Lecture 2, Benchmarks

The best method of measuring a computers performance is to use benchmarks. Some suggestions from my personal experience preparing a benchmark suite and several updates and personal benchmark experience are presented in pdf format. Lecture 2 Smaller time is better, higher clock frequency is better. time = 1 / frequency T = 1/F and F = 1/T 1 nanosecond = 1 / 1 GHz 1 microsecond = 1 / 1 MHz Definitions: CPI Clocks Per Instruction MHz Megahertz, millions of cycles per second MIPS Millions of Instructions Per Second = MHz / CPI MOPS Millions of Operations Per Second MFLOPS Millions of Floating point Operations Per Second MIOPS Millions of Integer Operations Per Second Do not trust your computers clock or the software that reads and processes the time. First: Test the wall clock time against your watch. time_test.c time_test.java time_test.f90 The program displays 0, 5, 10, 15 ... at 0 seconds, 5 seconds, 10 seconds etc.

demonstrate time_test if possible

Note the use of <time.h> and 'time()' Beware, midnight is zero seconds. Then 60 sec/min * 60 min/hr * 24 hr/day = 86,400 sec/day Just before midnight is 86,399 seconds. Running a benchmark across midnight may give a negative time. Then: Test CPU time, this should be just the time used by the program that is running. With only this program running, checking against your watch should work. time_cpu.c The program displays 0, 5, 10, 15 ... at 0 seconds, 5 seconds, 10 seconds etc. Note the use of <time.h> and '(double)clock()/(double)CLOCKS_PER_SEC' I have found one machine with the constant CLOCKS_PER_SECOND completely wrong and another machine with a value 64 that should have been 100. A computer used for real time applications could have a value of 1,000,000 or more. More graphs of FFT benchmarks The source code, C language, for the FFT benchmarks:

Note the check run to be sure the code works. Note the non uniform data to avoid special cases. fft_time.c main program fftc.h header file FFT and inverse FFT for various numbers of complex data points The same source code was used for all benchmark measurements. These were optimized for embedded computer use where all constants were burned into rom. fft16.c ifft16.c fft32.c ifft32.c fft64.c ifft64.c fft128.c ifft128.c fft256.c ifft256.c fft512.c ifft512.c fft1024.c ifft1024.c fft2048.c ifft2048.c fft4096.c ifft4096.c Some of the result files: P1-166MHz P1-166MHz -O2 P2-266MHz P2-266MHz -O2 Celeron-500MHz P3-450MHz MS P3-450MHz Linux PPC-2.2GHz PPC-2.5GHz P4-2.53GHz XP Alpha-533MHz XP Xeon-2.8GHz Athlon-1.4GHz MS Athlon-1.4GHz XP Athlon-1.4GHz SuSe What if you are benchmarking a multiprocessor? For example, a two core or quad core, then use both CPU time and wall time to get average processor loading: time_mp2.c for two cores time_mp4.c for quad cores time_mp8.c for two quad cores time_mp12.c for two six cores The output from a two cores is: time_mp2.out for two core Xeon The output from four cores is: time_mp4.out for Mac quad G5 The output from eight cores is: time_mp8_c.out for AMD 12-core The output from twelve cores is: time_mp12_c.out for AMD 12-core Similar tests in Java time_test.java time_cpu.java time_mp4.java for quad cores time_mp8.java for eight cores time_mp4_java.out for quad Xeon G5 time_mp8_java.out for 8 thread Xeon G5 OK, since these were old and I did not want to change them, they give some indications of performance on various machines with various operating systems and compiler options. To measure very short times, a higher quality, double-difference method is needed. The following program measures the time to do a double precision floating point add. This may be a time smaller than 1ns, 10^-9 seconds. A test harness is needed to calibrate the loops and make sure dead code elimination can not be used by the compiler. The the item to be tested is placed in a copy of the test harness to make the measurement. The time of the test harness is the stop minus start time in seconds. The time for the measurement is the stop minus start time in seconds. The difference, thus double difference, between the harness and measurement is

the time for the item being measured. Here A = A + B with B not known to be a constant by the compiler, is reasonably expected to be a single instruction to add B to a register. If not, we have timed the full statement. The double difference time must be divided by the total number of iterations from the nested loops to get the time for the computer to execute the item once. An attempt is made to get a very stable time measurement. Doubling the number of iterations should double the time. Summary of double difference t1 saved run test harness t2 saved t3 saved run measurement, test harness with item to be timed t4 saved tdiff = (t4-t3) - (t2-t1) t_item = tdiff / number of iterations check against previous time, if not close, double iterations The source code is: time_fadd.c fadd on P4 2.53GHz fadd on Xeon 2.66GHz Some extra information for students wanting to explore their computer: Windows OS Linux OS

What is in my computer?

start cd /proc control panel cat cpuinfo system device manager processor etc.

