



MPI parallelisation: Part 1

Jean-Paul Pelteret (jean-paul.pelteret@fau.de)

Luca Heltai (luca.heltai@sissa.it)

21 March 2018



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



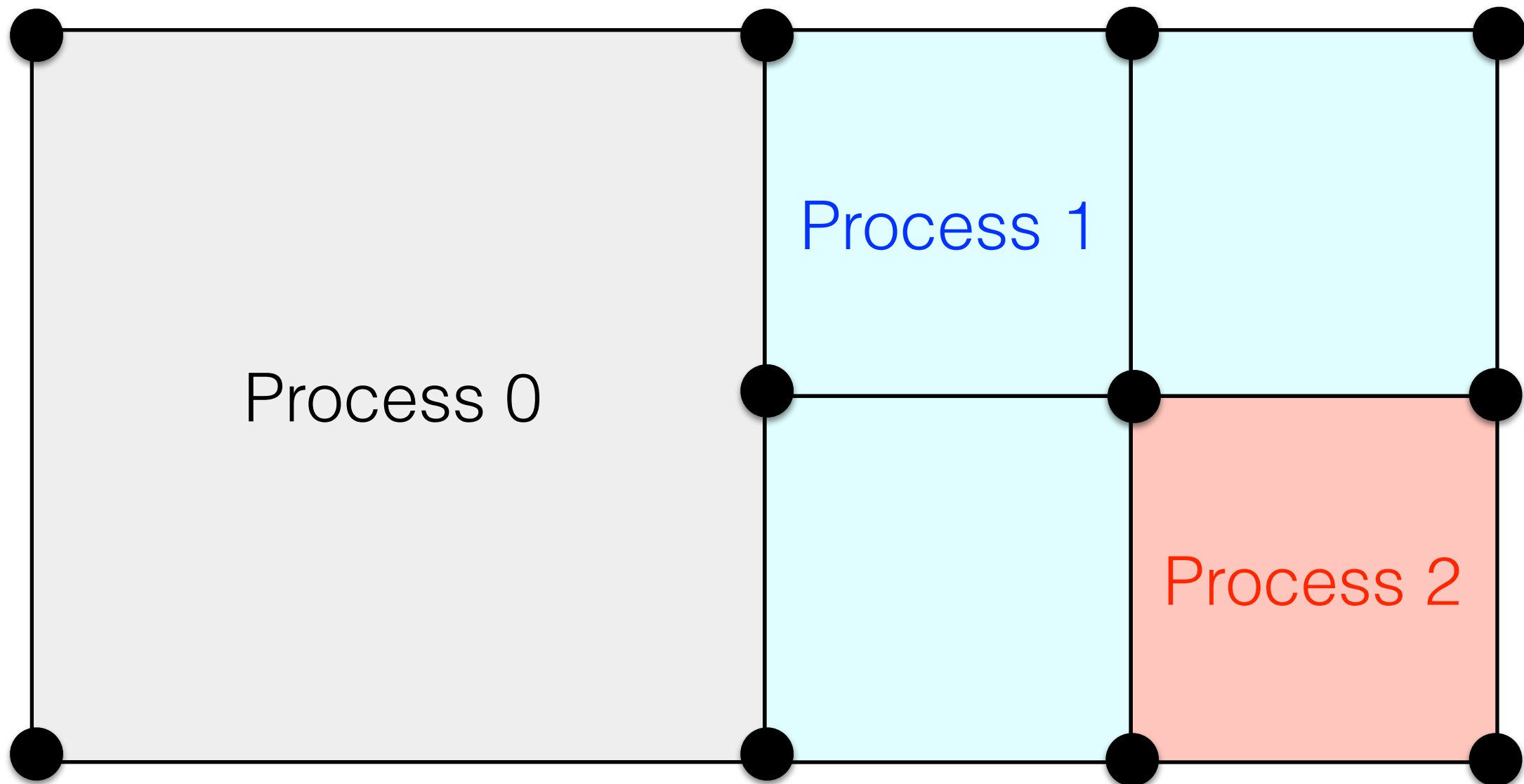
Aims for this module

- First introduction into parallel computing with deal.II
- Parallel distribution of degrees-of-freedom
 - Ownership concepts
- Setup data structures for parallel processing
- Assembly in parallel
- Synchronisation of distributed data
- Visualisation of distributed solutions

