Scientific Computing
Tools, Libraries & Frameworks

Yaakoub El Khamra
yaakoub@tacc.utexas.edu

Overview

- Supercomputing has been around for a while, so there is no shortage of a available (re-usable) codes that you can use
- Chances are you will be using "canned" software or at least "canned" software components/tools/libraries in your work (blas/lapack/boost/numpy or something similar)
- In most cases, you want to reduce the time spent in development and improve code performance, so you use a tool/library/framework with your code

Overview

- We will briefly describe the major tools, libraries and frameworks
- This overview is by no means comprehensive, there are lots of great tools/libraries and frameworks out there
- Some of these packages are available on our systems, others you will have to build yourself. Send in tickets if you run into problems

How to Pick a Tool/Library/Framework

- Picking a tool/library/framework (henceforth: a package) to use can be tricky
- I strongly recommend evaluating several packages before picking a winner

Things to consider:

- Does it do what you think it does? i.e. is it useful?
- Does it have a usable interface that is usable, comprehensive enough for your purpose, well documented and in a language you can access?

Things To Consier

- What about performance: is it MPI parallel? OpenMP parallel? Thread safe?
- Does it have support? Will it be around long enough for you to use it to your satisfaction? Is there a community you can go to for help?
- Do you need to extend its capabilities? Is it inherently extensible? Will you need to modify the internals extensively to get it to work to your satisfaction?
- Do you have existing code? Will the package play nice with it? Will it take over your main function? Will you have to re-write your existing code?

How to Pick a Tool/Library/Framework (cont)

Things to consider (cont)

- Is it open-source? Is it free? Is it available? Is it dead? Is it written in Basic/Fortran66/Lisp?
- Does it compile? Does it run? Does it give the right answers?

Keep in mind: when you use a package in your code, it becomes your code. You are responsible for the answers it gives and the implications thereof

Package Breakdown

- We need to distinguish between Tools, Libraries and Frameworks:
  - Tools: other people's code that you use separately from your own code (i.e. someone else's executable or script)
  - Libraries: a collection of routines/classes/values/functions that you can call from your own code (still your executable)
  - Frameworks: this is a blanket term and can be used to mean different things. In general, frameworks require you to re-write (sometimes import) your code to include components from the framework in order to use its functionality. If you no longer have a "main" function, you are using a framework.

Tools, Libraries or Frameworks?

Chances are you will not find tools, libraries and frameworks that all do the same thing. If you do, there are advantages and disadvantages to each, so choose wisely:
Tools are typically self contained. Compile and use. They are less customizable than libraries (limited interface) and much less extensible than frameworks. You can however get away without any programming (i.e. no code of your own) with a tool.

Tools, Libraries or Frameworks?

Libraries are meant to be used with your own code, so typically they have a good interface and a lot of flexibility in their use if the interface is well designed. If the interface changes, chances are you will have to change your code.

Keep in mind: mixed language programming (C/C++/Fortran) can be tricky and sometimes a library will not have an interface to the language you wrote you the rest of your code in

Tools, Libraries or Frameworks

Frameworks have less flexibility than libraries but offer a lot more in terms of capabilities. There are frameworks that offer automatic parallelization with built in adaptive mesh refinement, automatic checkpoint/restart mechanisms, dynamic load balancing, self monitoring and a whole lot more. The price you pay would be control over a lot of aspects of your code: scheduling, the main function and sometimes a few more intricate aspects of your code (e.g. no static global variables)

There was a time when frameworks were very popular. These days, there are only a few frameworks that are being actively developed. You will find a lot of tools for dedicated applications (e.g. only solving shallow water equations, or only doing a certain type of MD simulations). Libraries are most common today, since they potentially offer greatest flexibility in terms of usage, interface and extensibility

Popular/available packages

Logically, we will distribute packages based on what they have to offer. In broad, general terms, major categories are as follows:

- Mathematics: numerical methods, linear algebra, symbolic algebra, mathematical functions, statistics
- Physics: astronomy, cosmology, nuclear and particle physics, statistical physics
- Chemistry: molecular dynamics, quantum chemistry
- Biology: DNA alignment, phylogenetics
- Engineering: aeronautics, civil, structural, environmental, ecological, material science, naval engineering

Many more disciplines have packages that are used on supercomputers. I just happened to pick the ones I found to be popular on XSEDE/TeraGrid that I know of.

