Data Mining DST format

G.Gavalian (ODU)

Project Outline

- Data mining project aims to collect the data from multi-hadron runs in one place.
- Provide access to the whole data set to participating universities.
- Provide universal analysis tools for all data sets (data selection, momentum corrections, fiducial cuts).
- Provide easy framework for combining data from different data sets.
- Provide SOA based multi-process analysis.

How we plan to do it ?

- The processed data will be transferred into a new DST format based on HDF5.
- The data will be stored at ODU, and access will be provided using CLARA as event distributor.
- A framework was developed around the HDF5-DST's for easy (UNIFIED) analysis of the data.
- The framework provides Python front end with essentially no programming, compiling, and linking required.
- Complicated analysis can be also run on the server with the NTUPLE output.

Why another DST format ?

- Current existing formats used in CLAS lack features necessary for large data set analysis.
  - BOS does not have non-sequential READ/WRITE, not suitable for multi-process analysis.
  - Objects can not be stored in the BOS file, additional information (calibration constants, maps) can not be added to the file.
  - Rigid data formats, the data format has to be the same for all data sets.

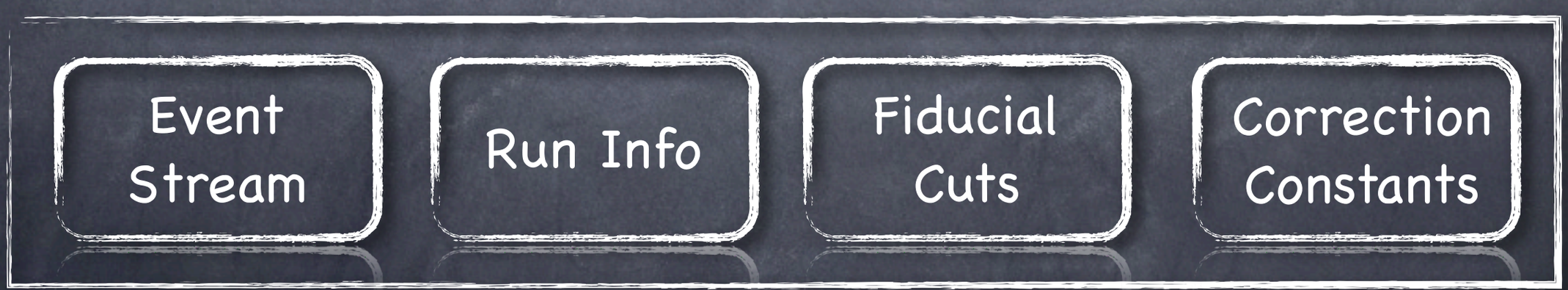
HDF5 Based DST format

- HDF5 is a hierarchy based file format widely used in scientific community.
- It comes as a standard installation for UNIX/OSX platforms.
- It can store in one file named data sets (maps), tables (dictionary) and variable length data streams.
- On top of HDF5 a library was developed to store data stream from CLAS, and a dictionary of the structures that are stored in the file.
- Named data sets keep run information such as beam energy, Torus current etc....

Advantages of HDF5

- Indexed structure of HDF5 provides ability to create data indices for specific events and there is no penalty for reading only selected events from the data stream.

- The run information is stored in the Data File. Downloaded sample of the file can be analyzed with the offline analysis code. No necessity to port databases to analyze a single file.



Data Structures

```
outFile = gooOutputStream('myEvents','simpleEventFile.hdf5')
outFile.defineBank('EVNT','pid,I16:px,F:py,F:pz,F')
for i in range(0,1000):
 outFile.writeBankRow('EVNT',0,{'pid': 211, 'px':0.1, 'py':0.2, 'pz':0.5})
```

