Porting Hypre's BoomerAMG to Heterogeneous Computer Architectures: Strategies, Experiences and Optimizations

ECP COMMUNITY BOF: Recent Advances in Selected Sparse Linear Solvers Libraries – Virtual Meeting

May 10, 2022

Ruipeng Li

LLNL-PRES-831941

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

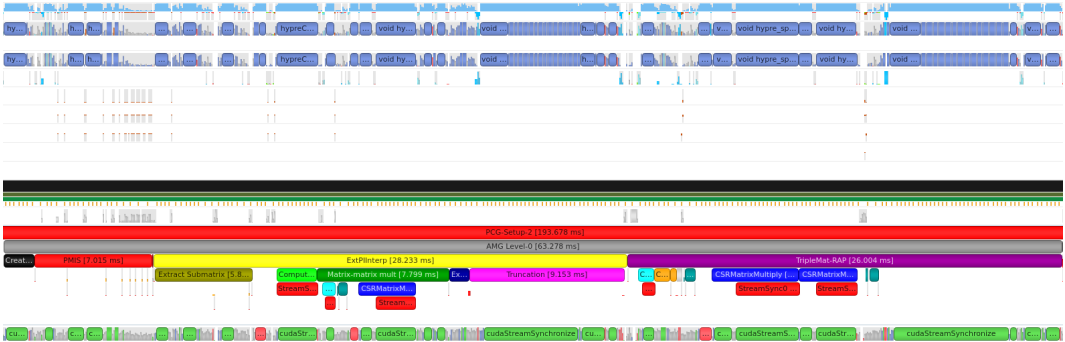
HPC transitioning to heterogeneous GPU architectures

- **Sierra** at LLNL, **Summit** at ORNL. NVIDIA V100
- **Perlmutter** at NERSC. NVIDIA A100
- **Frontier** at ORNL. AMD Instinct 250X
- **El Capitan** at LLNL, AMD Instinct GPUs
- **Aurora** at ANL, Intel GPUs
- All have **"Fat node"** architecture. Most of the FLOPS are on the GPUs

Complete AMG setup and solve on GPUs

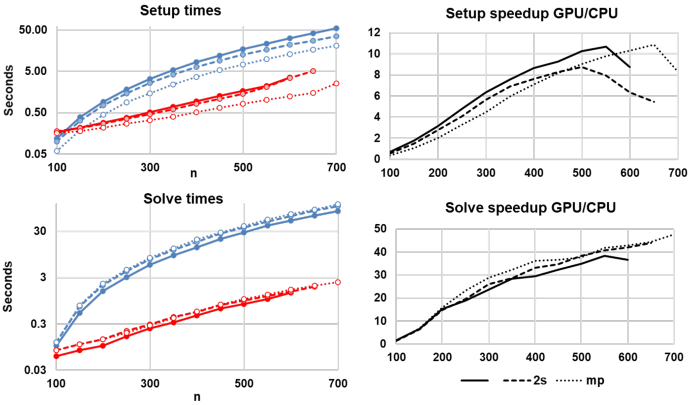
- AMG setup
  - Coarsening: PMIS
  - Interpolation: direct, MM-ext;
  - Aggressive coarsening; Multipass interpolation
  - ***RAP***
- AMG solve
  - Matrix-by-vector: save local $\boldsymbol{P^T}$
  - Smoother: Jacobi; Gauss-Seidel; Chebyshev and GMRES polynomials (NEW!)

A closer look at AMG setup on GPUs



Results for AMG-PCG on Crusher: 27-pt diffusion

- 1 node: CPU: 64 MPI, GPU: 8 GPUs, $\boldsymbol{n} \times \boldsymbol{n} \times \boldsymbol{n}$ grid



- ___ : default + Jacobi (0.85)
- - - - : 2s: +1 lvl aggr. coarsening, 2-stage interp.
- ..... : mp: +1 lvl aggr. coarsening, multipass interp.

Sparse matrix-matrix multiplications in AMG setup

1. Coarse-grid operator: $\boldsymbol{P^{\mathsf{T}}AP}$
2. Aggressive coarsening: 2nd stage coarsening $\boldsymbol{S^{(A)}_{CC} = (S^2 + S)_{CC}}$
3. MM-ext interpolation
   - PMIS requires *extended* interp. (De Sterck 2008) with distance-2 interp. sets
   - Original formulation is difficult for efficient implementation on GPUs
   - New formulation in matrix-matrix (MM) multiplications

$$\boldsymbol{W = -\left[(D_{FF} + D_\gamma)^{-1}(A^s_{FF} + D_\beta)\right]\left[D_\beta^{-1}A^s_{FC}\right] \equiv -\tilde{A}^s_{FF}\tilde{A}^s_{FC},}$$

