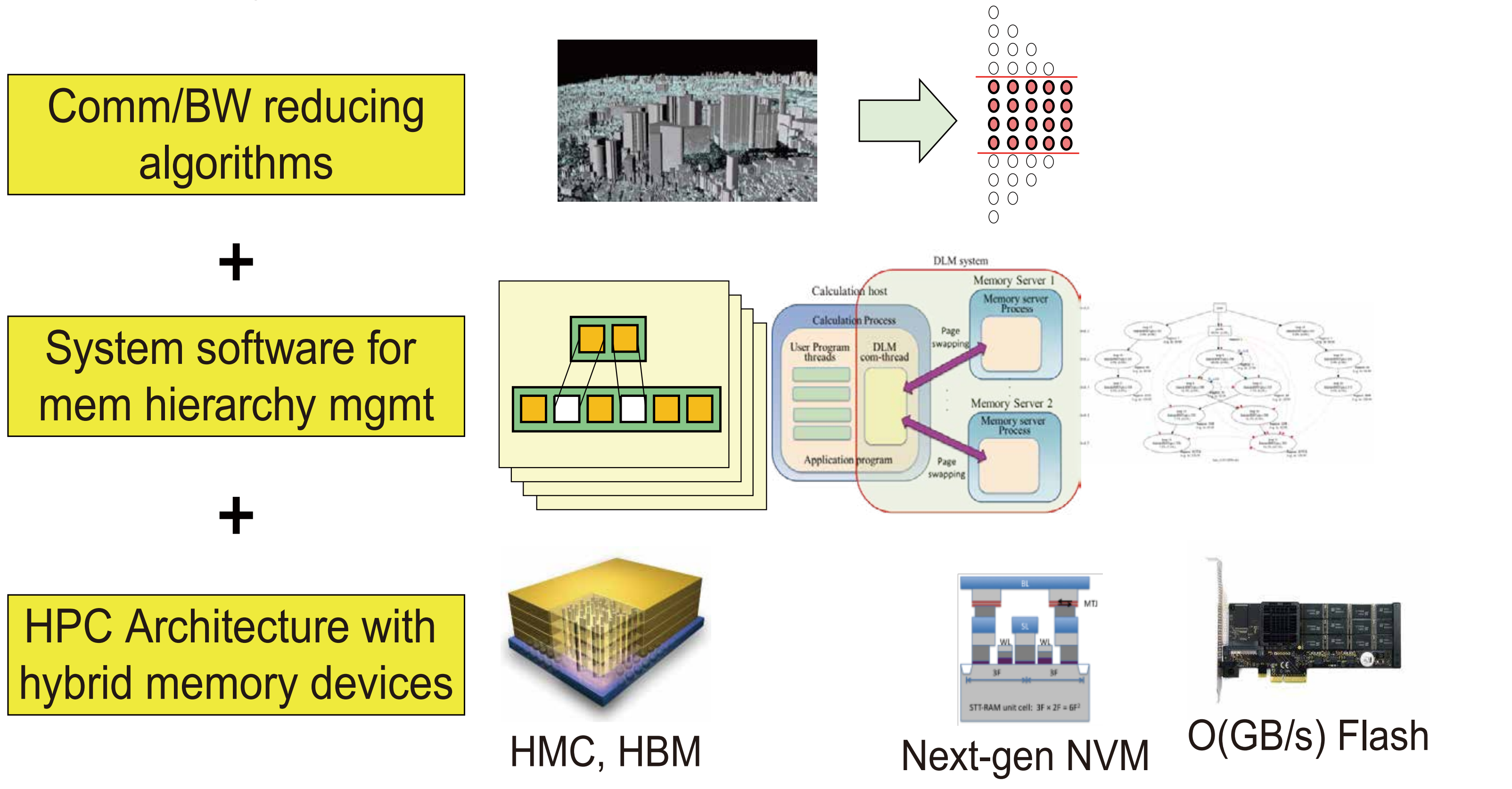
Dealing with Deeper Memory Hierarchy in Post-Petascale Era

Project Overview

On Exa-scale supercomputers, the “Memory Wall” problem will become even more severe, which prevents the realization of Extremely Fast&Big Simulations.

This project promotes research towards this problem via co-design approach among application algorithms, system software, architecture.


