Energy Efficiency and Performance Analysis of Monte-Carlo Event Generators for HEP

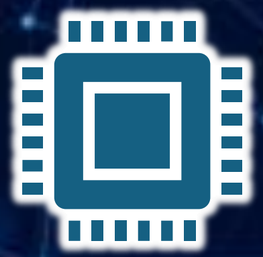
Raisa Rahman Richi, Franklin and Marshall College

Supervisors: Daniele Massaro, Stefan Roiser, Markus Schulz

12/08/2025

OBJECTIVE

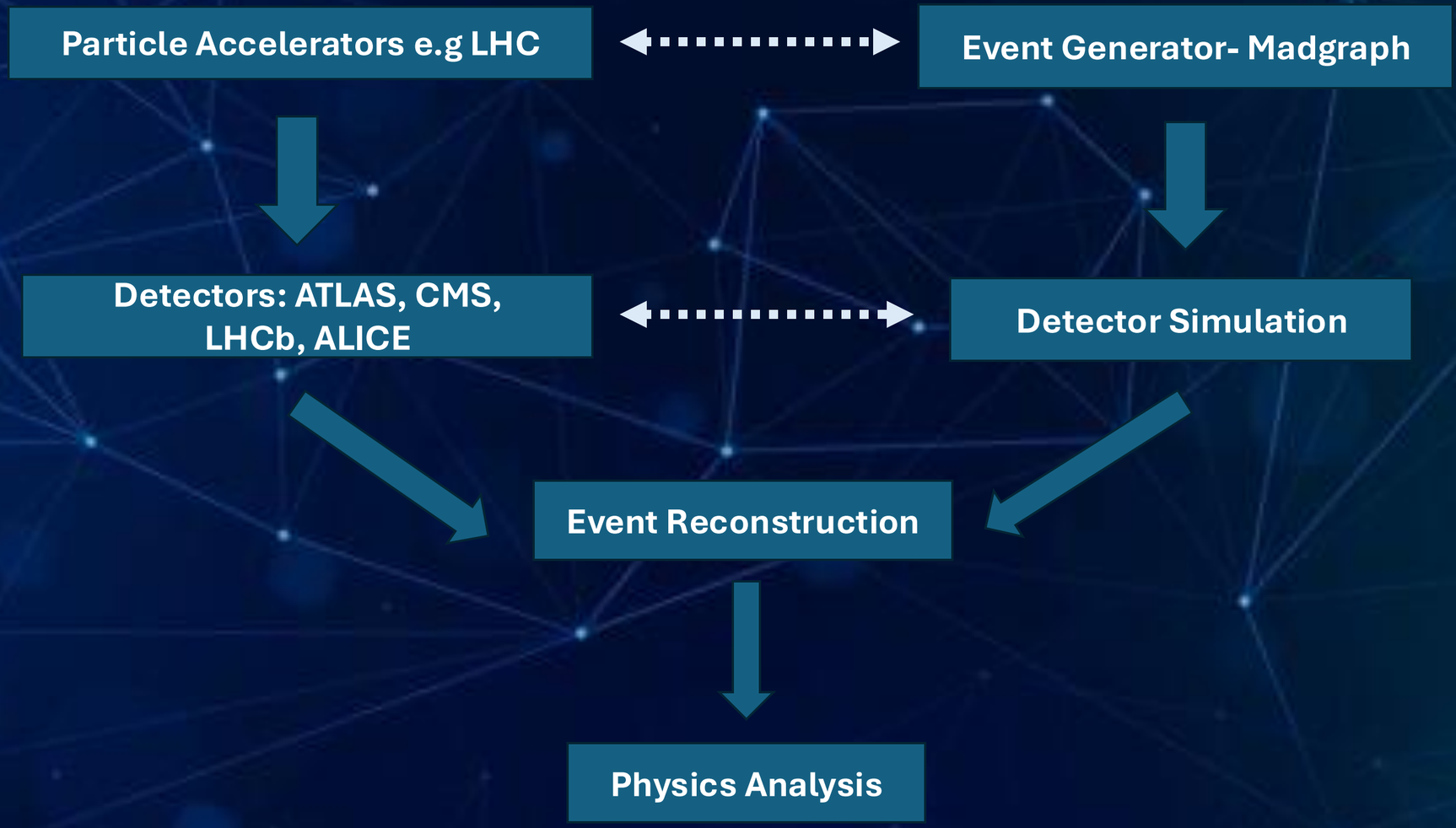
Study of power consumption and energy usage of **Madgraph** event generators



