Accelerated Data Analytics and Computing Workshop 7 – ORNL, TN
Mar. 25, 2019

Application Enhanced by AI and Transprecision Computing: Finite Element Earthquake City Simulation

Kohei Fujita, Takuma Yamaguchi

The University of Tokyo

THE UNIVERSITY OF TOKYO

A Fast Scalable Implicit Solver for Nonlinear Time-Evolution Earthquake City Problem on Low-Ordered Unstructured Finite Elements with Artificial Intelligence and Transprecision Computing

Tsuyoshi Ichimura[1,2,3], Kohei Fujita[1,3], Takuma Yamaguchi[1], Akira Naruse[4],
Jack C. Wells[5], Thomas C. Schulthess[6], Tjerk P. Straatsma[5], Christopher J. Zimmer[5],
Maxime Martinasso[6], Kengo Nakajima[7,3], Muneo Hori[1,3], Lalith Maddegedara[1,3]

[1]Earthquake Research Institute & Department of Civil Engineering, The University of Tokyo
[2]Center for Advanced Intelligence Project, RIKEN, [3]Center for Computational Science, RIKEN
[4]NVIDIA Corporation, [5]Oak Ridge National Laboratory
[6]Swiss National Supercomputing Centre, [7]Information Technology Center, The University of Tokyo

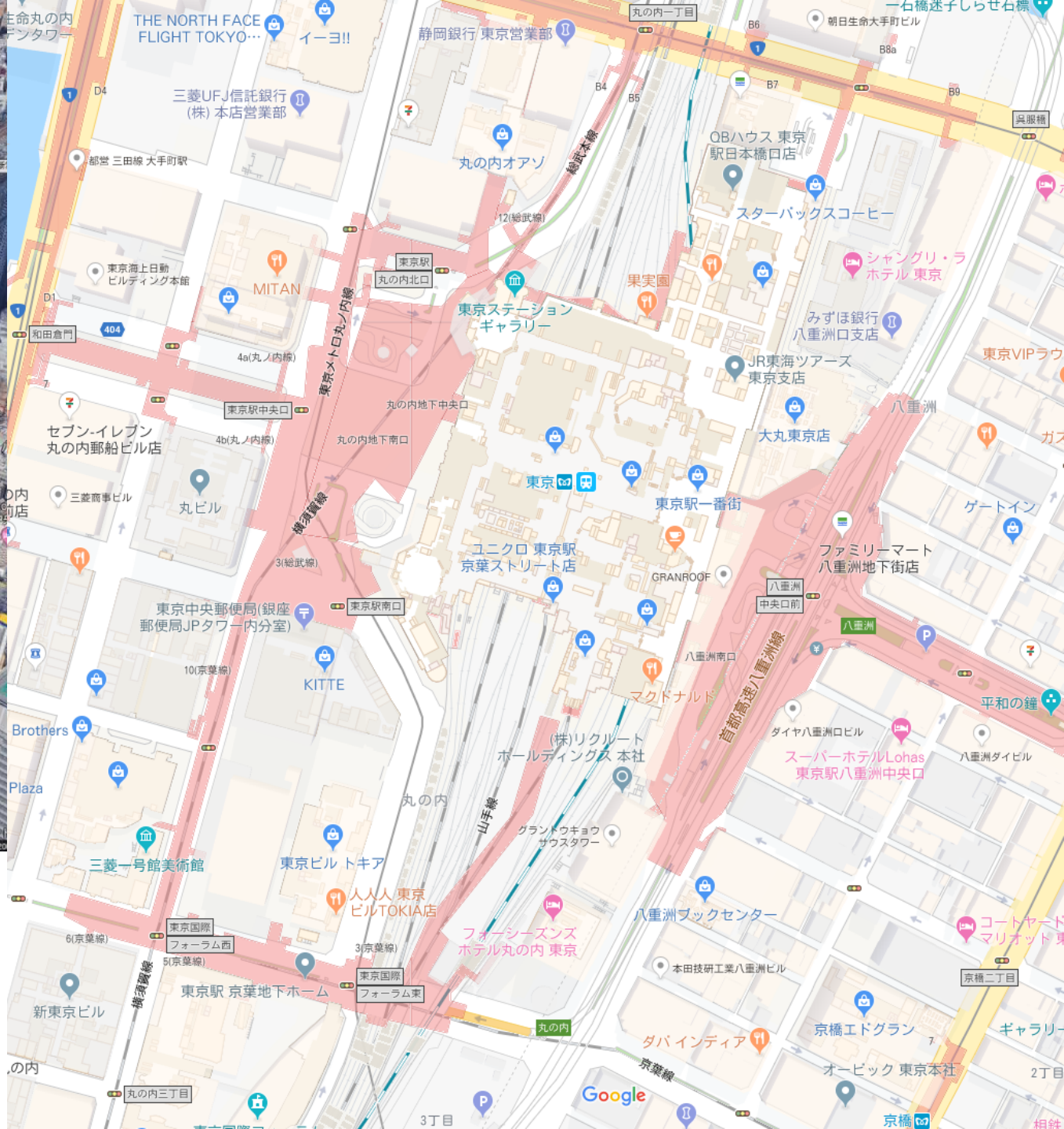
***Abstract*—To address problems that occur due to earthquake in urban areas, we propose a method that utilizes artificial intelligence (AI) and transprecision computing to accelerate a nonlinear dynamic low-order unstructured finite-element solver,**

straightforward to attain high performance, is overwhelming physics-based simulation and sweeping over the supercomputing field, even as if high peak performance is taken for granted. However, in the first place, reasonable development

Smart cities

- Controlling cities based on real-time data for higher efficiency
- Computer modeling via high-performance computing is expected as key enabling tool
- Disaster resiliency is requirement; however, not established yet

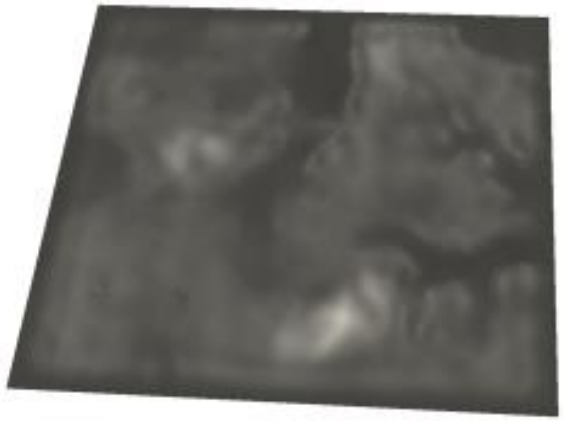
Example of highly dense city: Tokyo Station district



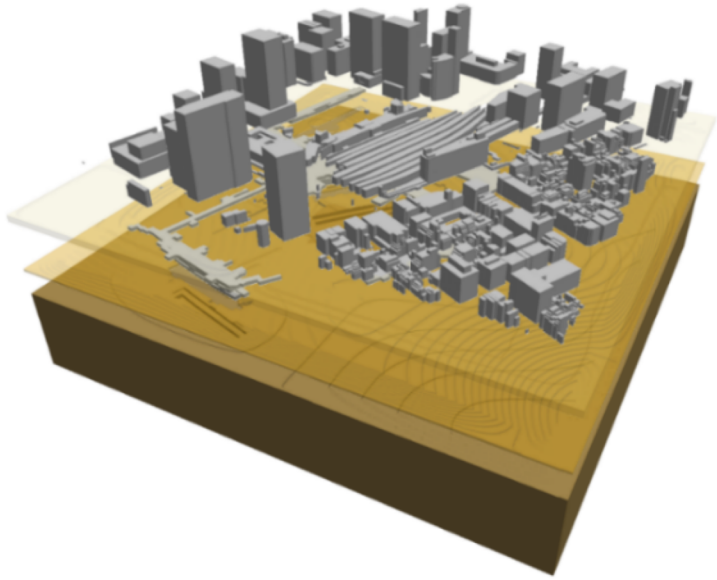
Fully coupled aboveground/underground earthquake simulation required for resilient smart city

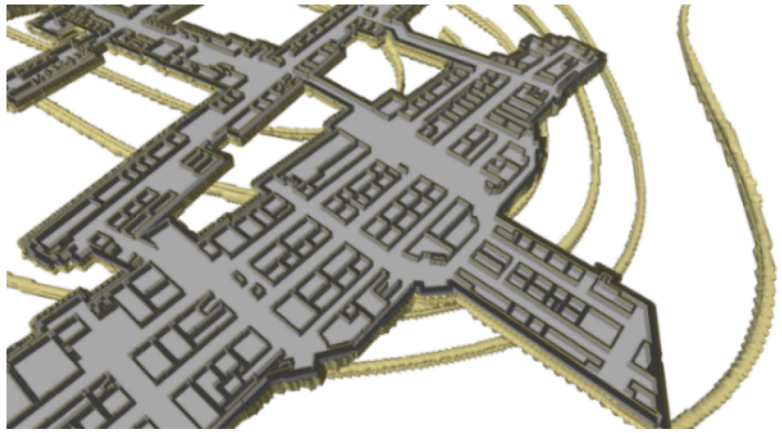
Earthquake modeling of smart cities

- Unstructured mesh with implicit solvers required for urban earthquake modeling
  - We have been developing high-performance implicit unstructured finite-element solvers (SC14 & SC15 Gordon Bell Prize Finalist, SC16 best poster)
- However, simulation for smart cities requires full coupling in super-fine resolution
  - Traditional physics-based modeling too costly
  - Can we combine use of data analytics to solve this problem?



