Fifteen Years of PFLOTRAN: Growing a state-of-the-art hydrologic flow and reactive transport simulator with PETSc —A retrospective and prospective look

Richard Tran Mills

June 17, 2015

Credits

R. Mills is the speaker, but this talk summarizes collaborative efforts of many people, and contains contributed material or results from

- Gautam Bisht, LBNL
- Nathan Collier, ORNL
- Lois Curfman-McInnes, ANL
- Glenn Hammond, SNL
- Satish Karra, LANL
- Jitendra Kumar, ORNL
- Peter Lichtner, OFM Research
- Barry Smith, ANL
- Vamsi Sripathi, Intel
- Hong Zhang, IIT and ANL

Notice and Disclaimers

Optimization notice

Optimization Notice

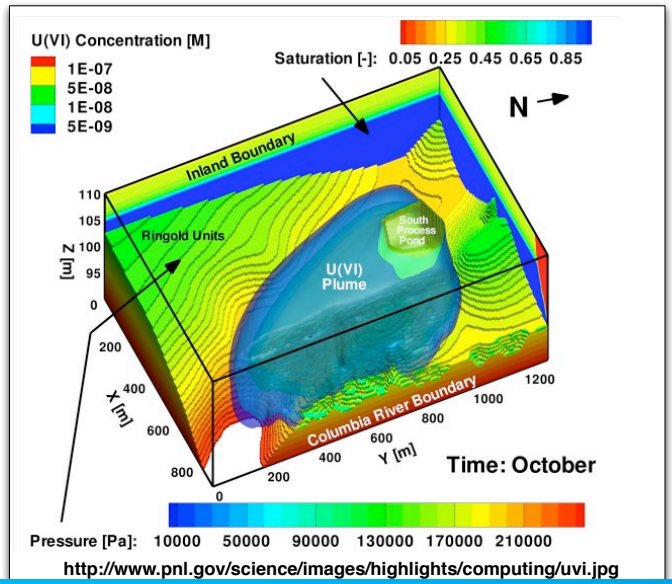
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Introduction/Outline

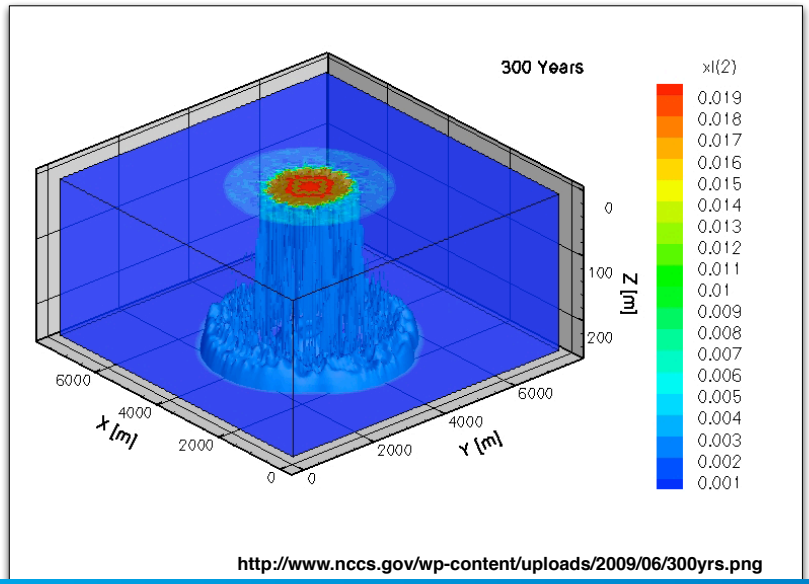
- The PFLOTRAN team has been using PETSc for 15 years, and this has enabled tremendous progress.
- Look back: How has PETSc helped the PFLOTRAN team solve their science programs on leadership-class computing platforms? How have PFLOTRAN team's needs shaped PETSc?
- The PETSc devs are not just world class solver library developers, but world class at library-application "co-design".
- PETSc stays relevant because the PETSc team listens to what application teams need—no throwing "solutions" over a fence.
- Look forward: Where do PFLOTRAN and PETSc go over the next 15 years, as the science goals and computing platforms evolve?

PFLOTRAN

- Open-source (download at bitbucket.org/pflotran) code that simulates multiscale-multiphase-multicomponent reacting flows in porous media
- FV and ~~MFD~~ discretizations; implicit or operator-split timestepping
- Multiple interacting continua; supercritical CO_2; geomechanics
- Comprehensive biogeochemistry: Ion activity models, ion exchange, aqueous speciation, aqueous-gas mass transfer, mineral precip./dissolution, sorption isotherms, surface complexation (equilibrium, kinetic, multirate), colloids, microbial reactions
- Built from ground-up with parallel scalability in mind; scales to O(100,000) cores on leadership-class supercomputers



http://www.pnl.gov/science/images/highlights/computing/uvi.jpg



http://www.nccs.gov/wp-content/uploads/2009/06/300yrs.png

Building PFLOTRAN with PETSc

- PFLOTRAN is built on top of the open-source Portable, Extensible Toolkit for Scientific Computing (PETSc). http://www.mcs.anl.gov/petsc
- PETSc provides highly scalable nonlinear and linear solvers and preconditioners, time steppers, parallel mesh management, multi-physics coupling capabilities, performance profiling, and more.
- PETSc has a well-established community of developers and users
  - PETSc has reached the legal drinking age! (In Canada, anyway.)
  - Dozens of individual contributors, hundreds of users
  - Used in over 629 publications
  - 3 Gordon Bell Prizes (+ another finalist)
  - Very active user and developer mailing lists

Building PFLOTRAN with PETSc

- Interaction between the PFLOTRAN and PETSc communities has been critical to our success!
- PFLOTRAN source code stays at manageable size because parallelism and solvers are handled by PETSc.
- PFLOTRAN requirements lead to improvements in PETSC:
  - Universal F90 interface
  - Improved BiCGStab implementation
  - Hierarchical/nested Krylov methods
- Various improvements made by PETSc community can immediately be leveraged by PFLOTRAN
- Core PETSc developers are members of the PFLOTRAN team, allowing rapid cross-fertilization of knowledge.

PFLOTRAN Collaboration

PFLOTRAN development is a collaborative effort with contributors from many national laboratories, universities, and private industry.

