Novel Uses for *xia2* in a Beamline Environment

Harvard Medical School, July 2012

Graeme Winter

Diamond Light Source

July 2012

diamond

Overview

- Context
- At Diamond Light Source
- Options for lower level interaction
- How does this help?

diamond

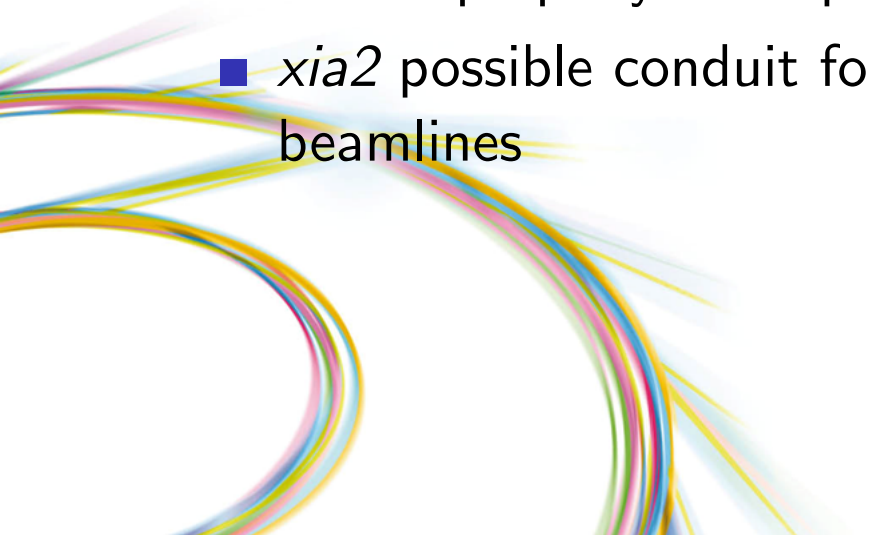
Acknowledgements

- Thanks to Nick Sauter for suggesting this - moving in an interesting direction
- *xia2* funded through BBSRC e-Science e-HTPX project, CCP4, BioXHit and Diamond Light Source
- Users, providers of test data, lots of people who have provided input into project, Diamond Light Source staff

diamond

Context / Background

- *xia2* initially developed as part of e-HTPX project, supported by CCP4 and EU, now Diamond Light Source
- Started work on DNA project in 2002 - kept this in mind
- Original thoughts behind xia2 were to cover scope from data collection through reduction to phasing / refinement
- Implemented middle bit
- *xia2* is properly free open source software
- *xia2* possible conduit for delivering DIALS software to beamlines



