MAUS - Feature #1312

Online reconstruction requires new API

15 July 2013 15:18 - Rogers, Chris

Status:	Open	**Start date:**	15 July 2013
Priority:	Normal	**Due date:**	
Assignee:	Rogers, Chris	**% Done:**	0%
Category:	Online reconstruction	**Estimated time:**	0.00 hour
Target version:	Future MAUS release		
Workflow:	New Issue		

Description

See #1310.

We need a new API that enables MAUS reducers to broadcast images in memory without write to disk, which appears to be too slow for convenient running. It would also be beneficial if reducers could operate on another node, e.g. the other onrec machines.

Expected that this will involve broadcasting histograms over a socket and picking them up in a viewer application.

Related issues:

Related to MAUS - Bug #1678: Outputters take data as argument	**Closed**	**07 May 2015**
Related to MAUS - Feature #1717: Offline reconstruction job in MLCR	**Closed**	**21 July 2015**

History

#1 - 21 October 2013 13:04 - Rogers, Chris

Job breaks down as:

1. Revise data structure to make a new event type in pure cpp. Data structure should look like:

```
Image -> RunNumber
      -> SpillNumber
      -> Input time stamp (i.e. best guess at when the data was generated; check )
      -> Output time stamp (i.e. when was the data
      -> List of canvas wrappers, each containing
         -> Description of histogram
            -> data displayed
            -> x axis definition
            -> y axis definition
         -> Boolean defining whether the histogram is good (communication with alarm handler?)
         -> TCanvas containing the histograms (or graphs)
```

   - ~~Concern - if I push a TCanvas across, will it pull child histograms/etc?~~
     - Yes, as long as the histogram is Drawn
   - ~~Concern - what are the data sizes like?~~
     - Looking at the output .root files from existing online recon, we have
     - imageReducePyTOFPlot.root - 22k
     - imageReducePyCkovPlot.root - 13k
     - i.e. highly manageable (these are canvases for an entire detector, so expect that * 20 reducers => of order few MB per canvas refresh)
2. Revise ReducePy* to write to the new data type
   - May be possible to do everything in ReducePyRootHistogram
3. Make new OutputCppCanvas broadcast the data as a TSocket
   - And writes to end of run directory
4. ROOT TApplication to handle and display the incoming messages

#2 - 05 December 2013 06:04 - Rogers, Chris

Propose

- Reducer modifications - 32 hours
- Outputter - 16 hours
- Sockets - 16 hours
- GUI - 40 hours
- Fixing celery - 32 hours
- Networking and load balancing in MLCR - 40 hours
- Documentation - 16 hours

