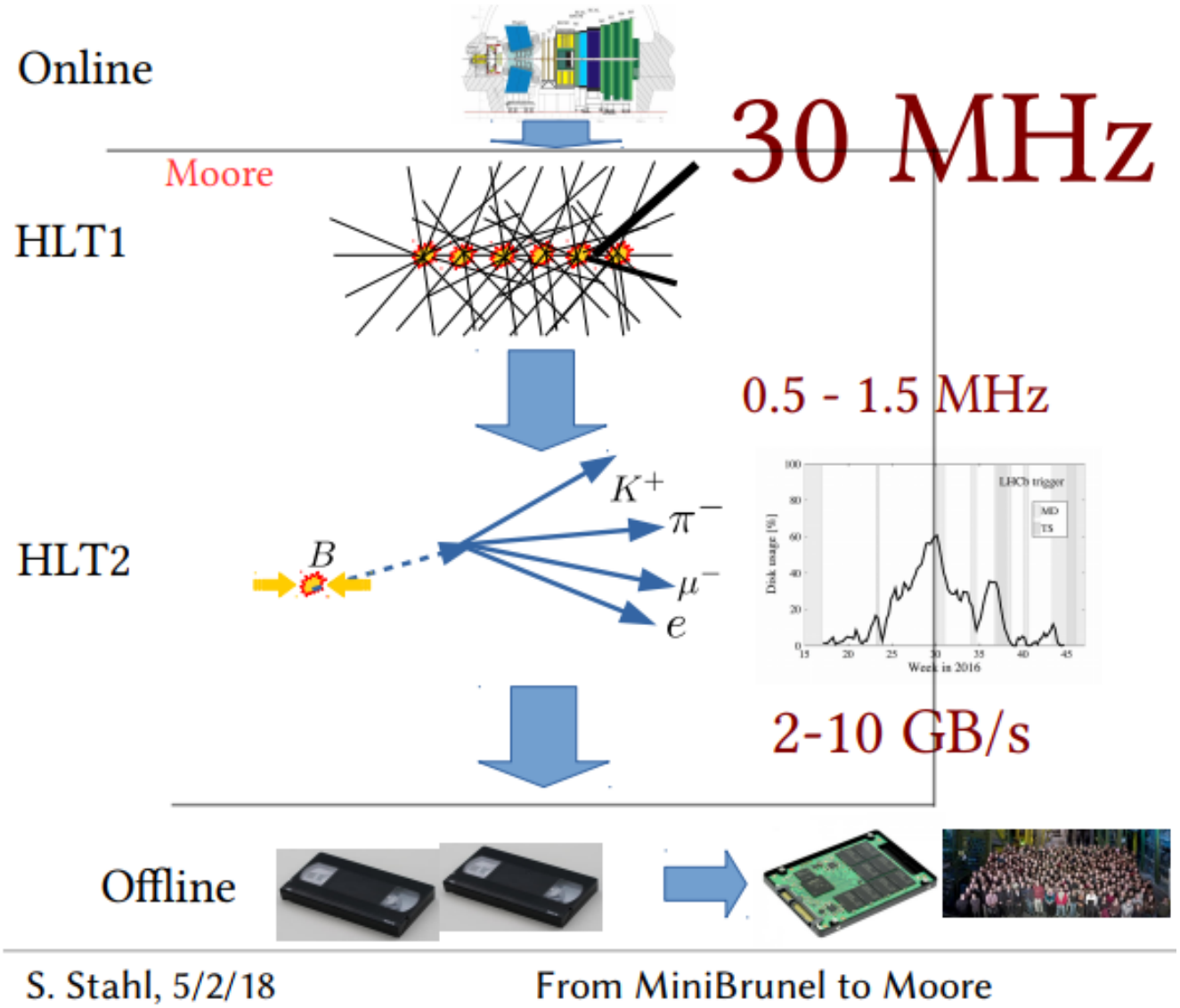
Overview

Monir Hadji

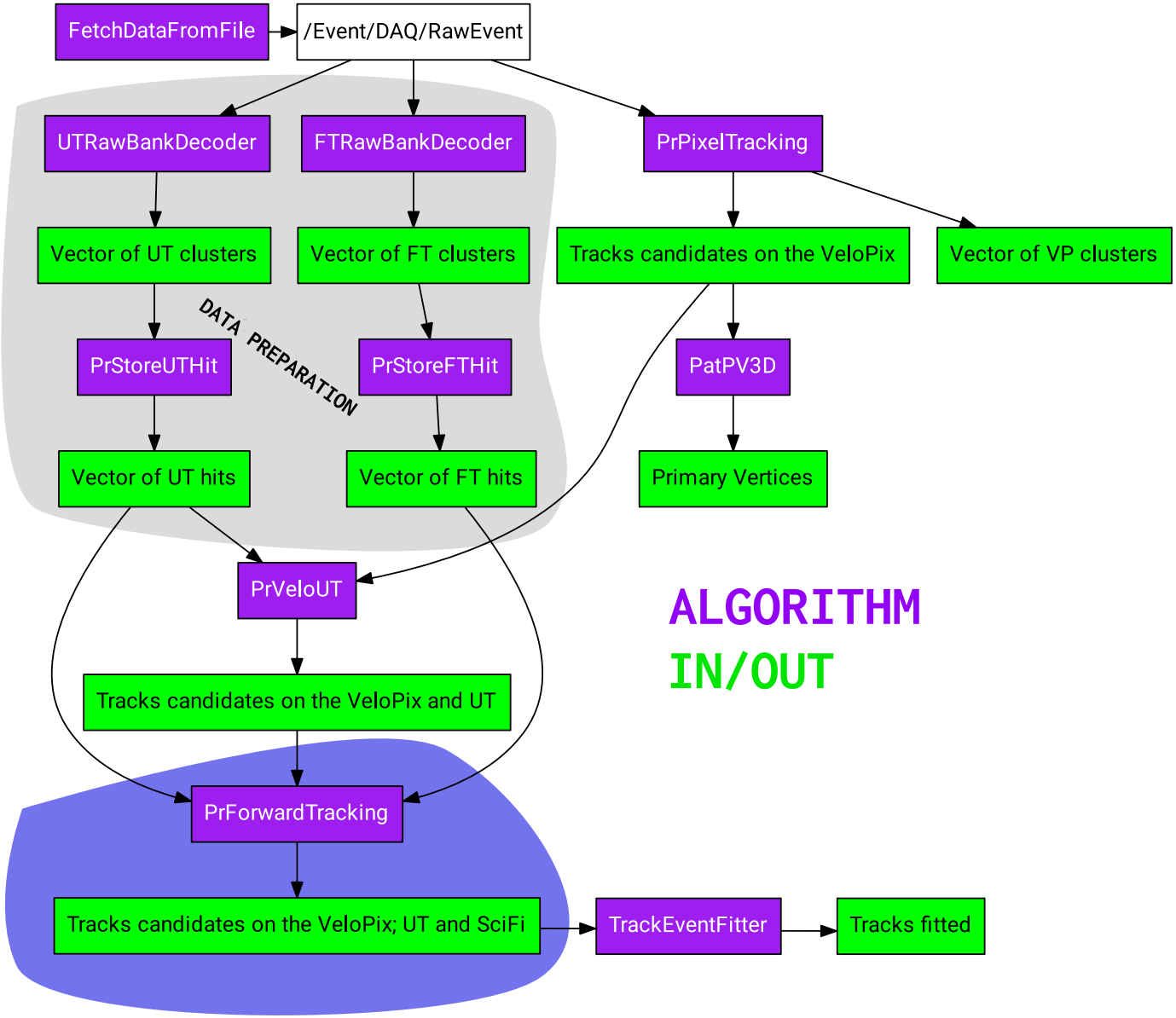
Université de Lyon

March 21, 2018

Upgrade global view



HLT1 Sequence



Overview

Data preparation process

Input

- Raw data of the detector

Output

- Hits information on a good structure and an optimized order for easy access during pattern recognition

Which data?

