Computational Methods in Astrophysics

Dr Rob Thacker (AT319E)

thacker@ap.smu.ca

Parallel Libraries/Toolkits

- BLACS
- ScaLAPACK
- Higher level approaches like PETSc
- CACTUS

Netlib

- The Netlib repository contains
  - freely available software, documents, databases of interest to the numerical & scientific computing communities
- The repository is maintained by
    - AT&T Bell Laboratories
    - University of Tennessee
    - Oak Ridge National Laboratory
- The collection is mirrored at several sites around the world
  - Kept synchronized
- Effective search engine to help locate software of potential use

High Performance LINPACK

- Portable and freely available implementation of the LINPACK Benchmark – ***used for Top500 ranking***
- Developed at UTK Innovative Computing Laboratory
  - A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary
- **HPL** solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers
  - Requires MPI 1.1 be installed
  - Also requires an implementation of **either** the **BLAS or** the Vector Signal Image Processing Library **VSIPL**
- Provides a testing and timing program
  - Quantifies the **accuracy** of the obtained solution as well as the time it took to compute it

BLACS

- Basic Linear Algebra Communication Subprograms
- Conceptual aid in design and coding (design tool)
  - Think of it as a communications library for linear algebra
- Associate widely known mnemonic names with communication
  - Improved readability and provides standard interface
  - “Self documentation”

BLACS data decomposition

	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

2d processor grid

Communication Modes:
All processes in row
All processes in column
All grid processes

Types of BLACS routines: point-to-point communication, broadcast, combine operations and support routines.

Communication Routines

- Send/Receive
  - Send (sub)matrix from one process to another:
  - _xxSD2D(ICTXT, [UPLO,DIAG], M, N, A, LDA, RDEST,CDEST)
  - _xxRV2D(ICTXT, [UPLO,DIAG], M, N, A, LDA, RSRC, CSRC)
- _ denotes datatype:
  - I (integer), S (single), D (double), C (complex), Z (double complex)
- xx denotes matrix type
  - GE = general, TR=trapezoidal

Point-to-Point example

```
 CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL,
&                      MYROW, MYCOL )

 IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
  CALL DGESD2D( ICTXT, 5, 1, X, 5, 1, 0 )
 ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
  CALL DGERV2D( ICTXT, 5, 1, Y, 5, 0, 0 )
 END IF
```

Contexts

- The concept of a communicator is imbedded within BLACS as a “context”
- Contexts are thus the mechanism by which you:
  - Create arbitrary groups of processes upon which to execute
  - Create an indeterminate number of overlapping or disjoint grids
  - Isolate each grid so that grids do not interfere with each other
- Initialization routines return a context (integer) which is then passed to the communication routines
  - Equivalent to specifying COMM in MPI calls

ID less communication

- Messages with BLACS are tagless
  - Generated internally within the library
- Why is this an issue?
  - If tags are not unique it is possible to create not deterministic behaviour (have race conditions on message arrival)
- BLACS allows the user to specify what range of IDs can use
  - This ensures it can be used with other packages

