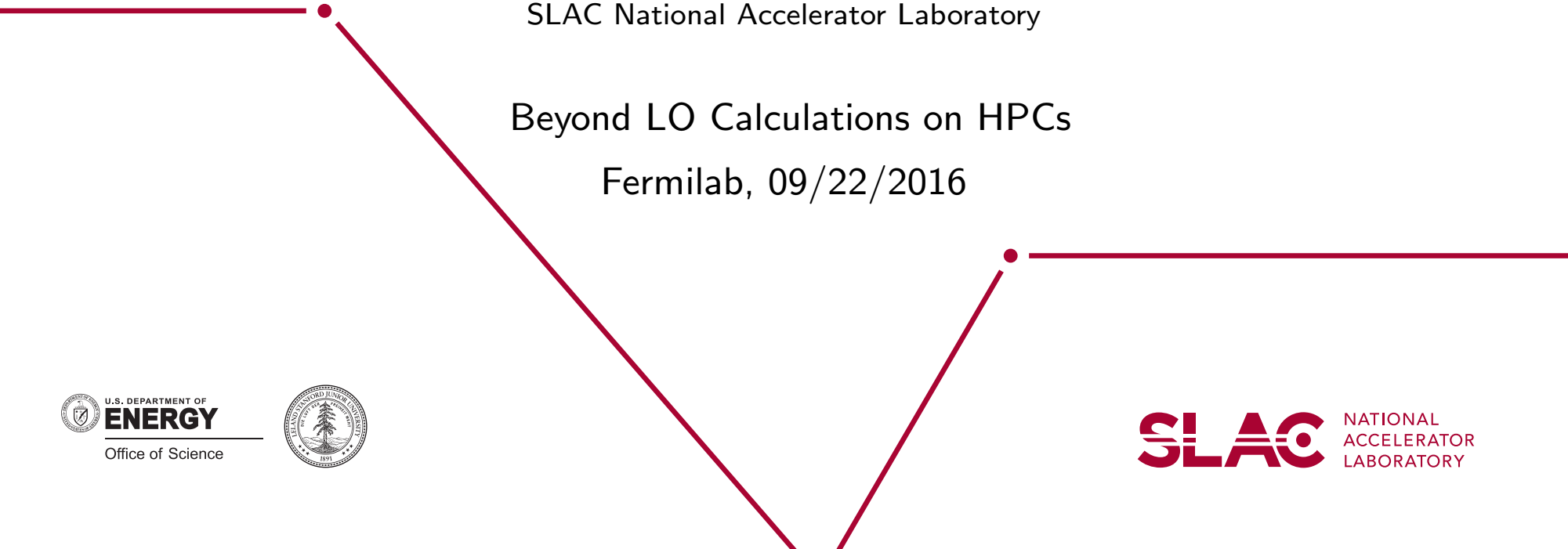
Sherpa NLO Generator
Software Operation and Organization

Stefan Höche

SLAC National Accelerator Laboratory

Beyond LO Calculations on HPCs

Fermilab, 09/22/2016



U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

NLO calculations and the art of collaboration

SLAC

- Tree amplitudes recycled into NLO real corrections and subtraction terms

$$\sigma_{\mathrm{NLO}} = \int \mathrm{d}\Phi_{\mathrm{B}} \sum \left(\mathrm{B} + \tilde{\mathrm{V}} + \mathrm{I} \right) + \int \mathrm{d}\Phi_{R} \sum \left(\mathrm{R} - \mathrm{S} \right)$$

- Virtual corrections computed separately (↗ next slide)
- Various automated NLO frameworks based on this idea
  - HELAC [Bevilacqua,Czakon,Garzelli,vanHameren,Kardos,Papadopoulos,Pittau,Worek]
  - Herwig7 [Bellm,Fischer,Gieseke,Grellscheid,Harrendorf,Nail,Papaefstathiou, Plätzer,Rauch[2],Reuschle,Richardson,Schichtel,Seymour,Siódmok,Wilcock]
  - MadGraph5 [Alwall,Frederix,Frixione,Hirschi,Maltoni,Mattelaer,Shao,Stelzer,Torrielli,Zaro]
  - Sherpa [Bothmann,Krauss,Kuttimalai,Li,Schönherr,Schulz,Schumann,Siegert,SH]
  - Whizard [Chokoufe,Hoang,Kilian,Ohl,Reuter,Stahlhofen,Teubner,Weiss]

