A Scalable Framework for Instant High-resolution Image Reconstruction

Peng Chen[1,3], Mohamed Wahib[3], Shinichiro Takizawa[3], Ryousei Takano[2], Satoshi Matsuoka[1,4]

1. Tokyo Institute of Technology, Dept. of Mathematical and Computing Science, Tokyo, Japan
2. National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
3. AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory, National Institute of Advanced Industrial Science and Technology
4. RIKEN Center for Computational Science, Hyogo, Japan

Background

- Computed Tomography (CT) is widely used
  - Medical diagnosis
  - Non-invasive inspection
  - Reverse engineering
- Possibility of obtaining high-resolution image
  - Rapid development in CT manufacturing
  - CMOS-based Flat Panel Detector (FPD, X-ray imaging sensor) become larger
    - 2048×2048, 4096×4096, etc
  - Micro focus x-ray become better and cheaper
- Complex computation for 3D image reconstruction
  - Filtering computation (or convolution), Back-projection
- The commonly used resolution : 256^3, 512^3, 1024^3

Problem statement

What happens if we start manipulating $(6k)^3$ and $(8k)^3$ volumes? [1]

Harry E. Martz, Clint M. Logan, Daniel J. Schneberk, and Peter J. Shull

- High-resolution CT image is important but not attainable
  1. Intensive computation
  2. Critical timing demanding for image reconstruction
  3. Huge memory capacity
     - 2048^3 : 32GB, 4096^3 : 256GB, 8192^3 : 2TB
- We use **ABCI supercomputer** (GPU-accelerated supercomputer) to solve this problem
- Challenges
  1. GPU is powerful in computation but **memory capacity** is limited
  2. How to optimize algorithms on **GPU**?
  3. How to use the **heterogeneous architecture** (CPUs, GPUs) ?
  4. How to optimally perform inter-process communication by **MPI** ?
  5. How to achieve **high performance** and **scaling**?

[1] Harry E. Martz, Clint M. Logan, Daniel J. Schneberk, and Peter J. Shull. 2017. X-ray imaging: fundamentals, industrial techniques, and applications. Boca Raton:CRC Press, Taylor & Francis Group.

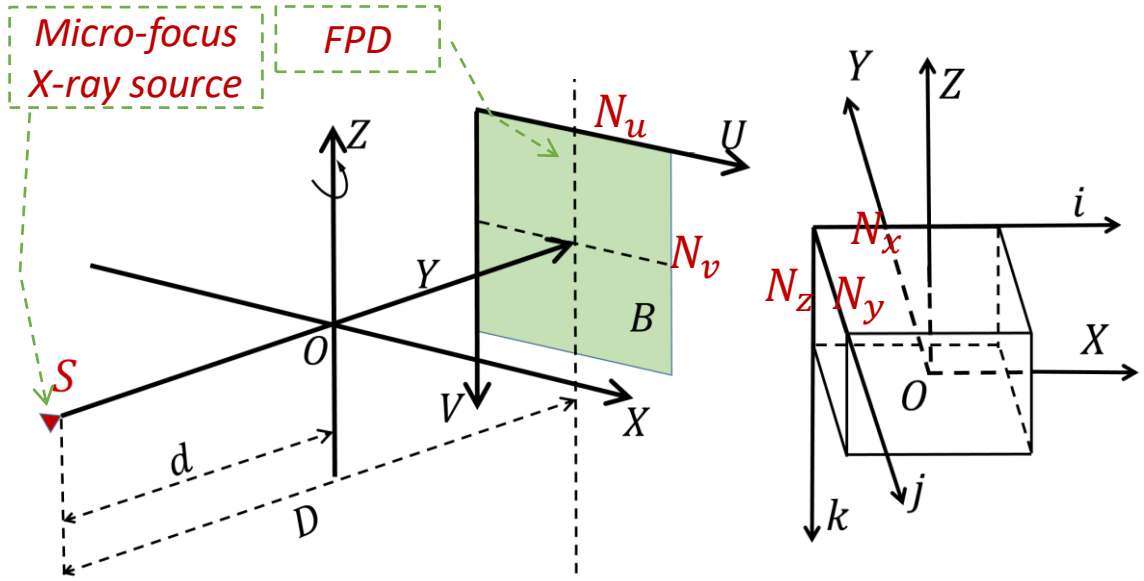
Contributions

1. We proposed a **novel** back-projection algorithm
2. We implemented an efficient CUDA kernel for back-projection
3. We take advantage of the heterogeneity of **ABCI** supercomputer
   - Use CPU for filtering computation
   - Use GPU for back-projection
4. We proposed a framework to generate high-resolution images
   - High performance
   - High scalability
5. Using up to 2,048 V100 GPUs on **ABCI**, the 4K and 8K problems can be solved within 30 seconds and 2 minutes, respectively (including I/O)

2K problem : $2048 \times 2048 \times 4096 \rightarrow 2048^3$
4K problem : $2048 \times 2048 \times 4096 \rightarrow 4096^3$
8K problem : $2048 \times 2048 \times 4096 \rightarrow 8192^3$

Introduction of Compute Tomography

- CT system can generate 3D image from a set of 2D projections (or images)
- Cone Beam Compute Tomography (CBCT)
- CBCT Geometry & Parameter



(a) CBCT geometry and trajectory (b) 3D volume geometry

Param	Description
N_p	the number of 2D projections
N_u, N_v	the width and height of a 2D projection, respectively
N_x, N_y, N_z	the number of voxels in X, Y, Z dimension, respectively

