Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

EXCELLERAT

Parallel SFC-based mesh partitioning and load balancing

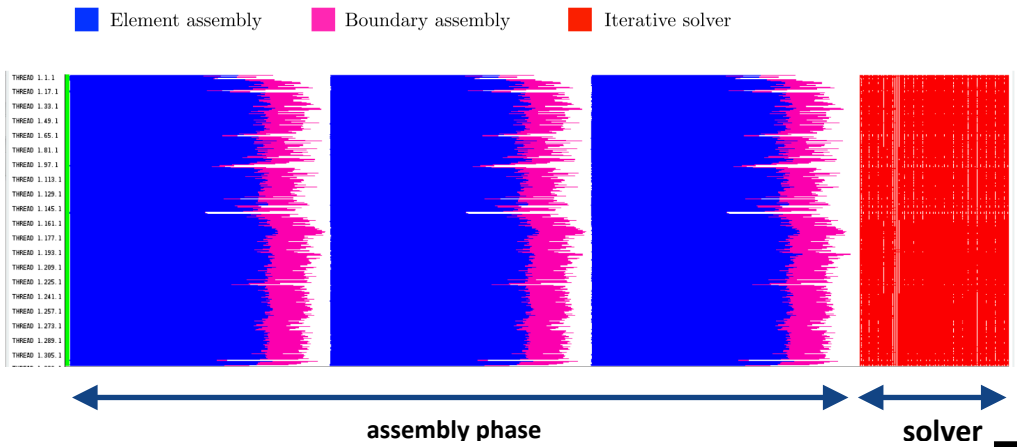
Ricard Borrell, Guillermo Oyarzun, Damien Dosimont and Guillaume Houzeaux

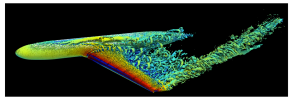
SC19, Denver

CASE department **Barcelona Supercomputing Center**

Load balancing problem

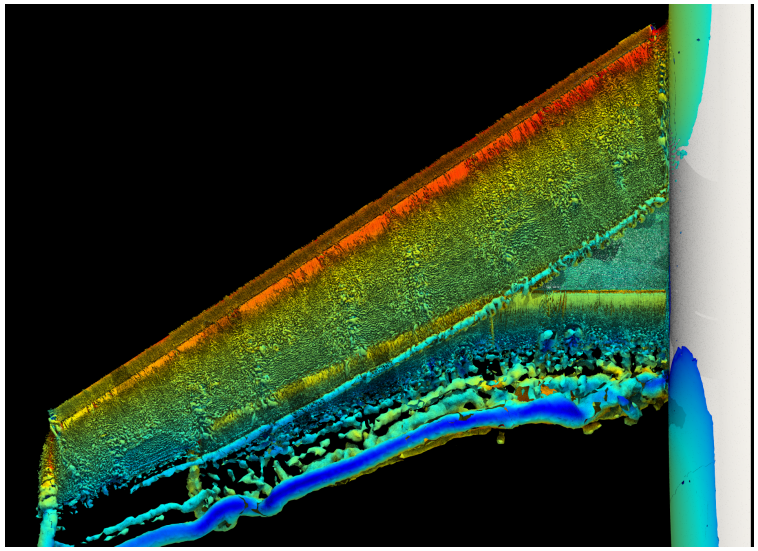
- **Load balance** is critical for parallel efficiency

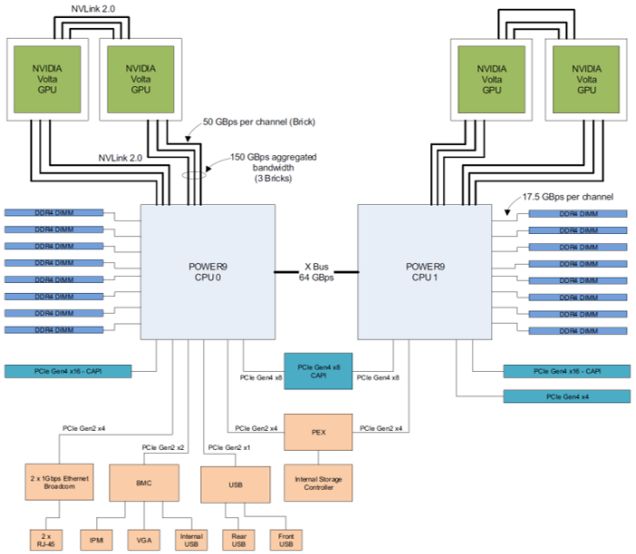




- Sources of imbalance: **heterogeneity** in meshes and hardware
- We aim to cancel “structural” imbalance by modifying underlying mesh partition (complementary to runtime mechanism)

Full Airplane simulation on MareNostrum POWER9 cluster





- **Airplane LES simulation with Alya FE multi-physics code**
- Up to 250M elements heterogeneous unstructured mesh
- Time integration based on a fractional step projection method (Runge-Kutta). Pressure correction with Prec. CG

- **52 compute nodes:**
  - 2 x POWER9 AC922 (20 cores)
  - 4 x Volta V100
  - NVLINK 2.0
  - 512 GB
- Mellanox EDR **interconnect**

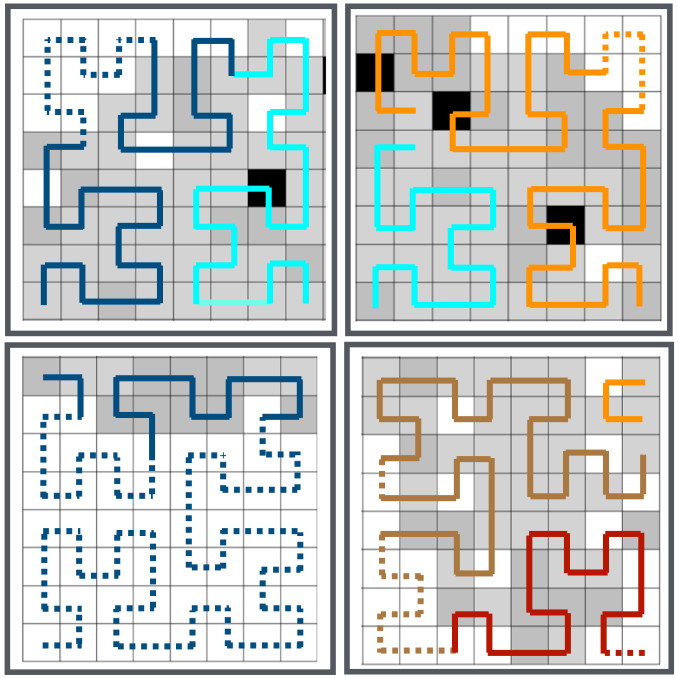
Outline

- SFC-based mesh partitioning
- Load balancing
- Co-execution

SFC-based mesh partitioning

SFC-based mesh partitioning

*

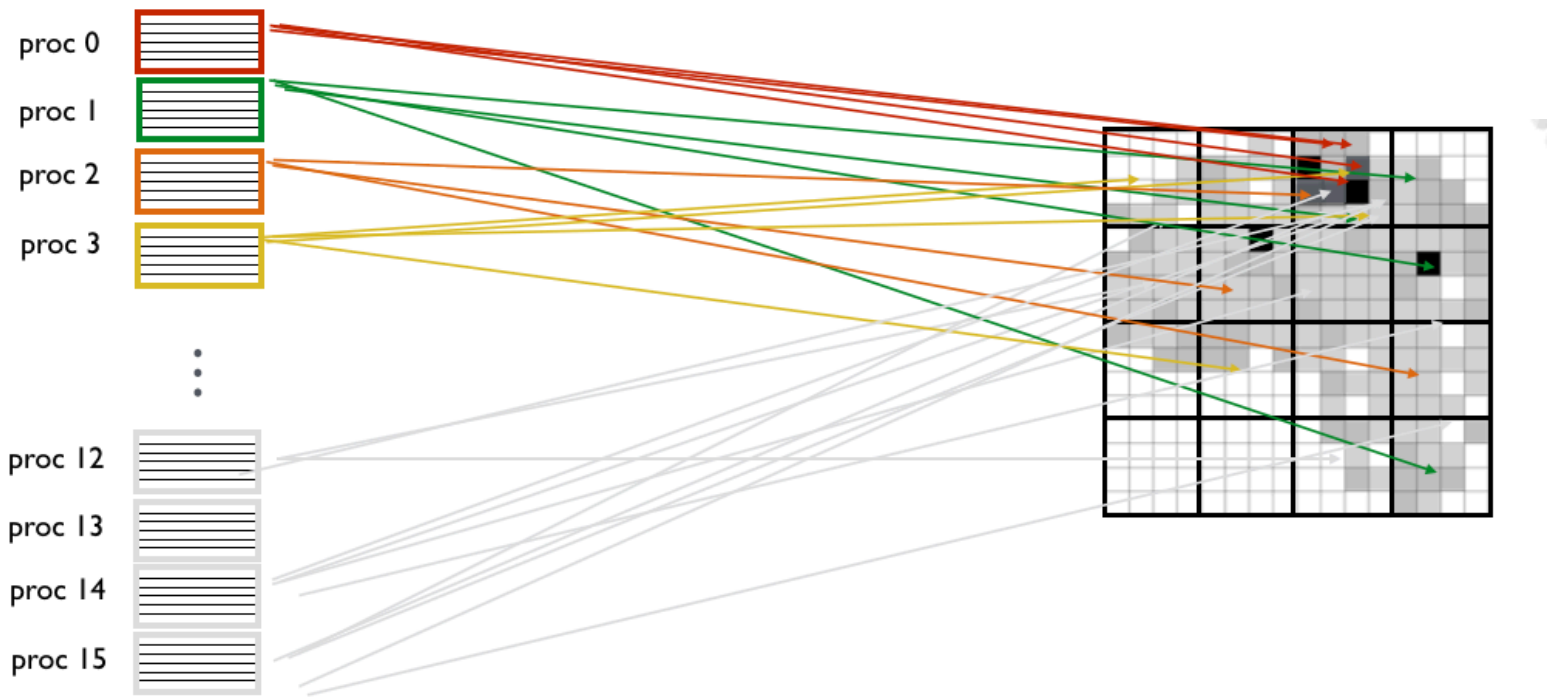


- Each parallel process performs a partition of the mesh contained on its **coarse bin**
- To perform a coherent partition each, partitioner needs to know the weight accumulated on previous coarse bins (according to coarse SFC order)
- Neglecting round-off errors **the resulting partition is independent of the number of processes** used to perform it