SC14, SC15 & SC16 solvers: ground simulation only

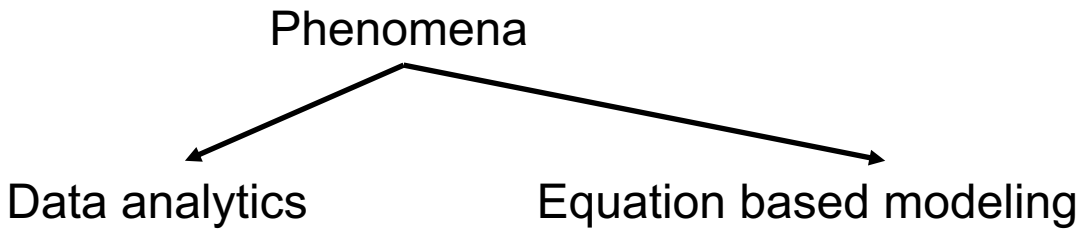




Fully coupled ground-structure simulation with underground structures

Data analytics and equation based modeling

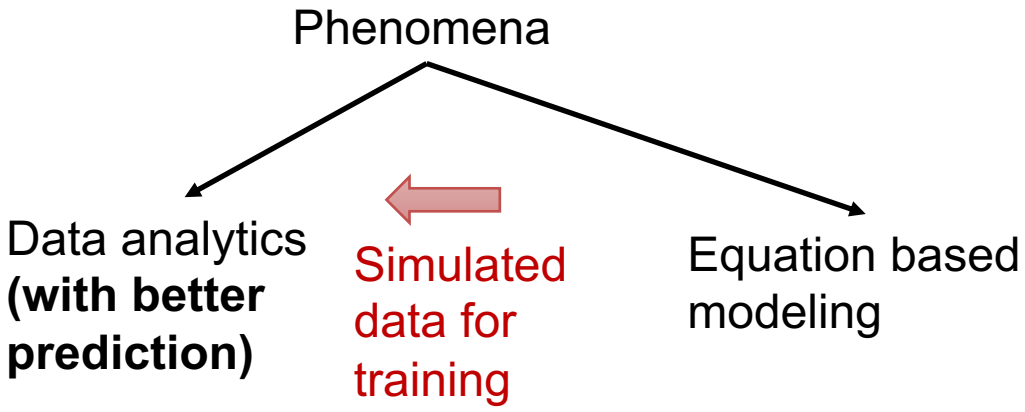
- Equation based modeling
  - Highly precise, but costly
- Data analytics
  - Fast inferencing, but accuracy not as high
- Use both methods to complement each other



Integration of data analytics and equation based modeling

- First step: use data generated by equation based modeling for data analytics training
  - Use of high-performance computing in equation based modeling enables generating very large amounts of high quality data
  - We developed earthquake intensity prediction method using this approach (SC17 Best Poster)

SC17

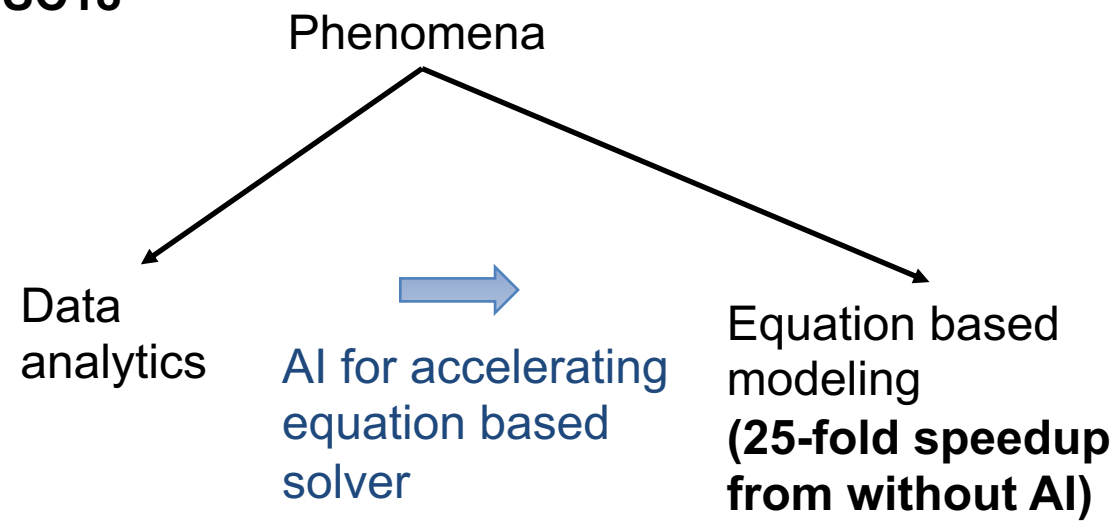


- SC14: equation based modeling
- SC15: equation based modeling
- SC16: equation based modeling
- **SC17: equation based modeling for AI**

Integration of data analytics and equation based modeling

- We extend this concept in this paper: train AI to accelerate equation based modeling

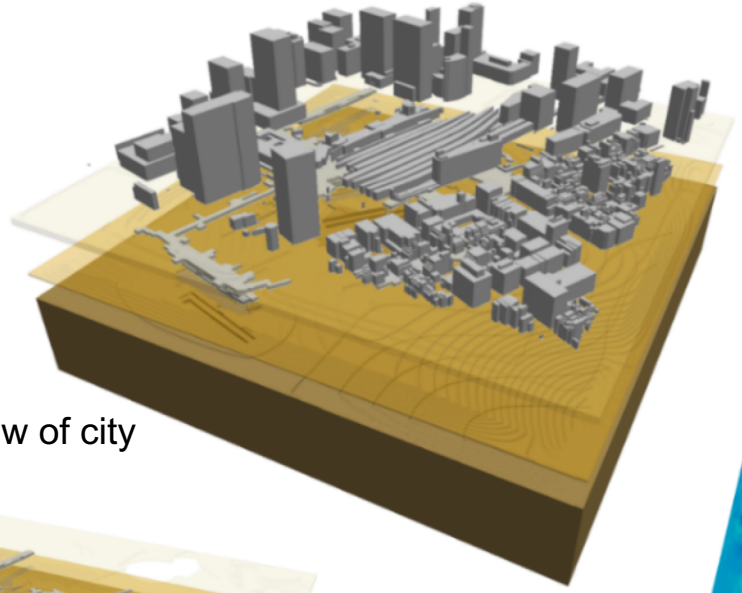
SC18



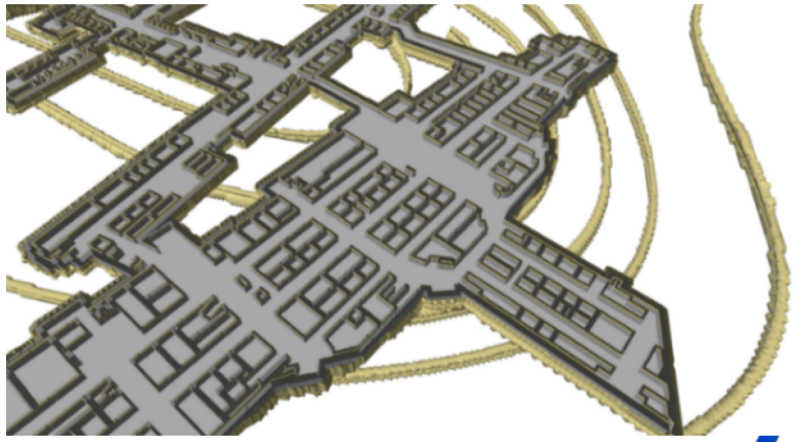
- SC14: equation based modeling
- SC15: equation based modeling
- SC16: equation based modeling
- SC17: equation based modeling for AI
- **SC18: AI for equation based modeling**

Earthquake modeling for smart cities

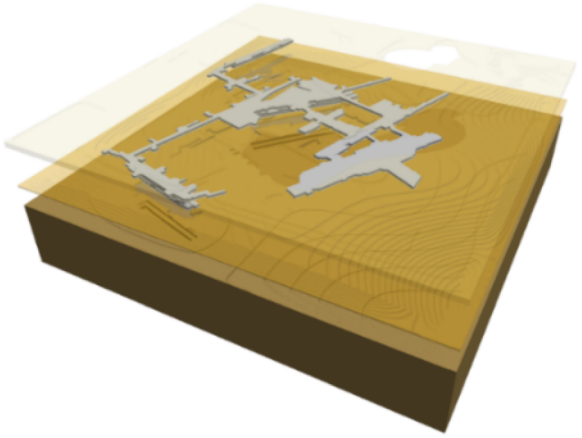
- By using AI-enhanced solver, we enabled fully coupled ground-structure simulation on Summit



