ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEC5

!CHAOS GENERAL STATUS REPORT

Alessandro Stecchi, Claudio Bisegni, Paolo Ciuffetti, Antonio De Santis, Giampiero Di Pirro, Alessandro D'Uffizi, Francesco Galletti, Riccardo Gargana, Andrea Michelotti, Massimo Pistoni, Dario Spigone, INFN/LNF, Frascati, Rome, Italy
Luciano Catani, INFN - Roma Tor Vergata, Rome, Italy

Abstract

!CHAOS[1] (Control system based on Highly Abstracted and Open Structure) is now mature and is being employed in real operational contexts. A dedicated infrastructure, recently installed at the LNF Computer Centre, houses the framework and provides control services to different LNF installations. The !CHAOS native capability of fast storage, based on the use of a nonrelational database, has been finalized and tested with applications demanding high bandwidth. Thanks to its scalable design, the fast storage allows to accommodate multiple sources with sub millisecond timing. The EU (Execution Unit) node has also been delivered and turned out to be a "Swiss Army knife" for processing both live and stored data, inserting feedbacks and in general for correlating data acquired by the CU (Control Units) nodes. A key feature of the EU is a plugin mechanism that allows to easily integrate different programming and scripting languages such as LUA, C++, Python, also exploiting the ROOT framework, the well known scientific tool from CERN. A comprehensive description of the !CHAOS evolution, of its performances and of its use, both in scientific and industrial contexts, is presented.

INTRODUCTION

The !CHAOS project started with the ambition to create an innovative control framework exploiting software technologies developed for high performance web services and therefore capable to handle millions of users accessing the services and interacting with one another.

The framework was designed to be scalable and cloud aware and, as a result, suitable for application in many different contexts, beyond those required by the scientific community.

The right way to look at !CHAOS is as a SaaS that specializes in controls. In fact, we often refer to it with the neologism of *Control as a Service* (CaaS).

Moreover, !CHAOS ranks in that niche — still partially unexplored — between Control Systems and DAQ Systems. Indeed, the horizontal scalability of the framework allows fast storage for multiple data sources with sub-milliseconds timing, granting a centralized time synchronization among them of the order of milliseconds. This feature — embedded by design in the !CHAOS architecture — is fundamental to make cross correlations among many heterogeneous data and also opens to new applications such as diagnostic, maintenance and predictive maintenance of large scientific facilities and industrial plants.

Data handling is also greatly eased by the adoption of non-relational databases and BSON/JSON data representation throughout the system.

The !CHAOS framework is currently used both in industrial and scientific collaborations.

SYSTEM OUTLINE

Back-end Layer

The framework relies on back-end services which provide core services such as data caching and permanent storage. The back-end services are:

- the Distributed Object Caching (DOC) that continuously store the latest datasets representing all the physical and abstract entities acquired or computed by the system;
- the Distributed Object Storage (DOS) that enqueues the above datasets in a persistent database;
- the metadata storage that holds configurations and preferences data — both of the system itself and the elements under control — in a persistent database.

The key-features of all the services are that they (i) build upon non-relational logic and (ii) handle datasets consisting of BSON objects (which ultimately are blobs of bytes). As a result, different data structures can be directly stored as key-value datasets, thus without any extra data parsing or manipulation and the need of setting up many different table structures. Moreover, the inherent schemaless nature of non-relational databases, greatly speed-up, as in the case of DOS service, the writing of permanent data, for the benefit of the !CHAOS DAQ performance.

Each of the framework's core services is not binded to any particular software technology or vendor. Different commercial, or open source, software can be adopted — as explained below — to implement the back-end functionality, the choice depending on the specific needs of the context. On the contrary, it is essential that DOC, DOS and metadata storage services can run as multiple instances on multiple machines, which is a prerequisite for the horizontal scalability of !CHAOS.

The current release of !CHAOS employs Couchbase® for the DOC and MongoDB© for the DOS and metadata storage. Similar products have been successfully employed and other are currently under evaluation (such as redis and Cassandra).

Both Couchbase and MongoDB deliver distributed architecture and provide compute, storage, and processing workload partitioning to meet ever-changing requirements.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEC5

Abstraction Layer

On top of the back-end layer lies the abstraction layer which provides !CHAOS services and a common application interface to the framework core services (Fig. 1). !CHAOS services interface with the back-end through dedicated drivers so that, if you want to use different products, it is sufficient to replace the related interface drivers. The !CHAOS services are:

- the Data Service (DS) that manages the data flow to and from the DOC and DOS services;
- the Metadata Service (MDS) that manages storage and retrieval of metadata to and from the DOS service.

