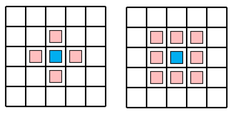
Algorithm Design for High Performance CFD Solvers on Structured Grids

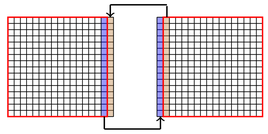
Hengjie Wang, Aparna Chandramowlishwaran (Advisor), *HPC Forge Lab, University of California-Irvine*

Motivations

Computational Fluid Dynamics (CFD) with structured grids has been widely utilized in many engineering disciplines such as Aerospace Engineering, Vehicle Design, etc. Its computation and communication are characterized by stencils and halo exchange.



Star and Box Stencils



Halo Exchange

We identify the following key performance limitations in start-of-the-art CFD algorithms and solvers:

- **Multi-Block Structured Grid Partitioner**
  - Biased towards minimizing communication volume
  - Use graph partitioner for complex structured grid
- **Distributed Stencil Computation**
  - Temporal tiling is not directly applicable to multi-block grids on distributed-memory systems
  - Limited overlap of computation and communication with temporal tiling
- **CFD + Deep Learning**
  - Problem-specific surrogate, i.e., unable to predict flow over unseen geometries in training

Contributions

Multi-Block Structured Grid Partitioner

- ✓ Unify key algorithmic knobs and network properties into one cost function
- ✓ Design new partition algorithms for structured grids
- ✓ Outperform state-of-the-art methods by 1.5-3x

Distributed Stencil Computation

- ✓ Optimal hybrid temporal tiling, up to 1.9x over Pluto
- ✓ Pipeline computation and communication
- ✓ Applicable to multi-block structured girds
- ✓ 1.3-3.4x over MPI-Funneled with space tiling on distributed machines

CFD + Deep Learning

- ✓ Extract local flow patterns via deep learning
- ✓ Predict flow over arbitrary geometries

Multi-Block Structured Grid Partitioner (*ICS19*)

- Unify the network properties, edge cuts (# messages) and communication volume into one cost function

$$\text{cost} = \text{latency} \cdot \text{cuts} + \frac{\text{volume}}{\text{bandwidth}}$$

Messages Generated by Partitioner

Network Properties

- Design new algorithms for partitioning

Cut large blocks: → **Group small blocks**

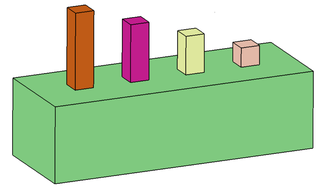
With new cost function:
- *REB*: Recursive Edge Bisection
- *IF*: Integer Factorization

Novel algorithms:
- *CCG*: Cut and Combine blocks Greedily
- *GGS*: Sweep blocks with Graph Growth methods

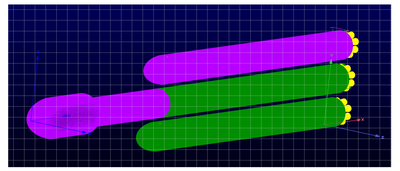
- Performance Evaluation

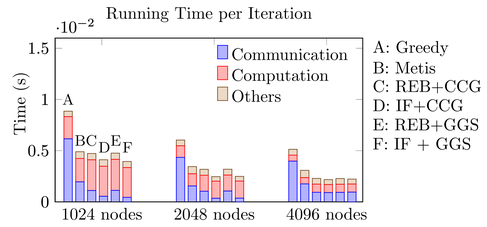
Evaluate the update of a Jacobi solver with Bump3D and a rocket model based on SpaceX's Falcon-Heavy on the Mira Supercomputer at the Argonne National Lab

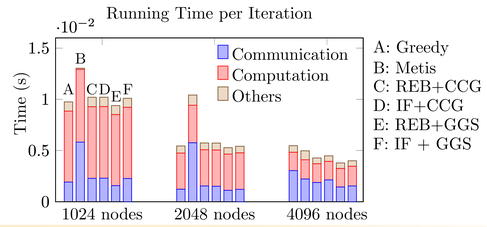
- Bump3D: 1 large block and 4 small blocks



- Rocket Model: 769 blocks of various sizes







Distributed Stencil Computation (*SC20*)

- Optimal Hybrid (MPI+OpenMP) Temporal Tiling
  - MPI and OpenMP's memory arrangement

The optimal decomposition of $2 \times N_j \times N_k$ by 2 processes or 2 threads. The *J-K* plane is 2x larger for threads.

