COSC 462

OpenMP Introduction

Piotr Luszczek

October 4, 2017

OpenMP

- OpenMP is a standard
  - Freely available at www.openmp.org
  - Open Multi-Processing
- Supported languages
  - C
  - C++
  - Fortran
- Enabled during compilation
  - Intel, GNU, LLVM, …
    - -fopenmp
    - -fopenmp=gomp
    - -fopenmp=iomp
  - Visual Studio
    - /openomp

- OpenMP is supported by modern compilers
  - GNU gcc, gfortran
    - GOMP
  - LLVM clang
    - In progress
  - Intel icc, ifc
    - iomp library
  - IBM xlc, xlf
  - Cray compiler
  - PGI compiler (now part of NVIDIA): pgcc, pgfortran
  - Microsoft Visual Studio
    - Not in Express version in 2010

OpenMP is a Threading Interface

- There are many thread APIs
  - POSIX threads
  - WinThreads
  - Intel Threading Building Blocks (TBB)
  - Cilk++
  - OpenMP
  - Java threads
  - ...
- Most of them are free or nearly so
- A lot of documentation available
  - Source code
  - Examples
  - Manuals
  - Tutorials
  - ...

Basic Premise of OpenMP

- Make development of threaded code
  - Easy
    - Threads are created, deployed, and destroyed with few code changes
  - Incremental
    - Only some parts of the code may need threading
    - By default, the rest of the code runs sequentially
  - Expose complex features when necessary
    - Direct locking of mutex locks
    - Accessing vector units
    - Using accelerators

OpenMP is NOT...

- Not meant for distributed memory parallel systems (by itself)
  - It is often combined with MPI
- Not implemented identically by all vendors
  - Despite a lot of code reuse and sharing of ideas
- Not guaranteed to use shared memory efficiently
- Not required to check for:
  - data dependencies
  - data conflicts
  - race conditions, or
  - deadlocks
- Not required to check conformance of user code
- Not provide compiler-generated automatic parallelization and/or directives to the compiler to assist such parallelization
- Not providing synchronous I/O to the same file when executed in parallel
  - The programmer is responsible for synchronizing I/O

History of OpenMP

- In the early 90's, vendors of shared-memory machines supplied similar, directive-based, Fortran programming extensions.
- The user would augment a serial Fortran program with directives specifying which loops were to be parallelized.
- The compiler would be responsible for automatically parallelizing such loops across the SMP processors.
- Implementations were all functionally similar, but were divergent.
- First attempt at a standard was the draft for ANSIX3H5 in 1994.
  - It was never adopted, largely due to waning interest as distributed memory machines became popular.
- The OpenMP standard specification started in the spring of 1997, taking over ANSI X3H5
  - Newer shared memory machine architectures started to become prevalent.
- Led by the OpenMP Architecture Review Board (ARB).

Goals of OpenMP

- Standardization
  - Provide a standard among a variety of shared memory architectures/platforms
- Lean and mean
  - Establish a simple and limited set of directives for programming shared memory machines
  - Significant parallelism can be implemented by using just 3 or 4 directives.
- Ease of Use
  - Provide capability to incrementally parallelize a serial program, unlike message-passing libraries which typically require an all or nothing approach
  - Provide the capability to implement both coarse-grain and fine-grain parallelism
- Portability
  - Supports Fortran (77, 90, and 95, 2003), C, and C++
- Public forum for API and membership

OpenMP Release History

- 1997
  - Version 1.0 for Fortran
- 1998
  - Version 1.0 for C/C++
- 1999
  - Version 1.1 for Fortran
- 2000
  - Version 2.0 for Fortran
- 2002
  - Version 2.0 for C/C++
- 2005
  - Version 2.5
- 2008
  - Version 3.0
- 2011
  - Version 3.1
- 2013
  - Version 4.0
- 2015
  - Version 4.5