CPU Performance Study of HGCalorimeter Code

Abhishek Das
Advisor- Dr. Mike Hildreth
University of Notre Dame

To summarise the work -

We modified the way the HGCDigitizerBase produces gaussian noise and adds it to the Simhits in the digitization step of a MC Simulation workflow to make it run faster.

What we found in the original HGCDigitizer Code

HGCalSimProducer has 4 subdigitizer modules - HEFrontDigitizer, HFNoseDigitizer EEDigitizer, HEBackDigitizer(HGCDigitizerBase runs on first 3 modules).

HFNoseDigitizer has 910300 active cell units(with unique DetIds) ; 910300 x 15=13,654,500 simhits/event

EEDigitizer has 4126824 active cell units ; 4126824 x 15 = 61,902,360 simhits/event

HEFrontDigitizer has 1882950 active cell units ; 1882950 x 15 = 28,244,250 simhits/event

In the original Digitizer code, for the digitization step these subdigitizers generate random gaussian noise(mean=0,stddev=1) for each simhit for each event to mix with the simhits.

Our modification...

We recognized the redundancy of generating millions of random numbers freshly for each event, especially when we saw from the CPU performance profiler that the random number generator consumes nearly 13.3%(for zero pileup case) of the total CPU time and worsening the time complexity of the digitizer step in the workflow.

Performance worsens further when we run the code for realistic high PU workflow.

Instead of generating random gaussian noise on the fly, we precompute and store 10000x15 = 150,000 random gaussian numbers(separately for each of the 3 digitizers mentioned) in a vector before starting to process the events.

In the digitization step, we hash each simhit uniformly to one of the random numbers stored in the vector and use that random number as noise for each event.

Workflow specification(for zero PU and classical mixing of PU) :

We experimented with the performance of the HGCal code in the digitization step on TTbar MC events at 13/14 TeV with a varied number of pileup mixing.

For zero pileup - step2 --conditions auto:phase2_realistic -s DIGI:pdigi_valid,L1,L1TrackTrigger,DIGI2RAW,HLT:@fake2 --datatier GEN-SIM-DIGI-RAW -n 10 --geometry Extended2023D31 --era Phase2C6 --eventcontent FEVTDEBUGHLT

For finite pileup mixing - step2 --conditions auto:phase2_realistic --pileup_input das:/RelValMinBias_14TeV/1/GEN-SIM -n 10 --era Phase2C6 --eventcontent FEVTDEBUGHLT -s DIGI:pdigi_valid,L1,L1TrackTrigger,DIGI2RAW,HLT:@fake2 --datatier GEN-SIM-DIGI-RAW --pileup AVE_35_BX_25ns --geometry Extended2023D31

Workflow specification(for premixing of PU on data) :

Step 3 - step3 --datamix PreMix --conditions auto:phase2_realistic --pileup_input file:step2.root -n 10 --era Phase2C8_timing_layer_bar --eventcontent FEVTDEBUGHLT --procModifiers premix_stage2 --filein file:step1.root -s DIGI:pdigi_valid,DATAMIX,L1,L1TrackTrigger,DIGI2RAW,HLT:@fake2 --datatier GEN-SIM-DIGI-RAW --geometry Extended2023D41

cmsRun job specification

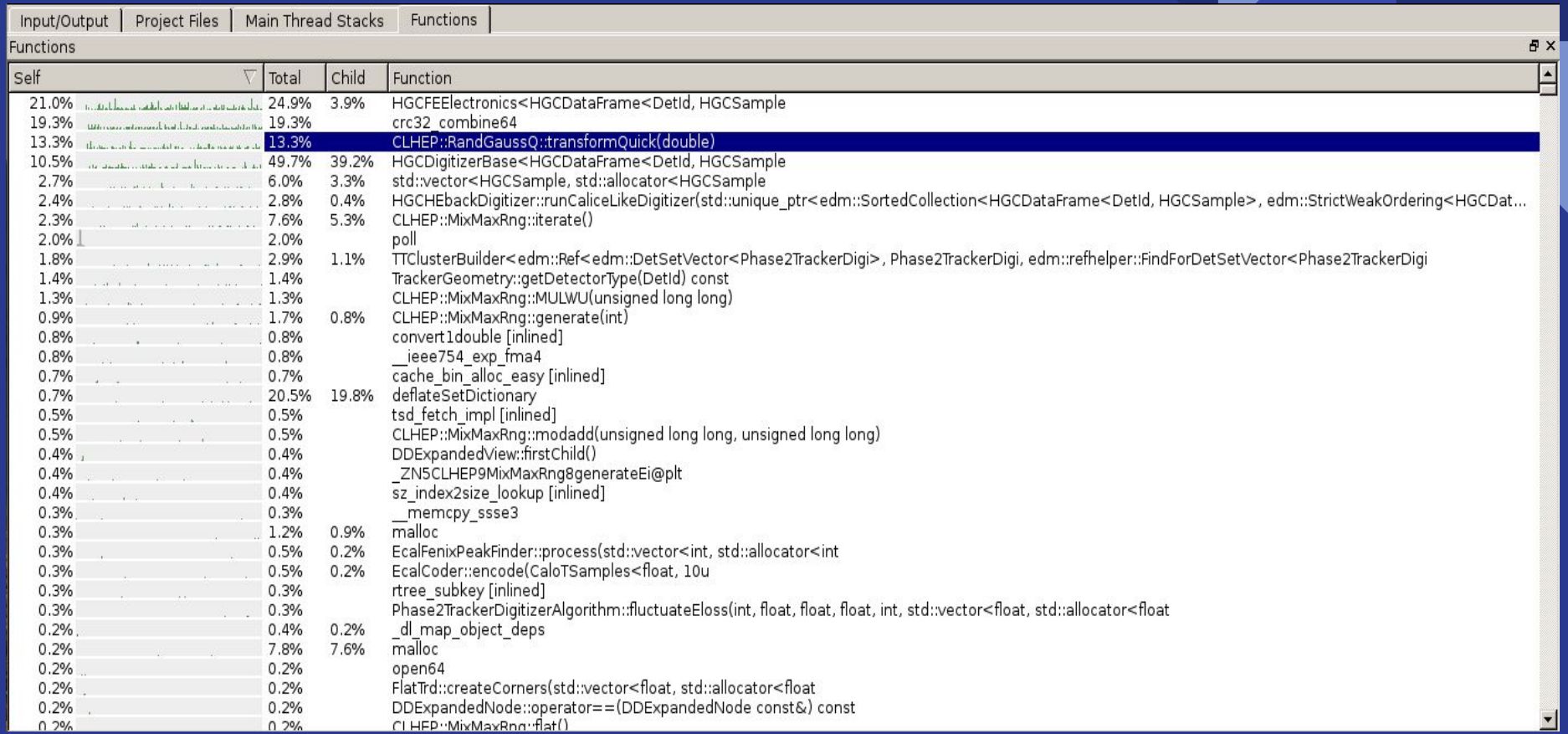
We used both Allinea MAP and igprof to study the performance of CPU while running the cmsRun job as a single process on 2 cores on a single node.

