Center for Scalable Application Development Software: Application Engagement

Ewing Lusk (ANL)
Gabriel Marin (Rice)
Application Engagement

• Workshops (2 out of 4) for outreach
  – Leadership Class Machines, Applications, and Performance
  – Scientific Data Analysis and Visualization for Petascale Computing

• Other application engagement
  – continued interactions with workshop participants
  – focused engagement with specific application groups
Workshop on Leadership Class Machines, Applications, and Performance

• **Goal:** Jumpstart productive application use of DOE’s large-scale facilities at ANL, ORNL, and NERSC

• **Target audience**
  – code developers; not necessarily P.I.’s
  – from DOE SciDAC and INCITE programs

• **Content (presented by 12 speakers)**
  – leadership machine architectures
  – programming at scale for performance
  – tools for understanding code behavior
  – I/O and visualization
  – hands-on sessions (time for hacking, with sysadmins available)
Attendees and Their Projects

• 2008: 25 attendees
  – 12 from projects with INCITE awards
  – 15 from other SciDAC projects

• Wide range of application areas, e.g.:
  – climate INCITE
  – FACETS SciDAC (fusion)
  – CSCAPES SciDAC
  – COMPASS SciDAC (particle accelerator modeling) also INCITE
  – biofuels INCITE
  – combustion in gas turbines INCITE
Interactions Born at the Workshops

- Revised approach to parallel I/O for sparse matrix data structures, used in part of UNEDF SciDAC. Interactions continued at recent Nuclear Physics Exascale meeting.
- Alternate file format and parallel I/O strategy for PHASTA. Continued at ALCF workshop in January.
- Turbulent flow project for Center for Turbulence Research at Stanford. I/O strategies on Franklin (NERSC machine). Ported code to BG/P. Continued interactions with ALCF staff.
- Worked with CERFACS on problem decomposition; student visited Argonne for further optimization work.
- Formed collaboration with NERSC on HDF5 issues.
- Worked closely over the year with GFMC part of UNEDF SciDAC.

Lasting relationships formed at Snowbird workshops
Focused Engagement Activities

- Need representative applications to drive compilers/tools research
- CScADS collaborates in PERI Tiger Teams
  - PERI Tiger Teams engage CScADS with applications
  - CScADS extends PERI Tiger Team efforts
  - develop software and methods to help application teams
    - GTC: particle-in-cell simulation of turbulent plasma in a tokamak
    - FLASH: block structured AMR to simulate astrophysical flashes
    - S3D: direct numerical simulation of combustion using dense arrays
The Challenge of Application Tuning

• Performance measurement tools identify
  – sections of code that execute inefficiently
  – loops that incur a high fraction of a particular hardware event

• Knowing where cache misses occur is seldom enough
  – data reuse is not a local phenomenon

• Assess tuning opportunities through modeling
  – identify performance bottlenecks due to application characteristics
    • instruction level parallelism, data reuse patterns, data layout
  – provide insight into code transformations for improving performance
    • PERI aims to provide guidance for next generation procurements
    • CScADS aims to help improve application performance
  – understand performance impact for a target architecture
    • understand mismatch between application and architecture
    • impact of ILP depends on machine width, execution unit types
Modeling Toolkit Design Overview

Object Code

- Binary Analyzer
  - Control flow graph
  - Loop nesting
  - Instruction dependences
  - BB instruction mix

Static Analysis

Binary Instrumenter

Instrument Code

Dynamic Analysis

- Execute
  - BB & Edge Counts
  - Memory Reuse Distance

Scalable Models

- Model Builder
  - Architecture neutral model
  - Evaluate

Cross Architecture Models

- IR code
- Modulo Scheduler

Performance Prediction for Target Architecture
Understanding Data Reuse

Identify data reuse patterns using reuse distance

• Characterize reuse patterns by a tuple of scopes
  – source, destination, carrying scopes

• Understand not only where cache misses occur
  – identify where data has been previously accessed
  – identify which algorithmic loop is driving the reuse

• important for understanding how to improve the reuse

GTC example

- spatial reuse

```
do kz=1,mzbig
  wz=real(kz)/real(mzbig)
  zdum=zetamin+deltaz*(real(k-1)+wz)
  do i=idiag1,idiag2
    ii=igrid(i)
    do j=1,mtdiag
      ...
      phiflux(kz+(k-1)*mzbig,j,i)= ...
    enddo
  enddo
enddo```

```
Missed Opportunities for Spatial Locality

• Problem: inefficient use of caches
  – long temporal or spatial data reuse
    • captured by memory reuse distance
  – unused data in cache lines
    • data fetched in blocks; some words never accessed
    • *fragmentation factor* = \[ \frac{\text{data fetched but never accessed}}{\text{total fetched data}} \]

