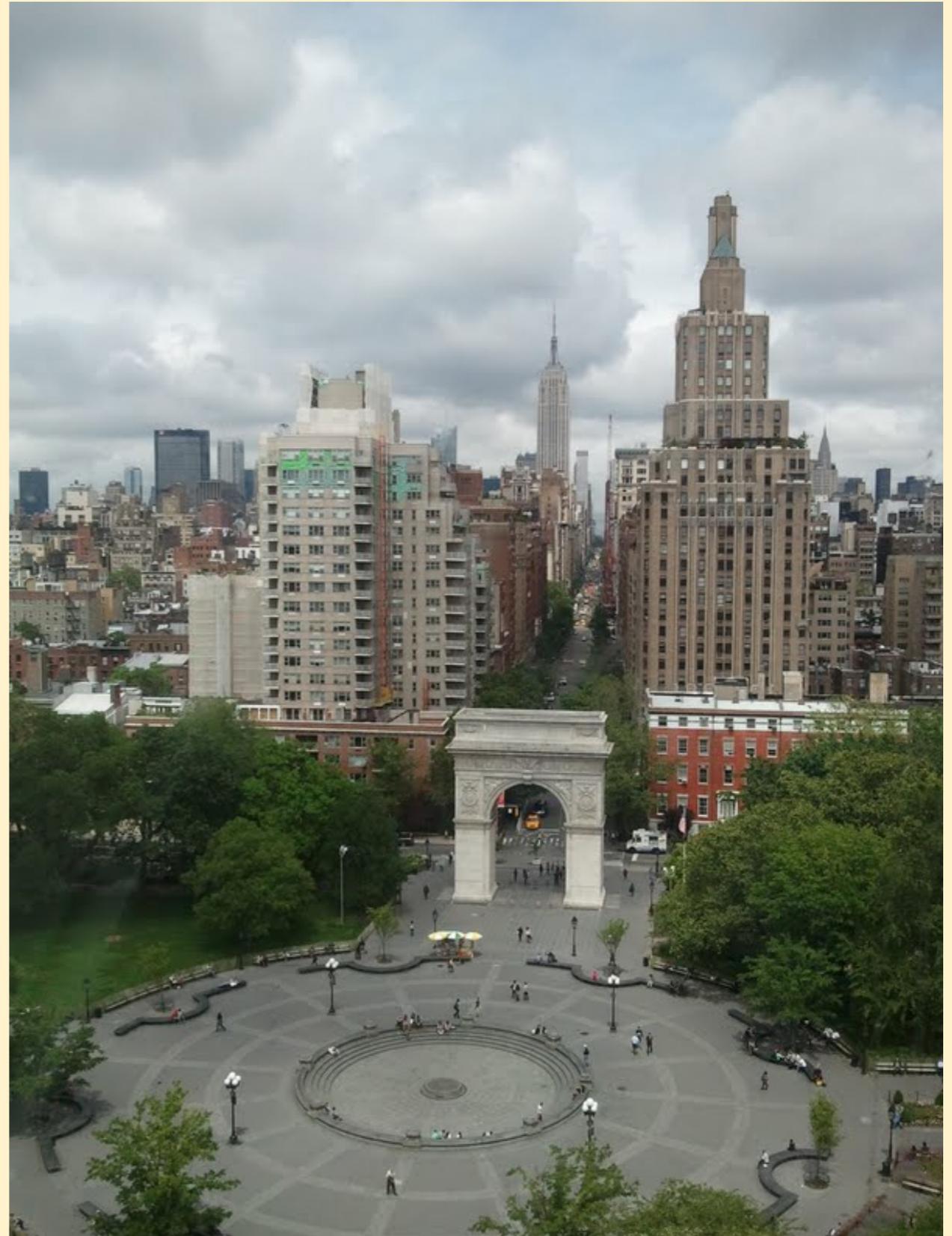


# Event Processing Track Summary

**Adam Lyon (Fermilab)**  
**Axel Naumann (CERN)**  
**Rolf Seuster (CERN)**

***CHEP 2012 @ NYC***



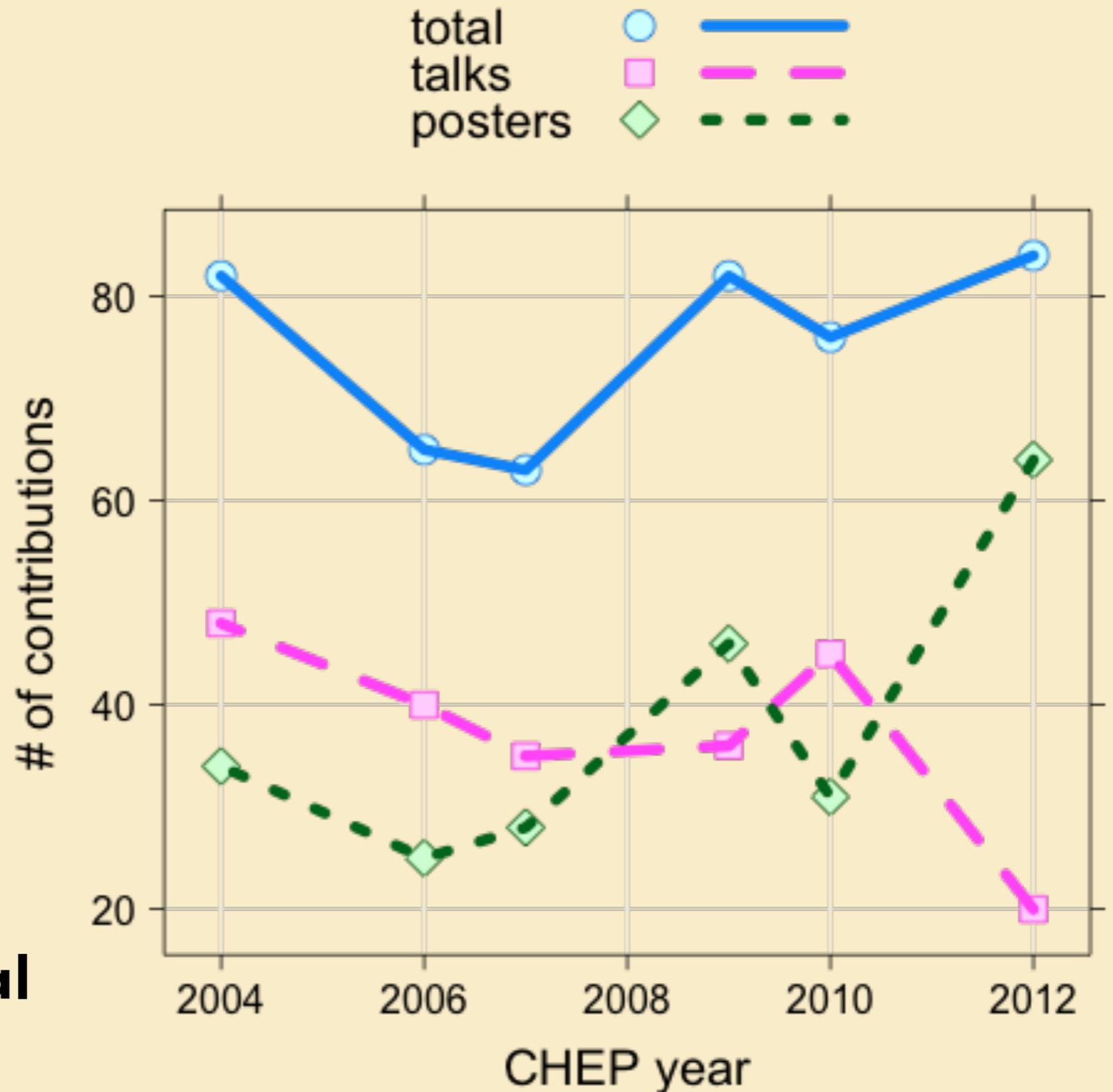
# How many contributions in event processing track?

