

**Reference:** LinuxAIOPerf\_UserManual.pdf  
**Version:** 1.0.0  
**Homepage:** <https://github.com/samatild/LinuxAiOPerf>

---

## LINUX ALL-IN-ONE PERFORMANCE

# USER MANUAL

### PURPOSE OF THE DOCUMENT

The purpose of this document is to provide comprehensive instructions on how to effectively utilize the Linux AIO Performance Checker. This tool is a script designed to gather performance metrics from a Linux-based system, subsequently generating a report in an HTML format. The resultant report can be seamlessly uploaded to a web application, facilitating a user-friendly display of the data. This manual will guide you through the process of using the Linux AIO Performance Checker, allowing you to maximize its features for optimal system performance analysis and management.

This document is made available under the terms of the **MIT License**. Unauthorized reproduction or communication of this document is prohibited without obtaining prior consent in accordance with the conditions outlined in the **MIT License**.

LinuxAIOPerf_UserManual.pdf		<a href="https://github.com/samatild/LinuxAiOPerf">https://github.com/samatild/LinuxAiOPerf</a>		
Author	Samuel Matildes	Version :	1.0.0	Page 1 / 34

## DOCUMENT HISTORY







Date	Version	Comments
20/02/2024	1.0.0	Initial Release

## CONTENTS

<b>3. Introduction.....</b>	<b>5</b>
3.1    WHAT IS LINUX AIO PERFORMANCE CHECKER & REPORT? .....	5
3.2    WHAT IS the 3 “W” approach?.....	5
3.3    How can Linux AIO contribute to 3 “W”s approach? .....	5
<b>4. Compatibility &amp; requirements .....</b>	<b>6</b>
4.1    linux collector script .....	6
4.2    Web app – report generator.....	6
<b>5. How to collect data .....</b>	<b>7</b>
5.1    Quick Start .....	7
5.2    Run Modes .....	8
5.2.1    MODE 1 - Live Data .....	8
5.2.2    MODE 2 – Watchdog .....	9
WATCHDOG – AUTO MODE .....	9
WATCHDOG – MANUAL MODE .....	9
WATCHDOG EXAMPLES .....	10
5.2.3    MODE 3 – Cronjob .....	13
Cronjob Setup – Recurrent.....	13
Cronjob Setup – at Specific time .....	14
<b>6. Using the Web application .....</b>	<b>15</b>
6.1    Access the web application .....	15
6.2    Uploading data .....	15
<b>7. Data Analysis.....</b>	<b>17</b>
7.1    System information Tab .....	17
7.2    Performance Analysis Tab .....	20
7.2.1    CPU Load Distribution .....	21
7.2.2    Memory utilization.....	23
7.2.3    Disk counters metrics per device .....	24
7.2.4    Disk counters device per metrics .....	25
7.2.5    Network Statistics .....	26
7.3    Process Information Tab .....	27
7.3.1    Process stats [CPU].....	28
7.3.2    Process stats [IO] .....	29
7.3.3    Process stats [memory] .....	30
7.3.4    top .....	31
7.3.5    iotop .....	32
<b>8. About.....</b>	<b>33</b>

## 1. ICONS USED IN THE MANUAL

In this document, the following icons are utilized to emphasize key points or significant concepts.

	Important information
	Good to know - Tricks
	<b>Risk</b> in front of a parameter setting or of a specific action
	Action to be avoided
	Mandatory action
	Sensitive or difficult procedure. To take into account necessarily

## 2. DISCLAIMER

By choosing to run the Linux AIO perf report, you, the user, acknowledge and agree to the terms outlined in this disclaimer.

This software is provided "as is," without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the software or the use or other dealings in the software.

The Linux AIO perf report is licensed under the terms of the MIT License. By running this report, you agree to the terms outlined in the MIT License. For full details of this license, please refer to the accompanying license file or the MIT License text provided with the source code.

Furthermore, the use of the Linux AIO perf report is at your own risk. The author or copyright holders are not responsible for any potential damage or issues that may arise from running this report.

By running the Linux AIO perf report, you signify your acceptance of this disclaimer. If you do not agree to this disclaimer, please do not use the Linux AIO perf report.

## 3. INTRODUCTION

---

### 3.1 WHAT IS LINUX AIO PERFORMANCE CHECKER & REPORT?

A tool created to simplify the collection and analysis of the most common Performance bottlenecks in Linux. Following a three “W” approach it helps co-relate System Information, counters and process activity logs to understand where the culprit may be.

### 3.2 WHAT IS THE 3 “W” APPROACH?

When: Problem occurrence / TIMESTAMP  
What: What is happening? What is the consequence of this issue?  
Who: Who is doing it? Who is the culprit?

### 3.3 HOW CAN LINUX AIO CONTRIBUTE TO 3 “W”S APPROACH?

The data collected by Linux AIO encompasses a variety of elements, including counters, process activity, and system logs. This information is readily accessible and easy to interpret via the web report's user-friendly tabs. Each tab is clearly labeled to allow users to quickly understand its content and navigate through the information efficiently.

## 4. COMPATIBILITY & REQUIREMENTS

### 4.1 LINUX COLLECTOR SCRIPT

The collector script is compatible with most modern Linux distributions as it uses standard OS commands to collect data.

It has been tested with the following Linux distributions:

- Ubuntu 18.04
- Ubuntu 20.04
- CentOS 7
- CentOS 8
- Red Hat Enterprise Linux 7
- Red Hat Enterprise Linux 8
- SUSE Linux Enterprise Server 15
- SUSE Linux Enterprise Server 12

#### NOT SUPPORTED (YET!)

- Debian 11 / 12
- Rocky Linux



**Mandatory:** `sysstat` and `iostat` packages are required for the script to execute. If they are not installed, the script will install them automatically. If the script is not able to install them, it will exit with an error message.

### 4.2 WEB APP – REPORT GENERATOR

The web application was built using Flask and utilizes HTML5 for its design. Therefore, it is compatible with any modern web browser.



However, the developer highly recommends using **Mozilla Firefox**. This is because Firefox has superior in-page caching capabilities, which enhances the browsing experience when loading counters into the report.

## 5. HOW TO COLLECT DATA

### 5.1 QUICK START

Directly download script from github page and execute it:

```
# Download
wget https://raw.githubusercontent.com/samatild/LinuxAiOPerf/main/build/linux_aio_perfcheck.sh

# Make it executable
sudo chmod +x linux_aio_perfcheck.sh

# Run it
sudo ./linux_aio_perfcheck.sh
```

Once the script executes, it will display a welcome screen with a **MENU** where the user can select the **MODE** that data is going to be collected.

```
=====
Linux All-in-One
Performance Collector
=====
* Unlocking advanced Linux metrics for humans *

=====
Package Validation
=====
[sysstat] Installed
[iotop] Installed

Dependencies met. Validation passed.

=====
Select Run Mode
=====

1 - Collect live data           (Now)
2 - Collect data via watchdog  (Triggered by High CPU, Memory, or Disk IO)
3 - Collect data via cron       (At a specific time)

=====

Enter the mode number:
```

Linux AIO Bash Script Initial Screen & Menu

## 5.2 RUN MODES

Run Mode	What it does	For what occasion
<b>Live Data</b>	It will collect data right away during a timespan from 10-900sec	Problem is happening now
<b>Watchdog</b>	It will setup a watchdog that will keep an eye for resource Utilization. Auto or Manual modes available	For when you don't know when the problem is going to happen.
<b>Cron</b>	It will setup a cronjob based on user instructions. There are 2 different cronjobs: Recurrent will repeat the collection based on user section. Not Recurrent will trigger the data collection at a specific time.	For when you know when the problem happens.

### 5.2.1 MODE 1 - LIVE DATA

The Live Data feature is designed to gather data immediately upon activation. It will prompt the user for confirmation and request the duration for which the data should be collected. The report will be based on current metrics and will not contain historical data.

