grafana-pcp Documentation

Release 3.0.3

Performance Co-Pilot

Feb 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		13
	2.5	Change Log	13
	2.6	Overview	19
	2.7	Authentication	20
	2.8	PCP Redis	21
	2.9	PCP Vector	22
	2.10	PCP bpftrace	23
	2.11	Multiple Vector Hosts	24
	2.12	Monitoring Containers	25
		Troubleshooting	

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 (Fedora)

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

For other distributions, please refer to the Installation Guide.

After Grafana and grafana-pcp are installed, you can enable the plugin: Open the Grafana configuration, go to Plugins, select *Performance Co-Pilot*, and click the *Enable* button.

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 setting of 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 not 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/

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

>pcp-app" -p 3000:3000 grafana/grafana
```

```
$ docker 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 grafana/grafana
```

2.2.4 From Source

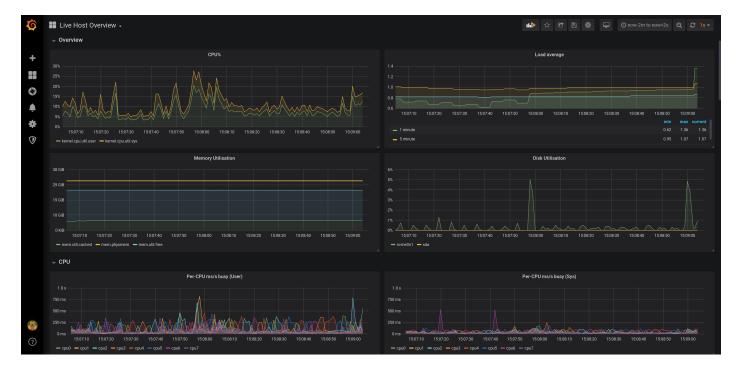
The yarn package manager, Go compiler, jsonnet and jsonnet bundler are required to build grafana-pcp.

```
$ git clone https://github.com/performancecopilot/grafana-pcp.git
$ make dist
$ 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





~ A		
disk.dev.read	1	
	<pre> disk.dev.read disk.dev.read bytes </pre>	disk.dev.read_bytes ×
Legend 🔅	<pre> disk.dev.read_merge disk.dev.read_rawactive </pre>	Type: <i>u64</i> Semantics: <i>counter</i>
	<pre> ø disk.dev.blkread </pre>	Units: <i>Kbyte</i> per-disk count of bytes read
+ Query		

