Deliverables

Deliverable Number	**D24.3**
Deliverable Title	**Cross site use case requirement report including comparison of existing solutions. (Month 12)**
Lead Beneficiary	**PSI**
Type	**Use case requirement**
Dissemination Level	**Public**
Due date of delivery	**Month 12**

Deliverable report

Premise

The present deliverable describes cross site use case requirements and reflects a comparison of existing data analysis solutions at the different facilities (see ANNEX 1). From the existing solutions a set of use cases is selected as suitable candidates for providing harmonised solutions that can be transferred from one to another site (see ANNEX 2).

The aim of the above described use case selection is to build up demonstrators for remote data analysis for a small number of archetypal experiments. The demonstrators will build on the HPC platforms of each participating institute. The demonstrators will be cloud based in those institutes where cloud technology is deployed. In the other institutes the demonstrator will run on standard HPC hardware. A web portal will ensure users to have a common user experience.

Description of Work

WP24: Demonstrator of a Photon Science Analysis Service

1. Introduction

The present document presents a comparison of existing data analysis solutions at different facilities and describes the selection procedure for the different use cases and their leading house facilities.

Deployment of complex data analysis frameworks and tool-chains is a common task at research facilities and frequently a major hurdle for scientists, hampering rapid data analysis and publications. Simple assembly of integrated and deployable applications reduces the RI efforts as well as it accelerates the scientific process.

The selected use case applications will be implemented as deployable packages, as pre-configured virtual machines or as containers. Virtual machines or containers provide encapsulated user environments, which can be archived together with the experimental data, thereby capturing valuable provenance data and strongly supporting reproducibility of the original experiment and data analysis workflows

2. Data analysis application USE CASE SURVEY amongst CALIPSOplus members

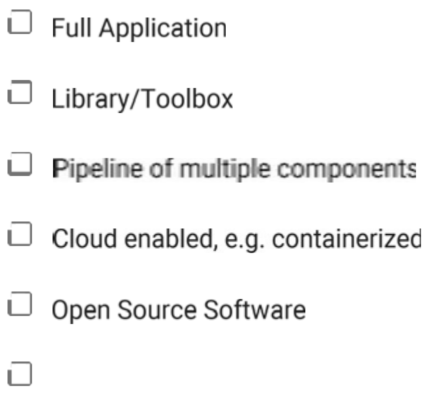
A survey using, google docs, was performed among the different CALIPSOplus members to compile a list with "reusable" scientific analysis software use case candidates at the different facilities.

Feasible candidates for use cases were the applications for online analysis of scientific data, which are at the "end" of an analysis chain, and therefore producing results which are directly useful for publications. However, any other application considered re-usable for other sites could be nominated. "Application" could also mean libraries, toolboxes or components from which full applications could be more easily derived. The important selection criterium is the “re-usability” at other sites.

In case a software is selected as use case, the corresponding site owning the selected software, will play the role of the "leading house". The leading house further develops the application and makes it available to the other sites for the local usage. Each site was able to propose several candidates for use cases. In the end a set of three application use cases, from different sites, was selected (see ANNEX1).

Questions of Survey were the following:

1. E-Mail-Address
2. Name of the application (or library,toolbox..)
3. Name and Email of contact person(s) (will be asked questions relevant for the use case selection process)
4. Name and email of main developer
5. Site ("leading house", e.g. ESRF)
6. Provided functionality and features
   a. What would be the steps needed to make the software available and usable at other sites. Mention current local dependencies which would need to be substituted
   b. Is further functional development needed or planned? For which features?
   c. If the software is already described on a software catalog e.g. https://software.pan-data.eu/ please provide URL
7. Select features of the software (multiple selections allowed)



☐ Full Application

☐ Library/Toolbox

☐ Pipeline of multiple components

☐ Cloud enabled, e.g. containerized

☐ Open Source Software

☐

Integration into a "portal" possible or even available

☐ Can be used for interactive analysis

☐ Can be used in batch queue systems

☐ Sonstiges:

8. How many users do use the software already today? Is there already a "community" using (and/or improving) the software?
9. What is the (foreseen) license under which the software can be used?
10. Do you know of similar applications already in use at other sites which have an overlapping functionality with the proposed software?
11. Technical details like provided application stack, language, needed, operating system etc.?
12. What level of software support for other users can realistically be offered for the software in future (type of support, amount of longterm available manpower etc)?
13. What documentation/user guide/tutorial is available or planned for the software?
14. Any other comment about the software which you consider relevant for the selection?

The detailed answers of beneficiaries CALIPSOplus can be found in the table with the outcome of the use case survey see ANNEX1. The table shows the broad application spectrum available at the different facilities. A selection was made from this table acording to the selection criteria that were described in section 2.

From the table in ANNEX 1, 5 use cases were selected namely:

1. ***CrsytFEL*** leading house DESY

   CrystFEL is a suite of programs for processing diffraction data acquired "serially" in a "snapshot" manner, such as when using the technique of Serial Femtosecond Crystallography (SFX) with a free-electron laser source. CrystFEL comprises programs for indexing and integrating diffraction patterns, scaling and merging intensities, simulating patterns, calculating figures of merit for the data and visualising the results. Supporting scripts are provided to help at all stages, including importing data into CCP4 for further processing***.***

2. ***Ptycho Shelves*** leading house PSI
   Ptychography is a technique that combines scanning and coherent diffractive lensless imaging. In its more conventional configuration a sample is scanned through a beam and for each scanning point a diffraction pattern is measured, making sure to have some degree of overlap between neighboring scanning positions. The diffraction patterns are then fed to a iterative reconstruction algorithm to reconstruct the complex-valued incident wavefield and complex-valued sample transmissivity. In doing this one does away with image forming optics and can obtain imaging resolution not bounded by their quality and aberrations.
   Ptycho Shelves is an innovative and conceptually-simple modular framework for ptychography reconstructions. With the constant development of algorithms for ptychography there is currently a plethora of different options for reconstruction algorithms, or engines. Furthermore, different engines offer different features for correction of experimental imperfections. Hence it becomes important to be able to stack together different modules such that different engines can be executed in series, with the output of one algorithm being the initial guess of the following. Along with the package come several of currently used ptychography algorithms and the framework allows for further expansion.

At some experimental stations there are multiple X-ray detectors available, or different options for computing sample positions based on several interferometric readings. Ptycho shelves also offers flexible modules for switching between different detectors, data preparation functions, or computations of sample positions.

3. ***Savu*** leading house Diamond.
   Savu is a Python package to assist with the processing and reconstruction of parallel-beam tomography data. The project originated in the Data Analysis Group at the Diamond Light Source (UK synchrotron) to address the growing, and increasingly complex, needs of the tomography community.
   Designed to allow greater flexibility in tomography data processing, Savu is capable of processing N-dimensional full-field tomography and mapping tomography data, along with concurrent processing of multiple datasets such as those collected as part of a multi-modal setup. Savu is currently in use across the tomography beamlines at Diamond to reconstruct both full-field tomography data and multi-modal, mapping tomography data.
   Savu is an object-oriented Python framework, with a modular plugin architecture. The framework handles the movement of the data, in serial or parallel on a local PC or cluster, and each plugin performs a specific independent task, such as correction, filtering, reconstruction. Savu process lists, tailored to a specific experiment and passed to the framework at runtime along with the data, detail the processing steps that are required. The process list is created using the Savu configurator tool, which stacks together plugins chosen from a repository.

4. **pyFAI** leading house ESRF
   2D area detectors like CCD or pixel detectors have become popular in the last 20 years for diffraction experiments (e.g. for WAXS, SAXS, single crystal and powder diffraction). These detectors have a large sensitive area of millions of pixels with high spatial resolution. The software package pyFAI has been designed to reduce SAXS, WAXS and XRPD images taken with those detectors into 1D curves (azimuthal integration) usable by other software for in-depth analysis such as Rietveld refinement, or 2D images (a radial transformation named caking in FIT2D). As a library, the aim of pyFAI is to be integrated into other tools with a clean pythonic interface or being used directly with the Jupyter interface. However pyFAI features also command line and graphical tools for batch processing, converting data into q-space (q being the momentum transfer) or 2θ-space (θ being the Bragg angle) and a calibration graphical interface for optimizing the geometry of the experiment using the Debye-Scherrer rings of a reference sample.

5. **PyMca** leading House ESRF
   X-ray microscopy is a common technique at synchrotron facilities that can be performed in different modes (scanning or full field) using a large variety of techniques (X-ray fluorescence, X-ray Absorption Spectroscopy, X-Ray Diffraction…) often combining more than one technique simultaneously. There is a clear need to have applications able to explore X-ray microscopy datasets. PyMca is one of them. The programs aXis2000 and MAPS are notable alternatives based on a commercial language (IDL). A port of MAPS to Python is going on.
   PyMca was initially developed to fulfill user needs related to X-ray Fluorescence Analysis (XRF) and it can be a considered a reference in that field like AXIL, GeoPIXE or GUPIX. It offers all what users can expect in that field (qualitative and quantitative analysis, interactive and batch processing, large variety of data formats support -including HDF5 NeXus-) while being the only free and open source alternative among the reference applications.
   Besides those specific applications, its visualization and data exploring capabilities make of PyMca a Swiss Army knife for the synchrotron user

Next steps

Based on the selected use cases listed in ANNEX 1&2 the next steps will be to prepare the software to make them available to other interested sites. In order to do this any existing local boundary conditions, at the different facilities, or special requierments need to be isolated and either removed or made adaptable to different environments.

Subsequently an example deployment on other sites, exploiting the outcomes of the other workpackage tasks, is foreseen

Conclusion

In this deliverable D24.3 we collected existing example applications for data analysis as used at the different facilities. The questionnaire above demonstrated the wide range of available applications. Often applications are mainly used at one or two sites only. The goal of WP24 is to make them available for the different sites.

