Accelerating PYTHIA8



LUND UNIVERSITY

Leif Lönnblad On behalf of the PYTHIA8 collaboration

Department Physics Lund University

CERN, 2023-11-13

Accelerating PYTHIA8

Leif Lönnblad

Outline

- Introduction
- Recent developments
 - ParallelPythia, Shower variations, biasing
- New developments
 - Hadronisation weights, Onium showers
- Future developments
 - Sectorised merging histories, GPUs,



[arXiv:2203.11601, A comprehensive guide to the physics and usage of PYTHIA8]

Is PYTHIA8 a bottleneck?

٩.	/			allgrind.out.3270 [./test09] — K	Cachegrind
File	View	Go	Settings H	вр	
	Open	< 6	Back ~ >	Forward ~ ^ Up ~ >	Cycle Estimation
Flat	Profile				• @
Sear	ch: Sea	rch Que	sry		ELF Object ~
Set	r e	LF Obje	ct		1
-	64.61 🔳				1
	28.07 🔳	libm-2.	26.so		
	3.23 🔳	libstdc	++.so.6.0.28		
	2.51 🔳	libc-2.2	6.50		
Inc	i. :	Self	Called	Function	1
-	94.56	0.00	999	Pythia8::Pythia::next(int)	
-	79.18	0.05	1 002	Pythia8::PartonLevel::next(P)	thia8::Event&,
	37.48	7.23	144 941	Pythia8::SimpleTimeShower:	pTnext(Pythia8:
	29.61	9.24	14 633 665	Pythia8::SimpleTimeShower:	pT2nextQCD(d
•	27.24	0.84	116 799	Pythia8::SimpleSpaceShower	::pTnext(Pythia
•	25.86	3.08	2 103 878	Pythia8::SimpleSpaceShower	::pT2nextQCD(d
	21.71	12.25	25 243 668	Pythia8::BeamParticle:::dMo	dified(int, int, do
1	11.08	4.96	25 243 668	Pythia8::PDF::xfSea(int, doub b)	le, double)
1	10.03	0.58	368 586	Pythia8::MultipartonInteract	ions::sigmaPT2s
r.	9.63	0.04	1 002	Pythia8::HadronLevel::next(P	ythia8::Event&)
£.,	9.05	0.30	4 015 909	Pythia8::LHAGrid1::xfUpdate	(int, double, do
1	8.85	12.00	4 015 908	Pythia8::LHAGrid1::xfxevolve	(double, double)
	7.95	0.04	116 798	Pythia8::MultipartonInteract	lons::pTnext(do
	6.20	0.14	2 001	Pythia8::ParticleDecays::de	ryAll(Pythia8::Ev
	6.11	0.56		Pythia8::ParticleDecays::de	ny(int, Pythia8::E
	4.98	1.75		Pythia8::Pythia::check()	
	4.77	4.90		Pythia8::BeamParticle:::dMod	
	4.72	0.00		Pythia8::TauDecays::decay(in pythia8::TauDecay)(in pythia8:	t, Pythia8::Even
	4.58	0.00		Pythia8::Pythia::init()	
	4.04	1.30		Pythia8::SigmaMultiparton::	
	3.63	0.00		Pythia8::HelicityMatrixEleme	nt::decayWeight
	3.59	5.55		Pythia8::Rndm::flat()	
	3.24	0.00		Pythia8::HelicityMatrixEleme	1 3
	3.23	0.20	931 892	Pythia8::HelicityMatrixEleme	nt::decayWeight

Accelerating PYTHIA8

In its *default* setup PYTHIA8 is fast.

One (13 TeV $t\bar{t}$) event takes ~10 ms

- \sim 40% Final state shower
- \sim 30% Initial state shower
- \sim 10% Multi parton interactions
- \sim 10% Hadronisation + decays
- \sim 50k random numbers (\sim 6%)
- \sim 4k PDF evaluations (~10%)

Leif Lönnblad

Is PYTHIA8 a bottleneck?

