ANEMOS

PaMPA : Parallel Mesh Partitioning and Adaptation

Cédric Lachat & François Pellegrini & Cécile Dobrzynski

January 31, 2013

Contents

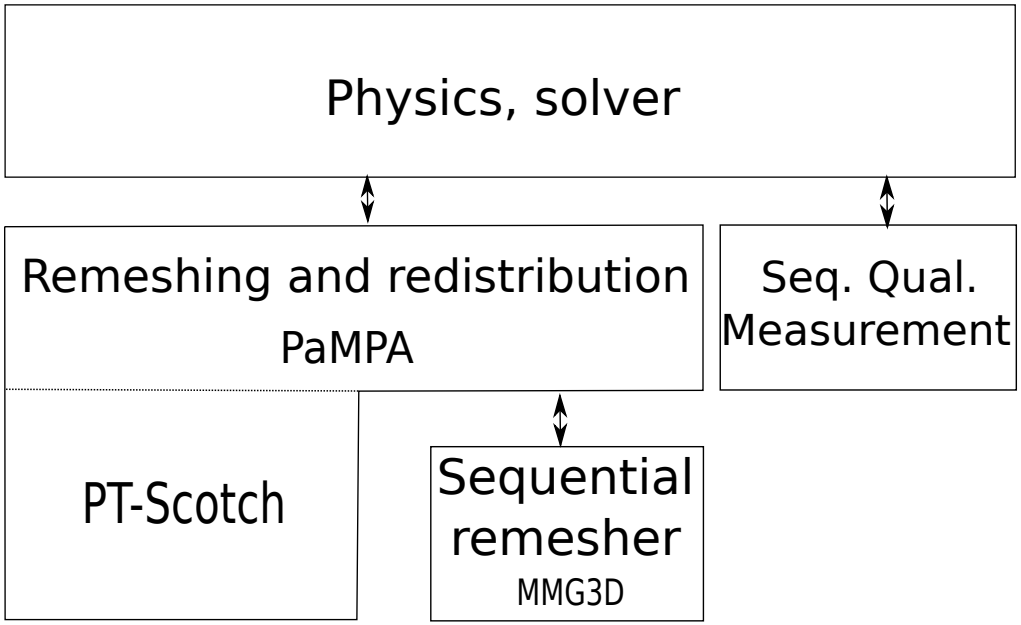
Common needs of solvers regarding meshes

- 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

What is PaMPA

- 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



Features of version 0.2

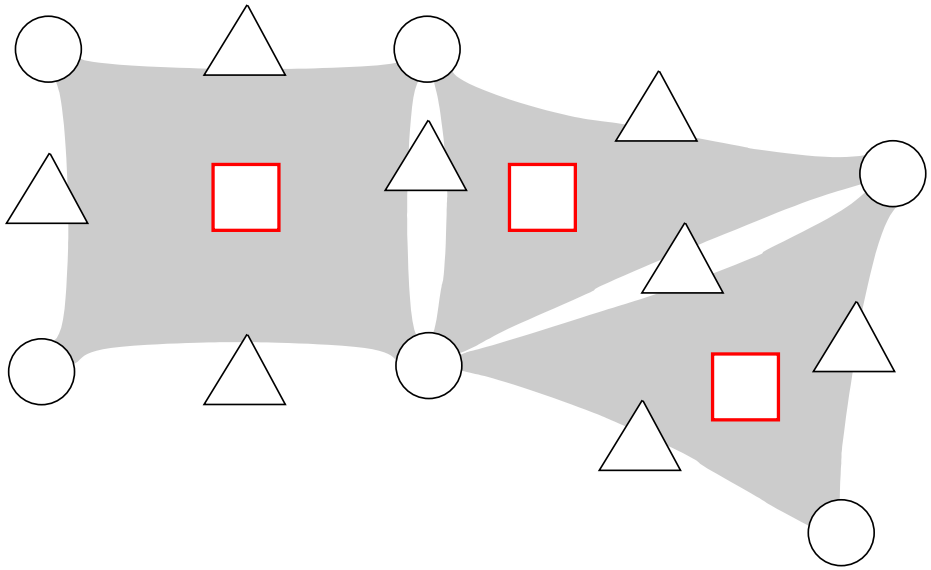
- Overlap greater than 1
- Parallel I/O
- Parallel partitioning
- Parallel mesh adaptation based on sequential remesher

Contents

Definitions

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

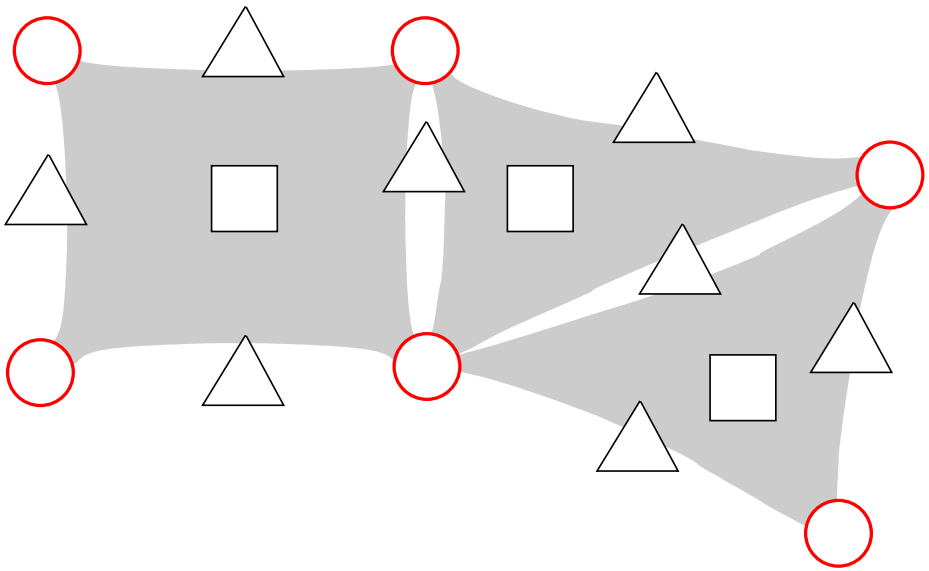
Top-level mesh entity
May bear some data (volume, pressure, etc.)



