Introduction to Fun4All

Fun4All + Simulations – the Framework
The Node Tree – our global data objects
Analysis Modules – Base Class, Data Object access
Building the show – auto tools
Running the show – Condor
Resources – cpu+disk

What is assumed

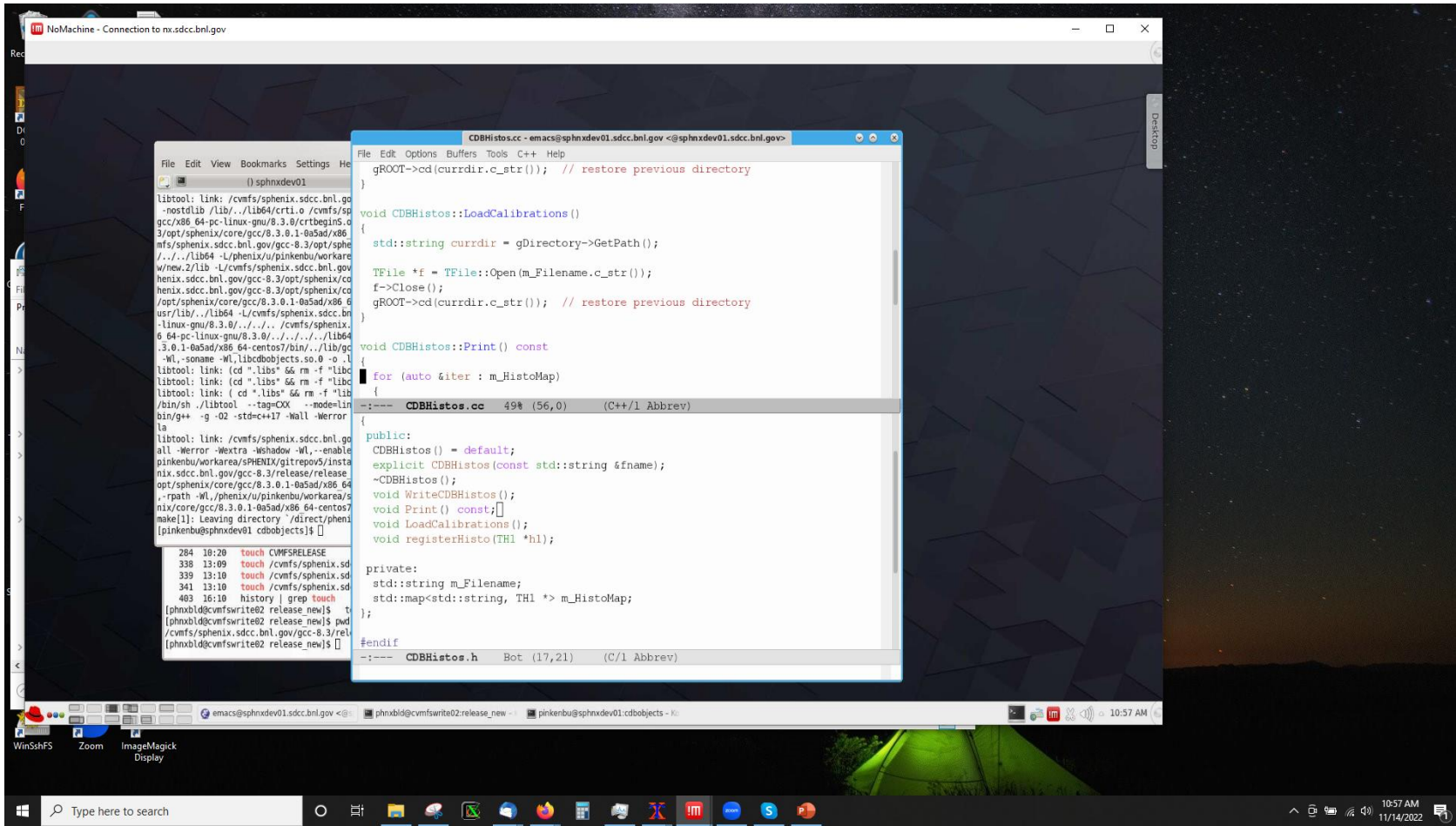
- Basic C++ knowledge (aka what is a base class and virtual methods)
- Basic familiarity with ROOT6 (what is a root macro, drawing histograms from ntuples)
- That you have looked at
  https://wiki.sphenix.bnl.gov/index.php/SPHENIX_software_day-1_checklist
  And for the workfest:
  https://wiki.sphenix.bnl.gov/index.php/SoftwareDevelopmentChecklist
- Can run our software, either via rcf account or
  https://github.com/sPHENIX-Collaboration/Singularity
  Where the container support is on a best effort basis

Highly recommended: NX

- I cannot overstate how much easier this will make your life in sPHENIX
- NX is remote desktop with X11 acceleration

→ https://www.sdcc.bnl.gov/information/services/how-use-nx-sdcc

- Allows connecting from any laptop/desktop (take your session with you
- Survives network outages (no more ssh timeouts)
- Graphics “just works”



Brief History of Fun4All

- Development started in 2002, in use by PHENIX from 2003 on (reconstruction and analysis of Run3 data)
- Underlying data structure – the phool node tree – was part of day1 of PHENIX software (PHENIX Object Oriented Language)
- Needed to get many subsystems who developed their code independently and without coordination under one umbrella
- Development driven by reconstruction and analysis needs – not by “beauty” or some abstract design considerations (or “rules”)
- Modularity is key – components can be added/modified/removed without having to modify bits and pieces all over the place
- Plenty of functionality added over the years, some of them waiting to be rediscovered
- 2011 Adding Geant4 as subsystem
- Split from PHENIX in 2015, lots of cleanup and modernization
  - Also involved in EIC Yellow Report simulations

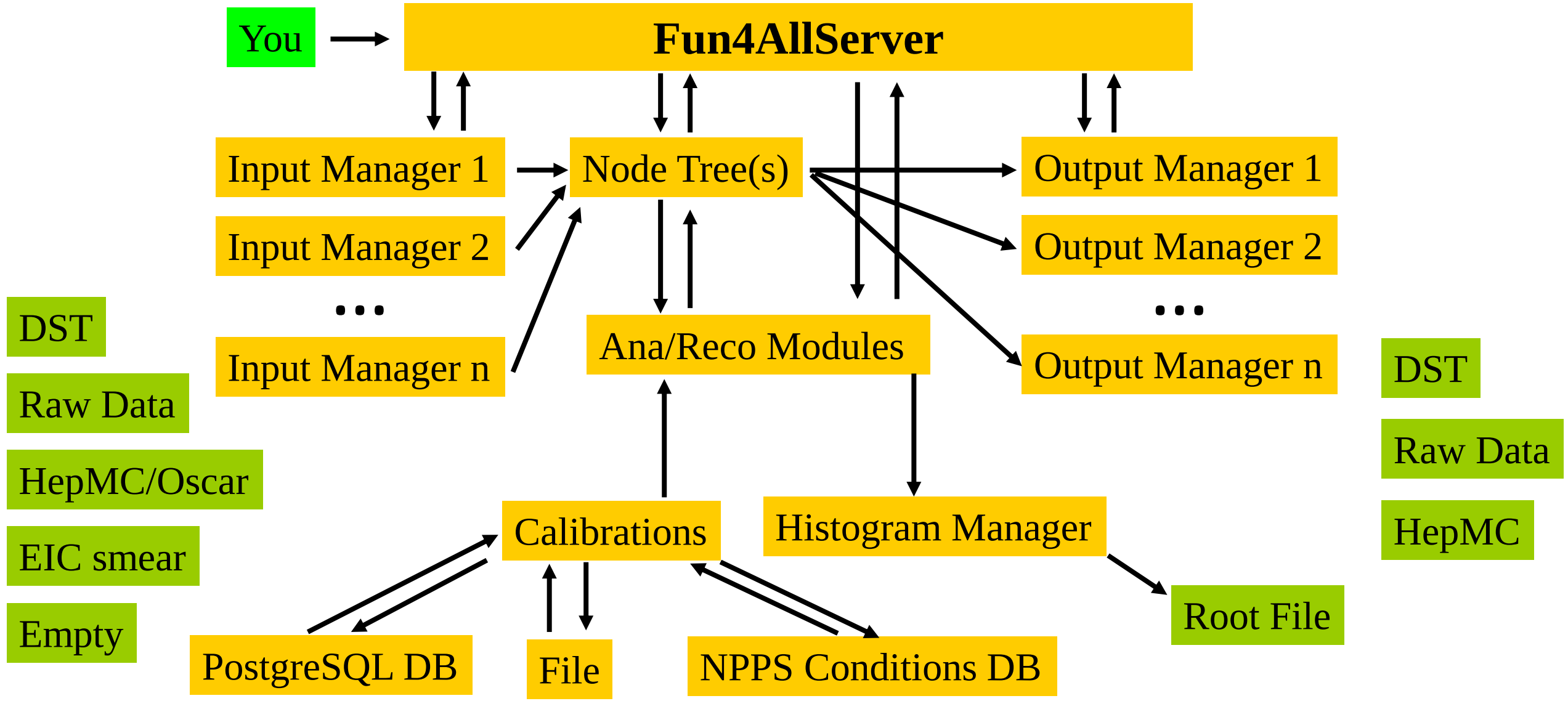
Code in
https://github.com/sPHENIX-Collaboration/coresoftware/tree/master/offline/framework

