Karlsruhe Institute of Technology

Technical University of Munich

Scalable distributed preconditioners in Ginkgo

Preconditioning 2024, Atlanta, June 10th, 2024

Pratik Nayak on behalf of the Ginkgo team

Outline

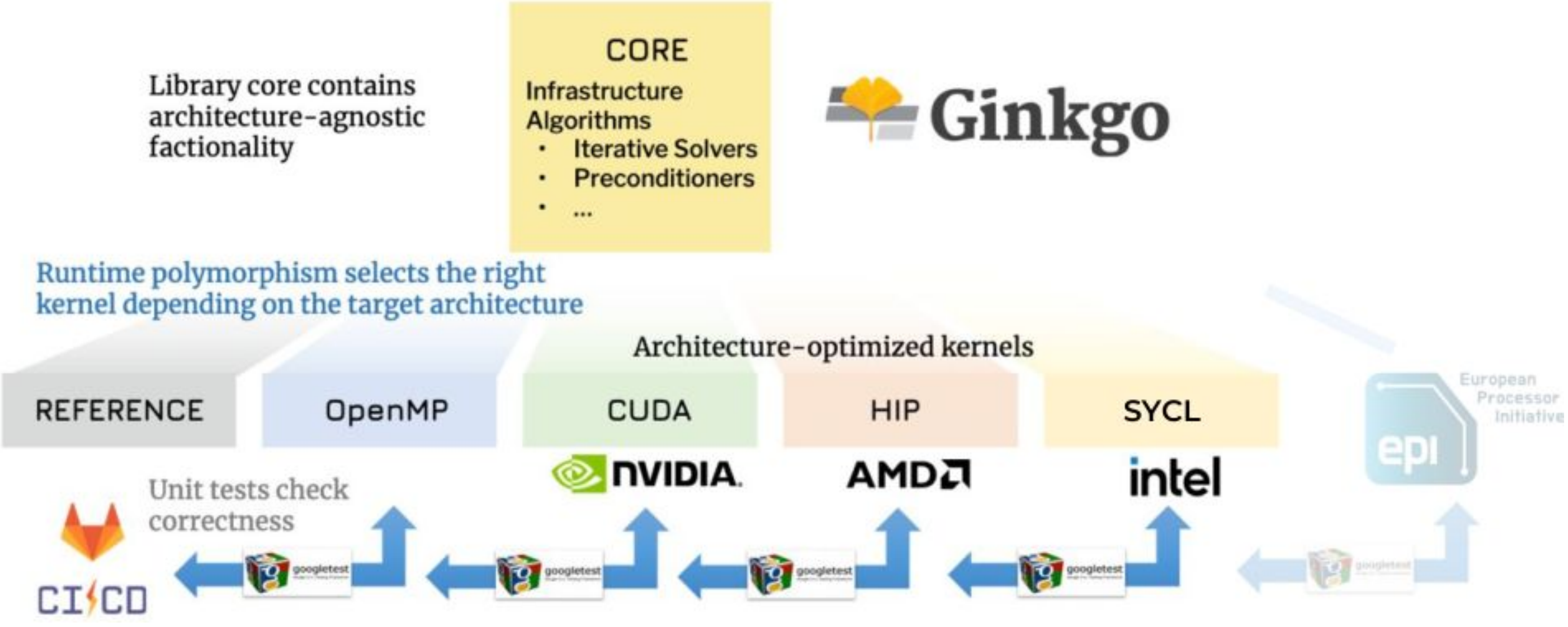
- Ginkgo design philosophy
- Composability
- (Distributed) preconditioning in Ginkgo
- Summary and future work

Ginkgo[1]

- A high-performance numerical linear algebra library
- Open-source, modern C++ (currently C++14 standard)
- Provides high-performant basic building blocks: SpMV, SpGeMM, etc
- Also features linear solvers, preconditioners and many utilities that ease application integration.
- Support for different hardware backends, using the vendor-native programming models.

[1]https://github.com/ginkgo-project/ginkgo

Ginkgo design[1]



[1]Ginkgo: A high performance numerical linear algebra library, JOSS, Aug 2020

Ginkgo: Features

	FUNCTIONALITY	OMP	CUDA	HIP	DPC++
Basic	SpMV	MPI	MPI	MPI	MPI
	SpMM	MPI	MPI	MPI	MPI
	SpGeMM	Single-GPU	Single-GPU	Single-GPU	Single-GPU
Krylov solvers	BiCG	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	BiCGSTAB	MPI	MPI	MPI	MPI
	CG	MPI	MPI	MPI	MPI
	CGS	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	GCR	MPI	MPI	MPI	MPI
	GMRES	MPI	MPI	MPI	MPI
	FCG	MPI	MPI	MPI	MPI
	FGMRES	MPI	MPI	MPI	MPI
	IR	MPI	MPI	MPI	MPI
	IDR	Single-GPU	Single-GPU	Single-GPU	Single-GPU
Preconditioners	Block-Jacobi	MPI	MPI	MPI	MPI
	ILU/IC	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	Parallel ILU/IC	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	Parallel ILUT/ICT	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	ISAI	Single-GPU	Single-GPU	Single-GPU	Single-GPU