- DOE National Laboratories
  - ANL
  - INL
  - LANL
  - LBNL
  - ORNL
  - PNNL
  - SNL
- Industry
  - BLOS International
  - AMPHOS[21]
- Universities
  - U. Arizona
  - U.C. Berkeley
  - C.U. Boulder
  - University of Chicago
  - Colorado School of Mines
  - U. of Bern
  - U. of Illinois at Urbana-Champaign
  - Illinois Institute of Technology
  - University of Michigan
  - Oregon State University
  - University of Tennessee, Knoxville
  - U. of Texas at Austin
  - U. of Wyoming

PFLOTRAN Early History: 2000–2006

2000: G. Hammond DOE CSGF practicum with P. Lichtner

- Initial PTRAN implementation. Using PETSc DA and SLES.
- One of the first Fortran PETSc users: Helped harden Fortran interfaces.
- Solidified relationship with PETSc developers to the point that "Barry and Matt could cuss us out and we wouldn't get offended." —Glenn

2003: R. Mills DOE CSGF practicum with P. Lichtner

- Initial PFLOW implementation. First use of SNES.

2004–2006: C. Lu postdoc w/ P. Lichtner

- Development of multiphase modules.

2005: R. Mills working with Cray X1 at ORNL

- CSRPERM matrix class (hybrid jagged-diagonal/ELLPACK format) implemented for vector machines.
- First contribution to PETSc.
- First time using distributed revision control (BitKeeper).

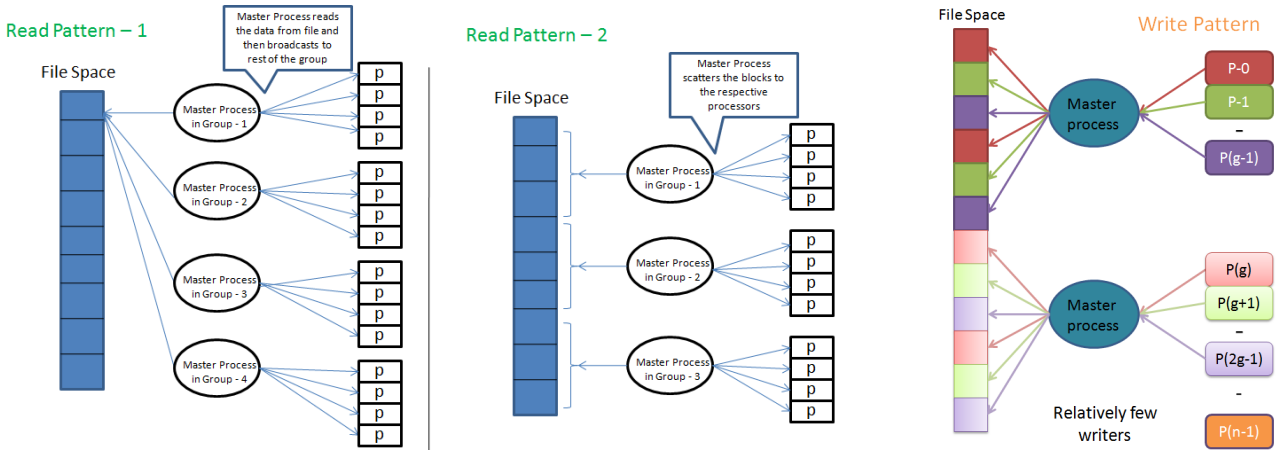
SciDAC-II: 2006–2011

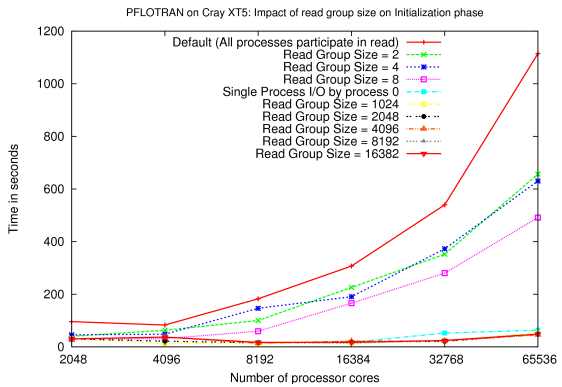
SciDAC-II funding (2006–2011) enabled intensive development.
INCITE (2007–2012) provided Jaguar Cray XT compute cycles.

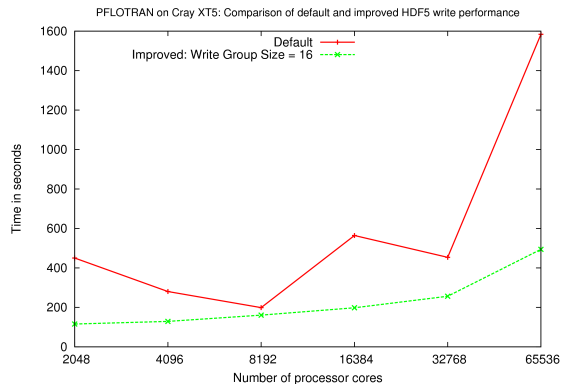
- Difficulties with compilers on Jaguar led to creation of "universal" F90 array interface (R. Mills and S. Balay, PETSc 2.3.3).
- 3rd Jaguar upgrade (2007) brought core count to 23K cores (from 10K)
  - First time that parallel IO became imperative.
  - First time that cost of allreduce became very noticeable "at scale" runs.

PFLOTRAN parallel scalability: I/O

- Added F90 support for PETSc binary IO, added MPI-IO and parallel HDF5 backends.
- For large parallel jobs, added two-phase I/O scheme that aggregates I/O requests (later encapsulated in SCORPIO library).

