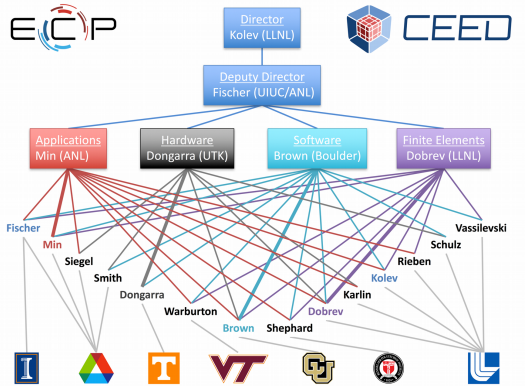
Center for Efficient Exascale Discretization

Jed Brown (CU Boulder) and the CEED team

Multicore 7 workshop, 2017-09-28

Goals & Team

- CEED is focused on the *development of next-generation discretization software and algorithms to enable efficient simulations for a wide range of science applications on future HPC systems*.
- Funding: $3.0M/year, 2 labs (LLNL, ANL), 5 universities

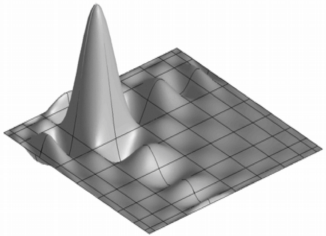


30+ researchers

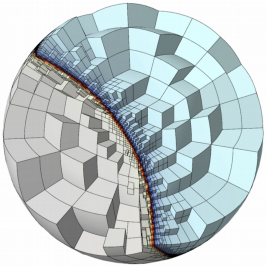
Co-design Motifs

- PDE-based simulations on **unstructured grids**
- **high-order** and **spectral** finite elements

✓*any order space on any order mesh* ✓*curved meshes,*
✓*unstructured AMR* ✓*optimized low-order support*

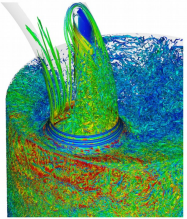


10th order basis function



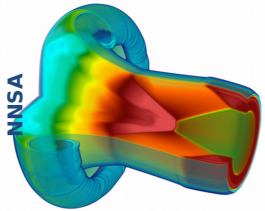
non-conforming AMR, 2nd order mesh

incompressible SEM / SC



6th order DNS turbulence (Nek)

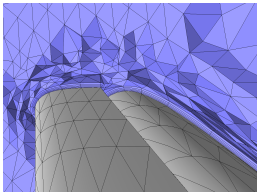
compressible FEM / NNSA



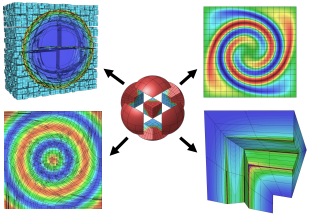
2nd order compressible shock hydro (MFEM)

High-Order Software Ecosystem

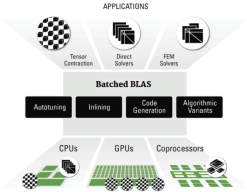
CEED
EXASCALE DISCRETIZATIONS



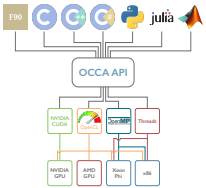
High-order Meshes



Unstructured AMR



Tensor contractions

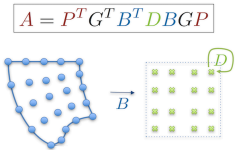


Performance portability

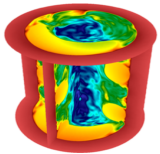
PETSc

hypre
high performance preconditioners

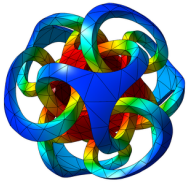
Scalable matrix-free solvers



High-Order Operator Format



General Interpolation



High-Order Visualization

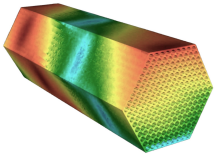
More info at: http://ceed.exascaleproject.org/fe

ECP EXASCALE COMPUTING PROJECT

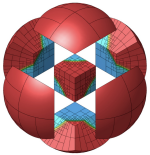
CEED Software Products

CEED's library model enables ECP apps to easily take advantage of the new discretization technologies

