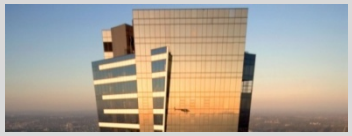






*Chatz <dchatterton@aconex.com>*
*Cowan <cowan@aconex.com>*
*Nathan <nscott@aconex.com>*

# System-Level Performance Analysis With PCP

# Welcome to Aconex!

- Scalable performance monitoring
- Performance Co-Pilot
- Brief introduction to Aconex
- PCP integration into Aconex application
- Practical examples

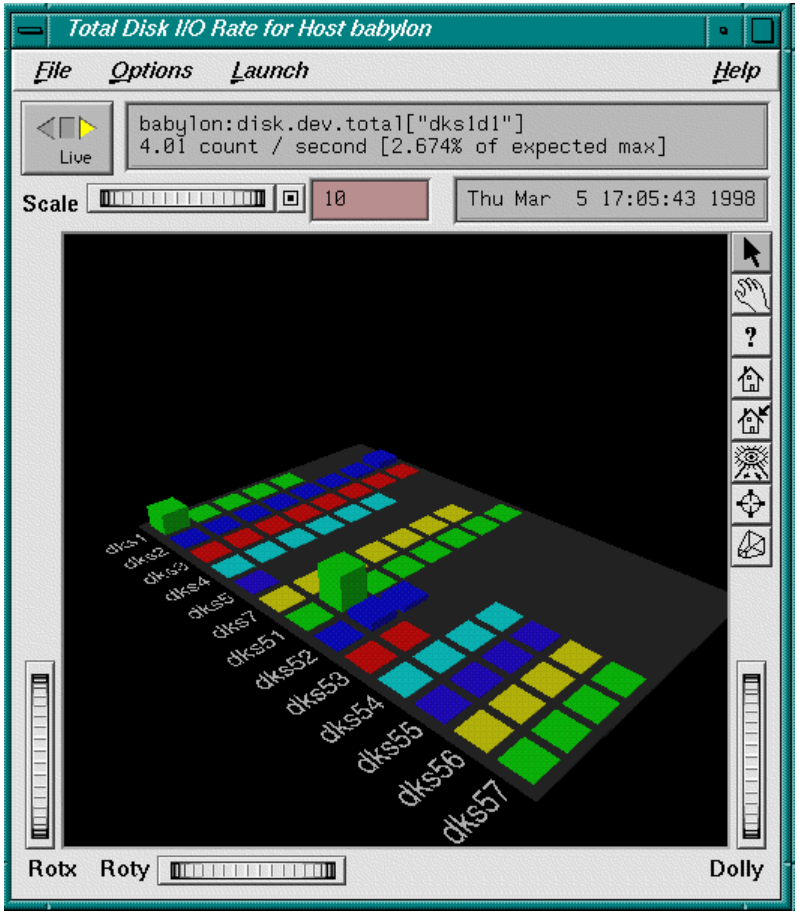
# Scalable Performance Monitoring

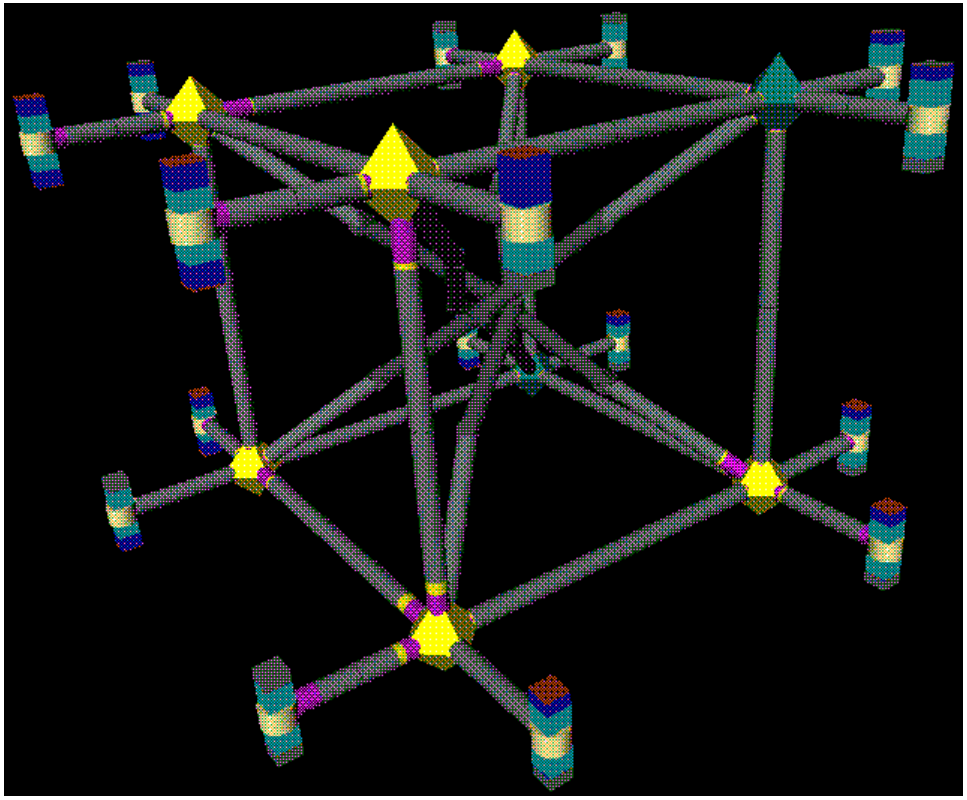
- SGI in Melbourne started developing Performance Co-Pilot 15 years ago
- Goal was to help SGI users understand very complex system performance problems
- Traditional tools
  - fail to scale to large machines
  - do not provide low level granularity
  - difficult to perform retrospective analysis
  - have poor visualisation of large data sets

