Karlsruhe Institute of Technology

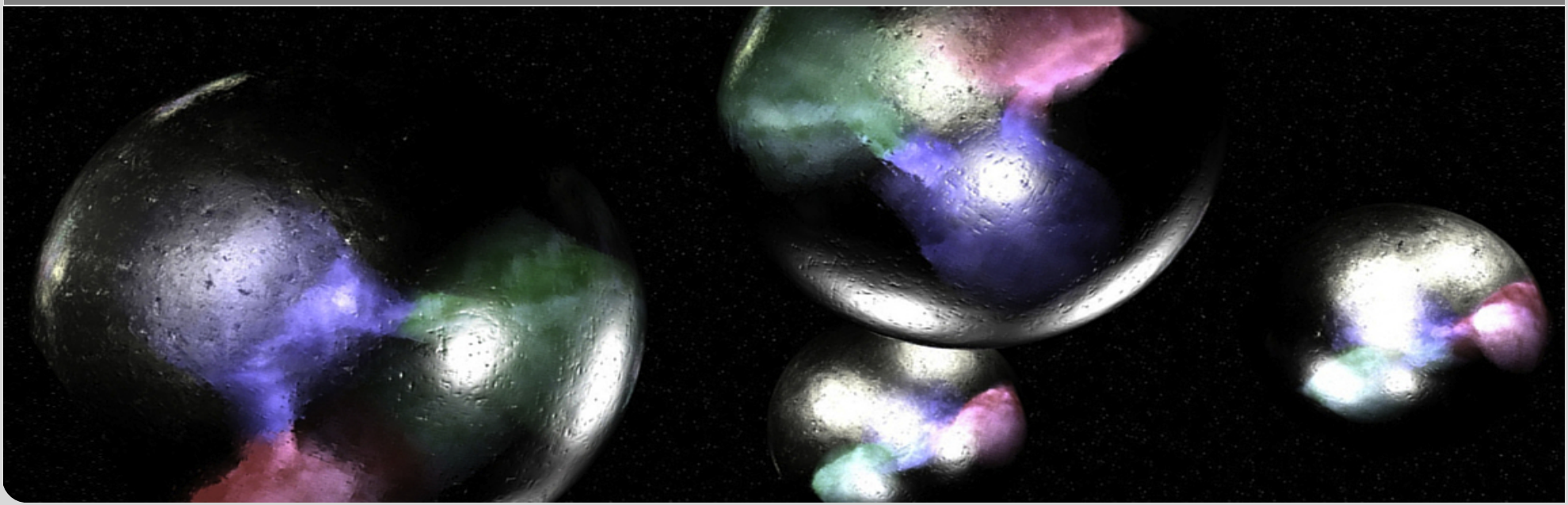
Belle II

Handling the Freak Events

and some Refactoring of the VXDTF2

Felix Metzner | 2nd March 2018

INSTITUT FÜR EXPERIMENTELLE TEILCHENPHYSIK (ETP)



Overview

Aim of current effort:

- Introduce further cuts to reduce not only memory consumption, but also runtime.
- Refactoring of the SegmentNetworkProducer and related the code.

Boundary Conditions for the presented Results:

Working with the software state of my feature branch:

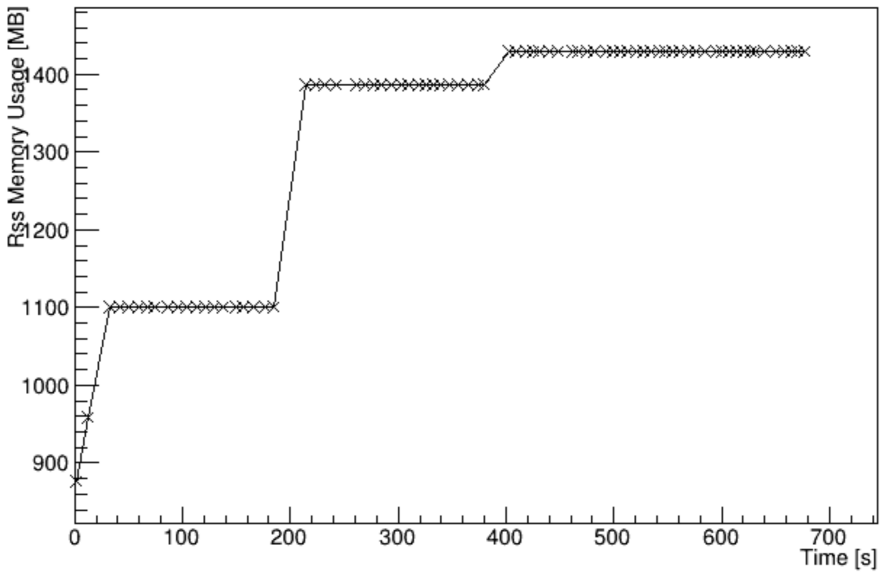
- Pull Request #1683: `Feature/find_path_rework_for_vxdtf2`
- Branched on February 11th from commit `b7cadcf267f`