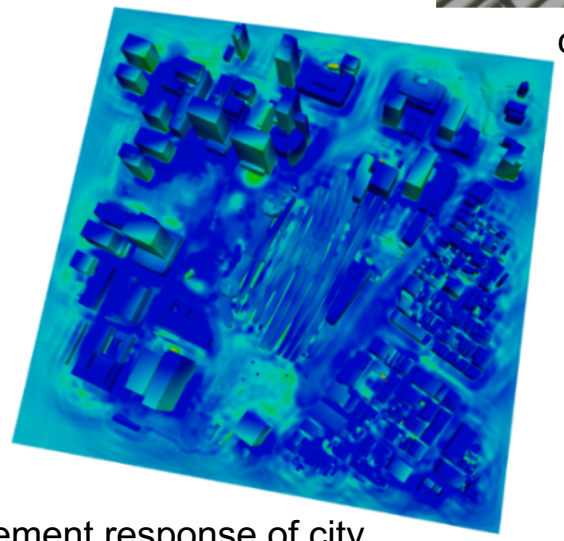
a) Overview of city model



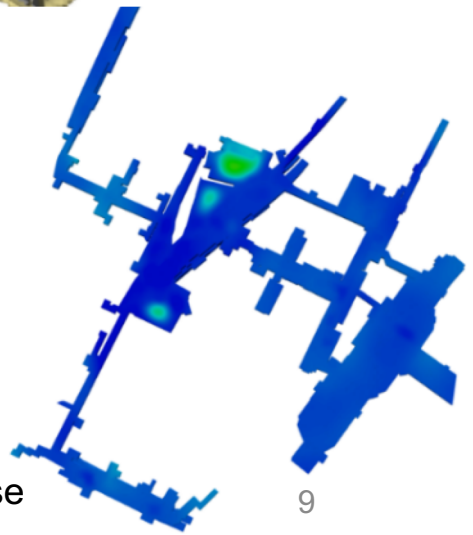
c) Close up view of city model



b) Location of underground structure



d) Displacement response of city



e) Displacement response of underground structure

Algorithm design of AI-enhanced solver

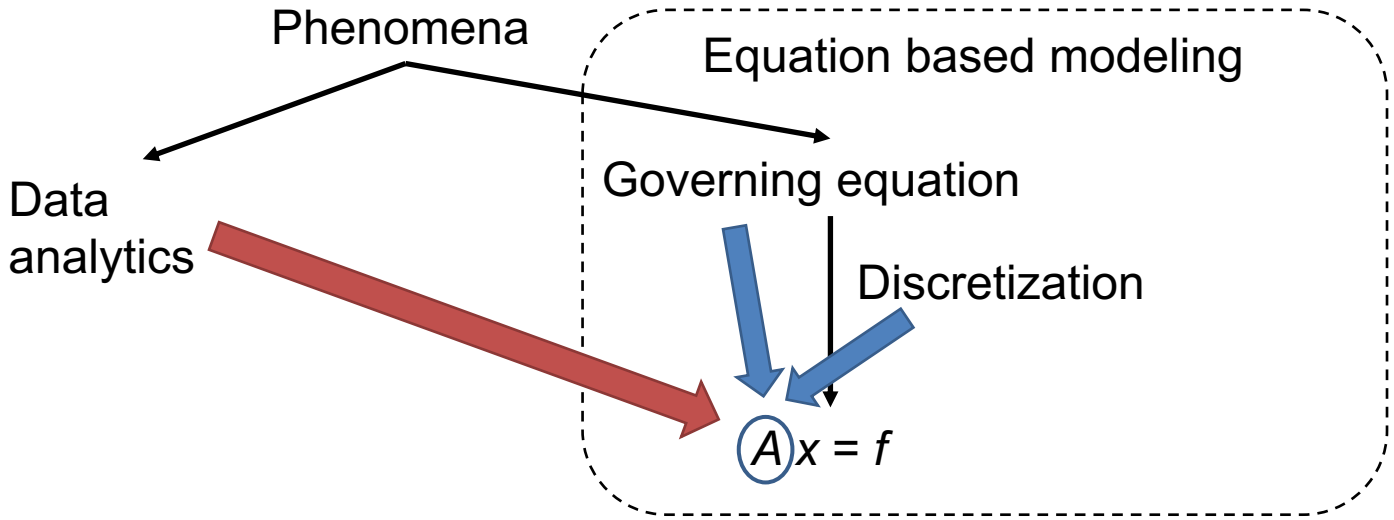
Difficulties of using data analytics to accelerate equation based modeling

- Target: Solve $A\,x = f$
- Difficulty in using data analytics in solver
  - Data analytics results are not always accurate
  - **We need to design solver algorithm that enables robust and cost effective use of data analytics, together with uniformity for scalability on large-scale systems**
- Candidates: Guess A^{-1} for use in preconditioner
  - For example, we can use data analytics to determine the fill-in of matrix; however, challenging for unstructured mesh where sparseness of matrix A is nonuniform (difficult for load balancing and robustness)

  ➡ Manipulation of A without additional information may be difficult…

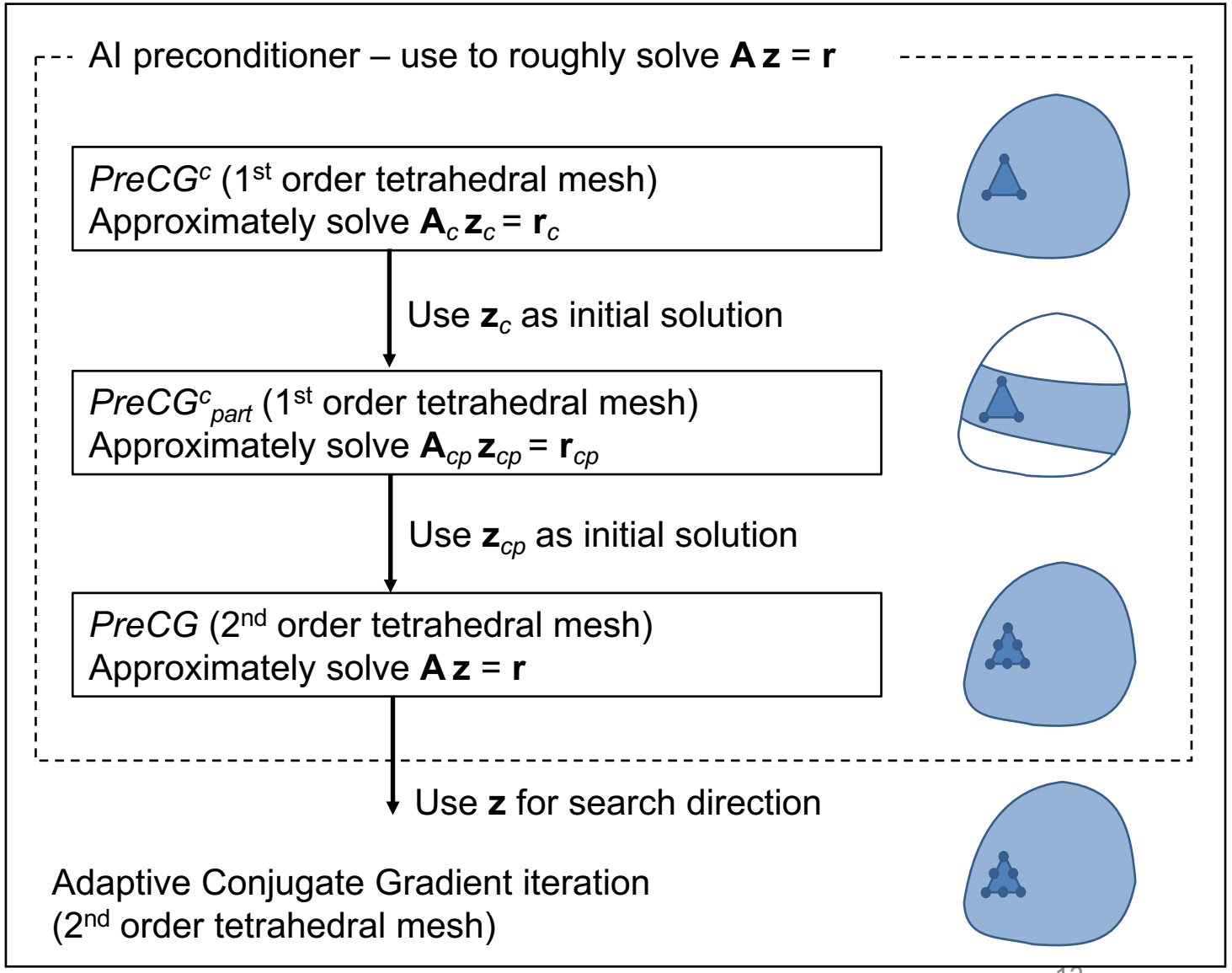
Designing solver suitable for use with AI

- Use information of underlying governing equation
  - Governing equation's characteristics with discretization conditions should include information about the difficulty of convergence in solver
  - Extract parts with bad convergence using AI and extensively solve extracted part