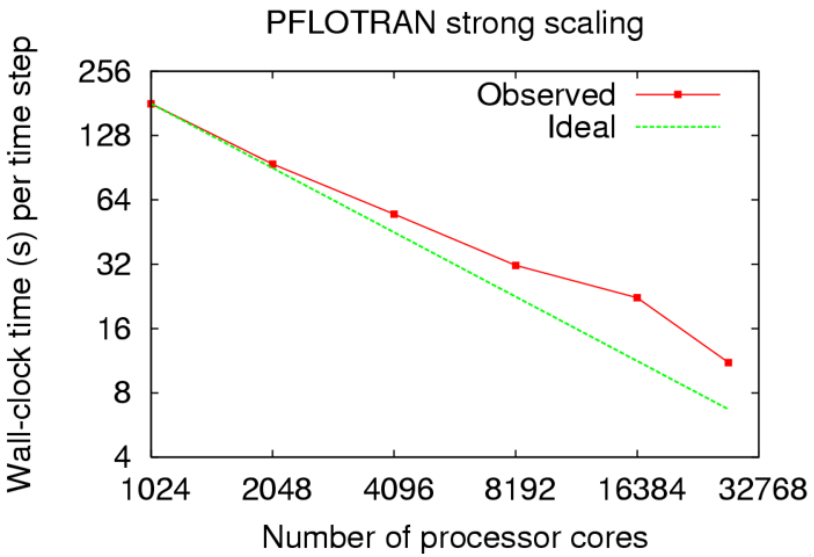






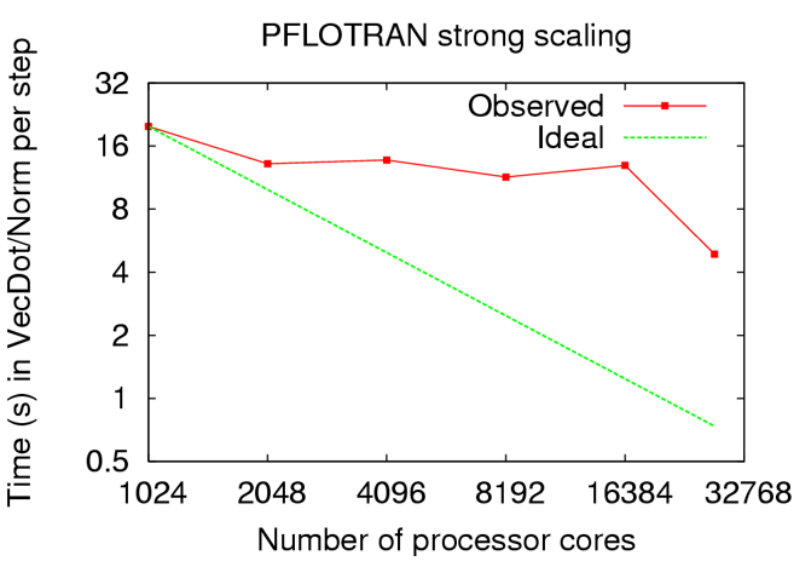
Cray XT series Krylov solver performance

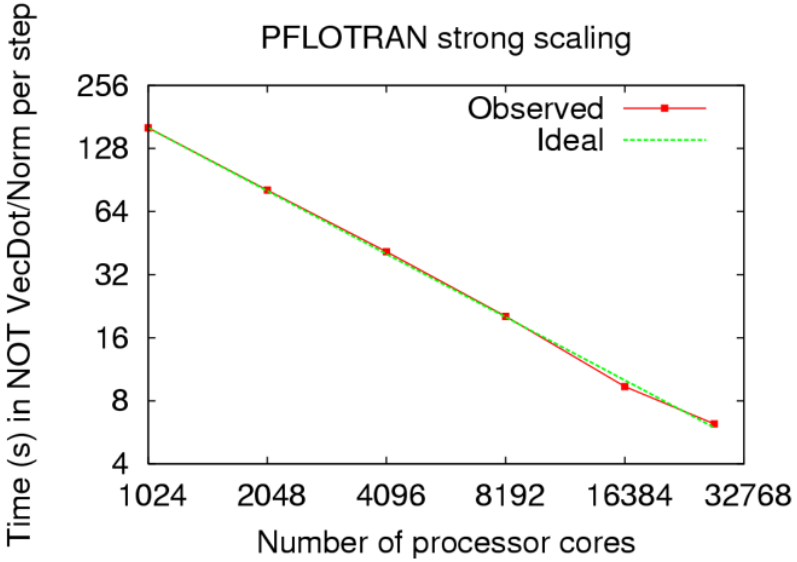
- Our workhorse solver for transient problems was inexact Newton (w/ line search if single-phase), BiCGStab inner solve, preconditioned w/ block-Jacobi (or additive Schwarz), ILU(0) on each subdomain.
- Why not multigrid? Difficulties with multiphase (particularly variable switching); difficult to do for multicomponent systems; block-Jacobi worked surprisingly well

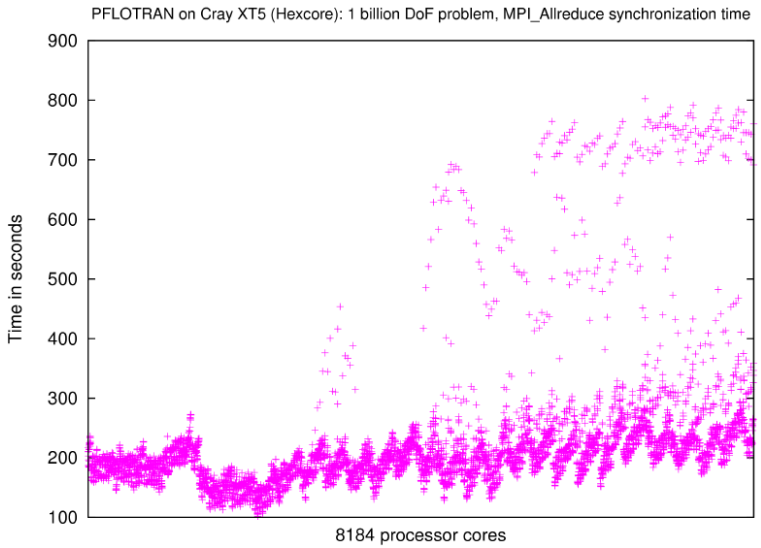


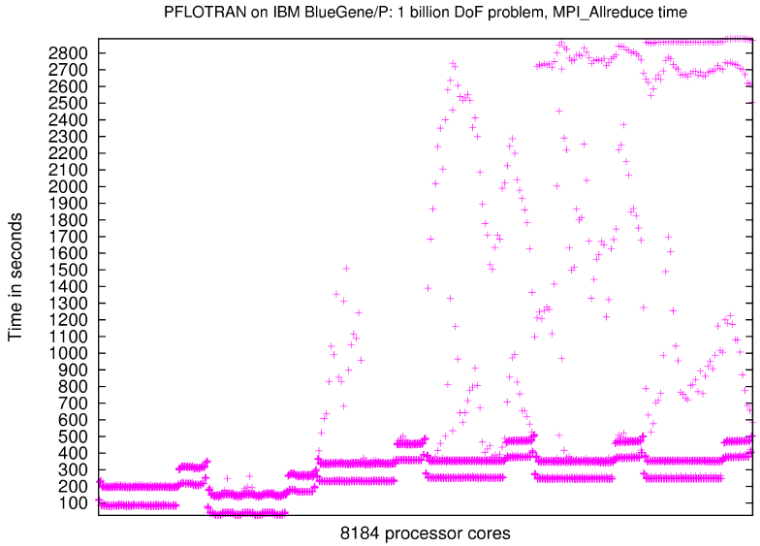
- Beyond $\sim$ 8000 cores, scalability became much worse.
- Not due to poorer performance of block-Jacobi as subdomains grew smaller.

“It’s the dot products, stupid!”