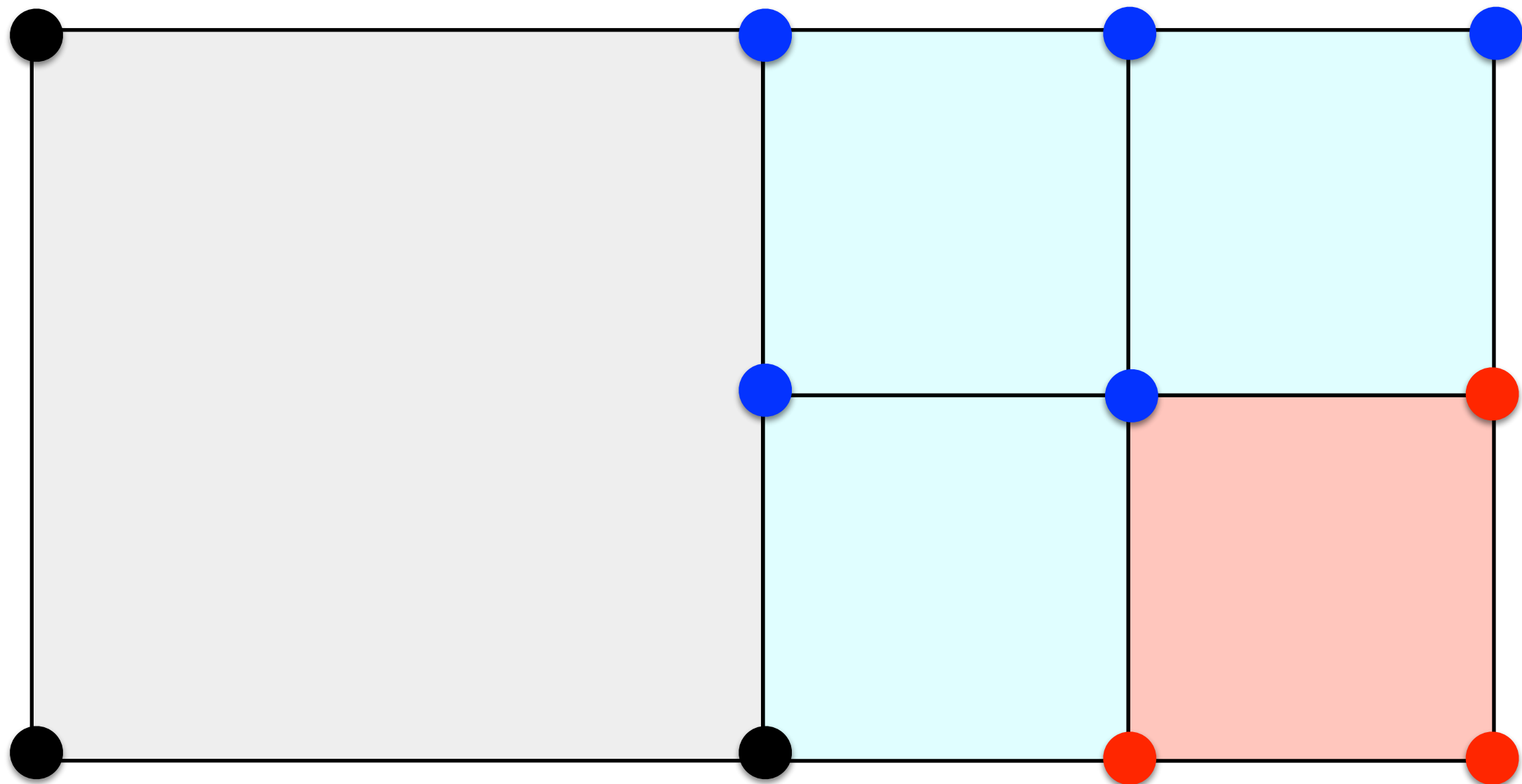
Reference material

- Tutorials
 - https://dealii.org/8.5.1/doxygen/deal.II/step_17.html
 - https://dealii.org/8.5.1/doxygen/deal.II/step_18.html
 - <http://www.math.colostate.edu/~bangerth/videos.676.39.html>
 - <http://www.math.colostate.edu/~bangerth/videos.676.41.html>
 - <http://www.math.colostate.edu/~bangerth/videos.676.41.25.html>
