The JANA Reconstruction Framework

David Lawrence - JLab

May 25, 2010

What is JANA?

- **J**Lab **ANA**lysis framework (more accurately, a *reconstruction)* framework
- C++ framework that formalizes the organization of algorithms and data transfer for event based processing
- True multi-threaded event processing
- Data-on-demand model
- Numerous features custom for experimental Nuclear Physics

Why Multi-threading?

Multi-core processors are already here and commonly used. Industry has signaled that this will be the trend for the next several years. Consequence: Parallelism is <u>required</u>

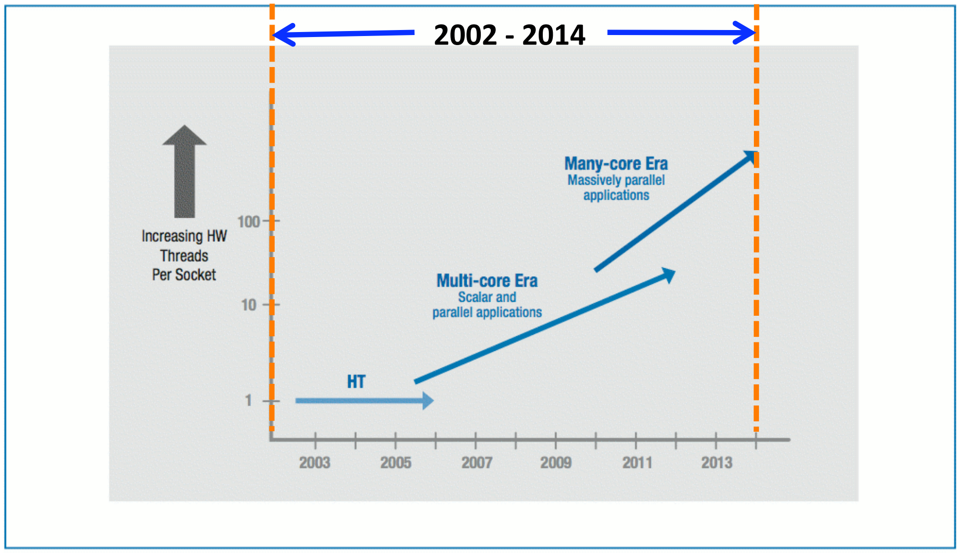
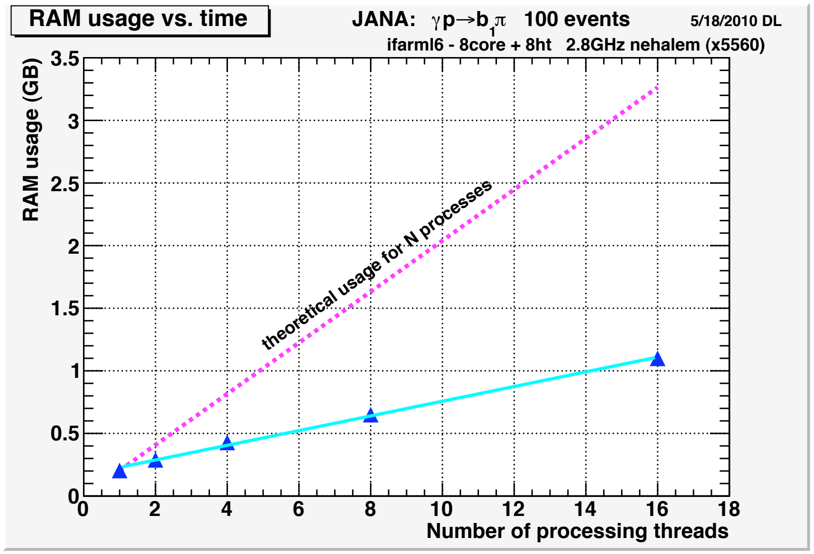
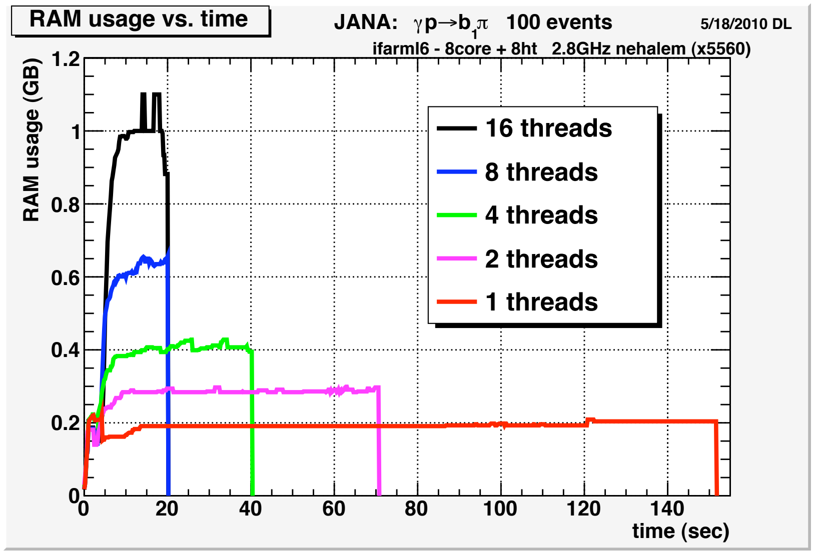


