
High-Performance Parallel Octave on Itanium using ParaM

Aakash Dalwani
Neil Ludban
David E. Hudak, Ph.D.
Ashok Krishnamurthy, Ph.D.

Overview

- DARPA High-Productivity Computing Systems (HPCS) Overview
 - HPC Challenge Benchmarks
 - Scalable Synthetic Compact Apps
 - GNU Octave
- High-Performance Technical Computing (HPTC) Programming Environment
- ParaM v1.0 (bcMPI) Software
- Parallel Octave and OSC Itanium cluster

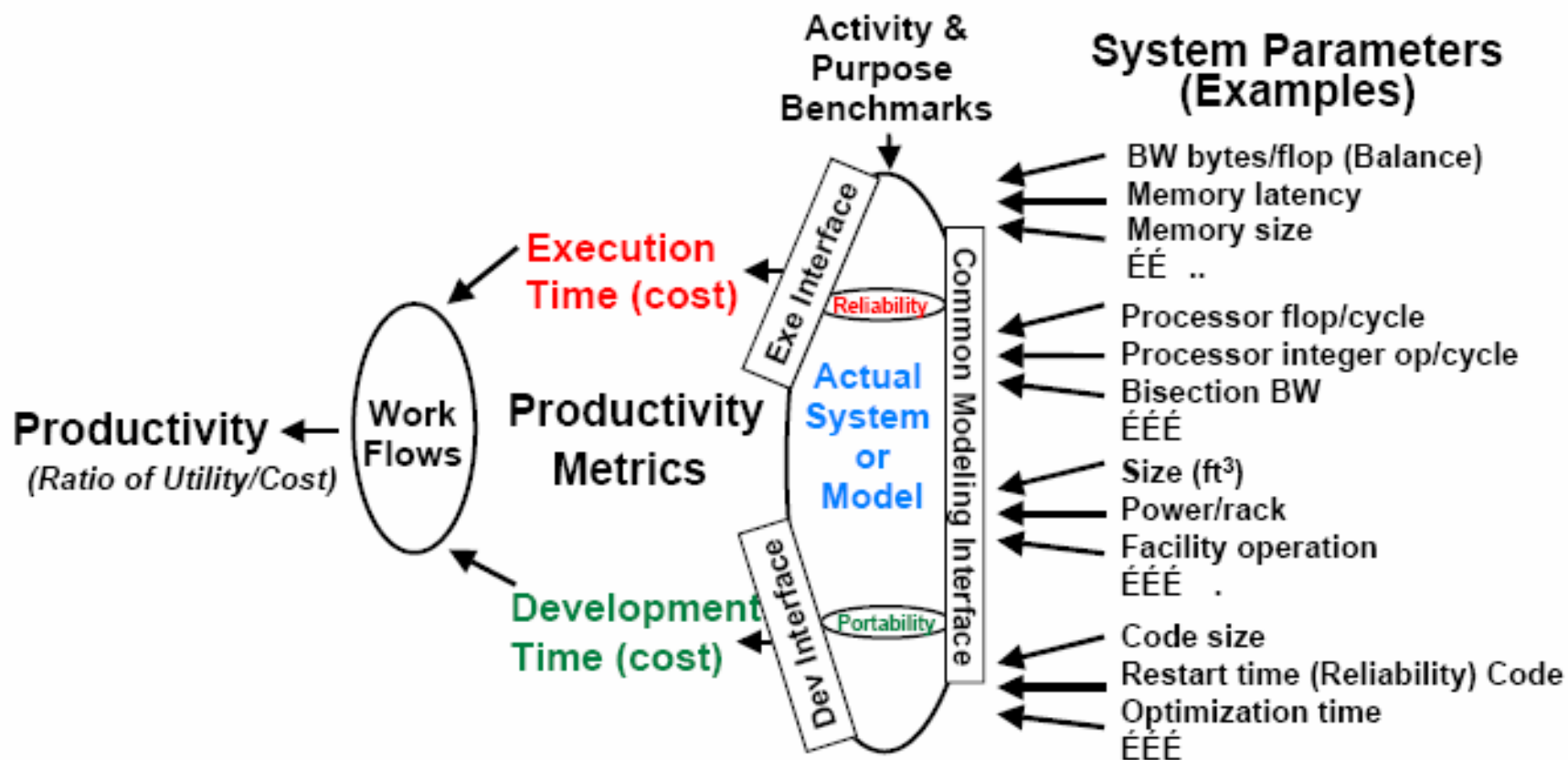
OSC

Innovations in computing,
networking, and education

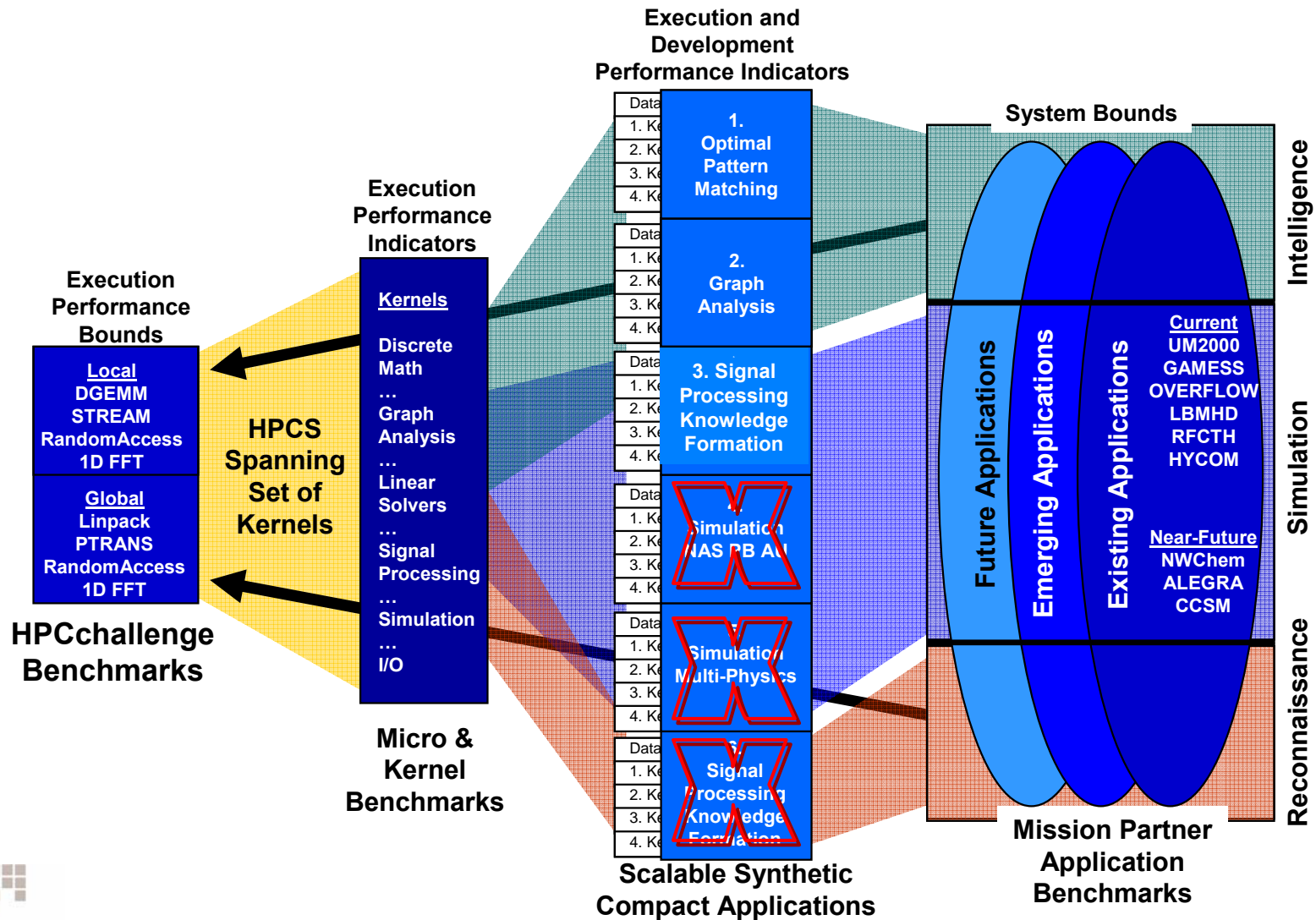
DARPA HPCS Overview

- Vision: Focus on the Lost Dimension of HPC - "User & System Efficiency and Productivity"
- Three vendors in Phase II (ending in June 2006):
 - Sun, IBM and Cray
- Productivity team: Develop tools and methodology for measuring productivity

Productivity Assessment Framework



HPCS Benchmark Spectrum

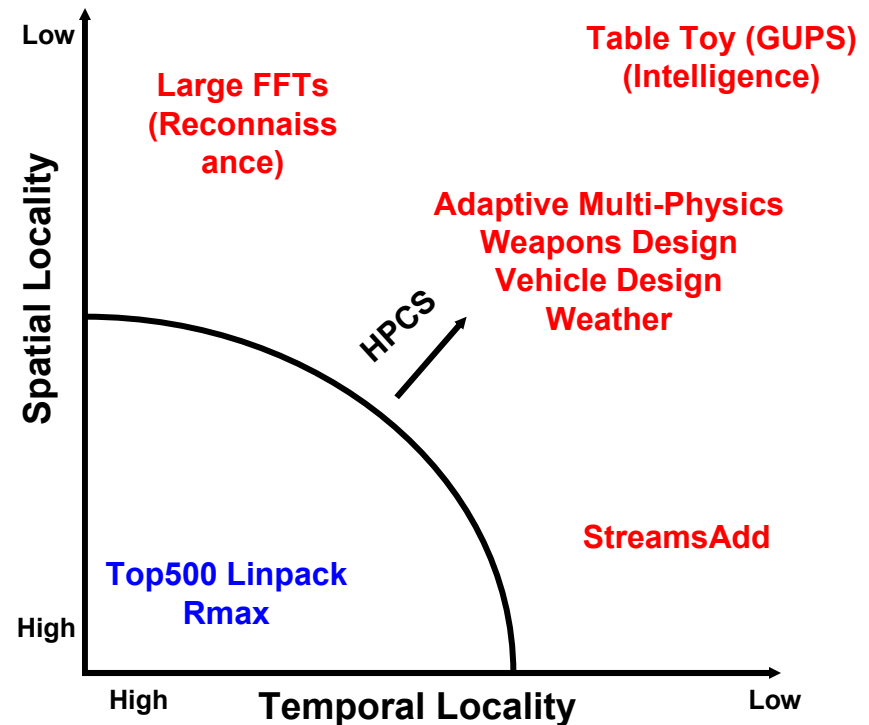


Why Octave/MATLAB ?

- **The HPCS Productivity team is developing both written and executable specifications for the Synthetic Scalable Compact Applications (SSCAs).**