 - <http://www.math.colostate.edu/~bangerth/videos.676.41.5.html>
 - <http://www.math.colostate.edu/~bangerth/videos.676.41.75.html>
- Documentation:
 - https://www.dealii.org/developer/doxygen/deal.II/group__distributed.html

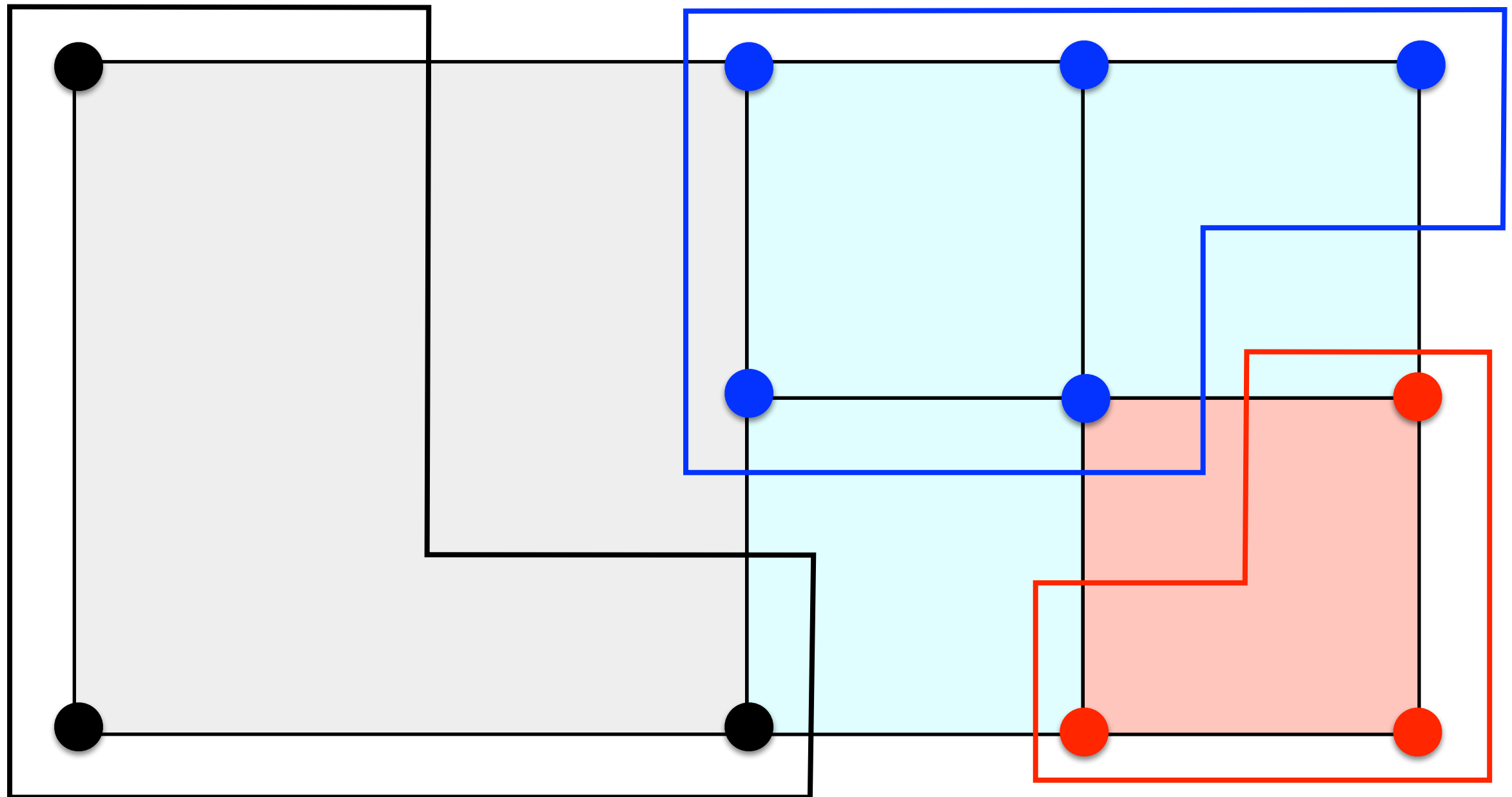
Distribution of degrees-of-freedom: Colourisation of cell ownership via graph partitioner



