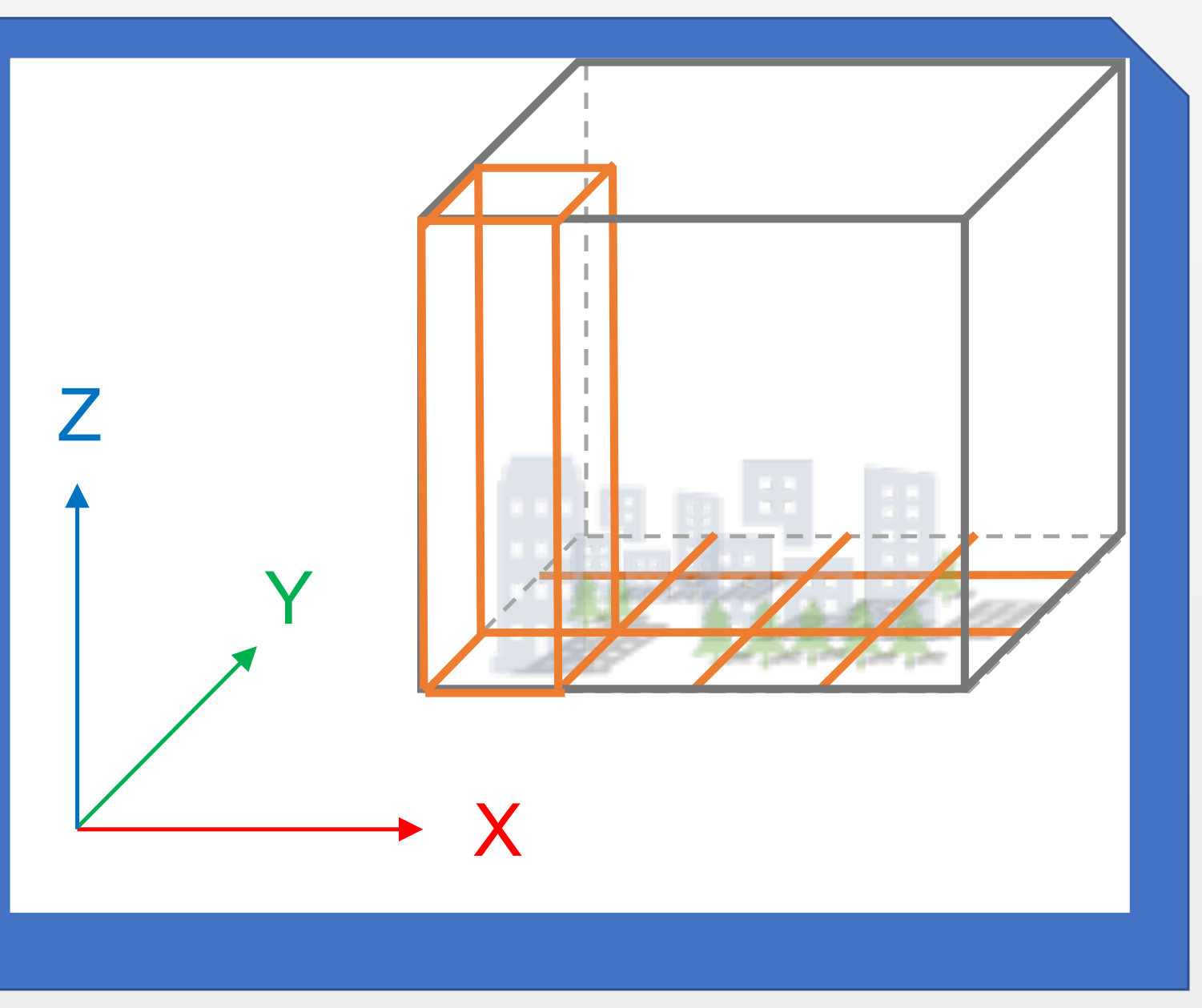
Implementation and Performance Evaluation of Parallel OpenACC Climate Code City-LES on GPU Cluster

Daisuke Tsuji [1], Hiroto Tadano [3], Taisuke Boku [3], Ryosaku Ikeda [3,*], Takuto Sato [2], Hiroyuki Kusaka [3]

1) Graduate School of Systems and Information Engineering, University of Tsukuba, Ibaraki, Japan.
2) Graduate School of Life and Environmental Sciences, University of Tsukuba, Ibaraki, Japan.
3) Center for Computational Sciences, University of Tsukuba, Ibaraki, Japan.
*) Present affiliation is Weathernews Inc.

