ACAT 2014

Summary of track1:

Computing Technology for Physics Research

Clara Gaspar, September 2014

Some Numbers

- **6 Plenary talks**
- **20 Parallel talks**
- **25 Posters**
- **Talk areas:**
  - Mostly Offline, some Online
  - Mostly Software, some Hardware
- **Thanks to the Session Advisors&Chairs:**
  - Axel Naumann, Niko Neufeld, Jiri Chudoba

Disclaimer

- **Apologies in advance:**
  - For not being able to mention all talks/posters…
  - For any Online bias…
  - For any mistakes, misunderstandings or omissions…

Trends

- **If I had to choose keywords for this summary:**

Optimization!!!

&

Improvement…

- **In the past there were:**
  - New tools, new features, new methodologies…
- **Although there is also some of it…**
  - Now the main aim is optimize… and improve…

Optimization…

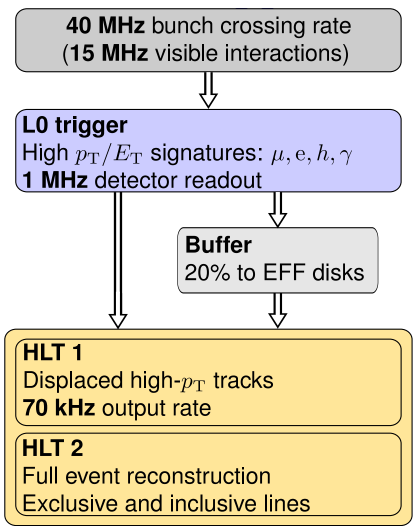
Optimize what?

- **Optimize:**
  - CPU Resources
  - Data I/O
  - Cost
  - Power Consumption
  - Speed
  - Performance in general…
- **Main motivation for Online, Offline, Hardware and Software developments**

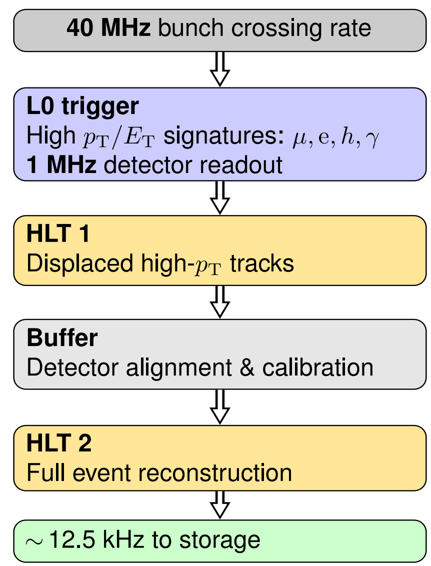
Optimize CPU Resources

- **LHCb High Level Trigger**
  - Optimization by using farm idle time (inter-fill gaps, machine shutdowns, etc.)

RUN 1



RUN 2

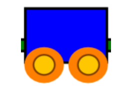


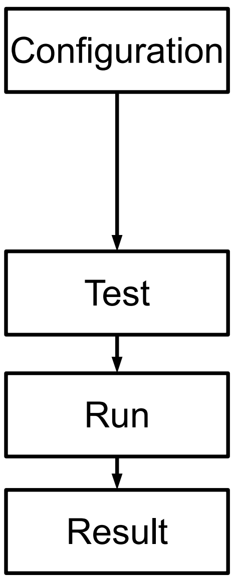
  - In the process: Improve Trigger Quality
  - For Upgrade: design detector for software…

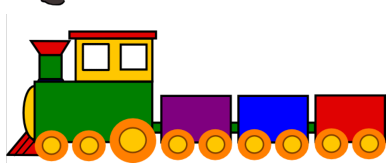
Presented by G. Raven

Optimize I/O

ALICE Analyses Train System

- Optimization by combining multiple analyses  in one grid job.





Train Status	2012	2013	2014 (extrapolated)
Users	60	127	188
Trains	42	69	79
Train runs	1537	4794	7446
Number of jobs	12 million	26 million	36 million
Train wagons/run	14.9	10.1	8.9
Part of the user analysis done with the trains	27%	57%	70%
Processed data	-	75 PB	130 PB
Turn around time	49 hours	22 hours	14 hours

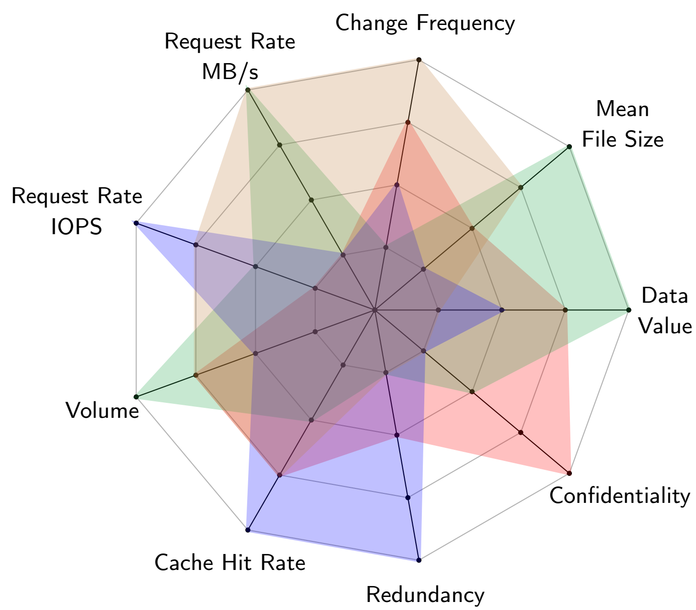
- In the process: Improve usability, management and turn around time

Presented by M. Zimmermann

Optimize I/O

Distributed File Systems

- Global federation of file systems · Hundreds of peta-bytes of data · Hundreds of millions of objects



Data Classes

- Home folders —
- Physics Data —
  - Recorded
  - Simulated
  - Analysis results
- Software binaries —
- Scratch area —

1. Distributed file systems stay
   - physics data processing applications use file system
   - the hierarchical namespace is a natural way to orga
2. Hard disks become data silos
   - We need to focus on optimal bandwidth utilization
   - Once written, we have to leave data where they are → storage and compute nodes coalesce

Optimize I/O

- **CVMFS**
  - SW distribution
    - Many Users
    - Scalable & Optimized

- **Planning for Distributed Workflows**

Consider entire GRID

- Several possible data sources.
- More complex network.
- Limited storage at sites.
- **How to distribute jobs by sites?**
- **Which file source to select?**
- **What is the optimal transfer path?**

What is Constrain Programming?

Constraint programming is a form of declarative programming.
Widely used in: scheduling, logistics, network planning, vehicle routing, production optimization, etc.

Use Case: STAR at BNL

Presented by R. Meusel / D. Makatun

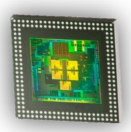
Optimize Cost

- **Massively Affordable Computing Project**
  - Optimization by using ARM SoC units

