MG4GPU CMS INTEGRATION STATUS

FOR CMS-MG JOINT MEETING

24.07.26

Jin Choi, Sapta Bhattacharya, Stefan Roiser, Andrea Valassi,
Olivier Mattelaer, Zenny Wettersten

CONTENTS

- **Gridpack Production**
- **Event Generation**

GRIDPACK PRODUCTION

PHYSICS PROCESS

Test Process

- Using Drell-Yan process with jet-binned configuration
  - ⇨ Most common bkg. in CMS Analyses
  - ⇨ No Madspin dep. (Not implemented in GPU yet)
  - ⇨ Growing complexity - Possible to check improvements & bottlenecks w.r.t. final state multiplicity

	DY+0j	DY+1j	DY+2j	DY+3j	DY+4j
diagrams	30	180	3120	27600	412560
processes	15	45	285	435	1455

```
import model sm-no_b_mass
set nb_core 10

define p = u d c s b u~ d~ c~ s~ b~ g
define j = p
define ell+ = e+ mu+ ta+
define ell- = e- mu- ta-
define nu = ve vm vt
define nubar = ve~ vm~ vt~

generate p p > ell+ ell- j j j j @0

output madevent_gpu DY4j_LO_5f_CUDA -nojpeg
```

DY4j_LO_5f_CUDA_proc_card.dat

```
set sde_strategy 1
set vector_size 8192
set cudacpp_backend CUDA
```

DY4j_LO_5f_CUDA_customizecards.dat

SET-UP

Test Process

- Using Drell-Yan jet-binned samples [run cards]
  - ⇨ Most common bkg. in CMS Analyses
  - ⇨ No Madspin dep. (Not implemented to GPU yet)
  - ⇨ Growing complexity - Possible to check improvements & bottlenecks w.r.t. final state multiplicity

	DY+0j	DY+1j	DY+2j	DY+3j	DY+4j
diagrams	30	180	3120	27600	412560
processes	15	45	285	435	1455

Comparison

- Cross-section: for least validation
- Timing: Using time command, compared full production time & ME integration time

SET-UP

- **Integrating MG4GPU to CMS-genproduction** [genproduction/mg4gpu]
- Based on the master branch(for RUN3 production) - updated patches for MG352/mg4gpu
- Workflow: Environment setup(e.g. CMSSW / CUDA) - download mg4gpu - apply patches
  - run ./gridpack_generation.sh
- Previously used git clone for downloading mg4gpu: large repo, takes ~ 10 min. to clone
  - ⇨ Compressed the repo in EOS area, untar the repo rather than downloading: ~ 4 min.
- Two major patches for mg4gpu side (for version mg4gpu_2024-03-14.tar.gz)

```
diff --git a/madgraph/interface/madevent_interface.py b/madgraph/interface/madevent_interface.py
index 8c509e83f..e6e7bd0dc 100755
--- a/madgraph/interface/madevent_interface.py
+++ b/madgraph/interface/madevent_interface.py
@@ -3966,7 +3966,8 @@ Beware that this can be dangerous for local multicore runs.""")
         Pdir = set([os.path.dirname(G) for G in Gdir])
         for P in Pdir:
             allG = misc.glob('G*', path=P)
-            for G in allG:
+            filG = [f for f in allG if not os.path.basename(f).startswith('Gpu')]
+            for G in filG:
                 if pjoin(P, G) not in Gdir:
                     logger.debug('removing %s', pjoin(P,G))
                     shutil.rmtree(pjoin(P,G))
```

some files start with "Gpu*" and erased when clearing some directories like G3*...

