Update on EvtGen – the new 3.0.0-beta release

Fernando Abudinén, John Back, Michal Kreps, Thomas Latham

MONASH University

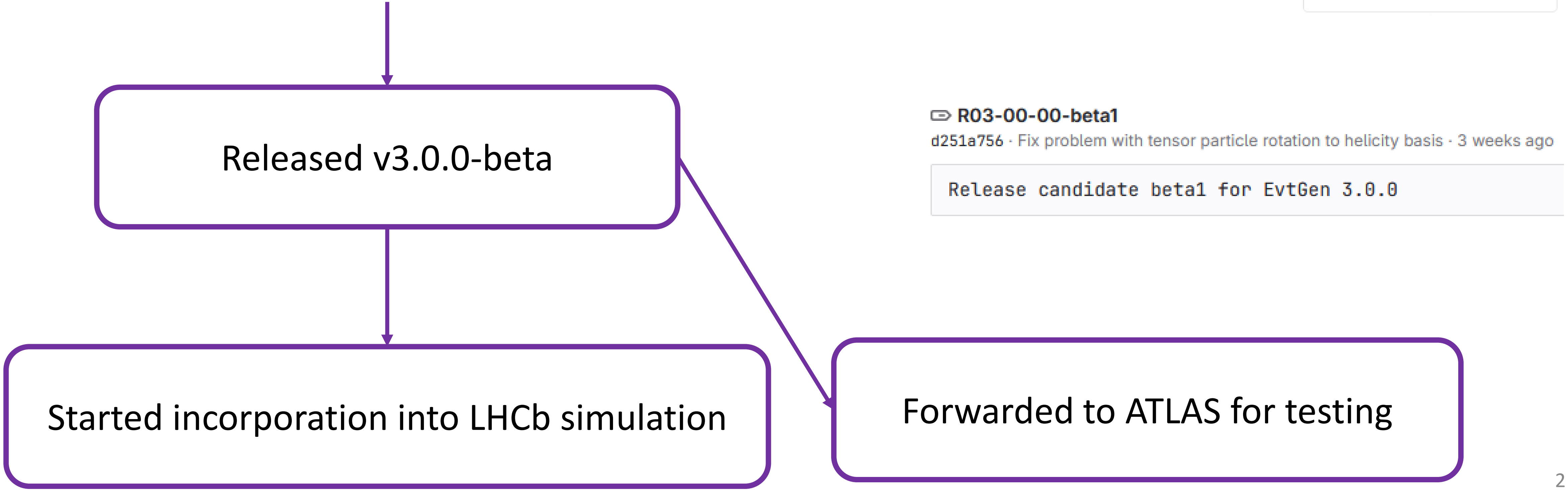
WARWICK
THE UNIVERSITY OF WARWICK

SWIFTHEP Workshop
November 12, 2024

News

- Propagated thread safety across the whole software framework
- Implemented multi-threading in testing framework
- Added tests for external dependencies
- Fixed bug with tensor particle rotation to helicity basis

Check it out!



Released v3.0.0-beta

Started incorporation into LHCb simulation

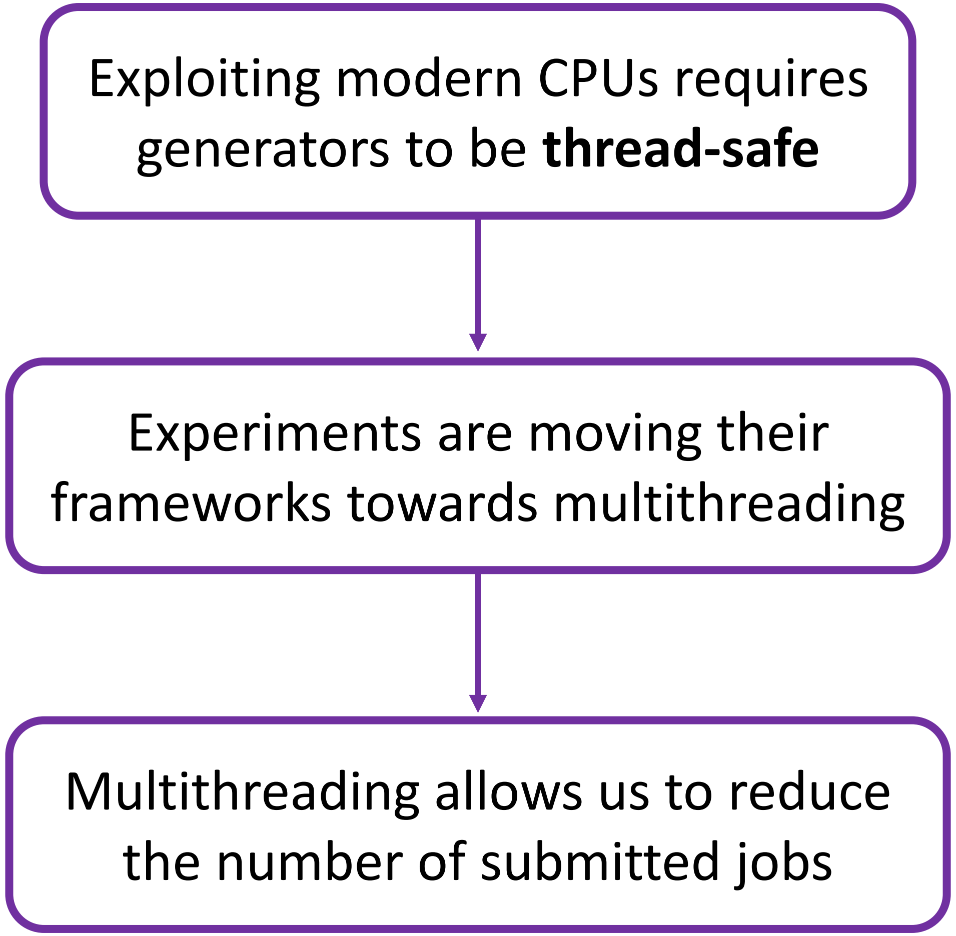
Forwarded to ATLAS for testing

R03-00-00-beta1

d251a756 · Fix problem with tensor particle rotation to helicity basis · 3 weeks ago

```
Release candidate beta1 for EvtGen 3.0.0
```

The need for thread safety

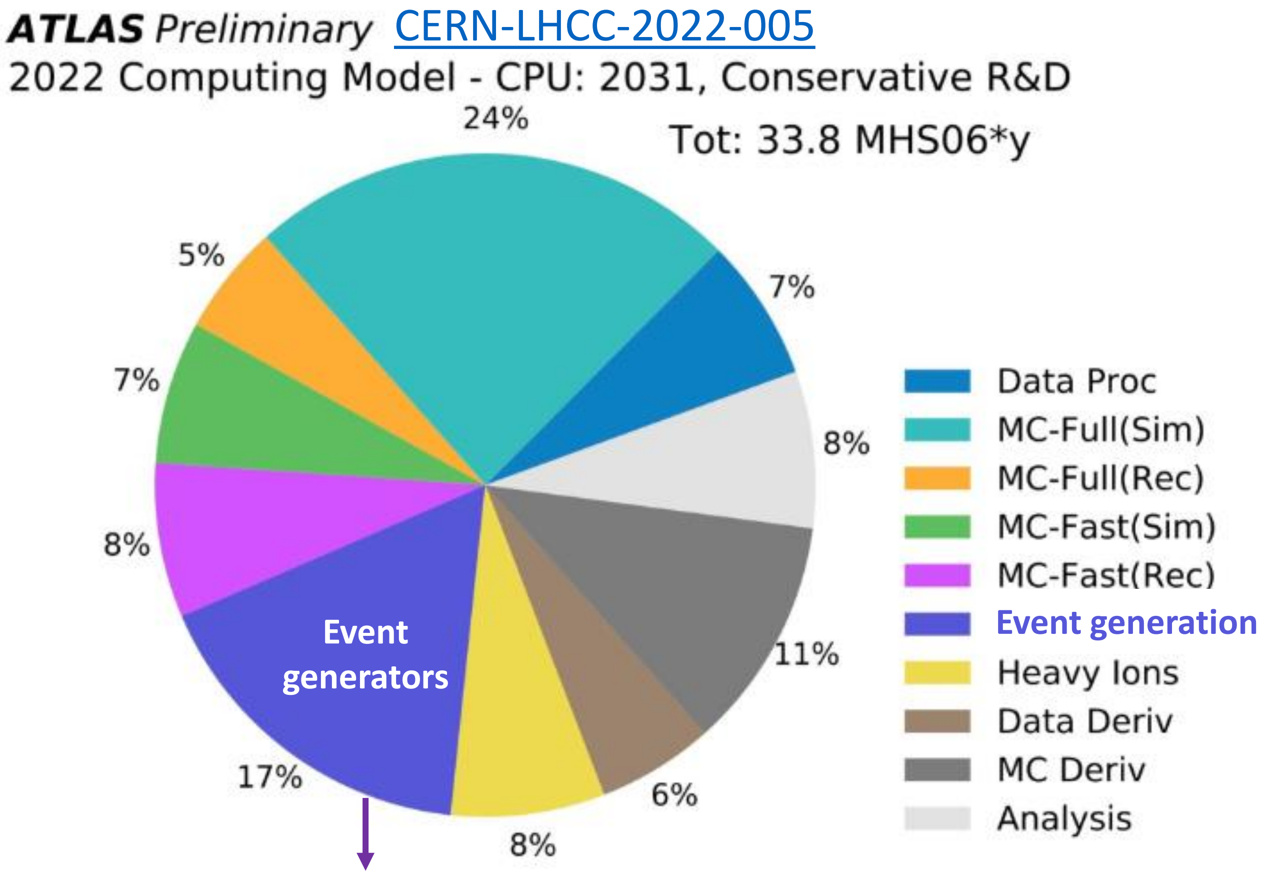


Exploiting modern CPUs requires generators to be **thread-safe**

Experiments are moving their frameworks towards multithreading

Multithreading allows us to reduce the number of submitted jobs

Projected CPU consumption at the ATLAS experiment



EvtGen is not the major offender among event generators but should anyway evolve to enable multithreading

