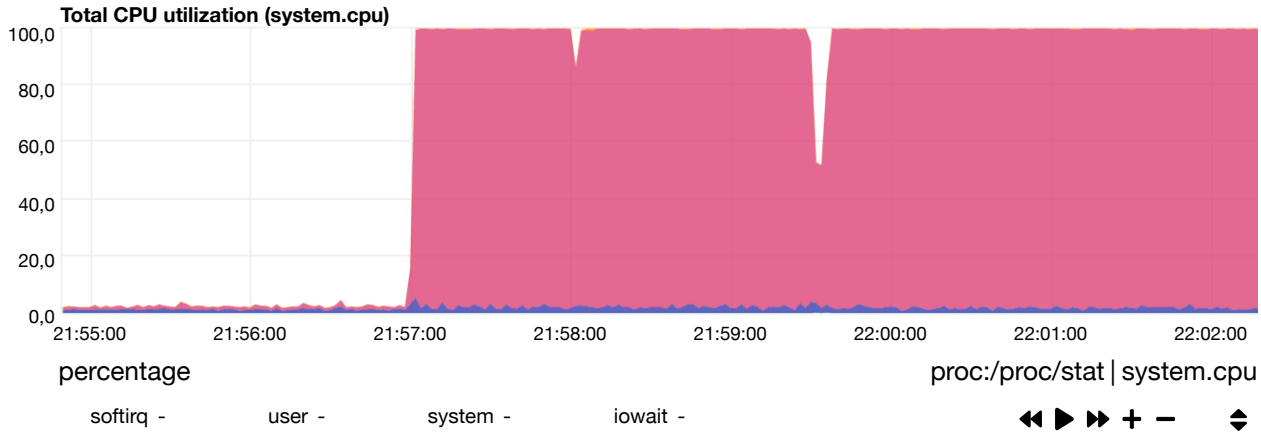
System Overview

Overview of the key system metrics.

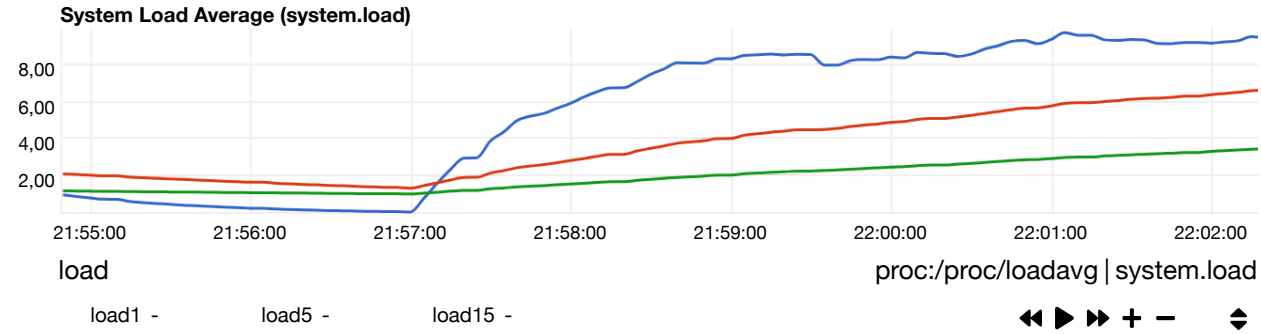
cpu

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the CPUs section and per application usage at the Applications Monitoring section.
Keep an eye on **iowait** (%). If it is constantly high, your disks are a bottleneck and they slow your system down.
An important metric worth monitoring, is **softirq** (%). A constantly high percentage of softirq may indicate network driver issues.



load

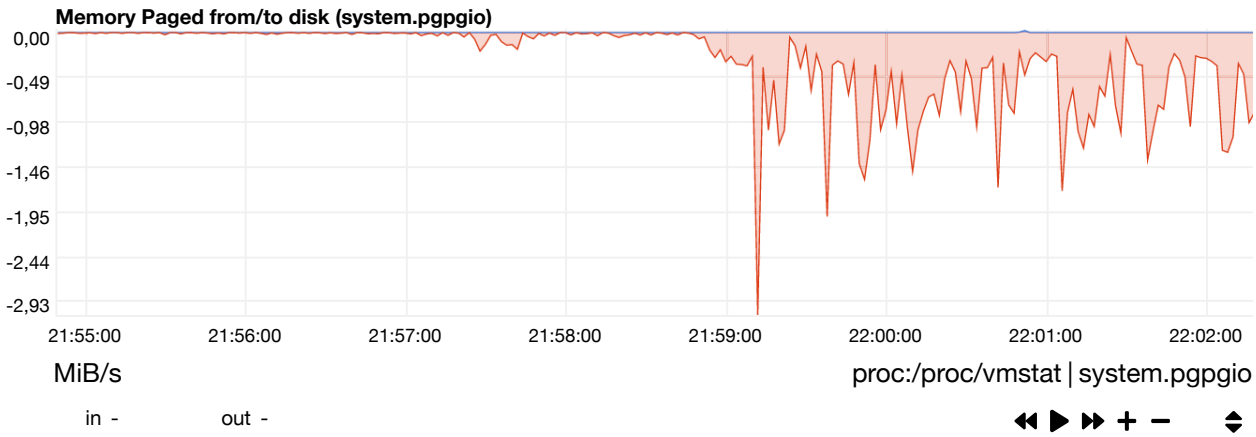
Current system load, i.e. the number of processes using CPU or waiting for system resources (usually CPU and disk). The 3 metrics refer to 1, 5 and 15 minute averages. The system calculates this once every 5 seconds. For more information check this wikipedia article (https://en.wikipedia.org/wiki/Load_(computing)).



disk

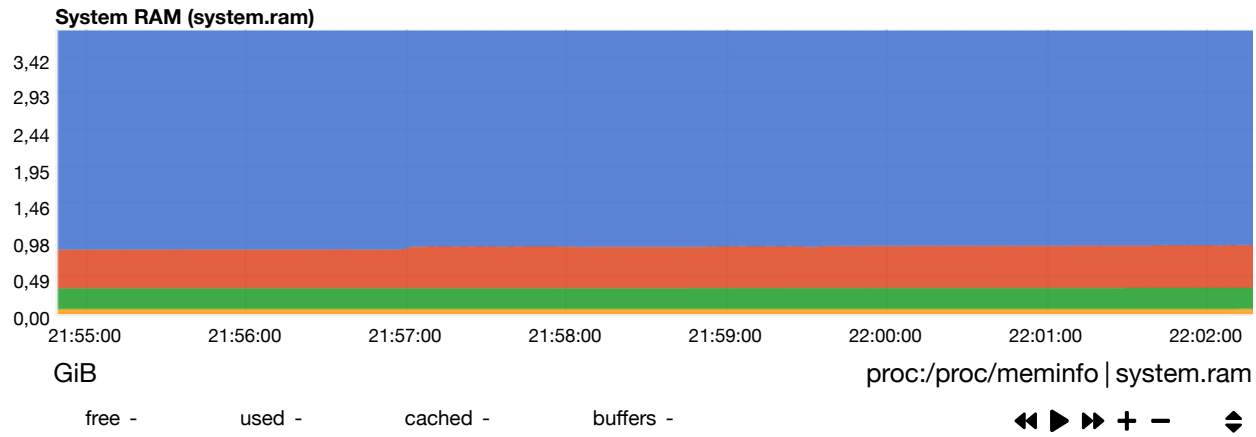
Total Disk I/O, for all physical disks. You can get detailed information about each disk at the Disks section and per application Disk usage at the Applications Monitoring section. Physical are all the disks that are listed in `/sys/block`, but do not exist in `/sys/devices/virtual/block`.

Memory paged from/to disk. This is usually the total disk I/O of the system.



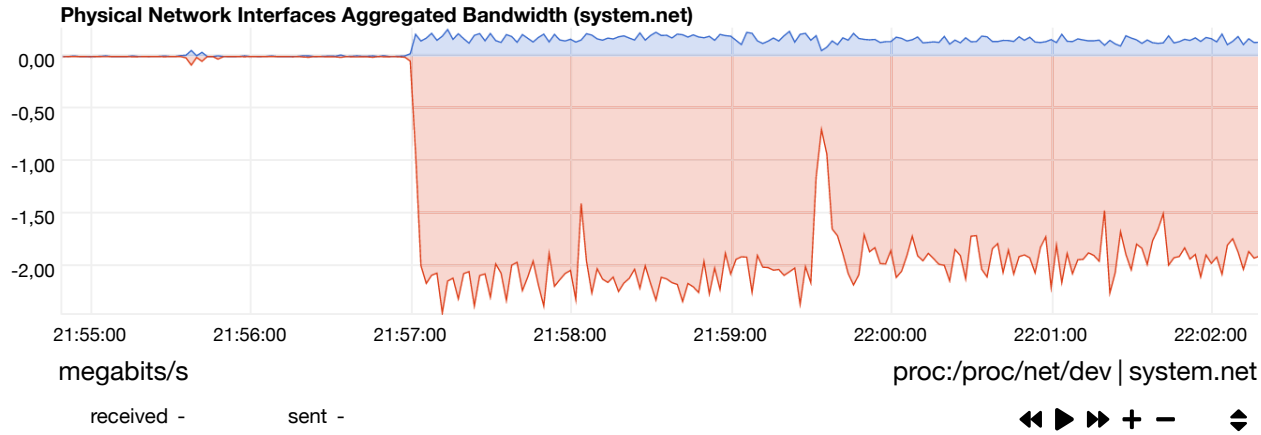
ram

System Random Access Memory (i.e. physical memory) usage.

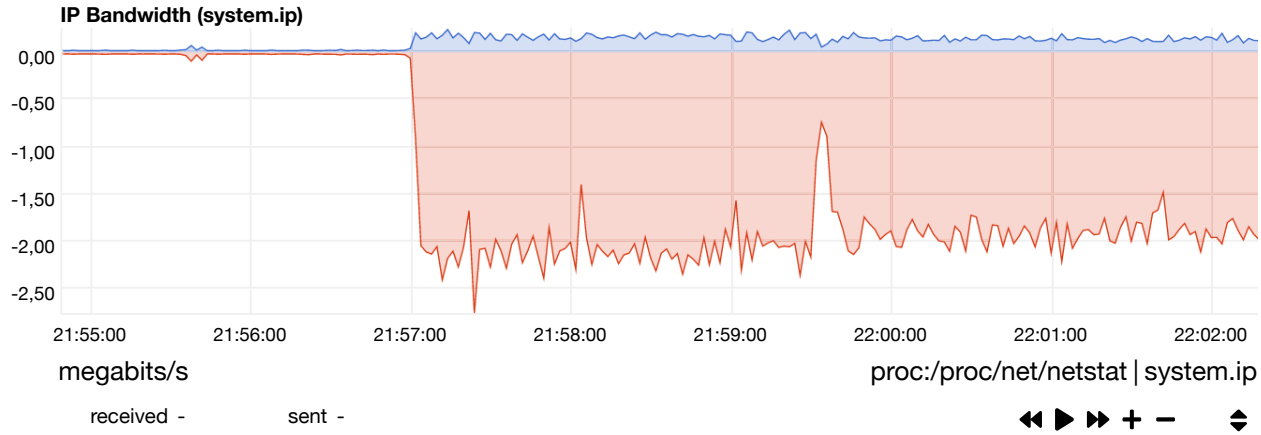


network

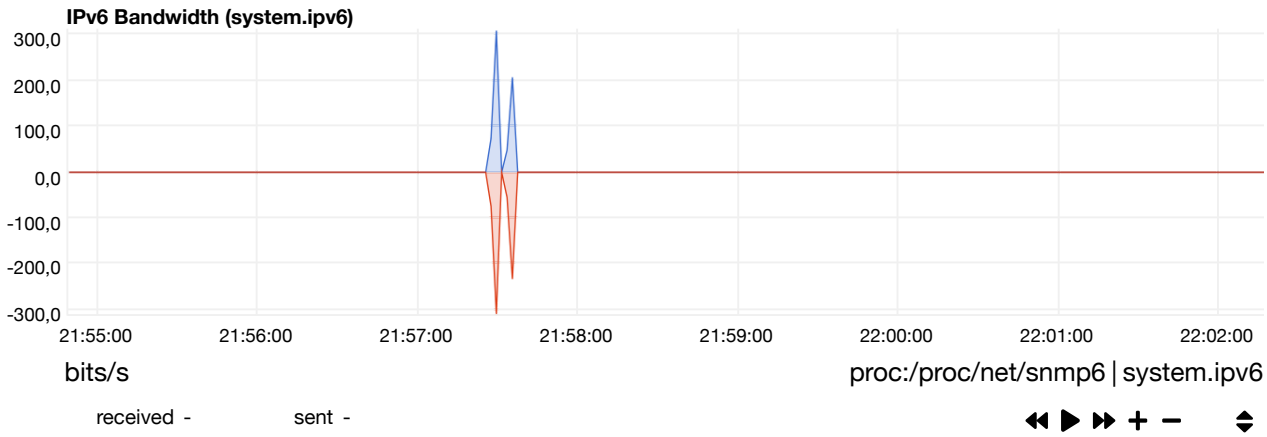
Total bandwidth of all physical network interfaces. This does not include `lo`, VPNs, network bridges, IFB devices, bond interfaces, etc. Only the bandwidth of physical network interfaces is aggregated. Physical are all the network interfaces that are listed in `/proc/net/dev`, but do not exist in `/sys/devices/virtual/net`.



Total IP traffic in the system.

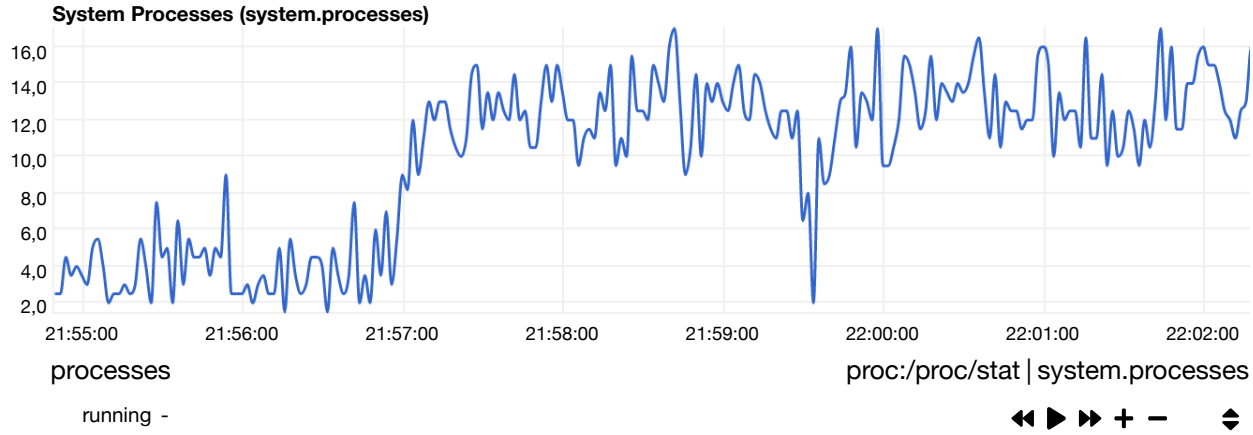


Total IPv6 Traffic.

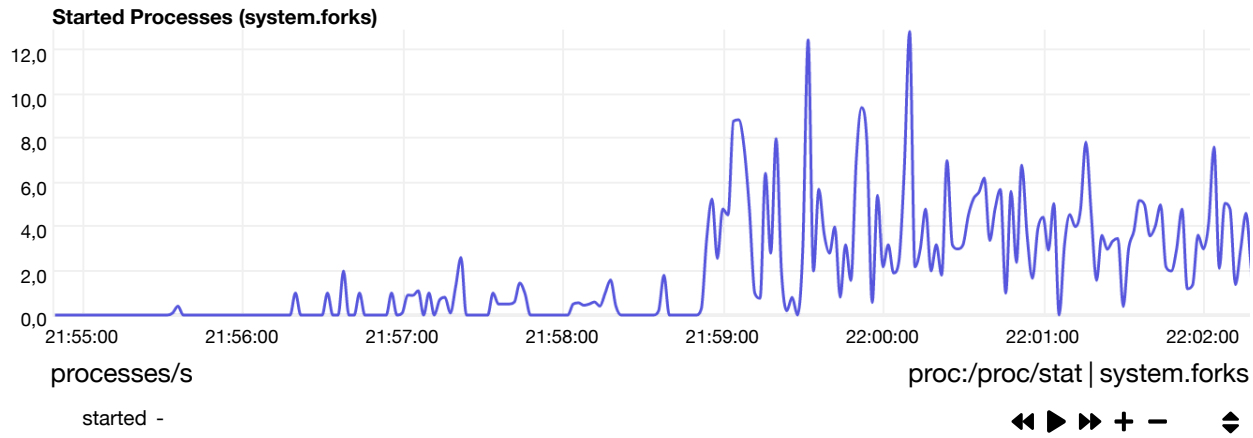


processes

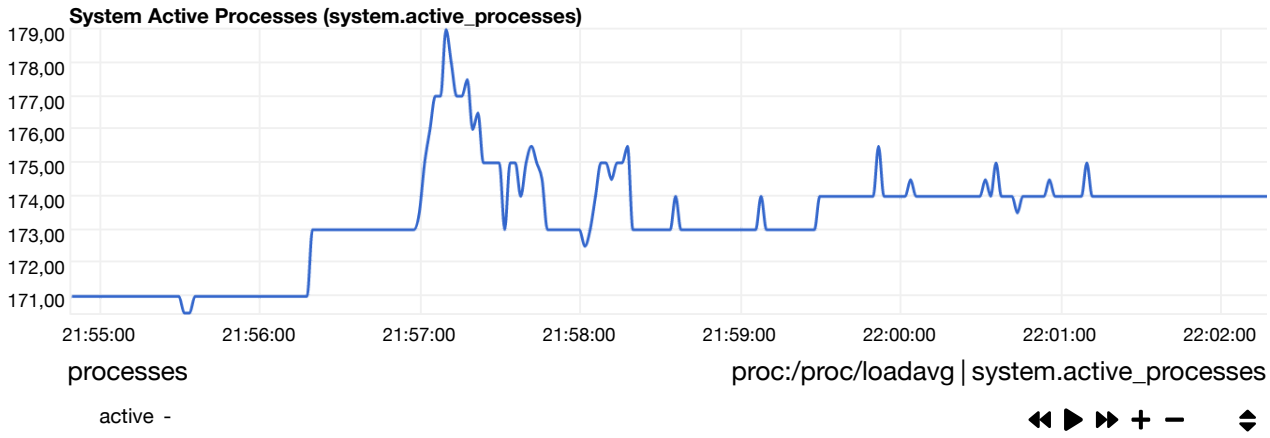
System processes. **Running** are the processes in the CPU. **Blocked** are processes that are willing to enter the CPU, but they cannot, e.g. because they wait for disk activity.



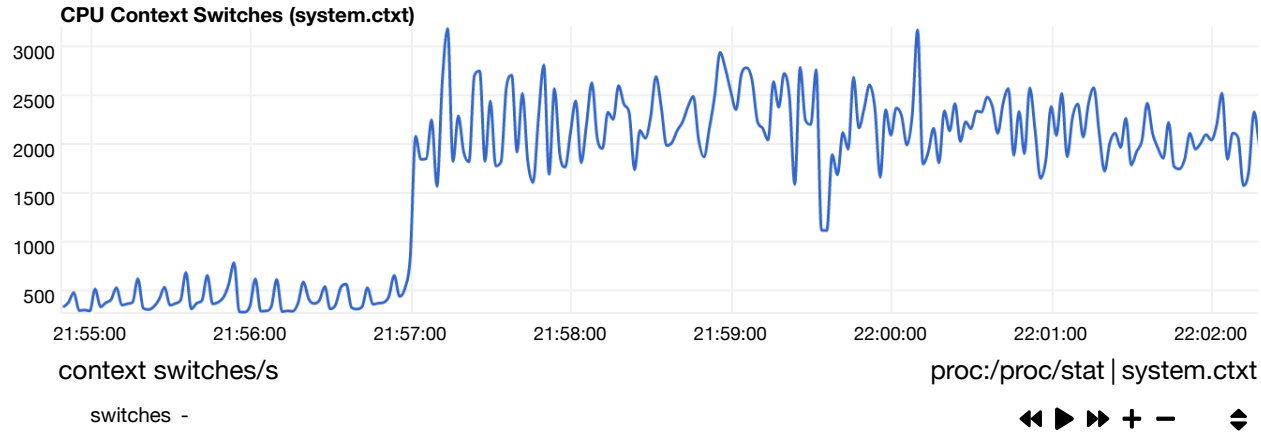
Number of new processes created.



All system processes.

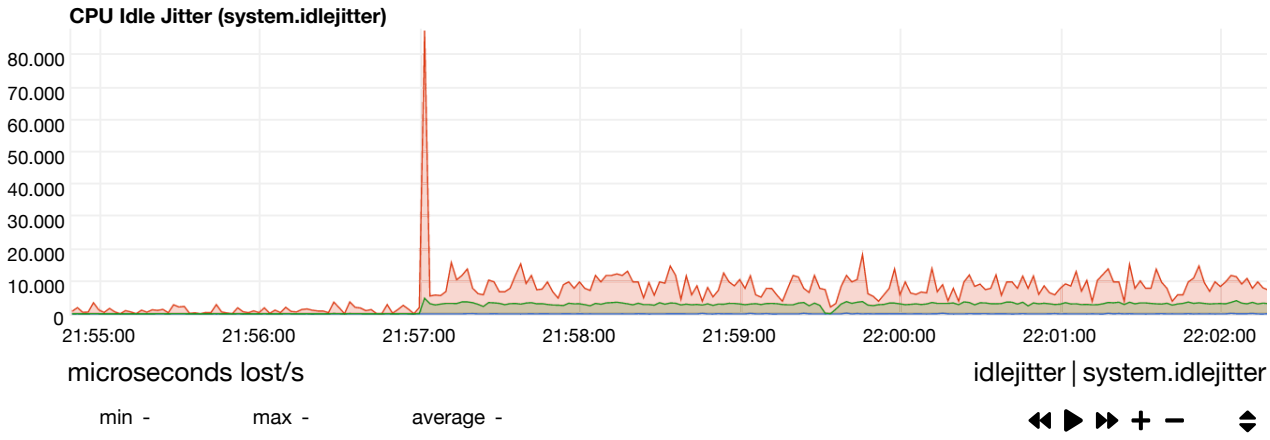


Context Switches (https://en.wikipedia.org/wiki/Context_switch), is the switching of the CPU from one process, task or thread to another. If there are many processes or threads willing to execute and very few CPU cores available to handle them, the system is making more context switching to balance the CPU resources among them. The whole process is computationally intensive. The more the context switches, the slower the system gets.



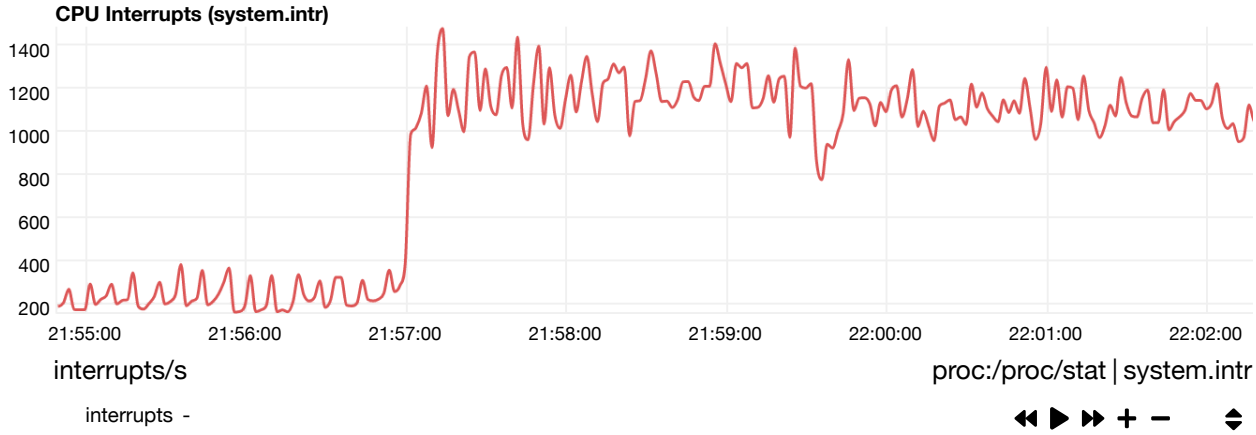
idlejitter

Idle jitter is calculated by netdata. A thread is spawned that requests to sleep for a few microseconds. When the system wakes it up, it measures how many microseconds have passed. The difference between the requested and the actual duration of the sleep, is the **idle jitter**. This number is useful in real-time environments, where CPU jitter can affect the quality of the service (like VoIP media gateways).

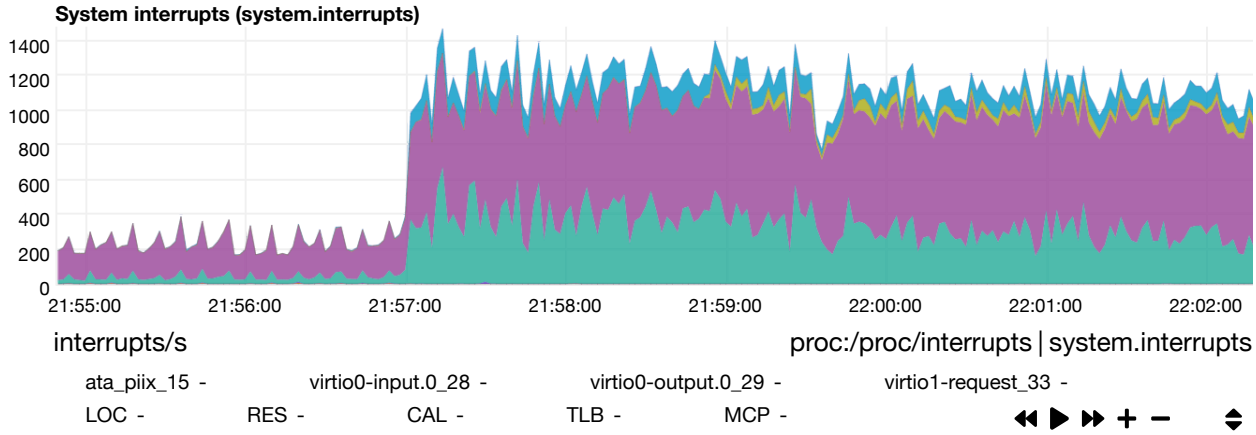


interrupts

Total number of CPU interrupts. Check `system.interrupts` that gives more detail about each interrupt and also the CPUs section where interrupts are analyzed per CPU core.

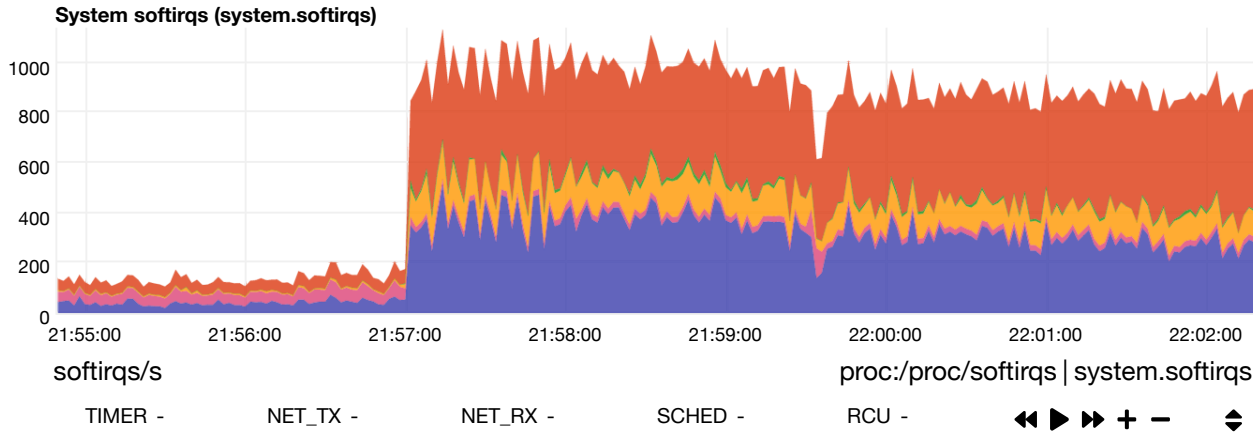


CPU interrupts in detail. At the CPUs section, interrupts are analyzed per CPU core.



softirqs

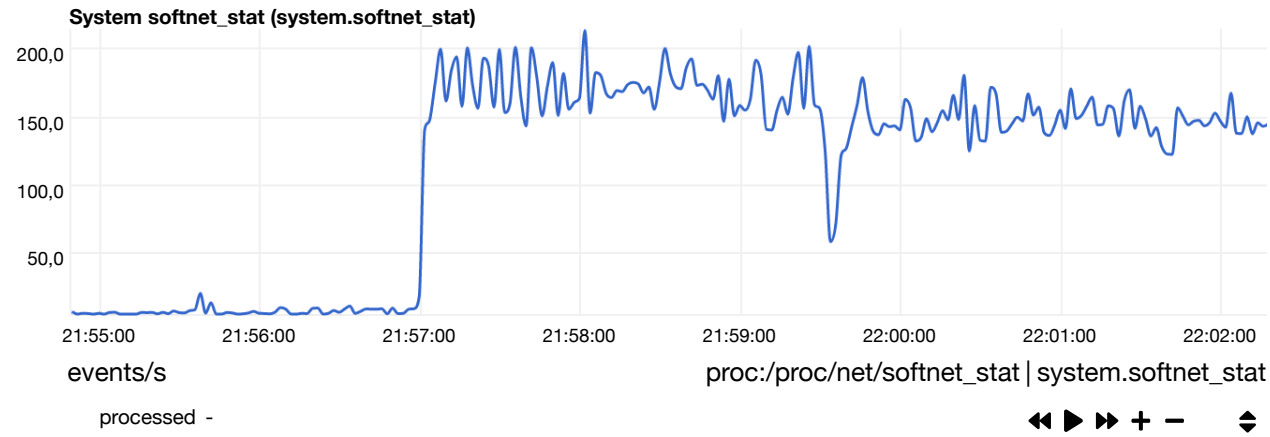
CPU softirqs in detail. At the CPUs section, softirqs are analyzed per CPU core.



softnet

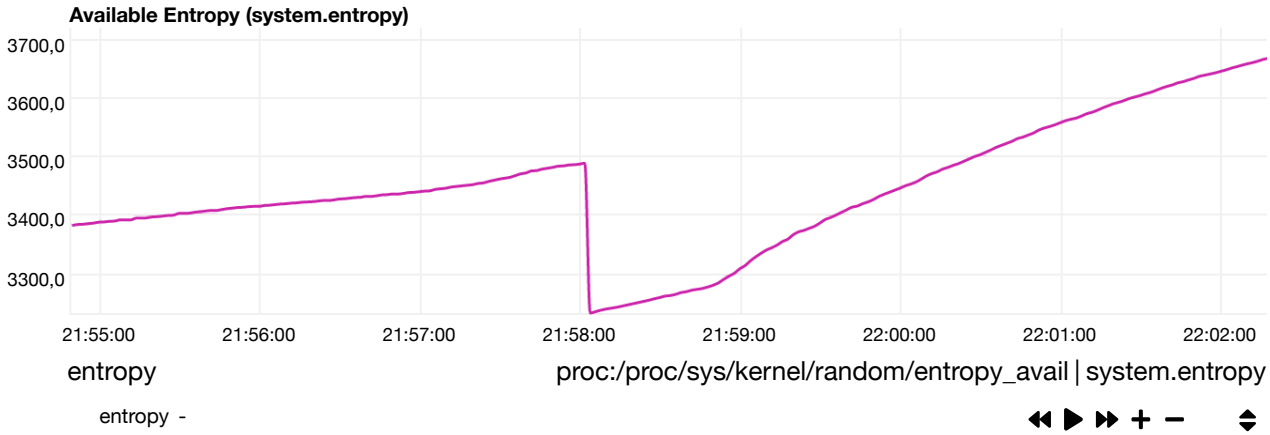
Statistics for CPUs SoftIRQs related to network receive work. Break down per CPU core can be found at CPU / softnet statistics. **processed** states the number of packets processed, **dropped** is the number packets dropped because the network device backlog was full (to fix them on Linux use `sysctl` to increase `net.core.netdev_max_backlog`), **squeezed** is the number of packets dropped

because the network device budget ran out (to fix them on Linux use `sysctl` to increase `net.core.netdev_budget` and/or `net.core.netdev_budget_usecs`). More information about identifying and troubleshooting network driver related issues can be found at Red Hat Enterprise Linux Network Performance Tuning Guide (https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf).



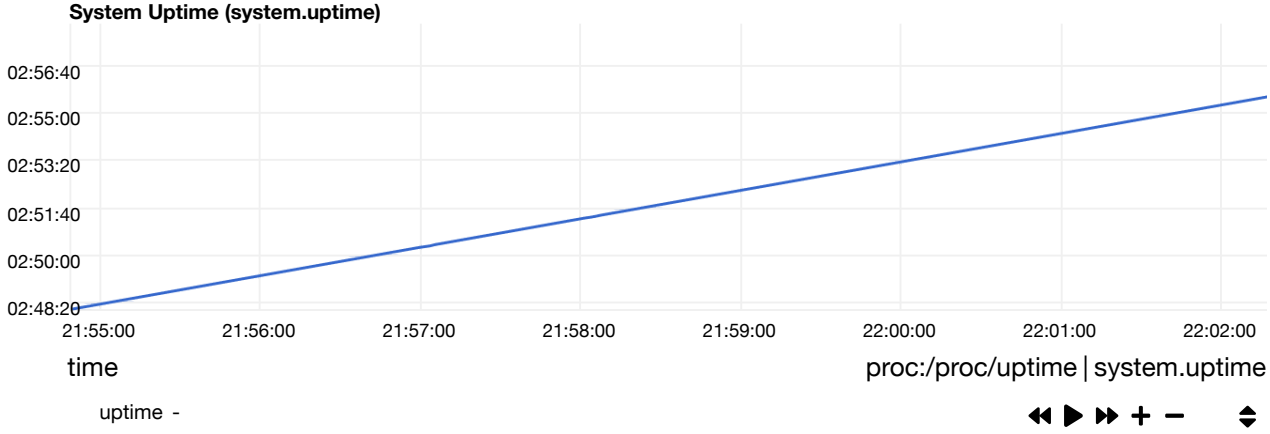
entropy

Entropy (https://en.wikipedia.org/wiki/Entropy_(computing)), is a pool of random numbers (/dev/random (https://en.wikipedia.org/wiki//dev/random)) that is mainly used in cryptography. If the pool of entropy gets empty, processes requiring random numbers may run a lot slower (it depends on the interface each program uses), waiting for the pool to be replenished. Ideally a system with high entropy demands should have a hardware device for that purpose (TPM is one such device). There are also several software-only options you may install, like `haveged`, although these are generally useful only in servers.



uptime

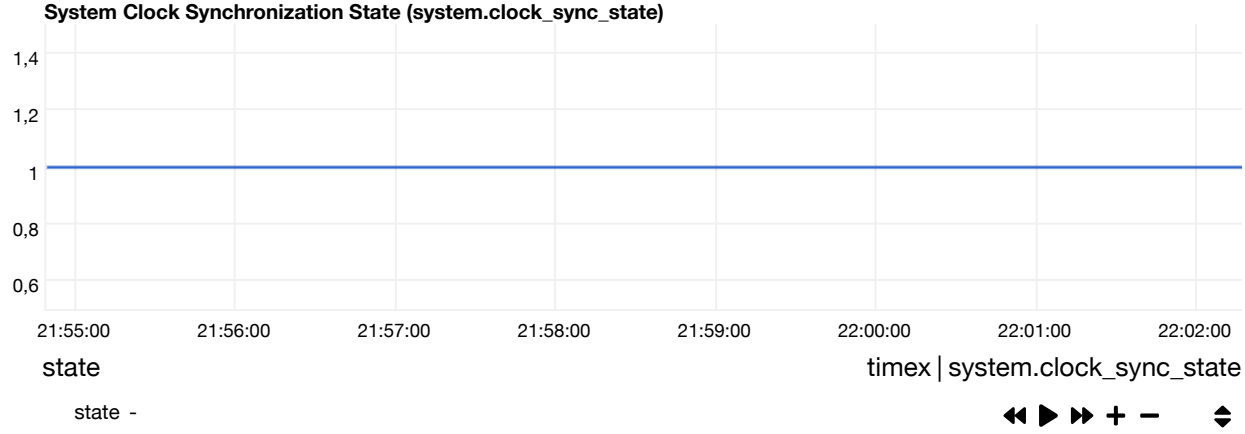
System Uptime (system.uptime)



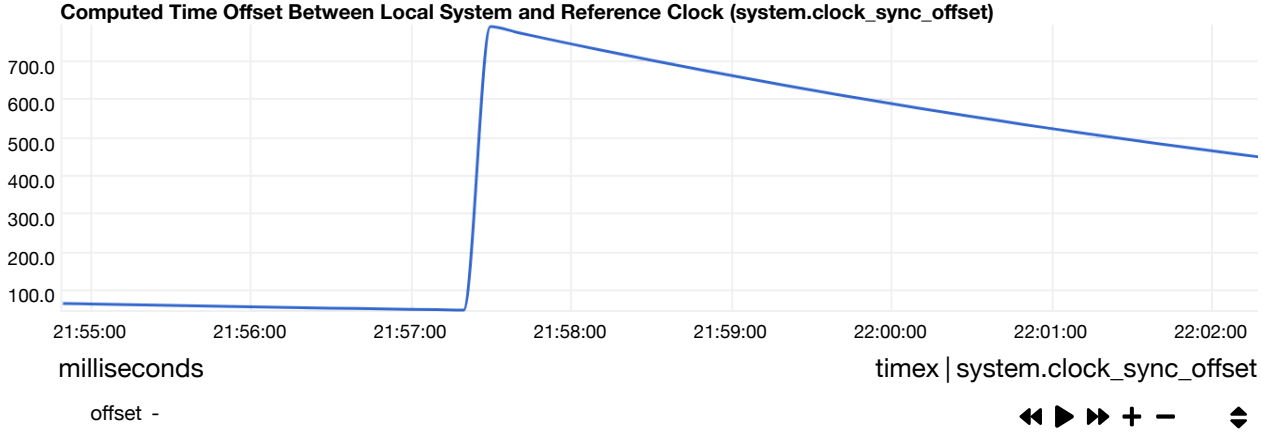
clock synchronization

State map: 0 - not synchronized, 1 - synchronized

System Clock Synchronization State (system.clock_sync_state)

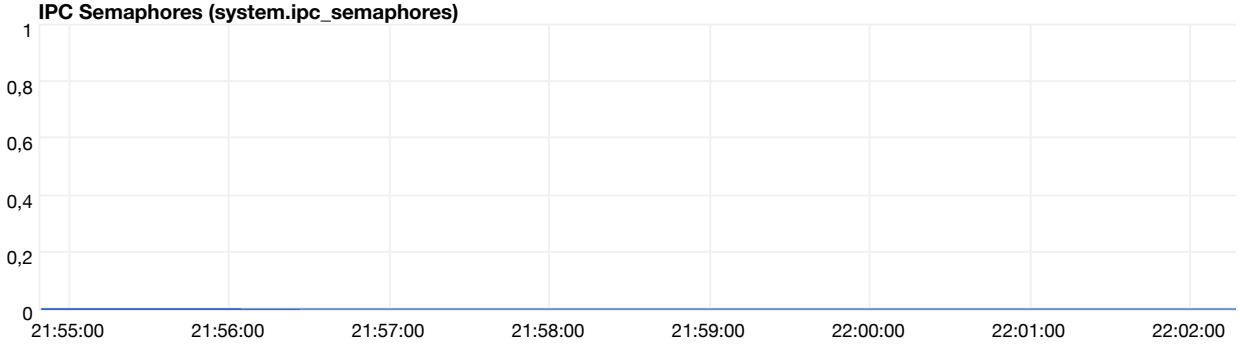


Computed Time Offset Between Local System and Reference Clock (system.clock_sync_offset)



ipc semaphores

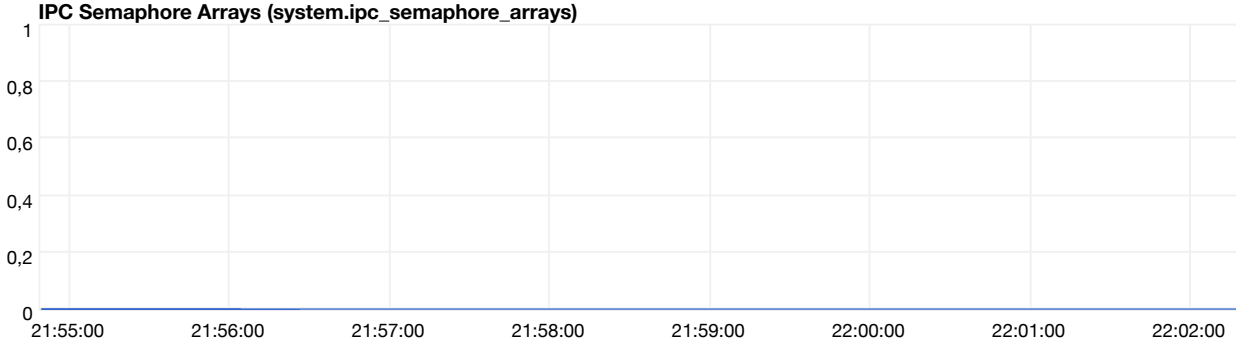
IPC Semaphores (system.ipc_semaphores)



semaphores proc:ipc | system.ipc_semaphores

semaphores -

IPC Semaphore Arrays (system.ipc_semaphore_arrays)

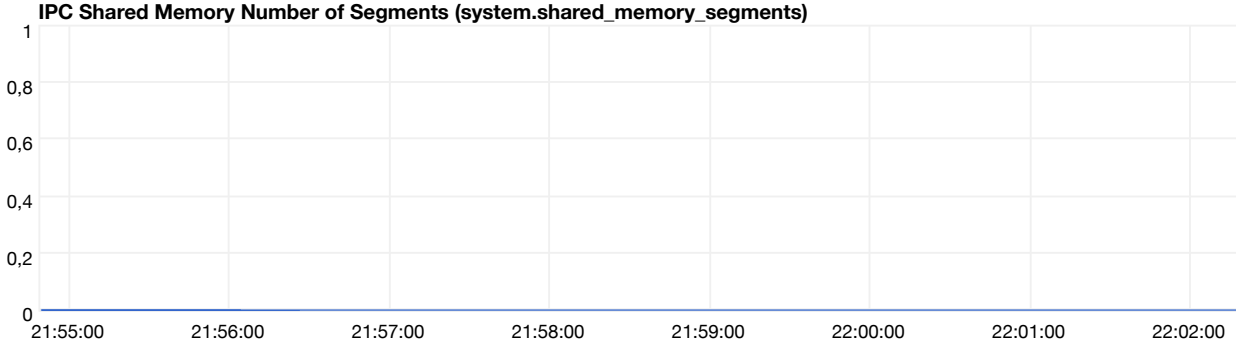


arrays proc:ipc | system.ipc_semaphore_arrays

arrays -

ipc shared memory

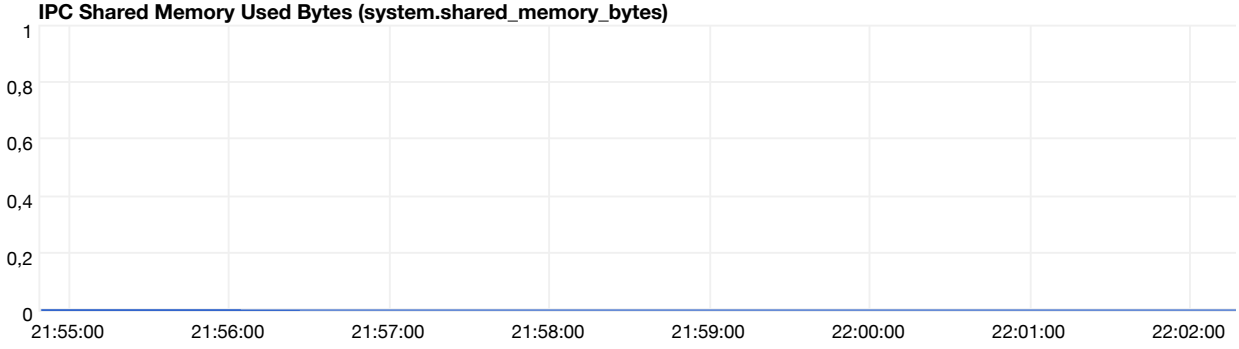
IPC Shared Memory Number of Segments (system.shared_memory_segments)



segments proc:ipc | system.shared_memory_segments

segments -

IPC Shared Memory Used Bytes (system.shared_memory_bytes)



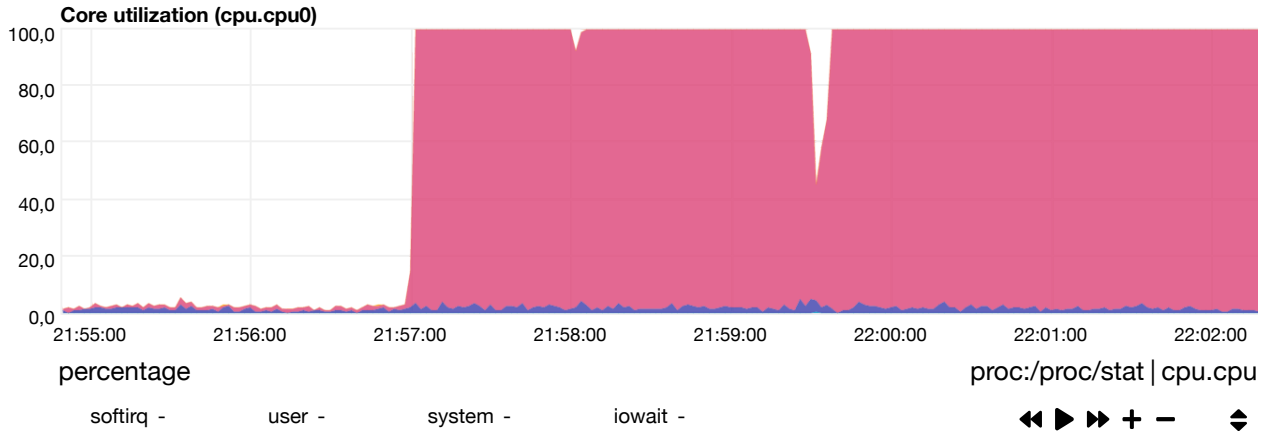
bytes proc:ipc | system.shared_memory_bytes

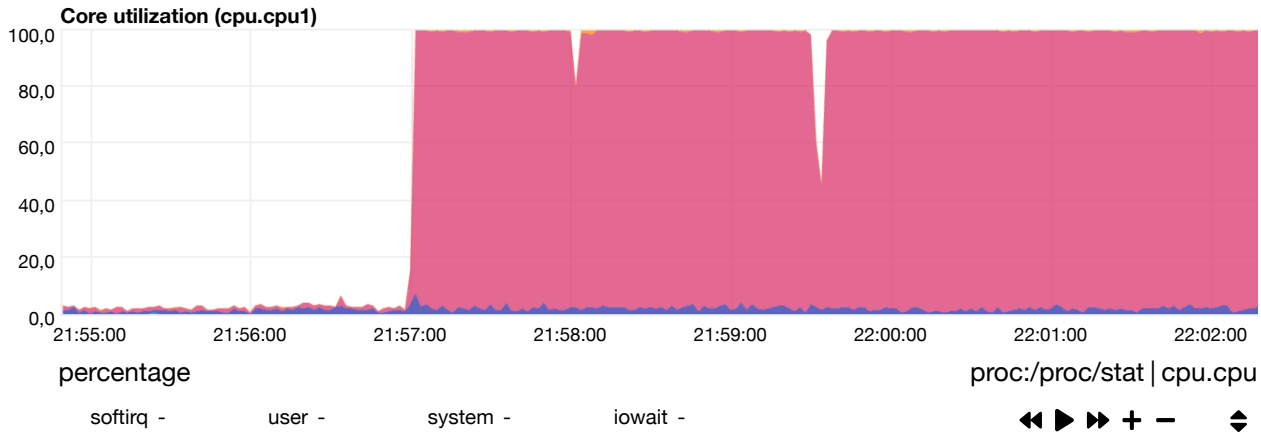
bytes -

CPUs

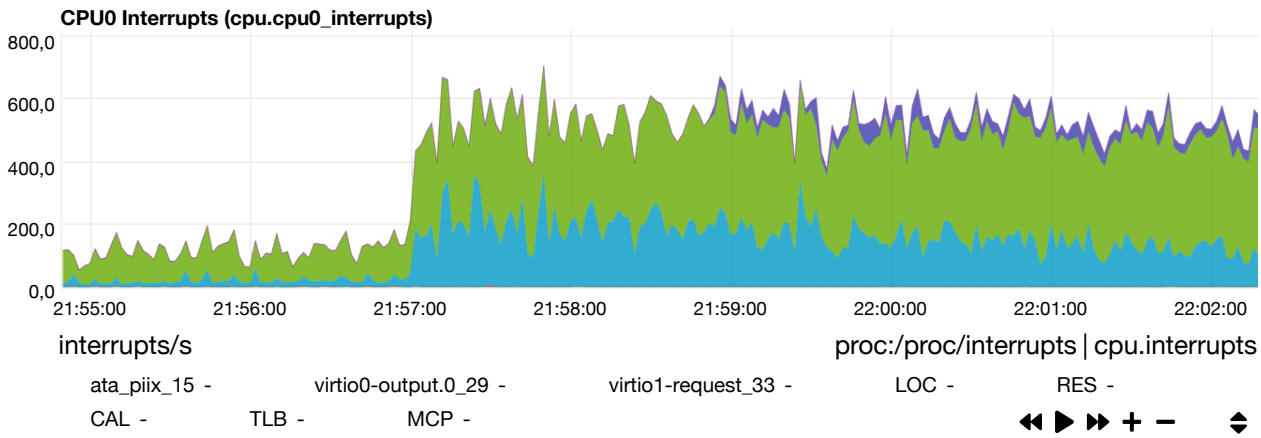
Detailed information for each CPU of the system. A summary of the system for all CPUs can be found at the System Overview section.

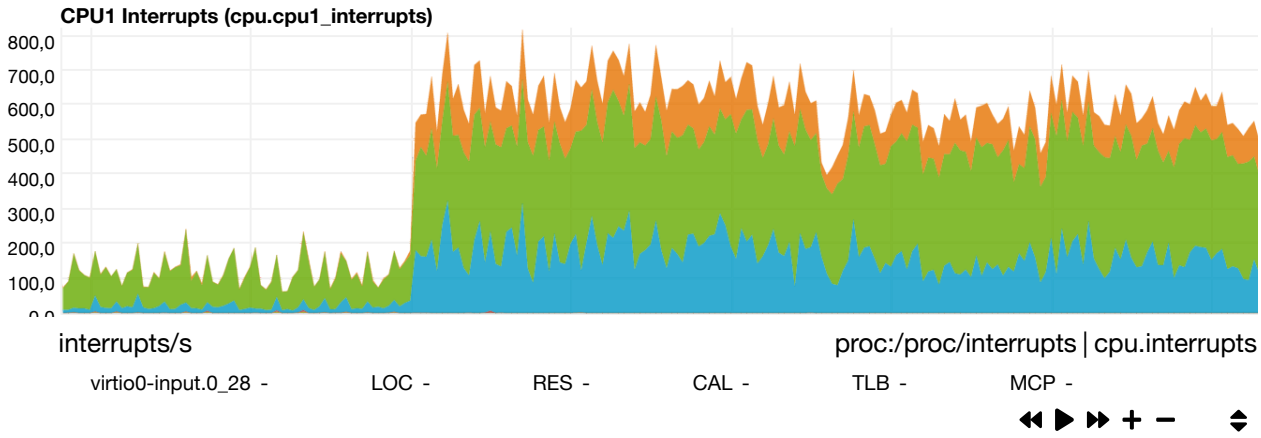
utilization



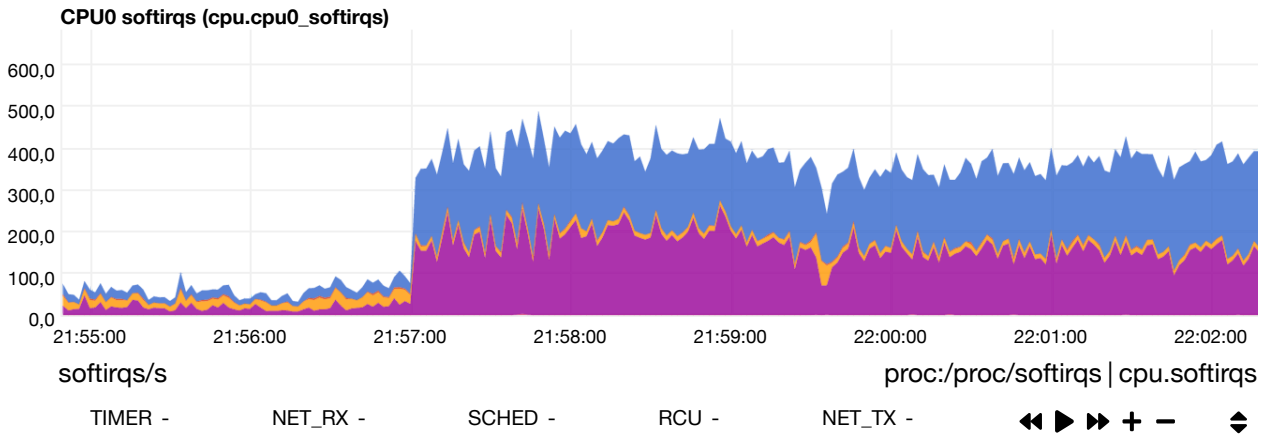


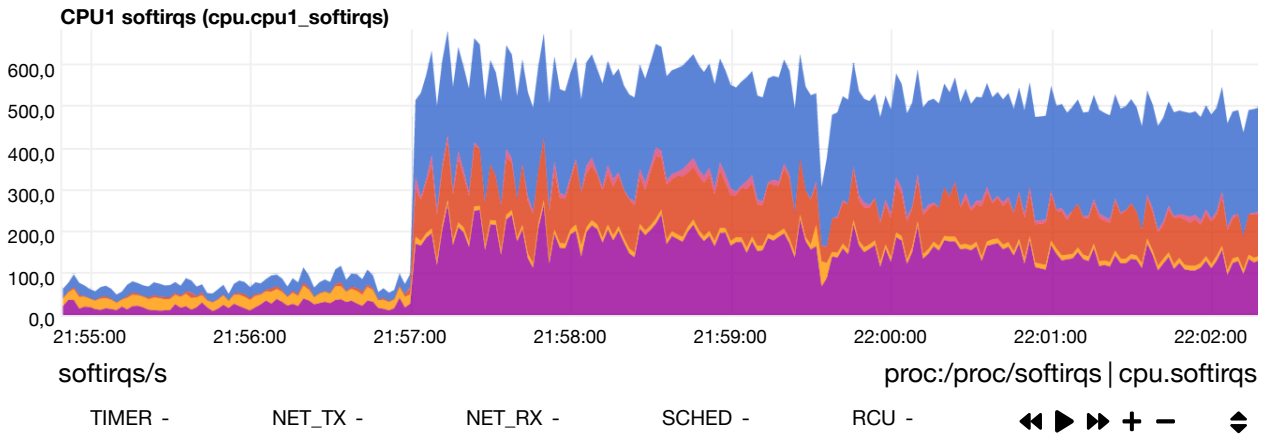
interrupts





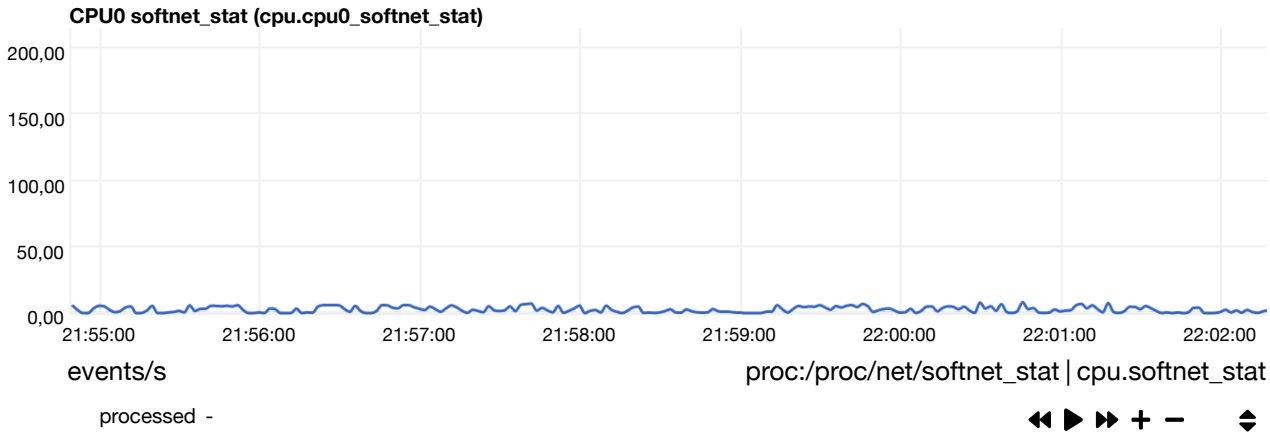
softirqs

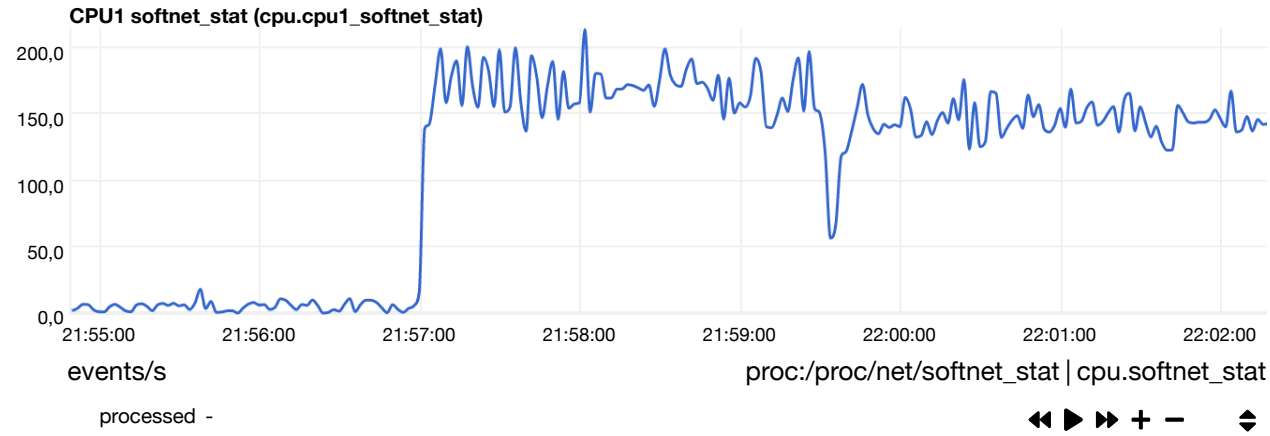




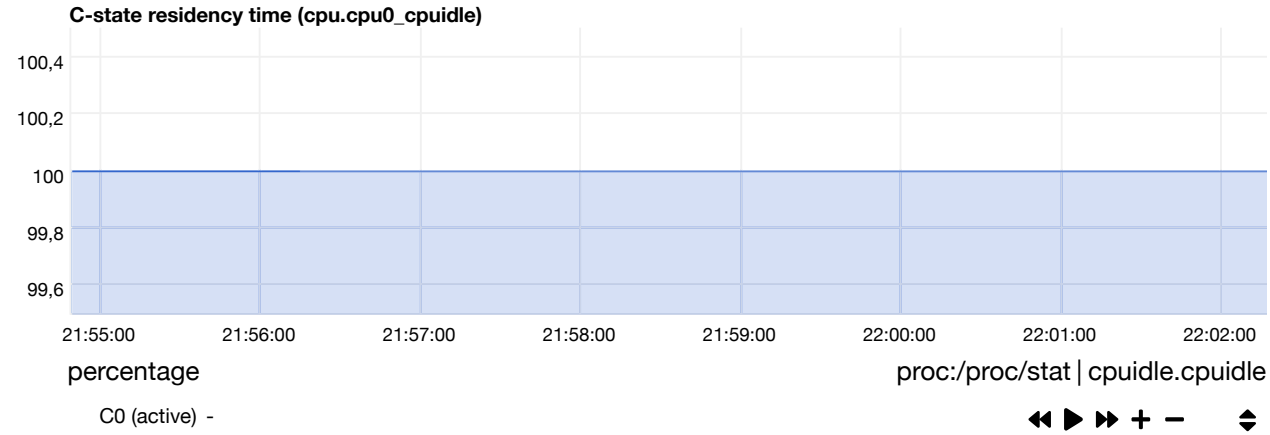
softnet

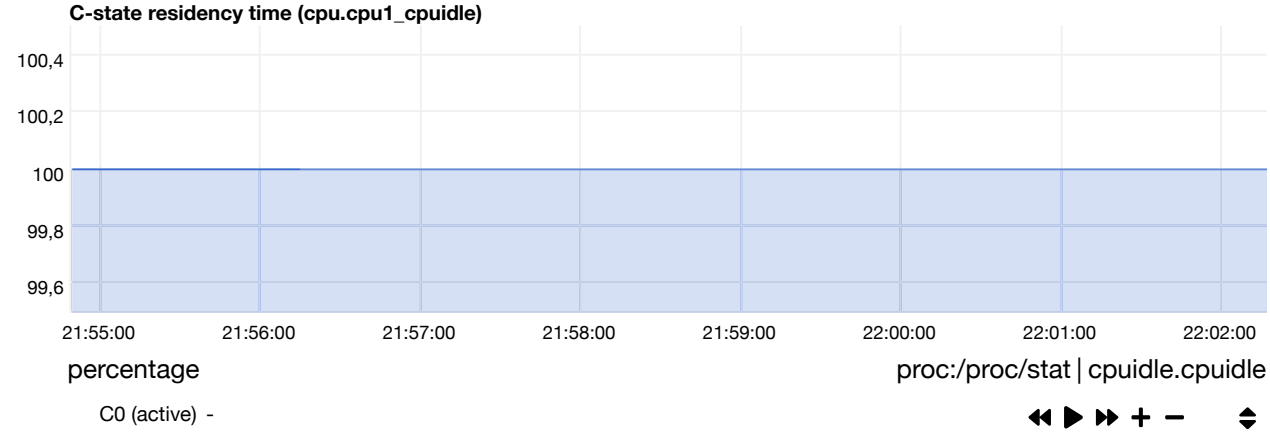
Statistics for per CPUs core SoftIRQs related to network receive work. Total for all CPU cores can be found at System / softnet statistics. **processed** states the number of packets processed, **dropped** is the number packets dropped because the network device backlog was full (to fix them on Linux use `sysctl` to increase `net.core.netdev_max_backlog`), **squeezed** is the number of packets dropped because the network device budget ran out (to fix them on Linux use `sysctl` to increase `net.core.netdev_budget` and/or `net.core.netdev_budget_usecs`). More information about identifying and troubleshooting network driver related issues can be found at Red Hat Enterprise Linux Network Performance Tuning Guide (https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf).


CPU0 softnet_stat (cpu.cpu0_softnet_stat)
200,00
150,00
100,00
50,00
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
events/s
proc:/proc/net/softnet_stat | cpu.softnet_stat
processed -


CPU1 softnet_stat (cpu.cpu1_softnet_stat)
200,0
150,0
100,0
50,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
events/s
proc:/proc/net/softnet_stat | cpu.softnet_stat
processed -

cpuidle


C-state residency time (cpu.cpu0_cpuidle)
100,4
100,2
100
99,8
99,6
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
percentage
proc:/proc/stat | cpuidle.cpuidle
C0 (active) -

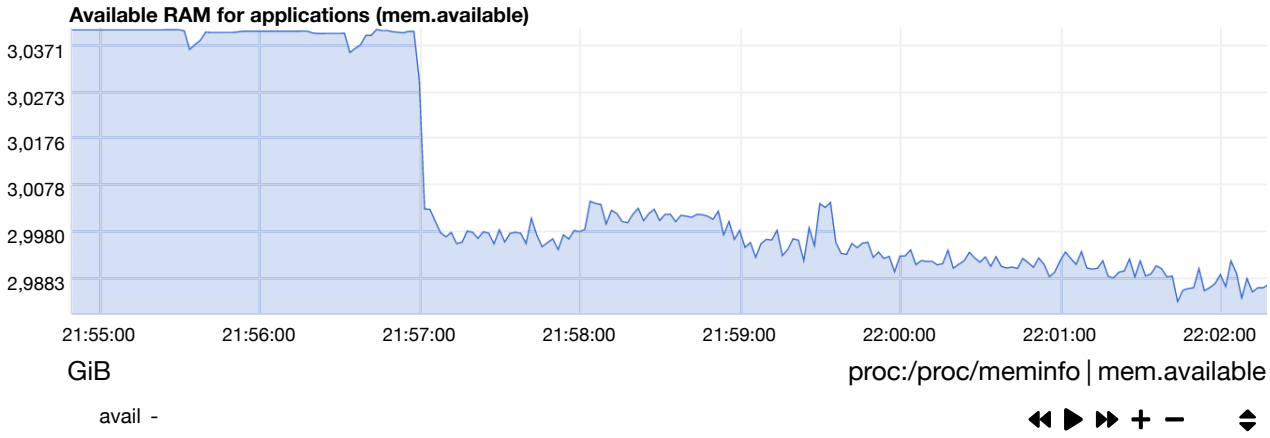

C-state residency time (cpu.cpu1_cpuidle)
100,4
100,2
100
99,8
99,6
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
percentage
proc:/proc/stat | cpuidle.cpuidle
C0 (active) -

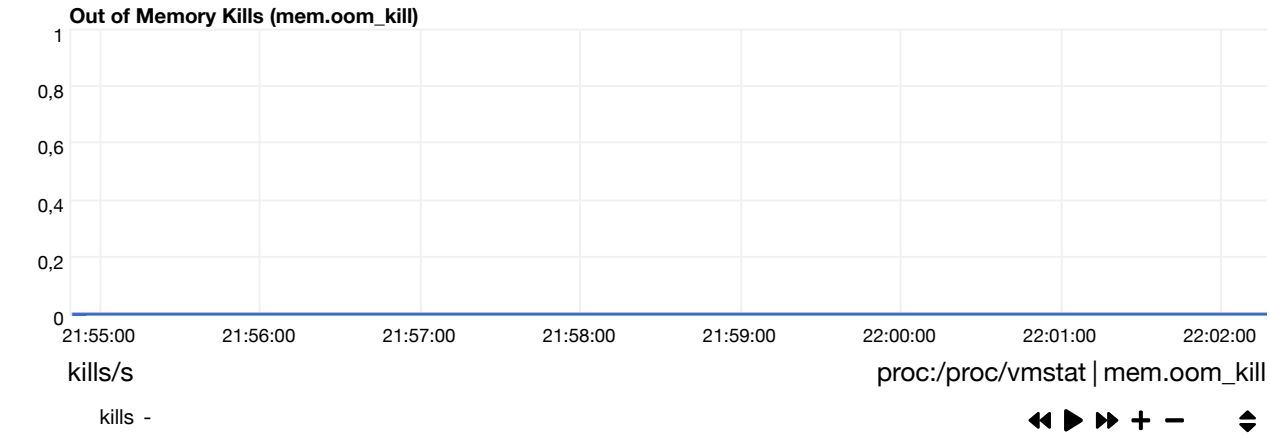
Memory

Detailed information about the memory management of the system.

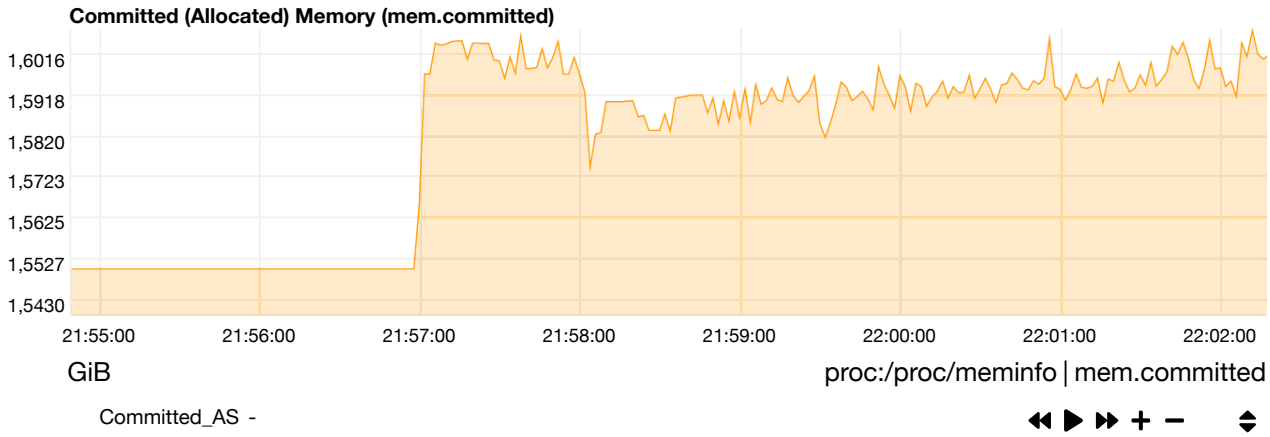
system

Available Memory is estimated by the kernel, as the amount of RAM that can be used by userspace processes, without causing swapping.

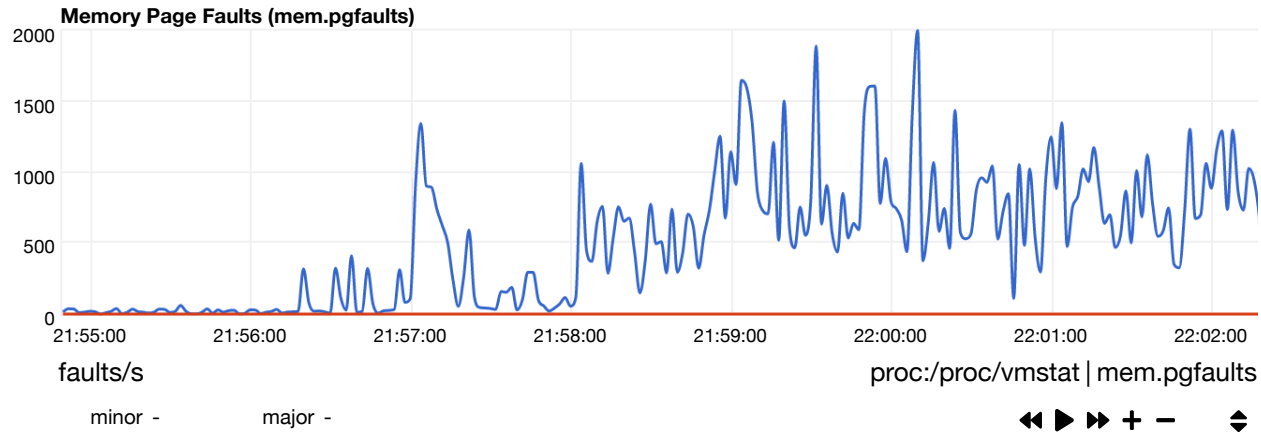




Committed Memory, is the sum of all memory which has been allocated by processes.

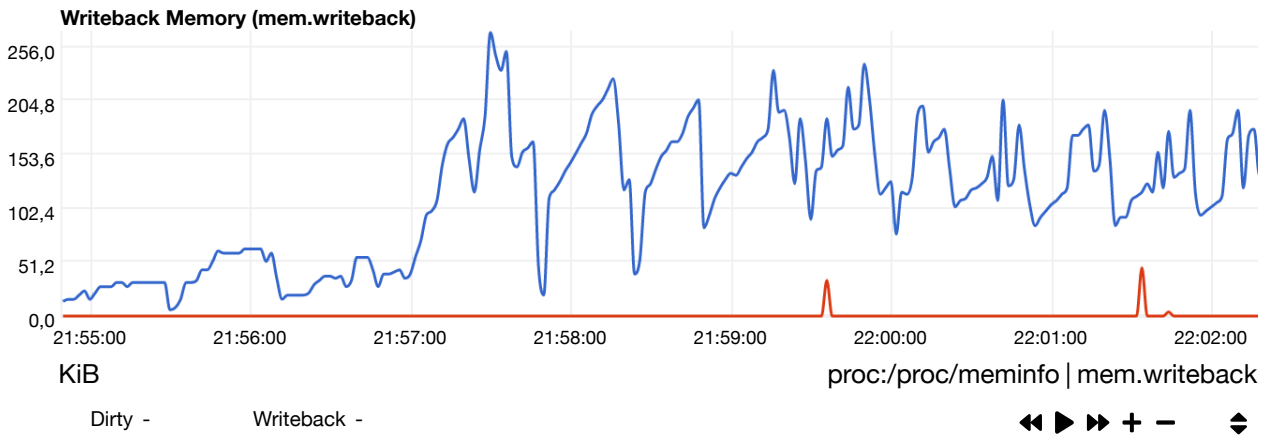


A page fault (https://en.wikipedia.org/wiki/Page_fault) is a type of interrupt, called trap, raised by computer hardware when a running program accesses a memory page that is mapped into the virtual address space, but not actually loaded into main memory. If the page is loaded in memory at the time the fault is generated, but is not marked in the memory management unit as being loaded in memory, then it is called a **minor** or soft page fault. A **major** page fault is generated when the system needs to load the memory page from disk or swap memory.

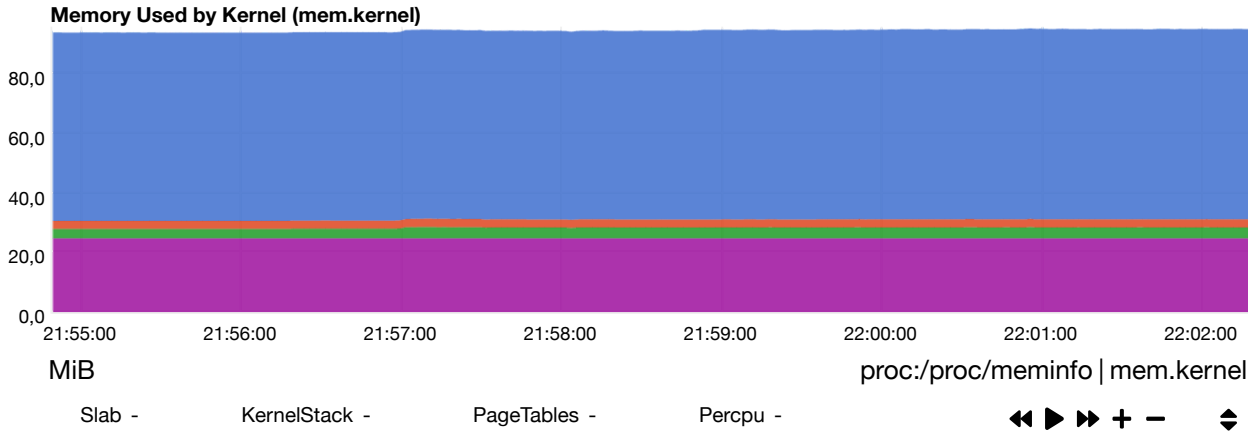


kernel

Dirty is the amount of memory waiting to be written to disk. **Writeback** is how much memory is actively being written to disk.