- **Cost (in time and money) of creating the SSCAs was a key criterion.**
- **The decision was to create the executable specification in Octave and MATLAB.**
- **Octave provides the ability to port SSCAs to new platforms.**

HPC Challenge Benchmarks

- <http://icl.cs.utk.edu/hpcc/>
- **Benchmarks:**
 - HPL
 - DGEMM
 - STREAM
 - PTRANS
 - RandomAccess
 - FFTE
 - Comm. bandwidth & latency



HPC Challenge 2005 Results

- **G-FFT [Gflop/s]**
 1. **IBM BG/L 2311 (NNSA)**
 2. **IBM BG/L 1112 (Watson)**
 3. **IBM Power5 966 (LLNL)**

- **G-RandomAccess [GUPS]**
 1. **IBM BG/L 35.47 (NNSA)**
 2. **IBM BG/L 17.29 (Watson)**
 3. **Cray X1E 7.69 (ORNL)**

HPCS goal: 64,000

- **G-HPL [Tflop/s]**
 1. **IBM BG/L 259 (NNSA)**
 2. **IBM BG/L 67 (Watson)**
 3. **IBM Power5 58 (LLNL)**

HPCS goal: 2,000

- **G-STREAM [TB/s]**
 1. **IBM BG/L 160 (NNSA)**
 2. **IBM Power5 55 (LLNL)**
 3. **IBM BG/L 40 (Watson)**

HPCS goal: 6,500

Systems from the latest TOP500 (Nov 2005)
sorted by rank:
#1, #2, #3, #4, #10, #14, #17, #35, #37, #71, #80

