Hybrid Programming Panel

Panel, P2S2 Workshop, ICPP 2010

Allen D. Malony

malony@cs.uoregon.edu

Dept. Computer and Information Science

Performance Research Laboratory

University of Oregon

Hybrid Programming – What does it mean?

- Hybrid implies a combination (mixing) of different elements
- Hybrid programming – the use of different programming:
  - Techniques and methods
  - Languages? algorithms?
  - Parallelism / computational models
- Why use hybrid programming?
  - Three factors: *necessity*, *productivity*, *performance*
  - Explicit and implicit hybrid programming
- Does hybrid programming require heterogeneity?
- Hybrid programming versus high-level programming model?
- What about the rest of the hybrid programming environment?

Hybrid programming is easy compared to this ...

The Netherlands Bicycle Band, Zurich Police Band Festival, 2010

Is MPI Hybrid Programming?

- Yes!
  - "Programming" of communications
  - Mixed with programming of "rank-level" computation
- Necessity?
  - Need some way to send messages somehow
- Performance?
  - Can optimize in various ways (e.g., asynchronous)
- Productivity?
  - Supports effective computation model (e.g., SPMD)
- Mix in other hybrid elements
  - Multi-threading at rank level (e.g., OpenMP)
  - Libraries of various sorts

Other Hybrid Programming Alternatives

- Use of multiple languages for different purposes
  - Example: Python as a coordination / controller language
- Use of multiple parallel computation models
  - task parallelism + data parallelism
  - components
- Use of parallel libraries implementing high-level functionality
  - Parallel data management
  - Domain-specific algorithms
- Use of runtime and system-level functionality
  - Different forms of scheduling
  - Memory management and I/O
- Regard hybrid programming as a computational model

Heterogeneous Parallel Systems

- Heterogeneous parallel systems motivate hybrid programming
- Availability of heterogeneous hardware technology
  - Multicore processors, manycore accelerators
  - High-performance engines, special purpose components
- Performance is the main driving concern
- Role of hybrid programming
  - Necessity?
    - Heterogeneous hardware may require different programming
  - Performance?
    - What is required to enable and realize optimizations
  - Productivity?
    - Productivity objectives and relative importance (e.g., portability)

Tension Between Performance and Productivity

- Panel chair sets up as a contrast between a "single programming model" and a "hybrid programming model"
- Single programming model everywhere (high-level model)
  - High-level (rich) expression so application works unchanged
  - Necessarily implies some form of transformation system
- Hybrid programming model
  - Applications deal with heterogeneous architecture explicitly
  - Implies modification of program for particular scenarios
- Which approach is best for performance?
- Which approach is best for productivity?
- Are these really in tensions vis-à-vis programming model?

Hmm, seems like we have been down this road ...

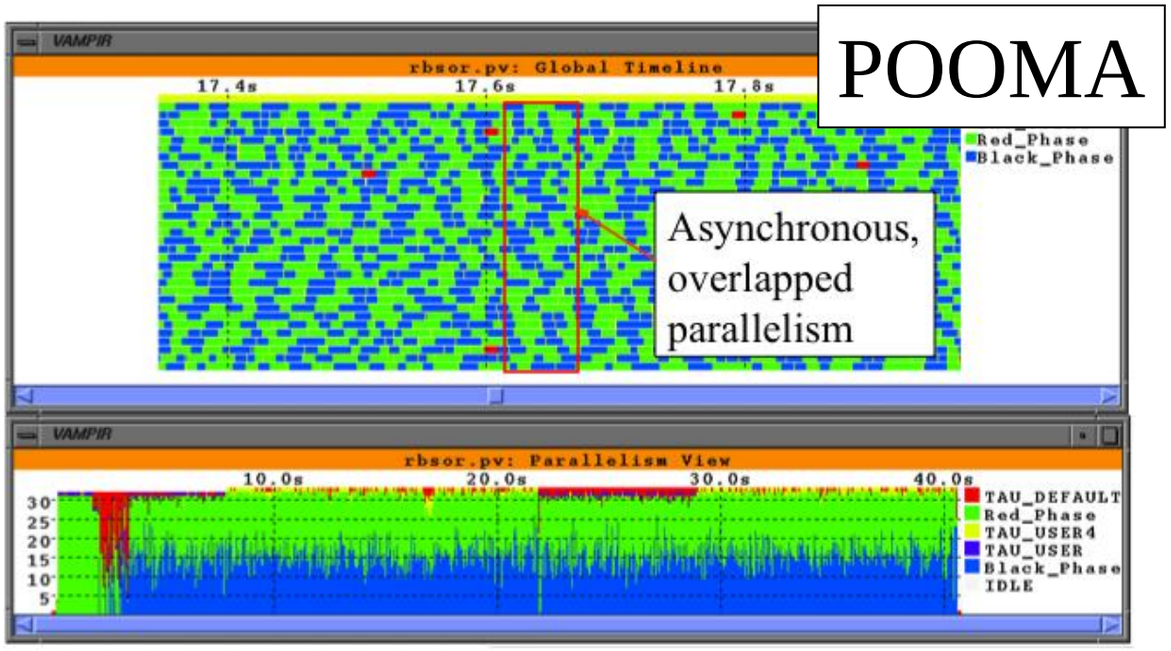
- High-Performance Fortran (HPF)
  - Data decomposition expressed at a high level
  - HPF compilers generate compute and communication code
  - Transformations attempt to optimize communication
  - Perceived as not able to produce high performance reliably
- POOMA
  - High-level object-oriented language systems
  - C++ with array objects
  - Compiler optimization with parallel expression templates
  - Runtime optimization with asynchronous threading RTS
- Both are gone now