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

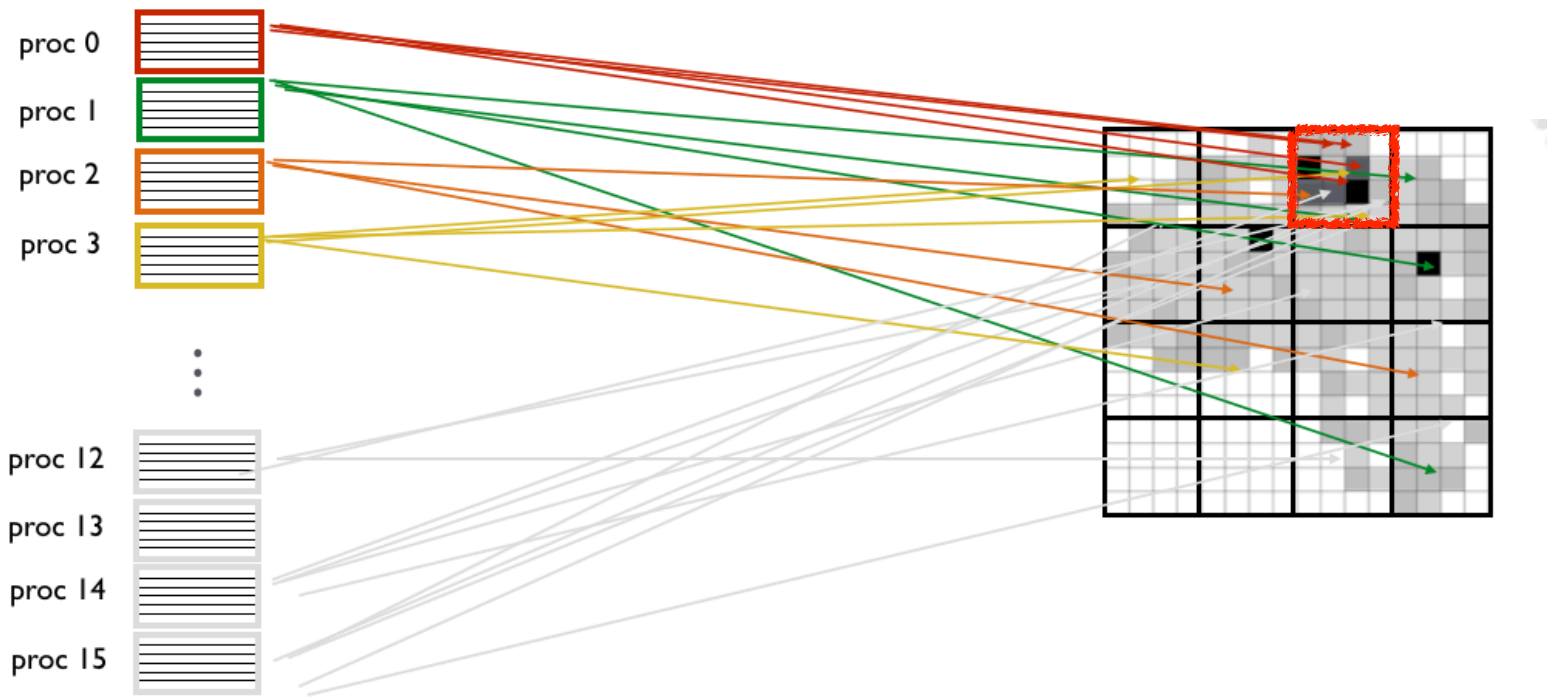
R. Borrell, et al., *Parallel mesh partitioning based on space filling curves*. Computers and Fluids, *2018*.

SFC-based mesh partitioning



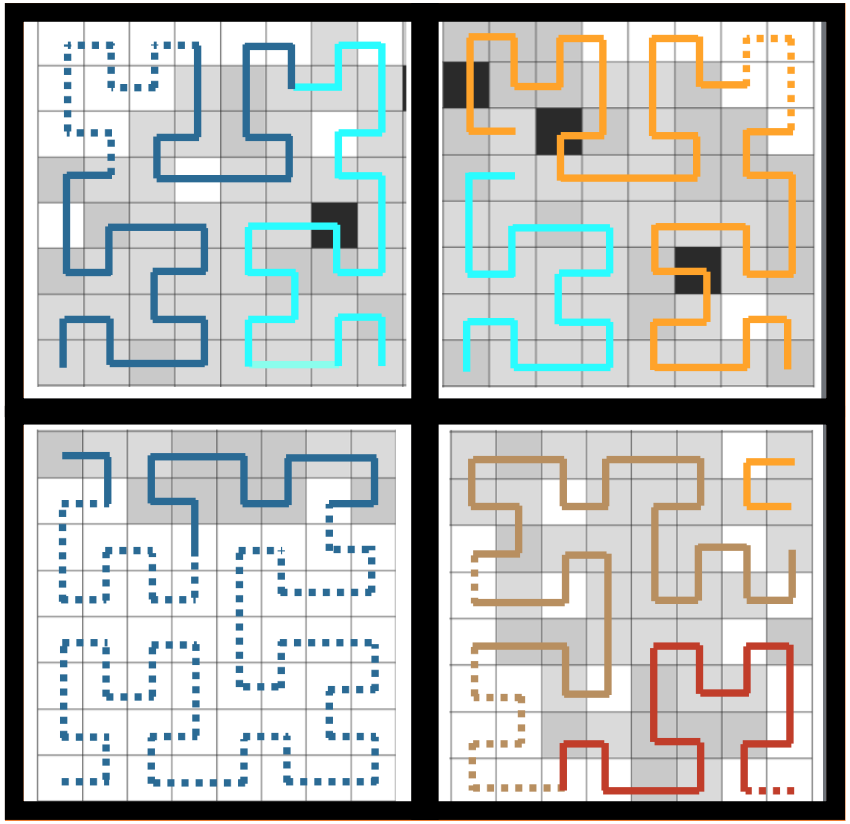
R. Borrell, et al., *Parallel mesh partitioning based on space filling curves*. Computers and Fluids, *2018*.

SFC-based mesh partitioning



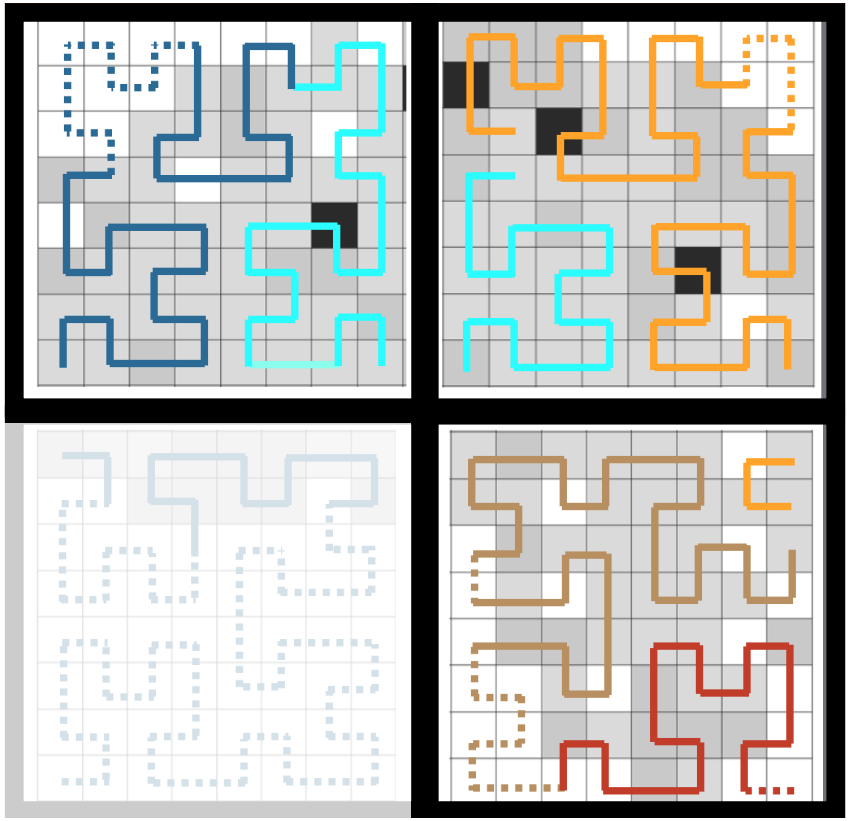
R. Borrell, et al., *Parallel mesh partitioning based on space filling curves*. Computers and Fluids, *2018*.

Optimization: recursive approach



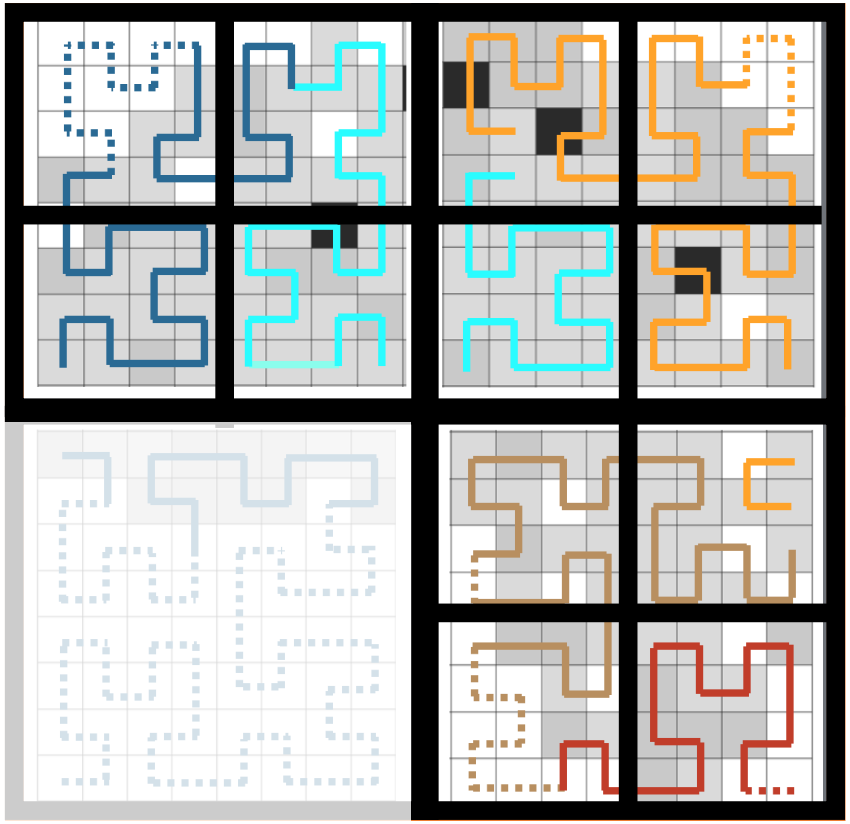
- Recursive process to eliminate the bins that are entirely contained into a single subdomain

Optimization: recursive approach



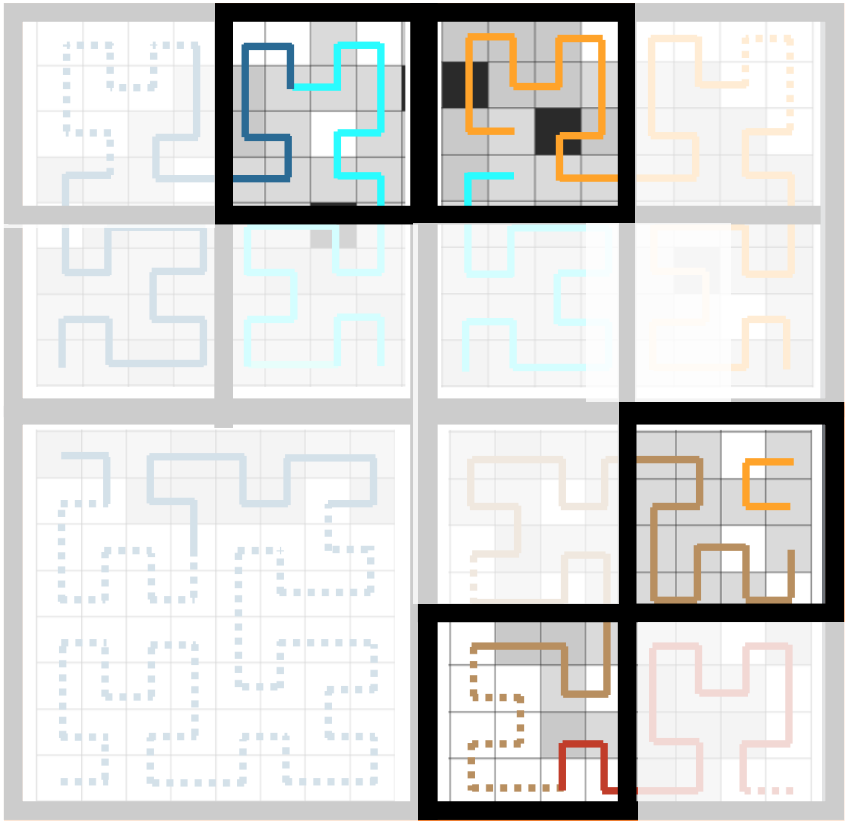
- Recursive process to eliminate the bins that are entirely contained into a single subdomain

