# TokaGLINT: A Scalable GPU-Tailored Implicit Solver for Full 3D Tokamak Electromagnetic Simulations

Zifan Yang, Haoyuan Zhang, Jialin Li, Wu Yuan, Xiazhen Liu,
Jian Zhang, Jianyuan Xiao, and **Shan Liang**

Computer Network Information Center, CAS  University of CAS
Tsinghua University  University of Science and Technology of China

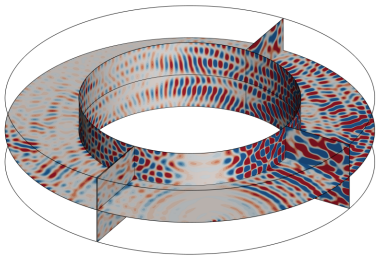
Presented by **Shan Liang**

SC26 • Chicago, Illinois, USA • November 15–20, 2026

# Background: Full 3D Tokamak Electromagnetic Simulations

**Long term electromagnetic field particle related physical simulation**, i.e., wave heating, current drive, and energetic-particle interactions with the background plasma in tokamaks

- **EM fully-kinetic model**
- **Symplectic structure-preserving PIC**
- In **explicit symplectic PIC**, the time step is constrained by the field CFL limit when the density is low or when a reduced ion-to-electron mass ratio is used
- CN-FDTD **field-implicit method**: no EM CFL limit, preserves the symplectic structure exactly, retains charge conservation, and integrates seamlessly (discretization and current deposition) with SymPIC



3D electric field excited by a point source in an EAST-referenced domain.

**SymPIC has been applied to full 3D tokamak simulations at extreme scale (SC21). The challenge is to upgrade its EM solver to an implicit formulation.**

# Target Linear System: CN-FDTD Curl-Curl Equation

- EM field equations (Faraday's and Ampère's laws)

$$\begin{cases} \dfrac{\partial \boldsymbol{E}}{\partial t} = \nabla \times \boldsymbol{B} - \boldsymbol{J}, \\ \dfrac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times \boldsymbol{E}, \end{cases}$$

- Applying the CN-FDTD scheme and eliminating $\boldsymbol{B}$ gives

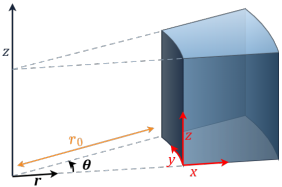
$$\left( \boldsymbol{I} + \frac{\Delta t^2}{4} \operatorname{curl}_d^{\mathrm{bw}} \operatorname{curl}_d^{\mathrm{fw}} \right) \boldsymbol{E}^{n+1} = \boldsymbol{b}$$

  $\operatorname{curl}_d$ - discrete curl operator under curvilinear coordinate system
- After some algebraic manipulations

$$\tilde{\boldsymbol{E}}^{n+1} + \tfrac{\Delta t^2}{4} \boldsymbol{Q} \tilde{\operatorname{curl}}_d^{\mathrm{bw}} \left( \boldsymbol{Q} \tilde{\operatorname{curl}}_d^{\mathrm{fw}} \tilde{\boldsymbol{E}}^{n+1} \right) = \boldsymbol{R}\boldsymbol{b}$$

  $\boldsymbol{Q}$ - diagonal, depends only on $\boldsymbol{x}$

cylindrical-coordinate formulation
(same as in SymPIC)
$x = r - r_0, y = r_0\theta, z = z$




Cartesian-like logical-grid curl
$\tilde{\operatorname{curl}}_d^{\mathrm{fw}} \boldsymbol{F}_{i,j,k} =$

$$\begin{bmatrix} \dfrac{\boldsymbol{F}_{z,i,j+1,k} - \boldsymbol{F}_{z,i,j,k}}{\Delta y} - \dfrac{\boldsymbol{F}_{y,i,j,k+1} - \boldsymbol{F}_{y,i,j,k}}{\Delta z} \\ \dfrac{\boldsymbol{F}_{x,i,j,k+1} - \boldsymbol{F}_{x,i,j,k}}{\Delta z} - \dfrac{\boldsymbol{F}_{z,i+1,j,k} - \boldsymbol{F}_{z,i,j,k}}{\Delta x} \\ \dfrac{\boldsymbol{F}_{y,i+1,j,k} - \boldsymbol{F}_{y,i,j,k}}{\Delta x} - \dfrac{\boldsymbol{F}_{x,i,j+1,k} - \boldsymbol{F}_{x,i,j,k}}{\Delta y} \end{bmatrix}$$

# TokaGLINT: Three Contributions

1. **GPU-tailored HASM preconditioner.** Co-design of domain decomposition framework and GPU-friendly fast subdomain solvers.