## CHEP 2012

**84 contributions**

**20 talks, 64 posters  
(6 talks merged)**

**I'll concentrate on  
the talks and give  
perspective on the  
contributions in general**

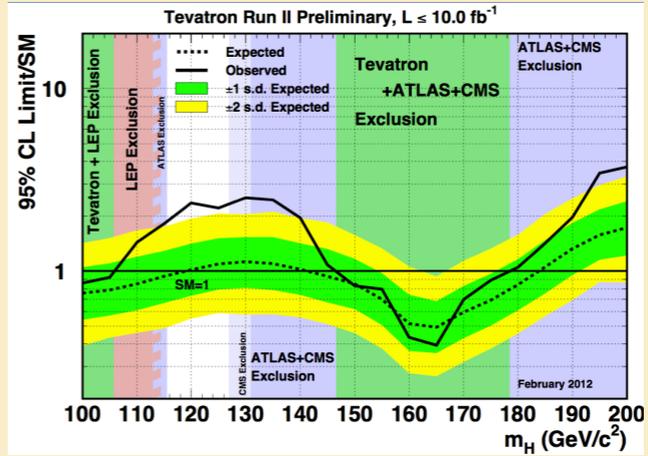
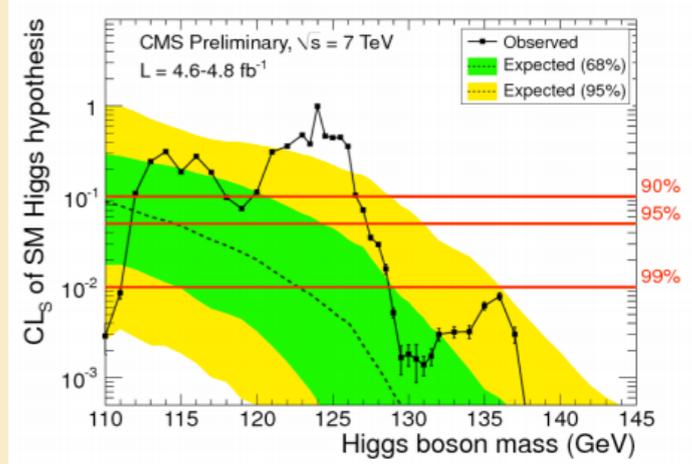
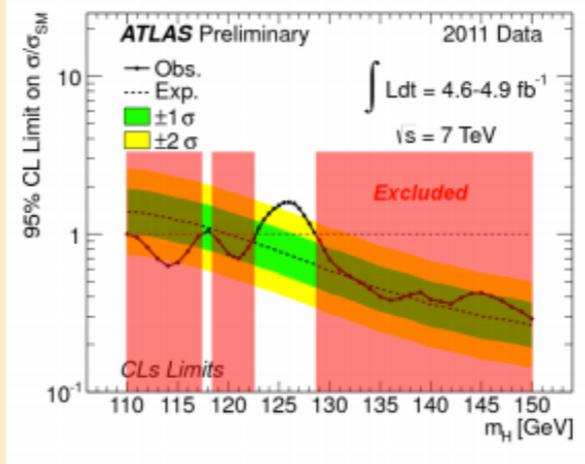
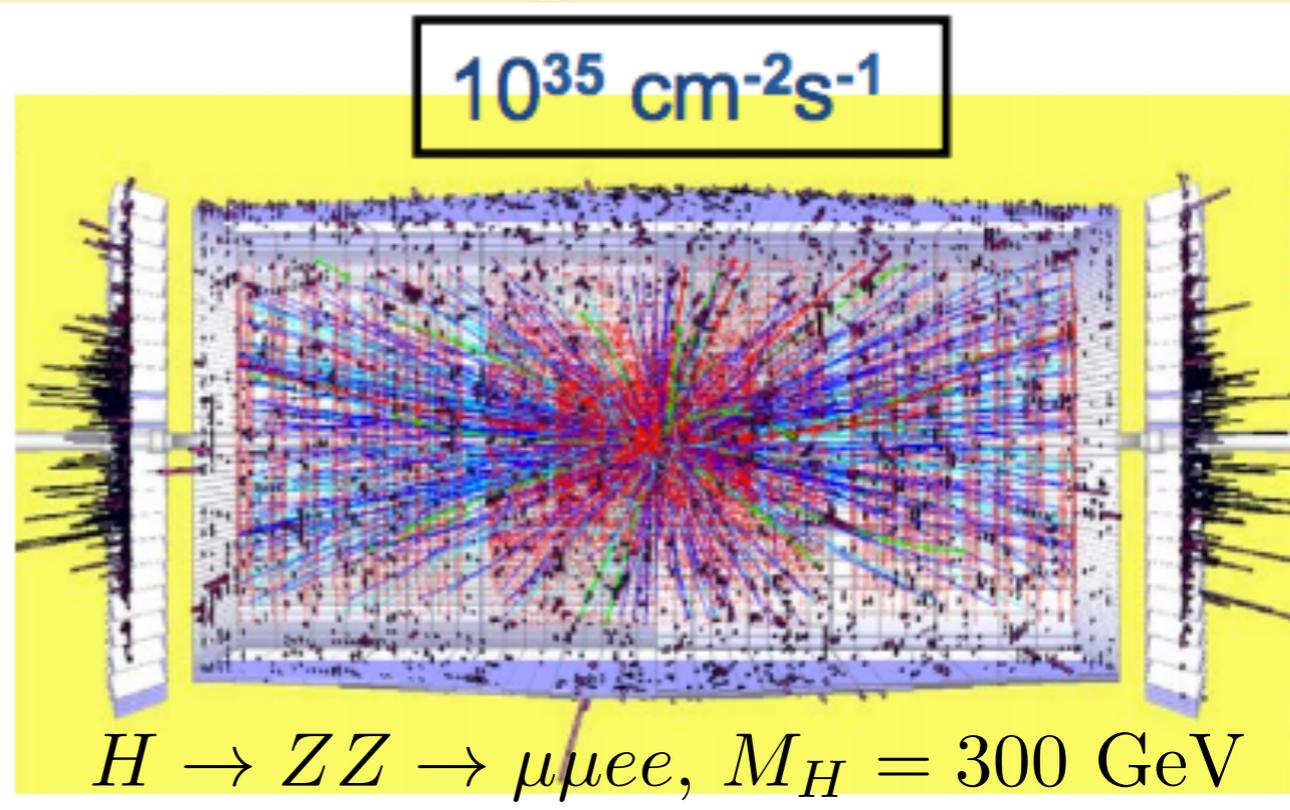


# What does event processing mean?

Go from events...  
 algorithms (e.g. tracking)  
 calibration/alignment  
 simulation  
 event display

