

Track Summary Event Processing, Simulation and Analysis

Peter Elmer (Princeton University)

Rolf Seuster (TRIUMF)

Florian Uhlig (GSI)

Statistics

CHEP 2013

59(53) contributions

31(27) poster

28(26) talks

1 Vidyo presentation

CHEP 2012

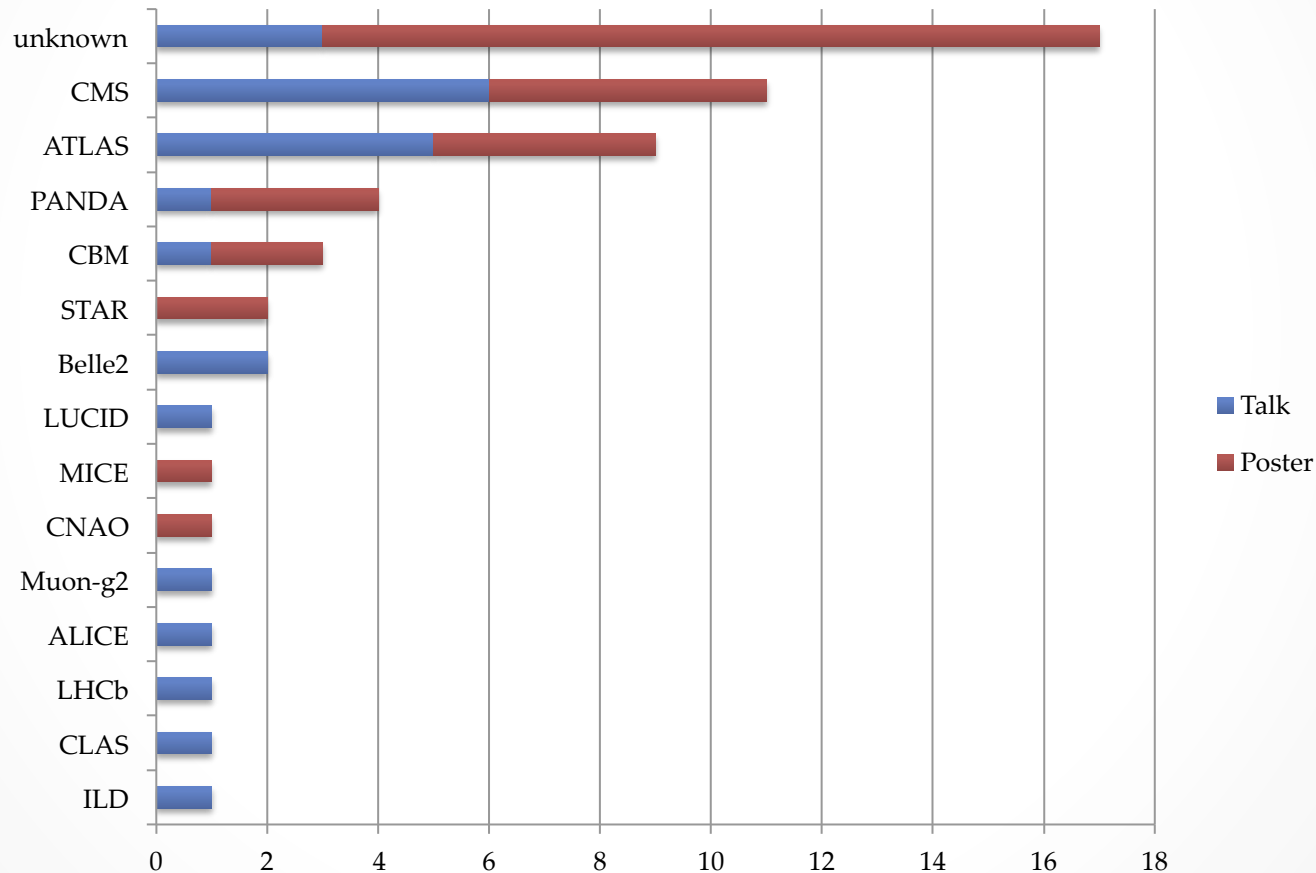
84 contributions

64 posters

20 talks

- Between 20 and 50 participants in the different sessions
 - Much less then in previous CHEPs
 - Many people probably in track 5: Software Engineering, Parallelism & Multi-Core
- Many people jumped between the different tracks
- In the presentation I will focus on the talks and try to give an overview over the topics not the talks
-

Which experiments are represented?



Outline

- Common Frameworks
- Concurrency
- Algorithms
- Pileup Simulation
- Future Simulation for LHC
- Everything else

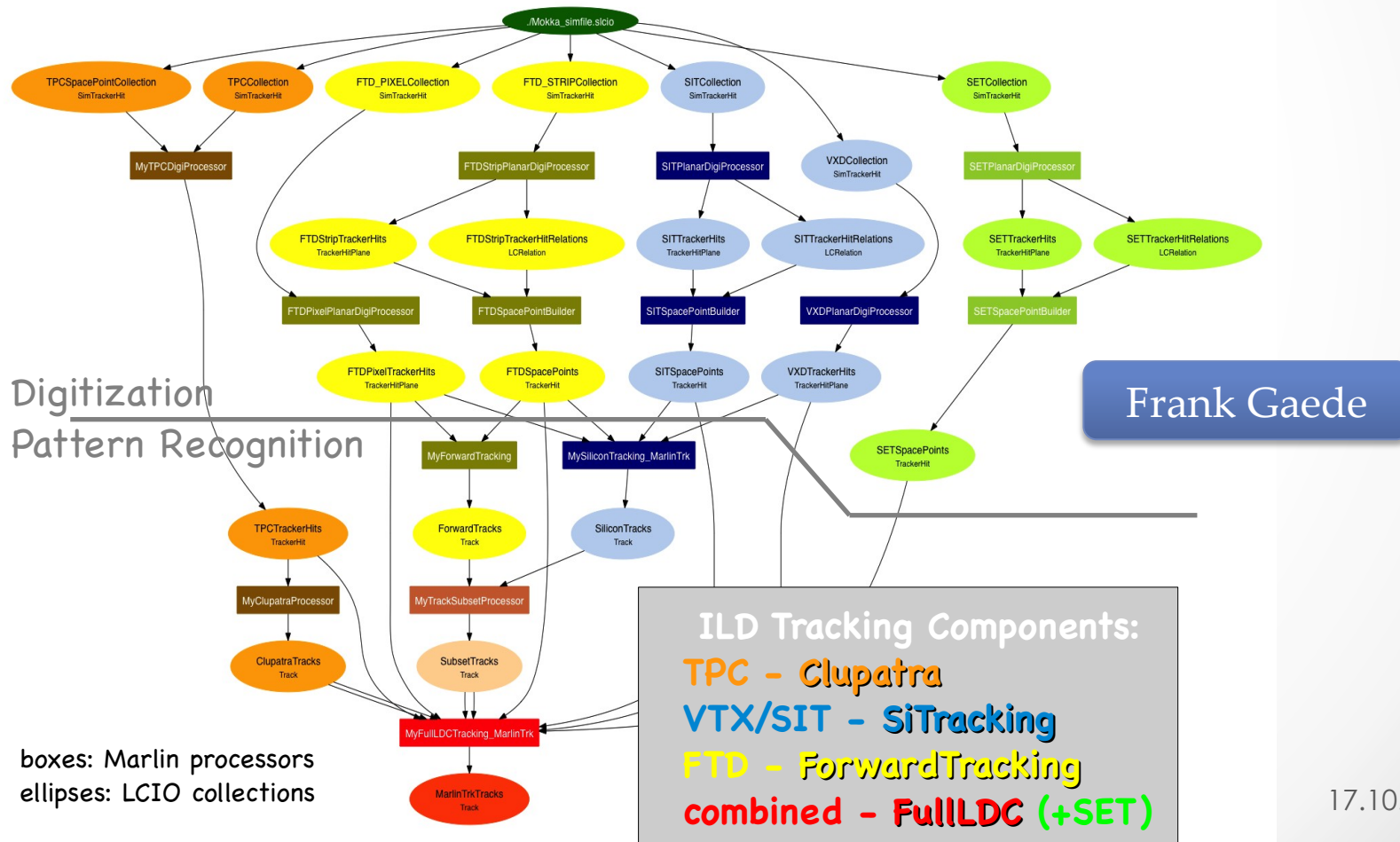
Common Frameworks

- Many different frameworks presented
 - For sure the big and well known ones ATLAS(Gaudi) LHCb(Gaudi), CMS
 - Many others
 - International Large Detector@ILC

