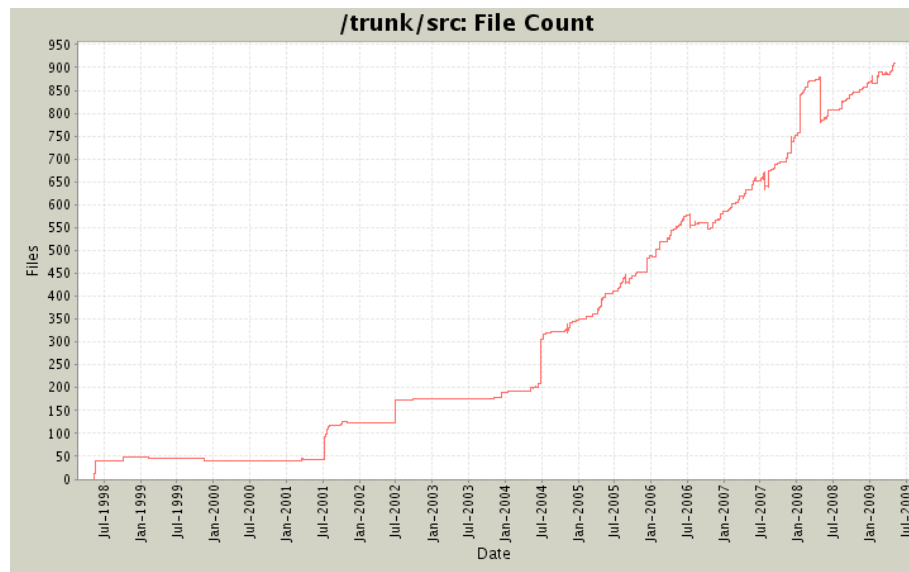
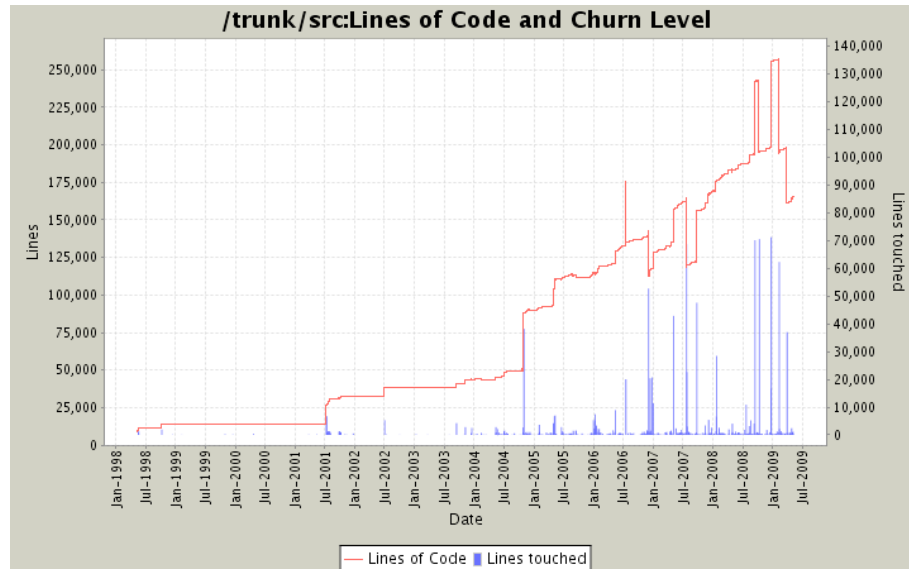


