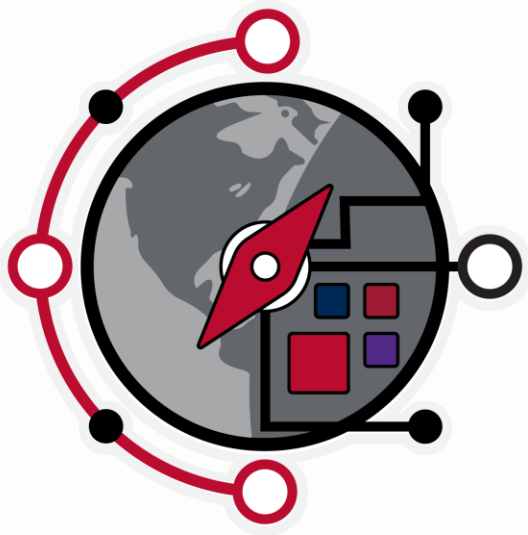




Center for
Optimized Modern
Parallel Adaptive
System Software

Integrated Project: Sparse Solver Performance Optimization

AMANDA BIENZ · UNIVERSITY OF NEW MEXICO

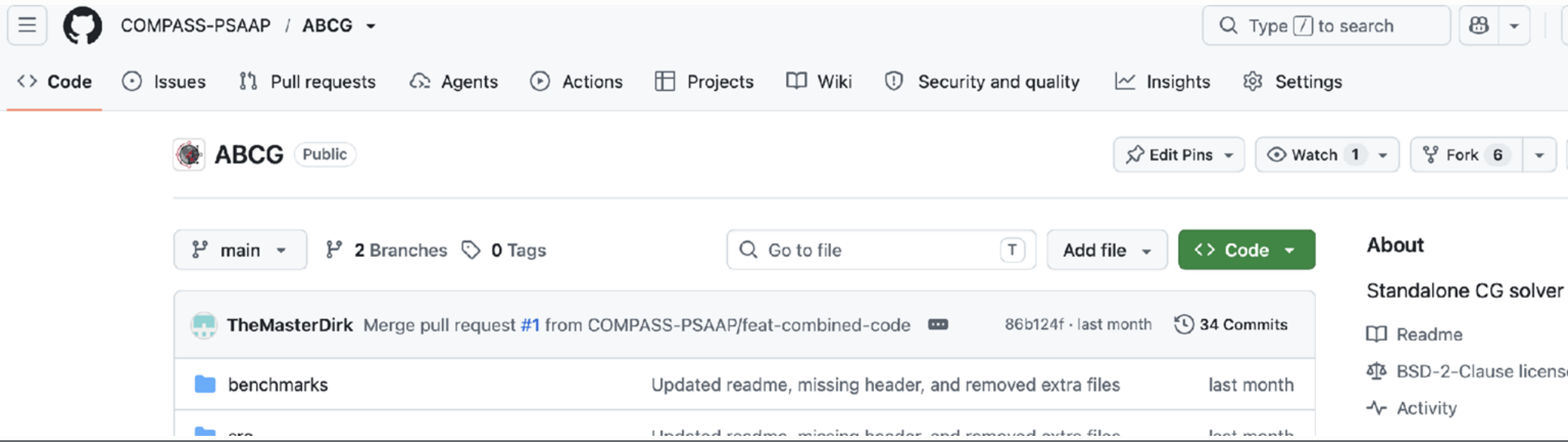
PSAAP 4 ANNUAL MEETING · OCTOBER 29, 2006

Sparse Solvers Roadmap

	Year 1 (FY26)				Year 2 (FY27)				Year 3 (FY28)				Year 4 (FY29)				Year 5 (FY30)			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
AI Performance / Modeling	Analyze AI model sensitivity				Model data collection & augmentation				Decision under uncertainty				Model testing & validation							
					Research model uncertainty quantification				Research model transfer learning				Enhanced model for optimization integration							
Cross-Layer / Optimization	Assess sparse solver optimizations		Assess Beatnik optimization			Layer-by-layer optimization		Search-based optimization				Integrate model uncertainty				Fully integrate models				
			Scheduler optimization sensitivity		Assess NNSA app optimization opportunities															
Solver Design	Beatnik fast multipole support		Z-Model adaptive meshing		Compare Krylov solver variants				Parallelize topological transitions				Specialize models for physics problems				Integrate models into production codes			
	Develop compressible interface model				Model topological transitions				Multi-scale reactive interface methods				Parallelize multi-scale reactive solver							
					Parallelize compressible interface model				Combine Krylov enhancements into unified solver											
System Software	MPI/Kokkos stream triggered communication				Additional MPI/Kokkos extensions								Time-based communication primitive optimization							
	Create initial RAPIDS primitives				Additional RAPIDS extensions								Cross-layer control of system software APIs							
			Kokkos memory layout optimizations		Develop System Tuning APIs															
Integration & / Assessment	Design app/system workflow				Assess Z-Model/FIESTA				Assess Krylov Variants											
	Design/deploy system infrastructure				Assess NNSA application optimization				Assess integrated hydro codes											
Education & / Workforce	Create bootcamp materials				Develop tradeoff analysis materials								Create AI model course materials				Offer courses with integrated materials			
			Deploy bootcamp modules		Create conference tutorial materials				Offer HOTI/SIAM/SC tutorials											

ABCG : COMPASS CG Solver

- Research-based CG solver
- Less optimized than NNSA CG solvers such as *hypra* and Belos
- Students benchmarking and optimizing as first pass, before integrating optimizations into larger codebases



The screenshot shows the GitHub interface for the 'ABCG' repository under the 'COMPASS-PSAAP' organization. The repository is public and has 2 branches and 0 tags. The main branch is selected. The repository description is 'Standalone CG solver'. The repository has 1 watch and 6 forks. The repository is managed by 'TheMasterDirk'. The repository has 34 commits. The repository has a README, BSD-2-Clause license, and activity. The repository has a 'benchmarks' folder. The repository has a 'src' folder. The repository has a 'Go to file' search bar. The repository has an 'Add file' button. The repository has a 'Code' button. The repository has an 'About' section. The repository has a 'main' branch. The repository has 2 branches. The repository has 0 tags. The repository has a 'Go to file' search bar. The repository has an 'Add file' button. The repository has a 'Code' button. The repository has an 'About' section. The repository has a 'main' branch. The repository has 2 branches. The repository has 0 tags. The repository has a 'Go to file' search bar. The repository has an 'Add file' button. The repository has a 'Code' button. The repository has an 'About' section.

