Load Balancing and Data Migration in a Hybrid Computational Fluid Dynamics Application

Esteban Meneses

Patrick Pisciuneri

Center for Simulation and Modeling (SaM)

University of Pittsburgh

University of Pittsburgh

High Performance Computing

Computer Science

Scientific Computing

Center for Simulation and Modeling (SaM)

HPC researchers/consultants

★ ★ ★ ★ ★

Educational

Research

Technical

521 users

Sciences

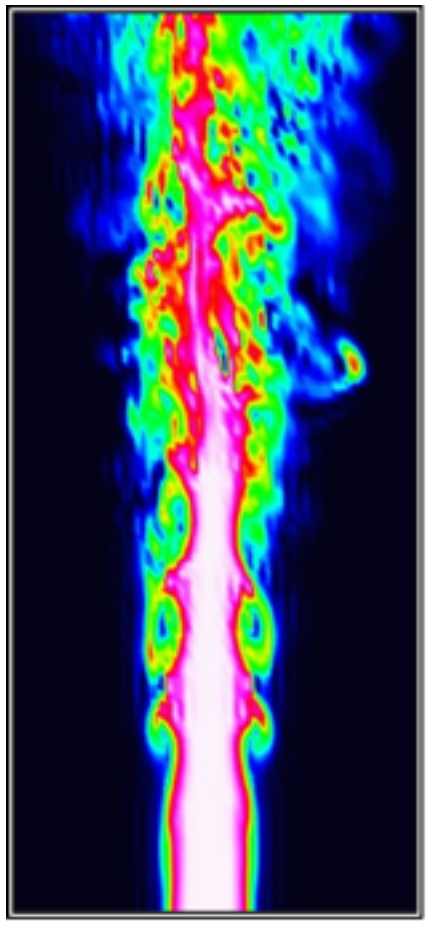
Health

Engineering

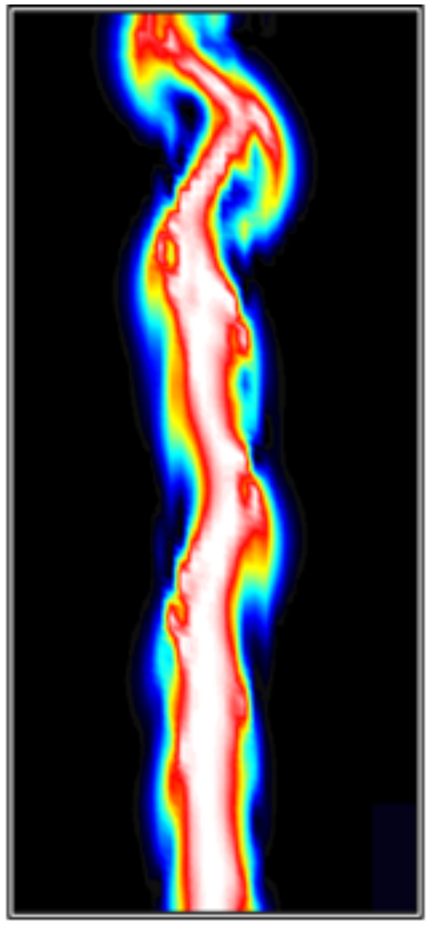
8,040 cores

91% utilization in 2014

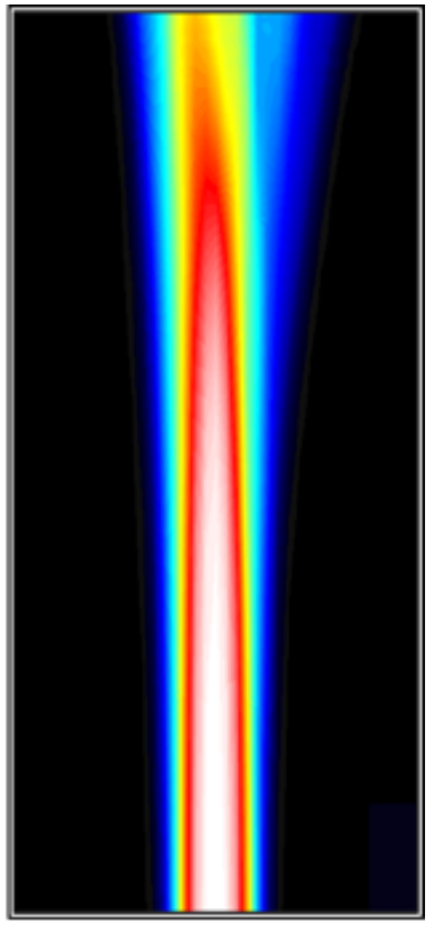
IPLMCFD


Direct Numerical Simulation

(DNS)


Large Eddy Simulation

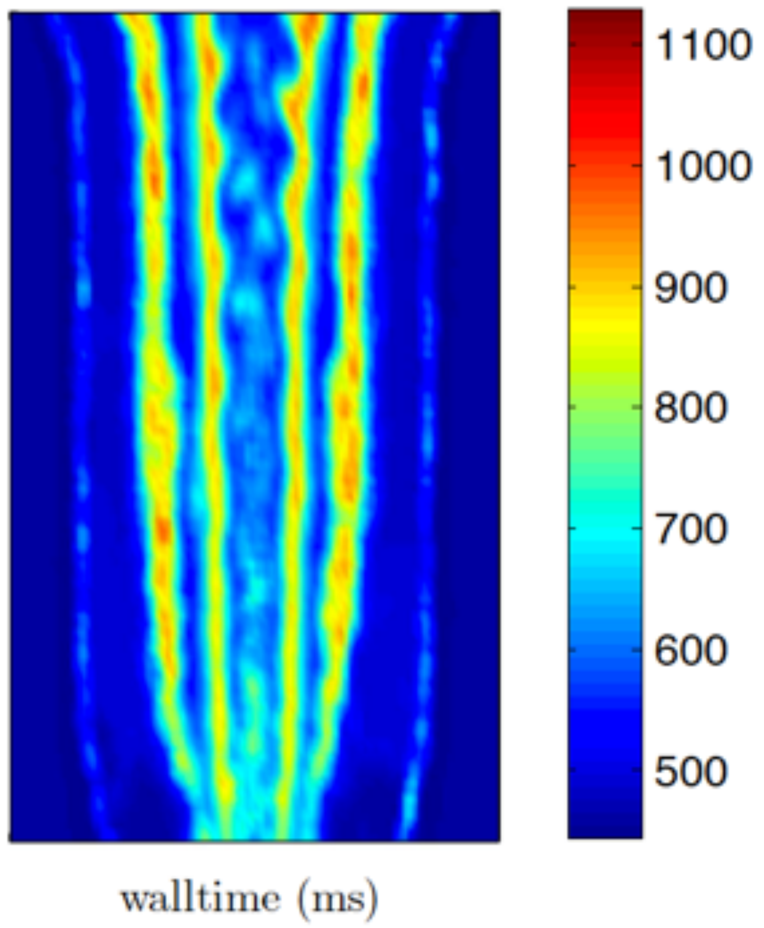
(LES)

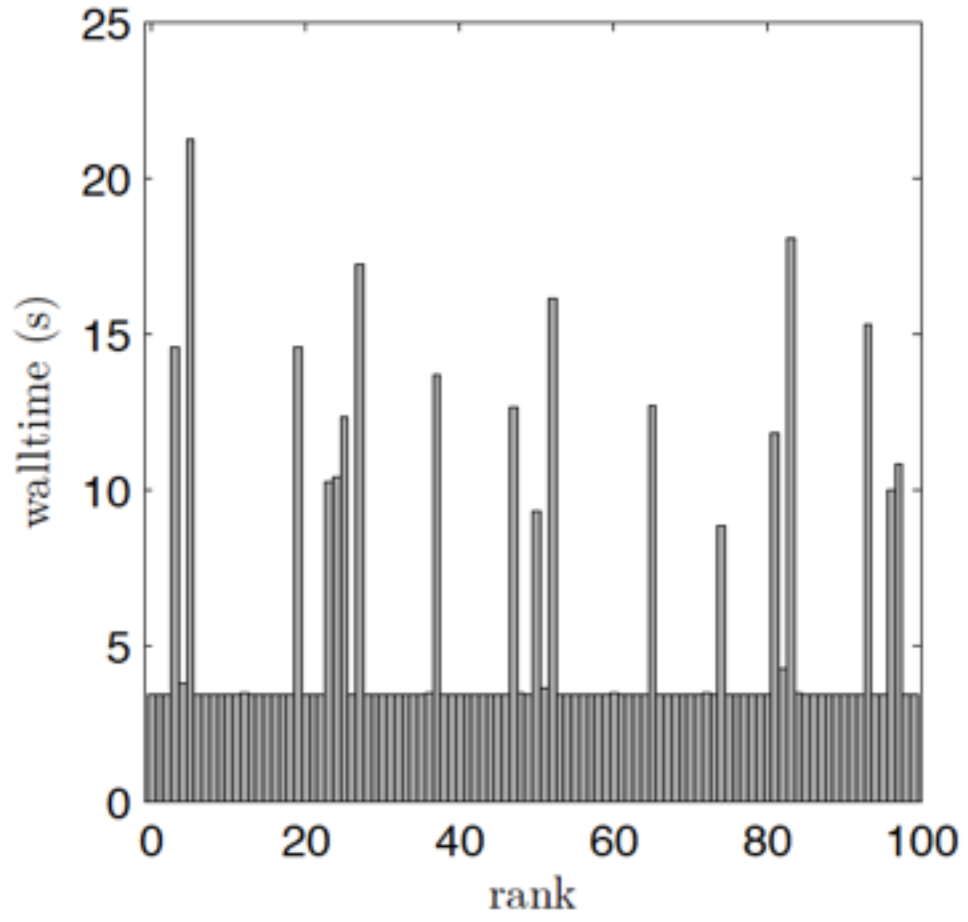

Reynolds Averaged Simulation

(RANS)

- A massively parallel solver for turbulent reactive flows.
- LES via filtered density function (FDF).