aconex

# Scalable Performance Monitoring

- Many resources



- Complex architectures

aconex

# Scalable Performance Monitoring

- How do you monitor a 512 processor machine
  - top does not scale!
- How do you monitor a cluster of 20x512 processor machines?
- How do you know if what you are observing today has happened before?
- How do you correlate today's event with other activity on the system?


aconex

# Our goal tonight

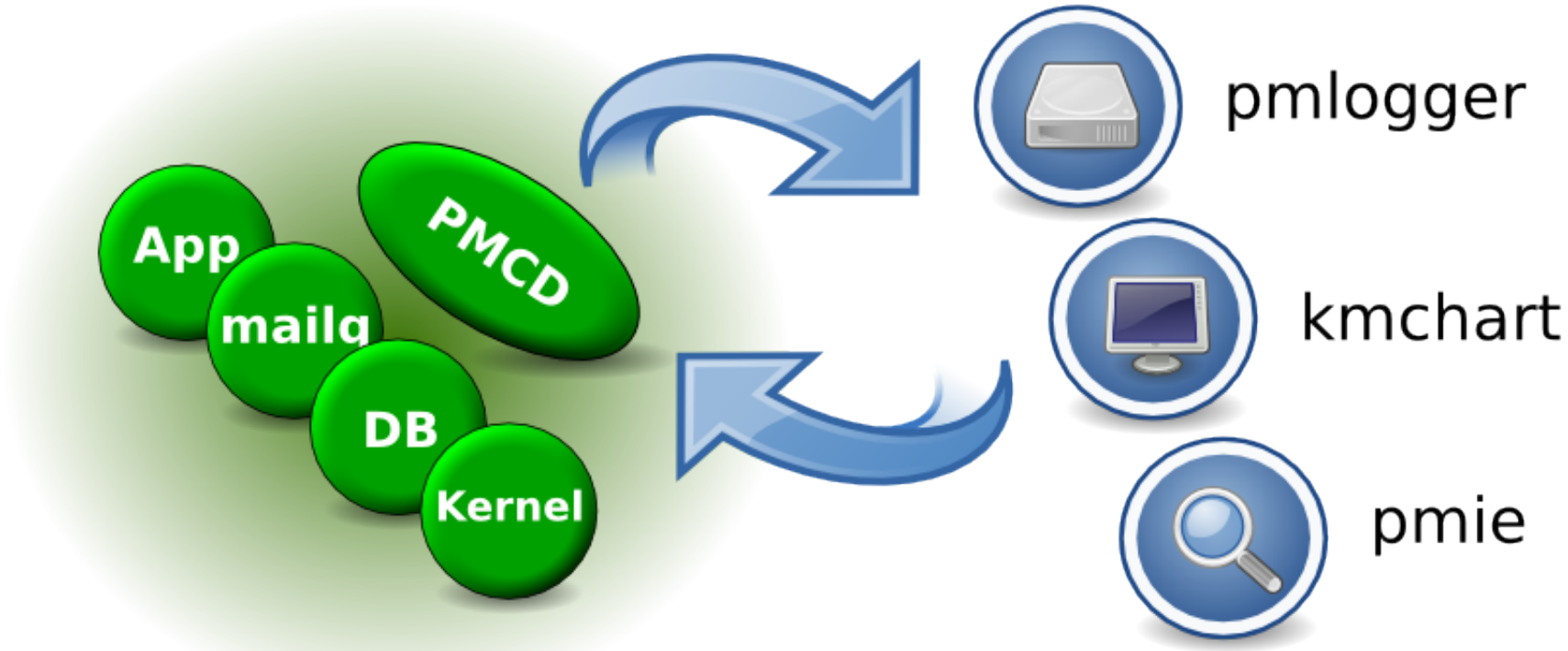
- Demonstrate how Aconex has taken an open source performance monitoring toolset and integrated it into our Java application
  - Overview of PCP and how it works
  - How it is integrated
  - Differences in monitoring a JVM to an Operating System
  - Examples of how we have used PCP in anger
- All the software you will see tonight has been developed here in Melbourne

aconex

# PCP Overview

- What is PCP?
  - Open source **toolkit** for system level performance analysis
  - Live and historical
  - Extensible (monitors, collectors)
  - Distributed