# Hall-D Software Status

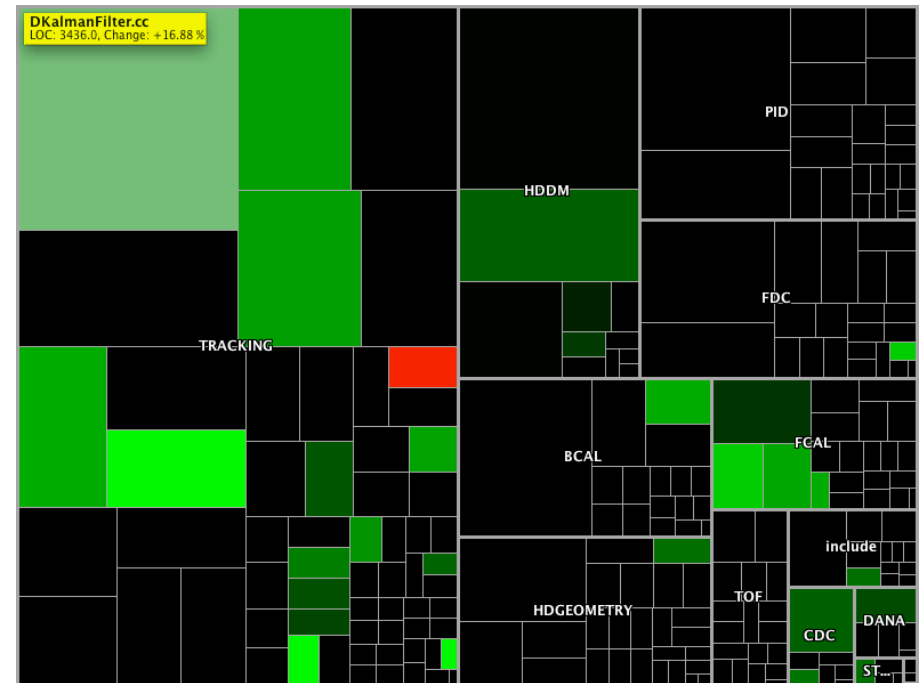
May 12, 2009

David Lawrence

# Repository Activity



5/12/09



## Developer of the Month

Month	Author	Lines
<a href="#">May 2009</a>	<a href="#">davidl</a>	178
<a href="#">April 2009</a>	<a href="#">davidl</a>	3985
<a href="#">March 2009</a>	<a href="#">davidl</a>	919
<a href="#">February 2009</a>	<a href="#">davidl</a>	2856
<a href="#">January 2009</a>	<a href="#">staylor</a>	665
<a href="#">December 2008</a>	<a href="#">davidl</a>	69991
<a href="#">November 2008</a>	<a href="#">staylor</a>	1795
<a href="#">October 2008</a>	<a href="#">davidl</a>	11666
<a href="#">September 2008</a>	<a href="#">zihlmann</a>	59062
<a href="#">August 2008</a>	<a href="#">davidl</a>	5288
<a href="#">July 2008</a>	<a href="#">zihlmann</a>	6361
<a href="#">June 2008</a>	<a href="#">davidl</a>	1255
<a href="#">May 2008</a>	<a href="#">davidl</a>	1567

One tagged release of Hall-D source since last collaboration meeting:

*release-2009-02-04*

# Repository Changes

- Start counter 40-stave geometry (not default)
- CDC Geometry modified to reflect final design
- Gas Cerenkov detector removed\*
- Control cards in *hdgeant*
  - Pattern key size extended from 4 to 16 characters
  - SAVEHITS enable/disable “no hit” events in output
  - SHOWERS\_IN\_COL enable/disable showers in collimators
  - PLOG sample momentum from log distribution for particle gun
  - TLOG sample theta angle from log distribution for particle gun
- *DMagneticFieldMapSpoiled* class added to allow simulation or reconstruction with a “spoiled” field
- Updated *invariant\_mass\_hists* plugin which provides an example of how to use reconstructed values in an analysis

*\*or will be very soon*

# ... Repository Changes

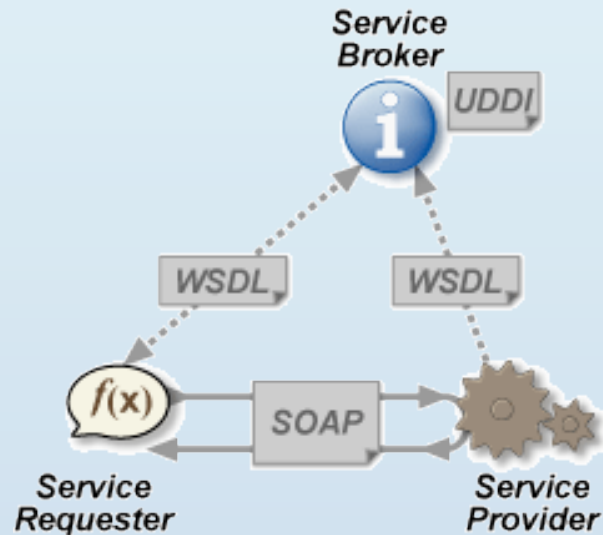
- Detector numbering scheme
  - ID number increases as lab coordinate increases
  - TOF and FCAL updated (others OK)
- Added material map for tracking
  - Simon's radlen map (deprecated)
  - Beni's DRootGeom class (Kalman and ALT1 fitters)
- hddsGeant3.F removed from repository
- DTrackHitSelector formalizes hit selection
- Calorimetry
  - BCAL segments drawn in hdview2
  - BCAL threshold based on readout device
  - BCAL dark hits added to response (post *hdgeant*)
  - FCAL radiation hard inner layer
  - TwoGammaFit updated to include both pre and post fit photons

# Framework Development

- JANA releases since last meeting:
  - Jan. 25     release 0.4.9
  - Mar. 10     release 0.5.1
  - May 1       release 0.5.2
- New features:
  - Optional recording and dumping of calibration requests
  - Option to have framework maintain ownership of calibration constants
  - Discovery mechanism for calibration system
  - gSOAP and calibration DB access through Web Service
  - Optional dumping of configuration parameters at end of job

...slide shown at CHEP09 ...

