SE-292 High Performance Computing

Profiling and Performance

R. Govindarajan
govind@serc

Performance Measurement and Tuning

- Tools to help you measure the performmance of program
- Determining program execution time

```
% time a.out
real    0m0.019s
user    0m0.014s
system  0m0.002s
```

  - Gives elapse time, user, and system
- Tools to identify the important parts of your program for perf. Improvement

  Concentrate optimization efforts on those parts

2

Amdahl's Law

- Which part of the program to optimize?
- Amdahl's Law:

  Speedup is limited by the part of program which does not benefit by the optimization

  IOW, Sp ≈ 1/s !
- Implies concentrate on part of the program where maximum time is spent!

3

Timing

Timing: measuring the time spent in specific parts of your program

- Examples of `parts': Functions, loops, …
- Recall: Different kinds of time that can be measured (real/wallclock/elapsed vs virtual/CPU)

1. Decide
   - which time you are interested in measuring
   - at what granularity
2. Find out what mechanisms are available and their granularity of measurement

4

Timing Mechanisms

- `gettimeofday`
  - Real time in seconds and microseconds since 00:00 1/1/1970
  - Q: Overflow of 32b second value?
- `getrusage`
- `times` system call
- High resolution timers
  - Example: `gethrtime`

5

Profiling

- Profiler: tool that helps you identify the `important' parts of your program to concentrate your optimization efforts
- Profile: breakup (of execution time) across different parts of the program
- Can be done by adding statements to your program (instrumentation) -- so that during execution, data is gathered, outputted and possibly processed later
- Automation: where a profiling tool adds those instructions into your program for you

6

Profiling Mechanisms

- Levels of Granularity typically supported
  - Function level
  - Statement level
  - Basic block level: A basic block is a sequence of contiguous instructions in a program with a single entry point (the first instruction in the basic block) and a single exit point (the last instruction in the basic block)
- Two kinds of profile data
  - execution time
  - execution counts
- We will look at examples of profiling mechanisms at the function and basic block level

7

Prof: UNIX Function Level Profiling

- Usage

```
% cc –p program.c  /generates instrumented a.out
% a.out            / execution; instrumentation
                    / generates data and mon.out
% prof             / processing of profile data
```

- Output gives a function by function breakup of execution time
- Useful in identifying which functions to concentrate optimization efforts on

8

Output:

%Time	Seconds	CumSecs	#Calls	Name
56.8	0.50	0.50	1000	_baz
27.4	0.24	0.74	1000	_bar
15.9	0.14	0.88	500	_foo
…				
0.0	0.00	0.88	1	_main
0.0	0.00	0.88	3	_strcpy

9

Prof: How it Works

- Instrumentation does three things
  1. At entry of each function: increment an execution count for that function
  2. At program entry: make a call to system call **profil** to get execution times
  3. At program exit: write profile data to output file that can later be processed by prof
- profil(): execution time profiler
  - Generates an execution time histogram, execution time in each function

10

Profil: What it does

- One of the parameters in call to profil is a buffer
- Used as an array of counters initialized to 0
- Array elements are associated with contiguous regions of program text
- During execution, PC value is sampled (once every clock tick, default: 10 msec); triggered on timer interrupt
- Corresponding buffer element is incremented
- Later associated with a function; time weight of 10 msec used to estimate CPU times

11

Using prof

- From how it works, we understand that
  - Granularity is at best 10 msec
  - Generated profile could differ for multiple runs of a program with same input!
  - Could be completely wrong; observe that there could be a particular function that just happens to be running each time the timer interrupt occurs
- Some usage guidelines
  - Run under light load conditions
  - Run a few times and see if results vary a lot
  - Note that function execution counts are exact, while execution times are estimates

12