Monitoring with

Performance Co-Pilot

David Disseldorp
ddiss@suse.com

Presentation Overview

- Introduction to Performance Co-Pilot (PCP)
- Demonstration
- Monitoring Your Application with PCP

What is Performance Co-Pilot?

- PCP is a system level performance monitoring toolkit
- Collection, monitoring and analysis of system metrics

Cross platform: Linux, Mac OS and Windows

End-to-end: Hardware, Core OS, services and applications

- Distributed architecture

Monitoring of local and remote nodes

- Real-time or retrospective

Live system or archive

- Pluggable

New agents system metrics within PCP

What is Performance Co-Pilot?

- Roles broadly divided into two categories

P[illegible]ers: Collect and export performance metrics

C[illegible]umers: Record, visualize, monitor and analyze performance

- Hosts may operate as producers, consumers or both

Multiple consumers may connect with one or more producers

Core Components

Performance Metric Domain Agents

Extracts metric data from a system component for exposure within PCP

Performance Metrics Collector Daemon (PMCD)

Coordinates handling of fetch requests between monitoring applications and agents

pmlogger

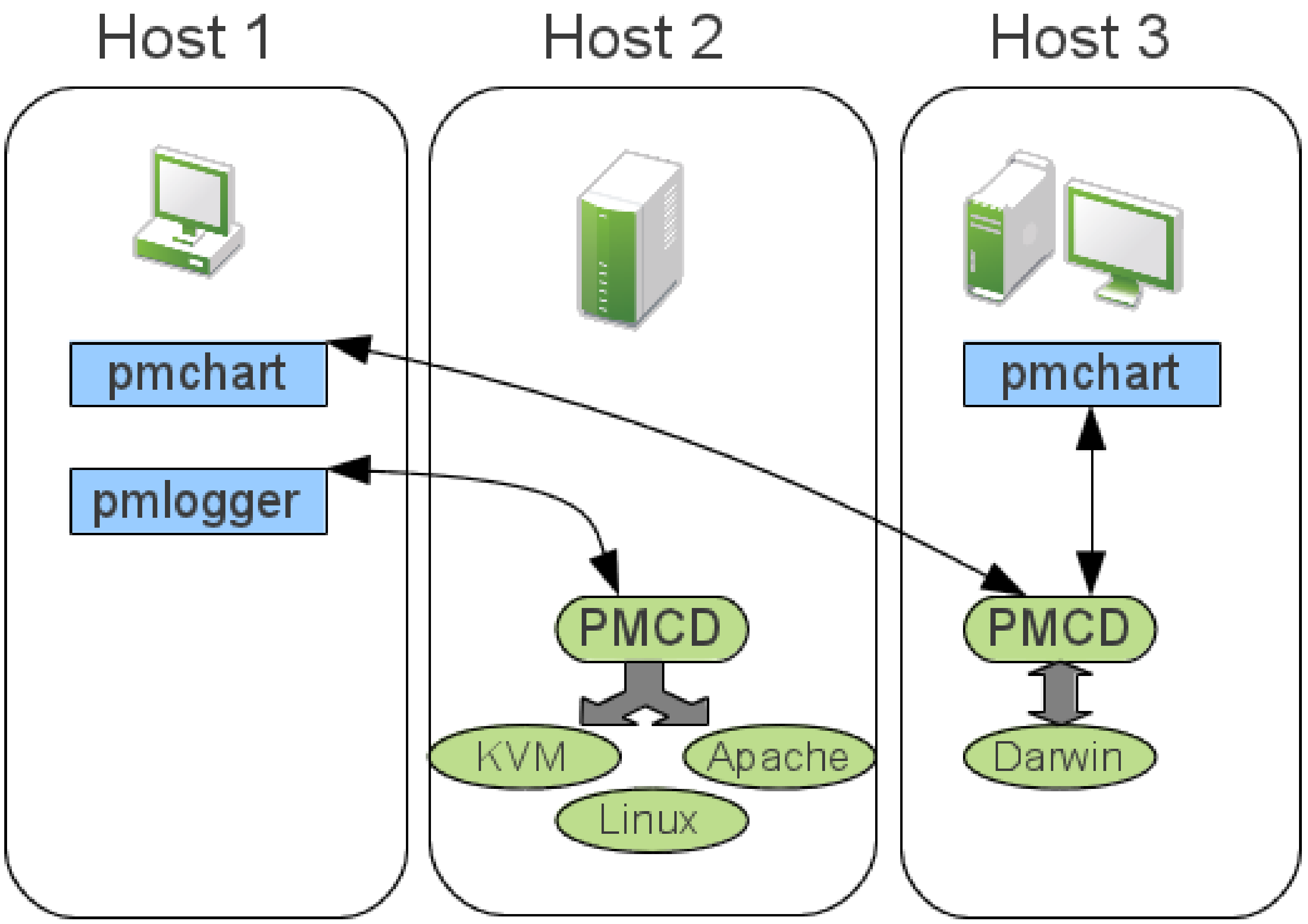
Utility to capture and store metrics exported by PMCD