Definitions

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

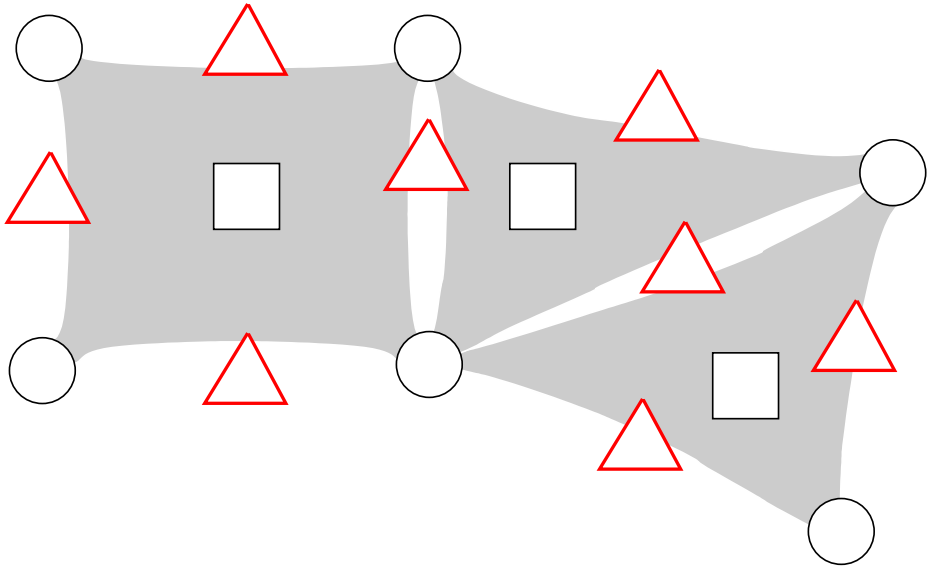
May bear some data (geometry, etc.)



Definitions

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

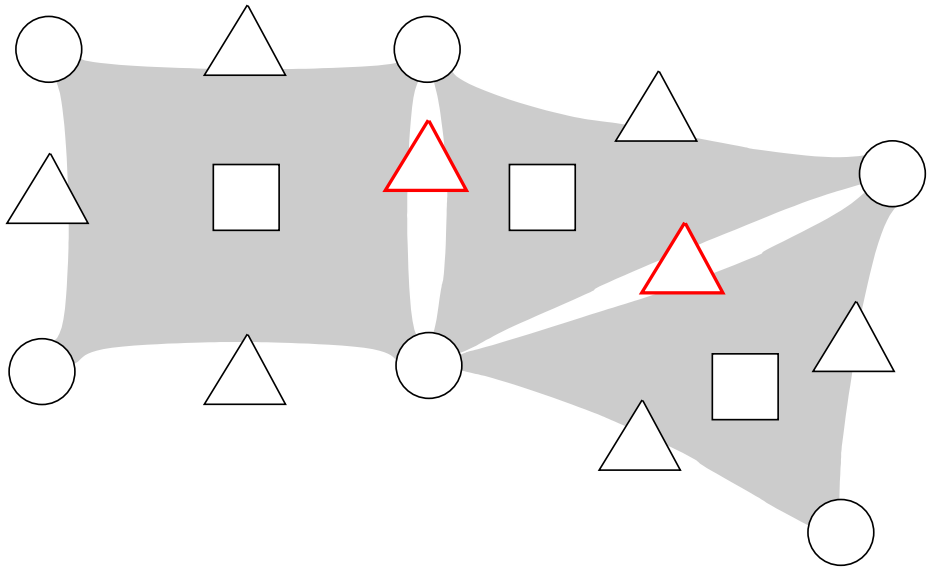
May bear some data (flux, etc.)



