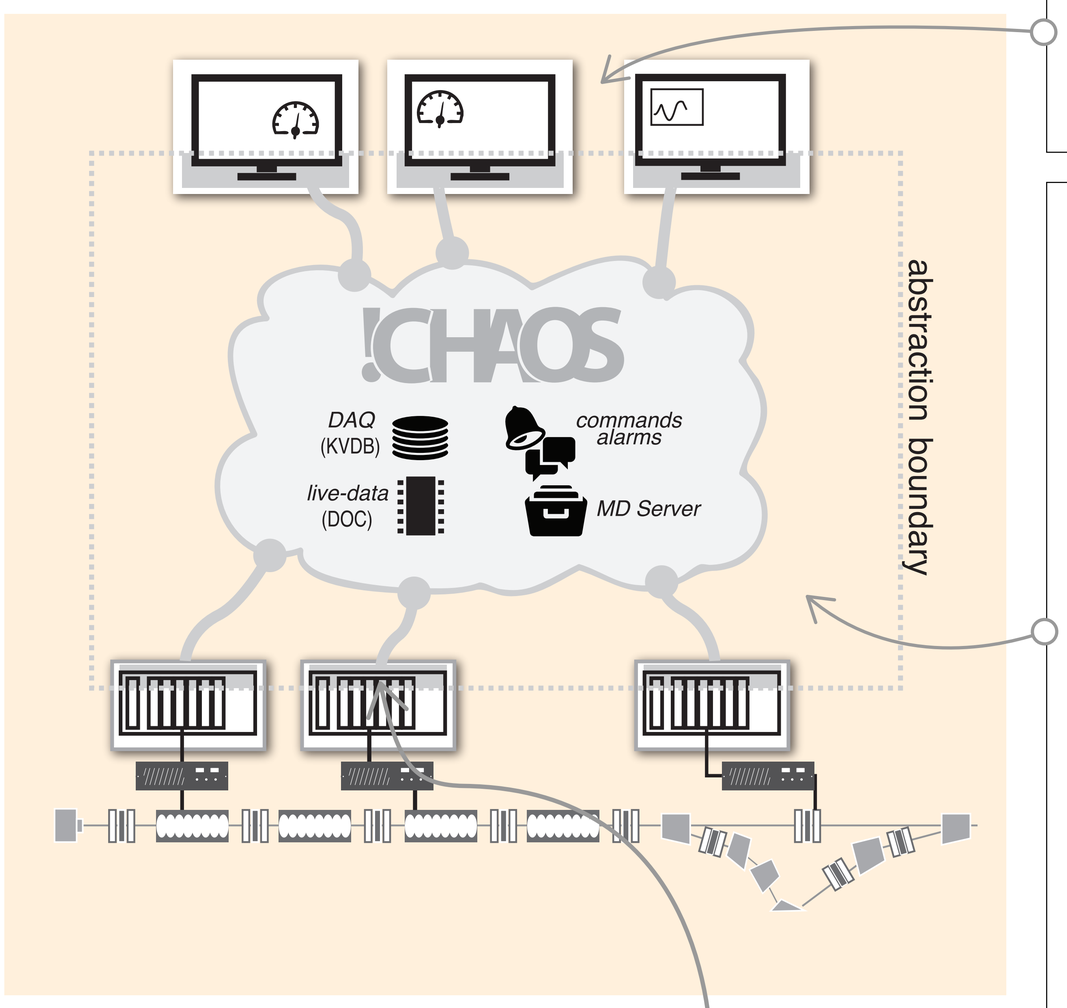
!CHAOS: the "Control Server" Framework for Controls

L. Catani, INFN-Roma Tor Vergata, Roma, Italy
F. Antonucci, C. Bisegni, A. Capozzi, G. Di Pirro, L.G. Foggetta, F. Iesu, N. Licheri, M. Mara, G. Mazzitelli, A. Stecchi
INFN/LNF, Frascati (RM), Italy

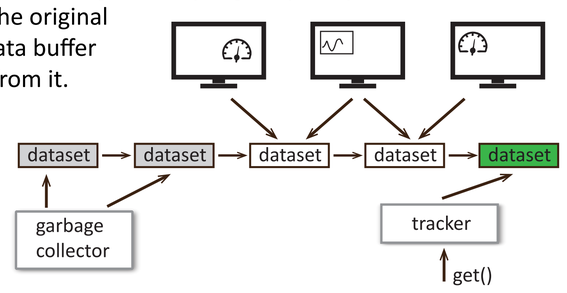
We report on the progress of !CHAOS, a framework for the development of control and data acquisition services for particle accelerators and large experimental apparatuses. !CHAOS introduces to the world of controls a new approach for designing and implementing communications and data distribution among components and for providing the middle-layer services for a control system. Based on software technologies borrowed from high-performance Internet services !CHAOS offers, by using a centralized highly-scalable cloud-like approach, all the services needed for controlling and managing a large infrastructure. It includes a number of peculiar features such as high abstraction of services, devices and data, easy and modular customization, extensive data caching for enhancing performances, integration of all services in a common framework. Since the !CHAOS conceptual design was presented two years ago the INFN group have been working on the implementations of services and components of the software framework. Most of them have been completed and tested for evaluating performance and reliability. Some services are already installed and operational in experimental facilities at LNF.