From this set of applications 3 example applications were described in more detail as a preparation for the next step, which ultimately aim for making the corresponding data analysis software reusable across the scientific user community.

ANNEX 1 Table with outcome of the use case survey

ANNEX 2 Detailed use case descriptions of the 3 selected use cases

Copy of Collect Calipso JRA2 Use Case Candidates (Responses)

Email address	Name of the application (or library,toolbox..)	Name and Email of contact person(s) (will be asked questions relevant for the use case selection process)	Name and email of main developer	Site ("leading house", e.g. ESRF)	Provided functionality and features	What would be the steps needed to make the software available and usable at other sites. Mention current local dependencies which would need to be substituted	Is further functional development needed or planned ? For which features ?	If the software is already described on a software catalog e.g. https://software.pan-data.eu/ please provide URL	Select features of the software (multiple selections allowed)	How many users do use the software already today ? Is there already a "community" using (and/or improving) the software	What is the (foreseen) license under which the software can be used	Do you know of similar applications already in use at other sites which have an overlapping functionality with the proposed software ?	Technical details like provided application stack, language, needed, operating system etc	What level of software support for other users can realistically be offered for the software in future (type of support, amount of longterm available manpower etc)	What documentation/user guide/tutorial is available or planned for the software	Any other comment about the software which you consider relevant for the selection
alun.ashton@diamond.ac.uk	DAWN	Jacob.Filik@diamond.ac.uk	Jacob.filik@diamond.ac.uk	Diamond	Data Visualisation, Data processing (SAXS, XPDF, XRD, ARPES, PEEM, XAS….), Scientific Python IDE, Data webserver functionality. GUI or headless.	Already used in multiple sites.	Continuous developments with ~5 releases a year. some details on http://dawnsci.org/	No.	Full Application, Library/Toolbox, Pipeline of multiple components, Open Source Software, Integration into a "portal" possible or even available, Can be used for interactive analysis, Can be used in batch queue systems	Over 150 downloads a month, extensive use at Diamond Beamlines.	EPL (Eclipse Public License)	No single application but overlaps with domain applications e.g. PyFAI	Supported on Linux, Windows and MacOS. Main application in Java/RCP with possible Python interactions. Download bundle has all dependencies.	Over 15 active multi site developers for over 5years, support given for international synchrotron users/data processing as fits in with Diamond use cases, reduced support for other domains on a case by case basis. No current perceived reduction in support.	http://dawnsci.org/, including youTube videos etc., training and other workshops conducted on a domain/beamline basis.	Build on extensible framework that can be extended from within the application, also has fledgling marketplace to exchange and distribute extensions. Some components spun out into own open source projects e.g. Eclipse January. Is one of the flagship projects in the international Eclipse Science and Industrial Working Group.
alun.ashton@diamond.ac.uk	Savu	mark.basham@diamond.ac.uk	nicola.wadeson@diamond.ac.uk	Diamond	Flexible framework for massively parallel, cluster enabled data processing of high multidimensional 6D+ and multimodal (absorption, florescence, diffraction) tomography data.	Already used on multiple sites, standard install with main dependency Python and any required scientific libraries e.g. PyFAI, PyMCA, TomoPy, Astra	Future developments available : https://waffle.io/diamondlightsource/savu, e.g. use of object stores as data transfer mechanism.	No	Full Application, Library/Toolbox, Pipeline of multiple components, Open Source Software, Integration into a "portal" possible or even available, Can be used in batch queue systems, MPI	continuous usage @diamond, installed on many sites e.g. Max-IV, University of Hull, STFC.	Apache V2/GPLv3	not in entirety, elements or similar functionality available e.g. TomoPy	Python/Linux/HDF5/MPI	Already running workshops at similar facilities, as well as local/similar sites, both for users and contributors. Given the domain, its unlikely that support does not fit within DLS operational requirements.	http://savu.readthedocs.io/en/latest/ documentation, user guide etc available.	Relatively young (under 3yrs) project, would be invaluable for researchers to have access to commercial or private cloud processing due to the nature of the data. Some work already underway with STFC to explore these possibilities.
alun.ashton@diamond.ac.uk	SuRVoS	mark.basham@diamond.ac.uk	imanol.lungeo@diamond.ac.uk	Diamond/Nottingham University	Interactive segmentation of tomography datasets using machine learning and vision methodologies.	in use on other sites, needs GPU cards local or e.g. AWS.	Predominantly algorithm development and application to new tomography use cases	no	Full Application, Cloud enabled, e.g. containerized, Open Source Software, Integration into a "portal" possible or even available, Can be used for interactive analysis	Growing community due to its time saving properties in image segmentation.	Apache V2	Commercially available packages and some similar endeavours in CryoEM	Python/Linux/CUDA	best efforts online, currently only one dedicated developer with long term support.	https://diamondlightsource.github.io/SuRVoS/	Already used on AWS during training workshops.
j.kelling@hzdr.de	SpekNG	Nils Schmeißer, n.schmeisser@hzdr.de	Jeffrey Kelling, j.kelling@hzdr.de	HZDR	decomposition of multi-component spectra obained by e.g. EXAFS, TRLFS	software can work stand-alone	implementation of more analysis methods, UI improvements, tie-in to datamanagement systems		Full Application, Library/Toolbox, Open Source Software, Integration into a "portal" possible or even available, Can be used for interactive analysis, Can be used in batch queue systems	The software is currently in a closed beta with six users. One of its features makes it a modern successor to the program described in [1], which provides a prospective user community. [1] A. Roßberg, T. Reich, and G. Bernhard. Analytical and Bioanalytical Chemistry, 376(5):631–638, Jul 2003.	GPLv3+	Similar functionality is usually implemented in collections of Matlab scripts within the target community.	C++, webtoolkit (web UI), armadillo (BLAS)	In the long-term we plan for up to one person to maintain an develop the software.	The web-UI is self-explanatory to users in the field. Required documentation for the headless variant for batch systems will be provided in text form. Code-documentation is produced using Doxygen.	
j.kelling@hzdr.de	PIConGPU as a Service	g.juckeland@hzdr.de, m.bussmann@hzdr.de	Sebastian Starke, s.starke@hzdr.de	HZDR	This project aims to provide a user-friendly and interactive web-interface for high-performance plasma physics simulations. Our front-end provides guidance during the setup of simulation parameters, manages running and past simulations on a high performance computing (HPC) system. It also provides visualizations of data produced by both actives and completed runs. Our current back-end use-case is the highly efficient and widely used Particle-in-cell code PIConGPU[1] where abstraction of the involved simulation set-up procedure is expected to open this application to an even wider group of users. The front-end is designed to be adaptable to other use-cases, e.g. ab-initio or Monte Carlo codes, in the future. [1] M. Bussmann, H. Burau, T. E. Cowan, A. Debus, A. Huebl, G. Juckeland, T. Kluge, W. E. Nagel, R. Pausch, F. Schmitt, U. Schramm, J. Schuchart, and R. Widera. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13, page 5:1, New York, NY, USA, 2013. ACM.	Adding support for local flavors of HPC batch and module systems.	Currently PIConGPU use-cases for the Eupraxia[2] and EUCALL[3] projects are being implemented. [2] http://www.eupraxia-project.eu/ [3] https://www.eucall.eu/		Pipeline of multiple components, Open Source Software, Integration into a "portal" possible or even available, Can be used for interactive analysis, Can be used in batch queue systems	The service set-up at HZDR is provided for researchers in the Eupraxia and EUCALL projects.	GPLv3+	To our knowledge no similiar UI is in use at other sites. Due its work-flow abstracting functionality, it bears some similarity with proprietary cloud services offered in other areas, i.e. machine learning.	Python, Jupyter, Batch-system any back-end simulation tool, at the moment PIConGPU	This infrastructure is a main service being developed by the computing department at HZDR, with more than one person available to drive it. Main development of PIConGPU is being done by the institute for plasma physics at, HZDR. This allows new PIC use-cases to be implemented quickly.	An installation guide for the front-end can be made available. User documentation for the front-end is not required, apart form description displayed to the user during simulation set-up.	
manuel.guizar-sicairos@psi.ch	cSAXS Matlab scanning SAXS package	Manuel Guizar Sicairos (manuel.guizar-sicairos@psi.ch)	CXS group	SLS	Basic functionality for reading data, radially integration of data. Assemble of scanning SAXS data into orientation and degree of orientation images based on different q-ranges.	Matlab. A supported C-compiler for mex functions. Software is available online. Matlab functions need the cSAXS Matlab base package. https://www.psi.ch/sls/csaxs/software	In our group we continuously develop novel methods for analysis of SAXS data, such advances will be incorporated into the package.		Pipeline of multiple components, Open Source Software, Can be used for interactive analysis	Software today is used in experiments at the beamline. Also a few users that need to reprocess some data use it.	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.	Yes, but I don't know them specifically neither which ones are available online.	Matlab. Linux functions are used for the data preparation.	From our group, close to zero for non-users.	Users guide and description of functions. A journal publication exists that gives some details on the functionality (http://dx.doi.org/10.1088/1367-2630/11/12/123016).	
manuel.guizar-sicairos@psi.ch	SAS tensor tomography	Manuel Guizar Sicairos (manuel.guizar-sicairos@psi.ch)	CXS group	SLS	Processing of SAS data for tensor tomography. Reconstruction of anisotropic SAS signal in 3D.	Matlab functions need the cSAXS Matlab base package. Preparation of the inputs for tensor tomography can be done with the cSAXS scanning SAXS package. https://www.psi.ch/sls/csaxs/software	In our group we continuously develop novel methods for analysis of SAXS data, such advances will be incorporated into the package.		Pipeline of multiple components, Open Source Software, Can be used for interactive analysis	Software today is used in experiments at the beamline. Also a few users that need to reprocess some data use it.	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.	No	Matlab. Linux commands are used for the data preparation.	From our group, close to zero for non-users.	User guide and description of functions. A journal publication exists that describes the method. (http://dx.doi.org/10.1038/nature16056)	
manuel.guizar-sicairos@psi.ch	cSAXS Matlab tomography package	Manuel Guizar Sicairos (manuel.guizar-sicairos@psi.ch)	CXS group	SLS	Preprocessing, alignment and reconstruction of tomograms from phase and amplitude projections.	Matlab with parallel toolbox, cSAXS Matlab base package, CUDA for GPU functions https://www.psi.ch/sls/csaxs/software	In our group we continuously develop novel methods for processing and analysis of tomography data, such advances will be incorporated into the package.		Pipeline of multiple components, Open Source Software, Can be used for interactive analysis	Software today is used in experiments at the beamline. Also a few users that need to reprocess some data use it.	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.	Yes, ASTRA has its capability to reconstruct tomograms on its own, we use ASTRA lower level functions and apply them to our specific needs.	Matlab. Linux commands are used for the data preparation	From our group, close to zero for non-users.	User guide and description of functions. Journal publications exist that describe different functionalities of the package.	
cxs@fake.ch	Ptychography	https://www.psi.ch/coherent-x-ray-scattering/people	https://www.psi.ch/coherent-x-ray-scattering/people	SLS	Preprocessing of data and ptychographic reconstruction with various engines. An innovative and simple framework for ptychography that allows easy addition of reconstruction engines and flexible.	Software wrapper and prototype engines are in Matlab. One of the reconstruction engines is highly optimized in C++, can be compiled and run independently with standard libraries. Matlab functions need the cSAXS Matlab base package (https://www.psi.ch/sls/csaxs/software).	In our group we continuously develop novel methods for ptychography and these would be incorporated in this framework.		Pipeline of multiple components, Open Source Software, Can be used for interactive analysis, Can be used in batch queue systems	Software today is used in experiments at the beamline. Also a few users that need to reprocess some data use it.	Custom PSI license similar to share-alike, non-commercial.	Yes, with partial overlap. Ptypy, PtychoLib, PyNX, CAMERA SHARP.	Matlab. Linux commands are used for the data preparation. Optimized C-code engine can be run independently from Matlab.	From our group, close to zero for non-users.	There is documentation in an internal wiki which could be shared. A journal publication is also planned.	There are worries about having any individual name associated with this package in any shared platform. Hence as contact information contact the CXS group. https://www.psi.ch/coherent-x-ray-scattering/people

