Plan for Today

Announcements

– Already have 2 paper presentation requests.
– Obtaining example research computations for projects is still in progress.
– Start considering what programming model example you might want to do: Chapel, Julia, OpenACC, Cuda, OpenMP, UPC, TBB, Cilk, ...

Performing a histogram computation in OpenMP

– Slides courtesy Mohamed Zahran at NYU.

Evaluating Programming Models

Preliminary Feedback
Qualitative Evaluation Criteria for Parallel Programming Models and Mechanisms that Separate Algorithms from Implementations for Parallel Patterns

Christopher Krieger, Andrew Stone, Michelle Mills Strout
FIT 2010 and ParaPLOP 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
- Propose evaluating programming models by realizing implementation strategy patterns from the OPL ([http://parlab.eecs.berkeley.edu/wiki/patterns/patterns](http://parlab.eecs.berkeley.edu/wiki/patterns/patterns))
  - SPMD, Loop Par, Fork/Join, BSP, Task Queue, …, Dist Array
Example of Significant Tangling

Figure 1: Iteration space and for loop implementations for a small example loop with dependences much like those in the Smith-Waterman code [28].
Mechanisms Paper

- Programming models evaluated
  - Chapel (Programming Language)
  - OpenMP (Pre-processing directives)
  - STAPL (Standard Template Adaptive Parallel Library)

- Key Criteria
  - Programmer control over implementation details (good)
  - Tangling is how interspersed those details are with algorithm code. (bad)
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/Goal

CSc 620 1/28/16
Parallelizing Example Computations
Categorizing Algorithm/Implementation Separation

- Put features into categories
  - Interspersed
  - Invisible
  - Hinted
  - Separated
  - Injected
  - Delegated

- Paper puts features of Chapel, OpenMP, and STAPL into categories based on how they work in context of implementation patterns from OPL (Our Pattern Language)
Moving Forward

- 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?
Other Criteria for Evaluating Programming Models

Performance of important benchmarks or applications
- Execution time
- Power and energy usage

Programmability
- SLOC, source lines of code
- User studies
- Qualitatively comparing examples

Survey on Parallel Programming Models
- System architecture
- Programming methodologies
- Worker management
- Workload partitioning scheme
- Task-to-worker mapping
- Synchronization
- Communication model
Next Time

Reading
– Qualitative Evaluation Criteria for Parallel Programming Models
– Mechanisms that Separate Algorithms from Implementations for Parallel Patterns

Class
– Example paper discussion about Karlin Paper.

Due
– Paper review for the Karlin Paper, if using this as one of your 8 reviews.
– Wednesday there will be a quiz due as long as quiz is posted by Monday evening.