Practically, both DS and MDS act as *routers*: they dynamically dispatch and gather data to and from the nodes of the back-end, including, bandwidth optimization, workload balance and handling of back-end nodes failure. This design guarantees the scalability of the system and even enhances it, as the balancing logics exerted by the DS and MDS drivers can be optimized to meet particular needs.

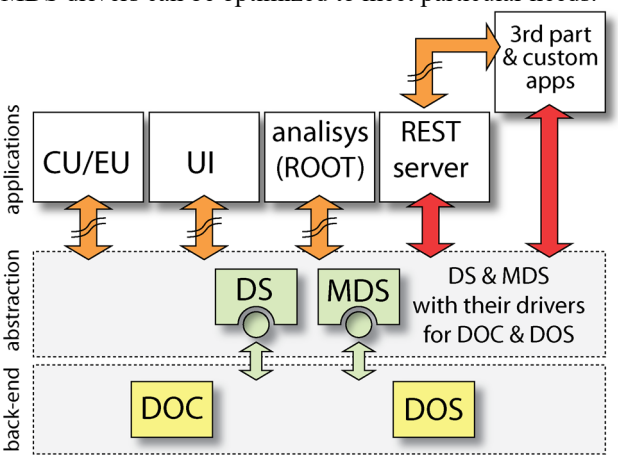

Figure 1: Layout of the !CHAOS framework.

The DOC & DOS services are implemented by Couchbase & MongoDB. The DS & MDS adapt to different back-end services by changing the corresponding drivers. In the applications layer, the segmented arrows indicate that the node can operate from outside the !CHAOS cloud, whilst the countinuous ones indicate that — for optimal performance — the node has to be within the cloud.

Applications Layer

Above the abstraction layer there is the applications layer, where different kinds of nodes perform the functions of raw data production, calculation and correlation, access to data, system metric and access from external applications. A short description of the !CHAOS main nodes is presented here below.

Control Unit When it comes to control systems, the tasks of acquisition and actuation of devices are among the most important because they are at the heart of the whole management of a plant. The CUs is the node that continuously acquire data from a device and operate it upon commands or given conditions. In !CHAOS, a device is set out by a virtualization process that — starting from a physical object — concludes with the definition of a set of meaningful variables (dataset) that fully describe it from the vantage point of the control and a set of actions you want to be able to perform on it (commands).

Each device is managed by a dedicated CU that continuously acquires it, refreshes its dataset, associates it with a timestamp and pushes it — through the DS — into the DOC and DOS. The frequencies of those pushes can be set independently. The CU can also perform advanced data processings on the spot and fill the dataset with derived quantities or complex functions result (FFT, pedestal calculation, mobile average, etc...).

Each remote command coming from a UI/EU/Control application/MDS is enqueued in a priority queue and executed accordingly. For every command the programmer can define four different handlers that encode four different states of a command: start, acquire, feedback and end. Typically, a command is in start at the beginning, then it loops at a given frequency between acquire (device acquisition) and feedback (device actuation) and exit on end.

Execution Unit The Execution Unit node (EU) is a strong point of !CHAOS being a very convenient way to inject processes in the system and have them execute in an standardized and managed manner. An EU is structurally similar to a CU with the difference that it doesn't access hardware but data already acquired, instead. An EU can receive commands, issue multiple commands to multiple EUs and CUs and access both live and stored data. These capabilities make it a very versatile tool that can execute automatic feedbacks, make correlation among live and stored data, execute background analysis on big quantities of stored data, build and issue macro-commands, implement complex procedures and so on.

In the current version of !CHAOS the EUs can access data through the DS from DOC and DOS so that they are very similar to batch processes. Developers are working to implement a stream processing mode where EU will be connected among other nodes (CU, EU) through input and output ports realizing this way processing pipelines where EUs are able to check-in to one or more CUs or EUs and directly receive continuous streamings of live datasets from them, which will dramatically increase the bandwidth and — as a result — their responsiveness. Currently an EU algorithm can be expressed in C++ native language, CERN ROOT or lua scripting. Next developments envision to include python and nodejs scripting. To support streaming processing and total abstraction of EU processing algorithms we'll support a graphical tool where the user can build its processing pipeline simply connecting "boxes" (CU-EU, EU-EU) connecting input terminals of a EU to output terminals of a CU, EU.