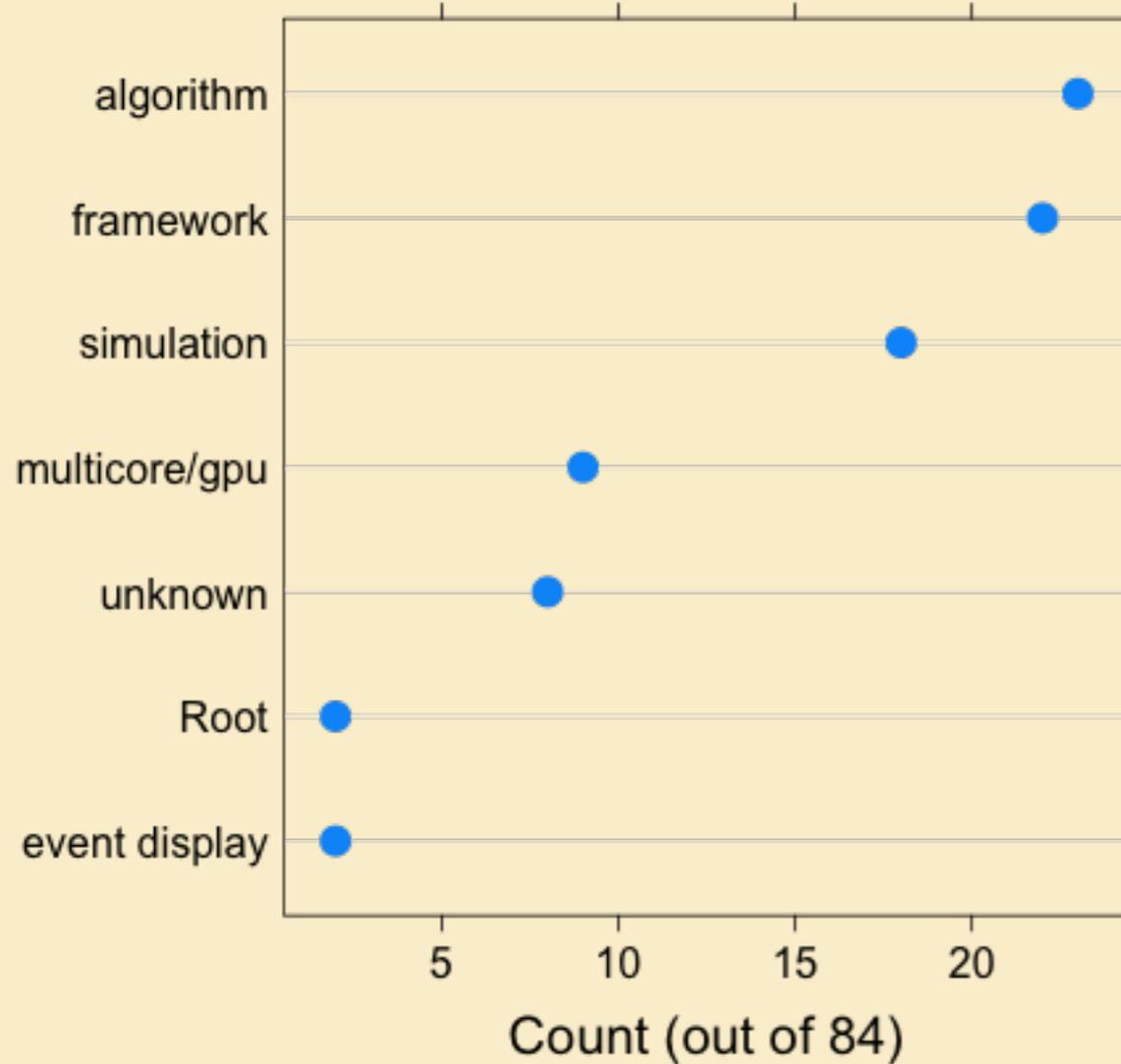


... to physics results  
 frameworks  
 multicore/gpus  
 Root

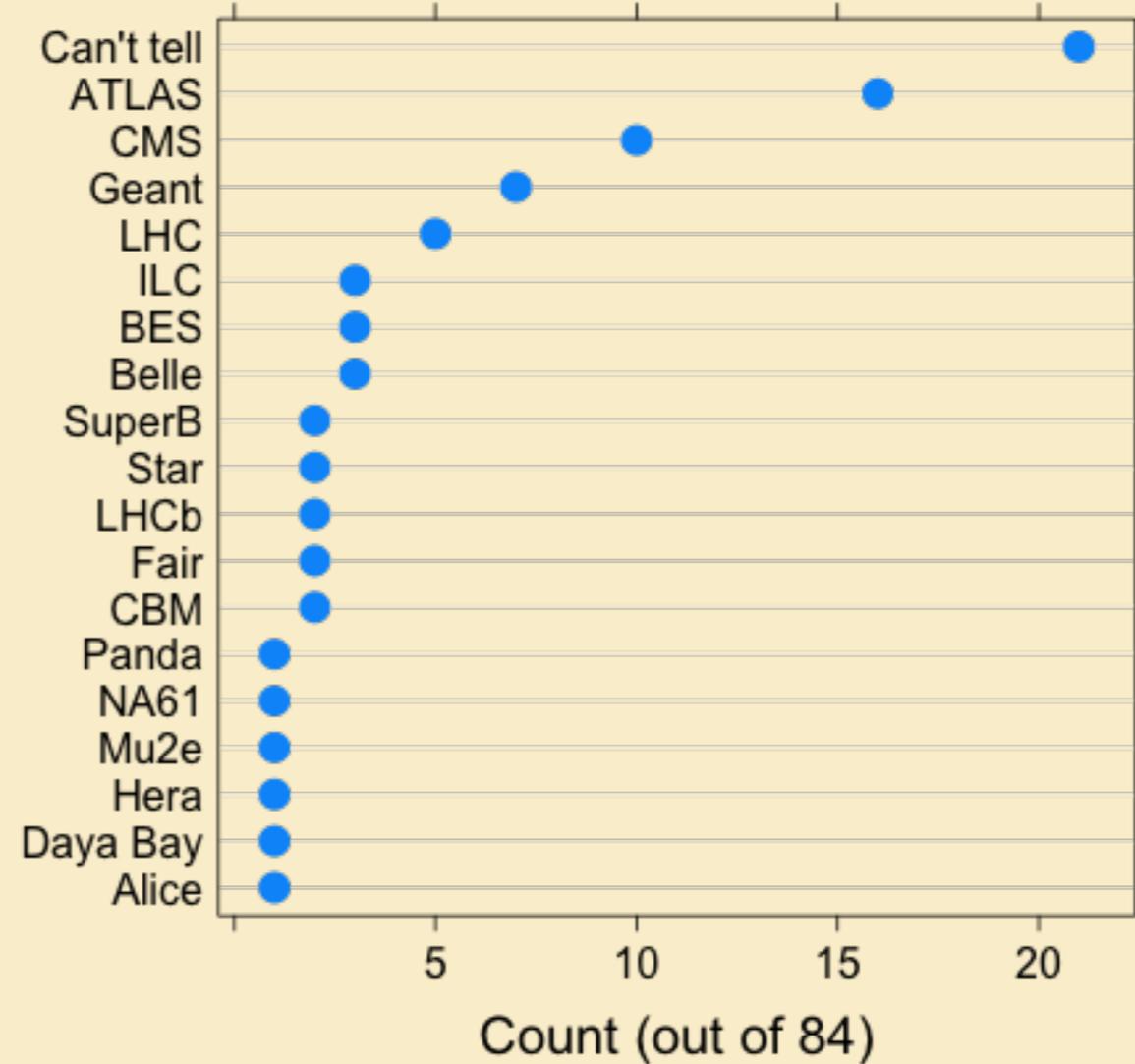


# How are the categories represented?

## Contributions by category

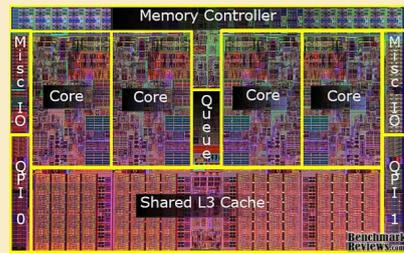


## Contributions by collaboration

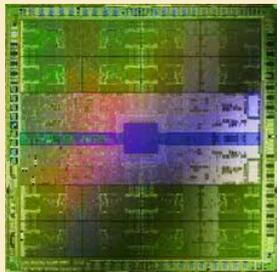


**Some of the categories overlap (e.g. multicore algorithms)**

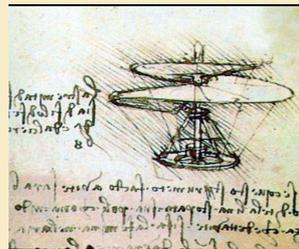
# Outline



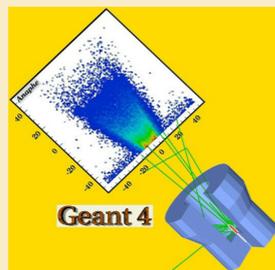
## Processing on multicores



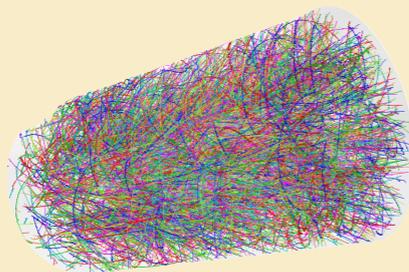
## Processing on GPUs



## Common frameworks



## Simulations



## Reconstruction algorithms

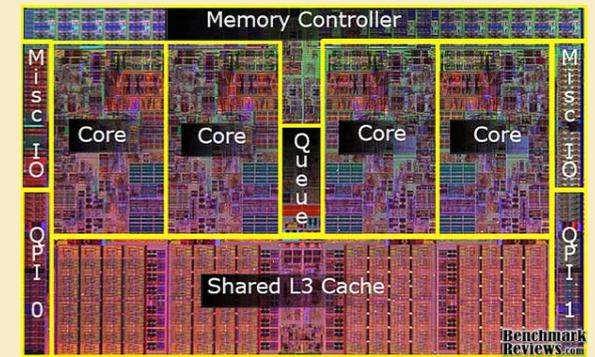
```
var dsname = "JetStream";
var files = DataSetFinder.FindROOTFilesForDS(dsname);
var data = ROOTLING.QueryableCollectionTree.Create(files);

var p = from evt in data
        from j in evt.Jets
        where Math.Abs(j.Eta) < 2.0
        select j;

p.Plot("jetpt", "jet pt", 100, 0.0, 100.0, j => j.PT / 1000.0);
```

## Everything else

# Processing on multicores is becoming a necessity



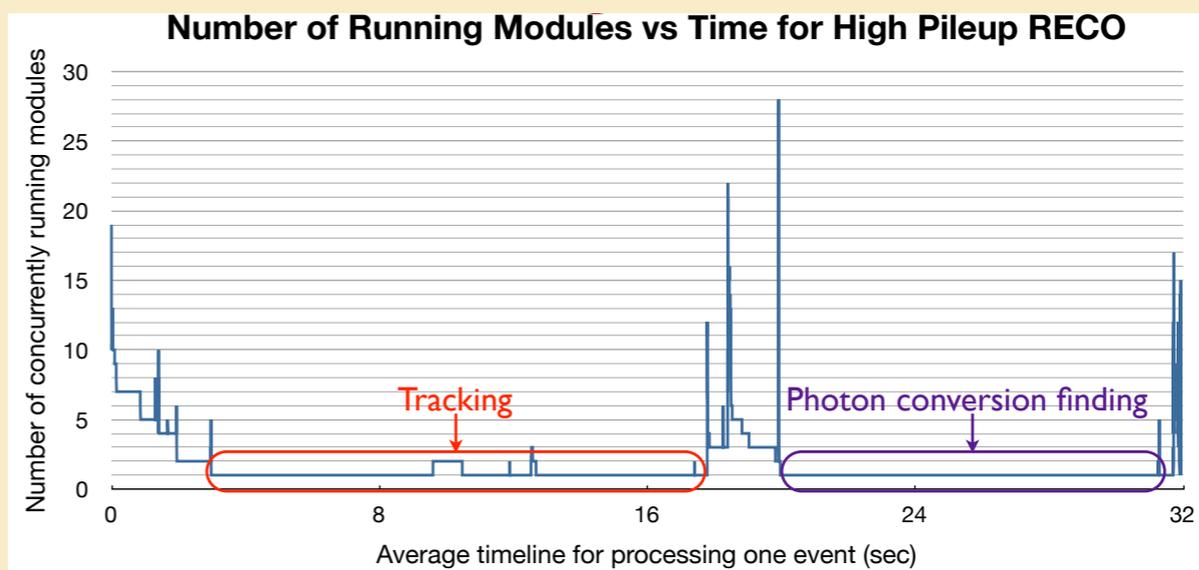
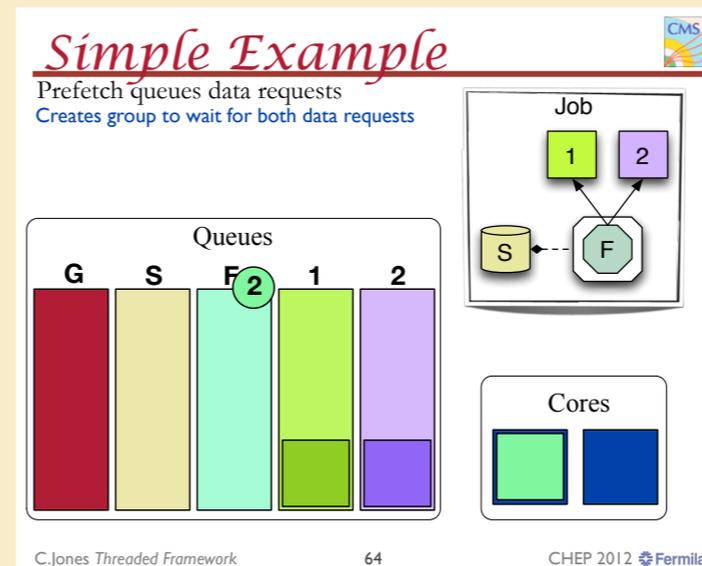
*Parallelize the steps within processing of one event*

**History: Parallelization of processing steps first mentioned in CHEP2010 summary**

CMS investigates threading frameworks

Jones

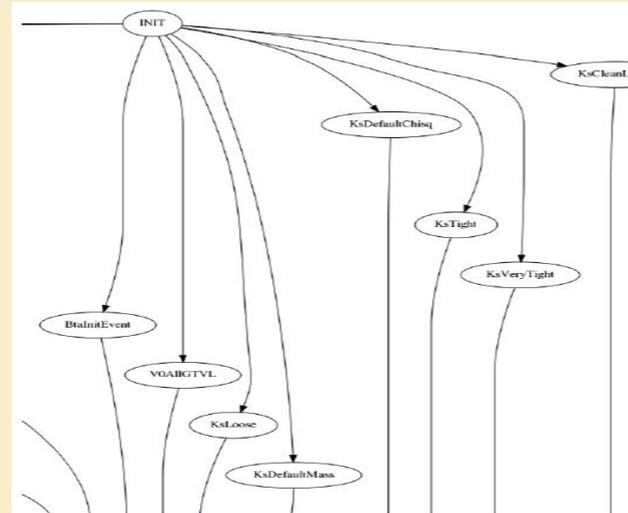
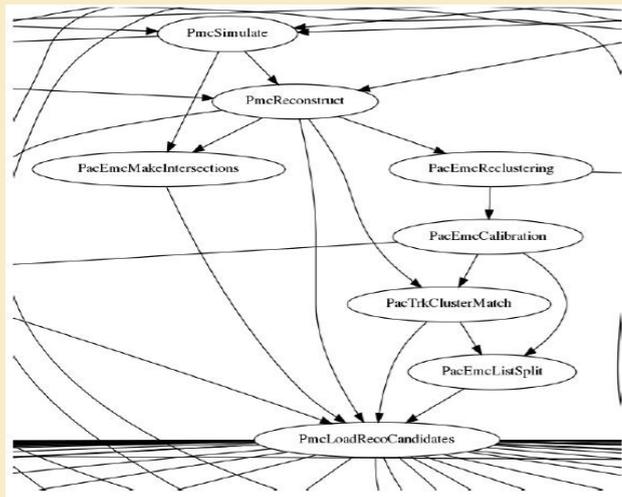
Tibdispatch (Apple):  
lightweight task queues with dependencies