Pie Very Go Settings Hep Open Settings Hep Spect Extination Internation Settings Spect Extination Spect Extination Setting Ext Object Settings Settings Settings 24.00 Immodel Settings Settings Settings Settings 24.00 Immodel Settings Setings Setings <th>٠.</th> <th>,</th> <th></th> <th></th> <th>allgrind.out.3270 [./test09] — K</th> <th>Cachegrind</th>	٠.	,			allgrind.out.3270 [./test09] — K	Cachegrind				
Part Profile Company ELF Object Starth:	File	View	Go	Settings H	elp					
Barth Court Bar Object Seff Ed Object Ed Object Edd Start Bugshaaso 102 102 102 323 Integrate 102 102 102 102 323 Integrate 102 1	Ð	Open	< E	lack ~ >	Forward ~ ~ Up ~ >	Cycle Estimation				
Set LP object 64.61 Higgsprasses 22.87 Binzysprasses 22.87 Binzysprasses 22.87 Binzysprasses 22.87 Binzysprasses 22.87 Binzysprasses 22.87 Binzysprasses 22.88 Binzysprasses 22.91 Binz 2 20.56 22.91 Binz 2 20.57 22.91 Binz 2 20.57 22.91 Binz 2 20.57 22.91 Binz 2 20.57 23.91 Binz 2 20.57 23.91 Binz 2 20.57 23.92 Binz 2 20.57 23.92 Binz 2 20.57 23.92 Binz 2 20.57 23.92	Flat	Profile				· @				
44.64 is its insprises on 44.64 is its insprises on 44.64 is its insprises on the insprise of the insprise of the insprises of the inspr	Sear	ch: Sea	rch Que	ny		ELF Object ~				
44.64 is its insprises on 44.64 is its insprises on 44.64 is its insprises on the insprise of the insprise of the insprises of the inspr	Solt	r e	E Oble	at						
24.00 10m-22.00 3.32 #ibide/1-50.0.28 2.33 #ibide/1-50.0.28 2.34 #ibide/1-50.0.28 2.35 #ibide/1-50.0.28 2.34 #ibide/1-50.0.28 2.35 #ibide/1-50.0.28 2.34 #ibide/1-50.0.28 2.35 #ibide/1-50.0.28 2.34 #ibide/1-50.0.28 2.35 #ibide/1-50.0.28 7.47 7.31 #ibide/1-50.0.28 2.74 7.31 #ibide/1-50.0.28 2.74 9.24 #ibide/1-50.00000000000000000000000000000000000										
32.3 a librative - sa 0.23 23.9 a										
Incl. Self Called Procision 9 456 0.00 999 ph/shatz-ph/staznact(rot) 9 748 7.01 1.00 ph/shatz-ph/staznact(rot) 9 749 7.31 1.44 918 ph/shatz-filter/staznact(rot) 9 749 7.31 1.44 918 ph/shatz-filter/staznact(rot) 9 749 9.71 1.51 919 ph/shatz-filter/staznact(rot) 1 10 4.01 100 99 ph/shatz-filter/staznact/staznact(rot) ph/shatz-filter/staznact/staz										
9438 0.0 999 phtats:Pythats:rest(nt) 9748 0.0 10.2 phtats:Pythats:rest(nt) 9749 phtats:Pythats:rest(nt) 10.2 phtats:Pythats:rest(nt) 9749 9.2 14.03.06 phtats:Pythats:rest(nt) 9749 9.2 14.03.06 phtats:Pythats:rest(nt) 9749 9.2 14.03.06 phtats:SimpleTimeShower;pTractCOL 9749 10.2 phtats:SimpleTimeShower;pTractCOL phtats:SimpleTimeShower;pTractCOL 9749 10.2 9.7514.65 phtats:Pythats:Rest(nt) nt) 9749 10.2 phtats:Rest(nt) nt) nt) 9749 10.2 phtats:Rest(nt) nt) nt) 9749 10.2 phtats:Rest(nt) nt) nt) 9749 phtats:Rest(nt) nt) nt) nt) nt) 9750 10.5550 phtats:Rest(nt) nt)		2.51 🔳	libc-2.2	6.50						
71.11 0.05 1032 pythat2-threat-net-surghythat2-threat, 37.46 3.24 1033 pythat2-threat-net-surghythat2-threat, 37.47 3.24 1033 pythat2-threat-net-surghythat2-threat, 37.48 1.23 14.631.06 pythat2-threat-net-surghythat2-threat, 37.48 1.24 10.67 pythat2-threat-surghythat2-threat/threat-net-surghythat2-threat, 37.48 1.24 10.67 pythat2-threat-surghythat2-threat/threat-net-surghythat2-threat, 1.25 2.24 9.47 10.07 pythat2-threat-surghythat2-threat/threat-net-surghythat2-threat, 1.10 4.05 2.244 pythat2-threat-surghythat2-threat, 9.21 0.24 4015.059 pythat2-threat, 9.23 0.41 1.07 pythat2-threat, 9.24 1.07 pythat2-threat, threat, 9.25 2.07 pythat2-threat, threat, 9.20 2.17 pythat2-threat, threat, 9.21 2.07 pythat2-threat, threat, 9.21 2.07 pythat2-threat,<	Incl	. :	Self	Called	Function					
