Validating GR for RHEL4

Michael Kuss is kindly leading the charge to check that our simulation results when running RHEL4 builds of GlastRelease are at statistically equivalent to our RHEL3 builds.

Simulations Run

GR **v17r34p0rhel4B**: http://glast-ground.slac.stanford.edu/SystemTests/?releaseVersionId=11881 OutputLevel was set to Debug.
Use **v17r34p0** as a reference which is the RHEL3 version of the systests. Please note **v17r34p0B** is another RHEL3 run and the results are seemingly identical..though OutputLevel was set to Debug. The resulting log file may be of interest.

Of particular interest has been the **BackGndMixDC2** test.

Full ROOT files are available on u17/systests/GlastRelease where one uses the name of the systest to drill down into the directory structure.

System test report with more plots is here

Testing on Real Data Run 274279559 (Sept 13, 2009)

Using **GR v15r47p12gr1**

http://glast-ground.slac.stanford.edu/DataProcessing/run.jsp?runid=274279559

while the RHEL4 files can be found in the DEV Data Catalog. Here are the most relevant ones:
root://glast-test-rdr.slac.stanford.edu//glast/Data/Flight/Level1/LPA/dev/1.77/merit/r0274279559_v003_merit.root
root://glast-test-rdr.slac.stanford.edu//glast/Data/Flight/Level1/LPA/dev/1.77/digi/r0274279559_v001_digi.root
root://glast-test-rdr.slac.stanford.edu//glast/Data/Flight/Level1/LPA/dev/1.77/svac/r0274279559_v001_svac.root

A diff was performed using Luca Baldini's diff tool, the results are displayed in this PDF from Anders. Note from Anders: "We do expect CalCfpEnergy (and CTBBestEnergy) to change for quite a few events." Anders suggests: "Have a look at p108 (bottom plot), p109 (both plots), 110 (top plot)."

Calling All GR Package Owners

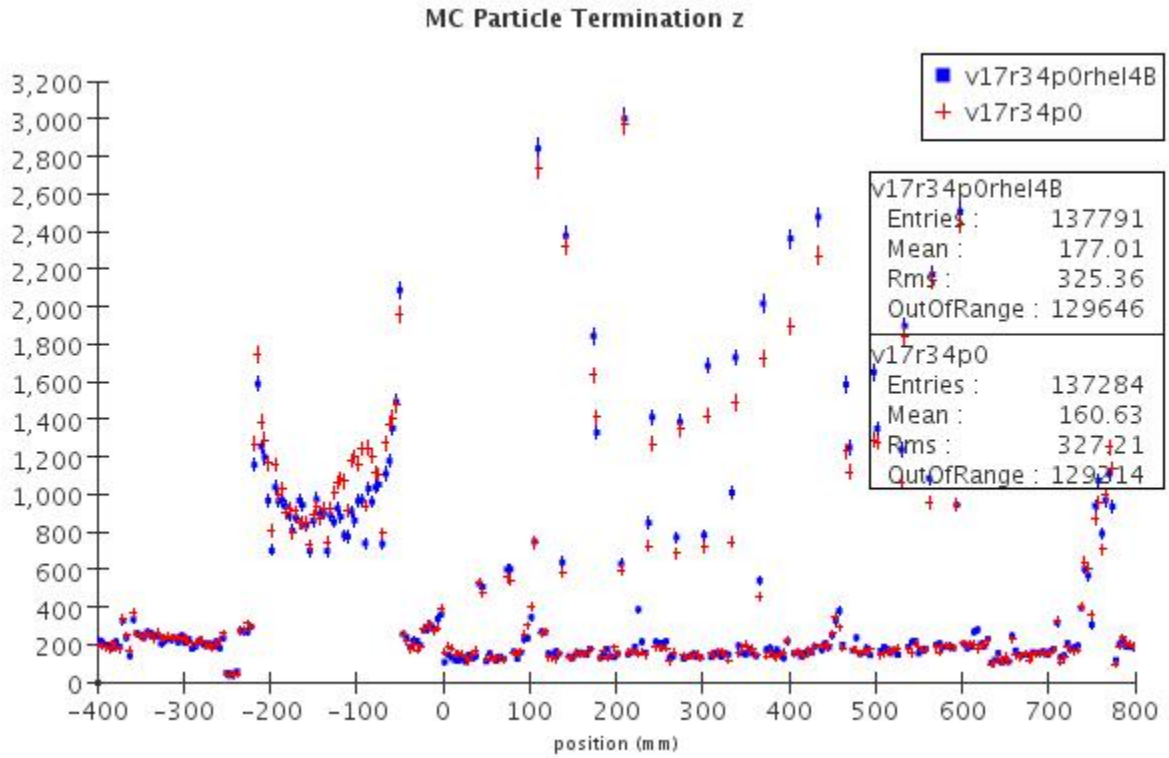
It is requested that all GR Package owners take a hard look at the warnings associated with their packages in the RHEL4 builds. Not sure what packages you own?? Check the list:

https://confluence.slac.stanford.edu/display/SAS/List+of+GR+Package+Owners+as+of+September+8+2009

