



# Atmospheric Modeling , Simulation and Visualization Using CUDA

Priyanka Sah (mcs082816@cse.iitd.ac.in)

Subodh Kumar(subodh@cse.iitd.ac.in)

Department of Computer Science and Engineering,  
Indian Institute Of Technology Delhi, India.

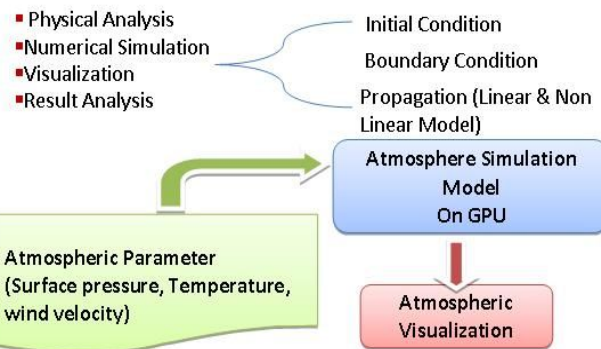
## Problem

The Laboratory Meteorological Dynamics (LMD) by CNRS weather model is used extensively for research and weather forecasting purposes. Simulation of atmospheric climate is one of the most challenging computational tasks because of its numerical complexity and simulation time. The numerical simulations must be obviously achieved faster than in real time to use them in decision support.

## Abstract

- The objective of this project is to find ways to accelerate time critical applications like the real time forecasting and climate prediction .
- The algorithm includes solving a set of equations involving several partial differential equations used for describing the behavior of atmosphere. This makes the algorithm not only memory intensive but also compute-intensive.
- This efficient parallel algorithm solves the Navier Stroke equations that are used to model atmospheric phenomenon and study natural climate variability.
- The entire solution to the second order PDE is computed on architecture supporting massive core parallelism (GPU with CUDA).
- For better performance and speed-up, shared memory model and memory coalescing techniques of CUDA are used.

## Principle of Atmospheric Model



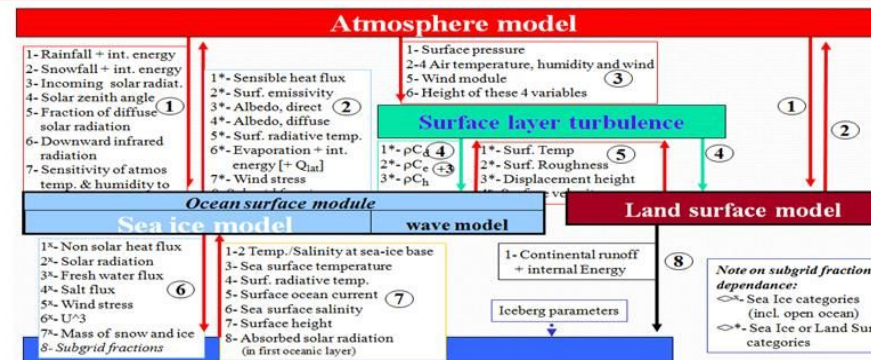
### VISUALIZATION

GrADS Grid Analysis and Display System – Allows interactive real time visualization of atmospheric parameter (pressure/wind) with a domain.

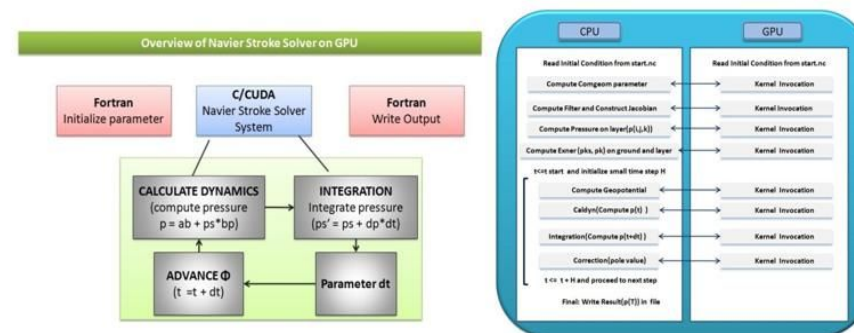
## Experimental Test Case

- Domains Grid Size : 72 latitude 19 longitude 19 vertical level 146 latitude 192 longitude 19 vertical level
- Simulation Time : 1 day,10 days
- Systems used : Nvidia 9800 GTS, Nvidia Telsa C 1060
- Implementation : Compute Unified Device Architecture