- GUI Test script (or equivalent) - 16 hours
- Load Test - 8 hours

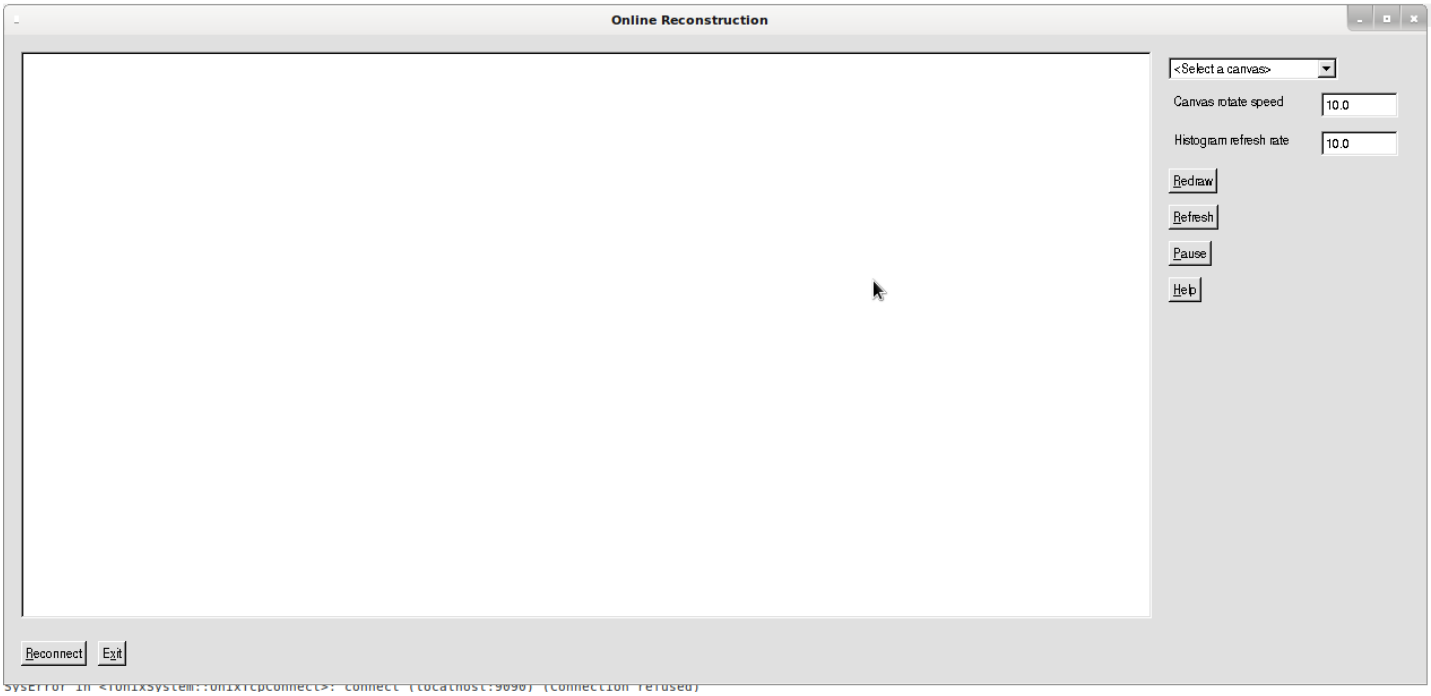
Let's see how it goes...

#3 - 05 December 2013 15:53 - Rogers, Chris

Have a working server/socket pair (in PyRoot) and a first stab at the data structure pushed to lp:~chris-rogers/maus/1312 - **5 hours**

#4 - 09 January 2014 12:27 - Rogers, Chris

- File Online_Reconstruction.png added



Rapid prototype of online recon - plan:

- **<Select a canvas>** - list of canvases to display. Also option to rotate canvases (i.e. loop over all canvases in the list)
- **Canvas rotate speed** - float enabling user to set speed at which we flip from one canvas to the next in seconds. Each time we flip to a new canvas it is automatically reloaded, unless GUI is Paused.
- **Canvas refresh rate** - float enabling user to set speed at which current canvas is reloaded, unless GUI is Paused.
- **Redraw** - force a redraw of the current canvas
- **Refresh** - force a reload of all canvases from the socket, even if GUI is paused.
- **Pause/Unpause** - click on Pause to force the GUI to not reload or rotate current canvas. The button changes to display Unpause; click on Unpause to let the GUI reload or rotate current canvas.
- **Help** - display help on currently displaying canvas.

- **Reconnect** - attempt to reconnect to the socket (in case of network issues)
- **Exit** - close the Online Reconstruction GUI

#5 - 09 January 2014 12:28 - Rogers, Chris

- File deleted (Online_Reconstruction.png)

#6 - 09 January 2014 12:29 - Rogers, Chris

- File Online_Reconstruction.png added

#7 - 09 January 2014 14:44 - Rogers, Chris

By email:

On 09/01/14 13:39, Taylor, Ian wrote:

> Looks good. A couple of comments / questions:
>
> a) Will I be able to have multiple windows connected at the same time? I'm thinking we might want to pause something, wait 5 minutes, and compare it to an updated version. Or just look at multiple plots in parallel.

I can make it a feature request/test condition to be able to run multiple GUIs in parallel. Does that meet the requirement?

> b) If I pause the canvas, and then click redraw, what will it draw? New or old. I mention it because I can see a situation where someone zooms in on the X axis, to investigate a peak, and then clicks redraw to get back to the original, but actually ends up with new data. Probably not a major problem, maybe I'm just requesting a 'reset' button be added, which doesn't refresh the data, just resets the display.

Redraw redraws the currently displayed histogram with the existing axes. Refresh reloads the latest histogram from the socket and draws it.

