Christmas gifts for CSE

Ravikishore Kommajosyula

Christmas gifts for CSE (aka) Open source Packages for Scientific Computing

Ravikishore Kommajosyula

The ideal Christmas gift

- Writing a working code is (more or less) trivial....atleast if you do this course ;-)
- Efficient code is not always trivial… hardware etc.
- Ideal scenario: Tested, highly efficient code to replace the compute intensive part (kernel) of your code
- Good News: A huge variety of open-source codes available on the Internet
- Today: A brief overview on different types of Packages available and how to use them

Overview

- Scientific computing codes typically have a huge compute intensive part, which use a standard pool of methods
- Replacing compute intensive parts with optimized code (could) give big improvements
- Briefly discuss major tools, libraries and frameworks
- This presentation is by no means comprehensive, but only indicative of the scope of available tools

Types of Packages

- Tools: A fully functional code (or executable) that does a task. Used separately from our code. Ex. – Visualization tools
- Libraries: A collection of functions / classes / values that can be used in our code.

  Ex. – GCC Math libraries etc.
- Frameworks: A code providing generic functionality can be selectively changed by additional user-written code. Ex. – OpenNI
- You call a library, A framework calls you

Tools, Libraries or Frameworks?

- Tools are typically self contained and less customizable / extensible.
- Libraries are made to have a good interface and lot of flexibility for using with our code
- Frameworks have less flexibility than libraries, but offer a lot more in terms of capabilities.
- Libraries are most common (esp. in SC) ….from now on, focus of presentation on Libraries

GSL - GNU Scientific Library

- Numerical library for C and C++ programmers. Interfaces available to several languages
- Portable, fast, modern algorithms

Complex Numbers	Roots of Polynomials
Special Functions	Vectors and Matrices
Permutations	Sorting
Differential Equations	Linear Algebra
Interpolation	Fast Fourier Transforms

and much more…

- Started in May 1996. Latest version (gsl-1.16 released in July 2013)