2.3.2 PCP bpftrace



	b pftrace	System A	nalysis +									⊪∲ ☆		* 🖵		
	√ Disk															
H								Blo	ck I/O latency							
	33 ms — 16 ms —															
3	8 ms — 4 ms —															
	2 ms															
÷	511 μs — 255 μs —															
	127 μs															
D	31 μs — 15 μs —															
	20:05:20	0 20:05:25	i 20:05:30 20:05	i:35 20:05:40 20:05:45 :	20:05:50 20:0	5:55 20:06:00	20:06:05 2	0:06:10 20:06	:15 20:06:20 20:06	5:25 20:06	30 20:06:35 20:06:40	20:06:45 20:06:5	0 20:06:55	20:07:00	20:07:05 20:07:10	20:07:15
	✓ Filesystem	ı														
	40K cos							virtua	file system calls							
	4UK OPS															
	0 ops 20:0			0:05:35 20:05:40 20:05:45		20:05:55 20:06:00					06:30 20:06:35 20:06:40	20:06:45 20:06		20:07:00	20:07:05 20:07:10	20:07:15
		ige — vfs_get	attr — vfs_getattr_nose			eadlink — vfs_rename	e — vfs_statfs		fs_statx_fd — vfs_unlink		vfs_writev — vfs_fadvise — v	vfs_mkdir 🗕 vfs_symlini	k — vfs_fallocate			
	Network bpftrace ops											thi ≹ ☆			now-2m to now+2	
	bpftrace o ups 20:1 - vfs_fsync_ran - vfs_setxattr	17:05 20:1 ige — vfs_get	7:10 20:17:15 2 attr — vfs_getattr_nose	101720 201725 201730 c ─_Ys_lock_file ─_yfs_open ─_ _create_mount ─_yfs_get_super ─			e — vfs_statfs		fs_statx_fd 🗕 vfs_unlink		:18:15 20:18:20 20:18:23 vfs_writev — vfs_fadvise — v		18:35 20:18:40) 20:18:45	20:18:50 20:18:55	
2	bpftrace 0 tps 20:1 — vfs_fsync_ran	17:05 20:1 ige — vfs_get	7:10 20:17:15 2 attr — vfs_getattr_nose			eadlink — vfs_rename	e — vfs_statfs		fs_statx_fd 🗕 vfs_unlink				18:35 20:18:40) 20:18:45	20:18:50 20:18:55	
•	bpftrace o ups 20:1 - vfs_fsync_ran - vfs_setxattr	17:05 20:1 ige — vfs_get	7:10 20:17:15 2 attr — vfs_getattr_nose		vfs_read — vfs_r — vfs_get_tree —	eadlink — vfs_rename	e — vfs_statfs		fs_statx_fd 🗕 vfs_unlink				8:35 20:18:40 k — vfs_fallocate) 20:18:45	20:18:50 20:18:55	
+	<pre>bpftrace uppa20:1 - vfa_fsync_ran - vfa_setxattr - Network TIME •</pre>	17:05 20:1 ige – vfs_get – vfs_mknod PID	7:10 20:17:15 2 attr — vfs_getattr_nose	c — vfs_lock_file — vfs_open — _create_mount — vfs_get_super — trace TCP acce RADDR	vfs_read — vfs_r - vfs_get_tree — pt() RPORT	eadink — vfs_renama vfs_parse_fs_param — LADDR	e — vfs_statfs vfs_parse_fs_s LPORT	- vfs_statx - v tring - vfs_remov BL	fs_statx_fd — vfs_unlink rexattr — vfs_link TIME ~		vfs_writev — vfs_fadvise — v	5 20:18:30 20:1 vfs_mkdir — vfs_symin trace TCP co SADDR	8:35 20:18:40 k — vfs_fallocate) 20:18:45 - vfs_mdir SPORT	20:18:50 20:18:55 - vfs.getxattr - vfs.ge DADDR	5 20:19:00 ebcattr_alloc DPORT
	■ bpftrace □ vipa20:1 □ vi5_fsync_ran □ vi5_sexattr ▼ Network TIME ▼ 20:17:55	17:05 20:1 ige — vfs_get — vfs_mknod PID 3888	7:10 20:17:15 2 attr — vfs_getattr_nose — vfs_test_lock — vfs COMM prmcd	c – vfs.jock_file – vfs.gept – Loreate_mount – vfs.get_super – trace TCP acce RADDR 127.0.0.1	vfs_read — vfs_r - vfs_get_tree — pt() RPORT 38034	eadink — vfs.rename vfs.parse_fs.param = LADDR 127.0.0.1	e vfs_statfs vfs_parse_fs_s 	vfs_statx v ttring vfs_remov BL 0/5	fs_statx_fd — vfs_unlink rexattr — vfs_link TIME ~ 20:18:53		vfs_writev — vfs_fadvise — v COMM curl	5 20:18:30 20:1 yfs_mkdir — vfs_symin trace TCP co SADDR 192:168.0.80	8:35 20:18:40 k — vfs_fallocate	20:18:45 - vfs_mdlr SPORT 33536	20.18:50 20.18:55 - vfs_getxattr - vfs_ge DADDR 152.199.19.160	5 20:19:00 etxattr_alloc DPORT 443
	 bpftrace oup vfa.fsync.ina vfa.fsync.ina vfa.sexattr Network TIME • 20:17:55 20:17:55 	17:05 20:1 ige vfs_get vfs_mknod PID 3888 3888	2710 2017:15 4 attr = vfs_getattr_nose = vfs_test_lock = vfs COMM pmcd pmcd	c	vfs.read — vfs.r - vfs.get_tree — pt() RPORT 38034 38032	eadink — vfs_rename vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1	e vfs_statfs vfs_parse_fs_s LPORT 44321 44321	Urfs_statx v v string v fs_remov	fs.statx_fd → vfs.unlink exattr → vfs_link TIME → 20:18:53 20:18:50		vfs,writev – vfs_fadvise – v COMM curl wget	5 2018-30 20:1 4fs_mådir — vfs_symin trace TCP oc SADDR 192.168.0.80 192.168.0.80	8:35 20:18:40 k — vfs_fallocate	20:18:45 - vfs_mdir SPORT 33536 59034	20:18:50 20:18:55 vfs.getxattr - vfs.ge DADDR 152:199.19.160 192:30.253.120	20:19:00 etxattr_alloc DPORT 443 443
	■ bpftrace □ • vf15yer − vf15y	17:05 20:1 gg → vfs_ger → vfs_mknod PID 3888 3888 3888	7:10 2017:15 2 attr = vfs.getatts_nose = vfs.getatts_nose = vfs.getatts_nose = vfs.getatt_nose = vfs.getatt_nose pmcd pmcd pmcd	c = vfs,bock,fie = vfs,open = constre_mount = vfs_open = trace TCP acce RADDR 127.0.0.1 127.0.0.1 127.0.0.1	vfs.read — vfs.r - vfs.get_tree — pt()	eadlink — vfs_renamk vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1	e vfs_statfs vfs_parse_fs_s LPORT 44321 44321 44321	BL 0/5 0/5 0/5	fs_statx_fd → vfs_unlink exattr → vfs_link TIME → 20:18:53 20:18:50 20:18:50		-vfs_writevvfs_fadvisev COMM curl wget	5 2018:30 20: cfs,kdir — vfs,.ymin trace TCP cc SADDR 192.168.0.80 192.168.0.80	835 20:18:40 k → vfs_fallocate	20:18:45 • vfs_rmdir SPORT 33536 59034 59154	201850 201855 vfs.getxattr - vfs.ge DADDR 152.199.19.160 192.30.253.120 140.82.118.4	20.19-00 prbuttralloc DPORT 443 443 443