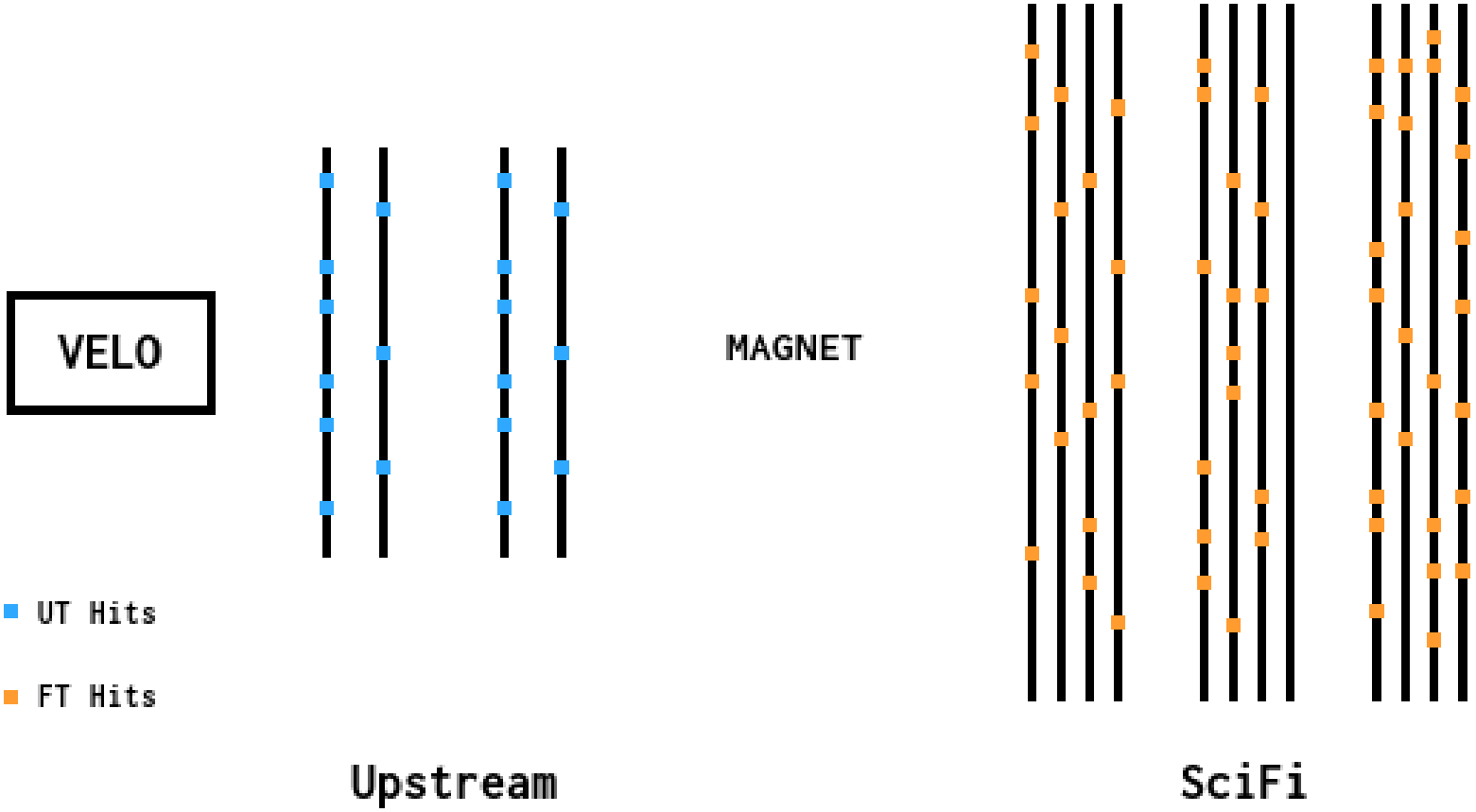
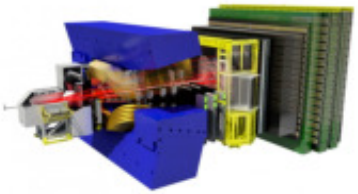


Figure: Schematic detector structure - scale absolutely not respected

Global view of the process



→ **RawBanks** →

`{FT|UT}RawBankDecoder`

Decode using informations from geometry

↓

`{FT|UT} CLUSTERS`

Difficult to extract some informations for pattern recognition

↓

`PrStore{FT|UT}Hit`

Construct FT UT Hits with common info and with a specific sort for easy access

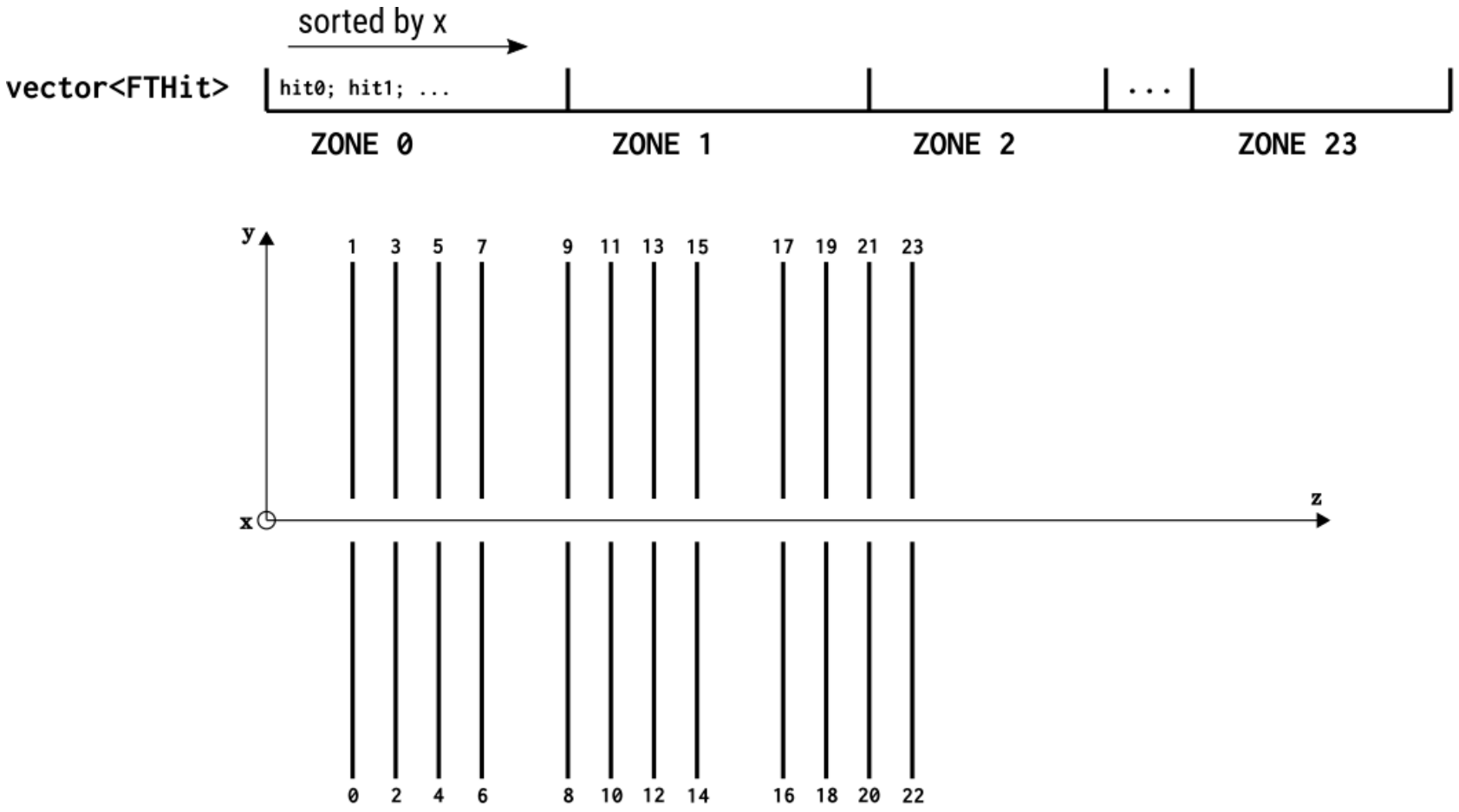
was inefficient mainly because agnostic of the initial input cluster order

↓

`{FT|UT} HITS`

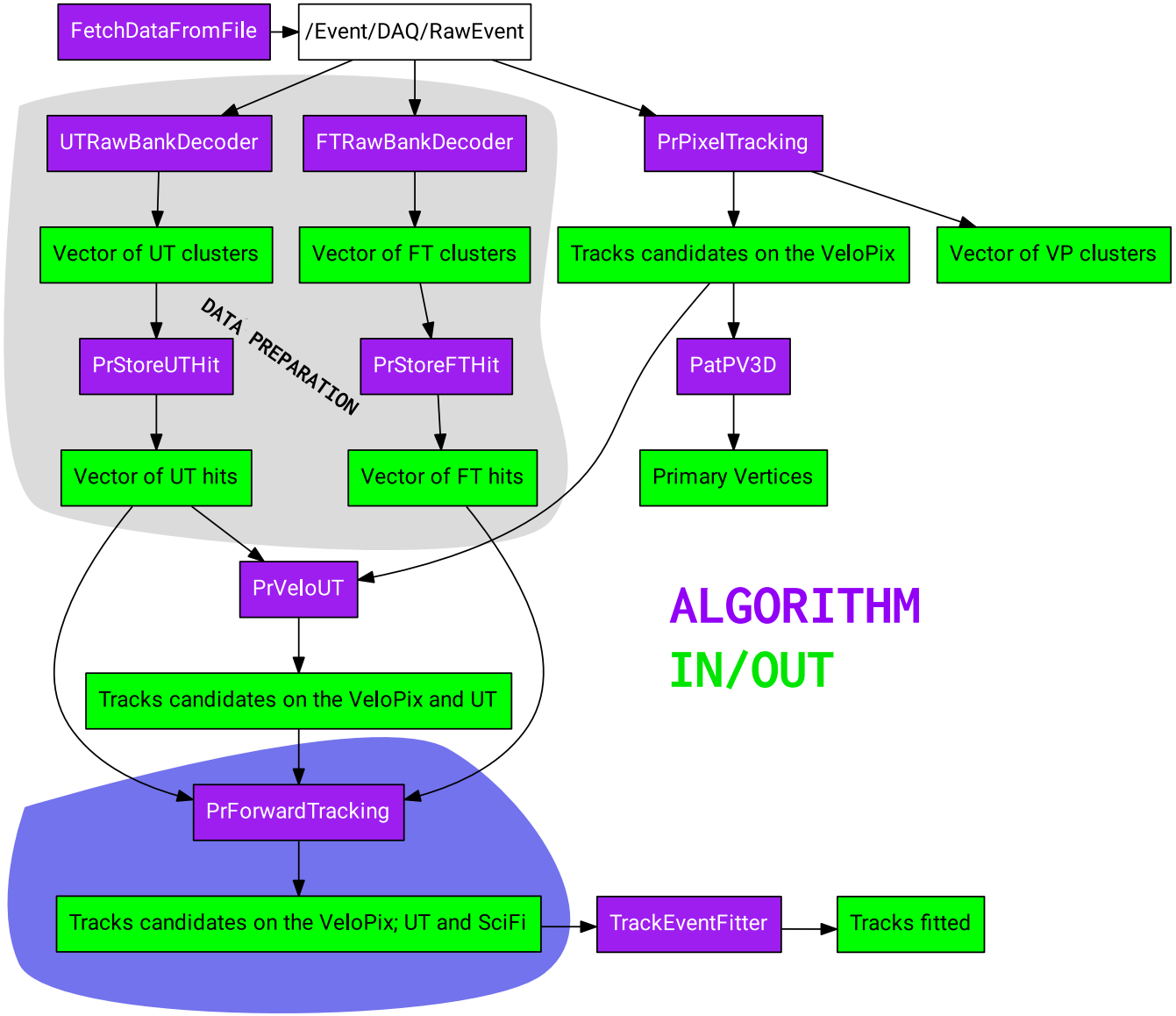
Aims to be with a structure optimized as much as possible for pattern recognition

