grafana-pcp Documentation

Release 3.2.1

Performance Co-Pilot

Nov 24, 2021

Getting started

1	Featu	ires	3
2	Getti	ng started	5
	2.1	Quickstart	5
	2.2	Installation	
	2.3	Screenshots	7
	2.4		14
	2.5	Change Log	14
	2.6	Overview	21
	2.7	Authentication	22
	2.8	PCP Redis	23
	2.9	PCP Vector	24
	2.10	PCP bpftrace	25
	2.11	Multiple Vector Hosts	26
	2.12	Monitoring Containers	27
	2.13	Troubleshooting	28

Performance Co-Pilot (PCP) provides a framework and services to support system-level performance monitoring and management. It presents a unifying abstraction for all of the performance data in a system, and many tools for interrogating, retrieving, and processing that data.

CHAPTER 1

Features

- analysis of historical PCP metrics using pmseries query language
- · analysis of real-time PCP metrics using pmwebapi live services
- enhanced Berkeley Packet Filter (eBPF) tracing using bpftrace scripts
- dashboards for detecting potential performance issues and show possible solutions with the checklist dashboards, using the USE method [2]
- full-text search in metric names, descriptions, instances [1]
- support for Grafana Alerting [1]
- support for derived metrics (allows the usage of arithmetic operators and statistical functions inside a query) [2]
- automated configuration of metric units [1,2,3]
- · automatic rate and time utilization conversion
- heatmap, table [2,3] and flame graph [3] support
- auto-completion of metric names [1,2], qualifier keys and values [1], and bpftrace probes, builtin variables and functions [3]
- display of semantics, units and help texts of metrics [2] and bpftrace builtins [3]
- legend templating support with \$metric, \$metric0, \$instance, \$some_label, \$some_dashboard_variable
- container support [1,2]
- support for custom endpoint and hostspec per panel [2,3]
- support for repeated panels
- · sample dashboards for all data sources

[1] PCP Redis [2] PCP Vector [3] PCP bpftrace

CHAPTER 2

Getting started

- Quickstart
- Installation

2.1 Quickstart

2.1.1 Installation

Please see the *Installation Guide*. There is a simple method using the package manager for Red Hat-based distributions, otherwise it can be installed from source, from a pre-built plugin bundle from the project's GitHub releases page, or as a container.

Make sure to restart Grafana server and pmproxy after installation the plugin. Eg.

```
$ sudo systemctl restart grafana-server
$ sudo systemctl start pmproxy
```

Installation is not finished until you also enable the Performance Co-Pilot plugin via the Grafana Admin configuration:

Open the Grafana configuration, go to Plugins, select *Performance Co-Pilot* and click the *Enable* button on it's page. This will make the PCP data sources and some dashboards available.

2.1.2 Data Sources

Before using grafana-pcp, you need to configure the data sources. Open the Grafana configuration, go to Data Sources and add the *PCP Redis*, *PCP Vector* and/or *PCP bpftrace* datasources.

The only required configuration field for each data source is the URL to pmproxy. In most cases the default URL http://localhost:44322 can be used. All other fields can be left to their default values.

Note: Make sure the *URL* text box actually contains a value (font color should be white) and you're not looking at the placeholder value (light grey text).

Note: The Redis and bpftrace data sources need additional configuration on the collector host. See *PCP Redis* and *PCP bpftrace*.

2.1.3 Dashboards

After installing grafana-pcp and configuring the data sources, you're ready to open the pre-installed dashboards or create new ones. Each data source comes with a few pre-installed dashboards, showing most of the respective functionality. Further information on each data source and the functionality can be found in the *Data Sources* section.

2.2 Installation

2.2.1 Distribution Package

Distribution Package is the recommended method of installing grafana-pcp.

Fedora

```
$ sudo dnf install grafana-pcp
$ sudo systemctl restart grafana-server
```

2.2.2 GitHub Release

If there is no package available for your distribution, you can install a release from GitHub. Replace X.Y.Z with the version of grafana-pcp you wish to install.

2.2.3 Container

You can also run Grafana with grafana-pcp in a container, using podman or docker. Keep in mind that with the default configuration, every container has its own isolated network, and you won't be able to reach pmproxy through localhost. Replace X.Y.Z with the version of grafana-pcp you wish to install.

```
$ podman run -e GF_INSTALL_PLUGINS="https://github.com/performancecopilot/grafana-pcp/

oreleases/download/vX.Y.Z/performancecopilot-pcp-app-X.Y.Z.zip;performancecopilot-

opcp-app" -p 3000:3000 docker.io/grafana/grafana
```

(continued from previous page)

2.2.4 From Source

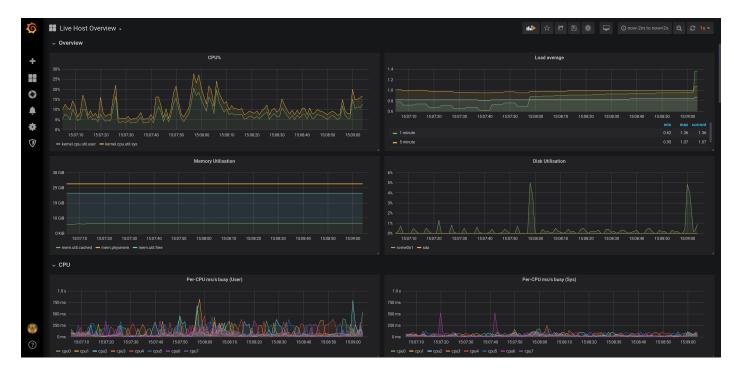
The yarn package manager, Go compiler, jsonnet and jsonnet bundler are required to build grafana-pcp.

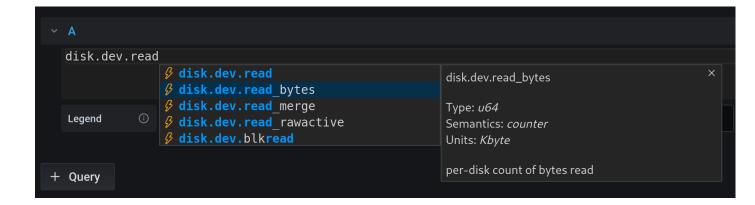
```
$ git clone https://github.com/performancecopilot/grafana-pcp.git
$ make build
$ sudo ln -s $(pwd) /var/lib/grafana/plugins
$ sudo systemctl restart grafana-server
```

To list all available Makefile targets, run make help.