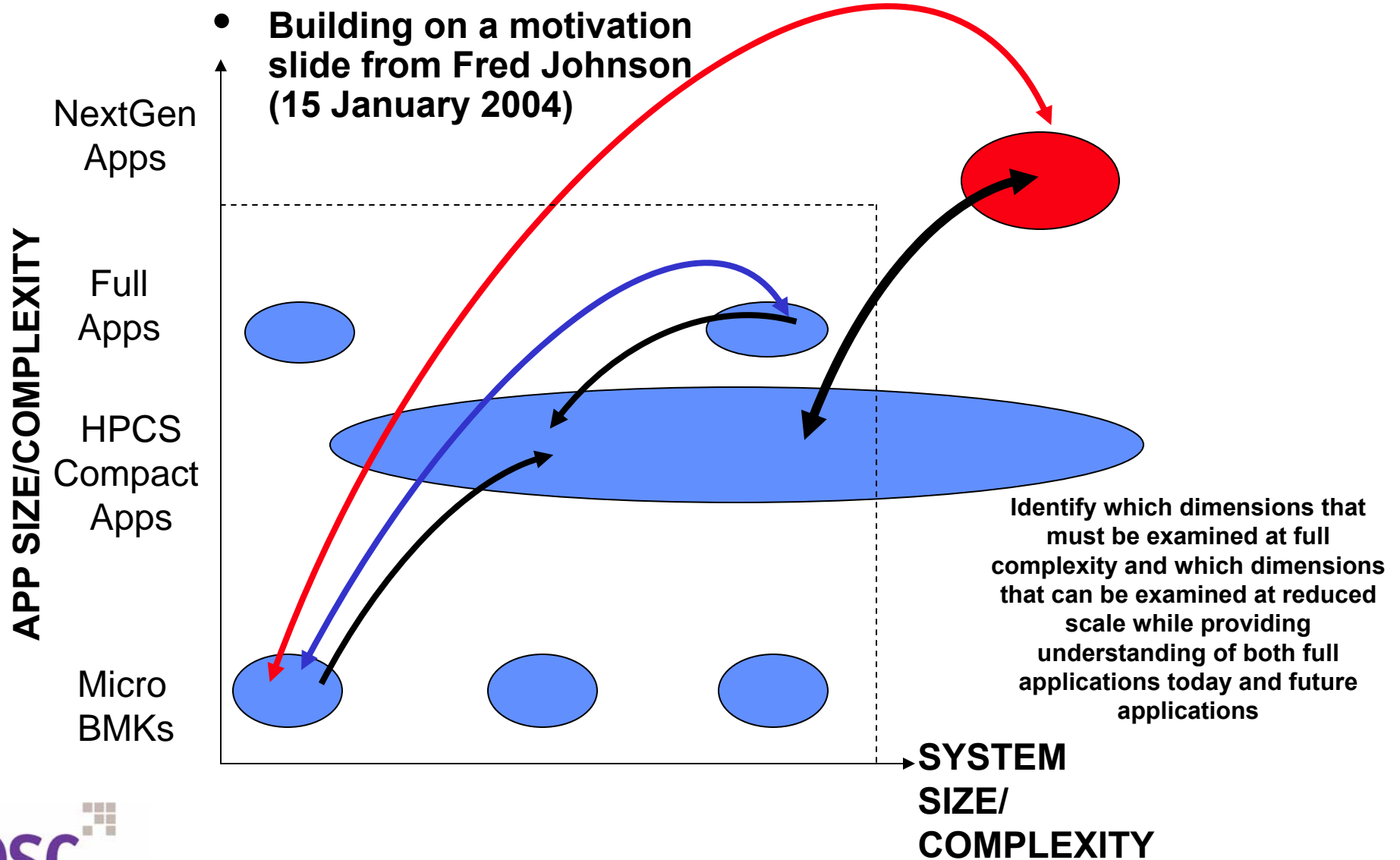
HPCC Awards BOF at SC|05

- **Class 1: Best Performance**
 - 4 awards go to IBM BG/L at NNSA
- **Class 2: Most Productivity**
 - Cray: C + MTA pragmas
 - IBM: UPC
 - Finalists:
 - Cray: UPC
 - Mathworks: Parallel Matlab
 - MIT: Cilk
- **Class 1 encouraged more submission to the HPCC database**
 - Submitted world's largest computer installation
- **Class 2 showcased less popular programming languages with working implementations**
 - Total of 10 competing submissions:
 - C with vendor libraries
 - C with vendor pragmas
 - C++ and OpenMP
 - Cilk
 - HPF
 - Parallel Matlab: Mathworks, StarP
 - UPC (3 times)

Synthetic Scalable Compact Applications: The Vision

- **To bridge the gap between scalable synthetic kernel benchmarks and (non-scalable) real applications — an important future benchmarking tool**
- **Must be representative of actual workloads within an application while not being numerically rigorous**
 - memory access characteristics
 - communications characteristics
 - I/O characteristics
 - etc.
- **Will have no limits on the distribution to vendors and universities**
- **Scalable Synthetic Compact Applications (SSCAs) will try to represent the wide spectrum of potential HPCS Mission Partner applications**

SSCAs: The Goal



SSCA #1: Bioinformatics

Intent

- **To develop a scalable synthetic compact application that has multiple analysis techniques (multiple kernels) identifying similarities between sequences of symbols**
 - Symbols can be identical, closely related, or entirely different
 - A symbol in one sequence can match a gap in another
- **Each of the five kernels is based on an application from bioinformatics, including**
 - Local alignment
 - Searching for similarities
 - Global alignment
 - Multiple alignment
- **Each kernel operates on either the original sequences, the results of the previous kernel, or both**
- **To be entirely integer and character based**
 - Except for incidental statistics

SSCA #2: Graph Analysis

Intent

- **To develop a scalable synthetic compact application that has multiple analysis techniques (multiple kernels) accessing a single data structure representing a directed asymmetric weighted multigraph with no self loops**
- **In addition to a kernel to construct the graph from the input tuple list, there will be three computational kernels to operate on the graph**
- **Each of the kernels will require irregular access to the graph's data structures**
- **No single data layout will be optimal for all three computational kernels**
- **To be entirely integer and character based**
 - **Except for statistics**

SSCA #3: Sensor Processing, Knowledge Formation and Data I/O

Intent

- **SSCA #3 Focuses on two stages:**
 - Front end image processing and storage
 - Back end image retrieval and knowledge formation
- **Two stages is representative of many areas:**
 - **Medical imaging (e.g.: tumor growth)**
 - Image many patients daily
 - Later compare images of same patient over time
 - **Astronomical image processing (e.g.: monitor supernovae)**
 - Image many regions of the sky daily
 - Later compare images of a region over time
 - **Reconnaissance monitoring (e.g.: enemy movement)**
 - Image many areas daily
 - Later compare images of a given region over time
- **Benchmark has a significant data I/O component**