High Data Throughput

Ethernet Interface

40 Gb/s



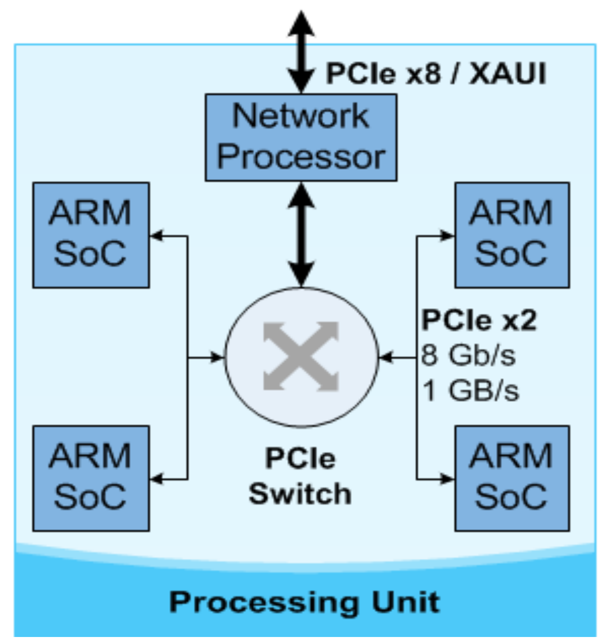
Multiple

System on Chips

> 60 GFLOPS

Appears as a

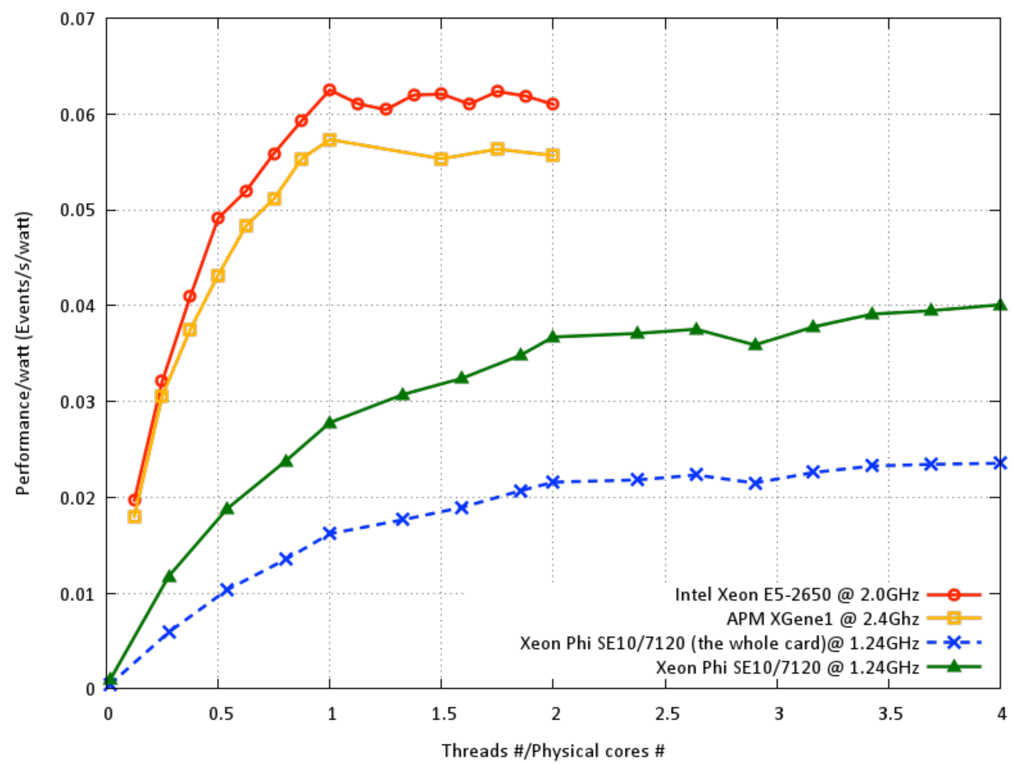
Single System



  - In the process: reduce power consumption

Optimize power usage

- **Optimization by using ARM (APM XGene1)**



- **Port CMS SW to ARM v8 64 bits**
- **ARM is a relevant platform…**

Presented by D. Abdurachmanov

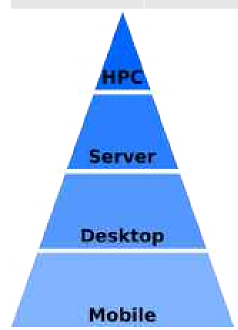
Optimize power usage

Mont-Blanc: Project objectives

- To deploy a prototype HPC system based on **currently available** energy-efficient embedded technology
  - Scalable to 50 PFLOPS on 7MWatt
  - Competitive with Green500 leaders in 2014
  - Deploy a full HPC system software stack

What's commodity nowadays?

	Servers		PC		Smartphones	
2012	8.7M		350M		725M	
2013	9.0M	+3%	315M	-9.8%	1000M	+38%



...and we are still ignoring tablets: >200M

- **Build a new class of sustainable computer: faster, cheaper, more efficient**

Presented by F. Mantovani

Optimize Power Usage

Optimization by using water cooling

10 MW Datacenter Design Match-up

kWatts	Best Practice	Free Air @ 20C	Free Air @ 35C	NREL + Apollo
IT Load	10000	10000	11530	10000
DC Fan Load	400	400	1614	0
Chiller Load	1706	0	0	0
Evap. Towers	0	0	0	284
Water Pumps	114	0	0	40
UPS Losses	500	0	0	0
Power Distribution Losses	900	900	1038	400
Humidification/DeHum	100	200	231	0
Lighting	2	2	2	2
IT Load PUE	1.37	1.15	1.25	1.07
Total Power Consumption	13722	11502	14415	10726

2.6M$ annual energy savings! (@ 10 cents / kWh)

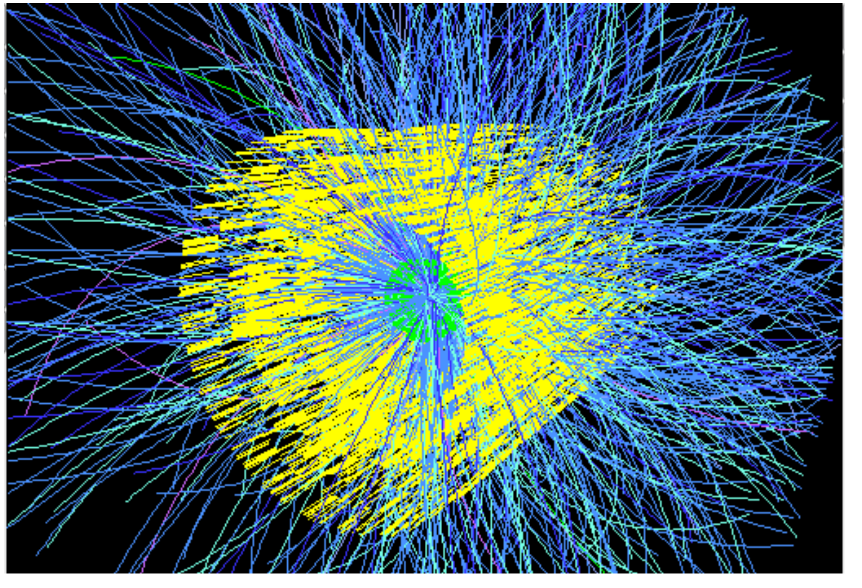
Presented by V. Saviak

Optimize Memory Usage

- **ATLAS FTK Simulation**
  - FTK HW implements a billion fold parallelism
  - Main Problem:
    One billion patterns
    (35 GB)
  - Solution:
    Split into parallel jobs
    Execute sequentially
    on grid nodes
    Combine results

Fast tracking is implemented in hardware with custom electronics

- Between the L1 trigger and HLT
  - **Provides full tracking for all events passing L1**