NLO calculations and the art of collaboration

SLAC

Large set of 1-loop tools available

- **Based on tensor reduction** [Binoth,Guillet,Pilon,Heinrich,Schubert] hep-ph/0504267 [Denner,Dittmaier] hep-ph/0509141, [Denner,Dittmaier,Hofer] arXiv:1604.06792
  - Golem95 [Binoth,Cullen,Greiner,Guffanti,Guillet,Heinrich,Karg,Kauer,Reiter,Reuter]
  - MadGolem [Binoth,Goncalves Netto,Lopez-Val,Mawatari,Plehn,Wigmore]
  - MadLoop [Hirschi,Frederix,Frixione,Garzelli,Maltoni,Pittau]
  - OpenLoops [Cascioli,Maierhöfer,Pozzorini]
- **Based on generalized unitarity** [Bern,Dixon,Dunbar,Kosower] hep-ph/9409265 [Ossola,Papadopoulos,Pittau] hep-ph/0609007, arXiv:0802.1876
  - BlackHat [Bern,Dixon,Febres-Cordero,Ita,Kosower,Maître,Ozeren,SH]
  - GoSam [Cullen,Greiner,Heinrich,Luisoni,Mastrolia,Ossola,Reiter,Tramontano]
  - HelacNLO [Bevilacqua,Czakon,Garzelli,vanHameren,Kardos,Papadopoulos,Pittau,Worek]
  - MadLoop [Hirschi,Frederix,Frixione,Garzelli,Maltoni,Pittau]
  - NJet [Badger,Biedermann,Uwer,Yundin]
  - OpenLoops [Cascioli,Maierhöfer,Pozzorini]
  - Rocket [Ellis,Giele,Kunszt,Melnikov,Zanderighi]

The quest for many jets

SLAC

$pp \to$ jets (7 TeV) real-radiation & subtraction only

σ_{R-S} [nb]	Number of jets				
n k_T-jets $p_{Tj} > 30(50)$ GeV, R=0.4	2 $\alpha_c = 0.03$	3 $\alpha_c = 0.01$	4 $\alpha_c = 0.003$	5 $\alpha_c = 0.001$	6 $\alpha_c = 0.001$
AMEGIC++ Comix	-1141(28) -1188(22)	455(5) **447(6)**	121(13) **117(10)**	– **21.0(2)**	– **2.39(5)**
Timing Ratio	0.70	**2.1**	**2.8**	–	–

$pp \to e^+e^-$+jets (7 TeV) real-radiation & subtraction only

σ_{R-S} [pb]	Number of jets					
n k_T-jets $p_{Tj} > 30$ GeV, R=0.4	0	1 $\alpha_c = 0.1$	2 $\alpha_c = 0.03$	3 $\alpha_c = 0.01$	4 $\alpha_c = 0.003$	5 $\alpha_c = 0.001$
AMEGIC++ Comix	30.6(1) 30.7(1)	25.2(3) 25.4(3)	16.7(2) **16.3(2)**	11.5(1) **11.6(1)**	– **5.68(4)**	– **2.00(2)**
Timing Ratio	0.4	0.5	**1.9**	**1.8**	–	–

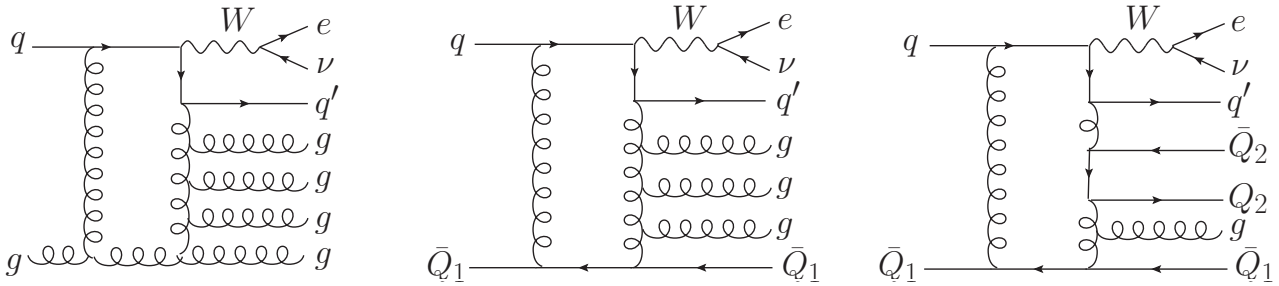
$pp \to e^+\nu_e$+jets (7 TeV) real-radiation & subtraction only