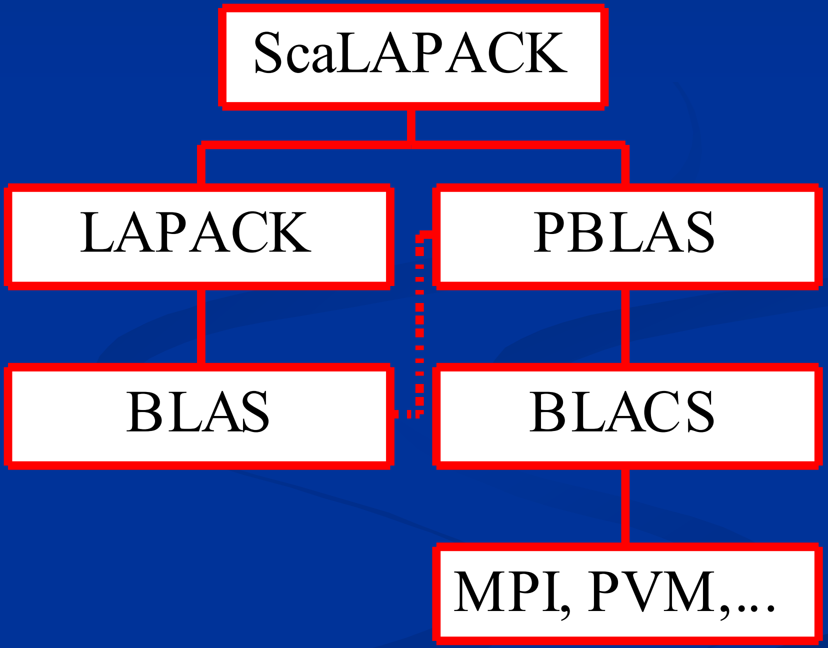
ScaLAPACK

- Scalable LAPACK
- Development team
  - University of Tennessee
  - University of California at Berkeley
  - ORNL, Rice U.,UCLA, UIUC etc.
- Support in Commercial Packages
  - Intel MKL and AMD ACML
  - IBM PESSL
  - CRAY Scientific Library
  - +others

Important details

- Web page http://www.netlib.org/scalapack
  - Includes ScaLAPACK User's Guide
- Language : Fortran
- Dense Matrix Problem Solvers
  - Linear Equations
  - Least Squares
  - Eigenvalue

Package dependencies



Components of the API

- Drivers
  - Solves a Complete Problem
- Computational Components
  - Performs Tasks: LU factorization, etc.
- Auxiliary Routines
  - Scaling, Matrix Norm, etc.
- Matrix Redistribution/Copy Routine
  - Matrix on PE grid1 -> Matrix on PE grid2

API (cont..)

