Analysis of Linux System Performance through CLI Tools

[1]Kushalaa Hande, [2]Sahana B, [3]Dr. Abhay A. Deshpande

[1]Student, [2]Assistant Professor, [3]Associate Professor
[1]Electronics and Communication,
[1]R.V College of Engineering, Bangalore, India.

Abstract : Linux operating system is vast as well as a very powerful operating system. The performance of such systems becomes crucial for running various applications. Bottlenecks in performance would ultimately lead to failure of the system. Hence, assessment of system performance is done to ensure that all the resources are utilized optimally. The system performance is given in terms of efficiency, accuracy and speed of execution of the program instructions. When it comes to Linux, there are several distributions such as Ubuntu, OpenSuse, CentOS and Fedora. With over 600 distributions available in Linux, command line interface tools provide a simple, standard platform for extracting system performance parameters. Thus, various tools are used to extract performance parameters such as CPU utilization, memory usage, network bandwidth, disk utilization and hardware description. Analysis of these parameters gives the overall performance of the system and helps pinpoint the bottlenecks causing degradation of the system. Once the causes for bottlenecks are found, solution strategies can be implemented accordingly to resolve the depreciation of performance.

***IndexTerms* - CLI, Linux, system performance.**

I. INTRODUCTION

A variety of operating systems have been developed and updated consistently to meet the requirements of the current technology. Linux is one of the most widely used open source operating system which has different distributions such as Ubuntu, Fedora, Suse and CentOS. The distributions have the same underlying kernel but differ in terms of the conventional software being used to interact with the hardware. The conventional software includes system technologies such as package managers, display servers and desktop environments. Thus, a variety of Linux operating systems are available for use based on the application requirements. With the existence of many such systems, it becomes imperative to analyze and ensure optimal system performance.

Computer systems are becoming incredibly intricate with the growth in technology. System performance analysis becomes a prerequisite for understanding and tuning computer frameworks to perform tasks proficiently. Performance is the measure of useful work achieved by a system. Performance is also assessed based on efficiency, accuracy and speed of executing computer instructions. System performance centers around quantitative design criteria for installed applications, their resource usage and it's trade-off with expense. Covering an expansive scope of additional practical criteria, system performance may, for instance, identify with apparent nature of utilitarian results, timing behavior, storage space, bandwidth utilization, heat dissipation and energy consumption. Such cross-cutting criteria are among the most significant ones for systems as they are regularly a key selling factor. The performance of any system can be assessed in quantifiable, technical terms, utilizing at least one of the measurements. Along these lines, a comparison of the performance can be made relative with different systems or a similar system after changes. A cutting-edge system comprises of various software for running applications on an operating system that executes on multiple processors. The system bottlenecks could show up at different dimensions, or they could be a combination of various events happening at distinct levels. The CPU, I/O system, memory system, BIOS programming of the chipsets, the operating system parts, and the software stack all impact the general system performance. An ideal system has the product design and software mapped accurately to the hardware. An ideal system also provides high availability, low transmission time, appropriate response time, high throughput and low utilization of computing resources.
The analysis of Linux operating system performance can be done easily with the help of command line interface (CLI) tools. All the performance parameters are determined by the kernel. The CLI tools facilitate the extraction of the system performance for further examining. The command line is basically a shell which receives commands as text interface from the user and gives it to the operating system for further processing. Thus, by utilizing these tools, performance can be evaluated without the hassle of dealing with the kernel of the system.

The main contribution of the paper is the examining and evaluation of system performance in Linux operating systems through the shell. With a variety of operating systems made available to the user, the extraction procedure remains the same due to the utilization of command line interface tools. Thus, a standardized procedure is implemented which gives rise to a wide range of performance metrics. The analysis of the metrics helps to pinpoint the bottlenecks in the performance and evaluate the overall performance of the system.

The rest of the paper is organized as follows. Section II gives an overview on the various performance metrics and how these are analyzed in Linux environments. Section III describes the Linux operating system and the features provided by it. Section IV defines the various system performance parameters essential for the running of a system. The tools used for extracting the parameters and the analysis of the extracted system metrics are described in Section V and VI. Finally, section VII states the conclusions drawn from the extraction and analysis of system performance.

II. Related Work

Since extraction of metrics involves understanding of operating systems and their attributes, understanding the features and functionalities of such systems are essential. Different operating systems can be classified based on the performance using the TCP/IP protocol headers. [1] Another method for identification of the operating system comes from penetrating the firewall of the system and detect the remote host. [2] Since Linux is the system in focus, the aspects of the Linux kernel and its architecture, as well as the various Linux distributions are very important. [3]

Once a broad understanding of Linux as well as the evolution of networks in Linux is obtained [4], the performance of various systems of Linux is assessed. Performance of Linux operating systems needs to be analysed in depth to understand the metrics that are to be extracted and all the parameters which affect the overall performance of the system. This not only depends on the processes running in the system but also on the applications deployed on the system. There are several causes for OS latency, the main contributors are non-pre-emptive sections in the kernel or driver, timer resolution and scheduling jitter. The plan to overcome these problems is to use a pre-emptive lock-breaking Linux with a high-resolution timer mechanism, in order to implement a predictable scheduling algorithm. [5] Performance of hardware components in a Linux environment is also described and dependent on the kernel analysis. [6] Methodologies used to improve the real-time performance of a Linux operating system such as dual-kernel method, clock granularity, interrupt mechanism and real-time scheduling algorithm implementations are made in the kernel. [7] The kernel has a great impact on the performance of the system and it can easily retrieve the required software information. Thus, using the kernel, Linux software performance is also tested. [8]

As the number of high-performance computers are growing and expanding, the number of applications utilizing them increases. This means that the operating system now consumes a part of the system performance to manage the hardware resources and schedule tasks to ensure proper distribution of resources between several nodes of the high-performance computers. [9] Apart from this, the various requirements in a real-time environment are to be assessed as well. [10] The real-time performance of operating systems in a multithreaded simulation of complex Linux systems also helps understand how general-purpose and real-time operating systems function. [11] While monitoring performance, it is imperative that both the application and the executing system is evaluated. Further, the tool analyzing the performance must not require much effort to apply to an application. This means that modifications must not be done to the user application. [12] Therefore, the tools used in the command line interface must ensure that no modifications are done to the system while extracting the metrics.
[13], [14], [15] and [16] consist of studies and research documentation regarding the Linux shell, Linux operating system and the Bourne Again shell scripting platform. These concepts lay out a foundation on which the concepts are understood in depth.

