GPU-Accelerated Matrix-Free Methods in Geophysics: Case Studies in pTatin3d and StagYY

Karl Rupp

https://karlrupp.net/

Freelance Computational Scientist
and
Institute for Microelectronics, TU Wien

Joint work with Dave May (Univ. Oxford, UK),
Patrick Sanan (Univ. Svizzera italiana, CH), Paul Tackley (ETH Zürich)

PASC Conference, Lugano
June 26-28, 2017

PASC Project: GeoPC (2013-2016)

- Infrastructure development for hybrid parallel smoothers for multigrid preconditioners
- Today's topic: GPU acceleration for SpMV
- More results: MS8, 17:00-17:30, Room C2-3

Platform for Advanced Scientific Computing

GPU-SpMV in pTatin3d

About pTatin3d

- Geodynamics modeling package
- Simulates long-term lithospheric deformation
- Solves heterogeneous Stokes problems

Discretization and Solver

- Inf-sub stable $Q_2 - P_1^{\text{disc}}$ elements
- (F)GMRES with multigrid preconditioner
- Matrix-free application of viscous block

Equations in Ω

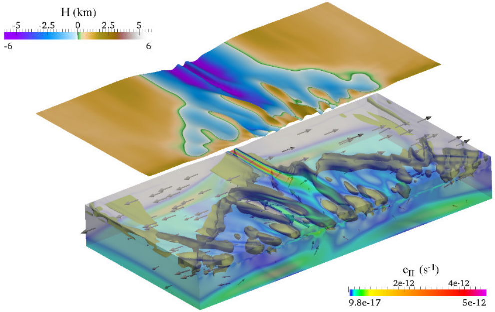
$$\nabla \cdot \left[2\eta(\mathbf{u}, p)\mathbf{D}(\mathbf{u})\right] - \nabla p = \mathbf{f}\,, \quad \text{where } \mathbf{D}(\mathbf{u}) := \frac{1}{2}\left(\nabla \mathbf{u}^{\mathrm{T}} + \nabla \mathbf{u}\right)\,,$$

$$\nabla \cdot u = \mathbf{0}$$

Fluid velocity $\mathbf{u}$, pressure p

Effective shear viscosity η

Body force $\mathbf{f}$



(D. May, J. Brown, L. Le Pourhiet, 2014)

Boundary Conditions

$\mathbf{u} = \bar{\mathbf{u}}$ on Γ_{D} (Dirichlet)

$\mathbf{u} \cdot \mathbf{n} = \bar{\mathbf{t}}$ on Γ_{N} (Neumann)

Field-Split for Nonlinear System

$$\begin{bmatrix} \mathbf{J}_{\mathrm{uu}} & \mathbf{J}_{\mathrm{up}} \\ \mathbf{J}_{\mathrm{pu}} & \mathbf{0} \end{bmatrix} = - \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_p \end{bmatrix}$$

$\mathbf{J}_{\mathrm{uu}}$ symmetric, positive definite

Schur complement $\mathbf{S} = -\mathbf{J}_{\mathrm{pu}}\mathbf{J}_{\mathrm{uu}}^{-1}\mathbf{J}_{\mathrm{up}}$ (expensive)

Approximate Preconditioner

$$\mathbf{P} = \begin{bmatrix} \tilde{\mathbf{J}}_{\mathrm{uu}} & \mathbf{0} \\ \mathbf{J}_{\mathrm{pu}} & \tilde{\mathbf{S}} \end{bmatrix}$$

Multigrid for $\tilde{\mathbf{J}}_{\mathrm{uu}} := \mathbf{J}_{\mathrm{uu}}$ with Jacobi-preconditioned Chebychev-smoother

Scaled mass-matrix for $\tilde{\mathbf{S}}$

Matrix-free Application of $\mathbf{J}_{\mathrm{uu}}$

Use hierarchical tensor basis for Q_2 elements on hexahedra:

$$A\mathbf{u} = \sum_{\text{elements } e} \mathcal{E}_e^{\mathrm{T}} D_\xi^{\mathrm{T}} \Lambda \Big((\nabla_{\mathbf{x}} \xi)^{\mathrm{T}} (\omega\eta) (\nabla_{\mathbf{x}} \xi) \Big) D_\xi \mathcal{E}_e \mathbf{u}$$

