LHCb Computing Upgrade

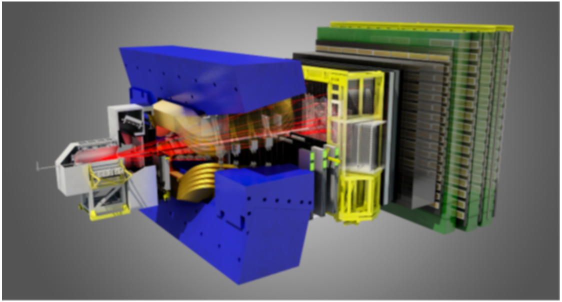
M.Corvo

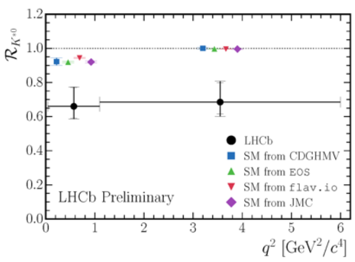
INFN and University of Ferrara

22 May 2017

1. Introduction
2. Computing challenge
3. Software solutions
   - Training
   - Data solution
   - Simulation solutions
4. Hardware resources
   - CPU
   - Storage
5. Analysis model
   - Working Group productions
6. Conclusions

The LHCb experiment Bignami

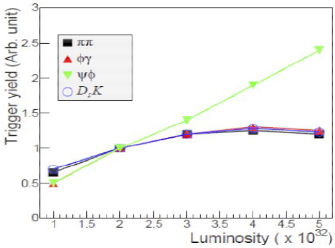


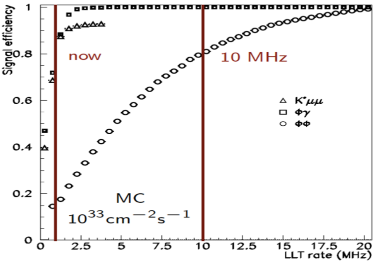


- LHCb is a single arm forward spectrometer designed for heavy flavor physics
- LHCb searches new physics in an *indirect way*, that is studying discrepancies wrt the Standard Model prediction.
- For example measuring an observable that the SM predicts to be equal to 1

Computing challenge(s) - The motivation

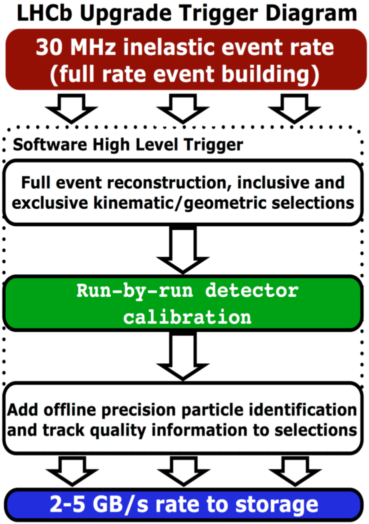
- Currently LHCb runs with an L0 trigger which reduces the bandwith of collected data from 40 MHz to 1 MHz, selecting the *interesting* events based on specific algorithms





- Due to the available bandwidth and the limited discrimination power of the hadronic L0 trigger, LHCb experiences the saturation of the trigger yield on the hadronic channels around $4 \cdot 10^{32} cm^{-2} s^{-1}$
- Increasing the first level trigger rate considerably increases the efficiency on the hadronic channels

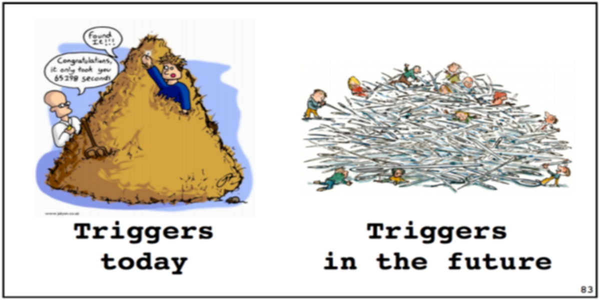
A triggerless solution



Run III Trigger configuration

- LHCb has already deployed a software trigger split in two, where the first stage performs a partial reconstruction and launches tasks for the detector calibration and alignment
- when calibration and alignment constants are available, the second stage of the trigger performs a full reconstruction whose quality is the same as the offline
- For the upgrade the L0 trigger will disappear and LHCb will exploit only a fully software trigger which will have an input rate of 30MHz