# Architecture



# Data Model

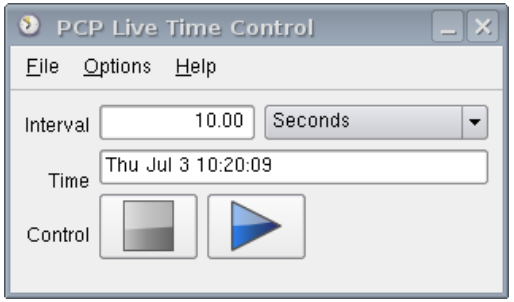
- Metrics come from one **source** (host / archive)
- Source can be queried at any interval by any monitor tool
- Hierarchical metric names

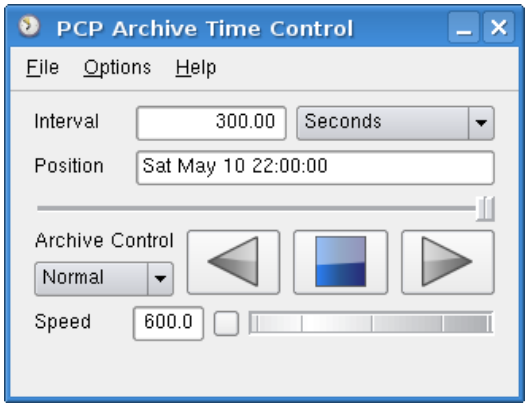
  e.g. disk.dev.read and aconex.response_time.avg
- Metrics are singular or set-valued (“instance domain”)
- Metadata associated with every metric
  - Data type (int, double, ...)
  - Data semantics (units, scale, ...)
  - Instance domain

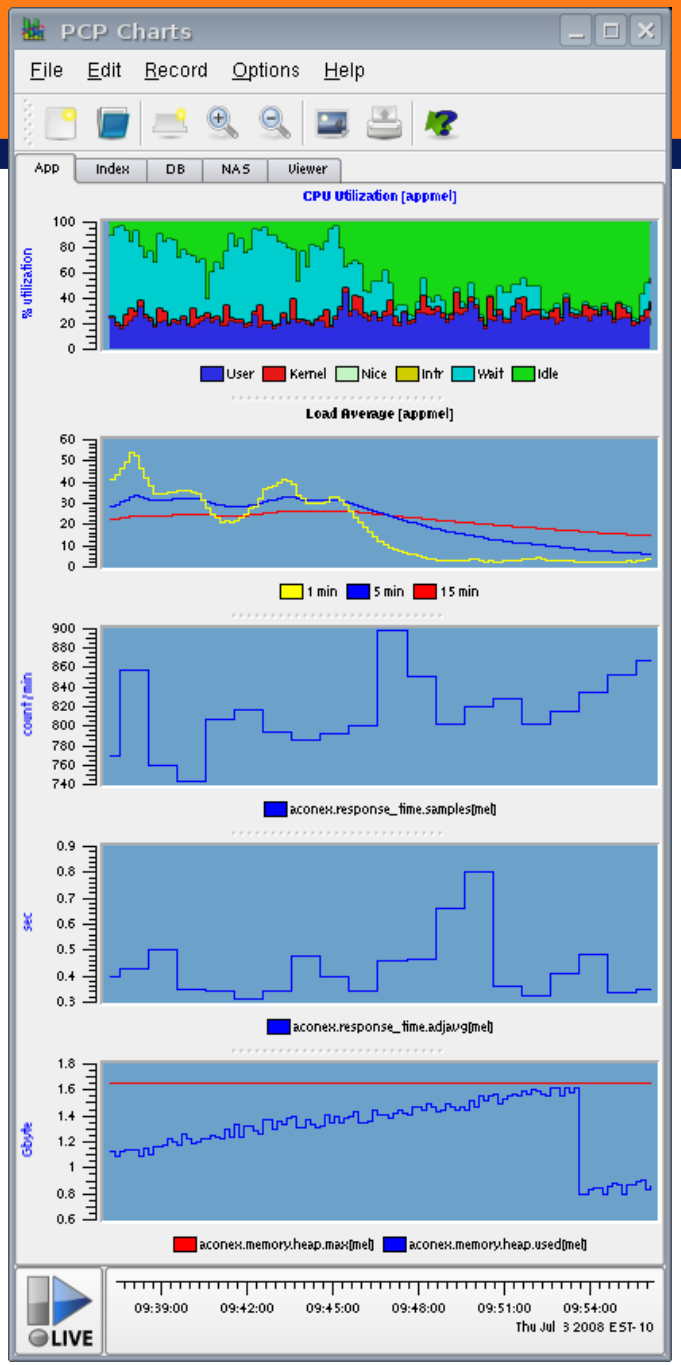
aconex

# Monitor tools

- pminfo, pmprobe, pmdumptext
- pmlogextract, pmlogsummary, pmwtf
- kmchart
- pmstat
- pmie
- acxstat
- acxtop



