A Multi-level Optimization method for Stencil Computation on the Domain that is bigger than Memory Capacity of GPU

Guanghao Jin
Tokyo Institute of Technology
JST-CREST
jingh@matsulab.is.titech.ac.jp

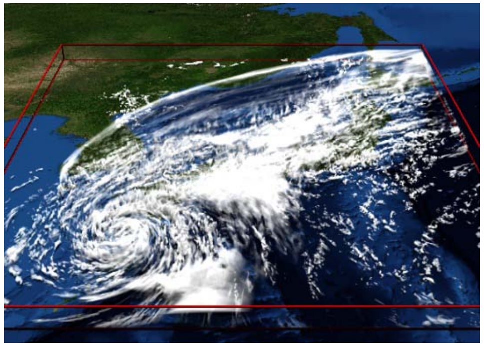
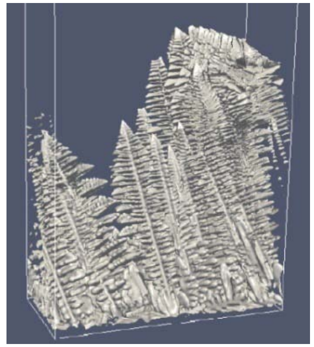
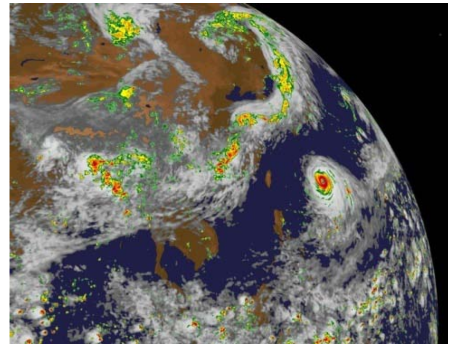
Toshio Endo
Tokyo Institute of Technology
JST-CREST
endo@is.titech.ac.jp

Satoshi Matsuoka
Tokyo Institute of Technology
JST-CREST
NII
matsu@is.titech.ac.jp

Presentation: Guanghao Jin

Stencil computation

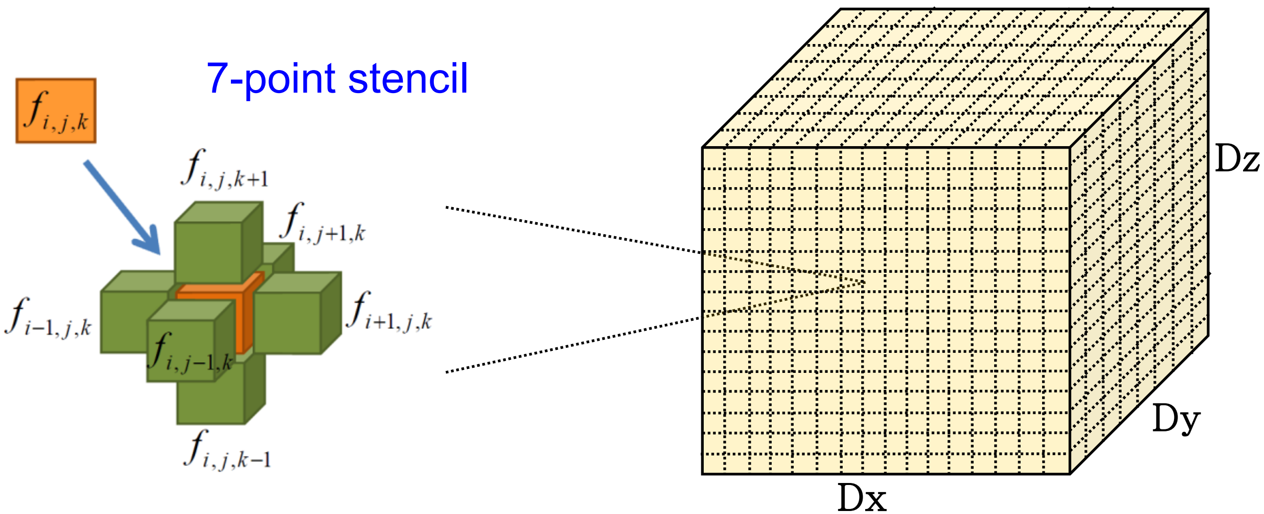
- Stencil computation (SC) is widely applied in scientific and engineering simulations.

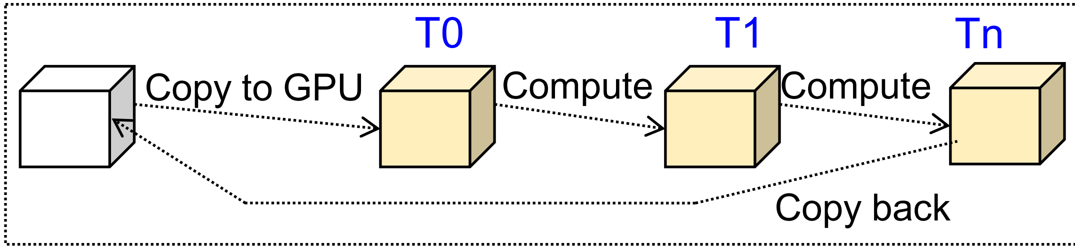
Fluid computation

- SC performs nearest neighbor computation on a spatial domain, updating each domain point based on its nearest neighbors, SC sweeps through the entire domain multiple times, called time steps.