COMPASS-PSAAP / ABCG

Type / to search

<> Code Issues Pull requests Agents Actions Projects Wiki Security and quality Insights Settings

ABCG Public

Edit Pins Watch 1 Fork 6

main 2 Branches 0 Tags

Go to file T Add file <> Code

TheMasterDirk Merge pull request #1 from COMPASS-PSAAP/feat-combined-code 86b124f · last month 34 Commits

benchmarks Updated readme, missing header, and removed extra files last month

src Updated readme, missing header, and removed extra files last month

About

Standalone CG solver

Readme

BSD-2-Clause license

Activity

ABCG – Current Status

- CPU-only CG with MPI Advance integration
- Simple ROCm version for Tuolumne
- Nathan Johnson (undergraduate student):
 - Adding NVIDIA version for Easley

$$r_0 = b - A x_0$$

SpMV: halo exchange

$$p_0 = r_0$$

$$\gamma_0 = (r_0, r_0)$$

MPI_Allreduce

for $i = 0, 1, \dots$

$$s = A p_i$$

SpMV: halo exchange

$$\alpha = \gamma_i / (s, p_i)$$

MPI_Allreduce #1

$$x_{i+1} = x_i + \alpha p_i$$

$$r_{i+1} = r_i - \alpha s$$

$$\gamma_{i+1} = (r_{i+1}, r_{i+1})$$

MPI_Allreduce #2

if $\sqrt{\gamma_{i+1}} < \text{tol}$: stop

$$\beta = \gamma_{i+1} / \gamma_i$$

$$p_{i+1} = r_{i+1} + \beta p_i$$

☐ SpMV: neighbor (point-to-point) communication

☐ Allreduce: global synchronization

Stream Triggering in ABCG

- Goal: extend strong scalability of NNSA solvers with stream triggering
- Students: **Greg Ziter-Glass** (undergraduate) and **Andy Borch** (PhD, started Fall'27)
- First step: Adding stream triggering into ABCG, testing performance beyond the strong scaling limit
- Next step: Investigate performance within Trilinos solvers such as Ifpack2 block Jacobi