EvtGen in a nutshell

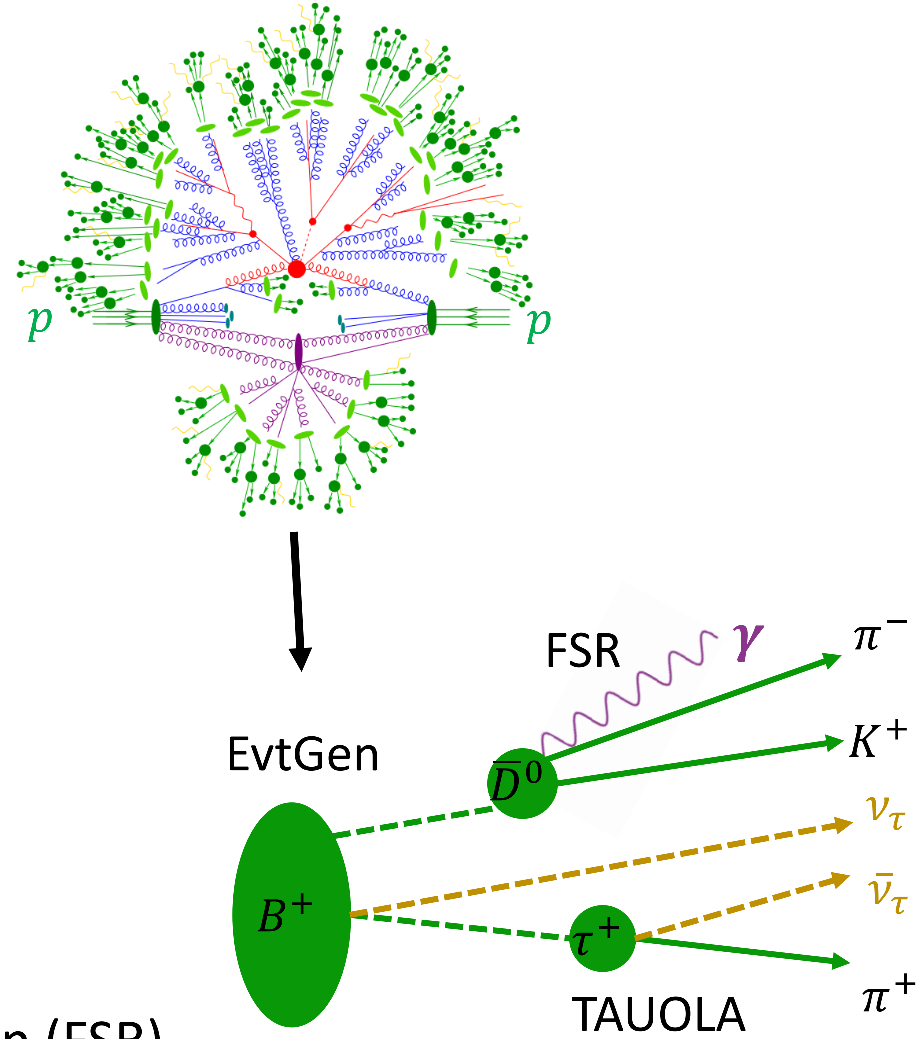
Simulation generator package specialised for decays of heavy particles containing b and c quarks.

- Contains 130 decay models for specific decays
- Maintains decay table with properties of $\sim 10^4$ explicit decays

External dependencies

- Uses Pythia8 for decays of generic quark configurations
- Uses TAUOLA for decays of τ particles
- Relies on PHOTOS or PHOTONS++ for final-state photon radiation (FSR)

Example collision simulated by PYTHIA8



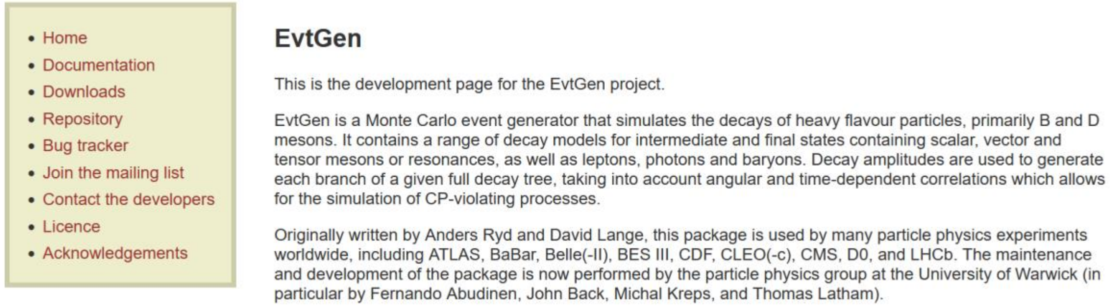
EvtGen status

- Developed in the 90’s, stable over past 10 years
- Changes mostly additions of new models by different collaborators
- Maintained at Warwick since 2012
- Started modernisation campaign in 2020 with help of research software engineers (RSEs) at Warwick

⇒ Main goal: enable thread safety

Heather Ratcliffe
Chris Brady

https://evtgen.hepforge.org



- Home
- Documentation
- Downloads
- Repository
- Bug tracker
- Join the mailing list
- Contact the developers
- Licence
- Acknowledgements

EvtGen

This is the development page for the EvtGen project.

EvtGen is a Monte Carlo event generator that simulates the decays of heavy flavour particles, primarily B and D mesons. It contains a range of decay models for intermediate and final states containing scalar, vector and tensor mesons or resonances, as well as leptons, photons and baryons. Decay amplitudes are used to generate each branch of a given full decay tree, taking into account angular and time-dependent correlations which allows for the simulation of CP-violating processes.

Originally written by Anders Ryd and David Lange, this package is used by many particle physics experiments worldwide, including ATLAS, BaBar, Belle(-II), BES III, CDF, CLEO(-c), CMS, D0, and LHCb. The maintenance and development of the package is now performed by the particle physics group at the University of Warwick (in particular by Fernando Abudinen, John Back, Michal Kreps, and Thomas Latham).

Mirror at https://gitlab.cern.ch/evtgen/evtgen
with continuous integration tests

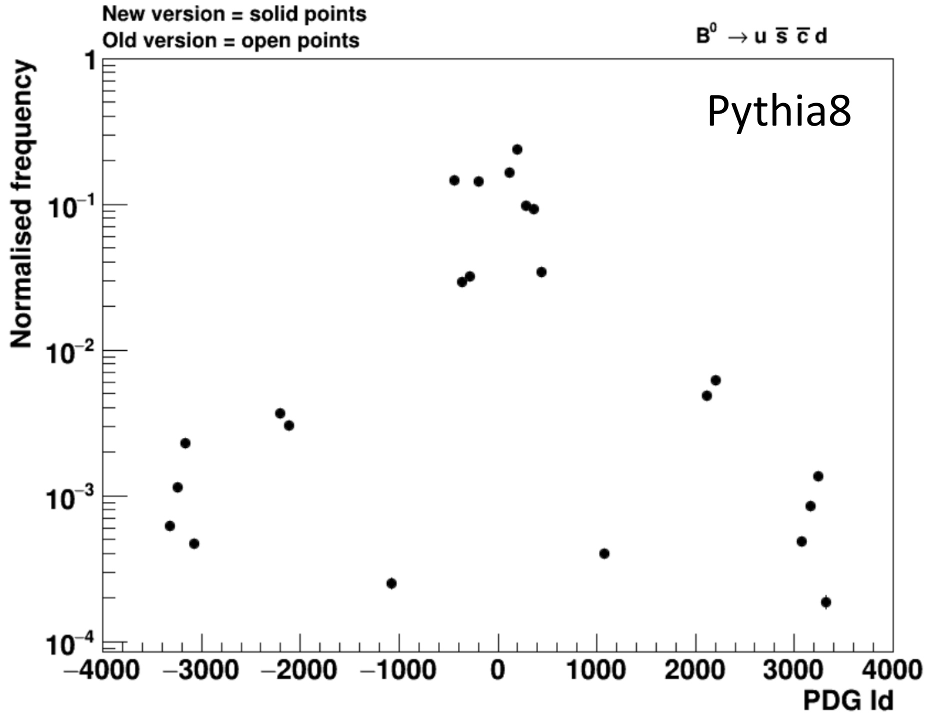
Validation

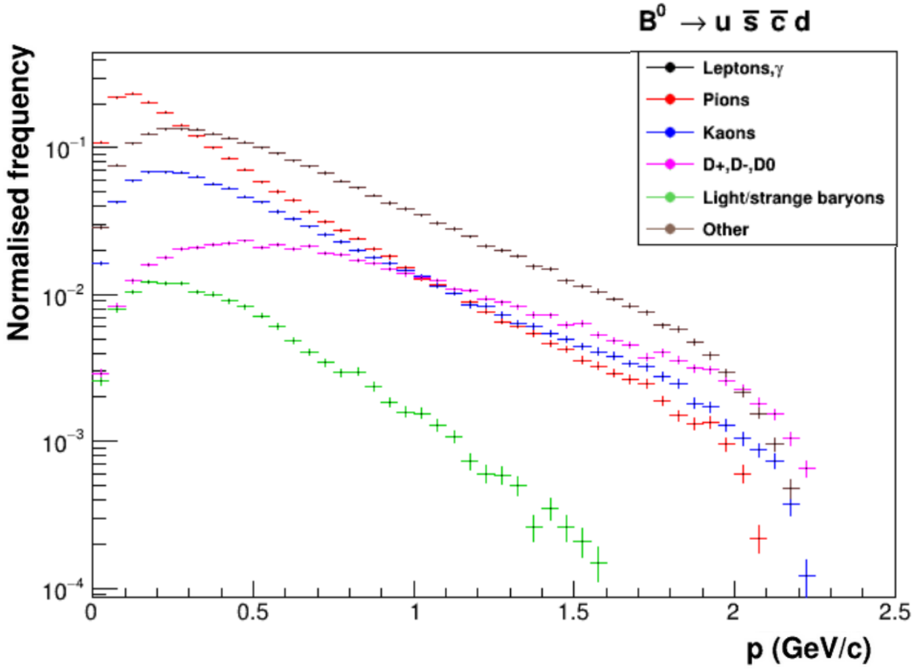
Simulation needs testing and validation after changes to ensure invariance of the physics models

- Implemented testing framework with common testing module and JSON configuration files
- Migrated all previous tests (covered only 40% of models) and added new ones to framework
- Tests cover all models and **external generators**
- Implemented **multi-threaded testing**

  ⇒ With possibility to use **std** or **tbb** thread models

Available also for users to facilitate testing/development





Challenges for multithreading in EvtGen

- Internal: structural limitations inside EvtGen
  - Global instance of random number generator
  - Global instance of particle properties and decay table

