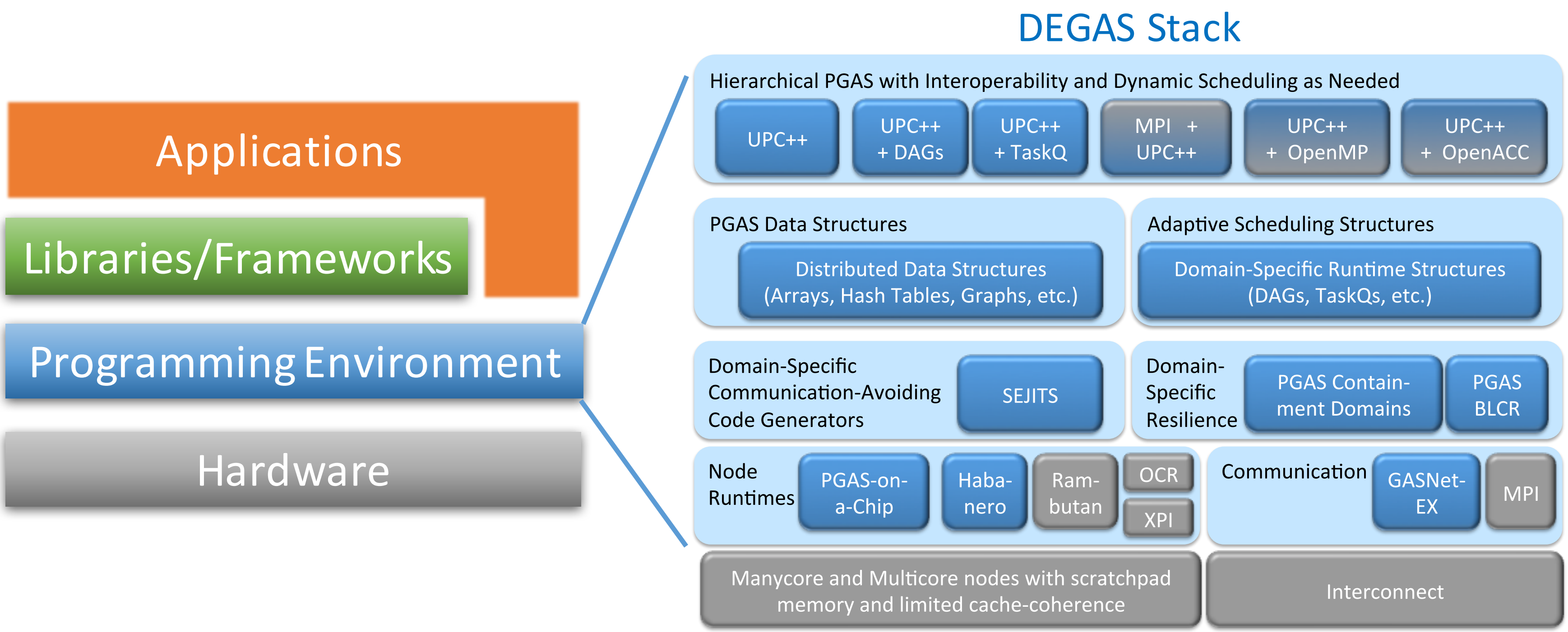
Dynamic Exascale Global Address Space

Presenter: Yili Zheng

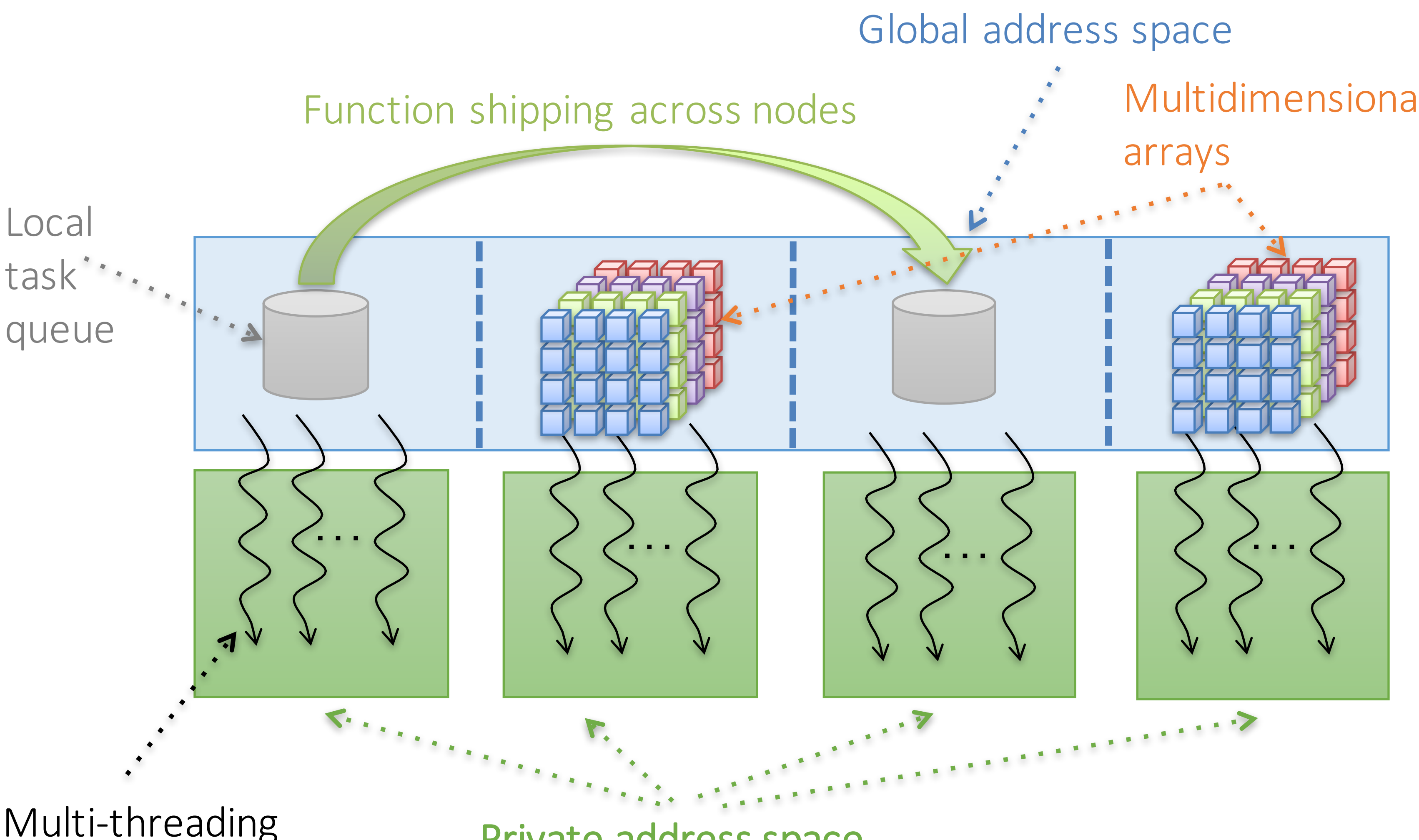
Katherine Yelick, LBNL PI
Vivek Sarkar & John Mellor-Crummey, Rice
James Demmel, UC Berkeley
Mattan Erez, UT Austin
Frank Mueller, NCSU
John Bachan, Scott Baden, Protonu Basu, Dan Bonachea, Paul Hargrove, Steven Hofmeyr, Costin Iancu, Khaled Ibrahim, Amir Kamil, Leonid Oliker, Eric Roman, John Shalf, Hongzhang Shan, Harsha Simhadri, Brian Van Straalen, Erich Strohmaier, Samuel Williams, Weiqun Zhang, Yili Zheng, *LBNL*
Laksono Adhianto, Vivek Kumar, Scott Warren, *Rice*
Plus a team of brilliant students

DEGAS Mission for XStack

Build a PGAS programming environment and toolkit that deliver high performance and productivity to DOE applications on current and future systems.



PGAS with Composable Extensions



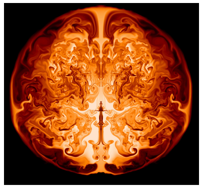
Data Structures and Runtime Support for Irregular Data-Intensive Applications

Speedups

- Distributed hash table
  - Applications: HipMer (genomics)

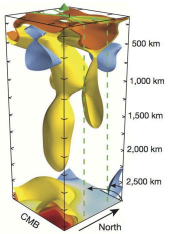
720x

- Irregular data exchange
  - Applications: AMR, HPGMG



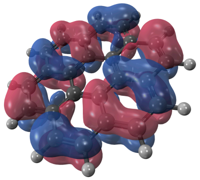
1.2x

- Irregular global matrix update
  - Applications: NWChem, seismic tomography



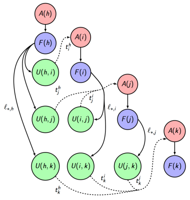
6x

- Distributed work queue
  - Applications: NWChem, Hartree-Fock



1.2x

- Dynamic task graph
  - Applications: Sparse symmetric matrix solver

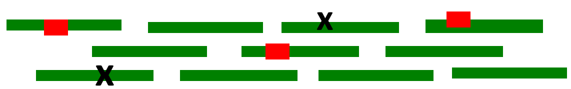


2x

HipMer: High-Performance Meraculous

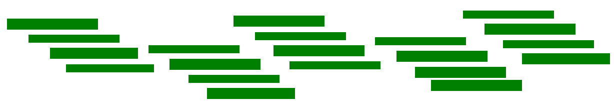
Meraculous Assembly Pipeline

reads



New fast & parallel I/O

k-mers

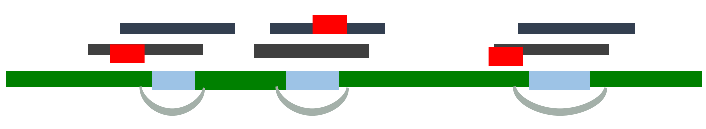


New k-mer analysis filters errors using probabilistic "Bloom Filter"

contigs

Graph algorithm (connected components) scales to 15K cores on NERSC's Edison

Scaffolding using Scalable Alignment



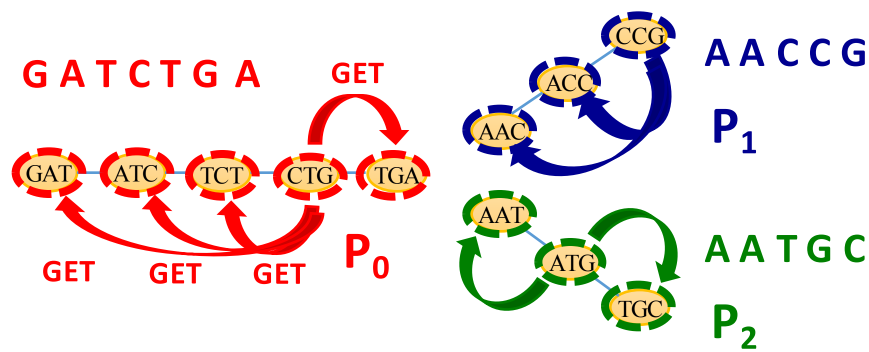
Typical Genome Data Size

Human  650 GB

Wheat  1200 GB

Salamander  1400 GB

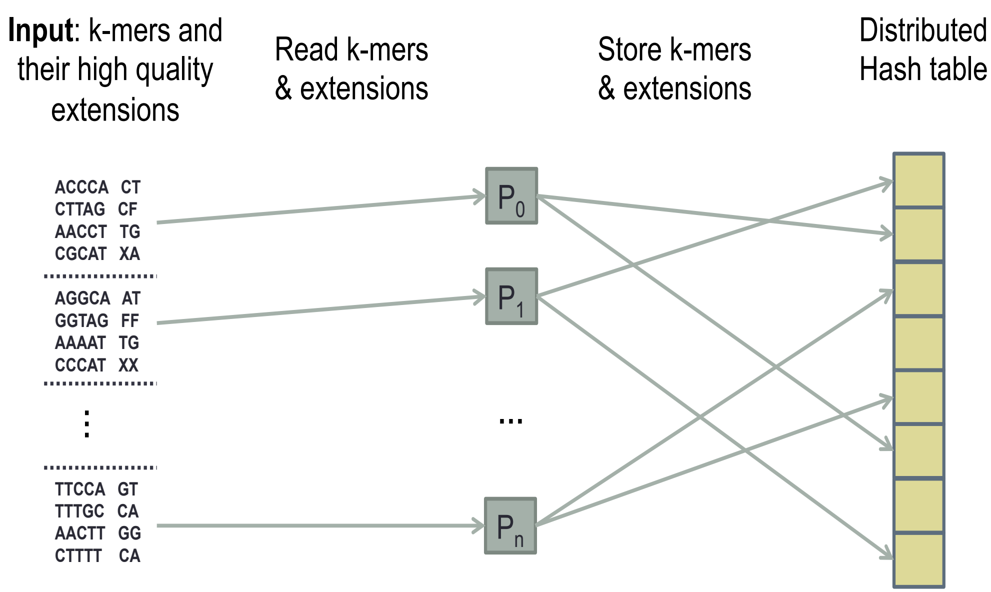
Represent genome by de Bruijn Graph

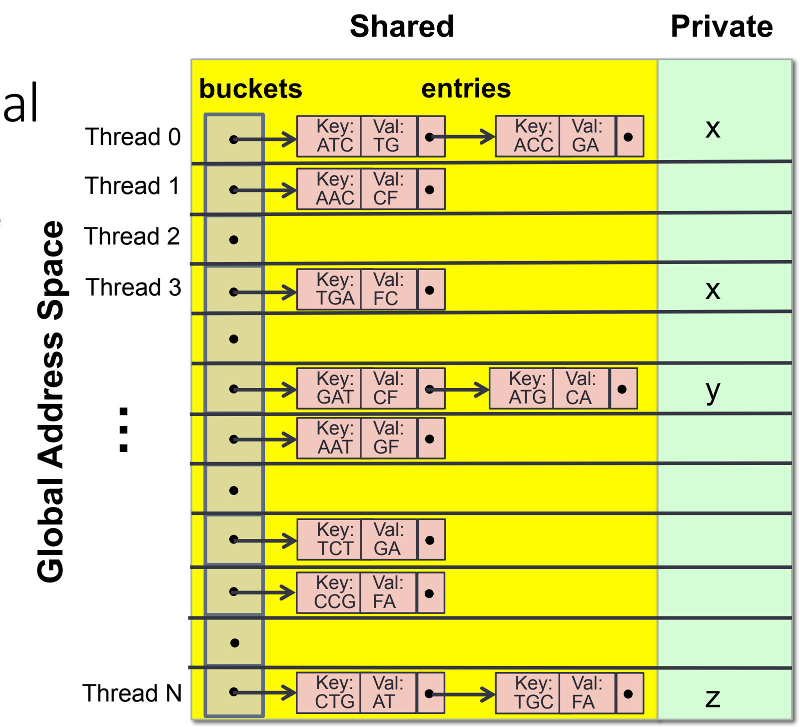


Georganas, Buluc, Chapman, Oliker, Rokhsar, Yelick, [Aluru,Egan,Hofmeyr] in SC14, IPDPS15, SC15