## Our Approach: Parallel Implementation

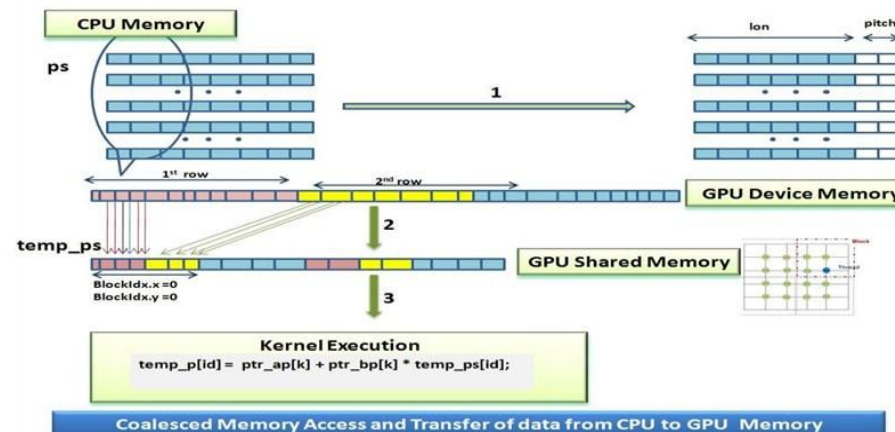


Divide the space (3D domain of size NX x NY x NZ) vertically into NZ horizontal level. Each level is a 2D grid consisting of a finite set of discrete points (NX x NY) at which atmospheric parameters are computed.

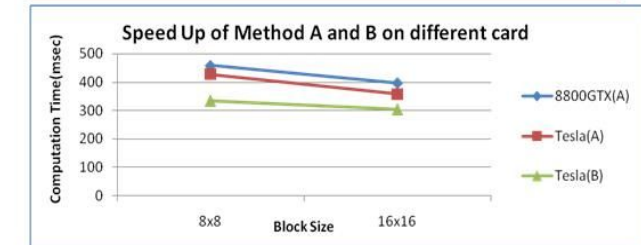


Numerical Solution includes three main parts :-

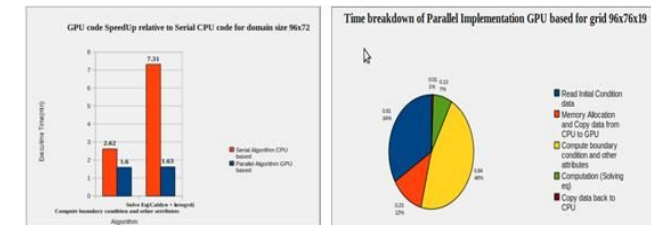
- In every integration step from initial time 't' to total integration time 'T', the derivative 'z' is calculated.
- Values at grid points of each horizontal level are computed by corresponding threads in parallel fashion.
- Any error is corrected and Boundary conditions are handled



## Results & Discussion



- Method A** - All data processed by the kernels are being read from a global memory of GPU card.
- Method B** - Most of the data processed by the kernels are being read from a shared memory. Data is copied once in global memory of GPU card.



## Performance Analysis & Comparison

Grid Size	CPU Implementation (1 day simulation)	GPU Implementation (1 day simulation)
72 x 96 x 19(Overall)	10 min	3.03min
(Solve Navier stroke Equation)Caldyn + Integrd	7.61 min	1.33 min
Other Attributes	2.37 min	1.67min
146 x 192 x 19(Overall)	16 min	4.02min

## Conclusion

### Performance Factors

- Grid Size - Best result on large computational problem
- Selection of Memory Access - Using Shared Memory
- Impact of different thread block configuration - Best result using 16 x 16 block size.

Majority of the total execution time on the GPU is actually spent on performing data transfers, not in the computation itself. For parallel implementation, total time required to transfer input data, compute results, and transfer output back to CPU memory is 10 min. Of that 10min , 24% is spent in the transfers, and only 60% is required for the actual computation. When ignoring the cost of data transfer and looking at compute rates alone, the GPU is able to execute the routine with approximately 1.5 min or over 8 – 15 times faster than the fastest CPU result.

This bandwidth bottleneck suggests that more of the computation possibly including the final visualization (GrADS Software) could be moved to GPU