Numb3rs



- The lack of an L0 trigger combined with an increase in luminosity leads to a heavy load of data to manage and digest
- We calculated an average of 2 to $5 GB/s$ hitting the storage after the trigger processing (now it's $\approx 700 - 800\,MB/s$)
- This means $\approx$ **50 to 125TB/day**
- The result is a big strain both on CPU efficiency (we will not reject events, but only classify them) and on storage systems

Framework and Turbo

- For the upgrade the two core concepts we will exploit are: HLT split and Turbo data
- HLT split already tested and operational
  - This makes possible to have **offline quality** online calibration and alignment
- Turbo data format already commissioned
  - contains only signal information, plus some more selected ones (see next slides)
- Whatever exits the pit will be good for analysis, so we **must** be efficient in exploiting the HLT power

Framework and Turbo

- For the upgrade the two core concepts we will exploit are: HLT split and Turbo data
- HLT split already tested and operational
  - This makes possible to have **offline quality** online calibration and alignment
- Turbo data format already commissioned
  - contains only signal information, plus some more selected ones (see next slides)
- Whatever exits the pit will be good for analysis, so we **must** be efficient in exploiting the HLT power

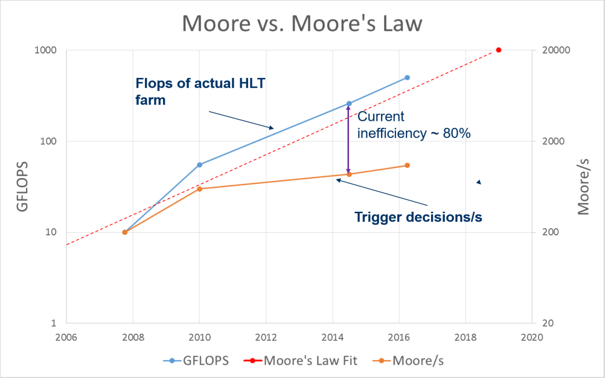
Framework and Turbo

- For the upgrade the two core concepts we will exploit are: HLT split and Turbo data
- HLT split already tested and operational
  - This makes possible to have **offline quality** online calibration and alignment
- Turbo data format already commissioned
  - contains only signal information, plus some more selected ones (see next slides)
- Whatever exits the pit will be good for analysis, so we **must** be efficient in exploiting the HLT power

Framework and Turbo

- For the upgrade the two core concepts we will exploit are: HLT split and Turbo data
- HLT split already tested and operational
  - This makes possible to have **offline quality** online calibration and alignment
- Turbo data format already commissioned
  - contains only signal information, plus some more selected ones (see next slides)
- Whatever exits the pit will be good for analysis, so we **must** be efficient in exploiting the HLT power

A new framework - why?



- The picture shows that currently we waste nearly 80% of computing power

A new framework - how?

- To fully exploit the hardware capabilities, LHCb started a (r)evolution with its framework, called Gaudi developed with Atlas
- Memory matters, so do cache misses
  - Drive algorithms towards vectorization and heavy use of AVX registers
  - Turn from AOS to SOA patterns
- Move towards full multithread
  - Turn the framework to a **functional** one, stateless algorithms run as indipendent tasks
- All these improvements require a deep restructuring of the code
- Algorithms must be redesigned and the data model must cope with thread safeness (avoid race conditions and 'performance killer' explicit locks)

A new framework - how?

- To fully exploit the hardware capabilities, LHCb started a (r)evolution with its framework, called Gaudi developed with Atlas
- Memory matters, so do cache misses
  - Drive algorithms towards vectorization and heavy use of AVX registers
  - Turn from AOS to SOA patterns
- Move towards full multithread
  - Turn the framework to a **functional** one, stateless algorithms run as indipendent tasks
- All these improvements require a deep restructuring of the code
- Algorithms must be redesigned and the data model must cope with thread safeness (avoid race conditions and 'performance killer' explicit locks)

A new framework - how?