4. Multipass interpolation in MM forms

Approximating the ideal $W^* = -A_{FF}^{-1}A_{FC}$

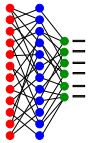
- MM-ext: $W = -(D_{FF} + D_\gamma)^{-1}(A_{FF}^s D_\beta^{-1} + I)A_{FC}^s \equiv -Y A_{FC}^s$
- $A = D + A^s + A^w$, $D_\beta = \mathrm{diag}(A_{FC}\mathbf{1}_c)$, $D_\gamma = \mathrm{diag}(A_{FF}^w \mathbf{1}_f + A_{FC}^w \mathbf{1}_c)$
- Easy to verify $-Y A_{FC}^s \mathbf{1}_c = \mathbf{1}_f$
- Assuming $A^w = \mathbf{0}$ and $D_\beta = -\mathrm{diag}(A_{FF}\mathbf{1}_f) = -D_\alpha$

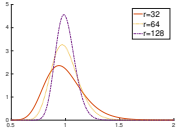
$$\begin{aligned} I + Y A_{FF} &= I + D_{FF}^{-1}((A_{FF} - D_{FF})D_\alpha^{-1} - I)A_{FF} \\ &= (I - D_{FF}^{-1}A_{FF})(I - D_\alpha^{-1}A_{FF}) \end{aligned}$$

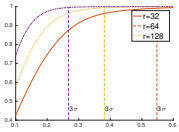
- $I + Y A_{FF} \approx \mathbf{0}$ if A_{FF} is strongly d.d, i.e., $Y \approx A_{FF}^{-1}$

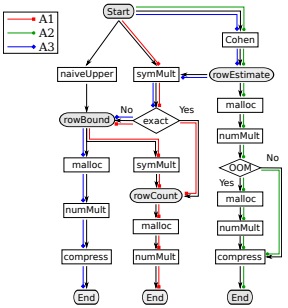
Optimizing sparse matrix-matrix kernel

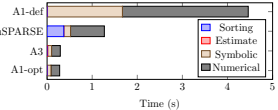
- Hash-based *sparse accumulators*; rows are *unsorted*
- Naive row NNZ bound: number of intermediate products
- *symbolic* multiplication: row NNZ *count* (or bound)
- *numeric* multiplication with adequate memory allocated
- Stochastic estimator (Cohen, 1997): good estimate
  - standard deviation $\sigma = 1/\sqrt{r-2}$. $\hat{z}' = \frac{\hat{z}}{1-3\sigma}$
  - $\hat{z}' \geq z$ and $\hat{z}' \leq \frac{1+3\sigma}{1-3\sigma} z$ with high probabilities







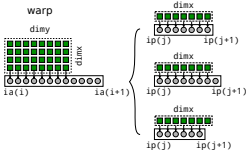




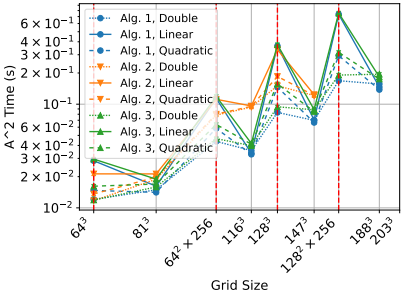
Squaring 125-point matrix

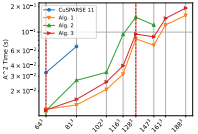
Optimizing sparse matrix-matrix kernel (cont'd)

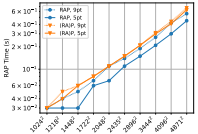
- Robust hash probings
  1. linear: $(h'(k) + i) \bmod s$
  2. quadratic: $(h'(k) + c_1 i + c_2 i^2) \bmod s$;
  3. double: $(h'(k) + i h_2(k)) \bmod s$
- Load balancing: binning with row NNZ
  - $\langle \texttt{SHMEM\_HASH\_SIZE}, \texttt{GROUP\_SIZE} \rangle = \langle 32, 2 \rangle \ldots \langle 16384, 1024 \rangle, \langle 16, 2 \rangle \ldots \langle 8192, 1024 \rangle$
- Load balancing: thread-group partition



- Triple matrix product kernel







GPU utilization deteriorates with smaller problems and coarser levels

Known issues that lower GPU utilization

1. Synchronizations: Thrust syncs stream after calls
   - (*temporarily*) replaced with custom kernels
2. D2H and H2D copies
3. Small kernels
4. MPI

- GPU utilization (single GPU)
- "CPU overhead" = kernel launch overhead + CPU computation

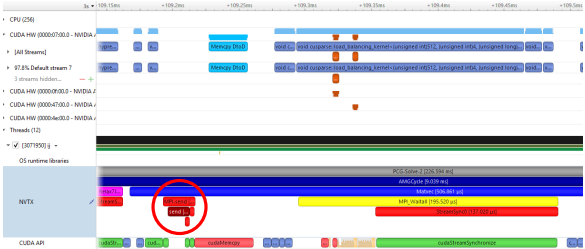
	100x100x100		100x100x200		200x200x200	
	setup	solve	setup	solve	setup	solve
time	49.7	38.1	79.3	60.7	252.9	186.5
GPU time	36.49	36.6	65.3	58.8	239.4	185.5
CPU overhead	26.58%	3.94%	17.65%	3.13%	5.34%	0.54%