Figure 1: Current and expected eras of Intel® processor architectures

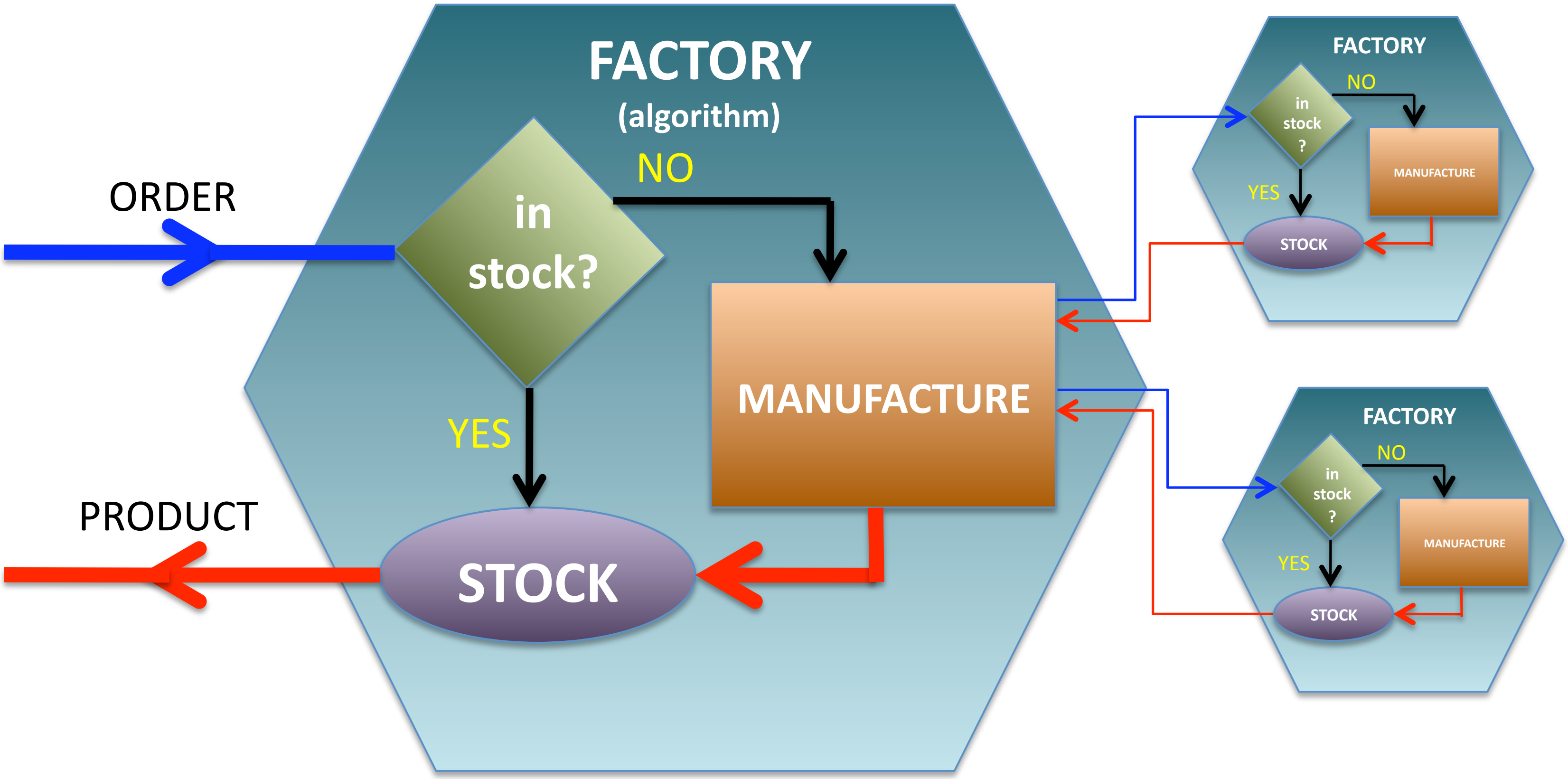




Maintaining a fixed memory capacity per core will become increasingly expensive due to limitations on the number of controllers that can be placed on a single die (#pins).

Example: Memory accounts for about 10%-25% of system cost today but that will increase by as much as 5%/year over the next several years leading to 50% of system cost going toward RAM

Factory Model

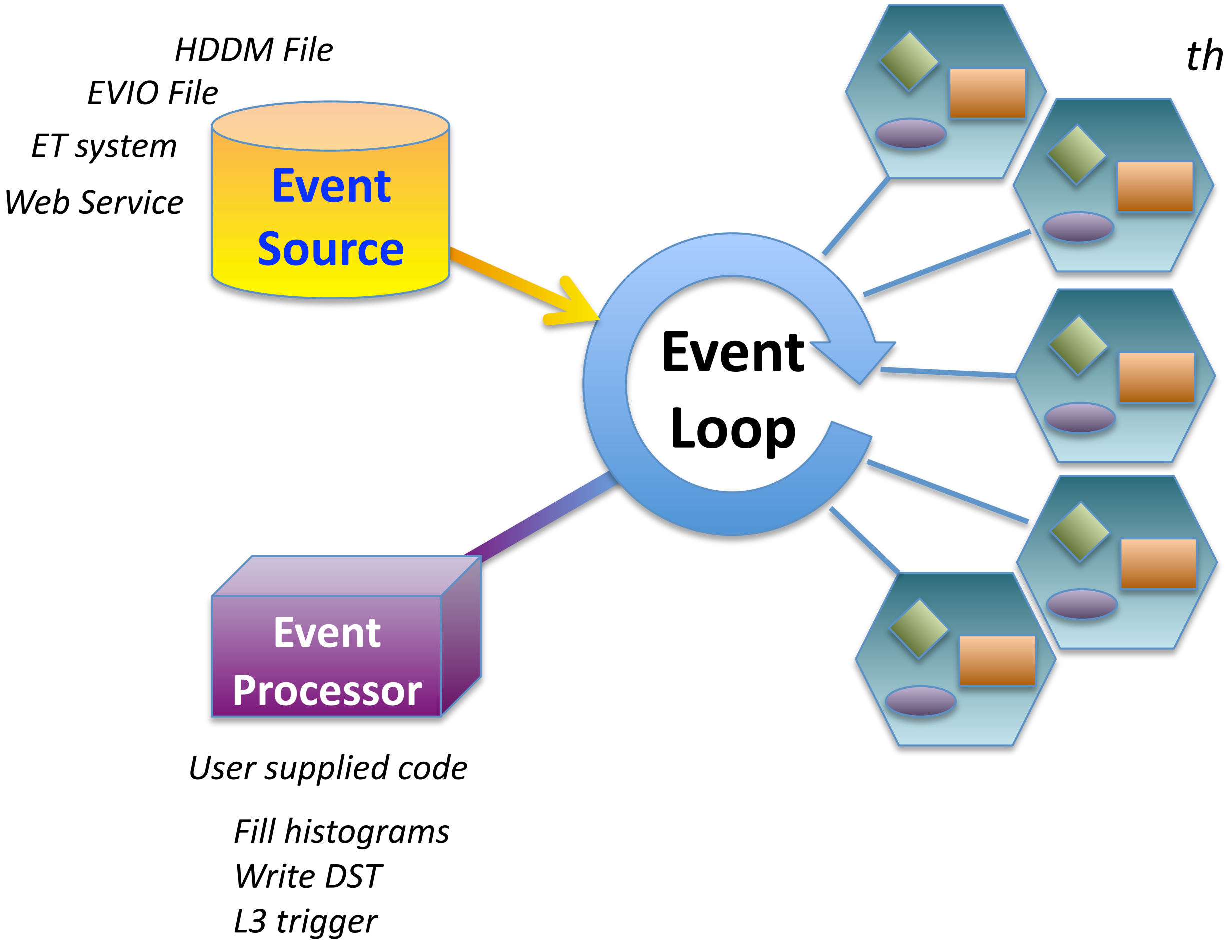


Data on demand = Don’t do it unless you need it

Stock = Don’t do it twice

Conservation of CPU cycles!