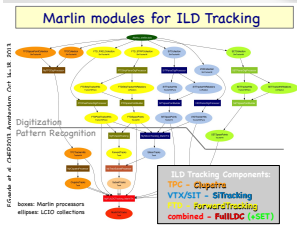
Common Frameworks

Marlin modules for ILD Tracking

F.Gaede et al. CHEP2013. Amsterdam, Oct 14-18, 2013

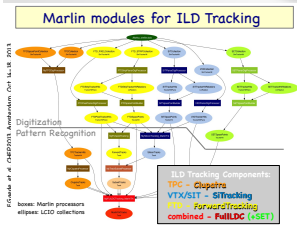


Common Frameworks

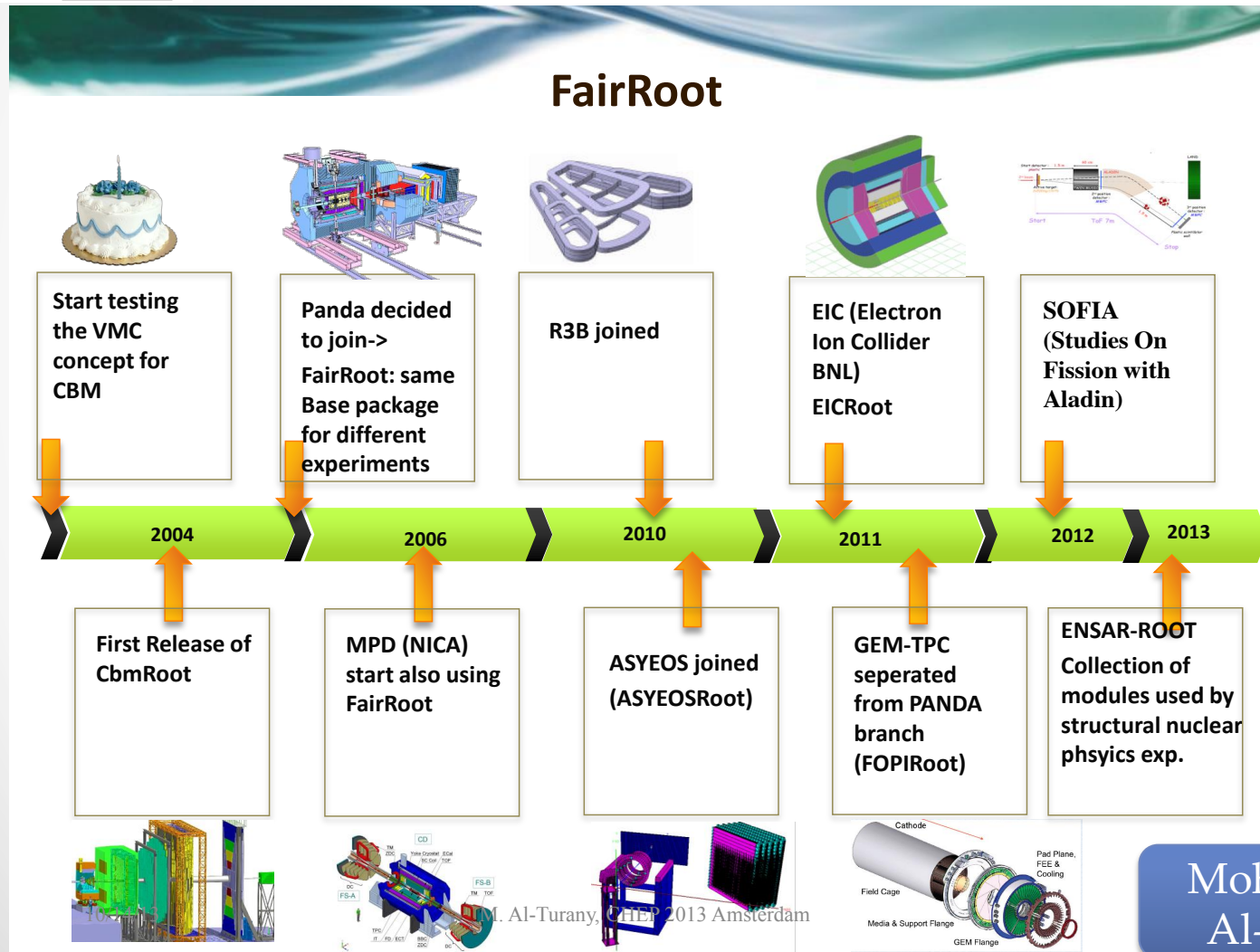


- Many different frameworks presented
 - For sure the big ones ATLAS(Athena), LHCb(Gaudi), CMS
 - Many smaller ones
 - International Large Detector@ILC(Marlin)
 - CBM, Panda (FairRoot)
 - Many other experiments meanwhile

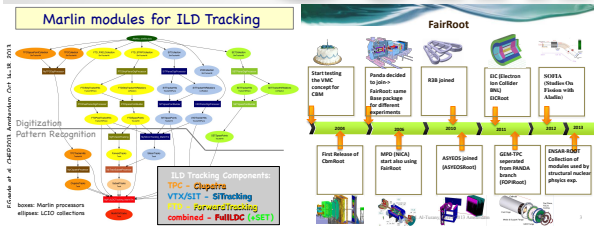
Common Frameworks



FairRoot

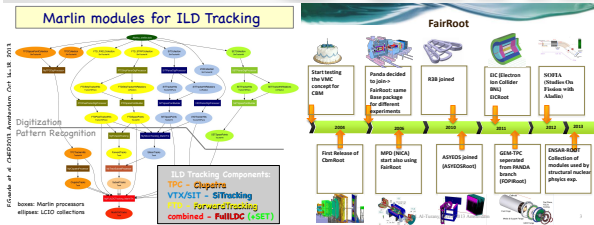


Common Frameworks

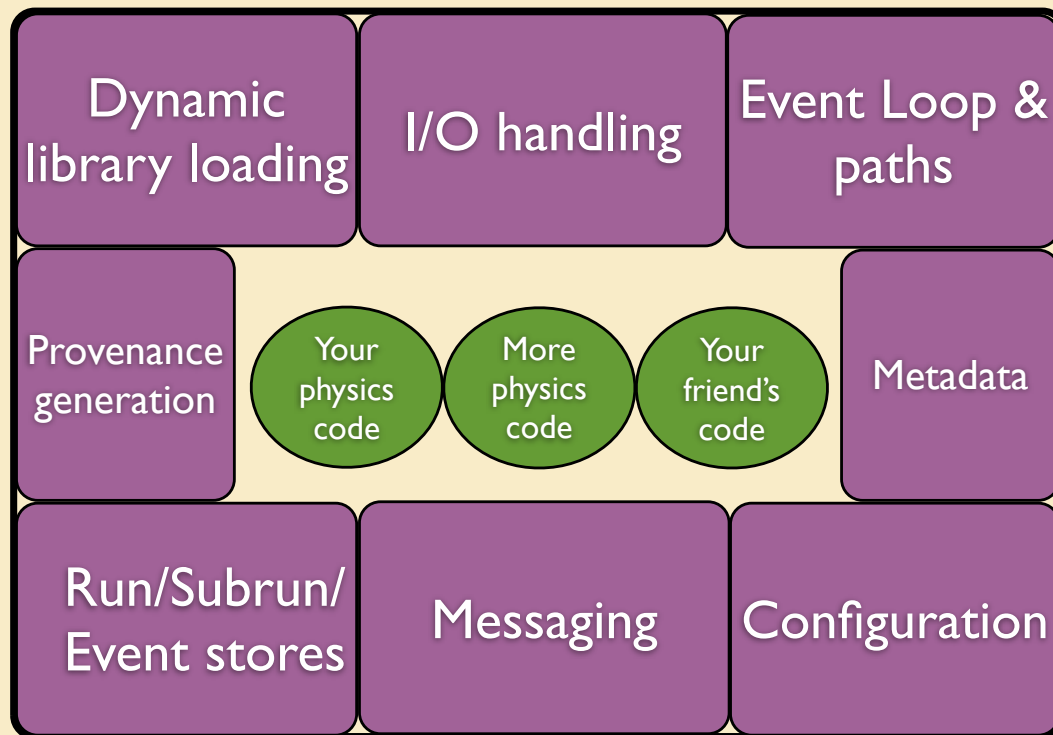



- Many different frameworks presented
 - For sure the big ones ATLAS(Athena), LHCb(Gaudi), CMS
 - Many smaller ones
 - International Large Detector@ILC(Marlin)
 - CBM, Panda (FairRoot)
 - Many other experiments meanwhile
 - Muon g-2 (ArtG4 based on Art, lite fork from CMS)
 - NOvA, Mu2e, MicroBoone, LBNE, ...


Common Frameworks



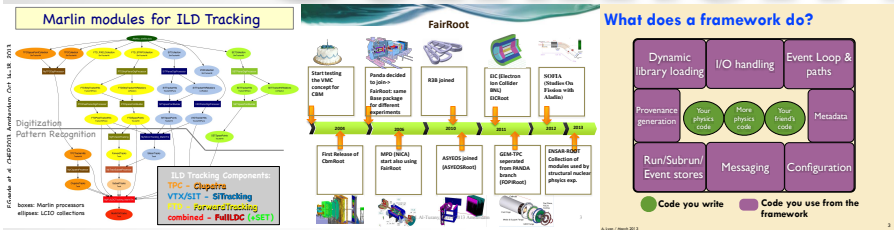
What does a framework do?



 **Code you write**

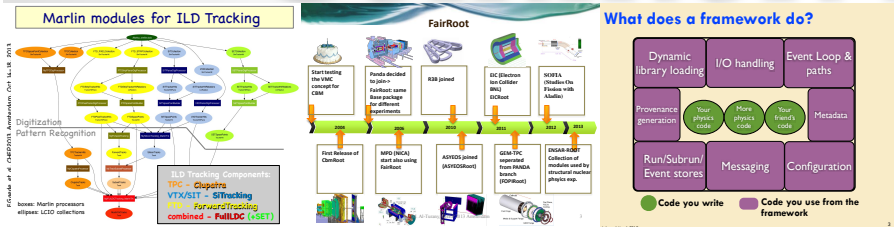
 **Code you use from the framework**

Common Frameworks



- Many different frameworks presented
 - For sure the big ones ATLAS(Athena), LHCb(Gaudi), CMS
 - Many smaller ones
 - International Large Detector@ILC(Marlin)
 - Muon g-2 (ArtG4 based on Art, lite fork from CMS)
 - NOvA, Mu2e, MicroBoone, LBNE, ...
 - CBM, Panda (FairRoot)
 - Many other experiments meanwhile
- Other frameworks
 - Geant4

Common Frameworks

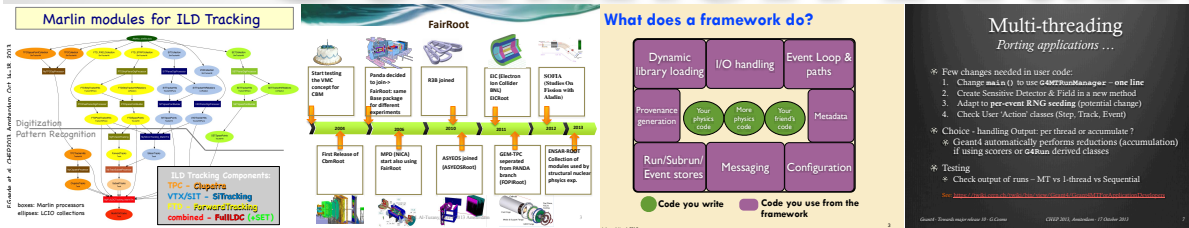


Multi-threading *Porting applications ...*

- ❁ Few changes needed in user code:
 1. Change `main()` to use `G4MTRunManager` – **one line**
 2. Create Sensitive Detector & Field in a new method
 3. Adapt to **per-event RNG seeding** (potential change)
 4. Check User 'Action' classes (Step, Track, Event)
- ❁ Choice - handling Output: per thread or accumulate ?
 - ❁ Geant4 automatically performs reductions (accumulation) if using scorers or `G4Run` derived classes
- ❁ Testing
 - ❁ Check output of runs – MT vs 1-thread vs Sequential

See: <https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4MTForApplicationDevelopers>

Common Frameworks



The image contains four small slides. The first slide, 'Marlin modules for ILD Tracking', shows a complex flowchart of modules. The second slide, 'FairRoot', shows a timeline of development milestones from 2006 to 2013. The third slide, 'What does a framework do?', is a diagram with boxes for 'Dynamic library loading', 'I/O handling', 'Event Loop & paths', 'Provenance generation', 'Run/Subrun/Event stores', 'Messaging', and 'Configuration', with a legend for 'Code you write' and 'Code you use from the framework'. The fourth slide, 'Multi-threading Porting applications ...', lists several changes needed in user code for multi-threading.

- Many different frameworks presented
 - For sure the big ones ATLAS(Athena), LHCb(Gaudi), CMS
 - Many smaller ones
 - International Large Detector@ILC(Marlin)
 - Muon g-2 (ArtG4 based on Art, lite fork from CMS)
 - NOvA, Mu2e, MicroBoone, LBNE, ...
 - CBM, Panda (FairRoot)
 - Many other experiments meanwhile
- Other frameworks
 - Geant4
 - RooStats, RooFit

Common Frameworks

Marlin modules for ILD Tracking

FairRoot

What does a framework do?

- Dynamic library loading
- I/O handling
- Event Loop & paths
- Provenance generation
- Run/Subrun/Event stores
- Message
- Configuration
- Metadata

Code you write (green) vs Code you use from the framework (purple)

Multi-threading

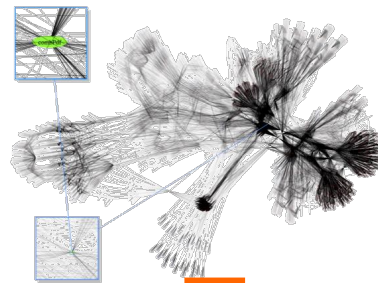
Porting applications ...