Output structure description for FT hits



-25% on this part by changing the process to perform this sort Rec!750 Rec!787 Rec!937

Next: PrForwardTracking



Overview

Forward tracking process

Algo: PrForwardTracking

Input

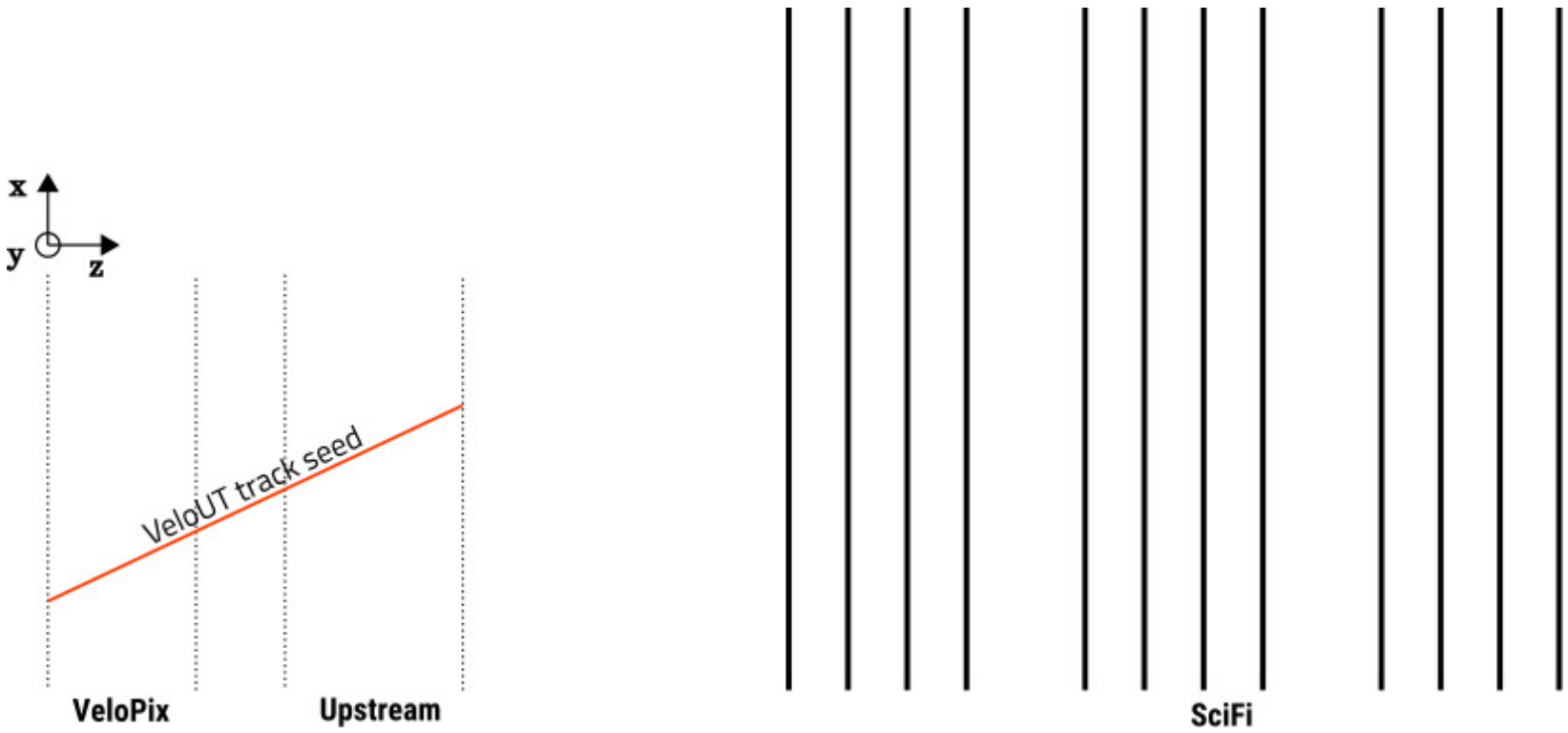
- FT Hits for the whole pattern recognition algorithm
- UT Hits for a final global fit to validate output candidates

