Using a DST as a big NTuple

Introduction about how to use GaudiPython together with ROOT to have a quick and simple access to data stored on a DST
Changes to previous versions

- **January 2008**
  - Try to make maximum advantage of latest changes in Configurables
  - Pack basic configuration in a python script, LHCbConfig.py. Makes example scripts cleaner

- **October 2008**
  - Moved example scripts to CVS: Vis/GaudiPythonTutorial, released with Panoramix.
  - After SetupProject Panoramix: python $GAUDIPYTHONTUTORIALROOT/python/Ex1.py

- **June 2008**
  - Update for Gaudi v19r9
  - Added examples: xx_multicore.py

- **March 2008**
  - Update for Gaudi v19r8, using of configurables
  - New place for example scripts: /afs/cern.ch/lhcb/group/panoramix/vol1/GaudiPythonTutorial/Gaudi_v19r8

- **September 2007:**
  - Tutorial for Software week
  - Reference to example scripts on afs at ~truf/public/GaudiPythonTutorial/

- **July 2007:**
  - Added slide about MCMuonInfo
  - Added list of more advance Python scripts
  - Add slides about event tag collection
  - Fix some typos
  - Add page for using iterators

- **June 2007:**
  - Replace rootSvc with aida2root
Useful links

- Python Home Page, Python Tutorial, Python Quick Reference
- GaudiPython Twiki

Tutorials & Presentations:
(Caution: many recent improvements, therefore presentations not anymore up-to-date in all areas)

- GaudiPython Tutorial (Pere Mato)
- Introduction to GaudiPython (Jose A. Hernando)
- Bender (Vanya Belyaev)
- Interactive analysis with Python, Examples (T.Ruf)
- PyRoot (Wim Lavrijsen)
Python

- Python is a scripting language.
- No compilation, ideal for interactive analysis, and fast prototyping.
- Python knows about C++ objects and methods via dictionaries. Full access to fast and debugged C++ algorithms and tools.
- Python allows to work with different applications at the same time: Gaudi + ROOT + OpenScientist +…, Python acts as glue
- You don’t have to read manuals or books before doing useful physics analysis!
Environment

Probably the most complicated part

- Software in LHCb is managed using CMT
- CMT sets up the environment for an application
  - Logical names, path to libraries, include files, …
- Developers would like to have environment as restricted as possible to avoid interferences
- End-users would like to have an environment with access to everything
- Best solution:
  - SetupProject Panoramix v16r3
  - 90% will also work with DaVinci or Brunel environment
Getting started

Three ways of running python:

- Execute a script:
  - `python ex0.py`
- Execute a script and return to python prompt:
  - `python -i ex0.py`
  - Exit python with ctrl-d (Linux) or ctrl-z (Windows)
- Entering commands by hand or by cut and paste:
  - `python`

Example scripts can be found at `$GAUDIPYTHONTUTORIALROOT/python/Ex1.py` or in CVS: `Vis/GaudiPythonTutorial`

For a full list of examples go [here](#)
Getting started, import, dir and help

- python
  >>> import GaudiPython

Everything inside GaudiPython will be in the namespace GaudiPython:
  >>> dir(GaudiPython)
  ['AppMgr', 'Bindings', 'CallbackStreamBuf', ...]
  >>> dir(GaudiPython.AppMgr())
  [ ... , 'addAlgorithm', 'algorithm', 'algorithms', 'config',
    'configure', 'createSvc', 'datasvc', 'declSvcType', 'detSvc',
    'detsvc', 'evtSel', 'evtSvc', 'evtsel', 'evtsvc', 'execute',
    'executeEvent', 'exit', 'finalize', ... ]

Also possible:
  >>> import GaudiPython as gaudi
  >>> dir(gaudi)
  ['AppMgr', 'Bindings', 'CallbackStreamBuf', ...]

dir(“something”) useful command to quickly see what you could do with “something”, or dir() to see what is around.

Another useful command: help(“something”)
With Gaudi v19r9, any LHCb application is configured and started with Python.

Essentially two steps
- Configuring
- Running

Configuring

```python
>>> from Gaudi.Configuration import *
>>> # load old option files
>>> importOptions('$STDOPTS/LHCbApplication.opts')
>>> importOptions('$STDOPTS/DstDicts.opts')
>>> # can also load python options
>>> importOptions('myStartup.py')
```

Running

```python
>>> appMgr = GaudiPython.AppMgr()
>>> appMgr.run(1)
```
Reading files: EventSelector and Transient Event Store

- If not specified in the option file, the input file can be specified using the eventselector:

  ```
  >>> sel = appMgr.evtsel()
  >>> sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
  # [] = list of files or one file
  ```

- Get event service for later use to access the transient event store

  ```
  >>> evt = appMgr.evtsvc()
  ```

- Read one event

  ```
  >>> appMgr.run(1)
  ```
NEW: LHCbConfig.py

- For better reading of the example scripts, the most basic configuration is moved to LHCbConfig.py
- It is enough to have in the user script
  ```python
  >>> from LHCbConfig import *
  >>> lhcbApp.DataType = 'DC06'
  or
  >>> lhcbApp.DataType = '2008'
  ```
- LHCbConfig.py sets up
  - reading of files and unpacking of containers,
  - starts the data on demand and particle property service
  - decoding of raw data banks
First Python script

```python
>>> from LHCbConfig import *
lhcbApp.DataType = 'DC06'

>>> # configure application manager
appConf = ApplicationMgr( Outputlevel = INFO )

>>> import GaudiPython

>>> # some additional features:
    import gaudigadgets

>>> # start application manager
    appMgr = GaudiPython.AppMgr()

>>> sel = appMgr.evtsel()

>>> sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])

>>> evt = appMgr.evtsvc()

>>> appMgr.run(1)

>>> evt.dump()
```

Could be executed as:

- **python Ex1.py**
  - After script completion, python stops. In this case, not very useful.

- **python --i Ex1.py**
  - After script completion, returns to python prompt. Useful for continuing interactive session.

- **python, followed by cut and paste from an editor**

- **In batch mode, ROOT needs --b argument otherwise will start graphics:**
  - python Ex1.py -b
Inspecting an event

Transient event store

```python
>>> evt.dump()
```

- Notice:
  - not all data are read, only requested.
  - Some data are packed, pSim, pRec, requires unpacking

```python
>>> evt['Rec/Vertex/Primary']
```

```python
>>> evt.dump()
```

```python
>>> evt['Rec/Vertex/Primary'][0]
```

```python
{ position : (-0.00195153, 0.0830339, -61.8848)
covMatrix : [[ 0.000513381, -4.41126e-05, 0.00189939, -4.41126e-05, 0.000114097, -0.000206983],
              [-0.00189939, 0.000114097, 0.00906074]]
chi2 : 0
nDoF : 25
extraInfo :
}
```

- dump all leaves: ```>>> evt.dumpAll()```
- forcing reading all below Rec:
  ```
  >>> allRec = gaudigadgets.nodes(evt, True,'Rec')
  ```
More details on Configuration
LHCbApp Configurable

```python
>>> from GaudiConf.Configuration import LHCbApp
>>> lhcbApp = LHCbApp()

Major difference to opts files: help is available

```{.python}

```python
>>> help(lhcbApp)
```

<table>
<thead>
<tr>
<th>Data descriptors defined here:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CondDBtag: Tag for CondDB. Default as set in DDDBConf for DataType</td>
</tr>
<tr>
<td>DDDBtag: Tag for DDDB. Default as set in DDDBConf for DataType</td>
</tr>
<tr>
<td>DataType: Data type, can be ['DC06','2008']. Default '2008'</td>
</tr>
<tr>
<td>EvtMax: Maximum number of events to process</td>
</tr>
<tr>
<td>Monitors: List of monitors to execute</td>
</tr>
<tr>
<td>Simulation: Flag to indicate usage of simulation conditions</td>
</tr>
<tr>
<td>SkipEvents: Number of events to skip</td>
</tr>
<tr>
<td>UseOracle: Flag to enable Oracle CondDB. Default False (use SQLDDDB)</td>
</tr>
</tbody>
</table>

