LBNL and University of Tsukuba CCS Joint Workshop
March 19-20, 2012

The ACTS Project overview and future directions

Osni Marques
Lawrence Berkeley National Laboratory
OAMarques@lbl.gov

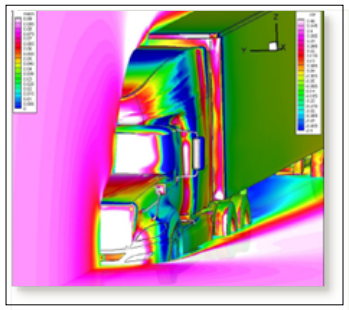
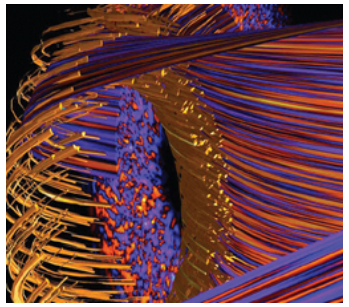
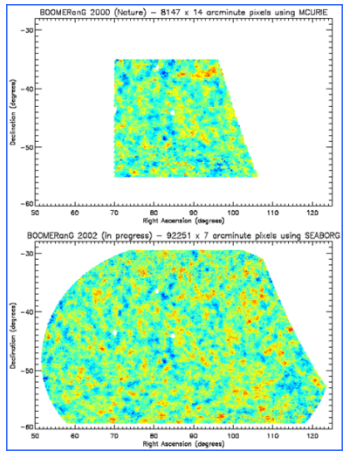
Motivation: HPC Applications

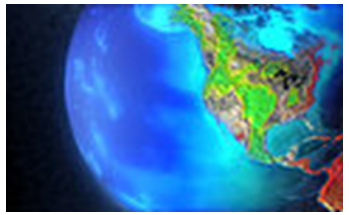
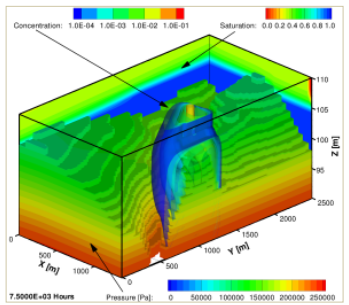
- Accelerator Science
- Earth Sciences
- Material Sciences
- Biology
- Chemistry
- Astrophysics

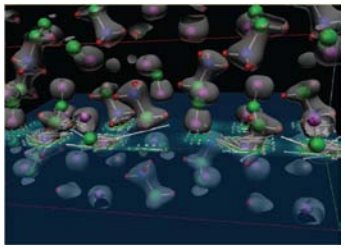
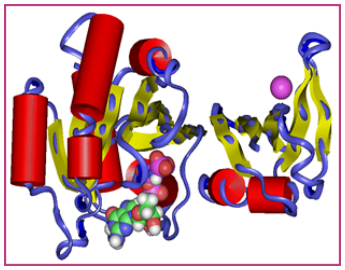
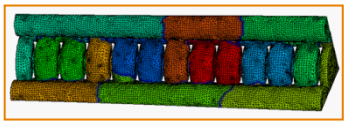
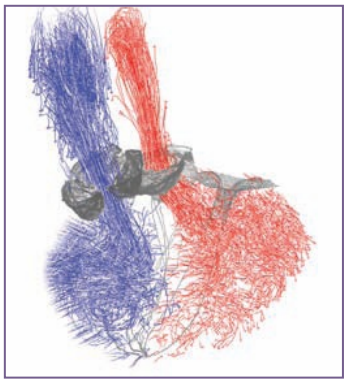
⋮

Commonalities

- Advancements in science and engineering
- Increasing demands for computational power
- Reliance on available computational systems, languages, and software tools