Optimization: recursive approach



- Recursive process to eliminate the bins that are entirely contained into a single subdomain

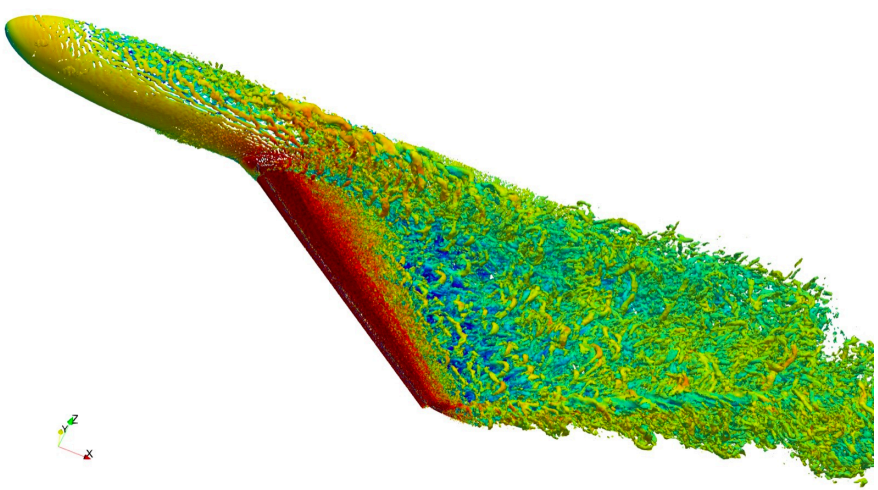
Optimization: recursive approach



- Recursive process to eliminate the bins that are entirely contained into a single subdomain
- **The partition into the remaining boxes can be carried out in parallel with the required level of granularity**

Initial comparison with Zoltan

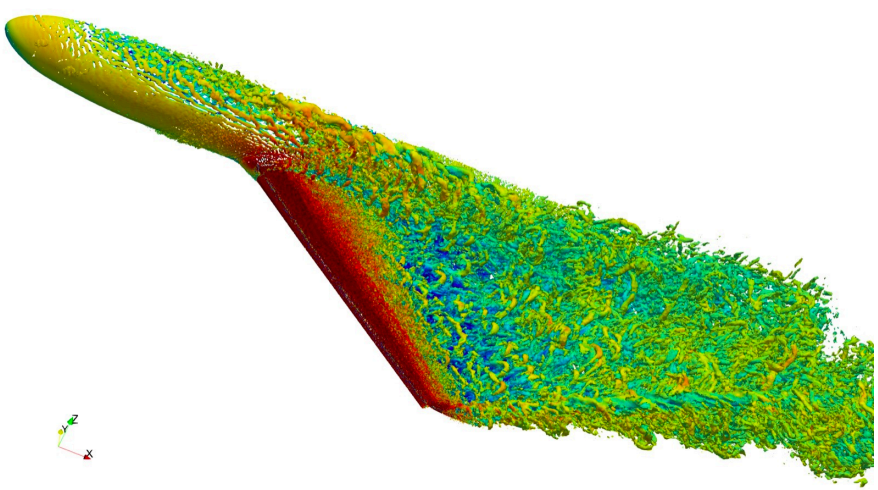
Partitions	Nodes used	LB in-house	LB Zoltan	Time (s) in-house	Time (s) Zoltan	Speedup
384	8	0.99	1.0	0.25	0.87	**3.5x**
768	16	0.99	1.0	0.15	0.54	**3.6x**
1536	32	0.99	1.0	0.10	0.48	**4.8x**
3072	64	0.99	1.0	0.07	0.50	**7.1x**
6144	128	0.99	1.0	0.08	0.79	**9.9x**



- Mesh Airplane 250M elements
- **Zoltan version 3.8.3** - Zoltan_LB_Partition with HSBC option
- Both Alya and Zoltan compiled with intel/18.4
- Tests performed on the MareNostrum IV supercomputer

Initial comparison with Zoltan

Partitions	Nodes used	LB in-house	LB Zoltan	Time (s) in-house	Time (s) Zoltan	Speedup
384	8	0.99	1.0	0.25	0.87	**3.5x**
768	16	0.99	1.0	0.15	0.54	**3.6x**
1536	32	0.99	1.0	0.10	0.48	**4.8x**
3072	64	0.99	1.0	0.07	0.50	**7.1x**
6144	128	0.99	1.0	0.08	0.79	**9.9x**

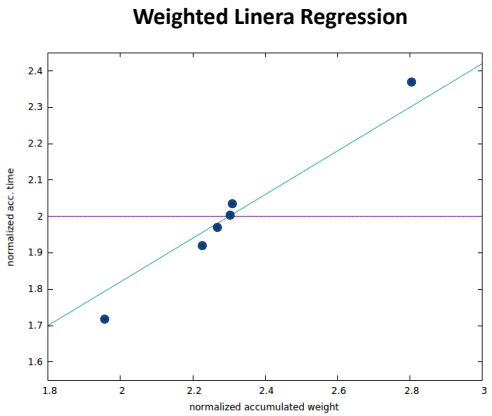


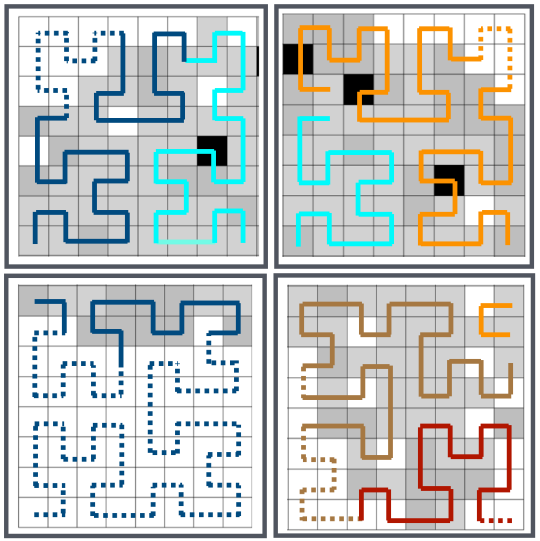
- Mesh Airplaine 250M elements
- Zoltan version 3.8.3 - Zoltan_LB_Partition with HSBC option
- Both Alya and Zoltan compiled with intel/18.4
- Tests performed on the MareNostrum IV supercomputer

Load Balancing

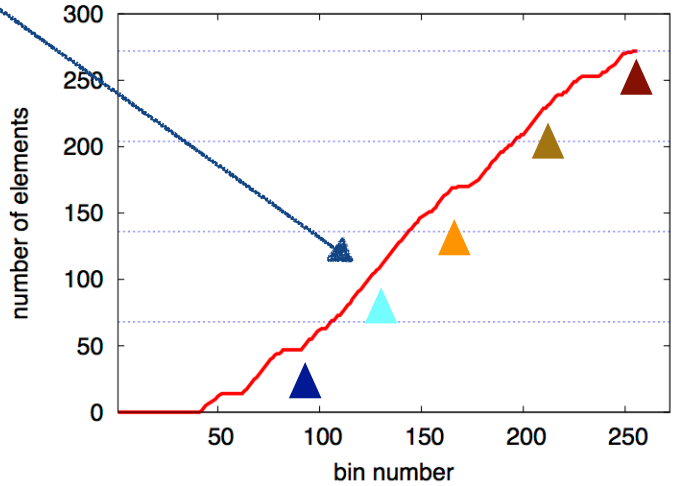
LB strategy

- Finding accurate model for partition is complex
- Many factors influence computing cost of each element: type, connectivity with other elements and data locality, execution device





*

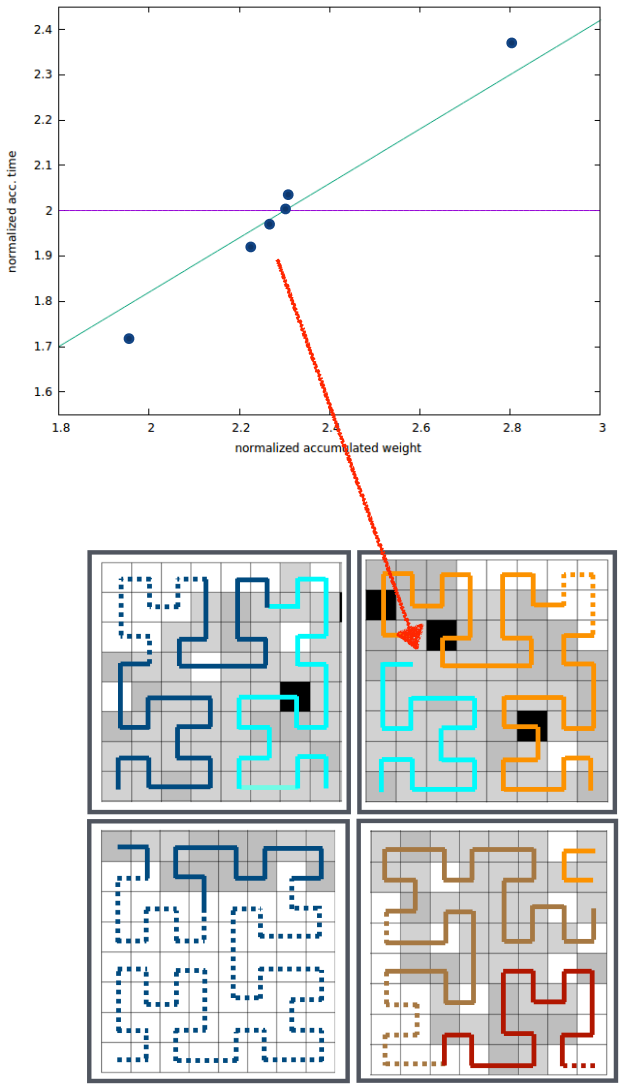


- **We define a (re-)partition according to runtime measurements: online approach**
- We define each splitting point of the SFC independently by using a linear regression from measurements of partitions

