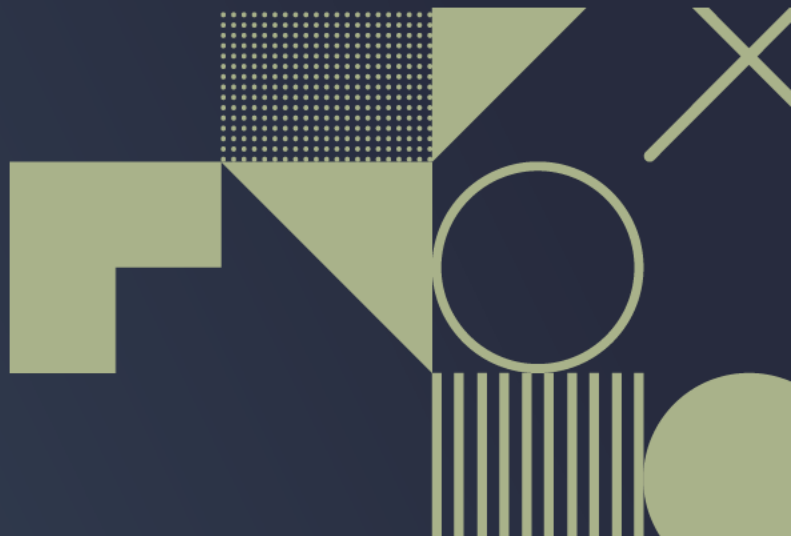
Enabling and Scaling the HPCG Benchmark on the Newest Generation Sunway Supercomputer with 42 Million Heterogeneous Cores