- To fully exploit the hardware capabilities, LHCb started a (r)evolution with its framework, called Gaudi developed with Atlas
- Memory matters, so do cache misses
  - Drive algorithms towards vectorization and heavy use of AVX registers
  - Turn from AOS to SOA patterns
- Move towards full multithread
  - Turn the framework to a **functional** one, stateless algorithms run as indipendent tasks
- All these improvements require a deep restructuring of the code
- Algorithms must be redesigned and the data model must cope with thread safeness (avoid race conditions and 'performance killer' explicit locks)

A new framework - how?

- To fully exploit the hardware capabilities, LHCb started a (r)evolution with its framework, called Gaudi developed with Atlas
- Memory matters, so do cache misses
  - Drive algorithms towards vectorization and heavy use of AVX registers
  - Turn from AOS to SOA patterns
- Move towards full multithread
  - Turn the framework to a **functional** one, stateless algorithms run as indipendent tasks

- All these improvements require a deep restructuring of the code
- Algorithms must be redesigned and the data model must cope with thread safeness (avoid race conditions and 'performance killer' explicit locks)

A new framework - how?

- To fully exploit the hardware capabilities, LHCb started a (r)evolution with its framework, called Gaudi developed with Atlas
- Memory matters, so do cache misses
  - Drive algorithms towards vectorization and heavy use of AVX registers
  - Turn from AOS to SOA patterns
- Move towards full multithread
  - Turn the framework to a **functional** one, stateless algorithms run as indipendent tasks

- All these improvements require a deep restructuring of the code
- Algorithms must be redesigned and the data model must cope with thread safeness (avoid race conditions and 'performance killer' explicit locks)

A new framework - Where do we stand?

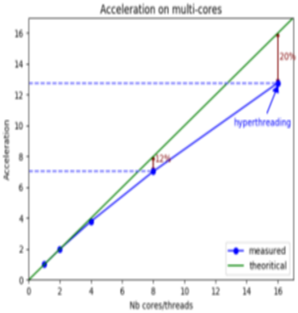
- New framework is there and so far almost 100 algorithms have been adapted to exploit it
- It's been partially ported to 2017 reconstruction sequence, with significant gain in performance for some subdetectors (e.g. the RICH)
- Performance measurements ongoing
  - Very good behavior with multithreading, good memory footprint and loss of only 10/20% CPU when using all cores

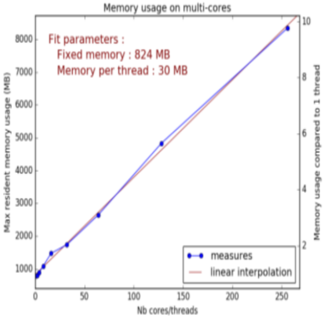
A new framework - Where do we stand?

- New framework is there and so far almost 100 algorithms have been adapted to exploit it
- It's been partially ported to 2017 reconstruction sequence, with significant gain in performance for some subdetectors (e.g. the RICH)
- Performance measurements ongoing
  - Very good behavior with multithreading, good memory footprint and loss of only 10/20% CPU when using all cores

A new framework - Where do we stand?

- New framework is there and so far almost 100 algorithms have been adapted to exploit it
- It's been partially ported to 2017 reconstruction sequence, with significant gain in performance for some subdetectors (e.g. the RICH)
- Performance measurements ongoing
  - Very good behavior with multithreading, good memory footprint and loss of only 10/20% CPU when using all cores





Spread the knowledge

- Software has considerably moved in the last years
  - Consider that in few years we switched from C++98 to C++17
- CPU power stopped moving towards higher frequencies, rather towards more tasks in parallel
- The last generation of physicists was used to think in an OO way
  - Huge chains of inheritance, big number of virtual functions
- HEP programming seems to turn back to the old good functional paradigm
  - → **Training is mandatory**
- C++ offers a wide range of smart solutions to improve performances
- Compilers also are no more the *black magic boxes*
  - Flags matter!

Spread the knowledge

- Software has considerably moved in the last years
  - Consider that in few years we switched from C++98 to C++17
- CPU power stopped moving towards higher frequencies, rather towards more tasks in parallel
- The last generation of physicists was used to think in an OO way
  - Huge chains of inheritance, big number of virtual functions
- HEP programming seems to turn back to the old good functional paradigm
  - → **Training is mandatory**
- C++ offers a wide range of smart solutions to improve performances
- Compilers also are no more the *black magic boxes*
  - Flags matter!