- state-of-the art CEED **discretization libraries**
  - ✓ *better exploit the hardware to deliver significant performance gain over conventional methods*
  - ✓ *based on MFEM/Nek, low & high-level APIs*



nek5000.mcs.anl.gov
High-performance spectral elements



mfem.org
Scalable high-order finite elements

CEED's proxies and general purpose libs target ECP vendors, STs, broader community

- **Ceedlings** - CEED kernels, bake-off probs & miniapps
  - ✓ *main tools to engage vendors & external projects*
- CEED **broadly applicable libraries**

MAGMA
icl.cs.utk.edu/magma
LAPACK for GPUs, multi/many-core

libocca.org
Lightweight performance portability

GSLIB

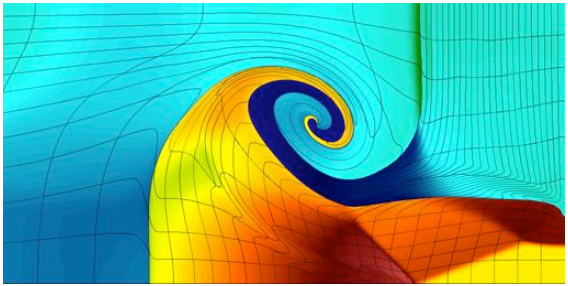
PUMI
Parallel Unstructured Mesh Infrastructure

Holmes

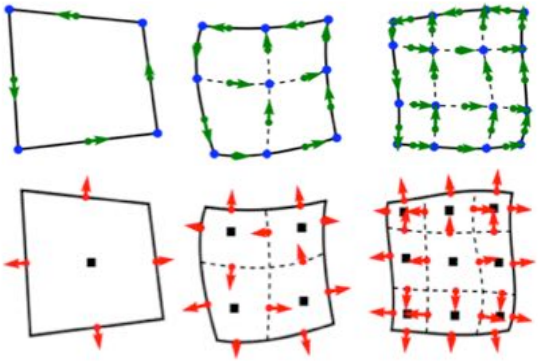
PETSc

Main deliverable: all CEED software freely available on GitHub at https://github.com/CEED
New releases: mfem-3.3, gslib, Laghos and NekCEM ceedling, …

Applicable to variety of physics



Compressible flow (ALE, 8th order)



Linear, quadratic and cubic finite element spaces on curved meshes

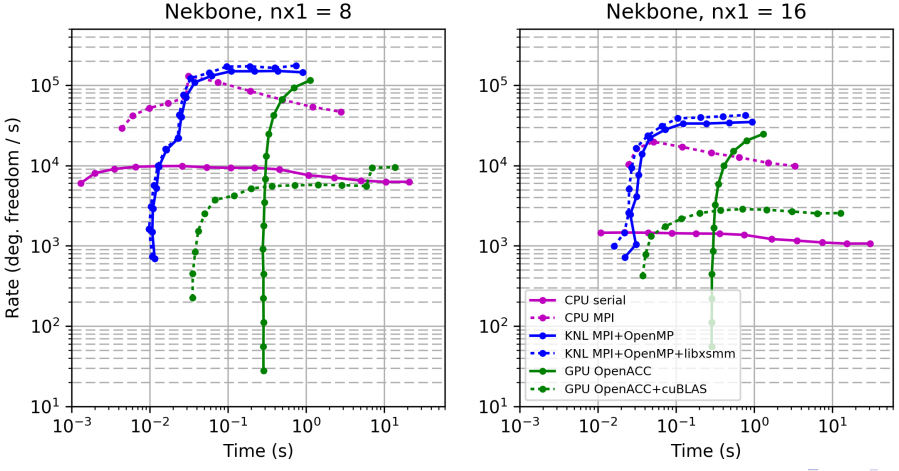
$H(grad)$ "nodes" $\xrightarrow{\nabla}$ $H(curl)$ "edges" $\xrightarrow{\nabla\times}$ $H(div)$ "faces" $\xrightarrow{\nabla\cdot}$ L_2 "zones"

High-order kinematics | High-order MHD | High-orde rad. diff. | High-order thermodynamics

de Rham complex

Nekbone

- Conjugate gradient spectral element benchmark with sum factorization
- Without multigrid preconditioner – a significant and interesting factor for Nek5000



$n_{1/2}$ and $t_{1/2}$

- Suppose a linear scaling algorithm
- Let $r(n)$ be the performance rate (e.g., DOF/second or GF/s) for local problem size $n = N/P$
- Let $r_{\max} = \max_n r(n)$ be the peak attainable performance
- $n_{1/2} = \min\{n : r(n) \geq \frac{1}{2} r_{\max}\}$
- $t_{1/2} = 2n_{1/2}/r_{\max}$

