Performance Tuning of Scientific Applications

David H Bailey, Lawrence Berkeley National Lab, USA (speaker)
DHB’s website:
http://crd.lbl.gov/~dhbailey
Why performance is important

- Highly-parallel scientific computing is widely used in science and technology:
  - Climate modeling – exploring scenarios for global warming.
  - Materials science – photovoltaics, batteries and nanoelectronics.
  - Astrophysics – supernova explosions, cosmology, data processing.
  - Physics – tests of standard model and alternatives.
  - Biology – model biochemical processes, develop new drugs.
  - Combustion – test designs for greater efficiency and less pollution.

- However, achieved performance is often poor – typically only 1-5% of peak. Common reasons include:
  - Limited parallel concurrency in key loops, resulting in poor load balance.
  - Suboptimal structure of loops and blocks.
  - Subtle interference effects in data communication channels.

- Low performance is unacceptable, not only because of the high purchase cost of state-of-the-art systems, but also because of the increasing cost of providing electrical power for these systems.
The Performance Engineering Research Institute (PERI)

- Performing research in theory, techniques and application of performance tuning for scientific computing.
- Funded by U.S. Dept. of Energy, Office of Science (SciDAC program).
- Participating members:
  - University of Southern Cal. (lead)
  - Argonne National Lab.
  - Lawrence Livermore National Lab.
  - University of California, San Diego
  - University of North Carolina/RENCI
  - Lawrence Berkeley Natl. Lab. (asst. lead)
  - Oak Ridge National Lab.
  - Rice University
  - University of Maryland
  - University of Tennessee, Knoxville
  - University of Utah
- Principal research thrusts:
  - Performance modeling and analysis.
  - Automatic performance tuning.
  - Application analysis.
Semi-automated performance modeling methodology:
- Performance trace runs obtain profiles of applications.
- Performance probes obtain profiles of computer systems.
- A “convolution” approach combines application and system profiles to produce quantitative predictions of performance.

Uses:
- Permits scientists to understand the bottlenecks in their codes and future potential for parallel scalability.
- Permits computing facility managers to plan future requirements and improve the selection process of large systems.

Recent advances include:
- Techniques to significantly reduce the volume of trace data required.
- Techniques to extrapolate models to larger future systems.
- Extensions of modeling methods to encompass energy consumption.
- Applications to both DOD and DOE computational workloads.

Credit: Allan Snavely, UCSD
Performance modeling at LBNL

- Erich Strohmaier’s ApexMAP: A simple modeling framework based on dataset size, spatial locality and temporal locality.
- Samuel Williams’ “roofline” model: Compares achieved performance to a “roofline” graph of peak data streaming bandwidth and peak flop/s capacity.

![Performance modeling diagram]

- **Opteron 2356 (Barcelona)**
- **Arithmetic Intensity**
- **GFlop/s**
- **Peak performance (73.6 Gflop/s)**
- **Kernel #1**
- **Kernel #2**
- **Kernel #3**
- **Each kernel's performance bound**
- **Each kernel's range in arithmetic intensity**
- **Performance roofline (Y-coordinate is performance)**
- **Processor-bandwidth roofline (slope is bandwidth)**
Background: We have found that most computational scientists are reluctant to learn and use performance tools in day-to-day research work.

Solution: Extend semi-automatic performance tuning techniques, such as those developed for specialized libraries like FFTW (FFTs) and ATLAS (dense matrix computation), to the more general area of large-scale scientific computing.
The PERI autotuning framework

- HPC Toolkit (Rice)
- ROSE (LLNL)
- CHiLL (USC/ISI and Utah)
- ROSE (LLNL)
- Orio (Argonne)

- OSKI (LBNL)
- PerfTrack (LBNL, SDSC, RENCI)
- Active Harmony (UMD)
- GCO (UTK)
Applications of PERI research

- PERI research tools and expertise have been applied to numerous scientific application codes, in many cases with notable results.
- Even modest performance improvements in widely-used, high-profile codes can save hundreds of thousands of dollars in computer time.

Examples:
- **S3D (Sandia code to model turbulence):**
  - Improved exp routine (later supplanted by improved exp from Cray).
  - Improved set of compiler settings.
  - Achieved 12.7% overall performance improvement.
  - S3D runs consume 6,000,000 CPU-hours of computer time per year, so 762,000 CPU-hours are potentially saved each year.
- **PFLOTRAN (LANL code to subsurface reactive flows):**
  - Two key PETSc routines (17% of run time) and a third routine (7% of run time) were each accelerated by more than 2X using autotuning.
  - 40X speedup in initialization phase, and 4X improvement in I/O stage.
  - Overall 5X speedup on runs with 90,000 or more cores.
SMG2000

- SMG2000: A semicoarsening multigrid solver code, used for various applications including modeling of groundwater diffusion.
- PERI researchers integrated several tools, then developed a “smart” search technique to find an optimal tuning strategy among 581 million different choices.
- Achieved 2.37X performance improvement on one key kernel.
- Achieved 27% overall performance improvement.
Autotuning the central SMG2000 kernel

Outlined code (from ROSE outliner)

for (si = 0; si < stencil_size; si++)
  for (kk = 0; kk < hypre__mz; kk++)
    for (jj = 0; jj < hypre__my; jj++)
      for (ii = 0; ii < hypre__mx; ii++)
        rp[((ri+ii)+(jj*hypre__sy3))+(kk*hypre__sz3)] -=
          ((Ap_0[((ii+(jj*hypre__sy1))+(kk*hypre__sz1))]+((A->data_indices)[i][si]))*
           (xp_0[((ii+(jj*hypre__sy2))+(kk*hypre__sz2))+((*dxp_s)[si]))));

CHiLL transformation recipe

permute([2,3,1,4])
tile(0,4,TI)
tile(0,3,TJ)
tile(0,3,TK)
unroll(0,6,US)
unroll(0,7,UI)

Constraints on search

0 ≤ TI , TJ, TK ≤ 122
0 ≤ UI ≤ 16
0 ≤ US ≤ 10
compilers ∈ {gcc, icc}

Search space:

122^3 x 16 x 10 x 2 = 581,071,360 points

Credit: Mary Hall, Utah
Search for optimal tuning parameters for SMG kernel

Parallel heuristic search (using Active Harmony) evaluates 490 total points and converges in 20 steps.