2. **Fast exact local solver.** Decouple unknowns based on discrete transformations; tensor-structured operations enable automated geometry-driven operator fusion and batching.

3. **Performance and application.** 90.1% weak and 53.9% strong scaling efficiency across more than 10,000 GPUs; 2.67× single-node speedup over HIP-enabled HYPRE; integrated into SymPIC and validated in full-3D EAST tokamak EM simulations.

# Pipelined BiCGStab with HASM preconditioning

**Pipelined BiCGStab + ASM preconditioning + exact local subdomain solves**

## Preconditioned Pipelined BiCGStab

**Input:** Matrix, RHS, Preconditioner
**Output:** Solution

1: Initialization
2: **loop**
3: **computation** local axpby, dot-product partial sums
4: Begin global sum allreduce 1
5: Apply HASM preconditioner
6: Apply SpMV with halo-exchange overlap
7: End global sum allreduce 1
8: **computation** local axpby, dot-product partial sums
9: Begin global sum allreduce 2
10: Apply HASM preconditioner
11: Apply SpMV with halo-exchange overlap
12: End global sum allreduce 2
13: **computation** local axpby
14: **end loop**

The global reductions are overlapped with SpMV and preconditioner applications.

- **Compute-bound**: HASM subdomain solves are compute-intensive.
- **Limited neighbor communication**: Intra-GPU inter-subdomain communication uses HBM.

The bottleneck caused by global reductions and overlapping-region communications in large-scale parallel execution can be substantially mitigated.

# Fast Exact Local Solver: Compact Operator

## ⓧ, ⓨ, ⓩ

$\forall \boldsymbol{T} = \{T_{ij}\} \in \mathbb{R}^{n\times n}$ , 3D field $\boldsymbol{q} = \{q_{ijk}\} \in \mathbb{R}^{n^3}$, define

$$\boldsymbol{T} \,ⓧ\, \boldsymbol{q} = T_{im} q_{mjk}, \quad \boldsymbol{T} \,ⓨ\, \boldsymbol{q} = T_{jm} q_{imk}, \quad \boldsymbol{T} \,ⓩ\, \boldsymbol{q} = T_{km} q_{ijm}$$

e.g. $\frac{\partial q}{\partial y}$, 3D structured grid, 1st-order forward-difference, $\Delta y = 1$

- SPMV $\boldsymbol{D}_y \cdot \boldsymbol{q}$
- GEMM $\boldsymbol{D}_{fw}$ ⓨ $\boldsymbol{q}$

$$\boldsymbol{D}_y = \begin{bmatrix} -1 & \overbrace{0 \;\cdots\; 0}^{n-1} & 1 & & & \\ & -1 & & 1 & & \\ & & \ddots & & \ddots & \\ & & & -1 & & 1 \\ & & & & & -1 \end{bmatrix}_{n^3\times n^3}$$

$$\boldsymbol{D}_{fw} = \begin{bmatrix} -1 & 1 & & \\ & -1 & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}_{n\times n}$$



F.Y.M. Wan, An in-core finite difference method for separable boundary value problems on a rectangle, *Studies in Applied Mathematics* 52(2):103–113, 1973.