**Parallelization deep within algorithms may be necessary for more improvements**

# Parallelize SuperB Fast Simulation

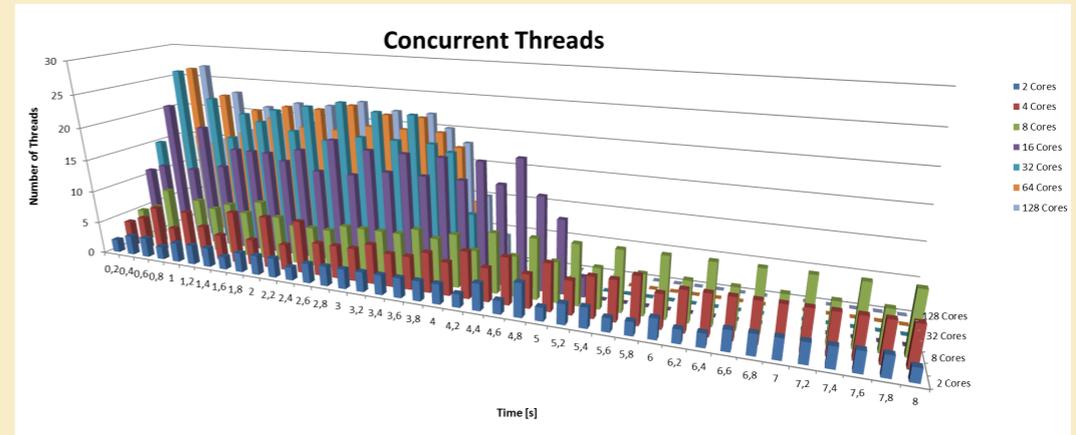
Corvo



**Dependencies make parallelization difficult**

**These can run in parallel**

**Use Intel Threading Building Blocks (TBB) and SuperB framework**



# Parallelize Particle Reconstruction (KFParticle)

Kulakov

## Structure of KFParticle

Low level (for developers, basic functionality)

- Transport functions
- Calculation of distances and deviations between particles, a particle and a point
- KF mathematics
- Constraints

**Used by ALICE and CBM**

Intermediate level (for advanced users)

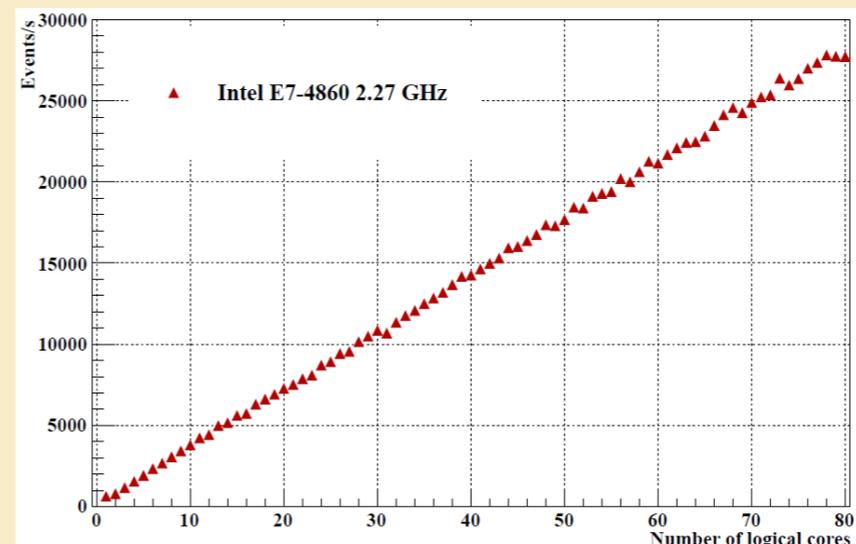
- Feasibility studies
- Reconstruction of particles
- Reconstruction of decay chains

High level or KFParticle-Light (for users and triggering)

- Reconstruction of standard decays ( $K^0$ , hyperons,  $D^0$ -mesons,...)
- Reconstruction of event topology
- On-line selection of events

Parallelized with TBB

Linear scalability on multicore machines

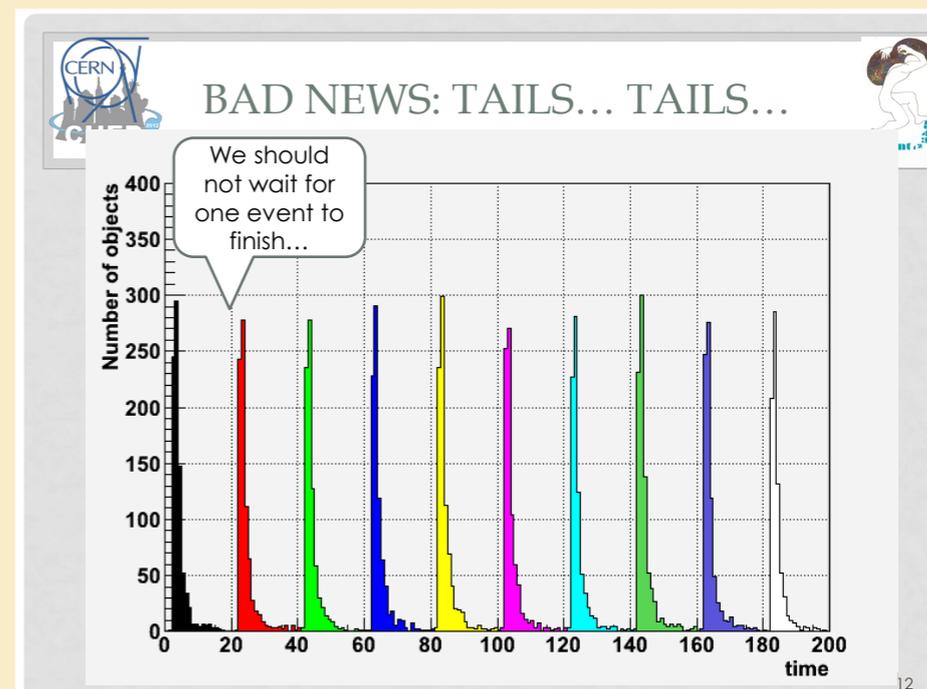


Given n threads each filled with 1000 events, run them on specific n logical cores, 1 thread per 1 core. AuAu mbias events at 25 AGeV

# Re-thinking Particle Transport in multicore environment

Complete rethinking required to take advantage of new hardware

e.g. propagate particles in the same volume all together (blur the boundary between events)



## But i/o can be troublesome

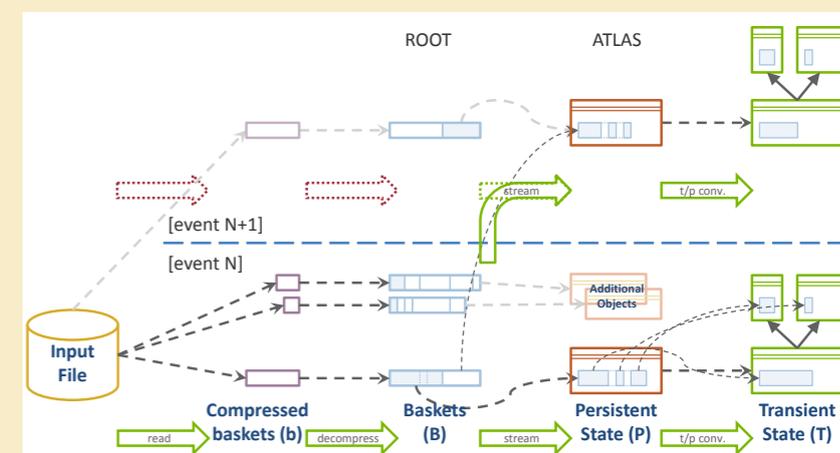
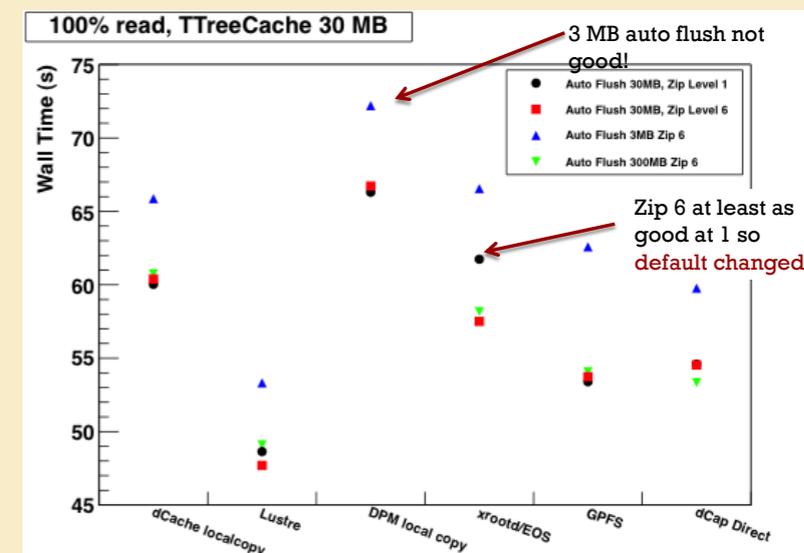
Bhimji & von Gemmeren

### ATLAS tests different Root i/o configurations and investigates multicore issues

e.g. zip level, autoflush, basket/branch organization (e.g. reduce the number of events per basket)

**Beware of decompressing the same basket multiple times**

### Careful optimization of event data storage and Root i/o configuration is necessary



# Multicore processing summary

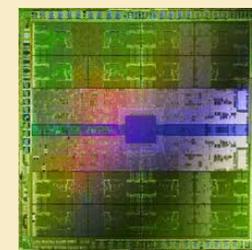
**Multicore processing is becoming a necessity**

**Many tools exist (libdispatch, TBB)**

**But there are some caveats (i/o)**

**CHEP2013 Prediction: Lots of reports about success of deep parallelization of algorithms**

# Processing on GPUs



GPUs can be amazingly fast for suitable algorithms  
(100x, 200x, ...)

