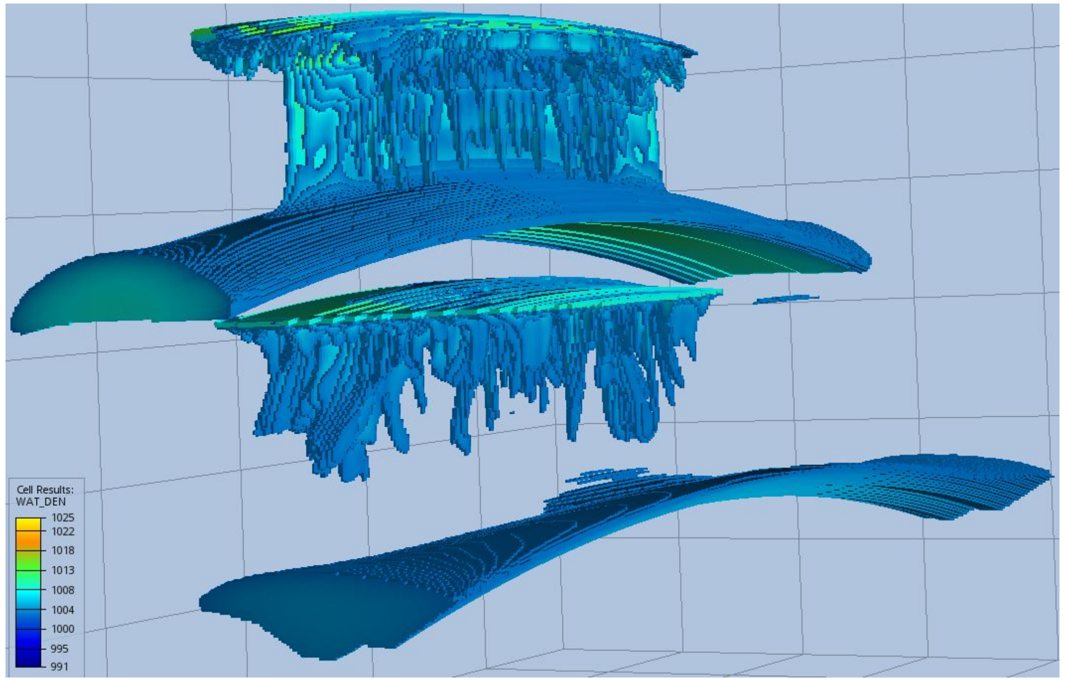
The CuIstl framework and new GPU preconditioners

Tobias Meyer Andersen (SINTEF)



GPU support for the OPM linear solver

Goals:

- Improve runtime of the linear solver
- Create testbed for combining OPM with GPUs

Challenges:

- Code duplication
- Vendor lock-in
- Is the problem parallelizable enough?

DUNE Istl

Distributed and Unified Numerics Environment

- Heavily templated
  - numerical schemes
  - Iterative solver algorithms
- Types implement linear algebra operations
- Types also responsible for parallel execution

CuIstl

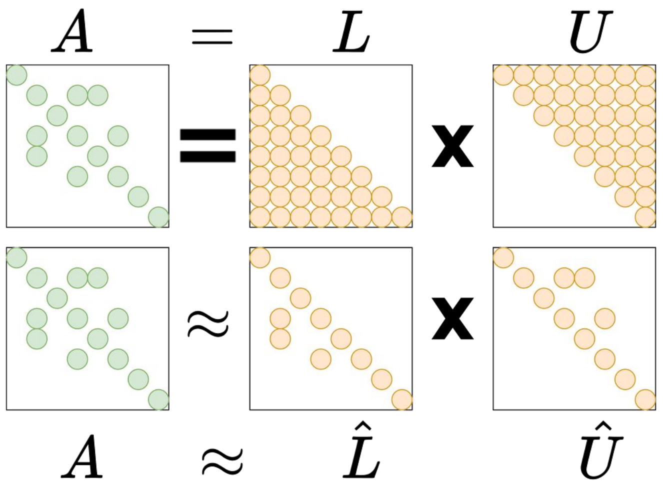
- GPU code in CUDA
- DUNE compatible CUDA classes
    - Support dense vectors and sparse matrices
    - Use CUDA libraries to implement basic linear algebra operations
        - CuBlas
        - CuSparse
    - Dune::BiCGSTAB<CuIstl::CuVector>
    - The linear solver will automatically run this on the GPU