**Timespan:** From 10 seconds to 900 seconds (5 Minutes)

```
Enter the mode number: 1
=====
Live Data Capture
=====
Enter the capture duration in seconds (minimum 10, maximum 900)
Duration: 30
Starting data capture for 30 seconds...

Starting
Gathering general system information (...)
Initializing performance capture
Elapsed Time / Total: 10 / 30 (seconds)
Elapsed Time / Total: 20 / 30 (seconds)
Elapsed Time / Total: 30 / 30 (seconds)
Capture Complete.

=====
Report Creation
=====
Creating tarball and cleaning the trash.
Script execution completed.
Output file: /root/rhel88_20240215_112115_linuxaioperfcheck.tar.gz
```

Live Data being Collected Example



## [WATCHDOG SETUP]

### 5.2.2 MODE 2 – WATCHDOG

Watchdog is intended for scenarios when the user doesn't know when the problem is going to happen.

- If the user knows what resource is affected, he can decide which resource to monitor.
- Users can also define a custom Threshold to monitor each resource (ex: 100% of CPU). On this mode, the user can select the duration of the collection between 1 and 300 seconds.
- Default options will monitor CPU, Memory and Disk activity and will trigger the collection at 80% with the default duration of 60 seconds.



**WARNING:** Setting up a resource watchdog will run a separate process in the background. This process will monitor CPU, memory, and disk IO utilization until threshold is reached.

### WATCHDOG – AUTO MODE

This sub-mode will:

- Monitor CPU, Memory and Disk IO
- Threshold: > 80% for any of the associated resources
- Duration: 60 seconds

Whenever the conditions above are met, watchdog will trigger a report collecting for 60 seconds duration and exit.

### WATCHDOG – MANUAL MODE

This sub-mode gives the user the ability to choose which resources to monitor and the corresponding thresholds to trigger the report capture.

#### User Inputs:

- Monitor CPU: Select if you wish to include CPU monitoring on the watchdog. Value calculated from CPU (per core) utilization. Threshold from 0-100% can be selected.
- Monitor Memory: Select if you wish to include memory monitoring. Value is calculated by the non-free %util memory. Threshold from 0-100% can be selected.
- Monitor Disk IO: Select if you wish to monitor Disk IO. The %util value of iostat is used to calculate how much disk usage is currently happening. It will trigger **ANY** available disk. Threshold from 0-100% can be selected.

## [WATCHDOG SETUP cont.]

### WATCHDOG EXAMPLES

Outlined below are several scenarios illustrating the significant impact of using the watchdog feature. It's important to emphasize that watchdog features are particularly beneficial when the user may not know the exact timing of an issue but is aware of its consequences.

#### Example 1: High CPU Utilization (MANUAL MODE)

*"I have High CPU utilization from time to time, but I can't specify when it happens."*

For the scenario above we will set up a CPU watchdog that will trigger whenever CPU utilization is above 80%:

- **Mode** Manual
- **Monitor CPU?** Yes
- **Set CPU threshold (0-100)** 80
- **Monitor Memory?** No
- **Monitor Disk IO?** No
- **Duration** <Desired report duration in Seconds>

This is how the output is going to be after setting up the CPU watchdog. Here we can see that we have a new associated PID 16069 that will be responsible for monitoring the selected resource types:

```
Choose a mode for the Resource Watchdog:
1 - Auto   : Monitor CPU, memory, and disk at 80% Threshold. Duration: 60sec
2 - Manual : Choose resources and Thresholds. Duration: 1-300sec
Enter 1 or 2: 2
Resource Monitoring Choices:
Monitor CPU? (yes/no): yes
Set CPU threshold (0-100): 80
Monitor Memory? (yes/no): no
Monitor Disk IO? (yes/no): no
Duration (1-300sec): 60
[User Input]
Associated WATCHDOG pid
[root@rhel88 ~]# Resource watchdog started. PID: 16069
```

Watchdog Setup Example

## [WATCHDOG SETUP cont.]

The watchdog PID will run in system background and initiate a report capture once the threshold value is reached. From the screenshot below, we can see that the watchdog was triggered and that it collected a 60 second report.

```
[root@rhel88 ~]# Resource watchdog started. PID: 16069

[root@rhel88 ~]# Starting data capture for 60 seconds ...

Starting
Gathering general system information (...)
Initializing performance capture
Elapsed Time / Total: 10 / 60 (seconds)
Elapsed Time / Total: 20 / 60 (seconds)
Elapsed Time / Total: 30 / 60 (seconds)
Elapsed Time / Total: 40 / 60 (seconds)
Elapsed Time / Total: 50 / 60 (seconds)
Elapsed Time / Total: 60 / 60 (seconds)
Capture Complete.

=====
Report Creation
=====

Creating tarball and cleaning the trash.
Script execution completed.
Output file: /root/rhel88_20240215_124647_linuxaioperfcheck.tar.gz
```

Capture Initiated by Watchdog Example

It is expected that the report contains the associated resource exhaustion. You can upload the report to the web app and analyze which process was responsible for High CPU utilization. This part is covered on [Process Stats \[CPU\]](#) section of the report.

## [WATCHDOG SETUP cont.]

### Example 2: High Resource Utilization (AUTO MODE)

*“From time to time I have high resource utilization, but I don’t know why, neither I know when it’s going to happen.”*

It's a common scenario where we might not fully understand what's affecting our system, necessitating the collection of comprehensive information to diagnose the issue. Further, it's often the case that the timing of these issues is unpredictable. Under such circumstances, running the watchdog in AUTO mode proves to be highly effective. This mode continuously monitors the system, capturing valuable data when problems arise, thus providing a targeted approach to identifying and resolving the issue.

For the scenario above we will set up AUTO watchdog mode, this will trigger if any resource exceeds an 80% threshold.

```
Choose a mode for the Resource Watchdog:

1 - Auto   : Monitor CPU, memory, and disk at 80% Threshold. Duration: 60sec
2 - Manual : Choose resources and Thresholds. Duration: 1-300sec

Enter 1 or 2: 1
[root@rhel88 ~]# Resource watchdog started. PID: 46473
```

Auto Mode Selected

Like the previous example, we can see that it also generates an associated process that will act as a watchdog and monitor the associated resources.

Here, I manually allocated more than 80% of my system memory, and immediately the watchdog initiated a capture. The default duration for AUTO mode is 60 seconds and it can't be changed.

```
[root@rhel88 ~]# Starting data capture for 60 seconds ...

Starting
Gathering general system information ( ... )
Initializing performance capture
```

Captured triggered by Auto Mode

## [CRONJOB SETUP]

### 5.2.3 MODE 3 – CRONJOB

The Cron feature of the Linux AIO Performance Checker is particularly designed for instances where a user can predict when a system issue will occur. Upon selection of this feature, users will be prompted to specify whether the cronjob should be set up to run recurrently (such as every hour or every day) or just once. The cronjob is then configured and executed in accordance with the user's preference. This functionality ensures that data collection coincides with the anticipated occurrence of the system issue, thereby facilitating efficient problem identification and resolution.



**WARNING:** cronjobs are not automatically removed once added via the script. It's the user responsibility to manually delete the created cronjobs using "crontab -e".

### SETUP CRONJOB – RECURRENT

Recurrent cronjob will repeat the report collection with a specific interval. Example: Collect every 5 minutes.

- **Do you want the job to run recurrently?** Yes
- **Enter the hour - Every (0-23) hours:** Specify at which hour of the clock
- **and (0-60) Minutes:** Specify at which minutes of the clock
- **Duration:** Report collection duration in seconds

```
no crontab for root
Cron job added: 0 */10 * * * /root/linux_aio_perfcheck.sh --collect-now 10
Restarting chronyd service ...
chronyd service restarted.
```

Cronjob output from Cronjob setup



**ATTENTION:** Once the cronjob is set up, a cron entry will be added to root user crontab which will collect at the time specified by the user.

## [CRONJOB SETUP cont.]

## SETUP CRONJOB – AT SPECIFIC TIME

A fixed custom time for collection can be specified. In this case, the report will be collected based on the single cron entry:

The user will be prompted to enter the scheduled based on the UNIX standard cron entry system:

```
# * * * * * command to execute
# |
# | |
# | | |
# | | | |
# | | | | day of week (0 - 7)
# | | | | month (1 - 12)
# | | | | day of month (1 - 31)
# | | | | hour (0 - 23)
# | | | | min (0 - 59)
```

## How to work with crontab Format



**Hint:** If you need some visual assistance setting up the time in the right crontab format, visit <https://crontab.guru/> the quick and simple editor for cron schedule expressions by Cronitor.