**What algorithms are suitable? How to integrate into workflows?**

**History: First mention in CHEP10 summary**

## GPUs for and IceCube simulation

Skarlupka

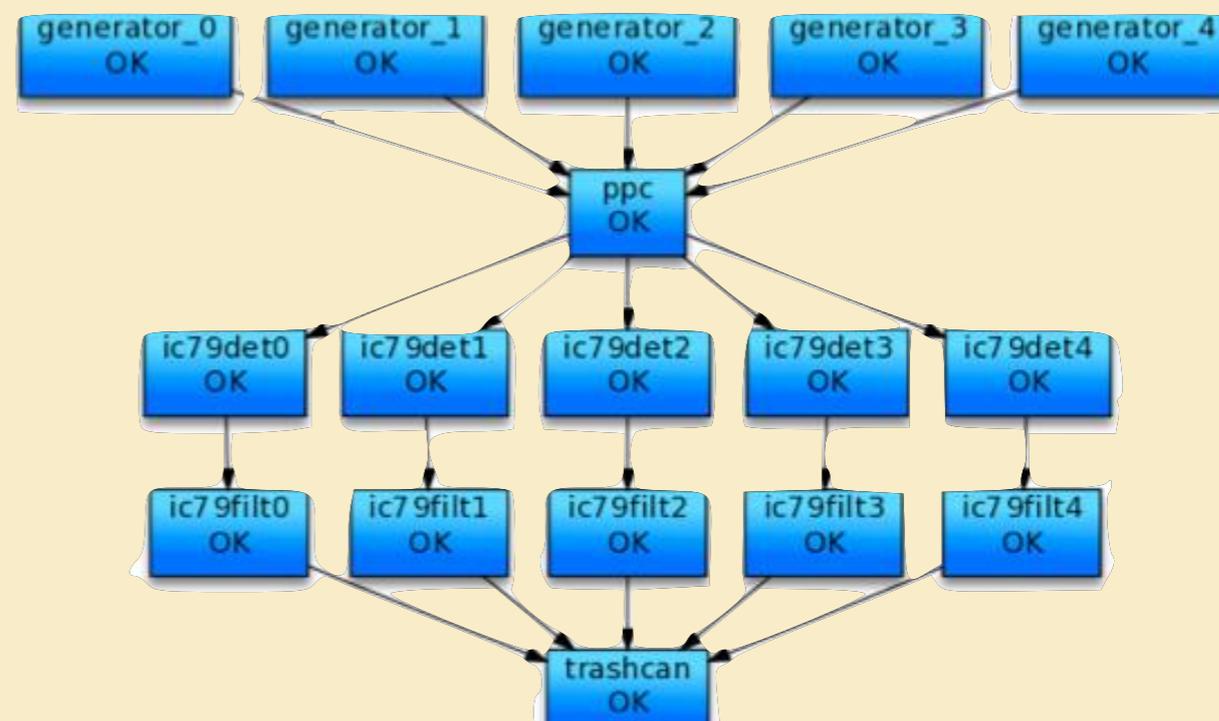
**25 GB photon “tables” library is anti-Grid**

**Do calculation in GPU cluster for photon propagation**

**150x speed improvement!**

**But can it be integrated in the simulation workflow?**

**Working with Condor team to incorporate GPUs in Condor cluster (configure, submit, create GPU slots)**



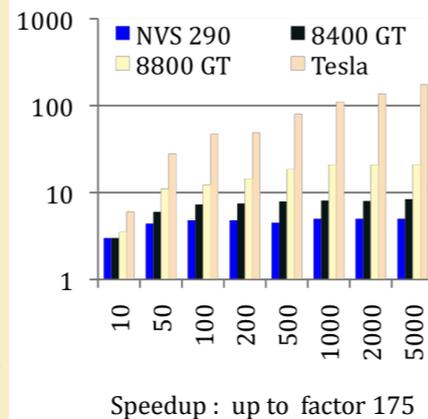
**No hardware trigger! Reduce 1 TB/s to 1 GB/s with realtime reconstruction at  $10^7$  events/s! Estimate 60K cores/experiment needed!**

**CBM: NVIDIA GPUs, use texture memory to hold magnetic field (interpolation for free!)**

**PANDA: Compare GPU to FPGA Helix tracking algorithm; rewritten for GPUs**

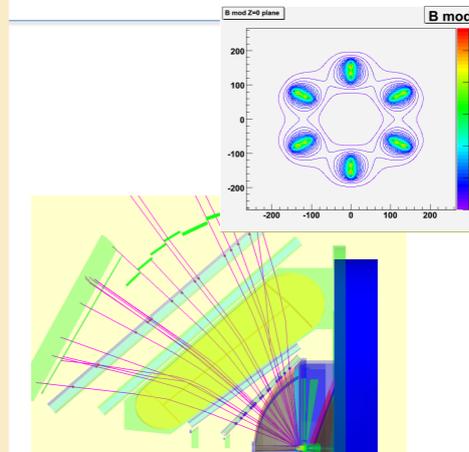
We start using more GPU specific features:  
Texture memory for field maps

Track propagation (RK4) using PANDA Field



Trk/Event	NVS 290	8400 GT	8800 GT	Tesla
10	3	3	3.5	6
50	4.4	6	11	28
100	4.8	7.3	12.3	47
200	4.8	7.5	14.5	49
500	4.5	7.9	18.5	80
1000	5	8.1	21	111
2000	5	8	21	137
5000	5	8.4	21	175

The same concept was applied using real data and existing field (HADES experiment at GSI)  
Track propagation (RK4) in magnetic field on GPU



Propagation /Event	Tesla (CPU/GPU)
10	11
50	15
100	15
200	24
500	34
700	41

Track Propagation Speedup by a factor 40

**For track finding: GPU is 200x faster than CPU  
GPU is 30% faster than FPGA**

**NOTE: GPU code still need optimization**

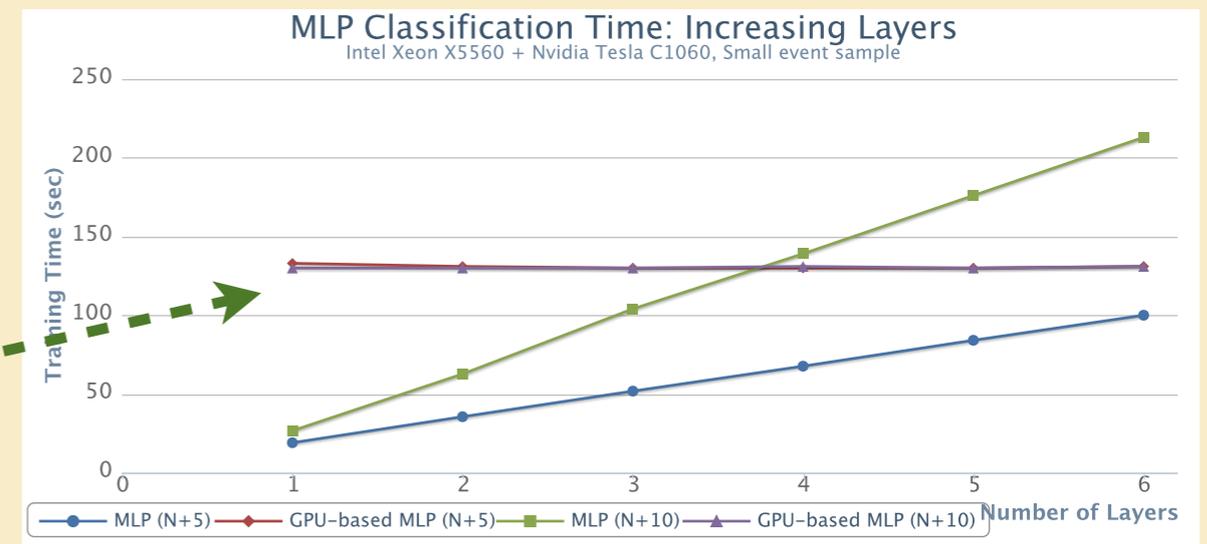
## Feasibility study for GPUs in TMVA

Washbrook

**Studying MLP (multi-layer perceptrons) artificial neural networks first**

**Can't parallelize training by events  
Instead parallelize by neurons**

**GPUs win for complex NNs**



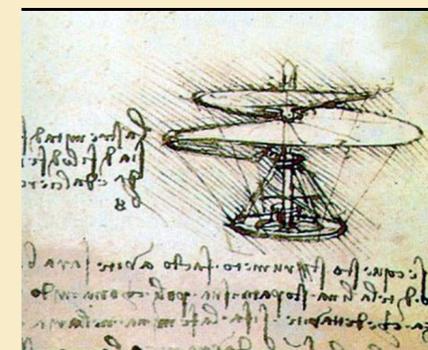
# GPU processing summary

**Speedups are incredibly tantalizing and looks like in reach**

**CHEP2013 Prediction : GPU processing matures and will be fully integrated into workflows**

# Common frameworks

A Framework supplies services for analysis code:  
i/o handling, persistency, event data model, dispatching,  
object links, ...



