Parallel Computation of Axisymmetric Jets

SciPy.in 2010

Nek Sharan

Department of Aerospace Engineering
Indian Institute of Technology Bombay

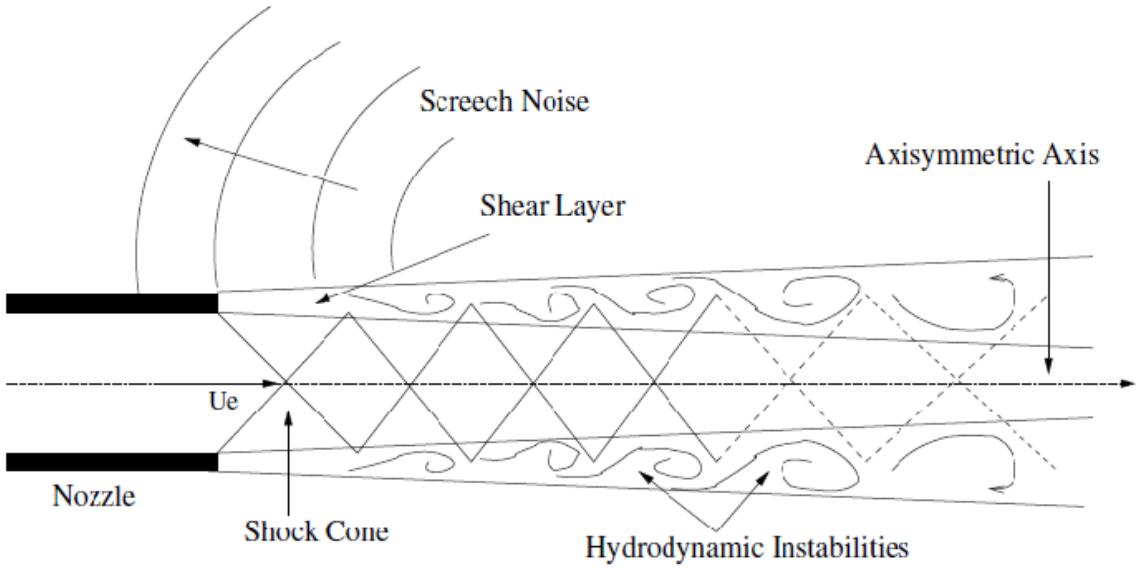
December 13, 2010

Outline

Outline

Flow Physics

- Imperfectly expanded jet
- Resonant screech loop
  - Instability wave growth
  - Instability-shock interaction
  - Acoustic feedback
  - Receptivity processses