!CHAOS keywords: scalability, abstraction, integration of services, distributed object caching, non-relational key/value database, binary serialization, object relational mapping, memcached, mongodb, msgpack, BSON....

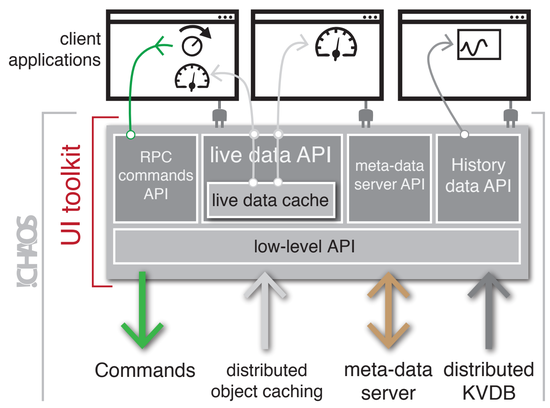


We studied the possibility to introduce a circular buffer for sharing data among client applications. As an example, consider a GUI with a graph showing the last N values of a certain variable. When the GUI starts to fetch data from the KVDB, he User Interface Toolkit underneath allocates a lock-free circular buffer and provides the pointer to the buffer to the graph's manager (see Fig.6). The buffer, with size equal to the depth of the graph, is updated by the tracker taking into account the refresh rate of the device and the sampling of the graph.

If another GUI panel or application needs the same data or a part of it, e.g. the most recent value of the buffer or a downsampling of data, the User Interface Toolkit instead of opening another stream of data uses the buffered data to feed the second client. Instead, if the second client needs a longer buffer or higher sampling frequency the buffer is resized accordingly and the original "owner" of the data buffer will extract data from it.