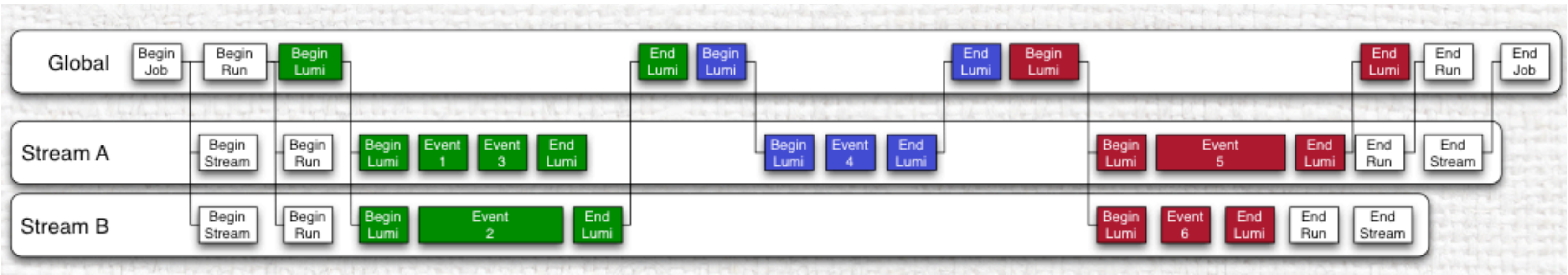
Presented by A. Vaniachine

Optimize Memory Usage

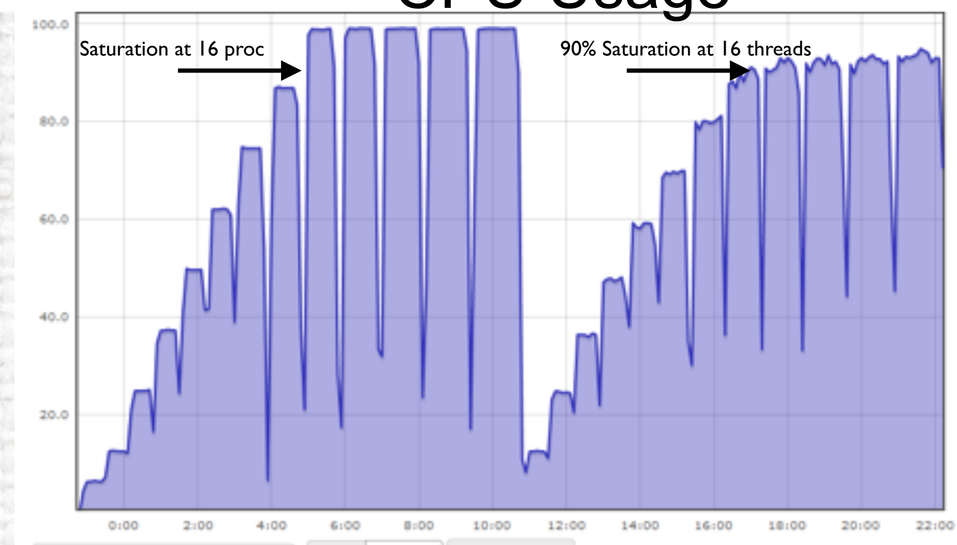
- **By using concurrency**
  - In LHC Offline Frameworks:
    - Gaudi in LHCb, ATLAS, FCC, HARP, Fermi, etc.
    - Threaded Framework CMS
    - Athena (Gaudi derivative) in ATLAS
  - In many-core not enough memory/core
  - Threads share more memory than independent processes (although forking helps)
  - But multi-threading an application brings many synchronization problems:
    - Workflow, data access, etc…

Concurrency in CMS

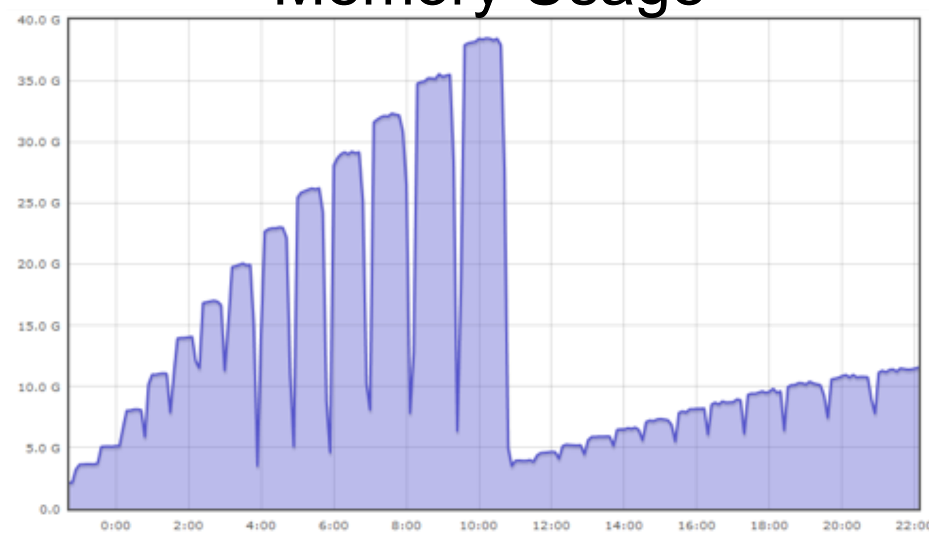
- Use Intel's Threaded Building Blocks



CPU Usage



Memory Usage

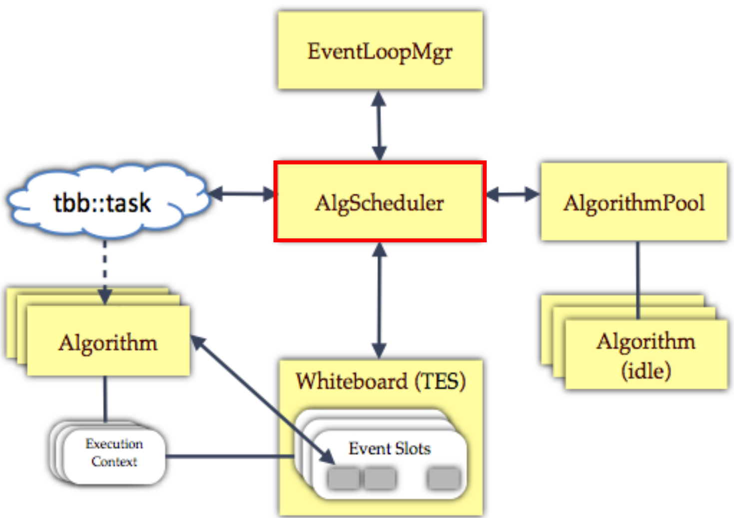


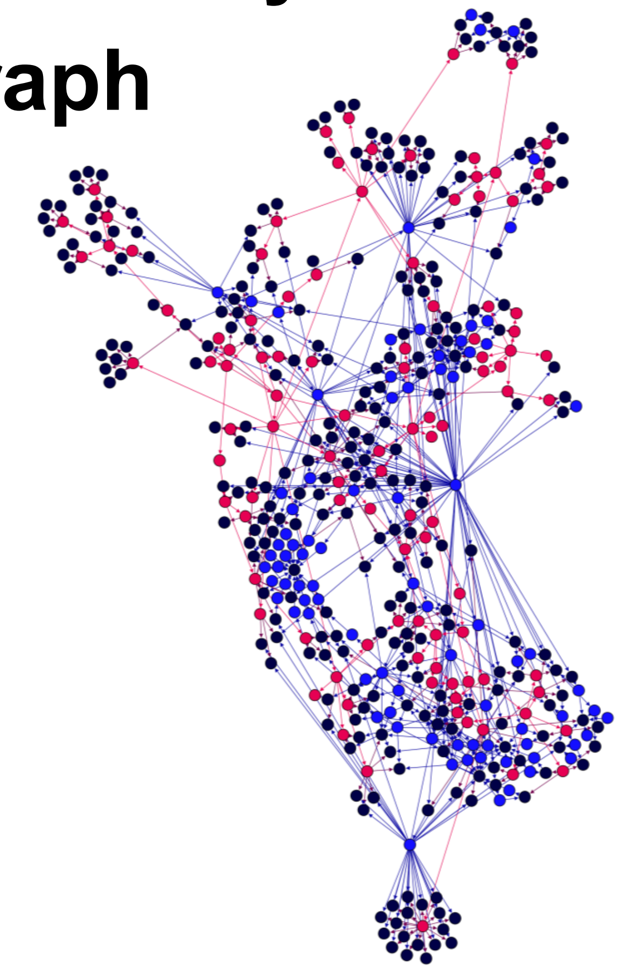
- **99.3% of reconstruction runs in parallel**