City-LES

- City-LES is climate simulation code based on Large Eddy Simulation
- It can simulate urban area with buildings, street-side tree plantations and dry-misters
- Has been developing by Center for Computational Science (CCS), University of Tsukuba
- 2D domain decomposition by MPI on X-Y plane
- Applied OpenMP to Z dimension



OpenACC Acceleration

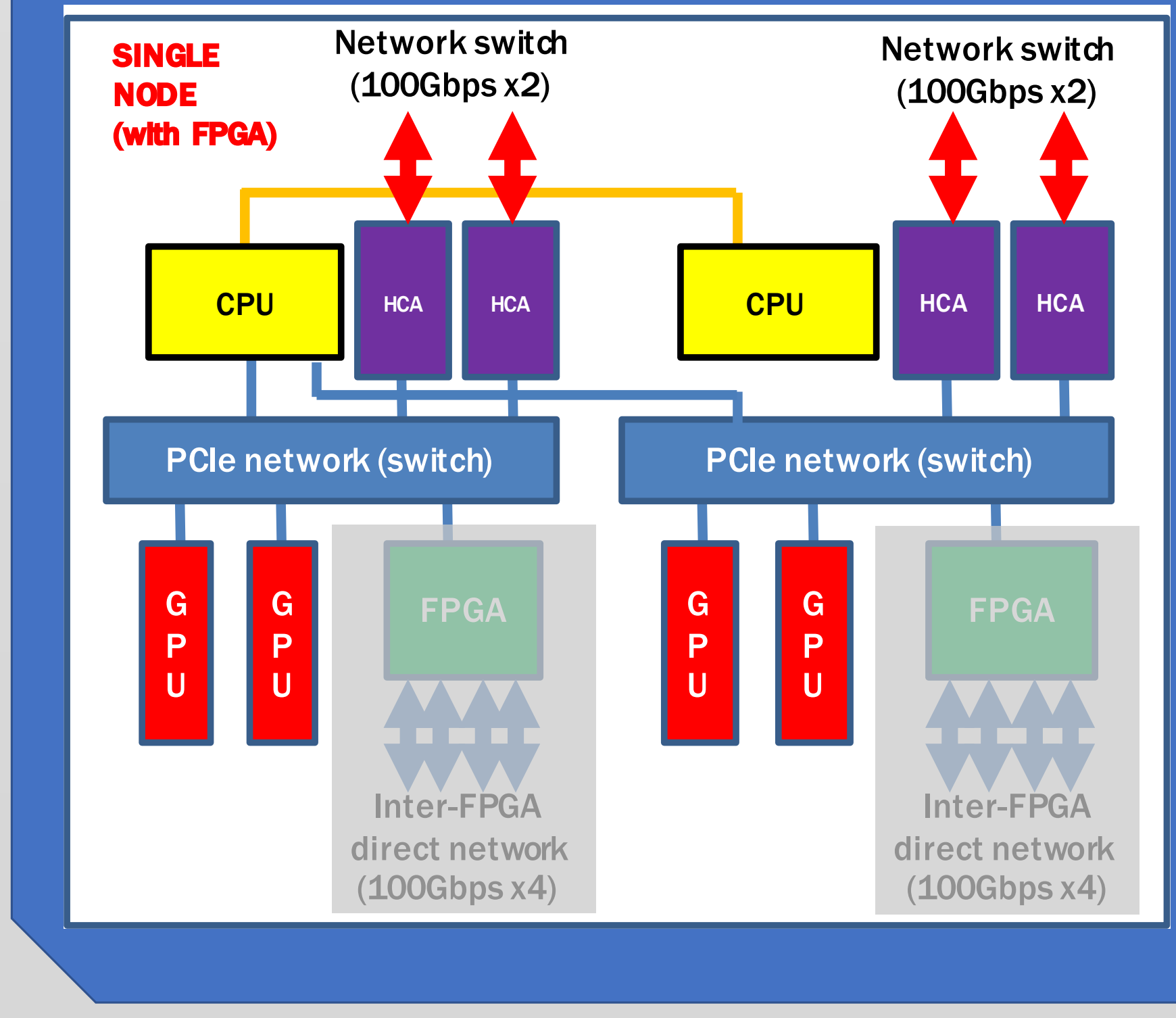
- At first, we applied CUDA Fortran to City-LES
- It is required certain amount of data movement between CPUs and GPUs when a partial code of City-LES is ported to CUDA Fortran because some parts are still executed on GPUs
- It is need to implement entire computation parts of City-LES on GPUs completely for removing data movements between CPUs and GPUs
- However a small computation part is not cost-effective to implement on GPU
- So we decided to apply OpenACC for them to minimize the coding complexity
- We have developed a GPU-ready version of City-LES removed data movement as much as possible

```
subroutine parent()
  implicit none
  . . .

  !– Small subroutines be run on
  !– GPUs to remove data movements
  call child_A()
  call chile_B()
  call child_C()

  !– Loops be wanted to accelerate
  do k = 1, ksize
    do j = 1, jsize
      do i = 1, isize
        . . .
      enddo
    enddo
  enddo
end subroutine parent
```

