This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

Matrix Element Calculations on the GPU

Joshua Isaacson, Enrico Bothmann, Walter Giele, Stefan Höche, Max Knobbe

Based on arxiv:2106.06507

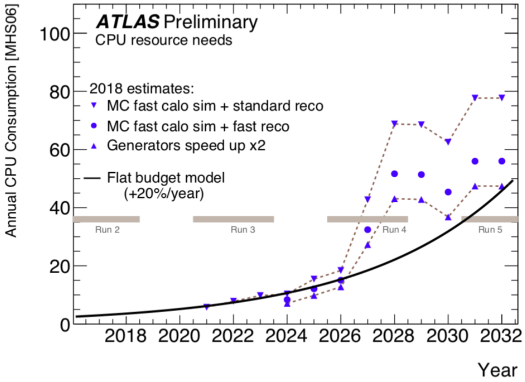
ML4Jets 2021

7 July 2021

FERMILAB-SLIDES-21-055-T

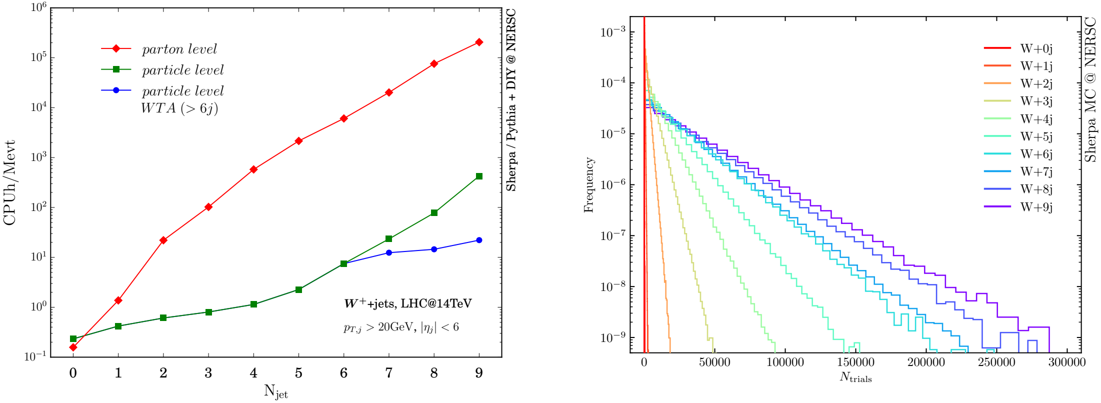
Motivation

- LHC requires large number of Monte Carlo events
- Due to CPU costs, MC statistics will become significant uncertainty



[ATLAS]

Motivation



[S. Höche, S. Prestel, H. Schulz, 1905.05120]

- Time to generate an event dominated by hard process not shower
- Large computational cost for unweighting at high multiplicity

ML Works to Reduce Cost

Phase Space Generation from Samples

- Requires a large sample before training
- GAN and VAE based [1707.0028, 1901.00875, 1901.05282, 1903.02433, 1907.03764, 1909.01359, 1909.04451, 1912.08824, 2008.06545, 2008.08558, etc.]

Phase Space Generation from Random Numbers

- Generates events as needed
- Normalizing flow based [2001.05478, 2001.05486, 2001.10028, 2104.04543]
- See talk from Timo Janßen from yesterday for more details

ML Works to Reduce Cost

Phase Space Generation from Samples

- Requires a large sample before training
- GAN and VAE based [1707.0028, 1901.00875, 1901.05282, 1903.02433, 1907.03764, 1909.01359, 1909.04451, 1912.08824, 2008.06545, 2008.08558, etc.]

