Dynamic Verification of Data Race Freedom

Problem: Explicit-parallel programming is too complex
- Parallel programming environments prioritize efficiency
- Performance or performance/watt
- Full multi-core utilization requires extra work
- As a result, domain experts first create sequential codes
- Expressivity/scalability might be better than raw efficiency

Problem: Automatic parallelization is limited
- The holy grail of compiler research
- Heresies to perform high-level optimizations
- SW/HW speculative techniques have some success
- Scalability limited by incidental true dependencies
- All approaches have limited application coverage

Solution: Leverage DSL expressivity
- Domain Specific Languages allow for higher productivity
- Programming by providing high-level data-types and ops tailored to domain
- Relations, triangles, matrices, filtered traversal, ...
- Expresses intent without implementation artifacts

Solution: Use DSLs to provide scalable implicit parallelism
- DSL author encodes parallelism from DSL user
- High-level interface gives flexibility to implementation
- Declarative description of parallelism & locality patterns
- Portable and scalable specification of parallelism
- Adjust data structures, scheduling as system scales up

DELITE: A Framework for Building Parallel DSLs

Dynamic Expression Tree Rewriting using Strongly Typed Proxies

// factory method hides implementation type during construction
object Expr { def apply(value: Int): Expr = new Expr(value) }

// trait hides implementation type during use, injects operations into both proxy and concrete_impl
trait Expr extends DeliteDSLType with HasImpl.ExprImpl {
  def * (rhs: Expr) = run(a: Impl, b: Impl) => a.add(b))
}

class Impl(val value: Int) extends Expr {
  protected def add(rhs: Impl) = Expr(val + rhs.val)
}

class Proxy extends DeliteProxy[Expr,Impl] with Expr

DELITE: Simplifying Parallel Programming with Domain Specific Languages (DSLs)
Pervasive Parallelism (PPL) Lab. (http://ppl.stanford.edu)
Hassan Chafi, Nathan Bronson, Arvind Sujeeth, Kunle Olukotun

The Era of Many-Cores is Now

Dynamic Verification of Data Race Freedom

DELITE: Simplifying Parallel Programming with Domain Specific Languages (DSLs)
Pervasive Parallelism (PPL) Lab. (http://ppl.stanford.edu)
Hassan Chafi, Nathan Bronson, Arvind Sujeeth, Kunle Olukotun

Status and Next Steps

Status:
- Initial DSL targeting ML applications implemented
- Basic DELITE proxy system implemented
- Initial implementation of Unified Class Hierarchy

Next steps:
- Complete prototype implementation of DELITE runtime
- Pattern matching to express domain-specific op rewrite rules
- Collect initial performance numbers

Contact information

Group Website: http://ppl.stanford.edu
E-mail Addresses (Students):
(hchafi, nbronson, asujeeth)@stanford.edu
E-mail Address (Faculty Advisor):
kunle@stanford.edu