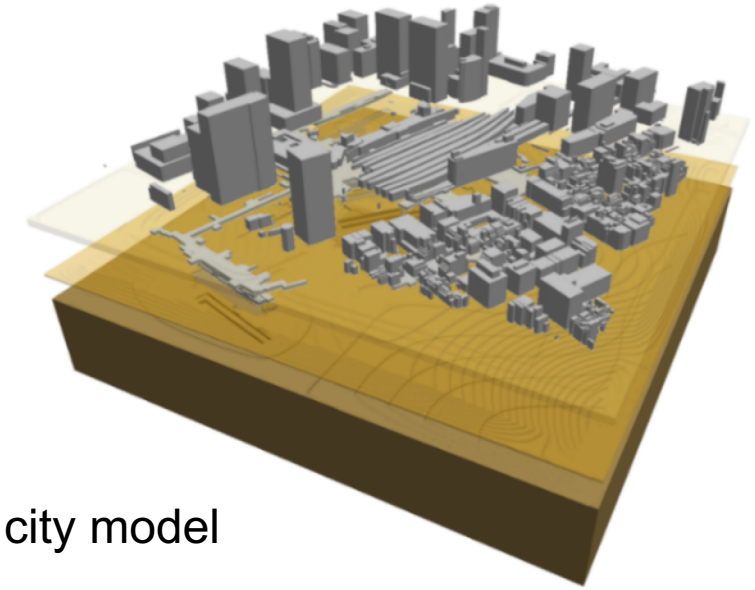
Solver suitable for use with AI

- Transform solver such that AI can be used robustly
  - Select part of domain to be extensively solved in adaptive conjugate gradient solver
  - Based on the governing equation's properties, part of problem with bad convergence is selected using AI

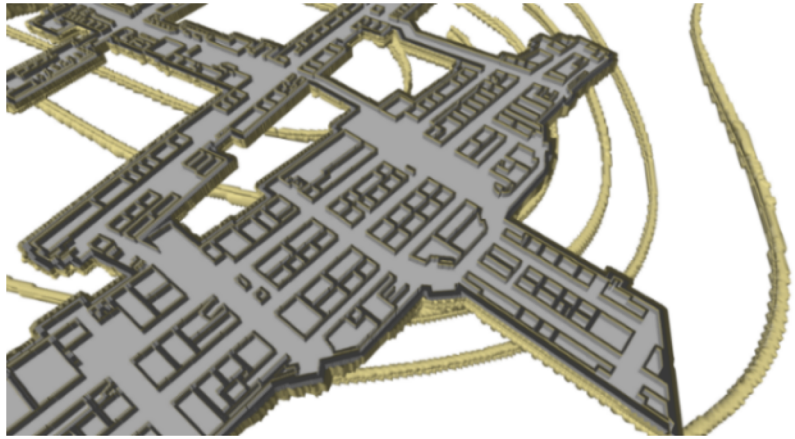


How to select part of problem using AI

- In discretized form, governing equation becomes function of material property, element and node connectivity and coordinates
  - Train an Artificial Neural Network (ANN) to guess the degree of difficulty of convergence from these data



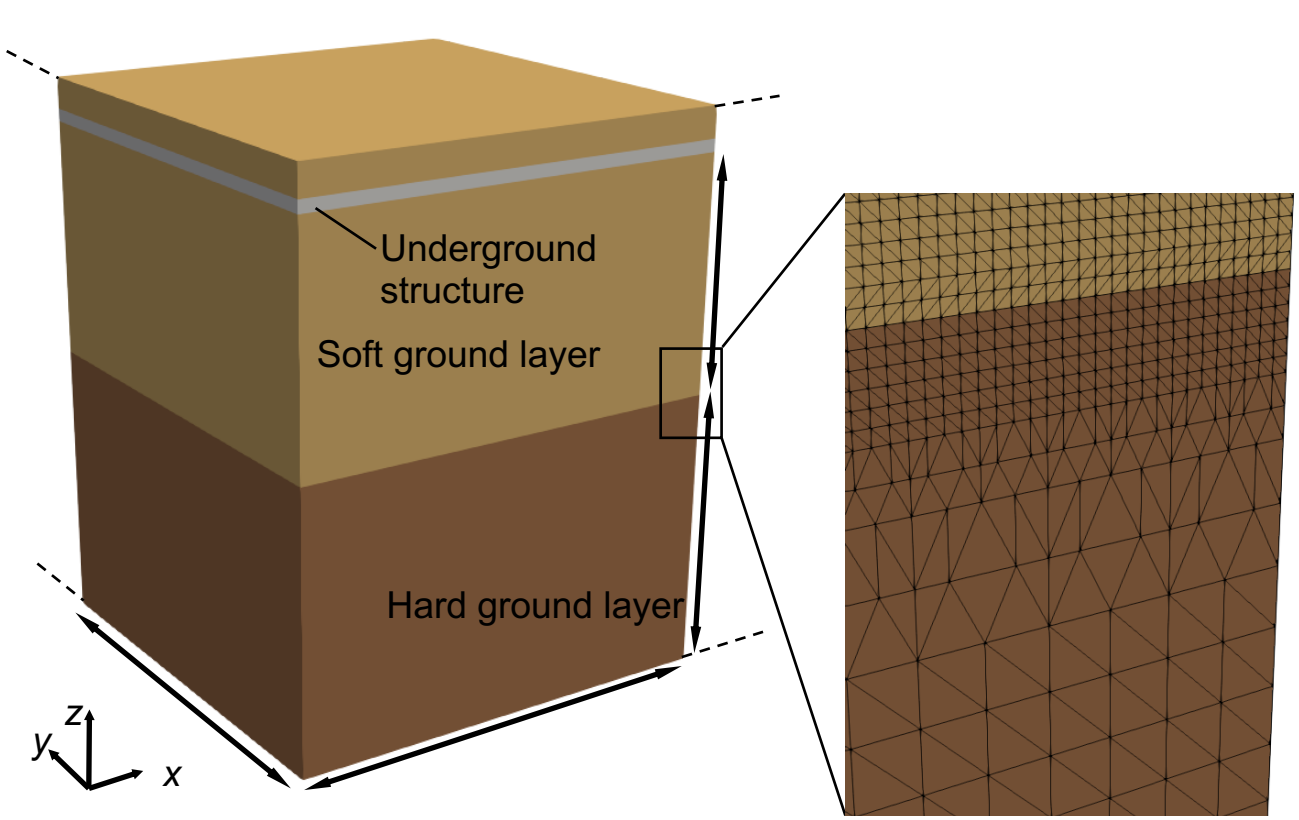
Whole city model

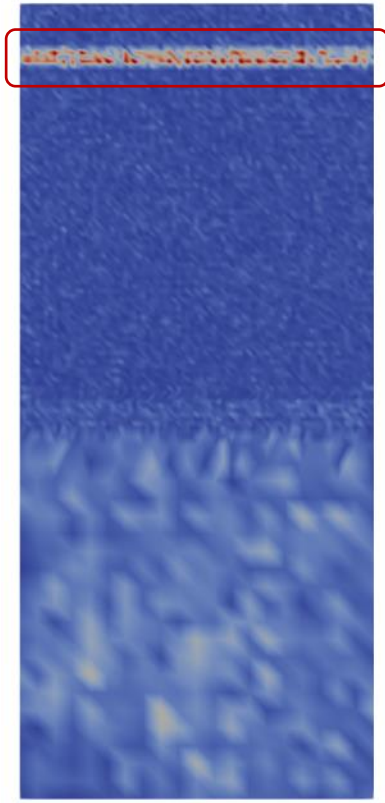


Extracted part by AI (about 1/10 of whole model)

Example of part selection using AI

- About 1/10 of domain is selected using generated ANN
  - Cost per iteration of selective solving is 1/10 of standard solver





Error distribution

Performance of solver with AI

- FLOP count decreased by 5.56-times from PCGE (standard solver; Conjugate Gradient solver with block Jacobi preconditioning)

	Without AI	With AI
CG iterations	132,665	88
PreCG^{c} iterations	-	5,803
PreCG$^{c}_{part}$ iterations	-	26,826
PreCG iterations	-	3,103
FLOPS count	184.7 PFLOP	33.2 PFLOP

- Name developed solver MOTHRA (iMplicit sOlver wiTH artificial intelligence and tRAnsprecision computing)

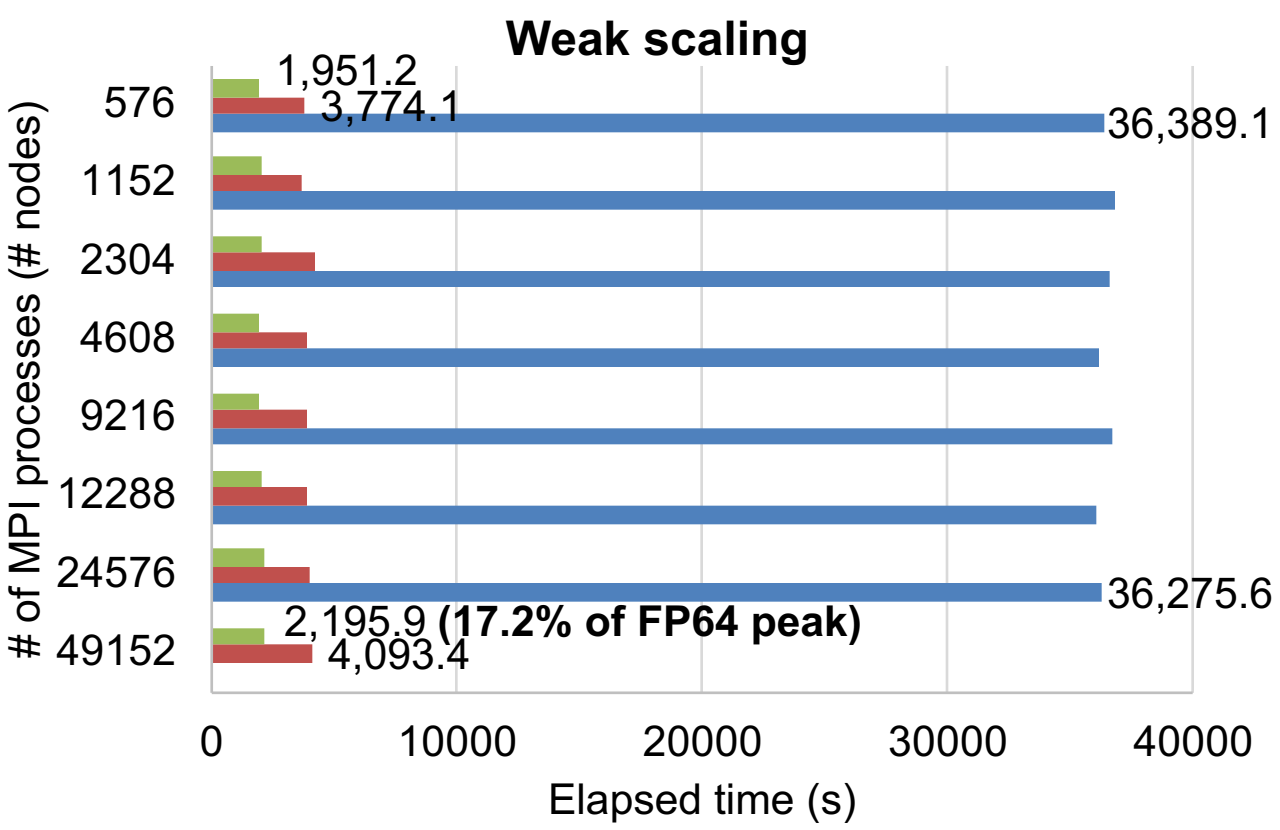
Performance of AI-enhanced solver of K computer