DC06 MC setting:

```python
>>> lhcbApp.DataType = 'DC06'
>>> lhcbApp.Simulation = False  # SIMCONDB didn't exist yet
```

2009 MC setting:

```python
>>> lhcbApp.DataType = '2008'
>>> lhcbApp.Simulation = True
```
Database Tags, default tag

- If no tag is specified, the default tag = most recent tag is used
- No problem for DC06, since by construction all tags are compatible
- For 2008 MC, until mass production starts, recent tag might not be compatible with the tag used in the simulation
- For real data, if reconstruction runs from scratch, default tag is the preferred option. If only a partial re-reconstruction is running, tag should be consistent with previous tag used in the reconstruction.
- The tag should be provided by the Bookkeeping together with list of files. Not in place yet.
- The application versions and tags used are available for every event:

```python
>>> print evt['MC/Header'], evt['MC/DigiHeader'], evt['Rec/Header']
```
Tags, continued

```python
>>> evt['MC/Header']
    randomSeeds : [1, 2, 1501426441, 0]
    applicationName : Gauss, applicationVersion : v35r1
    condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]

>>> evt['MC/DigiHeader']
    applicationName : Boole, applicationVersion : v16r3
    condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]

>>> evt['Rec/Header']
    applicationName : Brunel, applicationVersion : v34r0
    condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]
```
And now to something completely different
Getting started: ROOT

```python
>>> from ROOT import *

- Everything from ROOT will be in the global namespace.
- Convenient, but not so efficient. Better:

```python
>>> from ROOT import TH1F, TBrowser, TCanvas

- Python ROOT and CINT ROOT commands are similar, python is simpler

```python
>>> h = TH1F('h_id', 'nr. of tracks', 100, 0., 500.)
>>> dir(h)
['=', 'AbstractMethod', 'Add', 'AddAt', 'AddBinContent', 'AddDirectory', ..., 'Fill', 'FillBuffer', 'FillN', 'FillRandom', 'FindBin', 'FindObject', 'Fit', 'FitPanel', ...

>>> h.Fill(10)
>>> h.Draw()
>>> b = TBrowser()
```

Command completion:

- `h.Dr` and `Tab`
Hbook inspired, filling by identifier:

```python
fh = gROOTFindObjectAny
hstore =[] # you are the owner
# booking
for n in range(21) :
    hstore.append(TH1F('sensor'+str(n),' ADC counts',100,0.,255.))
...
# filling
rc = fh('sensor'+str(n)).Fill(adc)
...
# saving
f=TFile('histos.root','recreate')
for h in hstore :
    h.Write()
f.Close()

# retrieving
f=TFile('histos.root')
fh = fFindObjectAny

# drawing
fh('sensor'+str(10)).Draw()
```
Access to doxygen

```python
>>> from gaudigadgets import doxygen
>>> h = TH1F('h', '...')
>>> doxygen(h)
```

```python
>>> cl = GaudiPython.gbl.LHCb.VeloCluster()
>>> doxygen(cl)
```

```python
>>> pv = evt['Rec/Vertex/Primary'][0]
>>> doxygen(pv)
```
Inspecting an event, cont.

```python
>>> h = TH1F('h',' nr. of tracks',100,0.,500.)

Containers

```python
>>> tc = evt['Rec/Track/Best']
>>> result = h.Fill(tc.size())
```

List of objects

```python
>>> for t in tc :
    print t

result = hp.Fill(t.p())
```

```python
>>> dir(tc[3])
[...
'addInfo', 'addToAncestors', 'addToLhcbIDs', 'addToMeasurements', 'addToNodes',
'addToStates', 'ancestors', 'charge', 'checkFitHistory', 'checkFitStatus', 'checkFlag',
'checkHistory', 'checkPatRecStatus', 'checkType', 'chi2', 'chi2PerDoF', ..., 'momentum',
'nDoF', 'nLHCBIDs', 'nMeasurements', 'nMeasurementsRemoved', 'nStates', 'nodes', 'p', ...,
'stateAt', 'states', 'type', 'typeBits', 'typeMask']
```
Inspecting an event, full example

```python
# root has to come first, otherwise set to batch mode by Gaudi
from ROOT import TH1F

# get the basic configuration from here
from LHCbConfig import *
lhcbApp.DataType = "DC06"
ApplicationMgr( OutputLevel = INFO )
import GaudiPython
appMgr = GaudiPython.AppMgr()
sel = appMgr.evtsel()
sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
evt = appMgr.evtsvc()
h = TH1F('h',' nr. of tracks',100,0.,500.)
hp = TH1F('hp',' momentum of tracks',100,0.,50000.)
for n in range(100) :
    appMgr.run(1)
    tc = evt['Rec/Track/Best']
    result = h.Fill(tc.size())
    for t in tc :
        result = hp.Fill(t.p())
    h.Draw()
    hp.draw()
```
Unpacking MC data

appConf = ApplicationMgr( OutputLevel = INFO, AppName = 'Ex3' )
appConf.TopAlg += ['UnpackMCParticle','UnpackMCVertex'])

- **ApplicationMgr** is the configurable of the Application manager.
- **What to configure:** help(appConf) or appConf.properties().keys()

More elegant: python –i Ex3b.py

```python
appConf.ExtSvc += ['DataOnDemandSvc']
DataOnDemandSvc().Algorithms +=
["DATA='/Event/MC/Particles' TYPE='UnpackMCParticle'"]
DataOnDemandSvc().Algorithms +=
["DATA='/Event/MC/Vertices' TYPE='UnpackMCVertex'"]
```

By default in LHCbConf.py
Unpacking Raw data

- Done by data on demand service.
- Configured in LHCbConf.py and $STDOPTS/DecodeRawEvent.py

```python
# get the basic configuration from here
from LHCbConfig import *
lhcbApp.DataType = "DC06"
appConf = ApplicationMgr( OutputLevel = INFO, AppName = 'Ex3c' )
import GaudiPython
appMgr = GaudiPython.AppMgr()
.sel = appMgr.evtsel()
.sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
evt = appMgr.evtsvc()
appMgr.run(1)
veloClusters = evt['Raw/Velo/Clusters']
print 'size of VeloCluster container:', veloClusters.size()
ItClusters = evt['Raw/IT/Clusters']
print 'size of ItCluster container:', ItClusters.size()
Muoncoords = evt['Raw/Muon/Coords']
print 'size of Muon container:', Muoncoords.size()
```
Containers, index, keys

```
MCParticles = evt['MC/Particles']
# first particle
mcp = MCParticles.containedObjects()[0]
# particle with key 0
mcp = MCParticles[0]

veloClusters = evt['Raw/Velo/Clusters']

(The key for VeloClusters is a VeloChannelID object, therefore direct access by “index”,
evt['Raw/Velo/Clusters'][0], does not work.)

aVeloCluster = veloClusters.containedObjects()[0]
doxygen(aVeloCluster)
```
Another example: `rewind()`

```python
from gaudigadgets import panorewind
hpt = TH1F('hpt','pt of bees',100,0.,10000.)
for n in range(100) :
    appMgr.run(1)
    mc = evt['MC/Particles']
    for mcp in mc :
        if mcp.particleID().hasBottom() :
            result = hpt.Fill( mcp.pt() )
    hpt.Draw()    # plot Root histogram
panorewind()    # start again from first event

hpt2 = TH1F('hpt2','pt of charm',100,0.,10000.)
for n in range(100) :
    appMgr.run(1)
    mc = evt['MC/Particles']
    for mcp in mc :
        if mcp.particleID().hasCharm() :
            result = hpt2.Fill( mcp.pt() )
    hpt2.SetLineColor(2)
    hpt2.Draw('same')
```

This worked in the past with `sel.rewind()`. After changes to Gaudi state machine, requires some tricks.
More object oriented approach:

```python
appConf.HistogramPersistency = "ROOT"
HistogramPersistencySvc.OutputFile = "Ex3e.root"
import GaudiPython
class MyAlg(GaudiPython.PyAlgorithm):
    def execute(self):
        evh = evt['Rec/Header']
        mcps = evt['MC/Particles']
        print 'event # = ', evh.evtNumber()
        h1.fill(mcps.size())
        return True

hist = appMgr.histsvc()
h1 = hist.book('h1', '# of MCParticles', 40, 0, 5000)
appMgr.addAlgorithm(MyAlg())
appMgr.run(10)
```

Books AIDA histogram

In case somebody really wants HBOOK output:

```python
appConf.HistogramPersistency = "HBOOK"
HistogramPersistencySvc.OutputFile = "Ex3e.hbook"
```
Displaying Aida histograms with Root

- Histogram is “inside” Gaudi defined by AIDA interface

```python
>>> hist.dump()
/stat
/stat/h1
>>> hist['h1']
Histogram 1D "# of MCParticles" 40 bins[0.000000,5000.000000]
```

- Get access to histogram

```python
>>> aida2root = GaudiPython.gbl.Gaudi_Utils.Aida2ROOT.aida2root
>>> rh1 = aida2root(hist['h1'])
>>> rh1.Draw()
>>> rh1.Fit('gaus')
```
Setting up environment:  

```
from LHCbConfig import *
lhcbApp.DataType = "DC06"
appConf = ApplicationMgr( OutputLevel = INFO, AppName = 'myJob' )
import GaudiPython
appMgr = GaudiPython.AppMgr()
sel = appMgr.evtsel()
sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
evt = appMgr.evtsvc()
appMgr.run(1)
```

Be able to read sim, digi, dst, rdst, mdf files.

Inspection, help, documentation:

```
aTrack = evt['Rec/Track/Best'][0]
dir(aTrack), help(aTrack), doxygen(aTrack)
```

Read whole event and dump leaves

```
evt.dumpAll()      # requires import gaudigadgets
```

Book, fill and draw histograms

```
from ROOT import *
hpt = TH1F('hpt','pt of bees',100,0.,10000.)
hpt.Fill(pt)
hpt.Draw()
```
Access to Geometry, Detector Elements, Conditions

python -i Ex4.py

```
from LHCbConfig import *

lhcbApp.DataType = "2008"
lhcbApp.Simulation = False  # default, uses LHCBCONDB

lhcbApp.Simulation = True  # uses SIMCONDB, didn’t existed for DC06

```
Useful abbreviation for classes:

- **XYZPoint** = GaudiPython.gbl.ROOT.Math.XYZPoint
- **XYZVector** = GaudiPython.gbl.ROOT.Math.XYZVector

Instantiation of the objects:

- **aPoint** = XYZPoint(0., 0., 10.)
- **aVector** = XYZVector(1., 1., 5.)
- print aVector.x(), aVector.y(), aVector.z()

You can also do in existing python session:

```python
from Ex5 import *
```

In case you changed the script later on:

```python
import sys
reload(sys.modules['Ex5'])
```
Access to Tools

Principle

- `tsvc = appMgr.toolsSvc()`
- `aTool = tsvc.create('name_of_tool', interface='Interface_of_tool')`

Ex6.py:

- `velopotool = tsvc.create('VeloClusterPosition', interface='IVeloClusterPosition')`
- `aCluster = evt['Raw/Velo/Clusters'].containedObjects()[0]`

(The key for VeloClusters is a VeloChannelID object, therefore direct access by “index”, `evt['Raw/Velo/Clusters'][0]`, does not work.)

```python
>>> clusInfo = velopotool.position(aCluster)
>>> clusInfo.fractionalError
0.17767
```
Access to Tools, cont.

Ex6b.py:

```python
extrap = appMgr.toolsvc().create('TrackMasterExtrapolator',
       interface='ITrackExtrapolator')
State = GaudiPython.gbl.LHCb.State
s_origin = State()
      setState(...)  
|    void State(const Gaudi::TrackVector& state)
|    void State(double x, double y, double z, double tx, double ty, double qOverP)

# create a state at origin with slow tx=0.1, ty=-0.1 and p = 5000 MeV/c
s_origin.setState(0.,0.,0.,0.1,-0.1,1./5000.)
s_extrap = s_origin.clone()

# extrapolate to z=12m
result = extrap.propagate(s_extrap,12000.)
print 'extrapolated state: x,y,z = %8.2f, %8.2f, %8.2f' %(s_extrap.x(),s_extrap.y(),s_extrap.z())
print 'extrapolated state: tx,ty,p = %8.2f, %8.2f, %8.2f' %(s_extrap.tx(),s_extrap.ty(),s_extrap.p())
```

>>> extrapolated state: x,y,z = 3048.075, -1213.064, 12000.000
>>> extrapolated state: tx,ty,p = 0.378, -0.100, 4971.968
Access to Services

Ex7.py:

- `appConf.ExtSvc += ['TransportSvc']`
- `appMgr = GaudiPython.AppMgr()`
- `appMgr.initialize()`
- `tranSvc = appMgr.service('TransportSvc','ITransportSvc')`
- `a = XYZPoint(0,0,0)`
- `b = XYZPoint(500.,350.,2000.)`
- `radlength = tranSvc.distanceInRadUnits(a,b)`

See also: python -i rad_map06.py and [http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Radiation_and_Bfield_maps.htm](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Radiation_and_Bfield_maps.htm)
Access to Services, cont.

Ex7b.py:

```python
# ParticlePropertySvc setup in LHCbConf.py
>>> appMgr.initialize()
>>> partSvc = appMgr.ppSvc()
>>> partSvc.find(ParticleID(521)).mass()
5279.0
>>> partSvc.find(ParticleID(521)).lifetime()
0.001671
>>> partSvc.find(ParticleID(521)).name()
'B+

>>> print partSvc.find(ParticleID(521))
B+ PDG: 521, Q: +1, mass: 5.2791 GeV, ctau: 491.06 um

† only available with import PartProp.decorators, done in LHCbConfig.py
```
Access to Services, cont.

Ex7c.py:

```python
appConf.ExtSvc += ['MagneticFieldSvc']
magSvc = appMgr.service('MagneticFieldSvc','IMagneticFieldSvc')
a = XYZPoint(0.,0.,5000.)
v = XYZVector()
magSvc.fieldVector(a,v)
>>> v.y()
-0.00103710293

bintegrator =
    appMgr.toolsvc().create('BIntegrator',interface='IBIntegrator')
Bdl = XYZVector()
def Double(d) : return d + 0.0 # this is a little trick
zCenter = Double(0.)
b = XYZPoint(0.,0.,2500.)
result = bintegrator.calculateBdlAndCenter(a,b,0.,0.,zCenter,Bdl)
>>> Bdl.x()
-0.0020624343736251353
```

Thomas Ruf
DST as Big NTuple
from LinkerInstances.eventassoc import *

MCParticle = GaudiPython.gbl.LHCb.MCParticle
Track = GaudiPython.gbl.LHCb.Track

... Following for every event

### MC relation

ltrack2part = linkedTo(MCParticle, Track, 'Rec/Track/Best')
lpart2track = linkedFrom(Track, MCParticle, 'Rec/Track/Best')
t = evt['Rec/Track/Best'][0]

for mcp in ltrack2part.range(t):
    print mcp
        { momentum : (923.68, -113.08, 3783.46, 3898.72)
          particleID : { pid : 211

for tt in lpart2track.range(mcp):
    print tt.key()
        0
        15 # MCParticle is reconstructed twice
Examples:

```python
from gaudigadgets import getEnumNames

Enums = getEnumNames('LHCb::RawBank')

BankType = Enums['BankType']

f = open('raw_buffer.txt', 'w')

f.write(' banktype sourceID word hex string \

i = 0

for k in BankType.keys():
    b = BankType[k]
    nTell1 = rb.banks(k).size()
    for m in range(nTell1):
        size = int((rb.banks(k)[m].size() / 4) + 0.5)
        for l in range(size):
            word = rb.banks(k)[m].data()[l]
            t2 = (word >> 16) & 0x0000ffff
            t1 = word & 0x0000ffff
            if word < 0:
                word = t2 * 2**16 + t1
            f.write('%15s %6d  %.8x %10s \
            %d
            i+=1
            f.close()

webbrowser.open(os.getcwd() + '/raw_buffer.txt')
```

Quite useful for studying data sizes of L1 boards, see also [Raw Data sizes](#)
Reading a MDF file

Calorimeter example:

```python
from LHCbConfig import *
lhcbApp.DataType = "2008"

appConf = ApplicationMgr( OutputLevel = INFO )
EventSelector(Input = ["DATA= 'castor:/castor/cern.ch/grid/lhcb/data/2008/RAW/LHCb/PHYSICS/24080/024080_0000054568.raw' SVC='LHCb::MDFSelector'" ] )

import GaudiPython
appMgr = GaudiPython.AppMgr()
evt = appMgr.evtsvc()
evt = appMgr.run(1)
print '+++++++', evt['Raw/Hcal/Digits'].size()
```

Run PatSeeding:

```bash
python -i Ex10.py
```
```python
class myAlg(GaudiPython.PyAlgorithm):
    def execute(self):
        vc = evt['Raw/Velo/Clusters']
        for cl in vc:
            sensorNr = cl.channelID().sensor()
            h = hlist[sensorNr]
            hl = hlandau[sensorNr]
            for i in range(cl.size()):
                success = h.Fill(cl.strip(i))
                success = hl.Fill(cl.totalCharge())
        return True
```

See also [http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python_for_cyclists.htm](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python_for_cyclists.htm)
Origin of secondaries, see http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Plotsofsecondaries.htm

```python
hlist = {}

hlist['MC/IT/Hits'] = TH2F('horigin_IT', 'r vs z origin of MCHits IT', 1000, -100., 13000., 100, 0., 2000.)

hlist['MC/TT/Hits'] = TH2F('horigin_TT', 'r vs z origin of MCHits TT', 500, -100., 4000., 100, 0., 2000.)

...  

for h in hlist:
    mch = evt[h]
    for ahit in mch:
        pos = ahit.mcParticle().originVertex().position()
        r = pos.rho()
        z = pos.z()
        if r>3.:
            result = hlist[h].Fill(z,r)
```

That is all.
The rest is ROOT presentation.
B Candidates

Ex11.py:

```python
from LHCbConfig import *
lhcbApp.DataType = "2008"
lhcbApp.Simulation = True
importOptions('${PANORAMIXROOT}/options/Panoramix_DaVinci.py')
importOptions('${CCBARROOT}/options/DoDC06SelBs2Jpsi2MuMuPhi2KK_lifetime_unbiased.opts')
appConf = ApplicationMgr( OutputLevel = INFO,AppName = 'Ex11' )
...

sel.open(['PFN:castor:/castor/cern.ch/user/t/truf/MC2008/00003402_121314_s.dst'])
...
h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5500.)
while 0 < 1:
    appMgr.run(1)
    # check if there are still valid events
    if evt['Rec/Header'] == None : break
    cont = evt['Phys/DC06selBs2Jpsi2MuMu_Phi2KK/Particles']
    if cont != None :
        for b in cont :
            success = h_bmass.Fill(b.momentum().mass())
    h_bmass.Draw()
```

All you need for a mass plot

Starting again: >>> gaudigadgets.panorewind()
Ex17b.py:

```python
appConf.ExtSvc += ['TagCollectionSvc/EvtTupleSvc']
import GaudiKernel.SystemOfUnits as units
file = 'PFN:castor:/castor/cern.ch/user/l/lshchuts/stripping/SETC_newDST_v31/1.root'
EventSelector( Input = ['COLLECTION="TagCreator/1\" DATAFILE=%s TYP='POOL_ROOT' SEL='(PreselHeavyDimuon>0)'''%file] )
FileCatalog().Catalogs = ['xmlcatalog_file:newDST_v31.xml']

h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5500.)
def my_loop():
    while 0<1 :
        appMgr.run(1)
        if evt['Rec/Header'] == None : break
        mybees = evt['Phys/Phys/PreselHeavyDimuon/Particles']
        if mybees != None :
            for b in mybees :
                rc=h_bmass.Fill(b.measuredMass()/units.GeV)
        my_loop()
    h_bmass.Draw()
```
Creating a microDST

**Basic configuration of the script:**
- `selection_name = 'DC06selBs2JpsiPhi_unbiased'`
- `saveMC = True`
- `sequence = 'Seq'+selection_name`

**Configure microDST algorithms:**
- `from Configurables import CopyRecHeader, CopyMCParticles, CopyParticles, CopyPrimaryVertices, CopyParticle2PVLink, CopyRelatedMCParticles, CopyFlavourTag, CopyODIN, ParticleCloner, MCParticleCloner`
- `copyParticles = CopyParticles()`
- `copyParticles.InputLocation = "Phys/"+selection_name+"/Particles`

# Copy related MCParticles and also associations between
- `copyMC = CopyRelatedMCParticles()`
- `copyMC.InputLocation = copyParticles.InputLocation`
- `copyMC.ICloneMCParticle = "MCParticleCloner"`
- `copyMC.addTool(MCParticleCloner,name="MCParticleCloner")`
- `copyMC.MCParticleCloner.ICloneMCVertex = "MCVertexCloner"`

# add to sequence
- `seq = GaudiSequencer(sequence)`
- `seq.Members += [copyParticles,copyMC,CopyPrimaryVertices(),CopyRecHeader(),CopyODIN()]`

**Define outputstream**
- `microDSTStream = OutputStream('MicroDSTStream')`
- `microDSTStream.ItemList = ["/Event/microDST#99"]`
- `fname = "DATAFILE='PFN:"+output_name+"' TYP='POOL_ROOTTREE' OPT='REC'"
- `microDSTStream.Output = fname`
- `ApplicationMgr().OutStream.append(microDSTStream)" python microDSTEx0.py"
Creating a microDST, cont.

Execution:

- # keep control of event loop
- appMgr.algorithm('MicroDSTStream').Enable = False
- Nevents = 0
- while 1>0 :
  - appMgr.run(1)
  - if evt['DAQ/ODIN']==None : break
  - if evt['Phys/DC06selBs2JpsiPhi_unbiased/Particles'] != None :
    - if evt['Phys/DC06selBs2JpsiPhi_unbiased/Particles'].size() > 0 :
      - # copy selected event to output file
      - appMgr.algorithm('MicroDSTStream').execute()
  - Nevents += 1
- print "Number of selected events: ", Nevents
Required running of microDSTEx0.py first

**microDSTEx1.py:**