Using parallel **GPU** architecture

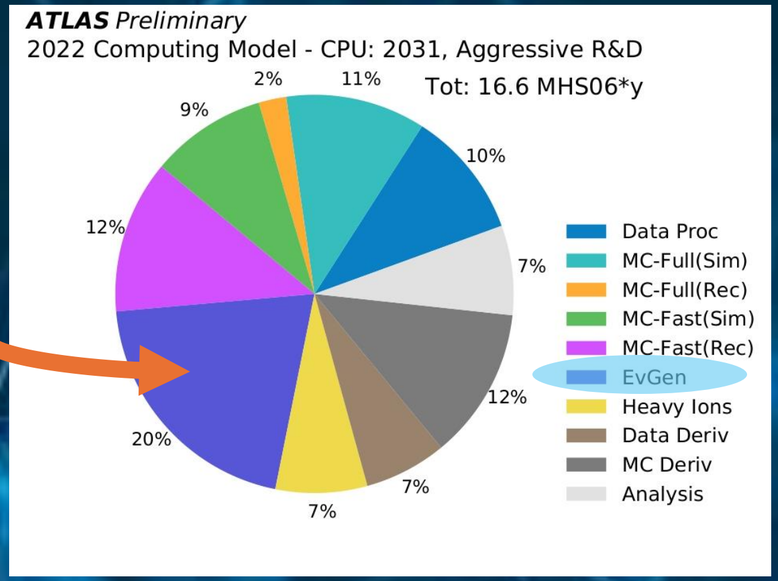
Vectorized CPU architecture like **AVX2 and AVX512**

Big Picture


Particle Accelerators e.g LHC
Event Generator- Madgraph
Detectors: ATLAS, CMS, LHCb, ALICE
Detector Simulation
Event Reconstruction
Physics Analysis

MOTIVATION

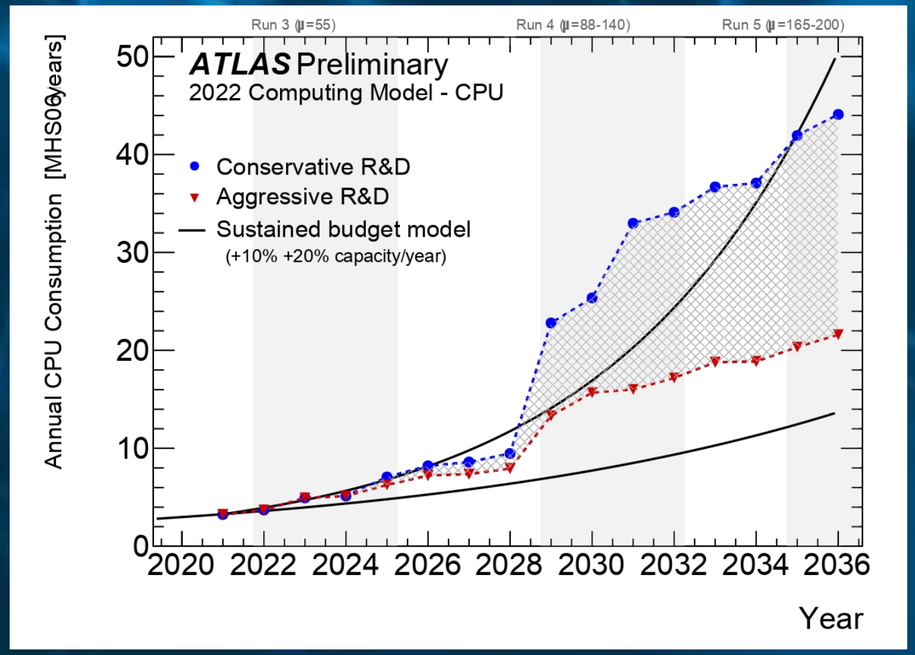
- Monte Carlo generators like MadGraph use 20% of computing resources
- More CPU or alternative hardware is needed
- Consider environmental impact



[ATLAS HL-LHC Computing Conceptual Design Report,2022]

Approach

- Use SIMT GPUs like Nvidia and AMD
- Use SIMD CPUs like AVX2 and AVX-512



STEPS

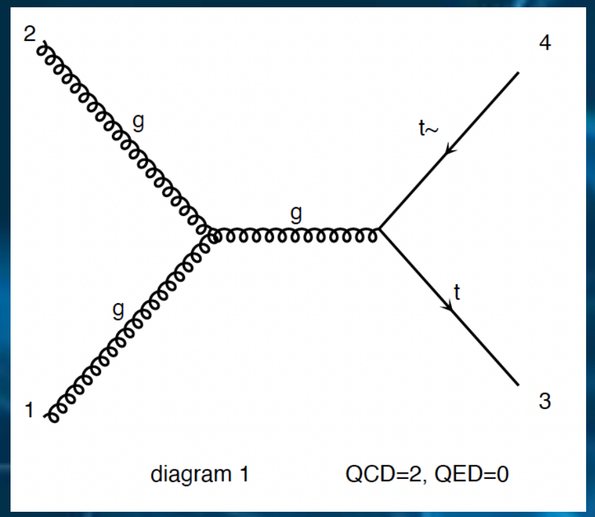
1. Single Thread Madevent:

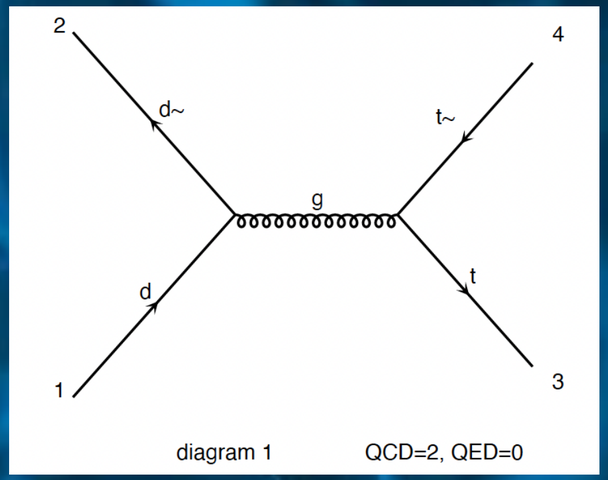
- Madevent_gpu: to run processes on GPU
- Madevent_simd: to run processes on vector CPUs
- Runs the madevent generator for a single scattering sub-process

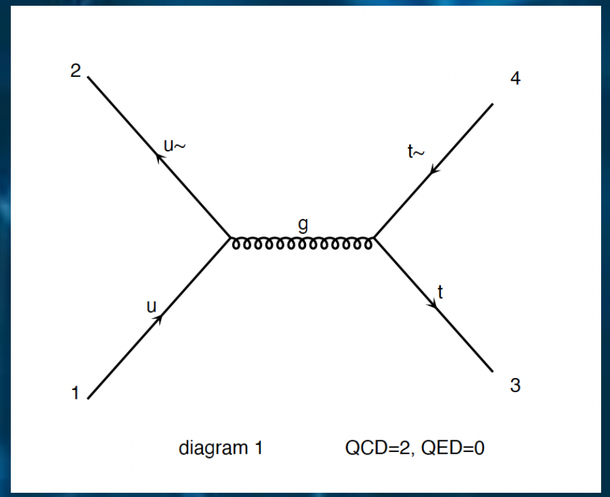
```
import model sm
generate p p > t t~
output madevent_gpu PROC_ppttx_gpu
launch PROC_pp_ttx_gpu
set cudacpp_backend cuda
```

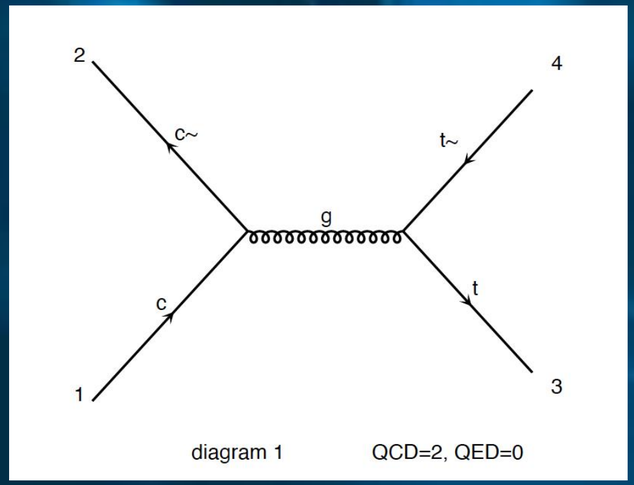
2. WITH GRIDPACK: (Used by Experiments)

