# G4HP Update 8/11/22

Robert Chirco

ILLINOIS INSTITUTE OF TECHNOLOGY

# G4HP

-Geant4-based package that simulates a beam of particles incident on a cylindrical target

-includes scripts to calculate hadron production yields and invariant cross sections using several physics lists

-motivation: PPFX currently includes MC that uses FTFP_BERT → DUNE will rely on simulations using QGSP_BERT → need new MC

-in the process can make G4HP more user-friendly

# Current Workflow

1. $ ./ProcessG4HP –njobs=[nj] –nevts=[ne] –physics=[l] –particle=[p] –energy=[e] –outdir=[path/to/pnfs]
2. $ ./CreateYields [e] [text_file.txt] [yields_file.root] [qe_file.txt] [bool]

   -text_file.txt contains absolute paths to g4hp ntuples

   -bool=true for not including particles from fast decays in the histograms

3. $ ./CreateInvXS [e] [POT]  [yields_file.root] [invxs_file.root]

…and no formal way to convert these into usable ntuples for PPFX

# 1. g4hp_job.sh

-called by ProcessG4HP.py → actually runs the g4hp executable on the grid and moves the files to the output directory in pnfs

-modified to run CreateYields as well

# 2. CreateYields.C

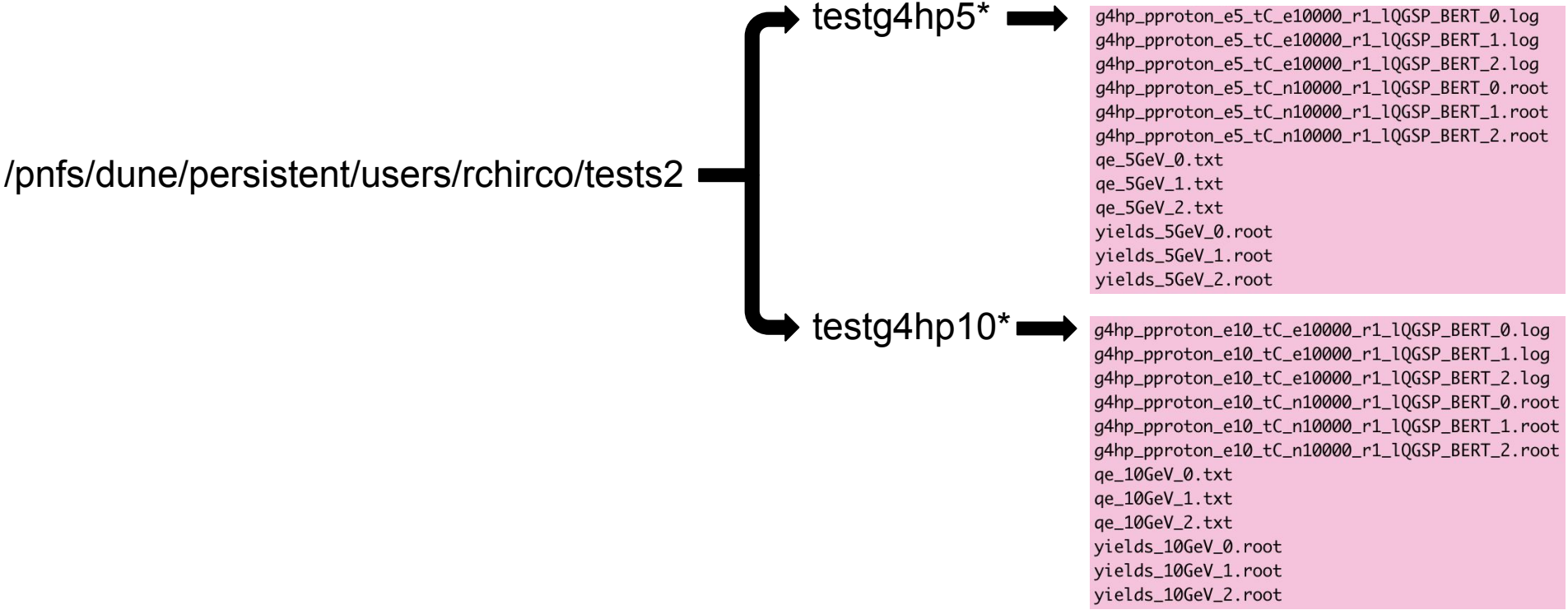
-can take a very long time to run locally → will integrate this step into a grid job

