Parallelized DeepFlow

We changed the project title since our parallelization performs well on both images and videos

Team: Shiyu Huang, Hongxiang Qiu, Zeyu Zhao, Zongren Zou

DeepFlow and Existing Work

- DeepFlow algorithm
  - For a pair of consecutive images, DeepFlow algorithm minimizes following energy function (non-convex and non-linear), by solving Euler-Lagrange equations (variational method), which is a general technique for minimizing energy functions, or objective functions with integral $E(\boldsymbol{w}) = \int_{\Omega} E_D + \alpha E_S + \beta E_M \boldsymbol{dx}$

$$E_D = \delta\Psi\left(\sum_{i=1}^{c} \boldsymbol{w}^\top \bar{\boldsymbol{J}}_0^i \boldsymbol{w}\right) + \gamma\Psi\left(\sum_{i=1}^{c} \boldsymbol{w}^\top \bar{\boldsymbol{J}}_{xy}^i \boldsymbol{w}\right) \qquad E_S = \Psi\left(\|\nabla u\|^2 + \|\nabla v\|^2\right) \qquad E_M = c\phi\Psi(\|\boldsymbol{w} - \boldsymbol{w}'\|^2)$$

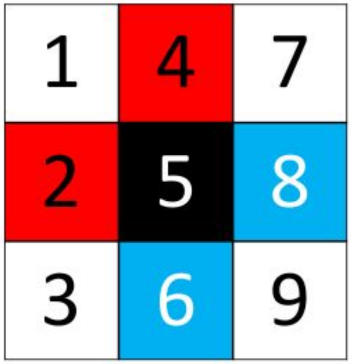
  - Existing work uses SOR method to iteratively solve those equations, with no parallelization (serial). SOR part takes over 78% of total running time.

- Data
  - Data can be any two consecutive images or a video
  - Need for HPC: two-way flow for 20-minute 1080p video takes 29.5 days!

- Problem
  - SOR (existing implementation) has serial dependencies and thus not parallelizable

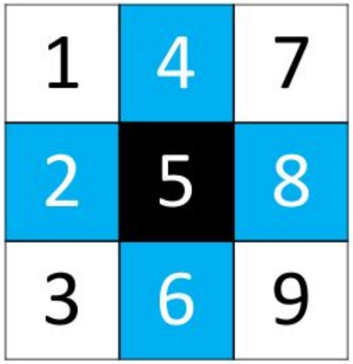
Our Solution

- Algorithm Change
  – Besides SOR, there are other parallelizable iterative methods for solving linear equations. Namely, **Jacobi** and **RedBlack SOR (RBSOR)**



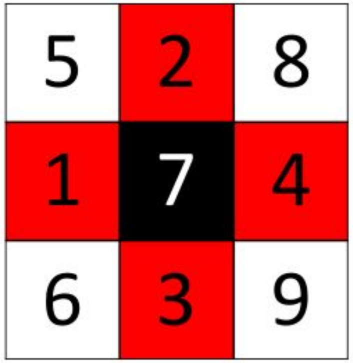
1	4	7
2	5	8
3	6	9

SOR



1	4	7
2	5	8
3	6	9

Jacobi

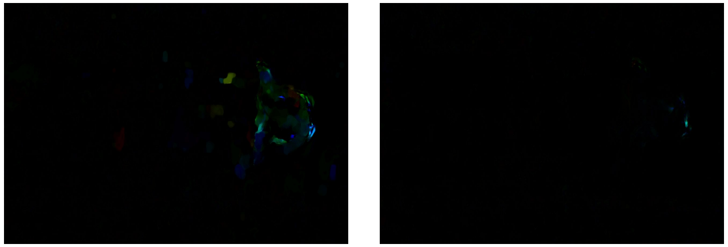


5	2	8
1	7	4
6	3	9

RBSOR (Advanced Feature)

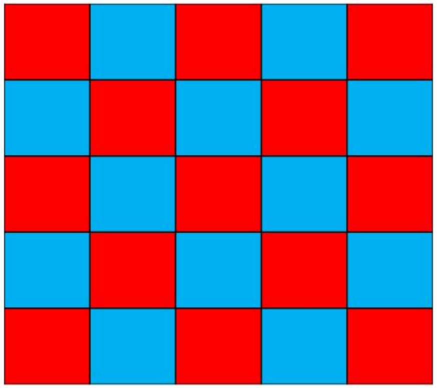
Numbers in cells are serial calculation order. We are calculating black cell, it depends on the last iteration values of blue cells and current iteration values of the red cells.