⇒ Needed structural changes identified and first combination of solutions found

- External: limitations from dependences
  - TAUOLA (Fortran)
  - PHOTOS (C)

⇒ Overcoming limitations from dependences are more challenging as they are external

Current solution

With help of research-software engineers:
Heather Ratcliffe, Chris Brady

Implemented modifications to enable thread-safety:

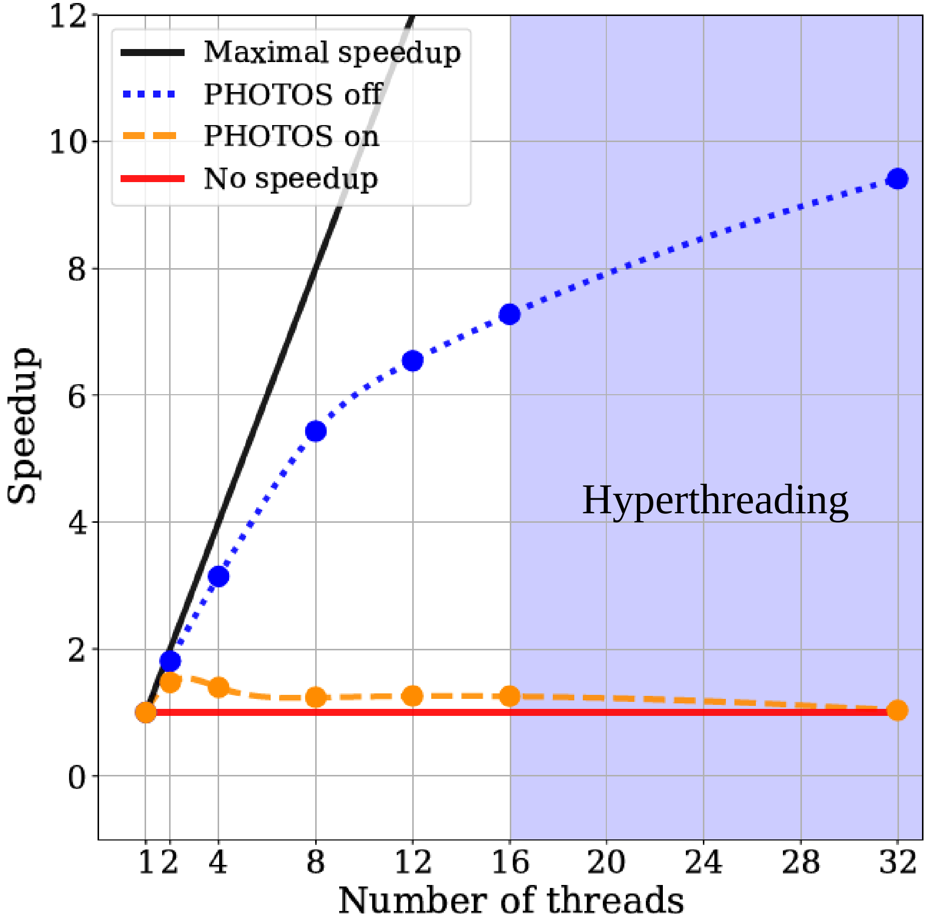
- Converted **static** objects to **static const** (or **static thread_local**)
- Global singleton objects made **thread_local**
- Serialized (**mutex**) calls to PHOTOS, PHOTONS++ and TAUOLA
- Introduced function to set random seed per event

⇒ Current preliminary status reaches thread safety

⇒ Passes all tests for all decay models, and external generators

⇒ Performance remains largely limited by external dependencies

⇒ Study alternatives for photon radiation and τ simulation

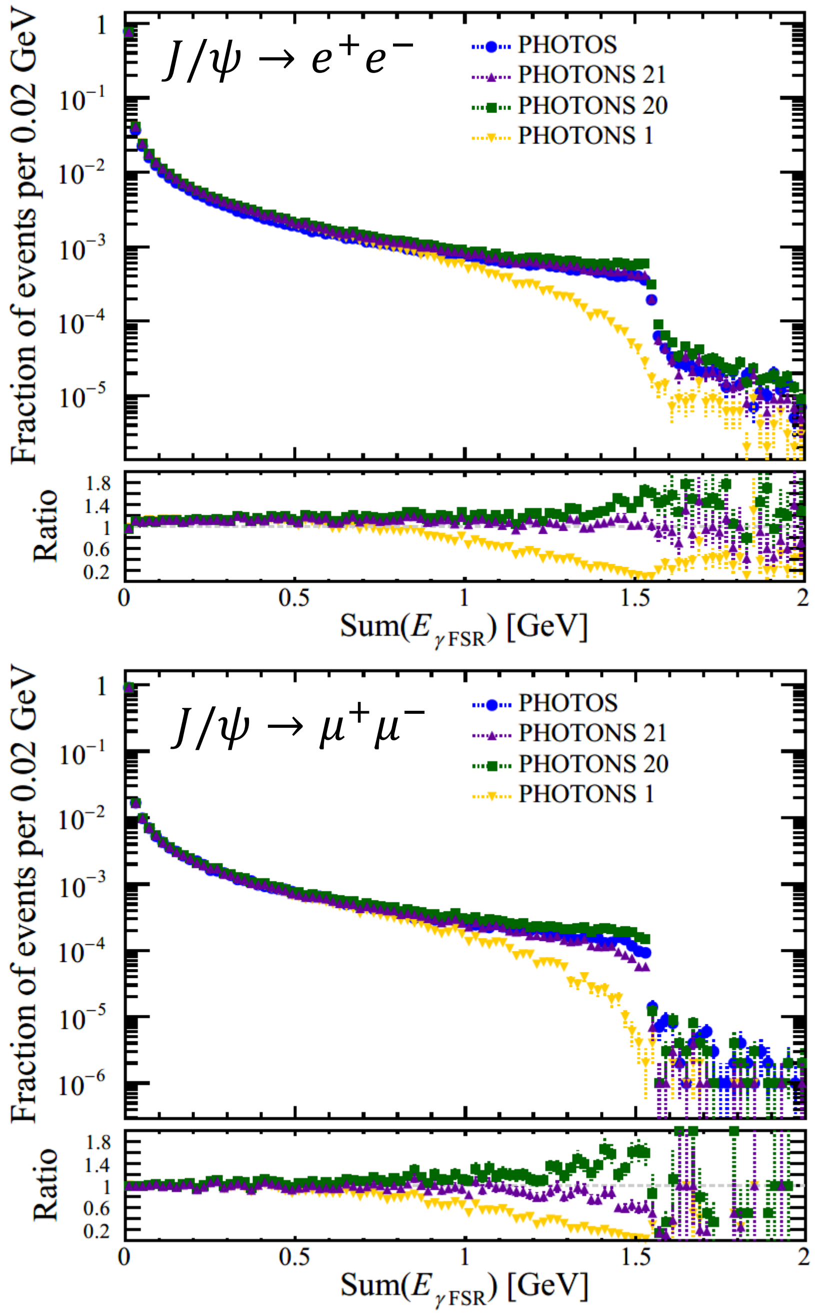


Sherpa’s PHOTONS++ for FSR

- PHOTONS++ in Sherpa can simulate emission of soft photons based on YFS approximation (mode 1)
- If switched on also hard photons based on collinear approximation (mode 2)
  - Approx. matrix-element corrections (mode 20) or
  - Exact matrix-element corrections (mode 21)
- With mode 1: fewer hard photons compared to PHOTOS (PHOTOS has matrix-element corrections implemented)
- With mode 2: generally good agreement with PHOTOS

$\Rightarrow$ Implemented switches for systematic studies

Amount of radiated energy



Interface between EvtGen and Sherpa

With help from Marek Schönherr and Frank Krauss

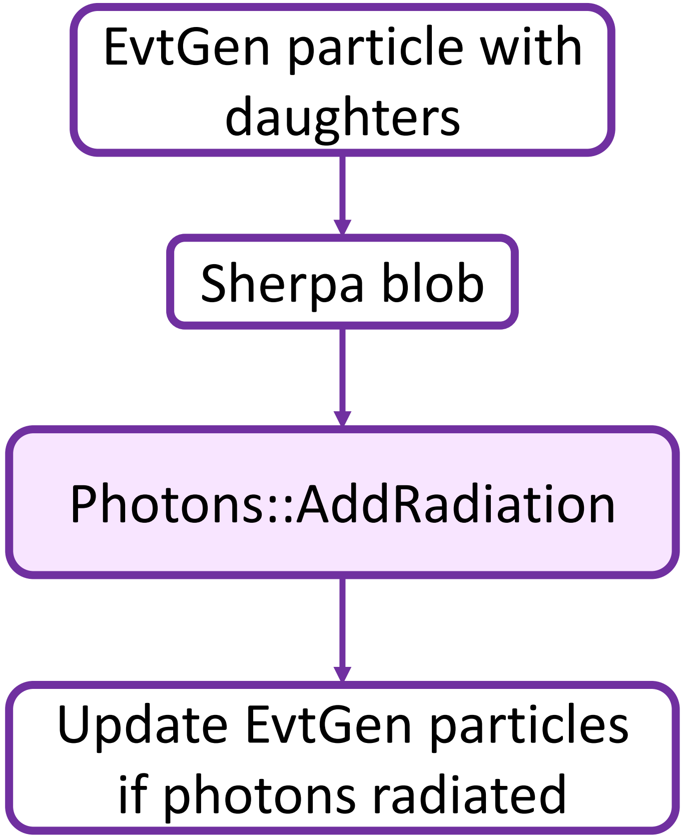
- EvtGen objects translated directly into Sherpa objects (and back)
- EvtGen RNG loaded as external library inside Sherpa
- Updates Sherpa’s **KF_Table** to include custom EvtGen’s particles
- Sherpa’s PHOTONS++ not thread-safe yet ⇒ **mutex**
- Supporting Sherpa 2.x.x releases