PGAS Hash Table for Efficient Graph Construction and Traversal

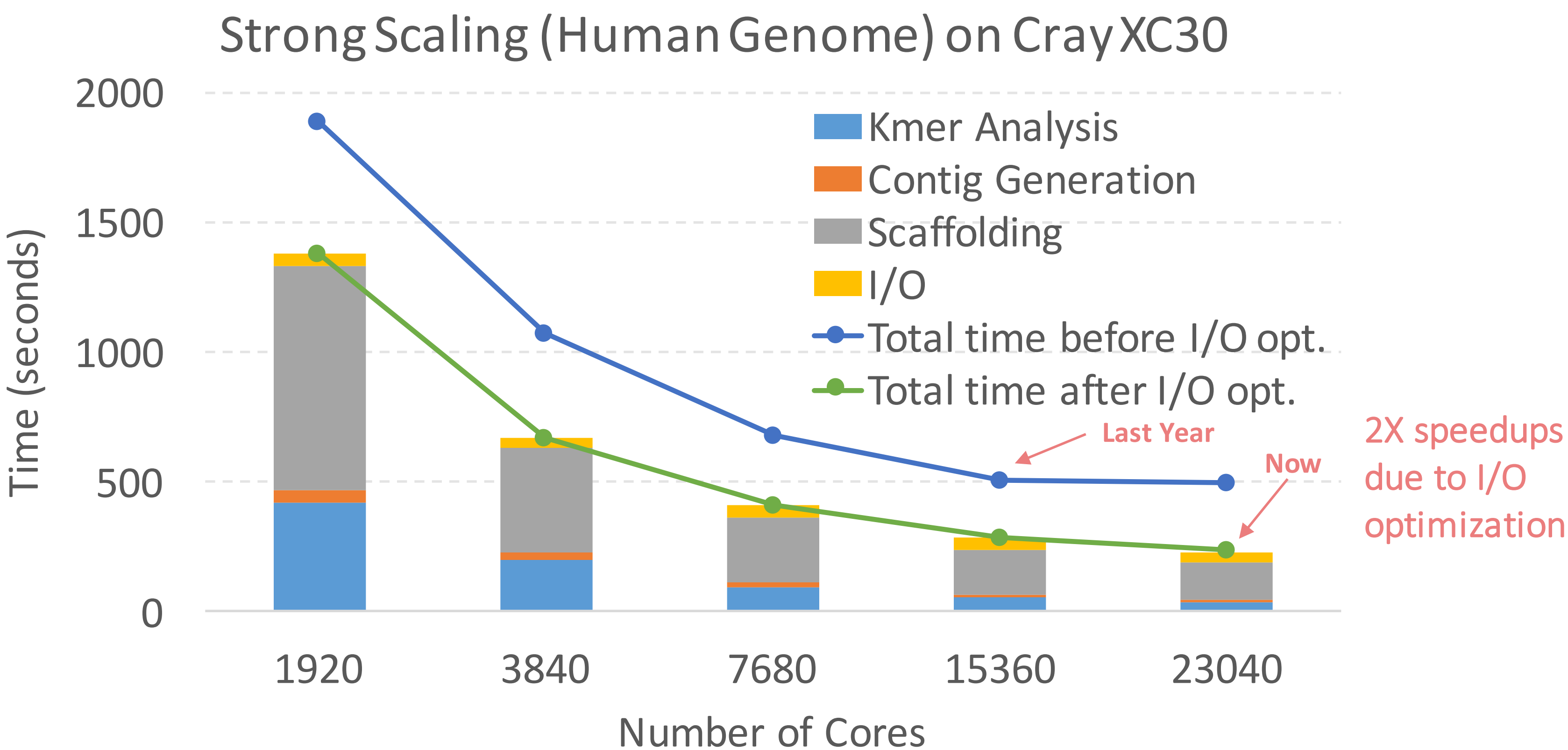
- Implement the de Bruijn graph by a distributed hash table written in UPC
- Parallel graph construction and traversal





The distributed hash table data structure can be applied to similar type of problems.

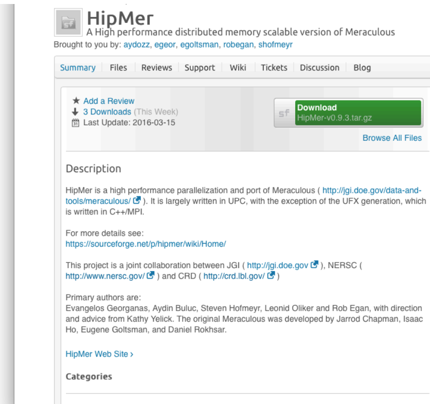
New HipMer Results

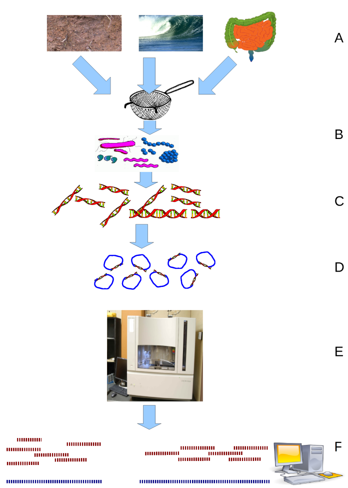


- Complete assembly of human genome in **4 minutes using 23K cores**
- **720x faster than the original Meraculous** due to the combination of algorithmic innovations, massive parallelization, and optimized C code

Recent Progress on HipMer

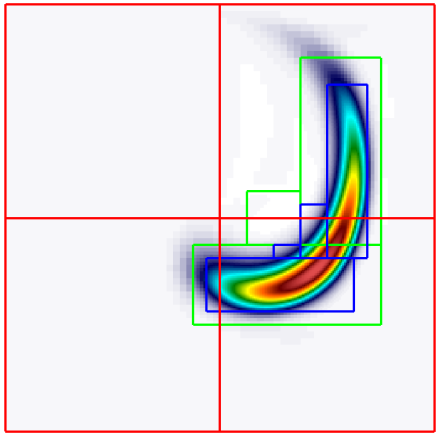
- Tackling the metagenome assembly grand challenge
- HipMer released!
  https://sourceforge.net/projects/hipmer
- Collaborating with NERSC on creating a web portal

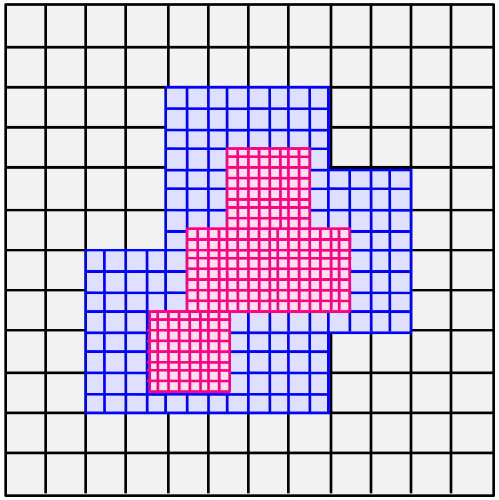




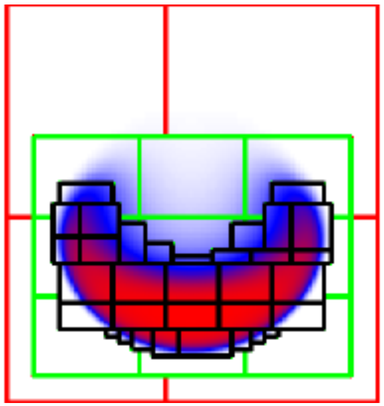
Adaptive Mesh Refinement

- AMR allows the method to dynamically adapt the multilevel grid hierarchy on which the equations are solved.
- Finer level composed of union of regular subgrids but the union may be irregular
- Intensive and dynamic data exchange communication required
  - Between levels
  - Neighbors within the same level

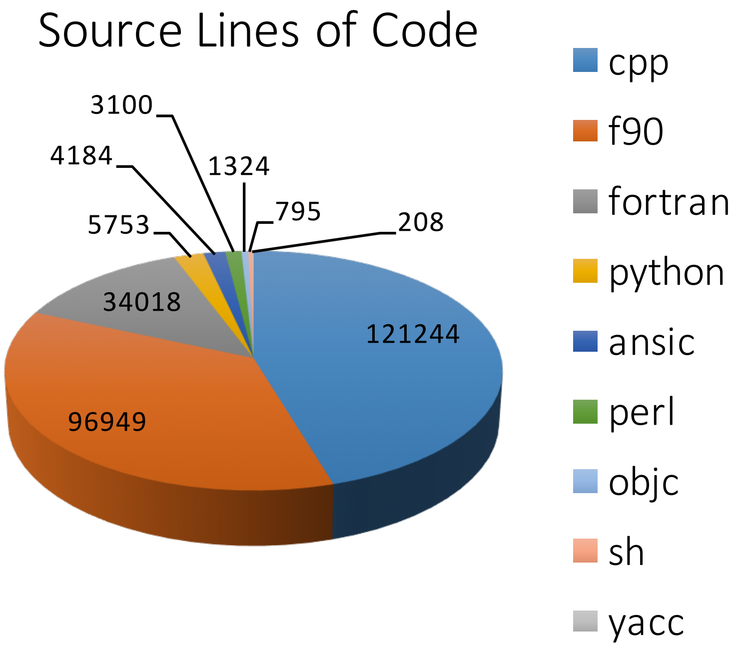




BoxLib AMR Framework

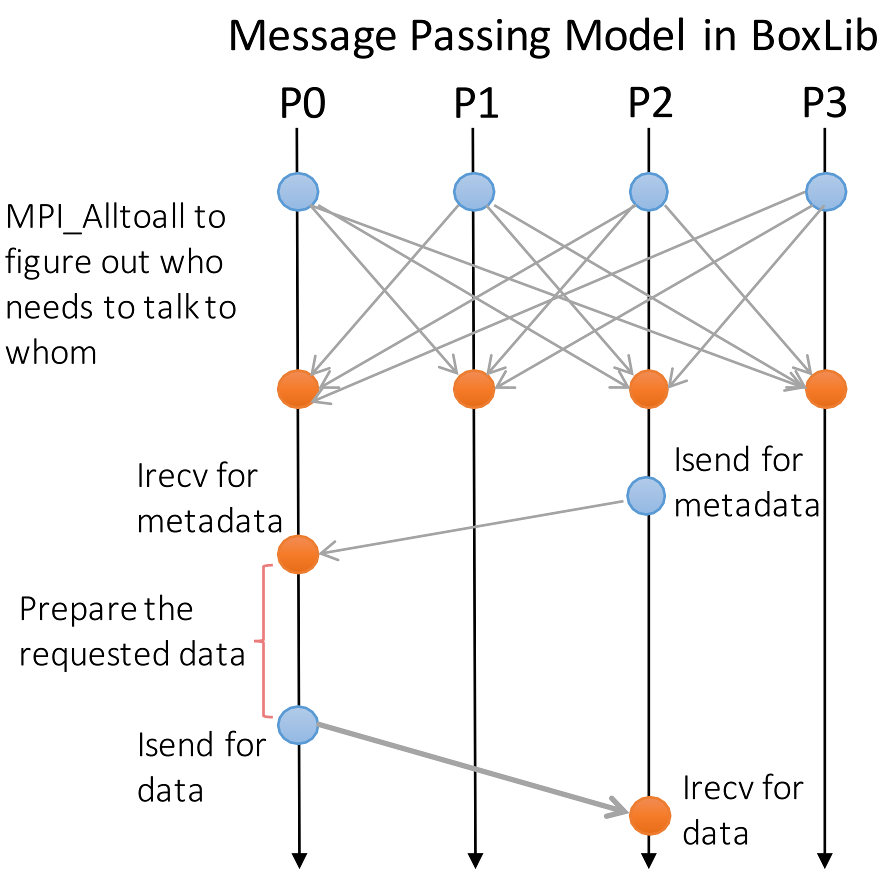


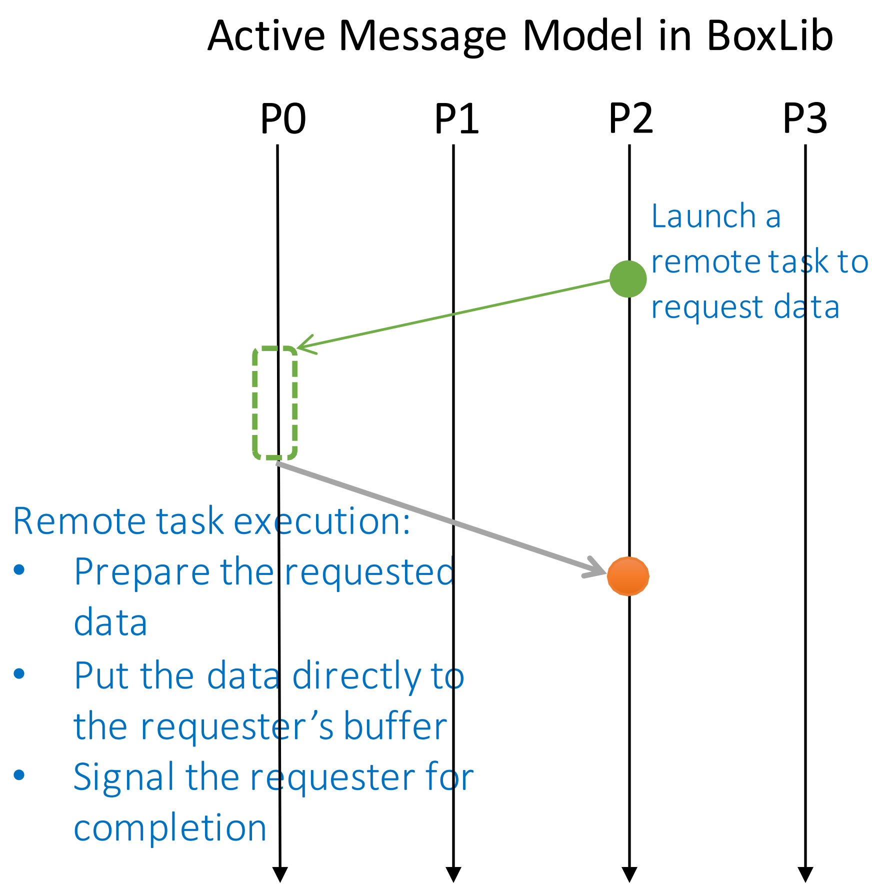
- BoxLib mostly written in C++ and Fortran 90 with MPI+OpenMP
  - BoxLib development effort estimated by SLOCCount: 70.77 Person-Years ($24.77M)
- Need an incremental adoption strategy with maximum code reuse
- Collaboration and integration are key!



