Template for MC Generation in Gino

Monte Carlo generation of events will involve creating runs from scratch with no external agent triggering runs.

The initial sample files can be found on SLAC unix at

/nfs/farm/g/glast/u13/MC-tasks/

the task configuration information is in the <task-name>/config directory.

Caveats at time of writing

- a modified GlastSvc is needed to use an environment variable in the job options file for the run number. It is available in GlastSvc v9r12p1, and appeared in GlastRelease as of HEAD1.403; it is not in EngineeringModel yet!

Configuring the pipeline

Configuration is done from the glast-ground web server:

http://glast-ground.slac.stanford.edu/

Here is a sample xml file for upload to Gino:

```
<?xml version="1.0" encoding="UTF-8"?>
<pipeline
        xmlns="http://glast-ground.slac.stanford.edu/pipeline"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://glast-ground.slac.stanford.edu/pipeline http://glast-ground.slac.stanford.edu
/pipeline/pipeline.xsd">

        <name>allGamma-GR-HEAD1.403</name>
        <type>SimReconDigi</type>
        <dataset-base-path>/nfs/farm/g/glast/u13/MC-tasks/allGamma-GR-HEAD1.403/rootData/$(RUN_NAME)</dataset-
base-path>
        <run-log-path>/temp/</run-log-path>

        <executable name="GleamWrapper" version="v1r0">
                /nfs/farm/g/glast/u13/MC-tasks/allGamma-GR-HEAD1.403/config/GleamWrapper.pl
        </executable>

        <batch-job-configuration name="long-job"  group="glastgrp" queue="long">
        <working-directory>/nfs/farm/g/glast/u13/MC-tasks/allGamma-GR-HEAD1.403/config</working-directory>
                <log-file-path>/nfs/farm/g/glast/u13/MC-tasks/allGamma-GR-HEAD1.403/rootData/$(RUN_NAME)/log<
/log-file-path>
        </batch-job-configuration>

        <file name="digi" type="DIGI" file-type="root"/>
        <file name="merit" type="merit" file-type="root"/>
        <file name="recon" type="RECON" file-type="root"/>
        <file name="mc" type="MC" file-type="root"/>

        <processing-step name="gleam" executable="GleamWrapper" batch-job-configuration="long-job">
                <output-file name="digi"/>
                <output-file name="merit"/>
                <output-file name="recon"/>
                <output-file name="mc"/>
        </processing-step>
</pipeline>
```

In addition there are two files that the pipeline uses to run Gleam:

- GleamWrapper.pl - run by the pipeline
- gleam.pl - prepares environment variables for use in the Gleam shell script

In general these can be reused from task to task without modification (except gleam.pl does execute allGamma.sh - this could be made generic).

Configuring Gleam

There are 2 files that need to be modified in this process:

- allGamma.txt - job options file
  - the only items to modify are the source name and number of events
  - note that the run identifier is a character string (even though you don't have to do anything with it at this stage)
- allGamma.sh - shell script that launches Gleam
  - modify the CMTCONFIG and GlastRelease and Gleam versions

Submitting Jobs

A perl script is available to meter jobs into the pipeline (this allows bumping against the database connection limit). It will soon be available in the glast account directory, but is now in ~richard/GLAST/Pipeline/submitTasks.pl

export PDB_HOME=/u/gl/glast/pdb
and make sure it is in your path.

submitTasks.pl <max-run> <taskName> <maxNum2submitNow> <padLength>

createRun.pl <taskName> <runName>
deleteRun.pl <taskName> <runName>

are other handy commands. Note that deleteRun does not clean the output files off disk.

Accessing files via the pipeline catalogue

A Root class has been written to query the database to find requested files and create a TChain of them. These are in cvs at

http://www-glast.stanford.edu/cgi-bin/viewcvs/users/richard/pipelineDatasets/#dirlist

```
bash-2.05b$ source /nfs/farm/g/glast/u13/MC-tasks/utilities/setupOracleRoot.sh
bash-2.05b$ root -l
root 0 .L /nfs/farm/g/glast/u13/MC-tasks/utilities/pipelineDatasets/v0/rh9_gcc32/libpipelineDatasets.so
root 1 pipelineDatasets* p = new pipelineDatasets();
root 2 int sc = p->selectDatasets("allGamma-GR-HEAD1.403","merit");
root 3 TChain* c = p->makeChain("MeritTuple");
```

This example found alll files for the task; one can provide a run range or a list of runs to choose from. Note that the runs are character strings, and you have to know the padding depth for now (eg '001' is different from '0001'!).

Pruning/concatenating Ntuple files

The Root code also lives on disk at SLAC at

/nfs/farm/g/glast/u13/MC-tasks/utilities

There is a handy-dandy perl script now to do the pruning

```
$ source setupOracleRoot.sh
$ runPrune.pl '<taskName>' '<cut>'
```

this will fire up Root in batch mode and run RUN_Prune.cxx, producing an tuple file named <taskName>-ntuple-prune.root in the current working directory. It is not have to be run from the utilities directory (and is best not to!).

The standard cut is

TkrNumTracks>0