SET-UP

- **Integrating MG4GPU to CMS-genproduction** [genproduction/mg4gpu]
- Based on the master branch(for RUN3 production) - updated patches for MG352/mg4gpu
- Workflow: Environment setup(e.g. CMSSW / CUDA) - download mg4gpu - apply patches
  - run ./gridpack_generation.sh
- Previously used git clone for downloading mg4gpu: large repo, takes ~ 10 min. to clone
  - ⇨ Compressed the repo in EOS area, untar the repo rather than downloading: ~ 4 min.
- Two major patches for mg4gpu side (for version mg4gpu_2024-03-14.tar.gz)

```
diff --git a/madgraph/various/systematics.py b/madgraph/various/systematics.py
index 28eaed0..5f787de 100644
--- a/madgraph/various/systematics.py
+++ b/madgraph/various/systematics.py
@@ -169,7 +169,7 @@ class Systematics(object):
         self.orig_ion_pdf = False
         self.ion_scaling = ion_scaling
         self.only_beam = only_beam
-        if isinstance(self.banner.run_card, banner_mod.RunCardLO):
+        if self.banner.run_card.LO:
             self.is_lo = True
             if not self.banner.run_card['use_syst']:
                 raise SystematicsError('The events have not been generated with use_syst=True. Cannot evaluate systematics error on thes
```

self.banner.run_card does not work with use_syst option

ENVIRONMENTS

HPCs

- lxplus800(GPU): AMD EPYC 7313 16-core processor (AVX2 support), A100 GPU → repeatedly halted
- SNU-server: Intel(R) Xeon(R) CPU E5-2699 v3 (72 cores, AVX2 support), no GPU
  → tested FORTRAN DY+4j gridpacks
- **lxplus condor**: Randomly matched to
  w/o GPU: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (48 CPUs)
  Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (64 CPUs)
  w/ GPU: AMD EPYC 7313 16-core processor (AVX2 support), A100 GPU

Sidenotes

- For testing CPU usage in lxplus condor,
  randomly matches to the nodes with 48/64 cores + AVX2 supports
- There is 4 A100 GPU node but the gridpack production failed if there is multiple GPUs

ENVIRONMENTS

- **HPCs**
- lx
- Sl
- lx
- w
- **Si**
- Fr
- ra
- T

OpenStack project with GPU flavors in pass-through mode

This option is identical to the one described in the Projects section, except that GPU flavors will be assigned to your project. You can then launch instances with GPUs. The available flavors are:

Flavor Name	GPU	RAM	vCPUs	Disk	Ephemeral	Comments
g1.xlarge	V100	16 GB	4	56 GB	96 GB	[^1], deprecated
g1.4xlarge	V100 (4x)	64 GB	16	80 GB	528 GB	[^1]
g2.xlarge	T4	16 GB	4	64 GB	192 GB	[^1], deprecated
g2.5xlarge	T4	168 GB	28	160 GB	1200 GB	[^1]
g3.xlarge	V100S	16 GB	4	64 GB	192 GB	[^1]
g3.4xlarge	V100S (4x)	64 GB	16	128 GB	896 GB	[^1]
g4.p1.40g	A100 (1x)	120 GB	16	600 GB	-	[^1], AMD CPUs
g4.p2.40g	A100 (2x)	240 GB	32	1200 GB	-	[^1], AMD CPUs
g4.p4.40g	A100 (4x)	480 GB	64	2400 GB	-	[^1], AMD CPUs

itedly halted

XSECS

Least validation

Compatible

	FORTRAN [pb]	CPP [pb]	CUDA [pb]
DY+0j	5704 \pm 10.11	5711 \pm 1.053	5710 \pm 1.484
DY+1j	3539 \pm 8.096	3535 \pm 1.263	3536 \pm 1.442
DY+2j	2228 \pm 3.143	2236 \pm 0.503	2237 \pm 0.4618
DY+3j	1375 \pm 1.265	1387 \pm 0.3515	1385 \pm 0.3288
DY+4j	883.4 \pm 0.3813	845.8 \pm 0.21	843.8 \pm 0.2022

A bit large errors / different xsecs for FORTRAN?

FORTRAN: Original MG
CPP: Vectorized CPU
CUDA: GPU

PRODUCTION TIME

Results (full production)

	48 Intel	48 Intel, avx2	16 AMD + 1 A100 GPU
	FORTRAN	CPP	CUDA
DY+0j	7m 59s	8m 38s	8m 1s
DY+1j	9m 27s	21m 3s	17m 20s
DY+2j	21m 24s	85m 6s	71m 25s
DY+3j	293m 38s	698m 41s	274m 44s
DY+4j	19362m 13s	18509m 11s	7682m 17s
	72 cms2	64 Intel, avx2	
	13.5 days...	12.8 days...	5.3 days!