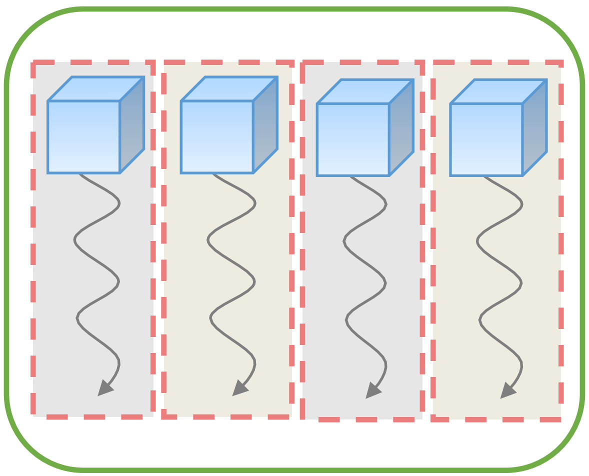
Active Messages Simplify Communication Workflow

In the following example, P2 needs data from P0.

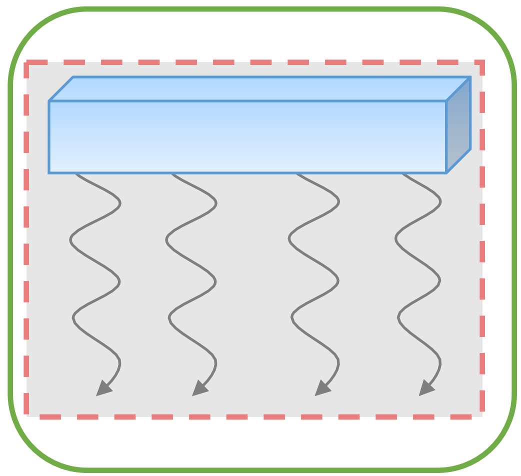




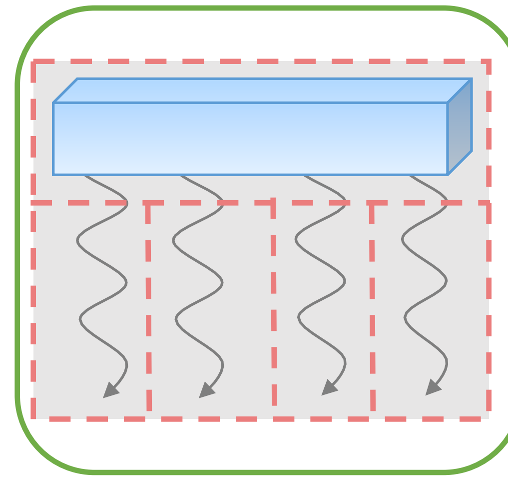
PGAS for Efficient Communication and Data Sharing within a Node



Pure Message Passing Model: no data sharing



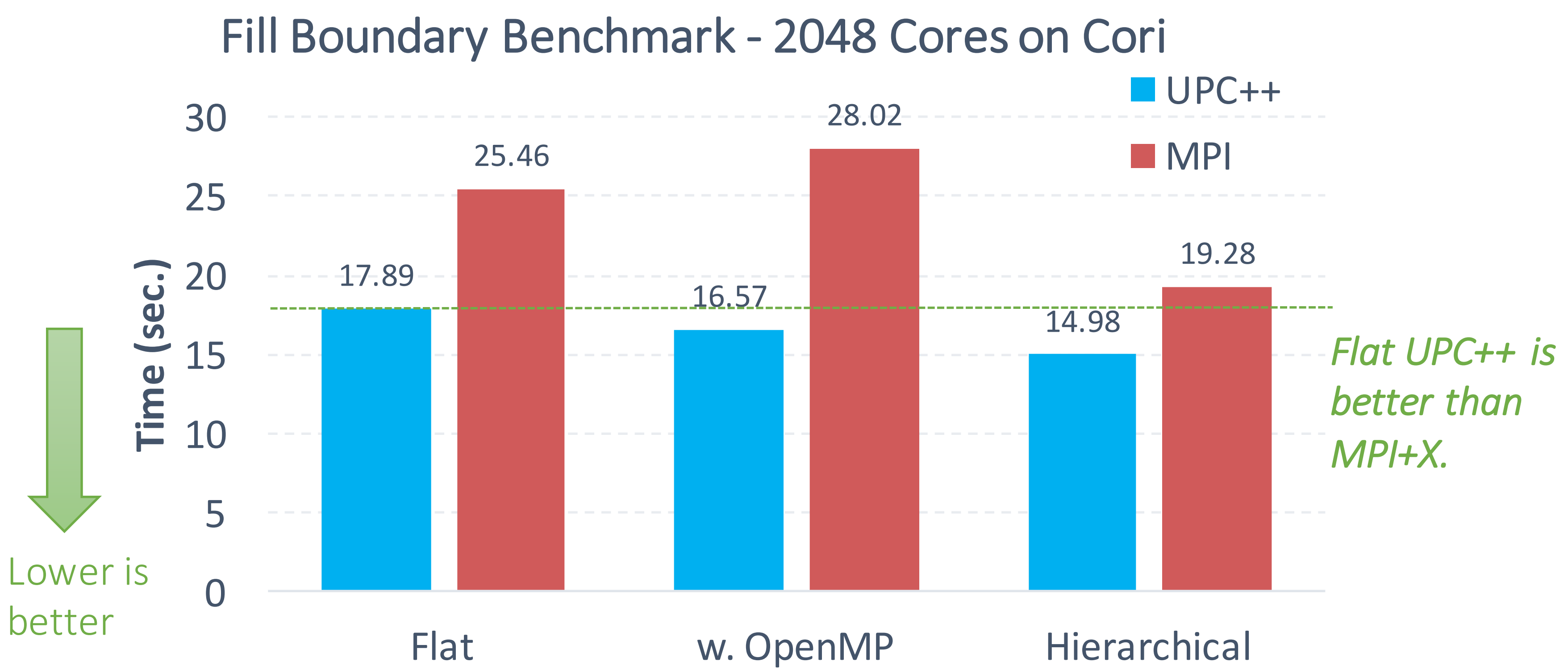
Pure Multi-threading Model: share all data



PGAS Model: selectively share data

- Pure message passing model is good for data protection and parallel network injection.
- Pure multi-threading model is good for sharing data and intra-node communication.
- PGAS (process-shared-memory) provides both advantages.

Communication Performance Improvement in BoxLib



Flat: use only one programming model.
Hierarchical: use one programing model but handle on-node communication through shared-memory (e.g., MPI+MPI)

Full Application Performance: Compressible Astrophysics (CASTRO)

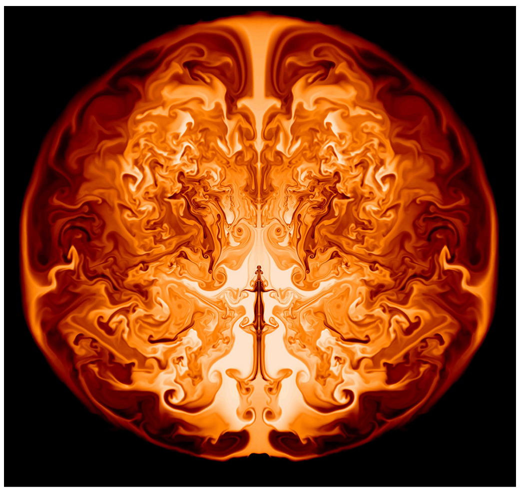
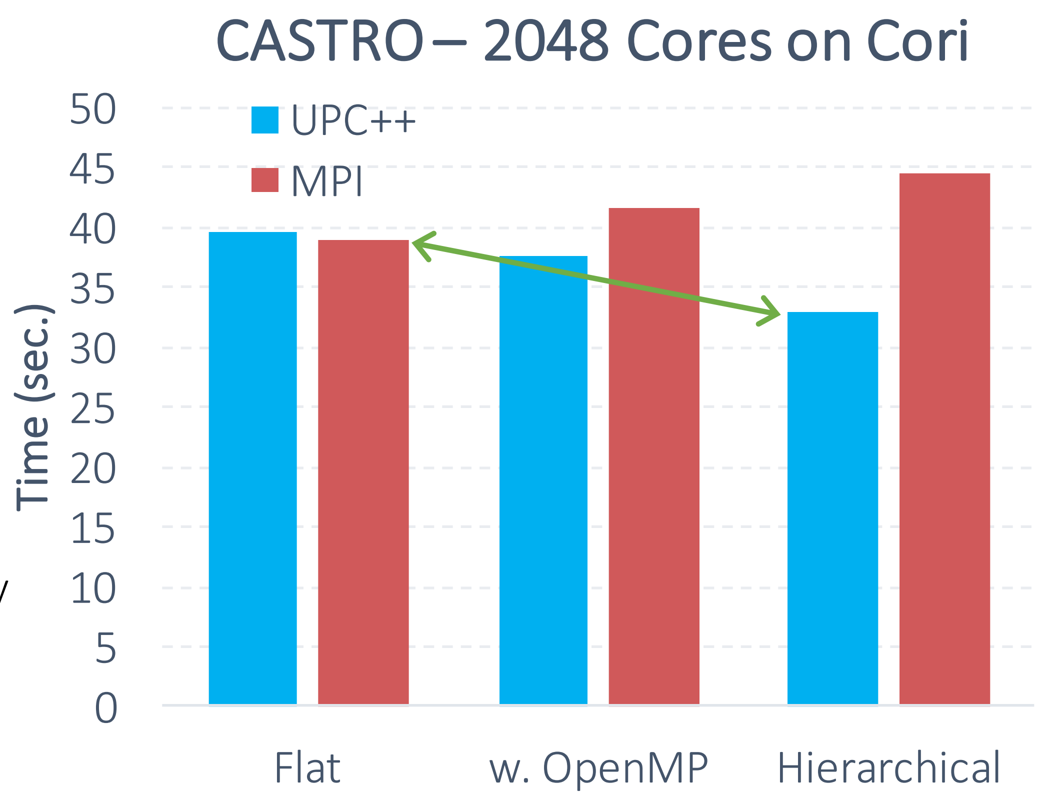


Image Credit: Ken Chen, University of California, Santa Cruz

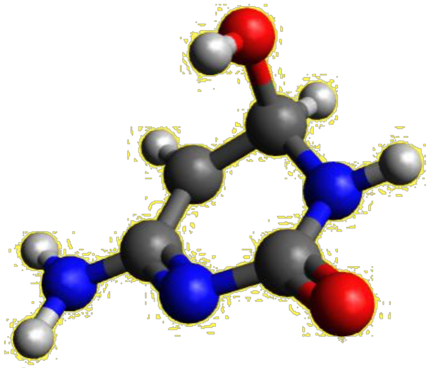


The best UPC++ version (Hierarchical) is 18% faster than the best MPI version (flat).

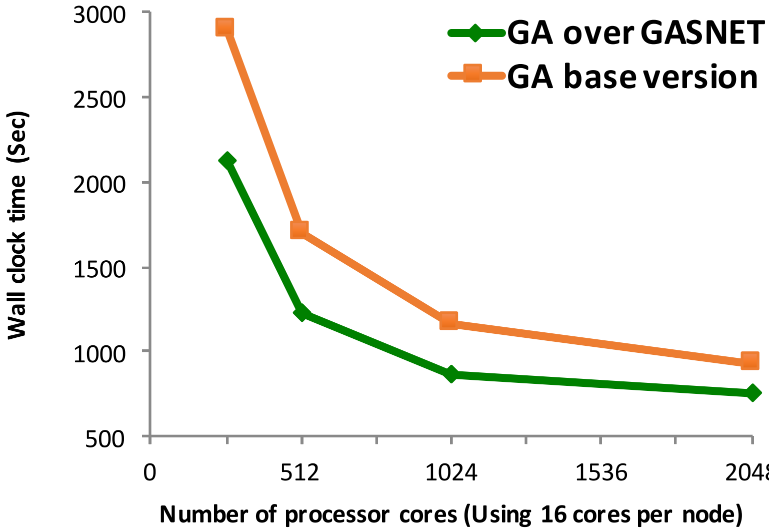
Flat: use only one programming model.
Hierarchical: use one programing model but handle on-node communication through shared-memory.