> c) I can imagine two sets of canvases, shifter level and expert level. The rotate function probably doesn't need to display the latter set. Could we have a button to produce a box, with tick boxes for each canvas, so that only those ticked are included in the rotate.

I will add the requirement.

> z) Can we get a real rotate option, where it spins around really fast? Don't really need it, but after misunderstanding initially, I kind of want it now.

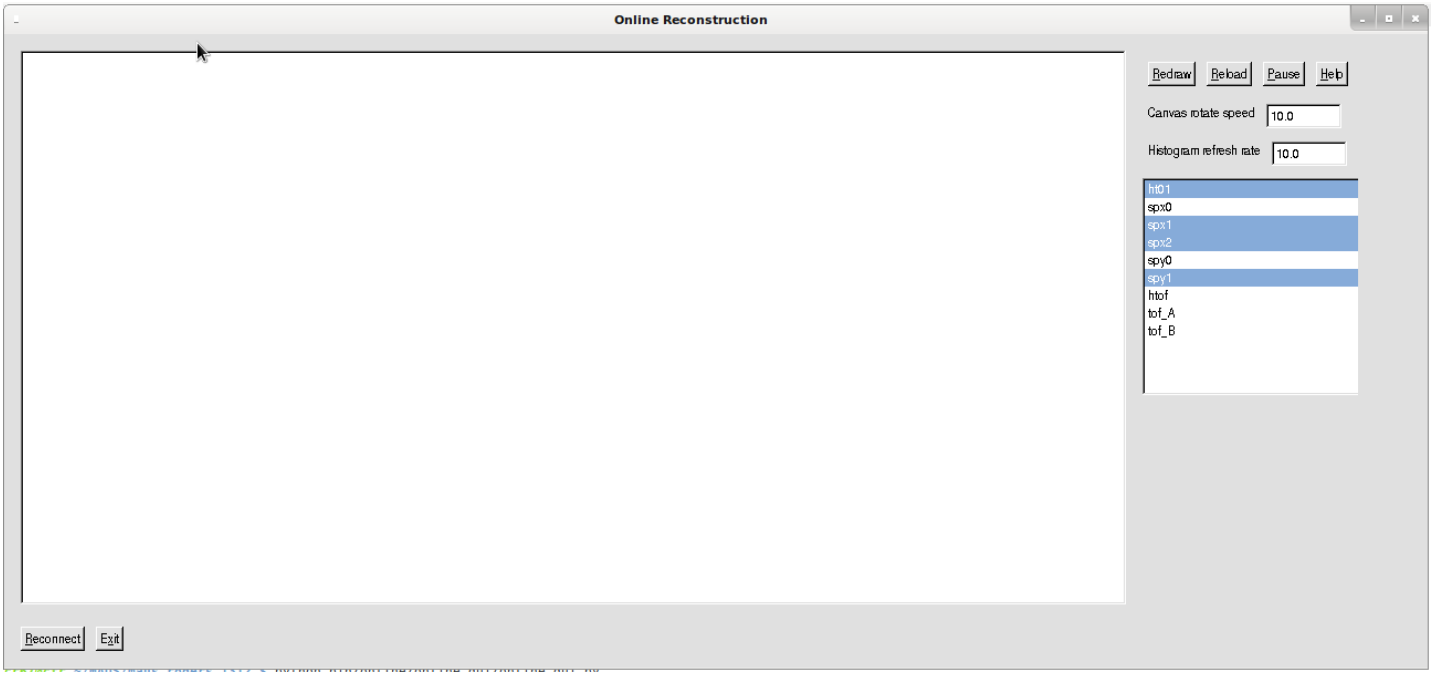
I will ask Craig to put the monitor on a turntable. Or maybe that's not what you meant.

> I've had some other ideas, but rejected them as probably being unimportant or irellevant to MICE. But if I change my mind, I'll let you know.
>
> Ian _____ From: Chris Rogers <chris.rogers@stfc.ac.uk> Sent: 09 January 2014 12:30 To: Taylor, Ian Subject: Online recon GUI
>
> I did a prototype of online recon GUI. Please let me know if you think it is reasonable (before I start implementing logic)...
>
> http://micewww.pp.rl.ac.uk/issues/1312#note-4
>
> Cheers, Chris

#8 - 09 January 2014 15:41 - Rogers, Chris

- File Online_Reconstruction_2.png added

Something like this



#9 - 09 January 2014 15:45 - Taylor, Ian

But can I select to view canvas tof_A, without losing the selection of user level stuff?

