Herding Penguins with Performance Co-Pilot

Ken McDonell

Performance Tools Group

SGI, Melbourne

Overview

- System-level performance management issues.
- Finding the useful performance data.
- Performance Co-Pilot - an open and extensible architecture.
- Automated performance monitoring.
- Monitoring and managing quality of service.

System-level performance management Issues

- Growing complexity of application architectures.
- Primary emphasis on end-user performance, not operating system performance.
- Centralized monitoring for distributed applications.
- Automate the mundane tasks, so people can tackle the hard problems.

Useful performance data

sgi

Like Chicken Man, it's everywhere …

- Hardware instrumentation
- Operating system kernel
- Libraries
- Service layers, daemons, etc.
- Applications

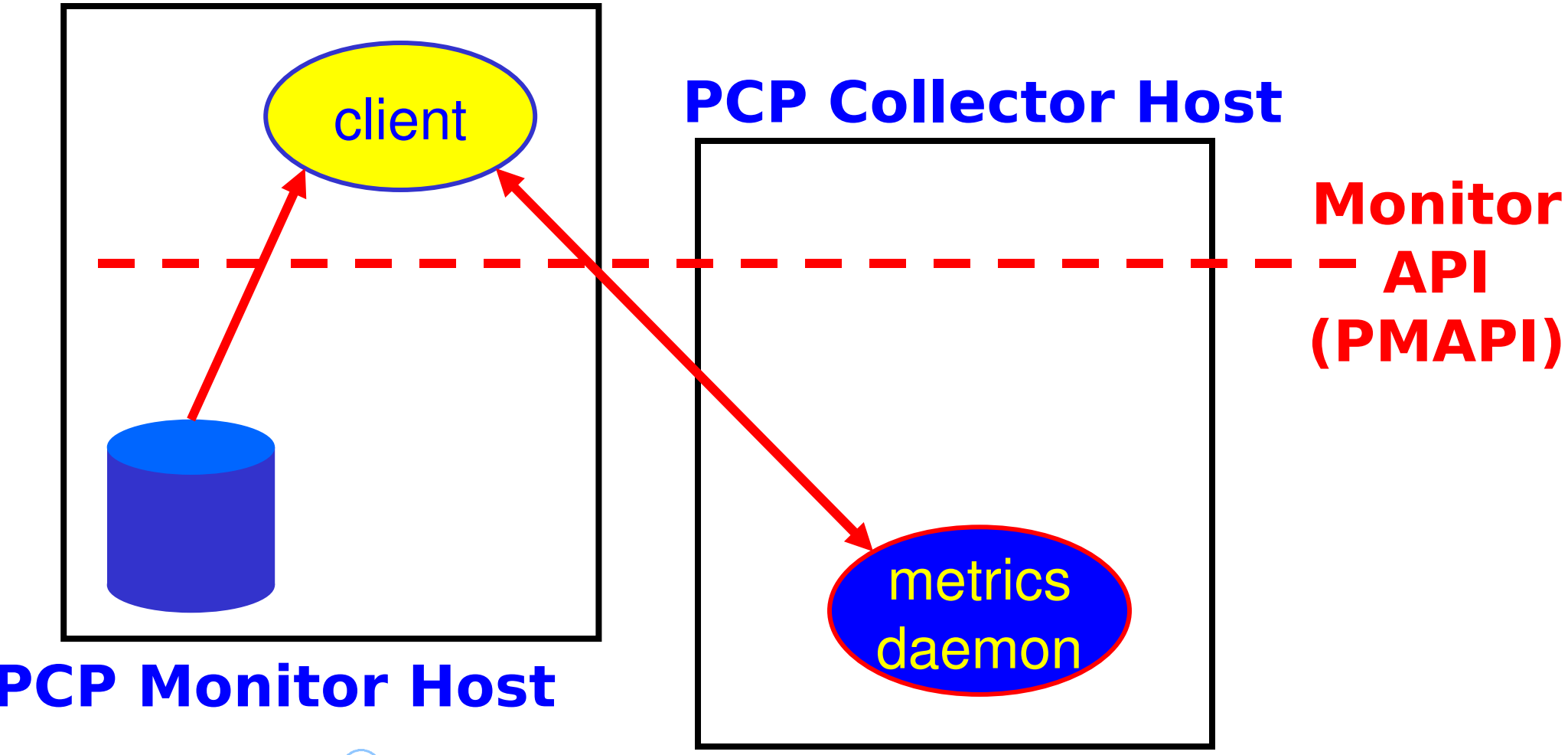
Spanning multiple hosts.

