ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA194

LIMA: LIBRARY FOR IMAGE ACQUISITION A WORLDWIDE PROJECT FOR 2D DETECTOR CONTROL

S. Petitdemange, L. Claustre#, A. Homs##, E. Papillon, R. Homs, D. Naudet, A. Henry, ESRF, Grenoble, France,
A. Noureddine, F. Langlois, SOLEIL, Gif-sur-Yvette, France,
G. Mant, STFC/Daresbury Laboratory, Warrington, UK

Abstract

The LIMA project started in 2009. The goal was to provide a software library for the unified control of 2D detectors. LIMA is a collaborative project involving synchrotrons, research facilities and industrial companies. LIMA supports most detectors used for X-ray detection or other scientific applications. Live display is supported via a video interface and most of the native video camera image formats are supported. LIMA provides a plug-in architecture for on-line processing which allows image pre-treatment before saving e.g. noise reduction algorithm or automatic X-ray beam attenuation during continuous scans. The library supports many file formats including EDF, CBF, FITS, HDF5 and TIFF. To cope with increasing detector acquisition speed, the latest LIMA release includes multi-threaded, parallelized image saving with data compression (gzip or lz4). For even higher throughput a new design, based on a distributed multi-computer architecture, of the LIMA framework is envisaged. The paper will describe the LIMA roadmap for the coming years.

INTRODUCTION

LIMA was born to address the problem of controlling 2D detector in the context of beamline (BL) control systems [1]. An important number of detectors need to be integrated in order to operate BL experiments and different approaches had been followed in the past in order to optimise efforts in this (never-ending) integration process. Based on the accumulated experience at the ESRF, LIMA has been built on top of the following paradigms:

- Clear separation between image generation and image processing
- Use of events and threads in order to better use system resources
- Control-system agnostic library that can be included in different kinds of applications
- High-performance code in C++, which can be bound to other high-level languages

Structure

The implementation of these concepts was made using the plugin philosophy, shown in Figure 1. A LIMA core library contains the code for image processing and a camera plugin is in charge of generating the images. The visible part of the library core is the Control Layer, exporting to the user the generic configuration and control of the image acquisition and processing. The camera plugin, also referred to as the Hardware Layer, registers to the Control Layer through a well-defined *hardware interface,* which contains different functional, independent blocks called *capabilities*. The *capabilities* control generic functionality that can be present in 2D detectors, covering different domains like image manipulation, external synchronisation, video streaming, among others. Three capabilities are mandatory for all plugins: generic detector information, frame synchronisation and image buffer control. Others, like pixel binning, region-of-interest (RoI) selection and shutter control are optional.

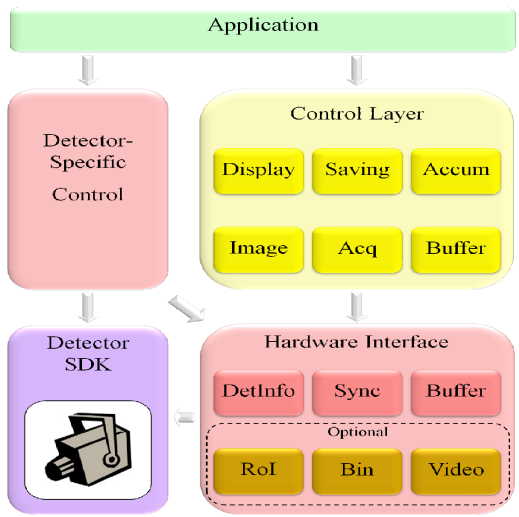

Figure 1: general LIMA layout.

Once the capabilities are discovered and configured, the control layer can start an acquisition of a sequence of frames. It is the responsibility of the plugin to inject each new acquired frame, which enters into the processing chain.

