Obtaining Performance in Heterogeneous Systems

Keith D. Cooper

Department of Computer Science
Rice University
Houston, Texas

This talk builds on work done under sponsorship of DARPA, DOE (LACSI), NSF, and Texas Instruments.

The views expressed in this talk do not represent official views of any of those funding agencies or of Rice University.
Performance of Homogeneous Systems

- At the start of LACSI, we could achieve 5 to 10% of peak
- Using tools, we have improved that situation
  - An optimist might say we achieve 10 to 20%
- To achieve improvement, we have pursued several paths
  - Performance diagnosis tools (HPC Toolkit)
  - Autotuning strategies (Our work, VISTA, PERI, ...
  - Library based strategies — ATLAS, OSKI, SANS, FFTW, Telescoping languages, ...
  - and many others
- All that work has helped
  - Performance has doubled
  - Still, lots of cycles are not being used
Performance on Heterogeneous Systems

Some of the Impediments to Performance

• Partitioning work across resources
  • Match capabilities of resource to needs of code (issue of kind)
  • Match work to available resources (issue of quantity)
  • State of the art today:
    — Static partition (source level) of code to different units
    — Compile sources for each unit and run it

• Managing locality
  • Memory hierarchy management
    — Multicore may disrupt perceived wisdom on cache blocking
    — Bandwidth & latency management are still critical issues
  • Communication optimization
    — We’ve moved internode problems inside individual nodes

Some related work on Grid scheduling & execution

Feedback loop

⇒ situation will get worse with smaller hardware features
Performance on Heterogeneous Systems

To make effective use of these systems, we will need all the advances made by the HPC community in the past 40 years

• High-quality compilers & libraries
• Autotuning software systems
• Accurate tools for performance monitoring / debugging

I’m afraid, however, that we will take a cut & paste approach

• Patch together existing tools (& throw away information)
• Retrofit new behavior to old languages & tools
• Add compute power w/o addressing performance problems

I want to make three points

• Tools work, runtime optimization, novel uses for resources
Performance on Heterogeneous Systems

Toolchain Issues

(or “How will you program it?”)

• Talks that I’ve seen suggest using existing compilers & tools
  ♦ Separate compilers for various computational units
  ♦ Another compiler to partition work & data
  ♦ Flexible compute resources (FPGAs) further complicate picture

• To build applications, we need a good toolchain
  ♦ Here is one vision
    — Admittedly, for heterogeneous **embedded** systems
One Toolchain Vision

Application Problem

Source Application

High level compiler/automated partitioning and optimization of application across heterogeneous processing system

Exposed Intermediate Layer: Code Partition + Mappings

System Model

Low level device-specific compilers and libraries

Communication developed during high level partitioning

Device Type 1
Device Type 2
Device Type 3
Device Type 4
Device Type 5

Feedback loop to provide performance and resource management data. Used to improve overall system performance utilizing Cognitive Techniques

LACSI Symposium Panel, October 2006
Three distinct cycles of adaptation from the DARPA HECS Workshop.
Performance on Heterogeneous Systems

Toolchain Issues

• Talks that I’ve seen suggest using existing compilers & tools
  ♦ Separate compilers for various computational units
  ♦ Another compiler to partition work & data
  ♦ Flexible compute resources (FPGAs) further complicate picture

• Use of multiple independent compilers may limit success
  ♦ Hard to perform coordinated adaptation
  ♦ Loss of knowledge between various steps
    — Partitioning, node optimization, code generation, runtime

• To achieve performance, we need to do everything right
  ♦ May require a coordinated, multi-level compilation system
  ♦ Next slide is from Yelick’s talk in autotuning workshop
Compiler folks always seem to call for a new research program to produce the next great compiler. A serious research & development effort will be needed to build modern tools that support heterogeneous systems.
Runtime Optimization & Reoptimization

An Old, Simple Idea

• Classic arguments against it are cost & complexity
  ♦ “In a ‘real’ scientific code, we cannot afford the cycles”

• Clearly, we have the cycles to afford runtime improvement
  ♦ We are only using 20% of the available cycles

• We need to consider co-processes for code improvement, for data rearrangement, prefetching, work scheduling, ...
  ♦ Take a dynamic view of managing compile time and runtime

*I am not claiming that (generalized) runtime optimization is an answer to all of the challenges. Instead, I am saying that we need to use these techniques as one of our tools for attacking performance issues in heterogeneous systems.*
Novel Uses of Hardware Resources

Can we use “new” resources to solve problems other than the flop rate?

• Use FPGAs to build application-specific resources that address memory bottlenecks?
  ♦ High-speed facility for moving memory objects to mitigate conflicts in L1 or L2?
  ♦ Space to save contexts for application-managed concurrency

• Dedicate compute resources to dynamic performance-related issues
  ♦ Code reoptimization
  ♦ Data reorganization

Discussions I’ve seen focus on adding FLOPs and Ops, rather than improving the other details that determine performance.

Spending runtime resources on “overhead” can improve overall performance.
Toolchain Issues

The Next Great Compiler must accommodate

- Partitioning application across heterogeneous units
- High-quality optimization & code generation for all units
- Retargeting across broad range of compute units
- Use of feedback-based tuning at all levels
  - Model-based prediction of parameters, strategies, & effects
  - Empirical evaluation and exploration
  - Adaptation across optimizations & within optimizations

These demands have a “mom and apple pie” flavor.

They are approaching conventional wisdom.