- Reconstruction Problem Definition :

$$N_u \times N_v \times N_p \rightarrow N_x \times N_y \times N_z$$

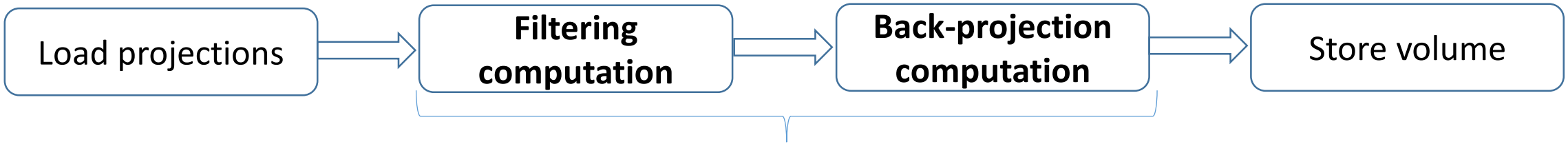
- Performance metrics :

$$GUPS = N_x * N_y * N_z * N_p / T$$

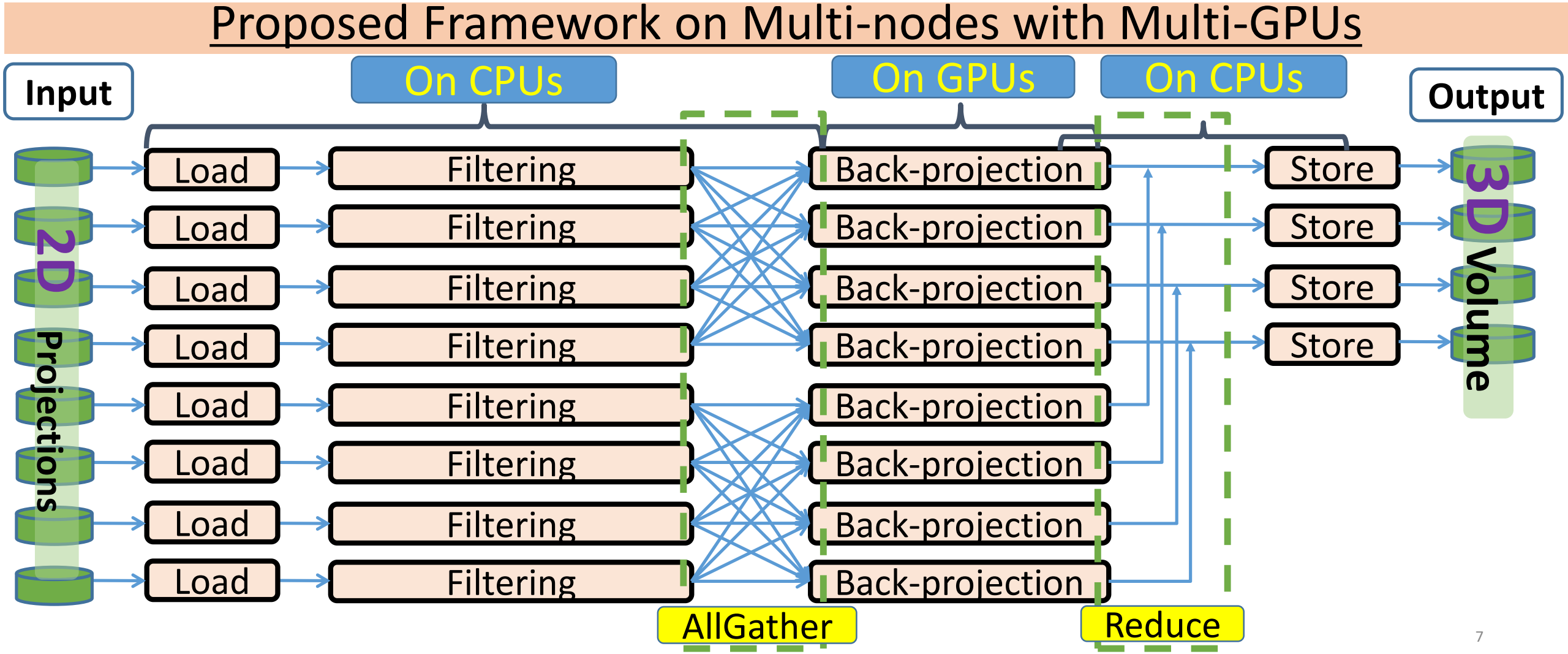
where T is execution time in a unit of second.

FDK algorithm

- Presented by Feldkamp, Davis, and Kress in 1984 (36 years ago)
- FDK is also known as the Filtered Back-projection (**FBP**) algorithm
- FBP method is indispensable in most of the practical CT systems
- Intensive computation for 3D image reconstruction
  - Filtering computation, $O(Log(N)N^2)$
  - Back-projection computation, $O(N^4)$
- FFT primitive is required in Filtering computation
  - Intel IPP, MKL, cuFFT, etc.



