System Tests

• What for?
  – Comprehensive tests producing histograms & diagnostic statistics
  – A subset of the suite a subsystem would have for its own performance reviews
  – Multiple test configurations possible
  – Could be applicable to packages other than Gleam
    • eg CalDigi single crystal tests
  – Run on Gleam tags and releases (subtle distinction 😊)
  – Tracked in database with web plotting display capabilities
  – Comparison to standards (deemed ‘correct’ by package owners!)
  – Tests to be discovered and run automatically by the Release Manager
    • will capture results for the db

• Proposed Implementation Details
  – Basic idea: RM discovers system test in package cvs repo
    • Runs script that sets everything up
      – Gleam configured by jobOptions
      – Root macro to analyze output Root files and produces histos
    • Retrieves info; inserts into db
  – Add SysTest directory to cvs package structure
    • Subdirectory for each test holding files relevant to that test
      – Shell script, jobOptions, Root macro + fragments per subsystem
      – Use env variable to redirect file output
  – Root macro based on RootTreeAnalysis
    • Require standard fragments from contributors
      – Histo definitions
      – Filling histos
      – MC; Digi (TKR, CAL, ACD; Recon (ditto); Trigger

See LAT-TD-00875 in Cyberdocs for Test Plan
What to Test?

- Expect to have a handful of test suites defined, based on source configuration:
  - Select from vertical, angled: muons, gammas, e’s(?), p’s, All_gamma, bkg-mix
  - Could envisage different test histograms tailored to the source.
    - eg look at hadron interaction length for protons as opposed to EM for gammas.
    - ‘pencil’ beams more diagnostic than ‘survey’ All_gamma, bkg-mix sources
- Should evolve as we learn which distributions are most telling to performance.
- Run test_Gleam (merit on 100 MeV gammas) as performance test.
  - Track PSF, Aeff in db.
  - Keep track of ChronoSvc output per package
  - etc
Concrete Example: CalDigi Single Xtal

- Example of a subsystem system test
  - Use xml to define a single CAL crystal: CsIElement
  - Single crystal centered at (0,0,0)
  - Use patch source to map out response along the length of the crystal
  - Look at resolution

- This example uses userAlg, but will use Gleam in future, configured by jobOptions

Example of shell script & running Root macro

noric10:richard> more systest_CALSingleCDEDigi
#!/bin/sh

cd $GLASTUSER/glastPackTest/userAlg

glastpack.pl login
cd $GLASTUSER/glastPackTest/test/SingleCDE-Electron

glastpack.pl run userAlg v2r1 userApp.exe
  SGLASTUSER/glastPackTest/test/SingleCDE-Electron/SingleCDEjobOptions.txt

root -b start.c RUN
mv Histograms.root SingleCDE-ElectronHistos.root

noric10:richard> more RUN
{
  gROOT->Reset();
gROOT->LoadMacro("CALSingleCDEDigi.cxx");

  RootTreeAnalysis* m= new
    RootTreeAnalysis("digi.root","","mc.root");

  m->Go();
m->histFile->Write();
}
void RootTreeAnalysis::DigiHistDefineCAL() {
    histFile->cd();

    TH1F *CALDIGICOUNT = new TH1F("CALDIGICOUNT", "Cal Digi multiplicity", 50, 0, 50);
    TH1F *CALAD = new TH1F("CALAD", "Cal Digi ADC - both faces", 200, 0, 1000);
    TH1F *CALADCN = new TH1F("CALADCN", "Cal Digi ADC - NEG", 200, 0, 1000);
    TH1F *CALADCP = new TH1F("CALADCP", "Cal Digi ADC - POS", 200, 0, 1000);
    TH2F *CALADCNMCX = new TH2F("CALADCNMCX", "Cal Digi ADC - NEG vs MC x", 50, -200, 200, 200, 0, 1000);
    TH2F *CALADCPMCX = new TH2F("CALADCPMCX", "Cal Digi ADC - POS vs MC x", 50, -200, 200, 200, 0, 1000);
    TH1F *RESIDUAL = new TH1F("RESIDUAL", "Cal Asy - MC POS", 200, -200, 200);
    TH1F *RESIDUALMCX = new TH1F("RESIDUALMCX", "Cal Asy - MC POS vs X", 50, -200, 200, 200, 0, 1000);
    TH2F *CALASYMCX = new TH2F("CALASYMCX", "Cal Light asy vs MC x", 50, -200, 200, -0.4, 0.4);
    TH1F *CALRANGE = new TH1F("CALRANGE", "Cal Digi Range - both faces", 10, 0, 10);
    TH1F *CALEAVE = new TH1F("CALEAVE", "Cal Digi Energy - faces/2", 200, 0, 1000);
    TH1F *CALEAVETOTAL = new TH1F("CALEAVETOTAL", "Cal Digi Energy - faces/2 summed", 200, 0, 10000);
    TH1F *CALLAYER = new TH1F("CALLAYER", "Cal Digi Layer", 15, 0, 15);
    TH1F *CALTOWER = new TH1F("CALTOWER", "Cal Digi Tower", 20, 0, 20);
    TH1F *CALCOLUMN = new TH1F("CALCOLUMN", "Cal Digi Column", 20, 0, 20);
};