Benchmarks Implementations

Language	HPC Challenge				Scalable Synthetic Compact Applications (SSCAs)		
	HPL	RandomAccess	STREAM	FFT	Bioinformatics (SSCA#1)	Graph Theory (SSCA#2)	Sensor and IO (SSCA#3)
Written Spec	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)	0.5 (LL)	1.0 (LL)	0.8 (LL)
C					0.5k1 (PSC)		0.5 (ISI)
C + MPI	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)	0.5k1 (PSC)		
C + MPI + OpenMP	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)	1.0 (UTK)			
C + OpenMP						Eugene Loh, Sun (Eugene.Loh@sun.com)	
UPC		1.0 (ISI)			0.5k1 (PSC)	1.0 (GT/UNM)	
C + pthreads					0.5 (GT/UNM)	1.0 (GT/UNM)	
C++							1.0 (LL/MITRE/CodeS)
Fortran							0.5io (LMCO)
Matlab	1.0 (LL)	1.0 (LL)	1.0 (LL)	1.0 (LL)	0.5 (LL)	1.0 (LL)	0.8 (LL)
MatlabMPI						1.0 (LL)	0.8 (LL)
Matlab + mexGA	OSC	OSC	OSC	OSC	OSC	OSC	OSC
StarP	1.0 (UCSB)	1.0 (UCSB)	1.0 (UCSB)	1.0 (UCSB)	1.0int (UCSB)		0.5 (UCSB)
pMatlab	1.0 (LL)	1.0 (LL)	1.0 (LL)	1.0 (LL)	1.0 (LL)		1.0 (LL)
Octave	1.0 (OSC)	1.0 (OSC)	1.0 (OSC)	1.0 (OSC)	0.8 (OSC)	1.0 (UW)	0.8 (OSC)
Octave+mexGA	OSC	OSC	OSC	OSC	OSC	OSC	OSC
Python							
Python+MPI		1.0 (UTK)	1.0 (UTK)				
Java					0.5k1 (PSC)	1.0int (GT/Sun)	
Chapel	1.0 (Cray)	1.0 (Cray)	1.0 (Cray)	1.0 (Cray)	0.5 (Cray)	1.0int (Cray)	
X10		1.0 (IBM)			0.5k1 (PSC)	1.0 (UNM)	
Fortress							
Black = released Blue = completed Green = in progress	LL = MIT Lincoln Labs UTK = Univ. of Tennessee, Knoxville OSC = Ohio Supercomputer Center UW = University of Wisconsin GT = Georgia Tech UNM = University of New Mexico UCSB = Univ. of California, Santa Barbara PSC = Pittsburgh Supercomputer Center LMCO = Lockheed Martin CodeS = ??? ISI = Univ. of Southern California, Information Sciences Institute						

<http://www.highproductivity.org/060104-BenchmarkStatus.htm>

High Performance Technical Computing (HPTC)

- Efficiency
 - ...otherwise, programmers will stick with a traditional programming model
- Productivity
 - Reduce time to **first solution**
 - Rapid prototyping
 - Reduce time to **correct solution**
 - Must identify and isolate bugs
 - Requires “execution transparency”
 - Reduce time to **best solution**
 - Optimized for given machine
 - Requires “performance transparency”

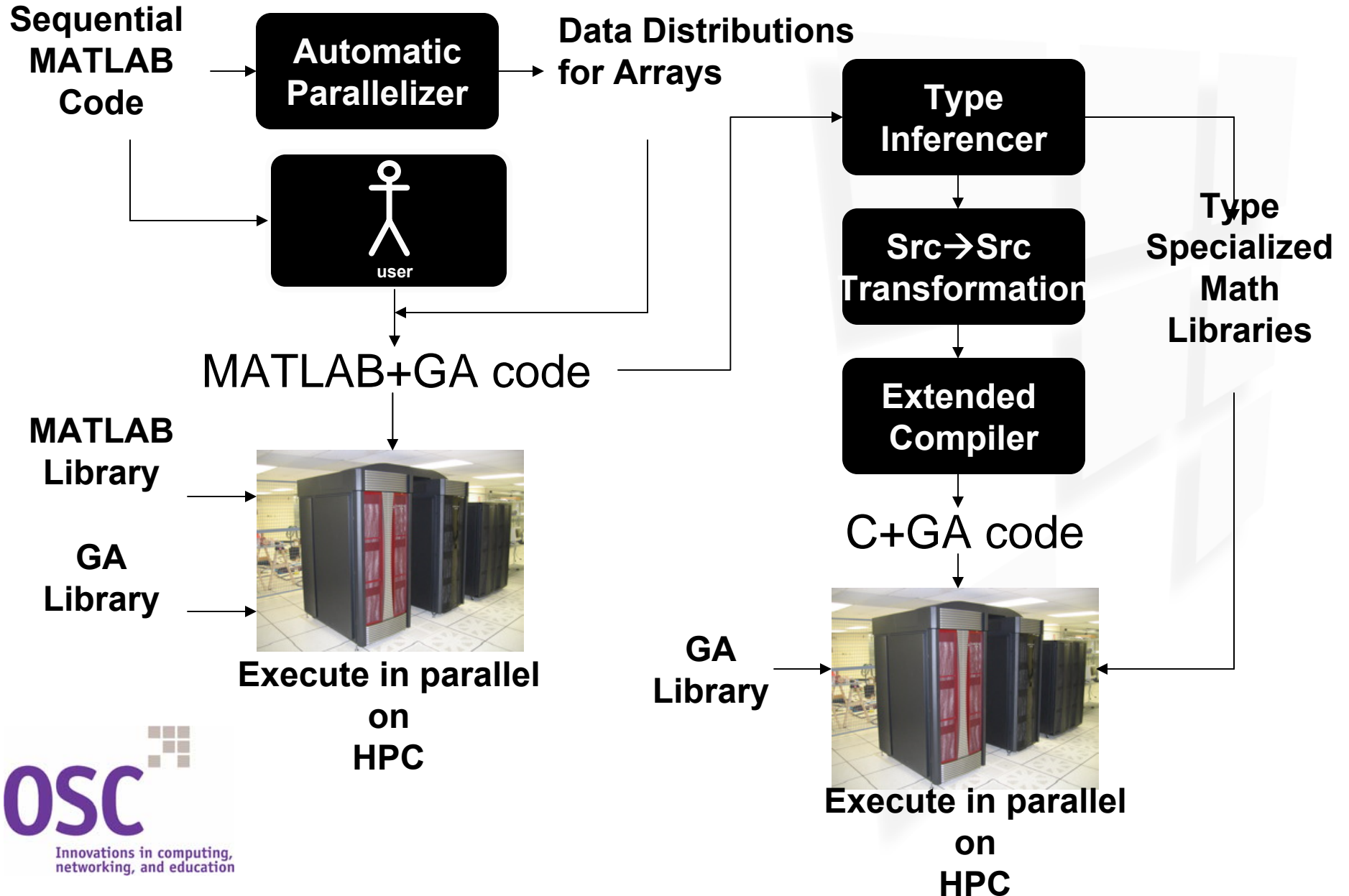
HPTC Programming Environment Choices

- Numeric/symbolic interpreters
 - MATLAB, GNU Octave, MAPLE, Mathematica
 - Optimized for technical computing domain
 - Good numeric library support: linear algebra, FFTs, ...
 - Widely used for prototyping
 - Interprocess communication and parallel execution support needed