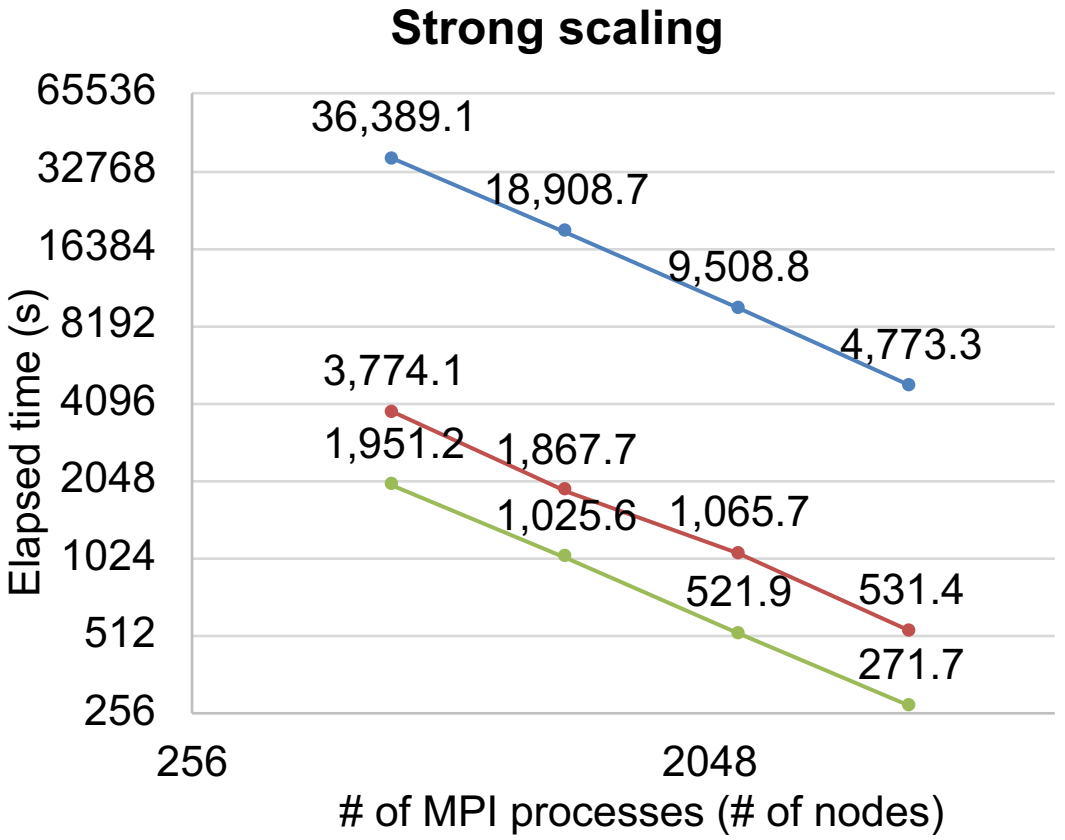
- Measure performance on CPU-based K computer
- Compare performance of solvers
  - PCGE (standard solver)
  - GAMERA (SC14 Gordon Bell Prize finalist solver, with multi-grid & mixed-precision arithmetic)
  - MOTHRA (developed solver)

K computer: 8 core CPU x 82944 node system
with peak performance of 10.6 PFLOPS

Performance of AI-enhanced solver on K computer

- Solver designed to have uniform load across large number of processors
  - Excellent load-balancing and scalability





MOTHRA (Developed) GAMERA (SC14) PCGE (Standard)

GPU implementation of AI-enhanced solver

Porting Strategy

- Our algorithm exhibits good performance/scalability on CPU-based supercomputer
- Same algorithm can be effective on GPU-based systems…?
  - Already designed for good scalability
  - Arithmetic count is reduced by AI in the solver

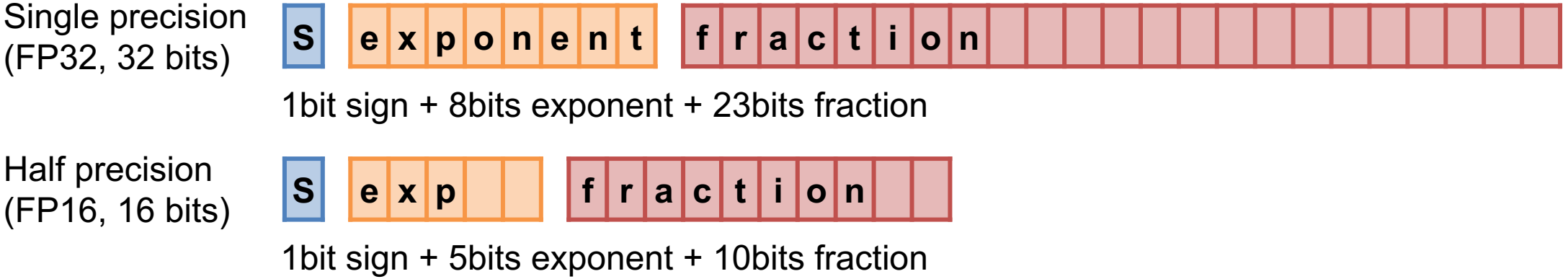
Requirements for GPU-based system

- Inter-node throughput of the system is relatively lower than previous supercomputer
- To attain higher performance, we have to reduce point-to-point communication cost more carefully
  - We have been using FP32-FP64 variables
  - Transprecision computing is available due to adaptive preconditioning

	K computer	**Piz Daint**	**Summit**
CPU/node	1 × SPARC64 VIIIfx	1 × Intel Xeon E5-2690 v3	2 × IBM POWER 9
GPU/node	-	1 × NVIDIA P100 GPU	6 × NVIDIA V100 GPU
Peak FP32 performance/node	0.128 TFLOPS	9.4 TFLOPS	93.6 TFLOPS
Memory bandwidth	512 GB/s	720 GB/s	5400 GB/s
Inter-node throughput	5 GB/s in each direction	10.2 GB/s	25 GB/s

Introduction of FP16 variables

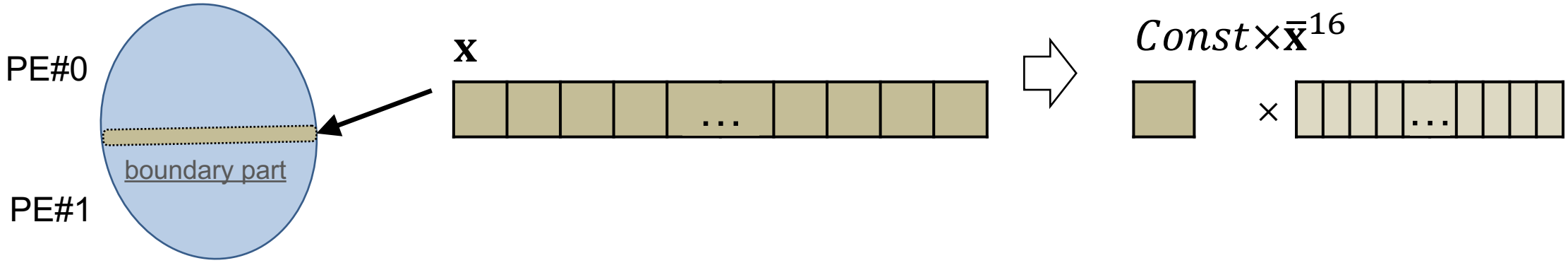
- Half precision can be used for reduction of data transfer size (Later used again in computation part)



- Using FP16 for whole matrix or vector causes overflow/underflow or fails to converge
  - Smaller exponent bits → small dynamic range
  - Smaller fraction bits → no more than 4-digit accuracy