-starts by looping over a text file and adding each line (each g4hp ntuple to a TChain) → does not work on the grid

-changed to loop over a directory path

$ ./CreateYields [e] [directory_path] [yields_file.root] [qe_file.txt] [bool]

# 1. + 2. Output of ProcessG4HP.py (Updated Version)




* testg4hp5 and testg4hp10 are the output directories created each time ProcessG4HP is called, where 5 (10) refers to the energy (in GeV) of the simulated beam

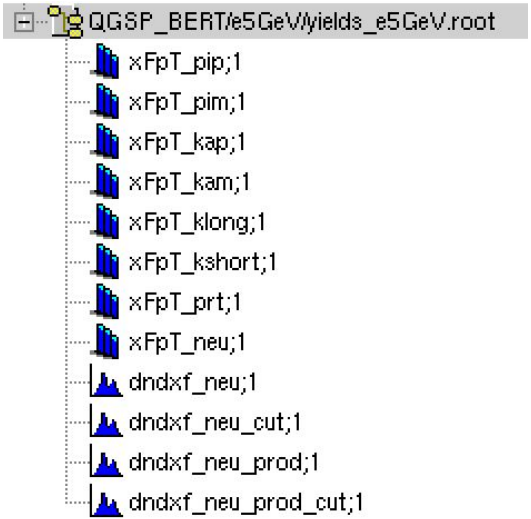
# 3. CreateInvXS.C

-now creates histograms for $K_{long}$ and $K_{short}$ → PPFX contains MC with these particles

$ ./CreateInvXS [e] [POT] [yields_file.root] [invxs_file.root]

# Structure of Yields/InvXS Files

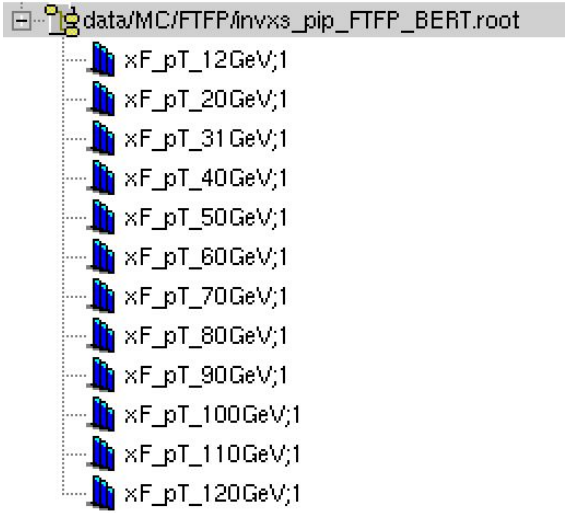
-organized by particle





# Structure of PPFX MC Files

-organized by energy



# 4. CreatePPFX.C (new!)

-takes two text files with the paths to invariant cross section and yields ntuples and converts them into QGSP_BERT (or other physics list) ntuples for PPFX

$ ./CreatePPFX [invxs_files.txt] [yields_files.txt] [physics_list]

-invxs_files.txt: text file with all invxs file paths

-yield_files.txt: text file with all yields file paths

-physics_list: a string that specifies the

physics list used in the G4HP simulations

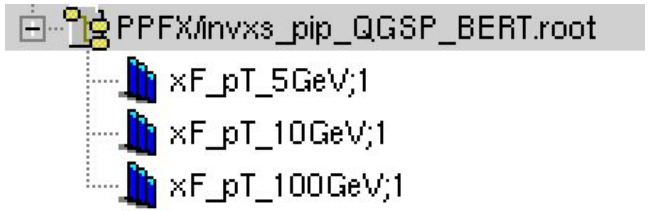
(i.e. FTFP_BERT, QGSP_BERT, etc.)

-each text file contains files named by increasing energy, where each ntuple is organized by particle

-PPFX files are the opposite → ntuples are named by the particle and are organized by increasing energy

Output of CreatePPFX:

```
-bash-4.2$ ls PPFX
invxs_kam_QGSP_BERT.root      invxs_pim_QGSP_BERT.root
invxs_kap_QGSP_BERT.root      invxs_pip_QGSP_BERT.root
invxs_klong_QGSP_BERT.root    invxs_prt_QGSP_BERT.root
invxs_kshort_QGSP_BERT.root   yield_neu_QGSP_BERT.root
invxs_neu_QGSP_BERT.root
```



# make_ntuples.sh (new!)

-combines steps 2, 3, and 4 into one script → user provides specific/path that contains subdirectories from each g4hp grid job

source make_ntuples.sh [specific/path]