General Purpose Packages

The following packages are general purpose math packages. These packages know NOTHING of the physical problem but offer everything from mesh decomposition to solvers.

We will cover the major ones that are noteworthy and popular. There are obviously many other packages that you can use out there.

Mathematics: BLAS

Obviously a lot of these packages are used in conjunction with other domain specific packages, and so this is a good place to start. BLAS: Basic Linear Algebra Subprograms. BLAS libraries provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations. Because BLAS libraries are usually efficient, and widely available, they are commonly used in the development of high quality linear algebra software. Many implementations exist: netlib's BLAS, GotoBLAS, mkl, acml etc... and C/C++ and Fortran interfaces. Single core and multithreaded versions available.

PBLAS: parallel version of BLAS, containing the same 3 levels of functions (scalars & vectors, vector-vector and vector-matrix).

LAPACK
Linear Algebra PACKage

LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.
LAPACK uses a lot of level 3 BLAS functions, is available in many implementations (mkl, acml and of course netlib) with C/C++/Fortran interfaces.

ScaLAPACK

The ScaLAPACK (or Scalable LAPACK) library includes a subset of LAPACK routines redesigned for distributed memory (i.e. MPI) parallel computers. It assumes matrices are laid out in a two-dimensional block cyclic decomposition. The fundamental building blocks of the ScaLAPACK library are distributed memory versions (PBLAS) of the Level 1, 2 and 3 BLAS, and a set of Basic Linear Algebra Communication Subprograms (BLACS) for communication tasks that arise frequently in parallel linear algebra computations. In the ScaLAPACK routines, all interprocessor communication occurs within the PBLAS and the BLACS. One of the design goals of ScaLAPACK was to have the ScaLAPACK routines resemble their LAPACK equivalents as much as possible.
ScaLAPACK can be thought of as an MPI enabled LAPACK. C/C++ and Fortran interfaces available in mkl/acml and netlib implementations

ARPACK, PARPACK, PLAPACK

- ARPACK is a collection of Fortran77 subroutines designed to solve large scale eigenvalue problems
- PARPACK is a parallel version of ARPACK
- PLAPACK parallel dense linear algebra algorithms (similar to ScaLAPACK but with better performance)

Not much in terms of development of ARPACK and PARPACK for about a decade. No new release or bugfixes. While the code is still available, you might want to be be careful if you are hoping to get a lot of support. Since the codes are F77, you can interface from C/C++ and Fortran
PLAPACK is well documented (entire book as a user guide) and has active support. It even has a spiritual successor: FLAME.

FFTW 2 & 3

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). I believe that FFTW, which is free software, should become the FFT library of choice for most applications.
mkl and acml both implement FFT's but FFTW is a popular choice.

There was a major overhaul of the FFTW interface from 2.x to 3.x. Furthermore 3.3-beta1 supports distributed-memory transforms.

GSL, MPFR and GMP

- The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite. Supports C/C++, for Fortran use FGSL
- The MPFR library is a C library for multiple-precision floating-point computations with correct rounding. MPFR is based on GMP.
- GMP is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface. GMP has a C/C++ interface but no Fortran support

ParMETIS and SCOTCH

- ParMETIS is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. ParMETIS extends the functionality provided by METIS and includes routines that are especially suited for parallel AMR computations and large scale numerical simulations. The algorithms implemented in ParMETIS are based on the parallel multilevel k-way graph-partitioning, adaptive repartitioning, and parallel multi-constrained partitioning schemes. Supports C/C++ and F90
- SCOTCH is a software package and libraries for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning. It has C/C++ and Fortran90 support.

Zoltan

Zoltan is a set of libraries that provide parallel partitioning, load balancing and data-management services. It can be used to build unstructured mesh codes (FEM/FVM). It has interfaces to both ParMETIS and SCOTCH for parallel partitioning and can perform dynamic load balancing and ghost zone exchange.