Physicist can concentrate on algorithms and results

Big experiments have resources to write their own frameworks;  
Small experiments may not

Several labs are offering many-experiment shared frameworks

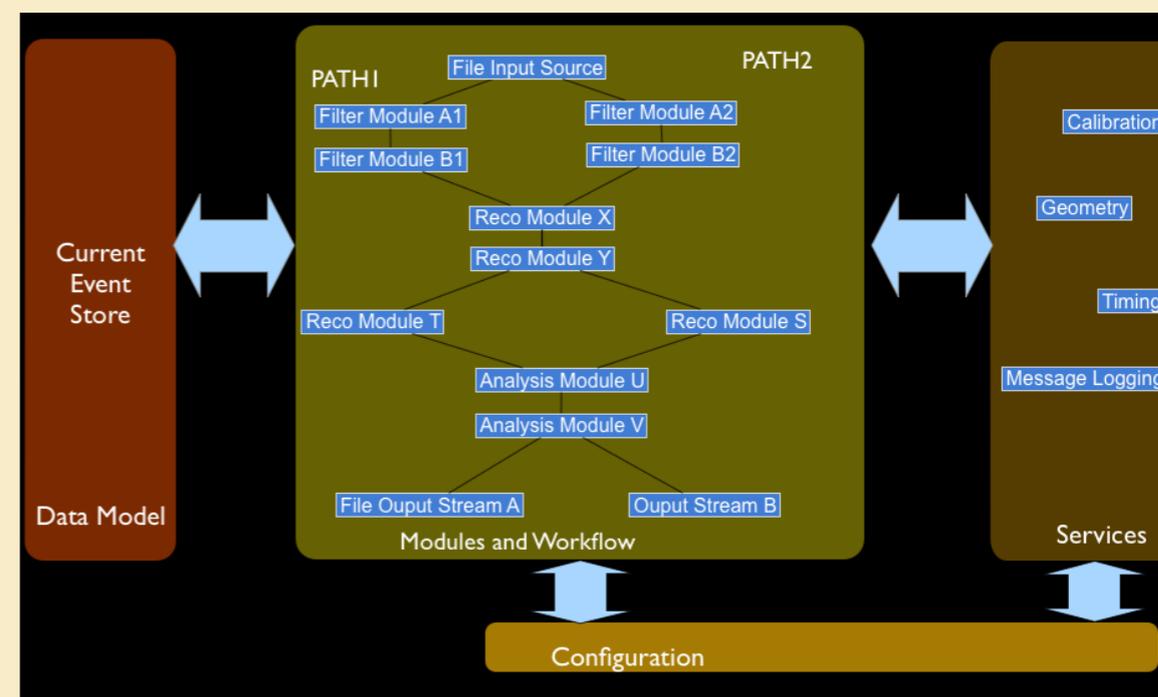
art: Common framework written by Fermilab SCD

Green

A “lite” forked version of the CMS framework

Used by Fermilab Intensity Frontier Experiments  
(NOvA, g-2, Mu2e, LBNE, MicroBoone) and  
inquiries from SuperB

New for Fermilab SCD, sociology  
working out well so far



New for this CHEP

# FairRoot: GSI's common analysis framework

Based on Root (uses Root's executable)

Proof Compatible, GPU support

Experiments write code and macros that specialize FairRoot. e.g. CBM, PANDA, ASYEOS, R3B

History: First mentioned in CHEP2007 summary

## FairRoot

5

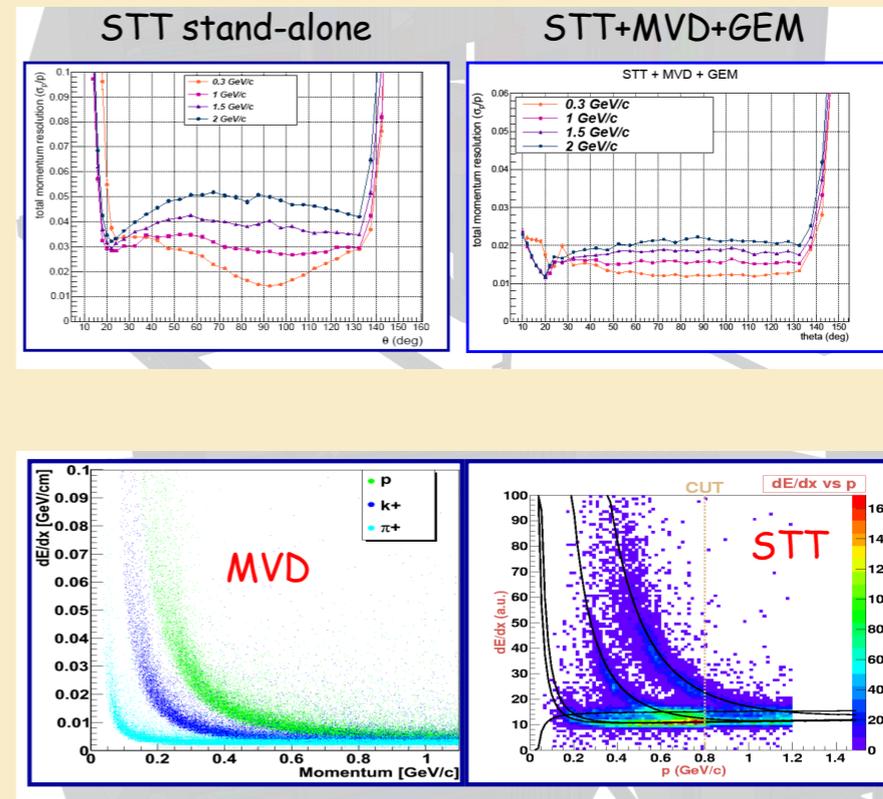
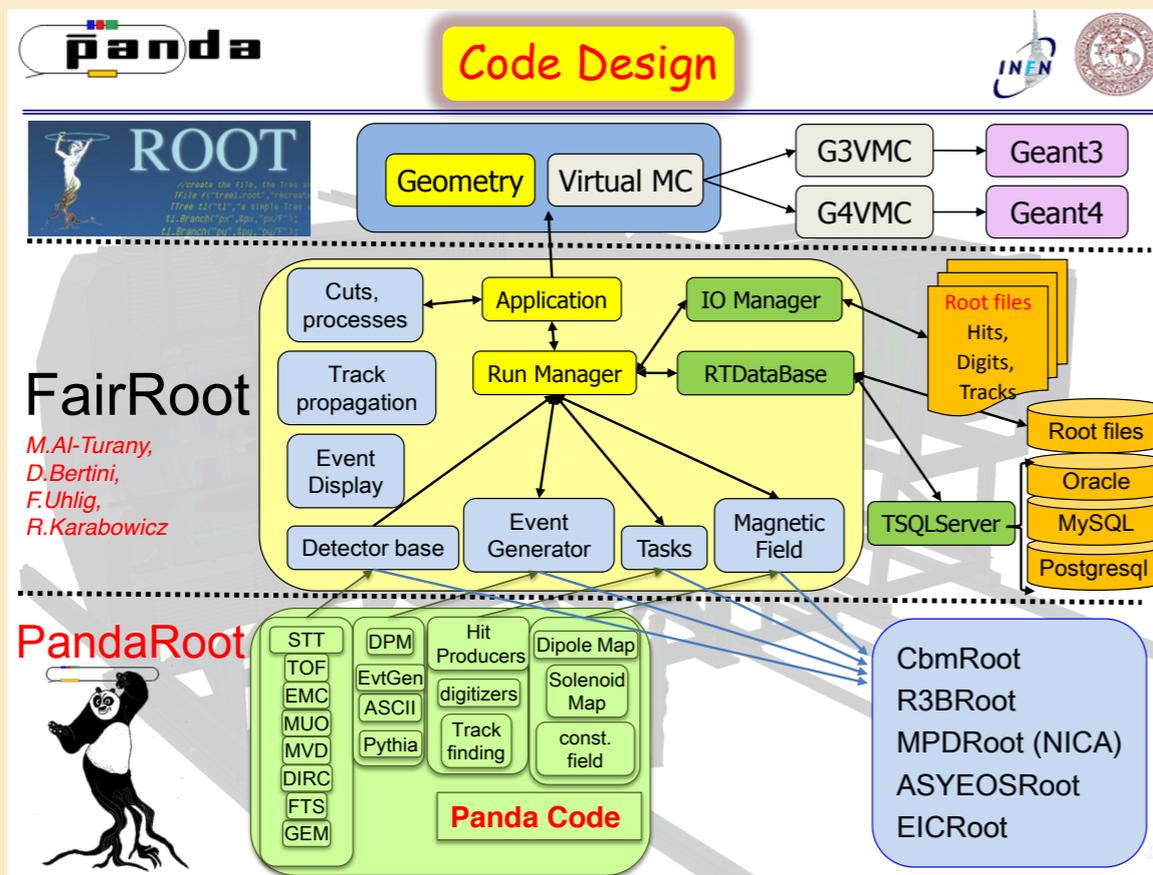
- Framework for simulation, reconstruction and data analysis
- Very flexible
  - No executable
    - Use plug-in mechanism from Root to load libraries only when needed
    - Use Root macros to define the experimental setup or the tasks for reconstruction/analysis
    - Use Root macros to set the configuration (Geant3, Geant4, ...)
  - No fixed simulation model
    - Use different simulation models (Geant3, Geant4, ...) with the same user code (VMC)

Florian Uhlig

CHEP 2012, New York

22.05.12

# PandaRoot



# Common framework summary

**Labs are providing this important software to small experiments**

**Important collaborations with their experiments**

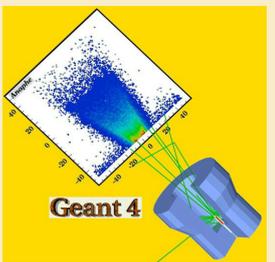
**CHEP2013 Prediction: FairRoot and variants become more sophisticated; art matures with a large use base**

# Simulations

Accurate simulations are critical for achieving results

How do we know the simulations are correct?

How can the code be more efficient?

