Практическая работа №4 Быстрая презентация бизнес идеи

Process time Linux is a multitasking operating system which means that even when you have a single CPU, you can run several processes at the same time. You can connect to your server via SSH and look at the output of htop while your web server is delivering the content of your blog to your readers over the internet. How is that possible when a single CPU can only execute one instruction at a time? The answer is time sharing. One process runs for a bit of time, then it is suspended while the other processes waiting to run take turns running for a while. The bit of time a process runs is called the time slice. The time slice is usually a few milliseconds so you don't really notice it that much when your system is not under high load. (It'd be really interesting to find out how long time slices usually are in Linux.) This should help explain why the load average is the average number of running processes. If you have just one core and the load average is 1.0 , the CPU has been utilized at 100%. If the load average is higher than 1.0 , it means that the number of processes wanting to run is higher than the CPU can run so you may experience slow downs or delays. If the load is lower than 1.0 , it means the CPU is sometimes idleing and not doing anything. This should also give you a clue why sometimes the running time of a process that's been running for 10 seconds is higher or lower than exactly 10 seconds. Process niceness and priority When you have more tasks to run than the number of available CPU cores, you somehow have to decide which tasks to run next and which ones to keep waiting. This is what the task scheduler is responsible for. The scheduler in the Linux kernel is reponsible for choosing which process on a run queue to pick next and it depends on the scheduler algorithm used in the kernel. You can't generally influence the scheduler but you can let it know which processes are more important to you and the scheduler may take it into account. Niceness ( NI ) is user-space priority to processes, ranging from -20 which is the highest priority to 19 which is the lowest priority. It can be confusing but you can think that a nice process yields to a less nice process. So the nicer a process is, the more it yields. From what I've pieced together by reading StackOverflow and other sites, a niceness level increase by 1 should yield a 10% more CPU time to the process. The priority ( PRI ) is the kernel-space priority that the Linux kernel is using. Priorities range from 0 to 139 and the range from 0 to 99 is real time and 100 to 139 for users. You can change the nicesness and the kernel takes it into account but you cannot change the priority. The relation between the nice value and priority is: PR = 20 + NI so the value of PR = 20 + (-20 to +19) is 0 to 39 that maps 100 to 139. You can set the niceness of a process before launching it. nice -n niceness program Change the nicencess when a program is already running with renice . renice -n niceness -p PID Here is what the CPU usage colors mean: Blue: Low priority threads (nice > 0) Green: Normal priority threads Red: Kernel threads http://askubuntu.com/questions/656771/process-niceness-vs-priority Memory usage - VIRT/RES/SHR/MEM A process has the illusion of being the only one in memory. This is accomplished by using virtual memory. A process does not have direct access to the physical memory. Instead, it has its own virtual address space and the kernel translates the virtual memory addresses to physical memory or can map some of it to disk. This is why it can look like processes use more memory than you have installed on your computer. The point I want to make here is that it is not very straightforward to figure out how much memory a process takes up. Do you also want to count the shared libraries or disk mapped memory? But the kernel provides and htop shows some information that can help you estimate memory usage. Here is what the memory usage colors mean: Green: Used memory Blue: Buffers Orange: Cache VIRT/VSZ - Virtual Image The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out and pages that have been mapped but not used. VIRT is virtual memory usage. It includes everything, including memory mapped files. If an application requests 1 GB of memory but uses only 1 MB, then VIRT will report 1 GB. If it mmap s a 1 GB file and never uses it, VIRT will also report 1 GB. Most of the time, this is not a useful number. RES/RSS - Resident size The non-swapped physical memory a task has used. RES is resident memory usage i.e. what's currently in the physical memory. While RES can be a better indicator of how much memory a process is using than VIRT , keep in mind that this does not include the swapped out memory some of the memory may be shared with other processes If a process uses 1 GB of memory and it calls fork() , the result of forking will be two processes whose RES is both 1 GB but only 1 GB will actually be used since Linux uses copyon-write. SHR - Shared Mem size The amount of shared memory used by a task. It simply reflects memory that could be potentially shared with other processes.