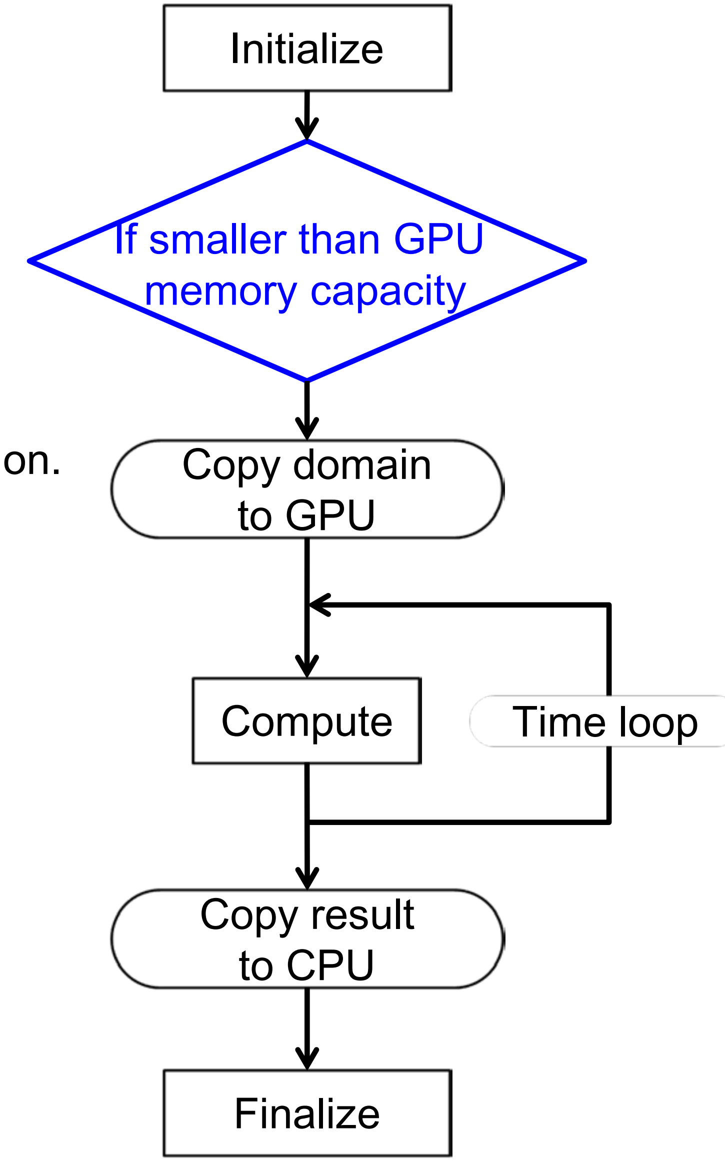
7-point stencil



Usual method on GPU

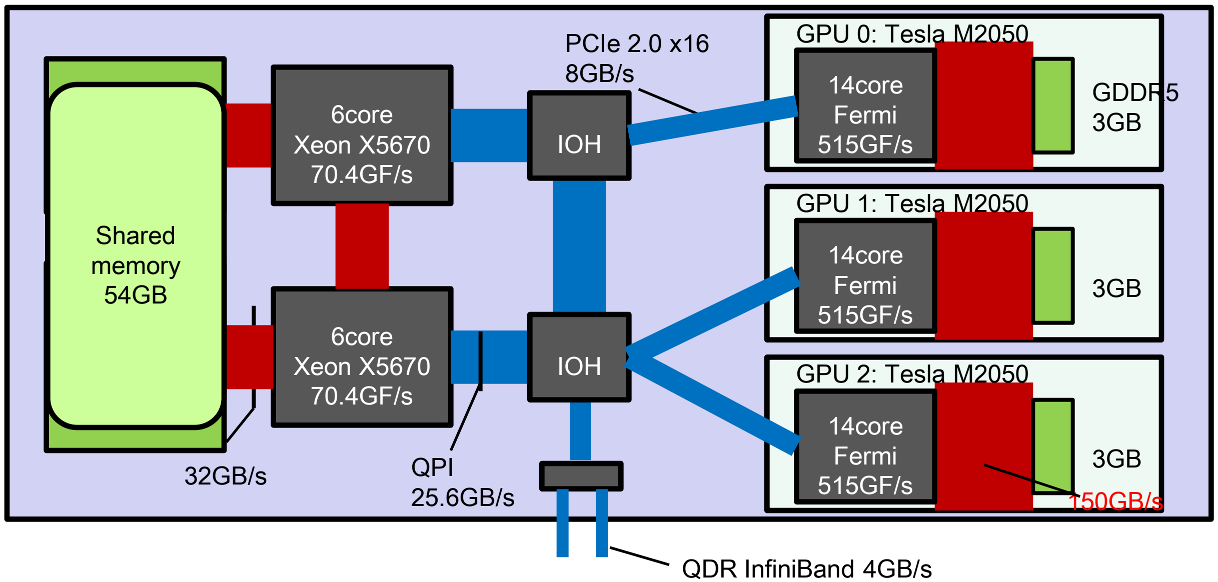


- The domain is initialized on CPU and sent to GPU.
- There are various flavors of iterative sweeps of stencil computation. The most commonly used technique is double buffering, which uses two grids, one designated for reading domain while the other is designated for writing result of domain in the current time step. For the next time step, the roles of the grids are swapped, and the grid that was written to is now read from.
- The final result will be copied from GPU to CPU.
- The domain is limited by the memory capacity of GPU
  As the domain grows for accuracy reason,
  more GPUs have to be employed to extend memory capacity.



TSUBAME 2.0

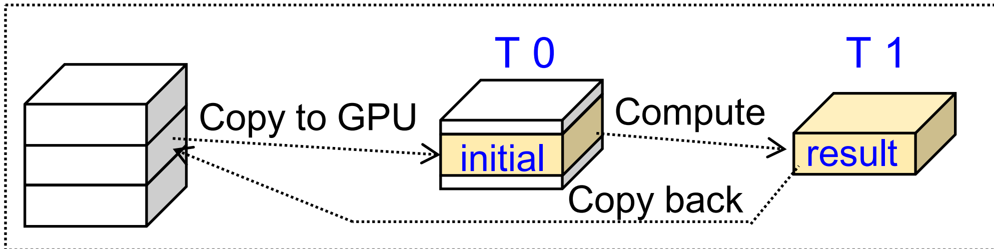
- The main part of TSUBAME2.0 consists of 1,408 Hewlett-Packard Proliant SL390s nodes. Each node has two sockets of 6-core Intel Xeon X5670 CPU (Westmere-EP) 2.93 GHz and 54GB DDR3 host memory. Each node is equipped with three Tesla M2050 GPUs which is attached to distinct PCI Express bus 2.0 x16 (8GB/s). Each GPU has 3 GB GDDR5 SDRAM device memory.



- It is great challenge that how to use both device memory and host memory efficiently. Enable the computation on the Domain that is bigger than Memory Capacity of GPU. We start this research from single GPU case.

- The Domain that is bigger than Memory Capacity of GPU ⇢ Bigger domain
  The Domain that is smaller than Memory Capacity of GPU ⇢ Smaller domain