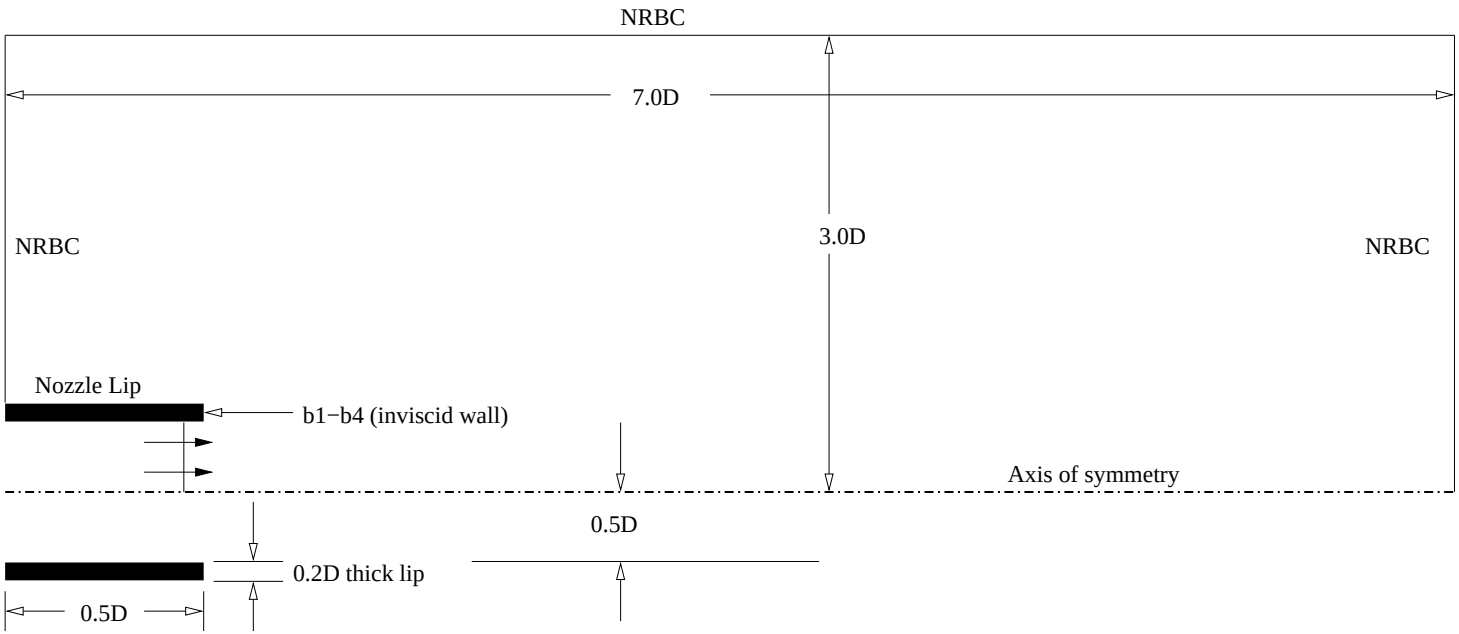


In-house C code

- Favre-filtered Axi-symmetric Navier Stokes Solver
- Spatail discretization - 5th order WENO scheme
- Advancement in time by 2nd order TVD Runge-Kutta time stepping
- Smagorinsky's eddy viscosity model for subgrid scale modeling
- Constants for subgrid models -> based on DNS results of Erlebacher et al
- Accepts input parameters in text file
- Uses procedural programming approach

Flow conditions and Computational domain

Flow Variables	P (N/m^2)	T (K)	ρ (Kg/m^3)	u (m/s)	v (m/s)
Ambient Condition	101320	288.15	1.225	0	0
Nozzle Exit Condition	128119.35	240.12	1.8591	310.61	0



Parallel computing

- Shared memory
  - All processors access all memory as global address space
  - Changes in a memory location effected by one processor are visible to all other processors
  - Advantage: Data sharing between tasks is fast and uniform
  - Disadvantage: Lack of scalability between memory and CPUs
- Distributed memory
  - Processors have their own local memory
  - Memory addresses in one processor do not map to another processor
  - Advantage: Memory is scalable with number of processors
  - Disadvantage: Synchronization between tasks is programmer's responsibility

Outline

Optimization with Cython

Implementation	Average CPU time for a iteration	Speedup
Python	193.34 s	
In-house C code	0.61 s	316.9x

- Compiling in Cython
- Declaration of variables
- Declaration of numpy arrays
- Switching off boundscheck and wraparound
- Multi-threading

Software	Version
Python	2.6
Cython	0.12.1
matplotlib	0.99.1.1

Performance

Implementation	[1]Average CPU time for an iteration	Speedup
Python	193.34 s	
Compilation in Cython	174.97 s	1.11x
Variable type declaration	155.79 s	1.24x
Numpy array declaration	1.19 s	162.5x
boundscheck and wraparound	0.94 s	205.7x
In-house C code	0.61 s	

