New Developments In Analysis

I. Hrivnacova, IPN Orsay (CNRS/IN2P3)

21st Geant4 Collaboration Meeting,
15 September 2016, Ferrara

Outline

- New features
  - Batch plotting
  - Handling parameters
  - Merging ntuples
  - Other developments & Plans

Batch Graphics

- Introduced in 10.2
- Histograms and profiles plotting can be activated using G4AnalysisManager functions:

```
// Activate plotting of 1D histogram
analysisManager->SetH1Plotting(id, true);
   // etc for H2, H3, P1, P2
```

- Or via UI command (still to be implemented)

```
/analysis/h1/setPlotting id true|false
/analysis/h1/setPlottingToAll  true|false
   ## etc for h2, h3, p1, p2
```

Plotting Style

- Can be set via UI command:

```
/analysis/plot/setStyle  styleName
```

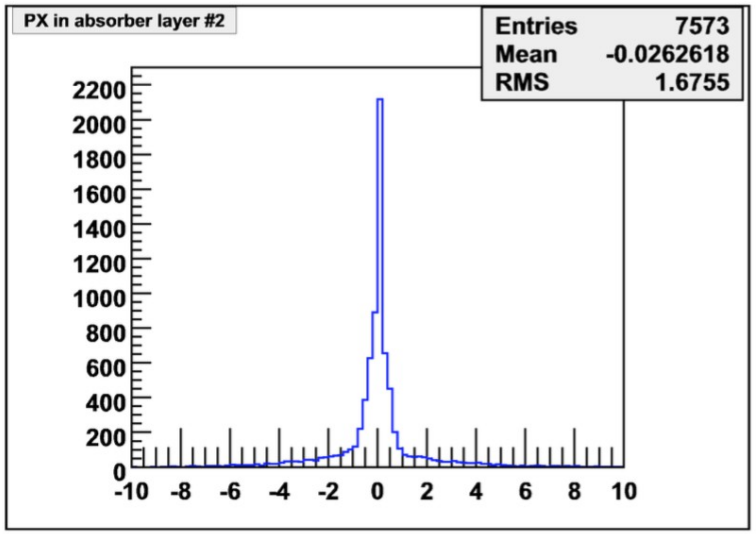
- If Geant4 libraries are built with support for Freetype font rendering, three plotting styles are available:
  - ROOT_default: ROOT style with high resolution fonts (default)
  - hippodraw: hippodraw style with high resolution fonts
  - inlib_default: PAW style with low resolution fonts")
- otherwise only the inlib_default style can be used

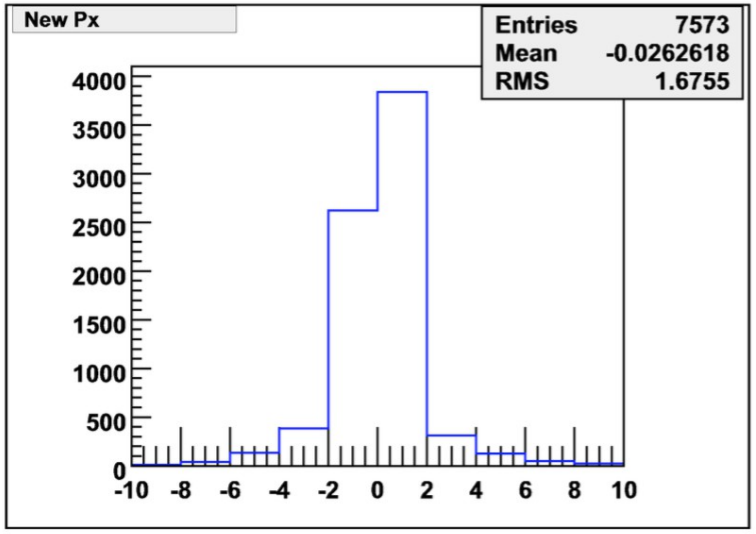
Plot Layout & Dimensions

- Can be set via UI commands

```
/analysis/plot/setLayout  columns rows
/analysis/plot/setDimensions  width height
```

- The page size of the graphics output is fixed to A4 format.
- The plot layout is defined by the number columns and the number of rows in a page.
  - [1, 2, 3] x [1, 2, 3, 4, 5] - for the styles with high resolution fonts
  - [1, 2] x [1, 2, 3] - for the style with low resolution fonts (inlib_default)
- Plot dimensions represent the plotter window size (width and height) in pixels.
- Opening more configuration parameters for users customisation can be considered in future according to the users feedback.