• Approach
  – compute fragmentation factors for references
    • use static analysis to understand access stride and fraction of data never accessed
  – report misses due to data fragmentation at each level
GTS: simulates turbulent plasma in tokamak reactors
- 3D particle-in-cell code; 1D decomposition along toroidal direction
  - charge: deposit charge from particles to grid points
  - solve: compute the electrostatic potential and field on grid points
  - push: compute the force on each particle from nearby grid points

Used measurement and modeling tools developed at Rice with CScADS support to pinpoint performance losses
- poor spatial locality due to vector of structures representation for ions
- unrealized opportunities for temporal reuse between loops over ions

Improving node performance
- manually transform to structure of vectors
- manually apply fusion and blocking to improve temporal reuse
- transmit improvements back to GTC/GTS code teams
GTS: Node Performance Improvements

- Metrics normalized to measurements of original code
- Lower is better
GTS: Locality Degrades as Ions Swirl

- Locality is best when particles are sorted in cell order
  - potential computation uses cell data only
  - charge deposition and particle pushing involve interactions between particles and cells
- Initially particles are uniformly distributed in cell order

![Graphs showing cell distribution over time steps](image)
• Locality is best when particles are sorted in cell order
  — potential computation uses cell data only
  — charge deposition and particle pushing involve interactions between particles and cells

• Over time, the particle distribution diverges from cell order
GTS: Potential Improvement from Reordering

- Locality degrades gradually at run-time
- Assumptions:
  - periodic particle reordering restores locality and performance
  - performance degrades at similar rate after each sorting step
GTS: Compute Optimal Sorting Interval

- **Notations**
  - $f(x) = \text{time step cost function}$
  - $C = \text{cost of sorting}$
  - $G(t) = \text{gain from sorting every } t \text{ time steps}$

- **Find } t \text{ that maximizes } G(t) \text{ over } N \text{ steps} \quad G(t) = \sum_{k=1}^{N-1} \left( \int_{kt}^{(k+1)t} f(x) \, dx - \int_0^t f(x) \, dx - C \right)

- **Find } t \text{ that minimizes } h(t) = \text{average time step cost with sorting} \quad h(t) = \frac{1}{t} \left( \int_0^t f(x) \, dx + C \right)
Parallel Performance Results

- Combined optimizations reduce GTS execution time by
  - 37% on Itanium2 cluster
  - 21% on Cray XT and Cray XD1
FLASH suffers from poor spatial locality due to data layout

(values predicted for Sedov test case)

Array `unk` – main data structure holding cell centered data for PARAMESH – has variable index on innermost dimension.

- Total L3 miss count
- L3 cache misses due to fragmentation of data in cache lines: 22% of total

Array statement at line 115 accounts for 78% of fragmentation misses.
Application Engagement: S3D

- Direct numerical simulation (DNS) of turbulent combustion
  - state-of-the-art code developed at CRF/Sandia
    - PI: Jaqueline H. Chen, SNL
  - 2007/2008 INCITE awards at NCCS
  - pioneering application for 250TF Jaguar system

- Extend performance analysis work of PERI Tiger team
  - use HPCToolkit to locate single-core performance bottlenecks
    - compiler inserted array copies
    - streaming calculations with low data reuse
    - loop nests with recurrences
  - identified opportunities for compiler-based improvement
  - enhanced LoopTool to address S3D’s needs
  - improved loop nests with LoopTool’s semi-automatic transforms
    - transformed code is now part of S3D’s source base
  - used HPCToolkit to assess multicore scaling issues

J H Chen et al 2009 Computational Science and Discovery 2 015001 (31pp)
S3D: What Opportunities Exist?

5D loop nest:
2D explicit loops
3D F90 vector syntax

initialize
update

performance
problem
data streams
in/out of memory

reuse
reuse
reuse
reuse
LoopTool: Loop Optimization of Fortran

Rice University’s tool for source-to-source transformation of Fortran (transformation subset shown)

Loop Unswitching

if P then
  S1
else
  S2
  S3

Controlled Loop Fusion

S1
S2
S3
S4

Unroll and Jam

do k = 1,n
  do j = ...
    S1(j,k)
  enddo
  S3
  S4
endo
do k = 1,n-1,2
  do j = ...
    S1(j,k)
    S1(j,k+1)
  enddo
endo
Optimization of S3D Diffusive Flux Loop

Transformation Log:
- scalarization (4 stmts)
- loop unswitching (2 conditions)
- fusion (loops within 4 outer nests)
- unroll-and-jam (2 loops)
- peeling excess iterations (4 nests)

2.94x faster than original (6.7% total savings)
Ongoing Work

- Beginning to study new applications
  - PFLOTRAN, POP
- Modeling toolkit
  - replacing EEL (licensed) with Dyninst toolkit (open source)
    - create open source multi-platform tool
  - status: testing instrumentation using GTC on Opteron platform
- LoopTool
  - being prepared for deployment in 2009