FP16 for point-to-point communication

- FP16 MPI buffer only for boundary part
  - To avoid overflow or underflow, Original vector $\mathbf{x}$ is divided into one localized scaling factor $Const$ and FP16 vector $\bar{\mathbf{x}}^{16}$
- Data transfer size can be reduced
- $Const \times \bar{\mathbf{x}}^{16}$ does not match $\mathbf{x}$ exactly, but convergence characteristic is not changed for most problems

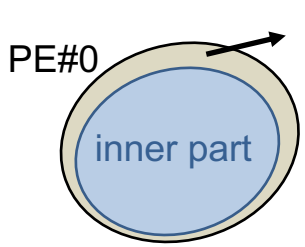


Overlap of computation and communication

i-th time step

1 : $\mathbf{r} = \mathbf{Ax}$
2 : synchronize $\mathbf{q}$ by point-to-point comm.
3 : $\mathbf{r} = \mathbf{b} - \mathbf{r}$; $\mathbf{z} = \mathbf{M}^{-1}\mathbf{r}$
4 : $\rho_a = 1$; $\alpha = 1$; $\rho_b = \mathbf{z} \cdot \mathbf{r}$; $\gamma = \mathbf{z} \cdot \mathbf{q}$
5 : synchronize ρ_b, γ by collective comm.
6 : ***while*** $(|\mathbf{r}_i|/|\mathbf{b}_i| > tolerance)$ ***do***
7 : $\beta = -\gamma\rho_a/\alpha$
8 : $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$; $\mathbf{p} = \mathbf{z} + \beta\mathbf{p}$
9 : $\mathbf{q} = \mathbf{Ap}$
10: synchronize $\mathbf{q}$ by point-to-point comm.
11: $\rho_a = \mathbf{p} \cdot \mathbf{q}$
12: synchronize ρ_a by collective comm.
13: $\alpha = \rho_b/\rho_a$; $\rho_a = \rho_b$
14: $\mathbf{r} = \mathbf{r} - \alpha\mathbf{q}$; $\mathbf{z} = \mathbf{M}^{-1}\mathbf{r}$; $\rho_b = \mathbf{z} \cdot \mathbf{r}$; $\gamma = \mathbf{z} \cdot \mathbf{q}$
15: synchronize ρ_b, γ by collective comm.
16: ***enddo***

- We solve $A\,x = f$ for each time step using Conjugate Gradient method
- Point-to-point communication is overlapped with matrix vector multiplication



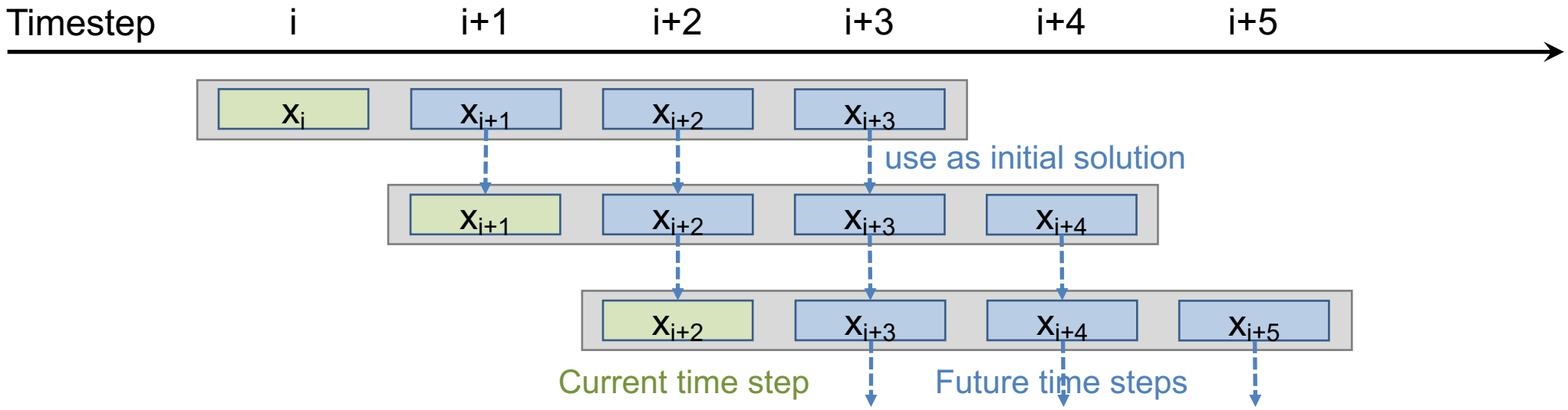
boundary part:
send/receive between other MPI processes

1. boundary part computation
2. inner part computation & boundary part communication

- However, this communication is still bottleneck of the solver

Overlap of computation and communication

- Introduction of time parallel algorithm
  - Solve four time steps in the nonlinear analysis in parallel
  - Compute 1 current time step and 3 future time steps used for initial guesses
  - Leads to improved peak performance and short time-to-solution
    - Arithmetic count for one iteration increases
    - Highly accurate initial solutions can be used



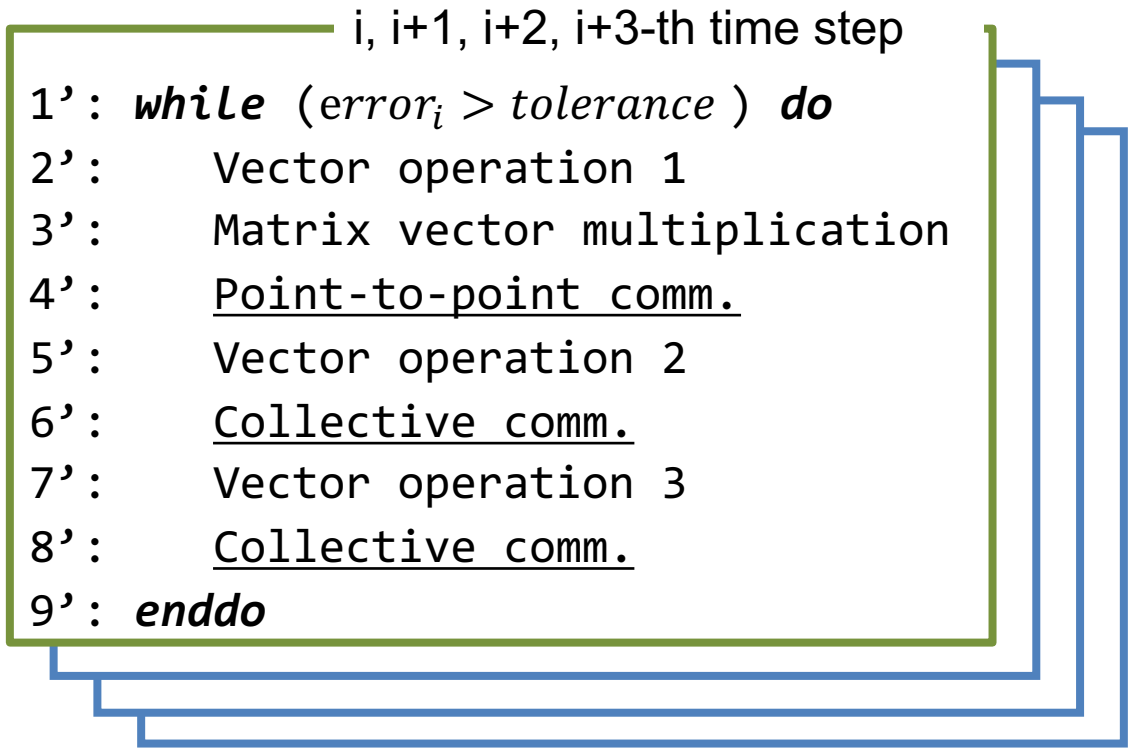
Overlap of computation and communication

i, i+1, i+2, i+3-th time step

```
1 : r = Ax
2 : synchronize q by point-to-point comm.
3 : r = b − r;  z = M^{-1}r
4 : ρ_a = 1;  α = 1;  ρ_b = z · r;  γ = z · q
5 : synchronize ρ_b,γ by collective comm.
6 : while ( |r_i|/|b_i| > tolerance ) do
7 :     β = −γρ_a/α
8 :     x = x + αp;   p = z + βp
9 :     q = Ap
10:     synchronize q by point-to-point comm.
11:     ρ_a = p · q
12:     synchronize ρ_a by collective comm.
13:     α = ρ_b/ρ_a ;  ρ_a = ρ_b
14:     r = r − αq; z = M^{-1}r;  ρ_b = z · r;  γ = z · q
15:     synchronize ρ_b,γ by collective comm.
16: enddo
```

- Conjugate Gradient method with time-parallel algorithm
  - Compute four vectors at each line
  - Looks complicated, but consists of simple operations

Overlap of computation and communication



- Simplified loop
  - Computation part
    - 3 groups of vector operations
    - 1 sparse matrix vector multiplication
  - Communication part
    - 1 point-to-point communication
    - 2 collective communication
- We modify algorithm to reduce communication cost by utilizing multiple time steps and vectors