pmchart

GUI providing charting of metrics in real-time as well as retrospectively

What is Performance Co-Pilot

Distributed Architecture



Use Cases

- Administrative

Tracking of resource utilization

Understand how workload effects usage

Capacity planning

Benchmarking

- Debugging

System postmortem

Identification of performance regressions

Side by side comparison of behavior across application versions

Isolation of problematic behavior

Performance Metric Domain Agents

- Exports metrics obtained from underlying data source

Each agent is responsible for a specific metrics domain

Communicates with PMCD on the local system

- Many agents currently available

Linux & Windows

CPU, Disk, Memory, Network, Filesystems, Per-process statistics

Hypervisors

Databases

Sendmail

Performance Metrics Collection Daemon

- One pmcd process running per-host

Manages metrics extraction from all agents

Routes client requests to one or more agents, aggregates response

Maintains namespace for all metrics exposed

Name space maps external metric names to internal numeric identifiers

- Accepts connections from monitoring utilities

Single point of contact for local or remote monitoring agents

Listens on TCP port 44321 by default

Primitive host based access control

Authentication and encryption not currently supported

pmlogger

- Command line utility for metrics archival

Concurrent logging of local and remote hosts

Simple list style configuration

What metrics should be collected and how frequently

- Archive playback via pmchart and pmval

Retrospective analysis of system state

Compare archives from working and non-working situations

Archives self-contained allowing analysis off-site

- Tools for archive management

pmlogger_daily, pmlogger_check, pmlogger_merge

Log rotation and aggregation, set and forget

pmlc

Dynamic runtime re-configuration

pmchart

- Visualization of metrics
- Fully Customizable charting canvas

Multiple metrics (of same units) per chart

Multiple charts per tab

Line, bar, stacked bars, area plot, utilization

Save window preferences as a “view”

Collection of charts, metrics, graph styles, legends, labels

Integrated time control

VCR style stop, play, record, rewind paradigm

Source metrics from one or more live systems

Alternatively one or more archives

Other tools

- pmie

Evaluate rules or expressions

Perform an action

Automatic detection of performance anomalies

- pmlogsummary

Calculate statistics across an archive time window

- pmstore

Selectively modify state in an agent

Toggle debug flags, enable optional instrumentation, etc.

- pmval

Command line based dumping of values in realtime or from archive

- pmproxy

Proxy PCP requests between a head node and an isolated network

- Parfait[6]

Externally maintained Java library capable of exposing metrics to PCP

Demonstration

Writing your own agent

- What values am I capturing?
- Metrics definition

Semantics

e.g. *counter* (value is monotonically increasing)

Units

e.g. *bytes*

Data type

e.g. *64-bit unsigned int*

Instances

e.g. *eth0, eth1*

Transient

Value

Instances

- Namespace

Each PMDA requires a unique domain identifier

Writing your own agent

- What language?