Experiment Environment

- Cygnus Hybrid Cluster at CCS for experiment
- 2 Intel Xeon CPUs, 4 NVIDIA V100 GPUs and 2 additional Intel Stratx10 FPGAs
- One IB HCA per GPU
- Deneb : 48 CPU + GPU nodes
- Albireo : 32 CPU + GPU + FPGA nodes
- Using only GPUs and CPUs on this implementation (no FPGA)
- Using GPUDirect for MPI communications on GPUs
- Up to 32 nodes for scaling performance experiment

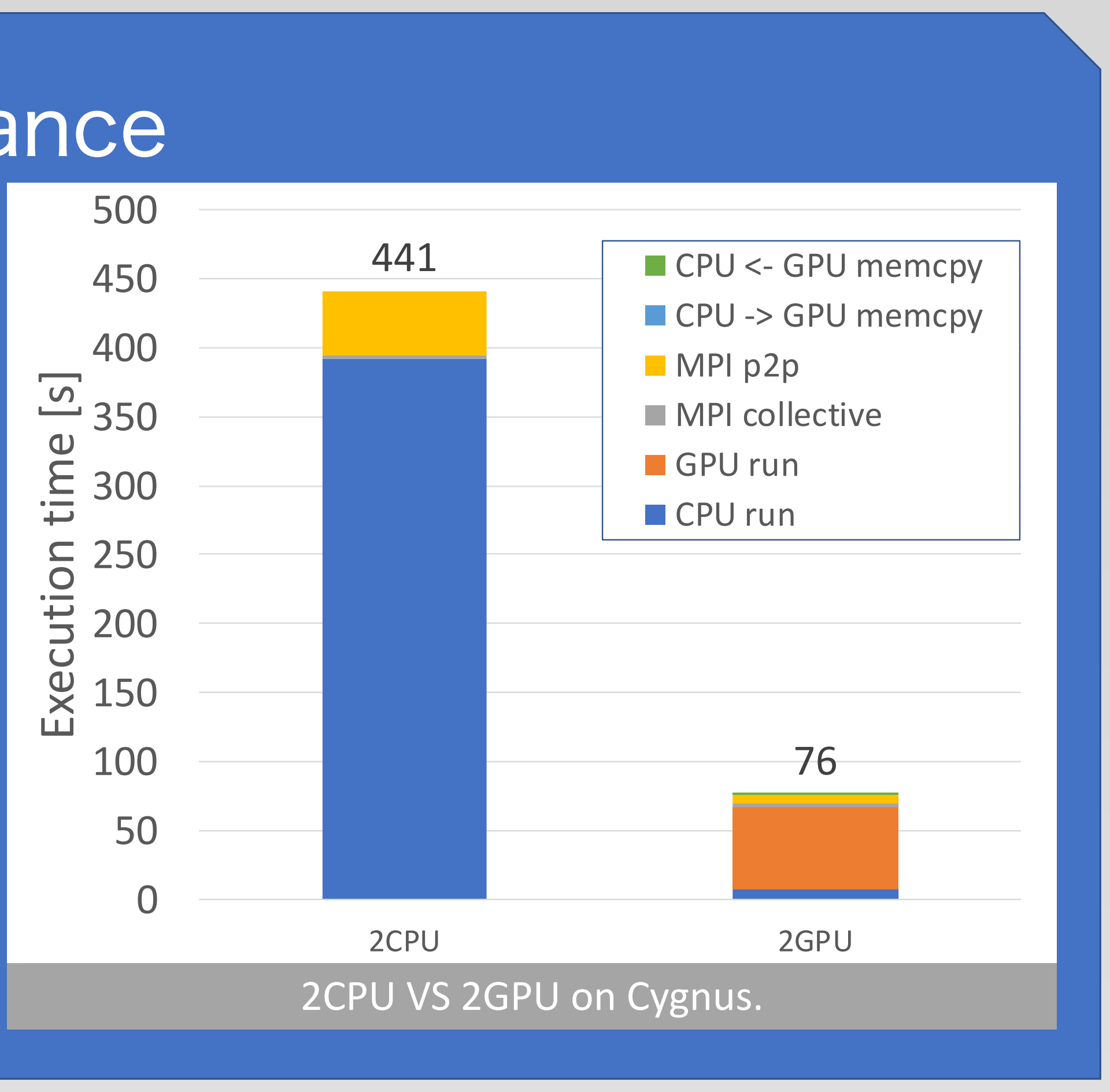
Source : CCS. https://www.ccs.tsukuba.ac.jp/press_cygnus_20190326/