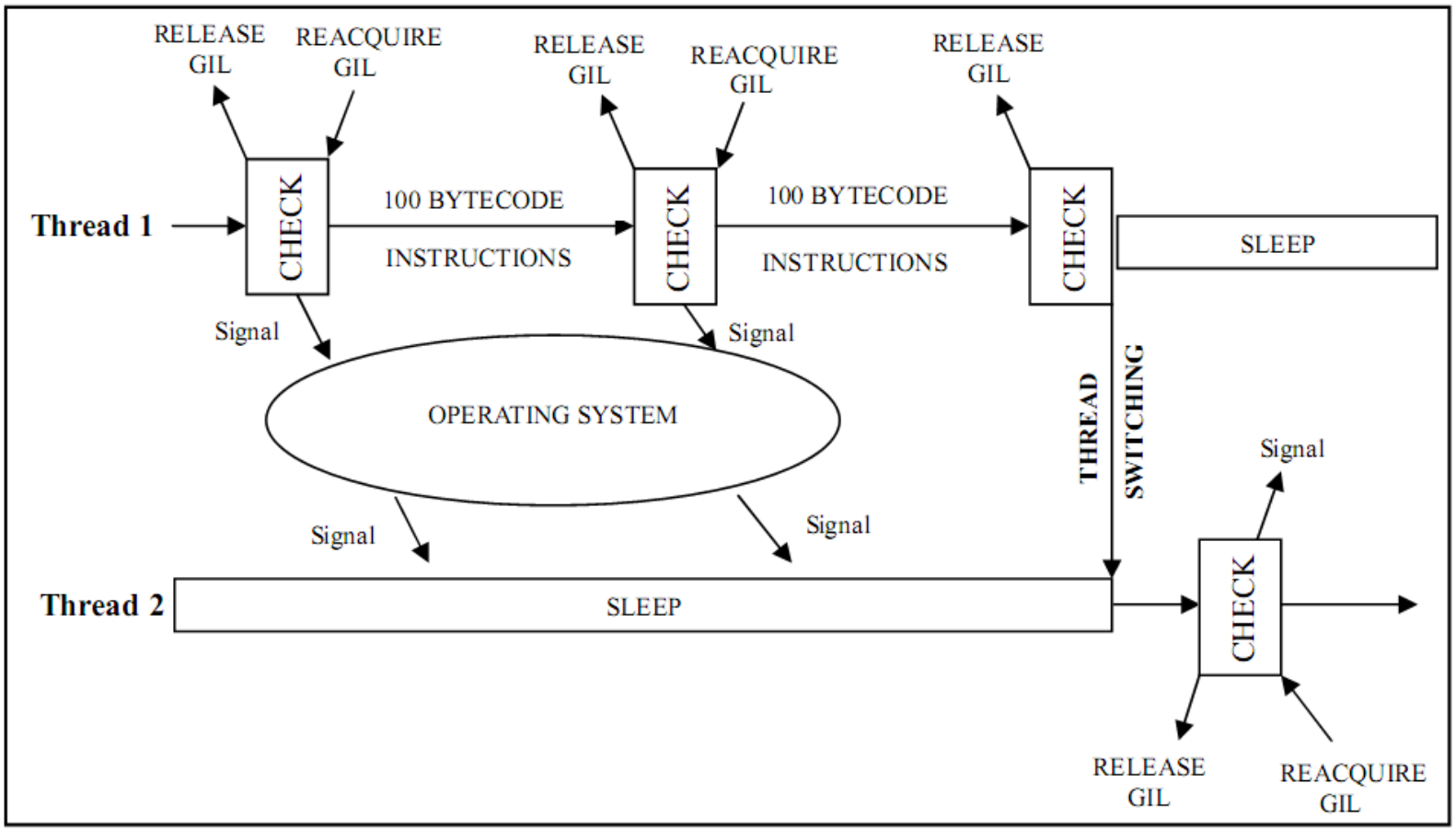
Processor : Intel(R) Core(TM)2 Duo CPU E7500@ 2.93GHz with 3072 KB cache size

[1]Average CPU time computed by averaging the time for 3 iterations on a 350 x 300 uniform mesh

Multi-threading

- No OpenMP support
- Threading module of Python employed
- Global Interpreter Lock (GIL)
  - Controls thread execution
  - Ensures exclusive access to interpreter internals
  - Necessary for a Python operation
- Python does not have a thread scheduler
- *with nogil* statement was used to get rid of GIL