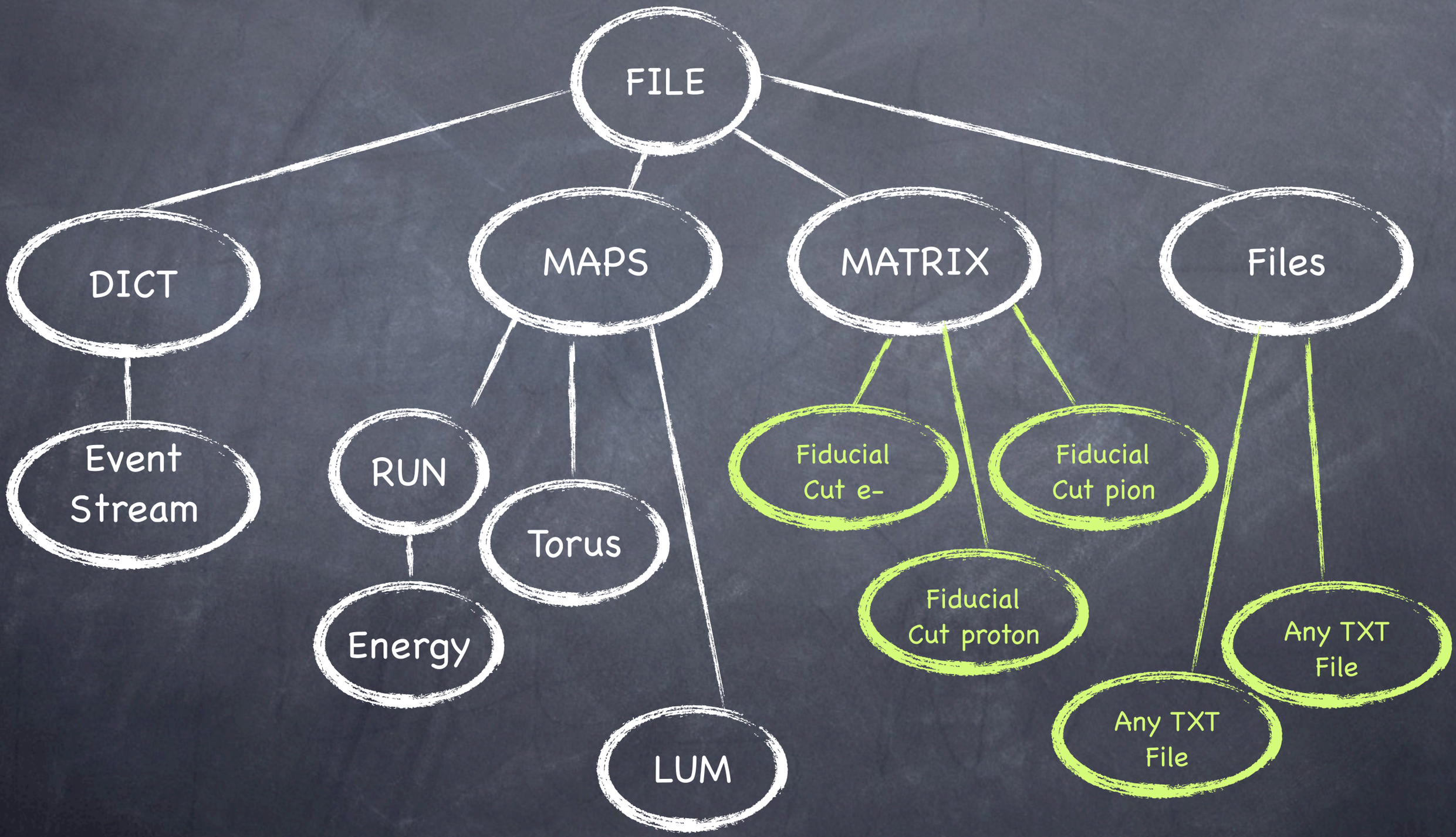
Field	Type
pid	INT
px	FLOAT
py	FLOAT
pz	FLOAT

```
outFile = gooOutputStream('myEvents','simpleEventFile.hdf5')
outFile.defineBank('EVNT','vx,F:py,F:vy,F:pz,F:vz,F:px,F:pid,I16')
for i in range(0,1000):
 outFile.writeBankRow('EVNT',0,{'pid': 211, 'px':0.1, 'py':0.2, 'pz':0.5})
```

Field	Type
vx	FLOAT
py	FLOAT
vy	FLOAT
pz	FLOAT
vz	FLOAT
px	FLOAT
pid	INT

```
inFile = gooInputStream('EventReader','simpleEventFile.hdf5',0)
while (1):
    evntBank = inFile.getBankRow('EVNT',k)
   print eventBank[‘px’], eventBank[‘py’],eventBank[‘pz’]
```