- **Do you want the job to run recurrently?** No
- **Insert Time:** Use \* \* \* \* format illustrated above.
- **Duration** Report collection duration.

This will setup a cron entry based on the time that you requested on the user root crontab:

```
Cron job added: 30 2 * * * /root/linux_aio_perfcheck.sh --collect-now 20
Restarting chronyd service...
chronyd service restarted.
```

### Cronjob setup based on specific - at time

## [USING THE WEB APPLICATION]

### 6. USING THE WEB APPLICATION

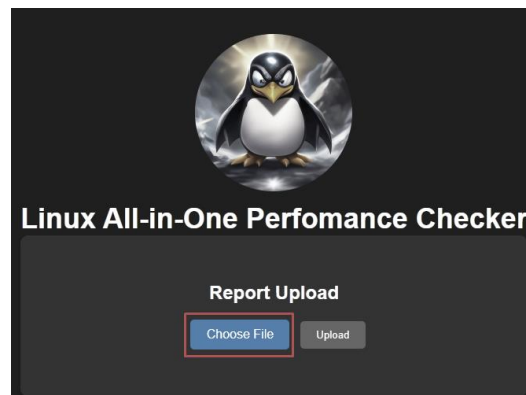
The Linux AIO Performance Checker is designed not only to collect valuable performance data but also to present it in a manner that is easy to understand. The report, once generated, can be uploaded to a specially designed web application. This application presents the data in a user-friendly format, making it simple for users to navigate and comprehend.

#### 6.1 ACCESS THE WEB APPLICATION

To access the web application visit: <https://linuxaioperf.matildes.dev/>

#### 6.2 UPLOADING DATA

On the main page of the web app, there's an upload form. Here you will need to hit "Choose File" and select the tarball that was generated by the collector script.



Web App Homepage



Only upload the generated tarball file by the collector report. The web app will not process any other files or file formats rather than the ones generated by the bash script.

Please wait for the report to be fully loaded. It may take a while to load up all the counters, and you will see a message such as:

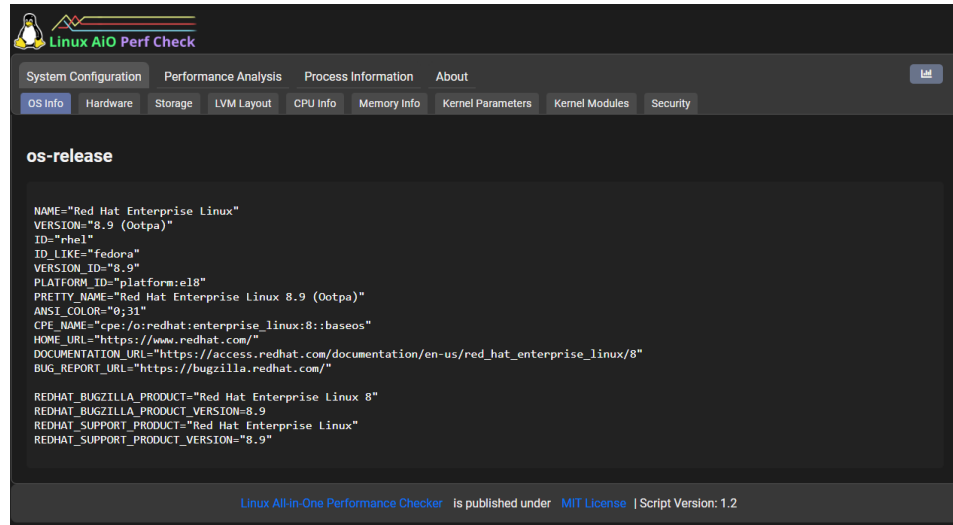
Digesting a bazillion counters, Please wait...

Loading the counters on the Web Report Page

## DATA ANALYSIS - REPORT

### 7. DATA ANALYSIS WITH THE WEB APP

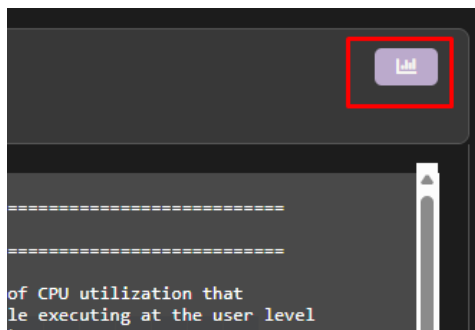
Once the report is loaded into the web application, the user will be presented with multiple tabs.



Initial Display on the Report Page

#### SECTIONS | Subsections

System Configuration	OS Info	Hardware	Storage	LVM Layout	CPU Info
	Memory Info	Kernel Parameters	Kernel Modules		Security
Performance Analysis	CPU Load Distribution Metrics/Device	Memory Utilization	Disk Counters	Disk Device/Metrics	Counters Network Stats
Process Information	Show the percentage of CPU utilization that occurred while executing at the system level (kernel).				
%iowait	Process stats [CPU] top	Process stats [IO] iotop	Process stats [Memory]		
%steal	Show the percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.				
%idle	Show the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.				



**HINT:** On the top right corner of the web page, there's a button. This button will collapse a box containing all counters used in the report together with their respective



## 7.1 SYSTEM INFORMATION TAB

From the system information tab, it's important to highlight that it contains raw OS info along with visual interpretation of LVM structure.

### 7.1.1 OS Release

Displays output of `/etc/os-release`

#### os-release

```
NAME="Red Hat Enterprise Linux"
VERSION="8.9 (Ootpa)"
ID="rhel"
ID_LIKE="fedora"
VERSION_ID="8.9"
PLATFORM_ID="platform:el8"
PRETTY_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:redhat:enterprise_linux:8::baseos"
HOME_URL="https://www.redhat.com/"
DOCUMENTATION_URL="https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 8"
REDHAT_BUGZILLA_PRODUCT_VERSION=8.9
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="8.9"
```

Example of OS-release output

### 7.1.2 Hardware

- `lshw`
- `dmidecode`

### 7.1.3 Storage

- `lsscsi`
- `lsblk -f`
- `df -h`
- `ls -l dev mapper`
- `parted -l`



**Hint:** You may combine the output of dev mapper output with `lsblk` to determine which device mapper number (e.g.: `dm-6`) corresponds to which mountpoint.

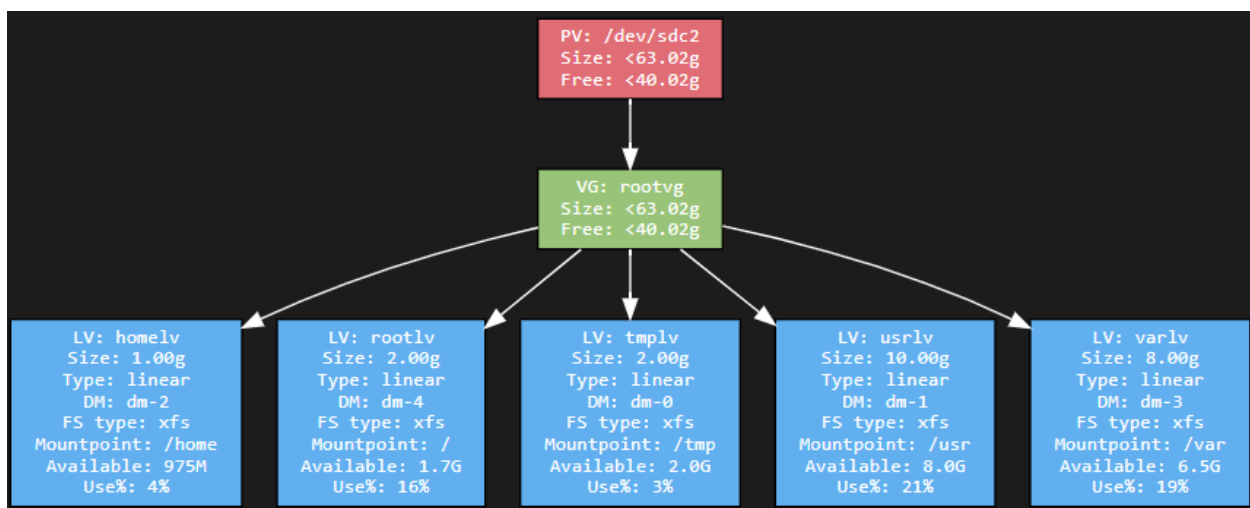
## SYSTEM INFORMATION TAB

### 7.1.4 LVM Layout

LVM Layout is a visual interpretation of the current LVM setup and will only be available if LVM is configured for the system that the report was collected.