- * Few changes needed in user code:
 1. Change `wait()` to `use getStreamManager - one line`
 2. Create Sensitive-Detector & Field in a new method
 3. Adapt to per-event RNG seeding (potential change)
 4. Check User Action names (log, Track, Event)
- * Choice - handling Output: per thread or accumulate?
 - * Geant4 automatically performs reductions (accumulation) if using scorers or G4Run derived classes
- * Testing
 - + Check output of runs - MT vs 1-thread vs Sequential

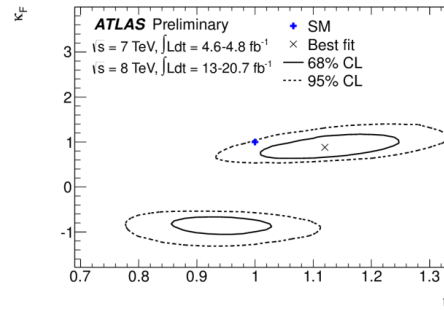
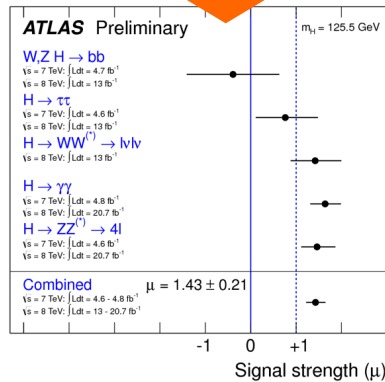
An excursion – Collaborative analyses with workspaces

- Workspaces allow to share and modify very complex analyses with very little technical knowledge required
- Example: Higgs coupling fits

Full Higgs model



Signal strength in 5 channels



Confidence intervals on Higgs fermion, boson couplings



$$\sigma(gg \rightarrow H) * BR(H \rightarrow \gamma\gamma) \sim \frac{\kappa_F^2 \cdot \kappa_V^2 (\kappa_F, \kappa_V)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \rightarrow qq'H) * BR(H \rightarrow \gamma\gamma) \sim \frac{\kappa_V^2 \cdot \kappa_F^2 (\kappa_F, \kappa_V)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(gg \rightarrow H) * BR(H \rightarrow ZZ^{(*)}, H \rightarrow WW^{(*)}) \sim \frac{\kappa_V^2 \cdot \kappa_V^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \rightarrow qq'H) * BR(H \rightarrow ZZ^{(*)}, H \rightarrow WW^{(*)}) \sim \frac{\kappa_V^2 \cdot \kappa_V^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \rightarrow qq'H, VH) * BR(H \rightarrow \tau\tau, H \rightarrow b\bar{b}) \sim \frac{\kappa_V^2 \cdot \kappa_F^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

Reparametrize in terms of fermion, boson scale factors

Common Frameworks

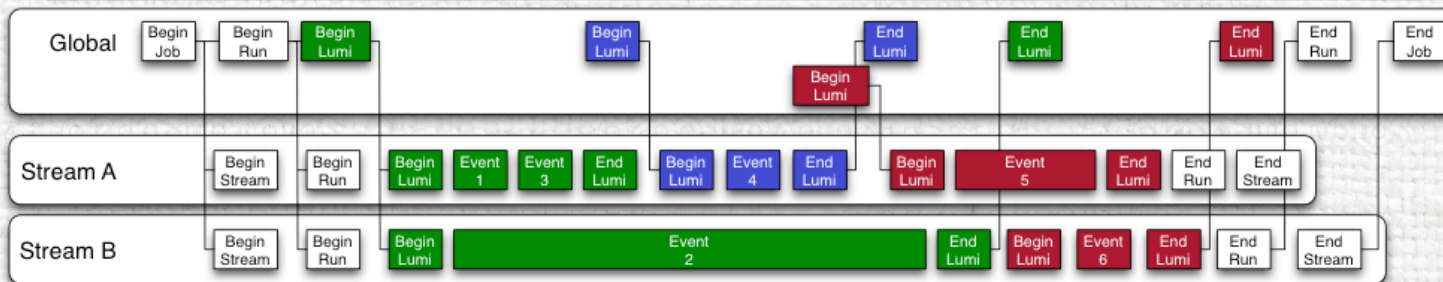
- Many different frameworks presented
 - For sure the big ones ATLAS(Athena), LHCb(Gaudi), CMS
 - Many smaller ones
 - International Large Detector@ILC(Marlin)
 - Muon g-2 (ArtG4 based on Art, lite fork from CMS)
 - NOvA, Mu2e, MicroBoone, LBNE, ...
 - CBM, Panda (FairRoot)
 - Many other experiments meanwhile
- Other frameworks
 - Geant4
 - RooStats, RooFit
- DRY, Code reuse, Consolidation
 - Make better use of your resources (manpower, money, ...)
 - More help from other users
 - Benefit from improvements done by your colleagues
-

Concurrency

- CHEP2013 Prediction: Lots of reports about success of deep parallelization of algorithms (Adam Lyon)
- CHEP2013: Different approaches to solve the problem
- CMS:
 - Run multiple events in parallel, within one event run multiple modules in parallel, and within one module run multiple tasks in parallel
 - Use Intel Threaded Building Blocks (TBB) for all the parallelization

Concurrency

Concurrent Transitions



Global

Sees transitions on a 'global' scale

see begin of Run and begin of Lumi when source first reads them
sees end of Run and end of Lumi once all processing has finished for them

Multiple transitions can be running concurrently

Events are not seen 'globally'

Stream

Processes transitions serially

begin run, begin lumi, events, end lumi, end run

Multiple streams can be running concurrently each with own events

One stream only sees a subset of the events in a job

Present CMS framework is equivalent to running with only one stream

Paths and EndPaths are a per Stream construct

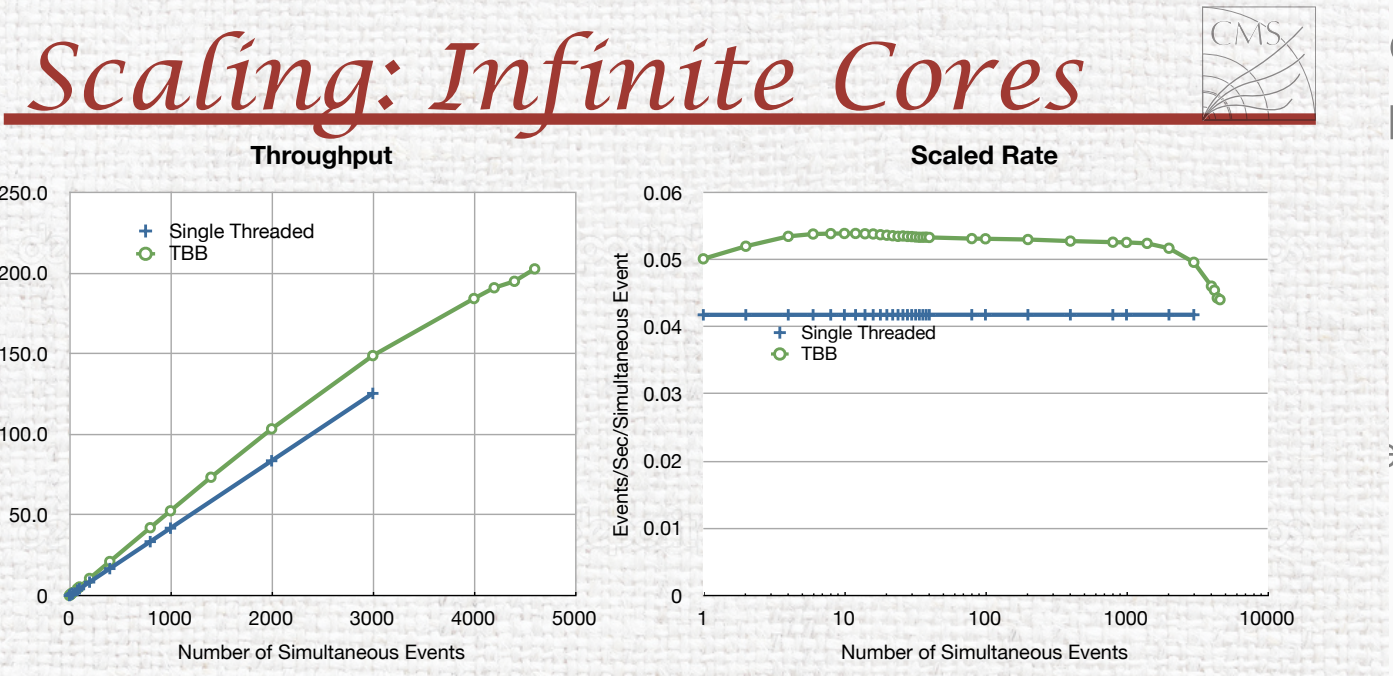
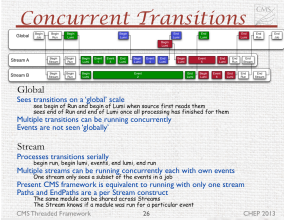
The same module can be shared across Streams

The Stream knows if a module was run for a particular event

Elizabeth
Sexton-Kennedy

17.10.13 • 18

Concurrency



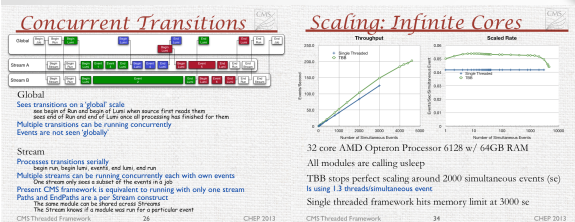
32 core AMD Opteron Processor 6128 w/ 64GB RAM

All modules are calling usleep

TBB stops perfect scaling around 2000 simultaneous events (se)
 Is using 1.3 threads/simultaneous event

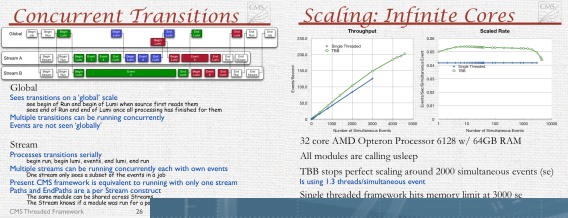
Single threaded framework hits memory limit at 3000 se

Concurrency



- CHEP2013 Prediction: Lots of reports about success of deep parallelization of algorithms (Adam Lyon)
- CHEP2013: Different approaches to solve the problem
- CMS:
 - Run multiple events in parallel, within one event run multiple modules in parallel, and within one module run multiple tasks in parallel
 - Use Intel Threaded Building Blocks (TBB) for all the parallelization
- ATLAS:
 - Use scheduler to start task when input data is ready
 - New scheme is implemented using TBB

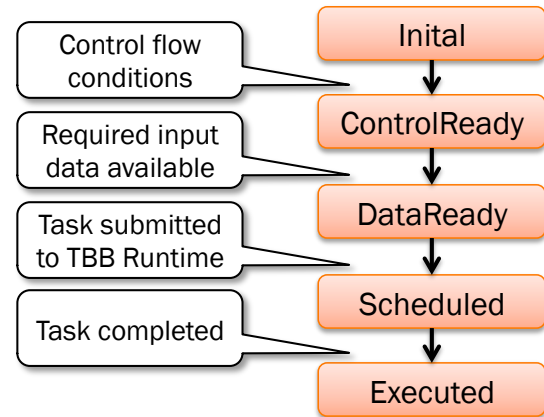
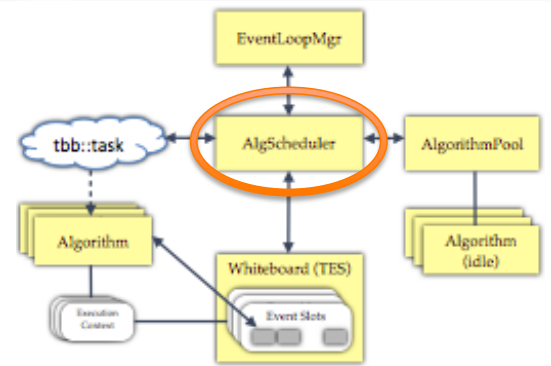
Concurrency