- The test plots from test03 with the default style and layout

Handling Parameters

- The classes for users parameters management were added in 10.2 release for the purpose of simplification of users application code.
  - In analysis category but independent from other analysis classes
- First implementation following the requirements from Luciano Pandola (Advanced Examples WG) and Michel Maire (responsible of B1 and B3 examples)
- Their usage is demonstrated in the basic examples B1 and B3a.

- Further integration in the Geant4 framework in Geant4 10.3 version is under discussion

B1 example

Geant4 10.1.

```
class B1Run : public G4Run {
  public:
    ...
    // method from the base class
    virtual void Merge(const G4Run*);
    void AddEdep (G4double edep);
    // ...
  private:
    G4double  fEdep;
    G4double  fEdep2;
};
```

Geant4 10.2, 10.3.

```
#include "G4Parameter.hh"
...
class B1RunAction : public G4UserRunAction {
  public:
    ...
    // method from the base class
    void AddEdep (G4double edep);
    // ...
  private:
    G4Parameter<G4double>  fEdep;
    G4Parameter<G4double>  fEdep2;
};
```

Run class and Merge() method are not needed

B1 example (cont.)

Geant4 10.2.

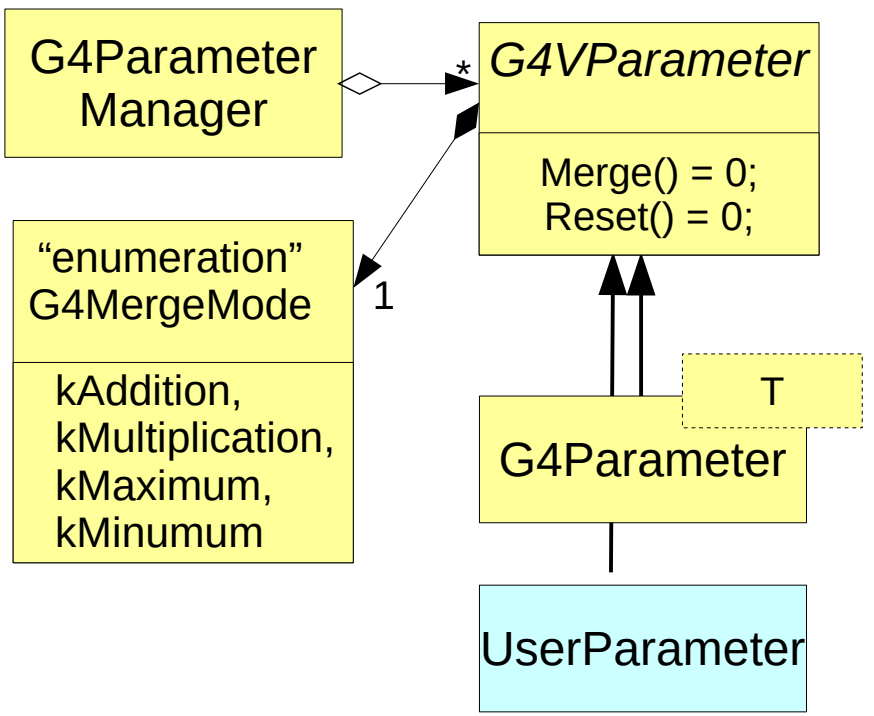
```
#include “G4ParameterManager.hh”
...
B1RunAction::B1RunAction()
: G4UserRunAction(),
  fEdep("Edep", 0.),
  fEdep2("Edep2", 0.)
{
  //Register parameter to the parameter manager
  G4ParameterManager* parManager = G4ParameterManager::Instance();
  parManager->RegisterParameter(fEdep);
  parManager->RegisterParameter(fEdep2);
}

void B1RunAction::EndOfRunAction(const G4Run* run) {
  ...
  // Merge parameters
  G4ParameterManager* parManager = G4ParameterManager::Instance();
  parameterManager->Merge();
  ...
}
```