# Aconex Overview

- **$210bn worth of projects in 70 countries**
- **135,000 users and 3,500 client organizations**
- **300 people, 24/7 local support, in 35 cities**

- Aconex provides online information management for construction, engineering & energy
- A sophisticated workflow engine that supports established industry processes in complex, fast-moving project environments
- SaaS application delivery, one login, multi project

***Gartner (Oct 2007):***
*Collaboration market is evolving in response to demand for a coherent set of capabilities, processes and services that span communication, coordination, communities and informal social interactions.*

# Large projects need collaboration systems to operate effectively and manage risk

- Hundreds of companies come together on large projects
- Most communication is between companies
- The volume of documents and correspondence on projects is huge
- Participants on large projects are increasingly geographically dispersed
- Risk management / litigation necessitates keeping a strong audit trail of project information

*Participants:* *3,030*
*Locations:* *US, Europe, Asia, ME*
*Information:* *9.6m docs & correspondence*
*Data managed:* *4.2TB*

aconex

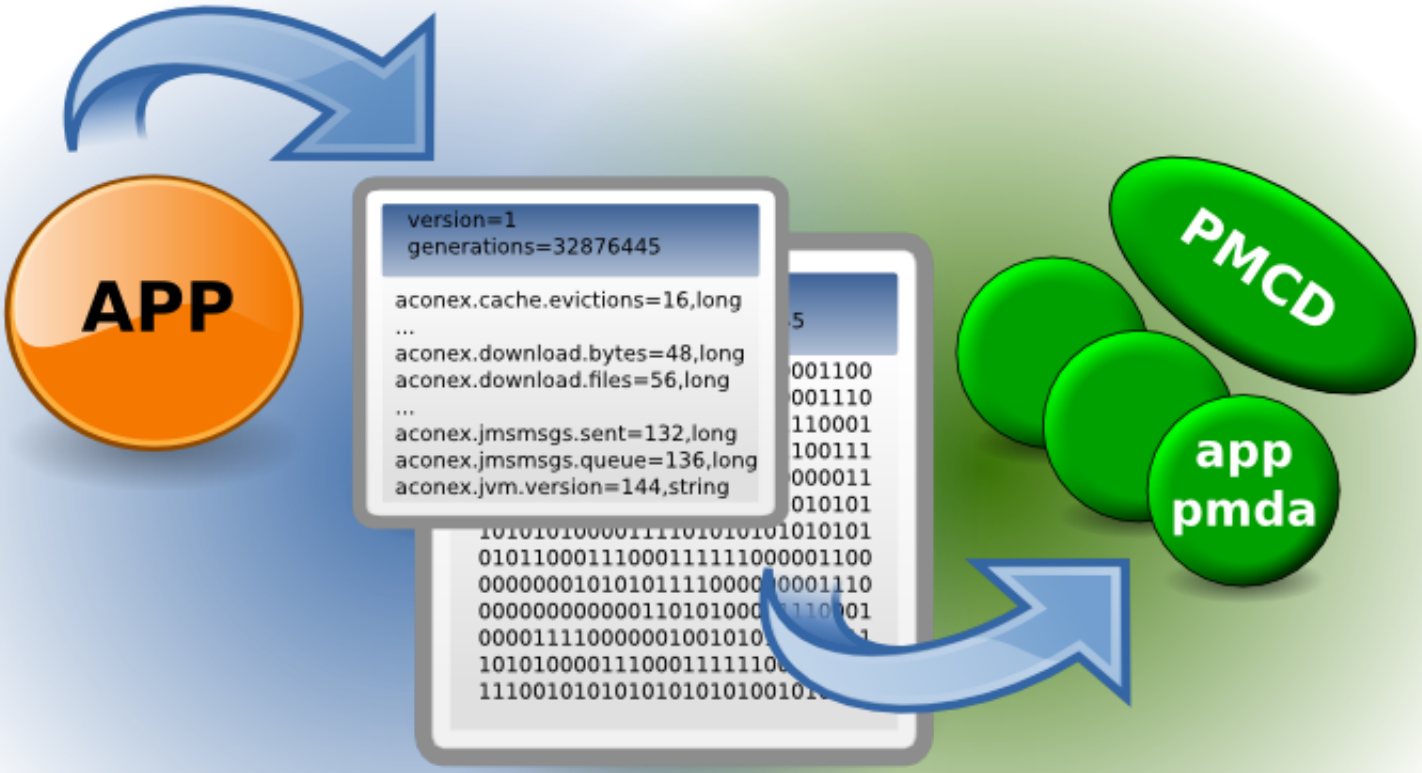
# What's hard about monitoring Java?

