Fast solvers for implicit Runge-Kutta

Jed Brown jedbrown@mcs.anl.gov (ANL and CU Boulder)
Debojyoti Ghosh (ANL)

Copper Mountain, 2014-04-08
This talk: http://59A2.org/files/20140408-FastIRK.pdf

Motivation

- Hardware trends
  - Memory bandwidth a precious commodity (8+ flops/byte)
  - Vectorization necessary for floating point performance
  - Conflicting demands of cache reuse and vectorization
  - Can deliver bandwidth, but latency is hard
- Assembled sparse linear algebra is doomed!
  - Limited by memory bandwidth (1 flop/6 bytes)
  - No vectorization without blocking
- Spatial-domain vectorization is *intrusive*
  - Must be unassembled to avoid bandwidth bottleneck
  - Whether it is “hard” depends on discretization
  - Geometry, boundary conditions, and adaptivity

Sparse linear algebra is dead (long live sparse . . .)

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

- Problem: popular algorithms have nested data dependencies
  - Time step
    - Nonlinear solve
      - Krylov solve
        - Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix while in cache)

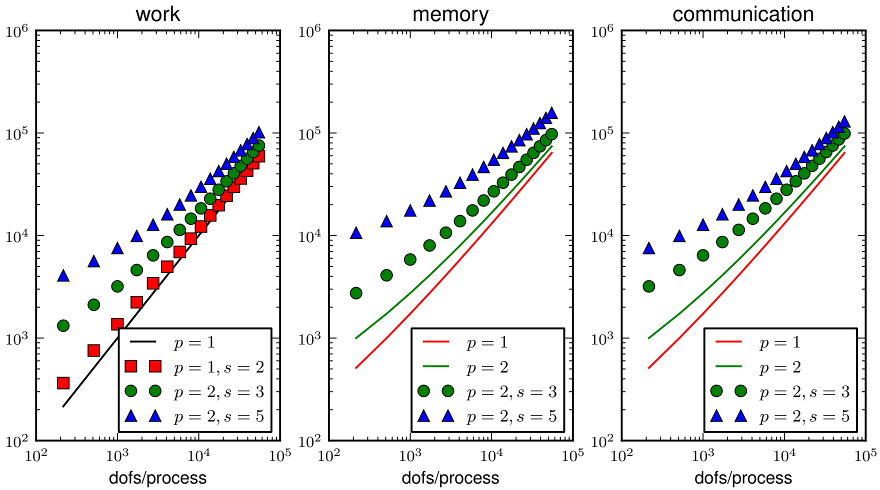
Sparse linear algebra is dead (long live sparse . . .)

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

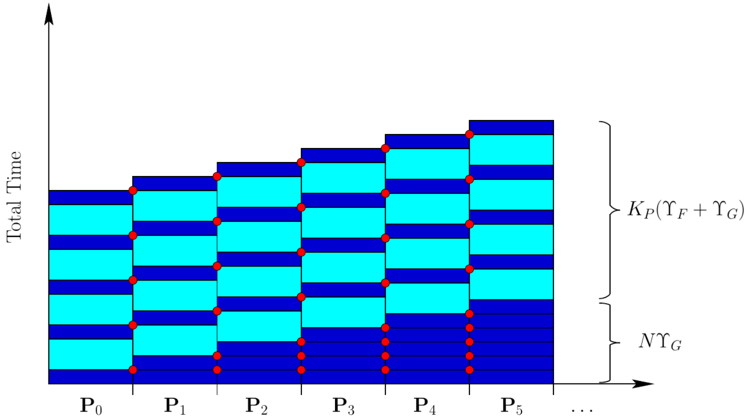
- Problem: popular algorithms have nested data dependencies
  - Time step
    - Nonlinear solve
      - Krylov solve
        - Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix while in cache)

Attempt: *s*-step methods in 3D



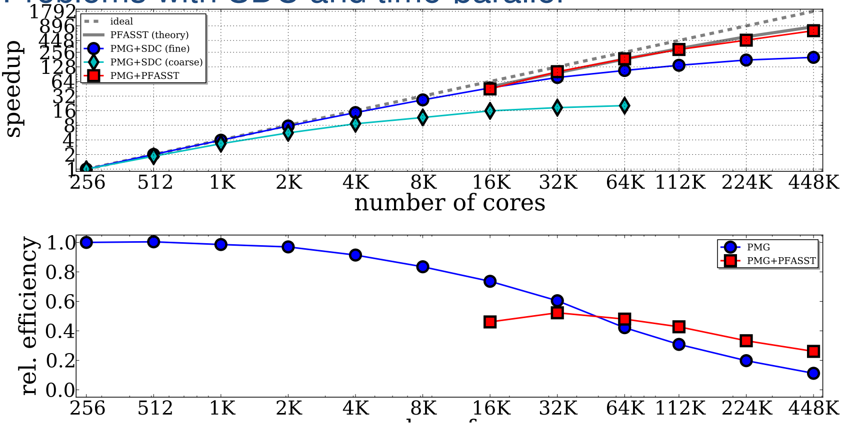
- Limited choice of preconditioners (none optimal, surface/volume)
- Amortizing message latency is most important for strong-scaling
- *s*-step methods have high overhead for small subdomains

Attempt: distribute in time (multilevel SDC/Parareal)



- PFASST algorithm (Emmett and Minion, 2013)
- Zero-latency messages (cf. performance model of *s*-step)
- Spectral Deferred Correction: iterative, converges to IRK (Gauss, Radau, ...)
- Stiff problems use implicit basic integrator (synchronizing on spatial communicator)