- Pre-compiled package
- Runs on multiple sub-processes

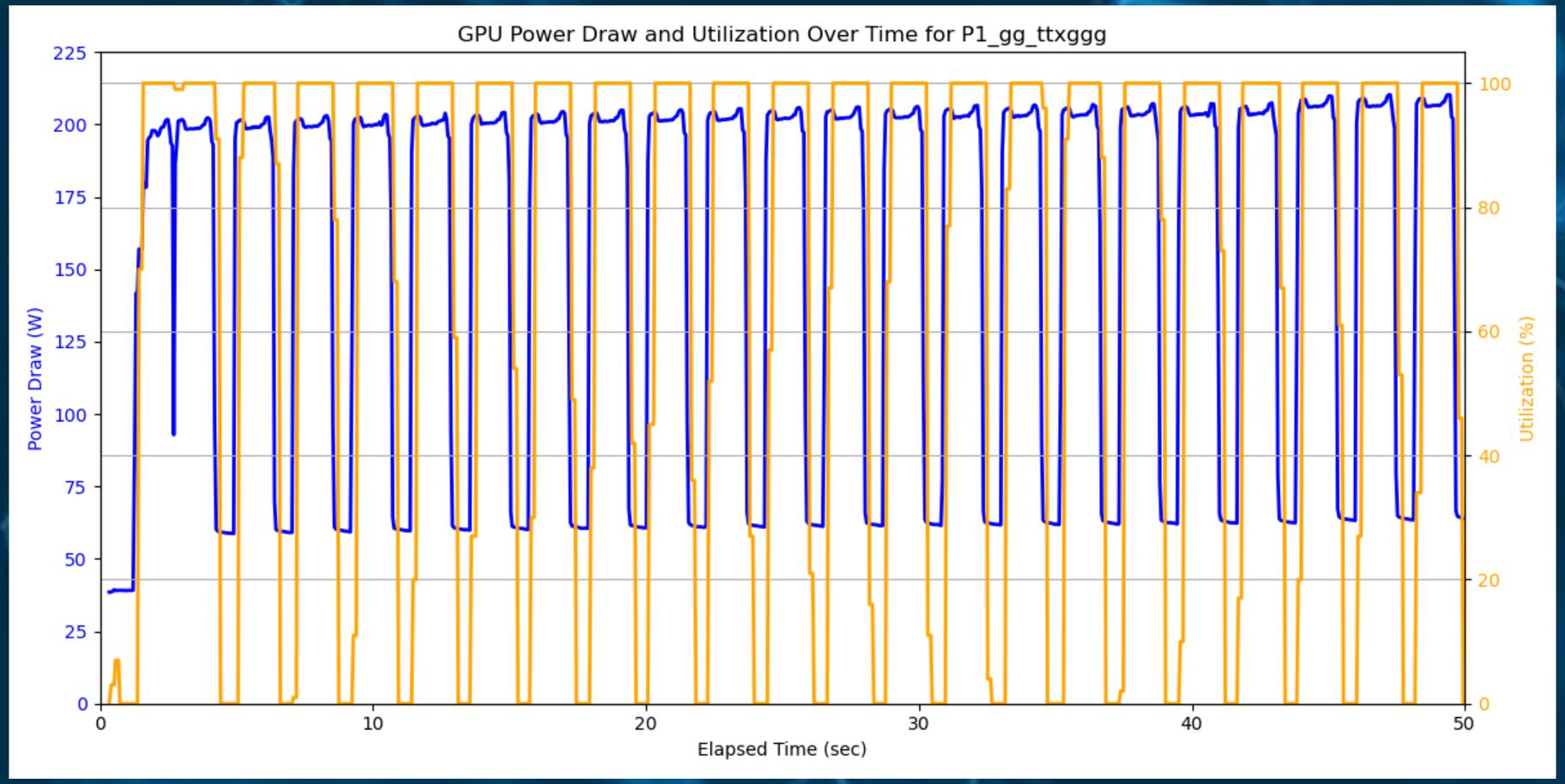








GPU Power and Utilization

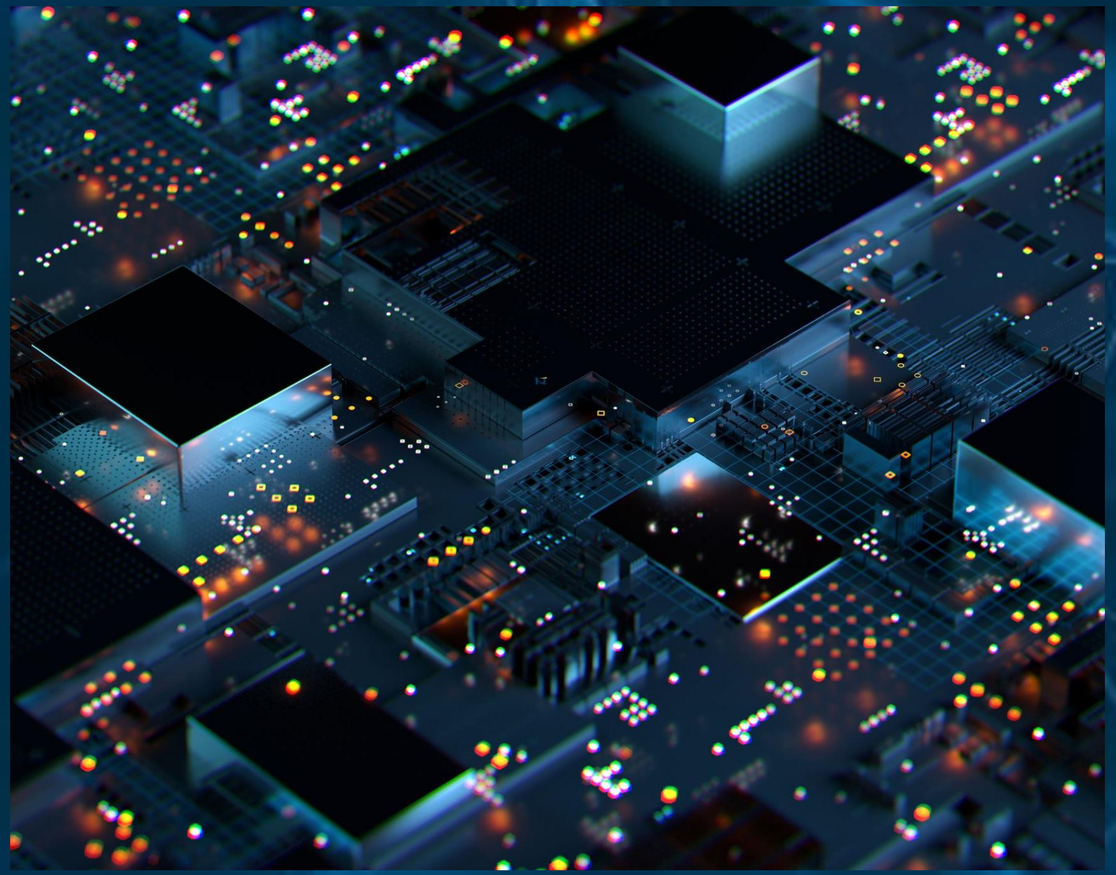


Single thread run done using Nvidia V100 GPU

Reliable Power Measurements on Gridpacks

- Use Cluster – CISM
- Execute commands via Slurm (with --exclusive flag)
- Generate gridpacks for various hardware and processes
  - Hardware: cuda, fortran, cppavx2
  - Node : mb-mil101
  - Processes: g g > t t~ ng, u u~ > e+ e- ng
- Developed automated scripts for gridpack generation and job launching
- Run the run.sh on a single node for:
  - Different values of threads and event number
  - Log start and end times of each execution

Future Steps



- Record the power consumption of GPU and CPU on gridpacks and produce plots
- Calculate energy by integrating power over time
- Calculate the CO2 emissions from the energy https://app.electricitymaps.com/map/72h/hourly

Thank you!

Email | raisa.rahman.richi@cern.ch

LinkedIn | Raisa Rahman Richi

Website | openlab.cern

Scan Me!

CERN openlab

Backups

Generating Gridpacks

```
genfile="generate_${process_name}_$backend"

cat > "$genfile" <<EOF
generate $process
EOF

if [ $backend == "cuda" ]; then
    echo "output madevent_gpu ${process_name}_$backend -nojpeg"  >> $genfile
else
    echo "output madevent_simd ${process_name}_$backend -nojpeg" >> $genfile
fi

echo "launch" >> $genfile
echo "set gridpack True" >> $genfile
echo "set cudacpp_backend $backend" >> $genfile

# Run MadGraph
../bin/mg5_aMC $genfile

mkdir -p gridpack_${backend}_${process_name}
mv ${process_name}_$backend/run_01_gridpack.tar.gz gridpack_${backend}_${process_name}/
echo "Gridpack moved to gridpack_${backend}_${process_name}/"

echo "Ending full script at $timestamp"
```

Launching with Sbatch

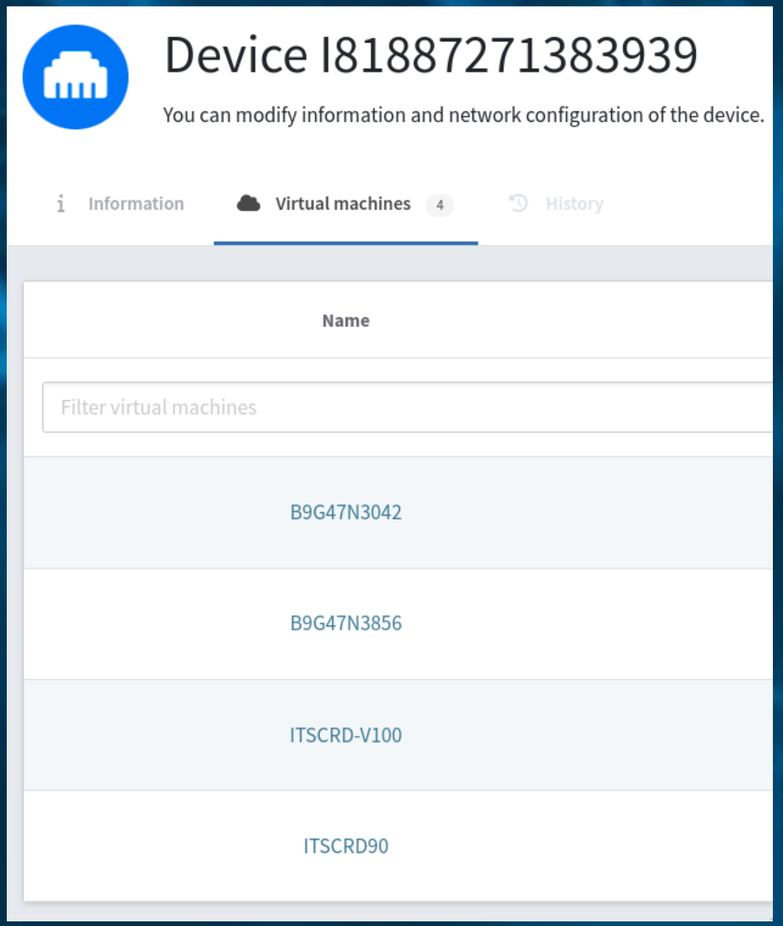