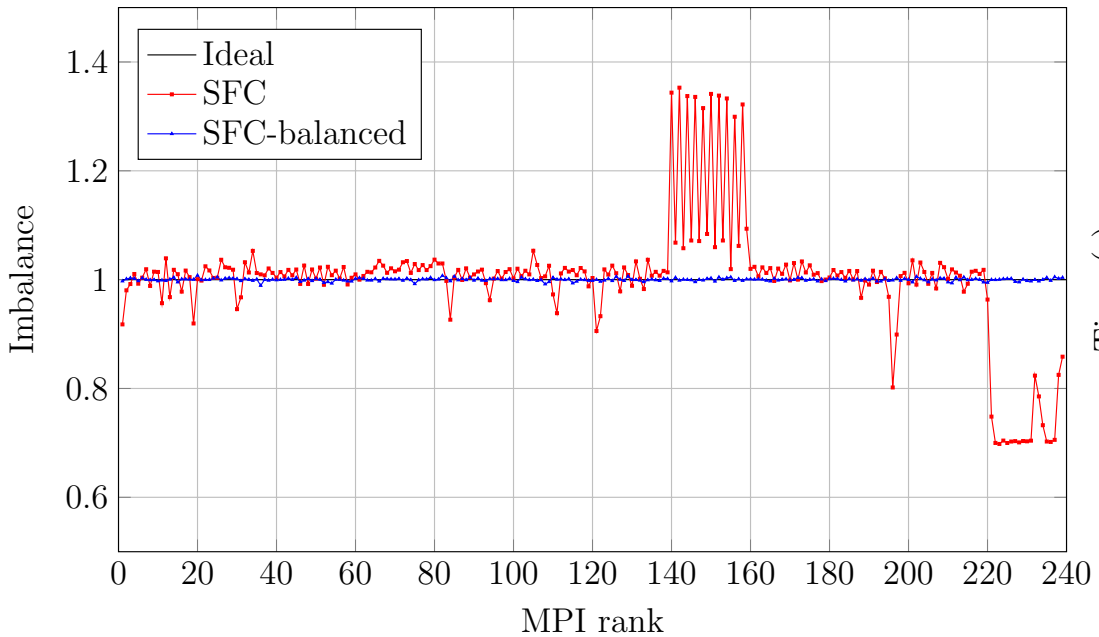
LB strategy

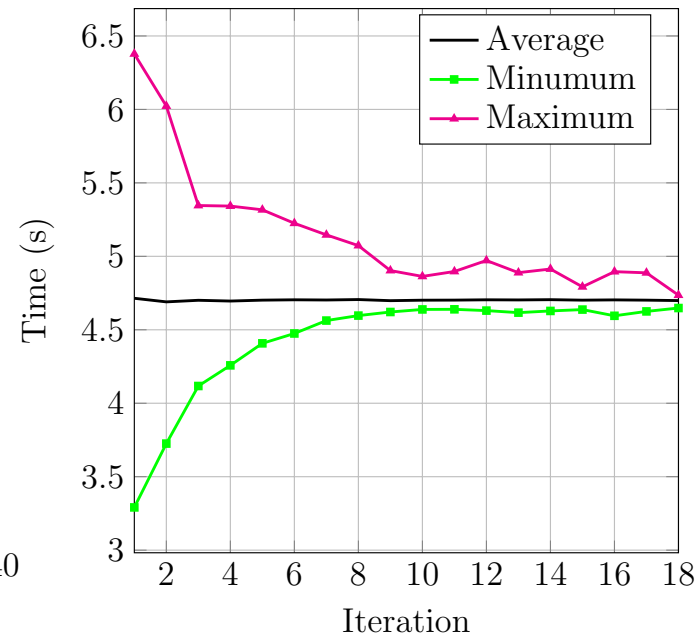
- We define **each splitting point of the SFC independently** by using a linear regression from measurements of partitions
  - **Independent variable**: Load accumulated until the i'th splinting point
  - **Dependent variable:** Sum of elapsed time spent by the i'th first MPI-processes
    - ✓ For executions with only one type of device this is almost independent of the load distribution between the i'th first parallel processes

Weighted Linera Regression



Load balancing on the POWER9 AC922 CPUs





$$I_k = \frac{t_k}{\bar{t}}$$

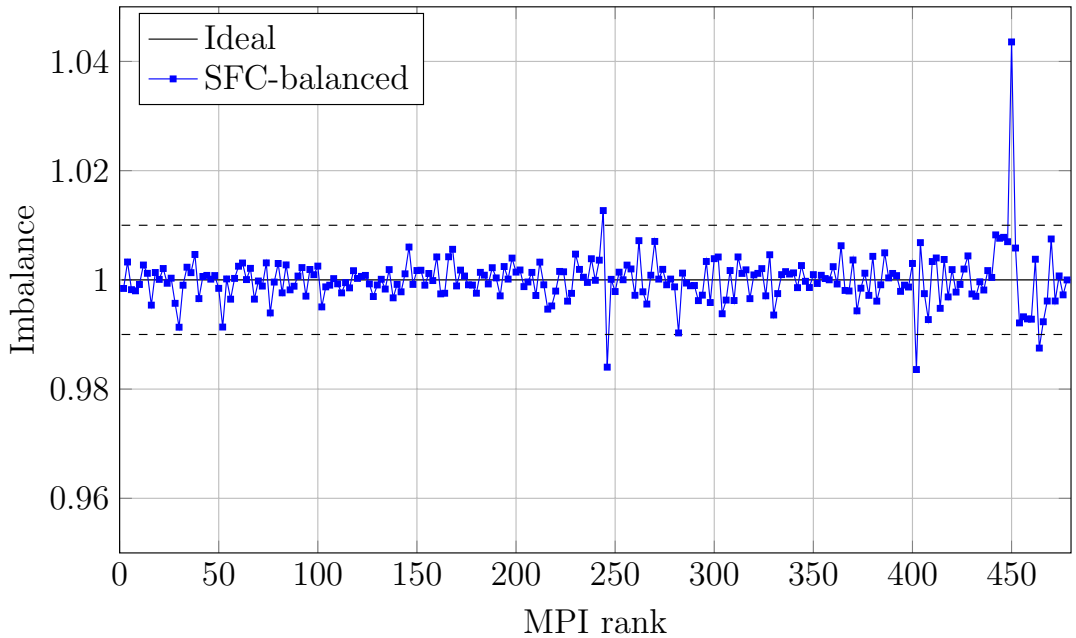
time *kth* MPI process

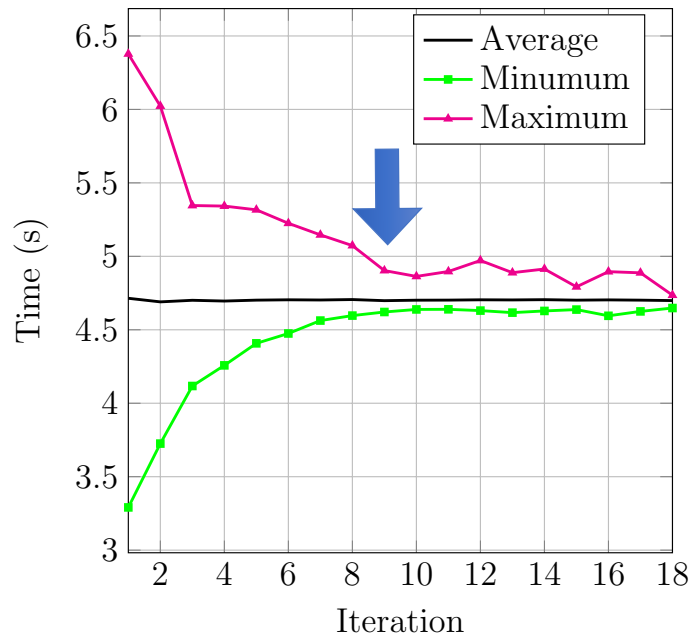
average time

- Full Airplane LES simulation
- **Only assembly phase considered** (not solver)
- **176 M elements mesh**
- **12 nodes (**480 CPU-cores)
- **2 OmpSs threads per MPI-process**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Load balancing on the POWER9 AC922 CPUs

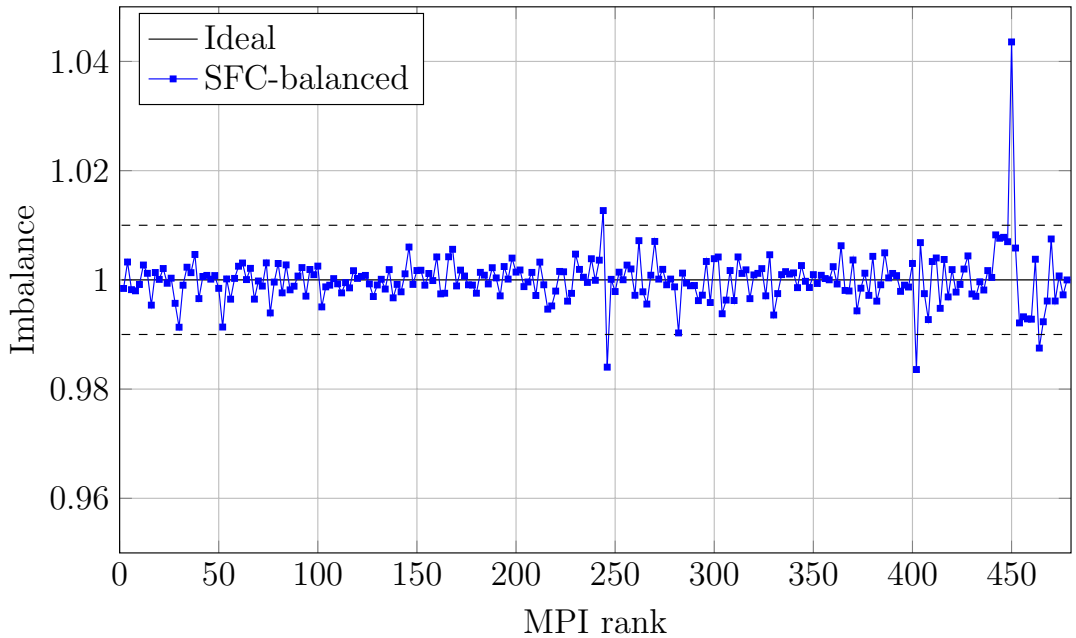


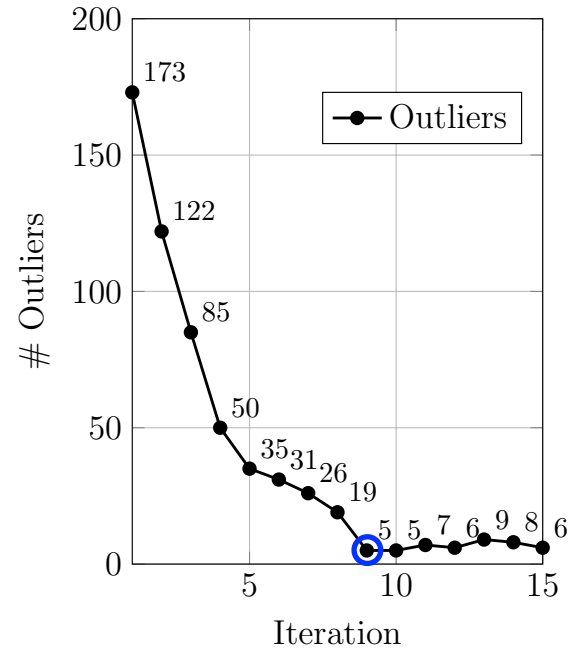


- Full Airplane LES simulation
- **Only assembly phase considered** (not solver)
- 176 M elements mesh
- **12 nodes (**480 CPU-cores)
- **2 OmpSs threads per MPI-process**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Load balancing on the POWER9 AC922 CPUs



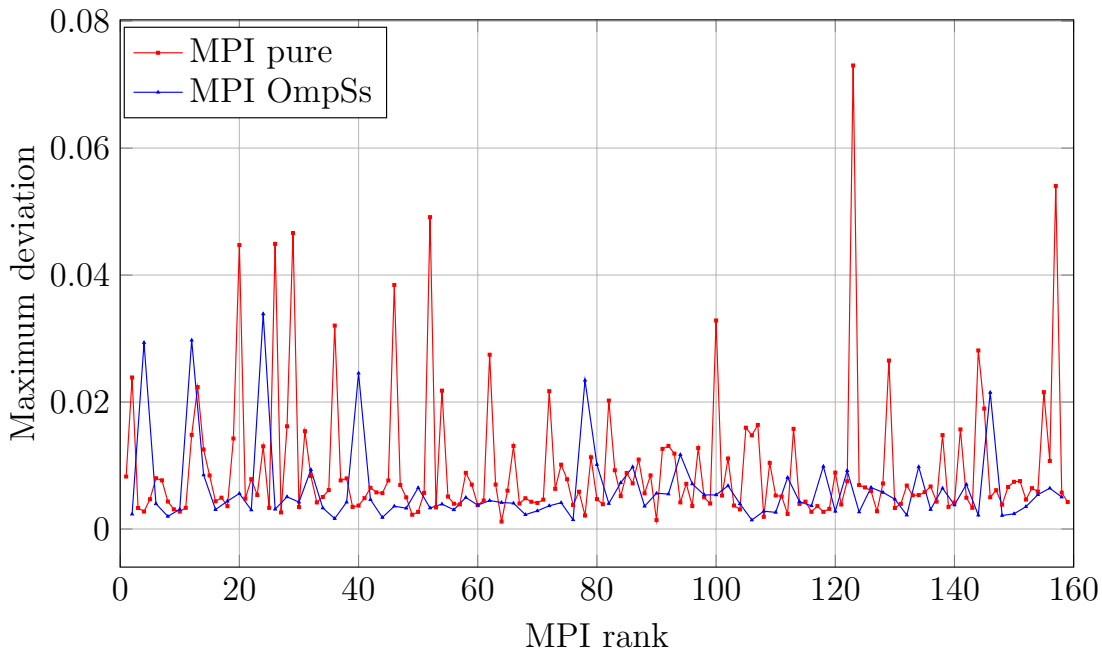


- At certain point, outlier time measurements appear randomly on different MPI - processes
- This situation shows up when hardware noise becomes significant
- We observed a notable noise on the MareNostrum P9 cluster

