

time_measurement_audit

Analysis of time measurement in `algo`. Prepared with assistance of OpenCode agent with `DeepSeek-V4.1-Flash` model. Not every line was verified by human, so take it with a grain of salt.

Time measurement audit of the `algo` subdirectory

Scope: all time measurements in `algo/` based on the two APIs

1. xpu timers: `xpu::scoped_timer`, `xpu::push_timer`, `xpu::pop_timer` (plus `xpu::t_add_bytes` for throughput), and
2. Monitor classes (`StartTimer`, `StopTimer`, `IncrementCounter`), i.e. `cbm::algo::ca::Monitor` / `MonitorData` specializations (`ca::TrackingMonitor`, `evreco::Monitor`, `evselect::Monitor`, `kfp::V0FinderMonitor`).

The audit lists every usage and collects observations on measurement contamination (QA, monitoring, copies), truncated/over-extended measurement ranges, unmeasured sections and other anomalies.

An appendix covers ad-hoc `std::chrono` / `ca::Timer` usages and the `online/device/CbmRecoDevice.cxx` case (outside `algo`, included on request).

1. API semantics (what the numbers actually mean)

xpu timers (`external/xpu/src/xpu/host.h`, `external/xpu/src/xpu/detail/timers.cpp`)

- `xpu::push_timer(name)` / `xpu::pop_timer()` implement a per-thread stack (`thread_local`): the popped interval is attached as a child of the enclosing timer. `pop_timer()` throws if the stack is empty.
- Nested timers are *not* subtracted from the parent: parent wall time contains the child time (by design).
- `xpu::scoped_timer` is the RAI wrapper; it can store the result in an `xpu::timings*`.
- `xpu::t_add_bytes(n)` adds bytes to the **innermost** timer only (`timers.cpp:80-82`). Throughput of a parent therefore does not include nested byte counts unless bytes are added on the parent directly.
- `add_kernel_time` / `add_memcpy_time` / `add_memset_time` add to **all** timers on the thread's stack, so kernel/copy details are visible in every enclosing timer (requires profiling).
- Because the stack is thread-local, push/pop must happen on one thread; an exception or early `return` between push and pop leaves a stale frame that pollutes later nesting on that thread.

Monitor timers (`algo/ca/core/utils/CaTimer.h`, `CaMonitorData.h`, `CaMonitor.h`)

- `StartTimer(key)` stores `Clock::now()`; `StopTimer(key)` accumulates the interval (`fTotal`, `fNofCalls`, min/max). There is no stack: `Start` on an already-running timer silently restarts it; a missing `Stop` loses the interval.
- `MonitorData::AddMonitorData(other, parallel)` and `Timer::AddTimer(other, parallel)` sum totals when `parallel == false` and take the **maximum** total when `parallel == true` (approximation for multi-thread merging).
- `Reset()` clears all counters/timers; several algorithms call it per event/timeslice, so the monitor contents are only meaningful together with how often they are merged/reset.
- Bug in `Timer::AddTimer` (`CaTimer.h:126-129`): when adopting `other`'s maximum, `fMaxCallIndex` is computed from `other.fMinCallIndex` instead of `other.fMaxCallIndex`.

2. Inventory: xpu timer usages

2.1 Always-active xpu timers

File:lines	Timer name	Measured region	Notes
unpack/CommonUnpacker.h:106	CommonUnpacker.<subsystem> (scoped)	whole DoUnpack : MS descriptor scan, per-MS unpack, resize, merge, sort	includes intermed allocation/merge
unpack/CommonUnpacker.h:125,151	Unpack (push/pop)	OpenMP loop over microslices calling per-MS algorithms	result discarded; i region
unpack/CommonUnpacker.h:153,158	CommonUnpacker.<sys>.Resize	output vector resize and digi count	result discarded
unpack/CommonUnpacker.h:160,168	CommonUnpacker.<sys>.Merge	parallel per-MS digi copy into output	t_add_bytes at 11 inside
unpack/CommonUnpacker.h:170,173	CommonUnpacker.<sys>.Sort	DoSort (time sort)	t_add_bytes at 11
evbuild/EventBuilder.cxx:22,55	EventBuilder	event windowing/build over all triggers, event selector, monitor counters	monitor.time set BuildEvent digi c excludes return
trigger/HitMultTrigger.cxx:24,32	HitMultTrigger	GetHitTimes (copy of hit time stamps) + trigger algorithm	xpu::t_add_bytes includes input ext
trigger/TimeClusterTrigger.cxx:27,66	TimeClusterTrigger	clustering loop + monitor counters	sorted-input chec (good); used both triggers
trigger/V0Trigger.cxx:21,73	V0Trigger	full track-pair loop incl. QA histogram fills and PCA	QA fills at 38-40, timer
detectors/bmon/Calibrate.cxx:61,148	bmon::Calibrate	calibration loop + insert-sort of output	monitor.fTime
detectors/tof/Calibrate.cxx:40,132	tof::Calibrate	calibration loop + insert-sort of output	monitor.fTime
detectors/tof/Hitfind.cxx:91,106	tof::Hitfind.ChanSort	distributing digis per RPC	monitor.fSortTim
detectors/tof/Hitfind.cxx:115,177	tof::Hitfind.FindHit	OpenMP per-RPC clustering + flattening	monitor.fTime ; i construction at 18
detectors/trd/Hitfind.cxx:155,178 (active RunRowParallel)	trd::Hitfind.DigiModuleSort	digi bucketing per module/row	popped, then deli (monitor.sortTim
detectors/trd/Hitfind.cxx:191,304	trd::Hitfind.BuildHits	OpenMP clusterizing, hit building, row merging, buffer flattening	hitsOut construc time sort (312-316
detectors/trd/Hitfind.cxx:349,372 (only MERGE_BY_MODULE)	trd::Hitfind.DigiModuleSort	same bucketing	kept here (not zer
detectors/trd/Hitfind.cxx:374,436 (only MERGE_BY_MODULE)	trd::Hitfind.BuildClusters	clusterizing + hit finding	monitor.timeClus
detectors/trd/Hitfind.cxx:448,527 (only MERGE_BY_MODULE)	trd::Hitfind.FindHits	module merge + flatten	output constructic
detectors/sts/HitfinderChain.cxx:229	sts::HitfinderChain.SortHits (scoped)	SplitHitsIntoStreams + SortHitsWithinPartition (100 streams)	post-processing;
detectors/sts/HitfinderChain.cxx:367	sts::HitfinderChain.Allocate (scoped)	AllocateDynamic buffer allocation	allocation measur
detectors/sts/HitfinderChain.cxx:423	sts::HitfinderChain.CountDigisByModule	digi counting	t_add_bytes at 4
detectors/sts/HitfinderChain.cxx:498	sts::HitfinderChain.FlattenDigis	digi flattening/copy to flat buffer	t_add_bytes at 4

