PaMPA : Parallel Mesh Partitioning and Adaptation

Cédric Lachat & François Pellegrini

June 14, 2012

Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

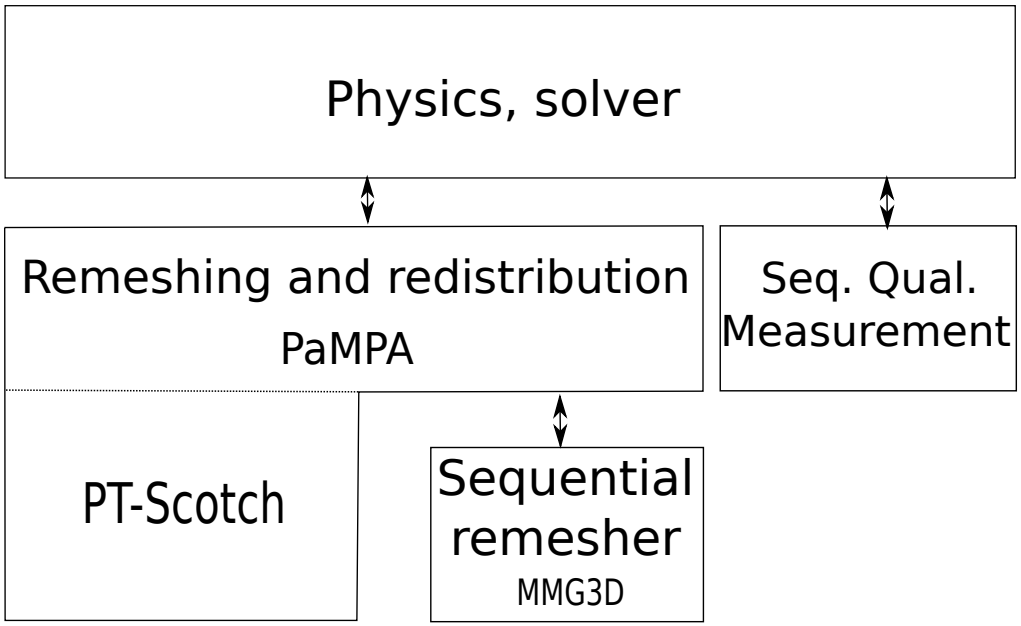
Example: Laplacian equation using finite element method

Some results

Upcoming features

Middleware

- PaMPA: Parallel Mesh Partitioning and Adaptation
- Middleware library managing the parallel repartitioning and remeshing of unstructured meshes modeled as interconnected valuated entities
- The user can focus on his/her "core business":
  - Solver
  - Sequential remesher
    - Coupling with MMG3D provided for tetrahedra



Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

Example: Laplacian equation using finite element method

Some results

Upcoming features

Common needs of solvers regarding meshes:

- Handling of mesh structures
- Distribution of meshes across the processors of a parallel architecture
  - Handling of load balance
- Data exchange across neighboring entities
- Iteration on mesh entities
  - Entities of any kind: e.g. elements, faces, edges, nodes, ...
  - Entity sub-classes: e.g. regular or boundary faces, ...
  - Inner or frontier entities with respect to neighboring processors
  - Maximization of cache effects thanks to proper data reordering
- Dynamic modification of mesh structure
  - Dynamic redistribution
- Adaptive remeshing

Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

Example: Laplacian equation using finite element method

Some results

Upcoming features

Constraints

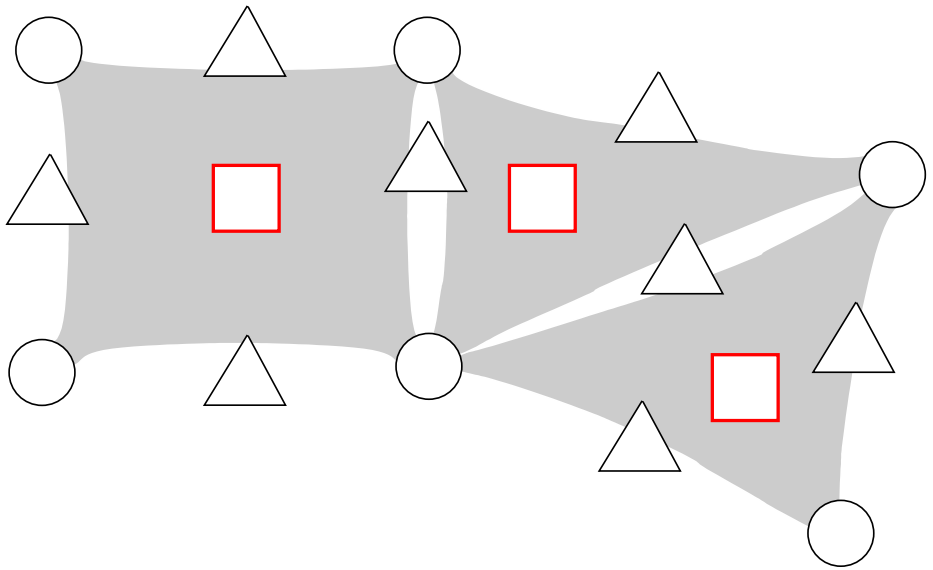
- Distributed data
- Mesh data should not be replicated for the sake of scalability
- Minimization of data exchanges
- Abstraction from actual data structure implementations

Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

Top-level mesh entity

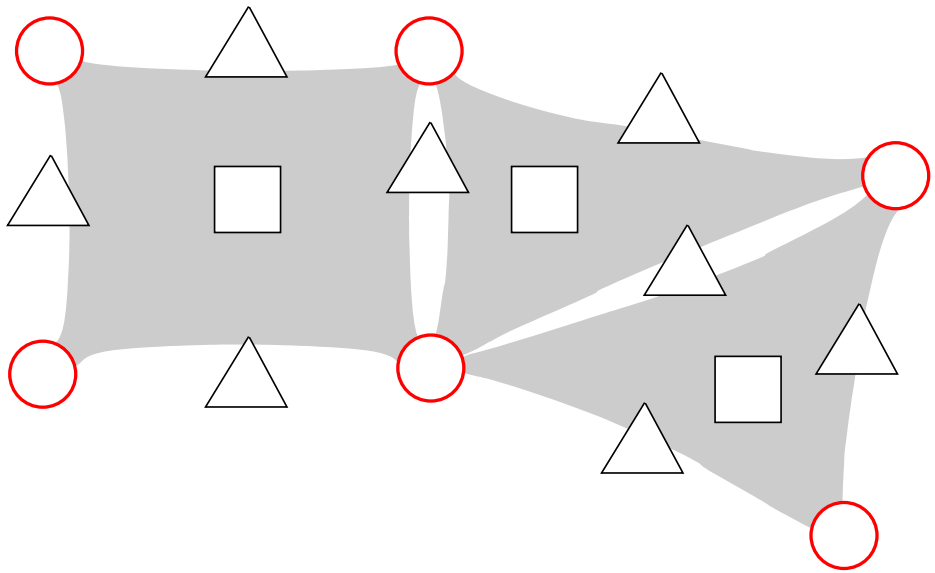
May bear some data (volume, pressure, etc.)



Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

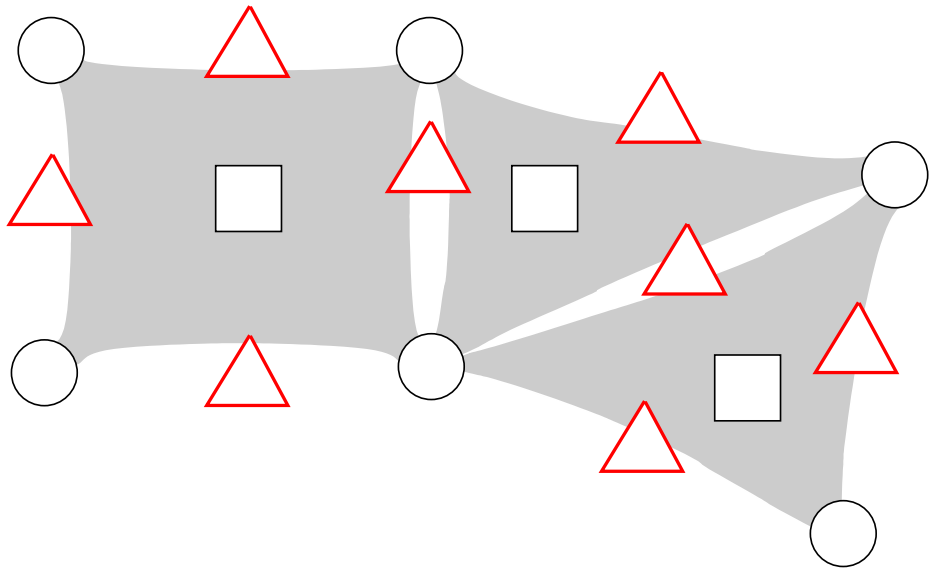
May bear some data (geometry, etc.)