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

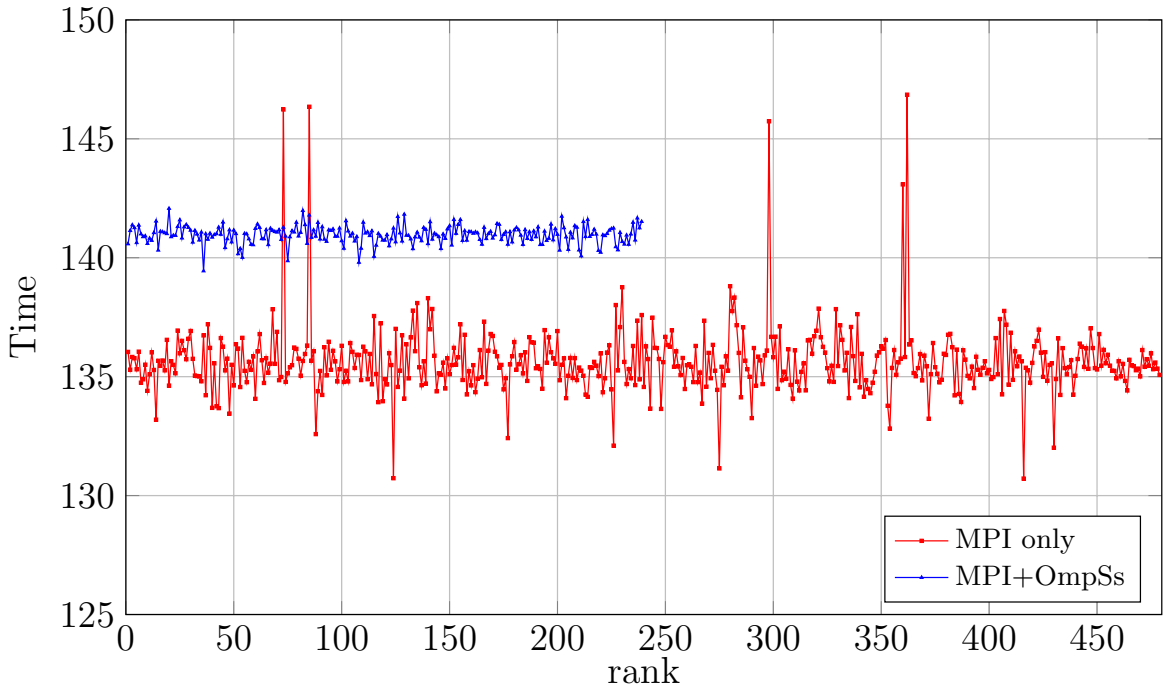
Load balancing on the POWER9 AC922 CPUs

- Maximum deviation with respect to the average for 10 identical executions (same nodes and partition)



- Full Airplane LES simulation
- Only **assembly phase** considered (not solver)
- 30 M elements mesh
- 4 nodes (160 CPU-cores)

Load balancing on the POWER9 AC922 CPUs



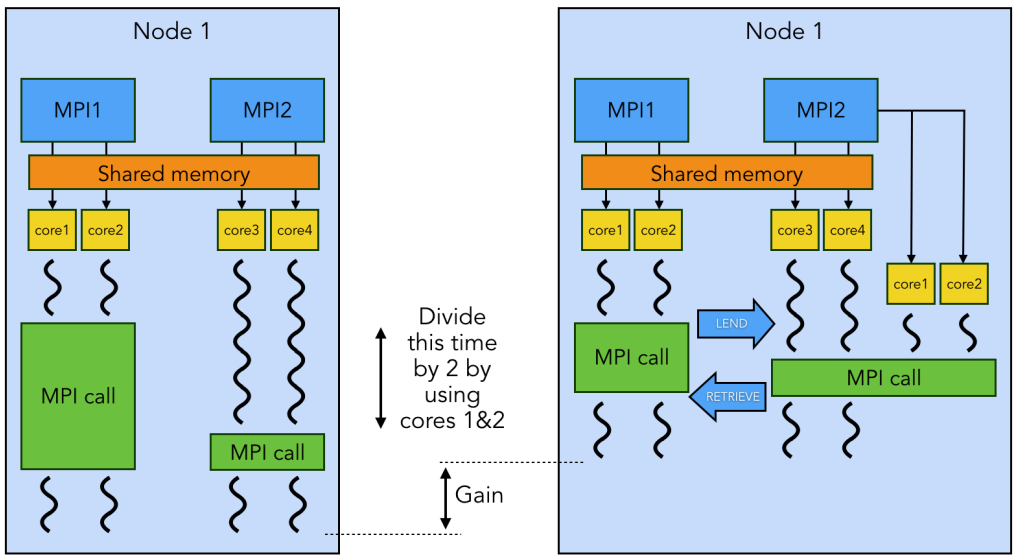
- MPI + OmpSs performs better than pure-MPI because its more resilient to system noise

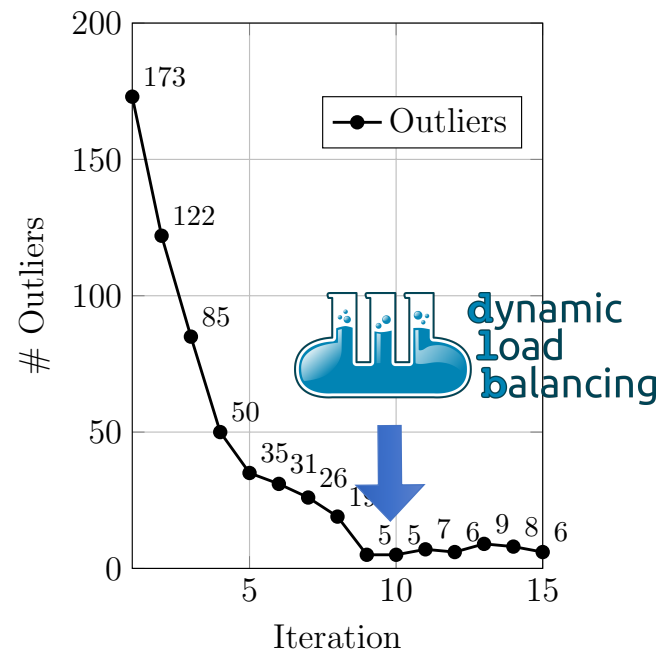
- Full Airplane LES simulation
- Only **assembly phase** considered (not solver)
- 176 M elements mesh
- 12 nodes (480 CPU-cores)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Load balancing on the POWER9 AC922 CPUs

- Next step: add an DLB as an additional runtime mechanism





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

https://pm.bsc.es/dlb

Multilevel approach

	Approach	Scope
SFC-based mesh repartitioning	Corrects partition according to time measurements from previous iterations	Distributed memory
DLB	Reacts instantaneously by launching threads into idle resources	Shared memory
OmpSs	Dynamically distributes work between threads	Shared memory

- The three levels are complementary
- Correcting the underlying mesh partition (level1) increases the efficiency and reduces overhead of the rest of the instantaneous runtime mechanism

CPU-GPU co-execution

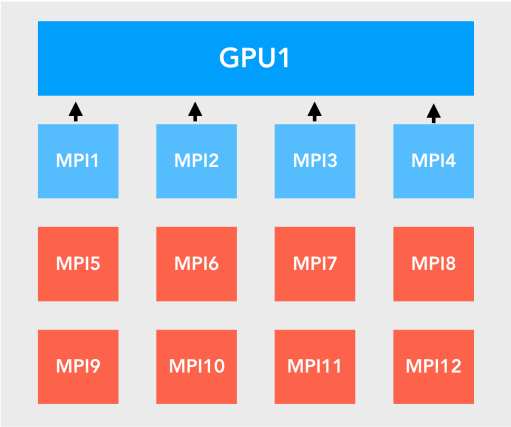
Multi-code approach

- job.sh

```
srun –multi-prog input.conf
```

- input.conf:

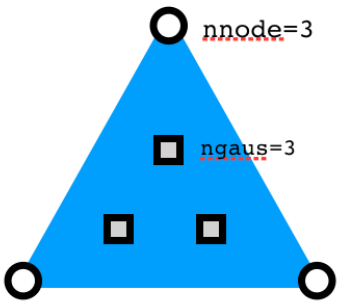
```
0-7  alya_cpu.x airplaine
8-11 alya_gpu.x airplaine
```



- A specific executable generated for each device (from same code) using optimal compiler and/or compiling options
- Work load distributed based on SFC approach

Unified code: data restructuing

- Number of nodes and Gauss points nnode, ngaus
- Shape function N(nnode,ngaus)
- Jacobian x Gauss point weights Jac(PACK_SIZE,ngaus)
- Element matrix Ae(PACK_SIZE,nnode,nnode)




```
do ig = 1,ngaus
  do jn = 1,nnode
    do in = 1,nnode
      Ae(in,jn) = Ae(in,jn) + Jac(ig) * N(in,ig) * N(jn,ig)
    end do
  end do
end do
```

Assembly of PACK_SIZE elements: help the compiler

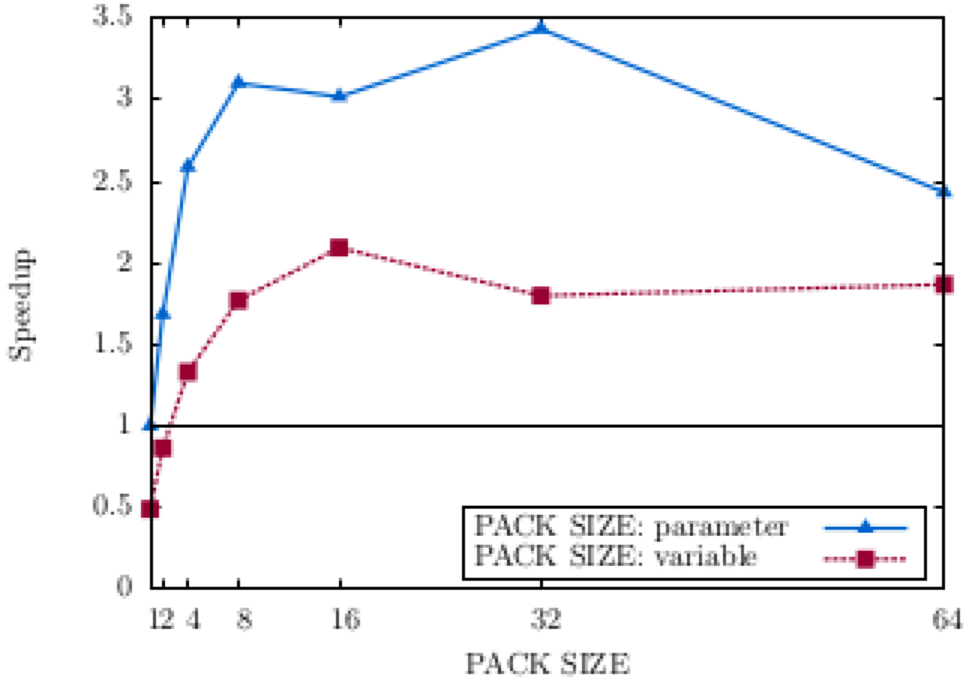
```
do ig = 1,ngaus
  do jn = 1,nnode
    do in = 1,nnode
      Ae(1:PACK_SIZE,in,jn) = Ae(1:PACK_SIZE,in,jn) + Jac(1:PACK_SIZE,ig) * N(in,ig) * N(jn,ig)
    end do
  end do
end do
```

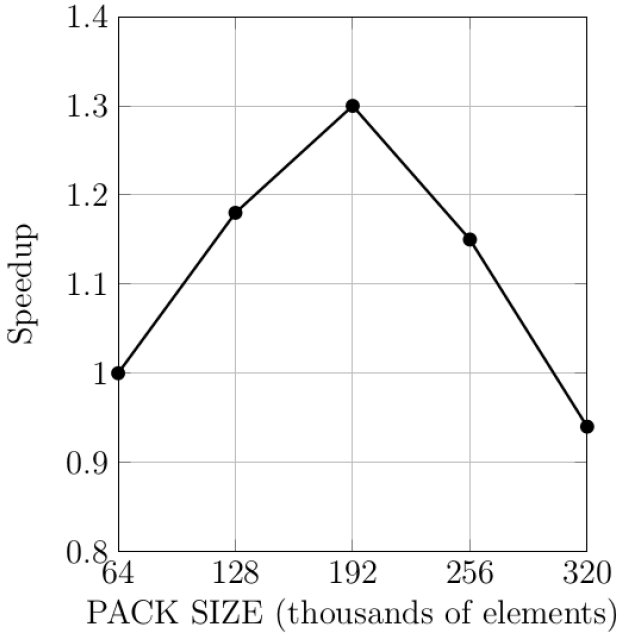
- New dimension created on the arrays, to groups elements of the same type that can be computed simultaneously without race condition
- Used with different granularity (PACK_SIZE) for SIMD in CPU or SIMT in GPU
- PACK_SIZE can be decided at compile time or be a variable of the program.