Processlib

A helper library *Processlib* was developed to implement the frame processing chain. It allows defining a sequence of tasks to be executed to each acquired frame. Tasks can run sequentially or in parallel, depending if they modify the source image or not. They are executed by a pool of threads, which is dimensioned depending on the number of available CPU cores. Operations on different frames can be parallelised, allowing data acquisitions to run faster than a traditional single-CPU

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA194

approach. As mentioned before, the library is based on events; each task can have an *end callback* that notifies its result is ready.

Features

Some detectors provide a raw image that requires a preliminary treatment before being used. For instance, multi-module panels can feed data that requires a geometric reconstruction, or coded pixel data that must be translated into intensity values. For this purpose LIMA provide an entry point for frame reconstruction, to be executed at the beginning of the processing chain.

If not (completely) done by the hardware, the image can be manipulated to meet the user requests in terms of pixel binning, horizontal/vertical flip, rotation and RoI selection, which is then published as the *base image*. Background subtraction, flat-field correction and pixel masking can be applied to the *base image* to obtain the *final image*. At this point the image can be saved, or used for data reduction, like RoI counters and *beam position monitor* (BPM) functionality.

Collaboration

Shortly after its creation and introduction into the ESRF BL control system, SOLEIL Synchrotron joined the LIMA development in a collaboration framework. Following this initiative, other large facilities, research institutes and detector manufacturers have joined the active LIMA collaboration.

NEW FEATURES

In order to satisfy new requirements for experiment control, the LIMA library core has been extended with several new features.

Hardware Layer

Some detector libraries allow native data saving, such as the Dectris Pilatus file saving in CBF format. A new *HwSaving* capability has been added to activate this feature from the Control layer saving interface.

In the same way, the hardware plugin can provide the software frame reconstruction method to be applied by LIMA/*Processlib*. For these cases, the *HwReconstruction* capability formalises the specification and activation of such software tasks.

Finally, the *HwVideo* capability has been extended with the Auto-Gain mode. Video cameras included in LIMA like the *Basler* support this functionality.

Control Layer

In some experiments, the camera image configuration is changed between different, well-defined modes. For instance, optical microscope configurations can require different flipping and rotation modes. The same need can apply to saving parameters, which can be specific to different kind of experiments. An interface based on *libconfig* [2] has been added to LIMA, which allows identifying configurations of the different subsystems (image, acquisition, saving, etc). Pre-defined configurations can be arbitrarily activated, with the possibility of being saved in persistent files.

Some X-ray diffraction images consist in circular patterns, which can be reduced by azimuthal integration. Region-of-Interest (RoI) counters have been extended to arc-like shapes, in order to calculate the evolution of diffraction ring intensities. These counters are used, for instance, in the online construction of sinograms in diffraction tomography experiments.

Extensions have been made in the accumulation acquisition mode. Information on pixel saturation above a pre-defined threshold can be retrieved in accumulation mode, necessary to detect non-linearity in the summed image. A saturated mask is generated for every accumulation, indicating, for each pixel, how many base images had the value exceeding the threshold. In addition to that it is possible to retrieve how many pixels saturate on each acquired frames. A callback can be registered to notify that over-exposure affects a number of pixels above a predefined value, a direct indication of a strong dose. It can be used in the Equipment Protection System to execute an action against sensor over-exposure, like shutter closing or an optimal selection of automatic filters. Finally, a specific offset subtraction can be applied on every base image before being accumulated, avoiding large background accumulation for non photon-counting detectors like the CMOS Andor Zyla.

The frame processing task interface has been explicitly exported to Python, allowing the execution of pure-Python code on every acquired frame. Concerning the *Processlib* execution dynamics, the task priority is now adjusted to its age, also known as aging scheduling, favouring older tasks to run first and avoid starvation. This forces a continuous flow of frames data execution chain, avoiding old frame stacking when the system is saturated.

A *peak finder* algorithm has been added to the processing task library, providing faster single-peak BPM functionality.

Data Saving

The HDF5 file format has been natively included into LIMA. The Nexus standard structure is respected, supporting multi-data set files.