Definitions

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

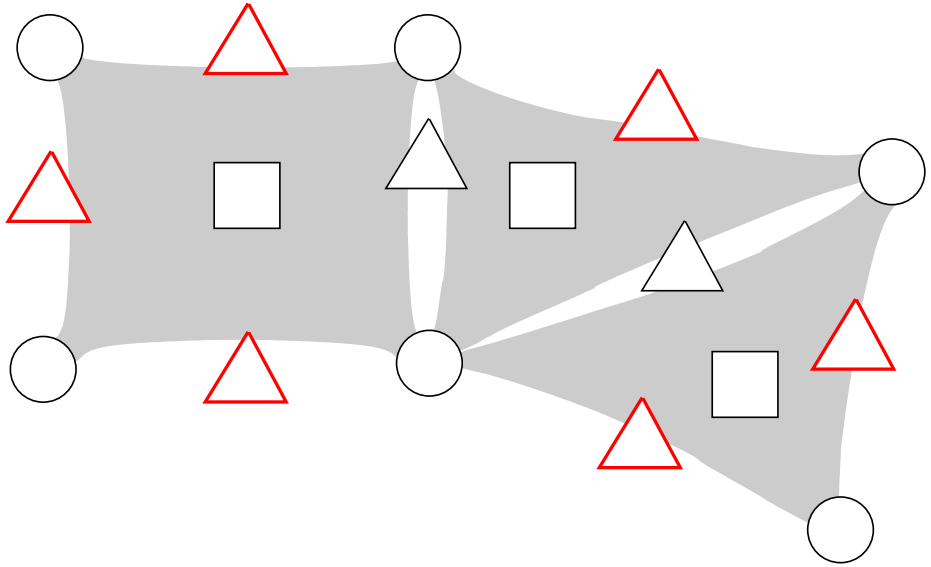
Regular mesh edge



Definitions

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

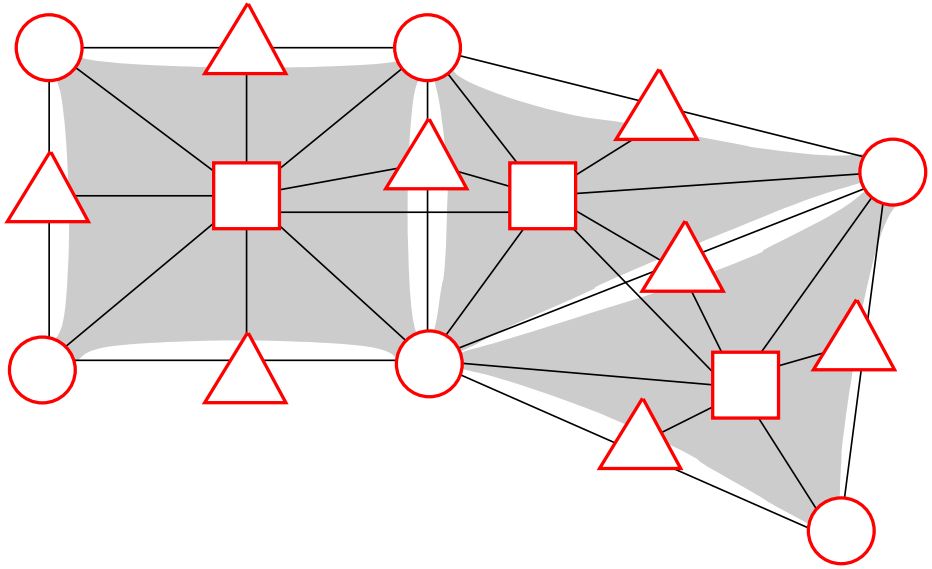
Boundary mesh edge



Definitions

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

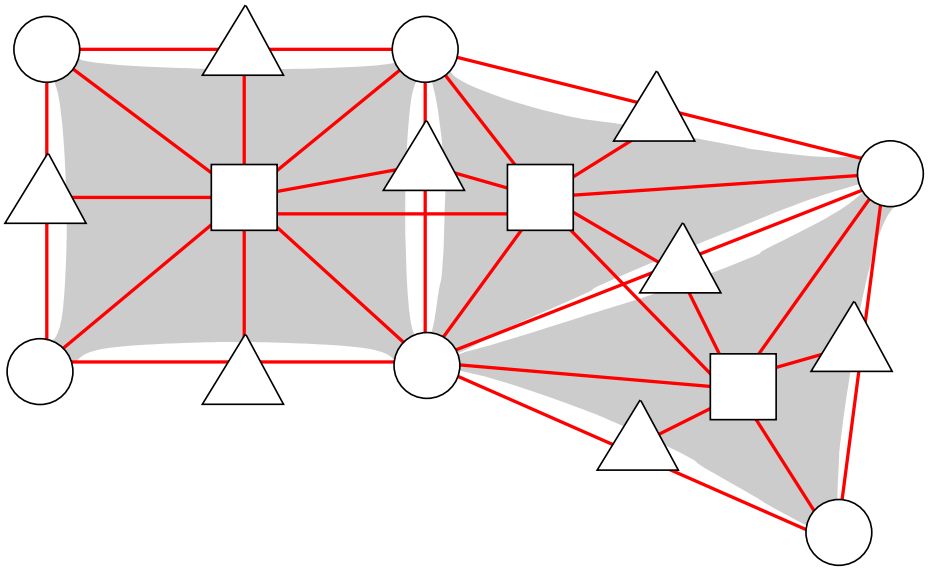
What all entities are in fact...



Definitions

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

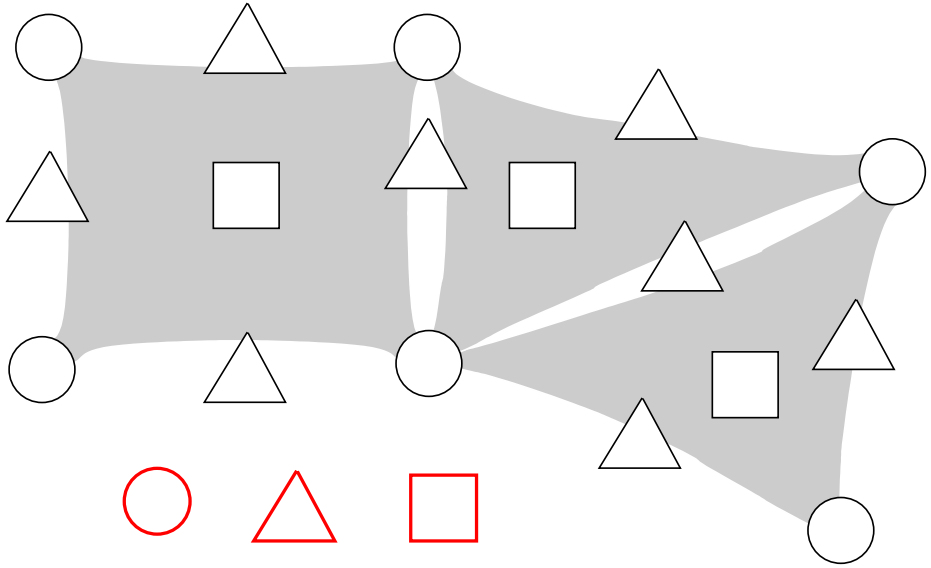
Subset of edges between vertices belonging to prescribed entity types