For 100 events with zero pileup -

The step2 with original version of the HGCal code took 2449.2 seconds.

The modified version of the HGCal code helped finish the step 2 in 1846 seconds.(~75% of the time taken by original code)

Screenshot from MAP Profiler for Original Digitization Code(PU=0)

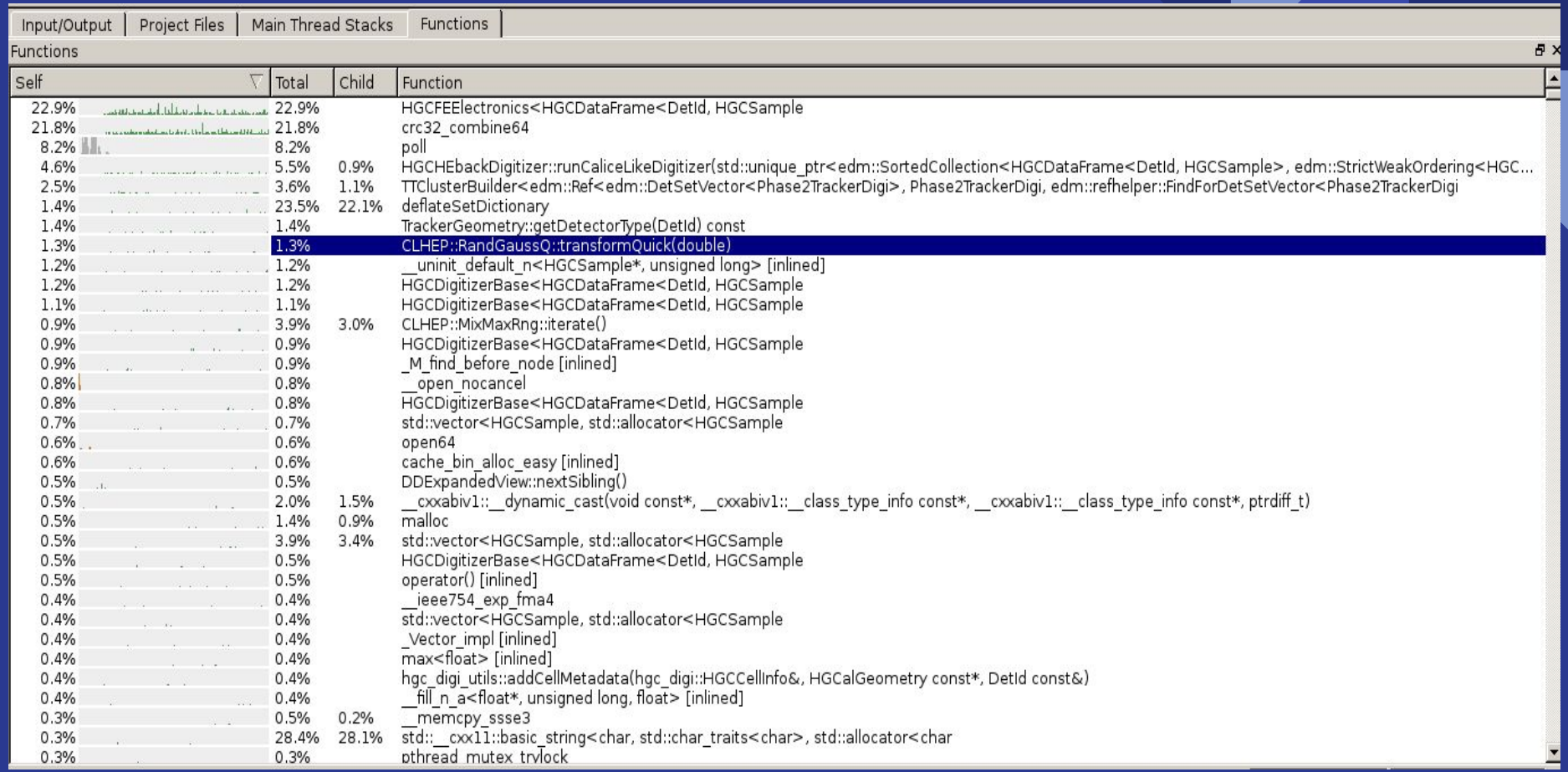


Input/Output | Project Files | Main Thread Stacks | Functions

Functions

Self	Total	Child	Function
21.0%	24.9%	3.9%	HGCFEElectronics<HGCDataFrame<DetId, HGCSample
19.3%	19.3%		crc32_combine64
13.3%	13.3%		CLHEP::RandGaussQ::transformQuick(double)
10.5%	49.7%	39.2%	HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
2.7%	6.0%	3.3%	std::vector<HGCSample, std::allocator<HGCSample
2.4%	2.8%	0.4%	HGCHEbackDigitizer::runCaliceLikeDigitizer(std::unique_ptr<edm::SortedCollection<HGCDataFrame<DetId, HGCSample>, edm::StrictWeakOrdering<HGCDat...
2.3%	7.6%	5.3%	CLHEP::MixMaxRng::iterate()
2.0%	2.0%		poll
1.8%	2.9%	1.1%	TTClusterBuilder<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForDetSetVector<Phase2TrackerDigi
1.4%	1.4%		TrackerGeometry::getDetectorType(DetId) const
1.3%	1.3%		CLHEP::MixMaxRng::MULWU(unsigned long long)
0.9%	1.7%	0.8%	CLHEP::MixMaxRng::generate(int)
0.8%	0.8%		convert1double [inlined]
0.8%	0.8%		__ieee754_exp_fma4
0.7%	0.7%		cache_bin_alloc_easy [inlined]
0.7%	20.5%	19.8%	deflateSetDictionary
0.5%	0.5%		tsd_fetch_impl [inlined]
0.5%	0.5%		CLHEP::MixMaxRng::modadd(unsigned long long, unsigned long long)
0.4%	0.4%		DDExpandedView::firstChild()
0.4%	0.4%		_ZN5CLHEP9MixMaxRng8generateEi@plt
0.4%	0.4%		sz_index2size_lookup [inlined]
0.3%	0.3%		__memcpy_ssse3
0.3%	1.2%	0.9%	malloc
0.3%	0.5%	0.2%	EcalFenixPeakFinder::process(std::vector<int, std::allocator<int
0.3%	0.5%	0.2%	EcalCoder::encode(CaloTSamples<float, 10u
0.3%	0.3%		rtree_subkey [inlined]
0.3%	0.3%		Phase2TrackerDigitizerAlgorithm::fluctuateEloss(int, float, float, float, int, std::vector<float, std::allocator<float
0.2%	0.4%	0.2%	_dl_map_object_deps
0.2%	7.8%	7.6%	malloc
0.2%	0.2%		open64
0.2%	0.2%		FlatTrd::createCorners(std::vector<float, std::allocator<float
0.2%	0.2%		DDExpandedNode::operator==(DDExpandedNode const&) const
0.2%	0.2%		CLHEP::MixMaxRng::flat()

Screenshot from MAP Profiler for Modified Digitization Code(PU=0)



