Status of PRs towards a release
(plus CMS DY+jets and a few other things)

Andrea Valassi (CERN)

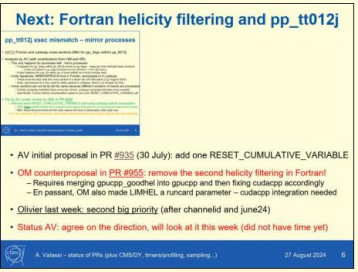
(describing also work done with and/or by Olivier - thanks!)

Madgraph on GPU development meeting, 17th September 2024
https://indico.cern.ch/event/1355161

(previous update was last week on September 3 – only mentioning changes since then)

gpucpp_for360 and master_for360

Next: Fortran helicity filtering and pp_tt012j



- Olivier two weeks ago: second big priority (after channelid/june24)
- Status AV: PR #986 (branch goodhel) based on Olivier's 955 is ready and tested
  - With its mg5amcnlo counterpart #137 (branch valassi_goodhel)
  - Note: this includes many bits of the upgrade to 3.6.0 (but Olivier has more... next slide)
- WIP now: update this to the latest gpucpp/master including june24
  - First issue: merge conflicts (maybe better solved by Olivier's gpuccp_for360, next slide)
  - Second issue: with my WIP version, I get a 0.1% **cross-section mismatch** #991

MERGED!

Next: Olivier's gpucpp_for360

- Olivier this week: third big priority (after channelid/june24 and goodhel)
  - Our cudacpp release is meant to be in v3.6.0...
- This is a series of patches that are needed on top of june24 and goodhel
  - They fix issues resulting from the update to 3.6.0 (in goodhel) and the interplay with the rest
- Status: WIP WIP
  - Done AV/OM: merged gpucpp_goodhel #138 into gpucpp_for360
  - To do: more conflict resolution, on the mg5amcnlo side
  - To do: and then, the integration with the cudacpp side

- Two developments mainly by OM: 'goodhel' helicity filtering and 'for360' v360 fixes
  - These two development were fused into the 'for360' branches, which was finally merged
    - #140 into gpucpp for mg5amcnlo (cherry-picking from #139 and #126, now closed)
    - #992 into master for madgraph4gpu (including #997 – and bits of #986 and #955, now closed)
  - *This completes(?) a 3-month saga of many independent branches (in parallel on 2 repos)*

Next priority: packaging (1)

A tale of two repositories (now)

mg5amcnlo
https://github.com/mg5amcnlo/mg5amcnlo

- *the* MG5AMC repo (previously launchpad)
- python framework, fortran codegen
- permissive NCSA-style license

A specific commit is in madgraph4gpu →

Important branches for GPU/SIMD work:

- **gpucpp** (the baseline: merge here!)
- gpucpp_june24 (channelid array) **MERGED**
- gpucpp_goodhel (new helicity filter) **WIP**
- gpucpp_for360 (complete 3.6.0 sync) **WIP**

madgraph4gpu
https://github.com/madgraph5/madgraph4gpu

- cudacpp plugin (cuda/c++ codegen)
- generated code, tests, results (+legacy stuff)
- more restrictive LGPL license

Includes mg5amcnlo as a git submodule

Important branches for GPU/SIMD work:

- **master** (the baseline: merge here!)
- master_june24 (...) **MERGED**
- master_goodhel (...) **WIP**
- master_for360 (...) **WIP**

Status: finally merged "june24" this week; now fixing the conflicts with "goodhel" and "for360"
Aim for a v3.6.0 release including the GPU/SIMD support... possibly by end September!?

Next priority: packaging (2)

Still a tale of two repositories (later)?

Option 1 – our assumption so far

mg5amcnlo
https://github.com/mg5amcnlo/mg5amcnlo
- *the* MG5AMC repo (NCSA-style)

Includes cudacpp as a git submodule ← in PLUGIN/CUDACPP_OUTPUT

mg5amcnlo_cudacpp *(OLD WIP AUG 2023)*
https://github.com/mg5amcnlo/mg5amcnlo_cudacpp
- cudacpp plugin (LGPL)

A specific commit is in mg5amcnlo

Option 2? – recent discussion AV/OM

mg5amcnlo https://github.com/mg5amcnlo/mg5amcnlo
Includes cudacpp as a subdirectory in PLUGIN/**CUDACPP_OUTPUT**

Advantages/Disadvantages?
- Option 1 gives cleaner separation; but merge conflicts with git submodules are hard
- Option 2 is easier to manage, but more monolithic; following up if licensing is ok

Next priority: packaging (3)

- I am not entirely sure which option I prefer – I ask here before doing real work...
  - Option 1 gives cleaner separation; but merge conflicts with git submodules are hard
  - Option 2 easier to manage, but more monolithic; following up (OSPO) if licensing is ok
  - (Option 3 keep madgraph4gpu and restructure it? probably better not...)

- *I have a slight preference for Option 2 however (i.e. a single repo)*
  - a specific version of cudacpp needs a ~specific version of mg5amcnlo
  - a specific version of mg5amcnlo needs a ~specific version of cudacpp
  - having them in a single repo simplifies this bi-directional dependency
    - and, again, simplifies the handling of PRs, which may be a complete mess with git submodules

- *Concrete proposal for mg5amcnlo?*
  - *mg5amcnlo has its own main branch for releases (currently branch "3.x" IIUC?)*
  - *permanently maintain our "gpucpp" branch now including PLUGIN/CUDACPP_OUTPUT*
    - *periodically merge gpucpp into 3.x (this is what other development lines do too, right?)*

- Things to do (AV), whether we go for Option 1 or Option 2
  - Prepare the move out of madgraph4gpu, including history and preserving links
  - Prepare some scripts for further resync from/to madgraph4gpu (there is still WIP there...)