Complete Event Reconstruction



Framework has a layer that directs object requests to the factory that completes it

Multiple algorithms (factories) may exist in the same program that produce the same type of data objects

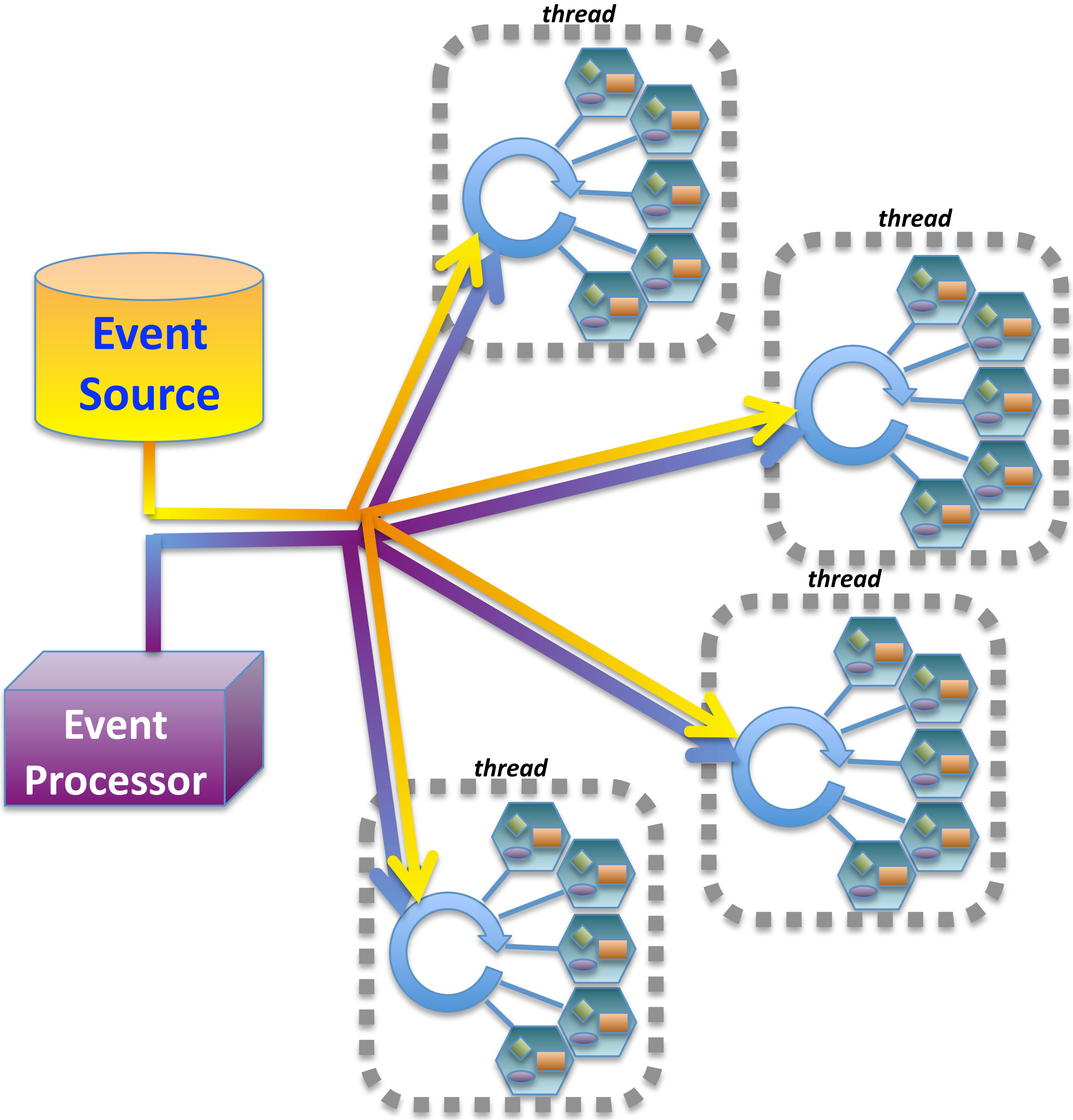
This allows the framework to easily redirect requests to alternate algorithms specified by the user at run time

Multi-threading

○ *Each thread has a complete set of factories making it capable of completely reconstructing a single event*

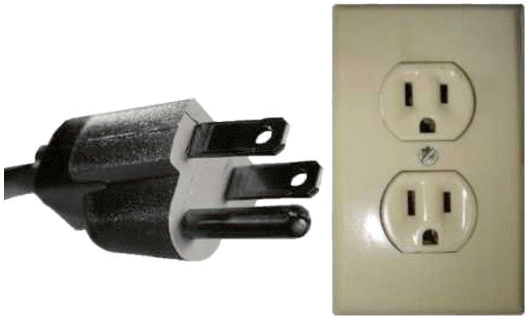
○ *Factories only work with other factories in the same thread eliminating the need for expensive mutex locking within the factories*

○ *All events are seen by all Event Processors (multiple processors can exist in a program)*



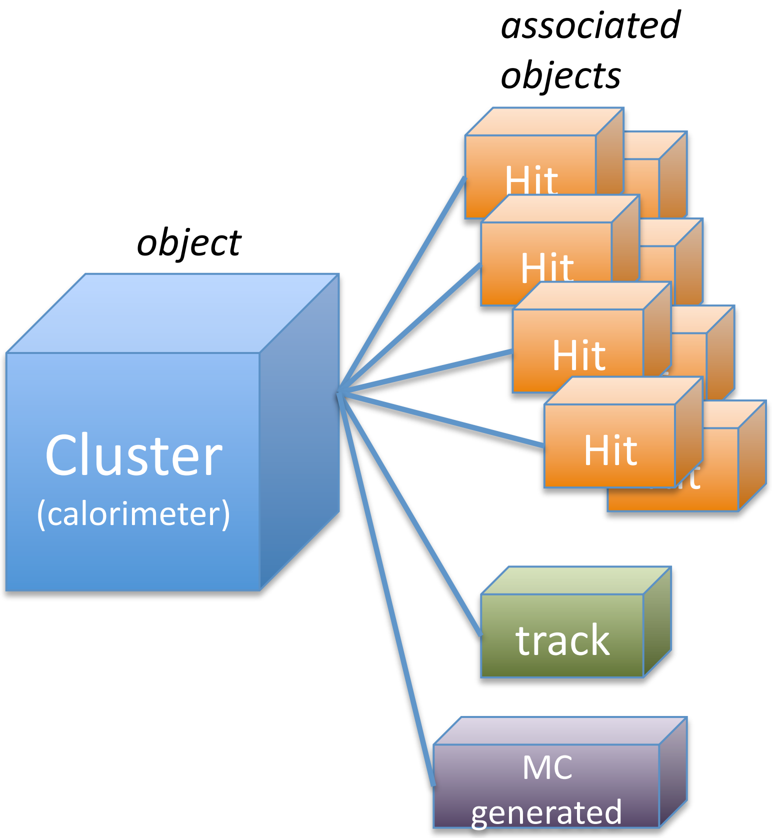
Plugins

- JANA supports plugins: pieces of code that can be attached to existing executables to extend or modify its behavior
- Plugins can be used to add:
  - Event Processors
  - Event sources
  - Factories (additional or replacements)



