Whither Programming Models?

Michael L. Scott

University of Rochester

Panel session
Workshop on Communication and Middleware for
Parallel Programming Models
Held in conjunction with IPDPS
Ft. Lauderdale, FL
April 2002

How did we get here?

- Tons of work on parallel languages and models in the late 70s and 80s
- Some if it bad, but much of it good, from a conceptual point of view
- But nobody offered a *really* big win, and we couldn't get good performance with good models, so we settled for good performance with poor models

And where exactly are we?

- ✓ Pthreads are sort of ok
- ✓ MPI and OpenMP are not!
  - too hard to use
  - not applicable to non-HPC-style systems

- → The challenge: come up with something better that isn't too different for the languages we already use
  - maintain programmer comfort
  - leverage existing tools (compilers in particular)

Recommendation #1

- ✓ Recognize that both shared memory and message passing have a place
  - Shared memory good for passive communication
  - Message passing good for active communication
  - Much of the time you can use either (matter of taste)
  - Sometimes you *need* one or the other; witness
    - events in shared memory systems
    - put() and get() in message passing systems

Recommendation #2

- ✓ Make the easy stuff easy
  - coherent shared memory (S-DSM) for
    - fast prototyping
    - non-performance critical code
  - global address space with put() and get()
    for performance tuning of shared memory code
  - 2-ended message passing for active communication

  - These models *can* comfortably co-exist within a single application

Recommendation #3

- ✓ Don't embed HPC assumptions
  - parallel computing is going mainstream -- think immersive 3-D video games
  - can't afford to assume
    - fixed number of processes
    - single process per processor
    - uniprogrammed workloads
    - homogeneous hardware
  - must accommodate
    - availability / replication
    - fault tolerance / recovery
    - language and machine heterogeneity
    - geographic distribution
    - persistence
  - potential big wins for grid computing *today*

Short term:

- ✓ Combine the convenience of S-DSM with the tunability of put() / get() and explicit messages

Longer term:

- ✓ Encourage research in more speculative models, and in aggressive parallelizing compilers