diamond

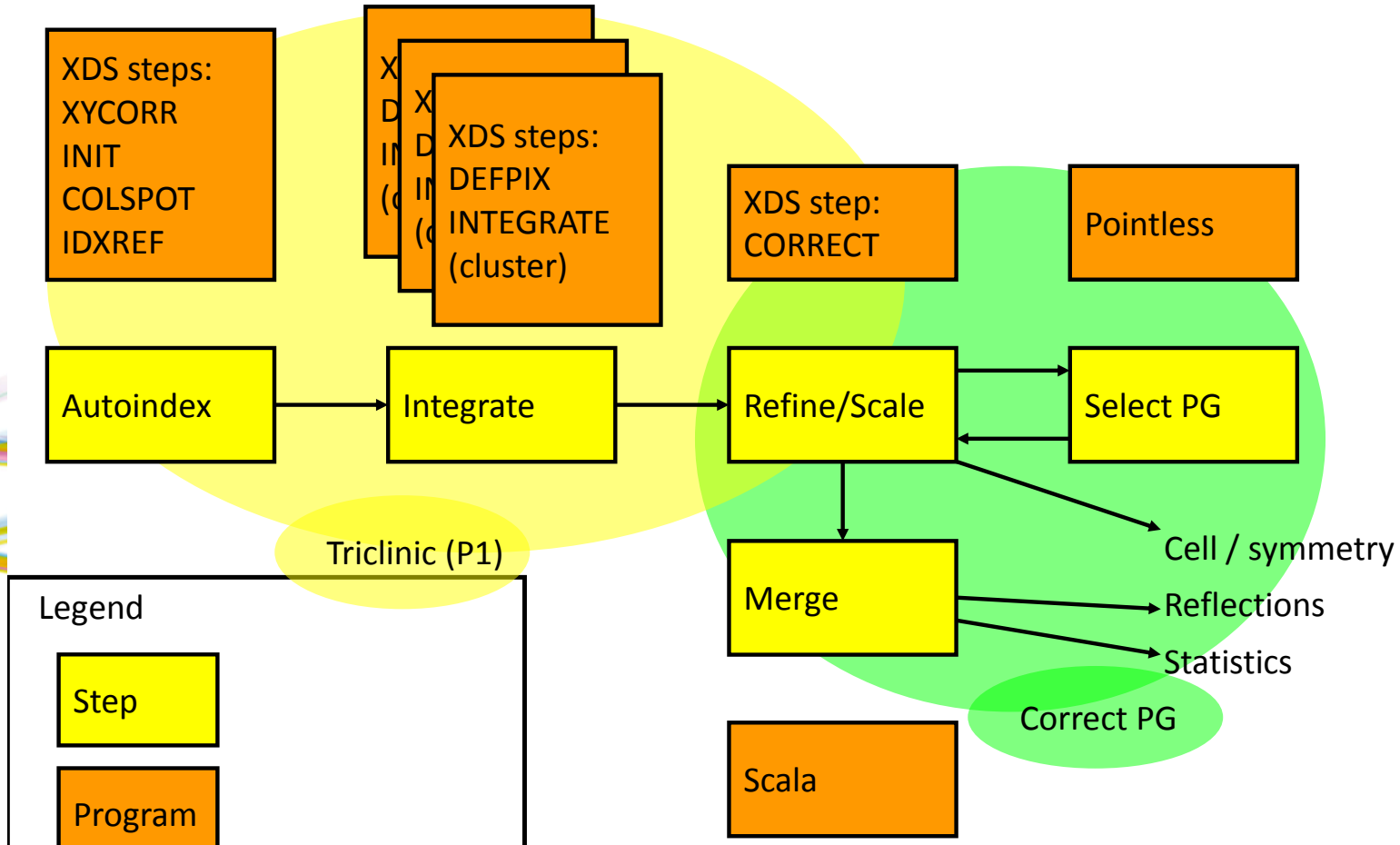
xia2 at Diamond Light Source

- Full functionality available to users manually
- Run automatically (has been so for five years or so) on a per-sweep basis - xia2 -blah -image /path/to/an/image
- Automatic running essentially fire and forget - results available to user, not integrated into data collection system
- Rather than providing fine grained user control, run several jobs
- Inspired development of fast DP - a very much cut down "script" for performing simple data analysis with XDS, in very short time

diamond

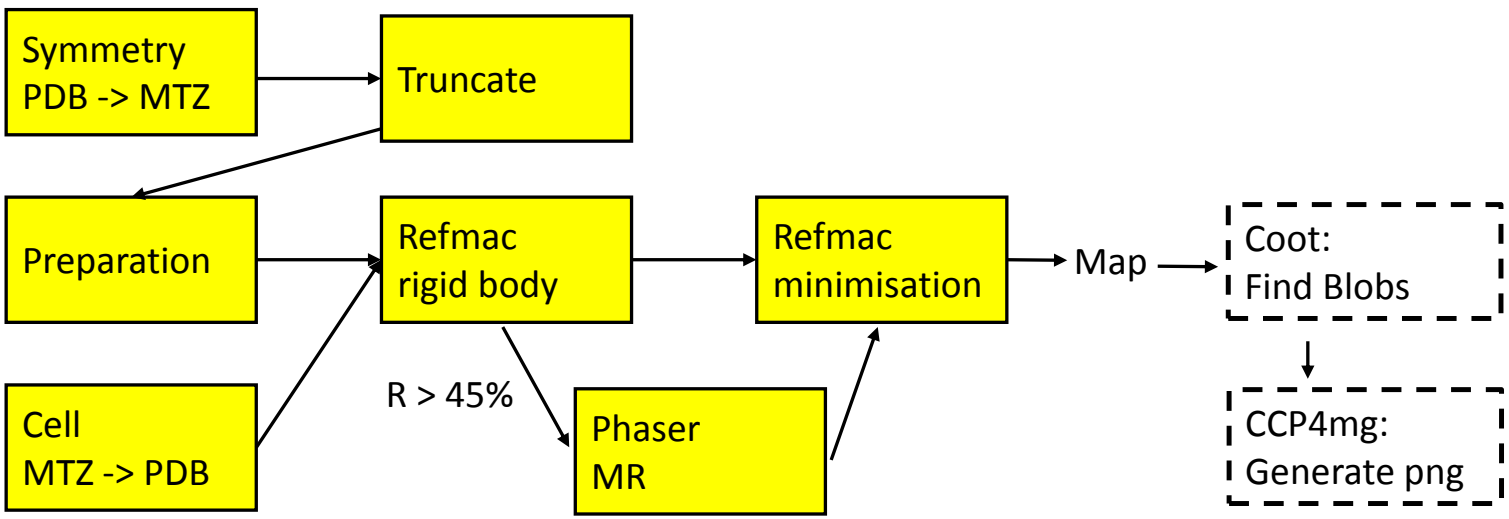
Fast DP

Fast processing with Fast DP - script which runs XDS on up to 480 cpus to deliver data reduction (frames to merged MTZ, merging stats) in under 2 minutes, even for 1800 frame Pilatus sets.