✓ *MPI Support* ✓ *Single-GPU Support*

	FUNCTIONALITY	OMP	CUDA	HIP	DPC++
Batched	Batched BiCGSTAB	MPI	MPI	MPI	MPI
	Batched CG	MPI	MPI	MPI	MPI
	Batched GMRES	MPI	MPI	MPI	MPI
	Batched ILU	MPI	MPI	MPI	MPI
	Batched ISAI	MPI	MPI	MPI	MPI
	Batched Block-Jacobi	MPI	MPI	MPI	MPI
AMG	AMG preconditioner	MPI	MPI	MPI	MPI
	AMG solver	MPI	MPI	MPI	MPI
	Parallel Graph Match	MPI	MPI	MPI	MPI
Sparse direct	Symbolic Cholesky	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	Numeric Cholesky	Single-GPU	Single-GPU	Single-GPU	
	Symbolic LU	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	Numeric LU	Single-GPU	Single-GPU	Single-GPU	
	Sparse TRSV	Single-GPU	Single-GPU	Single-GPU	
Utilities	On-Device Matrix Assembly	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	MC64/RCM reordering	Single-GPU			
	Wrapping user data	MPI	MPI	MPI	MPI
	Logging	Single-GPU	Single-GPU	Single-GPU	Single-GPU
	PAPI counters	Single-GPU	Single-GPU	Single-GPU	Single-GPU

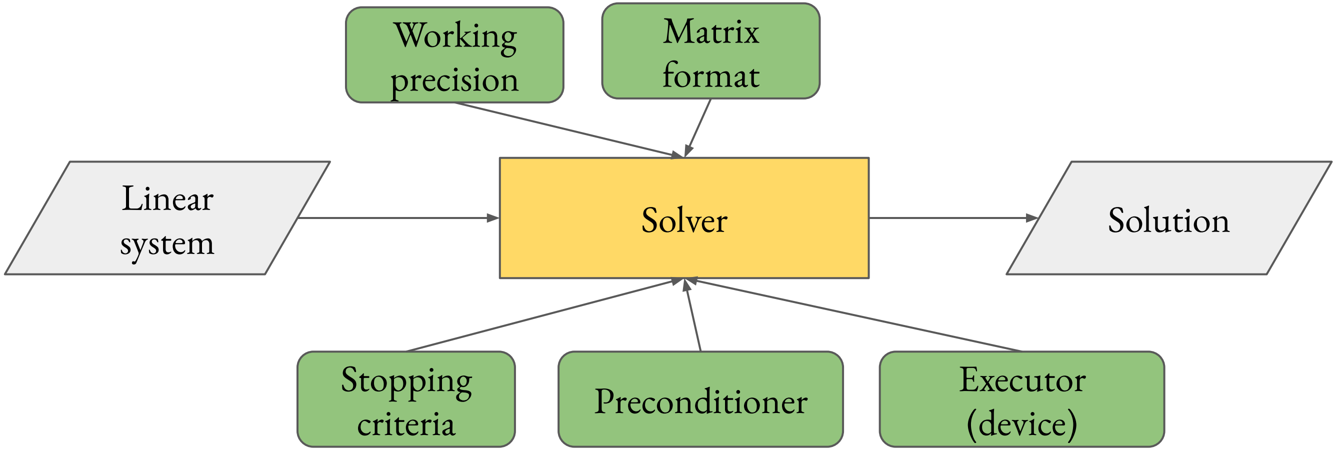
✓ *MPI Support* ✓ *Single-GPU Support*

Ginkgo example

```
using cg = gko::solver::Cg<>;
using iter = gko::stop::Iteration;
using residual_norm = gko::stop::ResidualNorm<>;
auto cg_factory  =
              cg::build()
                  .with_criteria(
                      iter::build()
                          .with_max_iters(20u).on(exec),
                      residual_norm::build()
                          .with_reduction_factor(tolerance)
                  .on(exec))
              .on(exec);
// generate the solver with an input LinOp.
auto cg_solver = cg_factory->generate(system_matrix);
```

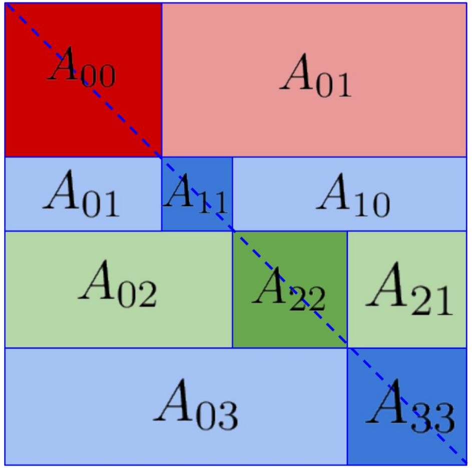
[1]Ginkgo: A high performance numerical linear algebra library, JOSS, Aug 2020