- LAPACK names with P prefix

PXYYZZZ

Computation Performed

Matrix Type

Data Types

Data Type	real	double	cmplx	dble cmplx
X	S	D	C	Z

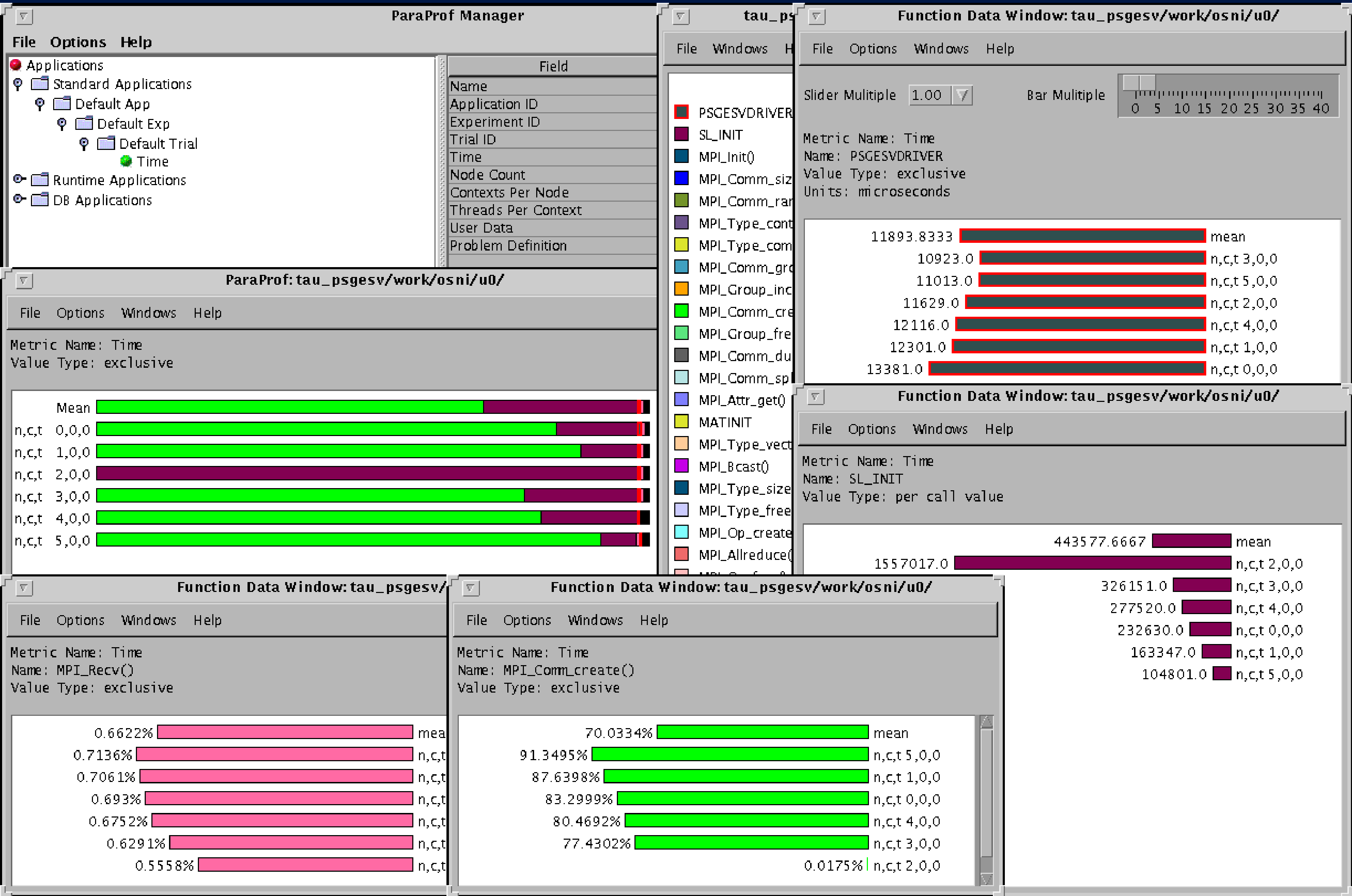
TAU

- Tuning and Analysis Utilities
  - University of Oregon development
  - http://www.cs.uoregon.edu/research/tau
- Program and performance analysis tool framework for high-performance parallel and distributed computing
  - TAU provides a suite of tools analysis of C, C++, FORTRAN 77/90, Python, High Performance FORTRAN, and Java programs

Useage

- Instrument the program by inserting TAU macros into the program (this can be done automatically).
- Run the program. Files containing information about the program performance are automatically generated.
- View the results with TAU's pprof, the TAU visualizer racy (or paraprof), or a third-party visualizer (such as VAMPIR)