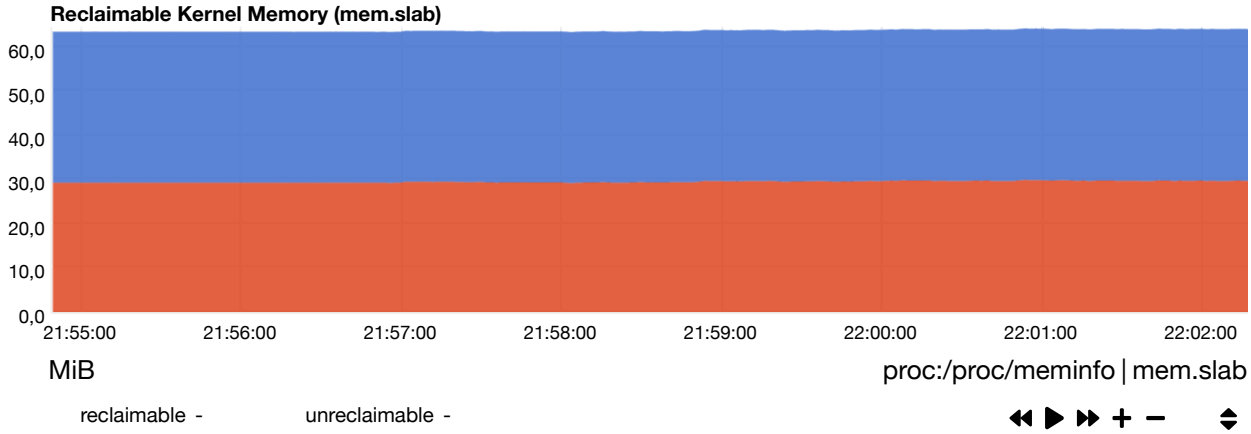


The total amount of memory being used by the kernel. **Slab** is the amount of memory used by the kernel to cache data structures for its own use. **KernelStack** is the amount of memory allocated for each task done by the kernel. **PageTables** is the amount of memory dedicated to the lowest level of page tables (A page table is used to turn a virtual address into a physical memory address). **VmallocUsed** is the amount of memory being used as virtual address space.



slab

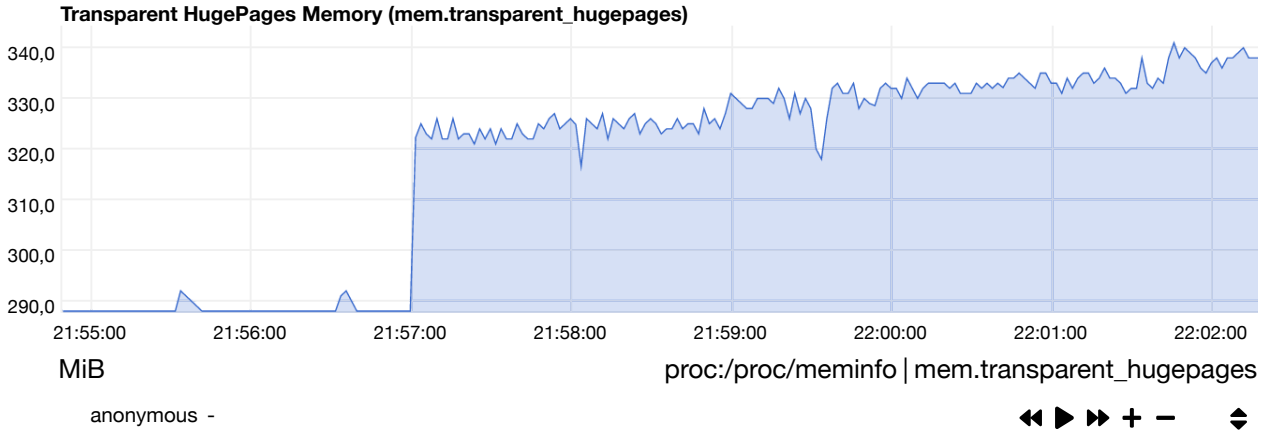
Reclaimable is the amount of memory which the kernel can reuse. **Unreclaimable** can not be reused even when the kernel is lacking memory.



hugepages

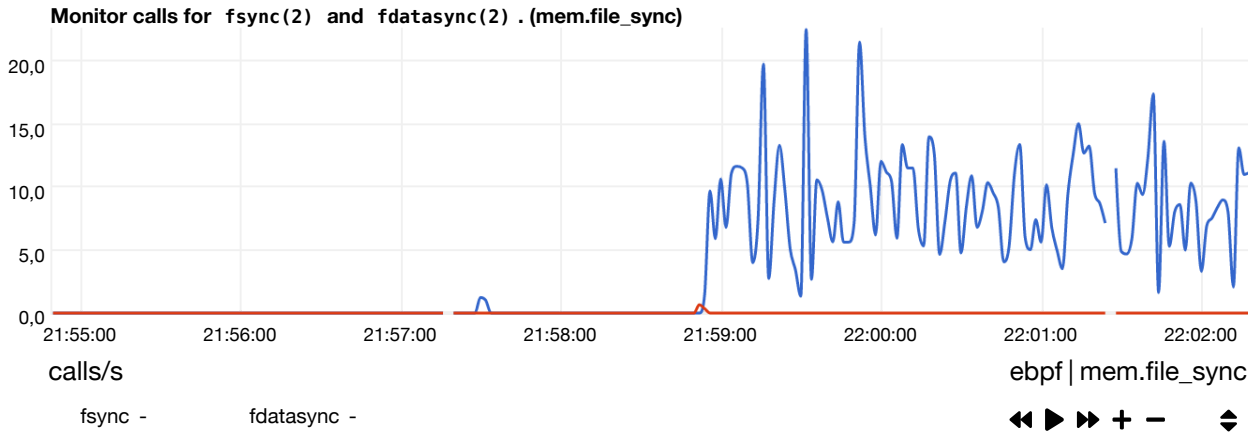
Hugepages is a feature that allows the kernel to utilize the multiple page size capabilities of modern hardware architectures. The kernel creates multiple pages of virtual memory, mapped from both physical RAM and swap. There is a mechanism in the CPU architecture called "Translation Lookaside Buffers" (TLB) to manage the mapping of virtual memory pages to actual physical memory addresses. The TLB is a limited hardware resource, so utilizing a large amount of physical memory with the default page size consumes the TLB and adds processing overhead. By utilizing Huge Pages, the kernel is able to create pages of much larger sizes, each page consuming a single resource in the TLB. Huge Pages are pinned to physical RAM and cannot be swapped/paged out.

Transparent HugePages (THP) is backing virtual memory with huge pages, supporting automatic promotion and demotion of page sizes. It works for all applications for anonymous memory mappings and tmpfs/shmem.

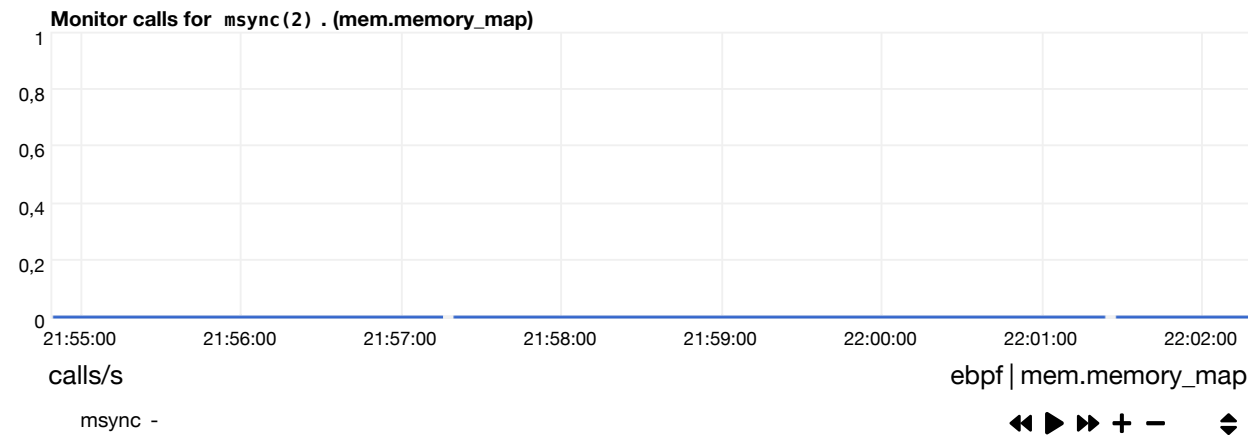


synchronization (eBPF)

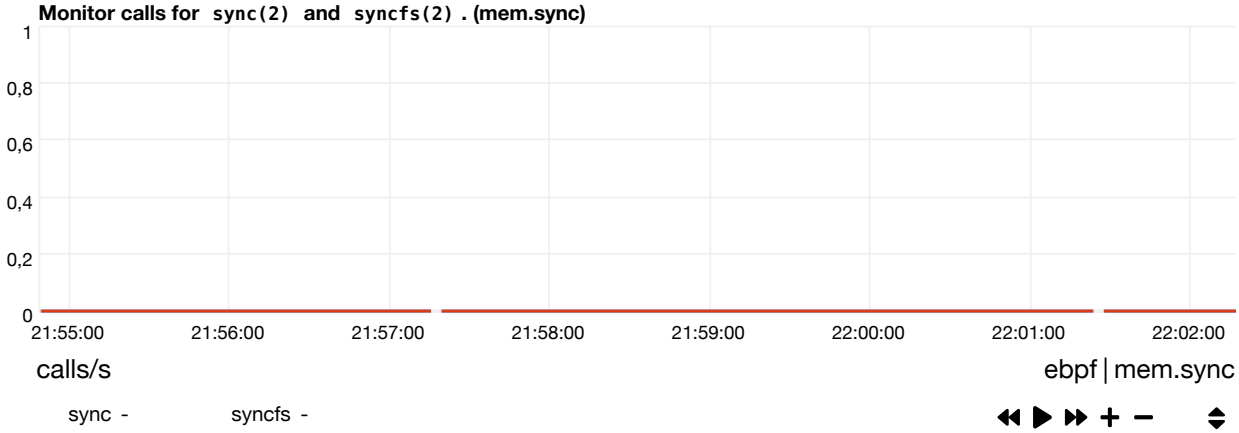
System calls for fsync() and fdatasync() (https://man7.org/linux/man-pages/man2/fsync.2.html) transfer all modified page caches for the files on disk devices. These calls block until the device reports that the transfer has been completed.



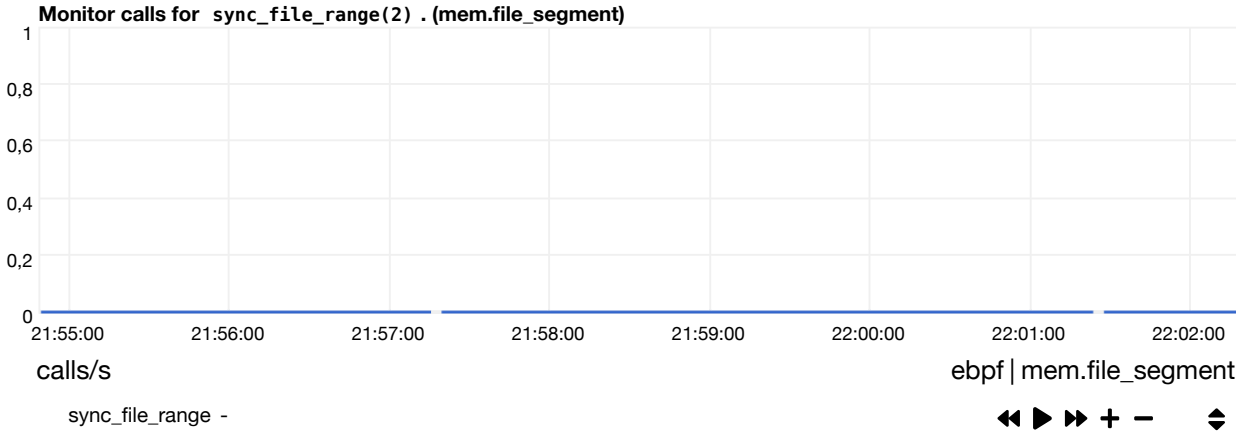
System calls for msync() (https://man7.org/linux/man-pages/man2/msync.2.html) which flushes changes made to the in-core copy of a file that was mapped.



System calls for sync() and syncfs() (https://man7.org/linux/man-pages/man2/sync.2.html) which flush the file system buffers to storage devices. Performance perturbations might be caused by these calls. The `sync()` calls are based on the eBPF syncsnoop (https://github.com/iovisor/bcc/blob/master/tools/syncsnoop.py) from BCC tools.



System calls for sync_file_range() (https://man7.org/linux/man-pages/man2/sync_file_range.2.html) permits fine control when synchronizing the open file referred to by the file descriptor fd with disk. This system call is extremely dangerous and should not be used in portable programs.

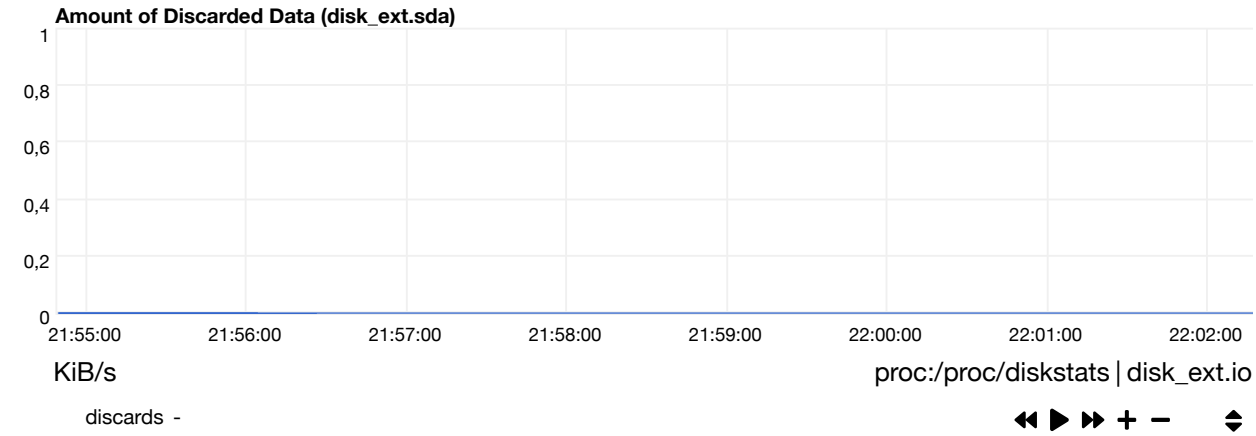


Disks

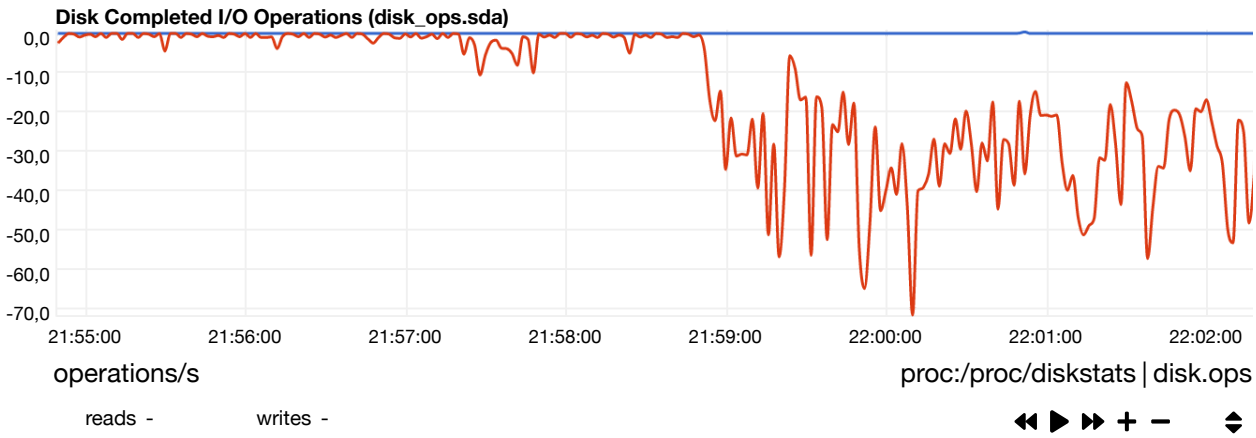
Charts with performance information for all the system disks. Special care has been given to present disk performance metrics in a way compatible with `iostat -x`. netdata by default prevents rendering performance charts for individual partitions and unmounted virtual disks. Disabled charts can still be enabled by configuring the relative settings in the netdata configuration file.

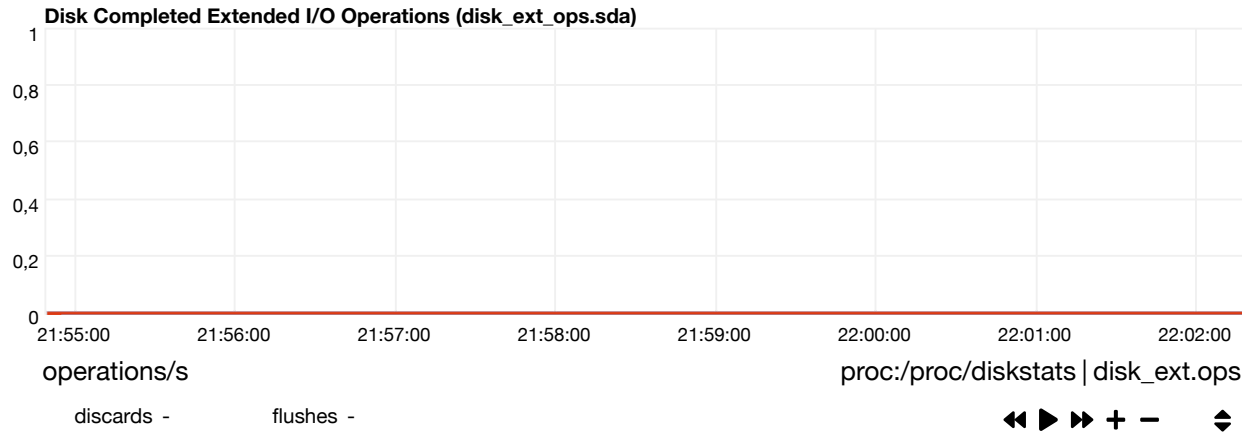
sda

Amount of data transferred to and from disk.

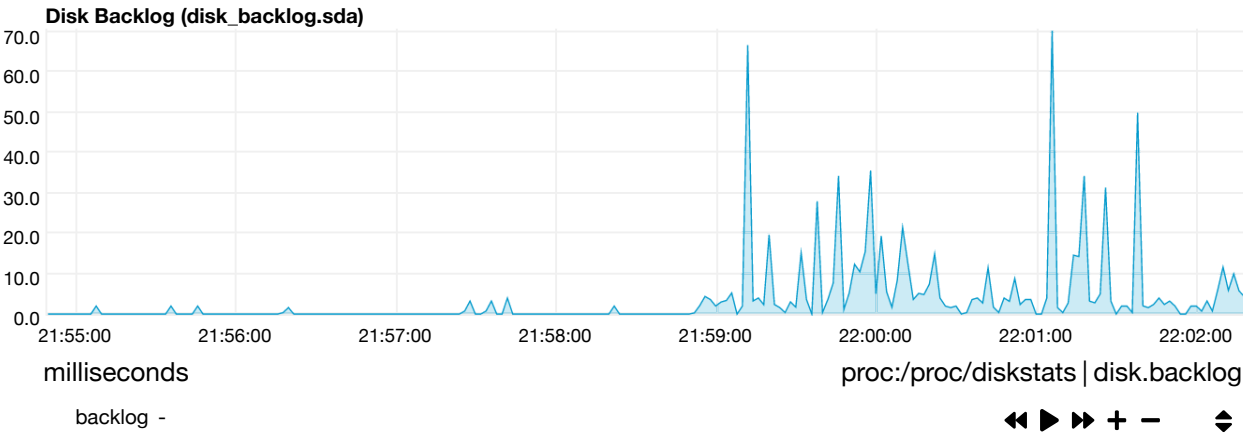


Completed disk I/O operations. Keep in mind the number of operations requested might be higher, since the system is able to merge adjacent to each other (see merged operations chart).

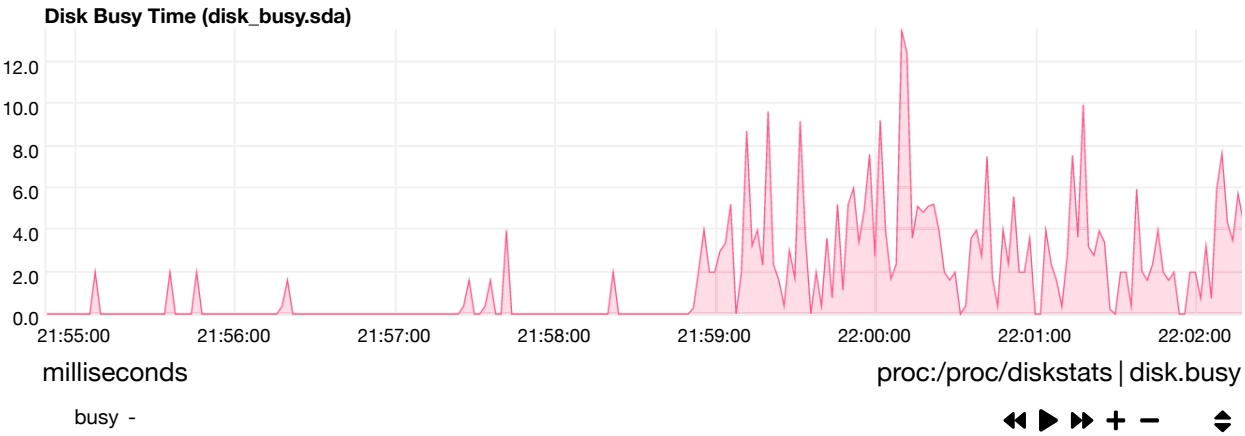




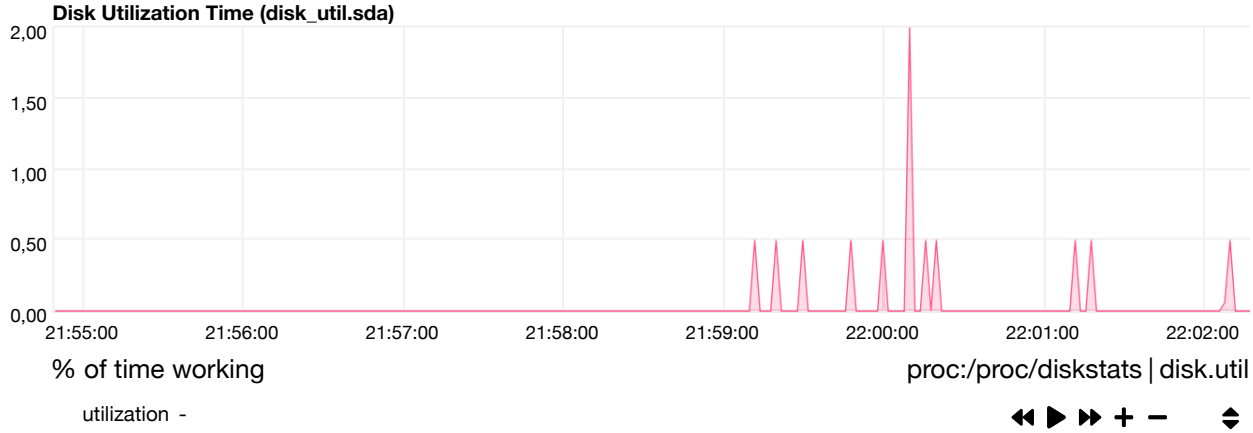
Backlog is an indication of the duration of pending disk operations. On every I/O event the system is multiplying the time spent doing I/O since the last update of this field with the number of pending operations. While not accurate, this metric can provide an indication of the expected completion time of the operations in progress.



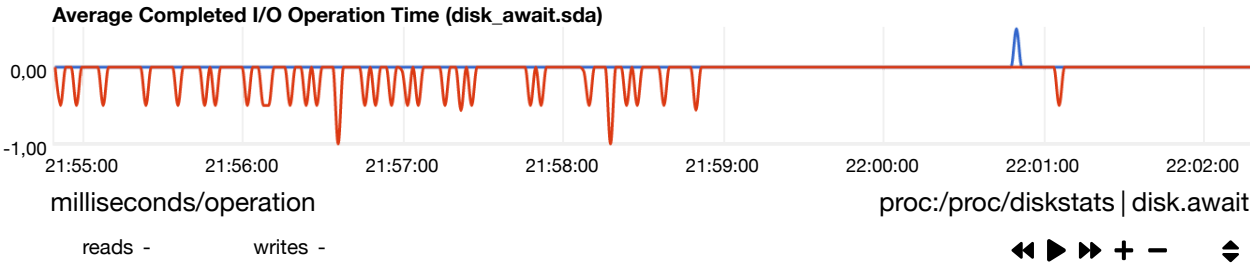
Disk Busy Time measures the amount of time the disk was busy with something.



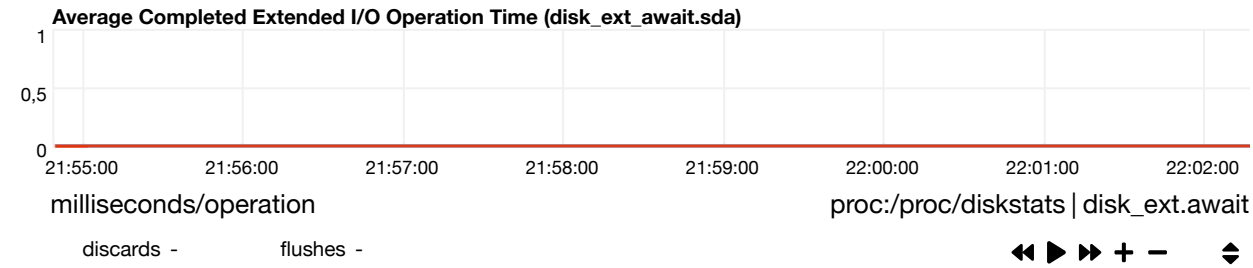
Disk Utilization measures the amount of time the disk was busy with something. This is not related to its performance. 100% means that the system always had an outstanding operation on the disk. Keep in mind that depending on the underlying technology of the disk, 100% here may or may not be an indication of congestion.



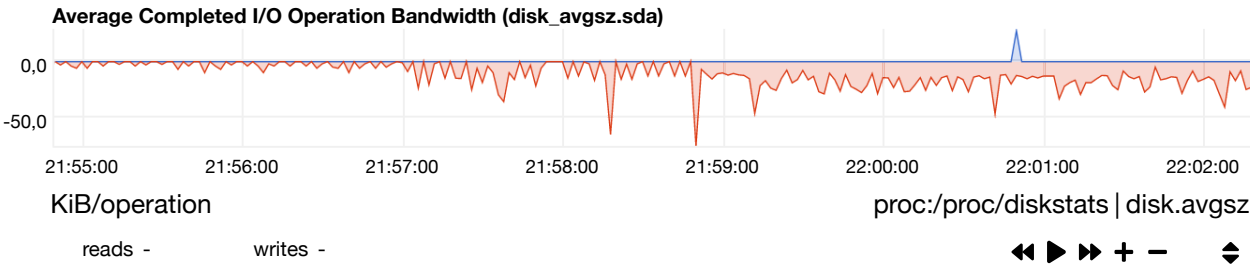
The average time for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.

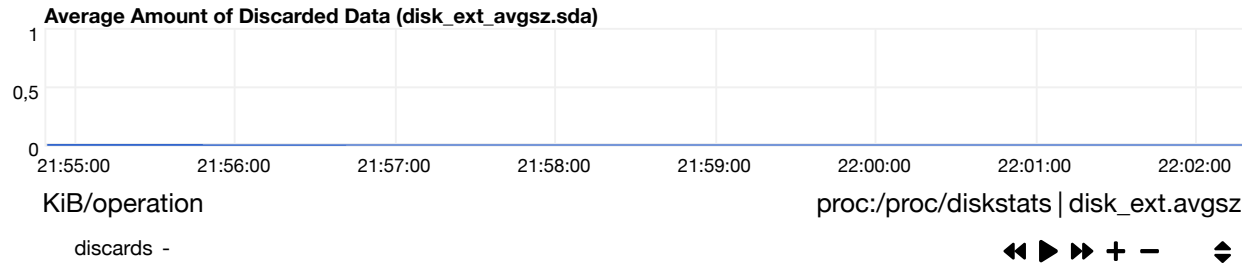


The average time for extended I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.



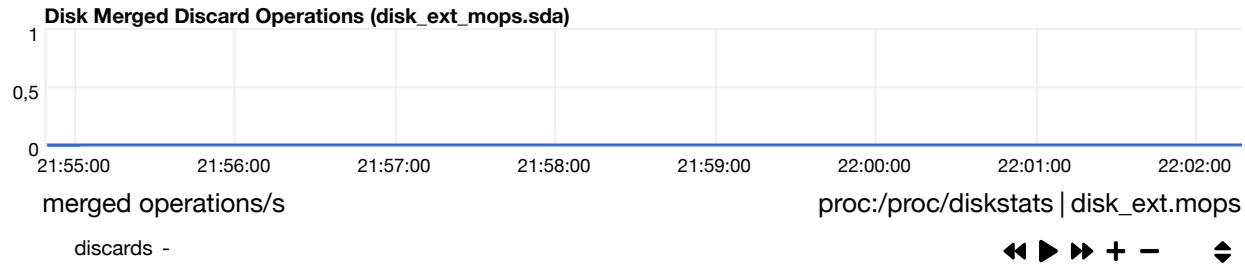
The average I/O operation size.



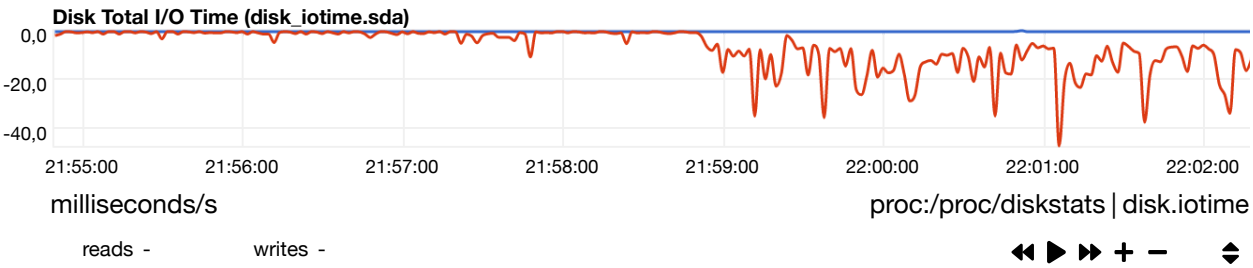


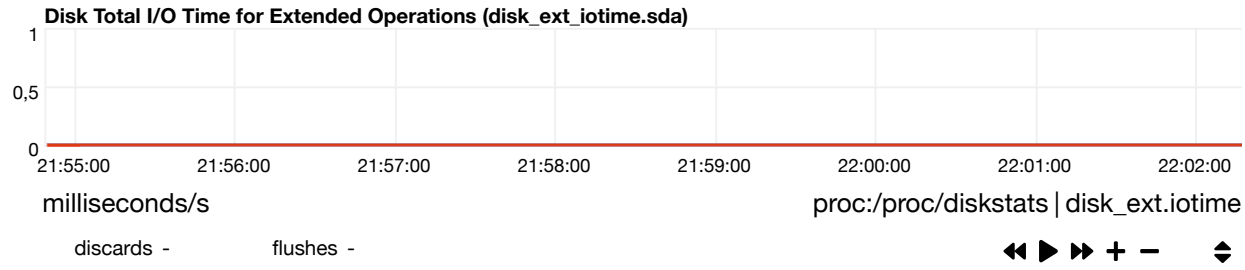
The average service time for completed I/O operations. This metric is calculated using the total busy time of the disk and the number of completed operations. If the disk is able to execute multiple parallel operations the reporting average service time will be misleading.

The number of merged disk operations. The system is able to merge adjacent I/O operations, for example two 4KB reads can become one 8KB read before given to disk.



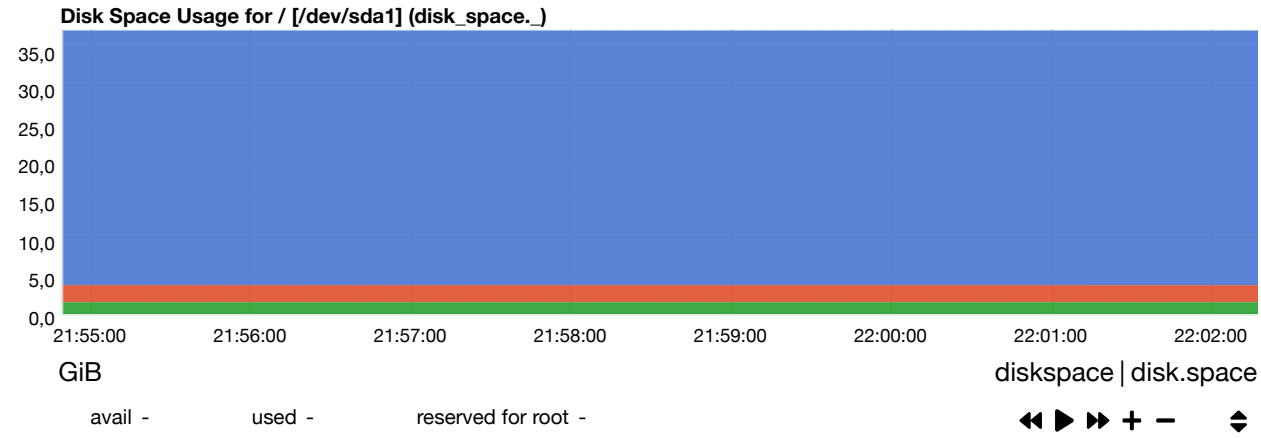
The sum of the duration of all completed I/O operations. This number can exceed the interval if the disk is able to execute I/O operations in parallel.



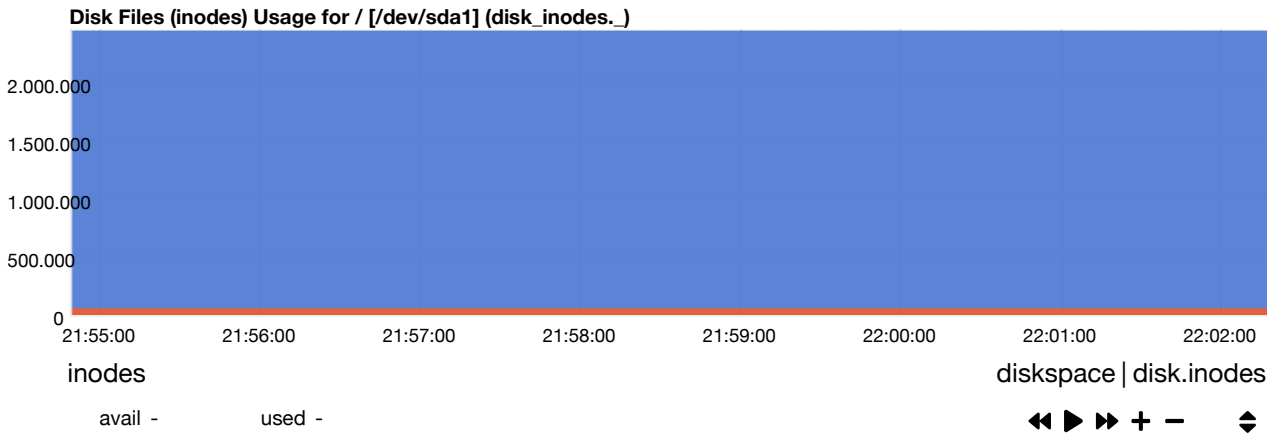


/

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

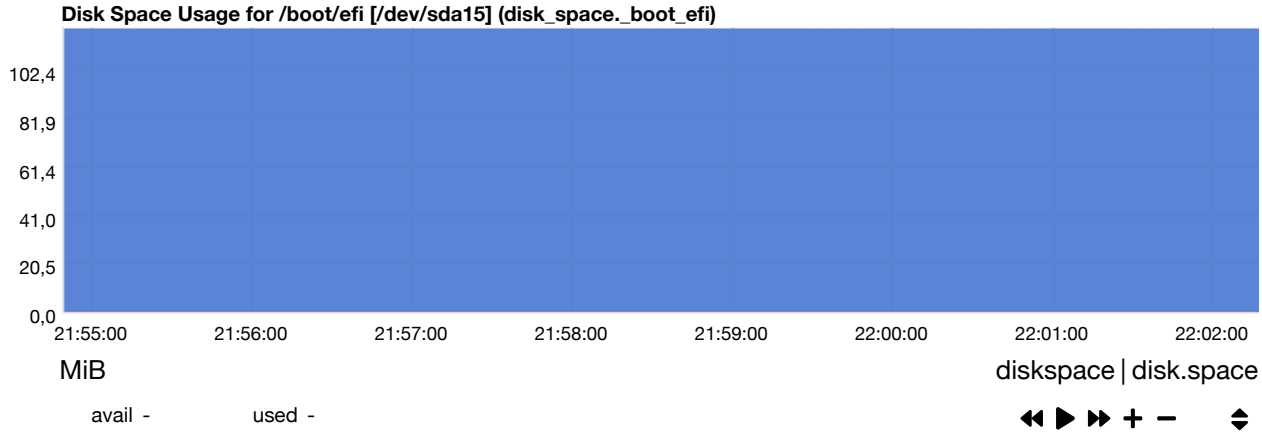


inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.



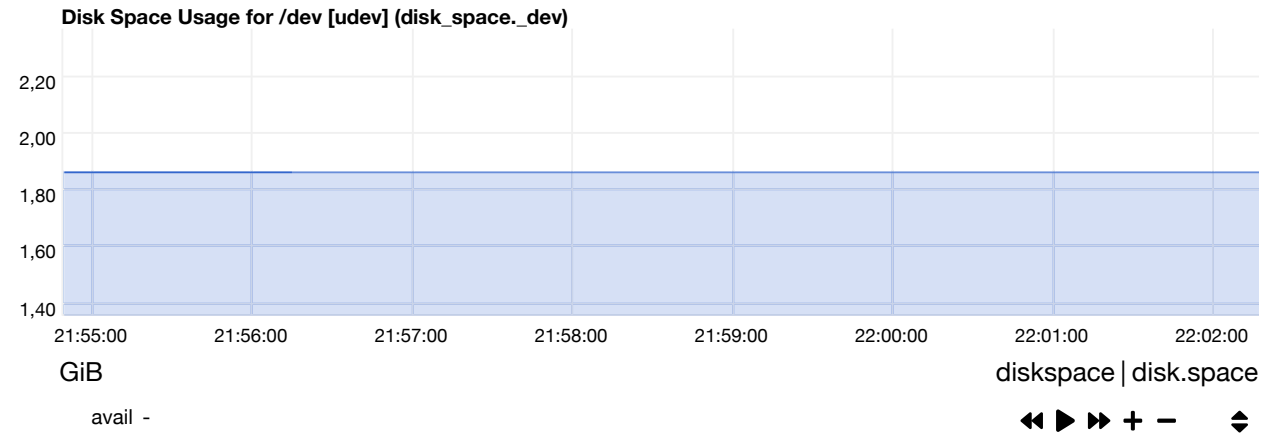
/boot/efi

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

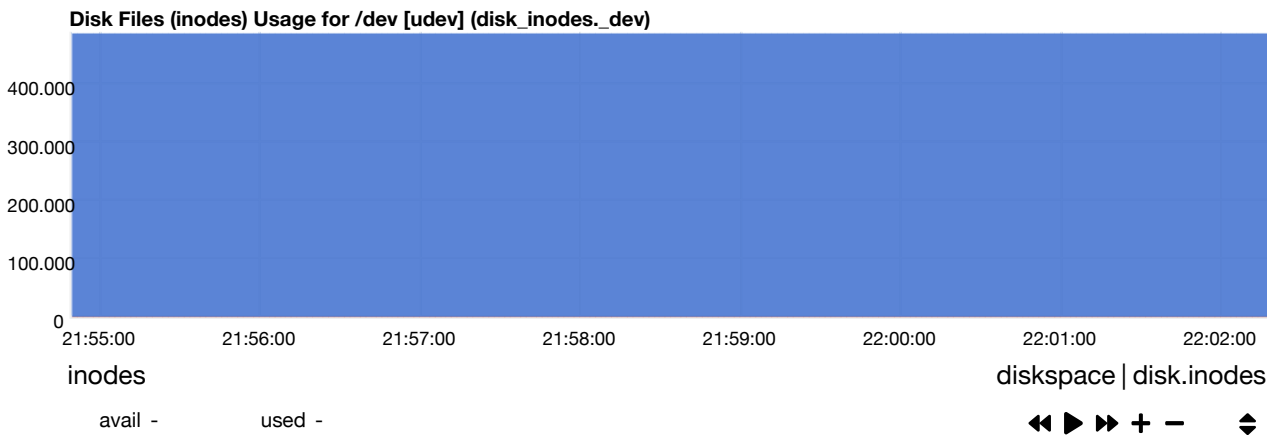


/dev

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

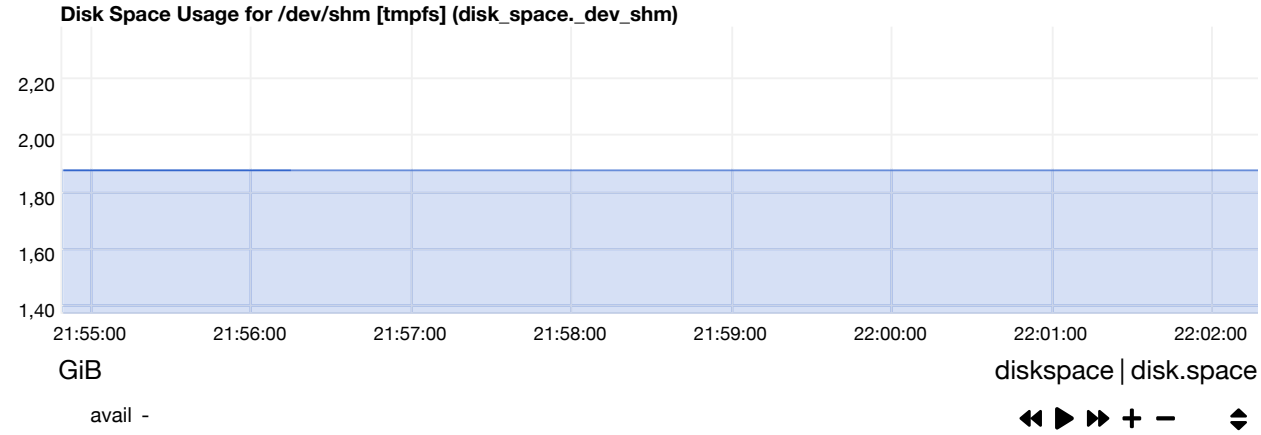


inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.

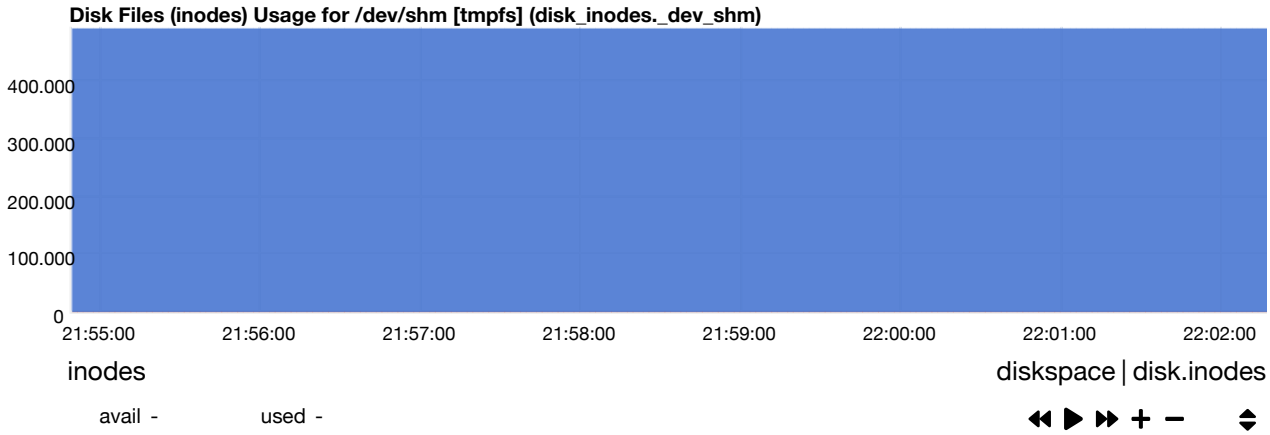


/dev/shm

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

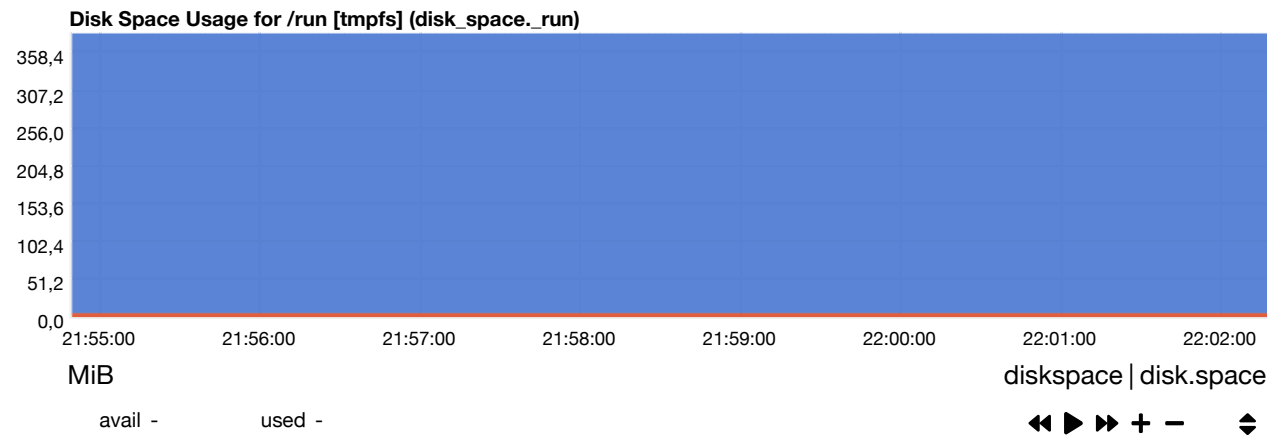


inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.

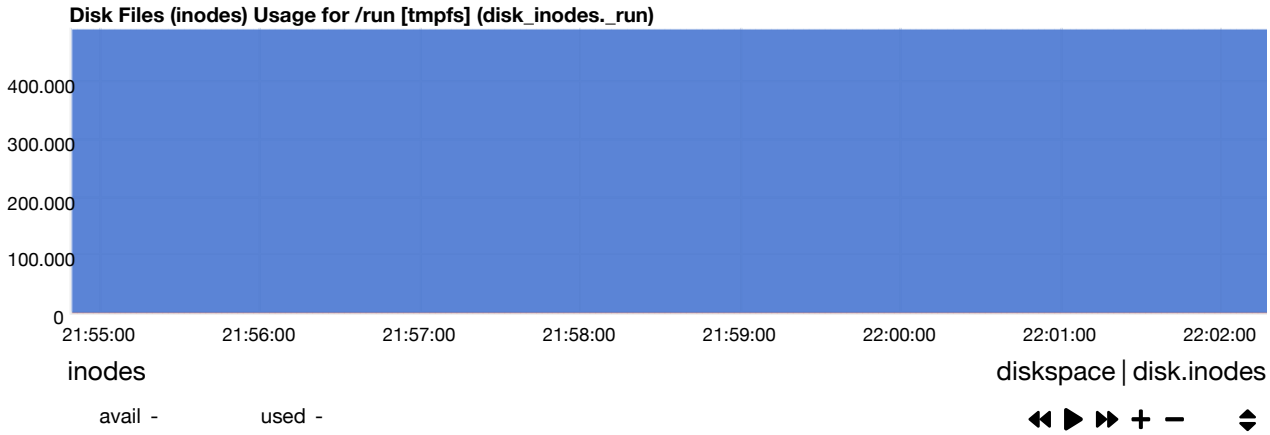


/run

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

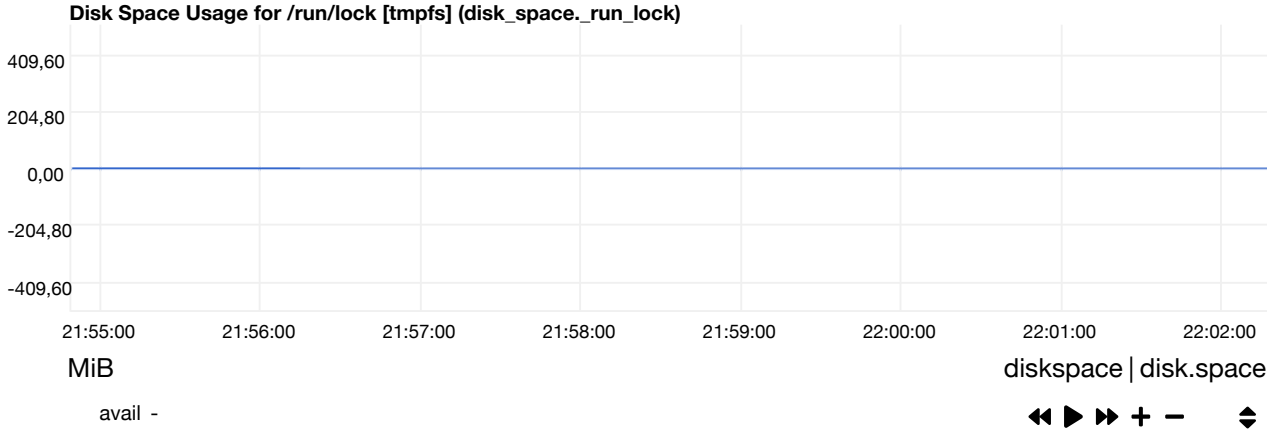


inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.

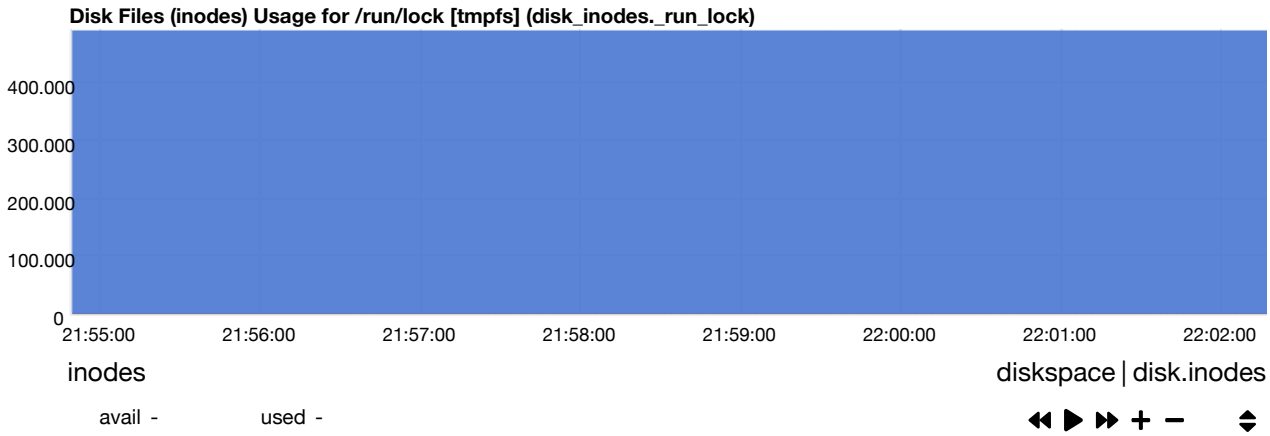


/run/lock

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.

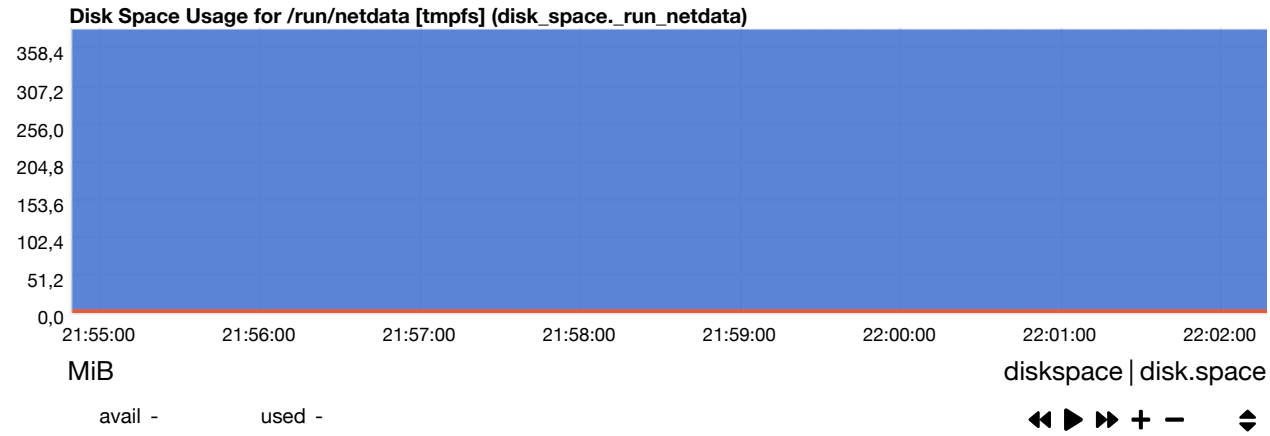


inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.

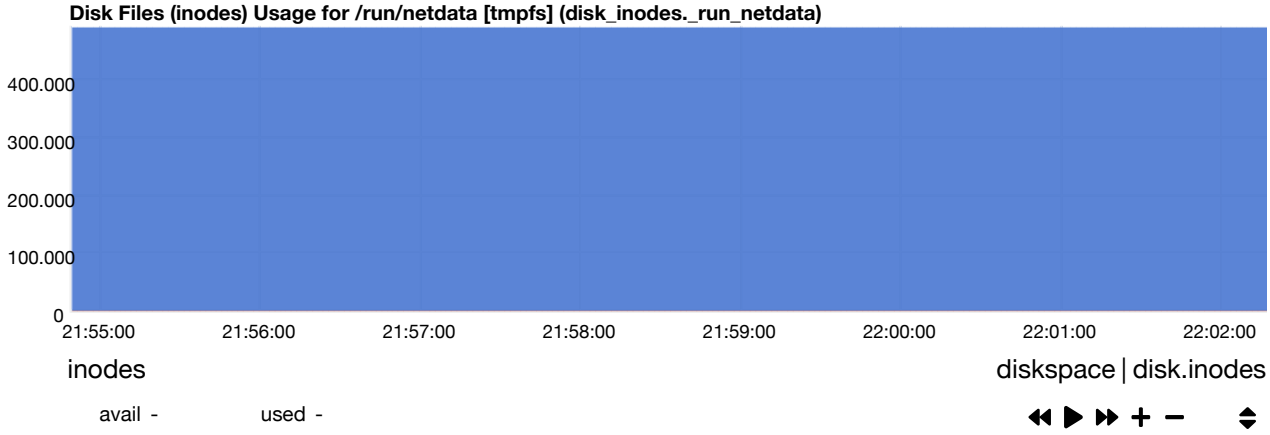


/run/netdata

Disk space utilization. reserved for root is automatically reserved by the system to prevent the root user from getting out of space.



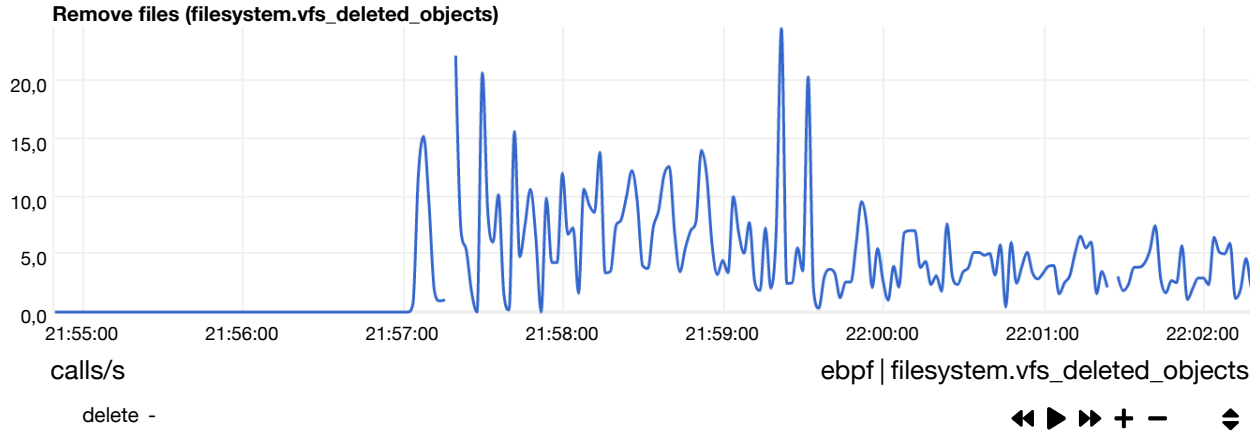
inodes (or index nodes) are filesystem objects (e.g. files and directories). On many types of file system implementations, the maximum number of inodes is fixed at filesystem creation, limiting the maximum number of files the filesystem can hold. It is possible for a device to run out of inodes. When this happens, new files cannot be created on the device, even though there may be free space available.



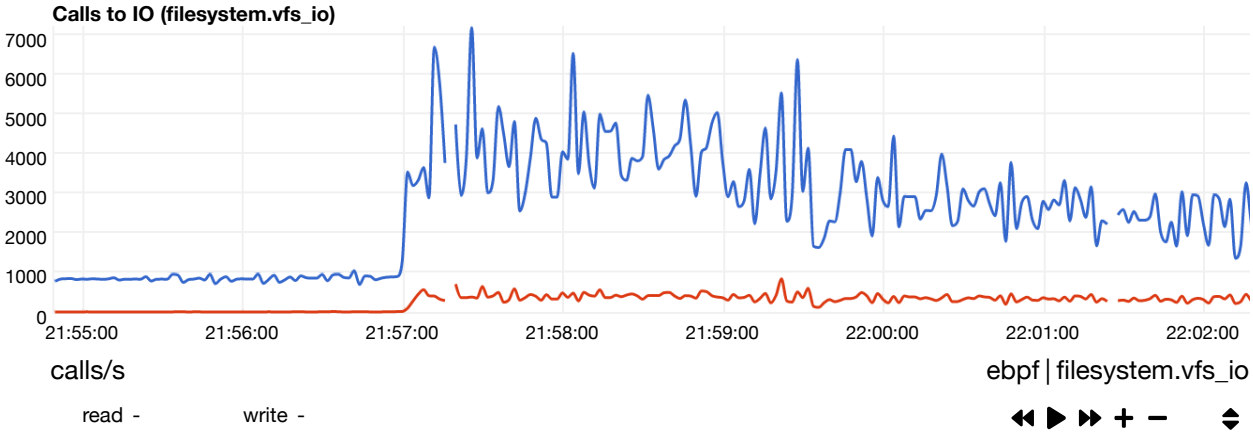
Filesystem

VFS (eBPF)

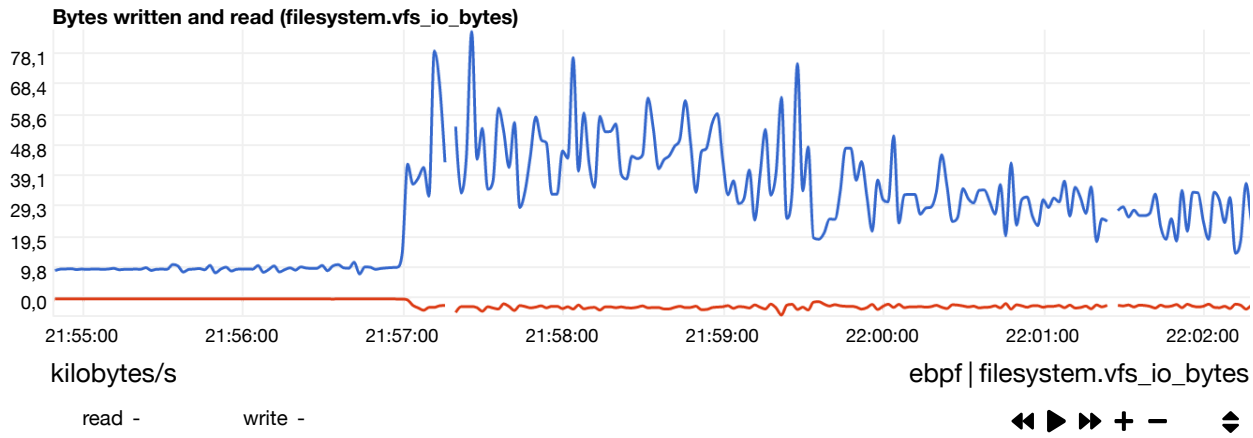
This chart does not show all events that remove files from the file system, because file systems can create their own functions to remove files, it shows calls for the function `vfs_unlink`.



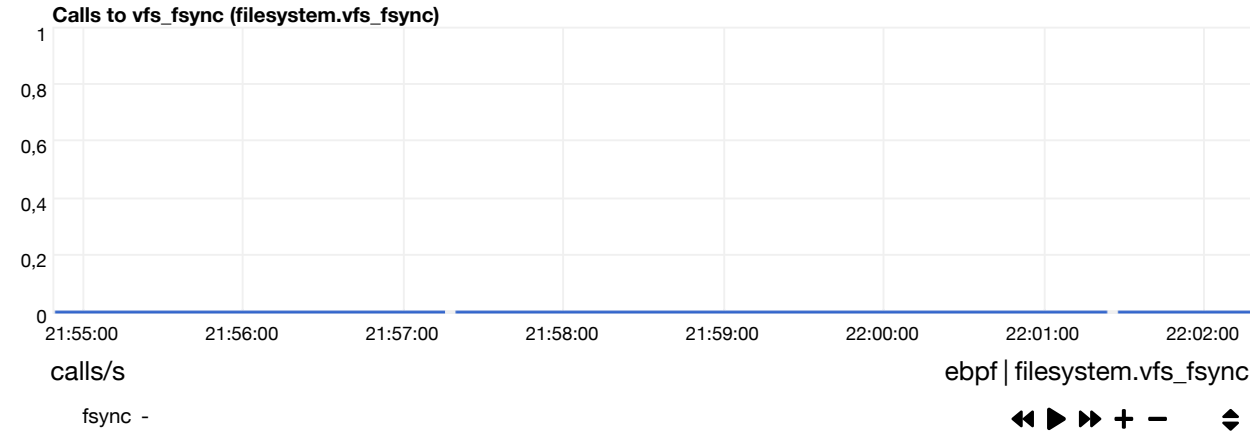
Successful or failed calls to functions `vfs_read` and `vfs_write`. This chart may not show all file system events if it uses other functions to store data on disk.



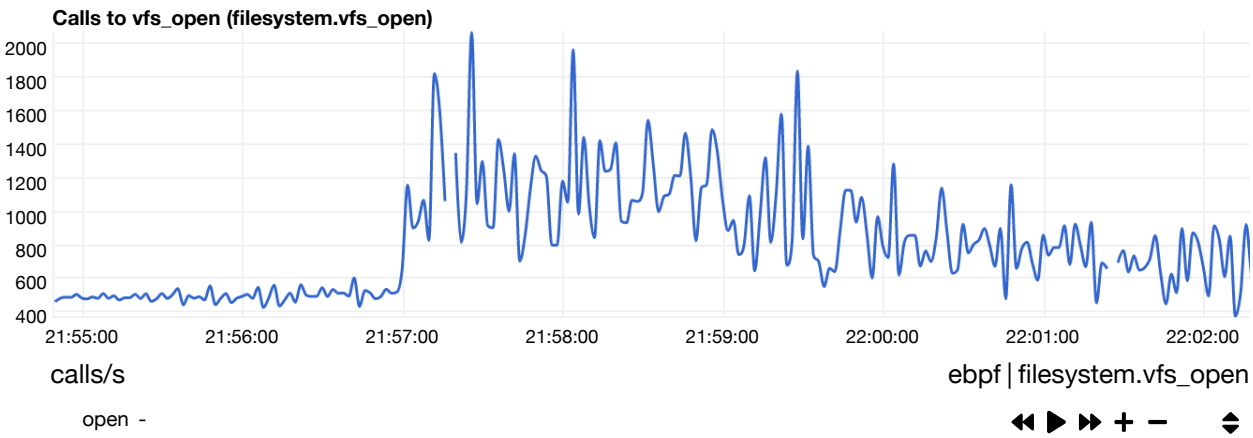
Total of bytes read or written with success using the functions `vfs_read` and `vfs_write`.



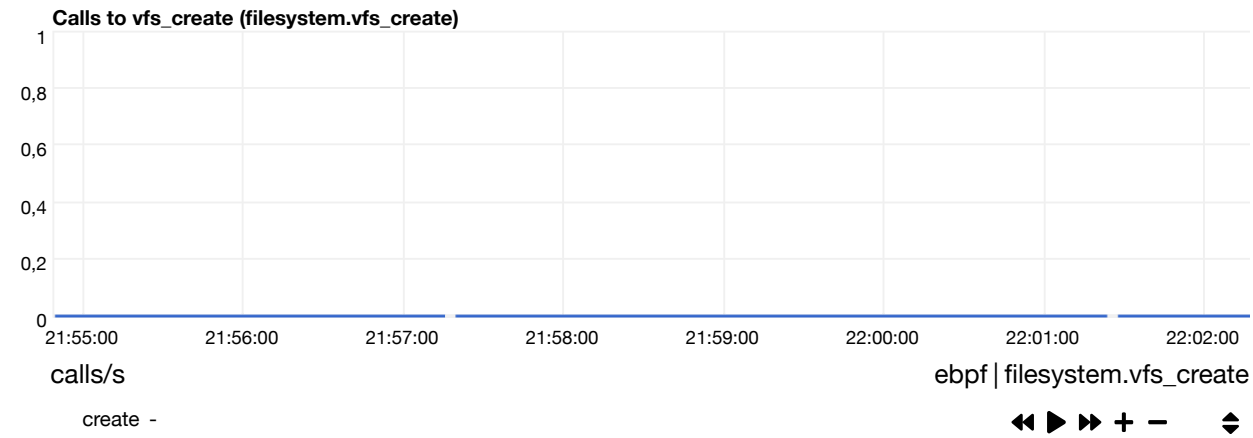
Successful or failed calls to functions `vfs_fsync` .



Successful or failed calls to functions `vfs_open` .



Successful or failed calls to functions `vfs_create` .

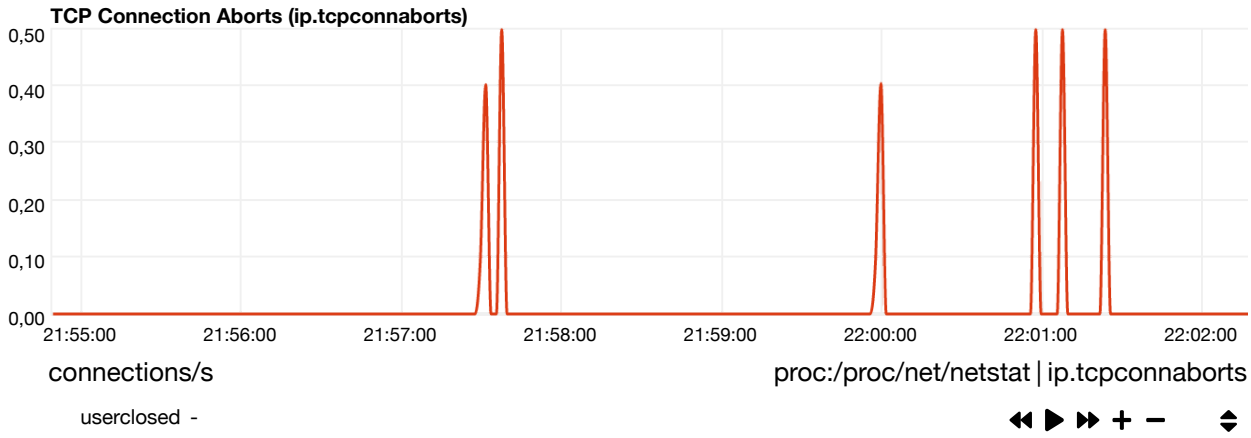


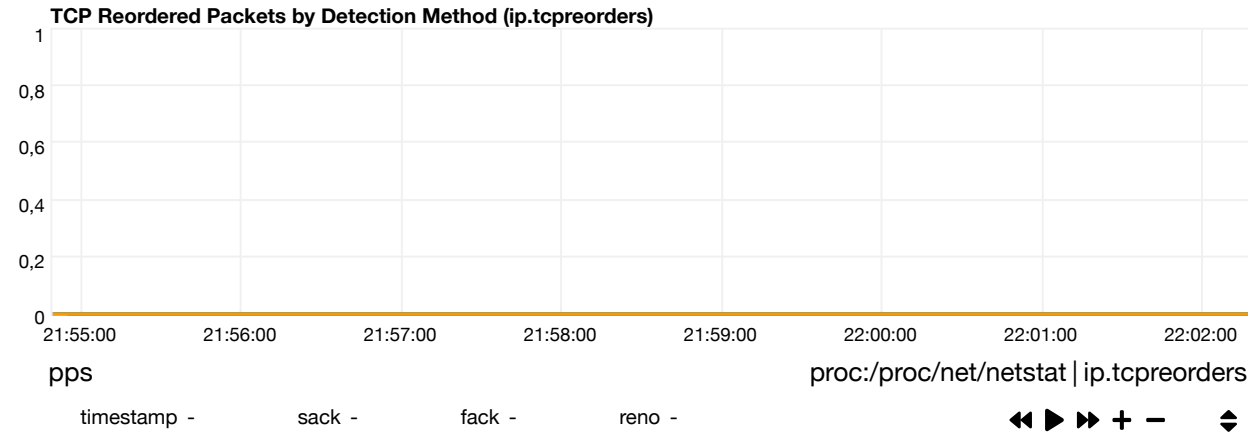
Networking Stack

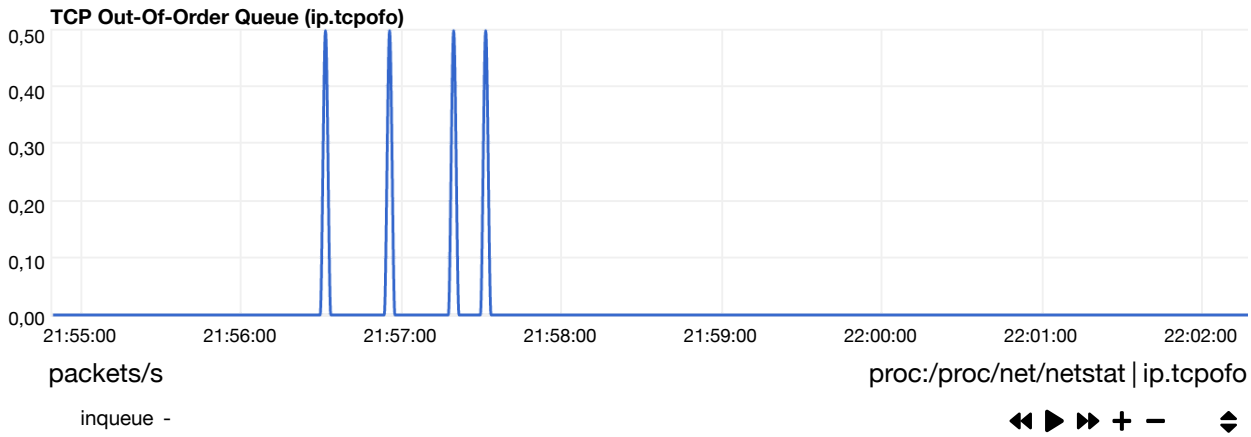
Metrics for the networking stack of the system. These metrics are collected from `/proc/net/netstat` , apply to both IPv4 and IPv6 traffic and are related to operation of the kernel networking stack.

tcp

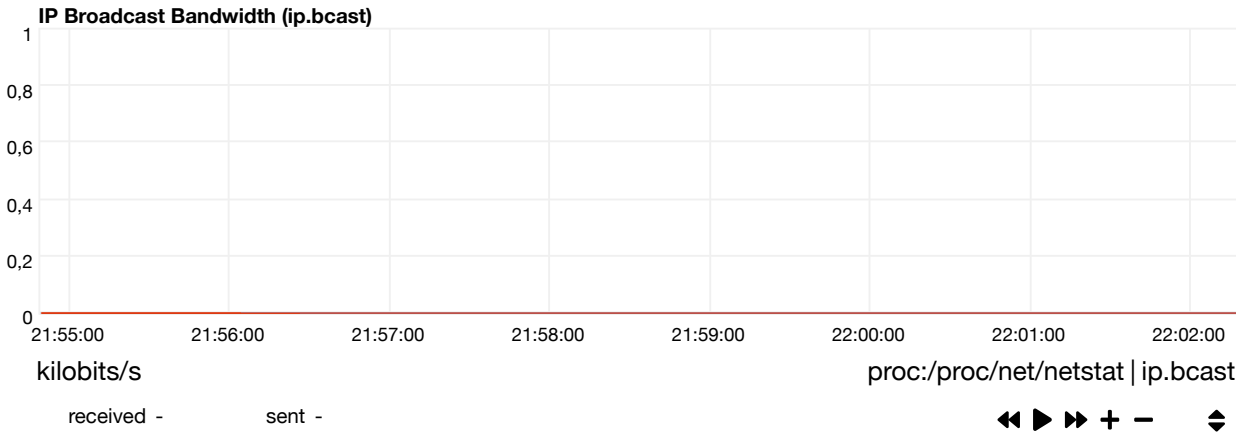
TCP connection aborts. **baddata** (`TCPAbortOnData`) happens while the connection is on `FIN_WAIT1` and the kernel receives a packet with a sequence number beyond the last one for this connection - the kernel responds with `RST` (closes the connection). **userclosed** (`TCPAbortOnClose`) happens when the kernel receives data on an already closed connection and responds with `RST` . **nomemory** (`TCPAbortOnMemory` happens when there are too many orphaned sockets (not attached to an fd) and the kernel has to drop a connection - sometimes it will send an `RST` , sometimes it won't. **timeout** (`TCPAbortOnTimeout`) happens when a connection times out. **linger** (`TCPAbortOnLinger`) happens when the kernel killed a socket that was already closed by the application and lingered around for long enough. **failed** (`TCPAbortFailed`) happens when the kernel attempted to send an `RST` but failed because there was no memory available.