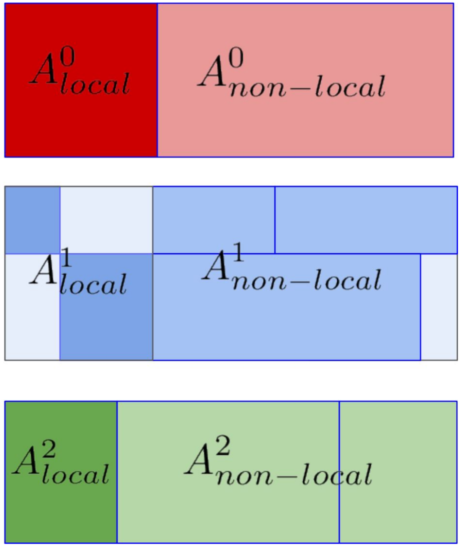
Ginkgo: Extreme composability



[1]Ginkgo: A high performance numerical linear algebra library, JOSS, Aug 2020

Ginkgo distributed matrix storage and SpMV



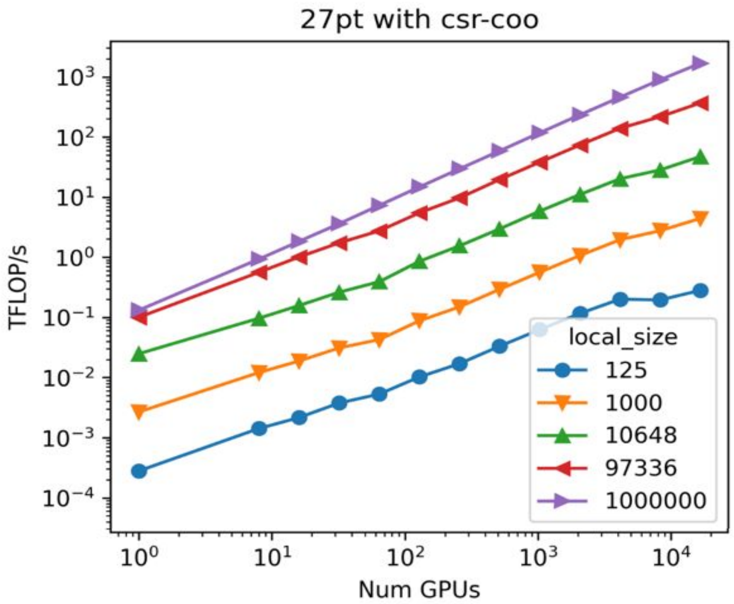


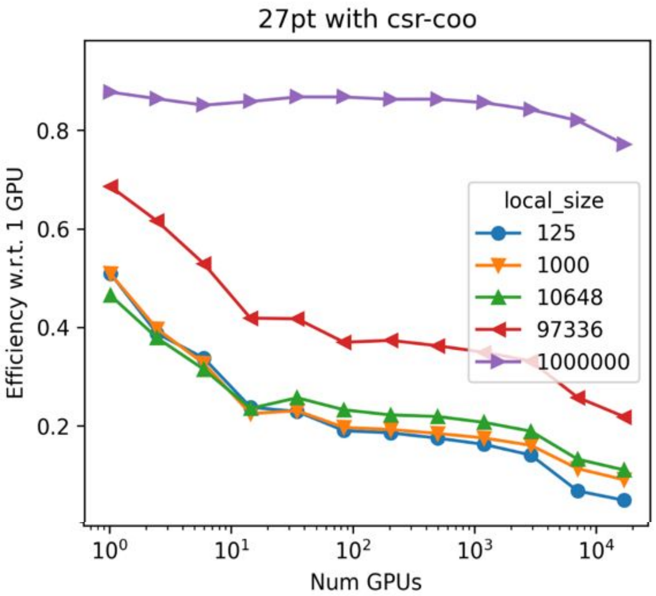
Algorithm 1 GINKGO's distributed sparse matrix vector product

1: **local_x** $\leftarrow x_{local}$, **local_b** $\leftarrow b_{local}$
2: $A_{local} \rightarrow$ APPLY(**local_b**, **local_x**) ▷ Local SpMV
3: $b \rightarrow$ GATHER_NON_LOCAL(**buffer**) ▷ Communicate the non-local vector
4: x_{local} += $A_{non-local} \rightarrow$ APPLY(**buffer**, **local_x**) ▷ Non-local SpMV