```python
from LHCbConfig import *
lhcbApp.DataType = "DC06"
appConf = ApplicationMgr( OutputLevel = INFO )
EventSelector().PrintFreq = 10000
import GaudiPython
import GaudiKernel.SystemOfUnits as units
locationRoot = '/Event/microDST/'
h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5500.)
def my_loop():
    while 0<1 :
        appMgr.run(1)
        if evt[locationRoot+'Rec/Header'] == None : break
        mybees = evt[locationRoot+'Phys/DC06selBs2JpsiPhi_unbiased/Particles']
        if mybees != None :
            for b in mybees :
                rc=h_bmass.Fill(b.measuredMass()/units.GeV)
import time
print time.clock()
my_loop()
p = time.clock()
```

All you need for a mass plot

16k events, 6-8 sec on lxplus
from ROOT import MakeNullPointer
part = MakeNullPointer(GaudiPython.gbl.LHCb.MCParticle)
MCDecayFinder = appMgr.toolsSvc().create('MCDecayFinder', interface='IMCDecayFinder')
MCDecayFinder.setDecay('[B0,B+,B_s0]cc')
MCDebugTool = appMgr.toolsSvc().create('PrintMCDecayTreeTool', interface='IPrintMCDecayTreeTool')
decaylist = []
mc = evt['MC/Particles']
MCDecayFinder.hasDecay(mc)
while MCDecayFinder.findDecay(mc, part) > 0:
    print 'append', part.key()
    print part
decaylist.append(part.clone())
maxDepth = 2
for decay in decaylist:
    MCDebugTool.printTree(decay, maxDepth)

daughters = GaudiPython.gbl.std.vector('LHCb::MCParticle*')()
for decay in decaylist:
    MCDecayFinder.descendants(decay, daughters)
Copying Events

python -i Ex13.py

Write a few events to a separate file

```python
importOptions(...
InputCopyStream().Output = "DATAFILE='PFN:myEvents.dst' TYP='POOL_ROOTTREE' OPT='REC'"
appConf = ApplicationMgr(
    OutputLevel = INFO, AppName = 'Ex13', OutStream = [InputCopyStream()]
)
import GaudiPython
appMgr = GaudiPython.AppMgr()
sel = appMgr.evtsel()
sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/voll/Bsjpsiphi_00001620_00000004_5.dst'])
evt = appMgr.evtsvc()
appMgr.algorithm('InputCopyStream').Enable = False

while 1>0:
    appMgr.run(1)
    if evt['Rec/Header'] == None: break
    cont = evt['Phys/DC06selBs2Jpsi2MuMu_Phi2KK/Particles']
    if cont != None:
        out = False
        for b in cont:
            if b.pt() > 10000.: out = True
        if out: appMgr.algorithm('InputCopyStream').execute()
```
Yes, Raw data only

```python
... 
importOptions('$LODUOPTS/LODUFromRaw.opts')
from Configurables import EventNodeKiller
rawwriter = OutputStream('RawWriter', Preload = False,
    ItemList = ['/Event#1','/Event/DAQ#1','/Event/DAQ/RawEvent#1','/Event/DAQ/ODIN#1'],
    Output = 'DATAFILE="PFN:L0yes.raw" TYP=POOL_ROOTTREE OPT=REC ')
appConf = ApplicationMgr( OutputLevel = INFO, AppName = 'Ex13b')
appConf.TopAlg = ['EventNodeKiller/EventNodeKiller','GaudiSequencer/L0FromRaw']
appConf.OutStream = [rawwriter]
EventNodeKiller().Nodes =
EventSelector().PrintFreq = 50

import GaudiPython
appMgr = GaudiPython.AppMgr()
sel = appMgr.evtsel()
sel.open(['PFN:D:/LHCb_data/Bsjpsiphi_00001620_00000004_5.dst'])
evt = appMgr.evtsvc()
appMgr.algorithm('RawWriter').Enable = False  # stop automatic execution of RawWriter
```
while 1>0 :
  appMgr.run(1)
  # check L0
  L0dir = evt['Trig/L0/L0DU']
  if L0dir == None : break  # probably end of input
  if L0dir.decision() > 0 :
    rc = appMgr.algorithm('RawWriter').execute() # output event

Verify that file makes sense:
python –i Ex13c.py

import gaudigadgets
sel.open(['PFN:L0yes.raw'])
evt = appMgr.evtsvc()
appMgr.run(1)
evt.dumpAll()
HLT Example

```python
...
# Default options to rerun L0 and run Hlt
importOptions ('$L0DROOT/options/ReplaceL0BanksWithEmulated.opts')
from HltConf.Configuration import *
hltconf = HltConf()
hltconf.oldStyle = False
hltconf.hltType = 'PA+LU+VE+MU+HA+PH+EL'
....
appMgr = GaudiPython.AppMgr(o)
.sel = appMgr.evtsel()
.sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
evt = appMgr.evtsvc()
hist = appMgr.histSvc()
appMgr.run(1)

import hltexamples
# nice summary printouts per event
hltexamples.10()
hltexamples.10_values()
hltexamples.10_candidates()
hltexamples.hlt_decisions()
hltexamples.hlt_selection_report()
```
**HLT Example, cont.**

```python
import os

appMgr.run(200)


aida2root(hist['HltCaloTrigger/1']).Draw()

aida2root(hist['HltHadRecoIPVelo/3']).Draw()
```

**From gaudigadgets**

```python
def histo_dump(hsvc, node=None):
    if not node:
        root = hsvc.retrieveObject('')
        if root:
            node = root.registry()
            return
        else:
            return
    if node.object() :
        for l in hsvc.leaves(node):
            histo_dump(hsvc, l)
    else:
        print(node.identifier(), 'Title:', hsvc[node.identifier()].title())
```

```python
/hstat/Hlt/1 Title: time
/hstat/Hlt/2 Title: time0
/hstat/Hlt/3 Title: rate
/hstat/HltAllAlleys/1 Title: t
/hstat/HltAllAlleys/2 Title: t
/hstat/HltAllAlleys/3 Title: r
/hstat/HltMuAlleys/1 Title: t
/hstat/HltMuAlleys/2 Title: t
/hstat/HltMuAlleys/3 Title: r
/hstat/RealMuAlleys/1 Title:
/hstat/RealMuAlleys/2 Title:
/hstat/RealMuAlleys/3 Title:
/hstat/HltMuConfLOSingleAll
/hstat/HltMuConfLOSingleAll
/hstat/HltMuConfLOSingleAll
/hstat/HltHadAlley/1 Title: time
/hstat/HltHadAlley/2 Title: time0
/hstat/HltHadAlley/3 Title: rate
/hstat/HltHadAlley/4 Title: L0DU
/hstat/HltHadAlley/5 Title: L0DDU
```