2.3 Screenshots

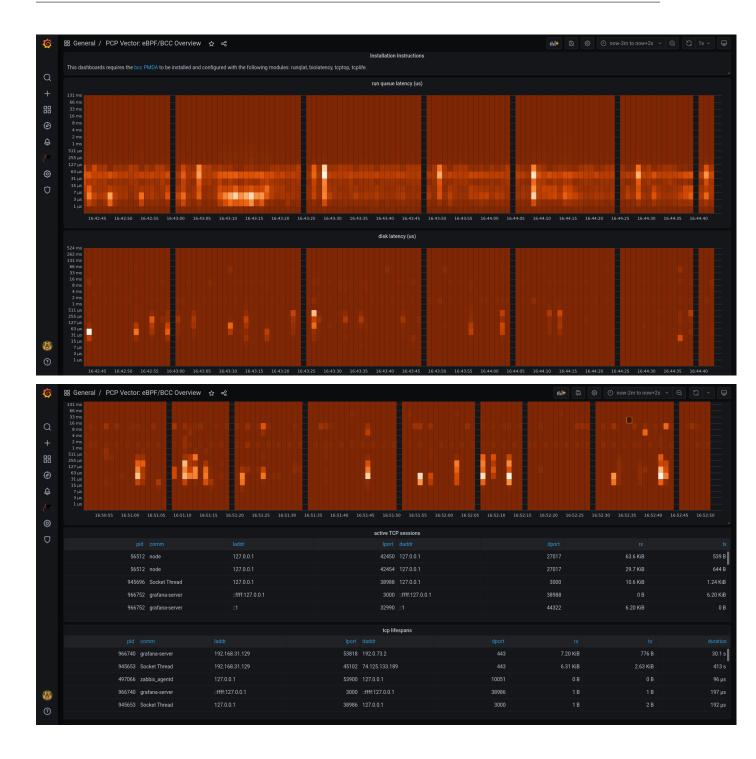
2.3.1 PCP Vector





Vector dashboards







2.3.2 PCP bpftrace



🗸 Disk														
							Dia-1-1/0 Internet							
66 ms —							Block I/O latency							
33 ms														
8 ms — 4 ms —														
2 ms — 1 ms —														
511 μs — 255 μs —														
127 μs — 63 μs —														
31 μs — 15 μs —														
	5:20 20:05:2	5 20:05:30 20:0	5:35 20:05:40 20:05:45	20:05:50 20:0)5:55 20:06:00 2	20:06:05 20:	:06:10 20:06:15 20:06:20 20:	06:25 20:06	:30 20:06:35 20:06:40	0 20:06:45 20:06:	50 20:06:55	20:07:00	20:07:05 20:07:10	20:07:1
✓ Filesyste	em													
							virtual file system calls							
						<u>×1</u>								
0 ops2	0:05:20 20:0	5:25 20:05:30 :	20:05:35 20:05:40 20:05:45	20:05:50	20:05:55 20:06:00	20:06:05	20:06:10 20:06:15 20:06:20	20:06:25 20	06:30 20:06:35 20:06-	40 20:06:45 20:0	6:50 20:06:55	20:07:00	20:07:05 20:07:10	0 20:07
	e System ,	Analysis +								ulit 🏠	C B i	* 🖵	now-2m to now+	-2s Q
	20:17:05 20: range — vfs_ge	17:10 20:17:15 tattr — vfs_getattr_nos	20:17:20 20:17:25 20:17:30 ec — yfs_lock_file — yfs_open — Is_create_mount — yfs_get_super			— vfs_statfs -			0:18:15 20:18:20 20:18 v/fs_writev — vfs_fadvise -		18:35 20:18:4	40 20:18:45	20:18:50 20:18:5	55 20:
	20:17:05 20: ,range — vfs_ge tr — vfs_mknoo	17:10 20:17:15 tattr — vfs_getattr_nos	ec — vfs_lock_file — vfs_open =		readlink — vfs_rename	— vfs_statfs -					18:35 20:18:4	40 20:18:45	20:18:50 20:18:5	55 20:
- vfs_fsync_ - vfs_setxatt	20:17:05 20: ,range — vfs_ge tr — vfs_mknoo	17:10 20:17:15 tattr — vfs_getattr_nos	ec — vfs_lock_file — vfs_open =	– vfs_read –– vfs_ –– vfs_get_tree ––	readlink — vfs_rename	— vfs_statfs -					18:35 20:18:4 nk — vfs_fallocat	40 20:18:45	20:18:50 20:18:5	55 20:
vfs_fsync_ vfs_setxatt	20:17:05 20: ,range — vfs_ge tr — vfs_mknoo	17:10 20:17:15 tattr — vfs_getattr_nos	ec — vfs_lock_file — vfs_open = is_create_mount — vfs_get_super	– vfs_read –– vfs_ –– vfs_get_tree ––	readlink — vfs_rename	— vfs_statfs -				3:25 20:18:30 20: - vfs_mkdir — vfs_symin	18:35 20:18:4 nk — vfs_fallocat	40 20:18:45	20:18:50 20:18:5	55 20: getxattr_allo
v ups — vfs_fsync_ — vfs_setxatt ✓ Network	20:17:05 20: ,range — vfs_ga tr — vfs_mknoo	17:10 20:17:15 tattr — vfs_getattr_nos — vfs_test_lock — v	ec — vfs_lock_file — vfs_open = fs_create_mount — vfs_get_super trace TCP acc	- vfs_read - vfs_ - vfs_get_tree - ept()	readlink — vfs_rename vfs_parse_fs_param —	— vfs_statfs • vfs_parse_fs_stri	– vfs_statx — vfs_statx_fd — vfs_unlink ing — vfs_removexattr — vfs_link		– vfs_writev – vfs_fadvise –	325 20:18:30 20: – vfs_mkdir — vfs_symir trace TCP ce	18:35 20:18:4 nk — vfs_fallocat	40 20:18:45 :e — vfs_rmdir	20:18:50 20:18:5 - vfs_getxattr - vfs_g	55 20: getxattr_allo
vfs_fsync_ vfs_fsync_ vfs_setxatt Network	20:17:05 20: range – vfsge tr – vfs_mknoo ; PID	17:10 20:17:15 tattr — vfs.getattr.nos — vfs.test.jock — v COMM	ec — vfs_jock_file — vfs_get_super (s_create_mount — vfs_get_super trace TCP acc RADDR	- vfs_read - vfs_ - vfs_get_tree - ept() RPORT	readink — vfs_rename vfs_parse_fs_param — LADDR	vfs_statfs • vfs_parse_fs_stri	Hs_statx - vfs_statx_fd - vfs_unlink ing - vfs_removexattr - vfs_link BL TIME -		v/s_writev — v/s_fadvise -	825 2018:30 20 - vfs_mkdir — vfs_symir trace TCP ct SADDR	18:35 20:18:4 nk — vfs_fallocat	40 20:18:45 ise — vfs_rmdir SPORT	2018:50 2018:5 - vfs.getxattr - vfs.g	55 20: getxattr_allo
vis_fsync_ vis_fsync_ vis_setxati Network TIME ▼ 20:17:55	20:17:05 20: range – vfs_ge tr – vfs_mknoo : PID 3888	7:10 20:17:15 tattr — vfs_getattr_nos — vfs_test_lock — vr COMM pmcd	ec = vfs_lock_file = vfs_open = fs_create_mount = vfs_get_super trace TCP acc RADDR 127.0.0.1	ept() RPORT 38034	readinx — vfs_rename vfs_parse_fs_param — LADDR 127.0.0.1	LPORT 44321	Hastatx - vfa_statx fo - vfa_sulink ng - vfa_emovexatr - vfa_link BL 0/5 20:18:53		v/s_writev - v/s_fadvise - COMM curl	825 20:18:30 20: = vfs_mkdir — vfs_symin trace TCP of SADDR 192:168.0.80	18:35 20:18:4 nk — vfs_fallocat	40 20:18:45 te — vfs_rmdir SPORT 33536	2018:50 2018: - vfs.getxattr - vfs.g DADDR 152.199.19.160	55 20: getxattr_allo
vrs.faync. vrs.faync. vrs.sexxatt Network TIME ▼ 20:17:55 20:17:55	20:17:05 20: range - vfs_ga tr - vfs_mknoo PID 3888 3888	7:10 20:17:15 tattr = vfs.getattr_nos = vfs_test.lock = vf COMM pmcd pmcd	ec vfs.jock.file vfs.open s s.oreate_mount vfs.get_super trace TCP acc RADDR 127.0.0.1 127.0.0.1	- vfs_read vfs_ vfs_get_tree ept() RPORT 	LADDR 127.0.0.1 127.0.0.1	LPORT 44321 44321	v f _L ,atat = vf _L ,atat. [d = vf _L ,atat. ng = vf _L ,emovesatt = vf _L ,link BL 0/5 0/5 20:18:50 20:18:50		- vfs_vvitev vfs_fadvise - COMM curl wget	825 20:18:30 20: - vfs_mkdir — vfs_symin trace TCP cr SADDR 192:168.0.80 192:168.0.80	18:35 20:18:4 ık — vfs_fallocat	40 20:18:45 10 vfs_rmdir 10 SPORT 10 33536 10 59034	2018:50 2018:5 vfs_getxattr — vfs_getxattr DADDR 152.199.19.160 192.30.253.120	55 20: getxattr_allo
 vfs_fsync_ vfs_fsync_ vfs_setxatt Network TIME = 20:17:55 20:17:55 20:17:55 	20:17:05 20: range vfs.gr tr vfs_mknoo PID 3888 3888 3888	7:10 20:17:15 tatt - Vs_getatt_nos - Vs_testock - v pmcd pmcd pmcd	ec – vfs.joct.dfe – vfs.get.appen – fs.cetelar_mount – vfs.get.apper trace TCP acc RADDR 127.0.0.1 127.0.0.1	- vfs_read vfs_ vfs_get_tree ept() RPORT 38032 38032 38030	eadlink = vfs_rename vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1 127.0.0.1	LPORT 44321 44321	v f _L ,atat = vf _L ,atatfd = vf _L ,utiak ng = vf _L ,emovesatt = vf _L ,liak BL 0/5 0/5 0/5 20.18.53 20.18.50 20.18.50		-vfs_vvitev vfs_fadvise - COMM curl wget wget	825 201830 20: → (fs,mkdir → (fs,symin trace TCP ci SADDR 192168.0.80 192168.0.80	18:35 20:18:4 ık — vfs_fallocat	0 20:18:45 = vfs_mdir SPORT 33536 59034 59154	2018:50 2018: vfs.gebattr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4	55 20: getxatir_alio
