FERMILAB-PUB-19-192-T, LU-TP 19-14, MCNET-19-09

Simulation of vector boson plus many jet final states at the high luminosity LHC

Stefan Höche,[1] Stefan Prestel,[2] and Holger Schulz[3]
[1]*Fermi National Accelerator Laboratory, Batavia, IL, 60510-0500, USA*
[2]*Department of Astronomy and Theoretical Physics, Lund University, S-223 62 Lund, Sweden*
[3]*Department of Physics, University of Cincinnati, Cincinnati, OH 45219, USA*

We present a novel event generation framework for the efficient simulation of vector boson plus multi-jet backgrounds at the high-luminosity LHC and at possible future hadron colliders. MPI parallelization of parton-level and particle-level event generation and storage of parton-level event information using the HDF5 data format allow us to obtain leading-order merged Monte-Carlo predictions with up to nine jets in the final state. The parton-level event samples generated in this manner correspond to an integrated luminosity of 3ab^{-1} and are made publicly available for future phenomenological studies.

I. INTRODUCTION

The production of a di-lepton pair or a lepton-neutrino pair at hadron colliders [1, 2] is both one of the most studied and one of the best understood reactions in high-energy physics. Calculations using fixed-order QCD perturbation theory have been performed up to N^{2}LO accuracy [3–6]. The production of an additional jet can be described fully exclusively at N^{2}LO accuracy [7–11], and the production of up to five (four) additional jets in lepton-neutrino (di-lepton) events can be described [12–14] at NLO accuracy, partly in a fully automated fashion [15–20]. Tree-level matrix element generators [21–27] are capable of predicting the production rates of a Drell-Yan lepton pair in association with any number of jets, limited only by computing power. These predictions can be merged with parton showers [28–35] to create simulations that include the dominant effects of Sudakov resummation and are at the same time leading order accurate at any jet multiplicity. Such simulations prove to be an invaluable tool for LHC data analyses [36].

While techniques for the automated computation of tree-level matrix elements have been devised long ago [37–39], practical computations have not advanced considerably beyond the state of the art of the early 2000s [24, 25]. This has adverse effects on experimental measurements at the LHC, where up to eight-jet final states are routinely probed [40–47], and nine-jet final states will be measured to excellent precision with an integrated luminosity of $3ab^{-1}$. We address this problem by providing a novel event generation framework, combining core functionalities of Comix [25] and Pythia [48]. Inspired by the preliminary studies on HPC computing for perturbative QCD performed during the Snowmass 2013 community planning process [49, 50] and by the successful parallelization of Alpgen [51], we construct a workflow for parton-level and particle-level event generation that provides a scalable solution from desktops to high-performance computers. Existing limitations of Comix [52] are addressed and the new algorithms are tested in $W^{\pm}$ and Z boson production with up to nine jets at the LHC. The results of our parton-level event generation campaign are publicly available [53–55] and can be used as an input to particle-level simulations in experimental analyses and phenomenology. The code base can be downloaded from [56].

This paper is organized as follows. Section II introduces the technical challenges of particle-level simulations at scale and Section III presents our new event generation framework. Section IV discusses its computing performance and presents some first physics results. Section V contains an outlook.

II. EFFICIENT ALGORITHMS FOR HIGH MULTIPLICITY MULTI-JET MERGING

Multi-jet merged event simulations at leading or next-to-leading order QCD accuracy are the de facto standard for making precise, fully differential signal and background predictions for Standard Model measurements and new physics searches at the LHC [36]. They provide a consistent combination of the fully differential resummation provided by parton showers with exact higher-order perturbative QCD predictions of events with resolved jets. Using the example of $pp \to Z$+jets, this implies that the merging combines the parton-shower calculation of $pp \to Z$ with a parton-shower calculation for $pp \to Z + j$, a parton-shower calculation for $pp \to Z + jj$, and so on. As each of the results to be merged is inclusive over additional QCD radiation, and the parton shower can in principle populate the entire multi-jet phase space, two aspects need to be addressed in any merging procedure:

1. The phase space of resolvable emissions in the parton shower must be restricted to the complement of the phase space in the fixed-order calculation. This procedure is called the *jet veto*, the observable used to separate the phase space is called the *jet criterion*, and the numerical value where the separation occurs is called the *merging scale*.

2. The fixed-order result must be amended by the resummed higher-order corrections implemented by the parton shower, in order to maintain the logarithmic accuracy of the overall calculation. This involves

(a) Re-interpreting the final-state configuration of the fixed-order calculation as having originated from a parton cascade [28]. This procedure is called *clustering*, and the representations of the final-state configuration in terms of parton branchings are called *parton-shower histories*.

(b) Choosing appropriate scales for evaluating the strong coupling in each branching of this cascade, thereby resumming higher-order corrections to soft-gluon radiation [57, 58]. This procedure is called *α_s-reweighting*.

(c) Multiplying by appropriate Sudakov factors, representing the resummed unresolved real and virtual corrections [30]. This is called *Sudakov reweighting*, and is usually implemented by *trial showers* [31].

Step 2 in this algorithm turns the inclusive $pp \to Z + nj$ predictions into exclusive results, which describe the production of *exactly* n jets according to the jet criterion.[1] They can then be added to obtain the merged result. Care has to be taken that the result for the highest jet multiplicity remain inclusive over additional radiation which is softer than the softest existing jet. This is known as the *highest multiplicity treatment*.

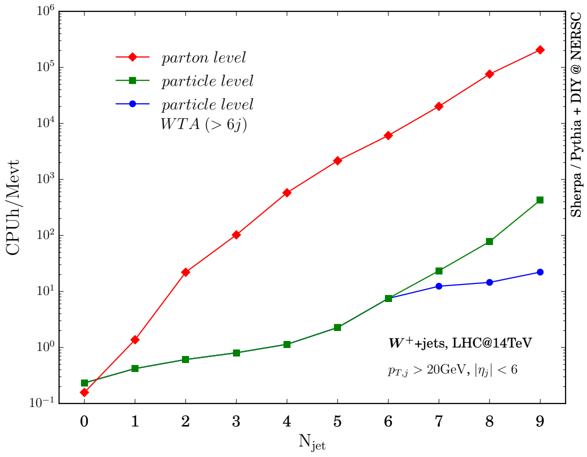

FIG. 1: Scaling of computation time (in CPU hours per 1 million events) for parton-level and particle-level event generation in multi-jet merged computations of W^{+}+jets at the LHC. We limit the number of quarks to ≤ 6 in $W^{+} + 6, 7$jet and to ≤ 4 in $W^{+} + 8, 9$jet final states. Green squares indicate results using the standard clustering procedure, while blue circles indicate results obtained from the winner-takes-all (WTA) approach as the number of jets exceeds six. See Sec. II C for details.