37.48 7.23 14.911 pythal:Simplifind:Simplifind:Hower:pTrackip/dials 2.94.01 9.24 14.933.06 pythal:Simplifind:Hower:pTrackip/dials 2.74 0.44 16.79 pythal:Simplifind:Hower:pTrackip/dials 2.74 0.44 16.79 pythal:Simplifyind:Hower:pTrackip/dials 2.74 0.44 16.79 pythal:Simplifyind:Hower:pTrackip/dials 2.71 12.75 22.534.06 pythal:Simplifyind:Hower:pTrackip/dials 1.71 12.75 22.534.06 pythal:Simplifyind:Hower:pTrackip/dials 0.80 0.52 25.436.66 pythal:Simplifyind:Hower:pTrackip/dials 0.81 0.02 pythal:Simplifyind:Hower:ptrackip/dials Pythal:Simplifyind:Hower:Ptrackip/dials 0.81 10.02 pythal:Simplifyind:Hower:Ptrackip/dials Pythal:Simplifyind:Hower:Ptrackip/dials 0.82 0.03 40.150.99 pythal:Simplifyind:Hower:Ptrackip/dials Pythal:Simplifyind:Hower:Ptrackip/dials 0.83 10.07 pythal:Simplifyind:Hower:Ptrackip/dials Pythal:Simplifyind:Hower:Ptrackip/dials Pythal:Simplifyintrall/Ptral:Simplifyind:Hower:Ptral:Ptrackip/dials <	-	94.56	0.00	999	Pythia8::Pythia::next(int)					
2941 9.24 443365 phtals:SimplifTikedSumm:pTinet(SQL) 2724 0.44 116779 phtals:SimplifTikedSumm:pTinet(SQL) 2736 0.44 116779 phtals:SimplifTikedSumm:pTinet(SQL) 2171 1225 224368 phtals:SimplifTikedSumm:pTinet(SQL) 1108 49 phtals:SimplifTikedSumm:pTinet(SQL) 1003 0.35 234368 phtals:SimplifTikedSumm:pTinet(SQL) 1014 49 234368 phtals:SimplifTikedSumm:pTinet(SQL) 1015 0.35 345368 phtals:SimplifTikedSumm:pTinet(SQL) 930 0.41 10528 phtals:SimplifTikedSummer,pTinet(SQL) 935 0.41 105788 phtals:SimplifTikedSummer,pTinet(SQL) 936 0.41 12078 phtals:SimplifTikedSummer,pTinet(SML) 110 2016 phtals:SimplifTikedSummer,Sml (SML) phtals:SimplifTikedSummer,Sml (SML) 120 0.12017 phtals:SimplifTikedSummer,Sml (SML) phtals:SimplifTikedSummer,Sml (SML) 120 0.12017 phtals:SimplifTikedSummer,Sml (SML) phtals:SimplifTikedSumer,Sml (SML) 1	-	79.18	0.06	1 002	Pythia8::PartonLevel::next(Py	thia8::Event&,				
22.4 0.44 116 739 phtals-SingleSpectRoversyThroutPittle 25.6 1.0 21.03 738 phtals-Bear/Bear/Bear/Bear/Bear/Bear/Bear/Bear/		37.48	7.23	144 941	Pythia8::SimpleTimeShower:	pTnext(Pythia8:				
2586 3.01 2133 779 pythall:SimplifypaceShower:p172wet2Cptd. 21737 1223 224 568 pythall:SimplifypaceShower:p172wet2Cptd. 1104 456 224 568 pythall:SimplifypaceShowEr:p172wet2Cptd. 100 456 254 568 pythall:SimplifypaceShowEr:p172wet2Cptd. 900 0.03 364 558 pythall:SimplifypaceShowEr:p172wet2Cptd. 901 0.03 364 558 pythall:SimplifypaceShowEr:p172wet2Cptd. 902 0.04 1002 pythall:SimplifypaceShowEr:p172wet2Cptd. 903 0.04 1002 pythall:SimplifypaceShowEr:p172wet2Cptd. 903 0.04 105 598 pythall:SimplifypaceShowEr:p172wptd. 101 0.04 105 199 pythall:SimplifypaceShowEr:p172wptd. 101 0.01 pythall:SimplifypaceShowEr:p172wptd. pythall:SimplifypaceShowEr:p172wptd. 102 0.14 2010 78 pythall:SimplifypaceShowEr:p172wptd. 103 0.14 2010 78 pythall:SimplifypaceExecutions(1174wptd. 103 210 78 pythall:Executions(1174wptd. pythall:Execution (1174wptd.		29.61	9.24	14 633 665	Pythia8::SimpleTimeShower:	pT2nextQCD(d				
17.1 12.5 12.5 12.6 <td< td=""><td></td><td>27.24</td><td>0.84</td><td>116 799</td><td>Pythia8::SimpleSpaceShower</td><td>::pTnext(Pythia</td></td<>		27.24	0.84	116 799	Pythia8::SimpleSpaceShower	::pTnext(Pythia				