Input/Output | Project Files | Main Thread Stacks | Functions

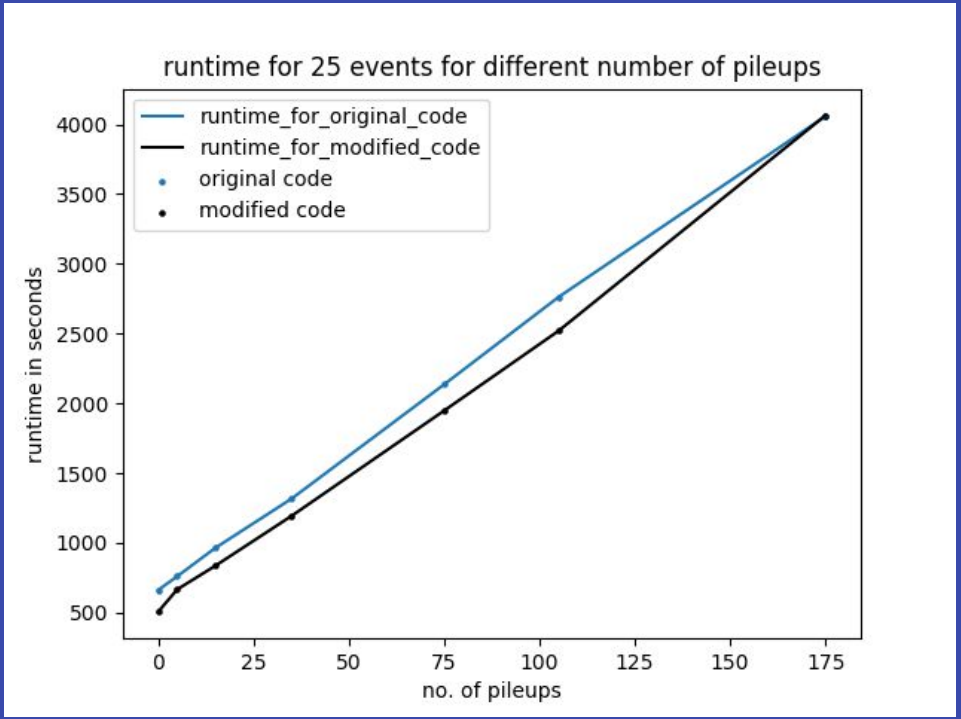
Functions

Self	Total	Child	Function
22.9%	22.9%		HGCFEElectronics<HGCDataFrame<DetId, HGCSample
21.8%	21.8%		crc32_combine64
8.2%	8.2%		poll
4.6%	5.5%	0.9%	HGCHEbackDigitizer::runCaliceLikeDigitizer(std::unique_ptr<edm::SortedCollection<HGCDataFrame<DetId, HGCSample>, edm::StrictWeakOrdering<HGC...
2.5%	3.6%	1.1%	TTClusterBuilder<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForDetSetVector<Phase2TrackerDigi
1.4%	23.5%	22.1%	deflateSetDictionary
1.4%	1.4%		TrackerGeometry::getDetectorType(DetId) const
1.3%	1.3%		CLHEP::RandGaussQ::transformQuick(double)
1.2%	1.2%		__uninit_default_n<HGCSample*, unsigned long> [inlined]
1.2%	1.2%		HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
1.1%	1.1%		HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
0.9%	3.9%	3.0%	CLHEP::MixMaxRng::iterate()
0.9%	0.9%		HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
0.9%	0.9%		_M_find_before_node [inlined]
0.8%	0.8%		__open_nocancel
0.8%	0.8%		HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
0.7%	0.7%		std::vector<HGCSample, std::allocator<HGCSample
0.6%	0.6%		open64
0.6%	0.6%		cache_bin_alloc_easy [inlined]
0.5%	0.5%		DDExpandedView::nextSibling()
0.5%	2.0%	1.5%	__cxxabiv1::__dynamic_cast(void const*, __cxxabiv1::__class_type_info const*, __cxxabiv1::__class_type_info const*, ptrdiff_t)
0.5%	1.4%	0.9%	malloc
0.5%	3.9%	3.4%	std::vector<HGCSample, std::allocator<HGCSample
0.5%	0.5%		HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
0.5%	0.5%		operator() [inlined]
0.4%	0.4%		__ieee754_exp_fma4
0.4%	0.4%		std::vector<HGCSample, std::allocator<HGCSample
0.4%	0.4%		_Vector_impl [inlined]
0.4%	0.4%		max<float> [inlined]
0.4%	0.4%		hgc_digi_utils::addCellMetadata(hgc_digi::HGCCellInfo&, HGCalGeometry const*, DetId const&)
0.4%	0.4%		__fill_n_a<float*, unsigned long, float> [inlined]
0.3%	0.5%	0.2%	__memcpy_ssse3
0.3%	28.4%	28.1%	std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char
0.3%	0.3%		pthread_mutex_trylock

Performance Plots(classical mixing)

We also ran tests to compare the performance of our modified code for different number of pileup mixing - 5,15,35,75,105,175.

The run-time for 25 events for step 2 with original and modified HGCal Digitizer Code is shown in the plot.

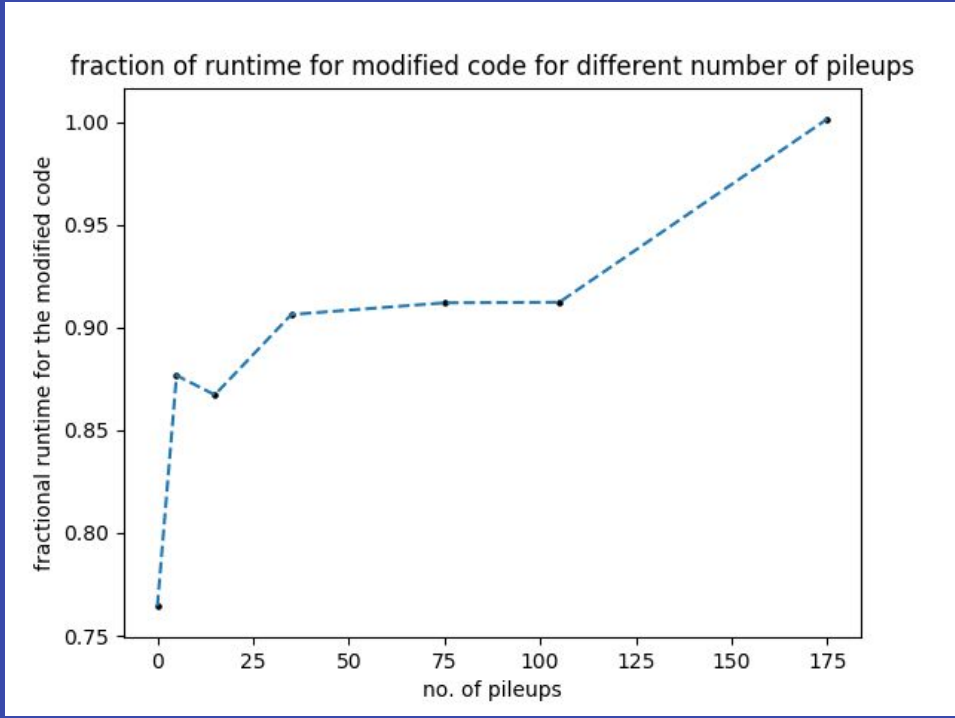


Performance Plots(classical mixing)

For lower number of pileup mixing, the digitization step using the modified HGCal digitizer code shows improved performance over time.