III. Linux Operating System

Linux is a computer operating system. An operating system consists of the software that manages the computer and allows the execution of software on it. Linux is one of famous versions of UNIX operating systems It is open source as its source code is accessible to everyone. Linux was structured keeping the UNIX system compatibility on mind. The functionalities of Linux and UNIX are similar. Some of the features of Linux are as follows:

- Open Source: Linux code is a community-based development project that is readily available to all.
- File System: The Linux file system is that of a hierarchical file system that arranges files and directories in a systematic order.
- Application support: It has its own software repository from which many applications can be installed and downloaded.
- Multiuser capability: Multiple users are able to access the same system resources such as memory, hard disk, etc. But in order to operate they have to use different terminals.
- Multitasking: By intelligently splitting the CPU time, more than one function can be performed instantaneously.
- Portability: Portability provides support for various hardware types.
- Security: Security is provided in three ways namely authorization, authentication, and encryption.
- Graphical User Interface: Linux is an operating system which is based on command line, but by installing packages it can be transformed to a graphical user interface.

But Linux is a kernel, it does not form the complete operating system. Free software is used to create packages which is integrated with the Linux kernel to form Linux distributions. If Linux needs to be installed, a distribution has to be chosen. The fundamental framework of each distribution is similar, however the subsystems built around it are distinct in nature. Every variant is delivered by different organizations meeting their own requirements to achieve their goals. The outcome is a unique variant of Linux which is meant for a slightly different set of clients and users. Also called as distros, some of them are lightweight which are better suited for older systems with slower hardware components, while some of the others are easier to install and are more user friendly than others. Some distributions even require significant and in-depth knowledge about Linux in order to use the distribution. Thus, the vast number of features as well as choices of distributions provided by Linux provides a suitable background for the study of system performance parameters.

IV. System Performance Parameters

There are several resources available to the system and the various applications running on that system. The performance of the system is based on the utilization of the resources. Hence, system performance metrics need to be analyzed and monitored. Some the key performance metrics are discussed below.

4.1 CPU Utilization

CPU utilization is the sum of work dealt with by a Central Processing Unit and the usage of processing resources. It is used to evaluate system performance. CPU utilization can change as per the amount and type of computing task since certain tasks require substantial CPU time while others require less CPU time. Process time is another name for CPU time and is the measure of time utilized by a CPU for processing instructions of a computer program or an operating system. CPU time is evaluated in clock ticks or seconds. CPU utilization demonstrates the burden on a processor in terms of percentage. For instance, a heavy load with just a couple of running programs may show inadequate CPU power support, or running programs covered up by the system monitor which indicates potential malware in the system.

4.2 Memory Usage

The type as well as amount of memory installed in a system depicts the amount of data that can be processed altogether, in a single cycle. Larger RAMs and faster memory facilitate the system to perform at a high speed. Applications which overload the RAM end up providing marginally slower speeds. Therefore, the memory allocated to the system needs to be assessed.
Linux executes a demand-based paging system for the virtual memory framework. Processes are allocated large virtual memory space. The appropriate pages are transferred between physical memory and the disk whenever the virtual memory is called. The kernel swaps older pages back and forth to the disk when there are no physical memory pages available. Code pages which have not been altered are just discarded, otherwise, these are written to swap areas. Disk reads and writes are much slower than physical memory access because disks are mechanical devices. The kernel begins spending much more time swapping pages than spending time on execution of programs when memory pages exceed the physical memory present on the system. The system slows down and starts thrashing. If this continues till the swap device is completely used, the system will come to a virtual standstill. The kernel attempts to put extra physical memory which is not in use to work as a disk cache or buffer. Recently and frequently accessed disk data in the memory is stored in the disk buffer. If this data needs to be retrieved again, it is taken from the cache, thereby increasing the performance. Although the buffer is mainly used for paging, it can also be used to shrink and grow the memory dynamically to put to good use. Hence, all the memory is used wisely.

Thus, the used memory, buffer and cache memory, shared memory all play a vital is depicting the effect of memory on the overall performance of the system.

4.3 Network Throughput and Bandwidth

Performance of any network is evaluated using network throughput and bandwidth. Throughput is the name given to the measure of information that can be sent and received within a particular time period. At the end of the day, throughput estimates the rate at which messages reach the destination effectively. Packet delivery is effectively and practically measured using throughput. Normal throughput tells the client the quantity of packets arriving at the destination. Successful arrival of packets at the destination ensures a high-performance service. If heaps of packets are being lost during transmission and become ineffective, at that point the performance of the system and the network will be poor. Checking system throughput is vital for associations hoping to monitor the real-time performance of their system as well as network and ensure effective delivery of packets. More often than not, network throughput is estimated in bits per second but now and then it is also estimated in data packets every second. Network throughput is estimated as an average which depicts the overall performance of the network. Low throughput indicates packet loss problems.

Utilizing throughput to gauge network performance is valuable while troubleshooting since it helps pinpoint the main problems and causes associated with a slow network. In any case, it is only one of three factors that decide performance of the network. The other two are packet loss and latency. Packet loss is a term used to determine the quantity of packets lost during transmission within a network. Latency defines the time taken for a packet to be transmitted from the source to its destination. It can be estimated in various ways like a one-way transfer or round-trip time.

Network bandwidth is a proportion of how much information or data can be sent and received at a time. more data can be sent forward and backward as the bandwidth is increased. The term bandwidth isn't utilized to quantify speed but instead to gauge capacity. Bandwidth can be estimated in megabits per second, bits per second and gigabits per second. The key aspect about data bandwidth is that larger bandwidth does not assure high network performance. On the off chance that throughput in the network is being influenced by packet loss, latency and jitter then the network will see delays regardless of whether a considerable amount of bandwidth is accessible.

Lack of the required processing power or bandwidth to accomplish the task at hand during the communication between two or more devices leads to network bottlenecks. Overburdened network communication devices, overloaded servers and loss of integrity of the network are a few reasons for bottlenecks seen in the network. The issues can be resolved by upgrading network hardware like hubs, access points and routers. Further, servers can be upgraded and added to the network to ensure proper network utilization.

4.4 Disk Usage

Advancement in innovation of computing technologies have generally focused on handling processing power and very little on the I/O and storage part of the system. This is the reason the CPU and GPU have progressed by a wide margin while storage of a system like the hard disk drive has just progressed reasonably. Storage capacity has indeed improved drastically, yet the I/O performance of the hard disk is slow and cannot keep up with the power provided by the processor. This is a result of the distinction in hardware design. The CPU is absolutely electronic while the hard disk is electromechanical and is very restricted by its mechanical parts. New storage choices like the strong state drive plan to eradicate this gap in performance. Storage performance has become a bottleneck in computing systems and applications; thus, they need to be assessed and maintained.