File:lines	Timer name	Measured region	Notes
<code>detectors/sts/HitfinderChain.cxx:586</code>	<code>sts::HitfinderChain.FlattenHits</code> (scoped)	CPU backend only (<code>if</code> at 582) copy of hit buckets	GPU branch (595-timer)
<code>detectors/sts/HitfinderChainCpu.cxx:93</code>	<code>sts::HitfinderChainCpu.SortHits</code>	streams split + sort	
<code>detectors/sts/HitfinderChainCpu.cxx:199</code>	<code>sts::HitfinderChainCpu.Allocate</code>	dynamic allocation	
<code>detectors/sts/HitfinderChainCpu.cxx:274</code>	<code>sts::HitfinderChainCpu.CountDigisByModule</code>	counting	<code>t_add_bytes</code> at 2
<code>detectors/sts/HitfinderChainCpu.cxx:377</code>	<code>sts::HitfinderChainCpu.FlattenDigis</code>	flatten/copy	<code>t_add_bytes</code> at 3
<code>detectors/sts/HitfinderChainCpu.cxx:497</code>	<code>sts::HitfinderChainCpu.FlattenHits</code>	flatten hit buckets	ends before offset 516)
<code>ca/TrackingChain.cxx:114</code>	<code>TrackingChain.CA</code> (scoped)	whole <code>Run</code> : input prep, <code>FindTracks</code> , output prep, QA exec, debug monitor dump (<code>kDEBUG</code> 245-255), monitor merge (257-258)	name/comment not forwarded to monitor
<code>global/Reco.cxx:583,837</code>	<code>Reco</code> (scoped)	whole timeslice processing	includes output file and <code>fQaManager-></code>
<code>global/Reco.cxx:593,631</code>	<code>Reco.Unpack</code> (scoped)	all <code>RunUnpacker</code> calls	includes <code>QueueUnpacker</code> digi QA execs (62)
<code>global/Reco.cxx:636,646</code>	<code>Reco.STS</code> (scoped)	STS hit finder	includes <code>QueueStsHitFinderQa-</code>
<code>global/Reco.cxx:651,662</code>	<code>Reco.TOF</code> (scoped)	TOF calibration + hit finder	includes <code>QueueTofCalibMet</code> (658-659) and TC
<code>global/Reco.cxx:667,679</code>	<code>Reco.TRD</code> (scoped)	TRD hit finder	includes <code>allDigi-</code> and <code>QueueTrdReco</code>
<code>global/Reco.cxx:689,707</code>	<code>Reco.CA</code> (scoped)	CA tracking chain	includes track <code>st-</code> <code>QueueTrackingMet</code> (inside <code>TrackingC</code>
<code>global/Reco.cxx:728,797</code>	<code>Reco.EventReco</code> (scoped)	event reconstruction + KFP selector + evselect/KFP monitor updates	KFP selector QA i

2.2 Compile-time-disabled GPU timers

`constants::gpu::GpuTimeMonitoring` and `constants::gpu::GpuTimeMonitoring_tracking` are both hard-coded `false` (`ca/core/pars/CaDefs.h:134,140`). Consequently all of the following code is dead in current builds:

File	Timer names (push/pop lines)
<code>ca/core/dev/CaGpuTimeMonitor.h</code>	<code>XpuTimings</code> struct with 18 arrays of <code>xpu::timings</code> ; <code>GpuTimeMonitor</code> class is not used anywhere
<code>ca/core/tracking/CaTrackFinder.cxx:577,582</code>	<code>ca::TrackFinder.gpuTFinIt</code> (single-window path; popped and logged immediately)
<code>ca/core/tracking/CaTrackFinderWindow.cxx</code> (both <code>CaTrackFinderSliceXPU</code> 213-361 and <code>CaTrackFinderSliceBatchXPU</code> 373-512)	<code>DataPreparationTime</code> , <code>CopyDataToGpuTime</code> , <code>SetupGridTime</code> , <code>SetupIterationDataTime</code> , <code>RunGpuTrackFittingAndMerging</code> , <code>CopyFittedTracksToCpu</code> , <code>CleanupGpuDataTime</code> , <code>TotalFin-</code>
<code>ca/core/tracking/CaTrackFinderWindow.cxx</code> (helpers)	<code>SetupParametersTime</code> / <code>SetInputDataTime</code> / <code>SetupMaterialMapTime</code> (1158-1175), <code>SetupItera-</code>
<code>ca/core/dev/CaGpuTrackFinderSetup.cxx</code>	<code>SetupGrid*</code> (267-487), <code>SetInputData*</code> (635-730), <code>Fitting1</code> (900-909), <code>Full</code> / <code>PrepareData</code> / <code>MakeSinglets</code> / <code>MakeDoublets</code> / <code>ResetDoublets</code> / <code>CompressDoublets</code> / <code>Fi-</code> <code>Full</code> / <code>PrepareData</code> / <code>SearchNeighbors</code> / <code>SearchNeighbors2</code> / <code>PrepareTreeRoots_time</code> / <code>Compr-</code> (1389-1643)
<code>ca/core/dev/CaGpuMultiWindowTrackingDev.cxx</code>	local <code>ca::Timer</code> s only (see Appendix A)

3. Inventory: Monitor (**StartTimer** / **StopTimer** / **IncrementCounter**) usages

Infrastructure definitions: `ca/core/utils/CaMonitorData.h:87,92,103,107` (**IncrementCounter** , **StartTimer** , **StopTimer**),
`ca/core/utils/CaMonitor.h:112,117,144,148` (monitor facade), `ca/core/utils/CaTimer.h` (implementation).

Monitor specializations: `ca/core/utils/CaTrackingMonitor.h` , `global/EventReconstructionMonitor.h` ,
`evselector/RecoEventSelectorMonitor.h` , `kfp/KfpV0FinderMonitor.h` .

3.1 **ca::TrackingMonitor** (`ca/core/utils/CaTrackingMonitor.h` , used in **TrackingChain** + **CaTrackFinder** + **CaTrackFinderWindow**)

Location	Timer keys	Measured region	Notes
<code>ca/TrackingChain.cxx:115-116</code>	TrackingChain	whole Run	starts after fCaMonitorData.Re Stop at 243
<code>ca/TrackingChain.cxx:119-121</code>	PrepareInputData	PrepareInput	includes ReadHits ReceiveInputData
<code>ca/TrackingChain.cxx:155-157</code>	PrepareStsHits	ReadHits<kSts>	
<code>ca/TrackingChain.cxx:160-162</code>	PrepareTrdHits	ReadHits<kTrd>	
<code>ca/TrackingChain.cxx:165-167</code>	PrepareTofHits	ReadHits<kTof>	
<code>ca/TrackingChain.cxx:172-174</code>	InputDataTransmission	ReceiveInputData (move of CA input data)	
<code>ca/TrackingChain.cxx:309-361</code>	CaHitCreation	per-hit loop body in ReadHits (ca::Hit fill, check, PushBackHit , index bookkeeping)	started/stopped in: per-hit loop
<code>ca/TrackingChain.cxx:235-240</code>	Qa	QA registration and Qa.Exec()	nested inside TrackingChain
<code>ca/TrackingChain.cxx:218-220</code>	counters RecoStsHit/RecoTofHit/RecoTrdHit	per-track counters in output prep	after TrackingChai No, inside (stop at
<code>ca/TrackingChain.cxx:355-359</code>	counters Undefined*Hit	rejected hits	
<code>ca/core/tracking/CaTrackFinder.cxx:141-144</code>	Tracking , PrepareTimeslice , counters TrackingCall , RecoHit	TrackFinder::FindTracks	Stop Tracking at Stop PrepareTimes 347
<code>ca/core/tracking/CaTrackFinder.cxx:354-357 / 370-380</code>	StoreTracksFinal	moving+concatenating per-thread tracks (multi- thread path includes join and merge)	
<code>ca/core/tracking/CaTrackFinder.cxx:383-384</code>	counters RecoTrack , RecoHitUsed		
<code>ca/core/tracking/CaTrackFinder.cxx:424-479</code>	TrackingThread , PrepareThread	per-thread work (GPU multi-window path stops both at 479)	PrepareThread stc 469 in CPU path
<code>ca/core/tracking/CaTrackFinder.cxx:476-478</code>	TrackingWindow	RunPreparedWindows (multi-window GPU)	includes SuppressNeighbour and output append
<code>ca/core/tracking/CaTrackFinder.cxx:500-571</code>	PrepareWindow	per-sub-timeslice hit selection	
<code>ca/core/tracking/CaTrackFinder.cxx:586-588 / 591-593</code>	TrackingWindow	GPU single-window / CPU tracking in window	
<code>ca/core/tracking/CaTrackFinder.cxx:607-634</code>	StoreTracksWindow	moving window tracks to thread output, hit-used flags	
<code>ca/core/tracking/CaTrackFinder.cxx:639</code>	TrackingThread stop		