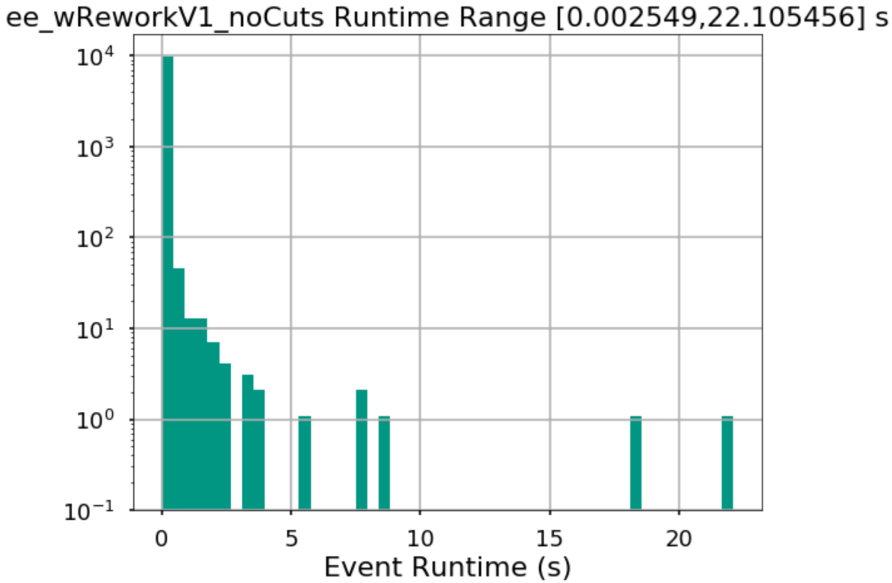
Study performed for **Bhabha and Υ(4S)** events for **Phase 2 and 3** with **10000 events** each.

The path included only the VXDTF2 and some validation modules.

Problematic Behaviour

When applying **no cuts** at all during the Segment Network creation and the Path Finding, a high **memory consumption and runtime** are observed for Bhabha events.

