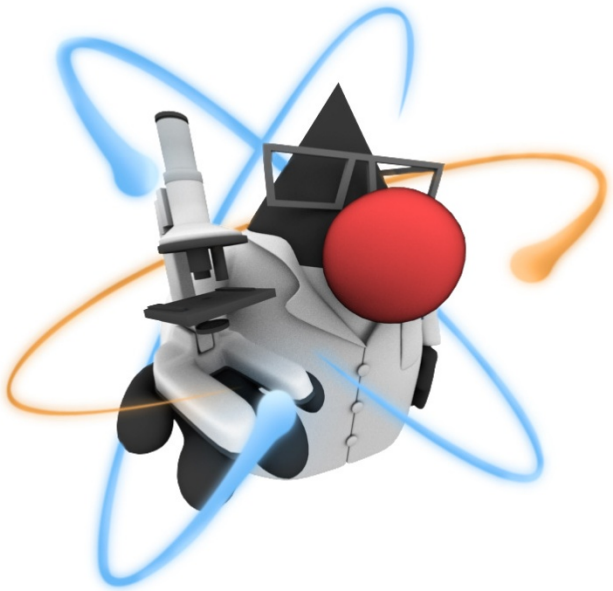
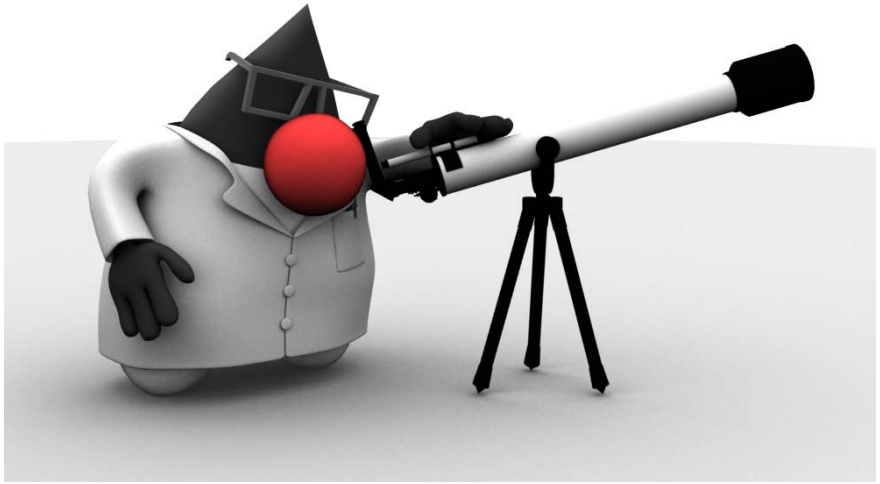
Use of Java in High-Energy and Astro-Physics





ACAT 2008

Tony Johnson

SLAC
NATIONAL ACCELERATOR LABORATORY

Overview

- Java Overview
  - Why should Java be of interest to HEAP
    - Recent additions to language
    - Open-Source Java
    - Other JVM languages
- Examples of use in Astro/HEP
  - Fermi LAT (previously GLAST)
    - Data Monitoring, Data Servers, WIRED, GLAST Pipeline
  - International Linear Collider Detector Development
    - Fully Java reconstruction and analysis framework
- Obstacles to Java Adoption in HEP
- Links to Open-Source Java libraries/tools

Not going to cover…

- This talk is mainly based on my personal experiences in using Java for HEP since the language came to public attention back in 1995.
- I have not made any attempt to describe all usage of Java in HEAP
  - Many uses of Java in accelerator controls systems at CERN, SLAC, Fermilab, elsewhere
  - CERN itself won a “Duke Award” presented at annual Java One conference for “entire collection of Java applications that CERN has developed for the installation and operation of the LHC”
    - EDH, GraXML, Technical Infrastructure Monitoring (TIM), GridPP real-time monitoring (RTM)
    - Many more such as EVO, …

Questions

The organizers of this session posed several questions they hoped I would address (paraphrased by me below):

- Given the fact that Java has invaded so many areas of computing, how do you explain that in our field the language is nearly invisible?
- All LHC experiments are now fighting with memory constraints, trying to run in 32 bit mode on 64 bits machines below the 2 GBytes threshold for economical reasons. Most reports claim that Java programs consume more memory. What is your experience with that?
- How does Java perform in realistic HEP environments (as opposed to micro benchmarks which are often cited but difficult to interpret).
- What is the distribution Java vs. C++ in experiments you are familiar with?
- What is your view on the growing importance of parameterized types in C++, also available at a lower scale in Java now.
- Do you see the situation being inverted in the coming years? What would be the key parameters that could make this happen?

Hopefully you will find some answers to these questions embedded in this talk

Why Java for HEAP?

- Java is a pure Object Oriented Language
- Simpler to learn and use than C++
  - Language design emphasizes ease-of-use, programmer productivity
    - Not hampered by historical baggage backwards compatibility with C
  - Lack of direct access to pointers eliminates large source of common errors in C++
    - Especially for less expert developers
  - Garbage collector takes care of freeing unused objects
    - Avoids many common programming errors
    - Avoids distorting OO design by removing need for “ownership” of Objects
  - Very powerful standard libraries build-in
    - Cross-platform GUI development
    - Huge number of open-source libraries
    - Libraries for scientific computing
      - Apache commons-math, JSci, FreeHEP, …
  - Physicist gets to concentrate on writing clean OO code to perform analysis tasks
    - Not understanding core dumps and learning difference between a pointer and a reference.
- Java is increasingly a (maybe the) mainstream OO language
  - Taught in many university courses
  - Overtaken C++ in popularity for “open-source” projects
  - Very widely used especially in the areas of
    - Web application development
    - Graphical enterprise applications
    - Other fields of science, especially astronomy, biology, …

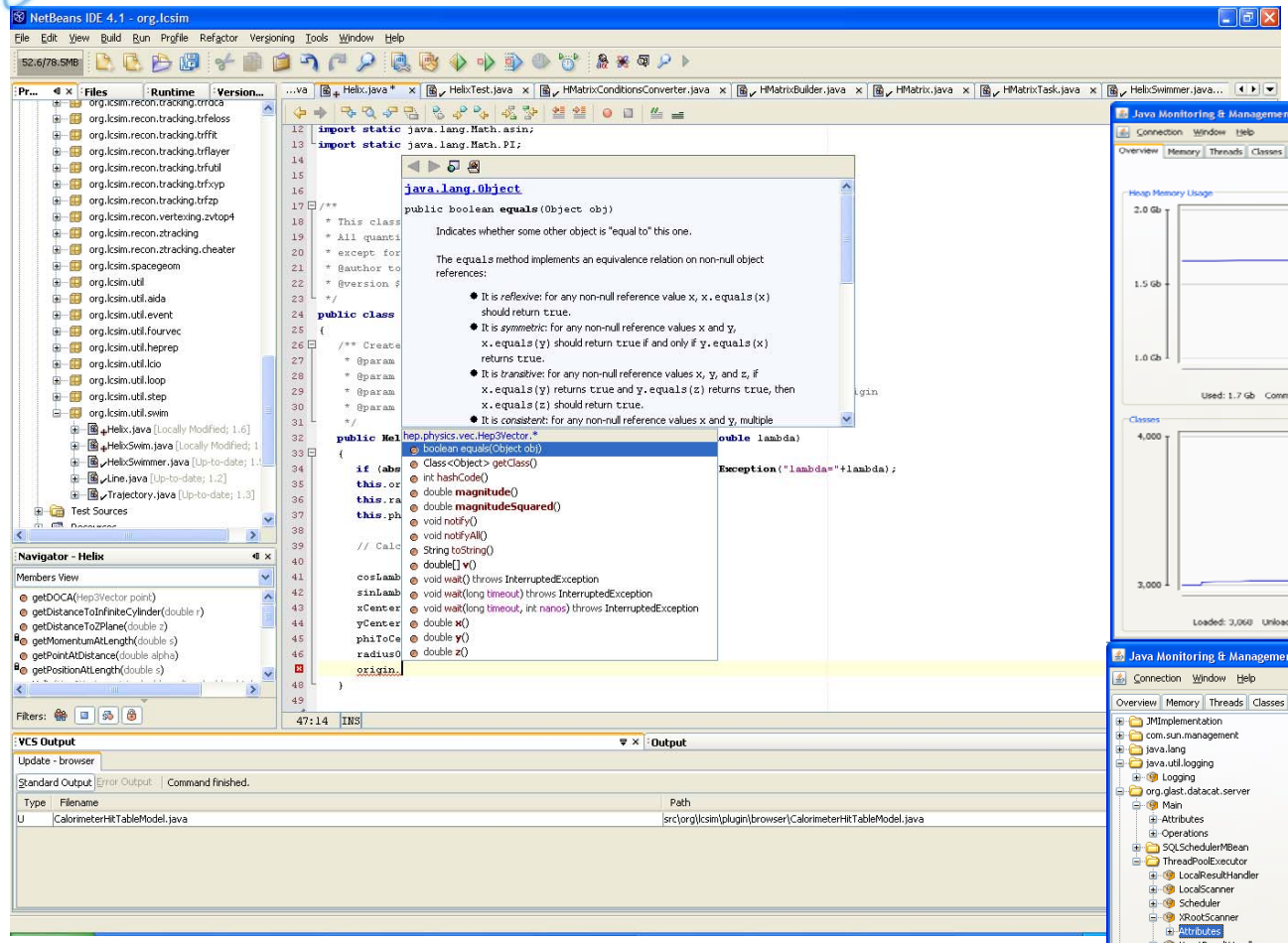
Why Java for HEAP?

- Platform independent, compile once just runs everywhere
  - Linux, Windows, Mac OSX
    - Saves a lot of time when supporting code on many platforms
  - Makes possible tools such as Java WebStart where user clicks a button on a web page to start an application with no need to have pre-installed any software.
- Full runtime access to information about classes (methods, member variables *etc*)
  - Directly usable by scripting languages, analysis toolkits, IO
  - Replaces need to invent these mechanisms ourselves (c.f. CINT dictionary)
- Performance of Java code is close to that of C++
  - Although Java is initially compiled to machine independent "byte-codes" these are converted to machine code at runtime
  - Dynamic (runtime) optimization can take into account actual usage patterns
    - Not available to static optimizers used by Fortran, C++
  - Garbage collection often more efficient than user written malloc/free (or new/delete)
  - Many benchmarks available on the web
    - Some show C++ faster than Java, others show Java faster than C++
    - Our experience is that overhead of garbage collection and factors like array bounds checking makes (well-written) Java slightly slower than (well-written) C++, but overhead is typically small
    - Often raw performance is irrelevant compared to savings in development time
      - Which in turn can lead to cleaner more optimized code to begin with

