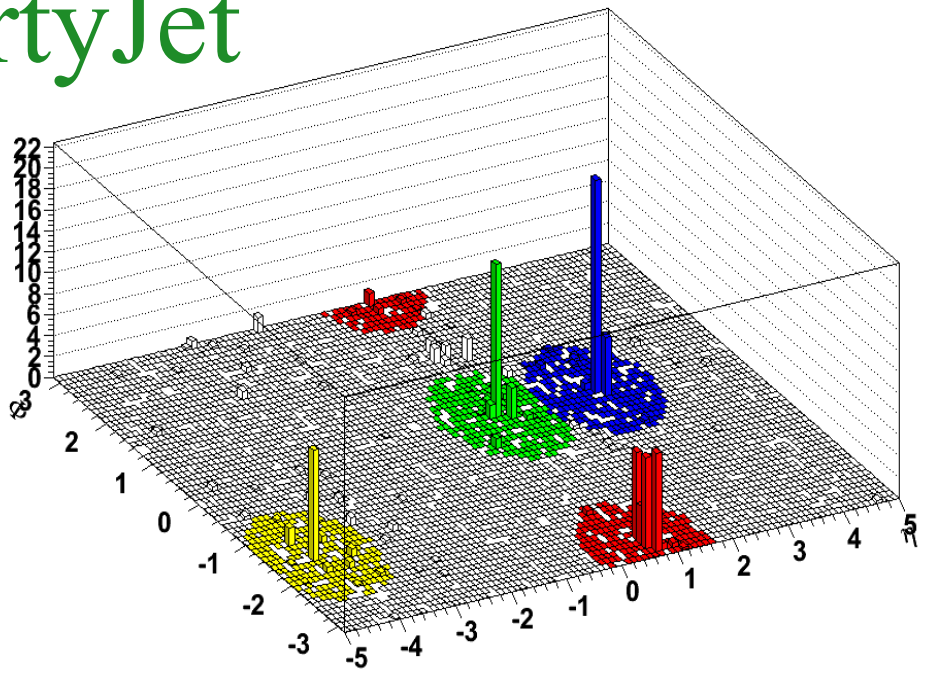
Tier 3 Use Example: D3PD making => JetFinding with SpartyJet

Brian Martin

April 2nd, 2010

ANL - Jamboree



Introduction

This is a collection of tasks one might do on a tier 3 for an analysis

- *In fact its pretty much how I use the tier3*

1. Local ATHENA use: Modify/Test EWPA package to produce sample D3PD
2. Pathena use: Submit to GRID using pathena to produce D3PDs for entire dataset
3. DQ2 use: Retrieve output dataset to Tier3.
4. Interactive ROOT: Run ROOT to design/test SpartyJet job.
5. Batch use: Run SpartyJet on entire dataset utilizing bach system (Arcond/Condor)
6. Ntuple use: Final analysis in compiled ROOT code to produce histograms

Exercises much of the Tier3 capabilities

Introduction

Every**W**here**P**hysics**A**nalysis – common,supported DPD making tools, analysis framework

- Very configurable
- Used extensively in ATLAS, ex. W/Z groups

Author: *Massimiliano Bellomo*

SpartyJet – Jet finding interfaces as well as core jet finding.

- Modular Jet Tool design
- Handles multiple types of input
- Native algorithms and built-in FastJet implementation
- Produces convenient ROOT output
- http://projects.hepforge.org/spartyjet/

Authors: *Joey Huston, Pierre-Antoine Delsart*

Kurtis Geerlings, Brian Martin

EWPA Example: D3PD production

Setup Athena:

```
> export ATLAS_LOCAL_ROOT_BASE=/export/share/atlas/ATLASLocalRootBase
> source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
> export ATLAS_TEST_AREA=/users/brianmartin/testarea_t3g/15.6.7
> localSetupGcc --gccVersion=gcc432_x86_64_slc5
> source /export/home/atlasadmin
        /temp/setupScripts/setupAtlasProduction_15.6.7.sh
```

Get Job options, configure and run:

```
> cp ~brianmartin/public/JamboreeApr2010/makeWjetsD3PD.py
> vim makeWjetsD3PD.py
> athena makeWjetsD3PD.py
```