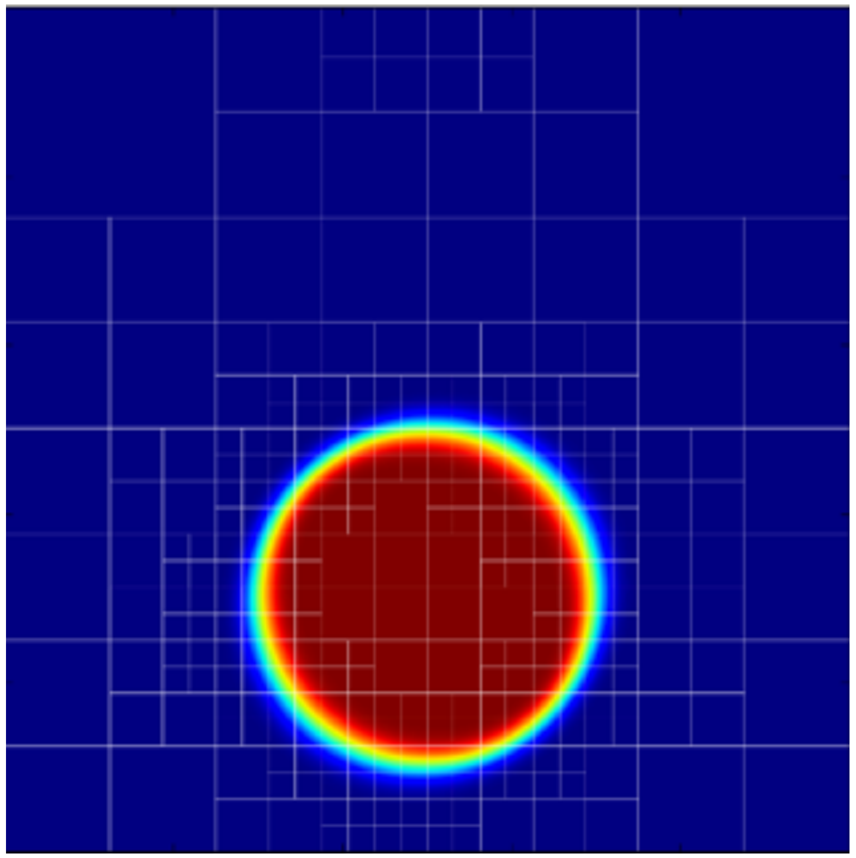
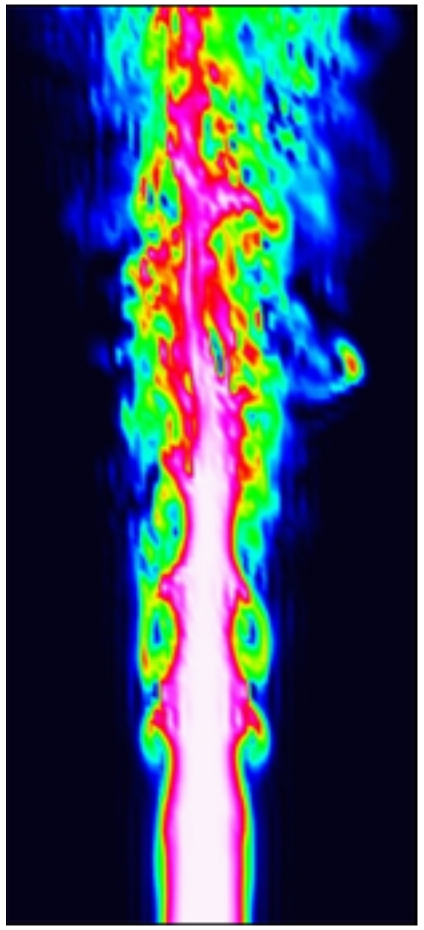
Load Imbalance





- IPLMCFD uses a graph partitioning library (METIS) to redistribute work.
- Requires to split execution between calls to repartition cells.

Reasons for Load Imbalance in CFD

Traditional	IPLMCFD
 Langer *et al*, SBAC-PAD, 2012.	
Adaptive Mesh Refinement	Chemical Reaction

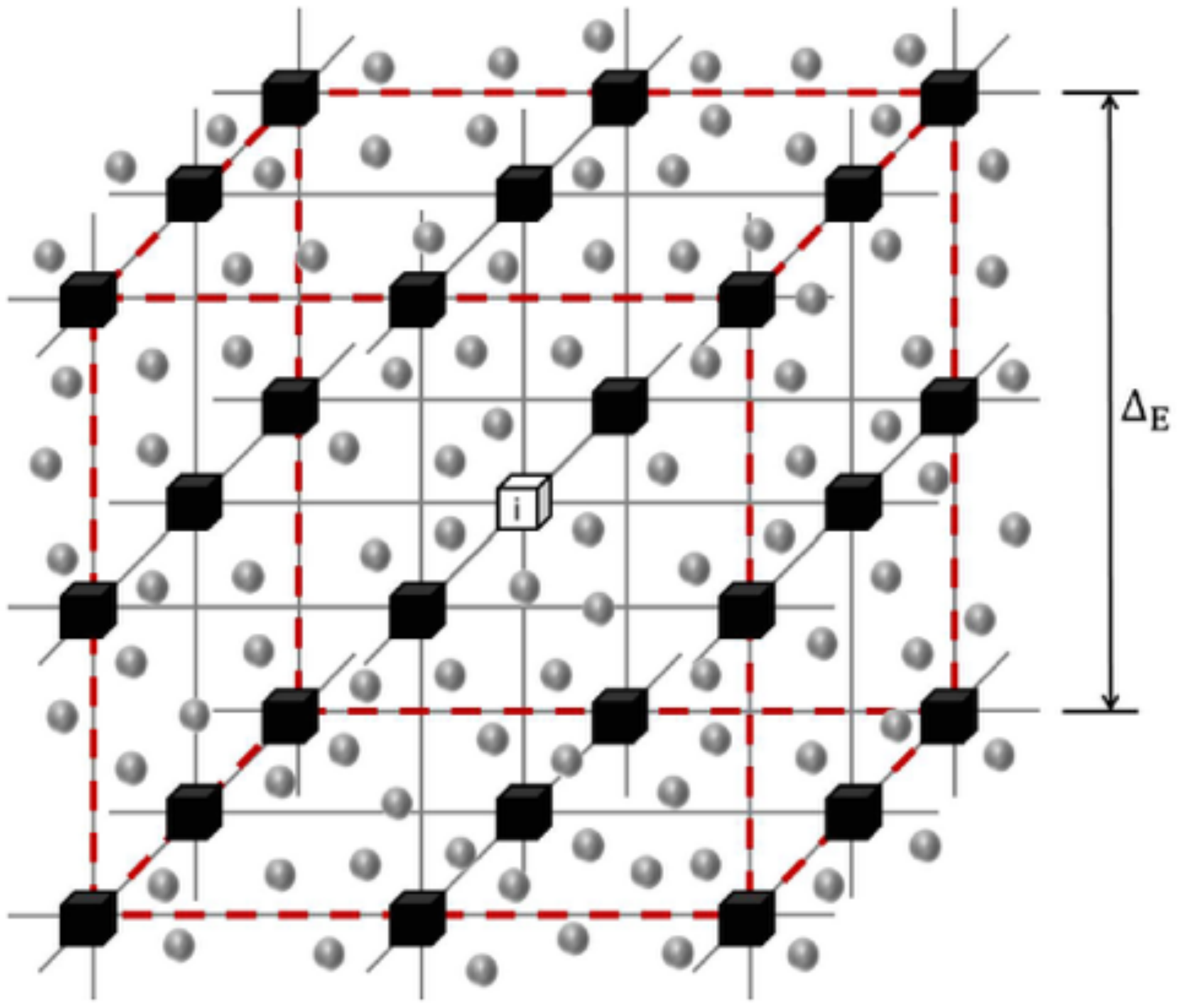
- Approaches:
  - ❖ Task-parallel
  - ❖ Zoltan
  - ❖ Charm++