Overview of the Proposed iFDK Framework



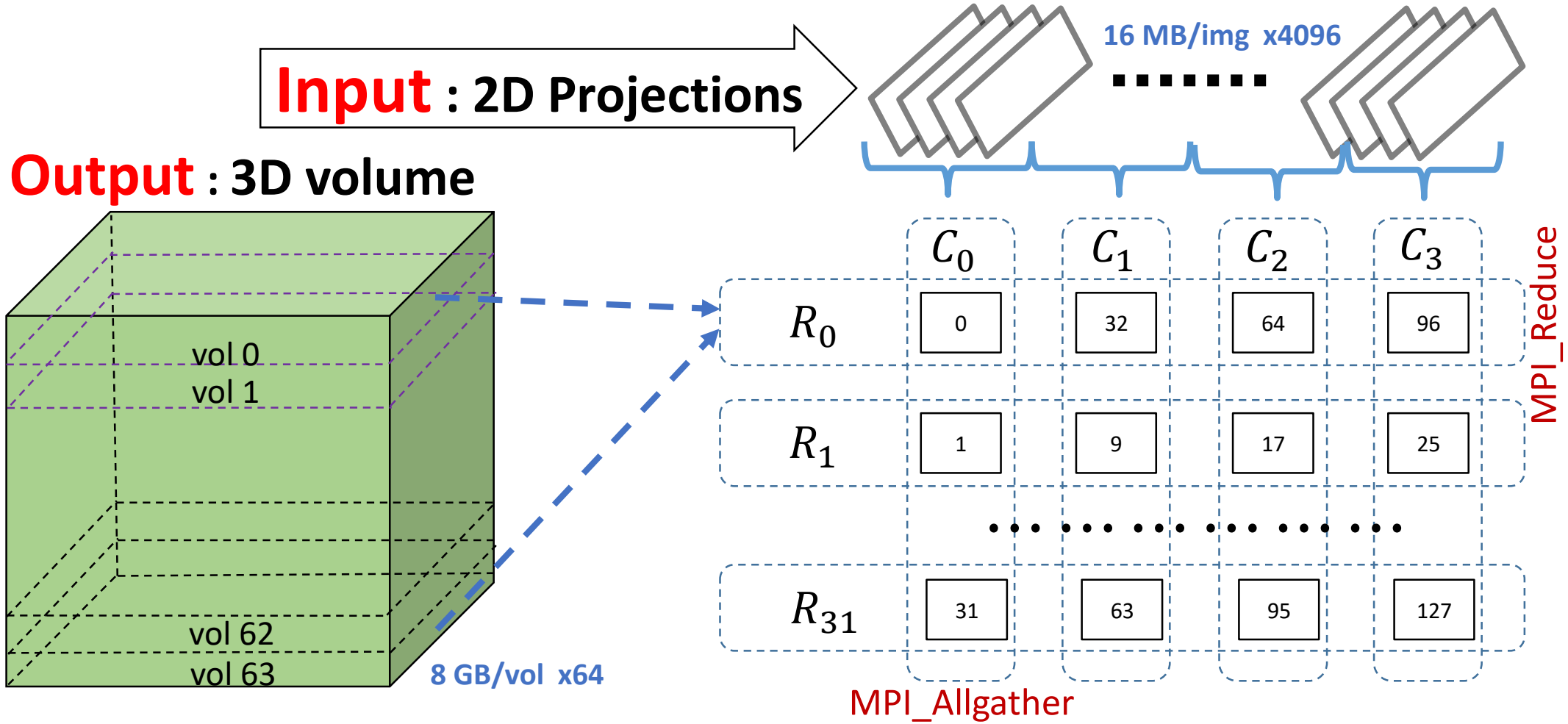
Proposed back-projection kernel on GPU

- We re-organize the loops
- We do not rely on texture cache
  - Use L1/L2 cache directly due to the better data locality
  - The locality is improved by using the transposed projections and volume
- We do not use texture interpolator
  - Achieve high precision of float32
- We compute a batch of 32 projections
  - Benefit to in-register accumulation
  - Reduce the global memory access
- We perform thread communication by shuffle intrinsic
  - Simple and efficient

Detailed CUDA kernel can be found in our paper

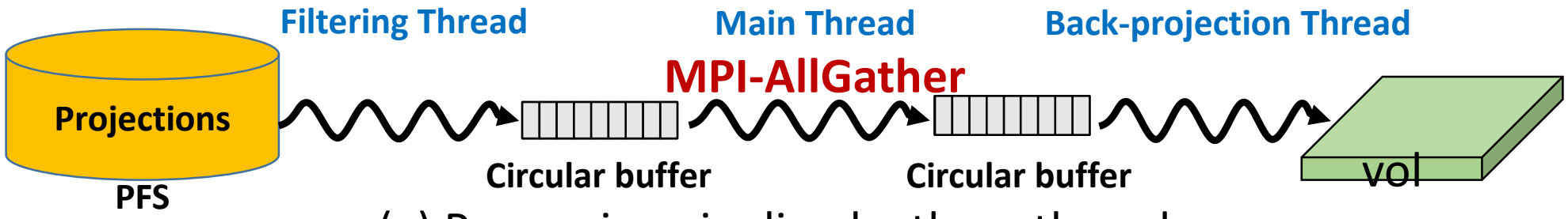
An example of Problem Decomposition Scheme

Use 128 GPUs (32 Nodes) to slove a 2K problem
Input: 4k count of 2k^2 image, Output: 4k^3

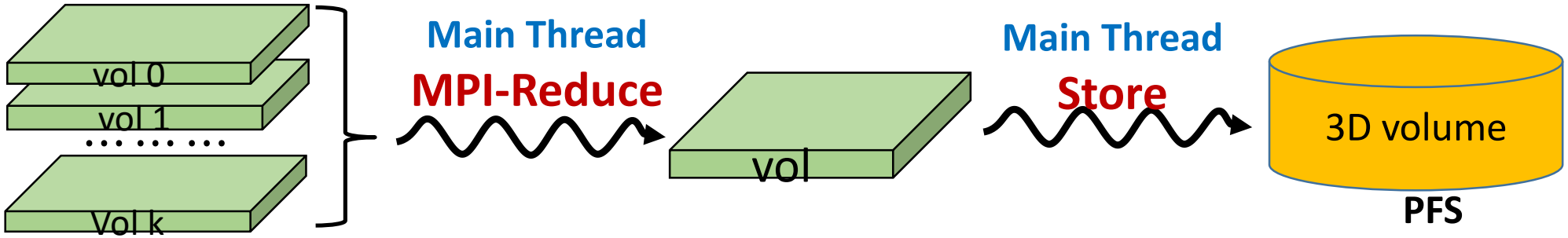


Orchestration and Overlapping in iFDK

- Each MPI rank launches two extra-threads by pthread library
- Filtering thread launches multiple OpenMP threads for filtering computation



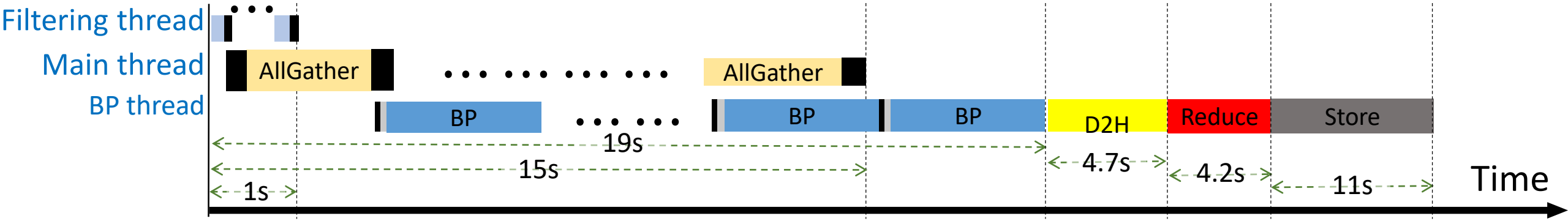
(a) Processing pipeline by three threads