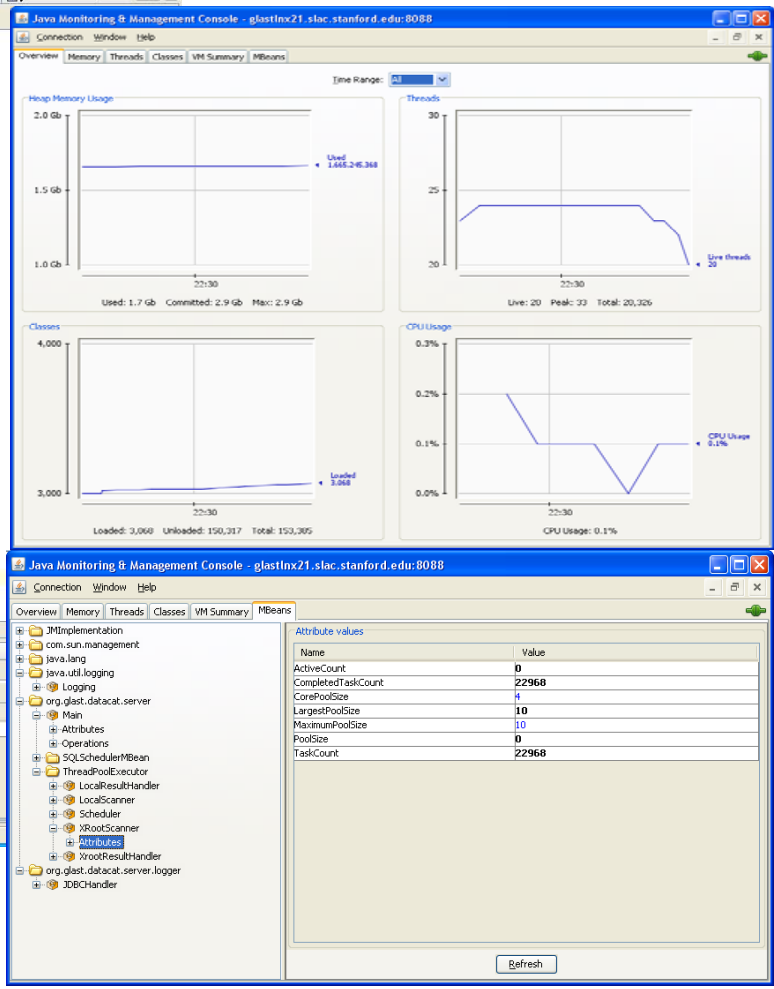
Why Java for HEAP

- Excellent tools (in many cases free)
  - IDE's
    - Eclipse, Netbeans, IDEA, …
    - Typically integrate editing, code completion, documentation viewing, refactoring, debugging, WYSIWYG GUI development, performance analysis, …
    - These IDE's now support many languages including C++
      - Java has lead the way in excellent IDE support
  - JMX
    - Ability to connect to running program to view statistics, memory usage and control program execution,
    - Perform memory and thread dumps on running programs
    - Support for user defined "mbeans" for dynamic access to program functionality
  - Build tools, ant, maven
    - Maven allows a project to be build from source with a single command that:
      - Downloads correct versions of all dependencies
      - Compiles the code
      - Runs unit and integration tests
      - Deploys library to allow it do be used as dependency for other projects
    - Also allows web site, documentation and reports to be generated and deployed in a single command
    - Configured with a single declarative project description (in XML)

Why Java for HEAP



Netbeans IDE



Jconsole

Open Source Java

- Although full source has always been available from Sun, only via restricted “research” license.
- Since 2007 full source code available under GPL.
  - “Classpath exception” exempts programs which use GPL Java from themselves having to be GPL’d.
  - Initially small amount of code for which Sun did not own full rights were provided only as binary “plugs”
    - RedHat produced 100% GPL release “IcedTea” with open-source replacements for restricted code
    - Sun and RedHat cooperating on getting these changes into the main branch
  - Open source Java endorsed by Richard Stallman (author of “The Java Trap”).
  - Even Java mascot “duke” available under open-source license
  - Java name and is still reserved for implementations that have passed the Java Compatibility Tests. These tests are not themselves open-sourced, although Sun has committed to making them available to open-source implementations.
  - Sun continues to also sell commercial Java licenses to OS/phone vendors
- Java beginning to appear as core part of many Unix distributions (e.g. Ubuntu 8.04)

Parameterized Types for Java

- Prior to Java 1.5 there was no support for parameterized types (c.f. C++ templates)
  - Need for many casts, especially when removing items from collections
    - Which could in turn lead to runtime exceptions due to incorrect casts
      - Rarely a problem in practice, but did make code somewhat ugly
- Java 1.5 (2004) added support for parameterized types (called generics)
  - Syntactically similar to C
    - Map<Track, Clusters> trackClusterMap = event.getTrackClusterMap();
    - for (Track t : trackClusterMap.keySet()) {
    - }
    - Java implementation maintains static typing, compile time checking
  - At the same time support for auto boxing and unboxing of primitives added
    - Syntax for adding primitives to collections becomes nicer
- These new features were somewhat controversial when they were added:
  - They do add some subtleties that some people thought Java could do without
  - Generics are a compiler feature, the information on the generic type is not available at runtime via introspection
    - There are no runtime checks on correct types when putting items into collections
  - I don't recall complaints that were not sufficiently like C++ templates

Other new features of Java

- Concurrency framework
  - Java has always had build-in support for threading, locks
  - All Java libraries are thread safe
    - (or clearly documented where there are restrictions)
  - Still the case that writing multi-threaded code (correctly) is very tricky
  - Concurrency library now a standard part of Java makes multi-threaded programs much easier to write
    - Provides well tested and general classes for simplifying concurrent programming
    - Very useful for taking advantage of multi-core machines
      - For example analyzing multiple events in parallel
- Java NIO framework
  - Makes it possible to write highly performant network and file IO libraries
    - Non blocking IO, memory mapped files, …

… other Java VM languages

- Many other languages have been ported to run on the Java VM
- “Dynamically languages”
  - Useful for quick tests and data analysis
  - All automatically gain full access to Java libraries and full access to objects created by those libraries (via reflection)
  - Ports of existing languages to JVM
    - JRuby, Jython
  - Developed from scratch for JVM
    - Pnuts, Groovy
- Scala
  - Compilable, statically typed language
    - 100% compatible with Java, JVM and existing libraries
  - Fully Object-Oriented
    - no distinction between primitives/objects as in Java
  - Smoothly integrates features of functional/OO languages
  - Supports used define types and operator overloading
  - Relatively new, but looks very interesting if you “must have” all the latest cool language features but want the performance/compatibility/portability of Java
  - http://scala-lang.org/

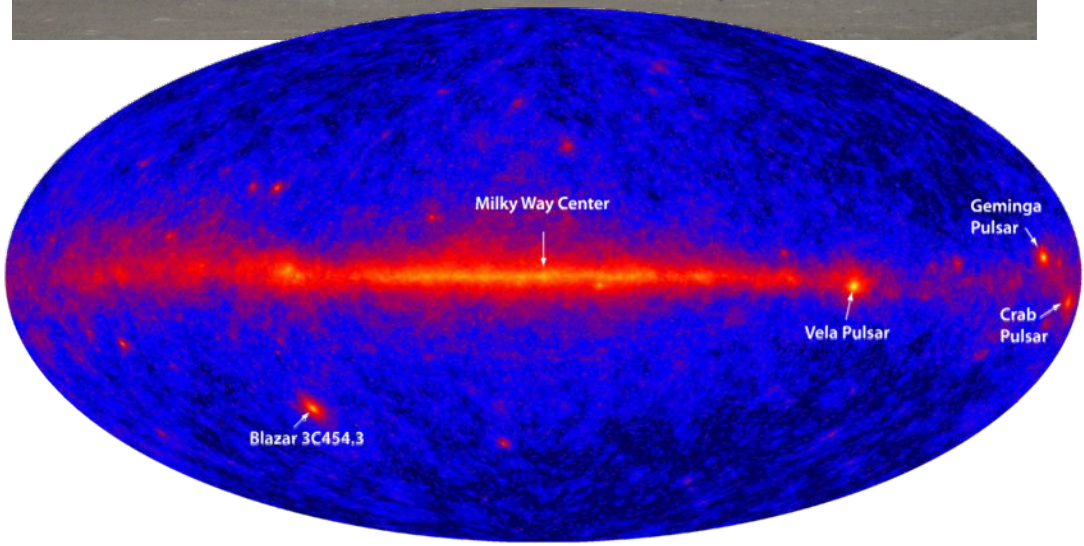
Examples of Java Use in HEAP experiments

Fermi LAT (formerly GLAST)

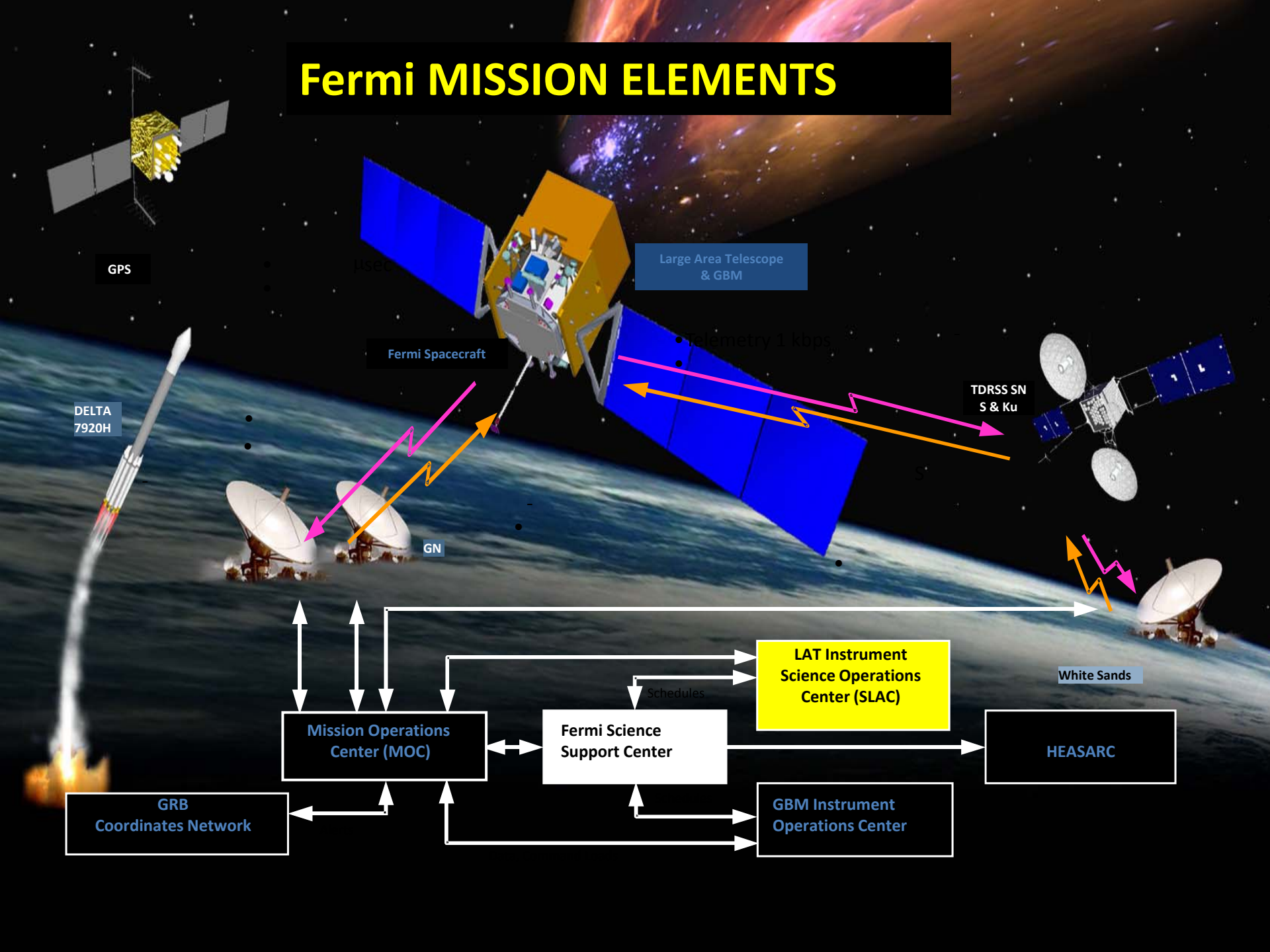
International Linear Collider Detector R&D.