Presented by E. Sexton-Kennedy

Concurrency in Gaudi

- New components for Concurrency
- Unified control and data graph



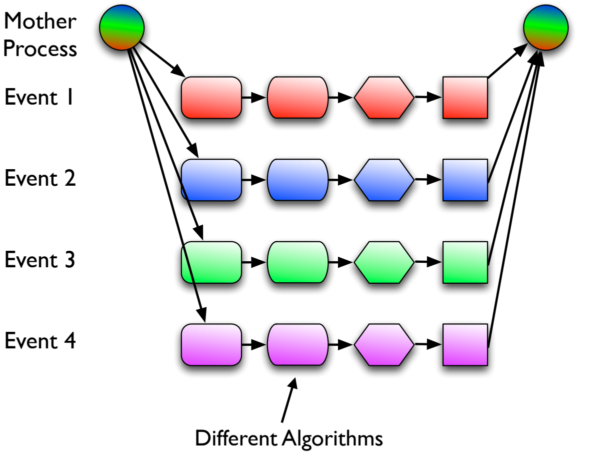


- Backward compatible
- To be adopted gradually…

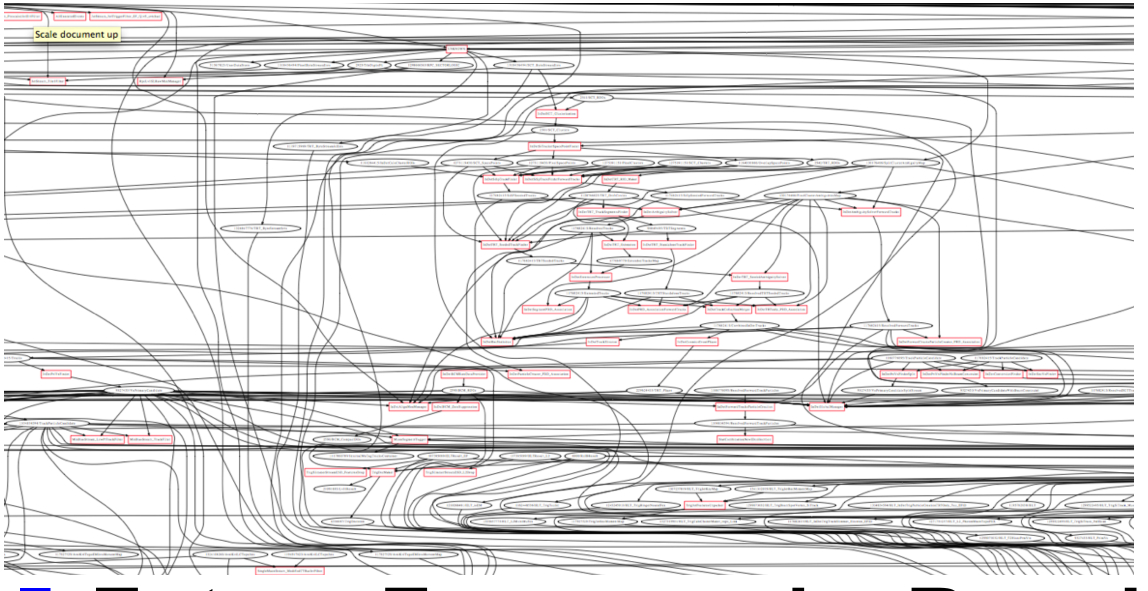
Presented by D. Funke

Concurrency in ATLAS

- Athena MP for RUN2



- ATLAS Reconstruction (snippet)

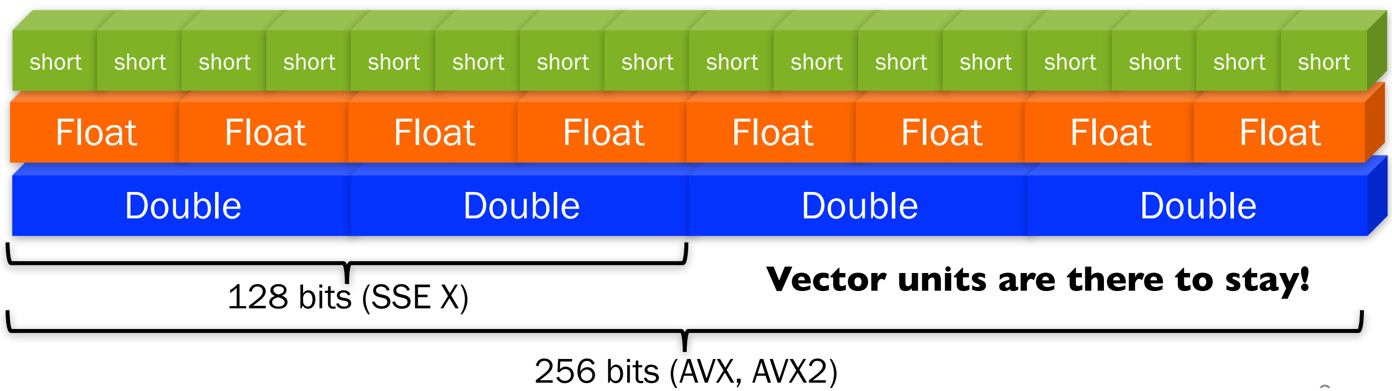


- Future Frameworks Requirements Group for RUN3

Presented by R. Jones

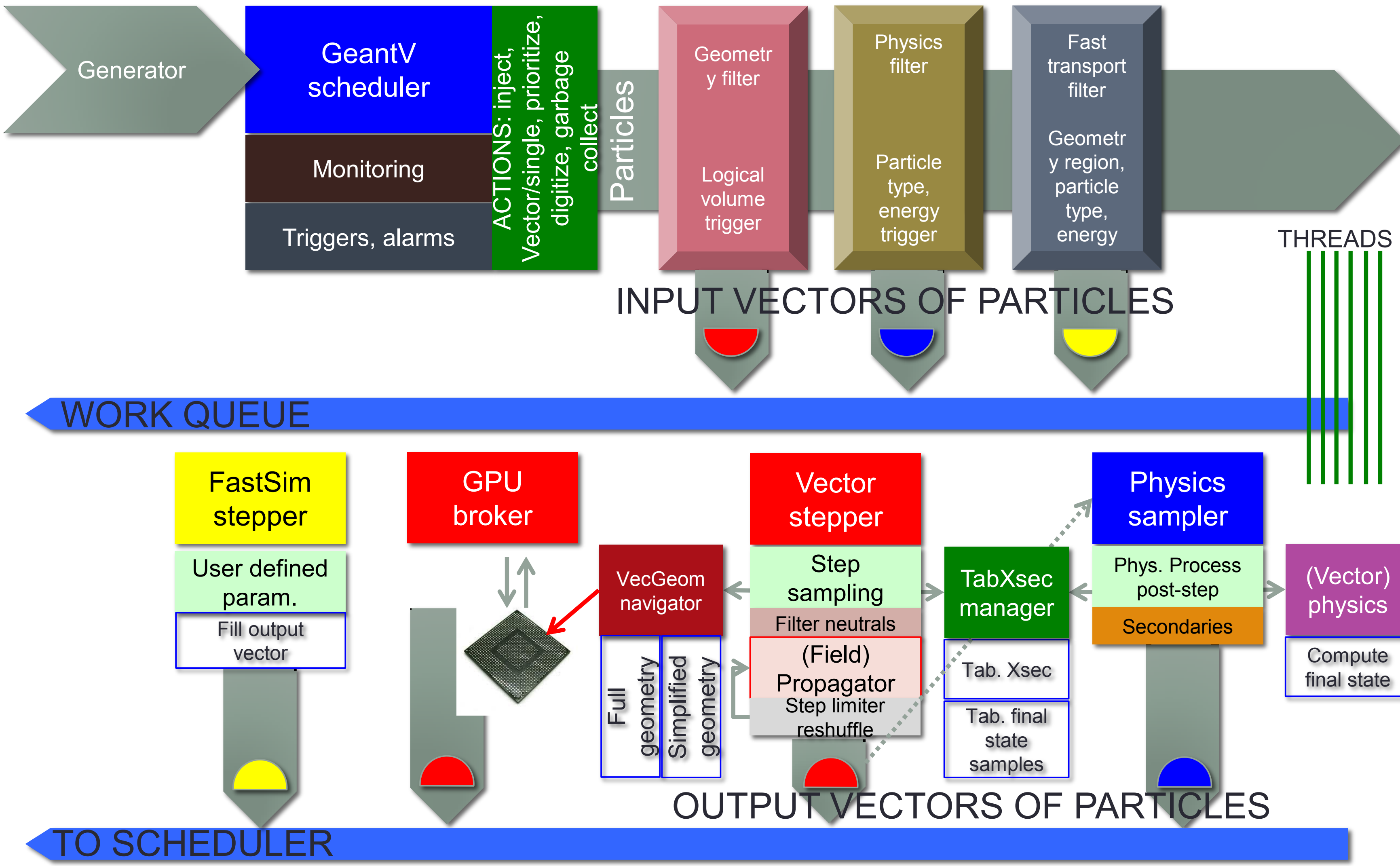
Optimize CPU Cycles

- **By Using Vectorization**
  - Vector instructions getting more important
  - Peak performance only when using them well