(b) Reduce and Store operations by Main Thread

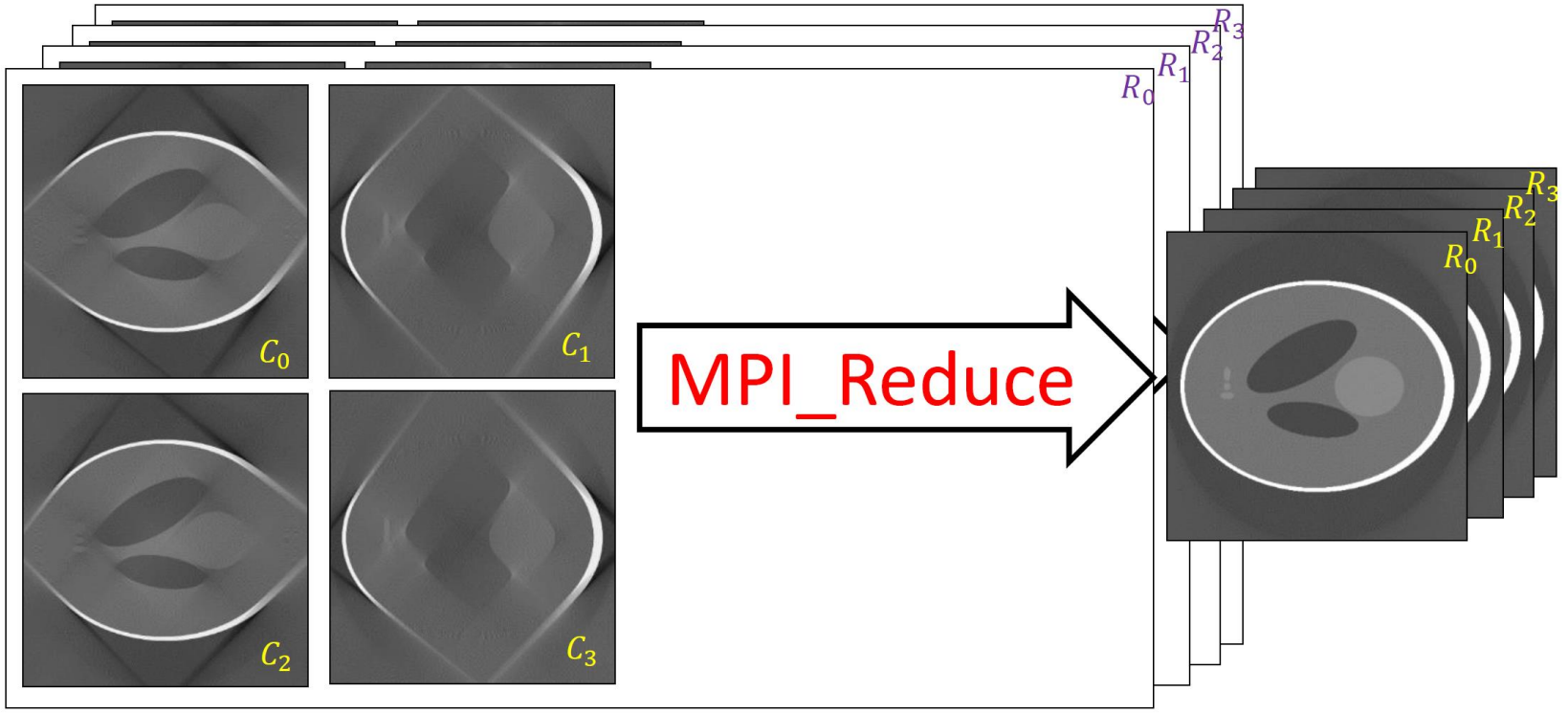
An example of achieved overlapping

- An example of pipeline to solve $2048 \times 2048 \times 4096 \rightarrow 4096^3$ problem
- Use 128 V100 GPUs
- Filtering thread processes 32 projections
- Main thread gathers 1024 projections
- Back-projection thread processes 1024 projections



An example of MPI_Reduce operation

- Use 16 GPUs to solve $2048 \times 2048 \times 4096 \rightarrow 2048^3$ problem
- Each GPU process a sub-volume of size 8GB



Performance model

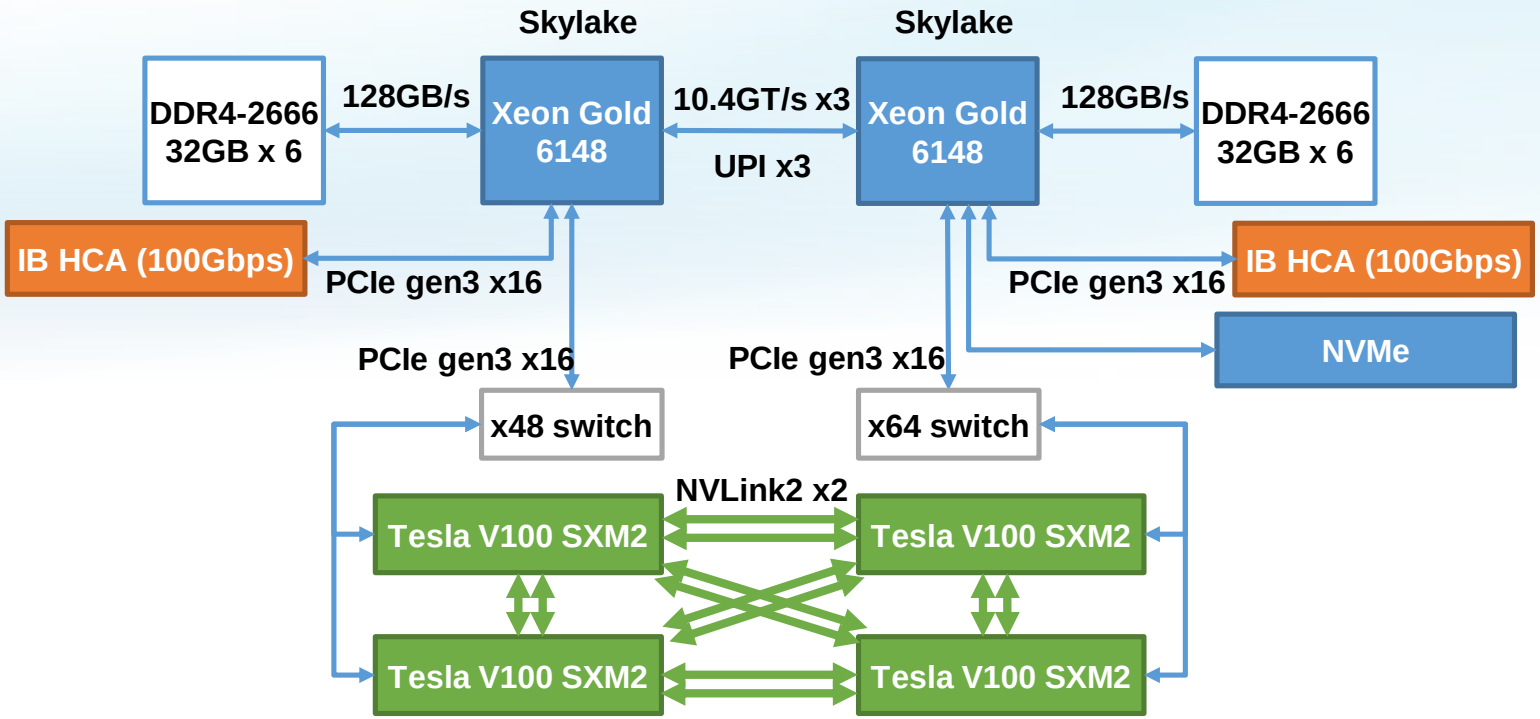
- Micro-benchmarking
  - To better understand the characteristics of our system
  - Measure the constant parameters of our system, e.g.
    - Bandwidth of Parallel File System (PFS)
    - Bandwidth of PCIe connector
    - Throughput of MPI primitives
- Building a performance model
  - We can predict the potential peak performance
  - We can justify the scalability of iFDK
    - iFDK scales with the number of GPUs (N_{gpus}) linearly
- **Detailed equations can be found in our paper**