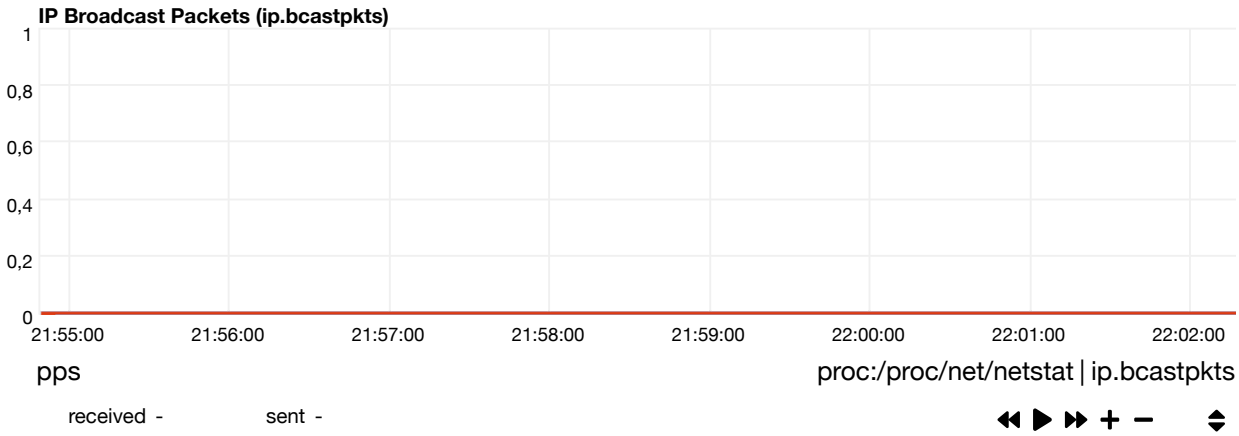






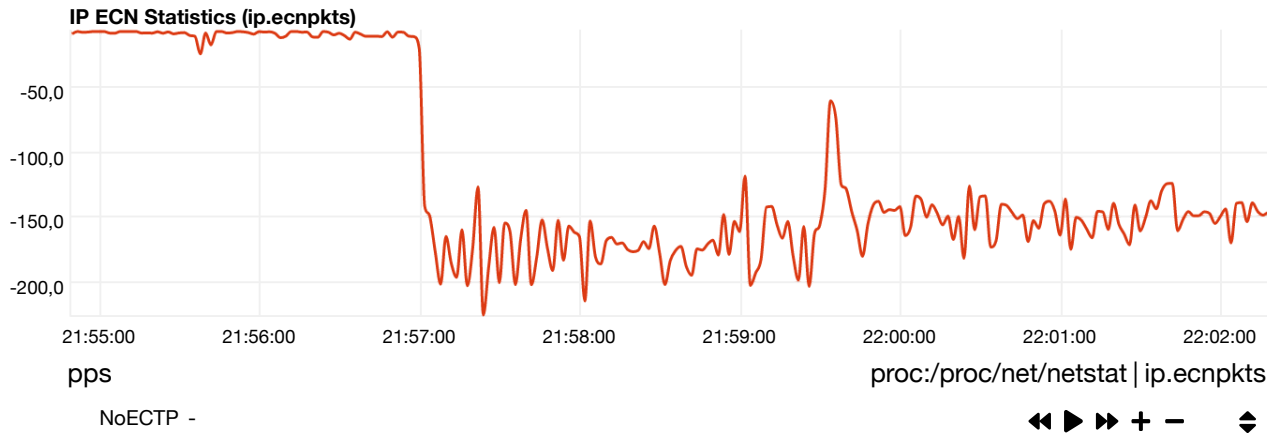
broadcast





ecn

Explicit Congestion Notification (ECN) (https://en.wikipedia.org/wiki/Explicit_Congestion_Notification) is a TCP extension that allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that may be used between two ECN-enabled endpoints when the underlying network infrastructure also supports it.

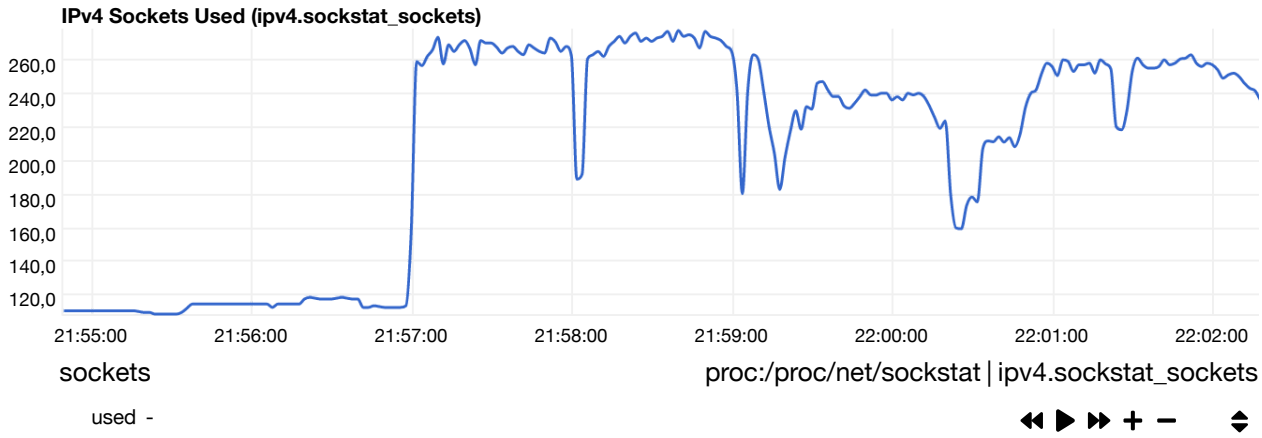


IPv4 Networking

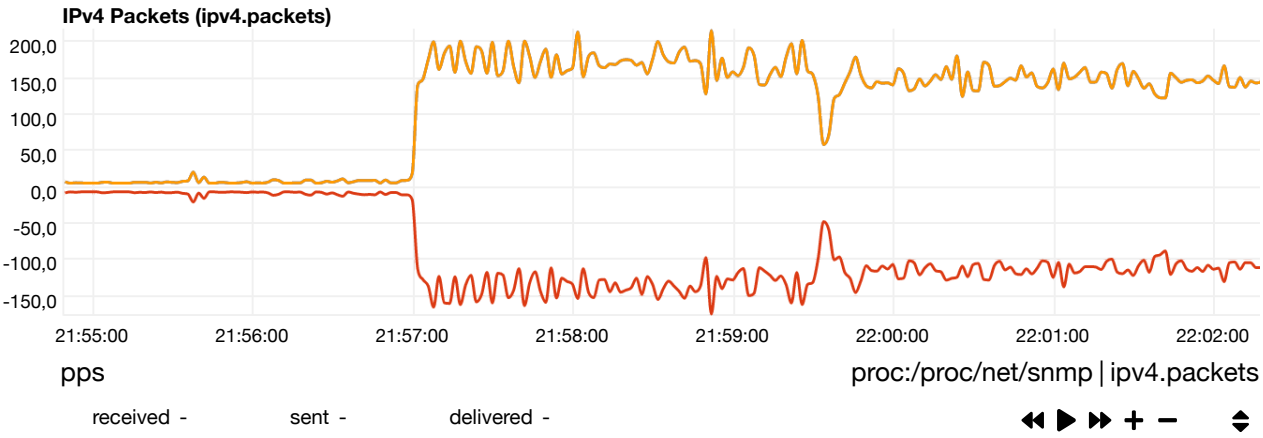
Metrics for the IPv4 stack of the system. Internet Protocol version 4 (IPv4) (https://en.wikipedia.org/wiki/IPv4) is the fourth version of the Internet Protocol (IP). It is one of the core protocols of standards-based internetworking methods in the Internet. IPv4 is a connectionless protocol for use on packet-switched networks. It operates on a best effort delivery model, in that it does not guarantee

delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol, such as the Transmission Control Protocol (TCP).

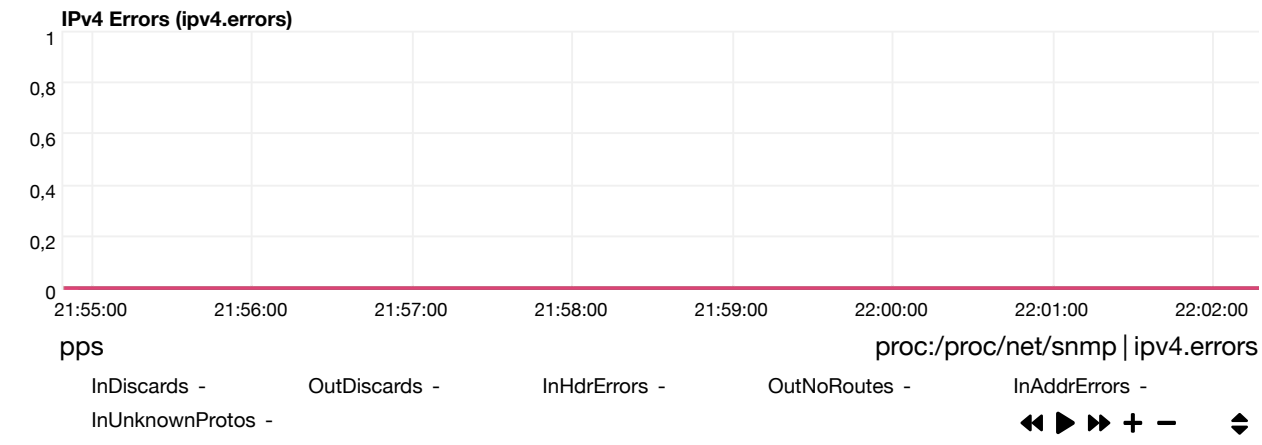
sockets



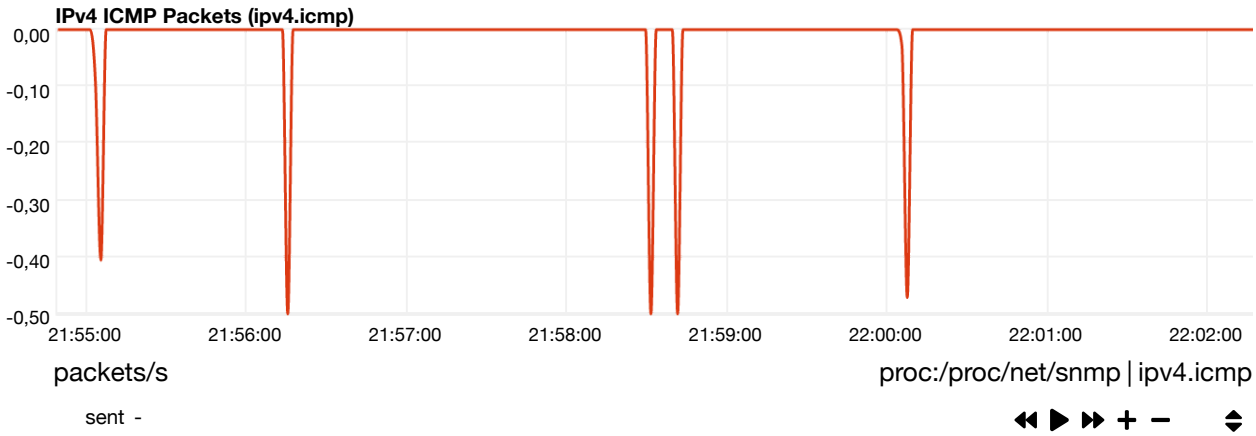
packets

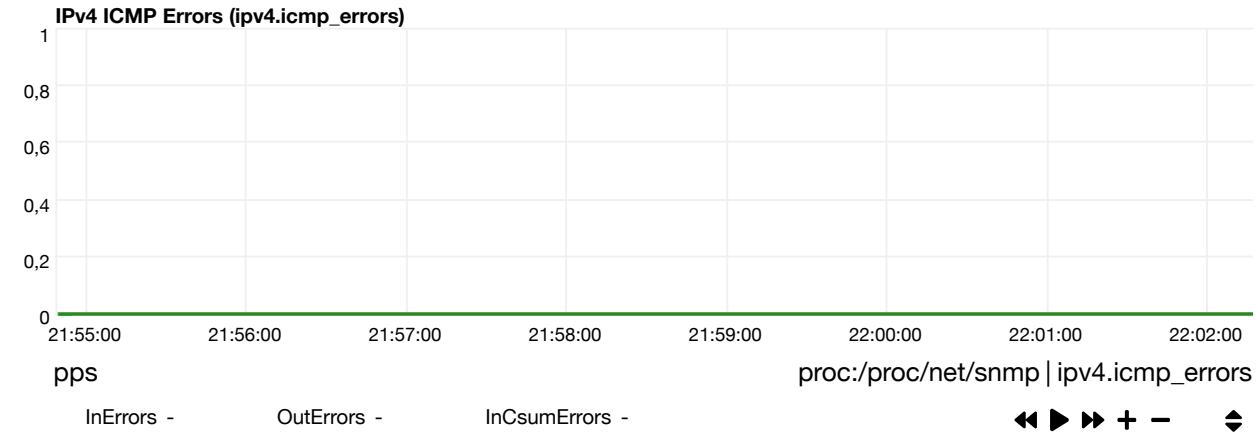


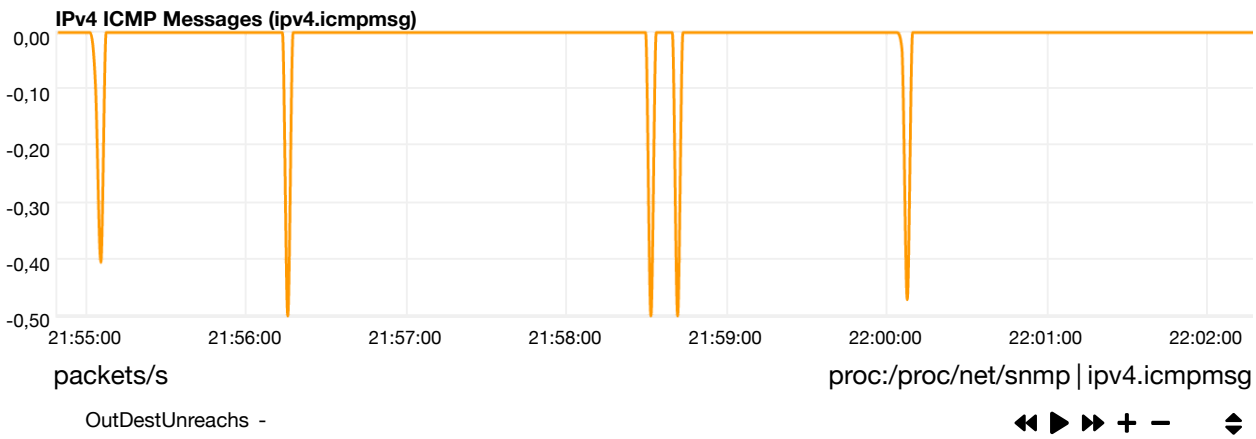
errors



icmp

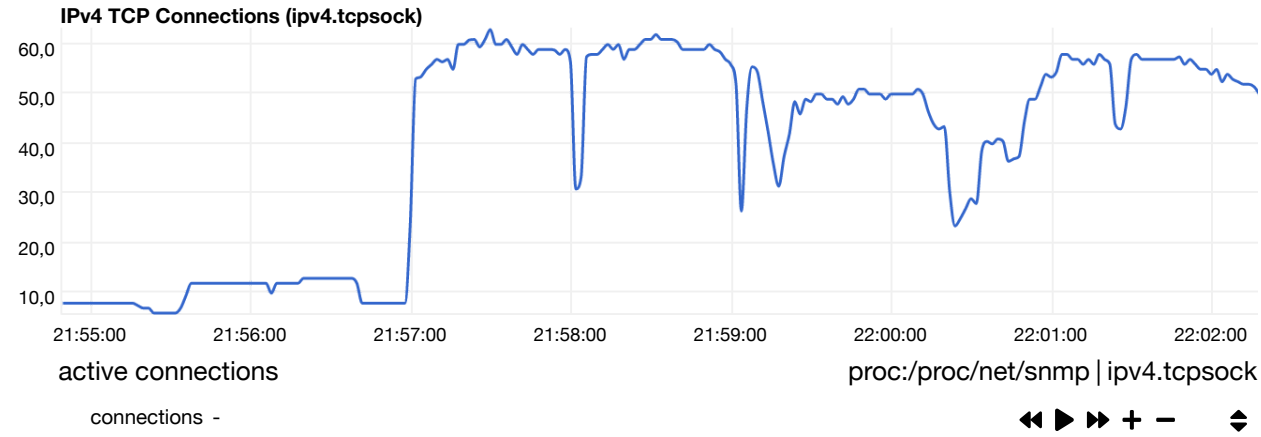

IPv4 ICMP Packets (ipv4.icmp)
0,00
-0,10
-0,20
-0,30
-0,40
-0,50
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
packets/s
proc:/proc/net/snmp | ipv4.icmp
sent -

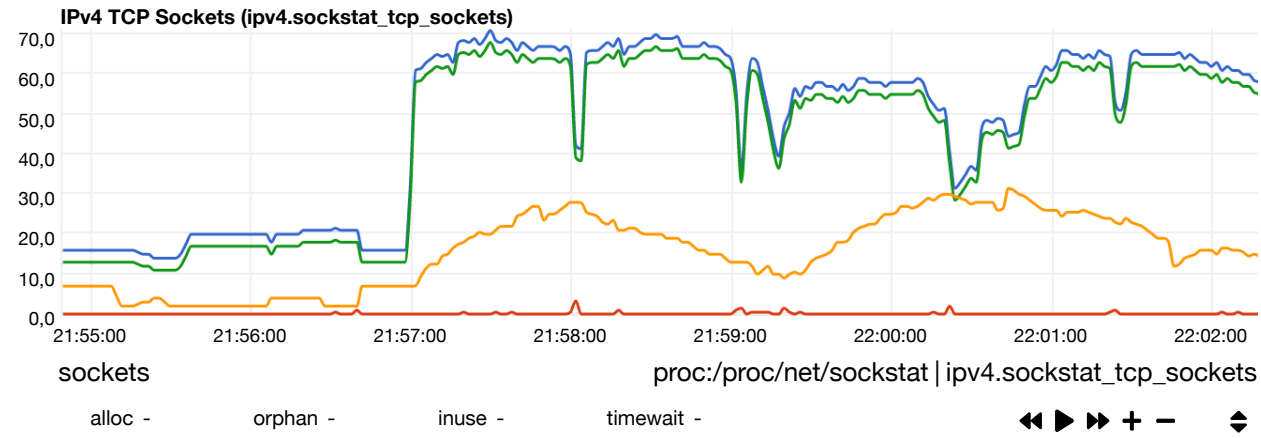

IPv4 ICMP Errors (ipv4.icmp_errors)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
pps
proc:/proc/net/snmp | ipv4.icmp_errors
InErrors -
OutErrors -
InCsumErrors -


IPv4 ICMP Messages (ipv4.icmpmsg)
0,00
-0,10
-0,20
-0,30
-0,40
-0,50
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
packets/s
proc:/proc/net/snmp | ipv4.icmpmsg
OutDestUnreachs -

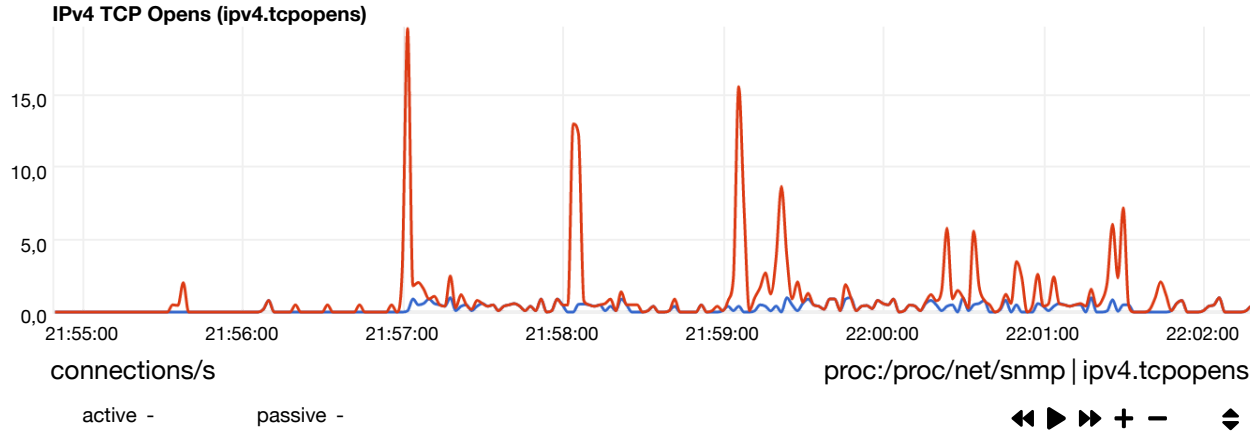
tcp

The number of established TCP connections (known as `CurrEstab`). This is a snapshot of the established connections at the time of measurement (i.e. a connection established and a connection disconnected within the same iteration will not affect this metric).



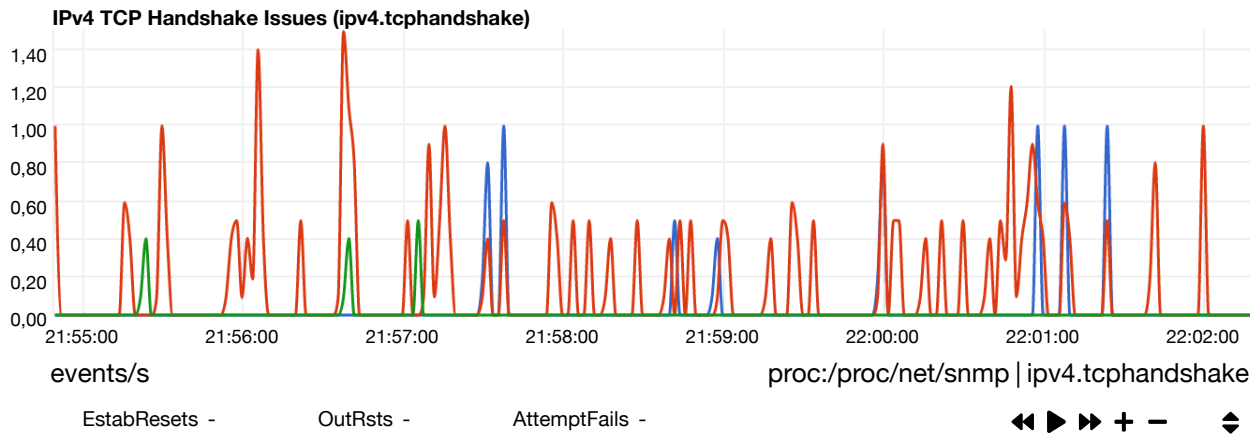


active or `ActiveOpens` is the number of outgoing TCP **connections attempted** by this host.
passive or `PassiveOpens` is the number of incoming TCP **connections accepted** by this host.

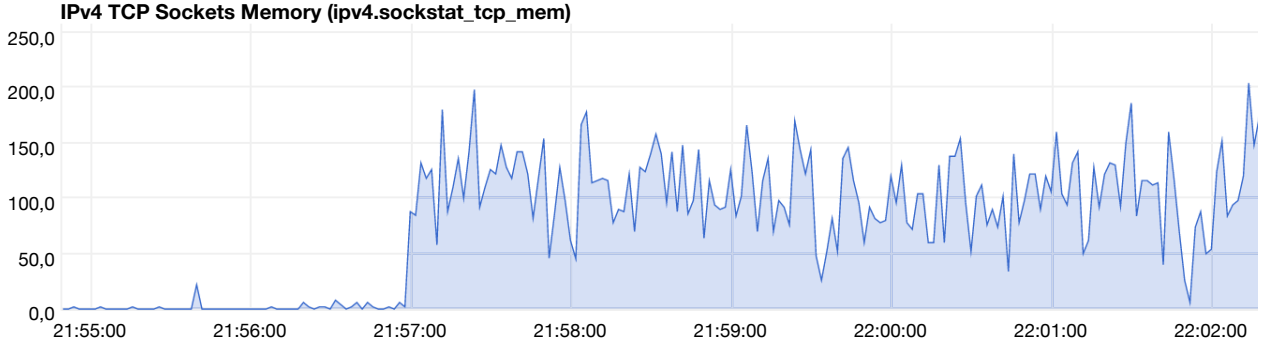


`InErrs` is the number of TCP segments received in error (including header too small, checksum errors, sequence errors, bad packets - for both IPv4 and IPv6). `InCsumErrors` is the number of TCP segments received with checksum errors (for both IPv4 and IPv6). `RetransSegs` is the number of TCP segments retransmitted.

`EstabResets` is the number of established connections resets (i.e. connections that made a direct transition from `ESTABLISHED` or `CLOSE_WAIT` to `CLOSED`). `OutRsts` is the number of TCP segments sent, with the `RST` flag set (for both IPv4 and IPv6). `AttemptFails` is the number of times TCP connections made a direct transition from either `SYN_SENT` or `SYN_RECV` to `CLOSED`, plus the number of times TCP connections made a direct transition from the `SYN_RECV` to `LISTEN`. `TCPSynRetrans` shows retries for new outbound TCP connections, which can indicate general connectivity issues or backlog on the remote host.



IPv4 TCP Sockets Memory (ipv4.sockstat_tcp_mem)

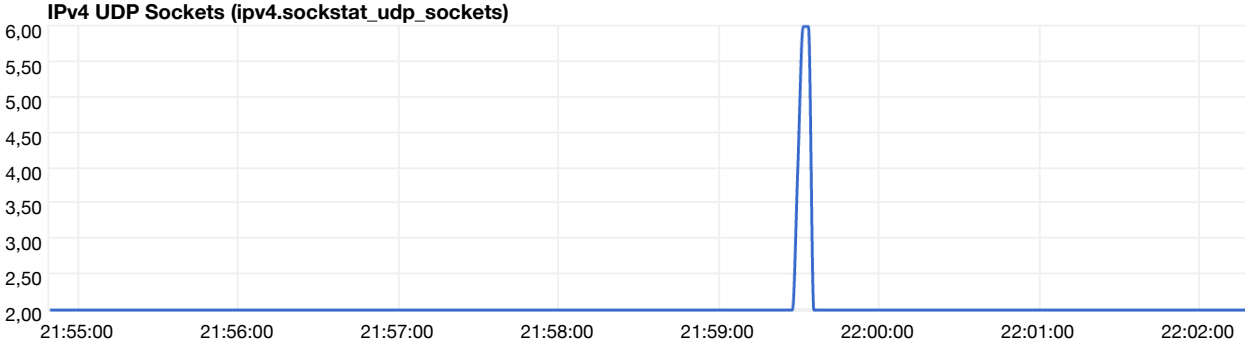


KiB proc:/proc/net/sockstat | ipv4.sockstat_tcp_mem

mem -

udp

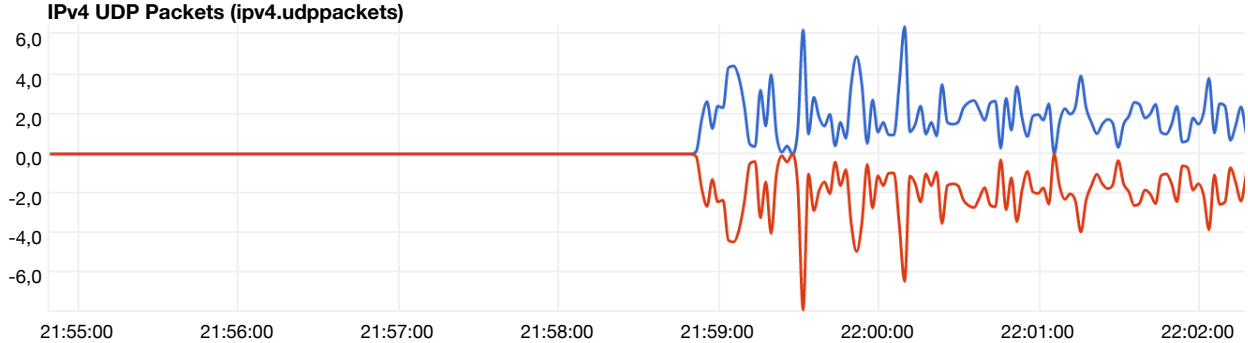
IPv4 UDP Sockets (ipv4.sockstat_udp_sockets)



sockets proc:/proc/net/sockstat | ipv4.sockstat_udp_sockets

inuse -

IPv4 UDP Packets (ipv4.udppackets)



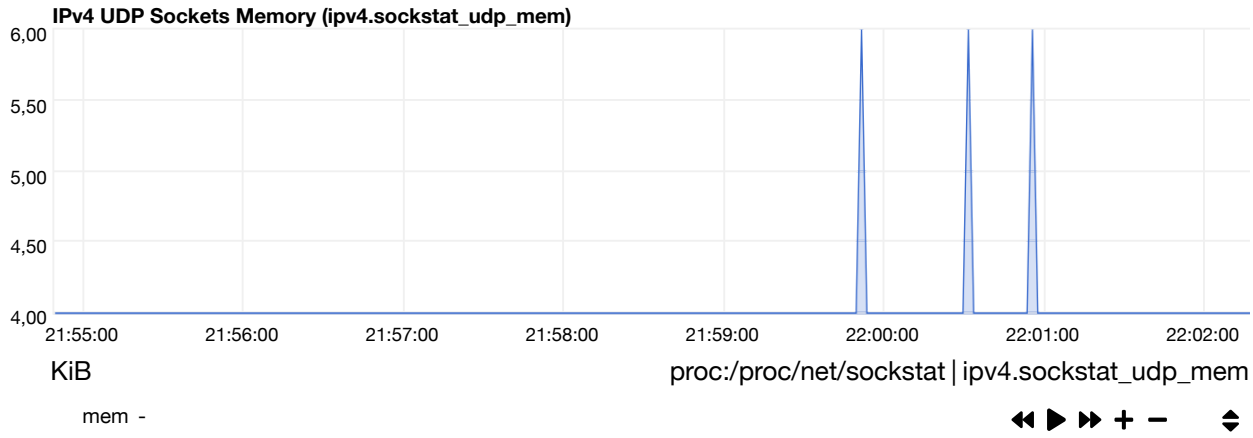
pps proc:/proc/net/snmp | ipv4.udppackets

received - sent -

IPv4 UDP Errors (ipv4.udperrors)

events/s proc:/proc/net/snmp | ipv4.udperrors

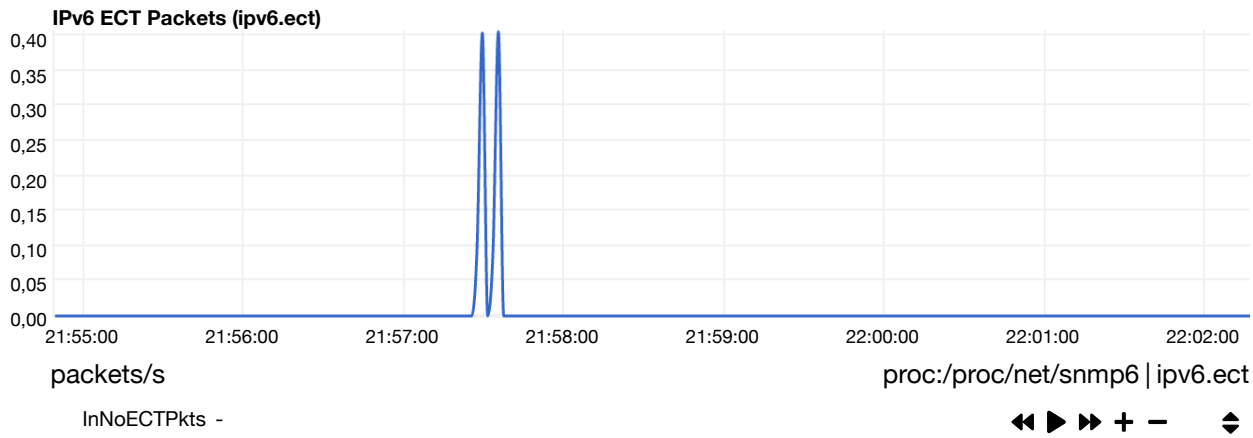
NoPorts -



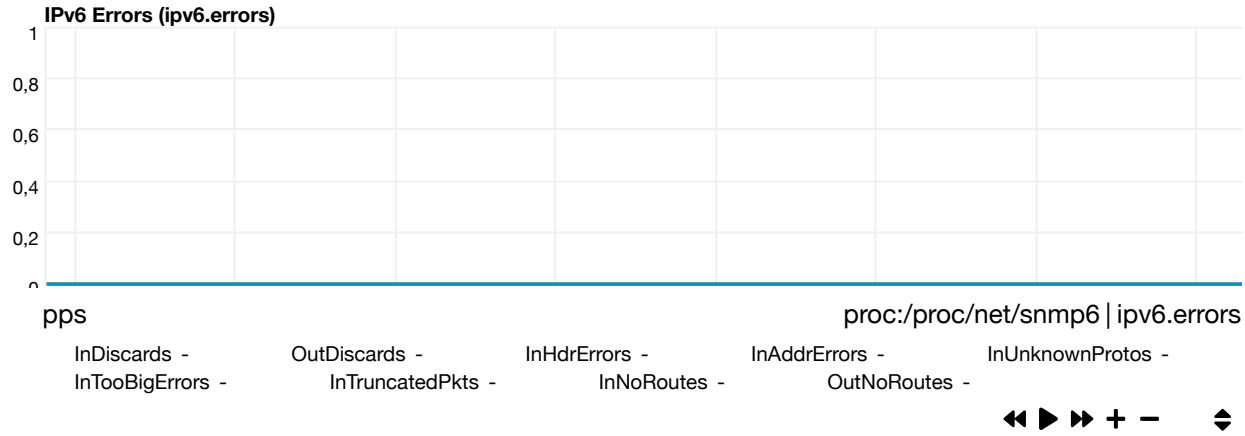
IPv6 Networking

Metrics for the IPv6 stack of the system. Internet Protocol version 6 (IPv6) (https://en.wikipedia.org/wiki/IPv6) is the most recent version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion. IPv6 is intended to replace IPv4.

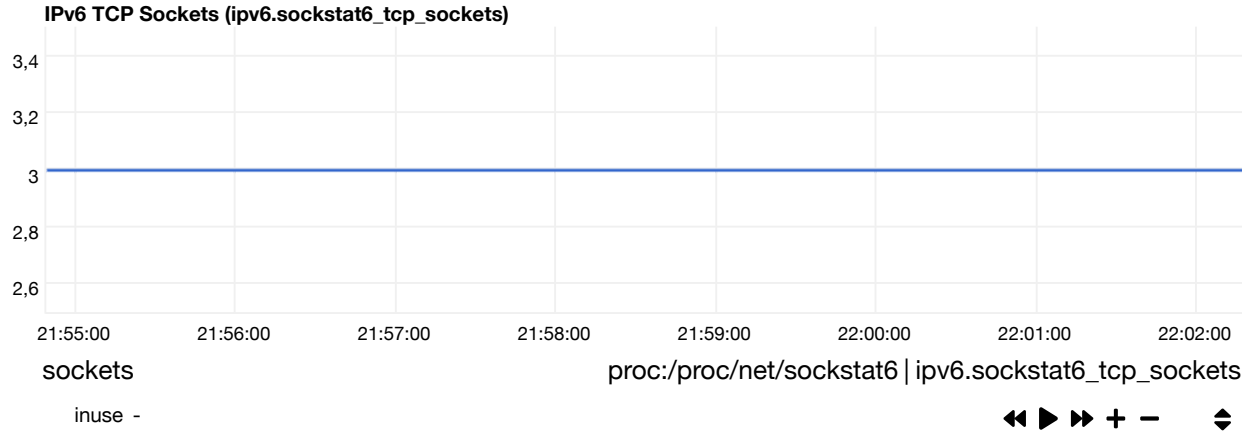
packets



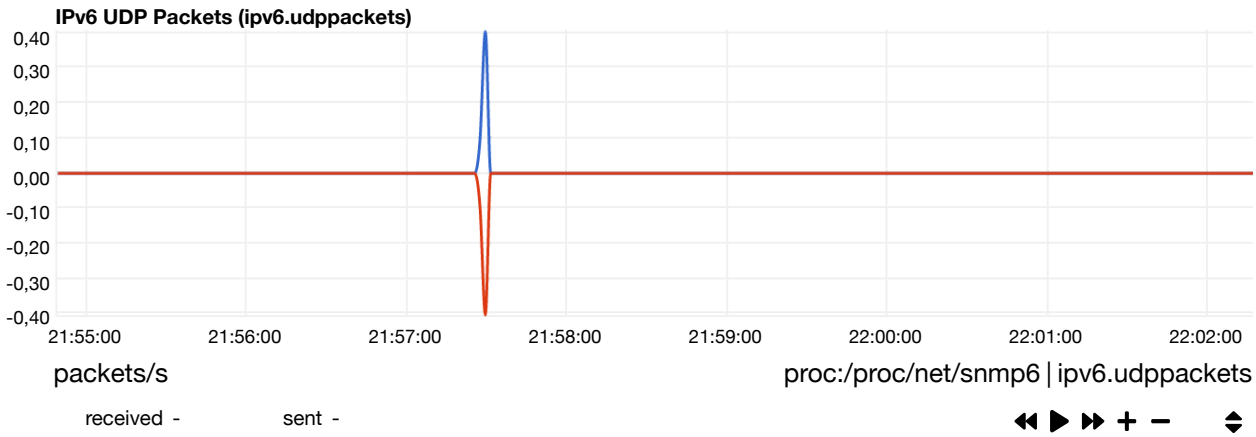
errors


IPv6 Errors (ipv6.errors)
1
0,8
0,6
0,4
0,2
0
pps
proc:/proc/net/snmp6 | ipv6.errors
InDiscards -
OutDiscards -
InHdrErrors -
InAddrErrors -
InUnknownProtos -
InTooBigErrors -
InTruncatedPkts -
InNoRoutes -
OutNoRoutes -

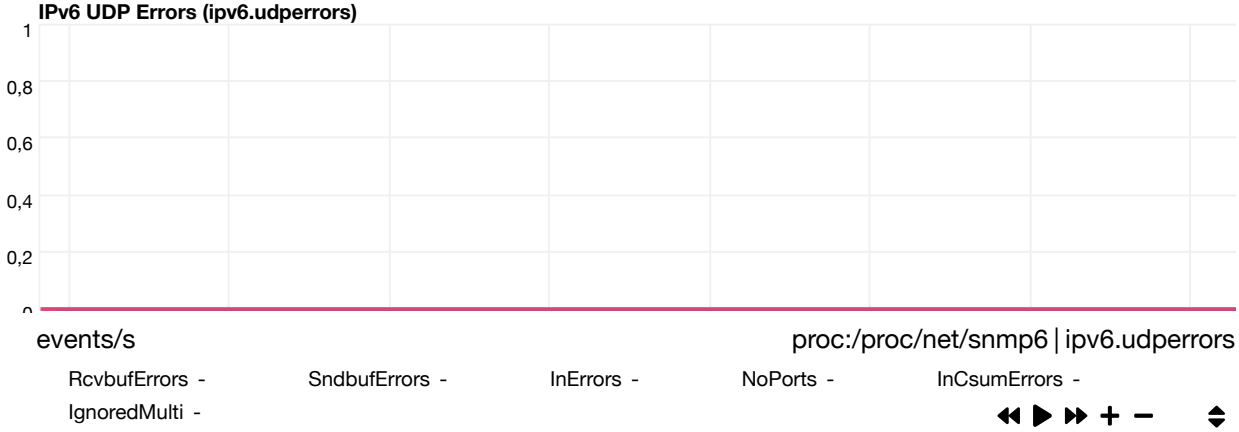
tcp6


IPv6 TCP Sockets (ipv6.sockstat6_tcp_sockets)
3,4
3,2
3
2,8
2,6
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
sockets
proc:/proc/net/sockstat6 | ipv6.sockstat6_tcp_sockets
inuse -

udp6

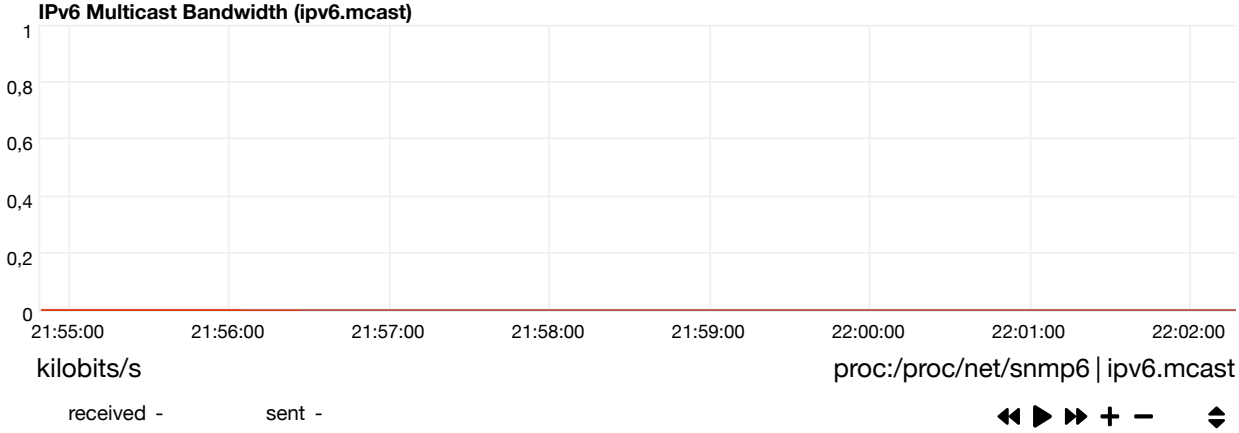

IPv6 UDP Packets (ipv6.udppackets)
0,40
0,30
0,20
0,10
0,00
-0,10
-0,20
-0,30
-0,40
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
packets/s
proc:/proc/net/snmp6 | ipv6.udppackets
received -
sent -

IPv6 UDP Errors (ipv6.udperrors)

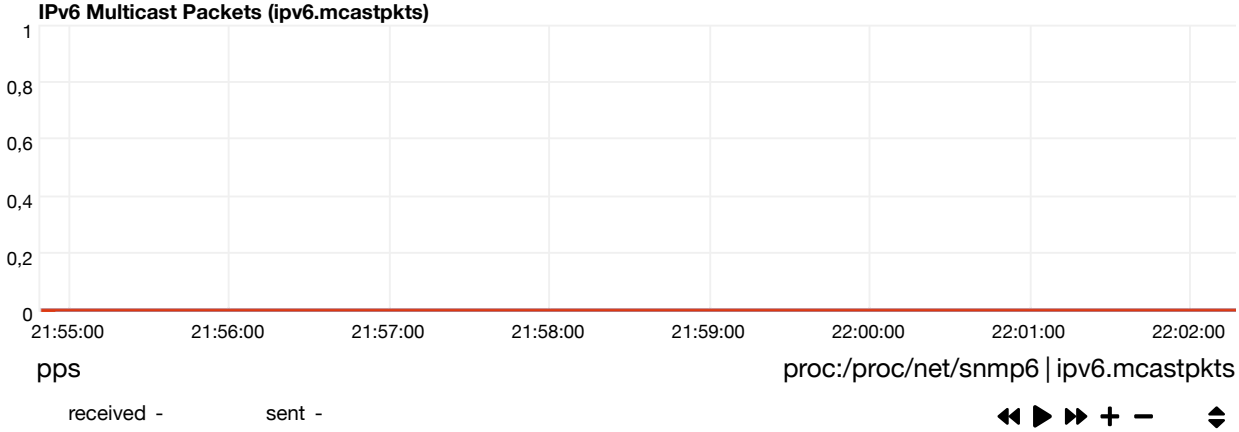


multicast6

IPv6 Multicast Bandwidth (ipv6.mcast)

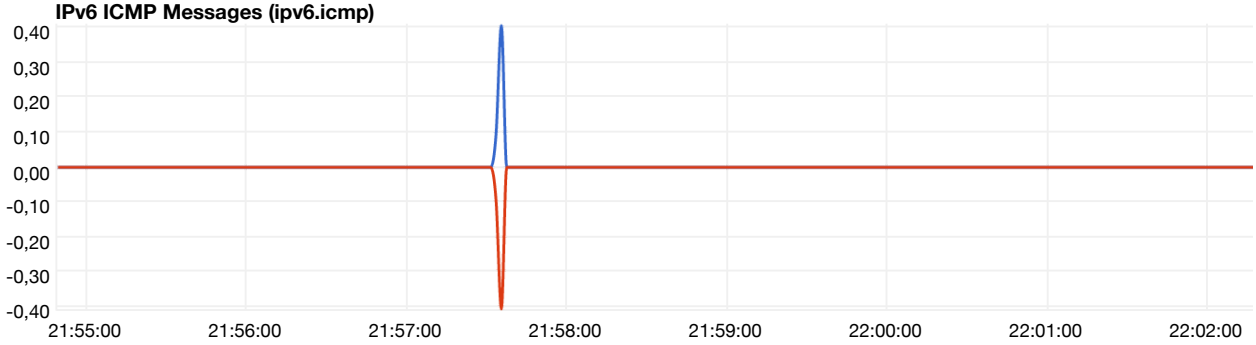


IPv6 Multicast Packets (ipv6.mcastpkts)



icmp6

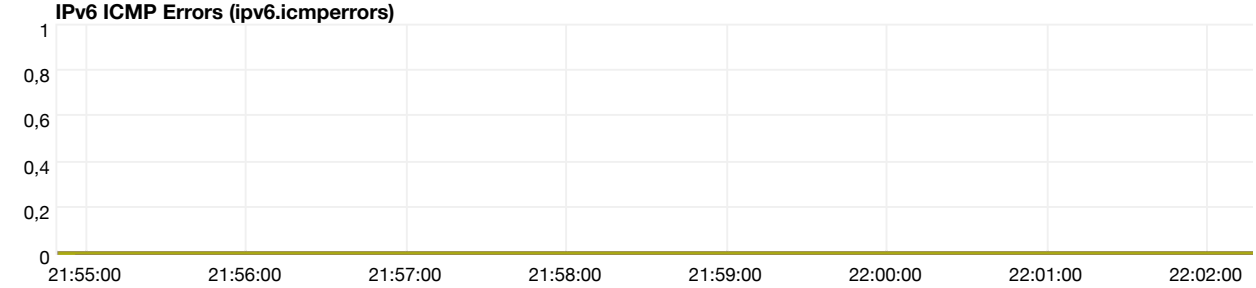
IPv6 ICMP Messages (ipv6.icmp)



messages/s proc:/proc/net/snmp6 | ipv6.icmp

received - sent -

IPv6 ICMP Errors (ipv6.icmperrors)



errors/s proc:/proc/net/snmp6 | ipv6.icmperrors

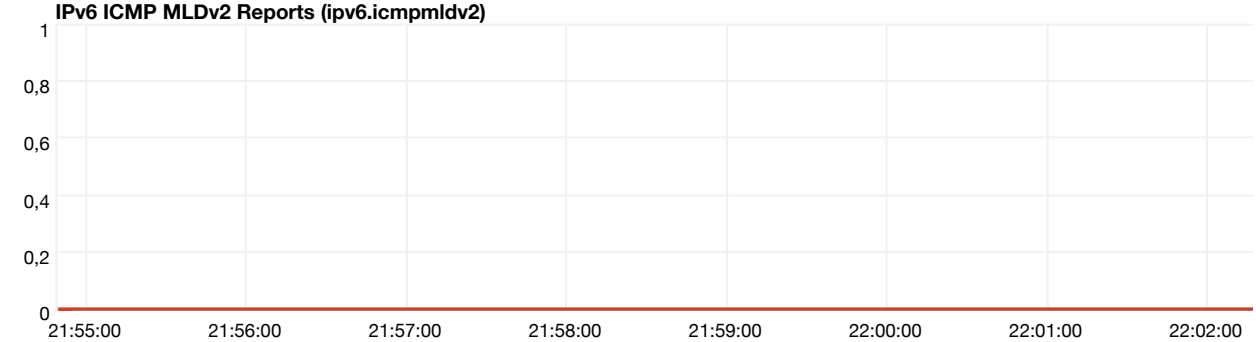
InErrors - OutErrors - InCsumErrors - InDestUnreachs - InPktTooBigs - InTimeExcds - InParmProblems - OutDestUnreachs - OutPktTooBigs - OutTimeExcds - OutParmProblems -

IPv6 Neighbor Messages (ipv6.icmpneighbor)

messages/s proc:/proc/net/snmp6 | ipv6.icmpneighbor

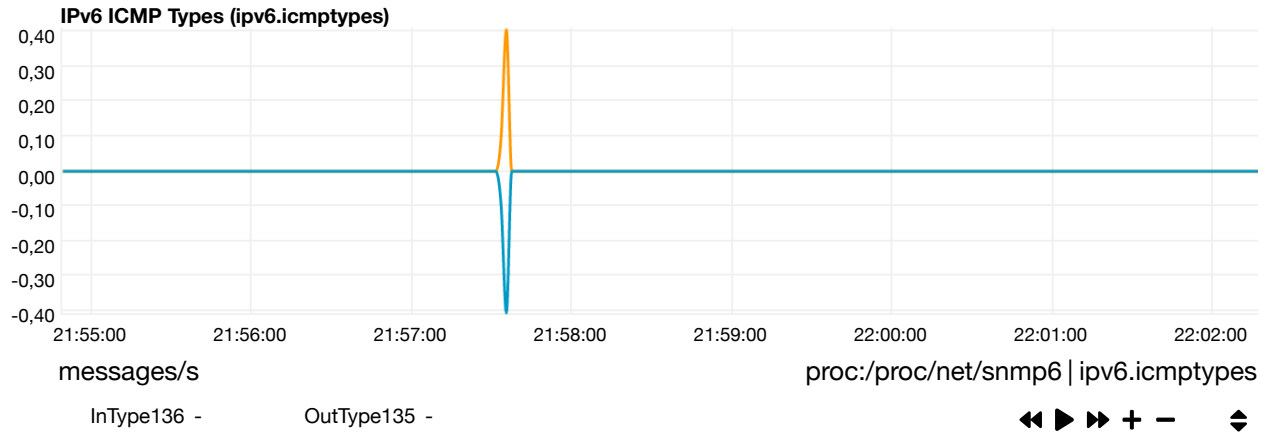
OutSolicits - InAdvertisements -

IPv6 ICMP MLDv2 Reports (ipv6.icmpmldv2)



reports/s proc:/proc/net/snmp6 | ipv6.icmpmldv2

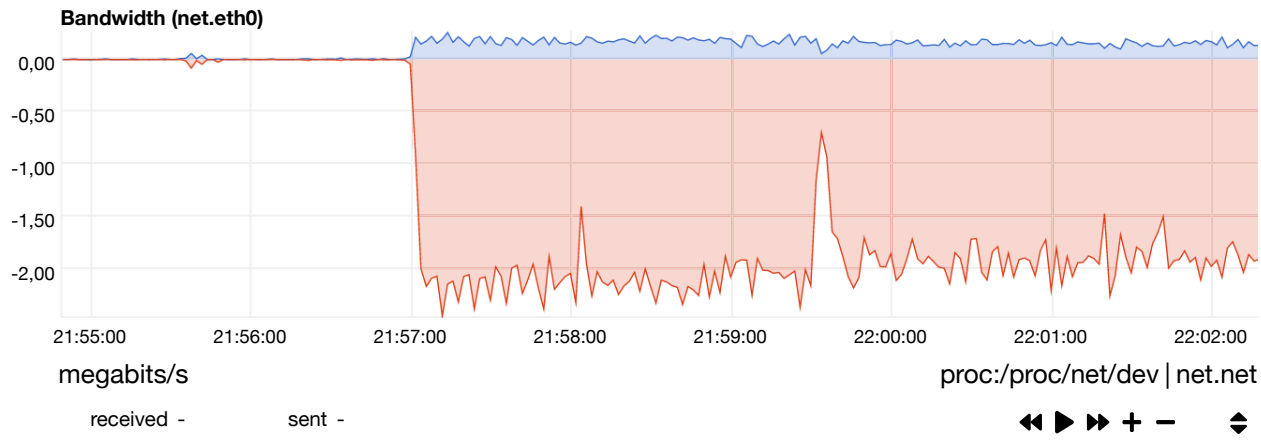
received - sent -

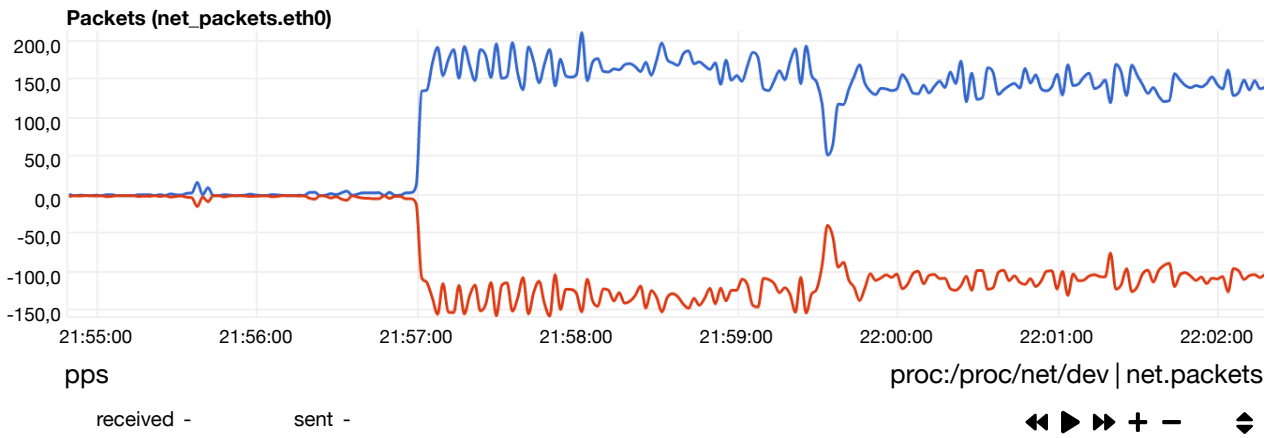


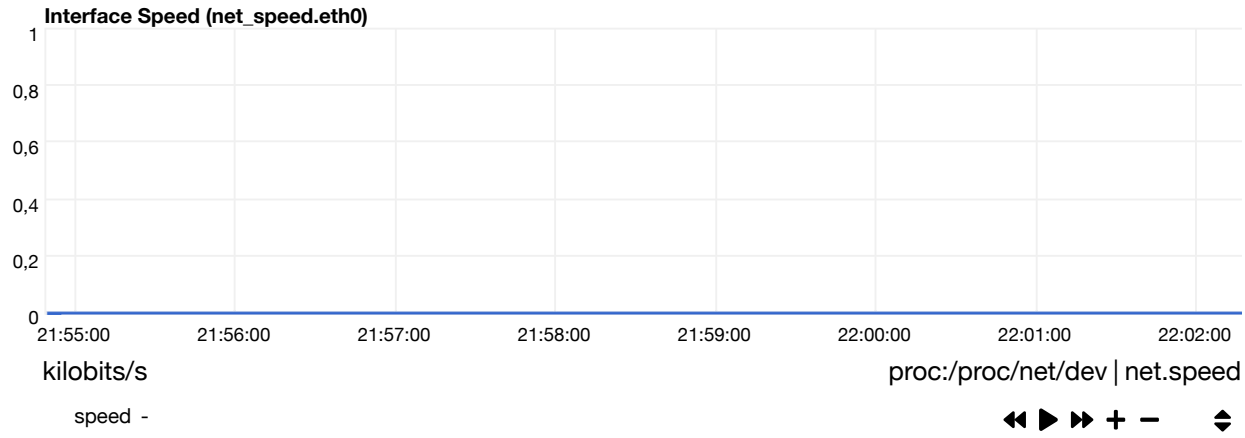
Network Interfaces

Performance metrics for network interfaces.

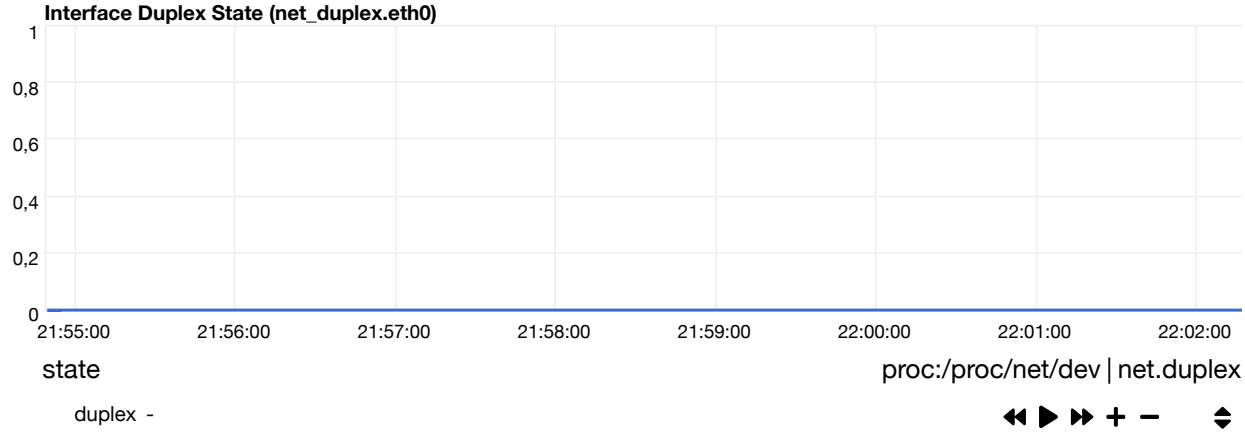
eth0



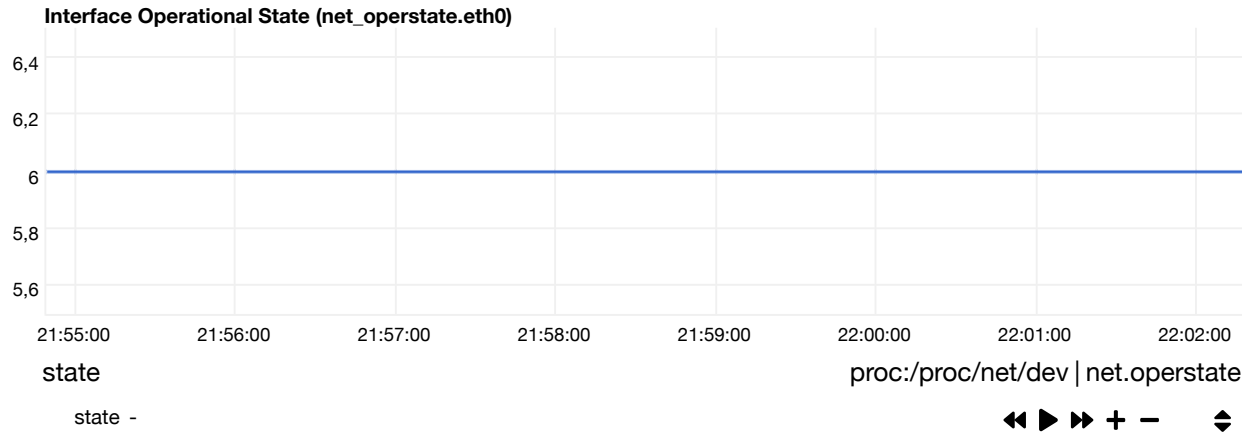




State map: 0 - unknown, 1 - half duplex, 2 - full duplex

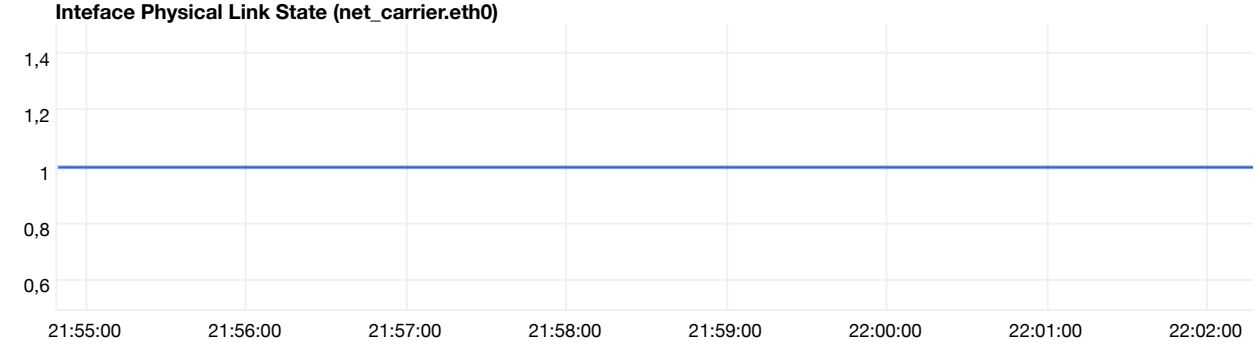


State map: 0 - unknown, 1 - notpresent, 2 - down, 3 - lowerlayerdown, 4 - testing, 5 - dormant, 6 - up



State map: 0 - down, 1 - up

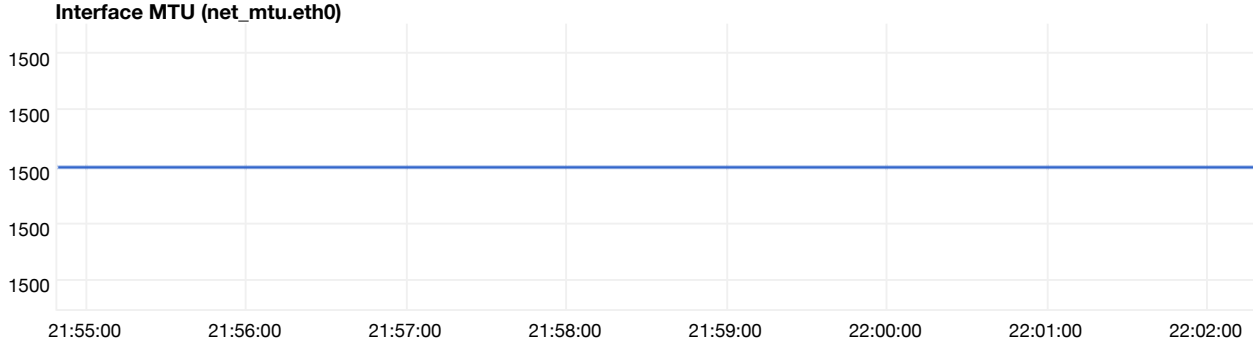
Inteface Physical Link State (net_carrier.eth0)



state proc:/proc/net/dev | net.carrier

carrier -

Interface MTU (net_mtu.eth0)



octets proc:/proc/net/dev | net.mtu

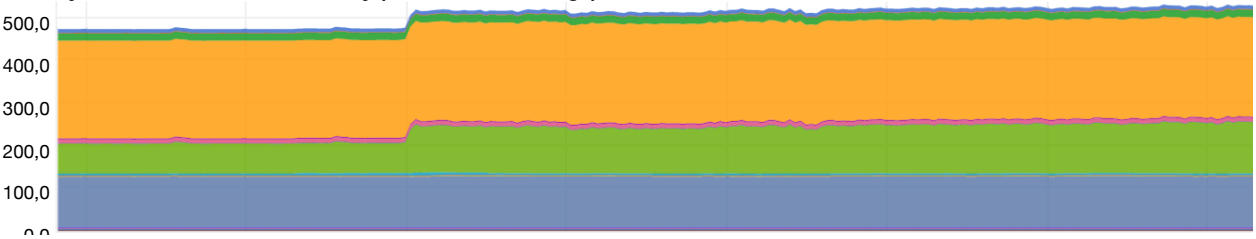
mtu -

systemd Services

Resources utilization of systemd services. netdata monitors all systemd services via CGROUPS (the resources accounting used by containers).

mem

Systemd Services Used Memory (services.mem_usage)



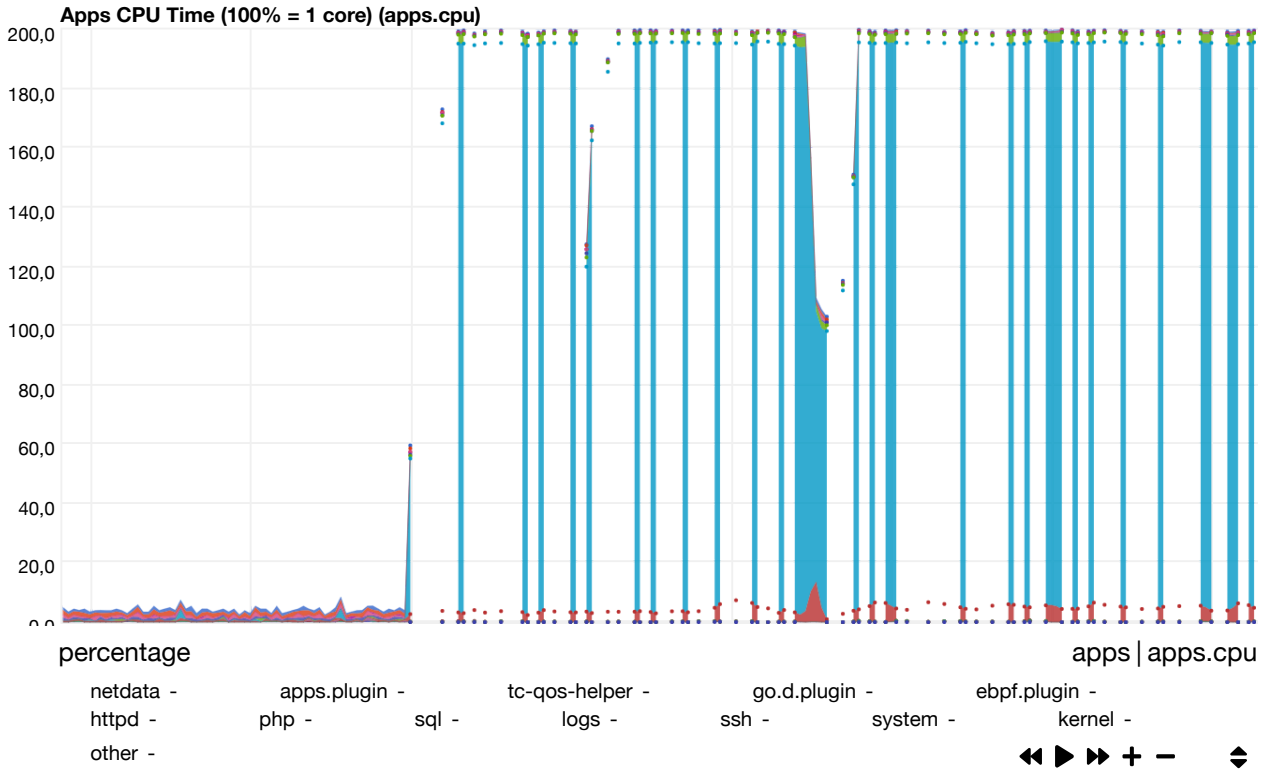
MiB cgroups:systemd | services.mem_usage

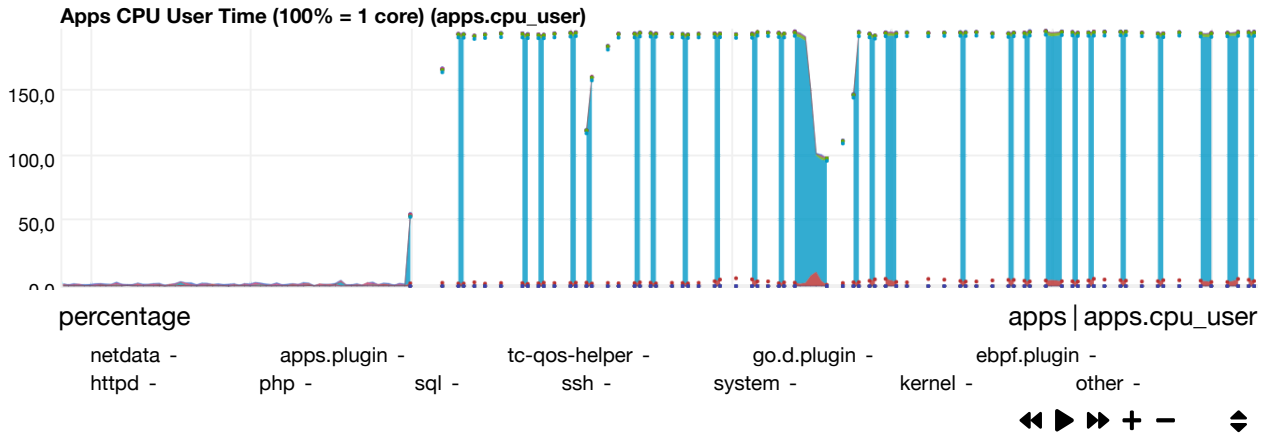
systemd-udevd - cron - nginx - mariadb - networking -
systemd-journald - atd - php7.3-fpm - ssh - cloud-final -
rsyslog - netdata - qemu-guest-agent - cloud-init - dbus -
systemd-timesyncd - systemd-logind -

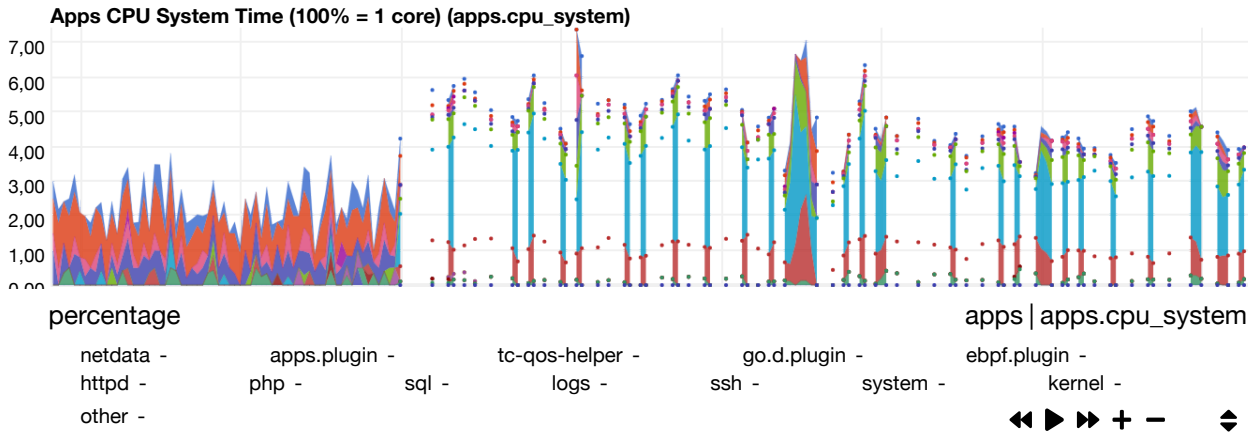
Applications

Per application statistics are collected using netdata's `apps.plugin` . This plugin walks through all processes and aggregates statistics for applications of interest, defined in `/etc/netdata/apps_groups.conf` , which can be edited by running `$ /etc/netdata/edit-config apps_groups.conf` (the default is here (https://github.com/netdata/netdata/blob/master/collectors/apps.plugin/apps_groups.conf)). The plugin internally builds a process tree (much like `ps fax` does), and groups processes together (evaluating both child and parent processes) so that the result is always a chart with a predefined set of dimensions (of course, only application groups found running are reported). The reported values are compatible with `top` , although the netdata plugin counts also the resources of exited children (unlike `top` which shows only the resources of the currently running processes). So for processes like shell scripts, the reported values include the resources used by the commands these scripts run within each timeframe.