Location	Timer keys	Measured region	Notes
ca/core/tracking/CaTrackFinder.cxx:390-394	AddMonitorData(monitor, true)	merge per-thread monitors (parallel=true → max totals), then reset	
ca/core/tracking/CaTrackFinder.cxx:490	counter SubTS	per sub-timeslice	
ca/core/tracking/CaTrackFinderWindow.cxx:129-131	InitWindow	ReadWindowData	CPU slice
ca/core/tracking/CaTrackFinderWindow.cxx:134-136	PrepareGrid	PrepareGrid	CPU slice
ca/core/tracking/CaTrackFinderWindow.cxx:146-183	FindTracks	iterations loop	stopped before fitti
ca/core/tracking/CaTrackFinderWindow.cxx:153-155	PrepareIteration	per-iteration prep	
ca/core/tracking/CaTrackFinderWindow.cxx:158-160	ConstructTriplets	per-iteration triplet construction	
ca/core/tracking/CaTrackFinderWindow.cxx:163-165	SearchNeighbours	per-iteration neighbour search	
ca/core/tracking/CaTrackFinderWindow.cxx:168-170	CreateTracks	per-iteration track candidates	
ca/core/tracking/CaTrackFinderWindow.cxx:173-181	SuppressHitKeys	hit-key suppression	
ca/core/tracking/CaTrackFinderWindow.cxx:186-188	FitTracks	track fitter	
ca/core/tracking/CaTrackFinderWindow.cxx:192-198	MergeClones , FitTracks	clone merging + refit	
ca/core/tracking/CaTrackFinderWindow.cxx:203-205	GlobalTrackExtension	global mode only	
ca/core/tracking/CaTrackFinderWindow.cxx:217-224	InitWindow , PrepareGrid	GPU single-window prep	
ca/core/tracking/CaTrackFinderWindow.cxx:257-365	FindTracks	GPU single-window: umbrella over iterations + fit/merge + D2H copy + cleanup	semantic mismatch CPU path
ca/core/tracking/CaTrackFinderWindow.cxx:263-276	PrepareIteration		
ca/core/tracking/CaTrackFinderWindow.cxx:279-290	ConstructTriplets	GPU triplet construction	
ca/core/tracking/CaTrackFinderWindow.cxx:322-324	PrepareGrid	RemoveUsedHitsFromGrid	StartTimer called (322 and 324), never stopped (see 4.4)
ca/core/tracking/CaTrackFinderWindow.cxx:337-340	FitTracks	GPU fitting/merging	
ca/core/tracking/CaTrackFinderWindow.cxx:348-350	CreateTracks	CopyFittedTracksXPU_d2h (D2H copy)	key reused for a di operation than CPL
ca/core/tracking/CaTrackFinderWindow.cxx:375-377	InitWindow	batch path	
ca/core/tracking/CaTrackFinderWindow.cxx:382-387	PrepareGrid	batch grid prep	
ca/core/tracking/CaTrackFinderWindow.cxx:423-516	FindTracks	batch umbrella	
ca/core/tracking/CaTrackFinderWindow.cxx:432-444	PrepareIteration		
ca/core/tracking/CaTrackFinderWindow.cxx:446-457	ConstructTriplets		
ca/core/tracking/CaTrackFinderWindow.cxx:472-474	PrepareGrid	RemoveUsedHitsFromGrid	correctly stopped (with single-window

Location	Timer keys	Measured region	Notes
ca/core/tracking/CaTrackFinderWindow.cxx:486-489	FitTracks		
ca/core/tracking/CaTrackFinderWindow.cxx:499-501	CreateTracks	D2H copy	
ca/core/tracking/CaTrackFinderWindow.cxx:766	counter Triplet	per-station triplet count	
ca/core/dev/CaGpuMultiWindowTrackingDev.cxx:204	counter SubTS	number of windows built	

3.2 evreco::Monitor (algo/global/EventReconstruction.cxx)

Location	Timer keys	Measured region	Notes
EventReconstruction.cxx:181 / 276	EventReconstruction	whole Exec	Stop after counters, includes PID update (263-265)
EventReconstruction.cxx:204-206	StsHitFinder	(*fStsHitFinder)(...)	step timer stopped before QA (210)
EventReconstruction.cxx:215-218	TofHitFinder	TOF calibrate + hit find	QA at 221 is outside
EventReconstruction.cxx:227-235	TrdHitFinder	TRD digi concatenation + hit find	copy at 230-233 inside
EventReconstruction.cxx:241-250	TrackFinder	fTracking->Run + result moves	
EventReconstruction.cxx:255-259	PvFinder	PV finder	QA at 260 outside
EventReconstruction.cxx:267-274	counters	EventsTotal , hits, tracks, PV counters	inside EventReconstruction
EventReconstruction.cxx:186-200	BmonHitFinder	commented-out BMON block	timer key exists but is never started

3.3 evselect::Monitor (algo/global/Reco.cxx)