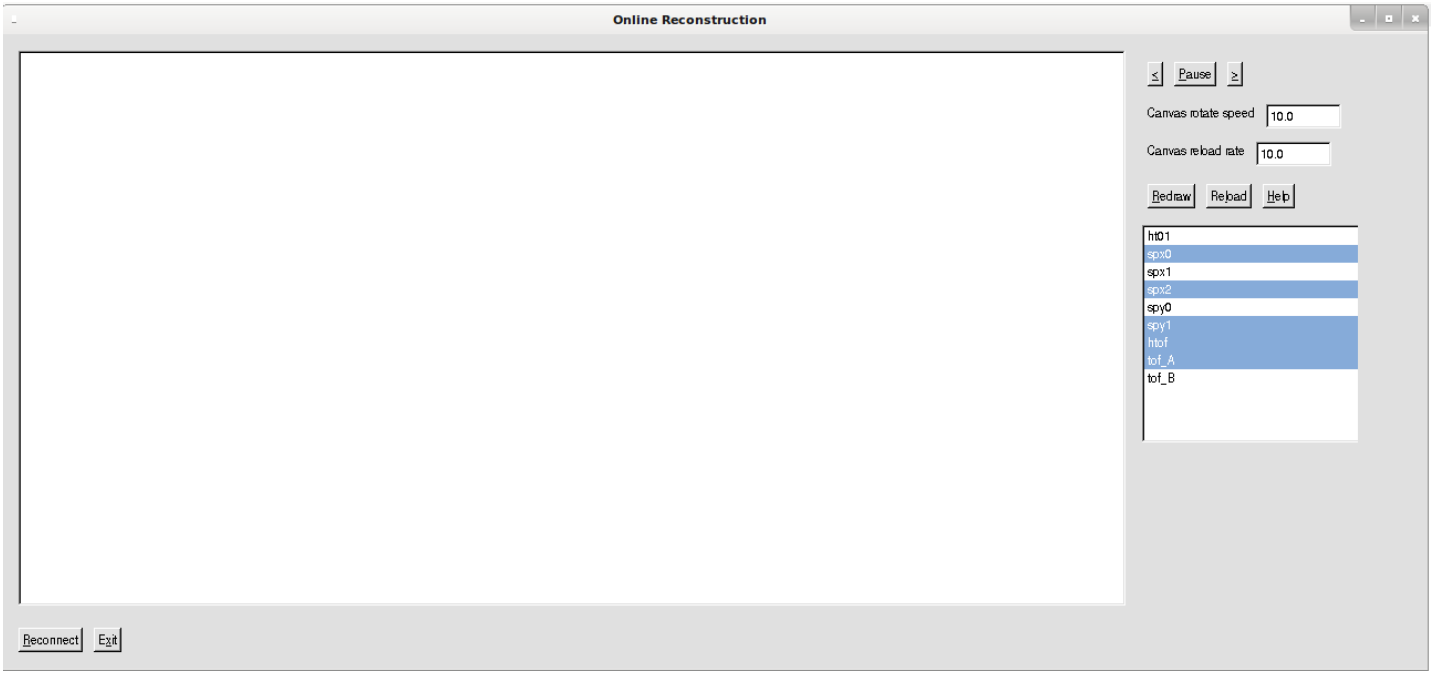
And am I annoying you yet?

#10 - 09 January 2014 16:02 - Rogers, Chris

- File Online_Reconstruction_3.png added

> And am I annoying you yet?

Actually - I'm happy to agree on the spec before I implement the logic so that I only have to do the logic coding once (I guess this might have an impact on the backend, though probably not, but that's why I'm trying to get front end specified in detail first)



- *<* - Go back one through the rotation
- *>* - Go forwards one through the rotation

If you prefer a drop down menu with a "Rotate" option, or many canvases, and the multiselect which determines what is in the rotation then I can do that. It feels a bit like duplicating data which is probably why I resist...