	 bpftrace via_faync_ran via_setxattr Network TIME = 20:17:55 20:17:55 20:17:55 20:17:55 20:17:55 	17:05 20:1 age • Vfs_gee • Vfs_mknod PID 3888 3888 3888 3888 3888	7:10 2017:15 2 attr = vfs.getattr.nose = vfs.getattr.nose = vfs.getatt.nose = vfs.getatt.ook = vfs pmcd pmcd pmcd pmcd pmcd	c = vfs,lock,fie = vfs,open = _ccease.mount = vfs_open = trace TCP acces PADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	vfs_read — vfs_r vfs_get_tree — pt()	eadink — vfs_rename vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1 127.0.0.1	e vfs_statfs vfs_parse_fs_s LPORT 44321 44321 44321 44321	BL 0/5 0/5 0/5 0/5 0/5	fs_stats_fd vfs_unlick exattr vfs_lick TIME 20:18:53 20:18:53 20:18:50 20:18:50 20:18:10		-vfs,writev vfs,fadvise v COMM curl wrget NetworkManager	5 2018:30 20: cfs.,nkdir — vfs.,symin trace TCP cc SADDR 192.168.0.80 192.168.0.80 192.268.0.80 2002.8388.880;3::	835 20:18:40 k → vfs_fallocate	20:18:45 • vfs.mdir SPORT 33536 59034 59154 50088	201850 201858 vfs.getxattr - vfs.get DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178:5000:	5 201900 tributtr,alloc 443 443 443 80
	bpftrace ogs ogs ortigspin,cran vit_spin,cran vit_senatit	2011 ge vfs_ge vfs_mknod PID 3988 3888 3888 3888 3888	710 201715 2 attr = vfs_getattr_nose = vfs_test_lock = vfs COMM pmcd pmcd pmcd pmcd pmcd pmcd pmcd	c = vfs,lock,fie = vfs,open = .cceater.mount = vfs,open = trace TCP acces PADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0 127.0.0 127.0.0	vfs_read — vfs_r vfs_get_tree — pt() RPORT 38034 38032 38030 38028 38026	eadink — vfs_rename vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	e vfs_statfs vfs_parse_fs_s LPORT 44321 44321 44321 44321	Ufs_statx V bl 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5	fs_stato_(d → vfs_unlisk weattr → vfs_leik TIME → 2018:53 2018:50 2018:50 2018:10		COMM curl wget NetworkManager NetworkManager	5 2018.30 201 5 2018.30 201 trace TCP oc SADDR 192.168.0.80 192.168.0.80 202.6388.8803:: 192.168.0.80	835 20:18:40 k → vfs_fallocate	20:18:45 - vfs.mdir SPORT 33536 59034 59154 50088 47540	201850 201856 vfs.getxattr - vfs.get DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198	5 2011900 tribtattr,alloc 443 443 443 80 80 80
+	bpftrace 201 vit_fore_ran vit_fore_ran vit_fore_ran vit_setuit Network TIME * 2017:55 2017:55 2017:55 2017:55 2017:55 2017:55 2017:55	17:05 20:11 ige	7:10 2017:15 2 attr = vfs.getattr.nose = vfs.getattr.nose = vfs.getatt.nose = vfs.ge	c. – 41, bbk fie – Vfa.gen, eigen creater, mourt – Vfa.gen, seper – trace TCP acce 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	vfs_read — vfs_r vfs_get_tree — pt() RPORT 38034 38032 38030 38028 38026 38026	eadirk	 vfs_statfs vfs_parse_fs_s vfs_parse_fs_s	Ufa_statx V tring Vfa_remov 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5	fs_state_di → fs_unitek exattr → vfs_iek 20:18:53 20:18:50 20:18:10 20:18:10 20:17:55		-vfs,writev vfs,fadvise v COMM curl wrget NetworkManager	5 2018.30 2017 5 2018.30 2017 trace TCP oc SADDR 192.168.0.80 192.168.0.80 202.8388.880€3:: 192.168.0.80 192.168.0.80 192.168.0.80	835 20:18:40 k → vfs_fallocate	20:18:45 - vfs,mdir 33536 59034 59154 50088 47540 38034	2018.50 2018.57 vfs.getxattr - vfs.get DADDR 152.199.19.100 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1	201900 5 201900 Withwattr_alloc 443 443 443 443 80 80 80 80 80 44321
	bpftrace ogs ogs ortigspin,cran vit_spin,cran vit_senatit	2011 ge vfs_ge vfs_mknod PID 3988 3888 3888 3888 3888	710 201715 2 attr = vfs_getattr_nose = vfs_test_lock = vfs COMM pmcd pmcd pmcd pmcd pmcd pmcd pmcd	c = vfs,lock,fie = vfs,open = .cceater.mount = vfs,open = trace TCP acces PADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0 127.0.0 127.0.0	vfs_read — vfs_r vfs_get_tree — pt() RPORT 38034 38032 38030 38028 38026	eadink — vfs_rename vfs_parse_fs_param = LADDR 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	e vfs_statfs vfs_parse_fs_s LPORT 44321 44321 44321 44321	Ufs_statx V bl 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5	fs_stato_(d → vfs_unlisk weattr → vfs_leik TIME → 2018:53 2018:50 2018:50 2018:10		COMM curl wget NetworkManager NetworkManager	5 2018.30 201 5 2018.30 201 trace TCP oc SADDR 192.168.0.80 192.168.0.80 202.6388.8803:: 192.168.0.80	835 20:18:40 k → vfs_fallocate	20:18:45 - vfs.mdir SPORT 33536 59034 59154 50088 47540	201850 201856 vfs.getxattr - vfs.get DADDR 152.199.19.160 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198	5 2011900 tribtattr,alloc 443 443 443 80 80 80