(uqus)	20.17.05 20. rrange - vfs_ge tr - vfs_mknod 20.17.05 20. 20.17.05 20	7:10 20:17:15 tatt - vfs_getatt_nos - vfs_test_lock - v pmcd pmcd pmcd pmcd	ec – vfs.joct.dfe – vfs.get.appen – fs.getetle_mount – vfs.get.appen trace TCP acc RADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1		LADDR 127 0.01 127 0.01 127 0.01 127 0.01 127 0.01 127 0.01 127 0.01 127 0.01	UPORT 44321 44321 44321 44321 44321 44321 44321			vfs_writev vfs_fadvise - COMM curl wget wget NetworkManager	825 201830 20. − vfs_mkdr = vfs_synthr trace TCP of SADDR 192.168.0.80 192.168.0.80 192.168.0.80 192.168.0.80	18:35 20:18:4 ık — vfs_fallocat	20:18:45 e - vfs_imdir 33536 59034 59154 50088	201839 20184 vfs.getaarr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1	55 20: getxattr_alko
(uqus	2017.05 200 range – vfs.grander tr – vfs.mknot 3868 3868 3868 3868	7:10 20:17:15 tatt - Yfs_detatt_nos - Yfs_tetUock - Y pmcd pmcd pmcd pmcd pmcd pmcd	ec - vfs.joct.dfe - vfs.get.apper - fs.geater.mount - vfs.get.apper RADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1		EADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	LPORT 44321 44321 44321 44321 44321	vfs_tatt = vfs_tatt_id = vfs_tatt_id vfs_tenovesatt = vfs_tatt 0/5 0/5 0/5 0/5 0/5 0/5 0/5 20:18:10 0/5 20:18:10		vfs_writev — vfs_faddrise = COMM curl wget wget NetworkManager NetworkManager	225 2018.30 20. − vfs_mkdr = vfs_synthr trace TCP of SADDR 192.168.0.80 192.168.0.80 192.838.8463. 192.168.0.80	18:35 20:18:4 ık — vfs_fallocat	20:18:45 e - vfs_mdir SPORT 33536 59034 59154 50088 47540	2018.30 2018.4 vfs.getuatr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198	55 20: getxattr_alko
(uqus)	2017.05 200 trange Vis.generation tr Vis.generation 3868 3868 3868 3868 3868	7/10 20:17:15 tatr - Vis_getatr_nos - Vis_getatr_nos - Vis_getatr_nos - Pimed - Pimed - Pimed - Pimed - Pimed - Pimed - Pimed - Pimed - Pimed	ec – vfs.joct./de – vfs.get.appen – fs_center_mount – vfs.get.appen – razer TCP acc 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1	-vfs_readvfs_get_tree -vfs_get_tree ept() 	LADDR 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1		• fs_stati • vfs_static_ld • vfs_unitek • ng • vfs_vernovesatit • vfs_unitek 0/5 20.18.53 20.18.53 0/5 20.18.50 20.18.10 0/5 20.18.10 20.18.10 0/5 0/5 20.17.55 0/5 3 20.17.55		vfs_writev — vfs_ladvise = COMIM curl wget wget NetworkManager pmwebd pmwebd	225 2018.30 20. − vfs_mkdr = vfs_symth trace TCP of SADDR 192.168.0.80 192.168.0.80 192.8388.w8c3: 192.168.0.80 192.168.0.80 192.168.0.80 192.168.0.80	18.35 20.18-4 + vfs_falocal connect() :a00.0 transmits	201845 201845 201845 201845 201845 20184 201845 20184 201845 20185 2018	201839 20184 vfs.getaarr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1	55 20: getxattr_alio
• vfs_fsync_ • vfs_fsync_ • vfs_setxatt • vfs_se	2017/05 200 range - vfsget r - vfsget PID 3888 3888 3888 3888 3888 3888 3888 3888 3888	7:10 20:17:15 tatr - Ys_getatr_nos - Ys_getatr_nos - Ys_getatr_nos - Pmcd - pmcd	ec – vfs.joct./de – vfs.get.appen – fs_center_mount – vfs.get.appen – raceetTCP acc 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1	-vfs.jread -vfs.jrea ept/) PPORT 38034 38032 38030 38028 38026 38024 38022 38026 38022 38022 0905 DADDR:D	LADDR 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1		• fs_ath • vf_s_ath - vf_s_ath - vf_s_ath • fs_ath • vf_s_ath - vf_s_ath - vf_s_ath • vf_s_emovesatt - vf_s_ath - vf_s_ath - vf_s_ath • vf_s - vf_s_ath - vf_s_ath - vf_s_ath - vf_s_ath • vf_s - vf_s - vf_s_ath - vf_s_at		vfs_writev — vfs_ladvise = COMM curl wget wget NetworkManager pmwebd pmwebd LADDR:LPORT	325 2018.30 20. - νfa_rradit - wfa_symbol 20. trace TCP col SADOR 192.168.0.80 192.168.0.80 20.2.8388.863: 192.168.0.80 127.0.0.1 127.0.0.1 127.0.0.1 trace TCP col	18.35 20.184 + vfs_fallocal onnect() ::a00.0 transmits RADDR:RP0	 20:18:45 20:18:45 90 90 90 90 90 91 91 94 94 95 96 96 97 96 97 97<td>201830 20183 vfs.getatt vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1 127.0.0.1</td><td>55 20:0- getxatr_allo STATE</td>	201830 20183 vfs.getatt vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1 127.0.0.1	55 20:0- getxatr_allo STATE
vis_fayne, vis_fayne, vis_setant vis_seta	2017.05 20. range - vfs.get vfs.get PID 3888 3888 3888 3888 3888 3888 3888 3888 3888 3888 3888 3888	7:10 20:12:15 tatr - Ys_detatr_nos - Ys_test_lock - v pmcd pmcd pmcd pmcd pmcd pmcd pmcd comM ksoftirqd/7	ec		LADDR 127.0.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1		• fig.ath • vig.ath • vig.ath • vig.ath BL 0/5 20.18.53 0/5 20.18.50 20.18.50 0/5 20.18.10 20.18.10 0/5 20.18.10 20.17.55 0/5 20.17.55 20.17.55		vfs_writev — vfs_ladvise = COMM curl wget NetworkManager pmwebd pmwebd LADDR:LPORT 2a02 8388-9aa7::a00.04	225 2018.0 20. - vf_s,rkdr - vf_s,vhir SADDR 192.168.0.80 192.168.0.80 2002.8388.8663 2002.8388.8663 192.168.0.80 127.0.0.1 127.0.0.1 Lizzo.1 2200.5	18.35 20.18.4 • vfs_fallocal onnect() ::a00.0 transmits RADDR:RP0 2:a04:fa87:5	SPORT 33536 59034 59154 50088 47540 38034 38032 RT 0000:4c41:535	201830 20184 vfs.getattr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178:5000: 152.19.134.198 127.0.0.1 127.0.0.1	55 20:0- getxattr_allo STATE LAST_
vis_repart vis_rectain vis_recta	2017.05 202 range - vfs_get r - vfs_mkono 2017.05 202 vfs_mkono 2017.05 202 PID 38888 38888 3888 3888 3888 38888 3888	27:10 20:17:15 tattr — vfs_getattr_nos → vfs_destJock — v pmcd	ec _ +fs.pot.dfa _ +fs.ppt.appr _ g.cmata_mourt _ +fs.pet.appr _ 127.0.0.1 127.0.0 127.	-vfs.jread - vfs.jet. ept() ept() 88032 38032	LADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	vis_parse_fs_starts UPORT 44321 44321 44321 44321 44321 44321 44321 44321 5TI ES ES			vfs_writev vfs_fadvise COMM curl wget NetworkManager pmwebd pmwebd LADDRLPORT 2a02.8388.9a87:a00.0/ 2a02.8388.9a87:a00.0/	225 2018.0 202 voff_nRdf - vf_agent trace TCP co SADDR 192.168.0.80 192.168.0.80 2002.8388.963: 192.168.0.80 1227.0.1 1227.0.1 trace TCP ref 42906	1835 20.184 x - vfs_fallocat connect() :a00.0 transmits RADDR:RPO 2a04.fa87.5 2a04.fa87.5	20:18:45 60 20:18:45 Formation SPORT 33536 59034 59154 50088 47540 38034 38032 RT 0000:4c41:535	201830 20184 • vfs.getoattr • vfs.ge DADDR 152, 199,19,160 192,20253,120 140,82,118,4 20014178:5000: 152,19,134,198 127,0,0,1 127,0,0,1 54,80	55 20:0 getxattr_allo STATE LAST_ LAST_
