Building Parameterized Performance Models for Black-Box Applications

Gabriel Marin, John Mellor-Crummey  Department of Computer Science Rice University

Why Performance Models?
- How well does this execution scale?
- Impact on design, number of processors
- How much resources does this execution need?
- What kind of resources are available?
- How should data and computation be mapped to resources?
- How will I know if my adaptation is necessary?
- How will a different architecture benefit this application?

Performance Modeling Challenges and Goals
- What to predict:
  - Computation time
  - Memory or cache utilization
  - Dynamic workload and cost
- Predict scaling as a function of:
  - Workload size
  - Environment size
  - Work characteristics
- Look at application binaries instead of source code
- Distill independent tools
- Model the system on applications
- Avoid all compiler dead code

Our Approach
Construct architecture-neutral Application Signatures
- Measure static characteristics
- Measure dynamic characteristics for multiple executions
- Use functions that characterize frequency and values
- Map against a library of functions
- Look at application binaries instead of source code
- Distill independent tools
- Model the system on applications
- Avoid all compiler dead code

Capturing Dynamic Behavior
- Analyze dynamic characteristics
  - Parallel computation profile: how much computation is happening in parallel
  - How much data is transferred between processors? What is the data transfer path?
  - How much time is spent in synchronization?
  - How much time is spent in memory accessing?

Performance Modeling Overview
- Build scalable, portable models useful for predicting performance
- Analyze dynamic characteristics
  - Parallel computation profile: how much computation is happening in parallel
  - How much data is transferred between processors? What is the data transfer path?
  - How much time is spent in synchronization?
  - How much time is spent in memory accessing?

Capturing Dynamic Behavior II
- Use a similar approach instrument all memory instructions to capture dynamic data about memory access locality

Predicting Computation Costs
- Use a similar approach instrument all memory instructions to capture dynamic data about memory access locality

Building an Architecture-neutral Model of Computation
- Build a model for the execution frequency of each task
- Build control flow graph
- Recover loop nesting structure
- Use counters model to recover frequency of executed instructions
- Compute path recovery of individual loops
- Compute path recovery of individual modules
- Use software only to implement loop in which it appears

Map Computation Model to a Target
- Input:
  - Instrumented model for loop execution frequency
  - Logs instruction time and resource dependencies
- Algorithm:
  - Build a native instruction set or use generic RISC instruction set
  - Simple, configurable Instruction set with target architecture
  - Use the frequent access paths
- Output:
  - Optimized execution time

Predicting Memory Hierarchy Access
- Track memory access distance for the model to cache hits vs. cache misses
- Track memory access distance for the model to cache misses

Performance Modeling Costs
- Use a similar approach instrument all memory instructions to capture dynamic data about memory access locality

Capturing Dynamic Behavior III
- Parallel performance depends on synchronization and data movement
- Capture a characterization of communication during execution
- When data is transferred
  - What processes communicate
  - Where communication occurs in the program

Modeling Memory Hierarchy
- Model a portable across different machine configurations
- Temporal reuse distance: distinct locations accessed between a pair of accesses to the same address
- Spatial reuse distance: distinct locations accessed between a pair of accesses to the same address
- Model the least distance between each pair of accesses
- Model the longest distance between each pair of accesses
- Build parameterized model for each bin

ACKNOWLEDGEMENTS
We would like to thank Richard Hanson and Ronald Goldman for providing valuable insight on computing parameterized curves from a set of measured points.