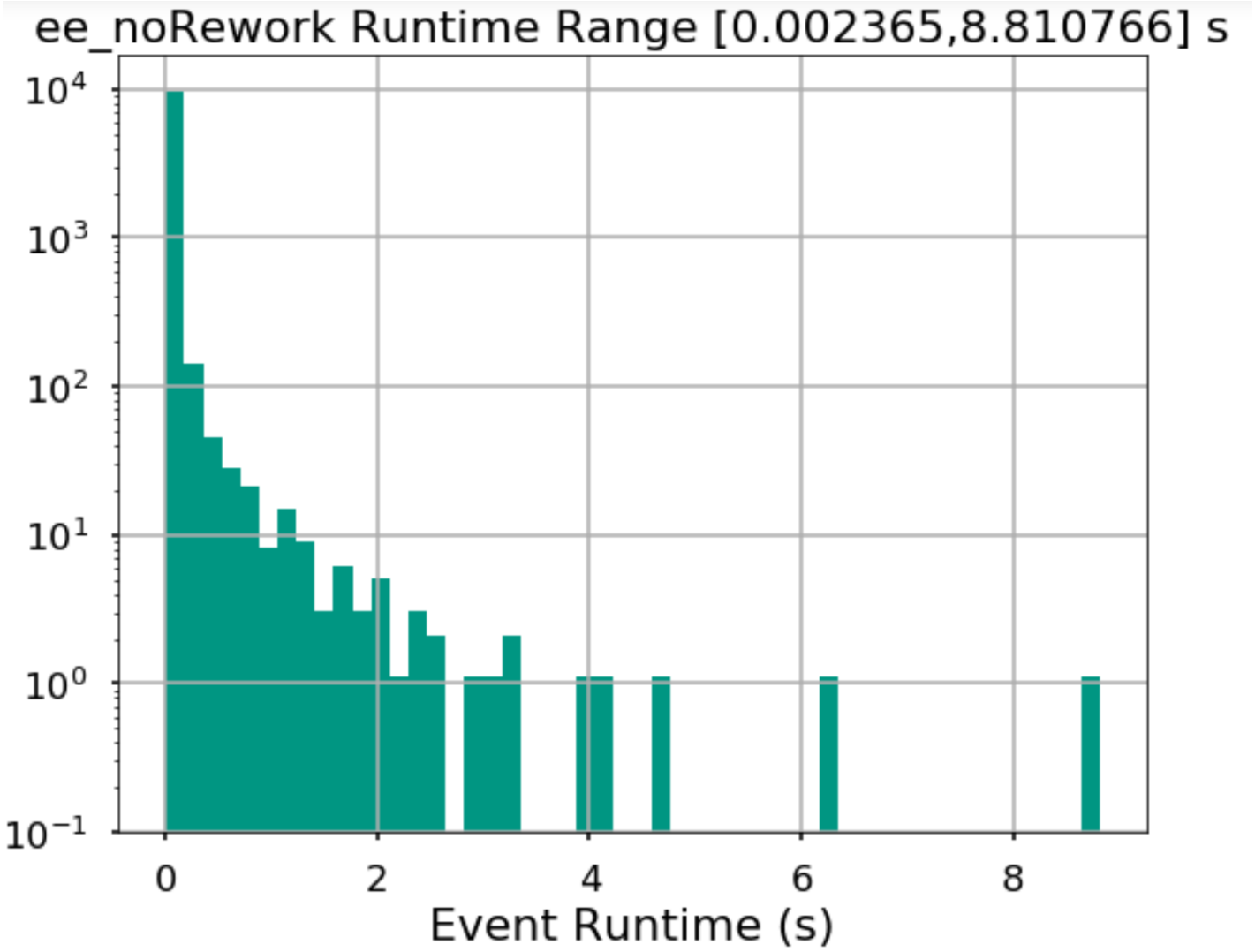




The memory consumption was tackled by introducing **a single cut** on the size of the Segment Network accepted for the Path Finding in the TrackFinderVXDCellOMat.

Problematic Behaviour

The processing time remained problematic for some Bhabha events, even after moving the cut into the SegmentNetworkProducer.



Exploding Combinatorics

In the search for the reason for the high processing time, several quantities have been studied:

Number of Active Sectors and the connections between them

= Sectors which contain SpacePoints (1st Step in the SNP)

Number of TrackNodes and the connections between them

= SpacePoints and connected pairs of them (2nd Step in the SNP)