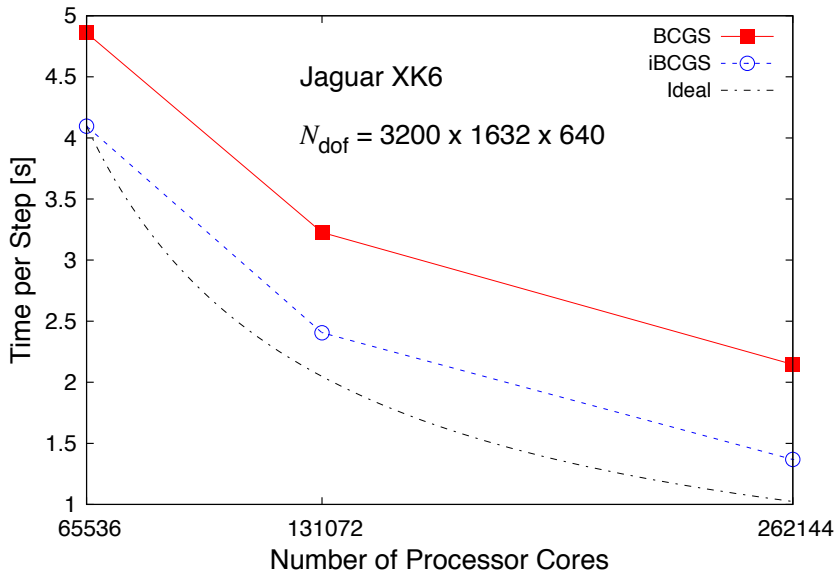




Cray XT: BiCGStab Improvements

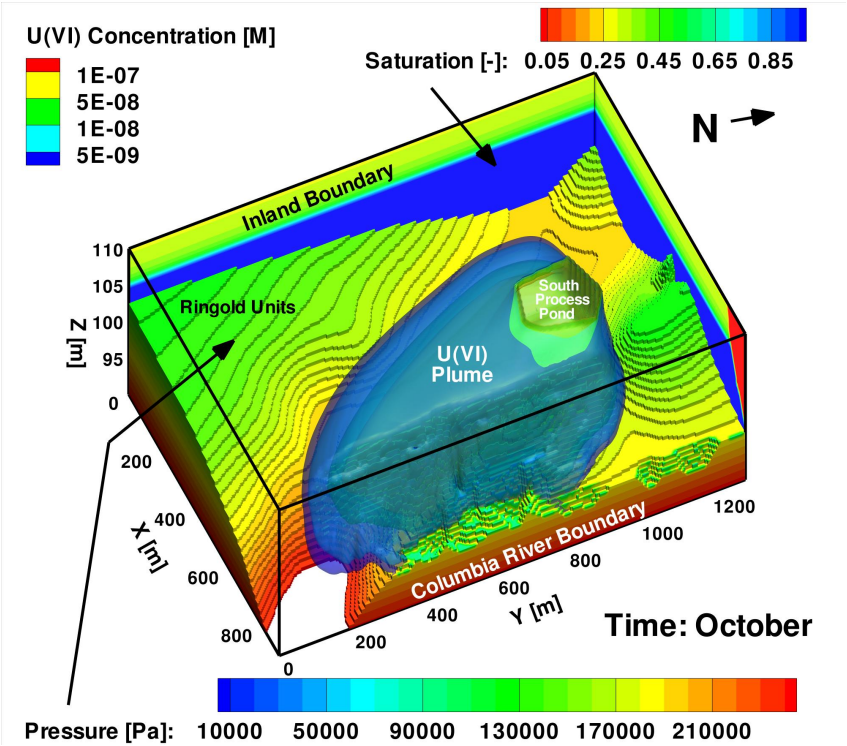
Cost of MPI_Allreduce() calls inside Krylov solver are big scalability barrier

- Reworked PETSc BiCGstab from 4 down to 3 allreduces/iteration (including convergence check)
- Also added Improved BiCGStab (IBCGS)
  - More complicated: requires transpose matrix-vector product, extra vector operations, ugly code
  - Only 2 MPI_Allreduce()s per iteration required; 1 if lagging residual norm calculation (at cost of an additional iteration)

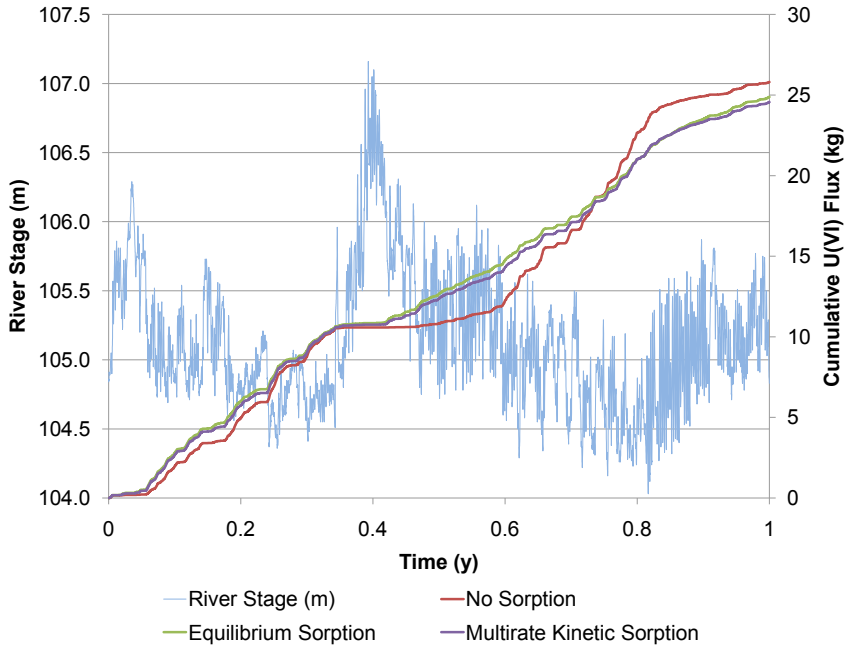


Hanford 300 Area U(VI) Plume Scale Modeling

Simulated U(VI) plume at Hanford 300 Area



Cumulative U(VI) flux to Columbia River



- 28M degrees of freedom
- Extreme transients
- 1-year simulation
- Hourly (8760) time steps

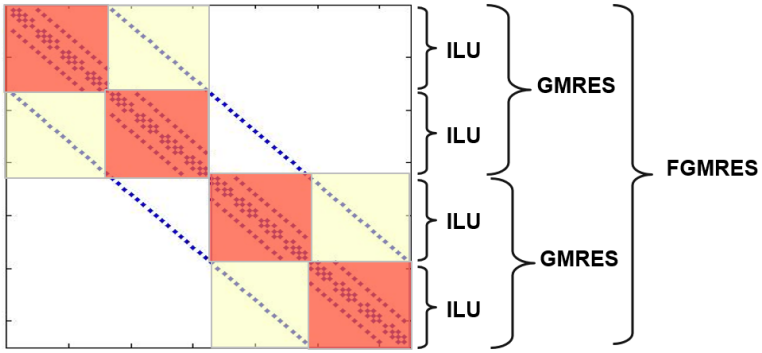
- 12 hours on 4096 Jaguar XT4 processor cores

Hammond and Lichter, WRR, 2010
DOE ASCR/BER SciDAC Groundwater

Hierarchical/Nested Krylov methods

- IBiCGStab improves runtime by $\sim 20\%$ for large jobs, but the Allreduce scalability problem is still there.
- (In defense of Allreduce: Orthogonalization is mathematically very powerful.)
- Allreduce performs quite well at smaller core counts ($\leq$ 10,000 or so).
- One idea: Use a hierarchical approach to trade (expensive) global reductions for lots of (cheaper) local ones.
  - Example: 100,000 core run, use block Jacobi with 100 blocks running FGMRES on 1000 cores, each of those solves runs FGMRES on 10 blocks of 100 cores
- Another idea: Use a nested approach with inner iterations that require no inner products (Chebyshev iteration here).

