RV COLLEGE OF ENGINEERING ®
Bengaluru - 59

(Autonomous Institution affiliated to VTU, Belagavi)

Internship Project Report on

CLUSTER PERFORMANCE WATCHER

Submitted By:

ANUBHAV DINKAR 1RV17EC019

ELECTRONICS & COMMUNICATION ENGINEERING

May 2019 - July 2019

ACKNOWLEDGEMENTS

I would like to thank Mr. Prasad Kollipara, Staff Engineer at Cray Supercomputers India, for guiding me along the entire project and teaching me several concepts which has added to my skillset.

I would also like to thank Mr. Prasad Balakrishnan, Head - R&D at Cray Supercomputers India, and Ms. Kamalashree, HR at Cray, for giving me the opportunity to work, and learn, as an intern at Cray.

I would finally like to extend my thanks to everyone else at Cray for supporting me in completing this project.

Table of Contents

ACKNOWLEDGEMENTS **2**

ABSTRACT **4**

INTRODUCTION **5**

NODES AND CLUSTERS 5
VIRTUAL MACHINES 5

METHOD OF OPERATION **7**

RESULT **9**

CONCLUSION **11**

REFERENCES AND SOFTWARE USAGES **12**

ABSTRACT

The project I have implemented aims to provide a visualization of the CPU and Memory usages of nodes in a cluster. The cluster is simulated as a connection between a host and Virtual Machines, via Paramiko, an SSH client. Through this project, performance metrics of the VMs can be tracked and stored on MySQL as a time series database, and then visualized using Grafana.

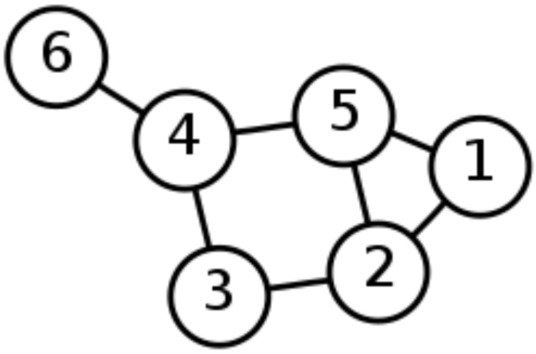
Using this visualization, users can track the metrics for different processes and users on a node, and take necessary actions if needed. Furthermore, tracking these metrics can help in improving the performances of nodes in a cluster, while using only one server, or node, to store this data, rather than having to individually track the performances of each and every node, a process that can be very inefficient.

Thus, this project aims to be relatively more efficient in giving a detailed report of the individual performances of every node over a long period of time.

INTRODUCTION

Nodes and Clusters

In computer science, nodes are devices or data points on a large network. A mobile phone, a PC, a printer, etc., can be considered as nodes connected to a single network. A node has a programmed or engineered capability that enables it to recognize, process or forward transmissions to other nodes.



Simplified representation of a cluster of nodes, where each number represents a node

Nodes are often arranged into tree structures. These nodes may contain a value or condition, or even serve as another independent data structure. A Cluster of nodes, is simply a collection of these nodes. In the case of a Local Area Network (LAN) or a Wide Area Network (WAN), each device has its own network address. In the case of Internet or an Intranet, many physical network nodes are host computers, known as Internet nodes, identified by an IP Address.