Comm/BW reducing algorithms

+

System software for mem hierarchy mgmt

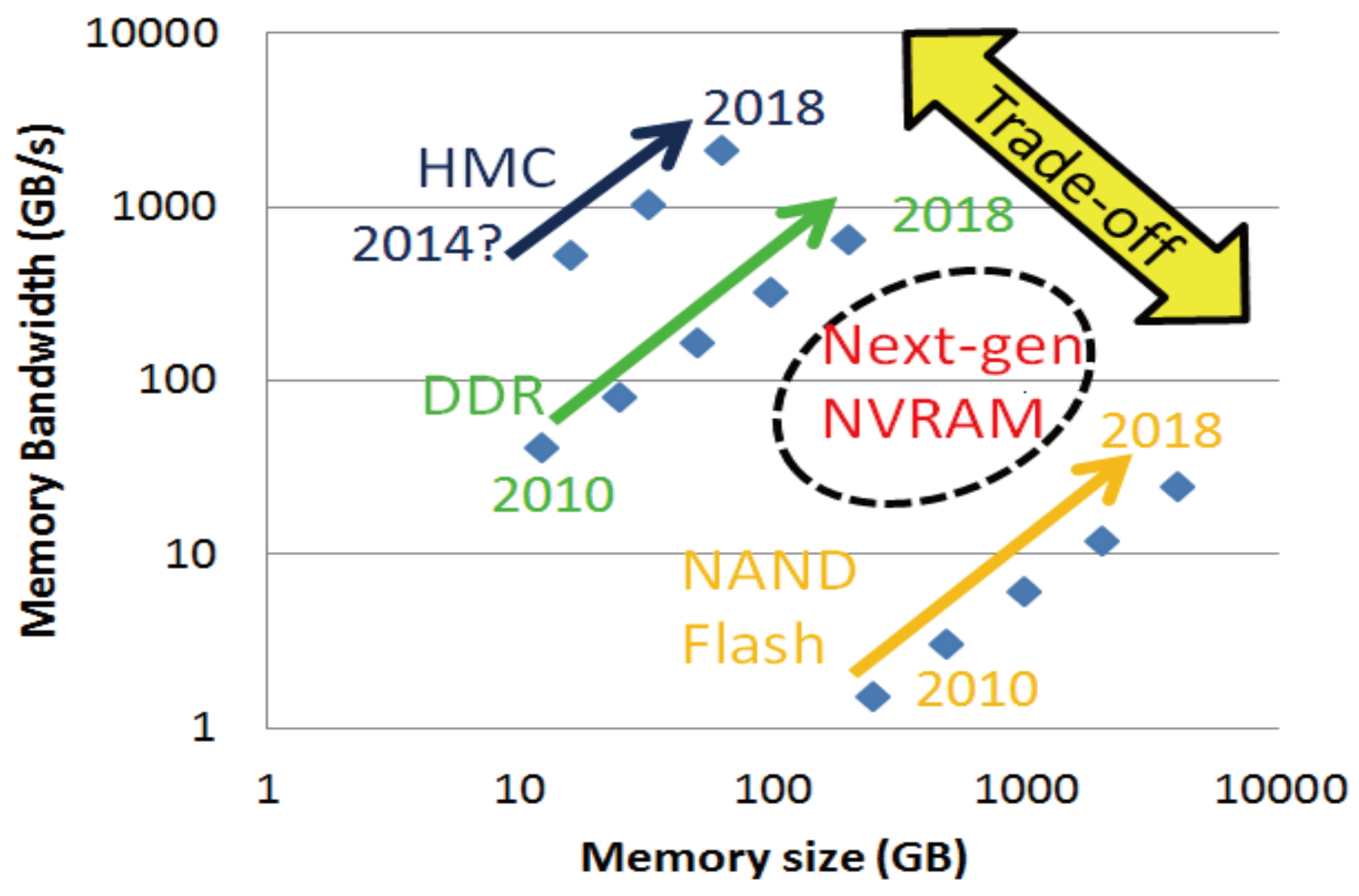
+

HPC Architecture with hybrid memory devices

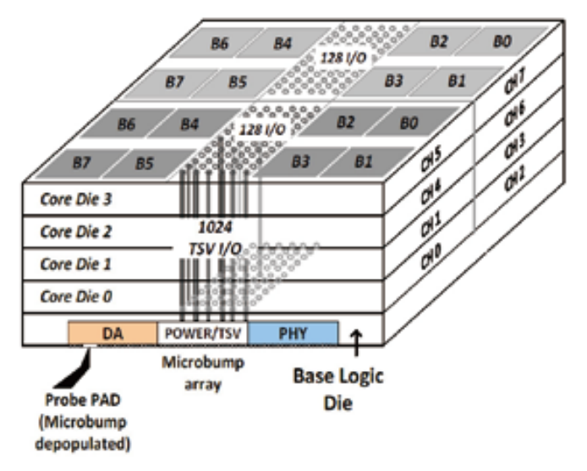
Target: Realizing Extremely Fast&Big simulations of O(100PB/s) & O(PB) in Exa-scale era