With increasing number of pileup mixing, the advantage is lost as increased number of IO operations significantly increases the overhead.

(our initial guess!!!)



Performance for Premix Workflow

For mean PU no. 175, the step 3 of premix workflow took 11646 seconds to run on 100 events with original HGCal-digi code, whereas our modified digitization helped it finish in 11050-11182 seconds, i.e. it helps run job ~5.2-4% faster.

The average time taken by the digitizer subroutine and the function calling Random number generator is also lowered significantly.

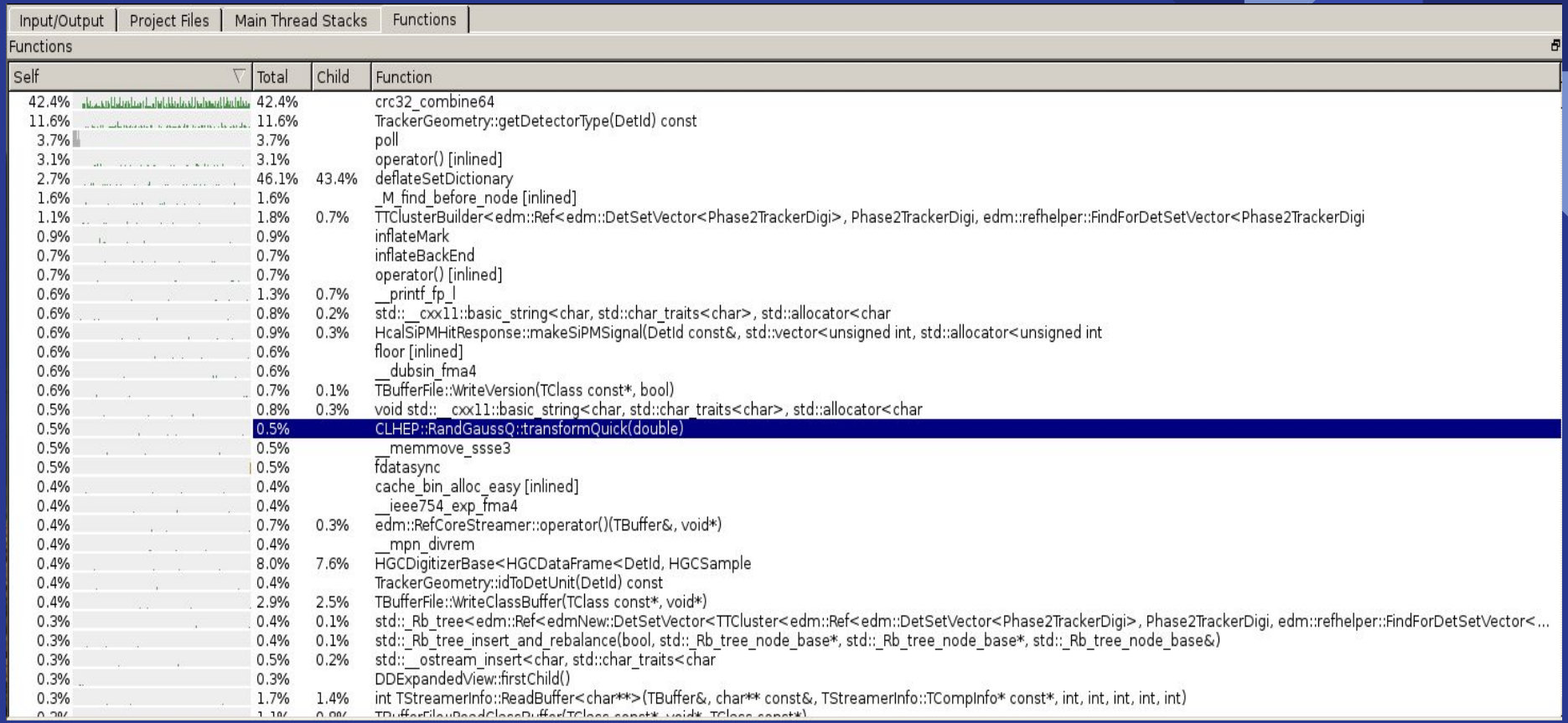
Screenshot from MAP Profiler for Original Digitization Code in Premix workflow(PU=175)