11.08 4.59 25.241.668 p-Opt-sale-DPF-sd5eapler, double, double, 10.00 0.38 366.568 p-Opt-sale-DPF-sd5eapler, double, double, 0.00 0.30 4015.079 p-Opt-sale-Def-sd6eapler, double, double, 0.00 0.30 4015.079 p-Opt-sale-Def-sd6eapler, double, double, 0.01 0.015.0756 p-Opt-sale-Def-sd6eapler, double, double, 0.02 0.015.0756 p-Opt-sale-Def-sd6eapler, double, double, 0.01 107.078 p-Opt-sale-Def-sd6eapler, double, double, 0.02 0.019 p-Opt-sale-Def-sd6eapler, double, double, 0.01 1.075 p-Opt-sale-Def-sd6eapler, double, double, 0.02 2.019 p-Opt-sale-Def-sd6eapler, double, double, 0.03 0.019 p-Opt-sale-Def-sd6eapler, double, double, 0.02 2.724 p-Opt-sale-Def-sd6eapler, double, 0.03 2.7259 p-Opt-sale-Def-sd6eapler, double, 0.04 2.7259 p-Opt-sale-Def-sd6eapler, double, 0.05 2.7259 p-Opt-sale-Def-sd6eapler, double, 0.04 0.019 P-Opt-sale-Def-sd6eapler, double, <td></td> <td>25.86</td> <td>3.08</td> <td>2 103 878</td> <td>Pythia8::SimpleSpaceShower</td> <td>::pT2nextQCD(d</td>		25.86	3.08	2 103 878	Pythia8::SimpleSpaceShower	::pT2nextQCD(d				
10.03 0.53 364.556 p/s/stat:2.Mdbpartor/street/street.sgmaPTzL. 9.65 0.04 1.052 p/s/stat:2.Mdbpartor/street/p/stat:2.vers/i) 9.65 0.05 p/s/stat:2.vers/in/stat:2.vers/i) 9.66 0.055 p/s/stat:2.vers/in/stat:2.vers/i) 9.66 1.052 p/s/stat:2.vers/in/stat:2.vers/in/stat:2.vers/i) 9.66 1.057 p/s/stat:2.vers/in/stat:2.vers/in/stat:2.vers/ii 0.01 1.057 p/s/stat:2.vers/in/stat:2.vers/ii 0.02 2.244 p/stat:2.vers/in/stat:2.vers/ii 0.03 1.0717 p/shat:3.vers/in/stat:2.vers/ii 0.03 1.0717 p/shat:3.vers/in/stat:2.vers/ii 0.03 1.0717 p/shat:3.vers/in/stat:2.vers/ii 0.04 1.0717 p/shat:3.vers/in/stat:2.vers/ii <td>•</td> <td>21.71</td> <td>12.25</td> <td>25 243 668</td> <td>Pythia8::BeamParticle:::dMod</td> <td>lified(int, int, do</td>	•	21.71	12.25	25 243 668	Pythia8::BeamParticle:::dMod	lified(int, int, do				
9.53 0.64 1.032 PythalE34d/ord.netE.net(PythalE2Wert(b)) 9.05 0.03 0.015 002 PythalE34d/ord.netE.net(PythalE2Wert(b)) 9.06 0.055 002 PythalE34d/ord.netE.net(PythalE6.040H) 9.07 0.04 1.057 002 PythalE34d/ord.netE.net(PythalE6.040H) 0.06 1.057 002 PythalE34d/ord.netE.net(PythalE34E) 0.014 0.01 1.050 2.017 PythalE34d/ord.netE.net(PythalE34E) 0.01 1.050 2.017 PythalE34d/ord.netE.net(PythalE34d/ord.netE.net(PythalE34E) 0.01 1.051 2.015 PythalE34d/ord.netE.net(PythalE34	1									
9.05 0.04 0.05 0.09 physical:U4004137:dbp4erpti, double, double, 8.85 1.02 0.0155908 physical:U4004137:dbp4erpti, double, double, 9.86 1.02 0.0155908 physical:U4004137:dbp4erpti, double, double, 0.00 0.14 0.018 physical:U4004137:dbp4erpti, double, double, 0.01 0.02 physical:U400497:dbp4erpti, double, double, physical:U400497:dbp4erpti, double, 0.01 0.02 20107 physical:U400497:dbp4erpti, double, double, physical:U400497:dbp4erpti, double, 0.01 0.01 27157559 physical:U400497:dbp4erpti, double, physical:U400497:dbp4erpti, double, physical:U400497:dbp4erpti, double, 0.02 2745 physical:U400497:dbp4erpti, double, physical:U400497:dbp4erpti, double, physical:U401497:dbp4erpti, double, 0.03 2745 physical:U400497:dbp4erpti, double, physical:U401497:dbp4erpti, double, physical:U401497:dbp4erpti, double, 0.03 2344:dbg3 physical:U401497:dbp4erpti, double, physical:U401497:dbp4erpti, double, physical:U401497:dbp4erpti, double, 0.03 234:dbg3 physica:U401497:dbp4erpti, double, physic	1				- /					