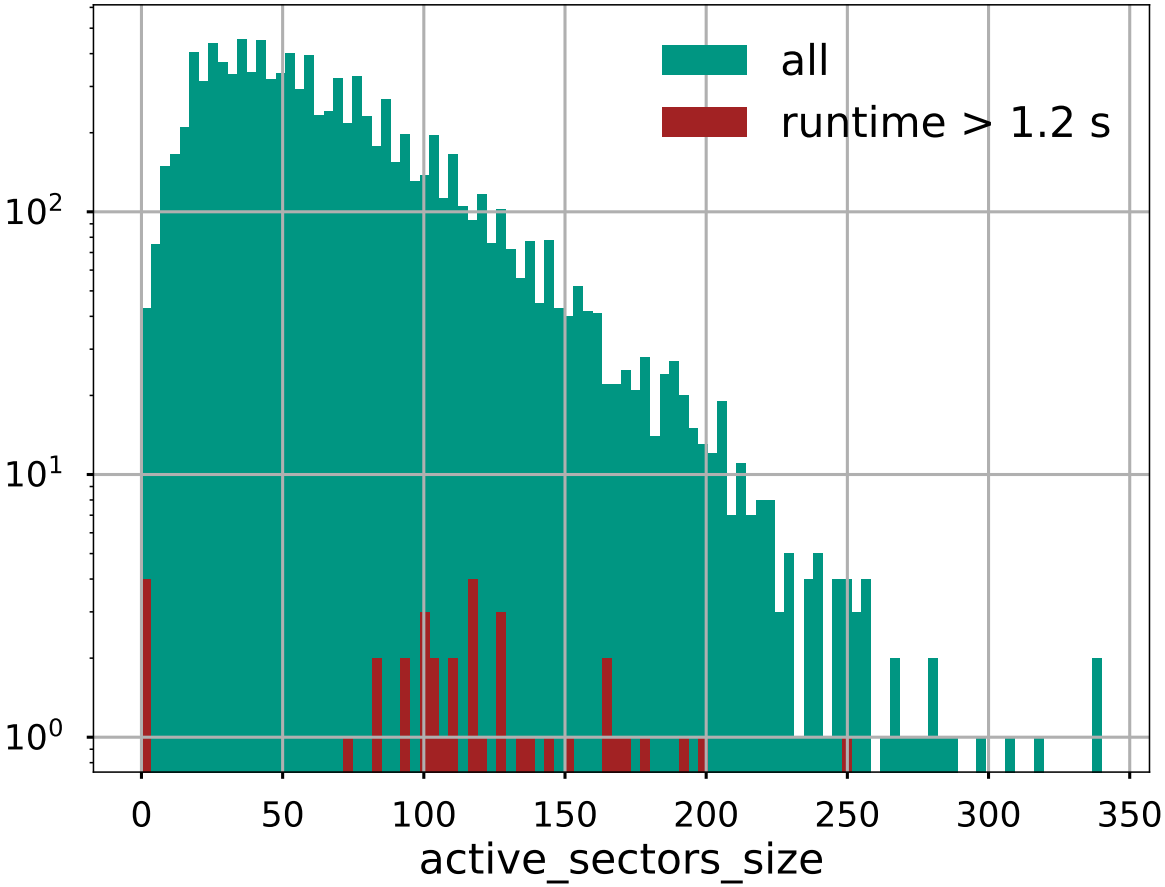
Number of Segments and the connections between them

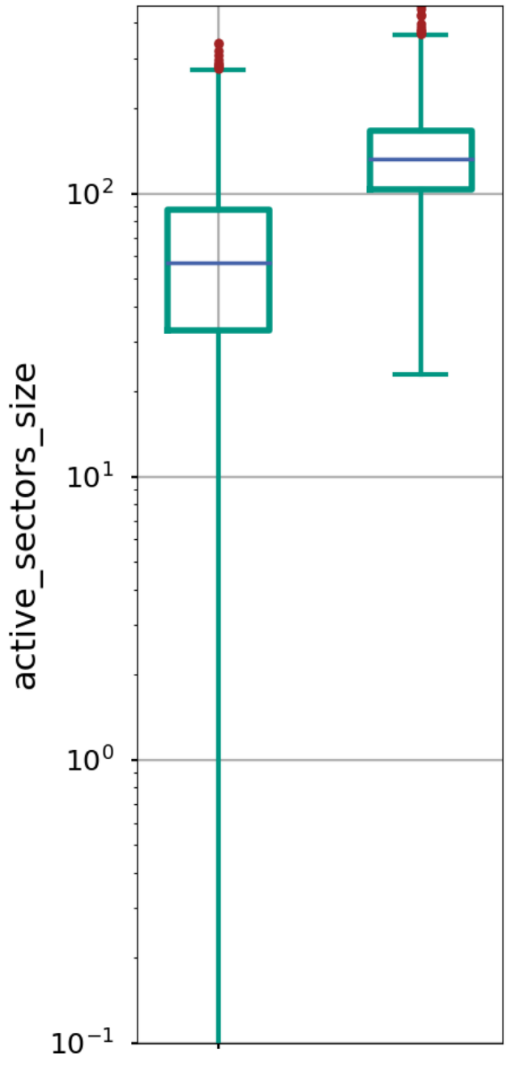
= 2-SP-Combinations and connected pairs of them (3rd Step in the SNP)