Fermi LAT (previously GLAST)

- Gamma-Ray Telescope launched in June 2008



Fermi MISSION ELEMENTS



Science Processing Software Overview

- C++ Simulation/Reconstruction based on Geant4, Gaudi, Root
- Intermediate (digi, recon, mc) data stored as Root files, data presented to analysis users as FITS files. Most science analysis tools written in Python or C++ using FITS files.
- Data storage uses Scalla (xrootd). Total data volume stored ~10TB/month, but only about 1GB/month distilled “astronomy” data.
- Monitoring of data quality, data access, event display and control of data processing performed via Java web applications.

	Test Name	Date	CPU (secs)	Memory (MB)	Plots (All/Fail)	Links
✗	ACDDigi	Apr 10, 2006	0	NA	0 / 0	
✗	ACDTop	Apr 10, 2006	0	NA	0 / 0	
✓	AllGamma	Apr 11, 2006	8805	377	111 / 17	log meta-data
✓	BackGndAvg	Apr 10, 2006	10	1	0 / 0	log meta-data
✗	CALSingleCrystal	Apr 10, 2006	5	1	0 / 0	
✓	VerticalGamma100GeV	Apr 11, 2006	30596	278	110 / 18	log meta-data
✓	VerticalGamma100MeV	Apr 11, 2006	8138	416	110 / 23	log meta-data
✓	VerticalGamma10GeV	Apr 11, 2006	8926	318	110 / 14	log meta-data
✓	VerticalGamma1GeV	Apr 11, 2006	8657	391	110 / 25	log meta-data

DisplayTag

Root files

xrootd

freehep-rootio

User's request

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

IIS

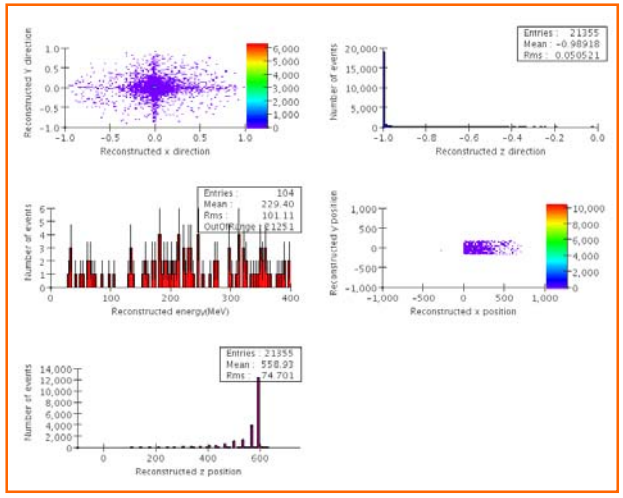
JSP in Tomcat Servers

jdbc

DB

AIDATLD

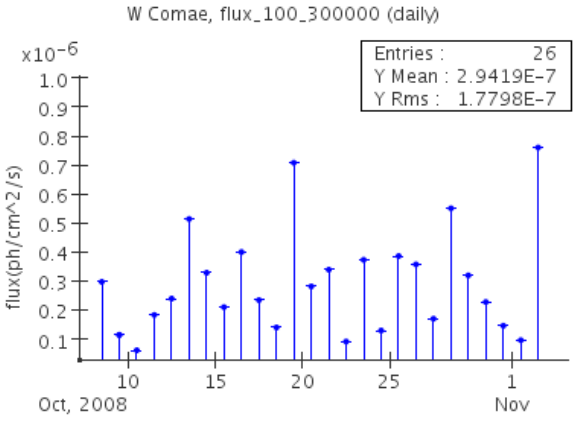
AIDA/JAIDA



Deliveries/Runs processing status

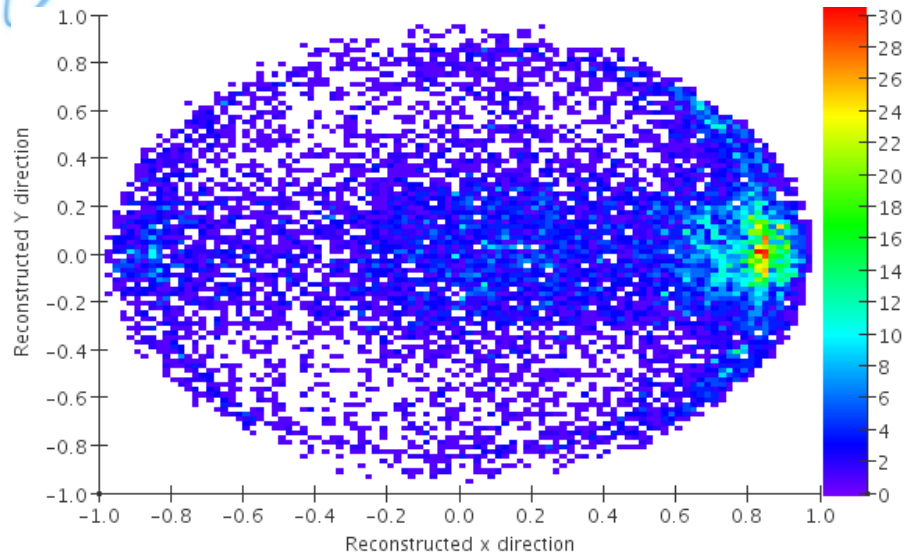
Delivery		FASTCopy		HalfPipe	Runs			L1Proc					GRB Search
Id	Time (UTC)	Proc	Logs	Proc	Id - Start MET	Status	Intent		Proc	Status	Logs	Data Mon	Proc
81104002	Nov/04/2008 01:45:27		15		247449893	InProgress	nomSciOps			Running	13 160	Di | Me | Cal	
					247443912 R	Complete	nomSciOps			Running	134	Di | Cal	
81104001	Nov/04/2008 01:24:25		23		247443912 R	Complete	nomSciOps			Running	3518 20 9 181	FM | Re | Me	
					247437920	Complete	nomSciOps			Running	173	Di | Me | Cal	
					247431894	Complete	nomSciOps			Running	173	Di | Me | Cal	
					247425813 R	Complete	nomSciOps			Running	200	FM | Di	
81103011	Nov/03/2008 23:47:06		15		247425813 R	Complete	nomSciOps			Running	1 3727	Re | Me | Cal	
					247419587 R	Complete	nomSciOps			Running	1 3 3724	FM | Di | Re | Me | Cal	
81103010	Nov/03/2008 20:32:38		17		247419587 R	Complete	nomSciOps			Running	1 3 17 3707		
					247415830 R	Complete	nomSciOps			Complete	1 1 3726	FM | Di | Re | Me | Cal	
					247410101 R	Complete	nomSciOps			Complete	1 6 3721	FM | Di | Re | Me | Cal	
81103009	Nov/03/2008 17:40:38		13		247410101 R	Complete	nomSciOps			Complete	2 3 3723		
					247404371 R	Complete	nomSciOps			Complete	1 2 3725	FM | Di | Re | Me | Cal	
81103008	Nov/03/2008 16:21:29		13		247404371 R	Complete	nomSciOps			Complete	1 3 3724		
					247398642 R	Complete	nomSciOps			Complete	1 2 3725	FM | Di | Re | Me | Cal	
81103007	Nov/03/2008 14:00:42		15		247398642 R	Complete	nomSciOps			Complete	1 2 3725		
					247392913 R	Complete	nomSciOps			Complete	1 2 3725	FM | Di | Re | Me | Cal	

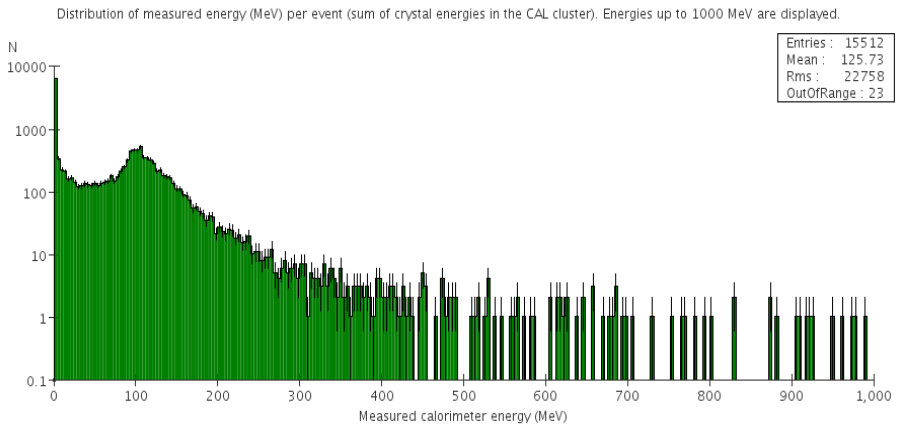
Data Representation/Export



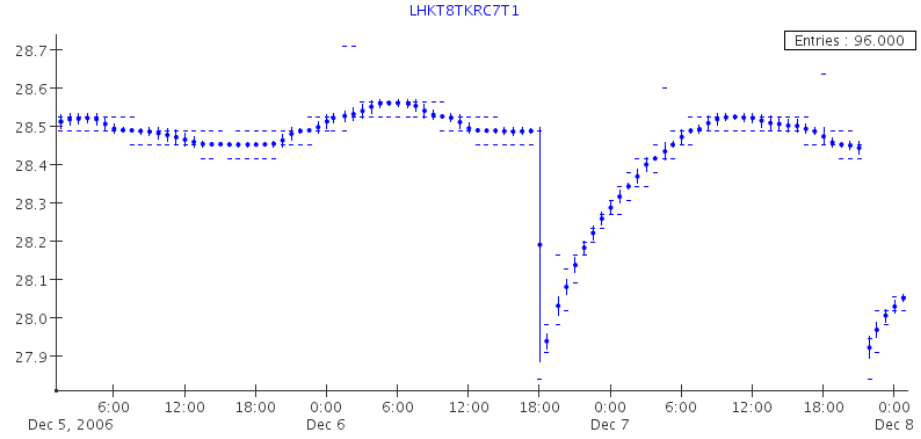
Date	MJD	MET	W Comae Mean	W Comae Rms
02-Nov-2008 12:00:00	54772	247320000	7.6E-7	2.16E-7
01-Nov-2008 12:00:00	54771	247233600	9.54E-8	2.41E-9
31-Oct-2008 12:00:00	54770	247147200	1.45E-7	7.34E-8
30-Oct-2008 12:00:00	54769	247060800	2.27E-7	7.44E-8
29-Oct-2008 12:00:00	54768	246974400	3.19E-7	1.09E-7
28-Oct-2008 12:00:00	54767	246888000	5.5E-7	1.61E-7
27-Oct-2008 12:00:00	54766	246801600	1.69E-7	6.94E-9
26-Oct-2008 12:00:00	54765	246715200	3.57E-7	1.18E-7
25-Oct-2008 12:00:00	54764	246628800	3.86E-7	1.26E-7
24-Oct-2008 12:00:00	54763	246542400	1.29E-7	4.83E-8
23-Oct-2008 12:00:00	54762	246456000	3.73E-7	9.69E-8