σ_{R-S} [pb]	Number of jets					
n k_T-jets $p_{Tj} > 30$ GeV, R=0.4	0	1 $\alpha_c = 0.1$	2 $\alpha_c = 0.03$	3 $\alpha_c = 0.01$	4 $\alpha_c = 0.003$	5 $\alpha_c = 0.001$
AMEGIC++ Comix	202(1) 200(1)	168(2) 167(2)	100(1) 96(1)	70.5(5) **69.5(4)**	– **36.0(3)**	– **13.3(2)**
Timing Ratio	0.4	0.9	0.6	**3.4**	–	–

W+5 jets at NLO

[Bern,Dixon,Febres Cordero,Ita,Kosower,Maître,Ozeren,SH] arXiv:1304.1253

- First computation of $2 \to 6$ process at NLO QCD accuracy



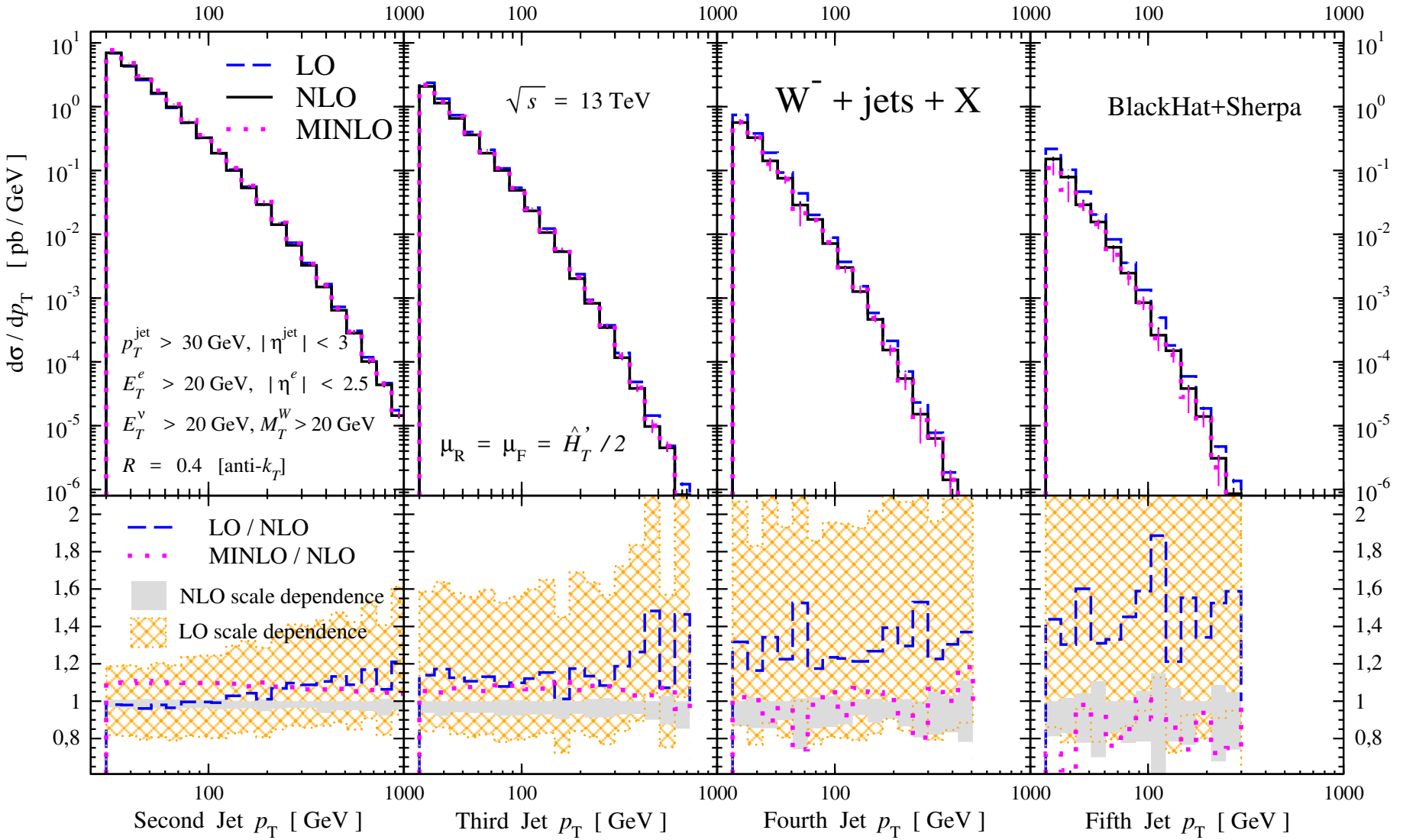
- Sherpa NLO MC framework using Comix [Gleisberg,SH] arXiv:0808.3674 combined with BlackHat [Berger et al.] arXiv:0803.4180, arXiv:0907.1984, arXiv:1009.2338
- BlackHat based on generalized unitarity $\to$ 1-loop amplitudes from on-shell tree amplitudes [Bern,Dixon,Dunbar,Kosower] hep-ph/9403226, hep-ph/9708239
  Cut-constructible part reduced to scalar integrals at integrand level
  [Ossola,Papadopoulos,Pittau] hep-ph/0609007, arXiv:0802.1876, [Forde] arXiv:0704.1835

