  ocean and reservoir modelling,
  mining, etc.



Supermeshing

- Was originally developed to perform data interpolation on two or more different unstructured meshes.
- Allows equations to be solved on a complex unstructured mesh while simultaneously using input data from a separate and completely different unstructured mesh.
- A supermesh is a mesh with the following property: each element in the supermesh is contained within exactly one element of each parent mesh.

Supermeshing Figure



Supermeshing algorithm

1. Identification of pairs of elements, one on each mesh, which intersect;

2. Generation of a mesh of their intersection (the "local supermesh");

3. The transfer of data onto this intersection mesh.

libsupermesh

Key capabilities:

- Easy to use
- Handles unstructured meshes with non-matching domain decompositions
- General purpose library (user specified compute functionality)
- Implements several new algorithms for identifying candidate pairs of intersection elements
- Has been tested on up to 10,000 cores on ARCHER
- Has been tested as a replacement intersector finder with Fluidity

Getting libsupermesh

libsupermesh has been compiled and tested on ARCHER (using GNU (**5.1.0**), Intel (**15.0.2.164**) and Cray (**8.4.1**) compilers).

- The library, manual and example programs are available at:

https://bitbucket.org/libsupermesh/

- Minimal dependencies to install
- Build instructions and cmake file provided
- Distributed under the terms of the GNU Lesser General Public License version 2.1

Compiling with libsupermesh

Include a single libsupermesh module file:
use libsupermesh_parallel_supermesh

You have to compile libsupermesh with the same compiler (and version) as your application.

libsupermesh development (serial)

- Existing Fluidity supermeshing code was extracted and copied to a standalone library (libsupermesh)
- CGAL[1] element intersection code was removed
- Wild Magic intersection code was removed
- Intersection code was replaced with an optimised implementation
- Supermeshing and intersection code was cleaned up – removing all dependencies with Fluidity data structures

1 http://www.cgal.org/

libsupermesh development (serial)

Several algorithms for identifying candidate pairs of elements that may intersect were implemented. All algorithms are based upon an axis-aligned bounding box (AABB) intersection predicate:

- Sort intersection
- Quadtree intersection finder
- Octree intersection finder
- R*-tree intersection finder[1]
- Advancing front intersection finder[2]

1 http://libspatialindex.github.io/
2 "P. E. Farrell and J. R. Maddison, "Conservative interpolation between volume meshes by local Galerkin projection", Computer Methods in Applied Mechanics and Engineering, 200, pp. 89-100, 2011"

libsupermesh development (serial)

The libsupermesh standalone library exposes several interfaces which return a local mesh of the intersection of two elements (a local supermesh). The following interfaces are supported:

- One dimension: interval intersection, intended primarily for code testing;
- Two dimensions: Intersection of two-dimensional convex polygons using the Sutherland- Hodgman clipping algorithm
- Three dimensions: Intersection of three-dimensional convex polyhedra using the “plane-at-a-time clipping” algorithm
- General dimensions: convenience interfaces which use one of the above mentioned methods

libsupermesh development (serial)

- Integration with Fluidity (as a replacement intersection and supermeshing implementation)
- libsupermesh has been added as an optional Fluidity component

libsupermesh testing (serial)

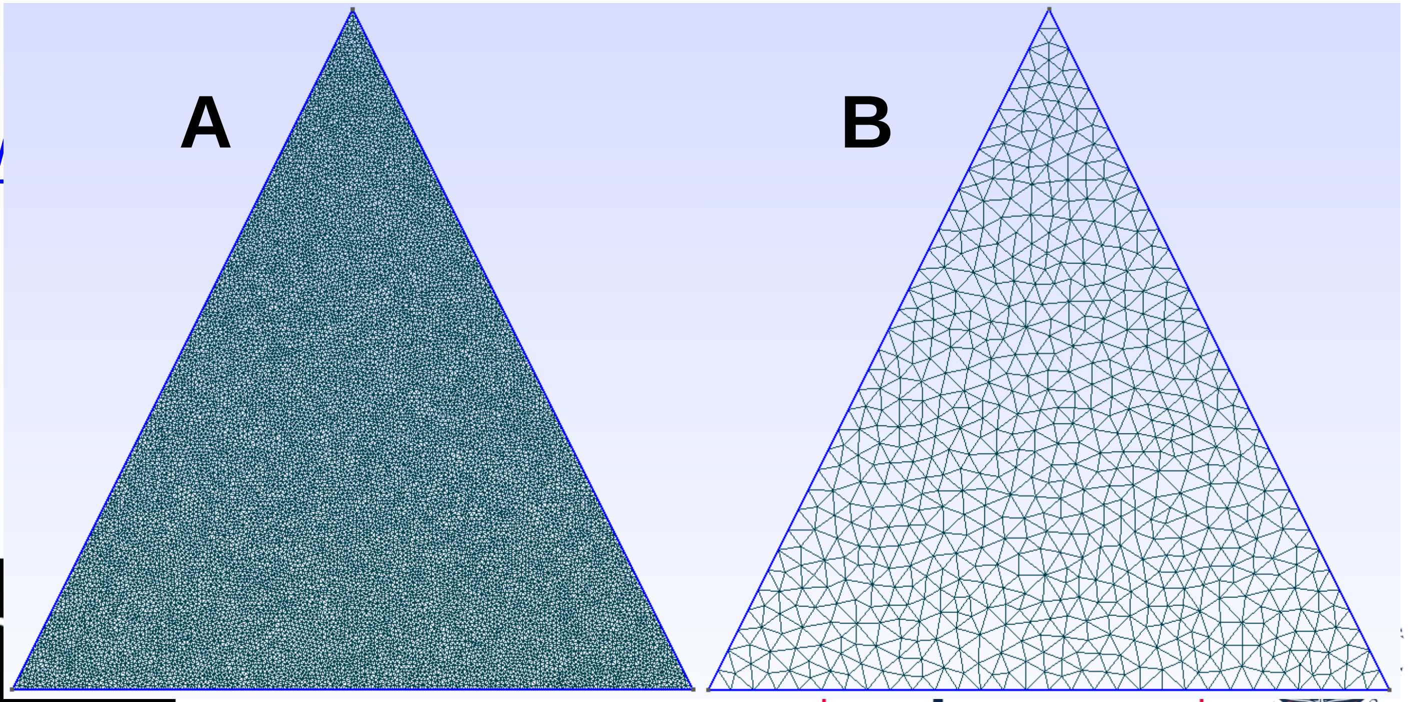
- Wrote a regression testing suite
- Used Fluidity regression testing suite
- Run tests with the following GNU Fortran options:

```
-O0 -g -Wall -fcheck=all
-ffpe-trap=invalid,zero,overflow,underflow
-finit-integer=-66666 -finit-real=nan -fimplicit-none
```

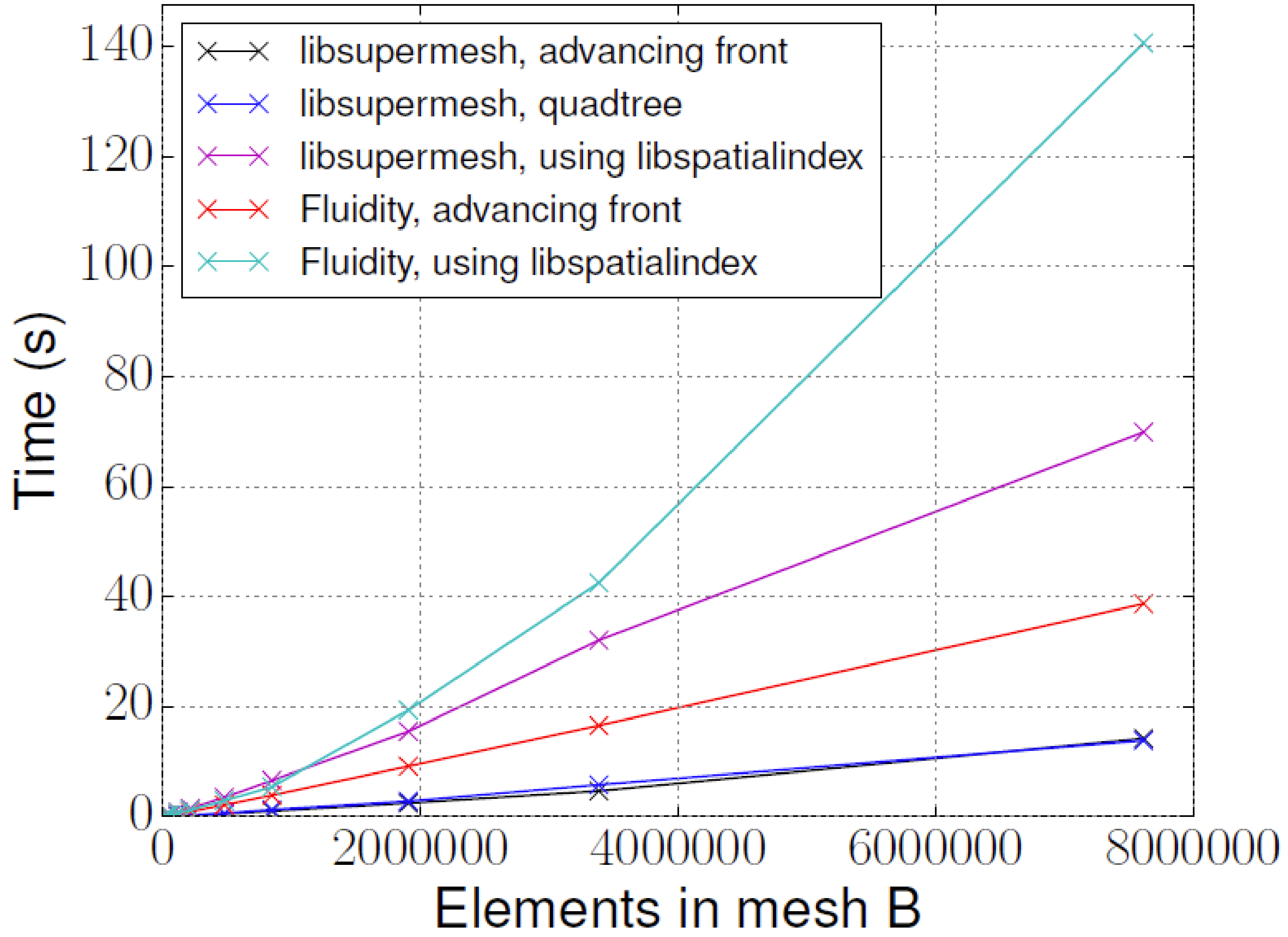
libsupermesh results (serial)

- The 2D/3D benchmarks take as input two quasi-uniform resolution unstructured triangle/tetrahedra meshes of an equilateral triangle/square pyramid domain, A and B, with mesh B having roughly one half the element size of mesh A.
- The meshes were generated using Gmsh[1].