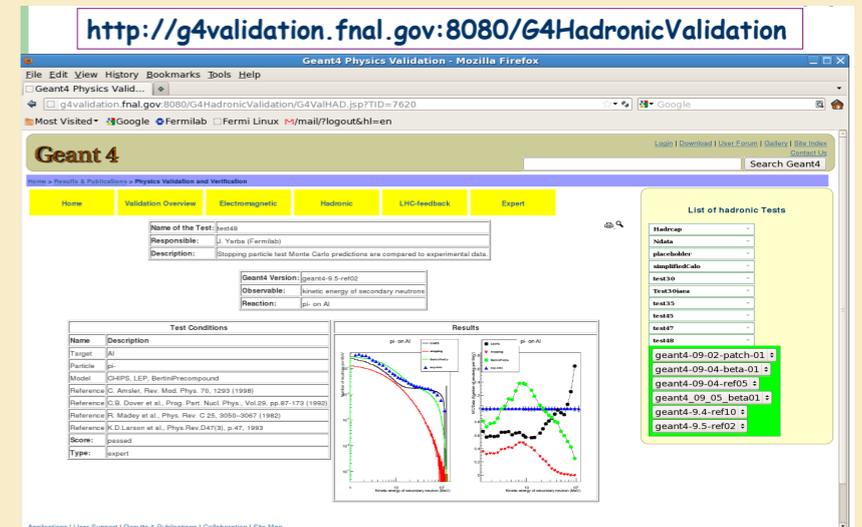
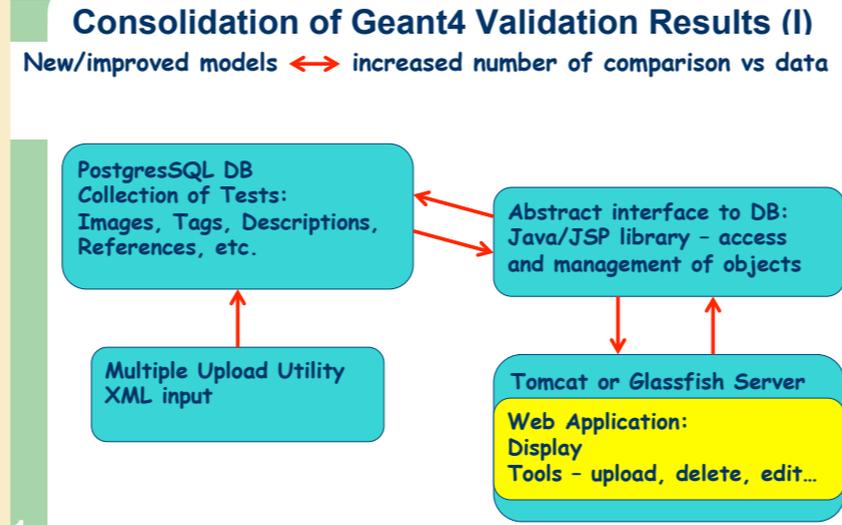
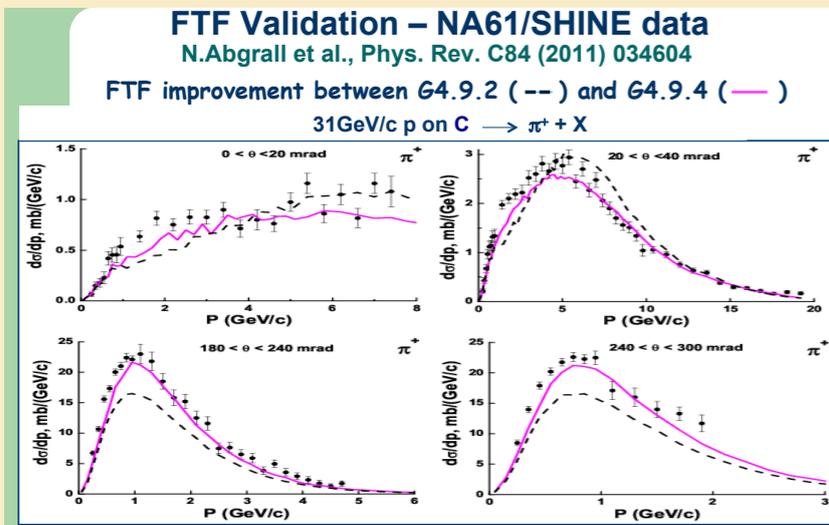


## Geant4 Hadronic Physics and Validation

Yarba

Validation of hadronic models, processes (model+xs) and physics lists

Compare models to each other in overlap regions and to experiments and organize results nicely



History: Mention of GEANT validation at nearly every CHEP

Coupling, Dependencies (must pull in a big system, like geometry, to test)

Duplicated Physics (same physics in different places)

Duplicated Numbers (same constants defined in multiple places)

Myth of slow models (a component algorithm can be the culprit)



## Smells

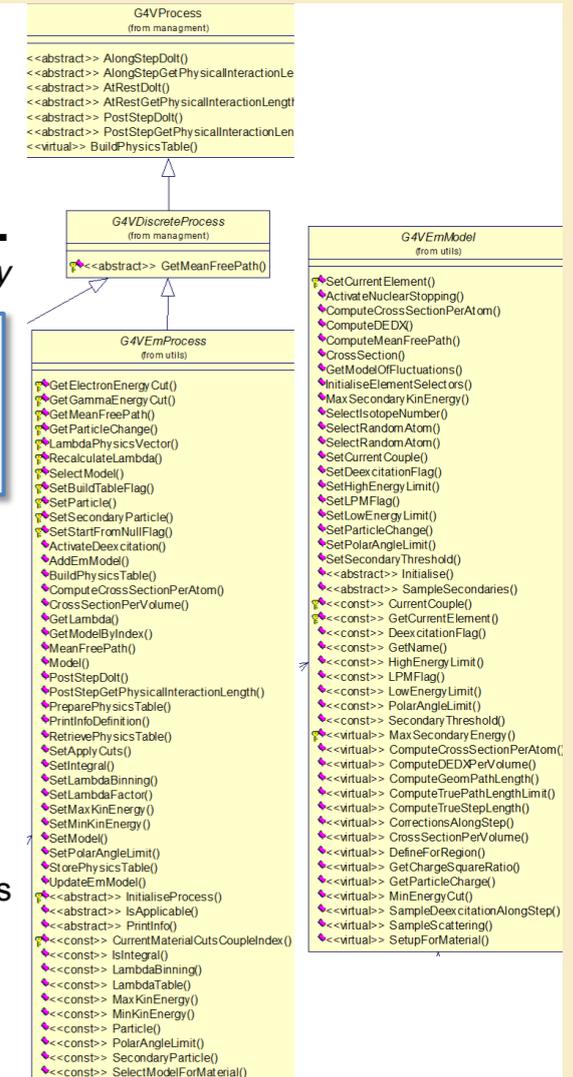
If it stinks, change it.

Grandma Beck, discussing child-rearing philosophy

M. Fowler, K. Beck et al.,  
**Refactoring: Improving the Design of Existing Code**

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchies
- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest

Maria Grazia Pia, INFN Genova



Prune, Trash and redo, Eliminate algorithms and use data

But none of this is easy, but one wins with with improved accuracy and performance

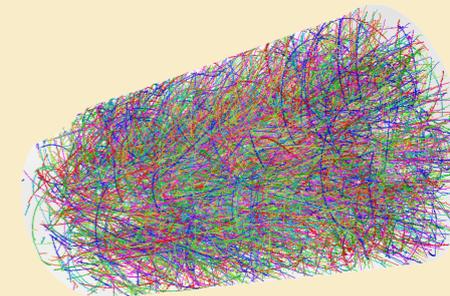
# Simulations summary

**Validation of physics becomes more sophisticated and better organized**

**Opportunities exist for improving accuracy and performance by improving the code**

**Prediction for CHEP2013: Geant4 validation continues. Perhaps reports of code improvements. Geant5?**

# Reconstruction algorithms



## Sophisticated detectors require sophisticated algorithms

### “Swimming” to correct lifetime bias from LHCb trigger

Cattaneo

Trigger selects on heavy flavor long lifetime, thus sculpting the lifetime distribution – needs to be understood

Replay each event through the trigger, varying the lifetime

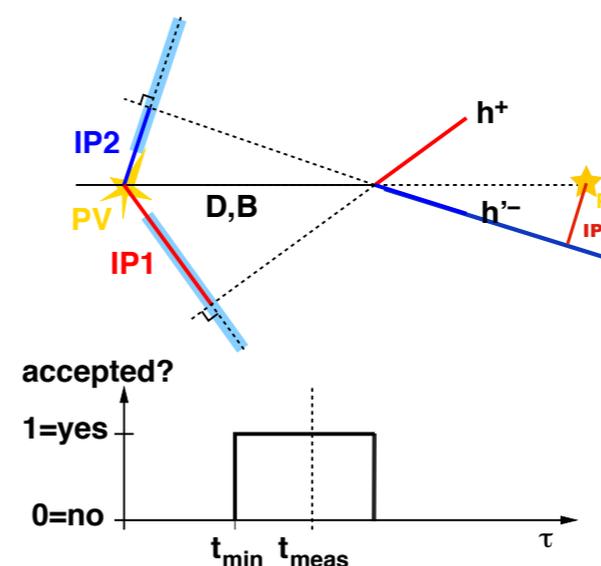
Obtain event-by-event lifetime acceptance function

Run the actual HLT code offline (reproducible software trigger is key)

Automated job handling for swimming

Processed 100M events (1%)

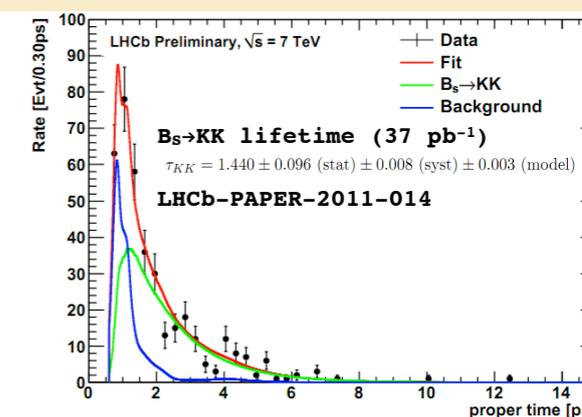
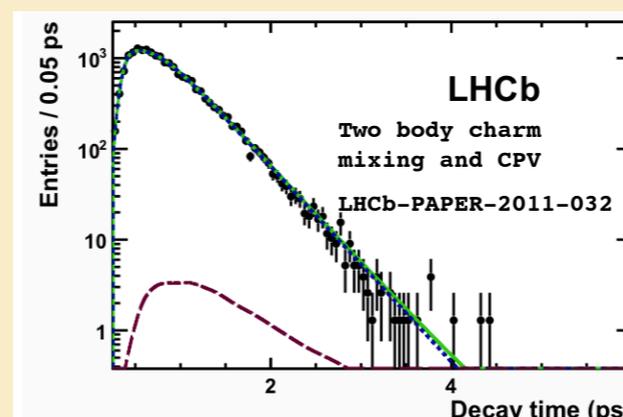
### Swimming in action