Global Arrays Over GASNet in NWChem

- Problem
  - Enabling UPC++ capabilities in NWChem
  - Transformation needs support Global Arrays Toolkit and UPC++ to limit disruption to large user base
- Solution
  - New Global Arrays Toolkit over GASNet
  - Transform current or add new capabilities with UPC++
- Impact
  - *Over 20% faster on Infiniband than the base Global Arrays over ARMCI solution in NWChem for coupled cluster simulation*



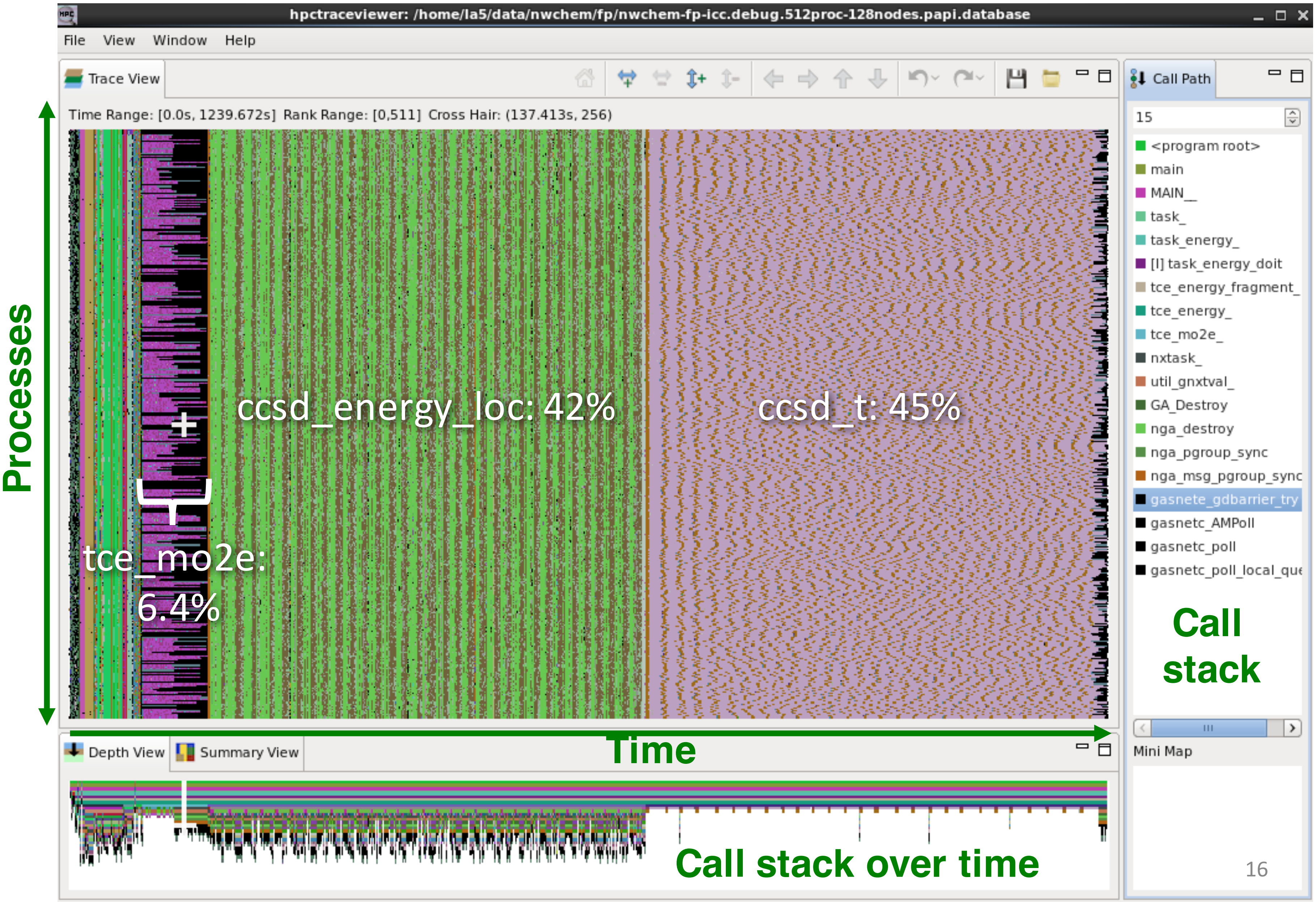
Cytosine-OH
TCE: CCSD(T)
268 basis functions
Correlating 49 electrons



Strong Scaling of GASNet Compared to ARMCI in NWChem on PNNL's Cascade with Infiniband Network

Bert de Jong

NWChem Execution Overview (Cytosine OH)



NWChem Analysis (Cytosine OH)

tce_mo2e

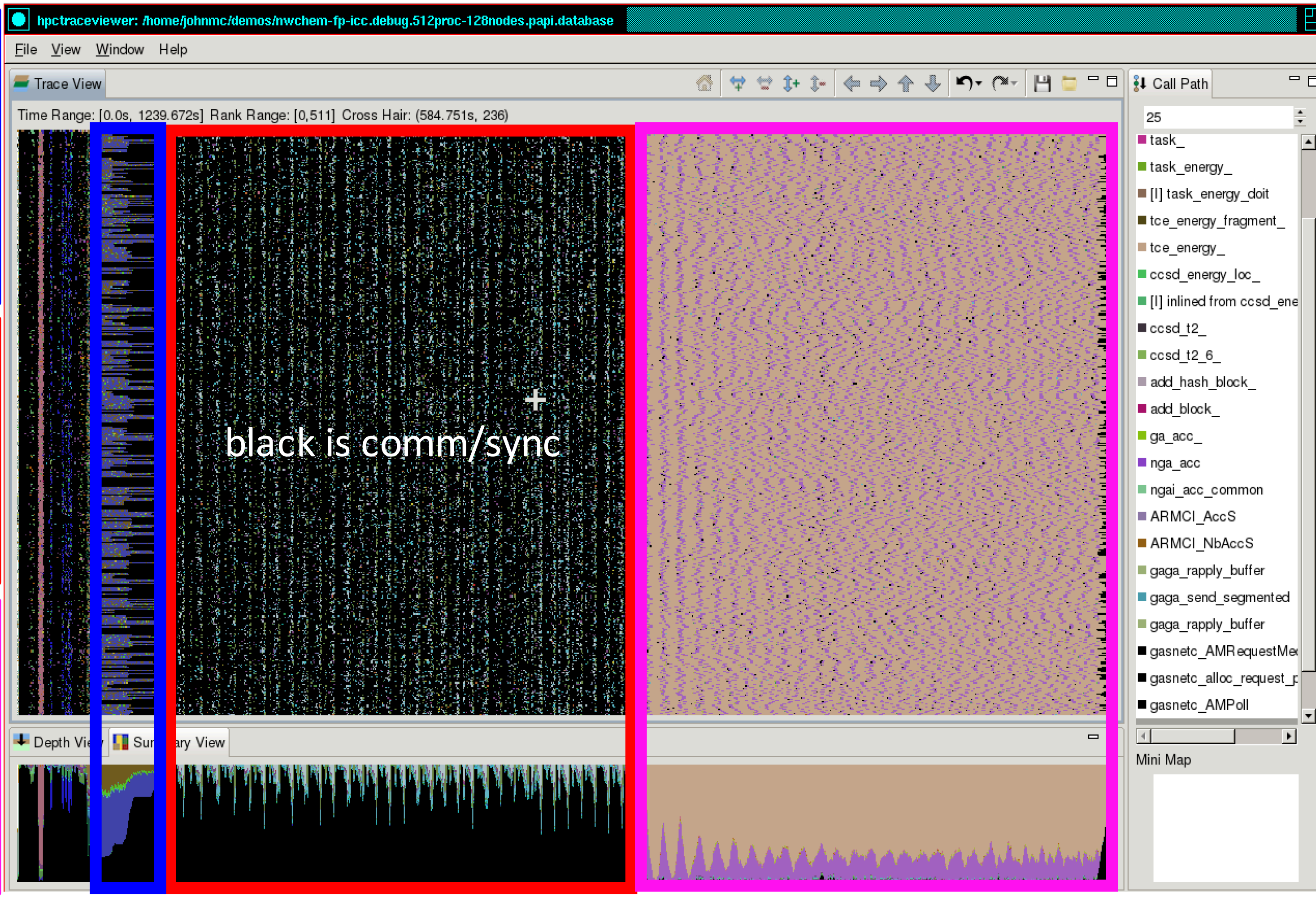
- 6.4% of execution
- 58% is barrier on behalf of nxtask
- <u>imbalance</u>: some ranks have no work

ccsd_energy_loc

- 42% of execution
- 43% is barriers on behalf of nxtask
- 38% is comm (get)

ccsd_t

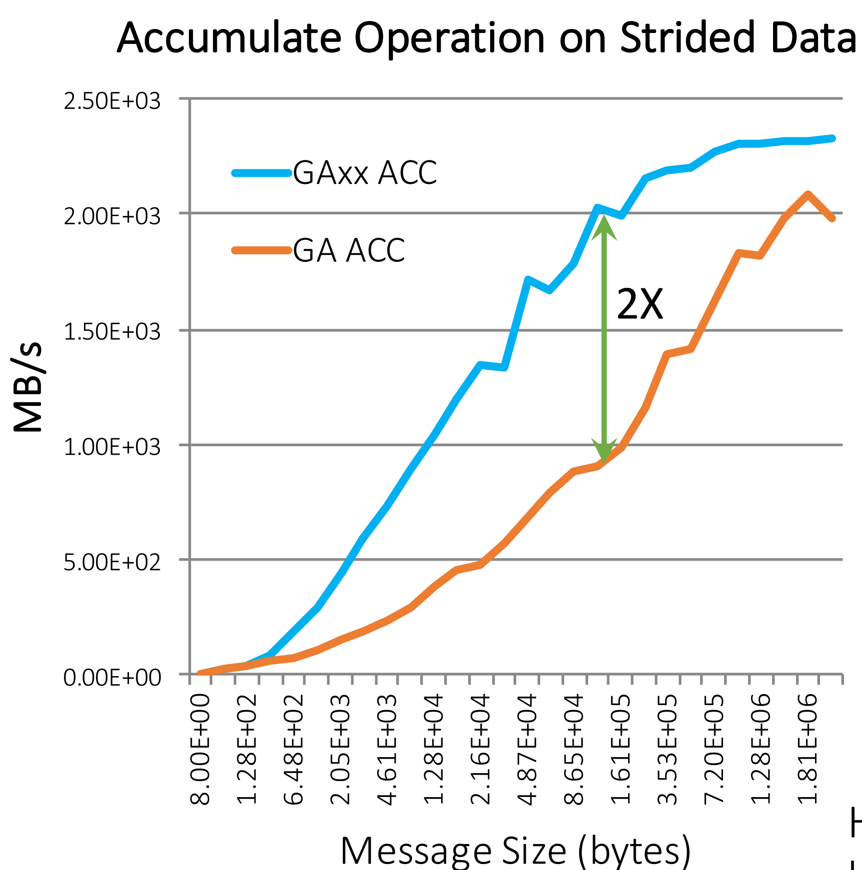
- 45% of execution
- 96% of FLOPs
- 3.3 cyc/FLOP
- 2.6% is comm or synch



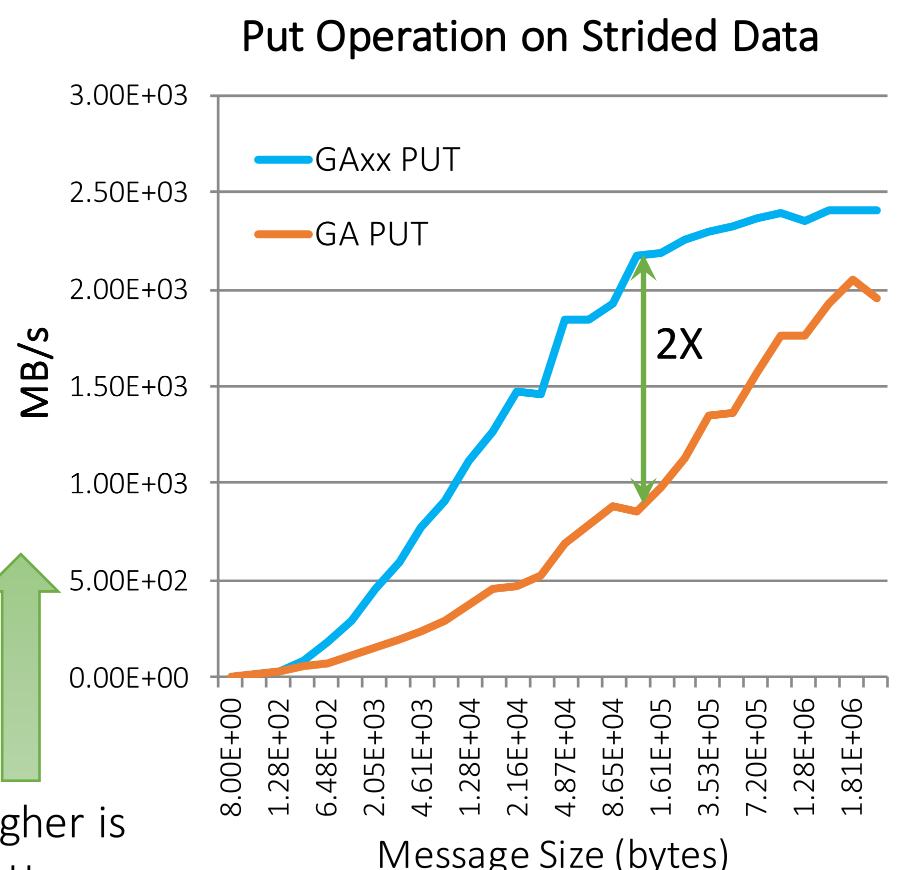
Overall conclusions

- No fundamental inefficiencies observed in GASNet communication substrate
- Insufficient and imbalanced parallelism seems to be the cause of comm and sync inefficiencies