One of the important parameters while considering disk usage is Input/Output Operations per Second (IOPS). IOPS measures the number of storage transactions processed through a system each second. Smaller data objects like web traffic logs can be measured using IOPS. The term IOPS stands for input/output operations per second. Frequently, this is a standout amongst the most touted measurements of any storage framework or storage cluster model. Particularly, when managing conventional storage clusters using HDD, IOPS were fundamentally significant as they were surely an indicator of the potential of performance in an array. The present drives can convey IOPS such that they result in a huge number of IOPS every second.

Another important metric to be considered for evaluating disk performance is disk throughput. It is a measure of what number of I/O activities can be done in a given time, usually in seconds. Numerous storage clusters are estimated in Megabyte per second. There are additionally typically two unique types for throughput namely, sustained and peak throughput. Sustained throughput is the measure of data that is constantly pushed by the device. Peak throughput measures the amount of data during burst times. It also describes the amount of data that flows through a point in the data path over a given amount of time. Throughput is usually the best storage metric when it comes to measurement of data that needs to be streamed quickly, such as videos and images.

Storage requirements can change among situations and are dependent on the particular application. A high performing storage environment can mean a wide range of things. Once in a while, high throughput or high IOPS mean high storage performance however it may not recount to the full story. Therefore, in addition to IOPS and throughput, latency must be utilized to quantify the performance of storage devices.

HDDs, SDDs and other long-term storage lead to bottlenecks in the disk usage as they are the slowest component within a server or a computer. Physical limits exist even for the fastest long-term storage devices. Thus, troubleshooting these bottlenecks are rather difficult. Reduction of fragmentation issues and augmenting data caching rates in RAM improves disk usage speed. Insufficient bandwidth should be addressed on a physical level. This is done by expanding RAID configurations and switching to faster storage devices.

4.5 Hardware Metrics

Performance of a system depends on the usage of resources, but it is the hardware and software capabilities which essentially enhance the system performance to achieve the required benchmarks. Without the correct hardware support, systems would collapse. The software performance depends on the programming of applications and the underlying operating system. Thus, hardware and software details of any system need to be analyzed thoroughly.

V. Command-Line Tools for Extracting System Performance

The extraction of the system performance metrics is done with the help of bash scripting through the command line interface. Bash scripting provides the feature of integrating various tools on the Linux platform which extracts the metrics. Apart from this, in Linux everything is a file system. Therefore, CPU, memory and process information are available in the */proc* file system. Thus, using bash scripts, the system metrics can be extracted. Various tools are used in bash scripting to extract the system performance metrics. These tools are available in repositories which can be downloaded and installed. Once installed, all the tools run the same on any Linux platform, thereby, providing a standard layout of assessment. The various tools used for extracting system metrics through the command line are described in the following section.

5.1 Tools for Extracting CPU Utilization

CPU information involves CPU utilization, load average and the number of cores, amongst several other parameters. Also, */proc/cpuinfo* file system also provides CPU information. Thus, some of the basic tools used for the extraction of CPU metrics are:

- Systems Activity Report (Sar): The Sar command line tool writes the contents of selected cumulative activity counters in the operating system for standard output. The accounting system writes, in seconds, the specified number of times spaced between the defined intervals based on the values in the count and interval parameters. If the parameter of the interval is set to null, the Sar command displays the average statistics for the time since the start of the system. If, without the count parameter, the interval parameter is specified, reports are generated continually. This command comes from the Sysstat package.
- Dstat: Dstat is a powerful, flexible and agile Linux system resource statistics tool. It comes with additional features, counters and is highly expandable, users with knowledge of Python can build their own plugins. Dstat allows a user to instantly view all the system resources, e.g. Compare disk usage in combination with IDE controller interrupts or directly compare network bandwidth numbers with disk throughput (at the same interval). Dstat also cleverly provides the most detailed column information and clearly shows the magnitude and unit of the output.
- Top: A dynamic real-time view of a running system is obtained by the top program as seen in Fig. 1. It can display system synopsis details as well as a list of tasks that the Linux kernel is currently managing. All user configurable are the types of system summary information displayed and the types, order and size of information displayed for tasks and that configuration can be made persistent through restarts. The program provides a limited interactive interface for manipulation of processes as well as a much larger personal configuration interface, thus, covering all aspects of its operation. And while top is mentioned throughout this document, the program can be renamed as per requirements. This new name, possibly an alias, is then displayed at the top and used when reading as well as writing a configuration file.
- Mpstat: The mpstat instruction writes to standard output tasks for each processor, with processor zero being the first. There are also reports of global average activities among all processors. If no activity is selected, the default report is the usage report of the CPU. When invoking the mpstat program, two sections of statistics are displayed. The first section shows the system configuration displayed when the command starts and when a system configuration change occurs. The second section reveals the usage statistics displayed in intervals and the values of these metrics are deltas from earlier intervals at any time.

Thus, various other commands are also used to extract the CPU metrics, mainly the CPU usage and load average of the system such as atop, htop, etc.

5.2 Tools for Extracting Metrics Depicting Memory Usage

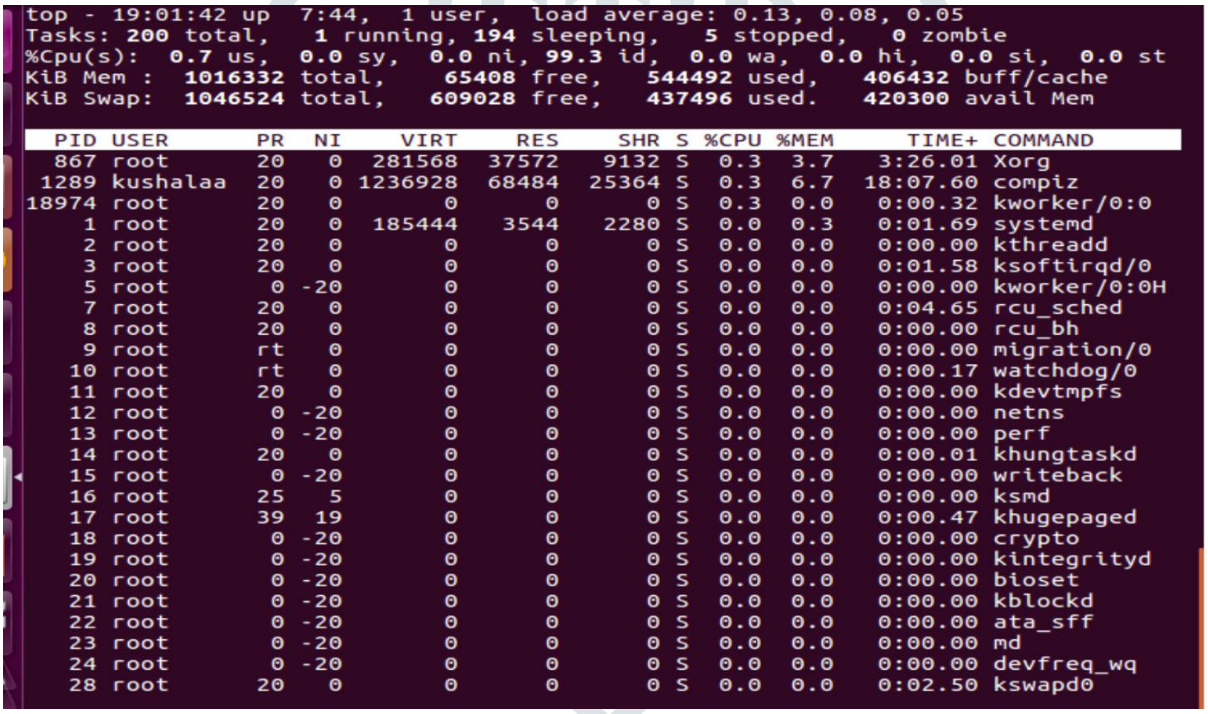
Memory usage is the main performance metric that needs to be analysed. Memory usage not only includes total memory, free memory and used memory but also involves active and inactive memory, buffer and cache memory, shared memory, etc. All these parameters need to be extracted and analysed accordingly as shown in Fig. 2. This is done through the use of command line tools. The tools essentially retrieve their information from */proc/meminfo* file system which contains all the memory related values. Thus, the various tools used for extracting memory information are as follows:

- Free: The total amount of free and used physical and swap memory in the system and the buffers used by the kernel are exhibited with the help of the free program. The column of shared memory should be ignored as it is obsolete.
- Virtual Memory Statistics (Vmstat): Vmstat has become a computer system monitoring tool that collects and displays summary on system memory, paging, interrupts, processes, and I / O block storages. Vmstat users can define a sampling interval that allows system activity to be directly measured in near-real time.

5.3 Tools for Extraction of Network Utilization

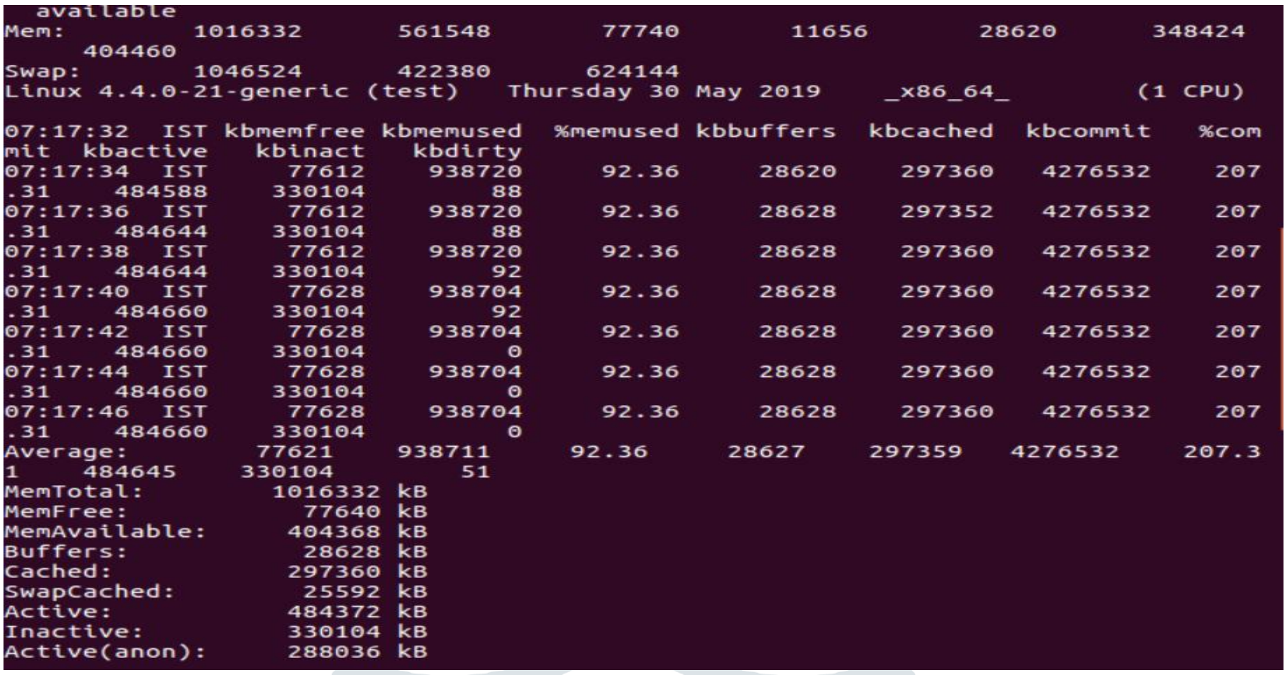
Network performance can be analyzed based on network bandwidth. Thus, network utilization is primarily extracted through the bandwidth being utilized by the system as depicted in Fig. 3. This is done with the help of tools such as:

- VnStat: VnStat is a Linux operating system network tool. It uses an interface for the command line. It keeps the specified interfaces with a log of hourly, daily and monthly network traffic, but is not a packet sniffer. It uses the kernel's network interface statistics as the source of information. This means that vnStat will not actually sniff any traffic and will also make sure that system resources are used lightly irrespective of network traffic rate. Traffic information from the proc filesystem will still be evaluated. This way, even without root permissions, vnStat may be used.
- Interface Statistics (IfStat): IfStat prints statistics based on the network interface. The tool keeps records of the historical data presented in history files and reveals only the difference between the last and the current call by default.