GSL – A simple example

- Sample code using Bessel function

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>
int main (void)
{
double x = 5.0 ;
// function call
double y = gsl_sf_bessel_J0 (x) ;
printf( "J0(%g) = %.18e\n", x, y ) ;
return 0 ;
}
```

- Easy to use library
- Function naming convention –
  - gsl_foo_fn (type double)
  - gsl_foo_int_fn (type int)
- Module naming convention –
  - gsl_foo (type double)
  - gsl_foo_int (type int)
- Support for long double
- All functions are thread-safe

BLAS – Basic Linear Algebra Subprograms

- Set of low-level kernel subroutines that perform matrix and vector operations
- Implemented in FORTRAN. C interface exists.
- Machine specific optimized BLAS libraries available for various computer architectures

Complexity	Operations
Level 1 – O(N)	Vector - Vector
Level 2 – O(N2)	Vector - Matrix
Level 3 – O(N3)	Matrix - Matrix

- Used in development of high quality codes. Ex. LAPACK, PETSC, GSL etc.

ATLAS – Automatically Tuned Linear Algebra Software

- Mature open source **implementation of BLAS**
- Provides portable performance
- Recommended as a way to automatically generate an optimized BLAS Library
- Provides C and Fortran 77 interface to BLAS routines and few routines from LAPACK
- Many mathematics applications like MATLAB, Mathematica, Scilab, Sage use it

LAPACK - Linear Algebra PACKage

- Software library for numerical linear algebra
- Provides routines for :
  - Solving system of linear equations
  - Linear least squares
  - Eigen value problems
  - Singular value decomposition etc.
- Written in FORTRAN. C++ Interface exists
- Exploits BLAS routines for Matrix operations
- Dense and Banded matrices are handled, but not general sparse matrices

BLAS,LAPACK Parallel Implementations

- PARBLAS: Parallel BLAS is an OPEN MP parallel (shared memory) version of BLAS.
- Any BLAS code with environment variable OMP_NUM_THREADS set can run on PARBLAS
- SCALAPACK: MPI parallel (distributed memory) implementation of a subset of LAPACK routines
- Scalable code that can be used in MPI programs
- Reliability of algorithms (error bound)
- Ease of use (interface to LAPACK and SCALAPACK are very similar)

FFTW – Fast Fourier Transform in the West

