PTP Synchronized Projects: Future Directions

John Eblen

September 14, 2013

Supporting “Other” Languages

- Traditionally support C/C++/Fortran
- Generic sync projects now allows for any language
- Additional support for specific languages commonly used for HPC?
- If so, which ones?

Main Contenders for HPC

- Python
  - Eclipse support: PyDev
  - Packages: mpi4py, SciPy, NumPy
  - Example: PyClaw (hyperbolic PDE solver)
- R
  - Eclipse support: StatET
  - Packages: Rmpi, snow, snowfall
  - Example: pbd-r (Programming with Big Data in R)
- Usually C underneath
- Can be quite difficult to setup

Other Contenders

- Matlab
  - MPI support, well-known and available
  - Requires a license
- Java
  - Built-in threading and concurrency support
  - Still has a stigma of being slow
- Chapel and X10: Still works in progress
- Perl and Ruby: Not much found...

Four Languages to Watch

- Clojure: A modern Lisp dialect for the JVM
- D: C++ successor?
- Julia: Matlab and R successor?
- Scala: Java successor?

Python in Parallel

- Multithreading
  - GIL (Global Interpreter Lock) limitation
  - Only one thread can run interpreter at a time
  - Programs can release GIL
  - NumPy does this for array operations (C = A + B)
  - IO operations also release GIL while waiting
- Symmetric Multiprocessing (SMP)
  - Avoids GIL by having multiple interpreters
  - Shared memory environment only, though
- Massive parallelism
  - Mpi4py and several others

R in Parallel

- No parallelism by default
- Memory problems
- Several solutions developed
  - snowfall
  - snow
  - rMPI

Cheap Parallelism

- Common scenario: Employ a supercomputer to run a single, non-parallel program on different data files
- Difficult to find examples of true, massively parallel Python and R programs
- Why?
  - Simple analysis or parsing scripts written by non-programmers (e.g. scientists)
  - Scripting languages are more common among non-expert users

Proposal 1

- Add PTP support for “embarrassingly parallel” programs
- Option for automatically-generated launcher script
- Simply another layer
  - mpirun -np 1024 <program> <args>
  - mpirun -np 1024 <launcher> <program> <args>
- Problem: launcher needs domain knowledge to map MPI rank to specific arguments
  - Allow launcher script to be editable?
  - Use “job number” variable in arguments?
  - Other?
- How to reduce?

Proposal 2

- Support easily running jobs on login nodes
  - Produce input files
  - Reduce results
  - Analyze results
- Options already discussed
  - Improve “Run as” menu option
  - Remote command line
- Generic support for different build systems

Build System Support

- Easier running of remote jobs is first step
- Intercept build requests
- Need automatic detection or project type
- Provide options somehow (e.g. cmake in-source build vs. out-of-source build)

CMake Support

- Simplest approach
  - Run CMake to generate make files
- Eclipse CDT4 Generator creates Eclipse CDT project from CMake projects
- Roland's idea: Enhance to create CDT build configurations
  - Provides a way to build project
  - Could provide discovery information!
- Could we tie this into Eclipse?
- General idea of creating build configurations from a build system?

Other Enhancements

- Easier setup, creation of synchronized projects
- Allow it to be a general Eclipse facility
- Remote indexing for specific languages
- Remote debugging for specific languages

References and Acknowledgments

- *Parallel programming with numpy and scipy*. http://wiki.scipy.org/ParallelProgramming
- Ryan R. Rosario. July 27, 2010. *Taking R to the Limit (High Performance Computing in R)*.
- Roland Schulz
- Dr. David Hudak