Definitions

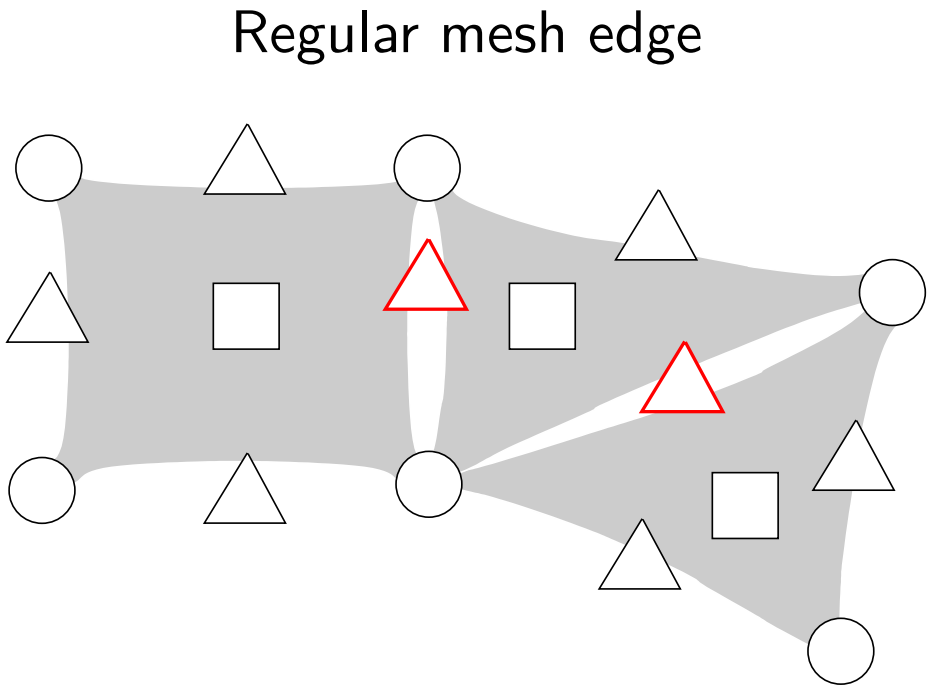
- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

May bear some data (flux, etc.)



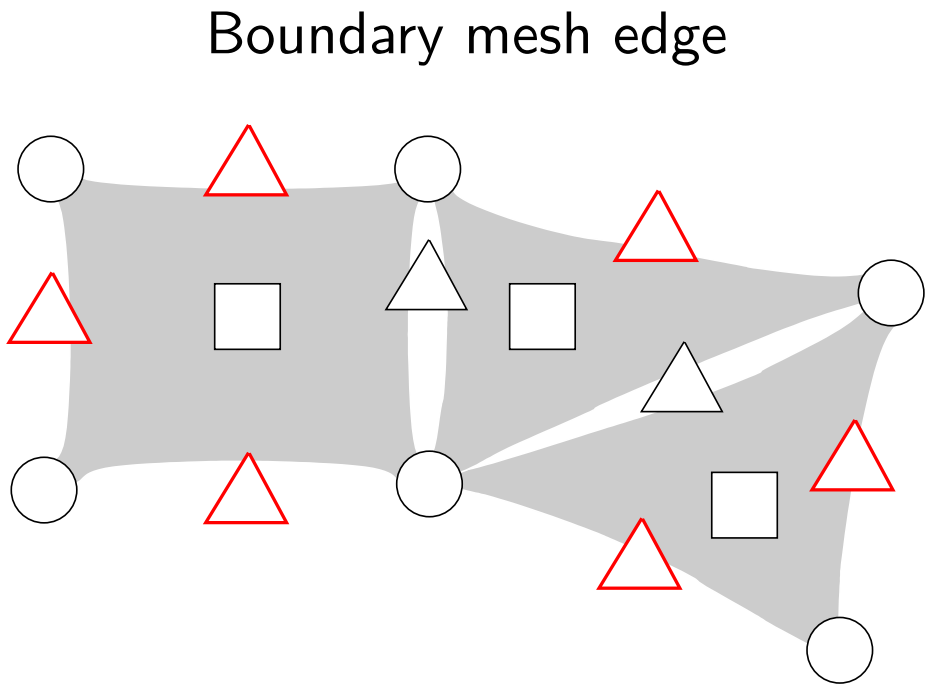
Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**



Definitions

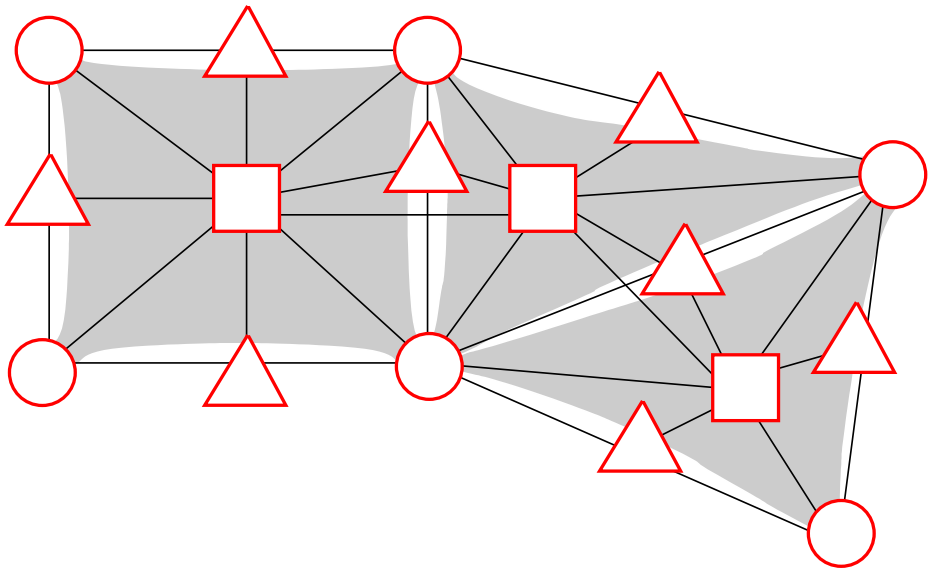
- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**



Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

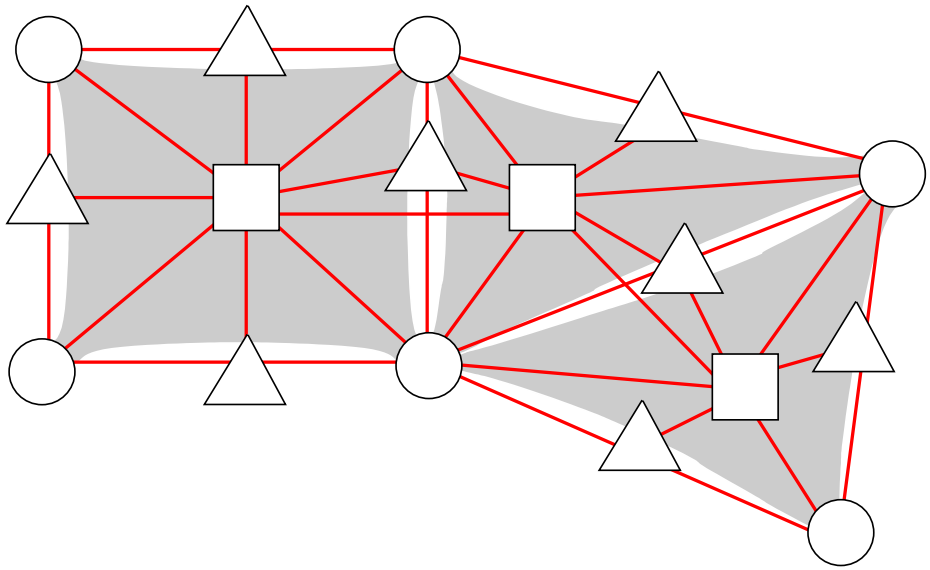
What all entities are in fact...



Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

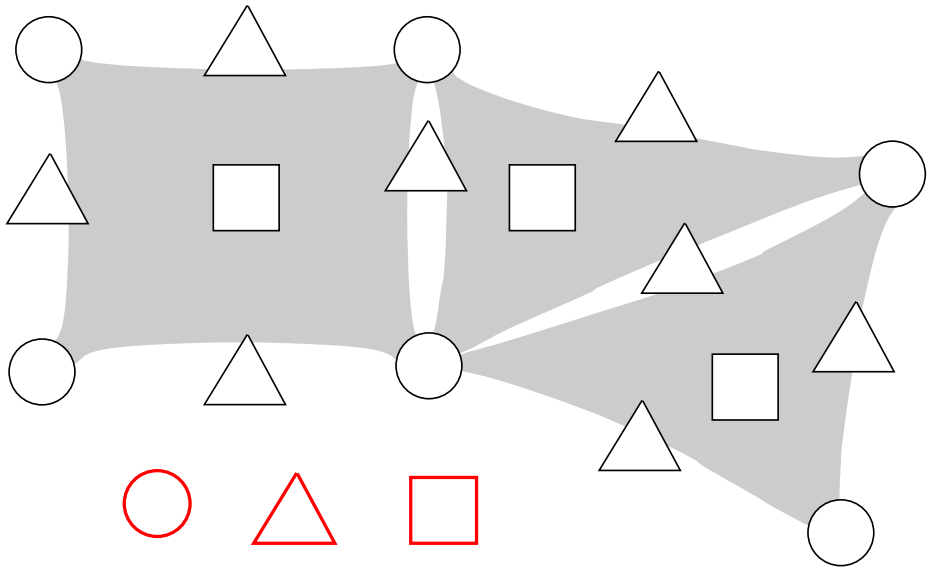
Subset of edges between vertices belonging to prescribed entity types



Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

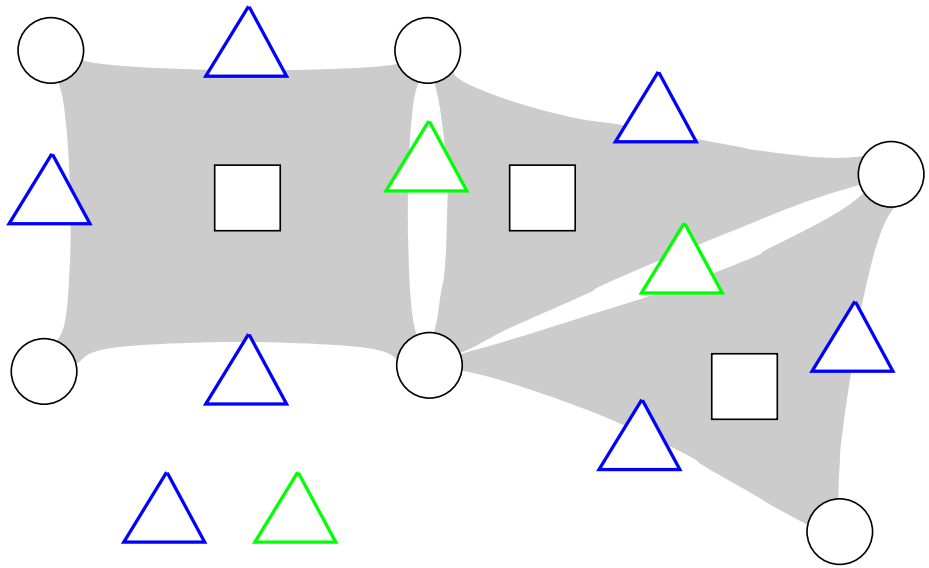
Subset of vertices bearing the same data



Definitions

- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

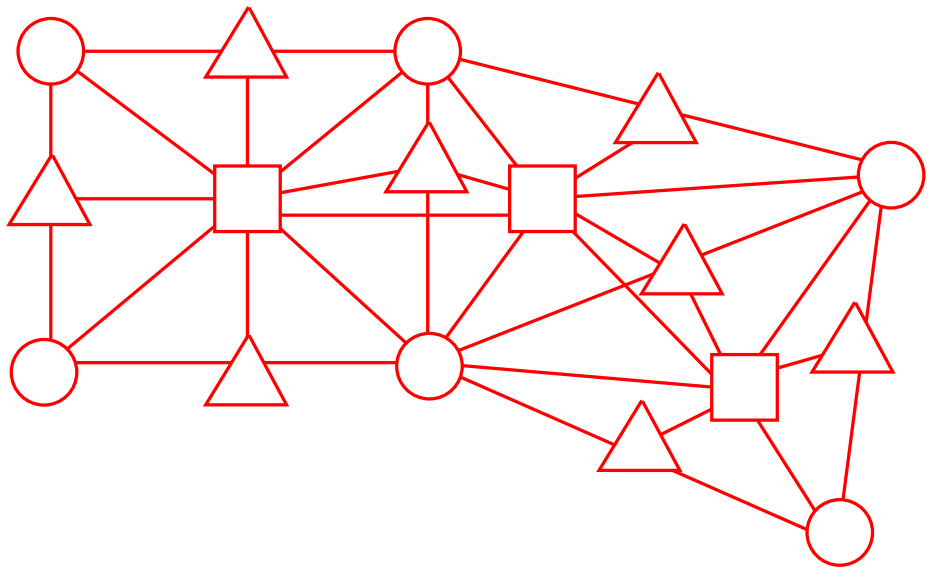
Subset of entity vertices that may bear specific data



Definitions

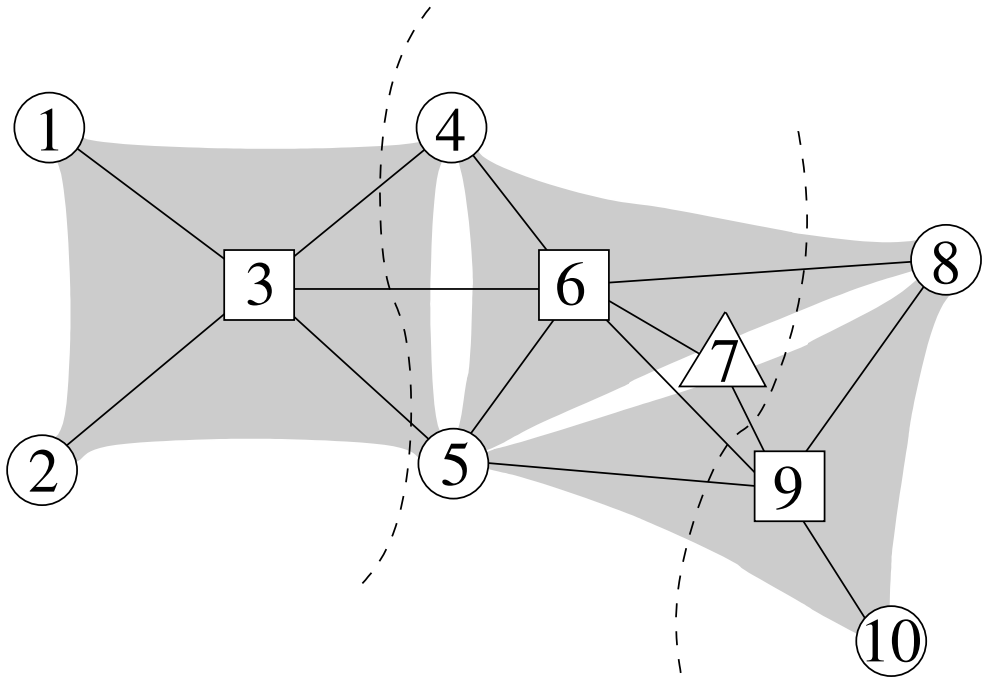
- **Element**
- **Node**
- **Edge**
  - **Internal**
  - **Boundary**
- **Vertex**
- **Relation**
- **Entity**
- **Sub-entity**
- **Enriched graph**