Definitions

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

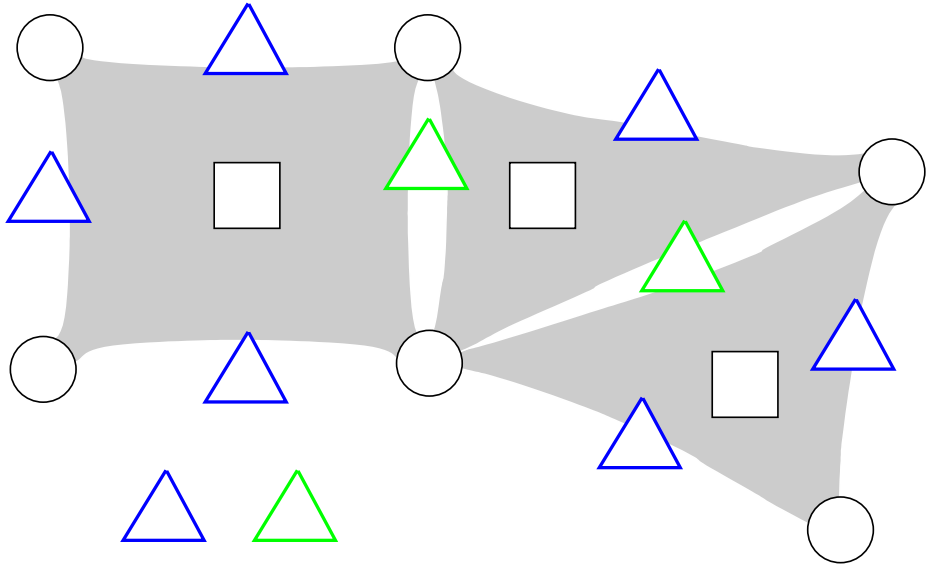
Subset of vertices bearing the same data



Definitions

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

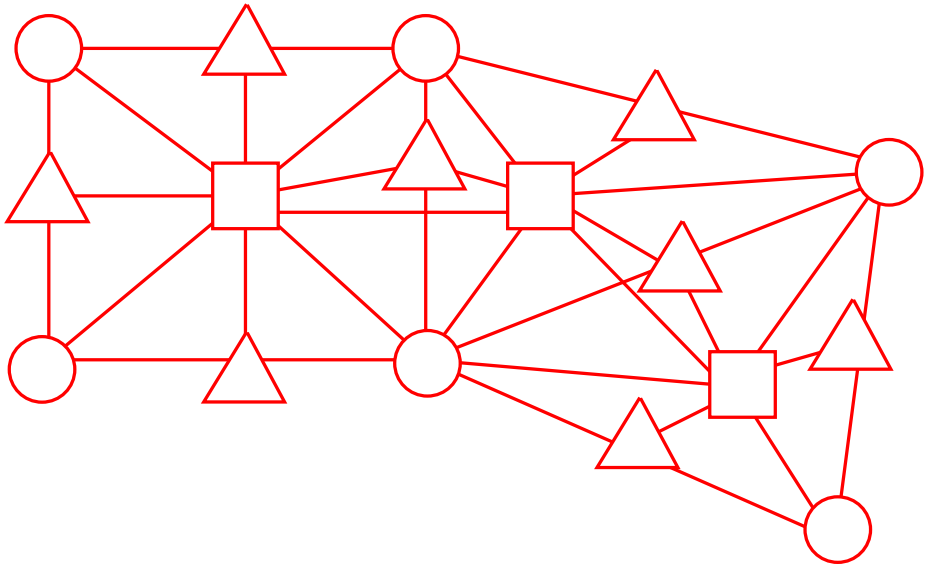
Subset of entity vertices that may bear additional specific data