What does it do for us

- Call the analysis/reconstruction modules in the order in which they were registered (correct ordering is responsibility of the user)
- Read input files (many different types)
- Write output files (different types, automatic saving of selected data objects)
- Somewhat manages the Node Tree - our storage for data objects (more later)
- Make snapshots at any state of the reconstruction/analysis
- Access to calibrations

Fun4All has been our workhorse for 20 years, running raw data reconstruction, analysis, simulations and embedding

Structure of our framework Fun4All



That's all there is to it, no backdoor communications – steered by ROOT macros

Reconstruction in a nutshell

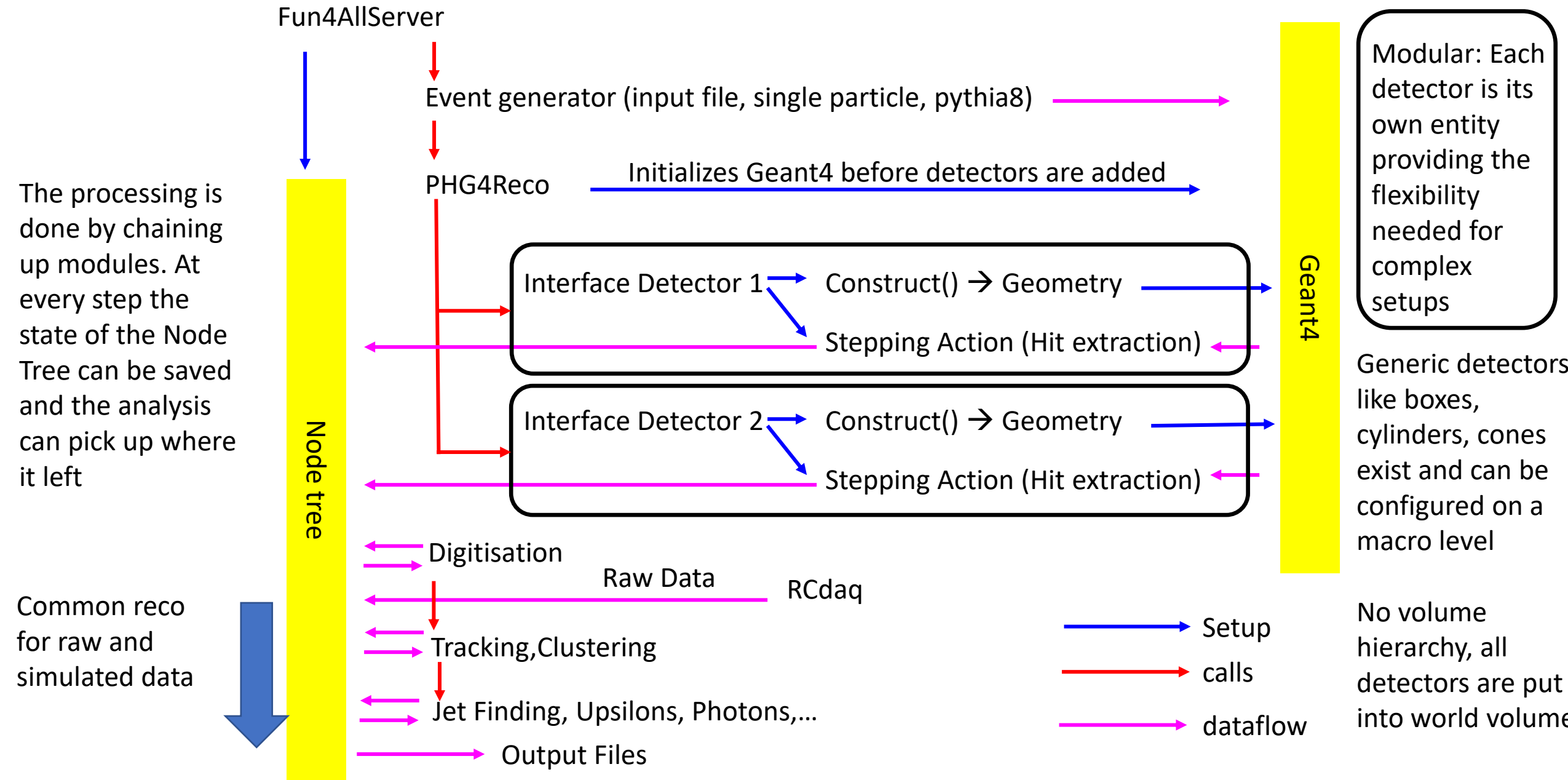
- Our reconstruction is a continuous (and ever expanding) linear chain of modules starting from the event generator (sims) and/or raw data up to your analysis module
- Modules read their input from the node tree, create their own nodes and write to them – or update existing objects
- The state of the chain (the objects on the node tree) can be saved at every step and the remaining modules can be later run reading the saved file
- Object can be written to separate files and later be recombined
  - We will reconstruct our calorimeters separately from the tracking

sPHENIX Simulations

- GEANT4 based (no GEANT3, no virtual Monte Carlo)
- Fully integrated into our analysis chain as Reconstruction Module
- Modular design – all detectors are self contained
- Configured on the macro level
- Generic detectors (boxes, cylinders, cones) exist
- sPHENIX subdetectors fully implemented
- 11 years of development

Tutorials:
https://github.com/sPHENIX-Collaboration/tutorials

G4 program flow within Fun4All



The processing is done by chaining up modules. At every step the state of the Node Tree can be saved and the analysis can pick up where it left

Common reco for raw and simulated data

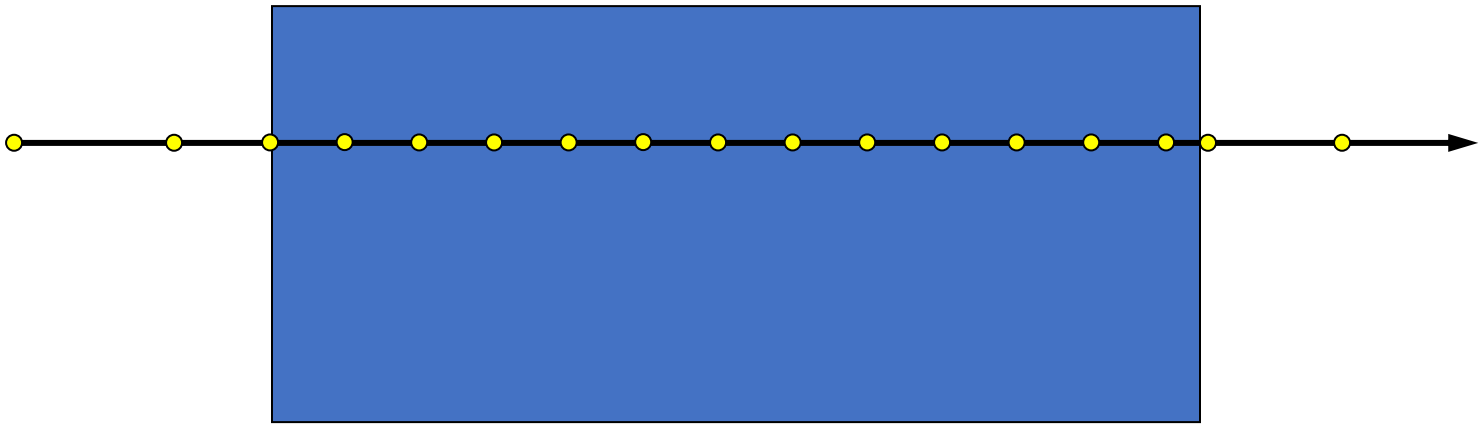
Modular: Each detector is its own entity providing the flexibility needed for complex setups