Whole set of vertices and relations
Every vertex belongs to one and only one entity (and sub-entity)

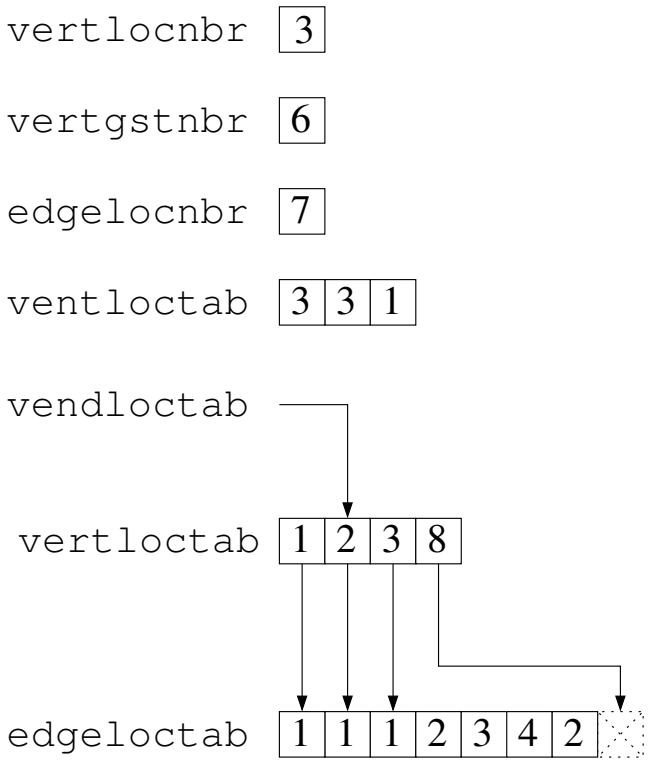


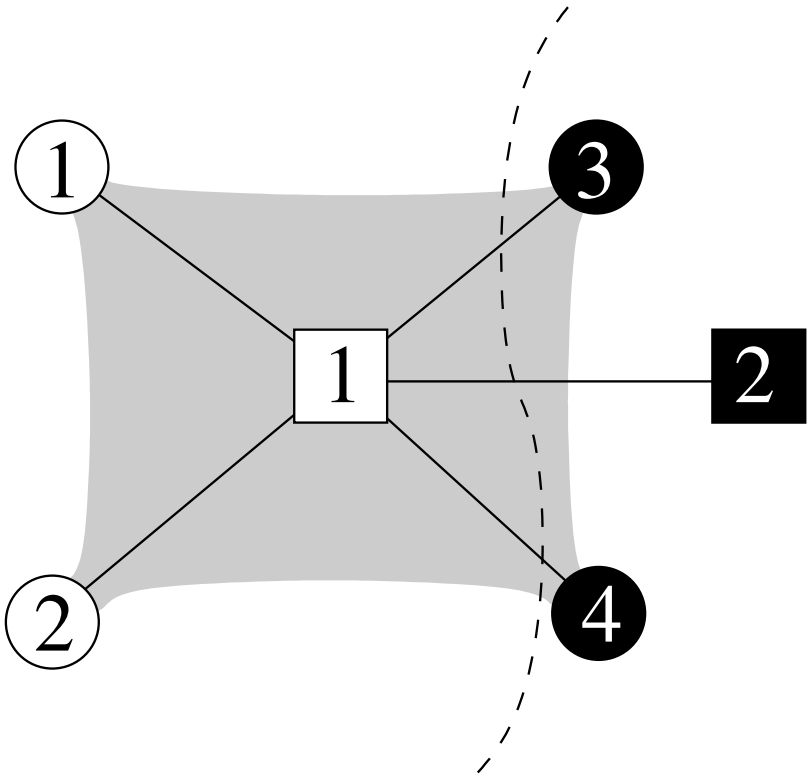
Global vue

`baseval`	1
`enttglbnbr`	3
`proccnttab`	3 4 3
`procvrttab`	1 4 8 11

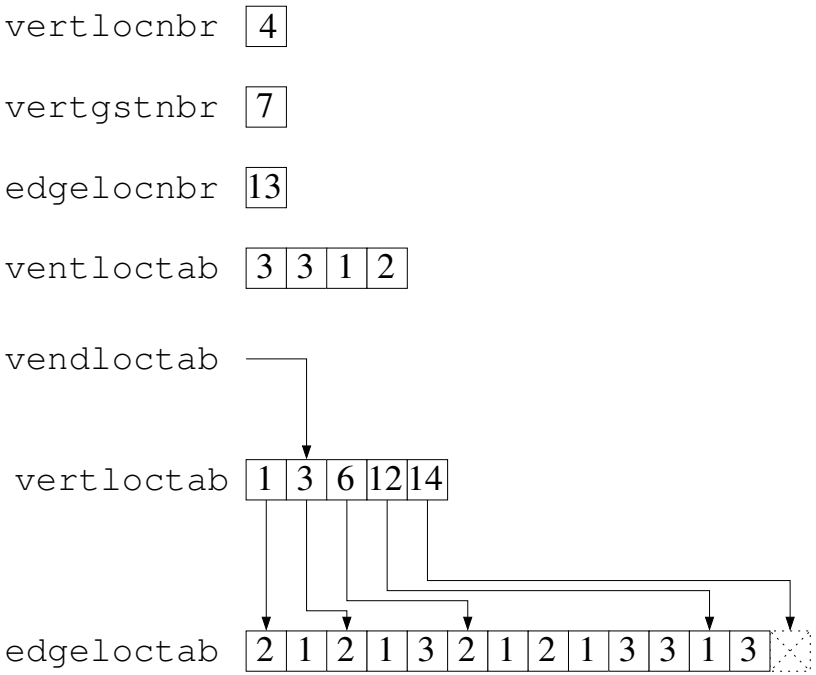


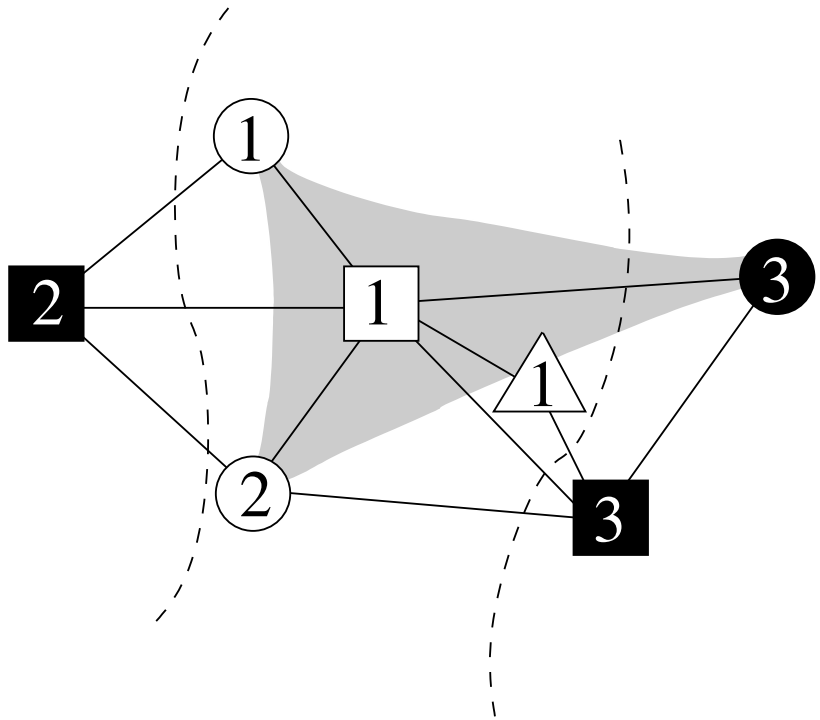
Local vision of process 0





Local vision of process 1



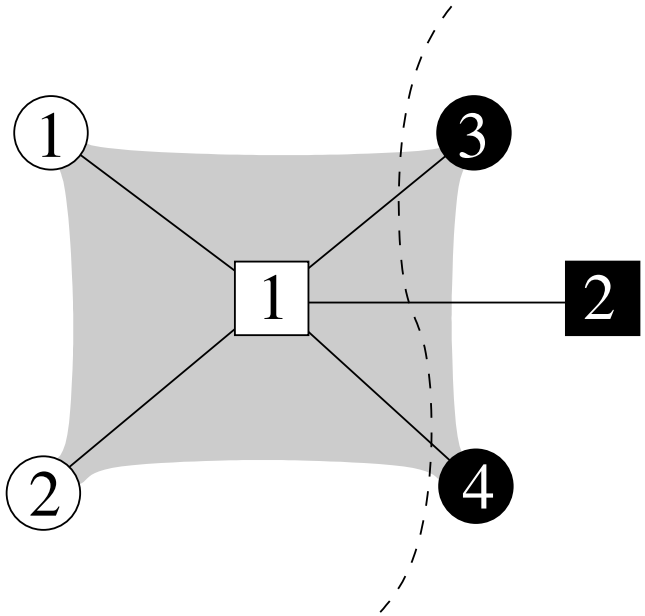


Per-entity data (1/2)