Weak scaling: SpMV on Frontier

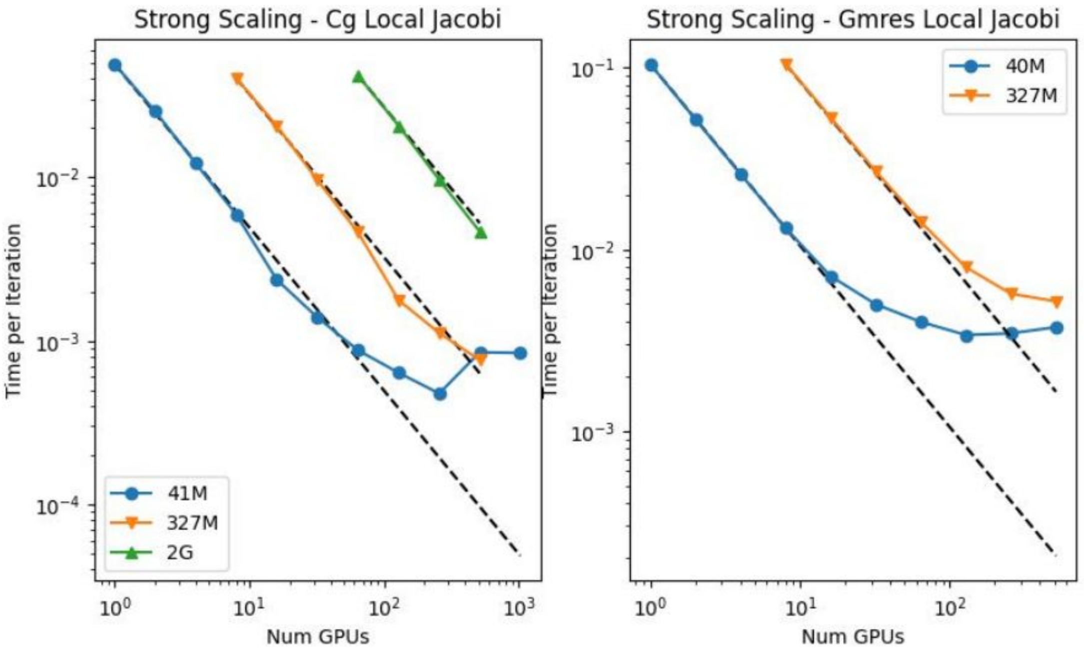
Cray MPICH + AMD MI250X on 16k GCDs





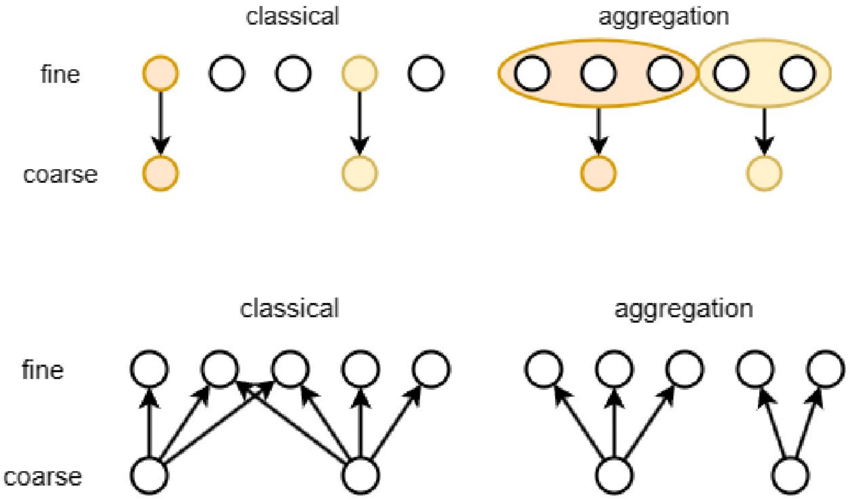
Weak scaling: problem size increases with parallel resources

Ginkgo distributed solver performance



Strong scaling on Frontier with
distributed Schwarz preconditioner.

Ginkgo distributed multigrid with PGM



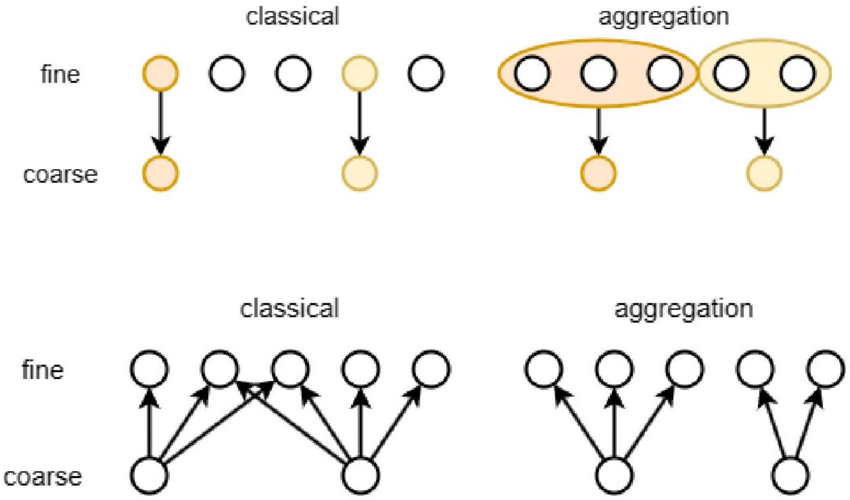
- Each rank does a local parallel graph match with its local and non-local matrices, based on strongest neighbor