Number of Collected Paths

= full Track Candidates (in the TrackFinderVXDCellOMat)

Exploding Combinatorics – Active Sectors

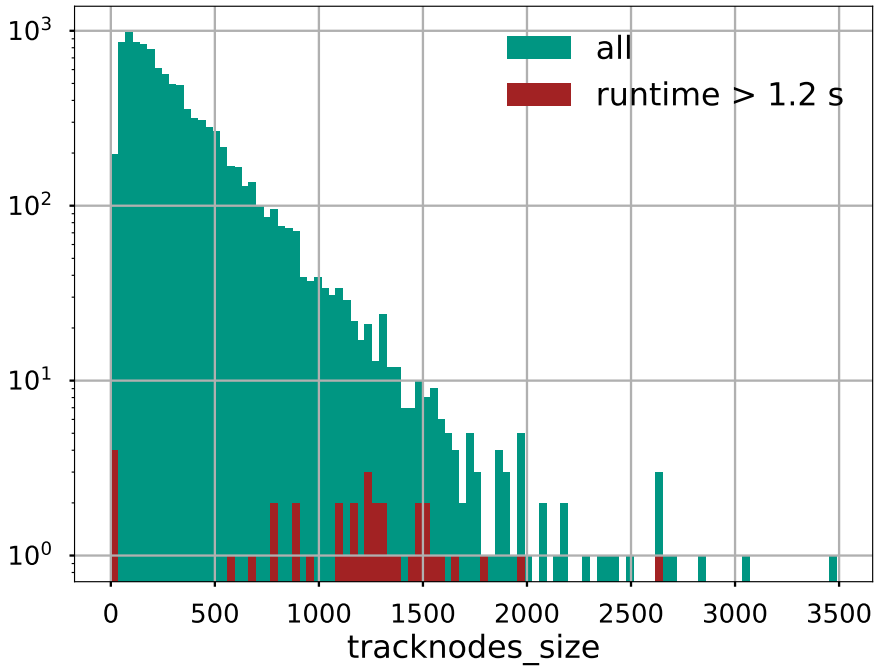


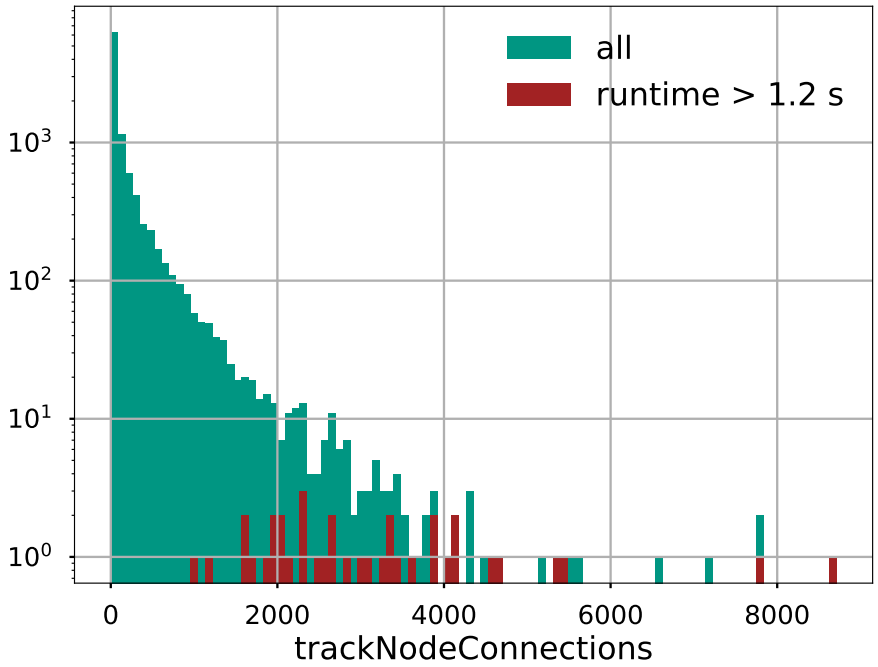


The Number of Active Sectors and the Connections of them does not allow to identify potential problematic events early on.