Input/Output | Project Files | Main Thread Stacks | Functions

Functions

Self	Total	Child	Function
40.7%	40.7%		crc32_combine64
11.6%	11.6%		TrackerGeometry::getDetectorType(DetId) const
4.7%	4.7%		HGCFEElectronics<HGCDataFrame<DetId, HGCSample
3.8%	3.8%		poll
3.1%	3.1%		CLHEP::RandGaussQ::transformQuick(double)
2.7%	43.9%	41.2%	deflateSetDictionary
2.1%	12.0%	9.9%	HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
1.5%	1.9%	0.4%	TTClusterBuilder<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForDetSetVector<Phase2TrackerDigi
1.0%	1.4%	0.4%	HGCDigitizer::accumulate(PHGCSimAccumulator const&)
0.9%	0.9%		inflateBackEnd
0.7%	0.7%		TBufferFile::WriteVersion(TClass const*, bool)
0.6%	1.5%	0.9%	tcache_dalloc_small [inlined]
0.6%	13.3%	12.7%	HGCDigitizer::finalizeEvent(edm::Event&, edm::EventSetup const&, CLHEP::HepRandomEngine*)
0.6%	1.1%	0.5%	__printf_fp_l
0.6%	0.6%		__memcpy_ssse3
0.5%	1.1%	0.6%	PreMixingDigiSimLinkWorker<edm::DetSetVector<PixelDigiSimLink
0.5%	0.9%	0.4%	CLHEP::MixMaxRng::iterate()
0.5%	0.7%	0.2%	std::vector<HGCSample, std::allocator<HGCSample
0.5%	0.5%		fdatasync
0.4%	0.5%	0.1%	std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char
0.4%	0.8%	0.4%	free
0.4%	1.1%	0.7%	int TStreamerInfo::ReadBuffer<char**>(TBuffer&, char** const&, TStreamerInfo::TCompInfo* const*, int, int, int, int, int)
0.4%	0.4%		local_Rb_tree_increment [inlined]
0.4%	0.4%		uselocale
0.4%	0.4%		madvise
0.4%	0.9%	0.5%	HcalSiPMHitResponse::makeSiPMSignal(DetId const&, std::vector<unsigned int, std::allocator<unsigned int
0.4%	0.4%		HcalSiPM::setNCells(int)
0.4%	0.4%		convert1double [inlined]
0.4%	0.4%		std::_Hashtable<unsigned int, std::pair<unsigned int const, unsigned int>, std::allocator<std::pair<unsigned int const, unsigned int
0.4%	0.4%		void std::vector<l1t::HGCalTriggerCell, std::allocator<l1t::HGCalTriggerCell
0.3%	0.3%		std::_Rb_tree<edm::Ref<edmNew::DetSetVector<TTCluster<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForD...
0.3%	0.3%		std::_Rb_tree_insert_and_rebalance(bool, std::_Rb_tree_node_base*, std::_Rb_tree_node_base*, std::_Rb_tree_node_base&)

Screenshot from MAP Profiler for Modified Digitization Code in Premix workflow(PU=175)