Email address	Name of the application (or library,toolbox..)	Name and Email of contact person(s) (will be asked questions relevant for the use case selection process)	Name and email of main developer	Site ("leading house", e.g. ESRF)	Provided functionality and features	What would be the steps needed to make the software available and usable at other sites. Mention current local dependencies which would need to be substituted	Is further functional development needed or planned ? For which features ?	If the software is already described on a software catalog e.g. https://software.pan-data.eu/ please provide URL	Select features of the software (multiple selections allowed)	How many users do use the software already today ? Is there already a "community" using (and/or improving) the software	What is the (foreseen) license under which the software can be used	Do you know of similar applications already in use at other sites which have an overlapping functionality with the proposed software ?	Technical details like provided application stack, language, needed, operating system etc	What level of software support for other users can realistically be offered for the software in future (type of support, amount of longterm available manpower etc)	What documentation/user guide/tutorial is available or planned for the software	Any other comment about the software which you consider relevant for the selection
frank.schluenzen@desy.de	CrystFEL	frank.schluenzen@desy.de	T.A.White http://www.desy.de/~twhite/crystfel/contact.html	DESY / CFEL	suite of programs for processing diffraction data acquired "serially" in a "snapshot" manner	no local dependencies. It's available as source, singularity and docker images	continuously developed	no	Full Application, Pipeline of multiple components, Cloud enabled, e.g. containerized, Open Source Software, Integration into a "portal" possible or even available, Can be used for interactive analysis, Can be used in batch queue systems, It's open software, but underlying software (ccp4, xds, etc) is strictly speaking not re-distributable	unknown. The community doing serial crystallography is small, but growing	GPL. CCP4 has it's own site-limited license.	no	Linux, MacOSX. Mostly C/C++ code	support is limited. on-site hands-on	detailed tutorials and lots of open access data	
andy.gotz@esrf.fr	PyFAI	Armando Sole - sole@esrf.fr	Jerome Kieffer - kieffer@esrf.fr	ESRF	Data reduction for diffraction e.g. powder diffraction, sax, waxs, dct, ...	Install pyfai + silx + jupyter (optional)	Jupyter notebook examples	https://software.pan-data.eu/software/73/pyfai	Library/Toolbox, Open Source Software, Can be used for interactive analysis, Can be used in batch queue systems	Yes	MIT	FIT2D	Python, C, OpenCL	Help with using and calibration	Complete documentation	
majid.ounsy@synchroron-soleil.fr	Passerelle EDM	majid.ounsy@synchrotron-soleil.fr	iSencia Belgium NV (erwin.de.ley@isencia.be)	SOLEIL	Modular server solution platform for decision process management. Applied for interactive and automated control, data acquisition and analysis processes. Core concepts are process models, rules-based analysis and integration with: •Python, JavaScript •Control and service bus-es (Tango, SOAP, REST, …) •Web forms UI for user interactions •Event-based process triggers •Resource/grid managers like SGE, SLURM Includes flexible clustering features & data grid. Fully-featured web UI incl graphical process editor, running & debugging, role-based security, asset versioning, execution traces etc. Configurable dashboards for overall process monitoring and detailed analysis visualization.	Passerelle EDM is a commercial product for which a license should be purchased. Installations can be hosted in the cloud or local on-site			Full Application, Library/Toolbox, Pipeline of multiple components, Cloud enabled, e.g. containerized, Integration into a "portal" possible or even available, Can be used for interactive analysis, Can be used in batch queue systems, core engine is open source	Used by thousands of operators and technicians in Belgian telecom. Used since > 10 years at Soleil, Also used since 5 years at ESRF Yes, there is a community : iSencia collaborates and co-creates the software with its customers. Core engine evolves in open source, newest version is the Eclipse Triquetrum project, within the open-source Eclipse Foundation's Science Working Group.	Commercial for the server application, EPL/APL for the core engine	None with all these functionalities	Based on Java running in a JBoss application server. Standalone or Docker image. Uses a relational Database for storing process models, configuration data, execution traces, … English/French/Dutch language, Cross platform, typically used on Linux variants, MS Windows PCs.		Help files, documentation wiki,	
ferenc.borondics@synchrotron-soleil.fr	Spectral Orange, Soon to be extended with other toolboxes. The name of the application will be Quasar	Ferenc Borondics (ferenc.borondics@synchrotron-soleil.fr)	Marko Toplak (marko.toplak@gmail.com)	SOLEIL	Environment for the analysis of spectral data : currently Infrared data, developing code to extend the capabilities to processing of general spectromicroscopy data, data mining, hyperspectral data analysis, statistical methods, machine learning, data visualization, multimethod analysis, plotting, publication quality figure generation	None, freely downloadable and open source	Yes, for multiple features	https://github.com/markotoplak/orange-infrared	Full Application, Library/Toolbox, Pipeline of multiple components, Open Source Software, Can be used for interactive analysis, Can be used in batch queue systems	Yes	GNU GPL	None with all these functionalities	Installable in multiple ways, English language, cross platform		Help files, video tutorials	This is an international collaboration to democratize data access and data processing for the scientific community.
majid.ounsy@synchrotron-soleil.fr	Foxtrot	Majid OUNSY (majid.ounsy@synchrotron-soleil.fr)	Raphael GIRARDOT (raphael.girardot@synchrotron-soleil.fr)	SOLEIL	SAXS and XRPDF data reduction	Self installed with a zip file (Java application)	Yes: Image stitching, cartography	http://www.xenocs.com	Full Application, Can be used for interactive analysis, Open Source Version for academic purposes	~50			Java, Python		Fully documented	
majid.ounsy@synchrotron-soleil.fr	Flamenco	Majid OUNSY (majid.ounsy@synchrotron-soleil.fr)	Gregory VIGUIER (gregory.viguier@synchrotron-soleil.fr)	SOLEIL	Soft XRay : ARPES, NEXAFS, Photo Electron Diffraction, Microscopy Scanning Imaging spectroscopies data reduction	Self installed with a zip file (Java application)	Yes		Full Application, Open Source Software, Can be used for interactive analysis				Java			

ANNEX 2 Use Case samples

CALIPSOplus WP24: Demonstrator of a Photon Science Analysis Service

Task 2, Deliverable 24.3

Name of Software/Package: CrystFEL

Data Analysis Use Case Example: MX

Lead: DESY

Introduction

CrystFEL is a suite of programs for processing diffraction data acquired "serially" in a "snapshot" manner, such as when using the technique of Serial Femtosecond Crystallography (SFX) with a free-electron laser source. CrystFEL comprises programs for indexing and integrating diffraction patterns, scaling and merging intensities, simulating patterns, calculating figures of merit for the data and visualising the results. Supporting scripts are provided to help at all stages, including importing data into CCP4 for further processing.

CrystFEL relies on the availability of one or more out of CCP4, XDS, DiRAX, FELIX, MOSFLM for indexing. Other dependencies are readily resolved by standard packages usually available from standard linux distributions.