- Examples:
  - Plugins for creating DST skim files
    - Reconstruction is done once with output to multiple files
    - `hd_ana --PPLUGINS=kaon_skim,ppi+pi-_skim run012345.evio`
  - Plugins for producing subsystem histograms
    - Single ROOT file has histograms from several pieces of code
    - `hd_root --PPLUGINS=bcal_hists,cdc_hists,tof_hists ET:GlueX`

Associated Objects



associated objects

object

Cluster
(calorimeter)

Hit

Hit

Hit

Hit

track

MC generated

○ A data object may be associated with any number of other data objects having a mixture of types

- ○ Each data object has a list of "associated objects" that can be probed using a similar access mechanism as for event-level object requests

```
vector<const DCluster*> clusters;
loop->Get(clusters);
for(uint i=0; i<clusters.size(); i++)
{
     vector<const DHit*> hits;
     clusters[i]->Get(hits);
     // Do something with hits …
}
```

Configuration Parameters

in a factory's init method one might write …

```
MIN_SEED_HITS = 4;
MAX_STEP_SIZE = 3.0; // cm
```

Variables are data members of factory class

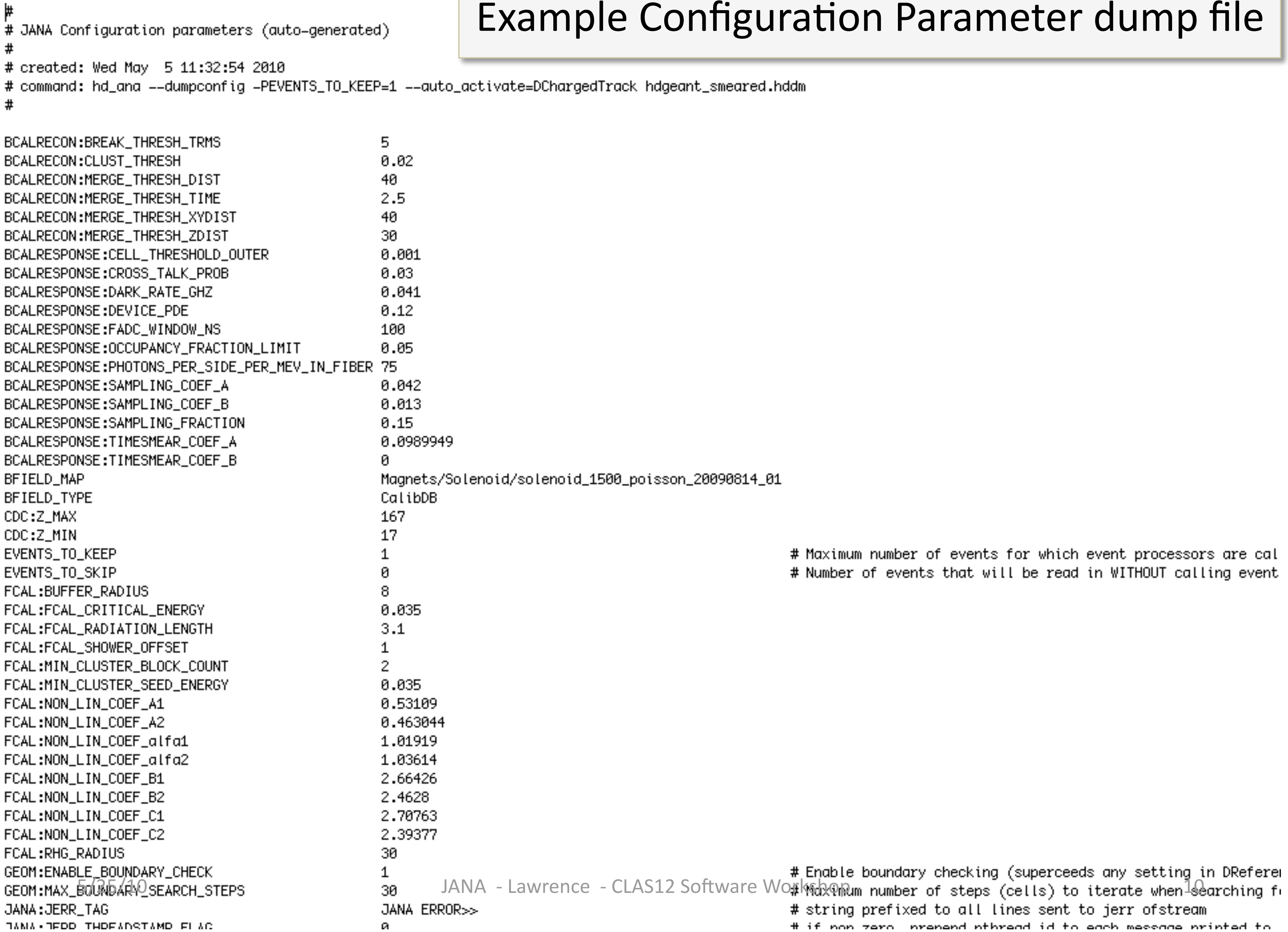
Value may be overwritten if user specifies a value at run time

```
gPARMS->SetDefaultParameter("TRKFIND:MIN_SEED_HITS",MIN_SEED_HITS);
gPARMS->SetDefaultParameter("TRK:MAX_STEP_SIZE" , MAX_STEP_SIZE
    , "Maximum step size in cm to take when swimming a track with adaptive step sizes");
```

***NEW:** Optional 3rd argument allows description to be stored with parameter*

- Parameters can be set via command line or configuration file
  - Complete list of parameters can be dumped using option `--dumpconfig`
    - Parameters can read in using option `--config=`*`filename`*