⇒ Available in EvtGen v3.0.0-beta

- Support for Sherpa 3.x.x in progress

⇒ Needs to iron out issue with lifetime of settings object

Workflow



EvtGen particle with daughters

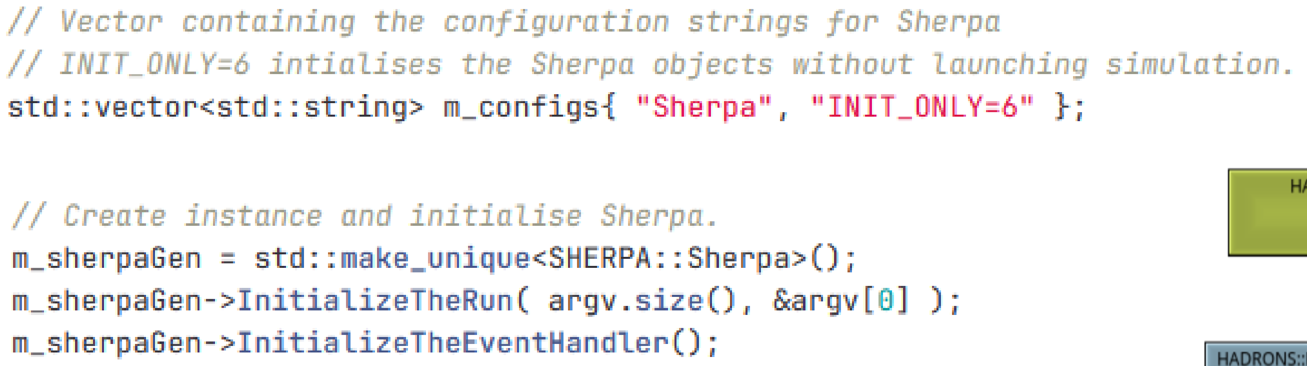
Sherpa blob

Photons::AddRadiation

Update EvtGen particles if photons radiated

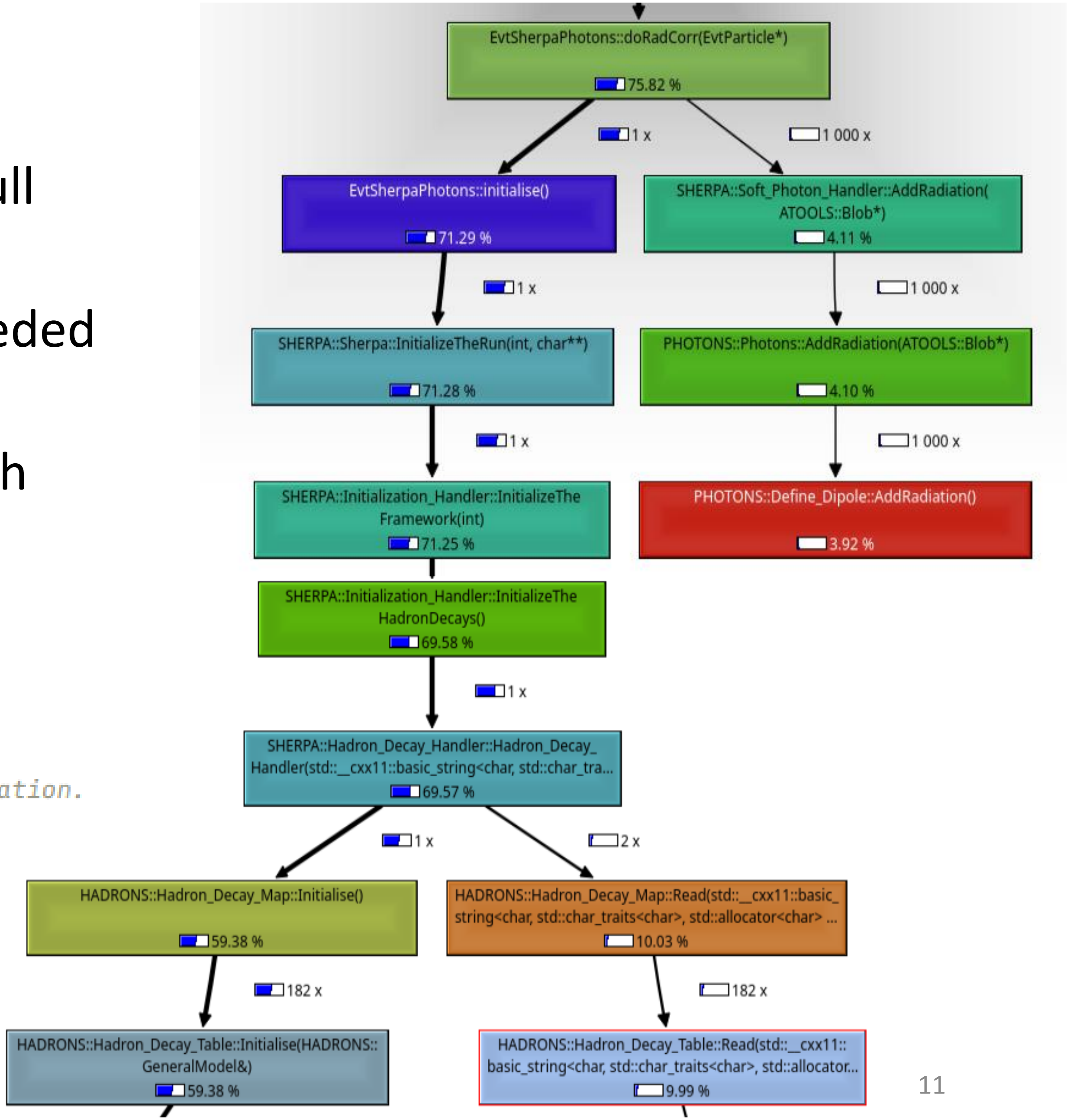
Initialisation of Sherpa

- For external dependencies EvtGen creates full Pythia/Sherpa objects
- Would be useful to initialise only objects needed by PHOTONS++
- Example: Sherpa’s initialisation takes as much time as $\sim 10^4$ decay events
- Several initialised objects are not used



```
// Vector containing the configuration strings for Sherpa
// INIT_ONLY=6 intialises the Sherpa objects without launching simulation.
std::vector<std::string> m_configs{ "Sherpa", "INIT_ONLY=6" };

// Create instance and initialise Sherpa.
m_sherpaGen = std::make_unique<SHERPA::Sherpa>();
m_sherpaGen->InitializeTheRun( argv.size(), &argv[0] );
m_sherpaGen->InitializeTheEventHandler();
```



EvtSherpaPhotons::doRadCorr(EvtParticle*)
75.82 %
1 x
1 000 x
EvtSherpaPhotons::initialise()
71.29 %
SHERPA::Soft_Photon_Handler::AddRadiation(ATOOLS::Blob*)
4.11 %
1 x
1 000 x
SHERPA::Sherpa::InitializeTheRun(int, char**)
71.28 %
PHOTONS::Photons::AddRadiation(ATOOLS::Blob*)
4.10 %
1 x
1 000 x
SHERPA::Initialization_Handler::InitializeThe Framework(int)
71.25 %
PHOTONS::Define_Dipole::AddRadiation()
3.92 %
SHERPA::Initialization_Handler::InitializeThe HadronDecays()
69.58 %
1 x
SHERPA::Hadron_Decay_Handler::Hadron_Decay_ Handler(std::__cxx11::basic_string<char, std::char_tra...
69.57 %
1 x
2 x
HADRONS::Hadron_Decay_Map::Initialise()
59.38 %
HADRONS::Hadron_Decay_Map::Read(std::__cxx11::basic_ string<char, std::char_traits<char>, std::allocator<char> ...
10.03 %
182 x
182 x
HADRONS::Hadron_Decay_Table::Initialise(HADRONS:: GeneralModel&)
59.38 %
HADRONS::Hadron_Decay_Table::Read(std::__cxx11:: basic_string<char, std::char_traits<char>, std::allocator...
9.99 %

Vincia QED shower for FSR

- Vincia parton shower evolution based on Antenna approximation (can be interleaved)
- Recently adapted to radiate off hadrons (previously supporting only leptons)
- Matrix-element corrections not implemented yet

⇒ A lot of room for improvement and validation

⇒ However, preliminary results look promising

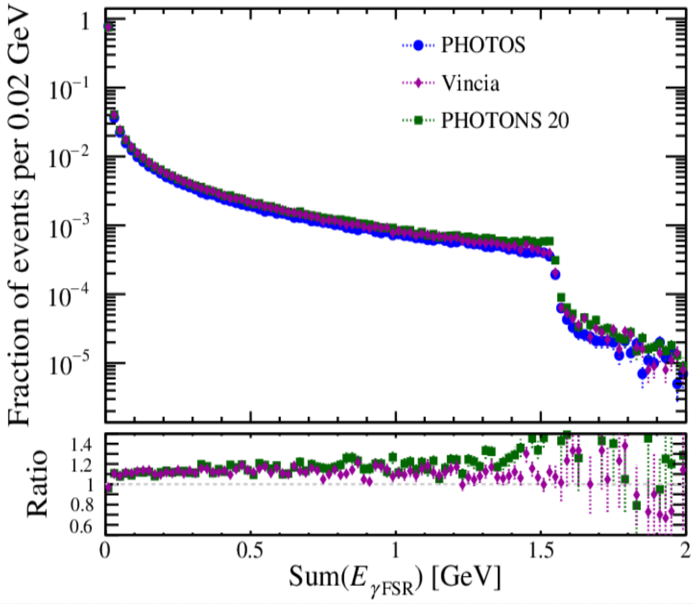
Technical aspects

- Vincia is embedded in Pythia8
- Algorithm implementation enables thread safety
- Developed EvtGen ↔ Vincia interface based on existing dependency with Pythia8 → Within Monash-Warwick alliance

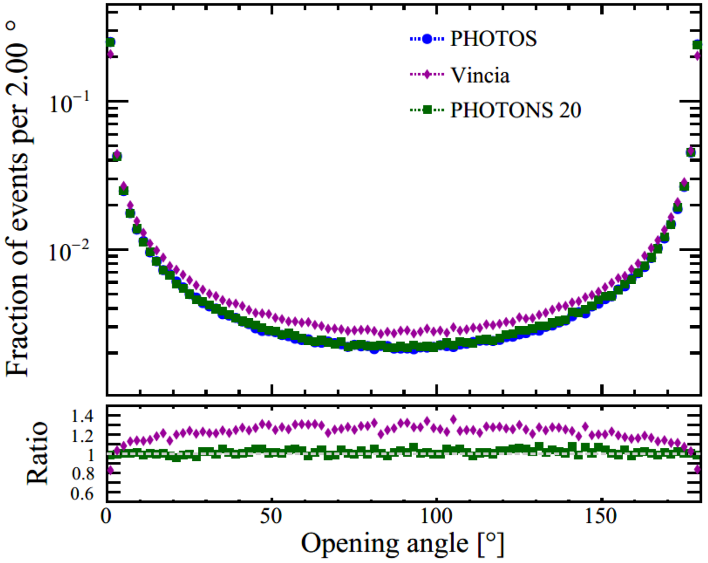
⇒ Not in EvtGen release as Vincia under development