$$A_{loop} = \sum d_i \; \square + \sum c_i \; \triangle + \sum b_i \; \text{—} + R + \mathcal{O}(\epsilon)$$

- Scale dependence studied using $H_{T,m} = \sum m_\perp$ and MINLO [Hamilton,Nason,Zanderighi] arXiv:1206.3572

W+5 jets at NLO

[new results courtesy of F. Febres-Cordero]



W+5 jets Performance on Cori

SLAC

Type	RAM/core [GB]	Init Time	MC error	# Cores & Time	# Cores per Node
l qq	5.0/4.1GB	35m	1.2%	512 × 1d	28
l gq	1.4/1.2GB	11m	0.6%	512 × 1.8d	32
l gg	0.3/0.2GB	2m	0.9%	512 × 3.5h	32
RS qq→6g	1.4/1.1GB	30m	1.9%	1024 × 8h	32
RS gq→5g1q	1.3/1.1GB	25m	2.5%	1024 × 16h	32
RS gg→4g2q	0.7/0.6GB	12m	2.9%	1024 × 4.5h	32
RS qq→4g2q	16/12.3GB	6h	10.5%	16 × 13d	9 *
RS gq→3g3q	6.2/4.8GB	1.7h	4.3%	1024 × 1.8d	23
RS gg→2g4q	1.3/1.0GB	24m	2.8%	512 × 16h	32
RS qq→2g4q	32/24.4GB	12h	22%	16 × 9d	4 *
RS gq→1g5q	6.9/5.4GB	2h	34%	512 × 42h	16
RS gg→6q	1.5/1.2GB	30m	6.7%	512 × 23h	32
RS qq→6q	23/16.5GB	4h	35%	16 × 12d	7 *

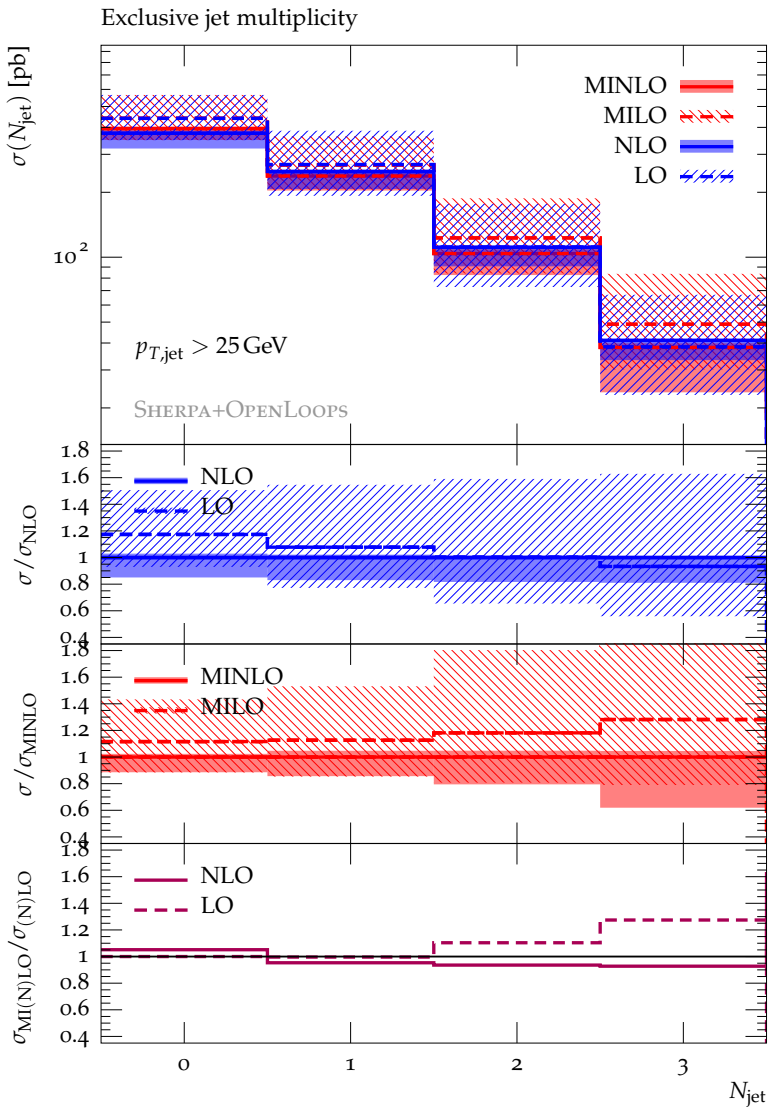
Disclaimer: This does not include the costs for NTuple production

