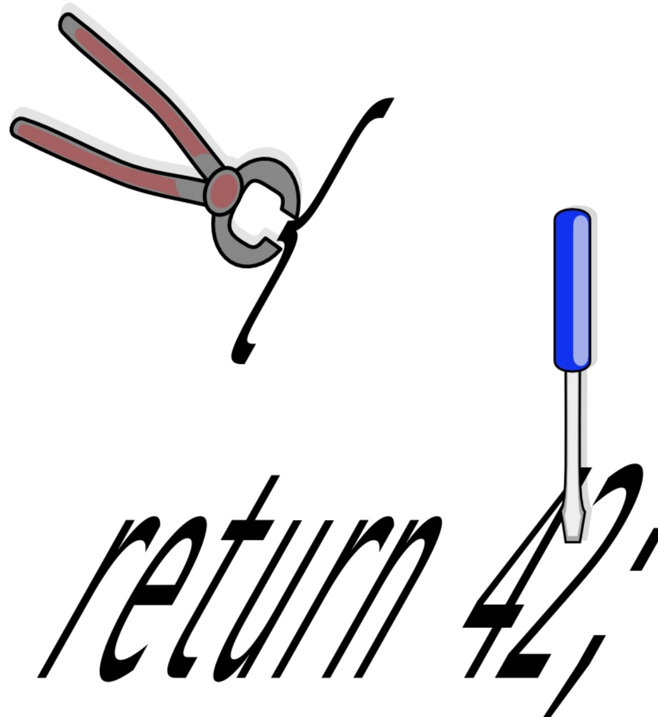
IRTG Lecture Week Bergen 2009
Programming of Multicore Systems
-
Programming Tools: Profiling



Programming & Profilers

- Often programs are too slow
  - Too much processing time taken up by parts of code
- These processing intensive parts have to be identified

- Profilers are programs to help in that task

Profilers

- **Profilers allow to**
  - **Determine call graph hierarchy of a program**
    - Which function was called by another function
    - By which other function was a function called
    - How often a function called another function
  - **See how much processing time was used for (almost) each source line**

Profiler

- **What is profiling**
  - **Profiling, is the investigation of a program's behavior using information gathered as the program executes (i.e. a form of dynamic program analysis).**
    - (Paraphrased from Wikipedia)
- **What is a profiler**
  - **A profiler is a performance analysis tool that measures the behavior of a program as it executes, particularly the frequency and duration of function calls.**
    - (Paraphrased from Wikipedia)
  - **A profiler can be used to show you where your programs take up the most time.**
    - (Paraphrased from me)

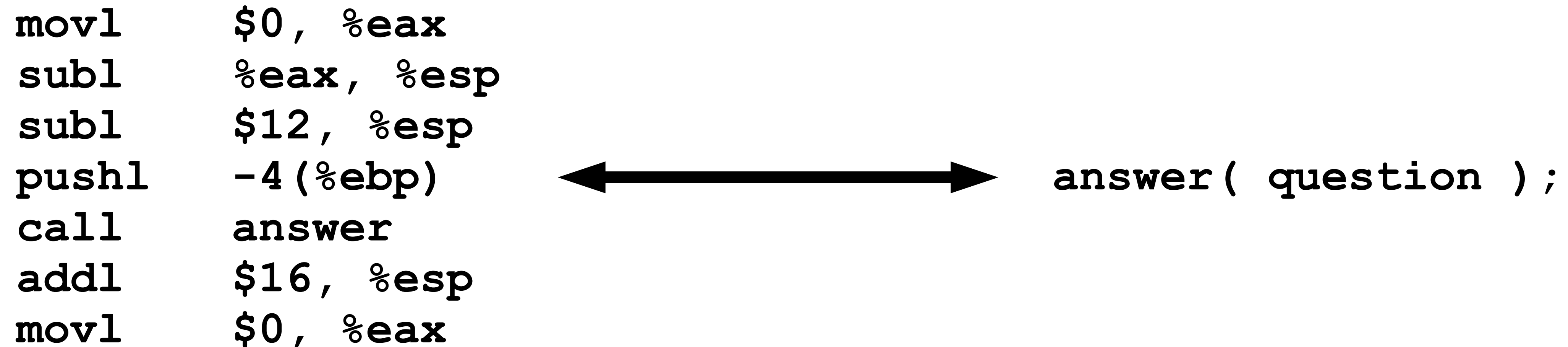
Find Your Inefficient Programming For You

What a Profiler Does

But it can show you likely locations of bottlenecks

Requirements

- **Profilers typically require same information about program as for a debugger**
  - Where variables and functions are located in memory
  - Link between generated object code and original source code