Phase Space Generation from Random Numbers

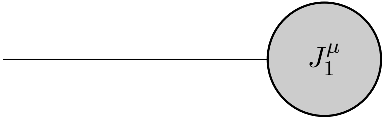
- Generates events as needed
- Normalizing flow based [2001.05478, 2001.05486, 2001.10028, 2104.04543]
- See talk from Timo Janßen from yesterday for more details

Both approaches will benefit from improved event generation time

Recursive Matrix Element Generation

Brends-Giele Recursion

- Reuse parts of calculation
- Most efficient for high multiplicity
- Reduces amplitude computations from $\mathcal{O}\left(n!\right)$ to $\mathcal{O}\left(3^n\right)$ for color-dressed and $\mathcal{O}\left(n^3\right)$ for color-ordered.
- $A(1,\ldots,n) = J_\mu(n)p_{1,n}^2 J^\mu(1,\ldots,n-1)$

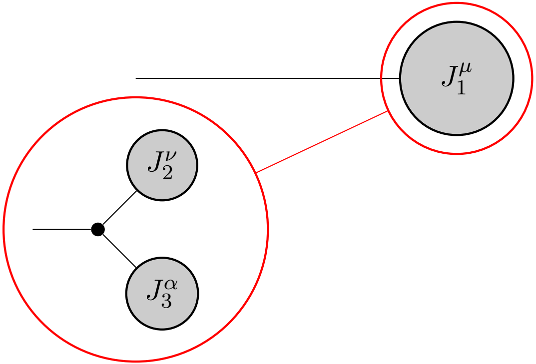


[Nucl. Phys. B306(1988), 759]

Recursive Matrix Element Generation

Brends-Giele Recursion

- Reuse parts of calculation
- Most efficient for high multiplicity
- Reduces amplitude computations from $\mathcal{O}(n!)$ to $\mathcal{O}(3^n)$ for color-dressed and $\mathcal{O}(n^3)$ for color-ordered.
- $A(1,\ldots,n) = J_\mu(n) p_{1,n}^2 J^\mu(1,\ldots,n-1)$



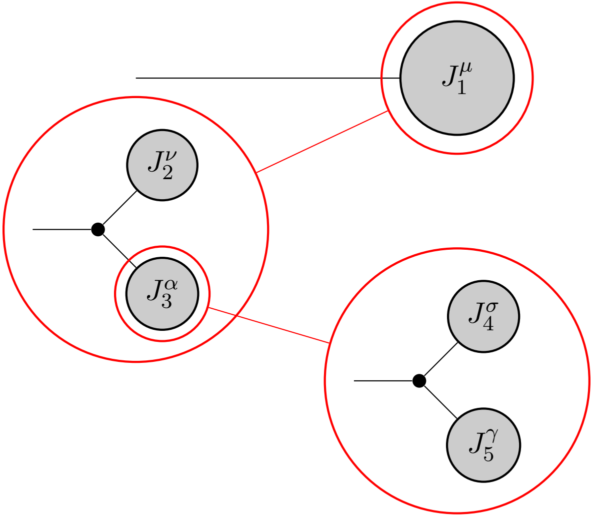
[Nucl. Phys. B306(1988), 759]

Recursive Matrix Element Generation

Brends-Giele Recursion

- Reuse parts of calculation
- Most efficient for high multiplicity
- Reduces amplitude computations from $\mathcal{O}\left(n!\right)$ to $\mathcal{O}\left(3^n\right)$ for color-dressed and $\mathcal{O}\left(n^3\right)$ for color-ordered.
- $A(1,\ldots,n) = J_\mu(n)p_{1,n}^2 J^\mu(1,\ldots,n-1)$

[Nucl. Phys. B306(1988), 759]