```
for process in "${processes[@]}"; do
    process_name=$(echo "$process" | xargs python -c "import sys; print(''.join(sys.argv[1:]).replace('>','_').replace('~','x').replace(' ','_'))")
    for backend in $backends; do
        echo "Submitting job for process: ${process_name} backend: $backend"
        if [ $backend == "cuda" ]; then
        sbatch --partition=gpu --exclusive --gres=gpu:1 --job-name=${process_name}_${backend} --output=logs/${process_name}_${backend}_%j.out producing_gridpack.sh $backend $process
        else
        sbatch --partition=gpu --job-name=${process_name}_${backend} --output=logs/${process_name}_${backend}_%j.out producing_gridpack.sh $backend $process
        fi
    done
done
```

```
# Define backends
backends="cuda fortran cppavx2"
processes=('g g > t t~' 'g g > t t~ g' 'g g > t t~ g g' 'u u~ > e+ e-' 'u u~ > e+ e- g' 'u u~ > e+ e- g g')
n_events=10000
sleep_time=200

nvidia-smi
cat /proc/cpuinfo

for process in "${processes[@]}"; do
    process_name=$(echo "$process" | xargs python -c "import sys; print(''.join(sys.argv[1:]).replace('>','_').replace('~','x').replace(' ','_'))")
    for backend in $backends; do
        echo "=== Running for backend: $backend and process: $process_name ==="
        cd /home/ucl/cp3/rrichi/MG5_aMC_v3_6_3/work_dr/gridpack_${backend}_${process_name}/
        tar xzf run_01_gridpack.tar.gz
        for p in 1 2 4; do
        echo "Sleeping for $sleep_time s before running $backend with p: $p, for process: $process_name"
        sleep $sleep_time
        echo "Starting running with p: $p, for process: $process_name  with $backend at $(date +"%Y-%m-%d_%H-%M-%S")"
        ./run.sh -p $p $n_events 23
        echo "Ending running with p: $p, for process: $process_name with $backend at $(date +"%Y-%m-%d_%H-%M-%S")"
        done
    done

    cd ..
done
```