Both real-time and historical data.

Performance Co-Pilot

- Client-server architecture for centralized monitoring of distributed processing.
- Integrated archive services for recording and replaying performance data.
- One API for all performance data.
- Public interfaces and extensible frameworks at all levels.
- Open source release at http://oss.sgi.com/projects/pcp/

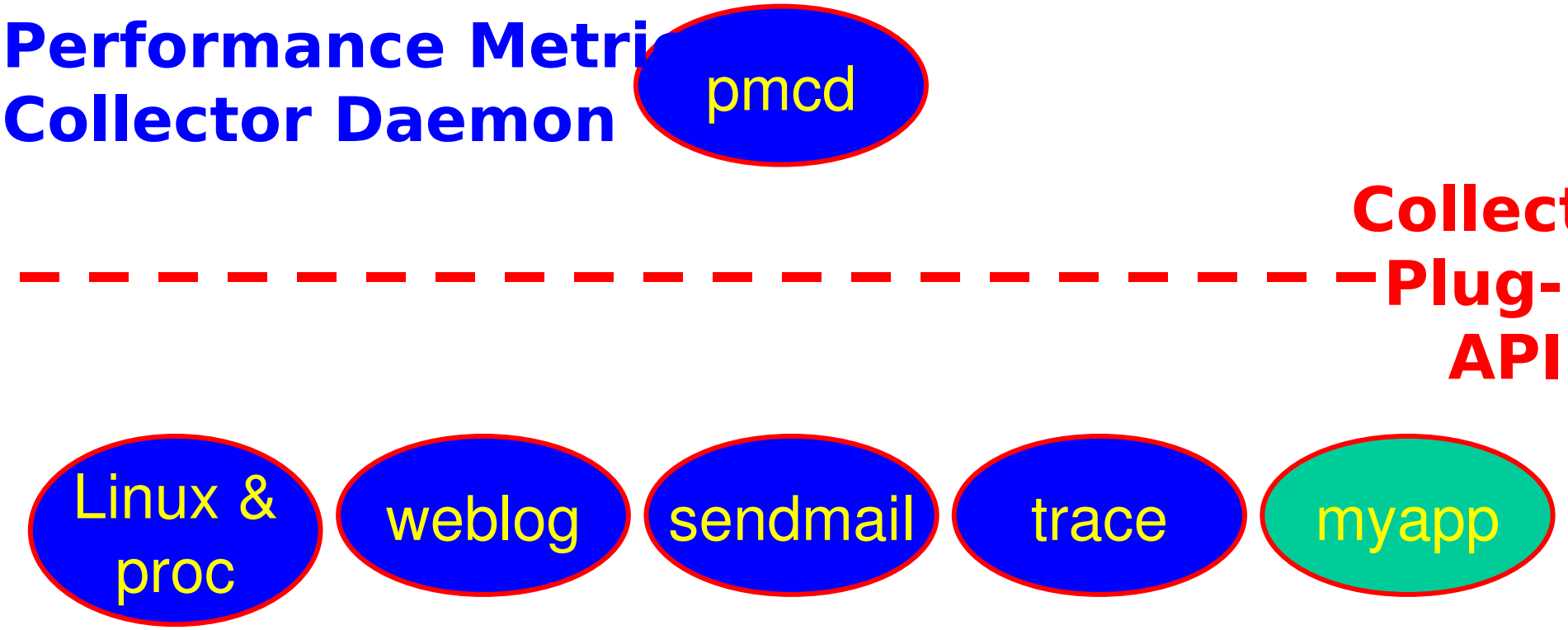
PCP architecture



PMAPI features

- Namespace services to discover available performance metrics.
- Metadata to describe metrics.
- Set-based data model for multiple instances and values.
- Pull-based retrieval for client-specified subsets of performance metrics.
- Archives and hosts as interchangeable sources of performance metrics.

PCP collector architecture