Qianchao Zhu, Hao Luo, Chao Yang, Mingshuo Ding, Wanwang Yin, Xinhui Yuan

November 2021



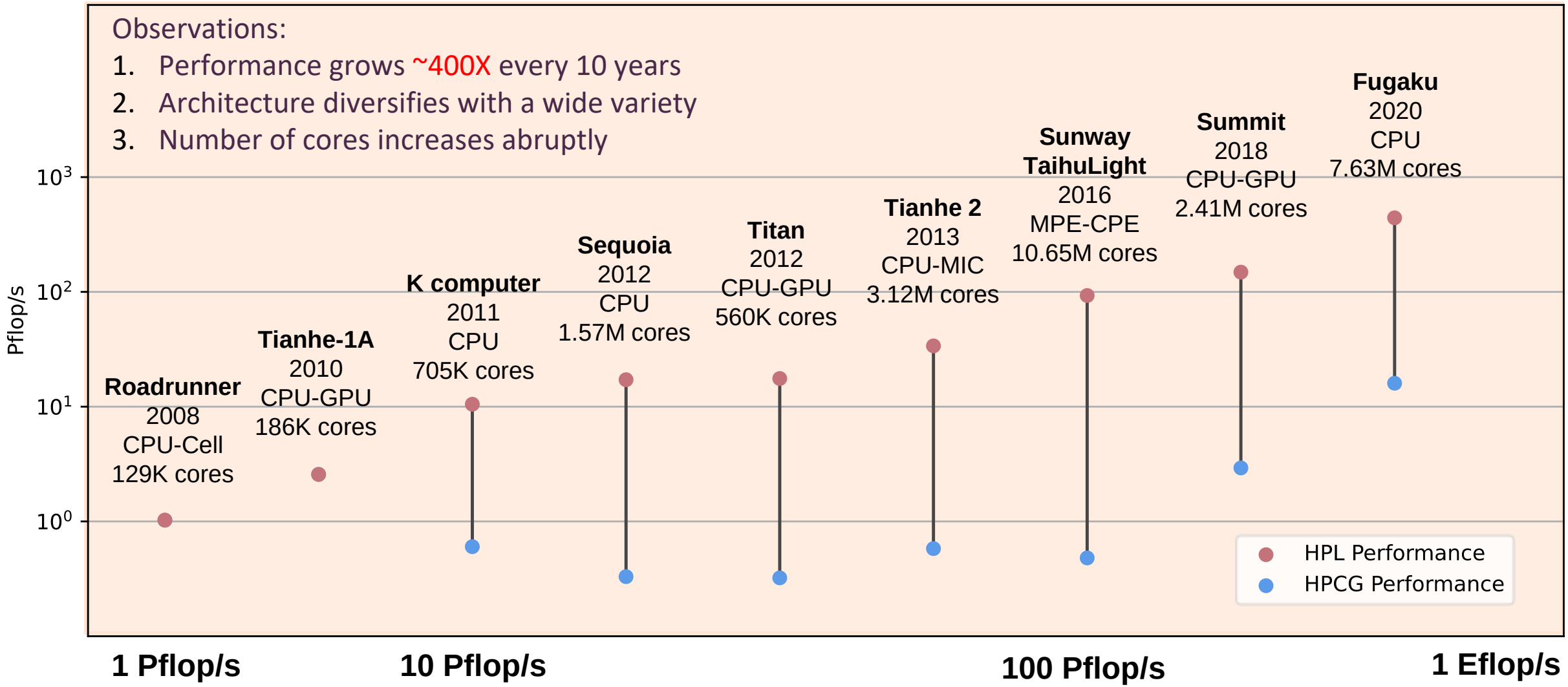
PEKING UNIVERSITY

Outline of the Talk

Outline of the Talk

Rapid Development of Leadership Supercomputers

PEKING UNIVERSITY



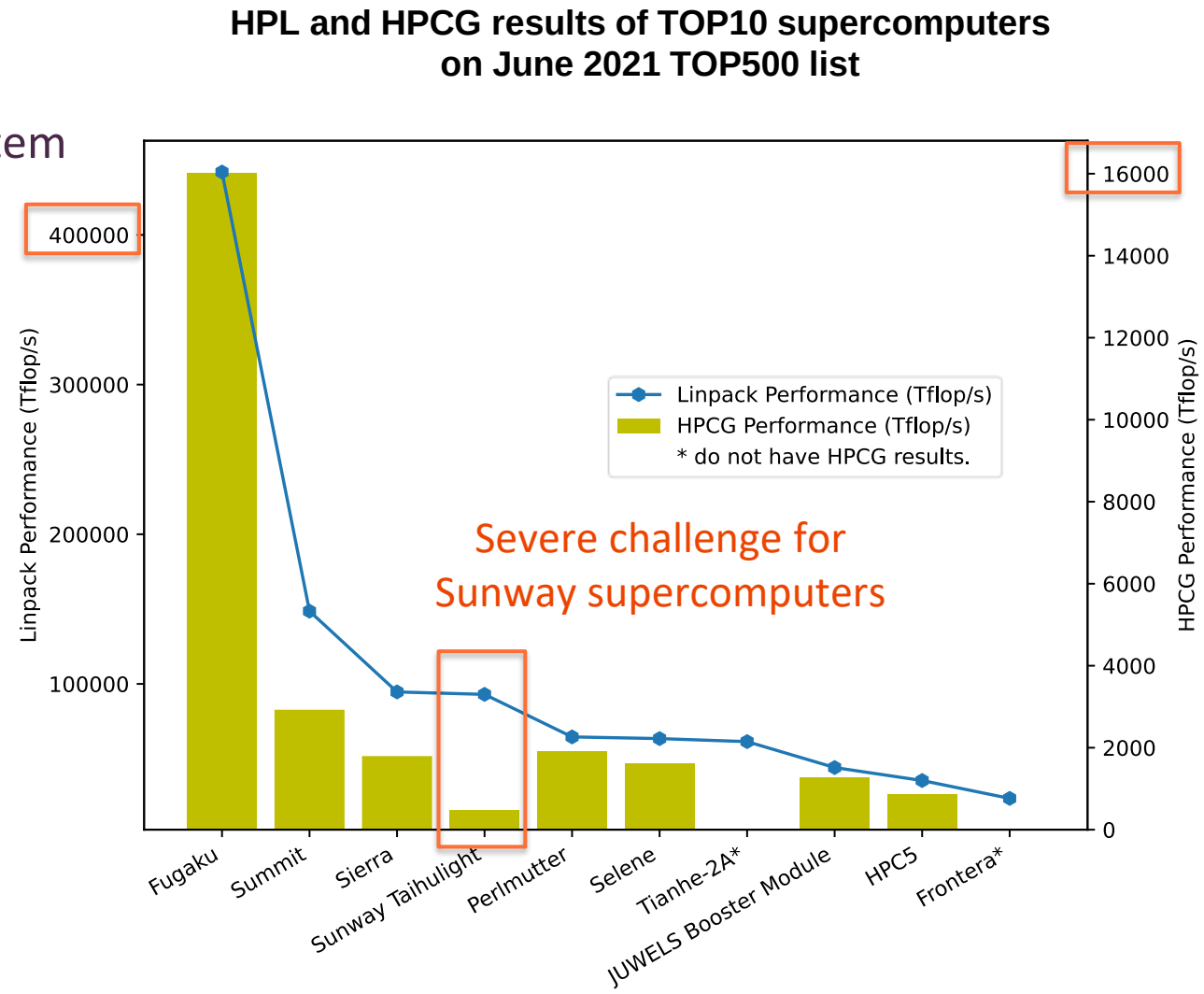
Heterogeneous Systems are Mainstream

Architecture of TOP10 supercomputers on June, 2021 TOP500 list

- Heterogeneous architecture has been recognized as an important approach to design supercomputers
- The total numbers of cores are substantially large, reaching a level of O(10^7)
- However, the performance of memory access increases much slower than the computing capability!

From HPL to HPCG

- High Performance Linpack (HPL):
  - Direct solver for large-scale dense linear system
  - High arithmetic intensity
  - Computation bound
- High Performance Conjugate Gradient (HPCG):
  - Iterative solver for large-scale sparse linear system
  - Low arithmetic intensity
  - Memory bandwidth bound
- Large gap between HPL and HPCG performance



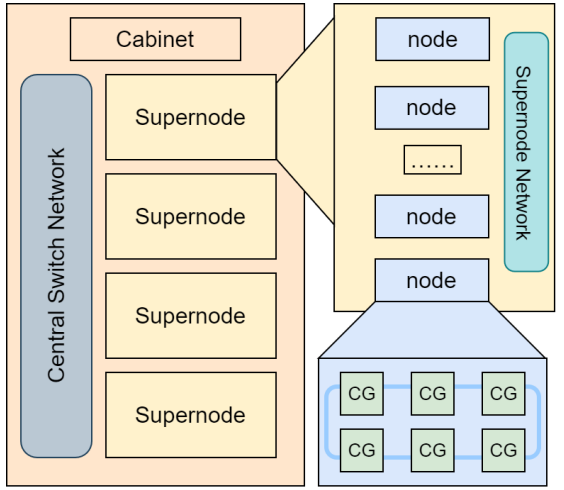
Outline of the Talk

The Newest Generation Sunway Supercomputer

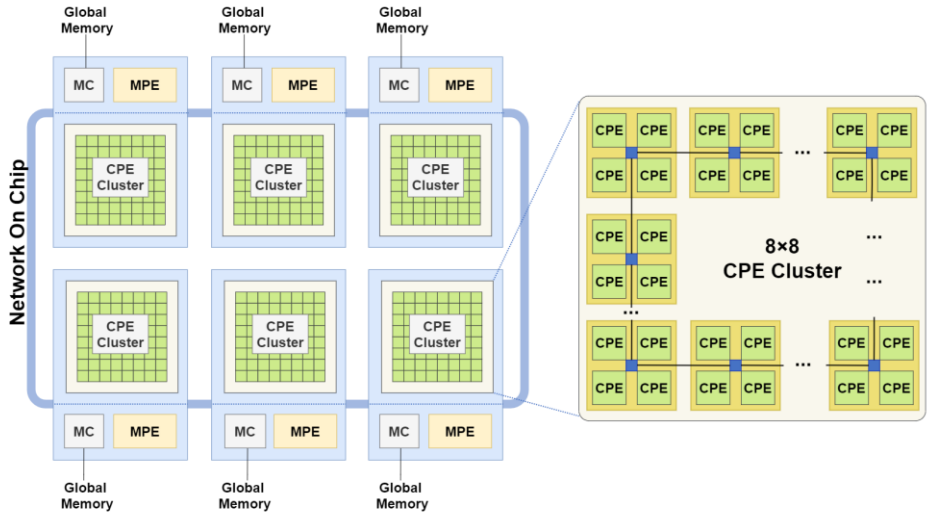
PEKING UNIVERSITY

- Over 108,960 nodes, 42 million heterogeneous cores
- Comprised of hundreds of supernodes through the central switch network, with each supernode consisting 256 computing nodes
- Each computing node is equipped with an SW26010-Pro manycore processor, based on heterogeneous architecture of on-chip integration

	SW26010	SW26010-Pro
Number of core groups	4	6
Peak performance	3.1 Tflop/s	14 Tflop/s (4.5X)
Memory bandwidth	136 GB/s	307 GB/s (2.25X)
Capacity of Local Device Memory (LDM) for each CPE	64 KB	256 KB (either 32KB or 128KB can be configured as fast data cache)
Supported vector instructions within MPE/CPE	256 bit / 256 bit	256 bit / 512 bit
Data exchange among the CPE cluster	Register Communication	Remote Memory Access (RMA)



