Parallel Programming Model

Design and Implementation of UPC on GPU clusters
- Hierarchical UPC (HUPC)
  - Hybrid execution model of fork-join and SPMD
  - Affinity aware hierarchical data parallelism
- HUPC Implementation on GPU clusters

Safe Parallel Programming using Dependence Hints
- Hints to specify possible dependences between possibly parallel tasks
- Hints can be incomplete or incorrect
- Extends Do-Across and OpenMP directives

Adaptive Task Creation Strategy for Work-stealing
- An adaptive task creation strategy controls the tasks granularity.
- A new data attribute taskprivate is introduced for workspace variables.

Compilation Methodology & Infrastructure

Iterative Compilation
- Possible to derive a robust iterative optimization strategy across data set
- Optimizing programs across data sets is much easier than previously anticipated

Expert-Assisted Compilation Methodology
- Pattern-Oriented Optimization Directives
- Extendable via developer interfaces
- Integrate experts’ experience into compilers

Optimization Adaptor Framework
- Utilizing similarity between algorithms
- Defining the difference via adaptor
- Reuse existing optimization experiences

Memory-Aware Compiler Optimizations

Misaligned Data Access Handling in Binary Translation
- An exception handler-based approach
- Achieved 13%-44% speedup.

Mitigating Memory Bandwidth Contention
- Bandwidth-aware scheduling
- Maintain bandwidth utilization
- 4.1% speedup on average, up to 11.7%

Improving Heap Memory Layout by Dynamic Pool Allocation
- A lightweight dynamic optimizer
- Exploit the Affinity of heap objects
- 13% speedup on average, up to 82%

Software-Hardware Cooperative DRAM Bank Partitioning
- Page coloring + XOR cache mapping
- 5.3% speedup on average, up to 15%

Dynamic register promotion of stack variables
- Exploit additional register resources
- Runtime alias detection on page protection

Global Tiling for Communication
- Minimal Parallelization on DSM
  - 0-1 integer linear programming
  - Loop tiling for non-rectangular area

Reliable Software Developing Environment

Level-by-level: Flow- and Context-Sensitive Pointer Analysis
- Analyzing pointers level by level in terms of their points-to levels
- Full sparse SSA form
- Full transfer function and meet function

Detecting and Eliminating Potential Violations of Sequential Consistency for concurrent C/C++ programs
- Combining Shasha/Snir’s conflict graph and delay set theory
- Effectively detected PVSC bugs in MySQL/Apache

Xiaoheng Feng, feng@ict.ac.cn Institute of Computing Technology, CAS