$r_0 = b - A x_0$ SpMV: halo exchange

$p_0 = r_0$

$\gamma_0 = (r_0, r_0)$ MPI_Allreduce

for $i = 0, 1, \dots$

$s = A p_i$ SpMV: halo exchange

$\alpha = \gamma_i / (s, p_i)$ MPI_Allreduce #1

$x_{i+1} = x_i + \alpha p_i$

$r_{i+1} = r_i - \alpha s$

$\gamma_{i+1} = (r_{i+1}, r_{i+1})$ MPI_Allreduce #2

if $\sqrt{\gamma_{i+1}} < \text{tol}$: stop

$\beta = \gamma_{i+1} / \gamma_i$

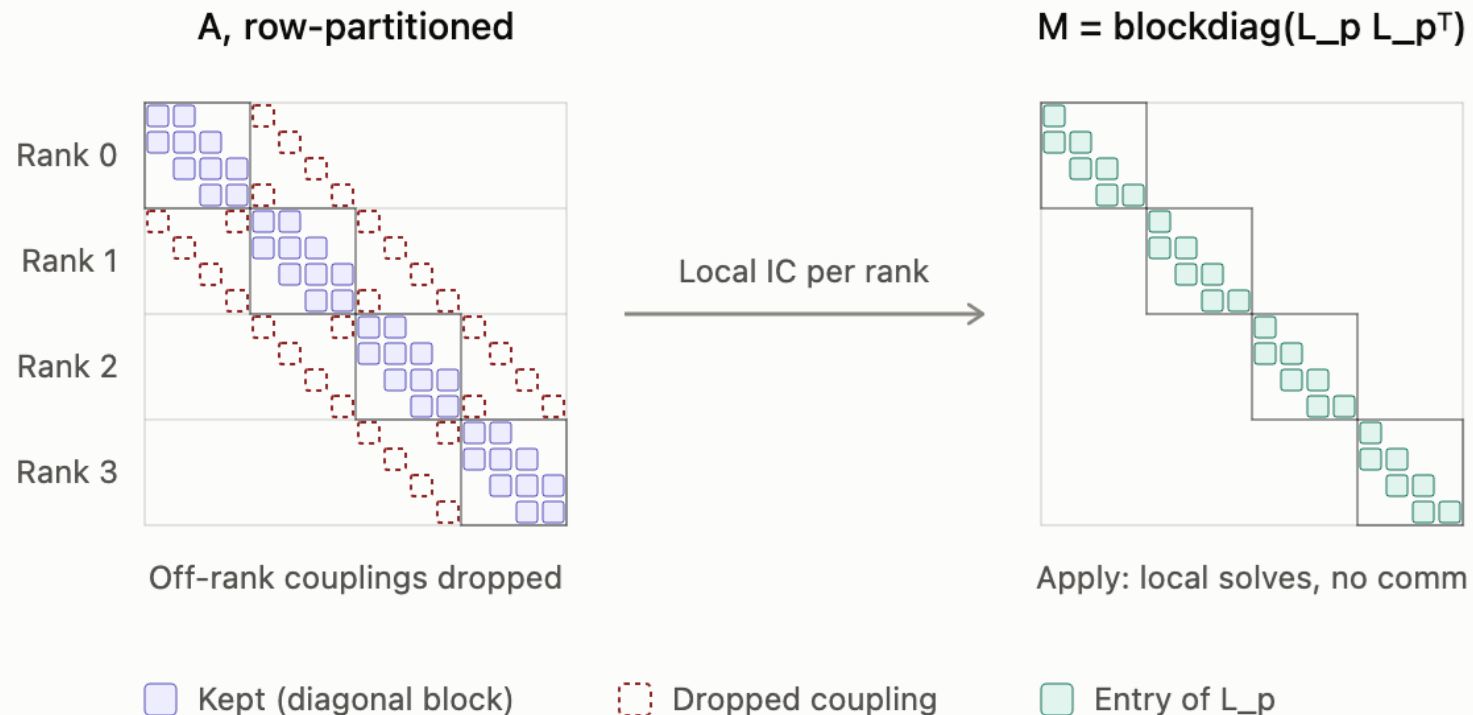
$p_{i+1} = r_{i+1} + \beta p_i$

☐ SpMV: neighbor (point-to-point) communication

☐ Allreduce: global synchronization

ABCG – Preconditioning and Asynchronous Variants

- Connor Gardiner (undergraduate student):
 - Added Incomplete Cholesky preconditioning



ABCG – Preconditioning and Asynchronous Variants

- Connor Gardiner:
 - Has implemented a pipelined CG version of ABCG

Standard CG

```
r0 = b - Ax0; p0 = r0
γ0 = (r0, r0)
for i = 0, 1, ...
    s = A pi
    α = γi / (s, pi)
    // blocking allreduce #1
    xi+1 = xi + α pi
    ri+1 = ri - α s
    γi+1 = (ri+1, ri+1)
    // blocking allreduce #2
    β = γi+1 / γi
    pi+1 = ri+1 + β pi
```

Pipelined CG

```
r0 = b - Ax0; w0 = Ar0
for i = 0, 1, ...
    γi = (ri, ri)
    δ = (wi, ri)
    // one fused lallreduce
    q = A wi // overlaps it
    β = γi/γi-1, α = γi/(δ - βγi/αi-1)
    // i = 0: β = 0, α = γ0/δ
    zi = q + β zi-1
    si = wi + β si-1
    pi = ri + β pi-1
    xi+1 = xi + α pi
    ri+1 = ri - α si
    wi+1 = wi - α zi
```

ABCG – Preconditioning and Asynchronous Variants

- Connor Gardiner:
 - Investigating progressing SpMV communication

```
// Parallel SpMV
void spmv(double alpha,
          ParMat& A,
          std::vector<double>& x,
          double beta,
          std::vector<double>& b,
          SpMVComm* comm)
{
    int proc, start, end;
    int tag = 0;

    comm->pack(x);

    comm->start();

    spmv(alpha, A.on_proc, x.data(), beta, b.data());

    comm->wait();

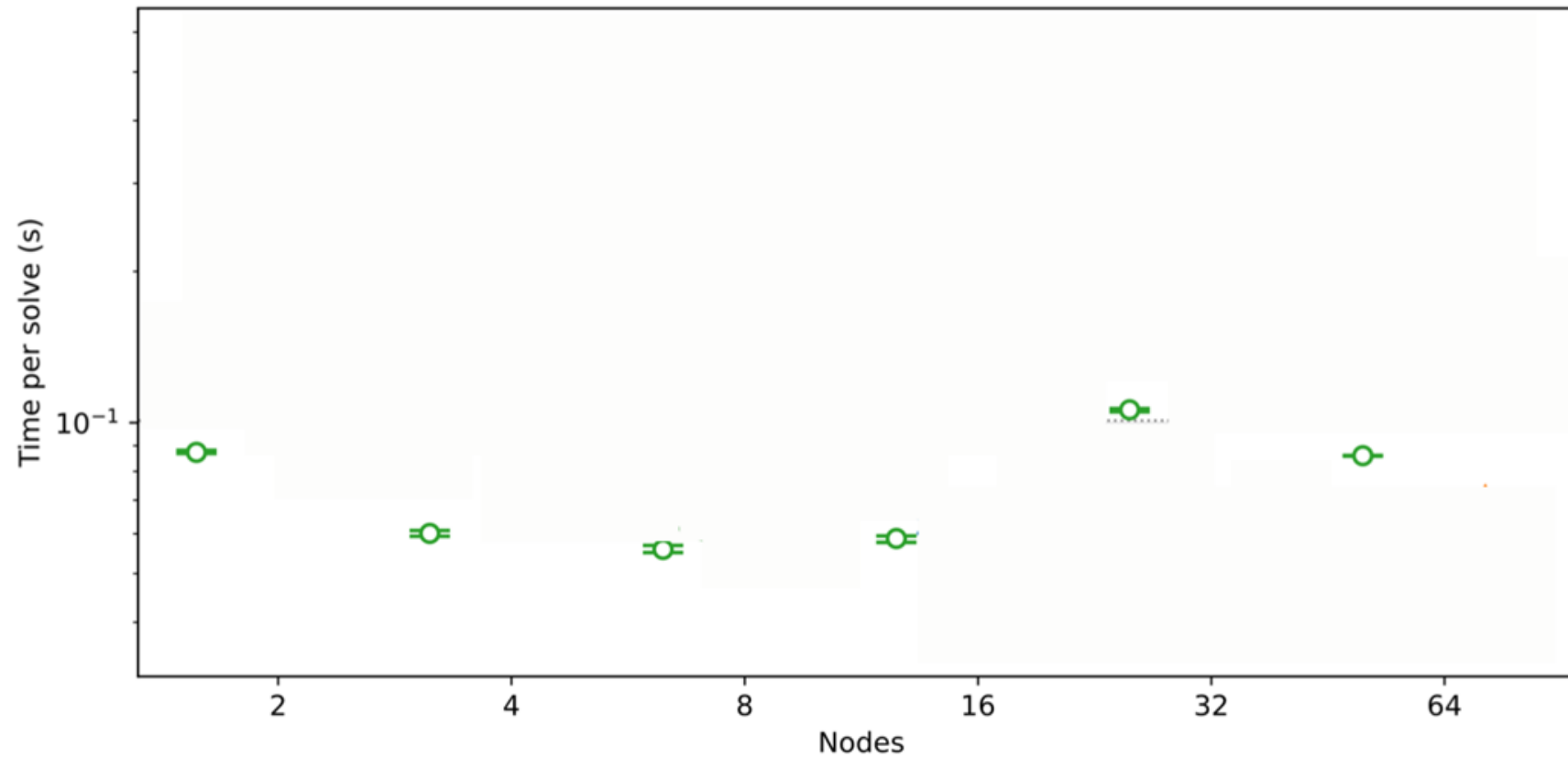
    spmv(alpha, A.off_proc, comm->get_recvbuf(), 1.0, b.data());
}
```