E.g. entity 1 on processor 0:

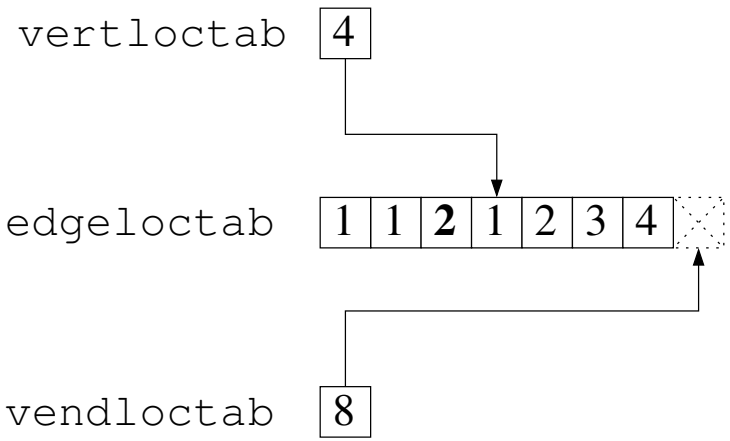
`vertlocnbr` 1

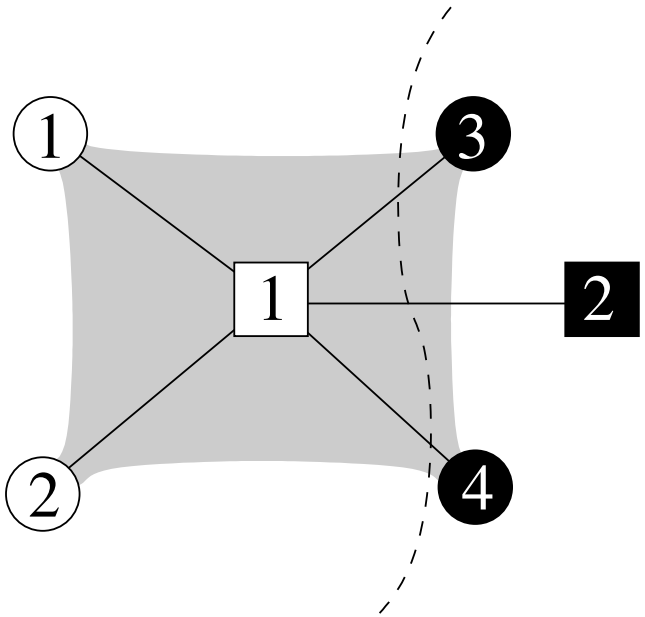
`vertgstnbr` 2



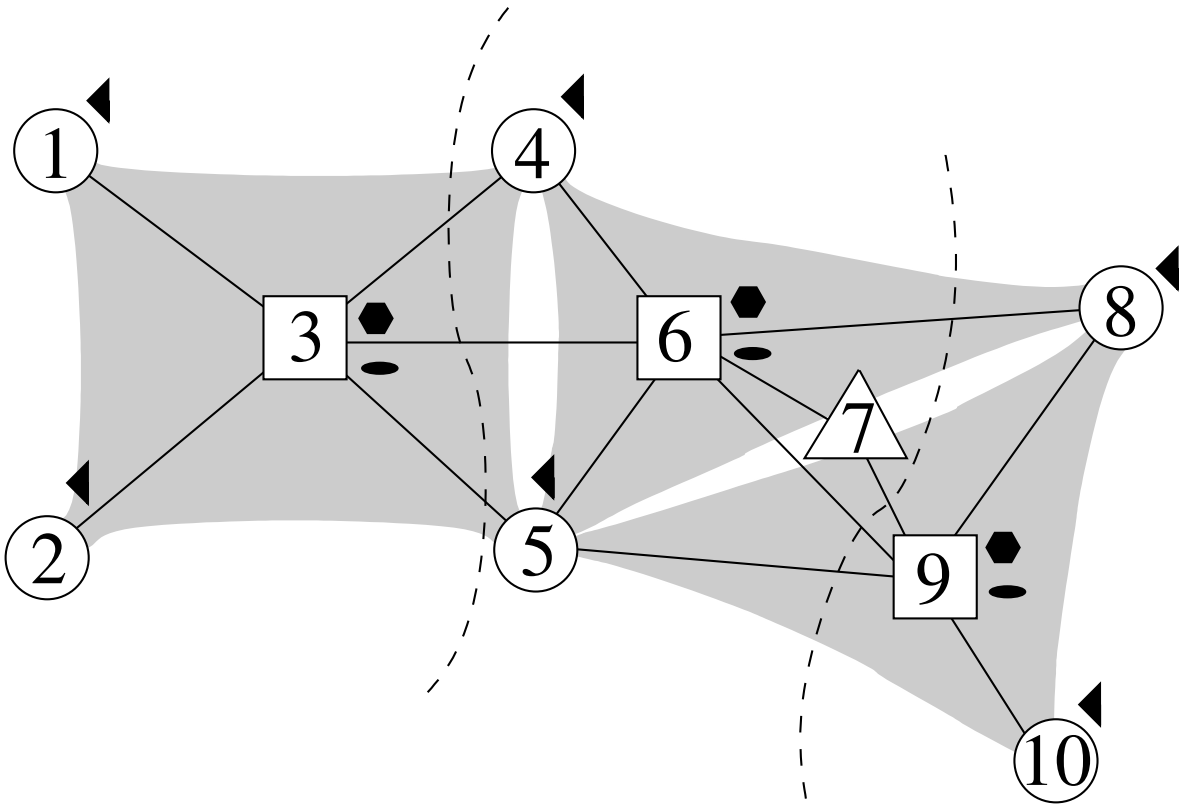
Per-entity data (2/2)

E.g. node neighbors of element 1 on process 0:





Data linking



Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

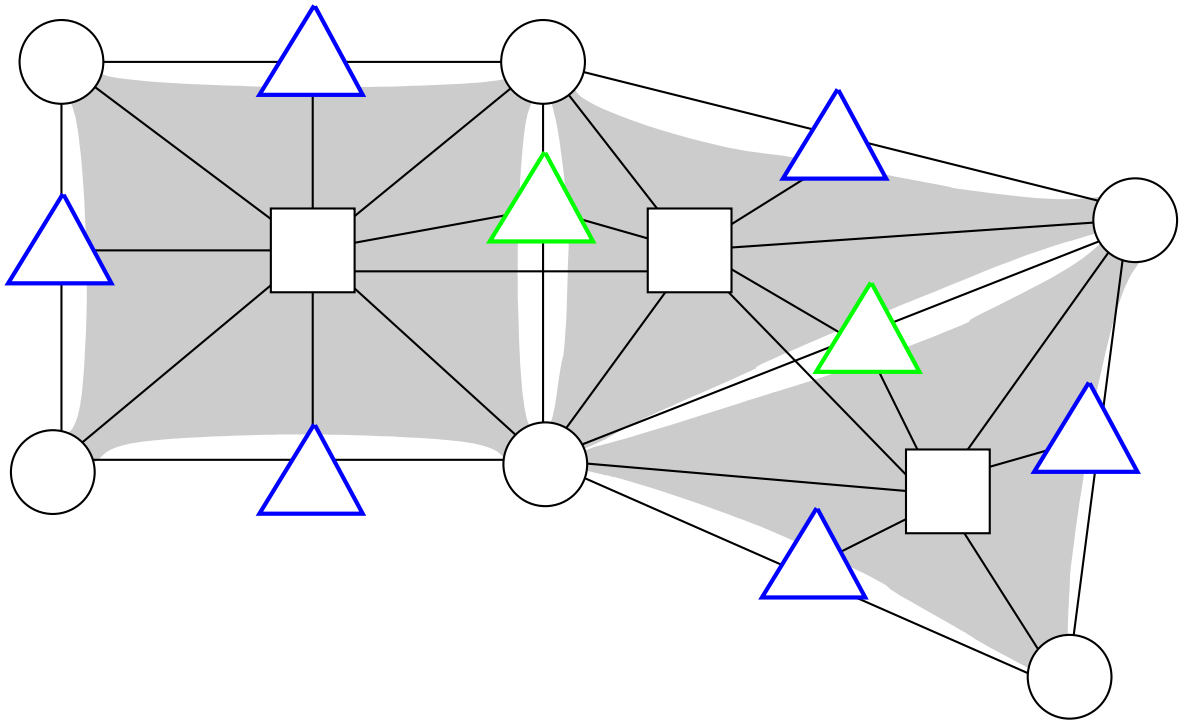
Example: Laplacian equation using finite element method

Some results

Upcoming features

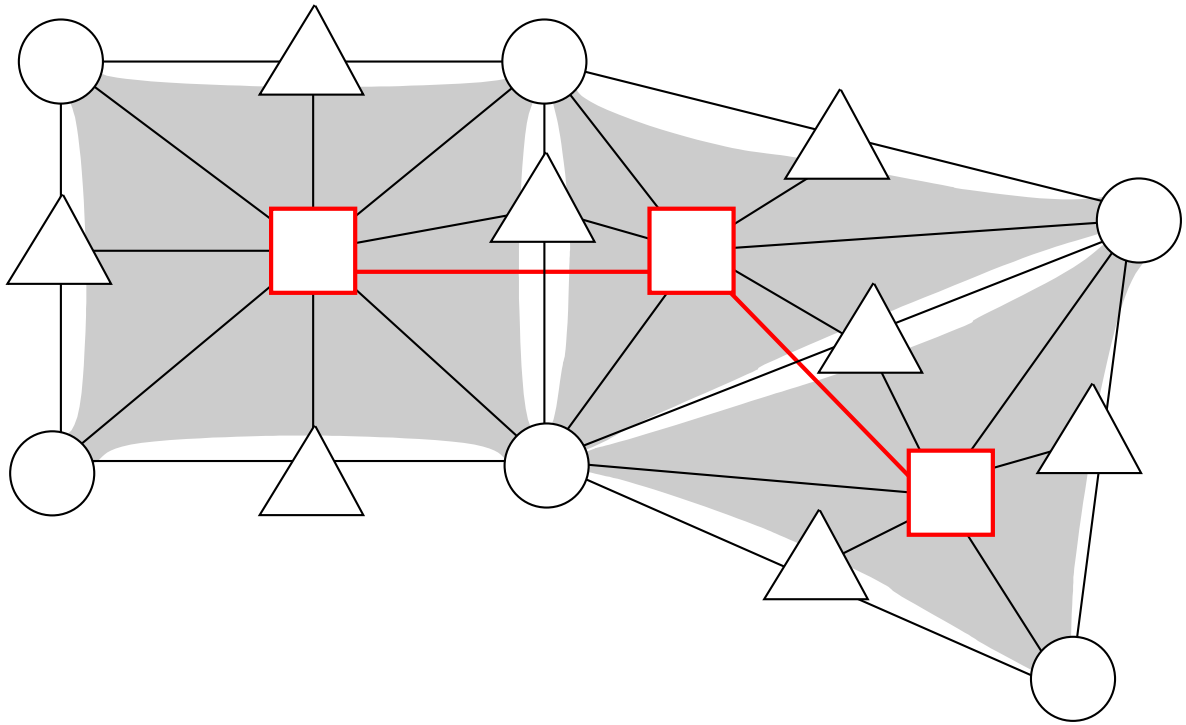
Entities

- Based (for C and Fortran interfaces)
- Simple
- Stable with respect to sub-entities



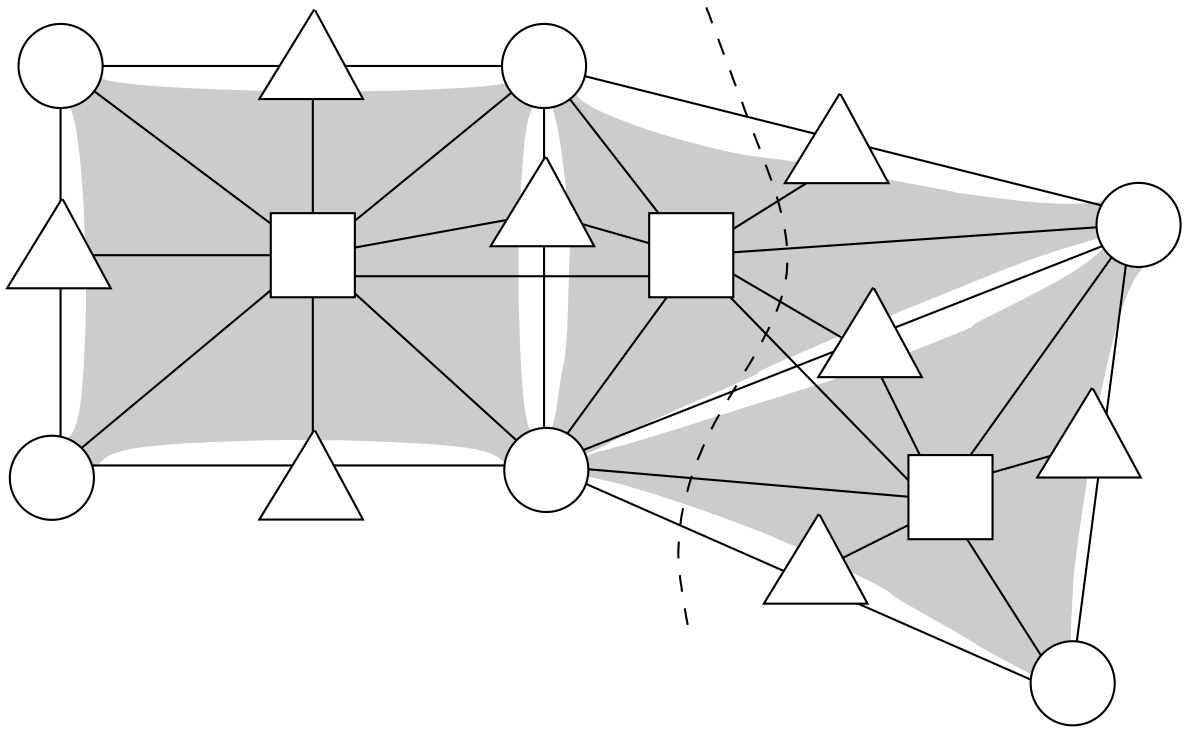
Partitioning

- Performed with respect to top-level entity



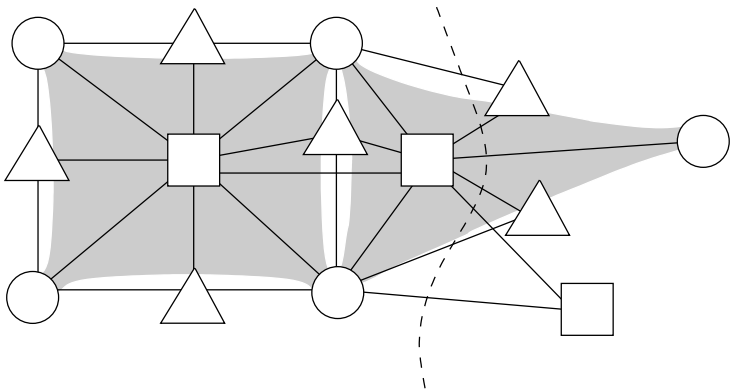
Overlap

- Of size 1 at the time being
- Extensible to any distance n
- Requires top-level relations