Agenda

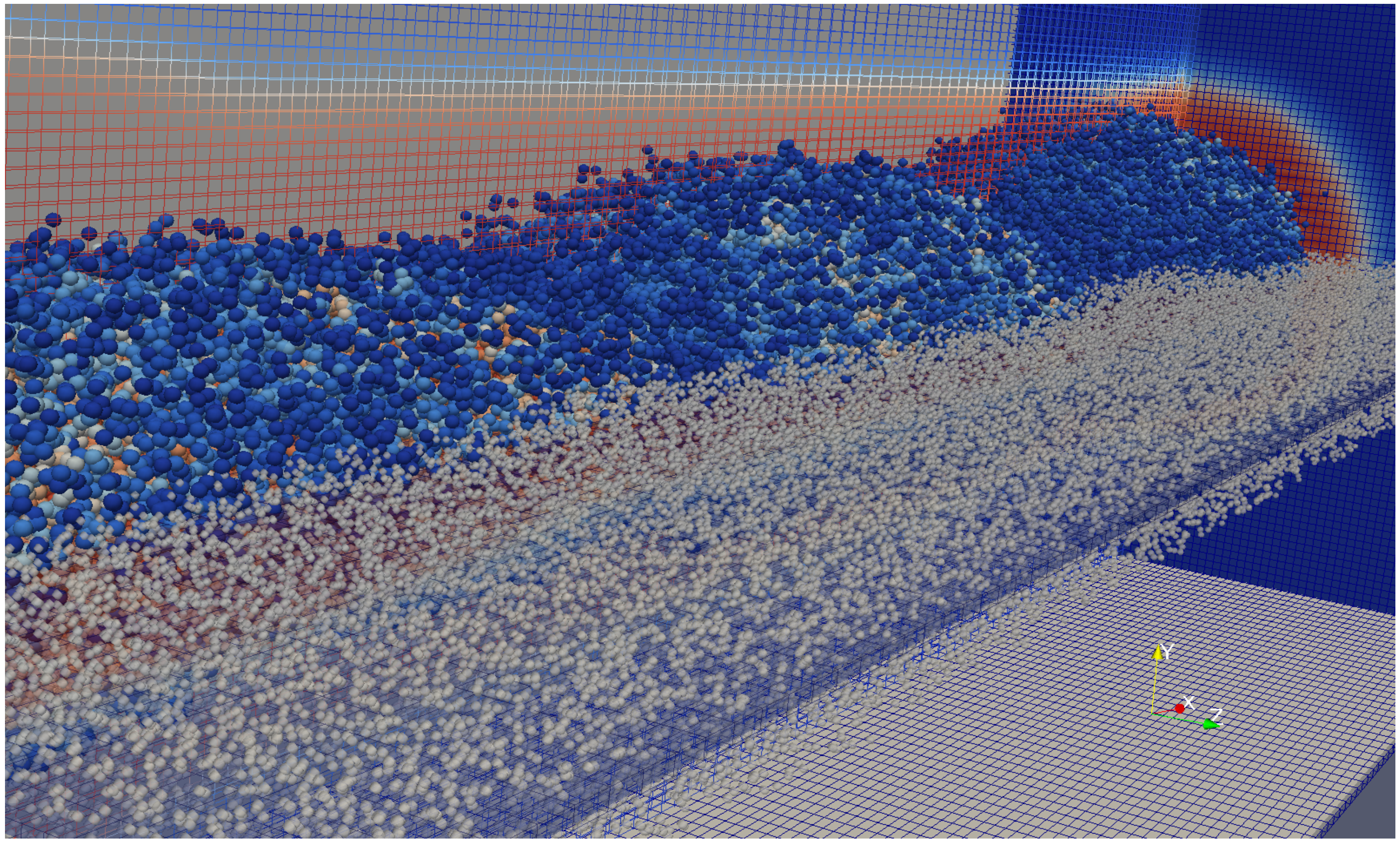
- IPLMCFD: A Hybrid Computational Fluid Dynamics Application
- Zoltan Library
- PaSR Benchmark
- Zoltan vs Charm++ Comparison

Hybrid CFD Application

- IPLMCFD: Irregularly Portioned Lagrangian Monte Carlo Finite Difference.
- Domain divided into cells, the atomic distribution unit.
- Ensemble of cells:
  - Same number of FD points.
  - Same number of MC particles.

