- GPU Cherenkov -

- Process Sequence -

and others

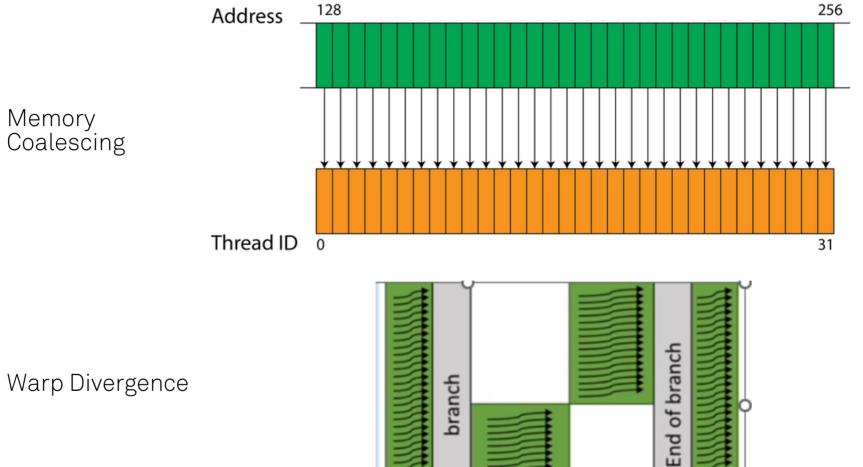
								L1 Inst	tructi	on Cac	he										
L0 Instruction Cache Warp Scheduler (32 thread/clk) Dispatch Unit (32 thread/clk)											L0 Instruction Cache Warp Scheduler (32 thread/clk) Dispatch Unit (32 thread/clk)										
	Register File (16,384 x 32-bit)											Register File (16,384 x 32-bit)									
INT32	INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32	FP	64					INT32	INT32	FP32	P32 FP32 FP		64						
INT32	INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32	FP	64		TENCOD CODE			INT32 INT32 FP32 FP32		FP	64								
INT32	INT32	FP32	FP32	FP64		TENSOR CORE				INT32 INT32 FP32 FP32 F			FP	64	TENSOR CORE						
INT32	INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32	FP	64					INT32 INT32 FP32	FP32	FP32	FP	64							
LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU			
	L0 Instruction Cache												L0 Instruction Cache								
		War	p Sch	eduleı	r (32 ti	hread	/clk)		5	Warp Scheduler (32 thread/clk)											
		Dis	spatch	Unit	(32 th	read/c	:lk)		Dispatch Unit (32 thread/clk)												
	Register File (16,384 x 32-bit)										Register File (16,384 x 32-bit)										
INT32	INT32	FP32	FP32	FP	64				וו	INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32	FP64 FP64 FP64 FP64 FP64 FP64		TENSOR CORE				INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32							INT32	INT32	FP32	FP32	FP	64	TENSOR CORE					
INT32	INT32	FP32	FP32							INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32							INT32	INT32	FP32	FP32	FP	64						
INT32	INT32	FP32	FP32						INT32 INT32 FP32 FP INT32 INT32 FP32 FP		FP32	FP64									
INT32	INT32	FP32	FP32									FP32	FP32	FP64							
INT32	INT32	FP32	FP32	FP64						INT32	INT32	FP32 FP32		FP64							
LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU			
192KB L1 Data Cache / Shared Memory																					
													(

SM L0 Instruction Cache Warp Scheduler (32 thread/cll Warp Scheduler (32 thread/clk Dispatch Unit (32 thread/clk) Dispatch Unit (32 thread/clk) Register File (16,384 x 32-bit) Register File (16,384 x 32-bit) INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 FP32 FP32 INT32 FP32 FP32 INT32 FP64 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 TENSOR CORE TENSOR CORE INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 FP32 FP32 INT32 FP32 FP32 INT32 FP64 4th GENERATION FP64 4th GENERATION INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 INT32 FP32 FP32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 LD/ LD/ ST ST LD/ LD/ LD/ ST ST ST LD/ ST LD/ LD/ ST ST LD/ ST LD/ ST LD/ ST LD/ LD/ ST LD/ ST LD/ ST SFU SFU ST Warp Scheduler (32 thread/cll Warp Scheduler (32 thread/clk Dispatch Unit (32 thread/clk) Dispatch Unit (32 thread/clk) Register File (16,384 x 32-bit) Register File (16,384 x 32-bit) INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 FP64 **TENSOR CORE** INT32 FP32 FP32 FP64 **TENSOR CORE** INT32 FP32 FP32 INT32 FP64 4th GENERATION FP32 FP32 4th GENERATION FP64 INT32 FP32 FP32 INT32 FP64 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 INT32 FP32 FP32 FP64 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 INT32 FP32 FP32 FP64 LD/ LD/ LD/ LD/ ST ST ST ST LD/ LD/ ST ST LD/ LD/ ST ST LD/ LD/ ST ST LD/ ST LD/ LD/ ST ST LD/ SFU SFU ST **Tensor Memory Accelerator** 256 KB L1 Data Cache / Shared Memory Tex Tex Tex Tex