$J/\psi \to e^+e^-$

Amount of radiated energy



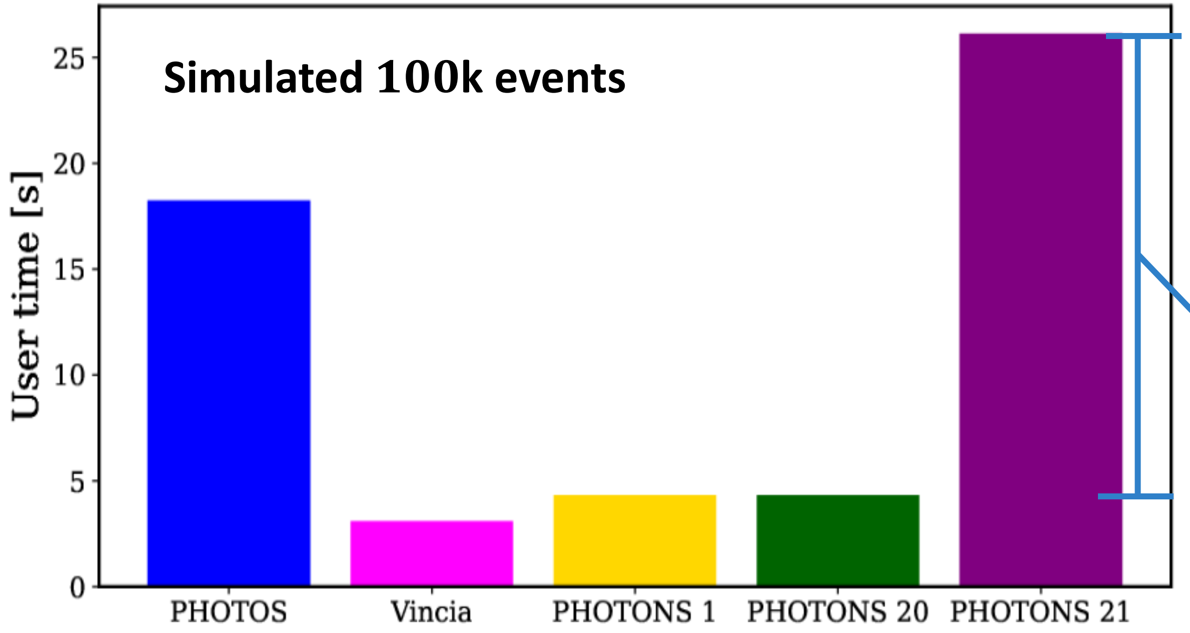
Angular distribution of photons

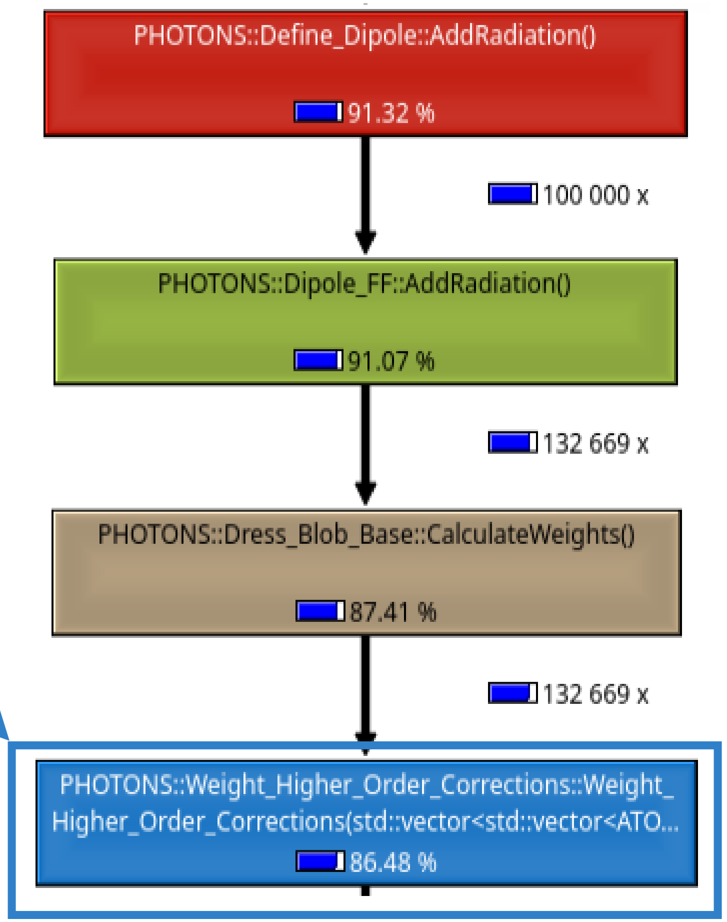


A word on timing

- Compare simulation time using $J/\psi \to e^+e^-$ decay as benchmark

⇒ Collinear singularities enhanced due to small electron mass




PHOTONS::Define_Dipole::AddRadiation()
91.32 %
100 000 x
PHOTONS::Dipole_FF::AddRadiation()
91.07 %
132 669 x
PHOTONS::Dress_Blob_Base::CalculateWeights()
87.41 %
132 669 x
PHOTONS::Weight_Higher_Order_Corrections::Weight_Higher_Order_Corrections(std::vector<std::vector<ATO...
86.48 %

⇒ Largest consumption by exact matrix-element calculation

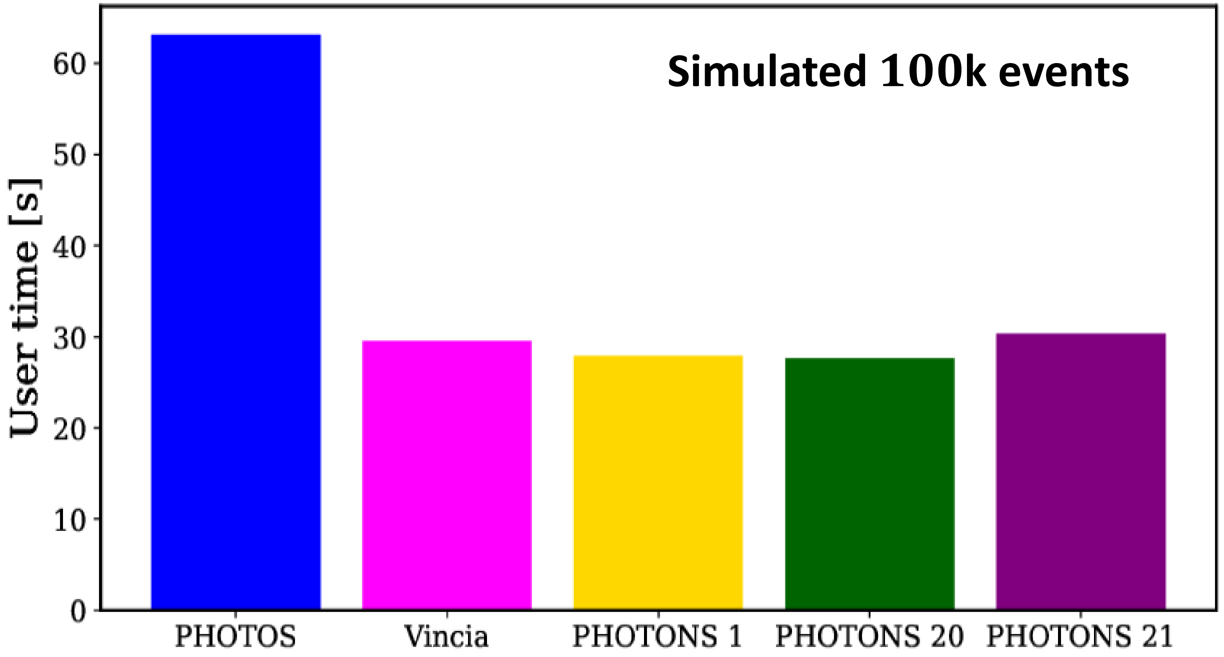
⇒ Good precision/time trade-off for option 20 (will use as default)