CEED Bake-Off Problems

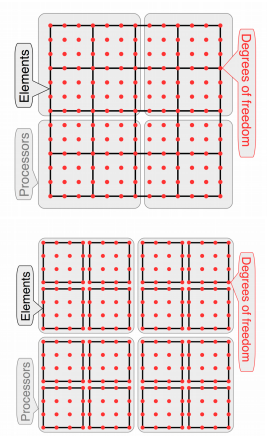
BP1: Solve {Bu=f}, where {B} is the mass matrix.

BP2: Solve the vector system $\{Bu_i=f_i\}$ with {B} from BP1.

BP3: Solve {Au=f}, where {A} is the Poisson operator.

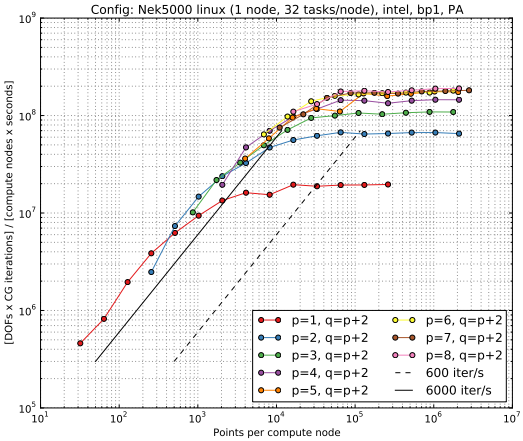
BP4: Solve the vector system $\{Au_i=f_i\}$ with {A} from BP3.

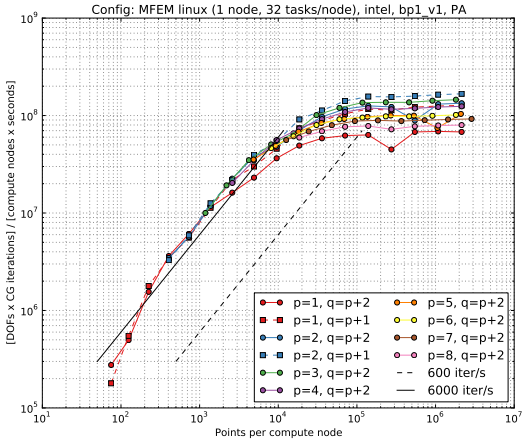
- Range of polynomial orders: {p=1, 2,...,8}, at least.
- Cover range of sizes: from 1 element/MPI rank up to the memory limit.
- BP1 and BP2 are relevant for many hyperbolic substeps in transport problems. BP3 and BP4 reflect pressure, momentum, and diffusion updates in fluid/thermal transport.
- Vector forms BP2 and BP4 reveal benefits of increased *data reuse* and of *amortized communication overhead.*
- Benchmark repo: https://github.com/CEED/benchmarks



BP terminology: T- and E-vectors of HO dofs

BP1 on KNL: Nek5000 and MFEM



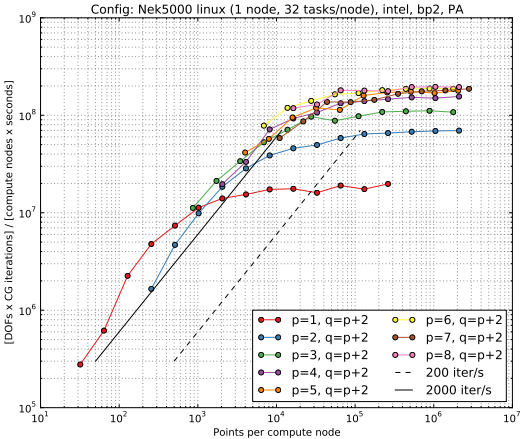


Nek5000 $n_{1/2} = 15k, t_{1/2} = 150\mu s$

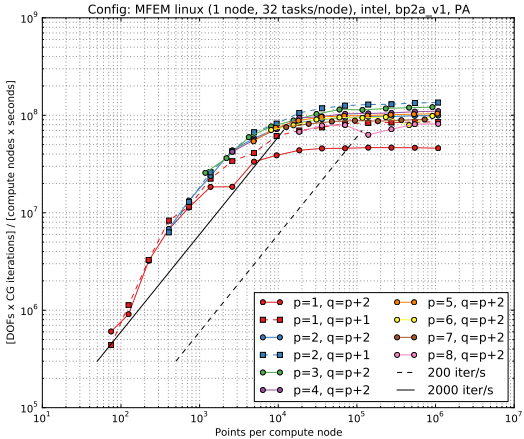
MFEM $n_{1/2} = 10k, t_{1/2} = 400\mu s$