Overlap of computation and communication

- 4 vectors are divided into 2 vectors × 2 sets
  - Point-to-point communication is overlapped with other vector operations
  - The number of collective communication is unchanged

i, i+1-th time step

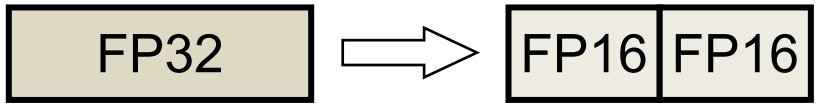
```
1’ : while (error_i > tolerance ) do
2’ :
3’ :    Collective comm.
4’ :    Vector operation 1
5’ :    Matrix vector multiplication
6’ :    Point-to-point comm.
7’ :    Vector operation 2
8’ :    Collective comm.
9’ :
10’:
11’:    Vector operation 3
12’: enddo
```

i+2, i+3-th time step

```
1’ : while (error_i > tolerance ) do
2’ :    Vector operation 2
3’ :    Collective comm.
4’ :
5’ :
6’ :    Vector operation 3
7’ :
8’ :    Collective comm.
9’ :    Vector operation 1
10’:    Matrix vector multiplication
11’:    Point-to-point comm.
12’: enddo
```

Low precision variables for computation part in the solver

- Manage to reduce communication cost in the solver
- Now, it's worth reducing computation cost to improve time-to-solution by using transprecision computing
  - FP21 for memory bound vector operations
  - FP16 for Element-by-Element kernel
    - Process 2 × FP16 variables on 2-element vector simultaneously and expect double performance



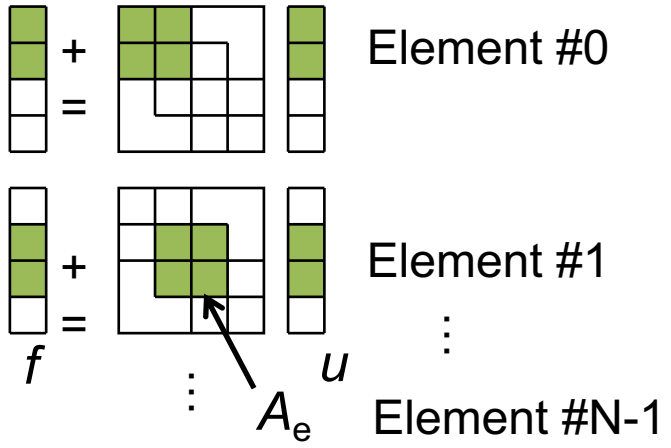
FP16 computation in Element-by-Element method

- Matrix-free matrix-vector multiplication
  - Compute element-wise multiplication
  - Add into the global vector

- Normalization of variables per element can be performed
  - To avoid underflow/overflow, we use values close to 1 in multiplication

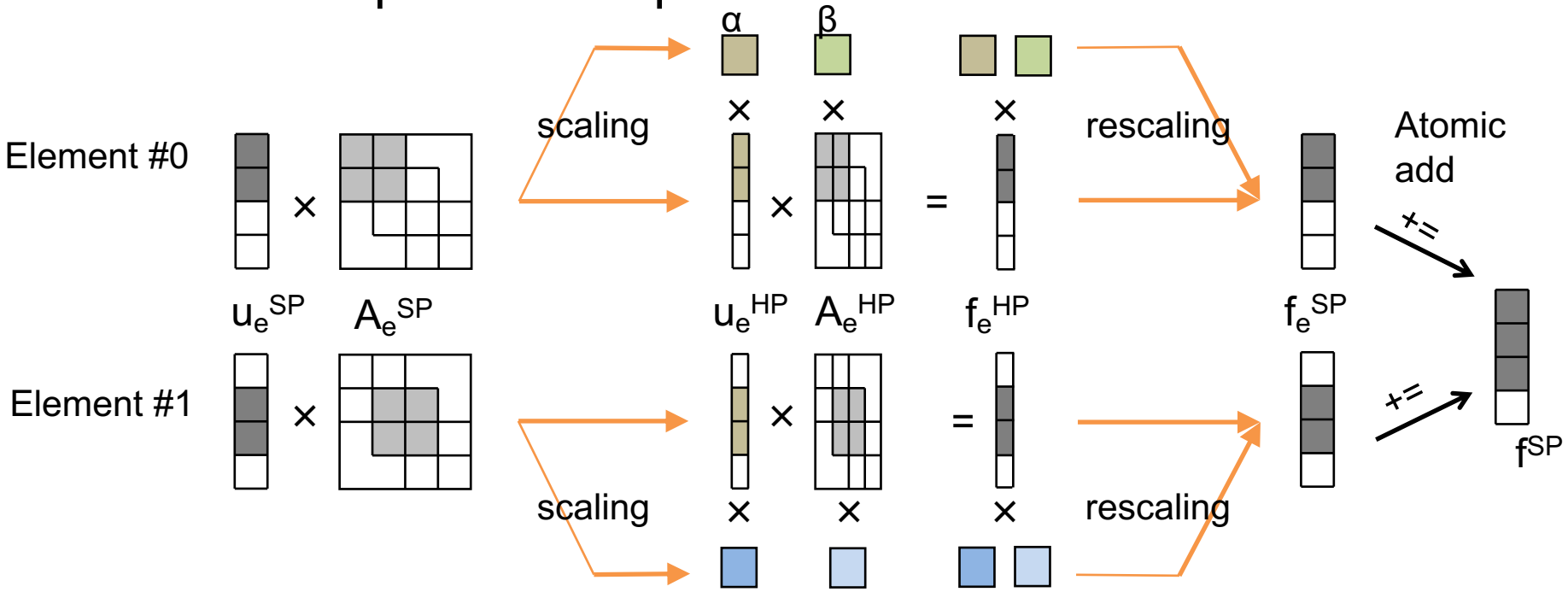
Element-by-Element
(EBE) method

$f = \Sigma_e P_e A_e P_e^T u$
[A_e is generated on-the-fly]



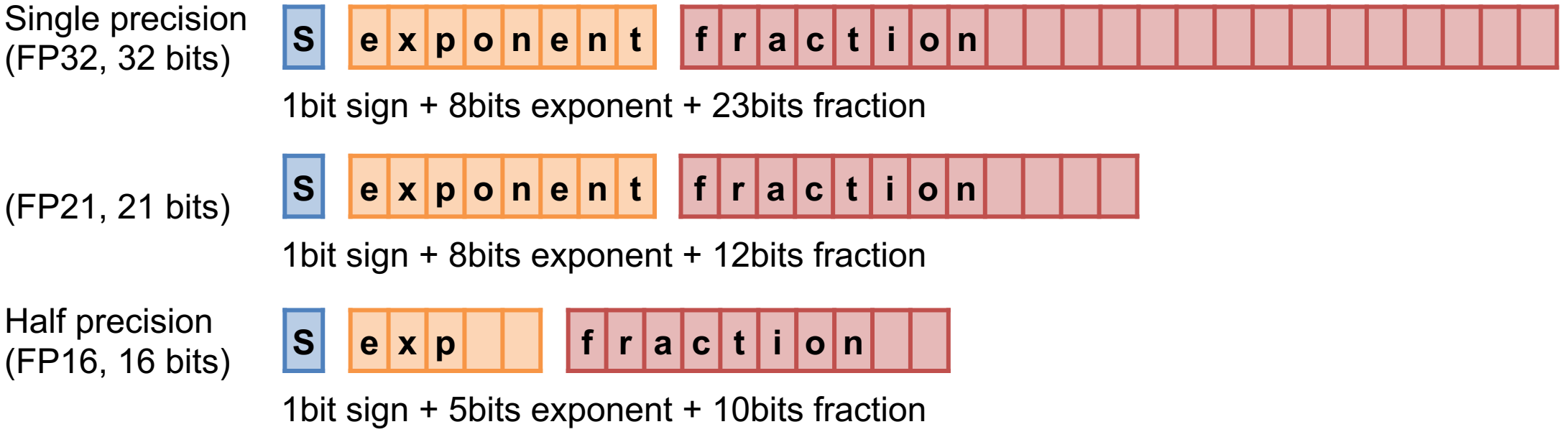
Implementation of FP16 computation

- Vectors u_e are scaled to avoid overflow/underflow in using half precision
- Element matrix A_e is generated on-the-fly and also scaled
  - reorder computation ordering so that values close to 1 are used
- Most costly multiplication can be computed in FP16
- Achieved 71.9% peak FP64 performance on V100 GPU



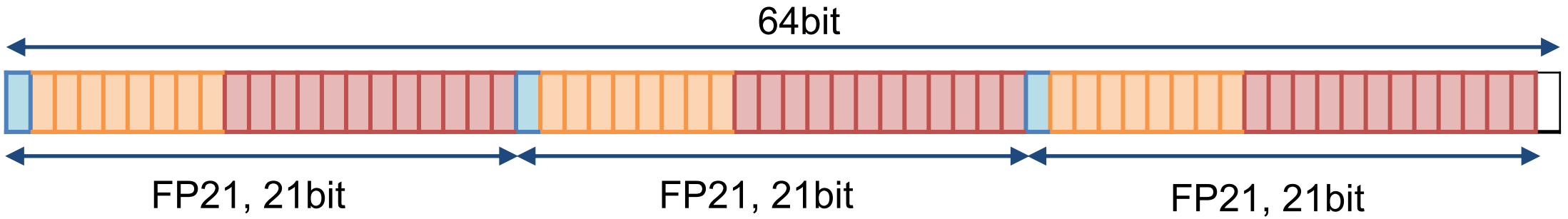
Introduction of custom data type: FP21

- Most computation in CG loop is memory bound computation
  - However, it's impossible to use FP16 for whole vector
- Trying to use FP21 variables for other memory bound computation