Description

CrystFEL is a suite of programs for processing Bragg diffraction data acquired with a free electron laser in a "serial" manner. Some of the particular characteristics of such data which call for a specialised software suite are:

- Each crystal is used for only one exposure, and there is no oscillation, rotation nor a large bandwidth or divergence. Therefore, many or all reflections are partially integrated.
- The crystals might be very small and the illumination highly coherent, leading to significant Fourier truncation effects on the detector.
- Many patterns, numbering tens of thousands or more, are required, so high throughput automated processing is important.
- The crystal orientations in each pattern are random and uncorrelated, which leads to special considerations during scaling and merging.

CrystFEL includes programs for simulating and processing patterns subject to the above characteristics. For a description of the core components see http://www.desy.de/~twhite/crystfel/manual.html

Additional components are described under http://www.desy.de/~twhite/crystfel/programs.html

Requirements

Prerequisites:

- HDF5 > 1.8.0.
- FFW3
- GSL
- GTK, Cairo, Pango, GDK-bixbuf, libPNG, libTIFF, zlib

All of the prerequisites are available as standard packages in linux distributions.

Optional indexing applications:

- MOSFLM
- CCP4
- XDS
- Felix
- DirAx

At least one of the indexing routines need to be installed. It’s advisable to have more than one of those.

Optional:

- OpenCL for GPU accelerated simulation of diffraction patterns.

Optionally, you can add OpenCL for GPU accelerated simulation of diffraction patterns. sed.

Software, Licenses, Hardware

Code: is available from http://www.desy.de/~twhite/crystfel/download.html or https://stash.desy.de/projects/CRYS/repos/crystfel/browse both for stable releases and development version.

Licenses

- CrystFEL GPL v3
- MOSFLM: unknown
- CCP4: The suite is available without cost to academic and non-profit institutions who will not be using the CCP4 package for commercial activities, subject to a completed CCP4 Academic License being returned to the CCP4 secretary. See http://www.ccp4.ac.uk/ccp4license.php
- XDS: XDS is free of charge for non-commercial applications
- Felix: unknown
- DirAx: unknown

Apart from CrystFEL itself none of the applications is actually open source.

- MOSFLM actually doesn’t declare a particular license, neither on the web-page nor inside the downloadable package. Source code is available.
- CCP4 requires a signed license formed. Usage is free for academic activities only. Source code is available.
- XDS is free for academic research. No source code available. XDS has an expiry date. Deployment needs to be updated accordingly.
- Felix is available as binary upon request from the authors. No specific license is declared in the corresponding publication
- DirAx does not specify a license. No source code available, only linux binaries for download.

References for the indexing packages:

- MOSFLM: https://www.mrc-lmb.cam.ac.uk/harry/imosflm/ver721/introduction.html
- CCP4: http://www.ccp4.ac.uk/
- XDS: http://xds.mpimf-heidelberg.mpg.de/
- Felix: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5541352/
- DirAx: http://www.crystal.chem.uu.nl/distr/dirax/install.html

The licenses for the indexing programs, in particular for the CCP4 package, make a redistribution of the entire framework at least difficult. The CCP4 license (ftp://ftp.ccp4.ac.uk/ccp4/academic_software_licence.pdf) explicitly states that “the Licensee may not distribute any CCP4 Application or any Derived Work based on any CCP4 Application to any third party, or share their use with any third party (whether free of charge or otherwise)”. Strictly obeying to the conditions, access to a packaged/containerized application framework has to be limited to scientists known to have agreed to the specific terms and only using it for purely academic approaches.

Hardware requirements:

CrystFEL has no special hardware requirements. The use of GPGPUs to accelerate the simulation of diffraction patterns is optional. Some parts of the package/pipeline have high memory demands. With less than 8GB per core the software might easily consume more than the available memory.

I/O (volumes, rates, formats)

CrystFEL uses mostly HDF5, but also support standard MX formats like cbf or mtz. Data volumes per dataset range between a few GB and a few 10TB. CrystFEL is multi-threaded and some of the indexing algorithms (e.g. XDS) are supporting hybrid OpenMP and MPI. The i/o rates hence depend on the available hardware (number of cores and free memory), the indexing program used, and the size of the dataset.

Application Programming Interfaces (if any)

none.

Anything else

Nothing to add.

Reference Data

For deployment tests we have been using the data referenced on the CrystFEL tutorial page, in particular the dataset available under http://www.cxidb.org/id-21.html. Corresponding publication: Liu et al., Science 342 (2013) p1521. Doi: 10.1126/science.1244142

Sample Data

Vast amount of sample data are available on http://www.cxidb.org/

Sample scripts/procedure/notebook/recipes

Described in detail on the CrystFEL tutorial page: http://www.desy.de/~twhite/crystfel/tutorial.html. Documentation and tutorials are excellent and very easy to reproduce even for very inexperienced users.

Platform(s)

- Preferably any (recent) Linux distribution. Runs on Intel, AMD or Power based platforms. Due to memory requirements ARM has not been investigated.
- Works for Mac OSX as well.
- Optional support for GPGPUs using OpenCL. So far only tested on Nvidia GPGPUs.
- Might also be capable of using Intel MICs but hasn't been tested.
- No support for Windows.

Benchmarks

Results

The runtime behavior of CrystFEL has been investigated particular in view of compute requirements. The tests were based on the well documented tutorials (see http://www.desy.de/~twhite/crystfel/tutorial.html).

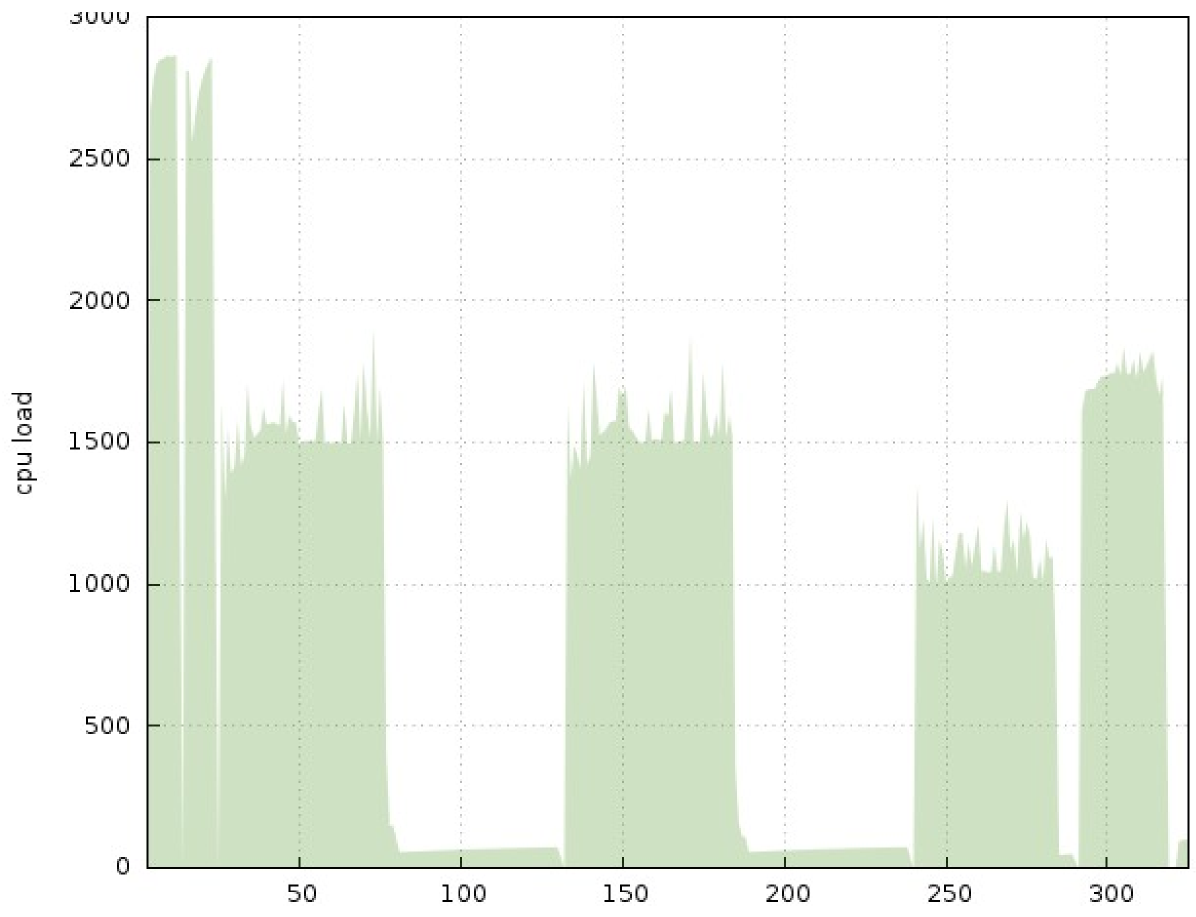
The dataset used is available as a 40GB tarball, containing 5775 images in HDF5 format. Each individual image has a size of roughly 6.6MB. The dataset provides a very typical example.

The tutorial was executed in various different environments. Running a containerized version of CrystFEL (docker, singularity) results in a virtually identical execution time. We therefore just report benchmarks for a bare-metal use case.

CrystFEL is in most parts embarrassingly parallel, and in some parts entirely serial. The application is however currently not MPI-capable. Scalability is hence dictated by the number of images versus the number of cores. The actual cpu load running data analysis on 32 cores is shown below to illustrate the resource consumption for the individual processing steps.