Hierarchical Krylov methods




```
mpirun -np np ./pflotran -pflotranin <pflotran_input>
  -flow_ksp_type fgmres -flow_ksp_pc_side right
  -flow_pc_type bjacobi -flow_pc_bjacobi_blocks ngp
  -flow_sub_ksp_type gmres -flow_sub_ksp_max_it 6
  -flow_sub_pc_type bjacobi -flow_sub_sub_pc_type ilu
```

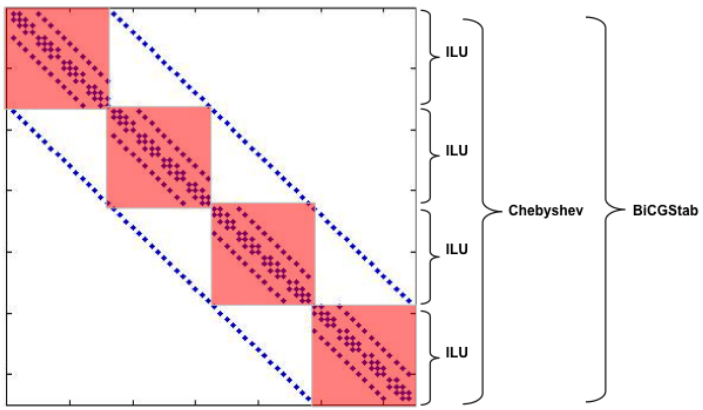
Num. of Cores (np) (mesh size)	Groups of Cores (ngp)	Num. of Cores per Group	Timestep Cuts	% Time for Inner Products	Outer Iterations	Execution Time (sec)
512	1	512	0	28	3,853	43.9
(256x256x256)	16	32	0	17	903	45.8
4,096	1	4,096	0	39	11,810	146.5
(512x512x512)	64	64	0	23	2,405	126.7
32,768	1	32,768	1	48	35,177	640.5
(1024x1024x1024)	128	256	0	28	5,244	297.4
98,304	1	98,304	7	77	59,250	1346.0
(1024x1024x1024)	128	768	0	47	6,965	166.2
160,000	1	160,000	9	72	59,988	1384.1
(1600x1600x640)	128	1,250	0	51	8,810	232.2

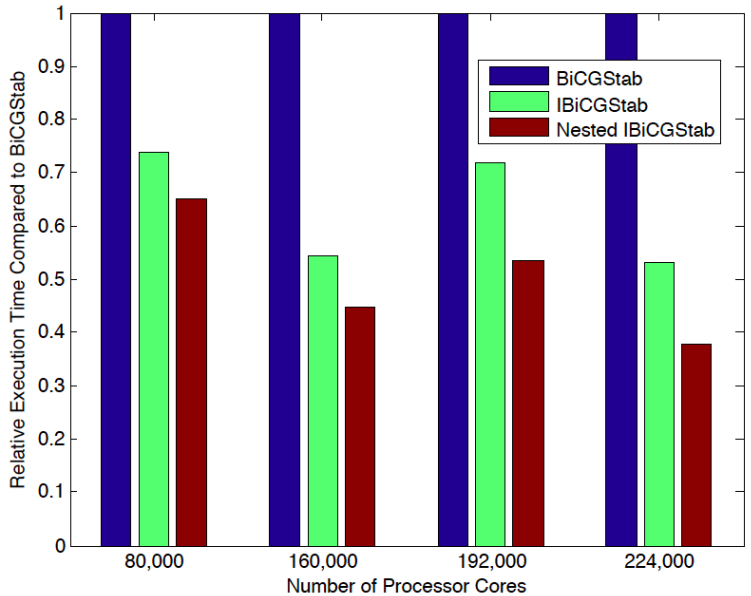
Work by L. C. McInnes, H. Zhang, B. Smith, and R. T. Mills. Results from Jaguary Cray XK6 at ORNL.

See `http://www.mcs.anl.gov/papers/P2097-0612.pdf`.

Nested Krylov methods

```
mpirun -np np ./pflotran -pflotranin <pflotran_input>
  -flow_ksp_type bcgs -flow_ksp_pc_side right -flow_pc_type ksp
  -flow_ksp_ksp_type chebyshev -flow_ksp_ksp_chebychev_estimate_eigenvalues 0.1,1.1
  -flow_ksp_ksp_max_it 2 -flow_ksp_ksp_norm_type none -flow_ksp_pc_type bjacobi
  -flow_ksp_sub_pc_type ilu
```





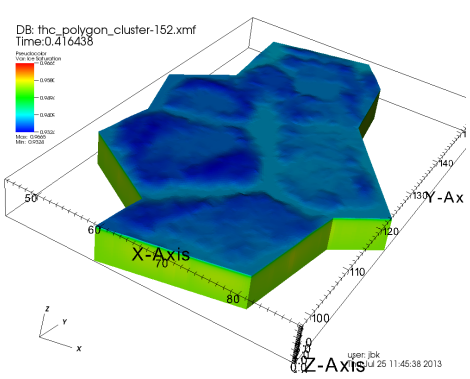
Work by L. C. McInnes, H. Zhang, B. Smith, and R. T. Mills. Results from Jaguary Cray XK6 at ORNL.

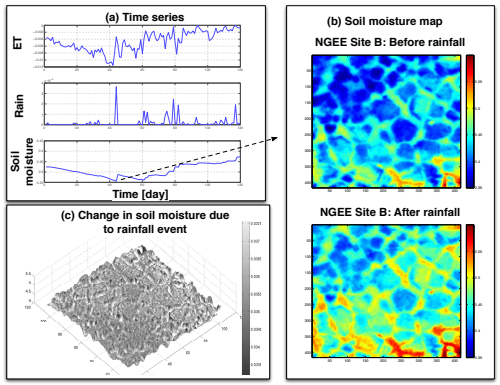
See `http://www.mcs.anl.gov/papers/P2097-0612.pdf`.