Target Architecture

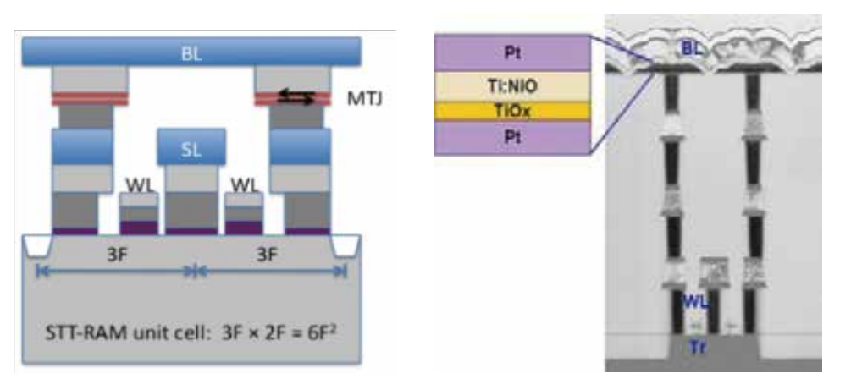
We consider near future HPC systems where each node has deeper memory hierarchy that consists of heterogeneous memory including NVM. We do not exclude usage of shared layer such as burst buffers.



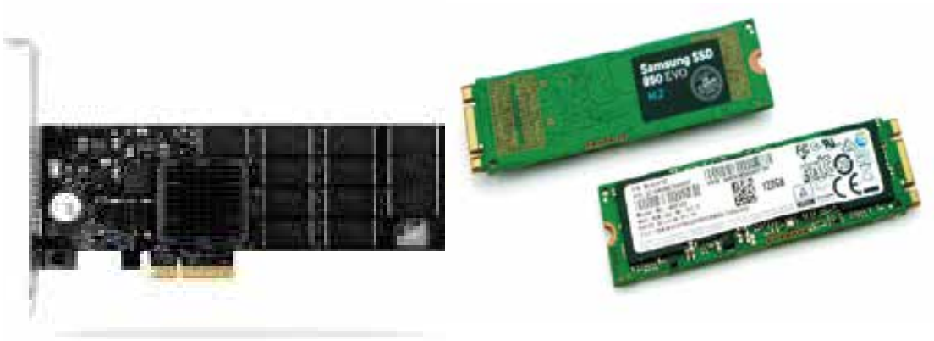
Power budget: ~400W for memory



High Bandwidth Memory (HBM):
DRAM chips are 3D-stacked with TSV technology. This has advantage in bandwidth over traditional DDR, but capacity tends to be limited.



Next-Gen Non-volatile Memory (NVM):
Several kinds of NVM such as STT-MRAM, ReRAM, FeRAM, 3D Xpoint will be available in near future. They have different properties in BW and capacity.



NAND Flash devices:
Not only SSDs with traditional SATA/SAS interfaces, but recent PCIe/m.2 SSDs with O(GB/s) bandwidth are already available.