C, Perl, Java

- Architecture

Separate process managed by PMCD

Dynamic Shared Object

PMCD is latency sensitive and must be stable

- How can the agent access these counters?

Memory mapped file, kernel proc file, IPC

Generic MMV agent

Self descriptive memory mapped files

Agent Example

Clustered Trivial Database (ctdb)

- ctdb already maintains per-node stats:

```
hex-14:~ # ctdb statistics
 num_clients          6
 total_calls          77
 pending_calls        0
 memory_used          73691
 max_call_latency     0.000549 sec
```

- Metrics without instance domain

Number of clients = instantaneous

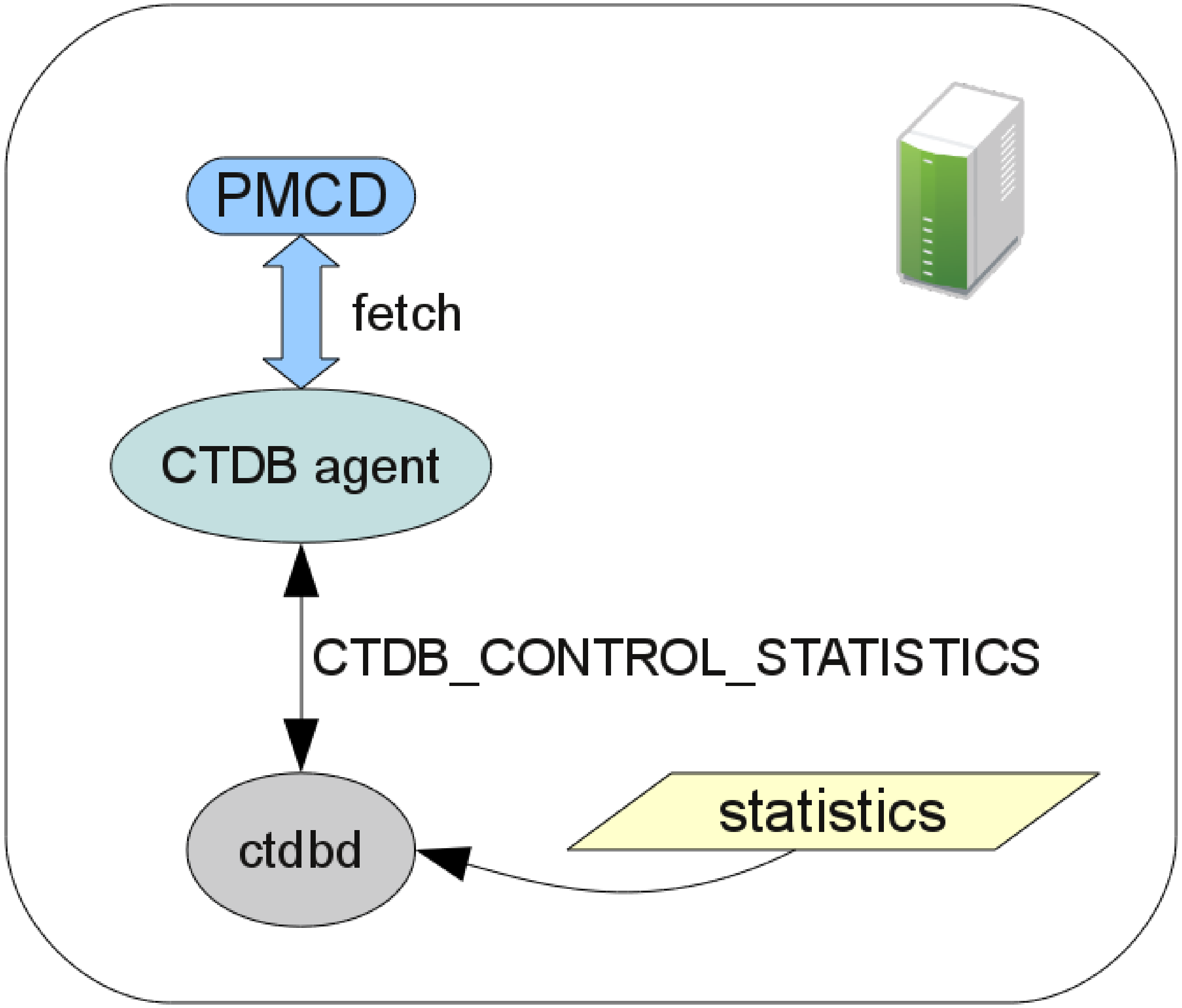
Total calls = monotonic increasing counter