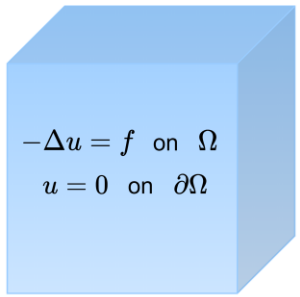
System architecture

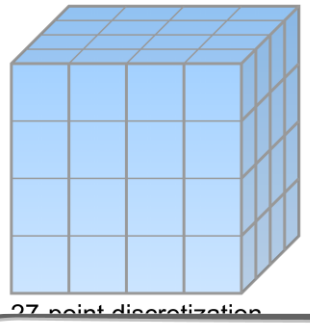


SW26010-Pro manycore processor

Outline of the Talk

The HPCG Benchmark





27-point discretization

ALGORITHM 1: Preconditioned Conjugate Gradient for $Ax = b$

input: $A, b, x_0, it_{max}, \varepsilon$

$r_0 \leftarrow b - Ax_0$

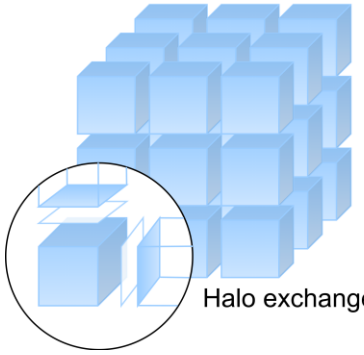
for $i = 1, 2,$ **to** it_{max} **do**

$z_i \leftarrow M^{-1} r_{i-1}$ ▷ *MG*

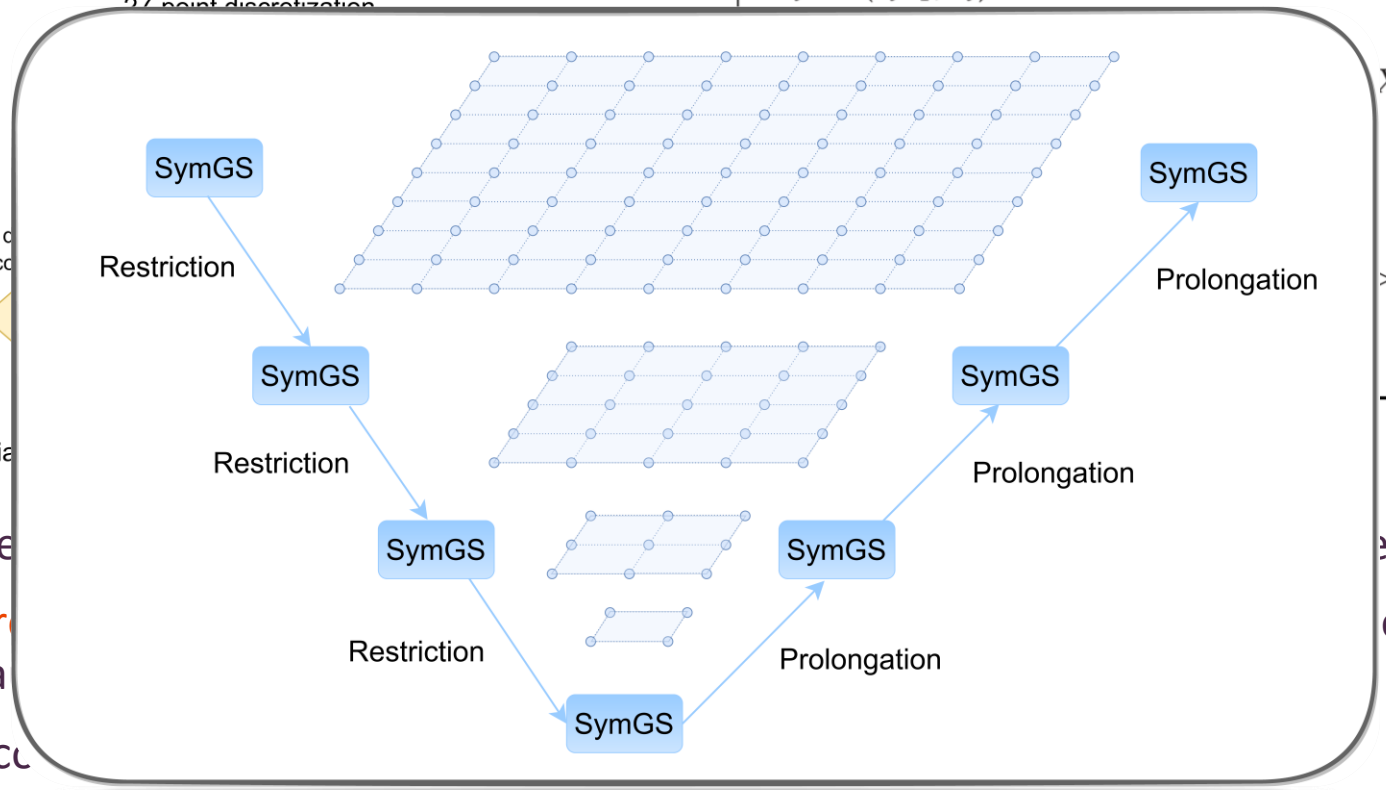
$s_i \leftarrow (r_{i-1}, z_i)$ ▷ *DOT*

XPBY

▷ *DOT*

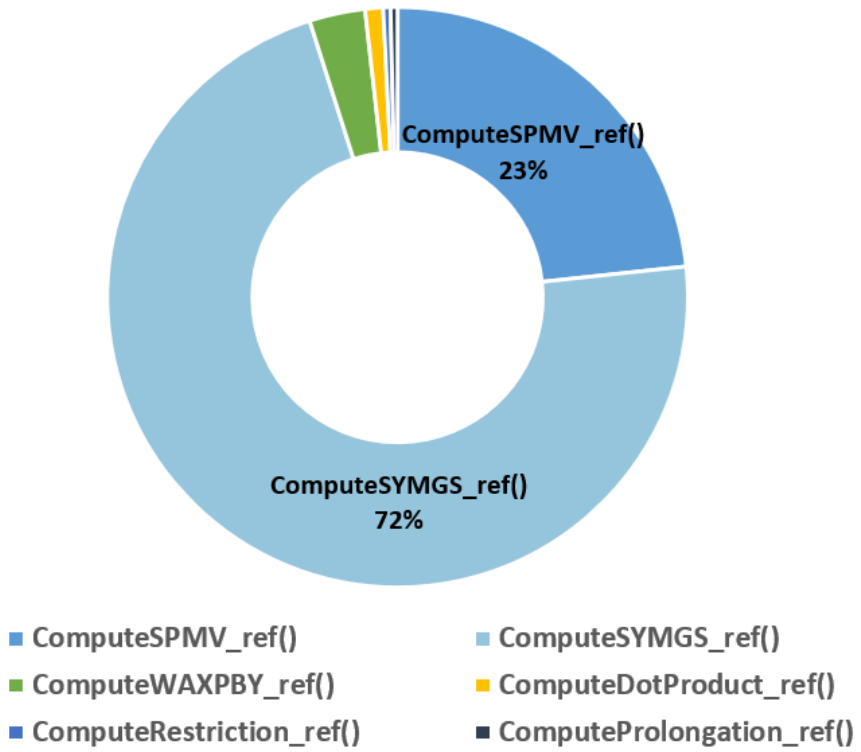


Halo exchange via



- HPCG solves sparse … equation in a 3D cube
- Main algorithm: pr… etric multigrid preconditioner with symmetric Ga…
- Two types of MPI c…

Profiling of the Reference HPCG Code



Run time breakdown

Major operations in each PCG iteration of HPCG

Operations	SymGS	SpMV	DotProduct	WAXPBY
Call counts	7	4	3	3
Arithmetic intensity (flop/byte)	0.152	0.156	0.125	0.125

- SPMV Computation (~23%)
  - lower computation-to-data-access ratios
  - indirect and irregular memory access
  - adequate parallelism
- SYMGS Computation (~72%)
  - lower computation-to-data-access ratios
  - indirect and irregular memory access
  - intrinsic data dependency – cause performance penalty

Test environment: SW26010-Pro, 8 MPI Processes and 256*128*256 for each subdomain, HPCG v3.1 reference

Outline of the Talk

Optimization 1: Fast Convergent Two-Level Blocking

PEKING UNIVERSITY

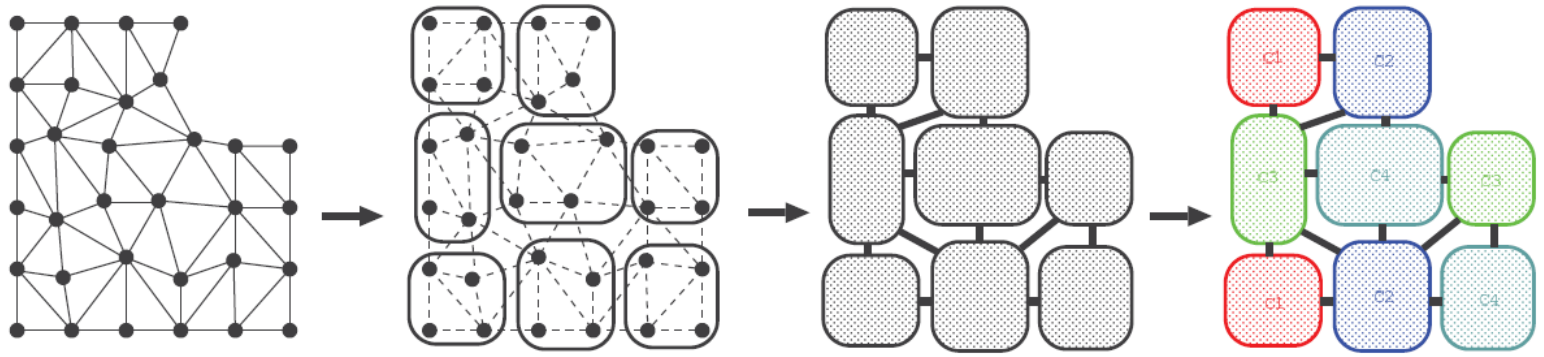
```
//SymGS Forward Computation
for(int i = 0; i < nrow; i++){
    double *Val = A.matrixVal[i];
    int *Idx = A.matrixIdx[i];
    int nz = A.nonzerosInRow[i];
    double Dia = A.matrixDiagonal[i][0];

    double sum = rv[i];
    for(int j = 0; j < nz; j++){
      sum = sum - Val[j] * xv[Idx[j]];
    }
    sum += xv[i] * Dia;
    xv[i] = sum /Dia;
}
```

Code of forward iteration in SymGS computation

- Strong data dependency
- Limited parallelism
- Poor data locality

How to parallelize a sparse triangular solver for a linear system on many-core heterogeneous architectures?