A. General Aspects of the Simulation

Technically, the merging algorithm described above involves multiple stages:

1. The computation of fixed-order results
2. The clustering and α_s reweighting
3. The parton shower and Sudakov reweighting

In the past, different implementations have combined these steps in different ways. Traditionally, the Sherpa event generator performs the jet clustering during the computation of the fixed-order result and optimizes the Monte-Carlo integrator based on the hard matrix element, including α_s reweighting. The Pythia event generator relies on external matrix element providers [60] to compute the perturbative inputs, and therefore a natural separation of Step 1 from the remainder of the calculation occurs. We argue that this also provides the more natural separation for improved compute performance. The reasons are twofold:

1. The parton shower and the clustering are probabilistic, in the sense that the number of particles produced in the shower, or the path chosen in the clustering are not known a priori. In contrast, the fixed-order perturbative calculations used as an input to the parton shower operate at fixed particle multiplicity, and always evaluate the same Feynman diagrams. By separating these two domains, we divide the program into two components with different program flow.

2. The computation of fixed-order results is very cumbersome at high multiplicity, even when making use of recursion relations. The corresponding unweighting efficiencies are usually very small. By comparison, both the parton shower and the jet clustering procedure are fast and consume significantly less memory. This is exemplified in Fig. 1. Separating the two domains and storing results of the fixed-order calculation into intermediate event files allows to reuse the computationally most expensive parts of the simulation for calculations with different parton-shower or hadronization parameters.

In the following we will discuss the problems related to fixed-order calculations and parton-shower simulations on HPC architectures in more detail, and present solutions that allow us to carry out simulations relevant for the high-luminosity LHC.

[1] Note that the jet criterion need not correspond to an experimentally relevant jet algorithm. It may be a purely theoretical construct as long as the infrared limits are properly identified [59].

B. Fixed-order Computations

Fixed-order computations usually proceed in two steps: The optimization stage uses adaptive algorithms like Vegas [61], and possibly multi channels [62], to better approximate the integrand, and therefore reduce the variance of the Monte-Carlo integral that is evaluated. During the integration or event generation stage, the integral is determined at high precision and weighted or unweighted events are generated, and possibly stored.

Both stages are ideally suited for MPI parallelization. During optimization, a large number of phase-space points must usually be generated to provide the input for adaptive algorithms. MPI parallelization can be achieved by independently producing a fraction of these points on each MPI rank and subsequently collecting the results. This process is repeated as needed. The computation of matrix elements can also be thread-parallelized using for example the techniques in [25, 63, 64]. During the event generation stage, any Monte-Carlo simulation is trivially parallelizable. The only complication are I/O operations related to the read-in of information related to the construction of the hard matrix elements and to the parameters of the adaptive integrator.

The related files can be large at high particle multiplicity, because the multi-channel integrator consists of many individual channels (typically one per diagram). We choose to store this information in a zip file by means of the libzippp interface [65]. We have implemented a parallelization layer into Comix, which both maps the file access through libzippp to a std::i/ostream, and enables read/write access to the actual zip file only on the master rank of the MPI executable. During the read operation, the file content obtained by the master rank is then broadcast to all ranks via the MPI Bcast function. We have implemented the same technology into the LHAPDF6 parton distribution library [66].

A significant complication occurs in the generation of unweighted events, which is most easily explained by using an example. Consider Fig. 2, which shows the distribution of the number of trials in the unweighting of $pp \to W^+ + nj$ events at leading order QCD with Comix. Let us assume that we want to generate 10^4 $W + 8j$ events. Generating them on a single rank, the timing per event will be determined by the average number of trials, which is given by the slope of the distribution. A trivial MPI parallelization strategy would be to generate $10^4/N$ events on each rank for N total ranks in the MPI run. This approach is bound to fail at large N, as can be seen easily: Assume that $N = 10^4$, and that one MPI rank generates an event with 250000 trials. The timing of the overall MPI run is then set by this rank, and until it has finished the remaining ranks are in an effective wait state. Therefore, the overall event generation time is 10^4 times the generation time for an event with 250000 trials. Generally, for an executable parallelized in this manner, the event generation time is set by the most inefficient rank. In order to efficiently parallelize the simulation, we might instead choose to generate weighted events and perform the unweighting independently. While the timing will be uniform in this case, weighted events require a large amount of disk storage, especially if the unweighting efficiency is as low as in the eight jet case. This holds true even if we store only the event weight.

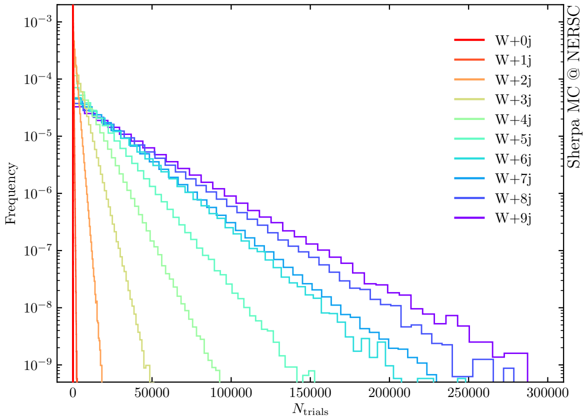

FIG. 2: Distribution of the number of trials in the unweighting of $pp \to W^+ + nj$ at the 14 TeV LHC. Jets are required to have $p_{T,j} > 20$ GeV and $|y_j| < 6$.