The Forward Scheduler

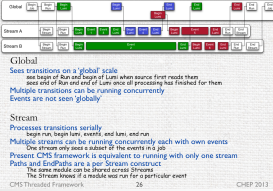
Keeps the state of each algorithm for each event

- Simple finite state machine
- Receive new events from loop manager
- Interrogate Whiteboard for new DataObjects
- Pull algorithms from AlgorithmPool if they are available
- Prepare a `tbb::task` for execution
- Update once `tbb::task` finished

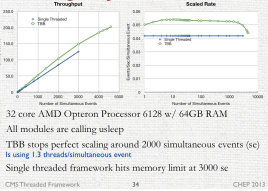


Concurrency

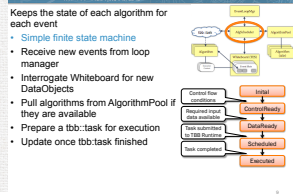
Concurrent Transitions



Scaling: Infinite Cores



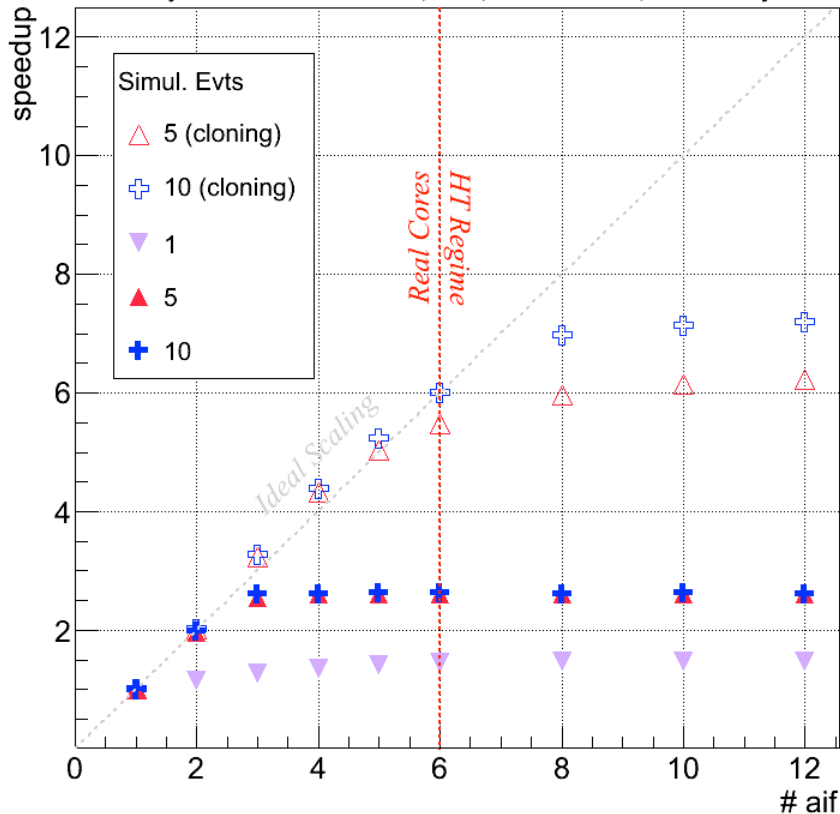
The Forward Scheduler



Scaling on One Processor (Access on)

MiniBrunel 10k evts

Preliminary: 2 sockets * 6 cores * 2 HT, SLC6, no boost malloc, 1 socket only



Multiple events in flight
Clone 3 most time consuming algs (1 copy per event in flight)

Linear scaling of speedup up to number of physical cores

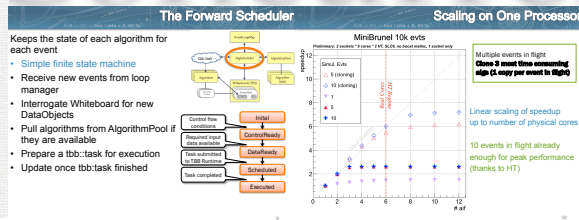
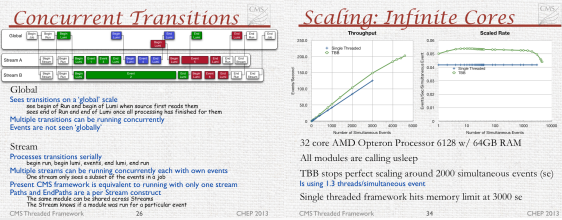
10 events in flight already enough for peak performance (thanks to HT)

ules in

Concurrency

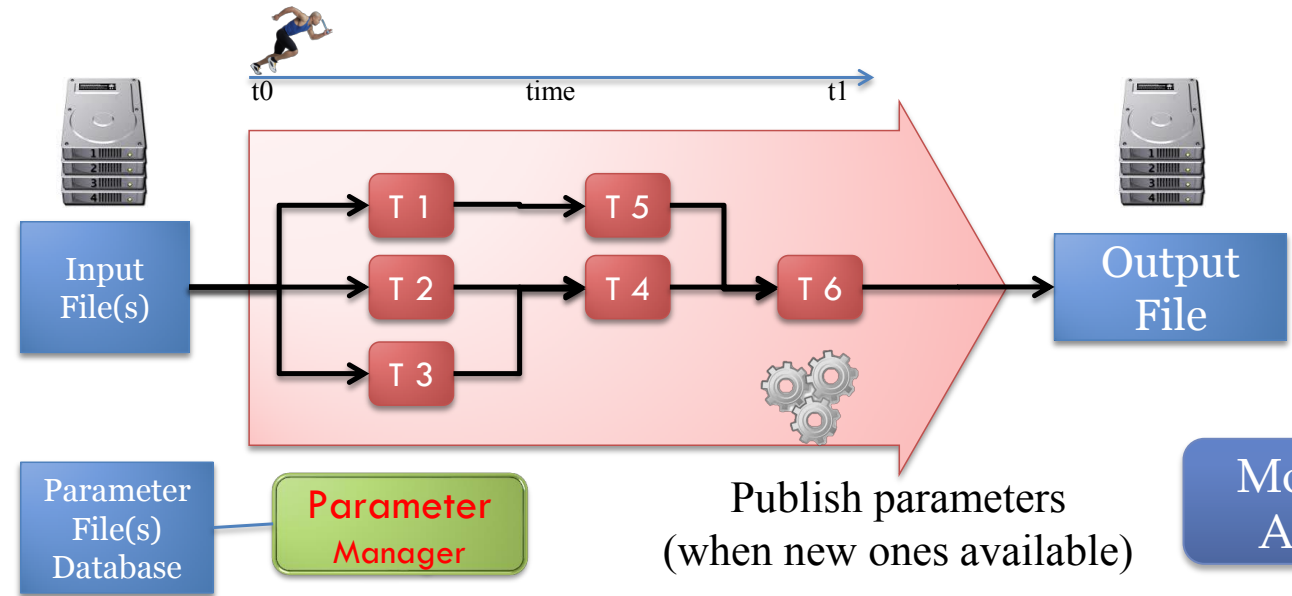
- CHEP2013 Prediction: Lots of reports about success of deep parallelization of algorithms (Adam Lyon)
- CHEP2013: Different approaches to solve the problem
- CMS:
 - Run multiple events in parallel, within one event run multiple modules in parallel, and within one module run multiple tasks in parallel
 - Use Intel Threaded Building Blocks (TBB) for all the parallelization
- ATLAS:
 - Use scheduler to start task when input data is ready
 - New scheme is implemented using TBB
- FairRoot
 - Use Multi-Process instead of Multi-Threading
 - Communication and synchronization through message (data) exchange

Concurrency



FairRoot: Where we are going ? (almost there!)

- Each Task is a process (can be Multi-threaded)
- Message Queues for data exchange
- Support multi-core and multi node



Publish parameters
(when new ones available)

Mohammad
Al-Turany

Concurrency

Concurrent Transitions

Global: Sees transitions on a 'Global' scale... Multiple transitions can be running concurrently. Events are not seen globally.

Stream: Processes transitions serially... Multiple streams can be running concurrently... Present CMS framework is equivalent to running with only one stream.

Scaling: Infinite Cores

32-core AMD Opteron Processor 6128 w/ 64GB RAM

TBB stops perfect scaling around 2000 simultaneous events (sc) is using 1.3 threads/simultaneous event

Single threaded framework hits memory limit at 3000 sc

The Forward Scheduler

Keeps the state of each algorithm for each event

- Simple finite state machine
- Receive new events from loop manager
- Interrogate Whiteboard for new DataObjects
- Pull algorithms from AlgorithmPool if they are available
- Prepare a tbb::task for execution
- Update once tbb::task finished

Scaling on One Processor

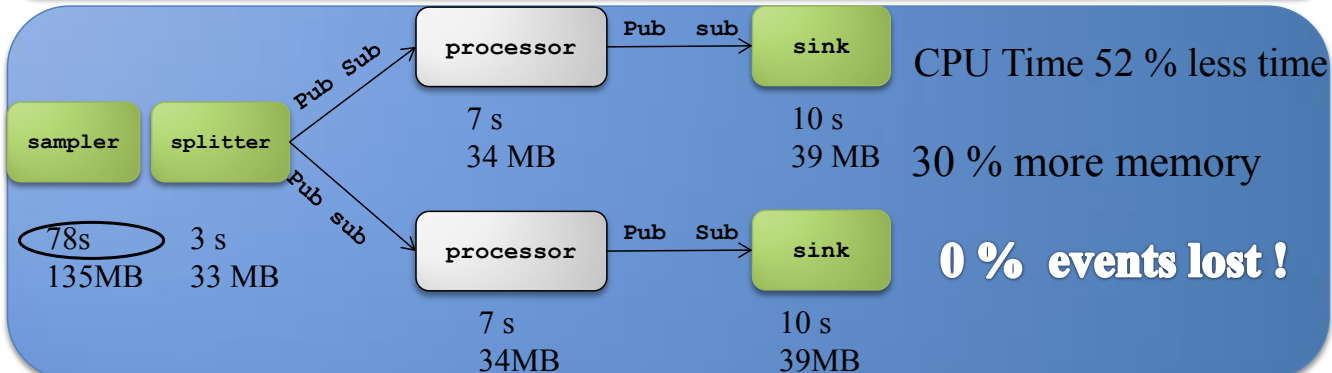
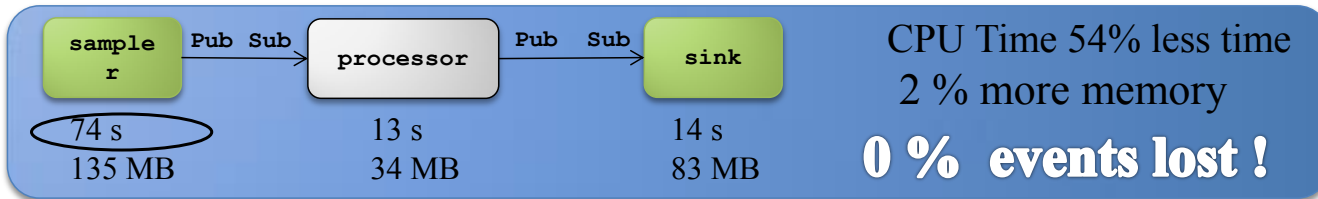
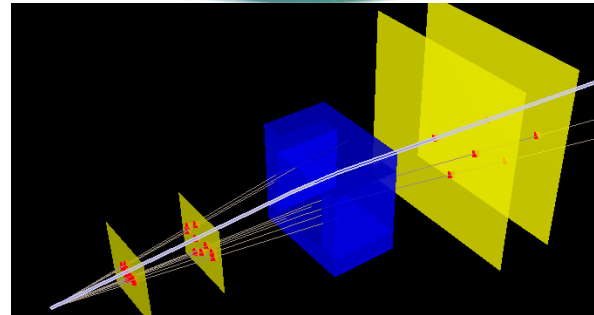
Multiple events in flight... 10 events in flight already enough for peak performance (thanks to HT)

FairRoot: Where are we going? (almost there!)