User Interface The User Interface node (UI) is where data are presented and user commands are issued. UI nodes can query the MDS for information about dataset structure and command set of any element, which allows for a dynamic adjustment of the control windows.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEC5

REST Server The REST server maps !CHAOS C++ APIs to HTTP/REST JSON APIs allowing external applications to access I/O, control and administration services of the framework. The !CHAOS BSON internal data representation is well suited for the implementation of WEB services based on JSON notation (since BSON is just a standardized binary serialization of JSON). Any external application can act as a CHAOS node (CU, EU, UI) according to needs. Most popular development environments have HTTP support for binding and integration of !CHAOS REST API. Currently !CHAOS has bindings for MathLab, LabVIEW, JavaScript, Python (basic), which allows users to write their own control, analysis or presentation application using the preferred environment.

Analysis Framework The CERN ROOT analysis framework with native C++ !CHAOS APIs is available both as a standalone application and as a plugin able to execute scripts from within an EU. The combination of a HEP well-known analysis tool with a Control System with DAQ capability, allows users to develop new complex and efficient control and data analysis algorithms.

HARDWARE INFRASTRUCTURE

As mentioned above, the ideal type of a !CHAOS installation is that of an infrastructure fit for use as a *cloud* and therefore offer a *CaaS* service to both local and remote users. Our goal was therefore to set up an installation able to give a boost to the framework, leading it beyond the prototyping and testing phases.

We decided to utilize all our financial resources in laying solid foundations for the infrastructure and use — on top of these — hardware that can be updated later.

We obtained a dedicated room from our Computer Centre with buffered power supply and air conditioning and acquired a 32 ports core switch CISCO Catalist 4500 X and six 48 ports CISCO C2960X for the network backbone.

Then we populated 8 racks with disused hardware obtained by courtesy of CERN, totalling 472 logical cores, 824 GB RAM and 500 TB HDs.

This hardware came out of production at the end of 2013, which means that, despite being discontinued, it is still able to fulfill the function of a pilot installation.

The infrastructure houses:

- a production cluster made of 8 servers (each with 24 logical cores, 72 GB RAM);
- a pre-production cluster made of 16 servers (each with 8 logical cores, 16 GB RAM);
- a development cluster made of 16 servers (each with 8 logical cores, 16 GB RAM);
- a services cluster made of 3 servers (each with 8 logical cores, 8 GB RAM);
- 6 spare machines (each with 16 logical cores, 16 GB RAM) not configured;
- 6 NetApp storage servers with multi 10Gb/s interfaces and ~500 TB of available disk space (NFS exported).

OVirt was chosen as a hypervisor because it is an open-source product and proved — in other projects — to be reliable and scalable, features essential for the !CHAOS framework. Two hypervisors have been installed: one for production and services and one for pre-production and development so as to keep separate the management of the two groups of physical and virtual machines.

The guest machines are installed with different Linux distributions: Ubuntu Server for the MongoDB nodes and CentOS for all the other services.

The infrastructure uptime is 100% since its start-up in October 2017.

As described below, the infrastructure already provides control services to different LNF scientific installations. We are also going to open it to technological installations, small laboratories and external users working on the various lines of our accelerators.

SOFTWARE MANAGEMENT

!CHAOS is a project released as opensource software under the EUGPL 2.0 license. The framework and all its processes are entirely developed in C++ (CXX11 & C98) using CMAKE (> 3.0) as build system. The supported compilers are gcc (≥4.8) and llvm. !CHAOS compiles on most Linux distributions and MacOS.

Continuous Integration

The adopted agile workflow envisages that new developments add features and/or fix bugs, namely the last stable release must always be an improvement respect to the previous one.

To guarantee this simple — but fundamental — assumption, we have created a set of top-down non-regression tests, spanning from UIs, CUs and EUs software up to lower level services.

The source code is stored on the INFN gitlab [2] server and the whole continuous integration workflow is handled by many *runner* machines that allow you to build and execute all the tests several time a day.

Each time a git merge request is issued, a set of scripts — pipelined by gitlab — executes all those tests for the different supported OSs. Should any test fail, the merge request is rejected, otherwise the code is merged.

For the static analysis of the code, we use suitable wrappers that are called at compilation time and generate reports. Those reports are then automatically uploaded on the cloud services *Coverity Scan* and *SonarCloud* which provide an excellent static analysis of the source code (Coverity Scan) together with a more in-depth analysis (SonarCloud) and permit to trust the quality of the written code. By adopting a continuos integration method, that automatically supervises code changes, dramatically enhances productivity also making possible a truly concurrent development.

SYSTEM PROFILE

Regarding the performance of mongoDB on the pre-production infrastructure, the system showed no evident bottlenecks. Since the available hardware is not updated and has limited resources, as the number of MongoDB shards

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEC5

increases, the tests tend to a limit that prevents verifying the real potential of both architecture and back-end.