Overview of ParaM

- **Overall objective:** Enable compiled execution of MATLAB/Octave programs on clusters and parallel systems, *without* requiring low-level communication mechanisms, such as explicit message-passing
- **Approach:**
 - Multifaceted
 - Explicitly Parallel Shared-Space Model for MATLAB/Octave
 - Data Parallel MATLAB/Octave
 - Intelligent Compilation of MATLAB to C
 - Automatic Data Distribution
 - Provide useful intermediate products

ParaM Overview



ParaM: Software Tools

- Parallel software tools
 - Goals: extensible, portable, scalable
 - Leverage existing code
 - Parallel execution libraries
 - Message Passing: MPICH, MPICH2, MVAPICH, OpenMPI
 - Partitioned Global Address Space: Global Arrays
 - Numeric/Symbolic Engines
 - GNU Octave or MATLAB
 - ParaM v1.0 (aka bcMPI)

■ Provides MPI interface to Octave and
MATLAB environments

MATLAB+MPI: Why another?

- Complementary projects fall into three categories:
 - MATLABMPI (MIT Lincoln Labs)
 - Interactive Supercomputing's StarP
 - MathWorks' Distributed Computing Toolbox and Engine (DCT/DCE)
 - Other projects (e.g., MPITB)

MATLAB+MPI: Why another?

- Complementary projects fall into three categories:
 - MATLABMPI (MIT Lincoln Labs)
 - MathWorks' Distributed Computing Toolbox and Engine (DCT/DCE)
 - Other projects (e.g., MPITB, MultiMATLAB, ...)
 - Limited data type support
 - Typically, real and complex arrays
 - Limited portability
 - One version of interpreter, one version of one MPI implementation bolted together...
 - Cannot substitute your preferred MPI implementation or your preferred version of the interpreter without hacking their Makefiles

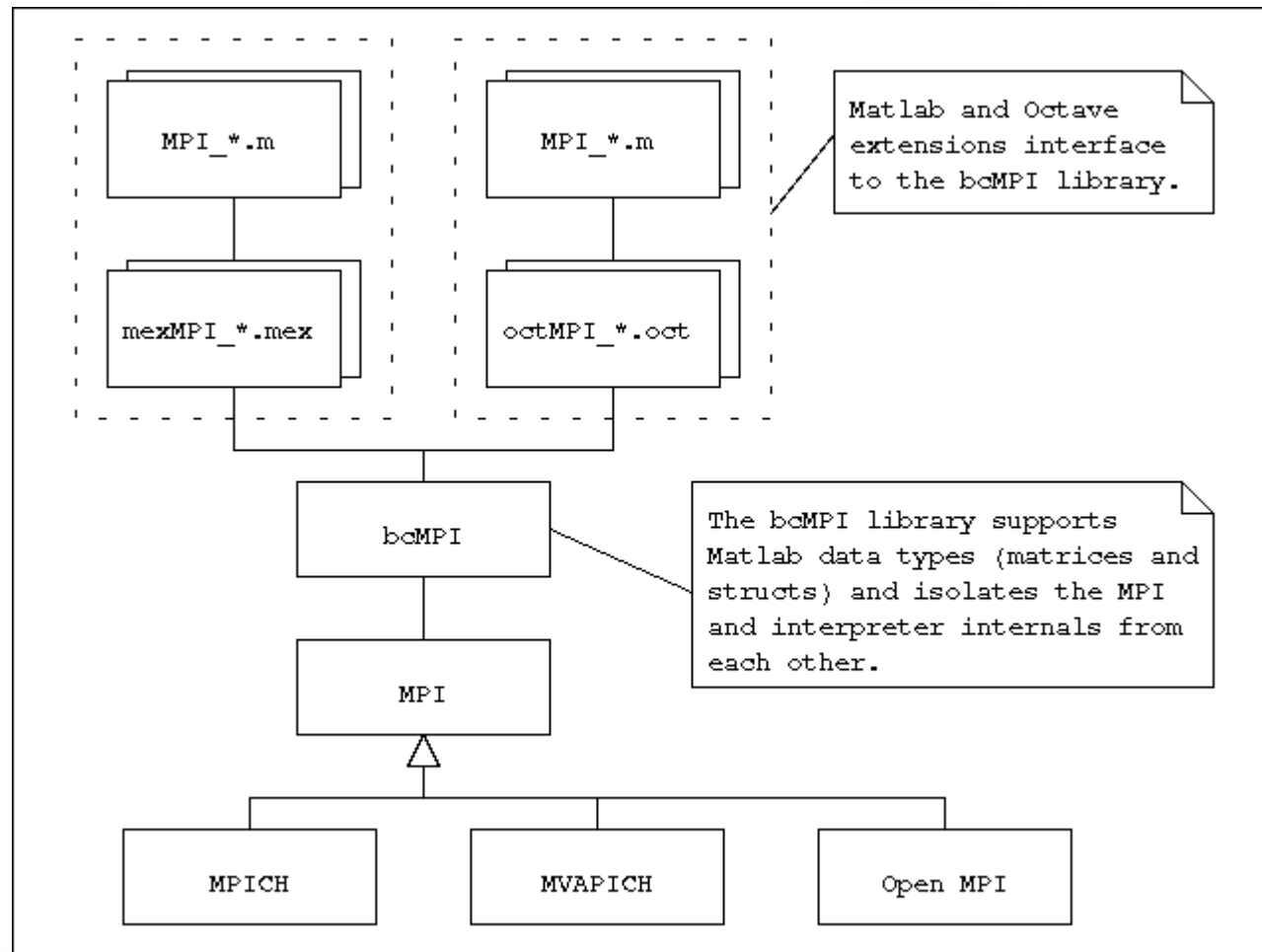
bcMPI Features

- Completeness
 - API “reasonably compatible” with MATLABMPI
 - Our MPI tags are numeric, theirs alphanumeric
 - bcMPI includes broadcast, barrier, reduce operations
 - MPI_Bcast, MPI_Barrier, MPI_Op_create, MPI_Reduce
 - bcMPI provides convenience functions
 - MPI_Initialized, MPI_Finalized, MPI_Wtime, MPI_Wtick
 - Supports many MATLAB data types
 - cell arrays, structs, strings
 - MATLAB-style help for commands

bcMPI Features

- Integration with standard HPC environments and tools
 - Job control, debugging, tracing
- Supports MATLAB or Octave
- Supports mpich and OpenMPI
 - Design build system to be easily extensible for other MPI libraries (e.g., mpich-gm for Myrinet)

bcMPI: Software Architecture



OSC Itanium 2 Linux Cluster



A distributed/shared memory hybrid of commodity systems based on the Intel IA64 (Itanium 2) architecture