```python
python -i Ex14.py
```
Event Display

python –i Ex15.py

...  
importOptions('$PANORAMIXROOT/options/PanoramixVis.py')

...

>>> from panoramixmodule import *

>>> toui()
http://pyprocessing.berlios.de/

- processing is a package which supports the spawning of processes using the API of the standard library’s threading module. It runs on both Unix and Windows.

See also presentation: Multiprocessors and GaudiPython by Pere Mato
http://indico.cern.ch/conferenceDisplay.py?confId=25001

Future version of GaudiPython should contain a "use processing v* LCG_Interfaces“, until then use:

- setenv PYTHONPATH /afs/cern.ch/sw/lcg/external/processing/0.51/${CMTCONFIG}/lib/python2.5/site-packages:${PYTHONPATH}
- Or: source multicore_setup

Also useful, pickleRoot.py: allows to return root objects
How to use it

- **GaudiPython configuration step**
  - `import GaudiPython`
  - `from pickleROOT import *`
  - `from processing import import Pool`

- **Encapsulate execution step in a function**
  - `def processFile(file) :`
    - `appMgr = GaudiPython.AppMgr()`
    - ...
    - `return something (should be pickleable)`
  - `#-----Use up to 8 processes`
  - `pool = Pool(8)`

Files = list of input files
- `result = pool.map_async(processFile, files)`
- `for r in result.get(timeout=10000) : print r`
Ex2b_multicore.py

- Loop over 16 input files, run DaVinci selection, fill histogram with B mass
- For each input file, histograms are returned and added up in the main program

On lxbuild, 8 core machine, real time:

- Single process: 258.7 sec | 264.0 sec
- 2 processes: 116.1 sec | 147.2 sec
- 4 processes: 95.5 sec | 92.4 sec
- 8 processes: 94.9 sec | Configuration dominated!!
- 8 processes (32 files): 115 sec | 4 processes: 165.1 sec
Ex3_multicore.py

- Loop over n input files, run full reconstruction, fill histogram with track fit chi2
- On lxbuild, 8 core machine, real time
  - Single process: 505 sec, 1 file or ~1 s / event
  - 8 processes: 983 sec, 8 files or ~0.25 s / event
Useful Modules

- **GaudiPython**: access to Gaudi and LHCb software
- **ROOT**: access to PyROOT
- **os, sys**: access to operating system
- **math**: basic math from Python
- **array**: like Python lists, restricted to basic values: characters, integers, floats
- **time**: access to system clocks
- **LinkerInstances.eventassoc**: access to LHCb linker classes
- **gaudigadgets**: some additional features
- **panoramixmodule**: access to LHCb Event Display
For more sophisticated analysis, kinematics, combinatorics, etc., look at Bender package == mirror of LoKi in Python

See other Tutorial
Current Performance Figures
Pentium 4 3GHz Windows XP

Fill size of container
- Rec/Track/Best
  - 57 kB and 14 ms / event
- Raw/Velo/Cluster
  - 34 kB and 21 ms / event
- MC/Particles
  - 89.4 kB and 45 ms / event

I/O, same for running C++
~ 4 MB / s
~ 2 MB / s, created from Raw
~ 2 MB / s, additional unpacking

Loop over container, fill one variable
- Rec/Track/Best
  - 115 objects and 4.5 ms / event
- Raw/Velo/Cluster
  - 1243 objects and 28 ms / event
- MC/Particles
  - 2428 objects and 83 ms / event

~30 μs / object

Simple N-tuple
- MCParticles: mom, oVx, eVX, pid
  - 2.5 kB, 2428 objects, 0.5ms / event

Thomas Ruf
DST as Big NTuple
Current Performance Figures

LXPLUS

- Fill size of container
  - Rec/Track/Best
    - (57 kB) and 4.9 ms / event
  - Raw/Velo/Cluster
    - (34 kB) and 3.3 ms / event
  - MC/Particles
    - (89 kB) and 12.7 ms / event

- Loop over container, fill one variable
  - Rec/Track/Best
    - 116 objects and 1 ms / event
  - Raw/Velo/Cluster
    - 1287 objects and 14 ms / event
  - MC/Particles
    - 2542 objects and 34 ms / event

- Simple N-tuple
  - MCParticles: mom, oVx, eVX, pid
    - 2.5 kB, 2542 objects, 0.4ms / event

I/O, same for running C++

~ 12 MB / s

~ 10 MB / s, created from Raw

~ 7 MB / s, additional unpacking

~10 μs / object
Current Performance Figures for a microDST

**LXPLUS**

- **Fill size of container**
  - Phys/…/Particles
    - (0.5 kB) and 0.36 ms / event
  - MC/Particles
    - (0.6 kB) and 0.33 ms / event
  - Rec/Track/Best
    - (4.6 kB) and 0.67 ms / event

- **Loop over container, fill histogram**
  - Phys/…/Particles
    - 1 object and 0.01 ms / event
    - Filling 4 histograms with 4 variables, no difference
  - MC/Particles
    - 9.7 objects and 0.13 ms / event
  - Rec/Track/Best
    - 6 objects and 0.03 ms / event

- **Example: Bs2JpsiPhi μDST**
  - 8.8 kB/event ⇒ $10^6$ events = 8.8 GB
  - one loop over $10^6$ events: ~7 minutes
  - MC part: 1.1 kB/event, Tracks: 4.6 kB/event
List of more advanced example scripts

1. **Ks reconstruction study**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/KsShorts/)

2. **Checking Measurements**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/checkMeasurements.py)

3. **Geant4 multiple scattering**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/checkGeant4_MS.py)

4. **Make IP and P resolution plots**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/IPandPresol_plot.py)

5. **Check Velo Position Tool**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/checkVeloPoTool.py)

6. **Get events on the grid via Event and Run number**
   
   [Link](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Grid.htm)

7. **Other inspirations at $PANORAMIXROOT/Examples/Python**

---

Thomas Ruf

DST as Big NTuple
7. **Fake B reconstruction**
   Reconstructs all b-particles using links to MCTruth. Provides mass resolution plots

   python –i fakeBreco.py

8. **Fake J/ψ reconstruction, as above**

9. **Hlt Velo Position A and C side**

10. **Make ROOT histograms and convert to HBOOK**

11. **Execute a list of sample scripts and search for new warnings and errors in the log files**

   python runall.py > log.txt
Other stuff, random collection
Properties of Algorithms

If you want to know which Algorithms are running with which parameters:

>>> appMgr.algorithms()
['ProtoPRecalibration', 'MakeSomeResonances', 'SeqDC06selBs2Jpsi2MuMu_Phi2KK', 'UnpackMCParticle', 'UnpackMCVertex', 'ChargedProtoCombineDLLsAlg', ... , 'DC06selBs2Jpsi2MuMu_Phi2KK']

>>> anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK')

>>> for x in anAlg.properties():
...    print x,getattr(anAlg,x)

EvtColDir Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK
... MotherFilterName MotherFilter ...

anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK.MotherFilter')

>>> for x in anAlg.properties():
...    print x,getattr(anAlg,x)

Selections ['B_s0 : VtxFilterCriterion/bsVertex']

anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK.MotherFilter.bsVertex')

>>> for x in anAlg.properties():
...    print x,getattr(anAlg,x)

MaxChi2 22.5

Alternative: Inspect the configurable of an algorithm
Alternative Way of managing your ROOT histos

- **Booking:**
  - `hstore = {}`
  - For every histogram, function:
    - `hstore([aUniqueKey]) = TH1F(aUniqueKey,'title',100,1.,0.)`

- **Filling:**
  - `hstore([aUniqueKey]).Fill(x)`

- **Retrieval from ROOT file:**
  - `from ROOT import gROOT, TFile, TROOT`
  - `def TROOT_getitem(self,item) :`
    - `return self.FindObject(item)`
  - `TROOT.__getitem__ = TROOT_getitem`
  - `hstore = gROOT._gROOT`
  - `file = TFile(....)`
  - `hstore[aUniqueKey].Draw()`
Python printout formatting

- `a=3.1415; b='text'
- `print 'a=',a, 'b=','b` : `a= 3.1415 b= text`
- `print 'a=%6.2f b=%s'%(a,b)` : `a= 3.14 b=text`
- `print 'a=%(X)03d'%(X:7)` : `a=007`
Define a helper class:

class irange(object) :
    def __init__(self, b, e):
        self.begin, self.end = b, e
    def __iter__(self):
        it = self.begin
        while it != self.end:
            yield it.__deref__() 
            it.__postinc__(1)

Then:

from gaudigadgets import irange as irange
GenEvtContainer = evt['Gen/HepMCEvents']
for genEvt in GenEvtContainer:
    pGenEvt = genEvt.pGenEvt()
    for particle in irange(pGenEvt.particles_begin(),pGenEvt.particles_end()):
        if particle.production_vertex() :
            vtx = particle.production_vertex()
            print "PDG ID %i production vertex z=%4.2F part out: %3i"
            %(particle.pdg_id()), vtx.position().z(),vtx.particles_out_size())
        else:
            print "PDG ID %i" %(particle.pdg_id())
Event Tag Collections

**Details about Event Tag Collections can be found [here](https://twiki.cern.ch/twiki/bin/view/LHCb/DC06StrippingHowTo):**

```python
etc = GaudiPython.AppMgr().evtcolsvc()
appMgr.initialize()
extc.dump()
```