- Empirical Convergence and Quality
  – SOR is the best (but not parallelizable)
  – RBSOR is better than Jacobi! We'll use RBSOR



Diff. for Jacobi (left) and RBSOR (right)

Parallelization Design

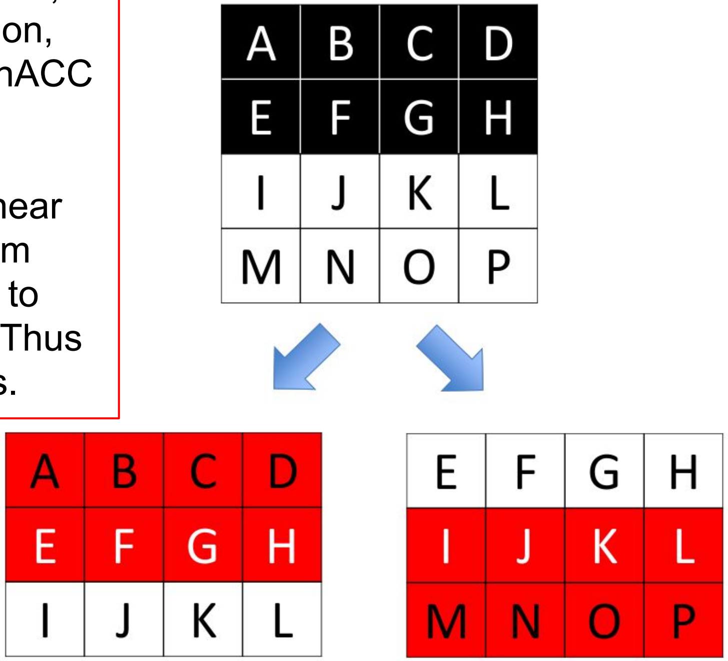


Shared-memory parallelization (OpenMP and OpenACC)

For OpenACC, data is copied to GPU only once.

Since pgcc does not support v4sf, we have to drop this optimization, which adds overheads to OpenACC implementation.

DeepFlow solves about 300 linear systems and each linear system has different size. So we have to reassign MPI jobs every time. Thus our MPI has a lot of overheads.



A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

A	B	C	D
E	F	G	H
I	J	K	L

E	F	G	H
I	J	K	L
M	N	O	P

MPI

The implementation of MPI is very complicated since linear solver is not the only part in DeepFlow algorithm.

MapReduce + OpenMP for Video

Worker nodes get frames and matches from HDFS and compute flows using OpenMP DeepFlow. The calculated flow is uploaded to HDFS.

MPI + OpenMP

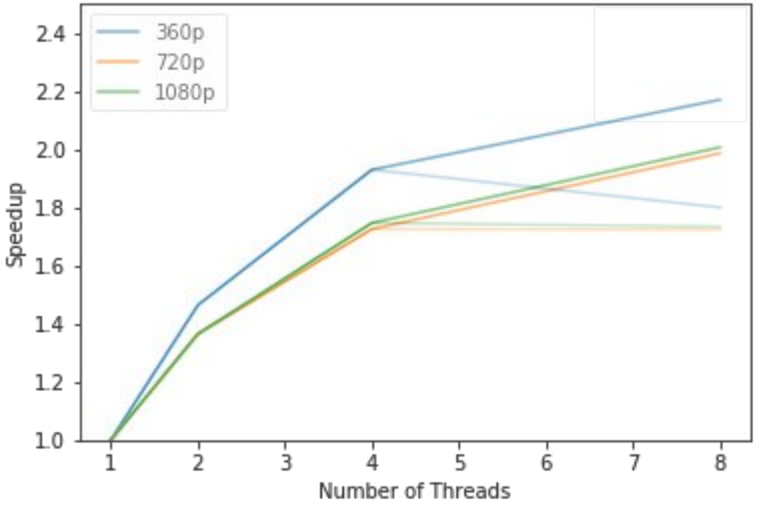
Performance Evaluation

- For 2 Consecutive Images

(Tested on AWS m4.2xlarge instances, all parallelization is based on RBSOR Serial)