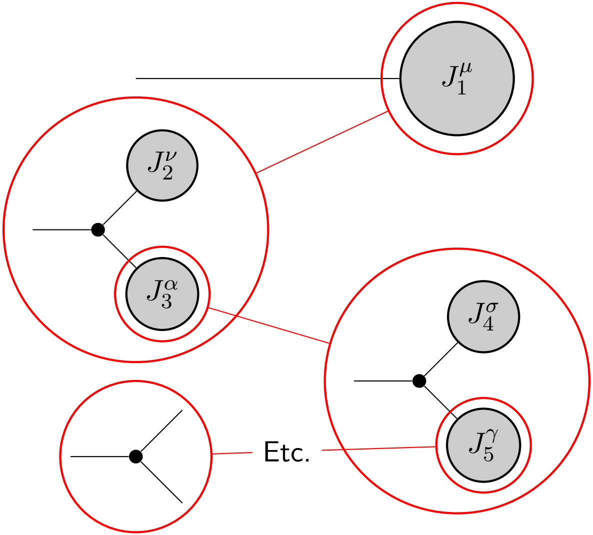


Recursive Matrix Element Generation

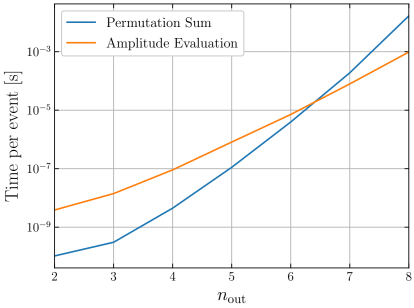
Brends-Giele Recursion

- Reuse parts of calculation
- Most efficient for high multiplicity
- Reduces amplitude computations from $\mathcal{O}(n!)$ to $\mathcal{O}(3^n)$ for color-dressed and $\mathcal{O}(n^3)$ for color-ordered.
- $A(1,\ldots,n) = J_\mu(n)p_{1,n}^2 J^\mu(1,\ldots,n-1)$

[Nucl. Phys. B306(1988), 759]



Color-Ordered vs. Color-Dressed



Scaling for Color-Ordered calculations

Color-Ordered (CO):

- Requires $(n-2)!^2$ color coefficients given by: $C_{\vec{\sigma}\vec{\sigma}'} = \sum_{a_1 \ldots a_n} (F^{a_{\sigma_2}} \ldots F^{a_{\sigma_{n-1}}})_{a_1 a_n} (F^{a_{\sigma'_2}} \ldots F^{a_{\sigma'_{n-1}}})^*_{a_1 a_n}$
- Need to sum over all permutations to obtain full amplitude

Color-Dressed (CD):

- Color summed at each vertex. No need to sum over permutations
- Can sample color to Monte-Carlo the color sum
- Need to store color information of the gluons at each vertex

Why a GPU Implementation?

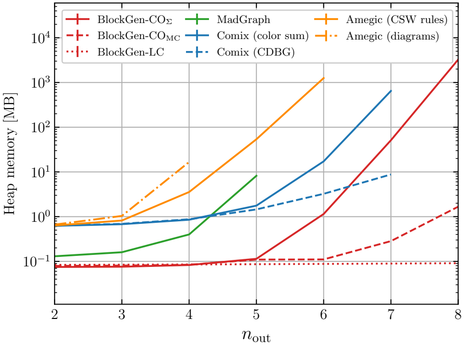
Next-Gen Supercomputer Aurora:

[https://alcf.anl.gov/aurora]

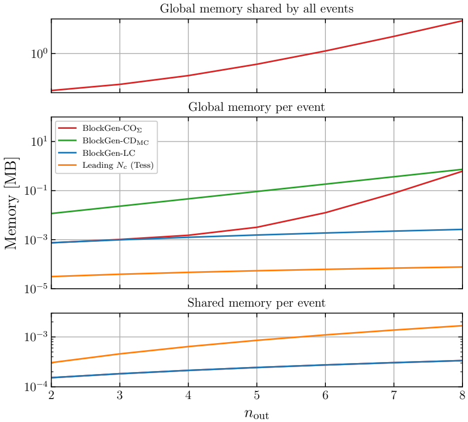
- Event generation is trivially parallelizable
- Aurora Compute Nodes:
  - 2 Intel Xeon processors
  - 6 Xeon arch-based GPUs
  - Unified Memory
- Take advantage of modern supercomputer setups
- ML is already on GPUs, only missing piece is event generation