- **Efforts in:**
  - Simulation: GeantV and Geometry (VecGeom)
  - And in ROOT

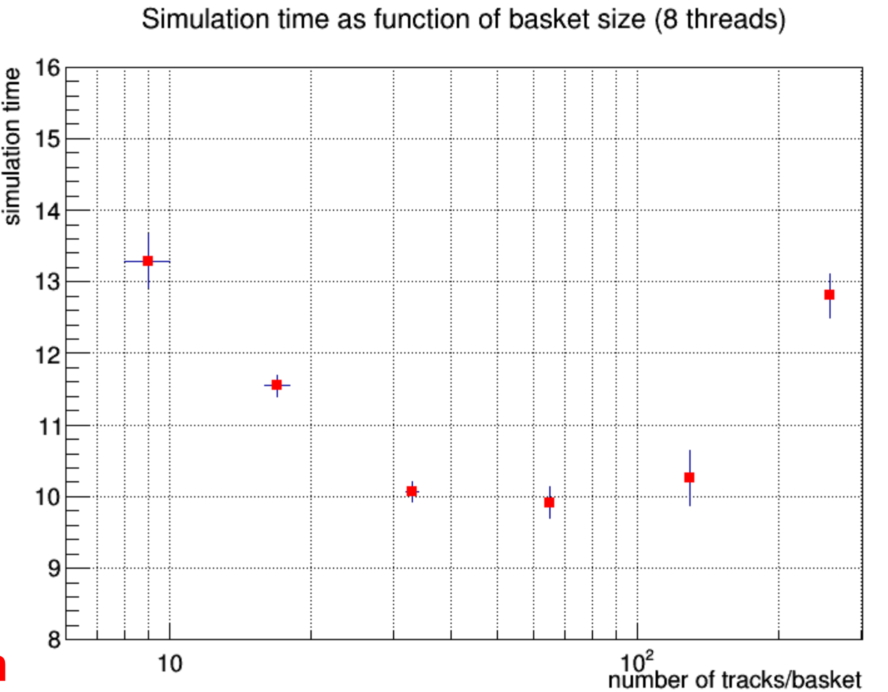
Vectorization in GeantV



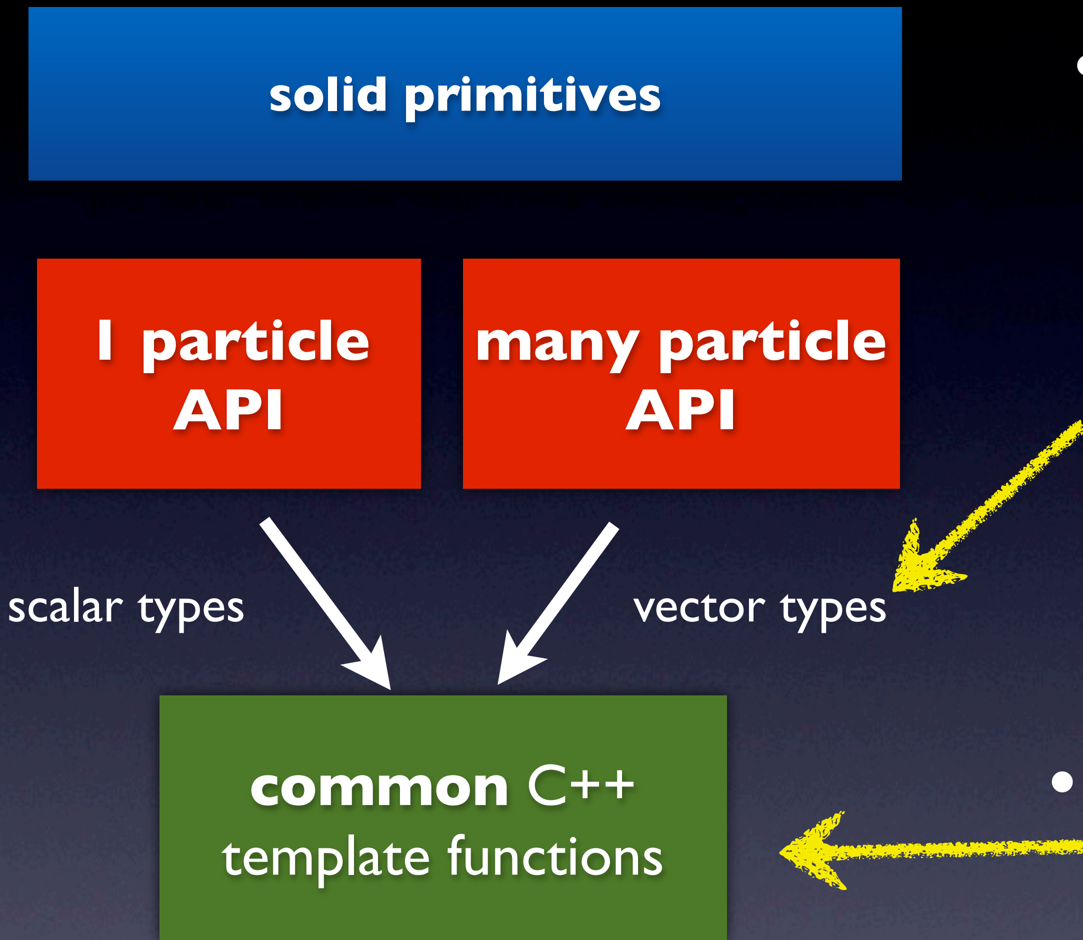
GeantV Virtualization

Parameters: basket size

- **The vector size is a major parameter of the model**
  - Impacts on vectorization potential
- **The optimum value depends on many parameters**
  - Such as geometry complexity, physics
  - To be explored for several setups
- **Small vectors = inefficient vectorization, dispatching becomes an overhead**
- **Large vectors = larger overheads for scatter/gather, more garbage collections (less transportable baskets)**
- **The differences in total simulation time can be as high as 30-40%**
  - Aiming for an automatic adjustment of vector size per volume
  - Performing at least as good as the optimum for fixed vector size



Vectorization in Geometry



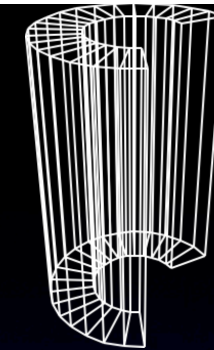
- reliable efficient SIMD vectorization achieved by using vector libraries (e.g. Vc) providing C++ approach to explicit vectorization