Output

- Tracks candidates propagated on the SciFi

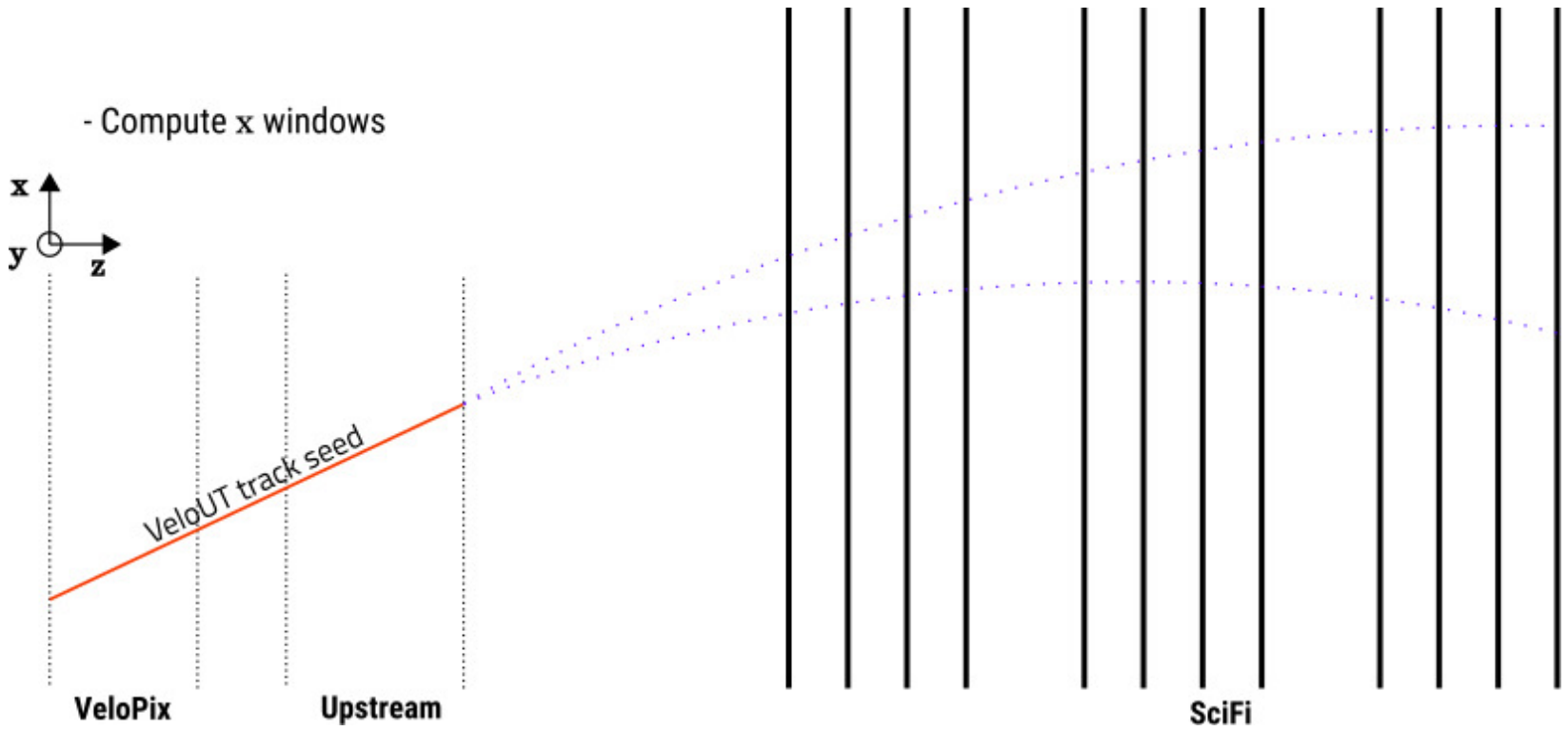
Global view of the process

Algo: PrForwardTracking



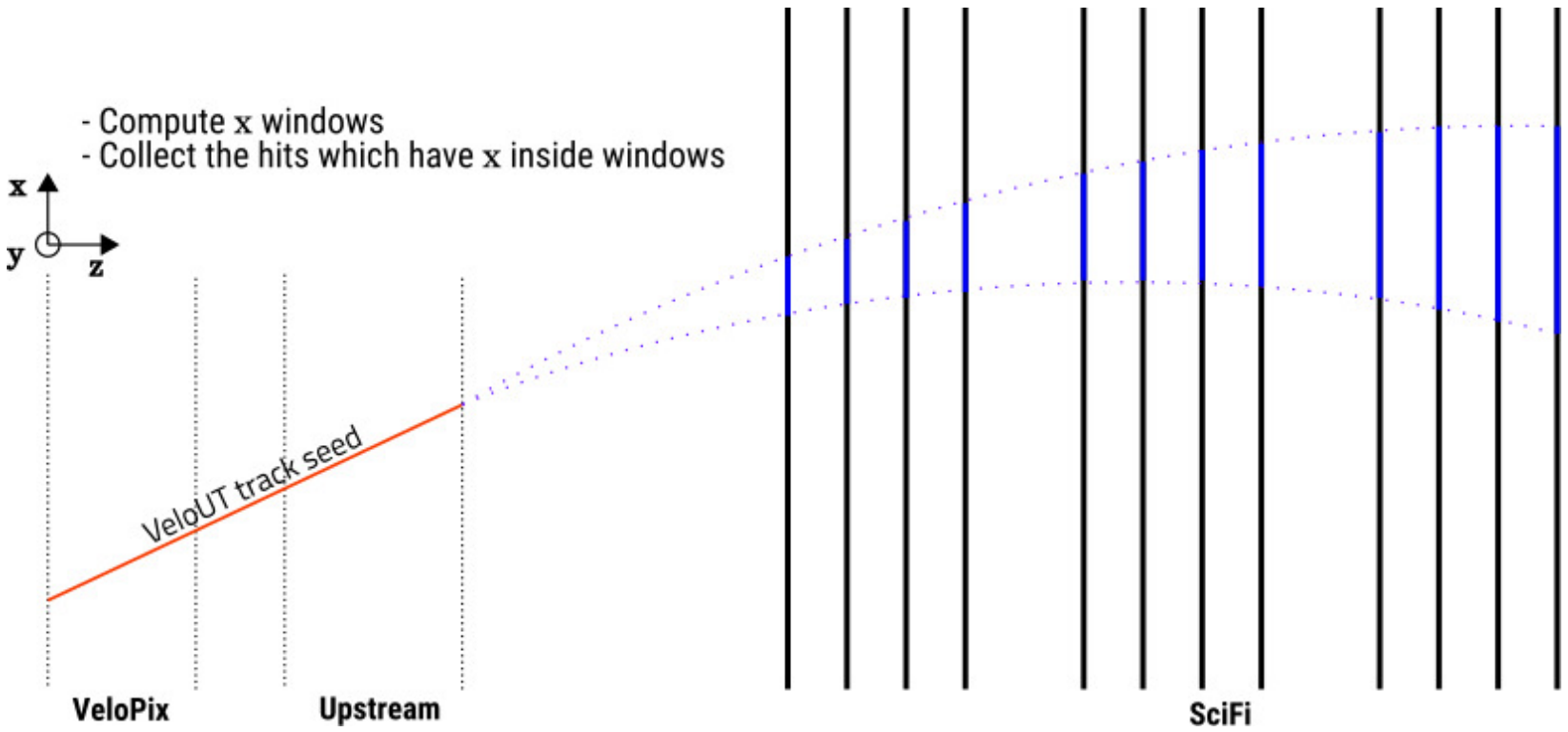
Global view of the process

Algo: PrForwardTracking



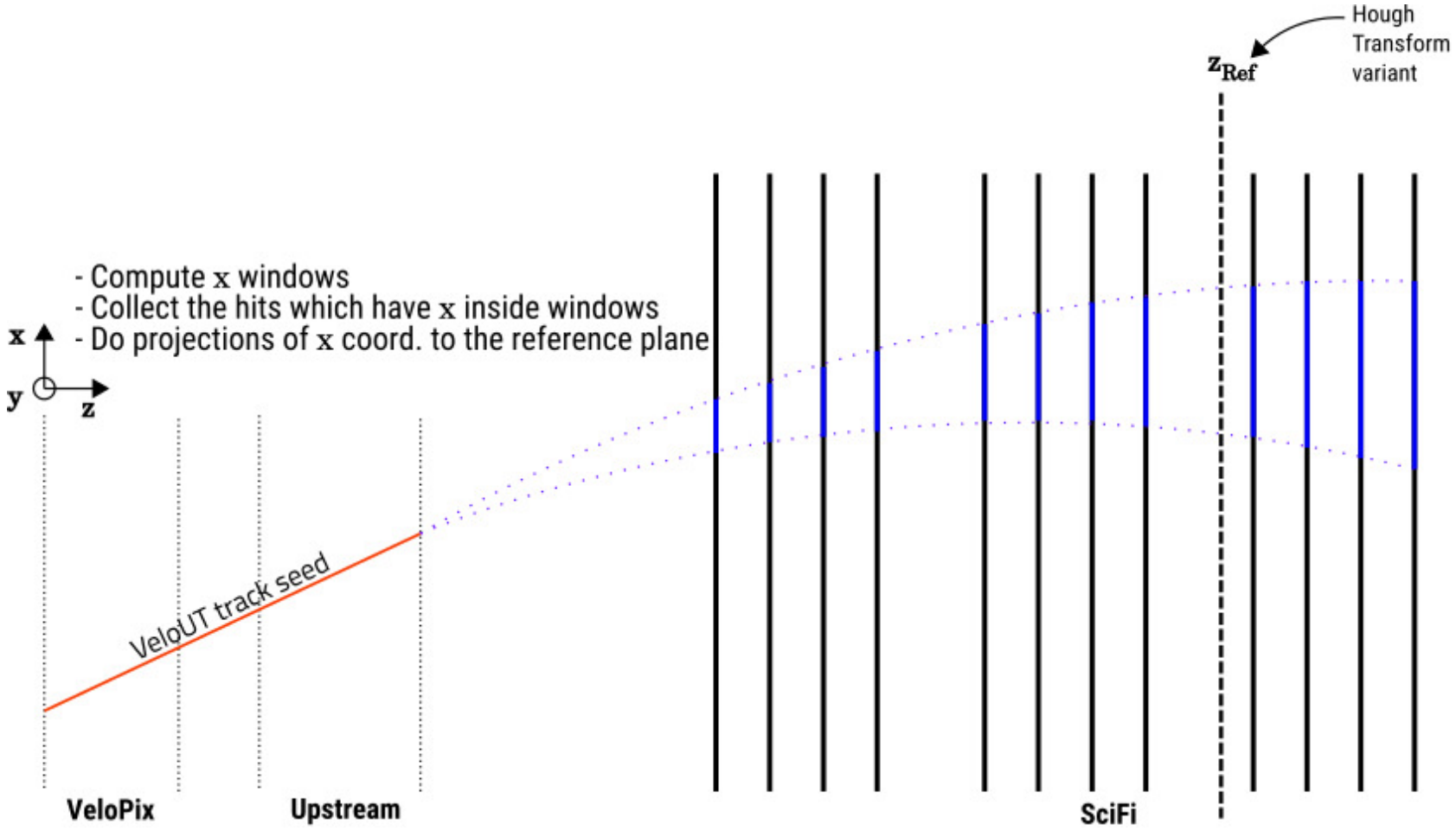
Global view of the process

Algo: PrForwardTracking



Global view of the process

Algo: PrForwardTracking



Global view of the process

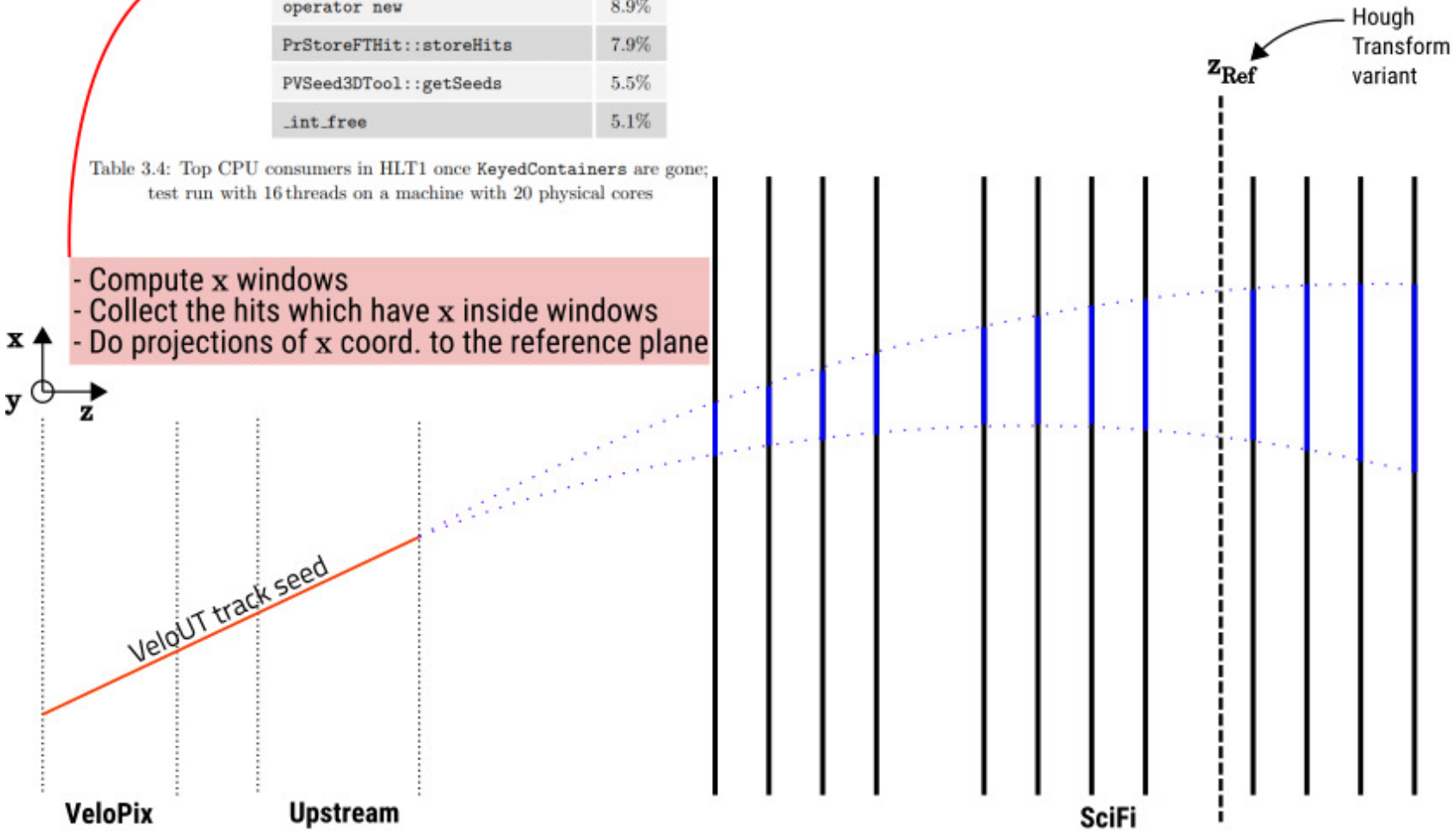
Algo: PrForwardTracking

From TDR