- Java's never standalone; interacts with
  - OS
  - Network
  - Database
- Can't monitor Java in isolation
- The JVM makes things more complicated
  - Java heap + GC means that native memory metrics are not so useful on their own
  - One more layer of indirection between everything
  - To the OS, JVM can just look like a big amorphous blob

# How do we collect the data?

- Needs to be
  - Fast
  - Low-impact
  - Not stress the subsystems we're trying to monitor!
- Basic monitoring framework is very simple + lightweight
  - Observer/Listener pattern
  - Monitor registers self with Monitorables
  - Monitorables inform Monitors when they change value
  - Most Monitorables are very lightweight
    - MonitoredCounter = wrapper around an AtomicLong, monotonically increasing (e.g. event counter, cumulative tally)
    - MonitoredValue = any sort of value, commonly AtomicLong/AtomicInteger (show current state – in-progress events)
    - Also have a polling implementation to monitor third-party systems (uses Timer to poll, updates if value changes)

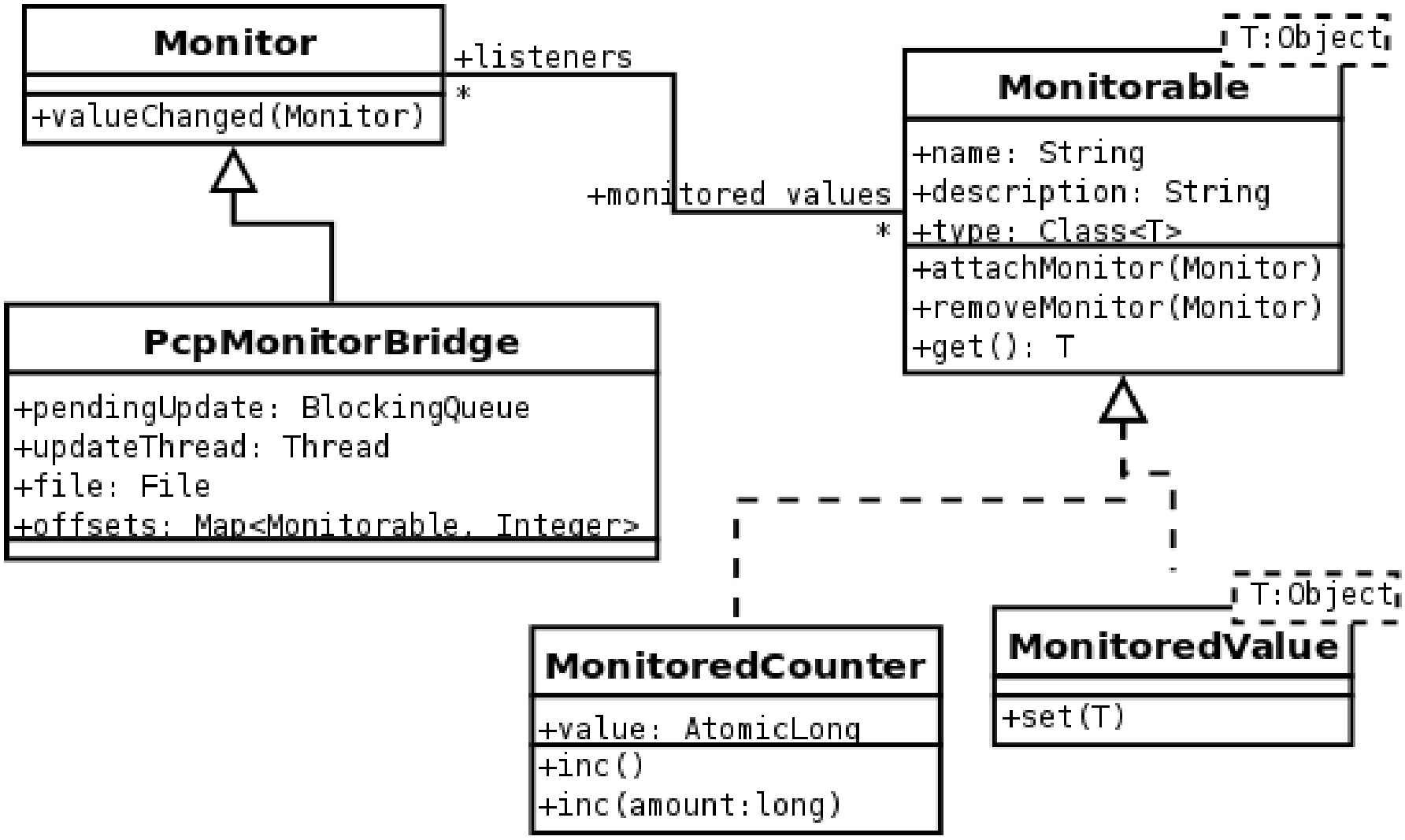
# Aconex PMDA


APP
version=1
generations=32876445
aconex.cache.evictions=16,long
...
aconex.download.bytes=48,long
aconex.download.files=56,long
...
aconex.jmsmsgs.sent=132,long
aconex.jmsmsgs.queue=136,long
aconex.jvm.version=144,string
PMCD
app
pmda