The total execution time for the entire tutorial (all indexing and integration steps but omitting visual, manual interventions) has been obtained depending on the number of cores:

Number of cores	1	2	4	8	16	32	64
Total execution time [s]	77585	40706	21050	12390	6006	3787	3270
Speedup	1.0	1.9	3.7	6.3	12.9	20.5	23.7



Running on 64 cores, the total execution time is roughly 3300 seconds, or a little more than 0.5s per image. Each image is processed 6 time during individual steps of data analysis. In average processing spends 0.09s per image and task.

Data locality is hence crucial. Fetching the tarball from cxidb takes more than 30 minutes, more than 50% of the total execution time on 64 cores. Fetching the tarball via https from a local cloud-store or via cvmfs reduces the transfer and extraction time to roughly 480s, which still contributes significantly to the execution time, but appears reasonable enough. Hosting the individual images for example on a cloud-store and accessing the data via https/davs is however not an option at all. Davs is poorly suited for multi-core access, stats on files are way too expensive and caches too small to allow efficient access. Tests using davs via fusemounts increased the execution time beyond any acceptable limit. It would be interesting to see the behavior on a combined HDF5-file served by HDFserver, which will be tested at a later stage.

The dataset used for the benchmarks is reasonably representative. However, some serial MX datasets are orders of magnitude larger (up to 200TB) and diffract to considerably higher resolution. The crystal characteristics (e.g. the lattice and unit cell dimension) are likewise affecting the number of Bragg reflections and hence the time of integration per frame/crystal. In particular the memory consumption, which was negligible for the test case, can easily exceed 16GB per core.

Deployment

Apart from potential licensing issues (see above) the software and optional components can readily be deployed at any site.

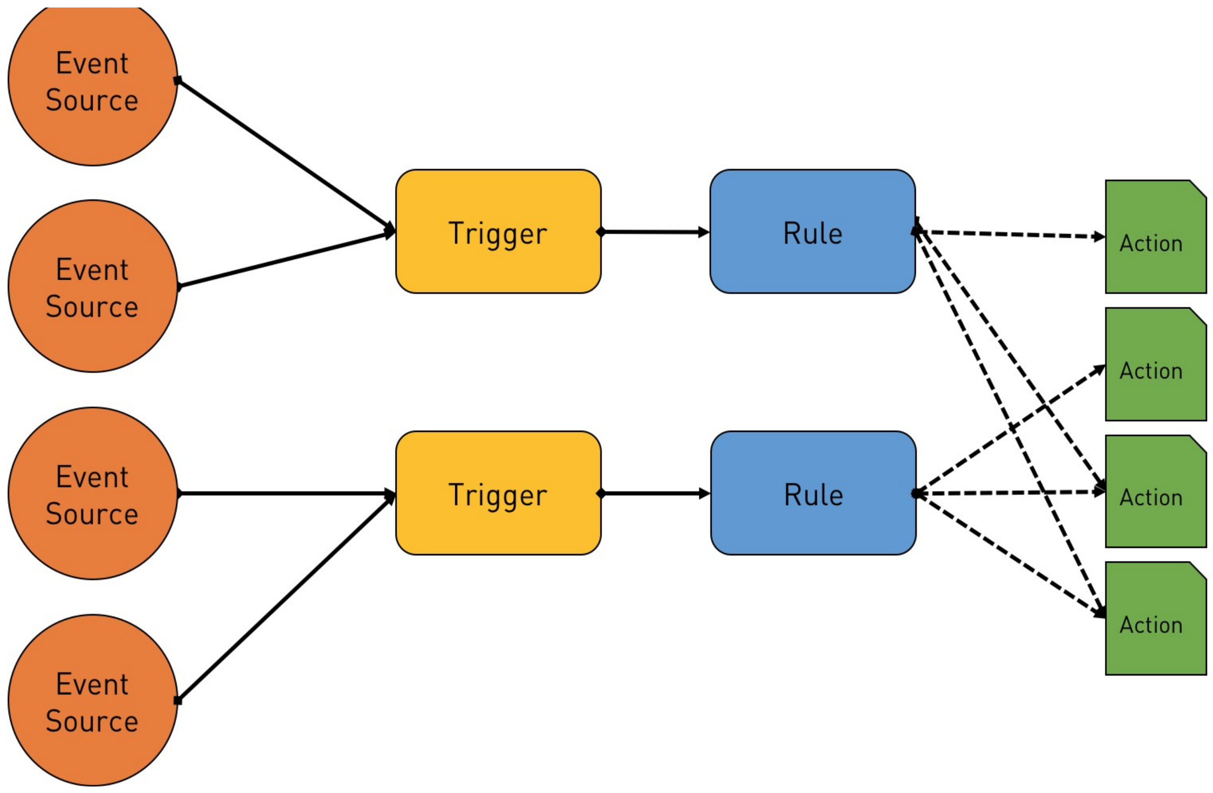
CrystFEL and some of the optional components are available as

- Packages (Centos 7, Debian)
- Dockerfile (recipe)
- Docker container (Centos 7 core image plus MPI components)
- Singularity container
- Sources
- Binary package (tar.gz) for Centos 7

First steps to wrap the application into a Jupyter notebook have been taken (see below). Some processing tasks can be easily embedded and executed for example on cloud function server fully exploiting elasticity of cloud infrastructures. The visual inspection of intermediate results however relies on a number of different tools, like perl, c+gtk, matplotlib. Embedding these into a jupyter notebook doubtlessly requires some adoption.

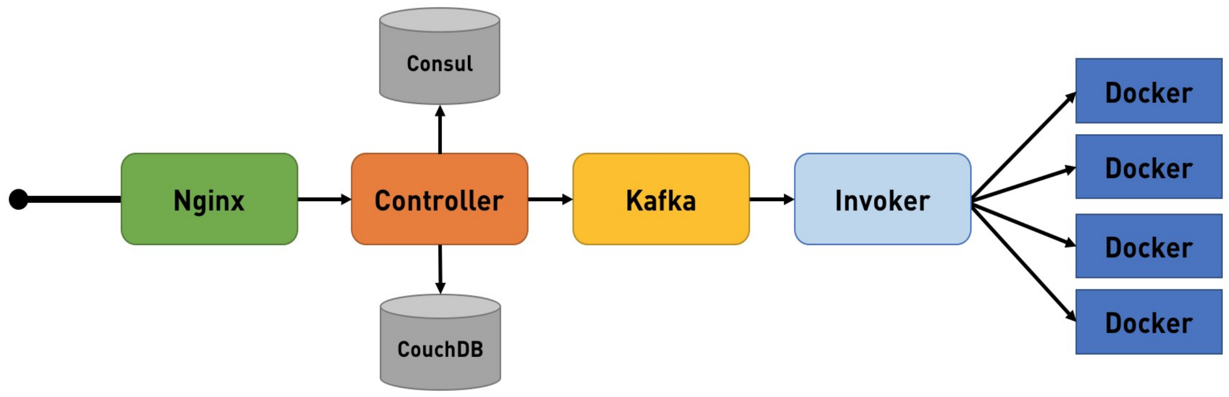
Concept of deploying CrystFEL on elastic cloud resources

Cloud function services (aka serverless cloud platforms) allow providing functions in a fully scalable manner. We are currently experimenting with OpenFaaS (https://github.com/openfaas/faas) and openWhisk (https://openwhisk.apache.org/) implementations, focusing on openWhisk.



Source: https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/

Both allow elastic deployment of functions in combination with docker swarm, kubernetes on an openstack platform. In essence, the function server launches a docker container for each function invocation. In combination with kafka (https://kafka.apache.org/) and prometheus (https://prometheus.io/) it allows to invoke functions on incoming messages or events, e.g. automatically process incoming images using CrystFEL (or any other pipeline like image calibration).



Source: https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/

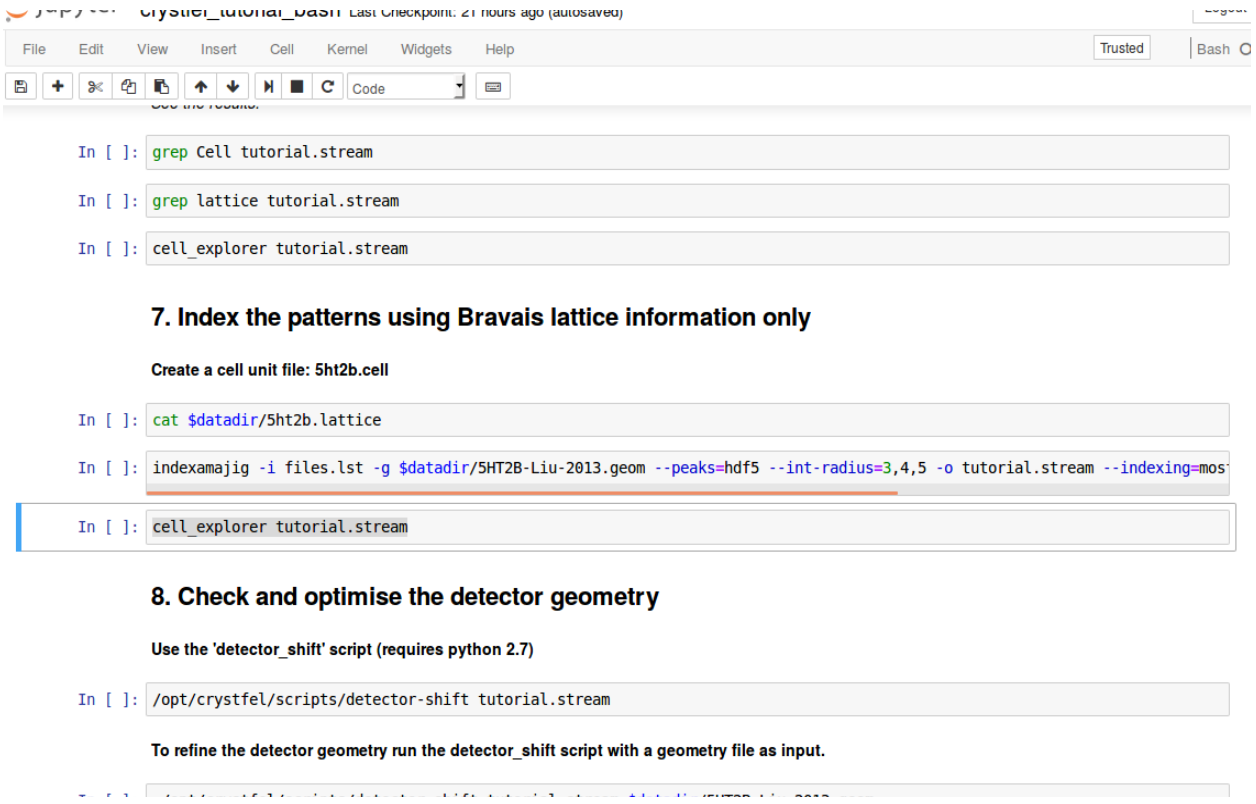
The entire infrastructure (openstack+kafka+prometheus+openWhisk) has been configured and is ready to deploy CrystFEL functions. Currently we are facing two minor problems: one problem is the aforementioned requirement of data locality. It's actually not foreseen to embed any network drives inside a function service. As a workaround, NFS shares have been embedded to allow local data hosting. The other problem relates to the partially non-redistributable nature of the software (e.g. ccp4). openWhisk per default only supports public docker-images hosted on dockerhub. We therefore stripped the docker image to contain exclusively GPL licensed software and use mosflm (https://www.mrc-lmb.cam.ac.uk/harry/imosflm/) for all indexing tasks. The stripped docker image will then be published on dockerHub enabling deploying CrystFEL as a serverless cloud function invokable from Jupyter notebooks or plain from the command line allowing to establish "CrystFEL as a service" on a cloud based JupyterHub instance. This is still work in progress.