- BG/Q has similar performance

BP2 on KNL: Nek5000 and MFEM

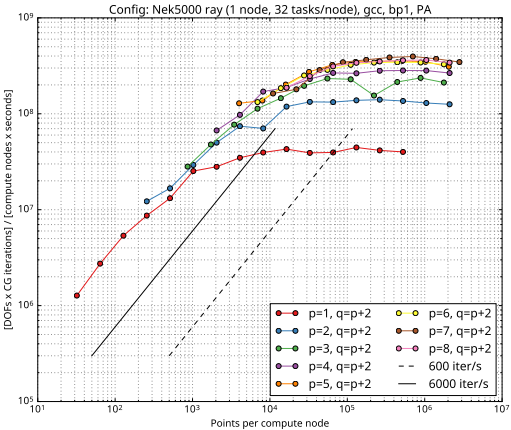


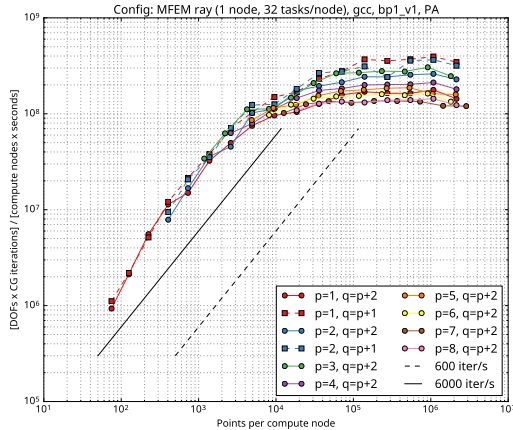
Nek5000 $n_{1/2} = 30k, t_{1/2} = 300\mu s$



MFEM $n_{1/2} = 15k, t_{1/2} = 300\mu s$

BP1 on Power8: Nek5000 and MFEM





- Nek5000 $t_{1/2} = 75\mu s$, MFEM $t_{1/2} = 200\mu s$

Table: Performance results for various machines using metrics defined in Sec. **??**. Each entry corresponds to Nek/MFEM results. Results sorted by problem and $t_{\frac{1}{2}}$ performance.

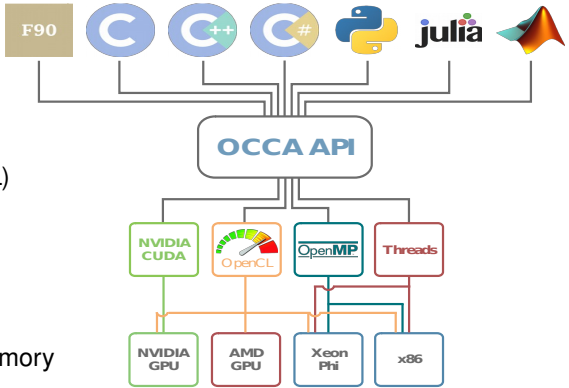
Machine/Problem	r_{max} (MDOFs)	$n_{\frac{1}{2}}$ (KDOFs)	$t_{\frac{1}{2}}$ (ms)
Ray/BP1-gcc	400/350	15/35	0.075/0.200
KNL32/BP1-intel	200/150	15/30	0.150/0.400
KNL32/BP1-gcc	100/100	20/8	0.400/0.160
Vulcan/BP1-gcc	35/40	15/10	0.857/0.500
KNL32/BP2-intel	200/100	30/15	0.300/0.300
KNL32/BP2-gcc	100/100	50/15	1.000/0.300
Vulcan/BP2-gcc	35/30	21/10	1.200/0.666
Vulcan/BP3-gcc	/30	/10	/0.666
Vulcan/BP3-xlc	/20	/10	/1.000
Vulcan/BP4-gcc	/20	/10	/1.000

Lightweight Performance Portability

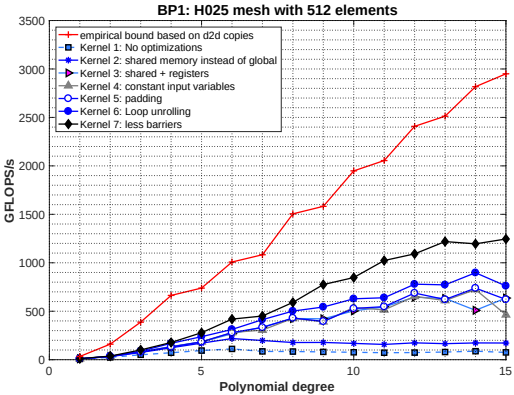
CEED/OCCA is an open-source library that provides an unified API for programming different types of devices, including CPUs, GPUs, Intel's Xeon Phi, FPGAs.