BID 12:00 4015 993 phthatE.UMGod12.sfeerodwegdouble, doublej 73:00 0.04 116 798 phthatE.MdGod12.sfeerodwegdouble, doublej 0.01 116 798 phthatE.MdGod12.sfeerodwegdouble, doublej 0.01 0.01 phthatE.MdGod12.sfeerodwegdoublej 0.01 0.01 phthatE.MdGod22.seerodwegdoublej 0.01 0.01 phthatE.MdGod22.seerodwegdoublej 0.01 27.010 phthatE.MdGod22.seerodwegdoublej 0.01 27.010 phthatE.MdGod22.seerodwegdoublej 0.01 27.012 phthatE.MdGod22.seerodwegdoublej 0.01 27.014 phthatE.MdGod22.seerodwegdoublej 0.01 27.014 phthatE.MdGod22.seerodwegdoublej 0.01 27.014 phthatE.MdGod22.seerodwegdoublej 0.01 0.01 phthatE.MdGod22.seerodwegdoublej 0.01 0.01 phthatE.MdGod22.seerodwegdoublej 0.02 27.41 phthatE.MdGod22.seerodwegdoublej 0.01 0.01 0.01 phthatE.MdGod22.seerodwegdoublej 0.02 0.01 0.01 phthatE.MdGod22.seerodwegdod22.	1				- , .					
739 0.04 116.788 p-tybale=ZhotGhotGhotGhotGhotGhotGhotGhotGhotGhotG	1									
6.20 0.14 2.001 phthatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespeckeespki(pythatE-bruckbeespki(pythatE-br	1									
6.11 0.56 277.017 phthatEparticleAnayscidecaylorit. PhthatELE 400 1.75 979.88 phthatEphtatEchted) 400.016 477 440 273.598 phthatEshearmhiteEchtedDetergetrit. double) 470 400 274.89 phthatEshearmhiteEchtedDetergetrit. double) 470 400 274.89 phthatEshearmhiteEchtedDetergetrit. double) 470 400 9.9 phthatEshearmhiteEchtedDetergetrit. double) 471 400 100 324.685 phthatEshearmhiteEchtedDetergetrit. double) 472 400 324.685 phthatEshearmhiteEchtedDetergetrit. 400.01717 473 5.00 107.172 phthatEshearmhiteEnthetDetergetrit. 400.01711 474 5.00 49.714444848464mftftt) 400.01711 400.01111 400.01111										
4.15 9.77 phthatE-typital_threft 4.77 4.00 2.73.59 phthatE-threft 4.77 4.00 2.74 phthatE-threft 4.70 0.00 (0) phthatE-threft 4.70 0.00 (0) phthatE-threft 4.71 0.00 (0) phthatE-typital_threft 4.72 0.00 (0) phthatE-typital_threft 4.73 3.24 68 phthatE-typital_threft threft 4.74 1.03 3.24 68 phthatE-typital_threft 4.75 1.07.71 phthatE-typital_threft threft 5.75 49.7514-65.75 phthatE-typital_threft threft										
4.77 4.90 3273 559 g-lythal:BeamPanticle:xtModPreptint, double) 4.70 0.00 274 g-lythal:ElemenPanticle:xtModPreptint, double) 4.90 0.00 274 g-lythal:ElemenPanticle:xtModPreptint, double) 4.91 0.00 9 g-lythal:ElemenPanticle:xtModPreptint, double) 4.04 1.30 324 665 g-lythal:Element:stargmanute, int, do 3.60 0.0717 g-lythal:Element:stargmanute, int, do 365 3.55 9.572 627 g-lythal:Element:stargmanute, int, do 365						y(int, Pythias::E				
4.72 0.00 274 g Pytha8:TauDecays:edecay(rt, Pytha8:Even 4.58 0.00 (i) g Pytha8:Evpthacitet() 4.64 1.30 324.665 g Pytha8:EsignaMultipartor:signa(int, int, do 3.53 0.00 1077 g Pytha8:Evelocitet() 3.59 5.55 49 572.637 g Pytha8:EvelocithartorElement:decayWeight						(Pren/int double)				
4.58 0.00 (0) # Pythal8:Srythia:Linit[] 4.64 1.30 324.685 # Pythal8:SrythiasLinit[] 3.63 0.00 10.717 # Pythal8:SrythiasLinit[] 3.63 0.00 10.717 # Pythal8:SrythiasLinit[] 3.59 5.55 49.572.637 # Pythal8:Endom/MatrixElement::decayWeight										
4.04 1.30 324 685 p-Ythia8::SigmaMultiparton:sigma(int, int, do 3.63 0.00 10 717 p-Ythia8::HelidtyMatrixElement::decayWeight 3.59 5.55 49 572 637 p-Ythia8::Rndm::flat)						c, r yonoo.cven				
3.63 0.00 10 717 ■ Pythia8::HelicityMatrixElement::decayWeight 3.59 5.55 49 572 637 ■ Pythia8::Rndm::flat]					- / / -	iomalint int do				
3.59 5.55 49 572 637 B Pythia8::Rndm::flat()										
3.24 0.00 10 716 Pythia8::HelicityMatrixElement::decayWeight		3.59			- , ,	,				
		3.24	0.00	10 716	Pythia8::HelicityMatrixEleme	nt::decayWeight				
3.23 0.20 931 892 B Pythia8::HelicityMatrixElement::decayWeight		3.23	0.20	931 892	Pythia8::HelicityMatrixEleme	nt::decayWeight				