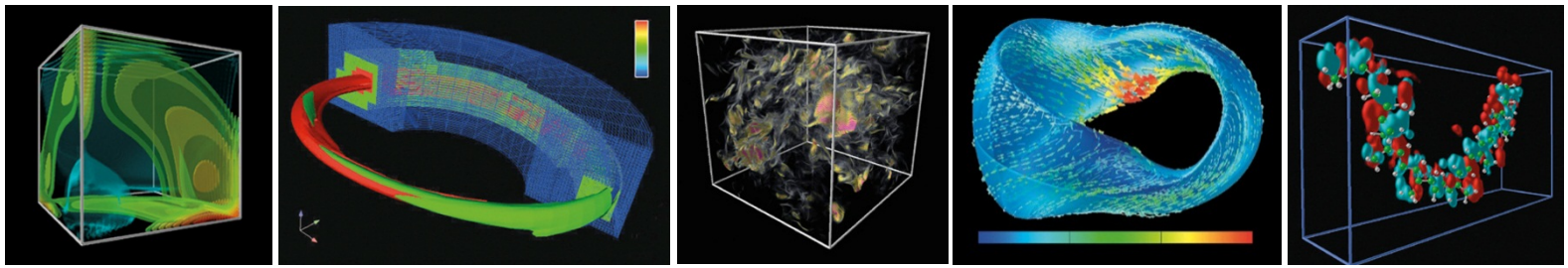



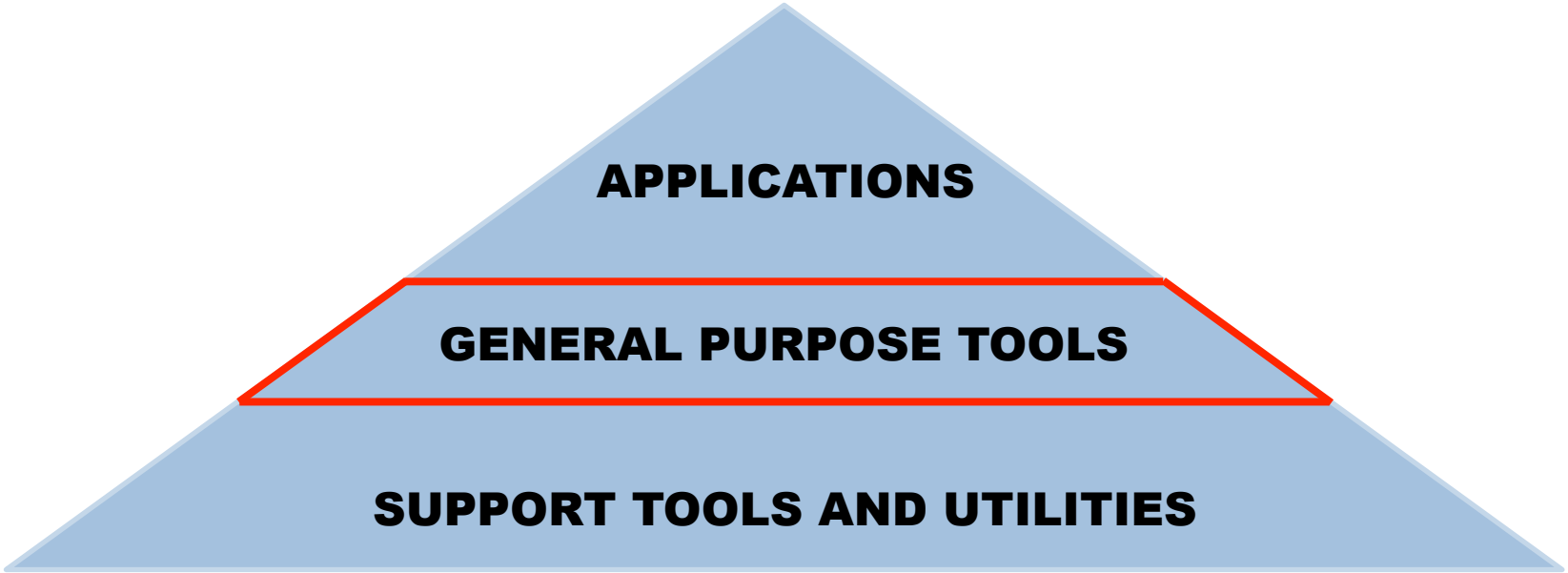
The DOE ACTS Collection

- **Goal:** The Advanced CompuTational Software Collection (ACTS) makes reliable and efficient software tools more widely used, and more effective in solving the nation's engineering and scientific problems

- Long term maintenance
- Independent test and evaluation
- Outreach and dissemination
- High level user support



Software Stack



HARDWARE

Leading technology paths (swim lanes):

- **Multicore: maintain complex cores, and replicate (x86 and Power7, Blue Waters, NGSC)**
- **Manycore/embedded: use many simpler, low power cores from embedded systems (BlueGene, Dawning)**
- **GPU/Accelerator: use highly specialized processors from gaming/graphics market space (NVidia Fermi, Cell, Intel Knights Corner/Larrabee)**

Risks in swim lane selection:

- **Select too soon: users cannot follow**
- **Select too late: fall behind performance curve**
- **Select incorrectly: subject users to multiple disruptive technology changes**

The DOE ACTS Collection: Current Functionalities

Category	Tool	Functionalities
Numerical	Trilinos	ge sparse linear systems
	Hypre	ge sparse linear systems (grid-centric interfaces)
	PETSc	ar and nonlinear systems of equations)
	SUNDIALS	ferential equations, nonlinear algebraic
	ScaLAPACK	
	SLEPc	
	SuperLU	large, sparse, nonsymmetric linear
	TAO	blems (nonlinear least squares, unconstrained minimization, bound constrained optimization)
Code Development	Global Arrays	Library for writing parallel programs that use large arrays distributed across processing nodes and that offers a shared-memory view of distributed arrays
	Overture	Framework for solving partial differential equations in complex geometries.
Code Execution	TAU	Set of tools for analyzing the performance of multi-language programs
Library Development	ATLAS	Tools for the automatic generation of optimized numerical software (dense linear algebra)

Linear Solvers (Iterative Methods):

- **CG**
- **GMRES**
- **CGS**
- **BiCGSTAB**
- **QMR**
- **SYMMLQ**

- **Various preconditioners**

- **Multigrid**

ased)

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Software Interfaces

```
CALL BLACS_GET( -1, 0, ICTXT )
CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL )
   :
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
   :
CALL PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB, INFO )
```

function call

(ScaLAPACK)