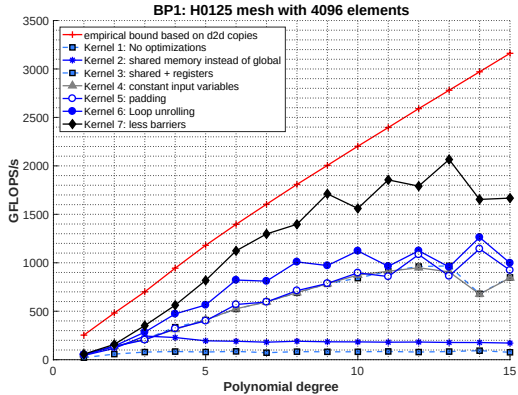
Features:

- Supported on many languages, such as C++, C, and Fortran
- *JIT compilation for kernels*
- Single kernel language for all backends (OKL)
- Currently supports Serial, OpenMP, CUDA, and OpenCL backends. Works with MPI
- MIT License, http://www.libocca.org
- Extensible backend API, allowing for future features. For example, support for unified memory in CUDA and mapped memory in OpenCL



OCCA optimizations for NVIDIA P100





Batched Computing Technology

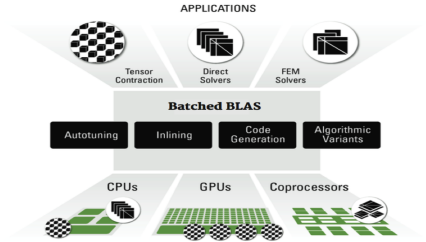
- Matrix-free basis evaluation needs efficient tensor contractions, e.g.,

$$C_{i1,i2,i3} = \sum_k A_{k,i1} B_{k,i2,i3}$$

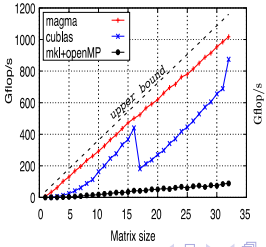
- **CEED/MAGMA** designed batched methods to split the computation in many small high-intensity GEMMs, grouped together (batched) for efficient execution:

Batch_{ $C_{i3} = A^T B_{i3}$, for range of i3 }

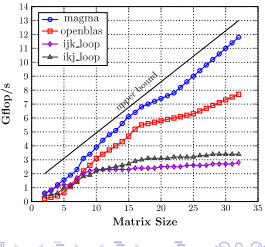
- Developed techniques needed for autotuning, code inlining, code generation (reshapes, etc.), algorithmic variants for different architectures.
- Achieve 90+% of theoretically derived peaks.
- Significantly outperform vendor libraries.
- Released through MAGMA.



Batched DGEMMs on GPU (P100, 100K)

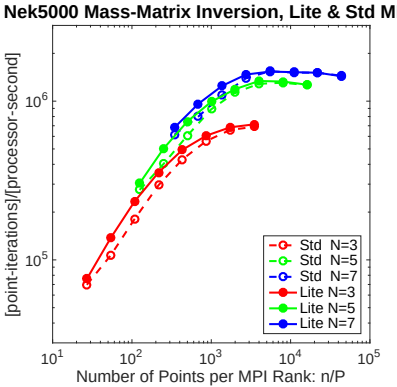


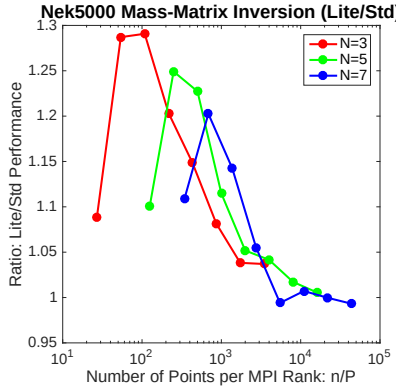
Batched DGEMMs on ARM (Tegra X1 : 4-core Cortex A57)