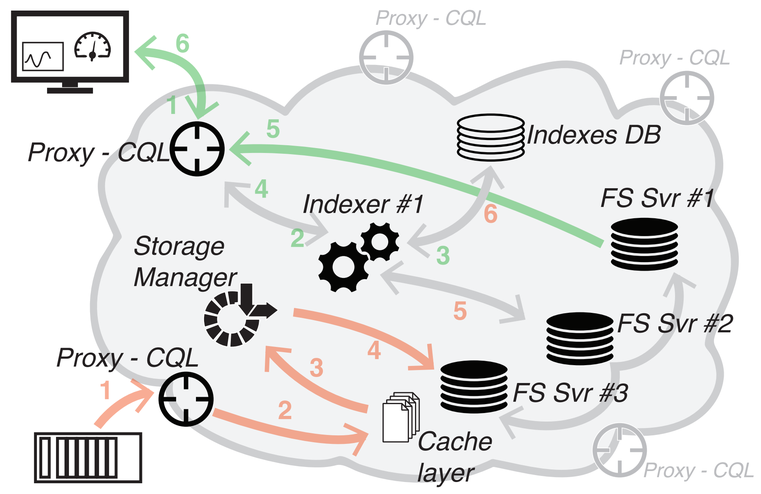


UI Toolkit

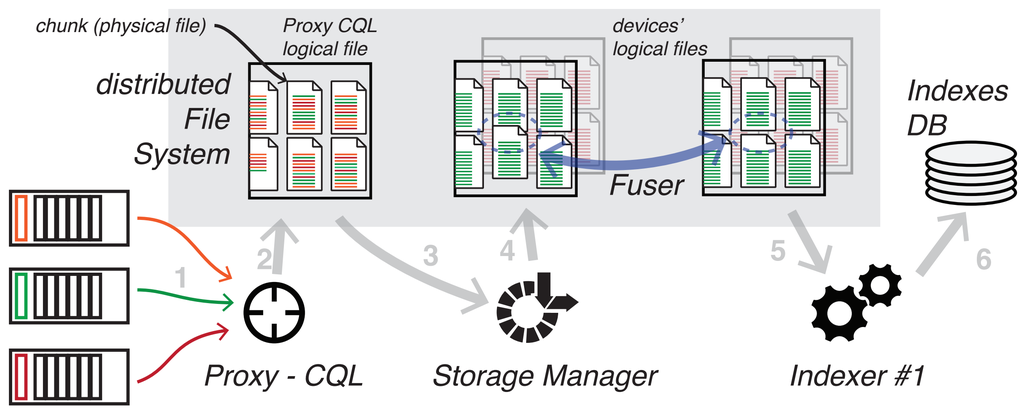


DAQ

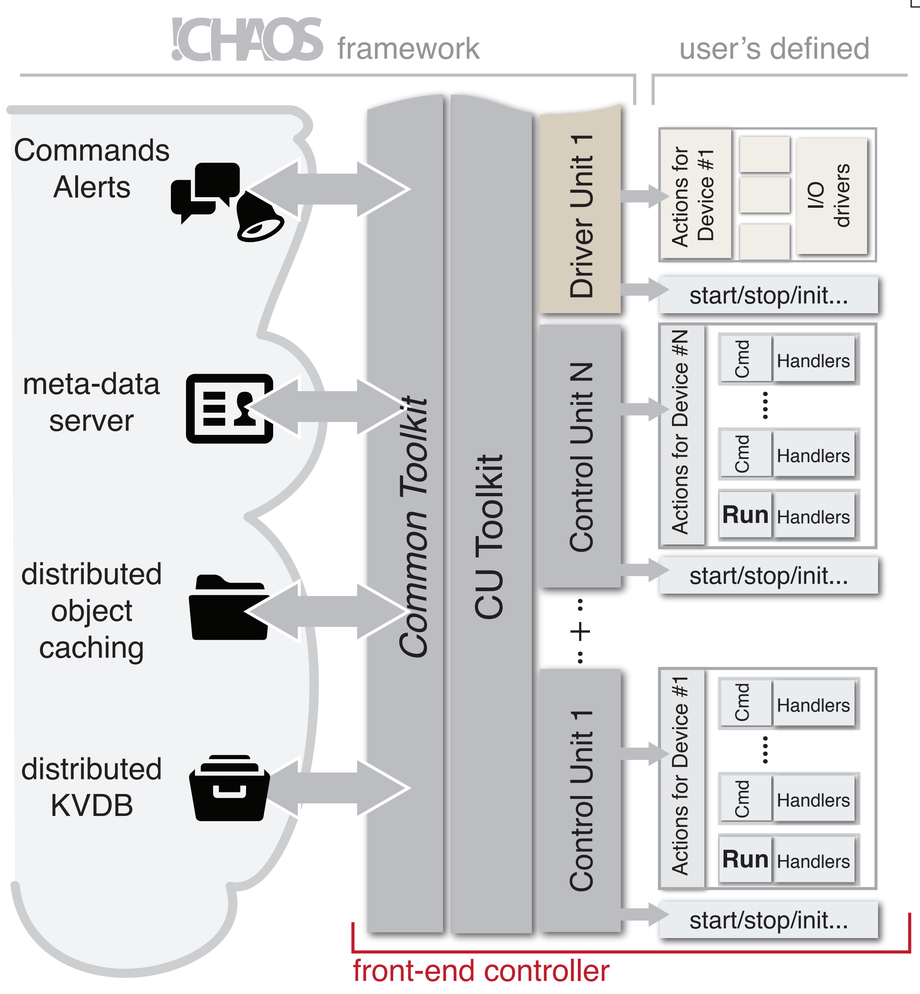
In !CHAOS the DAQ, i.e. the machine data acquisition system, is provided by the service we call History (HST) Engine. A distributed file system (FS) is used to store data produced by machine operations while a KVDB manages the indexes structure; candidates are Hadoop and MongoDB respectively. The functionalities of !CHAOS HST Engine are allocated to three dedicated components, or nodes, namely the CQL Proxy (where CQL stands for CHAOS Query Language), the Indexer and the Storage Manager.
Figure shows the data flow and the role of the before before mentioned nodes in data writing (red) and reading/querying operations. Grey lines are used to indicate internal actions and data flow.



For each CQL Proxy a logical path is created in the distributed FS (see Fig.5). At the same time, each proxy launches a pool of threads having the only task of allocating (2) the packets received by the proxy into files within the logical path associated to the Proxy, regardless from the device that sent the packet.



Control Library, Control Units and Device Management Plug-ins



Control Unit is the "container" for a device's front-end controls. Beside the standard methods: INIT, START, STOP and DEINIT, CU specialization is provided by the device's dataset, the actions implemented for it (e.g. commands and control loop) and the drivers for I/O components (see Fig. on the left).

At startup (INIT) the CU receives from the Meta Data Server (MDS) the dataset of the accelerator's device (equipment, diagnostic, sub-systems, etc.) assigned to that unit and the related actions. The default action, i.e. RUN, implements the accelerator's device control loop; other actions implement the commands for modifying the working state of the device or for executing more complex procedures.

In !CHAOS, I/O modules drivers are managed by CU-like components called Driver Unit. Similarly to CU they are specialized for a particular I/O module, or more in general a I/O service, by means of the module's Dataset including specifications such us the IP address and port for a network device or the Controller address and Node number for CAN units etc., and by specific management software.
Dataset also includes unique identifiers for each I/O channel of the module (e.g. each input channel of an ADC module).
At startup the Driver Unit initialize the I/O module and starts populating the embedded key/value shared memory with default values of its I/O channels.

At START the Driver Unit begins looping on the module I/O operations, triggered by HD or SW timing signals and refreshes the values in the share memory.

The unique identifier, defined for each I/O channel of any I/O module, is the only link between the user of the data and the producer of the data. When a new accelerator's device is added to the control system, the programmer searches the database for the I/O modules used by this device and associates to the I/O variables the unique identifier of the channel it uses.