*ex. [specific/path] = /pnfs/dune/persistent/users/rchirco/tests2*
*from slide 5*

1. Adds all [yields_file.root] of each directory in [specific/path] together with hadd into a total yields file [tot_yields_file.root]
2. Runs CreateInvXS on the [tot_yields_file.root] for each directory in [specific/path] and stores them in a new directory and subdirectory
3. Writes and sorts all [xs_file.root] and [tot_yields_file.root] into text files for use in CreatePPFX
4. Runs CreatePPFX

Output of make_ntuples.sh:

```
-bash-4.2$ ls QGSP_BERT
e100GeV  invxs_by_energy.txt
e10GeV   yields_by_energy.txt
e5GeV
-bash-4.2$ ls QGSP_BERT/e5GeV/
xs_e5GeV.root
yields_e5GeV.root
```

```
-bash-4.2$ ls PPFX
invxs_kam_QGSP_BERT.root     invxs_pim_QGSP_BERT.root
invxs_kap_QGSP_BERT.root     invxs_pip_QGSP_BERT.root
invxs_klong_QGSP_BERT.root   invxs_prt_QGSP_BERT.root
invxs_kshort_QGSP_BERT.root  yield_neu_QGSP_BERT.root
invxs_neu_QGSP_BERT.root
```

# New Workflow

1. $ ./ProcessG4HP –njobs=[nj] –nevts=[ne] –physics=[l] –particle=[p] –energy=[e] –outdir=[path/to/pnfs]
2. $ source make_ntuples.sh [specific/path]

# Next Steps

-Write a document summarizing all changes in detail

-Suggestions? Comments?

# Backup Slides

Robert Chirco

# Implementing QGSP_BERT

-Simulated 20 billion POT on carbon target QGSP_BERT in G4HP for: 12, 20, 31, 40, 50, 60, 70, 80, 90, 100, 110, 120, 158 [GeV]

-Calculated the production yields and cross sections → change file format to structure of the ntuples in data/MC/FTFP

    -ntuples for each particle type (invxs for pions, kaons, protons and yields for neutrons)

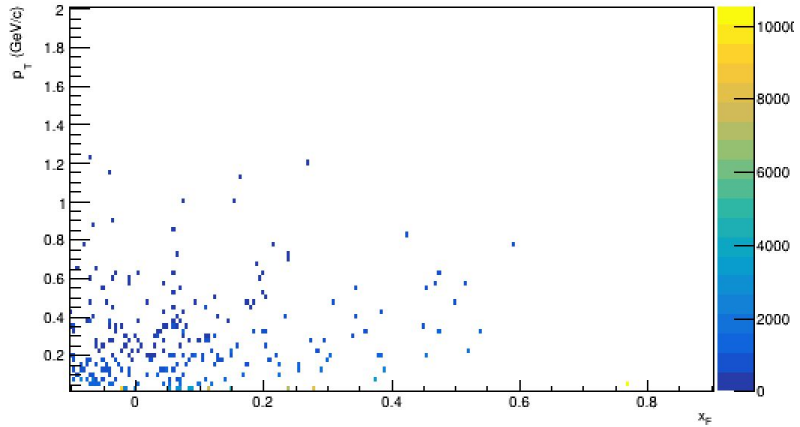
    -Each ntuple is arranged in order of increasing energy

-new directory data/MC/QGSP_v4_10_1_p02

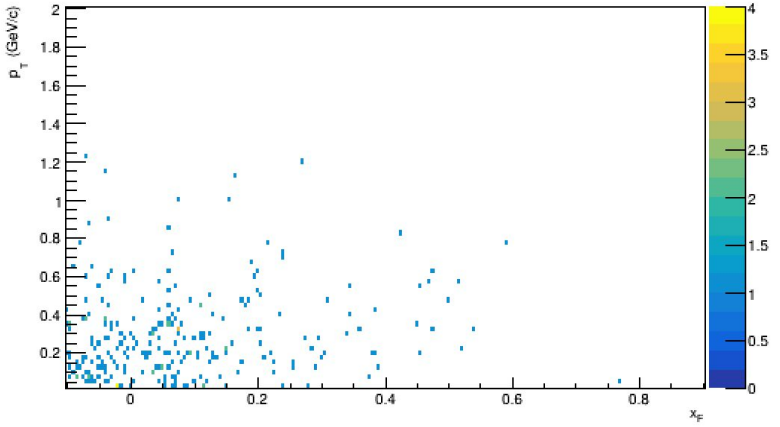
-ThinTargetMC.cpp calls the ntuples here based on $MODE = OPT or REF