# How do we get the data to PCP?

- PcpMonitorBridge is an (the only!) implementation of Monitor
- Maintains an update queue of changed values
- Creates a memory-mapped file (NIO) which gets mapped into PCP Agent's address space
  - Shared memory between JVM and C agent (memory == fast, and no I/O-related system calls)
  - As close to 0 overhead as we'll get
- Keeps a Map of Monitorables ⇒ file offsets
- Polling thread drains queue, writing new values to file
- Changes then visible to custom PCP agent
- Is also an Mbean (CompositeData), just for good measure

# How it hangs together



# aconex-pcp-bridge lifecycle

- PCP requires a fixed set of metrics
  - Need to assemble the list of metrics before handing over to PCP
  - Cannot dynamically add metrics at runtime
- MonitorableRegistry is (ugly) static class maintaining a singleton map of identifiers to Monitorables
- Monitorables register themselves with the Registry on creation
- Starting the PcpMonitorBridge 'freezes' the Registry, no more monitorables can be created
- Dependency injection would be a lot less disgusting here (and let us have multiple Monitors)

aconex

# External metrics we collect

- System-level metrics
  - network.interface.in.bytes
  - kernel.all.cpu.sys
  - disk.all.write_bytes
  - ...
  - Do these correlate with observed events?
  - We've found JVM bugs this way...
- 3rd-party systems + software
  - pdfq.length
  - nfs4.client.reqs
  - sqlserver.locks.all.wait_time
  - ...
  - Helps assess the impact of what we're doing in the JVM on other systems (or vice-versa)

# JVM Internal metrics

- Lots of information about the JVM, and what it's doing
- Often stuff exposed through JMX anyway
  - MBean polling Monitorable implementation
- Kinda boring, but useful:
  - aconex.memory.gc.full.count
  - aconex.memory.gc.minor.time
  - aconex.memory.permgen.used
  - aconex.memory.survivor.committed
  - aconex.jvm.compilations
  - ...
- Mostly memory-related, but a few more possible (and easy)
  - Classloader info
  - Thread counts
  - etc...

# Application-level stats

- Where it gets interesting (for us!)
- If in doubt, monitor it – it costs *almost* nothing and might be useful later
  - Cumulative counters: .logins, .upload.files, .download.files
  - Instantaneous state: .sessions.active
  - Unchanging values for sanity checking/correlation: .jvm_version
  - More advanced stats: .response_time.{min, max, avg, median, adjavg, samples}
  - Monitor other libraries:
    - .jmsmsgs.{sent, queued, processed}
    - .cache.*region*.{misses, requests, evictions, load_time, ...}
  - Instrument the DB (with JDBC Driver decorator):
    - .connections.{in_use, available}
    - .jdbc.queries.{active, total}
  - ... and about a bazillion more...

aconex

# And then we get tricky...

- Implementations of InputStream and OutputStream which count bytes as they pass through
  - Much more accurate than just counting instantaneously by filesize
- SessionMonitor (normally off) – calculates sizeof(session object)
  - .sessions.bin_256KB_2MB.{count, max, mean, min, total}
- Per-controller metrics – time each request at start/end
  - .controllers.*controllername*.{count, time}
- **But** we also know that each request is served by 1 thread; can snapshot other values at start/end of request and use as counter deltas
  - .controllers.*controllername*.{utime, stime, blocked.time, blocked.count, waited.time, waited.count}

aconex

# Challenges still exist...

- Getting transparency into completely isolated third-party systems is still difficult
- e.g. one standalone 3rd-party (Java-based) system was misbehaving, needed profiling
- How do we get that JVM monitored by PCP?
  - "Cuckoo's Egg" approach
  - Custom `org.apache.log4j.Appender`
  - Hijack class <init> process to set up PCP bridge
  - Monitor JVM
  - Find problem!

aconex

# Adding a new metric

```
    private final MonitoredCounter fileRequestCounter = new
MonitoredCounter("aconex.pdf.file_requests", "Number of file
requests by the PDF server");
...
    protected ModelAndView handleRequestInternal(...)  {
        ...
        fileRequestCounter.inc();
```

- Produce new version of pmda
  - Script runs against the memory-mapped file and produces new PMDA code, which is...
  - ...compiled and released via RPM (concurrent with app release)
  - Done!

```
$ pmval -r -t 1hour -a /archives/<...> aconex.pdf.file_requests
...
09:13:22.885                    680
10:13:22.885                    689
11:13:22.885                    696
```

aconex

# Aconex PMDA

- Metric set extended on each release
  - Both backward and forwards compatible, so production is usually updated (live) well in advance of new application releases
  - Engineering aware of operational need for instrumentation, so new code is always instrumented up front now
- Scripts largely automate the PMDA code changes
  - Not completely automated
  - Allows extra review and sanity checking :-)
- What to instrument?
  - Quailty of Service - application response time! throughput
  - Resource utilisation (esp. time) – CPU, I/O, disk
  - Activity (bytes, files transferred, emails sent)
  - Problem parameters – queue length, size of file

aconex

# Future directions

- We *really* want to open-source this work!
  - Monitoring framework (counters, values, etc)
  - Utility classes (wrapping Input/OutputStreams, JDBC driver, MBean bridge)
  - It's with the lawyers! We'll keep AJUG updated...
- Tidy up and make more generic
  - Custom controller metrics → generic (single-thread) event timing
- More detailed interaction metrics (per-thread if possible)
  - Time spent waiting for other servers, DB locks taken, etc
- Architecture tidy-up (isn't there always?) -- e.g. get rid of (or clean up) Registry ickiness
- Could easily extend to other Monitors
  - Expose to JMX directly?
  - Windows perfmon Monitor?