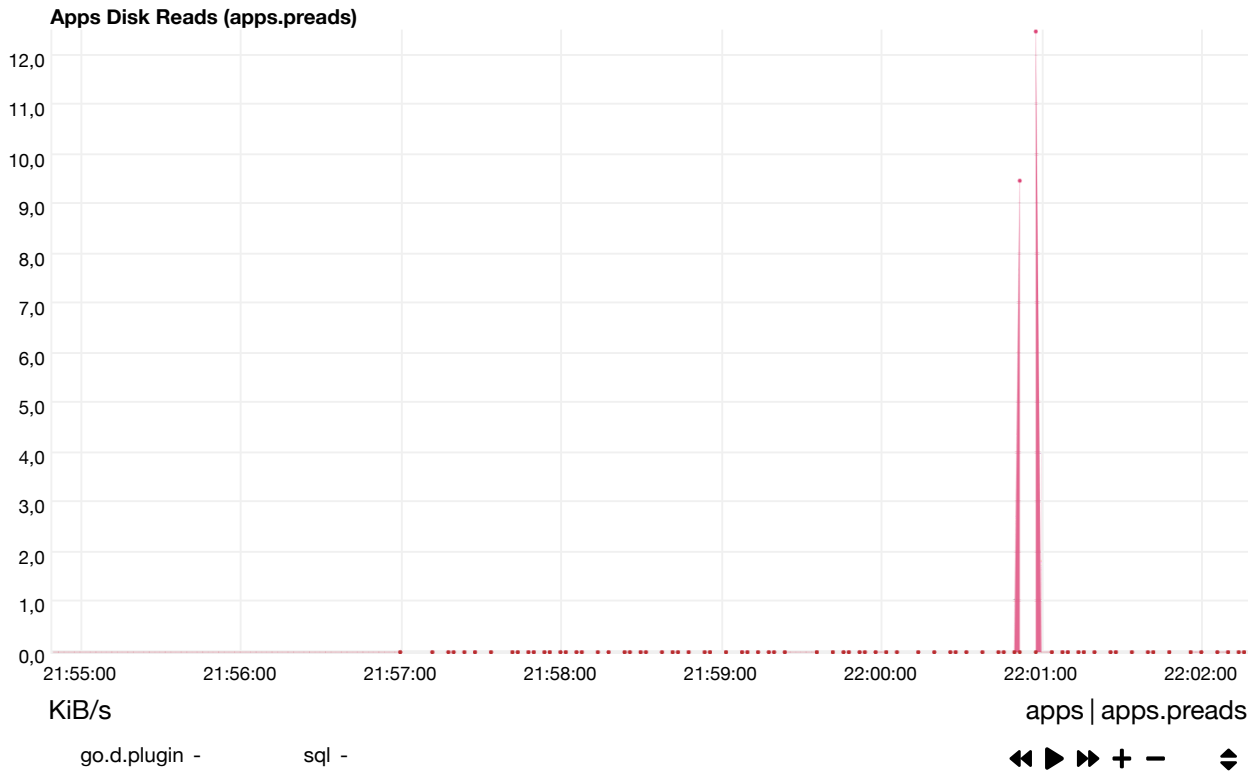
cpu



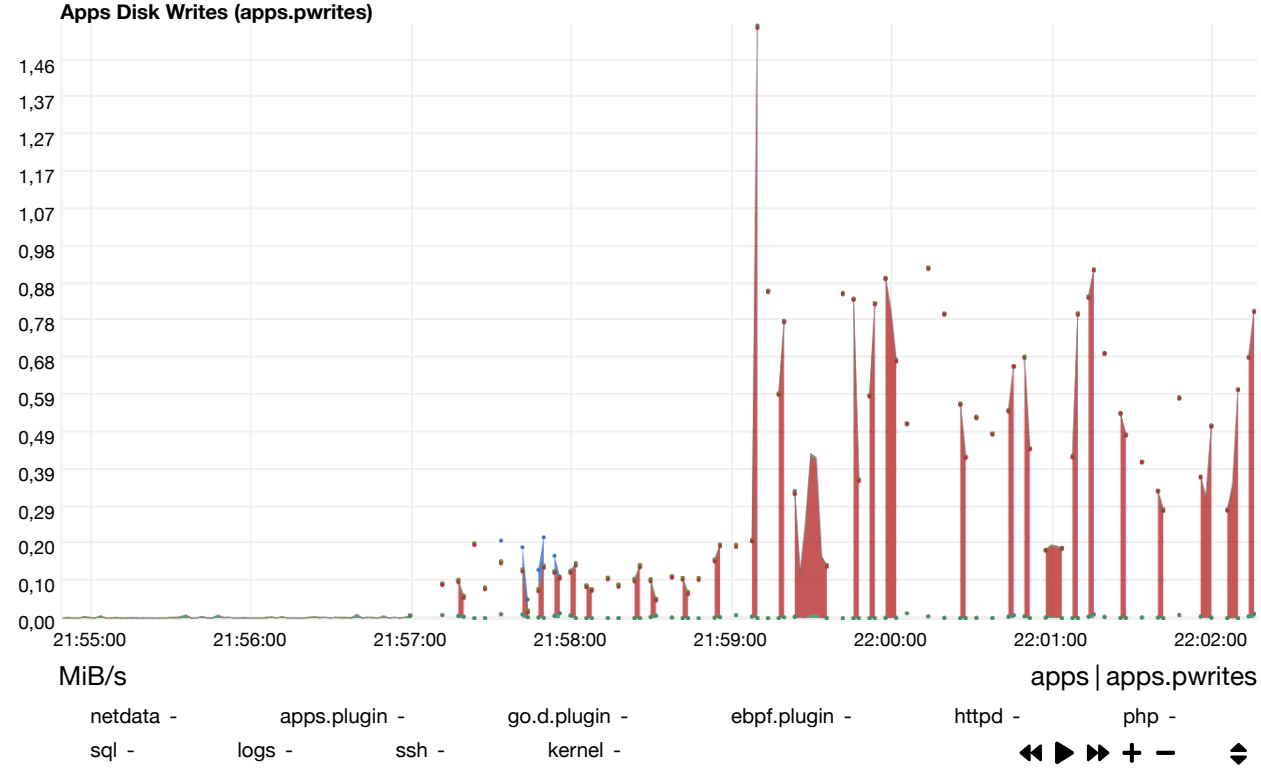



Apps CPU System Time (100% = 1 core) (apps.cpu_system)
7,00
6,00
5,00
4,00
3,00
2,00
1,00
percentage
apps | apps.cpu_system
netdata -
apps.plugin -
tc-qos-helper -
go.d.plugin -
ebpf.plugin -
httpd -
php -
sql -
logs -
ssh -
system -
kernel -
other -

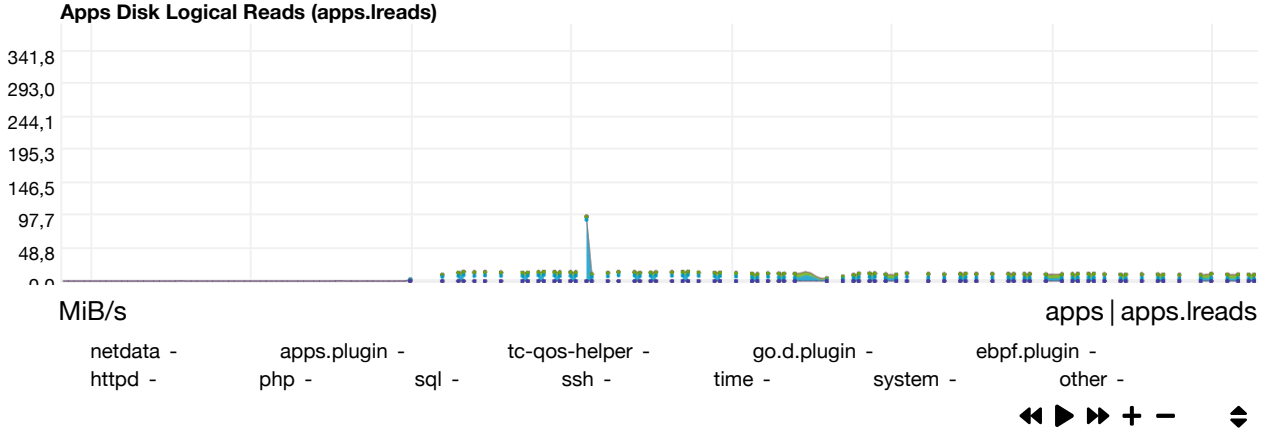
disk


Apps Disk Reads (apps.preads)
12,0
11,0
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
KiB/s
apps | apps.preads
go.d.plugin -
sql -

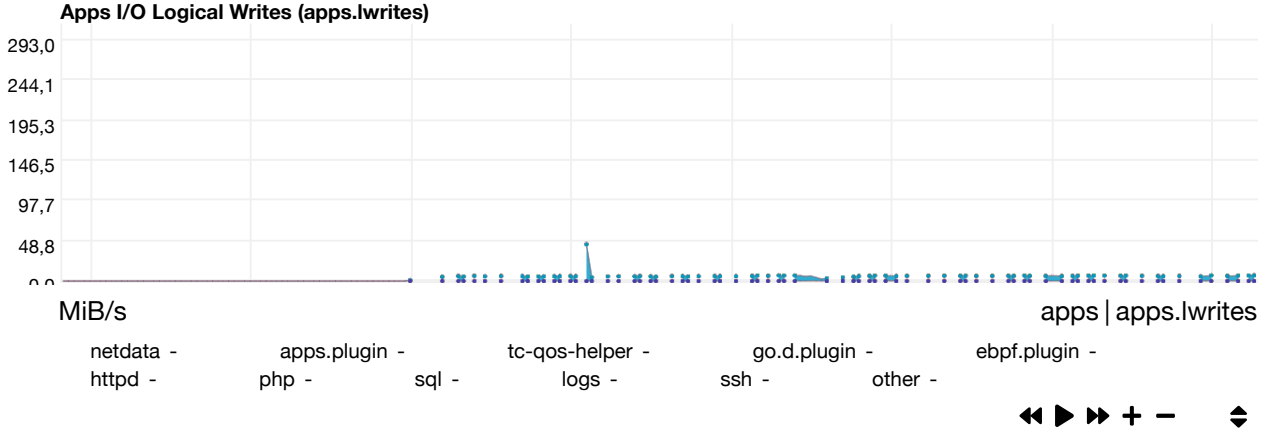
Apps Disk Writes (apps.pwrites)

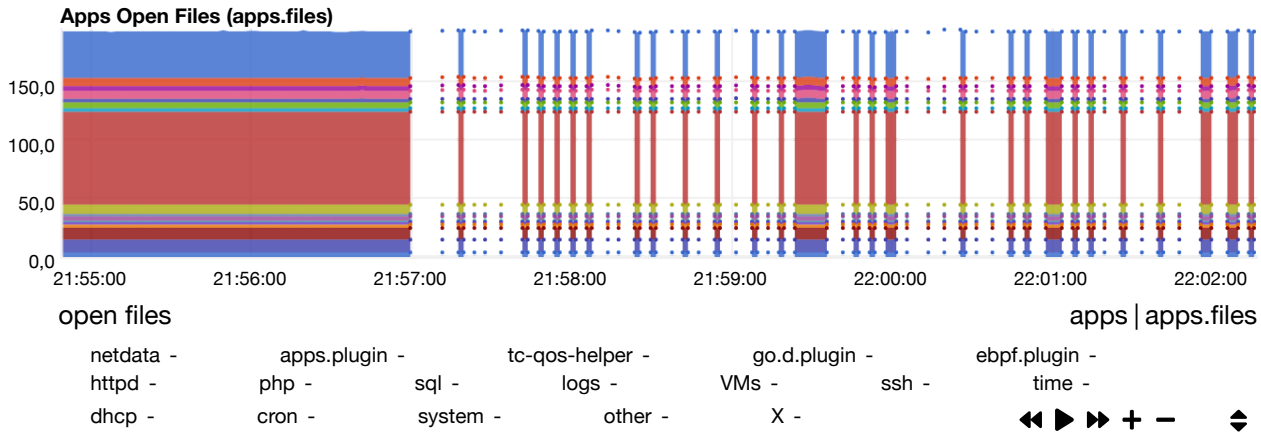


Apps Disk Logical Reads (apps.lreads)



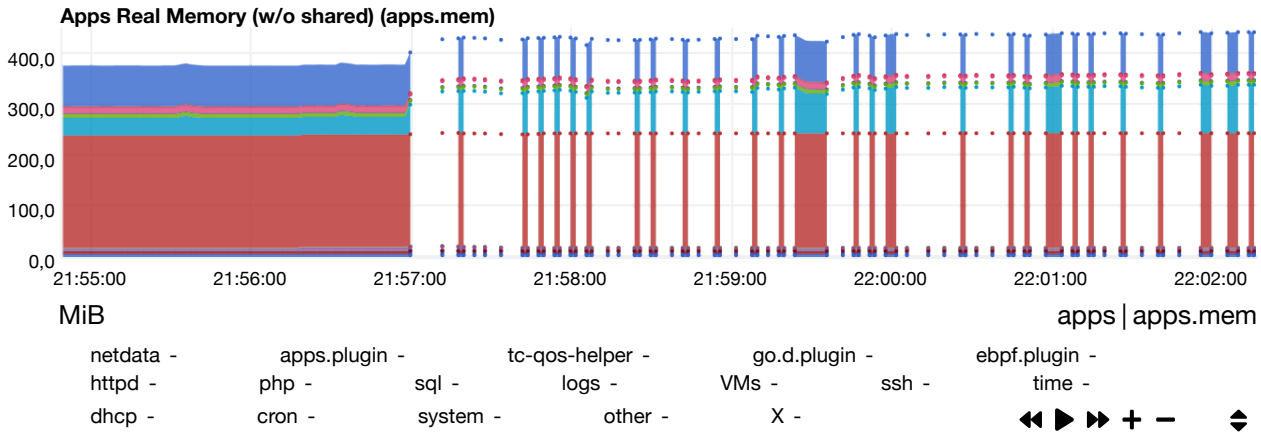
Apps I/O Logical Writes (apps.lwrites)



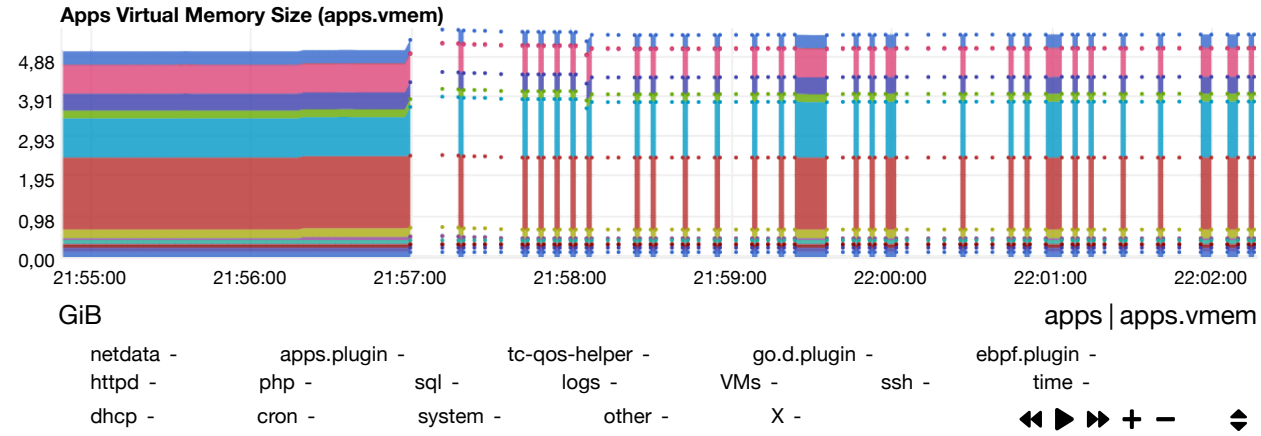


mem

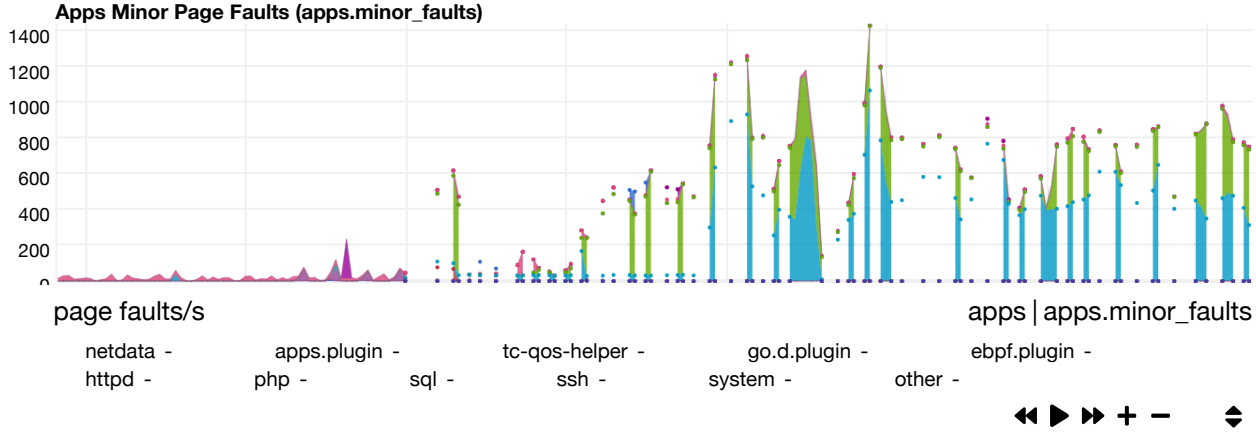
Real memory (RAM) used by applications. This does not include shared memory.



Virtual memory allocated by applications. Please check this article (https://github.com/netdata/netdata/tree/master/daemon#virtual-memory) for more information.

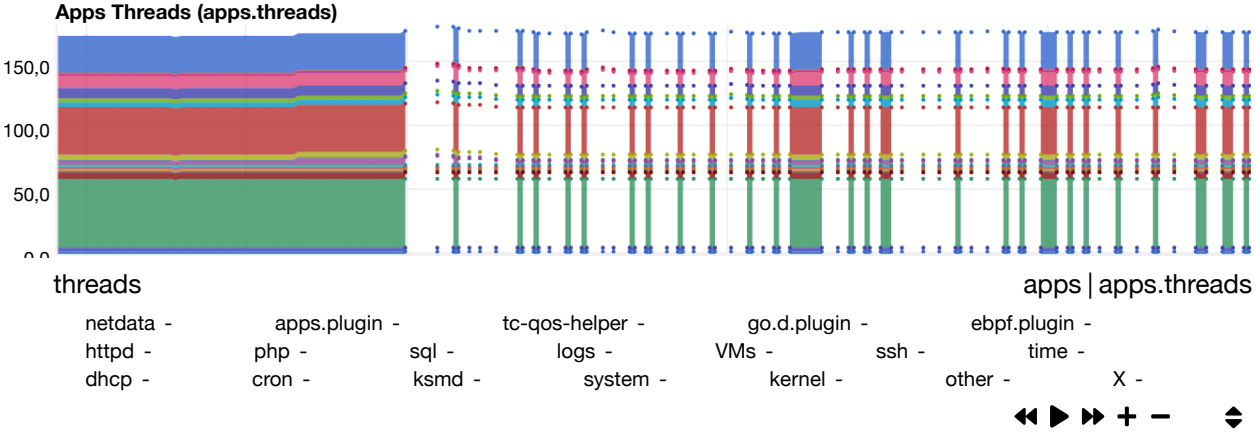


Apps Minor Page Faults (apps.minor_faults)

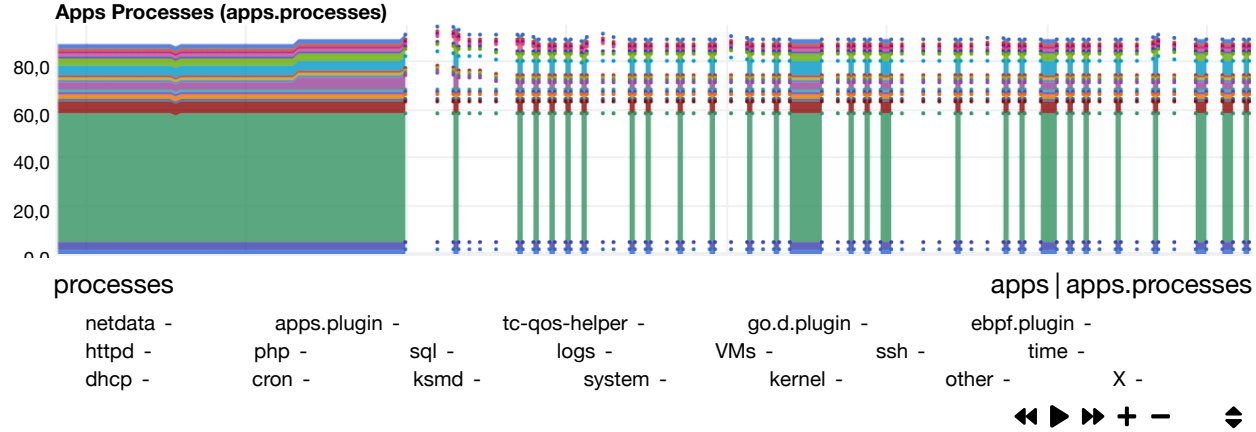


processes

Apps Threads (apps.threads)

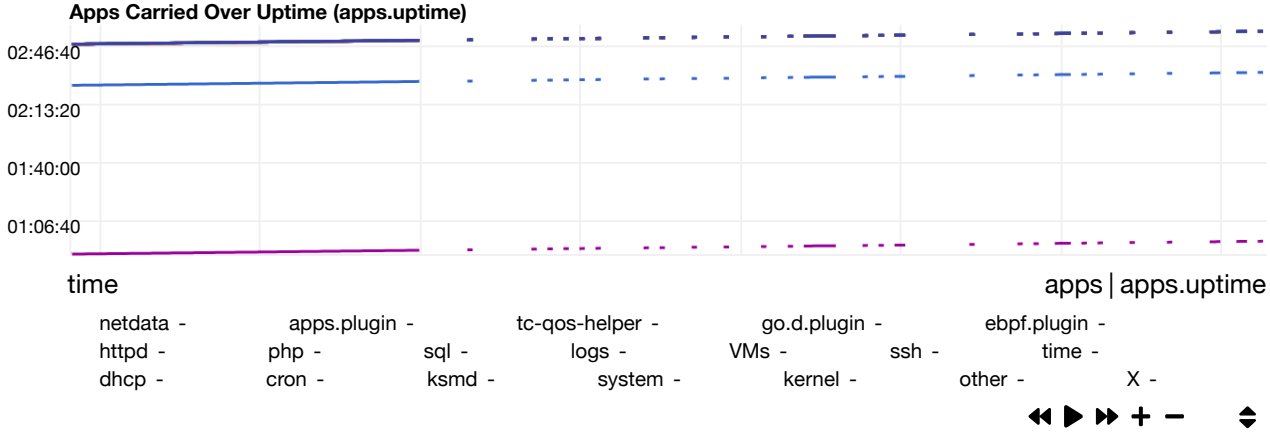


Apps Processes (apps.processes)

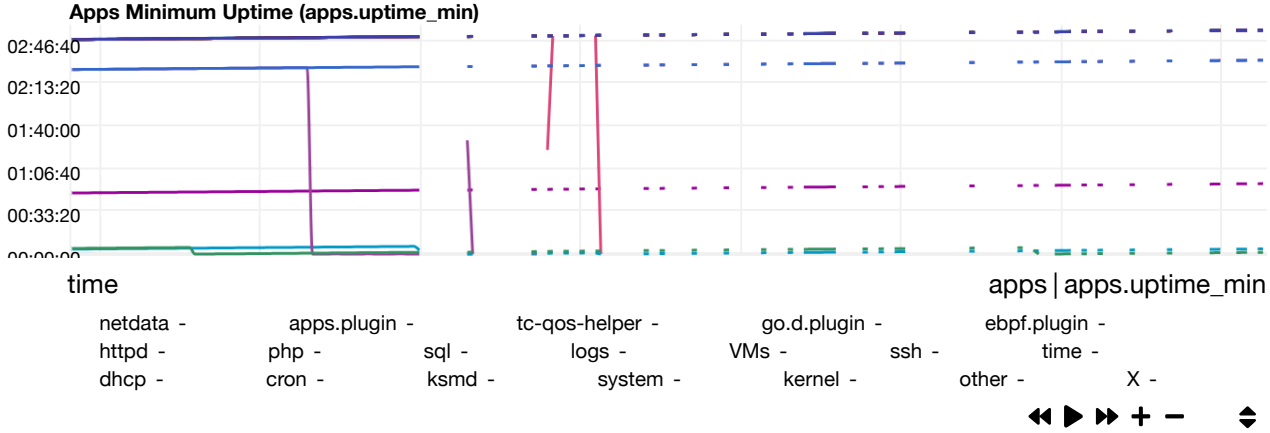


Carried over process group uptime since the Netdata restart. The period of time within which at least one process in the group was running.

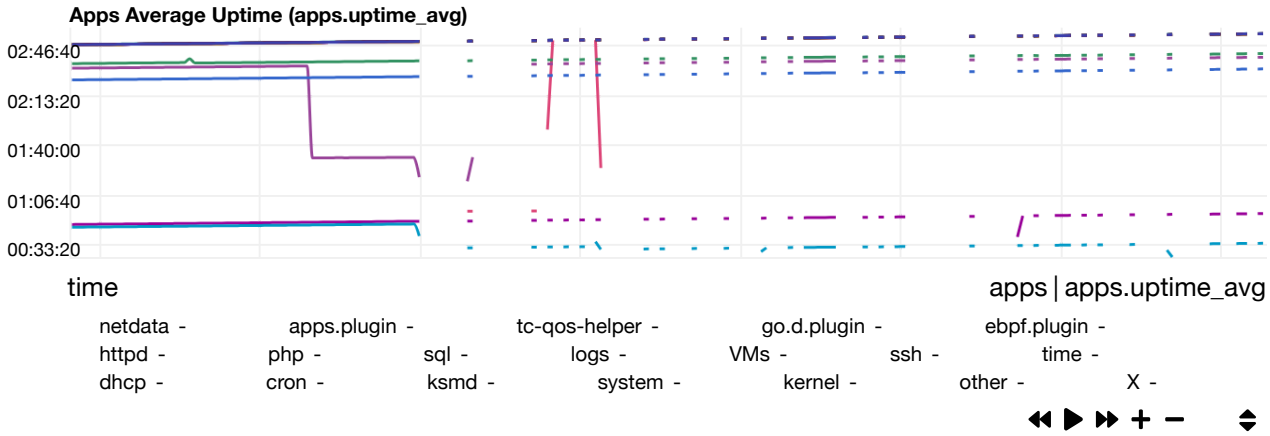
Apps Carried Over Uptime (apps.uptime)



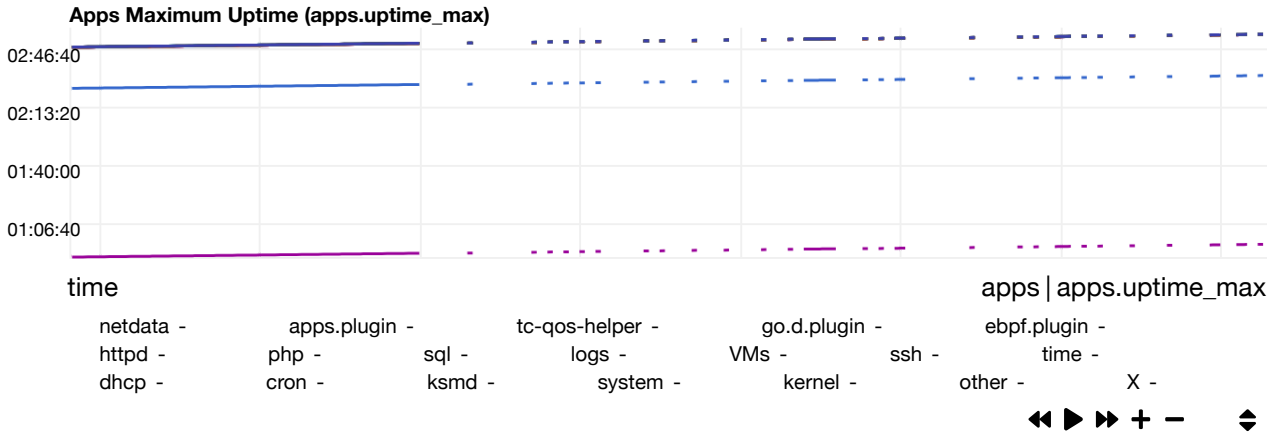
Apps Minimum Uptime (apps.uptime_min)



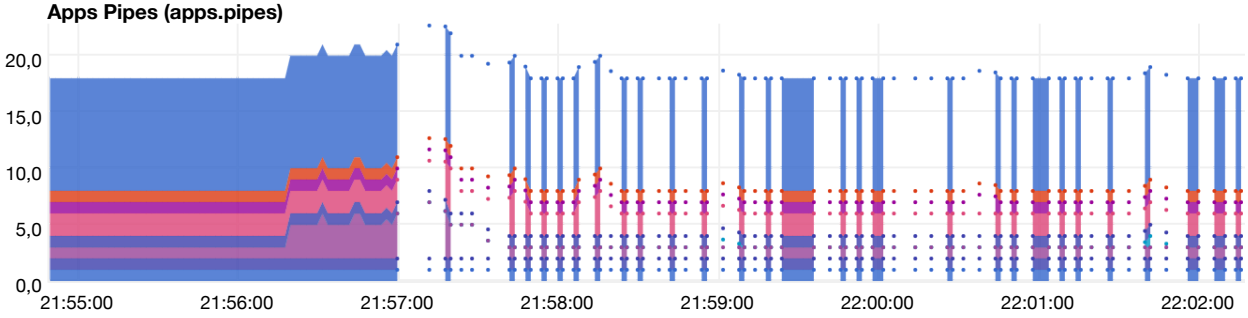
Apps Average Uptime (apps.uptime_avg)



Apps Maximum Uptime (apps.uptime_max)



Apps Pipes (apps.pipes)



open pipes — apps | apps.pipes

netdata - apps.plugin - tc-qos-helper - go.d.plugin - ebpf.plugin -
php - ssh - other - X -

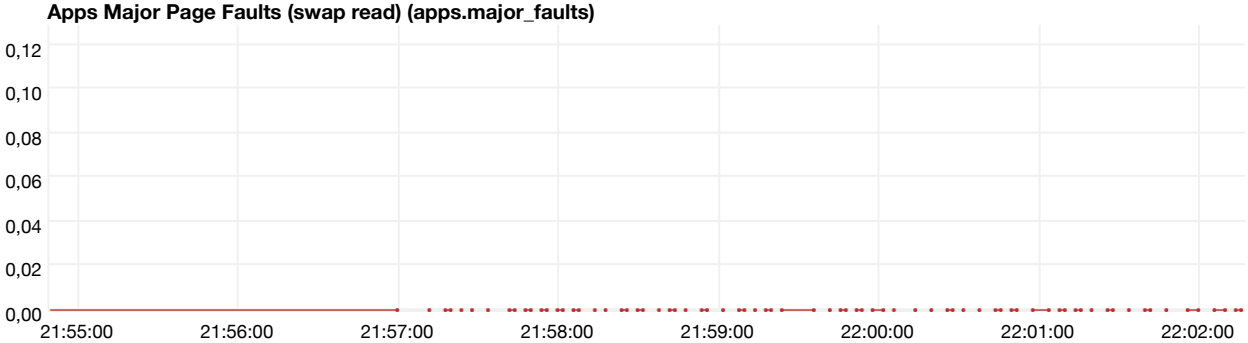
swap

Apps Swap Memory (apps.swap)

MiB — apps | apps.swap

netdata - apps.plugin - charts.d.plugin - python.d.plugin - tc-qos-helper -
go.d.plugin - ebpf.plugin - httpd - php - sql - logs -
VMs - ssh - time - dhcp - cron - ksmd - system -
kernel - other - X -

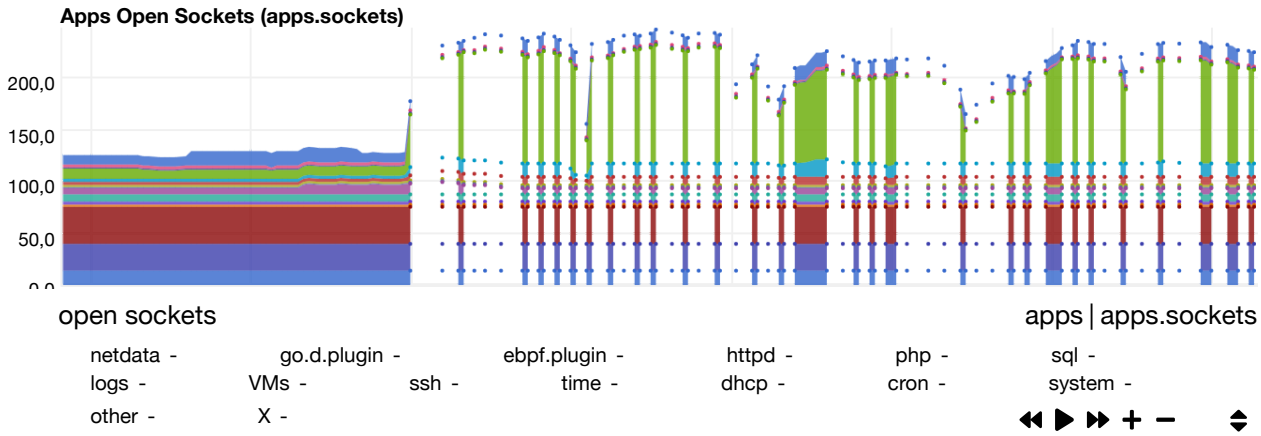
Apps Major Page Faults (swap read) (apps.major_faults)



page faults/s — apps | apps.major_faults

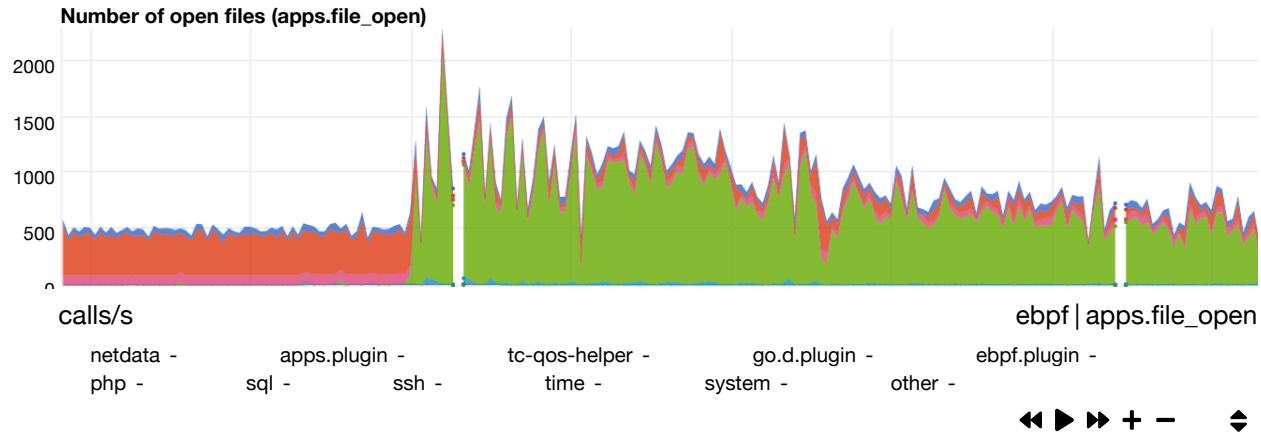
sql -

net

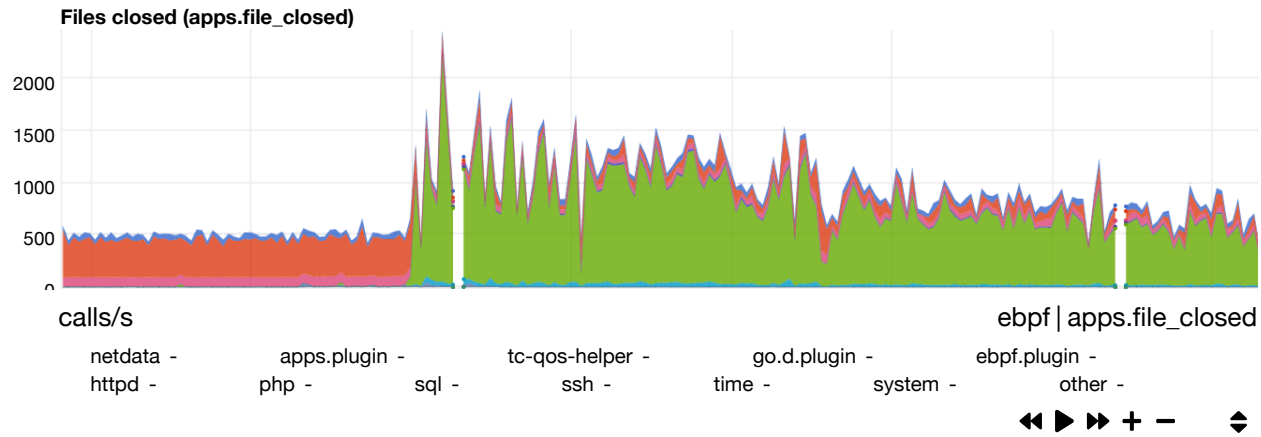


file (eBPF)

Calls to the internal function `do_sys_open` (For kernels newer than `5.5.19` we add a kprobe to `do_sys_openat2` .), which is the common function called from open(2) (https://www.man7.org/linux/man-pages/man2/open.2.html) and openat(2) (https://www.man7.org/linux/man-pages/man2/openat.2.html).



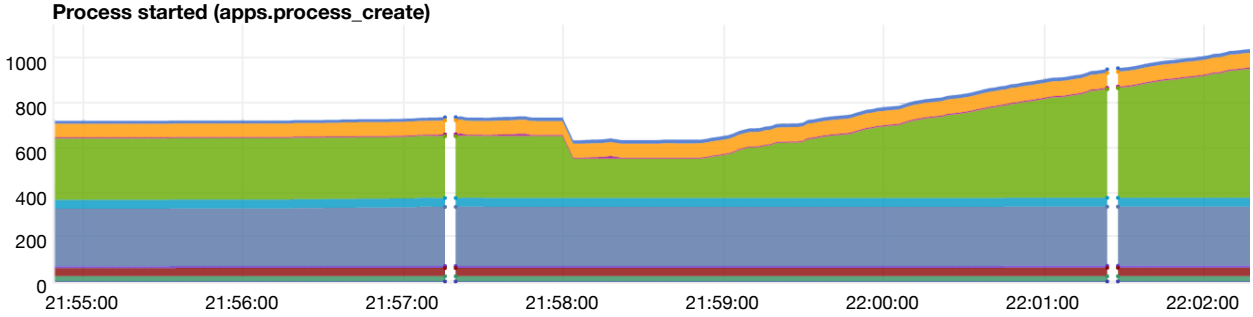
Calls to the internal function __close_fd (https://elixir.bootlin.com/linux/v5.10/source/fs/file.c#L665) or close_fd (https://elixir.bootlin.com/linux/v5.11/source/fs/file.c#L617) according to your kernel version, which is called from close(2) (https://www.man7.org/linux/man-pages/man2/close.2.html).



process (eBPF)

Calls to either do_fork (https://www.ece.uic.edu/~yshi1/linux/lkse/node4.html#SECTION00421000000000000000), or `kernel_clone` if you are running kernel newer than 5.9.16, to create a new task, which is the common name used to define process and tasks inside the kernel. Netdata identifies the process by counting the number of calls to sys_clone (https://linux.die.net/man/2/clone) that do not have the flag `CLONE_THREAD` set.

Process started (apps.process_create)

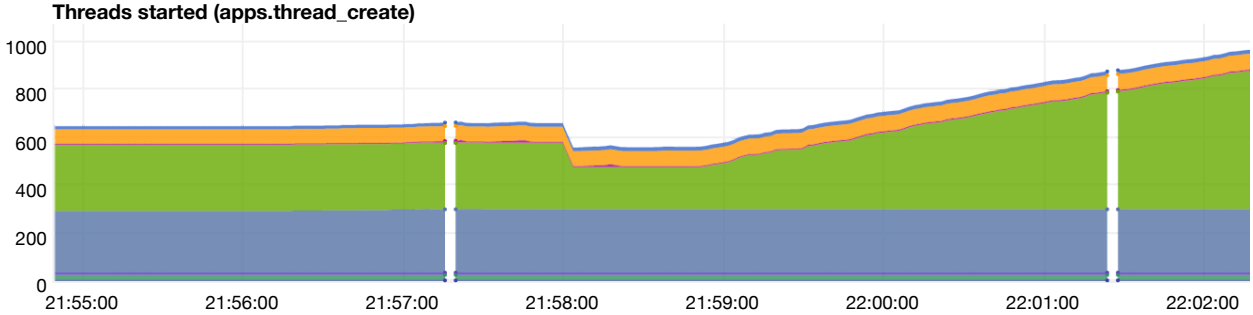


calls/s ebpf | apps.process_create

netdata - tc-qos-helper - go.d.plugin - ebpf.plugin - php - sql -
ssh - cron - kernel - other - X -

Calls to either do_fork (https://www.ece.uic.edu/~yshi1/linux/lkse/node4.html#SECTION00421000000000000000), or `kernel_clone` if you are running kernel newer than 5.9.16, to create a new task, which is the common name used to define process and tasks inside the kernel. Netdata identifies the threads by counting the number of calls to sys_clone (https://linux.die.net/man/2/clone) that have the flag `CLONE_THREAD` set.

Threads started (apps.thread_create)

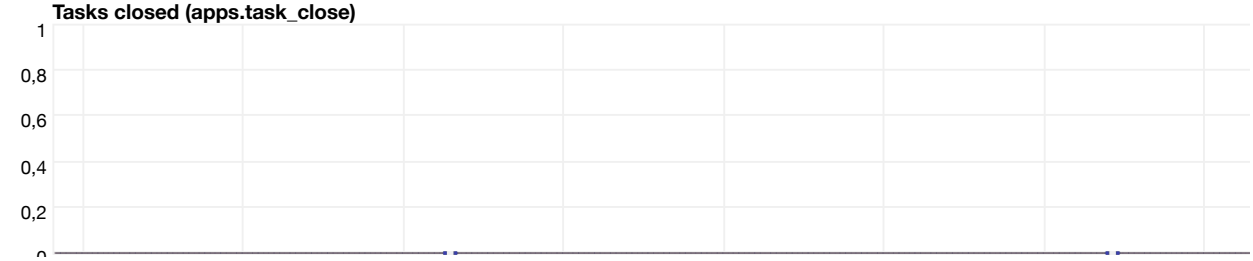


calls/s ebpf | apps.thread_create

netdata - tc-qos-helper - go.d.plugin - ebpf.plugin - php - ssh -
cron - other - X -

Calls to the functions responsible for closing (do_exit (https://www.informit.com/articles/article.aspx?p=370047&seqNum=4)) and releasing (release_task (https://www.informit.com/articles/article.aspx?p=370047&seqNum=4)) tasks.

Tasks closed (apps.task_close)

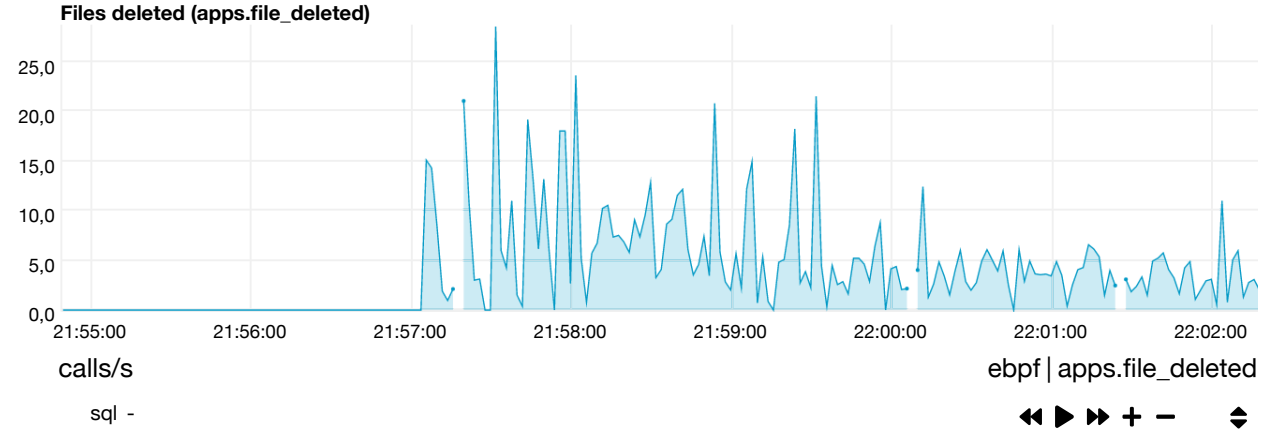


calls/s ebpf | apps.task_close

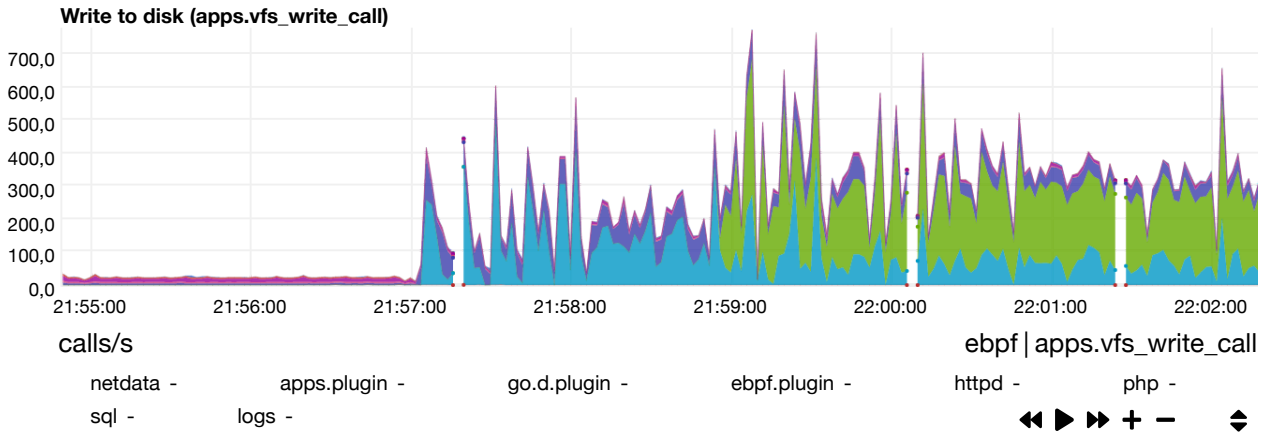
netdata - apps.plugin - python.d.plugin - tc-qos-helper - go.d.plugin -
ebpf.plugin - httpd - php - sql - logs - VMs - ssh -
time - dhcp - cron - ksmd - system - kernel - other -
X -

vfs (eBPF)

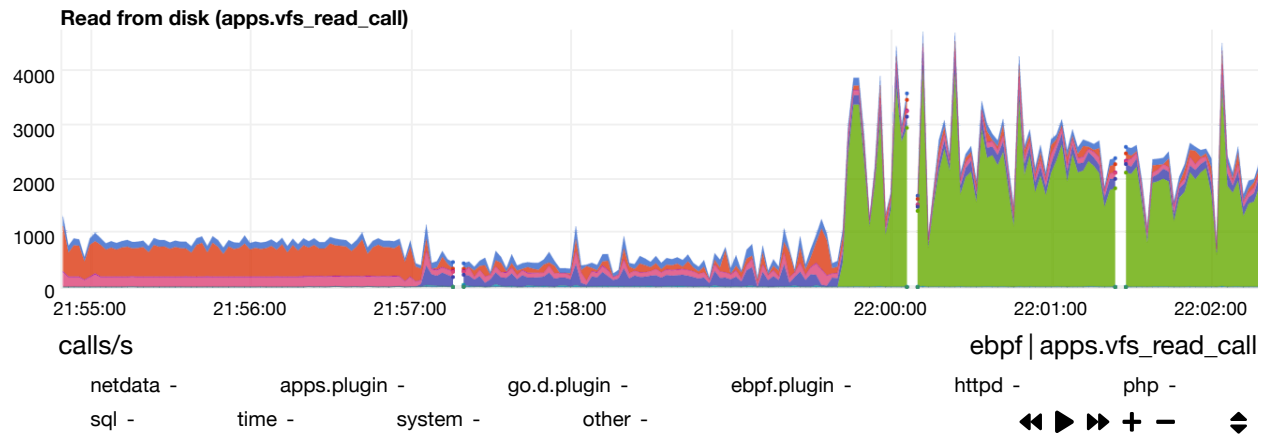
Calls to the function vfs_unlink (https://www.kernel.org/doc/htmldocs/filesystems/API-vfs-unlink.html). This chart does not show all events that remove files from the filesystem, because filesystems can create their own functions to remove files.



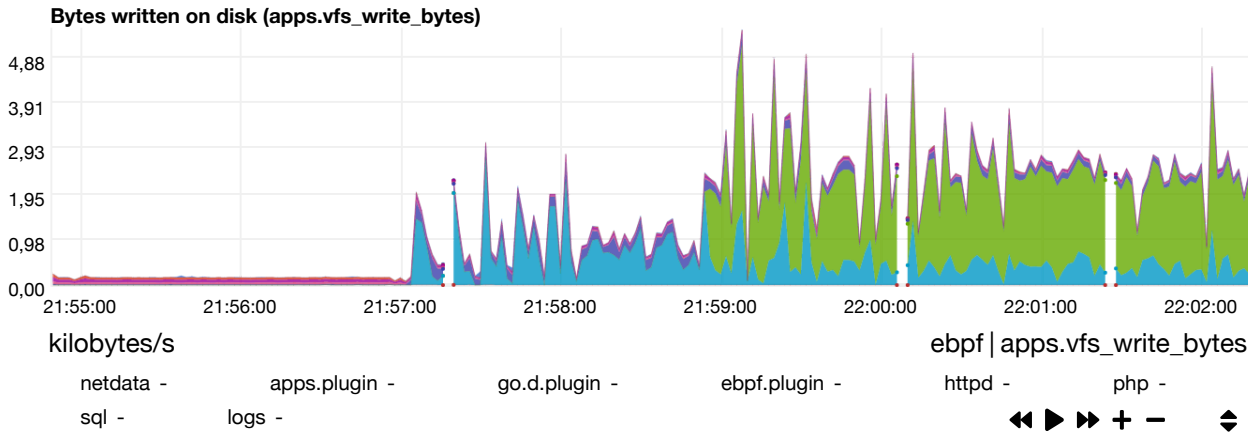
Successful calls to the function vfs_write (https://topic.alibabacloud.com/a/kernel-state-file-operation-__-work-information-kernel_8_8_20287135.html). This chart may not show all filesystem events if it uses other functions to store data on disk.



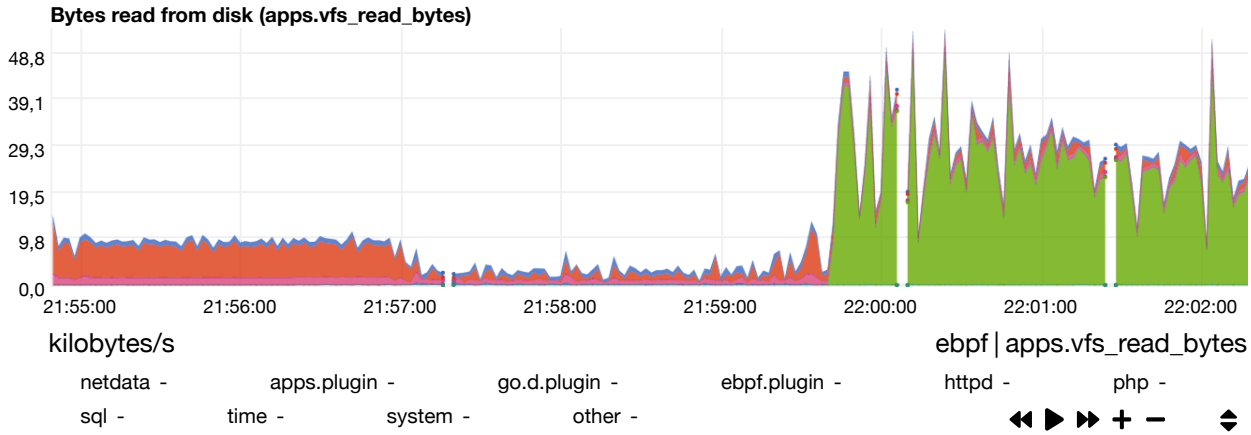
Successful calls to the function vfs_read (https://topic.alibabacloud.com/a/kernel-state-file-operation-__-work-information-kernel_8_8_20287135.html). This chart may not show all filesystem events if it uses other functions to store data on disk.

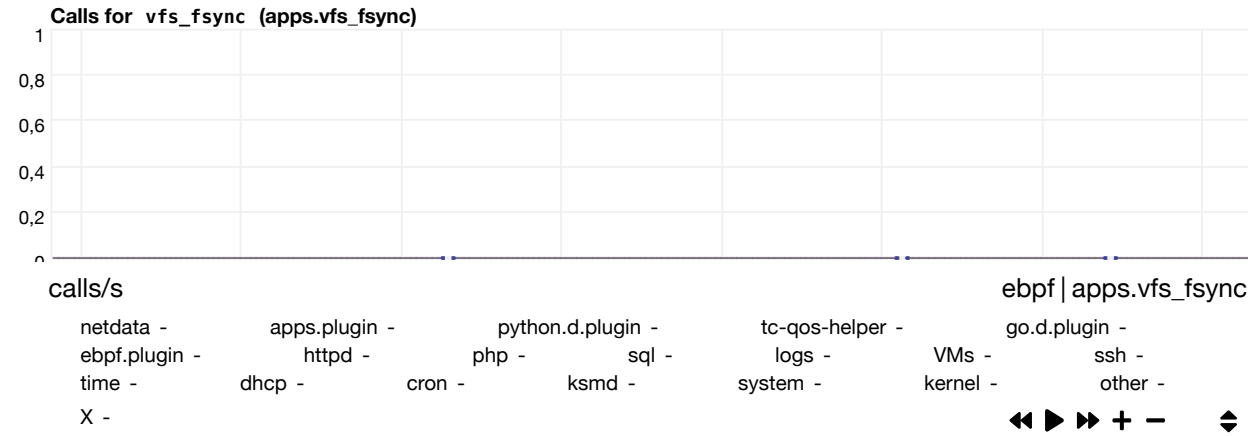


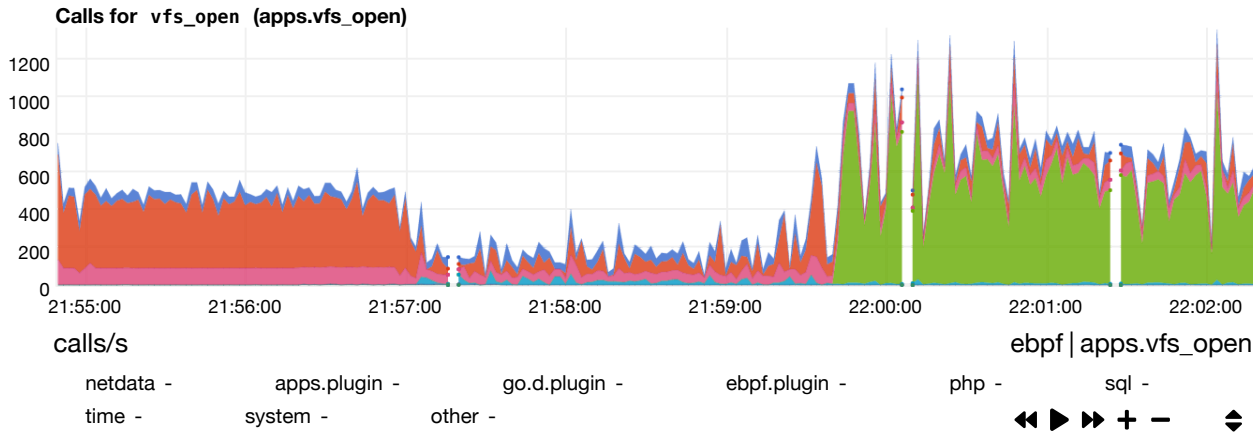
Total of bytes successfully written using the function vfs_write (https://topic.alibabacloud.com/a/kernel-state-file-operation-__-work-information-kernel_8_8_20287135.html).

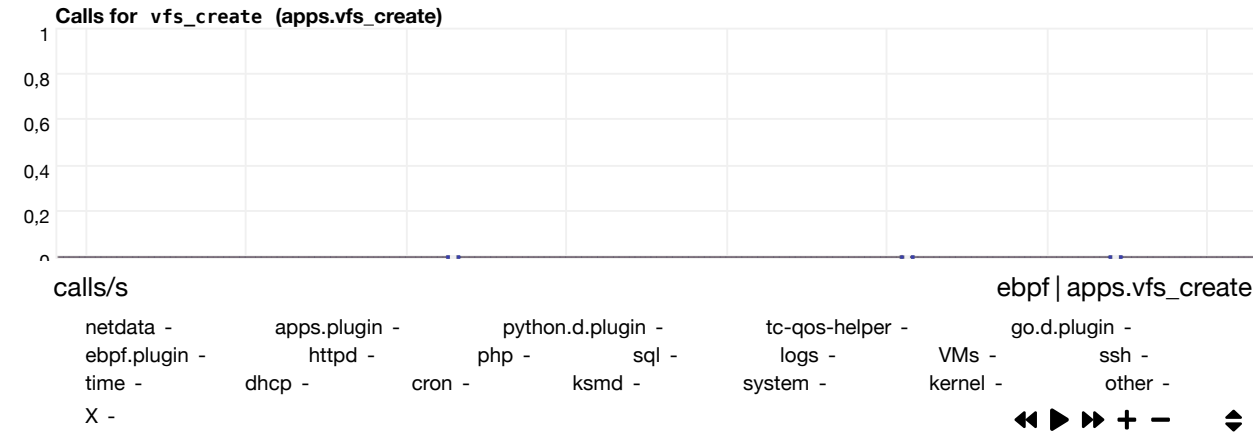


Total of bytes successfully read using the function vfs_read (https://topic.alibabacloud.com/a/kernel-state-file-operation-__-work-information-kernel_8_8_20287135.html).

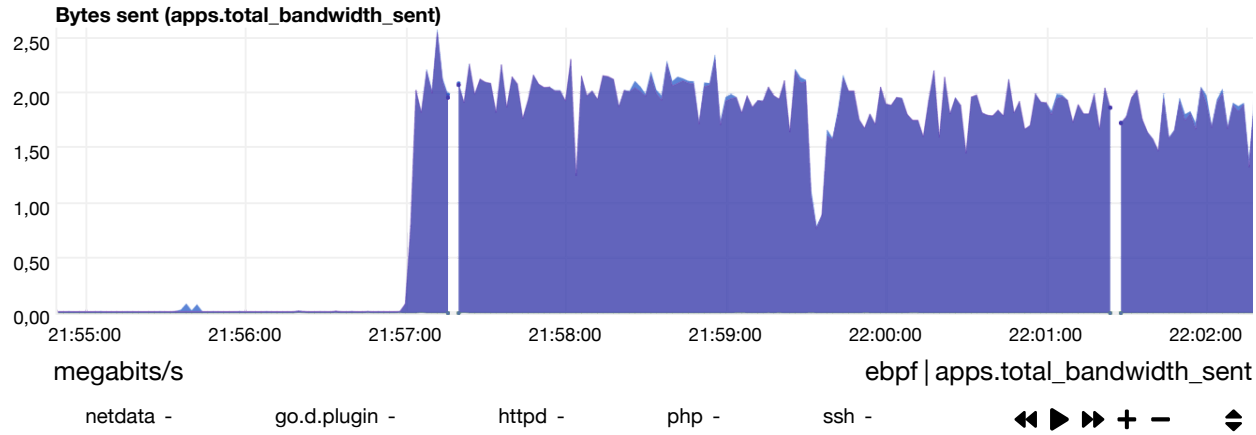


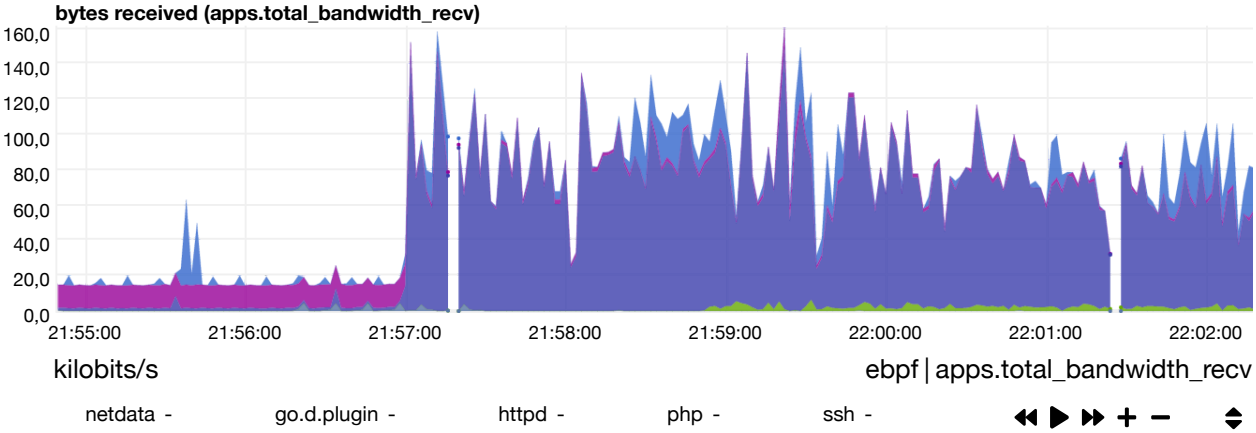



Calls for vfs_open (apps.vfs_open)
1200
1000
800
600
400
200
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
calls/s
ebpf | apps.vfs_open
netdata -
apps.plugin -
go.d.plugin -
ebpf.plugin -
php -
sql -
time -
system -
other -

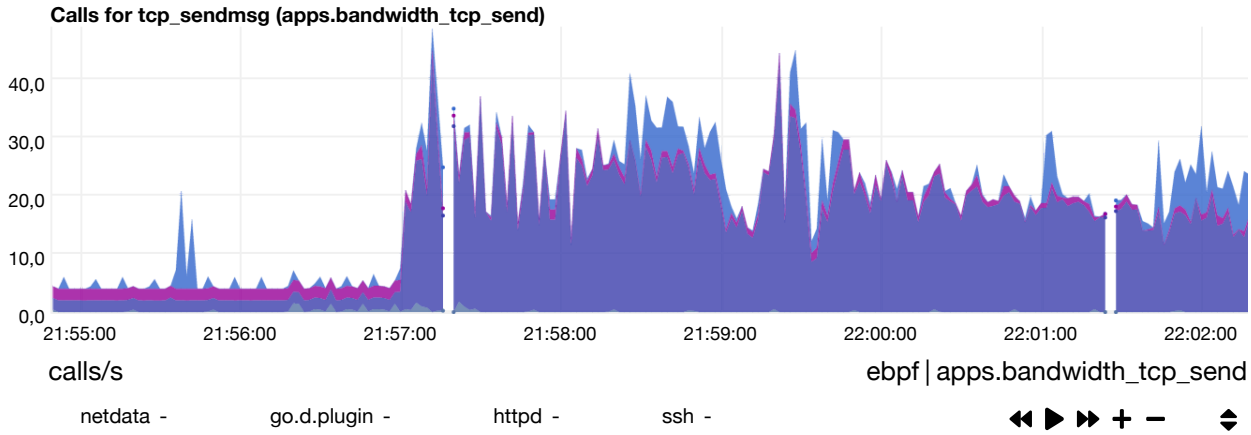

Calls for vfs_create (apps.vfs_create)
1
0,8
0,6
0,4
0,2
0
calls/s
ebpf | apps.vfs_create
netdata -
apps.plugin -
python.d.plugin -
tc-qos-helper -
go.d.plugin -
ebpf.plugin -
httpd -
php -
sql -
logs -
VMs -
ssh -
time -
dhcp -
cron -
ksmd -
system -
kernel -
other -
X -

net (eBPF)

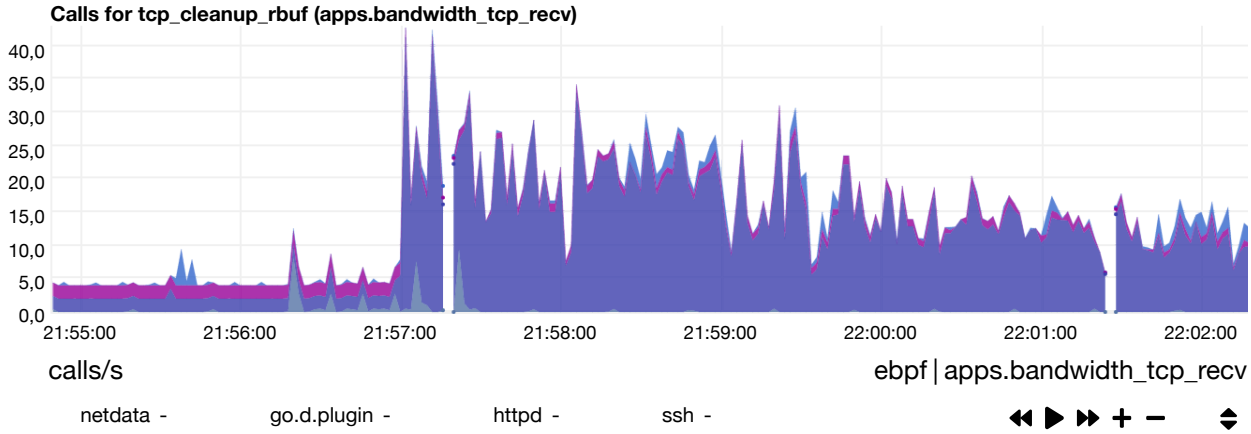

Bytes sent (apps.total_bandwidth_sent)
2,50
2,00
1,50
1,00
0,50
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
megabits/s
ebpf | apps.total_bandwidth_sent
netdata -
go.d.plugin -
httpd -
php -
ssh -


bytes received (apps.total_bandwidth_recv)
160,0
140,0
120,0
100,0
80,0
60,0
40,0
20,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
kilobits/s
ebpf | apps.total_bandwidth_recv
netdata -
go.d.plugin -
httpd -
php -
ssh -

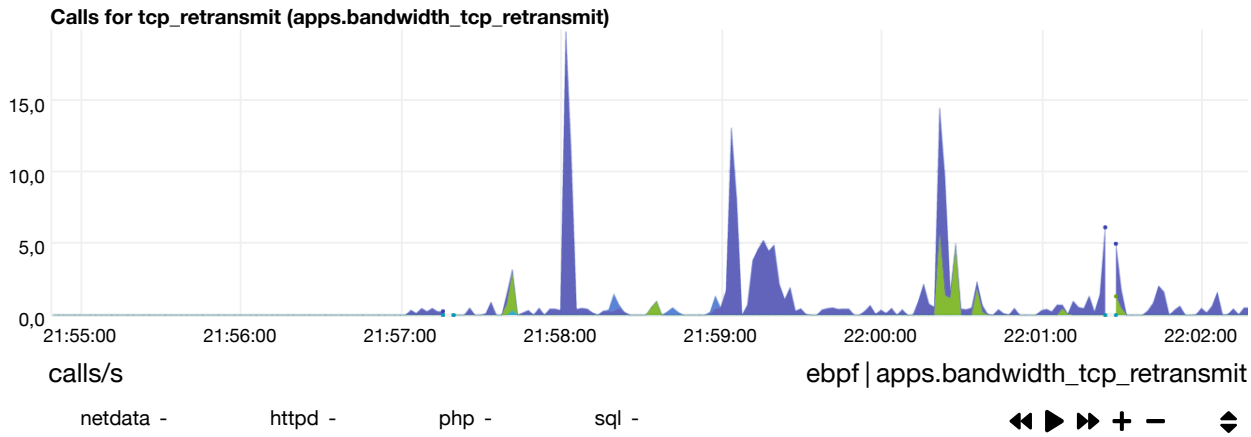
Calls for function `tcp_sendmsg` .



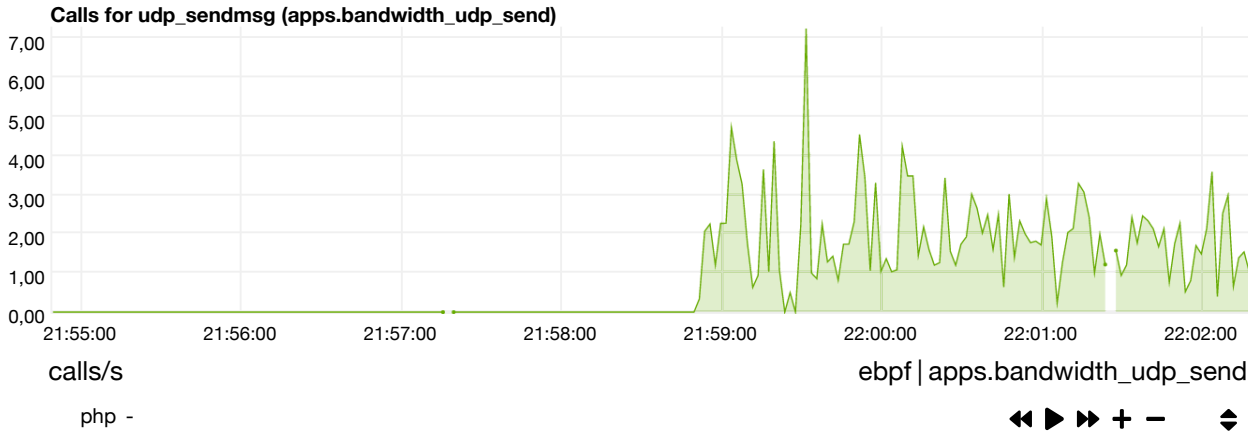
Calls for functions `tcp_cleanup_rbuf` . We use `tcp_cleanup_rbuf` instead `tcp_recvmsg` , because this last misses `tcp_read_sock()` traffic and we would also need to have more probes to get the socket and package size.



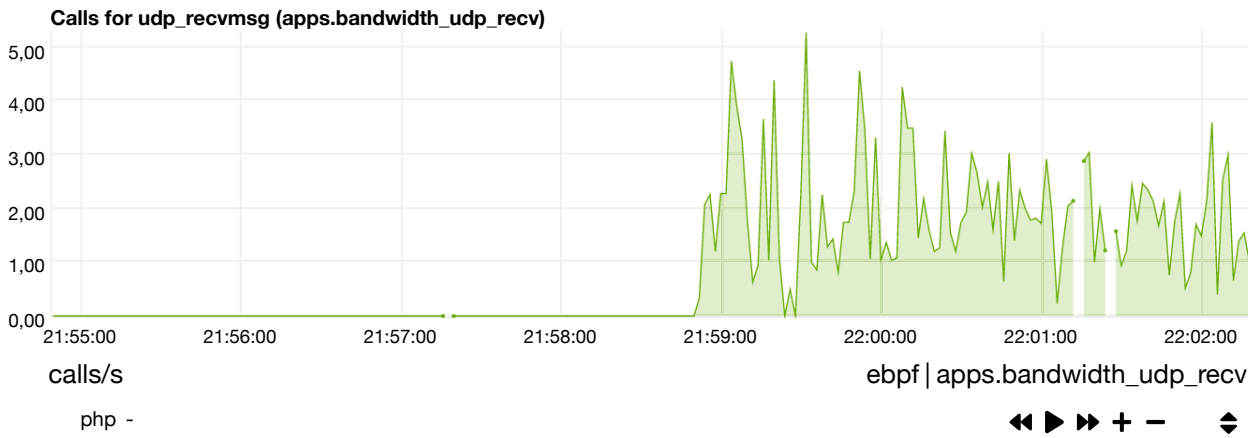
Calls for functions `tcp_retransmit_skb` .



Calls for function `udp_sendmsg` .



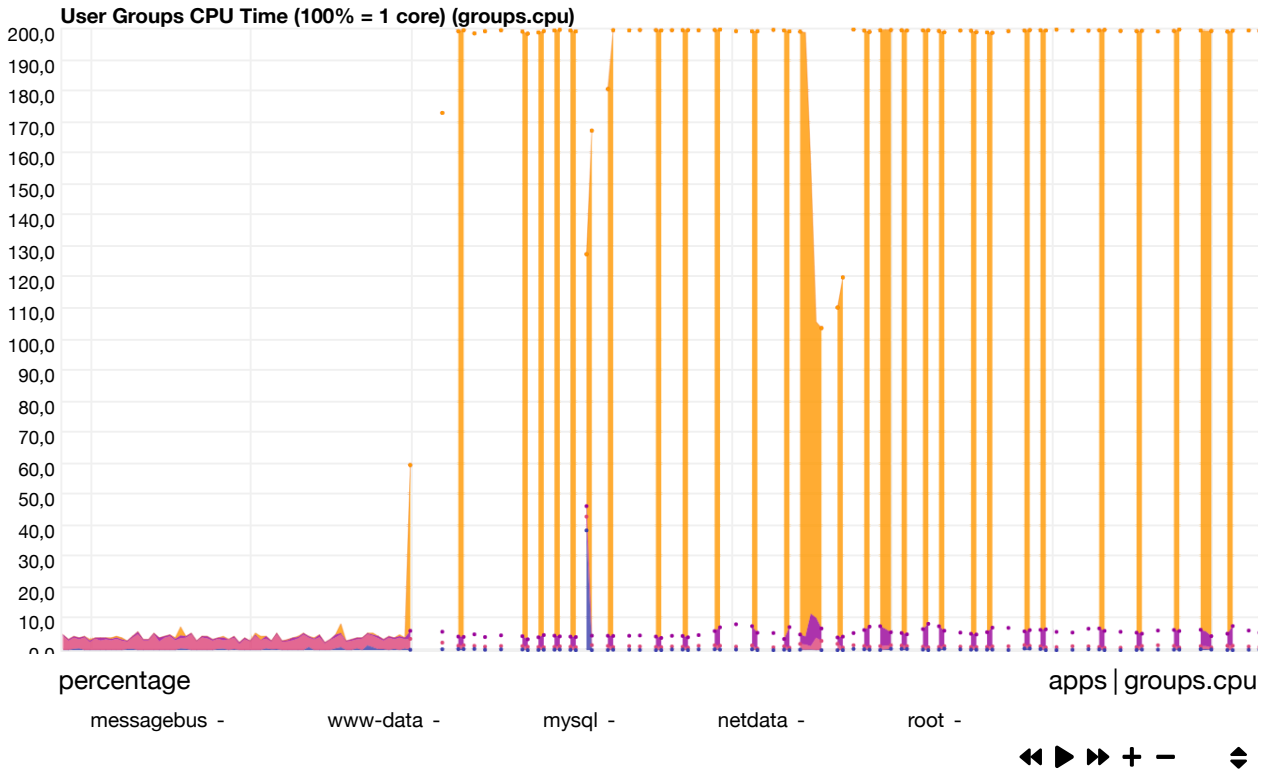
Calls for function `udp_recvmsg` .

