

MONITORING SERVER ON CPU UTILIZATION, RAM UTILIZATION, SWAP UTILIZATION

ABSTRACT

Monitoring server is used CPU utilization, RAM utilization, SWAP utilization. In this concept the CPU Utilization is top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of tasks currently being managed by the Linux kernel. The top command monitors CPU utilization, process statistics, and memory utilization. The top sections contains information related to overall system status – uptime, load average, process counts, CPU status, and utilization statistics for both memory and swap space. The “free” command usually display the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel. The “top” command provides a dynamic real-time view of a running system. Swap space is nothing but computer memory management involving swapping regions of memory to and from storage. We use python script to execute the output in Linux.

INTRODUCTION

This article takes us through various methods which discuss CPU utilization, RAM utilization, SWAP utilization. Monitoring CPU performance is critical to debugging system processes, making system decisions, handling system resources, and inspecting and evaluating system real-time. With the help of the “top” command, we can monitor the system in real-time. When we execute the top command, it will provide us the summary of the system along with the list of threads and process which are presently being managed by the Linux kernel. The RAM usage on a system is good to know for a few reasons. Firstly, it can give you some insight into whether or not it’s necessary to upgrade the amount of memory inside your server or computer. If you see the memory utilization regularly nearing full capacity, it could indicate that your system needs an upgrade. In the RAM utilization we use “free” Linux command provides a very quick and easy way to see a system’s current memory utilization. In the SWAP utilization to see swap size in Linux. Type the command is “swapon -s”. You can also refer to the /proc/swaps file to see swap areas in use on Linux. Type “free -m” to see both your ram and your swap space usage in Linux. Finally, one can use the top or htop command to look for swap space utilization on Linux too.

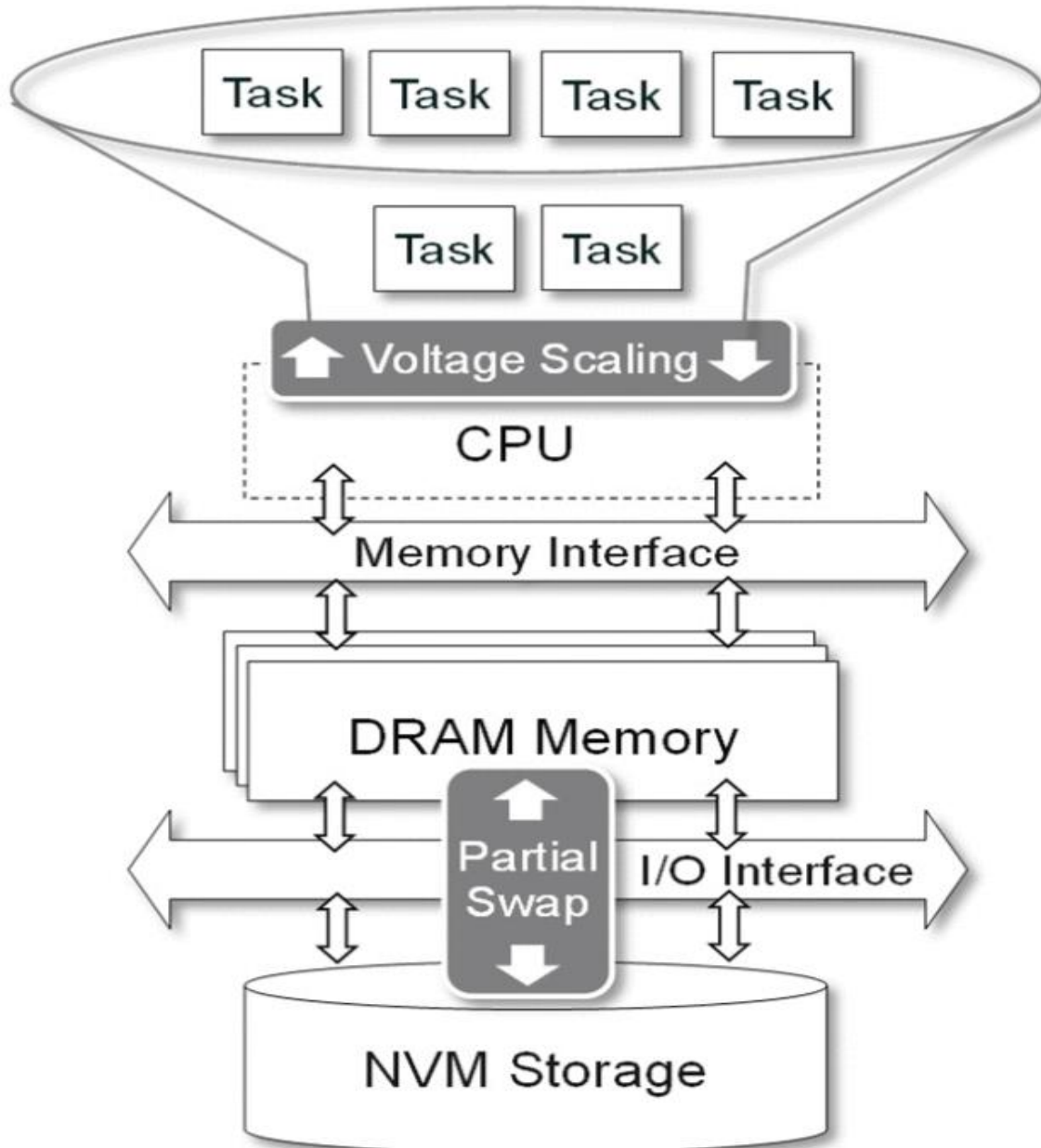
MODULES:

Os-The OS module in python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. you first need to import the os module to interact with the underlying operating system.