vts_faync_ vts_faync_ vts_setaat vts_seta	2017.05 20. range – vfspe tr – vfsmknae PID 3888 3888 3888 3888 3888 46 46 46	2710 201715 tatr — vfs_getatr_nos → vfs_desUck — v pmcd	ec – vfs.joct.dfa – vfs.get.asper , centar, mourt – vfs.get.asper RADDR 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1	-vfs.jread - vfs.jet. ept/l pet/l	Exaboration - Vfa_remarine vfa_pares_fs_param - 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	vfs_stafs = vfs_stafs vfs_parse_fs_stafs vfs_parse_fs_stafs 44321 44321 44321 44321 44321 44321 44321 555 ES ES	• fs_state • vfs_state_fd • vfs_state_fd • fs_state • vfs_state_fd • vfs_state_fd 0/s 20.18.53 20.18.53 0/s 20.18.50 20.18.53 0/s 20.18.50 20.18.50 0/s 20.18.50 20.18.50 0/s 20.18.50 20.18.50 0/s 20.18.50 20.18.10 0/s 20.17.55 20.17.55 0/s 20.17.55 20.15.03 TABLISHED 20.15.03 TABLISHED 20.14.36		vfs_writev vfs_Tadvise COMM Curl Curl Wget Wget NetworkManager NetworkManager pmwebd pmwebd LADDR:LPORT 2a02:8388-9as7::a00.0/ 2a02:8389-9as7::a00.0/ 2a02:8389-9as7:*a00.0/ 2a	225 2018.0 202 offmkdr = offsynta Crace TCP cr 5.0007 192.168.0.80 20202388.8483: 192.168.0.80 2020248.84843: 192.168.0.80 2020248.84843: 192.168.0.80 2020248.84843: 192.168.0.80 2020248.84843: 202054 20	1835 20.18.4 k → vfs_fallocat ennect() a00.0 transmits RADDR:RPO 2a04.fa87.5 2a04.fa87.5 2a04.fa87.5	SPORT 33536 59134 59134 59134 59336 59034 59034 59034 38034 38032 38032	2018.30 2018.4 • vfs.getoattr • vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1 127.0.0.1 54.80 54.80	STATE LAST_LAST_LAST_
vts_faync_ vts_faync_ vts_setaat vts_vts_setaat 20:17:55 20:18:57	2017.05 20. range – vfspe tr – vfsmknae PID 3888 3888 3888 3888 3888 3888 46 46 46 46 0	7:10 201715 tatr — vfs_getatr_nos → vfs_getatr_nos → vfs_getatr_nos → vfs_getatr_nos COMM COMM ksoftrqd/7 swapper/1	ec – vfs.joct.dfa – vfs.get.asper g.cenata.mourt – vfs.get.asper RADDR 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.0	-vfs.jread - vfs.jet.ive - eptil: eptil: eptil: 89034 38034 38032 38028 38028 38028 38028 38028 38022 0 8022 0 0 0 0 0 0 0 0 0 0 0 0 0	EADDR 127.0.1	vfs_statfs = vfs_statfs = 44321 44321 44321 44321 44321 44321 44321 44321 517 517 517 518 518 518 518 518 518 518 518 518 518	• fs_state - vfs_state_fd - vfs_united • fs_state - vfs_state_fd - vfs_united 0/5 20.18.53 20.18.53 0/5 20.18.53 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.10 20.17.55 0/5 20.17.55 20.17.55 0/5 20.17.55 20.17.55 1ABUSHED 20.16.33 20.14.36 1ABUSHED 20.14.15 20.14.15		vfs_writev vfs_Tadvise COMM curl curl wget NetworkManager NetworkManager pmwebd pmwebd LADDR1.PORT 2a02.8388-9a37:::00.04 2a02.8388-9a37:::00.04 2a02.8388-9a37:::00.04	225 2018.0 20 ovfl_mkdr - vfl_mkgr Crace TCP cr SADDR 192.168.0.80 192.168.0.80 2022.838.8483: 2022.838.8483: 2022.0.1 127.0.1 127.0.1 2202.6 2202.6 2205	1835 20.184 k → vfs_fallocat connect() a60.0 transmits RADDR.RPO 2404.fa87.5 2404.fa87.5 2404.fa87.5	SPORT 33536 59034 59134 59154 38034 38032 38032	2018.30 2018.4 vfs.getattr vfs.ge DADDR 152.199.19.160 192.30 253.120 140.82.118.4 20014178.5000: 152.19.134.198 127.0.0.1 127.0.0.1 127.0.0.1 54.80 54.80	STATE LAST_ LAST_ LAST_ LAST_ LAST_ LAST_
vfs_fayac_ vfs_fayac_ vfs_retatl vfs_vfsyster vfs_retatl 20:17:55 20:18:57	2017.05 20. range – vfsps tr – vfsmknack PID 3888 3888 3888 3888 3888 3888 46 46 46 46 46 0 0 0	2710 201715 tatr = V5_getatr_nos = V5_getatr_nos = V5_getatr_nos = V5_getatr_nos = 000000000000000000000000000000000000	ee – vfs.joct.dfa – vfs.get.japer g.cenate.mourt – vfs.get.japer RADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 0.0.2 SADDR:SPORT 0.023ad::a00.0.44323 0.023ad::a00.44323	-vfs.jread - vfs.jer. ept() - vfs.get.tree - ept() - vfs.get.tree - sept() - vfs.get.tree -	Baadirok - Vfl.remanne Vfl.remanne Vfl.remanne 127.0.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 127.0.1 4253.5441:58544 4553.5441:58546 4553.5441:58546 4553.5441:58546	vfs_starts = vfs_starts = 44321 44321 44321 44321 44321 44321 44321 44321 44321 557 555 555 555 555 555 555	• fs_state - vfs_state_id - vfs_united • fs_state - vfs_state_id - vfs_united 0/5 20.18.53 20.18.53 0/5 20.18.53 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.50 20.18.50 0/5 20.17.55 20.17.55 0/5 20.17.55 20.17.55 0/5 20.18.10 20.14.10 TABUSHED 20.14.36 20.14.36 TABUSHED 20.14.36 20.14.22 TABUSHED 20.14.11 20.14.11		Vfs_vvrtev vfs_Tadvise COMM curl wget wget NetworkManager provebd provebd LADDR:LPORT 2002:8388:9a37::000.04 2002:8388:9a37::000.04 2002:8388:9a37::000.04 2002:8388:9a37::000.04	200 201 2	1835 20184 k → vfs_fallocat connect() transmits RADDR.RPO 2a04.fa87.5 2a04.fa87.5 2a04.fa87.5	SPORT 33536 S9034 59134 S9034 59134 S9034 38032 RT 000:4c41:535 000:4c41:535 000:4c41:535	2018.30 2018.4 vfs.gebattr vfs.ge DADDR 152.199.19.160 192.30 253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 152.9.0.1 152.9.0.1 152.9.0.1 152.9.0.1 152.9.0.1	STATE LAST, LAST, LAST, LAST, LAST, LAST,
vts_faync_ vts_faync_ vts_setaat vts_vts_setaat 20:17:55 20:18:57	2017.05 20. range – vfspe tr – vfsmknae PID 3888 3888 3888 3888 3888 3888 46 46 46 46 0	7:10 201715 tatr — vfs_getatr_nos → vfs_getatr_nos → vfs_getatr_nos → vfs_getatr_nos COMM COMM ksoftrqd/7 swapper/1	ec – vfs.joct.dfa – vfs.get.asper g.cenata.mourt – vfs.get.asper RADDR 127.00.1 127.00.1 127.00.1 127.00.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.00.1 0.1 127.0	-vfs.jread - vfs.jer. ept() - vfs.get.tree - ept() - vfs.get.tree -	EADDR 127.0.1	vfs_starts = vfs_starts = 44321 44321 44321 44321 44321 44321 44321 44321 44321 555 555 555 555 555 555 555 555 555 5	• fs_state - vfs_state_fd - vfs_united • fs_state - vfs_state_fd - vfs_united 0/5 20.18.53 20.18.53 0/5 20.18.53 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.50 20.18.50 0/5 20.18.10 20.17.55 0/5 20.17.55 20.17.55 0/5 20.17.55 20.17.55 1ABUSHED 20.16.33 20.14.36 1ABUSHED 20.14.15 20.14.15		vfs_writev vfs_Tadvise COMM curl curl wget NetworkManager NetworkManager pmwebd pmwebd LADDR1.PORT 2a02.8388-9a37:::00.04 2a02.8388-9a37:::00.04 2a02.8388-9a37:::00.04	200 201 2	20184 vts_falocat vts_falocat vts_falocat vts_falocat vts_falocat	SPORT 33536 59034 59134 59154 38034 38032 38032	201830 20183 vfs.gstatt vfs.g DADDR 152.199.19.160 192.30 253.120 140.82.118.4 2001.4178.5000: 127.0.0.1 127.0.0.1 127.0.0.1 54.80 54.80 54.80 54.80 54.80	55 20:1