SINGLE NODE (with FPGA)
Network switch (100Gbps x2)
Network switch (100Gbps x2)
CPU
HCA
HCA
CPU
HCA
HCA
PCIe network (switch)
PCIe network (switch)
GPU
GPU
FPGA
GPU
GPU
FPGA
Inter-FPGA direct network (100Gbps x4)
Inter-FPGA direct network (100Gbps x4)

CPU	Intel Xeon Gold 6126
CPU Memory	DDR4 192GB (96 GB / CPU)
GPU	NVIDIA Tesla V100 (PCIe)
InfiniBand	Mellanox ConnectX-6 HDR100
Inter-node Network	100 Gbps x 4 x 80 = 4 GB/s (full bisection bandwidth)
Host OS	CentOS 7.6
Host Compiler	PGI Compiler 19.1
CUDA Version	CUDA 10.1
Problem Size	256 x 256 x 128 / process and 200 time-steps

Single node Performance

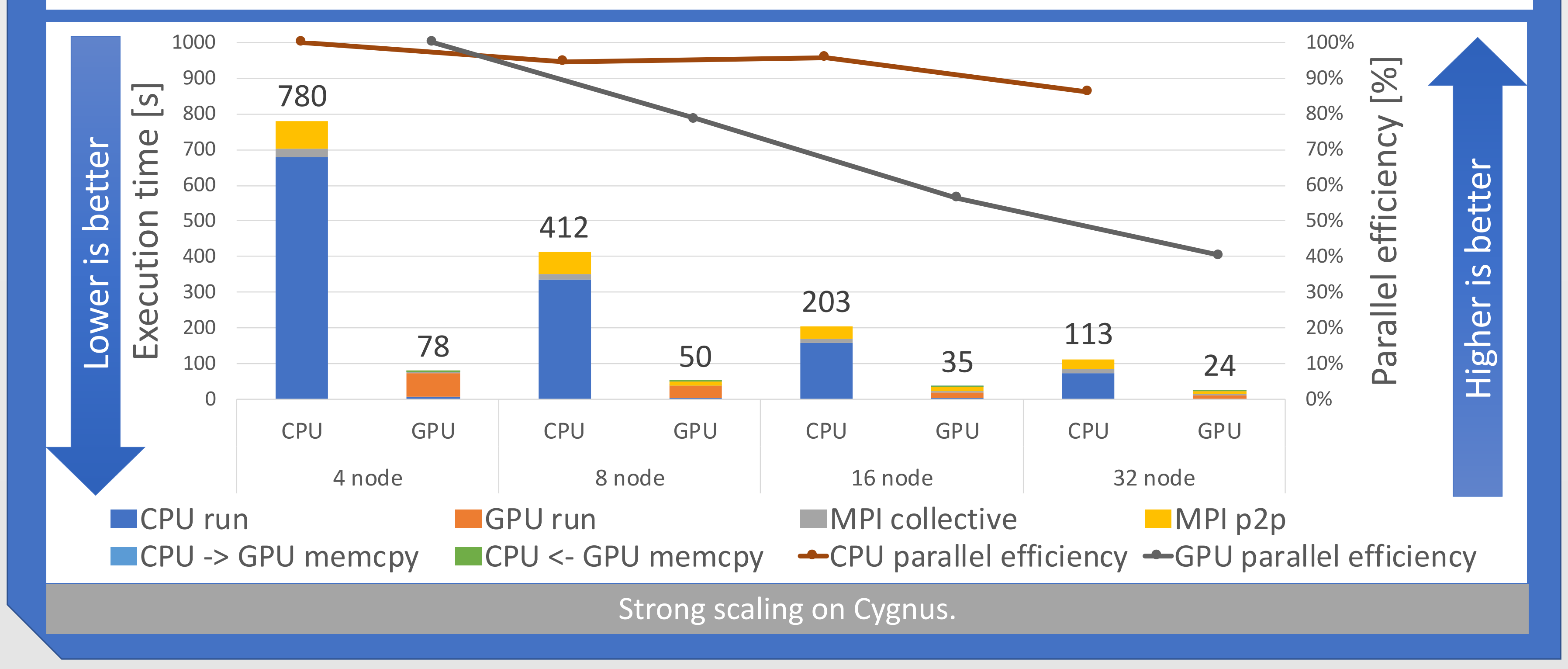
- Using 2 CPUs and 2 GPUs on single node with 2 MPI processes
- All GPU code is written in OpenACC
- Execution time with breakdown for CPU and GPU running time, data movement between CPU and GPU, and p2p and collective communications on MPI
- GPU accelerates the performance up to 5.79x to CPU-only case
- Data movement time is minimized by implementing all computation on GPUs



