Imperial College London

# Integrating DMPlex with Firedrake to enable scalable parallel I/O

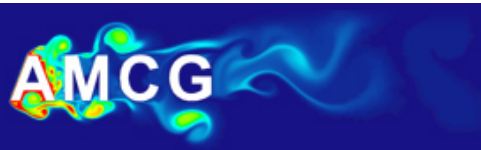
PRISM Technical Seminar Series

Michael Lange

AMCG, Imperial College London

May 29, 2014





Imperial College
London

# Unstructured Mesh I/O

**Doing unstructured mesh I/O is easy . . .**

**. . . we've done it a thousand times . . .**

- Multitude of mesh generators and formats:

  Gmsh, Cubit, Triangle, ExodusII, CGNS, SILO, . . .

- Various output formats and visualisation packages:

  VTK/VTU, Xdmf, ParaView, VisIt, EnSight, . . .

**. . . but doing it well is really hard!**

- Portability and interoperability
- Picking the right meta-data format
- **Parallel scalability**

# Unstructured Mesh I/O

The problem: No universally accepted format

> *I haven't met a file format that wasn't garbage, though some are less trashy than others...* *Jed Brown*

- So, everybody rolls their own, making interoperability between codes an even bigger nightmare.

As developers we only want to use a single API

- Multiple readers and writers to switch between file formats
- Comply with varying pre/post-processing toolchains

# PETSc Data Management

PETSc provides data management objects (DM)

- Provide an abstraction layer for mesh topology
  - Supports unstructured meshes (DMPlex)
  - Supports multiple mesh and I/O file formats
  - Decouples topology from field data
  - Multigrid: Can supply geometric data
- Manage parallel data layout
  - Create local and global data structures (Vec, Mat)
  - Integrated domain decomposition methods
  - Perform data migration (halo exchange)

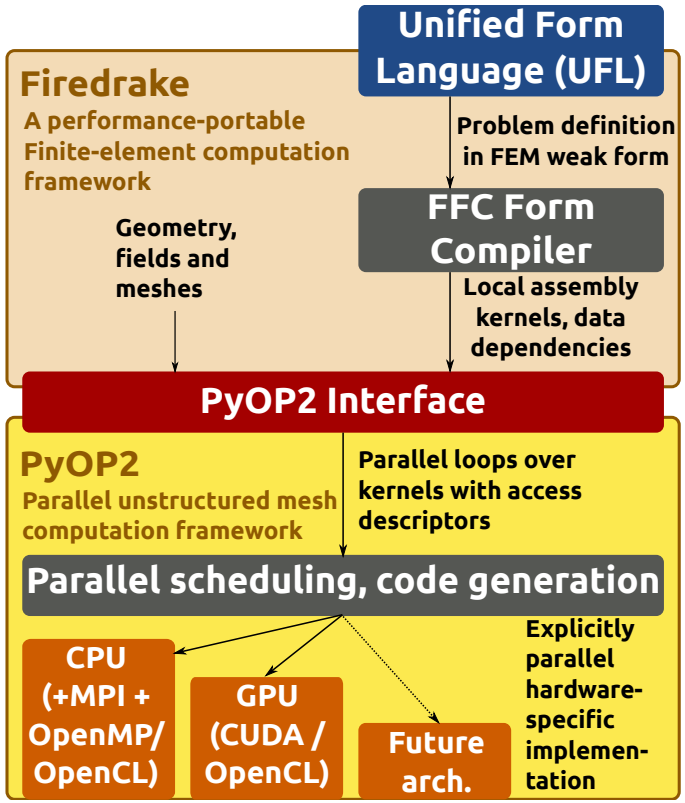
# Firedrake

Finite Element framework

- High-level abstraction (UFL)
- FE discretisation expressed symbolically

Performance Portability

- Low-level code generated at runtime
- Apply optimised kernel over domain in parallel loop



# The plan: Firedrake with DMPlex

Integrate DMPlex as meshing and topology component

- Replace legacy Fortran code
- Advantages:
  - Parallel decomposition on-the-fly
  - Can build simple meshes in memory (`UnitSquare/Cube`)
  - Access to mesh reordering techniques (RCM)
- Use DMPlex reader/writer routines to do I/O
  - Support multiple mesh formats
  - Parallel scalable solution output
  - Increase interoperability and portability!