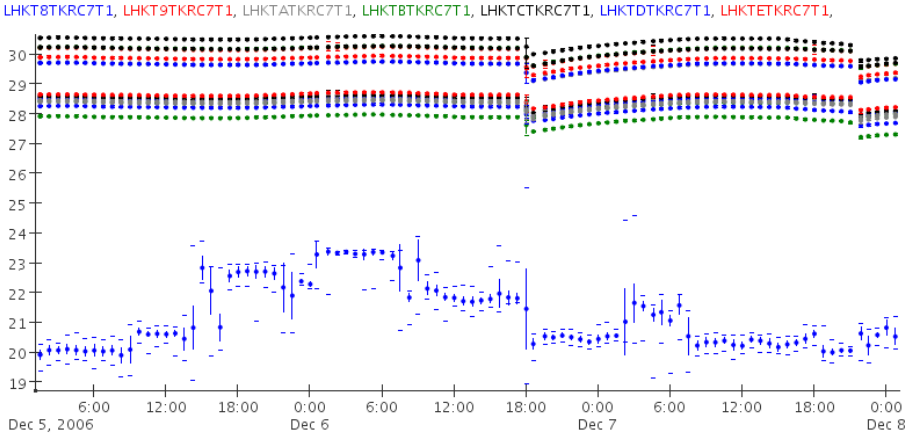
Data Quality Monitoring





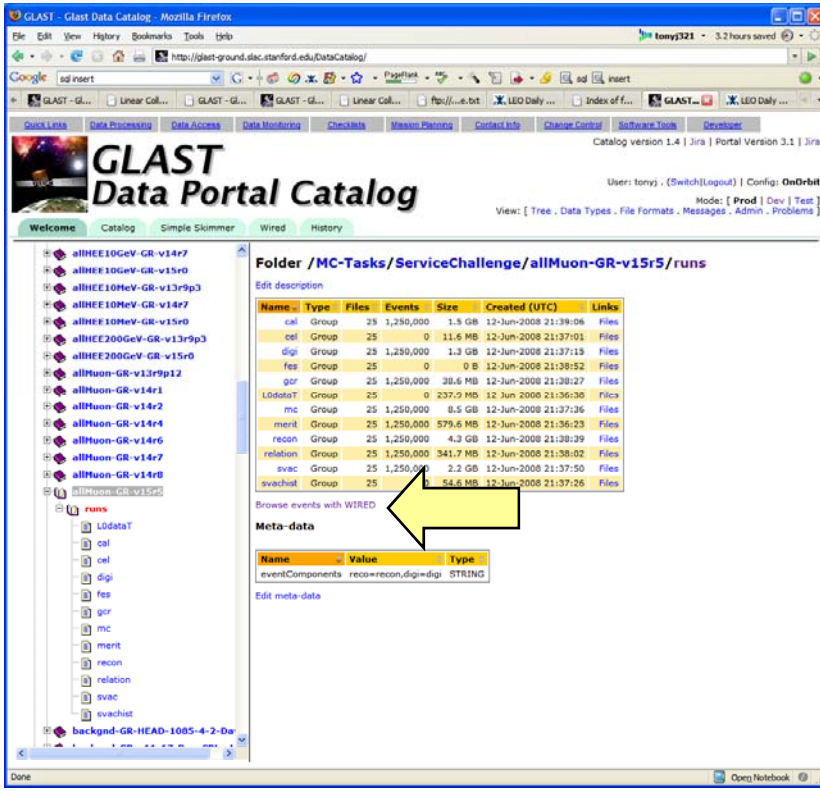
HouseKeeping Trending

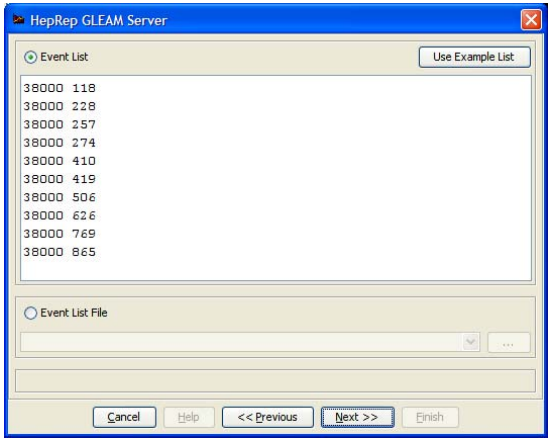




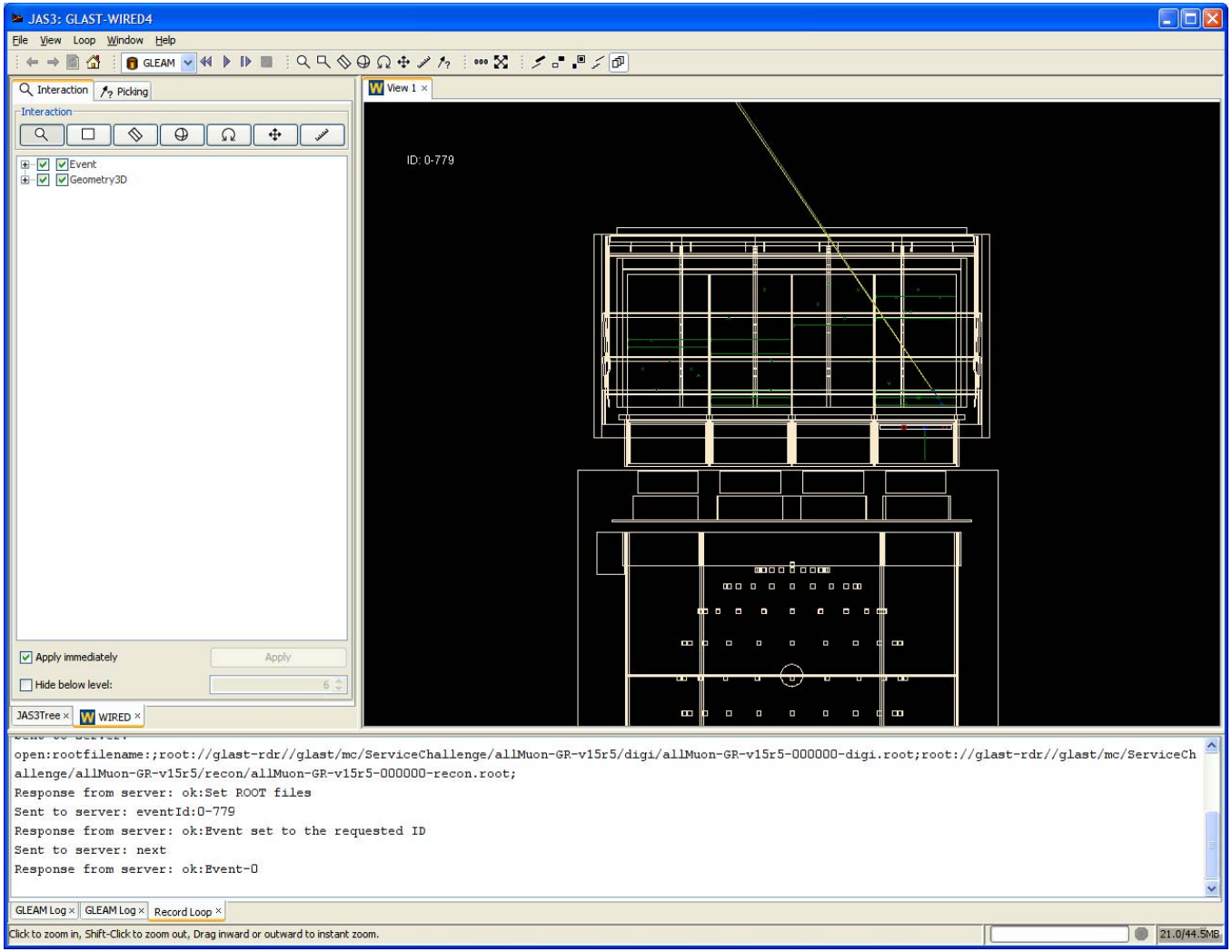
Wired4 Event Display for GLAST

- WIRED4
  - Java based experiment independent 3D event display
  - Many features including
    - Custom views, Dynamic Cuts
- For GLAST set up as web start application integrated into data catalog
  - Started by single-click on data catalog web interface. No prior installation required
    - Except Java itself – but always there thanks to EVO
  - Dialog allows user to select events of interest
  - Web service used to find location of files containing requested events from database
  - CORBA server used to fetch “HepRep” description of event
  - Event display appears…