* Computed on UCLA Hoffman 2 Cluster

$t\bar{t}$+3 jets at NLO

[Maierhöfer,Moretti,Pozzorini,Siegert,SH] arXiv:1607.06934

- First computation of $t\bar{t}$+3 jets at NLO / MINLO accuracy
- Sherpa NLO MC framework using Comix [Gleisberg,SH] arXiv:0808.3674 combined with OpenLoops [Cascioli,Maierhöfer,Pozzorini] arXiv:1111.5206
- Public results in NTuple format à la [BlackHat collaboration] arXiv:1310.7439 for easy analysis & recycling available at NERSC
- Scale dependence studied using $H_{T,m} = \sum m_\perp$ and MINLO [Hamilton,Nason,Zanderighi] arXiv:1206.3572 extended to massive partons



$t\bar{t}$+3 jets Performance on Bullet Cluster (SLAC)

SLAC

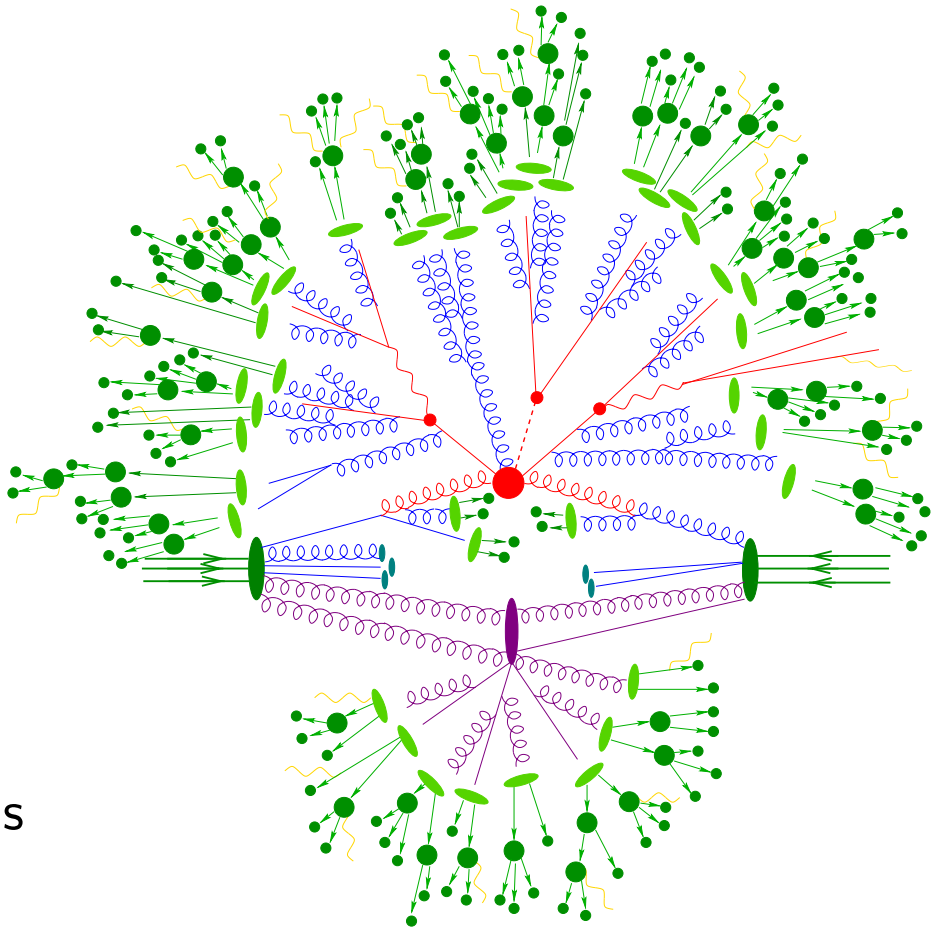
Type	RAM/core [GB]	Init Time	MC error	# Cores & Time
l gg→3g	85/127MB	20s	0.27%	64 × 4.5h
l qq→3g	68/110MB	13s	0.34%	64 × 1.1h
l gq→2g1q	72/114MB	17s	0.46%	64 × 58m
l gg→1g2q	48/90MB	10s	0.34%	64 × 31m
l qq→1g2q	150/187MB	43s	0.42%	64 × 3.2h
l gq→3q	87/126MB	25s	0.29%	64 × 1.3h
RS 2g→4g	501/574MB	4m 33s	1.4%	128 × 20.5h
RS 2q→4g	548/614MB	5m 21s	2.4%	128 × 7.7h
RS 1g1q→3g1q	553/621MB	4m 39s	0.88%	128 × 10.7h
RS 2g→2g2q	248/315MB	1m 29s	1.1%	128 × 4.3h
RS 2q→2g2q	1.9/1.9GB	12m 25s	1.1%	128 × 18.8h
RS 1g1q→1g3q	670/737MB	4m 41s	0.65%	128 × 1.08d
RS 2g→4q	208/274MB	1m	0.45%	128 × 4.5h
RS 2q→4q	1.8/1.8GB	9m 59s	0.61%	128 × 1.5d