Location	Timer keys	Measured region	Notes
Reco.cxx:738 / 759	EventProcessing	whole event loop (OpenMP parallel)	includes KfpSelection on thread 0
Reco.cxx:752 / 756	KfpSelection	fKfpSelector[iThread]->ProcessEvent for iThread == 0 only	comment at 751: counts only thread 0 and QA is included
Reco.cxx:767-770	counters EventsPv , PrimaryVertices	summed from per-thread EventReconstruction monitors	performed after the parallel loop
Reco.cxx:772-785	AddTimer(..., parallel=true) for EventReconstruction , BmonReco , StsReco , TofReco , TrdReco , Tracking , PvFinding	merges per-thread event-reco step timers (max total)	BmonReco is always zero (BMON disabled)
Reco.cxx:790-793	QueueEvselectMetrics , accumulate, ToString , Reset		

3.4 kfp::V0FinderMonitor (algo/kfp/KfpV0Finder.cxx)

Location	Timer keys	Measured region	Notes
KfpV0Finder.cxx:271-272 / 308	ProcessEvent	whole event	reset at 271 every event; early return at 286 skips the stop
KfpV0Finder.cxx:282 / 288	CollectT0	filling T0 container from BMON hits	not stopped on the 284-287 early return
KfpV0Finder.cxx:292-294	CollectDca	DCA estimation + PID assignment	
KfpV0Finder.cxx:298 / 307	FindV0Candidates	loop over T0 candidates (can call the body several times per event)	

Location	Timer keys	Measured region	Notes
<code>KfpV0Finder.cxx:160-199</code>	<code>PreselectTracks</code>	track preselection inside <code>FindV0Candidates</code>	nested
<code>KfpV0Finder.cxx:209-224</code>	<code>InitKfp</code>	KFParticleFinder input construction	nested
<code>KfpV0Finder.cxx:225-227</code>	<code>ExecKfp</code>	<code>ReconstructParticles</code>	nested
<code>KfpV0Finder.cxx:28,42,46,61,68,75,81,93,150-154,175,203-206,233</code>	counters	track/PID/event counters	some reset per T0 candidate (150-154)
<code>KfpV0FinderChain.cxx:140</code>	<code>AddMonitorData</code> per event	merges per-event monitor data	<code>Process</code> resets its monitor each event
<code>KfpV0FinderMonitor.h:51,93</code>	<code>PrepareContainers</code>	declared and named, never used	dead key

3.5 Monitor infrastructure — `AddTimer` semantics

`Reco.cxx:772-785` merges per-thread monitors with `parallel=true` (max of totals, sum of call counts), while `CaTrackFinder.cxx:390-394` merges per-thread CA monitors with `parallel=true` as well. This is a wall-clock approximation: totals are maxima, but `GetAverage()` = `fTotal/fNofCalls` mixes a max total with a summed call count and is therefore not a true average for multi-threaded runs.

4. Findings

4.1 Measurements that include things unrelated to the measured algorithm

- Reco.STS includes STS QA and monitoring** — `global/Reco.cxx:636-646` : `QueueStsRecoMetrics(stsHitfinderMonitor)` (643) and `fStsHitFinderQa->Exec(...)` (645) run inside the `xpu::scoped_timer`. The timer measures hit finding plus QA histogram filling plus metric queuing.
- Reco.T0F includes QA and metric queueing** — `global/Reco.cxx:651-662` : `QueueTofCalibMetrics` (658), `QueueTofRecoMetrics` (659), `fTofHitFinderQa->Exec` (661) are inside the timer.
- Reco.TRD includes QA, metric queueing and a digi copy** — `global/Reco.cxx:667-679` : the 1D+2D digi concatenation (670-675, a deliberate extra copy, cf. FIXME) and `QueueTrdRecoMetrics` (678), while there is no TRD QA in the ts path, are inside the timer.
- Reco.Unpack includes digi QA and metric queueing** — `global/Reco.cxx:592-631` : `RunUnpacker` itself calls `QueueUnpackerMetricsDet` (`Reco.cxx:920-921`), and `fMustDigiQa->Exec` / `fStsDigiQa->Exec` are called at 628-630, all inside the `Reco.Unpack` scope.
- Reco.CA includes sorting, metric queueing and CA QA** — `global/Reco.cxx:688-707` : `std::sort` of the tracks (a V0-trigger-only requirement, 702-705) and `QueueTrackingMetrics` (706) are inside; the CA QA (`TrackingChain::PrepareOutput`, `TrackingChain.cxx:234-241`) is nested inside as well.
- Outer Reco timer includes QA histogram shipping and output filtering** — `global/Reco.cxx:583-836` : `fQaManager->SendHistograms()` (834), per-subsystem output moves (800-827), logging and `DigiData` / `AuxDigiData` construction are all inside `Reco`.
- TrackingChain.CA is an umbrella including QA and debug I/O** — `ca/TrackingChain.cxx:114-129` : `PrepareOutput` (called at 129) contains the QA block (234-241), and — after `fCaMonitorData.StopTimer` — the `kDEBUG` per-timeslice monitor file dump (`TrackingChain.cxx:245-255`) and `fCaMonitor.AddMonitorData` a. still run under the xpu timer.
- V0Trigger includes QA histogram filling in the hot loop** — `trigger/V0Trigger.cxx:38-40,52-55,63-67` ; the same applies in spirit to the trigger selection loops.
- KfpSelection includes the KFP selector QA** — `global/Reco.cxx:752-756` around `KfpSelectorChain::ProcessEvent`, which calls `fQa->Exec(...)` (`KfpSelectorChain.cxx:62-68`). The code comment at `Reco.cxx:751` already acknowledges this.

