The Goal

- **Goal:** “Make it easier to develop real-world scientific computations with irregular domain decompositions”
- **Proposal:** DMI (Distributed Memory Interface)
  - A parallel and distributed programming framework based on a **global address space** model

What is DMI?

- A multi-threaded global address space framework for HPC applications
- A shared library for C
- Basic APIs:
  - `addr = DMI_mmap(size, num)`: Allocate the global address space with `num` pages of (not OS page size but arbitrary) `size` bytes in size
  - `Page-based consistency`
  - `DMI_read(addr, size, buf), DMI_write(addr, size, buf)`: Read/Write `size` bytes from/to the global address space `addr` to/from a local memory `buf`
- Other features (beyond this poster’s scope):
  - Support for dynamic joining and leaving of nodes
  - Live thread migration for automatic load balancing
  - APIs for optimizing data locality explicitly

What is the Problem?

- **Application:** Real-world Finite Element Methods (FEM)
  - An irregular domain decomposition and ordering
  - Titanium[1], Global-Arrays[2] and XcalableMP support only regular domain decompositions
- **Algorithm:** Repeat iterations until convergence
  1. Obtain the values of ghost points
  2. Store the values of interior points and the ghost points in a local buffer according to some suitable ordering
  3. Update the values of the interior points using a given connectivity
- **Programming bottleneck:** Exchanging the values of the ghost points
  - A programmer has to calculate the correspondence between global indexes and local indexes:
    1. Obtain the values of the ghost points
    2. Store the values of the interior points in a local buffer according to some suitable ordering
    3. Update the values of the interior points using a given connectivity
- **Performance Evaluation**
  - Environment: 8 cores × 16 nodes, 10 Gbit Ethernet
  - Experiments:
    1. Stress analysis using an FEM
    2. A real-world and hard-to-converge problem
    3. DMI vs the champion MPI program of the parallel programming contest
  - Results:
    - DMI is easier to program than MPI
    - Performance: `mpich2` vs `DMI` vs `OpenMPI`

DMI Read/Write

```
void DMI_read(size_t addr, size_t size, void *buf)
{
    // Implementation...
}

void DMI_write(size_t addr, size_t size, void *buf)
{
    // Implementation...
}
```

Internal implementation:

- Manage point values efficiently in a manner similar to local-view programming by **internally transforming the global indexes specified by a programmer into the local indexes**
- Aggregate multiple internal communications for the same processor into one message

Performance Evaluation

```
0 5 10 15 20 25
0 20 40 60 80 100 120 140
```

References