pprof


ParaProf Manager
File Options Help
Applications
Standard Applications
Default App
Default Exp
Default Trial
Time
Runtime Applications
DB Applications
Field
Name
Application ID
Experiment ID
Trial ID
Time
Node Count
Contexts Per Node
Threads Per Context
User Data
Problem Definition
ParaProf: tau_psgesv/work/osni/u0/
File Options Windows Help
Metric Name: Time
Value Type: exclusive
Mean
n,c,t 0,0,0
n,c,t 1,0,0
n,c,t 2,0,0
n,c,t 3,0,0
n,c,t 4,0,0
n,c,t 5,0,0
PSGESVDRIVER
SL_INIT
MPI_Init()
MATINIT
MPI_Bcast()
Function Data Window: tau_psgesv/work/osni/u0/
File Options Windows Help
Slider Mulitiple 1.00
Bar Mulitiple
0 5 10 15 20 25 30 35 40
Metric Name: Time
Name: PSGESVDRIVER
Value Type: exclusive
Units: microseconds
11893.8333 mean
10923.0 n,c,t 3,0,0
11013.0 n,c,t 5,0,0
11629.0 n,c,t 2,0,0
12116.0 n,c,t 4,0,0
12301.0 n,c,t 1,0,0
13381.0 n,c,t 0,0,0
Function Data Window: tau_psgesv/work/osni/u0/
File Options Windows Help
Metric Name: Time
Name: SL_INIT
Value Type: per call value
443577.6667 mean
1557017.0 n,c,t 2,0,0
326151.0 n,c,t 3,0,0
277520.0 n,c,t 4,0,0
232630.0 n,c,t 0,0,0
163347.0 n,c,t 1,0,0
104801.0 n,c,t 5,0,0
Function Data Window: tau_psgesv/work/osni/u0/
File Options Windows Help
Metric Name: Time
Name: MPI_Recv()
Value Type: exclusive
0.6622%
0.7136%
0.7061%
0.693%
0.6752%
0.6291%
0.5558%
Function Data Window: tau_psgesv/work/osni/u0/
File Options Windows Help
Metric Name: Time
Name: MPI_Comm_create()
Value Type: exclusive
70.0334% mean
91.3495% n,c,t 5,0,0
87.6398% n,c,t 1,0,0
83.2999% n,c,t 0,0,0
80.4692% n,c,t 4,0,0
77.4302% n,c,t 3,0,0
0.0175% n,c,t 2,0,0

Additional facilities

- TAU collects much more information than what is available through prof or gprof, the standard Unix utilities. Also available through TAU are:
  - Per-process, per-thread and per-host information (supports pthreads)
  - Inclusive and exclusive function times
  - Profiling groups that allow you to organize data collection
  - Access to hardware counters on some systems
  - Per-class and per-instance information
  - Separate data for each template instantiation
  - Start/stop timers for profiling arbitrary sections of code
  - Support for collection of statistics on user-defined events
- TAU is designed so that when you turn off profiling (by disabling TAU macros) there is no overhead

PETSc

- Portable, Extensible Toolkit for Scientific Computation
  - https://www.mcs.anl.gov/petsc/
  - Argonne lab development – used in 763 papers to date
  - 20 years old! Approach must have good points! ☺
- Suite of data structures and routines for the scalable (parallel) solution of PDEs
  - Intended for use in large-scale application projects
  - Not a black box solution though
- Easily interfaces with solvers written in C, FORTRAN and C++
- All components are designed to be interoperable
- Works in distributed memory environment using MPI