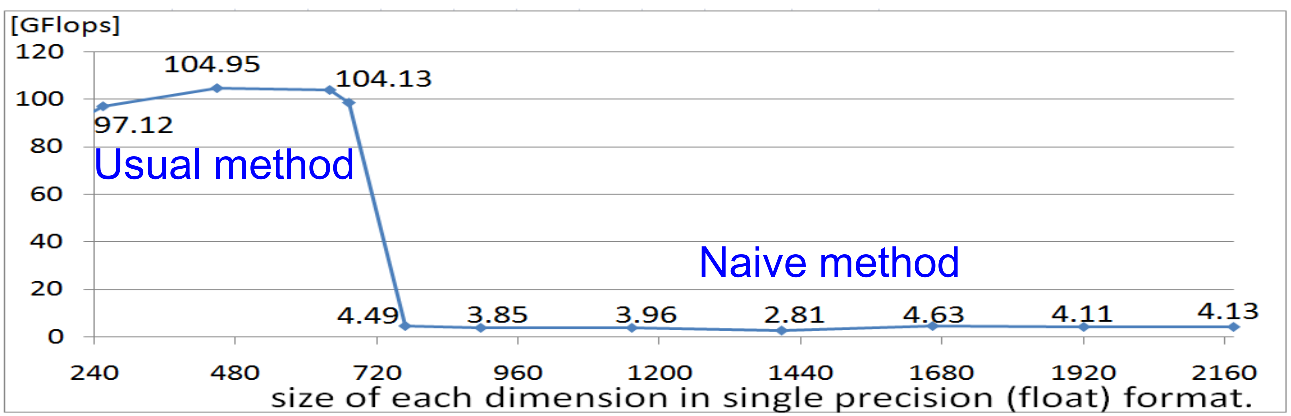
Naive method for bigger domain

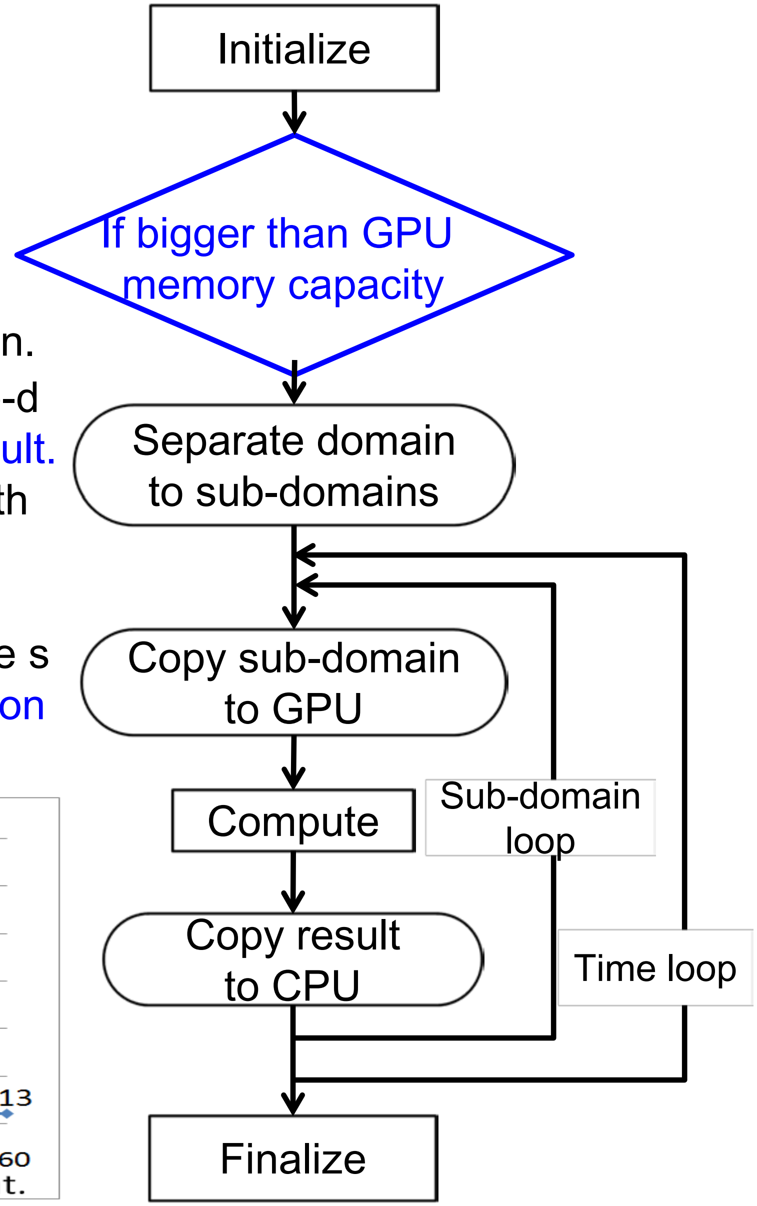


We separate the domain by Z direction to simplify the explanation.

- Separate the whole domain into sub-domains and copy each sub-domain(with ghost boundary) to GPU to compute 1 time step's result. Then it has to copy the result back and copy next sub-domain(with ghost boundary) to continue.

- Naive method copies each sub-domain to GPU to compute 1 time step and copy the result back. So, it causes frequent communication (via PCI-Express) between CPU and GPU.





Summary

Objective

Enable the computation on the domain that is bigger than GPU memory capacity.
Reach high performance at the same time

- Improve efficiency of GPU shared memory、GPU device memory、CPU memory.

How to

To improve locality, adopt 2-level temporal-blocking method

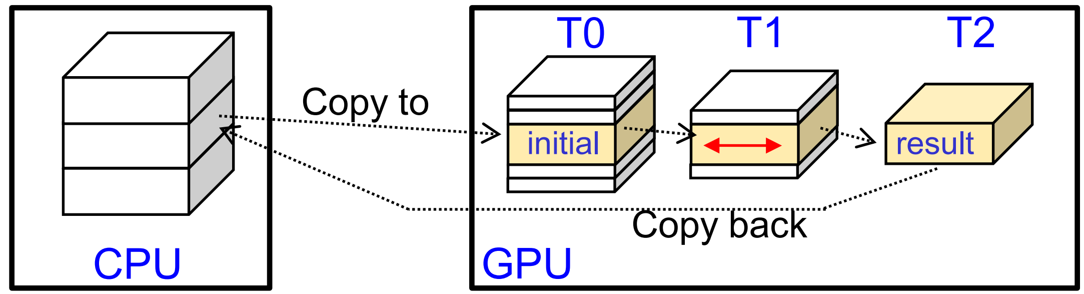
- Temporal-blocking to reduce communication via PCI
- Temporal-blocking for GPU kernel to reduce access times of global.

Furthermore, reduce redundant computation and communication.
Parallel communication with computation.

Temporal-blocking method

Multi-sub-domain Multi-time method(MM)

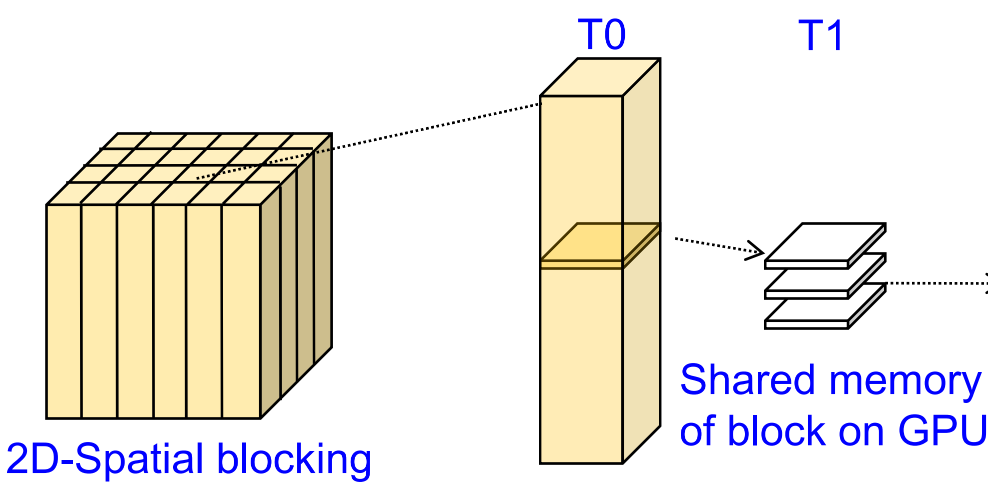
When it copies sub-domain to each GPU,
it will copy more ghost boundaries
to compute more time steps in local to
reduce communication times.

