CAFlux: A New EPICS Channel Archiver System*

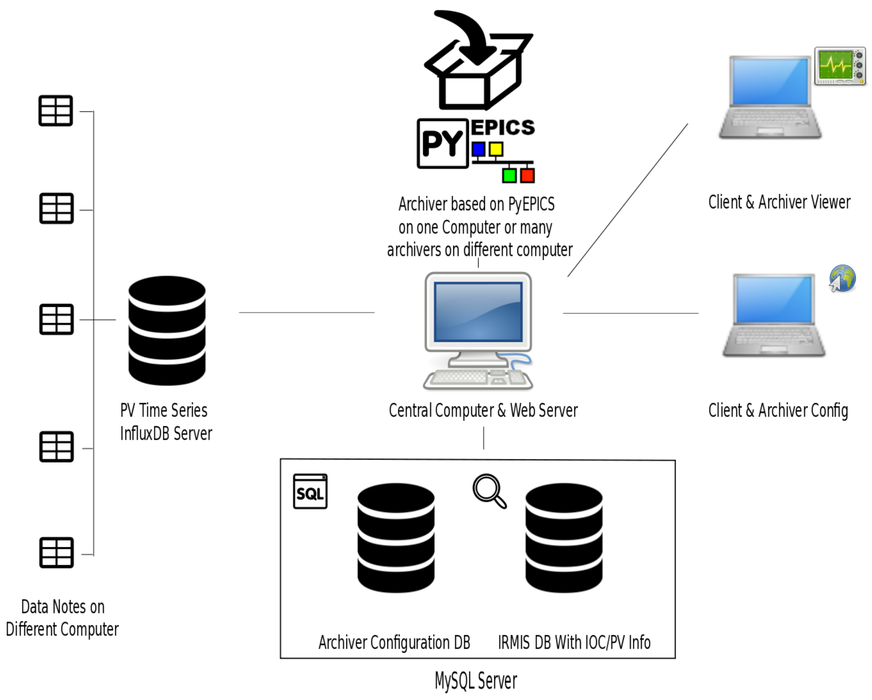
Kanglin Xu

Accelerator Operations and Technology - Instrumentation & Controls Group
Los Alamos National Laboratory
ICALEPCS 2019, Oct 7-11

Introduction

Different from the legacy archiver system, the *CAFlux* archiver system is built on *InfluxDB* database. By replacing the index and data file system with *InfluxDB* system, we have a more robust, compact and stable archiving storage. The multitier data collection engine is implemented by combining *Python* asynchronous programming and multithreaded programming. Both GUI and web applications for data retrieval and visualization have also been developed.

The CAFlux Architecture and Subsystems



A Diagram of CAFlux and Its Subsystems

- *A Data Collection Engine* is developed with *Python* asynchronous programming and multithreaded programming based on PyEPICS and EPICS channel access.
- *InfluxDB Server* is used as a data storage engine. InfluxDB is a database system optimized for storage and retrieval of time series data. It supports a few hundred nodes initially and is able to scale to a few thousand nodes. Its single-node edition is free but the clustering one is sold as closed-source.
- *HTTP Web Server* as a logic tier provides a web service for archiver configuration, data visualization and data stream.
- *MySQL Server* as another data tier holds all the user configuration information.

CAFlux Data Collection Engine

- The Data Collection Engine is designed as a multitier architecture, i.e. a lower level engine, a upper level manager and at the highest level,a monitor.
  - Lower level engine for core jobs: reading configurations, collecting and caching data, and saving data.
  - Lower level engine designed as a daemon and developed with the Python asyncio and threads module.
  - Upper level manager for monitoring the low level engine: checking the PID file, restarting the lower level daemon process if it is dead or in zombie status, logging error messages and sending emails if any issue occurs.
  - The upper level manager designed and implemented as simple as possible in order to make it more robust and stable enough to run 24 hours a day and 7 days a week with low probability for any issues.
  - The highest monitor is a simple script scheduled by *LINUX Cron Daemon* to run every 10 minutes to check the status of the running processes. When a process is dead, it will restart it and send an email notice.
- This engine is developed with combination of *Python* asynchronous programming and multithreaded programming. In this way, we can
  - circumvent complex implementation for synchronizing a large amount of threads;
  - still have enough working threads to handle a large amount of records with high writing frequency.