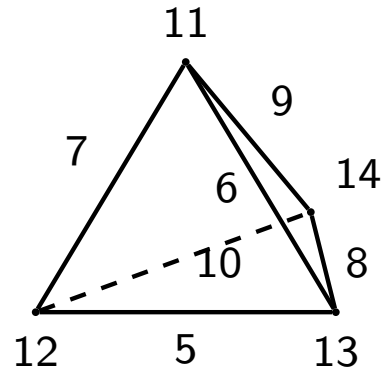
Imperial College
London

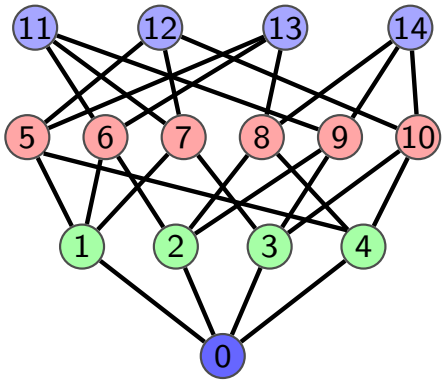
# PETSc Data Management

**DMPlex** - Topology abstraction

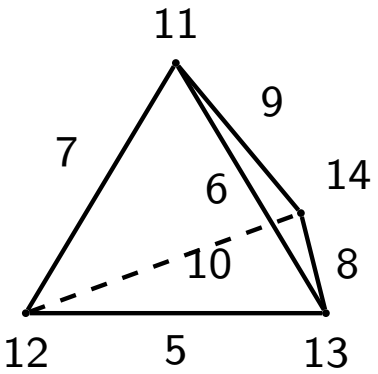
- Encodes topology in a graph
- Entities accessed by (co-)dimension

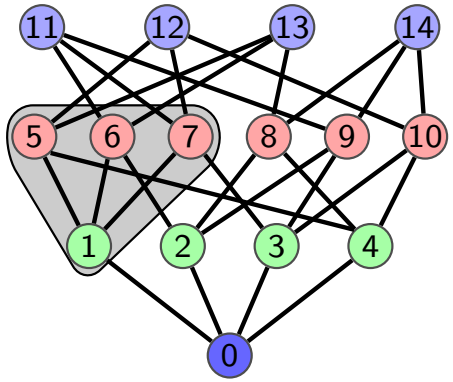
- Intermediate mesh representation allows support for various mesh formats:
  - ExodusII, CGNS
  - Gmsh (sequential)
  - `DMPlexCreateBoxMesh()`
  - `DMPlexCreateFromDAG()`
  - `DMPlexCreateFromCellList()`



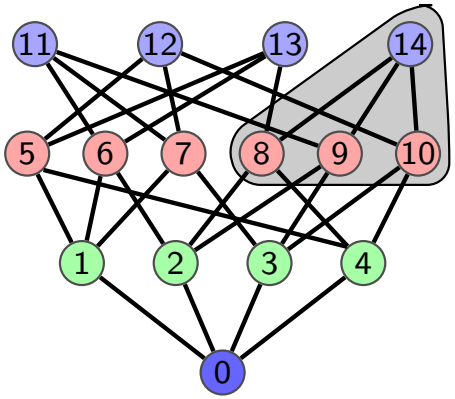



# PETSc Data Management

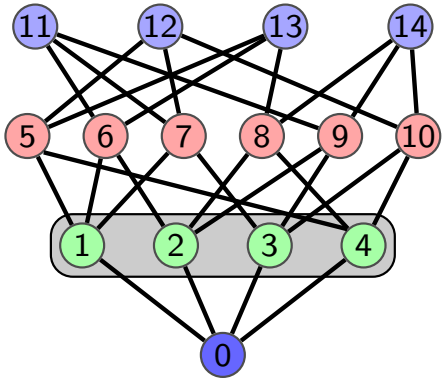


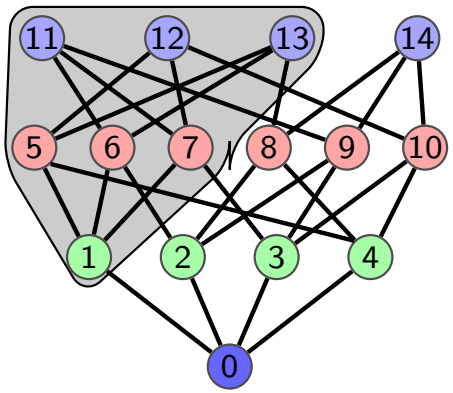

Cone(1)