Spread the knowledge

- Software has considerably moved in the last years
  - Consider that in few years we switched from C++98 to C++17
- CPU power stopped moving towards higher frequencies, rather towards more tasks in parallel
- The last generation of physicists was used to think in an OO way
  - Huge chains of inheritance, big number of virtual functions
- HEP programming seems to turn back to the old good functional paradigm
  - → **Training is mandatory**
- C++ offers a wide range of smart solutions to improve performances
- Compilers also are no more the *black magic boxes*
  - Flags matter!

Spread the knowledge

- Software has considerably moved in the last years
  - Consider that in few years we switched from C++98 to C++17
- CPU power stopped moving towards higher frequencies, rather towards more tasks in parallel
- The last generation of physicists was used to think in an OO way
  - Huge chains of inheritance, big number of virtual functions
- HEP programming seems to turn back to the old good functional paradigm
  - → **Training is mandatory**
- C++ offers a wide range of smart solutions to improve performances
- Compilers also are no more the *black magic boxes*
  - Flags matter!

Spread the knowledge

- Software has considerably moved in the last years
  - Consider that in few years we switched from C++98 to C++17
- CPU power stopped moving towards higher frequencies, rather towards more tasks in parallel
- The last generation of physicists was used to think in an OO way
  - Huge chains of inheritance, big number of virtual functions
- HEP programming seems to turn back to the old good functional paradigm
  - $\rightarrow$ **Training is mandatory**
- C++ offers a wide range of smart solutions to improve performances
- Compilers also are no more the *black magic boxes*
  - Flags matter!

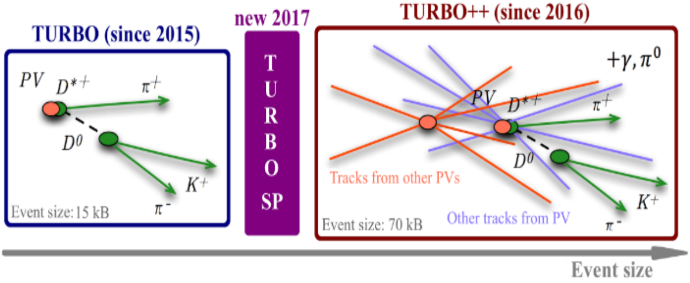
Training @*LHCb*

- LHCb started a wide training campaign to teach physicists the art of computer programming
- regular *Hackathons* take place at Cern during the computing workshops and software weeks

C++ course

Hackathon

A new data format



- To reduce the impact of the high throughput, LHCb designed a new data format called **Turbo**
- It's 'ready to use' for analysis, as offline quality reconstruction and PID allow to record just the decay tracks of interest

A new data format

- Turbo events are very 'thin' objects, nominal size should be 10KB thus opening the possibility to record more events
  - Some analysis will still need the full event recorded, so the output of the trigger will be a tunable mixture of Turbo and Full streams that can fit the 5 GB/s throughput
- Storage will be no more driven by the rate but by the total bandwith
  - The range of 2 to 5 GB/s allows a wide range of bandwidth, from a minimum 20 KHz (2 GB/s with 100 kB full events) to a maximum 100 kHz (5 GB/s with 50 kB Turbo events)

A new data format

- Turbo events are very 'thin' objects, nominal size should be 10KB thus opening the possibility to record more events
  - Some analysis will still need the full event recorded, so the output of the trigger will be a tunable mixture of Turbo and Full streams that can fit the 5 GB/s throughput
- Storage will be no more driven by the rate but by the total bandwith
  - The range of 2 to 5 GB/s allows a wide range of bandwidth, from a minimum 20 KHz (2 GB/s with 100 kB full events) to a maximum 100 kHz (5 GB/s with 50 kB Turbo events)