HHRT: System Software for Swap

Motivation

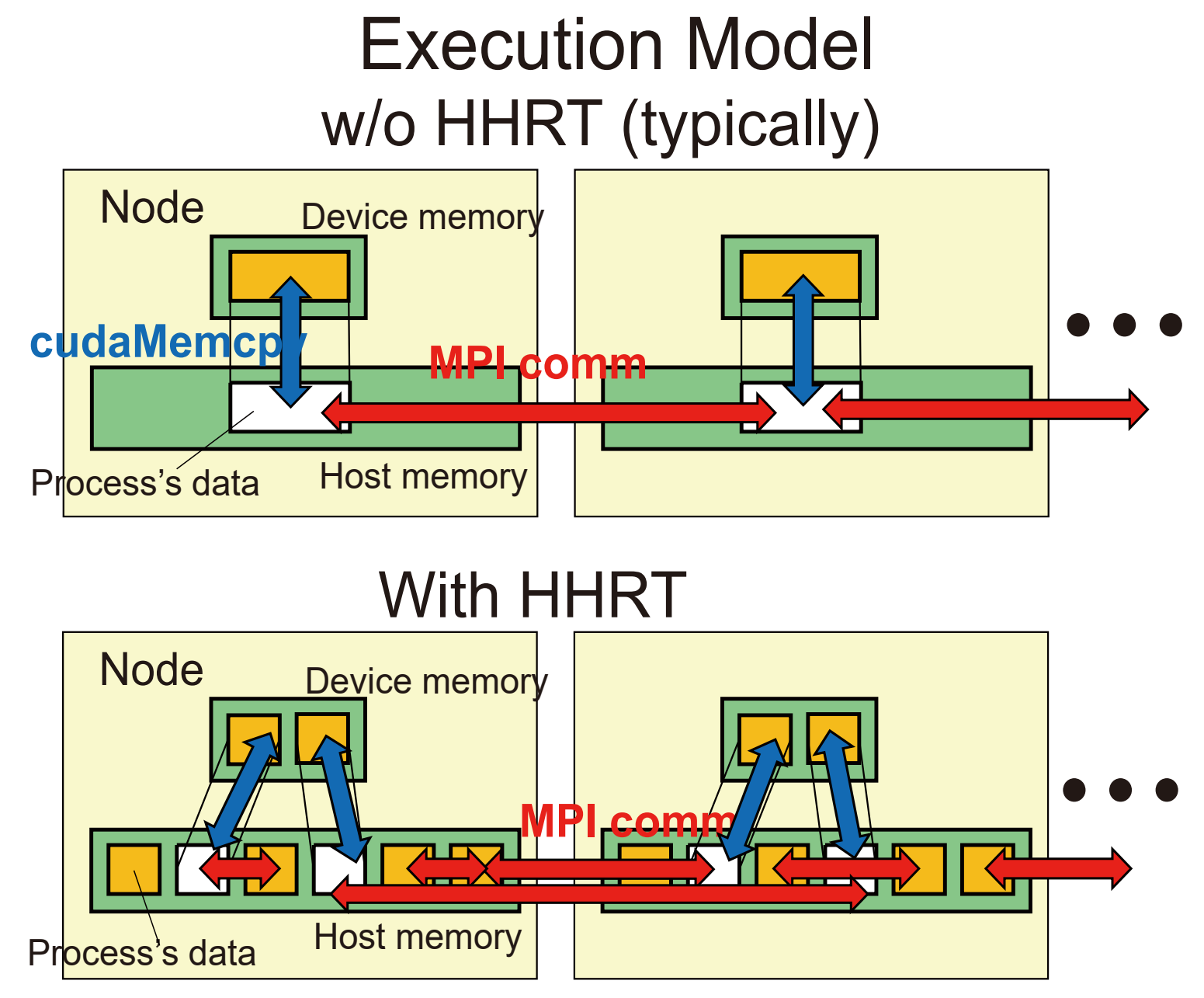
Towards achieving fast&big simulations, we need to exploit high speed of upper memory layer (e.g. GDDR/HBM on GPUs) and large capacity of lower memory layer (e.g. NAND Flash).