Preconditioners

Good preconditioners are vital for performance

First GPU preconditioner: Incomplete LU factorization

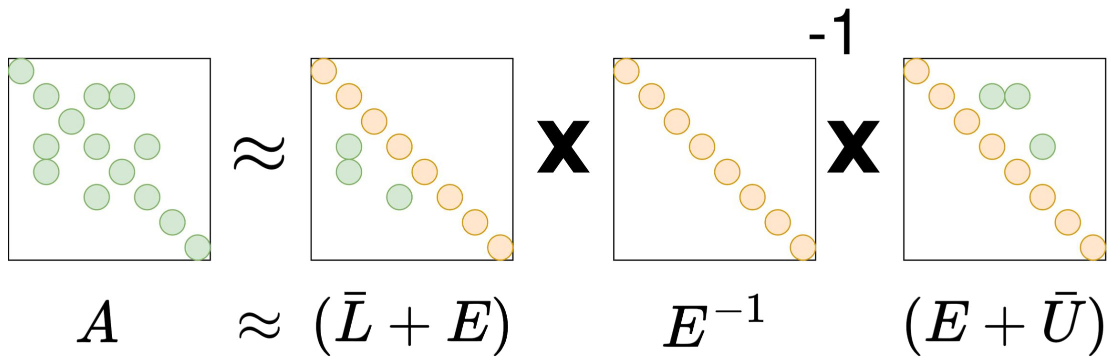
- Implemented using CUDA libraries
  - CuSparse ILU decomposition
    - LU decomposition with same sparsity
  - CuSparse triangular solve
    - Uses graph coloring to expose parallelizability
- CuIstl ILU(0) slower than multicore CPU
  - Confirms BDA results
  - Try something else



Diagonal ILU preconditioner

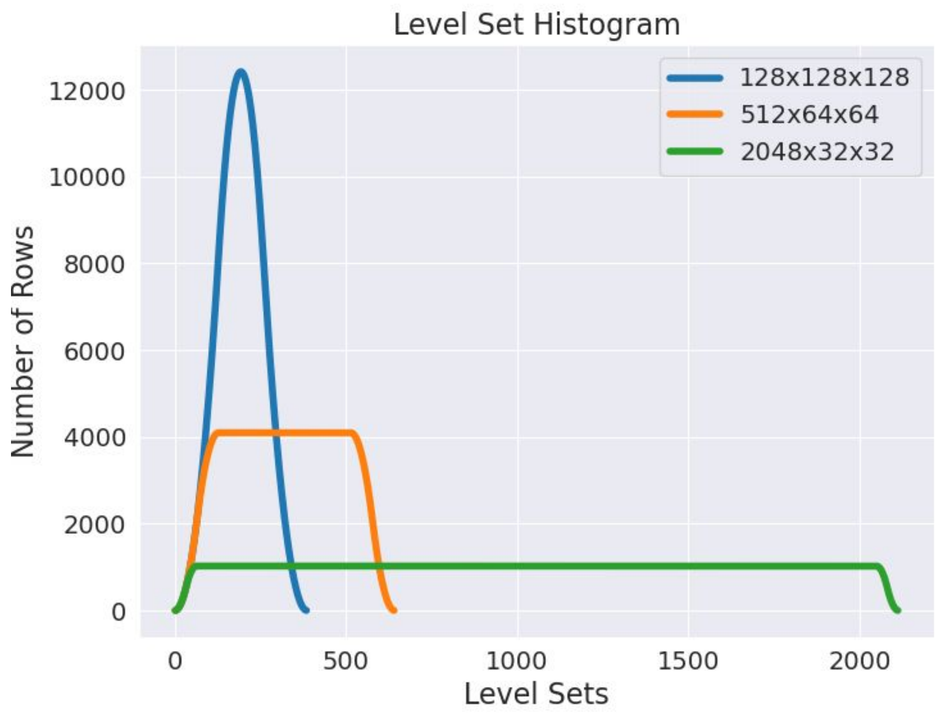
Second preconditioner: DILU

- A is estimated with these variables
  - $\bar{L}$, the strictly lower part of A
  - $\bar{U}$, the strictly upper part of A
  - E, the diagonal part of LU factorization
- Less memory usage than ILU(0)
  - Only stores diagonal from LU factorization!
  - Cheaper update, almost same apply
- Expected to be somewhat weaker than ILU(0)