A new data format

- Turbo events are very 'thin' objects, nominal size should be 10KB thus opening the possibility to record more events
  - Some analysis will still need the full event recorded, so the output of the trigger will be a tunable mixture of Turbo and Full streams that can fit the 5 GB/s throughput
- Storage will be no more driven by the rate but by the total bandwith
  - The range of 2 to 5 GB/s allows a wide range of bandwidth, from a minimum 20 KHz (2 GB/s with 100 kB full events) to a maximum 100 kHz (5 GB/s with 50 kB Turbo events)

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - $\rightarrow$ TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - → TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - $\rightarrow$ TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - $\rightarrow$ TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - $\rightarrow$ TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A new data format

- A sharp cut in the information content of the event is way too aggressive
  - We need to keep not only signal candidates, but also some reconstructed objects
  - $\rightarrow$ TurboSP (Turbo with Selective Persistence)
- Save disk space while permitting more information stored in the event
- As the physicist must have a clear analysis strategy, the selection of what to persist is crucial. Remember that what you don't write, is gone for good . . .
  - and your selection will run directly in the trigger, that is you'll have a one shot possibility to grab what you need

A word on simulation

- Having the full reconstruction held by the online farm, the strain on CPU moves to simulation
- During Run I the amount of simulated events was 15% of collected data
  - Increasing to 25 – 30% in Run II
- Now that trigger is about *classifying* rather than rejecting events, should we simulate everything coming out from the pit?
  - Not a chance, at least with the current generation+simulation software
- LHCb must move to a parametrized and fast simulation framework

A word on simulation

- Having the full reconstruction held by the online farm, the strain on CPU moves to simulation
- During Run I the amount of simulated events was 15% of collected data
  - Increasing to $25-30\%$ in Run II
- Now that trigger is about *classifying* rather than rejecting events, should we simulate everything coming out from the pit?
  - Not a chance, at least with the current generation+simulation software
- LHCb must move to a parametrized and fast simulation framework

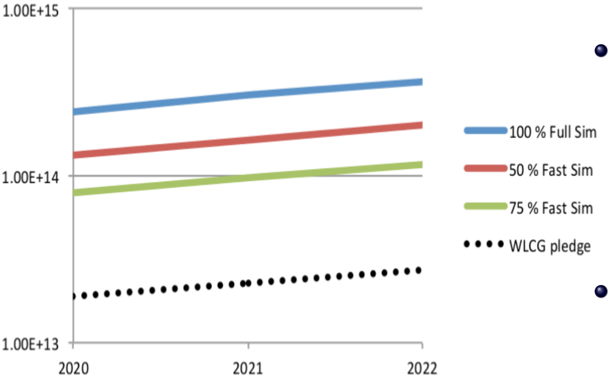
A word on simulation

- Having the full reconstruction held by the online farm, the strain on CPU moves to simulation
- During Run I the amount of simulated events was 15% of collected data
  - Increasing to $25 - 30\%$ in Run II
- Now that trigger is about *classifying* rather than rejecting events, should we simulate everything coming out from the pit?
  - Not a chance, at least with the current generation+simulation software
- LHCb must move to a parametrized and fast simulation framework

A word on simulation

- Having the full reconstruction held by the online farm, the strain on CPU moves to simulation
- During Run I the amount of simulated events was 15% of collected data
  - Increasing to $25 - 30\%$ in Run II
- Now that trigger is about *classifying* rather than rejecting events, should we simulate everything coming out from the pit?
  - Not a chance, at least with the current generation+simulation software
- LHCb must move to a parametrized and fast simulation framework

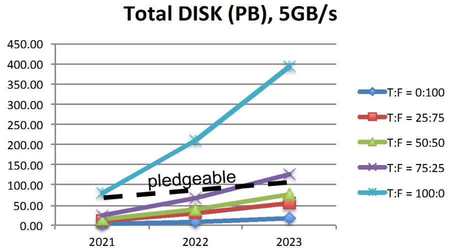
No free lunch anymore

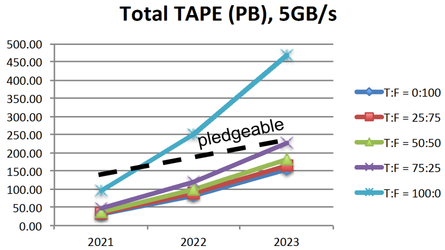


Requested HS06.s Sim evt vs pledges

- Assuming:
  1. 600 HS06.s for a full simulated event (currently already 3-5 times this value)
  2. FastSim $\approx$ 10% of FullSim
- Resources, which we assume will be available, won't fit the requests by far