516 processors for serial and distributed parallel computing

80 processors for shared memory computing

OSC Itanium 2 Linux Cluster Specifications

- Processors
 - Type Intel Itanium 2
 - Number of processors 516 (plus 16)
 - Clock Speed (296) 900MHz, (220) 1.3GHz
 - Local memory per node
 - Serial nodes 12 Gbytes
 - Parallel nodes 4 Gbytes
 - Peak per processor 3.6 Gflops, 5.2 Gflops
- Memory
 - Technology SDRAM
 - Architecture distributed/shared
 - Total system memory 1.2 Tbytes
- Interconnect Network
 - Technologies Myrinet, Gig-Ethernet
 - Topology Switch
 - Inter-PE communication rate ~6 Gb/second
- I/O
 - Peak I/O bandwidth 100 Mb/second
 - Attached disks 80 Gbytes per node
 - Communication interfaces TCP/IP, MPI over Myrinet
- Peak Performance ~2.2 Tflops



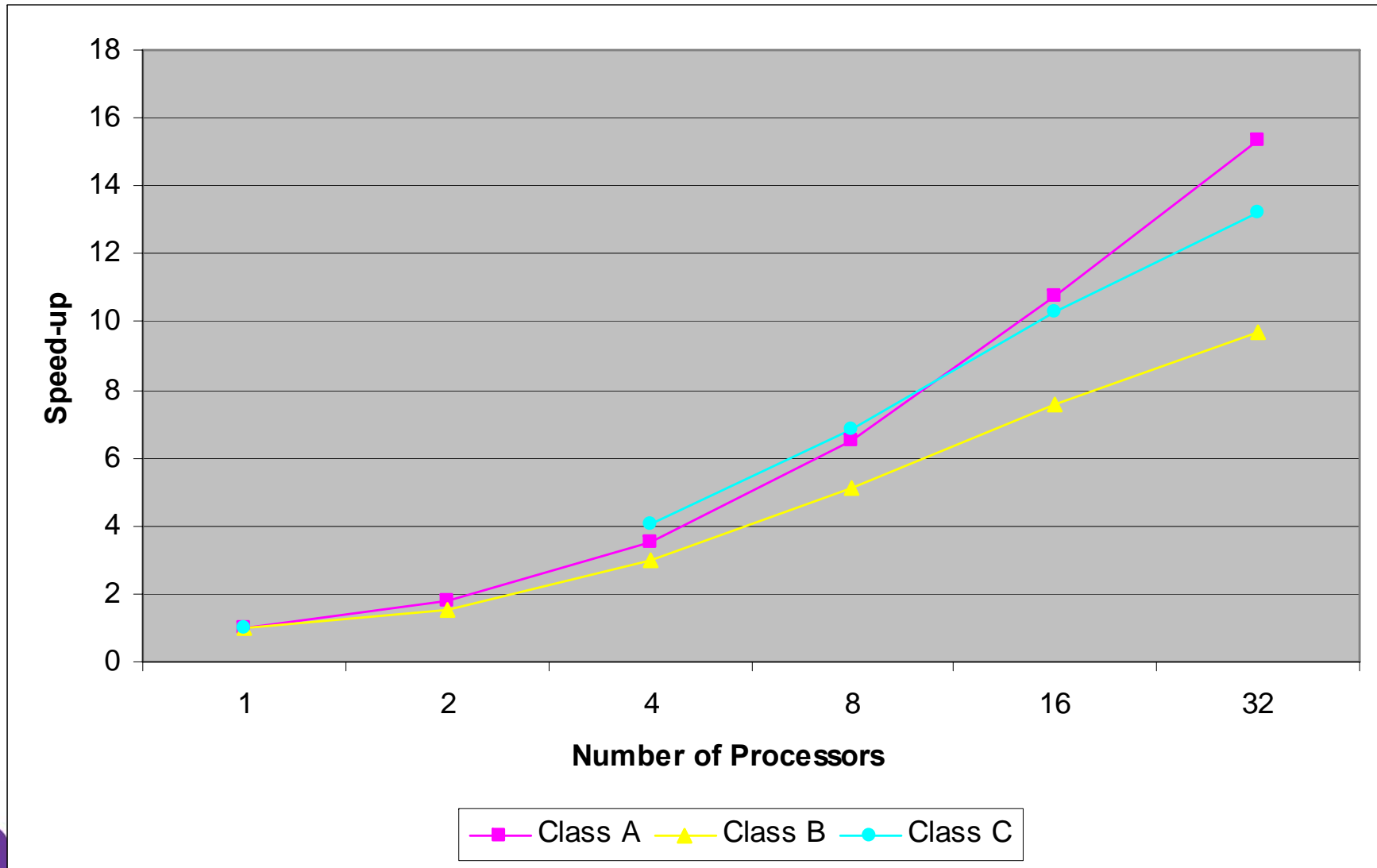
Innovations in computing,
networking, and education

Parallel Octave on OSC I2 Cluster

- Needed to build a good Octave for Itanium
 - Octave 2.9.4 provided Itanium and 64-bit support
 - bring in BLAS, LAPACK and FFTW
- Parallel support provided by bcMPI/mpich
- NAS FT benchmark tested



NAS FT Results



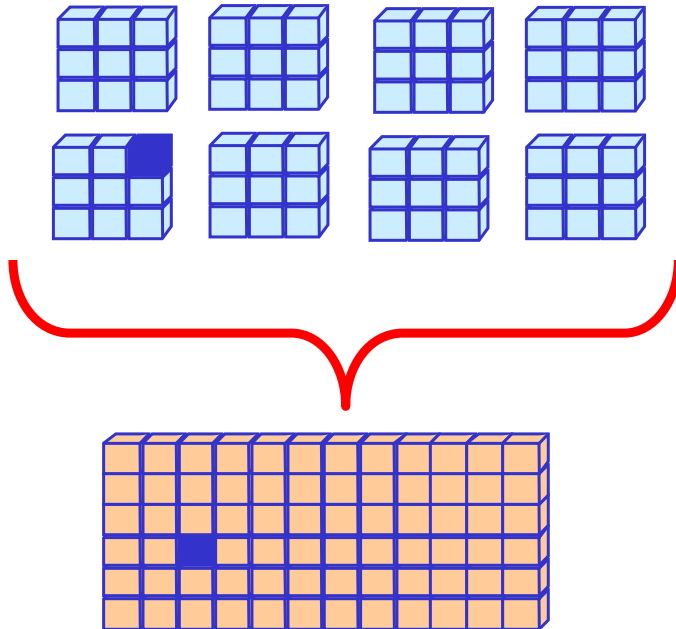
ParaM: Shared Space Model

- Uses Global Arrays (GA) shared-memory framework from PNNL
- Explicitly parallel, shared-memory model to write parallel MATLAB libraries and application programs (mexGA)
- Data parallel abstraction to be implemented over underlying explicitly parallel mexGA layer