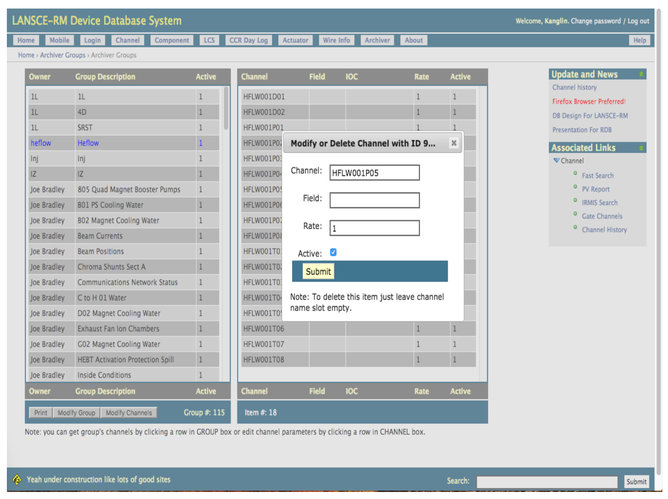
Threads and Asynchronous Task Loop

1. The main thread:
   - To read inputs, initialize global data containers and start up
   - To initialize CA library and create CA context
   - To split new work threads
   - To include an asynchronous task to check and log the health status of each work thread every few minutes and sleep the rest of the time.
2. Each work thread:
   - Have an asynchronous task loop to handle hundreds of records in a "*parallel*" manner including (1) get a record value from its cache and write it to the storage; (2) set a lower priority for a channel if its status is found to be disconnected, and repeat those steps at a predefined rate.

Implementation based on PyEPICS Channel Access

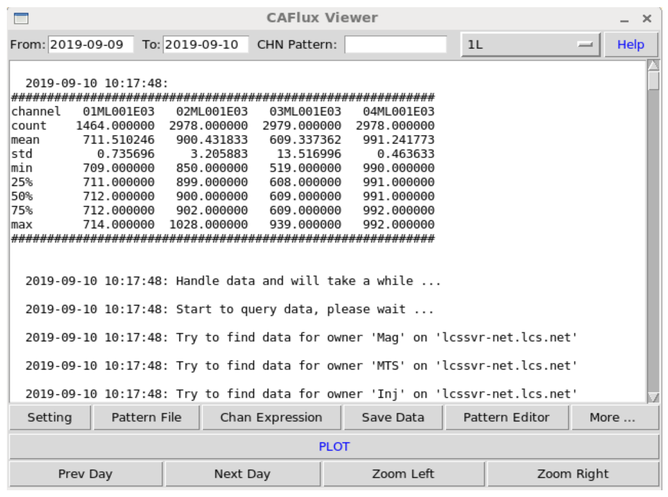
(1) Creating a channel for every PV; **(2)** Subscribing a connection callback function to CA and called by CA whenever a channel connection status changes; **(3)** Subscribing a PV callback function to CA to monitor PV values and called by CA whenever PV value changes; **(4)** The PV central caches updated by the PV callback function to cache the current value on the data collection engine.

CAFlux Web Based Configuration and Viewers



1) CAFlux configuration and management

2) Online data retrieval and data stream



3) CAFlux viewer

(1) The *CAFlux* configuration and management system is a 3-tier web application. The front-end uses *JQuery AJAX* and *JQuery UI* for a dynamic interface and web forms. The back-end implements *GET* and *POST* methods to handle requests and send data. The database is hosted on *MySQL* server. **(2)** The online data retrieval and stream is also a web application. Its front-end is developed with *Javascript* Graphing Library and *JQuery AJAX* while the back-end is built with *Python Django* to query data from *InfluxDB* data storage and response to requests. **(3)** The CAFlux viewer application with functionality like basic data statistics and simple arithmetic for record values is also available.

Conclusions and Acknowledgments

- Built on *InfluxDB* system, CAFlux has a safer, more compact and faster storage and is currently running continuously at **Los Alamos Neutron Science Center** of **Los Alamos National Laboratory**.
- A balance is achieved between archiver scalability and complexity with *Python* multithreaded and asynchronous programming.
- *PyEPICS* CA module is used in CAFlux data collection engine.

* Supported by the US Department of Energy, Los Alamos National Laboratory. Managed by Triad National Security, LLC for the DOE National Security Administration (Contract 89233218CNA000001).