ABCG – Preconditioning and Asynchronous Variants

- Connor Gardiner:
 - Added Incomplete Cholesky preconditioning
 - Has implemented a pipelined CG version of ABCG
 - Investigating progressing SpMV communication
- CG on 64 Processes of Easley, Dubcova2.pm
 - Standard CG: 169 iterations, 1.87 seconds
 - Preconditioned CG: 141 iterations, 1.72 seconds
 - Pipelined PCG: 1.54 seconds
 - Pipelined PCG with lallreduce: 1.39 seconds
 - Pipelined PCG with lallreduce and OMP Spmv: 1.31 seconds

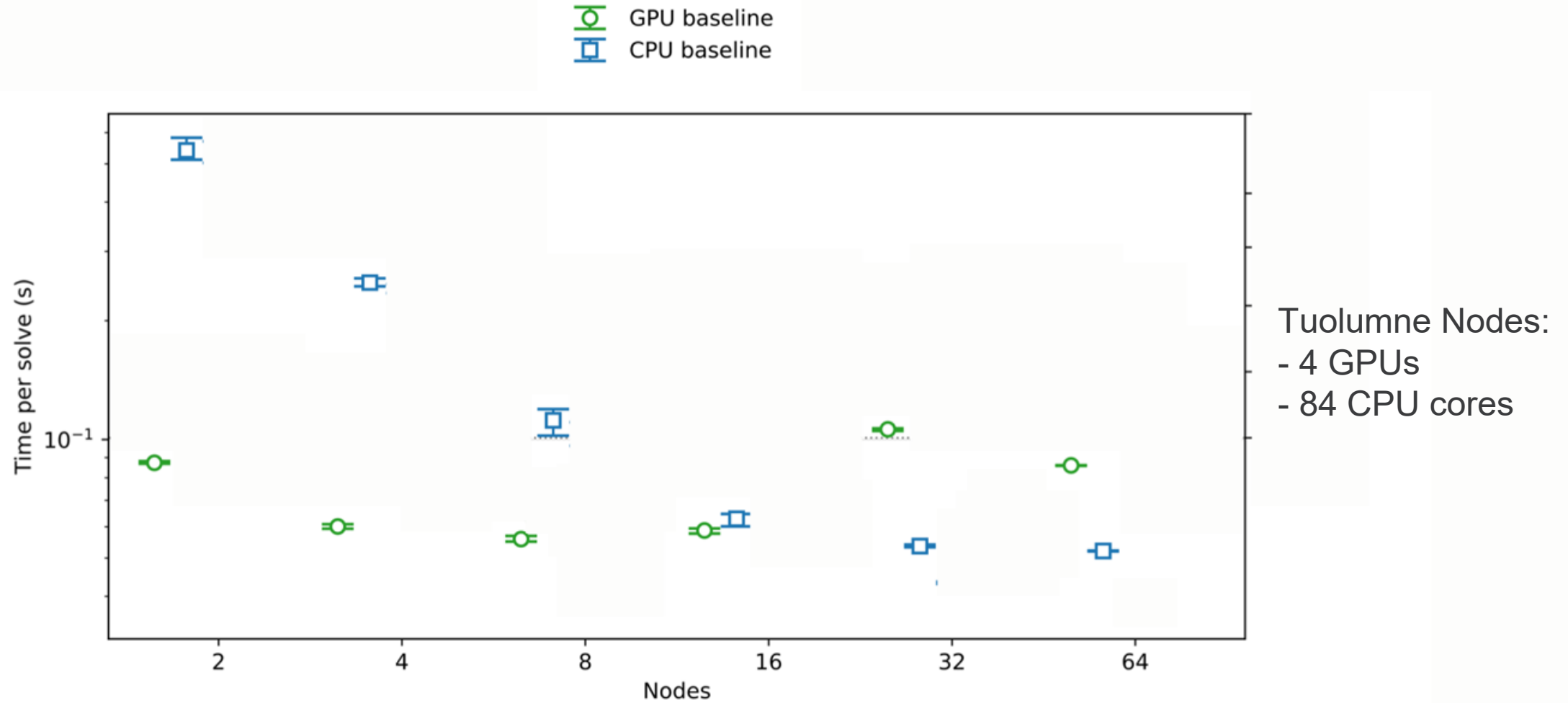
hypre – CG Performance

 GPU baseline



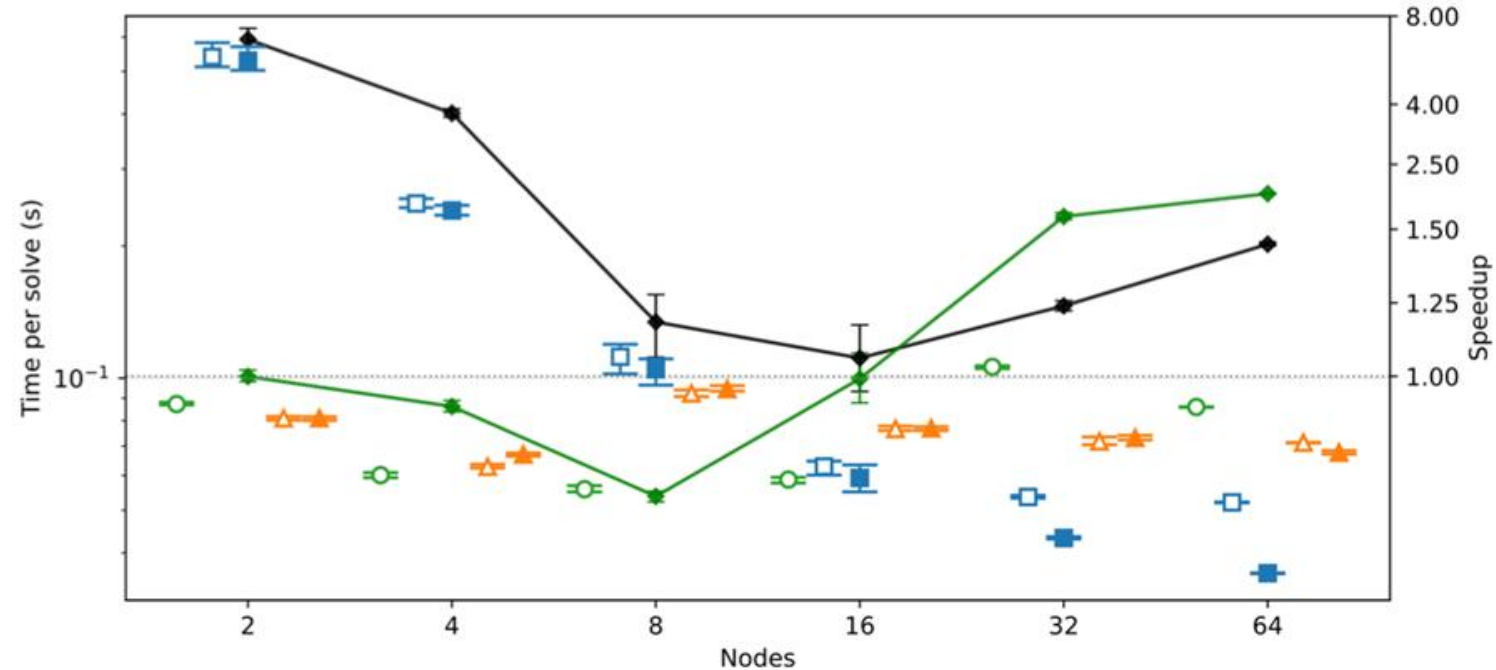
Tuolumne Nodes:
- 4 GPUs
- 84 CPU cores

hypre – CG Performance



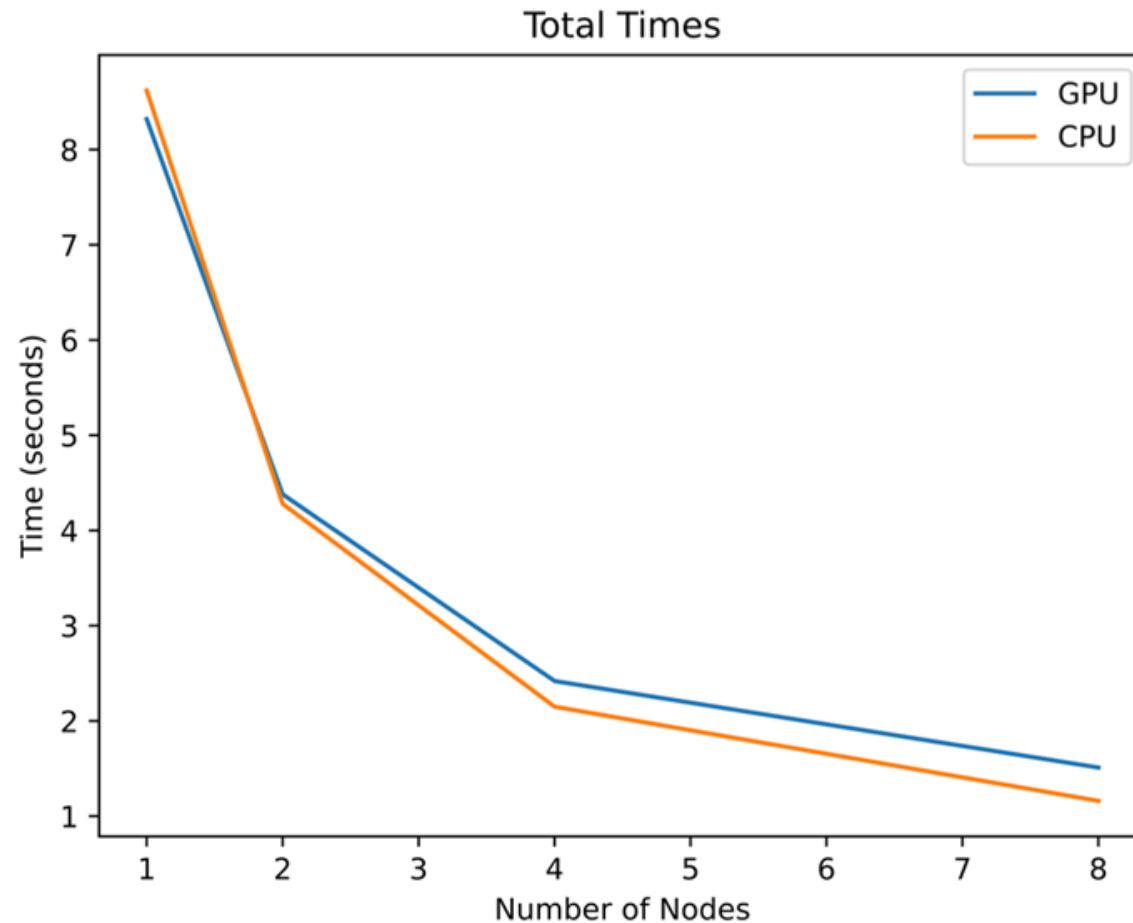
hypre – CG Performance

Speedup (best CPU baseline / best numaaware) GPU baseline CPU numaaware GPU CPX numaaware
Speedup (best GPU baseline / best numaaware) CPU baseline GPU CPX baseline



