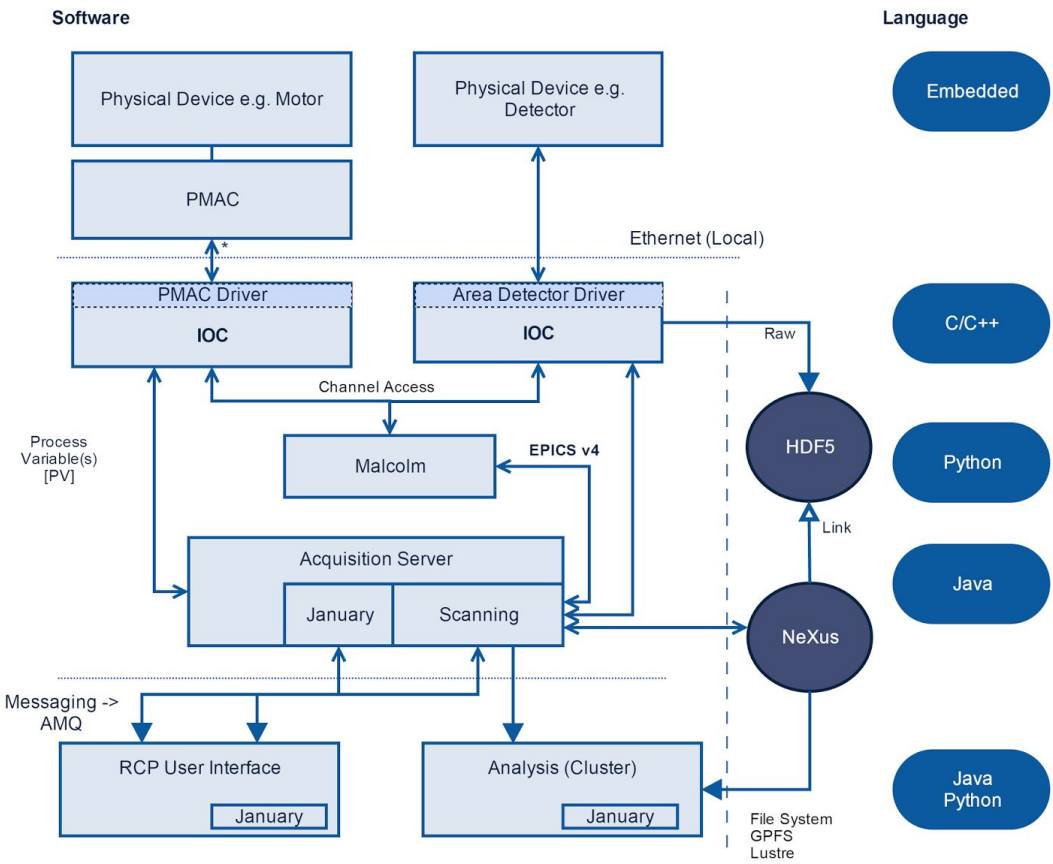
Hello,

I would like to introduce you to a new open source project for scanning hardware and completing scientific experiments. It has been donated to the Eclipse Foundation as an incubation project and has number of interesting features. The original project proposal can be read here: https://projects.eclipse.org/proposals/scanning



Scan Paths

The start of a scan is the scan path. It is a series of n-Dimensional motor/scalar positions provided by a generator pattern - we call them scan point generators. The possible paths are created using a factory, with each path requiring a model. The generators may be nested, limited only by their ability to be addressed. The paths scale to hundreds of millions of points because not all points are created in memory at the start of the scan, so really large scans are possible. The paths are defined in Python to make it easy for scientists to create custom generators on the fly, in the unlikely event that they cannot achieve the type of scan required. Unlimited regions of interest are allowed so the user may for instance create a grid and then draw various shapes to filter the scan points. A simple example of a scan path is a snake scan running over a grid of 2D positions of a stage motors for x and y.

Runnable Devices and Scannable Devices

When it comes time to run the scan, the motors and detectors are represented in code by the simple interfaces IRunnableDevice and IScannable. These classes are each managed by an OSGi service which allows devices to be added by a Spring layer or by creating and registering them with the server in Python. Once they are added, they provide the link to the hardware such as Area Detector or EPICS PVs, you can then start to run scans.

Scanning Algorithms

The scanning algorithm is also part of a factory pattern and allows users to override or create their own. The default algorithm uses the power of Java Executor Service. It splits up the devices by a concept known as level and uses a fully multi-threaded design to move motors in the most efficient way. Motors are moved while detectors are read out and the file is written. For those motors in a 2D scan which have a fast and slow axis on the same EPICS IOC, the power of the scan path design means that the code can look forwards to figure out each fast path segment and run lines close to the theoretical maximum.

User Interface

The project comes with a complete user interface for creating and running scans written in Eclipse RCP. The bundles which contribute the UI are modular and may be removed for those people without an RCP front end. The user interface interacts seamlessly with the python layer giving the user either the ability to submit without scripting or, for advanced users, the ability to copy the scan command into Python. It is easy to create Java-Swing, Python-QT or any front end because the scanning system is driven externally by JSON.

Python and Malcolm

The Scanning project is partly written in Python and Java. Most of the C-python device layer is contributed in a separate project called pymalcolm and the project interacts with Malcolm devices (which are IRunnableDevice) using the EPICSv4 protocol, http://epics-pvdata.sourceforge.net/.

Eclipse January, NeXus HDF5 -SWMR

Scanning includes the implementation of the January concept “ILazyWriteableDataset”. This means that if you are creating a device, you only need depend on January and your control layer (EPICS for instance). The device layer is modular and multi-threaded so it allows the full speed of HDF5-SWMR to be utilised without the need to mix up HDF5 and device code.

Online Analysis

For the first time, online analysis has been considered and designed into a scanning system from the start. This means that online analysis codes running on the cluster can run processing SWMR files on the data during the live data collection. The user interface allows the choose between different analysis or even create your own graphically using DAWN and then deploy the analysis with the scan. This is done efficiently using the new Single Write Multiple Read HDF5 library.

Builds and Tests

Scanning does not skimp on tests, it has hundreds of tests executing different scanning scenarios and different device options. It checks operation of the algorithms with machine topup and beam dump scenarios, it writes HDF5 SWMR files and checks the supported scan point generators. It gives the user interface composites a workover using SWTBot. For the first time this layer has be abstracted into a separately built (travisCi+maven) and tested(travisCI+junit+sonarqube) project which enables anyone to participate and contribute.

Getting Started

If you want to check out ‘Scanning’ and run it, providing you are experienced with targets and products it is easy to run by following these instructions:

https://github.com/eclipse/scanning/blob/master/GETTINGSTARTED.pdf