The parameters are initialized with a name and a value

The parameters not created via the manager have to be registered to it

Handling Parameters
Classes

- G4ParameterManager is a (thread local) singleton
  - Independent from other analysis managers
  - Has std::vector<G4VParameter*> and std::map<G4String, G4VParameter*>
- Provide functions both to create and to register a parameter
- Performs Merge() of all parameters
  - The merge mode can be selected per parameter
- Users can define their own parameters
  - Tested with std::map<G4String, G4int> used for processes counting in TestEm* examples



Handling Parameters
Improvements in 10.3

- The parameters names are optional
- The Merge mode is extended with kMaximum, kMinimum
- Added G4ParameterManager functions to access parameters via iterators or ids (the ids are defined in the order of registering)
- Merging is applied via a function defined according to the merge mode
  - Thanks to Jonathan Madsen for providing an example of the code
- New test08
  - It uses the scenario as defined in basic/B1 example with added use-cases for testing the parameters framework.

B1 example (cont.)

Geant4 10.3.

```
#include “G4ParameterManager.hh”
...
B1RunAction::B1RunAction()
: G4UserRunAction(),
  fEdep(0.),
  fEdep2(0.)
{
  //Register parameter to the parameter manager
  G4ParameterManager* parManager = G4ParameterManager::Instance();
  parManager->RegisterParameter(fEdep);
  parManager->RegisterParameter(fEdep2);
}


void B1RunAction::EndOfRunAction(const G4Run* run) {
  ...
}
```

The parameters name may be omitted.

The parameters not created via the manager have to be registered to it

The call to Merge() may be not necessary if called by kernel

Requirement for Merging Ntuples

- From A. Dotti
- The current strategy not to merge ntuples is the correct one for small/ moderate number of workers (being both threads or MPI ranks).

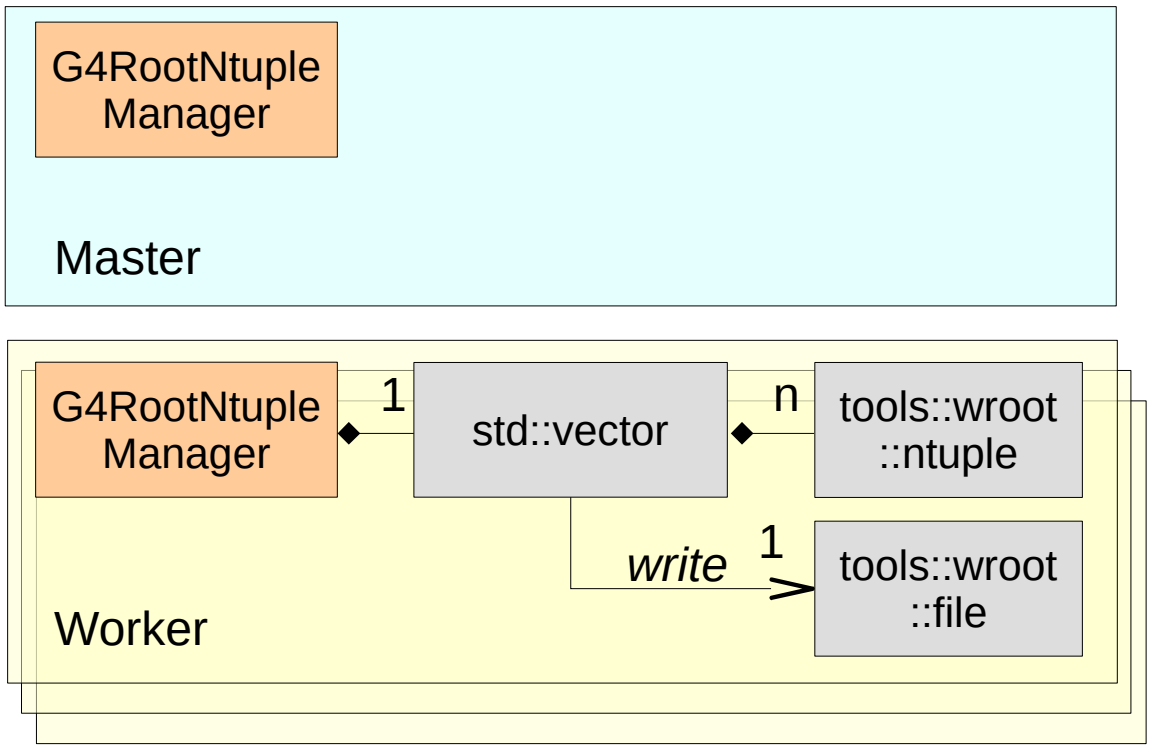
- Both MPI and MT are being used more and more often (see Hypernews):
  - In 2016 I will useTachyon2 supercomputer for physics validationO(100) workers per job
  - It is now an “easy” reality to run with O(1000) workers