$$|A_{i,j}| \geq |A_{i,k}|, \ \forall k \neq i,$$

- Aggregation indices are exchanged.
- Aggregation in local ranks.

[1]Portable Mixed precision Algebraic Multigrid on GPUs, Tsai, 2024

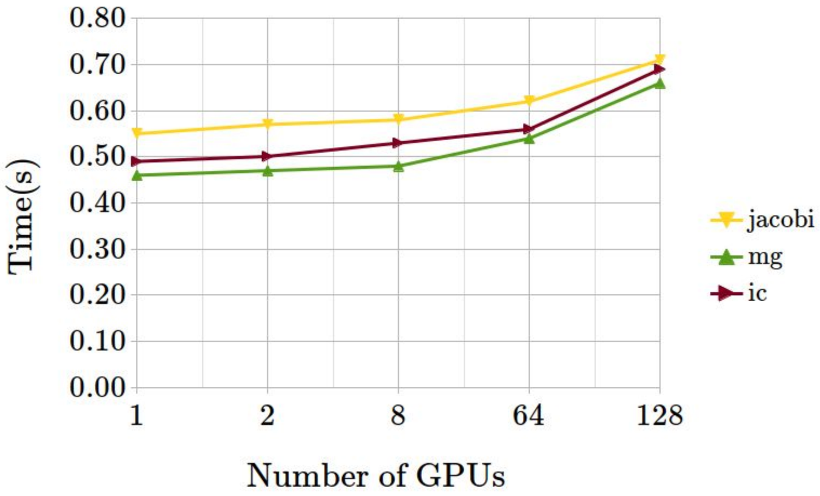
Ginkgo distributed multigrid with PGM

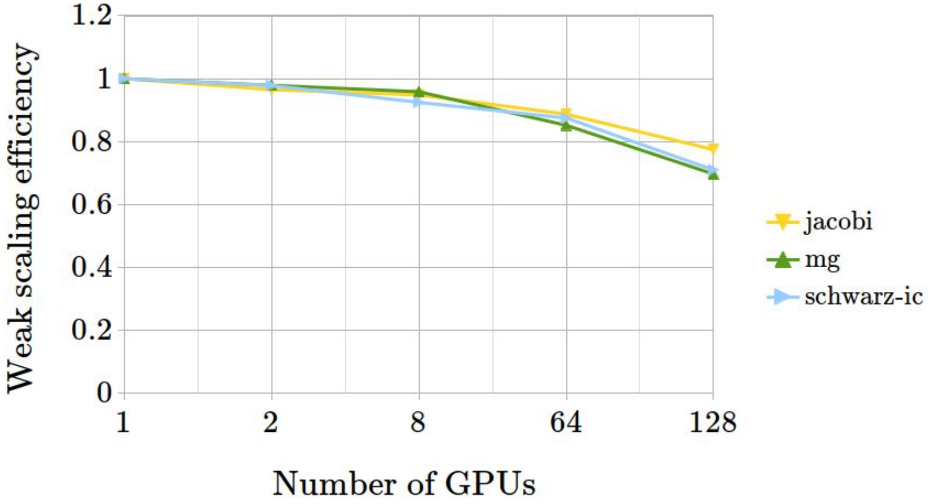


- Number of levels
- Post-, pre- and mid- smoothers
- Precision choice for vectors (IEEE double, float, half)
- Choice of coarse solver