Generic detectors like boxes, cylinders, cones exist and can be configured on a macro level

No volume hierarchy, all detectors are put into world volume

GEANT steps

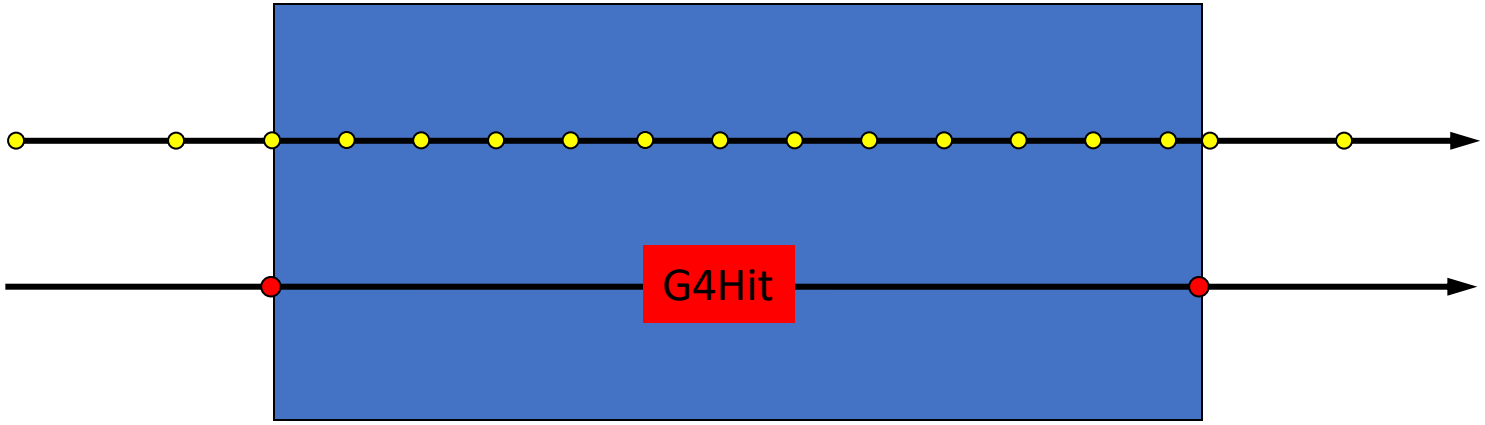
GEANT propagates particles one step at a time. The step size is determined by the physics processes associated with the current particle or when a boundary between volumes is crossed



After each step the user stepping method is called with a pointer to the current volume which has access to the full information (energy loss, particle momentum at beginning and end of step, …)

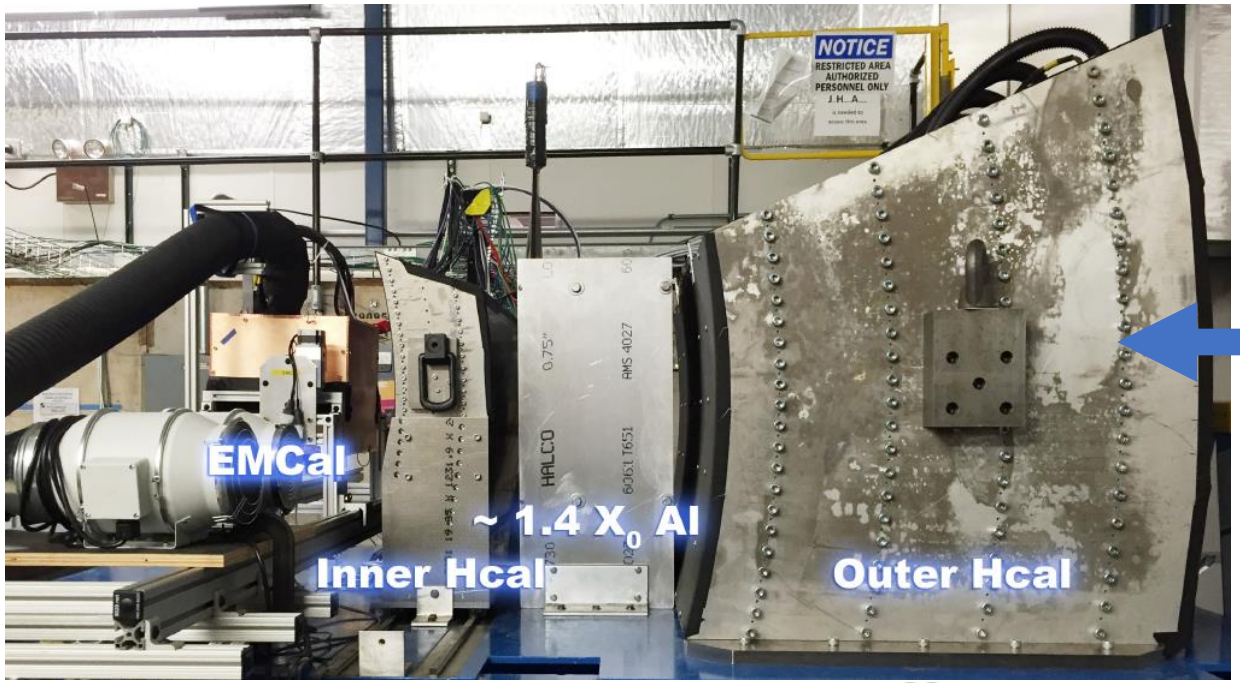
Our G4Hits (you'll hear us talking about them a lot)

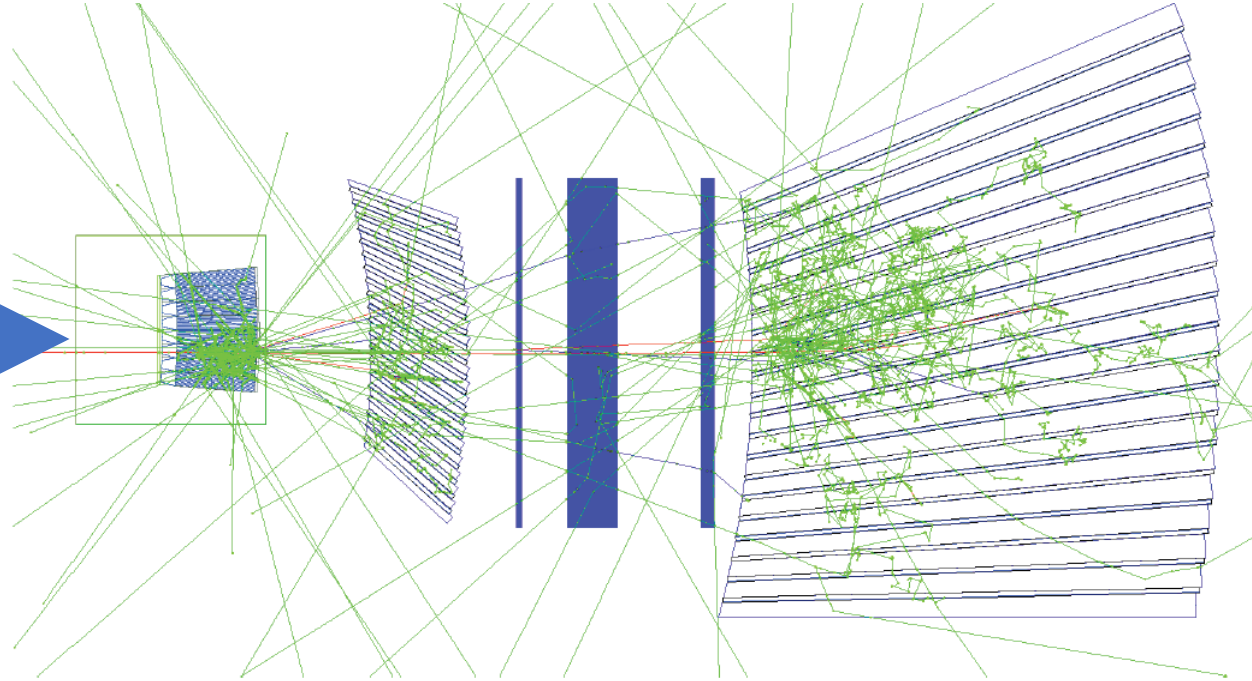
In our stepping method we add the energy loss in each volume and store the entry and exit coordinates and time (and subdetector specific info like ionization energy, light output,…)