Memory Requirements

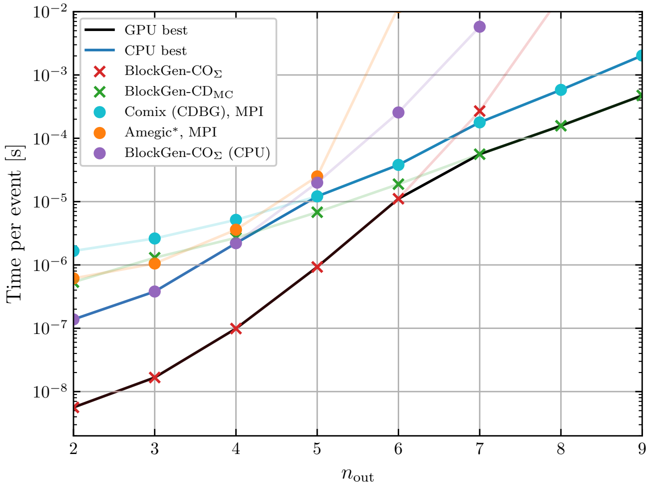
CPU Memory



GPU Memory

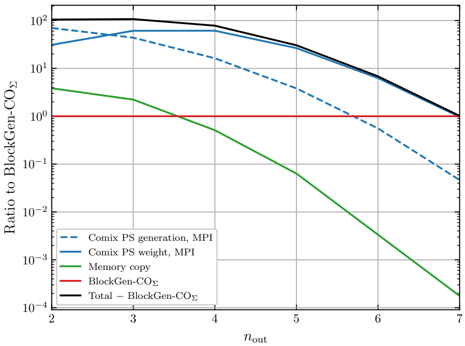


Timings



- CPU:
  Intel® Xeon®
  E5-2650 v2 8-core
  (2.60 GHz, 20 MB cache).
- GPU: NVIDIA V100
- CPUs are run with MPI with 16 threads to supply realistic chip-to-chip comparison

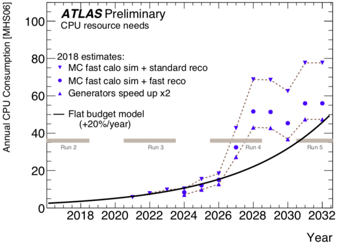
Future Steps



- Implement quarks and massive vector Bosons
- Develop hybrid calculation approach
- GPU Phase Space generator
  - Generate PS with cuts on CPU
  - Calculate PS weight and ME on GPU
  - Need to ensure memory transfer is not the bottleneck

Conclusions

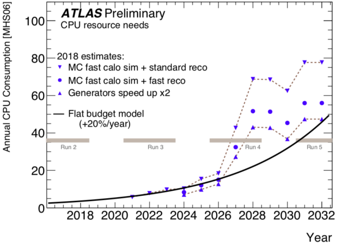
Cost of Event Generation:



- Matrix elements most expensive

Conclusions

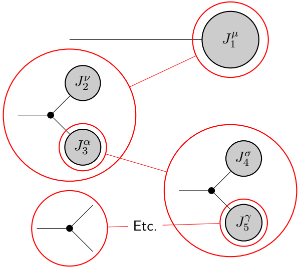
Cost of Event Generation:



- Matrix elements most expensive

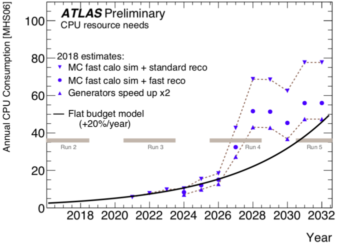
Brends-Giele:

- Optimal generation
- Event generation trivially parallelizable



Conclusions

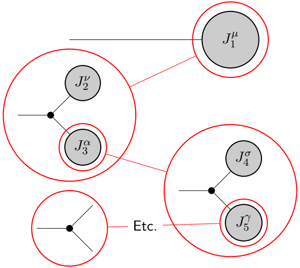
Cost of Event Generation:



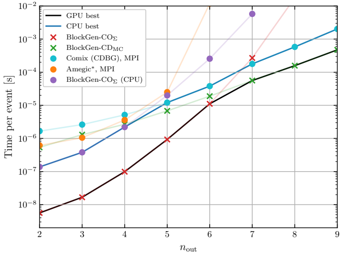
- Matrix elements most expensive

Brends-Giele:

- Optimal generation
- Event generation trivially parallelizable



Results:



- Speedup of approximately a factor of 10
- BlockGen-CO$_\Sigma$ is best for $n_{out} < 7$