Programming Models and Languages Session - Track 2

Rusty Lusk, ANL

Rajeev Thakur, ANL

Bill Gropp, UIUC

Tim Mattson, Intel

Tim Warburton, Virginia Tech

Brad Chamberlain, Cray

Sanjay Kale, UIUC

Kathy Yelick, LBNL and UC Berkeley

Programming Models

- A programming model is how you *think* about what the computer is doing when it executes your program.
  - This may not be what the computer *is* doing.
- For example, in a sequential programming model, you write the program *as if* the computer is executing your code one statement at a time, in the order you have written.
  - With today's architectures and compilers, featuring instruction-level parallelism and out-of-order execution, what really happens will be quite different.
- Programming models differ greatly in their *level of abstraction* (how far the the programmer's mental model is from what the computer actually does in response to the program (as partially directed by the compiler.
  - High-level (e.g. Prolog, Lisp, ML): portability, conciseness, "ease of programming"
  - Low-level (e.g. assembly language): performance

The Next Three Days

- In this track we will talk about a range of parallel programming models (between low-level and high-) and how to use the programming systems (libraries and languages) that implement them.
- The message-passing model: Processes with separate address spaces communicate with explicit messages.
  - System: MPI (this afternoon and tomorrow)
- The shared-memory model: A sequential programming model is augmented with hints to the compiler about what can be done in parallel.
  - System: OpenMP (Wednesday)
    - OpenMP also has extensions beyond the sequential model
- A hybrid model: Shared-memory parallelism combined with message passing
  - System: MPI + OpenMP (Tuesday afternoon)

The Next Three Days (cont.)

- The accelerator model: Extra hardware with limited (but very fast and parallel) capabilities is accessible to the “main” process.
  - Systems: CUDA, Open ACC, OCCA (Thursday morning)
- Higher-level approaches (Thursday afternoon):
  - System: Chapel (A sort of high-performance, parallel Python)
  - System: Charm++ (Task-based parallelism)
  - System: UPC++ (Partitioned Global Address Space model)
  - System: ADLB: (A simple task-based load-balancing system)