void RootTreeAnalysis::DigiCal() {
    TObjArray* cL = evt->getCalDigiCol();
    int nCalDigi = cL->GetEntries();
    ((TH1F*)GetObjectPtr("CALDIGICOUNT"))->Fill((Float_t)nCalDigi);
    int nLayer[8]={0,0,0,0,0,0,0,0};
    float eLayer[8]={0.,0.,0.,0.,0.,0.,0.,0.};
    float eTotal = 0.;
    for (int ic=0; ic < nCalDigi; ic++) {
        CalDigi* c=(CalDigi*)cL->At(ic);
        CalXtalId id = c->getPackedId();
        int layer = id->getLayer();
        CalXtalReadout* cRo=c->getXtalReadout0();
        float adcN = cRo->getAdc(0);
        float adcP = cRo->getAdc(1);
        float asy = (adcP-adcN)/(adcP+adcN - 200.);
        float resid = 1446.*asy - mcX;       // electrons
        ((TH1F*)GetObjectPtr("RESIDUAL"))->Fill(resid);
        ((TH2F*)GetObjectPtr("RESIDUALMCX"))->Fill(mcX,resid);
        float eAve = (adcN+adcP)/2.;
        ((TH1F*)GetObjectPtr("CALEAVE"))->Fill(eAve);
        ((TH1F*)GetObjectPtr("CALADC") )->Fill(adcN);
        ((TH1F*)GetObjectPtr("CALADCP") )->Fill(adcP);
        ((TH2F*)GetObjectPtr("CALADCNMCX"))->Fill(mcX,adcN);
        ((TH2F*)GetObjectPtr("CALADCPMCX"))->Fill(mcX,adcP);
        ((TH1F*)GetObjectPtr("CALASYMCX"))->Fill(mcX,asy);
        ((TH1F*)GetObjectPtr("CALRANGE"))->Fill(cRo->getRange0());
        ((TH1F*)GetObjectPtr("CALTOWER"))->Fill(cRo->getRange1());
        int tower = id->getTower();
        int column = id->getColumn();
        ((TH1F*)GetObjectPtr("CALLAYER"))->Fill(layer);
        ((TH1F*)GetObjectPtr("CALTOWER"))->Fill(tower);
        ((TH1F*)GetObjectPtr("CALCOLUMN"))->Fill(column);
    }
    ((TH1F*)GetObjectPtr("CALEAVETOTAL"))->Fill(eTotal);

Sample Output

We’ll see what happens when comparing scatterplots!
Misc

- We’ll need to properly document the Root macros and describe the histograms

- We’re setting up Carrot/Root/Apache to access Root histos over the web
  - Plus ability to plot/histogram quantities from the database
    - eg plot of memory usage for Gleam vs version
  - May run into hitch with SCS security folk for Carrot
    - run server outside SLAC firewall
    - fallback is JAS/Tomcat à la I&T/Eddy
    - So far, server is blocked at the firewall
Status

• CalDigi single crystal is working prototype of test
• Defining shell of Root macro now for contributors
  – Tracy/Michael set to contribute; Richard for CAL
• Test database is defined and implemented
  – Under test with Carrot
• Have demonstrated all needed display capability with Carrot
• Release Manager
  – must learn to find and submit tests
  – gather up output
  – how to deal with individual statistics (rather than histograms) which may be test specific in meaning and method of obtaining (eg a Landau fit to crystal energy deposit).