10. **HitMultTrigger includes input extraction** — `trigger/HitMultTrigger.cxx:24-32` measures `GetHitTimes` (building a `std::vector<double>` copy of all hit times) as part of the trigger.
11. **TRD CreateTracks on GPU = D2H copy** — `CaTrackFinderWindow.cxx:348-350` (and 499-501) reuse the `CreateTracks` key for `CopyFittedTracksXPU_d2h`; the same key means “track candidate construction” on the CPU path (`CaTrackFinderWindow.cxx:168-170`). Same for GPU `FindTracks` , which includes fitting/merging.

4.2 Measurements that end too early (miss part of the step)

1. **TOF hit finding** — `detectors/tof/Hitfind.cxx:177` pops before the output `clusterTs = PartitionedVector(std::move(clustersFlat), sizes, fRpcAddresses)` (182). Building/sorting the partitioned output is excluded.
2. **TRD hit finding (active row-parallel path)** — `detectors/trd/Hitfind.cxx:304` pops before `hitsOut = PartitionedVector(...)` (309) and before the per-partition time sort (312-316). The module-parallel variant (527 vs 532-539) has the same issue.
3. **TRD DigiModuleSort is measured and then thrown away** — `detectors/trd/Hitfind.cxx:178-179` (`monitor.sortTime = {}`); on the active path the bucketing time is intentionally zeroed, so the metric is always 0 while the module-parallel variant keeps it.
4. **STS CPU FlattenHits** — `detectors/sts/HitfinderChainCpu.cxx:497-504` stops before offset/address construction and `PartitionedSpan` assembly (508-516).
5. **PrepareTimeslice stops before XPU initialization** — `CaTrackFinder.cxx:347-348`; the first-call `InitializeXpuForGpuTrackingOnce` (device query, `xpu::initialize` , preload) is not covered by any timer except the outer `Tracking / Reco.CA` .
6. **TrackingChain monitor timer stops before monitor plumbing** — `TrackingChain.cxx:243` vs `fCaMonitor.AddMonitorData` (257) and `output.monitorData = fCaMonitorData` (258): the copy/merge of the monitor data is excluded (the xpu timer at 114 does include it, so the two measurements disagree).
7. **KfpV0Finder::Process early return** — `KfpV0Finder.cxx:284-287` : when there are no T0 hits the function returns before `StopTimer(CollectT0)` (288) and `StopTimer(ProcessEvent)` (308). Because the monitor is reset at the start of every event (271) the lost data does not leak into other events, but the call counts no longer match `EventsTotal` (see 4.5.5), and the function also returns an **uninitialized** `bool res` (line 286).

4.3 Sections worth measuring that currently are not

1. **STS core pipeline has no dedicated timers** — the kernels/sections `SortDigis` , `FindClusters / ChannelOffsets / CreateClusters` , `SortClusters` , `FindHits` (`detectors/sts/HitfinderChain.cxx:133-225` , `HitfinderChainCpu.cxx:66-82`) are only covered by the outer `Reco.STS` (or by kernel profiling under it). In the GPU chain, the D2H flatten in `HitfinderChain::FlattenHits` (595-609) is explicitly untimed (the CPU branch is timed), and `FlattenClusters` (`HitfinderChain.cxx:625` , CPU `HitfinderChainCpu.cxx:440`) has no timer at all.
2. **GPU multi-window preparation/cleanup** — `GpuMultiWindowTrackingDev::PrepareWindows` (281-347) falls between `PrepareThread` (stopped at `CaTrackFinder.cxx:469`) and `TrackingWindow` (started at 476), so it is not separately measured; `SuppressNeighbourWindows` (245-262) and `AppendWindowOutput` (264-269) are covered only as part of `TrackingWindow` .
3. **PID parameter update inside event reconstruction** — `EventReconstruction.cxx:263-265` is inside `EventReconstruction` but has no timer, although it is a per-event step.
4. **KFP selector internals** — `KfpSelectorChain::ProcessEvent` (`KfpSelectorChain.cxx:54-78` : PID, selector, QA, counter bookkeeping) has no timer of its own; it is only covered by `evselect::KfpSelection` (and that only for thread 0).
5. **Event-building orchestration** — `EventbuildChain` digi-time extraction (`EventbuildChain.cxx:92-93` , `148-217`) and the chain-level work are untimed; only the individual triggers and `EventBuilder` are timed.

6. `TrackingChain::PrepareOutput` — track/hit-index conversion, track length computation and counter bookkeeping (`TrackingChain.cxx:179-223`) are not separately timed.
7. **CA framework glue** — `CaFramework::FindTracks / ReceiveInputData` time is only inside the CA monitor `Tracking / InputDataTransmission`; there is no timer for `InitializeXpuForGpuTrackingOnce` (see 4.2.5).
8. **Reco.EventReco granularity** — the umbrella contains reconstruction, KFP selection and monitor merging (`Reco.cxx:728-797`) without an xpu sub-timer structure.

