Profiling

David Newman

(From Tom Logan)

(Slides originally from
Kate Hedstrom & Tom Baring)

Topics

- **Objectives of Profiling**
- **Compiler options**
- **Hardware performance counters**
- **Manual Timing**
- **Profiling**
  - Compile for profiling
  - Look at profiler output

Performance Analysis

- **Objectives of performance analysis**
  - Quantify the efficiency with which a program uses a *specific* computer architecture.
  - Recognize existence of problems in the program
    - Bottlenecks
    - Underperforming sections of code
    - Load imbalance
  - Isolate the source of problems to guide optimization efforts
  - Evaluate effect of optimization efforts

Optimization In a Nutshell

Repeat until code is fast enough

1. profile to find (new) bottleneck
2. try to optimize
3. verify program still correct

Compiler Listings

- **Static code analysis**
  - Compiler listings show optimizations the compiler performed on the code. If you're familiar with the code, can help predict performance problems before running it.
  - Use alone on simple programs
  - Typically used with dynamic performance analysis tools.
- **No overhead**

PGI Information Flags

- **-Minfo[=option[,option...]]**
  - Emits useful information to stderr
  - =mp prints OpenMP information
  - =par prints loop parallelization info
  - =all prints verbose vectorizer info to stdout

PGI Optimization Flags

-Ox	optimization level, from 0 to 3 (2 is default; can be used with –g)
-Mipa	inter-procedural analysis
-fast	highest level of optimization includes O3, IPA, and much more

gfortran (and gcc)

- Debugging Options
  -dletters -dumpspecs -dumpmachine -dumpversion -fsanitize=style -fsanitize-recover -fsanitize-recover=style -fasan-shadow-offset=number -fsanitize-undefined-trap-on-error -fcheck-pointer-bounds -fchkp-check-incomplete-type -fchkp-first-field-has-own-bounds
  -fchkp-narrow-bounds -fchkp-narrow-to-innermost-array -fchkp-optimize -fchkp-use-fast-string-functions -fchkp-use-nochk-string-functions -fchkp-use-static-bounds -fchkp-use-static-const-bounds -fchkp-treat-zero-dynamic-size-as-infinite -fchkp-check-read -fchkp-check-read
  -fchkp-check-write -fchkp-store-bounds -fchkp-instrument-calls -fchkp-instrument-marked-only -fchkp-use-wrappers -fdbg-cnt-list -fdbg-cnt=counter-value-list -fdisable-ipa-pass_name -fdisable-rtl-pass_name -fdisable-rtl-pass-name=range-list -fdisable-tree-pass_name
  -fdisable-tree-pass-name=range-list -fdump-noaddr -fdump-unnumbered -fdump-unnumbered-links -fdump-translation-unit[-n] -fdump-class-hierarchy[-n] -fdump-ipa-all -fdump-ipa-cgraph -fdump-ipa-inline -fdump-passes -fdump-statistics -fdump-tree-all -fdump-tree-original[-n]
  -fdump-tree-optimized[-n] -fdump-tree-cfg -fdump-tree-alias -fdump-tree-ch -fdump-tree-ssa[-n] -fdump-tree-pre[-n] -fdump-tree-ccp[-n] -fdump-tree-dce[-n] -fdump-tree-gimple[-raw] -fdump-tree-dom[-n] -fdump-tree-dse[-n] -fdump-tree-phiprop[-n] -fdump-tree-phiopt[-n]
  -fdump-tree-forwprop[-n] -fdump-tree-copyrename[-n] -fdump-tree-nrv -fdump-tree-vect -fdump-tree-sink -fdump-tree-sra[-n] -fdump-tree-forwprop[-n] -fdump-tree-fre[-n] -fdump-tree-vtable-verify -fdump-tree-vrp[-n] -fdump-tree-storeccp[-n] -fdump-final-insns=file
  -fcompare-debug[=opts] -fcompare-debug-second -feliminate-dwarf2-dups -fno-eliminate-unused-debug-types -feliminate-unused-debug-symbols -femit-class-debug-always -fenable-kind-pass -fenable-kind-pass=range-list -fdebug-types-section -fmem-report-wpa -fmem-report
  -fpre-ipa-mem-report -fpost-ipa-mem-report -fprofile-arcs -fopt-info -fopt-info-options[=file] -frandom-seed=number -fsched-verbose=n -fsel-sched-verbose -fsel-sched-dump-cfg -fsel-sched-pipelining-verbose -fstack-usage -ftest-coverage -ftime-report -fvar-tracking
  -fvar-tracking-assignments -fvar-tracking-assignments-toggle -g -glevel -gtoggle -gcoff -gdwarf-version -ggdb -grecord-gcc-switches -gno-record-gcc-switches -gstabs -gstabs+ -gstrict-dwarf -gno-strict-dwarf -gvms -gxcoff -gxcoff+ -gz[=type]
  -fno-merge-debug-strings -fno-dwarf2-cfi-asm -fdebug-prefix-map=old=new -femit-struct-debug-baseonly -femit-struct-debug-reduced -femit-struct-debug-detailed[=spec-list] -p -pg -print-file-name=library -print-libgcc-file-name -print-multi-directory -print-multi-lib
  -print-multi-os-directory -print-prog-name=program -print-search-dirs -Q -print-sysroot -print-sysroot-headers-suffix -save-temps -save-temps=cwd -save-temps=obj -time[=file]

