Click to prove you're human

You cant perform that action at this time. You cant perform that action at this time. What is this tool good for:Learn about how the Linux page cache works, and what syscalls exist tointerfere with its normal operation.Learn what hacks are necessary to intercept syscalls via their libc wrappers.What this tool is not good for:Controlling how your page cache is usedWhy do you think some random tool you found on GitHub can do better thanthe Linux Kernel?Defending against cache thrashingUse cgroups to bound the amount of memory a process has. See below orsearch the internet, this is widely known, works reliably, and does notintroduce performance penalties or potentially dangerous behavior like thistool does.Making a binary run fasternocache intercepts a bunch of syscalls and does lots of speculative work;it will slow down your binary.So then why does this tool exist?It was written in 2012, when cgroups, containerization etc. were all newthings. A decade+ on, they arent any more. Do this if you e.g. want to run a backup but dont want your system to slowdown due to page cache thrashing. If your distro uses systemd, this is very easy. Systemd allows to run aprocess (and its subprocesses) in a scope, which is a cgroup, and you canspecify parameters that get translated to cgroup limits.When I run my backups, I do:$ systemd-run --scope --property=MemoryLimit=500M --backup commandThe effect is that cache space stays bounded by an additional max 500MiB:Before:$ free -h total used free shared buff/cache availableMem: 7.5G 2.4G 1.3G 1.0G 3.7G 3.7GSwap: 9.7G 23M 9.7GDuring (notice how buff/cache only goes up by ~300MiB):free -h total used free shared buff/cache availableMem: 7.5G 2.5G 1.0G 1.1G 4.0G 3.6GSwap: 9.7G 23M 9.7G Use systemd-cgls to list the cgroups systemd creates. On my system, the abovecommand creates a group called run-u467.scope in the system.slice parentgroup; you can inspect its memory settings like this:$ mount | grep cgroup | grep memorycgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec latime,memory)$ cat /sys/fs/cgroup/memory/system.slice/run-u467.scope/memory.limit_in_bytes524288000 Install cgroup-tools and be prepared to enter your root password to initiallycreate cgroups.sudo env ppid=$$ sh -c ' cgcreate -g memory:backup ; echo 500M > /sys/fs/cgroup/memory/backup/memory.limit_in_bytes ; echo $ppid > /sys/fs/cgroup/memory/backup/tasks ;'After entering this, your shell is a member of that cgroup, and any new processspawned will belong to that cgroup, too, and inherit the memory limit. Thecgroups created like this wont be cleaned up automatically.More info: The nocache tool tries to minimize the effect an application has onthe Linux file system cache. This is done by intercepting the openand close system calls and calling posix_fadvise with thePOSIX_FADV_DONTNEED parameter. Because the library remembers whichpages (ie., 4K-blocks of the file) were already in file system cachewhen the file was opened, these will not be marked as "don't need",because other applications might need that, although they are notactively used (think: hot standby). Just type make. Then, prepend ./nocache to your command:./nocache cp -a ~/ /mnt/backup/home-$(hostname)The command make install will install the shared library, manpages and the nocache, cachestats and cachedel commandsunder /usr/local. You can specify an alternate prefix by usingmake install PREFIX=/usr.Debian packages are available, see note that nocache will only build on a system that hassupport for the posix_fadvise syscall and exposes it, too. Thisshould be the case on most modern Unices, but kfreebsd notably has nosupport for this as of now. For testing purposes, I included two small tools:cachedel calls posix_fadvise(fd, 0, 0, POSIX_FADV_DONTNEED) onthe file argument. Thus, if the file is not accessed by any otherapplication, the pages will be eradicated from the fs cache.Specifying -n will repeat the syscall the given number oftimes which can be useful in some circumstances (see below).cachestats has three modes: In quiet mode (-q), the exit statusis 0 (success) if the file is fully cached. In normal mode,the number of cached vs. not-cached pages is printed. In verbosemode (-v), an actual map is printed out, where each page that ispresent in the cache is marked with x.It looks like this:$ cachestats -v ~/somefile.mp3pages in cache: 85/114 (74.6%) [filesize=453.5K, pagesize=4K]cache map: 0: |x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x| 32: |x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x| 64: |x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x| | | | | | | | | | | | 96: | | | | | | | | | | | | | | | | |x|Also, you can use vmstat 1 to view cache statistics.For debugging purposes, you can specify a filename that nocache should logdebugging messages to via the -D command line switch, e.g. use nocache -D /tmp/nocache.log . Note that for simple testing the file /dev/stderrmight be a good choice. Without nocache, the file will be fully cached when you copy itsomewhere:$ ./cachestats ~/file.mp3pages in cache: 154/1945 (7.9%) [filesize=7776.2K, pagesize=4K]$ cp ~/file.mp3 /tmp$ ./cachestats ~/file.mp3pages in cache: 1945/1945 (100.0%) [filesize=7776.2K, pagesize=4K]With nocache, the original caching state will be preserved.$ ./cachestats ~/file.mp3pages in cache: 154/1945 (7.9%) [filesize=7776.2K, pagesize=4K]$ ./nocache cp ~/file.mp3 /tmp$ ./cachestats ~/file.mp3pages in cache: 154/1945 (7.9%) [filesize=7776.2K, pagesize=4K] The pre-loaded library tries really hard to catch all system callsthat open or close a file. This happens by "hijacking" the libcfunctions that wrap the actual system calls. In some cases, this mayfail, for example because the application does some clever wrapping.