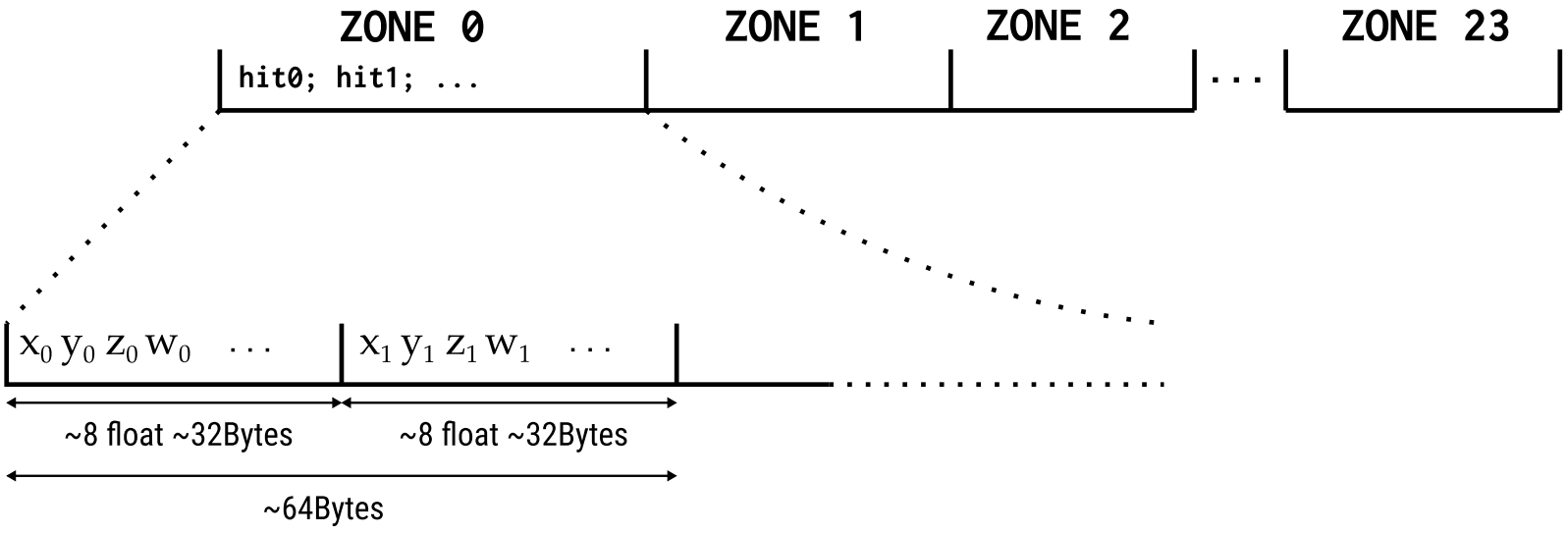
Function	CPU
PrPixelTracking::bestHit	19.2%
PrForwardTool::collectAllXHits	9.9%
operator new	8.9%
PrStoreFTHit::storeHits	7.9%
PVSeed3DTool::getSeeds	5.5%
_int_free	5.1%

Table 3.4: Top CPU consumers in HLT1 once KeyedContainers are gone; test run with 16 threads on a machine with 20 physical cores

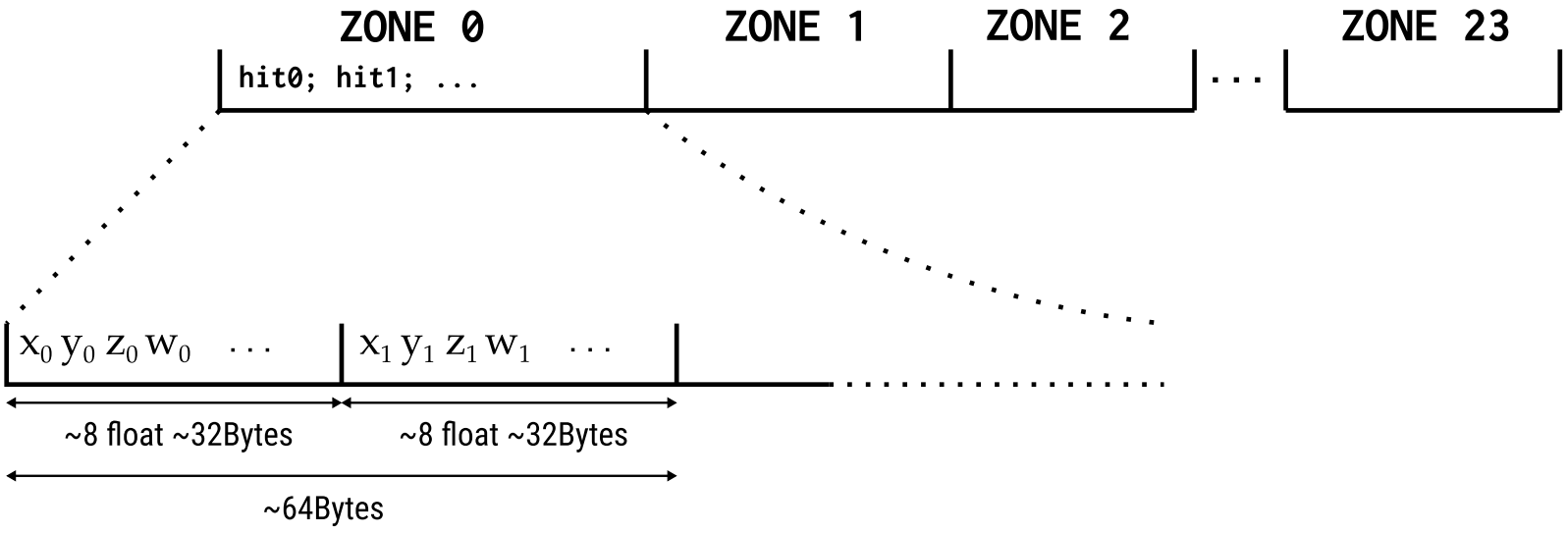
- Compute x windows
- Collect the hits which have x inside windows
- Do projections of x coord. to the reference plane