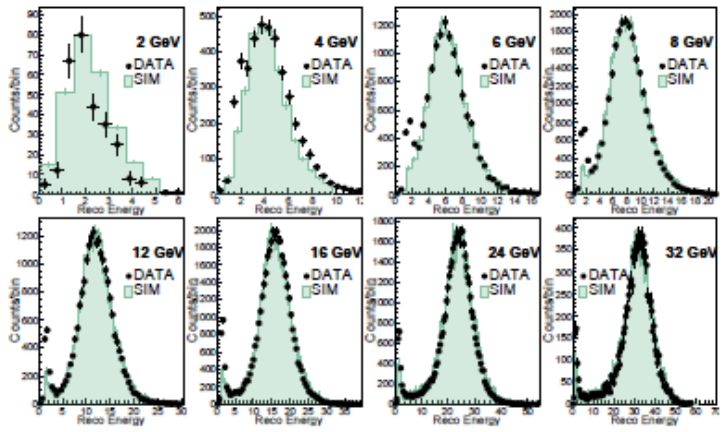


We also keep the ancestry for G4Hits so any hit can be traced back to a primary particle. To reduce size we do not store particles which do not leave G4Hits and are not in the ancestry of a particle which created a G4Hit

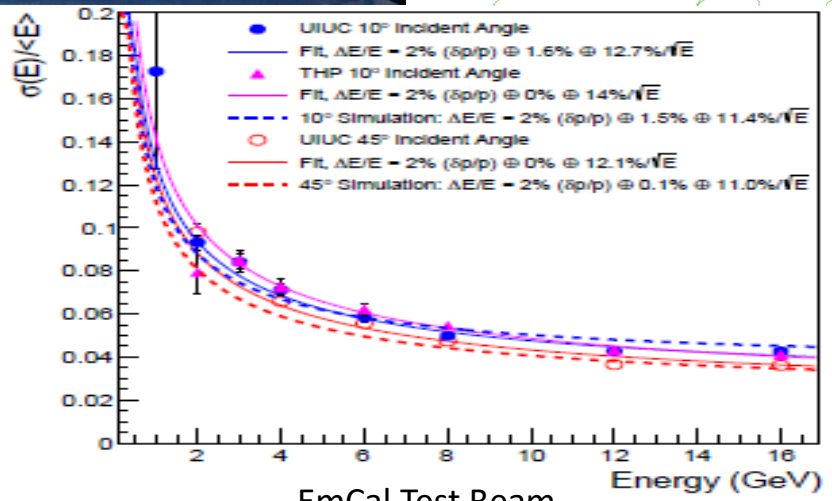
Combining data reconstruction and simulations







Hadronic Calorimeter Test Beam



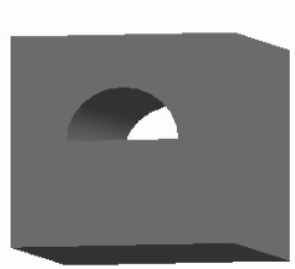
EmCal Test Beam

Simulation chain and reconstruction has been verified with real data.

Creating your own Detector

3 classes mandatory
- Subsystem
- Detector
- SteppingAction

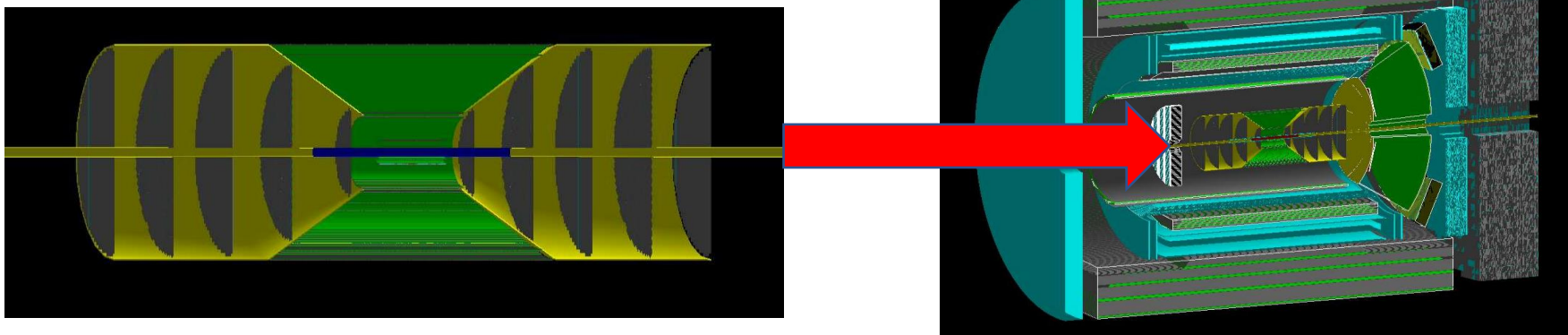
Optional
- Use of parameters
- DisplayAction



Examples implementing this beautiful box with a half pipe hole including macros:
https://github.com/sPHENIX-Collaboration/g4exampledetector

Template Example introduces script to add detector:
CreateG4Subsystem.pl <Detector Name>
options are:
--all : Also create autogen.sh, configure.ac and Makefile.am
--overwrite : overwrite existing files (handle with care, we only have snapshots of $HOME)
https://github.com/sPHENIX-Collaboration/g4exampledetector/tree/master/template

You can then add your detector to our existing setups on the macro level



What is this Node Tree which gets printed for each run?

List of Nodes in Fun4AllServer:
Node Tree under TopNode TOP
TOP (PHCompositeNode)/

DST (PHCompositeNode)/
- Sync (IO,SyncObjectv1)
- EventHeader (IO,EventHeaderv1)
- PHG4INEVENT (PHDataNode)
- PIPE (PHCompositeNode)/
  - G4HIT_PIPE (IO,PHG4HitContainer)
- MVTX (PHCompositeNode)/
  - G4HIT_MVTX (IO,PHG4HitContainer)
- INTT (PHCompositeNode)/
  - G4HIT_INTT (IO,PHG4HitContainer)
- TPC (PHCompositeNode)/
  - G4HIT_ABSORBER_TPC (IO,PHG4HitContainer)
  - G4HIT_TPC (IO,PHG4HitContainer)
- TPC_ENDCAP (PHCompositeNode)/
  - G4HIT_TPC_ENDCAP (IO,PHG4HitContainer)
- MICROMEGAS (PHCompositeNode)/
  - G4HIT_MICROMEGAS (IO,PHG4HitContainer)
- CEMC_ELECTRONICS (PHCompositeNode)/
  - G4HIT_CEMC_ELECTRONICS (IO,PHG4HitContainer)
- CEMC_SPT (PHCompositeNode)/
  - G4HIT_CEMC_SPT (IO,PHG4HitContainer)
- CEMC (PHCompositeNode)/
  - G4HIT_ABSORBER_CEMC (IO,PHG4HitContainer)
  - G4HIT_CEMC (IO,PHG4HitContainer)
  - G4CELL_CEMC (IO,PHG4CellContainer)
  - TOWER_SIM_CEMC (IO,RawTowerContainer)
  - TOWER_RAW_CEMC (IO,RawTowerContainer)
  - TOWER_CALIB_CEMC (IO,RawTowerContainer)