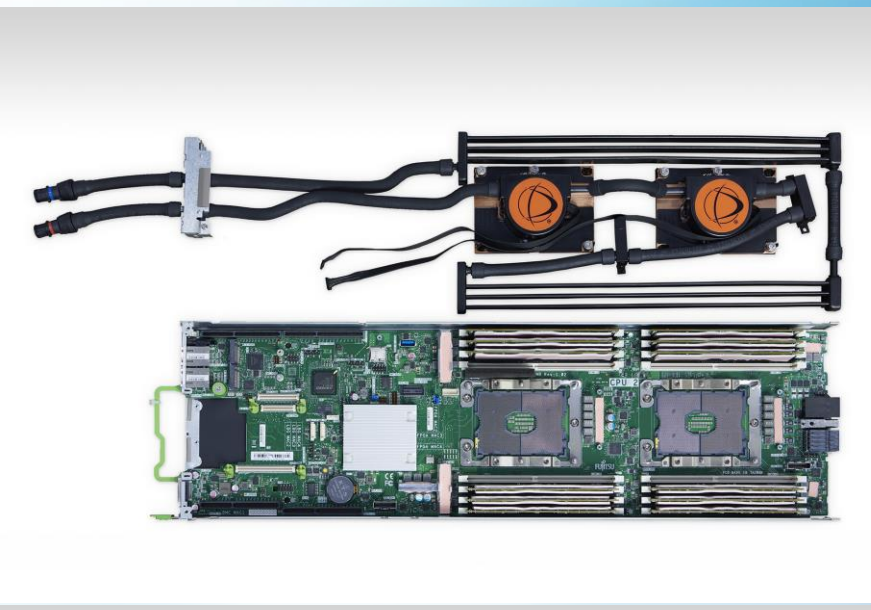
Evaluation environment

- ABCI supercomputer
  - Constructed and operated by AIST
  - 1,088 computing nodes, 4,352 Tesla V100 GPUs
- Software
  - CentOS 7.4
  - CUDA 9.0
  - Intel library 2018.2.199 (MPI, IPP)
  - RTK (Reconstruction Toolkit) 1.4.0
- Evaluation dataset
  - **Computation complexity is independent of the content of projections**
  - Use Shepp-Logan phantom
  - Generate projections by RTK library

ABCI Compute Node

FUJITSU PRIMERGY Server (2 servers in 2U)	
CPU	Xeon Gold 6148 (27.5M Cache, 2.40 GHz, 20 Core) x2
GPU	NVIDIA Tesla V100 (SXM2) x4
Memory	384GiB DDR4 2666MHz RDIMM
Local Storage	1.6TB NVMe SSD (Intel SSD DC P4600 u.2) x1
Interconnect	InfiniBand EDR x2





Evaluation on back-projection kernels

- A single Tesla V100 GPUs
- **Single precision**
- Up to **1.6 times faster** than baseline
- Performance summary
  - Better data locality
  - Advantaged use of L1 cache
  - Efficient data communication