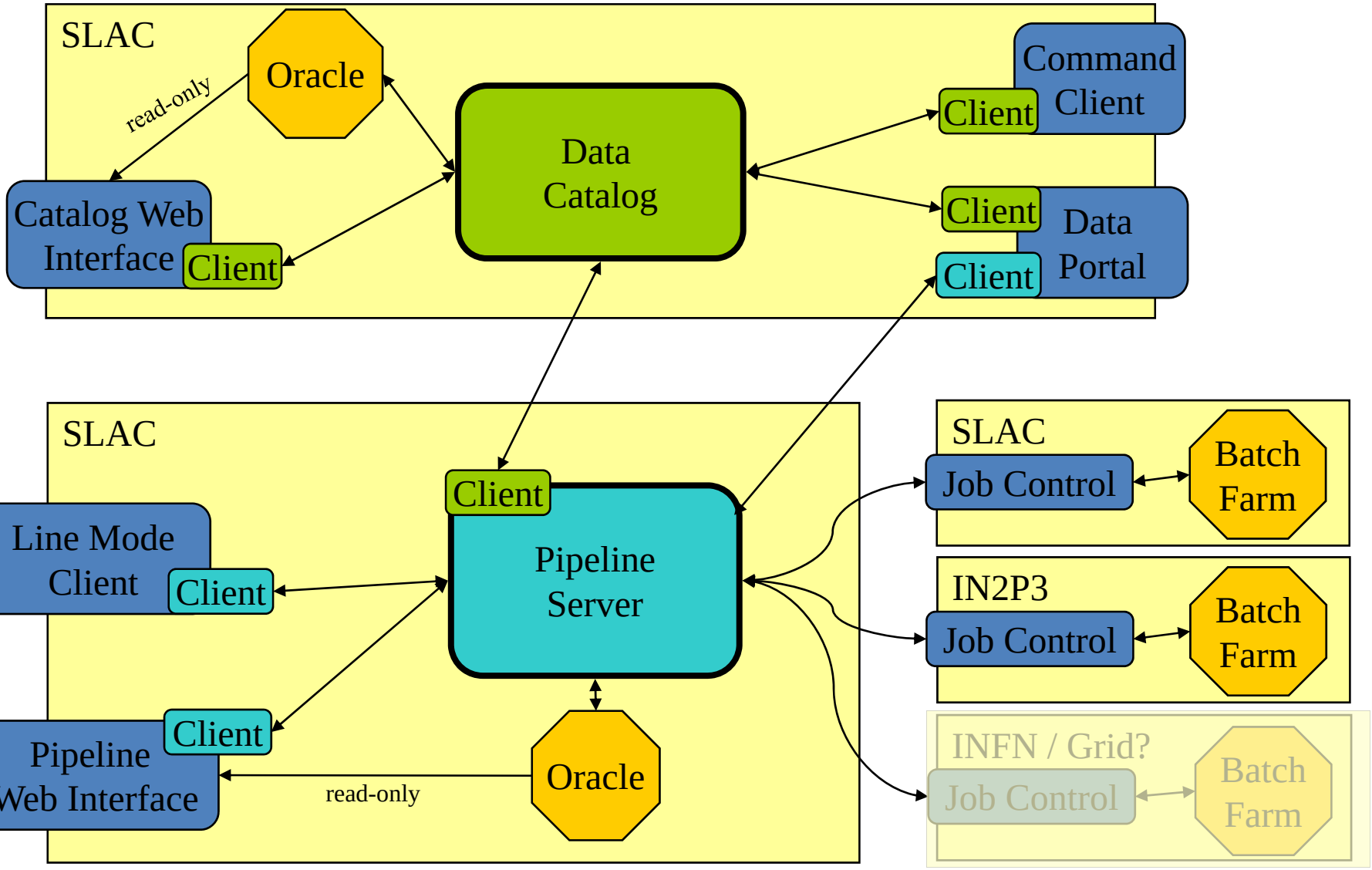




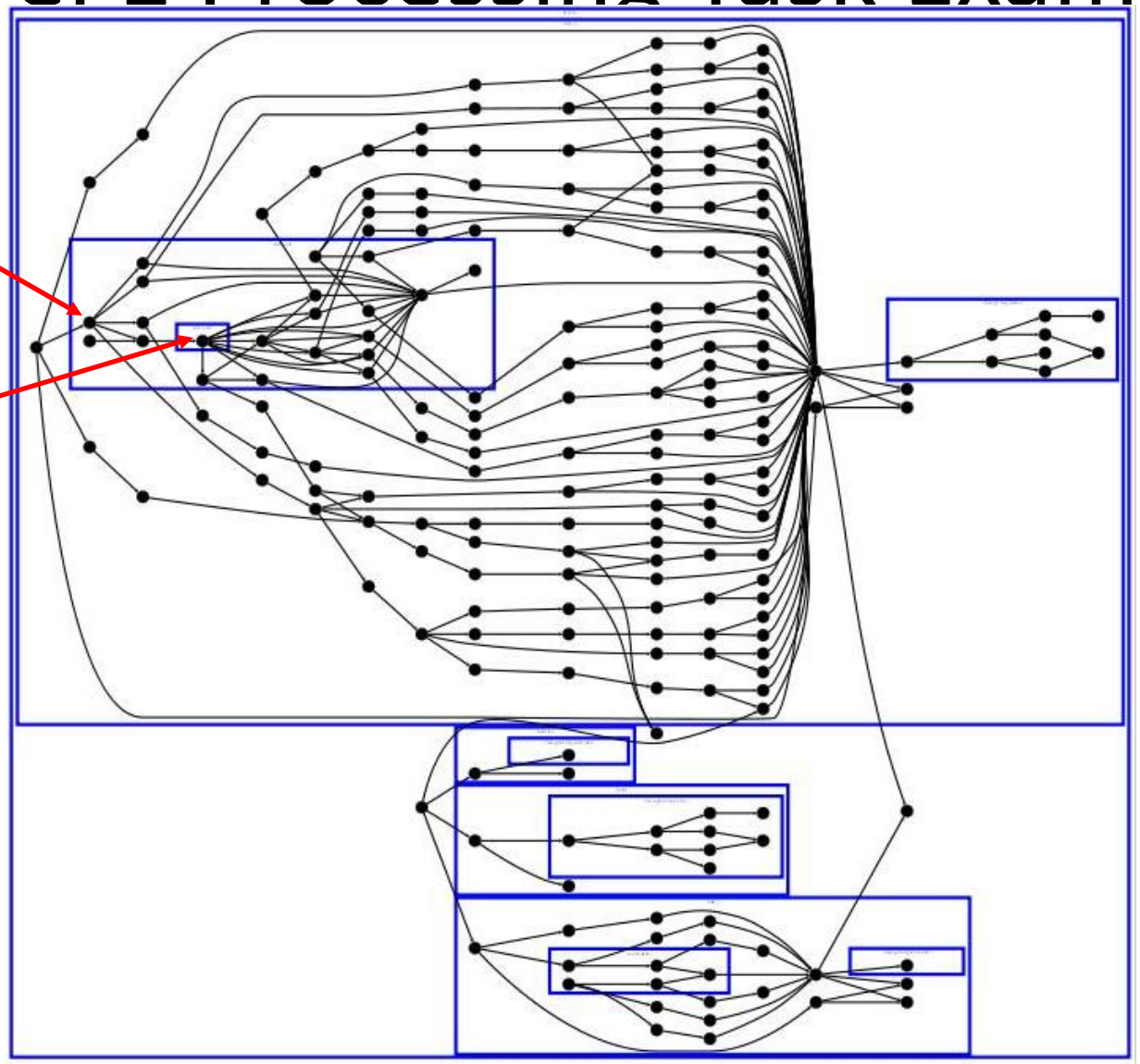
WIRED4 Event Display for GLAST



Fermi LAT Pipeline and Data Catalog Components



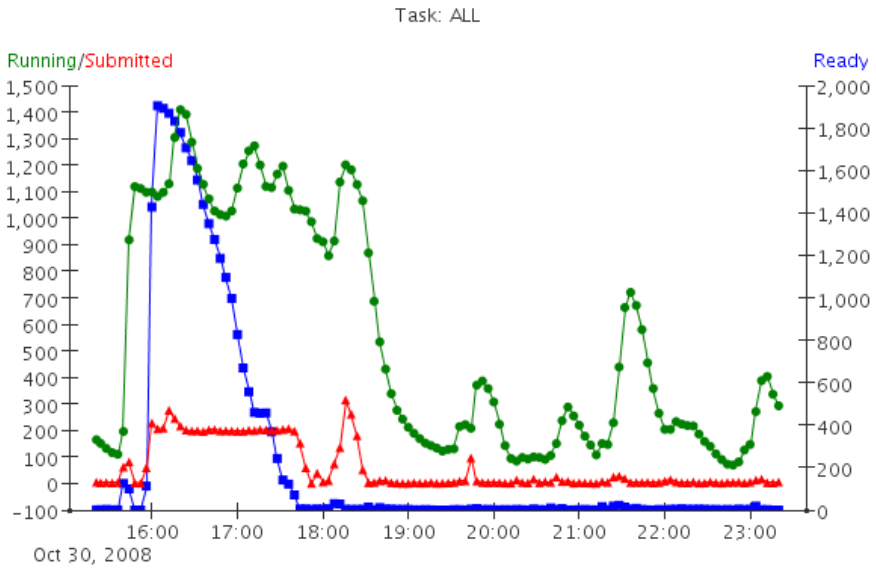
Level 1 Processing Task Example



Front End: Activity Plots

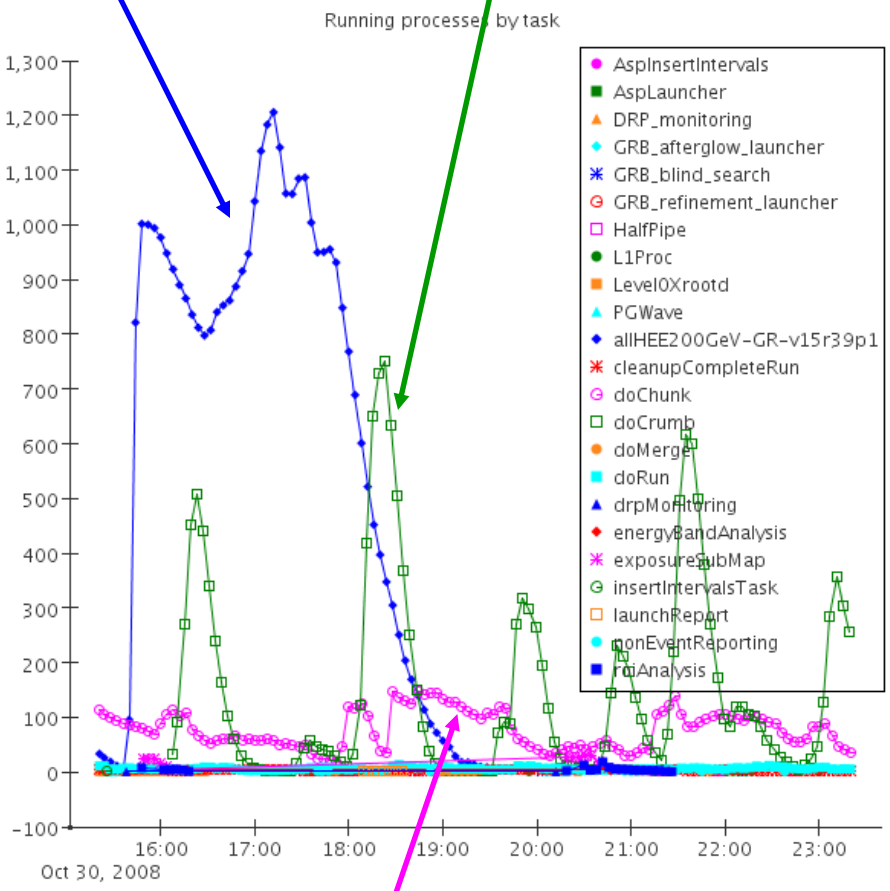
Select Task: All Tasks
Start: None End: None or last 8
Submit

Starting Date: Thu Oct 30 15:20:00 PDT 2008 - Ending Date: Thu Oct 30 23:20:00 PDT 2008
121 records found from table Minutes with group by 4