# Calibration Web Service



- Calibration constants will need to be accessible from remote computers via the internet
- Direct access to a database is problematic due to cybersecurity concerns
- Web services work over HTTP and so are the appropriate mechanism for remote access
- The *JCalibrationWS* class provides calibration constants through a web service
  - Implemented as a plugin so remote access can be added to an existing executable
  - Allows read-only access to calibration constants from anywhere in the world over HTTP (<http://www.jlab.org/Hall-D/cgi-bin/calib>)
  - Uses gSOAP, a C++ SOAP implementation
  - Currently works like a proxy for *JCalibrationFile* on server side, but could trivially be made to use another type of backend

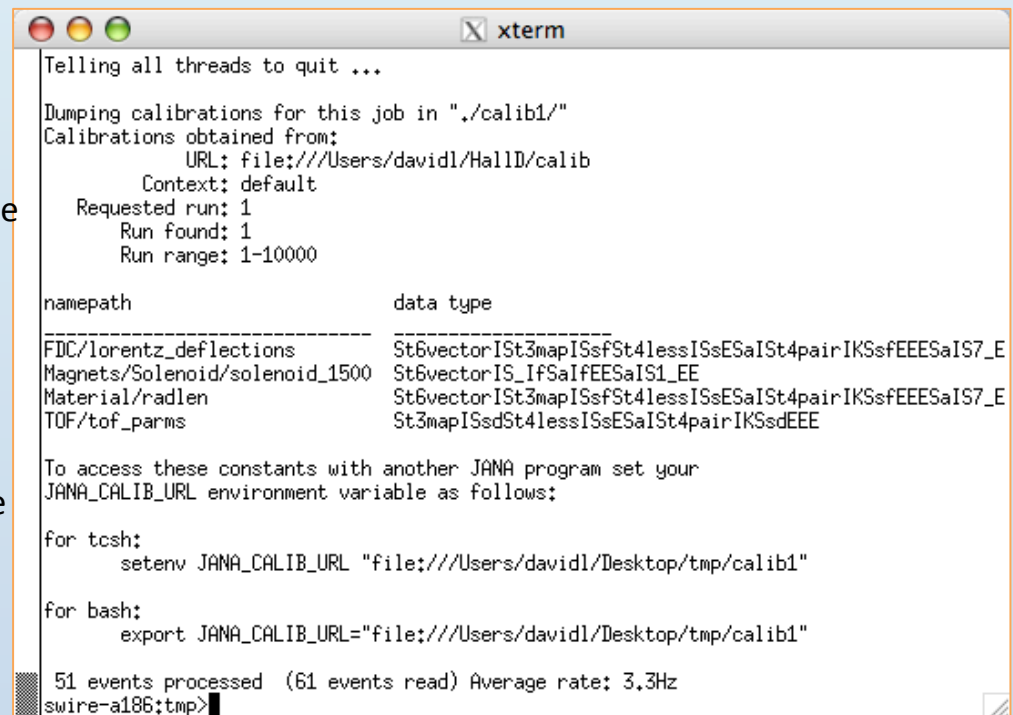
...slide shown at CHEP09 ...

# Saving a (semi-)complete set of calibration constants to the local disk

All JANA programs have the command line option:

`--dumpcalibrations`

- Records which *namepaths* are requested during a job and writes the constants into ASCII files compatible with *JCalibrationFile*
- Avoids copying and running entire database or even copying a “complete” set of calibration constants (which could include obsolete ones or ones not applicable to the current run/code version)



```
Telling all threads to quit ...

Dumping calibrations for this job in "./calib1/"
Calibrations obtained from:
  URL: file:///Users/davidl/HallD/calib
  Context: default
  Requested run: 1
  Run found: 1
  Run range: 1-10000

namepath                                data type
-----
FDC/lorentz_deflections                 St6vectorISfSt4lessISsESaIS4pairIKSsfEEESaIS7_E
Magnets/Solenoid/solenoid_1500          St6vectorIS_IFSaIFEEESaIS1_EE
Material/radlen                         St6vectorISfSt4lessISsESaIS4pairIKSsfEEESaIS7_E
TOF/tof_parms                          St3mapISsdSt4lessISsESaIS4pairIKSsdEEE

To access these constants with another JANA program set your
JANA_CALIB_URL environment variable as follows:

for tcsh:
  setenv JANA_CALIB_URL "file:///Users/davidl/Desktop/tmp/calib1"

for bash:
  export JANA_CALIB_URL="file:///Users/davidl/Desktop/tmp/calib1"

51 events processed (61 events read) Average rate: 3.3Hz
swire-a186:tmp>
```

...poster shown at CHEP09 ...

CHEP 2009  
Prague

# Multi-threaded Event Reconstruction with JANA

## JANA

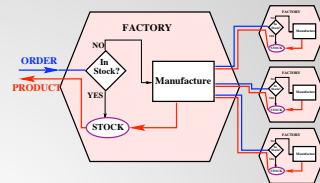
JANA is a multithreaded event reconstruction framework written in C++. It is designed to utilize all of the available cores of a CPU while processing high volume HENP data.