Providing CrystFEL as a service using Jupyter Notebooks

It turned out to be almost trivial providing CrystFEL embedded into a Jupyter Notebook (thanks to colleagues from European XFEL). The availability of a bash kernel (http://jeroenjanssens.com/2015/02/19/ibash-notebook.html) greatly facilitates the task. The installation of the bash kernel requires a few simple steps, which can also be done in user space:

- pip install bash_kernel --user
- python -m bash_kernel.install
- jupyter notebook --generate-config

Starting a jupyter server remotely in user space then allows to run CrystFEL painlessly in a web browser:



```
In [ ]: grep Cell tutorial.stream
In [ ]: grep lattice tutorial.stream
In [ ]: cell_explorer tutorial.stream
```

7. Index the patterns using Bravais lattice information only

Create a cell unit file: 5ht2b.cell

```
In [ ]: cat $datadir/5ht2b.lattice
In [ ]: indexamajig -i files.lst -g $datadir/5HT2B-Liu-2013.geom --peaks=hdf5 --int-radius=3,4,5 -o tutorial.stream --indexing=mos
In [ ]: cell_explorer tutorial.stream
```

8. Check and optimise the detector geometry

Use the 'detector_shift' script (requires python 2.7)

```
In [ ]: /opt/crystfel/scripts/detector-shift tutorial.stream
```

To refine the detector geometry run the detector_shift script with a geometry file as input.

It might become necessary to adopt the small graphical tools to visualize peak detection and cell refinement into Jupyter embedded applications. That's currently being investigated.

Having access to a compute cluster, the task of processing the 5775 images from the tutorial dataset can easily be split into small problems. For example on the DESY HPC platform the full integration of images can be done in less than 5 minutes from within a Juypter notebook, currently only limited by the number of concurrent process allowed on the platform. This reduces the computational time to roughly 0.1s/image and we expect that releasing the process limitation will further speed up processing to 0.01-0.05s/image. It shows that the processing pipeline is very well suited for cloud function based elastic services and that on-demand real time processing is quite possible without major developments.

User Support

CrystFEL comes with very detailed tutorials and manuals. The documentation is detailed enough to reproduce data analysis on existing openly accessible datasets. In addition, hands-on tutorials are being offered.

Future Developments, Roadmap

We are currently investigating the use of Function as a Service (openFaaS, openWhisk) to distribute the indexing part on arbitrary compute platforms, in particular openstack in combination with dockerswarm/kubernetes. This allows elastic scaling depending on the size of datasets in a very convenient manner. In combination with kafka/prometheus it also offers “on-the-fly” data processing as data arrive.

References

Primary references to cite when using CrystFEL and/or some of the optional components:

1. **CrystFEL**: T. A. White, R. A. Kirian, A. V. Martin, A. Aquila, K. Nass, A. Barty and H. N. Chapman. "CrystFEL: a software suite for snapshot serial crystallography". J. Appl. Cryst. 45 (2012), p335–341. doi:10.1107/S0021889812002312

2. **XDS**: Kabsch, W. (2010a). XDS. Acta Cryst. D66, 125-132.
3. **Felix**: K. R. Beyerlein, T. A. White, O. Yefanov, C. Gati, I. G. Kazantsev, N. F.-G. Nielsen, P. M. Larsen, H. N. Chapman and S. Schmidt: “FELIX: an algorithm for indexing multiple crystallites in X-ray free-electron laser snapshot diffraction images”. J. Appl. Cryst. (2017). 50, 1075-1083. https://doi.org/10.1107/S1600576717007506
4. **CCP4**: M. D. Winn et al. Acta. Cryst. D67 , 235-242 (2011): "Overview of the CCP4 suite and current developments". doi:10.1107/S0907444910045749]
5. **MOSFLM**: T.G.G. Battye, L. Kontogiannis, O. Johnson, H.R. Powell and A.G.W. Leslie (2011), Acta Cryst. D67, 271-281.
6. **DirAx**: Duisenberg, A.J.M.(1992). J. Appl. Cryst. 25, 92-96

Ptycho Shelves
Data Analysis Use Case Example

Lead: Paul Scherrer Institut

Introduction

Ptychography is a technique that combines scanning and coherent diffractive lensless imaging. In its more conventional configuration a sample is scanned through a beam and for each scanning point a diffraction pattern is measured, making sure to have some degree of overlap between neighboring scanning positions. The diffraction patterns are then fed to a iterative reconstruction algorithm to reconstruct the complex-valued incident wavefield and complex-valued sample transmissivity. In doing this one does away with image forming optics and can obtain imaging resolution not bounded by their quality and aberrations.

Ptycho Shelves is an innovative and conceptually-simple modular framework for ptychography reconstructions. With the constant development of algorithms for ptychography there is currently a plethora of different options for reconstruction algorithms, or engines. Furthermore, different engines offer different features for correction of experimental imperfections. Hence it becomes important to be able to stack together different modules such that different engines can be executed in series, with the output of one algorithm being the initial guess of the following. Along with the package come several of currently used ptychography algorithms and the framework allows for further expansion.

At some experimental stations there are multiple X-ray detectors available, or different options for computing sample positions based on several interferometric readings. Ptycho shelves also offers flexible modules for switching between different detectors, data preparation functions, or computations of sample positions.

Description

Ptycho-shelves is organized in MATLAB packages, which are used to organize the modules, *e.g.* core, engine, detector, etc. The different engines include: prototype

MATLAB scripts, some of them include MEX functions for speed; and the cSAXS high-performance C++ CPU code which is used in production at the cSAXS beamline. It allows preprocessing of data and ptychographic reconstruction with various engines.

Documentation will be provided as a general description of the different included engines, a step-by-step use guide with simulated data and a sample dataset.

Requirements

Basic prototype engines require Matlab 2018a, GPU enhanced codes require CUDA and Matlab parallel toolbox. The C++ CPU engine can be compiled and run independently with standard libraries.

Prerequisites for Ptycho Shelves framework:

- Matlab >= 2018a

Prerequisites for C++ engine:

- GCC >= 6.2.0
- Hdf5_serial >= 1.8.18
- Intel MKL >= 17.1
- openMPI >= 2.0.1

Prerequisites for GPU engine:

- CUDA >= 8.0
- Matlab parallel toolbox

Prerequisites for data cSAXS preparation:

- Hdf5_serial >= 1.8.18
- Python2.7 >= 2.3.0

Matlab functions need the cSAXS Matlab base package https://www.psi.ch/sls/csaxs/software.

Software, Licenses, Hardware

The software license can be found in
https://www.psi.ch/sls/csaxs/ComputingEN/License.txt

Hardware requirements

There Ptycho Shelves framework has no special hardware requirements. The C++ CPU engine was optimized for Intel Xeon processors and we strongly recommend to use modern processors with SIMD instructions AVX-512. The memory requirements will depend on size of the processed dataset but also on the used reconstruction method and number of reconstructions modes, i.e. coherence modes in illumination or number of separated object reconstructions. We strongly recommend to have at least 16GB RAM available.

The GPGPUs accelerated engines require at Nvidia cards with at least computational compatibility 2.0 and more than 8GB of internal GPU memory.

I/O (volumes, rates, formats)

The format for the raw input data for the Ptycho Shelves framework is defined for several application cases, i.e. different detectors and positioning systems, at cSAXS but ultimately can be adapted by the end user to new formats. The loaded raw data are stored to HDF5 with customized that can be used as inputs to the C++ CPU engine. The outputs are stored into a HDF5 format as well. Details of the input and output HDF5 structures will be provided in the final documentation.

Application Programming Interfaces (if any)

The C++ engine can called separately via a command line API. This allows remote batch processing and automatization. The Ptycho Shelves package does not contain any API and it is controlled by configuration scripts written in Matlab.