Global Arrays

Distributed dense arrays that can be accessed through a shared memory-like style

Physically distributed data



single, shared data structure with global indexing

e.g., access $A(4,3)$ rather than $\text{buf}(7)$ on task 2

Remote Data Access in GA

Message Passing:

identify size and location of data blocks

loop over processors:

if (me = P_N) then

pack data in local message buffer

send block of data to message buffer on P0

else if (me = P0) then

receive block of data from P_N in message buffer

unpack data from message buffer to local buffer

endif

end loop

copy local data on P0 to local buffer

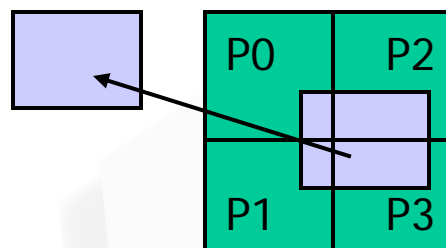
Global Arrays:

`GA_Get(g_a, lo, hi, buffer, Id);`

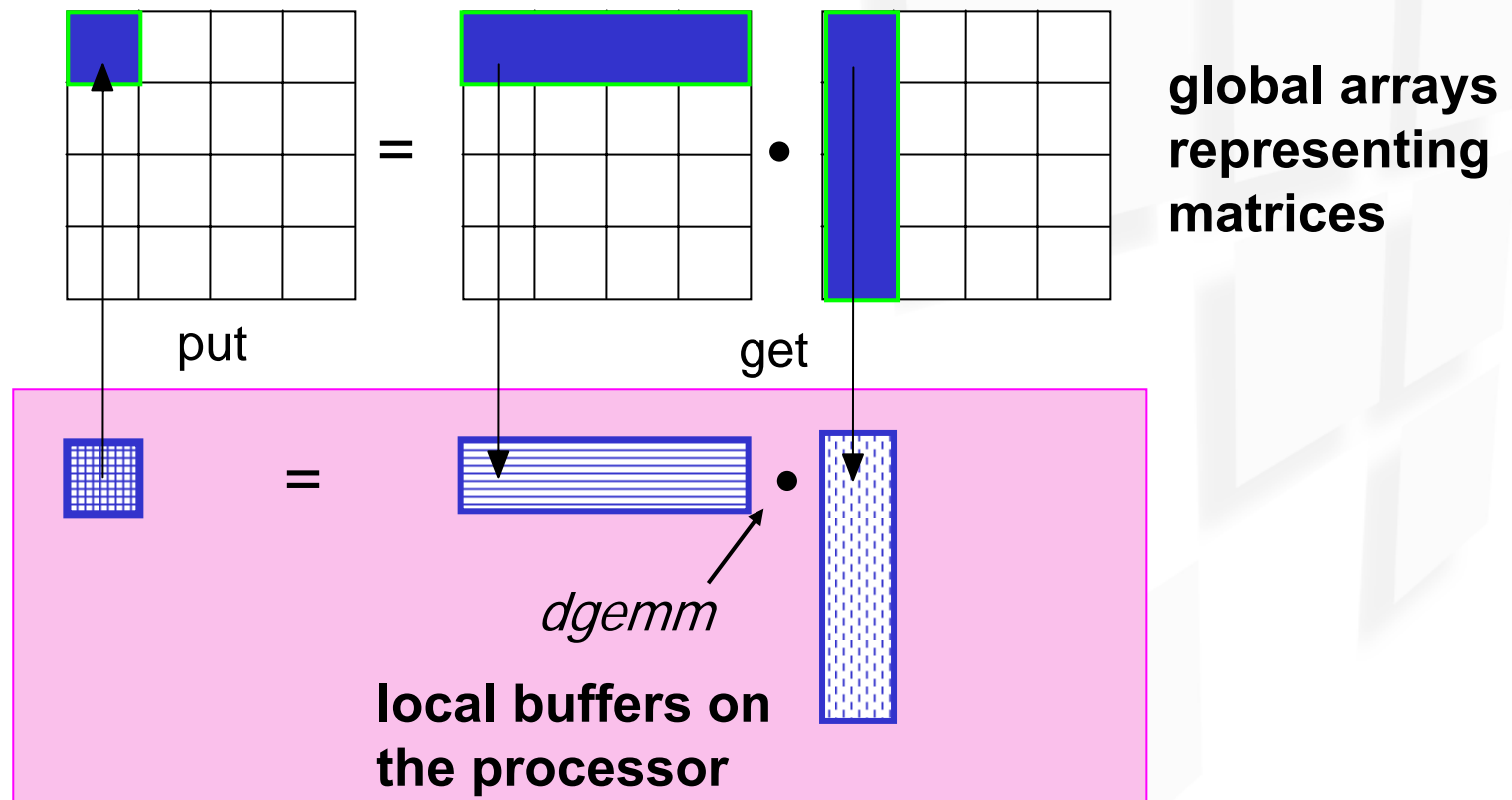
Global Array handle

Global upper and lower indices of data patch

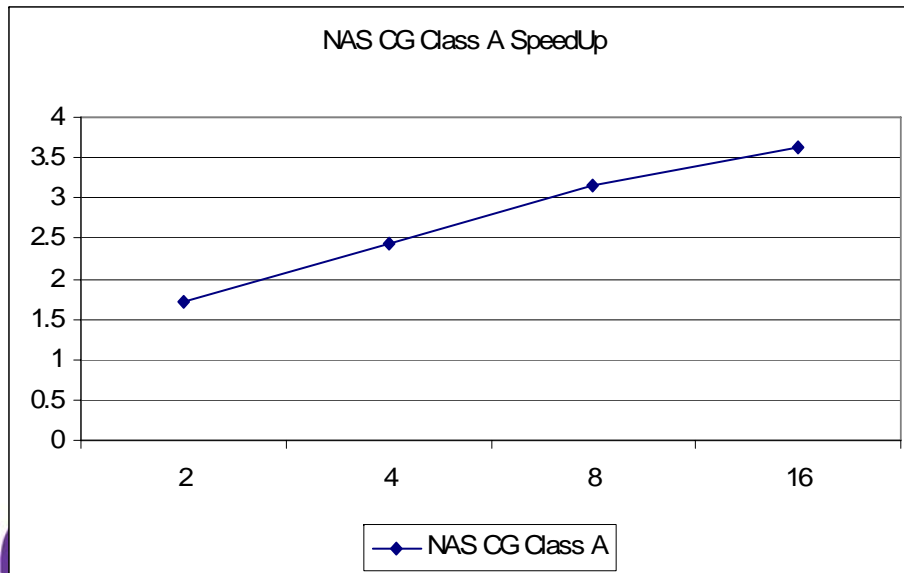
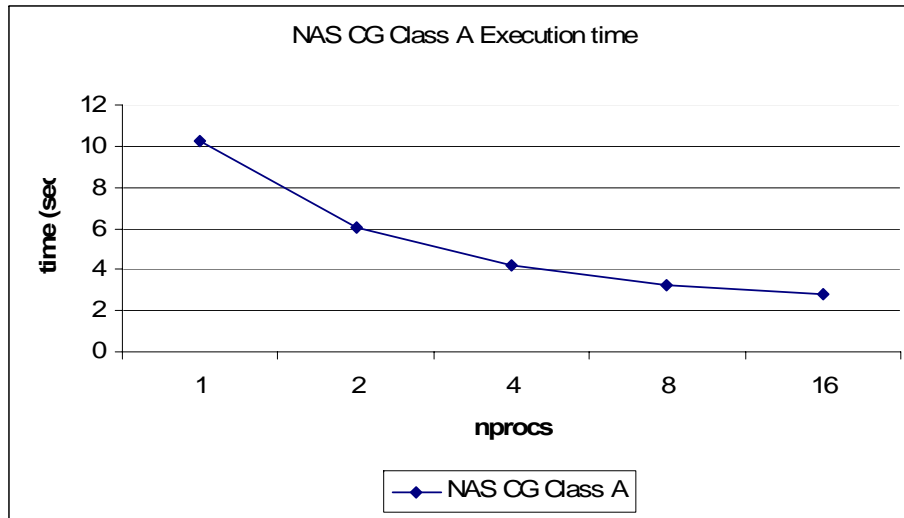
Local buffer and array of strides



Example: Matrix Multiply