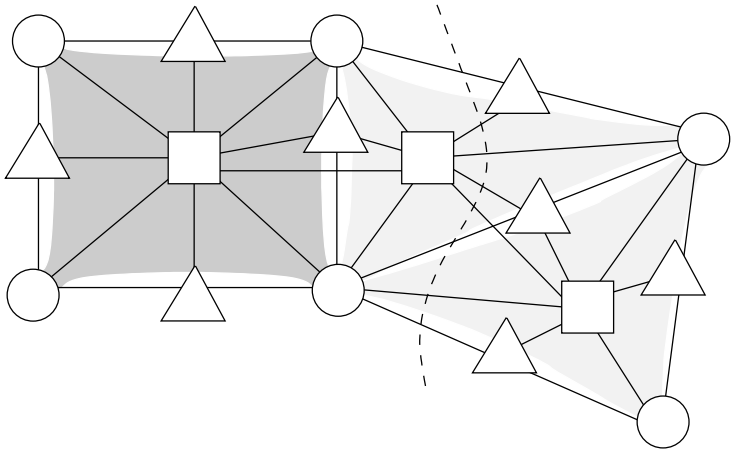


Local vue of process 0

- Halo of size 1

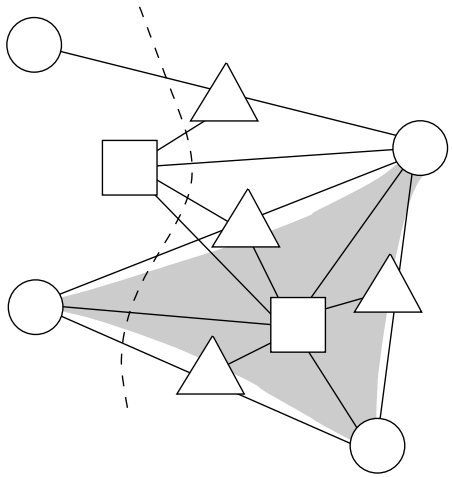


- Overlap of size 1

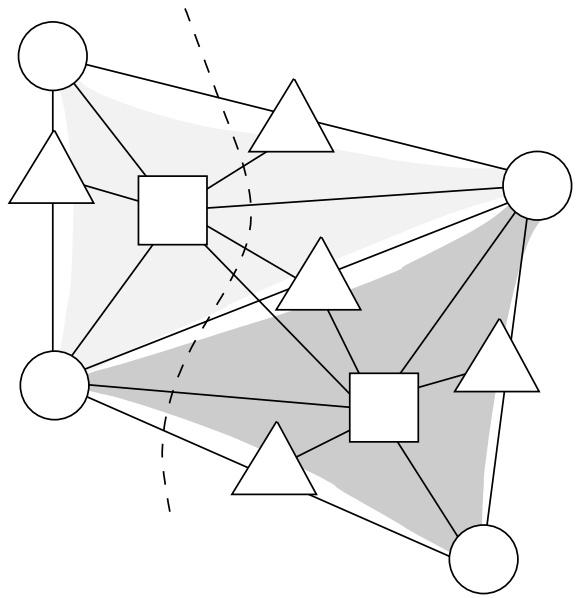


Local vue of process 1

- Halo of size 1



- Overlap of size 1



Iterators

- Entities - sub-entities
- Decrease of cache misses
- Debugging tools

```
PAMPA_dmeshItInitStart(mesh, ventlocnum, PAMPA_VERT_ANY, &it_vert);
PAMPA_dmeshItInit(mesh, ventlocnum, nentlocnum, &it_nghb);

while (PAMPA_itHasMore(&it_vert)) {
  vertlocnum = PAMPA_itCurEnttvertlocnum(&it_vert);
  printf("vertlocnum : %d\n", vertlocnum);
  PAMPA_itStart(&it_nghb, vertlocnum);

  while (PAMPA_itHasMore(&it_nghb)) {
    PAMPA_Num sngblocnum, engblocnum, mngblocnum, nsublocnum;
    engblocnum = PAMPA_itCurEnttvertlocnum(&it_nghb);
    mngblocnum = PAMPA_itCurMeshvertlocnum(&it_nghb);
    sngblocnum = PAMPA_itCurSubEnttvertlocnum(&it_nghb);
    nsublocnum = PAMPA_itCurnsublocnum(&it_nghb);
    printf( "[%d] ngb of %d with entity %d : %d(sub : %d) %d(ent : %d) %d(glb)\n",
                rank, vertlocnum, ventlocnum, sngblocnum, nsublocnum,
                engblocnum, nentlocnum, mngblocnum);
    PAMPA_itNext(&it_nghb);
  }
```

Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

Example: Laplacian equation using finite element method

Some results

Upcoming features

All steps

```
! Parallel Mesh Computations
!-----------------------------
      ! Processor 0 only:
CALL ReadMesh()                ! Read mesh on file
CALL GraphPampa()              ! Build graph of vertex neighbors for PAMPA
CALL PampaCentralizedMesh()    ! Build PaMPA global non distributed mesh

     ! On all processors:
CALL PampaDistributedMesh()    ! Build PAMPA distributed mesh:
!                                    1- Scatter centralized mesh on all processors
!                                    2- Call PAMPA mesh partionner
!                                    3- Redistribute distribute mesh
CALL ElementVolume()
CALL InitializeMatrixCSR()

! Solution computation
!----------------------

CALL InitSol()
CALL FillMatrix()
CALL SolveSystem()
```

Solve system: Jacobi (1/2)

```
UaPrec = 0.          ! Suppose A = L + D + U, system to solve : A x = b
DO irelax = 1, Nrelax
  res       = 0.
  DO is     = 1, MatCSR%Nin
    res0    = RHS(is)                                  ! res0  = b
    i1      = MatCSR%Ia(is)
    I1Fin   = MatCSR%Ia(is + 1) -1

    DO iv   = I1, I1Fin
      js    = MatCSR%Ja(iv)
      res0  = res0 - MatCSR%Vals(iv) * UaPrec(js)  !res0  = b - (L + U) x^n
    END DO
    Ua(is)  = res0 / MatCSR%Diag(is)               !x^n+1 = ( b - (L + U) x^n )/D
    res0    = res0 - MatCSR%Diag(is) * UaPrec(is)  !res0  = ( b - (L + D + U) x^n )
    res0    = res0 * res0                          !res0  = ( b - (L + D + U) x^n )
    res     = res  + res0
  END DO

  CALL PAMPAF_dmeshHaloValue(dm,                                              &
  &                          enttnum = PAMPA_ENTITY_NODE,                     &
  &                          tagnum  = INT(PAMPA_TAG_SOL, PAMPAF_NUM),        &
  &                          retval  = statut)
```

Solve system: Jacobi (2/2)

```
      deltaU     = 0.
      deltaU     = DOT_PRODUCT( UaPrec(:MatCSR%Nin) - Ua(:MatCSR%Nin) ,
                                UaPrec(:MatCSR%Nin) - Ua(:MatCSR%Nin) )
      res0       = 0.
      res0       = DOT_PRODUCT( RHS    (:MatCSR%Nin)                  ,
                                RHS    (:MatCSR%Nin)                  )
      UaPrec     = Ua

      deltaUglb = 0.
      CALL MPI_ALLREDUCE(deltaU, deltaUglb, 1,type_real,MPI_SUM,MPI_COMM_WORLD,code)
      deltaUglb = sqrt(deltaUglb)

      resglb     = 0.
      CALL MPI_ALLREDUCE(res, resglb, 1, type_real, MPI_SUM, MPI_COMM_WORLD, code)
      resglb     = sqrt(resglb)

      res0glb    = 0.
      CALL MPI_ALLREDUCE(res0, res0glb, 1, type_real, MPI_SUM, MPI_COMM_WORLD, code)
      res0glb    = sqrt(res0glb)

      resglb     = resglb / res0glb

  END DO ! end loop on irelax
```