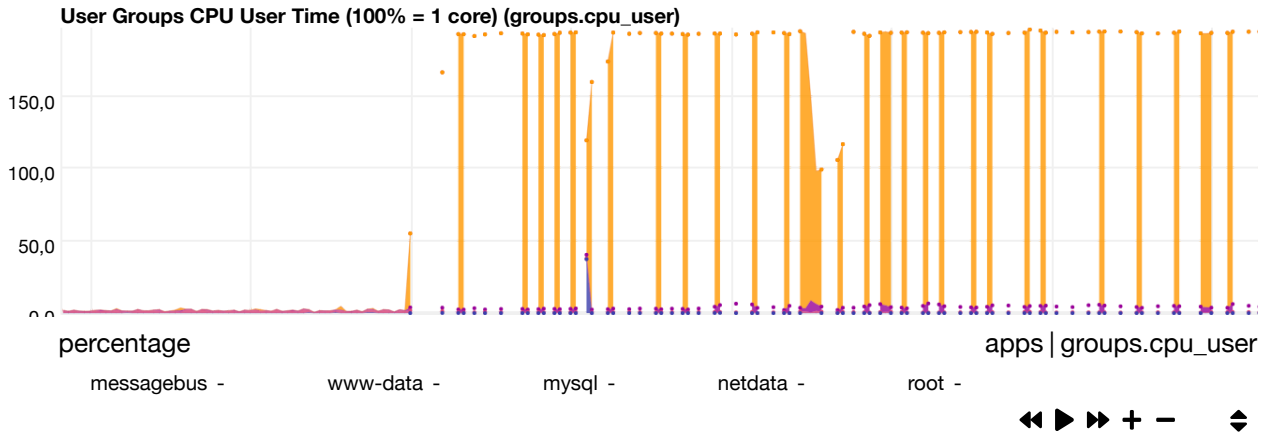


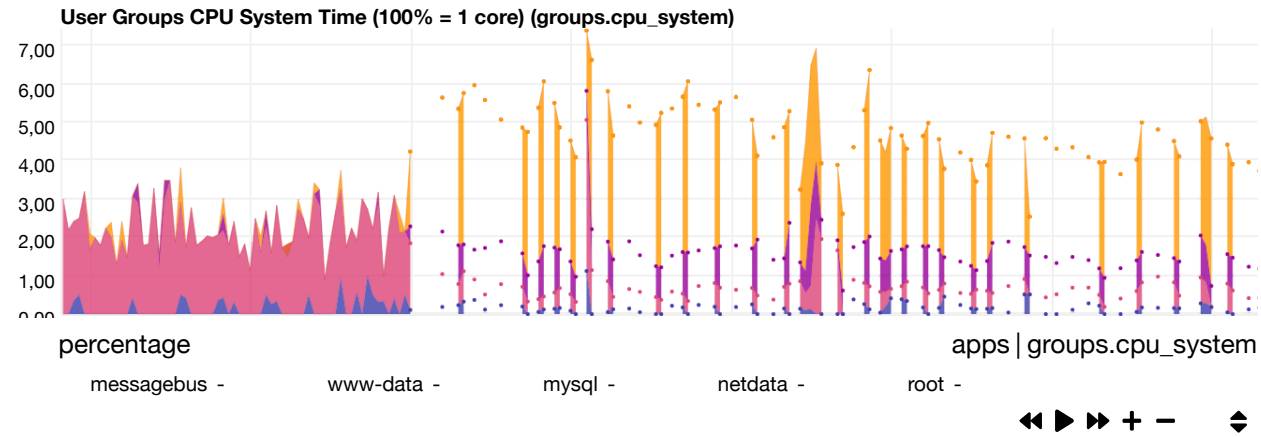
User Groups

Per user group statistics are collected using netdata's `apps.plugin` . This plugin walks through all processes and aggregates statistics per user group. The reported values are compatible with `top` , although the netdata plugin counts also the resources of exited children (unlike `top` which shows only the resources of the currently running processes). So for processes like shell scripts, the reported values include the resources used by the commands these scripts run within each timeframe.

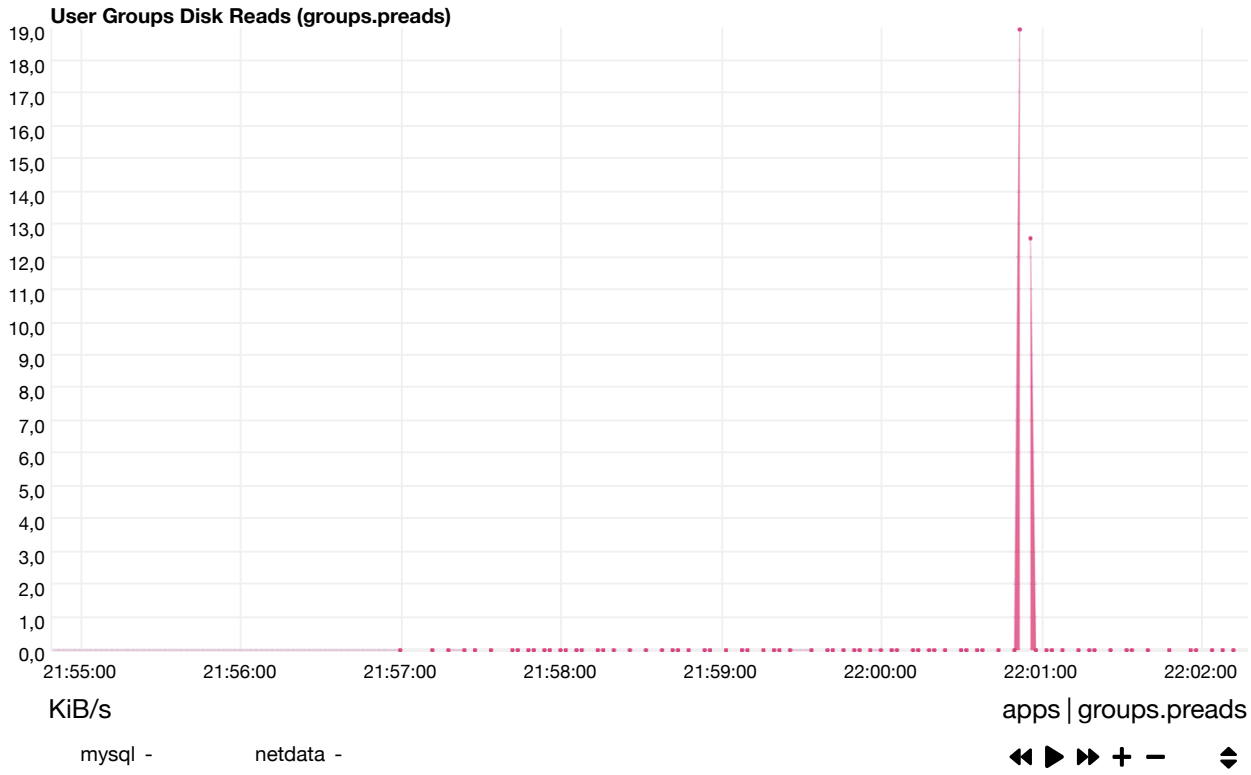
cpu

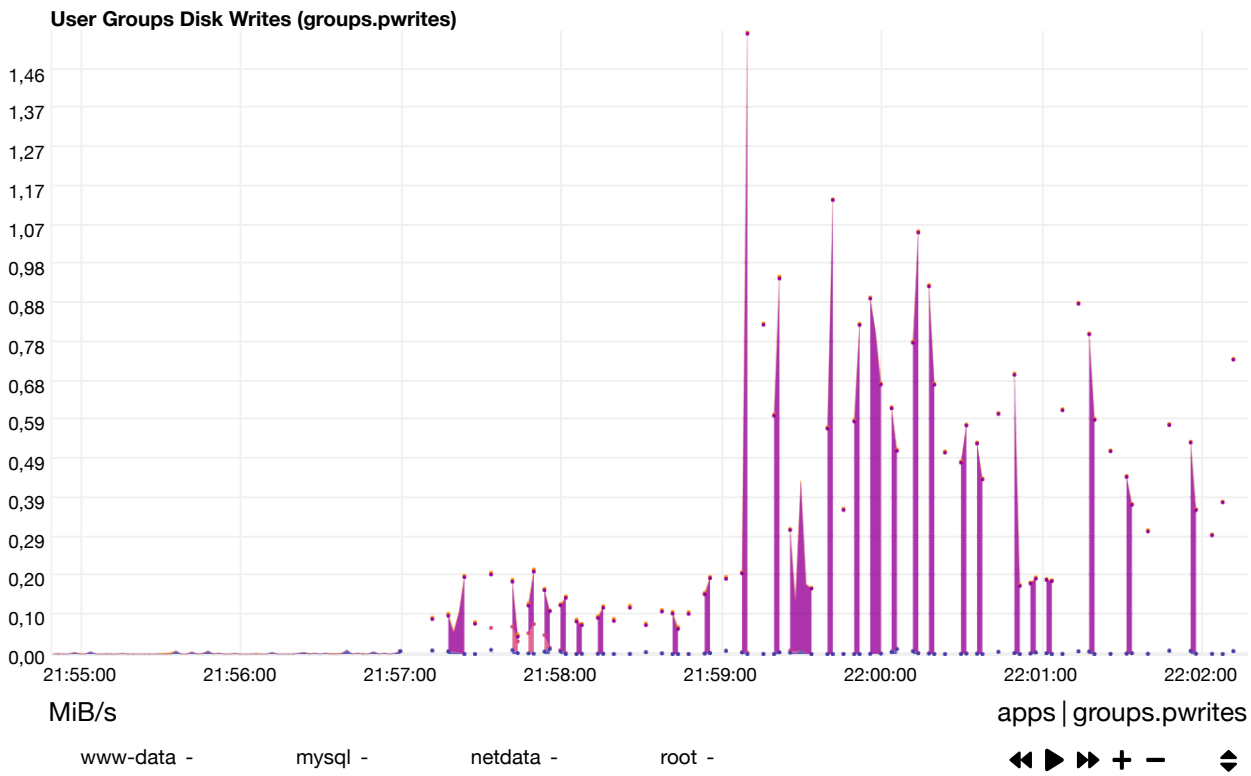

User Groups CPU Time (100% = 1 core) (groups.cpu)
200,0
190,0
180,0
170,0
160,0
150,0
140,0
130,0
120,0
110,0
100,0
90,0
80,0
70,0
60,0
50,0
40,0
30,0
20,0
10,0
percentage
apps | groups.cpu
messagebus -
www-data -
mysql -
netdata -
root -

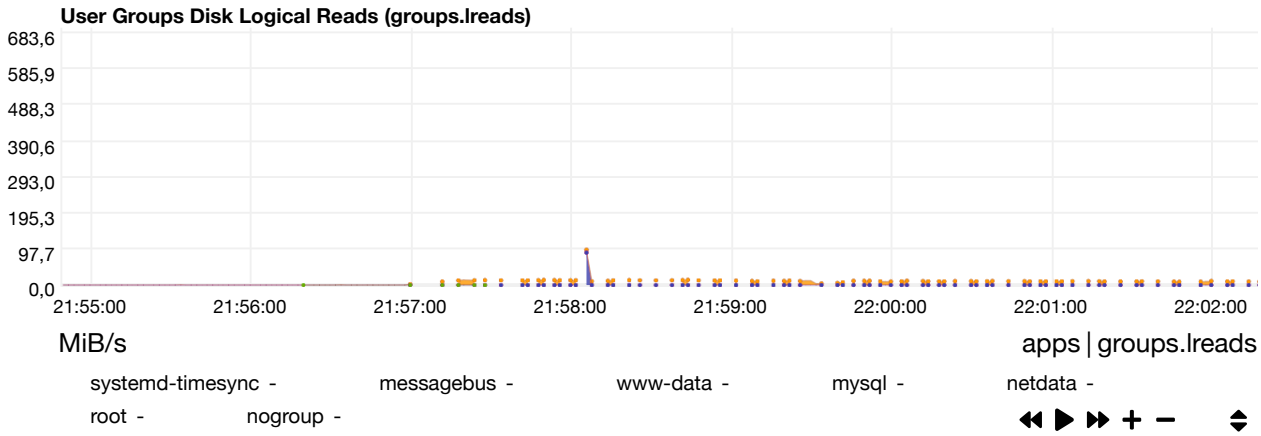

User Groups CPU User Time (100% = 1 core) (groups.cpu_user)
150,0
100,0
50,0
percentage
apps | groups.cpu_user
messagebus -
www-data -
mysql -
netdata -
root -

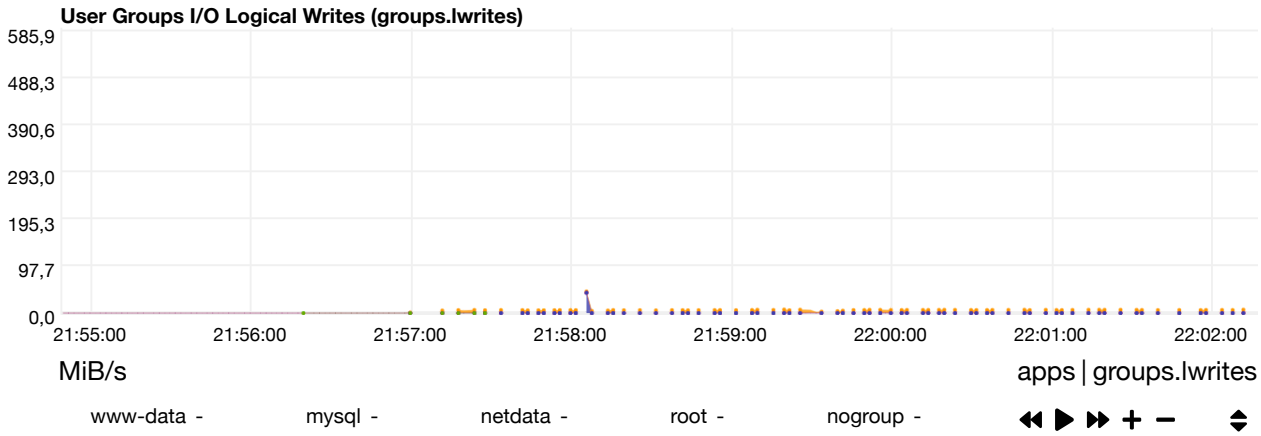

User Groups CPU System Time (100% = 1 core) (groups.cpu_system)
7,00
6,00
5,00
4,00
3,00
2,00
1,00
percentage
apps | groups.cpu_system
messagebus -
www-data -
mysql -
netdata -
root -

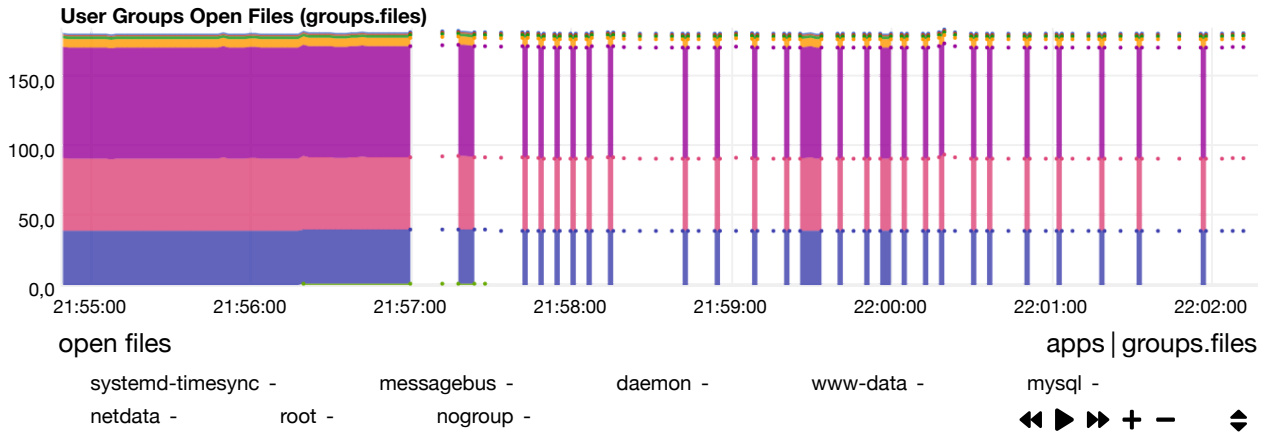
disk


User Groups Disk Reads (groups.preads)
19,0
18,0
17,0
16,0
15,0
14,0
13,0
12,0
11,0
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
KiB/s
apps | groups.preads
mysql -
netdata -


User Groups Disk Writes (groups.pwrites)
1,46
1,37
1,27
1,17
1,07
0,98
0,88
0,78
0,68
0,59
0,49
0,39
0,29
0,20
0,10
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | groups.pwrites
www-data -
mysql -
netdata -
root -

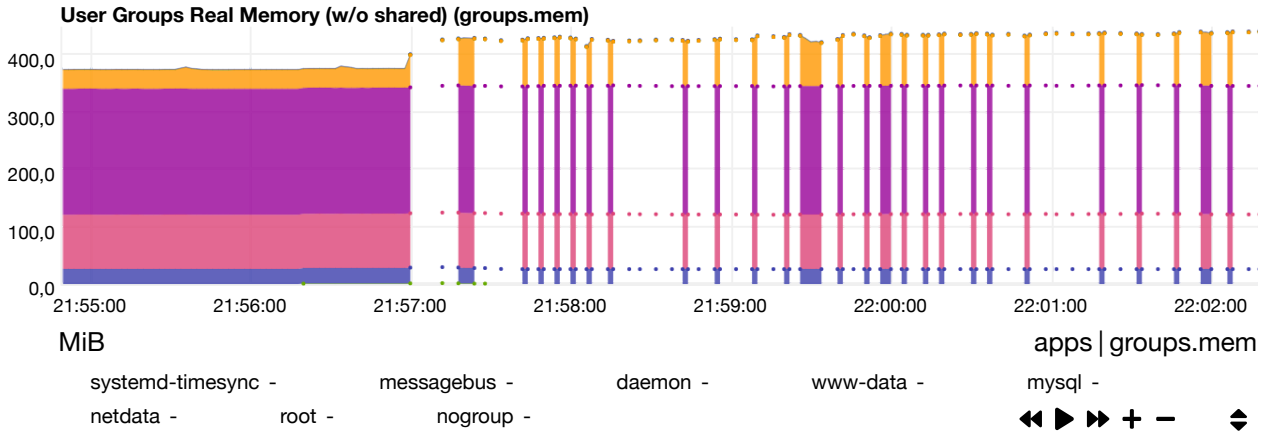

User Groups Disk Logical Reads (groups.lreads)
683,6
585,9
488,3
390,6
293,0
195,3
97,7
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | groups.lreads
systemd-timesync -
messagebus -
www-data -
mysql -
netdata -
root -
nogroup -


User Groups I/O Logical Writes (groups.lwrites)
585,9
488,3
390,6
293,0
195,3
97,7
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | groups.lwrites
www-data -
mysql -
netdata -
root -
nogroup -

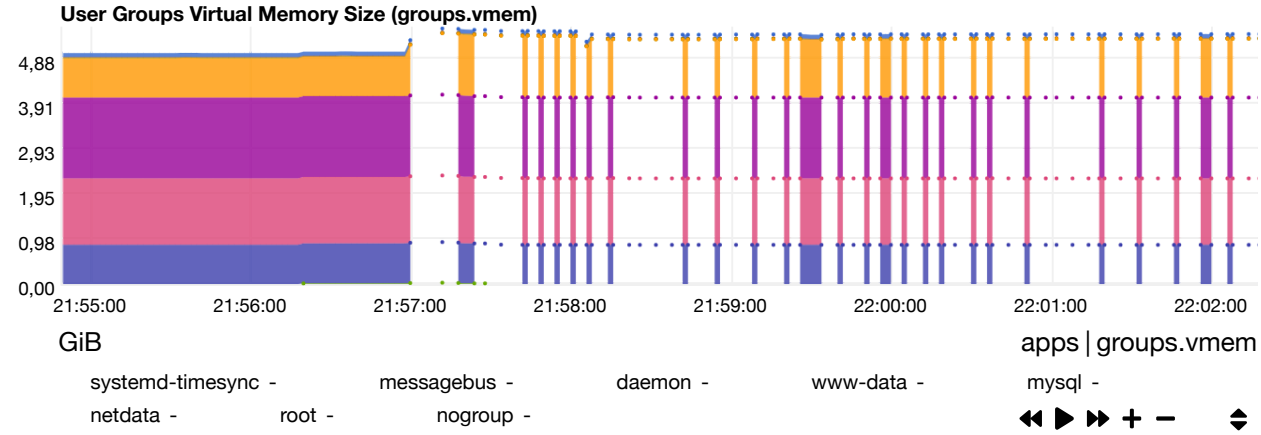

User Groups Open Files (groups.files)
150,0
100,0
50,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
open files
apps | groups.files
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
nogroup -

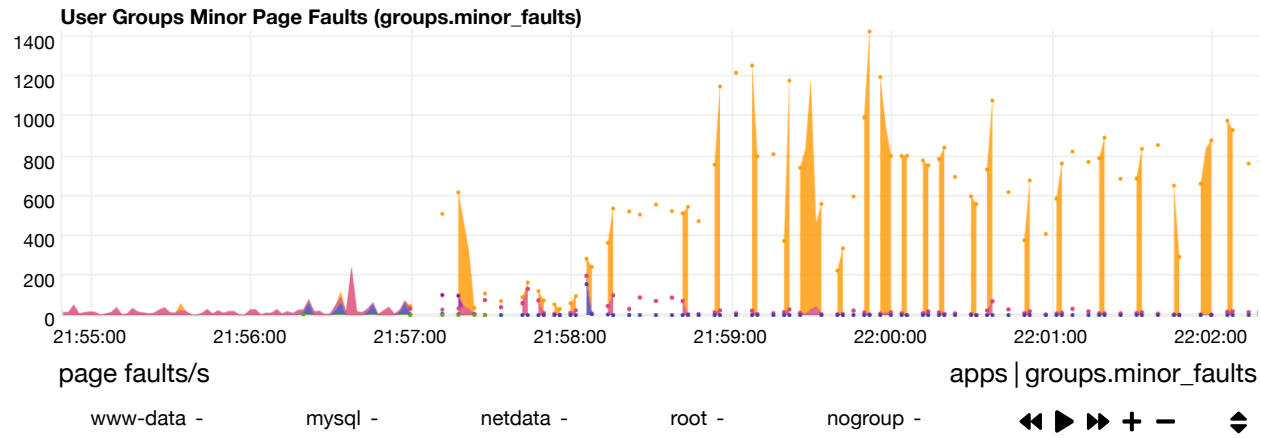
mem

Real memory (RAM) used per user group. This does not include shared memory.

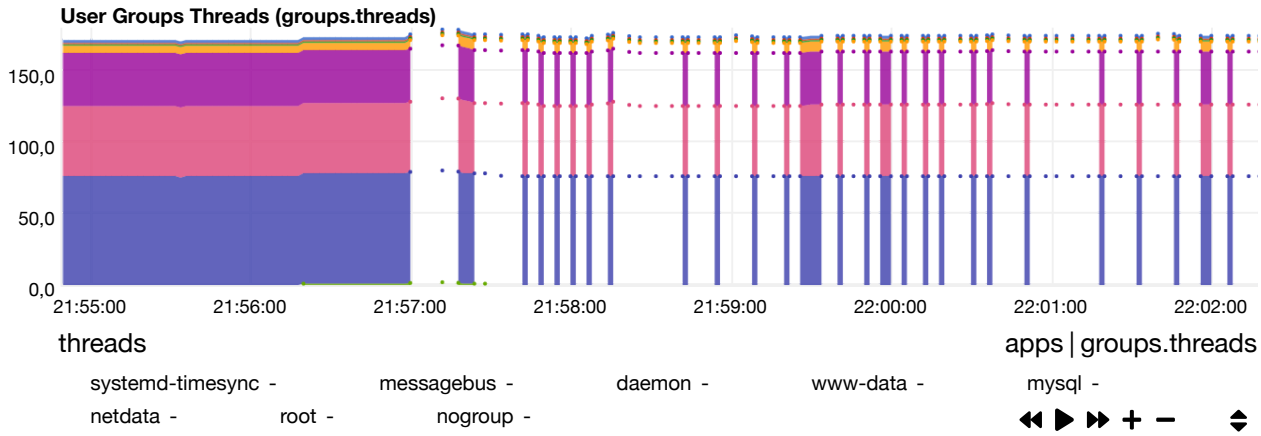


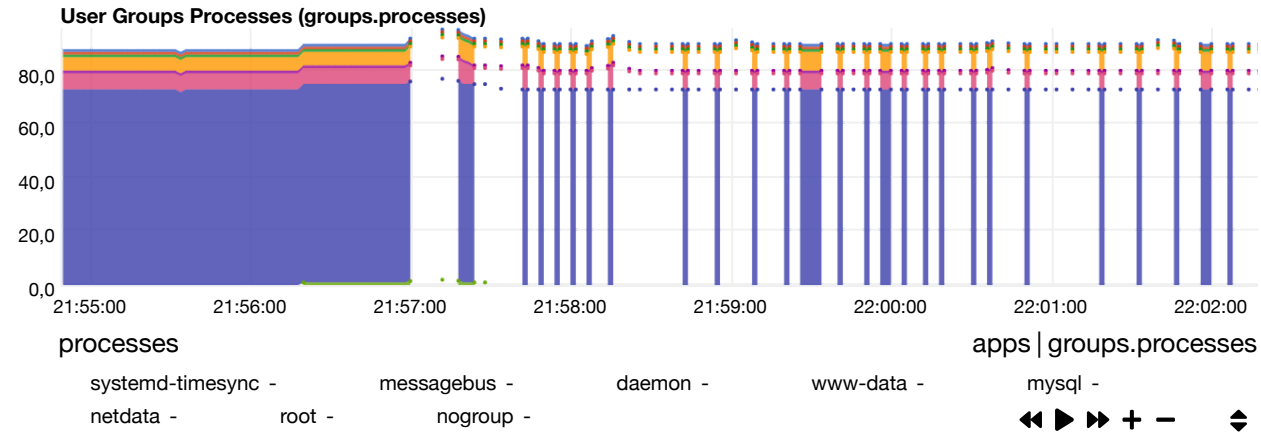
Virtual memory allocated per user group since the Netdata restart. Please check this article (https://github.com/netdata/netdata/tree/master/daemon#virtual-memory) for more information.



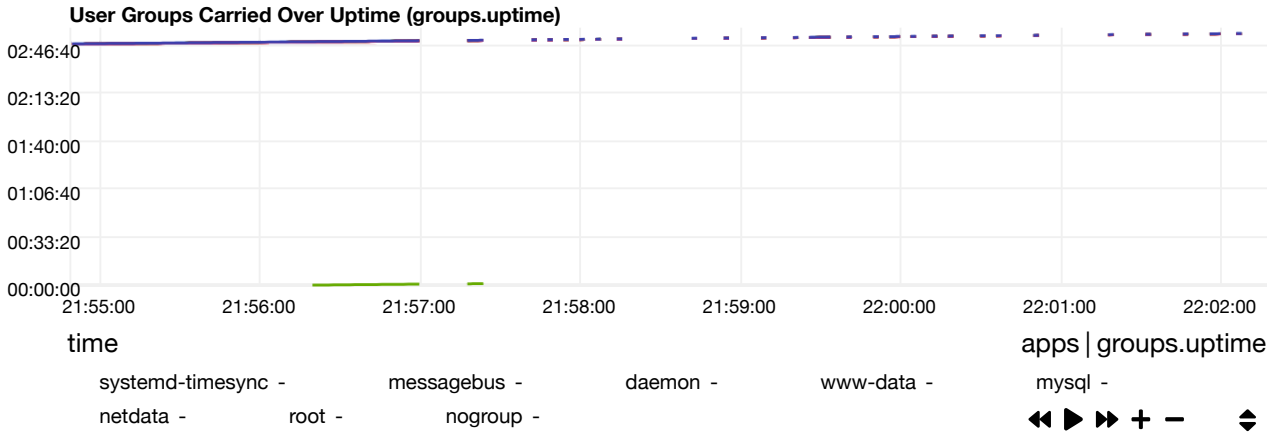


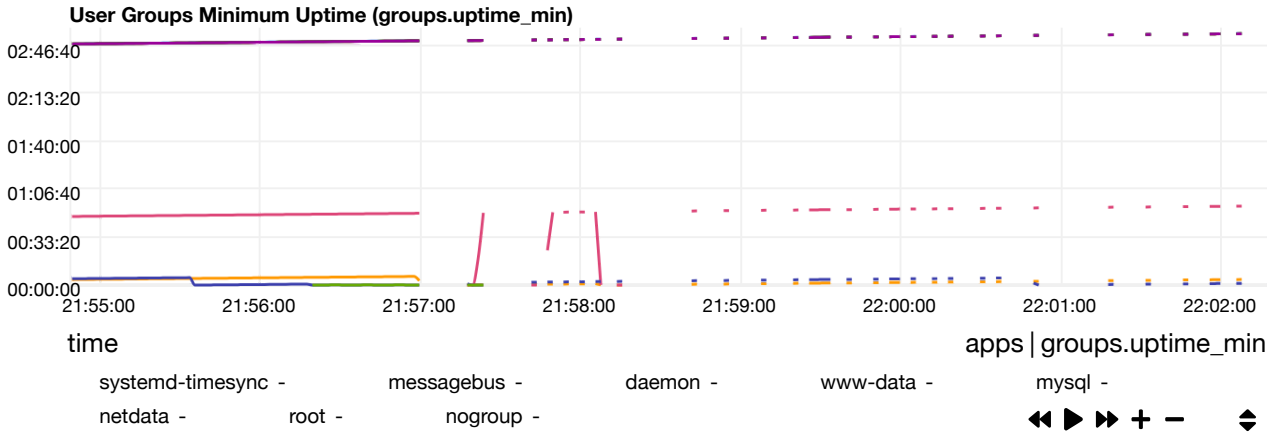
processes

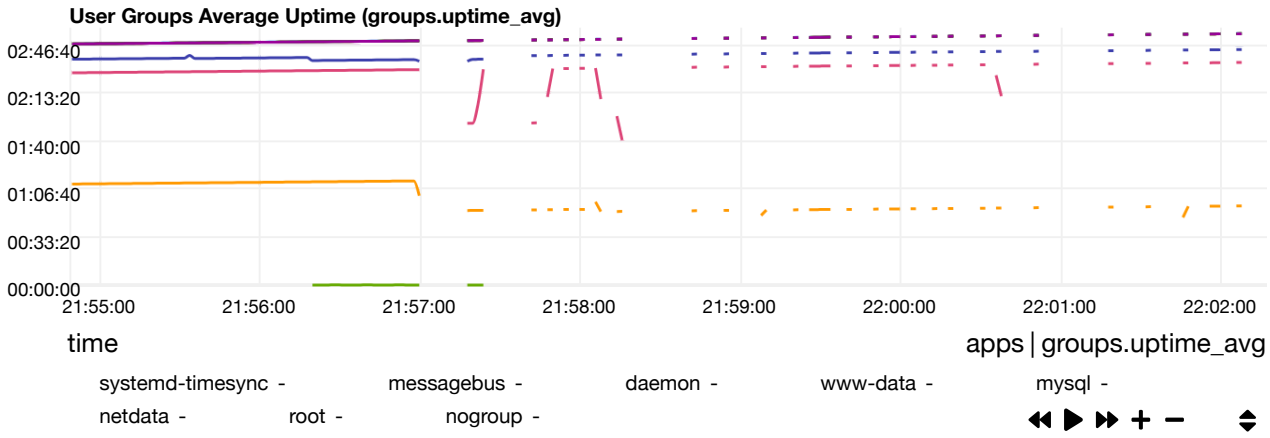


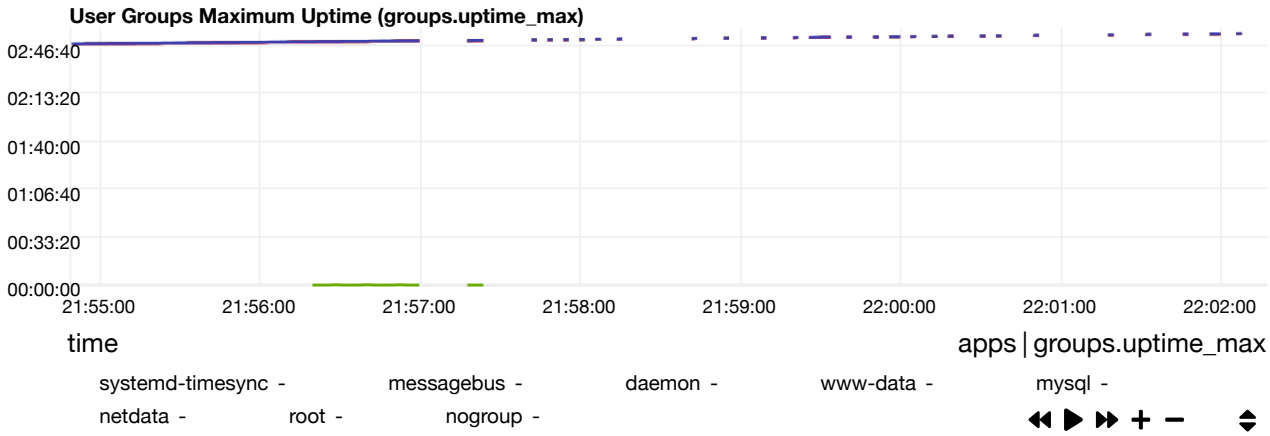


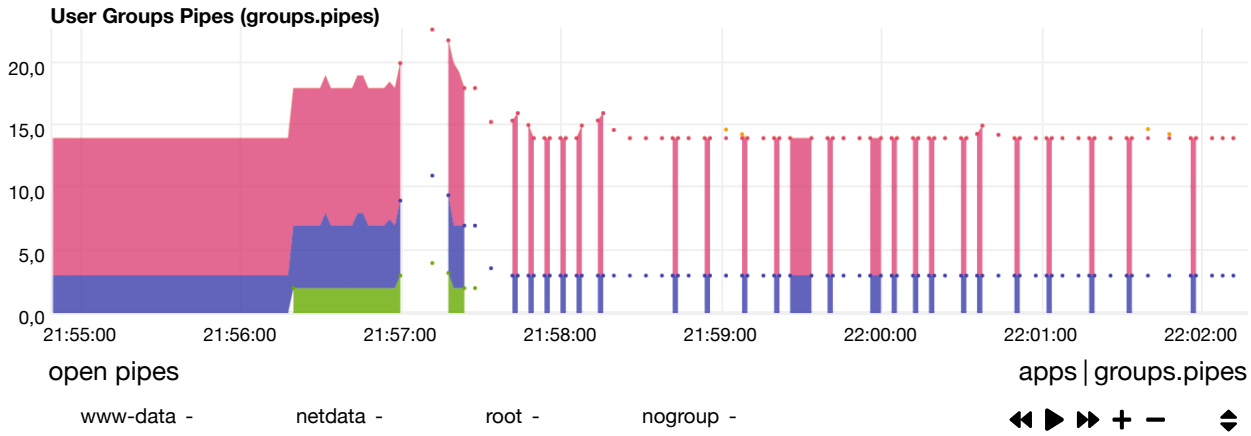
Carried over process group uptime. The period of time within which at least one process in the group was running.



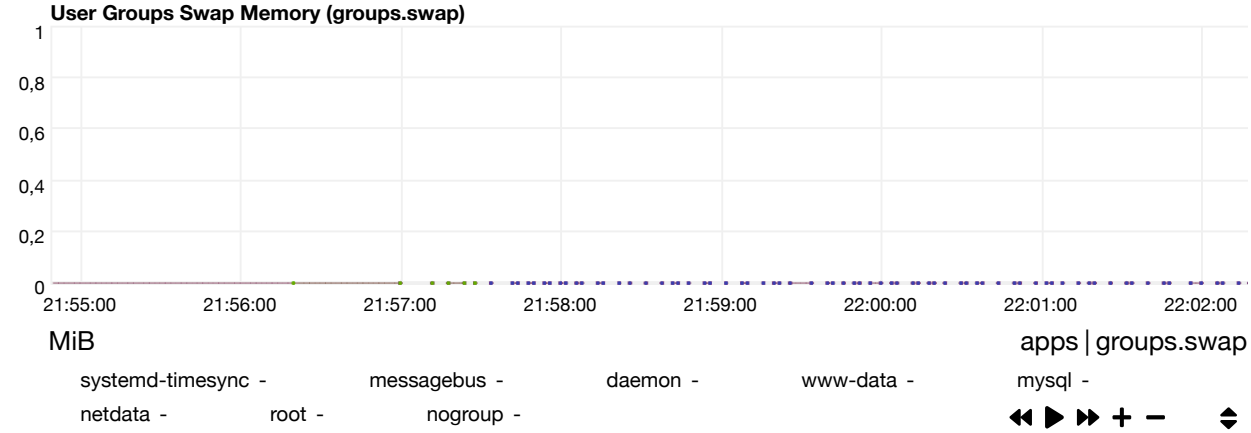


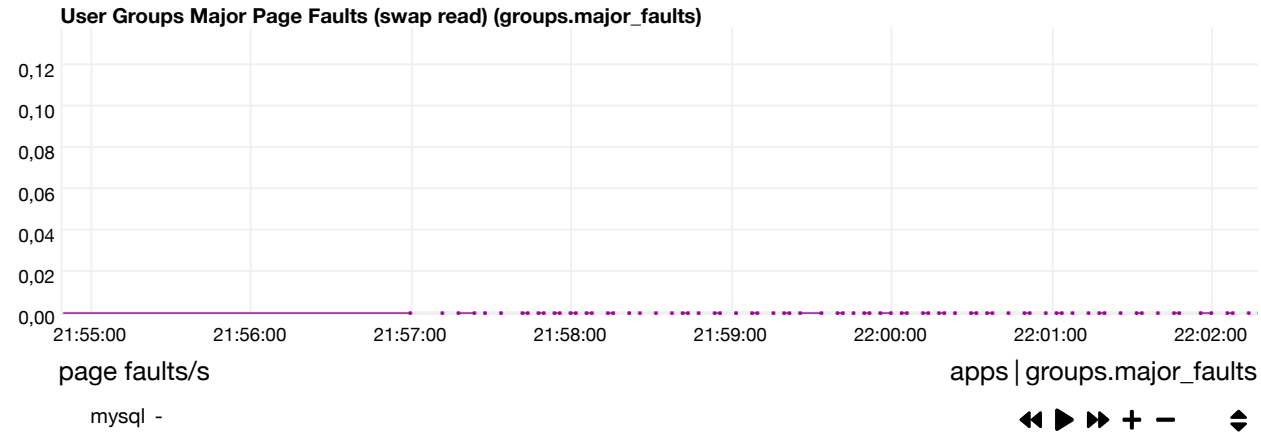

User Groups Average Uptime (groups.uptime_avg)
02:46:40
02:13:20
01:40:00
01:06:40
00:33:20
00:00:00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
time
apps | groups.uptime_avg
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
nogroup -


User Groups Maximum Uptime (groups.uptime_max)
02:46:40
02:13:20
01:40:00
01:06:40
00:33:20
00:00:00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
time
apps | groups.uptime_max
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
nogroup -

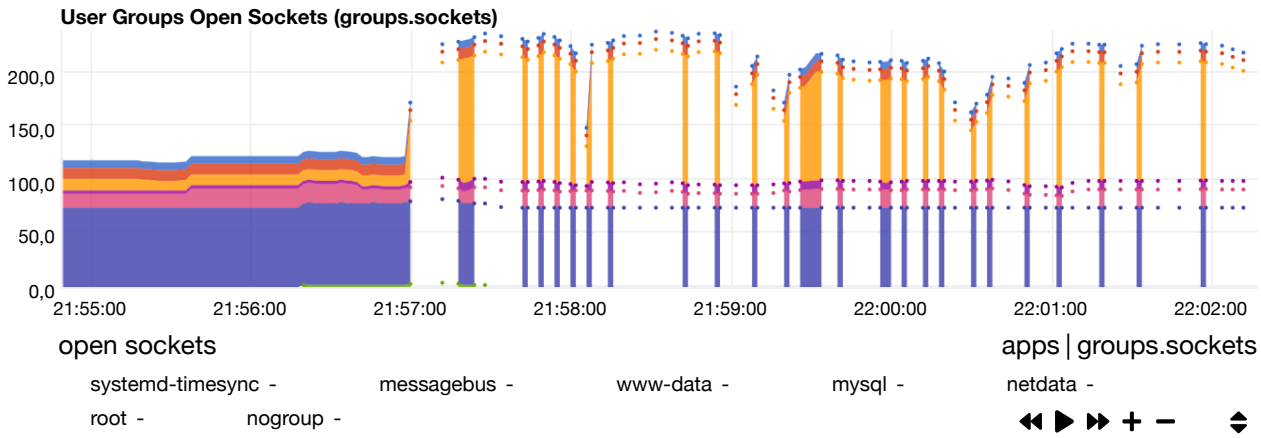

User Groups Pipes (groups.pipes)
20,0
15,0
10,0
5,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
open pipes
apps | groups.pipes
www-data -
netdata -
root -
nogroup -

swap


User Groups Swap Memory (groups.swap)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB
apps | groups.swap
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
nogroup -



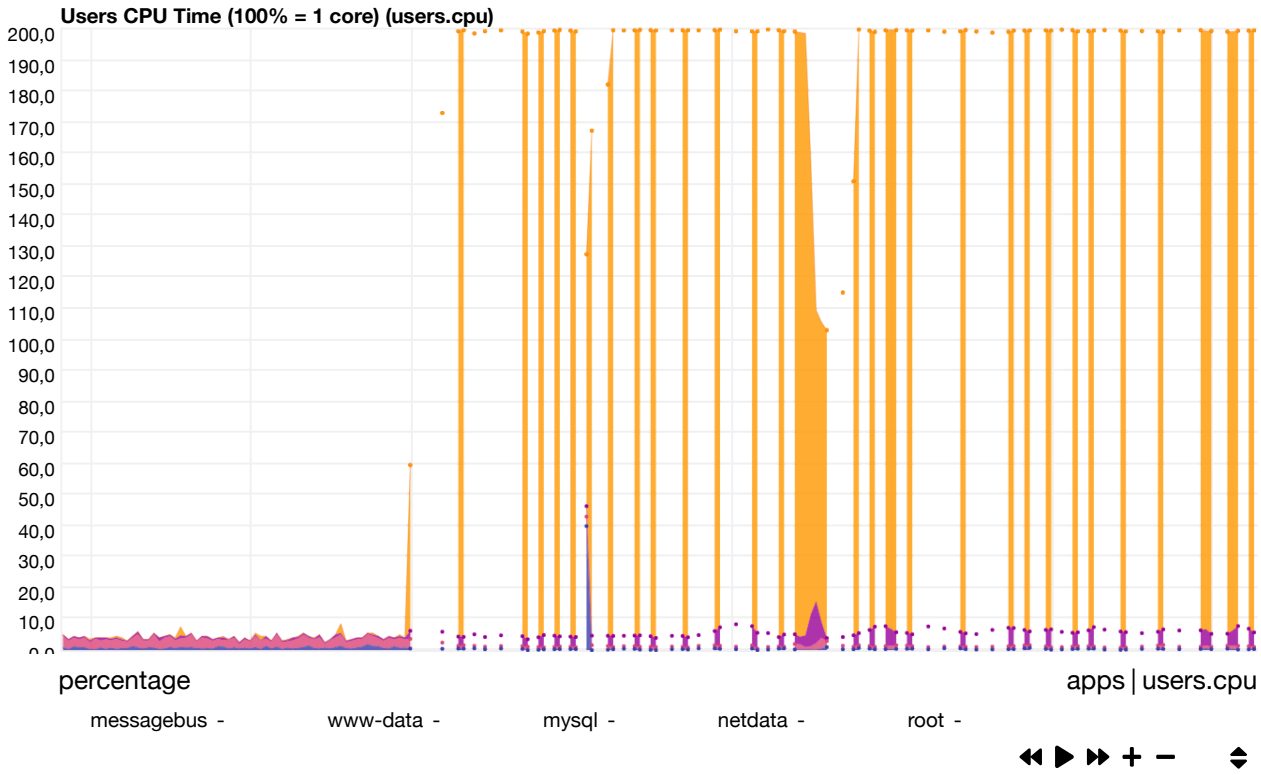
net

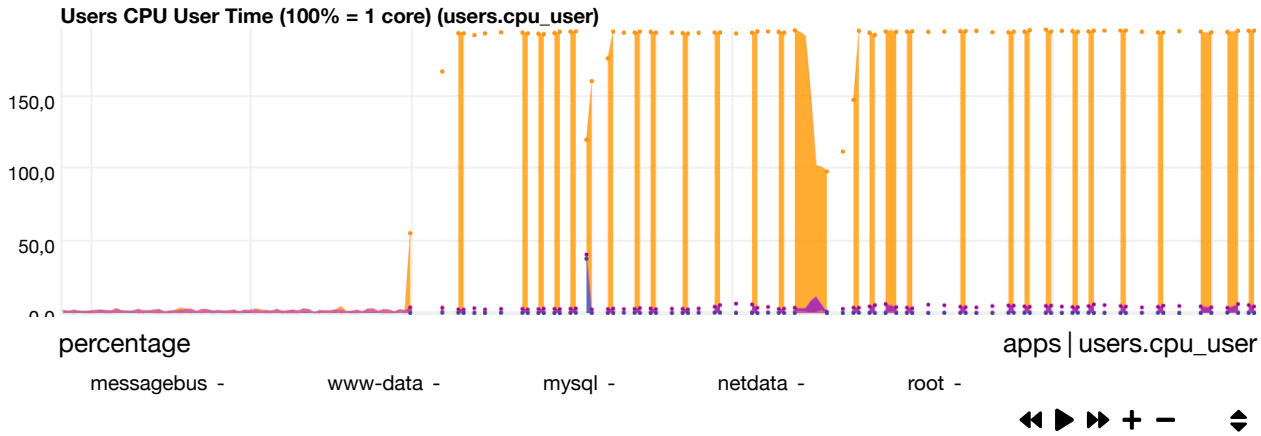


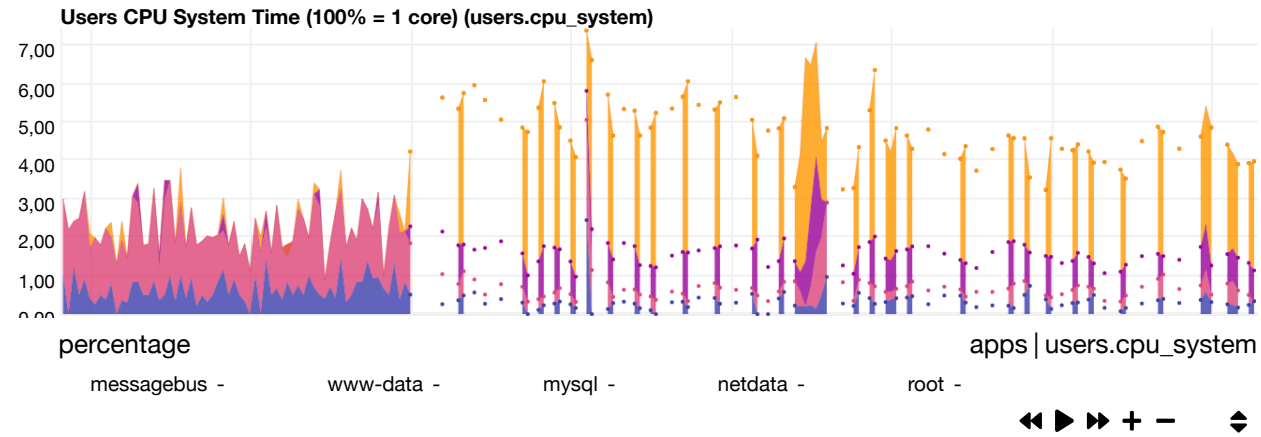
Users

Per user statistics are collected using netdata's `apps.plugin` . This plugin walks through all processes and aggregates statistics per user. The reported values are compatible with `top` , although the netdata plugin counts also the resources of exited children (unlike `top` which shows only the resources of the currently running processes). So for processes like shell scripts, the reported values include the resources used by the commands these scripts run within each timeframe.

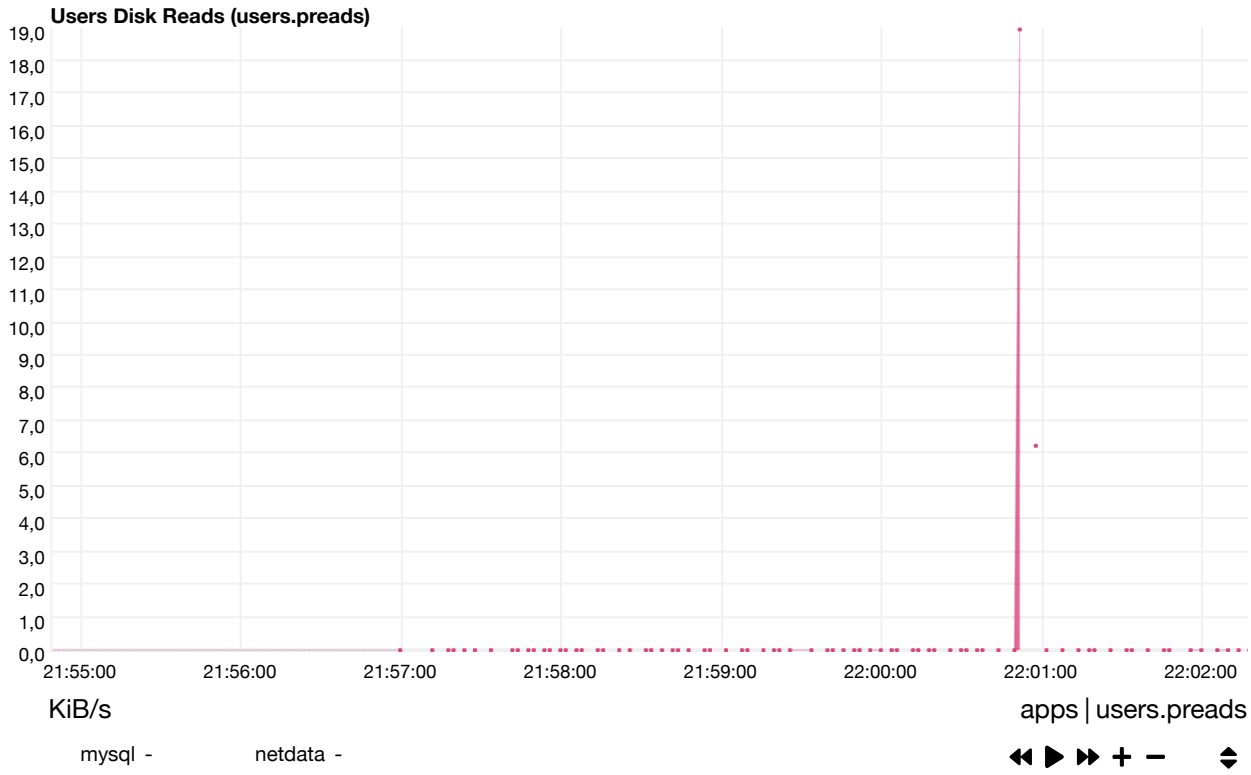
cpu

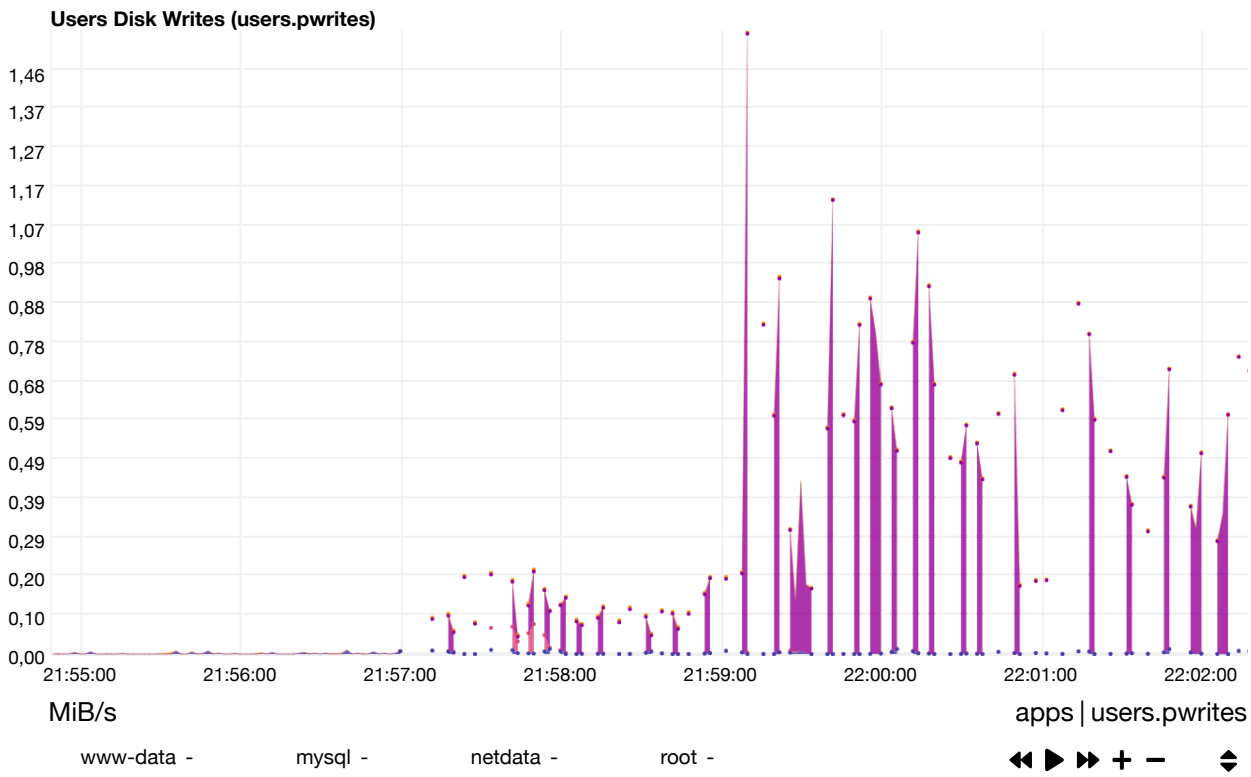

Users CPU Time (100% = 1 core) (users.cpu)
200,0
190,0
180,0
170,0
160,0
150,0
140,0
130,0
120,0
110,0
100,0
90,0
80,0
70,0
60,0
50,0
40,0
30,0
20,0
10,0
percentage
apps | users.cpu
messagebus -
www-data -
mysql -
netdata -
root -

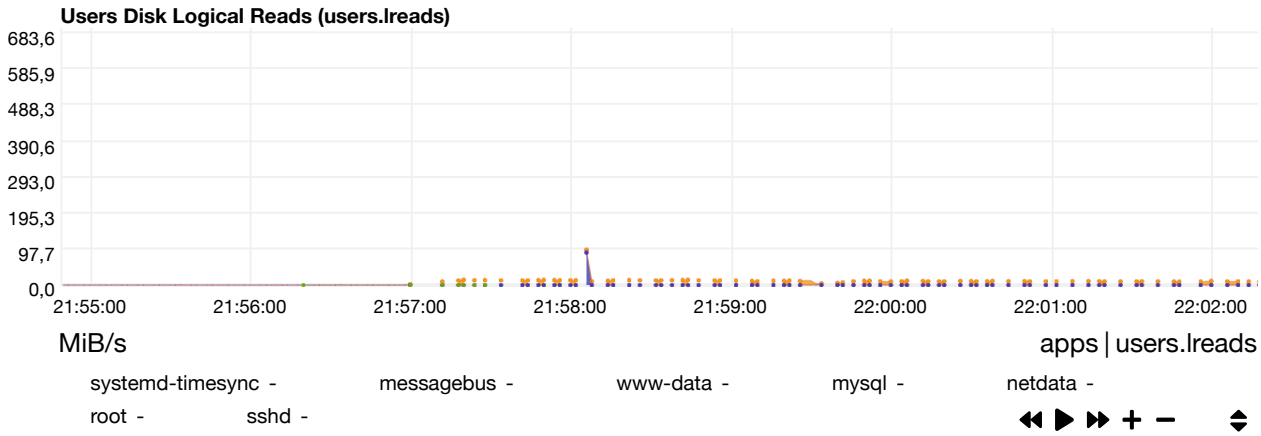

Users CPU User Time (100% = 1 core) (users.cpu_user)
150,0
100,0
50,0
percentage
apps | users.cpu_user
messagebus -
www-data -
mysql -
netdata -
root -

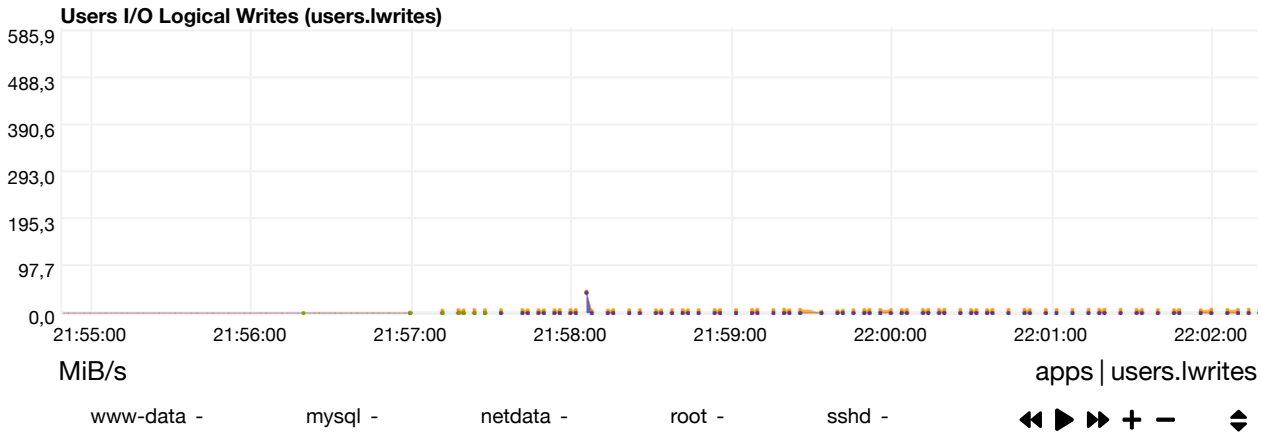

Users CPU System Time (100% = 1 core) (users.cpu_system)
7,00
6,00
5,00
4,00
3,00
2,00
1,00
percentage
apps | users.cpu_system
messagebus -
www-data -
mysql -
netdata -
root -

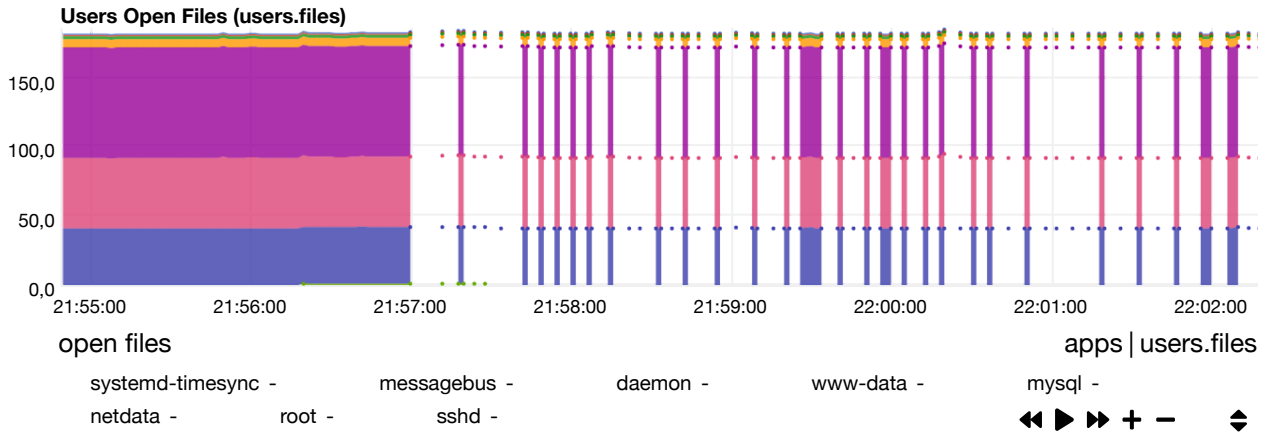
disk


Users Disk Reads (users.preads)
19,0
18,0
17,0
16,0
15,0
14,0
13,0
12,0
11,0
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
KiB/s
apps | users.preads
mysql -
netdata -


Users Disk Writes (users.pwrites)
1,46
1,37
1,27
1,17
1,07
0,98
0,88
0,78
0,68
0,59
0,49
0,39
0,29
0,20
0,10
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | users.pwrites
www-data -
mysql -
netdata -
root -

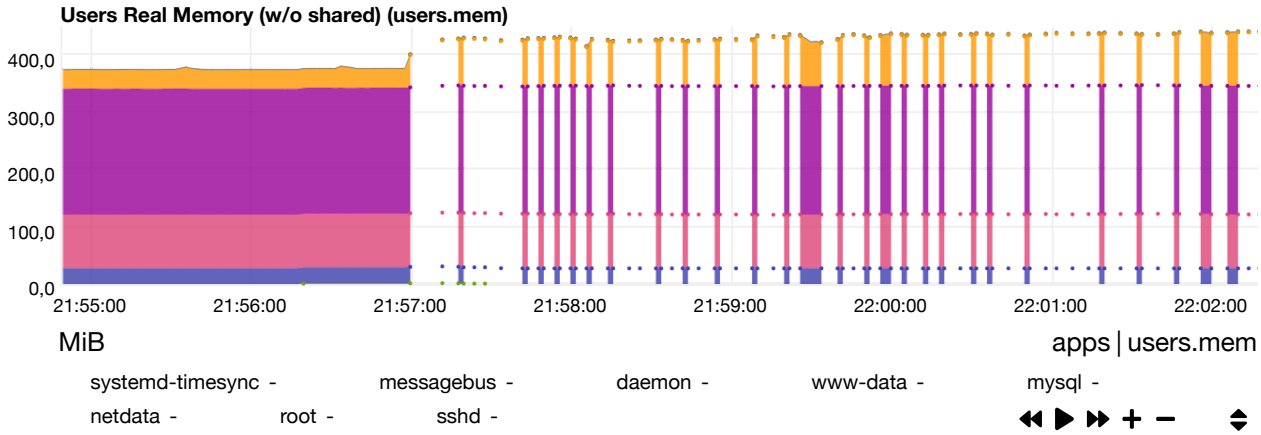

Users Disk Logical Reads (users.lreads)
683,6
585,9
488,3
390,6
293,0
195,3
97,7
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | users.lreads
systemd-timesync -
messagebus -
www-data -
mysql -
netdata -
root -
sshd -


Users I/O Logical Writes (users.lwrites)
585,9
488,3
390,6
293,0
195,3
97,7
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
apps | users.lwrites
www-data -
mysql -
netdata -
root -
sshd -

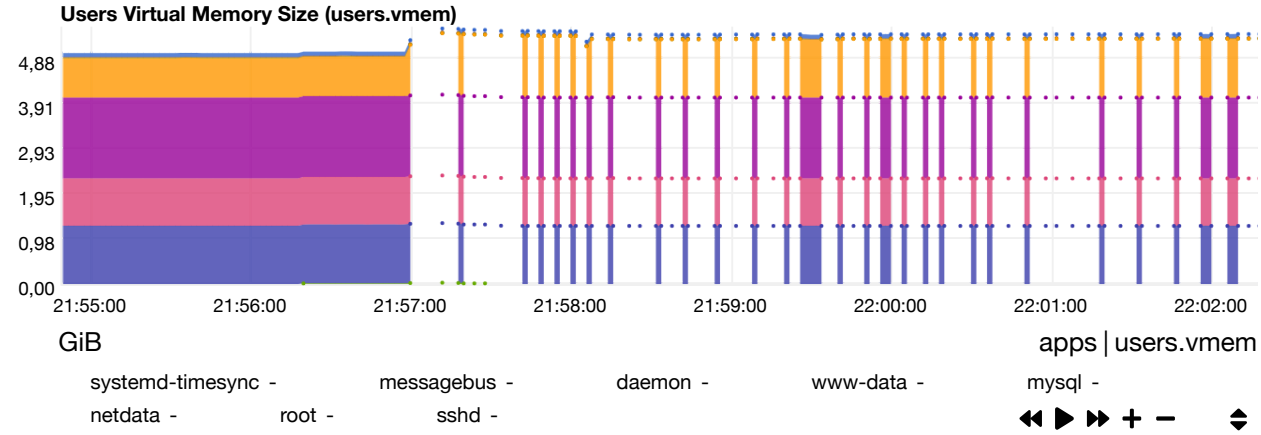

Users Open Files (users.files)
150,0
100,0
50,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
open files
apps | users.files
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
sshd -

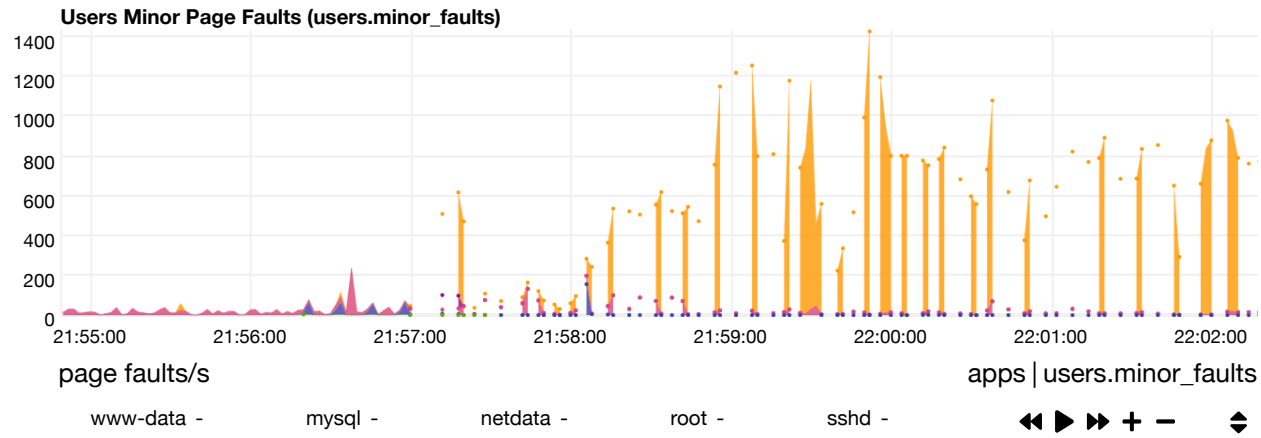
mem

Real memory (RAM) used per user. This does not include shared memory.



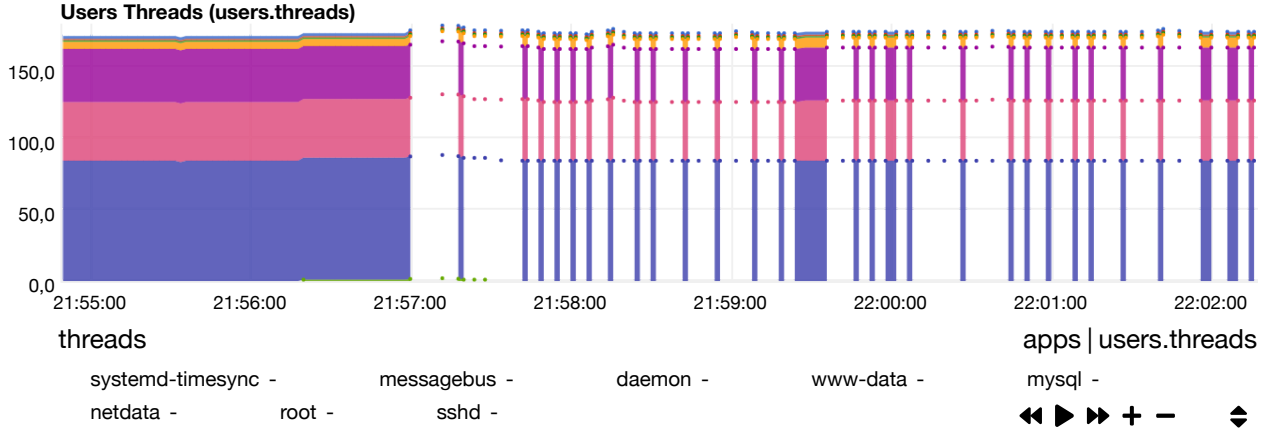
Virtual memory allocated per user. Please check this article (https://github.com/netdata/netdata/tree/master/daemon#virtual-memory) for more information.



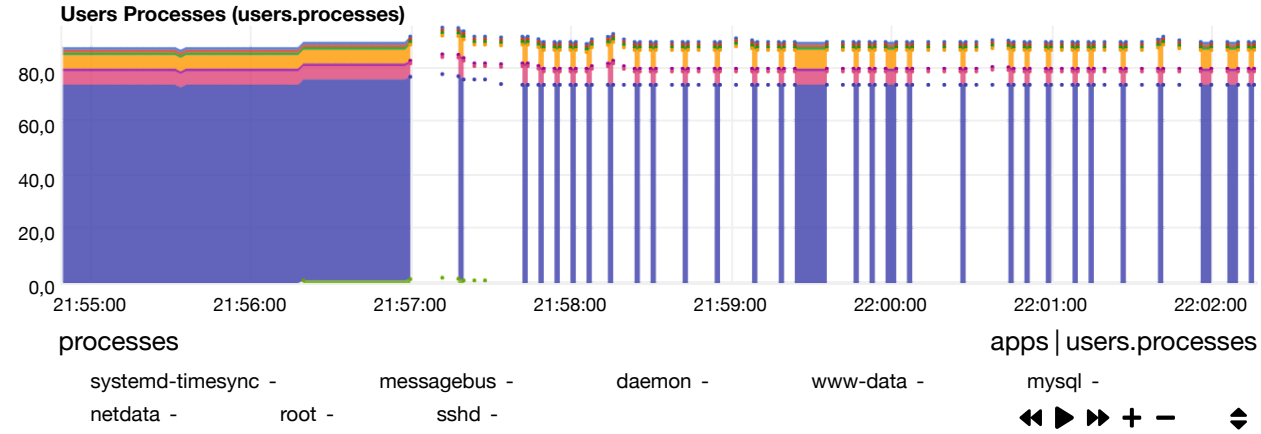


processes

Users Threads (users.threads)

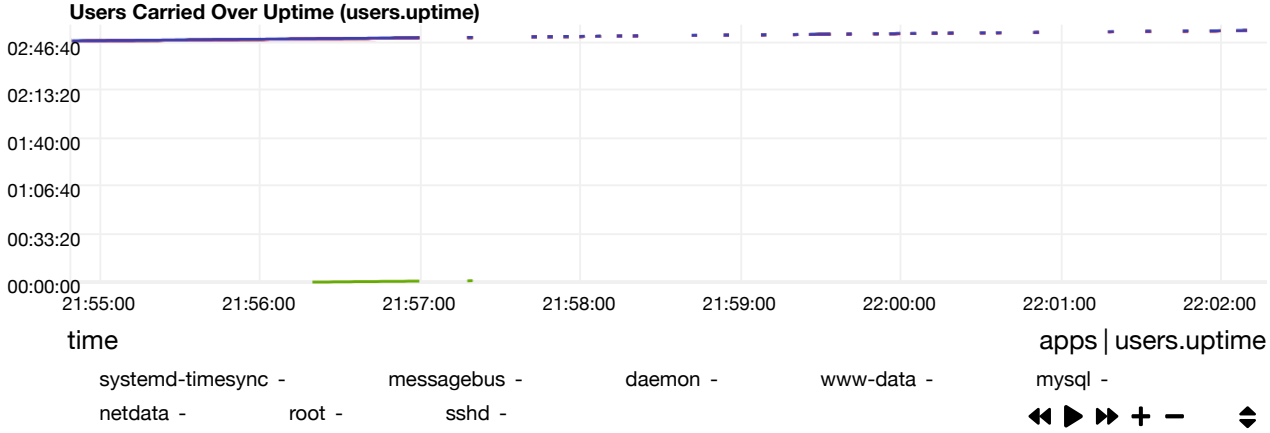


Users Processes (users.processes)

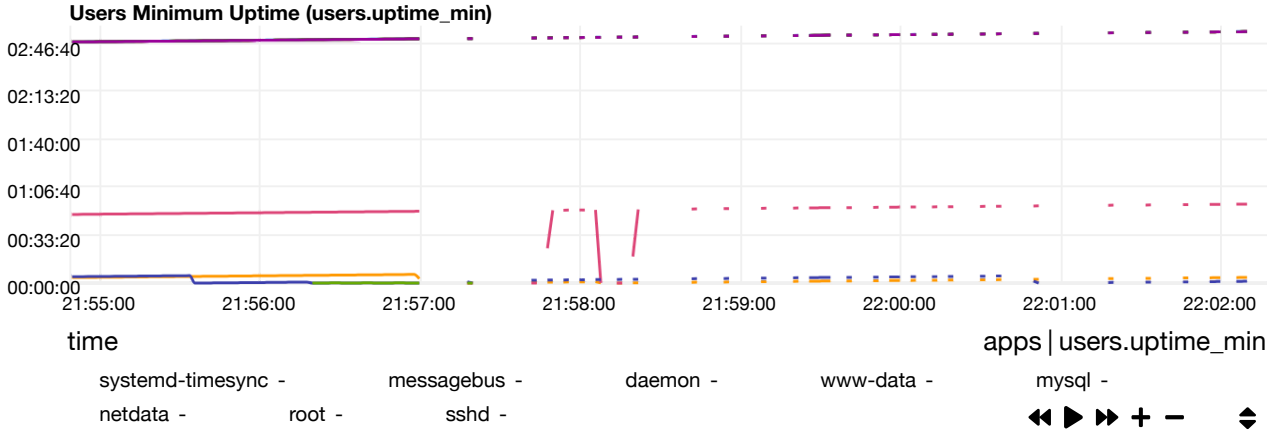


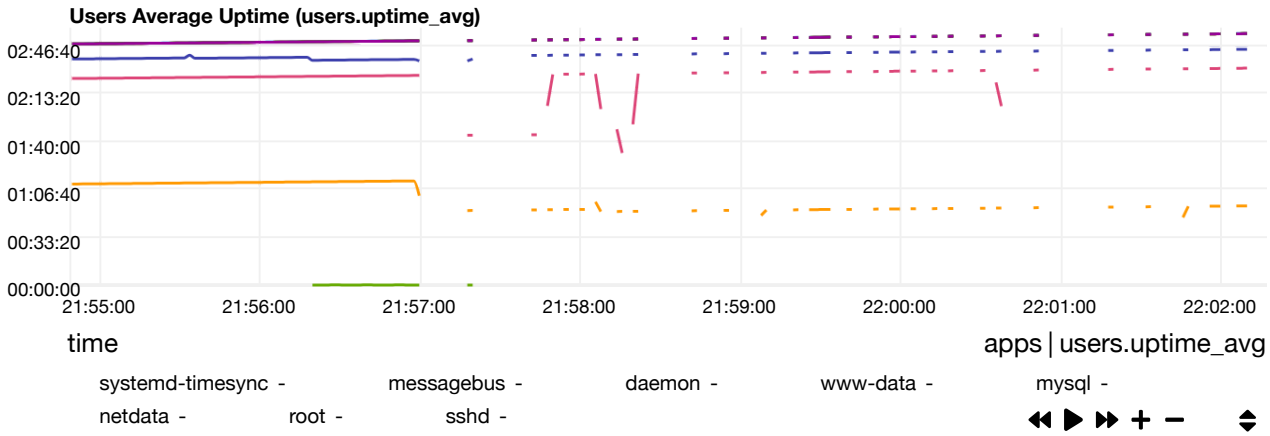
Carried over process group uptime since the Netdata restart. The period of time within which at least one process in the group was running.