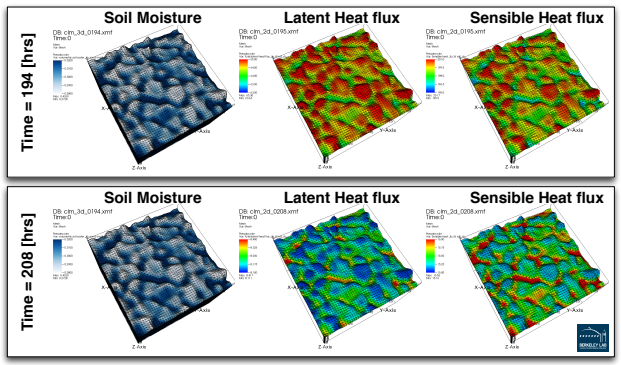
Post SciDAC-II: 2012—

PFLOTRAN development now part of various science projects, expanding into different directions

- Eco-hydro-climatology
  - Added surface water (first use of PETSc TS)
  - Coupling with Community Land Model (hydrology and biogeochemistry)
  - Freeze-thaw models for surface and subsurface

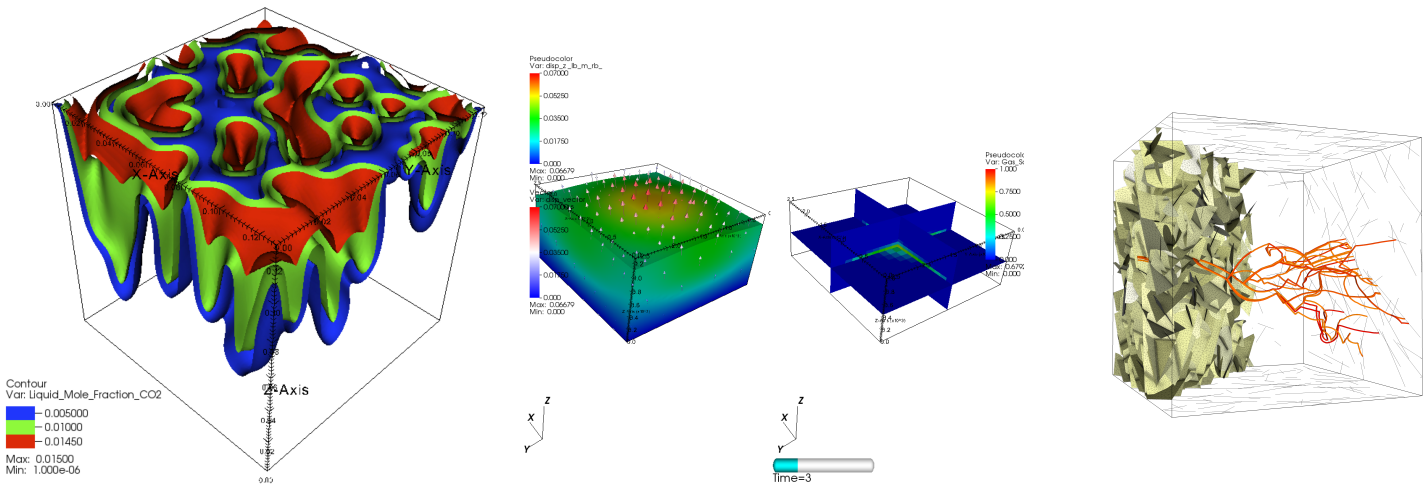




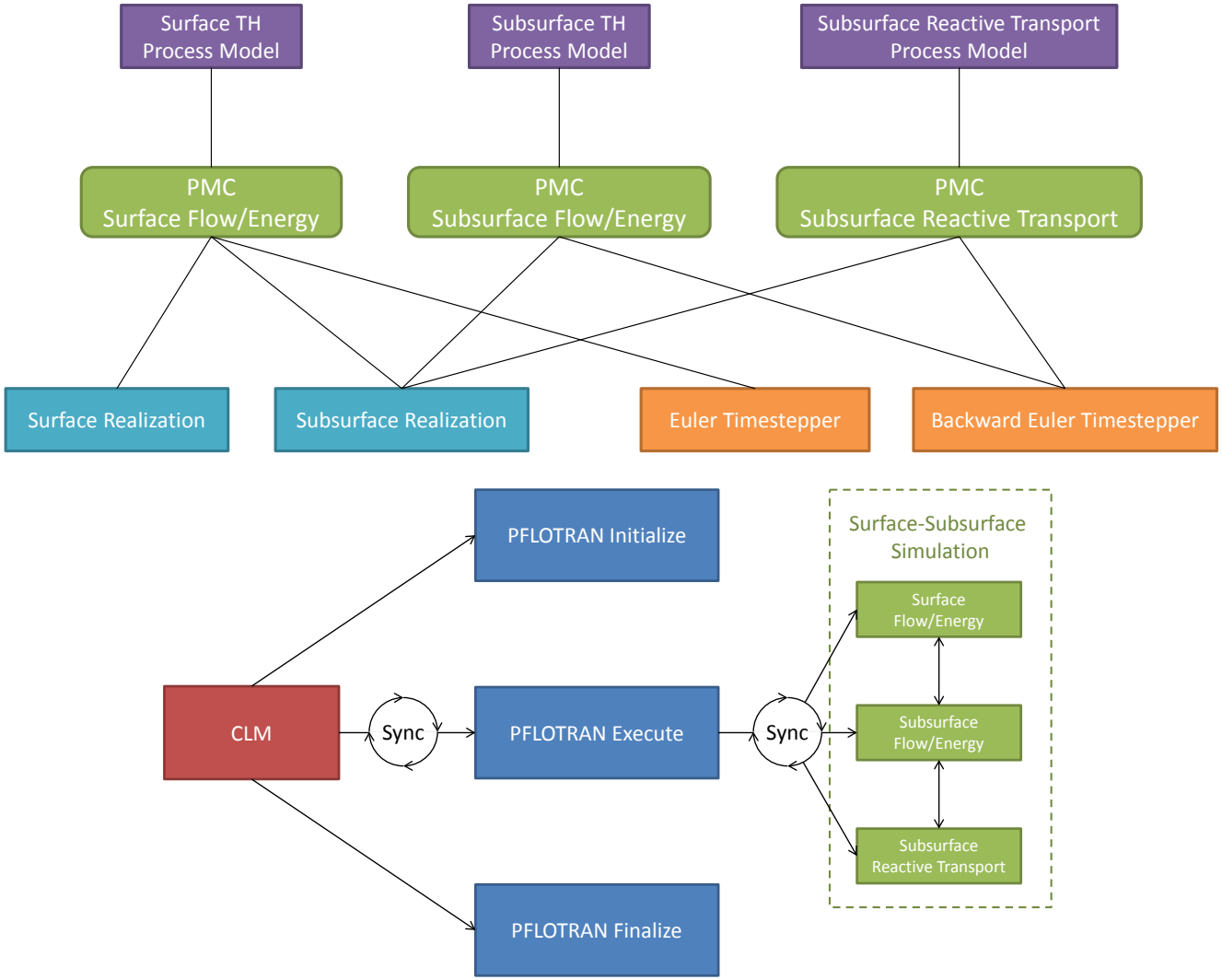


Post SciDAC-II: 2012—

- Spent nuclear fuel disposition
- Unconventional oil and gas extraction
- Induced seismicity due to injection
- CO2 sequestration and enhanced geothermal
  - Added geomechanical deformation (linear elasticity)
  - Further development of multiple interacting continua sub-grid model