# Fast Exact Local Solver: Mathematical Derivation

**Linear system with $3n^3$ unknowns**

$$\boldsymbol{E} + \frac{\Delta t^2}{4}\boldsymbol{Q}\,\tilde{\mathrm{curl}}_d^{\mathrm{bw}}\left(\boldsymbol{Q}\,\tilde{\mathrm{curl}}_d^{\mathrm{fw}}\boldsymbol{E}\right) = \boldsymbol{b}, \quad \boldsymbol{E} = (\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)^T, \quad \boldsymbol{b} = (\boldsymbol{b}_x, \boldsymbol{b}_y, \boldsymbol{b}_z)^T$$

**Compact tensor form**

$$\begin{cases} \left(\beta\boldsymbol{I} - \boldsymbol{H}^2 \otimes \boldsymbol{D}_{bw}\boldsymbol{D}_{fw} \,ⓨ\, -\boldsymbol{D}_{bw}\boldsymbol{D}_{fw} ⓩ\right)\boldsymbol{e}_x + \boldsymbol{H}^2\boldsymbol{D}_{fw} \otimes \boldsymbol{D}_{bw} \,ⓨ\, \boldsymbol{e}_y + \boldsymbol{D}_{bw} ⓩ \boldsymbol{D}_{fw} \otimes \boldsymbol{e}_z = \boldsymbol{b}_x \\ \boldsymbol{H}^{-1}\boldsymbol{D}_{bw}\boldsymbol{H} \otimes \boldsymbol{D}_{fw} \,ⓨ\, \boldsymbol{e}_x + \left(\beta\boldsymbol{I} - \boldsymbol{D}_{bw}\boldsymbol{D}_{fw} ⓩ -\boldsymbol{H}^{-1}\boldsymbol{D}_{bw}\boldsymbol{H}\boldsymbol{D}_{fw} \otimes\right)\boldsymbol{e}_y + \boldsymbol{D}_{bw} ⓩ \boldsymbol{D}_{fw} \,ⓨ\, \boldsymbol{e}_z = \boldsymbol{b}_y \\ \boldsymbol{H}\boldsymbol{D}_{bw}\boldsymbol{H}^{-1} \otimes \boldsymbol{D}_{fw} ⓩ \boldsymbol{e}_x + \boldsymbol{H}^2 \otimes \boldsymbol{D}_{bw} \,ⓨ\, \boldsymbol{D}_{fw} ⓩ \boldsymbol{e}_y + \left(\beta\boldsymbol{I} - \boldsymbol{H}\boldsymbol{D}_{bw}\boldsymbol{H}^{-1}\boldsymbol{D}_{fw} \otimes -\boldsymbol{H}^2 \otimes \boldsymbol{D}_{bw}\boldsymbol{D}_{fw} ⓨ\right)\boldsymbol{e}_z = \boldsymbol{b}_z \end{cases}$$

$\boldsymbol{H}$ - diagonal, depends only on x. $\boldsymbol{D}_{fw} = -\boldsymbol{D}_{bw}^T$ - 1D difference operator