  http://code.compeng.uni-frankfurt.de/projects/vc

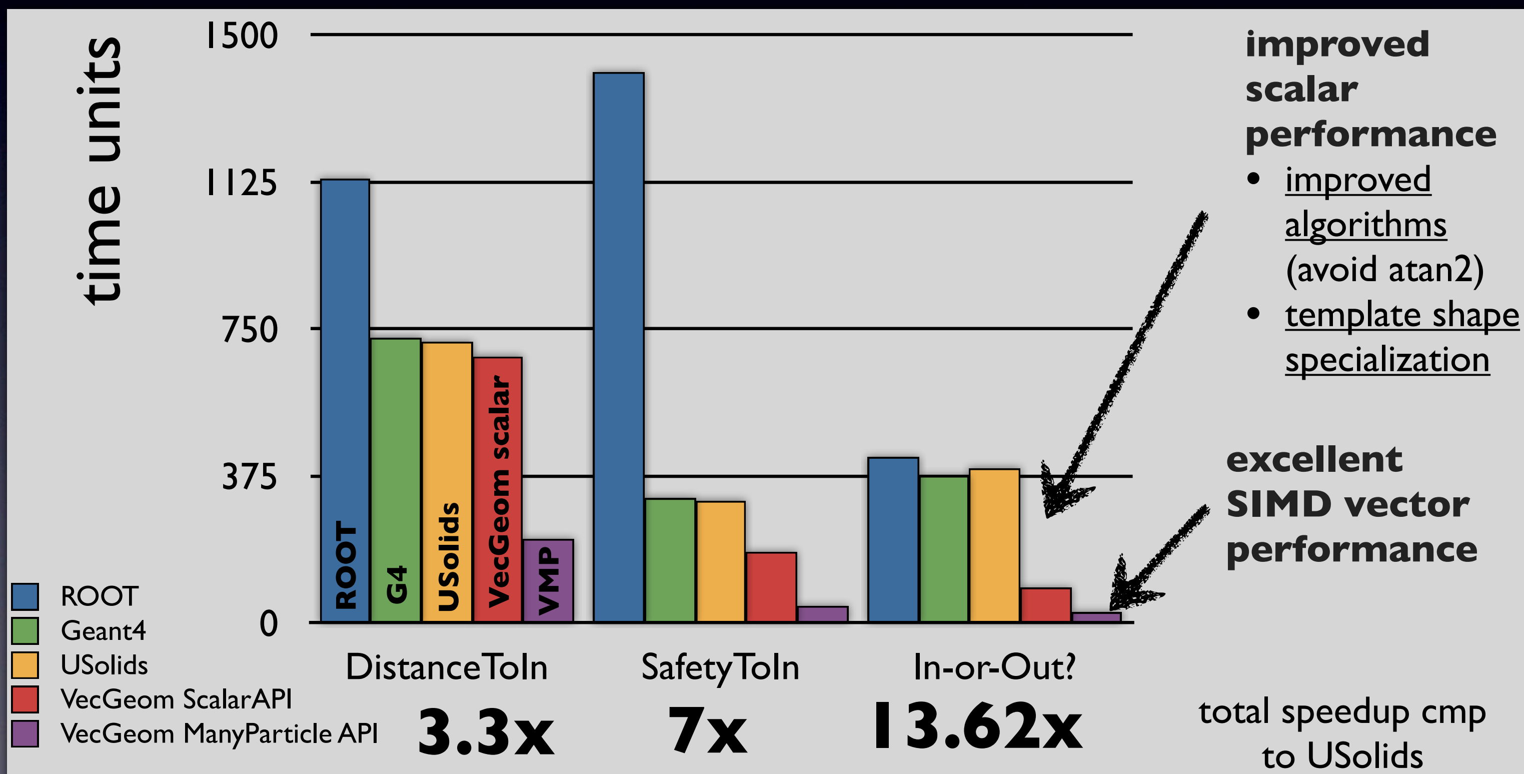
- template C++ programming solves code multiplication issue

Vectorization in Geometry

Performance case study: the tube segment



- **most used/important** shape primitive
- also **integral part** of complex shapes: polycone
- extremely **important to be as fast as we can**



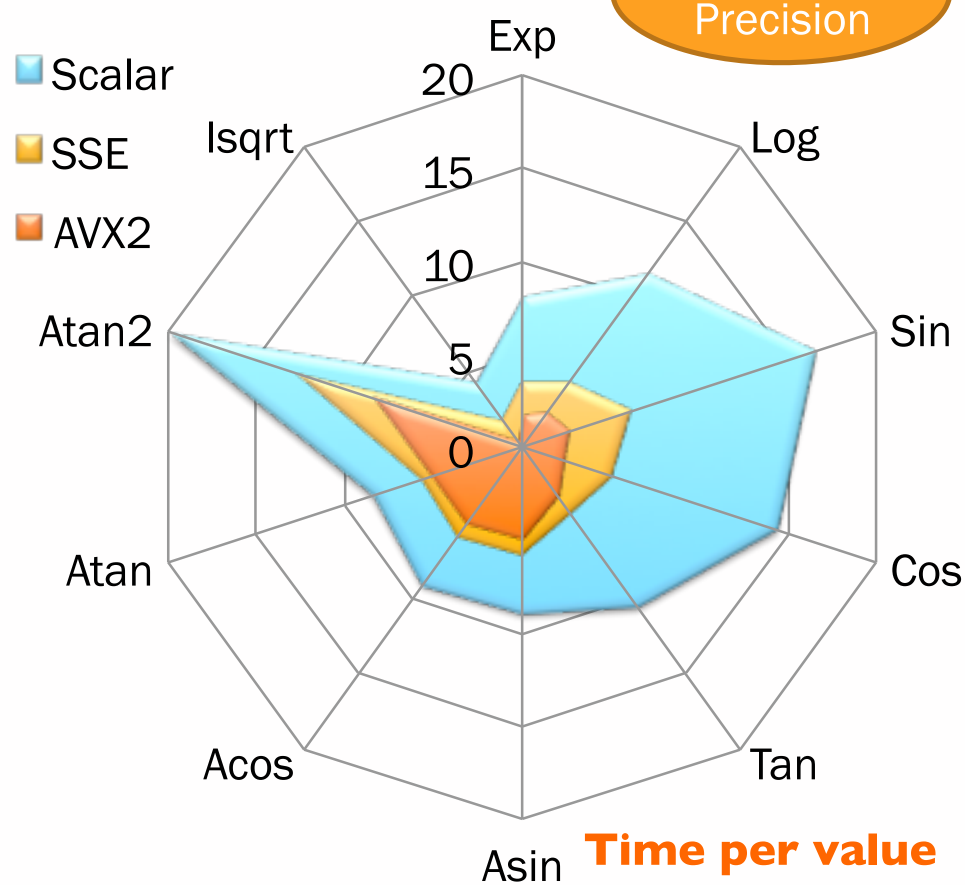
Vectorization in ROOT

Mathematics Library: Vdt

Exp	*8*	*3.5*	*1.7*
Log	*11.5*	*4.3*	*2.2*
Sin	*16.5*	*6.2*	*2.6*
Cos	*14.4*	*5.1*	*2.3*
Tan	*10.6*	*4.4*	*3.2*
Asin	*8.9*	*5.8*	*5*
Acos	*9.1*	*5.9*	*5.1*
Atan	*8.4*	*5.6*	*5.1*
Atan2	*19.9*	*12.7*	*8.4*
Isqrt	*4.3*	*1.8*	*0.4*

Time in **ns** per value calculated

Double Precision



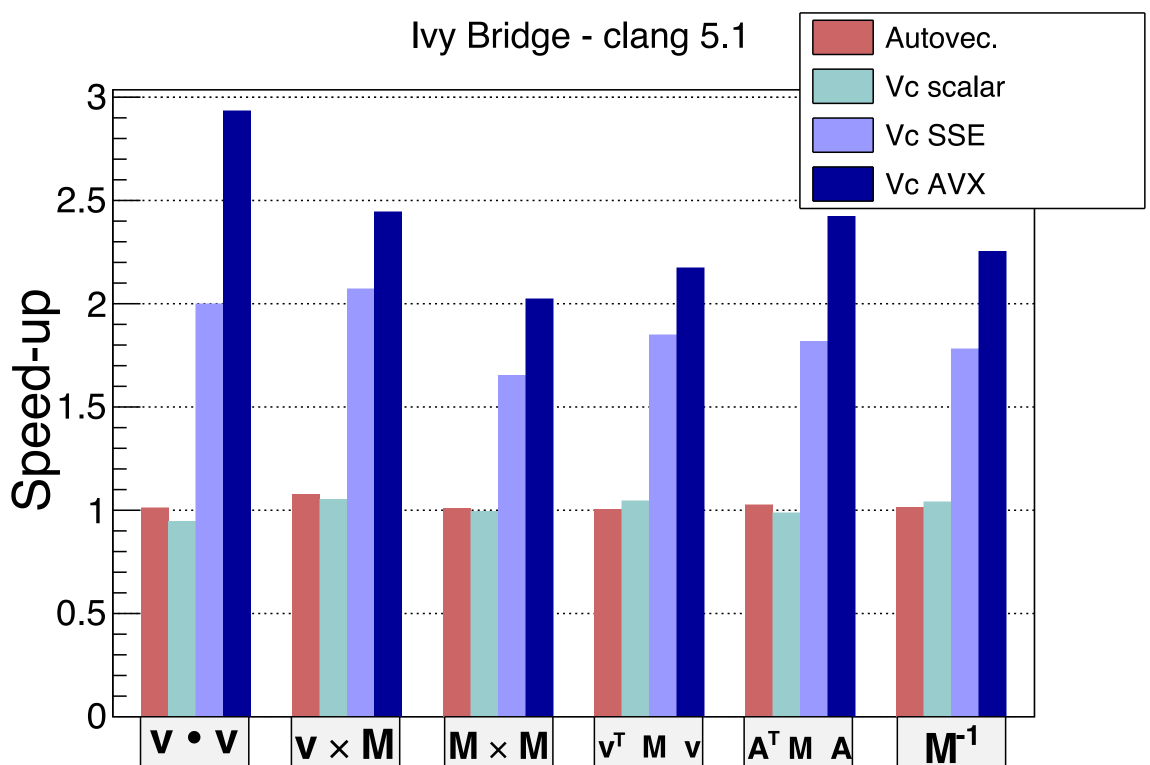
Time per value calculated

- **Effect of vectorisation clearly visible**

Vectorization in ROOT

Explicit vectorization using Vc Library

- Operations in SMatrix using `Vc::double_v` instead of `double`
  - speed-up obtained for processing operations on a list of 128 `SMatrix<double,5,5>` and `SVector<double,5>`