MPICH CH4: lightweight device layer

- CH4: faster offload, better fast path/inlining/IPO





High-level API

$$v^T F(u) \sim \int_\Omega v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u) \qquad v^T J w \sim \int_\Omega \begin{bmatrix} v \\ \nabla v \end{bmatrix}^T \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w \\ \nabla w \end{bmatrix}$$

$$u_e = B \mathscr{E}_e u \qquad \nabla u_e = \frac{\partial X}{\partial x} D \mathscr{E}_e u$$

$$J w = \sum_e \mathscr{E}_e^T \begin{bmatrix} B \\ D \end{bmatrix}^T \underbrace{\begin{bmatrix} I & \\ & \left(\frac{\partial X}{\partial x}\right)^T \end{bmatrix} W_q \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} I & \\ & \left(\frac{\partial X}{\partial x}\right) \end{bmatrix}}_{\text{coefficients at quadrature points}} \begin{bmatrix} B \\ D \end{bmatrix} \mathscr{E}_e w$$

- B and D are tensor contractions – independent of element
- Choice of how to order and represent gathers $\mathscr{E}$ and scatters $\mathscr{E}^T$
- Who computes the metric terms and other coefficients?
- Similar for Neumann/Robin and nonlinear boundary conditions

High-level API Proposal

$$v^T J w = \sum_e \mathscr{E}_e^T \begin{bmatrix} B \\ D \end{bmatrix}^T \underbrace{\begin{bmatrix} I & \\ & \left(\frac{\partial X}{\partial x}\right)^T \end{bmatrix} W_q \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} I & \\ & \left(\frac{\partial X}{\partial x}\right) \end{bmatrix}}_{\text{coefficients at quadrature points}} \begin{bmatrix} B \\ D \end{bmatrix} \mathscr{E}_e w$$

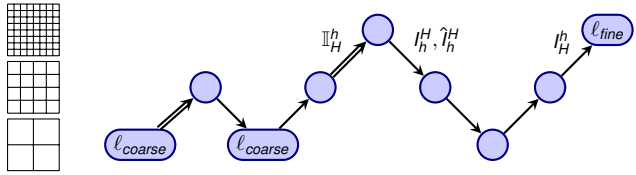
- Proposal:
  - Setup: specify $\mathscr{E}$ and local ordering choice for f
  - Apply: implementation handles batching, work buffers, and calling $f(u, \nabla u)$.
- User links to interface library
- Backend implementation switchable at run-time
- Two-phase implementation enables connectivity analysis and JIT

HPGMG: a benchmark for supercomputers

- `https://hpgmg.org`, hpgmg-forum@hpgmg.org mailing list
- Mark Adams, Sam Williams (finite-volume), Jed (finite-element), John Shalf, Brian Van Straalen, Erich Strohmeier, Rich Vuduc
- Gathering momentum, annual BoFs at Supercomputing since 2014
- Implementations

  Finite Volume memory bandwidth intensive, simple data dependencies, 4th order
  Finite Element compute- and cache-intensive, vectorizes, overlapping writes
- Full multigrid, well-defined, scale-free problem
- Matrix-free operators, Chebyshev smoothers