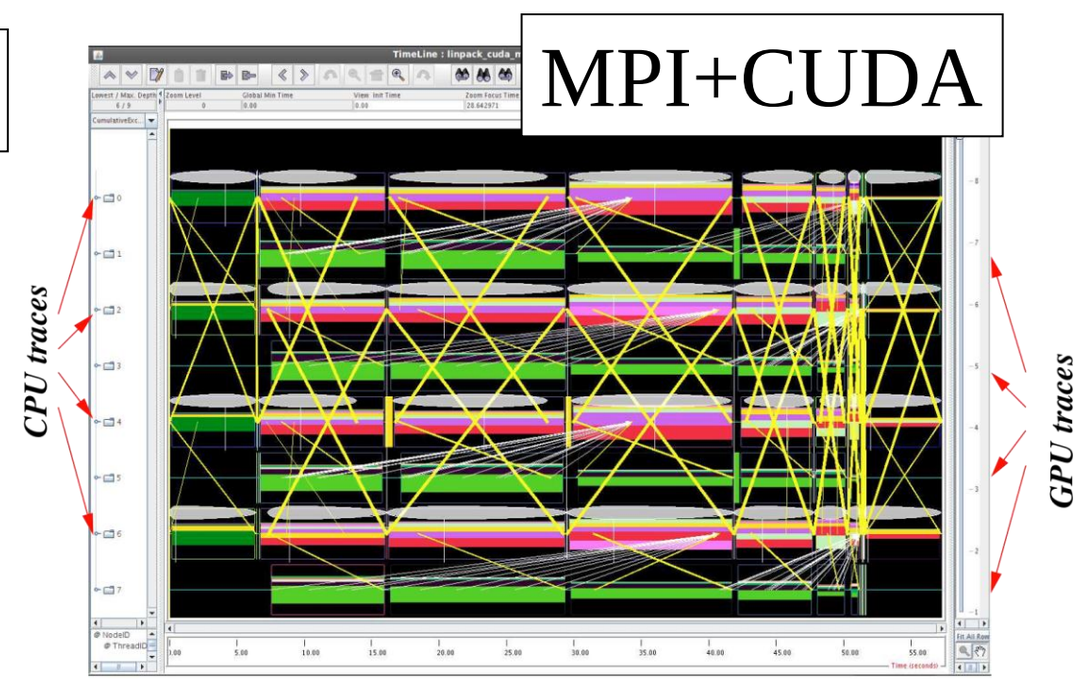
Panel Position

- Need programming systems that support the different (general) hybrid aspects of parallel systems
  - Regard as a multi-level programming framework (system)
- Need flexibility in the programming system for tradeoffs in abstraction and "exposure" of functionality/controls/...
  - Addresses both productivity and performance concerns
  - Allow for both expressivity and targetability
- High-level programming (language) models are challenged
  - Requires powerful transformation support for heterogeneity
- Consider high-level programming framework
  - Supports a programming /computational model
  - Uses language annotations, layered RTS, and libraries
  - Examples: Charm++, HMMP

Hybrid Programming and Performance Tools

- Parallel performance tools should support the programming model / computational model used by the application
- Important to map performance data
  - Associate performance measurements to program semantics
  - Use context to identify thread of execution
  - Use events to relate to abstractions through model levels
  - Capture interactions between events





Questions to Panelists

- While hybrid architectures are upon us, are we equipped to deal with them?
  No, but we will be
- Has the time for hybrid programming come?
  It has always been here
  - Do we need to expose hybrid architectures?
    Yes
  - Can we design high-level programming models, performance/debugging tools and runtime systems that can effectively abstract such architectural hybridness from the applications? Yes
  - Is it a good idea?
    Yes