Madgraph4gpu progress and WIP: bridge and libraries, clang-format, performance numbers and alpaka tests

Andrea Valassi (CERN IT-SC-RD)

Madgraph on GPU development meeting, 7th March 2022
https://indico.cern.ch/event/1131725
Overview of progress and WIP – last month

• Completed stateful bridge, shared libraries, Fortran standalone test

• Completed reformatting of epochX/cudacpp using clang format

• WIP on porting code generation from 2.7.0_gpu to 3.1.1_lo_vector
  – Done: moved to 3.1.1 as my default upstream version (but no change observed?!)
  – To do (me? Olivier?): integrate with Olivier’s multichannel interface

• Summary tables with performance numbers
  – AVX512/zmm on Julich (Juwels cluster login nodes)
  – CUDA vs Alpaka comparison

• AOB
  – ongoing discussion with Stefan on alphas
  – later on? resume work on kernel splitting
Fortran bridge and shared libraries

- PR #367 now completed and merged (“soldering and plumbing”!)
  - Fortran interface for stateful Bridge (pass Bridge pointer as INTEGER*8) #329
  - Cleanup of Makefiles (better structure and documentation) #362
  - Shared libraries (extend #361): libmg5amc_cxx.so and libmg5amc_cu.so
    - Using RPATH in the test executables ($ORIGIN on Linux, relative paths to be fixed on Mac #375)

- **Standalone Fortran test is fully implemented and integrated in the CI**
  - Fortran PROGRAM with Fortran momenta and ME arrays: fcheck.exe and fgcheck.exe
  - Delegate ME calculation (momenta→MEs) to Bridge from C++/CUDA lib (as in MadEvent)
  - Delegate random+Rambo (no input →momenta) to Bridge-like C++ Sampler
  - CI test (‘make check’):
    - expect same results from check.exe (C++) and fcheck.exe (Fortran + C++)
    - expect same results from gcheck.exe (CUDA) and fgcheck.exe (Fortran + CUDA)
Clang format

• As suggested by Stefan: I chose a .clang-format and applied it to epochX/cudacpp
  – Please give your feedback if you do not like my personal choice of style!... (PR #388)
  – Only very few /* clang-format off */ were necessary
  – Clang-format works quite well, but still has some limitations (e.g. __device__ is unknown)
  – Status: all code committed to epochX/cudacpp complies with the chosen format

• The cudacpp plugin produces coda that complies upfront with .clang-format
  – NB: code generation does NOT call clang-format, it produces formatted code by itself!
  – My code generation scripts however check explicitly that clang format is respected

• Funny side effects on performance
  – I removed a few parentheses in HelAmps.h to work around clang features (bugs?)
  – This streamlining ended up involving fewer operations, so slightly faster code ☺
What’s next for Fortran integration?

• Resync CUDA/C++ with Olivier’s latest developments in Fortran
  – Done: ported my code generation from 2.7.0 to Olivier’s latest 3.1.1 branch #360
  – To do (AV): clean up codegen plugin to avoid breaking the MG5aMC python (#341)
    • Waiting for a reproducer from Olivier to debug this issue – not sure if this is strictly needed or not
  – To do (AV? OM?): modify cudacpp to add extra parameter for single-diagram enhancement
    • Waiting for clarifications from Olivier – probably he will do it and was waiting for me on other PRs 😊
  – To do: test the full chain Fortran + CUDA/CPP
    • Personally I would suggest having a runtime switch, use Fortran MEs or use CUDA/C++ MEs
      – Easier comparison of physics results, easier comparison of performances

• Running of alphas QCD coupling #373 (→Stefan)
  – Discussions ongoing with SR, OM, AV

• ... anything else?
AVX512/zmm on Juwels Cluster login nodes

- I described this two weeks ago already – today I put the numbers on a slide! PR #381
  - https://github.com/valassi/madgraph4gpu/blob/a50980a219ecf1eb2442da89b1473563f4afd
    dfc/epochX/cudacpp/tput/summaryTable_juwels.txt