Full Multigrid (FMG): Prototypical Fast Algorithm

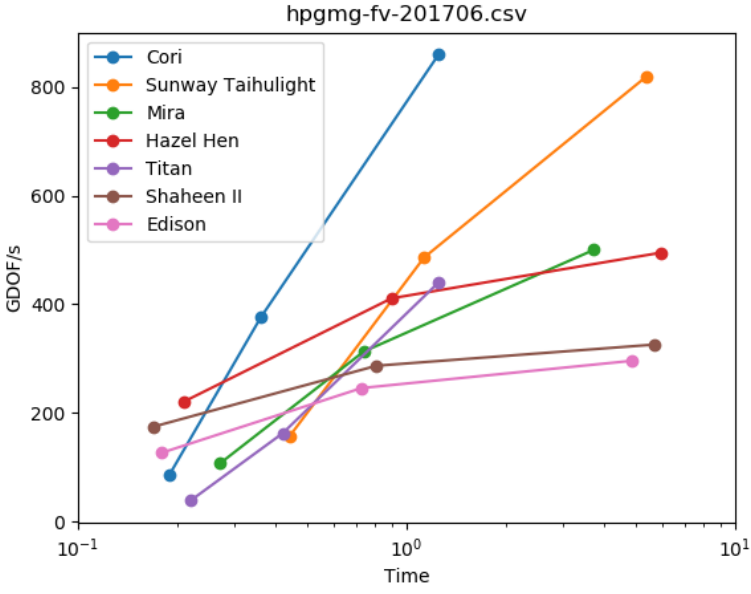


- start with coarse grid
- truncation error within one cycle
- about five work units for many problems
- no “fat” left to trim – robust to gaming
- distributed memory – restrict active process set using Z-order
  - $\mathcal{O}(\log^2 N)$ parallel complexity stresses network
- scale-free specification
  - no mathematical reward for decomposition granularity
  - don’t have to adjudicate “subdomain”

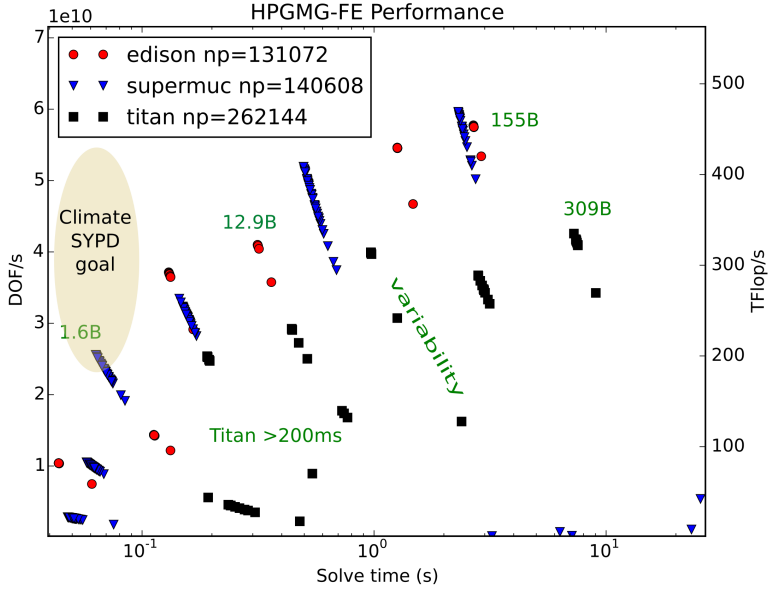
Multigrid design decisions

- Q_2 finite elements
  - Partition of work not partition of data – sharing/overlapping writes
  - Q_2 is a middle-ground between lowest order and high order
  - Matrix-free pays off, tensor-product element evaluation
- Linear elliptic equation with manufactured solution
- Mapped coordinates
  - More memory streams, increase working set, longer critical path
- No reductions
  - Coarse grid is strictly more difficult than reduction
  - Not needed because FMG is a direct method
- Chebyshev/Jacobi smoothers, $V(3,1)$ cycle
  - Multiplicative smoothers hard to verify in parallel
  - Avoid intermediate scales (like Block Jacobi/Gauss-Seidel)
- Full Approximation Scheme

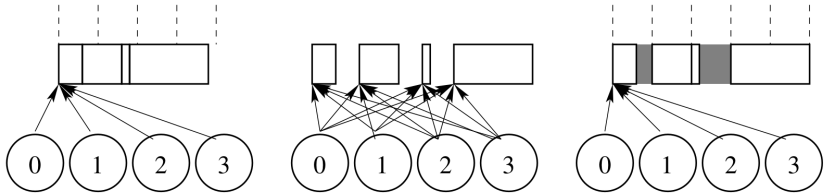
2017 HPGMG performance spectra



HPGMG-FE on Edison, SuperMUC, Titan



MPI-3: Halos or contiguous memory?