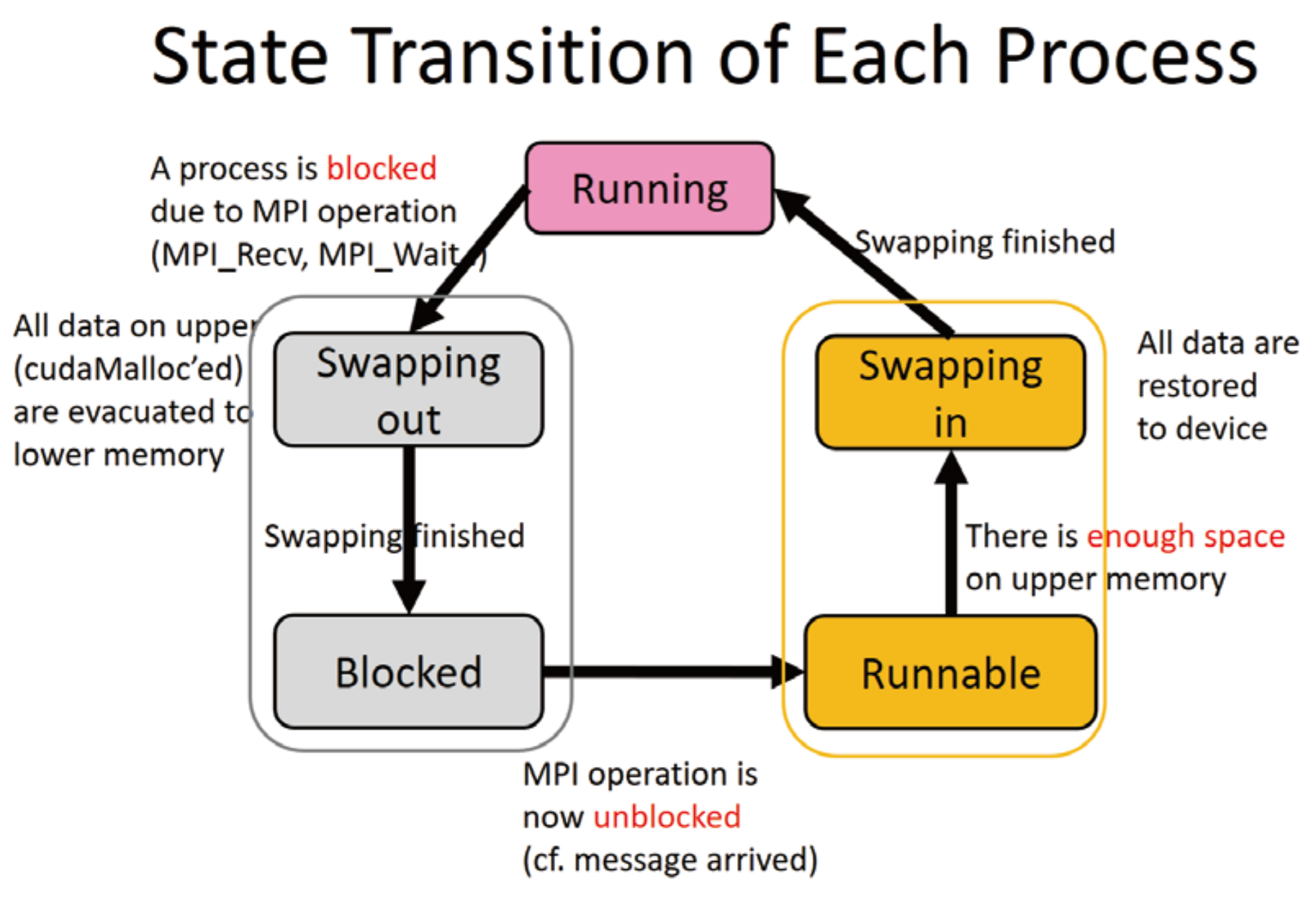
However, programming with consideration of memory hierarchy is troublesome.

Overview

To make memory hierarchy programming easier, we implemented system software, named ***HHRT (hybrid hierarchical runtime)***.

- HHRT automatically supports data swapping between GPU memory and lower layer memory (host memory, Flash SSD)
- HHRT supports “process-wise” swapping, not “page-wise” like OS.
- HHRT is used by user programs written in MPI and CUDA. It is a wrapper library for MPI/CUDA.
- Programmers still have responsibility to improve locality for better performance




State Transition of Each Process
Running
Swapping out
Blocked
Runnable
Swapping in
A process is blocked due to MPI operation (MPI_Recv, MPI_Wait)
All data on upper (cudaMalloc'ed) are evacuated to lower memory
Swapping finished
MPI operation is now unblocked (cf. message arrived)
There is enough space on upper memory
All data are restored to device
Swapping finished