DILU Implementation

- We write our own parallel implementation
  - Tried AMGX implementation (unsuccessful)
- Graph coloring extracts parallelism
- Grid dimensions influence parallelizability
  - “Cubic” grids are highly parallelizable
  - Long and narrow domains ⟶ more serial
  - Has to do with cell neighborships
- Rows reordered for better memory access
- Implemented on GPU and CPU (OpenMP)
- Serial CPU version added to DUNE



Parallelism visualized as histogram

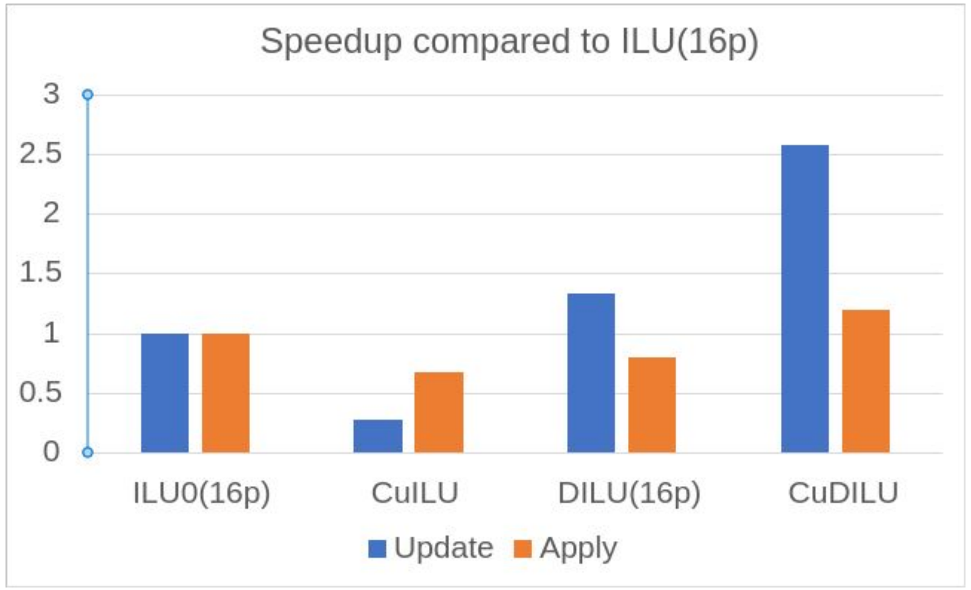
Y axis is number of parallel computations

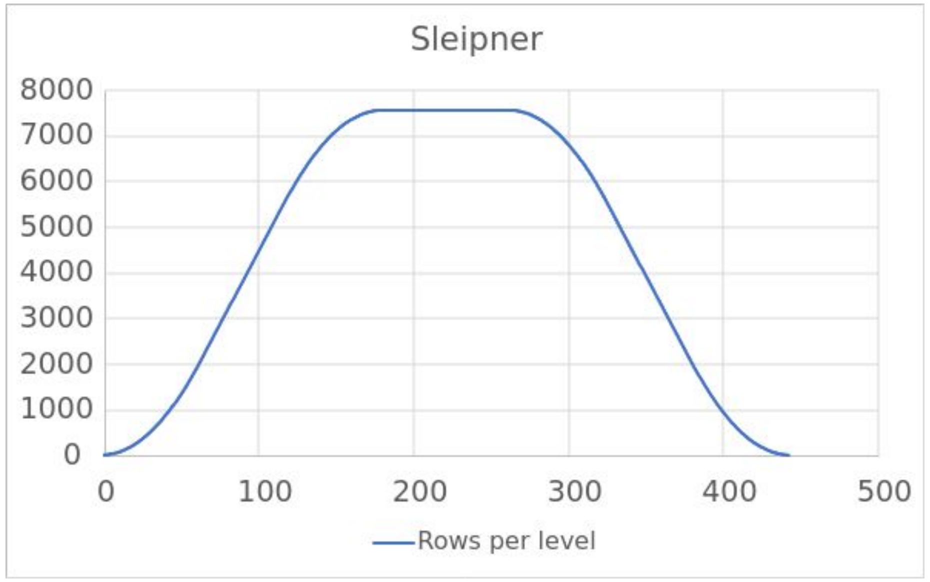
X axis is number of serial steps

CUDA DILU Beats CPU Counterpart

CUDA DILU is faster than ILU(0) and DILU on CPU for large cases

- Sleipner case with 2 million cells for instance:
  - High end consumer GPU and CPU



