Qualitative Evaluation Criteria for Parallel Programming Models

Christopher Krieger, Andrew Stone, 
Michelle Mills Strout

FIT 2010
Evaluating Performance and Programmability

- Implementation details tangle with algorithm specification to cause obfuscation
  - Data distribution and layout
  - Computation distribution and scheduling
- Programmer control enables performance
- Tangling hurts programmability
- We propose evaluating programming models by realizing implementation strategy patterns from the OPL (Mattson et al.)
  - SPMD, Loop Par, Fork/Join, BSP, Task Queue, …, Dist Array
Evaluation Criteria for Programming Models

Tangling and Programmer Control

- Manually applied loop optimizations
- Java synchronized keyword
- BLAS function calls
- Specification of thread affinity to core
- Weaved advice in an AOP language
- Hadoop configuration files
- Chapel iterators
- OpenMP CRITICAL directive
- Hadoop setNumMapTasks()
- Automatic parallelization
- Target/Gallop
Goal is to encourage the conscious development of program language constructs for the exposed and orthogonal specification of implementation details.

Questions for the community:
- Other important qualitative criteria?
- Additional parallel patterns for eval framework?
- What are some other construct examples?
- Missing categories in programmer control and tangling tradeoff space?