Performance Co-Pilot

Vaibhav Nagare
Associate Software Maintenance Engineer with Kernel
Email: vnagare@redhat.com / nagarevaibhav@gmail.com

Agenda

- Why do RedHat Support Engineers use PCP?
- Solving customer’s issue using PCP through support cases.
- Contributing to PCP.

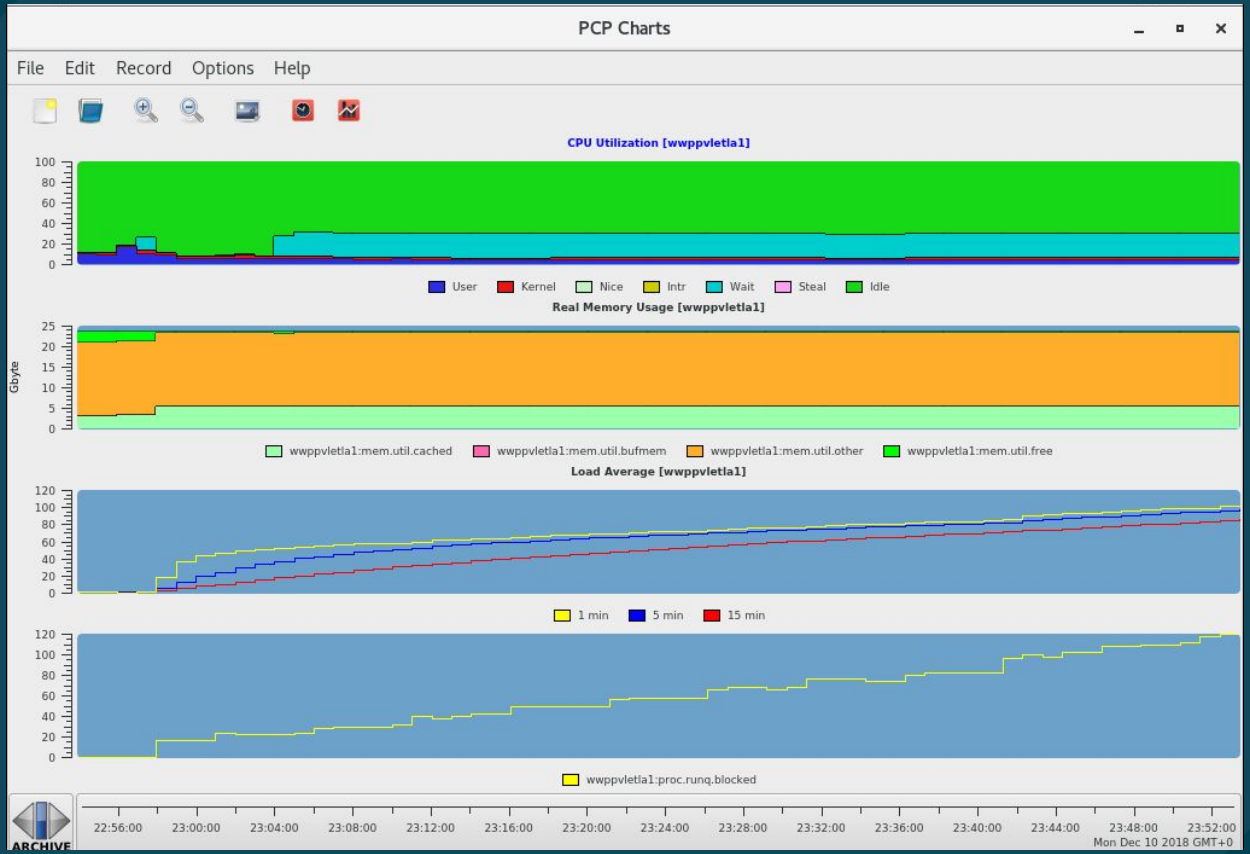
Why do RedHat Support Engineers use PCP?

- Single archive/ log can help :
  - To identify and isolate the issue.
  - Provide recommendations to fix the issue.
- No need to collect separate logs when issue occurs .
- Collection of system metrics.
- Live monitoring and analysis of system.
- Configurable.
- Graphs using pmchart.