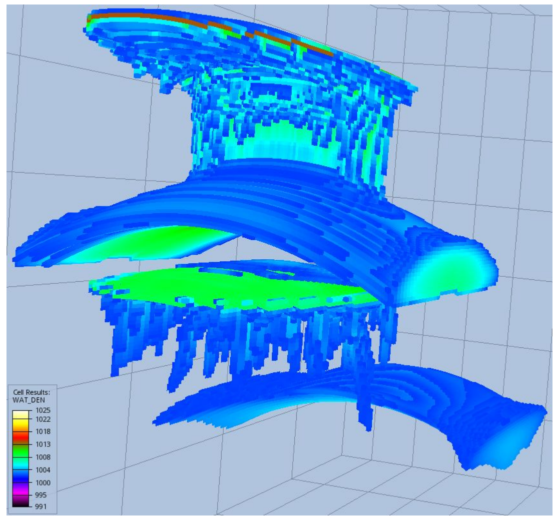


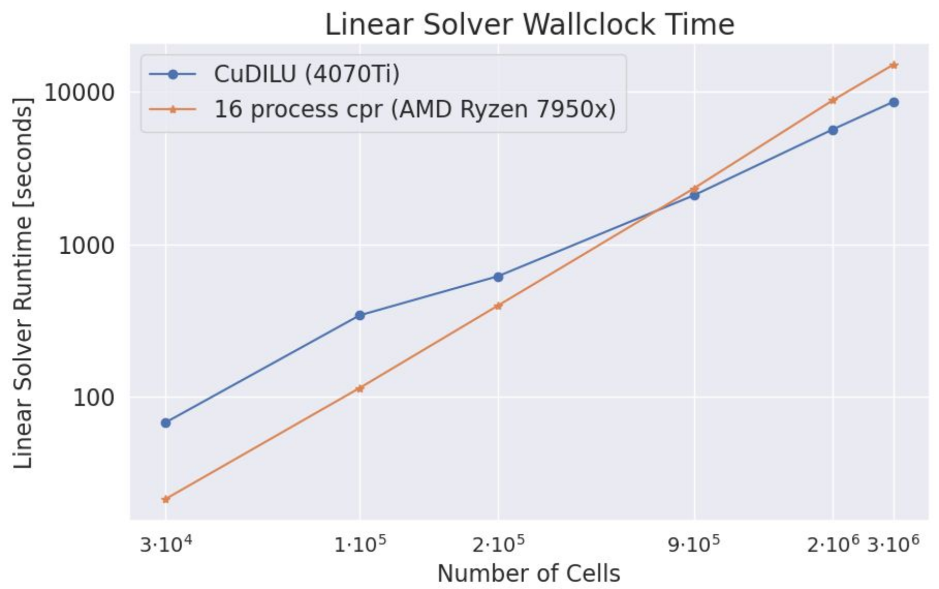
But OPM has much better preconditioners than ILU(0)...