bpftrace code editor

- A												
<u> </u>												
	15	.			4							
	16 17	tracepoint:sc	nea:scned	і_wакеup				_	and stan			nnaha 🛛
	18	Patimo[angs_prid] = psocs										
	19	}		,						L		probe
	20			tracepoint:sched:sched_migrate_task probe red_switch_tracepoint:sched:sched_move_numaprobe								
	21	tracepoint:sc	hed:sched	l_switch								probe
	22 23	{ if (angr-)n	nov state	тлс			ed:sched_					probe
	23	if (args->p @qtime[ar;	gs->prev	= - 1A3 pid] =								probe
	25	}	.		di accipi		ed:sched_ ed:sched					probe
	26					JTHC.SCH	eu.scheu_	proce				probe
	27	sns = @qtimes	e[args->r	next_pid];							
	28 29	if (\$ns) { @usecs =	hist((nse	acs _ ≮n	s) / 100	a)•						
	30	}		.cs - pii	13) / 1000	.,,						
	31	delete(@qti	me[args->	•next_pi	.d]);							
	32	}										
	33 24											
	34 35	END {										
	Legend	legend format		0	Format	Heatm	nap 🝷	URL	override l	JRL		
)								
11	۲. *		pacine Erec	المع ريدان	31011 2.0 Y	CULC ETC	chac y					
11		Sep-2018 Brendar	n Gregg Cre	eated thi								
13												
14 15	profil	e:hz:99										
16	/arg/											
17 18	argN	= lhist(cpu, 0,	1000 1).				varia	able	argN			
19	}		1000, 1/,						Argument to t	he function being trac	red (arg0, arg1,	etc.)
												,
Legend		d format	6 For	mat H	leatmap	URL		RL				
E												
10	* 13-	Sep-2018 Brenda	an Gregg C	reated t	his.							
11												
12 13	// 1nc	lude: @usecs										
14	kprobe	:blk_account_io_	start									
15	{	nt[ang0] = =										
16 17	@sta }	rt[arg0] = nsecs	<u>ز</u>									
18												
19		:blk_account_io_	done									
20 21	-/@star {	t[arg0]/										
22		cs = lhi										
23 24		te <mark>(@</mark> lhist()							function	lhist(int n, int mi	n, int max, int	step)
24 25	}									Produce a linear hi	stogram of value	es of B
26	END									Froduce a linear fil		
27 28	{ clea	r(@start);										
28	}											
Legend			6) F	ormat	Heatmap	- URL		e URL				

bpftrace flame graphs

€ PCP bpftrace Flame Graphs -		
	Kernel Stacks	
	12:03:15-12:05:06 Reset zoom Q. Search	
	12:03:15 - 12:05:06 Reset zoom Q Search	
entry_SYSCALL		
do_syscall_64+95		
x64_sys_sendt sys_sendto+238		
sock_sendmsg+87		
tcp_sendmsg+40		
tcp_sendmsg_loc tcp_push_pend		
tcp_write_xmit+		
ip queue xmit		
ip_output+113		
ip finish output		
do_softirq.part.0		
do softirq own		
do_softirq+238ret_from_f		
entry process backlog worker th		
do sy netif receive process o	i de la companya de l	
do wr ip rcv+188 delayed f vfs w ip local deliver fput+272 entry SY		
vfs w ip_local_deliver entry+272 entry_SY entry S do ite ip_local_deliver ent dentry do syscal	entry entry SYS entry SYSCAL ent do sy do syscall do syscall 64 do	
do_sysc do_ite ip_protocol_deli do evict+203 _x64_sys	x64 x64 sys x64 sys.epol x secondary startup 64+	
ksys_w sock tcp_v4_rcv+2870x shmem_ev ksys_ioctl	do fu do epol wait do s i start secondary 355	
	SYSCALL 64 after hwframe+68 fute ep poll+1 ep poll+1044 poll cpu startup entry+25 scall 64+95 fute schedule hr schedule hr schedule hr do idle+351 idle idle idle	
	write+95 schedule-57 schedule_idle+40 sc	
	rite+182 sched text start+679 entry S.	
root root root root root root root root	fd write+191 fnish_task_switch+123 do sysc.	·