Zoltan requires C/C++. There is no Fortran77 interface, there is a Fortran 90 interface. It is worth mentioning that Zoltan does not scale well into thousands of processors but for small to medium size problems (tens of millions of degrees of freedom) it can drastically reduce the time spent in development. Zoltan 3.5 was released last March, so it is actively developed (and funded).

SAMRAI

SAMRAI (Structured Adaptive Mesh Refinement Application Infrastructure) is an object-oriented C++ software library enables exploration of numerical, algorithmic, parallel computing, and software issues associated with applying structured adaptive mesh refinement (SAMR) technology in large-scale parallel application development. SAMRAI provides software tools for developing SAMR applications that involve coupled physics models, sophisticated numerical solution methods, and which require high-performance parallel computing hardware. SAMRAI enables integration of SAMR technology into existing codes and simplifies the exploration of SAMR methods in new application domains. Due to judicious application of object-oriented design, SAMRAI capabilities are readily enhanced and extended to meet specific problem requirements.

SAMRAI has not had a release since 2008 (not that I have found). The mailing lists are fairly active but you will want to assess the level of support you will get before investing heavily in it

Chombo

The Chombo package provides a set of tools for implementing finite difference methods for the solution of partial differential equations on block-structured adaptively refined rectangular grids. Both elliptic and time-dependent modules are included. Support for parallel platforms and standardized self-describing file formats are included. Chombo provides a distributed infrastructure for parallel calculations over block-structured, adaptively refined grids. Chombo's design is uniquely flexible and accessible.

The last release of Chombo was in 2009. It is a useful package if you are going to be solving PDE's with finite difference methods. It has C/C++ and Fortran interfaces.

PARAMESH

PARAMESH is a package of Fortran 90 subroutines designed to provide an application developer with an easy route to extend an existing serial code which uses a logically cartesian structured mesh into a parallel code with adaptive mesh refinement (AMR).

Alternatively, in its simplest use, and with minimal effort, it can operate as a domain decomposition tool for users who want to parallelize their serial codes, but who do not wish to use adaptivity.

The package builds a hierarchy of sub-grids to cover the computational domain, with spatial resolution varying to satisfy the demands of the application. These sub-grid blocks form the nodes of a tree data-structure (quad-tree in 2D or oct-tree in 3D). Each grid block has a logically cartesian mesh.

Aside: Adaptive Mesh Refinement: Way too many options

- Sundance
- Sumaa3D
- Sierra
- KeLP
- AGRIF
- Nirvana
- CTH-AMR
- Sfumato
- Athena
- Amiga
- FeTK
- Rhea
- deal.II
- LibMesh
- Sieve
- AMROC
- DAGH/Grace
- DukeAMR
- CCSE
- Paramesh
- Charm++
- HPC-MW
- Trellis
- Pyramid
- Cactus/Carpet
- Overture
- Enzo
- RAMSES
- Cart3D
- SAMRAI
- Chombo

Aside: Adaptive Mesh Refinement: The Free Ones

- KeLP
- AGRIF
- Nirvana
- Athena
- Amiga
- FeTK
- Rhea
- deal.II
- LibMesh
- Sieve
- AMROC
- DAGH/Grace
- DukeAMR
- CCSE
- Paramesh
- Charm++
- HPC-MW
- Pyramid
- Cactus/Carpet
- Overture
- Enzo
- RAMSES
- SAMRAI
- Chombo

Aside: Adaptive Mesh Refinement: The Actively Developed Ones

- Nirvana
- Athena
- Amiga
- FeTK
- Rhea
- deal.II
- Sieve
- LibMesh
- AMROC
-
- Paramesh
- Charm++
- Pyramid
- Cactus/Carpet
- Overture
- Enzo
- SAMRAI (maybe)
- Chombo (maybe)

Aside: Adaptive Mesh Refinement: The Ones That Scale well

- Rhea
- AMROC

-
-
-
- Paramesh
- Charm++ (in some cases)
- Pyramid
- Cactus/Carpet
- Overture
- Enzo
- SAMRAI
- Chombo

Aside: Adaptive Mesh Refinement: Elliptic and Parabolic Eqn. Support

-
-
-
- Paramesh
- Pyramid
- Cactus/Carpet
- SAMRAI
- Chombo

Aside: Adaptive Mesh Refinement

There are plenty of options, be careful which package/library/framework you use.