Disclaimer: This does not include the costs for NTuple production

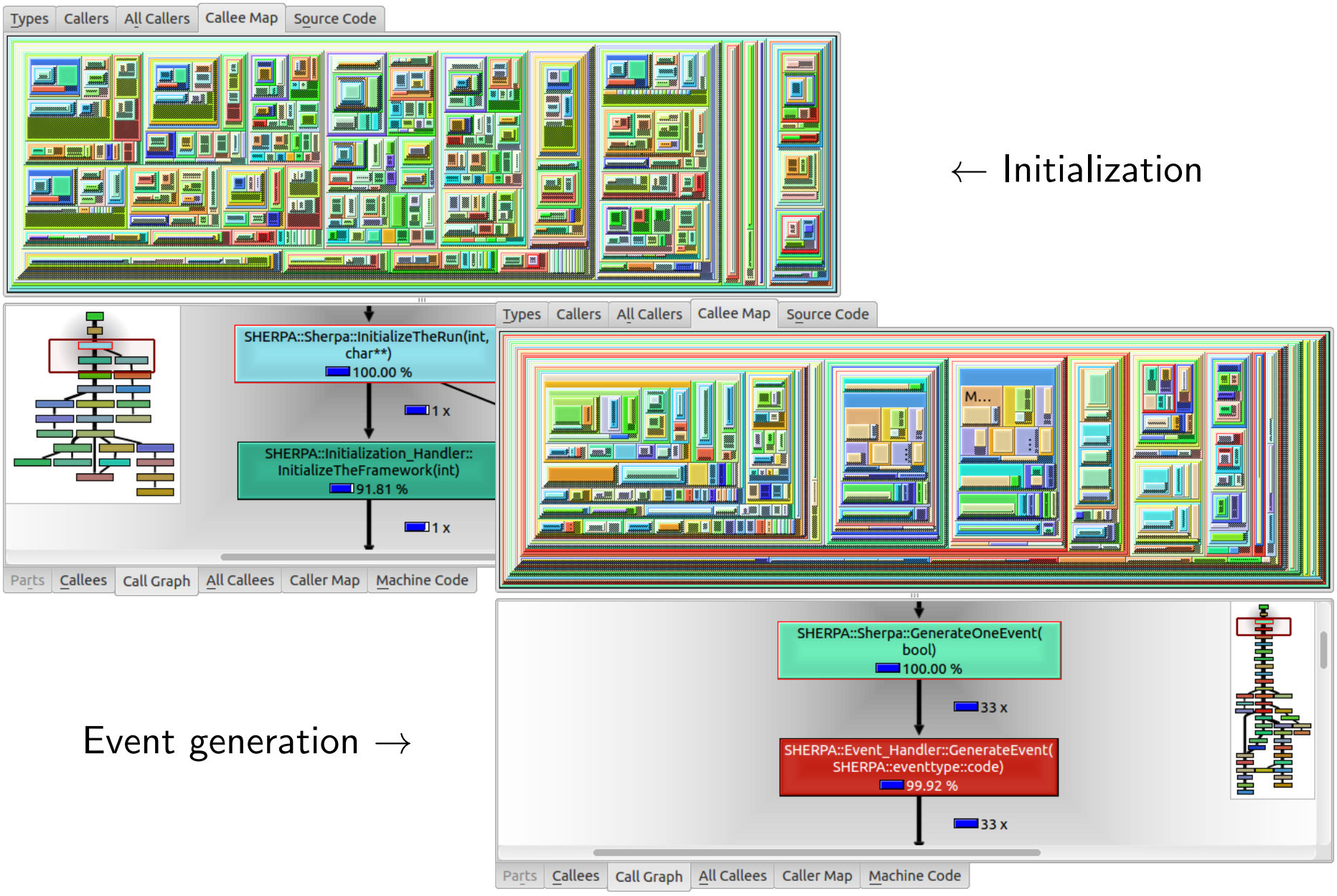
Physics content of Sherpa

[Gleisberg,Krauss,Schönherr,Schumann,Siegert,Winter,SH] arXiv:0811.4622
[Bothmann,Krauss,Kuttimalai,Li,Schönherr,Schulz,Schumann,Siegert,Zapp,SH] soon

- Matrix Element generators
  AMEGIC++ (SM)
  and Comix (SM, BSM)
- Parton shower based on
  Catani-Seymour subtraction
  and new dipole-like shower
- Multiple interaction model
  à la Pythia (non-interleaved)
- In-house cluster hadronization
  and interface to PYTHIA string
  fragmentation (cross-checks!)
- Built-in hadron decay package
  $\approx$ 400 hadrons, $\approx$ 2500 channels
- Photon emission generator
  based on YFS formalism



The caveat of object oriented code



Lessons from Mira - A sampler

SLAC

- **I/O - The usual suspect**