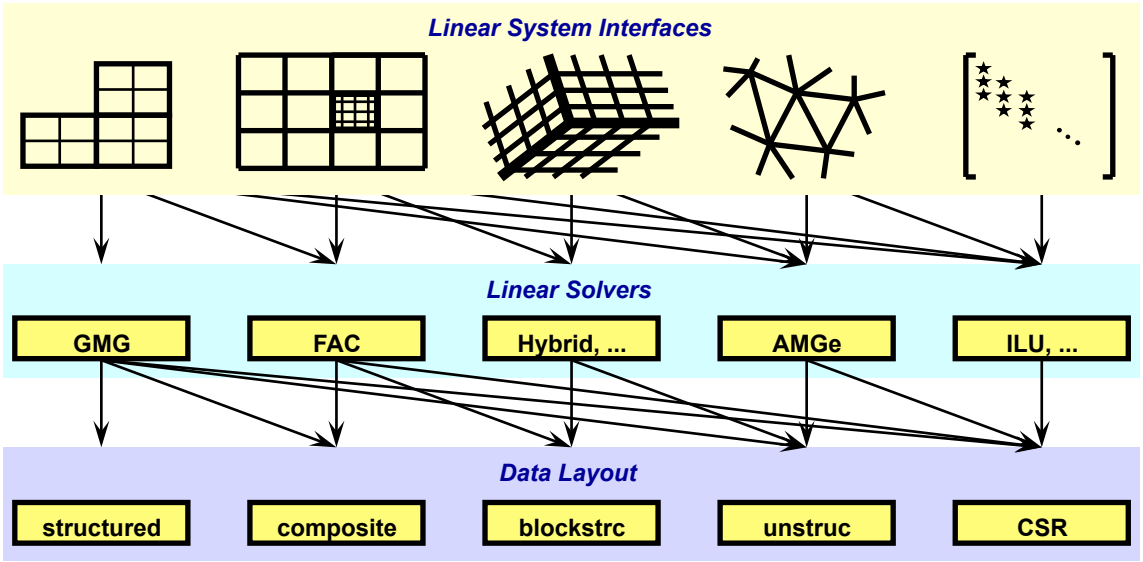
command line

(PETSc)

- -ksp_type [cg,gmres,bcgs,tfqmr,…]
- -pc_type [lu,ilu,jacobi,sor,asm,…]

More advanced:

- -ksp_max_it <max_iters>
- -ksp_gmres_restart <restart>
- -pc_asm_overlap <overlap>
- -pc_asm_type [basic,restrict,interpolate,none]



problem domain

(Hypre)

Questions for application developers

- How does performance vary with different compilers?
- Is poor performance correlated with certain OS features?
- Has a recent change caused unanticipated performance?
- How does performance vary with MPI variants?
- Why is one application version faster than another?
- What is the reason for the observed scaling behavior?
- Did two runs exhibit similar performance?
- How are performance data related to application events?
- Which machines will run my code the fastest and why?
- Which benchmarks predict my code performance best?
- ⋮

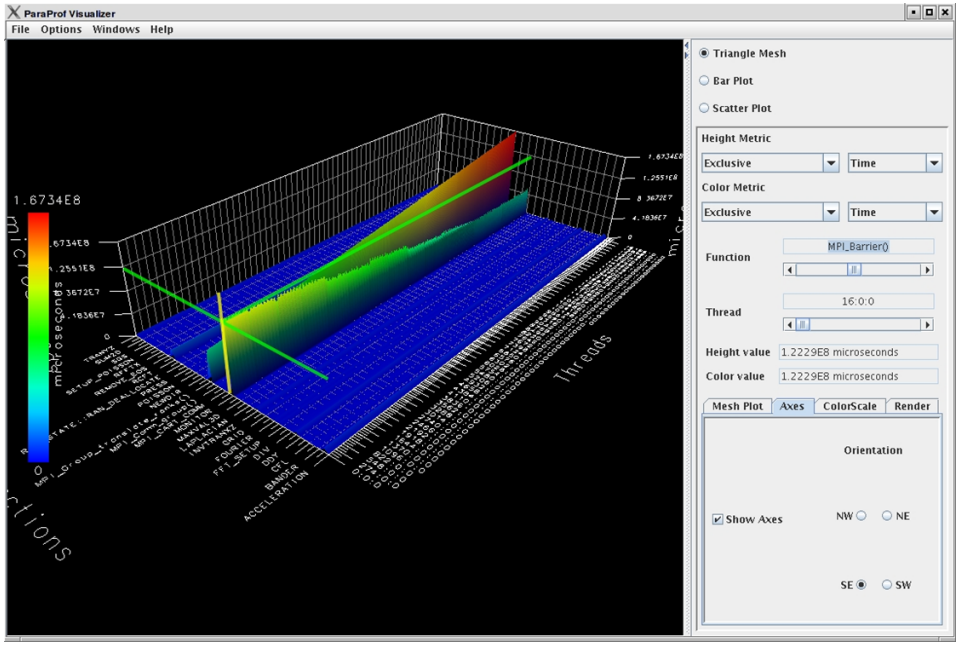
(*courtesy of Sameer Shende*)