Definitions

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

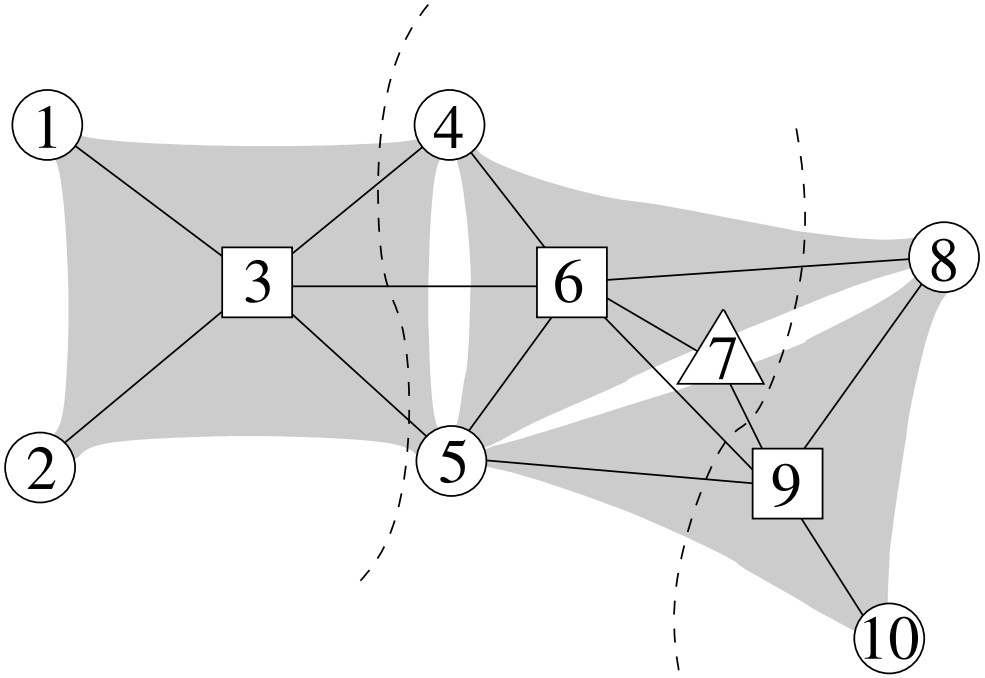
Whole set of vertices and relations
Every vertex belongs to one and only one entity (and sub-entity)



Global vue

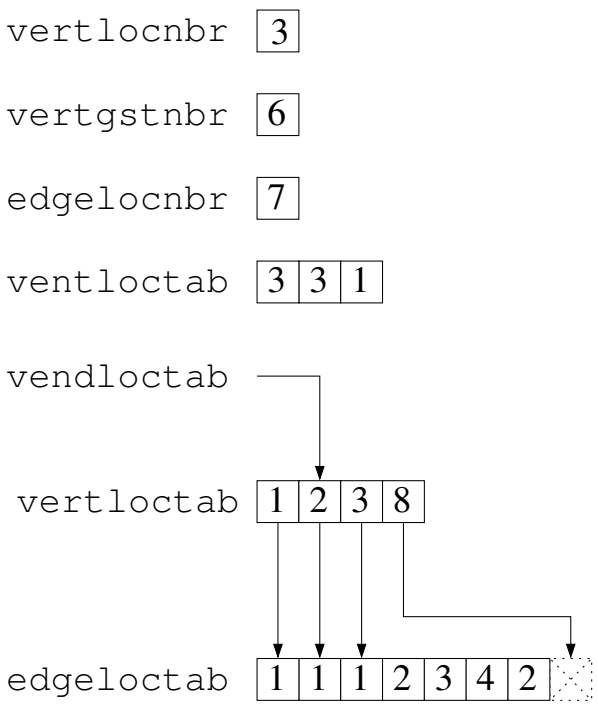
- All vertices have a global unique number

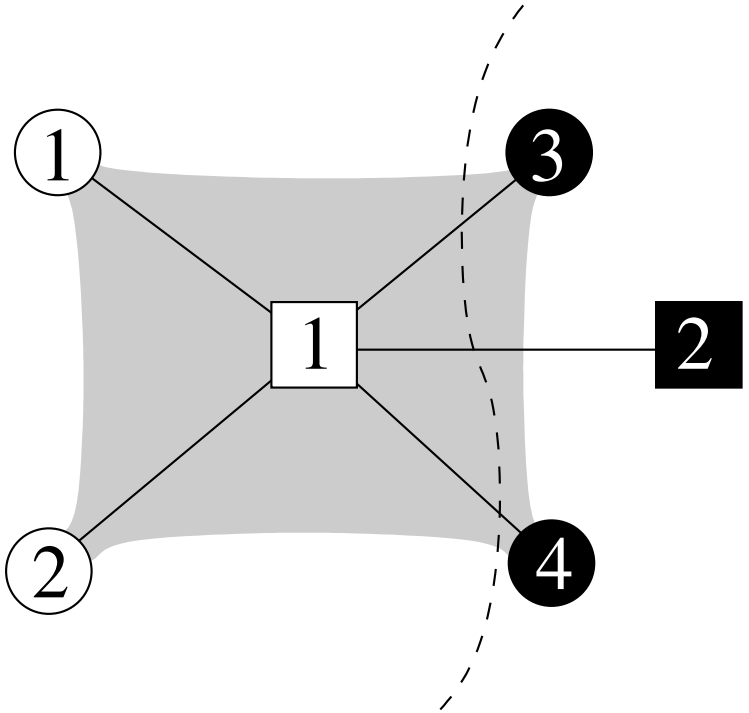
baseval	1
enttglbnbr	3
proccnttab	3 4 3
procvrttab	1 4 8 11