Reference Data

Sample Data

Ptychography simulated and experimental data will be provided in the environment.

Sample scripts/procedure/notebook/recipes

Ptycho Shelves toolkit is fully configurable via simple Matlab-based configuration scripts. We will provide a well commented example of such configuration script that can be simply adjusted by users for their specific needs.

Benchmarks

There is no need for benchmarks of the Ptycho Shelves framework, because it only provides a shared and robust environment for calling multiple ptychographic engines that will perform the computationally demanding ptychography reconstruction.

Performance of the engines will be benchmarked for several common application cases.

Platform(s)

Ptycho Shelves package is based on the Matlab environment, which makes it platform independent as long as Matlab with the required toolboxes is available. The implemented matlab engines may need to be recompiled for the specific platform. The high performance C++ CPU and GPGPU accelerated engines are supported only for recent Linux distributions.

Deployment

The source codes of the ptychographic engines will be provided along with a tutorial, where the necessary compilation steps will be described. Loading of the raw measurements needs to be adjusted for the specific experimental environment, however we will provide a way to generate artificial datasets that are important for initial tests.

User Support

Clear and detailed documentation will be provided so that the examples can be run in the to-be-defined environment. The documentation should provide sufficient details to allow users run their own reconstructions of the provided example datasets.

Future Developments, Roadmap

As new algorithms that should provide accelerated convergence or other ptychographic modalities such as extended depth of focus are developed by our group they will be deployed as new engines in this framework. They can be incorporated into the CALIPSO environment as they become available.

Name of Software/Package: Savu

Data Analysis Use Case Example: Tomography

Lead: Diamond Light Source

Introduction

Savu is a Python package to assist with the processing and reconstruction of parallel-beam tomography data. The project originated in the Data Analysis Group at the Diamond Light Source (UK synchrotron) to address the growing, and increasingly complex, needs of the tomography community.

Designed to allow greater flexibility in tomography data processing, Savu is capable of processing N-dimensional full-field tomography and mapping tomography data, along with concurrent processing of multiple datasets such as those collected as part of a multi-modal setup. Savu is currently in use across the tomography beamlines at Diamond to reconstruct both full-field tomography data and multi-modal, mapping tomography data.

Description

Savu is an object-oriented Python framework, with a modular plugin architecture. The framework handles the movement of the data, in serial or parallel on a local PC or cluster, and each plugin performs a specific independent task, such as correction, filtering, reconstruction. Savu process lists, tailored to a specific experiment and passed to the framework at runtime along with the data, detail the processing steps that are required. The process list is created using the Savu configurator tool, which stacks together plugins chosen from a repository.

Features

- Full-field and mapping tomography data processing
- Time resolved imaging
- multi-modal data processing
- Absorption, fluorescence, diffraction and ptychography data processing
- Handles N-dimensional data and multiple datasets
- Supports multiple data formats
- Runs in serial or parallel on your local machine
- Runs in parallel across a cluster
- Supports very large data processing with parallel HDF5 (not limited by RAM)

- Allows flexible data slicing (e.g. alternate between projection and sinogram processing)
- Plugin architecture with CPU and GPU plugins
- Processing tailored to a specific experimental setup
- Easy integration of new functionality

Savu is an open-source project, freely available on Github:

https://github.com/DiamondLightSource/Savu

Requirements

Software, Licenses, Hardware

Software: Most packages required by Savu are installed in the conda environment, packaged with Savu, during the installation process. Some of the these packages (hdf5, h5py and mpi4py), will be built against MPI libraries, via conda recipes during the installation process. The only other software requirement for the Savu framework is:

- MPI : Tested and working with OpenMPI >= 1.8.5

Other packages required for a full range of plugins are:

- CUDA : Tested and working with 7.0
- FFTW3

License: Savu is dual licensed under the Apache V2 and GPL V3 license (GPL V3 license is due to the use of The ASTRA Toolbox (https://www.astra-toolbox.com/) in two plugins, which are optional).

Hardware: GPUs are required for a full range of plugins, but are not mandatory. Savu is heavily I/O dependent and so for large data, cluster based processing, performance is significantly improved with infiniband network connection and has been shown to work well with GPFS and Lustre file systems.

I/O (volumes, rates, formats)

Savu uses parallel HDF5 to read and write data directly from file at every processing step (plugin). Since there is no RAM limitation, the size of data that can be processed depends on the hard drive storage available. Typical datasets range from a few GB to a few TB. The size of the data is typically increased on processing, and each intermediate processing step will create at least one HDF5 output file. These intermediate files can be stored in a temporary location. The final output is typically HDF5, but it is possible to have a range of input and output formats.

The I/O rates are high, but they will depend on the size of the data, the number of cores and the chosen processing steps.

Application Programming Interfaces (if any)

None.

anything else

Nothing to add.

Reference Data

Sample Data

Small test data and example process lists, for both full-field and mapping tomography, are packaged with Savu. A typical, full-sized, full-field tomography data set is available on zenoda https://doi.org/10.5281/zenodo.1181825.

Sample scripts/procedure/notebook/recipes

All currently available documentation for Savu can be found on readthedocs:

https://savu.readthedocs.io/en/latest/

Platform(s)

- Any Linux distribution.

Benchmarks

Results

Savu is a very flexible framework and provides a vast array of processing options for a number of different measurements, with processing tailored to a specific experimental setup. The problem is data parallel, and Savu optimises the load balancing based on the number of frames to process for each plugin and the number of MPI processes available.

Benchmarks are provided for a typical full-field tomography dataset (see Sample data above): For the auto-processing we perform on a subset of the data at Diamond and both a simple and complex processing pipeline applied to the full data.

The tests ran on one of our cluster hosts, where each node has 20 cores and 4 GPUs (although the GPUs were not used in these tests). Data is read from and written to a GPFS file system via infiniband network.

Preview Savu processing :

- **Auto-processing** : Approx 60 seconds on 1 node (This takes 9 seconds if you know the centre value, or somewhere in between if you have an idea of it and reduce the search region).
    1. Dark and flat field correction

2. Auto-centering
3. Tomopy Gridrec reconstruction

Full Savu processing :

- **Simple processing :** Approx 40 seconds on 9 nodes (180 cores)
    1. Dark and flat field correction
    2. Tomopy Gridrec reconstruction.
- **Complex processing :** Approx 106 seconds on 9 nodes
    1. Zinger Removal (high frequency scatter spots)
    2. Dark and flat field correction
    3. Distortion correction (using pre-calculated coefficients)
    4. Contrast enhancement
    5. Ring artefact removal
    6. Tomopy gridrec reconstruction.

Deployment

Savu is available to download and install from Github:

https://savu.readthedocs.io/en/latest/dls_installer/

This requires a tar.gz file to be downloaded that contains a bash script and details of packages and conda recipes required for the installation process. On executing the bash script, Miniconda is downloaded and installed, with all subsequent software packages installed (and built in some cases) into the conda environment. The installation process finishes with a range of tests, which make use of the test data and process lists that are packaged with Savu. To use Savu on a cluster, it may be necessary to update the launcher scripts to be compatible with the chosen scheduler and available resources.

User Support

Simple user documentation is available on readthedocs.

Future Developments, Roadmap

Continued development of experimental plugins, and performance optimisation.

References

The primary reference for the Savu framework is currently the archive paper:

Wadeson, N., & Basham, M. (2016). Savu: A Python-based, MPI Framework for Simultaneous Processing of Multiple, N-dimensional, Large Tomography Datasets. arXiv preprint arXiv:1610.08015.

All references for a particular Savu processing pipeline are available in the output nexus file. Savu comes with a tool to run against the nexus file to convert the entries into BibTex and EndNote formats

pyFAI
Data Analysis Use Case Example

Lead: ESRF

Introduction

2D area detectors like CCD or pixel detectors have become popular in the last 20 years for diffraction experiments (e.g. for WAXS, SAXS, single crystal and powder diffraction). These detectors have a large sensitive area of millions of pixels with high spatial resolution. The software package pyFAI has been designed to reduce SAXS, WAXS and XRPD images taken with those detectors into 1D curves (azimuthal integration) usable by other software for in-depth analysis such as Rietveld refinement, or 2D images (a radial transformation named *caking* in FIT2D). As a library, the aim of pyFAI is to be integrated into other tools with a clean pythonic interface or being used directly with the Jupyter interface. However pyFAI features also command line and graphical tools for batch processing, converting data into *q-space* (q being the momentum transfer) or 2θ-space (θ being the Bragg angle) and a calibration graphical interface for optimizing the geometry of the experiment using the Debye-Scherrer rings of a reference sample.

Description

PyFAI is the Python library for Fast Azimuthal Integration. The complete documentation is available online at:

- http://www.silx.org/doc/pyFAI/dev/
- http://pyfai.readthedocs.io/en/latest/

Requirements

The pyFAI is pip-installable with all dependencies available from http://pypi.org

Software, Licenses, Hardware

- Git repository: https://github.com/silx-kit/pyFAI
- SW (OSS) Licenses: MIT license
- Special HW requirements: GPU accelerated via OpenCL (optional)

I/O (volumes, rates, formats)

The pyFAI librarie itself does not handle I/O and relies on:

- h5py for reading HDF5 files
- FabIO for reading all other file-formats
- Numpy array saved on the disk are also possible

Application Programming Interfaces (if any)

While the API is described in: http://www.silx.org/doc/pyFAI/dev/api/modules.html, the basic usage is described in a couple of cookbooks like: http://www.silx.org/doc/pyFAI/dev/usage/cookbook/integration_with_python.html

Anything else

Advanced tutorials using the jupyter notebook are available at : http://www.silx.org/doc/pyFAI/dev/usage/tutorial/index.html

see the project source for jupyter notebooks (ipynb) files

Reference Data

Most tutorials in https://github.com/silx-kit/pyFAI/tree/master/doc/source/usage/tutorial are self-contained and will download the needed data from internet as needed. The proxy setting may be adjusted for this.