https://github.com/toshioendo/hhrt

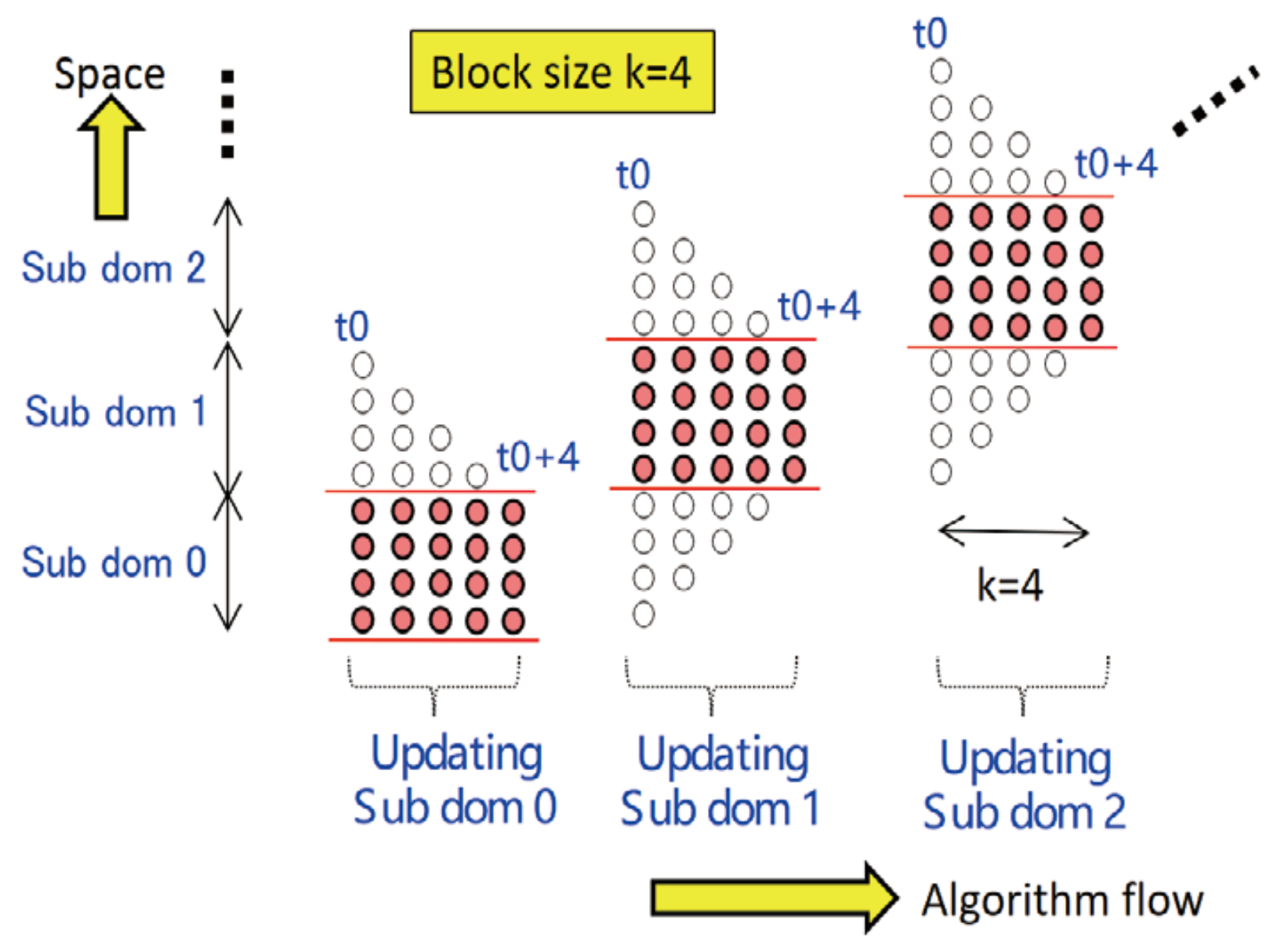
Integration with App Algorithm

Temporal blocking for stencil computation

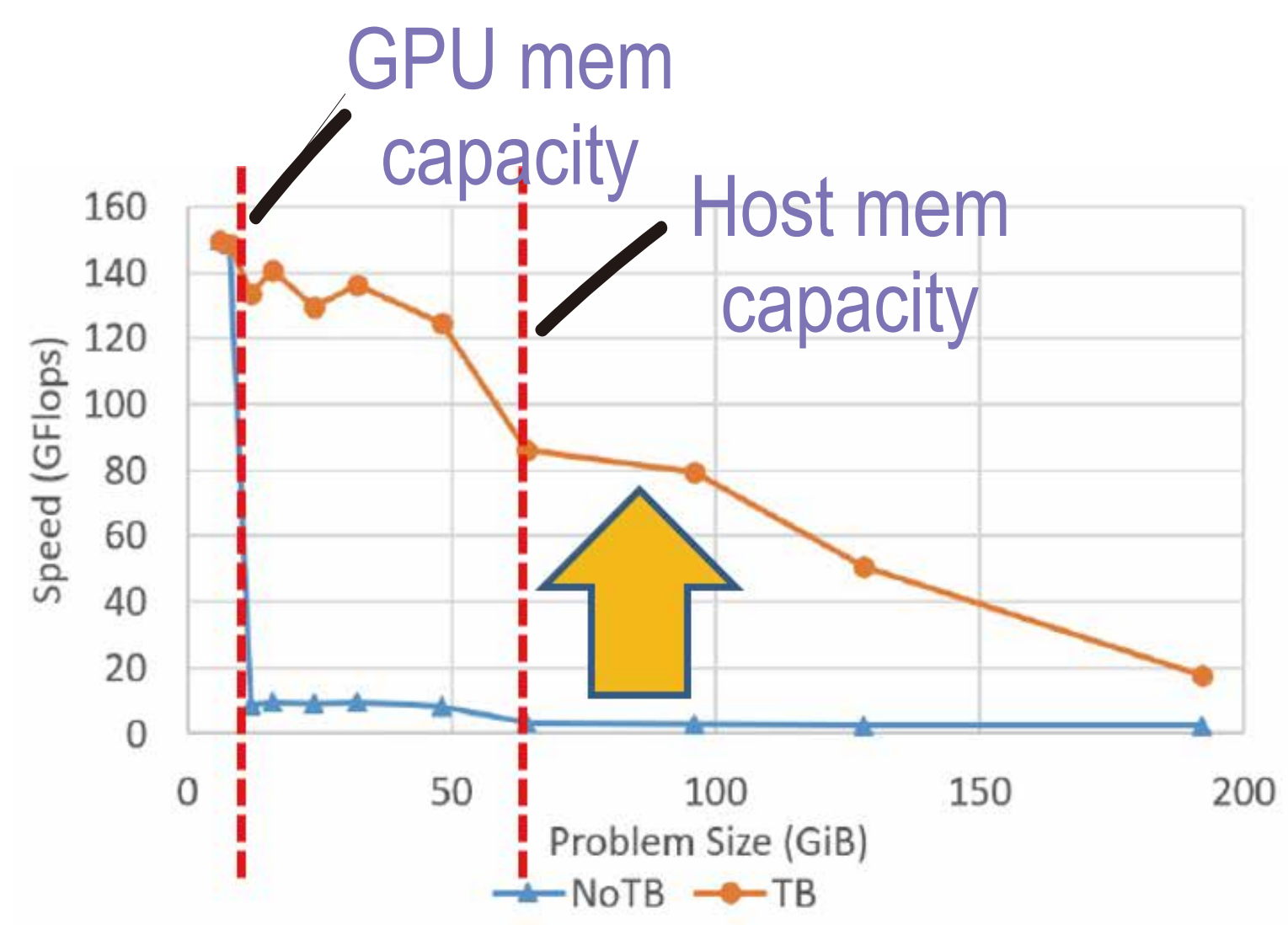
HHRT enables “larger” execution then upper memory, but app execution suffers from larger swapping cost, especially for stencil computation that has worse memory access locality.

A well known technique, “Temporal Blocking” (TB), which improves locality of stencil, achieves reasonable performance on HHRT.

