Practical High Performance Computing

Donour Sizemore

July 21, 2005

2005 ICE

Purpose of This Talk

- Define High Performance computing
- Illustrate how to get started

Preliminaries

What *is* high performance computing?

- Doing a lot of work
  - Computations that are memory, arithmetic, and/or communication intensive.
- This does not imply "big" computers.

...more formally

- For a given task, the user wants to minimize time.
- The performance bound is almost always floating point arithmetic.

The Environment Can Be a Shock

- In most cases this means using some Unix variant
- Text-based tools that are optimized for function, not ease of use.
- The initial time investment can be very large.

The Common Case

You have AMPL, Stata, or Matlab code that is too slow. For example:

- You're solving your model to 3 digits but want 6
- You're looking at 100 time periods, but want 100k.
- You were supposed to graduate in June, finish your thesis!

You Have a Number of Options

1. Port your code to C
2. Port your code to Fortran
3. Rewrite your core computations using vectorized routines
4. Hire an undergraduate to do all of the above.

...and the Answer is:

None of these.

- Before doing anything, profile!
- Be able to identify the lines of code that constitute the most processing time
- Isolate the time critical regions.

Now You Can Proceed

- Armed with this information you can make meaningful performance enhancements
- It far easier to get programming assistance when you can construct small examples of your problem

The Problem with Modeling Languages and Mathematical Programming Packages

- The built-in functions buy you a lot, and you pay for it.
- The focus is on what to compute, not how to compute it.
- For better speed,you often have to give this up.

Getting Cozy with the Hardware

To alleviate some of these issues, you can move to programming system with less overhead (closer to the actual hardware).

- Selective rewriting of compute kernels
- Introduction of special purpose libraries

Popular Mathematical Libraries

- blas, atlas, lapack – www.netlib.com
- acml (www.amd.com), mkl (www.intel.com)

This is often enough

- It's quite rare that an entire program needs to be rewritten.
- You still have the the high-level routines available for administrative tasks.

However, sometimes it isn't enough.

Parallel Computing

- You have N units of independent work, and P processing units.
- Split the work over the P processors
- Do the administrative book-keeping to build the final answer

Distributed vs. Shared Memory Models

1. Shared memory: threads, OpenMP
2. Distributed memory (clusters): MPI
3. DBPP - http://www-unix.mcs.anl.gov/dbpp/

Distributed Memory Benefits

- Cost effective for a large number of CPUs
- Relatively small shops can build them easily
- Can grow to meet demands (or funding)
- Flexibility to partition the hardware for different jobs

A Bit More About MPI

- Allows for several types of problem decomposition.
- Very good support from the scientific computing community.

MPI Example: Loop decomposition 1

```
do i = 1, K
  call some_work
end
```

MPI Example: Loop decomposition 2

```
start = K/P *  rank
end   = K/p * (rank + 1)

do i = start, end
  call some_work
end
```

MPI Example: Loop decomposition 3

Do the example for real!

Places To Go To Learn More

MPIch: http://www-unix.mcs.anl.gov/mpi/mpich/
open-mpi: http://www.open-mpi.org/

Where Can I Run These Things?

- Small clusters have popped up many research institutions.
- Larger clusters are much more accessible than people realize.
- Latest PC desktop and workstations

Parallelism in the Desktop

- Multicore desktops are available now.
- All major processor manufacturers have development paths migrate their chips to multicore.
- Operating system support is mature.
- Read the white papers at www.intel.com, www.amd.com

What about the really BIG machines

- National Labs
- Major Universities, worldwide

Random Points

- Learning about the machine has large payoffs
- Hug a programmer.

Wrapping up

- We've seen how to go about maximizing code for speed
- A few techniques have been introduced; selective rewriting, MPI
- I've provided some external links to direct everyone after this conference.