EWPA Example: job config

```
EvtMax = 100

EWDataType            = "SIMUL"
DetDescrVersion       = "ATLAS-GEO-08-00-02"
loadTruth             = True
loadTrigger           = True
loadTrack             = True
loadMuon              = True
loadElectron          = True
loadPhoton            = False
loadJet               = True
loadCluster           = True
loadTauJet            = False
loadBJet              = False
loadMet               = True

useExtrapolator       = False
useEWOverlapper       = False
useEWAssociator       = False
useEWPerformance      = False

writeD3PD             = True
readD2PD              = False

DPDOutput             = "EWPA.D3PD.root"
```

SpartyJet Algorithms

- SpartyJet directly implements some algorithms:
  - Atlas Cone&FastKt – CDF MidPoint&JetClu
  - Pythia CellJet – D0 Cone
- SpartyJet links to FastJet to allow use of FastJet algorithms
  - Currently this is setup to allow the native algorithms (kT, anti-kT, and Cambridge-Aachen) plus the SISCone plugin.
  - Many other algorithms can be used via the Plugin interface
  - FastJet Ysplitter interfaced as well
- SpartyJet implements various jet tools as well
  - Jet shapes, filters, Calorimeter Grid, JetArea correction, ...
  - These act on a jet list in succession
    - In SpartyJet a jet algorithm is simply a specific type of jet tool

SpartyJet Architecture

InputMaker

```
fillInput(int eventn,
      Jet::jet_list_t &inputList)
```

Reads an input collection of 4-vectors and converts to an **initial jets list**

JetAlgorithm

```
addTool(JetTool * tool);
execute(Jet::jet_list_t &inputJets,
      JetCollection &outputJets);
```

Feed initial jets list into sequence of JetTools

JetTool

jets list

JetTool

jets list

JetTool

jets list

JetTool

JetTool modify Jets in sequence

```
execute(JetCollection
&inputJets)
```

NtupleMaker

```
addJetVar(std::string jetname);
set_data(std::string jetname,
      JetCollection &theJets);
```

Handles ntuple creation of jet results

SpartyJet Architecture: Input

InputMaker

```
fillInput(int eventn,
      Jet::jet_list_t &inputList)
```

Reads an input collection of 4-vectors and converts to an **initial jets list**

- **NtupleInput** – uses ROOT to read particles from TTree
  - Most flexible, accepts:
    - px,py,pz,E
    - eta,phi,pt,E
    - eta,phi,pt,M
  - Requires user to know if branches are arrays or vectors (tree->MakeClass())
  - Or use shortcut in python

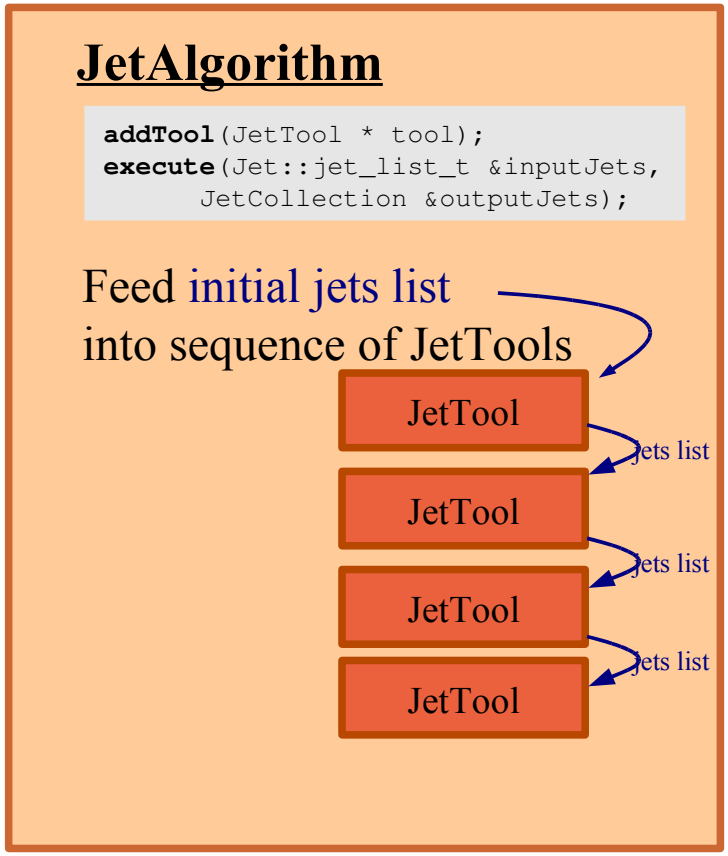
SpartyJet can handle several types of input:

- **StdTextInput** – four vector fields with some event demarcation
  - ".E" or ".e" ends an event
  - ".N" or ".n" begins an event
  - Only (px,py,pz,E) implemented for text
- **StdHepInput** – uses external stdhep libraries
- **CalchepPartonTextInput** – similar to StdText only all particles in single line (also only px,py,pz,E)
- **HepMCInput** – Text input with particle info (HepMCv2 and later)
- **LHEInput** – LesHouchesEvent format

SpartyJet Architecture: JetAlgorithm

Core of SpartyJet:

- Holds all the Jet Tools
  - Jet Tools can be added in three places:
    - Before jet finder
    - After jet finder
    - Directly to input
- Executes each JetTool on the JetCollection in succession
- Also contains a JetMomentMap for storing algorithm-wide quantities
  - Event characterization (like Thrust)



SpartyJet Architecture: JetTool

Worker of SpartyJet:

- Performs all the actions on the Jets
- Examples of JetTools:
  - The algorithms themselves (eg AtlasConeFinder inherits from JetTool)
  - SelectorTools – For filtering input and output based on kinematics or pdgId
  - JetAreaCorrectionTool – corrects jet pT based on event-determined low pT Activity and jet area
  - CalorimeterSimTool – puts particles in a simulated detector grid
  - NegEnergyTool – handles Jets with negative energy so that they behave in jet finding
  - EventShapeTools – ex JetThrust
  - JetMomentTools – ex Hull moments

JetTool modify Jets in sequence

```
execute(JetCollection &inputJets)
```

SpartyJet Architecture: Ntuple Maker

SpartyJet can store jet information in ntuples:

- Creates/Fills branches for each component of the jet four vector
- Has option to save the original input and include indices to match each constituent with the jet into which it was clustered
  - "MyKtJet_ind" indicates to which Jet each input belongs
- Creates/Fills branches for each Jet moment created by the JetTools
- User can choose whether branches are stored as arrays or vectors

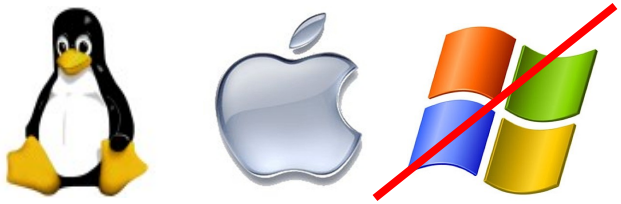
NtupleMaker

```
addJetVar(std::string jetname);
set_data(std::string jetname,
      JetCollection &theJets);
```

Handles ntuple creation of jet results

Getting SpartyJet

SpartyJet is now available on HEP Forge



Requirements: linux or mac OSX,

ROOT >=5.18, python >=2.4

- Before compiling on ASC Tier3g:

```
localSetupGcc -gccVersion=gcc432_x86_64_slc5
localSetupPython --pythonVersion=2.5.2
localSetupROOT --rootVersion=5.26.00-slc5-gcc4.3
```

- Get tarball from HEPForge and compile:

```
> wget http://projects.hepforge.org/spartyjet/spartyjet_3.4.1.tar.gz
> tar -xvzf spartyjet_3.4.1.tar.gz
> cd spartyjet_3.4.1
> make
```

Running SpartyJet

SpartyJet ships with a set of examples and input data

- Examples are organized by type:
  - Python scripts: spartyjet/examples_py/ Recommended
  - ROOT scripts: spartyjet/examples_ROOT/
  - Compiled C++ programs: spartyjet/examples_C
- All examples should be run from their directories