#11 - 09 January 2014 16:10 - Taylor, Ian

No, fair enough, as long as I can have two clients connected, then a 'user' version can exist while the expert is messing with another window. This removes the requirement to be able to select arbitrary canvas A while the loop is set for B, C, E. But that was my main initial point. You could just throw back in the <select a canvas> dropdown list from the initial proposal, and document the multi-select box to state that it only defines which canvases are in the rotation.

I'm happy with any of those solutions.

#12 - 10 January 2014 13:28 - Rogers, Chris

More thoughts on the general architecture. The current process architecture looks like:

```
                  |-> Recon ->|              |-> ReduceTOF  ->|              |-> Browser
                  |           |              |                |              |
Input -> DocStore |-> Recon ->|-> DocStore ->|-> ReduceCkov ->|-> MAUS-App ->|-> Browser
                  |           |              |                |              |
                  |-> Recon ->|              |-> Reduce...  ->|              |-> Browser

 DAQ      MongoDB    Celery       MongoDB        Subprocess     Django/Celery   Firefox
```

The requirement from Ian that we want to support multiple ROOT GUIs is fine, so now the process architecture looks similar except that we have removed MAUS-App:

```
                  |-> Recon ->|              |-> ReduceTOF  ->|                 |-> ROOT GUI
                  |           |              |                |                 |
Input -> DocStore |-> Recon ->|-> DocStore ->|-> ReduceCkov ->|-> ?Something? ->|-> ROOT GUI
                  |           |              |                |                 |
                  |-> Recon ->|              |-> Reduce...  ->|                 |-> ROOT GUI

 DAQ      MongoDB    Celery       MongoDB       Subprocess            ?            ROOT
```

Note the ?Something? on the right hand side - this cannot elegantly be MongoDB as MongoDB does not easily store binaries (e.g. ROOT TCanvas). My experience with MongoDB is also that it is optimised for many small transactions, whereas our interest is few large transactions. I am looking into the various ROOT parallelisation tools... it would be nice to be able to remove Celery at the same time which is giving me so much trouble elsewhere. So some sort of document store and distributed queue... note the doc store doesn't have to be distributed in the first instance as file sizes are relatively small. I hope docstore can be not distributed. Needs to be able to communicate with remote nodes however (where remote is in the same rack room).

#13 - 10 January 2014 14:01 - Rogers, Chris

The thing I would expect to use is rootd/TNetFile and wrap around this for remote operations. Using TNetFile means that we get for free:

- A Daemon utility to handle asynchronous connection requests, which contains:
  - Wrapper around all of the ROOT socket gubbins
  - A document queue/store

It looks very elegant. Concern is that the only way to update a TNetFile on the server may be to make a disk write operation, which may be slow. One major aim of the rewrite is to get around slowness associated with disk writes. Much better to hold everything in memory with rootd.

By this time I have figured out and have working prototypes for the ROOT socket gubbins, so I lean towards reimplementing this ourselves with the advantage that we can optimise for our use case (so we don't end up in the same tangle as we had for e.g. mongodb). Setting up a document queue should be relatively straight forward.

#14 - 13 January 2014 10:32 - Taylor, Ian

Wrt the disk write operation, recent versions of redhat OS have tmpfs as a core component. This presents system memory as disk space, such that a 'disk write' operation can occur without ever actually leaving RAM. As long as we pay attention to usage, and don't allow files to build up, then it should have minimal effect on the processing. And it is very simple to implement.

Otherwise, TNetFile looks like it should work just fine.

#15 - 14 January 2014 10:58 - Rogers, Chris

Having experienced necessity for a partial rewrite already, I would rather take more control than less - so prefer to make the docstore myself. This tmpfs concept, while nice, is also hacky. I would rather do it in a way in which I am happier with. So I will implement my own document queue.

ps: mucking around with GUI prototype and sockets - **12 hours**

#16 - 20 January 2014 16:01 - Rogers, Chris

Now have a reasonable protocol for messaging across sockets, passing data around and storing it. I probably still have some debugging to do and more work required on the server side data store. Pushed to bzr+ssh://bazaar.launchpad.net/~chris-rogers/maus/1312/ r1020. Looks like I may be able to set up something reasonable to replace the MongoDB/celery side as well, but I will try to get the path from Reducer -> ROOT GUI working and stable before fiddling with any of that stuff.