Presented by S. Wenzel

Optimize Speed

Memory Models in HPC

Transactional Memory

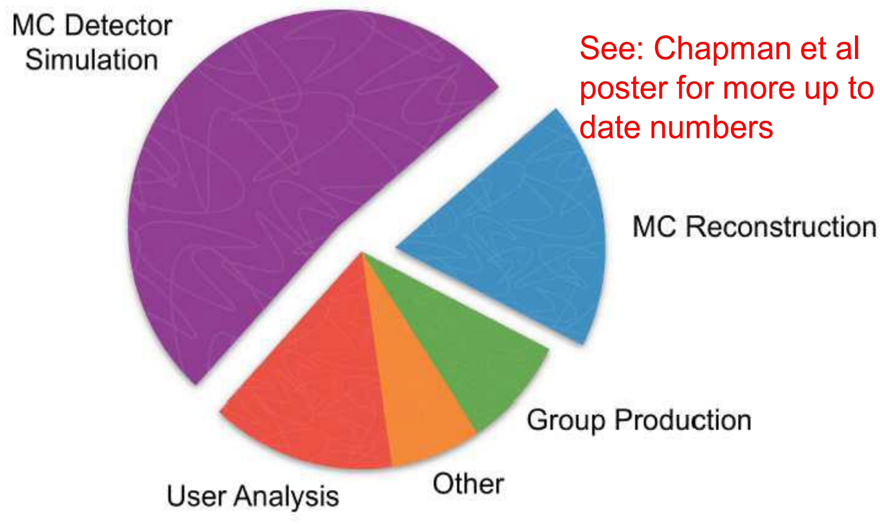
- Replaces waiting for locks with concurrency
- Allows non-conflicting updates to shared data
- Shown to improve scalability of short critical regions
- Promise of Transactional Memory
  - Program with coarse transactions
  - Performance like fine-grained lock
- Focus on correctness, tune for performance
  - Easier to reason about only a few transactions…
  - … only focus on areas with true contention
- Hardware TM implementation:
  - Intel's TSX, as of Haswell-EX (disabled in E and EP models due to a bug discovered in August 2014)
  - IBM's Blue Gene/Q, zEnterprise EC12, POWER8
- Compilers: vendor-specific, gcc-4.7

Optimize Speed

▌Fast Detector Simulation

- By using pre-generated samples or parametrizations

ATLAS GRID CPU utilization



See: Chapman et al poster for more up to date numbers

Compensate the lack of time and resources to produce MC samples by a faster approach

- Increase in throughput of O(10-100)

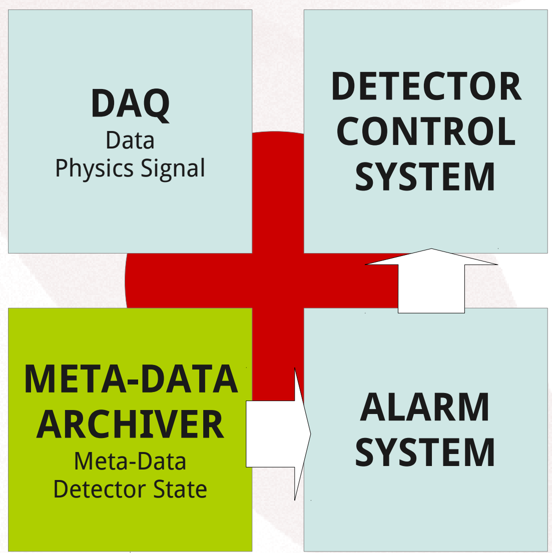
Fast simulation is an option for many analyses

- Price: physics performance, to be considered case by case

Improvements…

Improve Flexibility/Lifetime

Cornerstones of “a” Physics Experiment's Backend

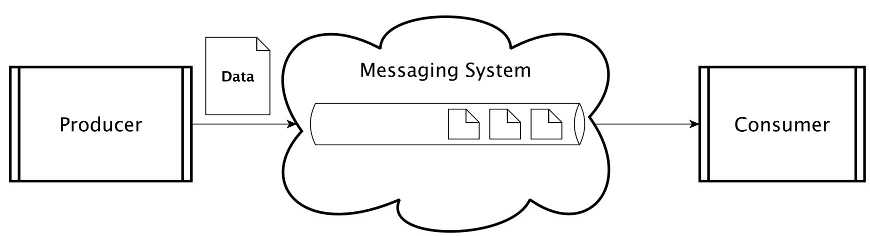


MIRA Framework started from this corner (+migration)

STAR

- **STAR Meta-Data Collection Framework** overview was presented
  - Message-Queuing service became an instrumental part of STAR online infrastructure
  - MQ-based: flexible, loosely coupled system
  - Accepted very well by STAR collaborators and detector experts, covers the monitoring needs of all 18 STAR subsystems now
  - Number of channels has increased to ~1700, or x15, number of data structures has increased to ~3000, or ~x25
- **Run 14 Extension: Complex Event Processing**
  - CEP features added and tested in 2014, now we are confident in its capabilities. Deploying for a full production usage in 2015
  - Proven be be helpful: a few alarms implemented in Run 14, saved months of work for the core team and users. More use-cases to be implemented for Run 15 and beyond.

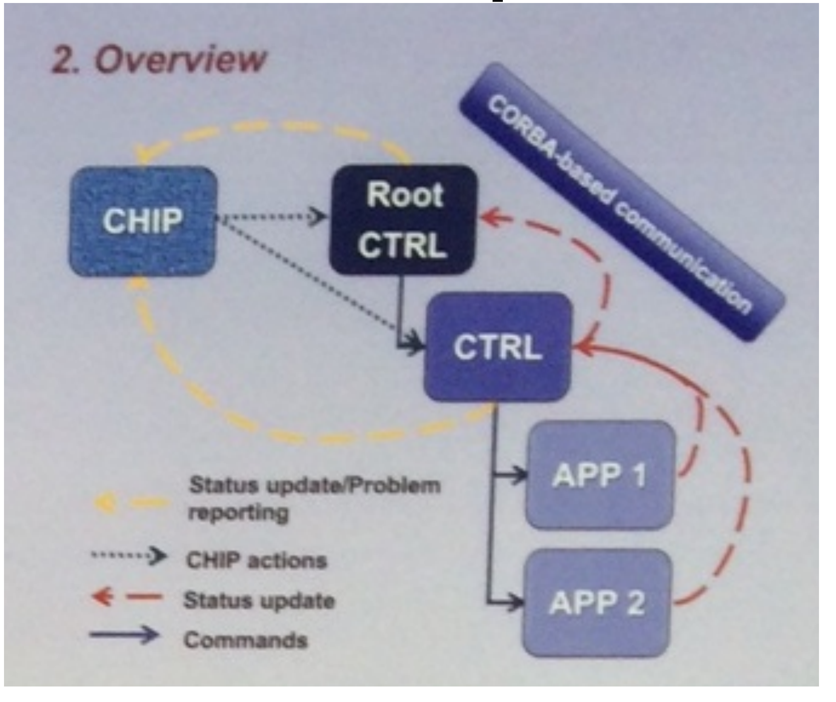
Messaging Systems



Presented by D. Arkhipkin / L. Magnoni

Improve Operations

- **The Error Reporting in the ATLAS TDAQ System**
- **Intelligent operations of the data acquisition system of the ATLAS Experiment at the LHC**