Input/Output | Project Files | Main Thread Stacks | Functions

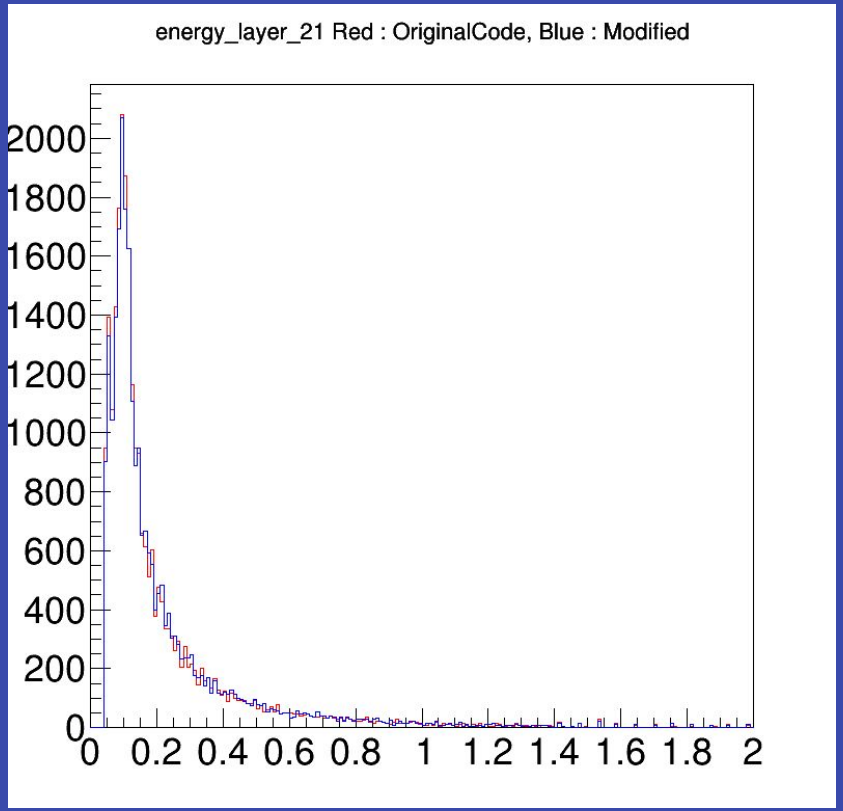
Functions

Self	Total	Child	Function
42.4%	42.4%		crc32_combine64
11.6%	11.6%		TrackerGeometry::getDetectorType(DetId) const
3.7%	3.7%		poll
3.1%	3.1%		operator() [inlined]
2.7%	46.1%	43.4%	deflateSetDictionary
1.6%	1.6%		_M_find_before_node [inlined]
1.1%	1.8%	0.7%	TTClusterBuilder<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForDetSetVector<Phase2TrackerDigi
0.9%	0.9%		inflateMark
0.7%	0.7%		inflateBackEnd
0.7%	0.7%		operator() [inlined]
0.6%	1.3%	0.7%	__printf_fp_l
0.6%	0.8%	0.2%	std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char
0.6%	0.9%	0.3%	HcalSiPMHitResponse::makeSiPMSignal(DetId const&, std::vector<unsigned int, std::allocator<unsigned int
0.6%	0.6%		floor [inlined]
0.6%	0.6%		__dubsin_fma4
0.6%	0.7%	0.1%	TBufferFile::WriteVersion(TClass const*, bool)
0.5%	0.8%	0.3%	void std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char
0.5%	0.5%		CLHEP::RandGaussQ::transformQuick(double)
0.5%	0.5%		__memmove_ssse3
0.5%	0.5%		fdatasync
0.4%	0.4%		cache_bin_alloc_easy [inlined]
0.4%	0.4%		__ieee754_exp_fma4
0.4%	0.7%	0.3%	edm::RefCoreStreamer::operator()(TBuffer&, void*)