Currently I plan to use iptables (firewall) to handle security and do not plan to implement any authentication routines. This may change later on.

24 hours

#17 - 28 January 2014 09:34 - Rogers, Chris

As of rogers - r1024

- Finished RootDocumentStore - lots of time was still spent mucking about with the sockets to get the data transfer more stable. Feel like it still isn't quite stable - tests show ~ 1% of messages get dropped just on localhost. But it will do.
  - Nb this exposes the same interface as the MongoDBDocStore but can handle natively Root binaries as well as json.
- Have proto OutputCppSocket but no tests. Need Taylor API to handle data transfer between reducers and outputter
- Working on the GUI now. So push mock image to RootDocumentStore, then try pushing into the GUI and see how it goes.

25 hours

#18 - 30 January 2014 16:24 - Rogers, Chris

I have now a working, tested GUI. I haven't run pylint yet. **16 hours**

Total time spent so far - 82 hours

Status:- compared to original estimates

Task	Time est	Time taken	% done
Reducer modifications	32 hours	0 hours	0 %
Outputter	16 hours	8 hours	50 %
Sockets	16 hours	54 hours	90 %
GUI	40 hours	20 hours	90 %
Fixing celery	32 hours	0 hours	0 %
Networking and load balancing in MLCR	40 hours	0 hours	0 %
Documentation	16 hours	0 hours	0 %
GUI Test script (or equivalent)	16 hours	0 hours	0 %
Load Test	8 hours	0 hours	0 %

The sockets took longer than I anticipated (but I knew that it would be a fiddly thing). However, I think now I have the sockets and a good messaging protocol, the rest should go quicker. In particular, this takes the risk off of the "Fixing celery" task - I have the inter-process messaging sorted, setting up a worker pool et al should be relatively easy. Still a slight concern that my messaging efficiency is only around 99 %, even on localhost, depending on congestion. I hope it doesn't dip much lower...

I should say the Reducer/Outputter modifications are blocked by changes required to the API to handle native passing of binary data between the Reducer and the Outputter. Once I have these done, I can proceed to deploy the new GUI to the MLCR - with the networking, celery and load testing to follow as an upgrade.

#19 - 31 January 2014 09:20 - Rogers, Chris

I realise there are another couple of tasks to add:

- Superimpose reference data set
  - Using existing regression testing code
- Alarm to C+M
  - Determine alarm limits
  - Implement alarm limits
  - Communication with EPICS

Alarm to C+M	32 hours	0 hours	0 %
Superimpose reference data set	16 hours	0 hours	0 %

May open as a new issue

#20 - 12 June 2015 17:15 - Rogers, Chris

So I finished the Reducer API modifications.

- I made ReducerBase<template T_IN, template T_OUT> that processes an object of type T_IN and spits out an object of type T_OUT

- I added an ImageData type that contains a vector of CanvasWrappers. The full data structure looks like

```
ImageData -> Image -> vector of CanvasWrappers -> ROOT::TCanvas
```

- I made an outputter called OutputPyRootImage that does the same job as the old OutputPyImage but accepts an ImageData type rather than a json type. Note that unlike other data types, ImageData is exclusively PyRoot and c++ ROOT - there is no string/json implementation. Such a thing is possible but hard, I would have to save the ROOT canvas to disk, convert to binary, load in memory as is currently done for the png files by OutputPyImage - it just sounds hard.
- Still need to go through the existing Reducers and adapt them to the new API. I guess it is a day's work or so.