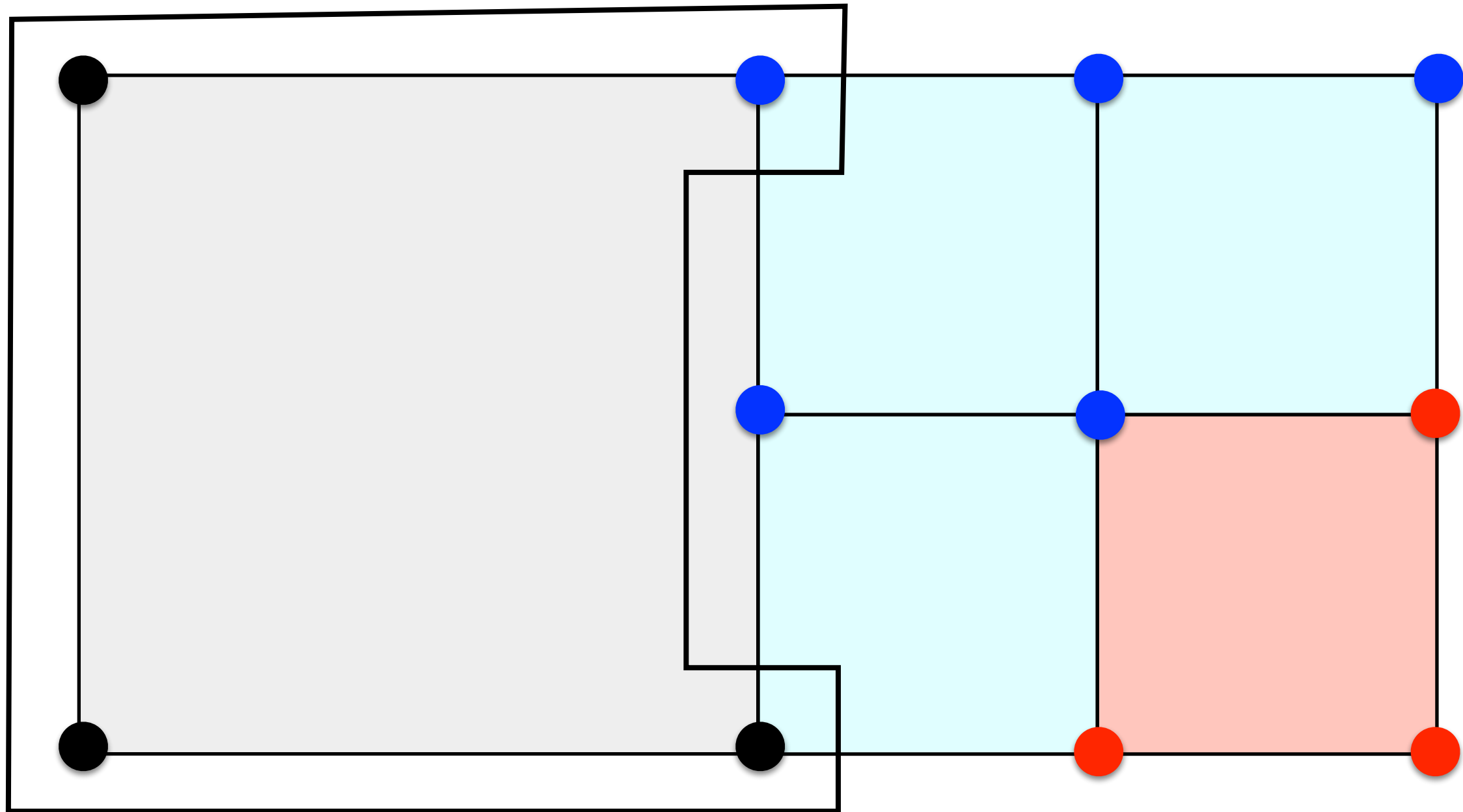
Distribution of degrees-of-freedom: Colourisation of DoFs via graph partitioner



