Flyscan / SOLEIL approach

Florent Langlois
With help from:
S. Poirier
C. Engblom
J. Bisou
S. Zhang

Major issues

Please redistribute the presentation time so that the major issues receive more attention. For example, approx. 3 min per major issues.

Allow different synchronization descriptions (when or on what positions the synchronization events should occur) e.g. involving shutter control, synchronizing with the accelerator events

In some cases there is a need for multiple different synchronization descriptions e.g. for controlling both detector exposure time and shutter opening in one experiment. Also, some experiments require advanced synchronization descriptions involving external events e.g. from the machine timing system. It is currently unavailable in Sardana. How do you cope with this issue? Do you have any user interface that facilitates this kind of configuration? What is your programmatic API for such configurations?

I-Synchronization

- 2 cases:
  - Hardware:
    - eg, for fast shutters, that should be opened/closed at some motor positions
    - use a dedicated HW to send triggers
    - At SOLEIL we use the Pandabox (or its ancestor: Spietbox)
    - This HW is managed by the Flyscan framework (via the plugins)
  - Software
    - For ‘slow’ shutters
    - Hooks plugins used
- No case for accelerator events (ie: Timing)
  - Pump-probe system could be integrated (via développement of new plugins)

Implement trajectory control using pseudomotors. Currently, only possible with motors which actually involve multiple physical actuators.

Currently in Sardana trajectory control is only possible with single-axis motors that involve multiple physical actuators. However, we have observed drawbacks in this approach, leading to inconsistencies in element states. How do you deal with this issues? Do you support motion trajectory controlled continuous scans? Also, pseudo motor and the motor controllers most probably will need to overlap in some sense e.g. some calculations will need to be repeated in both controllers (plugins).

II-Trajectories

- Not sure to get the point …
- Pseudo motors axes can be managed by PowerPMAC controllers (eg: Energy)
  - One or more motors physical motors and one or more pseudomotors managed by one Tango device
  - Managed by a Flyscan plugin
- Trajectories, eg pvt, are supported via Flyscan plugins (eg XPSGroup)
- PowerPMAC also support trajectories, but not yet used in Flyscan

Support for multiple capability controllers i.e motion and trigger gate in the same controller class – use capability composition approach.
eg. configure step per unit (conversion factor) on trigger/gate controller - currently it either needs to be configured in two places or one needs to use Tango interface to read it

In certain instances, like IcePAP or Pmac motion controllers, which have the capability to emit synchronization events, integrating them into Sardana necessitates the development of two controller classes: one for motors and another for trigger/gate functions. In some cases, this may involve creating proxies between the controllers to obtain specific information, such as obtaining step_per_unit for the trigger/gate using a proxy to the motor element. One possibility could be use an underneath singleton class that could be accessed from two controller classes. Do you have such architectural problems? How do you deal with them?

Some controllers may eventually implement optional features, such as power-on functionality, software limits protection, ROIs (Regions of Interest), or returning references to data rather than the data itself. These optional capabilities do not form part of the official API, except for Referable controllers implemented through subclassing. Incorporating these features into the official API would entail exposing unnecessary attributes to users when controllers do not support these capabilities. Ideally this should be solved by classes composition.

III-Motors Controllers

- Again, not sure to get the point…
- I think we don’t get those architectural problems
- Today we do not use motor controller to do triggering in Flyscan
    - But we have looked into, and studied some setups using the PowerPMAC as a position triggering source

Improve support high speed scans - improve data flow to avoid bottlenecks e.g. pseudo counters, data storage, etc.

Historically, we have experienced dead times in continuous scans that occur after the scan execution, just before the macro's conclusion. We suspect these delays could be due to factors like data storage delays, event accumulation on the client side, or data processing on the MacroServer side.

Recently, in Sardana, we have prepared PoC that involves directly pushing data to an in-memory database on the Pool side instead of relying on Tango events to traverse through the MacroServer and the recorders. This approach minimizes stress on the MacroServer and decouples data composition and storage from the data acquisition process.

Have you identified bottlenecks in your solutions? How do you deal with them?

IV-Deadtime

- Flyscan could have same kind of pb
- Our DataMerger has to finish the merge to allow a new scan
  - Currently we don’t have complaint from BL
- If the pb occurs, we can improve by:
  - Adding memory and use ramdisk system
  - Increase the network bandwith …

- A bigger solution would be to allow starting a new scan while the merge process is finishing

Minor issues

Please redistribute the presentation time so that the minor issues receive less attention. For example, approx. 1 min per minor issues.

Report the theoretical position of the record in the middle of the acquisition interval – currently, it is the beginning of the interval

Continuous scan acquires experimental channel data over a time/position interval. Usually the synchronization event starts the acquisition interval. Assigning the acquired data to the synchronization signal leads to the shift of the result by half of the interval. To correctly interpret the data the result of a channel acquisition should be assigned to the middle of the time/position interval. How do you deal with this difficulty?