	 bpftrace uque vitufgnc.ran vitufgnc.ran vitufgnc.ran Network TIME * 2017.55 	20.1 age - vfs.pst vfs.mknod 3888 3888 3888 3888 3888 3888 3888 3888	710 201715 / 2 attr = vfs_getattr_nose = vfs_set_lock = vfs pmed pmed pmed pmed pmed pmed pmed pmed	c = 41,Jock feie = 41,open = - c.center.mourt = 41,opet accept	vfs_read — vfs_rf ~ fs_get_tee _ pt() RPORT 38034 38032 38030 38028 38026 38024 38022 28030 28022 28030 28022 28030 28022 28030 28022 28030 28022 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28030 28032 28032 28030 28032 280	eadink — vfs_rename vfs_parae_fs_param = 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	e vfs_statfs vfs_parse_fs_m 44321 44321 44321 44321 44321 44321 44321	- vfa_statz v tting vfa_senov BL 0/5 0/5 0/5 0/5 0/5 0/5 0/5 0/5	rs_statc.fd → rfs_turitek weattr → vfs_lenk 2018:53 2018:50 2018:50 2018:50 2018:10 2017:55 2017:55		COMM curl curl wget NetworkManager pmwebd pmwebd	5 2018.30 2017 5 2018.30 2017 trace TCP oc SADDR 192.168.0.80 192.168.0.80 202.8388.880€3:: 192.168.0.80 192.168.0.80 192.168.0.80	8:35 20:18-4 (k - vfs,falocate connect() a00:0	20.1845 • vfs.mdir 33536 59034 59154 50088 47540 38034 38032	2018.50 2018.57 vfs.getxattr - vfs.get DADDR 152.199.19.100 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1	5 201900 5 201900 PPORT 443 443 443 80 60 44321 44321
	 bpftrace 201 vit_free,ran vit_sectatif Vit_sectatif Network 	2011 age - vfs.pst vfs.mknod 2888 3888 3888 3888 3888 3888 3888 3888	710 201715 2 attr = vfs_getattr_nose = vfs_test_lock = vfs COMM pmcd p	c = 41,Jock feie = 411,open = - creater.mount = 412,gel.sept = - FADDR 127.0.0.1 127.0.1 127.0.	vfs_read - vfs_r vfs_get_tree - pet() 38034 38032 38030 38028 38026 38024 38022 38022 ps	eadint vfs_rename vfs_parae_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	e — vfs_statfs _ vfs_parse_fs_m LPORT 44321 44321 44321 44321 44321 44321 44321 8555555555555555555555555555555555555	Uris data - v Uris - vt remove BL 0/5 0/5 0/5 0/5 0/5 0/5 0/5 1/5 1/5 0/5 1/5	rs_statc./d → vfs_unlink rs_statc./d → vfs_unlink TIME + 20.18.53 20.18.50 20.18.50 20.18.10 20.17.55 20.17.55 TIME +		COMM curl wget wget NetworkManager priwebd priwebd LADDR:LPORT	s 2018.30 20:1 ffs_mkdr — vfs_symin trace TCP cc SADDR 192.168.0.80 193.168 1	835 2018-40cate where vfs_falocate where() a00:0 ransmits RADDR:RP0F	20.1845 • vfs_mdir SPORT 33536 59034 59034 59154 50088 47540 38034 38032	201850 201858 vfs.getxattr - vfs.get 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	5 201900 DPORT 443 443 443 80 80 44321 44321 44321 5TATE
	 bpftrace 201 vit_free,ran vit_sectatif Network TIME - 2017:85 2017:85<th>17:05 20:11 ge − yfs_get − yfs_mknod 9 190 3888 3888 3888 3888 3888 3888 3888 38</th><th>710 201715 2 attr — Y5. getattr.nose — Y5. Jest Jock — Yfs COMM ksoftirqd/7</th><th>e. – 41, back file – Vfa.gene – </th><th>vfs_read vfs_r vfs_get_tree pt() 80032 38034 38032 38030 38028 38026 38022 38022 38022 58022 58022 00b0ce4::</th><th>Eadlink</th><th>e — vfs_statfs _ vfs_parse_fs_s LPORT 44321 44321 44321 44321 44321 44321 8 8 8 8 8</th><th>■ vfs_state. == v = vfs_steeners 0/5 0/5 0/5 0/5 0/5 0/5 0/5 5/5 5</th><th>Fs.tatk.(d) → (fs.unlick executir → (fs.jick) TIME * 2018/53 2018/53 2018/50 2018/10 2018/10 2017/55 2017/55 TIME * 2015/03</th><th></th><th>vfs,unitev — vfs,fashke — v COMM curl wget NetworkManager priwebd priwebd LADDR:LPORT 2a02.8388.9aa7:a00.0.422</th><th>s 2018.30 2017 frs_mkdr — vfs_symin trace TCP cc SADDR 192.168.0.80</th><th>8:35 20:18:40 k = vfs_falocate mnect() a00:0 ransmits RADDR:RP0F 2:004:fa87:50</th><th>20:18:45 20:18:45 SPORT 33536 59034 59154 50088 47540 38034 38032 27 27</th><th>201850 201858 vfs.gotxattr - vfs.got DADDR 152.199.19.100 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.01 127.0.01</th><th>2019-00 2019-0</th>	17:05 20:11 ge − yfs_get − yfs_mknod 9 190 3888 3888 3888 3888 3888 3888 3888 38	710 201715 2 attr — Y5. getattr.nose — Y5. Jest Jock — Yfs COMM ksoftirqd/7	e. – 41, back file – Vfa.gene – 	vfs_read vfs_r vfs_get_tree pt() 80032 38034 38032 38030 38028 38026 38022 38022 38022 58022 58022 00b0ce4::	Eadlink	e — vfs_statfs _ vfs_parse_fs_s LPORT 44321 44321 44321 44321 44321 44321 8 8 8 8 8	■ vfs_state. == v = vfs_steeners 0/5 0/5 0/5 0/5 0/5 0/5 0/5 5/5 5	Fs.tatk.(d) → (fs.unlick executir → (fs.jick) TIME * 2018/53 2018/53 2018/50 2018/10 2018/10 2017/55 2017/55 TIME * 2015/03		vfs,unitev — vfs,fashke — v COMM curl wget NetworkManager priwebd priwebd LADDR:LPORT 2a02.8388.9aa7:a00.0.422	s 2018.30 2017 frs_mkdr — vfs_symin trace TCP cc SADDR 192.168.0.80	8:35 20:18:40 k = vfs_falocate mnect() a00:0 ransmits RADDR:RP0F 2:004:fa87:50	20:18:45 20:18:45 SPORT 33536 59034 59154 50088 47540 38034 38032 27 27	201850 201858 vfs.gotxattr - vfs.got DADDR 152.199.19.100 192.30.253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.01 127.0.01	2019-00 2019-0