…

RUN (PHCompositeNode)/
- RunHeader (IO,RunHeaderv1)
- Flags (IO,FlagSavev1)
- FIELD_CONFIG (IO,PHFieldConfigv1)
- PIPE (PHCompositeNode)/
  - G4GEOPARAM_PIPE (IO,PdbParameterMapContainer)
  - CYLINDERGEOM_PIPE (IO,PHG4CylinderGeomContainer)
- MVTX (PHCompositeNode)/
  - G4GEOPARAM_MVTX (IO,PdbParameterMapContainer)
  - G4CELLPARAM_MVTX (IO,PdbParameterMap)
- INTT (PHCompositeNode)/
  - G4GEOPARAM_INTT (IO,PdbParameterMapContainer)
  - DEADMAP_INTT (IO,InttDeadMapv1)
  - G4CELLPARAM_INTT (IO,PdbParameterMap)
- TPC (PHCompositeNode)/
  - G4GEOPARAM_TPC (IO,PdbParameterMapContainer)
  - G4CELLPARAM_TPC (IO,PdbParameterMap)
  - G4TPCPADPLANE (IO,PdbParameterMap)
- TPC_ENDCAP (PHCompositeNode)/
  - G4GEOPARAM_TPC_ENDCAP (IO,PdbParameterMapContainer)
- MICROMEGAS (PHCompositeNode)/
  - G4GEOPARAM_MICROMEGAS (IO,PdbParameterMapContainer)
- CEMC_ELECTRONICS (PHCompositeNode)/
  - G4GEOPARAM_CEMC_ELECTRONICS (IO,PdbParameterMapContainer)
  - CYLINDERGEOM_CEMC_ELECTRONICS (IO,PHG4CylinderGeomContainer)
- CEMC_SPT (PHCompositeNode)/
  - G4GEOPARAM_CEMC_SPT (IO,PdbParameterMapContainer)
  - CYLINDERGEOM_CEMC_SPT (IO,PHG4CylinderGeomContainer)

…

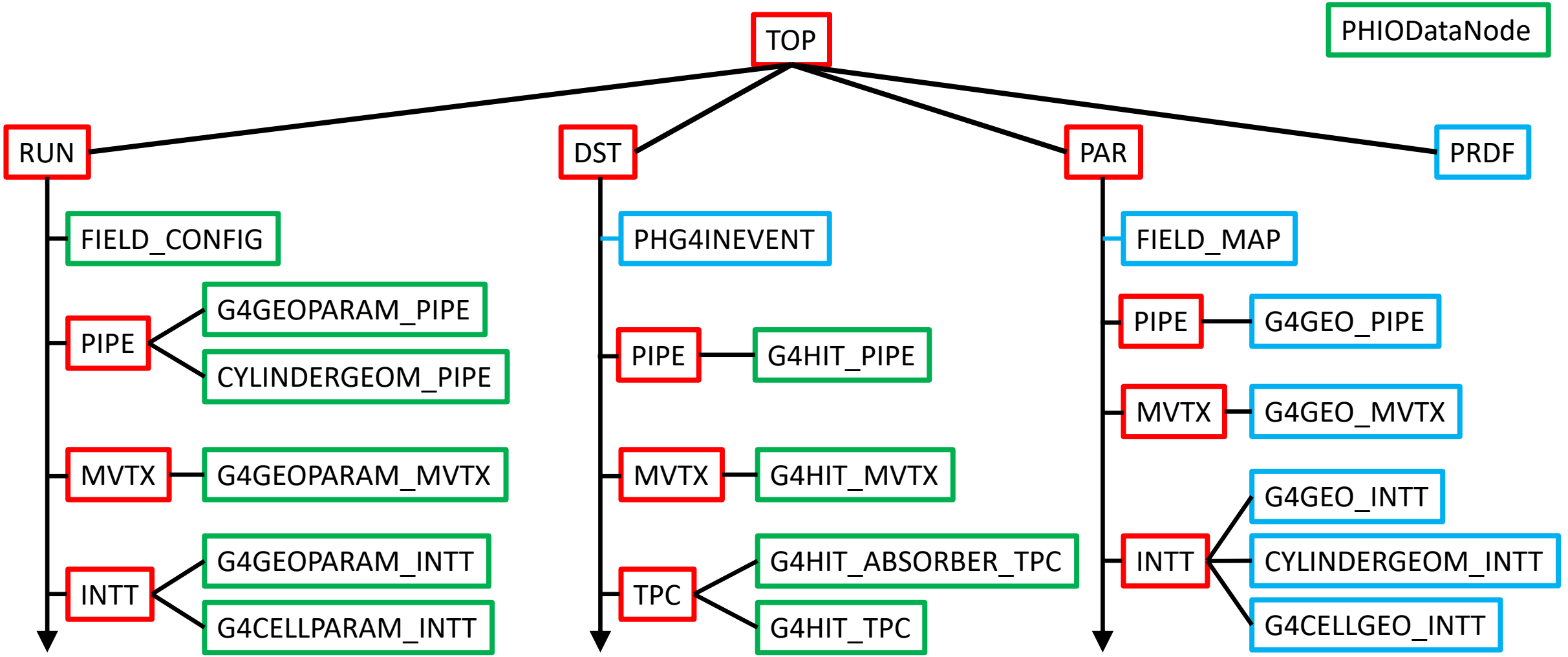
PAR (PHCompositeNode)/
- FIELD_MAP (PHDataNode)
- PIPE (PHCompositeNode)/
  - G4GEO_PIPE (PHDataNode)
- MVTX (PHCompositeNode)/
  - G4GEO_MVTX (PHDataNode)
- INTT (PHCompositeNode)/
  - G4GEO_INTT (PHDataNode)
  - CYLINDERGEOM_INTT (PHDataNode)
  - G4CELLGEO_INTT (PHDataNode)
- TPC (PHCompositeNode)/
  - G4GEO_TPC (PHDataNode)
  - G4CELLPAR_TPC (PHDataNode)
  - G4TPCPADPLANEPAR (PHDataNode)
- TPC_ENDCAP (PHCompositeNode)/
  - G4GEO_TPC_ENDCAP (PHDataNode)
- MICROMEGAS (PHCompositeNode)/
  - G4GEO_MICROMEGAS (PHDataNode)
- CEMC_ELECTRONICS (PHCompositeNode)/
  - G4GEO_CEMC_ELECTRONICS (PHDataNode)
- CEMC_SPT (PHCompositeNode)/
  - G4GEO_CEMC_SPT (PHDataNode)
- CEMC (PHCompositeNode)/
  - G4GEO_CEMC (PHDataNode)
  - G4CELLGEO_CEMC (PHDataNode)
- G4GDML_CONFIG (PHDataNode)
- HCALIN (PHCompositeNode)/
  - G4GEO_HCALIN (PHDataNode)

…

The Node Tree

- The Node Tree is at the center of the sPhenix software universe (but it's more or less invisible to you). It's the way we organize our data and make them accessible to modules
- **It is NOT a Root TTree**
- We have 3 different Types of Nodes:
  - PHCompositeNode: contains other Nodes
  - PHDataNode: contains any object
  - PHIODataNode: contains objects which can be written out to DST (Data Summary Tape
- PHCompositeNodes and PHIODataNodes can be saved to a DST and read back
- This DST contains root TTrees, the node structure is saved in the branch names. Due to Roots limitations not all objects can become PHIODataNodes. E.g. objects containing ACTS still do not work with ROOT6.
- Opening the file in root and poking with e.g. TBrowser or T->Draw() or reading works (not well for stl maps)
- We currently save 2 root trees in each output file, one which contains the eventwise information, one which contains the runwise information
- Input Managers put objects as PHIODataNodes on the node tree, output managers save selected PHIODataNodes to a file.
- Fun4All can manage multiple independent node trees