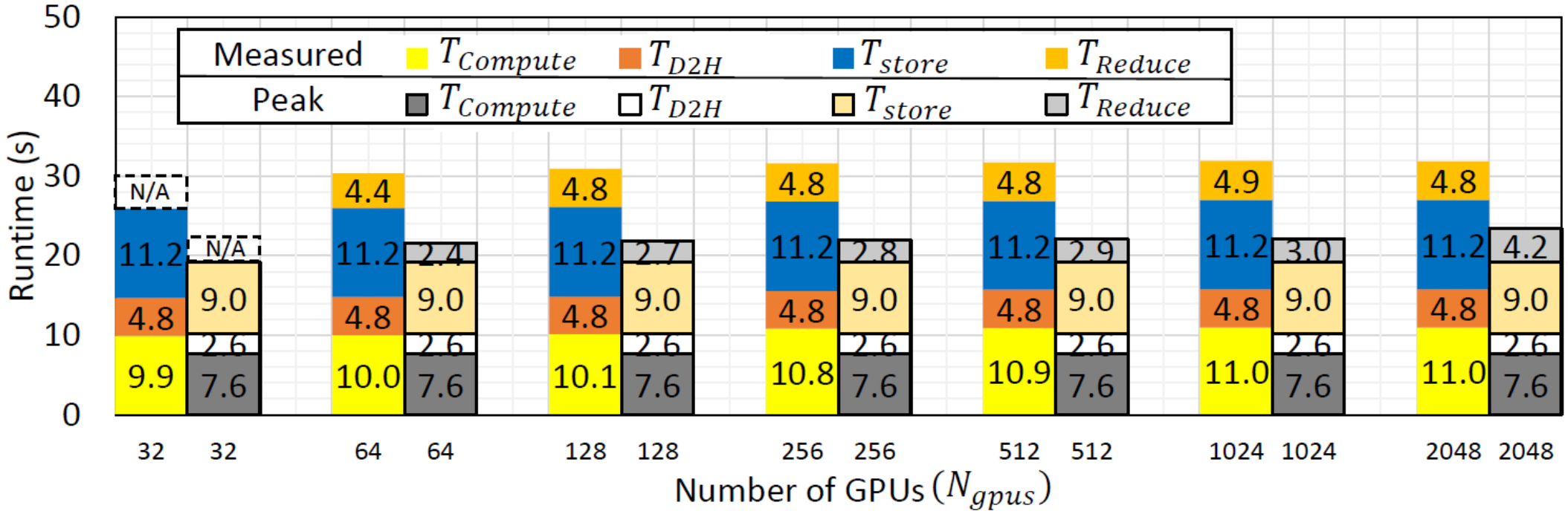
RTK

FDK poblems $(pixel \rightarrow voxel)$	Baseline (GUPS)	Ours (GUPS)
$512^2 \times 1k \rightarrow 128^3$	65.3	**118.0**
$512^2 \times 1k \rightarrow 256^3$	107.4	**188.6**
$512^2 \times 1k \rightarrow 512^3$	115.1	**206.0**
$512^2 \times 1k \rightarrow (1k)^3$	118.1	**211.4**
$512^2 \times 1k \rightarrow (1k)^2 \times 2k$	N/A	**212.7**
$(1k)^3 \rightarrow 128^3$	41.9	**27.2**
$(1k)^3 \rightarrow 256^3$	77.4	**83.7**
$(1k)^3 \rightarrow 512^3$	115.7	**190.3**
$(1k)^3 \rightarrow (1k)^3$	117.9	**205.7**
$(1k)^3 \rightarrow (1k)^2 \times 2k$	N/A	**207.9**
$(2k)^2 \times 1k \rightarrow 128^3$	16.1	**7.7**
$(2k)^2 \times 1k \rightarrow 256^3$	38.6	**24.1**
$(2k)^2 \times 1k \rightarrow 512^3$	80.2	**81.6**
$(2k)^2 \times 1k \rightarrow (1k)^3$	116.9	**186.9**
$(2k)^2 \times 1k \rightarrow (1k)^2 \times 2k$	N/A	**198.7**

The higher, the better

Weak scalability evaluation

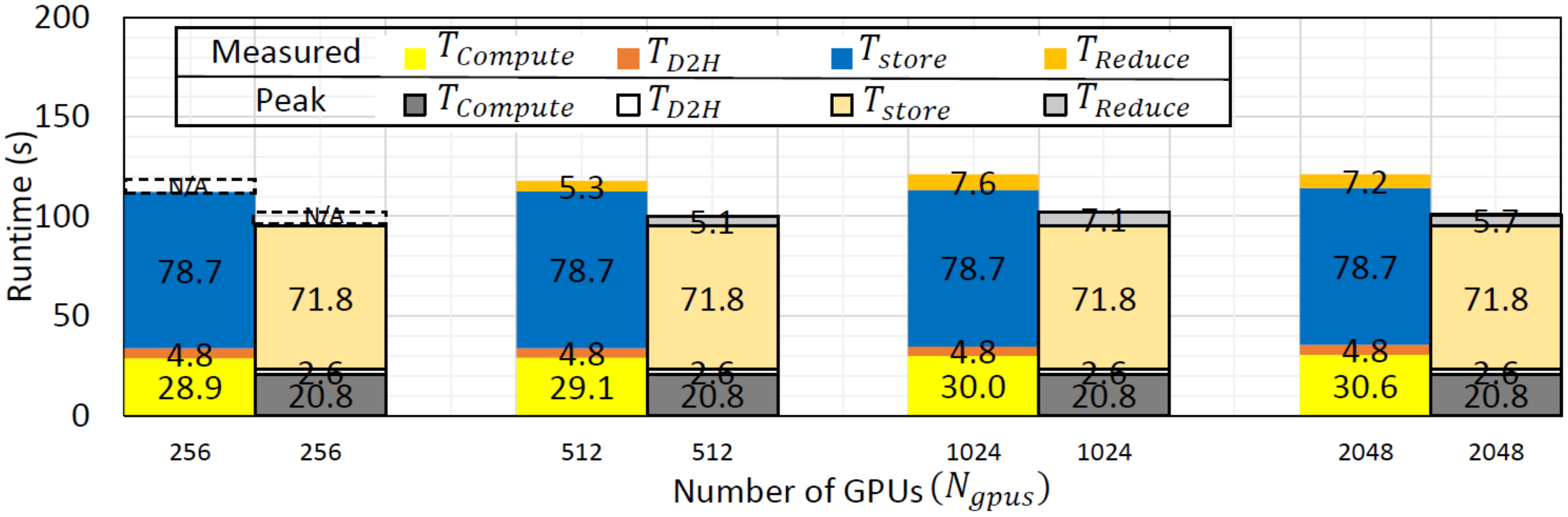
- Using up to 2048 GPUs to evaluate a **4K problem**
- Peak performance is predicted by our performance model
- We achieve outstanding weak scaling