The memory layout for hits

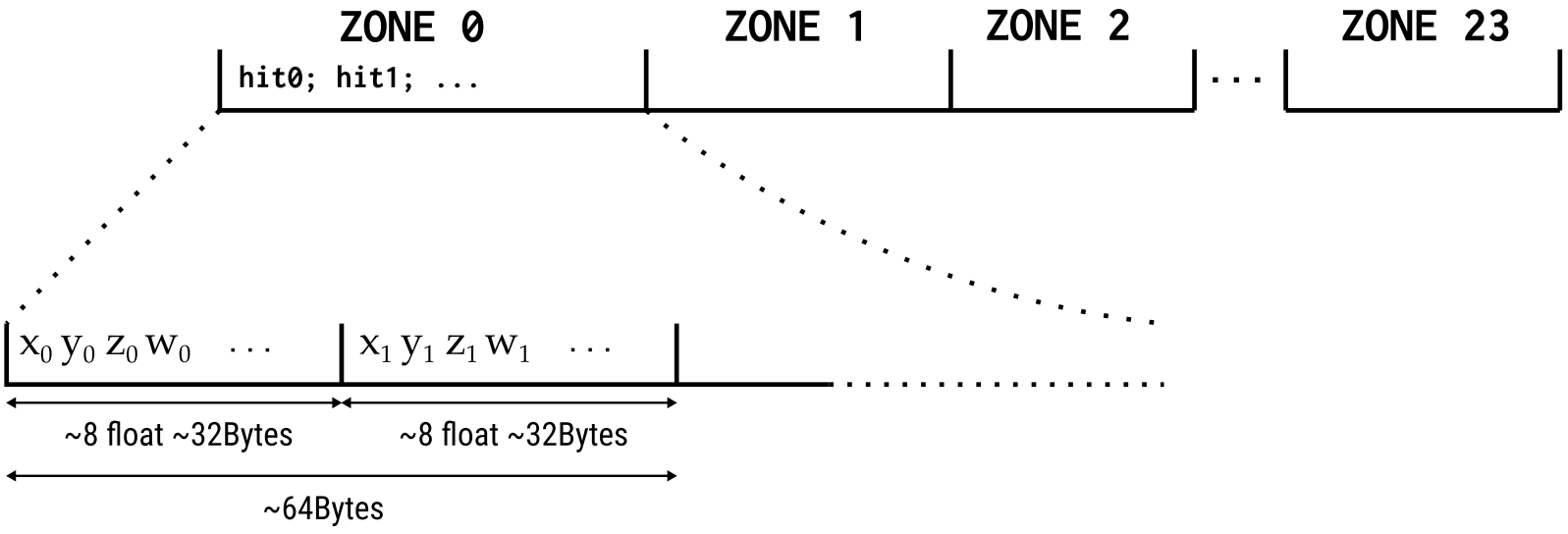


The memory layout for hits



a L1 cacheline (64Bytes) filled and **I can access only two x**

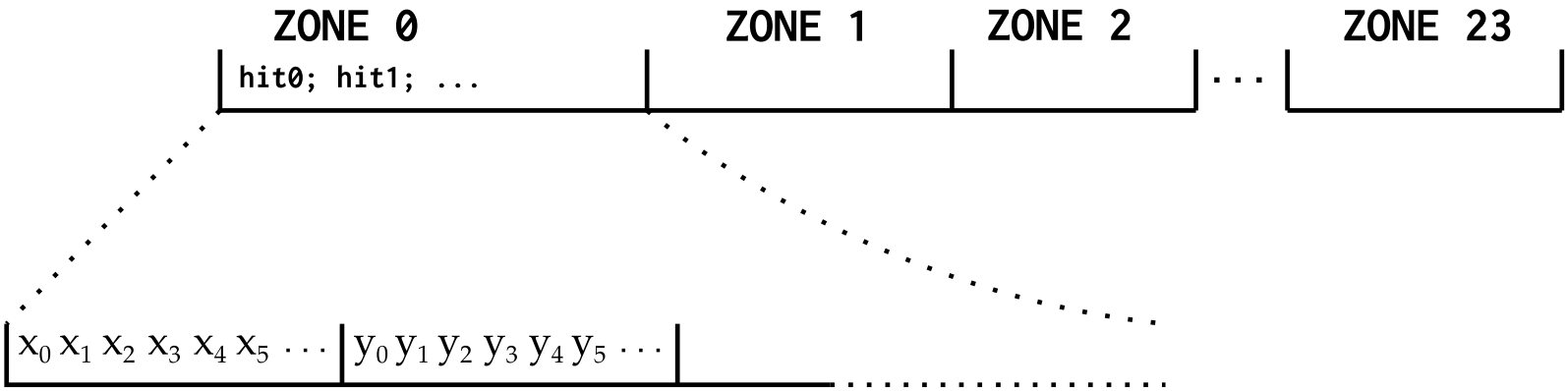
The memory layout for hits



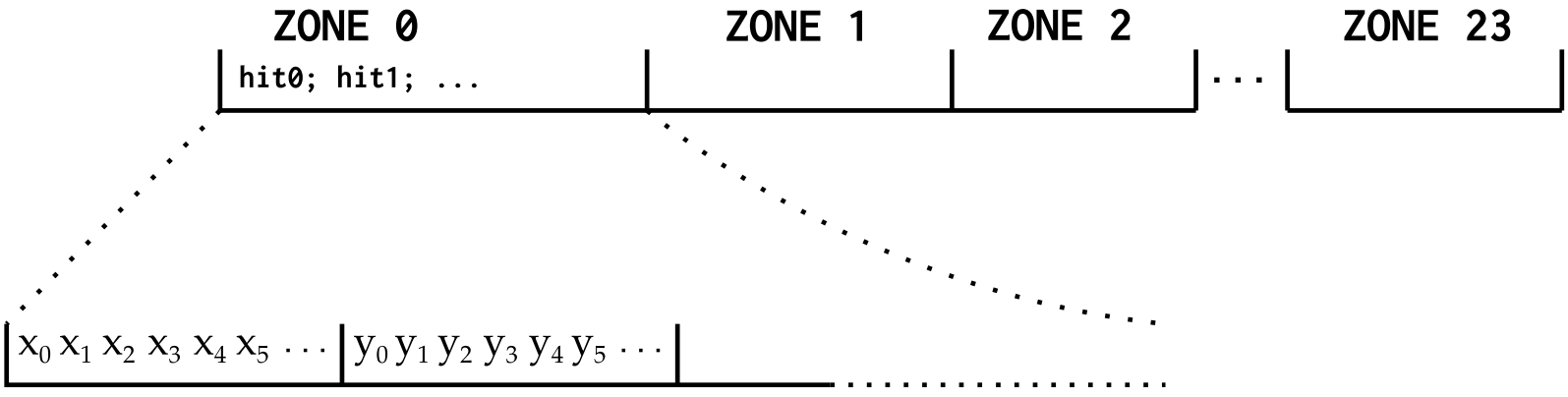
a L1 cacheline (64Bytes) filled and **I can access only two x**

would be good to have a better memory locality

Memory layout alternative



Memory layout alternative



a L1 cacheline (64Bytes) filled and **I can access `64Bytes/sizeof(float)=16` x**

SOA here, good idea?

Difficulties and hopes

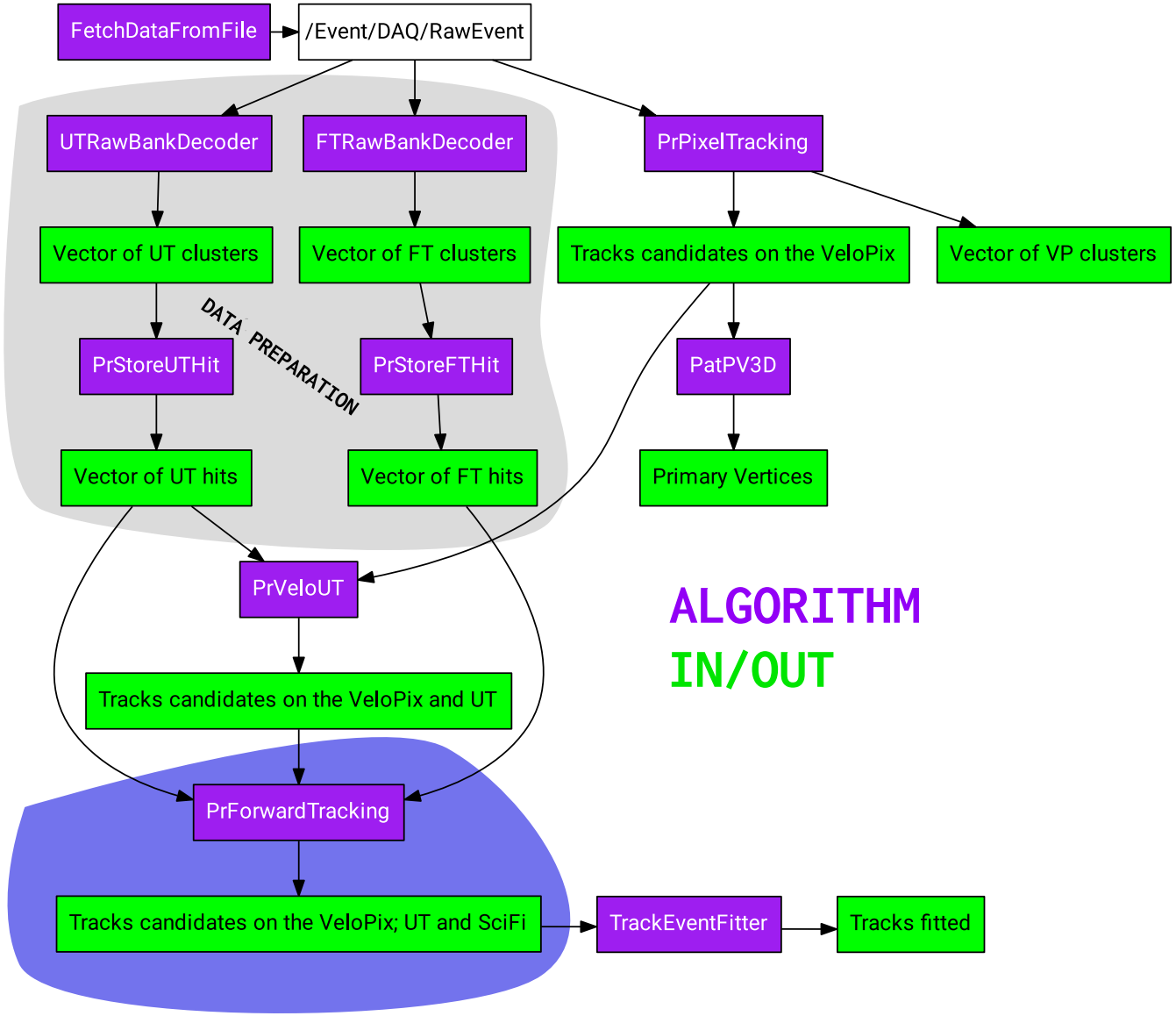
- Fitting and clustering are not independant
  - ↪ it runs concurrently
- Would like to change the algorithm and access to take advantage of the SOA memory layout
- Even if we have a SOA layout, possibility for (auto)vectorization will be difficult in that case
  - Hopes to be able to manually vectorize some parts

Results with the SOA layout

- Some parts vectorized and with good improvements
- But loose too much on some other parts, especially for parts with sort and merge
- It was also a good place to test Manuel's `SOAContainer`

Branch with the SOA layout: Rec/mhadji-prftracking-soalayout

Where are we on the HLT1 sequence?



Overview

Presentation of the last part of HLT1

Algo: TrackEventFitter

TrackEventFitter

- The final fitting part of pattern recognition tracks results using the Kalman Filter

TrackVectorFitter

- It is a tool used by the algorithm TrackEventFitter
- One of the alternatives to TrackMasterFitter