Problems with SDC and time-parallel



c/o Matthew Emmett, parallel compared to sequential SDC

- Iteration count not uniform in *s*; efficiency starts low
- Low arithmetic intensity; tight error tolerance (cf. Crank-Nicolson)
- Parabolic space-time (Greenwald and Brandt; Horton and Vandewalle)

Runge-Kutta methods

$$\dot{u} = F(u)$$

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}}_{Y} = u^n + h \underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{ss} \end{bmatrix}}_{A} F \begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}$$

$$u^{n+1} = b^T Y$$

- General framework for one-step methods
- Diagonally implicit: A lower triangular, stage order 1 (or 2 with explicit first stage)
- Singly diagonally implicit: all A_{ii} equal, reuse solver setup, stage order 1
- If A is a general full matrix, all stages are coupled, “implicit RK”

Implicit Runge-Kutta

$$
\begin{array}{c|ccc}
\frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\
\frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\
\frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}
$$

- Excellent accuracy and stability properties
- Gauss methods with s stages
  - order $2s$, (s, s) Padé approximation to the exponential
  - A-stable, symplectic
- Radau (IIA) methods with s stages
  - order $2s-1$, A-stable, L-stable
- Lobatto (IIIC) methods with s stages
  - order $2s-2$, A-stable, L-stable, self-adjoint
- Stage order s or $s+1$

Method of Butcher (1976) and Bickart (1977)

- Newton linearize Runge-Kutta system at u^*

$$Y = u^n + hAF(Y) \qquad \left[I_s \otimes I_n + hA \otimes J(u^*)\right] \delta Y = RHS$$

- Solve linear system with tensor product operator

$$\hat{G} = S \otimes I_n + I_s \otimes J$$

where $S = (hA)^{-1}$ is $s \times s$ dense, $J = -\partial F(u)/\partial u$ sparse

- SDC (2000) is Gauss-Seidel with low-order corrector
- Butcher/Bickart method: diagonalize $S = X \Lambda X^{-1}$
  - $\Lambda \otimes I_n + I_s \otimes J$
  - s decoupled solves
  - Complex eigenvalues (overhead for real problem)
- Problem: X is exponentially ill-conditioned wrt. s
- We avoid diagonalization
  - Permute $\hat{G}$ to reuse J: $G = I_n \otimes S + J \otimes I_s$
  - Stages coupled via register transpose at spatial-point granularity
  - Same convergence properties as Butcher/Bickart

MatTAIJ: “sparse” tensor product matrices

$$G = I_n \otimes S + J \otimes T$$

- J is parallel and sparse, S and T are small and dense
- More general than multiple RHS (multivectors)
- Compare $J \otimes I_s$ to multiple right hand sides in row-major
- Runge-Kutta systems have $T = I_s$ (permuted from Butcher method)
- Stream J through cache once, same efficiency as multiple RHS
- Unintrusive compared to spatial-domain vectorization or s-step

Convergence with point-block Jacobi preconditioning

- 3D centered-difference diffusion problem

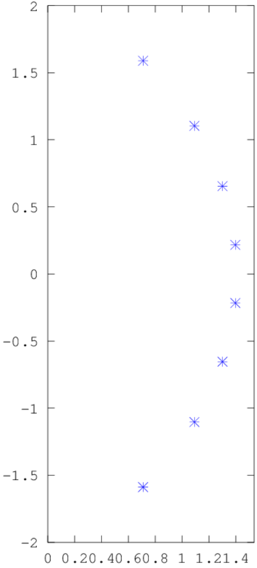
Method	order	nsteps	Krylov its.	(Average)
Gauss 1	2	16	130	(8.1)
Gauss 2	4	8	122	(15.2)
Gauss 4	8	4	100	(25)
Gauss 8	16	2	78	(39)

We really want multigrid

- Prolongation: $P \otimes I_s$
- Coarse operator: $I_n \otimes S + (RAP) \otimes I_s$
- Larger time steps
- GMRES(2)/point-block Jacobi smoothing
- FGMRES outer

Method	order	nsteps	Krylov its.	(Average)
Gauss 1	2	16	82	(5.1)
Gauss 2	4	8	64	(8)
Gauss 4	8	4	44	(11)
Gauss 8	16	2	42	(21)

Toward a better AMG for IRK/tensor-product systems



- Start with $\hat{R} = R \otimes I_s$, $\hat{P} = P \otimes I_s$

$$G_{\text{coarse}} = \hat{R}(I_n \otimes S + J \otimes I_s)\hat{P}$$

- Imaginary component slows convergence
- Idea: rotate eigenvalues on coarse levels
  Erlangga and Nabben *On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian*

Implicit Runge-Kutta for advection

Table: Total number of iterations (communications or accesses of J) to solve linear advection to $t = 1$ on a 1024-point grid using point-block Jacobi preconditioning of implicit Runge-Kutta matrix. The relative algebraic solver tolerance is 10^{-8}.

Family	Stages	Order	Iterations
Crank-Nicolson/Gauss	1	2	3627
Gauss	2	4	2560
Gauss	4	8	1735
Gauss	8	16	1442

- Naive centered-difference discretization
- Leapfrog requires 1024 iterations at CFL=1
- This is *A*-stable (can handle dissipation)

Outlook on IRK

- IRK *unintrusively* offers bandwidth reuse and vectorization
- No need for complex arithmetic (cf. Butcher and Bickart)
- Need polynomial smoothers for IRK spectra
- Change number of stages on spatially-coarse grids (p-MG, or even increase)?
- Experiment with SOR-type smoothers
  - Prefer point-block Jacobi in smoothers for parallelism
- Study efficiency for nonlinear problems
- Is it possible to speed up advection?
- Possible IRK correction for IMEX (non-smooth explicit function)
- PETSc implementation (parallel example running, interface in-progress)