Customer issue

- Issue -
  - Production server was in **hung** state.
  - We have **rebooted** the server post that we are able to login the server.
  - We rebooted the server around 1:30AM UKT.
- Expectation - Root cause analysis and probable fix.
- Data provided - sosreport and PCP-Logs.

Analysis: Previous day



Analysis


PCP Charts
File
Edit
Record
Options
Help
CPU Utilization [wwppvletla1]
100
80
60
40
20
0
User
Kernel
Nice
Intr
Wait
Steal
Idle
Real Memory Usage [wwppvletla1]
Gbyte
25
20
15
10
5
0
wwppvletla1:mem.util.cached
wwppvletla1:mem.util.bufmem
wwppvletla1:mem.util.other
wwppvletla1:mem.util.free
Load Average [wwppvletla1]
250
200
150
100
50
0
1 min
5 min
15 min
1,000
800
600
400
200
0
wwppvletla1:proc.runq.blocked

Analysis:

- Checking if the archive/ log has information of given timestamp:

```
--------------------------------------------------------------------------------
$ pmdumplog -z -L 20181211.00.25.0
Note: timezone set to local timezone of host "server" from archive

Log Label (Log Format Version 2)
Performance metrics from host server
    commencing Tue Dec 11 00:25:07.371939 2018
    ending     Tue Dec 11 01:27:07.979533 2018
Archive timezone: GMT
PID for pmlogger: 13389
```

Analysis:

- CPU utilization shows that most of system was in idle state :

```
$ pcp atopsar -c --archive 20181211.00.25.0 --hostzone | less
server 2.6.32-754.3.5.el6.x86_64  #1 SMP Thu Aug 9 11:56:22 EDT 2018  x86_64  2018/12/11
[..]

00:25:27  cpu  %usr %nice %sys %irq %softirq  %steal %guest  %wait %idle  _cpu_
00:25:27  all    29     0   15    0        0       0      0    190   566
             0     3     0    2    0        0       0      0      0    94
             1     3     0    2    0        0       0      0      0    94
             2     4     0    2    0        0       0      0      0    94
             3     5     0    1    0        0       0      0      0    94
             4     5     0    2    0        0       0      0     94     0
             5     3     0    2    0        0       0      0      0    94
             6     2     0    2    0        0       0      0      0    96
             7     2     0    2    0        0       0      0     96     0
[...]
01:27:07  all    30     0   17    0        0       0      0    189   565
             0     4     0    2    0        0       0      0      0    94
             1     4     0    2    0        0       0      0      0    94
             2     3     0    2    0        0       0      0      0    95
             3     3     0    3    0        0       0      0      0    94
             4     4     0    2    0        0       0      0     94     0
             5     4     0    2    0        0       0      0      0    94
             6     3     0    2    0        0       0      0      0    95
             7     4     0    2    0        0       0      0     94     0
```

Analysis:

- Checking further ‘pmstat’ shows:
  - System is idle.
  - Load average is elevated.
  - Memory usage is high.

```
# pmstat -a 20181211.00.25.0 --hostzone | tail -n 12
 loadavg                 memory       swap          io       system         cpu
   1 min   swpd   free        buff    cache  pi      po      bi  bo   in     cs  us  sy  id
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   64 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   64 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   64 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
  227.85 130728 592220  97208  4828m 0       0       0   63 9205  37K   4   2  94
```

Analysis:

- Further verification shows system's load average was indeed elevated !

```
$ pmval -f 3 kernel.all.load -a 20181211.00.25.0 --hostzone | head -n 25
Note: timezone set to local timezone of host "server" from archive

metric:    kernel.all.load
archive:   20181211.00.25.0
host:      server
start:     Tue Dec 11 00:25:07 2018
end:       Tue Dec 11 01:27:07 2018
semantics: instantaneous value
units:     none
samples:   3721
interval:  1.00 sec
00:25:07.371  No values available

                     1 minute      5 minute     15 minute
00:25:08.371          147.140       143.950       130.220
00:25:09.371          147.140       143.950       130.220
00:25:10.371          147.140       143.950       130.220
00:25:11.371          147.140       143.950       130.220
00:25:12.371          147.140       143.950       130.220
00:25:13.371          147.140       143.950       130.220
00:25:14.371          147.140       143.950       130.220
```

Analysis:

- System’s load average was elevated !

```
# pcp atopsar -p --archive 20181211.00.25.0 --hostzone | less
00:25:27  pswch/s devintr/s  clones/s  loadavg1 loadavg5 loadavg15      _load_
00:25:27   379799     94510      2.00    147.14   143.95    130.22
00:25:37   379800     94510      3.00    147.14   143.95    130.22
00:25:47   379799     94510      3.00    147.14   143.95    130.22
00:25:57   379800     94509      2.00    147.14   143.95    130.22
00:26:07   379799     94510      3.00    147.14   143.95    130.22
[...]
01:26:37   372228     92055      1.00    227.85   220.75    206.53
01:26:47   372229     92054      2.00    227.85   220.75    206.53
01:26:57   372228     92055      1.00    227.85   220.75    206.53
01:27:07   372228     92055      2.00    227.85   220.75    206.53
```

Q. How is load average considered to be elevated/ high?
- Load Average always depends upon the number of processes in 'D' state and number of processes in running/ runnable state i.e. in runq.

- This system has 8 cores, so load average up to 8 is OK and system will not face any issues but it went beyond 8, that means there were more than 8 processes in running, runnable state or uninterrupted state which elevated the load average.

- As system is rebooted we cannot find process(es) stuck in various state from sosreport because system doesn't maintains historical data of process(es) but interestingly **PCP** does!

Analysis:

- Let's check process present in various states:

  - thrslpu [uninterruptible state] : 104
  - thrrun [running state] : 1
  - thrslpi [interruptible state] : 507

```
# pcp atopsar -P -r 20181211.00.25.0 --hostzone | head -n 20
[..]
00:25:27  clones/s pexit/s  curproc curzomb    thrrun thrslpi thrslpu _procthr_
00:25:27      2.00    0.00      961        0         1      507     104
00:25:37      3.00    0.00      961        0         1      507     104
00:25:47      3.00    0.00      961        0         1      507     104
00:25:57      2.00    0.00      961        0         1      507     104
00:26:07      3.00    0.00      961        0         1      507     104
00:26:17      7.00    0.00      961        0         1      505     104
00:26:27      8.00    0.00      961        0         1      505     104
00:26:37      8.00    0.00      961        0         1      505     104
00:26:47      7.00    0.00      961        0         1      505     104
00:26:57      8.00    0.00      961        0         1      505     104
00:27:07      8.00    0.00      961        0         1      505     104
[...]
```

- Lets verify using pmval:

```
-----------------------------------------------
# pmval -f 3 proc.runq.blocked -a 20181211.00.25.0 --hostzone | head -n 25
Note: timezone set to local timezone of host "server" from archive

metric:    proc.runq.blocked
archive:   20181211.00.25.0
host:      server
start:     Tue Dec 11 00:25:07 2018
end:       Tue Dec 11 01:27:07 2018
semantics: instantaneous value
units:     count
samples:   3721
interval:  1.00 sec
00:25:07.371  No values available
00:25:08.371        198
00:25:09.371        198
00:25:10.371        198
00:25:11.371        198
00:25:12.371        198
00:25:13.371        198
00:25:14.371        198
00:25:15.371        198

Documentation source: pcp-4.3.0/src/pmdas/linux_proc/help:
===============================================
@ proc.runq.runnable number of runnable (on run queue) processes
Instantaneous number of runnable (on run queue) processes;
state 'R' in ps(1).

@ proc.runq.blocked number of processes in uninterruptible sleep
Instantaneous number of processes in uninterruptible sleep or parked;
state 'D' in ps(1).
```

- Details:

  - Runnable: 2
  - Blocked [D]: 100+

```
# pmval -f 3 proc.runq.runnable -a 20181211.00.25.0 --hostzone | head -n 20
Note: timezone set to local timezone of host "server" from archive

metric:    proc.runq.runnable
archive:   20181211.00.25.0
host:      server
start:     Tue Dec 11 00:25:07 2018
end:       Tue Dec 11 01:27:07 2018
semantics: instantaneous value
units:     count
samples:   3721
interval:  1.00 sec
00:25:07.371  No values available
00:25:08.371         2
00:25:09.371         2
00:25:10.371         2
00:25:11.371         2
00:25:12.371         2
00:25:13.371         2
00:25:14.371         2
00:25:15.371         2
```

Analysis:

- Memory usage was high when load average was elevated:

```
------------------------------------------------------------------------------
ATOP - server     2018/12/11  00:25:17      ----------      1s elapsed
PRC | sys    1h47m  | user  17h14m | #proc    961    | #zombie    0 | no     procacct |
CPU | sys      4%   | user    33% | irq        0%   | idle    757% | wait       6% |
Cpu | sys      1%   | user     4% | irq        0%   | idle     95%  | cpu000 w  0% |
Cpu | sys      1%   | user     4% | irq        0%   | idle     95%  | cpu003 w  0% |
Cpu | sys      0%   | user     4% | irq        0%   | idle     95%  | cpu002 w  0% |
cpu | sys      1%   | user     4% | irq        0%   | idle     95%  | cpu001 w  0% |
cpu | sys      0%   | user     4% | irq        0%   | idle     93%  | cpu004 w  2% |
cpu | sys      0%   | user     4% | irq        0%   | idle     96%  | cpu005 w  0% |
cpu | sys      1%   | user     4% | irq        0%   | idle     93%  | cpu007 w  2% |
cpu | sys      0%   | user     4% | irq        0%   | idle     96%  | cpu006 w  0% |
------------------------------------------------------------------------------------------------------------
CPL  | avg1  147.14 | avg5  143.95 | avg15 130.22 | csw 914598e3 | intr 44511e4 | <<
------------------------------------------------------------------------------------------------------------
MEM | tot    23.5G   | free  271.2M | cache   5.2G | buff   84.6M | slab  406.5M | <<
------------------------------------------------------------------------------------------------------------
SWP | tot    29.0G   | free   28.9G |              | vmcom  21.2G | vmlim  40.7G |
PAG  | scan 14533e3 | steal 1310e4 | stall    816 | swin    4777 | swout  34411 |
```

Analysis:

- Almost 20 GB of memory usage.

```
----------------------------------------
# pcp atopsar -m -r 20181211.00.25.0 --hostzone | head -n 20
[..]
00:25:27  memtotal memfree buffers cached dirty slabmem  swptotal swpfree _mem_
00:25:27    24032M    271M     84M  5287M    1M    406M    29695M  29573M
00:25:37    24032M    271M     84M  5287M    1M    406M    29695M  29573M
00:25:47    24032M    271M     84M  5287M    1M    406M    29695M  29573M
00:25:57    24032M    271M     84M  5287M    1M    406M    29695M  29573M
00:26:07    24032M    271M     84M  5287M    1M    406M    29695M  29573M
00:26:17    24032M    271M     84M  5288M    0M    406M    29695M  29573M
00:26:27    24032M    271M     84M  5288M    0M    406M    29695M  29573M
```

Analysis:

- There are more than 100 process stuck in D state:

```
-------------------------------------------------------------
# pcp atop -r 20181211.00.25.0 --hostzone > names
# cat names |  awk '{if ($9 ~ "D") print $0}' | wc -l
104
```

- Lets see which process are those:

```
-----------------------------------------------------
# cat names |  awk '{if ($9 ~ "D") print $0}'
  PID SYSCPU USRCPU  VGROW  RGROW  RDDSK  WRDSK  THR S CPUNR  CPU CMD
 3975 15m12s  48m47s  620.6M   34612K    0K     0K     6   D     0     2%    BESClient
 7389  2m23s   0.00s     0K      0K    0K     0K     1   D     0     0%    rpciod/0
 2235 41.19s  83.00s  171.2M    2548K    0K     0K     1   D     5     0%     vmtoolsd
31256  0.61s  88.57s  186.6M   52480K    0K     0K     1   D     4     0%    perl
19571 19.57s  36.60s  491.4M   138.2M    0K     0K     5   D     3     0%    dsmc
 3154 39.48s  13.45s  79136K    1080K    0K     0K     1   D     3     0%    zabbix_agentd
  135 23.54s   0.00s     0K      0K    0K     0K     1   D     3     0%    khugepaged
 2576  7.76s   2.72s  18404K    604K    0K     0K     1   D     7     0%    irqbalance
 3156  6.69s   2.52s  81208K    1332K    0K     0K     1   D     1     0%    perl
 3155  6.67s   2.52s  81208K    1324K    0K     0K     1   D     1     0%     perl
 [...]
```

- We can identify the function in which process is stuck using “pcp pidstat -B D -a <archive>” feature by Nikhil.