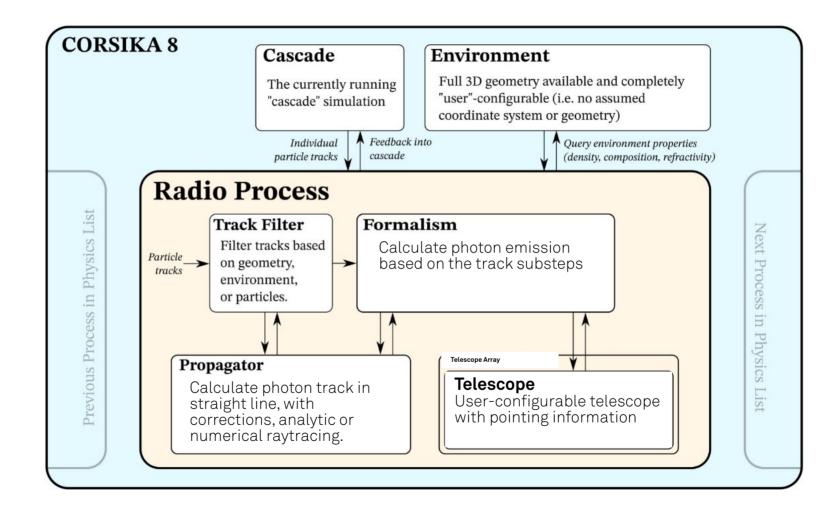
132x SMs H100

GPUs Why?

A100 108x SMs



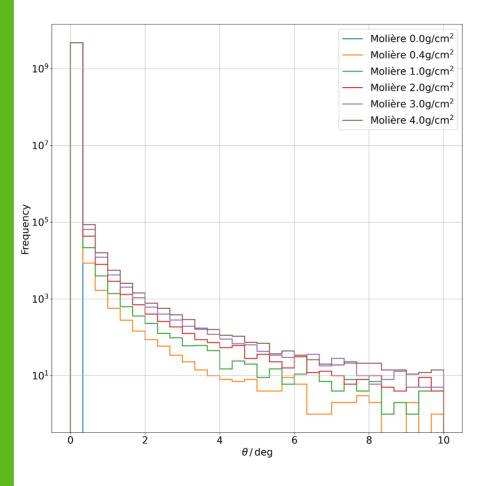
GPUs How?

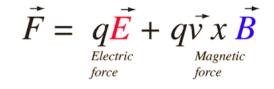


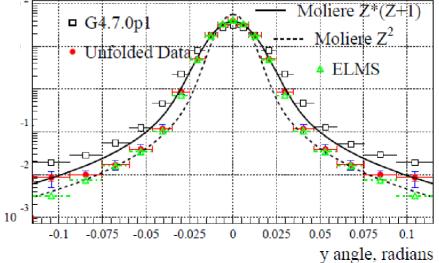
GPUs How?

- 1) Collect particle substeps (linear traces) and reduce to coordinate, direction, velocity and time
- 2) Transfer simplified particles to GPU
- 3) At least 10 particles per AU \rightarrow 320 per Warp to allow for effective latency hiding
- 4) Store surviving particles in local shared memory (locally & fast), go back to 3) if number is small
- 5) Generate photons (position, direction, Wavelength, Time) and write to memory
- 6) "Iterate" through photons and calculate straight line impact on horizontal plan, afterwards apply correction. Clip with array boundary's and store in shared memory.
- 7) 1 Telescope per warp, N Photons → Check for hit and store in global memory for download on host machine

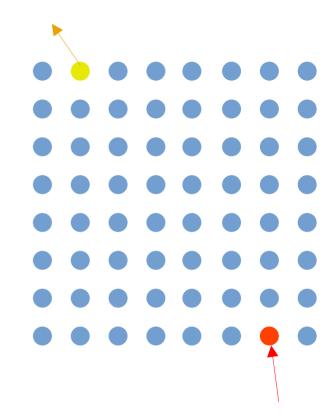








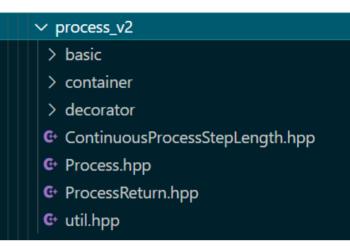
- Weight edges with probability / (1-p), take path with highest/lowest probability for path interpolation. Alternative: Potential field and gradient-descent
- Sample random path or optimal path?