In its *default* setup PYTHIA8 is fast.

One (13 TeV $t\bar{t}$) event takes ~10 ms

- \sim 40% Final state shower
- $\sim~$ 30% Initial state shower
- \sim 10% Multi parton interactions
- \sim 10% Hadronisation + decays
- \sim 50k random numbers (\sim 6%)
- \sim 4k PDF evaluations (\sim 10%)
- < 2% ME generation

Every step in PYTHIA8 depends on what happened in the previous step, so parallelisation is in general not possible.

Even where parallelisation is possible, the steps typically involves a lot of admin code that is not suited for e.g. GPUs.

But default PYTHIA8 is fast so no problem!



Paralellisation

Every event is independent, so PYTHIA8 itself can be parallelised

- Everything is perfectly thread safe
- OpenMP and HDF5 LHEF supported
- New wrapper class using std::thread



ParallelPythia

```
# from examples/main161.cc
#include "Pythia8/Pythia.h"
#include "Pythia8/PythiaParallel.h"
using namespace Pythia8;
int main() {
 // Use the PythiaParallel class for parallel generation.
 PvthiaParallel pvthia;
 pythia.readString("Beams:eCM = 8000.");
 pythia.readString("HardQCD:all = on");
 pythia.readString("PhaseSpace:pTHatMin = 20.");
 pythia.init();
 Hist mult("charged multiplicity", 100, -0.5, 799.5);
 // Use PythiaParallel::run to generate the specified number of events.
  pythia.run(10000, [&] (Pythia* pythiaPtr) {
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0:
    for (int i = 0; i < pythiaPtr->event.size(); ++i)
     if (pythiaPtr->event[i].isFinal() && pythiaPtr->event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
    // End of event loop. Statistics. Histogram. Done.
  }):
 pythia.stat();
  cout << mult;
  return 0:
```

Leif Lönnblad

Shower variations

With multiple weights per event we can save a lot of CPU time for scale variations etc.

Scale variations is easy for MEs. Showers are more tricky.

We have splitting functions and no emission functions

$$P(p_{\perp}^2) imes \exp\left(-\int_{p_{\perp}^2}^{p_{\perp,\mathrm{prev}}^2} dk_{\perp}^2 P(k_{\perp}^2)
ight)$$

 Using the veto algorithm we oversample with a simple overestimate function and throw to get the correct distribution.

[arXiv:1605.08352]

Accelerating PYTHIA8

Leif Lönnblad



Every time we accept/reject we can get weights for different variations,

$$\mathcal{P}(\mathcal{p}^2_{\perp}, lpha_{s}(\mu_{R}))
ightarrow \mathcal{P}(\mathcal{p}^2_{\perp}, lpha_{s}(\mathcal{C}\mu_{R})) \quad \Rightarrow \quad \mathcal{p}_{\mathsf{acc}}
ightarrow \mathcal{p}'_{\mathsf{acc}}, \mathcal{p}_{\mathsf{rej}}
ightarrow \mathcal{p}'_{\mathsf{rej}},$$

accept/reject according $\rho_{\mbox{\tiny acc}}$ and $\rho_{\mbox{\tiny rej}},$ and get an event weight

$$w_{ev} = \prod_i \frac{p'_i}{p_i}$$

Note that large reweighting factors may result in fluctuating weights, so the statistical uncertainty will increase:

$$N_{\rm ev, \, eff} = rac{(\sum w)^2}{\sum w^2}$$

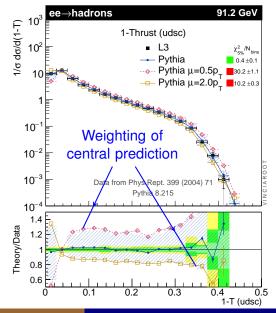


- μ_R scale for QCD emissions in FSR
- µ_R scale for QCD emissions in ISR
- inclusion of non-singular terms in QCD emissions in FSR
- inclusion of non-singular terms in QCD emissions in ISR
- PDF members of a PDF family in LHAPDF6
- individual PDF members of a PDF family in LHAPDF6