C++

```
cd  examples_C/
./simpleExample.exe
```

Python

```
source setup.sh
cd  examples_py/
./simpleExample.py
```

ROOT

```
cd  examples_ROOT/
root -l simpleExample.C
```

SpartyJet Example: run on EWPA D3PD

From /users/brianmartin/public/JamboreeApr2010/spartyExample.py

```
builder = SJ.JetBuilder(SJ.INFO) # Create the job manager: JetBuilder
# INFO describes the message output level

# Configure input ------------------------------
input = createNtupleInputMaker('EWPA.D3PD.root',inputprefix='TopoCluster')
builder.configure_input(input)

# Configure algorithms -------------------------
anti4 = SJ.fastjet.FastJetFinder('AntiKt4',fj.antikt_algorithm,0.4)
builder.add_default_alg(anti4) # Add to the builder

# Configure output-----------------------------
# Optional text output
#builder.add_text_output("simple.dat")
# Ntuple Output
builder.configure_output("SpartyJet_Tree","EWPA.SJ.root");

# Run SpartyJet
builder.process_events(10)
```

Pileup Study: Overlay minbias events

1. Produce EWPA D3PD with pileup (go it on the GRID)

```
localSetupPandaClient
pathena makeMBD3PD.py --inDS
mc09_7TeV.105001.pythia_minbias.merge.AOD.e517_s745_s746_r1098_r1113/ --outDS
user10.BrianThomasMartin.mc09_7TeV.105001.pythia_minbias.merge.DPD.e517_s745_s
746_r1098_r1113_TEST --noBuild --extOutFile EWPA.D3PD.root --dbRelease
ddo.000001.Atlas.Ideal.DBRelease.v080602:DBRelease-8.6.2.tar.gz --nFiles 1
```

2. Copy D3PD from GRID (in clean shell)

```
export ATLAS_LOCAL_ROOT_BASE=/export/share/atlas/ATLASLocalRootBase
source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
localSetupDQ2Client
voms-proxy-init -voms atlas
dq2-get
user10.BrianThomasMartin.mc09_7TeV.105001.pythia_minbias.merge.DPD.e517_s745_s
746_r1098_r1113_TEST
```

3. Use SpartyJet to overlay minbias event on signal events.

```
# Overlay Minbias events
mbInput =
createNtupleInputMaker('/users/brianmartin/public/JamboreeApr2010/Minbias.D3PD
.root', inputprefix='TopoCluster')
numMinBias = 4
builder.add_minbias_events(numMinBias,mbInput,True)
# Bool is whether to draw no. of mb events from poisson
```

Runs on Tier3g Batch using Condor

1. In a clean shell, setup Tier3g and gcc/python/ROOT

```
export ATLAS_LOCAL_ROOT_BASE=/export/share/atlas/ATLASLocalRootBase
source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
localSetupGcc -gccVersion=gcc432_x86_64_slc5
localSetupPython --pythonVersion=2.5.2
localSetupROOT --rootVersion=5.26.00-slc5-gcc4.3
```

2. Make simple job config file: spartyOnCondor.py

```
universe        = vanilla
executable      =/users/brianmartin/public/JamboreeApr2010/spartyExample.py
getenv          = True
arguments       =""
output          = job.local.out
error           = job.local.err
log             = job.local.log
WhenToTransferOutput = ON_EXIT_OR_EVICT
queue 1
```

3. Submit to condor

```
condor_submit spartyOnCondor.sub
```

SpartyJet Example: FastJet plugins

From FJExample.py

```
### JetPruning:
antikt10 = fj.JetDefinition(fj.antikt_algorithm,1.0)
antiktBIG = fj.JetDefinition(fj.antikt_algorithm,3.14*0.5)
prunePlugin = fj.FastPrunePlugin(antikt10,antiktBIG,0.1,0.5)
pruneJetDef = fj.JetDefinition(prunePlugin)
prune = SJ.FastJet.FastJetFinder(pruneJetDef,'Prune',False)
builder.add_default_alg(prune)
```

Many exciting new jet sub-structure tools being developed in the community:

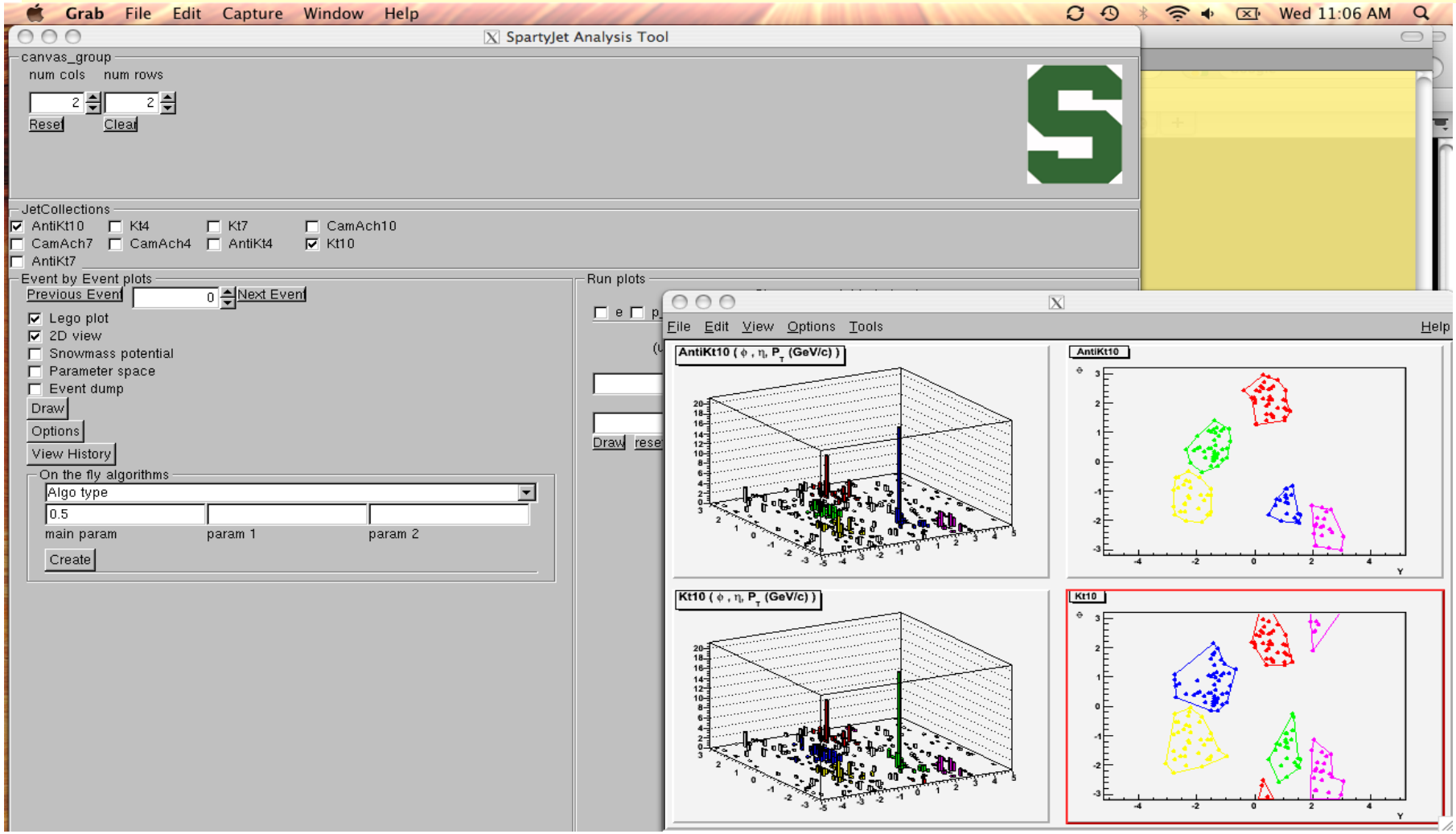
- Jet Pruning:
  http://www.phys.washington.edu/groups/lhcti/pruning/FastJetPlugin/
- Trimming: http://jthaler.net/jets/Jet_Trimming.html
- Variable R jets
- Various taggers

- These plugins can be used directly in SpartyJet

SpartyJet Example: gui (alpha)

guiExample.py

```
./guiExample.py
```



Conclusions

- The Tier3g setup has all the functionality to carry out analysis requiring diverse tools
- It is easy to use and well documented
  - Starting from never having run on this system before, I was able to do all of this in less than an hour
- Most importantly, SpartyJet runs on it.