Data Structures



Analysis Framework

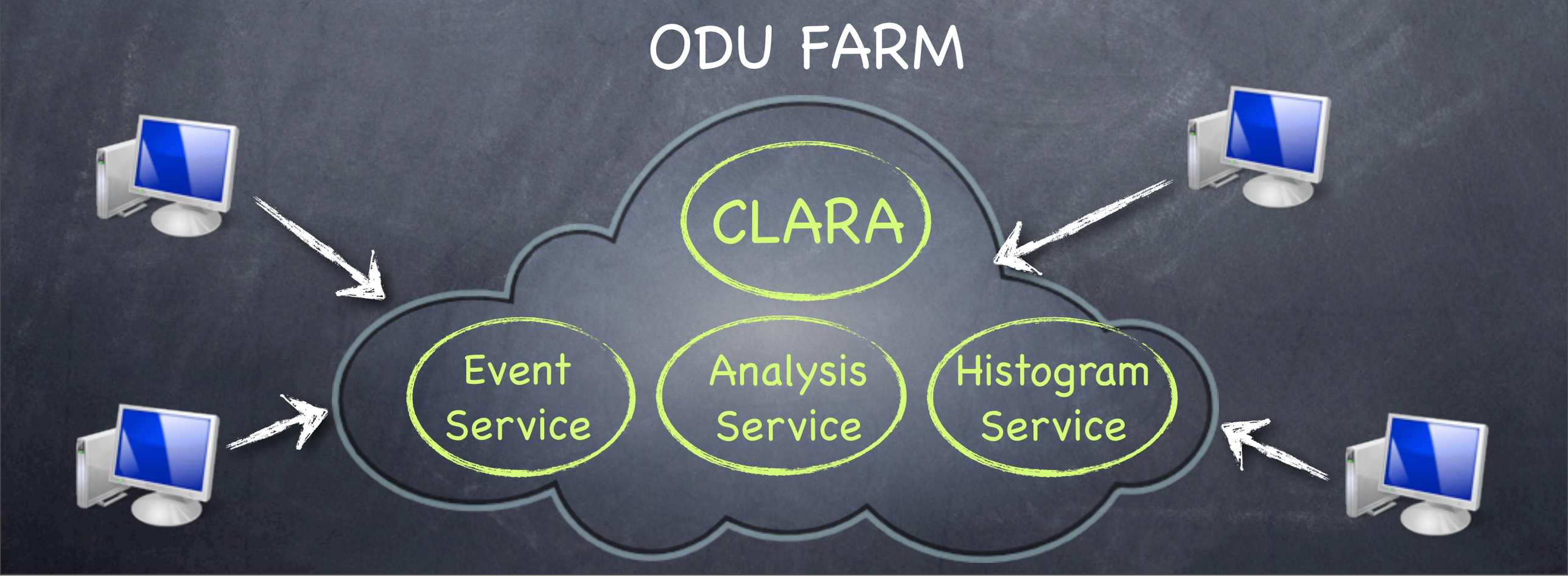
- Analysis framework works the same for all data sets with Correction, Fiducial Cuts and Event Constructor modules loaded automatically base on the run info loaded from the file.
- The data is indexed. Which means that the program will run only through events that correspond to selection criteria.
- The output can be controlled to be either of these:
  - Event Stream
  - Ntuples
  - Histograms
- Other analysis tools can be deployed to the server, requires a little advanced programming.

How does it work ? (python)