- Each Task is a process (can be Multi-threaded)
- Message: Queues for data exchange
- Support multi-core and multi node

Test 1: Reconstruction
20k Event 300 Tracks/event

root 162 s 241MB



Concurrency

Concurrent Transitions

Global
Sees transitions on a 'Global' scale
and begin of Run and begin of Loop when source first reads them
and end of Run and end of Loop when all processing has finished for them
Events are not seen globally

Stream
Processes transitions serially
begin run, begin loop, events, and last, end run
Multiple streams can be running concurrently each with own events
One stream only sees a subset of the events in a job
Present CMS framework is equivalent to running with only one stream
Paths and EndPaths are a per Stream construct
The entire module can be divided across Streams
The stream reads it globally and not per processor event

CMS Threaded Framework 26 CHEP 2013

Scaling: Infinite Cores

32-core AMD Opteron Processor 6128 w/ 64GB RAM
All modules are calling usleep
TBB stops perfect scaling around 2000 simultaneous events (sc)
Is using 1.3 threads/simultaneous event
Single threaded framework hits memory limit at 3000 sc

CMS Threaded Framework 34 CHEP 2013

The Forward Scheduler

Keeps the state of each algorithm for each event

- Simple finite state machine
- Receive new events from loop manager
- Interrogate Whiteboard for new DataObjects
- Pull algorithms from AlgorithmPool if they are available
- Prepare a tbb::task for execution
- Update once tbb::task finished

CHEP 2013

Scaling on One Processor

MiniBrunel 10k evts

Multiple events in flight
Cache is used for event exchange
10 events in flight already enough for peak performance (thanks to HT)

Linear scaling of speedup up to number of physical cores

CHEP 2013

FairRoot: Where we are going ? (almost there!)

- Each Task is a process (can be Multi-threaded)
- Message: Queues for data exchange
- Support multi-core and multi node

Input Files → [Tasks] → Output File
FairRoot Manager
Publish parameters (when new ones available)

CHEP 2013

Test 1: Reconstruction

20k Event 300 Tracks/event

Configuration	CPU Time	memory	events lost
1	54% less time	2% more memory	0% events lost
2	52% less time	30% more memory	0% events lost

CHEP 2013

- CHEP2015 Prediction: Lots of reports about success of parallelization of the frameworks
- It will be interesting to compare the different implementations when they are production ready

Algorithms

- Many talks from different collaborations
- Many algorithms are very specific designed for one experiment
 - CBM: Selected event reconstruction algorithms
 - Belle II: Track extrapolation using Geant4E
 -
- There are also developments which should be usable for a larger user community
 - CLAS: Bayesian Data Analysis in Baryon Spectroscopy
 - PANDA: Common Partial Wave Analysis Framework
 -
- How to find such developments which could be (re)used?
 - Database with information?
 - Web page?
- How can we come to a situation like with common frameworks?
-

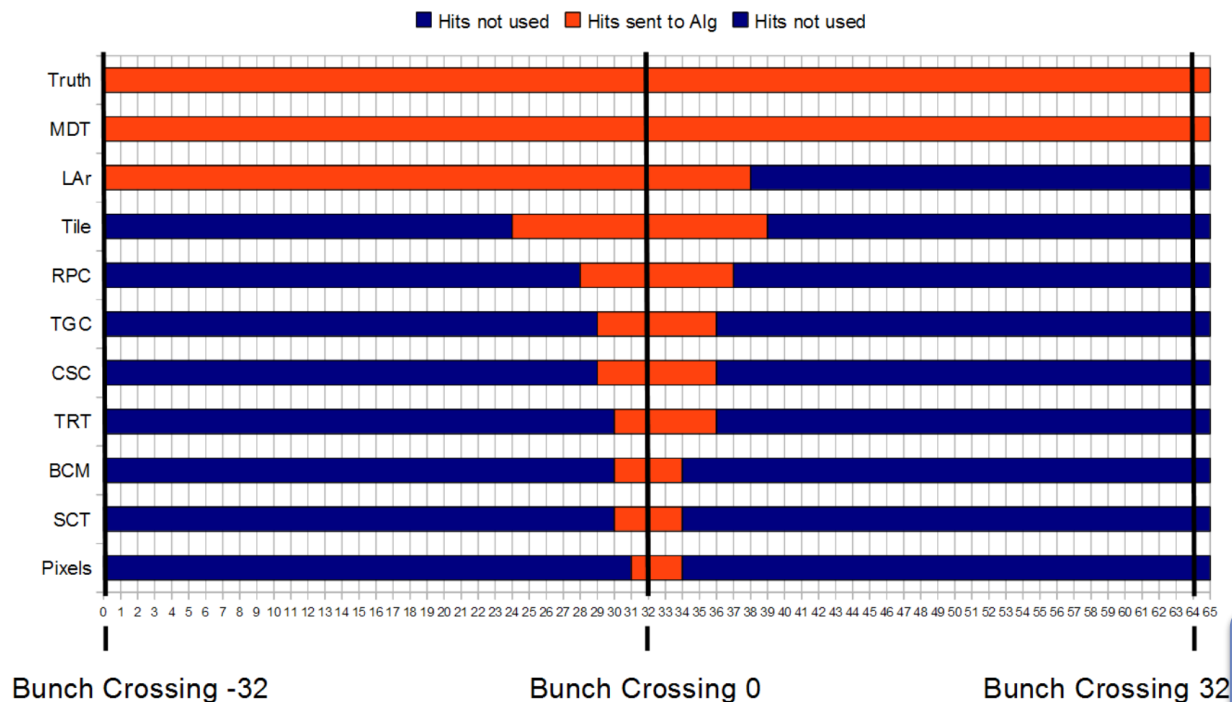
Pileup Simulation

- LHC exceeded expectations of pileup (PU) up to 40 interactions / crossing (design 23)
- Simulation has to keep up
- Geant 4 predictions reached enormous precision, at cost of high CPU consumption → improve its usage
- Overlay: use data for pileup 'simulation'
- other measures, e.g. use only those out-of-time pileup events to which detectors are sensitive

Pileup Simulation

Out of Time Pileup

- Different detectors sensitive to different time windows
- Cutting this down in simulation is critical for performance gains!
 - But including it is critical to get shadowing, saturation, and pulse effects right!



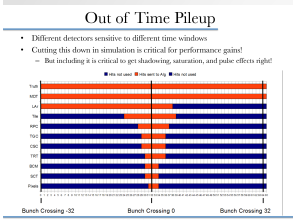
to 40

ion, at
sage

pileup

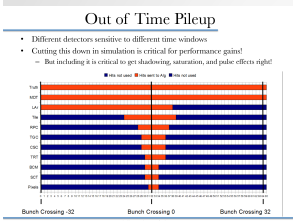
Zachary
Marshall

Pileup Simulation



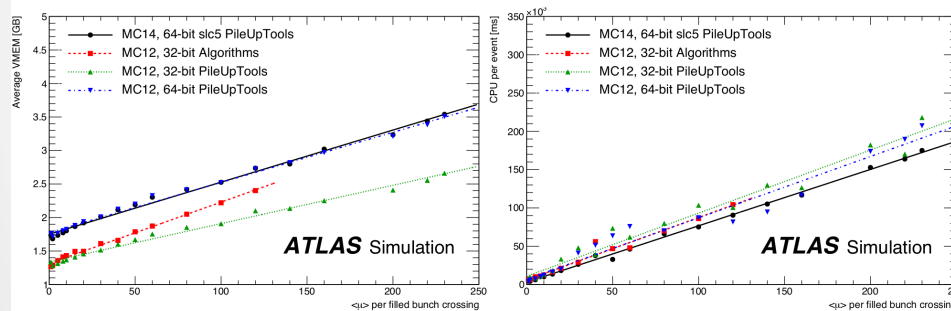
- LHC exceeded expectations of pileup (PU) up to 40 interactions / crossing (design 23)
- Simulation has to keep up
- Geant 4 predictions reached enormous precision, at cost of high CPU consumption → improve its usage
- Overlay: use data for pileup 'simulation'
- other measures, e.g. use only those out-of-time pileup events to which detectors are sensitive
- Simulations limited by CPU and/or memory
- need new ideas to reduce consumptions

Pileup Simulation



Computing Performance

- Obvious trade-off between CPU and memory
 - For high luminosity, we spend the CPU on I/O to avoid serious memory limitations (“Algorithms” → “PileUpTools”)
 - For low luminosity it’s possible to pay with some memory and save some CPU (32-bit → 64-bit, slc5 → slc6)
 - Memory shows much more regular growth; normal non-linear effects on CPU like changing from active memory to swap



16 Oct 2013 Z Marshall - Simulation of Pileup in ATLAS 14

• need new ideas to reduce

ions of pileup (PU) up to 40
design 23)

hed enormous precision, at

CPU/Memory Performance

- Some timing/performance results:

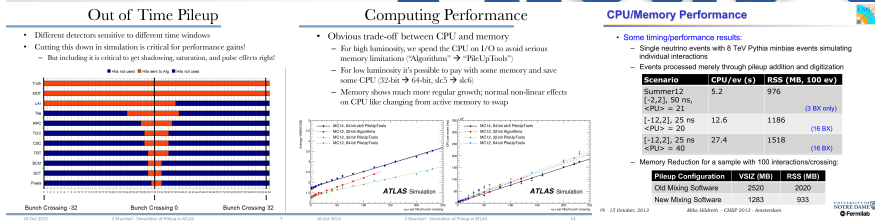
- Single neutrino events with 8 TeV Pythia minbias events simulating individual interactions
- Events processed merely through pileup addition and digitization

Scenario	CPU/ev (s)	RSS (MB, 100 ev)
Summer12 [-2,2], 50 ns, <PU> = 21	5.2	976 (3 BX only)
[-12,2], 25 ns <PU> = 20	12.6	1186 (16 BX)
[-12,2], 25 ns <PU> = 40	27.4	1518 (16 BX)

- Memory Reduction for a sample with 100 interactions/crossing:

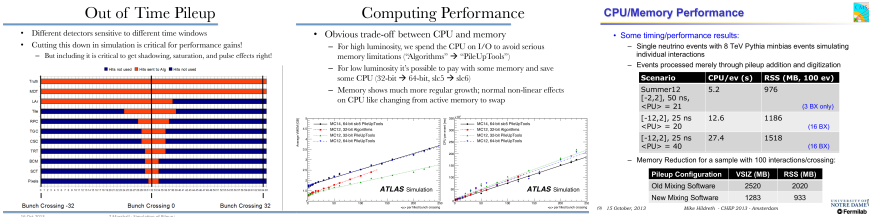
Pileup Configuration	VSIZ (MB)	RSS (MB)
Old Mixing Software	2520	2020
New Mixing Software	1283	1013

Pileup Simulation



- LHC exceeded expectations of pileup (PU) up to 40 interactions / crossing (design 23)
- Simulation has to keep up
- Geant 4 predictions reached enormous precision, at cost of high CPU consumption → improve its usage
- Overlay: use data for pileup 'simulation'
- other measures, e.g. use only those out-of-time pileup events to which detectors are sensitive
- Simulations limited by CPU and/or memory
- need new ideas to reduce consumptions
- Premixing of events

Pileup Simulation



Simulation meets Computation



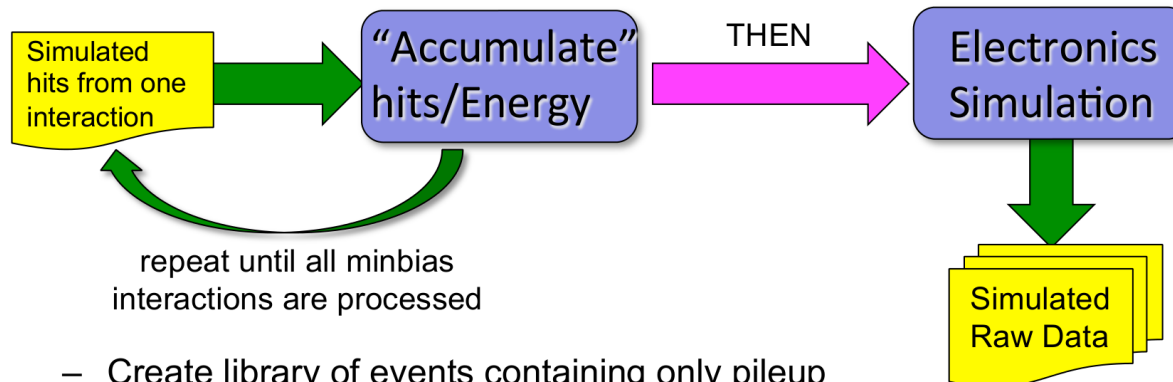
to 40

Even if the events are read sequentially, it still will require more than 2000 minbias events to produce a single MC event with appropriate pileup at sLHC luminosities

- nightmare for computing infrastructure if huge minbias event files have to be made available to each compute node for MC production

Potential Solution: "Pre-Mixing"

- For the pure minbias pileup simulation,



- Create library of events containing only pileup contributions, following pre-determined luminosity profile to calculate how many interactions to include

simulation, at usage

the pileup

Simulation at LHC in Future

- ATLAS/CMS in run-1 produced several billion MC events, even more will be needed in run-2 with higher luminosity & trigger output rates
- ATLAS / CMS investigated in speeding up simulation of their detectors
- frozen showers; fast parameterized simulations
- ATLAS also worries about simulating events under old conditions (trigger simulation)
- One way to reduce CPU time: simulate not all particles: Russian Roulette
 - Now also employed for calorimeters

•

Simulation at LHC in Future

Russian Roulette CPU Usage



- Comparison of CPU performance between 8 TeV and 14 TeV Simulation:

Events	Energy (TeV)	No RR	RR=10	Energy (TeV)	No RR	RR=10
MinBias	8	19.3s	15.2s 78.5%	14	21.5s	16.1s 74.2%
Z→e+e-	8	50.9s	33.4s 65.6%	14	116.9s	92.3s 78.7%
ttbar	8	87.1s	52.8s 60.6%	14	115.8s	74.3s 62.4%

← % of default

needed several billion MC needed in run-2 with output rates

is speeding up simulation

Only n and γ are biased in ECAL and HCAL; RR Factor 10 is used

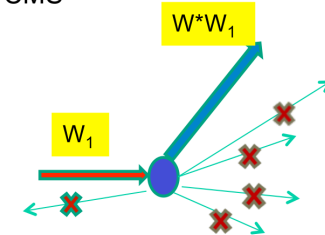
At 14 TeV, Zee becomes compatible in CPU with TTbar. Similar RR effects on

CMS software version CMSSW_6_2_0

Russian Roulette (RR) in CMSSW



- Method used in neutron shielding calculations for many years
 - Not necessary to track all low-energy particles in a shower
- Some fraction of low-energy particles are killed but remainder get higher weight
 - not suited for tracker, muon systems
 - direct CPU savings (for calorimeter simulation)
 - geometry independent
- RR may be enabled separately per particle type and detector region
 - n , γ - allow significant CPU savings for CMS
 - p , e^- - no visible effect so far
- Two parameters per particle
 - RR factor ($1/W$)
 - Upper energy limit



old conditions (trigge

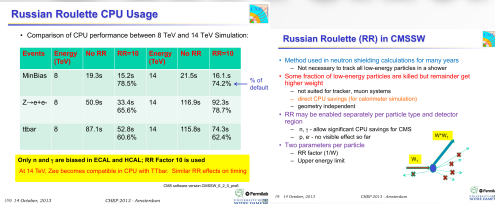
- One way to reduce C particles: Russian Rou
 - Now also employed for calc

Mike Hildreth

17.10.13



Simulation at LHC in Future



- ATLAS/CMS in run-1 produced several billion MC events, even more will be needed in run-2 with higher luminosity & trigger output rates
- ATLAS / CMS investigated in speeding up simulation of their detectors
- frozen showers; fast parametrized simulations
- ATLAS also worries about simulating events under old conditions (trigger simulation)
- One way to reduce CPU time: simulate not all particles: Russian Roulette
 - Now also employed for calorimeters
- Idea of a Integrated Simulation Framework
 - CPU can be reduced by up to factor of 3000
 - Then digitization and reconstruction becomes bottleneck → need also fast digi + reco !

Simulation at LHC in Future

Russian Roulette CPU Usage

• Comparison of CPU performance between 8 TeV and 14 TeV Simulation:

System	Energy (TeV)	No RR (%)	RR=10 (%)	Energy (TeV)	No RR (%)	RR=10 (%)
Mitlab	8	19.3s	15.2s	14	21.5s	16.1s
		78.5%			74.2%	
Z-netc	8	50.9s	33.4s	14	118.9s	92.3s
		65.6%			78.7%	
lbor	8	87.1s	52.8s	14	115.8s	74.3s
		60.6%			64.2%	

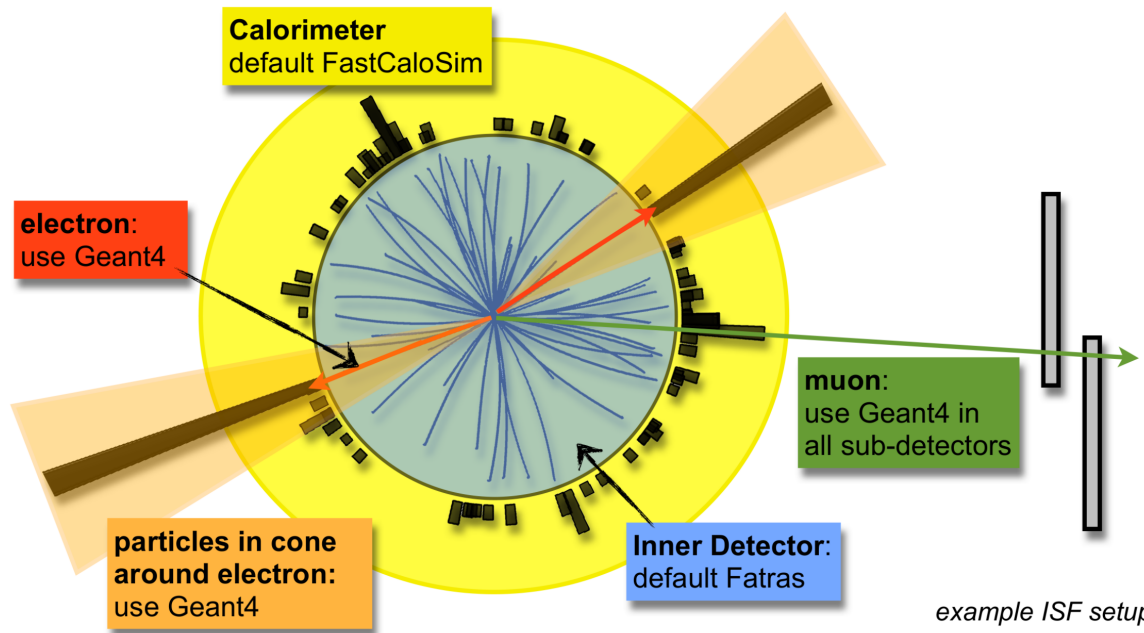
Only μ and γ are biased in ECAL and HCAL. RR Factor 10 is used for μ and γ . RR factor 100 is used for τ . RR factor 1000 is used for b .

Russian Roulette (RR) in CMSSW

- Method used in neutron shielding calculations for many years
- Not necessary to track all low-energy particles in a shower
- Some fraction of low-energy particles are killed but remember get higher weight
- Not suited for hadron, muon systems
- Good CPU savings for calorimeter simulations
- Geometry independent
- RR may be enabled separately per particle type and detector region
- allow different CPU savings for CMS
- allow to scale effect factor
- Two parameters per particle
- RR factor (RR)
- Upper energy limit



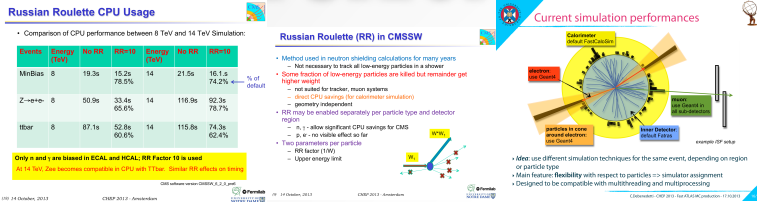
Current simulation performances



- **Idea:** use different simulation techniques for the same event, depending on region or particle type
- Main feature: **flexibility** with respect to particles => simulator assignment
- Designed to be compatible with multithreading and multiprocessing

