Automatic Performance Tuning for Multicore Architectures

Rudi Eigenmann

Purdue University
Why Autotuning? 
my bias

Ultimate goal: Dynamic Optimization Support For Compilers and More

- Runtime decisions for compilers are necessary because compile-time decisions are too conservative
  - Insufficient information about program input, architecture
  - When to apply what transformation in which flavor?
  - Polaris compiler has some 200 switches
    - Example of an important switch: parallelism threshold
  - Early runtime decisions:
    - Multi-version loops, runtime data-dependence test, 1980s

- My goals:
  - Looking for tuning parameters and evidence of performance difference
  - Go beyond the “usual”: unrolling, blocking, reordering
  - Show performance on real programs
Is there Potential?

You bet!

- Imagine you (the compiler) had full knowledge of input data and execution platform of the program.
Early Results on Fully-Dynamic Adaptation

- ADAPT system (Michael Voss - 2000)

Features:
- Triage
  - tune the most deserving program sections first
- Used remote compilation
  - Allowed standard compilers and all options to be used
- AL - adapt language

Issues:
- Scalability to large number of optimizations
- Shelter and re-tune
Recent Work
Offline Tuning - “Profile-time” tuning
Zhelong Pan

Challenges:

1. Explore the optimization space
   Empirical optimization algorithm - CGO 2006

2. Comparing performance
   Fair Rating methods - SC 2004
   - Comparing two (differently optimized) subroutine invocations

3. Choosing procedures as tuning candidates
   Tuning section selection - PACT 2006
   - Program partitioning into tuning sections

Two goals: increase program performance and reduce tuning time
Whole-Program Tuning

Search Algorithms
- BE: batch elimination
  - Eliminates “bad” optimizations in a batch => fast
  - Does not consider interaction => not effective
- IE: iterative elimination
  - Eliminates one “bad” optimization at a time => slow
  - Considers interaction => effective
- **CE: combined elimination (final algorithm)**
  - Eliminates a few “bad” optimizations at a time
- Other algorithms
  - optimization space exploration, statistical selection, genetic algorithm, random search
Performance Improvement

Tuning Goal: determine the best combination of GCC options

Tuning at the Procedure Level

(1) Tuning Section Selection (TSS)

(2) Rating Method Analysis (RMA)

(3) Code Instrumentation (CI)

(4) Driver Generation (DG)

(5) Performance Tuning (PT)

(6) Final Version Generation (FVG)

Pre-Tuning

During Tuning

Post-Tuning
Reduction of Tuning Time through Procedure-level Tuning

normalized tuning time

Whole PEAK

ammp applu api art equake mesa mgrid sixtrack swim wupwise GeoMean

2.33 7.06 11.21 4.03 1.79 2.33 3.38 4.22 2.59 1.61 3.36

62.22 50.99 105.76 69.23 89.28 50.59 87.32 102.97 68.28 50.59 102.97
Tuning Time Components

Percentage of the total time spent in tuning

- ammp
- applu
- apsi
- art
- equake
- mesa
- mgrid
- sixtrack
- swim
- wupwise
- Average

Ongoing Work
Seyong Lee

Beyond autotuning of compiler options

- New applications of the tuning system
  - MPI parameter tuning
  - Tuning library selection - (ScalaPack, ...)
  - OpenMP to MPI translator
TCP Buffer Size Effect on NPB

Target system: Hamlet (Dell IA-32 P4 nodes) clusters in Purdue RAC
Used MPI: MPICH1
Alltoall collective call performance (without segmentation)

Target system: Hamlet (Dell IA-32 P4 nodes) clusters in Purdue RAC
Used MPI: Open MPI 1.2.2
Segmentation Effect on Basic Linear Alltoall Algorithm

Target system: Hamlet (Dell IA-32 P4 nodes) clusters in Purdue RAC

Used MPI: Open MPI 1.2.2
A Related Project

- Autotuning in iShare - an Internet Sharing System

*Publish - Discover - Adapt*

1. Published autotuner (available)
2. Tuning upon matching discovered application and platform (current work)
Automatic Tuning for Multicore

- Starting point was the Polaris compiler - 200 switches
- Early results on dynamic serialization
- Goal: parallelizing compiler that never lowers the performance of a program
- OpenMP to MPI translation
- Tuning NICA architectures
  - Multicore + niche capabilities (accelerators and more)
Conclusions and Discussion

Dynamic Adaptation is one of the most exciting research topics

There are still issues to Sink your Teeth in
- Runtime overhead: when to shelter/re-tune
- Fine-grain tuning
- Model-guided pruning of search space
- Architecture of an autotuner
  - If we could agree, we could plug-in our modules
- AutoAuto - autotuning autoparallelizer
- How to get order(s) of magnitude improvement
  - Wanted: tuning parameters and their performance effects