  - Called Debugging Information
    - Provided by `gcc/g++` with `-ggdb3` switch

Valgrind

- **Profiling tool covered here:**
  - **valgrind**
- **Powerful instrumentation framework**
- **Multiple tools available**
  - **Memory checker to detects memory leaks and incorrect accesses (e.g. array boundaries)**
  - **Cache profiler to determine cache hits/misses**
  - **Call graph instruction based profiler to determine execution times for code**
    - Covered here
  - **Race condition detector for multi-threaded programs**
  - **...**

Valgrind

- **Version 3.3.0 installed in HLT cluster**
  - **/opt/HLT/tools/valgrind/valgrind-3.3.0**
  - **Script to set environment variables provided**
  - `. /opt/HLT/tools/valgrind/valgrind-3.3.0/bin/setenv.sh`
    - (Has to be sourced)
  - **Call graph profiler called as valgrind tool callgrind**
    - `valgrind --tool=callgrind`

Calling Valgrind

- **`valgrind --time-stamp=yes --log-file=profiler-test-%p-`date +%Y%m%d.%H%M%S`-valgrind.log \ --tool=callgrind --callgrind-out-file=profiler-\ test-%p-`date +%Y%m%d.%H%M%S`-callgrind.out \ examined-program program-arguments`**
  - `--time-stamp=yes`: **Include timestamps in log files**
  - `--log-file=profiler-test-%p-`date +%Y%m%d.%H%M%S`-valgrind.log`: **Name for log file**
  - `--tool=callgrind`: **Use the callgrind tool of valgrind**
  - `-- callgrind-out-file=profiler-test-%p-`date +%Y%m%d.%H%M%S`-callgrind.out`: **Name for output measurement file**
  - `examined-program program-arguments`: **Name of executable to be run with arguments**
  - **Output file names include PID (`%p`) and current timestamp**

Valgrind internals

- **Valgrind internally emulates CPU**
  - **(Can use arbitrary cache hierarchies for cache profiling if needed)**
  - **Can trap/profile essentially any event**
  - **Runs significantly slower than in non-profiling mode**
    - Factor 50 observed
  - **No special compilation information necessary**

Looking at results

- **In text form:**
  - `callgrind_annotate profiler-test-<PID>-<YYYYMMDD.HHMMSS>-callgrind.out`
    - Replace <PID> by PID, <YYYYMMDD.HHMMSS> by date in filename
    - Gives list of functions sorted by exclusive CPU time
      - Not including time of called functions
  - `callgrind_annotate –inclusive=yes profiler-test-<PID>-<YYYYMMDD.HHMMSS>-callgrind.out`
    - Gives list of functions sorted by inclusive CPU time
      - Does include time of functions called by listed function

Looking at results

- More intuitive using GUI
  - `kcachegrind`
- Provides exclusive & inclusive time
- Call graph hierarchy
- Direct source code view
  - (Relative) Time spent in each line of code

Kcachegrind GUI

NX - timm@dev0:1005 - dev0 (GPL Edition)

profiler-test-11803-20081201.151517-callgrind.out [/profiler-test-1 -seed 1227873456] - KCachegrind