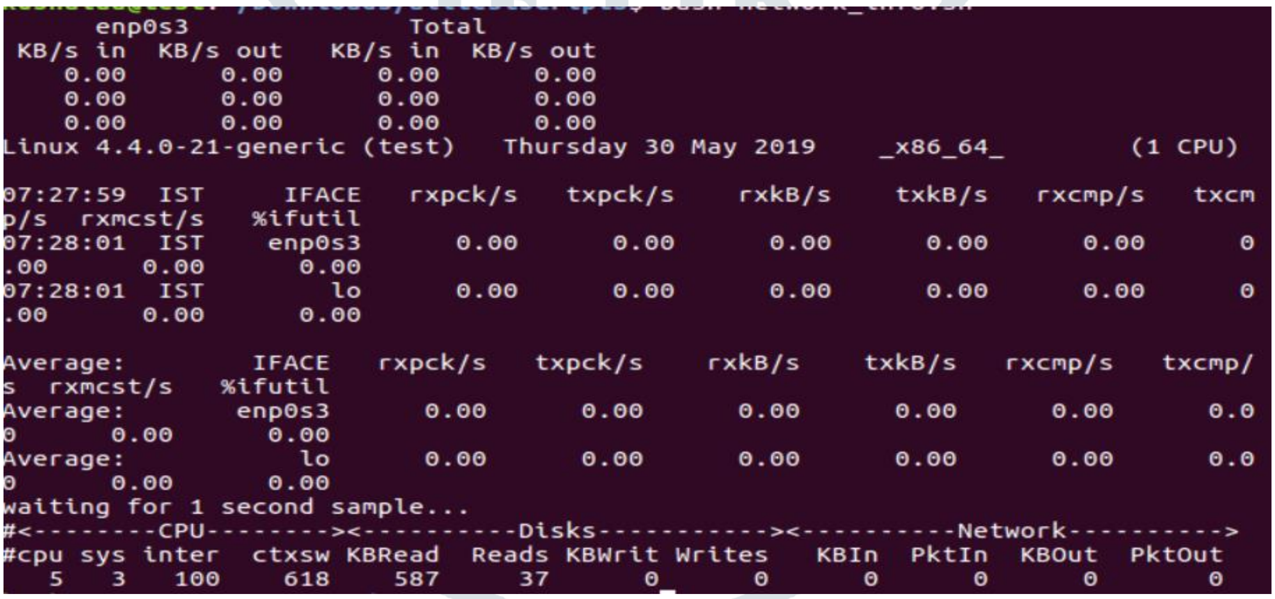
```
top - 19:01:42 up  7:44,  1 user,  load average: 0.13, 0.08, 0.05
Tasks: 200 total,   1 running, 194 sleeping,   5 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.0 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1016332 total,    65408 free,   544492 used,   406432 buff/cache
KiB Swap:  1046524 total,   609028 free,   437496 used.   420300 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  867 root      20   0  281568  37572   9132 S   0.3  3.7   3:26.01 Xorg
 1289 kushalaa  20   0 1236928  68484  25364 S   0.3  6.7  18:07.60 compiz
18974 root      20   0       0      0      0 S   0.3  0.0   0:00.32 kworker/0:0
    1 root      20   0  185444   3544   2280 S   0.0  0.3   0:01.69 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:01.58 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:04.65 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.17 watchdog/0
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   13 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 perf
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.01 khungtaskd
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   16 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   17 root      39  19       0      0      0 S   0.0  0.0   0:00.47 khugepaged
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 crypto
   19 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   22 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ata_sff
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 md
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 devfreq_wq
   28 root      20   0       0      0      0 S   0.0  0.0   0:02.50 kswapd0
```

Fig. 1. Output of Top for CPU utilization



```
  available
Mem:        1016332      561548      77740       11656       28620       348424
     404460
Swap:       1046524      422380      624144
Linux 4.4.0-21-generic (test)   Thursday 30 May 2019     _x86_64_        (1 CPU)

07:17:32  IST kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %com
mit  kbactive   kbinact   kbdirty
07:17:34  IST     77612    938720     92.36     28620    297360   4276532    207
.31    484588    330104        88
07:17:36  IST     77612    938720     92.36     28628    297352   4276532    207
.31    484644    330104        88
07:17:38  IST     77612    938720     92.36     28628    297360   4276532    207
.31    484644    330104        92
07:17:40  IST     77628    938704     92.36     28628    297360   4276532    207
.31    484660    330104        92
07:17:42  IST     77628    938704     92.36     28628    297360   4276532    207
.31    484660    330104         0
07:17:44  IST     77628    938704     92.36     28628    297360   4276532    207
.31    484660    330104         0
07:17:46  IST     77628    938704     92.36     28628    297360   4276532    207
.31    484660    330104         0
Average:        77621    938711     92.36     28627    297359   4276532    207.3
1    484645    330104        51
MemTotal:        1016332 kB
MemFree:           77640 kB
MemAvailable:     404368 kB
Buffers:           28628 kB
Cached:           297360 kB
SwapCached:        25592 kB
Active:           484372 kB
Inactive:         330104 kB
Active(anon):     288036 kB
```

Fig. 2. Memory Usage on Ubuntu



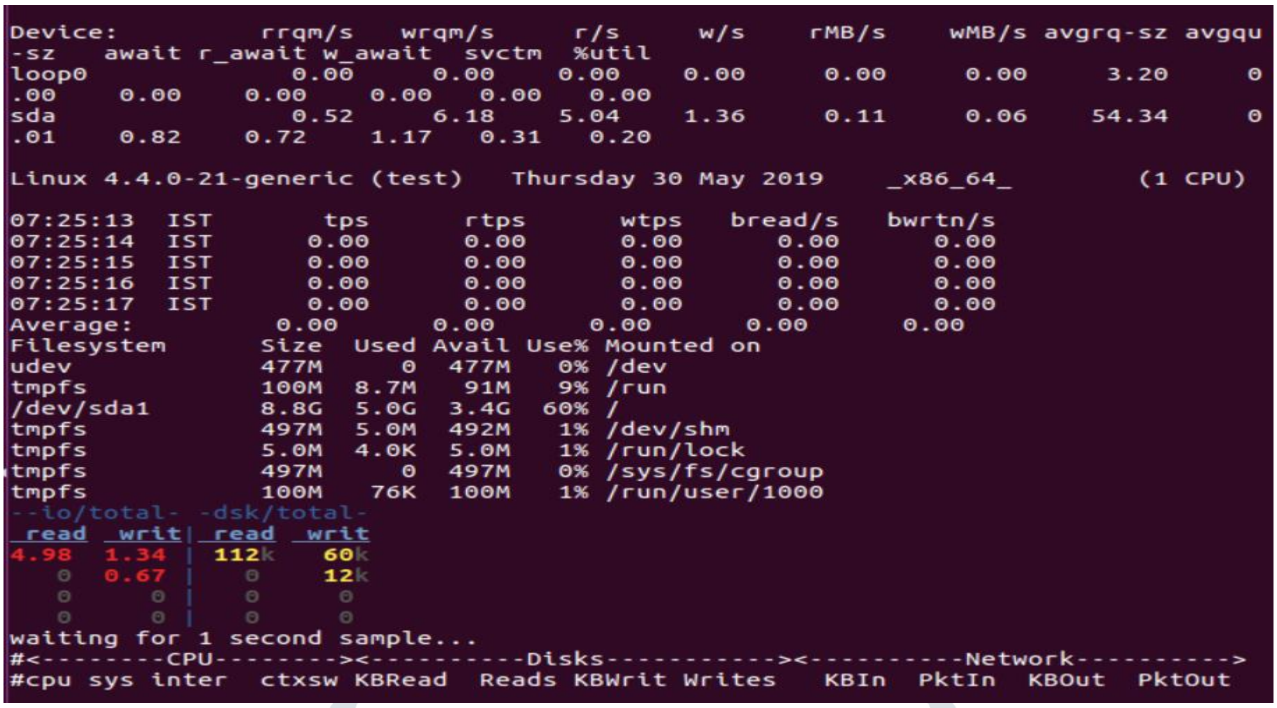
```
      enp0s3              Total
 KB/s in  KB/s out   KB/s in  KB/s out
    0.00      0.00      0.00      0.00
    0.00      0.00      0.00      0.00
    0.00      0.00      0.00      0.00
Linux 4.4.0-21-generic (test)   Thursday 30 May 2019     _x86_64_        (1 CPU)

07:27:59  IST     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcm
p/s  rxmcst/s   %ifutil
07:28:01  IST    enp0s3      0.00      0.00      0.00      0.00      0.00      0
.00      0.00      0.00
07:28:01  IST        lo      0.00      0.00      0.00      0.00      0.00      0
.00      0.00      0.00

Average:        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/
s  rxmcst/s   %ifutil
Average:       enp0s3      0.00      0.00      0.00      0.00      0.00      0.0
0      0.00      0.00
Average:           lo      0.00      0.00      0.00      0.00      0.00      0.0
0      0.00      0.00
waiting for 1 second sample...
#<--------CPU--------><----------Disks-----------><----------Network---------->
#cpu sys inter  ctxsw KBRead  Reads KBWrit Writes   KBIn  PktIn  KBOut  PktOut
   5   3   100    618    587     37      0      0      0      0      0       0
```