- Number of output ntuple files will become an issue

Merging Ntuples

- The optimal number of output files depends on the application:
  - One file per thread – inconvenient for users with small applications, problematic for large application running on supercomputers when running MT + MPI
    - The only possibility in 10.2.
  - One file per application may be more suitable for small applications
    - Working with g4tools 2.0.0-beta, not yet committed in SVN
  - Flexible solution: user can choose the number of ntuple output files
    - In development for 10.3
- New function in G4AnalysisManager:
  - SetNofNtupleFiles(G4int nofNtupleFiles)
  - Applicable in MT (MPI) mode
- This feature is planned to be available in 10.3 only for ROOT output type

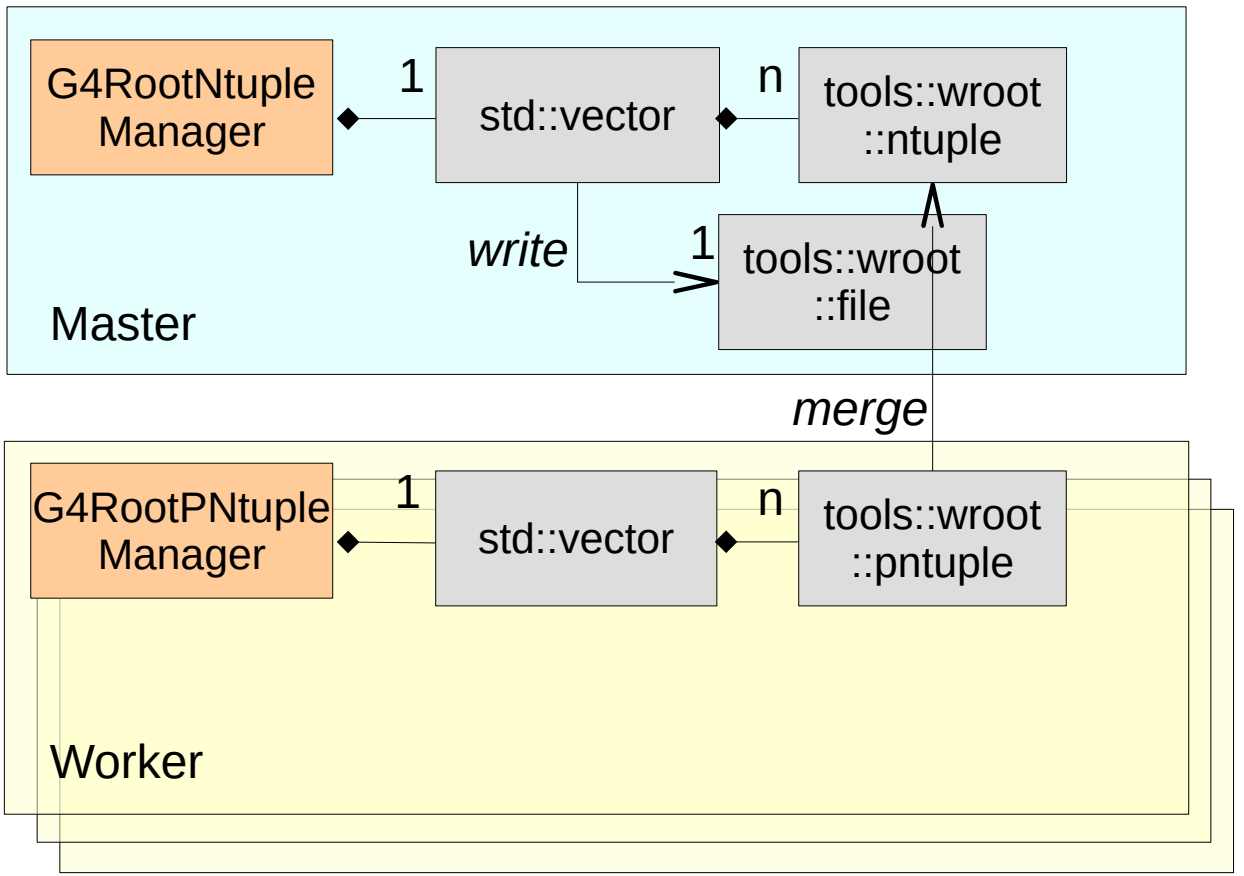
Design 10.2
No Merging



- The **ntuple** objects are created and handled by G4RootNtupleManager on each **worker**
- One ntuple file is created by each worker
- The number of ntuple output files = the number of threads