Thread Scheduling



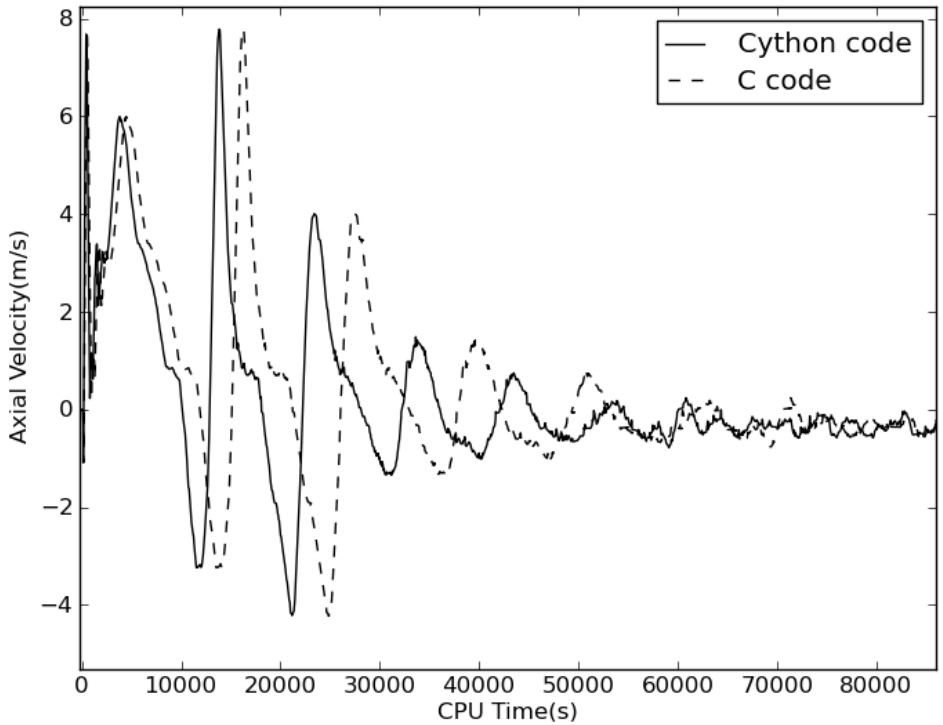
Final Performance

- Spatial derivative for axial and radial direction
  - independent of each other
  - computed on 2 threads

Implementation	Average CPU time for an iteration	Speedup
Python	193.34 s	
Compilation in Cython	174.97 s	1.11x
Variable type declaration	155.79 s	1.24x
Numpy array declaration	1.19 s	162.5x
boundscheck and wraparound	0.94 s	205.7x
Multi-threading	0.52 s	371.8x
In-house C code	0.61 s	

Significance

- LES computations -> 100 thousand iterations or more for developed flow
- Improvement of 0.09 s/iteration saves 2.5 hrs of CPU time

