Parallelize 2D Optical Flow Estimation Algorithm on Video

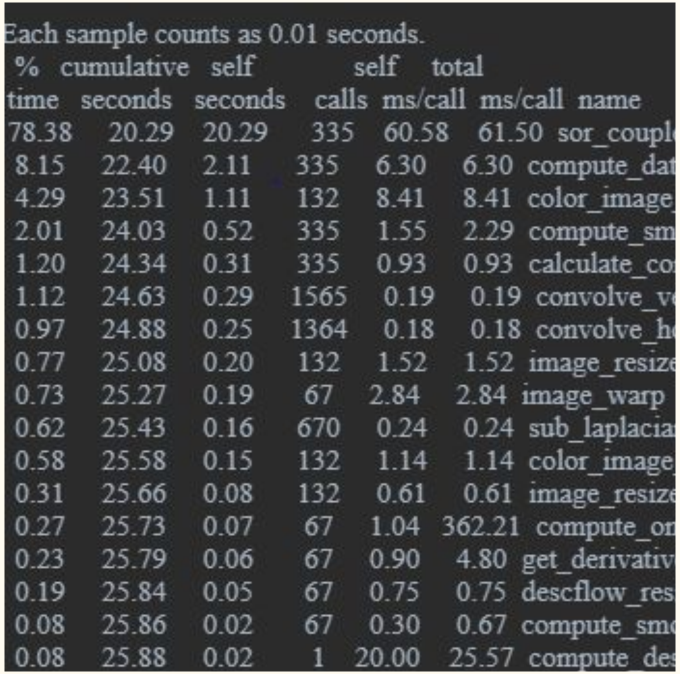
Progress Report

Group 5: Shiyu Huang, Hongxiang Qiu, Zeyu Zhao, Zongren Zou

DeepFlow algorithm

- For a pair of consecutive images, deepflow algorithm minimizes a non-convex and non-linear energy function: downsampling first, fixed point iterations then and **iterative method** to solve linear equations.
- Iterative methods, such as Gauss-Seidel, Jacobi method, have a lot of matrix operations in every iteration, and that's where we can apply parallelism and optimizations.
- Existing codes use SOR to solve linear equations iteratively.

- SOR part takes over 78% of total running time.
- The time complexity of this algorithm is $O(Mn^2)$



```
Each sample counts as 0.01 seconds.
 %   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
78.38     20.29    20.29      335    60.58    61.50  sor_coupl
 8.15     22.40     2.11      335     6.30     6.30  compute_dat
 4.29     23.51     1.11      132     8.41     8.41  color_image
 2.01     24.03     0.52      335     1.55     2.29  compute_sm
 1.20     24.34     0.31      335     0.93     0.93  calculate_co
 1.12     24.63     0.29     1565     0.19     0.19  convolve_ve
 0.97     24.88     0.25     1364     0.18     0.18  convolve_ho
 0.77     25.08     0.20      132     1.52     1.52  image_resize
 0.73     25.27     0.19       67     2.84     2.84  image_warp
 0.62     25.43     0.16      670     0.24     0.24  sub_laplacia
 0.58     25.58     0.15      132     1.14     1.14  color_image
 0.31     25.66     0.08      132     0.61     0.61  image_resize
 0.27     25.73     0.07       67     1.04   362.21  compute_on
 0.23     25.79     0.06       67     0.90     4.80  get_derivativ
 0.19     25.84     0.05       67     0.75     0.75  descflow_res
 0.08     25.86     0.02       67     0.30     0.67  compute_smo
 0.08     25.88     0.02        1    20.00    25.57  compute_des
```

Parallel Computing Structure

❖ **Big Data**: divide video into pairs of 2 consecutive images:
  ➢ MapReduce: Task Level

❖ **Big Compute**: in solving linear systems part, instead of SOR (Successive over-relaxation), we apply other solvers (Jacobi, RedBlack SOR), where data dependency is less:
  ➢ OpenMP+MPI: Loop Level
  ➢ OpenACC: Loop Level

❖ We partition video into multiple pairs of images and run deepflow algorithm on all of them: **Data Parallelism**

❖ We do operations on matrices, and operation on each element is independent within each iteration: **Function Parallelism**

Parallel Model: **Single Program - Multiple Data**

Overheads, expected speed-up, future improvements

- **Overheads**
  - Communications
    - MapReduce: network latency
    - MPI
  - Synchronization
    - OpenACC
    - OpenMP+MPI
- **Speed-up:** for a pair of 720p images, on 2-core CPU,
  - 12s using Jacobi with OpenMP
  - 18s using SOR optimized for serial
  - 24s without any optimization
  - Expected near-linear speed-up
  - Strong scaling (total problem size fixed)

- **Next steps**
  - apply MapReduce to process video
  - Try MPI
  - replace Jacobi with RedBlack SOR to achieve better convergence