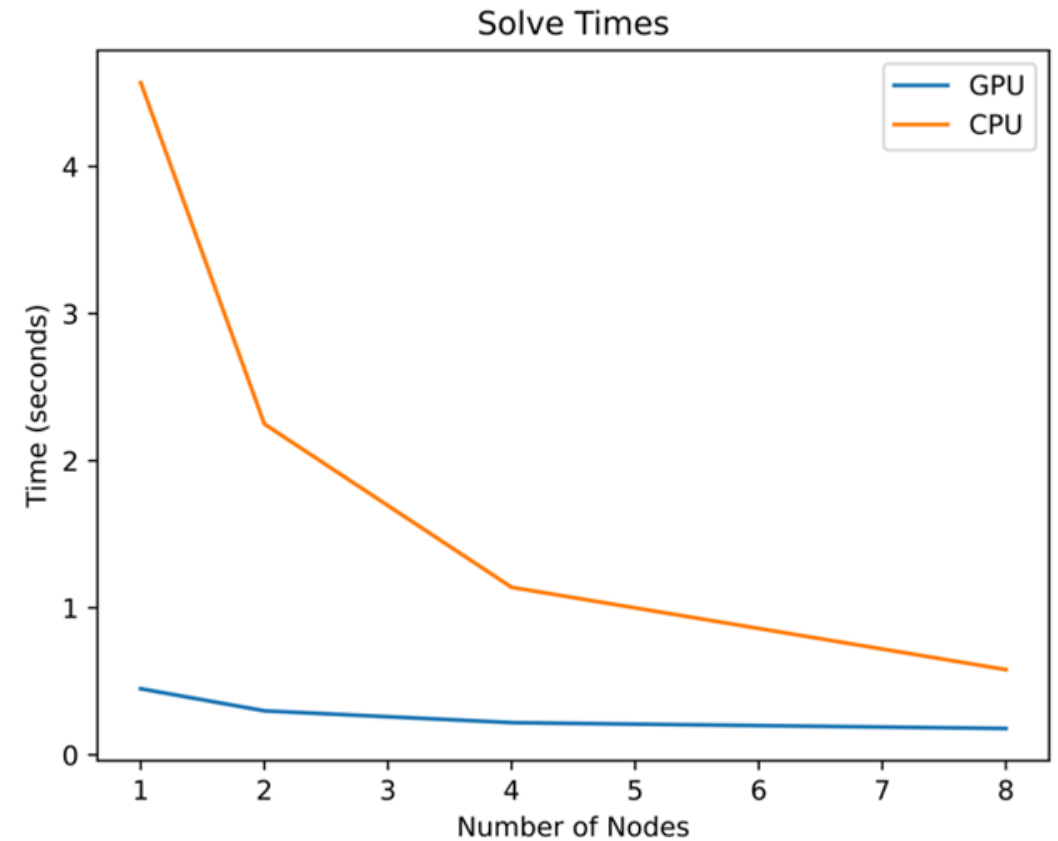
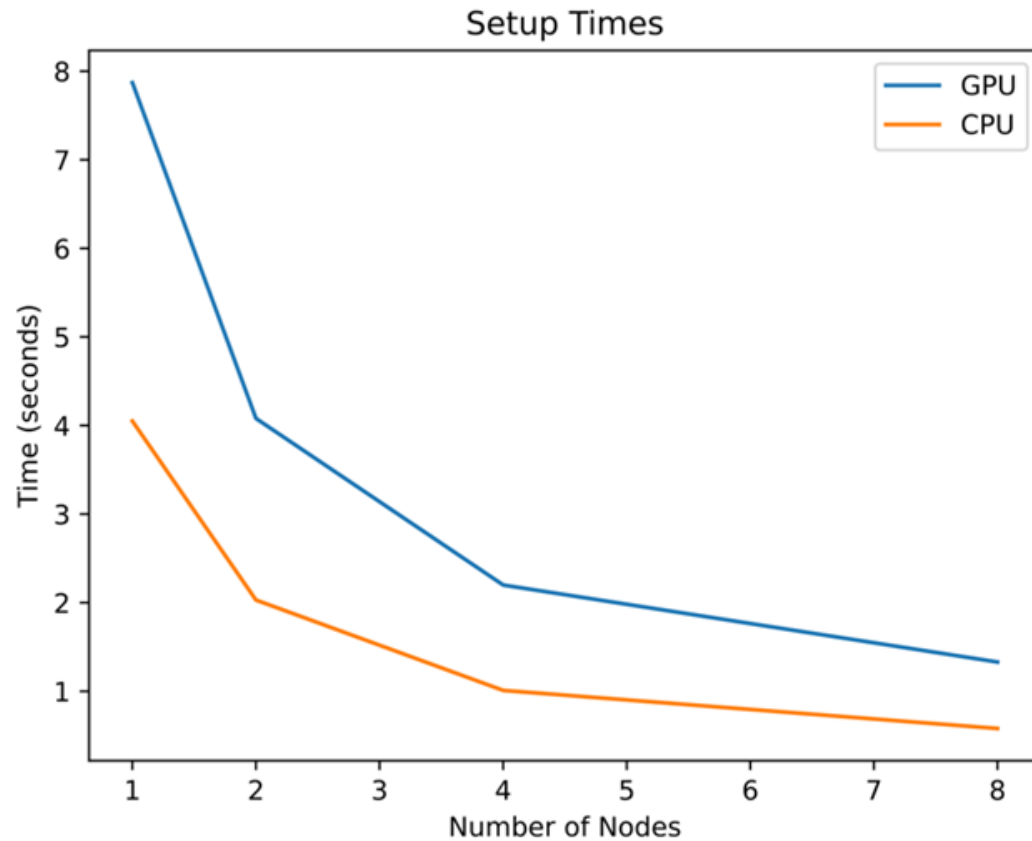
Mike Adams: Improving The Scalability of GPU-Aware Iterative Solvers Using Locality-Aware Collectives and All Available Processing Per Node