Collector plug-in API features

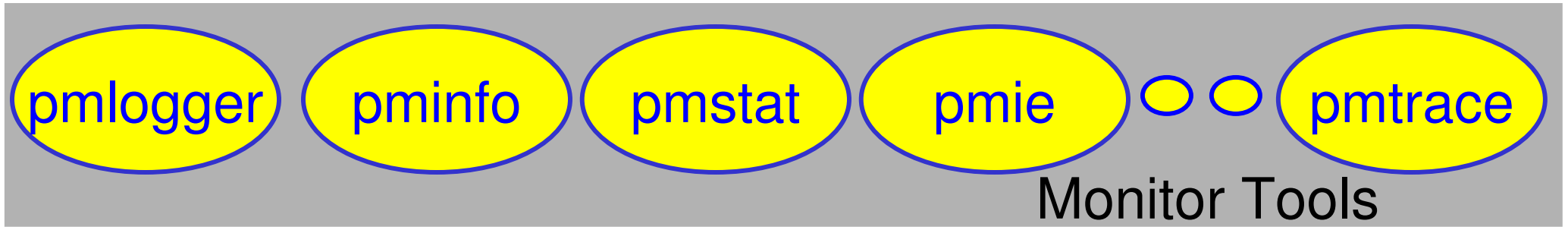
- Most of the complexity is encapsulated in a library … very short development times.
- Communication with PMCD via procedure call (DSO plug-ins), pipes or sockets.
- To access the real data, the plug-in (not PMCD) chooses an appropriate mechanism.
- Lots of source code examples.
- Great value in “roll your own” PMDAs.

Some efficiency issues

- PCP has to be part of the performance solution, not part of the problem!
- Expect PCP and plug-ins to consume less than 1% of one CPU on a collector host.
- PMCD is single threaded with service to completion for each client request.
- Plug-ins must respond quickly.
- Little state maintained in PMCD, clients do interval timing and rate calculations.