2.3.3 Metric Search

්	l ^e	Performance Co-Pilot Full-Text Metric Search		
+	Q disi		Search	日 Back To Bookmarks & Latest Searches
88		🔨 ics 🗹 Instances 🗹 Instance Domains	Search	
Ø	56 resul		Elapsed: 0.000758s	No bookmarks saved.
¢				
(Arr	disk	c.all.read		Search History:
8	total	read operations, summed for all disks		
ø	<i>⊳</i> м	1etric		
¢		c.all.write write operations, summed for all disks letric		
		Call.blkread read operations, summed for all disks Hetrie		
8	disk	c.all.blkwrite		
?	block	write operations, summed for all disks Ietric	→ Read More	

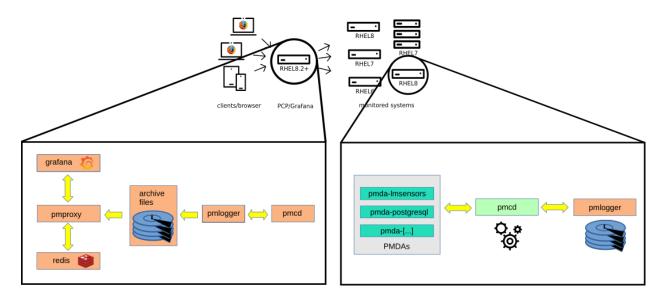


Fig. 1: © Christian Horn

2.4 Architecture

2.4.1 Monitored Hosts

Monitored hosts run the **Performance Metrics Collector Daemon (PMCD)**, which communicates with one or many **Performance Metrics Domain Agents (PMDAs)** on the same host. Each **PMDA** is responsible for gathering metrics of one specific domain - e.g., the kernel, services (e.g., PostgreSQL), or other instrumented applications. The **pmlogger** daemon records metrics from **pmcd** and stores them in archive files on the hard drive.

Since **PCP 5** metrics can also be stored in the redis database, which allows multi-host performance analysis, the **pmproxy** daemon discovers new archives (created by **pmlogger**) and stores them in a redis database.

2.4.2 Dashboards

Performance Co-Pilot metrics can be analyzed with Grafana dashboards, using the **grafana-pcp** plugin. There are two modes available:

- historical metrics across multiple hosts using the PCP Redis datasource
- live, on-host metrics using the PCP Vector datasource

The *PCP Redis* datasource sends *pmseries* queries to **pmproxy**, which in turn queries the redis database for metrics. The *PCP Vector* datasource connects to **pmproxy**, which in turn requests live metrics directly from a local or remote **PMCD**. In this case, metrics are stored temporarily in the browser, and metric values are lost when the browser tab is refreshed. The *PCP Redis datasource* is required for persistence.

2.5 Change Log

2.5.1 3.2.1 (2021-11-24)

• **dashboards**: add note about incompatibility of checklist dashboards with Grafana v8

• search: fix metric search form to make it compatible with Grafana v8

2.5.2 3.2.0 (2021-11-11)

- dashboards: new MS SQL server dashboard for PCP Redis
- dashboards: do not hide empty buckets in PCP Vector eBPF/BCC Overview dashboard
- dashboards: set revision for all dashboards
- redis: utilize query.options settings, same as PCP Vector
- redis: fix metric() function to return all metric names if no parameter is specified
- vector: perform rate conversion only if it's enabled in the query options (it is by default)
- **build**: add workaround to replace deprecated md4 hash algorithm with sha256 during build (md4 is unavailable in OpenSSL 3.0)
- build: update Node.js and Go dependencies, and grafonnet
- build: double-zip build artifacts in the CI workflow to preserve permissions (see actions/upload-artifact#38)
- **build**: add zip Makefile target, run grafana/plugincheck in CI workflow
- docs: add PCP Vector eBPF/BCC Overview dashboard screenshots

2.5.3 3.1.0 (2021-06-25)

- checklist: use new GraphNG component, show units in graphs, update help texts
- all: ensure Grafana 8.0 compatibility by replacing Angular.js based plugin config component with React
- **dashboards**: add pmproxy URL and hostspec variables to PCP Vector Host Overview and PCP checklist dashboards
- dashboards: show datasource field on all dashboards
- dashboards: mark all dashboards as readonly
- **bpftrace**: fix bpftrace error messages (don't append errors indefinitely)
- vector, bpftrace: use pcp://127.0.0.1 as default hostspec (no functional change)
- chore: update dependencies
- test: replace convey with testify for the Go tests

2.5.4 3.0.3 (2021-02-24)

- test: fix e2e tests by using another CSS selector
- chore: update dependencies
- docs: add container guide and screenshot

2.5.5 3.0.2 (2021-01-22)

• checklist: replace the storage metrics disk.dm. * with disk.dev. * (enables usage without device mapper)

2.5.6 3.0.1 (2020-12-22)

Enhancements / Bug Fixes

- redis: add auto-completions for new pmseries(1) language functions
- redis, vector: show error messages returned by the REST API
- vector, bpftrace: fix error messages regarding missing metrics
- vector: register derived metrics for every context
- vector: handle missing metric metadata responses
- checklist: fix metric name in storage warning dialog
- test: fix PCP Redis datasource test on 32bit architectures
- **build**: update dependencies

2.5.7 3.0.0 (2020-11-23)

Highlights of v3.0

- redis: support for Grafana Alerting
- redis: full-text search in metric names, descriptions, instances
- **vector**: support derived metrics, which allows the usage of arithmetic operators and statistical functions inside a query (pmRegisterDerived(3))
- vector: configurable hostspec (access remote PMCDs through a central pmproxy)
- vector: automatically configure the unit of the panel
- **dashboards**: detect potential performance issues and show possible solutions with the checklist dashboards, using the USE method
- dashboards: new MS SQL server dashboard (Louis Imershein)
- dashboards: new eBPF/BCC dashboard
- dashboards: new container overview dashboard with CGroups v2

Breaking Changes in v3.0

- **dashboards**: All dashboards are now located in the *Dashboards* tab at the datasource settings pages and are not imported automatically
- redis: Using label_values (metric, label) in a Grafana variable query is deprecated due to performance reasons. label_values (label) is still supported.

New Features

- redis: added instance.name and dashboard variables support in query editor
- redis: heatmap support
- dashboards: updated PCP Redis Metric Preview dashboards: added metric drop-down
- dashboards: added MS SQL server dashboard for Vector (Louis Imershein)

• chore: sign plugin

Enhancements / Bug Fixes

- redis: implement workaround if two values for the same instance and timestamp are received
- redis: send one instance labels request instead of one per instance
- redis: refresh instances only once per series
- redis: improved error messages
- vector: (internal) option to disable time utilization conversion
- vector: show error message when access mode is set to server & url override is set
- vector: disable redis backfill for now (pmseries and pmapi instance id's don't match)
- bpftrace: interpret all fields of CSV output as strings
- **dashboards**: moved dashboards to the datasource level: dashboards of interest can be imported using the dashboards tab of each datasource settings page
- dashboards: fix KB/s unit in dashboards, should be KiB/s
- dashboards: add installation instructions to BCC and bpftrace dashboards
- dashboards: update titles and add units to checklist dashboards
- search: fix datasource detection
- search: propagate error messages to the user
- poller: use timeout instead of interval to prevent overlapping timers
- **poller**: deregister targets immediately if endpoint changed
- chore: update build dependencies
- **test**: add unit tests to all datasources
- test: add End-to-End tests
- docs: update authentication guide to use scram-sha-256

2.5.8 3.0.0-beta1 (2020-10-12)

New Features

- redis: support for Grafana Alerting
- redis: full-text search in metric names, descriptions, instances
- vector: support derived metrics, which allows the usage of arithmetic operators and statistical functions inside a query, see pmRegisterDerived(3)
- **vector**: set background metric poll interval according to current dashboard refresh interval, do not stop polling while in background
- vector: automatically configure the unit of the panel
- vector: redis backfilling: if redis is available, initialize the graph with historical data
- vector: configurable hostspec (access remote PMCDs through a central pmproxy)

- vector: access context, metric, instancedomain and instance labels
- **dashboards**: checklist dashboard: detects potential performance issues and shows possible solutions to resolve them
- dashboards: eBPF/BCC dashboard
- dashboards: container overview dashboard with CGroups v2

Enhancements / Bug Fixes

- build: convert dashboards to jsonnet/grafonnet
- all: use latest Grafana UI components based on React (Grafana previously used Angular)

Redis datasource installation

Unfortunately it is not possible to sign community plugins at the moment. Therefore the PCP Redis datasource plugin needs to be allowed explicitly in the Grafana configuration file:

allow_loading_unsigned_plugins = pcp-redis-datasource

Restart Grafana server, and check the logs if the plugin loaded successfully.