Personally, I almost always choose PETSc.

PETSc

Portable Extensible Toolkit for Scientific computation

PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for parallelism.
PETSc has been used for modeling in all of these areas: Acoustics, Aerodynamics, Air Pollution, Arterial Flow, Bone Fractures, Brain Surgery, Cancer Surgery, Cancer Treatment, Carbon Sequestration, Cardiology, Cells, CFD, Combustion, Concrete, Corrosion, Dentistry, Earth Quakes, Economics, Fission, Fusion, Glaciers, Ground Water Flow, Mantel Convection, Magnetic Films, Material Science, Medical Imaging, Ocean Dynamics, Oil Recover, PageRank, Polymer Injection Molding, Polymeric Membrances, Seismology, Semiconductors, Rockets, Relativity, Surface Water Flow.

PETSc Continued

PETSc is intended for use in large-scale application projects. PETSc is easy to use for beginners. Moreover, its careful design allows advanced users to have detailed control over the solution process. PETSc includes a large suite of parallel linear, nonlinear equation solvers and ODE integrators that are easily used in application codes written in C, C++, Fortran and now Python.
PETSc provides many of the mechanisms needed within parallel application codes, such as simple parallel matrix and vector assembly routines that allow the overlap of communication and computation. In addition, PETSc includes support for parallel distributed arrays useful for finite difference methods.

PETSc Extensions

SLEPc is a software library for the solution of large scale sparse eigenvalue problems on parallel computers. It is an extension of PETSc and can be used for either standard or generalized eigenproblems, with real or complex arithmetic. It can also be used for computing a partial SVD of a large, sparse, rectangular matrix, and to solve quadratic eigenvalue problems.

PETSc4Py python extension for PETSc
PETScExt Block operator extensions to PETSc (think variable multiplexing)
libMesh provides a framework for the numerical simulation of partial differential equations using arbitrary unstructured discretizations on serial and parallel platforms. It supports adaptive mesh refinement computations in parallel while allowing a research scientist to focus on the physics they are modeling.

Trilinos

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages.

Packages include: Amesos, Anasazi, AztecOO, Belos, CTrilinos, Didasko, Epetra, EpetraExt, FEI, ForTrilinos, Galeri, GlobiPack, IFPACK, Ifpack2, Intrepid, Isorropia, Kokkos, Komplex, LOCA, Meros, Mesquite, ML, Moertel, MOOCHO, NewPackage, NOX, Optika, OptiPack, PAMGEN, Phalanx, phdMesh, Piro, Pliris, PyTrilinos, RTOp, Rythmos, Sacado, Shards, STK, Stokhos, Stratimikos, Sundance , Teko, Teuchos, ThreadPool, Thyra, Tpetra, TriKota, TrilinosCouplings, TriUtils, WebTrilinos, Zoltan

Trilinos Packages

- Basic Linear Algebra Libraries
- Preconditioners
- Solvers
- Nonlinear, Transient, and Optimization Solvers
- Eigensolvers
- Automatic Differentiation
- Domain Decomposition
- Mortar Methods
- Partitioning / Load Balancing
- Mesh Generation, Improvement, and Adaptivity
- Discretization Utilities
- PDE discretization tools
- Abstract Interfaces and Adapters
- Instructional
- Utilities

Trilinos (cont)

I do not personally use Trilinos directly

When I do use it, I use it underneath Dolfin (FEniCS)

FEniCS

The FEniCS Project is a collection of free software aimed at automated, efficient solution of differential equations. The project provides tools for working with computational meshes, finite element variational formulations of PDEs, ODE solvers and linear algebra.

FEniCS has C/C++ and Python interfaces (Parallel HPC Python FTW). It is almost ridiculously easy to use. I am using it on top of PETSc and Trilinos with both Python and C++ interfaces.