Two different data compression algorithms have been added to the EDF format: gzip (EDFGZ) and lz4 (EDFLZ4). Similar to CBF, frame compression is executed in parallel for an optimal use of the available CPU (cores). In particular, the compression code has been optimised for runtime speed.

The EDF format has been extended to dynamic frame concatenation (EDFConcat). A single image in the file can grow with the concatenation of new frames, similar to the stripe concatenation; the header is updated on each frame.

New parallel file-systems like GPFS feature an increased performance when multiple frame streams are used. LIMA now allows parallel frame saving in order to benefit from this functionality.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA194

High-level Interface

The control of the above-mentioned features is available in the Python/Tango LIMA interface. In addition to that, Tango events can be activated on the LIMA image counters, which are by default moderated to maximum update rate (25 fps).

Time-resolved 1D spectral measurements can generate a large number of small frames, which can be packed together in the stripe-concatenation mode. This data can retrieved in blocks of multiple frames from the Tango server, optimising the I/O to the client. This read-block mechanism has been extended to 2D data, transferring 3D image stacks to the client, useful in high frame rate applications.

The current LIMA/Python code has been ported to Python 3.6, both on Windows and Linux OS. The TANGO interface uses PyTango 9.

Visualisation

In addition to the existing visualisation mechanisms based on shared memories, openGL widgets and web servers, a new project for the LIMA graphical user interface has been developed (Fig. 2). It is built on top of the Taurus framework [3] and it contains in an ergonomic design the control of all the LIMA features, The GUI can either instantiate a local LIMA controller or connect to a remote TANGO server.

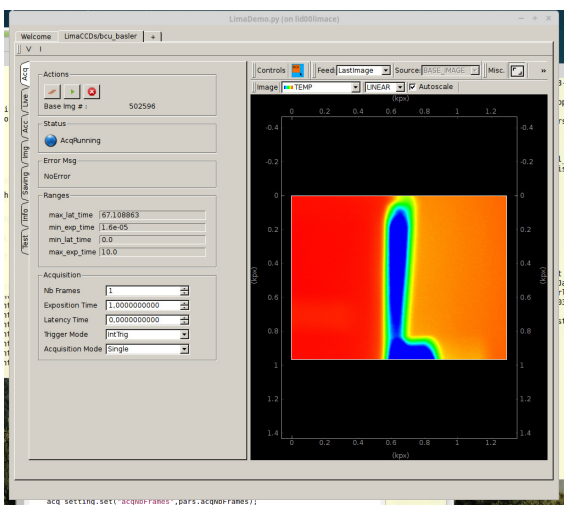

Figure 2: Display with Lima-GUI.

AtkPanel is the generic TANGO monitoring application, it helps in the development, diagnostic and operation of TANGO control environment. A TANGO device called *LiveViewer* has been developed to extend AtkPanel visualization to LIMA servers; figure 3 shows a snapshot of AtkPanel displaying *LiveViewer* image attribute.

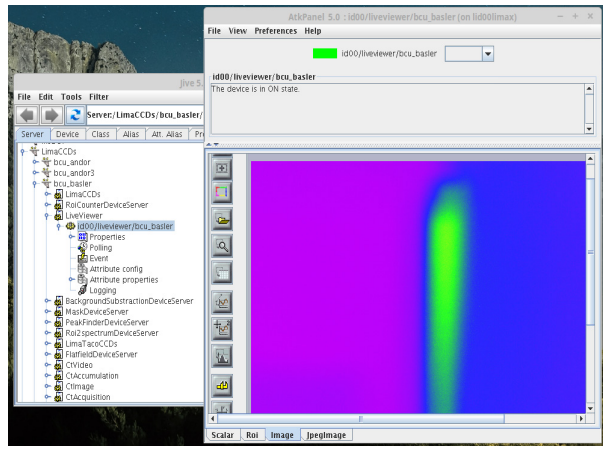

Figure 3: Display with Tango tool AtkPanel.

A plugin to the open-source *ImageJ* image processing framework was developed to control and read data from the *LiveViewer* LIMA TANGO device. It provides visualisation services in environments with strong Java image processing background.