Deprecated features

• redis: Using label_values (metric, label) in a Grafana variable query is deprecated due to performance reasons. label_values (label) is still supported.

2.5.9 2.0.2 (2020-02-25)

• vector, redis: remove autocompletion cache (PCP metrics can be added and removed dynamically)

2.5.10 2.0.1 (2020-02-17)

• build: fix production build (implement workaround for systemjs/systemjs#2117, grafana/grafana#21785)

2.5.11 2.0.0 (2020-02-17)

- **vector, bpftrace**: fix version checks on dashboard load (prevent multiple pmcd.version checks on dashboard load)
- vector, bpftrace: change datasource check box to red if URL is inaccessible
- redis: add tests
- flame graphs: support multidimensional eBPF maps (required to display e.g. the process name)
- **dashboards**: remove BCC metrics from Vector host overview (because the BCC PMDA isn't installed by default)
- misc: update dependencies

2.5.12 2.0.0-beta1 (2019-12-12)

• support Grafana 6.5+, drop support for Grafana < 6.5

2.5.13 1.0.7 (2020-01-29)

• redis: fix timespec (fixes empty graphs for large time ranges)

2.5.14 1.0.6 (2020-01-07)

- redis: support wildcards in metric names (e.g. disk.dev.*)
- redis: fix label support
- redis: fix legends

2.5.15 1.0.5 (2019-12-16)

- **redis:** set default sample interval to 60s (fixes empty graph borders)
- **build:** upgrade copy-webpack-plugin to mitigate XSS vulnerability in the serialize-javascript transitive dependency
- build: remove deprecated uglify-webpack-plugin

2.5.16 1.0.4 (2019-12-11)

Enhancements

- flame graphs: clean flame graph stacks every 5s (reduces CPU load)
- general: implement PCP version checks

Bug Fixes

- **build:** remove weak dependency (doesn't work with Node.js 12)
- build: upgrade terser-webpack-plugin to mitigate XSS vulnerability in the serialize-javascript transitive dependency

2.5.17 1.0.3 (2019-11-22)

• fix flame graph dependency (flamegraph.destroy error in javascript console)

2.5.18 1.0.2 (2019-11-12)

- handle counter wraps (overflows)
- convert time based counters to time utilization

2.5.19 1.0.1 (2019-10-24)

Flame Graphs

- aggregate stack counts by selected time range in the Grafana UI
- add an option to hide idle stacks

Vector

- fix container dropdown in the query editor
- remove container setting from the datasource settings page

Redis

• fix value transformations (e.g., rate conversion of counters)

All

• request more datapoints from the datasource to fill the borders of the graph panel

2.5.20 1.0.0 (2019-10-11)

bpftrace

- support for Flame Graphs
- context-sensitive auto-completion for bpftrace probes, builtin variables, and functions incl. help texts
- parse the output of bpftrace scripts (e.g., using printf()) as CSV and display it in the Grafana table panel
- sample dashboards (BPFtrace System Analysis, BPFtrace Flame Graphs)

Vector

- table output: show instance name in the left column
- table output: support non-matching instance names (cells of metrics which don't have the specific instance will be blank)

Vector & bpftrace

- if the metric/script gets changed in the query editor, immediately stop polling the old metric/deregister the old script
- improve pmwebd compatibility

miscellaneous

- help texts for all datasources (visible with the [?] button in the query editor)
- renamed PCP Live to PCP Vector
- logos for all datasources
- improved error handling

2.5.21 0.0.7 (2019-08-16)

• The initial release of grafana-pcp

Features

- retrieval of Performance Co-Pilot metrics from pmseries (PCP Redis), pmproxy, and pmwebd (PCP Live)
- automatic rate conversion of counter metrics
- auto-completion of metric names 1,2, qualifier keys, and values 2
- display of semantics, units, and help texts of metrics 1
- legend templating support with \$metric, \$metric0, \$instance, \$some_label
- container support
- support for repeating panels
- support for custom endpoint URL and container setting per query, with templating support 1
- heatmap and table support 1
- sample dashboards for PCP Redis and PCP Live

1 PCP Live 2 PCP Redis

Known Bugs

• the bpftrace datasource is work-in-progress and will be ready with the next release (approx. 1-2 weeks)

Thanks to Jason Koch for the initial pcp-live datasource implementation and the host overview dashboard.

2.6 Overview

2.6.1 PCP Redis

This data source queries the fast, scalable time series capabilities provided by the pmseries functionality. It is intended to query **historical** data across **multiple hosts** and supports filtering based on labels.

2.6.2 PCP Vector

The PCP Vector data source shows **live**, **on-host metrics** from the real-time pmwebapi interfaces. It is intended for an individual host, on-demand performance monitoring, and includes container support.

2.6.3 PCP bpftrace

The PCP bpftrace data source supports system introspection using bpftrace scripts. It connects to the bpftrace PMDA and runs bpftrace scripts on the host.

2.7 Authentication

Performance Co-Pilot supports the following authentication mechanisms through the SASL authentication framework: plain, login, digest-md5, scram-sha-256 and gssapi. This guide shows how to setup authentication using the scram-sha-256 authentication mechanism and a local user database.

Note: Authentication methods login, digest-md5 and scram-sha-256 require PCP 5.1.0 or later.

2.7.1 Requisites

Install the following package, which provides support for the scram-sha-256 authentication method:

Fedora/CentOS/RHEL

```
$ sudo dnf install -y cyrus-sasl-scram
```

Debian/Ubuntu

```
$ sudo apt-get install -y libsasl2-modules-gssapi-mit
```

2.7.2 Configuring PMCD

First, open the /etc/sasl2/pmcd.conf file and specify the supported authentication mechanism and the path to the user database:

mech_list: scram-sha-256
sasldb_path: /etc/pcp/passwd.db

Then create a new unix user (in this example poptestuser) and add it to the user database:

```
$ sudo useradd -r pcptestuser
$ sudo saslpasswd2 -a pmcd pcptestuser
```

Note: For every user in the user database, a unix user with the same name must exist. The passwords of the unix user and the /etc/pcp/passwd.db database are not synchronized, and (only) the password of the saslpasswd2 command is used for authentication.

Make sure that the permissions of the user database are correct (readable only by root and the pcp user):

```
$ sudo chown root:pcp /etc/pcp/passwd.db
$ sudo chmod 640 /etc/pcp/passwd.db
```

Finally, restart pmcd and pmproxy:

```
$ sudo systemctl restart pmcd pmproxy
```

2.7.3 Test Authentication

To test if the authentication is set up correctly, execute the following command:

```
$ pminfo -f -h "pcp://127.0.0.1?username=pcptestuser" disk.dev.read
```

2.7.4 Configuring the Grafana Datasource

Go to the Grafana datasource settings, enable **Basic auth**, and enter the username and password. Click the *Save & Test* button to check if the authentication is working.