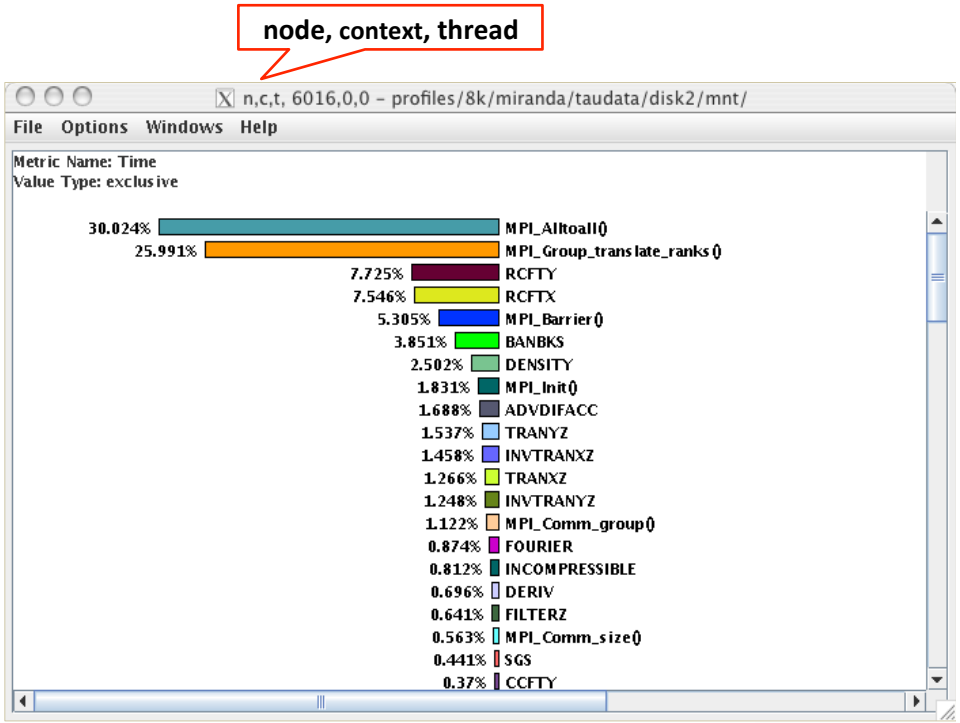
TAU: Performance Analysis

- **Profiling**: summary statistics of performance metrics (# of times a routine was invoked exclusive or inclusive time or hardware counts, calltrees and callgraphs, memory and message sizes etc)
- **Tracing**: when and where events took place along a global timeline (timestamped log of events, message communication events)
- Automatic instrumentation of source code (PDT)
- Runs on basically all HPC platforms

- 3D profile browser (paraprof)
- To use TAU, one only needs to set a couple of environment variables and substitute the name of the compiler with a TAU shell script

- Ex. Flat profile of Miranda (LLNL; hydrodynamics / Fortran + MPI) on an BG/L;

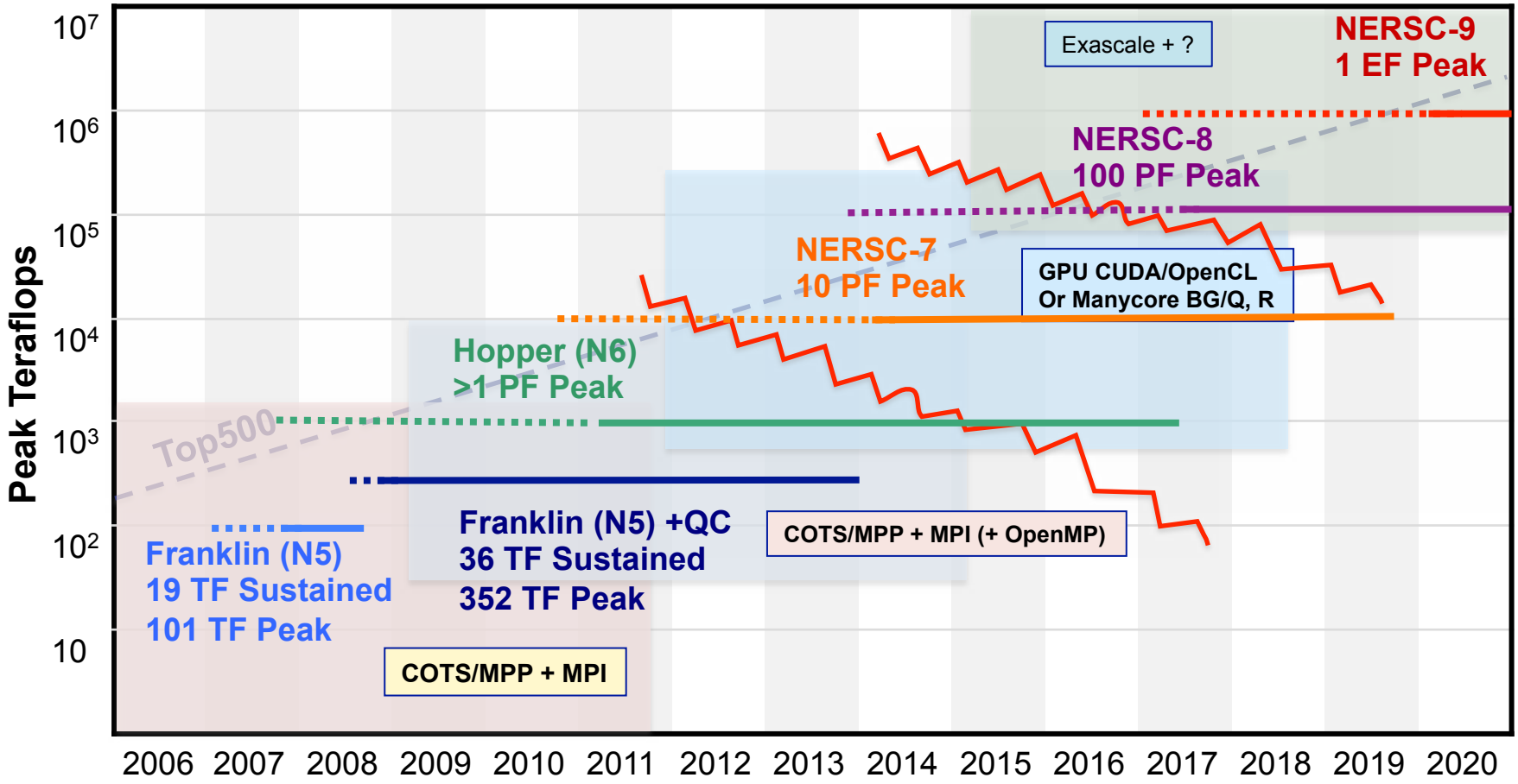




(See http://tau.uoregon.edu/tau.ppt)