[1]Portable Mixed precision Algebraic Multigrid on GPUs, Tsai, 2024

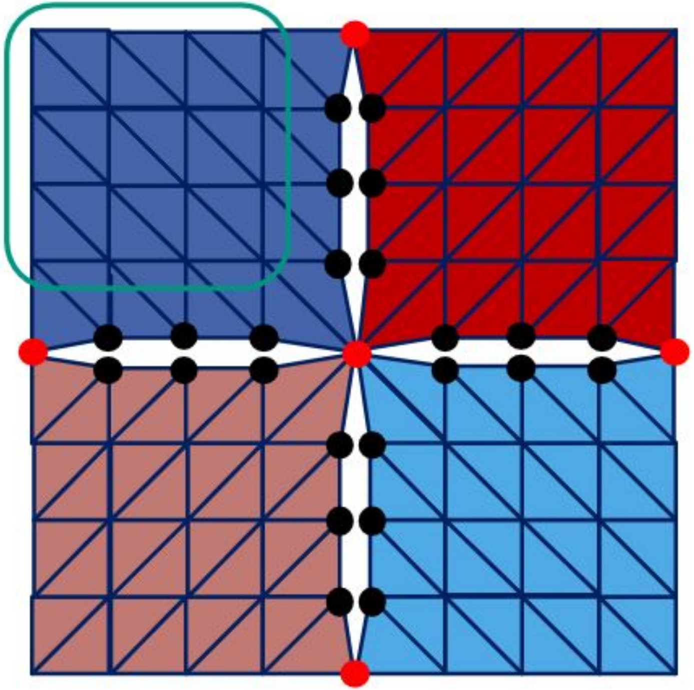
Ginkgo: Jacobi v/s Local IC v/s MG





Weak scaling on Karolina (A100) for 27 point stencil problem (1M per GPU) with CG

Balanced domain decomposition by constraints (BDDC)



- Consider local contributions to global stiffness matrices $A = \sum_{i=1}^{N} R_i^T A_i R_i$ independently
- Couple local systems via a coarse system

$$A_c = \sum_{i=1}^{N} R_{ci}^T A_{ci} R_{ci}$$

Where,

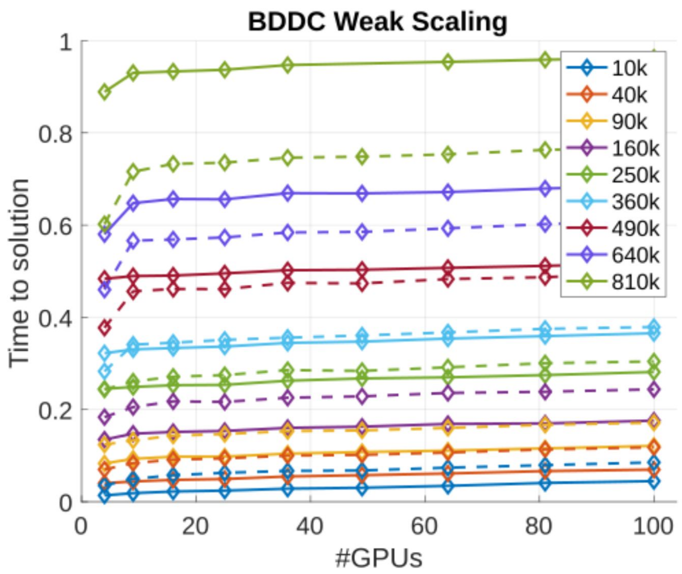
$$A_{ci} = \Phi_i^T A_i \Phi_i,$$

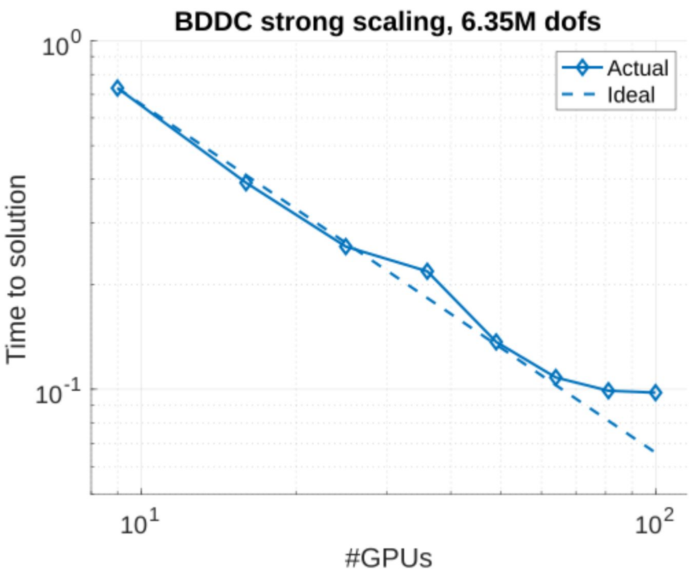
$$\begin{bmatrix} A_i & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} \Phi_i \\ \Lambda_i \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