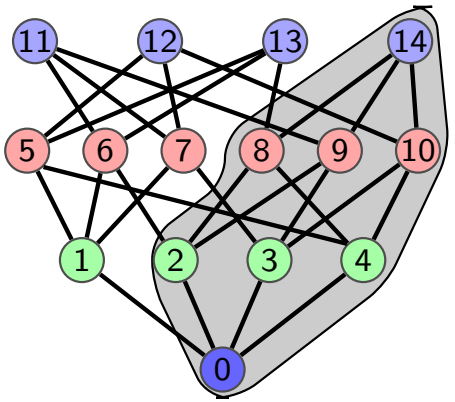

Support(14)




HeightStratum(1)




Closure(1)

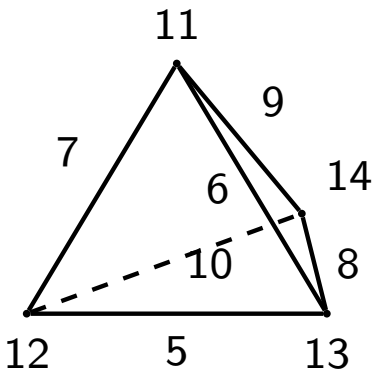


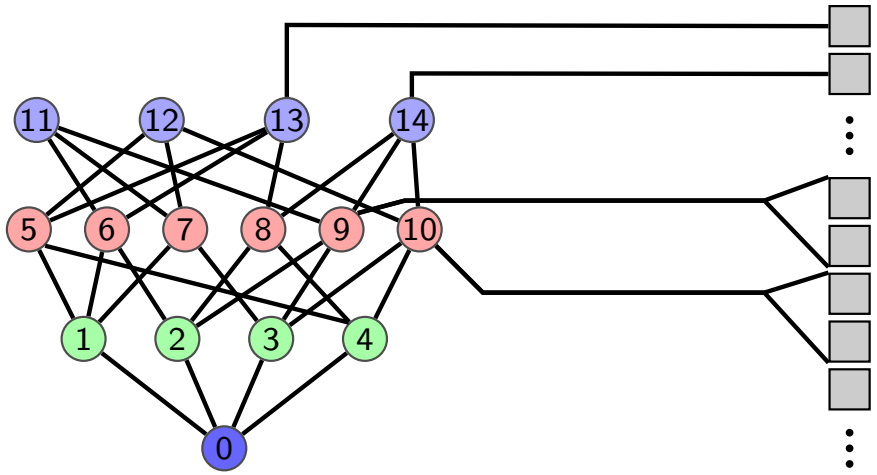

Star(14)

# PETSc Data Management

**Section** - Maps topology to Degrees of Freedom (DoFs)

- Index/offset pair for mapping into vector/array
- User defines number of DoFs in each (co-)dimension
- Describe DoF numbering (global and local)
- Can handle multiple fields per section (composable)



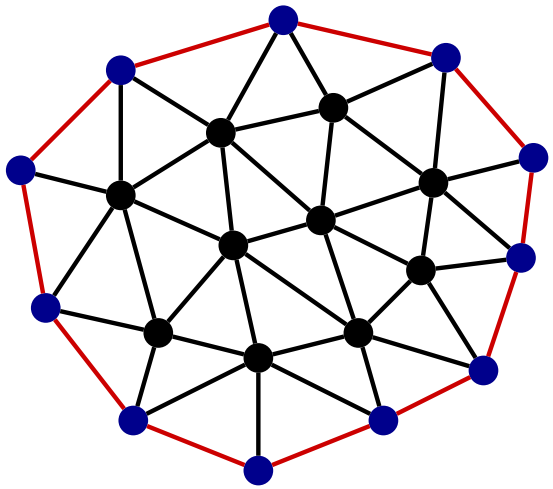



# PETSc Data Management

**DMLabel** - Mark sets of points

- Mark boundaries, regions, etc.
- Fast access to sub-sets of entities
- Filter points by label value