**Singular decomposition** $\boldsymbol{D}_{fw} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$

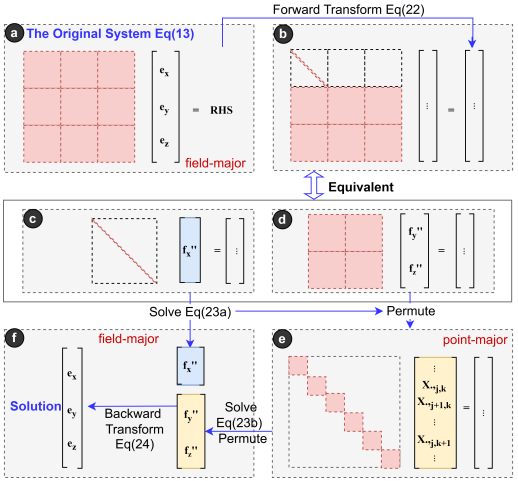
$$\begin{pmatrix} \boldsymbol{V} \,ⓨ\, \boldsymbol{V} ⓩ & & \\ & \boldsymbol{U} \,ⓨ\, \boldsymbol{V} ⓩ & \\ & & \boldsymbol{V} \,ⓨ\, \boldsymbol{U} ⓩ \end{pmatrix} \begin{pmatrix} \beta\boldsymbol{I} + \boldsymbol{H}^2 \otimes \boldsymbol{S}^2 \,ⓨ\, +\boldsymbol{S}^2 ⓩ & -\boldsymbol{H}^2\boldsymbol{D}_{fw} \otimes \boldsymbol{S} ⓨ & -\boldsymbol{D}_{fw} \otimes \boldsymbol{S} ⓩ \\ \boldsymbol{H}_1 \otimes \boldsymbol{S} ⓨ & \beta\boldsymbol{I} + \boldsymbol{S}^2 ⓩ - \boldsymbol{H}_1\boldsymbol{D}_{fw} \otimes & -\boldsymbol{S} ⓨ \boldsymbol{S} ⓩ \\ \boldsymbol{H}_2 \otimes \boldsymbol{S} ⓩ & -\boldsymbol{H}^2 \otimes \boldsymbol{S} ⓨ \boldsymbol{S} ⓩ & \beta\boldsymbol{I} - \boldsymbol{H}_2\boldsymbol{D}_{fw} \otimes + \boldsymbol{H}^2 \otimes \boldsymbol{S}^2 ⓨ \end{pmatrix} \begin{pmatrix} \boldsymbol{V}^T \,ⓨ\, \boldsymbol{V}^T ⓩ \boldsymbol{e}_x \\ \boldsymbol{U}^T \,ⓨ\, \boldsymbol{V}^T ⓩ \boldsymbol{e}_y \\ \boldsymbol{V}^T \,ⓨ\, \boldsymbol{U}^T ⓩ \boldsymbol{e}_z \end{pmatrix} = \boldsymbol{b}$$

$\boldsymbol{S}$ diagonal, no cross node dependencies $\implies$ system is decoupled in y and z directions
$\implies$ $n^2$ linear systems, each with $3n$ unknowns

$$e.g., \quad \Sigma_{i=1}^{n}\left(\alpha_{*i}\boldsymbol{f}_{xijk} + \beta_{*i}\boldsymbol{f}_{yijk} + \gamma_{*i}\boldsymbol{f}_{zijk}\right) = \boldsymbol{h}_*, \quad j,k = 1\cdots n, \quad *= x,y,z$$

$\beta\boldsymbol{I} + \boldsymbol{H}^2 \otimes \boldsymbol{S}^2 \,ⓨ\, +\boldsymbol{S}^2 ⓩ$ is diagonal $\implies$ $n^2$ linear systems, each with $2n$ unknowns

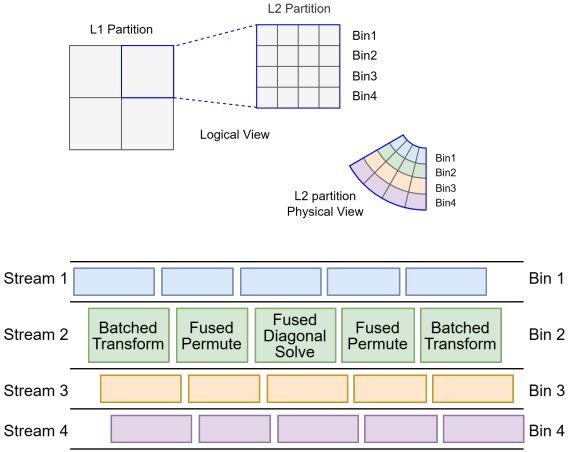
# Fast Exact Local Solver: Computational Path




- Forward transform: original physical-space system (a) → feature-space system (b)
- Reduce (b) to two equivalent subsystems: diagonal subsystem (c) + two-component subsystem coupled along $x$-lines (d)
- Field-major to point-major permutation (d) → (e)
- Independent block solves (d)
- Backward transform, gives solution (f)