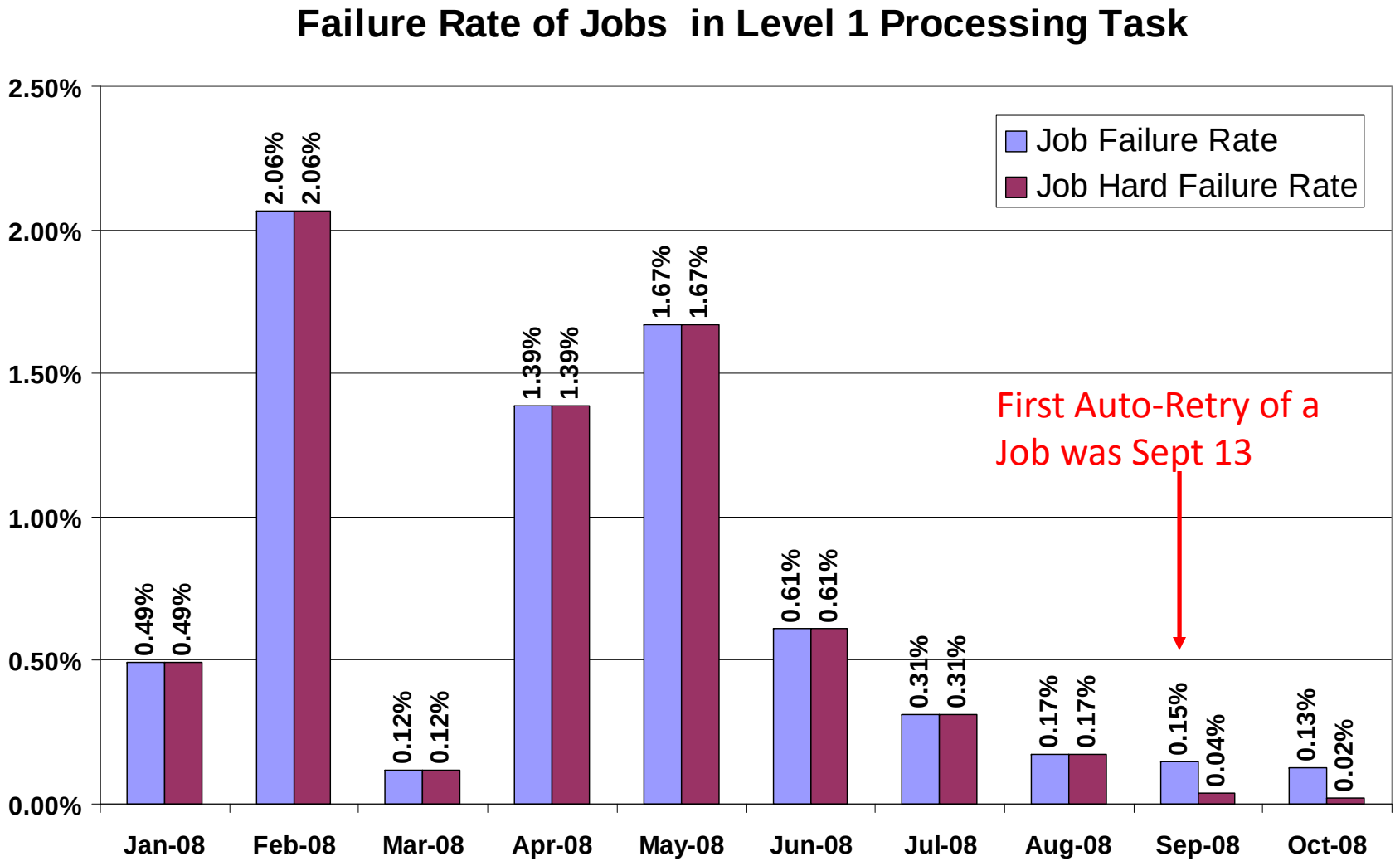
Simulation

L1 Reconstruction



L1 Digitization

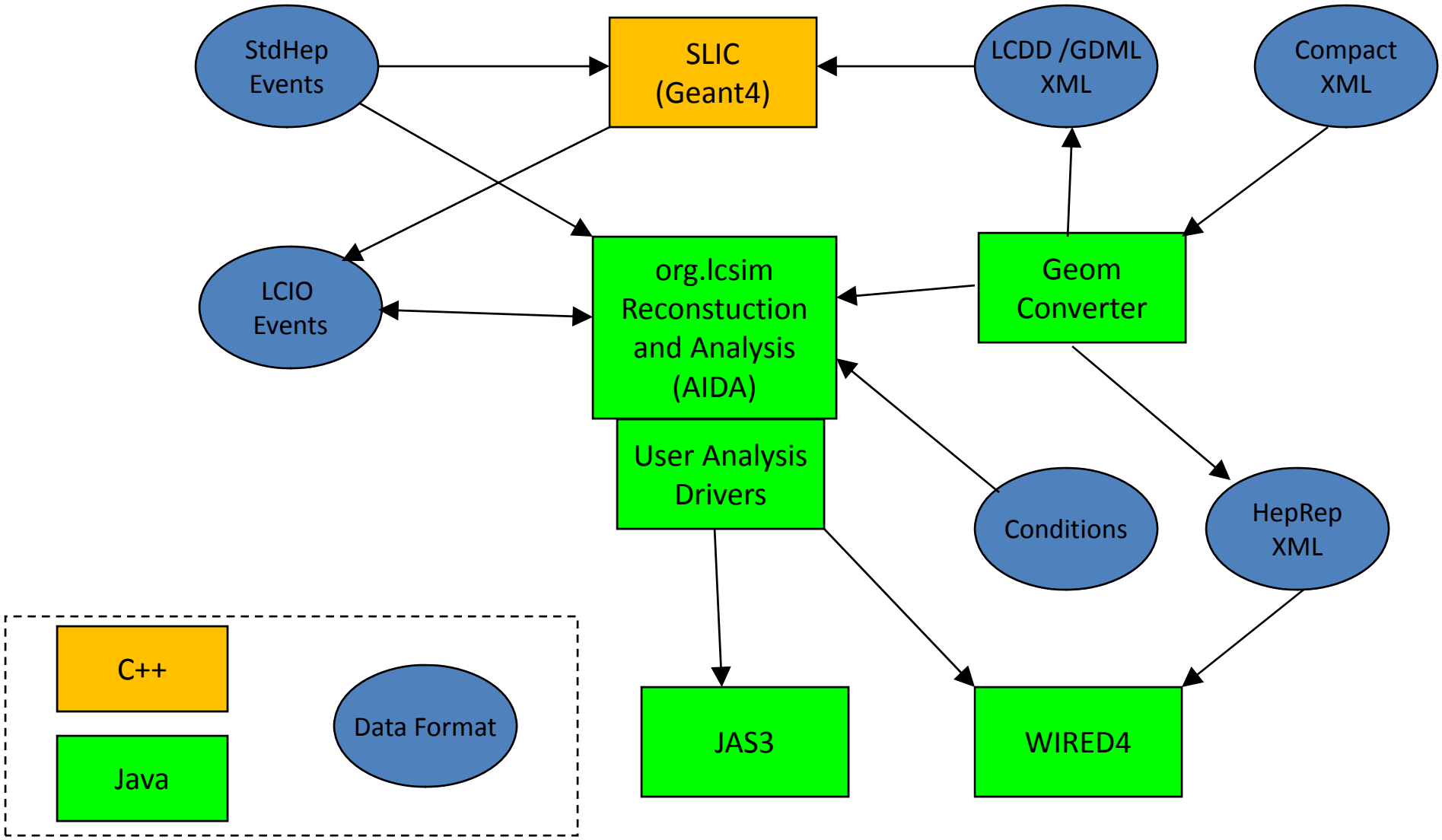
Auto-retry of Failed Jobs



ILC Detector Development

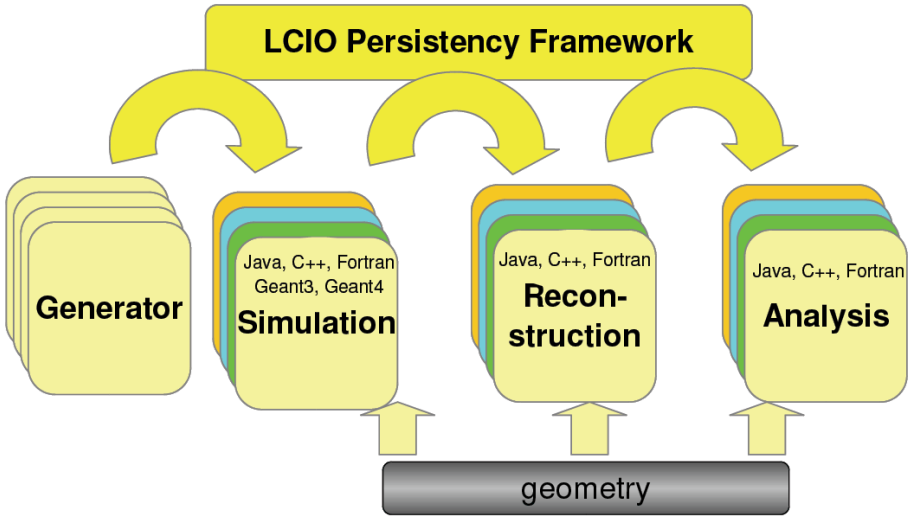
- Goal of ILC detector development is to study a variety of different possible detector configurations/technology for a future linear collider.
- Work has been ongoing for many years
  - Typically involves people who only work part time on this project
  - Students, post-docs typically work for 1 or 2 years then move on to something else
- Software needs to be flexible, very easy to learn and use
  - At past workshops we have distributed software suite on CD with goal “15 minutes from zero to physics”.
    - Windows, Mac, Linux

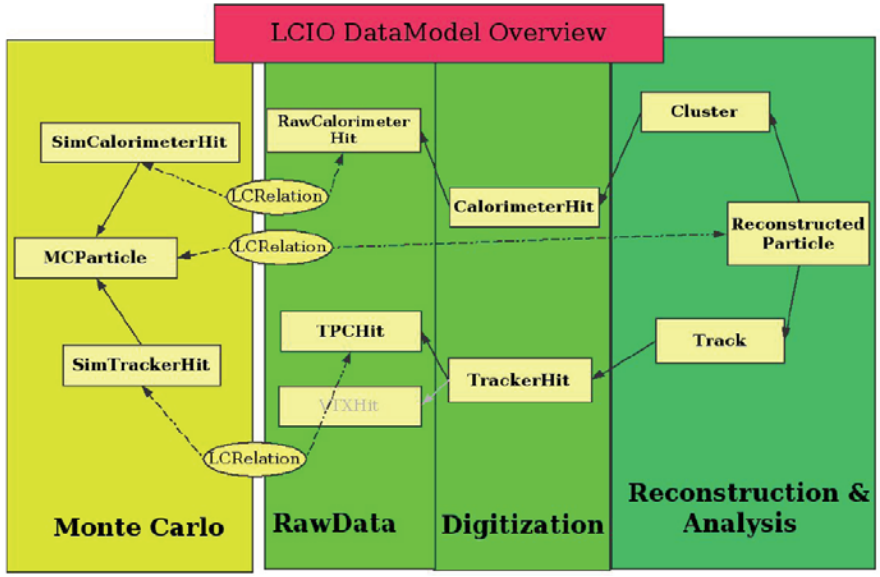
ILC Reconstruction/Simulation/Analysis Framework as used by SiD detector