Psutil – psutil (process and system utilities) is a cross-platform library for retrieving information on running processes and system utilization

PROPOSED SCHEME

In our task model, a real-time task set is defined as $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$, and the target system has CPU with a voltage scaling function and main memory supporting swap as shown in Figure . A task τ_i is characterized by $\langle C_i, T_i, M_i \rangle$, where C_i is the worst case execution time of τ_i with the default CPU voltage and no memory swap, T_i is the period of τ_i , and M_i is the memory footprint of τ_i . We consider periodic real-time tasks, and thus the deadlines are implicitly determined by the period.



WORKING PROCEDURE

CPU UTILIZATION- top command is used to show the Linux process. It provides a dynamic real-time view of the running system.

```
viktor@viktor-VirtualBox: ~  
top - 18:00:32 up 1 min, 1 user, load average: 1.61, 0.91, 0.35  
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 1.5 us, 1.5 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 5945.8 total, 4258.3 free, 611.1 used, 1076.4 buff/cache  
MiB Swap: 1162.4 total, 1162.4 free, 0.0 used. 5081.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	167736	11588	8308	S	0.0	0.2	0:01.20	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	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	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0-events
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
7	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/0:1-events
8	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/u8:0-events_unbound
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:00.15	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

RAM UTILIZATION & SWAP UTILIZATION - free command gives information about used and unused memory usage and swap memory of a system. By default, it displays memory in kb (kilobits).

Memory mainly is a part of hard disk drive that acts like a virtual RAM.

```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ free  
              total        used        free      shared    buffers     cached  
Mem:           1928144      1687068      241076           0       33252      705952  
-/+ buffers/cache:      947864      980280  
Swap:          1986556         3392      1983164  
sssit@JavaTpoint:~$
```

PROGRAM

```
>>import os
```

```
>>import psutil
```

```
>>threshold = 90
```

```
# CPU UTILIZATION-----
```

```
>>load1, load5, load15 = psutil.getloadavg()
```

```
>>cpu_usage = (load15/os.cpu_count()) * 100
```

```
>>if cpu_usage >= threshold:
```

```
...     print("The CPU usage is:", cpu_usage)
```

```
# RAM UTILIZATION-----
```

```
>>used_memory, total_memory, free_memory = map(int, os.popen('free -t -m').readlines() [-1].split() [1:])
```

```
>>if total_memory >= threshold:
```

```
...     print("RAM memory % used:", round((used_memory/total_memory) * 100, 2))
```

```
# SWAP UTILIZATION-----
```

```
>>used_memory, total_memory, free_memory = map(int, os.popen('free -t -m').readlines() [-2].split() [1:])
```

```
>>if total_memory >= threshold:
```


```
...     print("SWAP Usage is:", round((total_memory/free_memory) * 100, 2))
```

Explanation:

- The `psutil.getloadavg()` provides the load information of the CPU in the form of a tuple. The `psutil.getloadavg()` runs in the background and results gets updated every 5 seconds. The `os.cpu_count()` returns the number of CPU in the system.
- The `os` module is also useful for calculating the ram usage in the CPU. The `os.popen()` method with flags as input can provide the total, available and used memory.
- The `os` module is also useful for calculating the swap usage in the CPU. The `os.popen()` method with flags as input can provide the total, free and used memory.
- By the above program we will get output of above 90% threshold value for these three CPU utilization, RAM utilization and SWAP utilization.

Output 1:

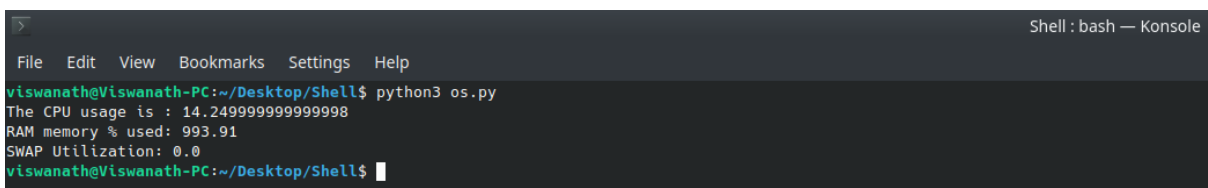
By using this code we will get the output which CPU utilization, ram utilization, swap utilization has used 90% above threshold value.



```
viswanath@Viswanath-PC:~/Desktop/Shell$ python3 os.py
RAM memory % used: 600.67
viswanath@Viswanath-PC:~/Desktop/Shell$
```

Output 2:

By using this code we will get the output which CPU utilization, RAM utilization, SWAP utilization has used 0% above threshold value.



```
viswanath@Viswanath-PC:~/Desktop/Shell$ python3 os.py
The CPU usage is : 14.249999999999998
RAM memory % used: 993.91
SWAP Utilization: 0.0
viswanath@Viswanath-PC:~/Desktop/Shell$
```

Conclusion:

As the size of data grows rapidly in modern embedded systems, the DRAM memory of the system keeps increasing, which accounts for a large portion of the power consumption. In this article, we presented a new real-time task scheduling scheme that supports partial swap in order to reduce the DRAM size of the system. To enable swap functions, we adopted high-speed NVM storage with predictable access latency, which allows for the feasible estimation of real-time task's worst case execution time. Unlike typical real-time systems that maintain entire footprint of tasks in memory, we place a certain part of real-time tasks in NVM storage and perform swapping. The ratio of swap for each task is determined based on the schedulability and the power-saving effect.