Open source PCP components

sgi

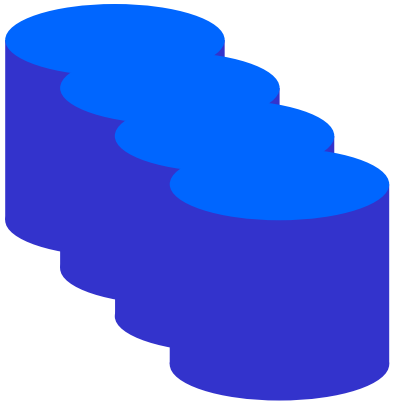


Monitor API

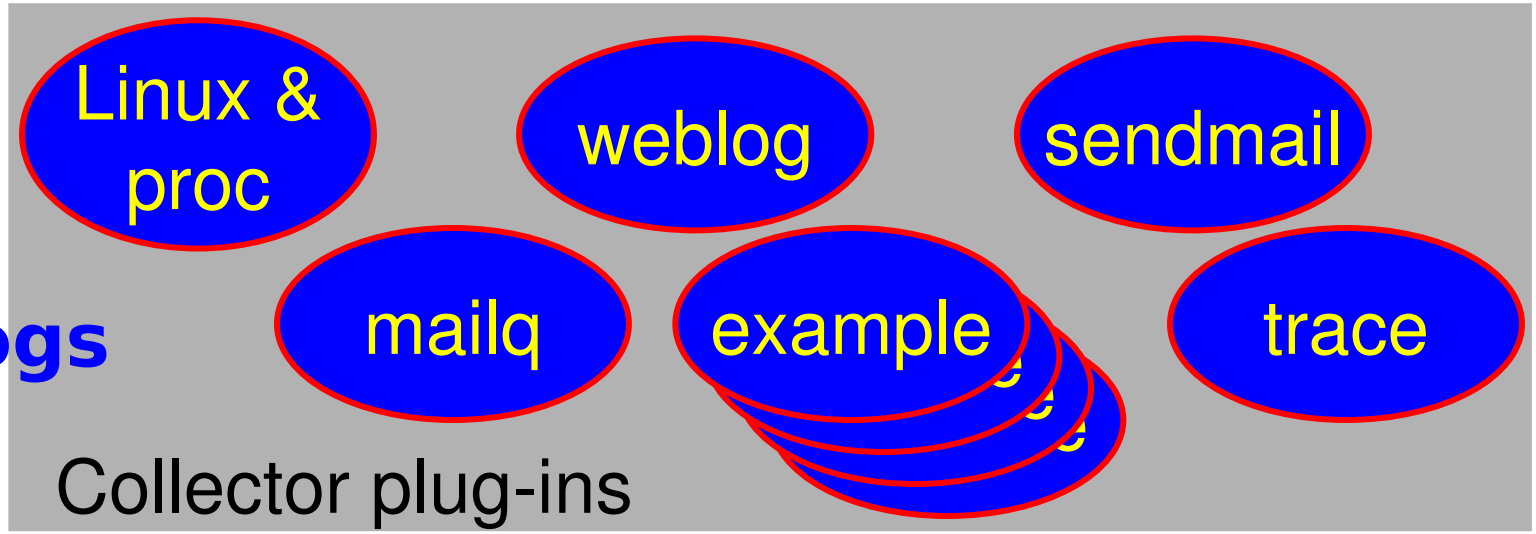
Trace API

pmcd

Collector plug-in API



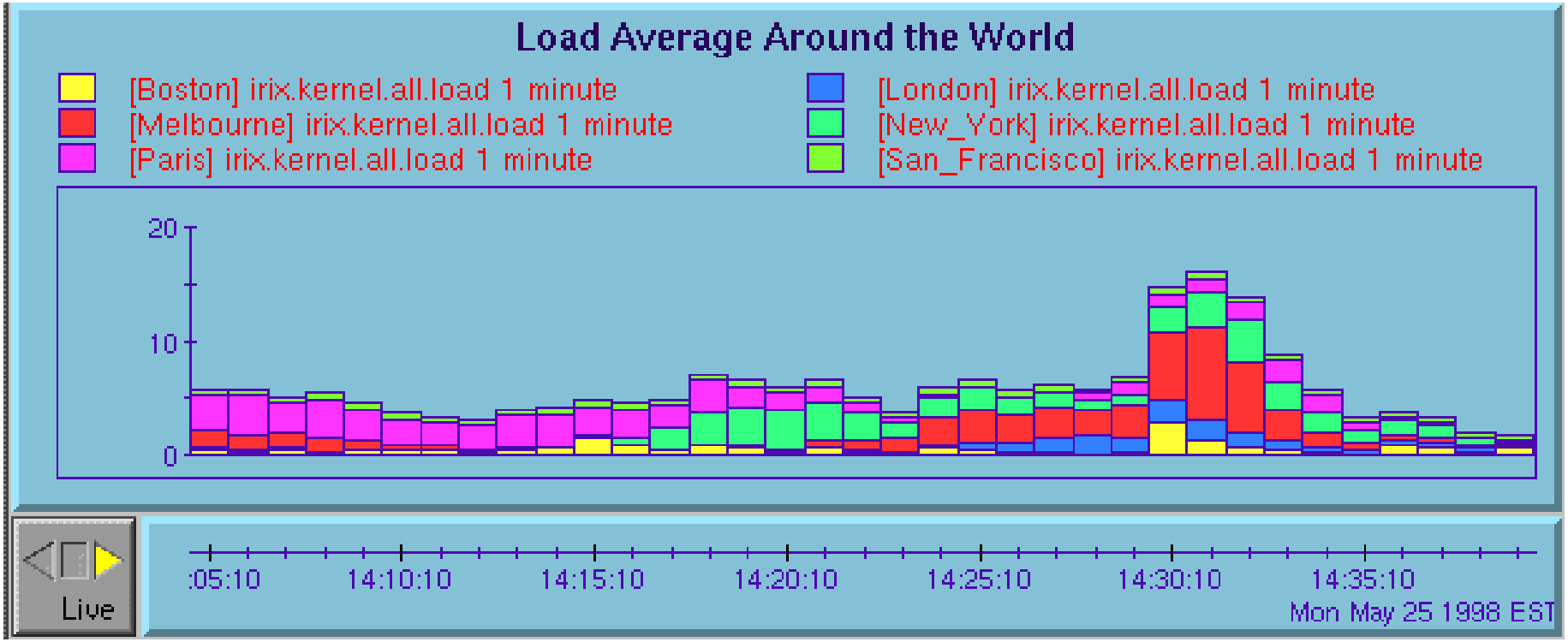
CP Archive Logs



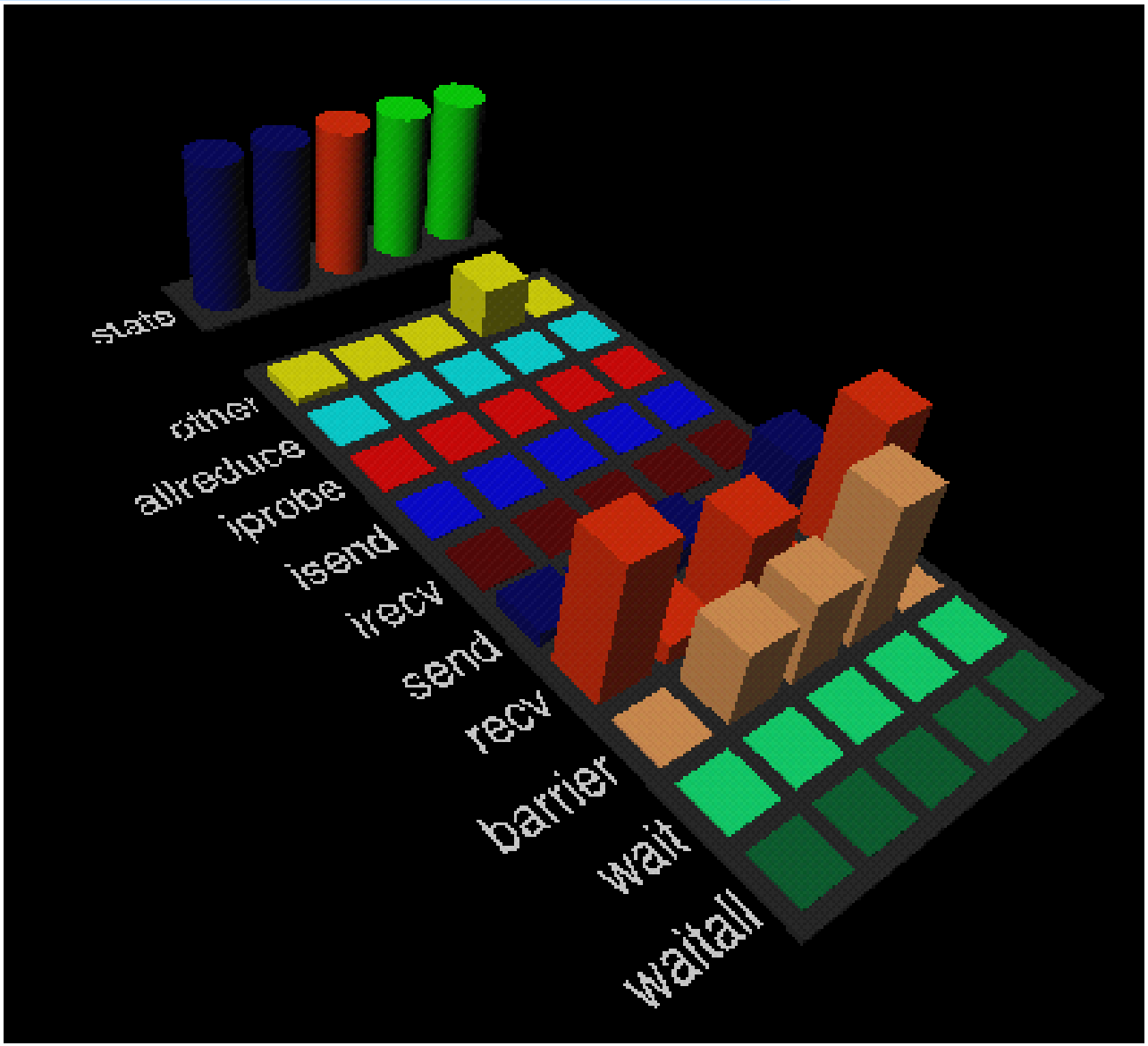
vmstat across the cluster

```
@ Wed Nov  3 12:55:08 1999
node     ld avg ... memory     swap          io     system          cpu
          1 min ...  cache   si   so     bi     bo    in    cs  us  sy  id
leesa      2.00 ...  17676    0    0      0      2   113    15   0   0 100
troppo     1.76 ...  40856    0    0      0    192   290   181  19  81   0
moomba     1.08 ...      ?    0    0   2512    467   674   280  13  18  70
gonzo      0.32 ...      ?    8    0    160      2  2289    57  10  26  64
thebeas    8.00 ...      ?    0    0      0      2   634    20  98   2   0
sandpit    0.00 ...      ?    0    0      0      0  1217    11   0   0 100
kuku       0.02 ...      ?    0    0      0      0  2468   108   2   1  98
snort     16.43 ...      ?   25   18    578   1468  8007   423  32  66   2
```

pmchart example



3-D visualization of application performance



Automated performance monitoring

- The PCP inference engine (pmie) evaluates predicate-action rules against a time series of performance data.
- Many common performance scenarios can be encoded in pmie predicates.
- Actions are very general … print, e-mail, insert event into management framework.
- Real-time for operations.
- Historical data for performance analysis.