2CPU VS 2GPU on Cygnus.

Parallel Execution (Strong Scaling)

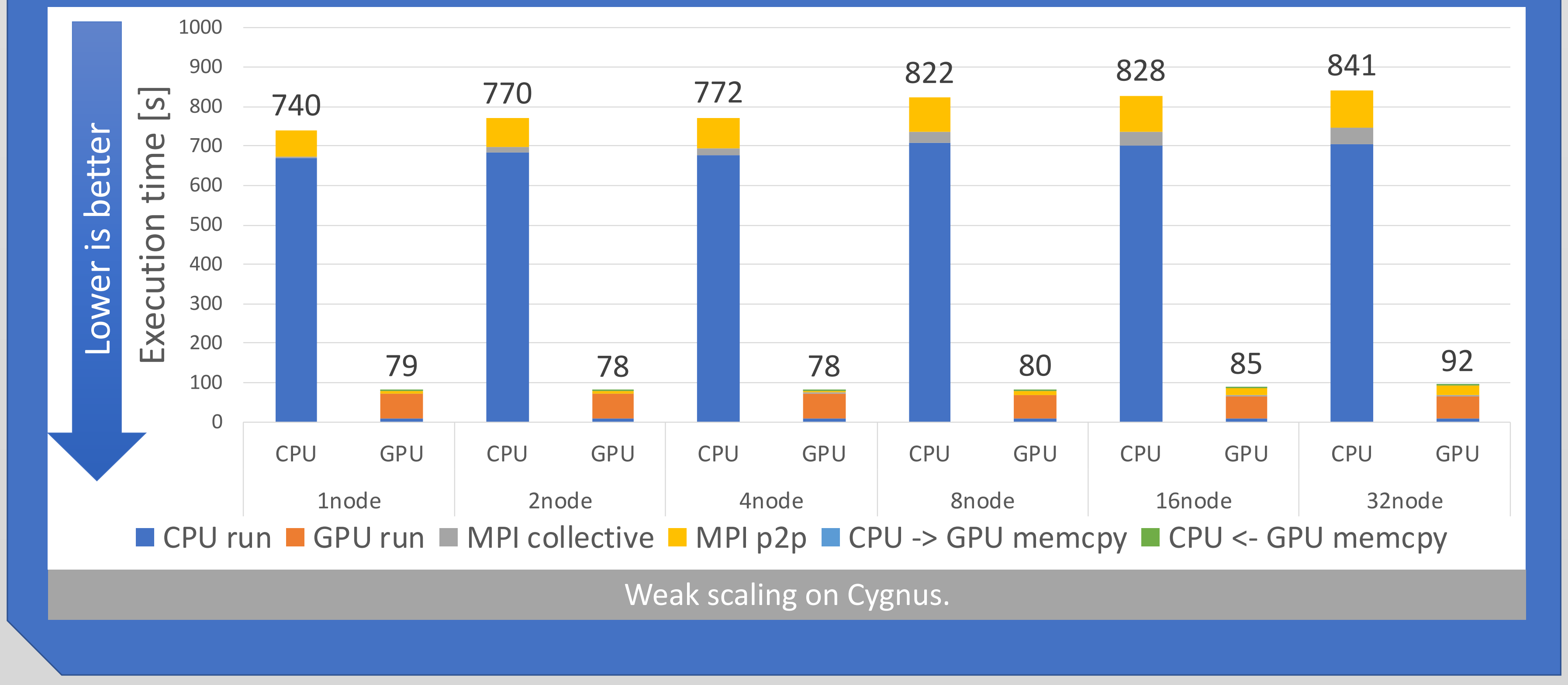
- Strong scaling of CPUs and GPUs from 4 nodes (16 GPUs) to 32 nodes (128 GPUs)
- 4 MPI processes run on each node and each process runs 6 OpenMP threads
- CPU parallel efficiency with 32 nodes is 86 % of 4 nodes, while GPU parallel efficiency is 40 %
- GPU running time with 16 nodes is 18.1 [s] and speed up to 3.62x from 4 nodes 70.0 [s] (efficiency is 91 %), however it is 5.77x with 32 nodes 11.0 [s](72 %)
- MPI p2p communication time with 32 nodes is 10.3 [s] (43 % of total), while the time of 4 node is 10 % of total one
- As increasing scale, problem size of each GPU is too small to accelerate running on GPUs and pack-unpacking cost of halo communications becomes heavy due to stride access, therefore the parallel efficiency becomes worse
- In other hand, data movement time between CPUs and GPUs is decreased in reverse-proportionally at scaling
- Conclusion: target problem size limits the strong scaling performance up to 4x from 4 nodes case