Block Multi-Color Ordering Method [1]



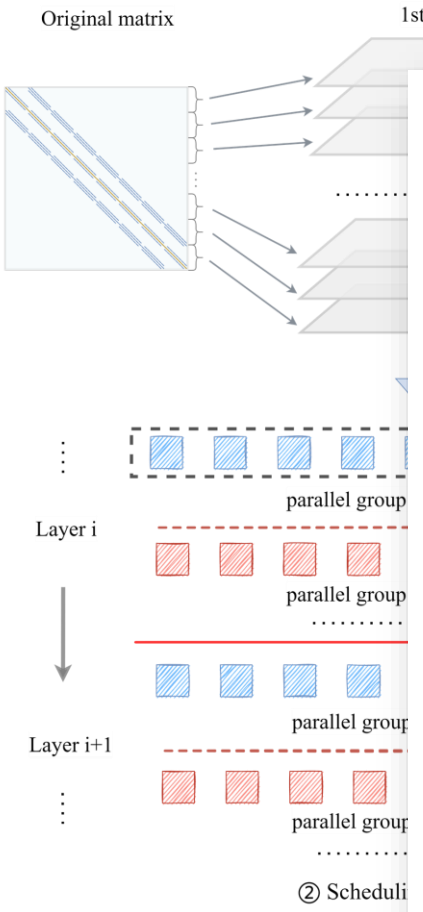
Transforming to graph **Blocking nodes** **Connecting blocks** **Coloring blocks**

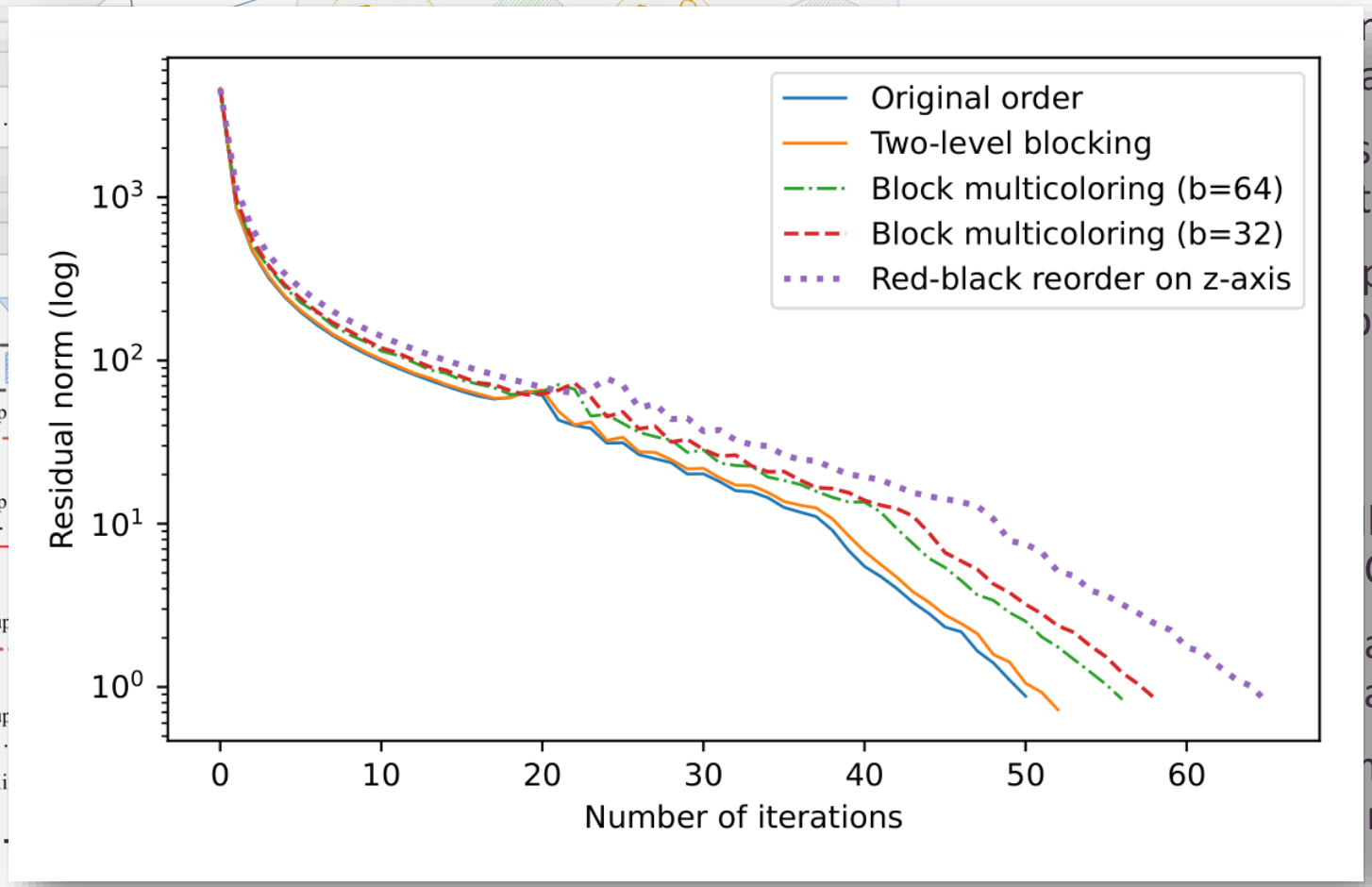
- Possible for thread-level parallelism
- Better data locality
- Poor convergence rate
  - **56** iterations (block size = 64 with 8 colors) vs **50** iterations (reference)

[1] Iwashita T, et al. Algebraic block multi-color ordering method for parallel multi-threaded sparse triangular solver in ICCG method[C] International Parallel and Distributed Processing Symposium. IEEE, 2012

Optimization 1: Fast Convergent Two-Level Blocking (cont'd)

How to improve the convergence rate of the block multi-coloring method?





ng scheme is applied to
al matrix

s created to distribute the
the CPEs equally

p can be processed
pawning multiple threads on

lelism is dropped, it is still
CPE cluster

ate is increased (only 51
ame problem)

nethod provides possibilities
nd kernel fusion

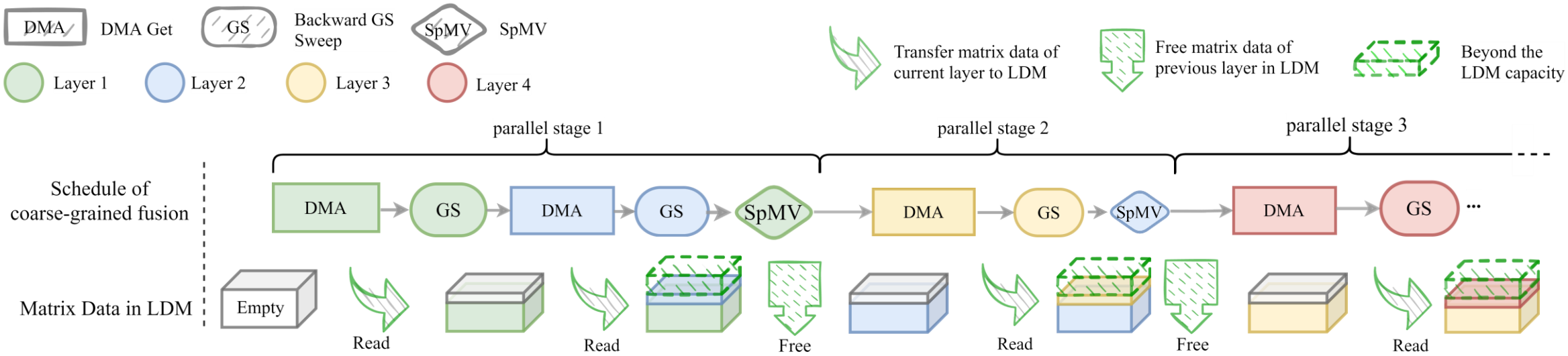
Optimization 2: Constraint-Aware Fine-Grained Fusion

$x_h \leftarrow x_h^* + (D_h + U_h)^{-1}(r_h - Ax_h^*)$ ▷ Backward Sweep

$r_h \leftarrow b_h - A_h x_h$ ▷ Residual (SPMV)

Can be fused together to reduce memory access

	SymGS	SpMV	Fused SymGS-SpMV
Arithmetic Intensity (flop/byte)	0.152	0.156	**0.226**



Coarse-grained fusion

- Failed to retain all required data in LDM due to the constraint of LDM capacity.