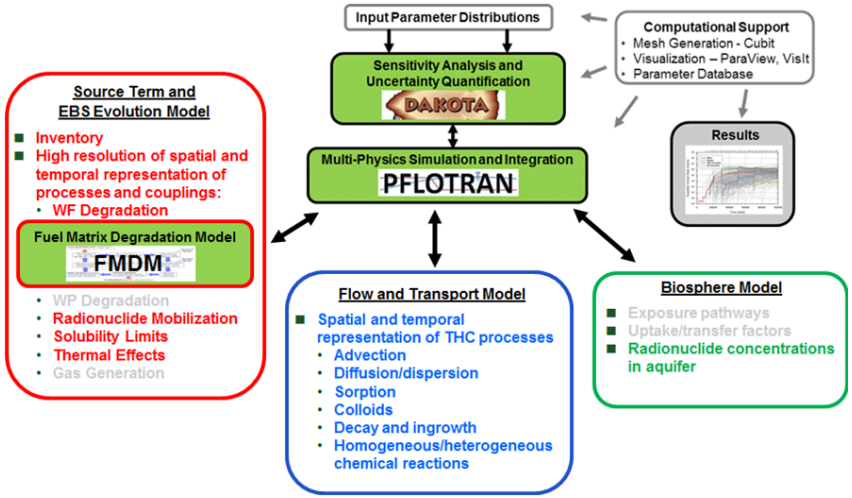
PFLOTRAN simulations are increasingly multi-process



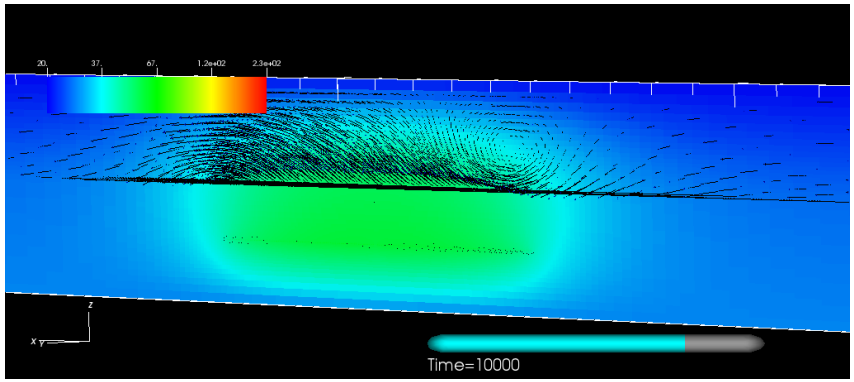
DOE Nuclear Energy Used Fuel Disposition Program

(former Yucca Mountain Program)

PFLOTRAN serves as the multi-physcis simulator within the Generic Disposal System Analysis framework

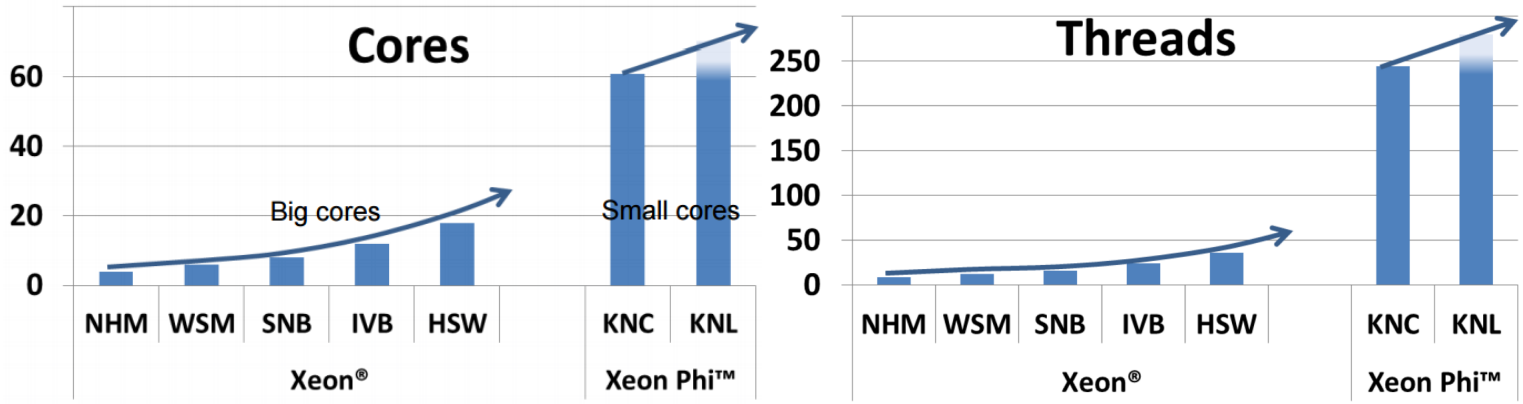


Thermally-induced convection cells at "hypothetical" repository



Hardware architectural trends

- More cores/threads on node and across machine



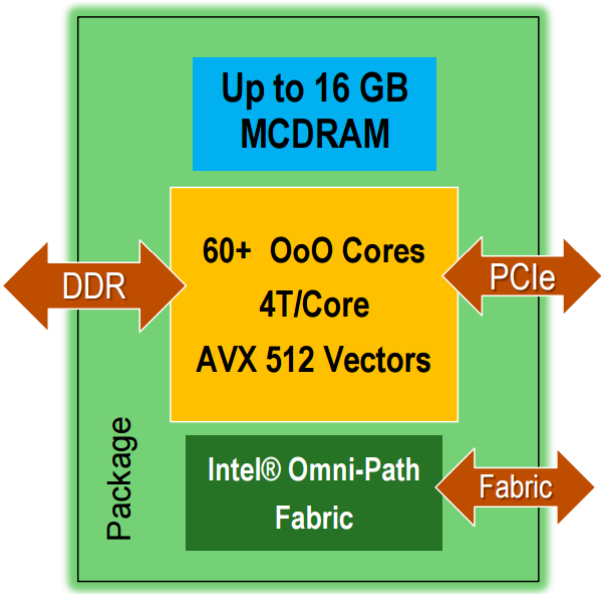
- More NUMA domains/level of storage hierarchy (DRAM NUMA domains, MCDRAM, NVRAM, IO subsystem)
- More data parallelism/fine-grained parallelism
  - AVX512 (512 bit vectors w/ FMA) coming in both Xeon Phi and Xeon
  - GPGPU trends