Example pmie rules

- **High 1 minute load average (over 1.2 per CPU)**

  kernel.all.load #'1 minute'
  > 1.2 * hinv.ncpu
  → … arbitrary action …

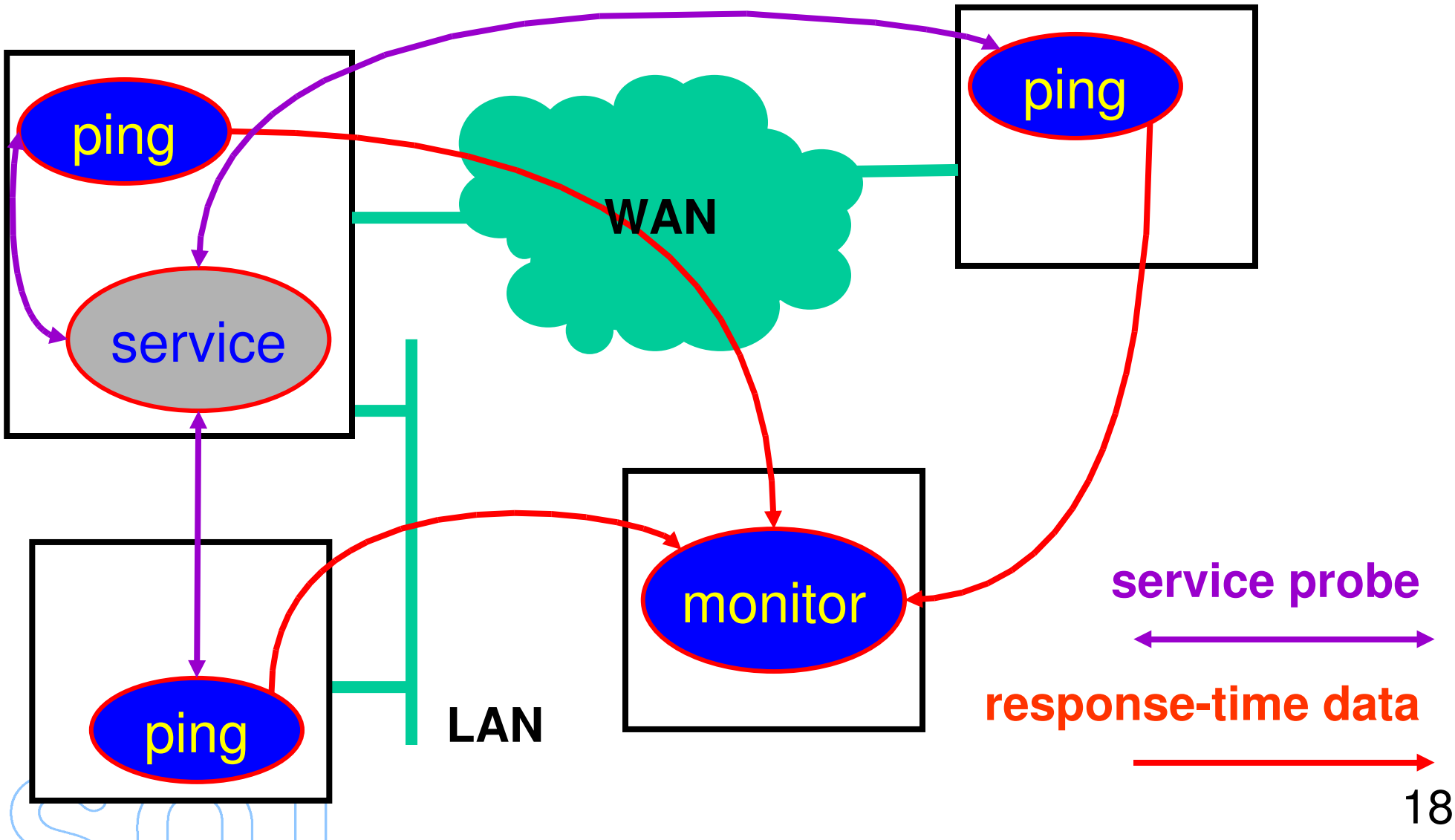
- **Single disk spindle busy (more than 60 I/Os per second)**