Sample Data

Sample data are provided as part of the tutorials.

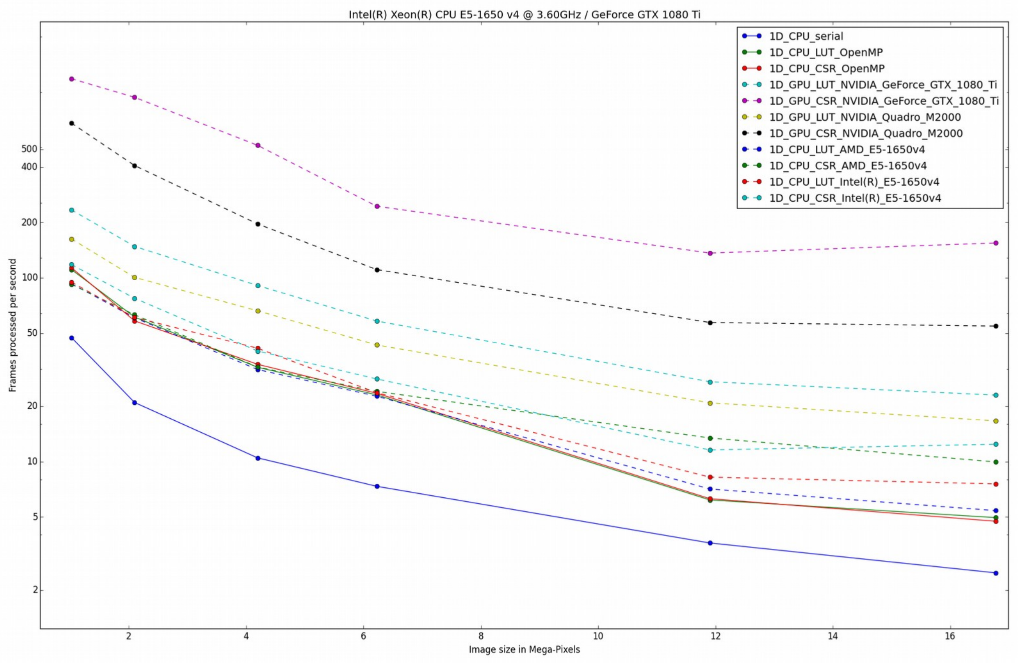
Sample scripts/procedure/notebook/recipes

Every tutorial is provided as a web-page and as a self-contained jupyter notebooks.

Benchmarks

Python being an interpreted programming language it is slower than most compiled languages. This is why pyFAI heavily relies on compiled extensions (using OpenMP) and off-load the most time-critical calculation on GPU using OpenCL (if possible) to provide state of the art performances with the convenience of an interpreted language.

Benchmarks are available within the documentation at http://www.silx.org/doc/pyFAI/dev/pyFAI.html#performances-and-migration-to-native-code



Platform(s)

PyFAI runs where the Python scientific stack (SciPy) is available:

- Windows
- Linux
- MacOS

While the continuous integration is validating only on amd64 hardware, the library is known to be running running on Intel i386, ARMv7 and ARMv8, MIPS32 and PowerPC64.

Results

Thanks to its speed, pyFAI has been used in some of the largest experiments involving X-Ray diffraction like: https://www.nature.com/articles/nature16060

Deployment

The software is available as source and pip-installable (pip install pyFAI) on any architecture. Binary wheels are provided for most common architectures. Other will require the presence of a C-compiler.

Packages/Container/notebook ...

- Debian packages (https://packages.debian.org/sid/pyfai)
- Conda packages (https://anaconda.org/conda-forge/pyfai)
- Red-hat 7 packages (http://pubrepo.maxiv.lu.se/rpm/el7/x86_64/)
- PIP: https://pypi.org/project/pyFAI

User Support

A mailing list, pyfai@esrf.fr, is publicly available. This is the best place to ask questions: the authors and many advanced users are there. To subscribe to this mailing list, send an email to pyfai-subscribe@esrf.fr.

The volume of email on the list remains low, and is archived at: http://www.edna-site.org/lurker. There are information about release of the software, new features available and meeting announcements. The archive also provides a knowledge-base of most frequently asked question before it gets integrated into the documentation.

If you think you are facing a bug, the best is to create a new issue on the GitHub page (you will need a GitHub account for that).

Direct contact with authors is discouraged: pyFAI is open source software that we develop to aid the research community in doing what they do best. While we do enjoy doing this, we would not be able to dream of spending nearly as much time with pyFAI as we do if it wasn’t for your support. Interest of the scientific community (via a lively mailing list) and citation in scientific publication for our software is one of the main criterion for ESRF management when deciding if they should continue funding development.

Future Developments, Roadmap

PyFAI is currently in version 0.16 (development branch) which mean some important features are still missing according to the authors. Nevertheless there are many unique features already available (and production ready) like:

- Any detector geometry: most X-ray detectors are already tabulated in pyFAI
- Anywhere in space: pyFAI is not limited to orthogonal nor centered detector setup
- Detector mounted on goniometers with hundreds of images from various positions

The version 1.0 may be:

- Equally usable from CLI, GUI or notebooks
- Provide proper error propagation
- Offer (user-defined) geometry refinement

But the roadmap of pyFAI may not be the one of ESRF beamlines… this is why pyFAI is an open-source project accepting pull-requests for your specific needs (if it is of general interest).

PyMca
Data Analysis Use Case Example

Lead: ESRF

Introduction

X-ray microscopy is a common technique at synchrotron facilities that can be performed in different modes (scanning or full field) using a large variety of techniques (X-ray fluorescence, X-ray Absorption Spectroscopy, X-Ray Diffraction…) often combining more than one technique simultaneously. There is a clear need to have applications able to explore X-ray microscopy datasets. PyMca is one of them. The programs aXis2000 and MAPS are notable alternatives based on a commercial language (IDL). A port of MAPS to Python is going on.

PyMca was initially developed to fulfill user needs related to X-ray Fluorescence Analysis (XRF) and it can be a considered a reference in that field like AXIL, GeoPIXE or GUPIX. It offers all what users can expect in that field (qualitative and quantitative analysis, interactive and batch processing, large variety of data formats support -including HDF5 NeXus-) while being the only free and open source alternative among the reference applications.

Besides those specific applications, its visualization and data exploring capabilities make of PyMca a Swiss Army knife for the synchrotron user

Description

Set of Python modules and applications for interactive and batch processing of X-ray Fluorescence and X-ray microscopy data.

Some of the capabilities of the program have been published in peer reviewed journals:

- Solé et al. A multiplatform code for the analysis of energy-dispersive X-ray fluorescence spectra. Spectrochimica Acta Part B 62 (2007) 63–68. DOI: 10.1016.j,sab.2006.12.002

- Cotte et al. Watching Kinetic Studies as Chemical Maps Using Open-Source Software. Analytical Chemistry 88 (2016) 6154–6160. DOI: 10.1021/acs.analchem.5b04819
- Nikbakht et al. An efficient approach to integrated MeV ion imaging. Ultramicroscopy 186 (2018) 112–119. DOI: 10.1016/j.ultramic.2017.12.014

Additional documentation is available at http://www.silx.org/doc/PyMca/dev/

Requirements

PyMca is pip-installable with all dependencies available from http://pypi.org

Software, Licenses, Hardware

- Git repository: https://github.com/vasole/pymca
- SW (OSS) Licenses: MIT license
- Special Hardware requirements: None
- Special Software requirements: C++ compiler to build the package from source

I/O (volumes, rates, formats)

Most common data formats supported. HDF5 and NeXus supported since 2010.

Application Programming Interfaces (if any)

Most of the modules building PyMca can be executed as stand-alone scripts illustrating the usage of that particular module. More specific aspects of the code are available at the documentation pages.

Reference Data

The code is shipped with data to perform basic tutorials and tests.

Sample Data

The code is shipped with data to perform basic tutorials and tests.

Sample scripts/procedure/notebook/recipes

Additional documentation is available at http://www.silx.org/doc/PyMca/dev/

Benchmarks

Lorem ipsum dolor sit amet

Platform(s)

PyMca runs wherever the Python package numpy is available. The use of the graphical interface requires one of PyQt5, PyQt4 or PySide installed. Therefore complete functionality is available in platforms where Qt can be installed:

- Windows
- Linux
- MacOS

While the continuous integration is validating only on amd64 hardware, PyMca is known to be running on Intel i386, ARMv7 and ARMv8, MIPS32 and PowerPC64.

Results

There are many examples of application of PyMca. The first article describing PyMca has been cited more than 800 times.

Deployment

The software is available as source and pip-installable (pip install pymca) on any architecture. Binary wheels are provided for MacOS and windows. Others will require the presence of a C++ compiler.

Packages/Container/notebook …

Besides availability from PyPI, PyMca is available as official package for most common linux distribution (Debian, Ubuntu, Fedora, …)

Ready to use binaries, not requiring any particular installation are provided for windows and MacOS too https://sourceforge.net/projects/pymca/files/pymca/

User Support

A mailing list, pymca-users@sourceforge.net, is publicly available. There are information about release of the software, new features available and meeting announcements. The archive also provides a knowledge-base of most frequently asked questions prior to integration into the documentation.

Bugs should be reported either at the mailing list or creating new issue at the GitHub page https://github.com/vasole/pymca/issues.

Future Developments, Roadmap

With its version published in 2004, PyMca is a mature piece of software. Current efforts are mainly focused on improving the documentation by providing complete on-line tutorials.