Introduction Recent deveopments

New developments





Accelerating PYTHIA8

Biasing

The same procedure may be used to bias your events, e.g. to get more $g \to b \bar{b}$

$$P_{bar b}^\prime(p_\perp^2)=CP_{bar b}(p_\perp^2)$$

but now the rôle of variation and standard setting are reversed, so we accept/reject according to $p'_{\rm acc}$ and $p'_{\rm rej}$, and get an event weight

$$w_{ev} = \prod_i rac{
ho_i}{
ho_i'}$$



Biasing works well for low probability processes, but for more common processes we are hit by large weight fluctuations.

The statistical uncertainty of an observable scales like a power of the inverse no-emission probability, Δ^{-h}

For an *n*-emission observable we get an uncertainty

$$\delta \mathcal{O}_n \propto \frac{\Delta^{-h}}{\sqrt{C^n}}, \quad \text{with} \quad h \approx C$$

You cannot bias MPI.



Leif Lönnblad

Hadronisation weights

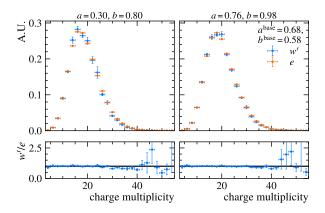
In addition, weight variations are soon also available for hadronisation parameters.

Works on the same principle as shower variations (with the same caveats), but this time modifying hit-and-miss algorithms for e.g. the Lund symmetric fragmentation function.

$$p(z) \propto rac{(1-z)^a}{z} e^{-bm_{\perp}^2/z}$$



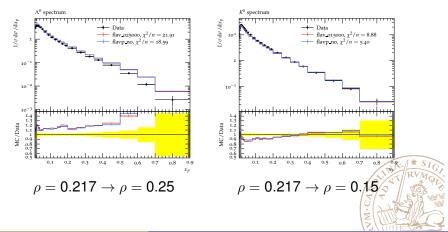
[arXiv:2308.13459]



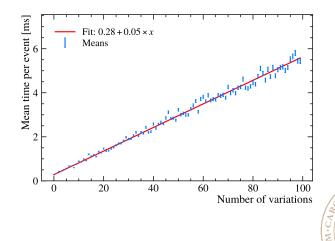


Flavour variations

 $\rho = \texttt{StringFlav:probStoUD}$



Timing



Leif Lönnblad

This is mainly useful for tuning

But again, if you can do weight variations you can do biasing (caveats still applies)





Onia showers

PYTHIA8 now includes shower splittings into charmonium and bottomonium splittings, e.g. $c \to J/\psi + c$ and $g \to O({}^3P_J) + g$

(it's unfortunately a bit slow)

Biasing with a constant factor available.



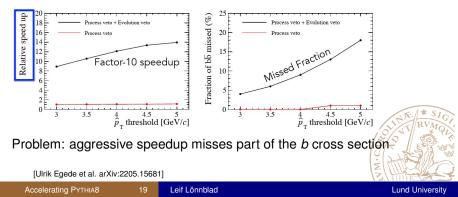
[available since version 8.310]

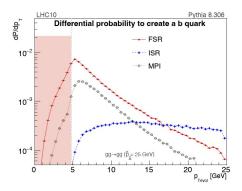
Leif Lönnblad

Biasing heavy flavours

An alternative to biasing is to simply try to veto uninteresting events at a very early stage.

Using UserHooks you can tell PYTHIA8 to veto an event, e.g. when the evolution scale has dropped below some scale, \hat{p}_T .





- ISR has the expected behaviour, FSR and MPI not.
- WIP: reformulate shower to have a more physical evolution scale so that all *b*s are produced at scales above m_b.



Lund University

Beyond the default PYTHIA8

While the default PYTHIA8 is fast, we sometimes need speed-up.

In addition PYTHIA8 includes important optional modules which are quite CPU demanding, e.g.

- Hadronic rescattering has complexity N²_h (CPU×2)
- QCD-based Colour reconnections, N²_d or even N³_d (CPU×15)
- CKKW-L type merging: N_l!



ME+PS merging

For CKKW-L style merging (UMEPS, NL3, UNLOPS, ...) you need to take all contributing shower histories into account for a given ME state.