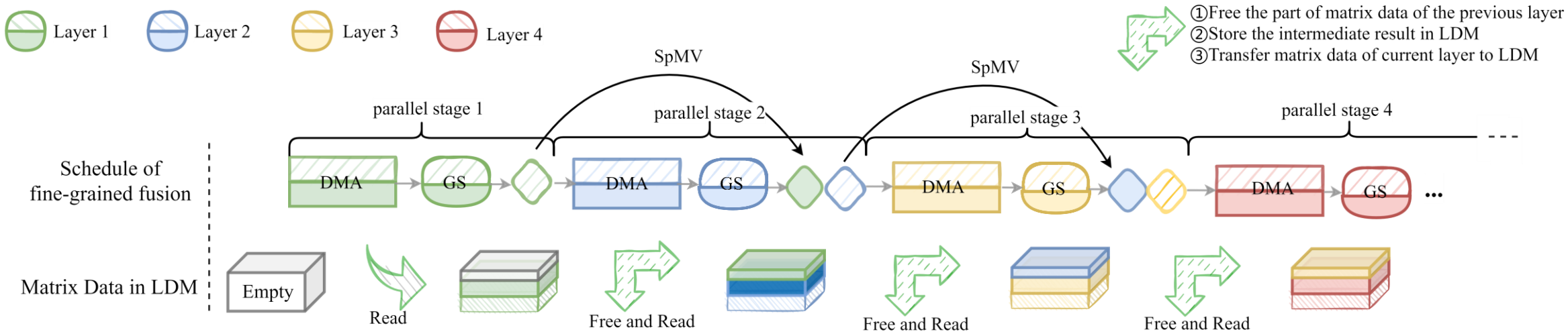
Optimization 2: Constraint-Aware Fine-Grained Fusion (cont'd)

$x_h \leftarrow x_h^* + (D_h + U_h)^{-1}(r_h - Ax_h^*)$ ▷ Backward Sweep

$r_h \leftarrow b_h - Ax_h$ ▷ Residual (SPMV)

Can be fused together to reduce memory access

- Split SpMV computation based on the diagonal of matrix

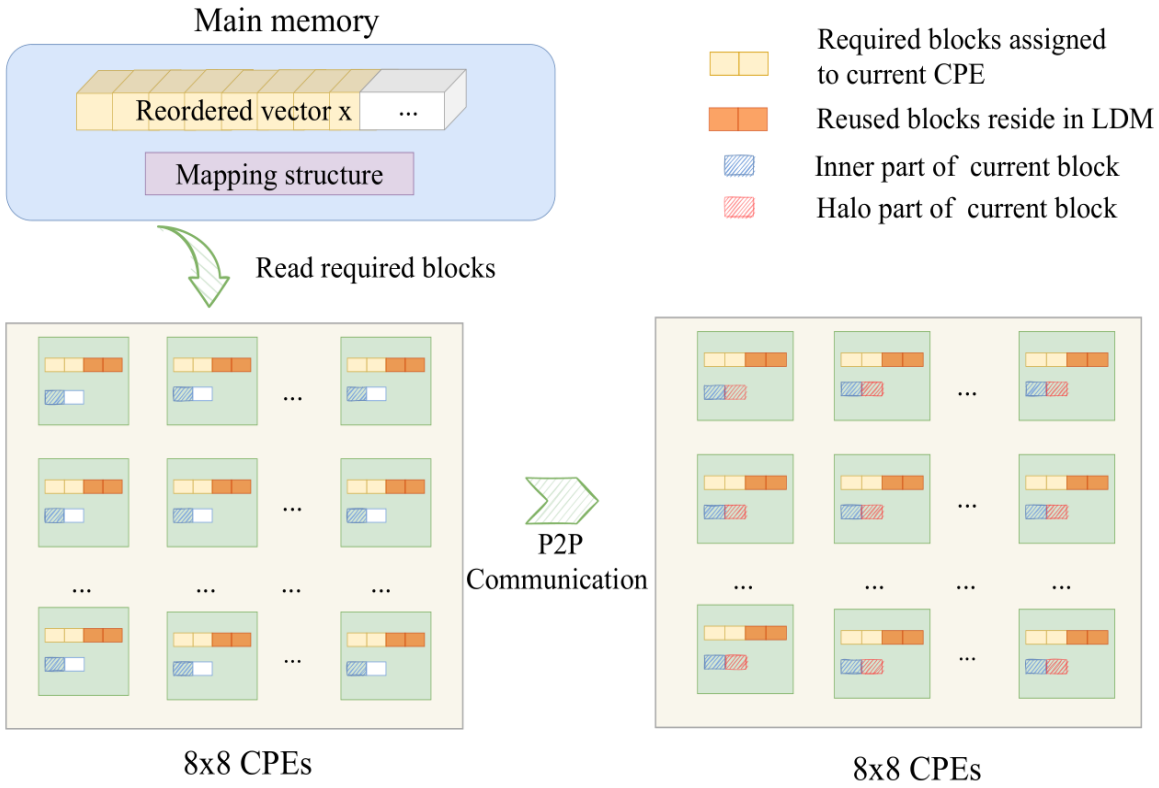


Fine-grained fusion

- Fine-grained SymGS-SpMV fusion relax the constraint of LDM capacity

Optimization 3: Low-Overhead Thread Collaboration

How to reduce the frequent data exchange of the solution vector x between the MPE and CPE cluster?



Data access based on the low-overhead thread collaboration approach for the solution vector x

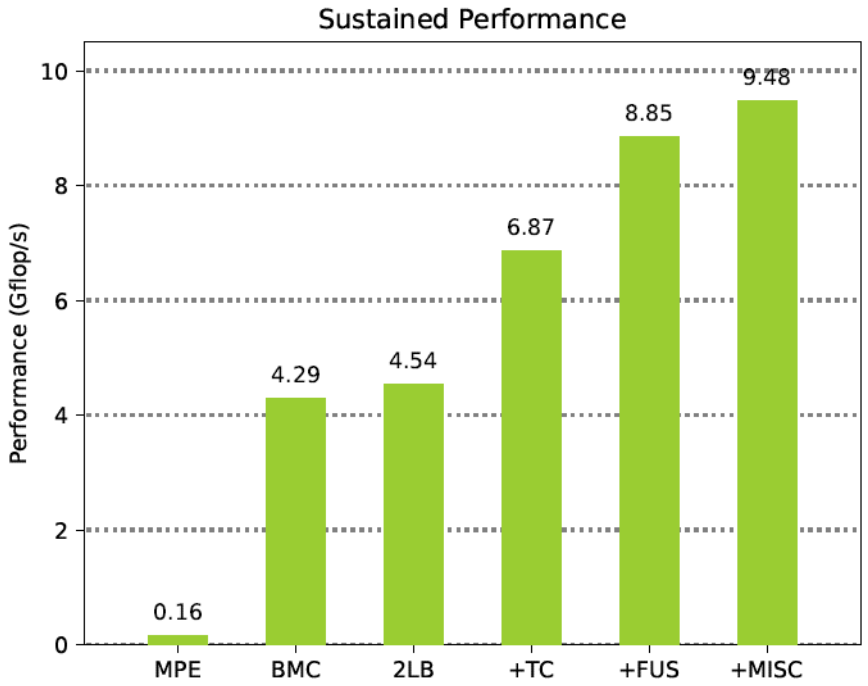
- The original indices at the subdomain level are converted to block level so that x can fit into the scope of the LDM
- The inner part can be transferred to the fast LDM via DMA directly
- Meanwhile, the halo part can be gathered with P2P communication via the universal address encoding
- Reused blocks can be kept resident in LDM to avoid data movement