4.4 Timer API misuse / bugs

1. **PrepareGrid is started twice and never stopped in the GPU single-window path** —

`ca/core/tracking/CaTrackFinderWindow.cxx:322-324`:

```
frMonitorData.StartTimer(ETimer::PrepareGrid);
if (iterInd != caIterations.size() - 1) { gpuTrackFinderSetup.RemoveUsedHitsFromGrid(); }
frMonitorData.StartTimer(ETimer::PrepareGrid); // should be StopTimer
```

The second call must be `StopTimer` (as in the batch path at 472-474). As written, `RemoveUsedHitsFromGrid` (GPU work + D2H sync) is never recorded and the timer is left running.

2. `monitor.sortTime = {}` discards a valid measurement (`trd/Hitfind.cxx:179`); on the default `RunRowParallel` path the `sortTime` metric is always zero.
3. **HitMultTrigger throughput is meaningless** — `trigger/HitMultTrigger.cxx:25` calls `xpu::t_add_bytes(1)`, so throughput is reported for one byte.
4. **Non-RAII push/pop is fragile** — in `CommonUnpacker.h:125-151`, `EventBuilder.cxx:22-55`, `HitMultTrigger/TimeClusterTrigger/V0Trigger`, `tof/Calibrate`, `bmon/Calibrate`, `tof/Hitfind`, `trd/Hitfind`, an exception between push and pop leaves a frame on the thread-local stack (`xpu::detail::pop_timer` cannot be reached). `xpu::scoped_timer` exists and is used elsewhere — using it consistently would remove this class of bugs. `TimeClusterTrigger.cxx` is the positive example: the sorted-input check (and its throw) happens before `push_timer`.
5. **Unbalanced push/pop in dead GPU code** — `CaGpuTrackFinderSetup.cxx:1466-1487`: if `GpuTimeMonitoring_tracking` is enabled, the `continue` at 1486/1487 skips the pop of `PrepareTreeRoots_time`, so the stack is left dirty for the rest of the function (and the final `pop_timer` at 1637 pops the wrong frame). The enclosing `GpuTimeMonitoring` pushes at 1391-1392 are then never popped either.
6. **KfpV0Finder::Process unbalanced timers + uninitialized return** — see 4.2.7.
7. **xpu::pop_timer() results discarded** — `CommonUnpacker.h:151,158,168,173` pop and drop the `xpu::timings` (data survives only as children of the enclosing `CommonUnpacker.*` timer, not as named monitor entries).
8. **CA PrepareGrid in the single-window path** — besides the missing stop, the initial grid preparation (222-224) is recorded only once, so the `PrepareGrid` metric has one call for the whole timeslice while the batch path accumulates one call per iteration.

4.5 Disabled, dead or misleading measurement code

1. **All GPU xpu timing is compiled out** — `GpuTimeMonitoring=false` (`CaDefs.h:134`) and `GpuTimeMonitoring_tracking=false` (`CaDefs.h:140`) disable the whole `xpu::push_timer` instrumentation of the GPU track finder (see 2.2). The supporting `GpuTimeMonitor` class (`CaGpuTimeMonitor.h`) is unused, and `GpuTrackFinderSetup::GetTimings()` / `fEventTimeMonitor` (`CaGpuTrackFinderSetup.h:164,211`) are never read; per-iteration results in `RunTrackConstruction` are accumulated into local floats and `printf`ed (`CaGpuTrackFinderSetup.cxx:1454-1461,1622-1633`) instead of being published.
2. **Hard-to-read dead code around RunFittingAndMerging** — `CaGpuTrackFinderSetup.cxx:892-935`: an `#if 0` block starts inside an `if constexpr` and ends inside another one, hiding the

`CloneMerging` / `FinalFitting2` timers and making the brace structure misleading.

3. Abandoned timers —

- `ETimer::PrepareContainers` is declared and named but never used (`kfp/KfpV0FinderMonitor.h:51,93`).
- `CaTrackFinder::FindTracksThread` starts a local `Timer` at 420-421 but the corresponding `Stop` /log is commented out (640-643); the object is dead.
- Commented-out `Timer` blocks at `CaTrackFinder.cxx:273-274,322-323` and `573-574,595` .
- `evreco::ETimer::BmonHitFinder` exists but the BMON block is commented out (`EventReconstruction.cxx:186-200`); `Reco.cxx:774-775` nevertheless merges it (always zero).

4. Timer-key name/semantics drift —

- GPU `CreateTracks` = D2H track copy, CPU `CreateTracks` = candidate construction.
 - GPU `FindTracks` includes fitting/merging/cleanup; CPU `FindTracks` is only the iterations loop.
 - `Timer::AddTimer` uses the wrong min/max call index (`CaTimer.h:128`).
5. **Monitor call-count bias for early returns** — `kfp::V0FinderMonitor::ProcessEvent / CollectT0` are not stopped when there are no T0 hits, while `ECounter::EventsTotal` is incremented (273). The printed average time is therefore per event *with* T0, not per processed event. The counter/timer mismatch is silent.
6. **kDEBUG block in TrackingChain** (`TrackingChain.cxx:245-255`) serialises a per-timeslice monitor file *inside* the production `TrackingChain.CA` xpu timer. There is also an impossible condition:
`if (fCaFramework.GetNofThreads() == 1 && fCaFramework.GetNofThreads() == 10)` (246) can never be true.
7. **SubTS semantics differ by backend** — window loop counts one per sub-timeslice (`CaTrackFinder.cxx:490`), the multi-window GPU adds the window count in `BuildWindows` (`CaGpuMultiWindowTrackingDev.cxx:204`); the same counter means different things per backend.