  some_inst
  disk.dev.total > 60
  → … arbitrary action …

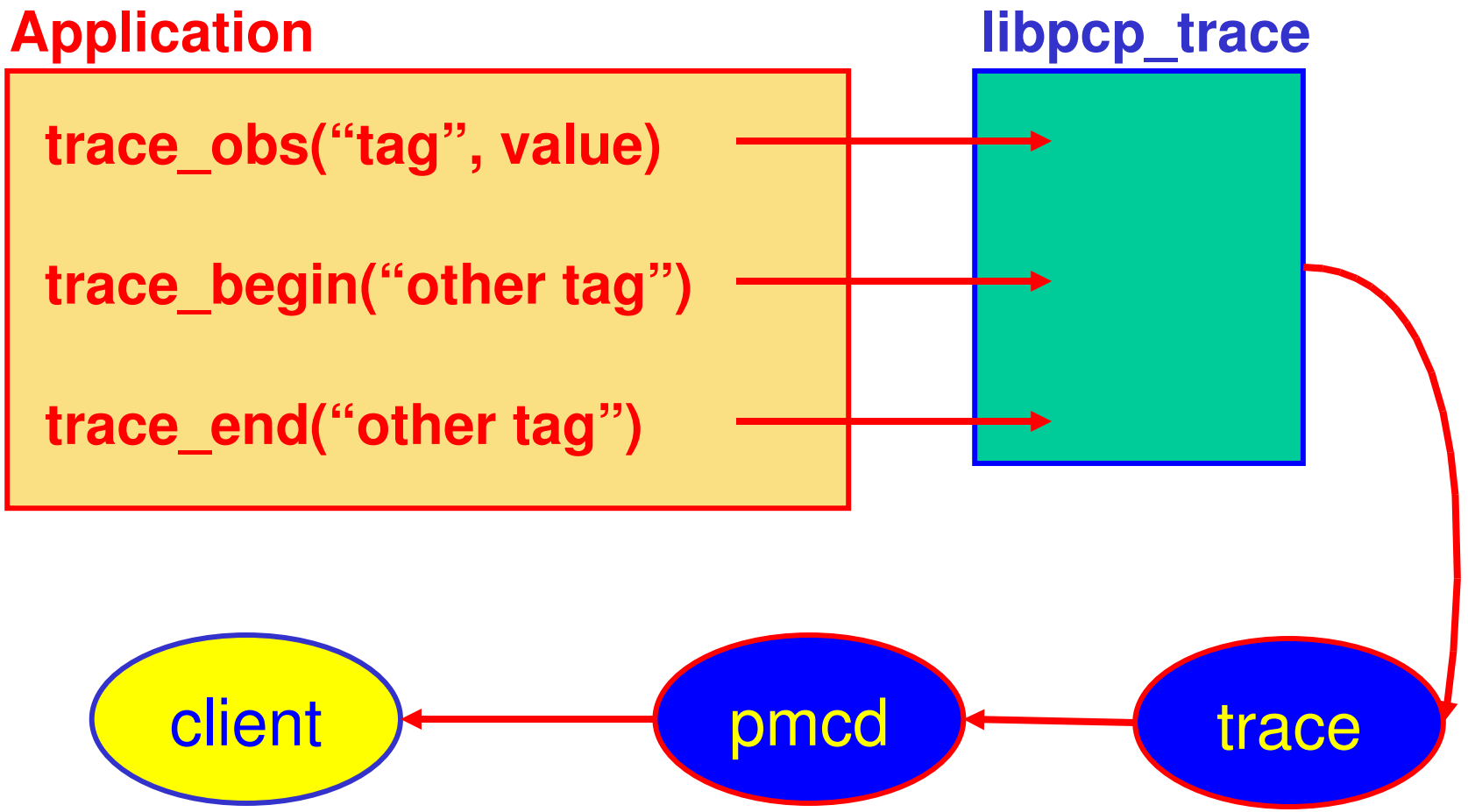
Monitoring and managing quality of service

- In many cases it is possible to emulate end-user service requests, e.g. “ping” a Web server or a DBMS or an inetd daemon.
- Construct a PMDA to periodically probe the service and measure the response time.
- Deploy the PMDA across the network and monitor the response times centrally.

Measuring end-user quality of service



Performance data from an application with libpcp_trace



Summary

- System-level performance management is a difficult undertaking.
- PCP provides an extensible framework, with a rich set of services and tools.
- Solving the hard problems requires
  - customization to collect relevant performance data
  - building customized monitoring tools
  - real-time and retrospective analysis
- PCP assists with all of these tasks.