Local vision of process 0

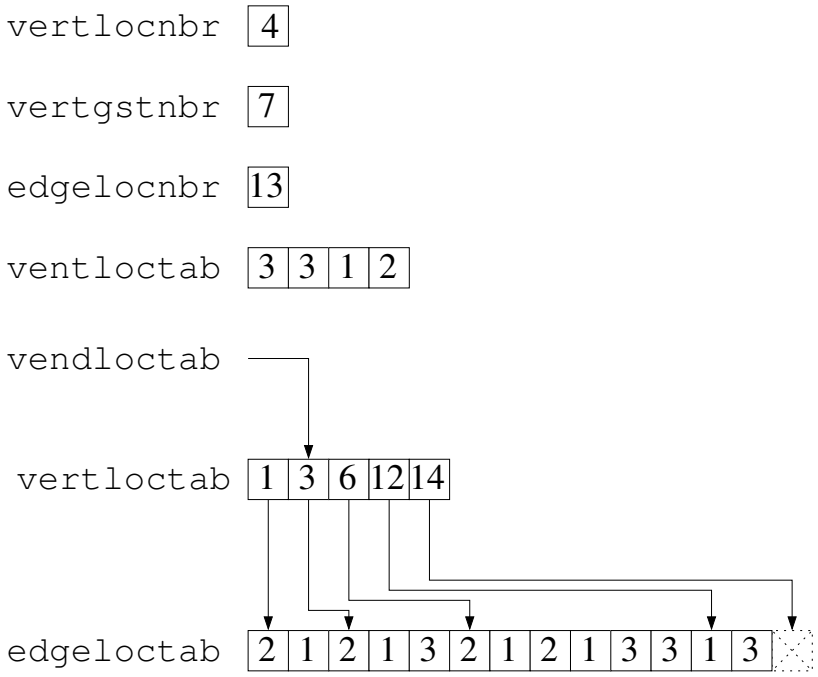
- All local and ghost vertices have a compact local index
  - Per-entity numbering

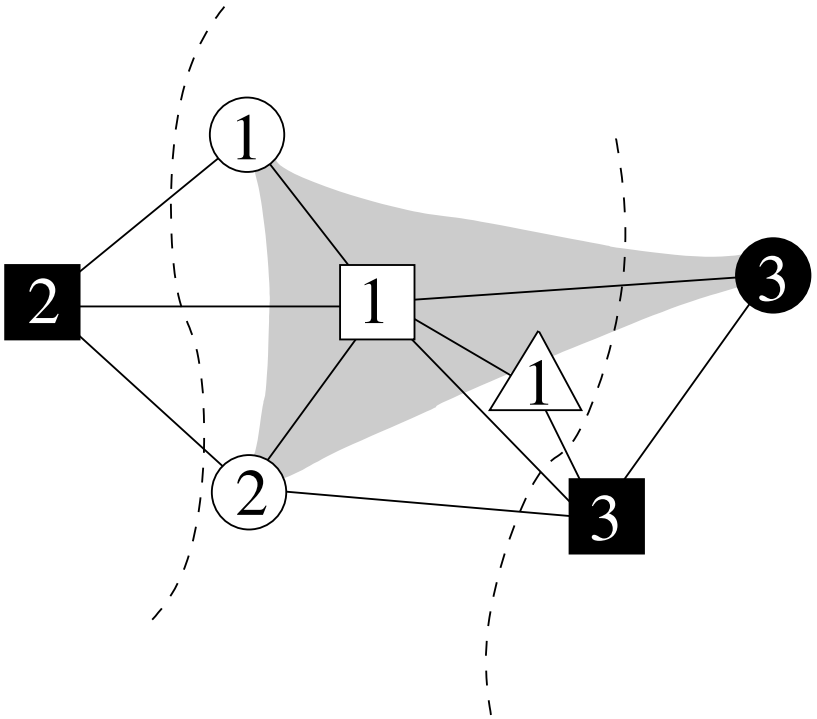




Local vision of process 1

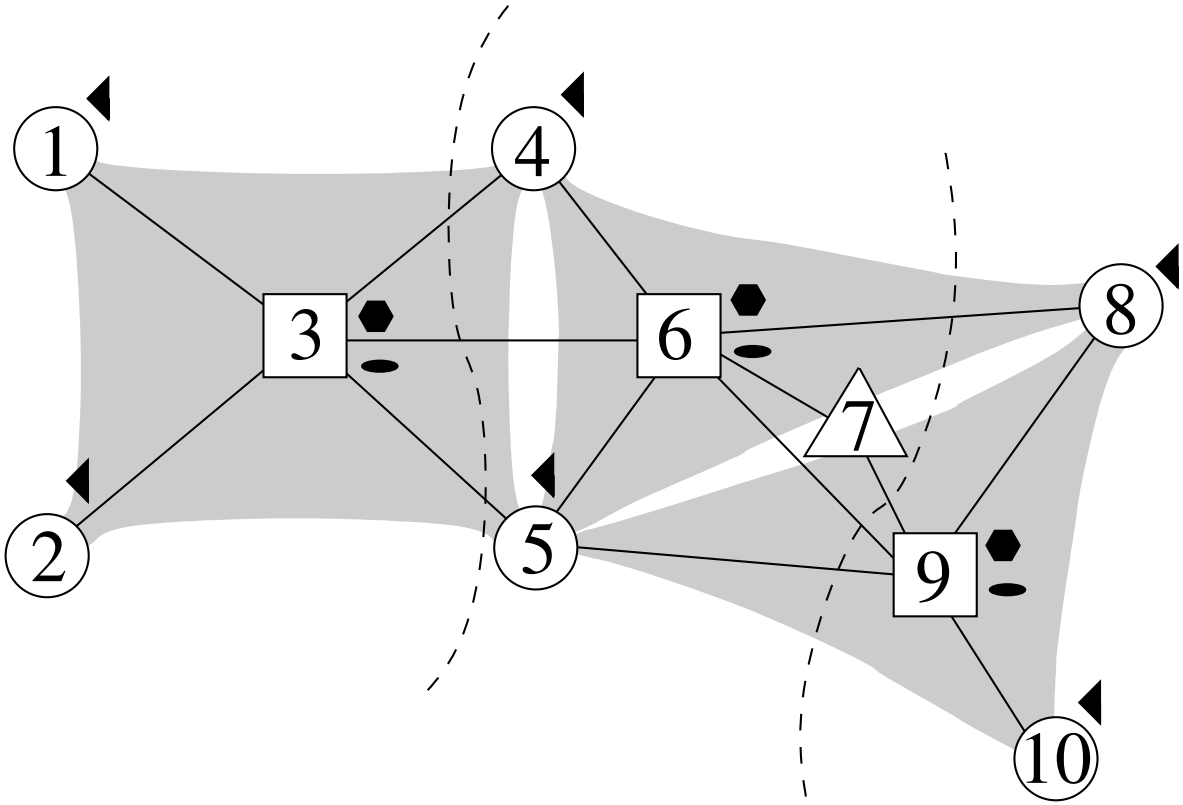
- All local and ghost vertices have a compact local index
  - Per-entity numbering