Nevertheless, before reaching these limits, it has been verified that the throughput increases proportionally to the number of shards.

It seems evident that a consistent measure of the system performance will be possible only by utilizing physical machines of a suitable rank.

Measurements

The frequency of I/O accesses it has been measured in a minimal !CHAOS installation on a server Intel-i7 - 16GB - 256 SSD - 3.6 GHz, composed by one Couchbase community-3.1.3 (as DOC) and one MongoDB 3.4.15 (as DOS), provided as docker nodes, plus a MDS node.

The measure consists in accessing the DOC & DOS services through the DS, from a minimal client written on purpose. The client mimes a producer node (CU, EU, Analisys) on pushes and a consumer node (EU, UI, Analisys) on pulls. All the measures are calculated as an average of 1000 accesses and repeated for different number of clients (in different thereads) running in parallel (Figs. 2 and 3).

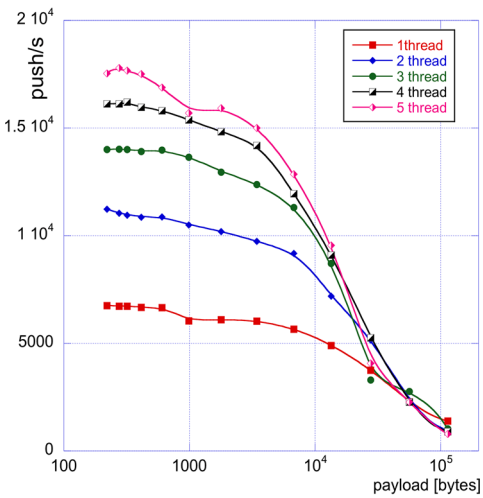

Figure 2: Push frequency vs. size of the payload at different numbers of threads.

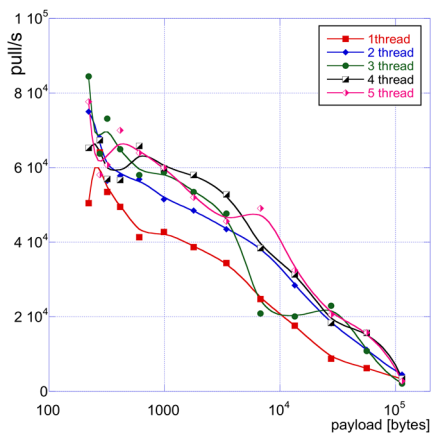

Figure 3: Pull frequency vs. size of the payload at different numbers of threads.

It can be noticed that the system is able to sustain the throughput as the number of threads increases. For higher numbers of threads, the load of the server gets too high and invalidate the measure.

By the end of November 2018, with the support of MongoDB Inc. Team a pre-production setup of MongoDB on physical machines will be arranged and will allow a quantitative measure for a multi-node configuration.

USE CASES

Beside the use of !CHAOS as control system in pilot installations at INFN-LNF: DAFNE transfer line, BTF, accumulator orbit [3]. In the following two paragraphs we present two further use cases with completely different objectives which use !CHAOS DAQ and analysis features.

Scientific

An independent data acquisition setup has been designed and realized in order to implement the fast luminosity monitor in view of the DAFNE future physics runs. Besides the total instantaneous luminosity the new diagnostic measures also the Bunch-by-Bunch luminosity. The following description focuses the attention on the systems engineering (for the scientific part refer DAFNE Luminosity Monitor [4].

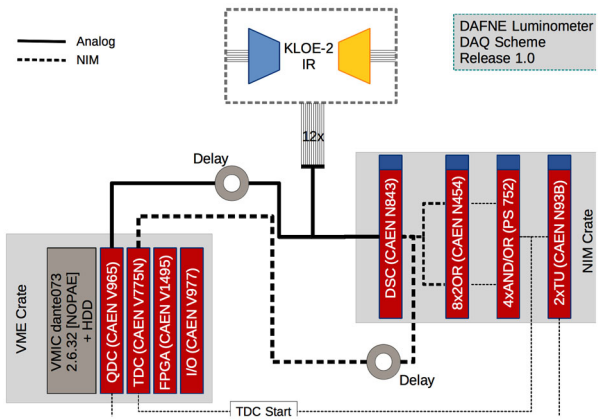

Figure 4: DAQ schema.

The new DAQ subsystem (Fig. 4) is composed by a 16 channels TDC (CAEN775N), 16 channels QDC (CAEN977), and a FPGA (CAENV1495) with few registers programmed as counters. A !CHAOS CU, running on an old VME Controller (VMIC linux 2.6.30), controls and acquires those devices at variable rates from 0 to 3KHz (depending on the collisions rate). The online computation of the Bunch-by-Bunch luminosity is performed by a ROOT application analysis connected with !CHAOS that correlates the data coming from the luminometer DAQ system with the data coming from the bunch charge monitor (acquisition 4096 points of a Tektronix scope).