- Constraints: Continuity on corners

  Averages on edges/faces

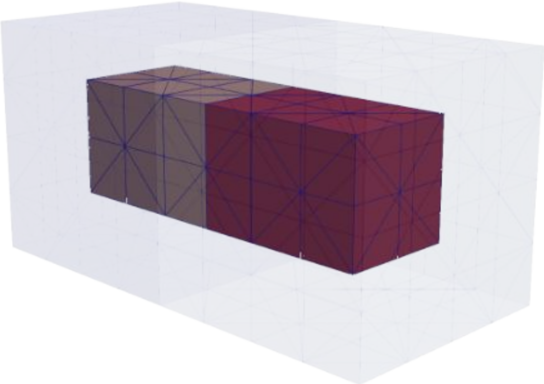
Ginkgo BDDC scaling

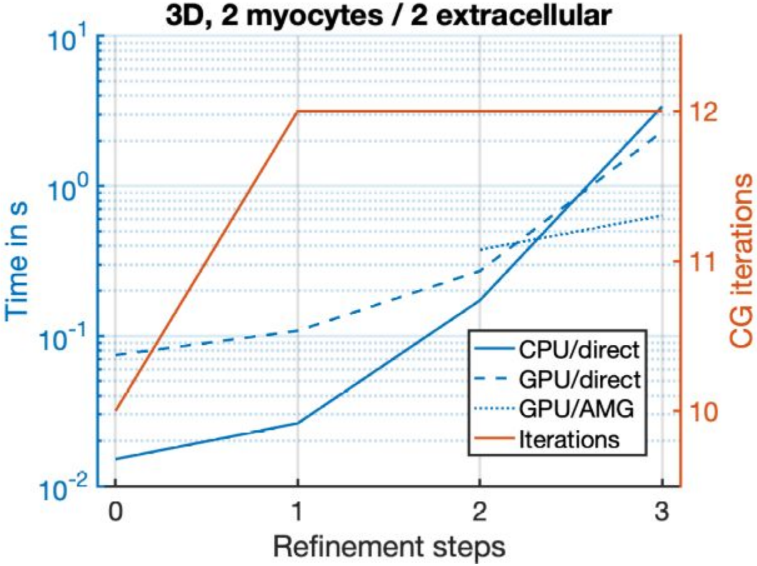




Strong and weak scaling on a 2D Poisson problem, A100 GPUs

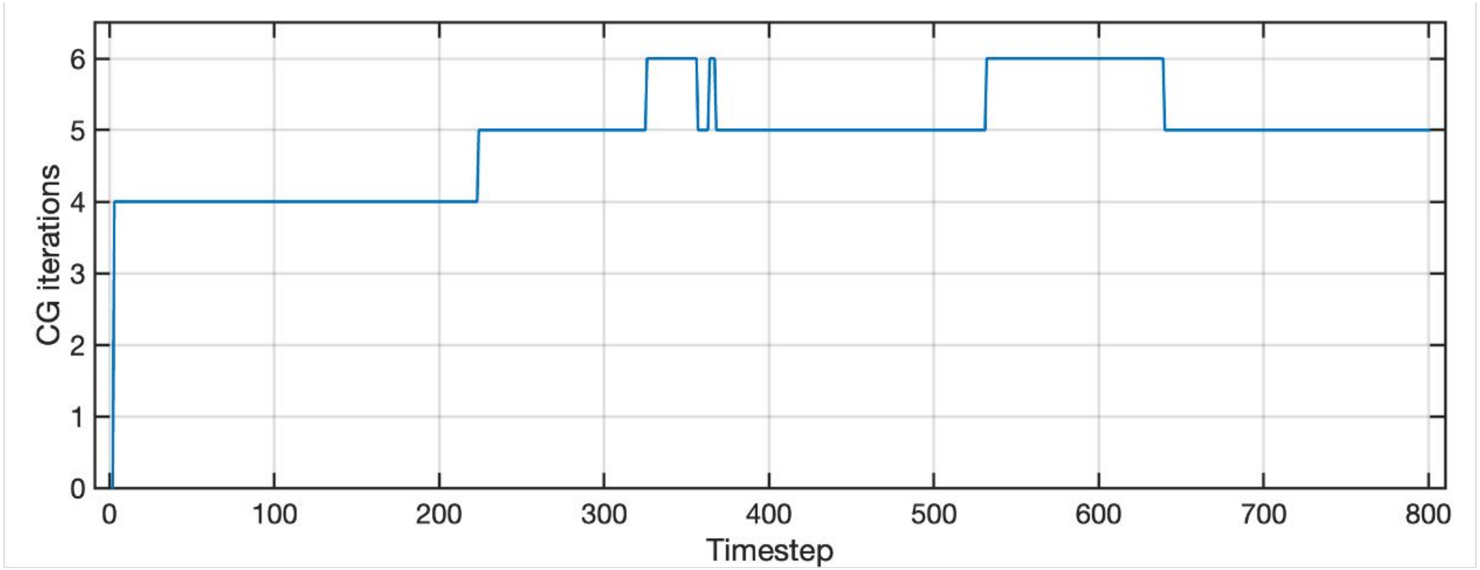
Ginkgo-OpenCARP example





4 ranks, 2 muscle cells, A100 GPU with CG and BDDC

Ginkgo-OpenCARP example



Linear solver iteration count over
different timesteps (for 3D example)

Summary and future work

- Ginkgo provides high performance with flexibility and composability
- Addition of other algebraic coarsening techniques
- Distributed ILU and ISAI-type preconditioners
- Evaluate other distributed preconditioners such as BPX, Optimized Schwarz

Thank you!