⇒ Potential speedup using Vincia or PHOTONS by about factor 4

Another word on timing

- Compare simulation time when simulating generic $\Upsilon(4S) \to B\bar{B}$

⇒ Benchmark for general use



⇒ No large difference between PHOTONS options in generic case

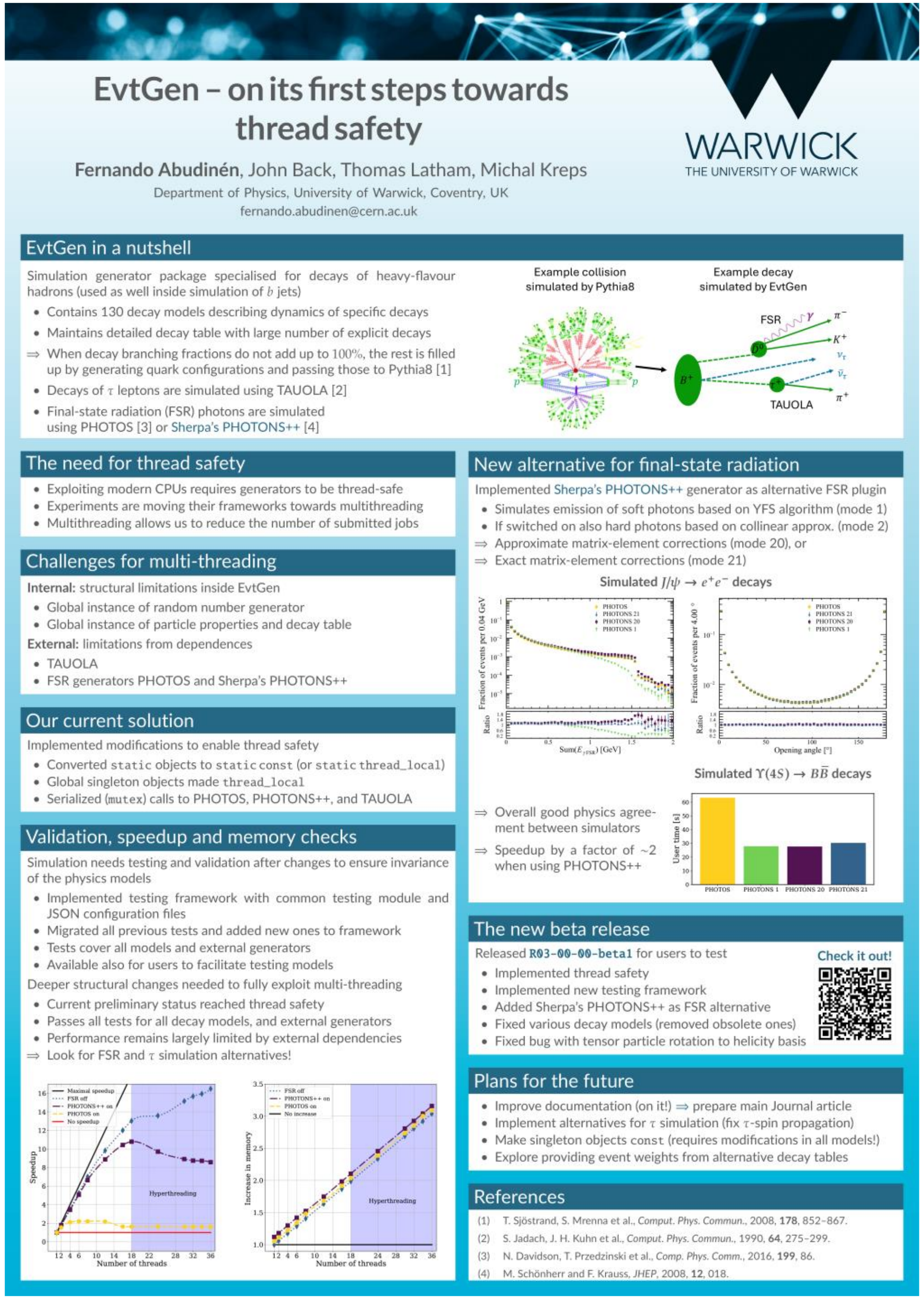
⇒ Potential speedup using Vincia or PHOTONS by about factor 2

New release R03-00-00-beta1

After the long campaign released R03-00-00-beta1 for users to test

- Implemented thread safety
- Implemented new testing framework
- Added Sherpa's PHOTONS++ as FSR alternative
- Fixed various decay models (removed obsolete ones)
- Fixed bug with tensor particle rotation to helicity basis

October 19 - 25, 2024
CHEP
2024



EvtGen – on its first steps towards thread safety

Fernando Abudinén, John Back, Thomas Latham, Michal Kreps
Department of Physics, University of Warwick, Coventry, UK
fernando.abudinen@cern.ac.uk

WARWICK
THE UNIVERSITY OF WARWICK

EvtGen in a nutshell

Simulation generator package specialised for decays of heavy-flavour hadrons (used as well inside simulation of b jets)

- Contains 130 decay models describing dynamics of specific decays
- Maintains detailed decay table with large number of explicit decays

⇒ When decay branching fractions do not add up to 100%, the rest is filled up by generating quark configurations and passing those to Pythia8 [1]

- Decays of τ leptons are simulated using TAUOLA [2]
- Final-state radiation (FSR) photons are simulated using PHOTOS [3] or Sherpa's PHOTONS++ [4]

Example collision simulated by Pythia8

Example decay simulated by EvtGen

The need for thread safety

- Exploiting modern CPUs requires generators to be thread-safe
- Experiments are moving their frameworks towards multithreading
- Multithreading allows us to reduce the number of submitted jobs

Challenges for multi-threading

Internal: structural limitations inside EvtGen

- Global instance of random number generator
- Global instance of particle properties and decay table

External: limitations from dependences

- TAUOLA
- FSR generators PHOTOS and Sherpa's PHOTONS++

Our current solution

Implemented modifications to enable thread safety

- Converted `static` objects to `static const` (or `static thread_local`)
- Global singleton objects made `thread_local`
- Serialized (`mutex`) calls to PHOTOS, PHOTONS++, and TAUOLA

Validation, speedup and memory checks

Simulation needs testing and validation after changes to ensure invariance of the physics models

- Implemented testing framework with common testing module and JSON configuration files
- Migrated all previous tests and added new ones to framework
- Tests cover all models and external generators
- Available also for users to facilitate testing models

Deeper structural changes needed to fully exploit multi-threading

- Current preliminary status reached thread safety
- Passes all tests for all decay models, and external generators
- Performance remains largely limited by external dependencies

⇒ Look for FSR and τ simulation alternatives!

New alternative for final-state radiation

Implemented Sherpa's PHOTONS++ generator as alternative FSR plugin

- Simulates emission of soft photons based on YFS algorithm (mode 1)
- If switched on also hard photons based on collinear approx. (mode 2)

⇒ Approximate matrix-element corrections (mode 20), or

⇒ Exact matrix-element corrections (mode 21)

Simulated $J/\psi \to e^+e^-$ decays

Simulated $\Upsilon(4S) \to B\bar{B}$ decays

⇒ Overall good physics agreement between simulators

⇒ Speedup by a factor of ~2 when using PHOTONS++

The new beta release

Released **`R03-00-00-beta1`** for users to test

- Implemented thread safety
- Implemented new testing framework
- Added Sherpa's PHOTONS++ as FSR alternative
- Fixed various decay models (removed obsolete ones)
- Fixed bug with tensor particle rotation to helicity basis

Check it out!

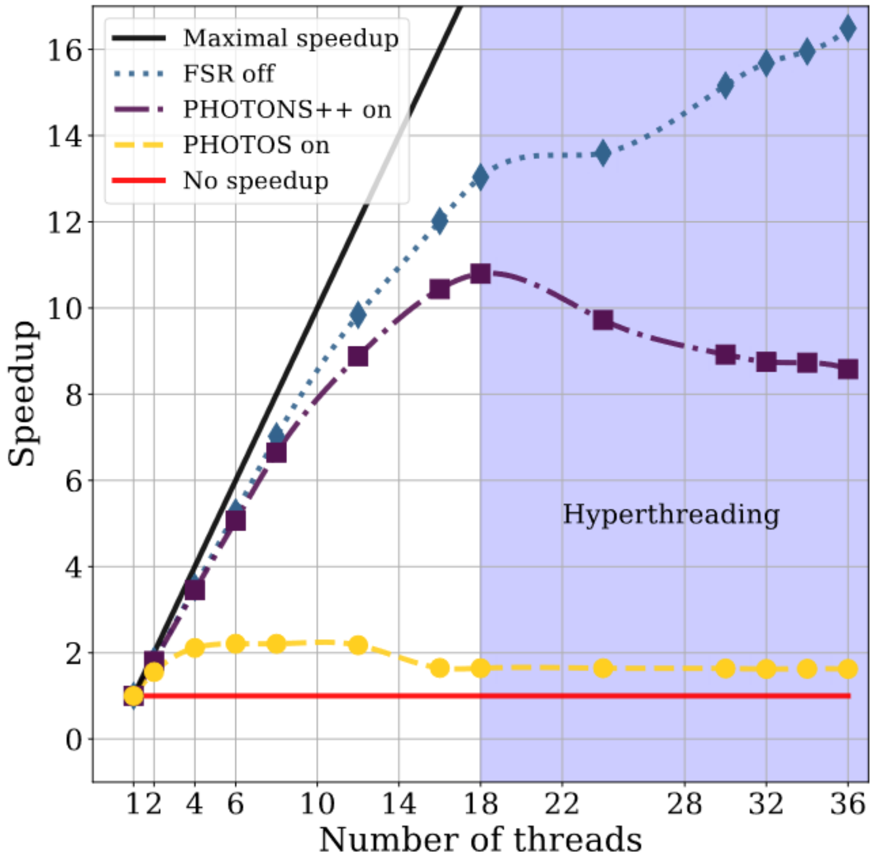
Plans for the future

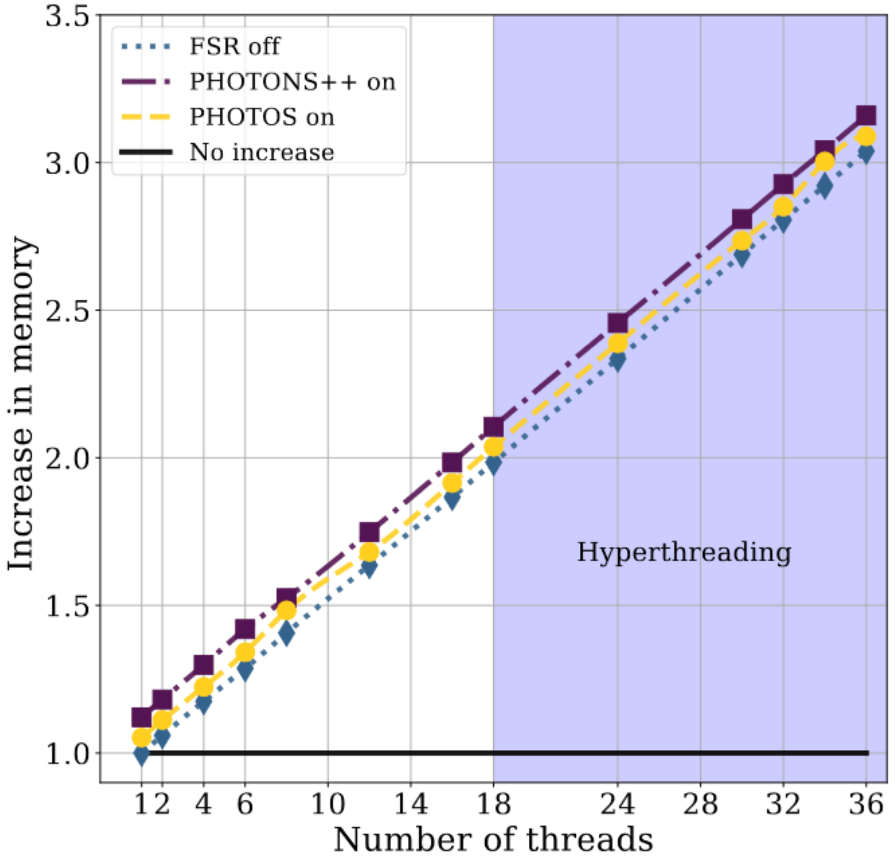
- Improve documentation (on it!) ⇒ prepare main Journal article
- Implement alternatives for τ simulation (fix τ-spin propagation)
- Make singleton objects `const` (requires modifications in all models!)
- Explore providing event weights from alternative decay tables

References

(1) T. Sjöstrand, S. Mrenna et al., *Comput. Phys. Commun.*, 2008, **178**, 852–867.
(2) S. Jadach, J. H. Kuhn et al., *Comput. Phys. Commun.*, 1990, **64**, 275–299.
(3) N. Davidson, T. Przedzinski et al., *Comp. Phys. Comm.*, 2016, **199**, 86.
(4) M. Schönherr and F. Krauss, *JHEP*, 2008, **12**, 018.