Weak scaling for $2048^2 \times N_p \rightarrow 4096^3$. $N_p = 16 * N_{gpu}$

Weak scalability evaluation

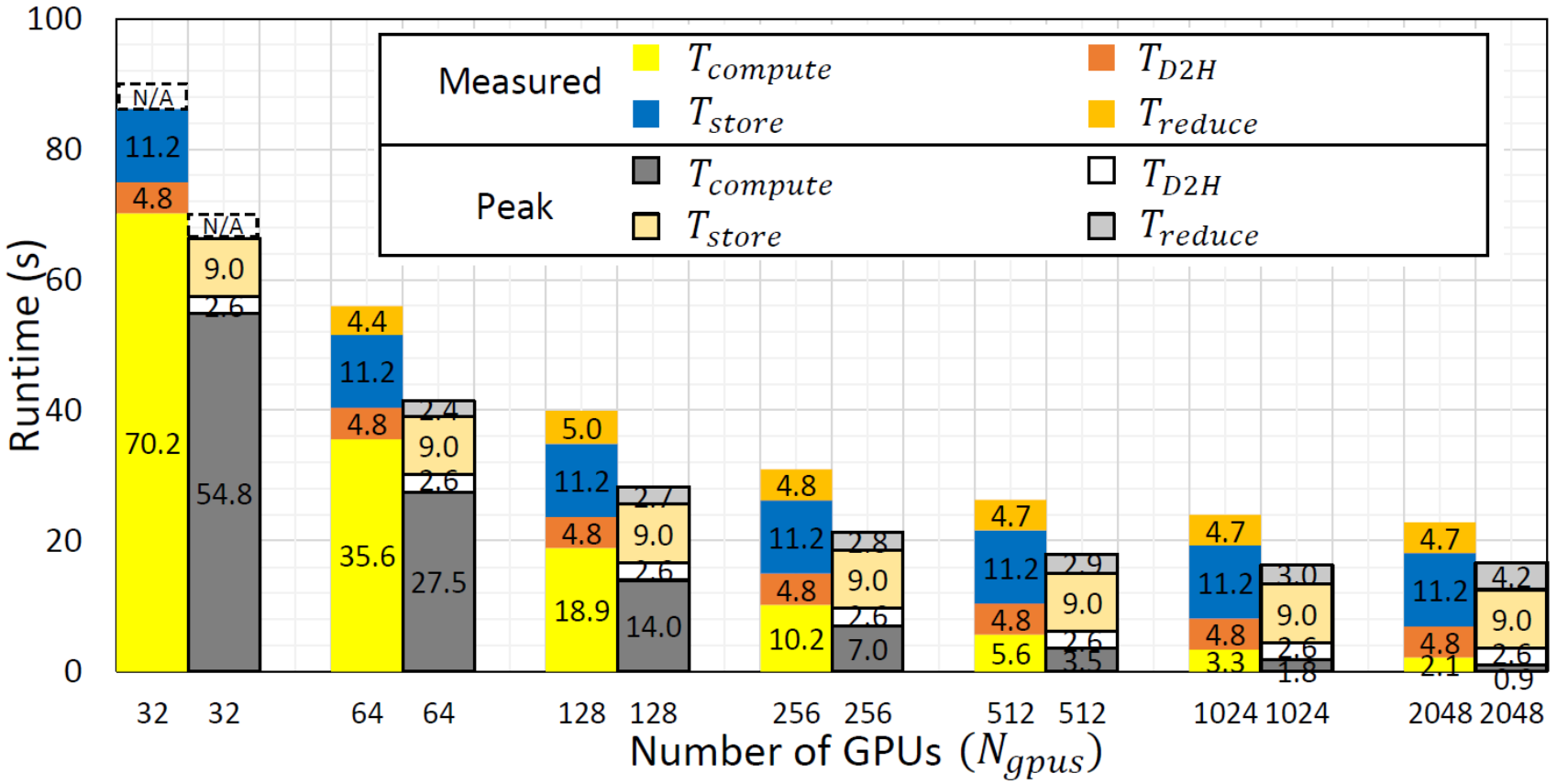
- Using up to 2048 GPUs to evaluate a **8K problem**
- Peak performance is predicted by our performance model
- We achieve outstanding weak scaling



Weak scaling for $2048^2 \times N_p \rightarrow 8192^3$. $N_p = 4^* N_{gpus}$

Strong scalability evaluation

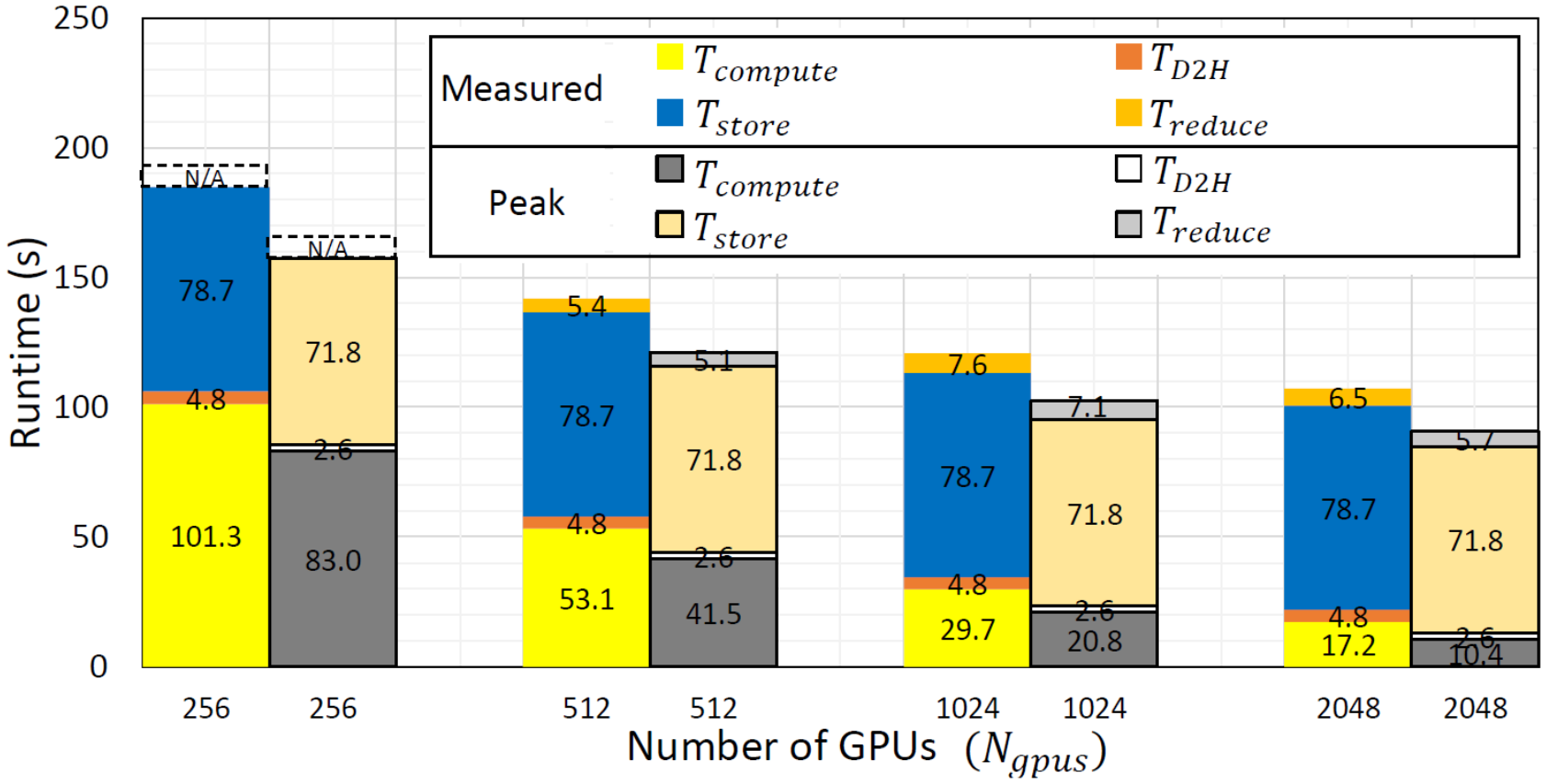
- Using up to 2048 GPUs to solve a **4K problem**
- Peak performance is predicted by our performance model
- We can achieve about 76% of the peak performance