One may attempt to solve this problem by a hybrid approach: Generate unweighted events, but limit the number of trials to a fixed number $N_{\mathrm{trial,max}}$ that can be defined by the user of the program and should be adapted to the efficiency and timing for any given partonic final state. If the program reaches $N_{\mathrm{trial,max}}$, and no event has passed the unweighting, an event of weight zero is returned in order to achieve more uniform timing. However, for a large number of nodes this approach is still insufficient. We have verified this by disabling the write-out of events[2] and comparing to weighted event generation, which then scales ideally. Based on this finding, we implement a technique similar to Alpgen [24], which was adapted to scale to over a million threads in [51]: We generate a fixed number of weighted events, unweight on the fly, and store only those events which pass the unweighting procedure. This leaves us with a variable number of unweighted events, which cannot be predicted, as it depends on the efficiency of the integrator and is subject to statistical fluctuations. However, the scaling behavior is determined solely by the I/O performance (cf. Sec. IV).

To reduce the memory footprint of the executable, we allow to manually partition the partonic processes gener-

[2] We would not be able to compare the scaling if we were to write out weighted events, due to the high storage requirements in the weighted event case. Disabling the writeout for unweighted events instead leaves the scaling behavior of the unweighted event generation intact.

ated by Comix into sets of irreducible groups with identical diagrammatic structure of the hard matrix elements.

C. Parton Showers and Merging

The simulation of parton showers and the associated jet clustering procedure behave intrinsically different from the fixed-order computations discussed above, and they involve their own, unique computational challenges. Since the parton shower is a Markov Chain Monte Carlo, the particle multiplicity and the flavor structure varies event by event, such that memory usage and program flow cannot be predicted. Since the typical parton multiplicity is $\mathcal{O}(10-100)$, and only one possible parton-shower history is generated per event, this does not present a problem. The jet clustering procedure as the inverse operation to the parton shower does however create a challenge: In order to determine the correct probability for any given history, one must construct *all* histories corresponding to a given parton-level final state in the fixed-order calculation [28]. For indistinguishable final-state particles, the number of possible histories grows at least factorially, depending on the parton-shower recoil scheme. Both timing and memory usage of the executable therefore increase rapidly with the final-state multiplicity. This is exemplified in Fig. 1 for the exclusive contributions to a W^++jets calculation in a multi-jet merged approach. The computational complexity increases by approximately a factor 1.5 for each additional final-state jet.[3] As argued in Sec. II B any non-uniform timing in the calculation will eventually break the scaling, even if the simulation is trivially parallelizable, as in the case of parton showers. Starting with $pp \to W/Z+7$jet final states, the jet clustering procedure also begins to exhaust the memory of modern computers (at ≈4 GB/core).

These problems must be tackled by changes to the underlying merging algorithms. We use the CKKW-L technique [31], and amend it as follows: In order to make the generation of parton-level events independent of the merging procedure, we use a standard k_T jet algorithm to regularize the hard matrix elements and to veto the parton shower. In parton-level configurations that exceed the complexity of a $W/Z+6$jet final state, we perform the first clustering steps using the “winner takes it all” (WTA) approach: The clustering with largest probability is chosen, independent of the parton-shower history of the clustered lower-multiplicity state. This procedure is repeated until the final state has been reduced by the jet clustering to a $W/Z+6$jet configuration. At this point, the standard algorithm resumes. We have verified that the change to physical cross sections and distributions arising from this WTA approach is at the level of a few percent, and is significantly lower than the renormalization and factorization scale uncertainties arising from the perturbative expansion. Event samples with different final-state multiplicities at parton level are processed independently, in order to make the timing of the clustering and subsequent parton showering as uniform as possible. We use ASCR’s DIY framework [67, 68] to parallelize the particle-level event simulation.

III. NEW FRAMEWORK FOR MASSIVELY PARALLEL PROCESSING

It remains to discuss the combination of the fixed-order and parton shower calculations described above into a consistent, multi-jet merged event simulation. This relies on the efficient communication of event properties at the parton level by means of intermediate event files that are similar in spirit to Les Houches Event Files (LHEF) [60]. The LHEF standard is widely used in the high-energy physics community. It is based on XML, which poses a challenge for I/O operations, in particular the simultaneous read access when processing event information in heavily parallelized workflows. We address this problem by proposing a new format based on HDF5 [69], which is designed specifically for processing large amounts of data on HPC machines. HDF5 uses a computing model not too dissimilar from databases. The data stored is organized in data sets, that can be thought of as tables of standard types such as integer and float. These data sets can be organized in groups in order to create hierarchical structures. We strive to keep the new HDF5 event file standard as similar to the LHEF standard as possible.

The LHEF format comprises global properties as well as event-wise properties [60]. Global properties include process information (i.e. the type of collisions), total cross-sections as well as reweighting information. The event-wise properties are the process ID, the event weight, the scale of the hard process as well as the values of $\alpha_{\rm QCD}$, $\alpha_{\rm QED}$ and of course the list of particles generated. The latter contain the momentum four-vectors, information on particle identification (charges, spin, lifetime) as well as their genealogy. We propose to organize the HDF5 structure of Les Houches events such that each quantity normally stored in an event-wise block of XML is instead written to independent HDF datasets. We suggest to have separate groups for global, event-wise and particle properties.

1. *Global properties*

We follow the general idea of the XML-tags used in LHE files and define a group *init*. The names of the data sets as well as their data types are summarized in Tab. I. Properties of the individual processes are stored in the group *procInfo*. The names of the data sets and their

[3] Note that the scaling for $N>6$ is improved by the winner takes it all approach explained below.

Dataset	data type
PDFgroupA	int
PDFgroupB	int
PDFsetA	int
PDFsetB	int
beamA	int
beamB	int
energyA	double
energyB	double
numProcesses	int
weightingStrategy	int

TABLE I: Datasets and data types in the *init* group.

Dataset	data type
procId	int
xSection	double
error	double
unitWeight	double
npLO	int
npNLO	int

TABLE II: Datasets and data types in the *procInfo* group.

data types are summarized in Tab. II.