Checks with R03-00-00-beta1





⇒ Better performance with new FSR alternative

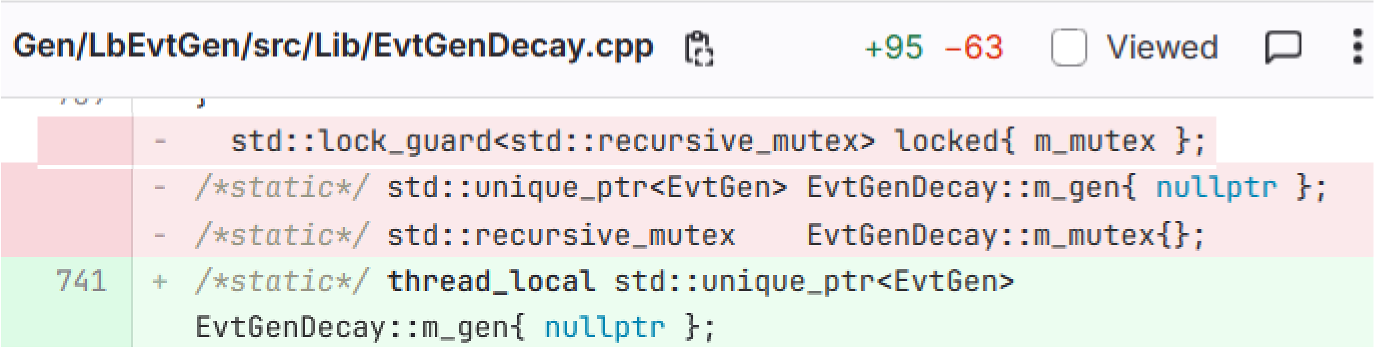
⇒ Deeper structural changes needed to fully exploit multithreading with increased memory sharing

Implementation into LHCb simulation

- Remove serialisation (**mutex**) for EvtGen
- One EvtGen instance per tread required for multi-threading

⇒ One initialisation per thread

- Set seed for each event
- Implement **const** correctness inside LHCb-only models



Gen/LbEvtGen/src/Lib/EvtGenDecay.cpp +95 −63

```
-   std::lock_guard<std::recursive_mutex> locked{ m_mutex };
- /*static*/ std::unique_ptr<EvtGen> EvtGenDecay::m_gen{ nullptr };
- /*static*/ std::recursive_mutex     EvtGenDecay::m_mutex{};
+ /*static*/ thread_local std::unique_ptr<EvtGen>
  EvtGenDecay::m_gen{ nullptr };
```

Plans for the future

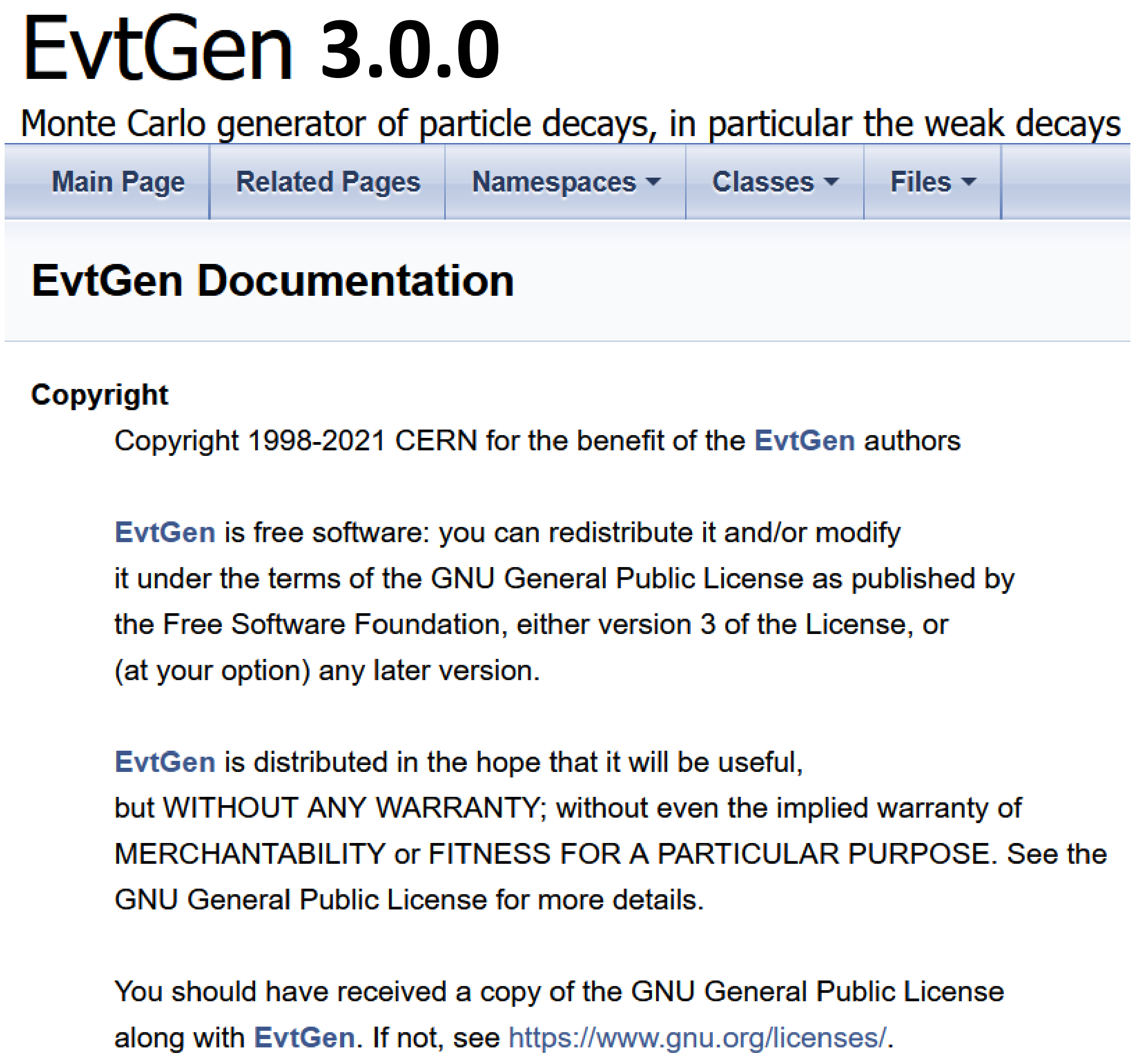
https://evtgen.hepforge.org/doc/doxygen/test

Immediate

- Improve Doxygen documentation (on it!)
- Prepare main Journal article

Long-term

- Make singleton objects **const** (requires modifications in all models!)
- Implement Vincia as FSR alternative
- Implement alternatives for τ particle simulation (fix spin propagation)
- Explore providing event weights from alternative decay tables



EvtGen 3.0.0

Monte Carlo generator of particle decays, in particular the weak decays

EvtGen Documentation

Copyright

Copyright 1998-2021 CERN for the benefit of the EvtGen authors

EvtGen is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

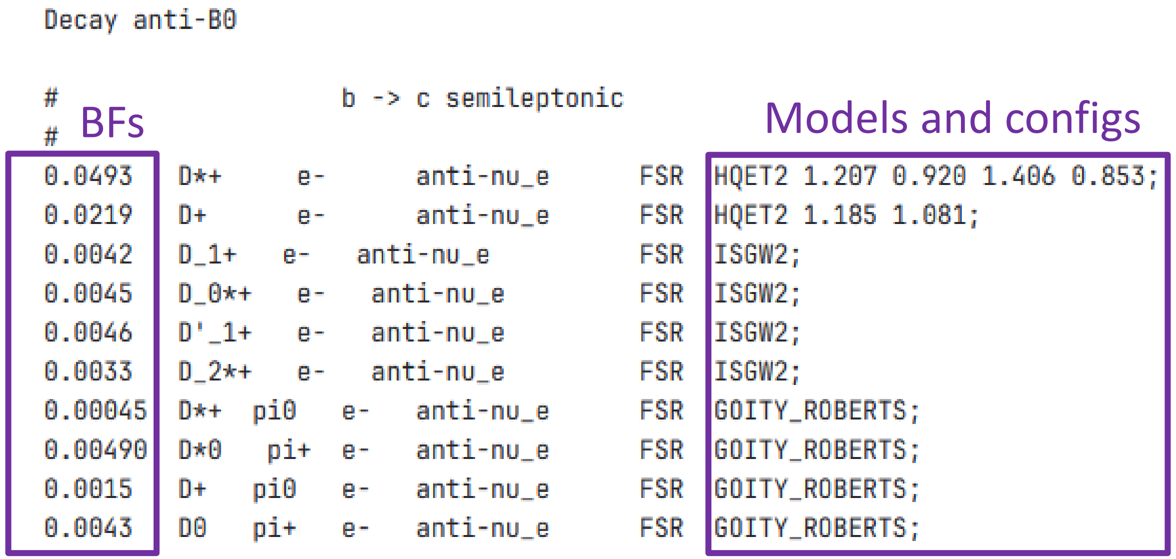
EvtGen is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with EvtGen. If not, see https://www.gnu.org/licenses/.