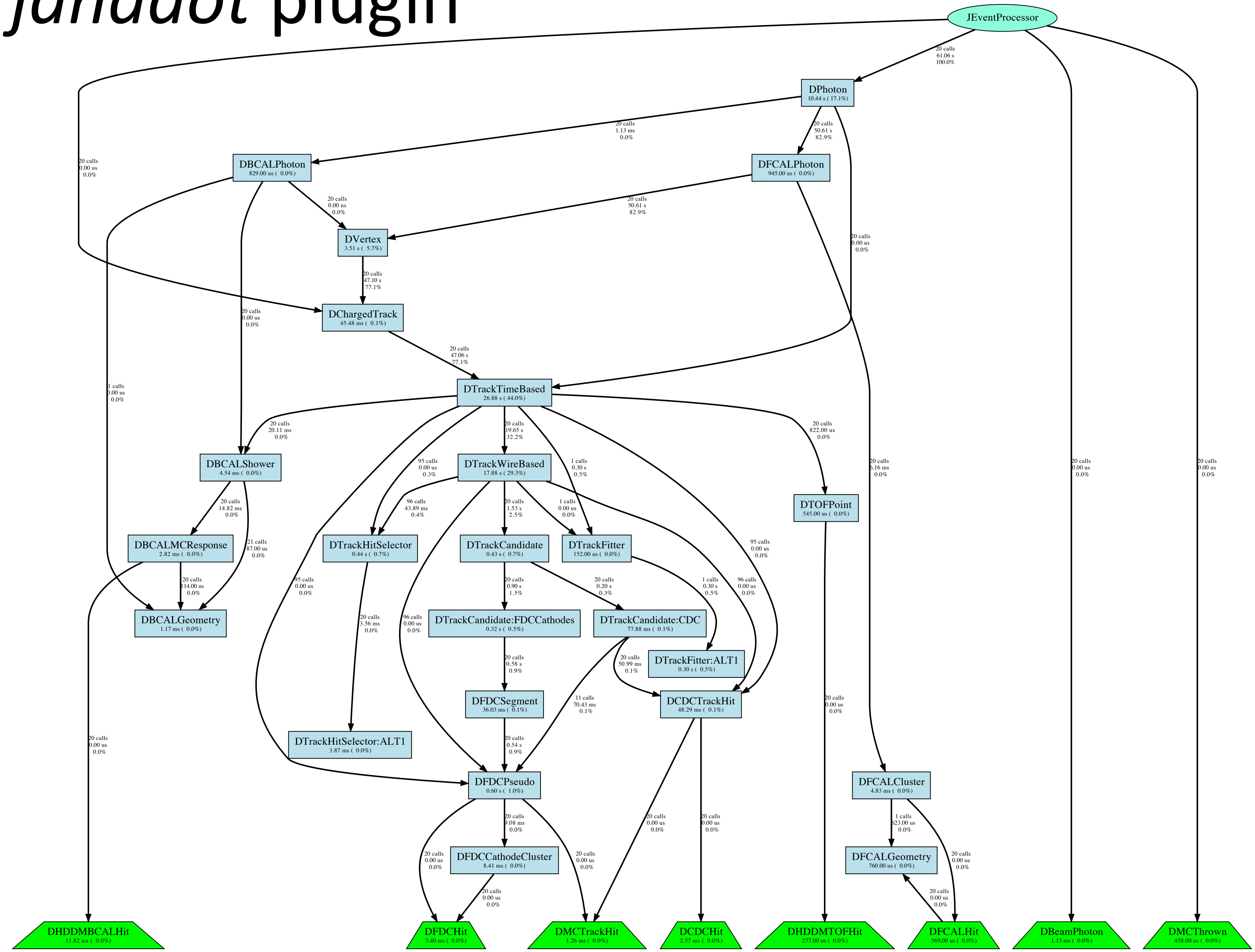
Example Configuration Parameter dump file



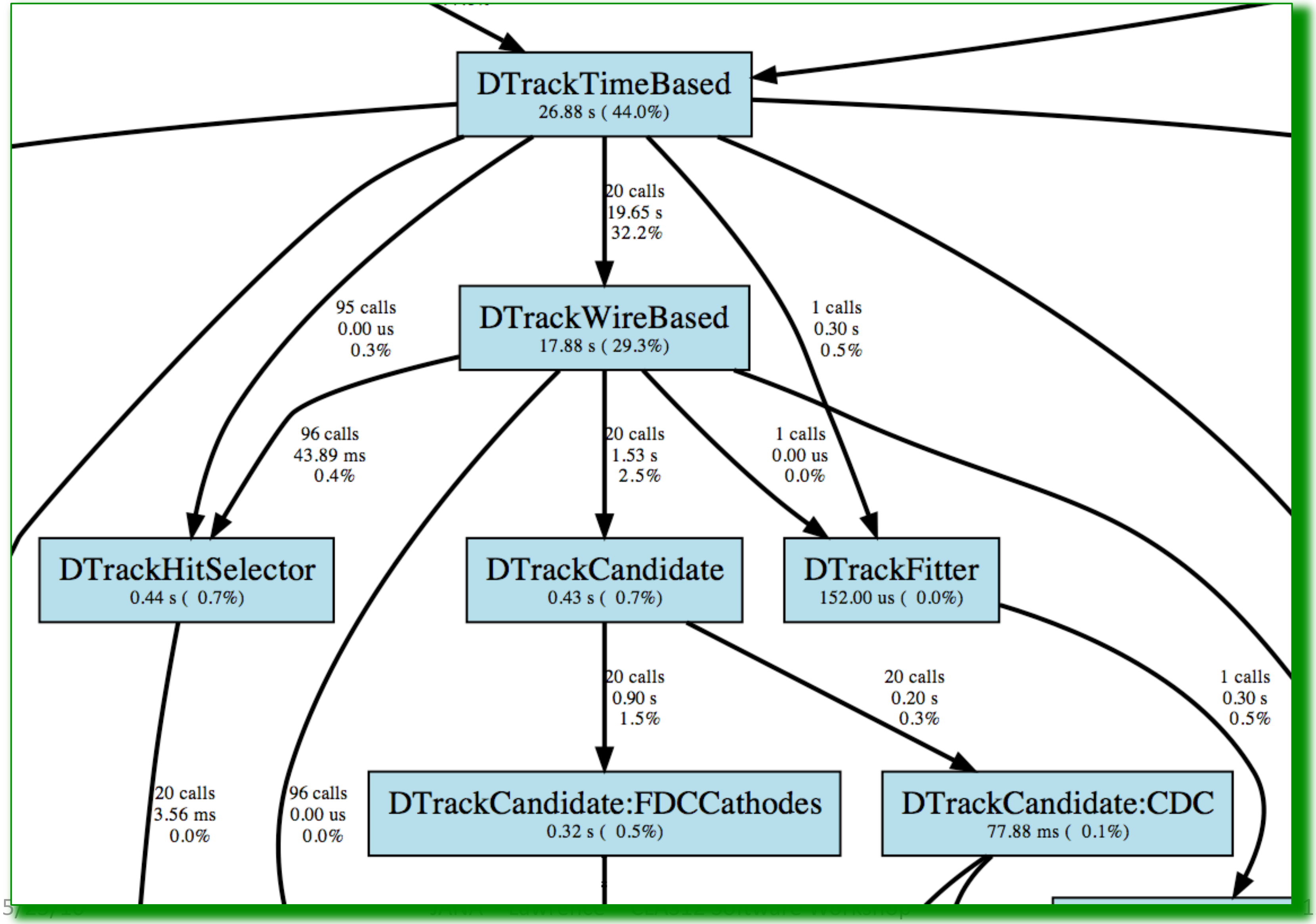
```
#
# JANA Configuration parameters (auto-generated)
#
# created: Wed May  5 11:32:54 2010
# command: hd_ana --dumpconfig -PEVENTS_TO_KEEP=1 --auto_activate=DChargedTrack hdgeant_smeared.hddm
#

BCALRECON:BREAK_THRESH_TRMS                    5
BCALRECON:CLUST_THRESH                         0.02
BCALRECON:MERGE_THRESH_DIST                    40
BCALRECON:MERGE_THRESH_TIME                    2.5
BCALRECON:MERGE_THRESH_XYDIST                  40
BCALRECON:MERGE_THRESH_ZDIST                   30
BCALRESPONSE:CELL_THRESHOLD_OUTER              0.001
BCALRESPONSE:CROSS_TALK_PROB                   0.03
BCALRESPONSE:DARK_RATE_GHZ                     0.041
BCALRESPONSE:DEVICE_PDE                        0.12
BCALRESPONSE:FADC_WINDOW_NS                    100
BCALRESPONSE:OCCUPANCY_FRACTION_LIMIT          0.05
BCALRESPONSE:PHOTONS_PER_SIDE_PER_MEV_IN_FIBER 75
BCALRESPONSE:SAMPLING_COEF_A                   0.042
BCALRESPONSE:SAMPLING_COEF_B                   0.013
BCALRESPONSE:SAMPLING_FRACTION                 0.15
BCALRESPONSE:TIMESMEAR_COEF_A                  0.0989949
BCALRESPONSE:TIMESMEAR_COEF_B                  0
BFIELD_MAP                                     Magnets/Solenoid/solenoid_1500_poisson_20090814_01
BFIELD_TYPE                                    CalibDB
CDC:Z_MAX                                      167
CDC:Z_MIN                                      17
EVENTS_TO_KEEP                                 1                                                  # Maximum number of events for which event processors are cal
EVENTS_TO_SKIP                                 0                                                  # Number of events that will be read in WITHOUT calling event
FCAL:BUFFER_RADIUS                             8
FCAL:FCAL_CRITICAL_ENERGY                      0.035
FCAL:FCAL_RADIATION_LENGTH                     3.1
FCAL:FCAL_SHOWER_OFFSET                        1
FCAL:MIN_CLUSTER_BLOCK_COUNT                   2
FCAL:MIN_CLUSTER_SEED_ENERGY                   0.035
FCAL:NON_LIN_COEF_A1                           0.53109
FCAL:NON_LIN_COEF_A2                           0.463044
FCAL:NON_LIN_COEF_alfa1                        1.01919
FCAL:NON_LIN_COEF_alfa2                        1.03614
FCAL:NON_LIN_COEF_B1                           2.66426
FCAL:NON_LIN_COEF_B2                           2.4628
FCAL:NON_LIN_COEF_C1                           2.70763
FCAL:NON_LIN_COEF_C2                           2.39377
FCAL:RHG_RADIUS                                30
GEOM:ENABLE_BOUNDARY_CHECK                     1                                                  # Enable boundary checking (superceeds any setting in DRefere
GEOM:MAX_BOUNDARY_SEARCH_STEPS                 30                                                 # Maximum number of steps (cells) to iterate when searching f
JANA:JERR_TAG                                  JANA ERROR>>                                       # string prefixed to all lines sent to jerr ofstream
JANA:JERR_THREADSTAMP_FLAG                     0                                                  # if non zero, prepend pthread id to each message printed to
```