- FORTRAN / CPP run x3-4 processes in parallel
  - production time not differ much with current HPC configuration
- Can see x2-3 improvement in DY+4j: ME Integration is much faster for high multiplicity processes

PRODUCTION TIME

Results (full time)

	72 Intel cores	72 Intel cores	batch job 16 AMD cores + 1 A100 GPU
	FORTRAN	CPP	CUDA
DY+0j	11m 31s	6m 32s	8m 1s

```
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
INFO:     P0_dxsx_taptamdxsx
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
INFO:     P0_uux_epemgg
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
INFO:     P0_ddx_epemgg
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
INFO:     P0_uux_taptamgg
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
INFO:     P0_ddx_taptamgg
INFO: Building madevent in madevent_interface.py with 'FORTRAN' matrix elements
```

Compilation (ME)

```
INFO:  Idle: 1,  Running: 8,  Completed: 281 [ current time: 16h42 ]
INFO:  Idle: 0,  Running: 9,  Completed: 281 [  0.02s  ]
INFO:  Idle: 0,  Running: 6,  Completed: 284 [  3.3s  ]
INFO:  Idle: 0,  Running: 3,  Completed: 287 [  10.5s  ]
INFO:  Idle: 0,  Running: 0,  Completed: 290 [  17.7s  ]
INFO:  Idle: 0,  Running: 0,  Completed: 290 [  17.7s  ]
sum of cpu time of last step: 58m04s
```

Execution (ME)

- Improvement can only be seen in DY+4j...

PRODUCTION TIME

Results (ME integration)

	48 Intel	48 Intel, avx2	16 AMD + 1 A100 GPU
	FORTRAN	CPP	CUDA
DY+0j	2.4s	1m 5s	17.7s
DY+1j	12.1s	2m 59s	31.6s
DY+2j	38.5s	8m 1s	2m 29s
DY+3j	3h 33m	6h 30m	33m 34s
DY+4j	315h 38m	244h 24m	108h 46m

72 cms2 64 Intel, avx2

- FORTRAN / CPP run x3-4 processes in parallel
  - production time not differ much with current HPC configuration
- Improvement starts with DY+3j (FORTRAN vs CUDA), ~x3 for DY+4j

SUMMARY

- **Comparing timing estimations for FORTRAN/CPP/CUDA**
- Compared HPC based improvements for gridpack production, for DY+0/1/2/3/4j LO
- Major bottleneck is **compilation time** for CUDA
- Both compilation and execution slow in CPP?
- With current usage, expecting highest gain in processes with **small no. of procces / >= 6 final states**

EVENT GENERATION

EVENT GENERATION

From CMS gridpacks

Basic command for evt generation would be:

```shell
./runcmsgrid.sh $NEVT $RANDOMSEED $NB_CORE
```

In this scripts, it does:

1. Set up production environment (i.e. CMSSW)
2. Modify madevent/Cards/me5_configuration.txt (e.g. nb_core...)
3. Determine the no. of evts to be generated in each iteration.
4. Generate events. i.e.

```shell
./bin/gridrun 5000 $SEED $GRAN
```

5. Combine events / Check the no. of evts / Add scale and PDF weights to LHE files.

EVENT GENERATION

From CMS gridpacks

Basic command for evt generation would be

```shell
./runcmsgrid.sh $NEVT $RANDOMSEED $NB_CORE
```

In this scripts, it does:

1. Set up production environment (i.e. CMSSW)
2. Modify madevent/Cards/me5_configuration.txt (e.g. nb_core...)
3. Determine the no. of evts to be generated in each iteration.
4. Generate events. i.e.

```shell
./bin/gridrun 5000 $SEED $GRAN
```

5. Combine events / Check the no. of evts / Add scale and PDF weights to LHE files.

Parallelize Event Generation?
- Current version of MG does not support nb_core option
- Results in the slides are based on nb_core=1

EVENT GENERATION

From CMS gridpacks

Basic command for evt generation would be

```shell
./runcmsgrid.sh $NEVT $RANDOMSEED $NB_CORE
```

In this scripts, it does:

1. Set up production environment (i.e. CMSSW)
2. Modify madevent/Cards/me5_configuration.txt (e.g. nb_core...)
3. **Determine the no. of evts** to be generated in each iteration.
4. Generate events. i.e.

```shell
./bin/gridrun 5000 $SEED $GRAN
```

5. Combine events / Check the no. of evts / Add scale and PDF weights to LHE files.

- Basic blocks are generating 5000 events in each iteration.
- For each iteration, need to prepare running directories again, e.g. copy & pasting ./madevent executable, ajobs, etc.
- Done by *process/madevent/bin/internal/restore_data*

FIRST BOTTLENECK

- **restore_data: copy & pasting, untarring each subprocess directories**

- It takes a bit long time to prepare running directories before actual execution of madevent...
  e.g. DY+4j has 1455(412560) processes(diagrams)

```
for i in `cat subproc.mg` ; do
    cd $i
    echo $i
    rm -f ftn25 ftn26 >& /dev/null
    if [[ -e $1_results.dat ]]; then
    cp  $1_results.dat results.dat  >& /dev/null
    else
    cp  results.dat $1_results.dat  >& /dev/null
    fi
    for k in G* ; do
    if [[ ! -d $k ]]; then
        continue
    fi
    cd $k
    for j in $1_results.dat ; do
        if [[ -e $j ]] ; then
        cp  $j results.dat
        else
        cp  results.dat $j
        fi
    done
    for j in $1_ftn26.gz ; do
        if [[ -e $j ]]; then
        rm -f ftn26 >& /dev/null
        rm -f $1_ftn26 >& /dev/null
        gunzip $j
        cp $1_ftn26 ftn26
        gzip $1_ftn26
        fi
    done
    cd ../
    done
    cd ../
done
```

```
run.sh 5000 10
Now generating 5000 events with random seed 10 and granularity 1
************************************************************
*                                                          *
*                    W E L C O M E to                      *
*          M A D G R A P H 5 _ a M C @ N L O               *
*                  M A D E V E N T                         *
*                                                          *
*                *                   *                   *
*                  *       * *       *                   *
*                    * * * * 5 * * * *                     *
*                  *       * *       *                   *
*                *                   *                   *
*                                                          *
*        VERSION 3.5.3_lo_vect                             *
*                                                          *
*    The MadGraph5_aMC@NLO Development Team - Find us at   *
*    https://server06.fynu.ucl.ac.be/projects/madgraph     *
*                                                          *
*             Type 'help' for in-line help.                *
*                                                          *
************************************************************
INFO: load configuration from /srv/work/process/madevent/Cards/m
INFO: load configuration from /srv/work/mgbasedir/input/mg5_conf
INFO: load configuration from /srv/work/process/madevent/Cards/m
Using default text editor "vi". Set another one in ./input/mg5_c
No valid eps viewer found. Please set in ./input/mg5_configurati
No valid web browser found. Please set in ./input/mg5_configurat
WRITE GRIDCARD /srv/work/process/madevent
generate 5000 events
P0_gg_epemuux
P0_gg_epemddx
P0_gg_taptamuux
P0_gg_taptamddx
P0_gu_epemgu
P0_gd_epemgd
P0_gux_epemgux
P0_gdx_epemgdx
P0_gu_taptamgu
P0_gd_taptamgd
P0_gux_taptamgux
P0_gdx_taptamgdx
P0_uu_epemuu
P0_uux_epemuux
P0_dd_epemdd
P0_ddx_epemddx
P0_uxux_epemuxux
P0_dxdx_epemdxdx
P0_ud_epemud
P0_uc_epemuc
P0_uux_epemddx
P0_uux_epemccx
P0_udx_epemudx
```

- Has been parallelized by MG team [restore_data]
  ~ 30m reduced for DY+4j after parallelization with 16 CPUs

EVENT GENERATION

From CMS gridpacks

Basic command for evt generation would be

```shell
./runcmsgrid.sh $NEVT $RANDOMSEED $NB_CORE
```

In this scripts, it does:

1. Set up production environment (i.e. CMSSW)
2. Modify madevent/Cards/me5_configuration.txt (e.g. nb_core...)
3. Determine the no. of evts to be generated in each iteration.
4. Generate events. i.e.

```shell
./bin/gridrun 5000 $SEED $GRAN
```