Fig. 3. Network Utilization

5.4 Command Line Tools for Extracting Disk Usage

Disk utilization and analysis of the same is required to handle storage intensive applications. Fig. 4. shows the various disk metrics obtained when the tools are used for obtaining disk utilization. Various tools can be used to extract disk related information:

- Input/Output Statistics (Iostat): Iostat program is used to monitor the input / output device loading of the system by examining the time the devices are engaged comparative to the average transfer rates. The iostat creates reports that can be used to modify system configuration to sustain physical disk input/output. Iostat is available in Sysstat package. The iostat command generates two reports in general, a report on the usage of the CPU and a report on all I /O statistics of the disks.
- Collectl: • Collectl is a command-line utility that can be used to collect performance data on the current status of the system. Unlike most other monitoring systems, Collectl does not focus on a limited number of system metrics. Rather, it can gather information on a variety of system resources, including CPU, disk, network, memory, sockets, inodes, luster, memory, NFS, processes, quadrics, slabs, etc.
- Dstat: Dstat can also be used to retrieve disk and process information along with CPU utilization.
- Systems Activity Report: Sar, like other tools, provides extensive list of system metrics. Disk utilization is depicted by using this tool.



```
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu
-sz   await r_await w_await  svctm  %util
loop0             0.00     0.00    0.00    0.00     0.00     0.00     3.20     0
.00    0.00    0.00    0.00   0.00   0.00
sda               0.52     6.18    5.04    1.36     0.11     0.06    54.34     0
.01    0.82    0.72    1.17   0.31   0.20

Linux 4.4.0-21-generic (test)   Thursday 30 May 2019    _x86_64_        (1 CPU)

07:25:13  IST        tps       rtps       wtps   bread/s   bwrtn/s
07:25:14  IST       0.00       0.00       0.00      0.00      0.00
07:25:15  IST       0.00       0.00       0.00      0.00      0.00
07:25:16  IST       0.00       0.00       0.00      0.00      0.00
07:25:17  IST       0.00       0.00       0.00      0.00      0.00
Average:          0.00       0.00       0.00      0.00      0.00
Filesystem      Size  Used Avail Use% Mounted on
udev            477M     0  477M   0% /dev
tmpfs           100M  8.7M   91M   9% /run
/dev/sda1       8.8G  5.0G  3.4G  60% /
tmpfs           497M  5.0M  492M   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           497M     0  497M   0% /sys/fs/cgroup
tmpfs           100M   76K  100M   1% /run/user/1000
--io/total- -dsk/total-
 read  writ| read  writ
4.98  1.34 | 112k   60k
   0  0.67 |    0   12k
   0     0 |    0     0
   0     0 |    0     0
waiting for 1 second sample...
#<--------CPU--------><----------Disks-----------><----------Network---------->
#cpu sys inter  ctxsw KBRead  Reads KBWrit Writes   KBIn  PktIn  KBOut  PktOut
```

Fig. 4. Extraction of Disk Metrics

5.5 Tools for Extracting Hardware Information

Various hardware features including the underlying Linux flavor, architecture and the number of virtual cores are extracted. Some of the tools used to extract the hardware information are as follows:

- Hostnamectl: Hostnamectl provides the right API for handling the hostname of the Linux system and changing its associated settings. The instruction also enables to change the hostname without the /etc/hostname file being located and edited on a given system. Hostnamectl provides information about the operating system as well.
- LsCpu: Lscpu assembles detailed information from sysfs and /proc/cpuinfo. The output of the tool can be tailored for human readability or parsing. For example, the information includes the number of nodes for threads, cores, CPUs, sockets, etc. CPU caches and cache sharing information, family, model, byte order, and stepping are also available.
- Hardware Lister (lshw): Lshw is a neat tool that produces detailed accounts on various hardware components such as firmware version, memory configuration, CPU version and speed, mainboard configuration, cache settings, USB, graphics card, network card, bus speed, etc. By reading files under /proc directory and DMI table, it yields hardware information. In order to detect the maximum amount of information, lshw must be run as a super user or only partial information will be reported. There is a special option in the lshw known as class, which will display specific hardware information in detail.
- List connected PCI devices (lspci): Lspci is a tool for depicting PCI bus details in the system and devices connected to it. A list of potential devices is reflected by default. In normal output, B / D / F (Bus, Device, Function) of the device is specified by the first three hexadecimal numbers. Connectivity to some sections of the PCI configuration space is restricted to root on many operating systems, so features are limited for normal users.

There are various other tools which have been developed for the purpose of studying the performance of a system. All of the tools more or less end up giving similar parameters.

VI. Analysis of Extracted Metrics

Once the metrics have been extracted through the usage of various tools, an interpretation of each metric would give an overall understanding of how the system is performing. Thus, the various metrics along with their analysis is provided in detail.

6.1 CPU Subsystem Metrics

The different types of CPU load percentages are used in depicting and analyzing the CPU utilization are as follows:

- User usage percentage (%user): This parameter indicates the percentage of CPU utilization that occurred while executing an application at the user level. A user space program is any process that doesn't belong to the kernel. Some user space processes are Shells, web servers, compilers, databases, the programs associated with the desktop, etc. It is typical that the majority of the CPU time should be spent running user space processes if the processor is not idle.
- System usage percentage (%system): Percentage of CPU utilized while executing at the kernel level is indicated by this field. All the system resources and processes are dealt with using the Linux kernel. When a user space process needs any resource from the system, for instance, when it needs to perform some I/O, allocate memory or needs to create a child process, then the kernel is running. In fact, the scheduler which determines the process that need to runs next is part of the kernel. The amount of time spent in the kernel should be very minimal.
- Priority percentage (%nice): Percentage of CPU utilization that occurred while carrying out processes at the user level with nice priority is given by this value. The priority level of a user space process can be changed by modifying its niceness. The %nice demonstrates how much time the CPU spent running user space processes that have been given a nice value thereby assigning priority. In systems where no processes have been given a priority, the nice value or number will be 0.