Reference derivative matrix D_ξ composed of $\hat{D} \otimes \hat{B} \otimes \hat{B}$, $\hat{B} \otimes \hat{D} \otimes \hat{B}$, and $\hat{B} \otimes \hat{B} \otimes \hat{D}$

Higher FLOP/Byte ratio and lower memory bandwidth requirements

Previous Work: AVX-Vectorization

Vectorize over elements: 4 elements per AVX register (double precision, 256 bits)

Details: May *et al.*, SC14 (2014)

GPU-accelerated SpMV, First Attempt

- One thread per element
- Same execution flow as CPU version
- Updates to result postponed and computed on host

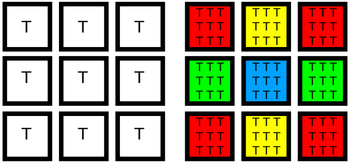
Observations

- It works!
- It's relatively slow (excessive register spilling)

mx=my=mz	AVX (1 proc)	CUDA	OpenCL
16	4.0	2.6	3.2
24	11.1	4.9	5.4
32	54.6	22.0	25.3
48	131.5	59.5	65.3

(Timings in seconds, Piz Daint prior to upgrade)

GPU-accelerated SpMV, Optimizations



- Stash GPU data to reduce host-device communication
- Use one warp per element
- Concurrent writes to result vector via atomics or coloring
- CUDA: Use warp shuffles for tensor computations

Observations

- GPU occupancy at 25 percent (enough to cover memory latencies)
- 166 GFLOPs (70 percent of 235, one FLOP per 4 warp shuffles)

mx=my=mz	AVX (1 proc)	CUDA, unoptimized	CUDA, optimized
16	4.0	2.6	1.0
24	11.1	4.9	2.2
32	54.6	22.0	8.1
48	131.5	59.5	14.6

(Timings in seconds, Piz Daint prior to upgrade)

Profiling Data for SpMV within Multigrid (Dual Xeon E5-2620 with Tesla K20)

Setup for SpMV (Gauss data, etc.): <1 percent

Copy field data: 21 percent

Kernel execution: **23 percent**

Copy result: 16 percent

Other (boundary conditions, etc.): 39 percent

mx=my=mz	AVX (1T)	AVX (2x12T)	AVX (12x2T)	CUDA	OpenCL
16	4.7s / 7.1	2.4s / 14.0	0.7s / 52.8	1.0s / 33.6	1.1s / 30.5
24	12.8s / 6.9	4.3s / 20.4	1.5s / 62.4	2.2s / 40.0	2.4s / 36.7
32	49.2s / 6.8	16.4s / 20.6	4.3s / 66.0	8.1s / 40.7	8.2s / 40.2
40	55.4s / 6.9	16.5s / 23.1	6.1s / 69.6	9.3s / 40.9	9.4s / 40.4
48	82.3s / 6.9	22.5s / 25.1	8.7s / 68.4	14.6s / 38.7	15.2s / 37.1

Time/GFLOPs

GPU-assisted Multigrid in StagYY

About StagYY

- Mantle convection solver
- Cartesian, 3D spherical shell, cylindrical domains
- Further details: P. Tackley, J. PEPI (2008)

Discretization and Solver

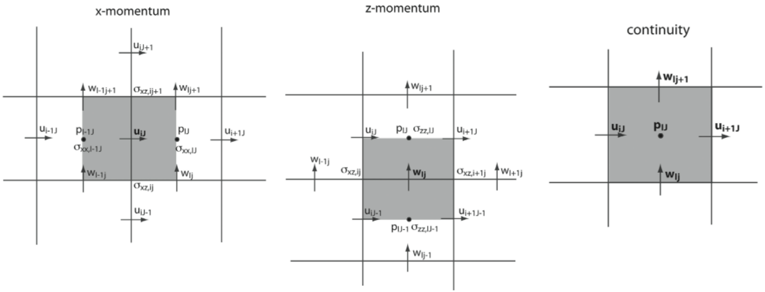
- Staggered differences finite volume method
- Multigrid solver (V- or F-cycles)
- Matrix-free residual evaluation and relaxation