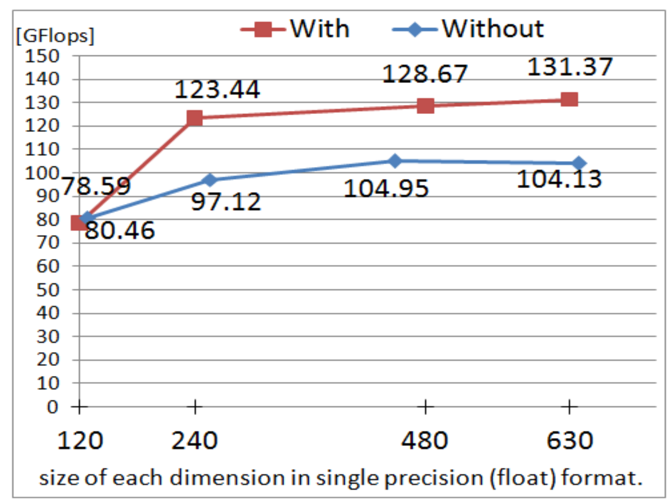


Compute sub-domain *i*

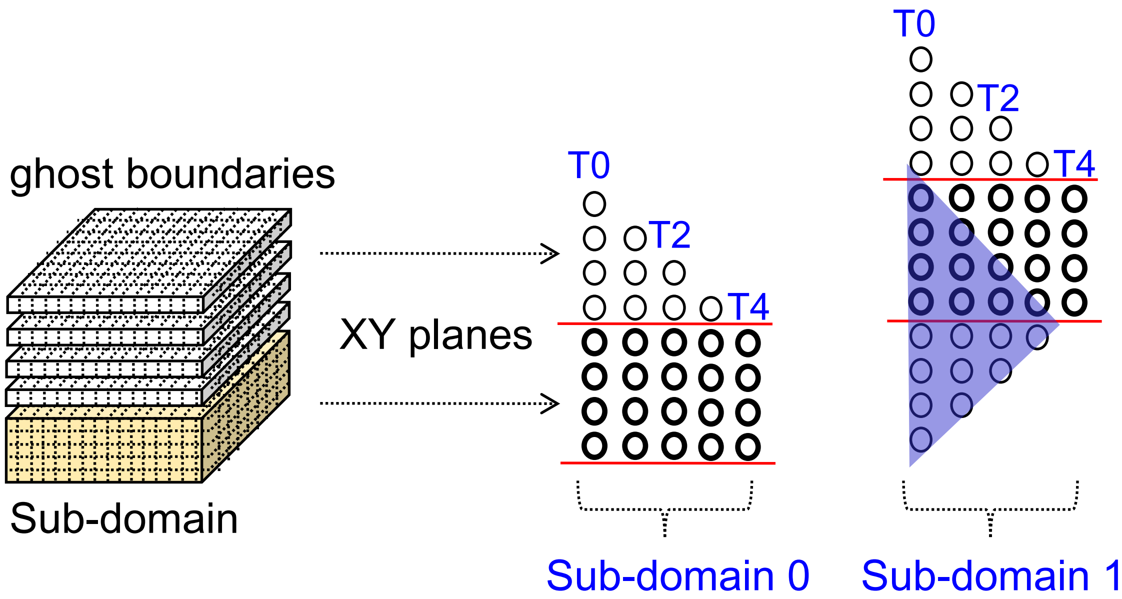
For GPU kernel

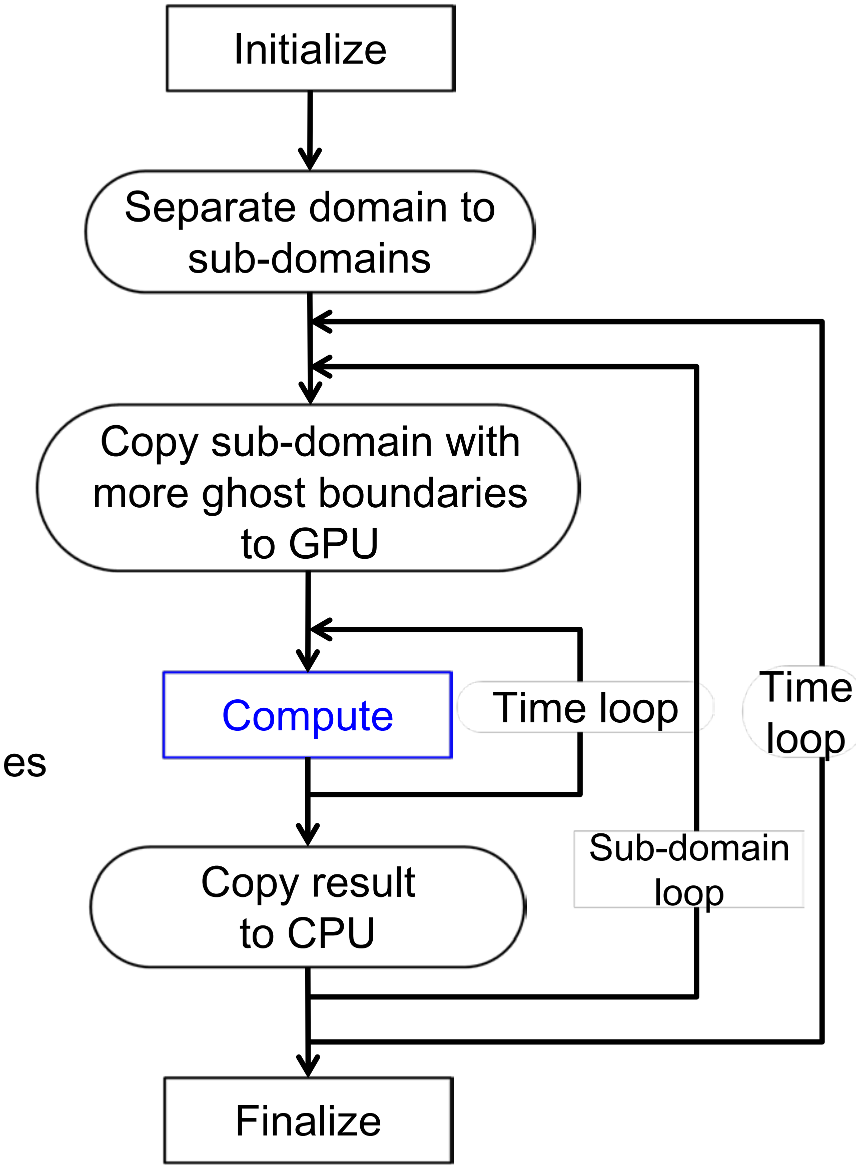
computing 2 time steps in 1 kernel as Figure explains.
It can reduce the cost of loading global memory.
As shared memory of GPU is limited, the time steps that can be computed in 1 kernel should be 2.