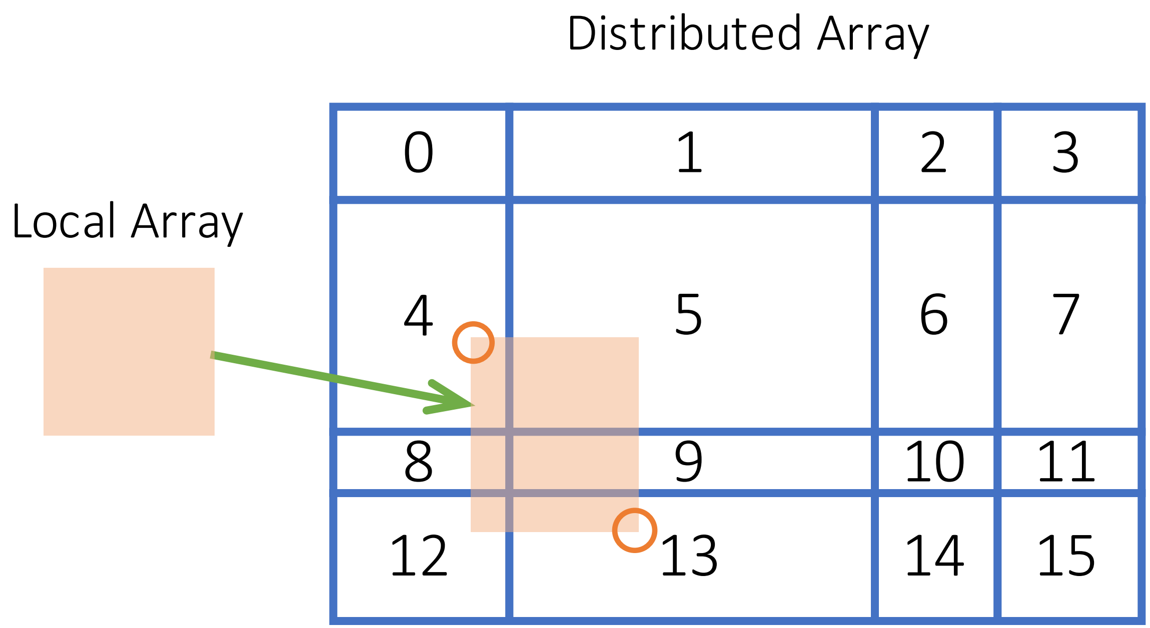
Better Strided Data Movement Using Active Message Pipelines



Higher is better



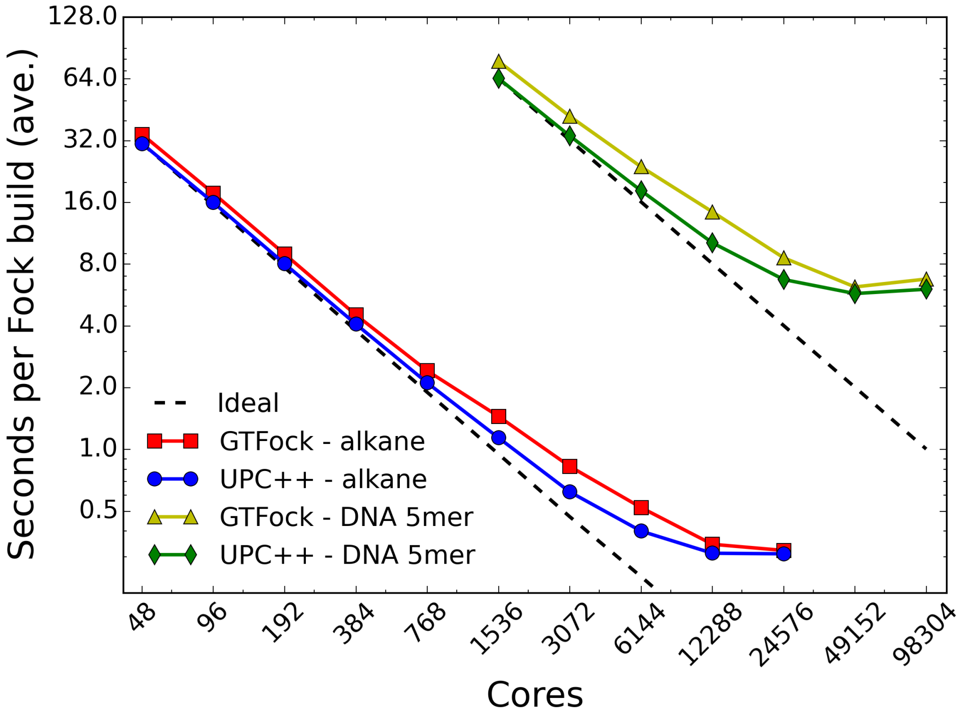
Irregular Submatrix Update

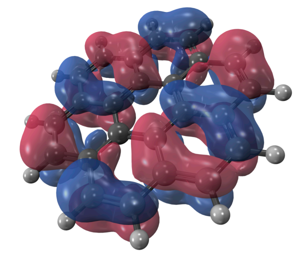


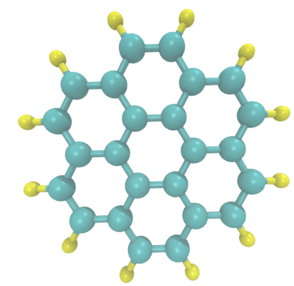
- Dynamic work stealing and fast atomic operations enhance load balance
- New distributed array abstraction delivers productivity and performance

UPC++ Fock Scales to 96K Cores

Strong Scaling on Edison (Cray XC30)



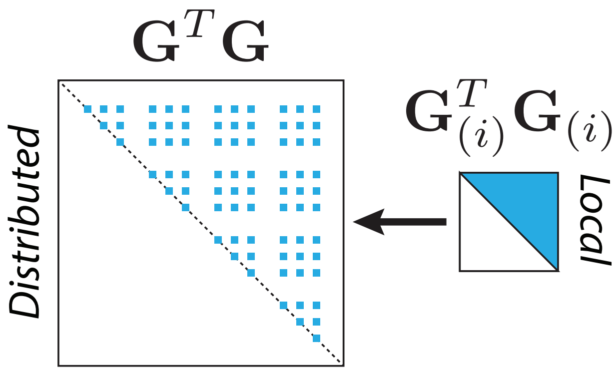




20% faster than the best existing solution

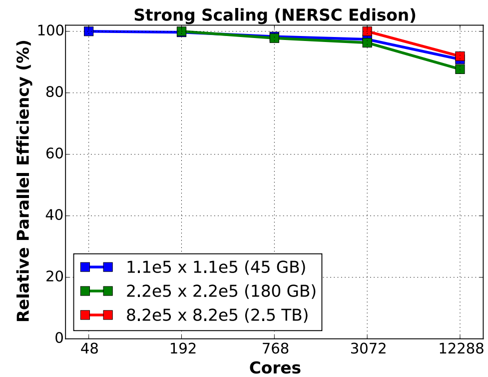
D. Ozog, A. Kamil, Y. Zheng, P. Hargrove, J. R. Hammond, A. Malony, W. de Jong, K. Yelick; *"A Hartree-Fock Application using UPC++ and the New DArray Library",* IPDPS 2016

UPC++ Enabled New Seismic Discovery

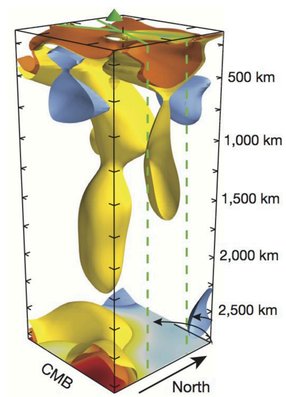



```
GtG[ix,ix] += GtG_i[:,:]
```

- Problem
  - Massive data don't fit in single memory
  - Dynamic and irregular update patterns
- Solution
  - PGAS + Asynchronous Remote Task Execution using Customized App Logics
- Impact
  - *First-ever whole-mantle seismic model from numerical waveform tomography*
  - *Reveals new details of deep structure not seen before*



Excellent parallel efficiency for strong scaling due to near complete overlap of computation and communication (IPDPS'15)



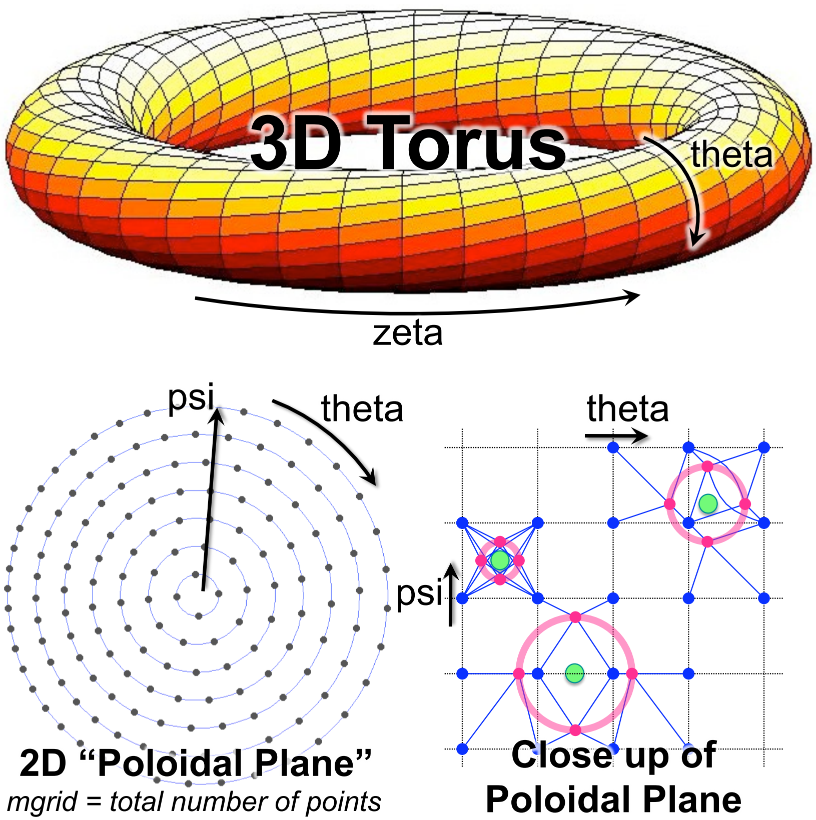
3D rendering of low-velocity structure beneath the Hawaii hotspot

Scott French, Barbara Romanowicz, *"Broad plumes rooted at the base of the Earth's mantle beneath major hotspots"*, ***Nature*, 2015**

NERSC 2016 Achievement Award for Innovative Use of HPC

Gyrokinetic Toroidal Code (GTC-P)

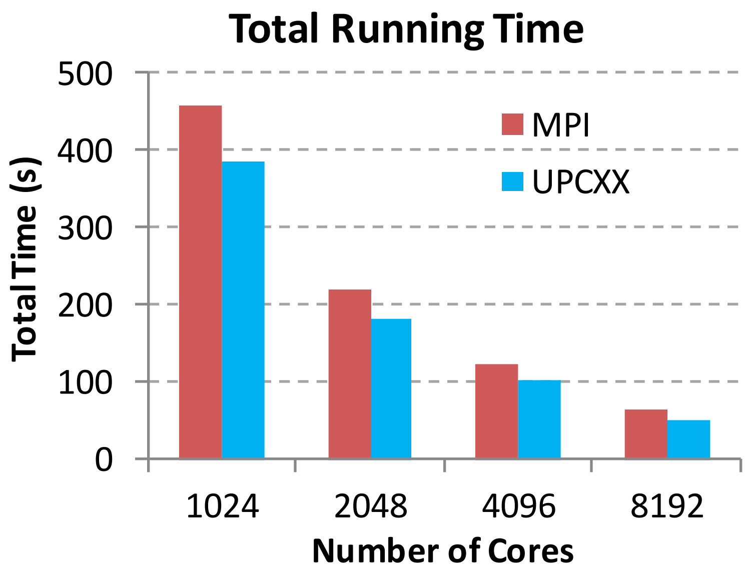
- A particle-in-cell (PIC) code that solves the five-dimensional (5D) gyrokinetic Vlasov equation in full, global torus geometry to address kinetic turbulence issues in magnetically-confined fusion experimental facilities tokamaks.

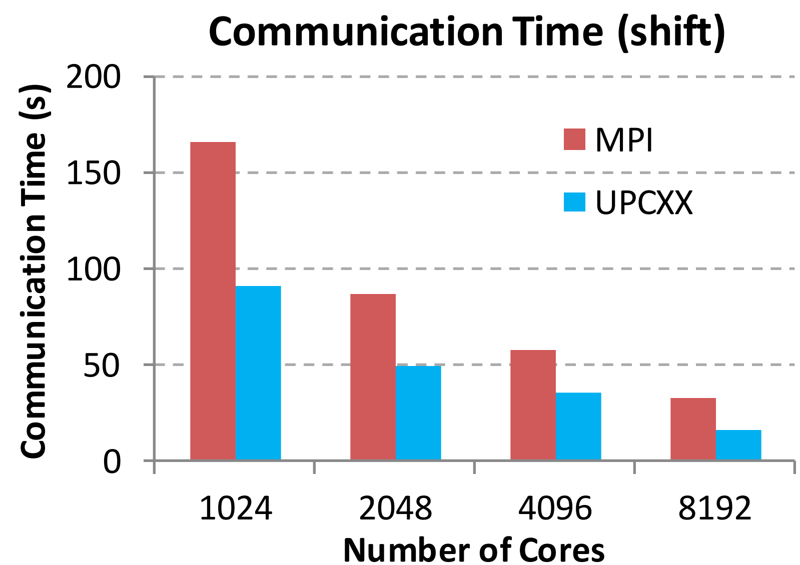


- A highly scalable code with three levels of parallelism and vectorization:
  - Toroidal domain decomposition
  - Poloidal domain decomposition
  - Particle decomposition
- **Network performance** becomes increasingly important factor for the overall performance
  - Using one-sided communication for performance improvement

PGAS GTC-P Performance Improvement

- *Total running time improved 18-28%*
- *Communication time improved 125-200%*





	Communication Time Breakdown		
	Pack	Transfer	Unpack
MPI	5.8	17.7	7.3
UPC++	10.7		0.0

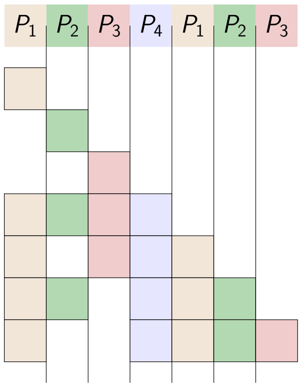
Overlap packing and data transfer

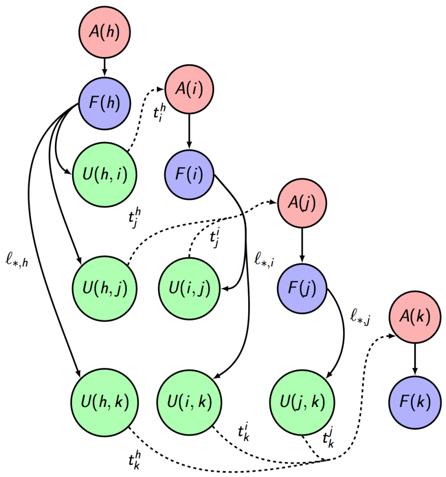
Directly write to the destination; no unpack