Technology Transition

… and impacts to a facility like NERSC

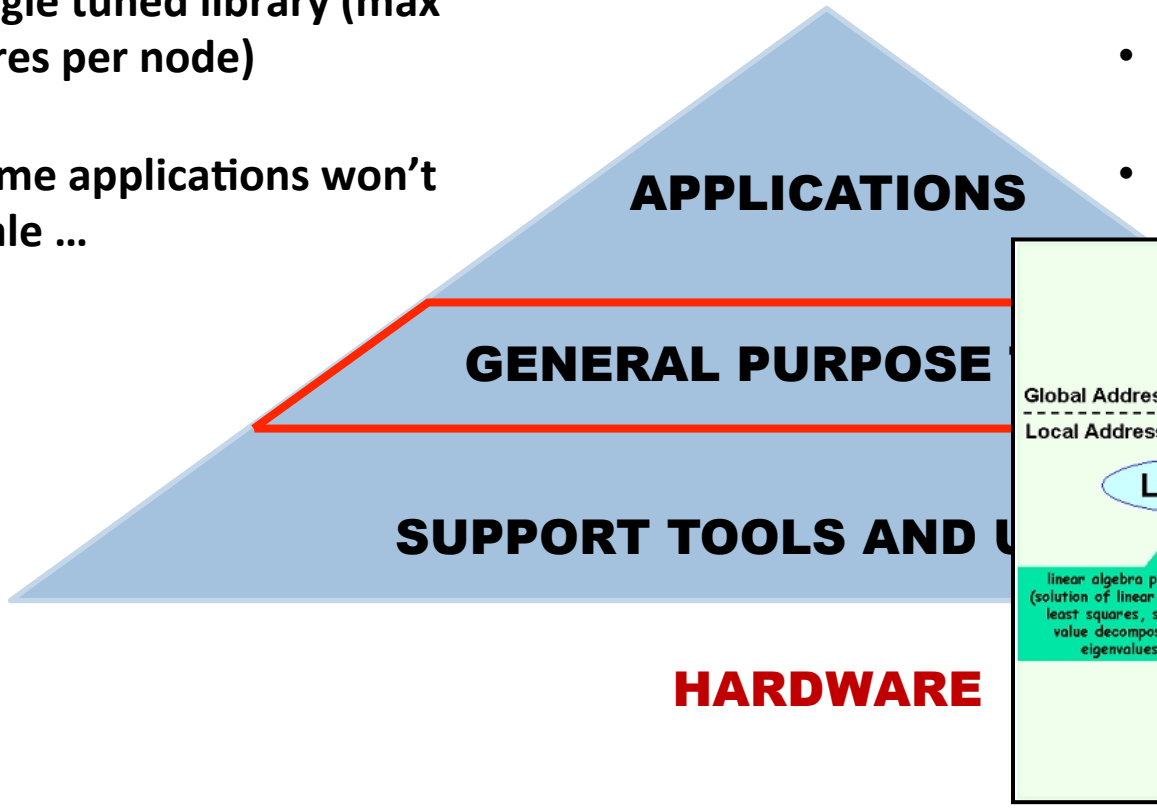


Source: Horst Simon & Kathy Yelick

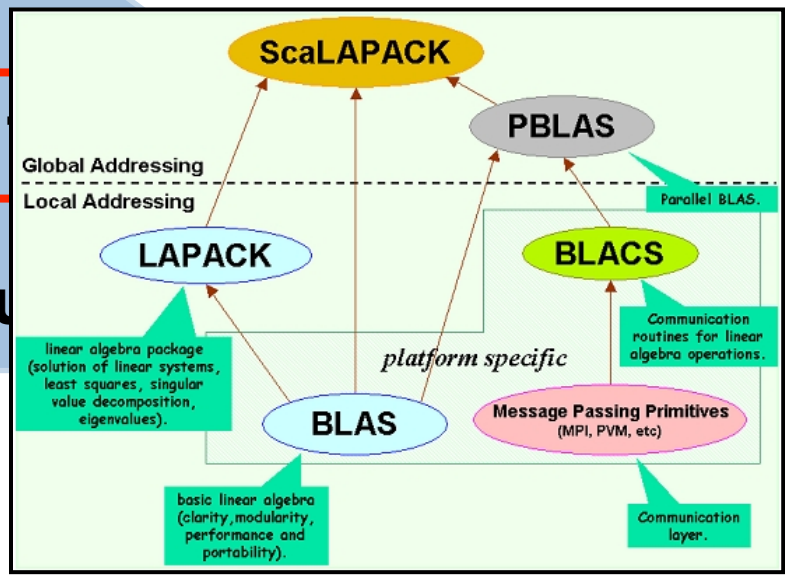
Parametric Research and Integration

- Hand-tuning algorithmic parameters can be cumbersome
- Auto-tuning produces a single tuned library (max cores per node)

- Some applications won't scale …



- Auto-tuned algorithmic parameters
- Auto-tuned libraries through steering parameters (#cores/ node)
- Run-time selection of tuned library
- Binary rewriting (?)