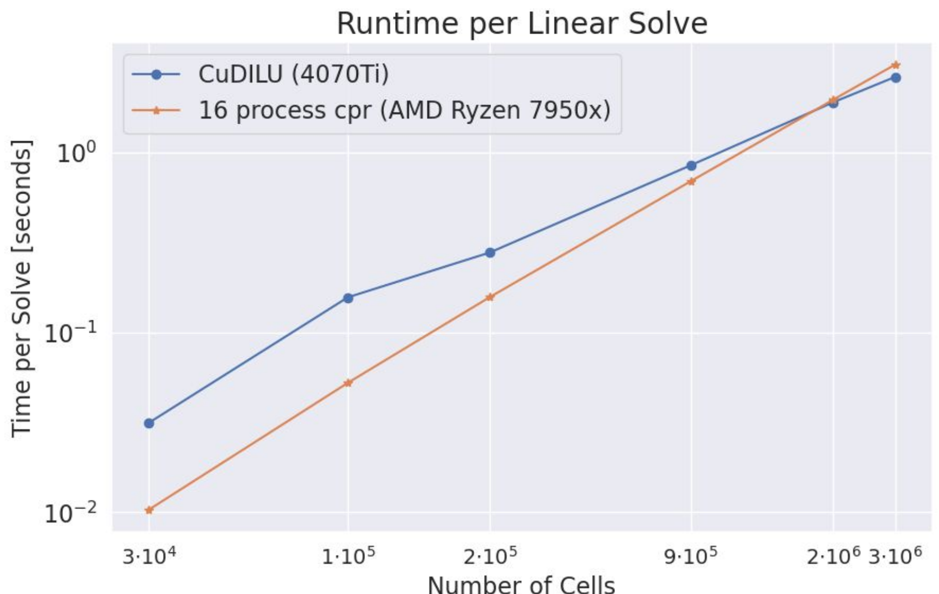
GPU DILU can be faster than CPU CPR

Culstl DILU vs CPU CPR

- SPE11C derived benchmarks with cubic grids
- Baseline CPR: two-stage preconditioner using AMG
  - More sophisticated, tailored for reservoir equations
  - Best preconditioner on the CPU
- GPU code is still faster for large simulations!
  - Each linear solve 17% faster on average




Linear Solver Wallclock Time
CuDILU (4070Ti)
16 process cpr (AMD Ryzen 7950x)
Linear Solver Runtime [seconds]
10000
1000
100
3·10^4
1·10^5
2·10^5
9·10^5
2·10^6 3·10^6
Number of Cells