Double precision
Speedup with respect to no SIMD (none):
AVX2 ~ 4.0
AVX512/ymm ~ 4.5
AVX512/zmm ~ 7.7

<table>
<thead>
<tr>
<th>CUO/none</th>
<th>gtt</th>
<th>gtttt</th>
<th>gttttt</th>
<th>gtttgg</th>
<th>gttttgg</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2048/256/12]</td>
<td>[2048/256/1]</td>
<td>[64/256/1]</td>
<td>[64/256/1]</td>
<td>[1/256/1]</td>
<td></td>
</tr>
<tr>
<td>CPP/none</td>
<td>2.17e+06</td>
<td>2.62e+05</td>
<td>3.35e+04</td>
<td>2.39e+03</td>
<td>1.00e+02</td>
</tr>
<tr>
<td>CPP/sse4</td>
<td>4.01e+06</td>
<td>4.36e+05</td>
<td>6.22e+04</td>
<td>4.59e+03</td>
<td>1.80e+02</td>
</tr>
<tr>
<td>CPP/avx2</td>
<td>8.26e+06</td>
<td>8.57e+05</td>
<td>1.34e+05</td>
<td>1.06e+04</td>
<td>4.04e+02</td>
</tr>
<tr>
<td>CPP/512y</td>
<td>8.48e+06</td>
<td>9.51e+05</td>
<td>1.45e+05</td>
<td>1.15e+04</td>
<td>4.51e+02</td>
</tr>
<tr>
<td>CPP/512z</td>
<td>1.10e+07</td>
<td>9.39e+05</td>
<td>2.09e+05</td>
<td>1.96e+04</td>
<td>7.70e+02</td>
</tr>
</tbody>
</table>

Double precision
Speedup with respect to no SIMD (none):
AVX2 ~ 7.5
AVX512/ymm ~ 8.3
AVX512/zmm ~ 14.4

<table>
<thead>
<tr>
<th>CUO/none</th>
<th>gtt</th>
<th>gtttt</th>
<th>gttttt</th>
<th>gttttgg</th>
<th>gttttgg</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2048/256/12]</td>
<td>[2048/256/1]</td>
<td>[64/256/1]</td>
<td>[64/256/1]</td>
<td>[1/256/1]</td>
<td></td>
</tr>
<tr>
<td>CPP/none</td>
<td>2.27e+06</td>
<td>2.72e+05</td>
<td>3.38e+04</td>
<td>2.50e+03</td>
<td>1.09e+02</td>
</tr>
<tr>
<td>CPP/sse4</td>
<td>8.13e+06</td>
<td>8.71e+05</td>
<td>1.11e+05</td>
<td>9.42e+03</td>
<td>4.04e+02</td>
</tr>
<tr>
<td>CPP/avx2</td>
<td>1.75e+07</td>
<td>1.52e+06</td>
<td>2.70e+05</td>
<td>2.15e+04</td>
<td>8.18e+02</td>
</tr>
<tr>
<td>CPP/512y</td>
<td>1.88e+07</td>
<td>1.75e+06</td>
<td>2.99e+05</td>
<td>2.28e+04</td>
<td>9.08e+02</td>
</tr>
<tr>
<td>CPP/512z</td>
<td>2.44e+07</td>
<td>1.97e+06</td>
<td>4.35e+05</td>
<td>4.03e+04</td>
<td>1.57e+03</td>
</tr>
</tbody>
</table>
Alpaka – performance tests (on Nvidia V100)

• The Alpaka codegen plugin is now fully functional - thanks a lot to David Smith!
  – This is based on the golden_epochX4 tag (older code base than current cudacpp)

• I did a few performance tests and integrated this with my throughput scripts
  – Also a few minor functional fixes – adapt to changes in MadgraphTest.h since goldenX4

• I would repeat these tests also on sycl and kokkos if you agree?
  • About Alpaka: should the plugin be ported to the latest cudacpp or not?