  - Sherpa pre-2.2.0 performs thousands of file operations at startup (hadron decay information, process information & libraries, integrator, ...) causing distress on Mira due to the large number of concurrent processes
  - 1^{st} attempt: Combine all process libraries into one – no luck
  - 2^{nd} attempt: Combine info files into SQLite databases – no luck
  - 3^{rd} attempt: Remove unnecessary stat calls – getting there
  - 4^{th} attempt: Clean up code – better, but still not optimal
- **Code design**
  - Automated codes are stupid, ME generators in particular
  - You can afford being stupid once, but not a million times over
  - 1^{st} attempt: Redesign dipole mapping in Amegic – better
  - 2^{nd} attempt: Redesign process initialization in Sherpa – even better

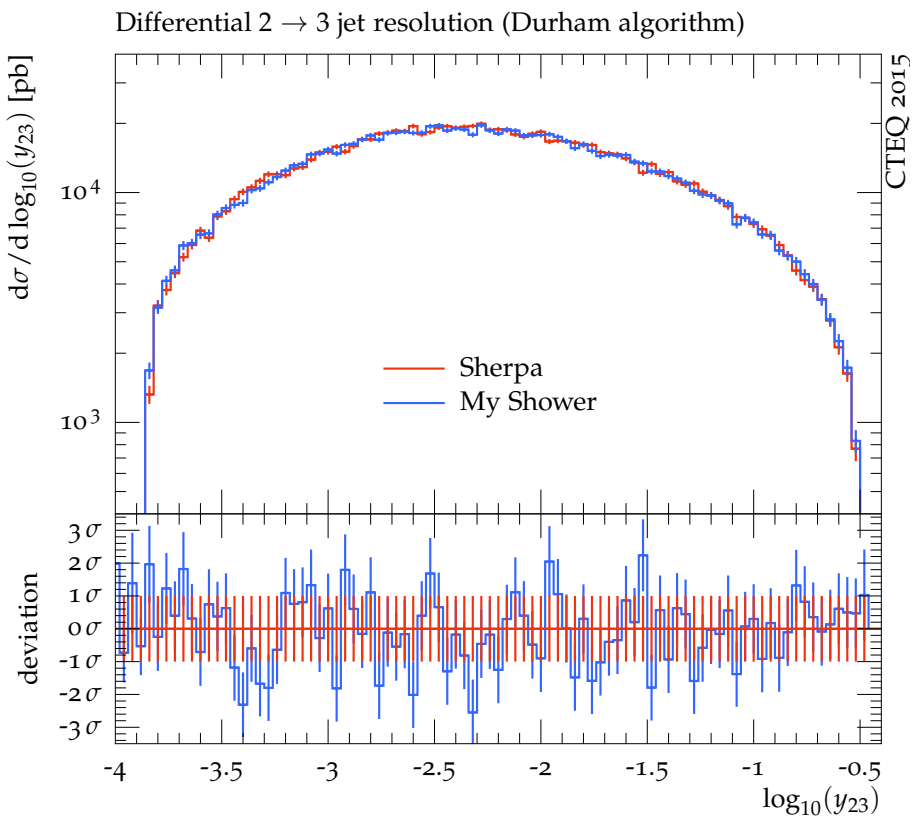
    Example on E5-2699: W+7j (all) 1.4GB, 1.5d to init, 15s to start
    W+8j (all) 5.9GB, 14d to init, 74s to start