5. **Combine events / Check the no. of evts / Add scale and PDF weights to LHE files.**
   This part does not take much time

SET-UP

Patches

After un-tarring gridpacks, two patches applied:

1. restore_data: To parallelize copy & past subprocesses directory - Utilize full CPU cores in machine
2. runcmsgrid.sh: Max Evts per iteration changed to 500k - avoid repeatedly call restore_data

```
75 max_events_per_iteration=500000
76
77 # define max event per iteration as 5000 if n_evt<45000 or n_evt/9 otherwise
78 #max_events_per_iteration=$(( $nevt > 5000*9 ? ($nevt / 9) + ($nevt % 9 > 0) : 5000 ))
79 # set starting variables
80 produced_lhe=0
81 run_counter=0
82 # if rnum allows, multiply by 10 to avoid multiple runs
83 # with the same seed across the workflow
84 run_random_start=$(($rnum*10))
85 # otherwise don't change the seed and increase number of events as 10000 if n_evt<50000 or n_evt/9 otherwise
86 #if [  $run_random_start -gt "89999990" ]; then
87 #    run_random_start=$rnum
88 #    max_events_per_iteration=$(( $nevt > 10000*9 ? ($nevt / 9) + ($nevt % 9 > 0) : 10000 ))
89 #fi
```

GENERATION TIME

Results

- Producing 100K events w/ single thread

	FORTRAN	CPP	CUDA
DY+2j	80m 10s	59s 2s	40m 2s
DY+3j	130m 51s	153m 46s	101m 25s
DY+4j	never ends (>4000m)	1366m 49s	426m 54s

- Improvement starts with DY+2j, ~x10 faster for DY+4j

INTEGRATION IN CMS RELVAL WORKFLOW

- **Integrated the workflow updated with GPU gridpack in the standard CMS RelVal workflow**
- ✓ Used the existing DY+4Jet workflow and updated the path to the gridpack
- ✓ Workflow directed toward GPU node [RelVal]
- ✓ Successfully produced events [JIRA]
- ✓ The full chain with pythia works and the gridpacks are now virtually production ready

SUMMARY

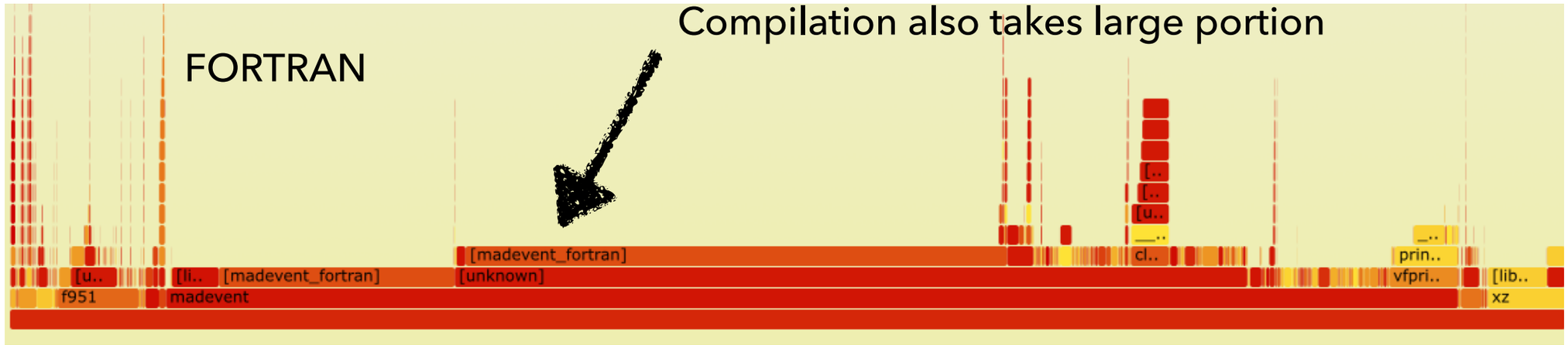
- **Event Generation**
- Compared thread-based event generation time
- Repeatedly copy & pasting subprocess directories matters
- Can see x10 improvement comparing DY+4j FORTRAN vs CUDA
- Implementation with RelVal chain has been tested