Access to Data Objects, understanding our Node Tree



Phool Node Tree for sPHENIX

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    PHG4INEVENT (PHDataNode)
    PIPE (PHCompositeNode)/
      G4HIT_PIPE (IO,PHG4HitContainer)
    MVTX (PHCompositeNode)/
      G4HIT_MVTX (IO,PHG4HitContainer)
    INTT (PHCompositeNode)/
      G4HIT_INTT (IO,PHG4HitContainer)
    TPC (PHCompositeNode)/
      G4HIT_ABSORBER_TPC (IO,PHG4HitContainer)
      G4HIT_TPC (IO,PHG4HitContainer)
    CEMC_ELECTRONICS (PHCompositeNode)/
      G4HIT_CEMC_ELECTRONICS (IO,PHG4HitContainer)
    CEMC_SPT (PHCompositeNode)/
      G4HIT_CEMC_SPT (IO,PHG4HitContainer)
    G4HIT_CEMC (IO,PHG4HitContainer)
    G4HIT_ABSORBER_CEMC (IO,PHG4HitContainer)
    HCALIN (PHCompositeNode)/
      G4HIT_ABSORBER_HCALIN (IO,PHG4HitContainer)
    ...
```

TOP: Top of Default Node Tree
Creation and populating of other node trees is possible (used for embedding)

You will see this printout of the node tree whenever the processing starts

Print it from the command line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

Phool Node Tree for sPHENIX

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    PHG4INEVENT (PHDataNode)
    PIPE (PHCompositeNode)/
      G4HIT_PIPE (IO,PHG4HitContainer)
    MVTX (PHCompositeNode)/
      G4HIT_MVTX (IO,PHG4HitContainer)
    INTT (PHCompositeNode)/
      G4HIT_INTT (IO,PHG4HitContainer)
    TPC (PHCompositeNode)/
      G4HIT_ABSORBER_TPC (IO,PHG4HitContainer)
  RUN (PHCompositeNode)/
    FIELD_CONFIG (IO,PHFieldConfigv1)
    PIPE (PHCompositeNode)/
      G4GEOPARAM_PIPE (IO,PdbParameterMapContainer)
      CYLINDERGEOM_PIPE (IO,PHG4CylinderGeomContainer)
    MVTX (PHCompositeNode)/
      G4GEOPARAM_MVTX (IO,PdbParameterMapContainer)
    INTT (PHCompositeNode)/
      G4GEOPARAM_INTT (IO,PdbParameterMapContainer)
...
```

DST and RUN Node: default for I/O

- DST – eventwise
- RUN - runwise

Objects under the DST node are reset after every event to prevent event mixing. You can select the objects to be saved in the output file. Subnodes like SVTX are saved and restored as well. DST/RUN nodes can be restored from file under other TopNodes
ROOT restrictions apply:
Objects cannot be added while running to avoid event mixing

You will see this printout of the node tree whenever the processing starts

Print it from the command line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

Phool Node Tree for sPHENIX

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    PHG4INEVENT (PHDataNode)
    PIPE (PHCompositeNode)/
      G4HIT_PIPE (IO,PHG4HitContainer)
    MVTX (PHCompositeNode)/
      G4HIT_MVTX (IO,PHG4HitContainer)
RUN (PHCompositeNode)/
    FIELD_CONFIG (IO,PHFieldConfigv1)
    PIPE (PHCompositeNode)/
      G4GEOPARAM_PIPE (IO,PdbParameterMapContainer)
      CYLINDERGEOM_PIPE (IO,PHG4CylinderGeomContainer)
    MVTX (PHCompositeNode)/
      G4GEOPARAM_MVTX (IO,PdbParameterMapContainer)
    INTT (PHCompositeNode)/
      G4GEOPARAM_INTT (IO,PdbParameterMapContainer)
PAR (PHCompositeNode)/
    FIELD_MAP (PHDataNode)
    PIPE (PHCompositeNode)/
      G4GEO_PIPE (PHDataNode)
...
```

Users can add their own PHCompositeNodes Under the TOP Node. But then resetting the objects is their responsibility.

The PAR node hold more complicated geometry Objects which we do not want to save on DST

You will see this printout of the node tree whenever the processing starts

Print it from the command line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

Phool Node Tree for sPHENIX

```
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
  DST (PHCompositeNode)/
    PHG4INEVENT (PHDataNode)
    PIPE (PHCompositeNode)/
      G4HIT_PIPE (IO,PHG4HitContainer)
    HCALIN (PHCompositeNode)/
      G4HIT_ABSORBER_HCALIN (IO,PHG4HitContainer)
      G4HIT_HCALIN (IO,PHG4HitContainer)
      G4CELL_HCALIN (IO,PHG4CellContainer)
      TOWER_SIM_HCALIN (IO,RawTowerContainer)
      TOWER_RAW_HCALIN (IO,RawTowerContainer)
      TOWER_CALIB_HCALIN (IO,RawTowerContainer)
      CLUSTER_HCALIN (IO,RawClusterContainer)
    BBC (PHCompositeNode)/
      BbcVertexMap (IO,BbcVertexMapv1)
    TRKR (PHCompositeNode)/
      TRKR_HITSET (IO,TrkrHitSetContainer)
      TRKR_HITTRUTHASSOC (IO,TrkrHitTruthAssoc)
      TRKR_CLUSTER (IO,TrkrClusterContainer)
      TRKR_CLUSTERHITASSOC (IO,TrkrClusterHitAssoc)
...
```

Type of Node is given (IO is PHIODataNode)

Class of Data IO Object is given (you will need to know this when accessing the data)

Caveat: You loose ownership once an object is put on the node tree. Fun4All deletes the node tree when cleaning up. Deleting nodes is not supported (if you give me a good reason I'll work on that but only for nodes not under DST or RUN)

You will see this printout of the node tree whenever the processing starts

Print it from the command line with
Fun4AllServer *se = Fun4AllServer::instance();
se->Print("NODETREE");

Your Analysis Module

You need to inherit from the SubsysReco Baseclass (offline/framework/fun4all/SubsysReco.h) which gives the methods which are called by Fun4All. If you don't implement all of them it's perfectly fine (the beauty of base classes)

- Init(PHCompositeNode *topNode): called once when you register the module with the Fun4AllServer
- InitRun(PHCompositeNode *topNode): called whenever data from a new run is encountered
- process_event (PHCompositeNode *topNode): called for every event
- ResetEvent(PHCompositeNode *topNode): called after each event is processed so you can clean up leftovers of this event in your code
- EndRun(const int runnumber): called before the InitRun is called (caveat the Node tree already contains the data from the first event of the new run)
- End(PHCompositeNode *topNode): Last call before we quit

If you create another node tree you can tell Fun4All to call your module with the respective topNode when you register your modue

Okay, How do I navigate the Node Tree which is in every argument????

You need to know the name of the node and the class of the object you want (e.g. some PHG4HitContainer version in the node called G4HIT_HCALIN (that's where the Fun4All printout of the Node Tree comes in handy)

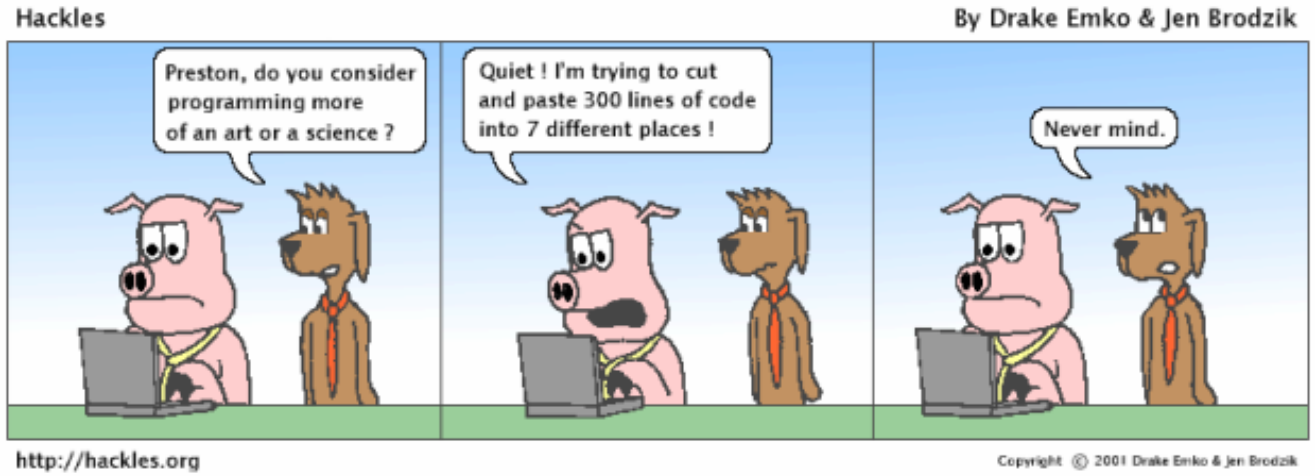
topNode of the node tree your module is registered with