Pending calls = instantaneous

Memory used = instantaneous, bytes units

Maximum call latency = instantaneous, seconds units

Agent Example
Clustered Trivial Database (ctdb)



Agent Example
Clustered Trivial Database (ctdb)

- Namespace definition

Per-metric identifiers

```
src> cat pmns
ctdb {
        num_clients             155:0:0
        total_calls             155:13:24
        pending_calls           155:14:25
        memory_used             155:19:30
        max_call_latency        155:23:34
}
```

Agent Example

Clustered Trivial Database (ctdb)

- Metrics definition

```
src> vi pmda_ctdb.c
static pmdaMetric metrictab[] = {
...
        /* max_call_latency */
        { NULL, { PMDA_PMID(23,34),
                        PM_TYPE_DOUBLE,
                        PM_INDOM_NULL,
                        PM_SEM_INSTANT,
                        PMDA_PMUNITS(0,1,0,0,PM_TIME_SEC,0) }, },
};
```

Agent Example

Clustered Trivial Database (ctdb)

- Main Program

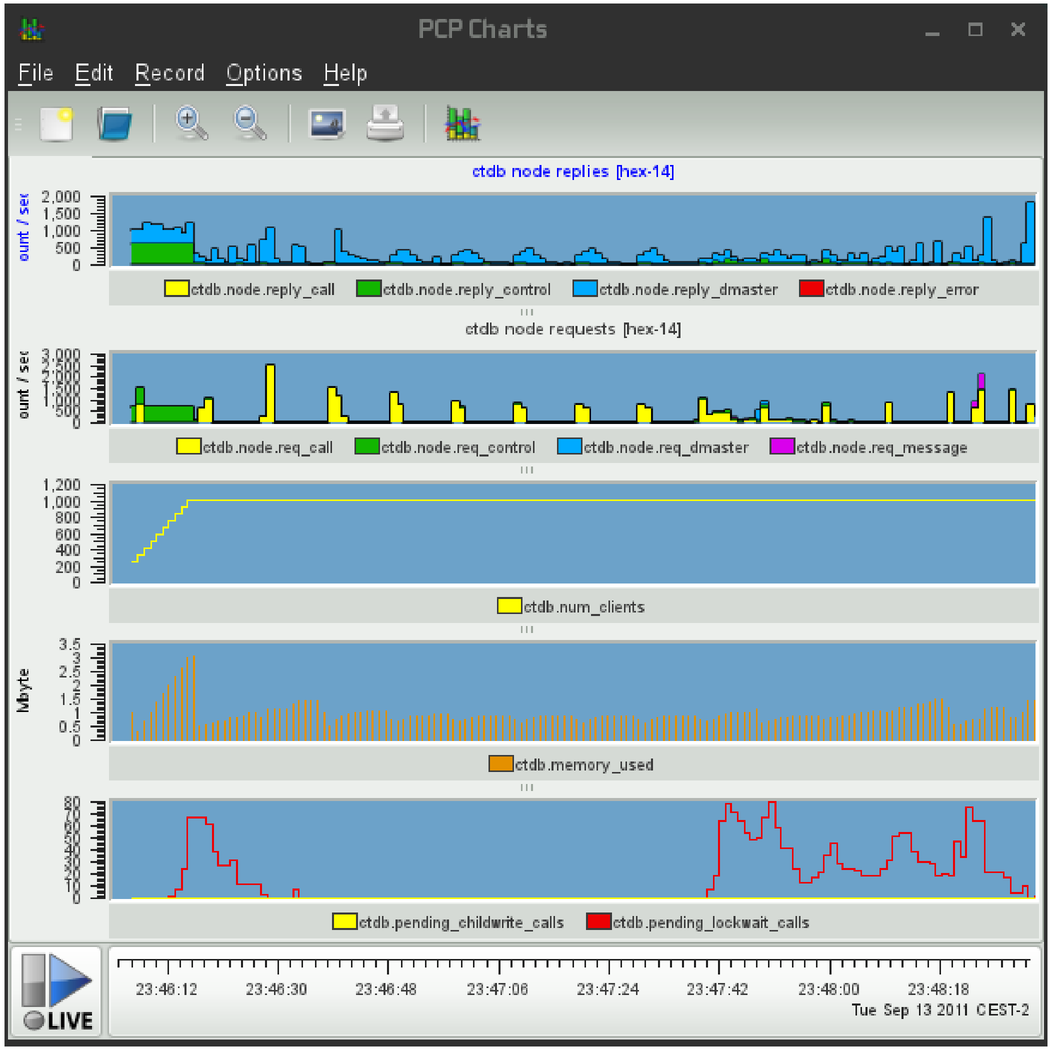
```
pmdaDaemon()              /* initialise daemon context */

pmdaSetFetchCallBack()    /* register handler for PMCD fetch requests */

pmdaInit()                /* export supported metrics */

pmdaConnect()             /* establish an IPC connection with PMCD */

pmdaMain()                /* main event loop */
```