1-Half position

- The measured motor position is acquired by the Pandabox, which optionnally do the mean (not the middle) between the start and end of the step
- Link to panda doc: https://pandablocks.github.io/PandABlocks-server/master/capture.html

Report latency time from the trigger/gate controller - currently it is reported only from the experimental channel controller to be considered when calculating the synchronization description.

In Sardana experimental channel controllers may report the minimal latency time that may eventually limit. In such case the synchronization pace and the moveables velocities are adjusted to comply with the “slowest” element. In some special cases the synchronization hardware may also require a minimum passive time (time between generating two synchronization signals), currently we do not consider this times. How do you deal with these cases?

2-Latencies

- Some detectors support gating (Xia Xmap) : so less problems
- For others, we empirically increase the trigger period
    - We have to work on this …

Master/slave configuration of trigger/gate elements in measurement group configuration (will require changes in the experiment configuration user interface and the trigger/gate controller interface)

In some complex cases it is necessary to use a chain of synchronization devices where one act on others to generate synchronization events e.g. motion controller generates synchronization triggers based on position and a counter card generates synchronization gates in time. How do you configure/program such relationships between the synchronizers?

3-Master/Slaves

- Pandabox support either time or position trigger modes
- Timebases plugins can be ordered
- Eg:
  - Order 1 = slave counter board
  - Order 2 = Master clock
- We cannot have 2 Master clocks : they should be Hardwarly synchronized

Measurement group configuration feed with default experimental channel configurations when a channel is added to the measurement group

Laboratory equipment do not change synchronization hardware cabling frequently. Experimental channels elements could be pre-configured in software with their corresponding synchronizers, so whenever these are planned to be used in a scan, the necessary configuration is as minimal as possible. This could be also extrapolated to other channels configurations e.g. plotting.

4-Measurement group configuration

- What is the issue ?
- We have configs

Decouple software synchronized experimental channels to allow different acquisition frequencies and avoid waiting for the slowest one – reduce dead time

In Sardana all software synchronized channels are triggered to start acquisition at the same time, and while any of them is still involved in an acquisition a new synchronization event is ignored. This leads to a lost of software synchronized triggers and produces gaps in the scan records which at the end could be masked with an interpolated data. Do you decouple the software synchronized experimental channels that are used in the scan so that the slower ones does not affect the faster ones?

5-Software Synchronized

- Use of the AttributeSampler system:
  - A specific HW: PandaUDPTrigger
  - Get the trigger from the Master Clock
  - And Send a udp frame to an AttributeSampler Tango device
  - This one will then do a Tango read on the defined Tango attribute , eg a temperature
- So we can have equipment whitout trigger in into the Flyscan
- But the Tango reading time has to be less than the Master Clock period !!
- As a quick workaround, we can use the Tango polling system

Add timeout for motion when there is a problem with never ending motion

Sardana implements a protection mechanism for hung acquisition e.g. buggy controller (plugin), buggy hardware or lost synchronization event. Basically, after the motion is finished we wait a given amount of time for the experiment channels to finish (in principle they should finish together with the motion, but we wait some extra time for the extraction of the last data). We also observed that motion may also hang - rather strange e.g. due to the too low velocity (could be easily solved by setting a velocity limit). Did you find this kind of hung motion problems? How do you solve them?

Add early timeout for no data e.g. missing triggers already after the motion started, or eventually add more feedback to the user, for not waiting for the late timeout only after the motion finished

In some cases, e.g. disconnected synchronizers and experimental channels, the acquisition may never start. In this case, there is no need to wait for the motion to finish in order to start counting the acquisition timeout. Such a situation could be discovered automatically by timeout of arriving data, or manually by the user after receiving more feedback on what is currently going on in the experiment.

6-Timeouts

- Scan Timeouts
  - Each plugin has a timeout parameter (motion plugins …)

- Early timeouts
  - We don’t have such solutions.
  - This is a good idea to manage this kind of timeout

Add a configuration interface for the master motor (the one used for synchronization purposes) in scans of pseudomotors.

Currently Sardana continuous scans can run only one synchronization description. In case of the synchronization description in the position domain it is based on the position of one motor. Currently in scans involving a pseudo motors it is not possible to choose which of the underneath motor should be used for the synchronization (master motor). Do you allow to select the master motor by configuration means? Have you found this necessity?

In case in the future we add support for multiple synchronization description would it make sense to support multi master motor synchronization?

7-Master Motor

- Not yet supported
- Could be implemented into the PowerPMAC which then would send pseudomotors simulated encoders signals to the the Pandabox

Validate measurement group configuration before step scan to prevent running them with start synchronization (not compatible)

Start synchronization (just one trigger, and then auto-triggering by the experimental channel hardware controller) is not compatible with the step scan. It should be prevented to start step scans with this synchronization configuration. Sardana does not protect against this kind of “wrong” configuration. Do you make this kind of protection in your solution/framework?

8-Step scans ?

- Didn’t understood…

Thank you