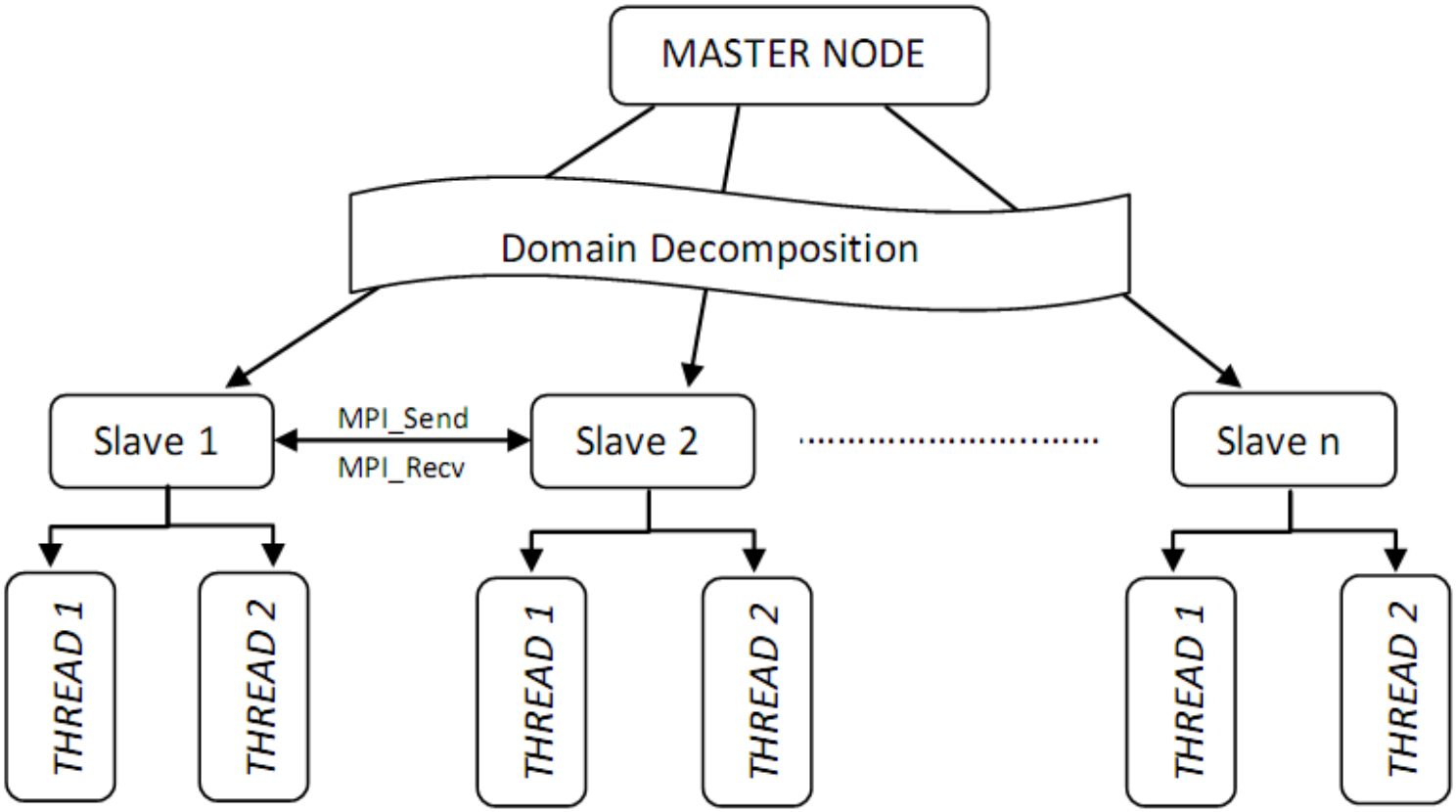


Outline

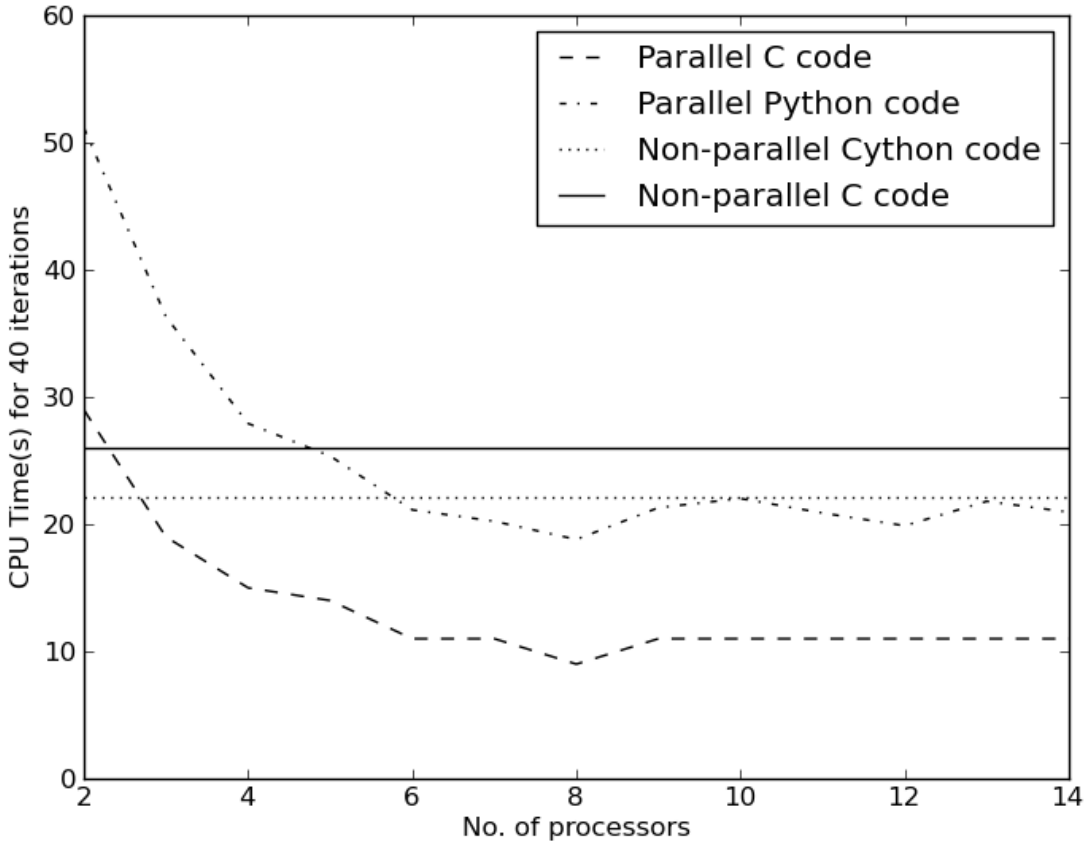
Parallel programming models

- Thread based model
  - Single process creates a number of sub-processes (threads)
  - Threads communicate with each other through global memory
  - OpenMP is used on the subdomain created by MPI based parallelization
- Message Passing model
  - Different sets of task running on different machines
  - Data exchange is through communication
  - Domain decomposition by radially splitting the domain