CERN openlab

CPU- Reliable?

No Because Shared!



CERN openlab

What is MadGraph?

Matrix Element Event Generator

Simulates Particle Collision

User defines initial and final particles

Generates Feynman Diagrams

Calculates physical quantities like cross sections

WHY MADGRAPH?

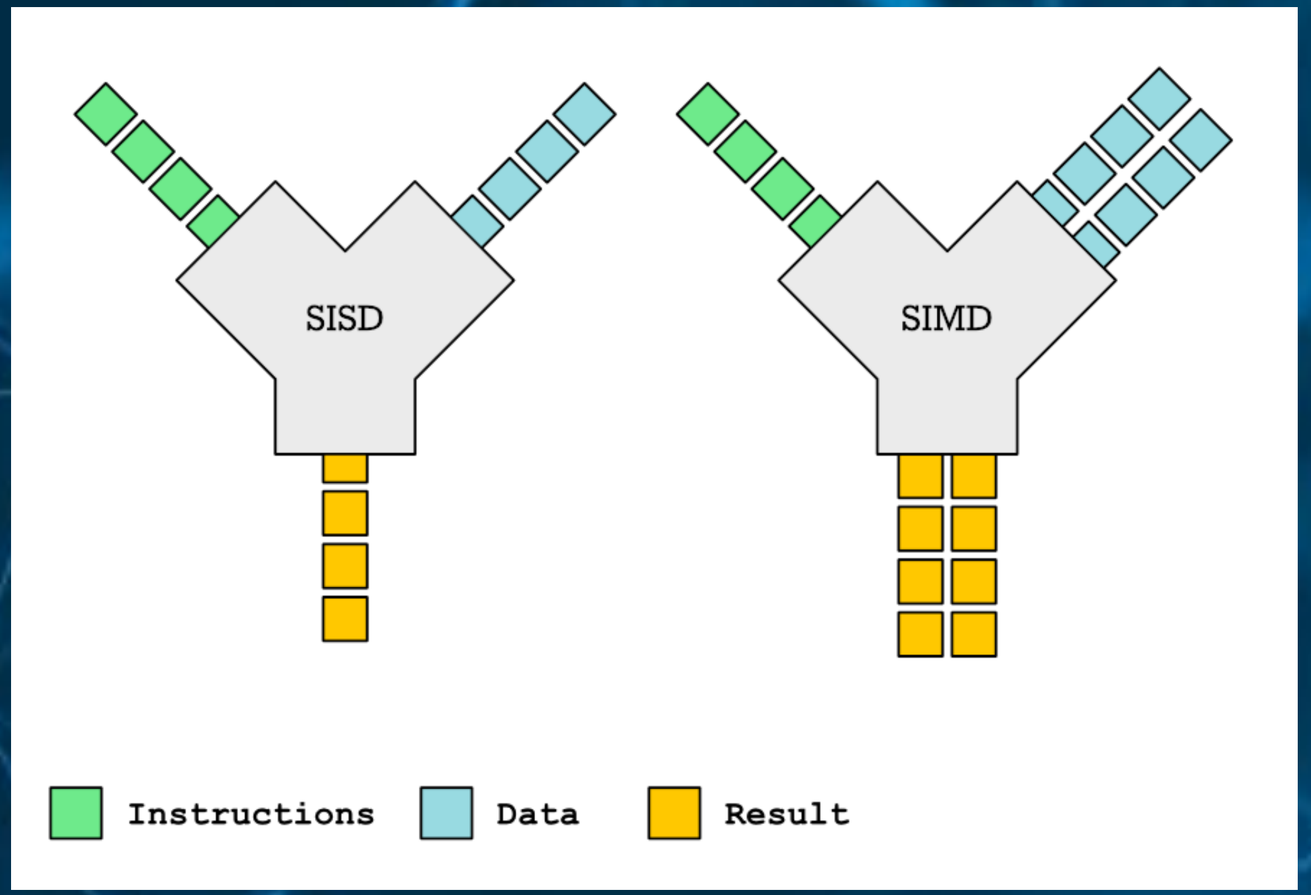
- **Vector CPUs & GPUs** are hard to utilize in HEP due to complex software workflows
- **MC detector simulations** involve **stochastic branching**, limiting data-parallel (SIMD) processing
- **Matrix Element Event Generators (like MadGraph)** compute scattering amplitudes at **independent phase-space points**. This makes them **ideal for SIMD & GPU acceleration**
- **Leverage hardware accelerators** to reduce compute time & energy consumption