Linking values to entities

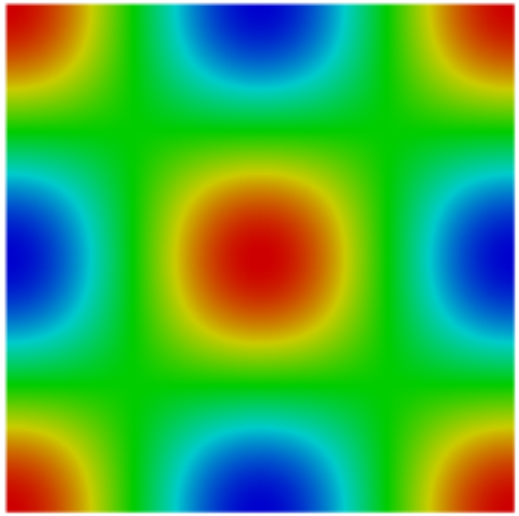
- Multiple value types can be associated with each entity
  - Value data structures can be split to improve cache usage
- Entities without values serve as iterators



Contents

Definition

- Solving 2D Poisson equation:
  - $\Delta u(x, y) = f(x, y)$
  - $g(x, y) = u(x, y)$ on the boundary Γ
- Test case:
  - $f(x, y) = -2 * \cos(x) * \cos(y)$ in the domain Ω
  - $g(x, y) = \cos(x) * \cos(y)$ on the boundary Γ
  - $u(x, y) = \cos(x) * \cos(y)$



Mesh properties

- Entities:
  - Elements
  - Nodes
  - Boundary edges
- Relations:
  - Element to element
  - Element to node
  - Element to boundary edge
  - Node to node
- Overlap of size 1
- Values:
  - Coordinates and solution on nodes
  - Type on boundary edges
  - Area, volume on elements

All steps

```
! On all processors:
CALL DistributedMesh() ! Build PaMPA distributed mesh:
!                        1- Read in parallel a centralized mesh
!                        2- Call PaMPA mesh partioner
!                        3- Redistribute distribute mesh
CALL ElementVolume()
CALL InitializeMatrixCSR()

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

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

FillMatrix

```
RHS = 0.
CALL PAMPAF_dmeshItInitStart (dm, ENTITY_ELEM, PAMPAF_VERT_ANY, it_vrt, ierr)
CALL PAMPAF_dmeshItInit (dm, ENTITY_ELEM, ENTITY_NODE, it_ngb, ierr)
DO WHILE (PAMPAF_itHasMore (it_vrt))
  jt   = PAMPAF_itCurEnttVertNum (it_vrt)
  Volt = VolEl (jt)
  ngb  = 0
  CALL PAMPAF_itStart (it_ngb, jt, ierr)
  DO WHILE (PAMPAF_itHasMore (it_ngb))
    ngb = ngb + 1
    is  = PAMPAF_itCurEnttVertNum (it_ngb)
    NuElemt(ngb)      = is
    CoorElemt(:,ngb) = Coor(:,is)
    PAMPAF_itNext (it_ngb)
  END DO
  CALL GradPhi (CoorElemt(:,1), CoorElemt(:,2), CoorElemt(:,3), GrdPhi)
  DO i = 1, Nsmplx
    is = NuElemt(i)
    DO j = 1, Nsmplx
      js   = NuElemt(j)
      JJac = Volt * Sum (GrdPhi(:,i) * GrdPhi(:,j))
      CALL assembly_addCSR (JJac,is,js)
    END DO ! loop on j
    RHS(is) = RHS(is) - Volt * SourceTerm (Coor(1,is), Coor(2,is)) / Nsmplx
  END DO ! loop on i
  PAMPAF_itNext (it_vrt)
END DO
```

Solve system: Jacobi (1/2)

```
UaPrec = 0.          ! Suppose A = L + D + U, system to solve : A x = b
CALL PAMPAF_dmeshItInit(dm, ENTITY_NODE, ENTITY_NODE, it_ngb, ierr)
DO irelax = 1, Nrelax
  res       = 0.
  CALL PAMPAF_dmeshItInitStart(dm, ENTITY_NODE, PAMPAF_VERT_BOUNDARY, it_vrt,  ierr)
  DO WHILE (PAMPAF_itHasMore(it_vrt))
    is = PAMPAF_itCurEnttVertNum(it_vrt)
    CALL PAMPAF_dmeshMatLineData(dm, ENTITY_NODE, is, I1, I1Fin, ierr)
    CALL PAMPAF_itStart( it_ngb,  is,  ierr)
    res0     = RHS(is)                                  ! res0  = b

    iv = i1
    DO WHILE (PAMPAF_itHasMore(it_ngb))
      js = PAMPAF_itCurEnttVertNum(it_ngb)
      PAMPAF_itNext(it_ngb)
      res0  = res0 - MatCSR%Vals(iv) * UaPrec(js) !res0  = b - (L + U) x^n
      iv = iv + 1
    END DO
    Ua(is)  = res0 / MatCSR%Diag(is)                    !x^n+1 = ( b - (L + U) x^n )/D
    PAMPAF_itNext(it_vrt)
  END DO