Optimization methods for bigger domain





MM

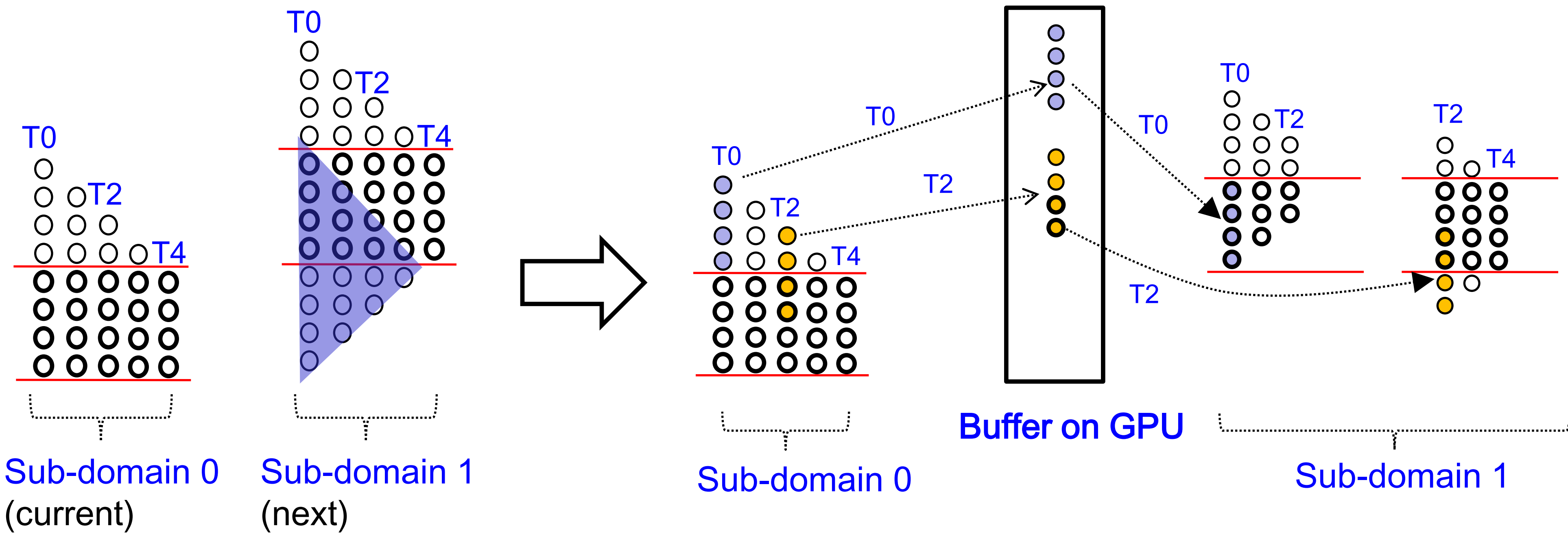
It separates the whole domain into sub-domains.
When copies sub-domain, it will copy more ghost boundaries to compute more time steps in local .

MMT

MM ＋ Temporal-blocking method for GPU kernel
※MM and MMT remain
redundant communication (ghost boundaries)
and computation (intermediate steps) problem.

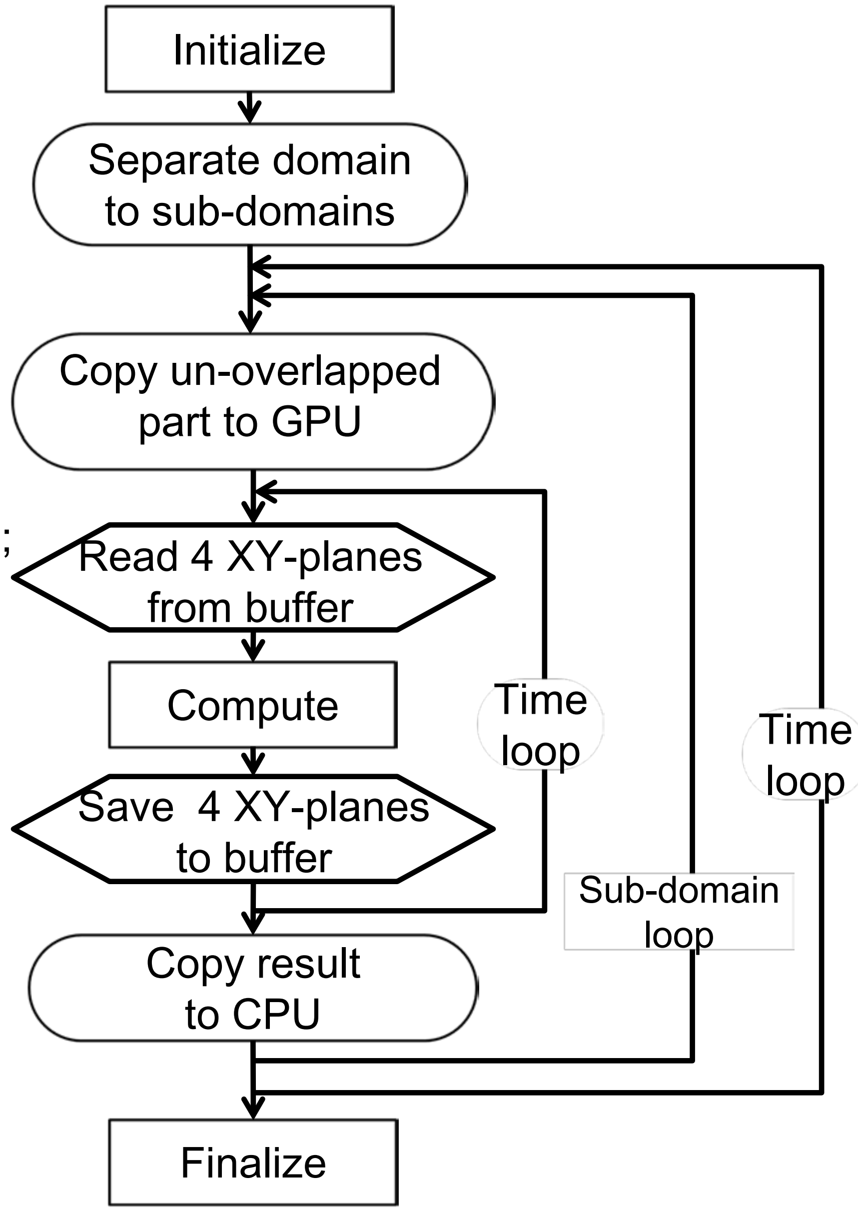
Buffer-copy method

- MM and MMT method have overlapped part between current and next.
- It store some overlapped part at current and reuse at next.
  - (1) It stores 4 overlapped XY-planes at every 2 time steps along the borderline (divides overlapped and un-overlapped parts) when computes current sub-domain
  - (2) When compute next sub-domain, it supplies 4 overlapped XY-planes to the correspondent un-overlapped part at every 2 time steps.
- By this way, it can figure out the correct result of un-overlapped part after every 2 time steps till final time step.

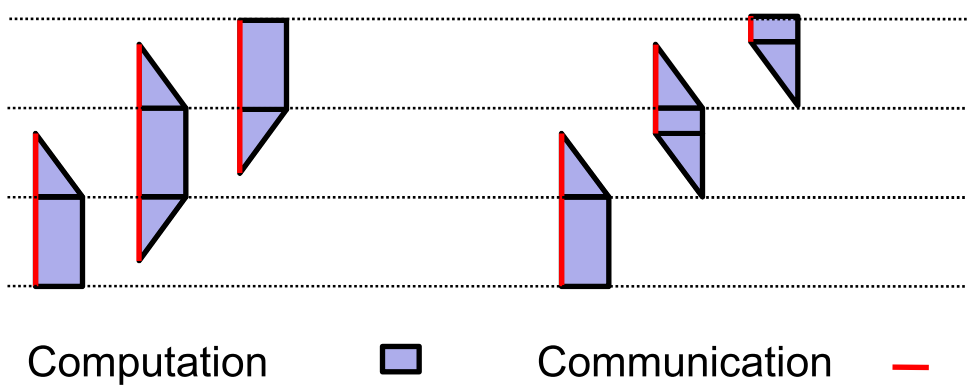


MMTB (MMT+ buffer-copy)

- **For**(i = 0 ; i < TTI ; i += TTS)
  - **For** (j = 0 ; j < NSD; j += 1){
    - // If sub-domain is in the middle
    - **Copy** un-overlapped initial from CPU to GPU;
    - **For**(k = 0; k < TTS; k += 2){
      - **Supply** 4 XY-planes from buffer;
      - **Read** Un-overlapped part & 4 XY-planes,
        Compute 2 time steps in 1 kernel;
      - **Store** 4 XY-planes to buffer for next sub-domain;
      - Swap the grids; **}**
    - **Copy** result from GPU to CPU;}}