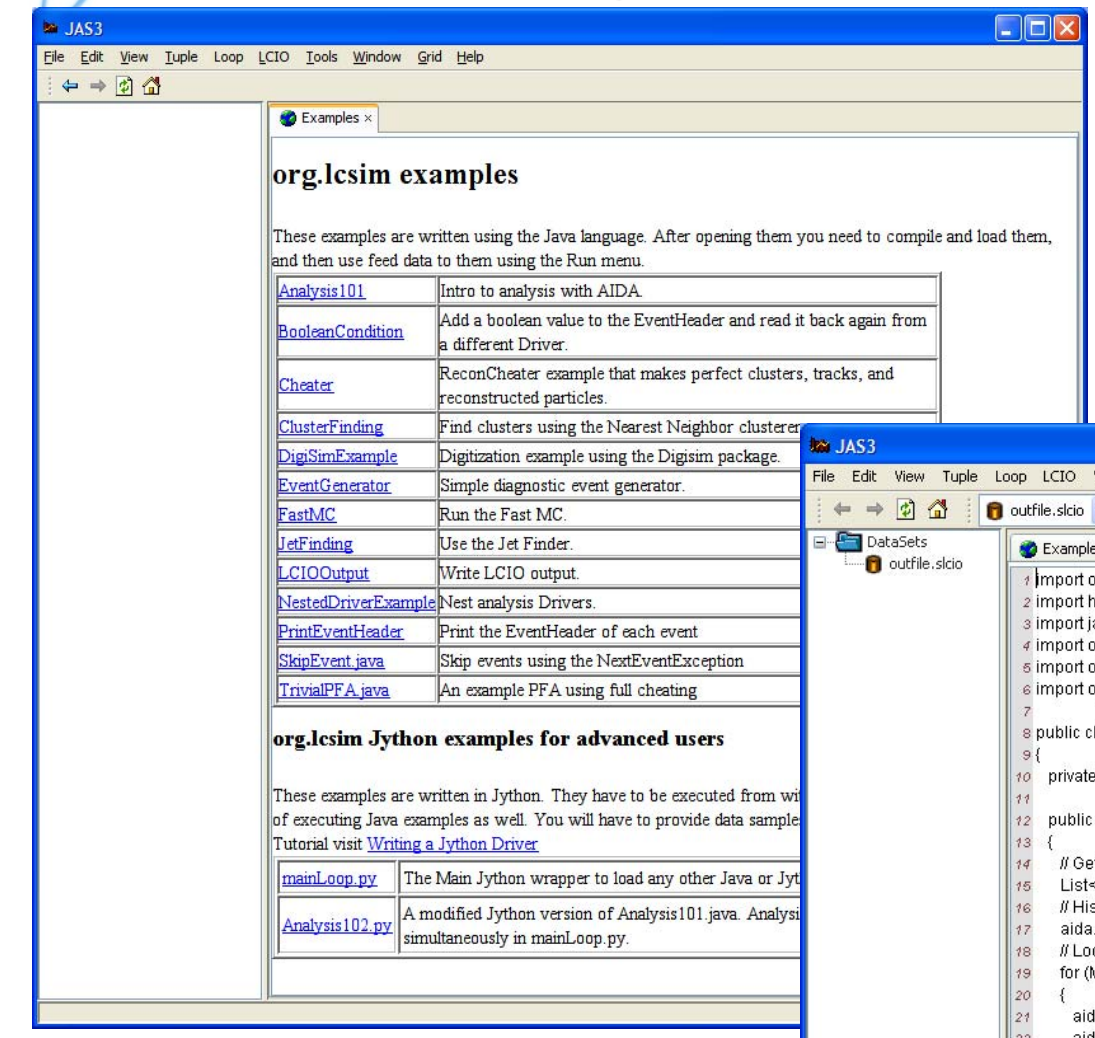
LCIO Persistency Framework

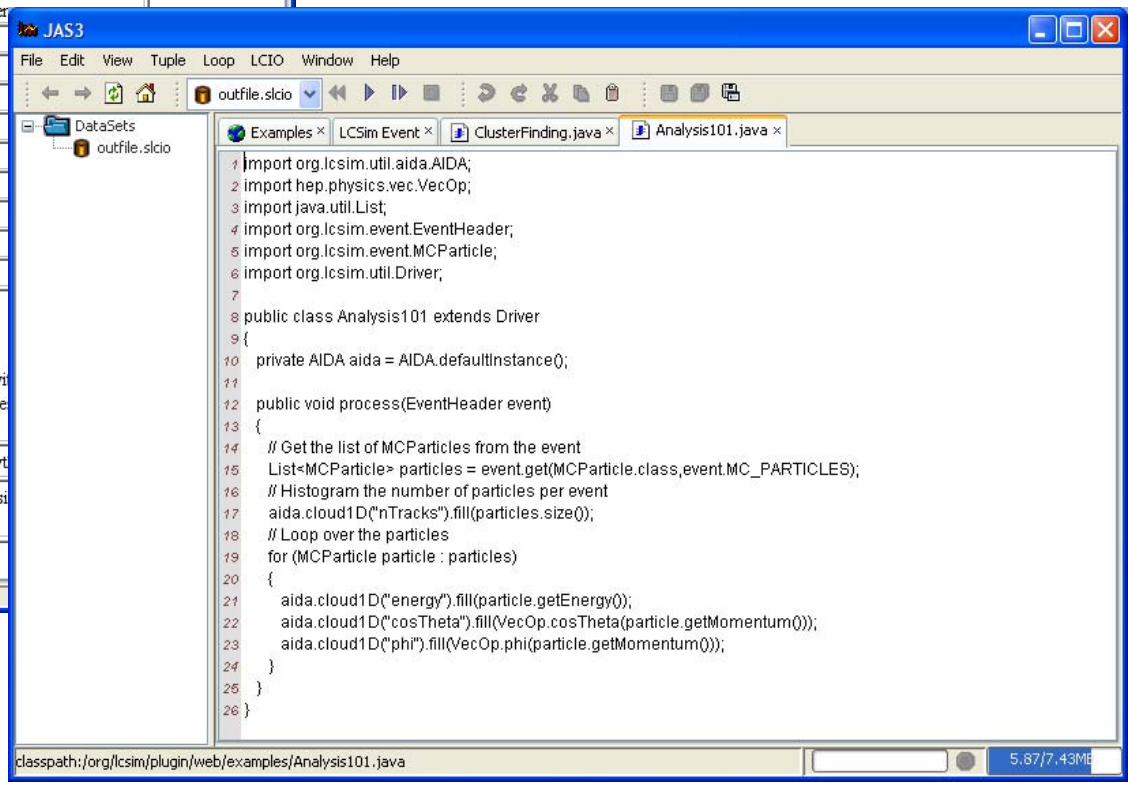
- Object model and persistency
  - Events
    - Monte Carlo
    - Raw
    - Event and run metadata
  - Reconstruction
  - Parameters, relations, attributes, arrays, generic objects, …
- All the ILC simulators write LCIO
  - Enables cross-checks between data from different simulators
  - Read/write LCIO from
    - Fast MC / Full Simulation
    - Different detectors
    - Different reconstruction tools