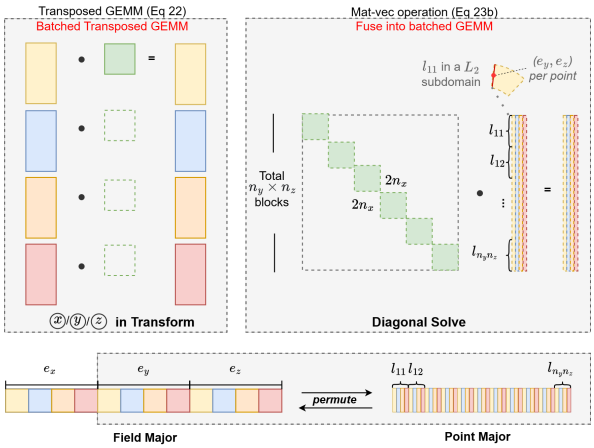
Memory footprint & Computational complexity $\mathcal{O}(N^{4/3})$

# HASM: Hierarchical Domain Decomposition




- **L1:** each overlapping subdomain is assigned to one MPI process and one GPU.
- **L2:** each L1 subdomain is further split into smaller overlapping subdomains.
- L2 subdomains at the same radial layers share metric tensors and are grouped into one bin to facilitate operator fusion and batching.
- Different bins are streamed concurrently.

# GPU Implementation: Fusion, Batching, and Layout




- Transforms natively adopt GEMM form and execute as batched GEMM within each bin.

$$\boldsymbol{V}^T \text{ⓨ } \boldsymbol{V}^T \text{ⓩ } \boldsymbol{e}_x$$

- Same-position block solves across L2 subdomains are fused into one GEMM.

$$\Sigma_{i=1}^{n} \left(\beta_{*i} \boldsymbol{f}_{yijk} + \gamma_{*i} \boldsymbol{f}_{zijk}\right) = \boldsymbol{h}_*$$

$$j, k = 1 \cdots n, \quad * = y, z$$

- A customized point-major layout makes the fusion possible.

$$(h_{y1}, h_{z1}, h_{y2}, h_{z2}, h_{y3}, h_{z3} \cdots)$$

# Experimental Platform and Setup

## Compute node

- 8 GPUs; 320 SIMD units per GPU
- 32.7 TFLOPS theoretical FP64 per GPU
- 64 GB HBM; 1.8 TB/s peak bandwidth
- High-speed intra-node accelerator links

## Network and software

- $4 \times 400$ Gbps RDMA-capable links
- HIP-compatible GPU kernels
- GPU-aware MPI
- Clang 17.0.0; Open MPI 5.0.3

Overlap layers $l = 3$; representative L1 sizes $64^3$–$128^3$ per GPU. Reported weak/strong times are averaged over 100 complete solves to convergence.

# Preconditioning Effect

## Comparison with other preconditioning methods

| $\Delta t$ | 1.0 | 2.0 | 4.0 | 8.0 | 16.0 |
|---|---|---|---|---|---|
| Baseline NON_PRE | 16 | 34 | 60 | 102 | 167 |
| JACOBI (PETSc) | 16 | 34 | 61 | 96 | 174 |
| SOR (PETSc) | 12 | 25 | 51 | 82 | 148 |
| ISAI (HYPRE) | 41 | MAX | 24 | 46 | 100 |
| ILUT (HYPRE) | 15 | 31 | 115 | MAX | DIV |
| AMG (HYPRE) | 5 | 9 | 18 | 37 | 63 |
| TokaGLINT's HASM ($l = 1$) | 3 | 5 | 8 | 12 | 19 |
| TokaGLINT's HASM ($l = 2$) | 2 | 3 | 6 | 9 | 14 |
| TokaGLINT's HASM ($l = 3$) | 2 | 3 | 4 | 7 | 11 |
| TokaGLINT's HASM ($l = 4$) | 2 | 2 | 4 | 7 | 10 |

- Fewer iterations than the general-purpose preconditioners, especially for large $\Delta t$
- Improved convergence as $l$ increases
- Trade-off between convergence and per-iteration cost ($l = 3$ delivers the optimal overall performance)

## Weak-scaling convergence

| Accelerators | L1 count | BiCGStab Steps |
|---|---|---|
| 8 | $2 \times 2 \times 2$ | 7.40 |
| 64 | $4 \times 4 \times 4$ | 7.90 |
| 512 | $8 \times 8 \times 8$ | 7.75 |
| 4096 | $16 \times 16 \times 16$ | 8.00 |

The results demonstrate robust and scalable convergence behavior for large-scale parallel simulations, which provides a solid basis for applying the proposed method to ultra-large-scale EM problems.