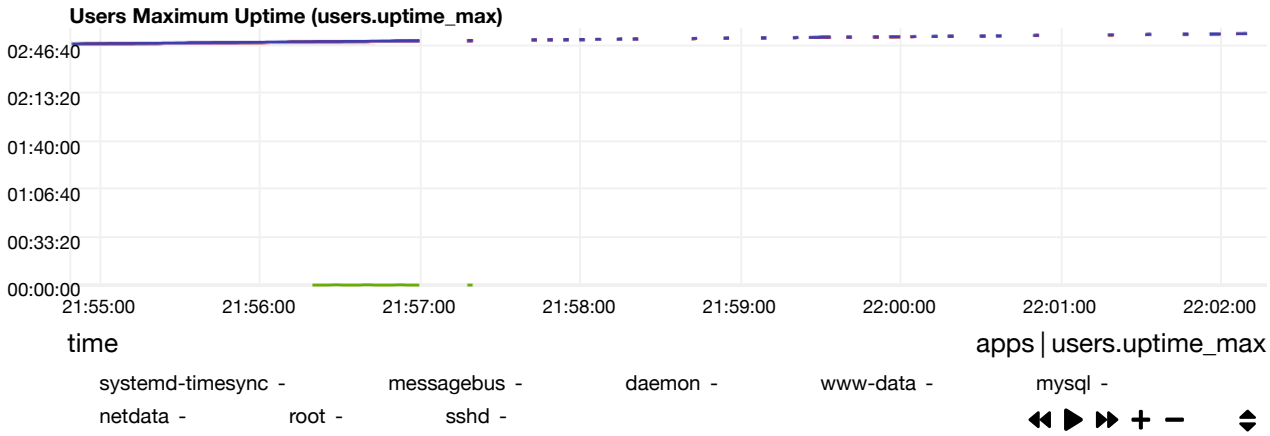
Users Carried Over Uptime (users.uptime)

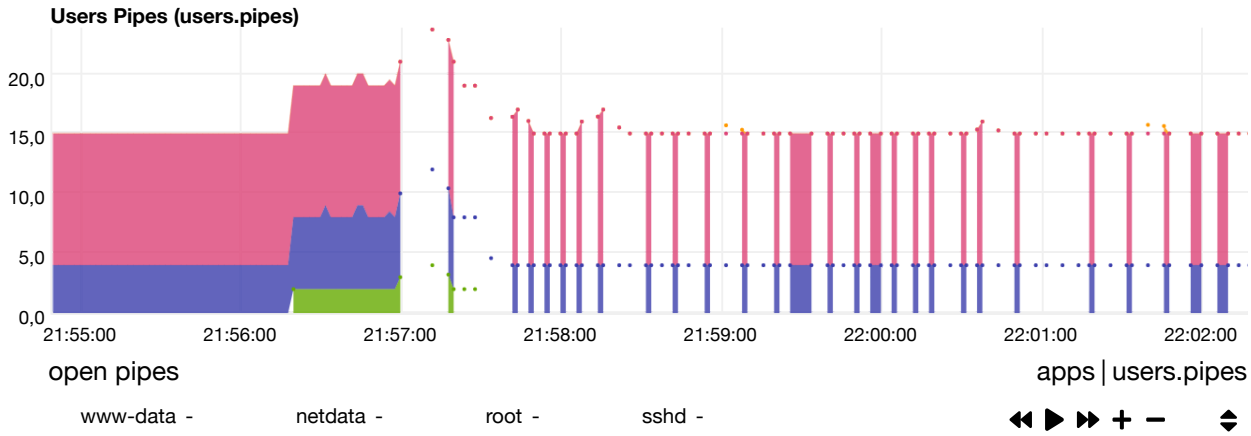


Users Minimum Uptime (users.uptime_min)

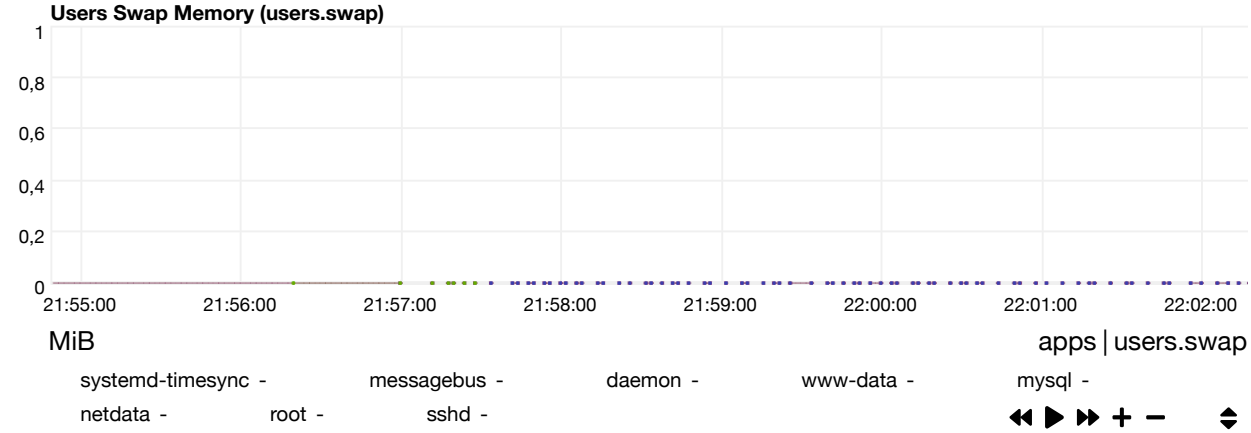


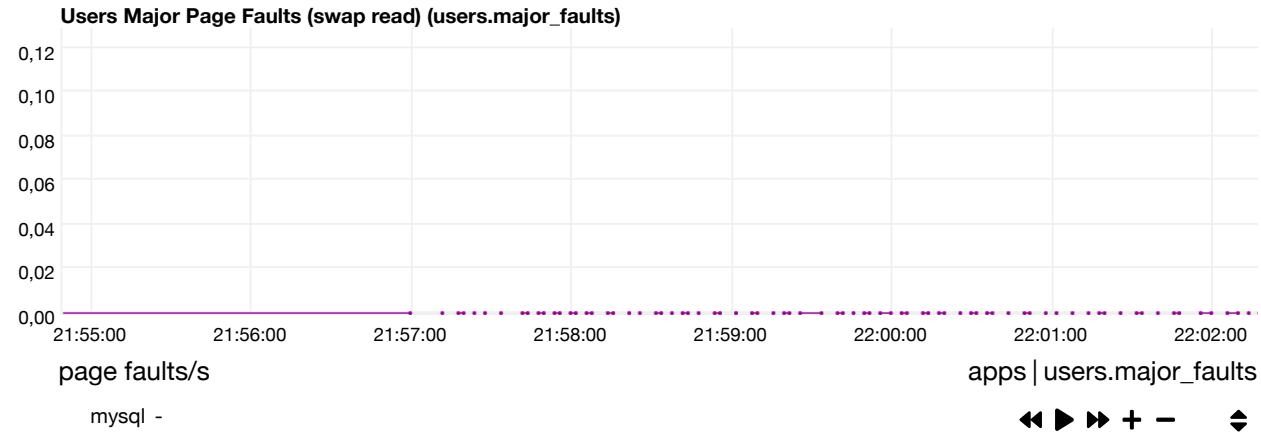

Users Average Uptime (users.uptime_avg)
02:46:40
02:13:20
01:40:00
01:06:40
00:33:20
00:00:00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
time
apps | users.uptime_avg
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
sshd -


Users Maximum Uptime (users.uptime_max)
02:46:40
02:13:20
01:40:00
01:06:40
00:33:20
00:00:00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
time
apps | users.uptime_max
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
sshd -

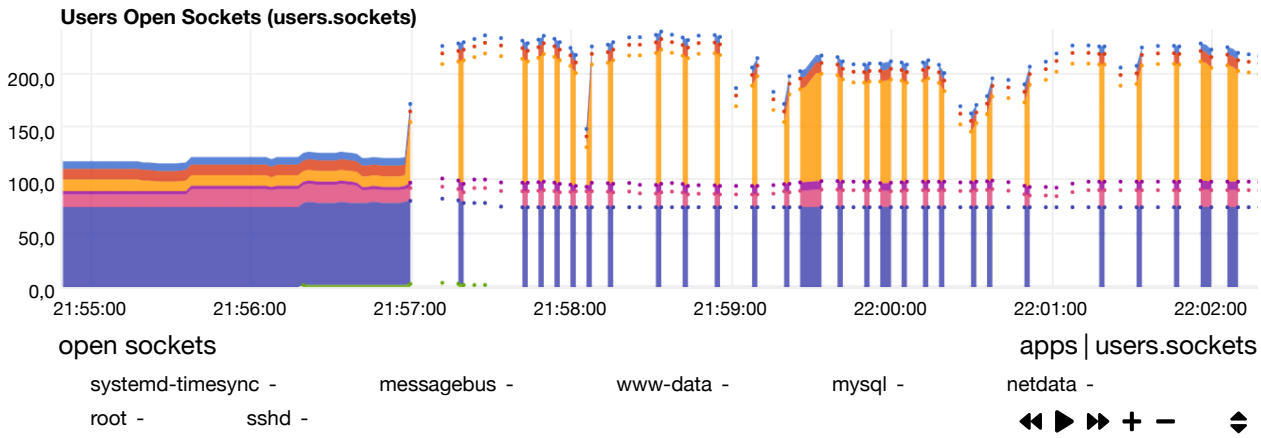

Users Pipes (users.pipes)
20,0
15,0
10,0
5,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
open pipes
apps | users.pipes
www-data -
netdata -
root -
sshd -

swap


Users Swap Memory (users.swap)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB
apps | users.swap
systemd-timesync -
messagebus -
daemon -
www-data -
mysql -
netdata -
root -
sshd -



net



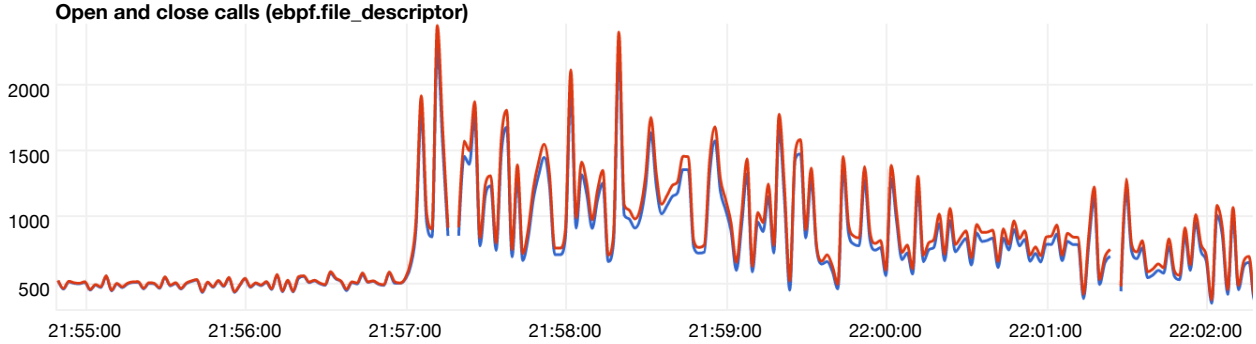
eBPF

Monitor system calls, internal functions, bytes read, bytes written and errors using eBPF .

File

Calls for internal functions on Linux kernel. The open dimension is attached to the kernel internal function `do_sys_open` (For kernels newer than `5.5.19` we add a kprobe to `do_sys_openat2`), which is the common function called from open(2) (https://www.man7.org/linux/man-pages/man2/open.2.html) and openat(2) (https://www.man7.org/linux/man-pages/man2/openat.2.html). The close dimension is attached to the function `__close_fd` or `close_fd` according to your kernel version, which is called from system call close(2) (https://www.man7.org/linux/man-pages/man2/close.2.html).

Open and close calls (ebpf.file_descriptor)



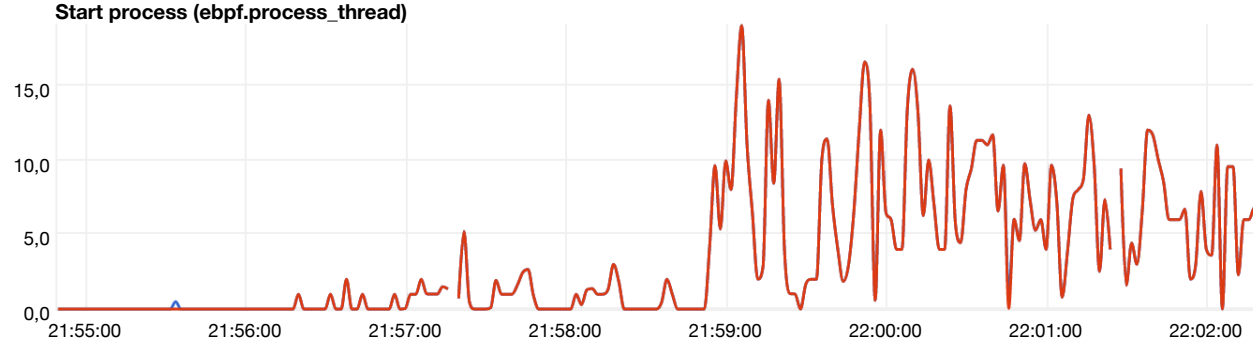
calls/s ebpf | ebpf.file_descriptor

open - close -

Process

Number of times that either do_fork (https://www.ece.uic.edu/~yshi1/linux/lkse/node4.html#SECTION00421000000000000000), or `kernel_clone` if you are running kernel newer than 5.9.16, is called to create a new task, which is the common name used to define process and tasks inside the kernel. Netdata identifies the threads by counting the number of calls for sys_clone (https://linux.die.net/man/2/clone) that has the flag `CLONE_THREAD` set.

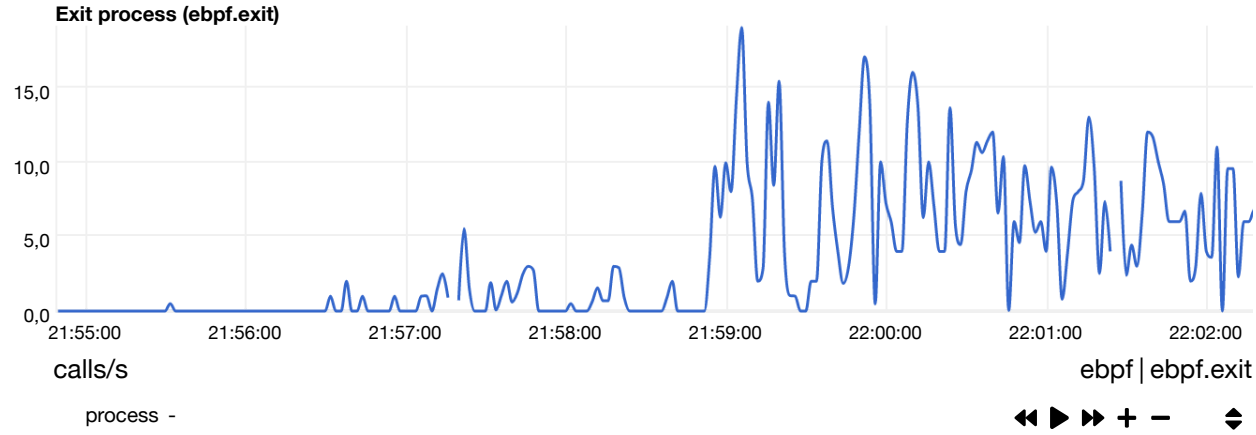
Start process (ebpf.process_thread)



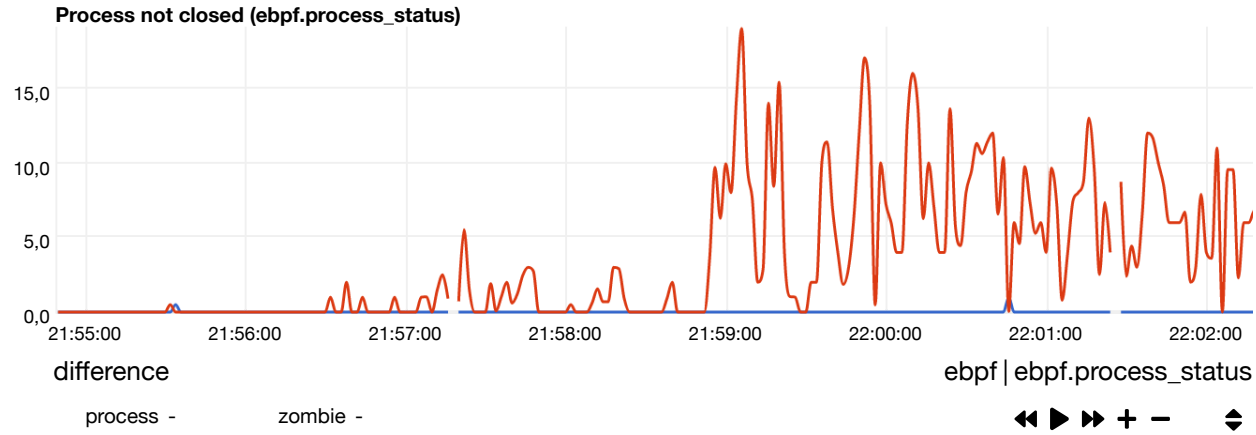
calls/s ebpf | ebpf.process_thread

process - thread -

Calls for the functions responsible for closing (do_exit (https://www.informit.com/articles/article.aspx?p=370047&seqNum=4)) and releasing (release_task (https://www.informit.com/articles/article.aspx?p=370047&seqNum=4)) tasks.

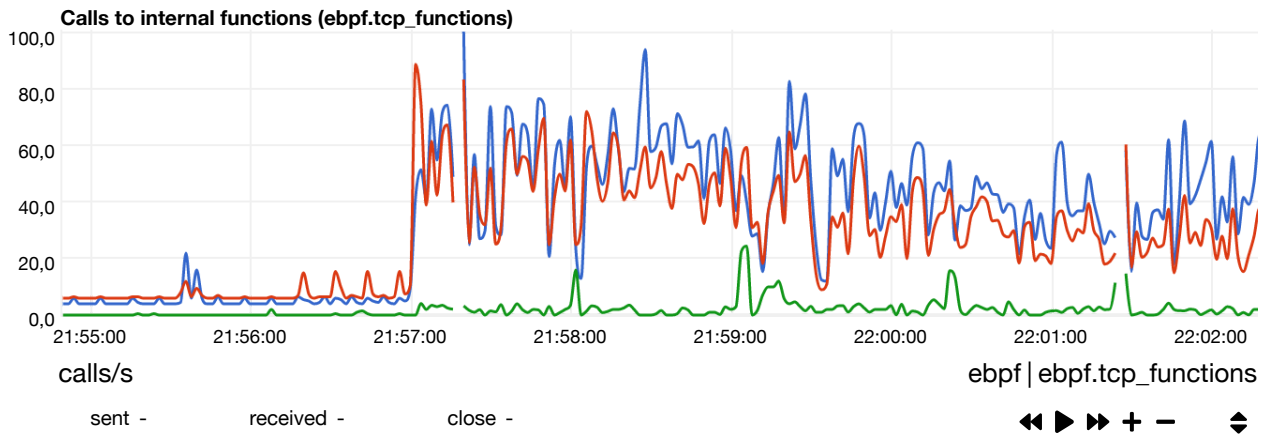


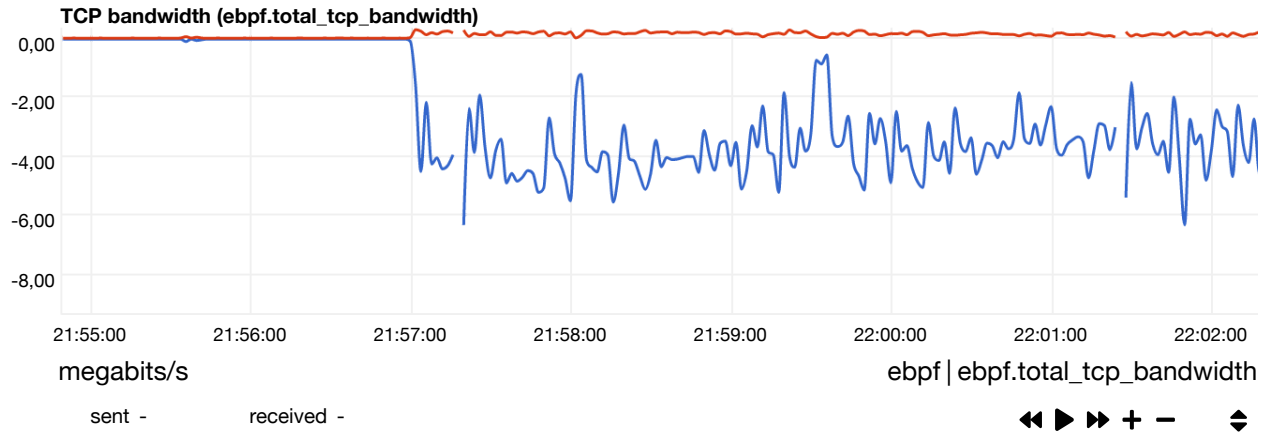
Difference between the number of process created and the number of threads created per period(`process` dimension), it also shows the number of possible zombie process running on system.



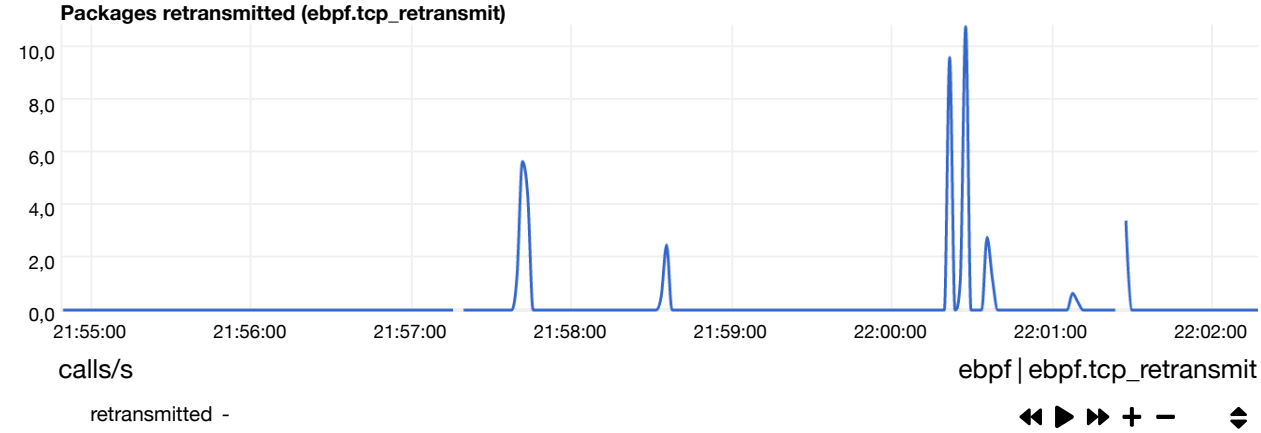
Socket

Successful or failed calls to functions `tcp_sendmsg` , `tcp_cleanup_rbuf` and `tcp_close` .

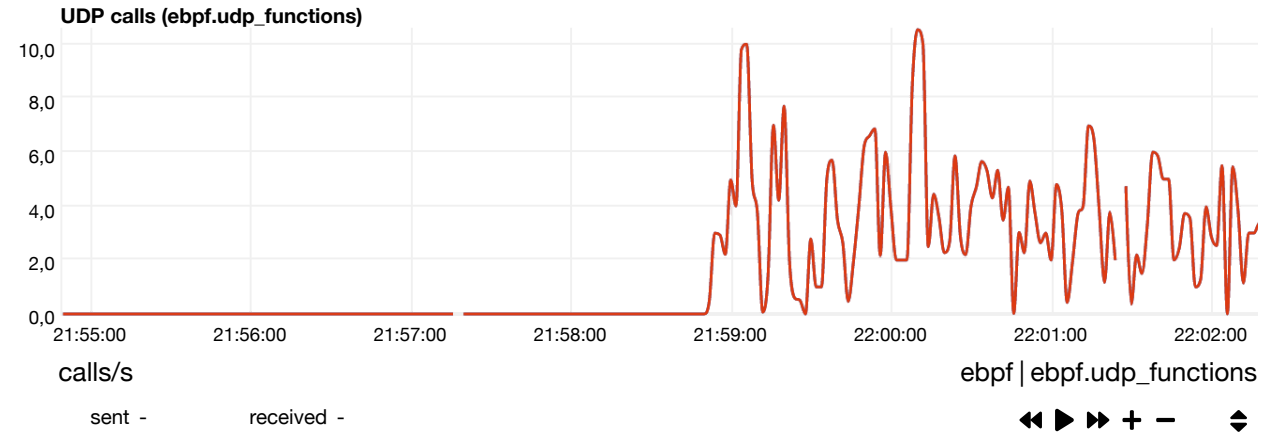


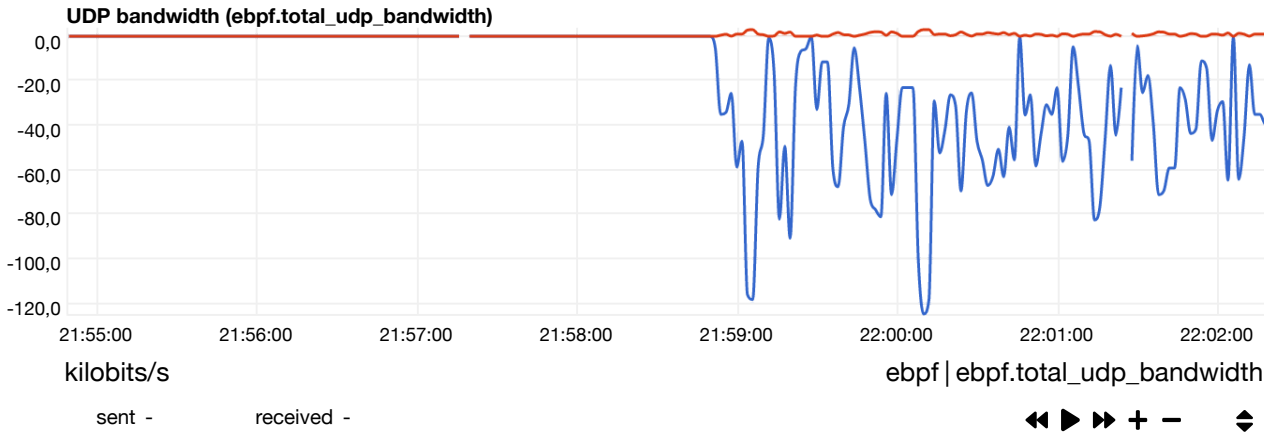


Number of packets retransmitted for function `tcp_retransmit_skb`.



Successful or failed calls to functions `udp_sendmsg` and `udp_recvmsg`.



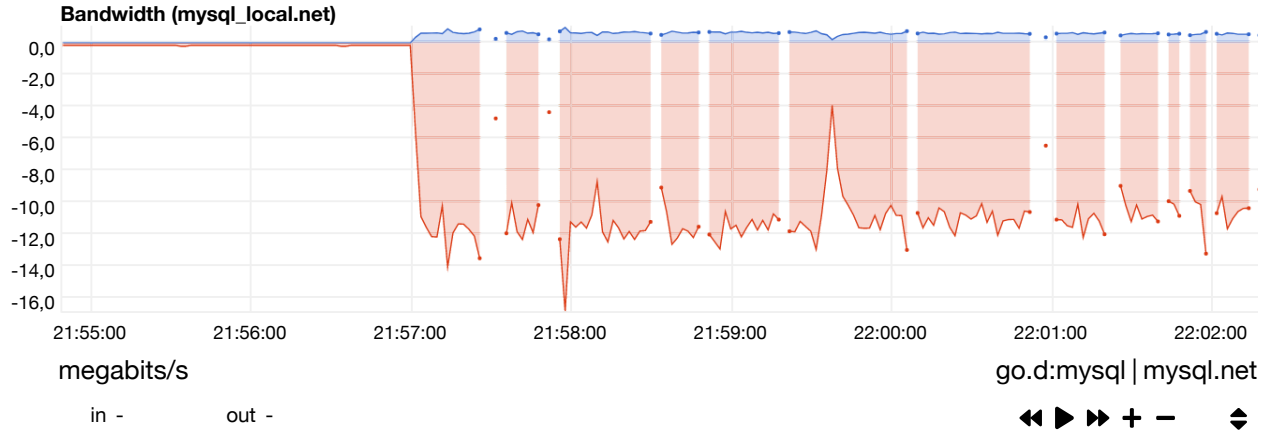


MySQL local

Performance metrics for **mysql**, the open-source relational database management system (RDBMS).

bandwidth

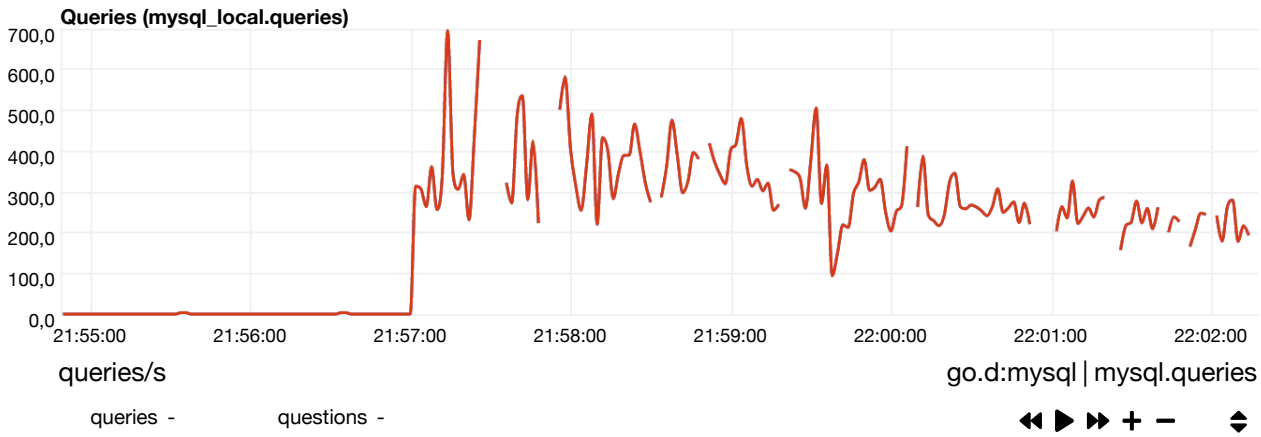
The amount of data sent to mysql clients (**out**) and received from mysql clients (**in**).

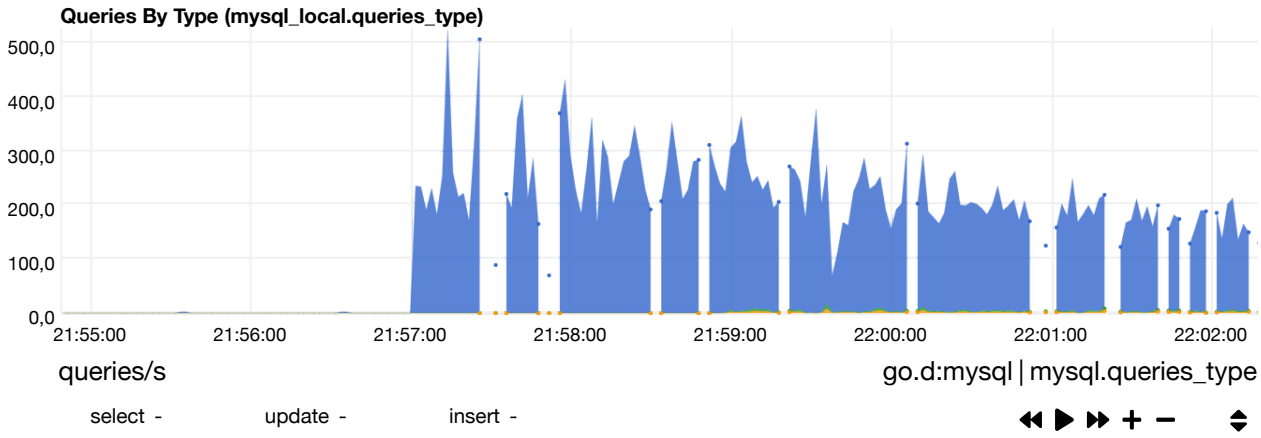


queries

The number of statements executed by the server.

- **queries** counts the statements executed within stored SQL programs.
- **questions** counts the statements sent to the mysql server by mysql clients.
- **slow queries** counts the number of statements that took more than long_query_time (http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_long_query_time) seconds to be executed. For more information about slow queries check the mysql slow query log (http://dev.mysql.com/doc/refman/5.7/en/slow-query-log.html).

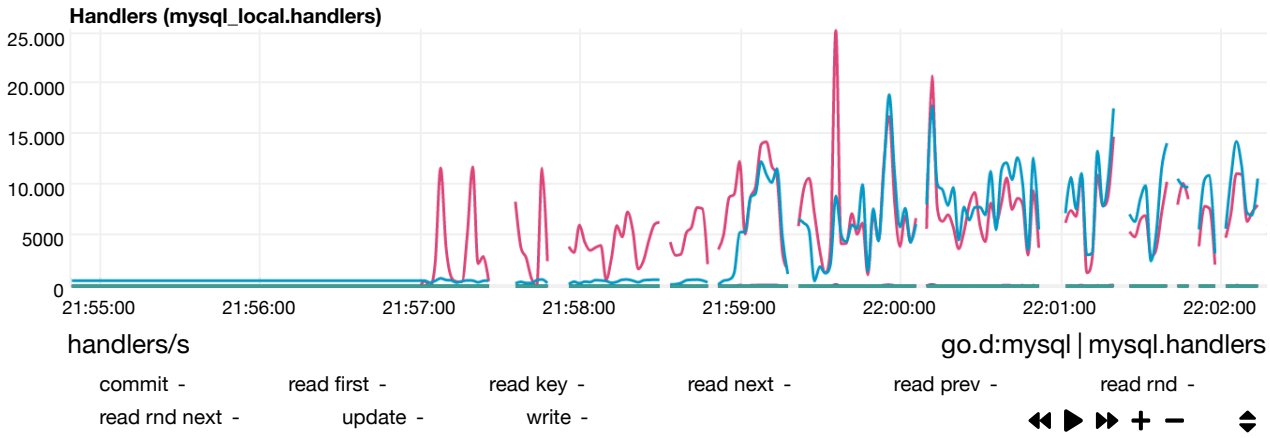



Queries By Type (mysql_local.queries_type)
500,0
400,0
300,0
200,0
100,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
queries/s
go.d:mysql | mysql.queries_type
select -
update -
insert -

handlers

Usage of the internal handlers of mysql. This chart provides very good insights of what the mysql server is actually doing. (if the chart is not showing all these dimensions it is because they are zero - set **Which dimensions to show?** to **All** from the dashboard settings, to render even the zero values)

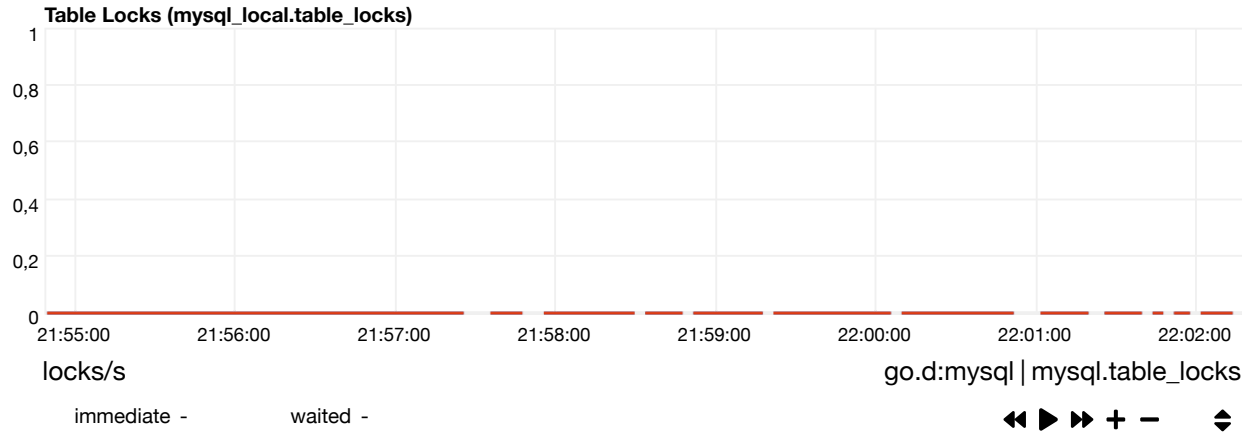
- **commit**, the number of internal COMMIT (http://dev.mysql.com/doc/refman/5.7/en/commit.html) statements.
- **delete**, the number of times that rows have been deleted from tables.
- **prepare**, a counter for the prepare phase of two-phase commit operations.
- **read first**, the number of times the first entry in an index was read. A high value suggests that the server is doing a lot of full index scans; e.g. **SELECT col1 FROM foo**, with col1 indexed.
- **read key**, the number of requests to read a row based on a key. If this value is high, it is a good indication that your tables are properly indexed for your queries.
- **read next**, the number of requests to read the next row in key order. This value is incremented if you are querying an index column with a range constraint or if you are doing an index scan.
- **read prev**, the number of requests to read the previous row in key order. This read method is mainly used to optimize **ORDER BY ... DESC**.
- **read rnd**, the number of requests to read a row based on a fixed position. A high value indicates you are doing a lot of queries that require sorting of the result. You probably have a lot of queries that require MySQL to scan entire tables or you have joins that do not use keys properly.
- **read rnd next**, the number of requests to read the next row in the data file. This value is high if you are doing a lot of table scans. Generally this suggests that your tables are not properly indexed or that your queries are not written to take advantage of the indexes you have.
- **rollback**, the number of requests for a storage engine to perform a rollback operation.
- **savepoint**, the number of requests for a storage engine to place a savepoint.
- **savepoint rollback**, the number of requests for a storage engine to roll back to a savepoint.
- **update**, the number of requests to update a row in a table.
- **write**, the number of requests to insert a row in a table.



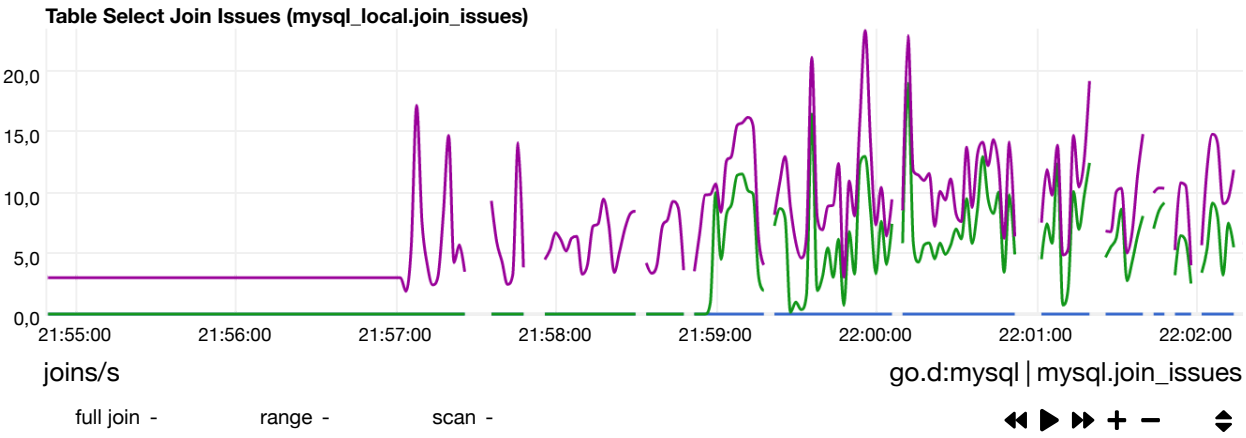
locks

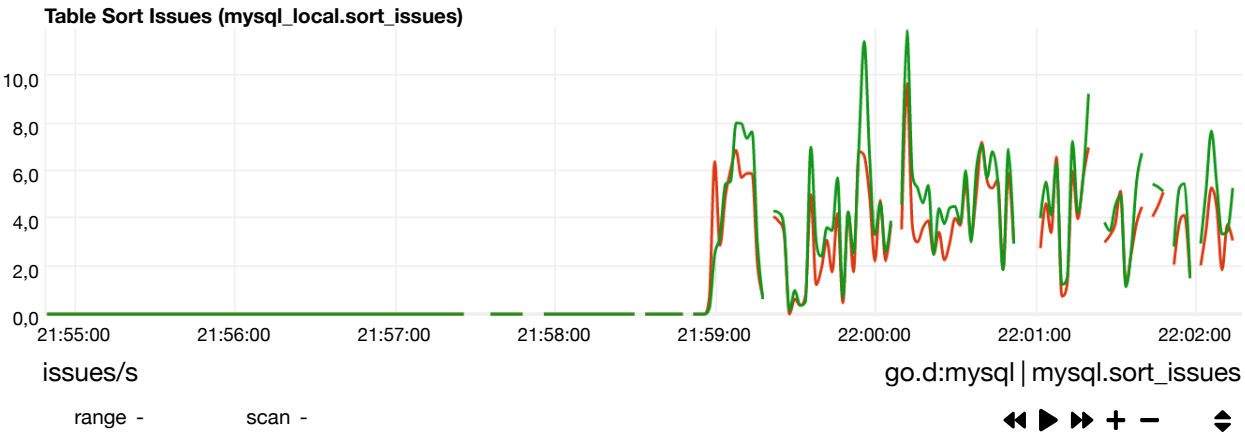
MySQL table locks counters:

- **immediate**, the number of times that a request for a table lock could be granted immediately.
- **waited**, the number of times that a request for a table lock could not be granted immediately and a wait was needed. If this is high and you have performance problems, you should first optimize your queries, and then either split your table or tables or use replication.

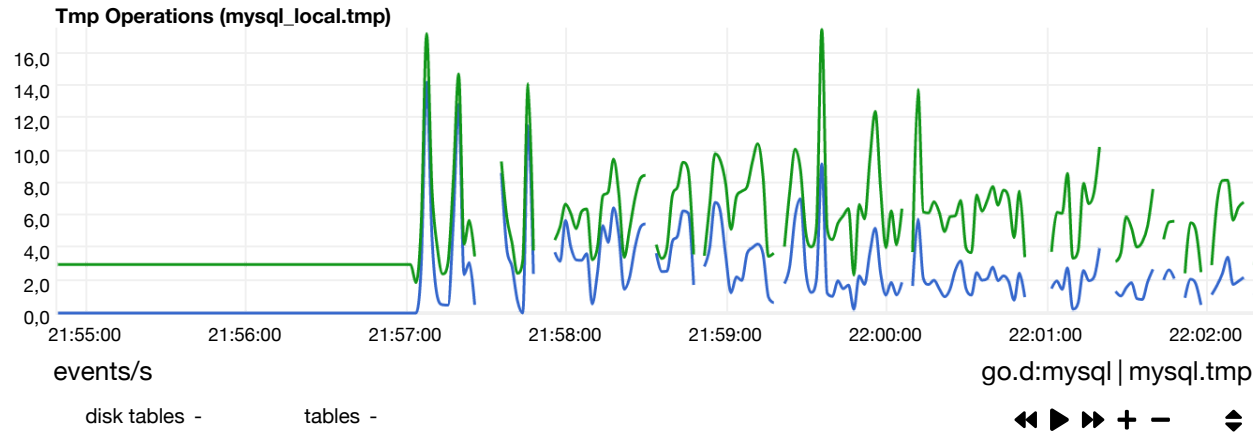


issues

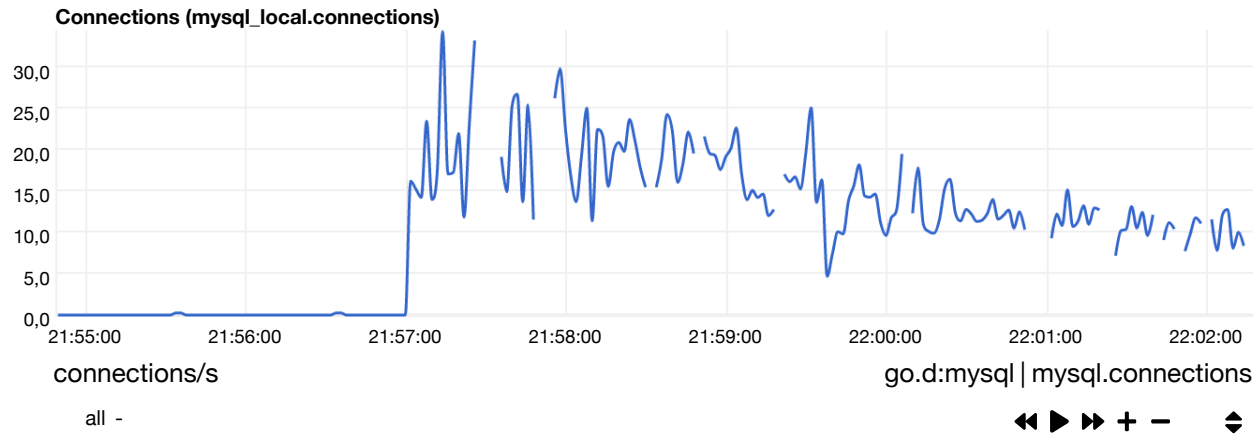


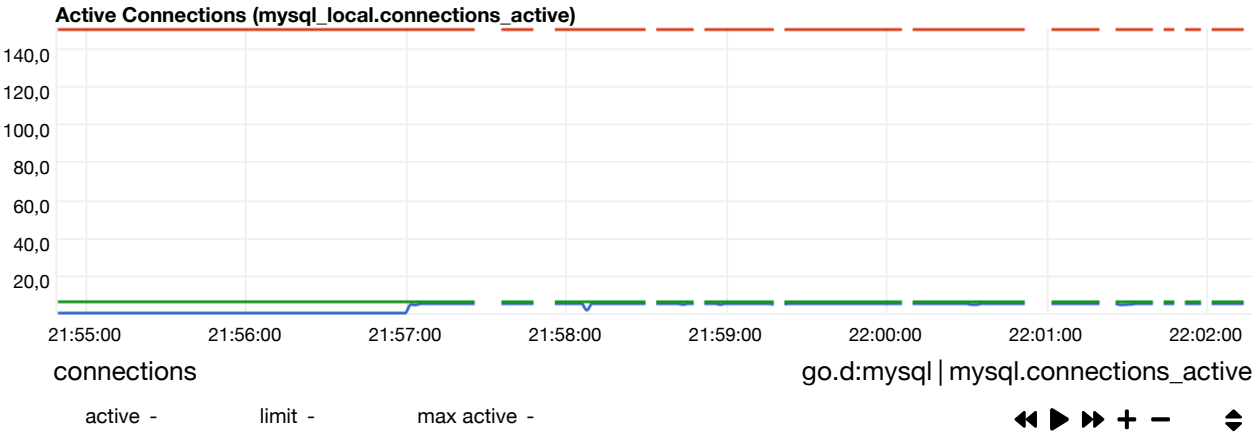


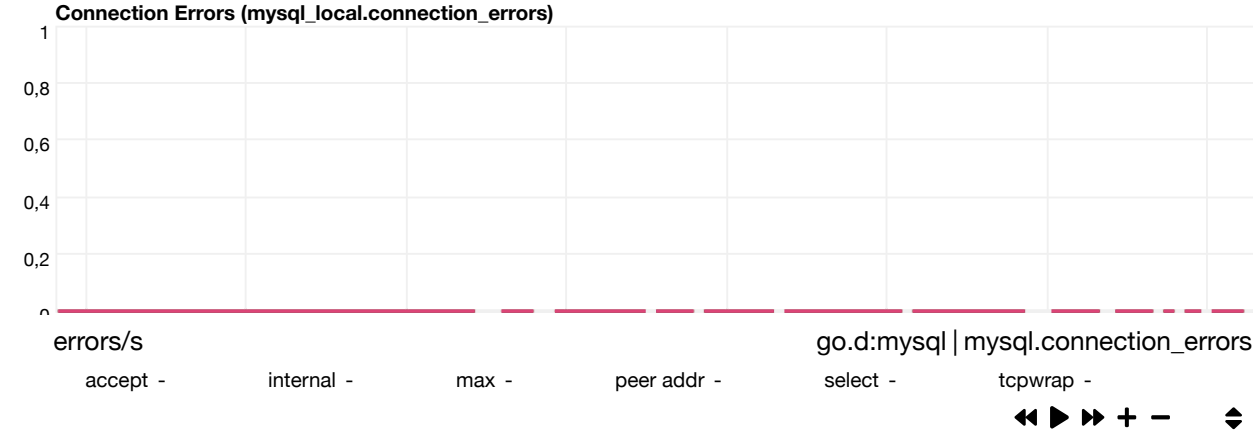
temporaries



connections

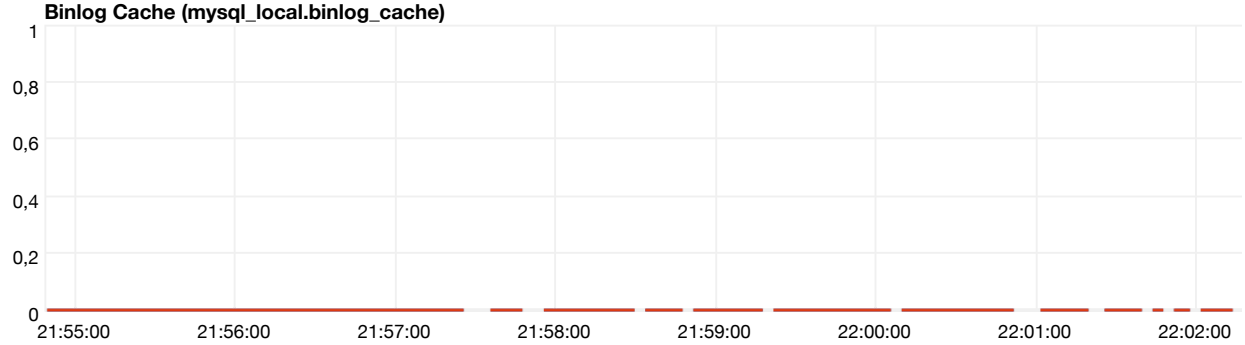






binlog

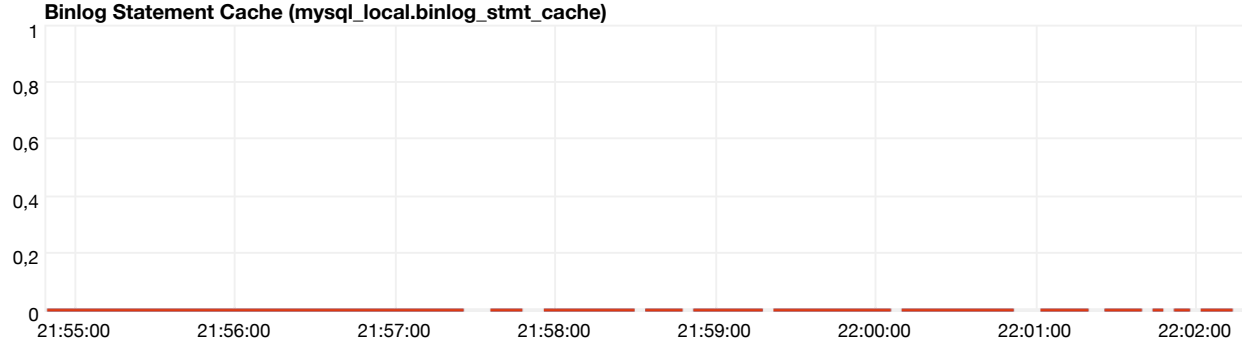
Binlog Cache (mysql_local.binlog_cache)



transactions/s go.d:mysql | mysql.binlog_cache

disk - all -

Binlog Statement Cache (mysql_local.binlog_stmt_cache)

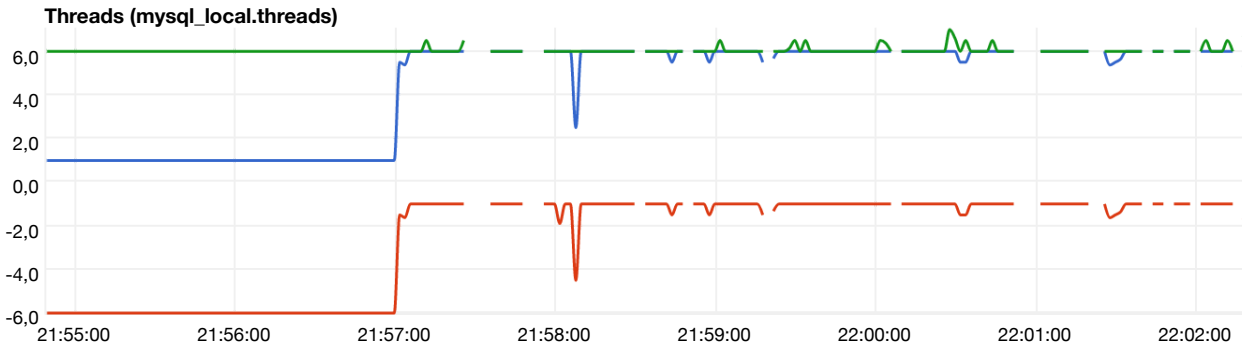


statements/s go.d:mysql | mysql.binlog_stmt_cache

disk - all -

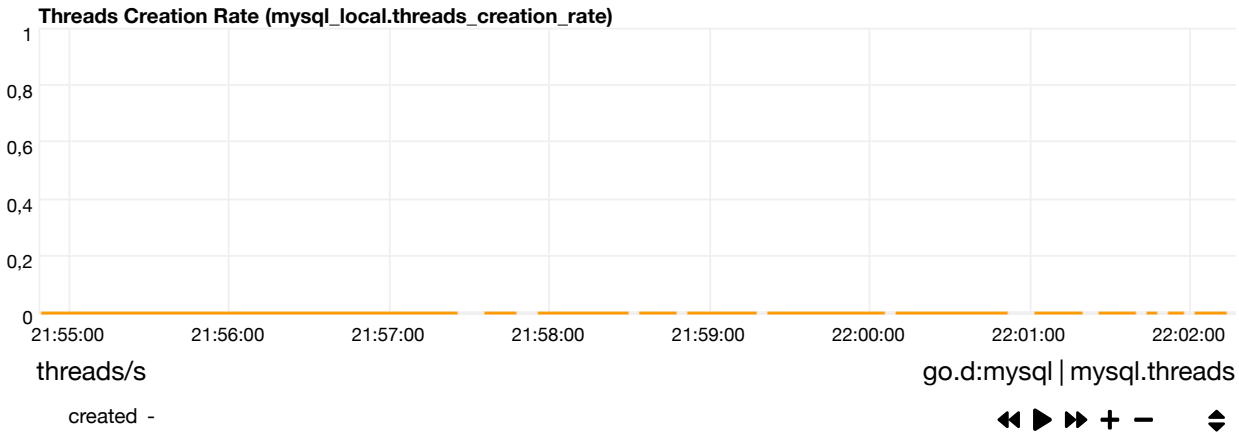
threads

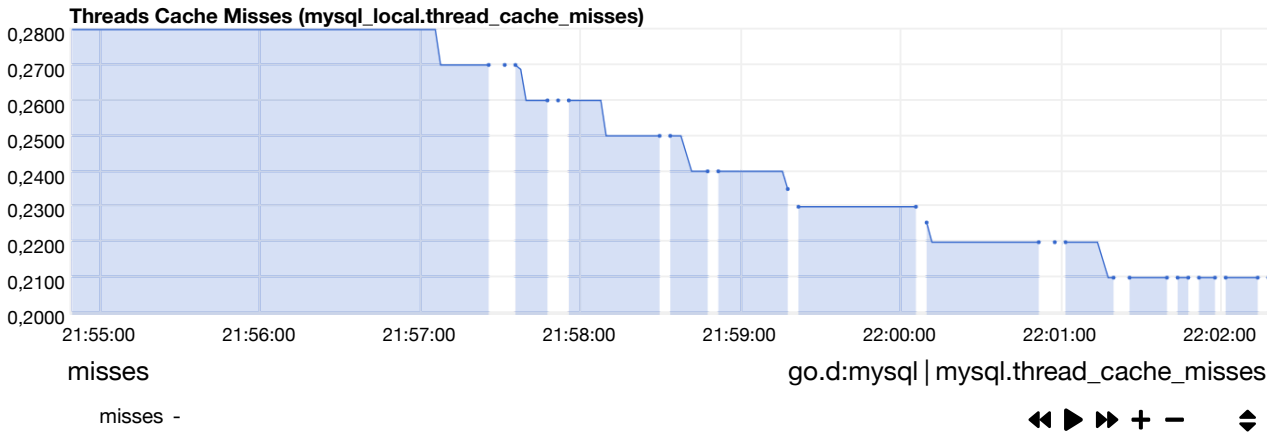
Threads (mysql_local.threads)



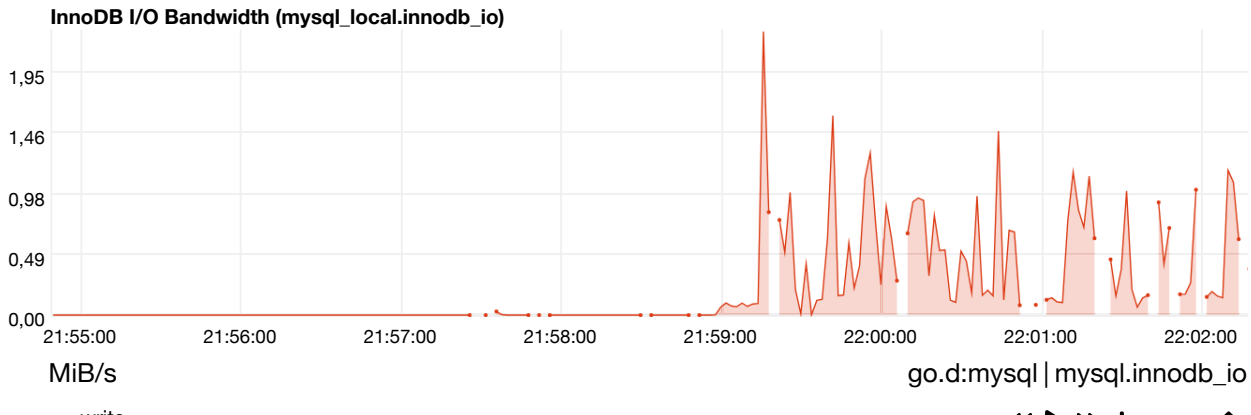
threads go.d:mysql | mysql.threads

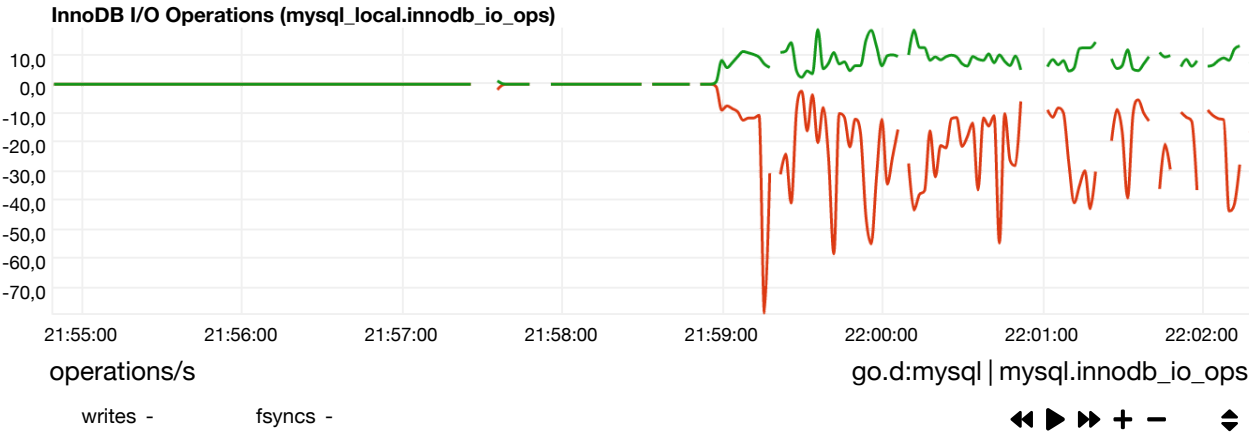
connected - cached - running -


Threads Creation Rate (mysql_local.threads_creation_rate)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
threads/s
go.d:mysql | mysql.threads
created -

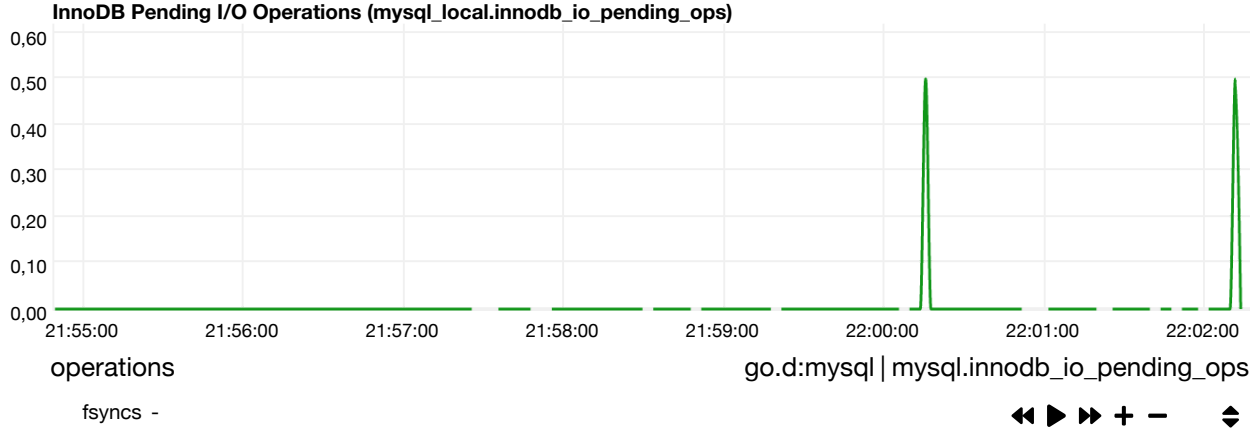

Threads Cache Misses (mysql_local.thread_cache_misses)
0,2800
0,2700
0,2600
0,2500
0,2400
0,2300
0,2200
0,2100
0,2000
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
misses
go.d:mysql | mysql.thread_cache_misses
misses -

innodb


InnoDB I/O Bandwidth (mysql_local.innodb_io)
1,95
1,46
0,98
0,49
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
go.d:mysql | mysql.innodb_io
write -

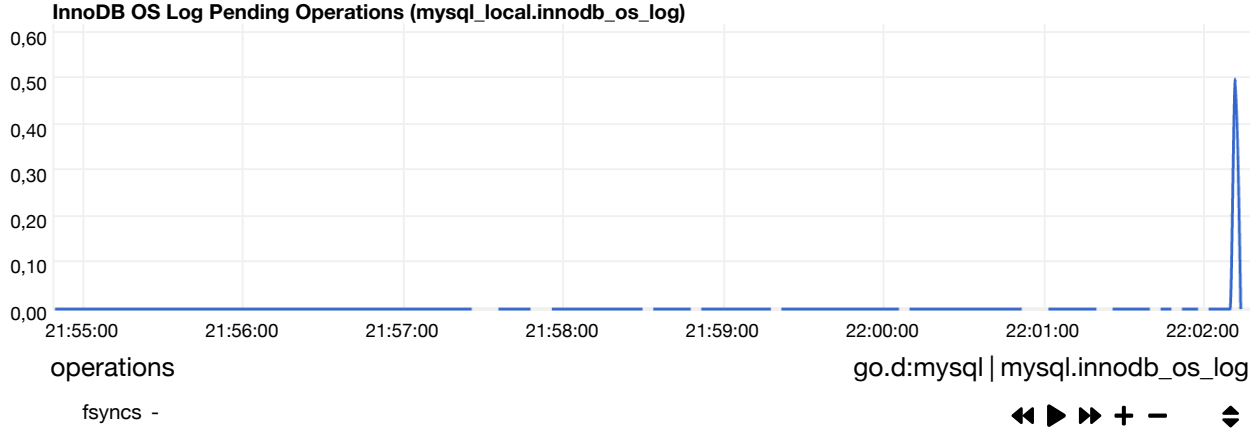

InnoDB I/O Operations (mysql_local.innodb_io_ops)
10,0
0,0
-10,0
-20,0
-30,0
-40,0
-50,0
-60,0
-70,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
go.d:mysql | mysql.innodb_io_ops
writes -
fsyncs -

InnoDB Pending I/O Operations (mysql_local.innodb_io_pending_ops)

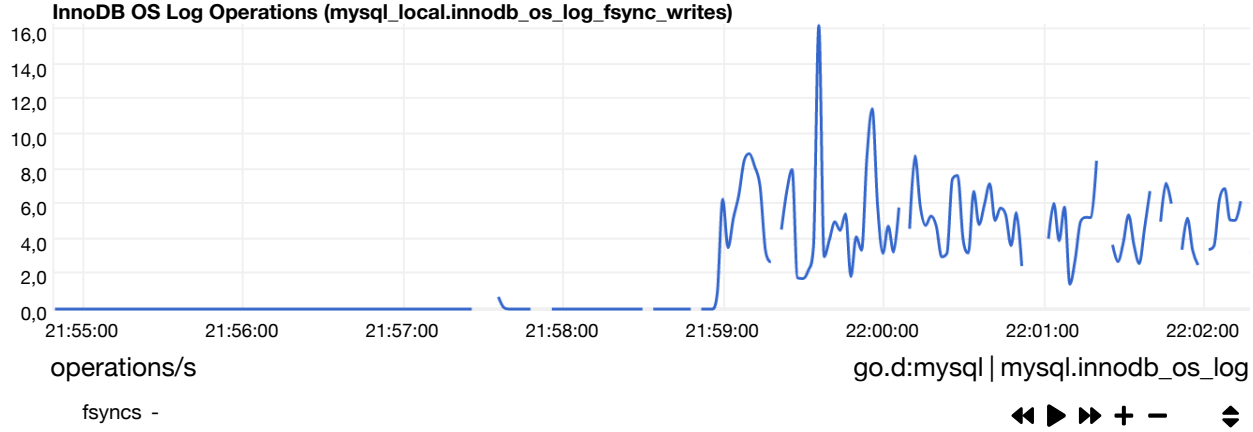


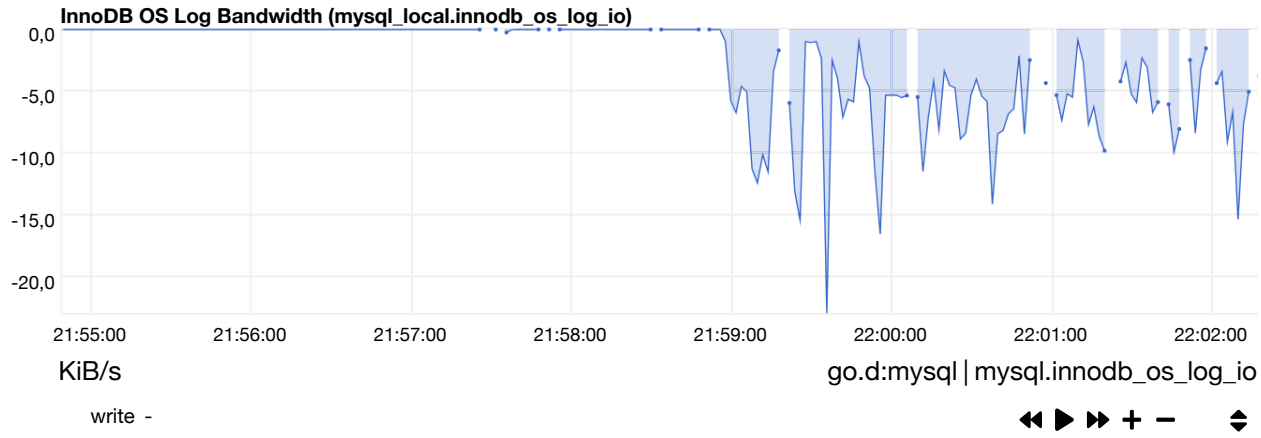
InnoDB Log Operations (mysql_local.innodb_log)

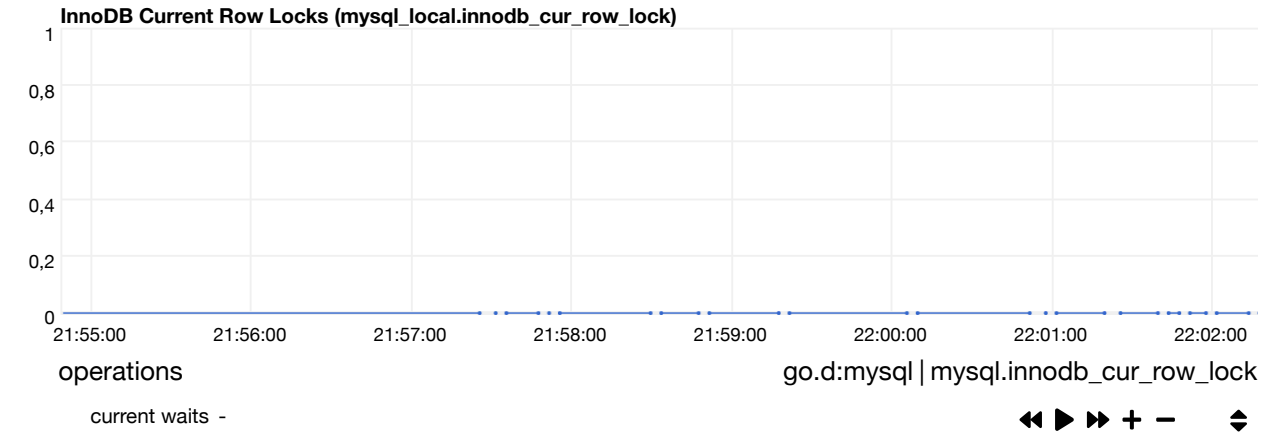
InnoDB OS Log Pending Operations (mysql_local.innodb_os_log)

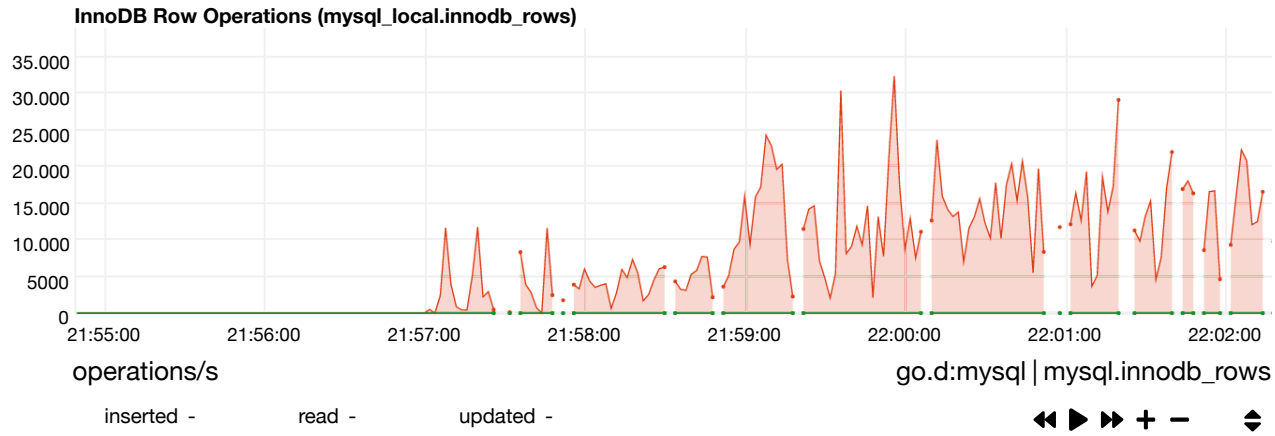


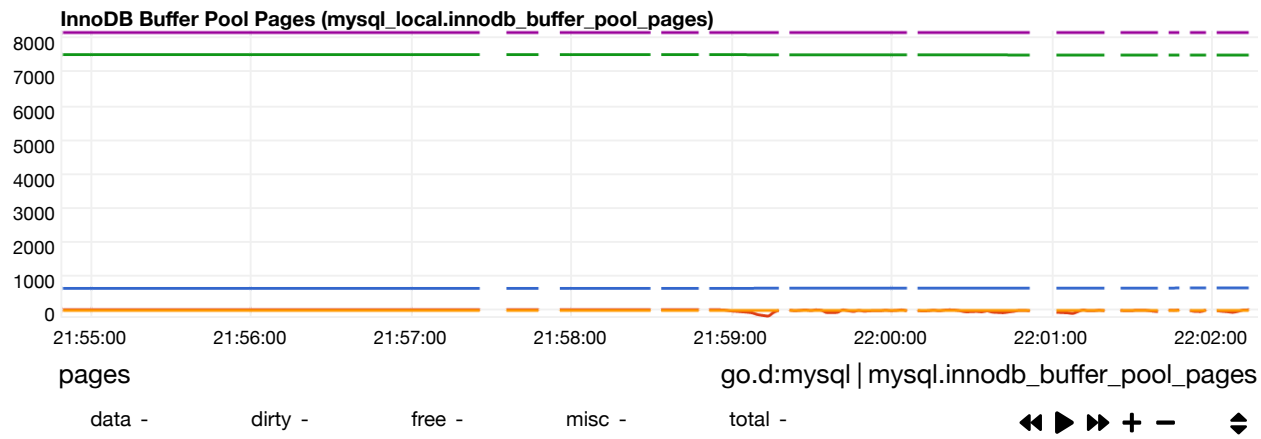
InnoDB OS Log Operations (mysql_local.innodb_os_log_fsync_writes)