Implementation of FP21 computation

- Not supported in hardware, used only for storing
  - FP21(stored)⇐bit operation⇒FP32(computed)
- FP21 × 3 are stored into 64bit array
  - We are solving 3D finite element solver, so x, y, and z components can be stored as one components of 64 bits array



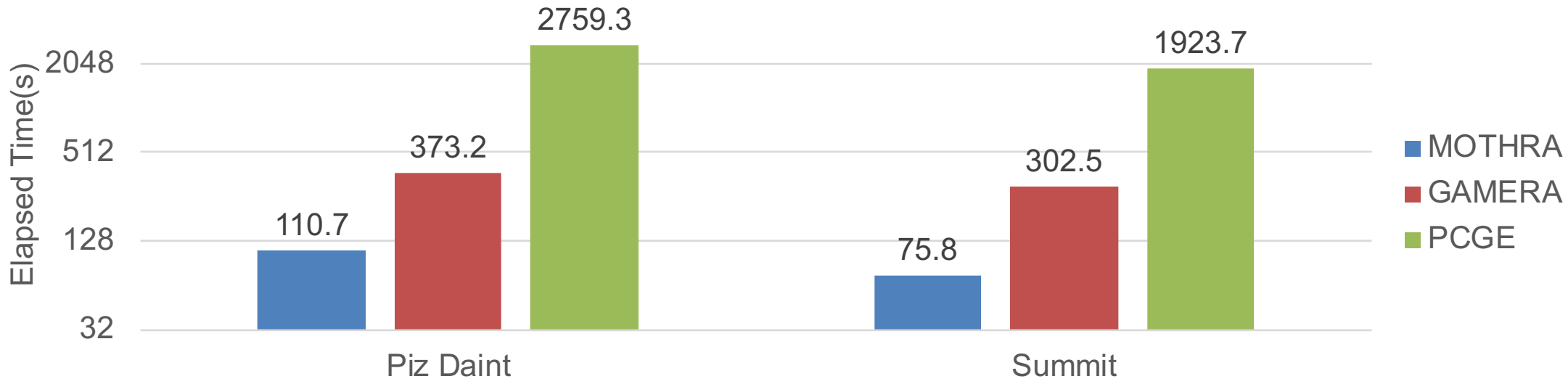
- 1/3 of memory consumption compared to FP64 variables

Performance measurement

On GPU-based supercomputer, Piz Daint and Summit

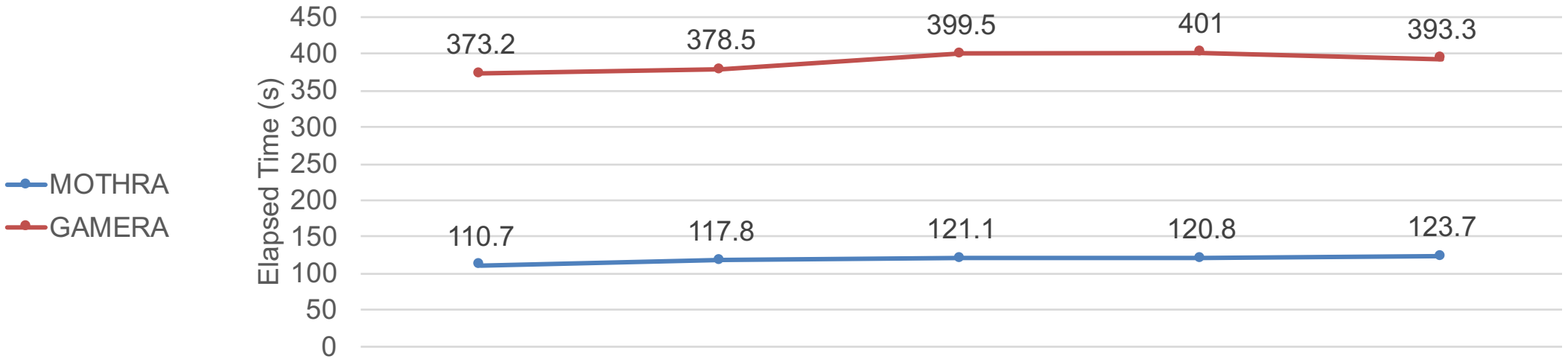
Performance comparison

- We solve the same problem as K-computer using 288 GPUs on Piz Daint & Summit
  - PCGE (conventional solver)
  - GAMERA (SC14 Gordon Bell Finalist solver)
  - MOTHRA (our proposed solver)
- MOTHRA is sufficiently faster than other solvers on Summit
  - 25.3-fold speedup from PCGE
  - 3.99-fold speedup from GAMERA
- Convergence characteristic is not largely changed even when we use FP16 & FP21



Weak scaling on Piz Daint

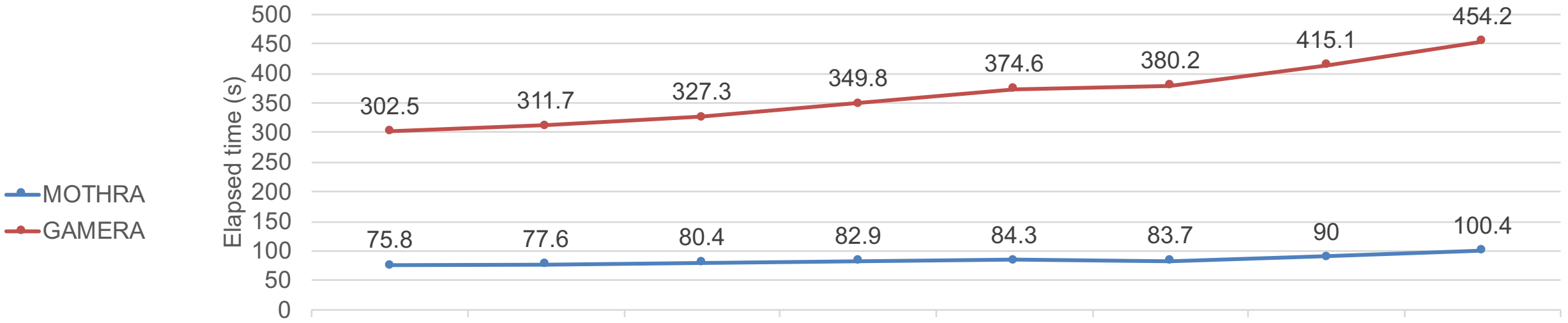
- MOTHRA demonstrates high scalability (89.5% to the smallest case)
  - Leading to 19.8% peak FP64 performance on nearly full system



# of GPUs	288	576	1152	2304	4608
# of node	288	576	1152	2304	4608
DOF	3.5×10^9	7×10^9	14×10^9	28×10^9	56×10^9
MOTHRA's efficiency to FP64 peak	22.1%				19.8%

Weak scaling on Summit

- Scalability greatly improves compared to previous solver GAMERA
- MOTHRA demonstrates high scalability
  - Leading to 14.7% peak FP64 performance on nearly full system



# of GPUs	288	576	1152	2304	4608	6144	12288	24576
# of node	48	96	192	384	768	1024	2048	4196
DOF	3.5×10^9	7×10^9	14×10^9	28×10^9	56×10^9	75×10^9	151×10^9	302×10^9
MOTHRA's efficiency to FP64 peak	19.5%							14.7%

Summary and future implications

- Combination with FP16-FP21-FP32-FP64 transprecision computation/communication techniques enabled high performance of
  - 25.3-fold speedup from standard solver
  - 3.99-fold speedup from state-of-the-art SC14 Gordon Bell Finalist solver
  - 14.7% peak FP64 performance on near full system of Summit (4096 nodes)

Summary and future implications

- Integration of data analytics and equation based modeling is one of the key questions in high performance computing
  - New class of algorithms is required for accelerating equation based simulation by data analytics
  - We accelerated earthquake simulation by designing a scalable solver algorithm that can robustly incorporate data analytics
- Idea of accelerating simulations with data analytics can be generalized for other types of equation based modeling
  - Future development of high-performance computer systems supporting both data analytics and equation based simulations is key tool for advance of science and engineering

Acknowledgments

Our results were obtained using the Summit at Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory (ORNL), Piz Daint at Swiss National Supercomputing Centre (CSCS), and K computer at RIKEN Center for Computational Science (R-CCS, proposal numbers: hp170249, hp180217).
We thank Yukihiko Hirano (NVIDIA) for coordination of the collaborative research project. We thank Christopher B. Fuson, Don E. Maxwell, Oscar Hernandez, Scott Atchley, Ver´onica Melesse-Vergara (ORNL), Jeff Larkin, Stephen Abbott (NVIDIA), Lixiang Luo (IBM), Richard Graham (Mellanox Technologies) for generous support concerning use of Summit.
We thank Andreas Jocksch, Luca Marsella, Victor Holanda, Maria Grazia Giuffreda (CSCS) for generous support concerning use of Piz Daint. We thank the Operations and Computer Technologies Division of RCCS and the High Performance Computing Infrastructure helpdesk for generous support concerning use of K computer. We thank Sachiko Hayashi of Cybernet Systems Co., Ltd. for support in visualizing the application example.
We acknowledge support from Post K computer project (Priority Issue 3 - Development of integrated simulation systems for hazards and disasters induced by earthquakes and tsunamis) and Japan Society for the Promotion of Science (18H05239, 26249066, 25220908, and 17K14719).