2. Event-wise properties

The information that is unique to a single event (except its particles) is stored in the *event* group. Table III gives an overview of the data set name and data types used.

The number *start* identifies the location of the first record in the additional group named *particle*, which stores information about individual particles that belong to each event. A summary of the information stored in the *particle* group is given in Tab. IV.

Dataset	data type
nparticles	int
start	int
pid	int
weight	double
scale	double
fscale	double
rscale	double
aqed	double
aqcd	double
npLO	int
npNLO	int
trials	double

TABLE III: Data sets and data types in the *event* group.

Dataset	data type
id	int
status	int
mother1	int
mother2	int
color1	int
color2	int
px	double
py	double
pz	double
e	double
m	double
lifetime	double
spin	double

TABLE IV: Data sets and data types in the *particle* group.

IV. NUMERICAL RESULTS

In this section we present first phenomenological results generated with the new event generation framework and discuss its computing performance. We consider proton-proton collisions at the high-luminosity LHC at $\sqrt{s} = 14$ TeV. We use the CT14 NNLO PDF set [70] and define the strong coupling accordingly. Our modified parton-level event generator is based on Comix [25] as included in Sherpa version 2.2.4 [71]. Our modified particle-level event generator, is based on Pythia 8 [48] and will be part of the next Pythia release.

Table V shows the parton-level cross sections for the event samples produced by Comix and used in the merging. Jets are defined using the k_T clustering algorithm with $R = 0.4$, $p_{T,j} > 20$ GeV and $|\eta_j| < 6$. Following the good agreement between parton-level and particle-level results established in [72], and the good agreement between fixed-order and MINLO [73] results established in [74], the renormalization and factorization scales are set to $\hat{H}'_T/2$ [18].

Figure 3 shows the scaling behavior of the parton-level calculation during the optimization stage. We limit the number of quarks to ≤ 6 in $W^+ + 6,7$jet and to ≤ 4 in $W^+ + 8,9$jet final states. The red line corresponds to a test of strong scaling. The number of non-zero points generated per rank and optimization step, which we denote as ${}^{p}\!/_{r}$, is inversely proportional to the number of ranks. The results deviate from the perfect scaling assumption shown by the red dotted line for fewer than $100\,{}^{p}\!/_{r}$. This is somewhat expected, as cut efficiencies tend to be non-uniform across various ranks when the number of points per rank is too low. The problem could in principle be addressed by not requesting a fixed number of non-zero phase-space points per optimization step, but by requesting a fixed number of points (zero and non-zero). However, such a procedure would not guarantee that the optimization can be carried out efficiently, because the adaptive Monte-Carlo integration depends on sufficient statistics [61, 62]. Fluctuations in the number of terms to be evaluated in the Monte-Carlo integration over color [75, 76] also contribute to the non-uniform timing, and therefore to the breakdown of scaling when the number of points per rank is too low. In the test of weak scaling, we keep the number of points per rank a constant and instead multiply the computation time by the square

$pp \to X + n$ jets	cross section [pb]									
X / n	0	1	2	3	4	5	6	7	8	9
W^+	9908(29)	2523(8)	1067(7)	404(4)	148(1)	49.3(5)	15.8(2)	5.2(2)	1.30(8)	0.330(6)
W^-	7496(21)	1898(6)	760(4)	278(2)	94(1)	29.8(3)	9.29(9)	2.71(7)	0.63(2)	0.170(3)
Z	1661(3)	464(1)	193.6(8)	72.2(3)	25.7(2)	8.61(8)	2.74(3)	0.82(2)	0.211(3)	0.057(1)

TABLE V: Inclusive cross sections at the LHC at $\sqrt{s} = 14$ TeV using the CT14nnlo PDF set and a correspondingly defined strong coupling. Jets are defined using the k_T clustering algorithm with $R = 0.4$, $p_{T,j} > 20$ GeV and $|\eta_j| < 6$.

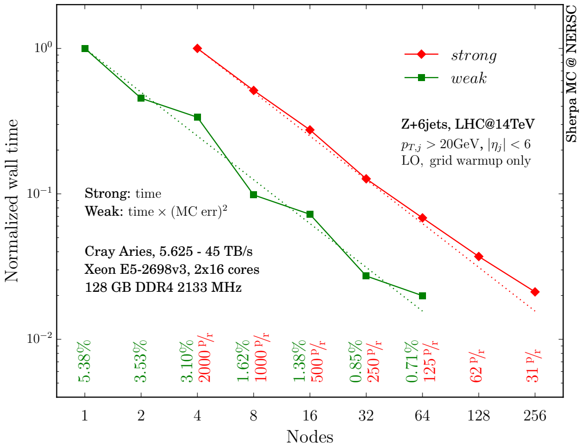

FIG. 3: Test of scaling behavior in the optimization stage of the parton-level calculation. In the strong scaling test (red) the number of points per rank ($^p/_r$) is inversely proportional to the number of ranks. In the weak scaling test (green), $^p/_r$ is constant and the computation time is multiplied by the square of the Monte-Carlo error.

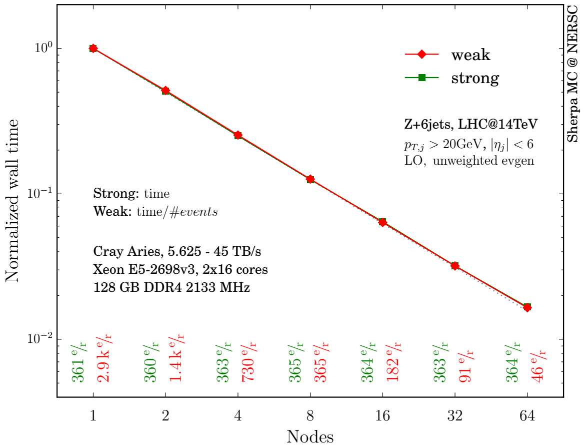

FIG. 4: Test of scaling behavior in the event generation stage of the parton-level calculation. In the strong scaling test (red) the number of events per rank, $^e/_r$, is inversely proportional to the number of ranks. In the weak scaling test (green) the computation time is divided by the number of events.

of the Monte-Carlo error. For a fixed number of non-zero points per rank, the error scales as $1/\sqrt{N}$, with N being the number of ranks. This behavior is what we observe on average, while the randomness in the Monte-Carlo integration generates fluctuations around the projected timing for ideal scaling. We remark, however, that the weak scaling test is rather academic. For a calculation like $W/Z + 9$jets, the timing of the overall Monte-Carlo integration is limited by the allocated computing time. The job shape as well as the number of points per rank need to be determined such that the wall time of the calculation can be minimized.