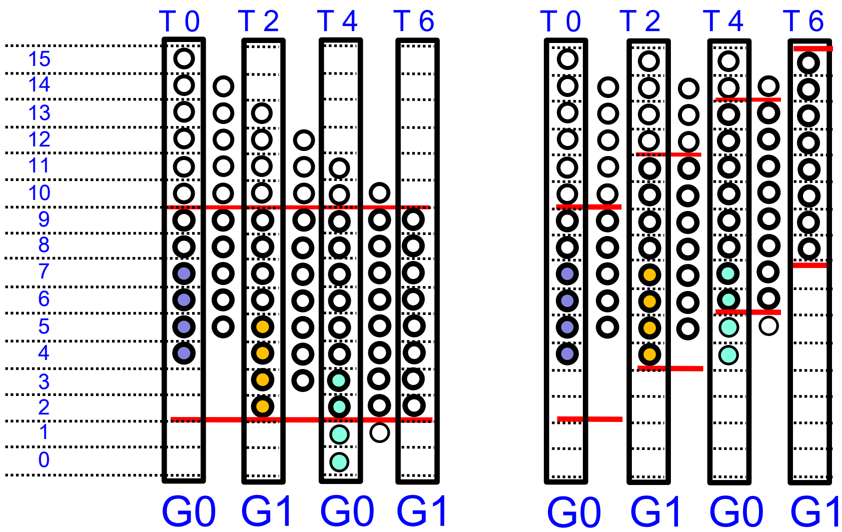


- **MMT vs. MMTB**



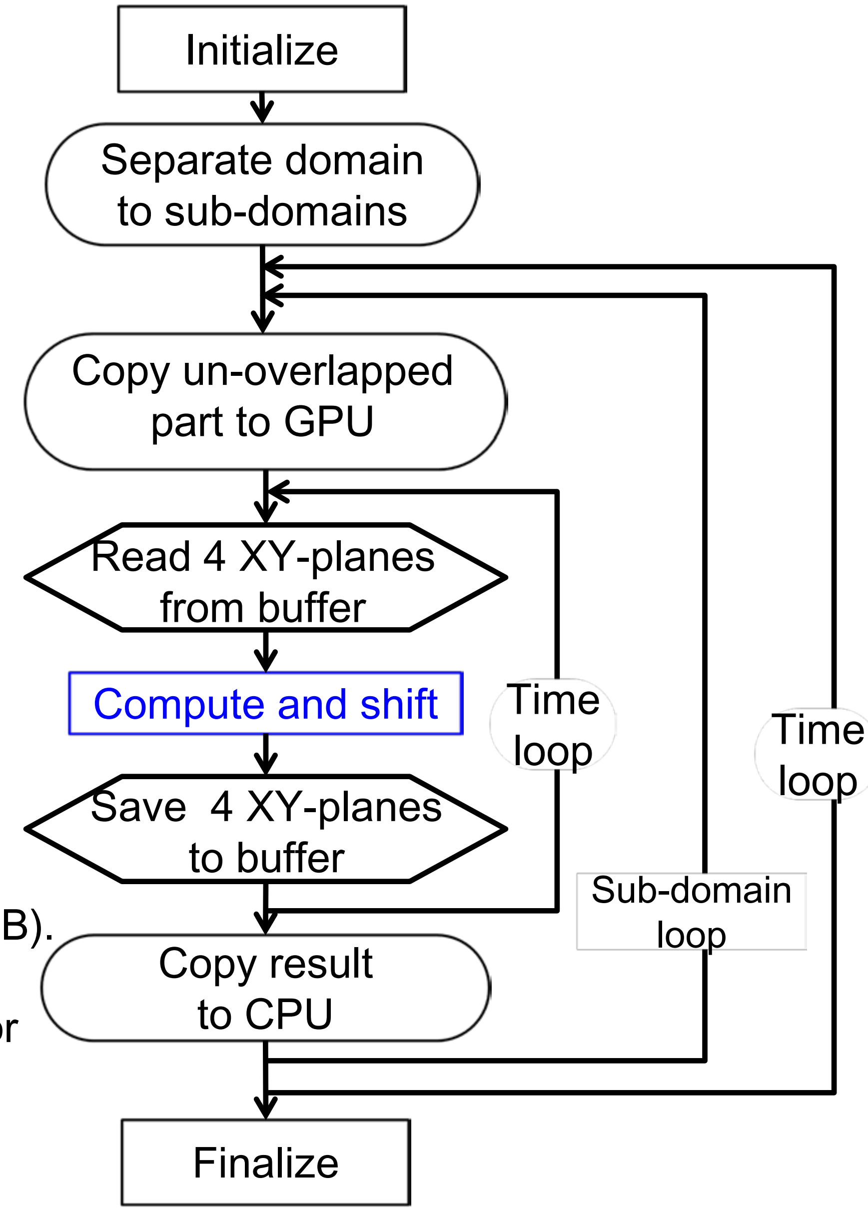
M-MMTB

- Although MMTB only computes un-overlapped part, it occupies more space than it needs as Figure explains. Memory-saving method shifts the result to fill the blank at each kernel.



