Compilers and Runtime Systems for Dynamically Adaptive Applications (a.k.a. autotuning?)

Rudi Eigenmann
Purdue University
Why Autotuning?  
my bias

- 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
- Idea for dynamic adaptation dates back to DARPAs HPCC program, early 1990s
- 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

“Amdahl’s law” of Autotuning

You are here
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
  - 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)
   - 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

Autotuning Workshop, Snowbird, July 2007
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

![Bar Chart]

- ammp: 62.22
- applu: 50.99
- api: 105.76
- art: 69.23
- equake: 63.14
- mesa: 89.28
- mgrid: 50.59
- sixtrack: 87.32
- swim: 36.96
- wupwise: 102.97
- GeoMean: 68.28

Normalized tuning time

Whole PEAK
Tuning Time Components

Percentage of the total time spent in tuning

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

Legend:
- TSS
- RMA
- CI
- DG
- PT
- FVG
Ongoing Work
Seyong Lee

- Biggest part of the tuning system is runtime
  - Compiler was just the first application

- 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
OpenMP to MPI Reduction Translation

OpenMP code

```c
!$OMP PARALLEL DO PRIVATE(J, K)
   DO J=1, nrows
      w(J) = 0.0
      DO K=row(J), row(J+1)
         w(J) = w(J) + a(K)*p(colidx(K))
      ENDDO
   ENDDO
```

Translation w/o reduction

```c
Call MPI_AllGather(…)
DO J=s_index, e_index
   w(J) = 0.0
   DO K=row(J), row(J+1)
      w(J) = w(J) + a(K)*p(colidx(K))
   ENDDO
ENDDO
ENDDO
```

Reduction Translation

```c
DO J=1, nrows
   w(J) = 0.0
   DO K=row(J), row(J+1)
      IF (colidx(K) is local)
         w(J) = w(J) + a(K)*p(colidx(K))
      ENDIF
   ENDDO
ENDDO
ENDDO
Call MPI_AllReduce(…)
```

Allgather vs. Allreduce (32 processors)

![Chart showing execution time comparison between Allgather and Allreduce]

Data size in bytes

Execution Time in usecs

Autotuning Workshop, Snowbird, July 2007
Variants of Communication Libraries for Sparse Matrix Vector Multiplication

- Simple Translation
  - without SMVM recognition
    
    Call MPI_AllGatherv(...)
    DO J=1, NA
    DO K=row(J), row(J+1)
    ... ENDDO
    ENDDO

- OPT1 (w/ SMVM recognition)

    DO J=1, NA
    DO K=row(J), row(J+1)
    ...
    ENDDO
    ENDDO
    Call MPI_AllReduce(...)

- OPT2 (w/ SMVM recognition)

    DO J=1, NA
    DO K=row(J), row(J+1)
    ...
    ENDDO
    ENDDO
    DO PID=1, NPROCS
    Call MPI_Reduce(...)
    ENDDO

- OPT3 (w/ SMVM recognition)

    DO J=1, NA
    DO K=row(J), row(J+1)
    ...
    ENDDO
    ENDDO
    DO I = 1, LOG2NPROCS
    Call MPI_IRecv(...)
    Call MPI_IWait(...)
    ENDDO

Autotuning Workshop, Snowbird, July 2007
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)
Conclusions and Discussion

Dynamic Adaptation is one of the most exciting research topics, but 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