Levels of Abstraction in Mathematical Software

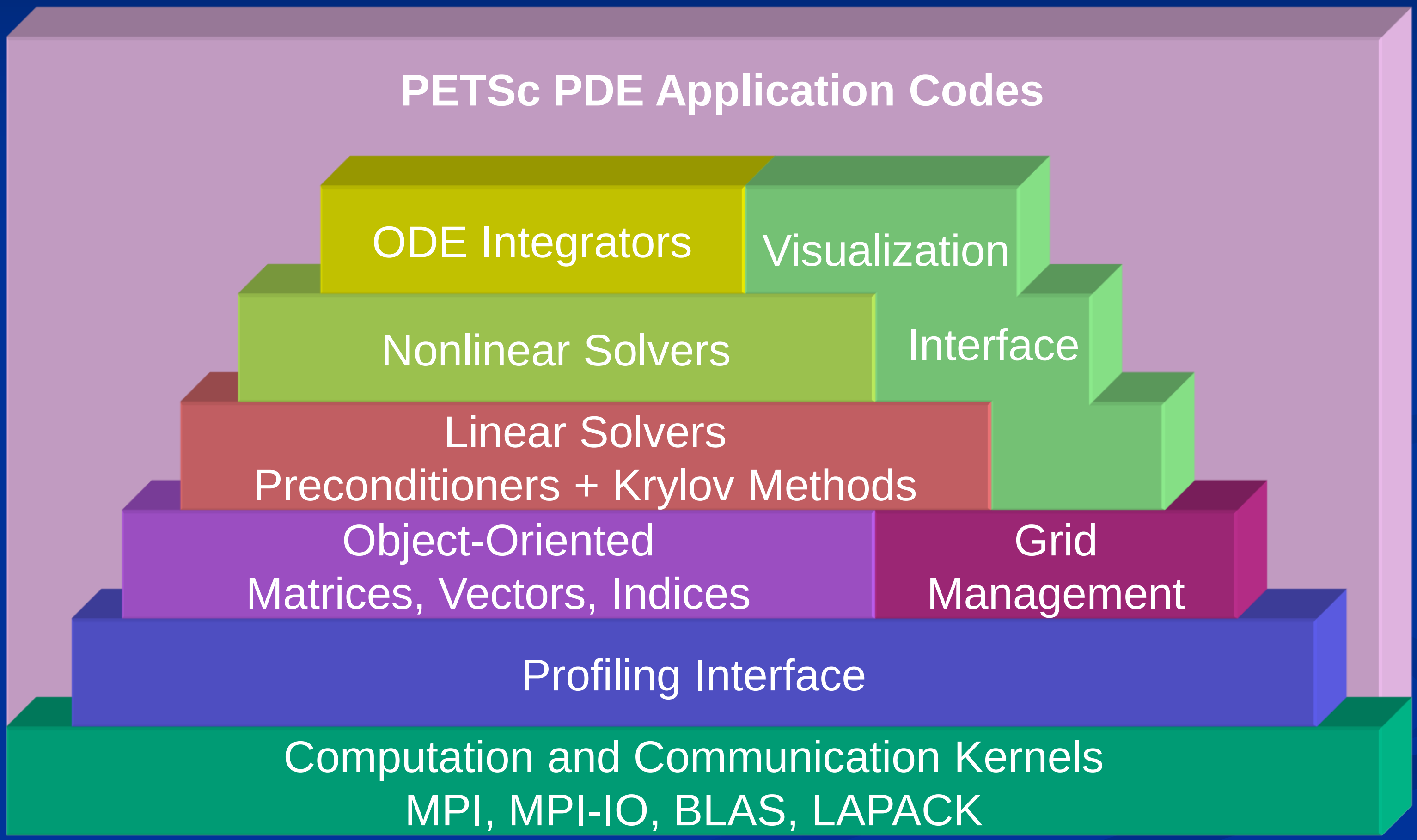
- Application-specific interface
  - Programmer manipulates objects associated with the application
- High-level mathematics interface
  - Programmer manipulates mathematical objects
    - Weak forms, boundary conditions, meshes
- Algorithmic and discrete mathematics interface
  - Programmer manipulates mathematical objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
- Low-level computational kernels
  - BLAS-type operations
  - FFT

PETSc emphasis (pointing to: Algorithmic and discrete mathematics interface)

Features

- Parallel vectors
  - scatters
  - gathers
- Parallel matrices
  - several sparse storage formats
  - easy, efficient assembly.
- Scalable parallel preconditioners
- Krylov subspace methods
- Parallel Newton-based nonlinear solvers
- Parallel timestepping (ODE) solvers
- Complete documentation
- Automatic profiling of floating point and memory usage
- Consistent interface
- Intensive error checking
- Portable to UNIX and Windows
- Over one hundred examples
- PETSc is supported and will be actively enhanced for the next several years.