0.4%	0.4%		__mpn_divrem
0.4%	8.0%	7.6%	HGCDigitizerBase<HGCDataFrame<DetId, HGCSample
0.4%	0.4%		TrackerGeometry::idToDetUnit(DetId) const
0.4%	2.9%	2.5%	TBufferFile::WriteClassBuffer(TClass const*, void*)
0.3%	0.4%	0.1%	std::_Rb_tree<edm::Ref<edmNew::DetSetVector<TTCluster<edm::Ref<edm::DetSetVector<Phase2TrackerDigi>, Phase2TrackerDigi, edm::refhelper::FindForDetSetVector<...
0.3%	0.4%	0.1%	std::_Rb_tree_insert_and_rebalance(bool, std::_Rb_tree_node_base*, std::_Rb_tree_node_base*, std::_Rb_tree_node_base&)
0.3%	0.5%	0.2%	std::__ostream_insert<char, std::char_traits<char
0.3%	0.3%		DDExpandedView::firstChild()
0.3%	1.7%	1.4%	int TStreamerInfo::ReadBuffer<char**>(TBuffer&, char** const&, TStreamerInfo::TCompInfo* const*, int, int, int, int, int)

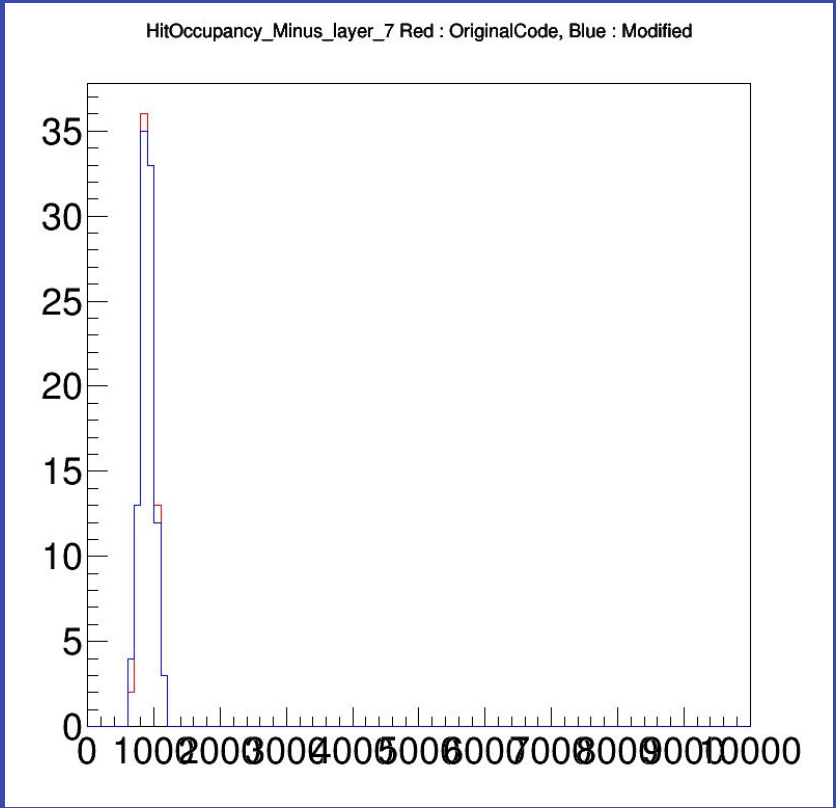
Comparable result at Digi and Reco level(classical Mixing)

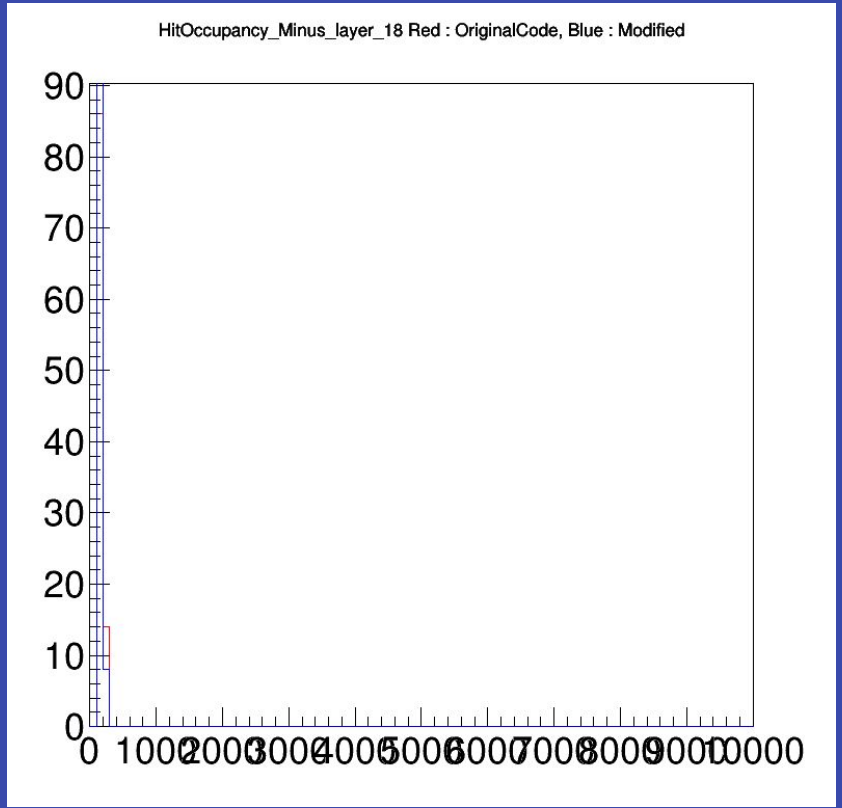
For the HESiliconSensitive module, we compared the validation plots of energy & HitOccupancy(at RecHit lavel) and ADC & DigiOccupancy(at digi level) obtained using our modified HGCal Digitizer code with those obtained using original HGCal code and found them to be nearly similar.