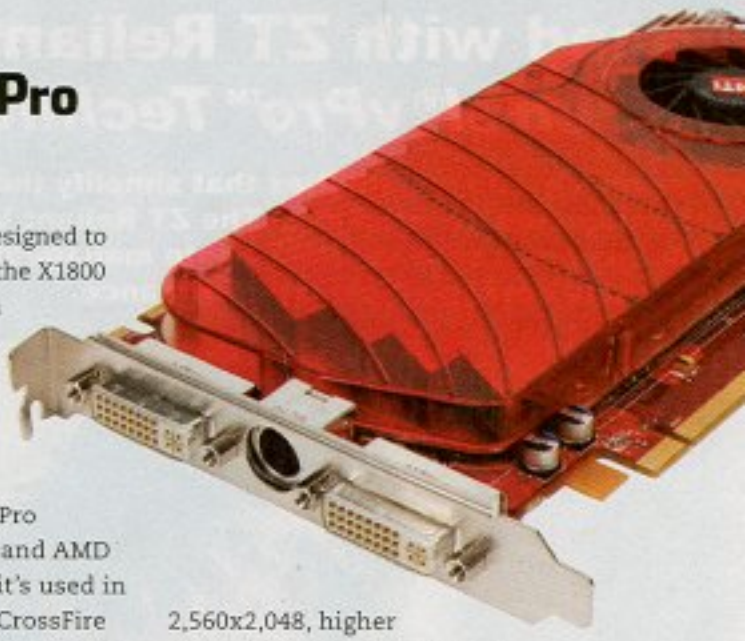
What processes are running in my computer?

ctrl-alt-del ps -el process top How do I easily time a program? command prompt time prog < input > output time prog < input > output time The time available through normal software calls may be updated less than 30 times per second to more than a million times per second. A general rule of thumb is to have the time being measured be 10 seconds or more. This will give a reasonable accurate time measurement on all computers. Just repeat what is being measured if it does not run 10 seconds. Some history about computer

time reporting. There were time sharing system where you bought time on the computer by the cpu second. There is the cpu time your program requires that is usually called your process time. There is also operating system cpu time. When there are multiple processes running, the operating system time slices, running each job for a short time, called a quanta. The operating system must manage memory, devices, scheduling and related tasks. In the past we had to keep a very close eye on how cpu time was charged to the users process verses the systems processes and was "dead time" the idle process, charged to either. From a users point of view, the user did not request to be swapped out, thus the user does not want any of the operating system time for stopping and restarting the users process to be charged to the user. Another historic tidbit, some Unix systems would add one microsecond to the time reported on each system request for the time. Never allowing the same time to be reported twice even if the clock had not updated. This was to ensure that all disk file times were unique and thus programs such as 'make' would be reliable. For more recent SPEC benchmarks, many see CPU integer benchmarks,SPECint, floating point benchmarks,SPECfp Some times you just have to buy the top of the line and forget benchmarks.

AMD ATI Radeon X1950 Pro

Impressive but pricey graphics



AMD'S ATI RADEON X1950 Pro graphics card is designed to replace the former midrange Radeon-line stalwart, the X1800 GTO. The card features some impressive numbers: a 575MHz core clock speed, and 256MB of DDR3 RAM running at 1,380MHz, faster than either the X1800 GTO or the competing nVidia GeForce 7900 GS. But you'll have to pony up for that performance, considering the X1950's $299 price.

Like most current 3D cards, the Radeon X1950 Pro requires a connection to your PC's power supply, and AMD recommends at least a 550-watt power supply if it's used in dual-card CrossFire mode. In fact, the revamped CrossFire design on this card is probably more exciting than the Radeon X1950 Pro chipset itself. With the new design, you can purchase two of the same card to double them up—you no longer need a special, more expensive CrossFire Edition card to anchor a master/slave arrangement. Also, AMD claims, because CrossFire's new connection is 24-bit, a CrossFire configuration can now be run at a resolution of 2,560x2,048, higher than the top-end resolution of 2,560x1,600 that nVidia's competing Scalable face (SLI) dual-card technology currently offers.

In our tests, the Radeon X1950 Pro stacked up against the GeForce 7900 GS, in both single- and mode. On our high-resolution Oblivion test, a pa on X1950 Pros looked especially impressive, cloc 45 frames per second on this notoriously difficu mark test. The GeForce took the lead on our Qua however, which surprised us, since Radeon card well on that test of late.

Overall, the Radeon X1950 Pro edges out the GeF GS slightly on performance. But, given the steep co Radeon, we can't recommend it over the less-expensive nVidia cards. —*Rich Brown*

EDITORS' RATING 7.0

0 BETTER ›› 10

PROS New CrossFire design is affordable, easy to set up; beats nVidia's competing card in certain tests

CONS nVidia's card wins out on some games; street price is high

Advanced Micro Devices
888-974-6728
ati.amd.com

Direct Price $299

Now find a display with 2,560 by 2,048 resolution! (other than the NASA display)