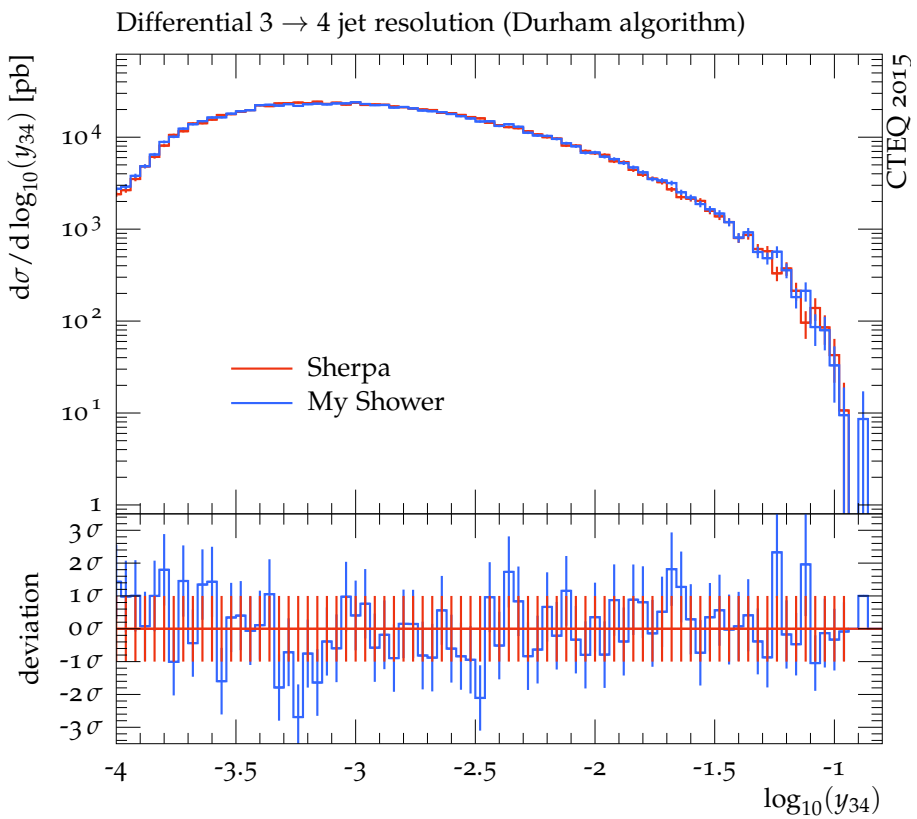
Classical examples of bad engineering and sacrificing speed for flexibility
Issues get worse with decreasing core-by-core performance (BG/Q, KNL!)

System agnostic codes - A case study

SLAC

- Dire parton shower simultaneously developed in Sherpa and Pythia to allow cross-validation [Prestel,SH] arXiv:1506.05057
- Subtle physics issue in e^+e^- →hadrons required third code to settle question which implementation is correct
  → this code now public as a "toy" shower [SH] arXiv:1411.4085





System agnostic codes - A case study

SLAC

- Python shower is very transparent $\rightarrow$ only **139 lines of code**
- Nevertheless it gives the **exact same prediction as Pythia & Sherpa**

Keep in mind that the Python code is tailored to the problem
Keep in mind that the same person has written the Sherpa code

Of course all of this is over-simplified, as LHC is pp, not e^+e^-
But we have an example for a case study at hand, so let's use it!

Platform	E5-2699@2.30GHz PowerEdge R730xd	E5-2660@2.20GHz PowerEdge M620	POWER7@3.7GHz IBM 8205-E6B
Sherpa	0.48ms	0.94ms	1.4ms
Python	2.3ms	5.2ms	9.2ms
PyPy	0.13ms	0.25ms	0.49ms

Seriously? Why am I wasting my time with C++?

Questions

- Do we need a next-to-next generation of MC generators?
  Example: Less object-orientation, more dedicated codes
- When is human brain more costly than computer brain?
  Do we stick to a slow but generic tool or redesign now?
- Should we trade speed for flexibility at programming level?
  Example: C++ vs Python, apparently a non-issue w/ PyPy
- How many different platforms are we talking about?
  Would a generic MC tool set have enough of a user base?