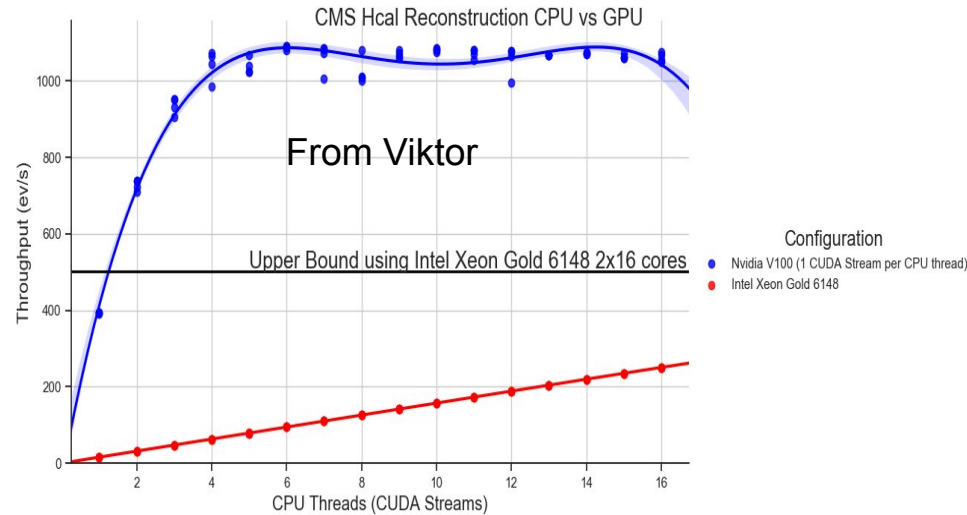
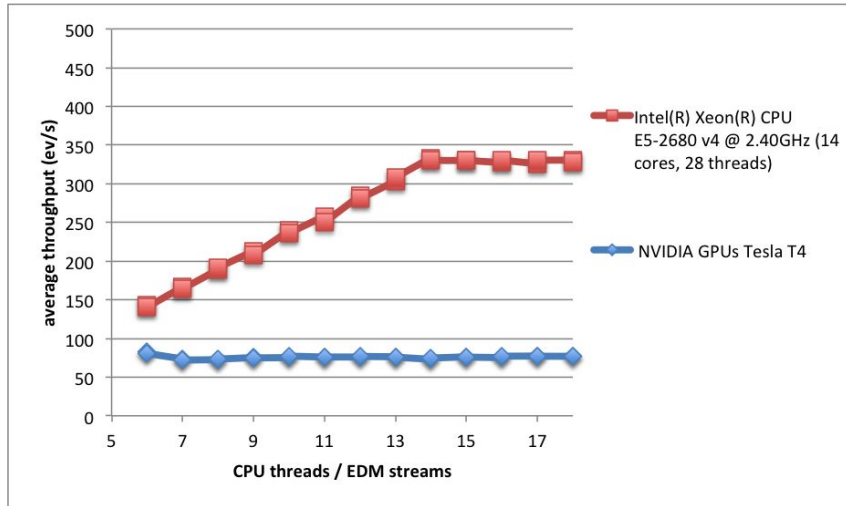


Mahi on GPU

Two workflow available with two performances

V1 = Only mahi on GPU
Not optimized yet

V2 = All HCAL sequence on GPU
(from umpacker to conditions and mahi) super optimized



Goal of the week

Focus on “V1” and check how much is :

1. Copy of CMSSW objects from CPU to GPU
2. memory saturation
3. recHit asynchronous (i.e. number of iterations are different for rechit)
4. ... or just bad cloning from CPU to GPU

Today started profiling with NVIDIA-Nsight-Compute-2019.4

The screenshot displays the NVIDIA Nsight Compute interface. At the top, the menu bar includes File, Connection, Debug, Profile, Tools, Window, and Help. Below the menu, there are buttons for Connect, Disconnect, Terminate, Profile Kernel, and various control icons. The main window shows a profiling report for a kernel named 'kernel_reconstruct' (Launch: 0 - 127 - kernel_reconstruct). The report includes the following summary information:

- Page: Source
- Process: All
- Launch: 0 - 127 - kernel_reconstruct
- Buttons: Add Baseline, Apply Rules
- Copy icon
- Current: 127 - kernel_reconstruct (5248, 1, 1) Time: 13.32msecond Cycles: 7,858,220 Regs: 255 GPU: Tesla T4 SM Frequency: 589.86 cycle/usecond CC: 7.5 Process: [42121] cmsRun
- View: Source
- File: MahiFit_gpu.cu
- Buttons: P+, P-, Sampling Data (All), and various analysis icons

The main part of the interface is a table showing the execution of the kernel's source code. The table has the following columns: #, Source, Live Registers, Sampling Data (All), Sampling Data (Not Issued), Instructions Executed, Predicated-On Thread Instructions Executed, and Mem. The source code is shown with line numbers from 477 to 505. The table provides performance metrics for each line of code, including the number of instructions executed and the number of instructions executed on threads that were predicated on.

#	Source	Live Registers	Sampling Data (All)	Sampling Data (Not Issued)	Instructions Executed	Predicated-On Thread Instructions Executed	Mem
477			0	0			
478	auto invDt = 0.5 / nnlswork_dt;		30	30	19,690	391,905	
479			0	0			
480	for (unsigned int ITS=0; ITS<nnlswork_tsSize; ++ITS) {		15	14	35,612	631,773	
481			0	0			
482	pulseShape.coeffRef(ITS+nnlswork_maxoffset) = pulseN[ITS+delta];		815	792	73,528	1,463,448	
483	pulseDeriv.coeffRef(ITS+nnlswork_maxoffset) = (pulseM[ITS+delta]-pulseP[ITS+delta]);		833	819	52,620	1,045,320	
484			0	0			
485	pulseM[ITS+delta] -= pulseN[ITS+delta];		15	10	21,000	418,128	
486	pulseP[ITS+delta] -= pulseM[ITS+delta];		85	83	21,000	418,128	
487	}		0	0			
488			0	0			
489	for (unsigned int ITS=0; ITS<nnlswork_tsSize; ++ITS) {		67	51	28,880	522,660	
490	for (unsigned int JTS=0; JTS<iTS+1; ++JTS) {		628	646	283,280	4,642,512	
491			0	0			
492	double tmp = 0.5*(pulseP[iTS+delta]*pulseP[JTS+delta] +		1,577	1,463	341,380	6,637,782	
493	pulseM[iTS+delta]*pulseM[JTS+delta]);		0	0			
494			0	0			
495	pulseCov(ITS+nnlswork_maxoffset,JTS+nnlswork_maxoffset) += tmp;		4,585	4,494	309,868	6,167,388	
496	if (JTS==ITS) pulseCov(JTS+nnlswork_maxoffset,ITS+nnlswork_maxoffset)		4,320	4,229	262,600	5,017,536	
497	}		0	0			
498	}		0	0			
499			0	0			
500	}		0	0			
501			0	0			
502	__device__		0	0			
503	void MahiFit::updateCov() const {		0	0			
504			0	0			
505	SampleMatrix invCovMat;		0	0			