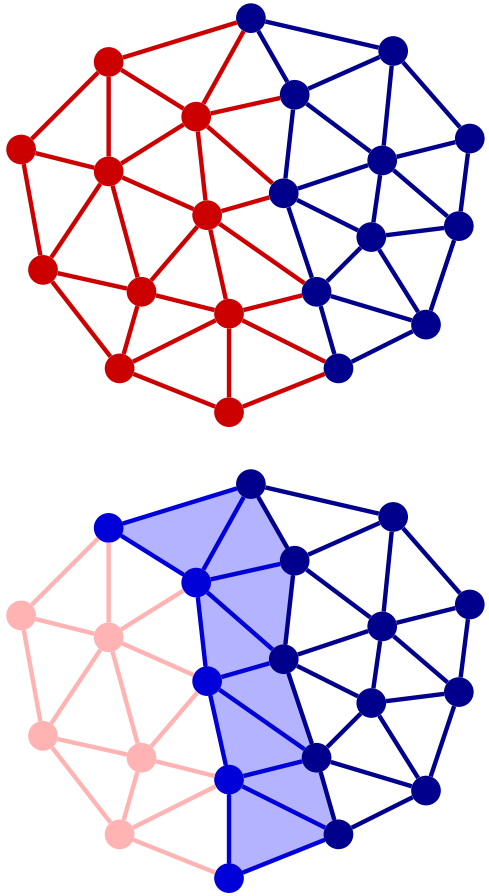
- `DMPlexMarkBoundaryFaces()`



# PETSc Data Management

**Parallel data movement**

- On-the-fly domain decomposition
  - Chaco
  - Metis/ParMetis
- Data migration
  - `DMPlexDistribute()`
  - `DMGlobalToLocal()`
- **PetscSF** - Star Forest
  - Maps local DoFs (leaves) to remote data (roots)
  - Communication routines: Gather/scatter, bcast, reduction

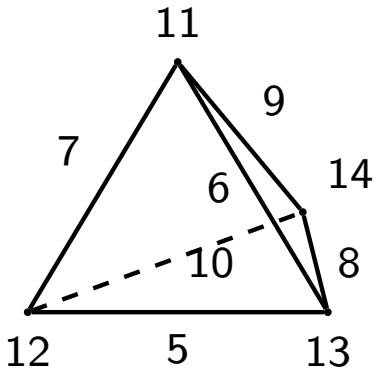


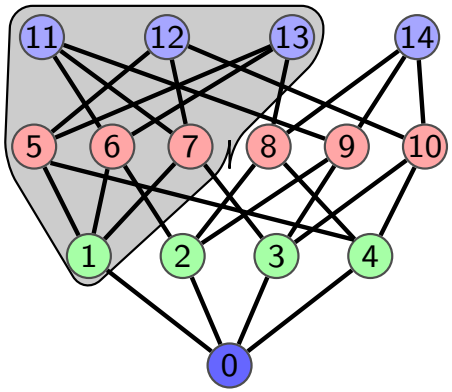
Imperial College
London

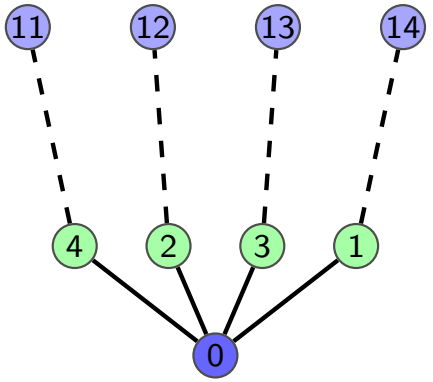
# Firedrake Integration

Challenge: Build Firedrake data structures

- Local numbering according to Fenics rules
  - Fenics rules: Local facet numbering based on lexicographical ordering of non-incident vertices
  - Requires reordering DMPlex's cell closures