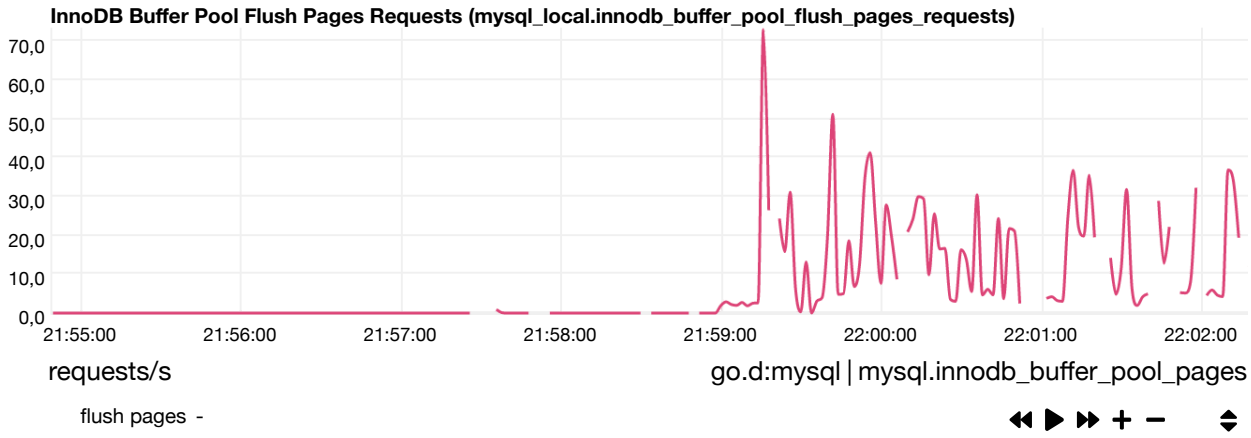


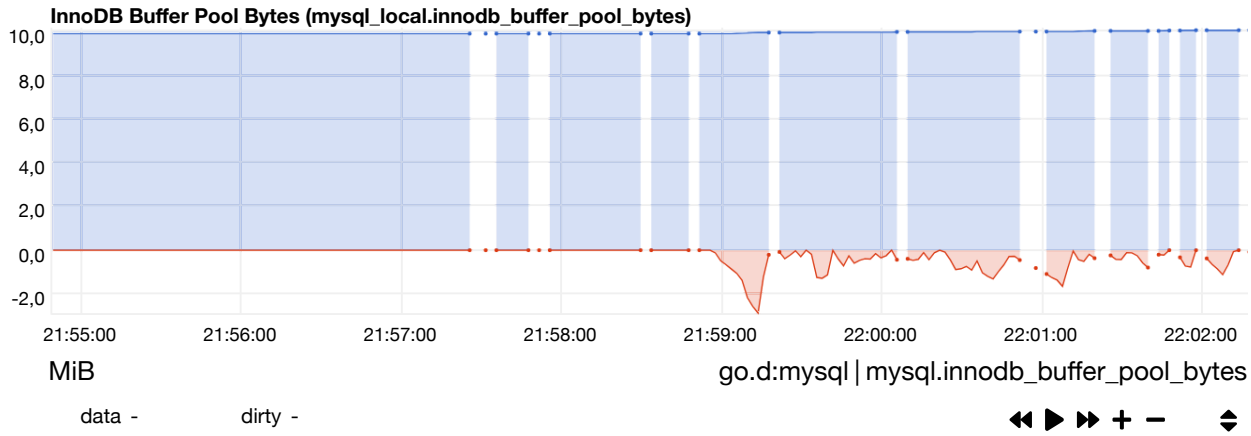

InnoDB OS Log Bandwidth (mysql_local.innodb_os_log_io)
0,0
-5,0
-10,0
-15,0
-20,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
KiB/s
go.d:mysql | mysql.innodb_os_log_io
write -

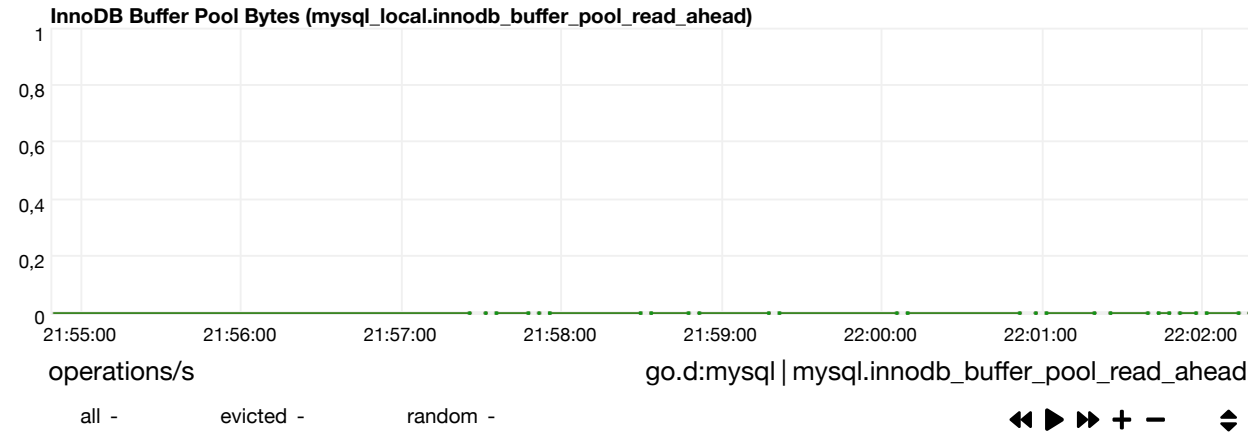

InnoDB Current Row Locks (mysql_local.innodb_cur_row_lock)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations
go.d:mysql | mysql.innodb_cur_row_lock
current waits -

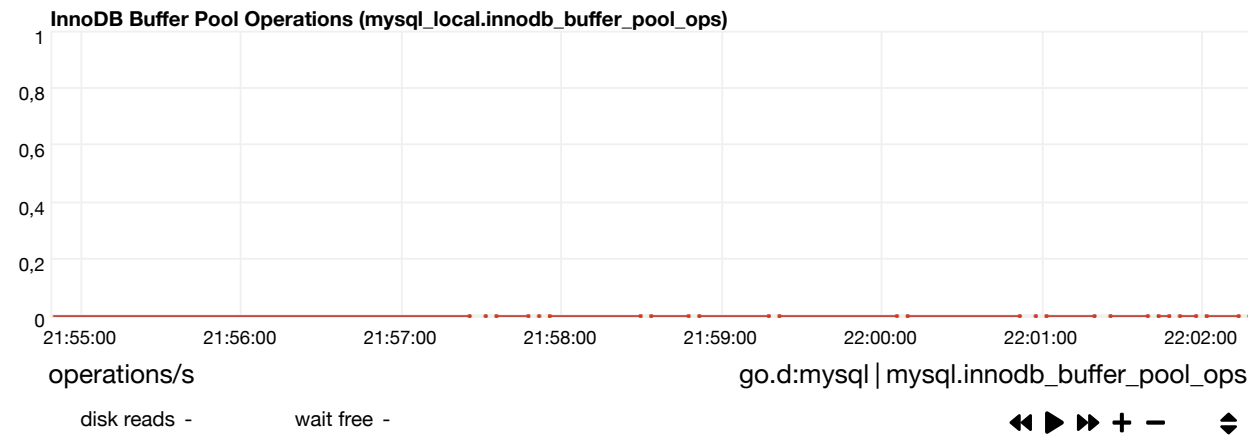

InnoDB Row Operations (mysql_local.innodb_rows)
35.000
30.000
25.000
20.000
15.000
10.000
5000
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
go.d:mysql | mysql.innodb_rows
inserted -
read -
updated -


InnoDB Buffer Pool Pages (mysql_local.innodb_buffer_pool_pages)
8000
7000
6000
5000
4000
3000
2000
1000
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
pages
go.d:mysql | mysql.innodb_buffer_pool_pages
data -
dirty -
free -
misc -
total -

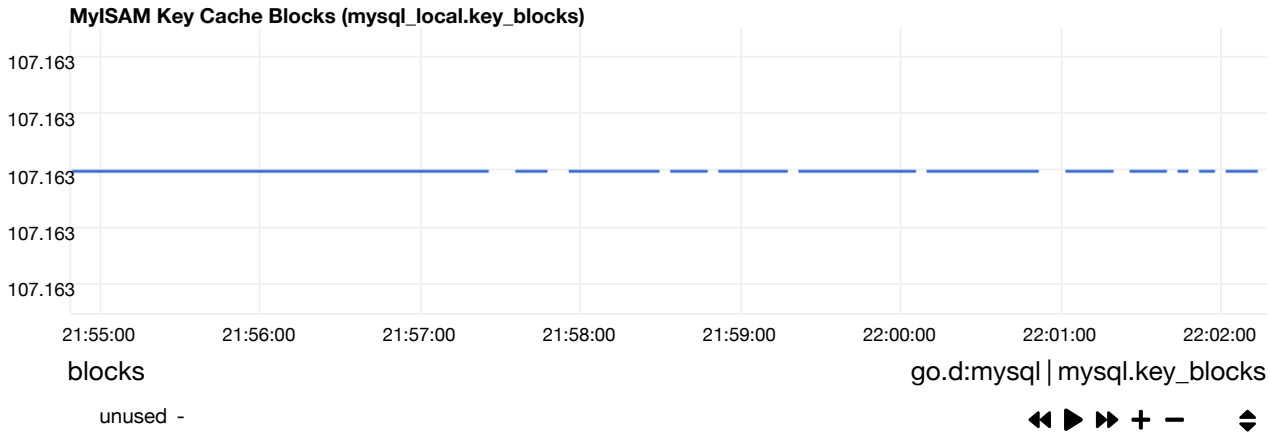

InnoDB Buffer Pool Flush Pages Requests (mysql_local.innodb_buffer_pool_flush_pages_requests)
70,0
60,0
50,0
40,0
30,0
20,0
10,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
requests/s
go.d:mysql | mysql.innodb_buffer_pool_pages
flush pages -

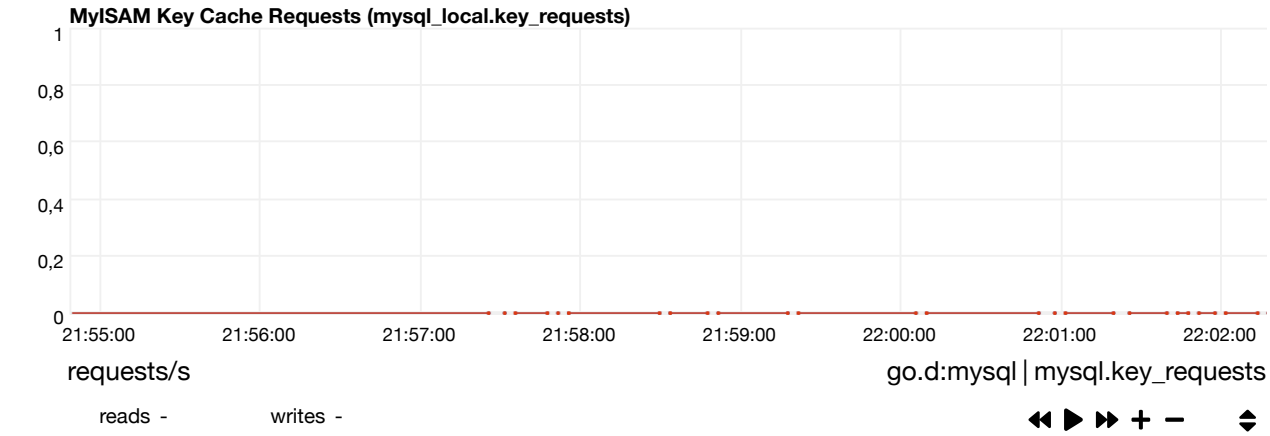

InnoDB Buffer Pool Bytes (mysql_local.innodb_buffer_pool_bytes)
10,0
8,0
6,0
4,0
2,0
0,0
-2,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB
go.d:mysql | mysql.innodb_buffer_pool_bytes
data -
dirty -

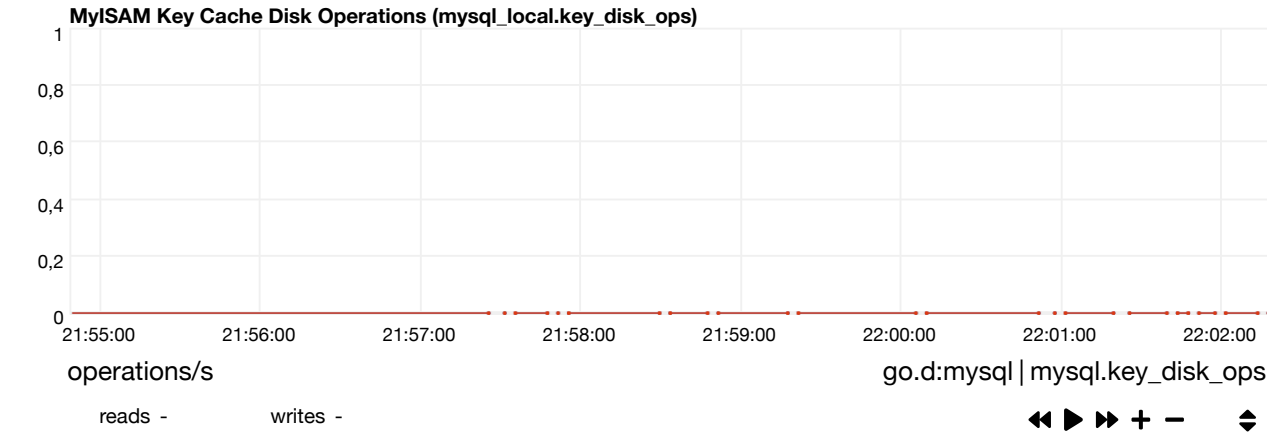

InnoDB Buffer Pool Bytes (mysql_local.innodb_buffer_pool_read_ahead)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
go.d:mysql | mysql.innodb_buffer_pool_read_ahead
all -
evicted -
random -


InnoDB Buffer Pool Operations (mysql_local.innodb_buffer_pool_ops)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
go.d:mysql | mysql.innodb_buffer_pool_ops
disk reads -
wait free -

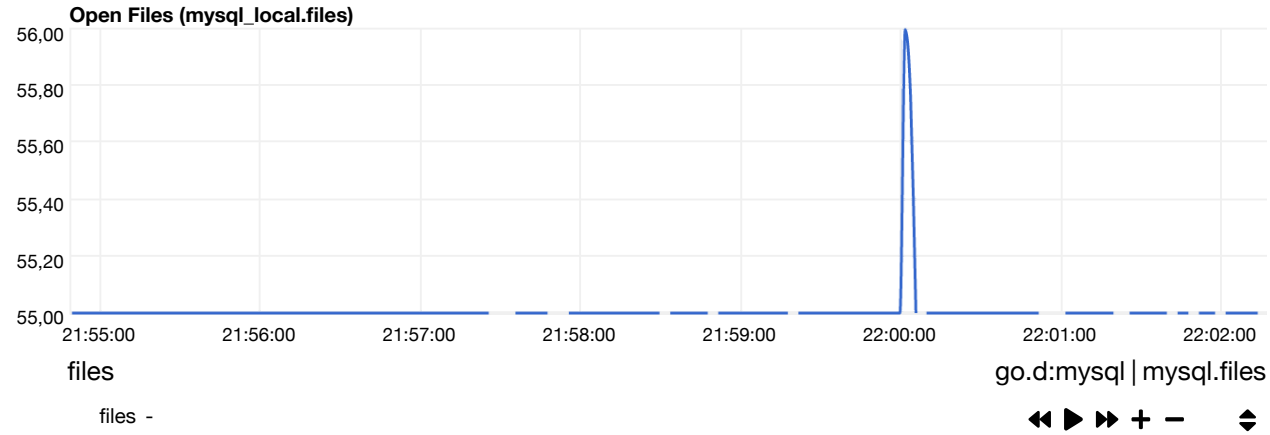
myisam

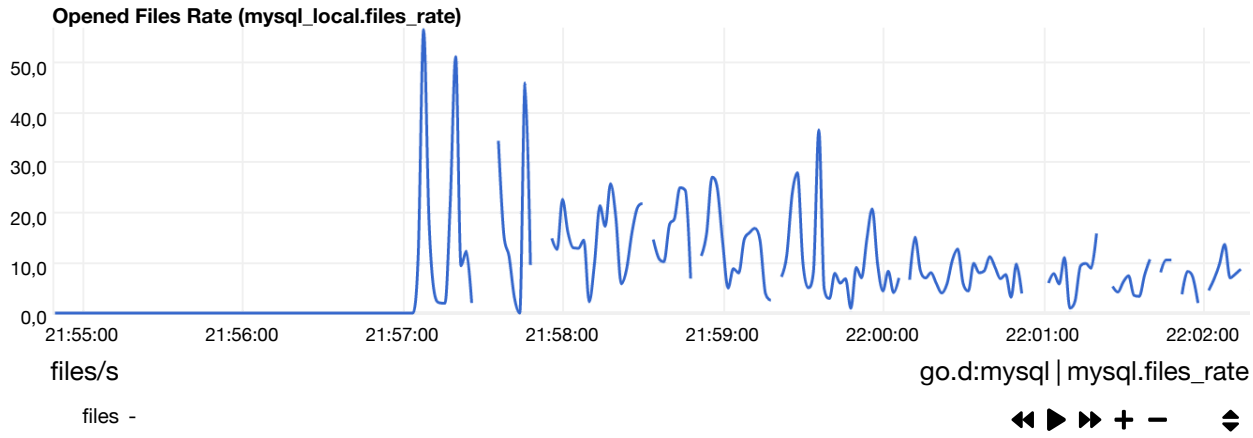

MyISAM Key Cache Blocks (mysql_local.key_blocks)
107.163
107.163
107.163
107.163
107.163
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
blocks
go.d:mysql | mysql.key_blocks
unused -


MyISAM Key Cache Requests (mysql_local.key_requests)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
requests/s
go.d:mysql | mysql.key_requests
reads -
writes -

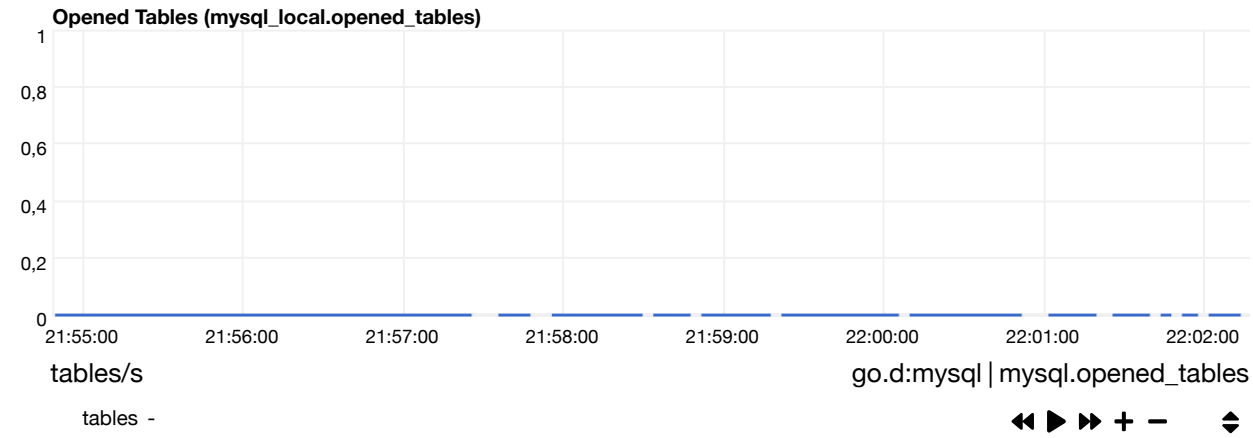

MyISAM Key Cache Disk Operations (mysql_local.key_disk_ops)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
go.d:mysql | mysql.key_disk_ops
reads -
writes -

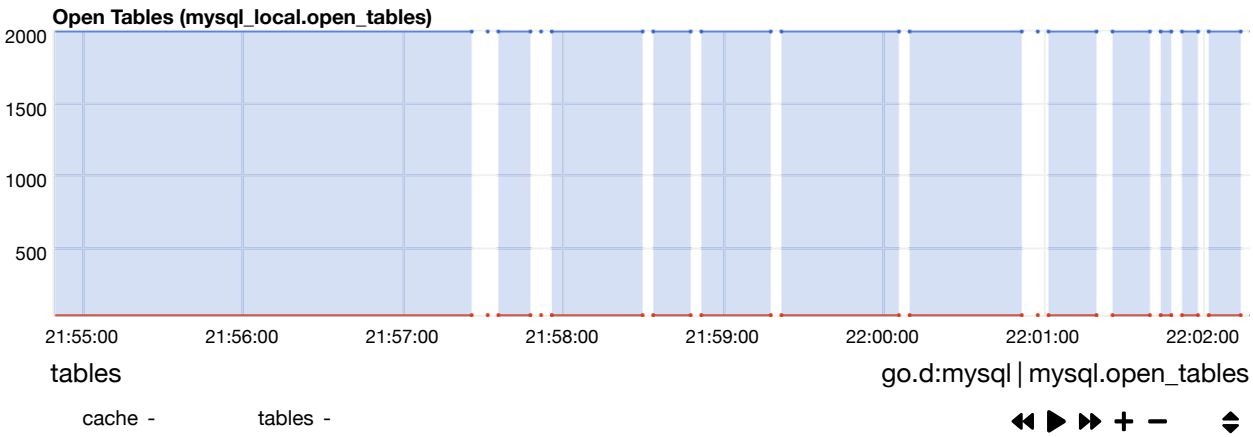
files


Open Files (mysql_local.files)
56,00
55,80
55,60
55,40
55,20
55,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
files
go.d:mysql | mysql.files
files -



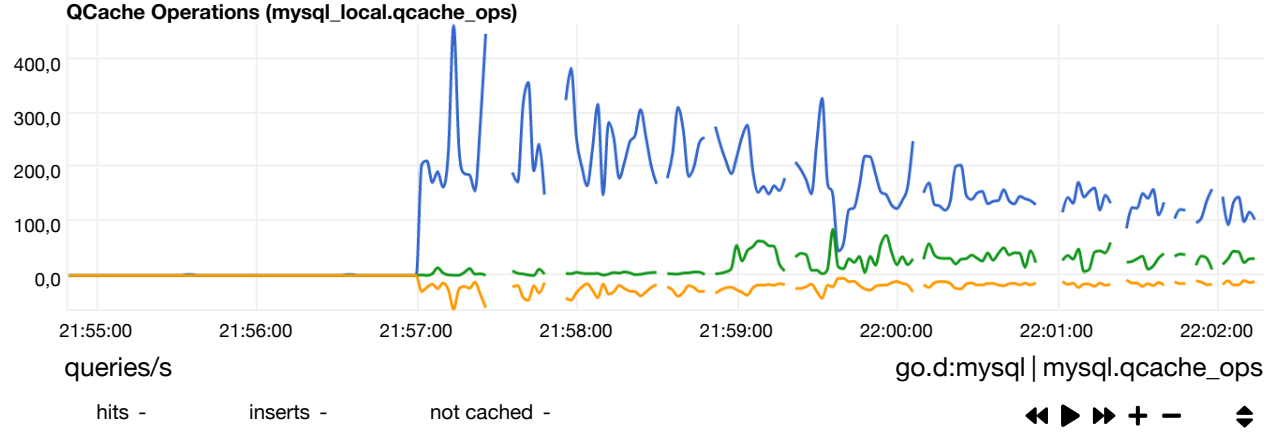
open tables





qcache

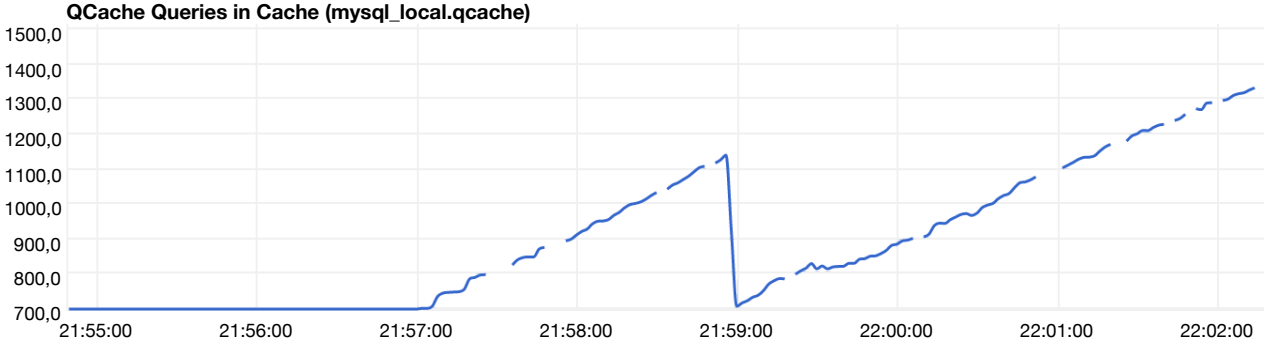
QCache Operations (mysql_local.qcache_ops)



queries/s go.d:mysql | mysql.qcache_ops

hits - inserts - not cached -

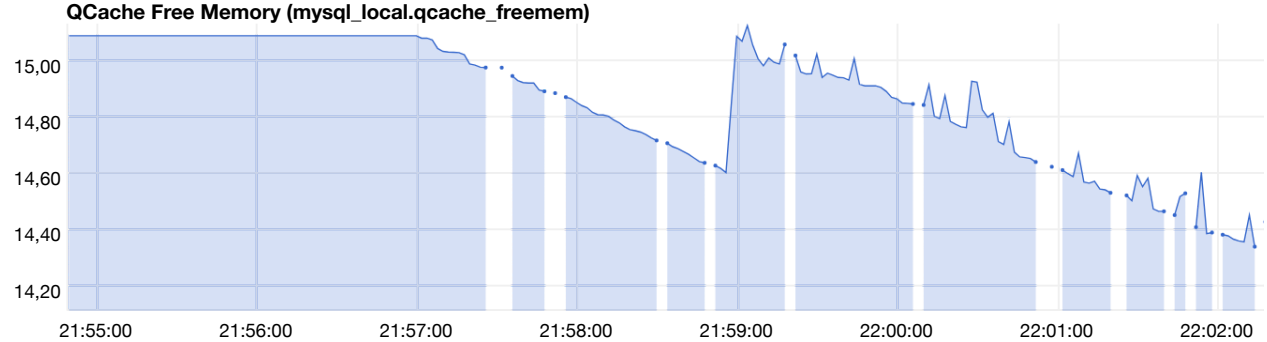
QCache Queries in Cache (mysql_local.qcache)



queries go.d:mysql | mysql.qcache

queries -

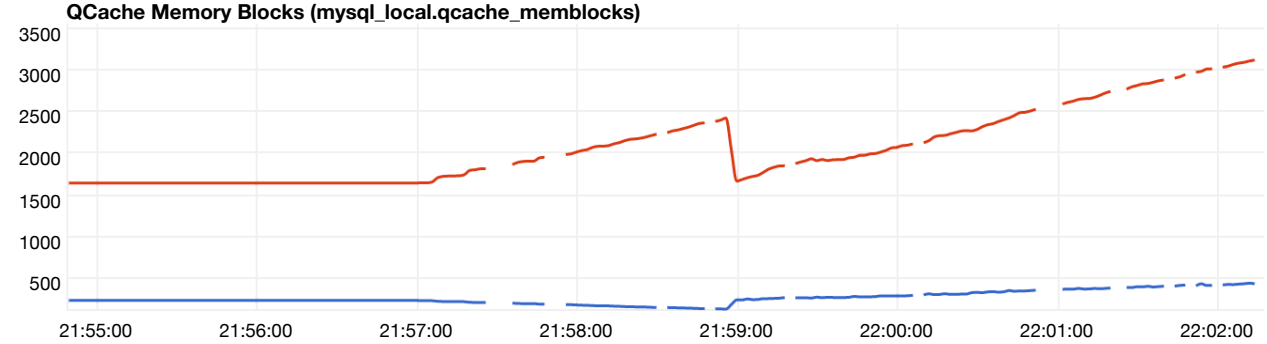
QCache Free Memory (mysql_local.qcache_freemem)



MiB go.d:mysql | mysql.qcache_freemem

free -

QCache Memory Blocks (mysql_local.qcache_memblocks)



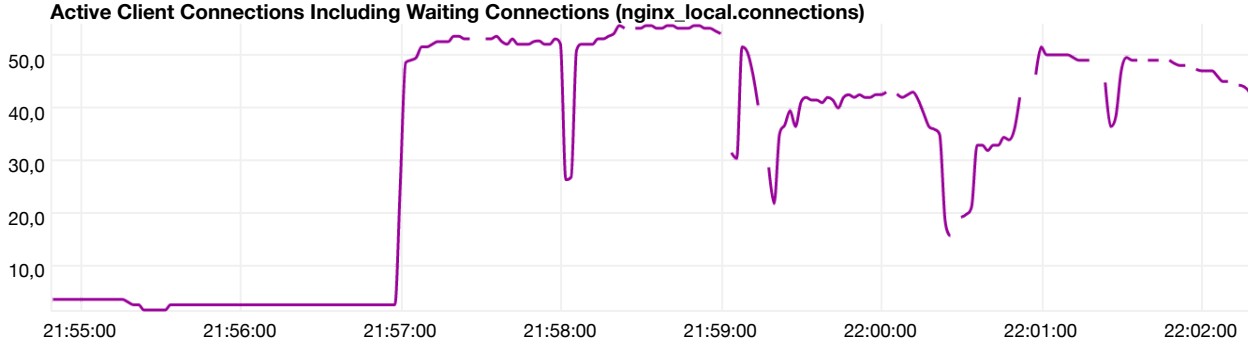
blocks go.d:mysql | mysql.qcache_memblocks

free - total -

nginx local

connections

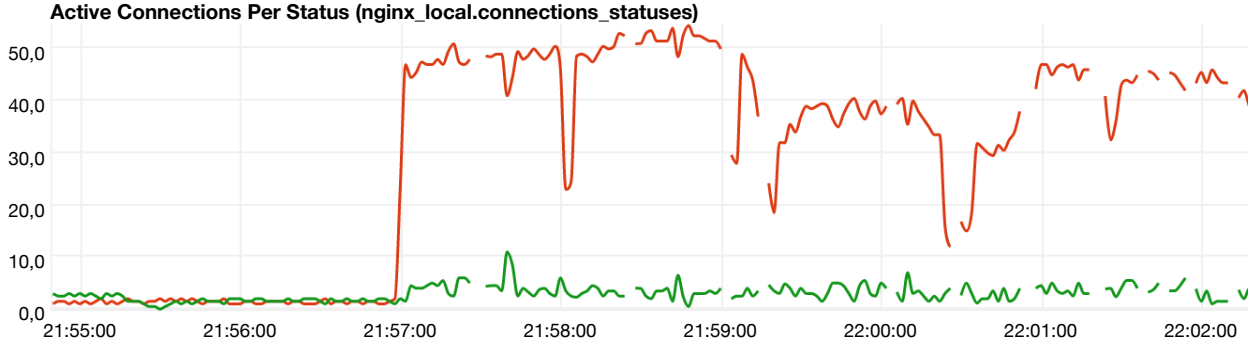
Active Client Connections Including Waiting Connections (nginx_local.connections)



connections go.d:nginx | nginx.connections

active -

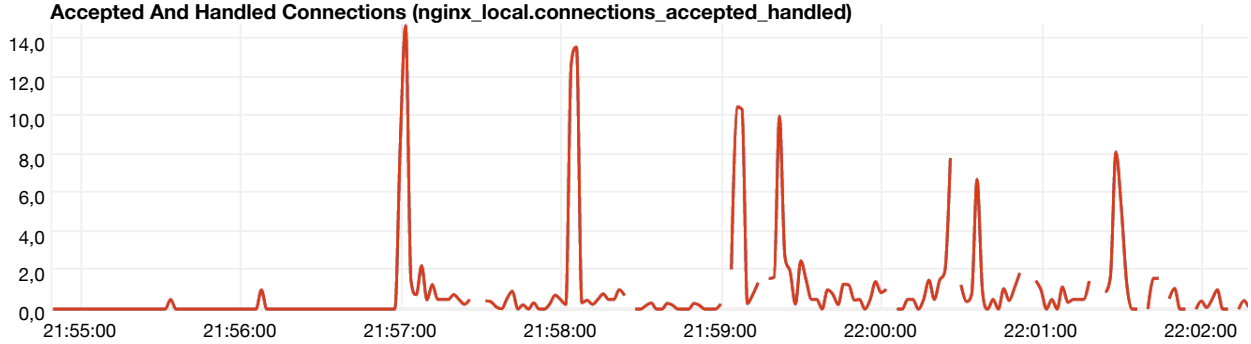
Active Connections Per Status (nginx_local.connections_statuses)



connections go.d:nginx | nginx.connections_status

writing - idle -

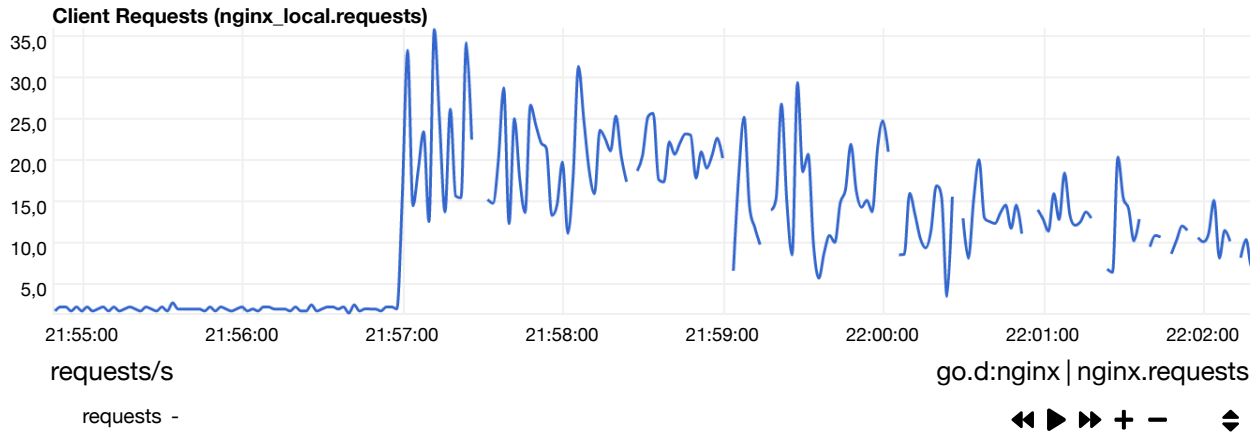
Accepted And Handled Connections (nginx_local.connections_accepted_handled)



connections/s go.d:nginx | nginx.connections_accepted_handled

accepted - handled -

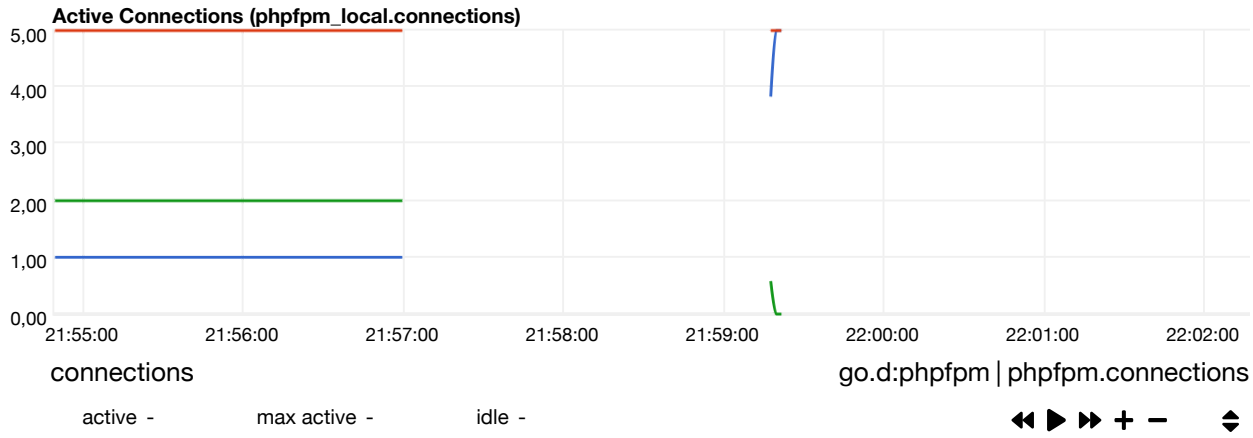
requests



PHP-FPM local

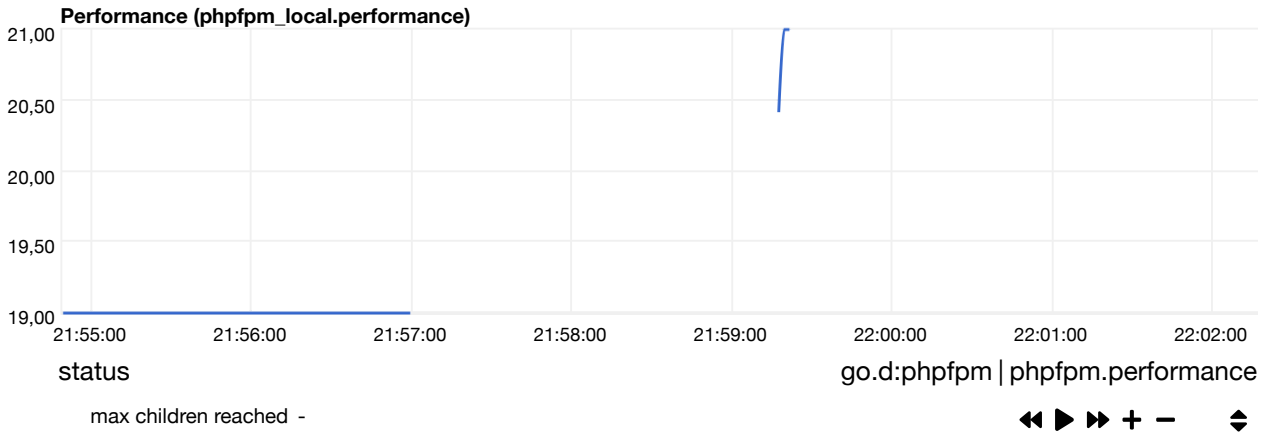
Performance metrics for **PHP-FPM**, an alternative FastCGI implementation for PHP.

active connections

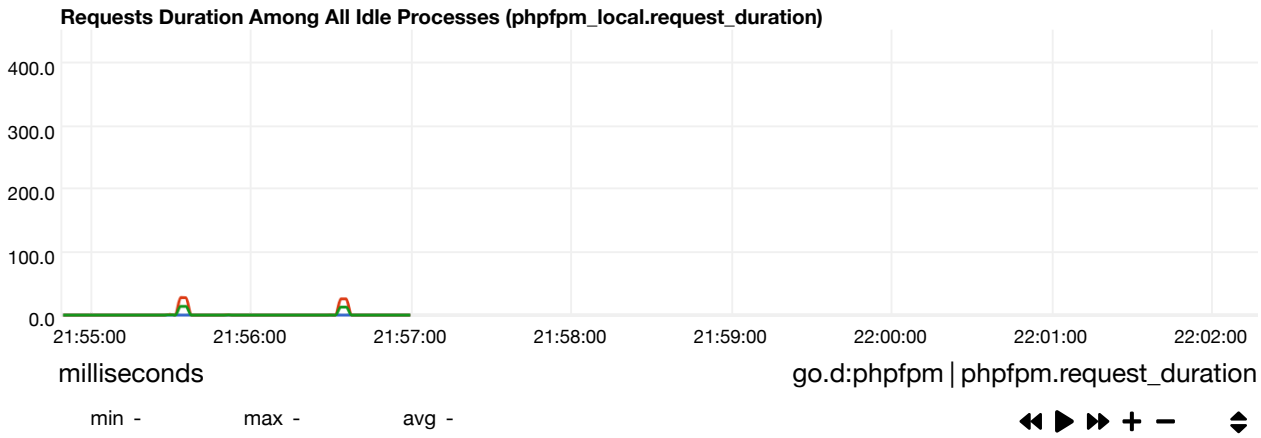


requests

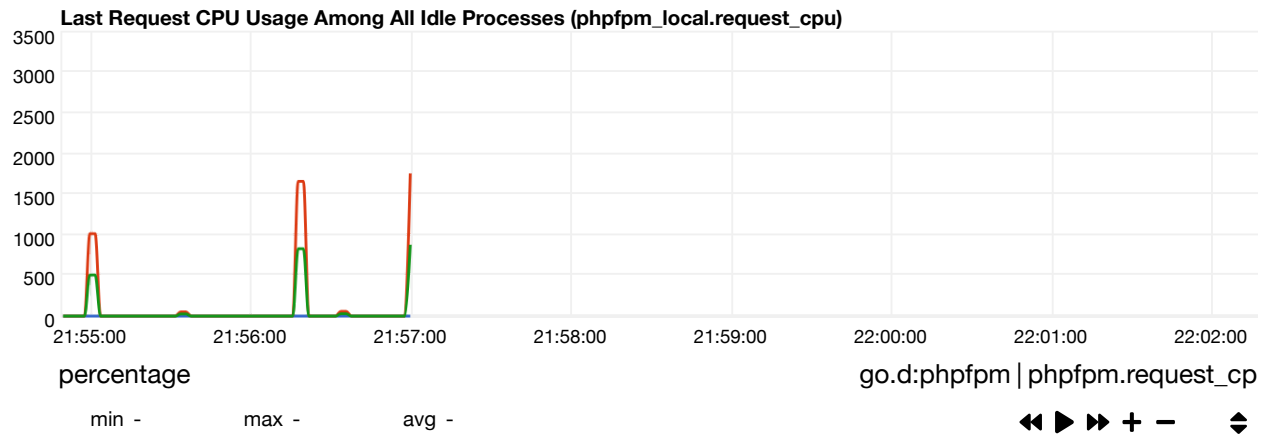
performance



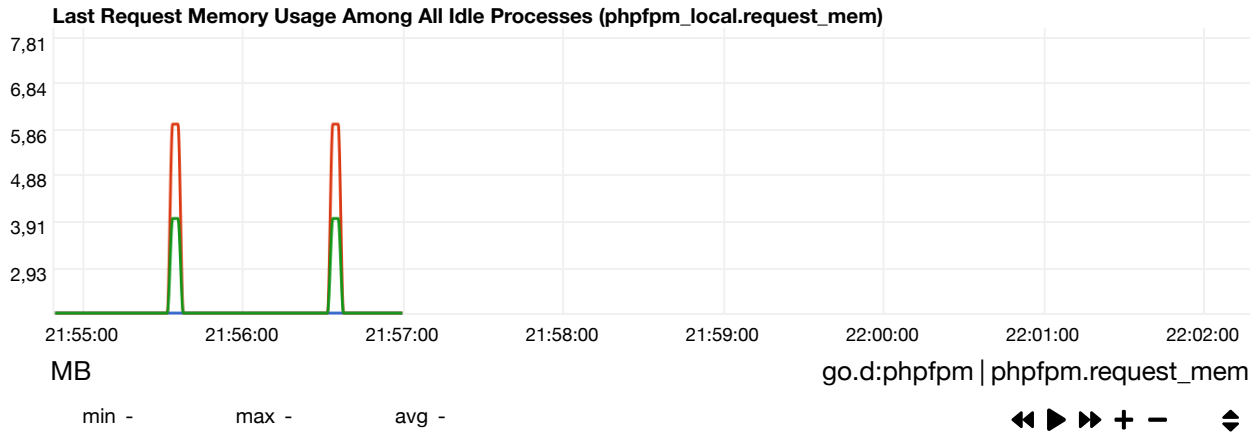
request duration



request CPU



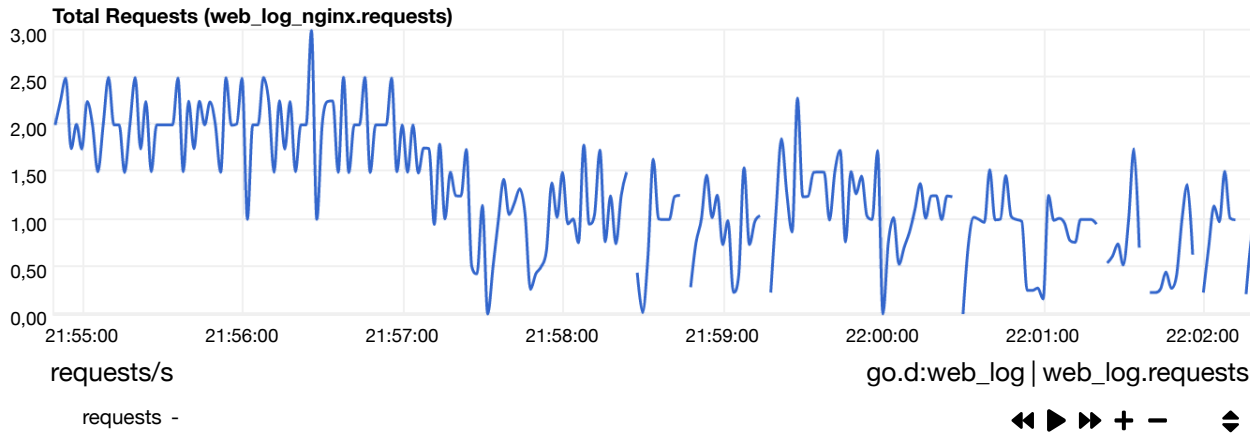
request memory

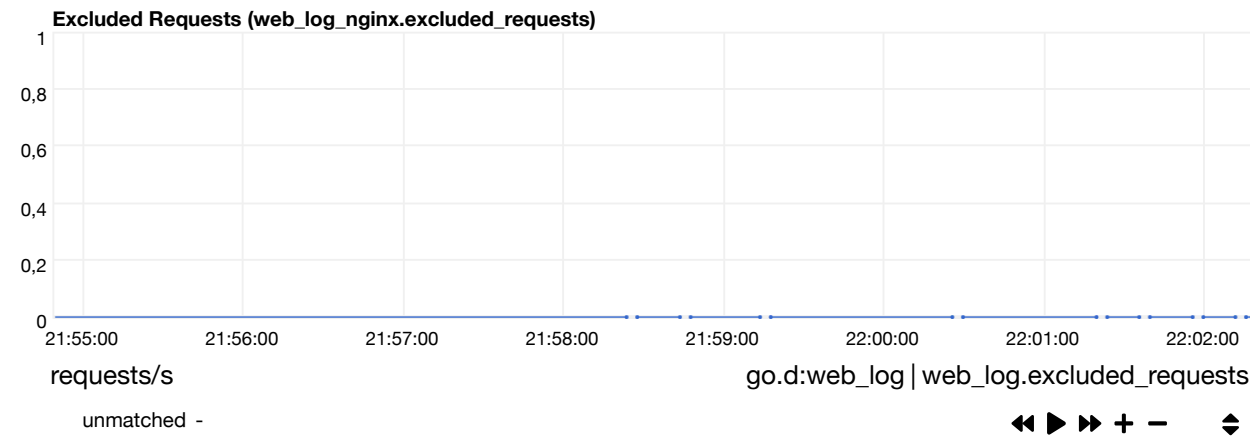


web log nginx

Information extracted from a server log file. `web_log` plugin incrementally parses the server log file to provide, in real-time, a break down of key server performance metrics. For web servers, an extended log file format may optionally be used (for `nginx` and `apache`) offering timing information and bandwidth for both requests and responses. `web_log` plugin may also be configured to provide a break down of requests per URL pattern (check `/etc/netdata/python.d/web_log.conf` (https://github.com/netdata/netdata/blob/master/collectors/python.d.plugin/web_log/web_log.conf)).

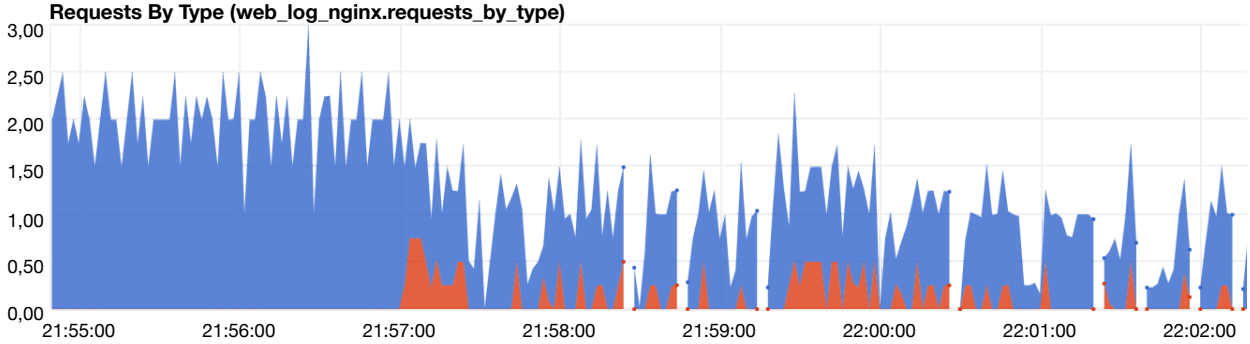
requests





Web server responses by type. `success` includes **1xx**, **2xx**, **304** and **401**, `error` includes **5xx**, `redirect` includes **3xx** except **304**, `bad` includes **4xx** except **401**, `other` are all the other responses.

Requests By Type (web_log_nginx.requests_by_type)

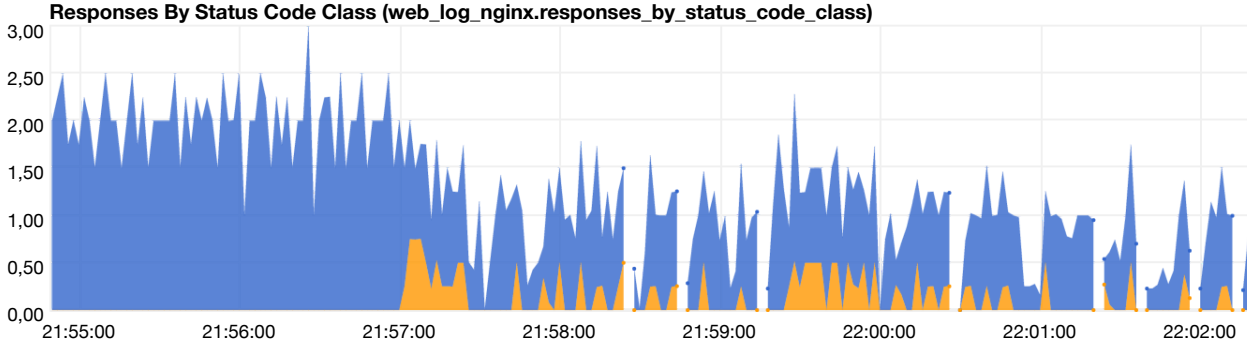


requests/s go.d:web_log | web_log.type_requests

success - bad -

responses

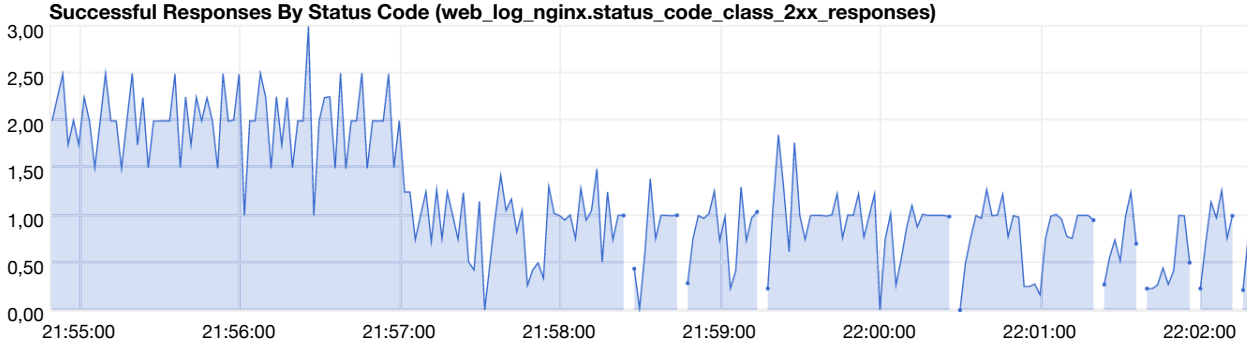
Responses By Status Code Class (web_log_nginx.responses_by_status_code_class)



responses/s go.d:web_log | web_log.status_code_class_responses

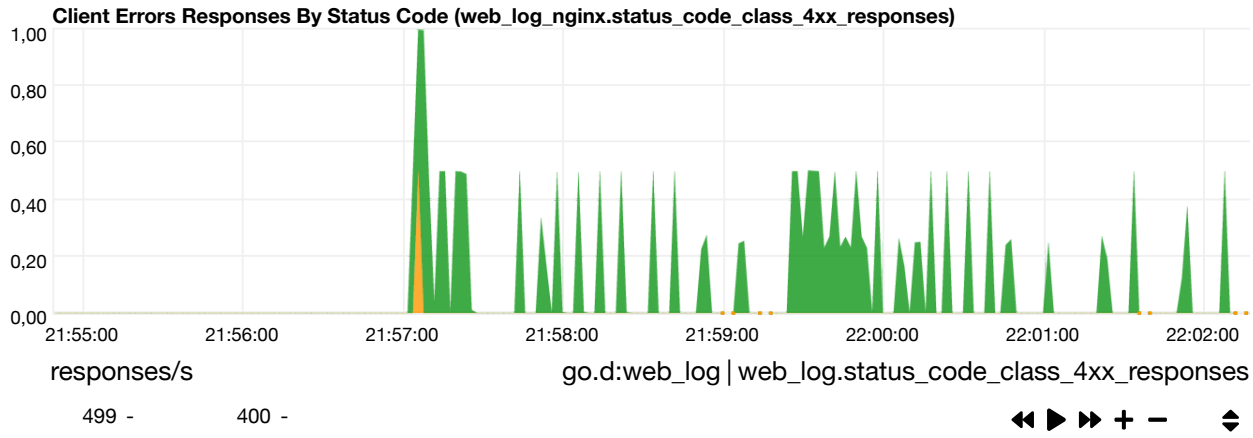
2xx - 4xx -

Successful Responses By Status Code (web_log_nginx.status_code_class_2xx_responses)



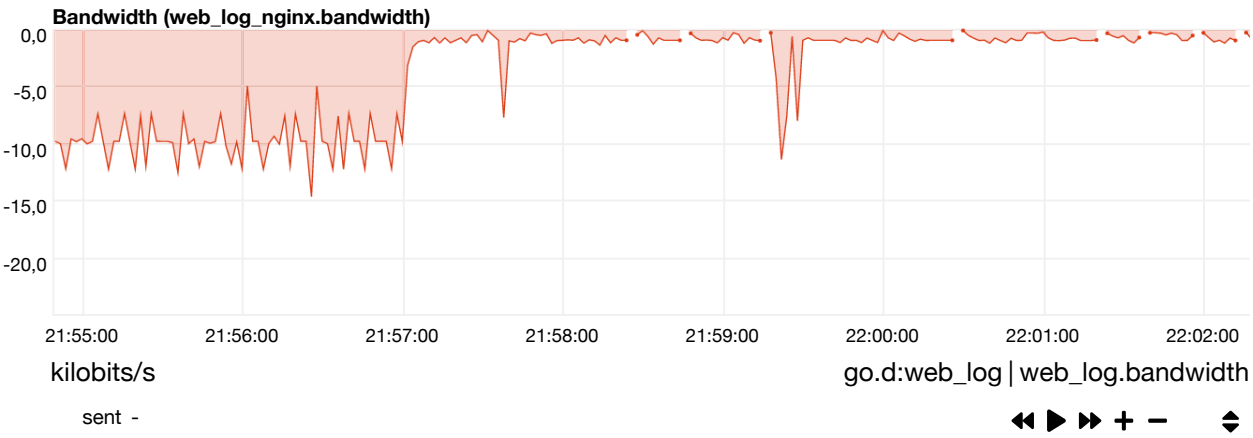
responses/s go.d:web_log | web_log.status_code_class_2xx_responses

200 -

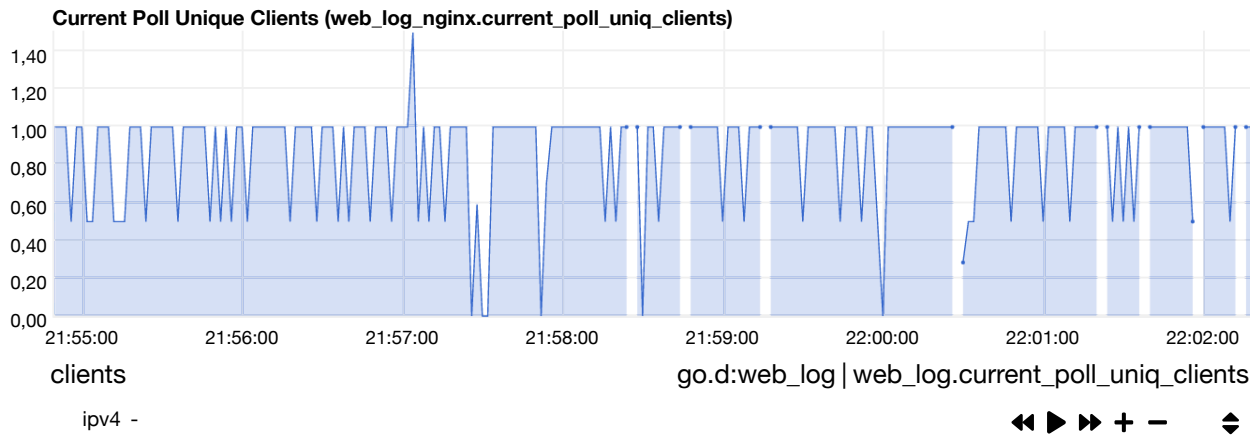


bandwidth

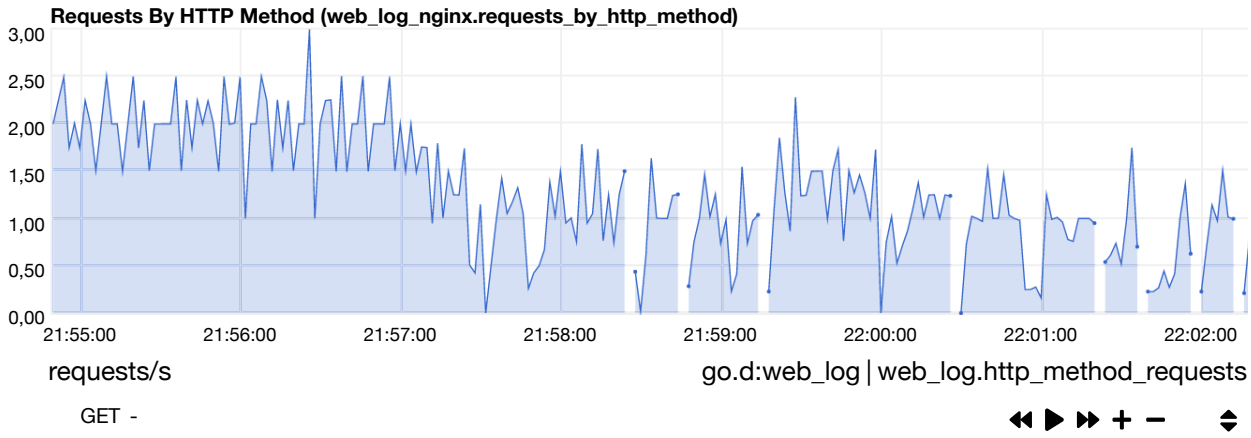
Bandwidth of requests (`received`) and responses (`sent`). `received` requires an extended log format (without it, the web server log does not have this information). This chart may present unusual spikes, since the bandwidth is accounted at the time the log line is saved by the web server, even if the time needed to serve it spans across a longer duration. We suggest to use QoS (e.g. FireQOS (http://firehol.org/#fireqos)) for accurate accounting of the web server bandwidth.



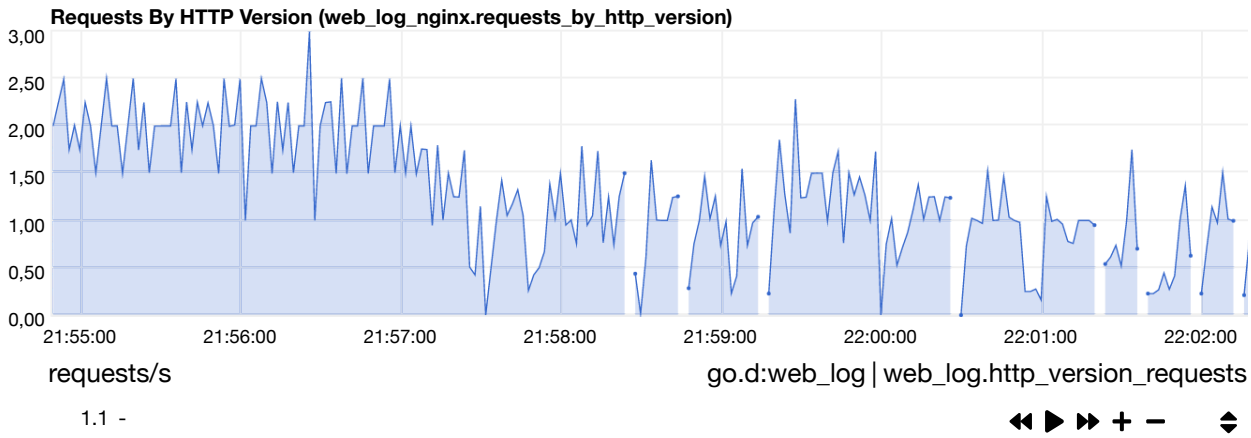
client



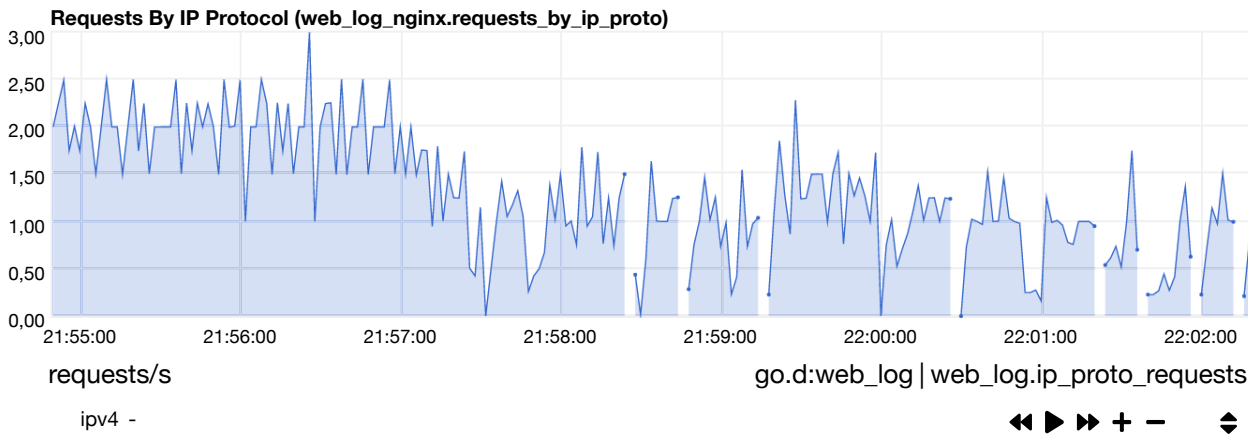
http method



http version



ip proto

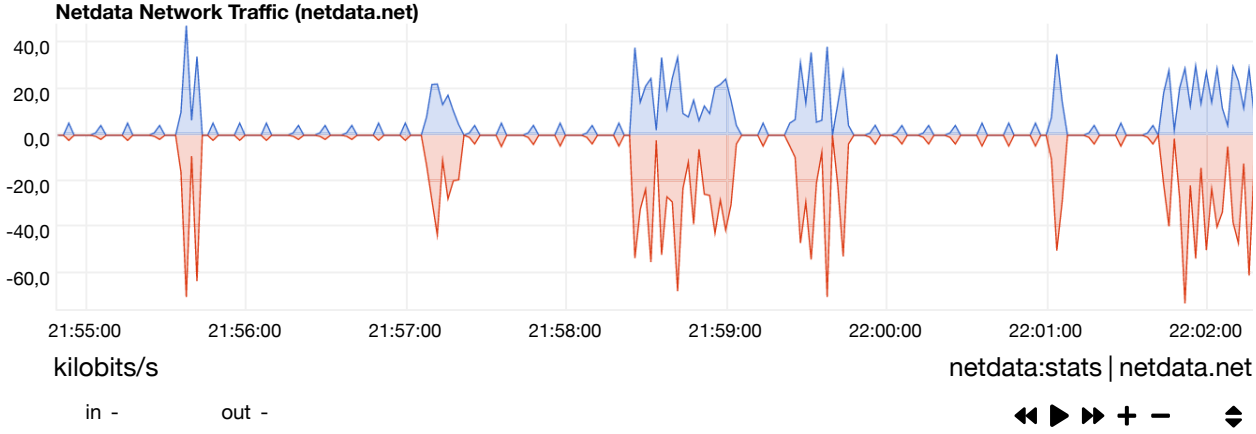


Netdata Monitoring

Performance metrics for the operation of netdata itself and its plugins.

netdata

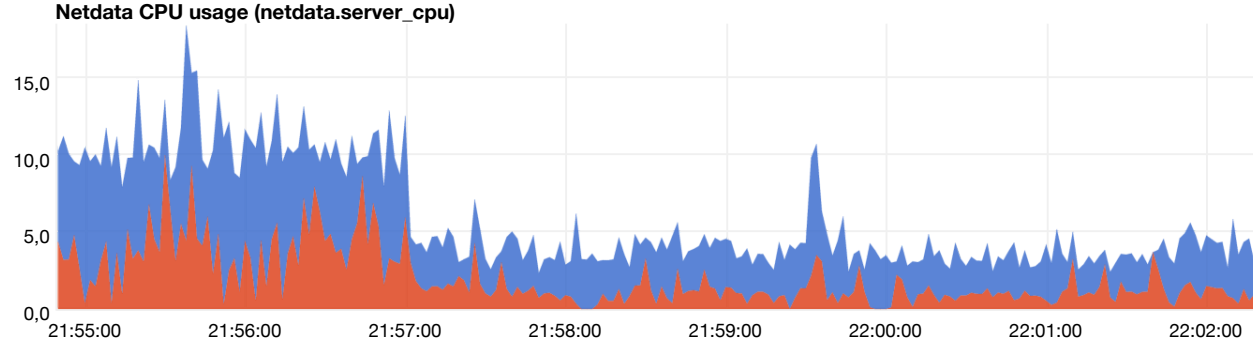
Netdata Network Traffic (netdata.net)



kilobits/s netdata:stats | netdata.net

in - out -

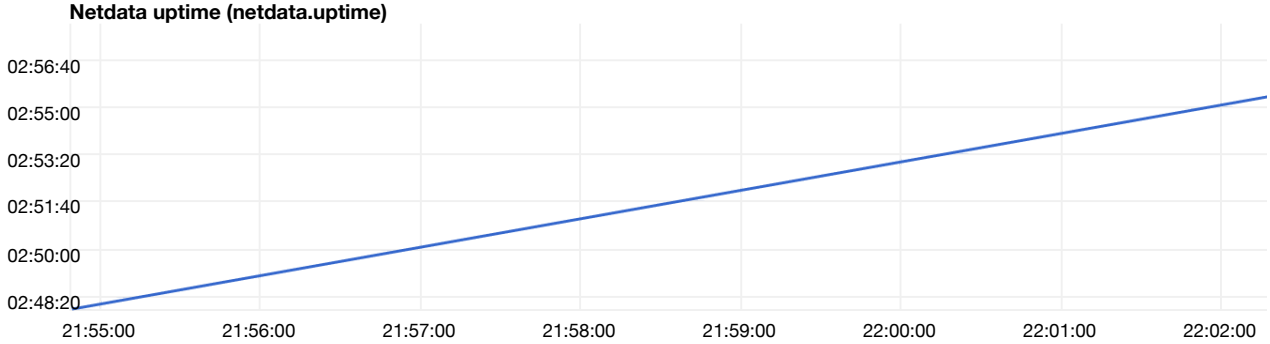
Netdata CPU usage (netdata.server_cpu)



milliseconds/s netdata:stats | netdata.server_cpu

user - system -

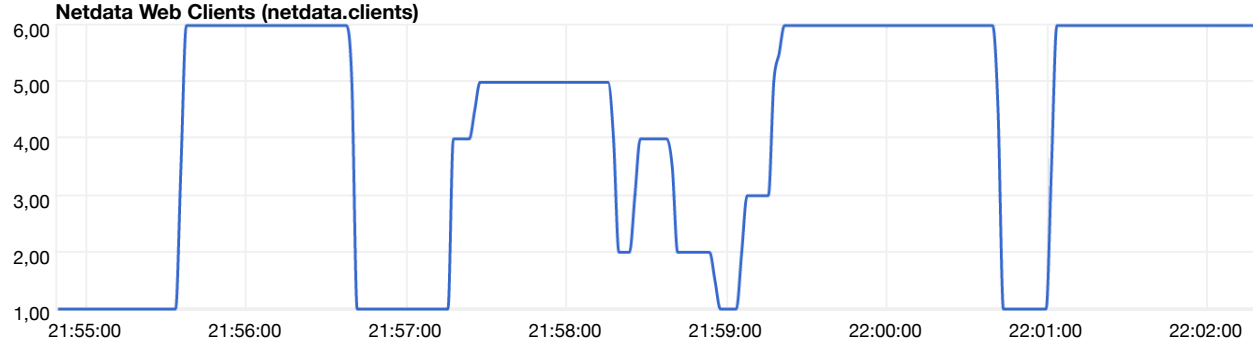
Netdata uptime (netdata.uptime)



time netdata:stats | netdata.uptime

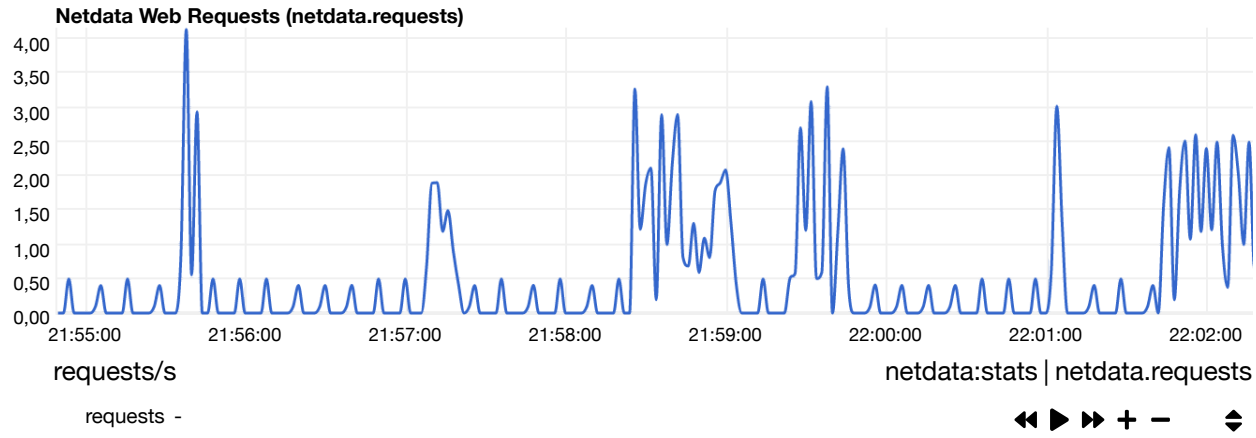
uptime -

Netdata Web Clients (netdata.clients)

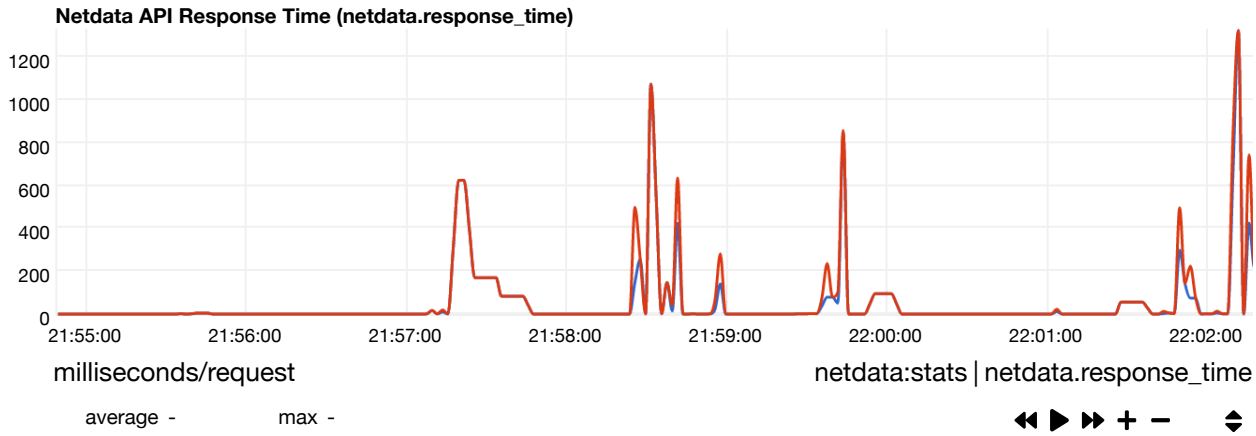


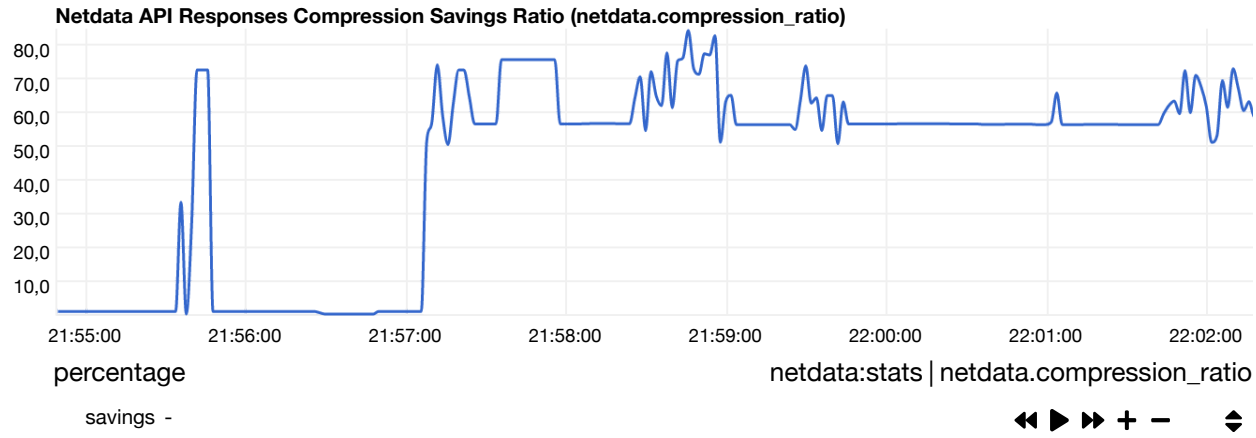
connected clients netdata:stats | netdata.clients

clients -



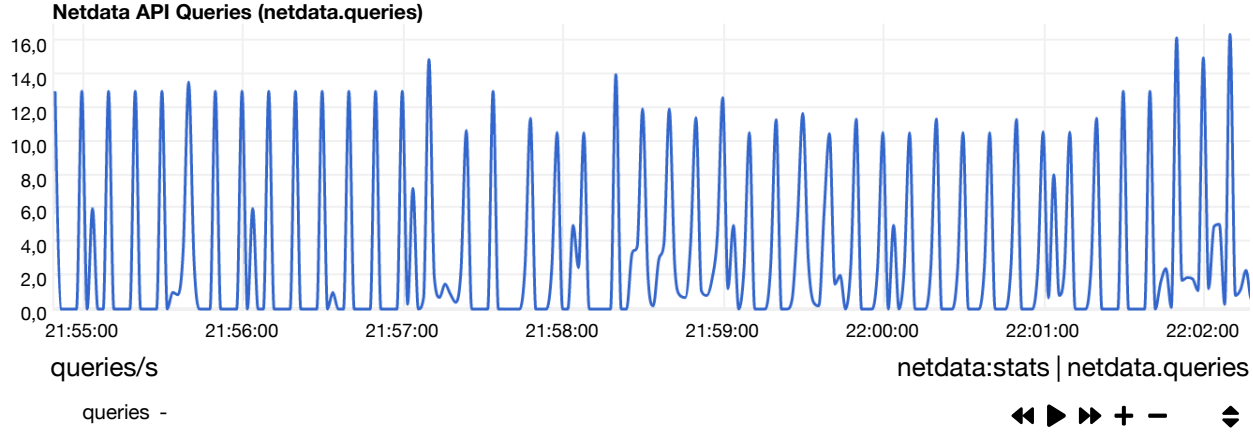
The netdata API response time measures the time netdata needed to serve requests. This time includes everything, from the reception of the first byte of a request, to the dispatch of the last byte of its reply, therefore it includes all network latencies involved (i.e. a client over a slow network will influence these metrics).