  CALL PAMPAF_dmeshHaloValueAsync(dm, ENTITY_NODE, PAMPA_TAG_SOL, req, ierr)
```

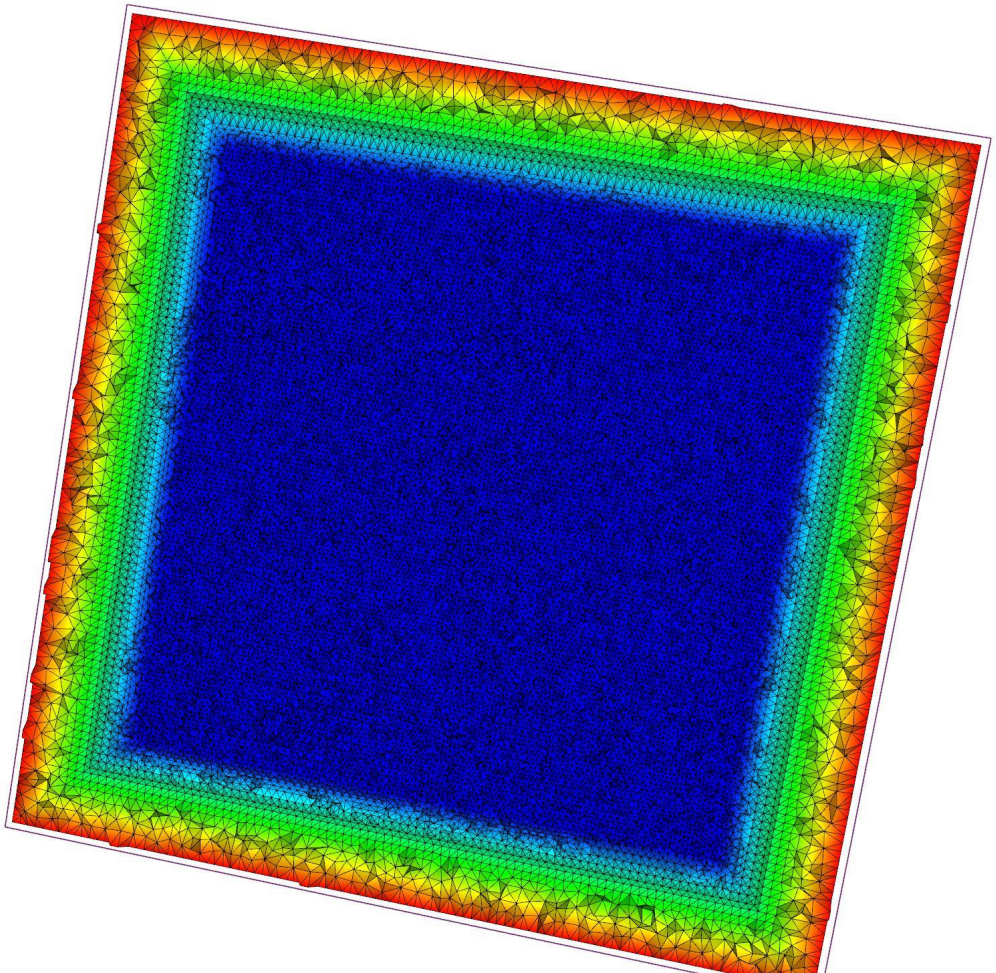
Solve system: Jacobi (2/2)

```
  CALL PAMPAF_dmeshItInitStart(dm, ENTITY_NODE, PAMPAF_VERT_INTERNAL, it_vrt , ierr)
  DO WHILE (PAMPAF_itHasMore(it_vrt))
    is = PAMPAF_itCurEnttVertNum(it_vrt)
    CALL PAMPAF_dmeshMatLineData(dm, ENTITY_NODE, is , I1 , I1Fin , ierr)
    CALL PAMPAF_itStart( it_ngb ,  is ,  ierr)
    res0     = RHS(is)                                  ! res0  = b

    iv = i1
    DO WHILE (PAMPAF_itHasMore(it_ngb))
      js = PAMPAF_itCurEnttVertNum(it_ngb)
      PAMPAF_itNext(it_ngb)
      res0  = res0 - MatCSR%Vals(iv) * UaPrec(js) !res0  = b - (L + U) x^n
      iv = iv + 1
    END DO
    Ua(is)  = res0 / MatCSR%Diag(is)                    !x^n+1 = ( b - (L + U) x^n )/D
    PAMPAF_itNext(it_vrt)
  END DO

  CALL PAMPAF_dmeshHaloWait(req , ierr)

  UaPrec     = Ua
END DO ! end loop on irelax
```

Contents

First results

- Adapt in parallel a mesh: 2M to 190M tetrahedra (on 64 to 256 processors)



Contents

Work in progress

- Release of version 0.2
  - Available soon from Inria Gforge
  - Licensed under GPL
- Quality of parallel adapted meshes
- Periodic meshes

Contents

Upcoming features

- Mesh definition with a grammar
- Face orientation and displacement
- Unbreakable relations
  - Partitioner will not cut these edges
  - E.g. to implement DG methods
- Multi-grid meshes
- Parallel I/O with HDF5
- Parallel mesh adaptation scalability

Thank you