Scientific Computation (at NC State)

Gary Howell, HPC/OIT

NC State University

gary_howell@ncsu.edu

Science

- Theoretical
- Experimental
- Computational

Science

- Theoretical Simplify to get ideal cases
- Experimental
- Computational Last 50 years – leave in as much complication as the computer can deal with

Science

- Theoretical Simplify to get ideal cases
- Experimental
- Computational Last 50 years – leave in as much complication as the computer can deal with
- Algorithm Development?

Efficiency

- Ease in setting up the computation
- Using resources efficiently

Efficiency

- Ease in setting up the computation High level languages are quick for programming .. can get the answer same day ?
- Using resources efficiently Can get a better quality answer from the same computational resource .. price of much work for the programmer ?

Command Line vs. Mouse

- Point and Click computing
- Command Line

Command Line vs. Mouse

- Point and Click computing Can only do what is already set up
- Command Line linux or cygwin from Windows or darwin from a Mac. Once you've laboriously constructed a sequence of commands, you turn it into a file and just type the name of the file.
- Need to use a native editor to linux, cygwin, darwin, as Windows dos files have an extra few symbols

High Level Languages

Can get to an answer quickly .. more powerful than a calculator ..

- Maple, Mathematica
- Matlab,
- Sas
- Python (numpy, scipy)

High Level Languages

Can get to an answer quickly .. more powerful than a calculator ..

- Maple, Mathematica Reduce
- Matlab, Octave
- Sas R
- Python (numpy, scipy) color are open source

High Level Languages

Can get to an answer quickly .. more powerful than a calculator ..

- Maple, Mathematica Reduce,Sage includes all the red colored languages
- Matlab, Octave
- Sas R
- Python (numpy, scipy) color are open source

Open Source codes are good when you move on and find that you might have to purchase your own license

Some more specialized packages

- finite elements
- fluids
- chemistry codes
- weather and oceanographic
- genetics codes

Some more specialized packages

- finite elements Abaqus, ANSYS
- fluids CFX
- chemistry codes Gaussian, VASP, Amber
- weather and oceanographic
- genetics codes

Some more specialized packages

- finite elements Abaqus, ANSYS Elmer
- fluids CFX OpenFOAM
- chemistry codes Gaussian, VASP, Amber lammps, nwchem, abinit, espresso, ..
- weather and oceanographic wrf wrf-chem, cmaq, ROMS ..
- genetics codes Mr. Bayes, BLAST

Lower Level Languages

- Fortran 77 – slightly higher level than C
- C – more versatile than Fortran

Lower Level Languages

- Fortran 77 – slightly higher level than C , Fortran 90
- C – more versatile than Fortran , C++

Fortran 90 and C++ are more "modern". Have features that allow large packages to be written and maintained. But are also more complicated than C or Fortran 77

Lower Level Languages

Fortran and C codes need to be compiled. Adds an extra step to the code writing process. Allows loops to execute efficiently.
For languages like R and Matlab-Octave, we can write the code a line at a time and execute it, good for verifying it as you go. For Fortran or C you have to compile then execute. If you want to step through the code, you attach a debugger to the compiled code.

Libraries

- dense linear algebra
- sparse linear algebra
- FFT
- numeric libraries (e.g. evaluate Bessel function, generate random number, compute integrals)

Linking to Libraries

To compile codes into executables in 2 phases.

- compile .f or .c (source code files) to .o object files by for example

  ```
  pgf90 -c foo.f
  ```

- Libraries are collections of .o files with an extension .a Fortran modules have extension .mod. The .a, .o and .mod files are linked together to get an executable "foo" by (for example)

  ```
  pgf90 -o foo *.o *.a *.mod
  ```

  Then we can run the executable by

Fast High level languages

If the high level code, e.g. Matlab as a wrapper for LAPACK, is spending most of its time in calls to LAPACK, it's pretty efficient.
You can sometimes get the high level code to call C or Fortran subroutines .. e.g. mex files in Matlab.

Converting Code

You can transition high level code to low level by rewriting it so that lines of the code correspond to library calls.
For example, "y =A*x" in Matlab corresponds to a call to the DGEMV subroutine (from BLAS) called in a Fortran or C code.

Parallelization Libraries

- Shared Memory
- Distributed Memory

Parallelization Libraries

- Shared Memory Processors can see the same variables .. access the same RAM, e.g. multi-core machines. Many NCSU computational nodes have 8 cores.
- Distributed Memory Processors are joined by a network, e.g., ethernet, programmer must specify what information is to be sent over the network. Hundreds of computational nodes are available.