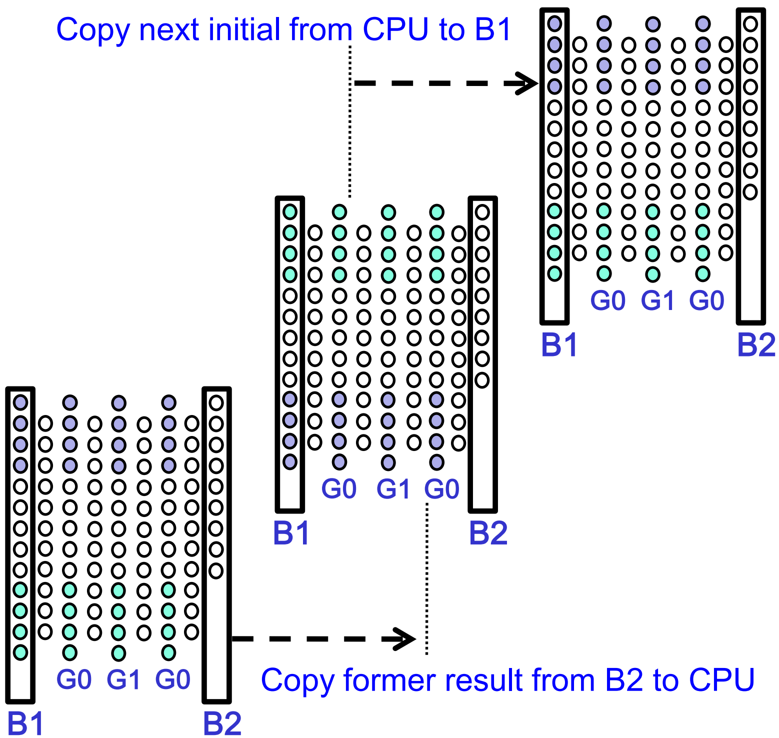
MMTB
Memory-saving method

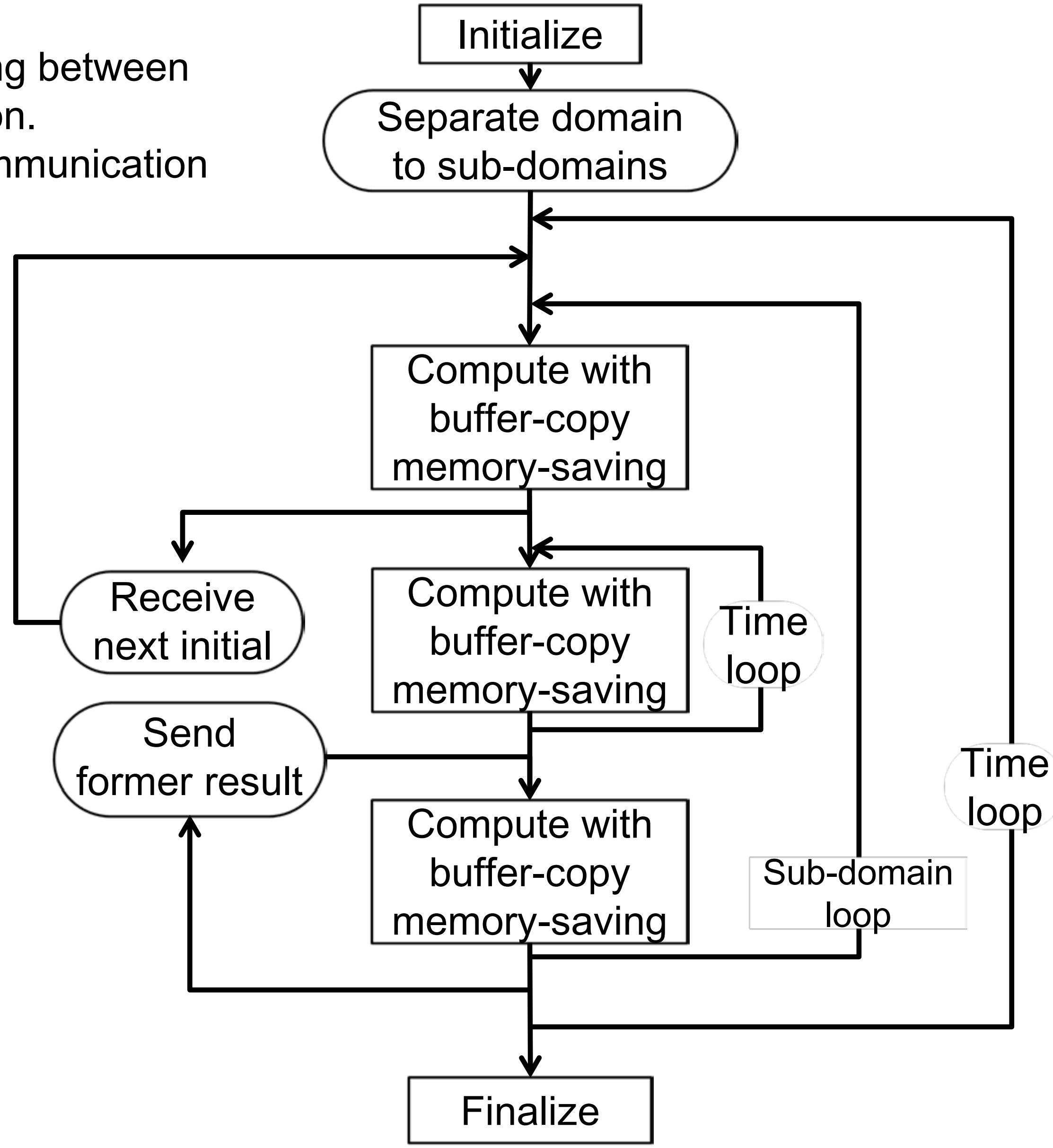
We call this method as M-MMTB (memory-saving + MMTB). Saving the memory space is attractive because we can use the saved space to contain more ghost boundaries, or to adopt bigger sub-domains. Both of them are expected to improve performance.



MP-MMTB

- MP-MMTB is further optimized by overlapping between computation and PCI-Express communication.
  It assigns 2 additional buffers to perform communication during the computation.
  B1 accepts initial of the next sub-domain.
  B2 sends the result of former sub-domain.





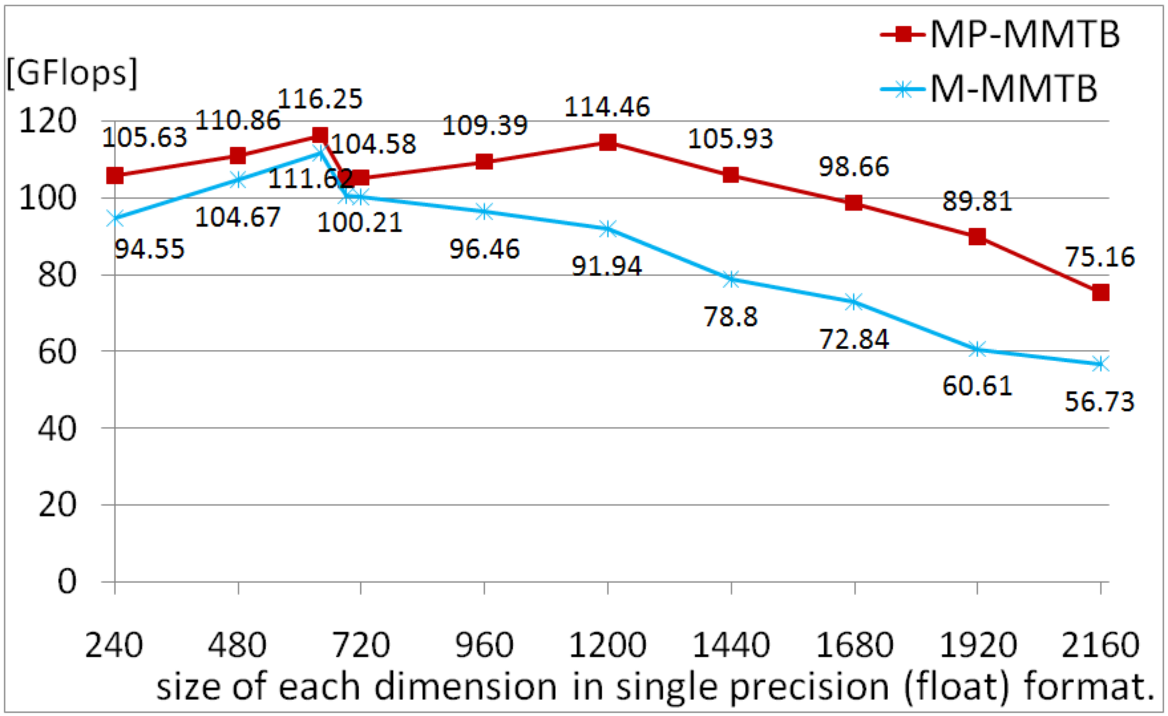
Performance evaluations

Environment

We evaluate our proposed methods on single GPU (NVIDIA Tesla “Fermi” M2050, 14 streaming multi-processor) of TSUBAME2.0. The host memory is 54 GB and device memory is 3 GB.
We select 7-point stencil computation for 3D diffusion equation.