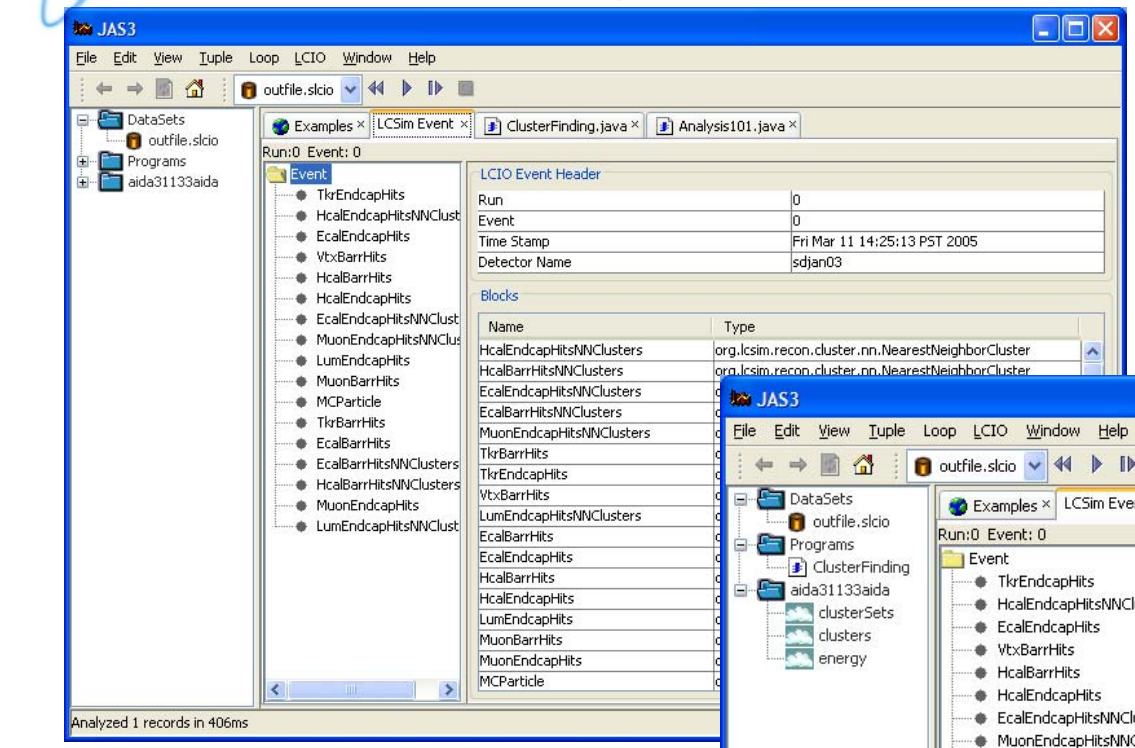


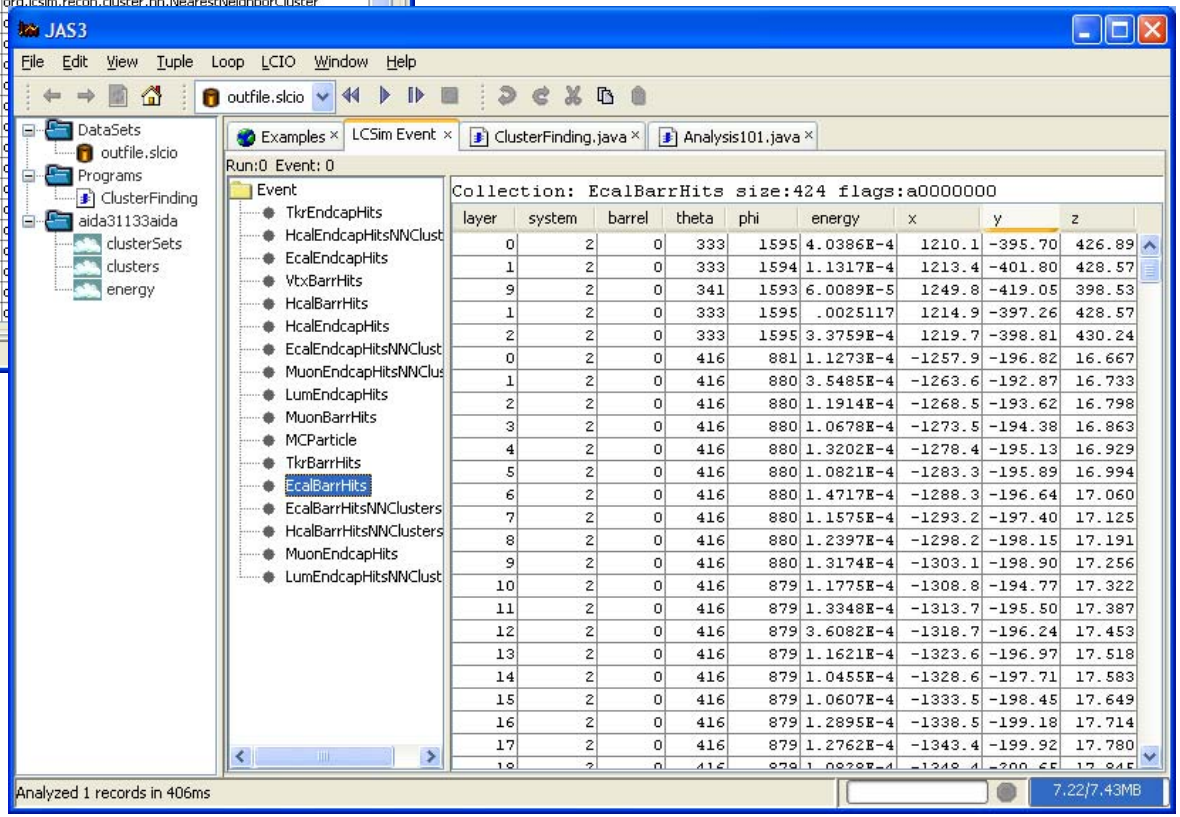
org.lcsim: Examples



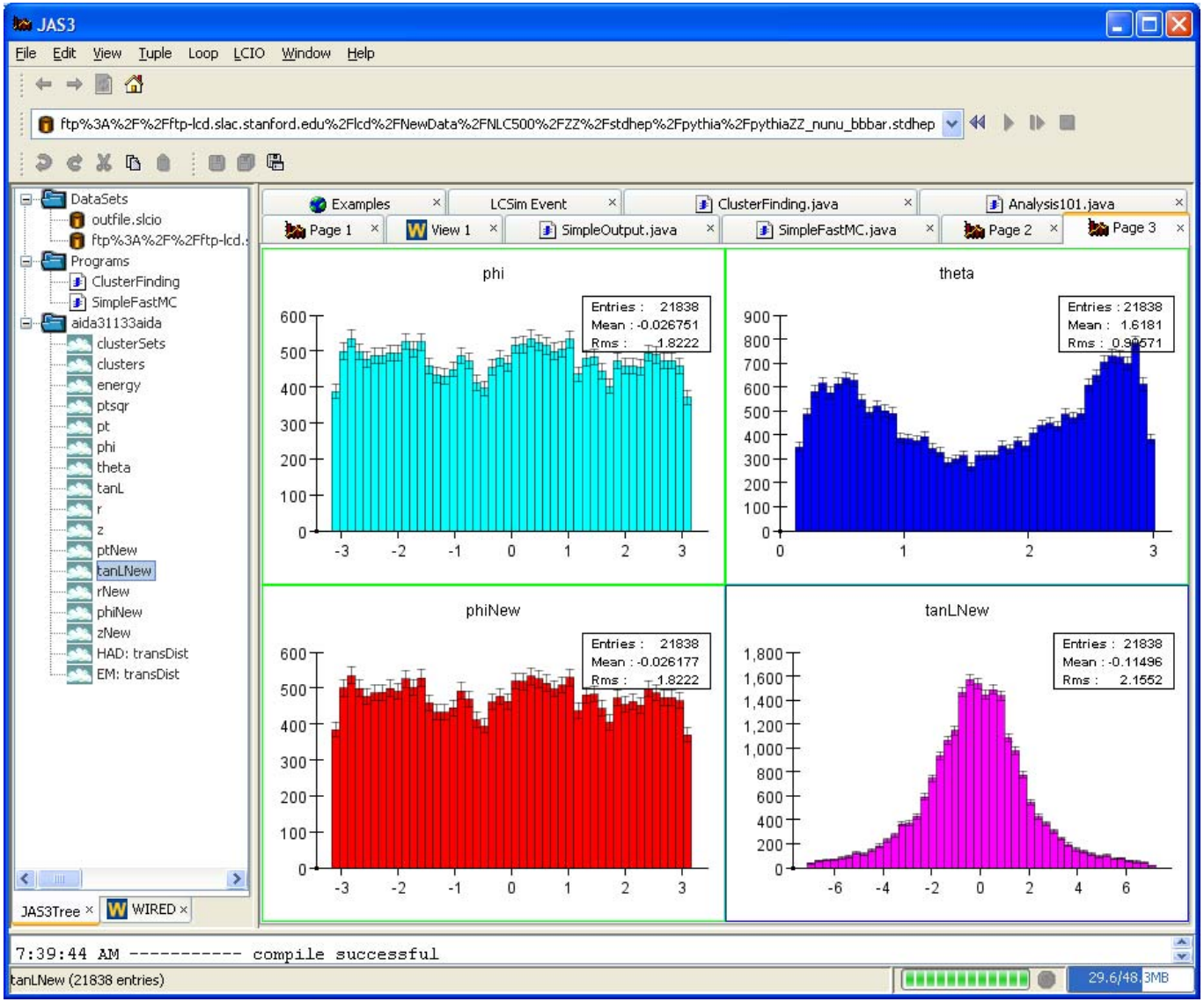


org.lcsim: Examples

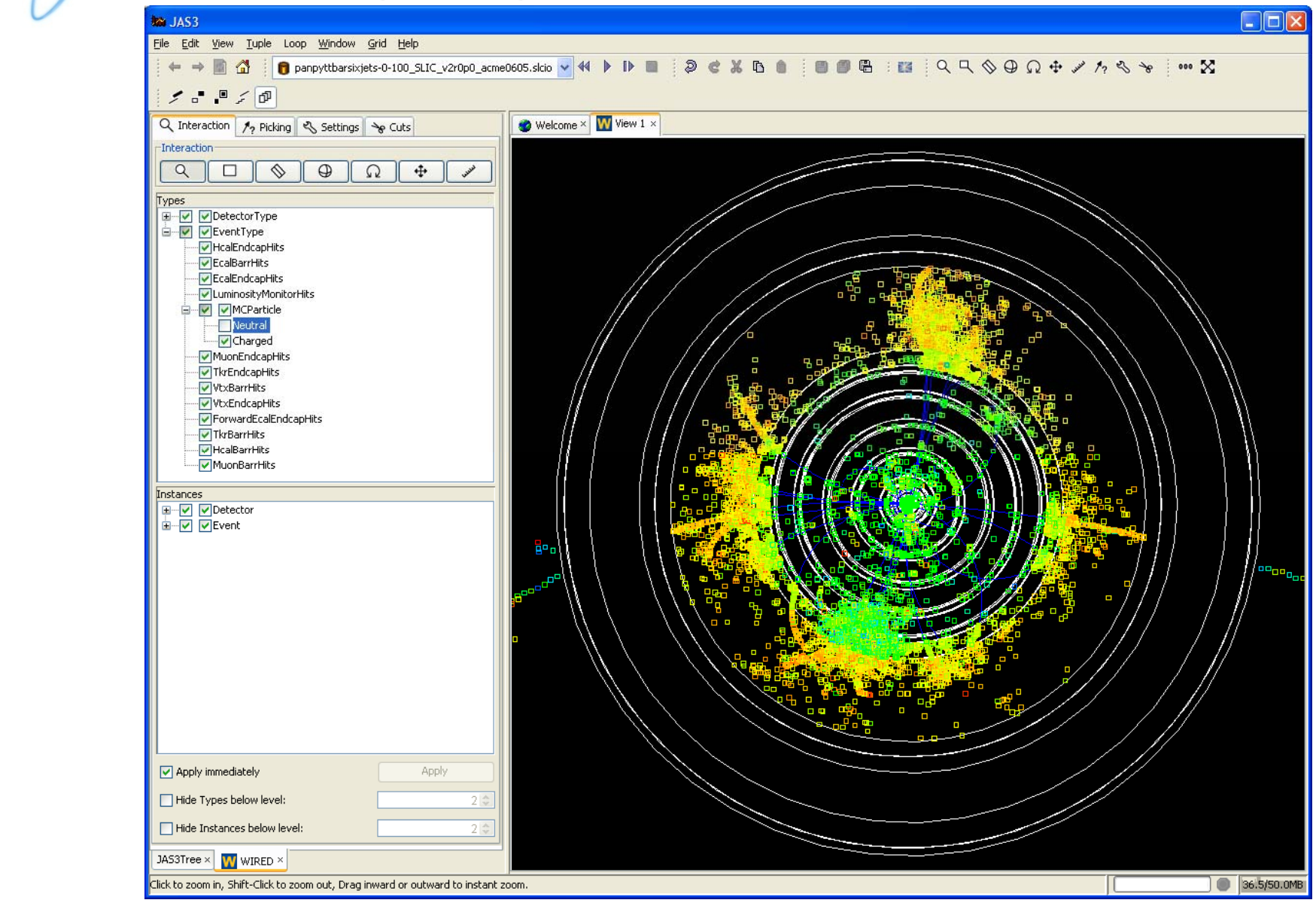




org.lcsim: Plot Viewing



Using org.lcsim with WIRED4



Org.lcsim Reconstruction

- Reconstruction package includes:
  - Physics utilities:
    - Jet finders, event shape routines
    - Diagnostic event generator, stdhep reader/translator
    - Histogramming/Fitting/Plotting (AIDA based)
    - Event Display
    - Processor/Driver infrastructure
  - Fast MC
    - Directly reads stdhep events (or LCIO events)
    - Track/Cluster smearing
    - Produces ReconstructedParticles
  - Reconstruction
    - Cheaters (perfect reconstruction)
    - Detector Response
      - CCDSim, Digisim
    - Clustering Algorithms
      - Cheater, DirectedTree, NearestNeighbour, Cone
    - Tracking Finding/Fitting Algorithms
      - TRF, SLD Weight Matrix, Kalman filter
    - Muon Finding, Stepper
    - Vertex Finding (ZvTop)
    - Track-Cluster association and Particle Flow analysis

Performance?

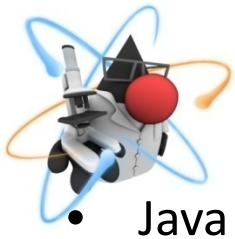
- We are in the process of simulating one years worth of ILC data (approximately 50 million events) at a variety of CM energies.
- For a standard model mix of events performance (preliminary) results look encouraging:
  - Geant4 (simulation, C++)
    - ~48 seconds/event
    - Peak memory ~480MB
    - Event Size: 165kB/event
  - org.lcsim (reconstruction, Java)
    - ~8 seconds/event
    - Peak memory ~450MB
    - Event Size: 220kB/event
- Caveat, both simulation and reconstruction use somewhat “idealized” geometries, no (mis)alignment of detector elements is currently implemented.
- Other ILC detector proponents have C++ based reconstruction programs (but somewhat different detectors and algorithms). It would in principle be possible to do a direct performance comparison.

Obstacles to Java Adoption in HEAP

- Technical issues
  - Large existing body of C++ code
    - Geant4, Root
    - LHC experiments
      - Currently dominate HEP software but made language choices when C++ was only viable alternative
      - Complexity of detectors and events at LHC push limits of simulation/reconstruction in ways which simpler events and/or detectors may not (e.g. Glast, ILC).
  - Difficulty of mixing Java and C++ in same program
    - It is possible using tools such as SWIG
      - We have experimented with converting Geant4 examples to Java
    - But takes significant manpower investment to create and maintain binding
  - Lack of IO standards in HEP
    - We have created Java libraries which can read (and write) Root files
      - Lack of formal documentation on IO model make this difficult to maintain
        - » Requires reverse engineering root C++ libraries, frequent updates to support new features
    - Interesting comparison with astronomy community where IO formats are standardized
      - FITS, HDF5
      - Language binding to almost all known languages (Java, .net, python, C, C++, …)
  - High-performance general purpose IO
    - There are many persistency toolkits available for Java but none which match the performance and flexibility of Root IO for Hep data
      - Tools exist within the language for scalable high-throughput IO which in principle make this possible.

Obstacles to Java Adoption in HEAP

- Non-Technical Issues
  - SUN
    - Pros
      - Generally viewed as doing excellent job of stewarding Java development
      - Has made huge manpower investment, and has open “Java Community Process” for developing language/library changes.
      - Release of Java as open-source software under GPL has made it possible for it to be included with many Linux distros and makes easier for others to contribute to development
    - Cons
      - Java’s strong identification with a single company makes people nervous
        - » Especially when that company’s future business model is cloudy
      - Sun makes money from licensing Java, but not by using Java in its own products
        - » Infuriatingly often leaves small but important issues unaddressed for many years
  - HEP Software model
    - Developing projects such as JAS, Wired, AIDA, FreeHEP take large investments of manpower
    - HEP activities at SLAC where much of this work has been done are shrinking
      - Big push to account all costs to specific experiments
        - » Makes developing software which spans multiple SLAC experiments surprisingly difficult
        - » Makes supporting users from experiments without direct SLAC involvement very hard



Outlook

- Java is currently used in a variety of ways in HEAP
  - Mostly outside of direct data simulation/reconstruction chain
  - I do not expect a mass exodus of LHC developers from C++ to Java
    - Too much investment in existing tools, which do an adequate job to keep users happy
    - Do expect to see increased use of Java in periphery of experiments, such as web applications where the is no compelling C++ alternative
- Experience from ILC detector work shows that Java can be used for full reconstruction/analysis chain
  - Easy for developers to work with
  - Easy for analysis users to learn and use
  - If we were starting real ILC detector development now (unfortunately we are not) we would almost certainly use Java.
  - Java remains a viable alternative for other post LHC experiments
    - Especially those with less manpower and less onerous computing requirements than LHC
- Relevance to LHC?
  - Will the issues that make it difficult for Java to interoperate with LHC era HEP tools make it difficult for other new technologies to be integrated as they become available?
  - Will the choice of tightly coupling simulation/reconstruction software and data storage to C++ look like a good choice in 10 years time?
  - Is C++ really the best language for data analysis?
    - Especially as new students and post-docs arrive

Links

- Much of the HEP software I have described is available as open-source projects. New users (and especially new developers) are welcome:
- FreeHEP – http://java.freehep.org/
- org.lcsim - http://www.lcsim.org/software/lcsim
- LCIO - http://lcio.desy.de
- SLIC - http://www.lcsim.org/software/slic
- LCDD - http://www.lcsim.org/software/lcdd
- JAS3 - http://jas.freehep.org/jas3
- AIDA - http://aida.freehep.org
- AIDATLD – http://aidatld.freehep.org
- WIRED - http://wired.freehep.org