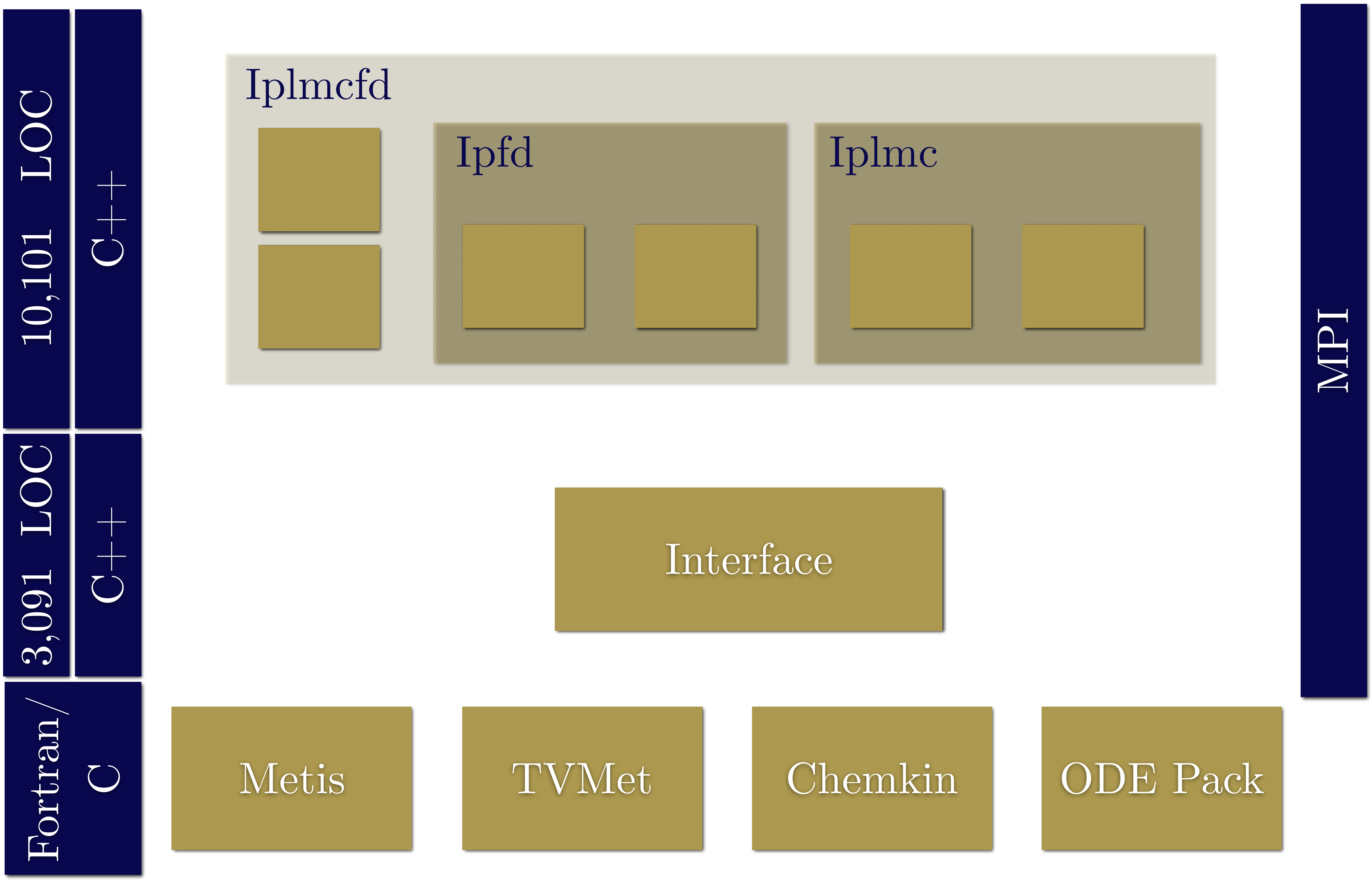


Computational Fluid Dynamics



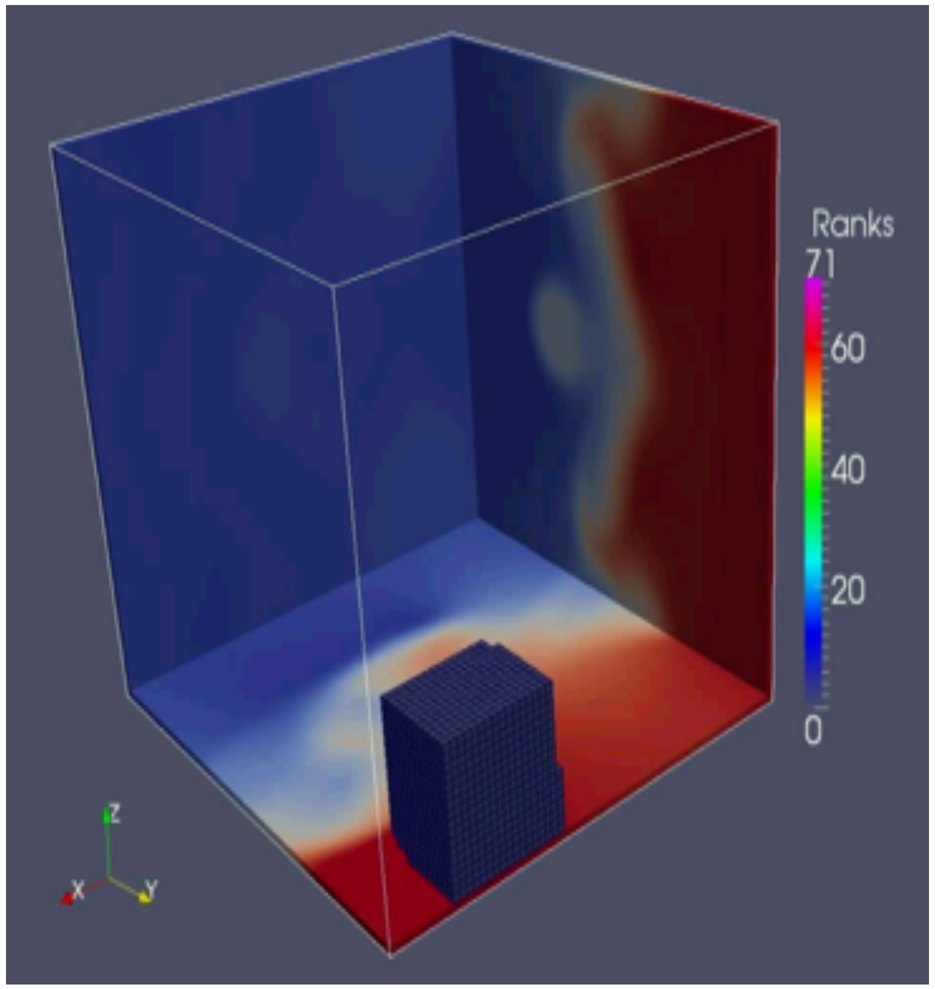
# Grids	# Particles	# Species	Required Memory GBs	GFLOP per iteration	# Iterations	Serial Run-time (1 GFLOP/s)
10^6	6×10^6	9	1.69	29.5	60,000	20.5 days
10^6	6×10^6	19	2.48	90.7	60,000	63 days
5×10^6	50×10^6	19	24.0	544.7	220,000	3.8 years

Code Structure



IPLMCFD

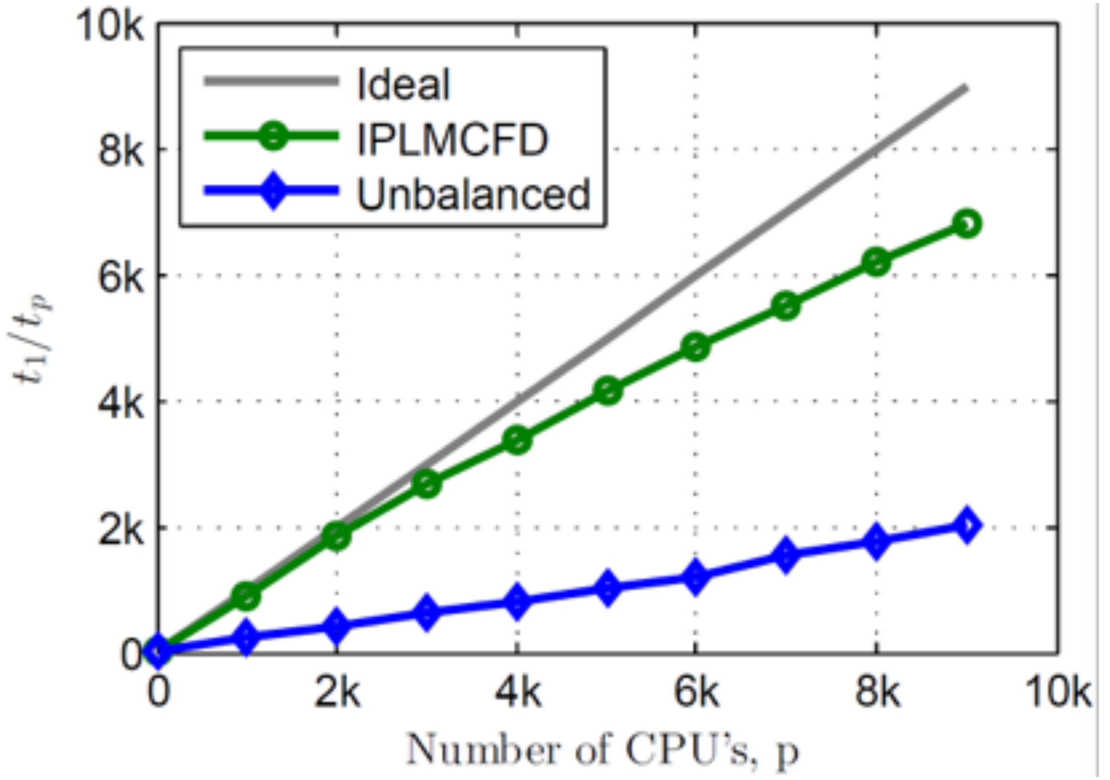
- A scalable algorithm for hybrid Eulerian/Lagrangian solvers.
- Goals:
  - Balance the computational load among processors through weighted graph partitioning.
  - To minimize the number of adjacent elements assigned to different processors (minimize the edge-cut).
- Irregularly shaped decompositions:
  - Disadvantages:
    - Nontrivial communication patterns
    - Increased communication cost.
  - Advantage (major):
    - Evenly distributed load among partitions.