File View Go Settings Help

Inclusive Time

Exclusive Time

Annotated Source

Call Graph

Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000...	...so
99.98	0.00	1	0x0000000000400F00	profiler-test-1: start.S
99.98	0.00	1	(below main)	libc-2.3.6.so
99.98	0.00	1	main	profiler-test-1: profiler-test-1...
99.98	4.91	100	Process	profiler-test-1: profiler-test-1...
43.78	5.07	200	Count16BitWords	profiler-test-1: profiler-test-1...
38.56	6.15	31 ...	std::vector<unsigned, std:...	profiler-test-1: stl_vector.h
23.83	5.88	25 3...	std::vector<unsigned, std:...	profiler-test-1: stl_vector.h
13.54	8.61	31 890 000	...	...stl_iterator.h
12.61	6.16	19 168 900	...	...ctor.h
9.43	5.33	31 890 000	...	...r.h
9.43	5.33	31 890 00...	...	...h
9.14	5.88	25 382 20...	...	...r.h
8.20	8.20	63 780 000	...	...ator.h
6.53	6.53	50 792 600	...	...iterator.h
6.53	3.27	25 385 800	std::vector...	...test-1: stl_vector.h
6.16	4.44	19 168 900	__gnu_cxx::new_allocator<...	profiler-test-1: new_allocator.h
5.38	3.30	13 492 100	bool __gnu_cxx::operator!...	profiler-test-1: stl_iterator.h
4.92	4.92	63 780 000	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
4.60	2.89	13 230 600	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
3.60	1.89	13 337 000	std::vector<StatData16Bit...	profiler-test-1: stl_vector.h
3.43	3.43	26 677 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
2.29	2.29	25 382 200	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
2.08	2.08	26 991 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
1.74	1.74	19 296 700	operator new(unsigned lon...	profiler-test-1: new
1.20	1.20	13 333 000	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
0.28	0.01	3 400	std::vector<unsigned, std:...	profiler-test-1: vector.tcc
0.24	0.02	51 200	std::vector<StatData16Bit...	profiler-test-1: stl_vector.h
0.24	0.00	6 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.24	0.00	6 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.24	0.00	6 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.23	0.00	7 000	__gnu_cxx::__normal_itera...	profiler-test-1: stl_algobase.h
0.23	0.00	7 000	__gnu_cxx::__normal_itera...	profiler-test-1: stl_algobase.h
0.23	0.00	7 000	unsigned* std::__copy_aux...	profiler-test-1: stl_algobase.h
0.23	0.00	7 000	unsigned* std::__copy<tru...	profiler-test-1: stl_algobase.h
0.22	0.00	7 000	memmove	libc-2.3.6.so
0.22	0.22	2 800	0x0000000000072990	libc-2.3.6.so
0.21	0.00	1 900	std::vector<StatData16Bit...	profiler-test-1: vector.tcc
0.11	0.00	3 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.11	0.00	3 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.11	0.02	3 800	__gnu_cxx::__normal_itera...	profiler-test-1: stl_uninitializ...
0.07	0.00	1 900	void std::_Destroy<__gnu_...	profiler-test-1: stl_construct.h
0.07	0.00	1 900	void std::_Destroy<__gnu_...	profiler-test-1: stl_construct.h
0.07	0.01	1 900	void std::__destroy_aux<...	profiler-test-1: stl_construct.h
0.03	0.03	231 700	__gnu_cxx::__normal_itera...	profiler-test-1: stl_iterator.h
0.03	0.01	104 500	std::vector<StatData16Bit...	profiler-test-1: stl_vector.h
0.02	0.02	76 600	void std::_Construct<Stat...	profiler-test-1: stl_construct.h

Count16BitWords

Types | Callers | All Callers | Source | Callee Map

#	Ir	Source ('/afsuser/timm/src/Workshop-HLT-CERN-200812/Profiling/standalone/profiler-...cpp')
52		{
53		StatData16Bit sd;
54	0.00	sd.fValue = value;
55	0.00	sd.fCount=1;
56	0.00	stats.push_back(sd);
	0.54	51200 calls to 'std::vector<StatData16Bit, std::allocator<StatData16Bit> >::push_back' (profiler-test-1: stl_vector.h)
57		}
58		}
59		
60	0.00	std::vector<StatData16Bit>::iterator iter = stats.begin();
	0.00	200 calls to 'std::vector<StatData16Bit, std::allocator<StatData16Bit> >::begin()' (profiler-test-1: stl_vector.h)
61	0.01	while (iter != stats.end())
	0.05	51400 calls to 'bool __gnu_cxx::operator!=<StatData16Bit*, std::vector<StatData16Bit, std::allocator<StatData16Bit> > ...
	0.03	51400 calls to 'std::vector<StatData16Bit, std::allocator<StatData16Bit> >::end()' (profiler-test-1: stl_vector.h)
62		{
63	0.01	unsigned short value = iter->fValue;
	0.01	51200 calls to '__gnu_cxx::__normal_iterator<StatData16Bit*, std::vector<StatData16Bit, std...
64	0.01	if (counts.size() <= value)
	0.14	51200 calls to 'std::vector<unsigned, std::allocator<unsigned> >::size' (profiler-test-1:...
65		{
66	2.98	while (counts.size() <= value)

main 100.00 %

100.00 %

Process 100.00 %

100.00 %

Count16BitWords 100.00 %

12.15 % | 10.50 % | 35.13 % | 8.23 % | 18.93 %

__gnu_cxx::__normal_iterator<StatData16Bit*, std::vector<StatData16Bit, std::...

std::vector<unsigned, std::allocator<unsigned> >::size

std::vector<StatData16Bit, std::allocator<StatData16Bit> >::end()

12.33 % | 8.60 % | 8.60 %

__gnu_cxx::__normal_iterator<unsigned

std::vector<unsigned, std::allocator<

std::vector<unsigned, std::allocator<

Caller Map | Parts | Call Graph | Callees | All Callees | Assembler

profiler-test-11803-20081201.151517-callgrind.out [1] - Total Instruction Fetch Cost: 7 773 991 105

1 2 3 4 5 6 7 8 timm@ms0: ~ | profiler-test-1.cpp | profiler-test-11803-20081201.151517-callg... | 17:10:59

KcacheGrind Hint for HLT

- **Need to specify source directory/-ies**
- **Menu *Settings***
  - ***Configure KcacheGrind...***
    - Dialog Tab *Annotations*
      - Button *Add*
- **Function unfortunately unstable in version installed in cluster**

Links

- http://valgrind.org/
- http://valgrind.org/docs/manual/QuickStart.html
- http://valgrind.org/docs/manual/manual.html