```
#include <g4hit/PHG4HitContainer.h>
#include <fun4all/getClass.h>

Myanalysis::process_event(PHCompositeNode *topNode)
{
   PHG4HitContainer *g4hits =
findNode::getClass<PHG4HitContainer>(topNode,"G4HIT_HCALIN");
 if (g4hits)
   …
}
```

Caveat: getClass will return the first node of a given name, if you have have multiple identically named nodes you need to search differently

Writing your own Analysis Module



To get help, type: CreateSubsysRecoModule.pl

Usage:
CreateSubsysRecoModule.pl <Module Name>
options:
--all : create also autogen.sh, configure.ac and Makefile.am
--overwrite : overwrite existing files

Creates <Module Name>.h and <Module Name>.cc with a cout in all available methods. Much better starting point than cutting and pasting from an existing module. See our tutorial (with macro to run it): https://github.com/sPHENIX-Collaboration/tutorials/tree/master/CreateSubsysRecoModule

The Makefile.am and configure.ac do not contain the instructions needed for i/o classes
Sorry - you will still need to cut and paste this from existing sources

Coding Conventions:
What is the problem we try to solve?

- The Simple one: Readability, our code will stay around for a long time, other people (e.g. me) will have to work with your code and maintain it. Uniform look and feel helps tremendously (clang-format does this for us)

→ https://wiki.sphenix.bnl.gov/index.php/Codingconventions#Style_reformatting_tools

- The difficult one: What do we allow to use (exceptions, gotos, ROOT,3rd party,…)
  - https://wiki.sphenix.bnl.gov/index.php/Codingconventions
    - Short and somewhat useful – please read
  - Long writeup: https://wiki.sphenix.bnl.gov/images/f/f0/Coding_standards.pdf
  - Start to use clang-tidy to apply some coding conventions
- In Reality: Most code is cut and pasted, we try to keep our examples and tutorials up to date and keep cleaning our software

sPHENIX Repositories

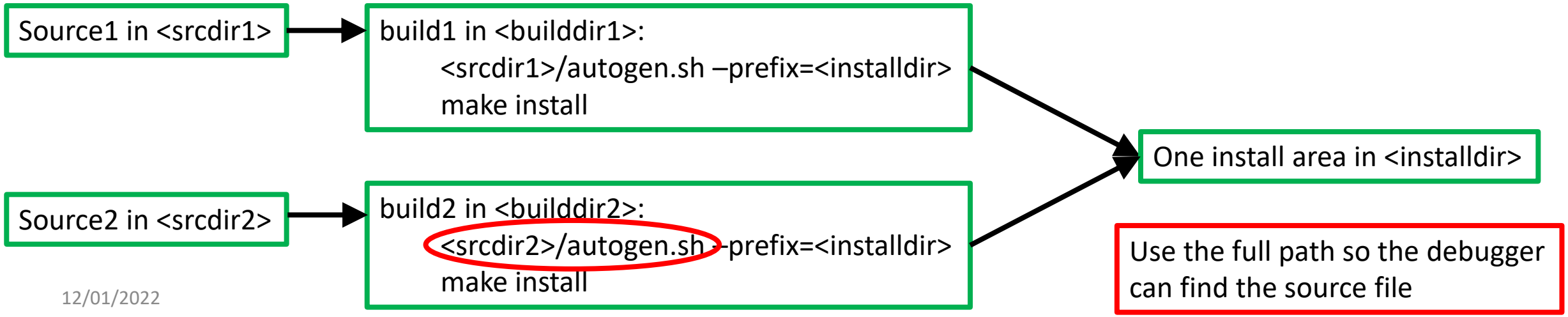
- https://github.com/sPHENIX-Collaboration/coresoftware main code repository
- https://github.com/sPHENIX-Collaboration/online_distribution Raw data handling software
- https://github.com/sPHENIX-Collaboration/macros standard macros
- https://github.com/sPHENIX-Collaboration/calibrations calibrations (outside of conditions DB)

- https://github.com/sPHENIX-Collaboration/analysis user analysis's
- https://github.com/sPHENIX-Collaboration/tutorials tutorials

How to build a package

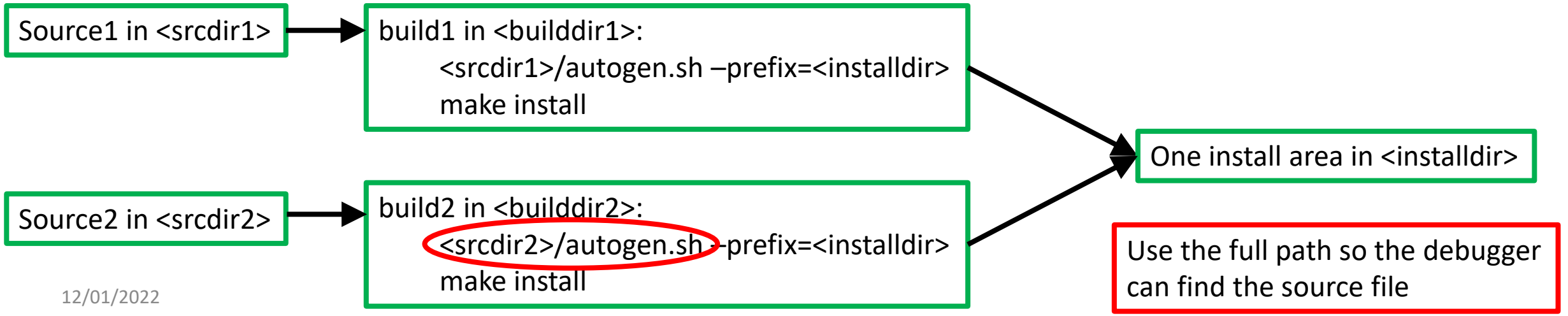
- We use autoconf/automake (configure) to build our code
  - This does put some files into your source area, be careful what you commit
- Each package (directory) is build by itself
  - You only have to build the package you are working on

Keep your source and build areas separately, use common install area



How to use your compiled package

- The default setup from the sphenix setup scripts does not add your installation area
- ROOT6 is peculiar with finding includes
- So we have a script for that:
  source $OPT_SPHENIX/bin/setup_local.csh (or sh) <installdir>



Different setups

We have many different builds of our software for different uses:

- new → default: daily build of our current software (the latest and greatest)
- play → test we are running (new G4 or root version or whatever we are testing)
- ana → weekly archival build (recommended for stable environment)
- new+insure → build using insure compiler (runtime code checker)
- clang → new build using clang instead of gcc
- debug → new build without optimizer

Switch between versions using the sphenix setup script:

source /opt/sphenix/core/bin/sphenix_setup.csh (or sh) –n <build type>
<build type>.n gets specific version (e.g. ana.115 sets up to use archival build 115)

All builds except ana have version numbers (1-4) and roll over after 4 days, you have to rebuild your private packages every 4 days to avoid inconsistencies between your build and the installed software. Use a fixed ana build for longer term projects (but update from time to time to a newer ana build)

Outside BNL

Running outside of RACF, you need either the keys for /cvmfs/sphenix.sdcc.bnl.gov or use /cvmfs/sphenix.opensciencegrid.org which is a publicly available mirror of the sdcc volume. Use:

source /cvmfs/sphenix. opensciencegrid.org/default/opt/sphenix/core/bin/sphenix_setup.csh (or sh) –n <build type>

Running on your local desktop/laptop:
We have singularity and docker containers with the SDCC farm image, our new build is tarred up with every build for distribution:
https://github.com/sPHENIX-Collaboration/Singularity

Container and its builds are provided on a best effort basis, you do need to run in SDCC "for real" to get access to disk space, input files and thousands of condor slots)

Resources