Selected parameters:
TI=122, TJ=106, TK=56, UI=8, US=3, Comp=gcc

Performance gain on residual computation:
2.37X

Performance gain on full application:
27.23% improvement

Credit: Mary Hall, Utah
Autotuning the triangular solve kernel of the Nek5000 turbulence code

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Original</th>
<th>Active Harmony</th>
<th>Exhaustive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>(u1,u2)</td>
<td>Speedup</td>
</tr>
<tr>
<td>pathscale</td>
<td>0.58</td>
<td>(3,11)</td>
<td>1.81</td>
</tr>
<tr>
<td>gnu</td>
<td>0.71</td>
<td>(5,13)</td>
<td>1.51</td>
</tr>
<tr>
<td>pgi</td>
<td>0.90</td>
<td>(5,3)</td>
<td>1.70</td>
</tr>
<tr>
<td>cray</td>
<td>1.13</td>
<td>(15,5)</td>
<td>1.61</td>
</tr>
</tbody>
</table>

Credit: Jeff Hollingsworth, Maryland
LBMHD (left): Implements a lattice Boltzmann method for magnetohydrodynamic plasma turbulence simulation.

GTC (right): A gyrokinetic toroidal code for plasma turbulence modeling.

Credit: Samuel Williams, LBNL
LS3DF
(LBNL work)

♦ LS3DF: “linearly scaling 3-dimensional fragment” method.
♦ Developed at LBNL by Lin-Wang Wang and several collaborators.
♦ Used for electronic structure calculations – numerous applications in materials science and nanoscience.
♦ Employs a novel divide-and-conquer scheme including a new approach for patching the fragments together.
♦ Achieves nearly linear scaling in computational cost versus size of problem, compared with $n^3$ scaling in many other comparable codes.
♦ Potential for nearly linear scaling in performance versus number of cores.

Challenge:
♦ Initial implementation of LS3DF had disappointingly low performance and parallel scalability.
2-D domain patching scheme in LS3DF

Total = \sum F \{ F_{FF} \}

Boundary effects are (nearly) cancelled out between the fragments:

System = \sum_{i,j,k} \{ F_{222} + F_{211} + F_{121} + F_{112} - F_{221} - F_{212} - F_{122} - F_{111} \}

Credit: Lin-Wang Wang, LBNL
LBNL’s performance analysis of LS3DF

LBNL researchers (funded through PERI) applied performance monitoring tools to analyze run-time performance of LS3DF. Key issues uncovered:

- Limited concurrency in a key step, resulting in a significant load imbalance between processors.
  - Solution: Modify code for two-dimensional parallelism.
- Costly file I/O operations were used for data communication between processors.
  - Solution: Replace all file I/O operations with MPI send-receive operations.
Resulting performance of tuned LS3DF

- 135 Tflops/s on 36,864 cores of the Cray XT4 Franklin system at LBNL.
  - 40% efficiency on 36,864 cores.
- 224 Tflops/s on 163,840 processors of the BlueGene/P Intrepid system at Argonne Natl. Lab.
  - 40% efficiency on 163,840 cores.
- 442 Tflops/s on 147,456 processors of the Cray XT5 Jaguar system at Oak Ridge Natl. Lab.
  - 33% efficiency on 147,456 cores.

The authors of the LS3DF paper were awarded the 2008 ACM Gordon Bell Prize in a special category for “algorithm innovation.”
Near-linear parallel scaling for up to 163,840 cores and up to 442 Tflop/s
Solar cell application of tuned LS3DF

- Single-band material theoretical photovoltaic efficiency is limited to 30%.
- With an intermediate state, the photovoltaic efficiency may increase to 60%.
- Proposed material: ZnTe:O.
  - Is there really a gap?
  - Is there sufficient oscillator strength?
- LS3DF calculation used for 3500 atom 3% O alloy [one hour on 17,000 cores of Franklin system].
- Result: There is a gap, and O induced states are highly localized.

Credit: Lin-Wang Wang, LBNL
For additional details:
Performance Tuning of Scientific Applications

Editors: Bailey (LBNL), Lucas (USC/ISI), Williams (LBNL); numerous individual authors of various chapters.
Publisher: CRC Computational Science, Jan 2011.
Contents:
1. Introduction
2. Parallel computer architecture
3. Software interfaces to hardware counters
5. Trace-base tools
6. Large-scale numerical simulations on high-end computational platforms
7. Performance modeling: the convolution approach
8. Analytic modeling for memory access patterns based on Apex-MAP
9. The roofline model
10. End-to-end auto-tuning with active harmony
11. Languages and compilers for auto-tuning
12. Empirical performance tuning of dense linear algebra software
13. Auto-tuning memory-intensive kernels for multicore
14. Flexible tools supporting a scalable first-principles MD code
15. The community climate system model
16. Tuning an electronic structure code