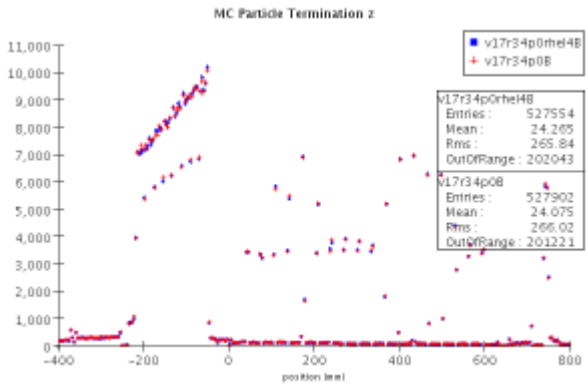
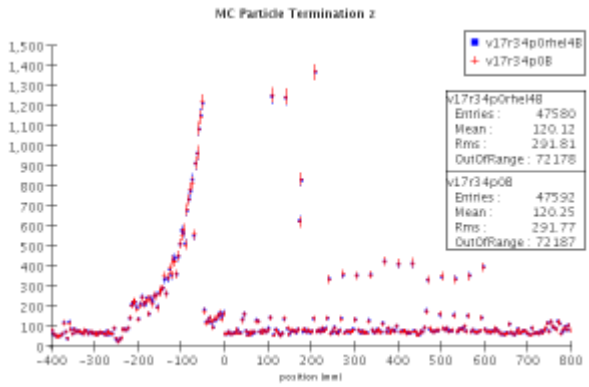
Update Sept 15,2009
Some owners have taken the time to fix up warnings in their packages. So far these tags have not propagated into any version of GR. No smoking gun has been found yet, but the clean up is to our own benefit regardless.

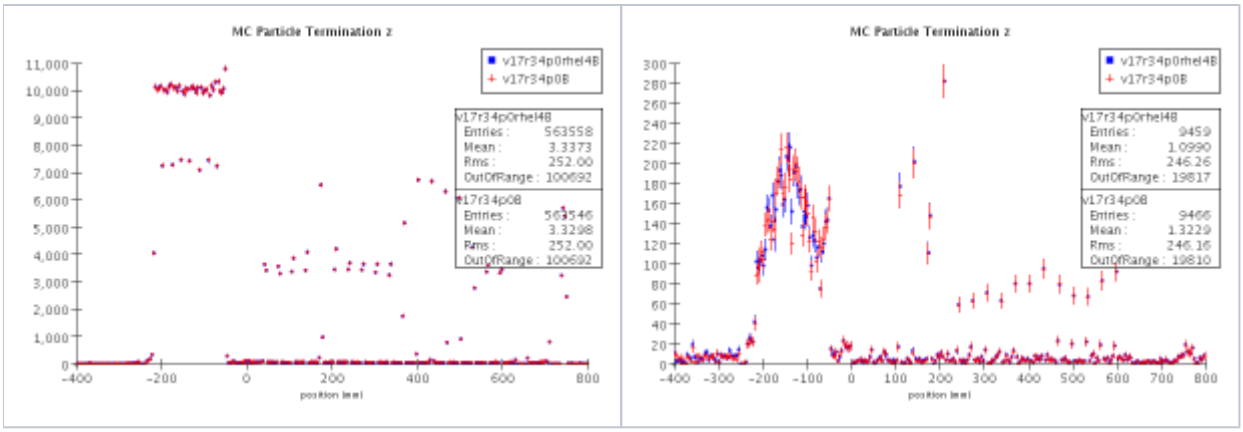
Questions Swirl Around MCTERMZ

Looking at BackGndMixDC2, one failed plot is MCTERMZ.