	360p	720p	1080p
Original Serial	4.598	19.382	42.582
RBSOR Serial	3.687	15.812	34.476
OpenACC (g3.4xlarge)	3.737	16.499	28.385
OMP (4 threads)	1.910	9.160	19.730
MPI (4 processes)	7.654	30.553	73.249
MPI (3) + OMP (4)	7.133	28.026	69.213

Time (**seconds**) spent for generating one flow for images

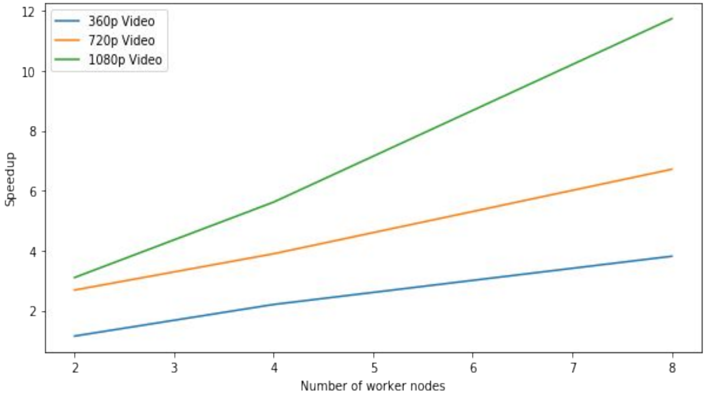


OpenMP Speedup Plot
(For 8 threads, we have results for 4core-8thread CPU and 8core-16thread CPU)

- For 1-second Video (MapReduce)

	SOR	OMP	2 nodes	4 nodes	8 nodes
720p	752	414	280	193	112

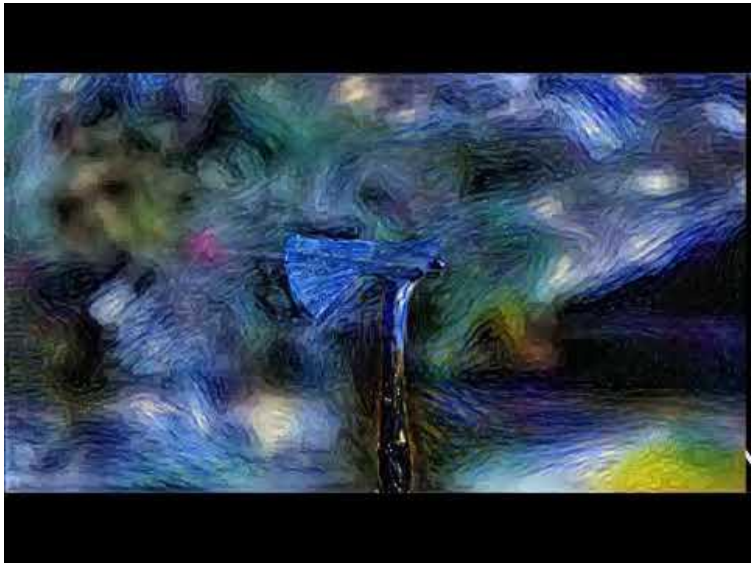
Time (**seconds**) spent for generating two-way flows



MapReduce + OpenMP Speedup Plot

Application

- DeepFlow is useful in
  - Object tracking and activity recognition
  - Motion based segmentation
  - Video processing
    - Slow motion video (Advanced Feature)
    - Stabilize synthesized video



Left: Stylization

Right: Slow Motion

Appendix

- Links
  - Source code and documentation: https://github.com/zeruniverse/CS205-project
  - Website Report: https://zeruniverse.github.io/CS205-project/
- Goals Achieved
  - Changed the linear system solver of DeepFlow to a parallelizable one (RBSOR)
  - Implemented multiple parallelization techniques
    - OpenMP
    - MPI (not working well)
    - OpenACC (not working well)
    - MPI + OpenMP (not working well)
    - MapReduce + OpenMP for video
  - Implemented two applications of DeepFlow (code in the above link)
- Citations
  - See here: https://zeruniverse.github.io/CS205-project/conclusion.html#reference