hydre – AMG Performance

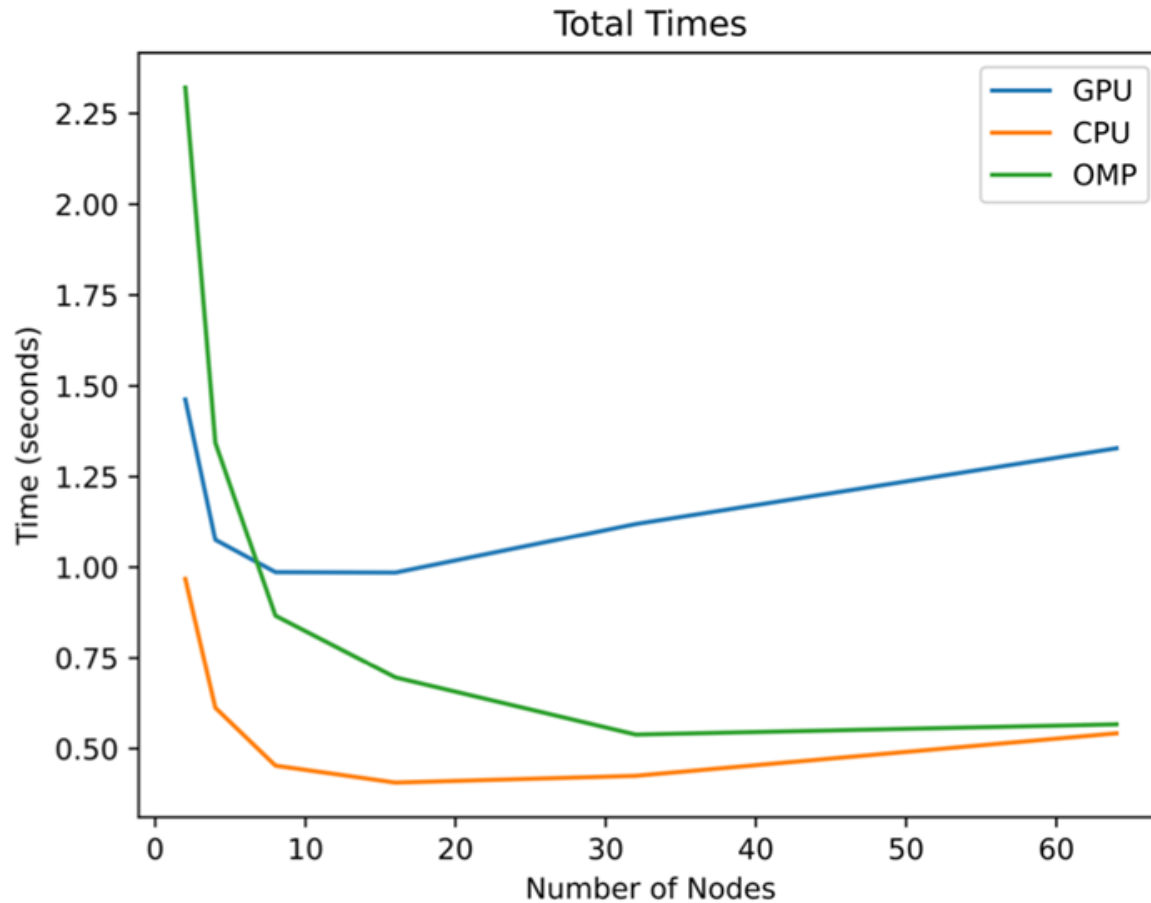


- 105pt Laplacian on Tuolumne
- 4 GPUs per node
- 84 CPU cores per node

hypr – AMG Performance



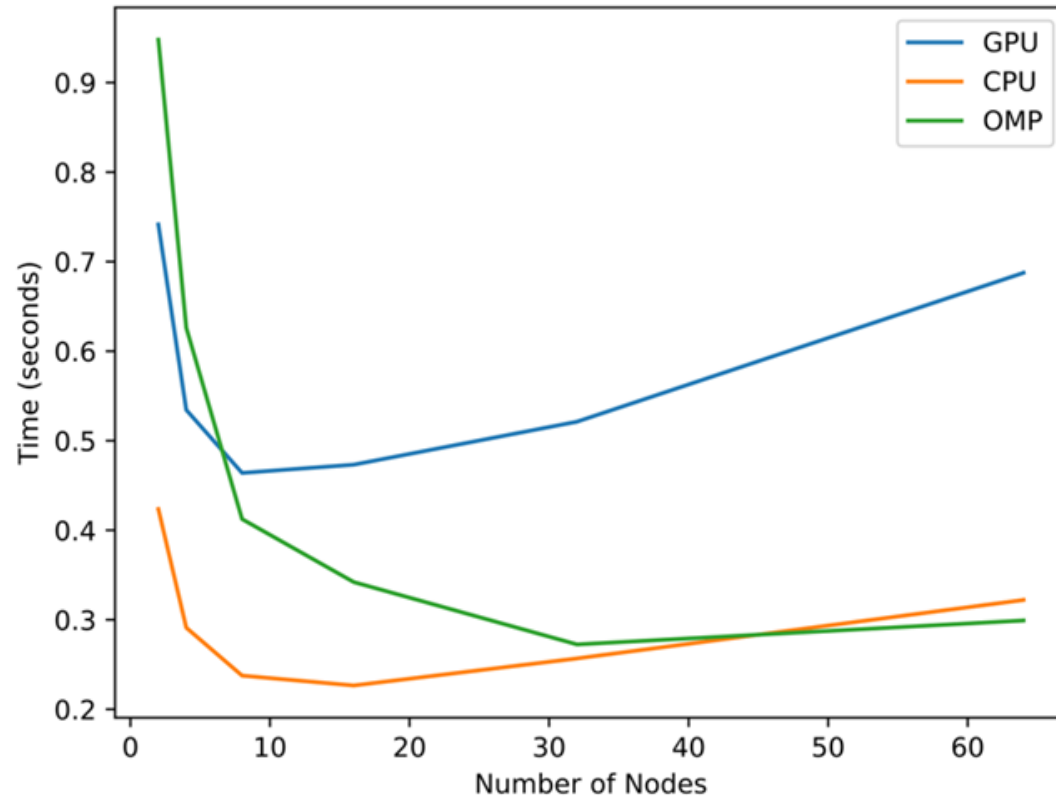
MueLu – AMG Performance



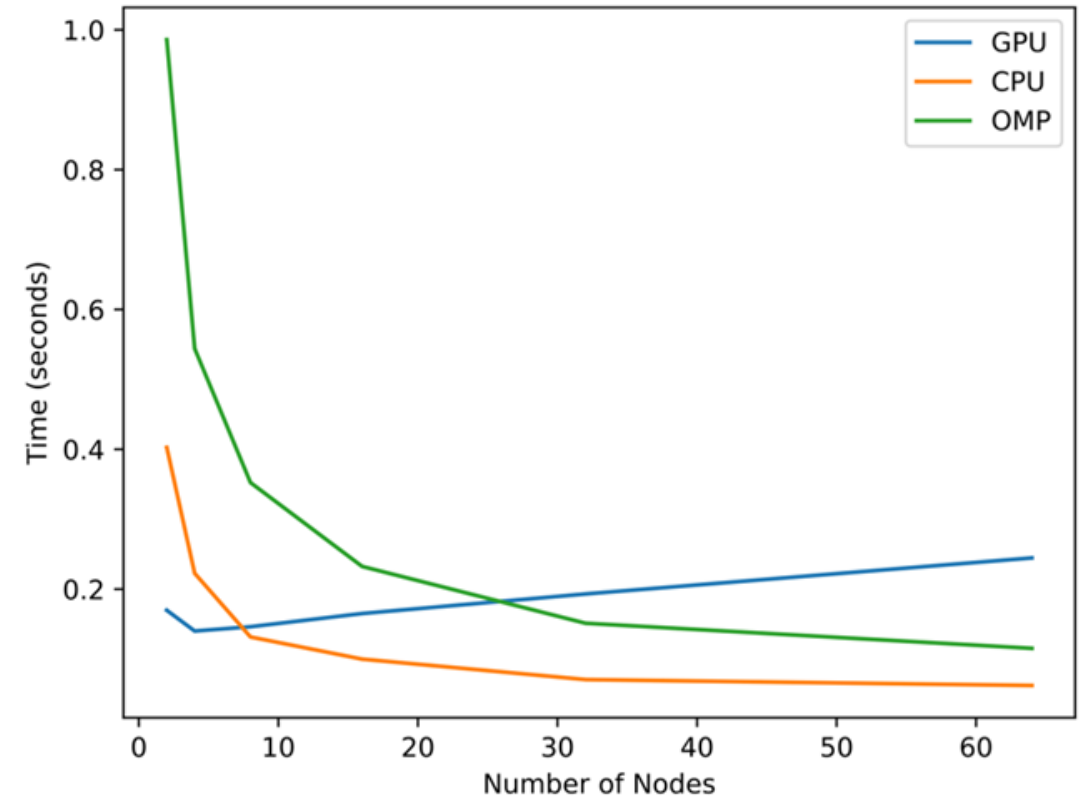
- 256 x 256 x 256 Laplace3D on Tuolumne
- 4 GPUs per node
- 84 CPU cores per node
- OpenMP:
 - 4 MPI processes
 - 21 threads per process

MueLu – AMG Performance

Setup Times



Solve Times

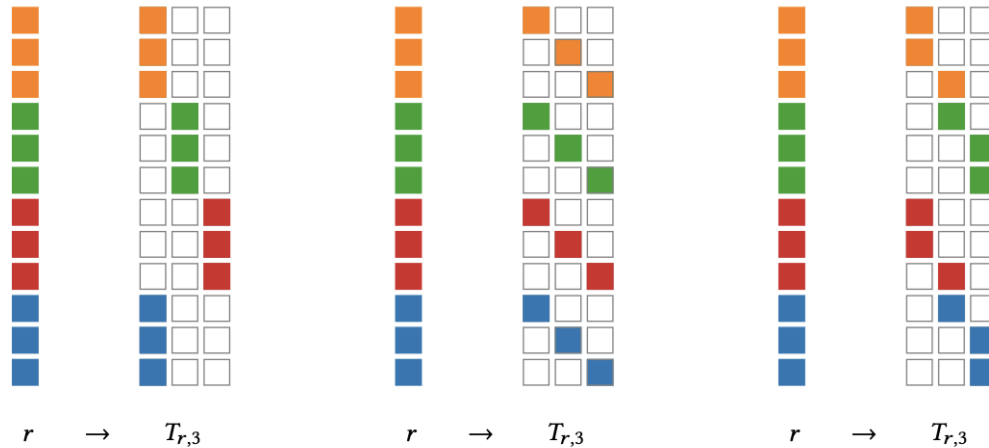


Plans for Coming Year

- Investigate trade-offs between standard and asynchronous CG variants
 - Impact of MPI Advance communication optimizations
 - Impact on GPU-Aware CG
- Optimizing *hypre* and MueLu AMG solves for emerging systems:
 - Creating a hybrid solve that uses CPUs + OpenMP in setup, GPUs in solve when warranted
 - Goal: investigate impact on MiniEM

Plans for Coming Year

- Enlarged Krylov Methods — RAPtor
 - Instead of a single right-hand-side vector, have a block of vectors
 - More computation, faster convergence. Has shown promise on GPU systems.
 - Currently no enlarged preconditioner
 - Collaborative effort: working on AMG preconditioner for block right hand side



Lockhart et al, Performance Analysis and Optimal Node-Aware Communication for Enlarged Conjugate Gradient Methods

Conclusions and Acknowledgements

- Linear solvers often have poor strong scalability, particularly when using GPUs
- Investigating methods for improving the performance of linear solvers, particularly at strong scaling limits, through stream-triggering, communication optimizations, increased asynchrony, and using all available resources.
- Investigating trade-offs of Krylov method variants

This work was performed with partial support from the National Science Foundation under Grant Nos. CCF-2338077 and CNS-2450092 and the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program Award DE-NA0004267.