Other issues towards the release

(incomplete list, random order)

Before the release:

- Understand and fix FPEs in DY+jets reported by CMS #942
- Check that results are the same with and without vector interfaces #678 (OM)
  - Understand xsec variation with vector_size (32 vs 16384) in DY+3jets #959
- Check that AMD GPUs are usable #942 – AV WIP today on LUMI (minor fixes)
- (Check that parameter cards are handled correctly #660)
- ...

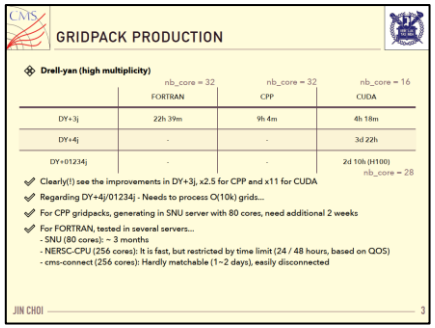
Possibly before the release (or at least before CHEP) – from CMS profiling:

- Multi-backend gridpacks PR #948
  - Event number in LHE file mismatch and “fail to reach target” #993
- Improved RDTSC timers/counters PR #962
- Gridpack python/bash profiling and more granular Fortran madevent profiling

Are the following needed before the release?

- Understand xsec mismatch (Fortran vs cudacpp) in DY+4jets reported by CMS #944

DY+3/4j speedups
(using multi-backend gridpacks)


GRIDPACK PRODUCTION

Drell-yan (high multiplicity)

	nb_core = 32 FORTRAN	nb_core = 32 CPP	nb_core = 16 CUDA
DY+3j	22h 39m	9h 4m	4h 18m
DY+4j	-	-	3d 22h
DY+01234j	-	-	2d 10h (H100) nb_core = 28

- Clearly(!) see the improvements in DY+3j, x2.5 for CPP and x11 for CUDA
- Regarding DY+4j/01234j - Needs to process O(10k) grids...
- For CPP gridpacks, generating in SNU server with 80 cores, need additional 2 weeks
- For FORTRAN, tested in several servers...
  - SNU (80 cores): ~ 3 months
  - NERSC-CPU (256 cores): It is fast, but restricted by time limit (24 / 48 hours, based on QOS)
  - cms-connect (256 cores): Hardly matchable (1~2 days), easily disconnected

JIN CHOI 3

- Jin: Fortran/C++ gridpack creation too slow
  - Cannot show event generation speedups
  - My suggestion: multi-backend gridpacks MBG
    - NB1: preliminary numbers! WIP PR #948
    - NB2: “fail to reach target” (not MBG specific?)

Essentially for DY+4j I get exactly the same trend with 500 events that I saw with 100 events

- AVX2 gives a factor 4 for MEs and overall a factor 3 speedup over Fortran
- AVX512 is not better than AVX2 (on a Silver Intel CPU; on a Gold Intel CPU I would expect maybe 8 for MEs and 6 overall faster than Fortran?)
- CUDA is a factor 200 better than Fortran for MEs and a factor 8 better than Fortran overall

```
pp_dy4j.mad/fortran/output.txt (#events: 81)
[GridPackCmd.launch] OVERALL TOTAL    21707.6095 seconds
[madevent COUNTERS]  PROGRAM TOTAL    21546.1
[madevent COUNTERS]  Fortran Overhead 1579.09
[madevent COUNTERS]  Fortran MEs      19967
------------------------------------------------------------
pp_dy4j.mad/cppnone/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    26745.1639 seconds
[madevent COUNTERS]  PROGRAM TOTAL    26584.9
[madevent COUNTERS]  Fortran Overhead 1608.51
[madevent COUNTERS]  CudaCpp MEs      24910.4
[madevent COUNTERS]  CudaCpp HEL      66.0341
------------------------------------------------------------
pp_dy4j.mad/cppsse4/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    14398.4664 seconds
[madevent COUNTERS]  PROGRAM TOTAL    14231.3
[madevent COUNTERS]  Fortran Overhead 1647.03
[madevent COUNTERS]  CudaCpp MEs      12550.6
[madevent COUNTERS]  CudaCpp HEL      33.7035
------------------------------------------------------------
pp_dy4j.mad/cppavx2/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    7335.2356 seconds
[madevent COUNTERS]  PROGRAM TOTAL    7114.43
[madevent COUNTERS]  Fortran Overhead 1683.7
[madevent COUNTERS]  CudaCpp MEs      5415.48
[madevent COUNTERS]  CudaCpp HEL      15.2596
------------------------------------------------------------
pp_dy4j.mad/cpp512y/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    6831.8971 seconds
[madevent COUNTERS]  PROGRAM TOTAL    6649.98
[madevent COUNTERS]  Fortran Overhead 1669.94
[madevent COUNTERS]  CudaCpp MEs      4966.24
[madevent COUNTERS]  CudaCpp HEL      13.8066
------------------------------------------------------------
pp_dy4j.mad/cpp512z/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    7136.2962 seconds
[madevent COUNTERS]  PROGRAM TOTAL    6958.96
[madevent COUNTERS]  Fortran Overhead 1636.28
[madevent COUNTERS]  CudaCpp MEs      5305.14
[madevent COUNTERS]  CudaCpp HEL      17.5447
------------------------------------------------------------
pp_dy4j.mad/cuda/output.txt (#events: 195)
[GridPackCmd.launch] OVERALL TOTAL    2523.7488 seconds
[madevent COUNTERS]  PROGRAM TOTAL    2234.93
[madevent COUNTERS]  Fortran Overhead 1820.36
[madevent COUNTERS]  CudaCpp MEs      97.9622
[madevent COUNTERS]  CudaCpp HEL      316.613
------------------------------------------------------------
```