Figure 4 displays the scaling behavior of the parton-level event generation. Both the strong (red) and the weak (green) scaling test show satisfactory performance up to 2048 cores (64 nodes) on the Cori system at NERSC [77][4] While the scaling behavior is much better than when requesting a fixed number of unweighted events (cf. the discussion in Sec. III), it is not yet ideal, because the file size for event storage is determined dynamically. Optimizing the HDF5 output parameters may lead to further improvements.

Figure 5 shows the scaling behavior in the particle-level event simulation with Pythia 8 [48] and DIY [68]. Both the strong (red) and the weak (green) scaling test show a deviation from the ideal pattern, which can be attributed to the I/O overhead. In this context it is important to note that we have set the number of events for the weak and strong scaling test to be the same at 2048 MPI ranks (64 nodes). This implies a larger I/O overhead in the weak scaling test and explains the slightly worse scaling behavior at smaller number of ranks. It exemplifies that even for relatively complex simulations, like $Z + 6$jets, the I/O overhead quickly begins to dominate.

Finally, in Fig. 6 we show a test of our new parton-level event generation framework during the optimization stage on the KNL nodes of Cori Phase II at NERSC [77]. The memory footprint of the executable needs to be kept at a minimum for this test, in order not to exceed the 1.4 GB RAM/core available on this architecture. We have exhausted all four hyperthreads per core through MPI, which reduces the available memory per rank to 350 MB. The performance is reduced compared to a

[4] In order to speed up the writeout of events, we have made use of the Burst Buffer.

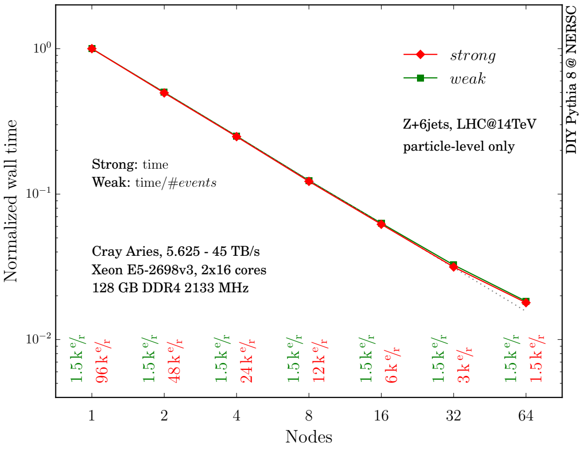

FIG. 5: Test of scaling behavior in the particle-level simulation. In the strong scaling test (red) the number of events per rank, $^e/_r$, is inversely proportional to the number of ranks. In the weak scaling test (green) the computation time is divided by the number of events.

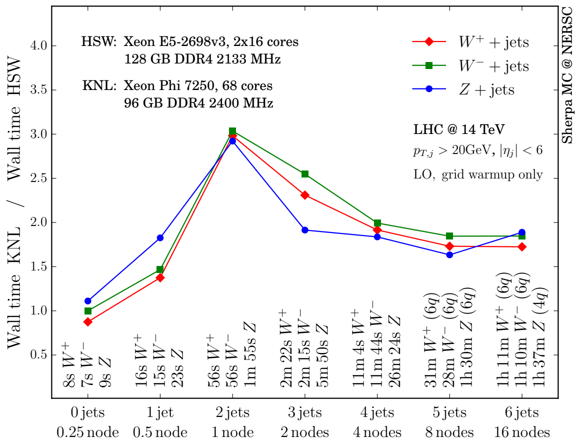

FIG. 6: Performance comparison of the parton-level event generation framework during the optimization stage between the KNL and Haswell architectures of Cori at NERSC [77].

Haswell architecture by a factor of two to three, depending on the jet multiplicity. Given the overall computing time needed for the optimization, this presents more an academic than a practical problem at this stage.

Figure 7 shows differential jet rates in Z+jets events using the k_T algorithm [78]. The colored lines represent the contributions from event samples of different jet multiplicity at parton level. It is interesting to note that high multiplicity configurations play a significant role in the $0 \to 1$ jet rate at high p_T.

Figure 8 shows the jet transverse momenta for a varying maximum jet multiplicity, $n_{\max}$, starting at the number of observed jets, N, and ranging up to $N + 3$ (if possible). As observed in [79, 80], the predictions for a low number of jets, N, stabilize at $n_{\max} = N+1$. It is interesting to observe that the correction to the $n_{\max} = N$ sample for $N \geq 3$ is negative at low p_T, indicating that the default parton-shower tune of Pythia 8.240 overestimates the emission rate in this configuration.