Now something different

Process Sequence V2

- Clearer Structure and divide between base clases, storage classes and modifications
- Removed several levels of templates and meta templates



```
Changes
```

```
You, last month | 1 author (You)

namespace corsika {

You, last month | 1 author (You)

template <ConceptProcess... TProcesses>

class ProcessSequence;

You, last month | 1 author (You)
```

```
template <ConceptProcess TProcess, ConceptProcess... TSequence>
class ProcessSequence<TProcess, TSequence...> : public ProcessSequence<TSequence...> {
    private:
        using process_type = typename std::decay_t<TProcess>;
        using sequence_type = ProcessSequence<TSequence...>;
```

TProcess process_;

- Clearer Structure and divide between base clases, storage classes and modifications
- Removed several levels of templates and meta templates
- ProcessSequence is now a variadic template, which removes code duplication

```
Changes
```

```
You, 6 days ago | 1 author (You)
                               template <typename TCondition, typename TSequence, typename USequence>
                                class SwitchProcess
                                    : public BaseProcessContainer<SwitchProcess<TCondition, TSequence, USequence>>,
                                     SwitchProcessBase {
                                  using switch type = typename std::decay t<TCondition>;
                                 using process1 type = typename std::decay t<TSequence>;
                                 using process2 type = typename std::decay t<USequence>;
                                private:
                                  TCondition select_; /// selector functor to switch between branch a and b, this is a
                                    /// reference, if possible
                                  corsika::ProcessSequence<TSequence>
                                     A_; /// process branch a, this is a reference, if possible
- Clearer Structure and d
                                  corsika::ProcessSequence<USequence>
                                      B_; /// process branch b, this is a reference, if possible
  base clases, storage cla
```

- Removed several levels of templates and meta templates

modifications

- ProcessSequence is now a variadic template, which removes code duplication
- SwitchProcess stores only ProcessSequences \rightarrow removes the remaining code duplication

- All currently used templated arguments are easility known by the stack → move over to shared class template Tstack and derive information's from it This <u>allows the user of virtual</u> functions and make modules callable from outside!
- For this Stack rework
- Cleanup unused/dead code (separate issues #595)
- Change interfaces to

```
You, 5 days ago | 1 author (You)
struct DecayProcess : virtual BaseProcess<DecayProcess> {
  public:
    template <typename TView>
    void doDecay(TView& view);
    template <typename TParticle>
    TimeType getLifetime(TParticle& particle);
```

Stack V1

- Currently Stack and data to store are heavily interlinked
- Data is stored by information → Array of positions, Array or direction, ... (Unconfirmed) Caching problem, every information require separate load
- Information stored and information required by modules are completely separate, not meeting requirements fails with <u>looooong</u> template errors.

Stack V2

- Separation between functionality and Data
- Data should be driven by selected modules e.g. automatically include history information if required

No LTO / IPO (Link Time / Inter procedural Optimization) → ODR (One Definition Rule) issues not catched → ABI (Application Binary Interface) issues not catched

cmake_minimum_required(VERSION 3.9.4)

include(CheckIPOSupported)
check_ipo_supported(RESULT supported OUTPUT error)

add_executable(example Example.cpp)

```
if( supported )
    message(STATUS "IPO / LTO enabled")
    set_property(TARGET example PROPERTY INTERPROCEDURAL_OPTIMIZATION TRUE)
else()
    message(STATUS "IPO / LTO not supported: <${error}>")
endif()
```

Function Parameter order arbitrary

- clang-tidy bugprone-easily-swappable-parameters



: range (range) {}

module_control.addOption(
 control::ControlOption<double>("test3", "test4", 1.0)
 .setConstraint(control::checks::RangeCheck<double>({.min=0.0, .max=1.0}))
 .setConstraint(control::checks::NeedsCheck<double>(option1)));

But C++ 20