Presented by S. Kolos / G. Avolio

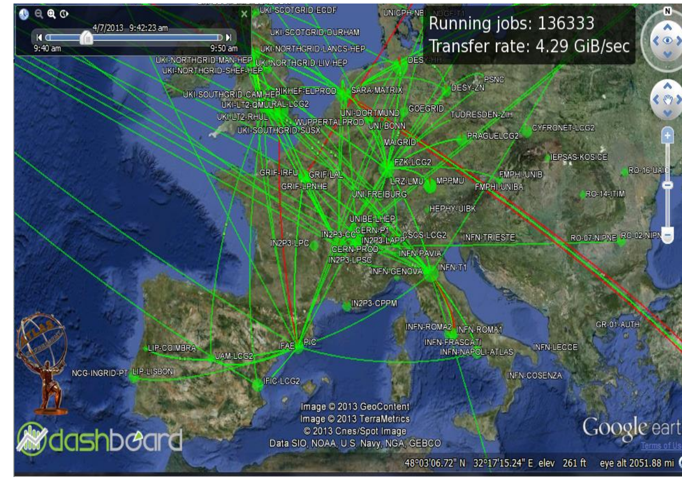
Improving Usability

- **Domain Specific Languages**
  - **CppLINQ**
    - Moving from imperative to declarative tools
    - Language integrated queries (C++ & SQL)
      - return from(range).where(is_prime).sum();
  - **LINQtoROOT**
    - Using Functional Languages and Declarative Programming to analyze ROOT data.
    - Functional queries over ROOT data in C#

Presented by V. Vasiliev / G. Watts

Improve Usability

- **Virtualization**
  - To extend beyond the grid: supercomputers and clouds
  - BELLE II Production System
  - BigPanDa
    - Location transparency of processing and data
  - DII-HEP project in Finland
  - Czech MetaCentrum
  - WLCG Tier-2 Prague

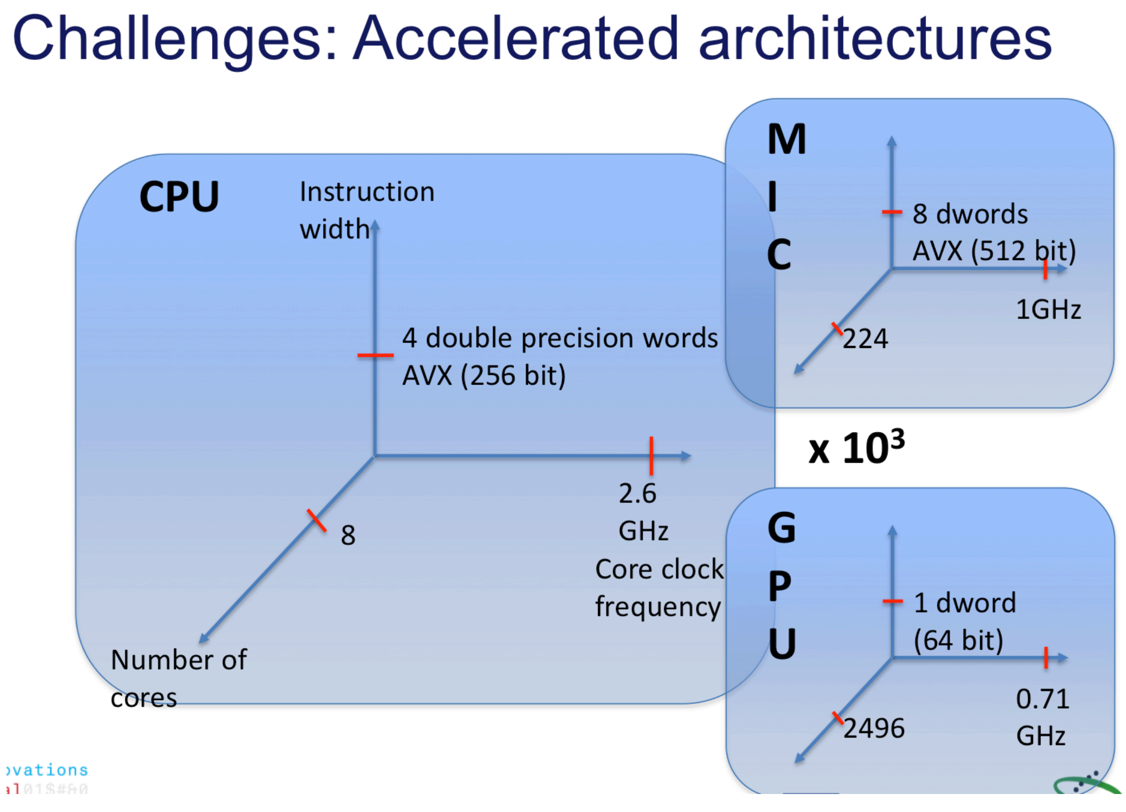


- virtualized infrastructure
  full-machine preemption, ondemand machines, virtual clusters
- distributed scheduling
- fairness model
  fairshare, multi-resource fairness

Presented by P. Krokovny / A. Klimentov / T. Lindén / S. Toth / D. Adamova,

The Future…

- **And Challenges of Scientific Computing**



- **Will not make our software simpler…**

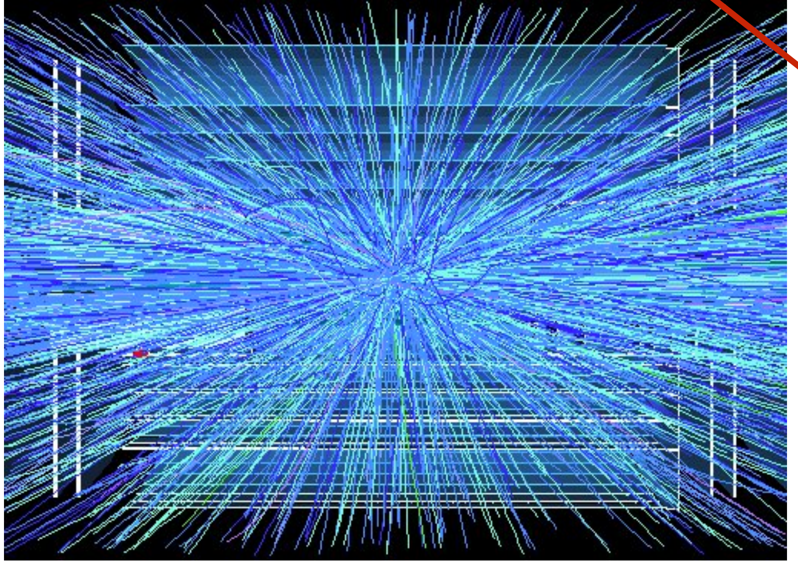
Conclusions I

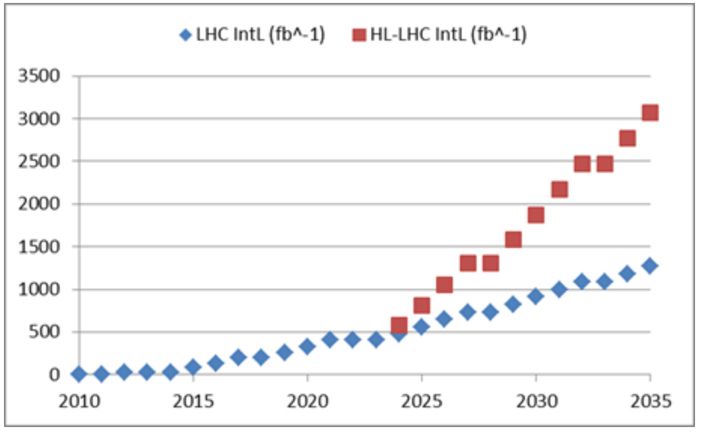
- In general we are preparing for:

Extreme Conditions

Event Complexity
x Rate

- Very high pile up
- Very high trigger acceptance rates
- Very challenging computing





Conclusions II

- **By making our software:**
  - More efficient
    - Concurrency, Vectorization, etc…
  - More flexible
    - To allow using more powerful and/or cheaper and/or power saving architectures: GPUs, ARMs, etc.
  - Requires a lot of work/expertise and becomes extremely complex
    - But we'd still like to keep it transparent and user-friendly (within frameworks, libraries, tools, etc)
  - We moved from evaluation to design & implementation
    - First results encouraging…