In conventional PS each possible phase-space point receives contributions from many possible branching "histories" (or "clusterings"). Approximately sums over (singular) diagrams to give the full singularity structure.

# branchings	1	2	3	4	5	6	7			
# histories	2	8	48	384	3840	46080	645120	F.1 * SI		
# histories \sim # feynman diagrams grow faster than factorial \sim										
							N.C.			

Sector Showers

The default VINCIA shower in PYTHIA8 is unique in being a "Sector Shower"

- Divide N-gluon Phase Space into N "sectors", with step functions.
- Each PS sector corresponds to one specific gluon being the "softest" in the event — the one you would cluster if you were running a jet algorithm (specifically one called ARCLUS)
- Inside each sector, only a single kernel is allowed to contribute (the most singular one)!

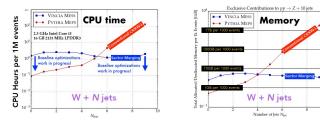
Sector Kernel = the eikonal for the soft gluon and its collinear DGLAP limits for z > 1/2.

Only a single (product of) kernel(s) contributes to each phase-space point. $N! \rightarrow 1!$

[Skands & Villajero arXiv:1109.3608, Brooks, Preuss, Skands arXiv:2003.00702]

Sector Merging with VINCIA Sector Shower

Available since PYTHIA 8.306



Extentsons are coming:

- Sectorized matching at NNLO (proof of concept: arXiv:2108.07133, arXiv:2310.18671)
- Sectorized iterated tree-level ME corrections (demonstrated in arXiv:1109.3608)
- Sectorized multi-leg merging at NLO

[Brooks & Preuss arXiv:2008.09468]



Preview: VINCIA NNLO+PS for $H \rightarrow b\bar{b}$

Coloretti, Gehrmann-de Ridder, Preuss, JHEP 06 (2022) 009

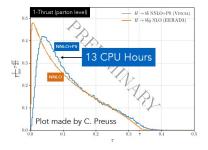
Fixed-Order Reference = EERAD3 NLO $H \rightarrow b\bar{b}g$: already Highly optimised



Uses analytical MEs, "folds" phase space to cancel azimuthally antipodal points, and uses antenna subtraction (\rightarrow smaller # of NLO subtraction terms than Catani-Seymour or FKS).

VINCIA NNLO + PS: uses the shower as phase-space generator: extremely efficient & everything formulated to be positive definite \implies no negative weights

> About factor 5 faster than EERAD3 (for comparable unweighted stats) + can be hadronised, etc.



Note:

NNLO accuracy in $H \rightarrow 2j$ implies NLO correction in first emission and LO correction in second emission.



Expectation: VINCIA NLO MEC approach order-ofmagnitude faster than anything less optimised than EERAD3

GPU plans

Both hadronic rescattering and QCD-based colour reconnections can be improved algorithmically.

QCD-CR actively being worked on with M. Kreps (Warwick): Purely code-based modifications may result in speed increases of factors 2-3.

But some of the N^2 and N^3 calculations should be possible to parallelise and be delegated to a GPU.

This would in particular be important for heavy ion collisions, where $N \sim O(1000)$ are common

Conclusions

PYTHIA8 may not be an important bottleneck, but there are improvements than can be made and are being made.





Other Thoughts (from Peter Skands)

Optimisation also crucial to reduce computational footprint / environmental impact

But funders do not (currently) score on this criterion at all

E.g., ERC allow a "Do No Significant Harm" statement — but assessors told **(in boldface)** to ignore it ARC does not even have such a statement. Not sure about other agencies… ?

Tricky choice if one has to compromise on scientific ambition? Some thoughts on this in "Computational scientists should consider climate impacts and grant agencies should reward them", <u>P Skands, Nature</u> <u>Rev.Phys. 5 (2023) 3, 137-138</u>

All grants I am connected with now include minimisation of footprint as explicit goal

ARC DP22 "Tackling the computational bottleneck in precision particle physics" — on sector-based approaches

ARC DP23 "Beautiful Strings" — on more efficient (and better) models of heavy flavour production, fragmentation, and decays (incl matching), QED showers in hadron decays, collective effects in fragmentation, and Colour Reconnections:

POST DOC AT MONASH NOW OPEN FOR APPLICATIONS

Monash-Warwick Alliance for Particle Physics: including optimisation and improvements to EVTGEN and PYTHIA for HF physics (incl QED showers and QCD CR)

Royal Society Wolfson Visiting Fellowship "Piercing the precision barrier in high-energy particle physics": to develop efficient techniques for NNLO matching and beyond + interact with PanScales and with Warwick on (multithreaded) multipole QED showers in hadron decays

DECRA 23 [L. Scyboz, Monash]: "Bridging the accuracy gap: High-precision parton showers for colliders" — on PanScales and VINCIA.