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEC5

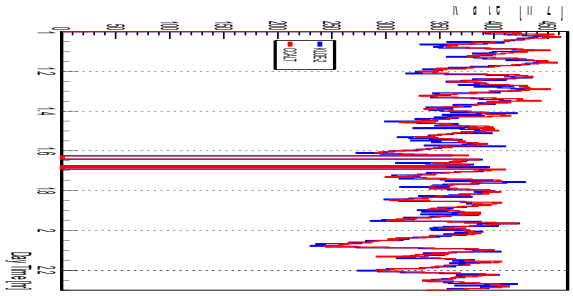

Figure 5: New (red) vs legacy luminometer (blue).

To monitor and check the results produced by the new DAFNE luminosity monitor !CHAOS also acquires the legacy luminosity monitor (Fig. 5), which updates every 15 s, and on-demand can also store beam camera frames at rate of 70 fps (Basler 640x480 8 bit) as overall cross check. All the correlations and checks of unrelated devices, without a common trigger or time, are made possible because the centralized !CHAOS infrastructure guarantees a time synchronization of the storicized datasets in the range of few milliseconds.

The possibility to have available many data coming from multiple sources that can be plotted and correlated each in real time, and the possibility to also control most of them greatly improved the productivity of the scientist team.

Industrial

INFN together with a world leader in the design and manufacture of automatic machines for the processing and packaging of pharmaceuticals, cosmetics, food, tea and coffee, is involved in a project named mAxima. The aim of this project is to setup a *cloud aware* infrastructure to acquire data from production line machines and perform predictive diagnostic to optimize their working point and prevent unwanted stops.

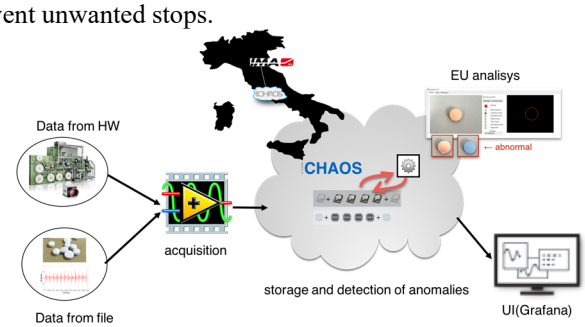

Figure 6: mAxima project: schema of DAQ.

The company provided to sensorize one of their package machines to measure: acceleration, temperature and pressure of some parts, and to acquire the images of pills that are going to be packaged. INFN provided a dedicated !CHAOS infrastructure suited to host processing predictive algorithms and their data flow (Fig. 6). Moreover created a !CHAOS support for Grafana [5]: an environment to create impressive WEB dashboards (Fig. 7). Since the real packaging machine was often not available, a synthetic flow of data were produced by a simulator able to inject respectively trends of errors in acceleration, pressure, temperature and different shapes of pills with zero or more defects, thorough this simulator we also tried algorithms to forecast failures and categorize images of bad and good pills.

Figure 7: Grafana WEB dashboard.

Currently we have hosted at INFN-LNF the infrastructure that is able to store the flow of data and produce real time forecast of failures of the given subset of parts, all these data are presented in a customizable WEB dashboard.

CONCLUSION

!CHAOS is a reality and has evolved from a mere Control System to a flexible framework offering also:

- a native DAQ system for processes with timing > 100 µs;
- a computing framework suitable to process data — tagged evenly— and produce feedbacks;
- a virtually unlimited horizontal scalability in terms of storage, memory and processing power;
- a cloud aware Control as a Service platform, suitable for different contexts (scientific, industrial, social, educational, consumer's).

REFERENCES

[1] L. Catani *et al.*, "Introducing a New Paradigm for Accelerators and Large Experimental Apparatus Control Systems", *Phys. Rev. ST Accel. Beams*, vol. 15, pp. 112804, 2012.

[2] https://about.gitlab.com/

[3] C.Bisegni *et al.*, "!Chaos Status and Evolution", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015. doi: 10.18429/JACoW-IPAC2015-MOPHA046

[4] A. De Santis *et al.*, "DAΦNE Luminosity Monitor", in *Proc. IPAC'18,* Vancouver, BC, Canada, Apr.-May 2018, pp. 338-340. doi:10.18429/JACoW-IPAC2018-MOPMF089

[5] https://grafana.com