Ownership of distributed DoFs: Locally owned degrees-of-freedom

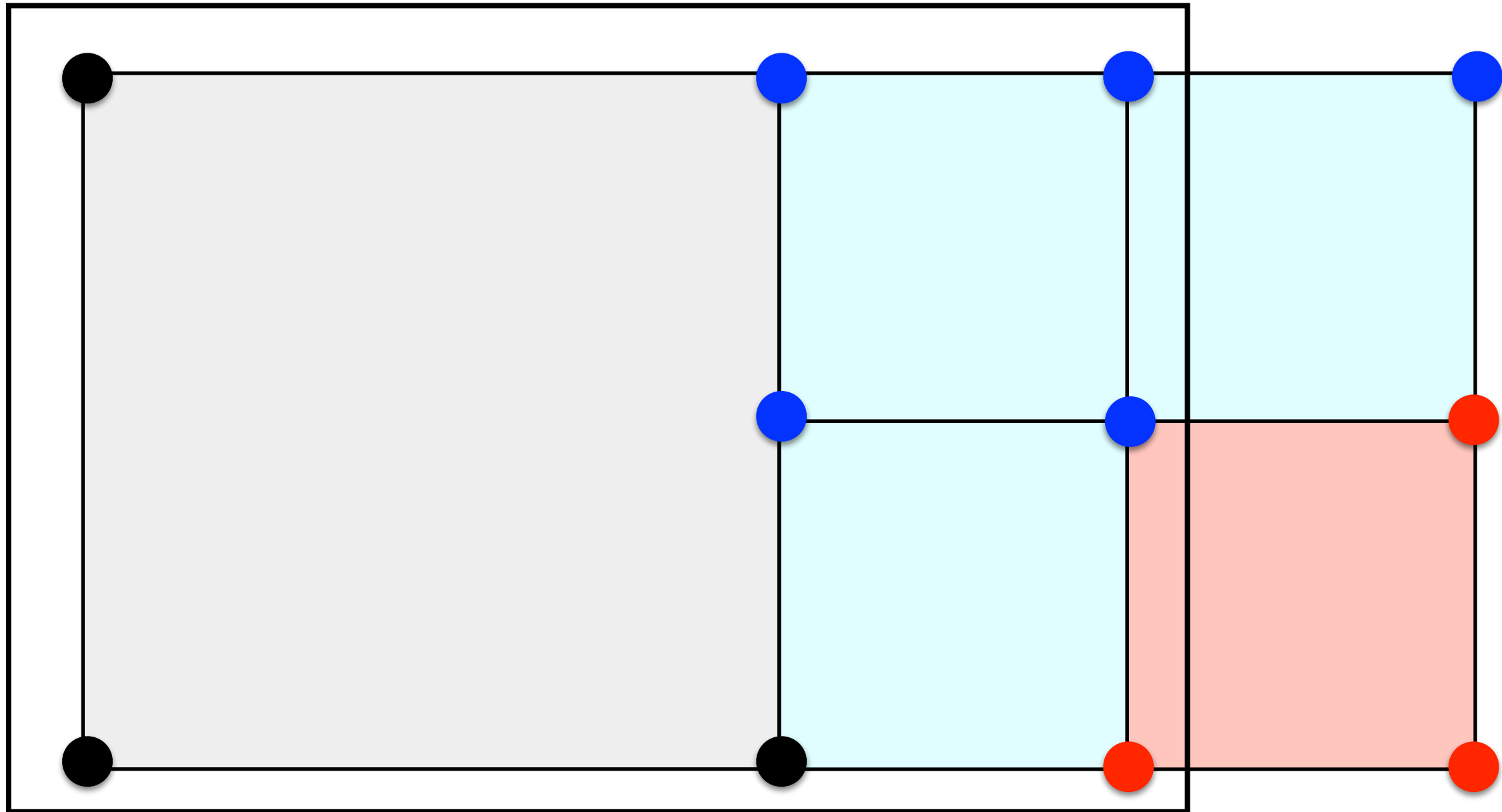


Distribution of DoFs: Locally relevant degrees-of-freedom (process 0)