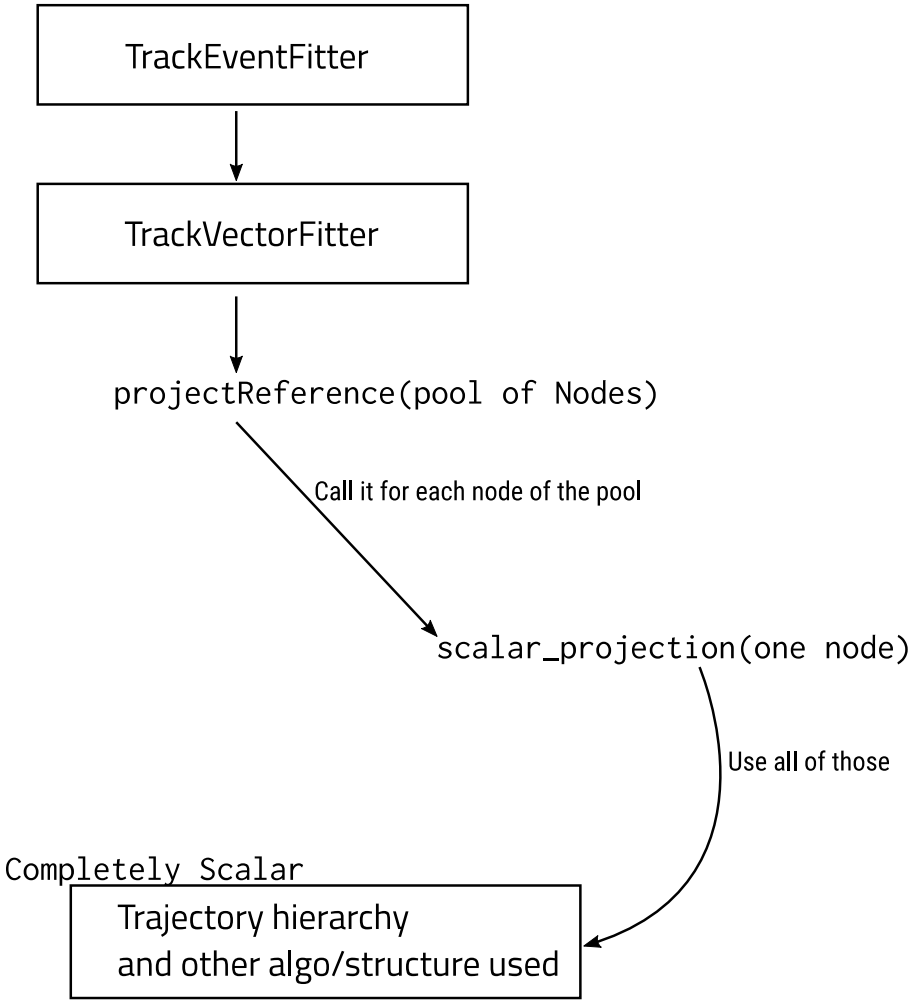
Initial state

with **intel® VTune™ Amplifier**

Callees	CPU Time: Total
TrackVectorFitter::operator()	100.0%
TrackVectorFitter::updateTransport	27.2%
for_each<__gnu_cxx::__normal_iterator<std::reference_wrapper<LHCb::Track>*, std::vector<std::re	26.8%
TrackVectorFitter::projectReference	13.0%
for_each<std::_List_iterator<std::reference_wrapper<Tr::TrackVectorFit::Track> >, TrackVectorFitter	12.2%
Tr::TrackVectorFit::TrackVectorFit::smoothFit<(bool)0>	7.4%
TrackVectorFitter::populateTracks	3.8%
Tr::TrackVectorFit::TrackVectorFit::smoothFit<(bool)1>	3.2%
Tr::TrackVectorFit::TrackVectorFit::initializeBasePointers<(bool)1>	1.4%
TrackVectorFitter::removeWorstOutlier	1.3%
Tr::TrackVectorFit::TrackVectorFit::initializeBasePointers<(bool)0>	1.3%
for_each<std::_List_iterator<std::reference_wrapper<Tr::TrackVectorFit::Track> >, TrackVectorFitter	0.7%
Tr::TrackVectorFit::TrackVectorFit::initializeBasePointers<(bool)1>	0.6%
for_each<std::_List_iterator<std::reference_wrapper<Tr::TrackVectorFit::Track> >, TrackVectorFitter	0.2%
for_each<std::_List_iterator<std::reference_wrapper<Tr::TrackVectorFit::Track> >, TrackVectorFitter	0.2%
LHCb::TrackFitResult::nOutliers	0.1%
std::__cxx11::list<std::reference_wrapper<Tr::TrackVectorFit::Track>, std::allocator<std::reference_	0.1%

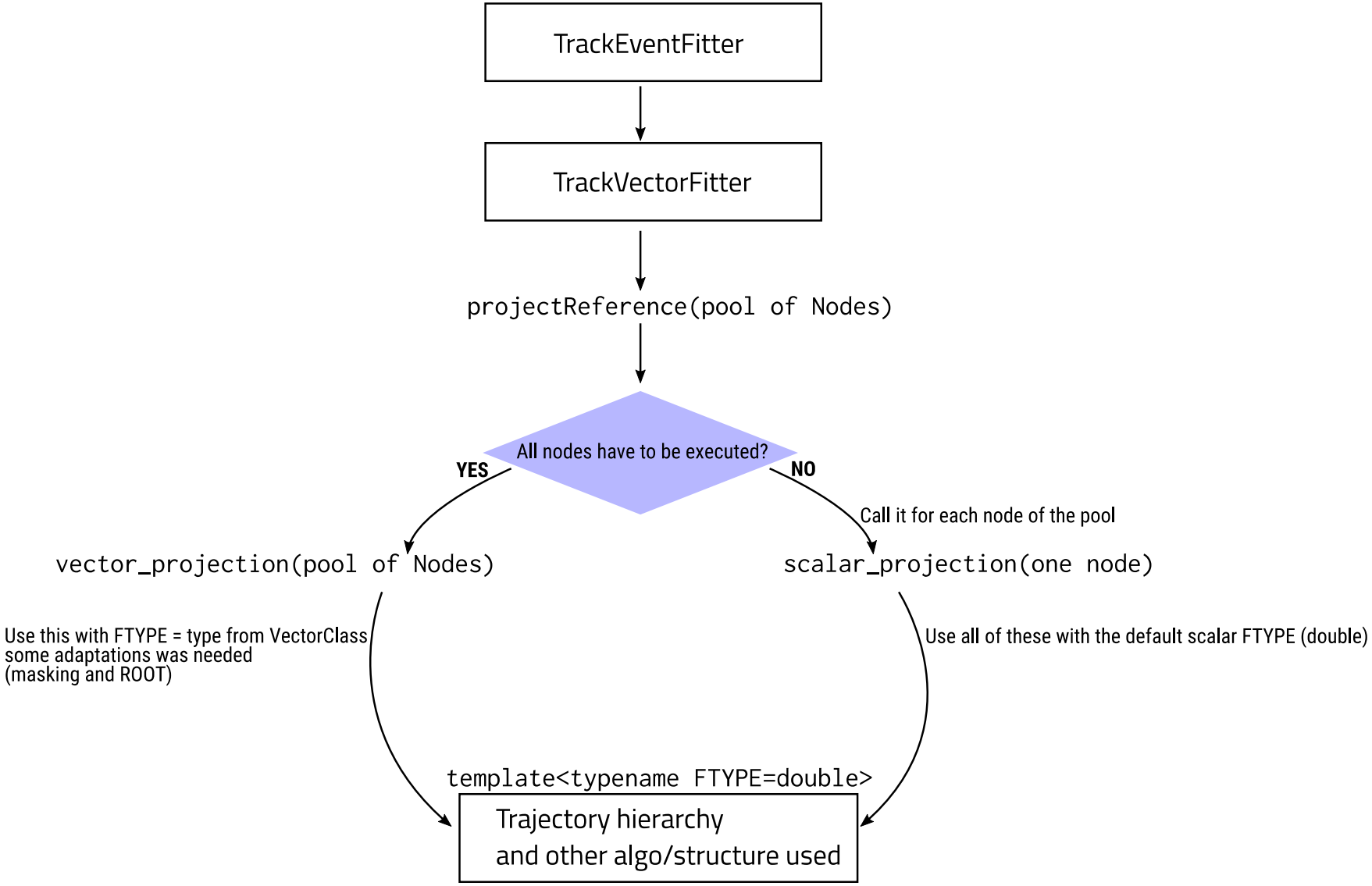
Initial structure

BEFORE



Global view of the vectorization

AFTER



Results with **callgrind**

BEFORE

hlt2.scalar.cg [python /home/mhadji/git/hackathon/Gaudi/InstallArea/x86_64+avx

File View Go Settings Help

Open Back Forward Up » Instruction Fetch

TrackVectorFitter::projectReference(std::...ectorFit::Sch::Blueprint<4ul> > >&)

Types | Callers | All Callers | Callee Map | Source Code

Event Type	Incl.	Self	Short		Formula
Instruction Fetch	1 539 244 930	61 054 262	Ir		
Scalar F32 Inst	0	0	IFP32x1		
Scalar F64 Inst	271 308 393	0	IFP64x1		
SIMDx2 F32 Inst	0	0	IFP32x2		
SIMDx2 F64 Inst	12 842 057	0	IFP64x2		
SIMDx4 F32 Inst	0	0	IFP32x4		
SIMDx4 F64 Inst	0	0	IFP64x4		
SIMDx8 F32 Inst	0	0	IFP32x8		
Cycle Estimation	1 539 244 930	61 054 262	CEst	=	Ir
F32 op	0	0	FP32	=	IFP32x1 + 2 IFP32x2 + 4 IFP32
F64 op	296 992 507	0	FP64	=	IFP64x1 + 2 IFP64x2 + 4 IFP64
SIMD 128b FP op	25 684 114	0	VFP128	=	4 IFP32x4 + 2 IFP64x2
SIMD 256b FP op	0	0	VFP256	=	8 IFP32x8 + 4 IFP64x4
SIMD FP op	25 684 114	0	VFP	=	2 IFP32x2 + VFP128 + VFP256
Scalar FP op	271 308 393	0	SFP	=	IFP32x1 + IFP64x1

AFTER

hlt2.vector.cg [python /home/mhadji/git/hackathon/Gaudi/InstallArea/x86_64+av

File View Go Settings Help

Open Back Forward Up » Instruction Fetch

TrackVectorFitter::projectReference(std::...ectorFit::Sch::Blueprint<4ul> > >&)

Types | Callers | All Callers | Callee Map | Source Code

Event Type	Incl.	Self	Short		Formula
Instruction Fetch	675 357 350	37 920 068	Ir		
Scalar F32 Inst	0	0	IFP32x1		
Scalar F64 Inst	44 364 780	0	IFP64x1		
SIMDx2 F32 Inst	0	0	IFP32x2		
SIMDx2 F64 Inst	1 422 040	0	IFP64x2		
SIMDx4 F32 Inst	0	0	IFP32x4		
SIMDx4 F64 Inst	65 524 905	0	IFP64x4		
SIMDx8 F32 Inst	13 606 230	0	IFP32x8		
Cycle Estimation	675 357 350	37 920 068	CEst	=	Ir
F32 op	108 849 840	0	FP32	=	IFP32x1 + 2 IFP32x2 + 4 IFP32x
F64 op	309 308 480	0	FP64	=	IFP64x1 + 2 IFP64x2 + 4 IFP64x
SIMD 128b FP op	2 844 080	0	VFP128	=	4 IFP32x4 + 2 IFP64x2
SIMD 256b FP op	370 949 460	0	VFP256	=	8 IFP32x8 + 4 IFP64x4
SIMD FP op	373 793 540	0	VFP	=	2 IFP32x2 + VFP128 + VFP256
Scalar FP op	44 364 780	0	SFP	=	IFP32x1 + IFP64x1