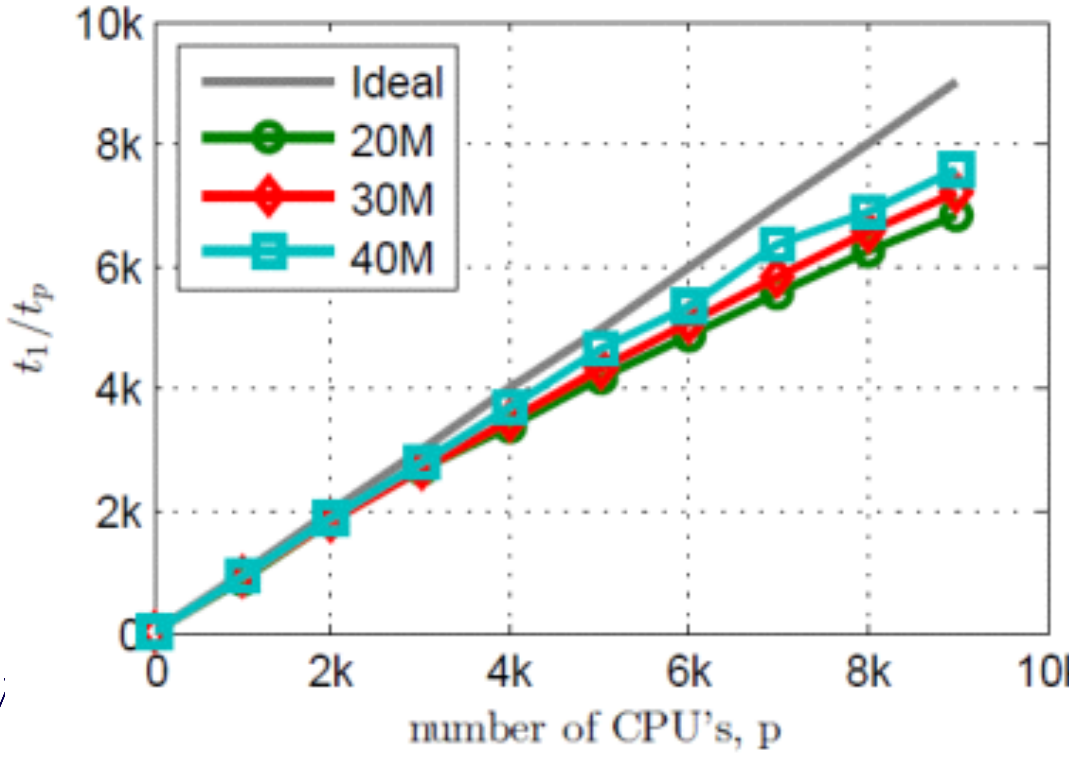


P. H. Pisciuneri *et al.*, *SIAM J. Sci. Comput.*, vol. 35, no. 4, pp. C438-C452 (2013).

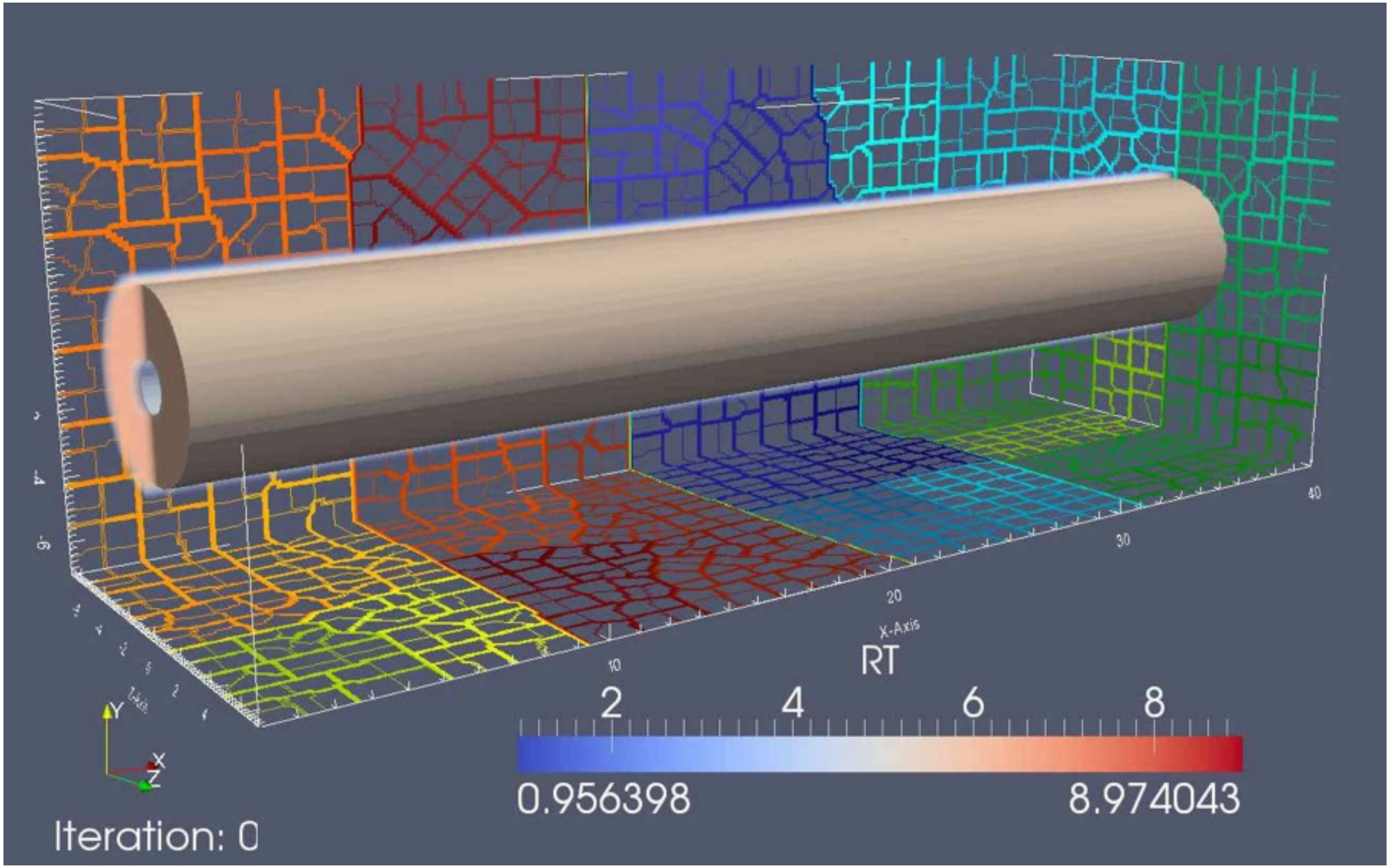
Strong Scaling

- Geometry:
  - 2.5 million FD points
  - 20 million MC particles
  - Chemistry: 9 species, 5-step
- Top:
  - Unbalanced: 22% efficiency (9K cores)
  - IPLMCFD: 76% efficiency (9K cores)
- Bottom:
  - Performance of IPLMCFD improves as the number of MC particles increases
  - IPLMCFD: 84% efficiency at 9k processors for 40M particles
- Timing:
  - The average of 10 iterations immediately after load balancing