Other Miscellaneous Optimizations

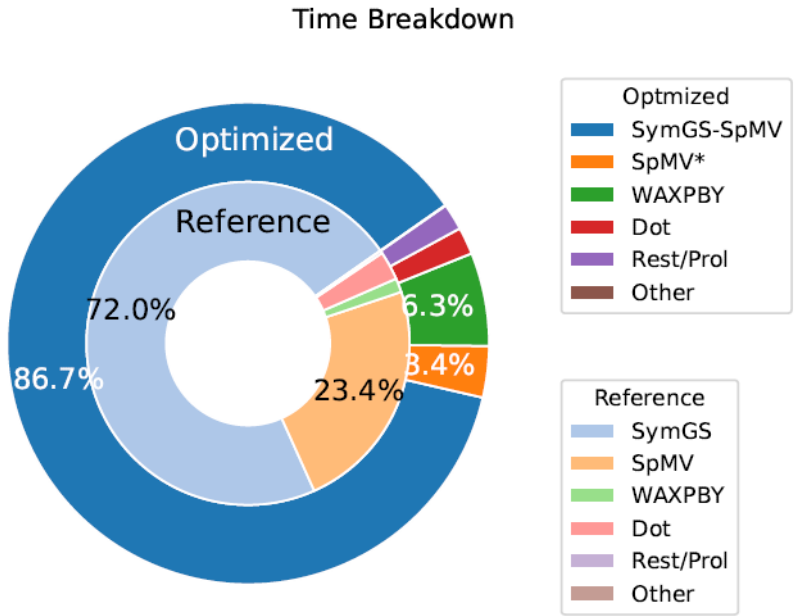
- Simplified ELLPACK format instead of the default CSR format to align the memory access
- Pin the data in the LDM at the coarsest layer
- SIMD vectorization is conducted automatically for Dot and WAXPBY
- Assign 128KB of the LDM as the data cache to improve locality for the restriction and prolongation
- Parallelize packing and unpacking of MPI messages between the neighbor processes
- Setup and pre-processing phase are also accelerated

Single-Process Experiments

Ablation tests with various optimization techniques (problem size: 256 x 128 x 256)

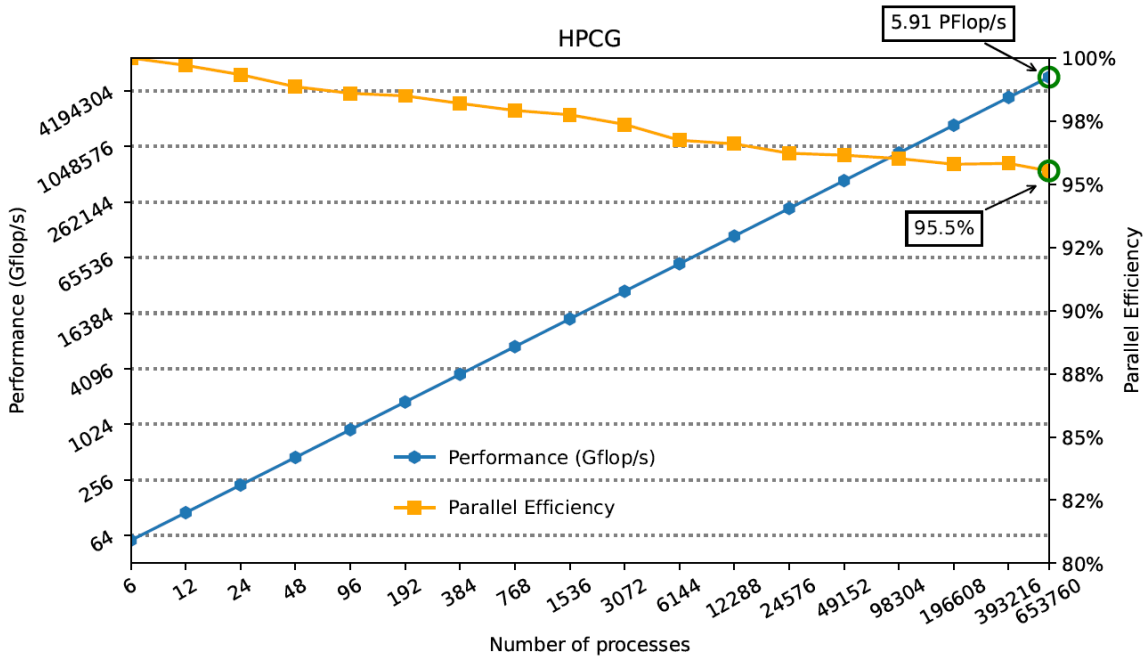


	Speedup	Note
MPE	baseline	reference implementation
BMC	26.8X	b = 32, c = 8 (56 iterations)
2LB	28.4X	m = 256, b = 16 (52 iterations)
+TC	42.9X	thread collaboration
+FUS	55.3X	fusion strategy
+MISC	**59.3X**	other techniques



Sustained 73.0% of the theoretical memory bandwidth

Scalability Tests



Weak scalability tests (subdomain size: 256 x 128 x 256)

- From 6 to 653,760 MPI processes (42 million cores)
- Maximum number of unknowns: 5.48 trillion
- Parallel efficiency: 95.5%
- HPCG performance: 5.91 PF

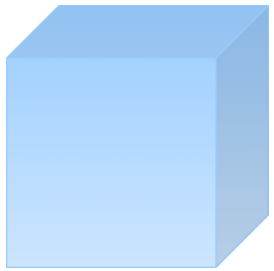
	Number of nodes	HPCG Performance (Tflop/s)	Total Memory Bandwidth (PB/s)	Performance Bandwidth Ratio (flop/byte)
Fugaku	158976	16004.5	162.8	0.0983
Summit	4608	2925.8	26.4	0.1108
Sunway TaihuLight	40960	480.8	5.59	0.0860
This work	108960	5907.1	33.5	0.1765

Outline of the Talk

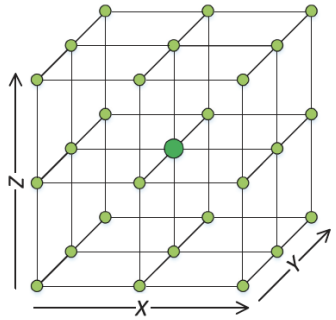
Beyond HPCG

What if we solve the same problem without fully obeying the specific rules of HPCG?

Major bottleneck of HPCG --- memory!



domain with grid



27-point stencil

sparse matrix

- Computation in a matrix free manner
  - avoid the explicit storage of the sparse matrix
  - problem size is enlarged now
  - make geometric information (domain, grid, stencil) available
- Memory bandwidth requirement is greatly alleviated
  - computational load is exposed
  - further optimizations are now possible
- Also possible to maintain data dependency of SymGS!

Optimization 1: Dependency Preserving Parallelization

How to maintain the original computing order with parallelism?