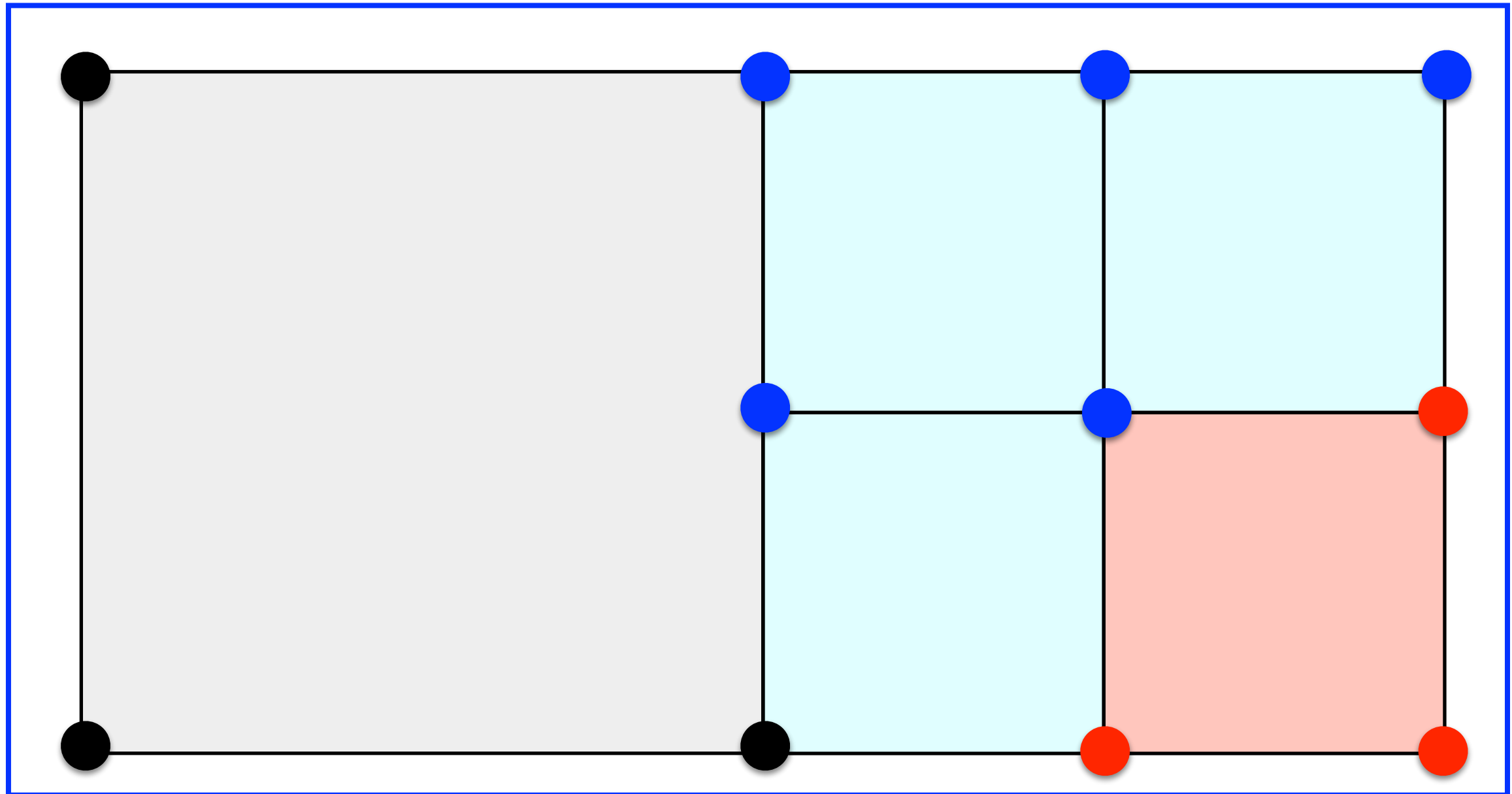
- Required for assembly, data output

Distribution of DoFs: Locally relevant degrees-of-freedom (process 0)