## Data On Demand

JANA's modified factory model produces data only on demand. This avoids wasting CPU cycles on reconstruction that is not needed for that particular event.

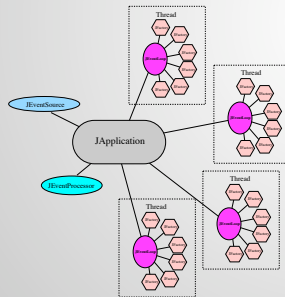
## Alternate Factory Model

Reconstruction code is built into factory classes as callback methods. Inputs come from other factories. In JANA's model, ownership of the objects stays with the factory. Only const pointers are passed, ensuring data integrity.



## Threading Model

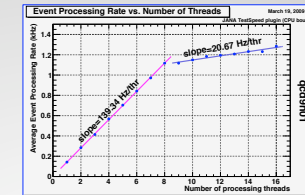
A JANA application consists of a single *JApplication* object and multiple *JEventLoop* objects (one for each thread). Each thread has its own complete set of factory objects that together, can completely process an event. JANA uses POSIX pthreads.



Website : <http://www.jlab.org/JANA>

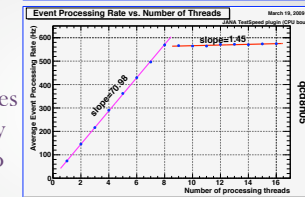
## CPU bound Jobs

CPU bound jobs benefit from the parallelism achieved through multi-threading. This benefit can be lost if the framework requires frequent mutex (un)locking. These plots show event processing rates for CPU bound jobs. The linear shape indicates proper scaling with the number of threads which implies virtually no overhead is imposed by the framework which is designed to minimize mutex (un)locking.



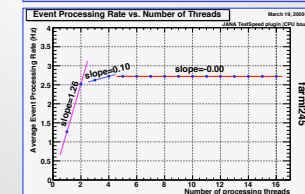
Intel Xeon (3560)  
2.8GHz  
Dual Processors  
8 cores/processor +  
hyperthreading

For this test, each  
hyperthread gave the  
equivalent of 1.9% of a  
full core



AMD Opteron (2352)  
2.1GHz  
Dual Processors  
4 cores/processor with  
**no** hyperthreading

Without  
hyperthreading, a  
modest improvement  
(2% of a core) is  
observed when  
processing threads are  
over-subscribed



Intel Xeon (circa 2004)  
2.8GHz  
Dual Processors with  
1 core/processor +  
hyperthreading

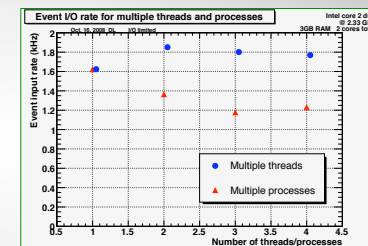
An older machine  
shows hyperthreads  
gaining only about 8%  
of a core.

## Hyperthreading

These plots also show how "hyperthreading" in Intel CPUs can improve performance for CPU bound jobs

## I/O bound Jobs

Multiple processes accessing the same disc leads to competition for the position of the read head. A multi-threaded process can stream events in sequence, dispatching them to individual threads resulting in faster event processing rates for I/O bound jobs as shown in these test results.



Notice: Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177. The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.