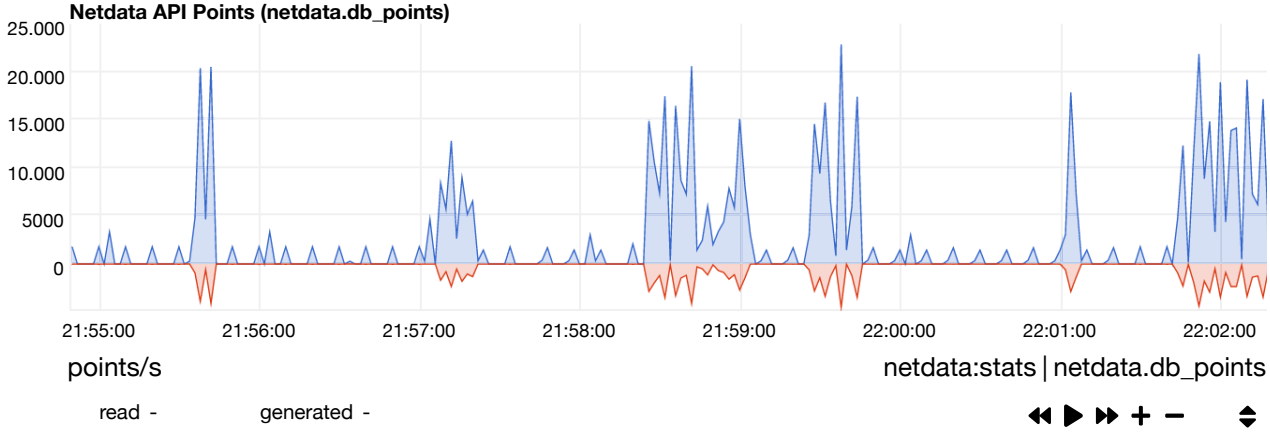


queries

Netdata API Queries (netdata.queries)

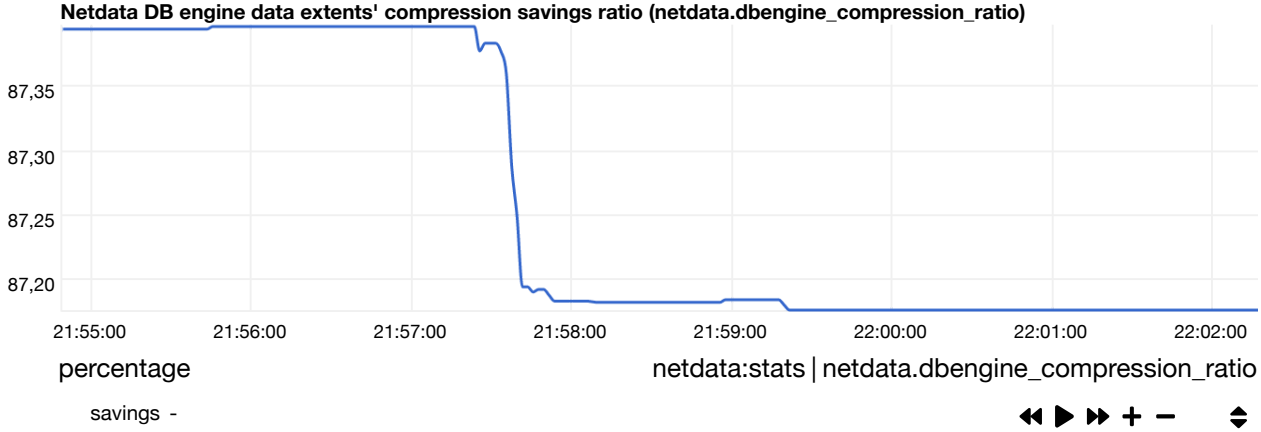


Netdata API Points (netdata.db_points)

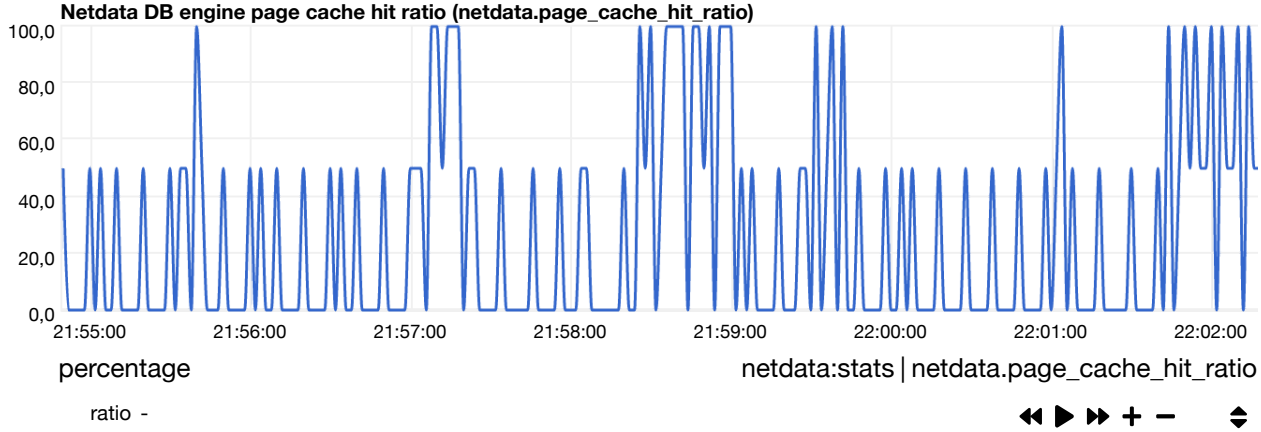


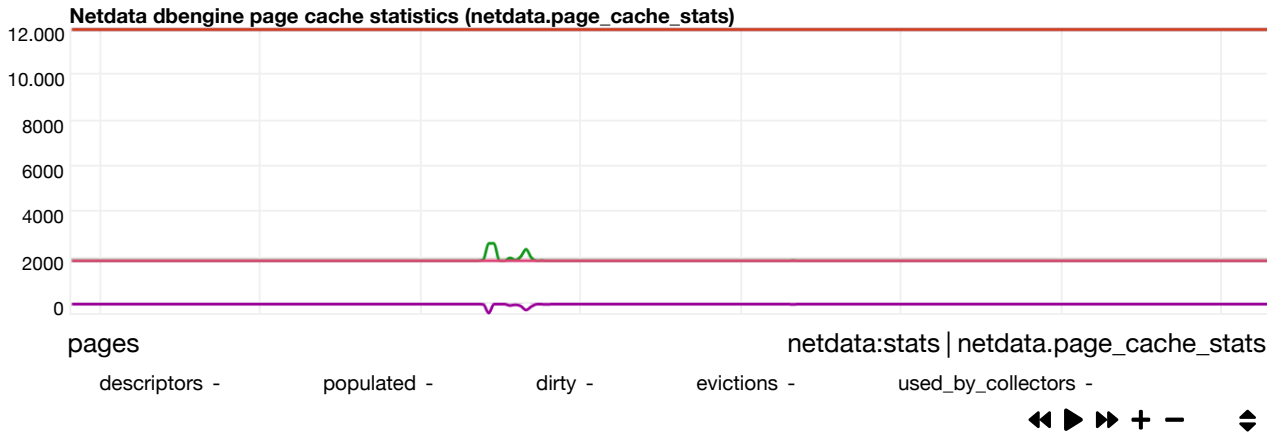
dbengine

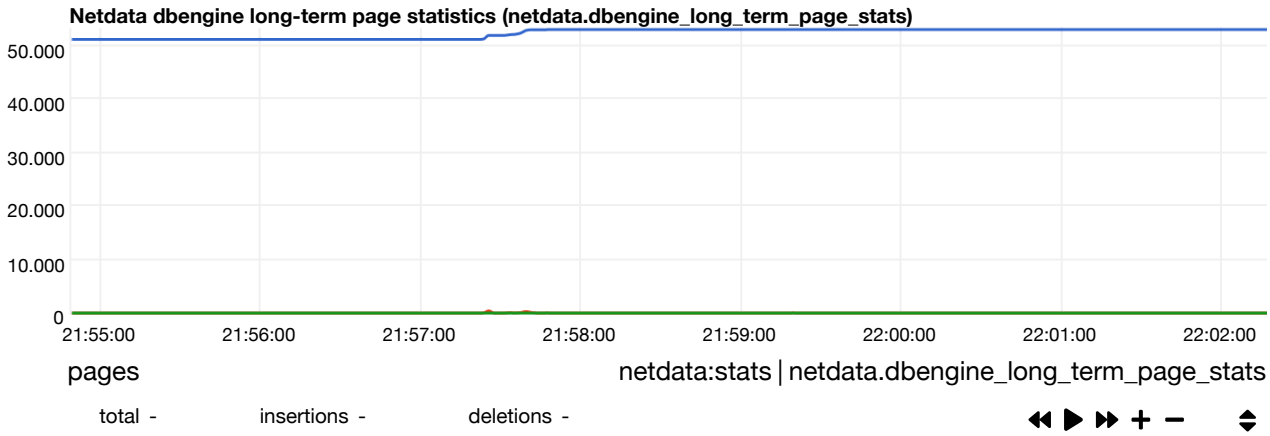
Netdata DB engine data extents' compression savings ratio (netdata.dbengine_compression_ratio)

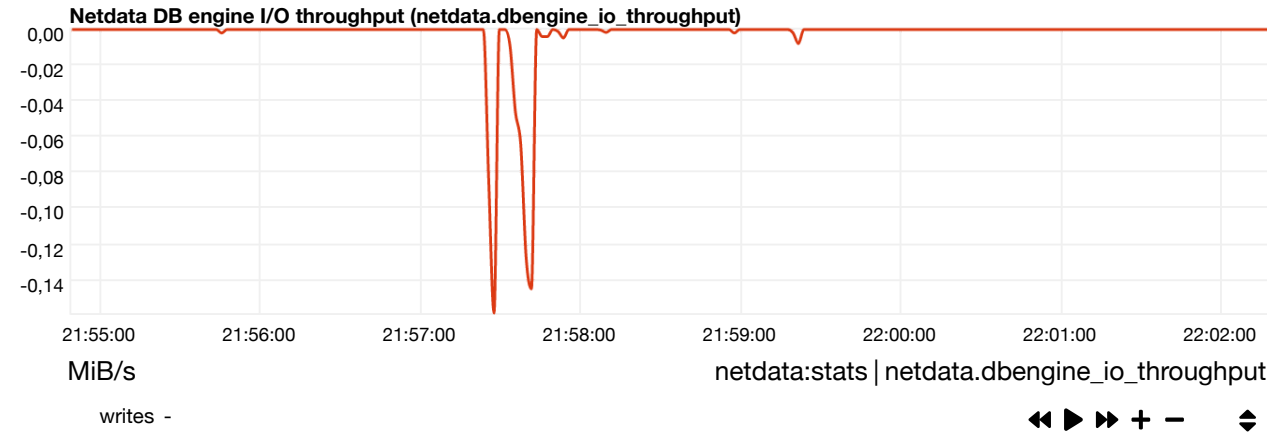


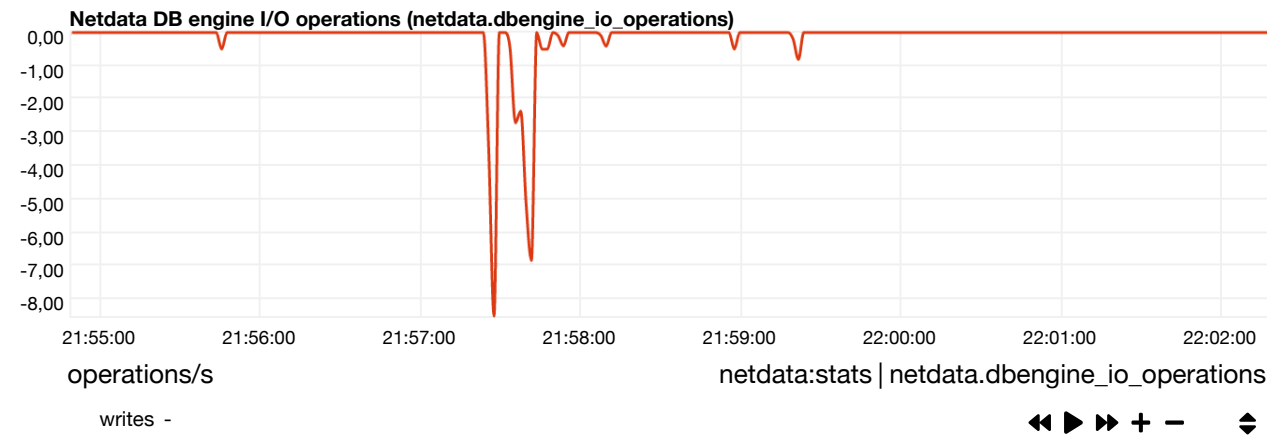
Netdata DB engine page cache hit ratio (netdata.page_cache_hit_ratio)

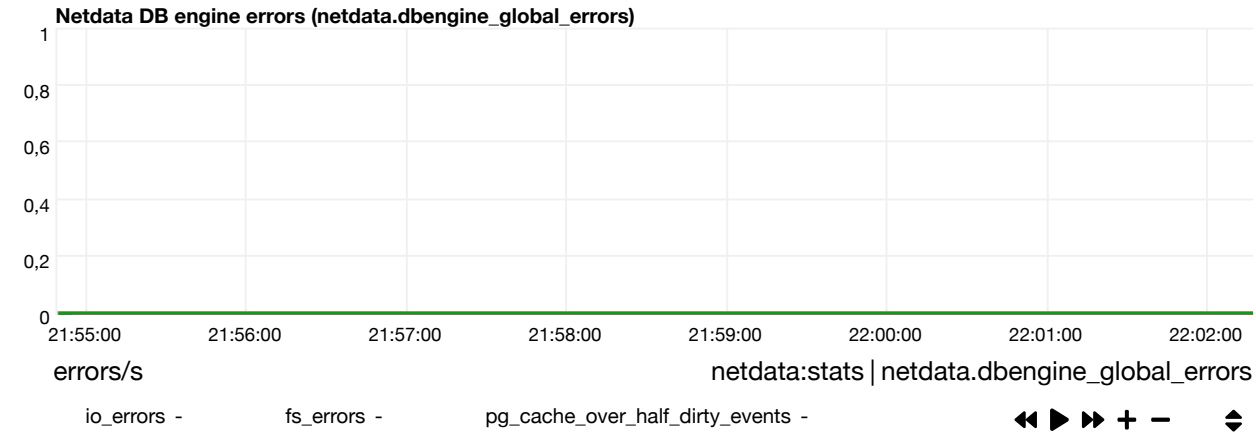


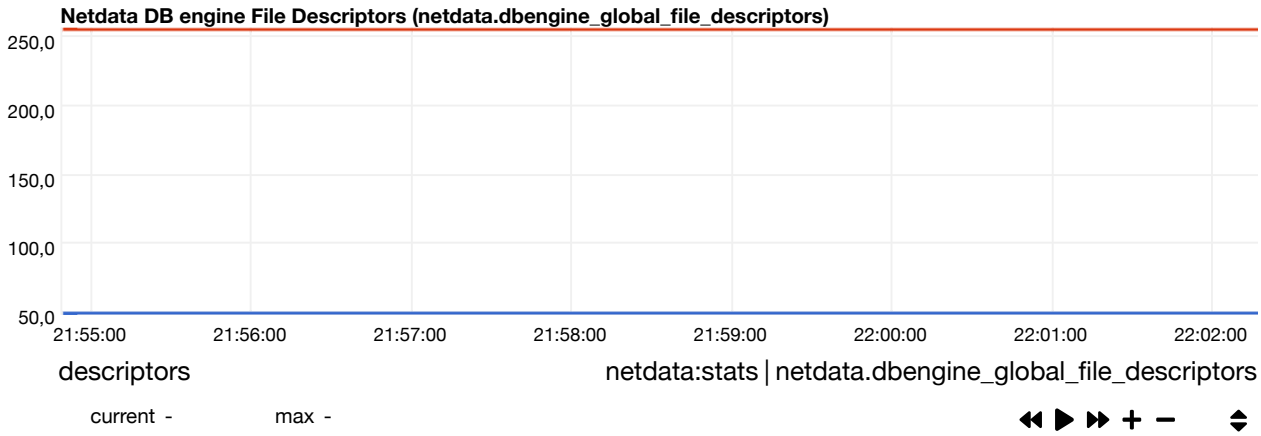

Netdata dbengine page cache statistics (netdata.page_cache_stats)
12.000
10.000
8000
6000
4000
2000
0
pages
netdata:stats | netdata.page_cache_stats
descriptors -
populated -
dirty -
evictions -
used_by_collectors -

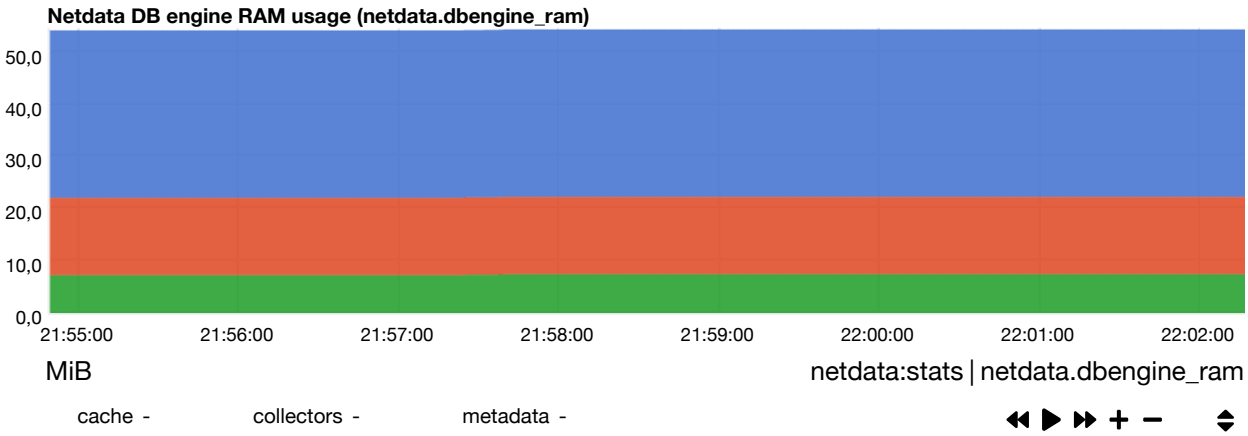

Netdata dbengine long-term page statistics (netdata.dbengine_long_term_page_stats)
50.000
40.000
30.000
20.000
10.000
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
pages
netdata:stats | netdata.dbengine_long_term_page_stats
total -
insertions -
deletions -


Netdata DB engine I/O throughput (netdata.dbengine_io_throughput)
0,00
-0,02
-0,04
-0,06
-0,08
-0,10
-0,12
-0,14
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB/s
netdata:stats | netdata.dbengine_io_throughput
writes -

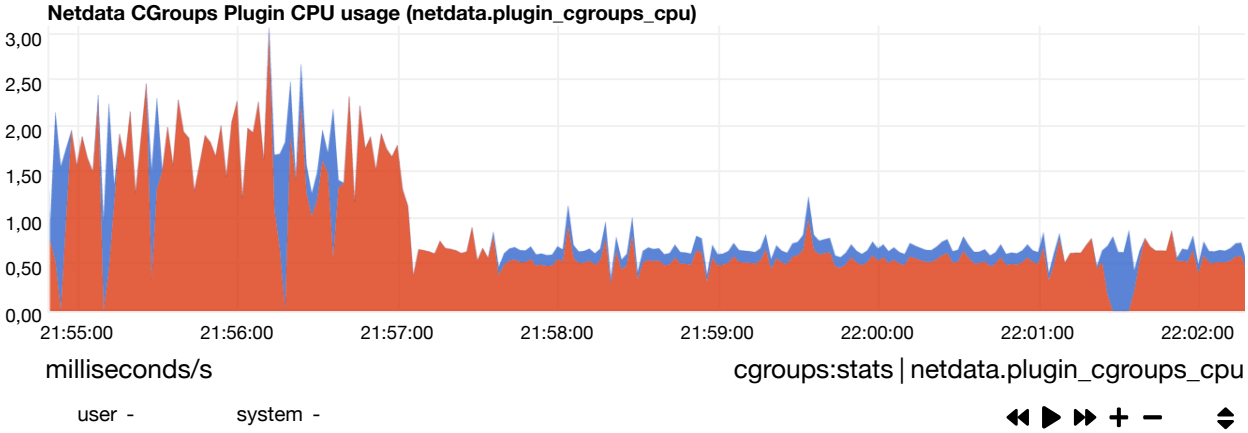

Netdata DB engine I/O operations (netdata.dbengine_io_operations)
0,00
-1,00
-2,00
-3,00
-4,00
-5,00
-6,00
-7,00
-8,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
operations/s
netdata:stats | netdata.dbengine_io_operations
writes -


Netdata DB engine errors (netdata.dbengine_global_errors)
1
0,8
0,6
0,4
0,2
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
errors/s
netdata:stats | netdata.dbengine_global_errors
io_errors -
fs_errors -
pg_cache_over_half_dirty_events -


Netdata DB engine File Descriptors (netdata.dbengine_global_file_descriptors)
250,0
200,0
150,0
100,0
50,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
descriptors
netdata:stats | netdata.dbengine_global_file_descriptors
current -
max -

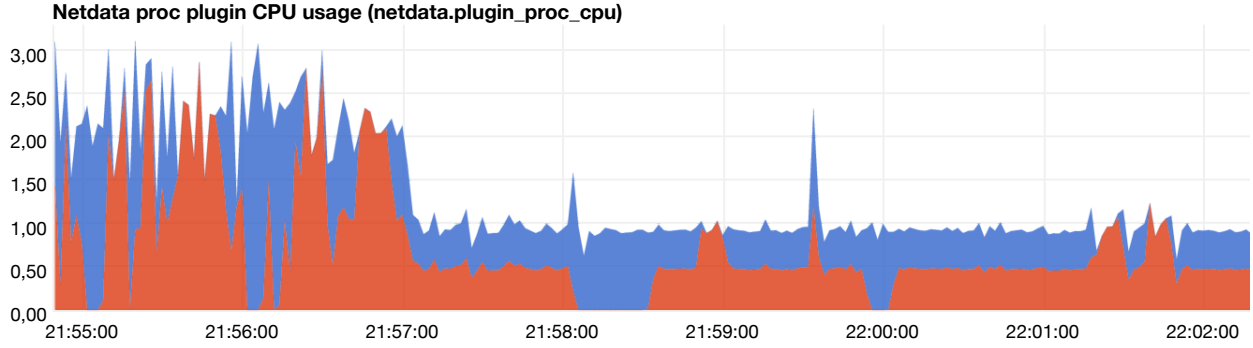

Netdata DB engine RAM usage (netdata.dbengine_ram)
50,0
40,0
30,0
20,0
10,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
MiB
netdata:stats | netdata.dbengine_ram
cache -
collectors -
metadata -

cgroups


Netdata CGroups Plugin CPU usage (netdata.plugin_cgroups_cpu)
3,00
2,50
2,00
1,50
1,00
0,50
0,00
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
milliseconds/s
cgroups:stats | netdata.plugin_cgroups_cpu
user -
system -

proc

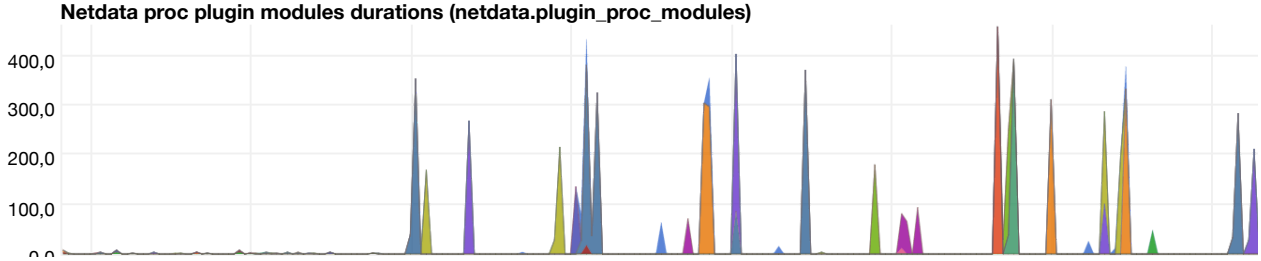
Netdata proc plugin CPU usage (netdata.plugin_proc_cpu)



milliseconds/s proc:stats | netdata.plugin_proc_cpu

user - system -

Netdata proc plugin modules durations (netdata.plugin_proc_modules)

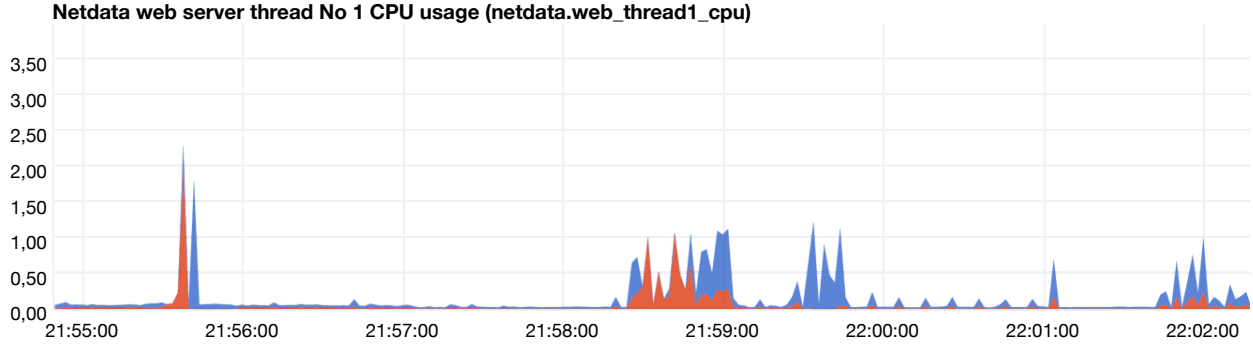


milliseconds/run proc:stats | netdata.plugin_proc_modules

stat - loadavg - entropy - interrupts - softirqs - vmstat -
meminfo - ksm - netdev - netwireless - sockstat - sockstat6 -
netstat - snmp - snmp6 - softnet - diskstats - mdstat -
ipc - power_supply -

web

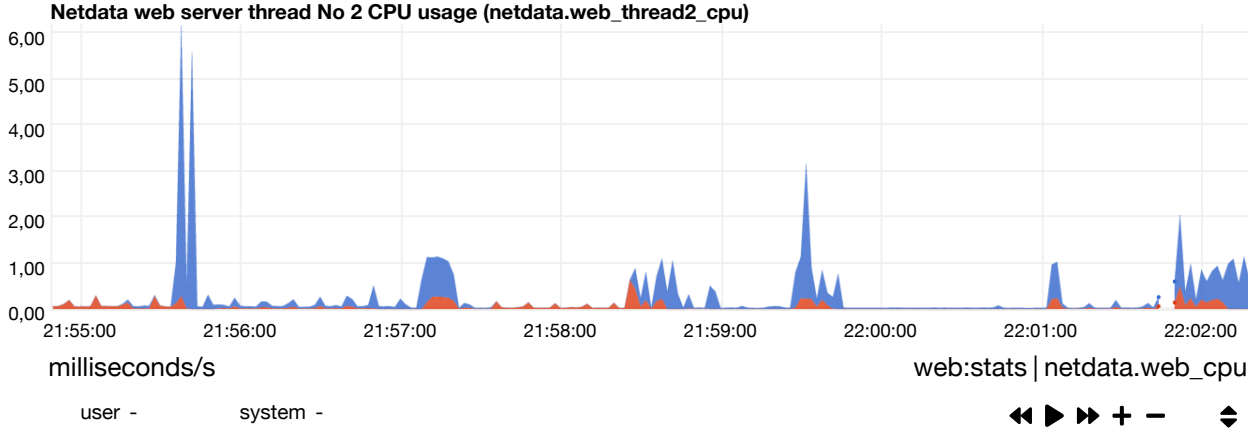
Netdata web server thread No 1 CPU usage (netdata.web_thread1_cpu)



milliseconds/s web:stats | netdata.web_cpu

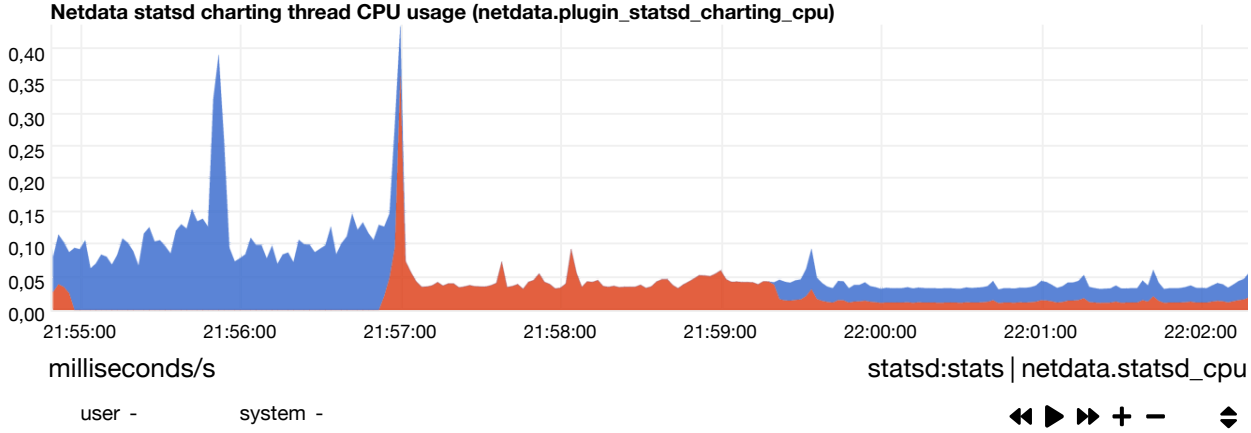
user - system -

Netdata web server thread No 2 CPU usage (netdata.web_thread2_cpu)

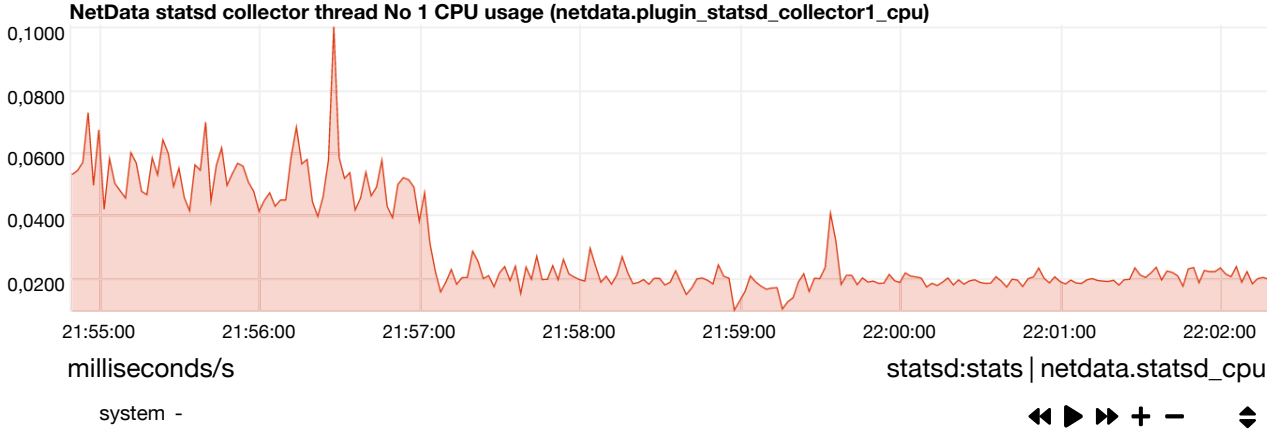


statsd

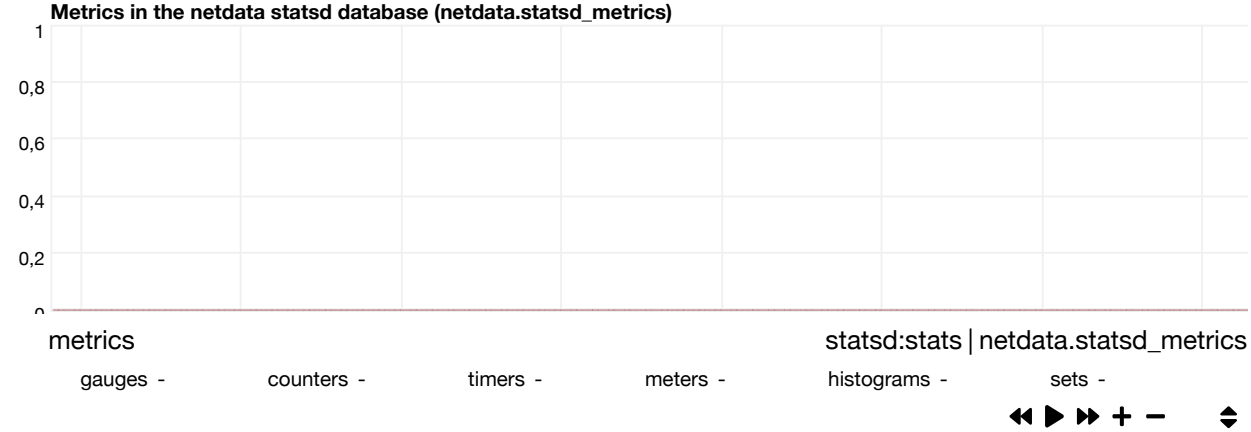
Netdata statsd charting thread CPU usage (netdata.plugin_statsd_charting_cpu)



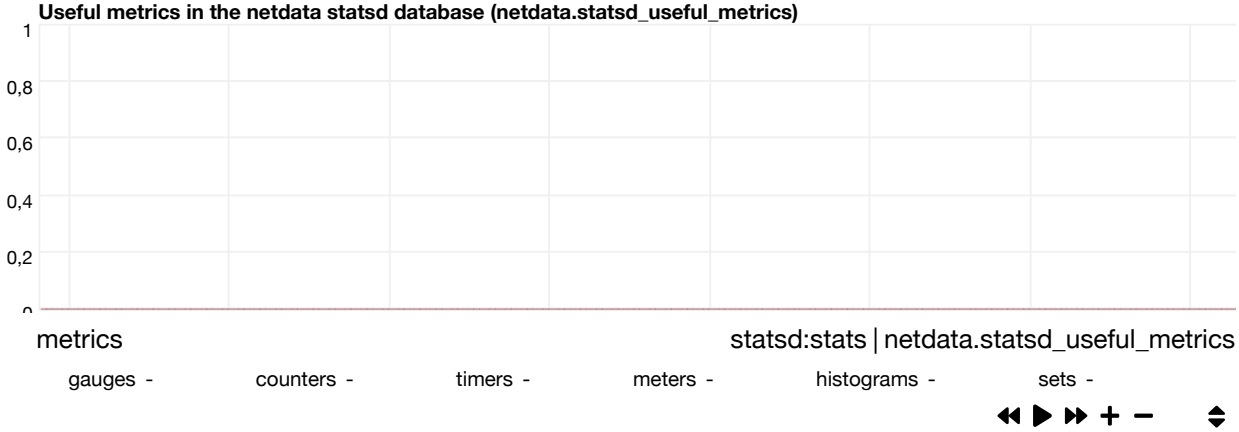
NetData statsd collector thread No 1 CPU usage (netdata.plugin_statsd_collector1_cpu)



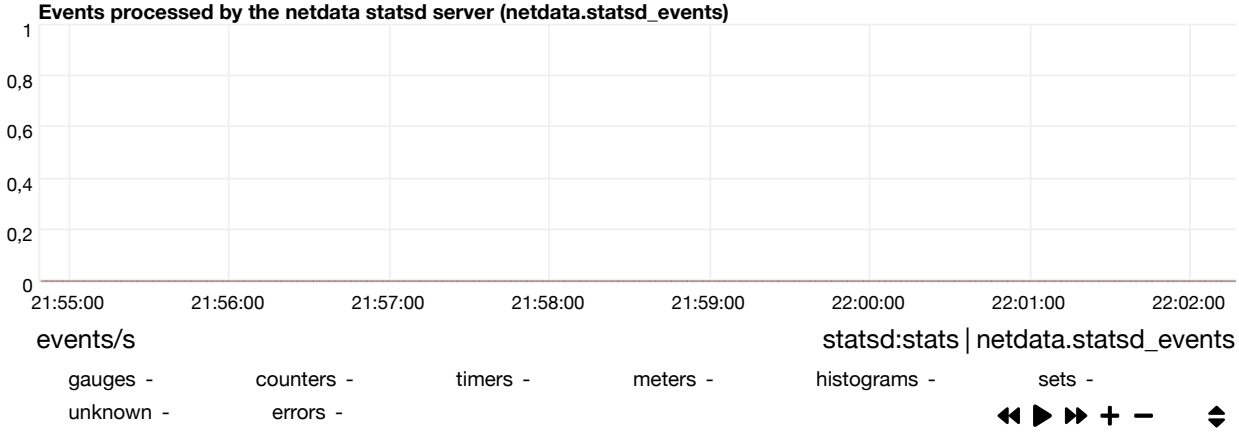
Metrics in the netdata statsd database (netdata.statsd_metrics)



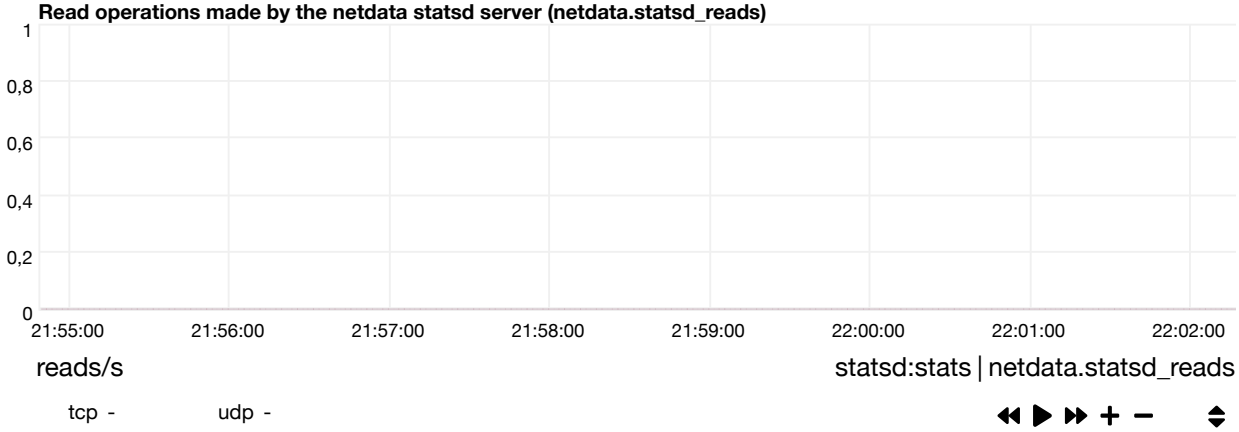
Useful metrics in the netdata statsd database (netdata.statsd_useful_metrics)



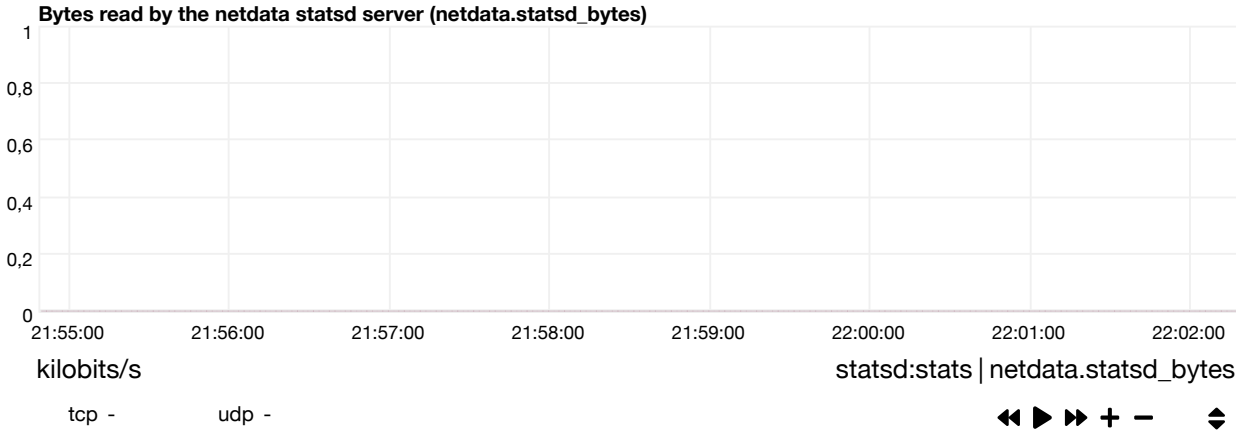
Events processed by the netdata statsd server (netdata.statsd_events)



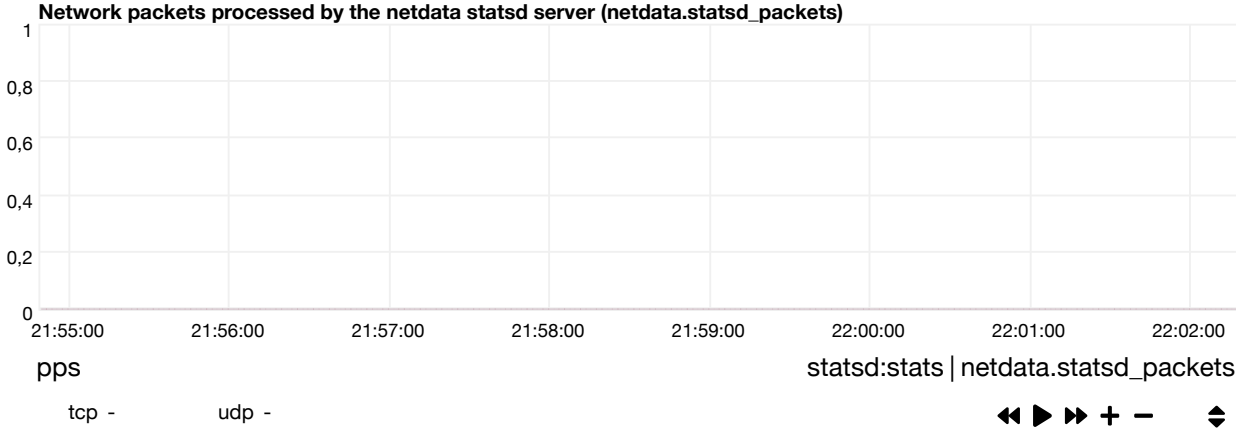
Read operations made by the netdata statsd server (netdata.statsd_reads)



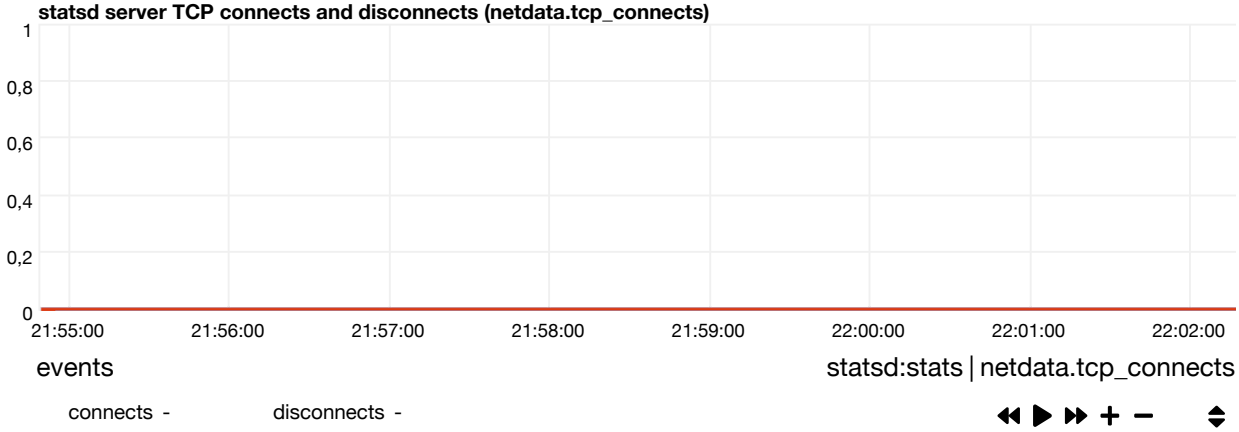
Bytes read by the netdata statsd server (netdata.statsd_bytes)



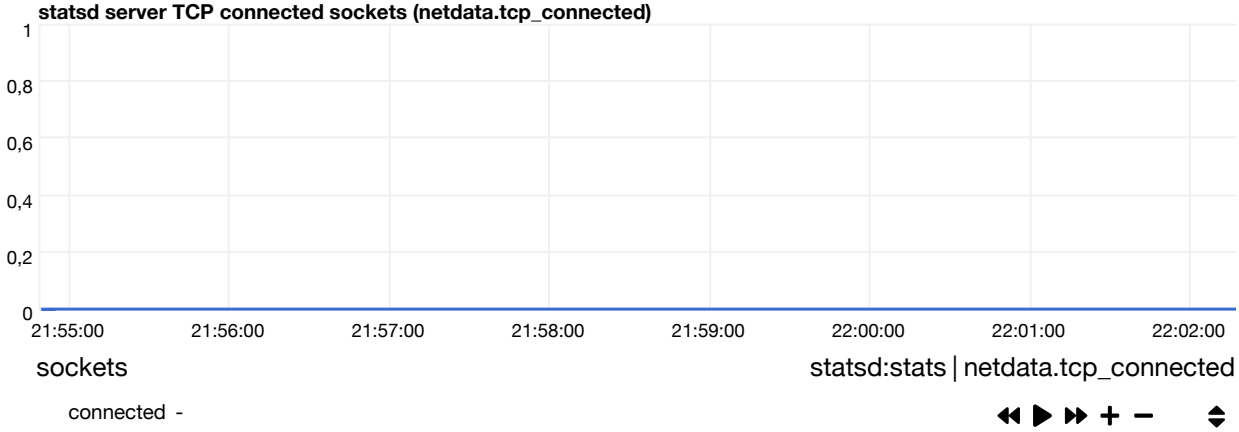
Network packets processed by the netdata statsd server (netdata.statsd_packets)



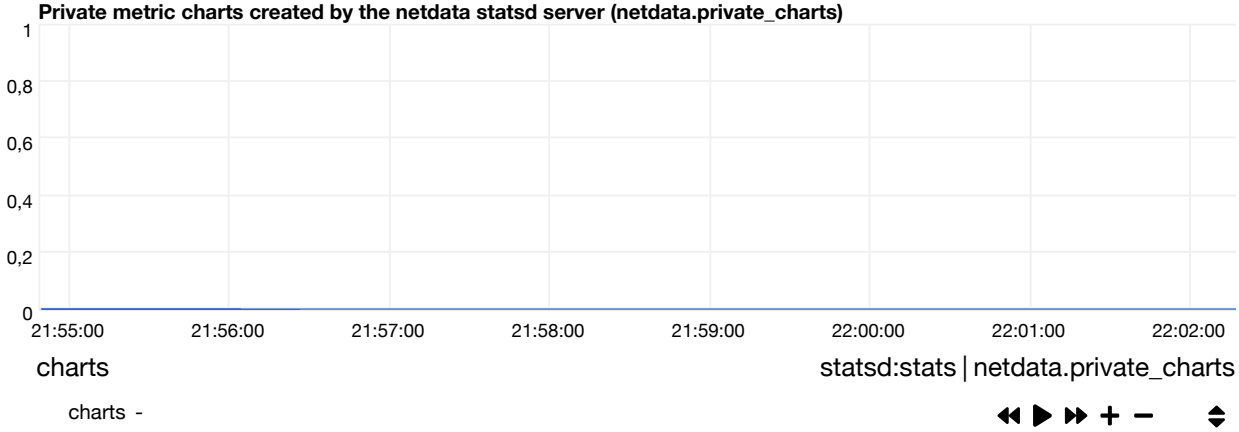
statsd server TCP connects and disconnects (netdata.tcp_connects)



statsd server TCP connected sockets (netdata.tcp_connected)

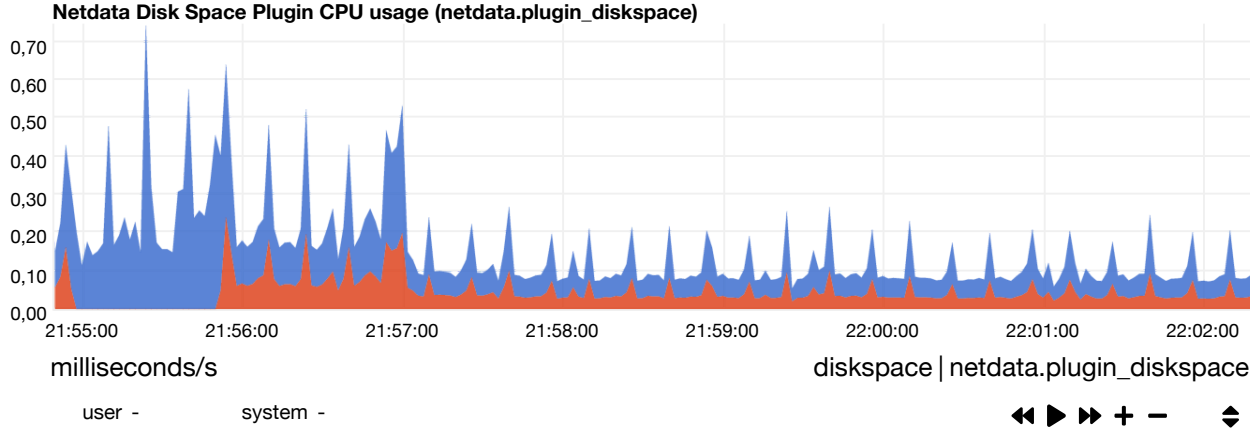


Private metric charts created by the netdata statsd server (netdata.private_charts)

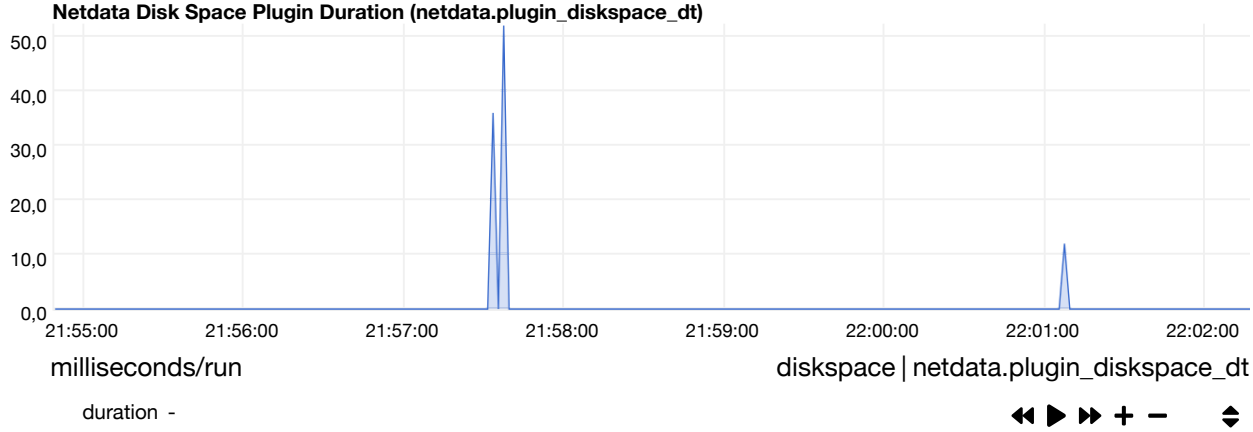


diskspace

Netdata Disk Space Plugin CPU usage (netdata.plugin_diskspace)

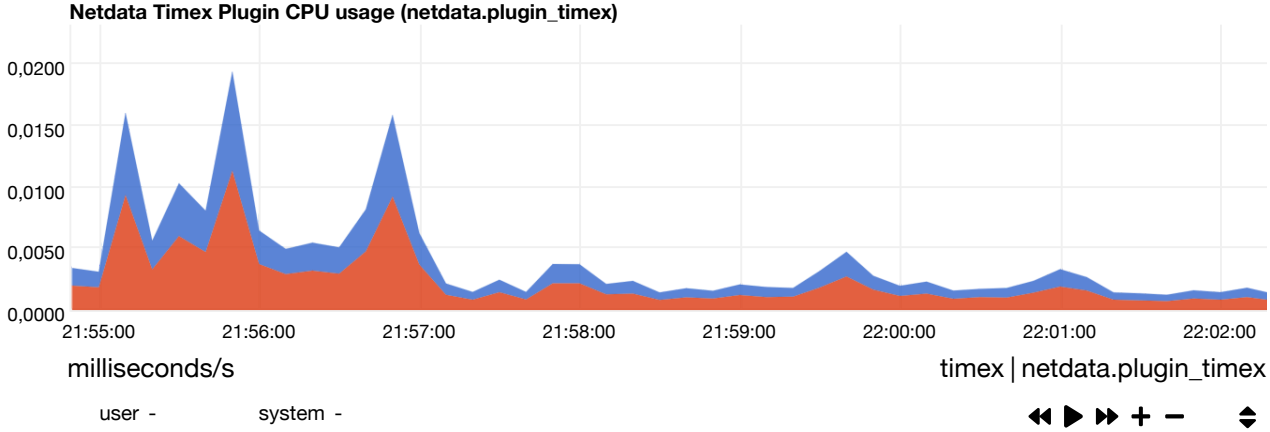


Netdata Disk Space Plugin Duration (netdata.plugin_diskspace_dt)

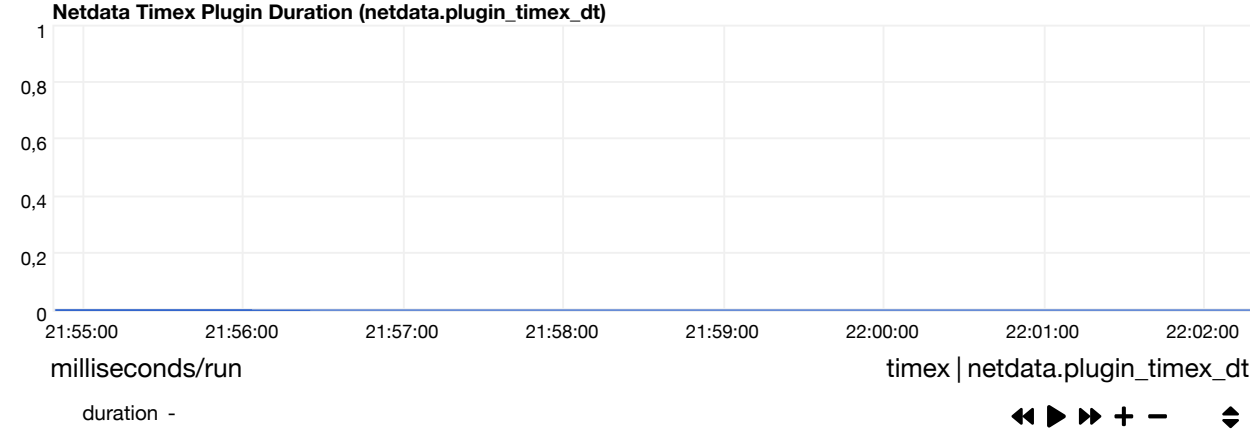


timex

Netdata Timex Plugin CPU usage (netdata.plugin_timex)

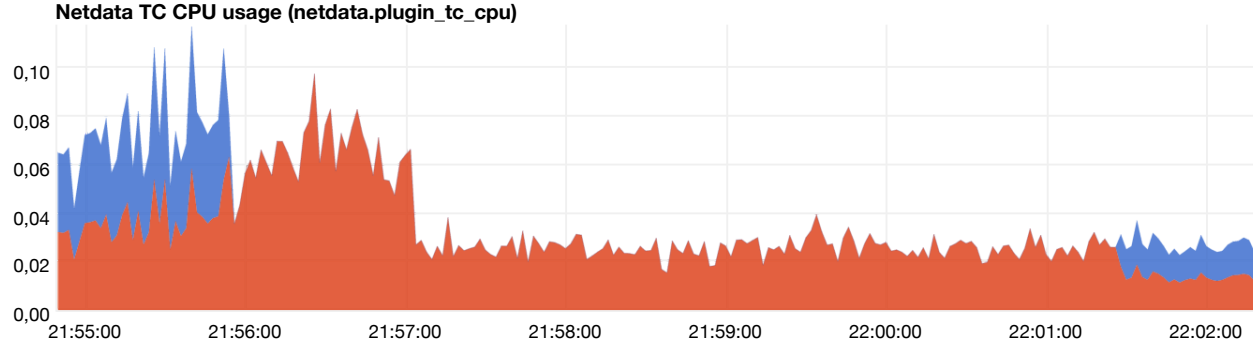


Netdata Timex Plugin Duration (netdata.plugin_timex_dt)



tc.helper

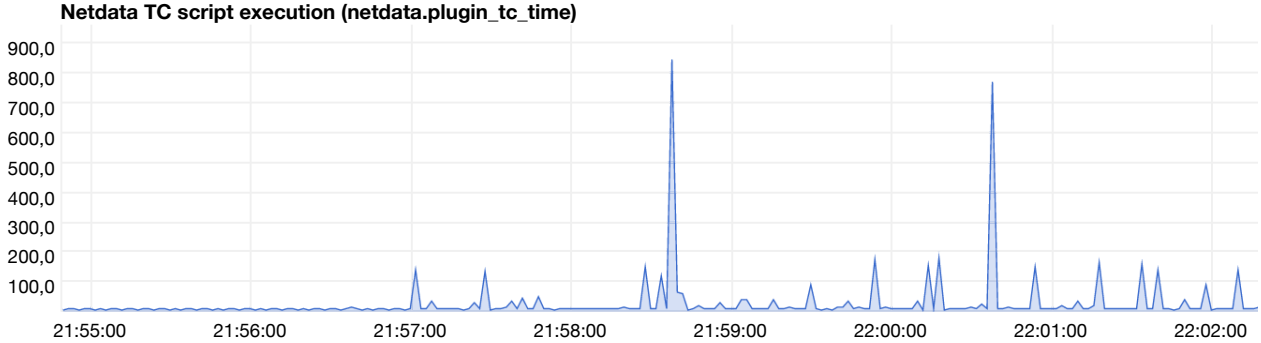
Netdata TC CPU usage (netdata.plugin_tc_cpu)



milliseconds/s tc | netdata.plugin_tc_cpu

user - system -

Netdata TC script execution (netdata.plugin_tc_time)

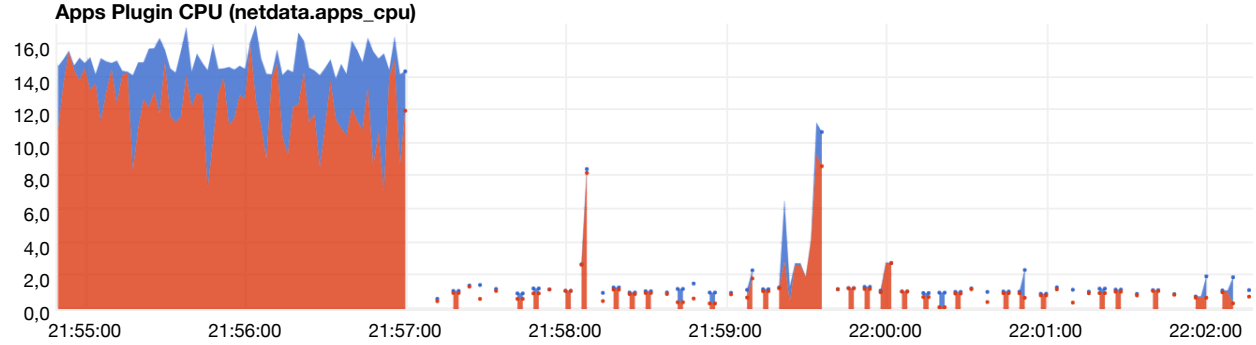


milliseconds/run tc | netdata.plugin_tc_time

run time -

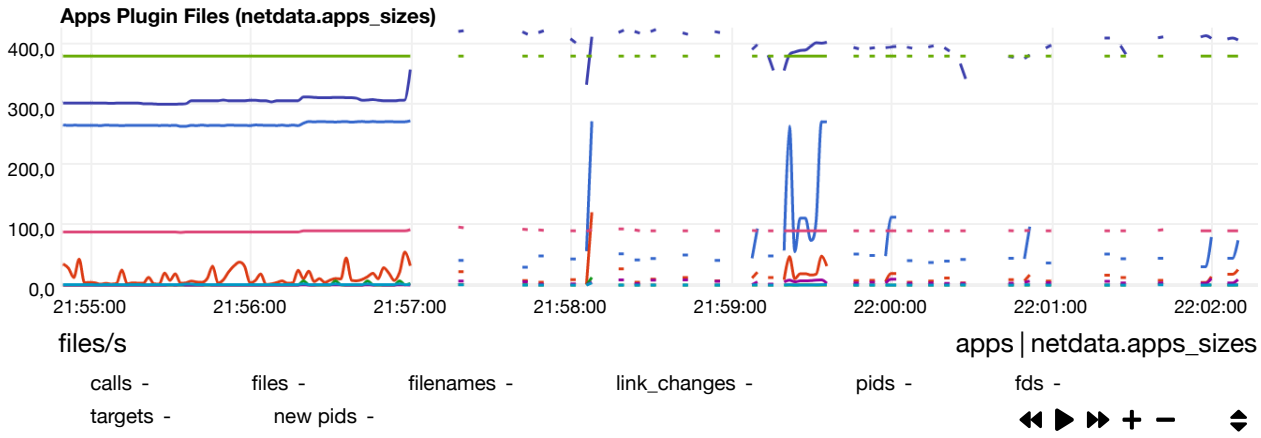
apps.plugin

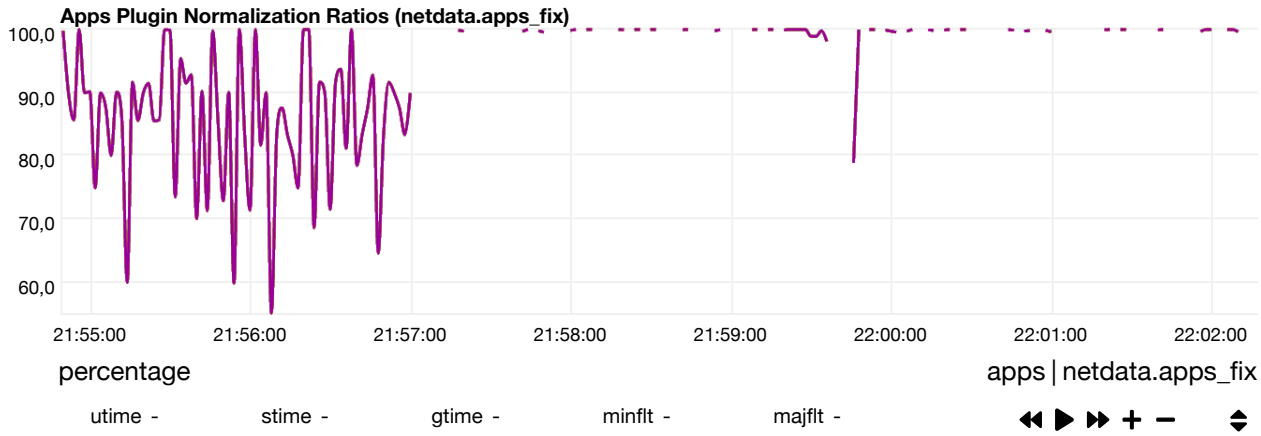
Apps Plugin CPU (netdata.apps_cpu)

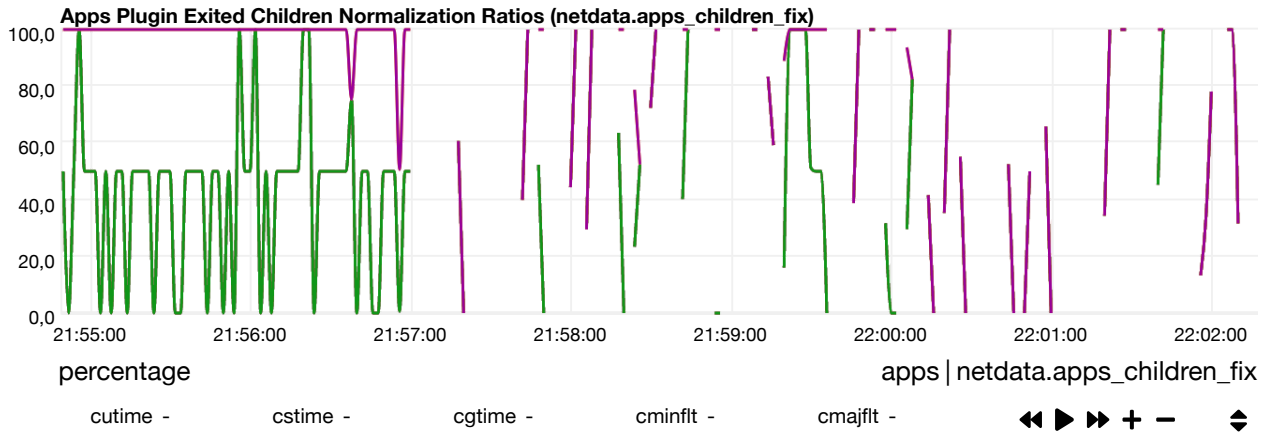


milliseconds/s apps | netdata.apps_cpu

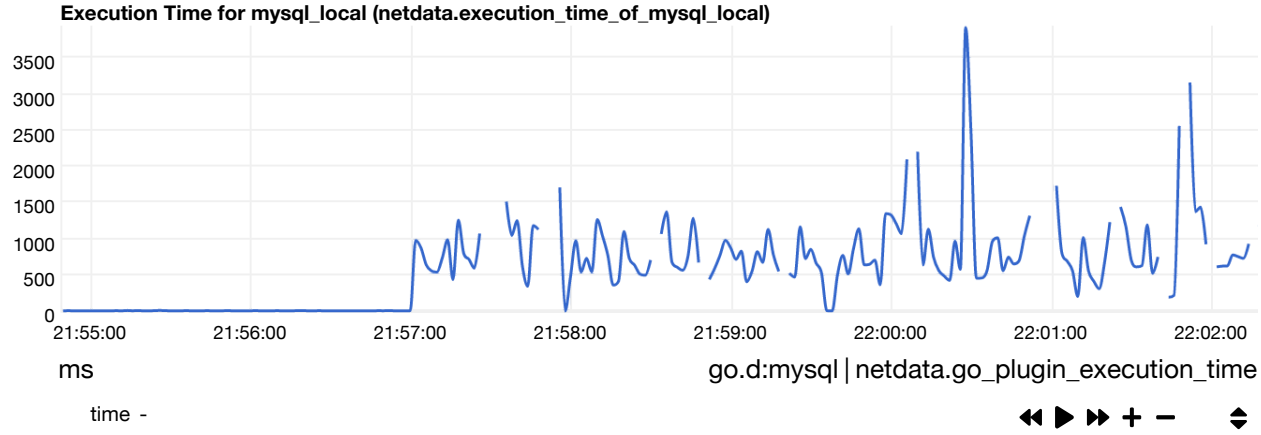
user - system -

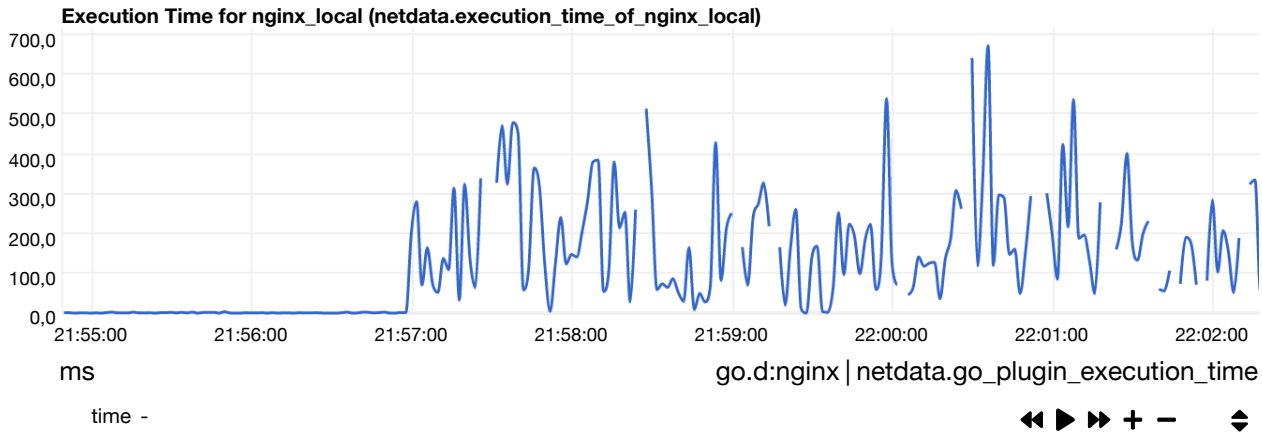

Apps Plugin Files (netdata.apps_sizes)
400,0
300,0
200,0
100,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
files/s
apps | netdata.apps_sizes
calls -
files -
filenames -
link_changes -
pids -
fds -
targets -
new pids -

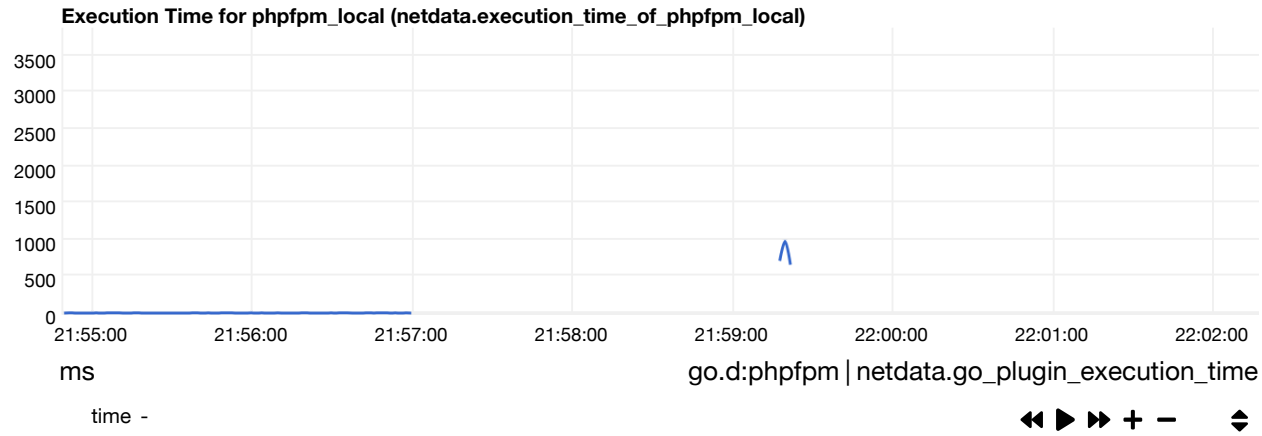

Apps Plugin Normalization Ratios (netdata.apps_fix)
100,0
90,0
80,0
70,0
60,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
percentage
apps | netdata.apps_fix
utime -
stime -
gtime -
minflt -
majflt -

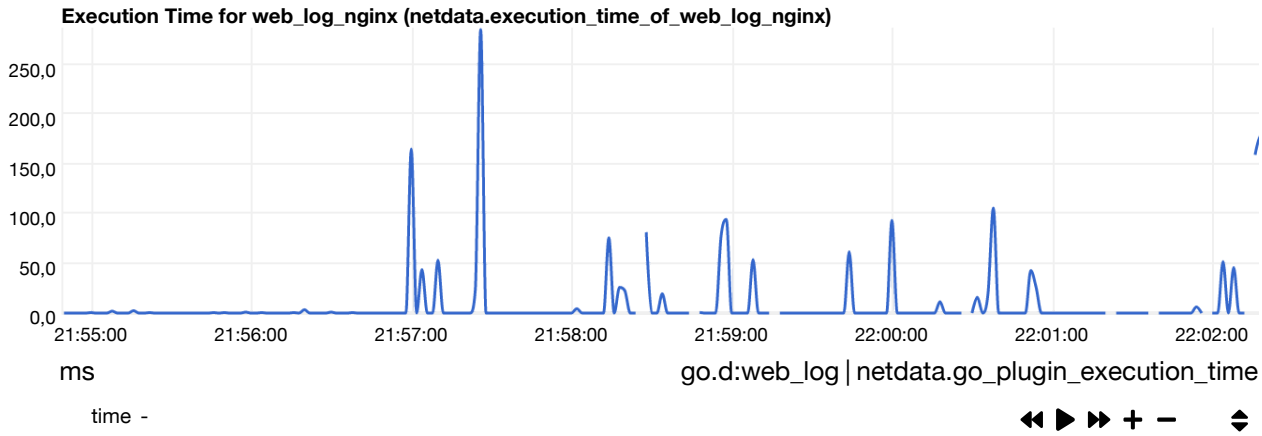

Apps Plugin Exited Children Normalization Ratios (netdata.apps_children_fix)
100,0
80,0
60,0
40,0
20,0
0,0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
percentage
apps | netdata.apps_children_fix
cutime -
cstime -
cgtime -
cminflt -
cmajflt -

go.d


Execution Time for mysql_local (netdata.execution_time_of_mysql_local)
3500
3000
2500
2000
1500
1000
500
0
21:55:00
21:56:00
21:57:00
21:58:00
21:59:00
22:00:00
22:01:00
22:02:00
ms
go.d:mysql | netdata.go_plugin_execution_time
time -







Netdata (https://github.com/netdata/netdata/wiki)