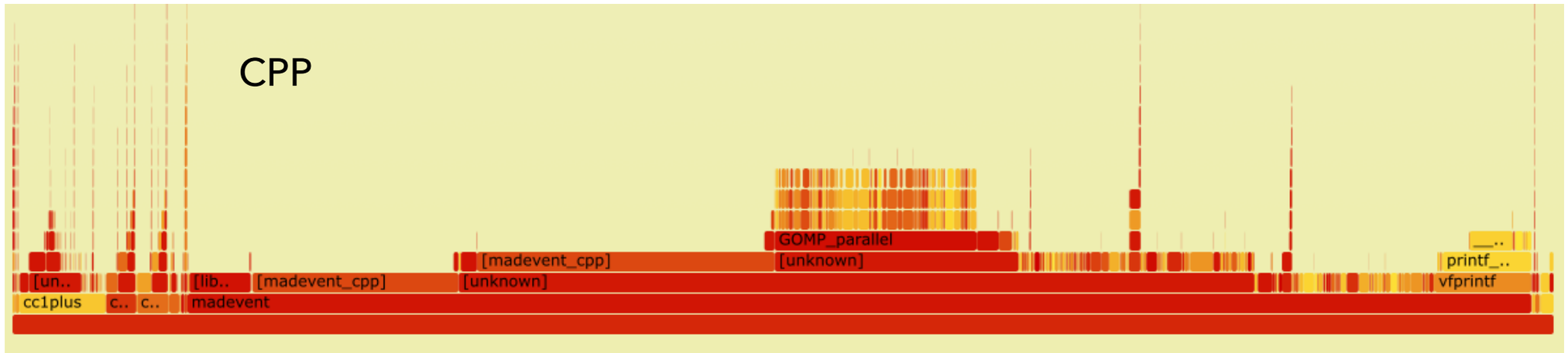
BACK UP

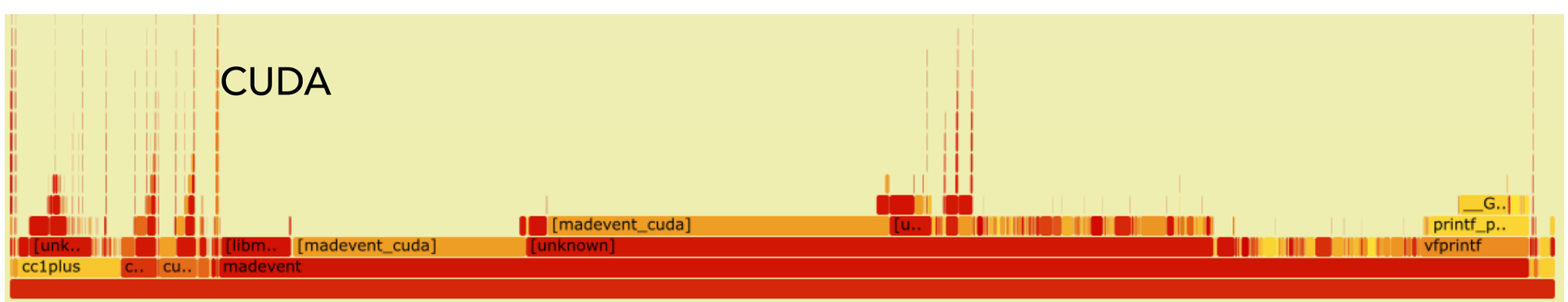
FLAMEGRAPHS - GRIDPACK PRODUCTION

DY+2j

Most of the time consuming part is still madevent...
Compilation also takes large portion