NAS kernel CG in MATLAB-mexGA

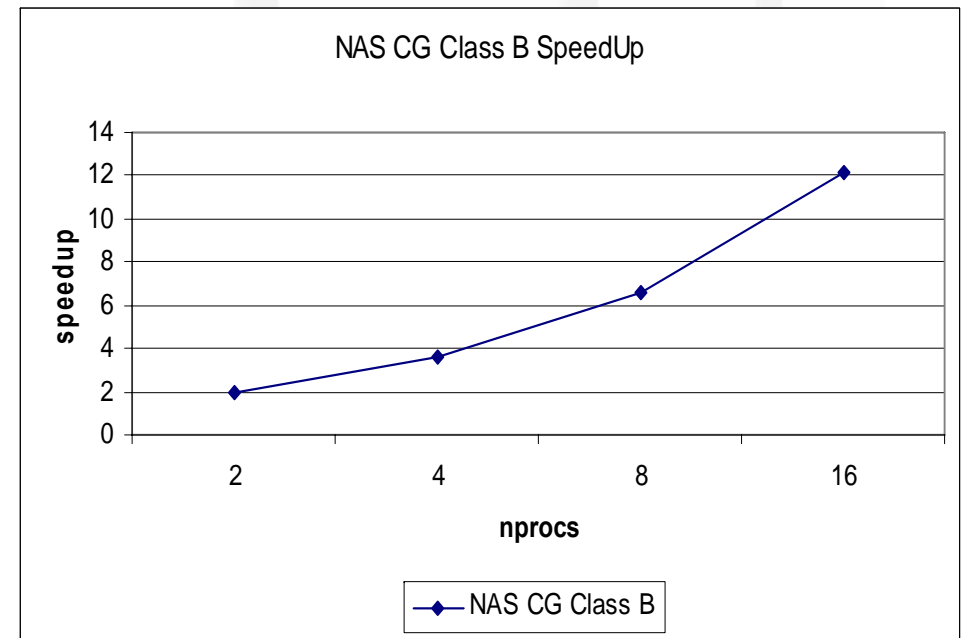
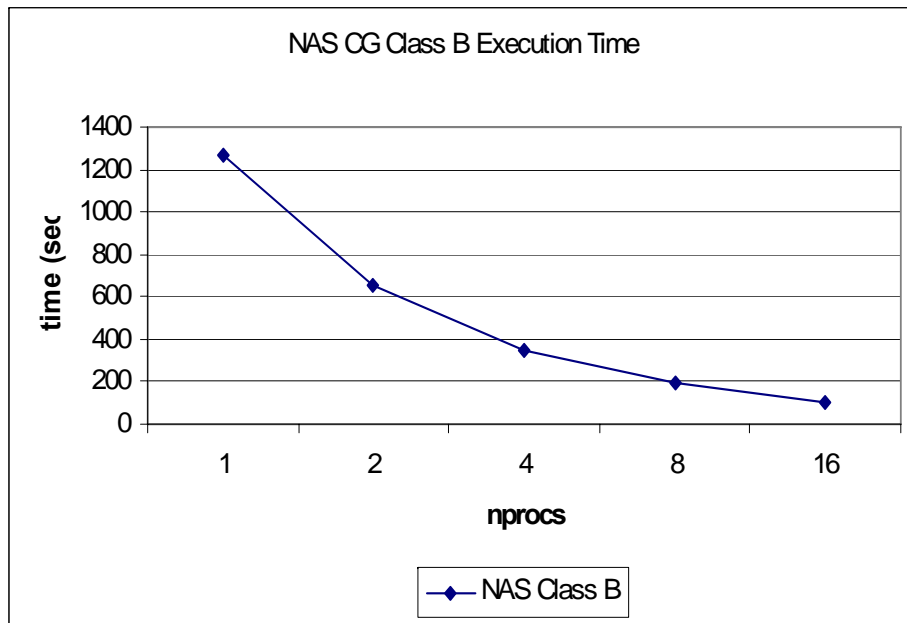


SLOC

Fortran	MATLAB
MPI	mexGA
~1000	161



MATLAB-mexGA NAS CG: Class B



ParaM: Next Steps

- bcMPI
 - HPC tool integration - supporting tracing from interpreter interface
 - Distribution of software
- ParaM
 - Complete, distribute mexGA code
 - Higher level of parallel abstraction
 - Global view compilation
 - Compilation of MATLAB code for higher single-processor performance



Conclusions

- MATLAB-style languages and environments (like GNU Octave) will play increasing role in HPC
- High productivity will be achieved through better programming environments that leverage today's HPC tools

HPCS Contributors



Ashok Krishnamurthy
Stan Ahalt
David Hudak
Aakash Dalwani



John Eaton



Andy Funk
Theresa Meuse
Julia Mullen
Jeremy Kepner



David Bader
Kamesh Madduri



Piotr Luszczek



Mark Mitchell
Stefan Seefeld



John Gilbert
Viral Shah