Structure of PETSc – Layered Approach



Functionality example: selected vector operations

Function Name	Operation
VecAXPY(Scalar *a, Vec x, Vec y)	$y = y + a*x$
VecAYPX(Scalar *a, Vec x, Vec y)	$y = x + a*y$
VecWAXPY(Scalar *a, Vec x, Vec y, Vec w)	$w = a*x + y$
VecScale(Scalar *a, Vec x)	$x = a*x$
VecCopy(Vec x, Vec y)	$y = x$
VecPointwiseMult(Vec x, Vec y, Vec w)	$w_i = x_i * y_i$
VecMax(Vec x, int *idx, double *r)	$r = max\ x_i$
VecShift(Scalar *s, Vec x)	$x_i = s+x_i$
VecAbs(Vec x)	$x_i = \|x_i\|$
VecNorm(Vec x, NormType type , double *r)	$r = \|\|x\|\|$

A Complete PETSc Program

```
#include petscvec.h
int main(int argc,char **argv)
{
  Vec x;
  int n = 20,ierr;
  PetscTruth flg;
  PetscScalar one = 1.0, dot;

  PetscInitialize(&argc,&argv,0,0);
  PetscOptionsGetInt(PETSC_NULL,"-n",&n,PETSC_NULL);
  VecCreate(PETSC_COMM_WORLD,&x);
  VecSetSizes(x,PETSC_DECIDE,n);
  VecSetFromOptions(x);
  VecSet(&one,x);
  VecDot(x,x,&dot);
  PetscPrintf(PETSC_COMM_WORLD,"Vector length %dn",(int)dot);
  VecDestroy(x);
  PetscFinalize();
  return 0;
}
```