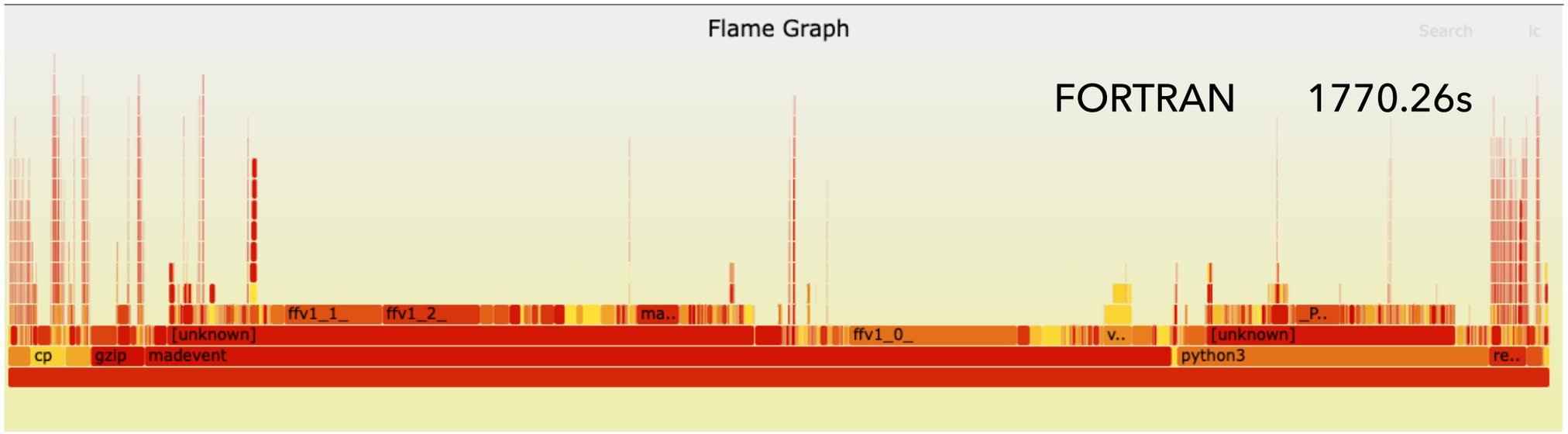


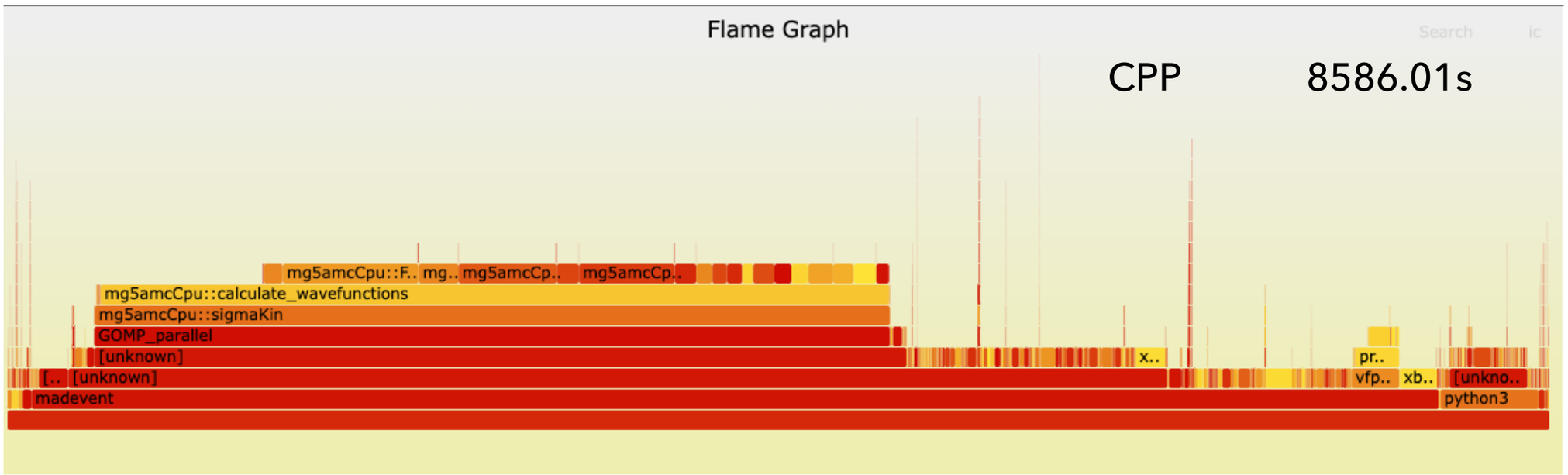
All jobs tested in lxplus800 node

FLAMEGRAPHS - EVENT GENERATION

DY+3j (generating 20000 events)

svg files in [lxplus]