Software Tools for Application Development, Portability and Performance

- **min**[*time_to_first_solution*] (prototype)
- **min**[*time_to_solution*] (production)
- **min**[*software-development-cost*]
- **max**[*software_life*]
- **max**[*resource_utilization*]

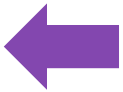
- Outlive Complexity
  - Increasingly sophisticated models
  - Model coupling
  - Interdisciplinary

  } *software evolution*
- Sustained Performance
  - Increasingly complex algorithms
  - Increasingly diverse architectures
  - Increasingly demanding applications

  } *long-term deliverables*

Partitioned Global Address Space (PGAS) Languages

- Abstract shared address space with control of locality for SPMD programming model
- Convenient for application development
- Languages
  - Unified Parallel C (UPC, *http://upc.lbl.gov*)
  - Co-Array Fortran (CAF, *http://www.co-array.org*)
  - Titanium (*http://titanium.cs.berkeley.edu*)
  - Fortress (*http://blogs.oracle.com/projectfortress*)
  - Chapel (*http://chapel.cray.com*)
  - X-10 (*http://x10-lang.org*)



- GASNET: low-level networking layer that provides high-performance communication primitives tailored for PGAS languages (*http://gasnet.cs.berkeley.edu*)

Co-array Fortran

- Image: each replication of a program
- Syntax:

```
real, dimension(10), codimension[*] : : x,y
x(:) = y(:)[q]    ! coarray y on image q is copied into coarray x on the executing image
```

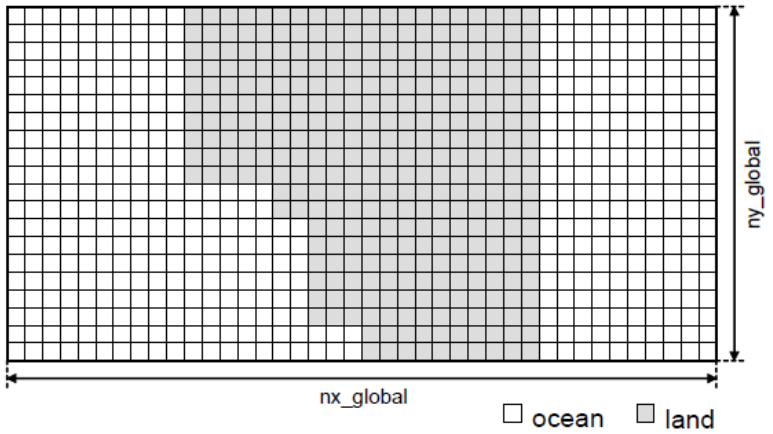
- Simple statements replace MPI calls (simpler, shorter, more maintainable code), no need to pack and unpack data (error prone)
- Synchronization

```
CRITICAL              ! Begin critical region
END CRITICAL          ! End of a critical region
SYNC ALL              ! Synchronize all images
SYNC IMAGES           ! Synchronize a specified subset of all images
SYNC MEMORY           ! Memory barrier
```

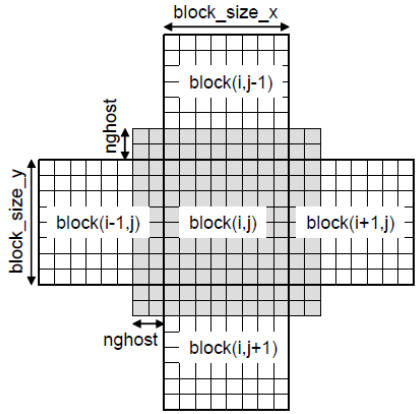
- See ftp.nag.co.uk/sc22wg5/N1801-N1850/N1824.pdf for details

Target Application: the Parallel Ocean Program (POP)

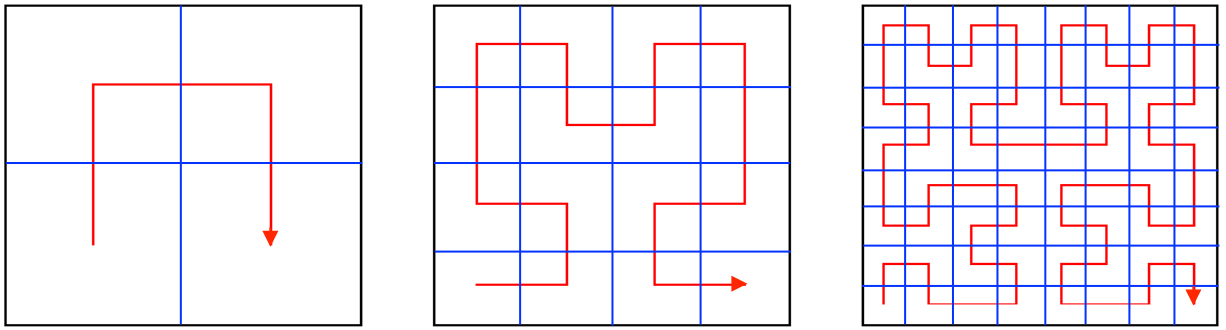
dipole or tripole grids are mapped into a 2D domain



full domain with shaded land cells (which are removed)