➤ For the "**really small**" $\mathbf{100^3}$ problem,

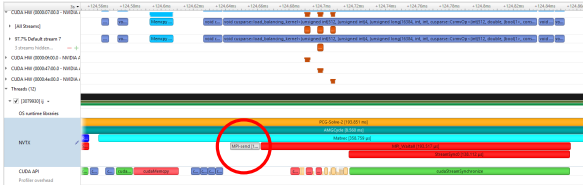
100x100x100 setup	mxl2	mxl3	mxl4	mxl5	mxl6	mxl7	mxl8
time	19.6	31.9	37.6	41.5	44.8	47.6	49.7
GPU time	17.9	28.6	31.9	33.8	35.1	35.7	36.49
CPU overhead	8.67%	10.34%	15.16%	18.55%	21.65%	25.00%	26.58%

Overlapping GPU execution with CUDA-aware MPI

- Pack halo data
- StreamSync
- MPI-Isend/Irecv
- local matvec
- MPI-waitall

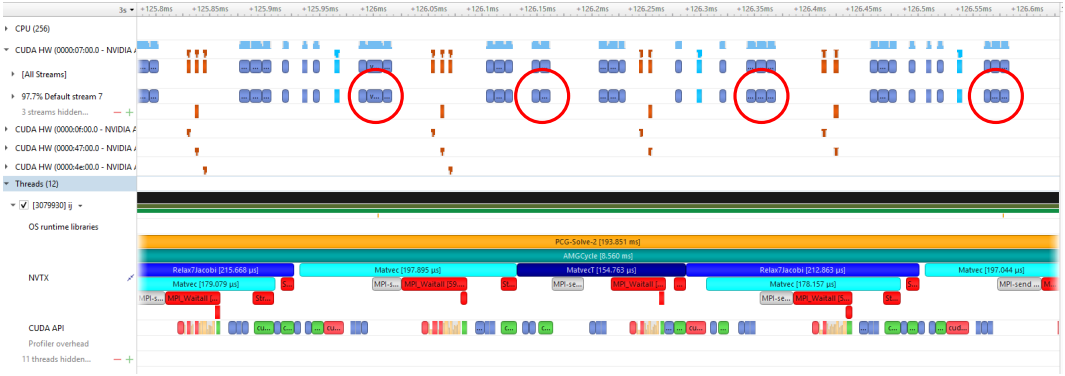


- Pack halo data
- StreamSync
- local matvec
- MPI-Isend/Irecv
- MPI-waitall



Overlapped MPI helps less at coarser levels

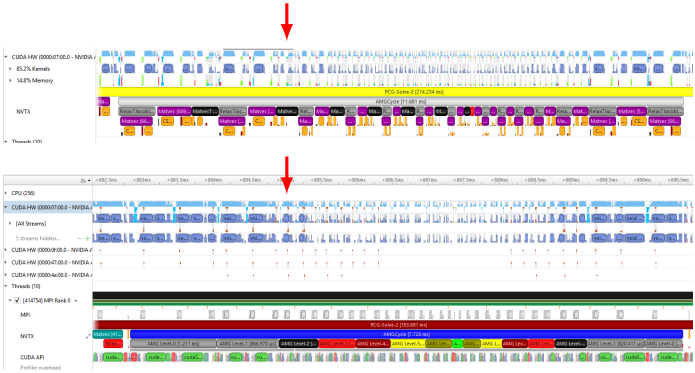
- Again, kernels are too small



Putting them together: more optimized AMG on GPUs

- AMG + PCG, 1 node, 4 A100 GPUs for $\mathbf{200^3}$ 7-pt Laplacian. Time (s)

	Setup	Solve
old	0.877	0.225
MallocAsync	0.190	0.223
Umpire	0.177	0.217
opt	0.156	0.137



- The work of optimized SpGEMM was not included in “opt”

Conclusion

- *hypre*'s structured solvers, AMG solvers, and more have been ported to heterogeneous GPUs
  - CUDA/HIP/oneAPI libraries ⇒ Thrust/rocPRIM/oneDPL ⇒ kernels
- AMG algorithms on GPUs have been "stabilized", and are under optimizations
- Port BoomerAMG to Intel GPUs
- More coarsening algorithms on GPUs other than PMIS
- GPU-appropriate smoothers: FSAI
- New Semi-structured solvers on GPUs

R. Falgout, R. Li, B. Sjogreen, U.M. Yang, and L. Wang, "Porting hypre to Heterogeneous Computer Architectures: Strategies and Experiences", Parallel Computing, 2021

CASC

Center for Applied
Scientific Computing

Lawrence Livermore
National Laboratory

THANK YOU!

Questions & Comments

li50@llnl.gov