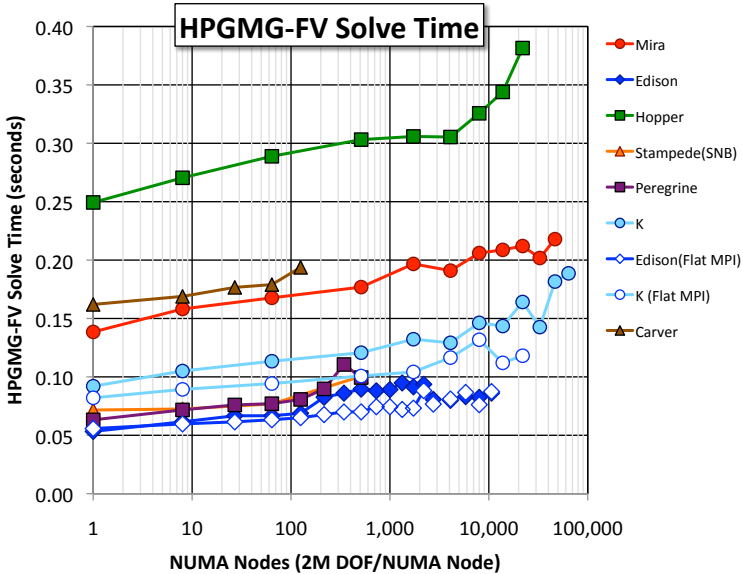
[Hoefler at al, 2013]

- Common assumption: halo copying is expensive
- Alternative is shared memory
- Cache utilization for 16^3 local domain with halos
  - Entire local region is contiguous; no partially filled cache lines
  - $18^3 * \texttt{sizeof(double)} = 46656B$
- 16^3 local domain embedded in contiguous memory
  - Avoid false sharing: align owned portion to cache-line boundaries
  - $32 \times 18 \times 18 * \texttt{sizeof(double)} = 82944B$
  - False sharing a serious problem if local sizes not divisible by line size

Messaging from threaded code

- Off-node messages need to be packed and unpacked
- Many MPI+threads apps pack in serial – bottleneck
- Extra software synchronization required to pack in parallel
  - Formally $O(\log T)$ critical path, T threads/NIC context
  - Typical OpenMP uses barrier – oversynchronizes
- `MPI_THREAD_MULTIPLE` – atomics and $O(T)$ critical path
- Choose serial or parallel packing based on T and message sizes?
- Is there always at least one hardware NIC context/core?
- What is lowest overhead approach to message coalescing?

HPGMG-FV: flat MPI vs MPI+OpenMP (Aug 2014)



Scalable Matrix-Free Solvers

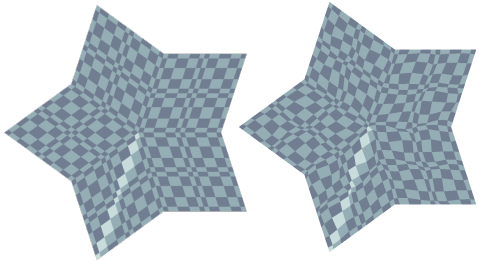
CEED
EXASCALE DISCRETIZATIONS

Topics

- *Preconditioning – critical for implicit apps*
- *Active research: p-Multigrid, sparsification, DD, high-order operator-based, ...*
- *Partners: PETSc, hypre, KAUST, ...*

PETSc

hypre
high performance preconditioners



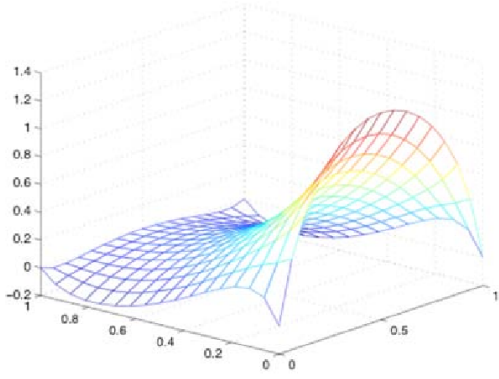
CEED Contacts

- Jed Brown
- Tzanio Kolev
- Panayot Vassilevski
- Tim Warburton
- Paul Fischer

More info at: http://ceed.exascaleproject.org/linalg

ECP EXASCALE COMPUTING PROJECT

Adaptive BDDC: robust coarsening/smoothing



[Mandel and Sousedik 2010]

- Theory: Mandel, Sousedík, Spillane, Pechstein, Dohrmann, Widlund
- Collaboration with Stefano Zampini (author of PETSc's PCBDDC)

- only low-order continuity between subdomains
- corrected by more technical subdomain smoother
- "deluxe" face balancing operator
- adaptive coarse spaces
  - solve face eigenproblems
  - choose coarse space using eigenvalue threshold λ_{thresh}
  - $\lambda_{\text{threshold}}$ a sharp, analytic function of global condition number
- guarantees convergence rate, not complexity – adaptive agglomeration

Implicit Runge-Kutta and MatKAIJ: “sparse” Kronecker products

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

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

$$u^{n+1} = u^n + h b^T F(Y)$$

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

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