Chiara
Debenedetti

Simulation at LHC in Future



Produced several billion MC events, finished in run-2 with higher luminosity

ISF performance: H \rightarrow gamma gamma

ISF simulation setup	Speedup	Accuracy
Full Geant4	1	best possible
Geant4 with FastCaloSim	~25	approximated calorimeter
Fatras with FastCaloSim	~750	all subdetectors approximated
Fatras with FastCaloSim simulate only particles in cones around photons	~3000	all subdetectors approximated event simulated only partially

gg \rightarrow H \rightarrow $\gamma\gamma$ no pileup

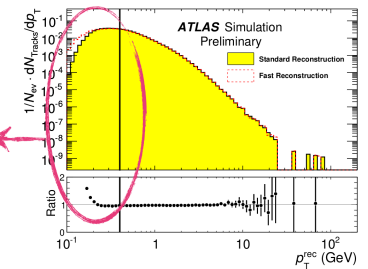
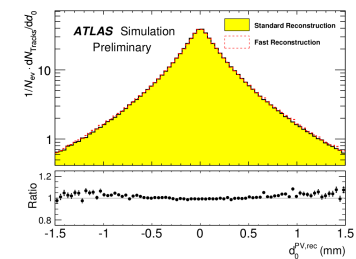
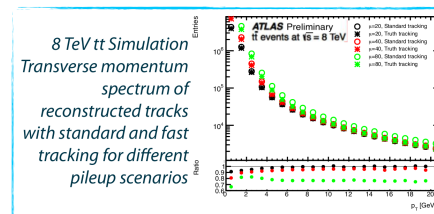
- Use of fast simulation \Rightarrow significant speedup
- Speed increased even further thanks to partial event simulation
- helps in reducing output size

C.Debenedetti - CHEP 2013 - Fast ATLAS MC production - 17.10.2013 20/28

- Now also employed for calorimeter
- Idea of a Integrated Simulation
 - CPU can be reduced by up to factor 10
 - Then digitization and reconstruction + reco !

in speeding up simulation of

Fast reconstruction: performance



- Good agreement with standard Reconstruction
- Significant speedup
- Difference at low momentum not significant
 - $p_T > 400$ MeV for standard ATLAS data and MC processing
- Fast reconstruction with better performance
 - inefficiency factor taken into account for low p_T particles

8 TeV minimum bias Simulation

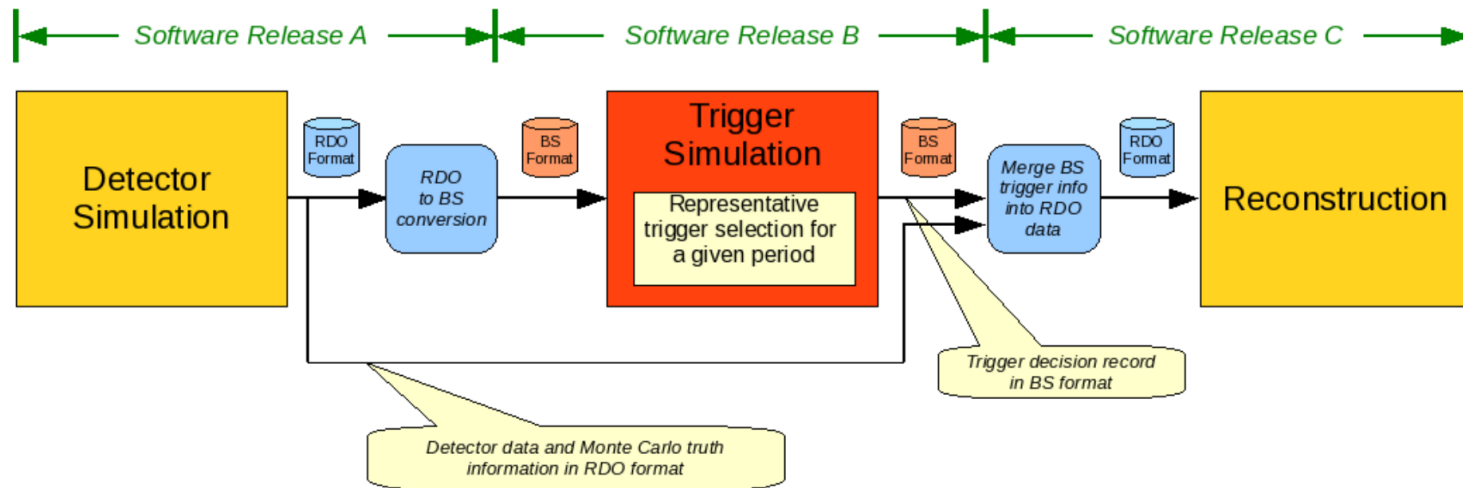
Everything else

- What about simulating old data?

Everything else

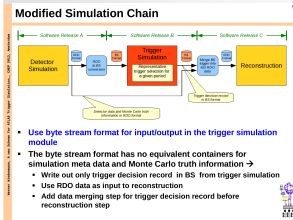
9

Modified Simulation Chain



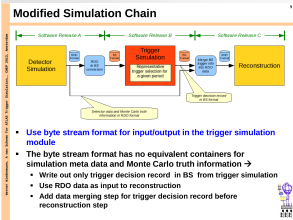
- Use byte stream format for input/output in the trigger simulation module
- The byte stream format has no equivalent containers for simulation meta data and Monte Carlo truth information →
 - Write out only trigger decision record in BS from trigger simulation
 - Use RDO data as input to reconstruction
 - Add data merging step for trigger decision record before reconstruction step

Everything else



- What about simulating old data?
- Speeding up the reconstruction.

Everything else



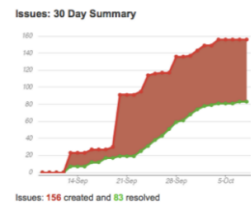
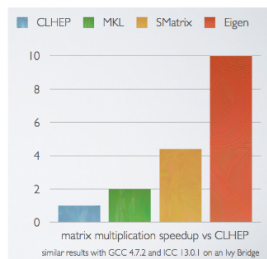
ATLAS, Technische Universität München



old data? reconstruction.

Efforts: Library Change

- Change from CLHEP to Eigen
 - Huge software migration
 - O(1000) packages affected
 - Eigen library functions can vectorize if compiled accordingly
- Exchanging the allocator
- Exchanging GNU libm
 - Under investigation: VDT, libimf



Robert Langenberg – CHEP 2013

10

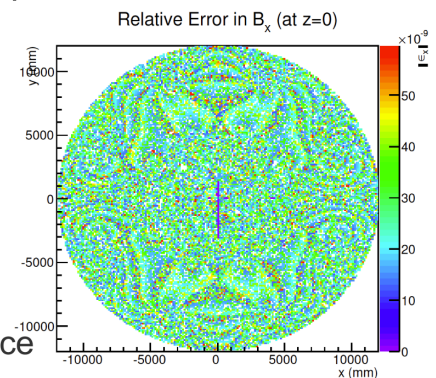
Robert Langenberg

ATLAS, Technische Universität München



Efforts: Magnetic Field

- Change from Fortran77 to C++
 - Code a lot more readable now
 - Reduced function call depth
- Adding field value cache
 - Greatly affects performance as particles are traced along their trajectory
- Unit Conversion Minimization
 - Affects accuracy and performance
- Make code autovectorizable and applying intrinsics
- Speed-up of 20% (reco) up to 60 % (single particle simulation)



17.10.13 42

Robert Langenberg – CHEP 2013

9

Everything else

Modified Simulation Chain

- Software Release A
- Software Release B
- Software Release C

Efforts: Library Change

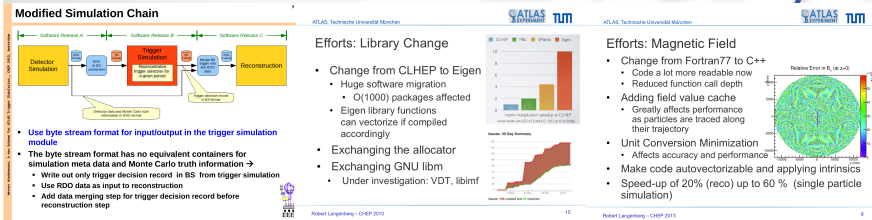
- Change from CLHEP to Eigen
 - Huge software migration
 - O(1000) packages affected
 - Eigen library functions can vectorize if compiled accordingly
- Exchanging the allocator
- Exchanging GNU libm
 - Under investigation: VDT, libimf

Efforts: Magnetic Field

- Change from Fortran77 to C++
 - Code is a lot more readable now
 - Reduced function call depth
- Adding field value cache
 - Greatly affects performance as particles are traced along their trajectory
- Unit Conversion Minimization
 - Affects accuracy and performance
- Make code autovectorizable and applying intrinsics
- Speed-up of 20% (reco) up to 60% (single particle simulation)

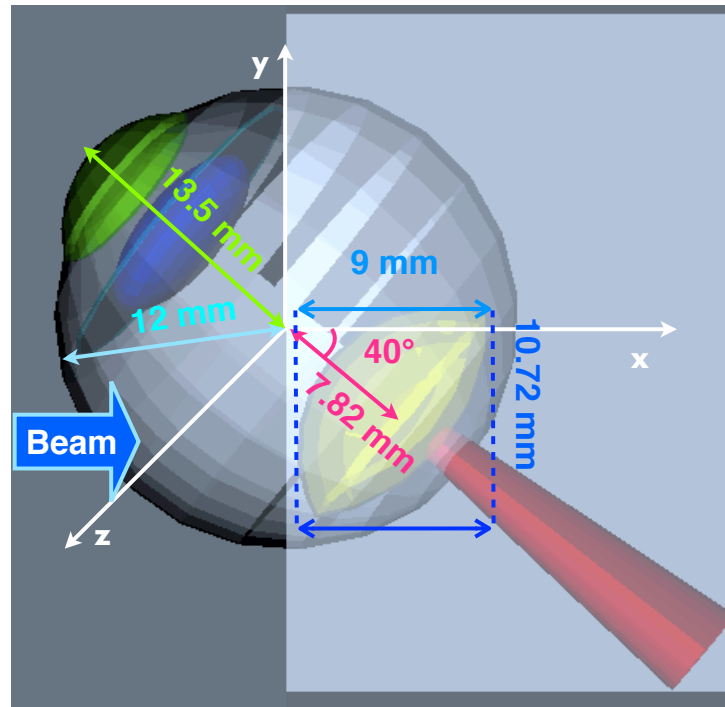
- What about simulating old data?
- Speeding up the reconstruction.
- Geant4 in hadron therapy

Everything else



The eye detector

- Eye anatomy deeply studied and a geometric schematization realized
- Accurate reproduction of all eye-components in the G4 simulation
- Dimensions parameterised as a function of the sclera radius
- Rotation possible to misalign tumour and sensitive sub-components



Everything else

Modified Simulation Chain

- Use byte stream format for input/output in the trigger simulation module
- The byte stream format has no equivalent containers for simulation meta data and Monte Carlo truth information →
 - Write out only trigger decision record in BS from trigger simulation
 - Use ROOT data as input to reconstruction
 - Add data merging step for trigger decision record before reconstruction step

Efforts: Library Change

- Change from CLHEP to Eigen
 - Huge software migration
 - O(1000) packages affected
 - Eigen library functions can vectorize if compiled accordingly
- Exchanging the allocator
- Exchanging GNU libm
 - Under investigation: VDT, libimf

Efforts: Magnetic Field

- Change from Fortran77 to C++
 - Code is a lot more readable now
 - Reduced function call depth
- Adding field value cache
 - Greatly affects performance as particles are traced along their trajectory
- Unit Conversion Minimization
 - Affects accuracy and performance
- Make code auto-vectorizable and applying intrinsics
- Speed-up of 20% (reco) up to 60% (single particle simulation)

The eye detector

- What about simulating old data?
- Speeding up the reconstruction.
- Geant4 in hadron therapy
- LUCID

Everything else

Modified Simulation Chain

Use byte stream format for input/output in the trigger simulation module

- The byte stream format has no equivalent containers for simulation meta data and Monte Carlo truth information →
- Write out only trigger decision record in BS from trigger simulation
- Use PID data as input to reconstruction
- Add data merging step for trigger decision record before reconstruction step

Efforts: Library Change

- Change from CLHEP to Eigen
 - Huge software migration
 - O(1000) packages affected
 - Eigen library functions can vectorize if compiled accordingly
- Exchanging the allocator
- Exchanging GNU libm
- Under investigation: VDT, libfmf

Efforts: Magnetic Field

- Change from Fortran77 to C++
 - Code is a lot more readable now
 - Reduced function call depth
- Adding field value cache
 - Greatly affects performance as particles are traced along their trajectory
- Unit Conversion Minimization
 - Affects accuracy and performance
- Make code auto-vectorizable and applying intrinsics
- Speed-up of 20% (reco) up to 60% (single particle simulation)

A brief history of LUCID

In 2008, the **Simon Langton Grammar School for Boys** entered a satellite experiment design competition run by the British National Space Centre (now **UK Space Agency**) and **Surrey Satellite Technology Limited** (SSTL).