nayak@kit.edu

Summary and future work

- Ginkgo provides high performance with flexibility and composability
- Addition of other algebraic coarsening techniques
- Distributed ILU and ISAI-type preconditioners
- Evaluate other distributed preconditioners such as BPX, Optimized Schwarz

nayak@kit.edu

Backup

- Preconditioner application:
  - Remove interior residuals:

$$v_1 = \sum_{i=1}^{N} A_{Ii}^{-1} r, \; r_1 = r - Av_1$$

  - Coarse grid and subdomain corrections:

$$v_2 = \sum_{i=1}^{N} R_i^T W_i (P_{i1} + P_{i2}) W_i R_i r_1, \; r_2 = r_1 - Av_2$$

where

$$P_{i1} = \Phi_i A_c^{-1} \Phi_i^T$$

$$\begin{bmatrix} A_i & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} P_{i2} x \\ \mu \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

Global Synchronization

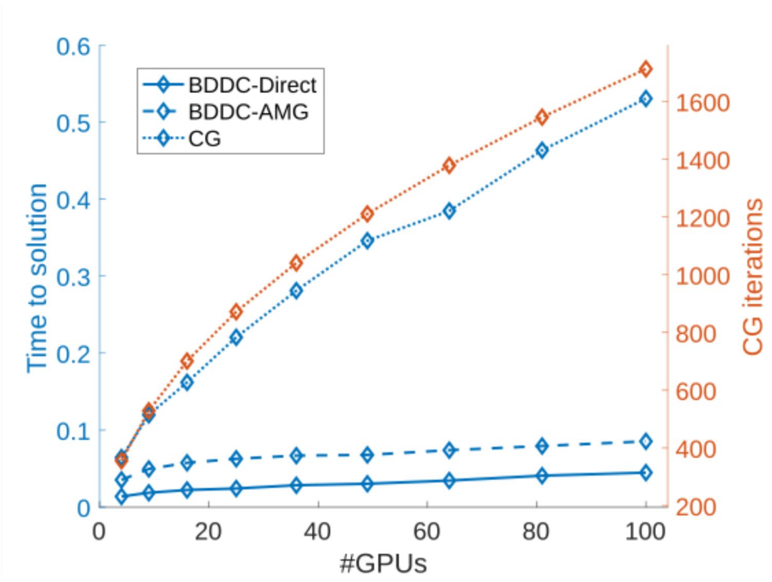
  - Correct interior dofs:

$$v_3 = \sum_{i=1}^{N} A_{Ii}^{-1} r_2$$

$$\Rightarrow \quad \mathrm{Pr} = v_1 + v_2 + v_3$$

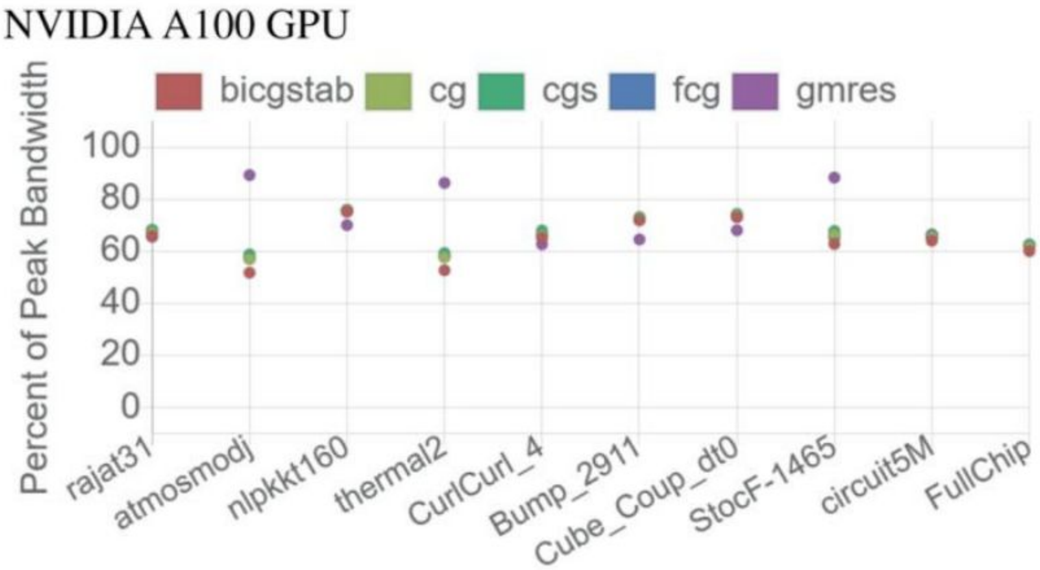
From: Dohrmann, 2007

Ginkgo BDDC



BDDC-Direct
BDDC-AMG
CG
Time to solution
CG iterations
#GPUs
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
1600
1400
1200
1000
800
600
400
200

Ginkgo: Features



Single GPU solver performance