Strong scaling on Cygnus.

Parallel Execution (Weak Scaling)

- Weak scaling of CPUs and GPUs from single node (4 GPUs) to 32 nodes (128 GPUs)
- 4 MPI processes run in each node and each process runs 6 OpenMP threads
- GPU City-LES execution time increase is limited up to 1.16x on 32 nodes from single node and the efficiency is kept to 86 % which is almost the same efficiency compared with CPU-only implementation
- MPI communication time increase at scaling, however GPU communication time is negligible thanks to high performance IB HCA with 100 Gbps for each GPU on node (4 IB HCA for 4 V100 GPUs)
- Data movement time between CPUs and GPUs is almost the same in any scale
- Computation time on CPUs and GPUs are also almost the same in each case
- Conclusion: Our implementation of GPU-ready City-LES has very high performance on weak scaling



Weak scaling on Cygnus.

Conclusion

- We apply OpenACC Fortran to City-LES for accelerating the computation performance
- OpenACC easily leads full porting of computation part from CPU to GPU even though with low parallelism resulting to great reduction of CPU-GPU data movement cost
- GPU parallel efficiency is less than that of CPU due to halo communications, however GPU computation time keeps 72 % of parallel efficiency up to 128 GPUs
- In weak scaling, GPU has good efficiency due to GPUDirect communication through network cards for each GPU

Future Work

- We do not enable heavy functions such as computing buildings yet in this research in order to make the problem easier, so we will apply OpenACC to these parts and evaluate performance in the case of enabled
- We will evaluate CPU and GPU performance in real problem simulation
- In real problem case, more GPUs will be needed due to small memory capacity of GPU, so we need to experiment bigger scale and evaluate the performance
- More optimization may be needed to OpenACC instead of simple OpenACC directives for more performance optimization

ACKNOWLEGEMENT This research is partially supported by " Communication-Computation Unified Supercomputing" project under MEXT' s "Next Generation Supercomputer R&D" program, Multi-Disciplinary Collaboration Research Program of CCS, University of Tsukuba, and Collaborative Research between CCS and Weathernews Inc.