- The Langton Ultimate Cosmic ray Intensity Detector (LUCID) would use Timepix detectors, developed by the Medipix Collaboration, to measure the space radiation environment in Low Earth Orbit.
- Designed by students, built by SSTL, now due to launch in February 2014.
- LUCID now part of [CERN@school](#), supported by UK Science and Technology Facilities Council (STFC) Large Award ST/J000256/1.



data?
tion.

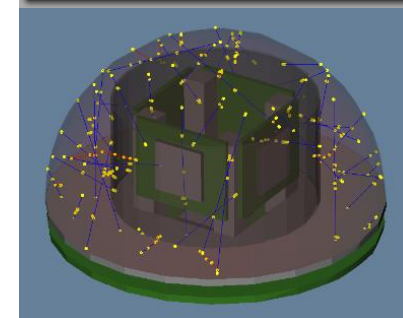
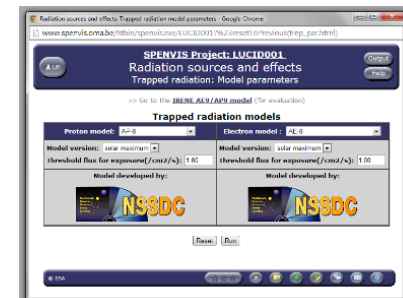
Particle source(s)

SPENVIS

- ESA-backed “Space Environment Information System” web portal.
- Spacecraft coordinate generators:
 - Input LUCID orbit details.
- Trapped radiation models:
 - AP-8 for protons and electrons;
 - Int. and diff. flux spectra.

GEANT4 General Particle Source (GPS)

- Hemi-spherical surface, energy sampled from flux spectra energy bins;
- Right: 50 10–20 MeV protons (“dome” made partially transparent for clarity).



Thank you

Modified Simulation Chain

- Use byte stream format for input/output in the trigger simulation module
- The byte stream format has no equivalent containers for simulation meta data and Monte Carlo truth information
- Write out only trigger decision record in BS from trigger simulation
- Use RDO data as input to reconstruction
- Add data merging step for trigger decision record before reconstruction step

Efforts: Library Change

- Change from CLHEP to Eigen
- Huge software migration
- O(1000) packages affected
- Eigen library functions can vectorize if compiled accordingly
- Exchanging the allocator
- Exchanging GNU libm
- Under investigation: VDT, libfmf

Efforts: Magnetic Field

- Change from Fortran77 to C++
- Code a lot more readable now
- Reduced function call depth
- Adding field value cache
- Greatly affects performance as particles are traced along their trajectory
- Unit Conversion Minimization
- Affects accuracy and performance
- Make code auto-vectorizable and applying intrinsics
- Speed-up of 20% (reco) to 60% (single particle simulation)

A brief history of LUCID

In 2008, the Simon Langham Grammar School for Boys entered a satellite experiment design competition run by the British National Space Centre now UK Space Agency and Surrey Satellite Technology Limited (SSL).

- The Langham Grammar Cosmic-ray Intensity Detector (LUCID) would use Timepix detectors, developed by the Medical Collaboration, to measure the space radiation environment in Low Earth Orbit.
- Designed by students, built by SSL, now due to launch in February 2014.
- LUCID was part of CERN's outreach, supported by UK Science and Technology Facilities Council (STFC) Large Award ST/J00025/1.

Particle source(s)

SEVENTY

- ESA-backed "Space Environment Information System" with partial Spacecraft coordinate generators
- Input LUCID orbit details
- Trapped radiation models:
 - AP 8 for protons and electrons;
 - Int. and diff. flux spectra.
- GEANT4 General Particle Source (GPS)
 - Isotropic spherical surface energy sampled from flux spectra energy bins;
 - Right: 10-20 MeV protons, 100MeV made partially transparent for clarity.

Russian Roulette CPU Usage

Comparison of CPU performance between 8 TeV and 14 TeV Simulation

Method	Energy (TeV)	Time (s)	Time (s)	Time (s)	% of default
MiniBibs	8	19.3s	15.2s	21.5s	16.1s
			75.2%		74.2%
Z+ne	8	50.9s	33.4s	116.9s	92.3s
			65.6%		78.7%
IBDR	8	87.1s	52.8s	119.8s	74.3s
			60.6%		62.6%

Only μ and ν are biased in ECAL and HCAL. RR Factor 10 is used
 6.14 TeV Zee becomes computable in CPU with 17.1sec. Similar RR effects on heavy

Current simulation performances

- Method used in neutron shielding calculations for many years
- Not necessary to track all low-energy particles in a shower
- Some fraction of low-energy particles are killed but remainder get higher weight
- Not suitable for track-based reconstruction
- Event CPU savings for "intermediate simulation"
- Geometry independent
- RR may be enabled separately per particle type and detector region
- N_{μ} = show significant CPU savings for CMS
- N_{ν} = 4-6x reduction effect on ATLAS
- Two implementations per particle:
 - RR Factor (RR)
 - Upper energy limit

ISF performance: H to gamma gamma

ISF simulation setup	Speedup	Accuracy
Full Geant4	1	best possible
Geant4 with FastCalcSim	~25	approximated subdetector
FastCalc with FastCalcSim	~750	all subdetectors approximated
FastCalc with FastCalcSim (simulate only particles in cores around photons)	~3000	all subdetectors approximated even simulated only partially

- Use of fast simulation \Rightarrow significant speedup
- Speed increased even further thanks to partial event simulation
- Helps in reducing output size

Fast reconstruction: performance

- Good agreement with standard Reconstruction
- Significant speedup
- Difference at low momentum not significant
- ~10-100 MeV for standard ATLAS data and MC processing
- Fast reconstruction with better performance
- Efficiency factor taken into account for low p_T particles

Out of Time Pileup

- Different detectors sensitive to different time windows
- Cutting this down in simulation is critical for performance gain!
- But including it is critical for studying saturation, and pulse effects right

Computing Performance

- Obvious trade-off between CPU and memory
- For high luminosities, we spend the CPU on I/O to avoid serious memory limitations ("Algorithm" \rightarrow "PileUp/IO")
- For low luminosities it's possible to pay with some memory and save some CPU (32-bit \rightarrow 64-bit, 32 \rightarrow 64)
- Memory shows much more regular growth: normal non-linear effects on CPU like changing from active memory to swap

CPU/Memory Performance

Some limiting performance results:

- Single neutrino events with 8 TeV Pythia mimics events simulating individual interactions
- Events processed merely through pileup addition and digitization

Scenario	CPU/ev (s)	RSS (MB, 100 ev)
Summer12 [2,2], 50 MeV	5.2	976 (3.8X only)
\rightarrow PU = 21	12.6	1186 (16.6X)
[12,2], 25 ns	27.4	1518 (16.6X)
\rightarrow PU = 40	40	

Memory Reduction for a sample with 100 interactions/crossing:

Pileup Configuration	VSIZE (MB)	RSS (MB)
Old Mixing Software	2520	2020
New Mixing Software	1283	933

Simulation meets Computation

Even if the events are read sequentially, it still will require more than 20000 mimics events to produce a single MC event with appropriate pileup at all MC luminosities

- nightmare for computing infrastructure if huge mimics event files have to be made available to each compute node for MC production
- Potential Solution: "Pre-Mixing"
 - For the pure mimics pileup simulation,
 - repeat until all mimics interactions are processed
 - Create library of events containing only pileup contributions, following pre-determined luminosity profile to calculate how many interactions to include

Concurrent Transitions

Global: Transitions on a global scale
 Multiple transitions can be running concurrently
 Events are not seen globally

Stream: Process transitions serially
 begin run, begin loop, event, and last, end on
 Multiple streams can be running concurrently each with own event
 CPU stream only sees transitions events in per
 Present CMS framework is equivalent to running with own stream
 Paths and Endpoints are a per Stream concept
 The same module can be shared across Processes
 The stream knows if module used for a particular event

Scaling: Infinite Cores

32 core AMD Opteron Processor 6128 w/ 64GB RAM
 All modules are calling sleep
 TBB stops perfect scaling around 2000 simultaneous events (sc)
 Is using 1.3 threads/simultaneous event
 Single threaded framework hits memory limit at 3000 sc

The Forward Scheduler

Keeps the state of each algorithm for each event

- Simple finite state machine
- Receive new events from loop manager
- Interrogate Whiteboard for new DataObjects
- Full algorithms from AlgorithmPool if they are available
- Prepare a tbb:task for execution
- Update once tbb:task finished

Scaling on One Processor

MinidBurl 10k evts

Multiple events in flight
 One 8 event flow consuming 80% CPU per event in flight

Linear scaling of speedup up to number of physical cores

10 events in flight already enough for peak performance (thanks to HT)

FairRoot: Where we are going? (almost there!)

- Each Task is a process (can be Multi-threaded)
- Message Queues for data exchange
- Support multi-core and multi-node

Publish parameters (when new ones available)

Marlin modules for ILD Tracking

ILD Tracking Components:
 TPC - Charge
 VTX/ST - Stripping
 ILD - Forward/Tracking combined - PullDC (LSET)

FairRoot

Start testing the new CPU!
 Pileup decided by CPU
 Pileup: same Base package for different geometries

First Release of FairRoot

MIO (MIO) Main package

APRIS (APRIS) Main package

GENA (GENA) Main package

Collection of simulation modules structural network

What does a framework do?

Dynamic library loading, I/O handling, Event Loop & paths, Provenance generation, Run/Subrun Event stores, Messaging, Configuration, Your physics code, Your physics code, Your physics code, Mesasdra

Code you write, Code you use from the framework

Multi-threading

Porting applications ...

- Few changes needed in user code:
 - Change main() to use `task_manager` - one line
 - Create Sensitive Detector & Field in a new method
 - Adapt to per-event RNG seeding (potential change)
 - Check User Action classes (Srv, Track, Event)
- Choice - handling Output: per thread or accumulate?
 - Geant4 automatically performs reductions (accumulation) if using scores or `GD4` derived classes
- Testing
 - Check output of `mt` - MT vs 1-thread vs Sequential

An excursion - Collaborative analyses with workspaces

- Workspaces allow to share and modify very complex analyses with very little technical knowledge required
- Example: Higgs coupling fits

Full Higgs model

Signal strength in 6 channels

Confidence intervals on Higgs formation, boson couplings

Reparametrize in terms of fermion, boson scale factors