	 bpftrace 201 vfLjmpc.nn Vf	17:05 20:11 ge → vfs_gel → vfs_mknod 9 PID 3 3888 3 3888 4 3 3888 3 3 3888 3 3 3888 3 3 3 3	710 201715 2 attr — Y5, getattr_noise — Y5, Let Jock — Y1 pmcd pmcd 4 pmcd 4 pm	e. – 41, bek fie – vfa.gen, and created consumer – vfa.gen, and trace TCP accel RADOR 127.0.0.1 127.0.0 127.	vit_read → vit_r vit_get.tread → vit_r pt() 38034 38033 38036 38026 38026 38026 38022 38022 DADR:DI DADR:DI 0.0:bce4::	eadlink → vfs_parae. [s_param - vfs_parae. [s_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 4253:5441:58544	e — vfs.statfs vfs.parse.fs.= LPORT 44321 44321 44321 44321 44321 44321 44321 44321 5 8 8 8 8 8 8 8 8 8 8 8 8 8		Fs.tatk.(d) → fs.unlick exattr → vfs.jick 20.18:53 20.18:50 20.18:10 20.17:55 20.17:55 TIME ~ 20.18:03 20.17:55		vfs,unitev — vfs,factvice — v COMM curl wget NetworkManager priwebd priwebd LADDR:LPORT 2a02.8388.9aa7:a00.04.22 2a02.8388.9aa7:a00.04.22	s 201830 2017 fts_mkdr — vfs_symin trace TCP oc SADDR 192,168.0.80 192,00.1 192,00	835 20.1840 + - vfs_falocate onnect() a00.0 ransmits RADDR.RP0F 2:904.fa87.50 2:904.fa87.50	20:18:45 20:18:45 VS_mdr 33536 59034 59154 50088 47540 38034 38032 20 20 20 20 20:18:45 20 20:18:45	201850 201857 vfs_genaam vfs_genaam vfs_gen	201900 201900 PPORT 443 443 443 40 80 80 44321 44321 44321 44321 44321 44321 44521 44521 44521 44521 44521
	 bpftrace 2019 vit_fync,ran, and the second seco	2017.05 2011 2017.05 201 2017.05.05 2017.05.05 2017.05.05 2017.05.05 2017.05	201715 2 attr - /fs. getattr.nose - /fs. getattr.nose - /fs - /fs. last_lock - /fs pmcd - pmcd - p	e	vit_uead → vit_u = vit_uget.tree → pt/ associate a	EADDR LADDR 127.0.0.1 127.0.0.	e — vfs.statfs vfs.parse.fs.= 44321 44321 44321 44321 44321 44321 44321 44321 5 8 8 8 8 8 8 8 8 8 8 8 8 8		TIME → 20.18.53 20.18.50 20.18.50 20.18.10 20.17.55 20.17.55 20.14.60 20.14.60 20.14.20		vfs_writev -vfs_factrice -vfs_factrice curl	s 2018:00 20:1 fs_mAdr — vfs_symin trace TCP cc SADDR 192:168.0.80 192:168.0.80 2002:8388.a8c3: 192:168.0.80 127:0.0.1 127:0.0.1 127:0.0.1 trace TCP ret 906 906	ess 20.1840 + - vfs_falocate 	2018.45 vfs_mdir 33536 59034 59154 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032	201850 201854 • vfs_getaattr • vfs_get DADDR 152.199.19.160 192.30.253.120 140.82.118.4 200141785000: 152.19.134.198 127.0.0.1 127.0.0.1 127.0.0.1 127.0.0.1	201900 201900 PPOET 443 443 443 80 80 80 44321 44321 44321 44321 44321 44321 44321 44321 44321 44321 44321
	 bpftrace 2019 vh.fopc.nn vh.setuati Network TIME • 2017:55 2017:55<!--</td--><td>2705 2017 ge → th_set → th_innered 2888 38888 3888 3888 3888 38888 3888 3888 3888 3888 3</td><td>710 201715 2 attr — Y5, getattr_noise — Y5, Let Jock — Y1 pmcd pmcd 4 pmcd 4 pm</td><td>c</td><td>vfu_uead → vfu_u = vfu_get.tree → pt/ 38034 38032 38032 38028 38028 38022 38024 38022 38022 000.004 0.00.004 000.004</td><td>EADDR - vfs_parae. 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 4253.5441.58540 4553.5441.58546 4553.5441.58546</td><td>e — vfs_gaarfs_fs_ vfs_parse_fs_s LPORT 44321 44321 44321 44321 44321 44321 44321 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6</td><td></td><td>TIME + 20.18.53 20.18.50 20.18.50 20.18.50 20.18.50 20.17.55 TIME + 20.13.10 20.14.10 20.17.55 20.14.30 20.14.30 20.14.31 20.14.32 20.14.34</td><td></td><td>vfs_writev vfs_factrice v curl v v wget v v NetworkManager v v prmwebd v v LADDR:LPORT 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422</td><td>s 2018/20 20:1 fts_mkdr = vfs_symbol trace TCP cc SADDR 192.168.0.80</td><td>835 20.1840 extra strain 44 a00:0 44 ransmits 2004/fa87.50 2004/fa87.50 2004/fa87.50 2004/fa87.50 2004/fa87.50</td><td>2018.45 2018.45 vfs_mdir 33536 59054 59054 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 39</td><td>201850 201854 • vfs_getattr • vfs_get DADDR 152.199.19.160 192.0253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1 127.0.0.1 480 480 480</td><td>201900 201900 PPORT 443 443 443 80 80 80 44321 44324 44324 44324 44324 44344 445444 4454444 4454444444444</td>	2705 2017 ge → th_set → th_innered 2888 38888 3888 3888 3888 38888 3888 3888 3888 3888 3	710 201715 2 attr — Y5, getattr_noise — Y5, Let Jock — Y1 pmcd pmcd 4 pmcd 4 pm	c	vfu_uead → vfu_u = vfu_get.tree → pt/ 38034 38032 38032 38028 38028 38022 38024 38022 38022 000.004 0.00.004 000.004	EADDR - vfs_parae. 