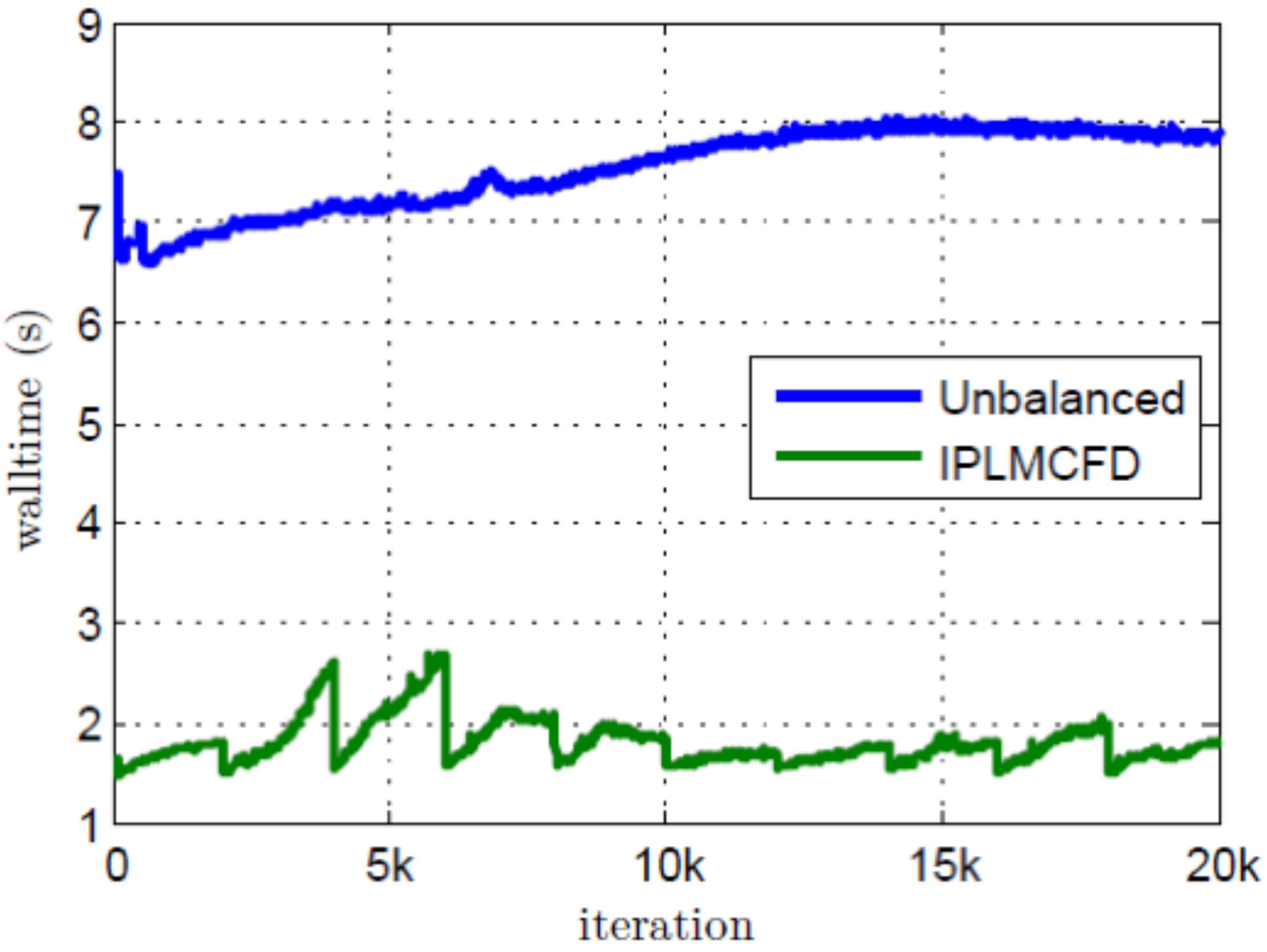


Simulation of a Premixed Flame



Temporal Performance of IPLMCFD

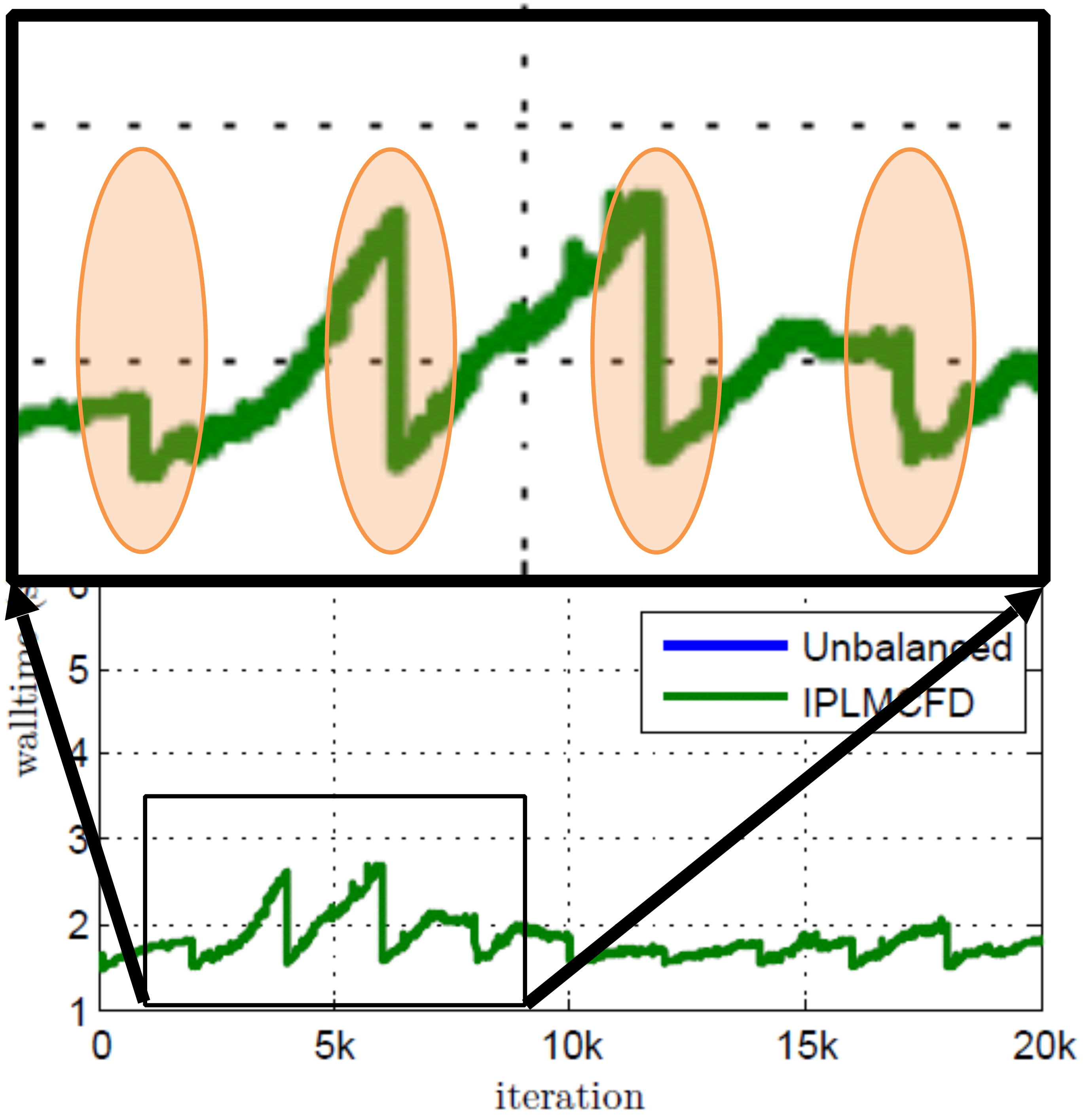
- Unbalanced: approx. static performance
- IPLMCFD: variable performance
  - Load balancing is performed approx. every 2000 iterations
  - Optimal performance immediately after load balancing
  - Performance degrades in time
- Potential walltime savings afforded by IPLMCFD for this example:



$\boxed{T_{\text{Unbalanced}} - T_{\text{IPLMCFD}} = 30 \text{ hours}}$

Cost of Repartitioning

- **Naïve approach:**
  - Immediately before load-balancing checkpoint the entire simulation
  - Restart the simulation with a new decomposition
  - Costly, involves:
    - Writing to shared filesystem
    - Simulation cleanup
    - Simulation startup
    - Reading from shared filesystem
  - Does not scale
  - $O(10^2 - 10^3)$ iterations in cost
- **Optimal approach:**
  - Repartitioning should be handled in memory
  - The new partition is aware of the previous partition, thus minimal data movement and interruption



Zoltan

- "*A toolkit of parallel combinatorial algorithms for unstructured and/or adaptive computations*".
- Sandia-OSU collaboration since 2000.
- Part of Trilinos package.
- Zoltan2 project in C++.

Dynamic load balancing
Parallel repartitioning

Data migration tools

Distributed data directories

Unstructured communication

Dynamic memory management

Zoltan IPLMCFD

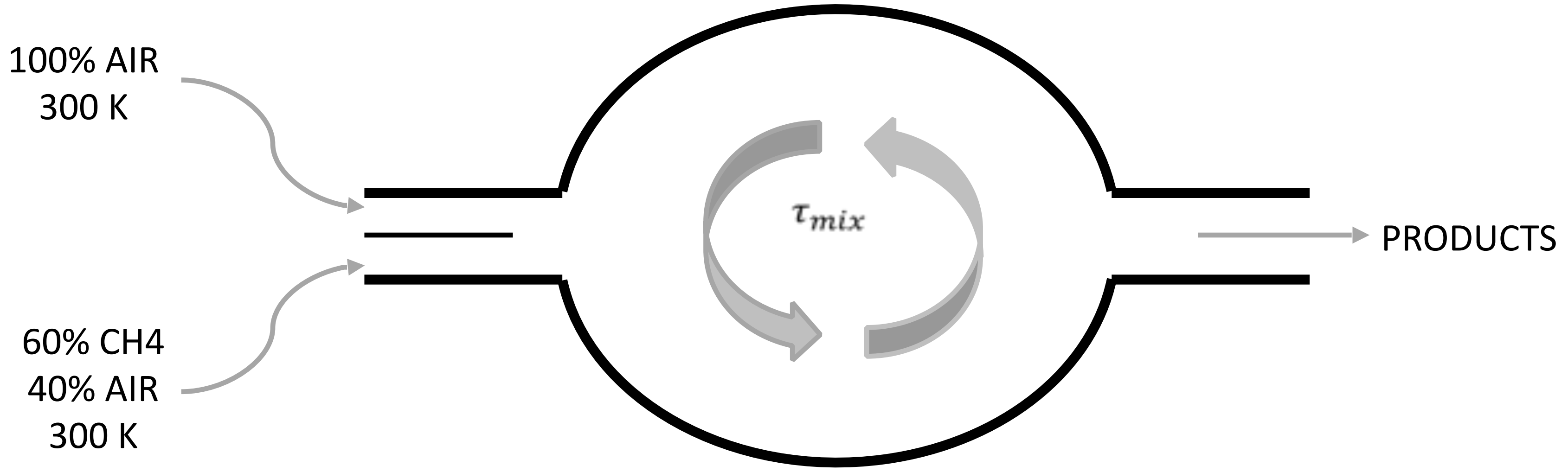
- Zoltan's callback function interface.
- Methodology:
  - ❖ Atomic unit $\longrightarrow$ cell (irregular subdomains).
  - ❖ Data registration $\longrightarrow$ number of objects, object weights.
  - ❖ Graph management $\longrightarrow$ number of edges, edge weights.
  - ❖ Migration $\longrightarrow$ pack/unpack functions.
  - ❖ Load balancing $\longrightarrow$ partition, repartition, refinement.
  - ❖ Global information $\longrightarrow$ distributed data directory.