Design 10.3

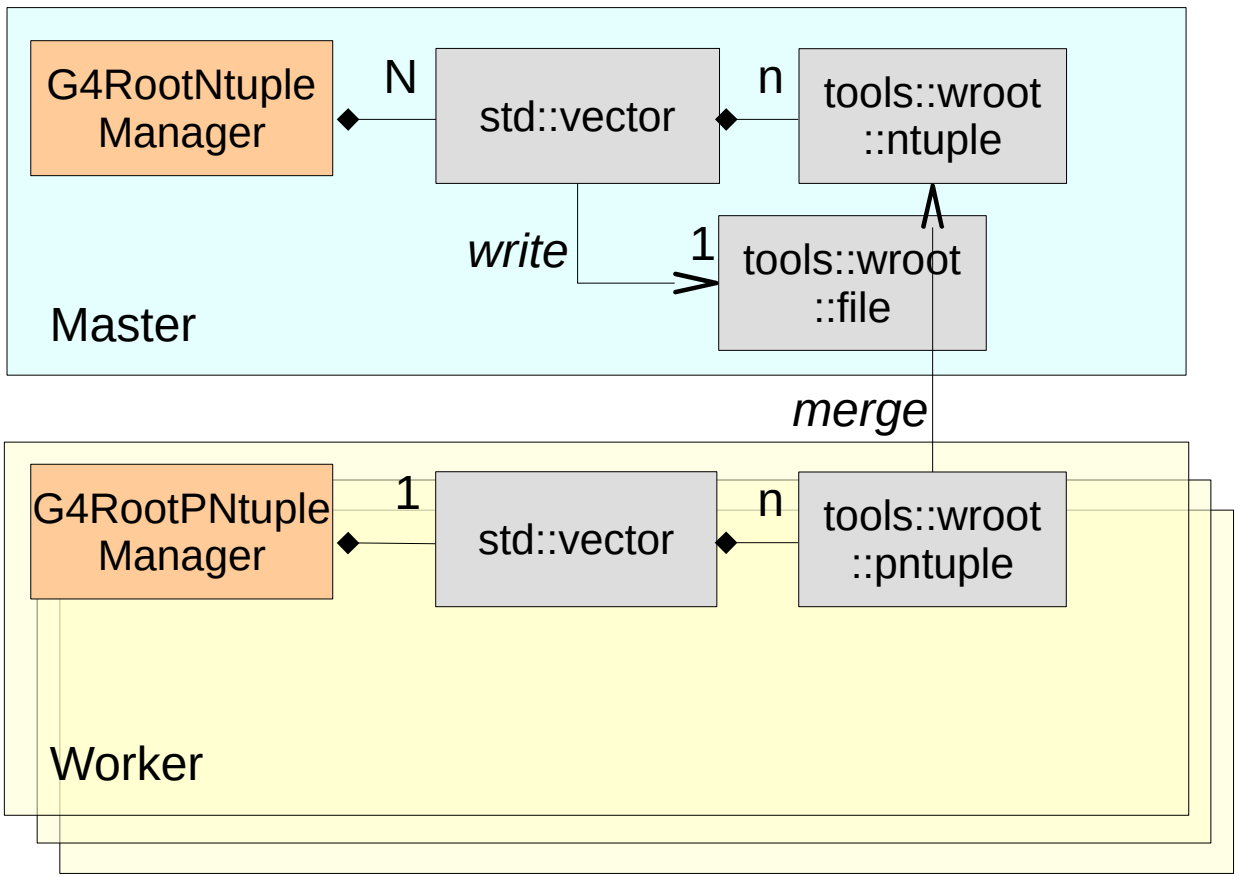
Merging Ntuples in One File



- The master **ntuple** objects are created and handled by G4RootNtupleManager on **master**
- The **pntuple** objects (*of new type)* are created and handled by G4RootPNtupleManager (new class) on **workers**
  - Associated with an ntuple on master
- Merging to the master ntuple happens on AddNtupleRow() call if the pntuple buffer is full and on Write() call
  - Requires locking