aconex

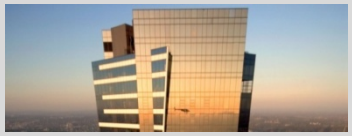
# Final steps

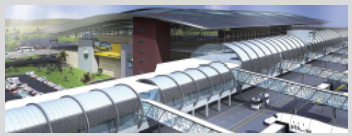
- pmie and Nagios – production alarms
  - *notifications = some_inst ( aconex.notification.jobs_failed > 0 )*
    *-> shell "send_mail Notification jobs failed: %v";*
  - Archives used to verify rules
  - Catch unusual resource utilisation, error conditions, and poor quality of service
- pmie and pin-point monitoring
  - Automate generation of threaddumps, heap capture, profiling
- kmchart – live and historical production monitoring
- pmdumptext / acxstat – ditto, console tools
- Benchmarking, new system configuration
- Planning for the future

aconex

# Further Information

- *http://www.aconex.com [ /Careers.html ]*
- *http://oss.sgi.com/projects/pcp*
- *http://www.ohloh.net/projects/pcp*
- *http://techpubs.sgi.com/ [ PCP Programmers Guide ]*

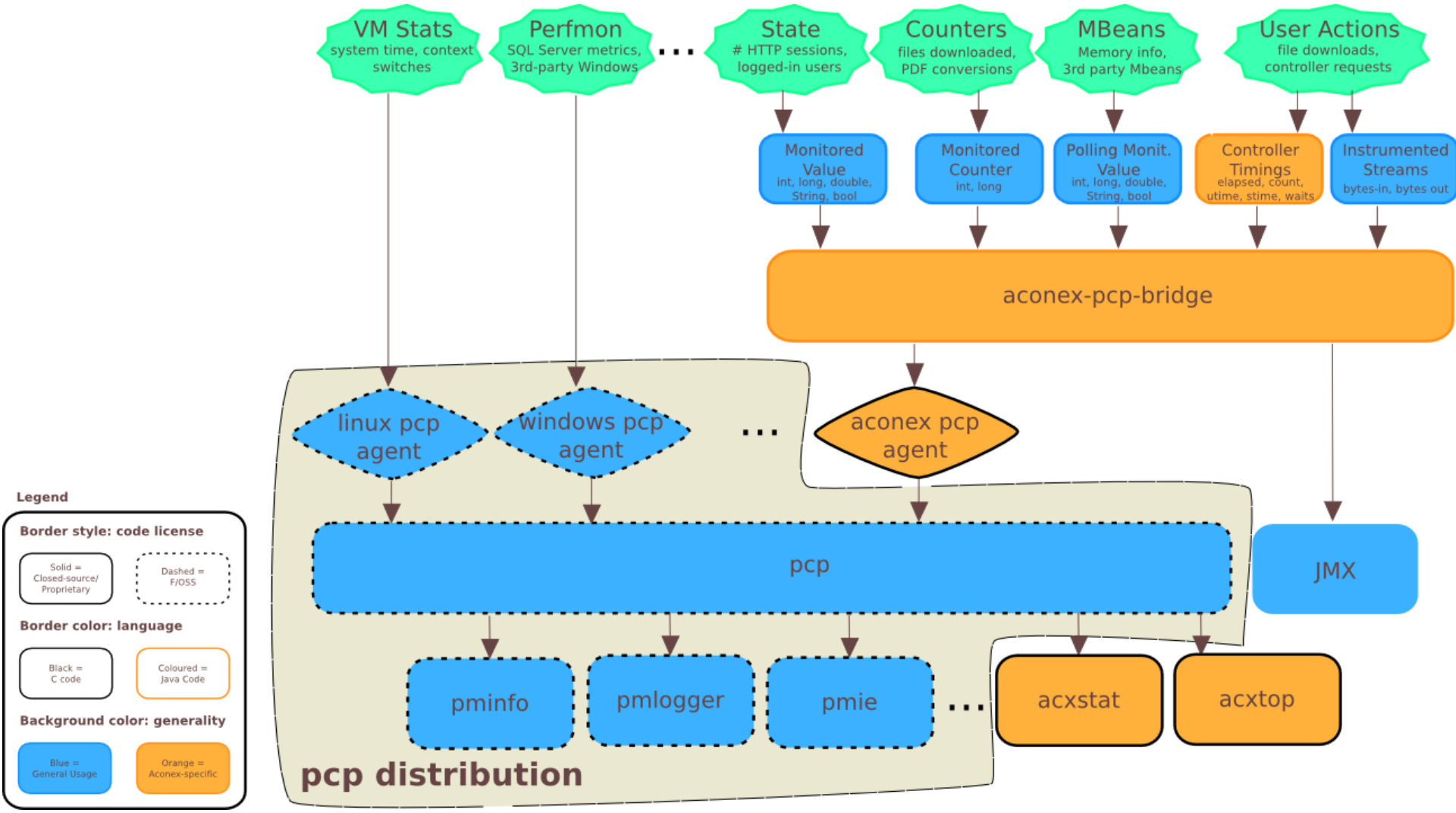



# Thank you

**Abu Dhabi Investment House** Faithful & Gould **Multi Development** Aedas **Foster and Partners** Multiplex **AJES** Gammon **Nishimatsu** Arif and Bintoak **GHD Global** NRY Architect **Atkins** Grocon **Ove Arup & Partners** Australand **Hassell** Page Kirkland Partnership **Bates Smart** Hongkong Land **Robert Bird & Partners** Baulderstone Hornibrook **Hsin Chong** ROK **Buchan Group** HWO Architects **Taisei** Capita Symonds **ING Real Estate** The Cox Group **Capitaland** Itochu **Turner & Townsend** Clifton Coney Group **Kajima** Vietnam Land SSG **Confluence** Laing O'Rourke **Wates Group** Cyril Sweett **Larsen and Toubro** WDA Architects **Davis Langdon & Seah** Las Vegas Sands **Woods Bagot** Dragages **Levett and Bailey** Woolworths **Dubai Sports City** LTW **WSP Group** Dubai Metro **Mainzeal** WT Partnership **Emirates Sunland** MKM Commercial Holdings **Airport Authority Hong Kong** InterContinental Hotels **Qantas** Australia Pacific Airport Management **Macquarie Bank** Ritz Carlton **Building Schools for the Future** Mandarin Oriental Hotels **Sydney Opera House** Colonial First State Property **McDonald's Restaurants** Taj Hotels Resorts and Palaces **DFS Group Melco Hotels And Resorts** Tradewinds Hotels and Resorts **Hyatt