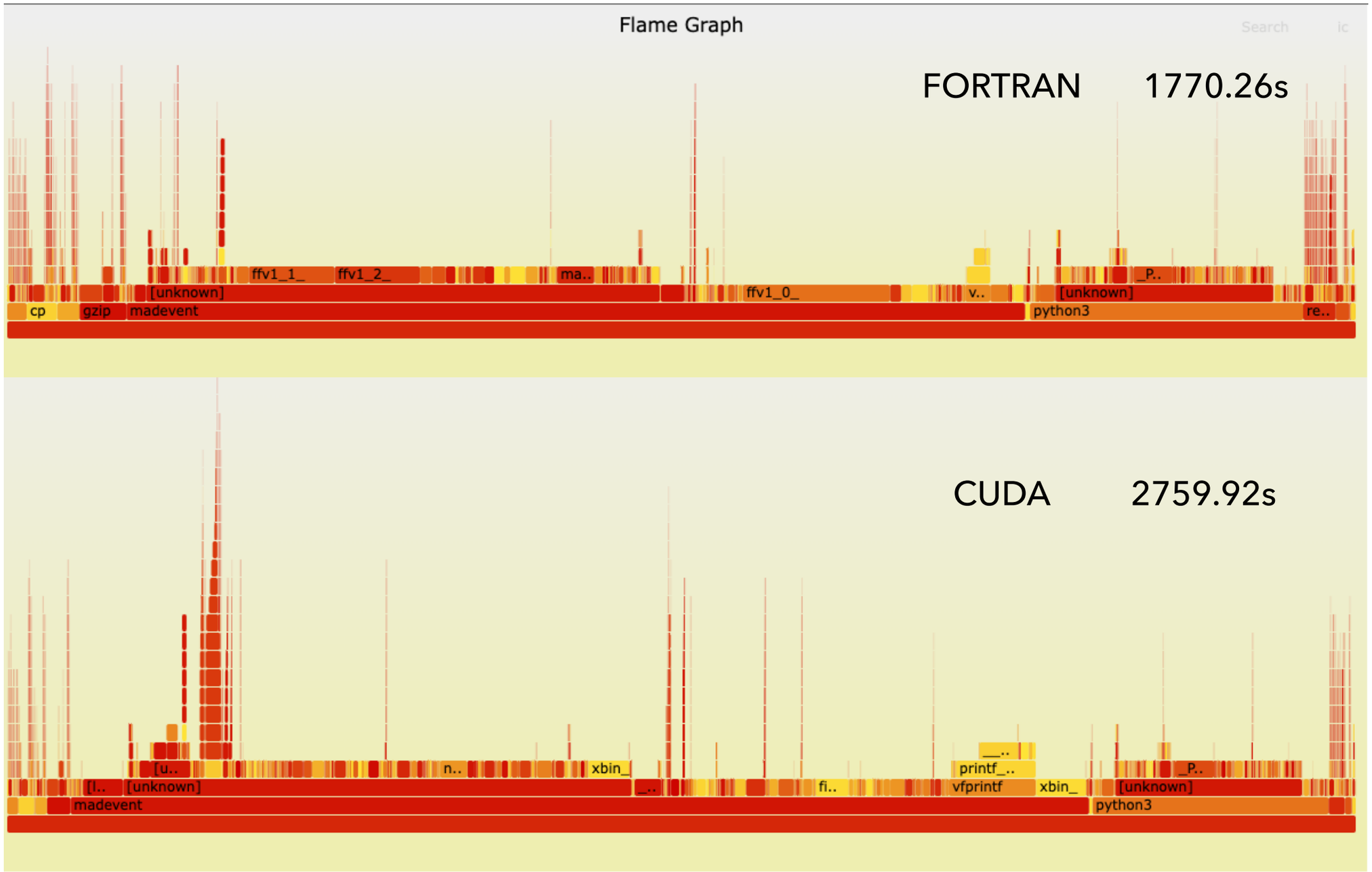


All jobs tested in lxplus8-gpu node

FLAMEGRAPHS - EVENT GENERATION

DY+3j (generating 20000 events)

svg files in [lxplus]



All jobs tested in lxplus8-gpu node