Classic method: level scheduling

- nodes on the same level can be processed in parallel
- poor data access locality of the solution vector

Our new dependency preserving parallelization method

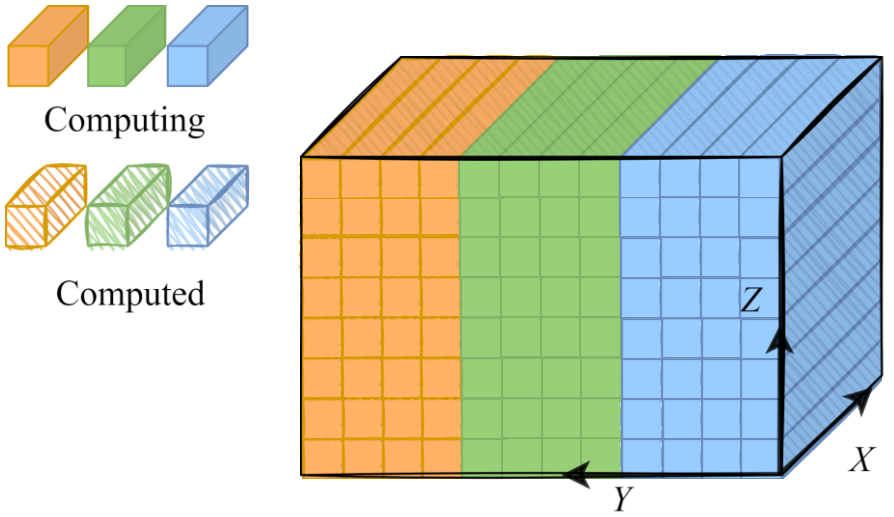
- 3D grid: $N_x \times N_y \times N_z$
- T be number of threads
- Task unit: a line along the x-axis direction of length N_x
- For each thread t $(t = 0, \dots, T-1)$
- Data range $[0, N_x) \times [\left\lceil \frac{N_y t}{T} \right\rceil, \left\lceil \frac{N_y (t+1)}{T} \right\rceil) \times [0, N_z)$
- Left counter F_t and right counter R_t

Workflow of a thread

- 1. Spin on counters before starting a new task via **union address** load instruction
- 2. Get required data from neighbor threads via **RMA** operation
- 3. Process this task
- 4. Update corresponding counter
- 5. Move on to another task

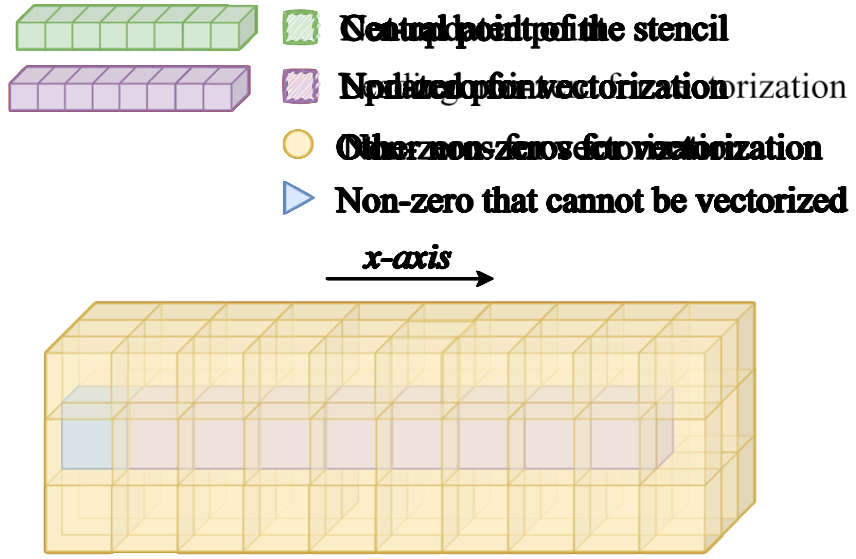
Example:

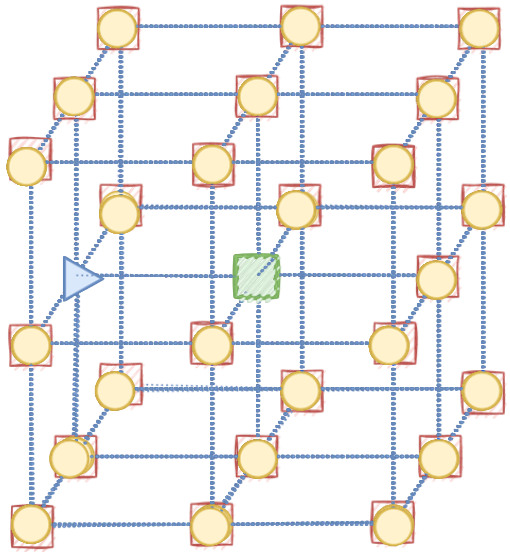
- 3 threads
- Grid size: 8 * 12 * 8



Schematic diagram of calculation sequence

Optimization 2: Dependency Preserving Vectorization





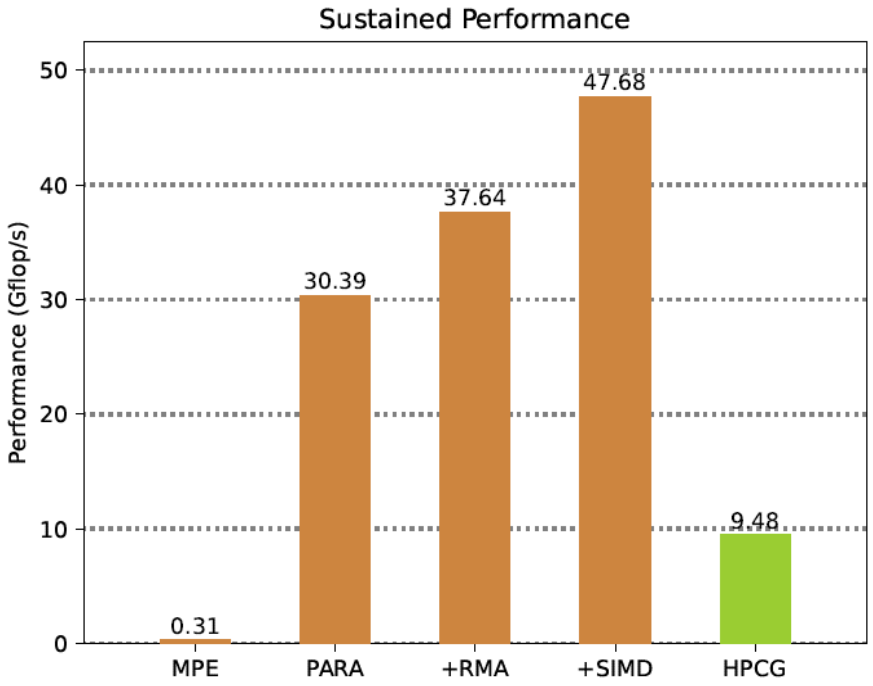
27-point Stencil

Vectorization of the SymGS computation for the 3D 27-point stencil

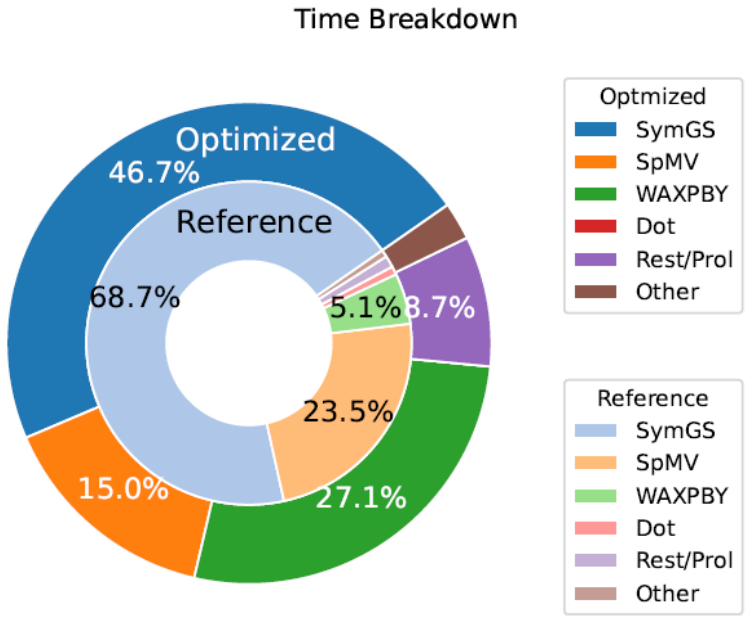
- Compute 25 out of the total 27 non-zero stencil points using SIMD unit
- Sequentially update the final results on the 8 central points
- Make full use of the 512bit SIMD unit and the data dependency is still preserved

