Beyond Automatic Performance Analysis

Prof. Dr. Michael Gerndt
Technische Universität München
gerndt@in.tum.de
Performance Analysis and Tuning is Essential
Performance Analysis at Scale

A high level description of the performance of cosmology code MADCAP.

Concurrency

Architecture

Source: David Skinner, NERSC
Performance Analysis for Parallel Systems

- Development cycle
  - Assumption: Reproducibility
- Instrumentation
  - Static vs Dynamic
  - Source-level vs binary-level
- Monitoring
  - Software vs Hardware
  - Statistical profiles vs event traces
- Analysis
  - Source-based tools
  - Visualization tools
  - Automatic analysis tools
• Automated search
  – Based on formalized performance properties

• Online analysis
  – Search performed while application is executing

• Distributed search
  – User specified number of analysis agents
  – Additional cores for agents

• Profile data only
  – even for MPI Waittime analysis
Properties

• StallCycles(Region, Rank, Thread, Metric, Phase)
  – Condition: Percentage of lost cycles >30%
  – Confidence: 1.0
  – Severity: Percentage of lost cycles

• StallCyclesIntegerLoads
  – Requires access to two counters

• L3MissesDominatingMemoryAccess
  – Condition: Importance of L3 misses (theoretical latencies)
  – Severity: Importance multiplied by actual stall cycles
Periscope Design

Interactive Frontend

Performance Analysis Agent Network

Master Agent

Communication Agent

Analysis Agent

MRI

Application with Monitor
Agent Search Strategies

• Application phase is a period of program’s execution
  – Phase regions
    • Full program
    • Single user region assumed to be repetitive
  – Phase boundaries have to be global (SPMD programs)

• Search strategies
  – Determine hypothesis refinement
    • Region nesting
    • Property hierarchy-based refinement
Agent Search Strategies

- Stall cycle analysis
- Stall cycle analysis with breadth first search
- MPI strategy
- OpenMP strategy
- OMP Scalability strategy
- Benchmarking strategy
- Default strategy
Crystal Growth Simulation

Temperature distribution at t = 10.0 sec and t = 20.0 sec.
Results

USER_REGION; cx.f:61; 78.361; IA64 Pipeline Stall Cycles
46.250; Stalls due to L1D TLB misses ...
36.018; L3 misses dominate memory access
30.294; Stalls due to waiting for FP register

CALL_REGION; cx.f:70; 52.324; IA64 Pipeline Stall Cycles
29.468; L3 misses dominate memory access
26.858; Stalls due to L1D TLB misses ...
23.972; Stalls due to waiting for FP register

velo; cx.f:622; 52.316; IA64 Pipeline Stall Cycles
29.465; L3 misses dominate memory access
26.854; Stalls due to L1D TLB misses ...
23.972; Stalls due to waiting for FP register

LOOP_REGION; cx.f:731; 32.512; IA64 Pipeline Stall Cycles
24.098; L3 misses dominate memory access
20.035; Stalls due to waiting for FP register
Integration in Eclipse (PTP)

Where is the problem?

What is the most severe problem?

Filter problems for region
PerSyst: Periscope System Monitoring

- Distributed fault tolerant architecture
- Incremental analysis

- Data Base
- High Level Agent
- Synchronisierung und Aggregation
- Analysis Agent
- Analysis Agent
- IO Agent
- IO Agent

1 Agent per Partition

Altix 4700
PerSyst: Data Reduction

- Aggregation of data in properties
- Aggregation per job

Data Base

High Level Agent

Analysis Agent

IO Agent

< 18 MB/Tag

< 244 MB/Tag

280 MB/Tag
AutoTune

- Automatic application tuning
  - Performance and energy

- Parallel architectures
  - HPC and parallel servers
  - Homogeneous and heterogeneous
  - Multicore and GPU accelerated systems
  - Reproducable execution capabilities

- Variety of parallel paradigms
  - MPI, HMPP, parallel patterns
Partners

Technische Universität München
Universität Wien
CAPS Entreprises
Universitat Autònoma de Barcelona
Leibniz Computing Centre
National University of Galaway, ICHEC
Autotune Approach

• Predefined tuning strategies combining performance analysis and tuning

• Plugins
  – Compiler based optimization
  – HMPP tuning for GPUs
  – Parallel pattern tuning
  – MPI tuning
  – Energy efficiency tuning
Periscope Tuning Framework

• Online
  – Analysis and evaluation of tuned version in single application run
  – Multiple versions in single step due to parallelism in application

• Result
  – Tuning recommendation
  – Adaptation of source code and/or execution environment
  – Impact on production runs
Autotuning Extension in HMPP

• Directives to provide optimization space to explore
  – Parameterized loop transformations
  – Alternative/specialized code declaration to specify various implementations

• Runtime API
  – Optimization space description
  – Static code information collect
  – Dynamic information collect (i.e. timing, parameter values)

```c
#pragma hmppcg(CUDA) unroll(RANGE), jam
for( i = 0 ; i < n; i++ ) {
    for( j = 0 ; j < n; j++ ) {
        VC(j,i) = alpha*prod+beta * VC(j,i);
    }
}
```
Energy Efficiency Plugin (LRZ)

1. Preparation phase:
   • Selection of possible core frequencies
   • Selection of regions for code instrumentation

2. “While”-loop (until region refinement):
   “For”-loop (all frequencies) :
     a) Set new frequency of tuned region
     b) Periscope analysis (instrumentation, (re-)compiling, start and stop experiment)
     c) Measure execution time + energy of tuned region
     d) Evaluate experiment
   “End for”

3. Evaluate results of all experiments in refinement loop

4. Store best frequencies-region combination
Parallel Pattern/MPI Tuning

- PP or MPI Plugin encloses a performance model (ex. M/W) based on Periscope-provided performance data
- Automatically generated tuning decisions are sent to Tuner
- Tuner dynamically modifies the application before next experiment
Expected Impact

- Improved performance of applications
- Reduced power consumption of parallel systems
- Facilitated program development and porting
- Reduced time for application tuning
- Leadership of European performance tools groups
- Strengthened European HPC industry
THANK YOU