Uncertainties associated with Decay Table

- EvtGen Decay Table does not consider BF uncertainties
- Each user can have a custom Decay Table
- Idea: evaluate uncertainties by varying the Decay Table and providing event weights for systematic variations

Example from general Decay table



```
Decay anti-B0

#                   b -> c semileptonic
#
0.0493    D*+     e-      anti-nu_e       FSR  HQET2 1.207 0.920 1.406 0.853;
0.0219    D+      e-      anti-nu_e       FSR  HQET2 1.185 1.081;
0.0042    D_1+   e-   anti-nu_e           FSR  ISGW2;
0.0045    D_0*+   e-   anti-nu_e          FSR  ISGW2;
0.0046    D'_1+   e-   anti-nu_e          FSR  ISGW2;
0.0033    D_2*+   e-   anti-nu_e          FSR  ISGW2;
0.00045   D*+  pi0   e-   anti-nu_e       FSR  GOITY_ROBERTS;
0.00490   D*0   pi+  e-   anti-nu_e       FSR  GOITY_ROBERTS;
0.0015    D+   pi0   e-   anti-nu_e       FSR  GOITY_ROBERTS;
0.0043    D0   pi+   e-   anti-nu_e       FSR  GOITY_ROBERTS;
```

Variation of BFs

- Requires bookkeeping ⇒ relatively straightforward

Variation of models (and model config)

- Calculate alternative amplitude for generated kinematic configuration
- Propagate alternative spin-density matrix through decay chain

⇒ Requires to modify the structure in all models plus expansion of bookkeeping

⇒ Can be implemented, but very challenging

Summary and outlook

- After a long campaign released R03-00-00-beta1

⇒ Enabled thread-safety (preliminary solutions)

⇒ New testing framework with multi-threading capability

⇒ Sherpa PHOTONS++ for FSR (2x faster than PHOTOS)

⇒ Various bugfixes (models, tensor-particle decays)

⇒ Currently working on documentation

⇒ Tag full release 3 afterwards

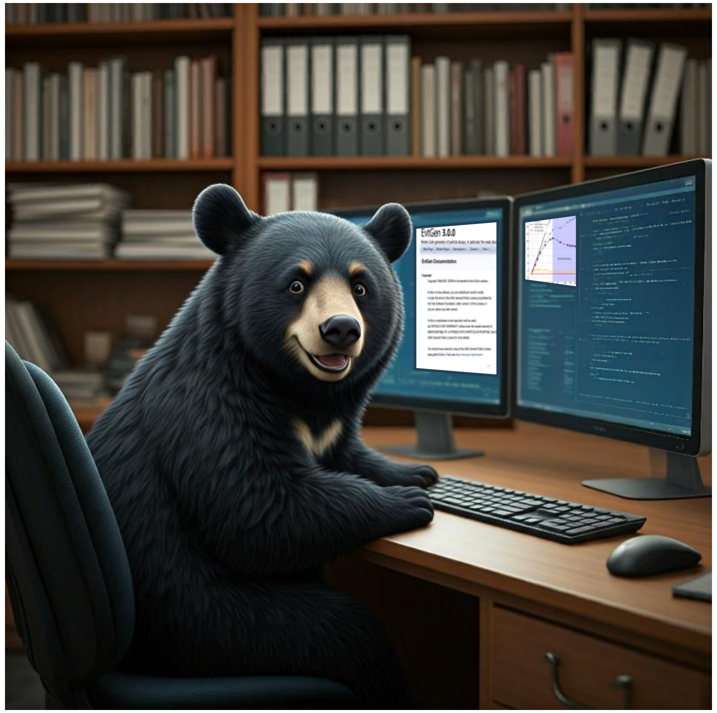
- Long-term plans

⇒ Make structural modifications to make singletons **const**

⇒ Implement VINCIA as FSR alternative

⇒ Continue work on alternatives for TAUOLA

⇒ Incorporate systematic variations of Decay Table



Generated by Gemini

Thanks for your attention!

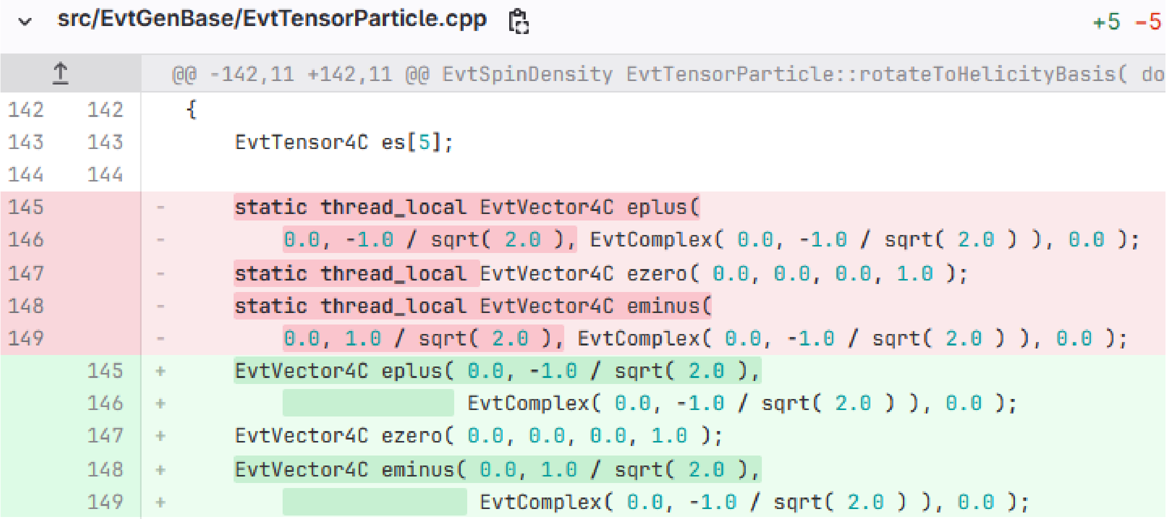
Backup

Bugfix for tensor decays

- Belle II uncovered bug in angular distributions for tensor particle decays
- Polarisation vectors were **static** and not properly reset after Euler rotations

⇒ Euler rotations were compounded

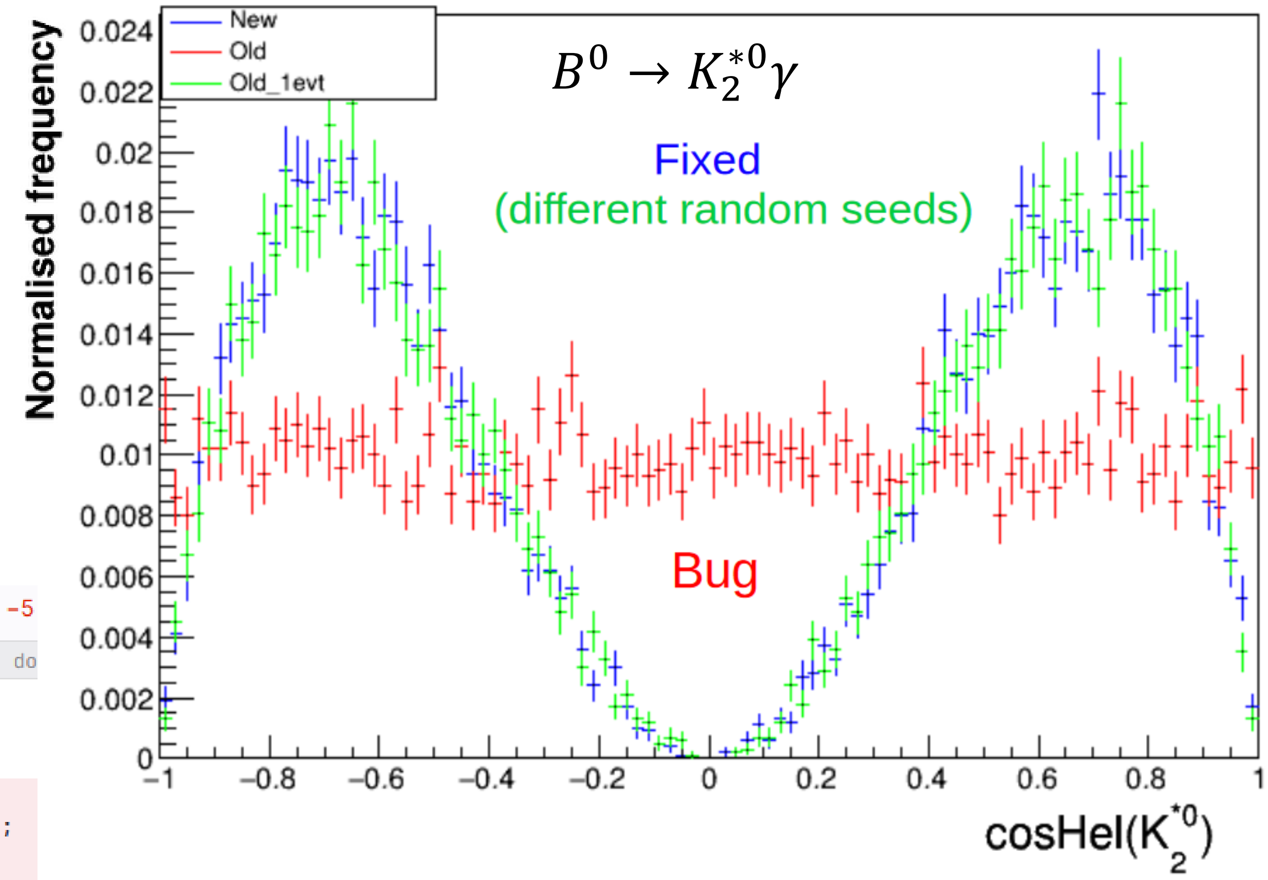
⇒ Fixed by removing the static modifier



```
src/EvtGenBase/EvtTensorParticle.cpp                                    +5 -5
@@ -142,11 +142,11 @@ EvtSpinDensity EvtTensorParticle::rotateToHelicityBasis( do
     {
         EvtTensor4C es[5];

-        static thread_local EvtVector4C eplus(
-            0.0, -1.0 / sqrt( 2.0 ), EvtComplex( 0.0, -1.0 / sqrt( 2.0 ) ), 0.0 );
-        static thread_local EvtVector4C ezero( 0.0, 0.0, 0.0, 1.0 );
-        static thread_local EvtVector4C eminus(
-            0.0, 1.0 / sqrt( 2.0 ), EvtComplex( 0.0, -1.0 / sqrt( 2.0 ) ), 0.0 );
+        EvtVector4C eplus( 0.0, -1.0 / sqrt( 2.0 ),
+                              EvtComplex( 0.0, -1.0 / sqrt( 2.0 ) ), 0.0 );
+        EvtVector4C ezero( 0.0, 0.0, 0.0, 1.0 );
+        EvtVector4C eminus( 0.0, 1.0 / sqrt( 2.0 ),
+                               EvtComplex( 0.0, -1.0 / sqrt( 2.0 ) ), 0.0 );
```

Helicity angle distribution



New
Old
Old_1evt
$B^0 \to K_2^{*0}\gamma$
Fixed
(different random seeds)
Bug
Normalised frequency
cosHel(K_2^{*0})