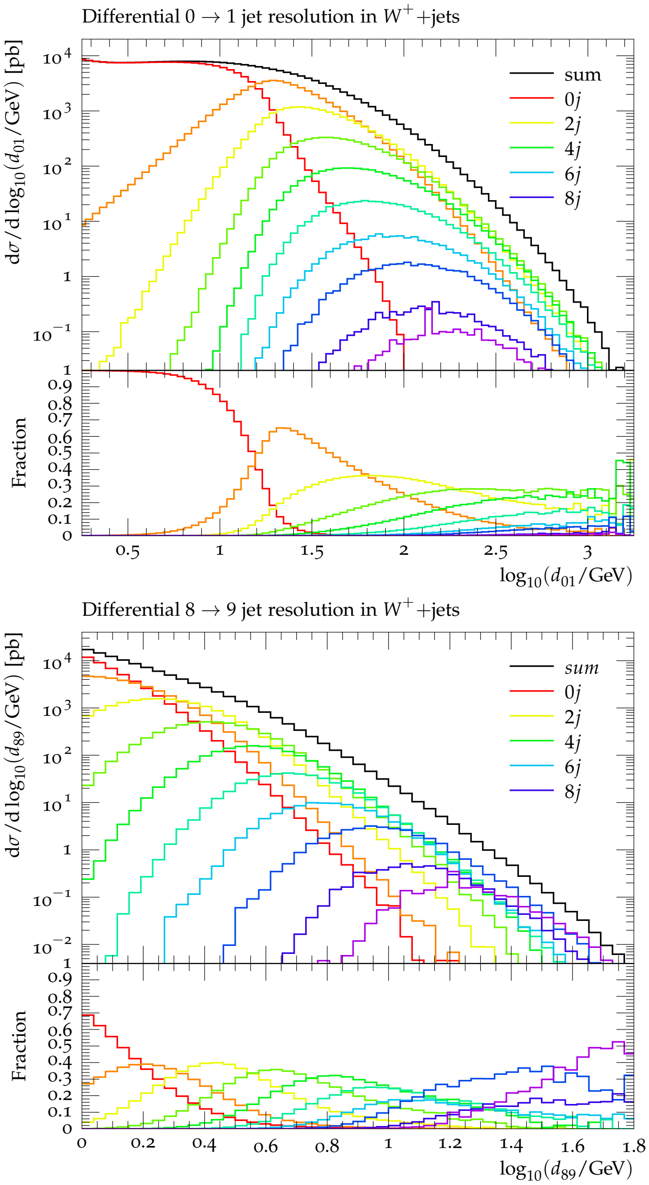

FIG. 7: Differential $0 \to 1$ and $8 \to 9$ k_T jet rates in Z+jets events. Colored lines represent the $Z + 0$-9 jet contributions in the merging.

V. CONCLUSIONS

We have presented a new event generation framework, suitable for massively parallel processing of multi-jet merged simulations for current and future collider experiments. Making use of high performance computing

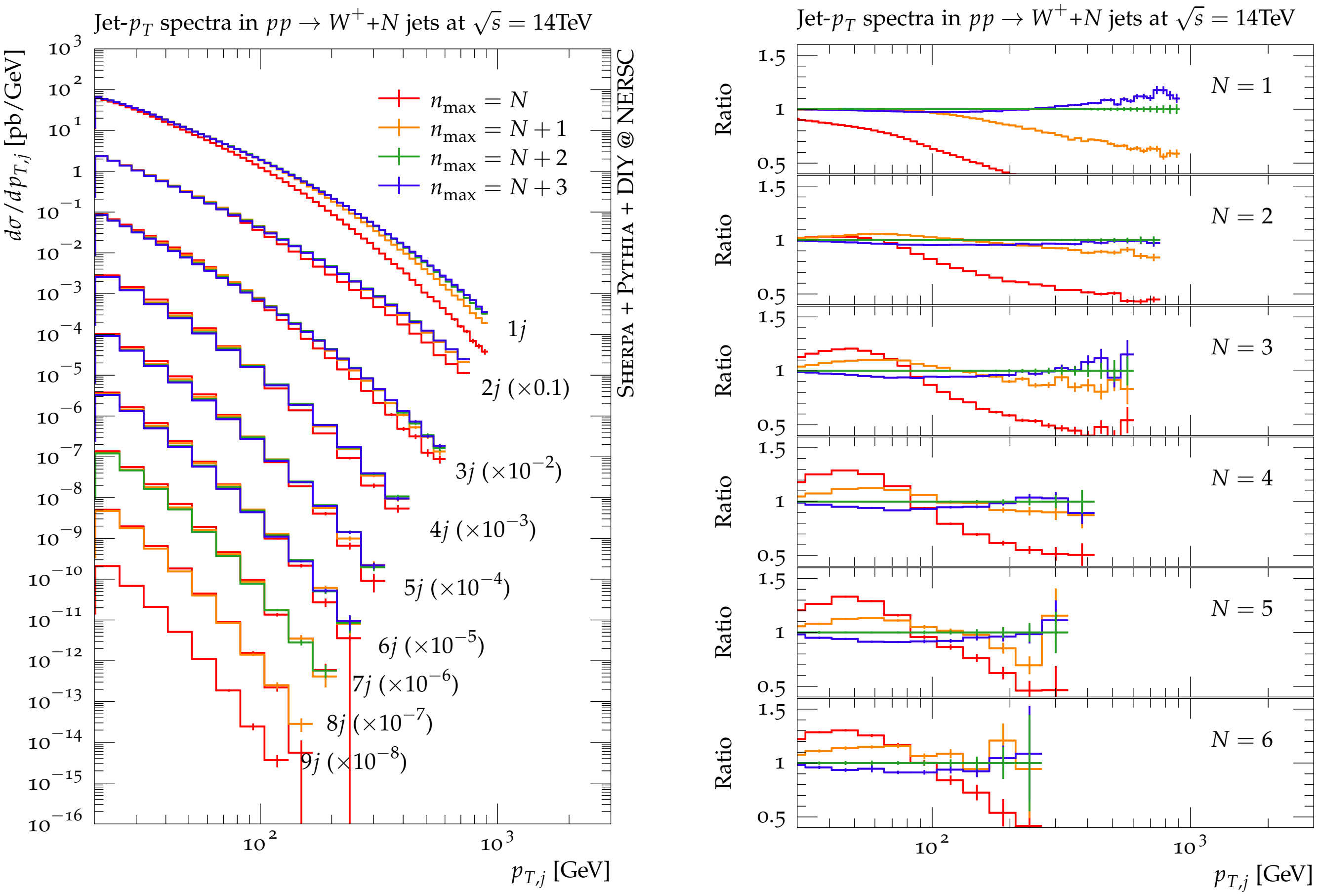

FIG. 8: Jet transverse momentum distributions in W^++jets events. We show a comparison of multi-jet merged simulations where the maximum jet multiplicity, $n_{\max}$, is set to the number of measured jets, N (red), to $N+1$ (green), $N+2$ (blue) and $N+3$ (purple).

resources, we have computed the first predictions for single vector boson production in proton proton collisions with up to nine jets described at leading order accuracy. This presents a significant improvement of the currently available simulations, which are capable of reaching leading order accuracy for only up to six jets in the final state. The results of our parton-level event generation campaign can be obtained from [53–55] and can be used as an input to particle-level simulations in experimental analyses as well as phenomenology. The modified parton-level and particle-level event generators, together with wrapper scripts and setups needed to perform similar simulations, are publicly available [56].