```python
tupl=etc['EventSelector.DataStreamTool_1/TagCreator/1']
<ROOT.NTuple::ColumnWiseTuple object at 0x3efb180>
```

```python
dir(tupl) does not yet yield any useful methods, DEAD END. Wait for feedback from experts. Meanwhile use ROOT.
```

```python
tupl = gROOT.FindObjectAny('(<local>_TagCreator_1')
<ROOT.TTree object ('<local>_TagCreator_1") at 0x3e97600>
```
```python
lvs = tupl.GetListOfLeaves()
for n in lvs:
    print n.GetName()
```

Address
token
event
run
entry
nPrim
nChargedProto
nNeutralProto
PreselB2DiMuon
PreselBd2KstarMuMu
PreselBd2Kstaree
PreselBs2MuMu
PreselBu2LLK
PreselBs2PhiEtac
PreselHeavyDimuon
PreselJpsi2ee
PreselUnbiasedJpsi2ee
...

Thomas Ruf
DST as Big NTuple
Muon MCInfo

```python
DataOnDemandSvc().AlgMap['/Event/Link/Raw/Muon/Coords'] = 'MuonCoord2MCParticleLink'
...
MCMuonDigitInfo = GaudiPython.gbl.LHCb.MCMuonDigitInfo
MuonDigit = GaudiPython.gbl.LHCb. MuonDigit
MCParticle = GaudiPython.gbl.LHCb.MCParticle
MuonCoord = GaudiPython.gbl.LHCb.MuonCoord
...
appMgr.run(1)
lcoord2part = linkedTo(MCParticle, MuonCoord, 'Raw/Muon/Coords')
l_mcpart = {}
muonCoords = evt['Raw/Muon/Coords']
# take first muonCoord as an example
obj = muonCoords.containedObjects()[0]
# find all mcparticles linked with this object
l_mcpart[obj]=[]
for mcp in lcoord2part.range(obj) :
    l_mcpart[obj].append(mcp)
# print info about muon tile and its related MCParticles
print l_mcpart
```

Thomas Ruf
DST as Big NTuple
def muondigithistory(mc):
    if mc.isGeantHit(): print 'IsGeant'
    if mc.isXTalkHit(): print 'IsXtalk'
    if mc.isBackgroundHit(): print 'IsBackg'
    if mc.isChamberNoiseHit(): print 'IsChamberNoise'
    if mc.isElNoiseHit(): print 'ElNoiseHit'
    if mc.isFlatSpilloverHit(): print 'FlatSpillover'
    if mc.isMachineBkgHit(): print 'MachineBkg'

    # Vector of digit
    digits = obj.digitTile()
    MCInfo = evt['MC/Muon/DigitsInfo']
    mci = MCMuonDigitInfo()
    for d in digits:
        info = MCInfo.link(d.key())
        mci.setDigitInfo(info)
    # print more info about MC history
    muondigithistory(mci)
For Windows Users

- Pythonwin: Editor, debugger, command completion

```python
>>> import gaudi
>>> appMgr = gaudi.AppMgr(outputLevel=0)
>>> appMgr
```

This defines Algorithm for event loop

```python
class myAlg(gaudi.module.PyAlgorithm):
    def execute(self):
        vc = evt['Raw/Velo/Clusters']
        for cl in vc:
            sensorNr = cl.channelID().sensor()
            if hlist.has_key(sensorNr):
                h = hlist[sensorNr]
                hl = hlandau[sensorNr]
            else:
                sensor = velo.sensor(sensorNr)
                title = sensor.name().replace(velo_loc,'')
                name = title.split('/')[2]
                h = TH1F(name, title, sensor.numberOfStrips(), 0, sensor.numberOfStrips)
                hlist[sensorNr] = h
```

Interactive Window

PythonWin 2.4.2 (#67, Sep 28 2005, 12:41:11) [MSC v.1310 32 bit (Intel) on win32]
Copyright 1994-2004 Mark Hammond (mhammond@skippinet.com.au) - see 'Help/About PythonWin'
Copyright Information.

```python
>>> import gaudi
>>> appMgr = gaudi.AppMgr(outputLevel=0)
```

addAlgorithm
algorithm
algorithms
config
configure
Gaudi Viewer (Radu Stoica)

Presentation [Link](http://indico.cern.ch/getFile.py/access?contribId=10&resId=1&materialId=slides&confId=3130)