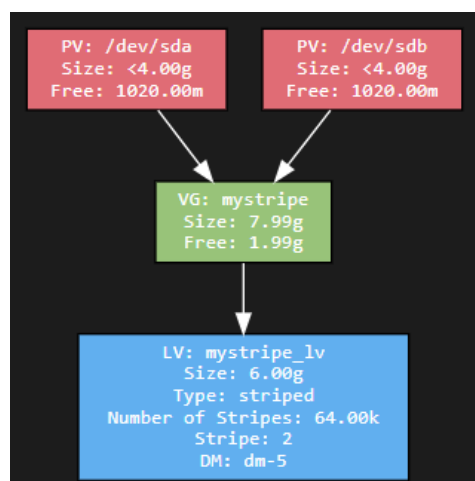
Visual representation contains an exhaustive description of volume details, including the fstype, file system utilization and mountpoint :)

#### Simple root volume



LVM Layout Diagram 1

#### RAID 0 – Stripe Volume



LVM Layout Diagram 2

## SYSTEM INFORMATION TAB

### 7.1.5 CPU Info

Displays content of `/proc/cpuinfo`

### 7.1.6 Memory Info

Displays content of `/proc/meminfo`

### 7.1.7 Kernel Parameters

Displays content of `"sysctl -a"`

### 7.1.8 Kernel Modules

Displays the content of `"lsmod"`

### 7.1.9 Security

Security will show what Security modules are enabled in the system.

**For Debian Based:** AppArmor Status

**For RHEL/SLES:** SELinux Status

### SELinux Status

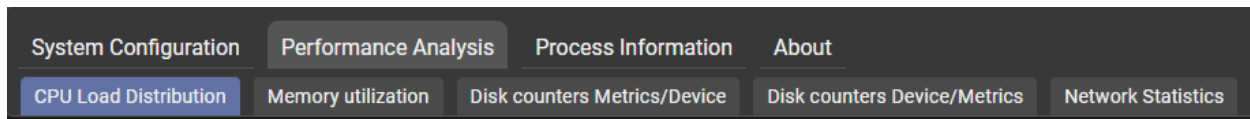
```
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Memory protection checking: actual (secure)
Max kernel policy version: 33
```

Example of Security Output for SELinux

## PERFORMANCE ANALYSIS TAB

### 7.2 PERFORMANCE ANALYSIS TAB

The Performance tab is a key feature in the Linux All in One performance and report tool. This tab provides a comprehensive overview of the real-time performance data captured during the report collection period. It is organized into four main sections: CPU, Memory, Disk IO, and Networking.



TAB Menu – Performance Analysis

1. **CPU:** This section displays the processor usage details of your system. It provides a real-time representation of the workload on your system's processor(s). This includes details such as the percentage of CPU used, which processes are using the most CPU, and how much CPU time each process has consumed.
2. **Memory:** This section presents the memory usage details. It shows the total amount of memory, the used memory, and the free memory in your system.
3. **Disk IO:** This section outlines the Input/Output operations on your system's storage devices. It offers insights into read and write speeds, which processes are performing these operations, and how much data is being read or written. This can help to identify any potential bottlenecks in your system's performance.
4. **Networking:** This section provides an overview of your system's network connections. It displays information about incoming and outgoing network traffic, including data transfer rates, connection statuses, and which processes are generating network traffic. This can be useful to identify any unusual network activity or to help optimize network performance.
- 5.

Each section provides a detailed analysis, allowing you to monitor the performance of your Linux system efficiently. By understanding the data presented in the Performance tab, you can take appropriate actions to maintain optimal system performance.



**IMPORTANT:** The time granularity for all metrics is 1 second. This ensures that we have per-second samples, increasing the overall chance to catch spikes and uncommon resource heaters. You may adjust the X-axis scale using the zoom feature to see the per-second samples.

All plots are generated using plotly.js CDN, more information can be found at <https://plotly.com/>

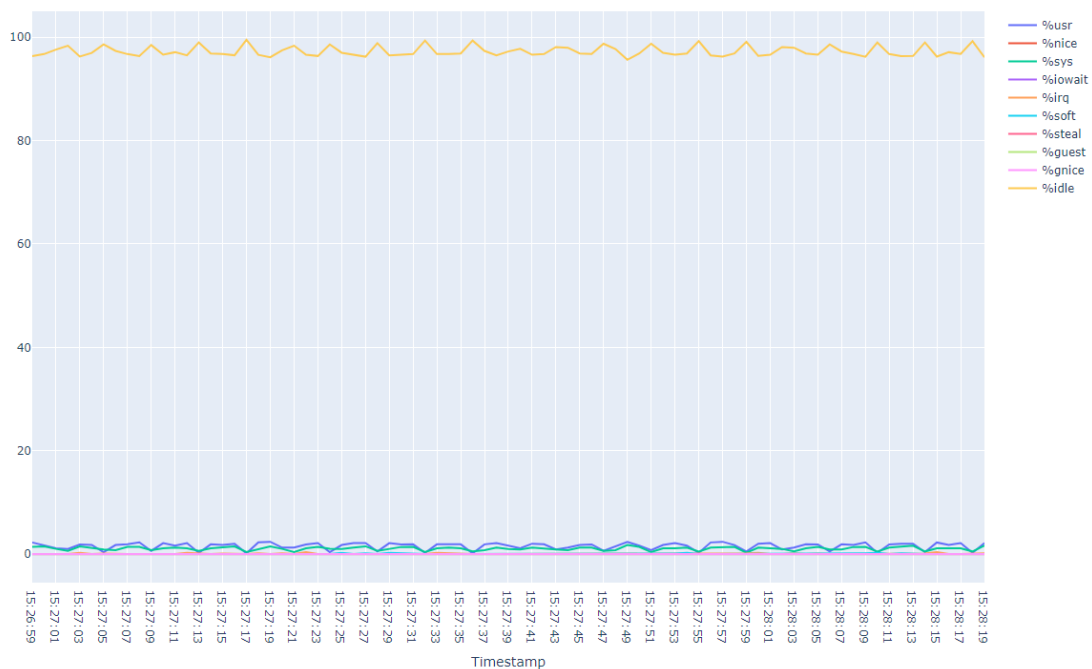
## PERFORMANCE ANALYSIS TAB - CPU LOAD DISTRIBUTION

### 7.2.1 CPU LOAD DISTRIBUTION

The first plot generated is the CPU Utilization Report. For multiprocessor systems, the CPU values are global averages among all processors. The plot has the following format:

%user	Show the percentage of CPU utilization that occurred while executing at the user level (application).
%nice	Show the percentage of CPU utilization that occurred while executing at the user level with nice priority.
%system	Show the percentage of CPU utilization that occurred while executing at the system level (kernel).
%iowait	Show the percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.
%steal	Show the percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.
%idle	Show the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