Temporal Blocking

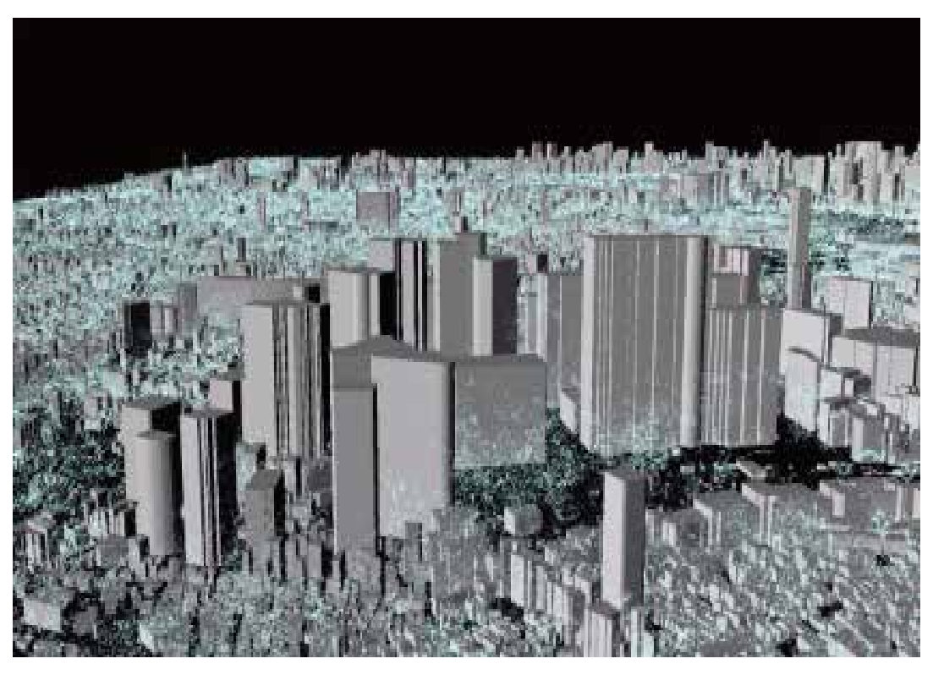


7-point stencil on a K40 GPU & 950pro m.2 SSD



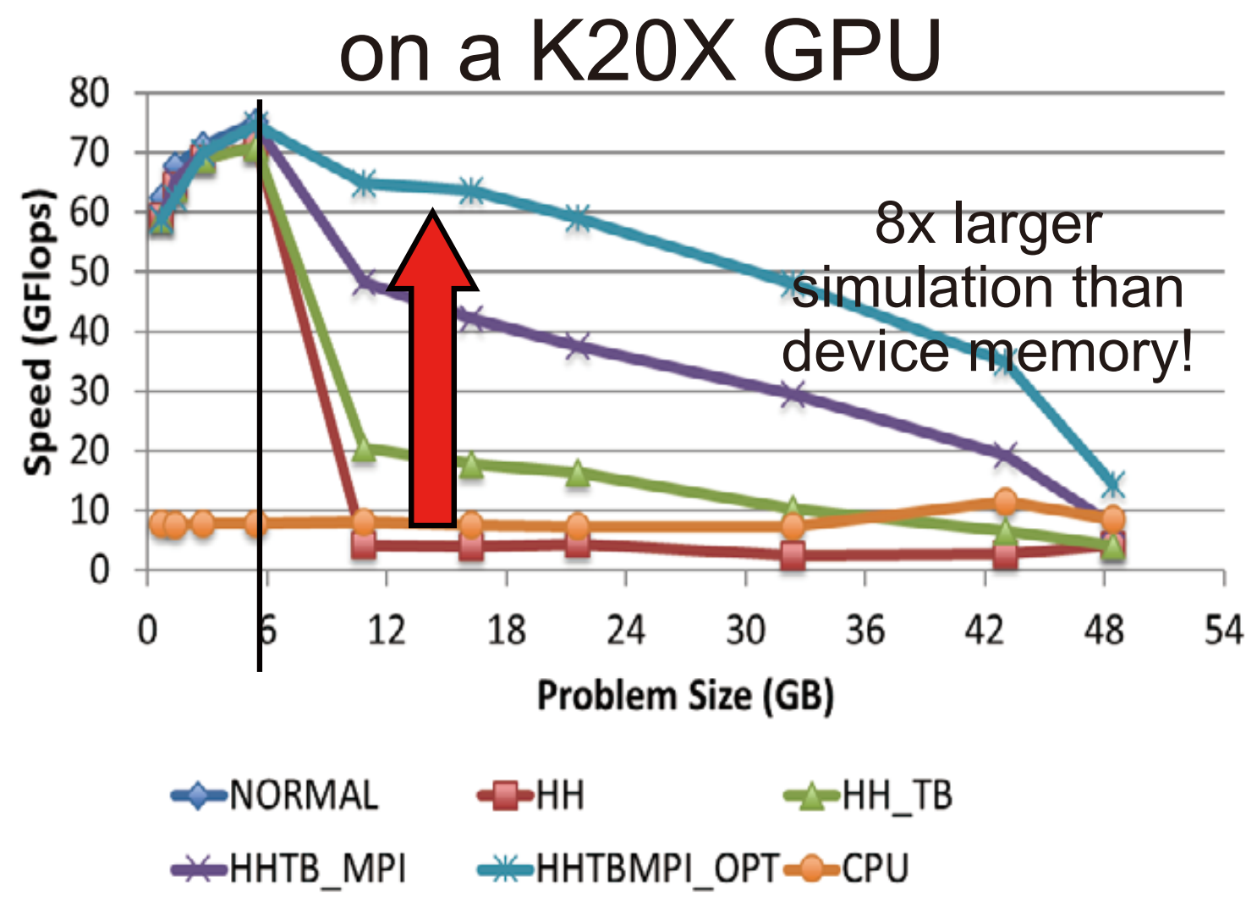
Integration with Real Simulation Application

We integrated our techniques with stencil-based city airflow simulation.



Original code on MPI+CUDA was developed by N. Onodera and T. Aoki. We integrated TB into it and executed on HHRT.

Airflow performance on a K20X GPU



T. Endo, G. Jin: Software Technologies Coping with Memory Hierarchy of GPGPU Clusters for Stencil Computations. IEEE Cluster 2014.
T. Endo, Y. Takasaki, S. Matsuoka: Realizing Extremely Large-Scale Stencil Applications on GPU Supercomputers. IEEE ICPADS 2015.
T. Endo: Realizing Out-of-Core Stencil Computations using Multi-Tier Memory Hierarchy on GPGPU Clusters. IEEE Cluster 2016.

PI: Toshio Endo (Tokyo Tech, endo@is.titech.ac.jp), supported by JST-CREST

http://www.gsic.titech.ac.jp/sc16