BLISS

The BLISS framework [4] is the evolution of the ESRF BL control system, aimed to address the future challenges of the EBS instrumentation. Completely based on Python, its modular architecture is being developed in a way that it can both satisfy simple configurations and evolve in complexity to fit modern BL requirements. As a key element in the data acquisition chain, LIMA devices are integrated into BLISS, featuring image readout, RoI counters and BPM data.

NEW DETECTORS

In addition to the previously reported [1] list of detectors (~20), the following hardware has been interfaced to LIMA:

- PSI Eiger 500k and 2M
- Dectris Pilatus3, Eiger
- DSG/STFC/Quantum Detectors: Hexitec, Merlin, XSPRESS 3 and Ultra
- Pixirad PX1 and PX8
- Andor sCMOS detectors: Zyla, Neo
- PCO 2K, 4K and Edge HS
- imXPAD pixel detectors
- Aviex PCCD-170170
- Hamamatsu Orca-Flash
- v4l2 : webcams and other compatible cameras
- Maxipix has been ported to C++ (from Python).
- Dectris Mythen3

The following detectors, previously supported only on Windows, are now ported to Linux OS:

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA194

- Dexela flat panel
- PCO SDK

CHALLENGES

Saving

The previous list of detectors shows the continuous trend towards higher performance data acquisitions. For instance, the following list summarises the maximum 2D data rate that can be generated:

- PCO Edge: 1 GByte/s
- Dectris Pilatus 3 2M, PSI Eiger 500k: 2 GByte/s
- Dectris Eiger 4M: 4 GByte/s
- PSI Eiger 2M: 8 GByte/s

By default, LIMA allocates as much memory as possible to hold the full acquisition in RAM. File saving is started in parallel as soon as the data arrives. Thus, for very long acquisitions, which do not fit on the backend computer RAM, the frame rate is basically limited by the compression factor and the effective data bandwidth to the storage subsystem. Saving on local disks can easily go to 1 GByte/s and higher, but size is currently limited to several TeraBytes. This capacity is not enough for intensive, high duty-cycle experiments without an effective dead time for data flushing from local disks to larger storage.

The ESRF has traditionally been equipped with high-performance central, shared storage servers, absorbing today up to 1 GByte/s per single 10 Gigabit Ethernet link. GPFS was deployed in the NICE storage systems, notably increasing the overall performance with small, compressed single-image files. For example, the Pilatus 3-2M detector data can be compressed by a factor 2-3, with an effective saving data rate of 700 MByte/s, thanks to the LIMA parallel saving and the improved compression speed mentioned before.

However, this performance cannot always be kept sustained for long periods (~1 day) if multiple BLs are generating that amount of data. Investigations are taking place to find out the optimal storage infrastructure topology. Alternative systems include dedicated buffer storage, decoupling the needs for online analysis from long-term archiving.

Computer Performance

2D detector backend computers are typically selected for high-performance resources, basically CPU processing power, available RAM and global I/O bandwidth. The explosion of high-speed X-ray detectors, in particular those that can be aggregated due to their modular design, is pushing to the limits of commercial available systems.

In order to operate the detectors to their maximum capabilities, the PCs must be tuned to their optimal performance, from the BIOS settings, like hyper-threading, memory speed and power management, to OS configuration like isolated CPUs, CPU affinity, real-time scheduling capabilities, network subsystem IRQ and memory options, among others.

Long-term Stability

LIMA has been ported to CMake build utility in order to simplify the management of different operating systems (Windows and Linux) and compiler variants. Evolution in Python version is also taken in charge by the CMake porting.

In order to go one step further towards long-term stability, LIMA has been installed into continuous integration (CI) platforms. It is based on GitHub services that allow connecting Travis-CI [5] for Linux and Appveyor [6] for Windows. New commits and pull-requests trigger automatic compilation batches and unit tests, with their corresponding reports [7].