Conservation Equations

$$\nabla \cdot (\rho \boldsymbol{v}) = 0 \qquad \text{(mass)}$$

$$\nabla \cdot \boldsymbol{\sigma} - \nabla p = \frac{\mathrm{Ra}.\boldsymbol{r}\rho(C, r, T)}{\Delta \rho_{\text{thermal}}} \qquad \text{(momentum)}$$

$$\rho C_{\mathrm{p}} \frac{\partial T}{\partial t} = -\mathrm{Di_s} \alpha \rho T v_r + \nabla \cdot (k \nabla T) + \rho H + \frac{\mathrm{Di_s}}{\mathrm{Ra}} \boldsymbol{\sigma} : \boldsymbol{\varepsilon} \qquad \text{(energy)}$$



(P. Tackley, Stagyy User Manual)

Iterated relaxation

```
- Update z-component of velocity
- Exchange values of vz at boundaries

- Compute pressure correction
- Exchange correction values at boundaries
- Apply pressure correction
- Update velocity based on new pressure
- Exchange boundary pressure and velocity

- Update y-component of velocity
- Exchange values of vy at boundaries

- Update x-component of velocity
- Exchange values of vx at boundaries
```

All moments simultaneously

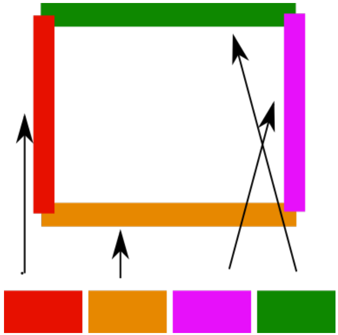
```
- Compute updates for all velocity components
- Exchange boundary pressure and velocity

- Compute correction for pressure p
- Exchange correction values at boundaries
- Apply pressure correction
- Update velocity based  on new pressure
- Exchange boundary pressure and velocity
```

GPU Data Handling

- Fields on each multigrid hierarchy reside on GPU
- Stack-like mechanism for resident GPU data

GPU Boundary Value Handling

- pTatin3d results: Pay attention to this
- MPI-like `gather` and `scatter` implemented



GPU Acceleration

One thread per residual entry

Common kernel code path for CUDA and OpenCL

Observations

It works!

Kernels fairly fast out-of-the-box!

nxtot=nytot=nztot	GPU, CUDA, xyz	GPU, CUDA, classic	Sequential	8 MPI ranks
32	0.27	0.28	0.36	0.08
64	1.21	1.39	3.04	0.50
128	5.66	5.33	24.4	4.4
256	38.4	25.7	timeout	36.1

(Timings in seconds, Piz Daint after upgrade)

GPU Profiling

60-65 percent of GPU-time spent on PCI-Express transfer

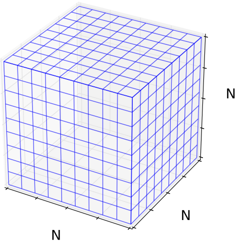
Checked: No unnecessary transfers, full bandwidth, etc.

nxtot=nytot=nztot	GPU, CUDA, xyz	GPU, CUDA, classic
CUDA memcpy HtoD	39.03%	49.98%
CUDA memcpy DtoH	20.78%	16.03%

Pitfall: GPUs are too fast for PCI-Express

Latest GPU peaks: 720 GB/sec from GPU-RAM, 16 GB/sec for PCI-Express

40x imbalance (!)



Compute vs. Communication

Take $N = 512$, so each field consumes 1 GB of GPU RAM

Boundary communication: $2 \times 6 \times N^2$: 31 MB

Time to process field on GPU: 1.4 ms

Time to load ghost data: **1.9 ms (!!)**

Summary

GPU-SpMV in pTatin3d

- 166 GFLOPs in GPU kernel achieved
- One thread per Q2 quadrature point
- Overhead due to host-device communication significant

GPU-assisted Multigrid in StagYY

- Residual evaluation and relaxations
- Reduced boundary data transfer between host and GPU (cf. gather and scatter)
- Heavily PCI-Express bandwidth limited

General Conclusions

- Starvation of modern GPUs for multigrid when attached via PCI-Express
- Kernel-level GPU integration insufficient for multigrid purposes