Software Challenges of Heterogeneity

Sudhakar Yalamanchili

Computer Architecture and Systems Laboratory
Center for Experimental Research in Computer Systems
School of Electrical and Computer Engineering
Georgia Institute of Technology

Sponsors: NSF, NVIDIA, Intel, AMD, LogicBlox
System Software Challenges of Heterogeneity

- **Execution Portability**
  - Systems evolve over time
  - New systems

- **Performance Portability**
  - New algorithms

- **Introspection**
  - Productivity tools

- **Application Migration**
  - Protect investments in existing code bases
Need for Execution Model Translation

Languages: Designed for Productivity

Execution Models (EM): Dynamic Translation of EMs to bridge this gap

Hardware Architectures – Design under speed, cost, and energy constraints
**Execution Portability:** Raise the Level of Abstraction

- Application portability as the system evolves
  - **On-line customization** to meet application & architecture diversity
- Refactoring and re-tuning of applications is expensive and impractical for many applications
  - Need to protect software investments
- Side effect free kernels
  - Analogy with assembly instructions

Kernels execute anywhere → Key to portability!
Domain Specific Compilation: Red Fox

\[ tr(x, y, z) \leftarrow E(x, y), E(y, z), E(x, z), x < y < z. \]

Joint with LogicBlox Inc.

H. Wu, G. Diamos

- Targeting Accelerator Clouds for meeting the demands of data warehousing applications
- In-core databases

LogiQL Queries → LogicBlox Front-End → Query Plan → RA-To-PTX (\(nvcc + RA-Lib\)) → Kernel IR → Kernel Weaver → IR Optimization
→ Red Fox RT → Machine Neutral Back-End

Translation Layer

Targeting Accelerator Clouds for meeting the demands of data warehousing applications

In-core databases
System Model

Large Graphs

Social
Media

Programming Models

Domain Specific Languages

Data Movement Optimizations

Compiler and Run-Time Support

System Abstractions

e.g. GAS, Virtual DIMMs, etc

Cluster Wide Hardware Consolidation

Hardware Customization
Summary: Dynamic Execution Environments

- **Core dynamic compiler and run-time system**
- **Standardized IR for compilation from domain specific languages**
- **Dynamic translation** as a key technology

**Productivity Tools**
- Correctness & Debugging
- Performance Tuning
- Workload Characterization
- Instrumentation

**Language Layer**

**Domain Specific Language**
- Datalog
- OpenCL
- CUDA
- DSLs?

**Dynamic Execution Layer**

**Language Layer**

- **Kernel IR**

**Introspection Layer**

- Harmony & Ocelot
System Software Challenges of Heterogeneity

- **Execution Portability**
  - Systems evolve over time
  - New systems

- **Performance Portability**
  - New algorithms

- **Introspection**
  - Productivity tools

- **Application Migration**
  - Protect investments in existing code bases
Motivation

Virtual ISA, e.g., HSAIL, PTX

Optimization

Language Front End

C/C++  OpenCL  CUDA  Python

• Application Validation
• Maintenance and Update
• Phase Behavior
• Impact of Deployment Environment

Key Idea: Code injection and JIT Compilation
Software Reliability Enhancement Framework

- Real time **customized** information available about GPU usage
- Use this information to drive SRE decisions

*Framework: On demand, customizable, transparent, and extensible, software reliability enhancement (SRE)*
Challenge of **Application-Level Energy Modeling**

- **Challenge**: How do we understand the energy implications of our decisions? Algorithms, data structures, etc.

- Note the variance of energy dissipation across different implementations of the same function.

---

Different implementations of BFS on different input data sets

*Courtesy H. Kim*
Thank You

Questions?