ghost cell or halo region



space-filling curve

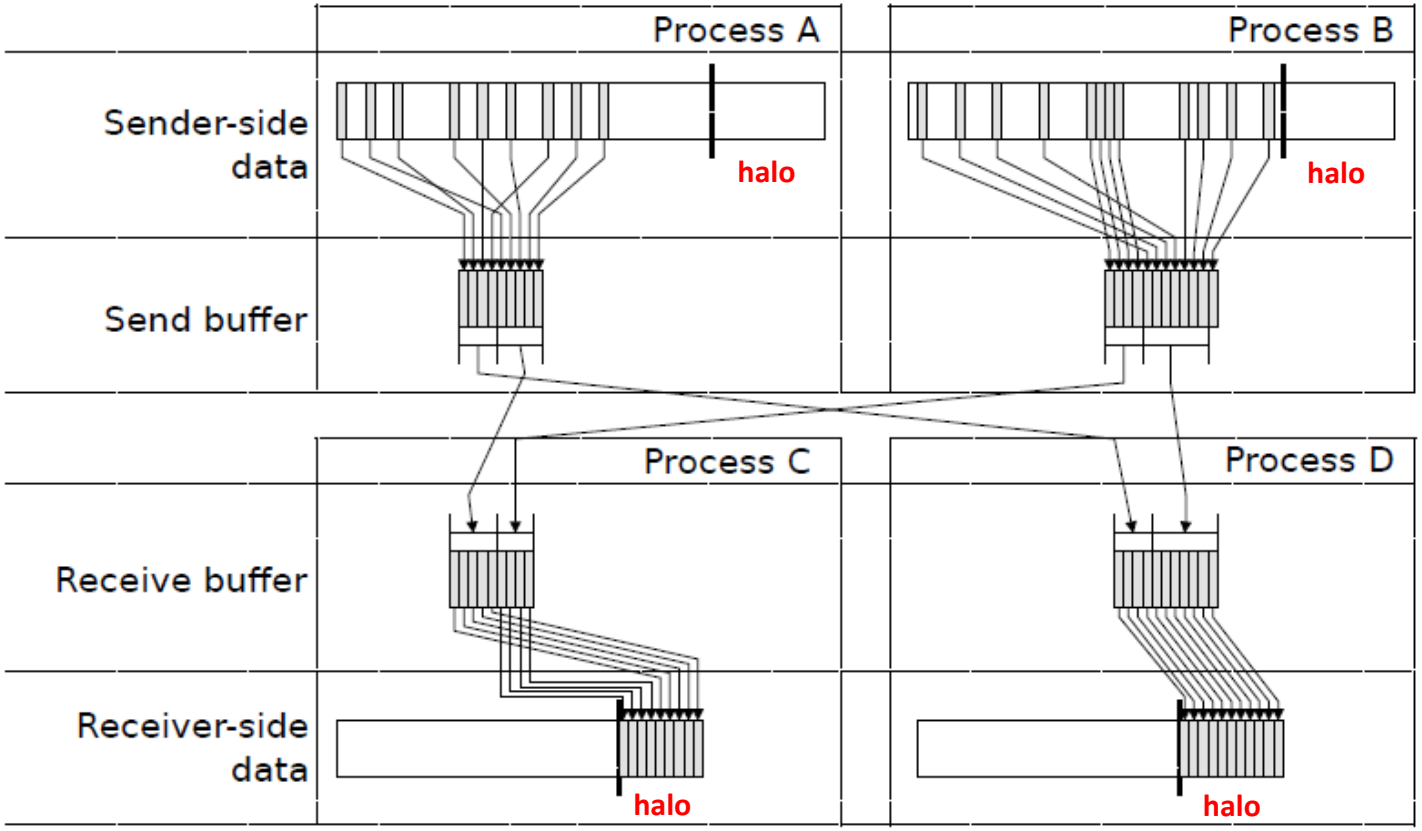
POP Components

- Baroclinic: density depends on the temperature and pressure, slow motion → finite differences
- Barotropic: density depends only on the pressure, fast motion → (preconditioned) conjugate gradient

- 1D data structure in the barotropic solver enables the elimination of land points in the 2D data structure
  - changes the matrix-vector multiply (indirect addressing)
  - uses CSR format

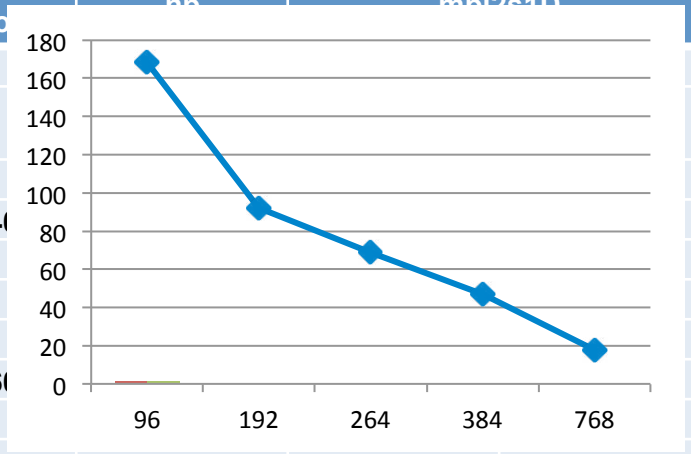
- Barotropic component as a miniapp
  - 13 versions (3 2D data, 10 1D data) by Stone, Dennis and Strout (CO State)
  - 9 point stencil
  - halo updates

