Performance Analysis Tools

Shirley Moore
shirley@cs.utk.edu
April 3, 2002

Optimizing Compiler

- Goal is efficient translation of higher-level language into the fastest possible machine language that accurately represents the high-level language source
- Optimization levels and options
  - No optimization
  - Basic optimization
  - Interprocedural analysis
  - Runtime profile analysis
  - Automatic parallelization

Classical Optimizations

- Copy propagation
- Constant folding
- Dead code removal
- Strength reduction
- Variable renaming
- Common subexpression elimination
- Loop-invariant code motion
- Induction variable simplification

Helping the Compiler

- Use straightforward coding style so as not to confuse the compiler
- Look at compiler listing to see where code may need to be restructured
  - Restructure loops
  - Simplify or rearrange memory references

Timing and Profiling

- Timing an entire program
  - UNIX time command outputs
    - User time
    - System time
    - Elapsed time
  - User time + system time = CPU time
  - Additional time output
    - Percent utilization
    - Average memory utilization
    - Blocked I/O operations
    - Page faults and swaps

Timing a Portion of a Program

1. Record the time before you start doing X
2. Do X
3. Record the time at completion of X
4. Subtract the start time from the completion time
Subroutine Profiling

- Most compilers provide a facility to automatically insert timing calls into your programs at the entry and exit of each routine at compile time or to enable sampling.
- A separate utility (e.g., prof, gprof) produces a report showing the percentage of time spent in each routine.
- Many performance analysis tools also provide this capability.

Types of Profiling

- Time-based
- Based on other metrics such as
  - Operation counts
  - Cache and memory event counts

Possible Profiles

- Sharp profile
  - Most of the time spent in one or two procedures
  - A minor optimization in a heavily executed line of code can have a great effect on overall runtime
  - Typical of engineering and scientific codes built around matrix solutions
- Flat profile
  - Runtime spread fairly evenly across many routines

Basic Block Profilers

- A basic block is a section of code with only one entrance and one exit.
- If you know how many time the basic block was entered, you know how many times each of the statements in the block was executed, which gives you a line-by-line profile.

Virtual Memory

- The virtual memory system can slow your program down if it is too large to fit in physical memory or is computing with other jobs for memory resources.
- Use vmstat (or similar utility) to monitor paging activity.

Trace-based Tools

- Collect timestamped trace records during runtime
- Often implemented using an MPI profiling library
- Display timeline view and/or statistical analysis of program execution
### Performance Analysis Tools

- DEEP/MPI
- MPE logging/Jumpshot
- Pablo Performance Analysis Tools
- Paradyn
- Scalea
- TAU
- Vampir
- VProf

### DEEP/MPI

- [http://www.psrv.com/DEEP mpi top.html](http://www.psrv.com/DEEP mpi top.html)
- Compile MPI program with *mpirun* compile driver
- Execute program
- View performance data using the DEEP/MPI interface
- Support the PAPI interface to hardware counters
- Can be used to analyze mixed MPI and shared memory (e.g., OpenMP) programs

### MPE Logging/Jumpshot

- Distributed with MPICH
- Developed for use with MPICH but can be used with other MPICH implementations
- Two log file formats
  - CLOG
  - SLOG
- Timeline view

### Pablo Performance Analysis Tools

- [http://www-pablo.cs.uiuc.edu](http://www-pablo.cs.uiuc.edu)
- Base library plus extensions for
  - I/O
  - MPI
  - MPI I/O
- Generates trace records in SDDF (Self Defining Data Format)
- SvPablo
  - Graphical user interface for instrumenting source code and browsing runtime performance data

### Paradyn

- [http://www.cs.wisc.edu/paradyn/](http://www.cs.wisc.edu/paradyn/)
- Dynamically inserts instrumentation into a running application and analyzes and displays performance information in real-time
- Performance Consultant attempts to automatically identify performance problems

### Scalea

- [http://www.par.univie.ac.at/project/scalea/](http://www.par.univie.ac.at/project/scalea/)
- Scalea Instrumentation System (SIS)
- SISPROFILING Library
- Analysis and Visualization System
  - Profile/Trace analysis
  - Overhead analysis
  - Multiple experiments analysis
TAU
- Tuning and Analysis Utilities
- http://www.acl.lanl.gov/tau/
- Portable profiling and tracing toolkit
- Supports MPI, OpenMP, and mixed MPI/OpenMP

Vampir
- http://www.pallas.de/pages/vampir.htm
- Commercial MPI performance analysis tool from Pallas
- Vampirtrace MPI profiling library + API
- Vampir analysis GUI
  - Timeline view
  - Statistics views

VProf
- http://aros.ca.sandia.gov/~cljansson/perf/vprof/
- Stands for “Visual Profiler”
- Routines to collect statistical profiling information
- Programs to view execution profiles
  - vprof
  - cprof

Tool Evaluation Criteria
- General
  - Ease of use
  - Functionality
  - Robustness
  - Scalability
  - Portability
  - Versatility
- Specific
  - Support for hybrid environments
  - Support for distributed heterogeneous environments
  - Analysis of MPI-2 I/O