Sparse Symmetric Matrix Solver

- Parallel sparse Cholesky solver **symPACK**
  - Symmetric matrices in many applications:

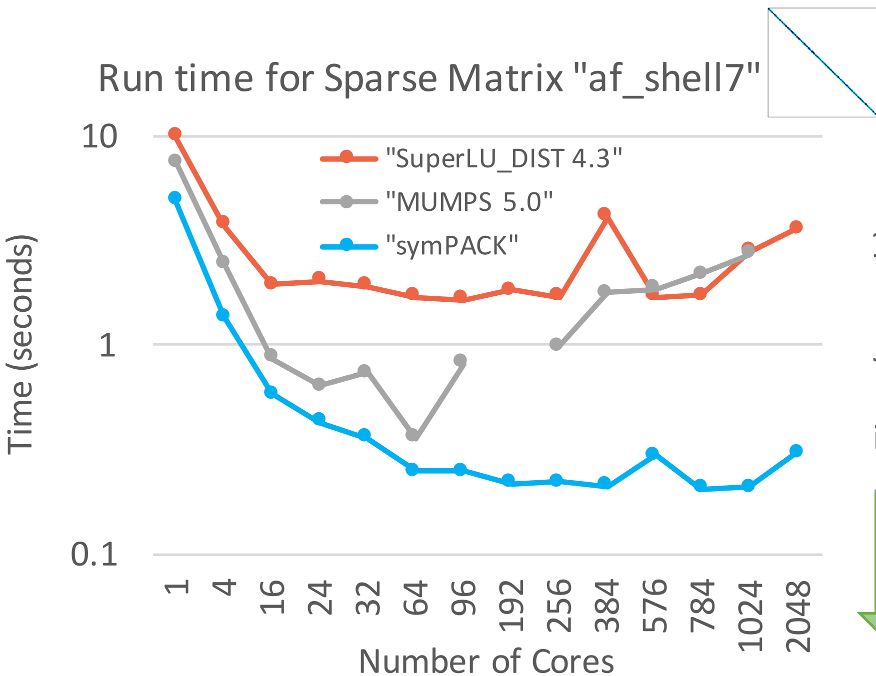
    Optimization problems , PDE discretization, …
  - Symmetry = less computations, lower memory
- Algorithm expressed by a DAG of tasks
- Asynchronous remote task execution and one-sided communication provided by UPC++
- Dynamic scheduling of local *ready* tasks

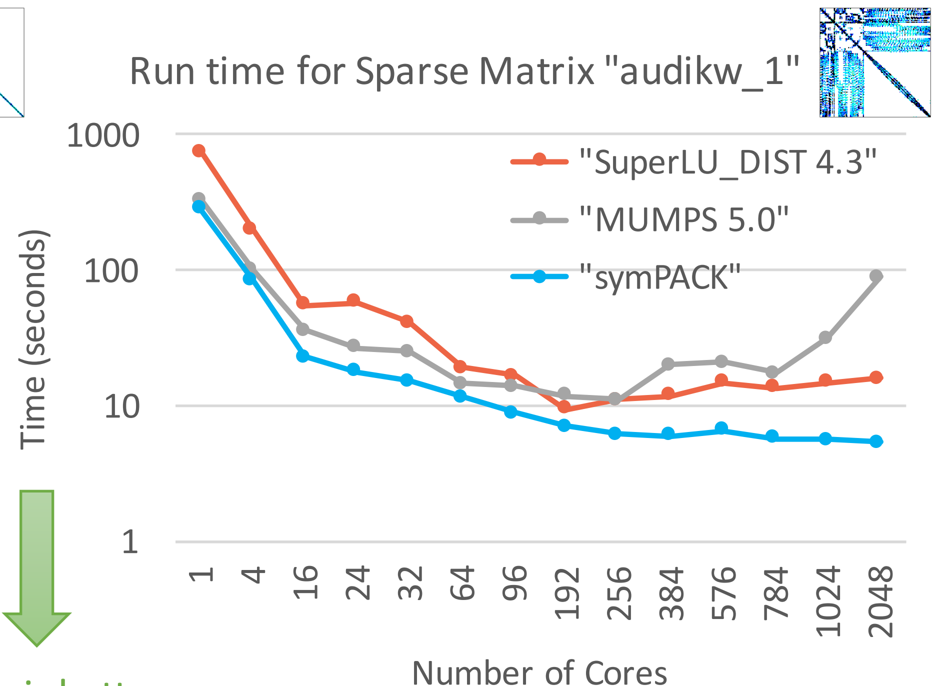




Mathias Jacquelin, Esmond Ng (FASTMath)

SymPACK Strong Scaling on Edison





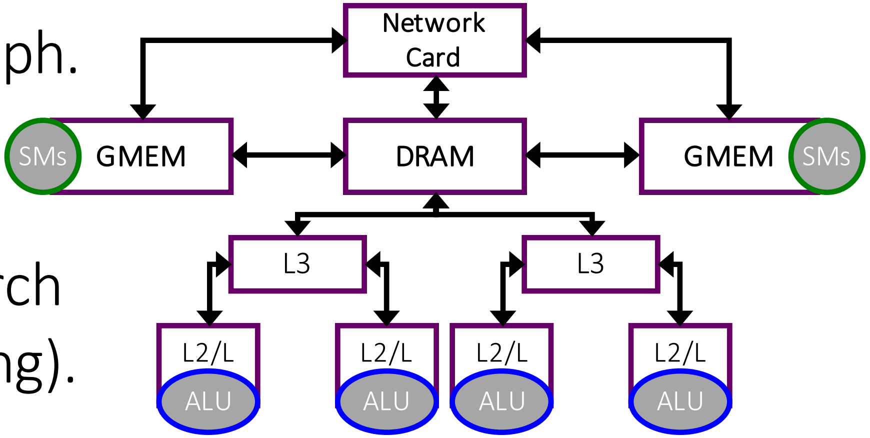
Lower is better

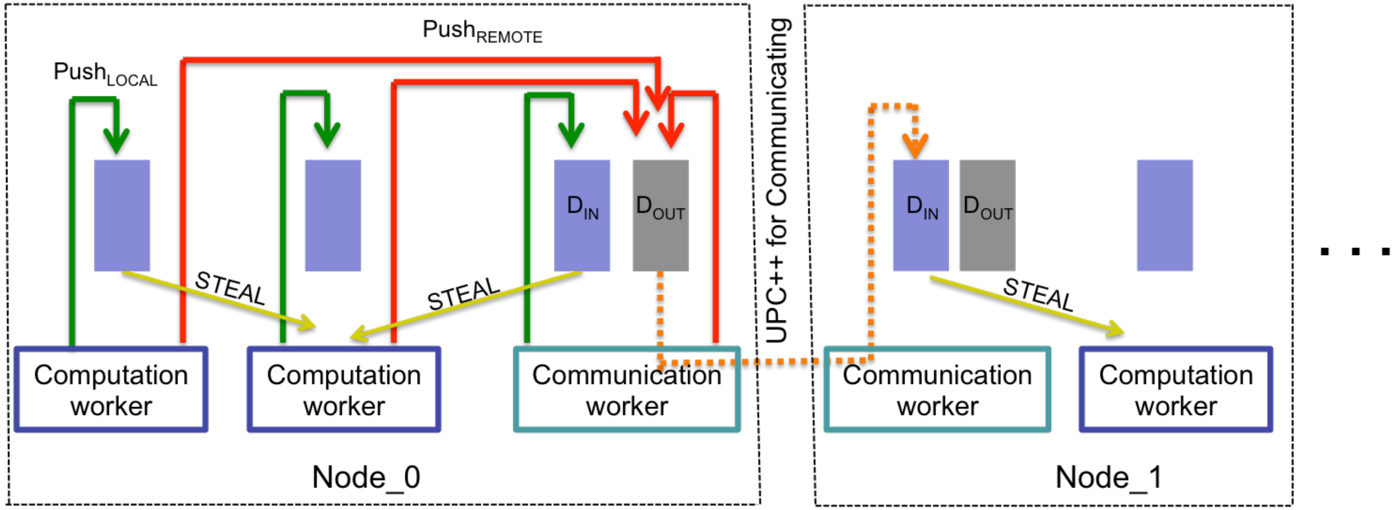
SymPACK supercharged by UPC++ is 45%-105% faster than MUMPS.

Note: SuperLU doesn't have special treatment for symmetric matrices so its runtime is expected to be higher since it performs a regular LU factorization instead of a Cholesky factorization.

Habanero-UPC++: Locality-Aware and Support Heterogeneous Architecture

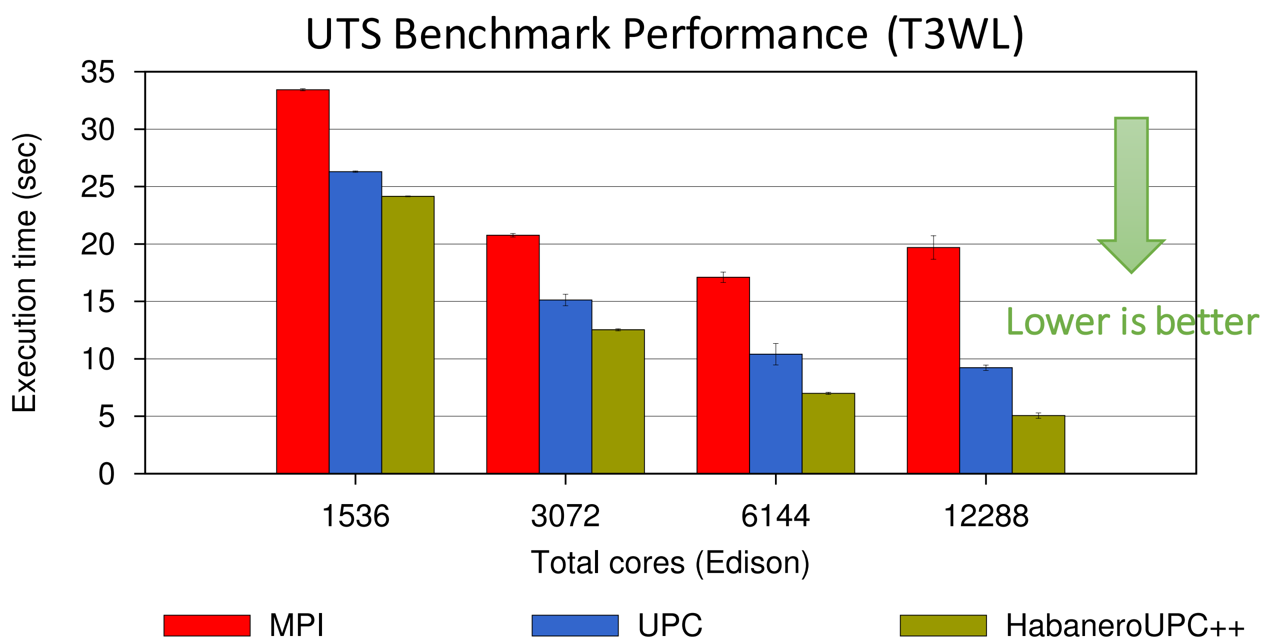
- Represent machine layout as a graph.
- Tasks are each associated with a locale; worker threads have static locale paths along which they search for tasks (generalized load balancing).





Distributed Load Balancing with Habanero-UPC++

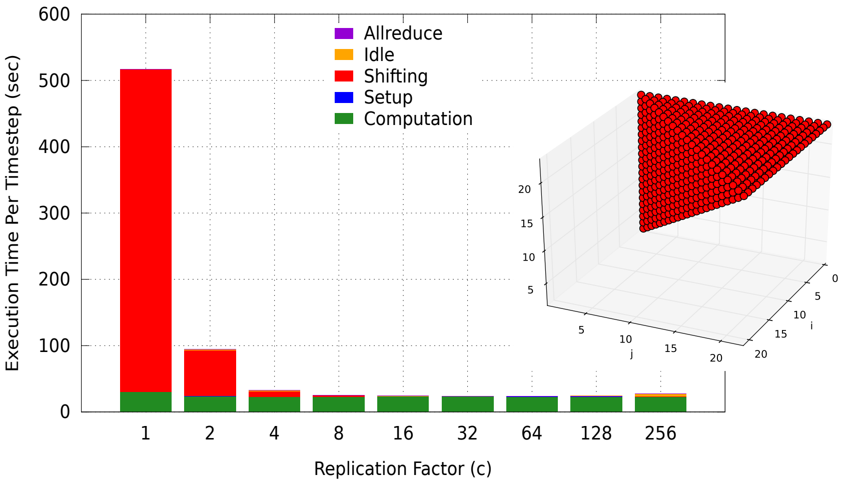
- A simple API to declare a **"locality-free"** task, which can participate in distributed load-balancing
- Habanero-UPC++ runtime uses a novel distributed work-stealing strategy that maximizes balance and minimizes overheads



Towards Communication-Optimal Compilers

- Many algorithms have provably-optimal variants
  - Linear algebra, dense/sparse
  - Direct N-body and now K-body
  - New Sparse/Dense for ML
- Generalize to compilers

Penporn Koanantakool and K. Yelick, SC'15



Thm: For nested loops, accessing arrays with subscripts that are linear functions of indices