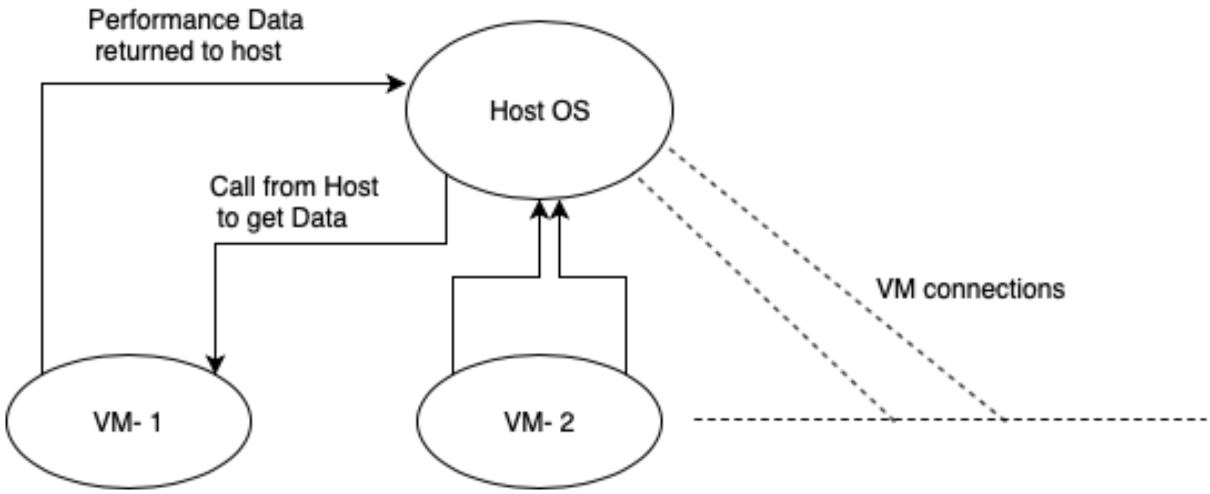
Virtual Machines

A virtual machine (VM) is a computer file, typically called an image, which behaves like an actual computer. By creating a virtual machine, in essence, a computer is created within another computer. It runs inside a window, much like any other application on the host computer, and gives the user the same experience using this virtual machine as that while using a host computer with that Operating System. The virtual machine is *sandboxed* from the rest of the system, meaning the software inside the virtual machine cannot escape, or tamper with, the host computer itself. This provides an ideal work environment for testing other operating systems including beta releases, accessing virus-infected data, creating operating system backups, and running software or applications on operating systems for which they were not initially intended.

Multiple VMs can run simultaneously on the same physical computer. For servers, the multiple operating systems run side-by-side with a software known as a hypervisor to manage them, while desktop computers typically employ one operating system to run the other operating systems within its program windows. Each VM provides its own virtual hardware, including CPUs, memory, hard drives, network interfaces and other devices.

The virtual hardware is then mapped to the real hardware on the physical machine, which saves costs by reducing the need for physical hardware systems along with the associated maintenance costs that go with it, and also reduces power and cooling demand.

For the purpose of this project, a Virtual Machine, running *Ubuntu 18.04*, was created and the metrics for this VM was monitored (or *watched*) using a Python script on the Host OS (*MacOS*).



Representation of the cluster of Virtual Machines (VM1, VM2…) that was created on a physical machine, on the Host OS, and the basic operation that is being performed

METHOD OF OPERATION

Using a Python script, the performance metrics (namely the CPU usage and the Memory usage of each user and each process) of the guest Operating systems (the Ubuntu VMs), were monitored on the Host Operating System (MacOS), or the physical machine. The Virtual Machines were installed using **Oracle VirtualBox.** This connection between the host and guest systems was done using **SSH** (Secure Shell). The SSH Client used here was **Paramiko.** The string of data that was obtained was parsed and the required data was extracted. Secondly, the **Average CPU and Memory usages, the maximum usages, and the Last 3 Values of each parameter** were calculated. Additionally, **a Flame Graph** to visualize the processes being executed was also created.

- Using a connection established via SSH, the below command:

```
ssh.exec_command("top -bn 1")
```

- This command returns all the required raw data, in a format as shown below:

```
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   10 root      20   0       0      0      0 I  6.2  0.0   0:00.71 rcu_sched
 2008 anubhav+  20   0   50632   3684   3124 R  6.2  0.2   0:00.02 top
    1 root      20   0  159948   9356   6796 S  0.0  0.5   0:01.75 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
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H-kb
    7 root      20   0       0      0      0 I  0.0  0.0   0:00.22 kworker/u2:0-ev
    8 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.29 ksoftirqd/0
   11 root      20   0       0      0      0 I  0.0  0.0   0:00.00 rcu_bh
```

The columns which are required are: "USER", "%CPU", "%MEM" and "COMMAND"

- The string of data obtained is also of the same format, and to obtain the required columns of data, Regular Expressions (RegEx) was used.

```
w___="[\d]+\s+([\w]+).\s+[\d]+\s+[\d]+\s+[\d]+\s+[\d]+\s+[\d]+\s.\s+([\d]+.[
\d]+)\s+([\d]+.[\d]+)\s+........([\w]+.*)"
```

- The data, now separated, is collected and stored in Dictionaries. Using this data, the average, past 3 values of CPU and MEM usages, and the maximum of CPU and MEM usages, per user and per command, are calculated.