# Why Hierarchy Matters

**L2 decomposition ablation**

| L2 count | L2 size | $10^{-8}$ | $10^{-10}$ | $10^{-12}$ |
|---|---|---|---|---|
| $4 \times 4 \times 4$ | $16^3$ | 49.8/6 | 63.8/8 | 74.6/9 |
| $2 \times 2 \times 2$ | $32^3$ | 63.0/6 | 71.7/7 | 92.1/9 |
| $1 \times 1 \times 1$ | $64^3$ | 209.0/6 | 238.9/7 | 302.7/9 |

- When solving with a single large subdomain yields poor efficiency.
- The L2 domain decomposition substantially boosts solving efficiency, with iteration counts staying consistent.
- The one iteration discrepancy can be attributed to different convergence paths taken to reach the prescribed tolerance.

# Performance versus HIP-Enabled HYPRE 2.32

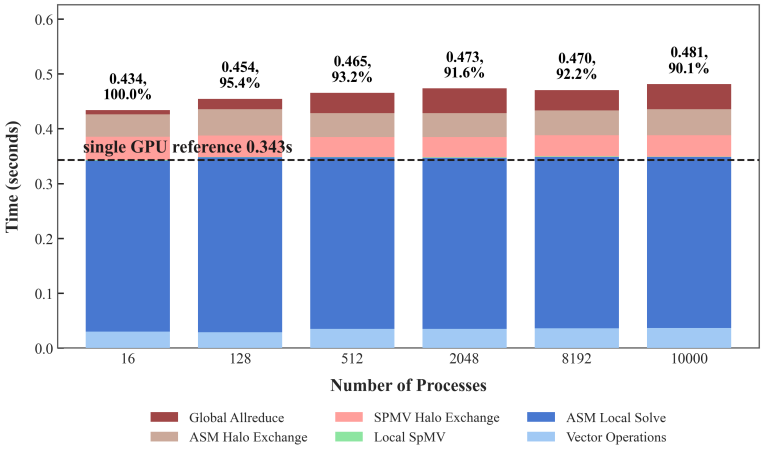
| Accelerators | 8 | 64 | 256 | 512 |
|---|---|---|---|---|
| HYPRE | 212.38/125 | 224.02/122 | 235.10/122 | 257.04/122 |
| TokaGLINT | 79.67/8 | 81.96/8 | 85.09/8 | 84.79/8 |
| Speedup | 2.67× | 2.73× | 2.76× | 3.03× |

Each entry is total solve time (ms) / iterations (count).

Baseline: unpreconditioned BiCGStab in HIP-enabled HYPRE.

TokaGLINT achieves a speedup of 2.67× on a single node (8 accelerators), and the speedup ratio further increases as the problem scale expands.

# Weak Scaling to 10,000 Accelerators

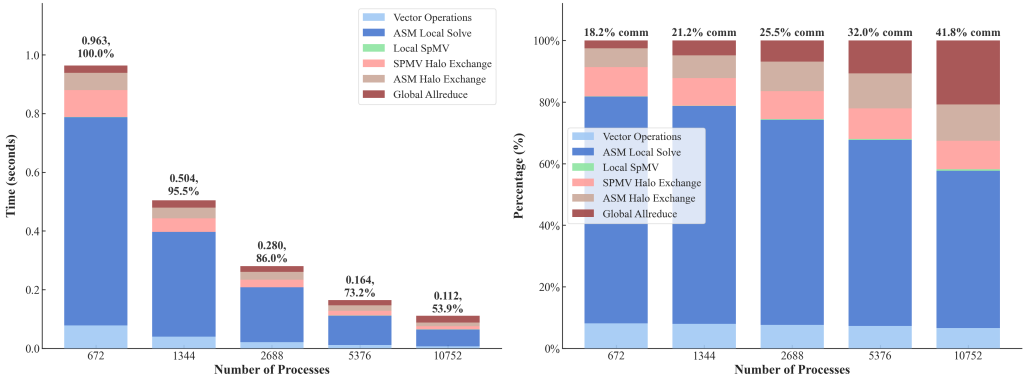



**625×**
scale increase

**90.1%**
parallel efficiency

The 16-accelerator case (2 nodes) is used as the scaling baseline because a single-GPU run has no L1 decomposition or inter-GPU communication.