# Some Histograms → 5 GeV, 10,000 POT, pC → $\pi^+X$

Invariant XS for $\pi^+$




Yields for $\pi^+$



# CreateYields.C (old)

```
void CreateYields(Int_t Mom, const char* infiles, const char*
out_histfile, const char* out_QEfile, bool include_ff){

  std::ifstream ifs;

  …

  TChain* evts = new TChain("hAinfoTree");
  ifs.open(infiles);

  …

  while (ifs.good()) {
    getline(ifs,line);
    std::cout<<line;
    if(line.find(".root")>10000)continue;
    evts->Add(line.c_str());
    std::cout<<"Adding ntuple at : "<<line<<std::endl;
  }

  ifs.close();

  …

}
```

# CreateYields.C (new)

```
void CreateYields(Int_t Mom, const char* directory_path, const
char* out_histfile, const char* out_QEfile, bool include_ff){
  …

  TChain* evts = new TChain("hAinfoTree");
  …

  DIR *dh;
  struct dirent * contents;

  dh = opendir ( directory_path );

  if ( !dh )
  {
    cout << "The given directory is not found";
    return;
  }
  while ( ( contents = readdir ( dh ) ) != NULL)
  {
    string linename = contents->d_name;
    size_t pos_dot = linename.find(".");
    if (pos_dot == 0) continue;
    evts->Add(Form("%s/%s",directory_path,linename.c_str()));
    std::cout<<"Adding ntuple at : "<<linename<<std::endl;
  }
}
```

# CreatePPFX.C Details

```
void CreatePPFX(const char* invxsfiles, const char*
yieldfiles, const char* physicslist){

  std::ifstream ifs;

  …

  TChain* evts = new TChain("hAinfoTree");
  ifs.open(invxsfiles);

  …

  while (ifs.good()) {
    getline(ifs,line);
    std::cout<<line;
    if(line.find(".root")>10000)continue;
    evts->Add(line.c_str());
    std::cout<<"Adding ntuple at : "<<line<<std::endl;
  }

  ifs.close();

  …

}
```

```
const Int_t Npart = 8;
const char* part[Npart]  = {"pip","pim","kap","kam","klong","kshort","prt","neu"};
TH2D* getvhist(TFile* fIn, int ipart);

void CreatePPFX(const char* invxsfiles, const char* yieldfiles, const char*
physicslist){

…

//loop over TFiles

  TObjArray *fileElements=chain->GetListOfFiles();

  TIter next(fileElements);
  TChainElement *chEl=0;
  while (( chEl=(TChainElement*)next() )) {
        TFile* f = new TFile(chEl->GetTitle(),"UPDATE");
        for(int jj=0;jj<Npart;jj++){
                TH2D* htemp = getvhist(f,jj);
                …
        }
  }
…
}

TH2D* getvhist(TFile* fIn, int ipart){
  //get histogram from file
  TH2D* vtmp;
  fIn->GetObject(Form("sigma_xFpT_%s",part[ipart]),vtmp);
  return vtmp;
}
```

# make_ntuples.sh Details

```
#!/bin/bash
…
pot=0
  for file in "$dir"/g4hp*.root;
  do
    nevts=`echo $file | cut -d'_' -f 5`
    #echo $f1 | wc -c
    pos_pot=`echo ${#nevts}`
    #echo $pos_pot
    jobpot=`echo ${nevts:1:$pos_pot}`
    pot=$(($pot+$jobpot))
  done
  echo "The POT is: "$pot

  string="hadd -f yields_e${job_e}GeV.root "
  for file in "$dir"/yields*.root;
  do
      string+="$file "
  done

  echo
  echo "Combining all the yields files now."
  echo "Running: $string"
  #echo "Running: hadd -f yields_e${2}.root $1/yields_${2}GeV_r{$3..$4}.root"
  #hadd -f yields_e${2}.root $1/yields_${2}GeV_r{$3..$4}.root
  eval " $string"

  echo
  echo "Creating the invariant cross sections."
  echo "Running: ./ana/CreateInvXS $job_e $pot yields_e${job_e}GeV.root xs_e${job_e}GeV.root"
  ./ana/CreateInvXS $job_e $pot yields_e${job_e}GeV.root xs_e${job_e}GeV.root

…
```