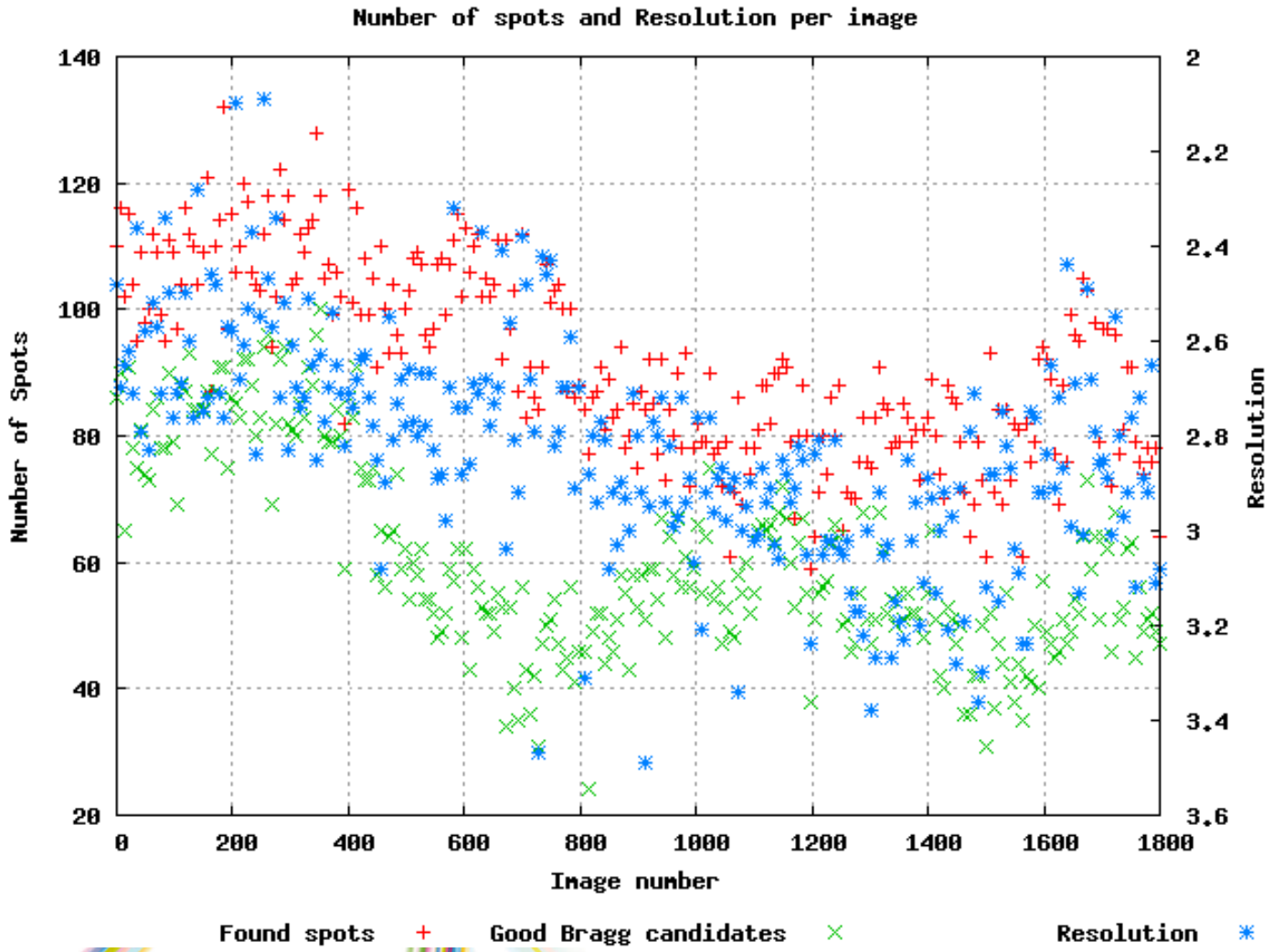
Dimple

Following Fast DP - compute difference map from known native structure, particularly useful for industry. Map available within a few minutes of the experiment.