Unified code: directives

```
#ifdef OPENACC
#define DEF_PACK ipack
   use openacc
#else
#define DEF_PACK 1:PACK_SIZE
#endif

#ifdef OPENACC
!$acc data create(…)  &
!$acc copyin(…)
!$acc parallel loop gang vector default(present)
do ipack = 1,VECTOR_SIZE
#endif

      do ig = 1,ngaus
        do jn = 1,nnode
          do in = 1,nnode
            Ae(DEF_PACK,in,jn) = Ae(DEF_PACK,in,jn) + Jac(DEF_PACK,ig) * N(in,ig) * N(jn,ig)
          end do
        end do
   end do

#ifdef OPENACC
end do
!$acc end parallel loop
!$acc end data
#endif
```

PACK_SIZE

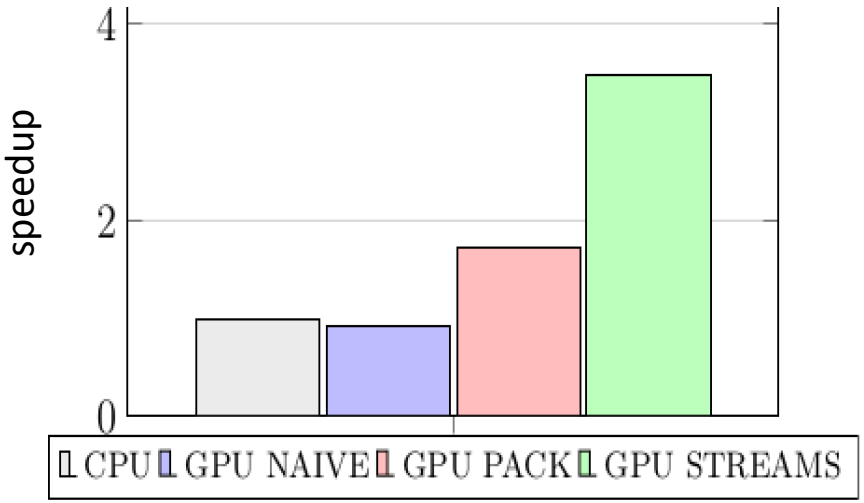




Speed up for various values of PACK SIZE. Left: POWER9 CPU AC922, Right: Nvidia Volta V100 GPU

CPU/CPU speedup

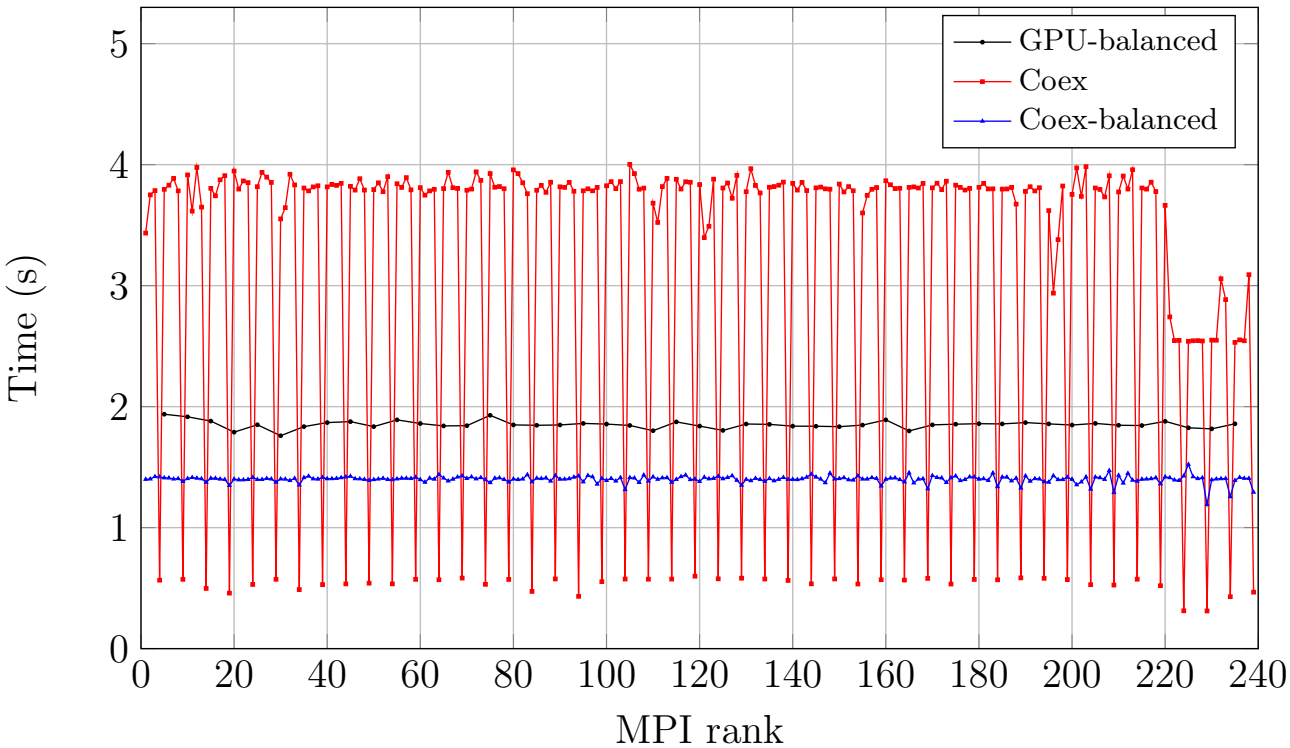
- Baseline CPU performance for all node (40 CPU-cores) compiled with GCC 8.1.0
- Similar implementation with openACC (-8% improvement) using PGI 18.5
- Tuning VECTOR_SIZE from Alya determines number of launches of the kernel (80% improvement with respect to previous)
- Adding concurrency to increase the occupancy of the kernel with streams (102% improvement with respect to previous)

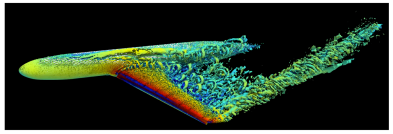


Airplane simulation 32M elements mesh

Power9 node: 2x Power9 20 cores + 4 x Volta V100 GPU

Load balancing on co-execution



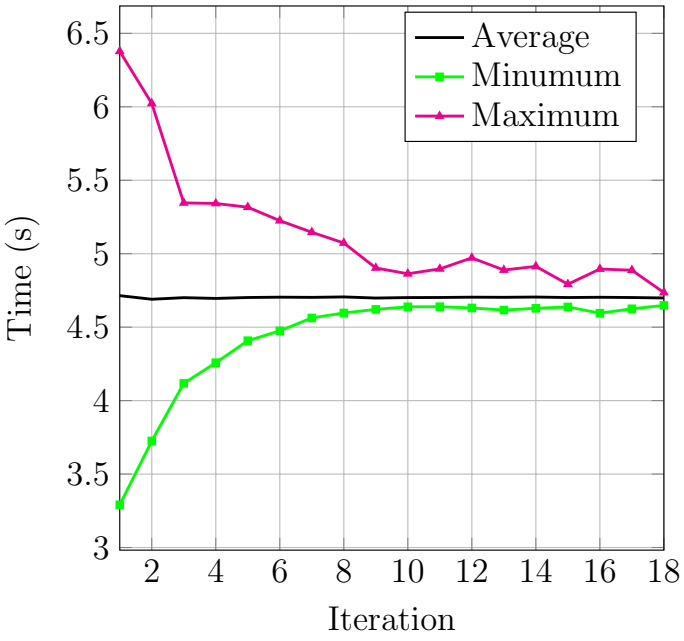


23%

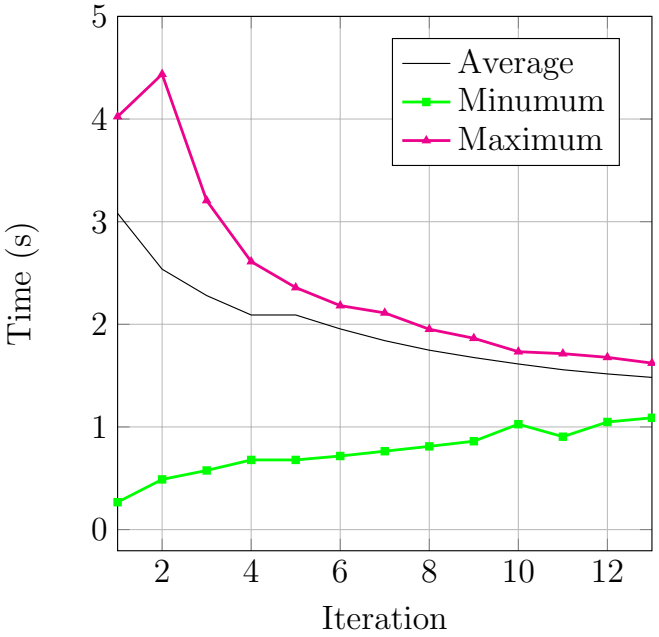
- A well-balanced co-execution is 23% faster than a well-balanced pure GPU execution
- The gain is equivalent than disposing of an additional GPU per node

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Load balancing in co-execution



Pure CPU



Co-execution

Concluding remarks

Concluding remarks

- We have presented an in-house SFC based mesh partitioner
- The efficiency of the partitioner (250M mesh ~0.1 sec) makes it suitable for dynamic load-balancing
- We have presented a robust load balancing strategy based on linear regression to define the SFC splitting points
- We have demonstrated the performance of the load-balancing strategy for an Airplane simulation on the heterogeneous MareNostrum POWER9 cluster
- **Next steps:**
  - Validate all these developments at a larger scale
  - Release an open-source version of the SFC-based mesh partitioner

ricard.borrell@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

THANK YOU!