(That is the reason why __openat_2 is defined: GNU tar uses thisinstead of a regular openat.)However, since the actual fadvise calls are performed right beforethe file descriptor is closed, this may not happen if they are leftopen when the application exits, although the destructor tries to dothat.There are timing issues to consider, as well. If you consider nocache cat , in most (all?) cases the cache will not be restored. Fordiscussion and possible solutions see experience showed that in many cases you could "fix" this by doingthe posix_fadvise call twice. For both tools nocache andcachedel you can specify the number using -n, like so:$ nocache -n 2 cat ~/file.mp3This actually only sets the environment variable NOCACHE_NR_FADVISEto the specified value, and the shared library reads out this value.If test number 3 in t/basic.t fails, then try increasing this numberuntil it works, e.g.:$ env NOCACHE_NR_FADVISE=2 make testOne could also consider that the fact pages are kept mean the kernelconsiders they are hot, and decide the overhead of allocating one byteper page for mincore and the actual mincore calls are not worth it whenthe kernel actually does keep some pages when it wants to.In this case you can either run nocache with -f or set theNOCACHE_FLUSHALL environment variable to 1, e.g.:$ nocache -f cat ~/file.mp3$ env NOCACHE_FLUSHALL=1 make testBy default nocache will only keep track of file descriptors less than 2^20that are opened by your application, in order to bound its memoryconsumption. If you want to change this threshold, you can supply theenvironment variable NOCACHE_MAX_FDS and set it to a higher (or lower) value.It should specify a value one greater than the maximum file descriptor thatwill be handled by nocache. Most of the application logic is from Tobias Oetiker's patch forrsync . Note however,that rsync uses sockets, so if you try a nocache rsync, onlythe local process will be intercepted.The max-age=N response directive indicates that the response remains fresh until N seconds after the response is generated.Cache-Control: max-age=604800Indicates that caches can store this response and reuse it for subsequent requests while it's fresh.Note that max-age is not the elapsed time since the response was received; it is the elapsed time since the response was generated on the origin server.So if the other cache(s) on the network route taken by the response store the response for 100 seconds (indicated using the Age response header field), the browser cache would deduct 100 seconds from its freshness lifetime.If the max-age value is negative (for example, -1) or isn't an integer (for example, 3599.99), then the caching behavior is unspecified. Caches are encouraged to treat the value as if it were 0 (this is noted in the Calculating Freshness Lifetime section of the HTTP specification).Cache-Control: max-age=604800Age: 100s-maxageThe s-maxage response directive indicates how long the response remains fresh in a shared cache.The s-maxage directive is ignored by private caches, and overrides the value specified by the max-age directive or the Expires header for shared caches, if they are present.Cache-Control: s-maxage=604800no-cacheThe no-cache response directive indicates that the response can be stored in caches, but the response must be validated with the origin server before each reuse, even when the cache is disconnected from the origin server. If you want caches to always check for content updates while reusing stored content, no-cache is the directive to use. It does this by requiring caches to revalidate each request with the origin server.Note that no-cache does not mean "don't cache". no-cache allows caches to store a response but requires them to revalidate it before reuse. If the sense of "don't cache" that you want is actually "don't store", then no-store is the directive to use. The must-revalidate response directive indicates that the response can be stored in caches and can be reused while fresh. If the response becomes stale, it must be validated with the origin server before reuse.Typically, must-revalidate is used with max-age.Cache-Control: max-age=604800, must-revalidateHTTP allows caches to reuse stale responses when they are disconnected from the origin server. must-revalidate is a way to prevent this from happening - either the stored response is revalidated with the origin server or a 504 (Gateway Timeout) response is generated. The proxy-revalidate response directive is the equivalent of must-revalidate, but specifically for shared caches only.no-storeThe no-store response directive indicates that any caches of any kind (private or shared) should not store this response. privateThe private response directive indicates that the response can be stored only in a private cache (e.g., local caches in browsers). You should add the private directive for user-personalized content, especially for responses received after login and for sessions managed via cookies.If you forget to add private to a response with personalized content, then that response can be stored in a shared cache and end up being reused for multiple users, which can cause personal information to leak.publicThe public response directive indicates that the response can be stored in a shared cache. Responses for requests with Authorization header fields must not be stored in a shared cache; however, the public directive will cause such responses to be stored in a shared cache. In general, when pages are under Basic Auth or Digest Auth, the browser sends requests with the Authorization header. This means that the response is access-controlled for restricted users (who have accounts), and it's fundamentally not shared-cacheable, even if it has max-age.You can use the public directive to unlock that restriction.Cache-Control: public, max-age=604800Note that s-maxage or must-revalidate also unlock that restriction.If a request doesn't have an Authorization header, or you are already using s-maxage or must-revalidate in the response, then you don't need to use public.must-understandThe must-understand response directive indicates that a cache should store the response only if it understands the requirements for caching based on status code.must-understand should be coupled with no-store for fallback behavior.Cache-Control: must-understand, no-storeIf a cache doesn't support must-understand, it will be ignored. If no-store is also present, the response isn't stored.If a cache supports must-understand, it stores the response with an understanding of cache requirements based on its status code.no-transformSome intermediaries transform content for various reasons. For example, some convert images to reduce transfer size. In some cases, this is undesirable for the content provider.no-transform indicates that any intermediary (regardless of whether it implements a cache) shouldn't transform the response contents.immutableThe immutable response directive indicates that the response will not be updated while it's fresh.Cache-Control: public, max-age=604800, immutableA modern best practice for static resources is to include version/hashes in their URLs, while never modifying the resources but instead, when necessary, updating the resources with newer versions that have new version-numbers/hashes, so that their URLs are different. That's called the cache-busting pattern.