All CPU Usage Data



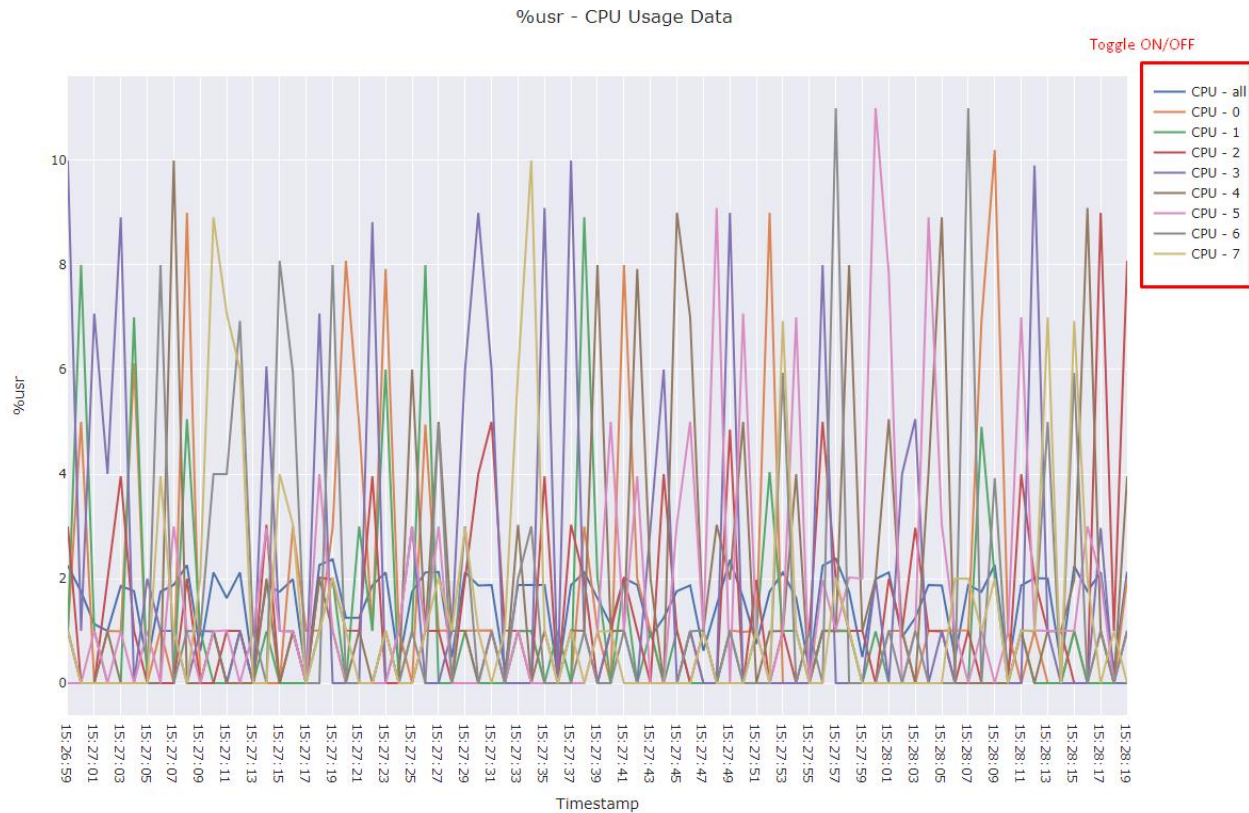
CPU Usage Example

## PERFORMANCE ANALYSIS TAB – CPU LOAD DISTRIBUTION

Then, after the main “ALL CPU Usage Data”, we will have a plot per CPU load type.

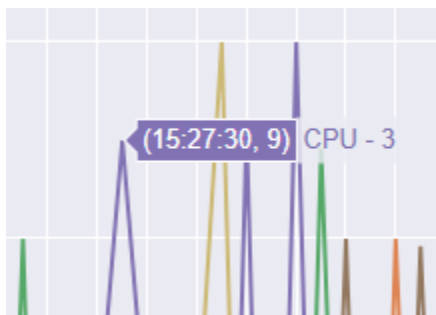
The plots that follow are more detailed, and contain per-core load distribution which can be toggled on/off from the right menu.

Here’s an example of %usr CPU time:



CPU %usr Example

Hovering the mouse pointer on top of a line gives us the CPU core and TIMESAMP and can be zoomed in if desired.



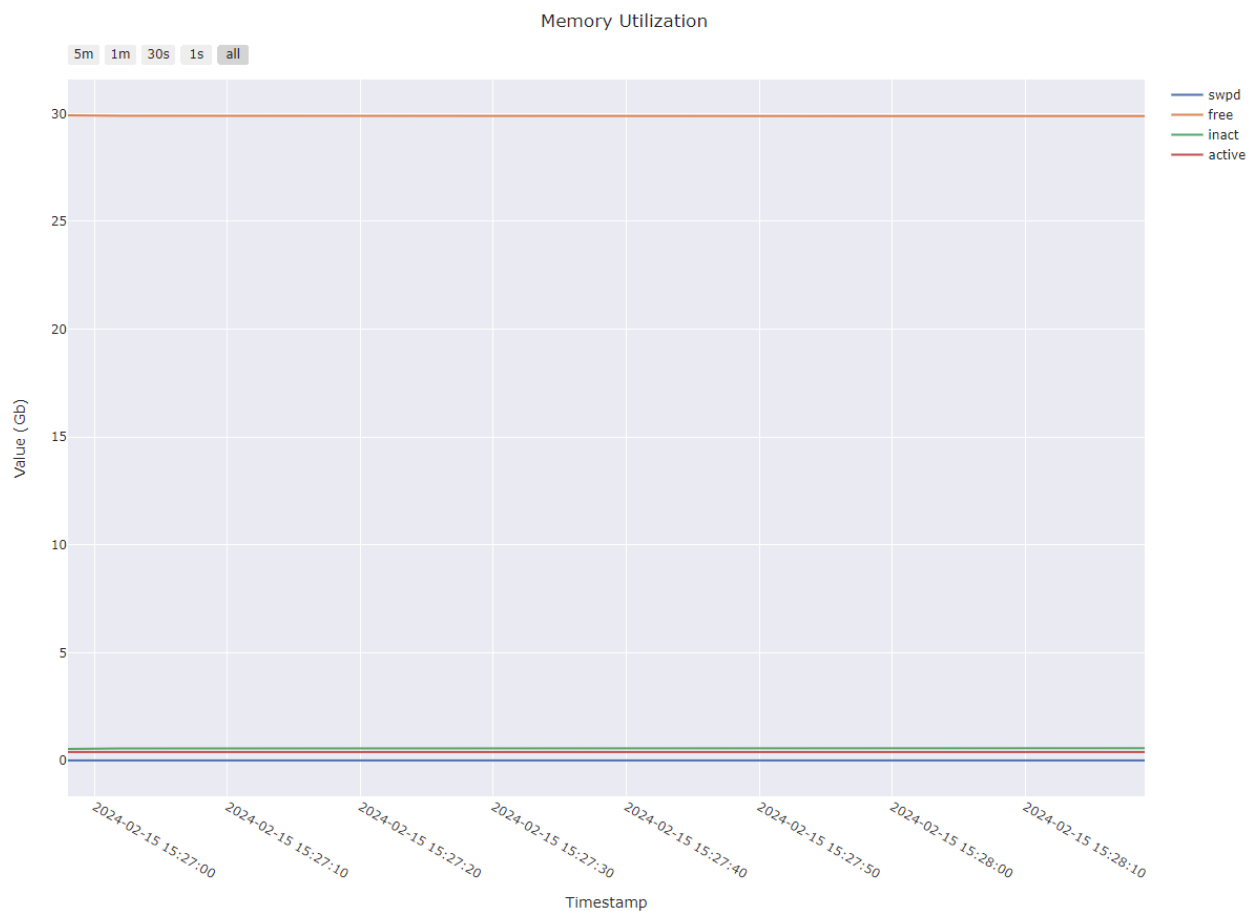
TIMESTAMP: 15:27:30  
UTILIZATION: 9%  
CPU CORE: 3

## PERFORMANCE ANALYSIS TAB – MEMORY UTILIZATION

### 7.2.2 MEMORY UTILIZATION

The memory utilization plot shows the following memory utilization values:

<code>swpd</code>	The amount of swap memory used.
<code>free</code>	The amount of idle memory.
<code>buff</code>	The amount of memory used as buffers.
<code>cache</code>	The amount of memory used as cache.
<code>inact</code>	The amount of inactive memory.
<code>active</code>	The amount of active memory