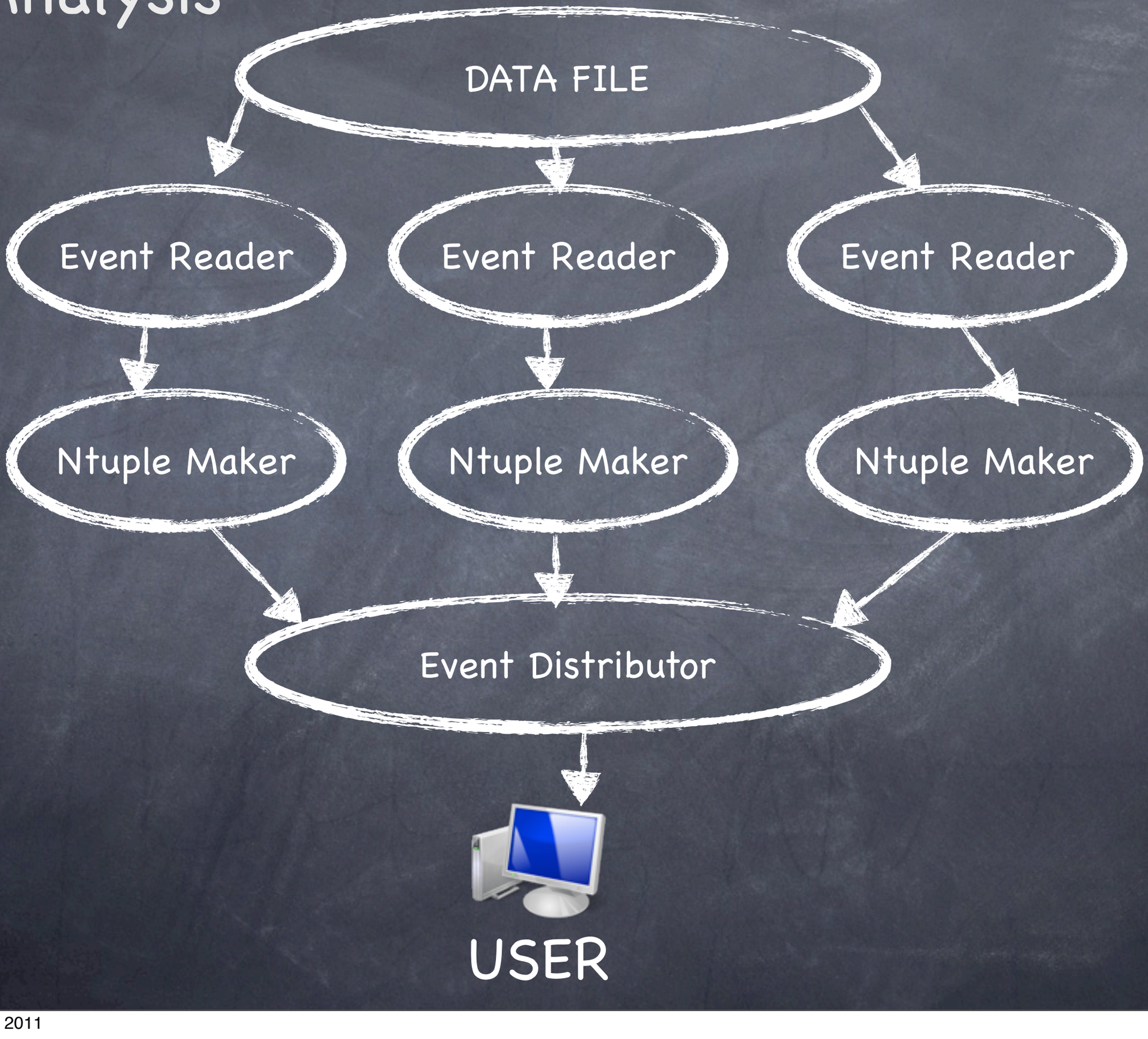
```
gooApp = gooPyApplication()
gooApp.setInput('e2:e1p1:c12')
//gooApp.setInput('file://myEventStream.hdf5')

gooApp.add('APP::set eventselection (11,2212,-211)')
gooApp.add('APP::set  fiducialcuts true')
gooApp.add('APP::set  momentumcorr  true')
//-------------------------
gooApp.add('NtupleMaker','NT1')
gooApp.conf('NT1:set  eventbank  EVNT')
gooApp.conf('NT1::set ntuplebank NTUP')
gooApp.conf('NT1::variable MxE   VECTMASS([b]+[t]-[11])')
gooApp.conf('NT1::variable MxEp VECTMASS([b]+[t]-[11]-[2212])')
//-------------------------
gooApp.add('HistogramMaker','HST1')
gooApp.conf('HST1::set ntuplebank NTUP')
gooApp.conf('HST1::cut  cMxE  CUT(MxE>2.&&MxE<4.)')
gooApp.conf('HST1::hist  H1MXEP  HIST(100,0.8,1.4,cMxE)')

//gooApp.runLocal() // Run analysis from local file

gooApp.runServer(20) // Run the analysis on the SERVER with 20 jobs
```

The Storage Scheme

- The Data will be stored at ODU (9T storage), with CLARA running as a front end to get the data.
- Several Services will be provided to get the data with some selection from the server, get analyzed data ntuples or histograms.
- A framework (GUI) will be provided to chose the data to analyze.