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 4253.5441.58540 4553.5441.58546 4553.5441.58546	e — vfs_gaarfs_fs_ vfs_parse_fs_s LPORT 44321 44321 44321 44321 44321 44321 44321 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6		TIME + 20.18.53 20.18.50 20.18.50 20.18.50 20.18.50 20.17.55 TIME + 20.13.10 20.14.10 20.17.55 20.14.30 20.14.30 20.14.31 20.14.32 20.14.34		vfs_writev vfs_factrice v curl v v wget v v NetworkManager v v prmwebd v v LADDR:LPORT 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422	s 2018/20 20:1 fts_mkdr = vfs_symbol trace TCP cc SADDR 192.168.0.80	835 20.1840 extra strain 44 a00:0 44 ransmits 2004/fa87.50 2004/fa87.50 2004/fa87.50 2004/fa87.50 2004/fa87.50	2018.45 2018.45 vfs_mdir 33536 59054 59054 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 39	201850 201854 • vfs_getattr • vfs_get DADDR 152.199.19.160 192.0253.120 140.82.118.4 2001.4178.5000: 152.19.134.198 127.0.0.1 127.0.0.1 480 480 480	201900 201900 PPORT 443 443 443 80 80 80 44321 44324 44324 44324 44324 44344 445444 4454444 4454444444444
	 bpftrace 2019 vfl.5ppC.rpr.2pl vfl.5ppC.rpr.	1705 201-01 92 - vfs.pm - vfs.mixod 38888 3888 3888 3888 3888 38888 3888 3888 3888 3888 3	2017.15 2 attr	c. – 4f,Jokć feie – Vfi,ogen, – ei c.centic.mouri – Vfi.2gen, and – races TCP acces RADDR 127.0.0.1	virund v	eadlink → vfs_paraet [s_param + vfs_paraet [s_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 25.0.5 25.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.58540 45.53.5441.585560 45.53.5441.58558 25.5451.5558 25.54	e — vfs_starfs vfs_partse_fs_s vfs_partse_fs_s 44321 44321 44321 44321 44321 44321 44321 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6		TIME + 2018:50 2018:50 2018:50 2018:50 2018:50 2018:50 2018:50 2018:50 2018:50 2018:50 2018:10 2017:55 2017:55 2014:60 2014:11		COMM - curl - wget - NetworkManager - pmwebd - LADDR:LPORT - 2a02:8388:9aa7::a00.0:42 - 2a02:8388:9aa7::a00.0:42 - 2a02:8388:9aa7::a00.0:42 -	s 2018.20 20:1 fts_mkdr = vts_symbol trace TCP cc SADDR 192.168.0.80	835 201840 + - vfs_falocate - vfs_falocate	2018.45 vfs_mdi 33536 59034 59154 59154 38034 38032 27 2000:4c41:535 000:4c41:535 000:4c41:535	201850 201857 415,000 201857 415,000 201457 201457 201457 2014578 20014178 20014 20014178 20014 20014 200 20014 20014 20014	DPORT 443 443 443 443 80 44321 4444 4444 4444
	 bpftrace 2019 vh.fopc.nn vh.setuati Network TIME • 2017:55 2017:55<!--</td--><td>2705 2017 ge → th_set → th_innered 2888 38888 3888 3888 3888 38888 3888 3888 3888 3888 3</td><td>201715 2 attr - yfs_optattr_norse - yfs_optattr_norse - yfs - yfs_optattr_norse - yfs - yfs_optattr_norse - yfs - princd - yfs - yfs</td><td>c</td><td>virunead — virun virunead — virun virunget.tree — virunget.tree — virunget.tree — virunget.tree — virunget.tree = virunget.tree = vir</td><td>EADDR - vfs_parae. 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 4253.5441.58540 4553.5441.58546 4553.5441.58546</td><td>e — vfs_starfs vfs_partse_fs_s LPORT 44321 44321 44321 44321 44321 44321 44321 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6</td><td></td><td>TIME + 20.18.53 20.18.50 20.18.50 20.18.50 20.18.50 20.17.55 TIME + 20.13.10 20.14.10 20.17.55 20.14.30 20.14.30 20.14.31 20.14.32 20.14.34</td><td></td><td>vfs_writev vfs_factrice v curl v v wget v v NetworkManager v v prmwebd v v LADDR:LPORT 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422</td><td></td><td>8:35 20:18-40 8:35 20:18-40 anneet() </td><td>2018.45 2018.45 vfs_mdir 33536 59054 59154 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 39</td><td>Contestion Contestion C</td><td>201900 201900 PPORT 443 443 443 80 80 80 44321 44324 44324 44324 44324 44344 445444 4454444 4454444444444</td>	2705 2017 ge → th_set → th_innered 2888 38888 3888 3888 3888 38888 3888 3888 3888 3888 3	201715 2 attr - yfs_optattr_norse - yfs_optattr_norse - yfs - yfs_optattr_norse - yfs - yfs_optattr_norse - yfs - princd - yfs - yfs	c	virunead — virun virunead — virun virunget.tree — virunget.tree — virunget.tree — virunget.tree — virunget.tree = virunget.tree = vir	EADDR - vfs_parae. 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 4253.5441.58540 4553.5441.58546 4553.5441.58546	e — vfs_starfs vfs_partse_fs_s LPORT 44321 44321 44321 44321 44321 44321 44321 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6		TIME + 20.18.53 20.18.50 20.18.50 20.18.50 20.18.50 20.17.55 TIME + 20.13.10 20.14.10 20.17.55 20.14.30 20.14.30 20.14.31 20.14.32 20.14.34		vfs_writev vfs_factrice v curl v v wget v v NetworkManager v v prmwebd v v LADDR:LPORT 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422 2a02.8388.9aa7.a00.0.422		8:35 20:18-40 8:35 20:18-40 anneet()	2018.45 2018.45 vfs_mdir 33536 59054 59154 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38032 47540 38034 38034 38032 47540 38034 38034 38032 47540 38034 39	Contestion C	201900 201900 PPORT 443 443 443 80 80 80 44321 44324 44324 44324 44324 44344 445444 4454444 4454444444444

