Performance Co-Pilot cheat sheet

Christian Horn <chorn@fluxcoil.net>

PCP basics

Installation
Package **zero-conf** pulls in dependencies, starts daemons, starts archiving of a default set of metrics. On RPM based distros (RHEL, Fedora, CentOS etc.):
dnf -y install pcp-zeroconf
Installation via Ansible playbook:
linux-system-roles.metrics
Verify pcp installation:
pcp
systemctl status pmcd pmlogger

Important tools
Tools from package 'pcp-system-tools' can be used with either the running pmcd, or PCP archive files:

- pcp atop
- pcp collectcl
- pcp free
- pcp iostat
- pcp dstat

Important Pathes
/var/lib/pcp/config/
/var/log/pcp/
/etc/pcp/

Working with metrics
Which metrics are offered by the running pmcd?
pminfo

Which metrics related to cpu are available?
pminfo | grep cpu

pmie

pmie, performance metrics interference engine, can react on defined metric states: send email on high load, and so on.
pmie –verbose –timestamp –interval 1
/etc/pcp/pmie/config.default
pmie –archive 20200512 –config <rules>

pmdas

PMDA installation
PMDA's are code pieces capable of reading metrics from their area like sensors, database, and so on. PMDS's can be searched as packages and installed, for example on yum4/dnf distros:
dnf search pcp-pmda
dnf install -y pcp-pmda-lmsensors
cd /var/lib/pcp/pmdas/lmsensors
less README
./Install

Anomaly search in archives

Which metrics are remarkably different in a certain timeframe? Example: we had I/O problems from 2am to 3am:
cat cull
rsyslog.
pmdiff -z -X ./cull -q 10 \
–start @10:00 –finish @10:30 \
–begin @12:00 –end @12:30 \
./archives/20120512 | less

What are the top 5 cpu and memory hogs?
pmrep proc.hog.cpu, proc.memory.rss \
-J 5 -1 -g -b MB

What are the current PIDs, and how much rss uses process with pid 75?
pmrep proc.smaps.rss -g | less
pmrep proc.smaps.rss -g -i '.*75.*'

Archive files

Basics
Which archive is pmlogger logging into?
pcp

Set a variable to current archive, and evaluate how many metrics are logged in the archive:
cd /var/log/pcp/pmlogger/<hostname>
pminfo -a <archivename>| wc -l

Have pmdiff point out 'significant peaks' in archives:
pmdiff -a <archivename>

Accessing metrics
Most basic access to metrics:
pmstat -t 1
pmrep -a <archivename><metric>
pcp -a 20180831.11.31 –origin @1pm \
dstat –time –disk –mem 60sec 10
pmrep -a 20220128 kernel.cpu -p -t 300 \
–hostzone -S @19:30

Graphical access
pmchart
dnf -y install pcp2pdf; pcp2pdf -a <arch>

Links

[1] kbase: Index of (PCP) articles, solutions, tutorials, white papers

[2] Ansible:https://github.com/linux-system-roles/metrics
https://github.com/performancecopilot/ansible-pcp

[3] Performance Co-Pilot site

[4] Articles: Solve performance mysteries with PCP / PCP and podman / PCP and dstat

Remote collection

Install PCP on clients
Setup client systems to offer metrics via pmcd: install pcp, open packet based firewall, enable remote access in pmcd:
yum -y install pcp
firewall-cmd –permanent –zone=public –add-port=44321/tcp
firewall-cmd –reload
if grep -q PMCD_LOCAL /etc/sysconfig/pmcd; then
sed -ie 's,PMCD_LOCAL.*,PMCD_LOCAL=0,' /etc/sysconfig/pmcd
else
echo 'PMCD_LOCAL=0' »/etc/sysconfig/pmcd
fi
grep PMCD_LOCAL /etc/sysconfig/pmcd
service pmcd restart
chkconfig pmcd on

Install PCP on collector system
On the collector, we install pcp-zeroconf which also sets up logging to archive files. We then set variable CLIENT to the clients name, create a config- and controlfile, and notify pmlogger of the changes. yum -y install pcp-zeroconf
CLIENT=rhel7u8a
/usr/libexec/pcp/bin/pmlogconf \
/var/lib/pcp/config/pmlogger/config.$CLIENT
optionally, execute the last command a second time
echo "$CLIENT.local n n PCP_LOG_DIR/pmlogger/$CLIENT.local" \
" -r -T30d -c config.$CLIENT" \
¿/etc/pcp/pmlogger/control.d/$CLIENT
/usr/libexec/pcp/bin/pmlogger_check