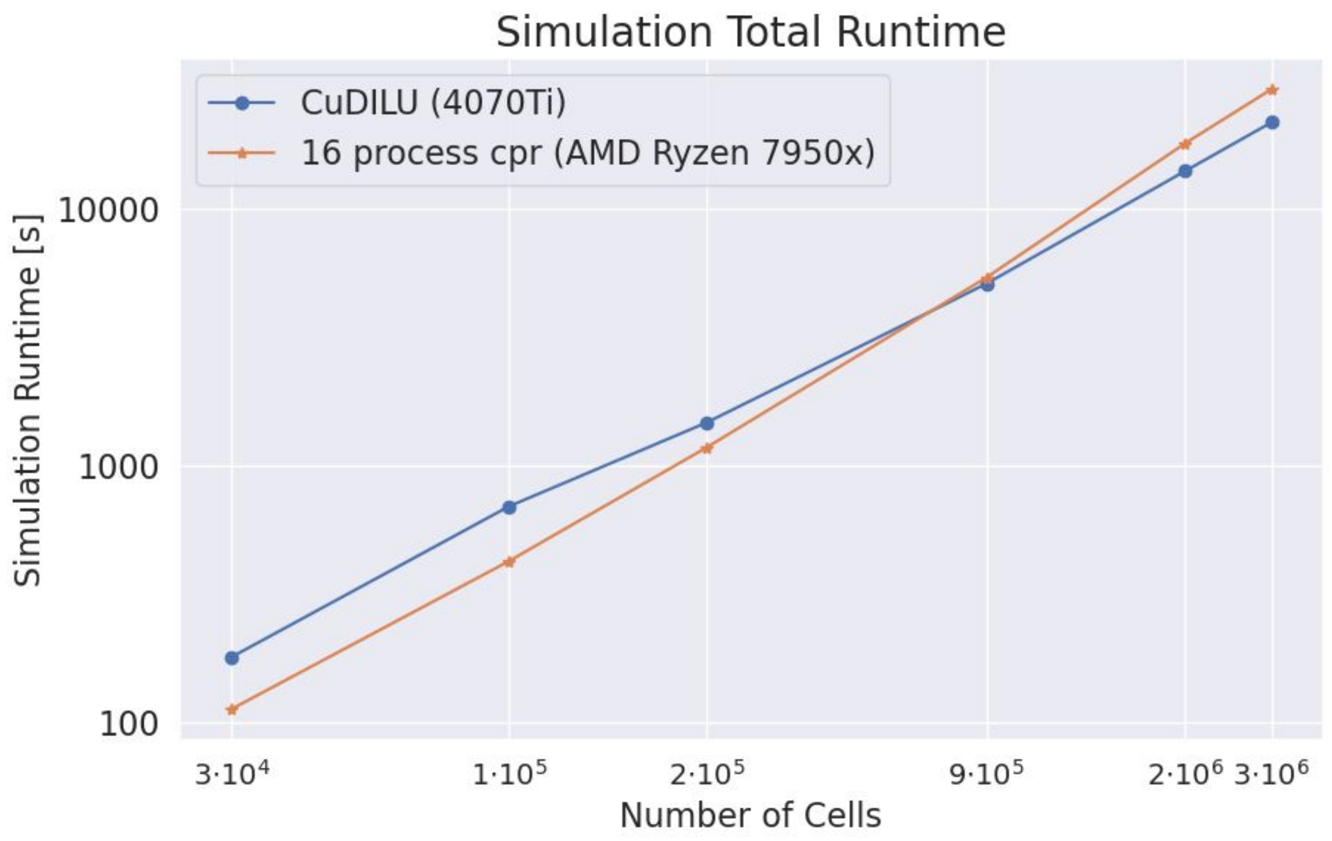
GPU Linear Solver Milestone

First full simulation with GPU + CPU > CPU with CPR

GPU + CPU: 21795s

CPU: 29292s (1.34 speedup)

- GPU: 39.6% linear solver
  - 33.3 its/linear solve
  - 5.7% of lin solv is constructor
    - graph coloring and mem transfer
  - The remaining 94.3%
    - Update (reuse coloring!)
    - Apply
- CPU: 51.6% linear solver
  - 4.4 its/linear solve

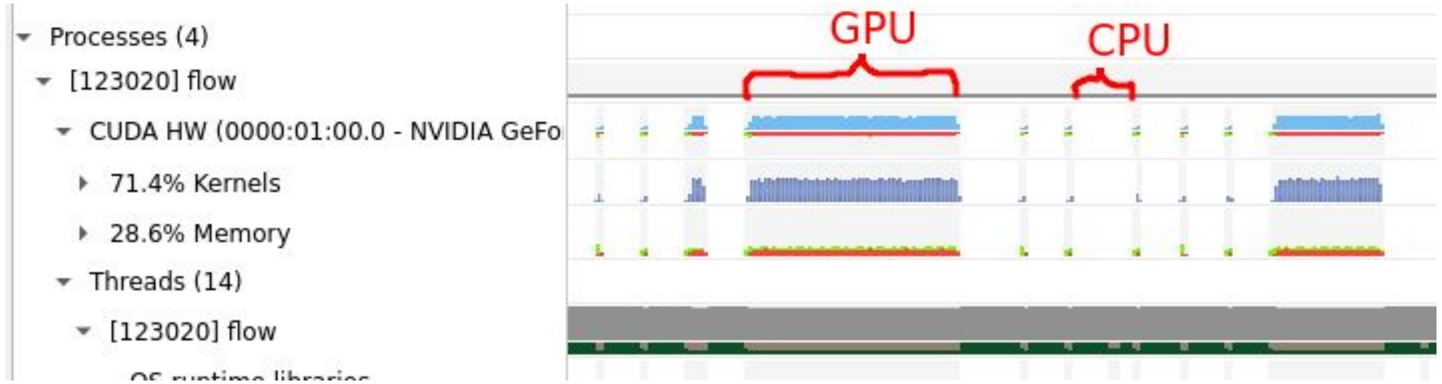


Linear Solver improvements in the making

- GPU Direct support for CUDA cards
  - Large speedup of memory transfers between GPUs
- Support for AMD cards
  - AMD cards supports HIP, and not CUDA
  - Hipify translates CUDA to HIP
    - Working proof of concept
  - Developers keep writing CUDA
    - Generate HIP code when needed only
- Slightly more memory efficient DUNE BiCGSTAB
- Large memory improvements in the linear solver
  - One excessive matrix copy on the GPU

Future work

- GPU parallelize more parts of the simulator
  - Petrophysical property evaluation
  - Linearization/Assembly (with AD)
- Test GPU CPR-AMG implementations from other libraries
  - Potentially combine with DILU as smoother
  - GPU SPAI expected to be very efficient for pressure system
- Only a small part of code supports GPUs
  - More significant in terms of runtime % on GPU


Processes (4)
[123020] flow
CUDA HW (0000:01:00.0 - NVIDIA GeFo
71.4% Kernels
28.6% Memory
Threads (14)
[123020] flow
GPU
CPU