Communication Pattern using 1D Structure



Performance Comparisons

hopper.nersc.gov: 153,216 compute cores Cray XE6 (6,384 compute nodes with 2 twelve-core AMD MagnyCours) and Gemini interconnect; 1º model; 120 CG iterations



grid size (per blo	np	mpi2s1D		caf1D				
				no sync images			sync images	
				MULTI_PULL	SINGLE_PUSH	SINGLE_PULL	SINGLE_PUSH	SINGLE_PULL
60x4				46.7	28.9	29.0	27.5	27.5
				22.2	12.6	11.5	10.9	10.7
				14.1	6.2	6.6	7.4	7.6
90x6				49.4	21.8	22.0	22.9	22.7
				28.3	11.3	11.8	12.6	12.8
120x80	96	168.6	168.0	211.4	170.0	170.4	171.3	171.3
	192	92.1	91.0	130.0	92.9	93.0	92.7	92.7
	264	68.6	69.0	109.4	70.5	70.6	70.3	70.2
	384	47.0	47.0	83.4	48.0	48.2	48.1	48.1
	768	17.8	18.0	57.9	19.4	19.6	20.9	19.8
180x120	96	193.2	191.5	261.0	193.4	193.4	194.6	194.6
	120	158.0	156.2	226.7	158.8	159.6	158.1	158.1
	192	108.4	108.7	176.2	110.9	111.2	109.8	109.7
	360	57.9	57.7	123.2	58.8	58.8	58.7	58.6

barrier a[k] = a a = a[k]

backup slides

ACTS Tools: numerical functionalities (1/3)

Computational Problem	Methodology	Algorithms	Library
Linear Equations	Direct Methods	*LU* factorization	ScaLAPACK (dense), SuperLU (sparse)
		Cholesky factorization	ScaLAPACK
		LDL^T factorization (tridiagonal matrices)	
		QR factorization	
		QR factorization with column pivoting	
		LQ factorization	
		Full orthogonal factorization	
		Generalized QR factorization	
	Iterative Methods	Conjugate gradient (CG)	AztecOO (Trilinos), PETSc
		GMRES	AztecOO, Hypre, PETSc
		CG Squared	AztecOO, PETSc
		Bi-CG-Stab	
		QMR	AztecOO
		Transpose free QMR	AztecOO, PETSc
		SYMMLQ	PETSc
		Richardson	
		Block Jacobi preconditioner	AztecOO, Hypre, PETSc
		Point Jacobi preconditioner	AztecOO
		Least-squares polynomials	
		SOR preconditioner	PETSc
		Overlapping additive Schwarz	
		Approximate inverse	Hypre
		Sparse LU preconditioner	AztecOO, Hypre, PETSc
		Incomplete LU (ILU) preconditioner	
	Multigrid	MG preconditioner	Hypre, PETSc
		Algebraic multigrid	ML (Trilinos), Hypre
		Semicoarsening	Hypre

ACTS Tools: numerical functionalities (2/3)

Computational Problem	Methodology	Algorithms	Library
Linear least squares	least squares	$min_x \|\| b-Ax \|\|_2$	ScaLAPACK
	minimum norm	$min_x \|\| x \|\|_2$	
	minimum norm least squares	$min_x \|\| x \|\|_2$ and $min_x \|\| b-Ax \|\|_2$	
Standard eigenvalue problems	iterative, direct	$Az=\lambda z$ for $A=A^T$ or $A=A^H$	ScaLAPACK (dense), SLEPc (sparse)
Generalized eigenvalue problems		$Az=\lambda Bz, ABz=\lambda z, BAz=\lambda z$	
Singular value decomposition		$A=U\Sigma V^T, A=U\Sigma V^H$	
Non-linear equations problems	Newton-based	Line search	PETSc, KINSOL (SUNDIALS)
		Trust regions	PETSc
		Pseudotransient continuation	PETSc
		Matrix-free	PETSc
Nonlinear optimization	Newton-based	Newton	OPT++, TAO
		Finite differences	OPT++
		Quasi-Newton	OPT++, TAO (LMVM)
		Nonlinear interior point	OPT++, TAO
	CG	Standard nonlinear CG	OPT++, TAO
		Limited memory BFGS	OPT++
		Gradient projection	TAO
	Direct Search	Without derivative information	OPT++
	Semismooth	Infeasible semismooth	TAO
		Feasible semismooth	

ACTS Tools: numerical functionalities (3/3)

Computational Problem	Methodology	Algorithms	Library
ODEs	Integration	Variable coefficient Adams-Moulton	CVODE (SUNDIALS)
	Backward differential	Direct and iterative solvers	
ODEs with sensitivity analysis	Integration	Variable coefficient Adams-Moulton	
	Backward differential	Direct and iterative solvers	
Differential-algebraic equations	Backward differential formula	Direct and iterative solvers	IDA (SUNDIALS)
Nonlinear equations with sensitivity analysis	Inexact Newton	line search	SensKINSOL (SUNDIALS)
Tuning and optimization	Automatic code generator	BLAS and some LAPACK routines	ATLAS

POP: block sizes for a 1° model (3600x2400 grid points)

blocks (each direction)	grid points (3600x2400)		points per block	halo
20	180	120	21600	604
30	120	80	9600	404
40	90	60	5400	304
60	60	40	2400	204
75	48	32	1536	164
100	36	24	864	124
150	24	16	384	84
200	18	12	216	64

Miniapps for Testing PGAS Languages

- Miniapps (as defined by M. Heroux et al., 2009)
  - Benchmarking in immature environments
  - Simple build process to enable easy porting
  - About 1000 SLOC
  - Performance proxy
  - Programmability proxy (code importance, incremental improvement)

POP Miniapp: Future Work

- Analysis of the communication pattern
- CAF based on GASNET on hopper
- CAF 2.0 (*http://caf.rice.edu*)
  - subsets known as teams, collective communication, and relative indexing of process images for pair-wise operations
  - topologies, which augment teams with a logical communication structure
  - global pointers in support of dynamic data structures, and
  - enhanced support for synchronization for fine control over program execution
  - asynchronous communication support for hiding communication latency