Contents

Introduction

Common needs of solvers regarding meshes

Data structures

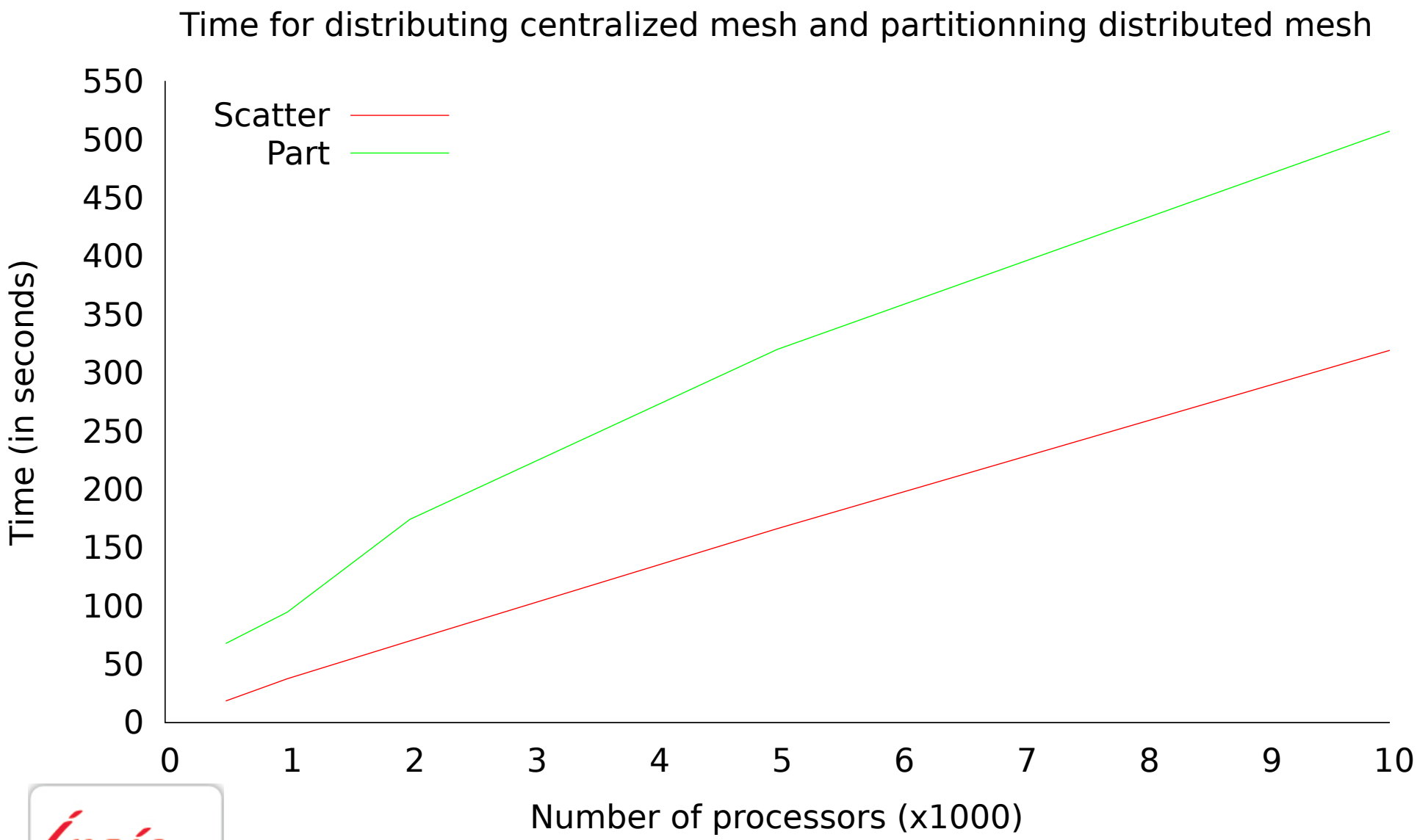
Version 0.1

Example: Laplacian equation using finite element method

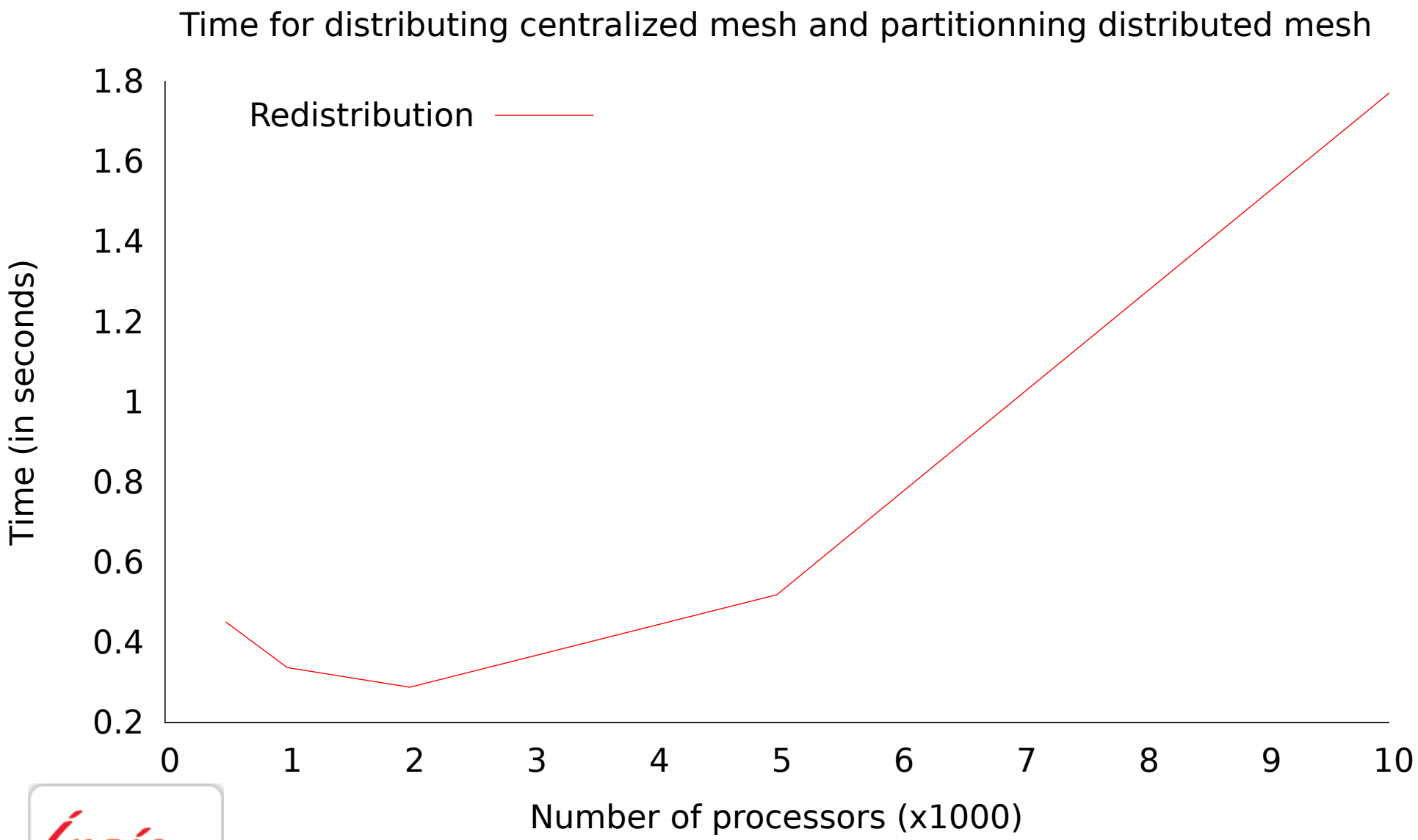
Some results

Upcoming features

First results (1/2)



First results (2/2)



Contents

Introduction

Common needs of solvers regarding meshes

Data structures

Version 0.1

Example: Laplacian equation using finite element method

Some results

Upcoming features

New features

- Main idea:
  - Parallel mesh adaptation
- Others:
  - Parallel I/O with HDF5
  - Overlap greater than 1
  - Unbreakable relations

Merci