FEniCS Poisson Equation with Dirichlet BC's and viz

```
from dolfin import *

# Create mesh and define function space
mesh = UnitSquare(32, 32)
V = FunctionSpace(mesh, "Lagrange", 1)

# Define Dirichlet boundary (x = 0 or x = 1)
def boundary(x):
    return x[0] < DOLFIN_EPS or x[0] > 1.0 -
 DOLFIN_EPS

# Define boundary condition
u0 = Constant(0.0)
bc = DirichletBC(V, u0, boundary)

# Define variational problem
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("10*exp(-(pow(x[0] - 0.5, 2)
+ pow(x[1] - 0.5, 2)) / 0.02)")
g = Expression("sin(5*x[0])")
a = inner(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds

# Compute solution
problem = VariationalProblem(a, L, bc)
u = problem.solve()

# Save solution in VTK format
file = File("poisson.pvd")
file << u

# Plot solution

plot(u, interactive=True)
```

SuperLU

SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library is written in C and is callable from either C or Fortran.

The library routines will perform an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices.

I use SuperLU underneath PETSc and FEniCS (i.e. through interfaces NOT directly). There was a release in 2010.

Nektar++

Nektar++ is an open source software library that provides a toolbox of data structures and algorithms which implement the spectral/hp element method, a high-order numerical method yielding fast error-convergence

Nektar++ is the continuation and adaptation of the Nektar flow solver. As opposed to its predecessor which focused on solving fluid dynamics problems, Nektar++ is implemented as a C++ object-oriented toolkit which allows developers to implement spectral element solvers for a variety of different engineering problems.
I used Nektar++ in the past. There is a development parallel layer, it is heavy on C++ (no fortran) and can be problematic to compile. Use with caution. They do offer top-notch support though

Domain Specific Packages

- Obviously we cannot go into detail about every particular package
- A great list is maintained at the TeraGrid software respository here: http://hpcsoftware.teragrid.org/Software/user/index.php
- The following is the list of packages per application domain with minimal information as to what the packages are used for
- You can easily find packages for your application with a simple internet search

Astronomy, Astrophysics, Cosmology

- CLHEP: Class library for High Energy Physics Description
- GADGET: cosmological N-body/SPH
- Cactus/Carpet: numerical relativity, cosmology, black holes
- ChaNGa: Charm N-body Gravity
- Enzo: AMR, grid-based hybrid code, cosmological structure formation
- FISH: 3D parallel MHD code for astrophysical applications
- ZEUS:several different numerical codes for astrophysical gas dynamics in two- and three-dimensions
- RAMSES hybrid, N-body and hydrodynamical code, dark matter component and the baryon gas for analyzing the structure and the distribution of galaxy clusters

Astronomy, Astrophysics, Cosmology

- HERACLES 3-D equations of radiative transfer coupled to hydrodynamics
- ASH 3-D MHD simulations in spherical geometry, turbulence in solar and stellar interiors.
- JUPITER a mutidimensional astrophysical hydrocode
- Many, many others...

Computational Chemistry

- CASTEP: simulating electronic relaxation to ground state for metals, insulators, or semiconductors
- CHARMM: general purpose molecular mechanics, molecular dynamics and vibrational analysis packages for modelling and simulation of the structure and behavior of molecular systems from an individual organic molecule to a large oligomeric protein in its solvent environment
- DMol3: first-principles (ab initio) quantum chemistry software package that performs the following basic tasks based DFT: Single-point energy Geometry optimization Frequency Transition-state search Geometry optimization and frequency Transition-state search and frequency Gradient Molecular dynamics Simulated annealing

Computational Chemistry (cont)

- DOCK: solves the problem of "docking" molecules to each other
- GAMESS: ab initio quantum chemistry to compute wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF, with CI and MP2 energy corrections available for some of these
- Gaussian: general purpose ab initio electronic structure package that is capable of computing energies, geometries, vibrational frequencies, transition states, reaction paths, excited states, and a variety of properties based on various uncorrelated and correlated wavefunctions

Computational Chemistry (cont)