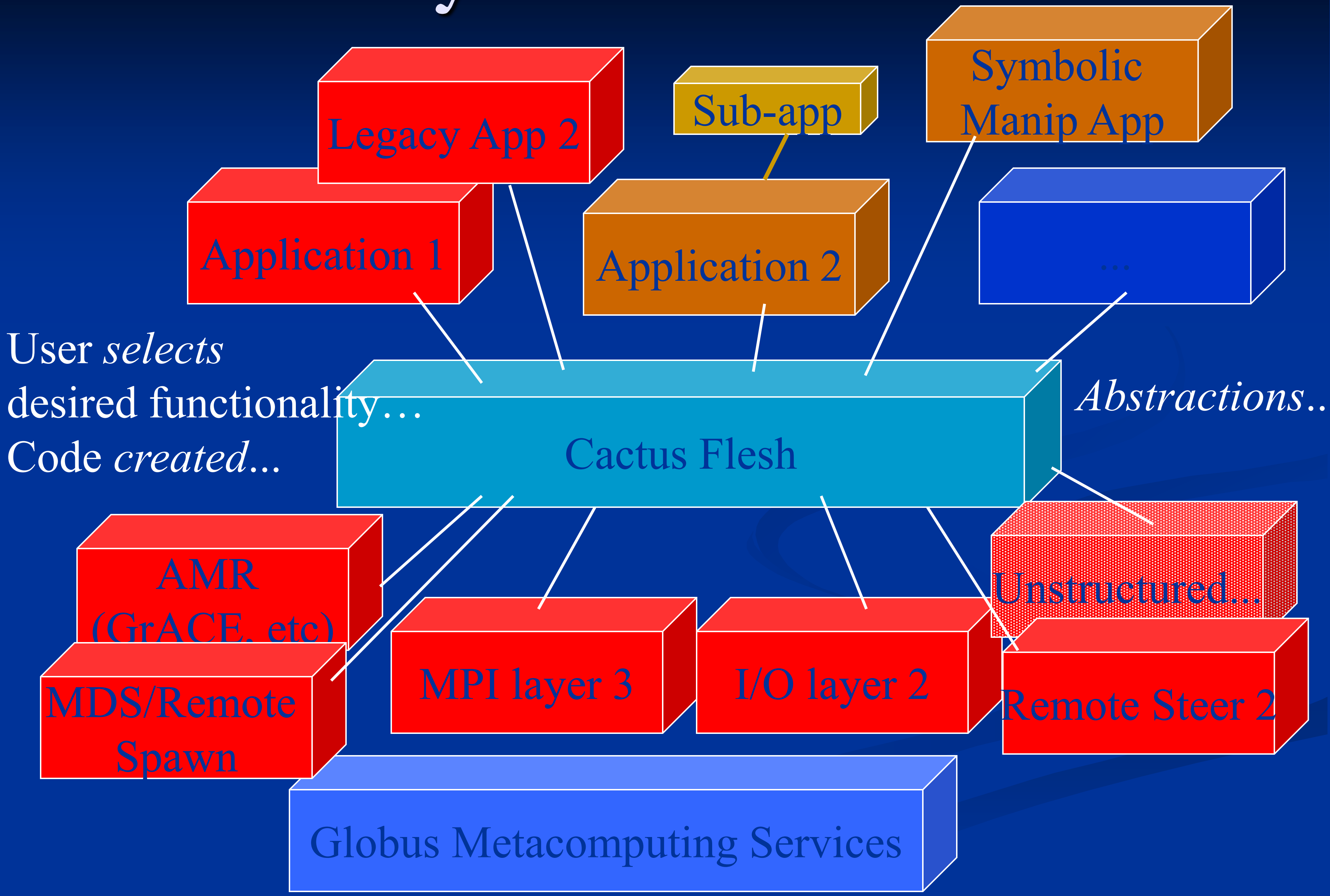
TAO

- Toolkit for Advanced Optimization
  - Now included in PETSc distribution
  - Another Argonne project
- Aimed at the solution of large-scale optimization problems on high-performance architectures
  - Suitable for both single-processor and massively-parallel architecture
- Object oriented approach

CACTUS

- http://www.cactuscode.org/
- Developed as response to needs of large scale projects (initially developed for General Relativity calculations which have a large computation to communication ratio)
- Numerical/computational infrastructure to solve PDE's
- Freely available, *Open Source* community framework
  - Cactus Divided in "Flesh" (core) and "Thorns" (modules or collections of subroutines)
  - Multilingual: User apps Fortran, C, C^{++}; automated interface between them
- Abstraction: Cactus Flesh provides API for virtually all CS type operations
  - Storage, parallelization, communication between processors, etc
  - Interpolation, Reduction
  - IO (traditional, socket based, remote viz and steering…)
  - Checkpointing, coordinates
- "Grid Computing": Cactus team and many collaborators worldwide, especially NCSA, Argonne/Chicago, LBL

Modularity of Cactus...



Cactus & the Grid

Cactus Application Thorns

Distribution information hidden from programmer

Initial data, Evolution, Analysis, etc

Grid Aware Application Thorns

Drivers for parallelism, IO, communication, data mapping

PUGH: parallelism via MPI

(MPICH-G2, grid enabled message passing library)

Single Proc

Standard MPI

Grid Enabled Communication Library

MPICH-G2 implementation of MPI, can run MPI programs across heterogenous computing resources

The Flesh

- Abstract API
  - evolve the same PDE with unigrid, AMR (MPI or shared memory, etc) without having to change any of the application code.
- Interfaces
  - set of data structures that a thorn exports to the world (global), to its friends (protected) and to nobody (private) and how these are **inherited**.
- Implementations
  - Different thorns may implement e.g. the evolution of the same PDE and we select the one we want at runtime.
- Scheduling
  - call in a certain order the routines of every thorn and how to handle their interdependencies.
- Parameters
  - many types of parameters and all of their essential consistency checked before running

VTK

- The Visualization Toolkit
  - http://public.kitware.com/VTK/what-is-vtk.php
- Portable open-source software system for 3D computer graphics, image processing, and visualization
  - Object-oriented approach
- VTK is at a higher level of abstraction than rendering libraries like OpenGL
- VTK applications can be written directly in C++, Tcl, Java, or Python
- Large user community
  - Many source code contributions

Summary

- One interesting note – portability continues to be a real issue with the design of APIs at a higher level of abstraction
- If you want to do big linear algebra there are numerous well optimized libraries
  - Lots of knowledge out there too
- PDEs are also reasonably well supported within existing library frameworks – but variety of available solvers is always an issue
- Packages with strong utility seem to survive