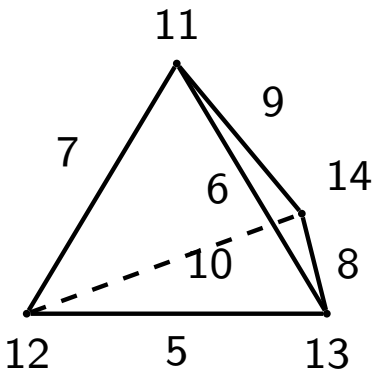



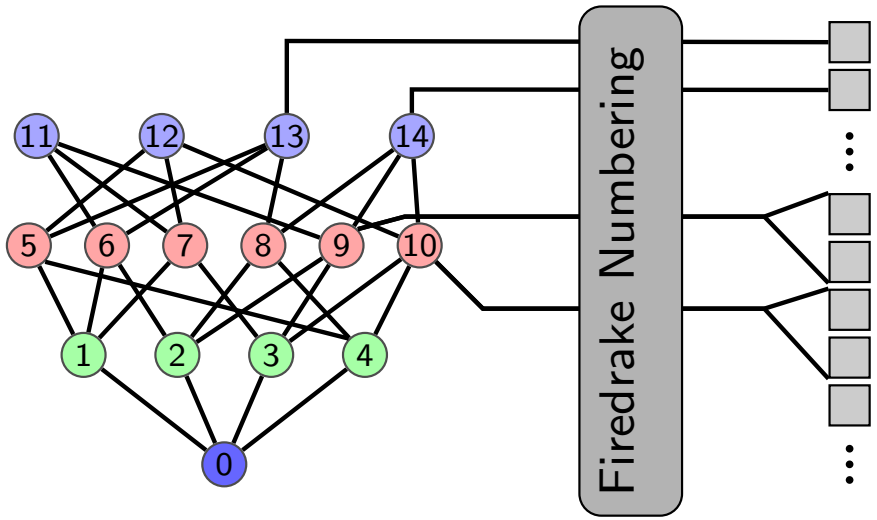



# Firedrake Integration

Challenge: Build Firedrake data structures

- Firedrake requires a specific global entity numbering
  - Build Firedrake numbering as permutation
  - Attach permutation to DMPlex object
  - Inherit permutation in all subsequent operations

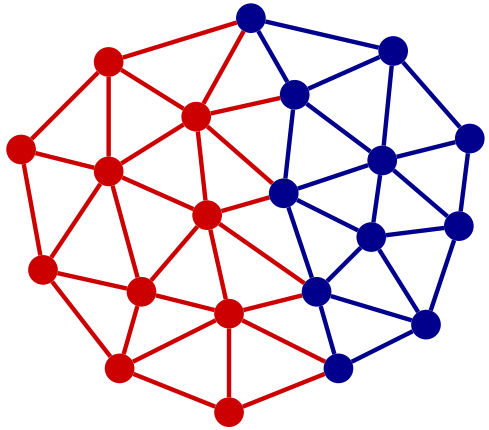


# Firedrake Integration

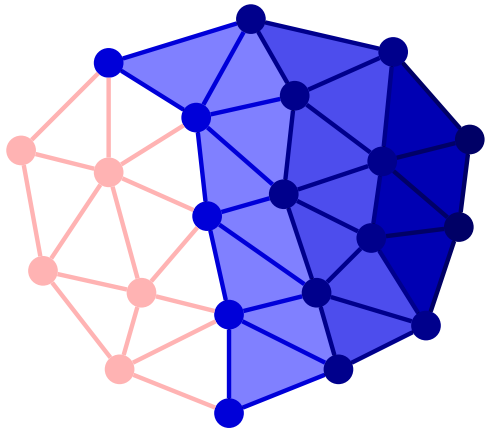
OP2 entity class ordering

- Core: No halo data required
- Owned: Requires halo update
- Halo: Remote contribution



Overlap communication with computation

- Staged kernel execution
- Maps are built for each entity class
- Mark classes on topology using DMLabels