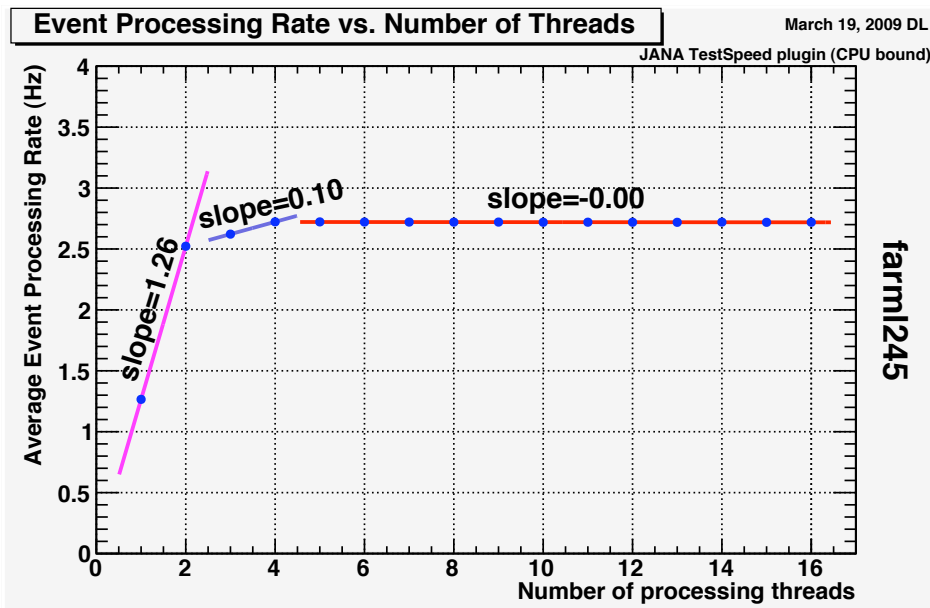
DAVID LAWRENCE david@jlab.org

5/12/09



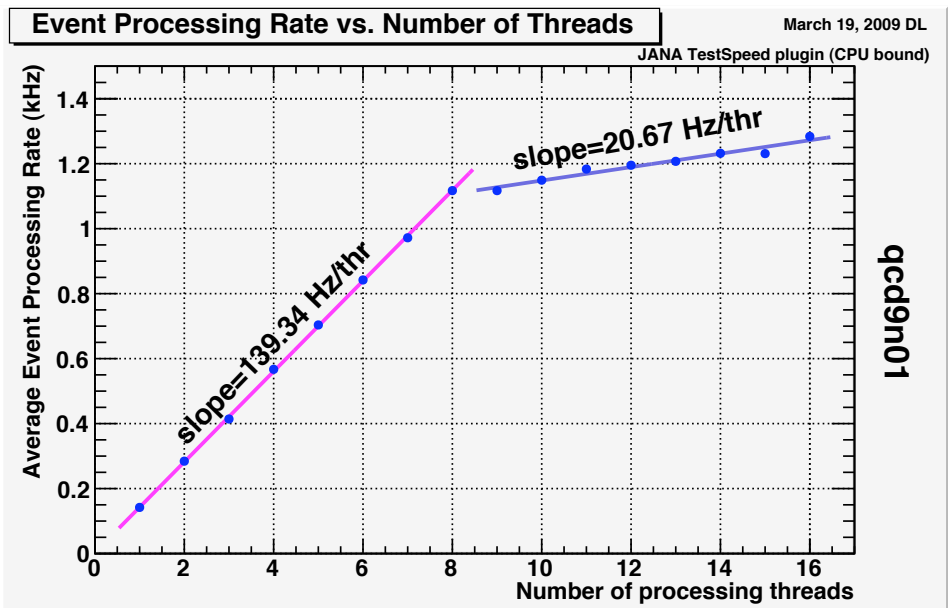
# Hyperthread Scaling

Intel Xeon (circa 2004)  
2.8GHz Dual Processors with  
1 core/processor + hyperthreading



*An older machine shows hyperthreads gaining only about 8% of a core.*

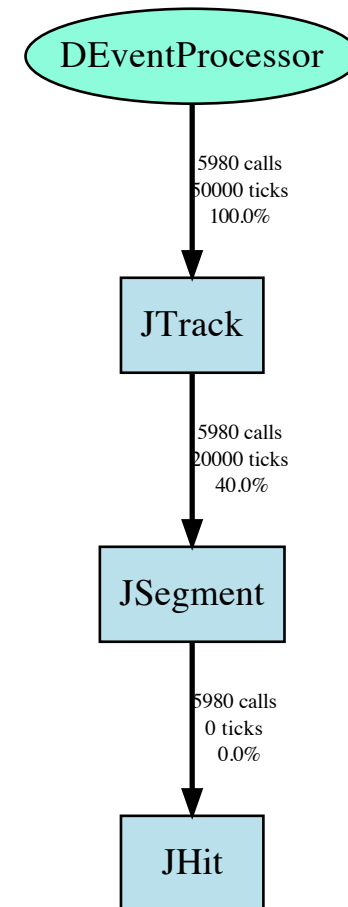
Intel Xeon (5560)  
2.8GHz Dual Processors  
8 cores/processor + hyperthreading