  - Temporal Tiling

Time-Space Tile

Fuse 5 iterations (5 colors) and reuse *J-K* planes while they reside in cache. Small *J-K* planes are easy to fit in cache.

  - MPI+OpenMP Temporal Tiling

Decompose *K* with processes → **Decompose *J* with threads** → **March in *I* with Temporal Tiling**

Reduce the size of *J-K* planes per process

Apply prefetching and SIMD to *K*

Reuse *J-K* planes

The performance is evaluated across 6 numerical schemes and compared to space tiling and Pluto (diamond tiling):

WJ7: 7pt weighted Jacobi for $\nabla^2 p = b$
WJ13: 13pt weighted Jacobi for $\nabla^2 p = b$
WJ27: 27pt weighted Jacobi for $\nabla^2 p = b$
Upwind: 2nd order upwind for $\partial_t \phi + \vec{u} \cdot \nabla \phi = 0$
Weno3: 3rd order WENO for $\partial_t \phi + \vec{u} \cdot \nabla \phi = 0$
Burgers: CD for $\partial_t \vec{u} + \nabla \cdot (\vec{u}\,\vec{u}) = \nabla^2 \vec{u}$

(Pluto was unable to tile Burgers equation.)

- Overlap Communication and Computation
  - Divide the domain into chunks and pipeline the chunks' execution

  - Strong and weak scalability on 16-128 Broadwell nodes

- Support Multi-Block Structured Grids
  - Use *DeepHalo* to break the inter-block dependency that prohibits temporal tiling for multi-block structured grids

```
//traverse time
for (int t=0;t<tEnd;++t){
  //traverse blocks
  for (int b=0;b<nBlocks;++b){
    get_block_size(b, size);
    //traverse space
    for (int i=0;i<size[0];++i)
      for (int j=0;j<size[1]; ++j)
        for (int k=0;k<size[2] ++k)
          compute(b, i, j, k);
  }
  //blocks' connections
  for (int b=0;b<nBlocks;++b)
    exchange_boundary(b, nHalo);
}
```

Inter-block dependency

```
//traverse time
for (int t=0;t<tEnd;t+=tFused){
  // fused iterations
  for (int tt=0; tt<tFused; ++tt ){
    //traverse blocks
    for (int b=0;b<nBlocks;++b){
      get_block_size(b, size);
      //traverse space
      for (int i=0;i<size[0];++i)
        for (int j=0;j<size[1]; ++j)
          for (int k=0;k<size[2] ++k)
            compute(b, i, j, k);
    }
  }
  //update deep halo
  for (int b=0;b<nBlocks;++b)
    exchange_boundary(b, nHalo*tFused);
}
```

DeepHalo

- Evaluate with a 6-block grid on 32 Broadwell nodes
  Report Speedup over MPI-Funneled with space tiling

CFD + Deep Learning (*ongoing*)

- Generalize Networks to Geometries Unseen in Training
  - Train the network to solve $\nabla^2 p = 0$ with data and physics

$$loss = \frac{1}{N}\Big(\sum (p - p^*)^2 + \alpha \sum (\nabla^2 p)^2\Big)$$

Multigrid + Finite Difference

  - Transfer predictions across different geometries

Ground Truth | Prediction, RMAE ~4% | Ground Truth | Prediction, RMAE ~6%

- Next Steps: Simple Flow Features → Complex Flows
  - Complex flows are composed by simple flow features

  - Learn simple flow features

Lid-Driven Cavity + *Channel Flow*

  - Predict for a more complex flow

Transfer

Backward facing step