janadot plugin



A closer look at *janadot*



Summary

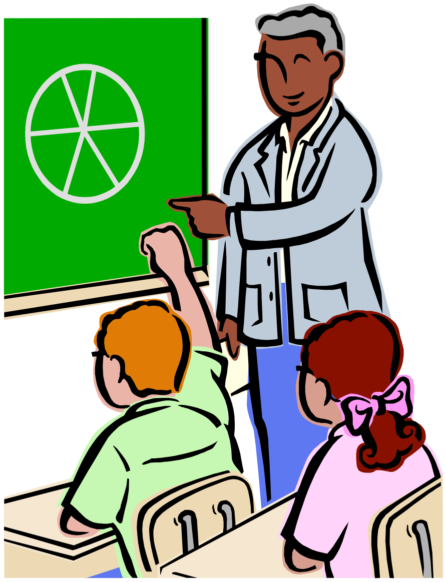
- JANA is a multi-threaded event reconstruction framework
- Designed to optimize CPU usage via models such as data-on-demand and object sharing
- Support for alternative algorithms and plugins make the framework highly configurable to end users as well as developers
- Built-in features *(janadot)* allow for both profiling and easily documenting

… and in closing …

- Additional features:
  - *janaroot* plugin for auto-magical generation of ROOT trees from data objects
  - *janactl* plugin allows for remote control and monitoring of process health via cMsg
  - Calibration/Conditions DB interface with working Web Service implementation (doi: 10.1088/1742-6596/219/4/042011)
  - XML geometry interface with integrated *Xpath* parser
  - Thread safe stream logging support

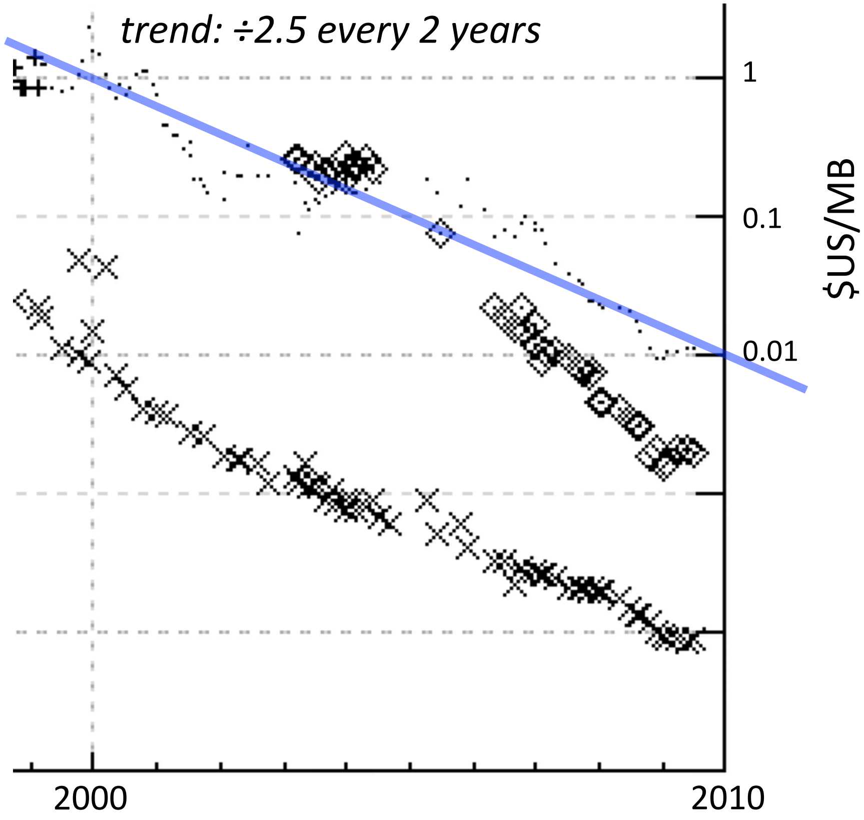
http://www.jlab.org/JANA doi: 10.1088/1742-6596/119/4/042018