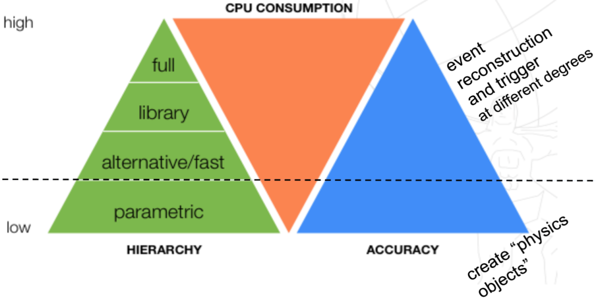
No free lunch anymore





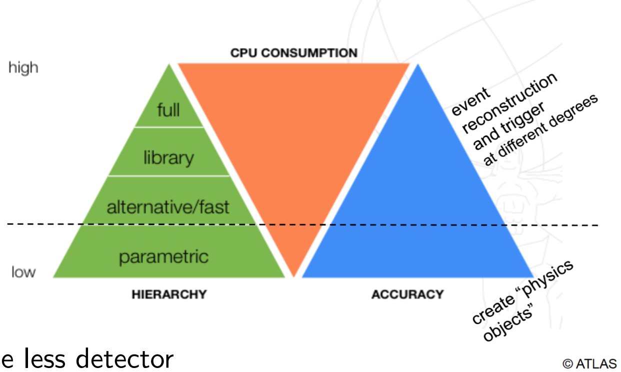
- Again these pictures show that the resources are far from being sufficient

Fast(er) simulation strategy



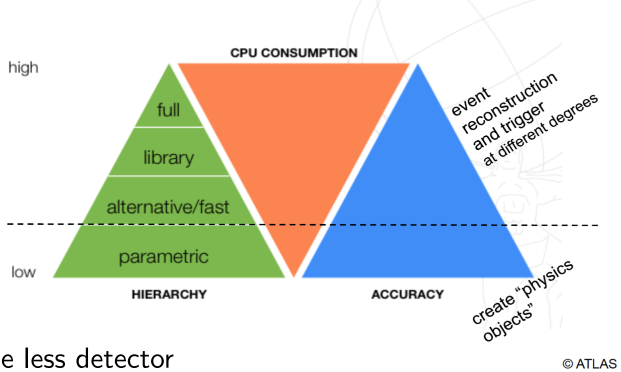
1. Simulate less detector
   - Simulate partial detector by turning off or fast-simulating components with large contribution
2. Simulate less particles
   - Simulate only particle from signal decay
   - Re-use of the underlying event (simulate one underlying for N signal)
3. Any linear combination of the two

Fast(er) simulation strategy



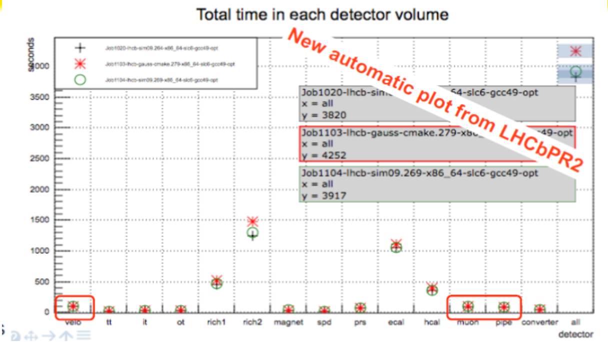
1. Simulate less detector
   - Simulate partial detector by turning off or fast-simulating components with large contribution
2. Simulate less particles
   - Simulate only particle from signal decay
   - Re-use of the underlying event (simulate one underlying for N signal)
3. Any linear combination of the two