Per-image analysis

Run DISTL on every image / 250 from wedge, plot - gives near real-time feedback on diffraction. Useful for characterising what happened to sample.



Benefits with this approach

- Lack of user interaction - they can get on with data collection
- Now considered by users a beamline component - they complain on feedback forms when it fails
- All data is processed during the beam time - users can plan experiments properly

diamond

Issues with this approach

- Lack of user interaction
- Timescale, hence Fast DP (next slide)
- Per-sweep *xia2* problem: does not sit nicely with e.g. MAD, multi-sweep data collection
- Feedback to user is not assertive i.e. direct feedback to data collection system
- *Does not handle incremental feedback well*

diamond

Lower-level interaction with *xia2*

- Main *xia2* program is a print statement (really) - all processing performed to deliver the results of this print statement
- However, the code is *just Python*
- Let's look at a lower level interaction

diamond

```
from Handlers.PipelineSelection import add_preference
directory, image = os.path.split(sys.argv[1])

from XProject import XProject
from XCrystal import XCrystal
from XWavelength import XWavelength

xp = XProject(name = 'example')
xc = XCrystal('demonstration', xp)
xw = XWavelength('native', xc)
xw.add_sweep('native', directory, image)

print 'Scaled data: %s' % xc.get_scaled_merged_reflections()['mtz']
```

diamond

Observations

- Code above will work
- It will also write a lot of junk to stdout
- May also generate extra directories - full fat version next slide

diamond

```
from Handlers.PipelineSelection import add_preference
from Handlers.Streams import streams_off
from Handlers.Environment import Environment

Environment.dont_setup()
streams_off()

add_preference('indexer', 'labelit')
add_preference('integrater', 'xdsr')
add_preference('scaler', 'xdsr')

directory, image = os.path.split(sys.argv[1])

from XProject import XProject
from XCrystal import XCrystal
from XWavelength import XWavelength

xp = XProject(name = 'example')
xc = XCrystal('demonstration', xp)
xw = XWavelength('native', xc)
xw.add_sweep('native', directory, image)

print 'Scaled data: %s' % xc.get_scaled_merged_reflections()['mtz']
```

Moving on

- This code will *silently* process your data, scale and return a merged MTZ file containing intensities and amplitudes
- Provided handle to xp or xc kept can interrogate nearly everything
- *Aha!* why can't I just add another sweep and get reflections again? Bugs
- Code was never written to work like this, though with a couple of hours work could be
- Move control of system from command-line input to Phil objects - more easily embedded

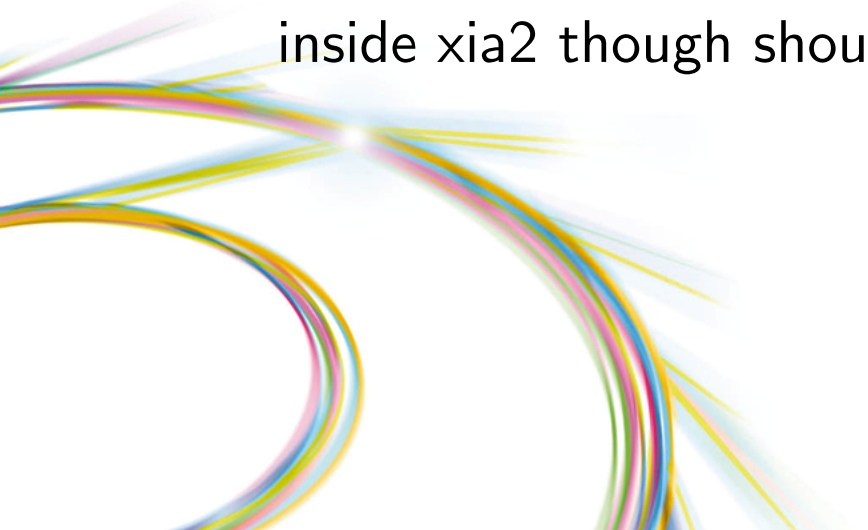
How does this help?

- Lower level access to *xia2* machinery
- More control from caller pespective
- Capability to provide user interface - user can (in principle) add and remove sweeps, tweak processing on live system
- Capability for system to keep adding sweeps (say multi-crystal environment) until complete data set achieved

diamond

What needs doing?

- Mainly resolving dependencies in the analysis - xia2 is dynamic enough already
  - e.g. when sweep added to wavelength, scaling needs repeating
  - e.g. when images added to a sweep integration needs repeating
- Probably a couple of days work - not hours, I tried the other day
- Testing - this is an approach which has never been tested inside xia2 though should work



diamond