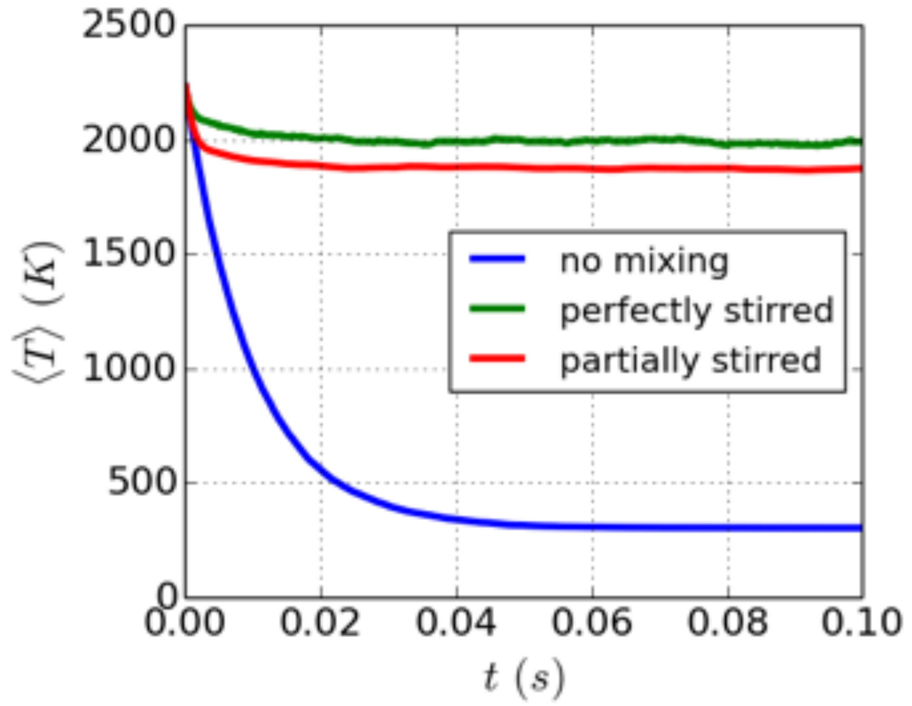
Charm++ IPLMCFD

- Goal: fully exploit Charm++ features.
- Methodology:
  - ❖ Atomic unit $\longrightarrow$ subdomain (regular subdomains).
  - ❖ Containing class $\longrightarrow$ 3D chare *array*.
  - ❖ Process-based data $\longrightarrow$ chare *group*.
  - ❖ Communication $\longrightarrow$ outermost level.
  - ❖ Structured control flow $\longrightarrow$ Structured Dagger.
  - ❖ Migration $\longrightarrow$ PUP methods.

Partially Stirred Reactor (PaSR)

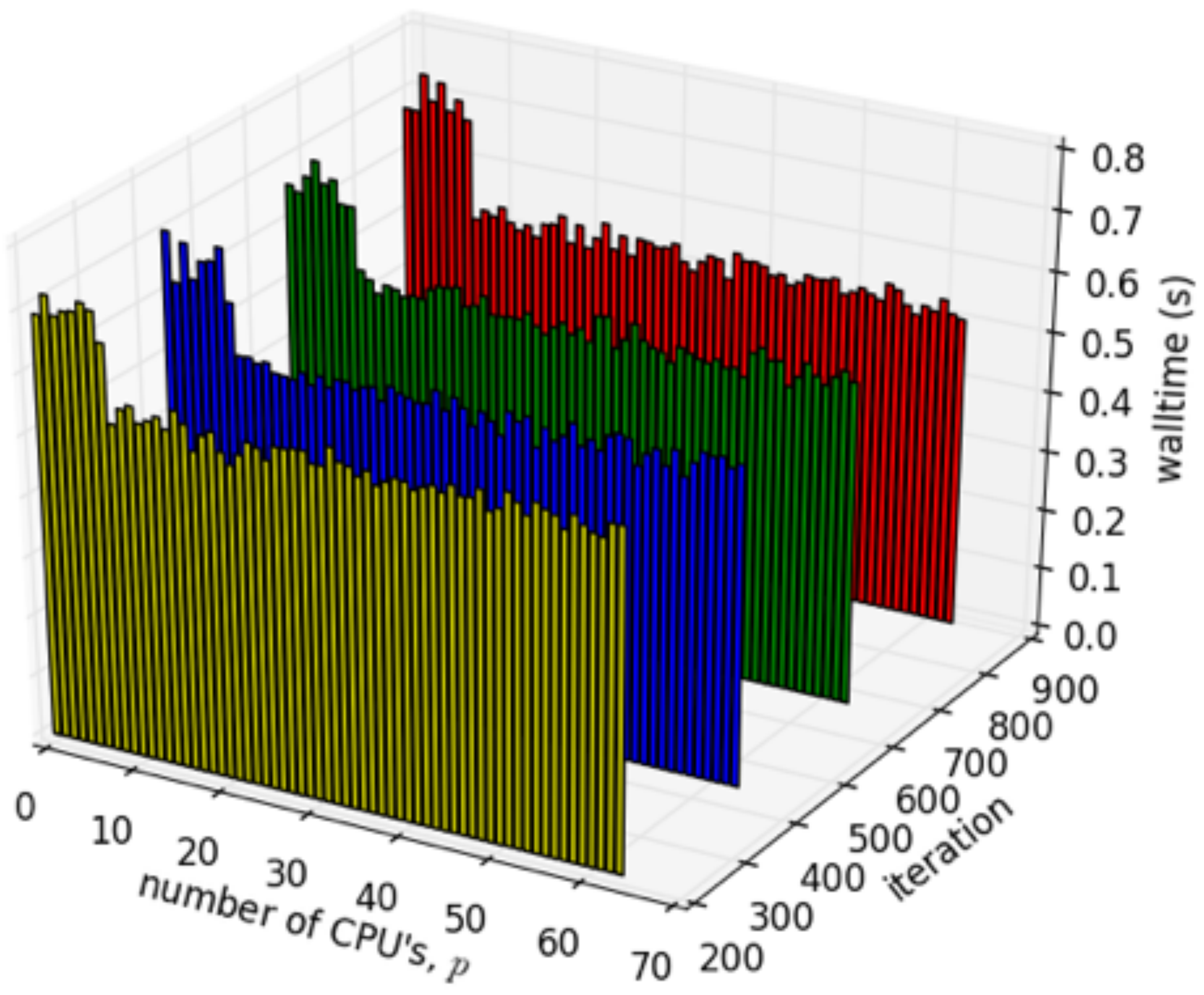
- Parameters:
  - IC: Stoichiometric mixture of methane&air reacted until equilibrium (T≈2230 K)
  - Simulation duration: $t_{end}=10\ \tau_{res}$
- Realizability:
  - Lower bound, no mixing
  - Upper bound, perfectly stirred