bpftrace code editor

▼ A										
	15									
	16	tracepoint:sc	hed:sched_	wakeup	, tracepo	oint:sch	ed:sched_			
	17	{			tracepo	pint:sch	ed:sched_kthre	ead_stop		probe
							probe			
	19 } 20 tracepoint:sched:sched_migrate_task					probe				
	21	tracepoint:sc	hed:sched_	_switch			ed:sched_move_			probe
	22	{		TAC			ed:sched_pi_se			probe
	23 24	@atime[ar	gs->prev_p	= 1A5			ed:sched_proce ed:sched_proce			probe
	25	}	<u> </u>	-			ed:sched_proce			probe probe
	26 27	¢nc - Aatim		out oid		Jine: Jen	curbencu_proce			pi obc
	27	\$ns = @qtim if (\$ns) {	elar.gs-sue	ext_pid	ۇ [
	29		hist((nsed	cs - \$n	<mark>s)</mark> / 1000	ð);				
	30 31	<pre>} doloto(Octi </pre>	molonge N	ovt ni	41).					
	32	<pre>delete(@qti }</pre>	mela.g2-21	lexc_pr	u]),					
	33									
	34 35	END r								
		{								
	Legend	legend format		0	Format	Heatm	ap 🔻 URL	override	URL	
10										
12 13	* 08-: */	Sep-2018 Brenda	n Gregg Crea	ated thi						
14										
15 16	/arg/	e:hz:99								
17 18	argN	- lbict(cpu 0	1000 1).				variable	argN		
18	աշքս }	= lhist(cpu, 0,	1000, 1),					Argument to	the function being tra	ced (arg0, arg1, etc.)
Lanand		format	C Farm		ootmon -		override UDI	J.		
Legend	legend	l format	6 Forn	hat H	eatmap 🖣	URL	override URL			
9 10	* 13-	Sep-2018 Brenda	an Gregg (r	eated th	nis.					
11										
12 13	// inc	lude: @usecs								
14	kprobe	:blk_account_io	_start							
15 16	{ @sta	rt[arg0] = nsec	c •							
17	}									
18 19	knnobe	:blk account io	done							
20		t[arg0]/	aone							
21 22	{ @use	<mark>cs</mark> = lhi								
23	dele	te(@ lhist()						function	lhist(int n. int m	in, int max, int step)
24 25	}									
26	END								Produce a linear h:	istogram of values of <mark>n</mark>
27 28	{ clea	r(@start);								
28 29	}									
				rmot	Hootrage	- UDI	override URL			
Legend	legen		6 Fo	rmat	Heatmap	▼ URL	overnue ORL			