Nb instructions divided by 2

total amount of fp op is bigger compared to before because masking

Results

with **intel® VTune™ Amplifier**

BEFORE

Callees	CPU Time: Total
TrackVectorFitter::operator()	100.0%
TrackVectorFitter::updateTransp	27.2%
for_each<__gnu_cxx::__normal_	26.8%
TrackVectorFitter::projectReferen	13.0%
for_each<std::_List_iterator<std:	12.2%
operator()	12.2%
TrackVectorFitter::updateMa	12.1%
MaterialLocatorBase::interse	0.1%
Tr::TrackVectorFit::TrackVectorF	7.4%
TrackVectorFitter::populateTrack	3.8%
Tr::TrackVectorFit::TrackVectorF	3.2%
Tr::TrackVectorFit::TrackVectorF	1.4%
TrackVectorFitter::removeWorst	1.3%
Tr::TrackVectorFit::TrackVectorF	1.3%

AFTER

Callees	CPU Time: Total
TrackVectorFitter::operator()	100.0%
TrackVectorFitter::updateTransport	30.3%
for_each<__gnu_cxx::__normal_itera	29.8%
for_each<std::_List_iterator<std::refe	13.6%
operator()	13.6%
TrackVectorFitter::updateMateri	13.6%
Tr::TrackVectorFit::TrackVectorFit::s	8.2%
Tr::TrackVectorFit::TrackVectorFit::s	3.8%
TrackVectorFitter::populateTracks	3.4%
TrackVectorFitter::projectReference	3.3%
TrackProjector::projectReference	1.6%
LHCb::Measurement::type	0.7%
TrackProjectorSelector::projector	0.2%
Tr::TrackVectorFit::Vector::TrackP	0.2%

Main related MR : Rec!869

makeNodes

BEFORE

Callees	CPU Time: Total
TrackVectorFitter::operator()	100.0%
TrackVectorFitter::updateTransp	27.2%
for_each<__gnu_cxx::__normal_	26.8%
TrackVectorFitter::projectReferei	13.0%
for_each<std::_List_iterator<std:	12.2%
operator()	12.2%
TrackVectorFitter::updateMa	12.1%
MaterialLocatorBase::interse	0.1%
Tr::TrackVectorFit::TrackVectorF	7.4%
TrackVectorFitter::populateTrack	3.8%
Tr::TrackVectorFit::TrackVectorF	3.2%
Tr::TrackVectorFit::TrackVectorF	1.4%
TrackVectorFitter::removeWorst	1.3%
Tr::TrackVectorFit::TrackVectorF	1.3%

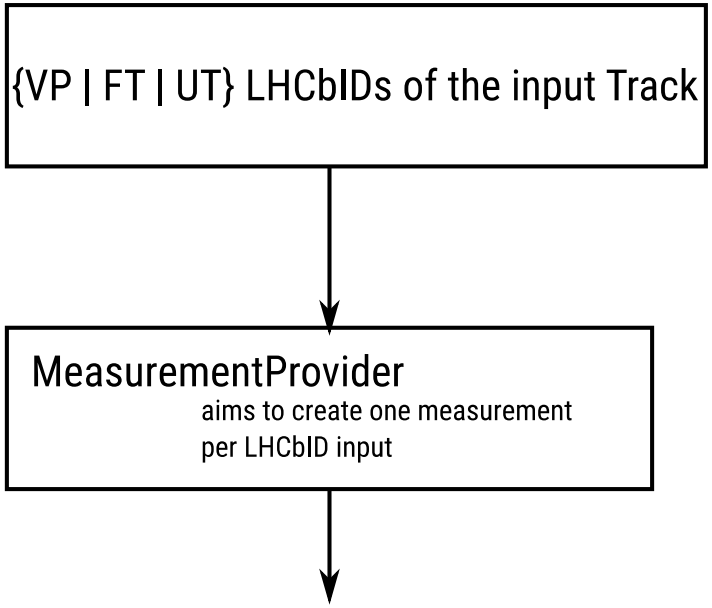
AFTER

Callees	CPU Time: Total
TrackVectorFitter::operator()	100.0%
TrackVectorFitter::updateTransport	30.3%
for_each<__gnu_cxx::__normal_itera	29.8%
for_each<std::_List_iterator<std::refe	13.6%
operator()	13.6%
TrackVectorFitter::updateMateri	13.6%
Tr::TrackVectorFit::TrackVectorFit::s	8.2%
Tr::TrackVectorFit::TrackVectorFit::s	3.8%
TrackVectorFitter::populateTracks	3.4%
TrackVectorFitter::projectReference	3.3%
TrackProjector::projectReference	1.6%
LHCb::Measurement::type	0.7%
TrackProjectorSelector::projector	0.2%
Tr::TrackVectorFit::Vector::TrackP	0.2%

makeNodes : creation of measurements

Overview

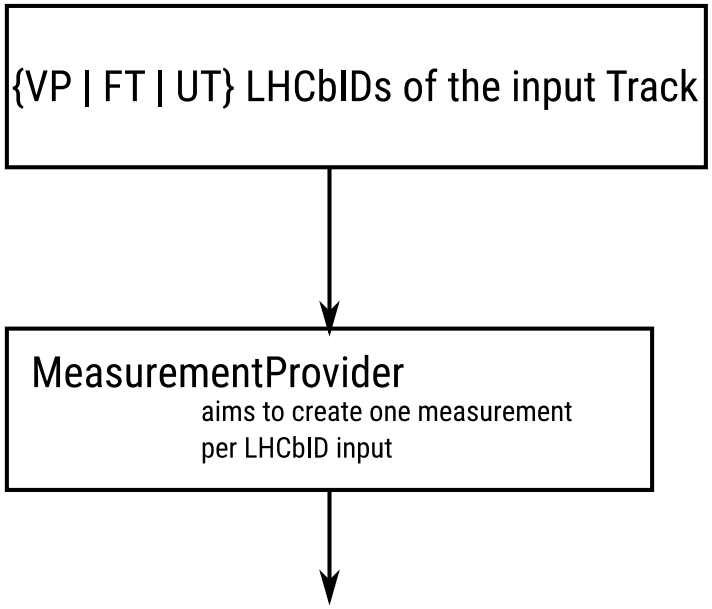
Current




```
for each LHCbID of the input track:
    search the cluster (from the TES) corresponding to the current LHCbID;
    new Measurement(cluster);
```

where do we get lhcbids?
at the end of pixeltracking, velout, forwardtracking: there is the final step of track creations
currently: "`track.lhcbids.insert(cluster.lhcbid())`" for each hits which have to be inserted on the track

Current



ordering of clusters is optimized for PrHits creation,
order not adapted for a `lower_bound`
-> it's currently a linear `find_if`

```
for each LHCbID of the input track:
    search the cluster (from the TES) corresponding to the current LHCbID;
    new Measurement(cluster);
```

for each lhcbid

logn search on ut and vp clusters
linear search on ft clusters

What I would like to try

{VP | FT | UT} ~~LHCbIDs~~ of the input Track Clusters

try to give directly clusters as input in order to get rid of the linear search, at least for FT and UT

FT and UT clusters are uint like LHCbIDs -> memory layout is the same
LHCbID info is included on the cluster -> backward compatible
if LHCbID needed somewhere else

MeasurementProvider
aims to create one measurement per ~~LHCbID~~ input cluster

~~ordering of clusters is optimized for PrHits creation, order not adapted for a `lower_bound` -> it's currently a linear `find_if`~~

```
for each LHCbID of the input track:
    search the cluster (from the TES) corresponding to the current LHCbID;
    new Measurement(cluster);
```

where do we get clusters instead of lhcbids?
at the end of pixeltracking, velout, forwardtracking:
"`track.~~lhcbids~~.insert(cluster~~.lhcbid()~~)`" for each hits which have to be inserted on the track

What I would like to try

{VP | FT | UT} ~~LHCbIDs~~ of the input Track Clusters

try to give directly clusters as input in order to get rid of the linear search, at least for FT and UT

FT and UT clusters are uint like LHCbIDs -> memory layout is the same
LHCbID info is included on the cluster -> backward compatible
if LHCbID needed somewhere else

MeasurementProvider
aims to create one measurement per ~~LHCbID~~ input cluster

~~ordering of clusters is optimized for PrHits creation,~~
~~order not adapted for a `lower_bound`~~
~~-> it's currently a linear `find_if`~~

```
for each LHCbID of the input track:
    search the cluster (from the TES) corresponding to the current LHCbID;
    new Measurement(cluster);
```

Avoid these allocations using boost::variant as suggested by Sebastien and Gerhard

Overview

What I have done

- Help to improve the hlt1 sequence
- Improved my english
- Learnt a lot of git stuff, thanks Sebastien and Rosen!
- Learnt a lot of c++ stuff, thanks to a lot of people!
- Learnt a lot of other stuff, thanks to everybody!
- Learnt tools for perf and wrote **a tutorial for measuring performance** given at some hackathon and computing workshop

What is next

September 2018

I am going to do my 2nd year of master at Université de Lyon with a specialization on digital geometry, computer graphics and GPU programming

The most important slide

Big thanks to everybody for this incredible experience

It was **really** amazing

A **huge** thank you to Sébastien
for having taught me **a lot**