Shared Memory

- Pthreads, Cilk, OpenMP, UPC
- OpenMP is particularly easy to use. Insert Pragmas in a Fortran or C code to run a loop in parallel. The code will still run in serial mode. See the short course on-line at hpc.ncsu.edu (under Courses).
- All our new HPC blades have at least 8 cores. Some have 16 cores and 128 GBytes of RAM. Anyone who can write a code in Fortran or C can try OpenMP at a fraction of the effort of writing the code.

Shared Memory Libraries

One way to take advantage of shared memory is just to call shared memory libraries.

- dense linear algebra
- sparse linear algebra
- FFT

Shared Memory Libraries

One way to take advantage of shared memory is just to call shared memory libraries.

- dense linear algebra – acml and mkl BLAS and LAPACK libraries. use 64 bit integer versions for large problems
- sparse linear algebra – SuperLU multi-threaded version
- FFT – Multi-thread version of fftw

Distributed Memory via MPI

MPI –Calls to Message Passing Interface library routines "mail" i data from one processor to another. Accomplished via MPI library calls inserted into Fortran or C (or R or Matlab or Java) routines.

Most computer codes which use many processors use MPI to communicate between processors. See short course at hpc.ncsu.edu for a short introduction.

Difficulties of MPI

- Messages are slow compared to computation. Starting a communication is the same time as $O(10^5)$ floating point adds or multiplies point adds or multiplies (flops). It takes only a few hundred flops to equal a fetch of a number from RAM (memory).
- Can require substantial reorganisation of code and careful distribution of data.
- Debugging can be difficult.

Advantages of MPI

- It's standard, available everywhere, will still work on the next comuter cluster you use.
- We (NC State) have thousands on computational nodes on-line. You can get an account to use them. (get your advisor to go to hpc.ncsu.edu and request an account).

Many MPI codes already exist

For example, the specialized packages from a previous slide.

- finite elements – Abaqus, ANSYS (Elmer)
- fluids – CFX (OpenFOAM)
- chemistry codes – Gaussian, VASP, Amber (lammps, nwchem, abinit, espresso, ..)
- weather and oceanographic – (wrf wrf-chem, cmaq, ROMS)
- genetic tree codes (Mr. Bayes, BLAST)

Distributed Memory Libraries

And of course many distributed memory libraries exist.

- dense linear algebra
- sparse linear algebra
- FFT

Distributed Memory Libraries

And of course many distributed memory libraries exist.

- dense linear algebra – SCALAPACK
- sparse linear algebra –PETsc or Trilinos
- FFT –distributed version of fftw

Running parallel codes on a cluster

To run on in serial, in linux we might type

```
./foo
```

and then wait for the prompt to come back. But if everyone does this on one of the cluster login nodes, then it locks up and we have to reboot. Generally, you're sharing the login node with lots of other users and really it's nothing but a "souped up PC" so you'd be better off running the code on your laptop.

Running jobs on a cluster

So on any cluster, you instead submit jobs to run in a queue. On our cluster you submit a job to the queue by

```
bsub < bfoo
```

where bfoo is a file you wrote specifying how much time the job will need, how many processors, and names of the files where output will be written.

The job or jobs may run a few days. You don't have to stay logged in, but can monitor the progress of the output files.

Running jobs on a cluster – II

For a serial job one line of bfoo might be "

```
./foo
```

(same as running from a command line.) For a parallel job the corresponding line might be

```
mpiexec_hydra ./foo
```

See the HowTo on hpc.ncsu.edu

GPUs are fast

Graphical Processing Units were developed to run games. They have high bandwidth and lots of simple processors. They can be programmed with

- OpenCL (standard)
- CUDA – specific to NVIDIA

Both are extensions to C. OIT HPC does not support GPUs, but there is some expertise in the CS department. For Example, Frank Mueller has a GPU cluster.

Disadvantages to GPUs

- Lower level programming.
- Not as many libraries available
- Better at single precision arithmetic.
- Arithmetic is substandard (don't round .. chop .. ?)
- Often only one portion of the code can run on the GPU .. poor bandwidth back to the main processor .. limited memory on the GPU

Distributed memory with attached G

The world's fastest computers typically have thousands of multi-cores (shared memory) processors with attached GPUS.
So if you want to scale your Matlab code to run really fast ?

Scaling high level code to run fast

Do as much as possible while the code is still in Matlab.

- You can insert MPI calls from the bcMPI library, so then you have parallel matlab (or parallel octave) code.
- Figure out how to do the matlab commands in terms of libraries (e.g. matrix operations are from LAPACK or BLAS or ..may run even faster if they are call from a GPU library).
- Figure out how to partition the data and see if you can manage that within Matlab. Finally you can convert it to Fortan or C.