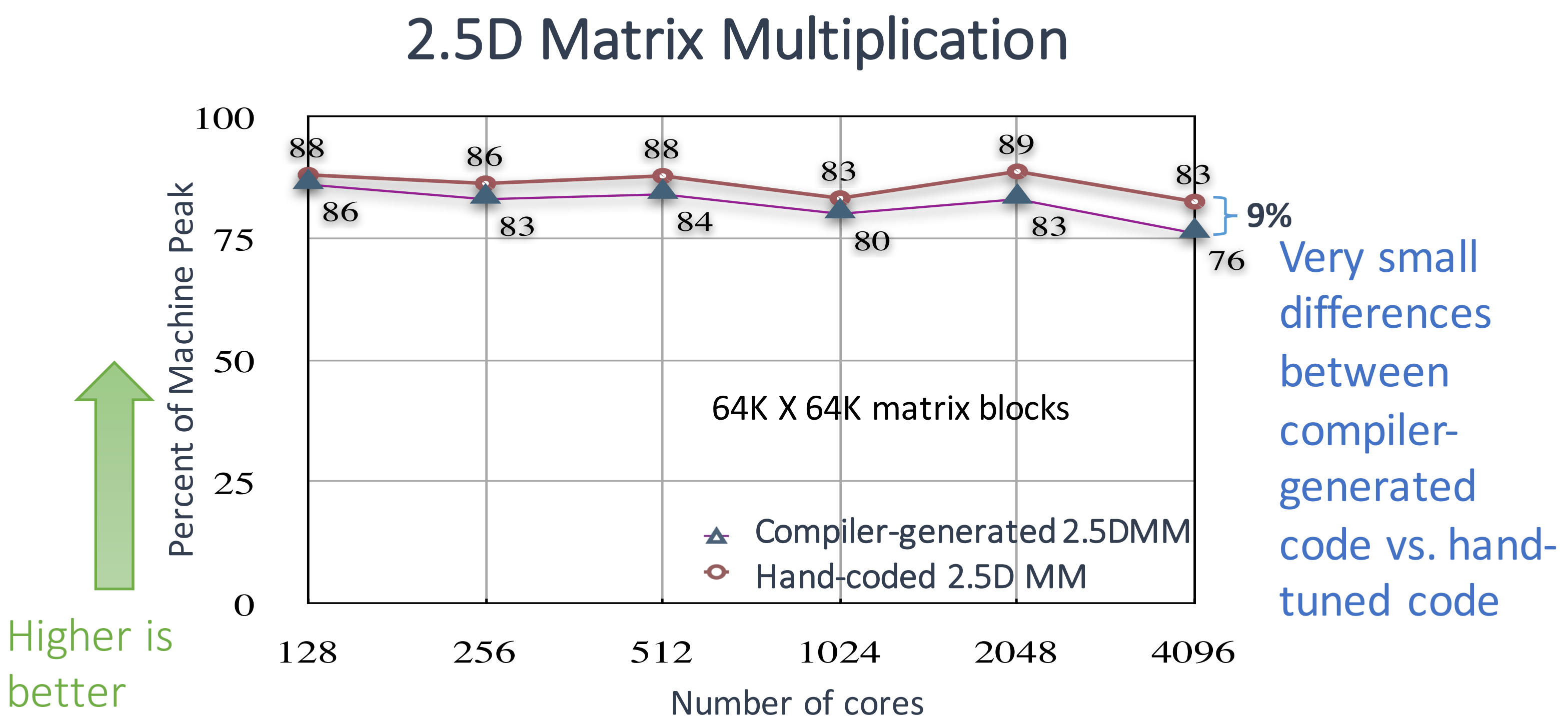
$$\#\text{words\_moved} = \Omega\,(\#\text{iterations}/M^{e})$$

for some e we can determine

Thm: Can sometimes determine the optimal tiles sizes up to constant factors

Christ, Demmel, Knight, Scanlon, Yelick

Compiling for Communication Avoiding Algorithms

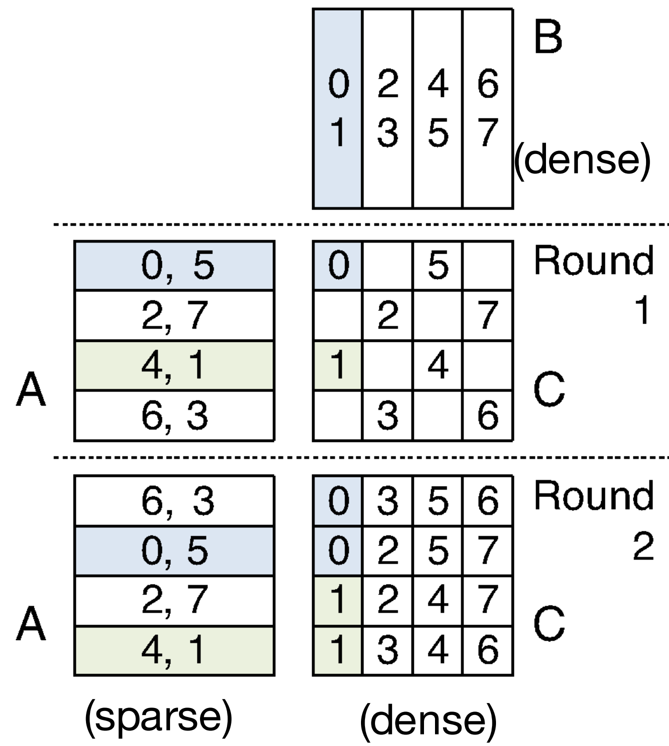


Compiler analysis and code generation for automating data movement to produce communication-optimal code

Karthik Murthy and J. Mellor-Crummey, PACT 2015

Sparse-Dense Matrix-Matrix Multiplication (SpDM^3)

- Building block of increasing number of applications
  - Machine learning and data analytics, algebraic multigrid, graph algorithms, quantum monte carlo simulations, etc.
- Communication can be the bottleneck
- Relatively understudied
  - Optimal parallel algorithms for dense-dense/sparse-sparse case move both matrix operands and are not always communication-optimal in this case.
- *New communication-avoiding algorithms move just the sparse matrix.* Observed up to ~100x speedup

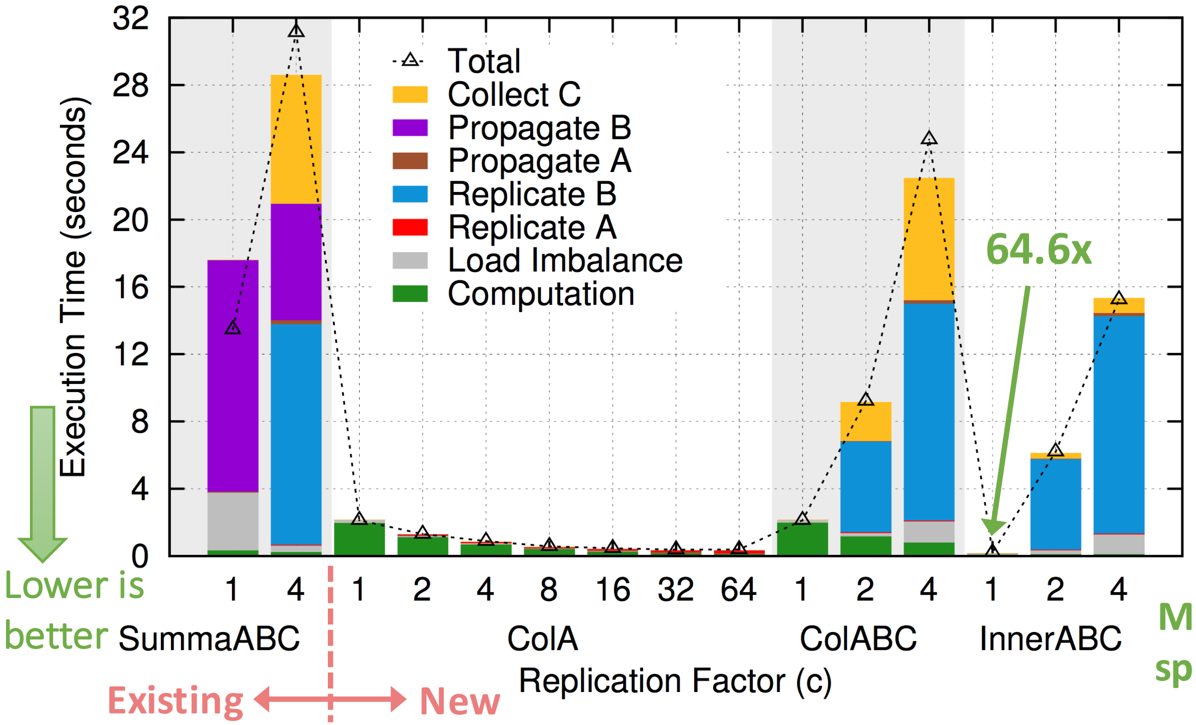


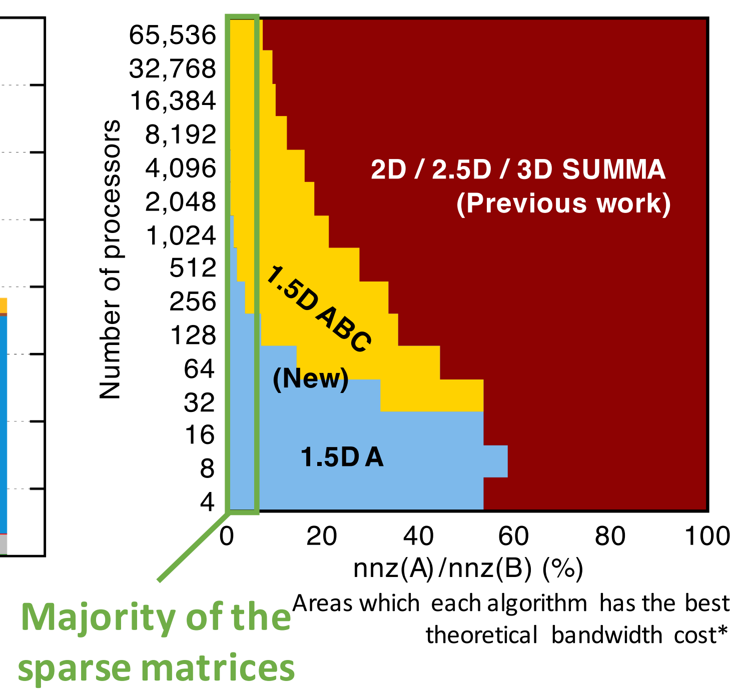
Koanantakool, Azad, Buluç, Morozov, Oh, Oliker, Yelick in IPDPS16

Communication-Avoiding SpDM^3

- Best choice depends on the # of nonzeroes of each matrix
- Also applicable to dense-dense/sparse-sparse cases with different # of nonzeroes

Cost breakdown for $A_{66k x 172k}$ (0.004% nnz) * $B_{172k x 66k}$ on 768 cores of Cray XC30





Areas which each algorithm has the best theoretical bandwidth cost*

*Assuming nnz(C) = nnz(B)

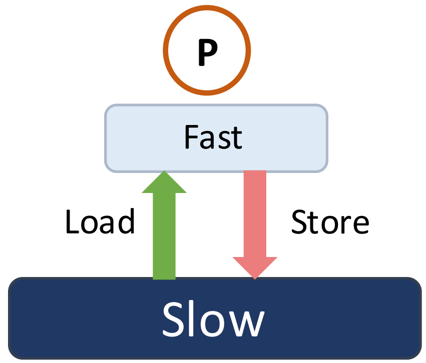
Koanantakool, Azad, Buluç, Morozov, Oh, Oliker, Yelick in IPDPS16

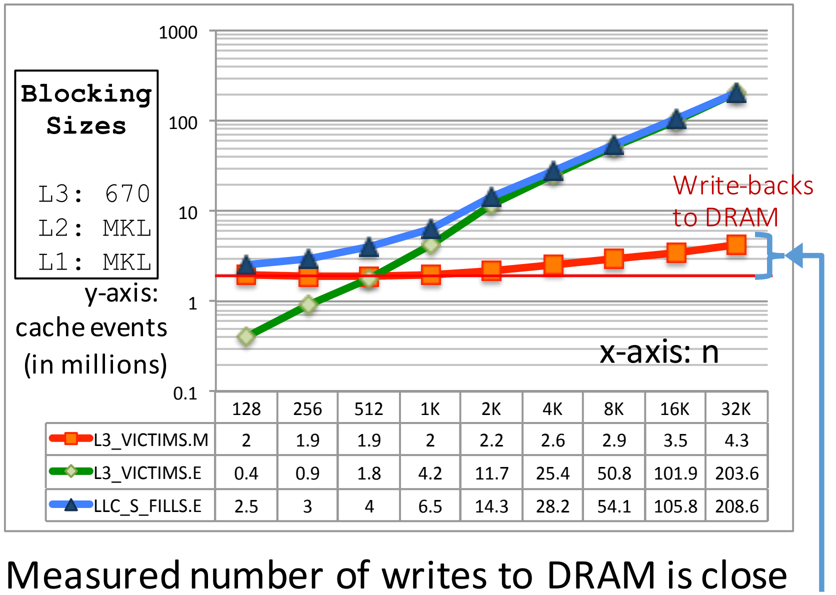
Write-Avoiding Algorithms

- Writes are more expensive than reads for some memory technology (NVM)
- Results:
  - Classical Direct LA solvers, N-body methods, and Krylov methods need asymptotically fewer writes than reads
  - Fast algorithms (FFT, Strassen) and Cache-Oblivious classical direct linear algebra cannot be write-avoiding.

Theoretical foundations for improving performance of algorithms on machines with NVM

Erin Carson, James Demmel, Laura Grigori, Nicholas Knight, Penporn Koanantakool, Oded Schwartz, Harsha Vardhan Simhadri, *"Write-Avoiding Algorithms"*, IPDPS'16





	128	256	512	1K	2K	4K	8K	16K	32K
L3_VICTIMS.M	2	1.9	1.9	2	2.2	2.6	2.9	3.5	4.3
L3_VICTIMS.E	0.4	0.9	1.8	4.2	11.7	25.4	50.8	101.9	203.6
LLC_S_FILLS.E	2.5	3	4	6.5	14.3	28.2	54.1	105.8	208.6

Measured number of writes to DRAM is close to the theoretical prediction.