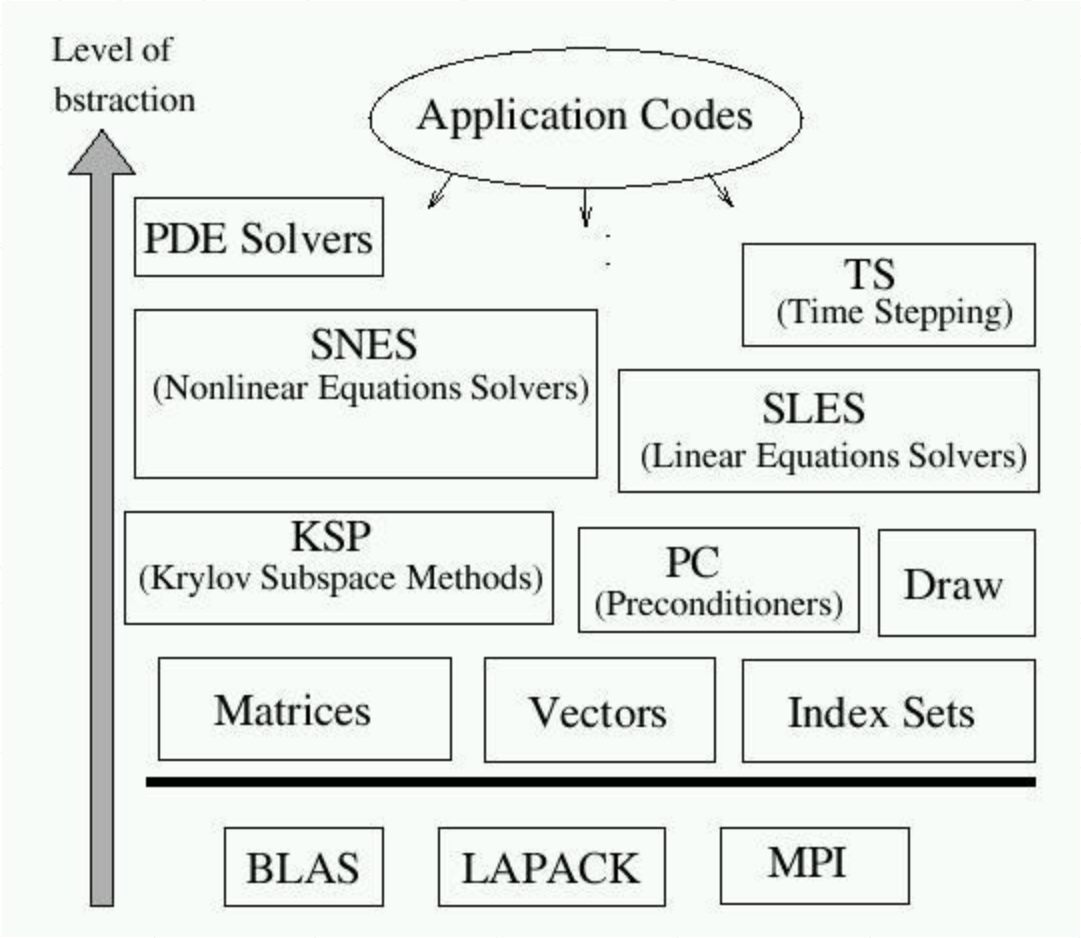
- A free, open source C subroutine library for computing the discrete Fourier Transform
- Handles one and multi dimensional problems
- Real and complex data of arbitrary size
- Versions for OpenMP and MPI parallelisation
- Support for SSE / SSE2 / AVX
- Portable to any platform with a C Compiler
- The FFT library of choice for most applications

PETSc

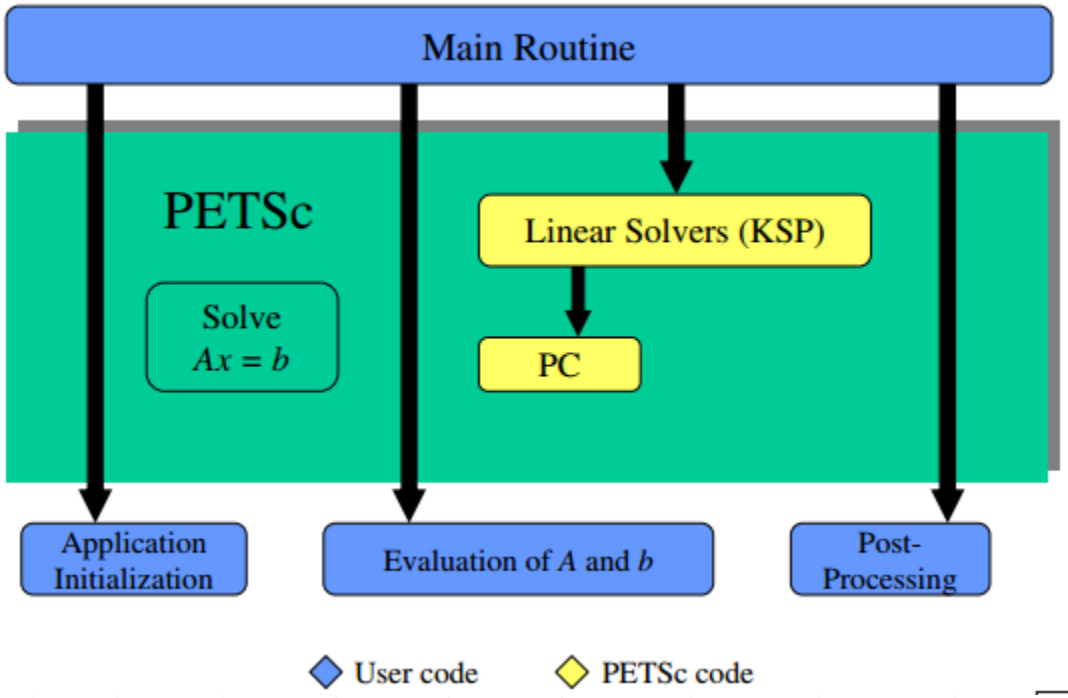
- Portable Extensible Toolkit for Scientific Computing
- A suite of data structures and routines for scalable solution of scientific applications modelled by PDEs
- Helps develop parallel, highly efficient PDE solvers
- Library while provides
  - Parallel vector / array operations
  - Matrix storage formats (sparse / distributed)
  - Linear and non linear solvers
  - ODE Integrators, Parallel grid / data management
- Uses BLAS and LAPACK in the background

PETSc….continued

- Schematic diagram showing different components of the PETSc library, categorized by the level of abstraction.
- User code uses PETSc at the desired level of abstraction



PETSc....a simple example



- A simple example of solving a linear system using a Krylov subspace solver. Notice that we are working at a very higher level of abstraction while running an MPI Parallel job

How to choose a Tool/Library/Framework

- Evaluating several packages before picking one
- Does it do what you think it does?
- Does it have a usable interface?
- Is it well documented?
- Is it MPI Parallel? OpenMP parallel? Thread safe?
- Does it have support? Is it extensible? Is it portable?
- Does the package integrate well with existing code?
- Is it open-source? Free? Available? Dead?
- When you use a package, it becomes YOUR code

Final Remarks

- Don’t re-invent the wheel. Chances are that someone already did it…and did it well enough
- Reduce development type by focusing on specifics
- Read the license / type of license carefully before you use the code. Some licenses require you to share your source code when you publish your work
- Always give due credits and reference other people’s work if you use it in yours
- Contribute to the society if you find something better

Thank you for your attention!
Merry Christmas!!

Ravikishore Kommajosyula