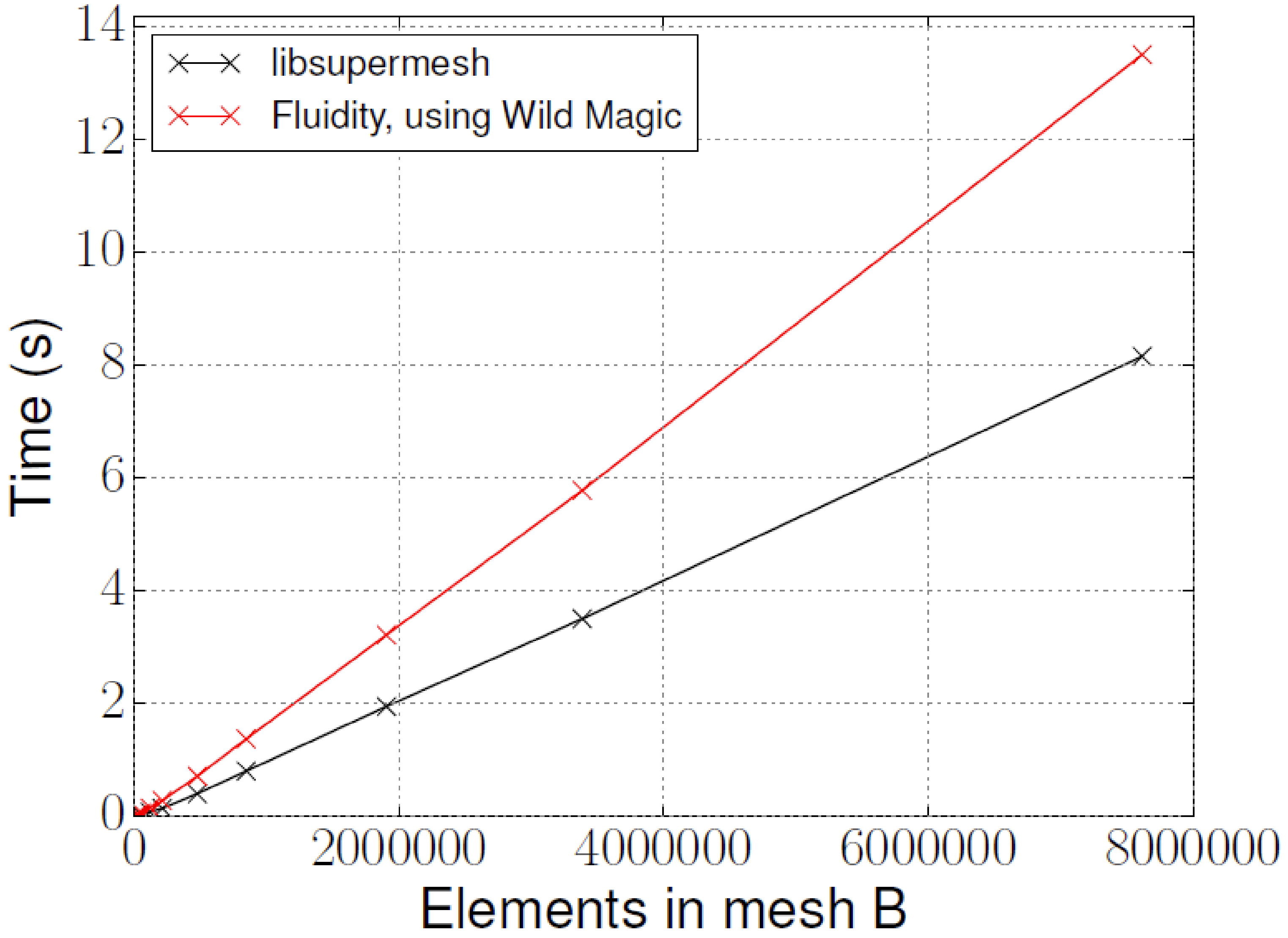
1 http://gmsh.info/



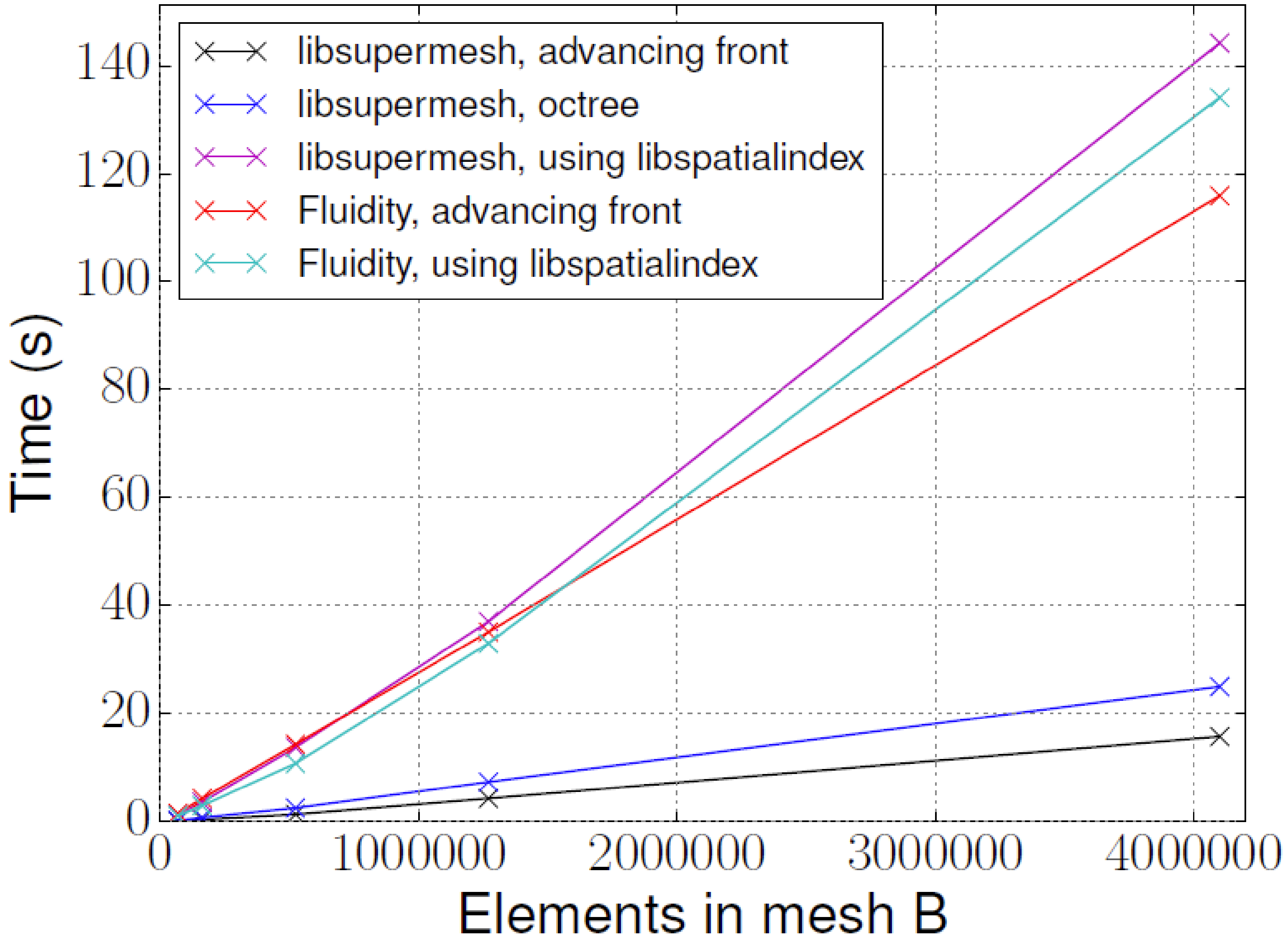
libsupermesh results (serial) – 2D Intersection Finder



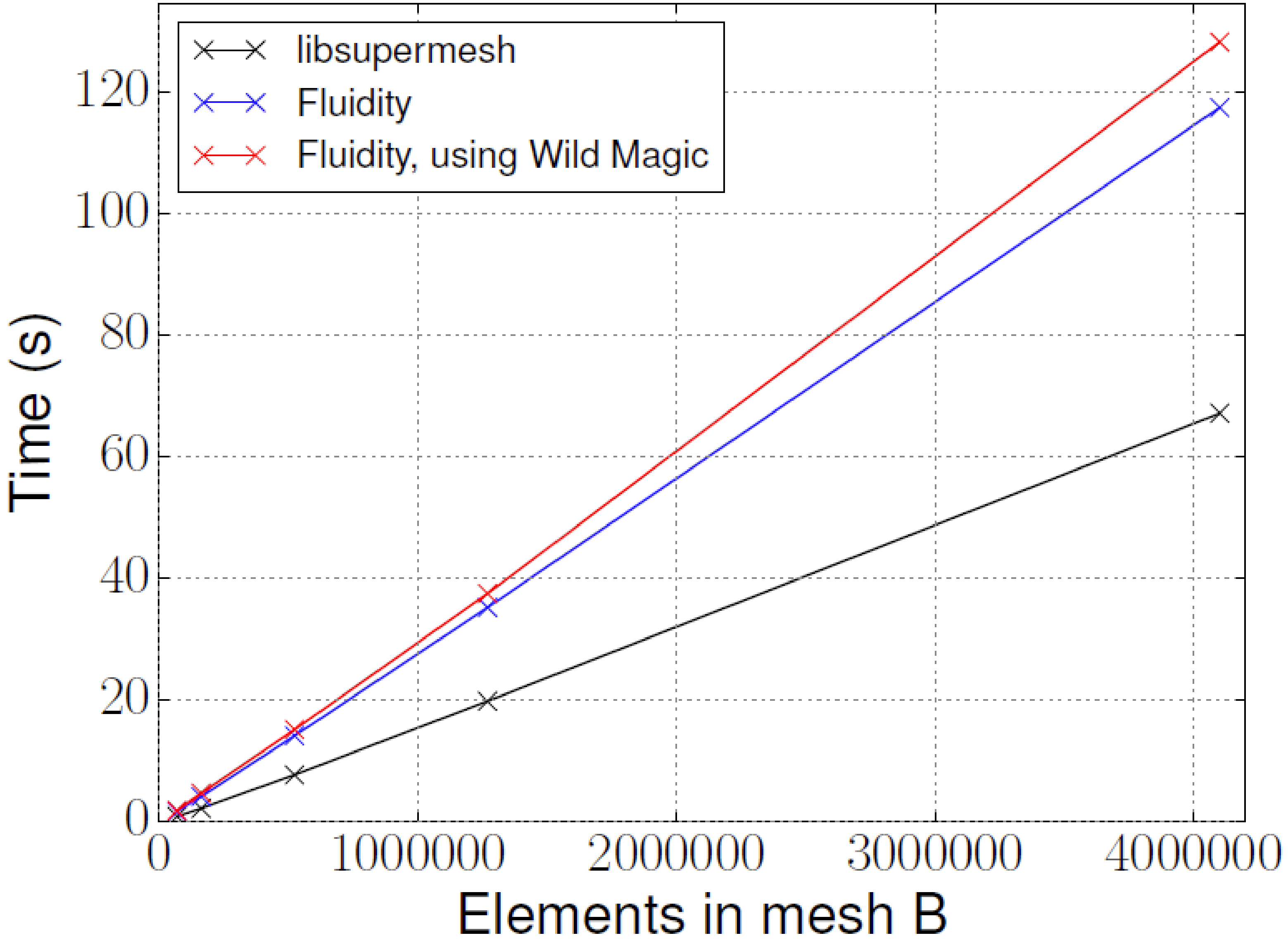
libsupermesh results (serial) – 2D Intersector