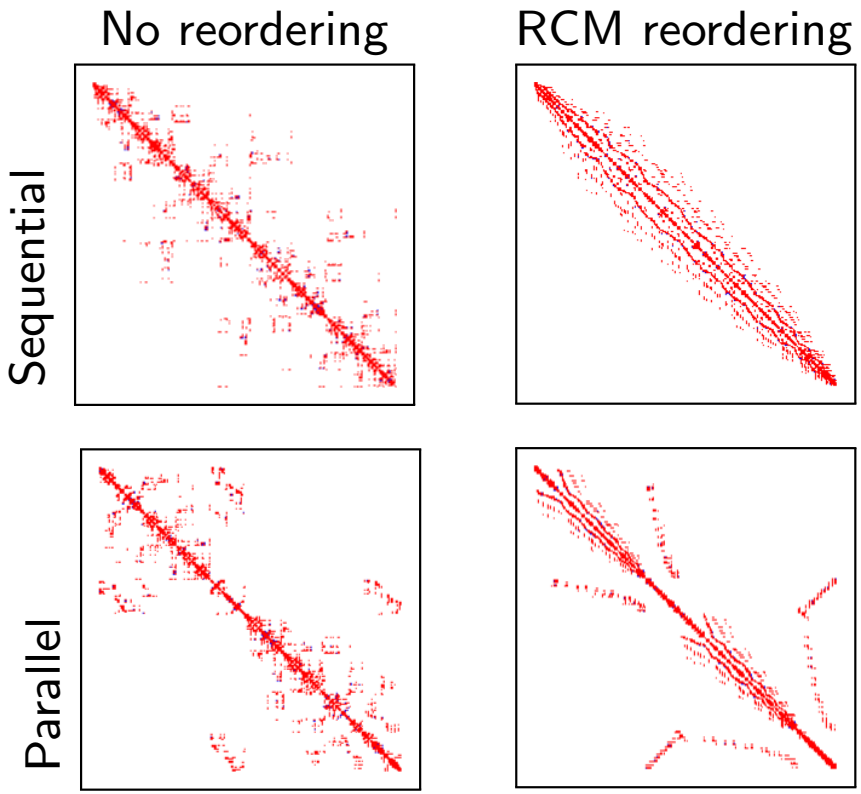
# Firedrake Integration

Mesh reordering

- Improves cache coherency
- DMPlex provides Reverse Cuthill-McKee (RCM)

Firedrake inherits reordering from DMPlex

- Extract cell reordering
- Apply *within* OP2 classes
- Make cell DoFs contiguous



Imperial College
London

# Unstructured Mesh I/O

DMPlex improves interoperability with new mesh interfaces:

| Format | Supported | Parallel | Requires |
|---|---|---|---|
| ExodusII | Both | Potential | NetCDF |
| CGNS | Both | Potential | (HDF5) |
| Xdmf | DMPlex | Potential | HDF5 |
| Gmsh (Ascii) | Both | No | |
| Triangle | Firedrake | No | |
| ... | | | |

**What about solution output and visualisation?**

# Unstructured Mesh I/O

The output format wishlist (taken from petsc-dev)

- Well documented and widely adopted
- Compatible with most post-processors/visualisation tools
  - Ability to read/write individual fields
  - Support for higher order elements
- Works as a checkpoint
  - Preserve boundary and sub-domain markers
  - Ability to reopen file on a different CPU count
- Parallel scalability
  - Single output file
  - Concurrent reads/writes (eg. HDF5)

# Unstructured Mesh I/O

The aim: A parallel, visualisable checkpointing cycle

- HDF5 for parallel output
  - Single output file with concurrent writes
- Xdmf for visualisation
  - Compatible with ParaView
  - PETSc provides script to write XML header
- Checkpointing
  - Write Plex data to enable re-start
  - Preserves DMLabels (boundary/region IDs)
  - Requires parallel read...

# Unstructured Mesh I/O

Parallel read is trickier than write

1. Read and distribute graph connectivity
2. Read mesh/field data in slabs (parallel)
3. Re-partition based on real topology

Currently no parallel re-partitioning

- Instead read serial and distribute
- Parallel requires closer ParMetis integration
- Work in progress...

Imperial College
London

# Conclusion

Firedrake now runs on DMPlex

- Much improved portability
- Increased interoperability
- Performance optimisations
  - Parallel domain decomposition
  - Mesh reordering

Parallel output via Xdmf/HDF5

- Prototype exists; work in progress . . .
- Will soon add checkpointing capability

Future work

- Fluidity + DMPlex: Different challenge, same solution?
- Parallel reads and re-partitioning

# Discussion

**Can we use DMPlex to sync I/O across the PRISM platform?**

**What makes a good output file format?**

**What mesh formats should be supported?**

| Format | Supported | Parallel | Requires |
|---|---|---|---|
| ExodusII | Both | Potential | NetCDF |
| CGNS | Both | Potential | (HDF5) |
| Xdmf | DMPlex | Potential | HDF5 |
| Gmsh (Ascii) | Both | No | |
| Triangle | Firedrake | No | |
| ... | | | |