  - Three grid point interface

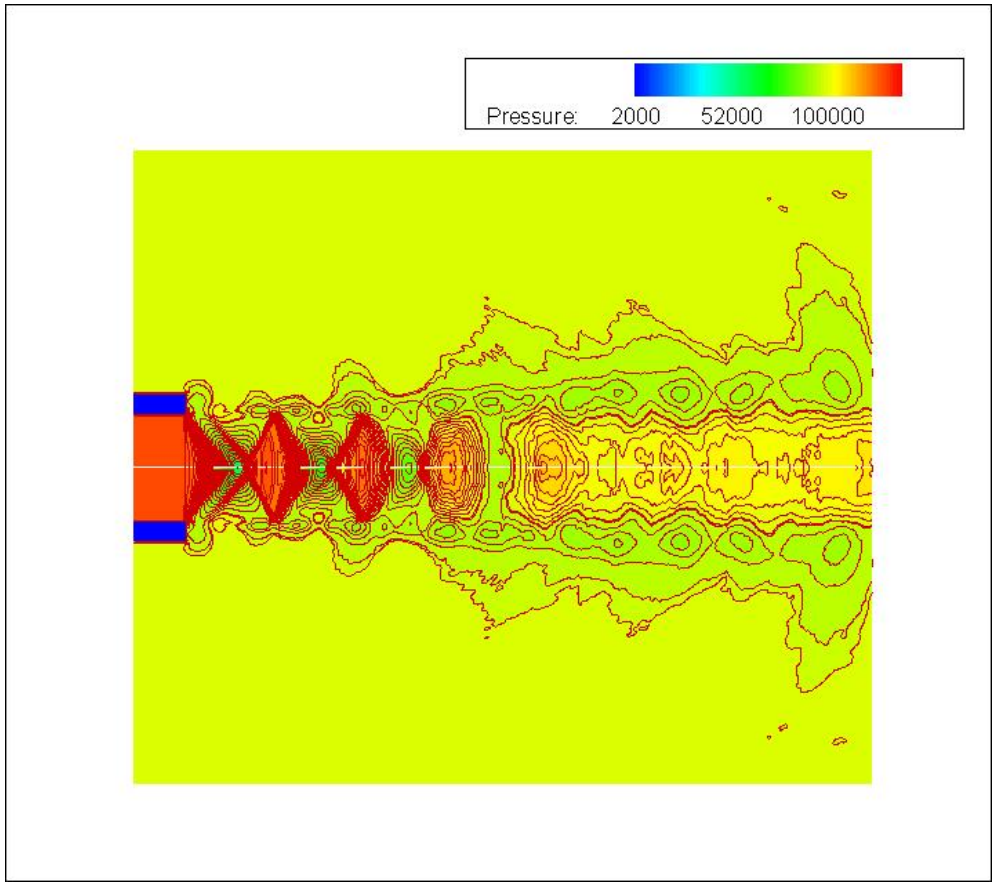
Parallel scheme



Performance results



Pressure contour



Outline

Conclusion

- Python, on optimization with Cython, provides performance equivalent to C code
- Shared and Distributed memory model successfully implemented
- Optimization of inter-processor communication could further enhance performance
- Computational results successfully matched with the experimental results in literature