bpftrace flame graphs

PCP bpftrace Flame Graphs -	
Kernel Stacks	
	Reset zoom Q. Search
12:03:15 - 12:05:06	Reset zoom Q Search
entry SYSCALL do syscall 64+95	
x 64 sys sendt	
sys_sendto+238	
sock_sendmsg+87 tcp_sendmsg+40	
tep sentansy rate	
tcp push pend	
top write milt+	
top_transmit_s ip_queue_xmit	
ip output+113	
ip finish output	
local bh_enabl	
do softing own	
do softirq+238 ret from f	i de la companya de l
net_rx_action+328 kthread+	
entry process backlog vorker th do sy netif receive process o	
do wr ip rcv+188 delayed f	
vfs_w ip local_deliverentry_SY	entry entry_SYS entry_SYSCAL ent
entry S do ite ip local deliver ent dentry do syscal do sysc do ite ip protocol deli do evict+203 x64 sys	do sys do syscall do syscall 64 do x64 x64 sys x64 sys epol x secondary startup 64+
kysys do te it py frozon uein do jent + 205 x share ev kys joint	do fu do epoll do epoll wait do s start secondary sarup 0++
vfs wri sock se tcp v4 do rcv+ do shmem tr do vfs io	futex ep poll+1 ep poll+1044 poll cpu_startup_entry+25
new_sy unix_st tcp_rcv_establis fut shmem_u drm_ioctl dos syscall_64+95	futex schedule_hr schedule_hrtimeout_ra do_idle+351
pipe_w sock_def_readable+60 wa truncate_i drm_ioctl page_fa ksys_write+95 wake up_common lock+138 try delete fro i915 gem entry_SYSCA do page vfs write+182	schedule+57 schedule_idle+40 entry S
raw when by common rock trade of the second	finish task switch+123 do sysc
root	

Metric Search

් ර	Performance Co-Pilot Full-Text Metric Search	
 	Q disk Sea Metrica V Instance Domains	earch 🕒 Back To Bookmarks & Latest Searches
@ 4 /*	56 results Elapsed: 0.000 disk.all.read	000758s No bookmarks saved. Search History:
\$ @ ()	total read operations, summed for all disks ♦ Metric → Read Mi	
	disk.all.write total write operations, summed for all disks ☉ Metric → Read Mr	
	disk, all, bikread block read operations, summed for all disks ▷ Metric → Read M	
5	disk.all, bikwrite block write operations, summed for all disks ℃ Metric → Read M	More

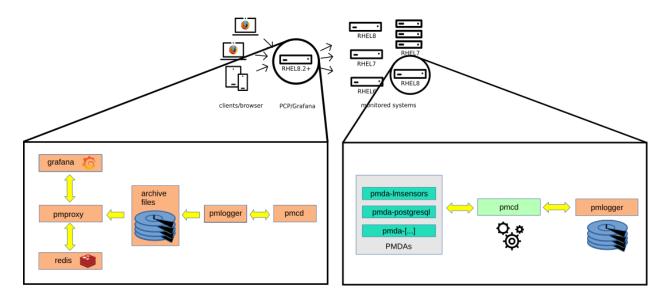


Fig. 1: © Christian Horn

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 3.0.3 (2021-02-24)

• test: fix e2e tests by using another CSS selector

- chore: update dependencies
- docs: add container guide and screenshot

2.5.2 3.0.2 (2021-01-22)

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

2.5.3 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

2.5.4 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

2.5.5 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.

2.5.6 2.0.2 (2020-02-25)

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

2.5.7 2.0.1 (2020-02-17)

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

2.5.8 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

2.5.9 2.0.0-beta1 (2019-12-12)

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

2.5.10 1.0.7 (2020-01-29)

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

2.5.11 1.0.6 (2020-01-07)

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

2.5.12 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

2.5.13 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

2.5.14 1.0.3 (2019-11-22)

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

2.5.15 1.0.2 (2019-11-12)

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

2.5.16 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

2.5.17 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

2.5.18 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:

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

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

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

2.8.6 Query Functions

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

Function	Description	Example	
metrics([patterr]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*)	1
	defined, all metrics are returned)		
label_values(lab	ereturns all label values for the specified label	label_values(hostna	name

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
```

```
Suuo ./instaii
```

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

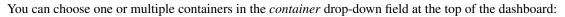
grafana-pcp includes the following (optional) dashboards:

- 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





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

When I try to add a datasource in Grafana, I get: "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 find out if so in \$PCP_LOG_DIR/pmproxy.log