We have shown that the algorithms employed in our simulation scale as far as can reasonably be expected, given the problem size of the computation. Limitations arise during the integration stage from the limited number of phase-space points that are generated in each integration step, and in the event generation stage from the I/O limitations of the host system. We achieve best performance with MPI parallelization on up to 2048 ranks (64 nodes) of the Cori system at NERSC [77].

Acknowledgments

We thank our colleagues in the Sherpa and Pythia collaborations, in particular Frank Siegert and Stephen Mrenna, for discussions and support. We are grateful to Marc Paterno, Jim Kowalkowski, Chris Green and Saba Sehrisch for help with HDF5. We thank Rick and the team behind HDFql for their support. We thank Taylor Childers for numerous fruitful discussions on code performance and for comments on the manuscript. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program, grants “HEP Data Analytics on HPC”, No. 1013935 and “HPC framework for event generation at colliders”. It was supported by the U.S. Department of Energy under contracts DE-AC02-76SF00515 and DE-AC02-07CH11359 and used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

[1] S. D. Drell and T.-M. Yan, Phys. Rev. Lett. **25**, 316 (1970), [Erratum: Phys. Rev. Lett.25,902(1970)].

[2] S. D. Drell and T.-M. Yan, Annals Phys. **66**, 578 (1971).

[3] G. Altarelli, R. K. Ellis, and G. Martinelli, Nucl. Phys. **B143**, 521 (1978), [Erratum: Nucl. Phys.B146,544(1978)].

[4] G. Altarelli, R. K. Ellis, and G. Martinelli, Nucl. Phys. **B157**, 461 (1979).

[5] R. Hamberg, W. van Neerven, and T. Matsuura, Nucl.Phys. **B359**, 343 (1991).

[6] W. L. van Neerven and E. B. Zijlstra, Nucl. Phys. **B382**, 11 (1992), [Erratum: Nucl. Phys.B680,513(2004)].

[7] A. Gehrmann-De Ridder, T. Gehrmann, E. W. N. Glover, A. Huss, and T. A. Morgan, Phys. Rev. Lett. **117**, 022001 (2016), 1507.02850.

[8] R. Boughezal, J. M. Campbell, R. K. Ellis, C. Focke, W. T. Giele, X. Liu, and F. Petriello, Phys. Rev. Lett. **116**, 152001 (2016), 1512.01291.

[9] R. Boughezal, X. Liu, and F. Petriello, Phys. Rev. **D94**, 113009 (2016), 1602.06965.

[10] A. Gehrmann-De Ridder, T. Gehrmann, E. W. N. Glover, A. Huss, and T. A. Morgan, JHEP **07**, 133 (2016), 1605.04295.

[11] A. Gehrmann-De Ridder, T. Gehrmann, E. W. N. Glover, A. Huss, and D. M. Walker, Phys. Rev. Lett. **120**, 122001 (2018), 1712.07543.

[12] W. Giele, E. N. Glover, and D. A. Kosower, Nucl.Phys. **B403**, 633 (1993), hep-ph/9302225.

[13] J. M. Campbell and R. Ellis, Phys. Rev. **D65**, 113007 (2002), hep-ph/0202176.

[14] J. M. Campbell, R. Ellis, and D. L. Rainwater, Phys. Rev. **D68**, 094021 (2003), hep-ph/0308195.

[15] R. Ellis, K. Melnikov, and G. Zanderighi, JHEP **04**, 077 (2009), 0901.4101.

[16] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres-Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maître, Phys. Rev. Lett. **102**, 222001 (2009), 0902.2760.

[17] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres-Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maître, Phys. Rev. **D82**, 074002 (2010), 1004.1659.

[18] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres-Cordero, D. Forde, T. Gleisberg, H. Ita, D. A. Kosower, and D. Maître, Phys. Rev. Lett. **106**, 092001 (2011), 1009.2338.

[19] H. Ita, Z. Bern, L. J. Dixon, F. Febres-Cordero, D. A. Kosower, and D. Maître, Phys.Rev. **D85**, 031501 (2012), 1108.2229.

[20] Z. Bern, L. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, D. Maître, and K. J. Ozeren, Phys.Rev. **D88**, 014025 (2013), 1304.1253.

[21] A. Kanaki and C. G. Papadopoulos, Comput. Phys. Commun. **132**, 306 (2000), hep-ph/0002082.

[22] C. G. Papadopoulos, Comput. Phys. Commun. **137**, 247 (2001), hep-ph/0007335.

[23] F. Krauss, R. Kuhn, and G. Soff, JHEP **02**, 044 (2002), hep-ph/0109036.

[24] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau, and A. D. Polosa, JHEP **07**, 001 (2003), hep-ph/0206293.

[25] T. Gleisberg and S. Höche, JHEP **12**, 039 (2008), 0808.3674.

[26] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer, JHEP **06**, 128 (2011), 1106.0522.

[27] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, JHEP **07**, 079 (2014), 1405.0301.

[28] J. André and T. Sjöstrand, Phys. Rev. **D57**, 5767 (1998), hep-ph/9708390.

[29] M. L. Mangano, M. Moretti, and R. Pittau, Nucl. Phys. **B632**, 343 (2002), hep-ph/0108069.

[30] S. Catani, F. Krauss, R. Kuhn, and B. R. Webber, JHEP **11**, 063 (2001), hep-ph/0109231.

[31] L. Lönnblad, JHEP **05**, 046 (2002), hep-ph/0112284.

[32] K. Hamilton, P. Richardson, and J. Tully, JHEP **11**, 038 (2009), 0905.3072.