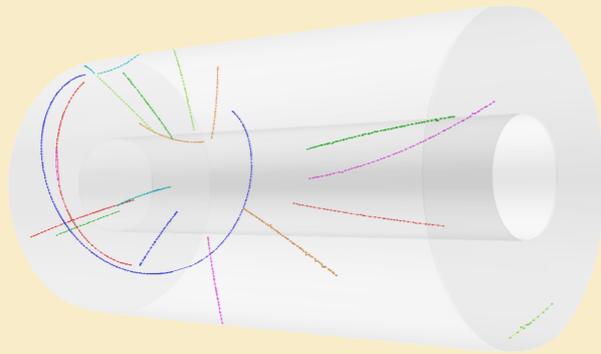
Because we can reproduce the trigger decisions offline, we can measure lifetime biases in a data driven way offline

Get an event-by-event acceptance by replaying the trigger decision for the full range of possible B/D lifetimes

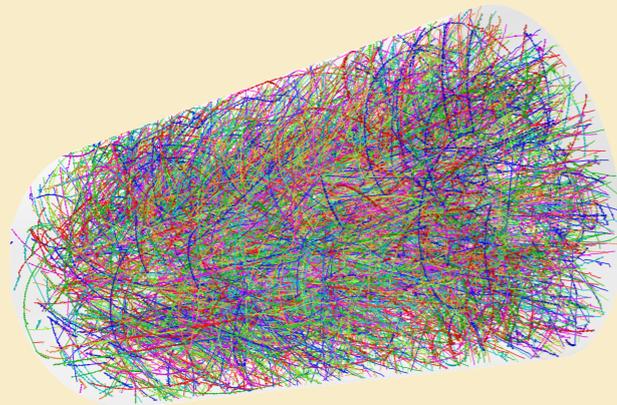
No trigger emulation needed, correct alignment and detector conditions automatically taken into account.



Easy



Not easy



GEM-TPC Reconstruction Chain Clustering Pattern Recognition PR Performance Event Deconvolution Conclusion

## Target Pointing

TUM Technische Universität München

- Tracks with their POCA close enough to the IP survive.
- With this technique, the amount of background tracks can be reduced by almost one order of magnitude.
- In this step it is more important to retain all physics tracks than to reject all background tracks.

GEM-TPC Reconstruction Chain Clustering Pattern Recognition PR Performance Event Deconvolution Conclusion

## Conclusion

TUM Technische Universität München

- 3-dimensional clustering and pattern-recognition algorithms for a high-rate GEM-TPC
- Efficient at *high track densities*
- Finds *all kinds of track topologies*.
- *Robust* against drift distortions.
- Excellent *seed values* for event-deconvolution and track-fitting
- Event-deconvolution feasible

Johannes Rauch on behalf of the GEM-TPC Collaboration — Pattern Recognition in a High Rate GEM-TPC 29

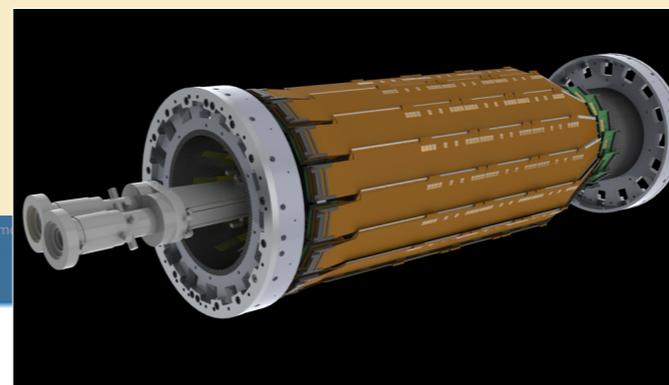
## Belle II Inner Tracking

New inner tracker (6 layers of silicon)

Very low momentum tracks, and strong material effects

Background detection with a Deterministic Annealing Filter (iterated Kalman filter with weights and annealing)

Nadler



HEPHY Institut für Hochenergiephysik

## Summary of results

- At moderate track energies BG detection works very well even if a track candidate contains as much BG as real hits and the  $\sigma$  of Impact parameters is only increased by 13 % to 20 %
- A single BG hit per track in any layer besides layer 1 has virtually no effect on the impact parameters.
- At low track energies the picture is worse. The main reason is the strong MSC.
- BG detection very different depending on layer: 2 still quite good, 6 comes close to random guessing

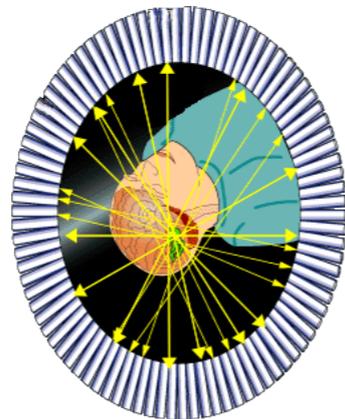
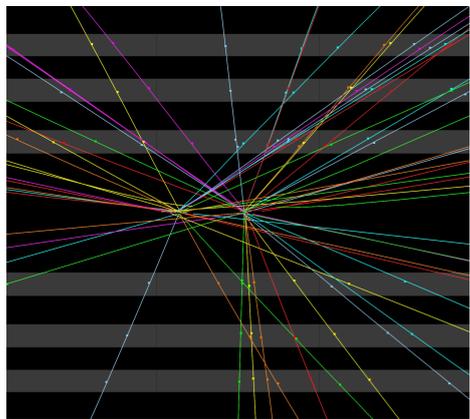
Moritz Nadler 22 / 23 HEPHY Wien & Belle II Collaboration

# Medical Imaging Inspired Vertex Detection

e.g. find primary vertices for  $H \rightarrow \gamma\gamma$

## Comparison: HEP & PET

	HEP	Positron Emission Tomography
Problem	Position of reaction/interaction	Position of tumor
Data	Tracks	Photon pairs from $e^+e^- \rightarrow \gamma\gamma$
Amount of data	$\langle N_{\text{Trk}}/V_{\text{tx}} \rangle \approx 20$	$> 10^6$ , <b>more is better</b>
Methods	Adaptive fitting / <b>Topological finding</b>	<b>Filtered Backprojection</b> → inv. Radon-transformation



universität**bonn**

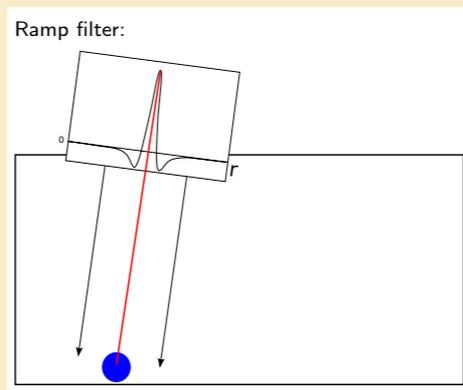
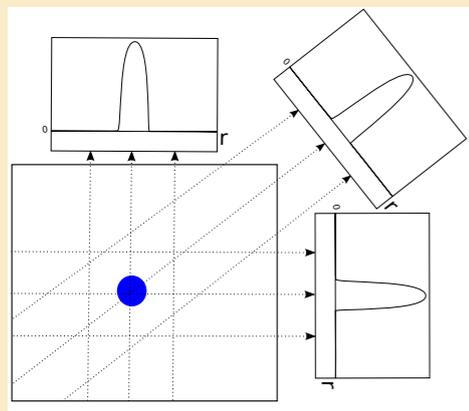
Navigation icons: back, forward, search, etc.

## Results

- MI is a **promising new ansatz** for vertex finding at high luminosities / with many tracks
- Comparison to adaptive vertex fitter (conditions similar to ATLAS/CMS)
  - ▶ **Higher efficiency and purity** at LHC design lumi + beyond
  - ▶ **Much faster** for high numbers of tracks / vertices (linear)
  - ▶ **No advantage** at low numbers of vertices

## Outlook:

- Fit found vertices to improve resolution
- Study effects of non-Gaussian track errors
- Applicable to b-tagging / boosted Higgs?



# Reconstruction algorithms summary

**Many novel new algorithms presented**

**Prediction for CHEP2013: Introduction of new algorithms is a constant at CHEP**

# Everything else

## Functional/declarative language for Root plotting

Watts

**Your postdoc has all the fun? Your student thinks you can't make a plot?**

```
var dsname = "JetStream";
var files = DataSetFinder.FindROOTFilesForDS(dsname);

var data = ROOTLINQ.QueryableCollectionTree.Create(files);

var p = from evt in data
        from j in evt.Jets
        where Math.Abs(j.Eta) < 2.0
        select j;
p.Plot("jetpT", "Jet pT", 100, 0.0, 100.0, j => j.Pt / 1000.0);
```

Get  
access to  
a TChain

All jets with  
 $|\eta| < 2.0$

Plot  $p_T$  in GeV

**Use C# (.NET language)  
with Language Integrated  
Query (LINQ)**

**SQL like queries**

**Expressions translated to C++**

**Caching**

**Composability (reuse selections,  
make dynamic selections)**

**Compatible with PROOF**

**Have thought? Plot it in minutes  
(for the professor - don't tell the postdocs & students :-)**

# Everything else summary

**Interesting “everything else” posters:**

**Physics data processing with protocol buffers**

**Creating 3D content in PDF**

# Summary of Event Processing

**Excellent talks and posters on a broad range of topics  
Congratulations to all contributors!**

**Enormous effort occurring in all of these areas**

**The general tone: Our software works, but we need to make it **go faster! Speed Matters!****

**Multicores and GPUs are the answer**

**CHEP2013: The parallelization solutions will come to fruition**

**Smaller experiments will take advantage of these improvements with common frameworks**

**Geant4 physics and code will improve; Geant5?**

**Sophistication of algorithms continue to increase (not necessarily complexity)**

**Professors make all the cool plots - postdocs/students get scooped :-)**