- Idle percentage (%idle): This depicts the percentage of time that the CPU or CPUs were idle during the time at which the system did not have an outstanding disk I/O request.
- Input/Output wait percentage (%iowait): This yields the percentage of time that the CPU or CPUs were idle during the time at which the system had an outstanding disk I/O request. Input or output operations, like writing or reading to a disk, are comparatively slower than the speed of a CPU. Although these operations happen appear to be extremely fast, they are still slow when it comes to the performance of a CPU. Sometimes, the processor has initiated a read or write operation but it has to wait for the result, there is nothing else to do. It is idle while waiting for that I/O operation to finish. This is indicated by %iowait.
- Load Average: Apart from percentages, load average also depicts the CPU utilization of a system. All UNIX-like frameworks customarily show the CPU load as 1-minute, 5-moment and 15-minute load averages. Basically, the load average depicts the fraction of time that the CPU is occupied. A CPU can be over-used, processes might be waiting for the CPU to end up accessible, so you could see usage rates over 1.00. The immaculate utilization of 1.00 per CPU implies that CPU is executing 100% of the time and no processes are waiting for that CPU to end up accessible. Obviously, a use of 1.00 per CPU would imply that there is no extra capacity to take an expanded load, so most administrators are stressed when they see utilization numbers persistently over 0.70.

The load average is also assessed in this duration. Higher %idle means that the system is not compute intensive whereas high %user depicts a large utilization of CPU. Apart from this, load average is assessed at three durations. High load average simply means that the system is overloading and that there are several processes waiting for the CPU time. Low load average means that there is no load on the CPU and hence it is not being utilized. A load average of 1.0 depicts that the system is at its capacity and cannot handle more load. But a high load average means several processes are waiting for the CPU and the CPU is not able to perform optimally.

6.2 Memory Subsystem Metrics

The memory usage is assessed through various memory parameters which is obtained during the extraction. To depict if the memory usage is high or not, a relative ratio of the used and total memory is required. The various extracted metrics include the following fields based on which computation is done. The essential parameters are as follows:

• Total: The total, physical memory of the system is given. This essentially indicates the size of the RAM used by the computer.
• Used: How of the total memory is being used by any process or application is given.
• Free: Free memory depicts the amount of memory that is completely unused.
• Buffers: Kernel buffers are required to execute kernel level tasks, therefore, memory used by the buffer present at the kernel is indicated.
• Cached: The amount of memory that is cached for fast and frequent access is presented by cached memory. This includes active and inactive cache memory which is extremely important in assessing the performance of cache in a system.

6.3 Network Subsystem Metrics

Building and running an IP network needs a thorough knowledge of the infrastructure as well as the efficiency of systems used within the network, including how each network device handles packets. Network engineers most often use the speed of interfaces expressed in bits per second (b/s) to refer to network device performance. While this data is helpful and significant, expressing performance in terms of b/s alone does not properly cover other significant performance metrics of network devices. Network devices obtain and forward packets using Layer 2 technologies such as Ethernet through physical interfaces. For these network connections, the description always involves bandwidth expressed as b/s. By performing simple mathematical transformations, the potential range of frames per second to be supported by the network can be determined.

- Maximum Frame Rate (Minimum Frame Size): The maximum frame rate of Ethernet is achieved by a single transmitting node with no collisions when Ethernet frames are at their smallest size.
- Maximum throughput (Maximum Frame Size): A single transmitting node achieves maximum Ethernet throughput without collisions when the Ethernet frames are at their maximum size.
- Transactions per second (t/s): This refers to the number of complete actions per second of a particular type. The measurement of t/s relates to more than just processing a single packet or even setting up a fresh connection; it relates to completing a full cycle of a particular action. Some devices use this metric in networking to define the implementation of some complicated process to packets to include a complete conversation.
- Received packets (rxpck/s): Total number of packets received per second.
- Transmitted packets (txpck/s): Total number of packets transmitted per second.
- Received (rxkB/s): Total number of kilobytes received per second.
- Transimitted (txkB/s): Total number of kilobytes transmitted per second.
- Received error packets (rxerr/s): Total number of bad packets received per second.
- Received packets dropped per second (rxdrop/s): Number of received packets dropped per second because of a lack of space in Linux buffers.
- Transmitted packets dropped per second (txdrop/s): Number of transmitted packets dropped per second because of a lack of space in Linux buffers
- Packet Out: Shows the numbered being transmitted out of the interface of the system.
- Packets In: Shows the number of packets being received from the interface of the system.
- Total: Depicts the total amount of data being transmitted to and from the interface. This is given in Mebibyte (MiB).
- Received Data (RX): Similar to packets in, it depicts the amount of data arriving into the interface in MiB.
- Transmitted Data (TX): Similar to packets out, it depicts the amount of data being pushed away from the interface to other systems in MiB.

6.4 I/O Subsystem Metrics

Disk utilization is analyzed based on the IOPS as well as the amount of space used by the device on the file system. A high percentage of disk usage of the different partitions on the device means that the system requires high storage capabilities. As for the IOPS, higher values indicate that the system is designed to handle more operations per second which means that the storage is definitely essential for the functioning of the system. Some of the parameters extracted that is assessed are as follows:

- Transfers per second (tps): Indicate the number of transfers per second that were issued to the device. Multiple logical requests can be combined into a single I/O request to the device. A transfer is of indeterminate size.
- Read Sector per second (rd_sec/s): Number of sectors read from the device. The size of a sector is 512 bytes.
- Write Sector per second (wr_sec/s): Number of sectors written to the device. The size of a sector is 512 bytes.
- Average request size (avgrq-sz): The average size (in sectors) of the requests that were issued to the device.
- Average queue size (avgqu-sz): The average queue length of the requests that were issued to the device.
- Await time: The average time (in milliseconds) 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.
- Percentage utilization (%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%.