Thus, according to our finding, our modified noise infusion model for the HGCal digitizer does not alter the physics output of the detector.

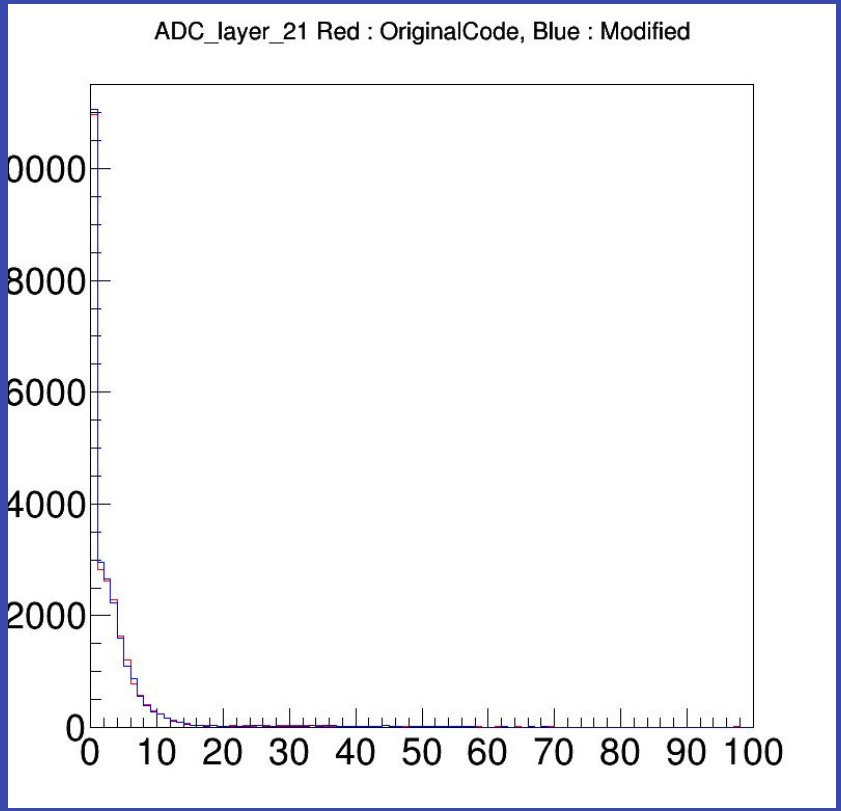


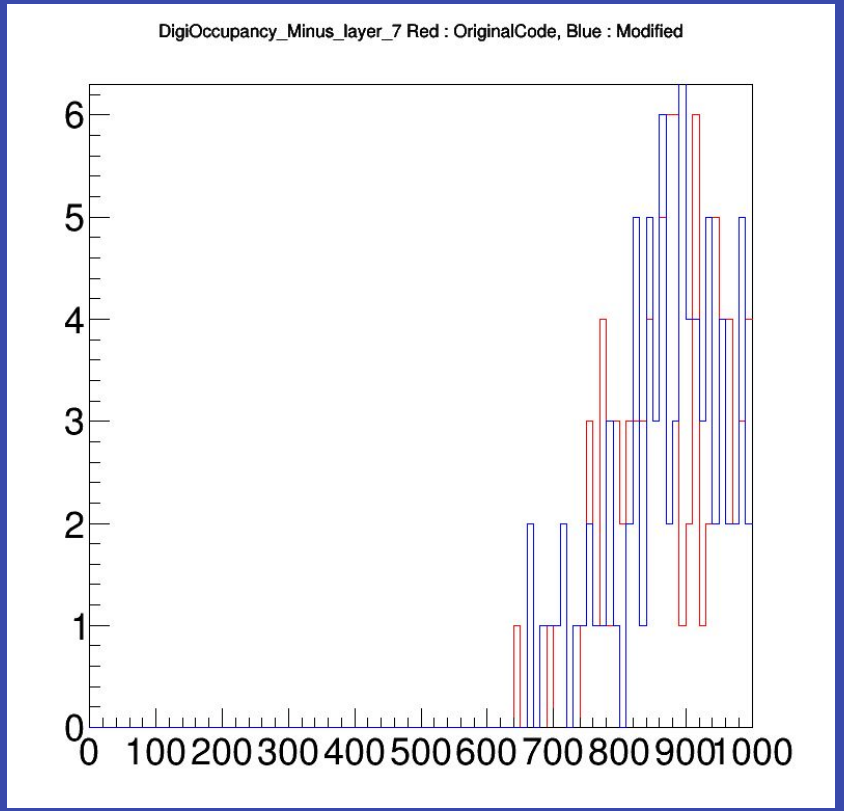
Comparable result at Reco level





Comparable result at Digi level





Thank You !!