Fast(er) simulation strategy



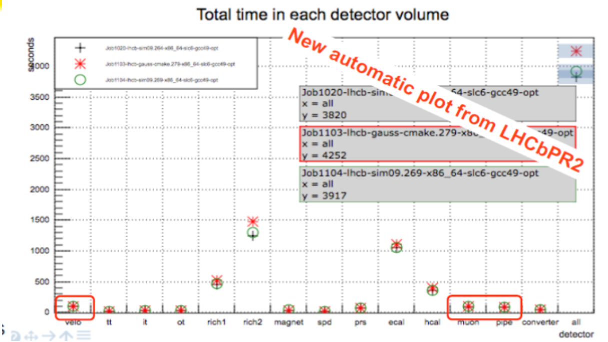
1. Simulate less detector
   - Simulate partial detector by turning off or fast-simulating components with large contribution
2. Simulate less particles
   - Simulate only particle from signal decay
   - Re-use of the underlying event (simulate one underlying for N signal)
3. Any linear combination of the two

Fast(er) simulation strategy



- Use of more performant tools (e.g. Geant 4.10 migration)
  - Some speedup plus better memory footprint in a multi-threaded environment
- Use of modern geometry packages
- More powerful hardware? (GPU, FPGA, KNL . . .)
  - yes, but commissioning of these tools is very time consuming as nothing in the current framework is able to handle them

Fast(er) simulation strategy



- Use of more performant tools (e.g. Geant 4.10 migration)
  - Some speedup plus better memory footprint in a multi-threaded environment
- Use of modern geometry packages
- More powerful hardware? (GPU, FPGA, KNL ...)
  - yes, but commissioning of these tools is very time consuming as nothing in the current framework is able to handle them

Fast(er) simulation strategy

- Innovative (at least for HEP) solutions involving ML techniques to generate realistic distributions of Calorimeter hits from training samples
  - Developed for image classification, the approach has been already applied to jets generation
- Fully parametric super fast simulation
  - Based on Delphes package
- Next step is to try to adopt a simplified geometry which should also guarantee some improvements in time

Fast(er) simulation strategy

- Innovative (at least for HEP) solutions involving ML techniques to generate realistic distributions of Calorimeter hits from training samples
  - Developed for image classification, the approach has been already applied to jets generation
- Fully parametric super fast simulation
  - Based on Delphes package
- Next step is to try to adopt a simplified geometry which should also guarantee some improvements in time

Fast(er) simulation strategy

- Innovative (at least for HEP) solutions involving ML techniques to generate realistic distributions of Calorimeter hits from training samples
  - Developed for image classification, the approach has been already applied to jets generation
- Fully parametric super fast simulation
  - Based on Delphes package
- Next step is to try to adopt a simplified geometry which should also guarantee some improvements in time

CPU resources

- CPU consumption will be driven by simulation
  - recall that **offline** processing will turn into **online** processing
- Current distributed model will stay
  - Plus the exploitation of different opportunistic resources (BOINC, TSystems . . . AWS?)
- Rethink the analysis model

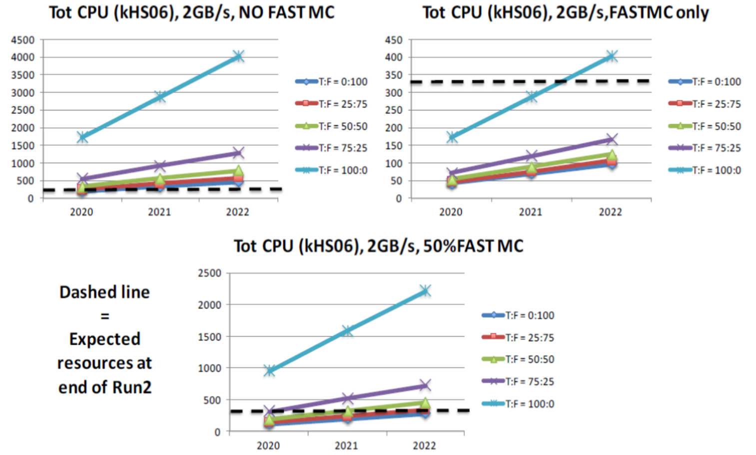
CPU resources

- CPU consumption will be driven by simulation
  - recall that **offline** processing will turn into **online** processing
- Current distributed model will stay
  - Plus the exploitation of different opportunistic resources (BOINC, TSystems . . . AWS?)
- Rethink the analysis model

CPU resources

- CPU consumption will be driven by simulation
  - recall that **offline** processing will turn into **online** processing
- Current distributed model will stay
  - Plus the exploitation of different opportunistic resources (BOINC, TSystems . . . AWS?)
- Rethink the analysis model

CPU resources



- Currently we could survive if we only had 25% of Turbo and 75% of Full
  - but this goes against our triggerless strategy and the attempt to reduce storage strains with Turbo . . .