Design 10.3

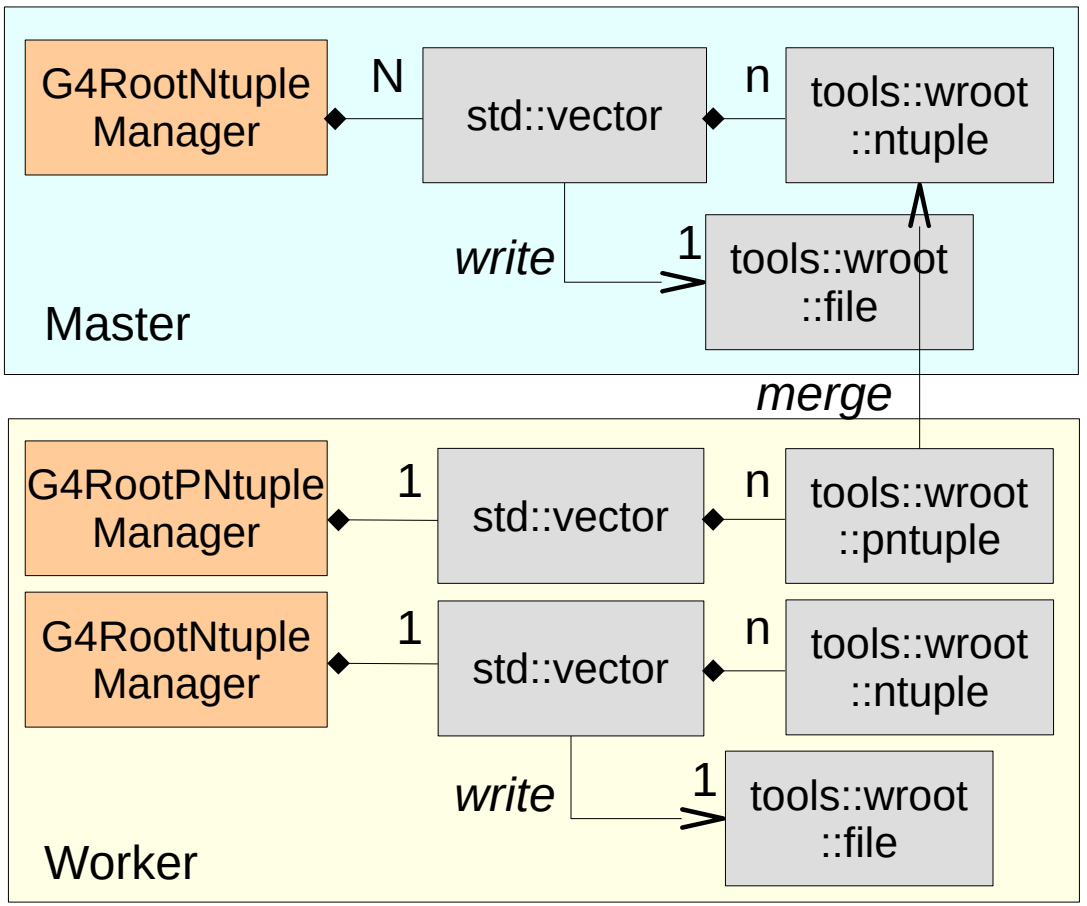
Merging Ntuples in N Files



- N vectors of cloned master **ntuple** objects are created and handled by G4RootNtupleManager on **master**
  - Each vector on master is associated with one file
- The G4RootNtupleManager class has to be extended to handle N vectors of ntuples

Design 10.3

Optional Merging Ntuples in N Files



- The new design combined with the old one
- The file(s) are created either by master (if merging is enabled)
  - The number of files can be selected by the user'
- Or by workers (if merging is disabled)
  - The number of files = the number of threads

Other Developments & Plans

- An example of usage of ntuple columns of vector type was provided in basic example B5 (in 10.2)
- Stop support for HBOOK in 10.3
  - The development and support for CERNLIB is stopped at CERN
  - The binaries are not provided for new platforms
- Features requested by users
  - Handling more files by analysis manager – still to be considered
- Continue addressing new requests from users