Note: Due to security reasons, the access mode Browser is not supported with authentication.

2.8 PCP Redis

2.8.1 Introduction

This data source provides a native interface between Grafana and Performance Co-Pilot (PCP), allowing PCP metric data to be presented in Grafana panels, such as graphs, tables, heatmaps, etc. Under the hood, the data source makes REST API query requests to the PCP pmproxy service, which can be running either locally or on a remote host. The pmproxy daemon can be local or remote and uses the Redis time-series database (local or remote) for persistent storage.

2.8.2 Setup Redis and PCP daemons

```
$ sudo dnf install redis
$ sudo systemctl start redis pmlogger pmproxy
```

2.8.3 Query Language

Syntax: [metric.name] '{metadata qualifiers}'

Examples:

```
kernel.all.load
kernel.all.load{hostname == "web01"}
network.interface.in.bytes{agent == "linux"}
```

Documentation of the pmseries query language can be found in the man page of pmseries.

2.8.4 Query Formats

Time Series

Returns the data as time series. If there are multiple series for a metric, all series will be shown as separate targets (i.e., a line in a line graph). For metrics with instance domains, each instance is shown as a separate target. If there are multiple queries defined, all values will be combined in the same graph.

Table

Transforms the data for the table panel. Two or more queries are required, and it will transform every metric into a column, and every instance into a row. The latest values of the currently selected timeframe will be displayed.

2.8.5 Legend Format Templating

Variable	Description	Example
\$expr	query expression	rate(disk.dm.avactive)
\$metric	metric name	disk.dev.read
\$metric0	last part of metric name	read
\$instance	instance name	sda
\$some_label	label value	anything

The following variables can be used in the legend format box:

2.8.6 Query Functions

The following functions are available for dashboard variables of type Query:

Function	Description	Example
metrics([pattern]returns all metrics matching a glob pattern (if no pattern is de-	metrics(disk.
	fined, all metrics are returned)	*)
label_names([pat	treturns) all label names matching a glob pattern (if no pattern is	label_names(host*)
	defined, all metrics are returned)	
label_values(lab	ereturns all label values for the specified label	label_values(hostna

2.9 PCP Vector

2.9.1 Query Formats

Time Series

Returns the data as time series. For metrics with instance domains, each instance is shown as a separate target (i.e., line in a line graph). If there are multiple queries defined, all values will be combined in the same graph.

Heatmap

Transforms the data for the heatmap panel. Instance names have to be in the following format: <lower_bound>-<upper_bound>, for example, 512-1023 (the bcc PMDA produces histograms in this format).

The following settings have to be set in the heatmap panel options:

Setting	Value
Format	Time Series Buckets
Bucket bound	Upper

Table

Transforms the data for the table panel. Two or more queries are required, and it will transform every metric into a column, and every instance into a row. The latest values of the currently selected timeframe will be displayed.

2.9.2 Legend Format Templating

The following variables can be used in the legend format box:

Variable	Description	Example
\$expr	query expression	rate(disk.dm.avactive)
\$metric	metric name	disk.dev.read
\$metric0	last part of metric name	read
\$instance	instance name	sda
\$some_label	label value	anything

2.10 PCP bpftrace

2.10.1 bpftrace PMDA installation

```
$ sudo dnf install pcp-pmda-bpftrace
$ cd /var/lib/pcp/pmdas/bpftrace
```

\$ sudo ./Install

2.10.2 Query Formats

Time Series

Shows bpftrace variables as time series. For bpftrace maps, each key is shown as a separate target (i.e. line in a line graph), for example @counts[comm] = count(). If there are multiple variables (or scripts) defined, all values will be combined in the same graph.

Heatmap

Transforms bpftrace histograms into heatmaps.

The following settings have to be set in the heatmap panel options:

Setting	Value
Format	Time Series Buckets
Bucket bound	Upper

Table

Transforms CSV output of bpftrace scripts into a table. The first line must be the column names.

2.10.3 Legend Format Templating

The following variables can be used in the legend format box:

Variable	Description
\$metric0	bpftrace variable name
\$instance	bpftrace map key

2.10.4 More Information

bpftrace PMDA README

2.11 Multiple Vector Hosts

In cloud environments, it is often desired to use the Vector datasource to connect to multiple remote hosts without configuring a new data source for each host. This guide shows a setup for this use case using Grafana templates.

2.11.1 Setup the Vector data source

Open the Grafana configuration, go to Data Sources, and add the *PCP Vector* datasource. Leave the URL field empty and select **Access: Browser**. Click the save button. A red alert will appear, with the text *To use this data source, please configure the URL in the query editor.*

2.11.2 Create a new dashboard variable

Create a new dashboard (plus icon in the left navigation - *Create - Dashboard*) and open the dashboard settings (wheel icon on the right, top navigation bar). Navigate to *Variables* and create a new variable with the following settings:

Setting	Value
Name	host
Туре	Text box

Leave the other fields to their default values. Save the new variable, go back to the dashboard, enter a hostname (for example, localhost) in the text box, and press enter.

2.11.3 Create a new graph

Add a new graph to the dashboard, select the *PCP Vector* datasource, enter a PCP metric name (for example disk. dev.read_bytes) in the big textbox, and enter http://\$host:44322 in the URL field. If you haven't already, select the time range to *last 5 minutes* and select the auto-refresh interval (top right corner) to 5 seconds, for example.

Now Grafana connects to http://localhost:44322 for this panel (if you have entered localhost in the host textbox). By changing the value of the host text box, you can change the remote host.

2.11.4 Setting the host by query parameter

You can also set the host by an URL query parameter. Add &var-host=example.com to the current query, or update the var-host query parameter in case it is already present in the current query string.

2.12 Monitoring Containers

2.12.1 Importing the dashboards

grafana-pcp includes the following (optional) dashboards:

- PCP Vector: Container Overview (CGroups v1)
- PCP Vector: Container Overview (CGroups v2)

You can import the corresponding dashboard on the PCP Vector datasource settings page.

Note: grafana-pcp before version 3.0.0 includes a single dashboard called **PCP Vector: Container Overview** which supports CGroups v1 only and is installed by default (i.e. no import is required).

2.12.2 Usage

You can choose one or multiple containers in the *container* drop-down field at the top of the dashboard:



2.12.3 Common Problems

My container doesn't show up

- make sure that the docker and/or podman PMDAs are installed
- currently PCP only supports containers started by the root user (there is an open feature request to change this)

2.13 Troubleshooting

2.13.1 Common Problems

HTTP Error 502: Bad Gateway, please check the datasource and pmproxy settings

When I try to add a datasource in Grafana, I get the following error: "HTTP Error 502: Bad Gateway, please check the datasource and pmproxy settings. To use this data source, please configure the URL in the query editor."

- check if pmproxy is running: systemctl status pmproxy
- make sure that pmproxy was built with time-series (libuv) support enabled. You can verify that by reading the logfile in /var/log/pcp/pmproxy.log

2.13.2 PCP Redis

Grafana doesn't show any data

• Make sure that pmlogger is up and running, and writing archives to the disk (/var/log/pcp/pmlogger/ <host>/*)

- Verify that pmproxy is running, time series support is enabled and a connection to Redis is established: check the logfile at /var/log/pcp/pmproxy/pmproxy.log and make sure that it contains the following text: Info: Redis slots, command keys, schema version setup
- Check if the Redis database contains any keys: redis-cli dbsize
- Check if any PCP metrics are in the Redis database: pmseries disk.dev.read
- Check if PCP metric values are in the Redis database: pmseries 'disk.dev.read[count:10]'
- Check the Grafana logs: journalctl -e -u grafana-server