- Two fetch callbacks from the event loop

Initial fetch request, then one per metric

Agent Example

Clustered Trivial Database (ctdb)


PCP Charts
File Edit Record Options Help
ctdb node replies [hex-14]
count / sec
🟨 ctdb.node.reply_call 🟩 ctdb.node.reply_control 🟦 ctdb.node.reply_dmaster 🟥 ctdb.node.reply_error
ctdb node requests [hex-14]
count / sec
🟨 ctdb.node.req_call 🟩 ctdb.node.req_control 🟦 ctdb.node.req_dmaster 🟪 ctdb.node.req_message
🟨 ctdb.num_clients
Mbyte
🟧 ctdb.memory_used
🟨 ctdb.pending_childwrite_calls 🟥 ctdb.pending_lockwait_calls
LIVE
23:46:12 23:46:30 23:46:48 23:47:06 23:47:24 23:47:42 23:48:00 23:48:18
Tue Sep 13 2011 CEST-2

Where can I get it?

- SGI project page

http://oss.sgi.com/projects/pcp/

- openSUSE Factory

Latest PCP base and GUI packages to ship with openSUSE 12.1

Outdated (base only) version in 11.4

- SUSE Gallery

“openSUSE Performance Co-Pilot” appliance

References

Performance Co-Pilot Website

http://oss.sgi.com/projects/pcp/

PCP Manual

http://oss.sgi.com/projects/pcp/pcp-gui.git/man/html/index.html

Performance Co-Pilot User's and Administrator's Guide

http://oss.sgi.com/projects/pcp/documentation.html

Performance Co-Pilot Programmer's Guide

http://oss.sgi.com/projects/pcp/documentation.html

Parfait – Java monitoring library

http://code.google.com/p/parfait/

Authentication and ACLs proposal

http://oss.sgi.com/archives/pcp/2011-06/msg00026.html

PCPIntro(1) Man page

RCE Podcast: Ken McDonell on Performance Co-Pilot

http://www.rce-cast.com/Podcast/rce-53-performance-co-pilot.html

PCP FAQ

http://oss.sgi.com/projects/pcp/faq.html

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
+www.suse.com

Join us on:
www.opensuse.org