[33] S. Höche, F. Krauss, M. Schönherr, and F. Siegert, JHEP **08**, 123 (2011), 1009.1127.

[34] L. Lönnblad and S. Prestel, JHEP **02**, 094 (2013), 1211.4827.

[35] S. Plätzer, JHEP **08**, 114 (2013), 1211.5467.

[36] A. Buckley et al., Phys. Rept. **504**, 145 (2011), 1101.2599.

[37] F. A. Berends and W. Giele, Nucl. Phys. **B294**, 700 (1987).

[38] F. A. Berends and W. T. Giele, Nucl. Phys. **B306**, 759 (1988).

[39] A. Ballestrero and E. Maina, Phys. Lett. **B350**, 225 (1995), hep-ph/9403244.

[40] G. Aad et al. (ATLAS), JHEP **07**, 032 (2013), 1304.7098.

[41] G. Aad et al. (ATLAS), Eur. Phys. J. **C75**, 82 (2015), 1409.8639.

[42] M. Aaboud et al. (ATLAS), JHEP **05**, 077 (2018), 1711.03296.

[43] M. Aaboud et al. (ATLAS), Eur. Phys. J. **C77**, 361 (2017), 1702.05725.

[44] V. Khachatryan et al. (CMS), Phys. Rev. **D95**, 052002 (2017), 1610.04222.

[45] A. M. Sirunyan et al. (CMS), Phys. Rev. **D96**, 072005 (2017), 1707.05979.

[46] A. M. Sirunyan et al. (CMS), Eur. Phys. J. **C78**, 965 (2018), 1804.05252.

[47] V. Khachatryan et al. (CMS), JHEP **04**, 022 (2017), 1611.03844.

[48] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands (2014), 1410.3012.

[49] S. Höche, L. Reina, M. Wobisch, et al. (2013), 1309.3598.

[50] L. Bauerdick, S. Gottlieb, et al. (2014), 1401.6117.

[51] J. T. Childers, T. D. Uram, T. J. LeCompte, M. E. Papka, and D. P. Benjamin, Comput. Phys. Commun. **210**, 54 (2017), 1511.07312.

[52] D. Benjamin, J. T. Childers, S. Hoeche, T. LeCompte, and T. Uram, J. Phys. Conf. Ser. **898**, 072044 (2017).

[53] H. Schulz, S. Höche, and S. Prestel, *Z + up to 9 jets parton level events at 14 TeV in HDF5* (2019), URL https://doi.org/10.5281/zenodo.2678039.

[54] H. Schulz, S. Höche, and S. Prestel, *W^+ + up to 9 jets parton level events at 14 TeV in HDF5* (2019), URL https://doi.org/10.5281/zenodo.2678055.

[55] H. Schulz, S. Höche, and S. Prestel, *W^- + up to 9 jets parton level events at 14 TeV in HDF5* (2019), URL https://doi.org/10.5281/zenodo.2678091.

[56] S. Höche, S. Prestel, and H. Schulz (2019), URL https://gitlab.com/hpcgen.

[57] D. Amati, A. Bassetto, M. Ciafaloni, G. Marchesini, and G. Veneziano, Nucl. Phys. **B173**, 429 (1980).

[58] S. Catani, B. R. Webber, and G. Marchesini, Nucl. Phys. **B349**, 635 (1991).

[59] G. P. Salam, Eur. Phys. J. **C67**, 637 (2010), 0906.1833.

[60] J. Alwall et al., Comput. Phys. Commun. **176**, 300 (2007), hep-ph/0609017.

[61] G. P. Lepage, J. Comput. Phys. **27**, 192 (1978).

[62] R. Kleiss and R. Pittau, Comput. Phys. Commun. **83**, 141 (1994), hep-ph/9405257.

[63] W. Giele, G. Stavenga, and J.-C. Winter, Eur. Phys. J. **C71**, 1703 (2011), 1002.3446.

[64] J. M. Campbell, R. K. Ellis, and W. T. Giele, Eur. Phys. J. **C75**, 246 (2015), 1503.06182.

[65] The Libzippp Developers (2019), URL https://github.com/ctabin/libzippp.

[66] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, and G. Watt, Eur. Phys. J. **C75**, 132 (2015), 1412.7420.

[67] T. Peterka, R. Ross, W. Kendall, A. Gyulassy, V. Pascucci, H.-W. Shen, T.-Y. Lee, and A. Chaudhuri, in *Proceedings of Large Data Analysis and Visualization Symposium LDAV'11* (Providence, RI, 2011).

[68] D. Morozov and T. Peterka, in *Proceedings of the 2016 IEEE Large Data Analysis and Visualization Symposium LDAV'16* (Baltimore, MD, 2016).

[69] The HDF Group (2017), URL https://support.hdfgroup.org/HDF5.

[70] S. Dulat et al., Phys. Rev. **D93**, 033006 (2016), 1506.07443.

[71] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, and J. Winter, JHEP **02**, 007 (2009), 0811.4622.

[72] J. Bellm et al. (2019), 1903.12563.

[73] K. Hamilton, P. Nason, and G. Zanderighi, JHEP **10**, 155 (2012), 1206.3572.

[74] F. R. Anger, F. Febres Cordero, S. Höche, and D. Maître, Phys. Rev. **D97**, 096010 (2018), 1712.08621.

[75] F. Maltoni, K. Paul, T. Stelzer, and S. Willenbrock, Phys. Rev. **D67**, 014026 (2003), hep-ph/0209271.

[76] C. Duhr, S. Höche, and F. Maltoni, JHEP **08**, 062 (2006), hep-ph/0607057.

[77] NERSC (2016), URL https://www.nersc.gov/users/computational-systems/cori.

[78] S. Catani, Y. L. Dokshitzer, M. H. Seymour, and B. R. Webber, Nucl. Phys. **B406**, 187 (1993).

[79] F. Krauss, A. Schälicke, S. Schumann, and G. Soff, Phys. Rev. **D70**, 114009 (2004), hep-ph/0409106.

[80] F. Krauss, A. Schälicke, S. Schumann, and G. Soff, Phys. Rev. **D72**, 054017 (2005), hep-ph/0503280.