6.5 Hardware Information

Hardware information on the other hand gives the kind of architecture and the underlying operating system. This is very helpful in creating a standard platform as packages for the tools are installed based on the operating system. The hardware information also helps depict whether the environment is virtual or real, thereby, providing a broader aspect on how resources are used. The hardware information also provides the clock frequency. Higher the clock frequency, more compute intensive the system becomes but the problem with analyzing performance with clock speed is that performance depends on the internal architecture of the processor as well. Thus, comparing systems with the same clock speed but different processors becomes pointless. The most important parameter provided by the hardware is the number of cores. The following information is analyzed:

- Clock Speed: The speed at which the processor can execute instruction sets is called clock speed. More instructions are executed by the processor when the clock speed is faster. The clock speed is expressed in megahertz or gigahertz. Processors with quicker clock speeds process information quicker than those with slower clock speeds. The clock speed is frequently coordinated into their model numbers. However, the efficiency of the processor's design decides how much real work a processor can do with the same number of clock cycles. It's one of the significant components that decides how well a CPU will perform in real-world circumstances.
- Number of cores: A component of the CPU that gets instructions and performs actions or calculations, in view of those instructions. These instructions can permit a program or software to perform a particular function. Processors can have multiple cores or a single core. A processor with two cores is known as a dual core processor, four cores is quad-core, and so on up to eight cores. As the number of cores increase, more instruction sets can be processed simultaneously, thereby increasing the speed of the system. There is improvement in the performance of the system due to the introduction of multiple cores within the CPU. The physical CPU unit is extremely small and fits into a socket. There is only a single CPU socket, multiple sockets each with their own cooling, power, etc. are not required. This provides a reduced latency as the communication between the cores is made quicker by being on the same chip.
- Cache memory: The purpose of a cache is to ensure this fast and smooth transition transfer of data from the hardware to the central processing unit. The importance of cache can be understood by analyzing the working of the entire process. The hard drive provides most of the information. On the request of an application, information will be fetched from the drive and delivered for processing to the CPU. Data transfer consumes a lot of time each time it is retrieved from the hard disk to the CPU. To overcome this, the RAM stores the temporary data from the hard disk. The processor now checks the RAM first for the information, only if it is not found in the RAM will it get the information or data from the hard drive. But with the increase of CPU speed, RAM could no longer keep up, thus, causing serious problems. To resolve these issues, cache was introduced. It is an extremely fast and small memory which is added to the processor for the storage of immediate instruction from the RAM. The cache can give information to the CPU without any lag as the speed of the cache matches that of the CPU. There are varying sizes of cache. The larger the cache size, the faster the data transfer resulting in efficient CPU performance.

VII. CONCLUSION

Due to the versatility and variety of features provided by Linux operating systems, it is adequate to utilize the Linux platform for studying the impacts of performance on these systems. It is found that the most crucial aspects of performance are related to the CPU, memory, network, disk and hardware specifications of the system. An ideal system would manage resources such that none of these parameters fall short on any assets and bottlenecks are avoided. In case bottlenecks arise, extensive analysis of the performance metrics would help mitigate the effects of such complications.

It can be concluded that the CLI tools used for extraction of system performance metrics are easy to use and provide a vast amount of information. These tools drastically reduce the man power required to extract and process system information which is normally obtained from the kernel. Thus, usage of command line tools such as top, dstat, iostat, etc extracts the system metrics which can be analysed based on the values seen corresponding to each metric.

There are several other tools which provide details on system performance. These tools need to be understood and compared with the preexisting tools so that the most accurate tool is utilized. The process of how each tool extracts the metrics also needs to be assessed to provide deeper understanding of the system.

REFERENCES

[1] Ahmet Aksoy and Mehmet Hadi Gunes, "Operating System Classification Performance of TCP/IP Protocol Headers," in Proc. IEEE International Conference on Local Computer Networks Workshops, Dubai, United Arab Emirates, November 2016, pp. 112-120.
[2] Xinchao Han and Xianfeng Du, "A New Method about Operating System Identification," in Proc. IEEE International Conference on Information and Financial Engineering, Chongqing, China, October 2010, pp. 882-885.
[3] Nandini U, Nivetha B and Shobhana D, "An Analysis of Linux Operating System," *International Journal of Trend in Research and Development*, vol. 3,no. 1, pp. 32-35, January 2016.
[4] Guanping Xiao, Zheng Zheng and Haoqin Wang, "Evolution of Linux Operating System Network," *Physica A: Statistical Mechanics and its Applications*, Elsevier, vol. 466(C), pages 249-258.
[5] Luca Abeni, Ashvin Goel, Charles Krasic, Jim Snow and Jonathan Walpole, "A Measurement-Based Analysis of the Real-Time Performance of Linux," in Proc. Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, San Jose, CA, USA, January 2003, pp. 1-10.
[6] Maria Dimakopoulou, Stephane Eranian, Nectarios Koziris and Nicholas Bambos, "Reliable and Efficient Performance Monitoring in Linux," in Proc. Of IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, March 2017, pp. 396-408.
[7] BI Chun-yue, "Analysis and Research on Improving Real-time Performance of Linux Kernel," *International Journal of All Research Education and Scientific Methods (IJARESM)*, vol. 5, no. 5, pp. 75-80.
[8] Li Yuanyuan, Xiao Peng and Deng Wu, "The Method To Test Linux Software Performance," in Proc. IEEE International Conference on Computer and Communication Technologies in Agriculture Engineering, Chengdu, China, August 2010, pp. 420-423.
[9] Roberto Giorgi, Marco Procaccini and Farnam Khalili, "Analyzing the Impact of Operating System Activity of different Linux Distributions in a Distributed Environment," in Proc. 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Florence, Italy, March 2019, pp. 422-429.
[10] Tommaso Cucinotta, Dhaval Giani, Dario Faggioli and Fabio Checconi, "Effective Real-Time Computing on Linux," Proceedings of the 12th Real-Time Linux Workshop (RTLWS), Italy, January 2010, pp. 1-11.
[11] Carlos Garre, Domenico Mundo, Marco Gubitosa and Alessandro Toso, "Real-Time and Real-Fast Performance of General-Purpose and Real-Time Operating Systems in Multithreaded Physical Simulation of Complex Mechanical Systems", *Mathematical Problems in Engineering*, vol. 2014, pp. 1-14, May 2014.
[12] Zexin Jiang, "A Linux Server Operating System's Performance Comparison using lmbench," in Proc. IEEE International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, April 2016, pp. 160-164.
[13] Andrew Solomon, Daniel Santamaria and Raymond Lister, "Automated Testing of Unix Command-Line and Scripting Skills," International Conference on Information Technology Based Higher Education and Training, Ultimo, NSW, Australia, July 2006, pp. 1-6.
[14] Mokhtar Ebrahim, Andrew Mallett, "Mastering Linux Shell Scripting", Packt Publishing, 2nd edition, 2018.
[15] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", John Wiley & Sons, 9th edition, 2013.
[16] Christopher Negus, "Linux Bible", John Wiley & Sons, 8th edition, 2012.