ROADMAP

Visualisation

Silx [8] is a data visualization and analysis framework for synchrotron applications. In the medium term strategy, it is envisaged to use it as the *de-facto* LIMA visualization mechanism.

High-performance Acquisitions

As analysed before, the continuous development of high-speed detectors pushes data acquisition performance to new levels. A single backend computer cannot always process, in the standard way, all data coming for an aggregated, multi-link detector system. An evolution of LIMA to multi-backend systems will be needed in the near future to cope with such challenging data acquisitions. At least two different data dispatching mechanisms are foreseen: a full frame is send to one backend PC, or all the frames coming from the same detector module are sent to a dedicated PC.

The frame processing chain in LIMA is currently linear shaped. That is, one frame at the hardware source is transformed into one frame at the end of the chain. Saving at different stages of frame processing is not currently supported by LIMA. The introduction of branches in the frame processing chain is expected to open new possibilities in image data reduction and online analysis. In the same way, a unified interface for different saving streams will help in the storage at different frame process stages, including data reduction software plugins, like PyFAI and sinogram generation.

LIMA controls directly the memory buffers to be used in image acquisition, but the global management of memory allocation for software processing tasks needs to be addressed. In particular, it is desirable to de-couple the dynamics of frame acquisition and processing. For instance, a new hardware acquisition could start while the frames from the previous acquisition are still being saved, reducing the dead-time in experimental sequences. This flexibility demands, among others, a finer control on memory buffer dynamics. Future development roadmap includes a review of the buffer memory infrastructure.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA194

Packaging and Deployment

A global packing solution is being studied inside the BLISS development framework and LIMA is also in the loop. The *Conda* package management system [9] is currently under test. From the deployment point of view, *Ansible* [10] is a candidate that is being considered. The *Supervisor* utility [11] can be used in the future as a generic tool for controlling LIMA control processes, typically TANGO device servers.

CONCLUSIONS

LIMA continues its way controlling diverse 2D detectors in synchrotrons and other large facilities. In production since 2010 in different control environments, it is a mature and stable library. New features and performance optimisations have been added to better satisfy scientific demands. At the same time, new detectors have been integrated, creating new challenges in terms of functionality and performance.

The roadmap envisages structural improvements that allow a better use of computing resources and expanding LIMA to multiple backend PCs.

The LIMA community keeps its active collaboration, contributing with new camera plugins and software tasks, as well as testing and debugging new releases in different environments.

ACKNOWLEDGEMENTS

As expressed above, important contributions in hardware plug-ins, saving interfaces, installation scripts and TANGO extensions have been made by collaborators, in particular at SOLEIL, PETRA III/DESY, FRM-II/TUM, MAX-LAB, ALBA, ADSC and Rayonix.

REFERENCES

[1] S. Petitdemange, L. Claustre, A. Homs, E. Papillon, R. Homs-Regojo: “The Lima Project Update”, Proceedings of ICALEPCS2013, FRCOAAB08, San Francisco, CA, USA.
[2] http://www.hyperrealm.com/libconfig/libconfig.html
[3] http://www.taurus-scada.org
[4] M. Guijarro_ A. Beteva T. Coutinho M. C. Dominguez C. Guilloud A. Homs, J. Meyer V. Michel E. Papillon M. Perez S. Petitdemange: “BLISS - Experiments Control for ESRF EBS Beamlines”, ICALEPCS2017, WEBPL05, Barcelona, SPAIN
[5] https://travis-ci.org/esrf-bliss/Lima
[6] https://ci.appveyor.com
[7] git://github.com/esrf-bliss/Lima.git
[8] J. Kieffer, P. Knobel, D. Naudet, P. Paleo, H. Payno, V.A. Sole, V. Valls, T. Vincent: “Python packages to support the development of data assessment, reduction and analysis applications at synchrotron radiation facilities”, 2017, doi:10.5281/zenodo.576042, http://www.silx.org
[9] https://conda.io
[10] https://www.ansible.com
[11] http://supervisord.org