Memory Utilization Plot Example



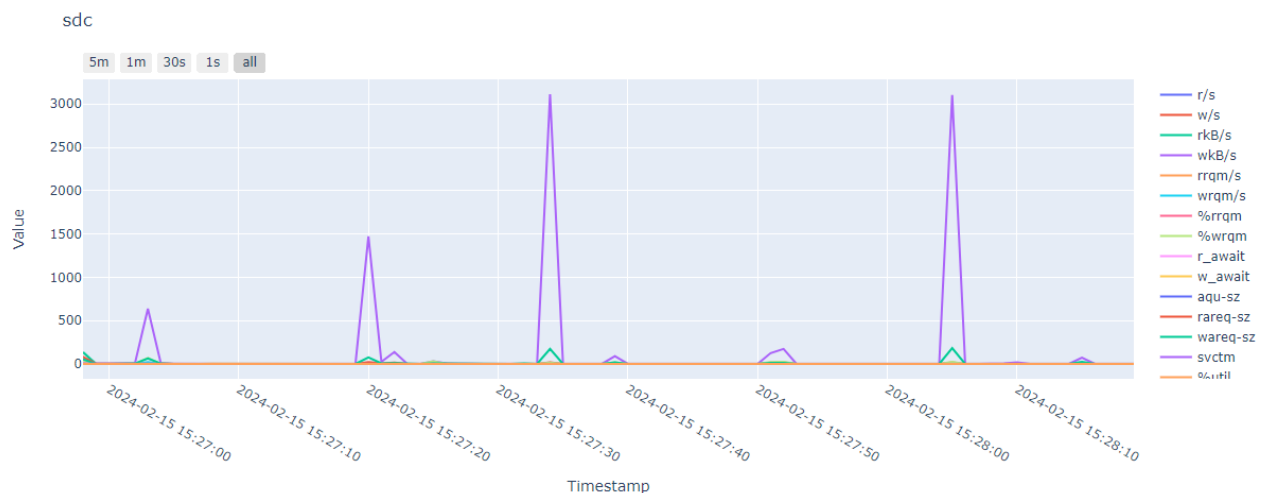
**Hint:** To better understand how Linux manages system memory, the author suggests a visit to the following website: <https://www.linuxatemyram.com/>

## PERFORMANCE ANALYSIS TAB – DISK METRICS PER DEVICE

### 7.2.3 DISK COUNTERS METRICS PER DEVICE

Here we cover the Disk device metrics per Disk, meaning that we will have a single Plot per disk, and multiple lines per disk counters:

r/s	The number of read requests that were issued to the device per second.
w/s	The number of write requests that were issued to the device per second.
rkB/s	The number of kilobytes read from the device per second.
wkB/s	The number of kilobytes written to the device per second.
rrqm/s	The number of read requests merged per second that were queued to the device.
wrqm/s	The number of write requests merged per second that were queued to the device.
%rrqm	The number of read requests merged per second that were queued to the device.
%wrqm	The number of write requests merged per second that were queued to the device.
r_wait	The average time (in milliseconds) for read requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
w_wait	The average time (in milliseconds) for write requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
aqu-sz	The average queue length of the requests that were issued to the device. Note: In previous versions, this field was known as avgqu-sz.
rareq-sz	The average size (in kilobytes) of the read requests that were issued to the device.
wareq-sz	The average size (in kilobytes) of the write requests that were issued to the device.
svctm	The average service time (in milliseconds) for I/O requests that were issued to the device. Warning! Do not trust this field any more. This field will be removed in a future sysstat version.
%util	Percentage of CPU time during which I/O requests were issued to the device (bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.



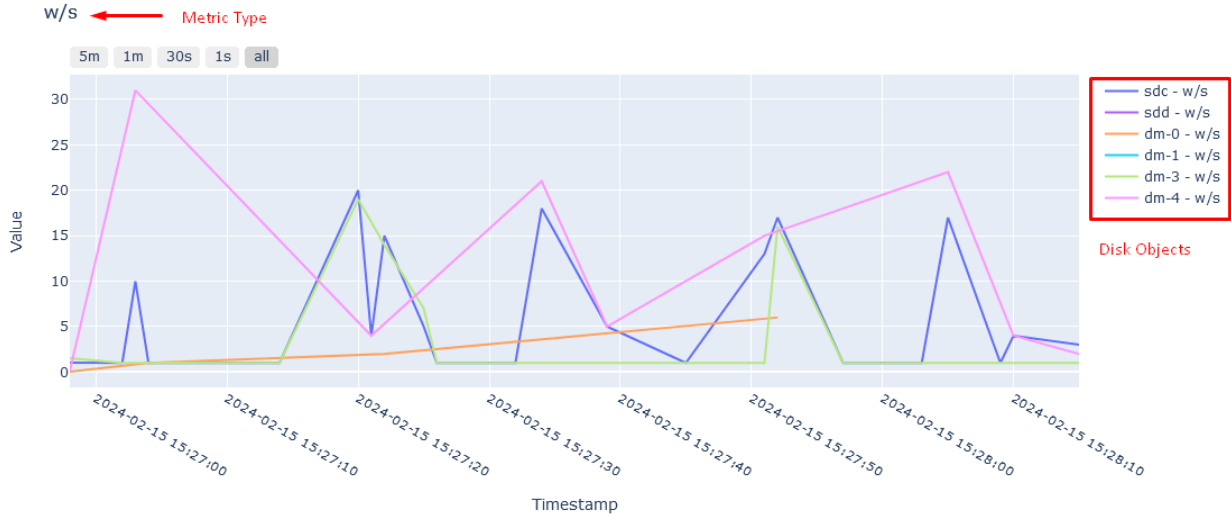
All metrics per single device



## PERFORMANCE ANALYSIS TAB – DISK DEVICE PER METRICS

### 7.2.4 DISK COUNTERS DEVICE PER METRICS

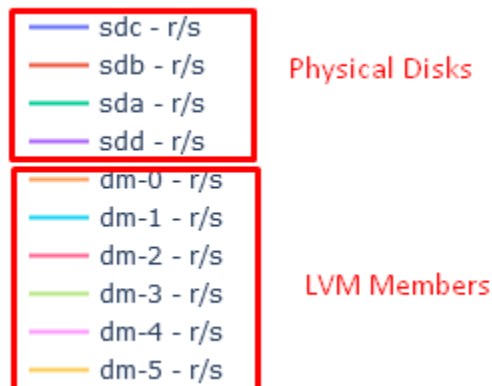
On this one, we will have the inverse of the previous tab, where we create a single plot per metric type and populate the disk objects per line.



Single metrics per all devices



**Hint:** The disks (physical) are represented usually by sdX naming convention. While LVM members are shown as dm-X. Refer to [LVM Layout](#).

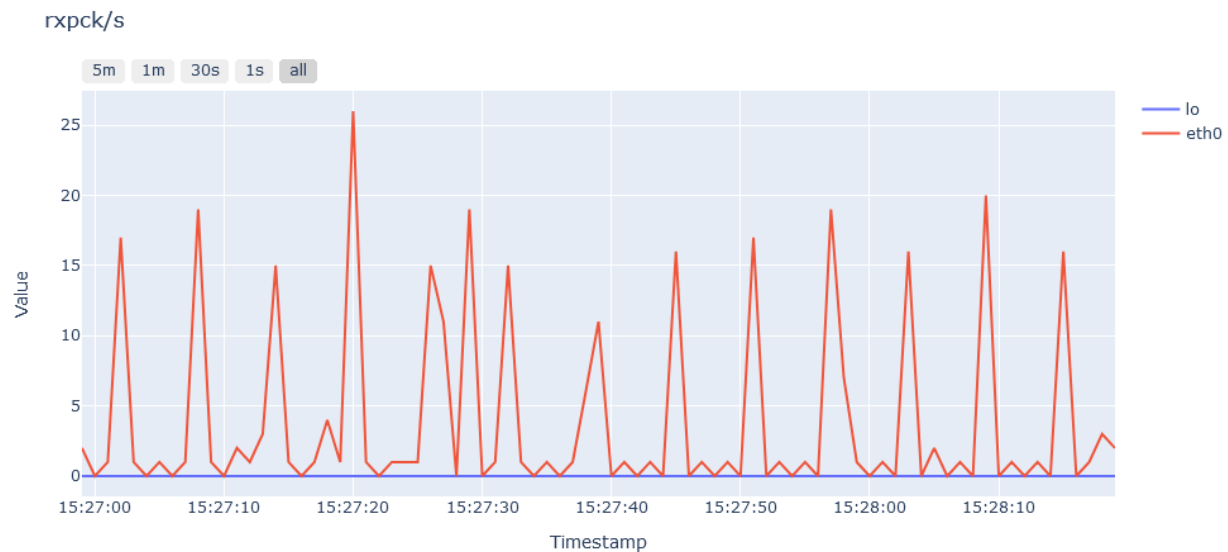


## PERFORMANCE ANALYSIS TAB – NETWORK STATISTICS

### 7.2.5 NETWORK STATISTICS

The network statistics page will display Network counters captured during the report collection. Each counter description is displayed on the table below:

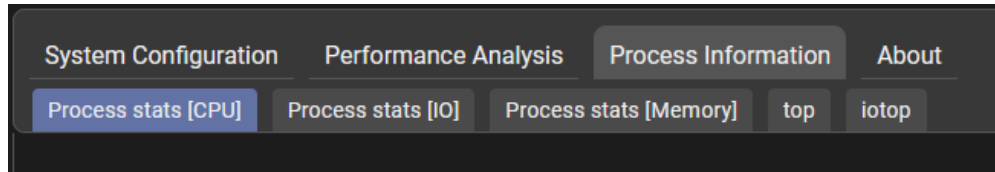
<b>rxpck/s</b>	Total number of packets received per second.
<b>txpck/s</b>	Total number of packets transmitted per second.
<b>rxkB/s</b>	Total number of kilobytes received per second.
<b>txkB/s</b>	Total number of kilobytes transmitted per second.
<b>rxcmp/s</b>	Number of compressed packets received per second (for cslip etc.).
<b>txcmp/s</b>	Number of compressed packets transmitted per second.
<b>rxmcast/s</b>	Number of multicast packets received per second.
<b>%ifutil</b>	Utilization percentage of the network interface. For half-duplex interfaces, utilization is calculated using the sum of rxkB/s and txkB/s as a percentage of the interface speed. For full-duplex, this is the greater of rxkB/s or txkB/s.



Network Packets Received per Second for loopback and eth0 Network Interfaces

## PROCESS INFORMATION TAB

### 7.3 PROCESS INFORMATION TAB



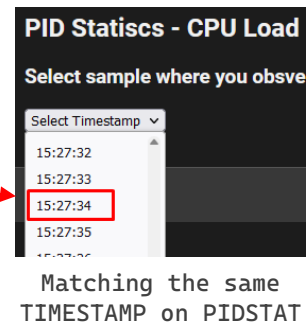
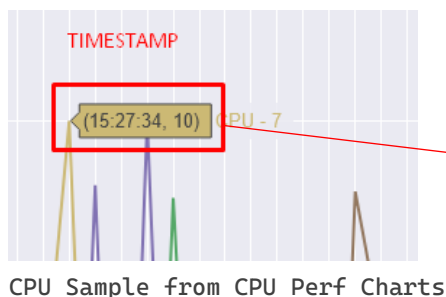
"Process Information" serves as the control center for monitoring all process-related metrics. The comprehensive display covers output from multiple **pidstat**, **top**, and **iotop** batch samples, offering a real-time snapshot of your system's operations. The granularity of information is fine-tuned to a **one-second interval**, enabling you to draw precise correlations with performance metrics from all other associated tabs. For instance, high CPU usage in the CPU Stats can be cross-verified with **pidstat [CPU]** to pinpoint the exact process that's over-consuming the CPU resources.

This tab has been meticulously structured to address the final '**W**' of our 3 '**W**'s approach - the '**Who**'. This refers to the identification of the '**culprit**' process that may be causing system slowdowns or disruptions. By accurately identifying the source of the problem, you can take appropriate actions to rectify the issue and maintain optimal system performance.



**Important Information:** To maximize the benefits of the "Process Information" feature, it is essential to understand the crucial role of **TIMESTAMP**. This element is key in effectively correlating performance counters with actual process information. **TIMESTAMP** provides a time marker for the performance charts, documenting the exact moment when a particular activity or data point was recorded.

This marker can be extremely useful when you need to track the performance of a specific process over a period of time or identify patterns in system behavior. Once you've identified the **TIMESTAMP** from the performance charts, you can then select it from the associated Process Stats combobox. This action will display a comprehensive breakdown of all process-related activities that occurred around that **TIMESTAMP**.



## PROCESS INFORMATION TAB

### 7.3.1 PROCESS STATS [CPU]

The Process Stats tab displays a variety of pidstat CPU related information, giving you a comprehensive look at how your system's processes are interacting with the CPU. The counters displayed in this tab include:

UID	The real user identification number of the task being monitored.
USER	The name of the real user owning the task being monitored.
PID	The identification number of the task being monitored.
%usr	Percentage of CPU used by the task while executing at the user level (application), with or without nice priority. Note that this field does NOT include time spent running a virtual processor.
%system	Percentage of CPU used by the task while executing at the system level (kernel).
%guest	Percentage of CPU spent by the task in virtual machine (running a virtual processor).
%wait	Percentage of CPU spent by the task while waiting to run.
%CPU	Total percentage of CPU time used by the task. In an SMP environment, the task's CPU usage will be divided by the total number of CPU's if option -I has been entered on the command line.
CPU	Processor number to which the task is attached.
Command	The command name of the task.



**Hint:** The output will highlight the following:

CPU Utilization Percentage above 60%

**YELLOW**

CPU Utilization Percentage above 80%

**RED**

**Output is ordered by PID number!**

PID Statistics - CPU Load Distribution												
Select sample where you observed high CPU utilization												
15:27:11												
Timestamp	UID	PID	%usr	%system	%guest	%wait	%CPU	CPU	Command			
15:27:11	0	1	0.00	0.00	0.00	0.00	0.00	0.00	3	systemd		
15:27:11	0	2	0.00	0.00	0.00	0.00	0.00	0.00	0	kthreadd		
15:27:11	0	3	0.00	0.00	0.00	0.00	0.00	0.00	0	rcu_gp		
15:27:11	0	4	0.00	0.00	0.00	0.00	0.00	0.00	0	rcu_par_gp		
15:27:11	0	5	0.00	0.00	0.00	0.00	0.00	0.00	0	slub_flushwq		
15:27:11	0	7	0.00	0.00	0.00	0.00	0.00	0.00	0	kworker/0:0H-events_highpri		
15:27:11	0	10	0.00	0.00	0.00	0.00	0.00	0.00	4	mm_percpu_wq		
15:27:11	0	11	0.00	0.00	0.00	0.00	0.00	0.00	0	rcu_tasks_rude		
15:27:11	0	12	0.00	0.00	0.00	0.00	0.00	0.00	0	rcu_tasks_trace		
15:27:11	0	13	0.00	0.00	0.00	0.00	0.00	0.00	0	ksoftirqd/0		
15:27:11	0	14	0.00	0.00	0.00	0.00	0.00	0.00	1	rcu_sched		
15:27:11	0	15	0.00	0.00	0.00	0.00	0.00	0.00	0	migration/0		
15:27:11	0	16	0.00	0.00	0.00	0.00	0.00	0.00	0	watchdog/0		
15:27:11	0	17	0.00	0.00	0.00	0.00	0.00	0.00	0	cpuhp/0		
15:27:11	0	18	0.00	0.00	0.00	0.00	0.00	0.00	1	cpuhp/1		
15:27:11	0	19	0.00	0.00	0.00	0.00	0.00	0.00	1	watchdog/1		
15:27:11	0	20	0.00	0.00	0.00	0.00	0.00	0.00	1	migration/1		
15:27:11	0	21	0.00	0.00	0.00	0.00	0.00	0.00	1	ksoftirqd/1		

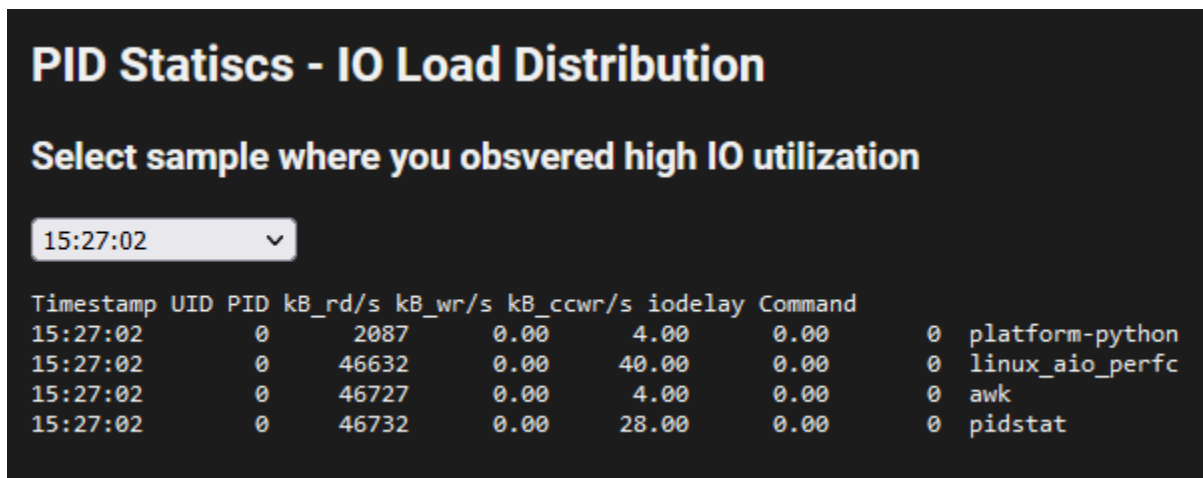
PID Stats - CPU Information

## PROCESS INFORMATION TAB

### 7.3.2 PROCESS STATS [IO]

The Process Stats IO tab provides a comprehensive view of pidstat I/O-related information, allowing you to monitor how your system's processes are interacting with the Input/Output operations. The counters displayed in this tab include:

UID	The real user identification number of the task being monitored.
USER	The name of the real user owning the task being monitored.
PID	The identification number of the task being monitored.
kB_rd/s	Number of kilobytes the task has caused to be read from disk per second.
kB_wr/s	Number of kilobytes the task has caused, or shall cause to be written to disk per second.
kB_ccwr/s	Number of kilobytes whose writing to disk has been cancelled by the task. This may occur when the task truncates some dirty pagecache. In this case, some IO which another task has been accounted for will not be happening.
iodelay	Block I/O delay of the task being monitored, measured in clock ticks. This metric includes the delays spent waiting for sync block I/O completion and for swapin block I/O completion.
Command	The command name of the task.



PID Stats - IO Information

## PROCESS INFORMATION TAB

### 7.3.3 PROCESS STATS [MEMORY]

The Process Stats Memory tab gives a detailed overview of pidstat Memory-related information, allowing you to monitor how your system's processes are utilizing memory resources. The counters displayed in this tab include:

UID	The real user identification number of the task being monitored.
USER	The name of the real user owning the task being monitored.
PID	The identification number of the task being monitored.
minflt/s	Total number of minor faults the task has made per second, those which have not required loading a memory page from disk.
majflt/s	Total number of major faults the task has made per second, those which have required loading a memory page from disk.
VSZ	Virtual Size: The virtual memory usage of entire task in kilobytes.
RSS	Resident Set Size: The non-swapped physical memory used by the task in kilobytes.
%MEM	The tasks's currently used share of available physical memory.
Command	The command name of the task.



**Hint:** The output will highlight the following:

MEMORY Utilization Percentage above 60%

**YELLOW**

MEMORY Utilization Percentage above 80%

**RED**

**Output is ordered by PID number!**

## PID Stats - Memory Load Distribution

Select sample where you observed high Memory utilization

15:27:01 ▼

Timestamp	UID	PID	minflt/s	majflt/s	VSZ	RSS	%MEM	Command
15:27:01	0	2087	36.00	0.00	444060	32440	0.10	platform-python
15:27:01	0	46632	251.00	0.00	17388	3488	0.01	linux_aio_perfc
15:27:01	0	46734	2.00	0.00	13576	7584	0.02	pidstat
15:27:01	0	46771	3254.00	0.00	65876	14212	0.04	iotop

PID Stats - Memory Information

## PROCESS INFORMATION TAB

### 7.3.4 TOP

The TOP Output tab provides a real-time dynamic view of the running system and displays a list of current tasks managed by the Linux kernel. It gives an ongoing look at processor activity in real time and shows information concerning the most CPU-intensive tasks, which can be sorted by CPU usage, memory usage and runtime. The counters displayed in this tab include:

```
top output - sampled

Select sample bellow

15:26:59 ▼

top - 15:26:59 up 4:21, 3 users, load average: 9.95, 9.64, 9.30
Tasks: 252 total, 2 running, 242 sleeping, 0 stopped, 8 zombie
%Cpu(s): 0.0 us, 0.8 sy, 0.0 ni, 99.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 31876.3 total, 30614.0 free, 524.1 used, 738.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 30958.7 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 46752 root        20   0   58716    4356   3600 R   6.2   0.0   0:00.01 top
    1 root        20   0 241496   10576   5220 S   0.0   0.0   0:04.39 systemd
    2 root        20   0      0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    5 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
    7 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
   10 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
   11 root        20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
   12 root        20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
```

TOP Output (Batch collected)

## PROCESS INFORMATION TAB

### 7.3.5 IOTOP

The iotop tab provides a real-time monitor that identifies the processes which are making significant read/write operations to your storage. It provides a valuable view of disk I/O usage by process or thread on your system. The counters displayed in this tab include:

iotop displays columns for the I/O bandwidth read and written by each process/thread during the sampling period. It also displays the percentage of time the thread/process spent while swapping in and while waiting on I/O. For each process, its I/O priority (class/level) is shown. In addition, the total I/O bandwidth read and written during the sampling period is displayed at the top of the interface.

```
iotop output - sampled

Select sample bellow - useful to understand which process is on top the Disk

Thu Feb 15 15:26:59 UTC 2024 ▼

Thu Feb 15 15:26:59 UTC 2024
Total DISK READ : 0.00 B/s | Total DISK WRITE : 0.00 B/s
Actual DISK READ: 0.00 B/s | Actual DISK WRITE: 0.00 B/s
  TID  PRIO  USER    DISK READ  DISK WRITE  SWAPIN      IO    COMMAND
b'    1 be/4 root      0.00 B/s   0.00 B/s ?unavailable? systemd --switched-root --system --deserialize 17'
b'    2 be/4 root      0.00 B/s   0.00 B/s ?unavailable? [kthreadd]'
b'    3 be/0 root      0.00 B/s   0.00 B/s ?unavailable? [rcu_gp]'
b'    4 be/0 root      0.00 B/s   0.00 B/s ?unavailable? [rcu_par_gp]'
b'    5 be/0 root      0.00 B/s   0.00 B/s ?unavailable? [slub_flushwq]'
b'    7 be/0 root      0.00 B/s   0.00 B/s ?unavailable? [kworker/0:0H-events_highpri]'
b'   10 be/0 root      0.00 B/s   0.00 B/s ?unavailable? [mm_percpu_wq]'
b'   11 be/4 root      0.00 B/s   0.00 B/s ?unavailable? [rcu_tasks_rude_]
b'   12 be/4 root      0.00 B/s   0.00 B/s ?unavailable? [rcu_tasks_trace]
b'   13 be/4 root      0.00 B/s   0.00 B/s ?unavailable? [ksoftirqd/0]'
```

IOTOP Output



## 8. ABOUT

---

The Linux All-In-One Performance Report Checker is a powerful tool designed to monitor and optimize the performance of your Linux system. It provides a comprehensive range of features that allow users to track and assess various performance metrics, including CPU usage, memory utilization, I/O operations, and process statistics.

This tool integrates several Linux utilities such as `pidstat`, `top`, `iostat` into a single platform, offering real-time and batch mode monitoring of system resources. The data is presented in an easy-to-understand format, complete with charts and graphs, making it simple to identify patterns and potential issues.

The Linux All-In-One Performance Report Checker was designed with a 3 'W's approach – 'What', 'When', and 'Who'. It helps you understand what is happening in your system, when it's happening, and who (which process) is responsible.

The tool's in-depth analysis capabilities allow you to pinpoint resource-heavy processes, monitor their impact on your system, and take necessary actions to optimize performance. The granularity of information is fine-tuned to a one-second interval, enabling users to draw precise correlations and make informed decisions.

The Linux All-In-One Performance Report Checker is an invaluable aid for system administrators, developers, and Linux enthusiasts who are keen on maintaining the optimal performance of their systems. It offers a user-friendly interface and a rich set of features that cater to both novice users and experienced professionals.

We are constantly working to improve and update the Linux All-In-One Performance Report Checker, ensuring it stays up-to-date with the latest advancements in Linux performance monitoring. We welcome feedback and suggestions from our users to make this tool even better.

Thank you for choosing Linux All-In-One Performance Report Checker as your trusted partner in Linux performance management.

## Author

Samuel Matildes

<https://www.linkedin.com/in/samuelpmatildes/>

<https://github.com/samatild/>