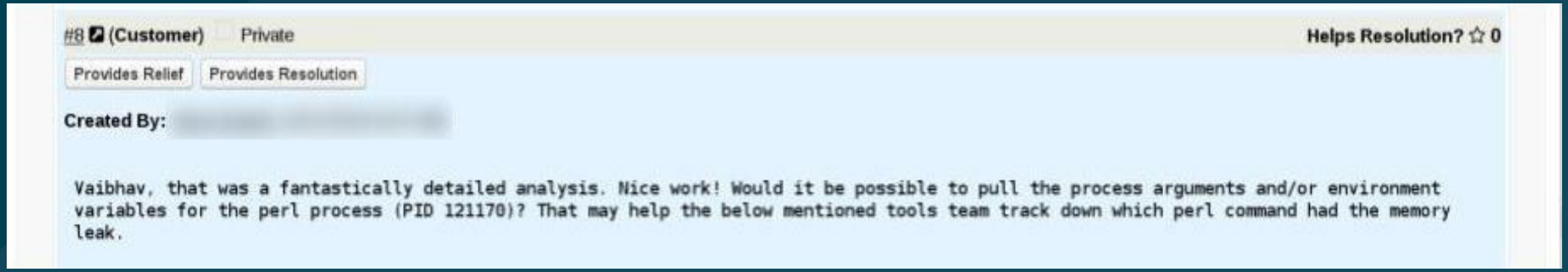
Analysis:

- Perl process was utilization was maximum memory:

  #pcp atop -m --archive 20181211.00.25.0| less

```
  PID VSTACK  VSIZE  RSIZE  PSIZE  VGROW  RGROW SWAPSZ  MEM CMD
121170   280K   3.0G  20.4G     ?K      0K  23664K 382.4M   77% perl          <<---
 19146   132K   3.4G 500.7M     ?K      0K   4352K 91548K    3% perl
 22891    10K 432.3M 317.1M     ?K  236.1M   14576K   828K    1% perl
  3632   132K 318.5M 20032K     ?K      0K    3664K 17856K    1% X
 22912   132K 84396K  5580K     ?K  84396K    5580K     0K    0% perl
 21646   132K 640.1M  5500K     ?K      0K    3272K  9656K    0% gnome-terminal
 18618   132K 735.6M  2848K     ?K      0K    -356K  2136K    0% gnome-session-
 19194   132K 451.6M  4152K     ?K      0K   -9556K  5048K    0% perl
  3518   132K 560.4M  3732K     ?K      0K   -12.3M  8256K    0% perl
 18618   132K 735.6M  3204K     ?K      0K   -5552K  2136K    0% perl
 19288   132K 367.5M  2680K     ?K      0K   -2412K   336K    0% gsd-a11y-setti
  2954   132K 601.9M  2664K     ?K      0K   -10.8M  7524K    0% perl
  3533   132K 982.4M  2612K     ?K      0K   -9028K  4496K    0% perl
  1551   132K 39816K  2568K     ?K      0K   -1380K   340K    0% systemd-journa
 19308   132K 367.5M  2520K     ?K      0K    -2516K   436K    0% gsd-mouse
```

Idea for new feature!


#8 (Customer) Private Helps Resolution? ☆ 0

Provides Relief | Provides Resolution

Created By: [redacted]

Vaibhav, that was a fantastically detailed analysis. Nice work! Would it be possible to pull the process arguments and/or environment variables for the perl process (PID 121170)? That may help the below mentioned tools team track down which perl command had the memory leak.

- Customer was interested to know process's arguments or environment variables.
- 'pidstat' has the option to display process arguments but it cannot be used with any/ same archive/ logs.
- 'ps' output is static.
- 'Top' does not maintains historical data.
- 'pcp pidstat' did not have this option.
- Thus I proposed an idea to add the feature which will display process arguments and environment variables to PCP maintainers.