So:

Reducer modifications	32 hours	16 hours	70 %

Pushed to bzr+ssh://bazaar.launchpad.net/~chris-rogers/maus/1312/ (unit tests pass locally except python style test is failing)

#21 - 23 June 2015 21:04 - Rogers, Chris

- Add: cron job to automatically restart the recon
- Some management for logging of crashes/errors, and/or rotating log files

#22 - 07 July 2015 17:21 - Rogers, Chris

Just having a discussion with Durga about how we close this issue. At the moment the overhead from using celery and mongodb is greater than the time saved by multiprocessing(!) due to the conversion to strings. So I will migrate to a model like

```
InputCppDAQData -> Transform -> ReducePyDoNothing -> OutputPySocket ==> RootDocumentDB ==> InputPySocket -> Ma
pPyDoNothing -> ReducePyBlah -> OutputPyImage ==> cache on disk ==> firefox
```

then

```
InputCppDAQData -> Transform -> ReducePyDoNothing -> OutputPySocket ==> RootDocumentDB ==> InputPySocket -> Ma
pPyDoNothing -> ReducePyBlah -> OutputPyRootImage ==> RootDocumentDB ==> OnlineGui
```

where ==> means we go into another process. Note that this has single threaded reconstruction... ways to then bring the multiprocessing back in.

#23 - 08 July 2015 11:31 - Rogers, Chris

Pushed to the test server. What is missing?

- Reducer modifications
- InputPySocket
- OutputPySocket

Durga said he would do the reducer modifications, let's see how we are doing on, say, Monday and then review. I would like to get this into the MLCR on Monday (so that we can run in a calm way on Tuesday if there is data taking Tuesday night), which means getting this into the trunk over the weekend so that all the tests can run and any crud can get tidied up. It is quite a big change. I would like to maintain the roll-back capability in the trunk.

#24 - 14 July 2015 16:00 - Rogers, Chris

I added InputPyDocStore and OutputPyDocStore to bzr+ssh://bazaar.launchpad.net/~chris-rogers/maus/1312a/

#25 - 23 July 2015 00:26 - Rogers, Chris

A few comments are in #1717

I modified InputPyDocStore and OutputPyDocStore to allow only one port. Previously I tried to implement a port list with failover to next port in the list if the current port did not connect, but that wasn't working. I added to analyze_data_online.py the doc_db management and port management, and removed the celeryd process. So this implements the data flow

```
InputCppDAQData -> Transform -> ReducePyDoNothing -> OutputPySocket ==> RootDocumentDB ==> InputPySocket -> Ma
pPyDoNothing -> ReducePyBlah -> OutputPyImage ==> cache on disk ==> firefox
```

Note some question mark over whether this dataflow reconastructs quickly enough. I think this change implements the major things required for other data flows (i.e. working RootDocumentDB).

I note that I did see once or twice the connection error

```
SysError in <TUnixSystem::AcceptConnection>: accept (Bad file descriptor)
```

I am slightly worried, I have had people complaining about unit tests falling over also. I can try to dig harder, I have already dug quite hard. It comes from some combination of ROOT and Unix sockets. Somehow ROOT gets into a mode where it refuses to accept new connections. The error is generated by line SysError("AcceptConnection" ... in file

https://root.cern.ch/root/html402/src/TUnixSystem.cxx.html

in function

```
int TUnixSystem::AcceptConnection(int sock)
{
   // Accept a connection. In case of an error return -1. In case
   // non-blocking I/O is enabled and no connections are available
   // return -2.

   int soc = -1;

   while ((soc = ::accept(sock, 0, 0)) == -1 && GetErrno() == EINTR)
      ResetErrno();

   if (soc == -1) {
      if (GetErrno() == EWOULDBLOCK)
         return -2;
      else {
         SysError("AcceptConnection", "accept");
         return -1;
      }
   }

   return soc;
}
```

I have not been able to figure out how to clear the error. It is probably caused by attempt to accept a connection from a socket which has already died. Somehow trying to connect on a different socket does not seem to help. ROOT darkness? Or Rogers coding?

#26 - 24 July 2015 10:02 - Rogers, Chris

We ran smoothly last night, we had one crash after about 5 hours of running. Unfortunately I screwed up so the log files were not saved (the log files for onrec03 production recon were saved, but these were useless).

https://micewww.pp.rl.ac.uk/elog/MICE+Log/3361

I saw a

```
SysError in <TUnixSystem::UnixRecv>: recv (Connection reset by peer)
```

. So some network error? I will set up a load test...

Files

Online_Reconstruction.png	14 KB	09 January 2014	Rogers, Chris
Online_Reconstruction_2.png	13 KB	09 January 2014	Rogers, Chris
Online_Reconstruction_3.png	10.4 KB	09 January 2014	Rogers, Chris