libsupermesh results (serial) – 3D Intersection Finder



libsupermesh results (serial) – 3D Intersector



libsupermesh development (parallel)

- The algorithm for constructing a local supermesh is trivially parallelisable*

* as long as the decompositions of the two meshes are perfectly aligned.

libsupermesh development (parallel)

In order to generalise the algorithm to perform parallel supermesh calculations on meshes that are not perfectly aligned, the following algorithm was implemented:

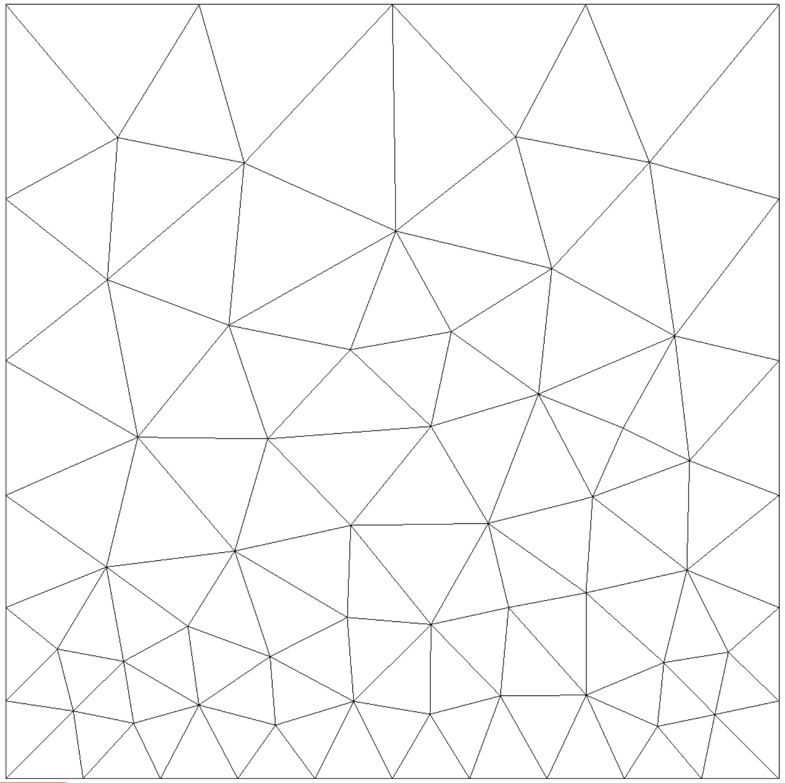
1. Communicate the axis-aligned bounding boxes (AABBs) of all mesh A partitions and all mesh B partitions using all-to-all communication;
2. For each mesh A partition whose AABB intersects with the local mesh B AABB:
   a) Identify local mesh B elements whose AABBs intersect with the AABB of the mesh A partition;
   b) Obtain data associated with these elements, and communicate these data via point-to-point communication.
3. Construct the intersection meshes for local mesh A and local mesh B elements, and perform calculations on these intersection meshes;
4. For each mesh B partition whose AABB intersects with the local mesh A AABB:
   a) Unpack data communicated in step 2b;
   b) Construct the intersection meshes for local mesh A and received mesh B elements, and perform calculations on these intersection meshes.

libsupermesh development (parallel) – Step 1

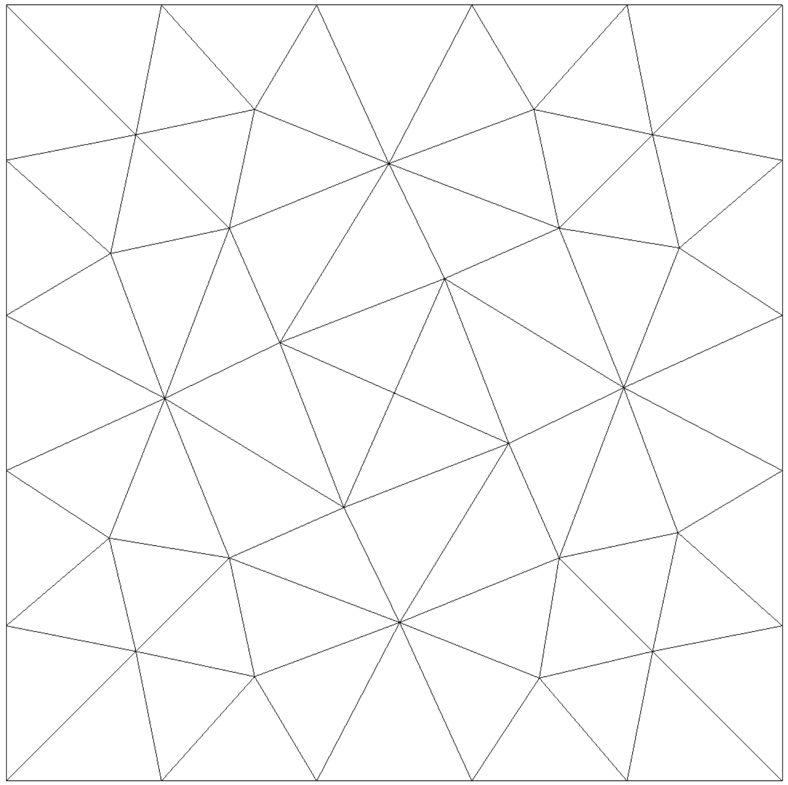
- Calculate the axis-aligned bounding box (AABB) of the local meshes (A and B)
- Use MPI all-to-all communication to distribute the bounding boxes across the whole domain
- After this step MPI processes know the bounding boxes of all mesh partitions

