Advanced Monitor/Subscription Mechanisms for EPICS

Ralph Lange, Helmholtz-Zentrum Berlin / BESSY II, 12489 Berlin, Germany
Andrew Johnson, Argonne National Laboratory, Argonne, IL 60439, USA
Leo Dalesio, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

Publish/subscribe systems need to handle the possibility that there are subscribers requiring notification at an update rate much lower than the publisher's natural frequency, or synchronized to external events. Feedback or pulse-to-pulse diagnostics are processed at rates in the 100 Hz or even multi kHz range, while many subscribers will not be able to process the data at this rate: e.g. archiving, visualization, and processing clients each require specific, different update rates. Sending more updates than required wastes processor and network bandwidth. A subscriber should be able to specify rate limiting factors or filters that are instantiated and guaranteed by the publisher. Many accelerators, especially pulsed machines, are using a hardware event system to distribute fiducials and events from a central event and/or frequency generator. These events should be integrated into the publish/subscribe system to support posting event synchronous updates to subscribers that require synchronized data. This paper investigates several approaches to provide these functionalities in the EPICS architecture.

Motivation

- Data processing rates have been drastically increasing
- Hardware event and timing systems are widely used

Variable update rates
Only few clients want updates at full rate

Event related updates
Clients want updates only during a certain time slot

Current Limitations

Update rates are fixed
"Data", "Archive", "Alarm Status" types
Configuration (dead bands for analog values) is done at the database level, for all clients

Event Correlation is limited
Events cause processing or provide time stamps
No filtering on event-derived system state for updates on unrelated records

Current Event Updates

One ***TCP connection*** per client

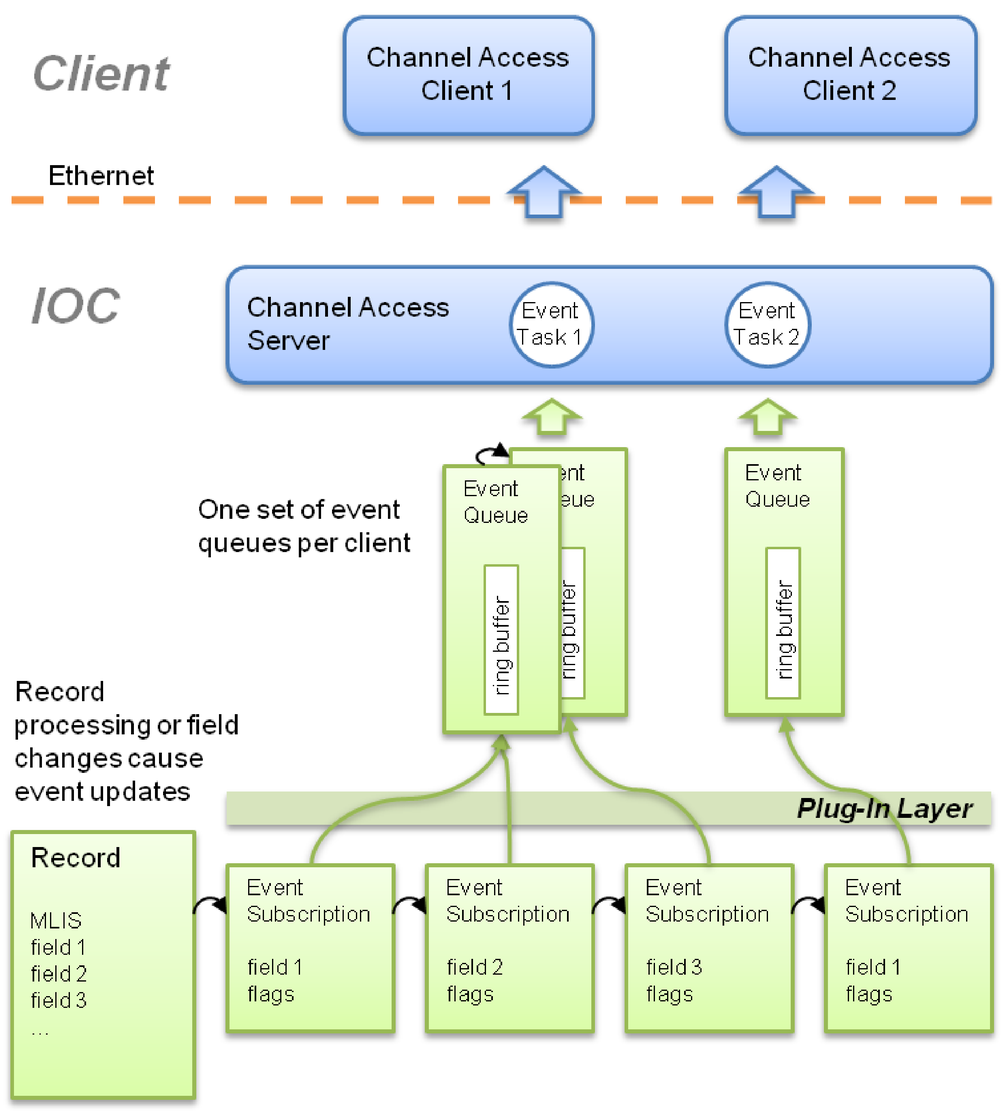
One ***event task*** ships out updates to the client

One set of ***event queues*** per client, buffering values when events are sent fast

List of ***event subscriptions***, one per client and subscription, linked to record

Record ***processing or field change*** walks through the list of event subscriptions, and puts update in the event queue on a match

Server-Side Plug-Ins



Mechanism

Stackable plug-ins are inserted between event subscriptions and event queues when the connection is made

Configuration

Clients can add ***JSON modifiers*** with plug-in configuration when they subscribe

Plug-in parses configuration when instantiated with subscription

Possibilities

Update rate limiting
Client specifies minimum/maximum rate

Update correlation
Clients specifies event-derived system state for updates being sent

Event data filtering
Sliding average, rate-of-change filters

Buffering of array data

… and many more ...

Support Libraries

Event correlator
Drivers can set or reset a system state, which will be recorded (with time stamps)
Plug-ins can check if record was processed during a certain state

Memory allocator
Efficient universal cache-based free list allocation for plug-ins
Implementations might use existing SLAB, SLUB or SLOB libraries

Project Status

Currently in design phase

Part of a 1-year effort to add functionality needed by NSLS-II, expected to be implemented during that time

First plug-ins will include update rate limiting and update correlation to system states

Work supported by U.S. Department of Energy (under contracts DE-AC02-06CH11357 resp. DE-AC02-98CH10886), German Bundesministerium für Bildung und Forschung and Land Berlin