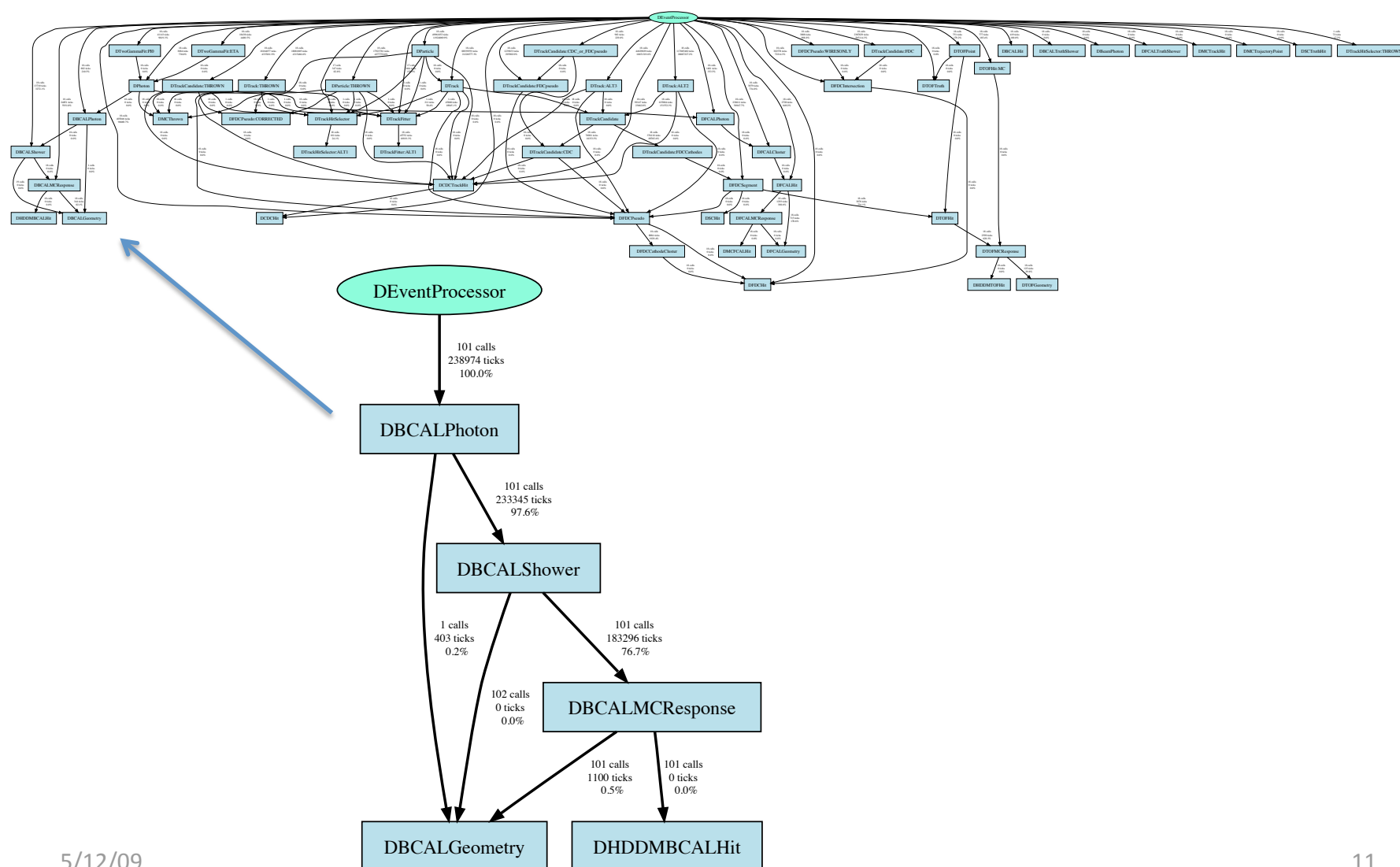
*For this test, each hyperthread gave the equivalent of 15% of a full core*

# Collaboration with Hall-B

- Hall-B continues to work toward a service oriented architecture (SOA)
  - Cyber security issues
  - Unknown performance benefit/deficit
  - Flexibility in choosing language for individual packages
- JANA in Hall-B
  - Early discussions suggested using JANA within *Clara* (the Hall-B SOA project name)
  - Recently, test framework setup by M. Ungaro (~1.5 hrs.)



# GlueX Reconstruction Dependency Graph



# JANA Publications

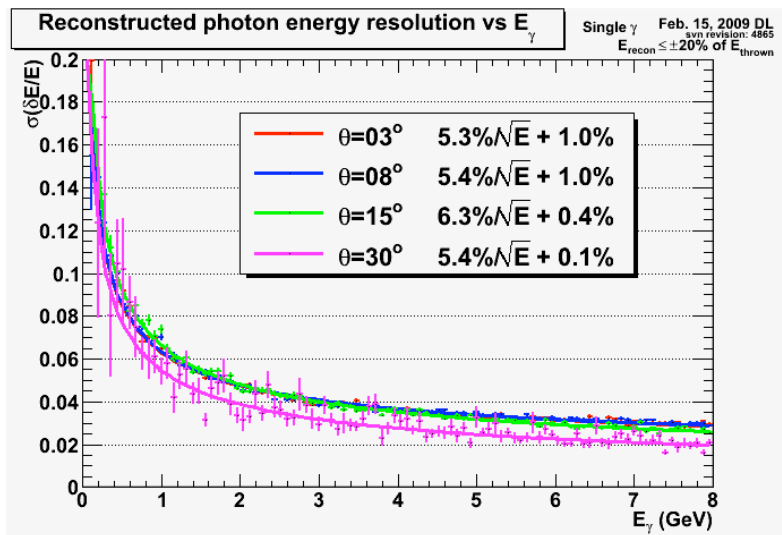
- *Multi-threaded event reconstruction with JANA*  
D. Lawrence 2008 *J. Phys.: Conf. Ser.* **119** 042018 (6pp)  
doi: 10.1088/1742-6596/119/4/042018
- *Multi-threaded event reconstruction with JANA*  
-in process- Proceedings of ACAT08 workshop
- *The JANA calibrations and conditions database API*  
-in development- Proceedings of CHEP09 conference