My journey towards contributing in PCP

- My contribution in PCP.
- Challenges faced in :
  - Building from source
  - Coding
  - Qa
  - Documentation
  - Submitting the merge request for the feature via ‘git’

My contribution in PCP

pcp-pidstat: Adding support for -I flag. #534

New issue

Merged **goodwinos** merged 2 commits into `performancecopilot:master` from `nagarevaibhav:master` on Aug 10, 2018

Conversation 7 | Commits 2 | Checks 0 | Files changed 9 | +1,775 −14

nagarevaibhav commented on Jul 30, 2018

Adding support for -I flag. The -I flag/option displays the process command name and all its arguments which works on live system and also on archives. In addition to this -I flag can also be used with other flags and multiple usage of -I is avoided by displaying incorrect message

`Adding support for -l flag. The -l flag/option displays the process c…` ✓ `12b9a9b`

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

redhat.

What is pcp pidstat -l ?

- Display the process command name and all its arguments.
- Works on live system.
- Also works on pcp archives/ logs. [USP]
- It can be used in combination with other flags.

```
# pcp pidstat -l -r --a 20190111.18.57.0
==============================
Timestamp       UID       PID  MinFlt/s       MajFlt/s      VSize   RSS     %Mem  Command
19:00:51          0     22873   5789.93         391.61    3099584 1434064 76.2  /usr/bin/perl -w ./perl 2900 <<----
19:00:51         89     22878      0.0            0.0      91876   168     0.01  cleanup -z -t unix -u
19:00:51          0     22879      0.0            0.0     226140   160     0.01  /usr/libexec/abrt-handle-event -i -e
post-create -- /var/spool/abrt/ccpp-2019-01-11-18:58:49-19394
19:00:51         89     19146      0.0            0.0      91732   860     0.05  /usr/bin/perl -w ./perl 1000
19:00:51         89      3533      0.0            0.0      91732   860     0.05  /usr/bin/perl -w ./perl 800
```

Challenges faced: Building from source

- Compilation failed [missing packages]

  Thanks to Iberk, who helped me with this.

- Install all packages that come after executing following command

  # ./qa/admin/check-vm -p

- Manually enabling required repositories and installing the packages.
- Compile and install.
  #./Makepgs
  # yum localinstall *.rpm -y

Coding: Implementation of idea

- Big thanks to Nikhil who mentored & helped me in understanding the “pcp pidstat” code.
- Finding the metric.
  #pminfo
- Feature should be able to:
  - Display the process command name and all its arguments.
  - Use as a standalone
  - In combination with other options/ flags as a standalone.
  - On archives/ logs
  - In combination with other options/ flags on archives.

- Further challenges:
  - Adding configuration in "pmlogconf" so that archives/ logs have required data needed by feature [thanks to mgoodwin for this].
  - Multiple usage of flags is avoided.

Challenges faced: Qa

- Understanding how ‘qa’ works.
- Write a test script.
- How to use archives for testing copied under qa/archives.
- Testing the feature using test script and archives.
- Current challenge:
    - Trying to understand why unit tests under “src/pcp/pidstat/test” are failing.

Challenges faced: Documentation

- Finding appropriate description for the feature.
  #man pcp pidstat

```
       reporting  timezone is the local timezone, which may not be the same as the timezone of the PCP
       archive).

  -l   Display the process command name and all its arguments.

  -? , --help
       Display help and exit
```

#pcp pidstat --help

```
  -R                    Report realtime priority and scheduling policy information.
  -r                    Report page faults and memory utilization.
  -k                    Report stack utilization.
  -f                    Format the timestamp output
  -B                    Report process state information. Use -B [all] or -B [comma separated states]. Use -B de
tail for showing time spent in every state per process
  -V, --version         display version number and exit
  -Z TZ, --timezone=TZ  set reporting timezone
  -z, --hostzone        set reporting timezone to local time of metrics source
  -l                    Display the process command name and all its arguments.
  -?, --help            show this usage message and exit
[root@vaibhav test]#
```

Submitting the merge request.

- Understanding the git.
- Setting username and email
  # git config --global user.name “name”
  # git config --global user.email “email”

- Fork -> git clone -> making changes in required files.
- Creating different branch [git branch <branchname>]
- Commit the changes along with commit message [git commit]
- Pushing the commit [git push remote or git push origin <branchname>]
- Open pull request.
- Submit the merge request.
- Follow up commits.

Contributions so far...

- Pcp pidstat -I : https://github.com/performancecopilot/pcp/pull/534
- pcp pidstat -zZ: https://github.com/performancecopilot/pcp/pull/512
- Follow up commit: https://github.com/performancecopilot/pcp/pull/553


QUESTIONS?


redhat.
THANK YOU