  Optimization Options
  -faggressive-loop-optimizations -falign-functions[=n] -falign-jumps[=n] -falign-labels[=n] -falign-loops[=n] -fassociative-math -fauto-profile -fauto-profile[=path] -fauto-inc-dec -fbranch-probabilities -fbranch-target-load-optimize -fbranch-target-load-optimize2
  -fbtr-bb-exclusive -fcaller-saves -fcheck-data-deps -fcombine-stack-adjustments -fconserve-stack -fcompare-elim -fcprop-registers -fcrossjumping -fcse-follow-jumps -fcse-skip-blocks -fcx-fortran-rules -fcx-limited-range -fdata-sections -fdce -fdelayed-branch
  -fdelete-null-pointer-checks -fdevirtualize -fdevirtualize-speculatively -fdevirtualize-at-ltrans -fdse -fearly-inlining -fipa-sra -fexpensive-optimizations -ffat-lto-objects -ffast-math -ffinite-math-only -ffloat-store -fexcess-precision=style -fforward-propagate
  -ffp-contract=style -ffunction-sections -fgcse -fgcse-after-reload -fgcse-las -fgcse-lm -fgraphite-identity -fgcse-sm -fhoist-adjacent-loads -fif-conversion -fif-conversion2 -findirect-inlining -finline-functions -finline-functions-called-once -finline-limit=n
  -finline-small-functions -fipa-cp -fipa-cp-clone -fipa-cp-alignment -fipa-pta -fipa-profile -fipa-pure-const -fipa-reference -fipa-icf -fira-algorithm=algorithm -fira-region=region -fira-hoist-pressure -fira-loop-pressure -fno-ira-share-save-slots -fno-ira-share-spill-slots
  -fira-verbose=n -fisolate-erroneous-paths-dereference -fisolate-erroneous-paths-attribute -fivopts -fkeep-inline-functions -fkeep-static-consts -flive-range-shrinkage -floop-block -floop-interchange -floop-strip-mine -floop-unroll-and-jam -floop-nest-optimize
  -floop-parallelize-all -flra-remat -flto -flto-compression-level -flto-partition=alg -flto-report -flto-report-wpa -fmerge-all-constants -fmerge-constants -fmodulo-sched -fmodulo-sched-allow-regmoves -fmove-loop-invariants -fno-branch-count-reg -fno-defer-pop
  -fno-function-cse -fno-guess-branch-probability -fno-inline -fno-math-errno -fno-peephole -fno-peephole2 -fno-sched-interblock -fno-sched-spec -fno-signed-zeros -fno-toplevel-reorder -fno-trapping-math -fno-zero-initialized-in-bss -fomit-frame-pointer
  -foptimize-sibling-calls -fpartial-inlining -fpeel-loops -fpredictive-commoning -fprefetch-loop-arrays -fprofile-report -fprofile-correction -fprofile-

Hardware Performance Monitors

- **Dynamic performance analysis**
- **Report overall performance statistics for a run of a program. E.g.,**
  - cpu time
  - cache misses per second
  - I/O reads/writes
  - MIPS
  - MFLOPS
- **Generally no overhead**
- **Available on PACMAN via PAPI**

Manual Timings

- **Programmer inserts "stopwatch" timers into the code to report time spent in specific regions of interest.**
- **Simplicity and self-reliance**
- **Valuable as a permanent component of large, complex codes.**
  - Track performance through the years, compiler versions, and different architectures
  - On-going monitoring of program health
- **Overhead**

Manual Timings (cont.)

- **Some timing routines:**
  - Fortran 90:
    - system_clock, date_and_time
  - System calls:
    - irtc, second, gettimeofday

Manual Timings Example

```
integer cnt, rate
…
do day = 1  , ndays
   …
   call system_clock (COUNT=cnt, COUNT_RATE=rate)
   t_start = REAL (cnt) / REAL (rate)

   call ice_model ()

   call system_clock (COUNT=cnt, COUNT_RATE=rate)
   t_end = REAL (cnt) / REAL (rate)

   t_ice_model = t_ice_model + (t_end - t_start)
   …
enddo
--------
print*,”Total time in ice_model”, t_ice_model
```

Profiling

- **Dynamic analysis, of running code**
- **1) Gather data and 2) produce reports on specific regions of the code, e.g.,**
  - subroutine / function
  - arbitrary user-defined regions
- **Specialized tools collect data on and generate reports on specific activity, e.g.**
  - MPI communication
  - I/O

Profiling (cont.)

- **Data Collection:**
  - Trace data
    - every entry into every region of interest is counted
  - Profile data
    - sampling of CPU's program counter to generate statistical approximation of time spent in different regions
- **Overhead**

Profiling (cont.)

- **Report generation**
  - "flat report"
  - "call tree" or "call graph"
  - Graphical reports
  - GUI interface

Basic Profiling

- Compile with -g -pg as well as usual -Ox
- Run as normal (some overhead)
- Produces gmon.out - look at with gprof [or other tools]
- Tells what routines are taking all the time

gprof

% time	cum sec	self sec	calls	name
25.6	403	403	11400	step2d
10.2	565	161	201	t3dmix2
9.4	713	148	201	pre_step3d
7.4	830	117	201	lmd_skpp
5.4	914	84	201	prsgrd
4.5	985	70	200	step3d_t
4.4	1055	70	201	rhs3d

TAU & PAPI

- **TAU & PAPI are performance monitoring/profiling applications available on PACMAN**
- **Not enough time to cover usage**
- **Check out newsletters 387, 381, and 385 for articles on usage**
- **Check out $SAMPLES_HOME on PACMAN for usage examples**

Summary of Profiling

- Find out how fast the code is
- Try different compiler options to see what optimization level works best for you (and gives the right answer)
- Find out where the code is spending all its time
- Look at the source listing to see what is going on there - see if it makes sense or can be sped up