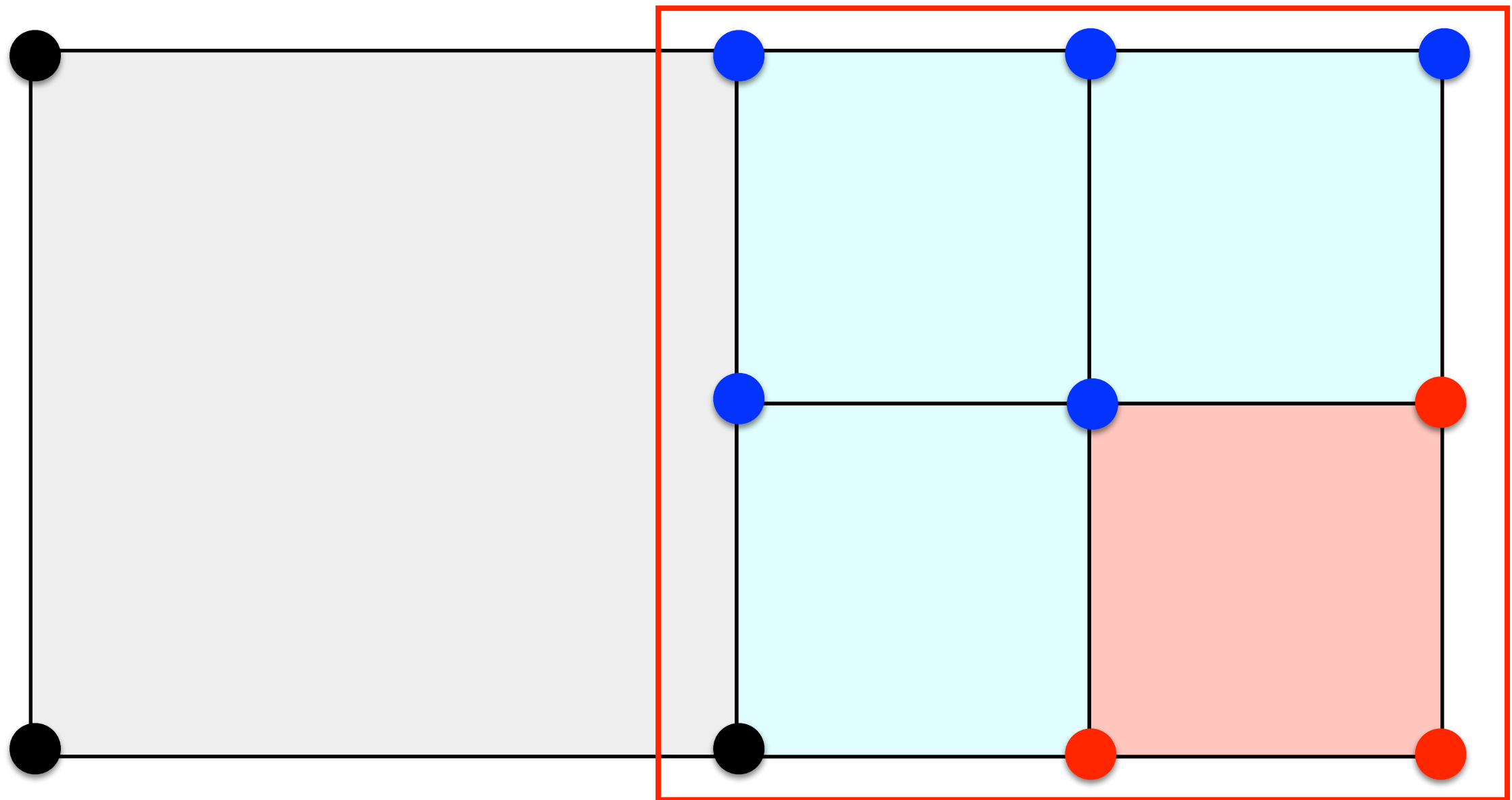
- Required for Kelly error estimator

Distribution of DoFs: Locally relevant degrees-of-freedom (process 1)



- Required for Kelly error estimator

Distribution of DoFs: Locally relevant degrees-of-freedom (process 2)



- Required for Kelly error estimator

Changes: New headers

- MPI
`#include <deal.II/base/mpi.h>`
- Parallel shared triangulation
`#include <deal.II/distributed/shared_tria.h>`
- Filtered iterator
`#include <deal.II/grid/filtered_iterator.h>`
- IndexSet
`#include <deal.II/base/index_set.h>`
- Trilinos linear algebra
`#include <deal.II/lac/trilinos_*>`
- Output filter
`#include <deal.II/base/conditional_ostream.h>`

Changes: Class definition

- MPI utility objects
 - MPI communicator
 - Number of processes, number of “this” process
 - Stream output assistant (filter)
- Triangulation type
- Sparse linear algebra objects
- IndexSets
 - Locally owned
 - Locally relevant

Changes: System setup

- Must determine the set of locally owned and locally relevant DoFs
 - Locally owned = those assigned to a particular MPI process
 - Locally relevant = those assigned to other processors, but are required to perform some action on the current process
- Distribution of sparsity pattern
 - Tell the locally defined sparsity pattern which entries require data exchange / may be written into by other processes
- Can interrogate information about problem distribution as viewed from other processors

Changes: Assembly

- Cell loop: Only cells owned by the MPI process
 - Can use filtered iterators
 - Can check within the cell loop
`cell->locally_owned();`
- All assembled data is initially localised to a process
- Final synchronisation of data between MPI processes
 - Accumulation of values on DoFs written into by more than one process
`[matrix/vector].compress(VectorOperations::add);`
 - Only once (outside of cell loop)

Changes: Linear solver

- Solver templated on Vector type
- Preconditioner type

Changes: Mesh refinement

- Kelly error estimator needs to know solution values on cells that are not owned by this current MPI process
- Need to provide view of solution with values for all locally owned and locally relevant DoFs

Changes: Postprocessing

- Write out portion of solution from each processor
 - deal.II writes these outputs on a per-cell basis
- Need to provide view of solution with values for all locally owned and locally relevant DoFs

Changes: Main function

- Setup MPI environment

```
Utilities::MPI::MPI_InitFinalize
```

```
mpi_initialization(argc, argv, 1);
```