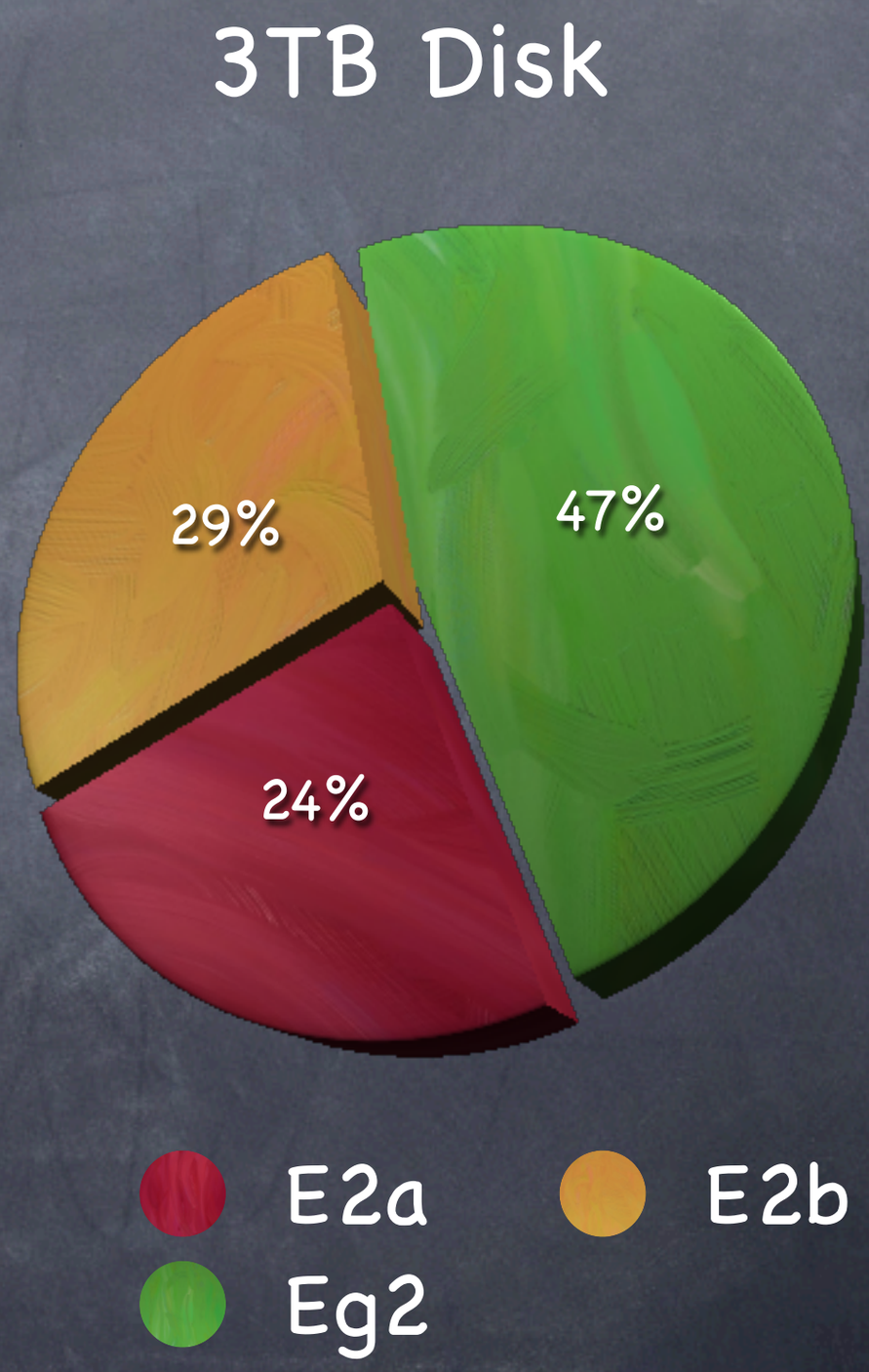


Analysis


DATA FILE
Event Reader
Event Reader
Event Reader
Ntuple Maker
Ntuple Maker
Ntuple Maker
Event Distributor
USER

Plans

- We plan to get data from 3 experiments within 6 month onto our disks.
- Set up our CLARA service for universal access from anywhere in the world (Mars access will come later).
- Index the data by event types, target, beam energy and torus current. So analysis can run through the data faster without reading the whole data set.
- Setup Fast-MC framework for all data sets.



Summary

- We will have multi-hadron data all stored at ODU site, organized in beam energy, target and torus field.
- Access will be provided to participating institutions.
- Analysis tools will be in place for FAST Monte-Carlo and Simulations.
- Basic convertors from HDF5-ROOT and ROOT-HDF5 are implemented. (May be HDF5-HBook in the future).
- The framework works in SOA mode as well as for the local analysis mode. User does not need map files databases, everything about the run is in the file.

The End

In Theory there is not difference between theory and practice.

In practice .... there is !