International** Ocean Park Corporation **Venus Assets** Al Habtoor **Connell Wagner** Nui Phao Mining **Alcoa** Earth Tech Engineering **Mitsubishi Corporation** Alinta **Evans & Peck** Mott Connell **Babcock & Brown** Global Offshore International **OMV** BHP Billiton **Global Process Systems** Parsons Brinckerhoff **Bilfinger Berger** Hitachi **Penta Ocean** Bluescope Steel **Hyder Consulting** Straits Hillgrove **British Gas India** Jindal Steel & Power **Tanker Pacific Offshore** Bulga Coal **Maunsell** Tiberon Minerals **ChemOil** McConnell Dowell **Waterman-Gore** CLP Engineering **Meinhardt** WorleyParsons **Connell Mott MacDonald** MinCraft Consulting **Xstrata Coal** Grocon **GHD Global** NRY Architect **Atkins** Grocon **Ove Arup & Partners** Australand **Hassell** Page Kirkland Partnership **Bates Smart** Hongkong Land **Robert Bird & Partners** Baulderstone Hornibrook **Hsin Chong** ROK **Buchan Group** HWO Architects **Taisei** Capita Symonds **ING Real Estate** The Cox Group **Capitaland** Itochu **Turner & Townsend** Clifton Coney Group **Kajima** Vietnam Land SSG **Confluence** Laing O'Rourke **Wates Group** Cyril Sweett **Larsen and Toubro** WDA Architects **Davis Langdon & Seah** Las Vegas Sands **Woods Bagot** Dragages **Levett and Bailey** Woolworths **Dubai Sports City** LTW **WSP Group** Dubai Metro **Mainzeal** WT Partnership **Emirates Sunland** MKM Commercial Holdings **Airport Authority Hong Kong** InterContinental Hotels **Qantas** Australia Pacific Airport Management **Macquarie Bank** Ritz Carlton **Building Schools for the Future** Mandarin Oriental Hotels **Sydney Opera House** Colonial First State Property **McDonald's Restaurants** Aedas **Foster and Partners** Multiplex **AJES** Gammon **Nishimatsu** Arif and Bintoak **GHD Global** NRY Architect **Atkins** Grocon **Ove Arup & Partners** Australand **Hassell** Page Kirkland Partnership **Bates Smart** Hongkong Land **Robert Bird & Partners** Baulderstone Hornibrook **Hsin Chong** ROK **Buchan Group** HWO Architects **Taisei** Capita Symonds **ING Real Estate** The Cox Group **Capitaland** Itochu **Turner & Townsend** Clifton Coney

# Appendix 1 – The present



# Appendix 2 – The future