- The data thus obtained, is stored in dictionaries of the format:
  - Key: User/Command, ex.: *root*
  - Value: Usage (CPU/MEM), ex.: *12.58*

- Now, the data is stored in a *MySQL Database,* using statements of the form:

```
sql = """INSERT INTO
`USERDATA`(username,cpu,mem,past_1CPU,past_2CPU,past_3CPU,
past_1MEM,past_2MEM,past_3MEM,maxCPU,maxMEM,time)
 VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
cursor.execute(sql, to_be_inserted)
dbb.commit()
```

- Above, *to_be inserted* is a tuple which contains the required data to be inserted into the table in a MySQL database.
- The intervals for which these processes run is configurable, and are declared as below:

```
long_delay = 90
short_delay = 30
interval_time = 5
```

  - *interval_time* – The number of seconds between consecutive calls of the *top* command.
  - *short_delay* – The total number of seconds the above described process needs to run.
- It is not usually required to keep the data collected this frequently, so an additional delay is added so that the range of data which can be seen is much wider. This is achieved by keeping track of the historical data, in separate dictionaries.
- This data is collected every *short_delay* seconds, and then stored in separate History tables in the sae MySQL database every *long_delay* seconds.
- Also, the last column in each of the tables is the *time* at which the data was inserted This enables the database to act as a **Time-series Database.**
- Finally, the historical tables are connected via the MySQL API available on Grafana, in order to visualize the data.
- Also using the below command a Flame Graph can also be generated:

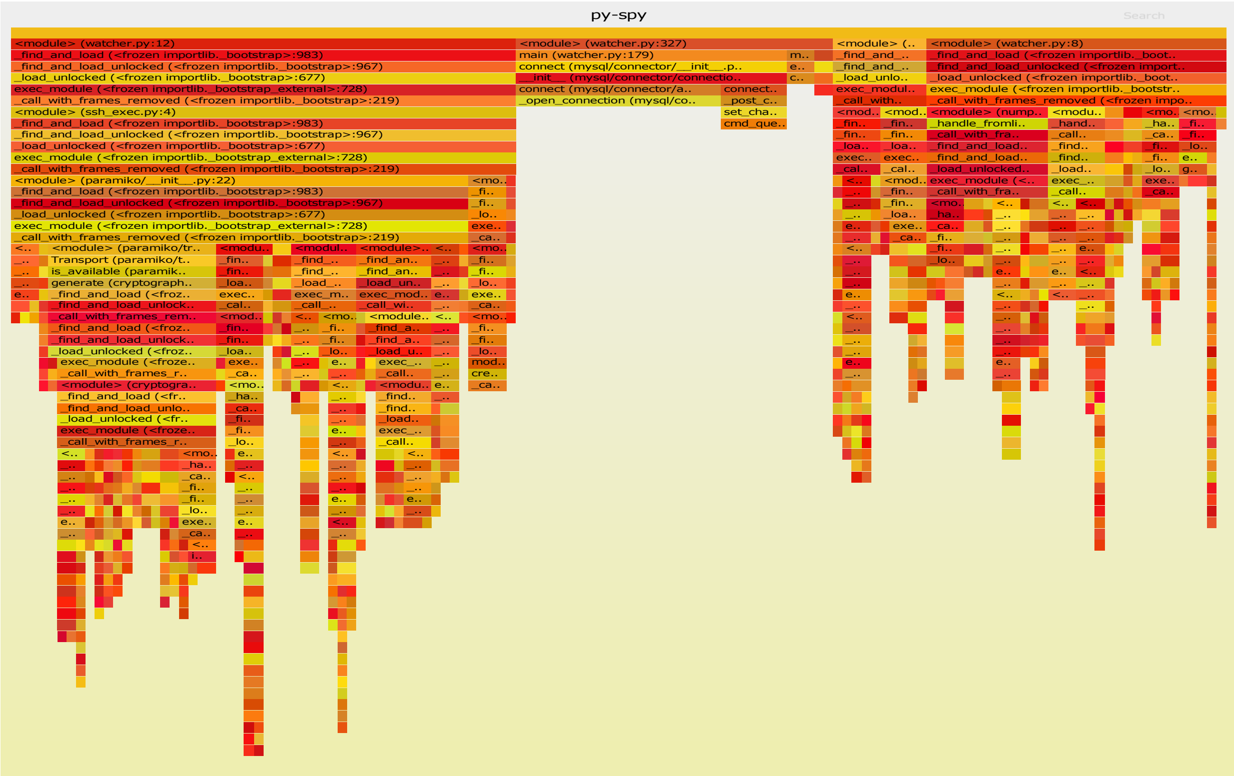
```
py-spy --flame profile.svg -- python3 watcher.py
```

The above procedure is achieved by using 5 Python scripts:

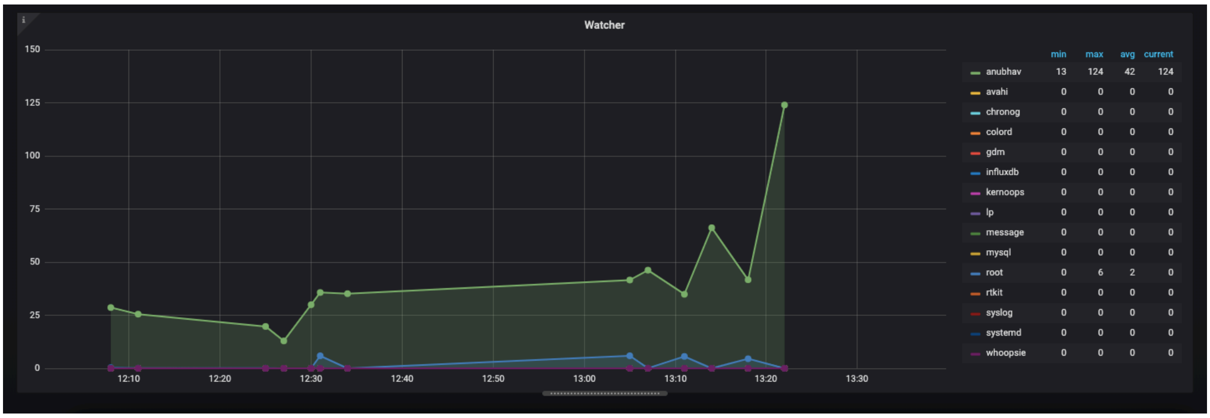
- *watcher.py* – Contains the core of the program, and makes calls to other scripts, as below, in order to achieve the required objective.
- *ssh_conf.py* – This file contains the SSH connection details:
  - *Server (or Remote Device IP)*
  - *Username*
  - *Password*
- *ssh_exec.py* - This contains the code to establish the connection between Host OS and the VM, and then call the *top* command.
- *sql_conf.py* – This contains the SQL Database details:
  - *host* – IP at which the database is present
  - *uname*- username for MySQL
  - *passwd* – password for MySQL
- *sql_exec.py* – This file contains two functions, one to create tables, and one to drop (or delete completely) the tables.

RESULT

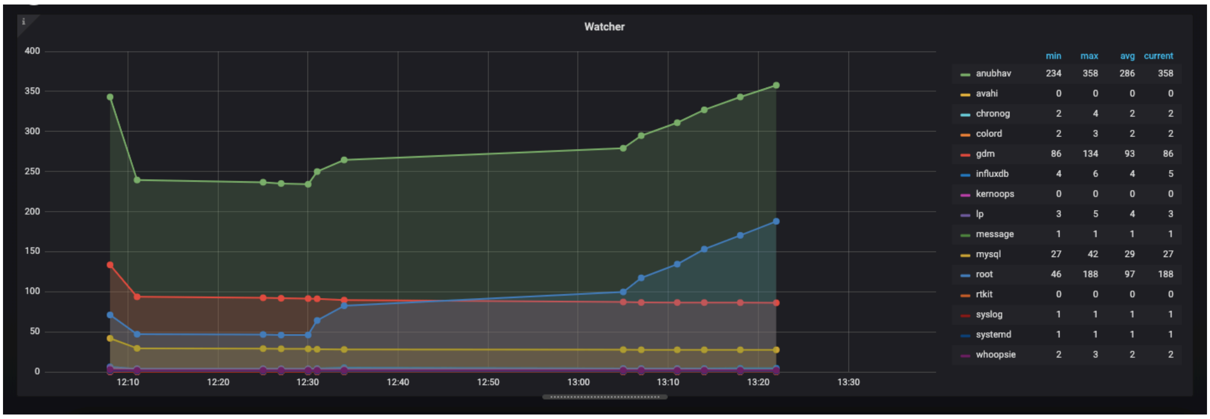
- Flame Graph:



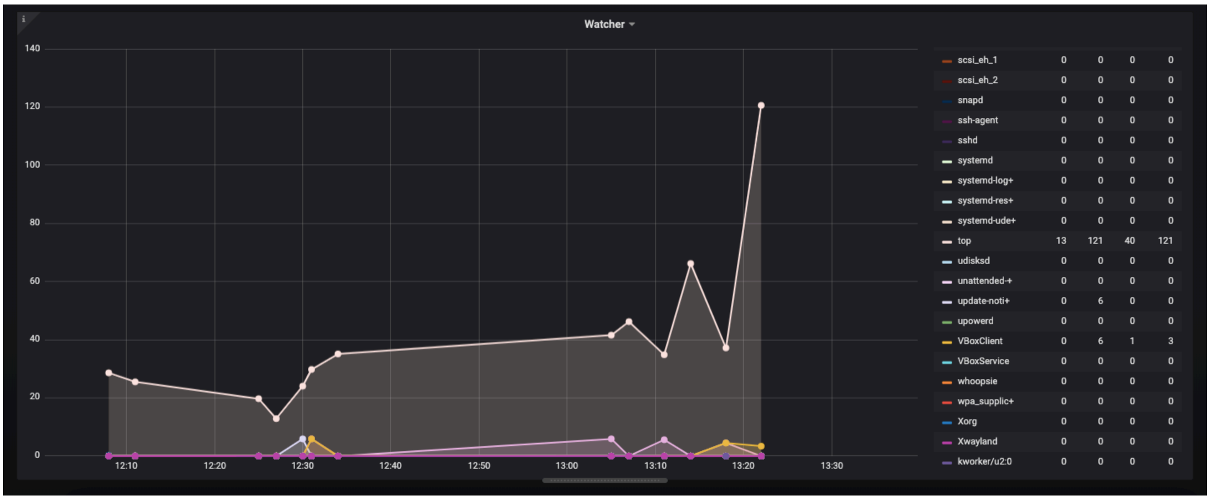
- User CPU:



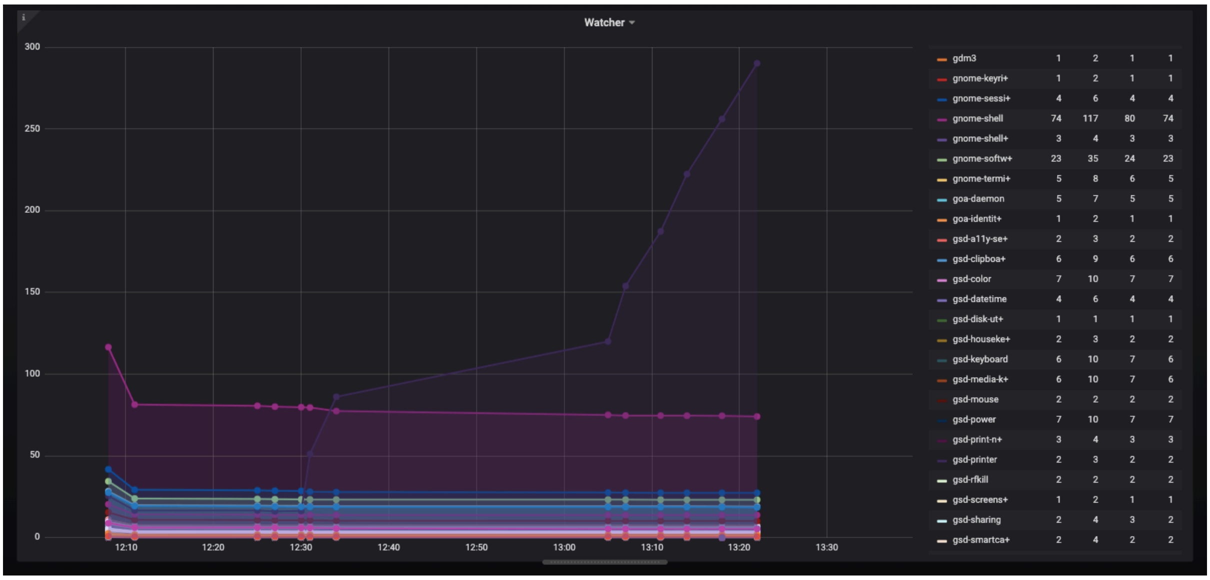
- User Memory:



- Process CPU:



- Process Memory



CONCLUSION

By implementing the project, users can keep track of which nodes (users/processes) are consuming higher amounts of Memory or using too much of the CPU. Therefore, the users can take necessary actions to reduce these parameters, so as to maintain a balance while performing several tasks over a longer period of time. Finally, this project needs only the IP addresses of each node for the SSH connection, and can then perform all these tasks on just one machine (or node), therefore making it more efficient than individually tracking the performance on each node.

REFERENCES AND SOFTWARE USAGES

- https://azure.microsoft.com/en-in/overview/what-is-a-virtual-machine/
- https://www.cbronline.com/what-is/what-is-a-node-4927877/
- http://www.brendangregg.com/flamegraphs.html
- https://glossary.atis.org/
- https://grafana.com
- https://www.w3schools.com/python/python_regex.asp
- SSH
- MySQL Server
- Oracle VirtualBox
- Paramiko - SSH Client for Python
- Py-Spy (to generate the Flame Graph)