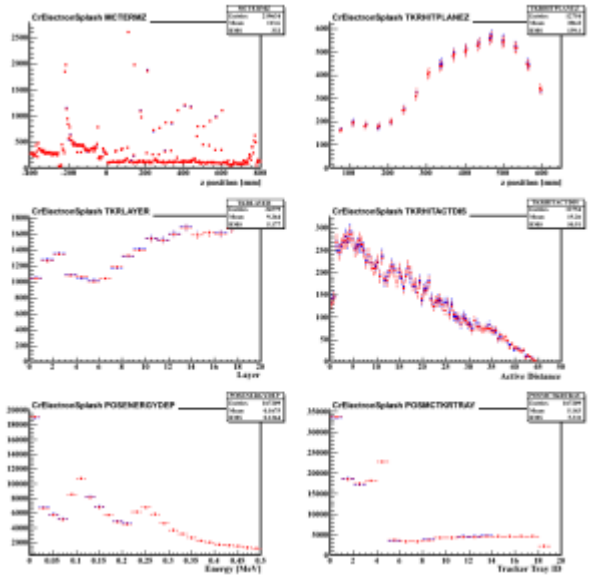
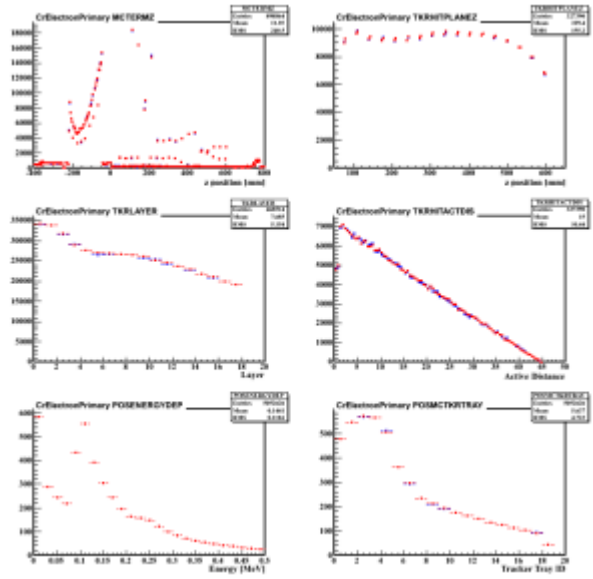
However, for the other tests, this distribution is very similar for the rh9 and the rhel 4 runs.

Vertical Proton 1 GeV	AllGamma
	
Vertical Muon 1 GeV	**Vertical Gamma 100 GeV**



And here are the results for dedicated CrElectronPrimary and CrElectronSplash runs for the rh9 and rhel4 builds. These are **not** composite flux sources. The plots that differ for the background mix runs are not identical, but are very similar as might be reasonably expected for minor differences caused by random number generators. This strongly supports the idea that it is just the balance among the composite mix that is changing in the MC generation and not anything in the actual performance of the reconstruction, etc.

Outputs available in
/afs/slac/g/glast/users/ehays/CrElectronPrimary
/afs/slac/g/glast/users/ehays/CrElectronPrimary_rhel4
/afs/slac/g/glast/users/ehays/CrElectronSplash
/afs/slac/g/glast/users/ehays/CrElectronSplash_rhel4

CrElectronSplash (40k)	CrElectronPrimary (40k)
	

Unsigned versus Signed Ints

There are a number of warnings in our builds concerning unsigned/signed match. These should be looked over and fixed.

An audit of existing JO properties is also in order to check for mismatches, as these will not be caught at compile time.

One specific JO issue concerns TriggerAlg.mask:

Trigger: there is something really weird with the trigger mask:

In the jobOptions:
TriggerAlg.mask = -1;

rh9_gcc32opt logs:
TriggerAlg INFO No trigger requirement
rhel_gcc34opt logs:
TriggerAlg INFO Applying trigger mask: 589cba38

In TriggerAlg::TriggerAlg() we find:
declareProperty("mask" , m_mask=0xffffffff);

```
and later, in TriggerAlg::initialize()
log << MSG::INFO;
    if(log.isActive()) {
        if (m_mask==0xffffffff) log.stream() << "No trigger requirement";
        else            log.stream() << "Applying trigger mask: " <<  std::setbase(16) <<m_mask <<std::setbase
(10);
        if( m_throttle) log.stream() <<", throttled by rejecting  the value "<< m_vetobits;
    }
log << endreq;
```

mask is an unsigned int. For the moment, could we rerun both systests with simply the Trigger.mask=-1; line commented, as all bits set is the default anyway? And note anyway that the code should be changed? rh9 interprets correctly in the sense that with (signed int)-1 0xffffffff is intended. How rhel4 comes up with 0x589cba38 beats me.

Suggested Fix:

Make TriggerAlg.m_mask a StringProperty and convert the value to an unsigned int via the tools available in facilities::Util. This has been tried out and seems to work.

Current Status:

Unfortunately, the systest results remain the same (comparing v17r34p0 versus v17r34p0B and v17r34p0rhel4 versus v17r34p0rhel4B, even after being sure that the Trigger.mask is set to the appropriate default. It seems that when the ConfigSvc is enabled, the trigger mask has no effect.

Exception in CalLikelihoodManagerTool

rhel4 also contains one event that threw an exception in CalLikelihoodManagerTool

Random Sequences Diverge

In regards to the BackGndMixDC2 run and the mix of events:

"As Richard mentioned, the two random sequences will diverge eventually. Hence, the particle mix for the 40k triggers is different. The biggest discrepancy is in "1002 CrElectronSplash", with 6409 trigger for rh9_gcc32opt vs. 6603 for rhel4. Consequently, the other sources are weighted stronger in rh9, in general."