- Gromacs: perform molecular dynamics for systems with hundreds to millions of particles. Designed for biochemical molecules like proteins and lipids, but it can also be used for research on non-biological systems, e.g. polymers
- Molden: displaying Molecular Density from the Ab Initio and the Semi-Empirical packages, such as GAMESS-UK , GAMESS-US, GAUSSIAN, MOLPRO, and Mopac/Ampac
- MOLPRO: ab initio programs for molecular electronic structure calculations with extensive treatment of the electron correlation problem through the multiconfiguration-reference CI, coupled cluster and associated methods.
- NWChem: computational chemistry package capable of performing calculations of molecular electronic energies and analytic gradients using ab-initio methods

Computational Chemistry (cont)

- ADF: density functional theory package based on Kohn-Sham approach to solving electronic structure problem
- AMBER: general purpose molecular mechanics and molecular dynamics packages with capability to compute free-energy changes
- BLAST: a set of similarity search programs designed to explore all of the available sequence databases regardless of protein or DNA
- CASTEP: simulating electronic relaxation to ground state for metals, insulators, or semiconductors
- CPMD: a plane wave/pseudopotential implementation of Density Functional Theory for ab-initio molecular dynamics.
- LAMMPS: classical molecular dynamics simulation code designed to run efficiently on parallel computers
- NAMD: A parallel molecular dynamics program for UNIX platforms designed for high-performance simulations in structural biology

Computational Chemistry (cont)

- Qchem: object oriented quantum chemistry program capable of computing energies, geometries, vibrational frequencies, transition states, reaction paths, excited states, and a variety of properties based on various uncorrelated and correlated wavefunctions
- Siesta: is both a method and its computer program implementation, to perform electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids.
- VASP: ab-initio quantum mechanical molecular dynamics simulation package using pseudopotential and plane wave basis set
- vmd: molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting
- SHELX: programs for crystal structure determination from single-crystal diffraction data

Computational Biology
Duplicates with chem removed

- Abyss: Assembly By Short Sequences - a de novo, parallel, paired-end sequence assembler
- BOXSHADE: Program for pretty-printing multiple alignment output
- Clustalw: Fully automatic program for global multiple alignment of DNA
- FASTA: DNA and Protein sequence alignment software
- GARLI: Heuristic phylogenetic package using GTR model
- IM: Isolation with Migration - population divergence modelling
- mpiBLAST: Parallel BLAST program using NCBI Blast as the core.
- MrBayes: Bayesian Estimation of phylogeny

Computational Biology (Cont)

- PHYLIP: inferring phylogenies (evolutionary trees)
- PhyloGibbs: Transcription factor binding sites finder
- Raster3D: set of tools for generating high quality raster images of proteins or other molecule
- RAxML: Maximum Likelihood-based inference of large phylogenetic trees
- Rosetta: prediction and design of protein structures, protein folding mechanisms, and protein-protein interactions
- Simwalk2: haplotype, location score, identity by descent, and non-parametric statistical analyses on any size of pedigree
- SOAPdenovo: short-read assembly method that can build a de novo draft assembly for the human-sized genomes
- velvet: Sequence assembler for very short reads

Fluid Flow and Finite Elements

- FIDAP: incompressible/compressible fluids, laminar/turbulent flows, two-phase flows, newtonian/Non-Newtonian/visco-elastic fluids, porous media, steady/transient, buoyancy, swirling flows, free surface (commercial)
- ANSYS: CFD, if you have the money for a license
- COMSOL: ditto
- Gambit: A grid generation package for structured and unstructured grids for CFD
- GASP: Compressible Flow Solver for Aerothermodynamic applications
- OpenFOAM: CFD and much more, if like most of us you prefer not tp spend money on a license
- OpenLB: CFD via lattice boltzmann
- Abaqus: general purpose finite element analysis package

Final Thoughts

- Chances are someone has tried to do something close to what you are trying to do in HPC, plenty of brilliant people out there
- Chances are they are offering their work (packages, documentation, support) for free for you to use
- Make sure you save yourself development time and run time by having a look at what others have done
- Please make sure to contribute: offer up your work for others to use and offer support or help write documentation whenever possible
- Please keep in mind that you should reference other people's work if you use it in yours (adding a citation or an acknowledgement costs nothing).