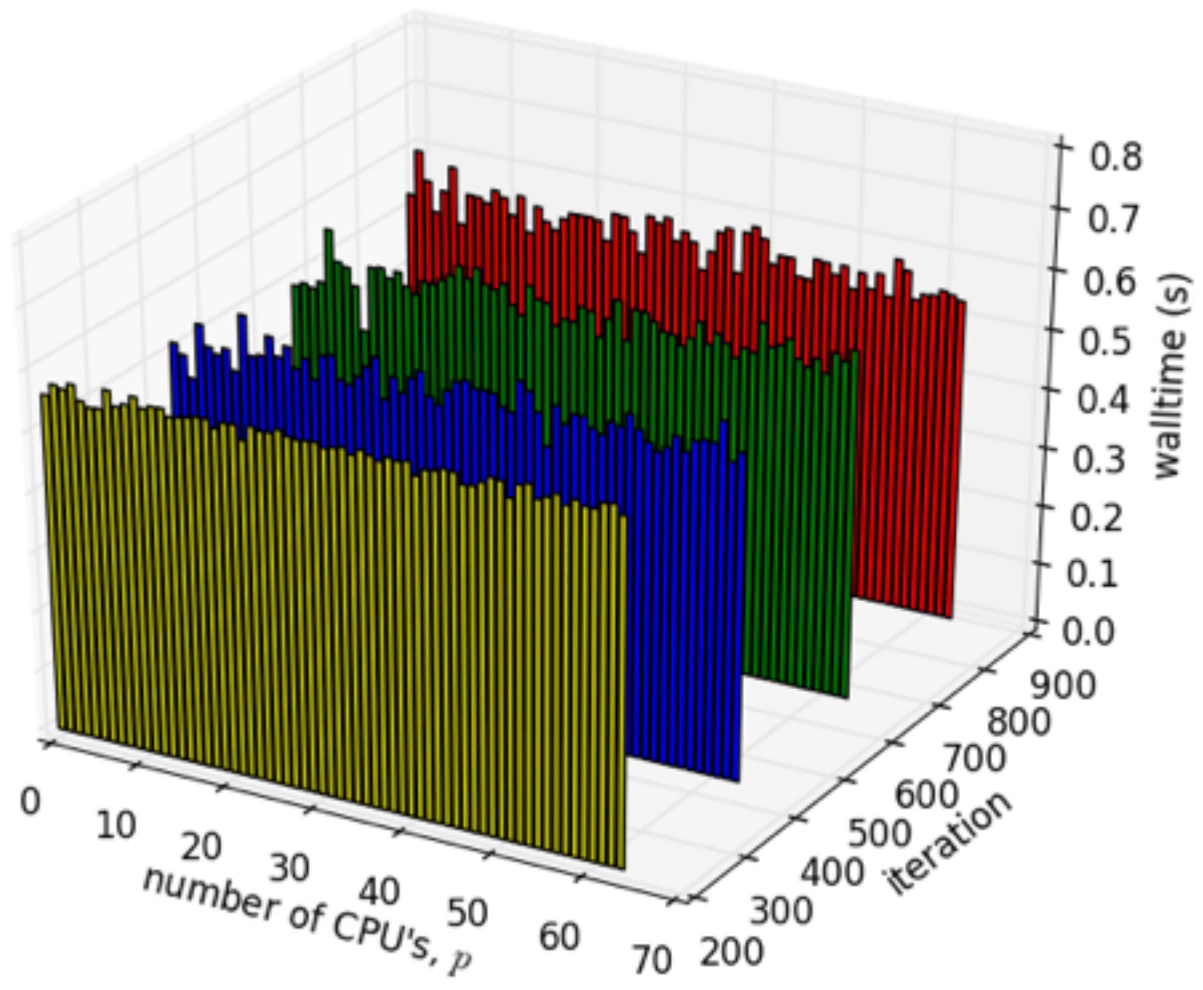


Dynamic Load-Balancing

Static Partition



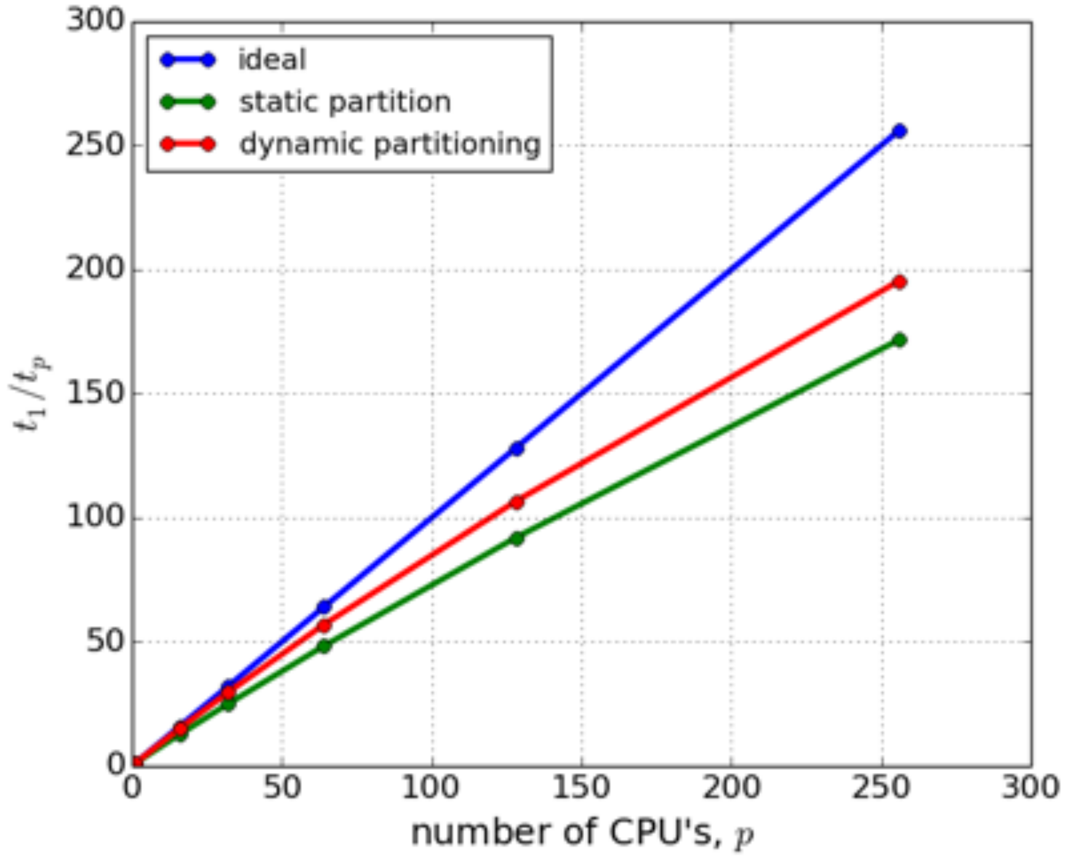
Dynamic Partitioning



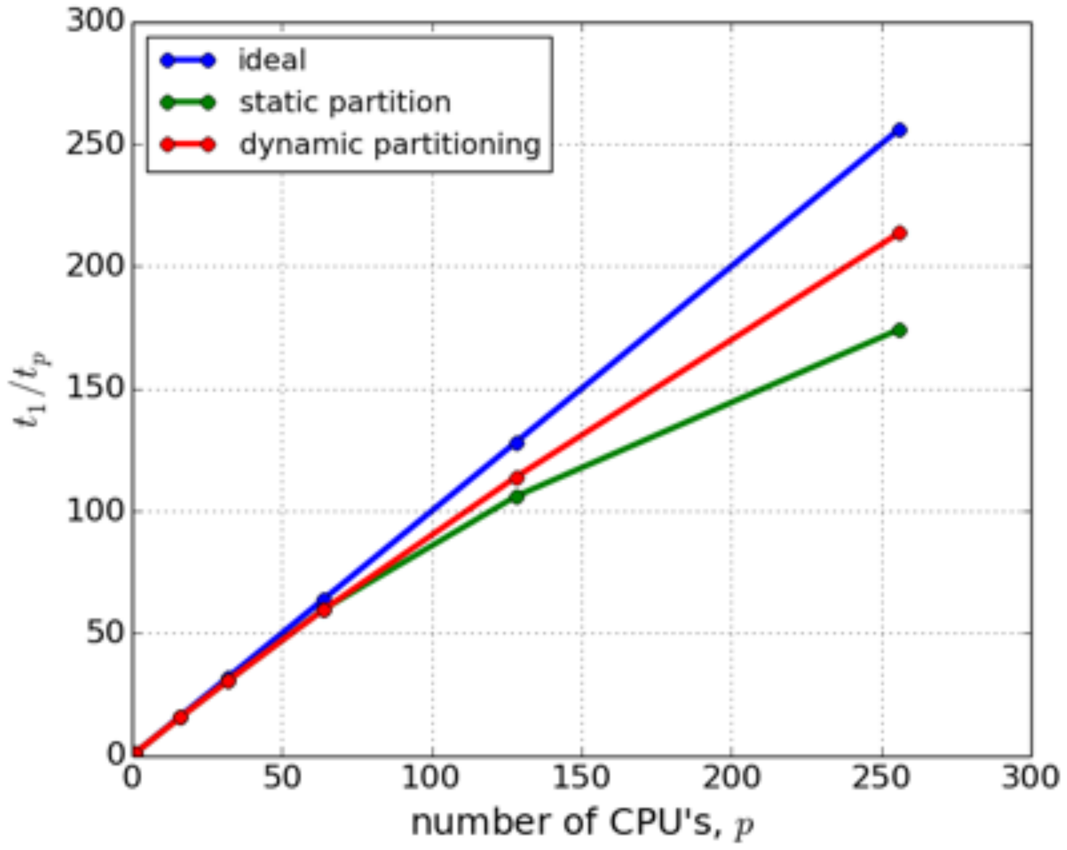
Strong Scaling

- Parameters:
  - ❖ 10,000 particles
  - ❖ Chemistry: 9 species, 5-step
- Timings over the entire simulation (Stampede)
  - ❖ The Zoltan and Charm++ timings include all overhead associated with repartitioning and data migration

ZOLTAN



Charm++



Programming Effort

	Zoltan IPLMCFD	Charm++ IPLMCFD
Startup	39	0
Object Graph Management	80	0
Data Migration	427	61
Load Balancing	40	3
Measured in lines of code (LOC)		

Charm++ Wishlist

- MPI $\longrightarrow$ Charm++ migration guide:
  - Instructions on using Charm++ with build systems.
  - Translating common MPI programming patterns.
  - Dealing with communication operations.
  - Highlighting opportunities for improvement.
- Parallel I/O documentation.
- Accelerator programming documentation.

Conclusions

- Competitive performance between Zoltan and Charm++ for adaptive simulations of turbulent reactive flows.
- Charm++ alleviates programming effort of infrastructure for adaptive computation.

Thank You!

Q&A