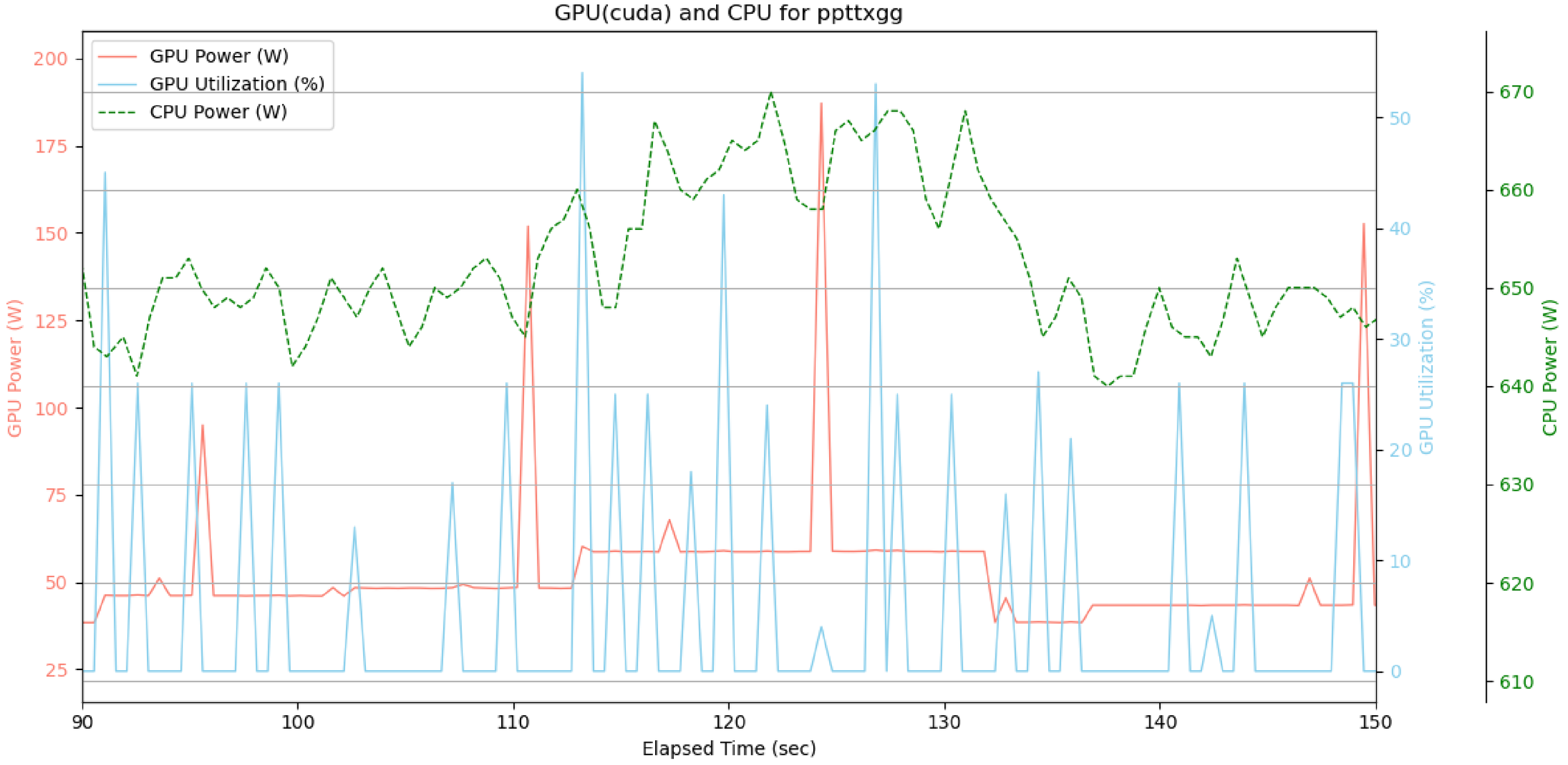
SISD (Single Instruction Single Data):