Storage

- Currently we keep two tape copies of raw and reconstructed data
- Two disk copies of stripped data for the analysis
- Within the computing upgrade we'll have no more raw, reconstructed and stripped data, but only Turbo (at least Turbo will be the dominant data format)
- We'll keep two copies of Turbo and one of MC

Storage

- Currently we keep two tape copies of raw and reconstructed data
- Two disk copies of stripped data for the analysis
- Within the computing upgrade we'll have no more raw, reconstructed and stripped data, but only Turbo (at least Turbo will be the dominant data format)
- We'll keep two copies of Turbo and one of MC

Storage

- Currently we keep two tape copies of raw and reconstructed data
- Two disk copies of stripped data for the analysis
- Within the computing upgrade we'll have no more raw, reconstructed and stripped data, but only Turbo (at least Turbo will be the dominant data format)
- We'll keep two copies of Turbo and one of MC

Storage

- Currently we keep two tape copies of raw and reconstructed data
- Two disk copies of stripped data for the analysis
- Within the computing upgrade we'll have no more raw, reconstructed and stripped data, but only Turbo (at least Turbo will be the dominant data format)
- We'll keep two copies of Turbo and one of MC

Analysis model

- Previously considered indexing of events proved to be too heavy in terms of modifications in Root persistency and compression
- Keep the current model based on **streaming** of the Turbo data
  - Produce a few (e.g. 10) 'level one' streams with Tesla, the software used to convert the data from HLT to Root format
  - Produce again a few (e.g. 10) 'second level' streams with a specific streaming application and merge, in case, the streams
- At the end each run will be made of O(100) streams with 100 files, each of 2GB size

Analysis model

- Previously considered indexing of events proved to be too heavy in terms of modifications in Root persistency and compression
- Keep the current model based on **streaming** of the Turbo data
  - Produce a few (e.g. 10) 'level one' streams with Tesla, the software used to convert the data from HLT to Root format
  - Produce again a few (e.g. 10) 'second level' streams with a specific streaming application and merge, in case, the streams
- At the end each run will be made of O(100) streams with 100 files, each of 2GB size

WG productions

- LHCb started already to use Working Group productions
- Centrally managed
  - WG define the needs and organize the requests
  - Output data are saved in a common location and available to WG members (and also to others)
  - Data are registered to the central BK system
  - Less scattered access to data

WG productions

- LHCb started already to use Working Group productions
- Centrally managed
  - WG define the needs and organize the requests
  - Output data are saved in a common location and available to WG members (and also to others)
  - Data are registered to the central BK system
  - Less scattered access to data

WG productions

- WG productions model can drive data management
- Cleaner movement of files and simpler data management operations
- More control on prioritization
  - Users should no more care about 'how long will my jobs stay queued?'

WG productions

- WG productions model can drive data management
- Cleaner movement of files and simpler data management operations
- More control on prioritization
  - Users should no more care about 'how long will my jobs stay queued?'

WG productions

- WG productions model can drive data management
- Cleaner movement of files and simpler data management operations
- More control on prioritization
  - Users should no more care about 'how long will my jobs stay queued?'

Conclusions

- Computing upgrade ideas well on their way
  - Computing Upgrade TDR is due end of 2017
  - Trigger, framework, data format, software, analysis ...
- Innovations are mostly welcome, but with 'a grain of salt'
- Paradox: trying to shrink the event size to save storage, we end up with more need for storage and CPU
  - But this is justified by the fact that with a given and fixed bandwidth, we are able to collect more physics

Conclusions

- Biggest effort is (IMHO) convince physicists that computing resources are not infinite but **finite**
  - You cannot push an elephant into a fish bowl
- Physicists should think a little more as computer scientists
- This is why **training is crucial**
- At INFN level, of course, but also at university level