4.6 Cross-cutting observations

1. **Nested double counting is intentional but easy to misread** — e.g. `Reco` > `Reco.CA` > `TrackingChain.CA` , `TrackingChain.CA` > CA monitor timers > `Qa` ; xpu reports children separately, so the sum of all sub-timers exceeds the wall time. The `algo/base/util/TimingsFormat.*` reports handle this, but any external consumer (metrics server, QA) that adds fields must not sum parent and child.
2. **Mixed measurement systems for the same section** —
- `Reco.CA` (xpu) vs CA monitor `Tracking / FindTracks` (Monitor API) vs `fCaRecoTime` (`CaTrackFinder.cxx:152,386` , `std::chrono`) describe the same operation with different start/end points.
 - `tof::Calibrate` is measured by xpu (`monitor.fTime`) while the TS-level calibrator monitor is queued as `tofCalibTimeTotal` (`Reco.cxx:999`) from the same `xpu::timings` — consistent, but other detectors use the Monitor API only.
 - `EventReconstruction` step timers (Monitor) are converted into `evselect` fields via `AddTimer` (`Reco.cxx:772-785`) while the xpu `Reco.EventReco` measures the same loop — two separate timelines for the same work, with different QA inclusion.
3. **Per-thread merge semantics differ** — CA merges with `parallel=true` (`CaTrackFinder.cxx:391`) and event reco merges with `parallel=true` (`Reco.cxx:772-785`), so “Total [s]” is a maximum across threads, while `N Calls` is a sum. Ratios/averages computed from these monitors are inconsistent in multi-thread runs.
4. **Timer start semantics hide missing stops** — `Start` on a running timer overwrites the previous start without any error; combined with `Stop` aggregation this makes bugs like 4.4.1 silent (no exception, just a missing entry).
5. **Measurement includes memory allocation and logging in several places** — `AllocateDynamic` scoped timers (`HitfinderChain*.cxx`), `L_(info)` logging inside `Reco` and `TrackingChain.CA` , `monitor.print()` / `ToString` formatting in metric queueing functions called inside timers (`Queue*Metrics` also call `fmt::format` and `L_(debug)`).

6. `xpu_scoped_timer` results are stored in per-TS monitor structs that are reset per call — e.g. `ProcessingMonitor procMon` is local to `Reco::Run` (`Reco.cxx:577`), so `PrintTimings / QueueProcessingMetrics` see one timeslice; accumulation is done separately in `fTimesliceTimesAcc` only when `CollectKernelTimes()` is on (`Reco.cxx:906`). Same for `ProcessingExtraMonitor` in `CbmRecoDevice` (appendix B).

Appendix A — Ad-hoc time measurements outside the two APIs (inside `algo`)

These use `std::chrono` or `ca::Timer` directly and are not part of either API; listed for completeness.

Location	Mechanism	What it measures	Note
<code>ca/core/tracking/CaTrackFinder.cxx:152,386-387</code>	<code>std::chrono::high_resolution_clock</code>	<code>fCaRecoTime</code> for one <code>FindTracks</code> call	excludes (390-39. fill; store logged a
<code>ca/core/tracking/CaTrackFinder.cxx:59-91</code>	<code>ca::Timer</code> (gated)	XPU initialization in <code>InitializeXpuForGpuTrackingOnce</code>	compilec (<code>GpuTime</code>
<code>ca/core/dev/CaGpuMultiWindowTrackingDev.cxx:77-338</code>	<code>ca::Timer</code> (gated)	window building/filling/batching	compilec
<code>algo/qa/QaManager.cxx:140-167</code>	<code>ca::Timer</code>	histogram summation across threads in <code>SumDataThreads</code>	only <code>L_</code> exported

Appendix B — `online/device/CbmRecoDevice.cxx` (outside `algo`)

Location	Timer	What it measures	Observation
<code>CbmRecoDevice.cxx:110-114</code>	<code>CbmRecoDevice.WriteArchive</code> (xpu scoped)	<code>makeStorableRecoResults</code> (111: event selection, <code>ToStorable</code> conversion and full copies of all digis/hits/tracks), <code>SizeBytes()</code> (112), <code>archive->put</code> (114)	The timer name suggests pure arc writing but the dominant cost can result conversion/copies; <code>makeStorableRecoResults</code> has // some of these copies can be avo made into moves (163)
<code>CbmRecoDevice.cxx:103-105</code>	<code>std::chrono</code>	<code>extraMonitor.timeIdle</code>	interval from the previous TS-end (<code>startFetchTS</code> set at 132) to the r next TS object is available; in Rele includes the deep copy <code>std::make_unique<files::Storable (*timeslice)</code> (96), so "idle" actual contains a data copy; also not set skipped timeslices; first iteration i initial fetch
<code>CbmRecoDevice.cxx:79 + 142-144</code>	<code>std::chrono</code>	wall time from <code>PreRun</code> to <code>PostRun</code>	includes fetch waits, archive write logging and the skipped-timeslice a different clock mechanism than chain <code>Reco</code> timer, so the two tota be cross-checked track-by-track
<code>CbmRecoDevice.cxx:121</code>	—	<code>QueueProcessingExtraMetrics(extraMonitor)</code>	correctly outside any timer