libsupermesh example – meshes

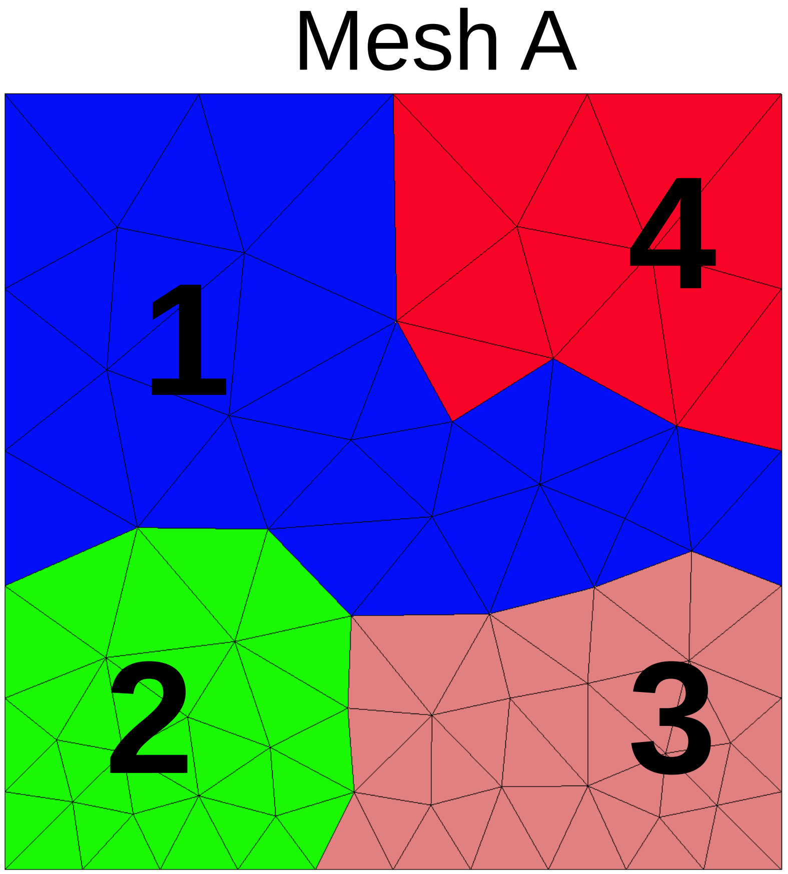
Mesh A

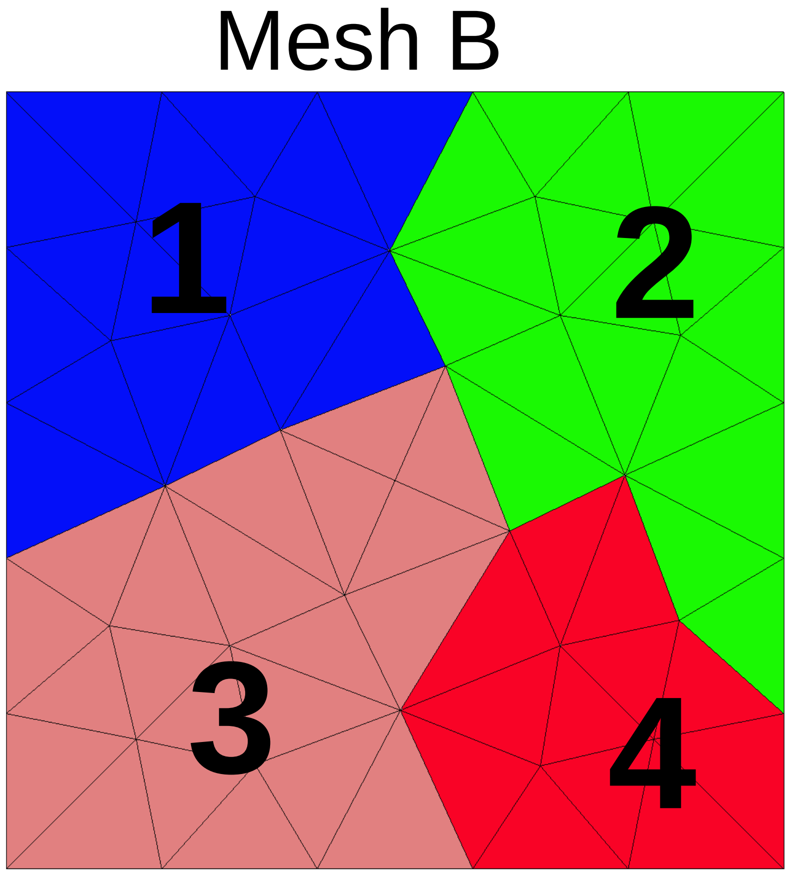


Mesh B



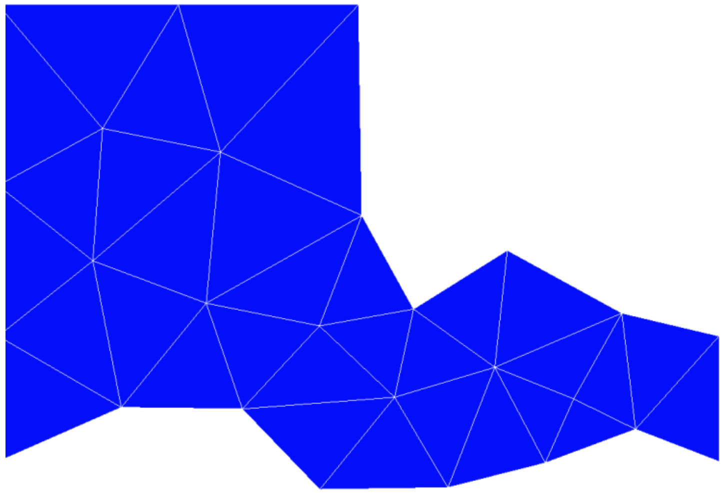
libsupermesh example – partitioned meshes



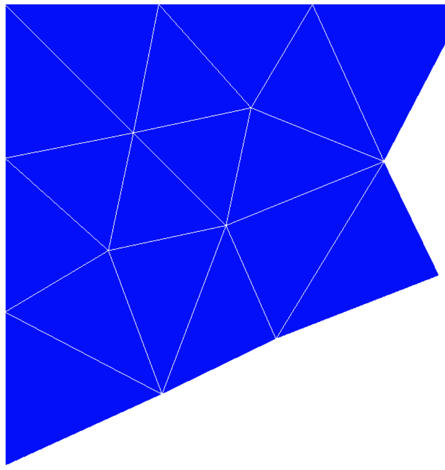


libsupermesh example – MPI Proc 1 view

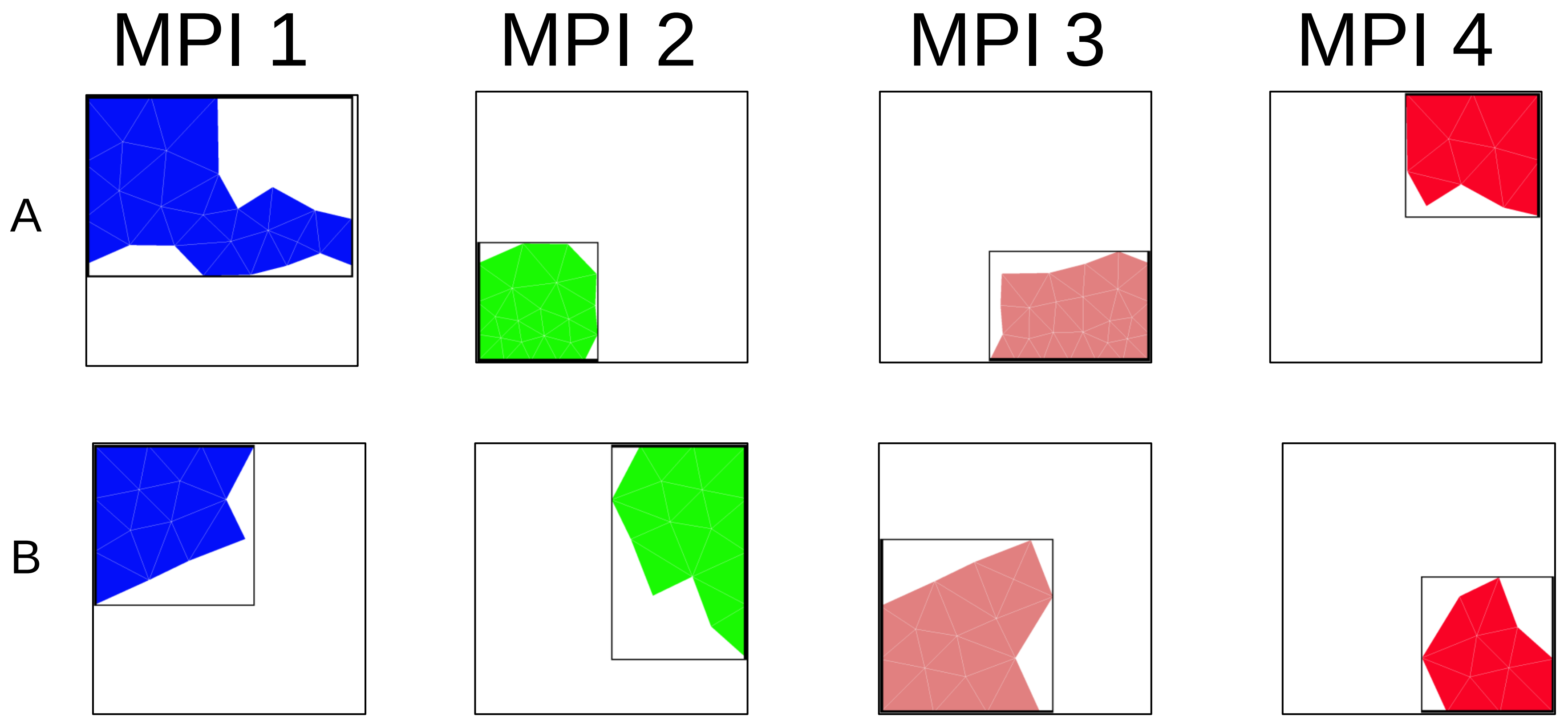
Mesh A



Mesh B



libsupermesh example – Step 1

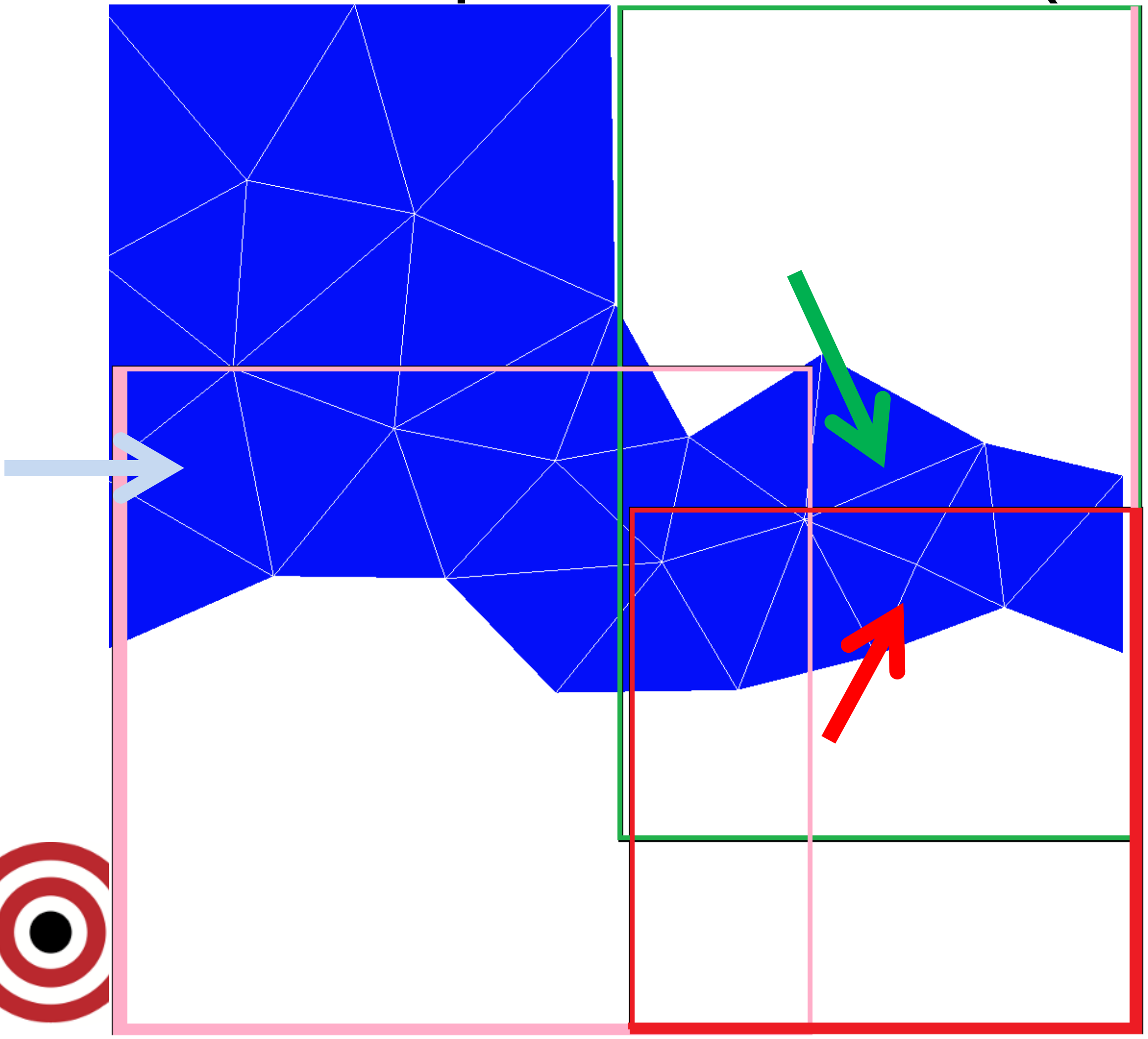


libsupermesh development (parallel) – Step 2

- Each MPI process runs a test on the bounding boxes of each mesh A partition, communicated in step 1, with the local mesh B partition
- If the bounding boxes intersect then some of the local mesh B elements may intersect with some mesh A elements on a different process (further processing in Step 2a)
- Similarly each process runs a test on the bounding boxes of each mesh B partition, communicated in step 1, with the local mesh A partition
- If the bounding boxes intersect then some of the local mesh A elements may intersect with some mesh B elements stored on a remote process. An MPI_RECV is issued.