Strong scaling for $2048^2 \times 4096 \rightarrow 4096^3$.

Strong scalability evaluation

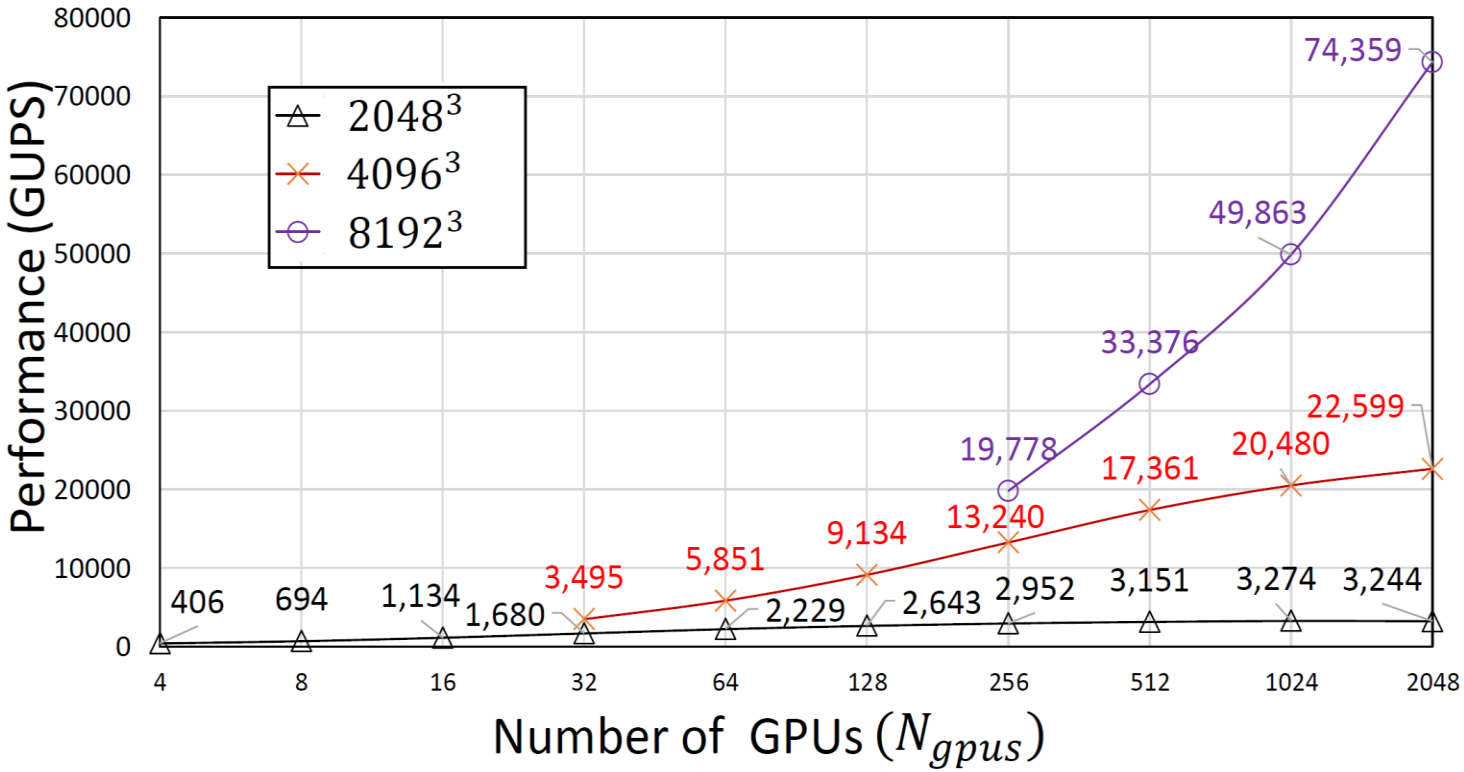
- Using up to 2048 GPUs to solve a **8K problem**
- Peak performance is predicted by our performance model
- We can achieve about 76% of the peak performance



Strong scaling for $2048^2 \times 4096 \rightarrow 8192^3$.

Performance

- Extremely high performance
- Higher computational intensity, better scalability
- Bottleneck becomes the data movement
- Over **two order of magnitude** faster than a single Tesla V100 GPU
- **Solve any FDK problems instantaneously**



The achieved performance of solving 2K, 4K, and 8K problems

Conclusion

1. We proposed a general FDK algorithm
2. We implemented an efficient CUDA kernel for back-projection
3. We proposed a framework (iFDK) to generate high-resolution image
   - Two characteristics:
     - Pipeline processing
     - Parallel computation
   - Take advantage of the heterogeneity of **ABCI** supercomputer
     - Use CPU for filtering computation
     - Use GPU for back-projection
   - Almost ideal Strong and weak scaling
4. On **ABCI**, using up to 2,048 V100 GPUs to solve a 4K and 8K problems within 30 seconds and 2 minutes, respectively (including I/O).

ABCI is not only specified for AI, but also general for other HPC applications!

Future work

- Research on rendering High-resolution image on ABCI
- Research on compressing the High-resolution images
- Provide an image reconstruction service via cloud
- Challenge our system using full-nodes (namely 4K GPUs) again