GASNet

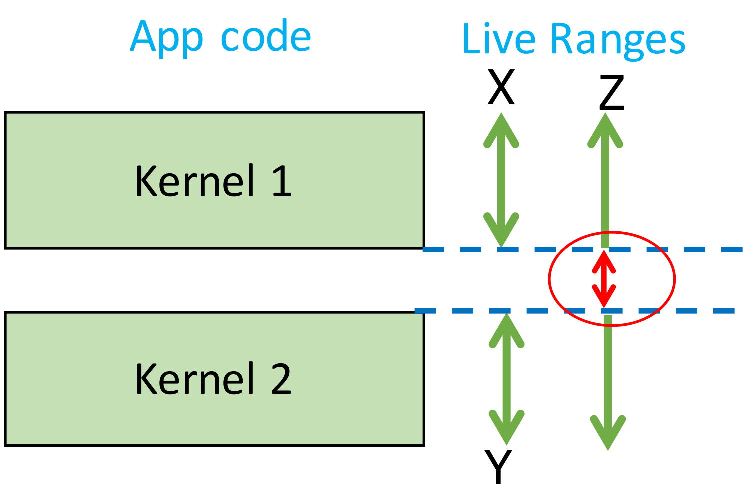
- GASNet is "Global Address Space Networking"
  - A communications library for Partitioned Global Address Space (PGAS) languages and libraries, supporting RMA (Put/Get) and Active Messages.
  - A project of Lawrence Berkeley National Laboratory (LBNL) and the University of California at Berkeley (UCB), begun in 2002 to support UPC and Titanium.
  - Runs on everything from laptops to supercomputers.
- GASNet has become the de facto standard in its field, with projects using it for their communications including:
  - Unified Parallel C ("UPC")
    - Berkeley UPC (LBNL and UCB)
    - GNU UPC (Intrepid Technology)
    - Clang UPC (Intrepid Technology)
    - UPC for Cray XT (Cray)
  - Fortran 2008 Coarrays
    - OpenUH Fortran compiler (UH)
    - OpenCoarrays for gfortran
    - CAF for Cray XT (Cray)
  - CAF 2.0 (Rice)
    - A superset of Fortran 2015
  - OpenSHMEM (UH and ORNL)
    - Reference implementation
  - Legion (Stanford)
  - UPC++ (LBNL)
  - Habanero-UPC++ (Rice and LBNL)
  - Global Arrays / NWChem (LBNL)
    - Emerging prototypes
  - Titanium (UCB)
  - Cray Chapel (Cray)
  - And more …

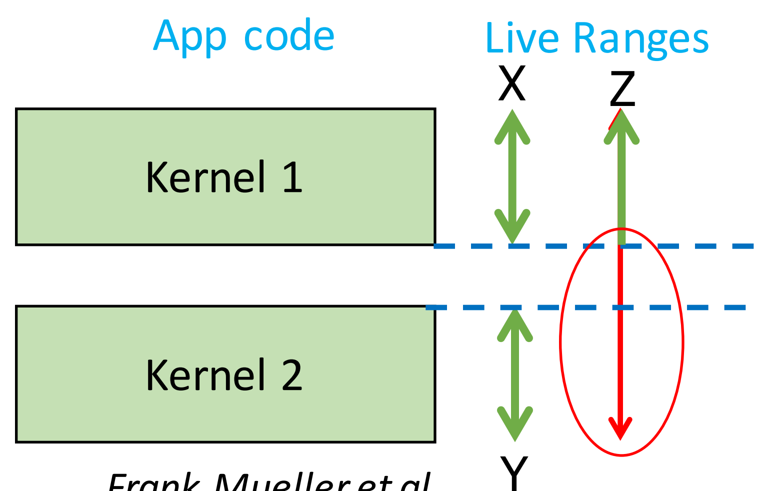
GASNet-EX

- GASNet-EX modernizes GASNet for Exascale
- Incorporates 15 years worth of "lessons learned"
- Recognizes that requirements have changed significantly
  - From few to hundreds of CPU threads per NIC
  - From modest to huge memory per node (and thus NIC)
  - From PGAS to Asynchronous PGAS (APGAS) languages
- Major modernization themes include
  - Standardize existing extensions to GASNet
  - Support multiple clients (e.g. hybrid apps)
  - Support resilient clients
  - Support threads as first-class entities
  - Better manage "time" (polling)
  - Better manage "space" (buffers)
  - Discard some legacy baggage

End-to-end Resilience

- Resilience against ***soft errors*** in HPC
- Many existing algorithm-based fault tolerance (ABFT) techniques only protect data within a kernel but errors may happen when data live across kernels (regions)
- E2E resilience protects data structures spanning across phases of alternating resilience techniques





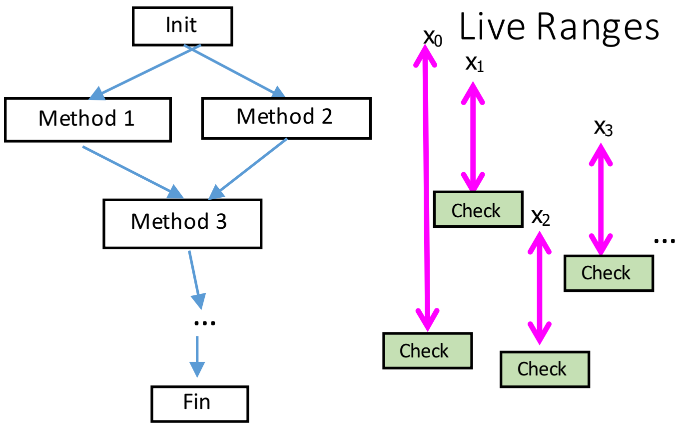
Frank Mueller et al, NCSU

End-to-end Resilience (cont.)

- Add pragma for critical data structures
- Checker method
  - Convergence tests (numeric solvers)
  - Checksum of data
- Recovery method
  - Forward recovery or
  - Restore data (checkpoint)

```
Load(A); Load(B)
#pragma protect Check(isOK(A),isOK(B))
         Recover(Fix(A), Fix(B))
mmult(A,B,C)
Load(D)
#pragma protect Check(isOK(C), isOK(D))
         Recover(Fix(C),Fix(D))
mmult(C,D,E)
#pragma protect Check(isOK(E))
         Recover(Fix(E))
Store(E)
```

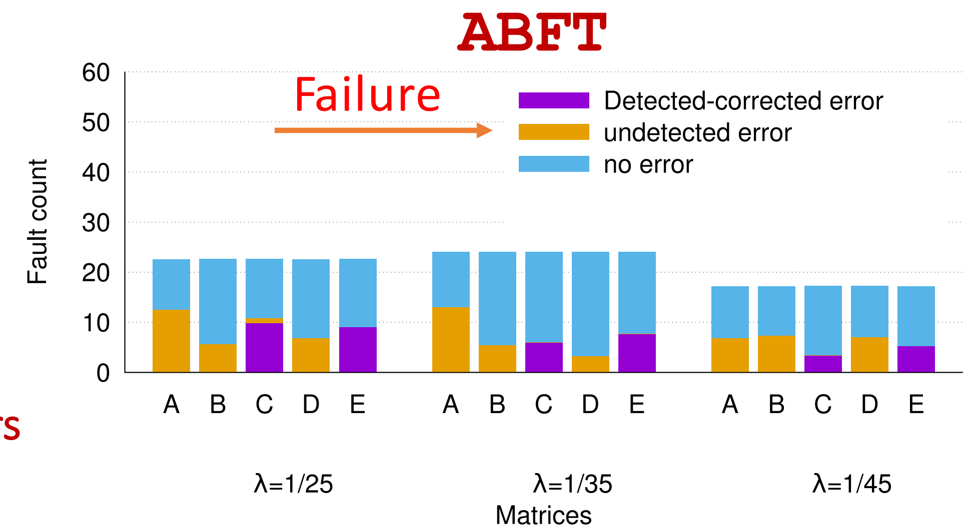
- Add check after ***last use*** per variable
- SUCCESS: No action is taken
- FAIL: Correct the value or recompute



Results: Fault Count (ABFT vs. End-to-end)

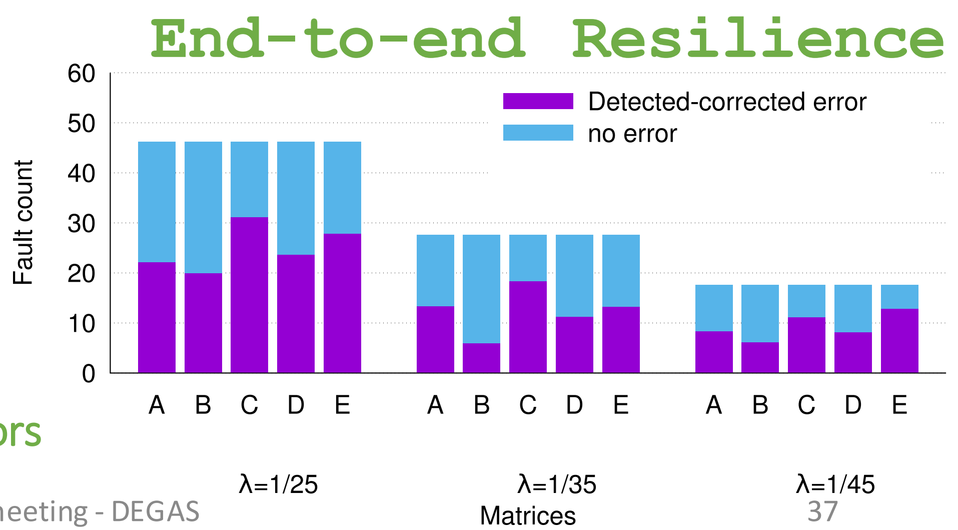
- Blocked matmult, 2560x2560
- Matrices are 2560 x 2560
- Fault rates: one per 25/35/45 secs

ABFT → Failure cases are undetected errors



- End-to-end performs checks after last use of every matrix (ABFT matrix checksum)
- End-to-end might trigger re-computation when cannot fix errors

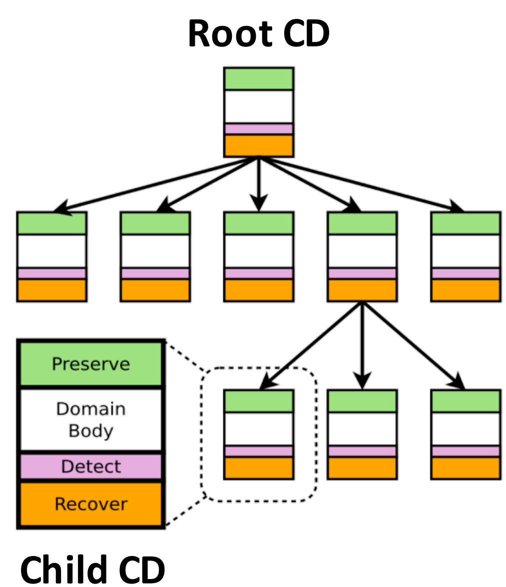
End-to-end → can still contain all the errors



Containment Domains for DEGAS

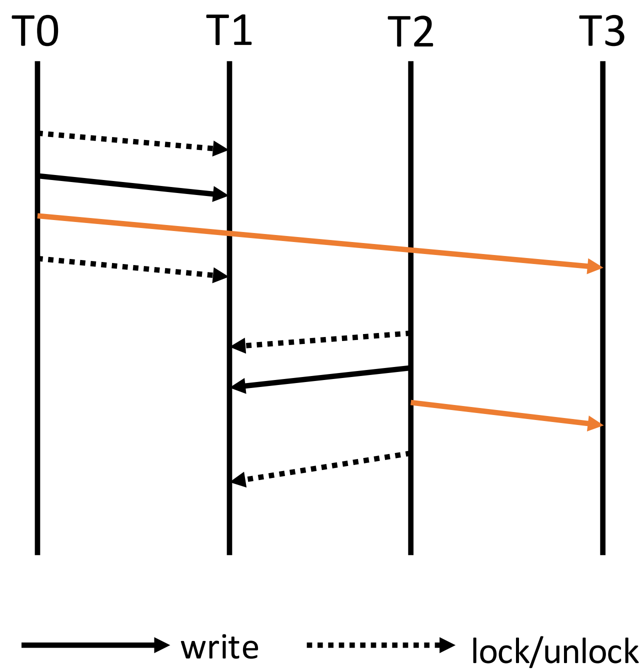
- Allow applications to express resilience concerns
    - Simple consistent abstraction
    - Define consistent state points for PGAS resilience
    - Enable resilience optimizations
- Provide partial rollback and error handling for applications
- Serve as a driver for defining resilience of other system components: GASNet-EX

  Components
    - *Preserve* data on domain start
    - *Compute* (domain body)
    - *Detect* faults before domain commits
    - *Recover* from detected errors
    - Semantics
    - Erroneous data never communicated
    - Each CD provides recovery mechanism



Consistent Localized PGAS Recovery

- Semantics and protocols for distributed PGAS recovery not well researched before
- We identified gaps in a very-recently proposed protocol [Besta and Hoefler, 2014]
  - Some race-free cases cannot be consistently replayed
  - Inefficiencies in implementation for fine-grained recovery
- We developed a new protocol and defined semantics
  - Introduce a network-level counter to order incoming writes from remote nodes
  - Races between local accesses and remote writes disallowed



- Designed a new logging framework for highly-localized recovery

CD DEGAS Accomplishments

- New protocol and semantics for consistent distributed recovery in PGAS systems
  - Determined requirements for ensuring global consistent view
  - Closed gaps in prior-work semantics
- Developed new designs for improving recovery locality
  - Combining local and remote logging options to minimize global recovery actions
- Helped define and determine GASNet-EX resilience
- UPC++ implementation of CD runtime prototype
  - Support of CD management, preservation, and restoration
  - Full support for strict CDs
  - Support for communication logging and runtime logging for relaxed CDs (currently partial support)

DEGAS Software Technologies Pipeline

Communication-Avoiding Algorithms

Theories and Designs

CA Compiler, CD for PGAS, End-to-End Resilience, Snowflake

Research Prototypes

UPC++, Habanero-UPC++, AMR, Chemistry, Genomics, Machine Learning, Seismic

Application Demonstrations

UPC, GASNet, BLCR

Production Software

Over 50 Papers!

Demos Tonight!

- *Leveraging HipMer via NERSC Web Portal*
- *Containment Domains Resilience*
- *Understanding the Performance Characteristics of PGAS Codes*

Software Products:

http://crd.lbl.gov/departments/computer-science/CLaSS/research/DEGAS/degas-software-releases

Publications:

http://crd.lbl.gov/assets/Uploads/FTG/Projects/DEGAS/DEGAS-products-April2016.pdf