libsupermesh example – Step 2

MPI process 1 view (local)

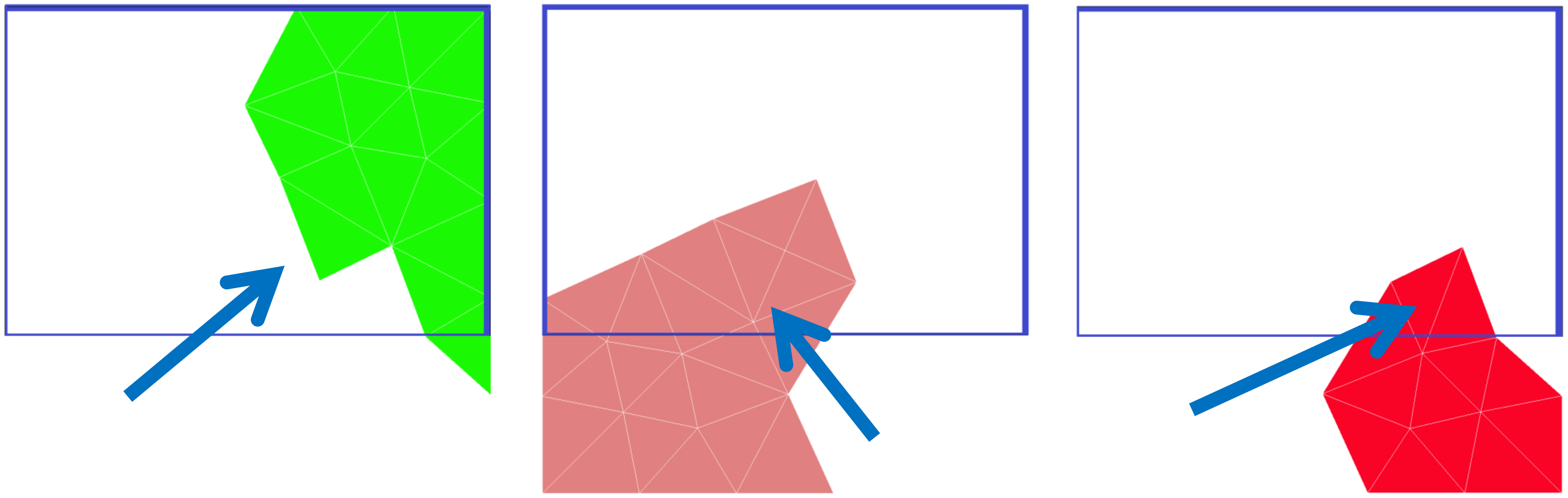


Local
MPI Process 1 tests and expects intersections with **remote** MPI processes 2, 3 and 4.
Posts MPI_RECVs.

libsupermesh example – Step 2

MPI process 2, 3, 4 view (remote)

Some local elements intersect with AABB.



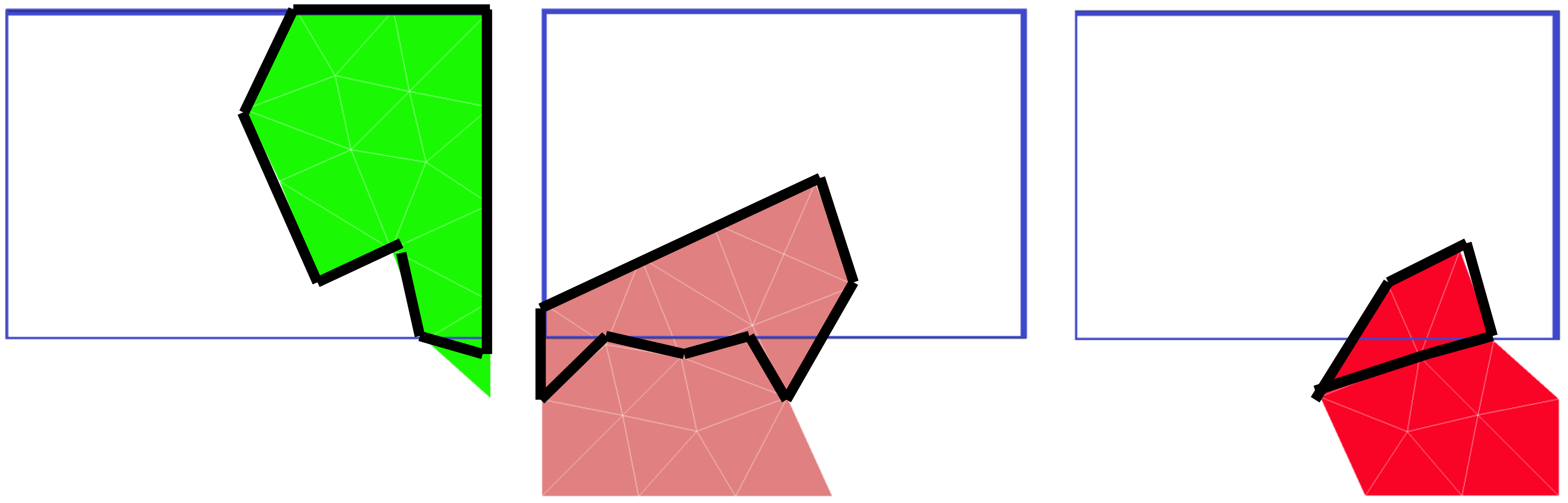
libsupermesh development (parallel) – Step 2a

- In this step each local mesh B element bounding box is tested against the bounding box of the mesh A partition identified in step 2
- The local mesh B elements which intersect with the bounding box of the mesh A partition are marked for sending.

libsupermesh example – Step 2a

MPI process 2, 3, 4 view (remote)

MPI processes 2, 3, 4 test and send elements that intersect.



libsupermesh development (parallel) – Step 2b

- In this step the local mesh B elements to be communicated to remote processes are known
- A user provided callback function is used to create a packed array containing necessary user data
- Once the data have been packed, libsupermesh creates a packed MPI message containing additional meta-data. The packed MPI message has the following format:

1. Number of elements (MPI_INTEGER);
2. Number of mesh vertices (MPI_INTEGER);
3. Connectivity of mesh vertices (flat array of MPI_INTEGER);
4. Coordinates of mesh vertices (flat array of MPI_DOUBLE_PRECISION);
5. Size of user supplied data (MPI_INTEGER);
6. User supplied data (MPI_BYTE).

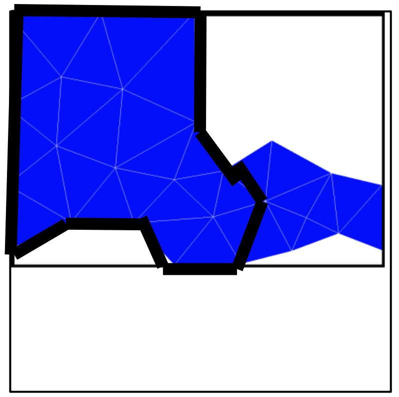
libsupermesh development (parallel) – Step 3

- This step handles the case where elements in the local mesh A and local mesh B intersect.
- If there is a local intersection, then the intersection meshes can be constructed and all relevant calculations can be performed using a user provided callback function

libsupermesh example – Step 3

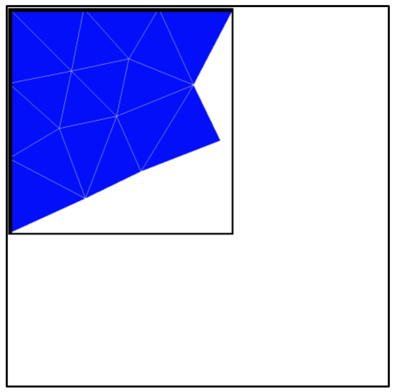
MPI Process 1

Local mesh A:



This area can be calculated, locally.
There is no need to communicate.

Local mesh B:

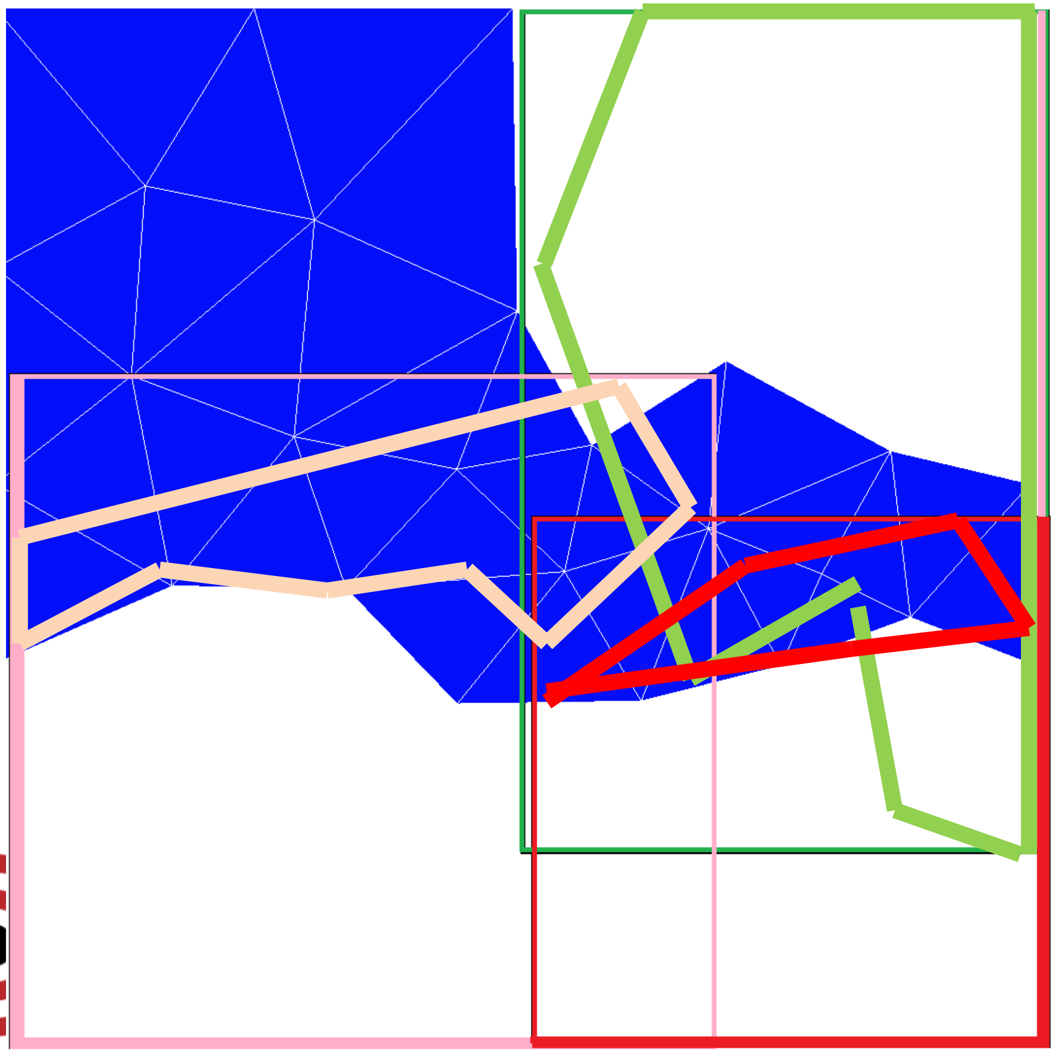


libsupermesh development (parallel) – Step 4a

- The receiving MPI process knows if the local mesh A AABB intersects with the mesh B partition (based on the test in step 2)
- The receiving MPI process call MPI_Probe (since the size of the MPI message is not known)
- If the first MPI_INTEGER of the received MPI message is equal to 0; the message is discarded (since it was false positive)
- If the first MPI_INTEGER of the received MPI is not equal to 0; the message is unpacked and a user specified unpack function is called.

libsupermesh example – Step 4a

MPI process 1 view (local)



Local
MPI Process 1 calls MPI_Probe in order to receive elements from the remote MPI procs. MPI Process 1 has now enough remote elements to construct the local supermesh.

libsupermesh development (parallel) – Step 4b

- The final step constructs the intersection meshes of the communicated mesh B elements and the local mesh A elements.
- The candidate intersection identification is performed using the libspatialindex R*-tree algorithm (the faster quadtree (2D) and octree (3D) were completed late in the project).

Simple application that uses libsupermesh

You can follow the examples and tests in **/src/tests** directory.
Pseudo-code:

1. Start of application.
2. Include the libsupermesh_parallel_supermesh module
3. Set up mesh data (libsupermesh supports reading files which are in Triangle[1] or TetGen[2] .node and .ele format) for mesh A and B.
4. Call supermesh*
5. libsupermesh will compute the local supermesh.
6. It will then apply the user provided calculation function to the supermesh.
7. At the end of the call each MPI process will have a result for the local supermesh.

1 https://www.cs.cmu.edu/~quake/triangle.html

2 http://wias-berlin.de/software/tetgen/

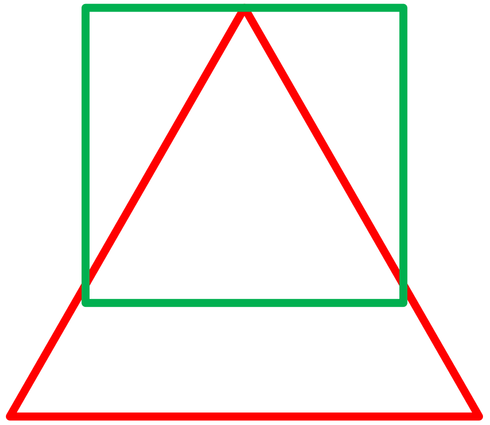
libsupermesh results (parallel)

- The 2D and 3D benchmarks take as inputs two meshes (A and B) and construct an intersection mesh
- The 2D benchmarks mesh A is a triangle shaped mesh, whereas mesh B is a square shaped mesh
- The 3D benchmarks use a pyramid shaped mesh (as mesh A) and a cubed shaped mesh (as mesh B)
- The meshes were generated using Gmsh[1]
- The Fluidity tool “fldecomp” was used to partition the meshes.

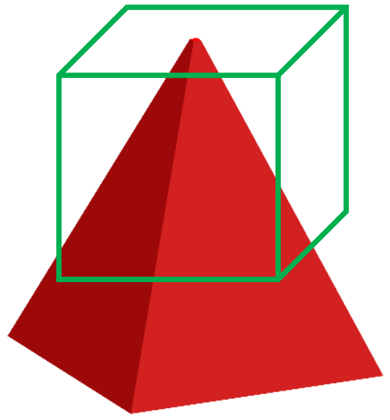
1 http://gmsh.info/

libsupermesh results (parallel) - meshes

2D



3D



libsupermesh results (parallel) - benchmarks

Small and large benchmarks:

- 33,065,204 elements for mesh A and 29,399,556 elements for mesh B (2D)
- 33,077,698 elements for mesh A and 27,301,039 elements for mesh B (3D)

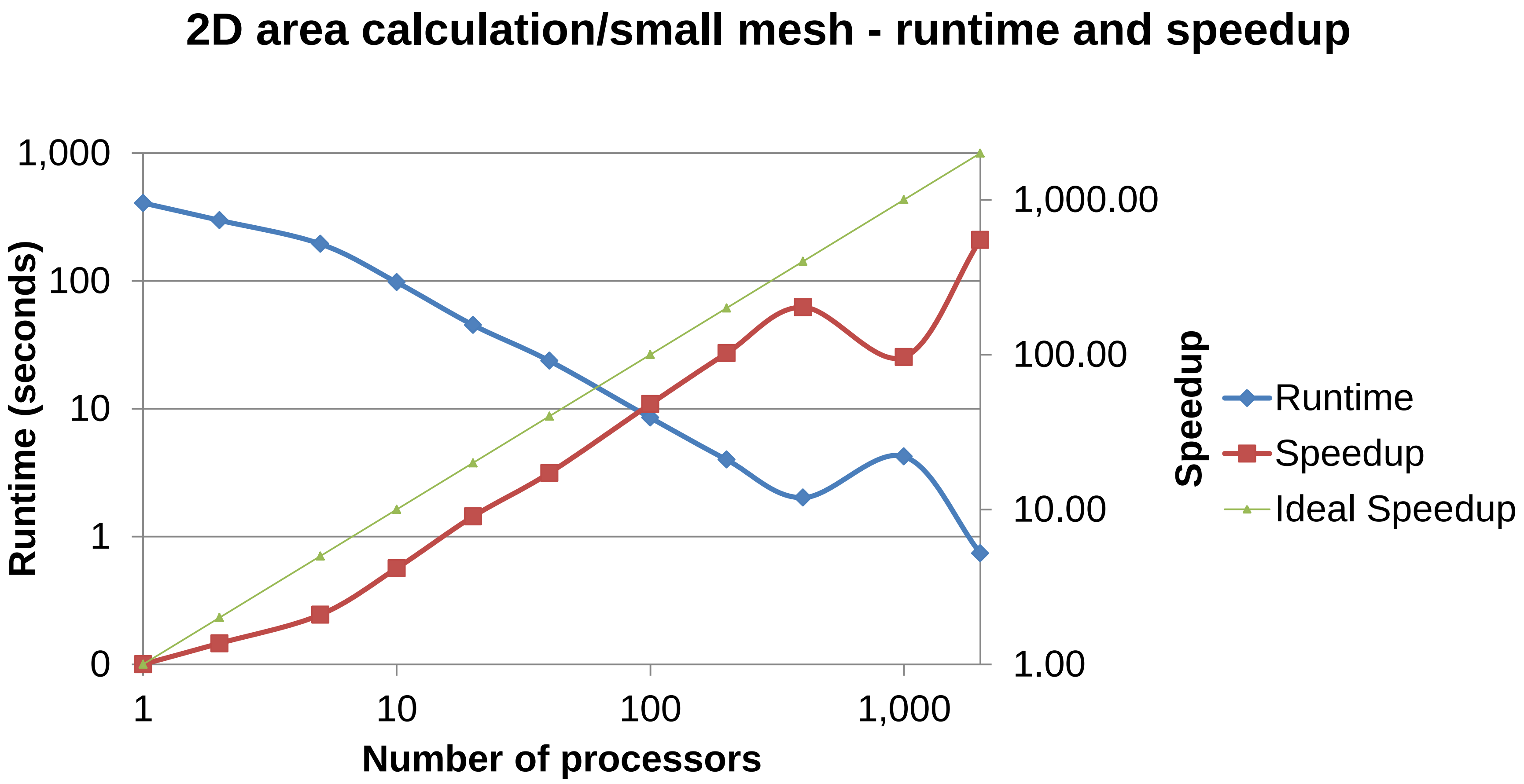
- 297,512,852 elements for mesh A and 264,549,836 elements for mesh B (2D)
- 141,873,169 elements for mesh A and 128,459,529 elements for mesh B (3D)