Backup Slides



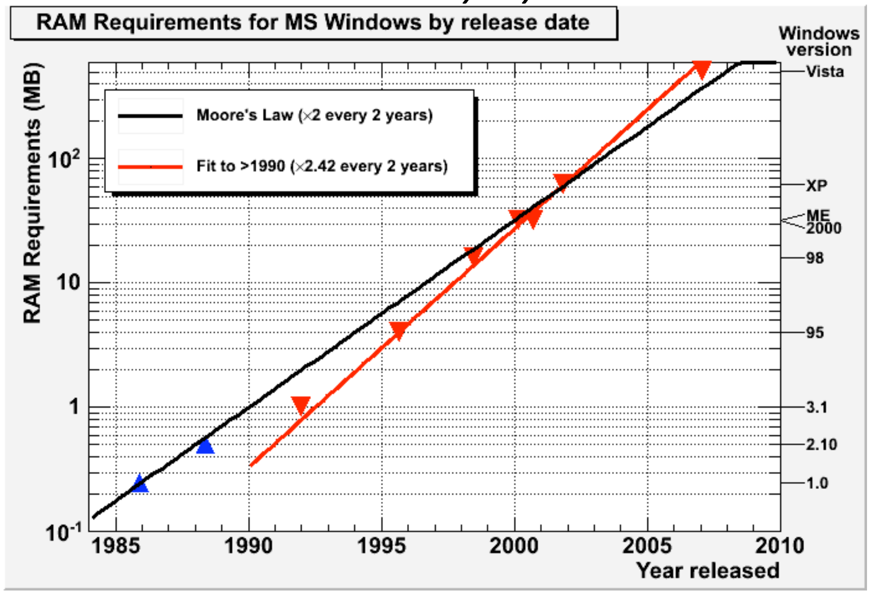
Memory Prices

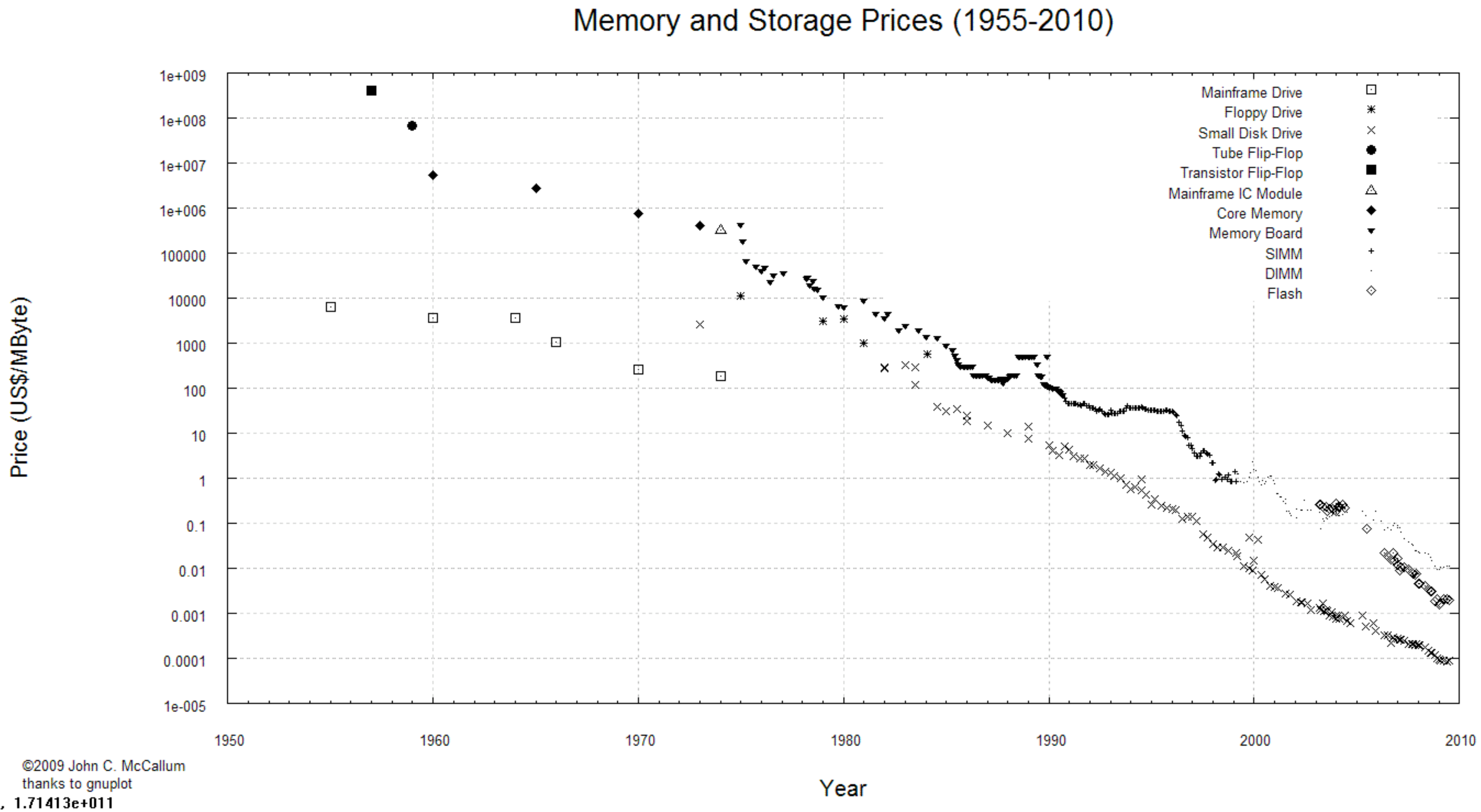
trend: ÷2.5 every 2 years



Memory Requirements

trend: ×2.4 every 2 years





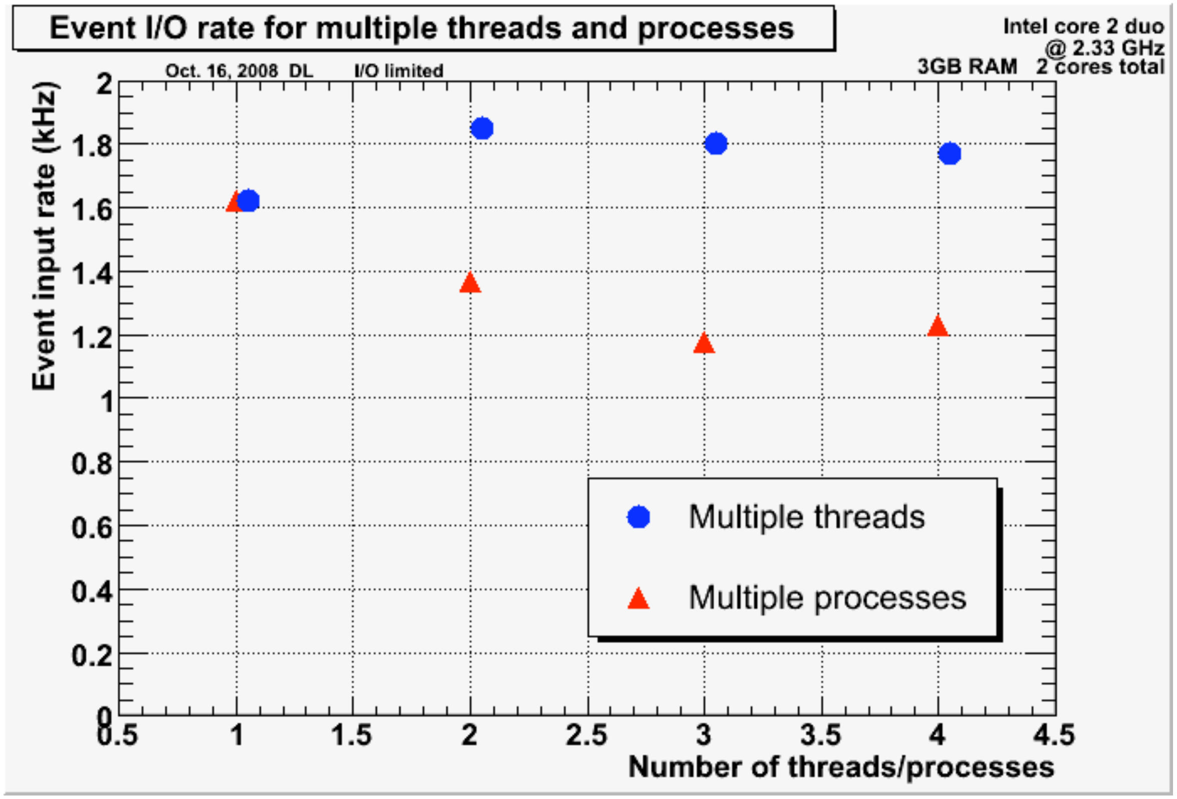
©2009 John C. McCallum
thanks to gnuplot

1937.77, 1.71413e+011

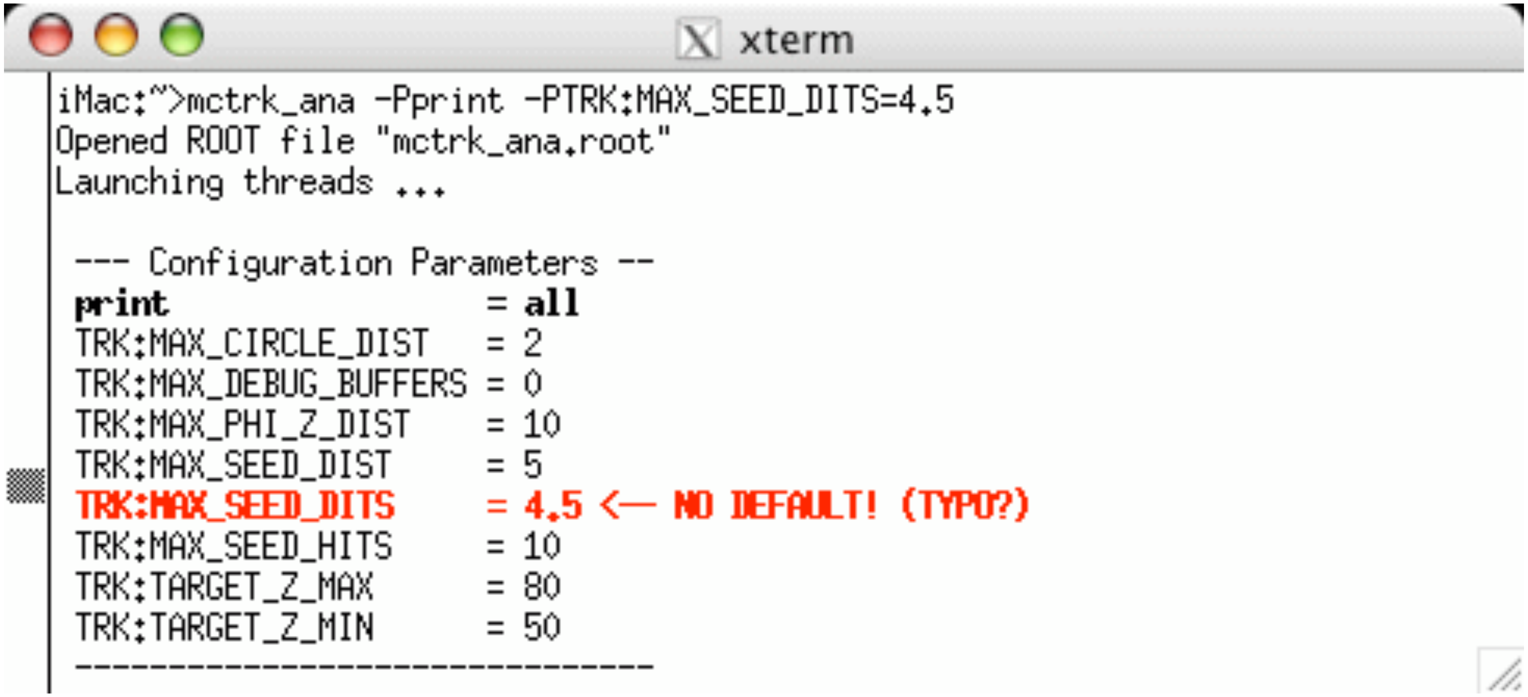
I/O Scaling

Multiple processes simultaneously reading and writing to the same local disk will cause the disk head to thrash, ultimately leading to an I/O bottleneck

Multiple threads will stream events from a single file leading to much less competition for the head position



Configuration Parameters



```
iMac:~>mctrk_ana -Pprint -PTRK:MAX_SEED_DITS=4.5
Opened ROOT file "mctrk_ana.root"
Launching threads ...

 --- Configuration Parameters --
 print                   = all
 TRK:MAX_CIRCLE_DIST     = 2
 TRK:MAX_DEBUG_BUFFERS   = 0
 TRK:MAX_PHI_Z_DIST      = 10
 TRK:MAX_SEED_DIST       = 5
 TRK:MAX_SEED_DITS       = 4.5 <— NO DEFAULT! (TYPO?)
 TRK:MAX_SEED_HITS       = 10
 TRK:TARGET_Z_MAX        = 80
 TRK:TARGET_Z_MIN        = 50
 -------------------------------
```