Upcoming DOE "leadership-class" supercomputers

For open science:

- Cori (NERSC)
  - $\sim$ 1400 dual socket nodes w/ Intel® Xeon® v3 ("Haswell") Processors, 16 cores per socket
  - Over 9,300 single socket nodes w/ 2nd gen Intel® Xeon Phi™ Processors ("Knights Landing"—KNL), w/ up to 16GB on-package, high-bandwidth memory
  - Cray Aries dragonfly topology interconnect
- Aurora (ALCF)
  - Over 50,000 nodes with 3rd gen Intel® Xeon Phi™ Processors
  - Over 8 PB aggregate on-package high-bandwith memory and persistent memory
  - 2nd gen Intel® Omni-Path Architecture with silicon photonics
- Summit (OLCF)
  - $\sim$ 3,400 compute nodes w/ multiple IBM POWER 9s and NVIDIA Volta GPUs per node
  - $>$ 512GB combined high-bandwidth memory and DDR4
  - 800 GB NVRAM for burst buffer or extended memory
  - Dual-rail Mellanox EDR InfiniBand in non-blocking fat-tree

Knights Landing Overview



- Available as standalone, self-boot CPU—no offload bottleneck; binary compatible with Intel® Xeon® Processors[1]
- 60+ Silvermont-based CPUs, 4 threads per core, out-of-order execution
- AVX512 vector units
- 2D mesh on-die interconnect
- MCDRAM on-package memory: 400+ GB/s bandwidth[2]
- Intel® Omni-path Fabric

Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. [1] Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). [2] Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory.
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to `http://www.intel.com/performance`.

Possible programming paradigms

- Hybrid MPI/OpenMP (or pthreads, etc.)?
  - To ensure data locality and avoid fork/join overhead, would want to adopt SPMD-style programming. Possible, but nontrivial in OpenMP.
- Offload to "accelerators"
  - PFLOTRAN team looked at this in 2010 with Cray and NVIDIA after ORNL Titan announcement.
  - Biggest GPU-target kernel was about 18%.
  - Far too intrusive: potential benefit not large enough to justify major rewrite.
- Stick with MPI. (But use MPI-3 functionality.)
  - Keep de-facto SPMD-style parallelization (*everything private by default*), but with constructs to leverage shared memory where beneficial.
  - No major application code rewrite needed; no complications from mixing two programming models that don't know about each other.

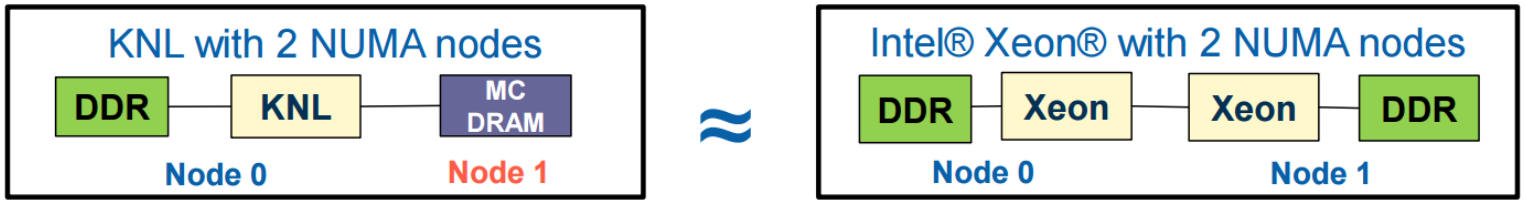
Leveraging MPI-3 in PETSc

- Use neighborhood collectives.
  - Add neighborhood collectives implementation for PetscSF.
  - Add VecScatter implementation on top of PetscSF.
- Support MPI_Win_allocate_shared() in a DM.
  - Extend concept of a "local" vector from PETSC_COMM_SELF to a shared-memory communicator from MPI_Comm_split_type()?
  - Or, rather, abstract to several levels of "local": Private to a rank, shared within a NUMA domain, shared within a node, ...
  - If you want to avoid duplicating halos inside shared memory regions, this breaks nice indexing of ghost points?

Approaches for using MCDRAM

Cache mode: Let the hardware handle it.

Flat mode: MCDRAM exposed as a separate NUMA node



- Can do low-level management with libnuma and mmap().
- Can use memkind (`https://github.com/memkind`) for partitioned management of heap. E.g.,

```
hugetlb_str = (char *)memkind_malloc(MEMKIND_HUGETLB, size);

hbw_preferred_str = (char *)memkind_malloc(MEMKIND_HBW_PREFERRED, size);
```


Supporting multiple kinds of memory in PETSc

- Simple option: Use size thresholds for automatic greedy allocation of high-bandwidth memory.
  - Largest data structures tend to be most bandwidth-intensive; small ones can fit in cache
- More complicated: Add PetscAdvMalloc() that accepts an advisory context, provide way to tag objects (Vec, Mat) to be used w/ associated malloc()s.

Problem: Making the right placement decisions based on a priori reasoning may be impossible.

- Do placement/migration based on measured importance. Barry suggests counting VecGetArray[Read]()s.
- In many cases, decisions need to account for what is going on external to PETSc as well. Optimal approach may need OS and/or middleware support.

Possible algorithmic/numerical/science approaches

- Focus more on ensembles than "hero" runs (e.g., uncertainty quantification, ensemble Kalman filters)
- Hierarchical Krylov with "physics-based" treatment of subdomains
- Algebraic multigrid with PCGAMG (possibly inside flexible BiCGStab)
- FAS/nonlinear multigrid
  - Promises much greater efficiency through reduced memory motion
  - (We were waiting for Matt to rewrite Sieve/DMPlex 4–5 times first.)
- Structured AMR (again, but with p4est this time)
  - See Toby Isaac's talk tomorrow morning.
  - Intel Parallel Computing Center (Adams, Knepley, Brown)

Conclusion

- We owe huge thanks to Barry, Matt, Satish, Lois, Hong, Jed, and many other developers of PETSc!
- PFLOTRAN development would not be possible without PETSc:
  - We'd waste too much effort managing meshes, distributed data structures, writing solvers, even building third party libraries.
  - Our ability to experiment and to adapt to particular problems and machines would be greatly diminished without PETSc's runtime configurability.
- And PFLOTRAN development would not be fun!
  - Thanks for all of the mailing list ~~polemics~~ banter over the years: Lots of entertainment and good ideas!
- We're looking forward to the next 20 years!