# The *hdparsim* Project

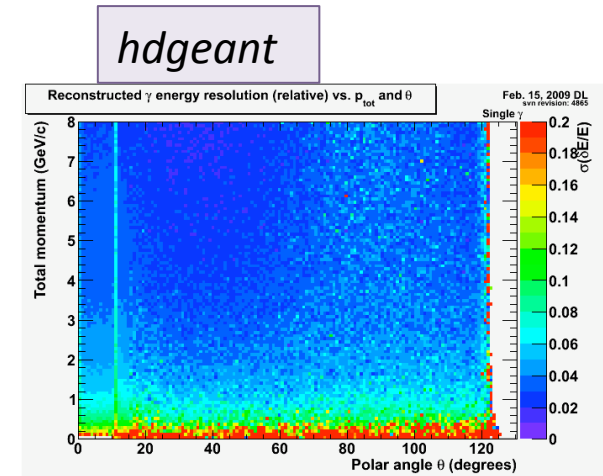
- The *hdparsim* plugin takes tables of energy/momentum resolution, angular resolutions, and efficiencies that are stored in ROOT files and uses them to smear generated values
- Source code is kept here:  
<https://halldsvn.jlab.org/repos/trunk/src/programs/Simulation/plugins/hdparsim>
- Resolution tables are available on the web, and automatically downloaded when the plugin is used.

# Photon Reconstruction

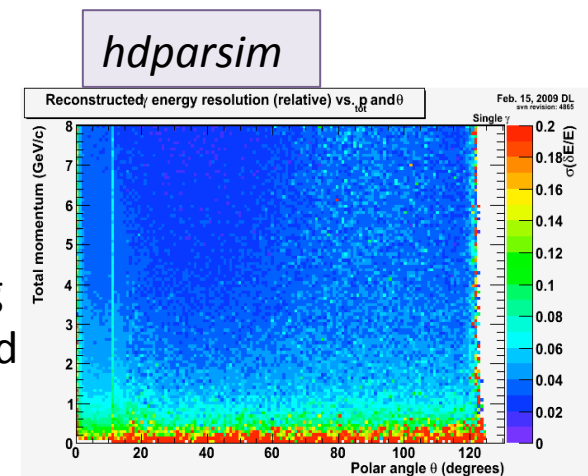
Getting resolutions from simulation with full reconstruction using *DPhoton*



3.2M photons  
simulated and  
reconstructed



100M photons  
parametrically  
simulated using  
*hdgeant* derived  
resolutions



# Performance

- It took about 35 minutes to produce a file of 100M generated events with 1 photon each on my laptop
- It took about 20 minutes to process all 100M events with *hdparsim*
- Charged tracks will take the same amount of time as neutrals since they are indexed and smeared in exactly the same way.
- Charged track simulation reconstruction rates:
  - Simulation (hdgeant): ~44Hz
  - Full reconstruction: ~2-10 Hz
  - Parametric: ~80 kHz

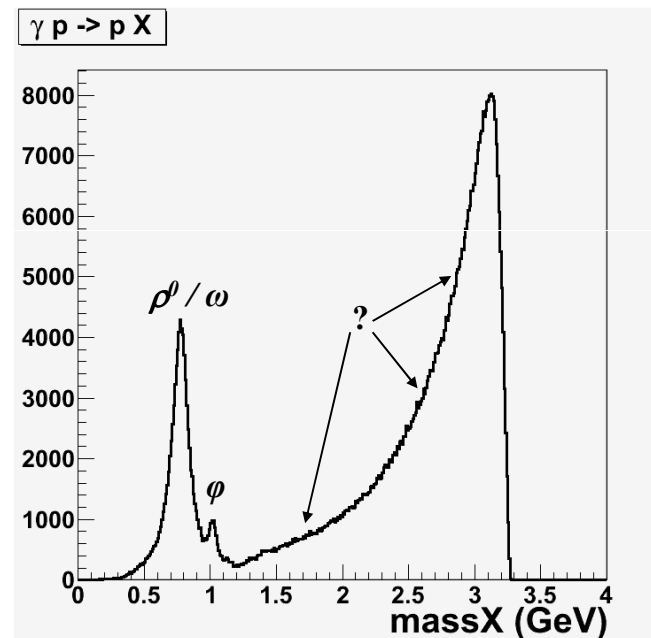
```
264 Feb 15 09:42 run_test.csh
593454704 Feb 15 09:46 genphoton.ascii
1320005967 Feb 15 10:17 output.hddm
660854 Feb 15 10:17 hd_res_photon.root ← downloaded
646808612 Feb 15 10:38 hd_root.root
```

# Using *hdparsim* with pythia generated events

*Slide from Mike Dugger's presentation at April 27 Physics  
Working group meeting*