```python
gaudipy -d -f
$AFSROOT/cern.ch/lhcb/group/tracking/vol1/0001378_00000002_5.dst
```

Gaudi Viewer

Not anymore maintained
Gaudi Viewer
Zsolt Lazar, Ronan McNulty

- Ideas presented at Core Software meeting, 28 March 2007
- To facilitate start-up of newcomers
Panoramix

Three steps

1. Select container in Event Tree
2. Start Data Browser under Event Menu
   • Select doxygen class information
   • Select how many objects to dump
     • Information dumped depends on the implementation of the fillstream method by the class author
## Full list of Examples

<table>
<thead>
<tr>
<th>Script</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessBookkeeping.py</td>
<td>access to production info, requiring prodid</td>
</tr>
<tr>
<td>checkGeant4_MS.py</td>
<td>compare Geant4 MCHits with LHCb extrapolator</td>
</tr>
<tr>
<td>checkMeasurements_pg.py</td>
<td>detailed check of Velo Cluster resolutions</td>
</tr>
<tr>
<td>checkMeasurements_ST.py</td>
<td>detailed check of ST Cluster resolutions</td>
</tr>
<tr>
<td>createPrivateDict.py</td>
<td>create a private dictionary for simple classes: int, double, string</td>
</tr>
<tr>
<td>database_localcopy.py</td>
<td>make a local copy of database</td>
</tr>
<tr>
<td>decodePUS.py</td>
<td>python decoding of the PUS detector</td>
</tr>
<tr>
<td>dst_sizes.py</td>
<td>measure I/O, only works on Windows</td>
</tr>
<tr>
<td>Ex0.py</td>
<td>simplest script</td>
</tr>
<tr>
<td>Ex10b.py</td>
<td>reading mdf file, running PatSeeding</td>
</tr>
<tr>
<td>Ex10.py</td>
<td>reading mdf file, calorimeter</td>
</tr>
<tr>
<td>Ex11b.py</td>
<td>Reconstruct B, plot B mass, convert to HBOOK</td>
</tr>
<tr>
<td>Ex11c.py</td>
<td>Reconstruct B, plot B mass, convert to HBOOK, DC06</td>
</tr>
<tr>
<td>Ex11d.py</td>
<td>Reconstruct B, plot B mass, lines for proto2mc link</td>
</tr>
<tr>
<td>Script</td>
<td>Short Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ex11.py</td>
<td>reconstruct B, plot B mass</td>
</tr>
<tr>
<td>Ex12b.py</td>
<td>Decay tree, increase output level via python opts</td>
</tr>
<tr>
<td>Ex12.py</td>
<td>Decay tree</td>
</tr>
<tr>
<td>Ex13b.py</td>
<td>copy only Raw banks for L0 triggered events</td>
</tr>
<tr>
<td>Ex13c.py</td>
<td>Read back L0 triggered events</td>
</tr>
<tr>
<td>Ex13d.py</td>
<td>Copy events with B candidates</td>
</tr>
<tr>
<td>Ex13.py</td>
<td>Make one to one copy of input DST using InputCopyStream</td>
</tr>
<tr>
<td>Ex14.py</td>
<td>Running Hlt, example printouts</td>
</tr>
<tr>
<td>Ex15.py</td>
<td>visualize a track</td>
</tr>
<tr>
<td>Ex16.py</td>
<td>read HepMCEEvents and dump info</td>
</tr>
<tr>
<td>Ex17b.py</td>
<td>read ETC, Event Tag Collection. make mass plot</td>
</tr>
<tr>
<td>Ex17.py</td>
<td>read ETC, Event Tag Collection</td>
</tr>
<tr>
<td>Ex1b.py</td>
<td>read DST, print header</td>
</tr>
<tr>
<td>Ex1.py</td>
<td>read DST, unpack all packed containers read DST, unpack all packed containers</td>
</tr>
<tr>
<td>Script</td>
<td>Short Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Ex_multicore.py</td>
<td>examples for running script on multicore machines</td>
</tr>
<tr>
<td>Ex1_multicore.py</td>
<td></td>
</tr>
<tr>
<td>Ex2b_multicore.py</td>
<td></td>
</tr>
<tr>
<td>Ex2c_multicore.py</td>
<td></td>
</tr>
<tr>
<td>Ex2_multicore.py</td>
<td></td>
</tr>
<tr>
<td>Ex2_singlecore.py</td>
<td></td>
</tr>
<tr>
<td>Ex3_multicore.py</td>
<td></td>
</tr>
<tr>
<td>Ex3_singlecore.py</td>
<td></td>
</tr>
<tr>
<td>Ex3.py</td>
<td></td>
</tr>
<tr>
<td>Ex2.py</td>
<td>read DST, fill ROOT histogram</td>
</tr>
<tr>
<td>Ex3b.py</td>
<td>DataOnDemand example</td>
</tr>
<tr>
<td>Ex3c.py</td>
<td>print size of containers, access to doxygen documentation</td>
</tr>
<tr>
<td>Ex3d.py</td>
<td>rewind file and fill another histogram</td>
</tr>
<tr>
<td>Ex3e.py</td>
<td>direct access to Aida histograms</td>
</tr>
<tr>
<td>Ex3.py</td>
<td>add algorithms to TopAlg</td>
</tr>
</tbody>
</table>
### Full list of Examples, cont.

<table>
<thead>
<tr>
<th>Script</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex4b.py</td>
<td>measure time to access different det elements</td>
</tr>
<tr>
<td>Ex4.py</td>
<td>access Velo detector element</td>
</tr>
<tr>
<td>Ex5.py</td>
<td>create objects, XYZPoint and Vector</td>
</tr>
<tr>
<td>Ex6b.py</td>
<td>use TrackMasterExtrapolator</td>
</tr>
<tr>
<td>Ex6.py</td>
<td>use Velo position tool</td>
</tr>
<tr>
<td>Ex7b.py</td>
<td>use ParticlePropertyService, ppSVC</td>
</tr>
<tr>
<td>Ex7c.py</td>
<td>use Magnetic Field Service</td>
</tr>
<tr>
<td>Ex7.py</td>
<td>use Transport Service</td>
</tr>
<tr>
<td>Ex8.py</td>
<td>use linker tables to access MC truth</td>
</tr>
<tr>
<td>Ex9.py</td>
<td>dump raw bank in HEX and ASCII format</td>
</tr>
<tr>
<td>Ex_Castor.py</td>
<td>access to castor information</td>
</tr>
<tr>
<td>Ex_mdf.py</td>
<td>open an MDF file and dump an event</td>
</tr>
<tr>
<td>ExMuonEff.py</td>
<td>check muon id efficiency as function of pt</td>
</tr>
<tr>
<td>Ex_TisTos.py</td>
<td>run Hlt1+Hlt2 and check for signal Tis Tos</td>
</tr>
</tbody>
</table>
## Full list of Examples, cont.

<table>
<thead>
<tr>
<th>Script</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fakeBreco.py</td>
<td>reconstruct B decays by following links from MC to reconstructed tracks</td>
</tr>
<tr>
<td>fakeJpsireco.py</td>
<td>reconstruct J/psi by following links from MC to reconstructed tracks</td>
</tr>
<tr>
<td>FullMonty.py</td>
<td>run Gauss/Boole/Brunel in one go</td>
</tr>
<tr>
<td>FullSequence.py</td>
<td>run reconstruction and phys sequence in one go</td>
</tr>
<tr>
<td>gaudigadgets.py</td>
<td>additional useful gaudipython methods</td>
</tr>
<tr>
<td>HLTandOffline.py</td>
<td>compare Hlt tracking and offline tracking</td>
</tr>
<tr>
<td>hltexamples.py</td>
<td>hltexample module</td>
</tr>
<tr>
<td>HltVeloPos.py</td>
<td>plot primary vertex positions of Velo A and C side</td>
</tr>
<tr>
<td>IPandPresol_plot.py</td>
<td>determine and plot IP and momentum resolution and pulls</td>
</tr>
<tr>
<td>KsCheck.py</td>
<td>investigate Kshort reconstruction</td>
</tr>
<tr>
<td>L0DUextract.py</td>
<td>write L0 triggered events with additional user info</td>
</tr>
<tr>
<td>L0DUreadtest.py</td>
<td>read back L0 triggered events</td>
</tr>
<tr>
<td>LHCbConfig.py</td>
<td>module for basic LHCb software configuration</td>
</tr>
<tr>
<td>microDSTEx0.py</td>
<td>write a microDST</td>
</tr>
</tbody>
</table>
### Full list of Examples, cont.

<table>
<thead>
<tr>
<th>Script</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>microDSTEx1.py</td>
<td>read a microDST</td>
</tr>
<tr>
<td>myBrunel.py</td>
<td>main script for Brunel</td>
</tr>
<tr>
<td>myGauss.py</td>
<td>main script for Gauss</td>
</tr>
<tr>
<td>myBoole.py</td>
<td>main script for Boole</td>
</tr>
<tr>
<td>ParallelMonty.py</td>
<td>run Gauss/Boole/Brunel in one go parallel on n processors</td>
</tr>
<tr>
<td>pickleROOT.py</td>
<td>not needed anymore, now included in ROOT</td>
</tr>
<tr>
<td>PrimVX.py</td>
<td>compare vertices reconstructed in different Velo halves</td>
</tr>
<tr>
<td>protonDecay_multicore.py</td>
<td>look for strange elastic events</td>
</tr>
<tr>
<td>rad_map06.py</td>
<td>produce map of radiation length at various positions</td>
</tr>
<tr>
<td>rawbuffer_sizes_2008.py</td>
<td>determine rawbuffer sizes</td>
</tr>
<tr>
<td>ReadLogFiles_2008.py</td>
<td>read logfiles on the internet</td>
</tr>
<tr>
<td>ReadLogFiles.py</td>
<td>plot position of MCHits</td>
</tr>
<tr>
<td>readSIM_MCHits.py</td>
<td>direct access to Aida histograms</td>
</tr>
<tr>
<td>Reco.py</td>
<td>run track reconstruction</td>
</tr>
<tr>
<td>Script</td>
<td>Short Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>replace.py</td>
<td>replace string A with B in files of current directory</td>
</tr>
<tr>
<td>RootExa1.py</td>
<td>create, fill and plot a root histogram</td>
</tr>
<tr>
<td>RootExa2.py</td>
<td>simulate Landau distributions, save file and read back</td>
</tr>
<tr>
<td>runall.py</td>
<td>run a list of selected scripts and examine log files</td>
</tr>
<tr>
<td>run_Ex10b.py</td>
<td>re-edits Ex10b.py for different TAE slots and runs it</td>
</tr>
<tr>
<td>selv19r9.py</td>
<td>write out selected events</td>
</tr>
<tr>
<td>TGhost.py</td>
<td>print info about ghosts</td>
</tr>
<tr>
<td>VeloAlignment_0.py</td>
<td>velo misalginment scenario, input config file</td>
</tr>
<tr>
<td>VeloCluster.py</td>
<td>make hitmaps and Landau distributions for all Velo sensors</td>
</tr>
<tr>
<td>VeloDet06.py</td>
<td>access Velo detector element, plot velo sensor geometry</td>
</tr>
<tr>
<td>velopostool_check.py</td>
<td>check Velo position tool</td>
</tr>
</tbody>
</table>