Single-Process Performance

Ablation tests with various optimization techniques
(problem size: 512 x 256 x 512, i.e., 8X larger than that for the HPCG version)

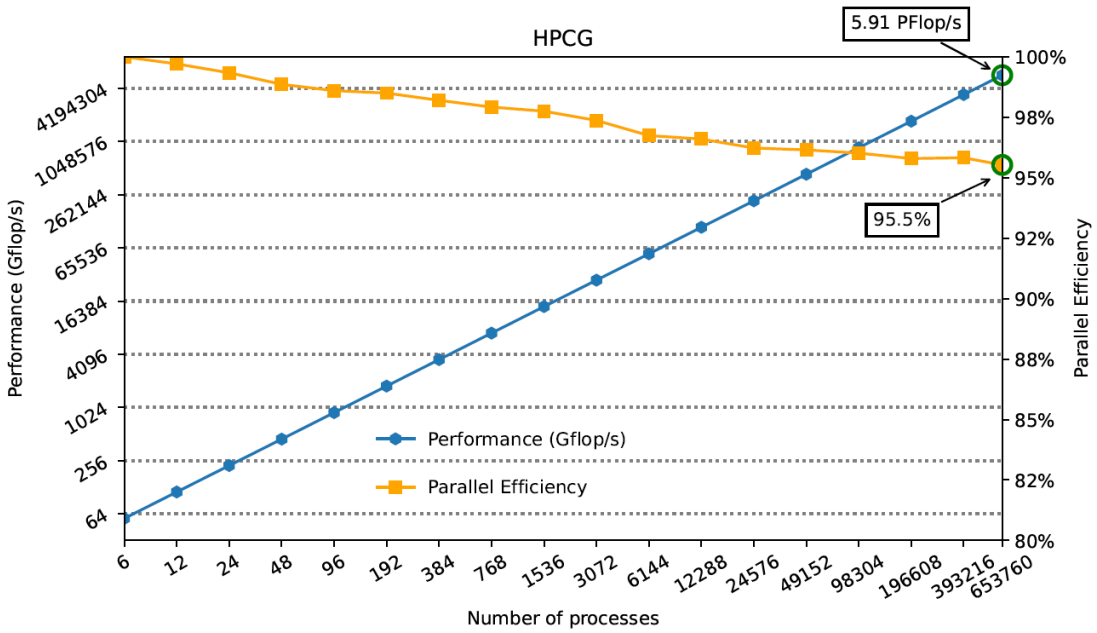


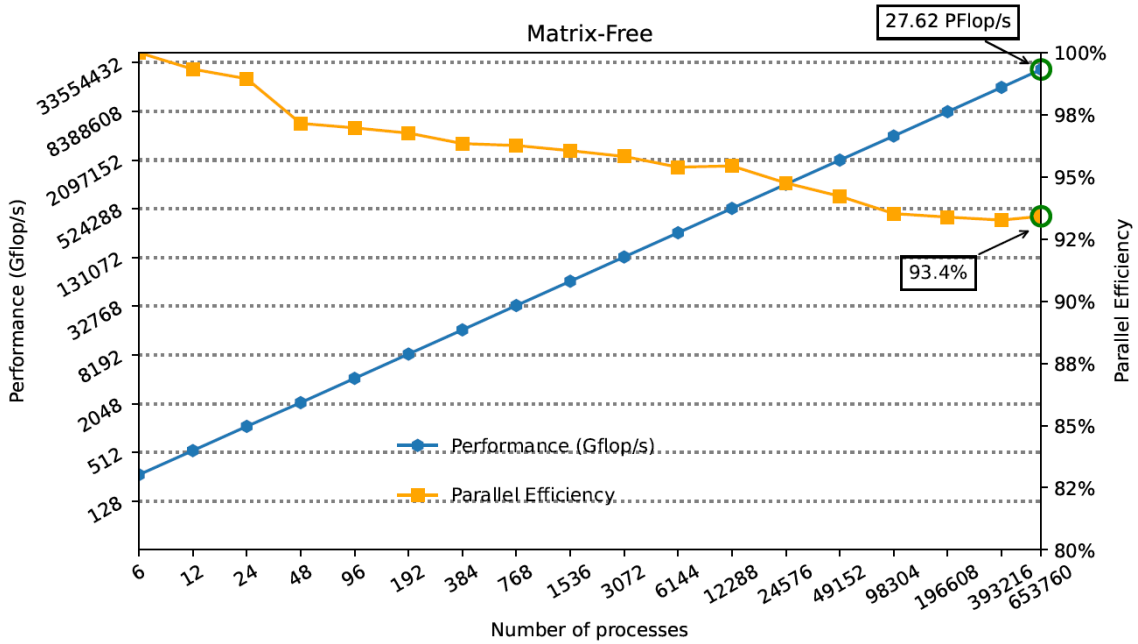
	Speedup	Note
MPE	baseline	reference implementation
PARA	98.0X	dependency preserving
+RMA	121.4X	transfer data using RMA
+SIMD	153.8X	vectorization



Speedup of 5.02X with respect to the optimized version of HPCG

Scalability Tests





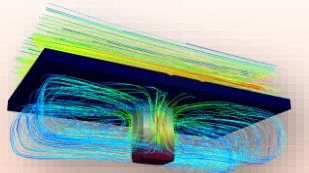
- Maintaining good scalability from 6 all the way to 653,760 MPI processes (42 million cores)
- Allowing larger problem due to memory saving (largest problem size: 43.8 trillion unknowns)
- Boosting sustained performance from 5.91 PF to 27.6 PF (4.67X increase)

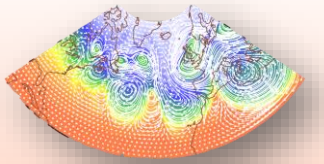
Outline of the Talk

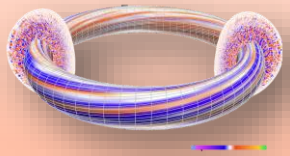
Conclusions and Outlooks

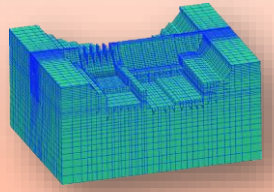
- Scaling the HPCG benchmark to over 42 million cores with >90% parallel efficiency
- Optimizations of HPCG Benchmark- 5.91 PF for 5.48 trillion unknowns
  - Fast Convergent Two-Level Blocking
  - Constraint-Aware Fine-Grained Fusion
  - Low-Overhead Thread Collaboration
- Optimizations of HPCG without obeying its rules - 27.6 PF for 43.8 trillion unknowns
  - Dependency Preserving Parallelization
  - Dependency Preserving Vectorization
- Methods presented may be generalized to other heterogeneous many-core systems with limited fast local memory and/or shared cache

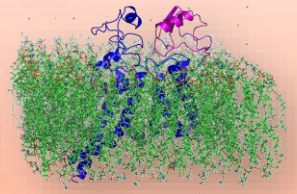
Serve as a test-drive experience and provide insights for deploying scientific and engineering applications.


Computational Fluid Dynamics


Atmospheric Dynamics


Tokamak Magnetic Restraint


Solid Mechanics


Molecular Dynamics