## Mass $X$ from $\gamma p \rightarrow p X$

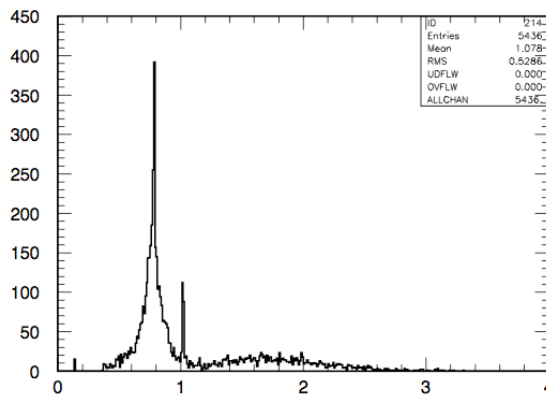
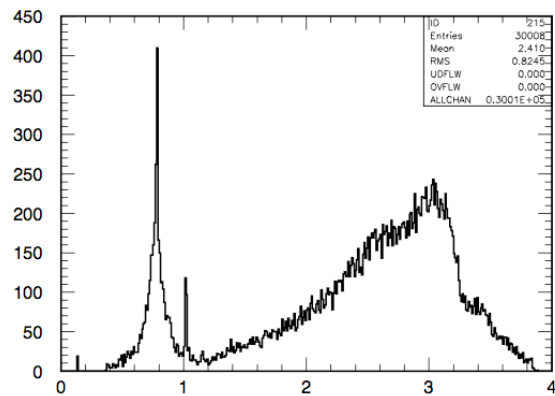
- Vector mesons are clearly visible
- The rest of the background is comprised of a nearly random-looking selection of hadrons



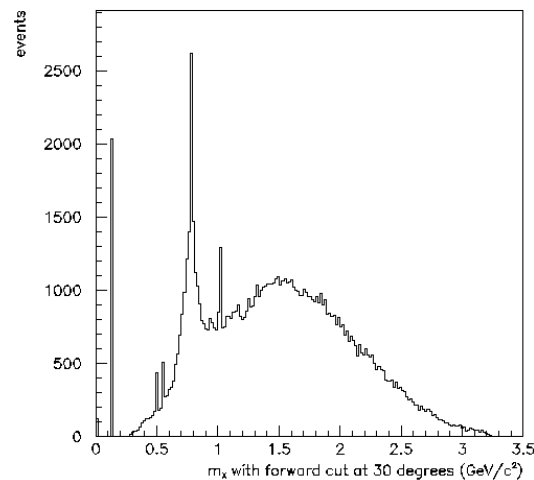
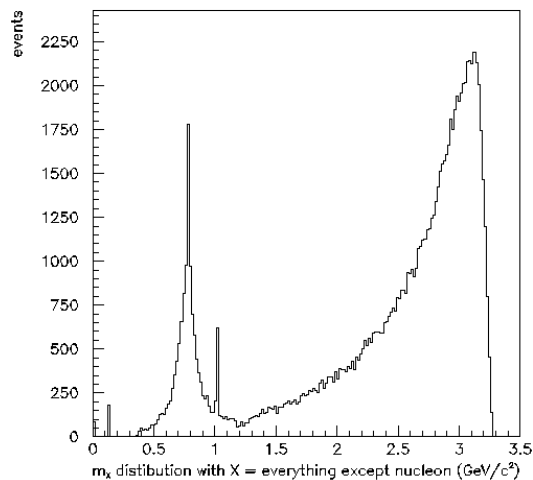


# t-dependence in pythia generated events?

2009/04/28 18.28



Eugene:  $E_\gamma > 6\text{ GeV}$   
Cut on proton being  
produced at vertex



Richard:  $E_\gamma = 8.5\text{--}9.0\text{ GeV}$   
Cut on lab angle ( $30^\circ$ )

# Software Brainstorming on April 22<sup>nd</sup>

--- non-prioritized ---

- Tracking
  - Transition region (*between FDC and CDC*)
  - Kalman
  - Swimming algorithm (*verify consistency with GEANT*)
  - Standard definitions (*use common set of histograms, etc. to compare the 3 tracking codes*)
    - Finding
    - Fitting
  - multi-track events
  - FDC geometry update: Simon
  - Local Lorentz correction for FDC hits
  - CDC geometry update: Beni (*more or less done already*)
  - Alternate tracking philosophies
  - Tracking efficiency over-all: single tracks, multi-tracks
- Simulation
  - Parametric MC
    - Update/expand resolution tables (*need proton table and possibly Kaon table*)
- Miscellaneous
  - New release (*... of Hall-D source code*)
  - Calibration database: firewall penetration (*web service*)
  - Milestone review
  - Reconstruction->PWA interface

# GlueX Software Coordinator

- Congratulations to Mark Ito who is the new GlueX software coordinator!
  - Several nominations for Mark
  - No other nominees
  - Election forgone, Mark declared winner
  - Officially took over on April 22<sup>nd</sup> 2009