# Strong Scaling Beyond 10,000 Accelerators

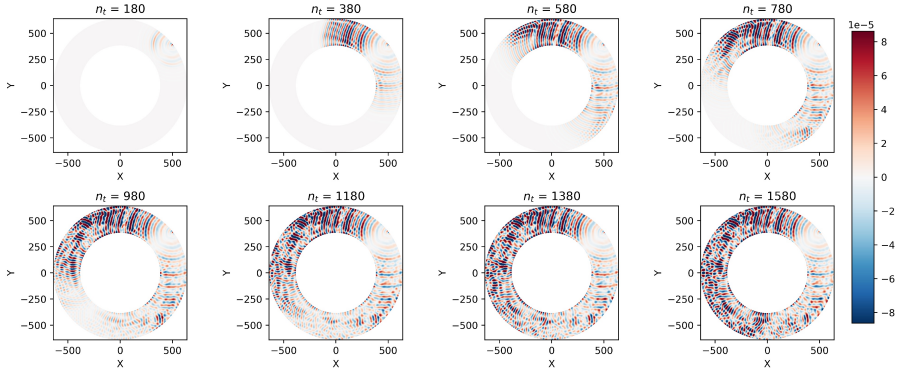



**672** $\rightarrow$ **10,752** accelerators  53.9% parallel efficiency  6 iterations in every configuration

- Thanks to the communication-hiding BiCGStab solver, global Allreduce operations are overlapped with preconditioning and SpMV, effectively reducing communication overhead.
- The observed efficiency loss mainly comes from the growing Allreduce proportion and reduced local computation time.

# SymPIC Validation: Wave Propagation

**Wave-excitation case**: wave source is located at $x = n_x \Delta x, y = 102\Delta y, z = n_z \Delta z/2$

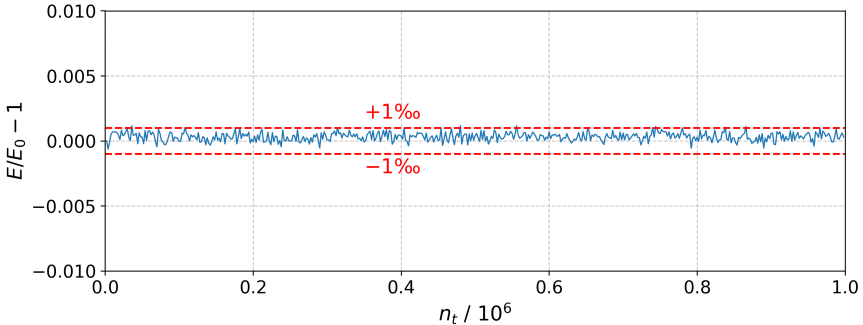



$E_y$ on the $z$-midplane

The evolution clearly shows EM-wave propagation inside the toroidal domain.

# Long-Time Numerical Fidelity

**3D magnetized electron toroidal plasma simulation**: $\boldsymbol{B} = B_0\vec{\boldsymbol{j}}$




$10^6$

time steps

$\pm 1‰$

energy variation

It verifies the advantage of the present field-implicit symplectic PIC scheme for cylindrical coordinates.

# Conclusion

- TokaGLINT is a highly scalable, hardware-aware linear solver for the CN-FDTD discretization of Maxwell's equations in curvilinear coordinates.
- A hierarchical domain decomposition framework is coupled with tensor-structured fast subdomain solvers tailored for curvilinear coordinate systems.
- It delivers a 2.67× single-node speedup, 90.1% weak-scaling efficiency, and 53.9% strong-scaling efficiency across more than 10,000 GPUs.
- Deployment within SymPIC enables efficient implicit EM field solutions while preserving rigorous numerical fidelity for large-scale tokamak simulations.

Thank you !