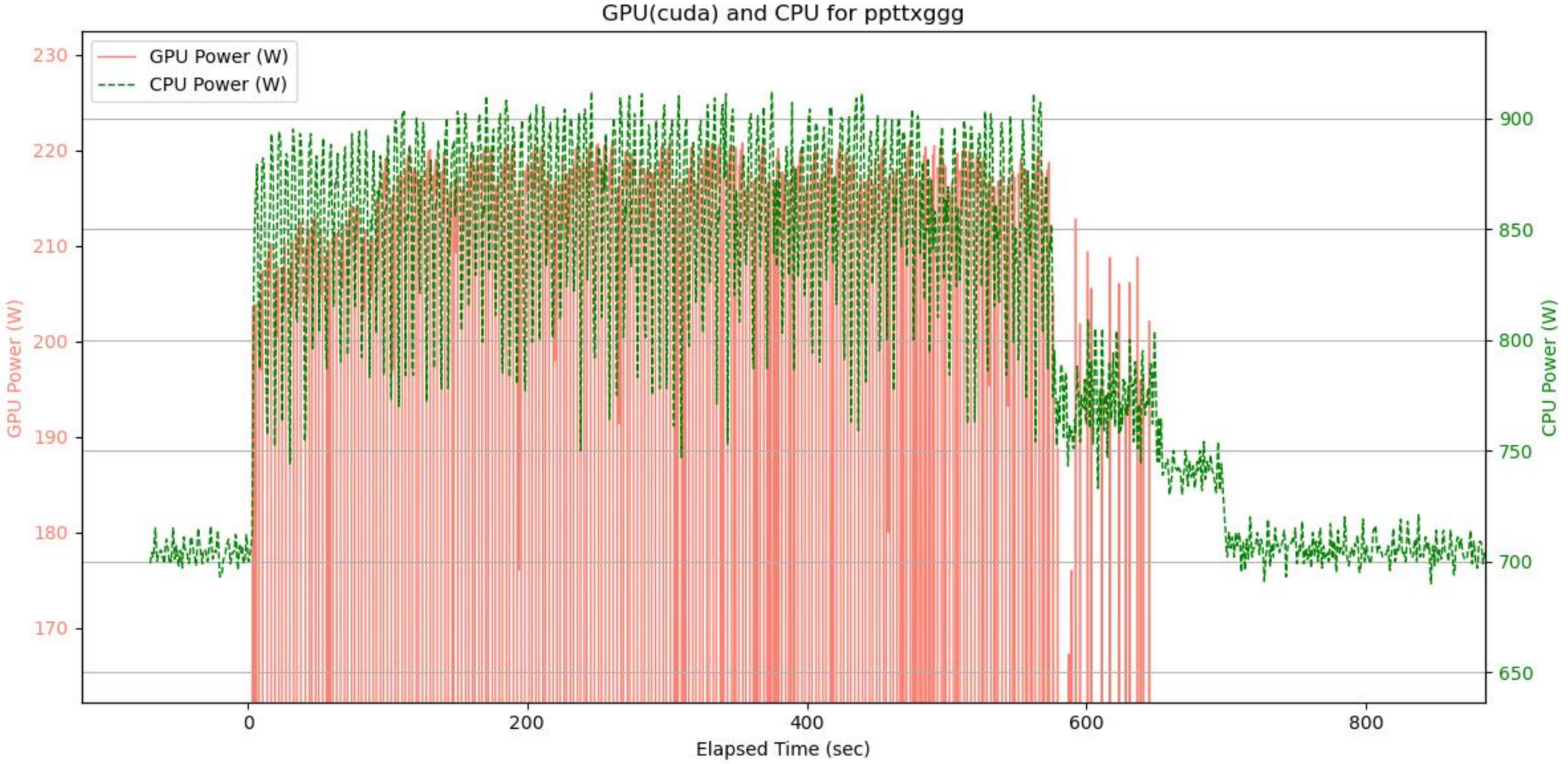
- Single processor executes one instruction at a time
- Requires more execution time
- Cannot scale well with increased computational needs

SIMD (Single Instruction Multiple Data):

- Process data points in parallel
- Excels in tasks requiring repetitive operations, reducing execution time
- Ensures proper memory alignment for faster execution




GPU(cuda) and CPU for ppttxgg
GPU Power (W)
GPU Utilization (%)
CPU Power (W)
GPU Power (W)
GPU Utilization (%)
CPU Power (W)
Elapsed Time (sec)
200
175
150
125
100
75
50
25
50
40
30
20
10
0
670
660
650
640
630
620
610
90
100
110
120
130
140
150


GPU(cuda) and CPU for ppttxggg
GPU Power (W)
CPU Power (W)
230
220
210
200
190
180
170
900
850
800
750
700
650
0
200
400
600
800
Elapsed Time (sec)
GPU Power (W)
CPU Power (W)