libsupermesh results (parallel) - benchmarks

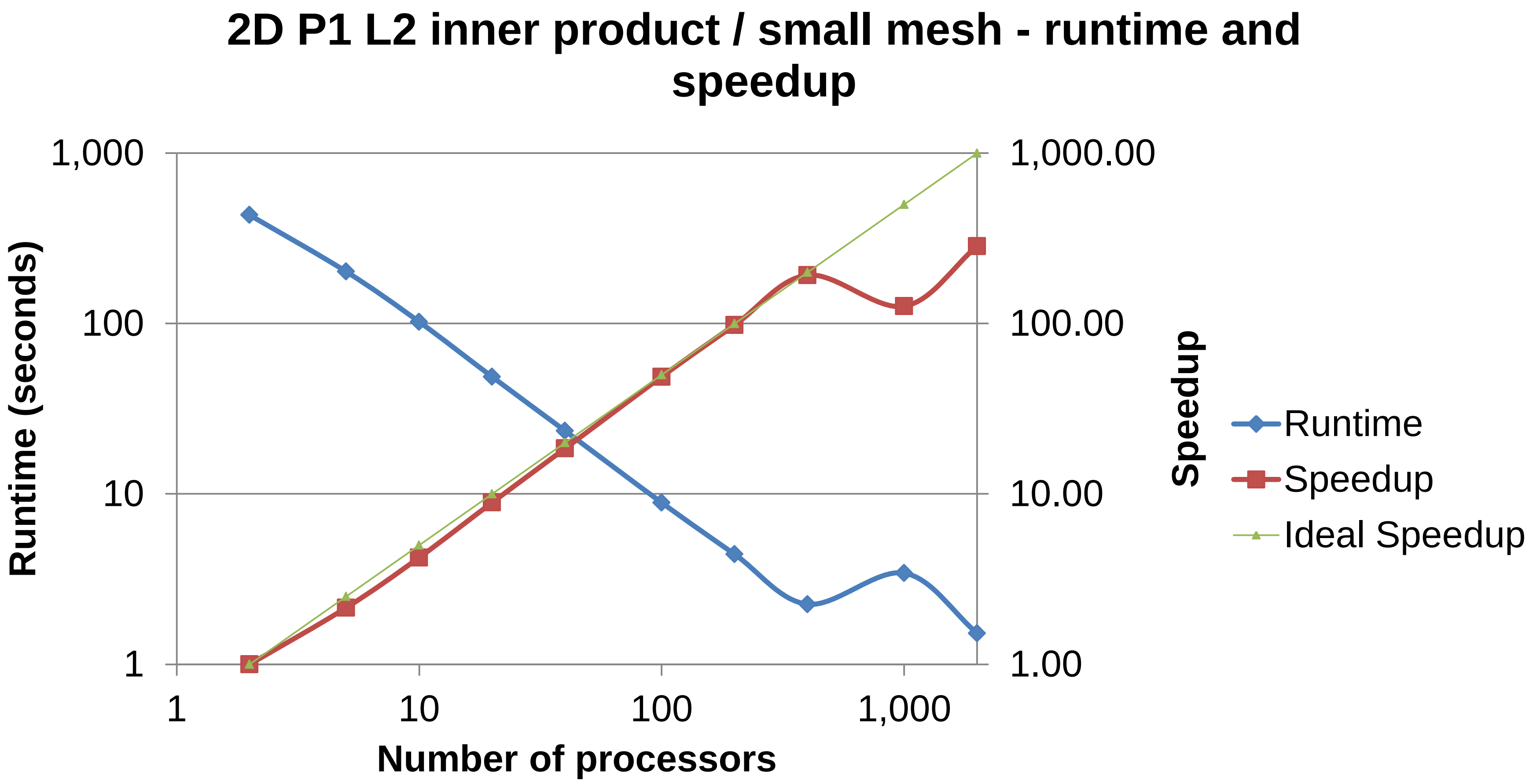
3 different types of benchmarks:

- Compute the area (2D) or volume (3D) of the mesh intersection region
- Compute the area (2D) or volume (3D) of the mesh intersection region and also the L^2 inner product of two piecewise linear continuous functions defined using a standard P1 Lagrange basis
- Compute the area (2D) and also the L^2 inner product of two piecewise quadratic continuous functions defined using a standard P2 Lagrange basis.

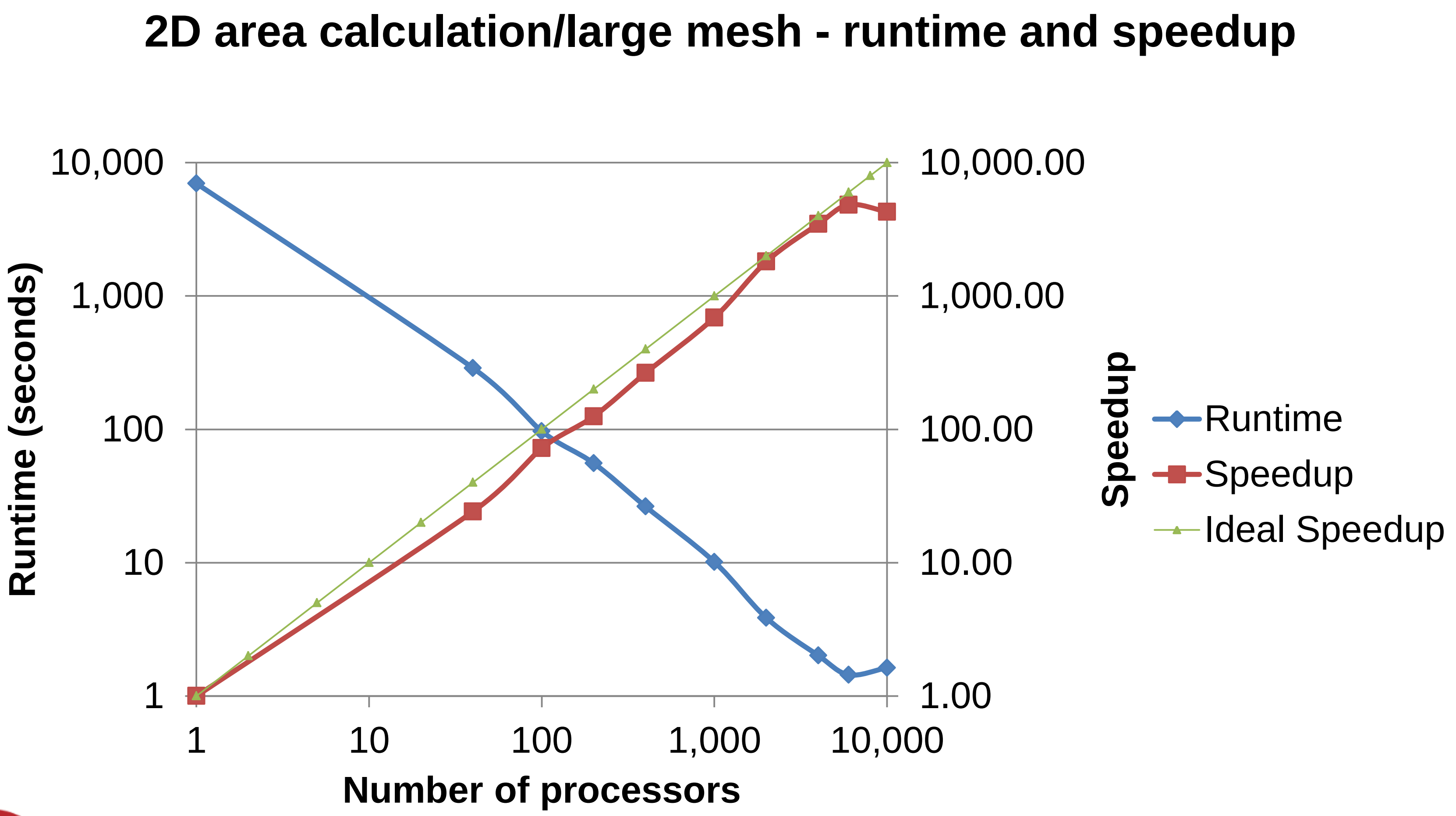
libsupermesh results (parallel) – 2D small



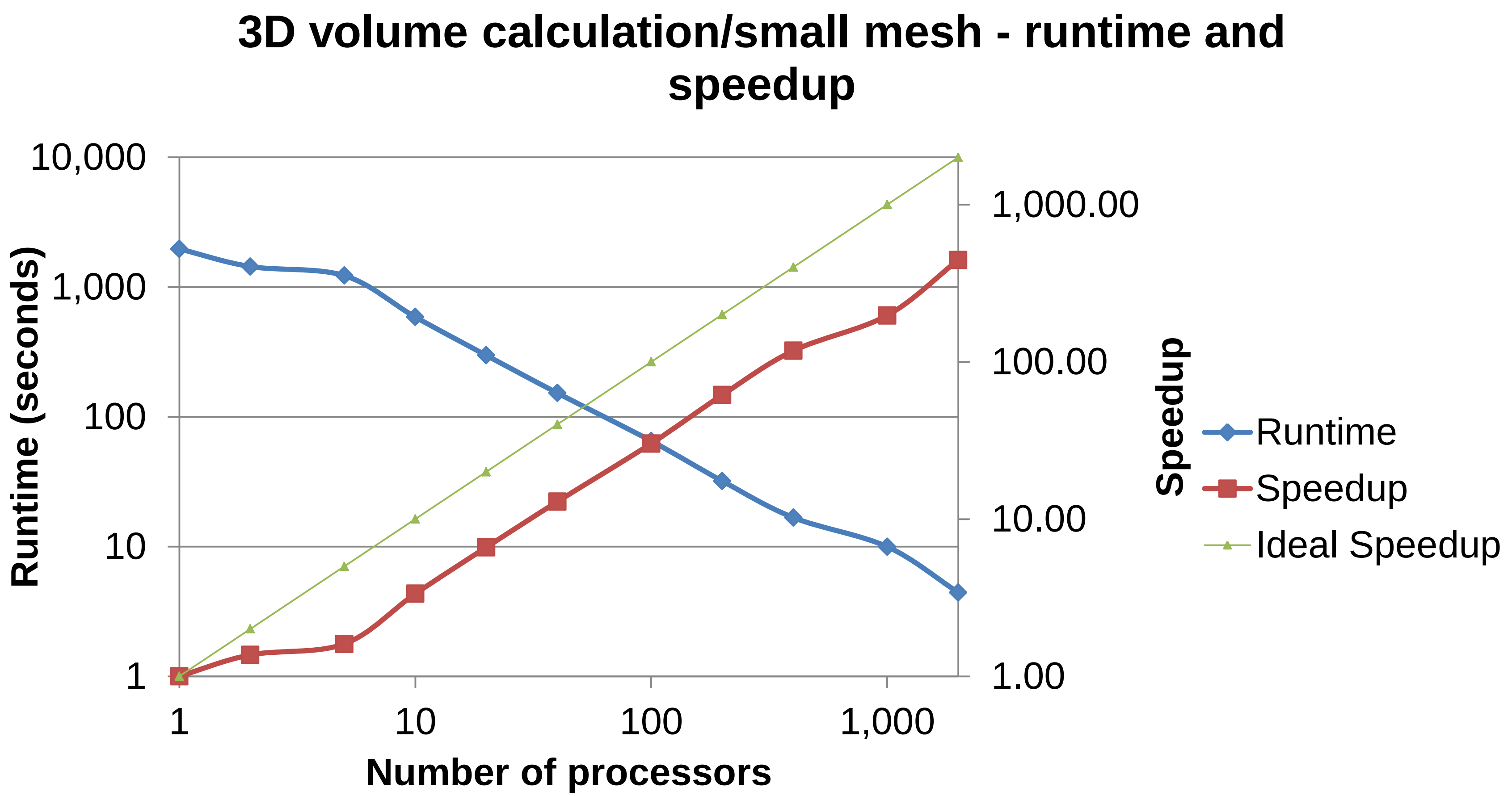
libsupermesh results (parallel) – 2D small