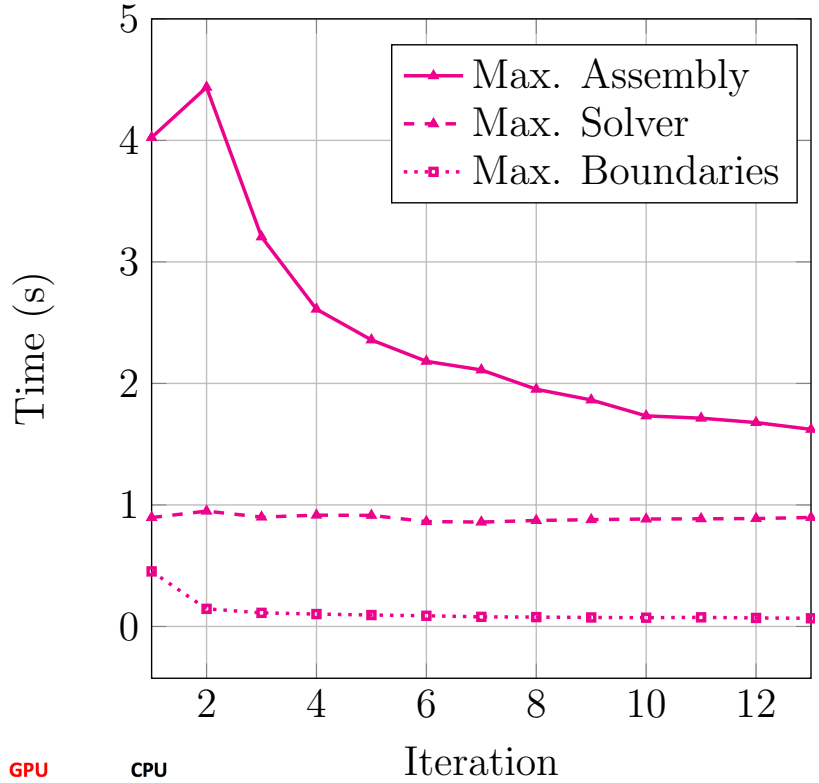
ricard.borrell@bsc.es

EXCELLERAT

www.bsc.es

Appendix: simulation phases

Load balancing in co-execution

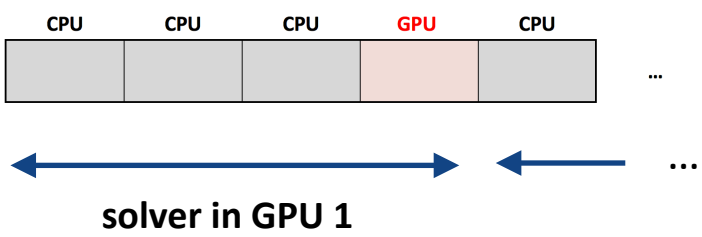


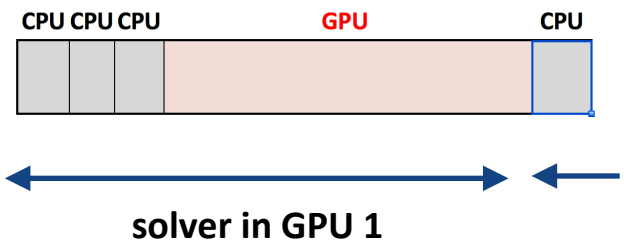
time reduction

60%

-0.1%

85%

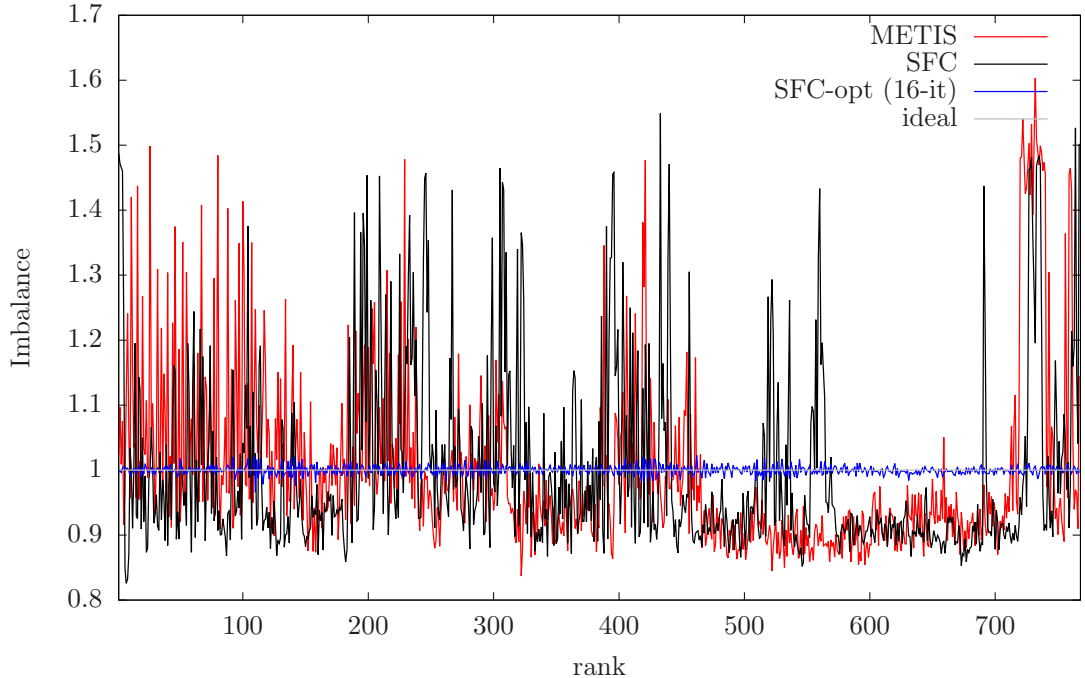


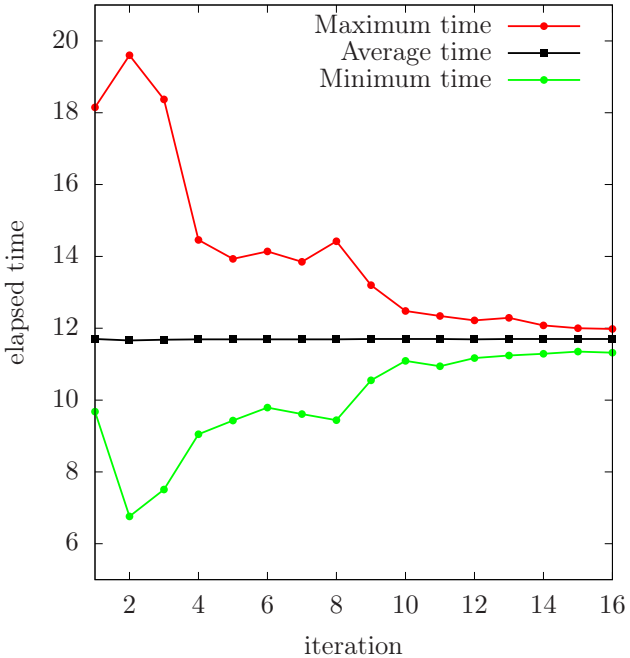


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

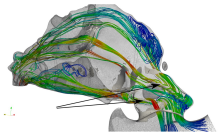
Appendix: comparison with METIS

Comparison with METIS



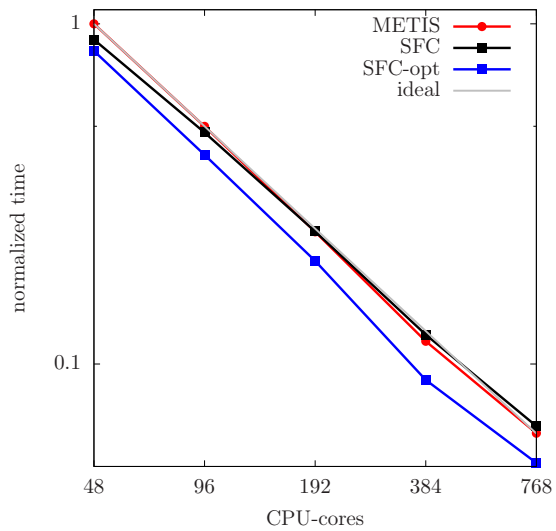


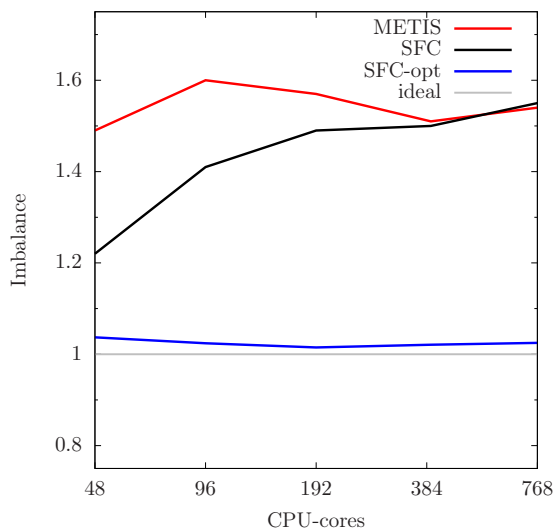
Respiratory system 23.8M - 768 CPU-cores (16 nodes MN4)

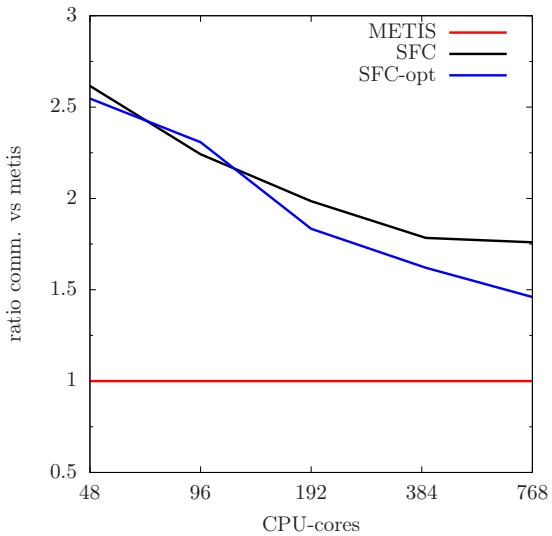


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Comparison with METIS

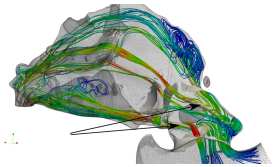






up to 23% time reduction!

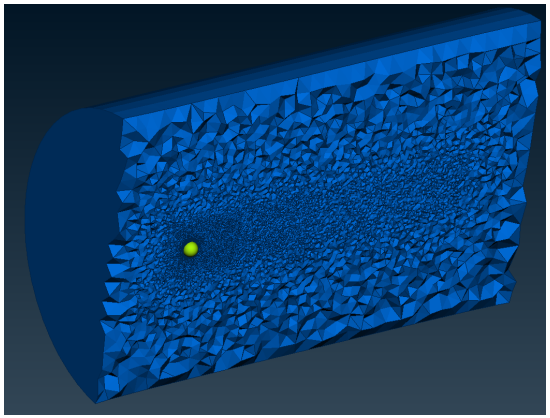
Respiratory system 23.8M - time step

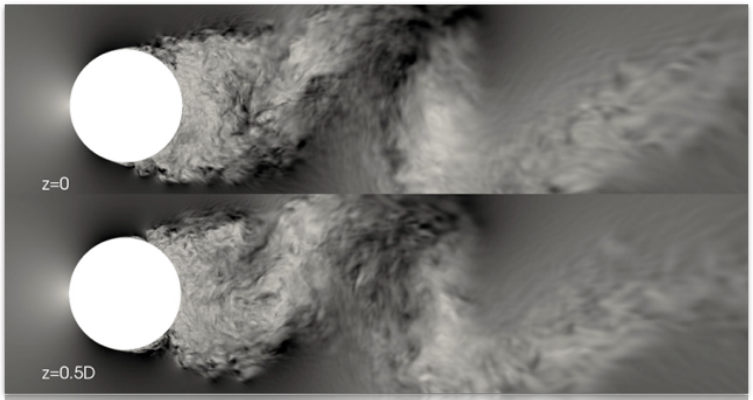


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Appendix: Piz Daint

Piz Daint





Applications

- Basic research on turbulent flows around bluff bodies

Model

- Navier-Stokes (LES)