MP-MMTB vs. M-MMTB: 240×240×240 ~ 2160×2160×2160

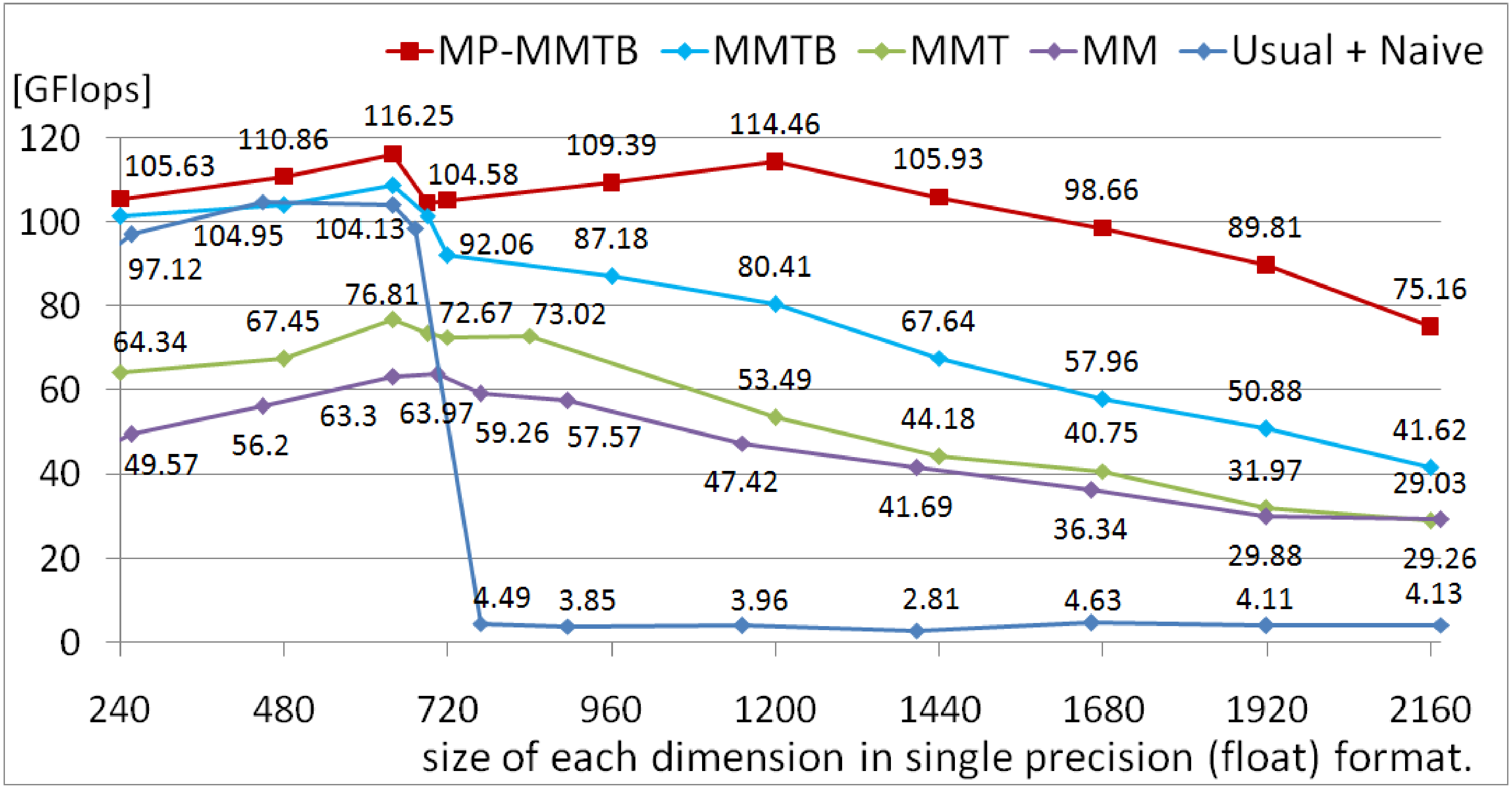
As Figure shows, MP-MMTB has better performance than M-MMTB since it can parallel the computation and communication.



Performance evaluations

MP-MMTB vs. Other methods

- MP-MMTB has more than 1.35 times better performance than other methods on an average.
- MP-MMTB has better performance than usual method on the smaller domains and 16.74 times better performance than naive method on the bigger domains.



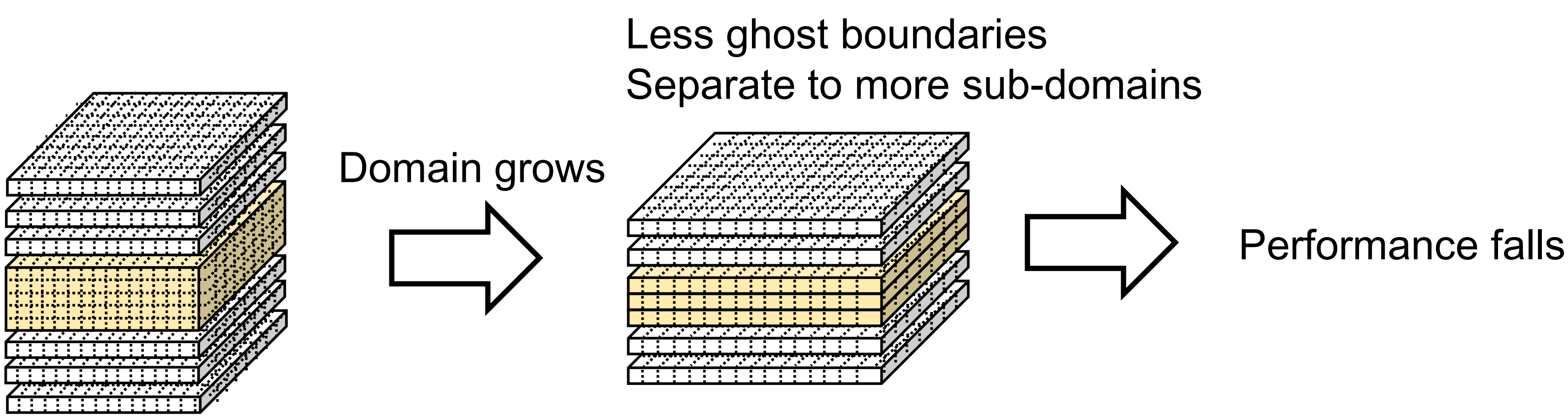
Limitation

- GPU memory is shared by
  2 grids (inside computation)
  2 buffers (communication)
  1 buffer (buffer-copy)

$$Dx \times Dy \times (Dz / NSD + 4) \times 4 + Dx \times Dy \times TTS \times 2 \leq \text{GPU memory capacity} \quad (1)$$

$$TTS < Dz / NSD \quad (2)$$

Dimension size		120	240	480	640	690	720	840	1200	1440	1680	1920	2160
MP-MMTB	NSD	2	2	2	2	3	3	8	16	30	48	80	108
	TTS	60	120	240	166	230	154	112	72	48	34	24	20



Conclusion

In this paper,
we propose a multi-level optimization method for the stencil computation on the domain that is bigger than the memory capacity of GPU while reaches high performance.

- It applies 2-level temporal-blocking method to enable fast computation on bigger domain
- Utilizes Buffer-copy method to reduce redundant cost.
- Applies memory-saving method to save space.
- Parallel communication and computation to achieve higher performance.

To achieve scalability, we will do research about multi-GPU case .

Question?

Ghost boundary

※ If the domain is divided into sub-domains, each sub-domain needs adjacent points which may belong to the other sub-domains. We call these adjacent points on the other sub-domains as ghost boundaries.