Accounts

- Native sPHENIX account
  - Shell is bash
  - $HOME under /sphenix/u
  - 3GB quota on home disk
  - Write permission to /sphenix/user
  - Interactive machines sphnx01, sphnx02
- Any sdcc account can be used after sourcing the sphenix setup script
  - Cannot test non PHENIX accounts (STAR,…), let me know about problems
  - sPHENIX resources (disk+cpu) only available for sPHENIX accounts
    →Probably you should consider getting an sphenix account
- SDCC policy requires separate accounts, no more adding of sphenix group to existing accounts

Disk space

- For your use:
  - sPHENIX home dir 3GB quota
    - Fast nvram appliance
  - /sphenix/users 5TB quota
- Common simulations:
  - /sphenix/sim/sim01: 200TB
  - /sphenix/sim/sim02: 20TB
- Data (testbeam)
  - /sphenix/data/data01: 10TB
  - /sphenix/data/data02: 250TB
- TG space (soon): 2PB
- Lustre: 35PB (60PB in 2023)
  - /sphenix/lustre01/sphnxpro

"I back up my files religiously. I pray nothing happens to them."

Except home dir, nothing is backed up, so don't ask. Home dir backup uses snapshots which go back for a week.

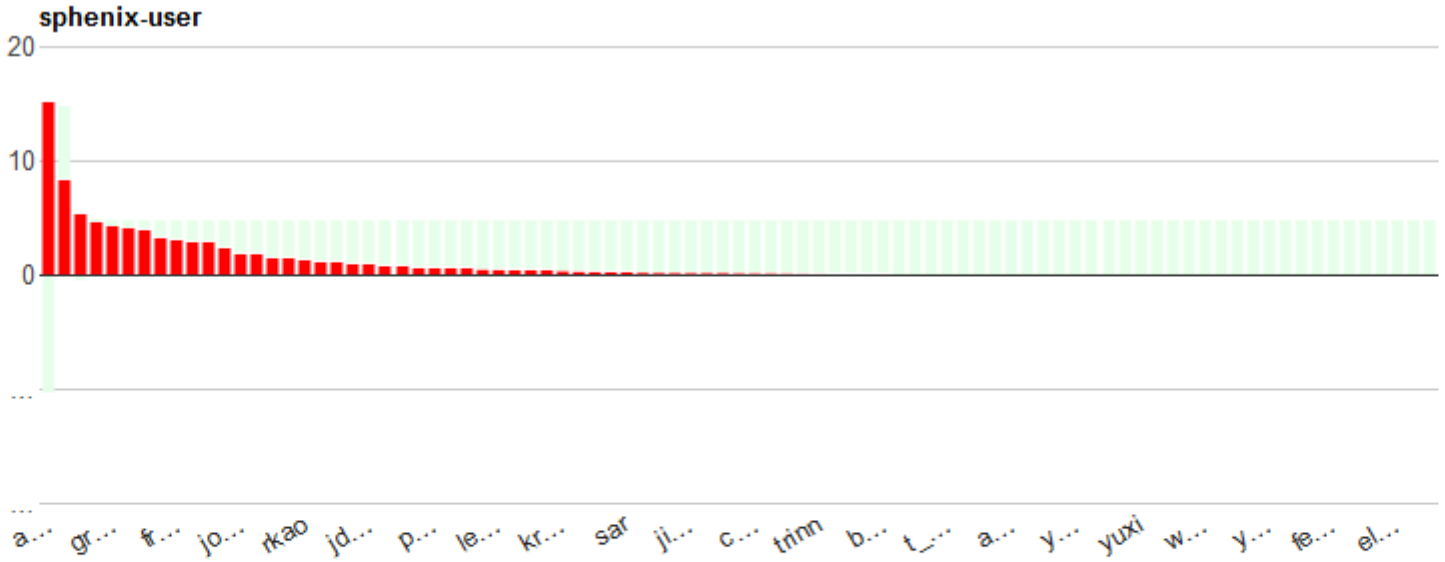
Users can restore their own files from snapshots from /sphenix/u/.snapshot

Standard disclaimer:
It's a bit of a mess, the plan is to keep /sphenix/users for users and put Sim and data disks under sphnxpro account (move user files off those)

Check your own gpfs usage on /sphenix/user

https://monitoring.sdcc.bnl.gov/Facility/GCE/GPFS/index.html

- atlasgpfs01
- gpfs
- gpfs02
  - lsst
  - astro
  - sphenix
    - sphenix-data01
    - sphenix-data02
    - sphenix-data03
    - sphenix-decadal
    - sphenix-sim01
    - sphenix-sim02
    - sphenix-user
  - cvmfst0
    - cvmfst0
  - phenix
  - biology
  - usatlas
  - eic
- gpfs04
- all



User

	User	Size TB	Used TB	Free TB
1	root	0		
2	abhisek	5	15.34	-10.34
3	gcha	5	5.49	-0.49
4	bogui	5	4.71	0.29
5	gregtom3	5	4.34	0.66
6	xuanlipx	5	4.2	0.8
7	giorgian	5	4.1	0.9
8	lajoie	5	3.34	1.66
9	frawley	5	3.17	1.83
10	isibf5y	5	3.04	1.96
11	chase	5	2.98	2.02
12	vassalli	5	2.48	2.52
13	jouellette	5	1.99	3.01
14	anatrain	5	1.91	3.09
15	aromeroh	5	1.61	3.39
16	yuhw	5	1.53	3.47
17	rkao	5	1.42	3.58
18	yzhai	5	1.31	3.69

A lot faster than sending me a mail asking if your quota is exceeded 

Condor

More details in our wiki:
https://wiki.sphenix.bnl.gov/index.php/Condor

Current condor usage:
https://web.sdcc.bnl.gov/Facility/LinuxFarm/cm.html

- sdcc has 50000 condor slots in a shared pool (mainly PHENIX, ATLAS, STAR nodes)
  - 3 cpu day limit
  - 1.5GB default memory limit
- sPHENIX lumped together with PHENIX “owns” ~17000 of those
  - we have priority for this number of slots
  - If other nodes are idling we can use those additional slots
  - Main use for PHENIX is the analysis taxi
- sPHENIX farm: 14400 cores for Mock Data Challenge + Simulation production

- sPHENIX simulation jobs need around 4GB
  - Add *request_memory = 4GB* to the condor job file

Accessing Files from our Mock Data Challenge

- Separated into Runs, where a run corresponds to a certain geometry
  - Current default is the old geometric hcal (pre-engineering drawings)
  - Caveat – productions might differ in details, lots of testing none of them is "final"
- CreateFileList.pl gives lists of available files which can be directly used in Fun4All, details in our wiki
  - https://wiki.sphenix.bnl.gov/index.php/MDC2_2022#Access_Samples
  - *"CreateFileList.pl"* will give a short summary of options
- Jets need to be reconstructed, we need Tracks, Calorimeter Clusters and the Truth Info for the evaluation (type 11 are 30 GeV pythia8 jets, request only 100 events to make this quick):
  - CreateFileList.pl -type 11 DST_TRACKS DST_TRUTH DST_CALO_CLUSTER -n 100

HELP!!!!!!!

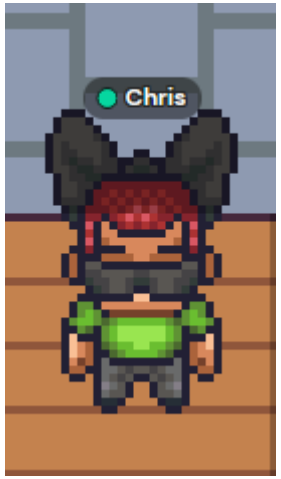
- Subscribe to the software list:
  https://lists.bnl.gov/mailman/listinfo/sphenix-software-l

- Debugging tools
  https://wiki.sphenix.bnl.gov/index.php/Tools

- Mattermost chat (BNL hosted open source slack). SDCC accounts can sign up. Due to recent security incident no email invites for the time being:
  https://chat.sdcc.bnl.gov/sphenix/channels/sphenix-software

- Office hours, Tuesdays at 3:00pm EST in gathertown (link expires after 30 days, update in Mattermost)
  https://app.gather.town/invite?token=cvDkUTYhxFU0JNpU5SrMnQE6v-JC-VCT

- Last *resort:*
  Send mail to Jin Huang or Chris Pinkenburg (both works best)



Look for this guy

What's next?

- It's only 120 days before sPHENIX sees beam
  → Now is high time to get involved
- Our software is a work in progress – it will continue to evolve (rapidly) for use by sPHENIX
  →New ideas are very welcome, contributions even more so
- But **keep it simple** – our software is a tool – not a monument