Homogeneous mesh

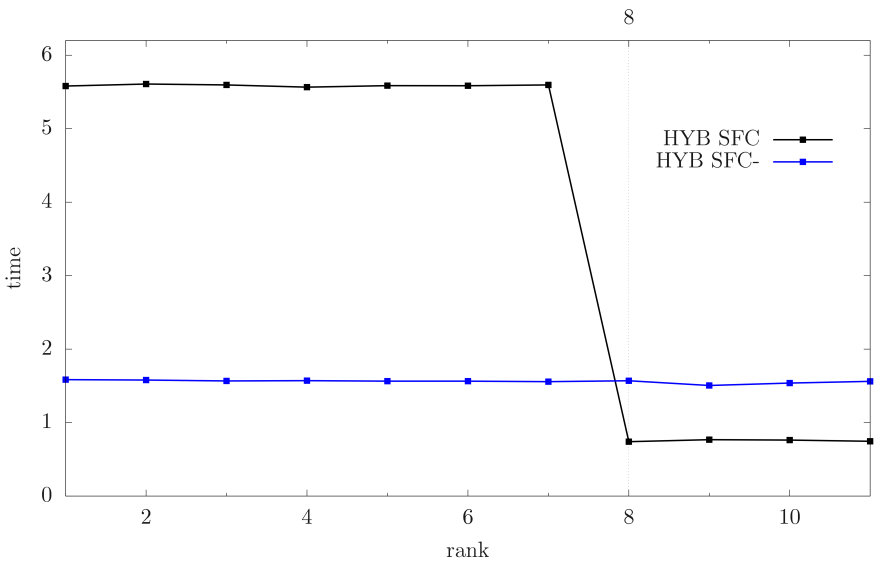
- 2.5 M elements: tetrahedra (Piz Daint)
- 20 M elements: tetrahedra (Piz Daint)

Piz Daint hybrid nodes

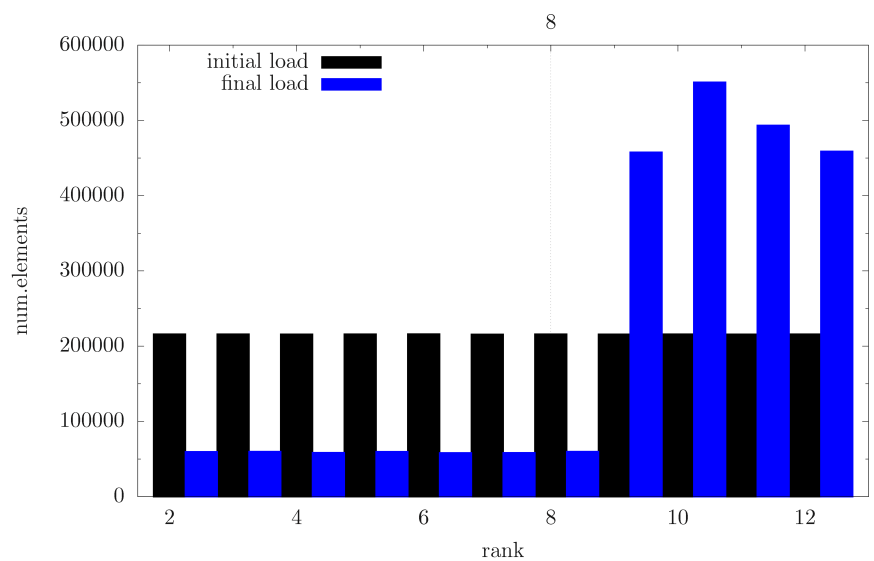
- Intel® Xeon® E5-269016 12 cores
- NVIDIA® Tesla® P100
- Interconnect: Aries routing and communications ASIC, and Dragonfly network topology

Piz Daint

Initial/final blance



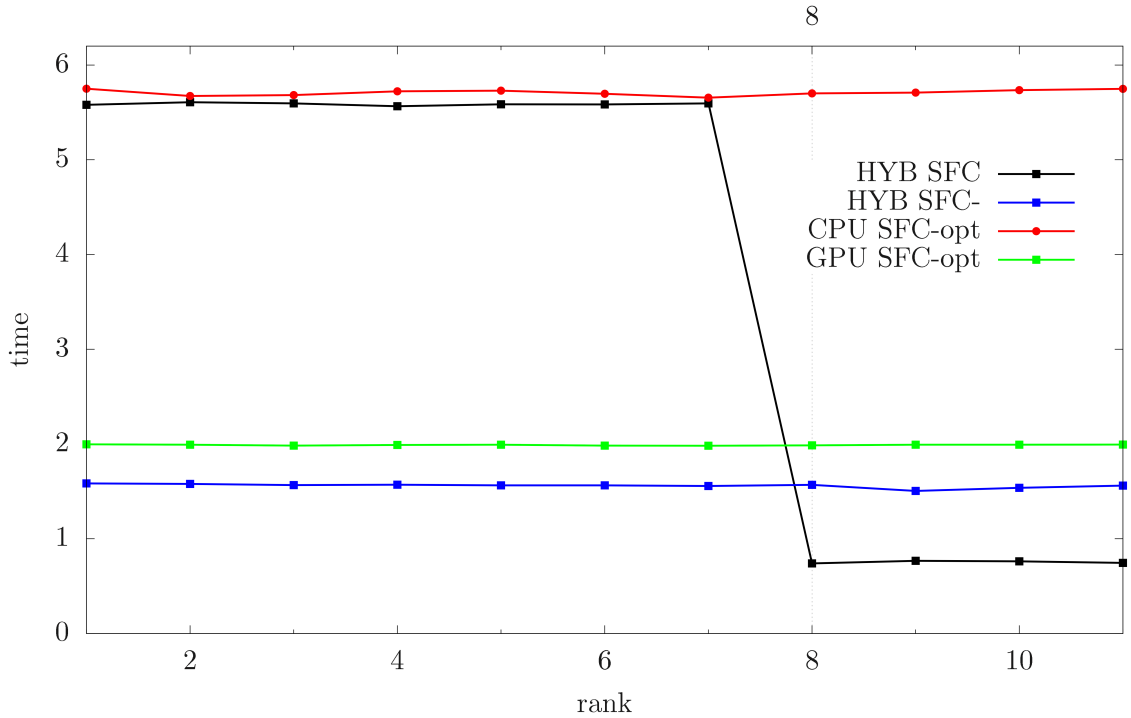
Initial/final load distribution



Sphere 2.5M

~8X more load for GPU threads

Piz Daint



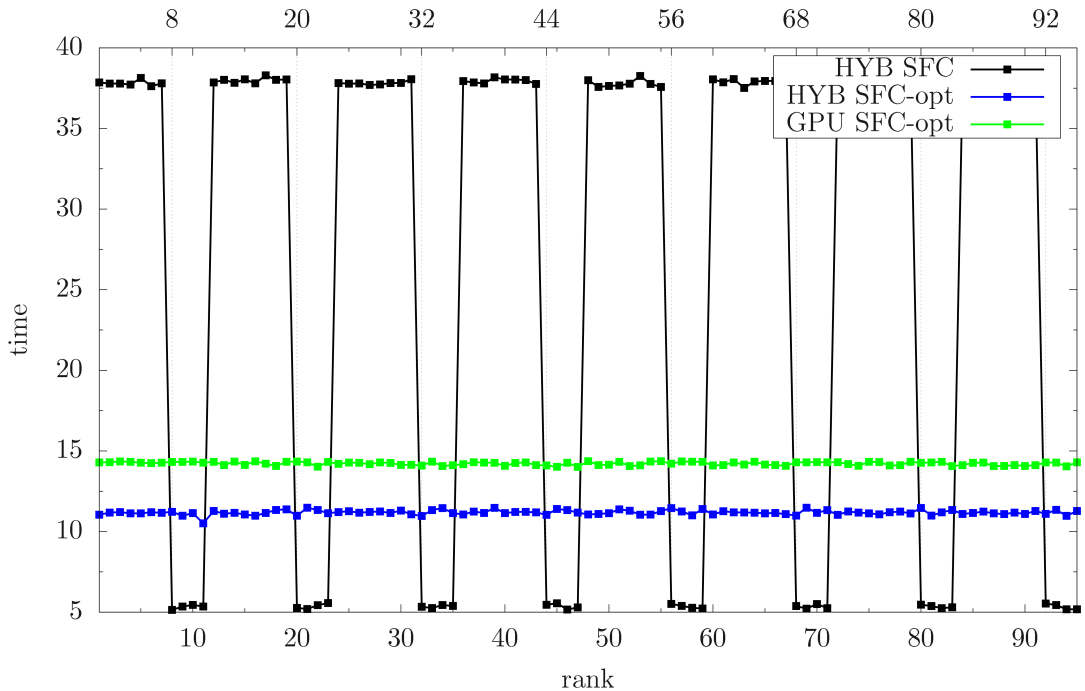
- 21% faster than the pure-MPI execution

21%

ratio GPU /CPU = 2.99 (PGI compiler)
max profit: 25%

Sphere 2.5 M

Piz Daint

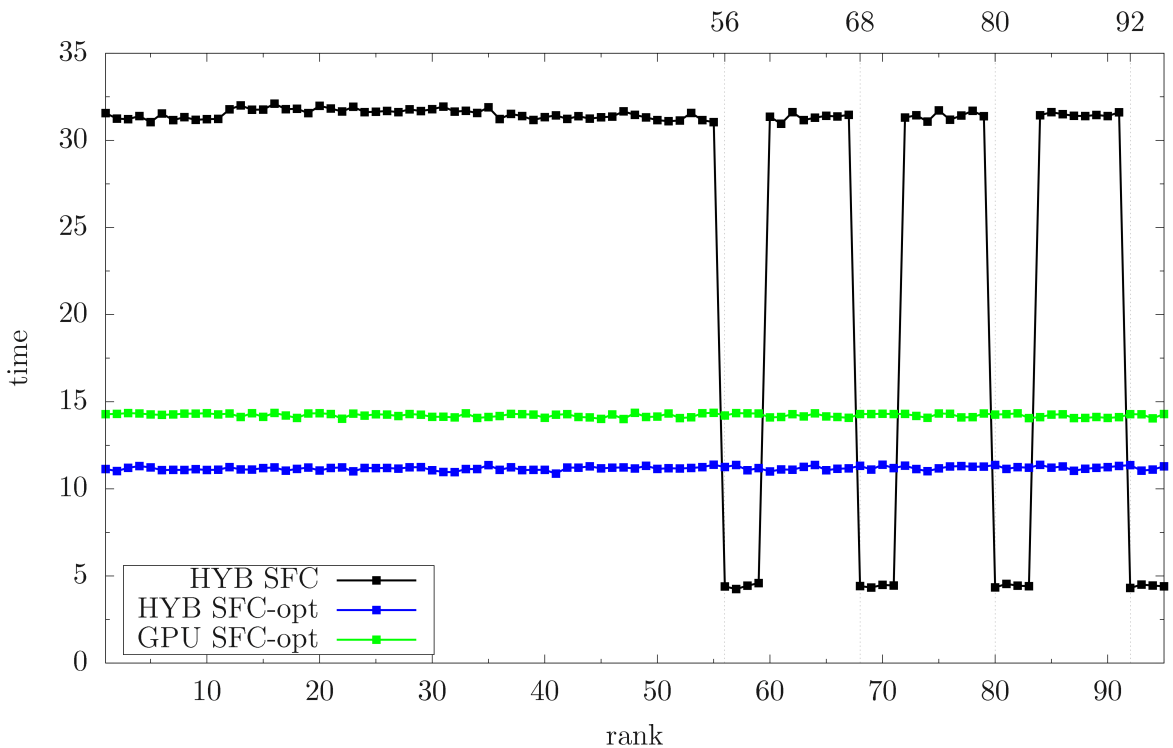


22%

- 8 nodes: on each node 4 mpi ranks running on the GPU

Sphere 20M

Piz Daint



Sphere 20M

- Using GPUs in half of the nodes