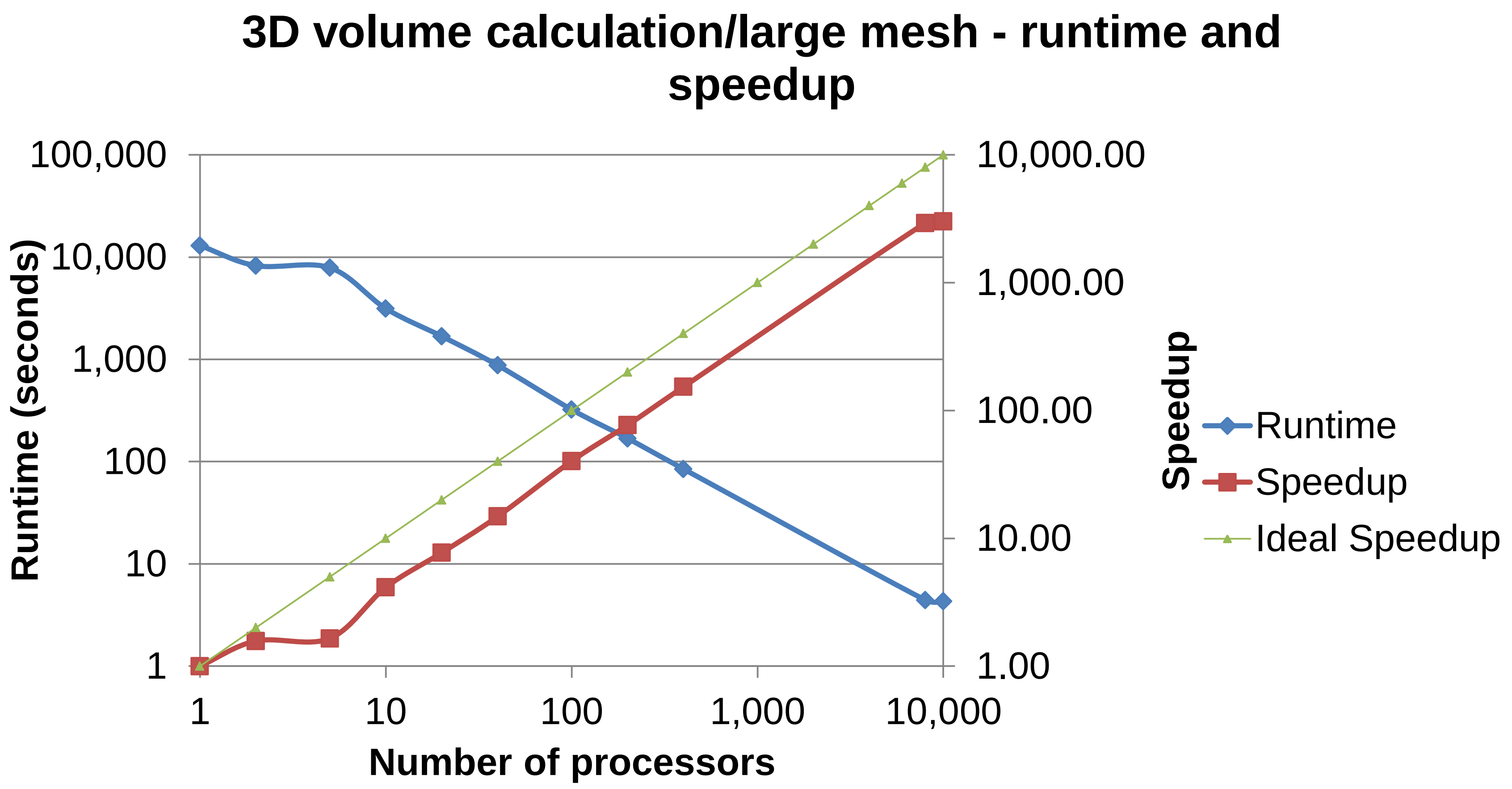
libsupermesh results (parallel) – 2D large



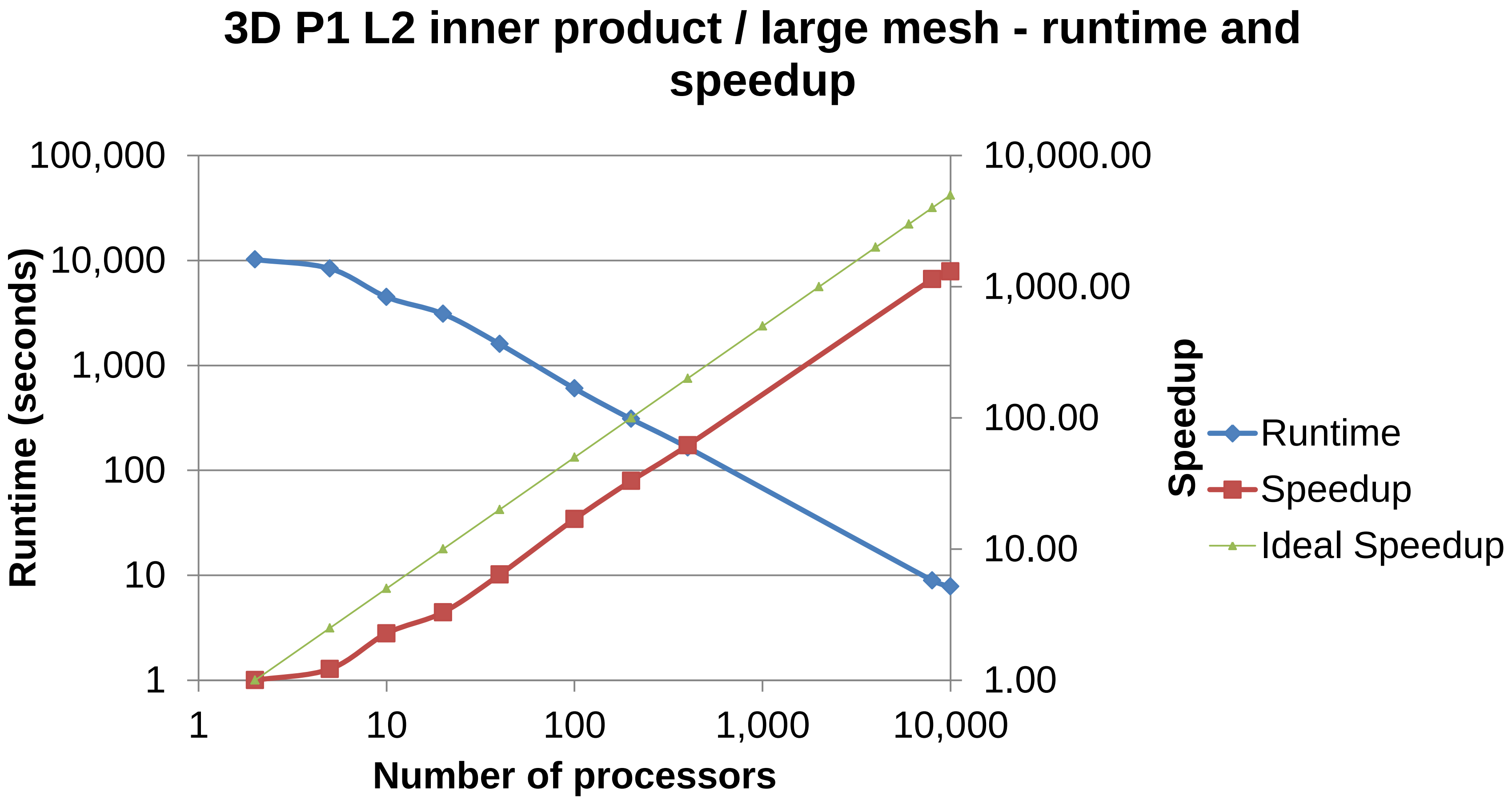
libsupermesh results (parallel) – 3D small



libsupermesh results (parallel) – 3D large



libsupermesh results (parallel) – 3D large



Summary

- Improvements to the serial intersection finder algorithms used by Fluidity
- Implementation of an algorithm for parallel supermeshing, with non-matching domain decompositions
- Parallel general purpose supermeshing library
- libsupermesh is available under an open source license through public repositories
- Optimised and benchmarked; it can scale up to 10,000 cores for a one hundred million degree of freedom problem with acceptable performance
- libsupermesh is available through Fluidity

Thank you

https://fluidityproject.github.io/

https://www.archer.ac.uk/

https://bitbucket.org/libsupermesh/

Questions

Slide title

- Content
  - here

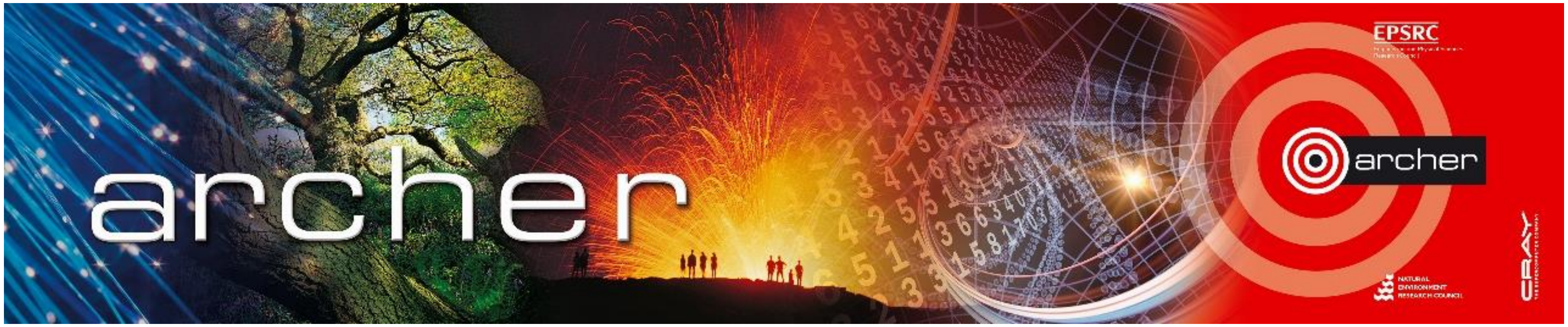
- And more
  - including pictures as well I hope

http://www.archer.ac.uk/training/

- Face-to-face courses
  - timetable, information and registration
  - material from all past courses

- Virtual tutorials
  - timetable plus slides and recordings from past courses
  - please leave feedback on previous tutorials after viewing material

- Technical forum
  - http://www.archer.ac.uk/community/techforum/
  - recordings of previous meetings



Goodbye!

Thanks for attending

Please leave feedback at:
www.archer.ac.uk/training/feedback/online-course-feedback.php