We already knew that the hits are rather localized in such events...

Exploding Combinatorics – TrackNodes





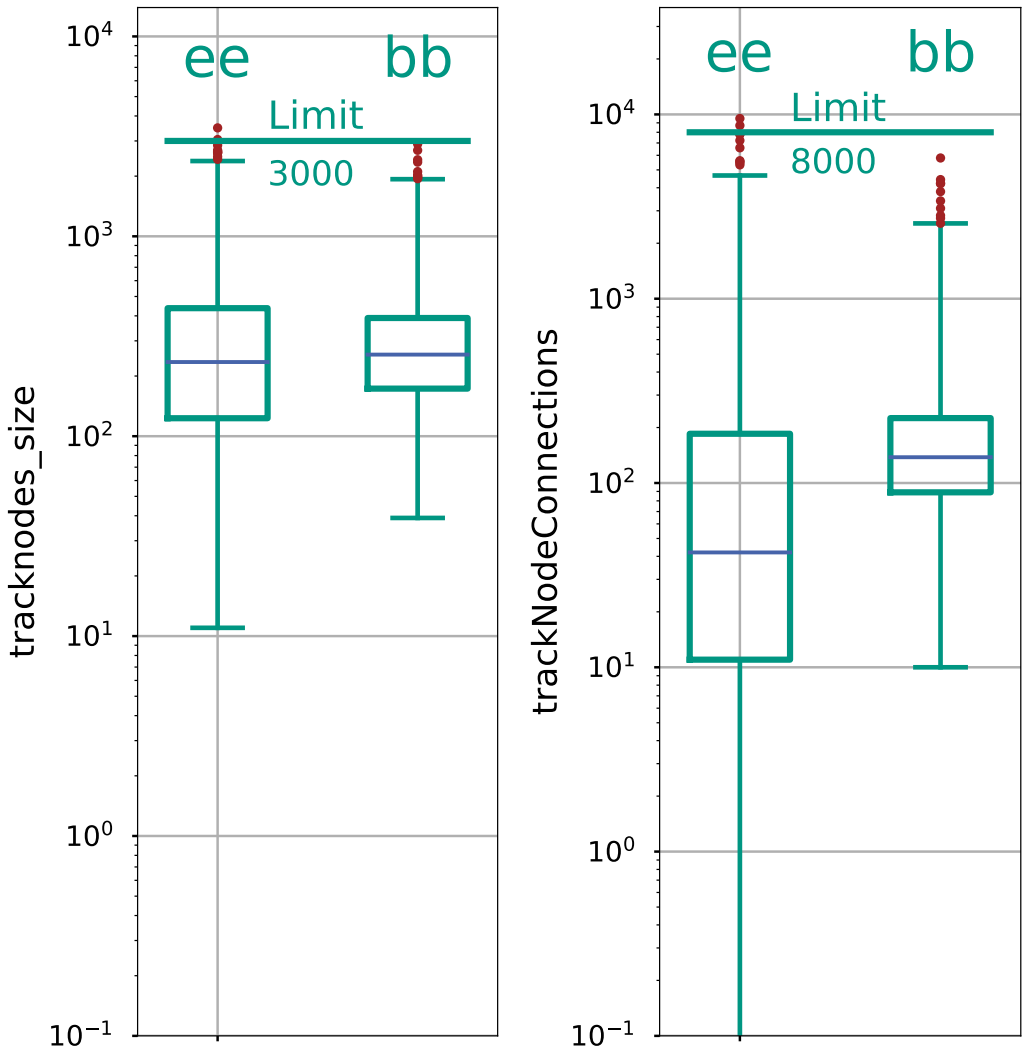
The first possible candidate to get rid of problematic events is the **Number of TrackNode Connections** made in the 2nd Step of the SegmentNetworkProducer.

Exploding Combinatorics – TrackNodes

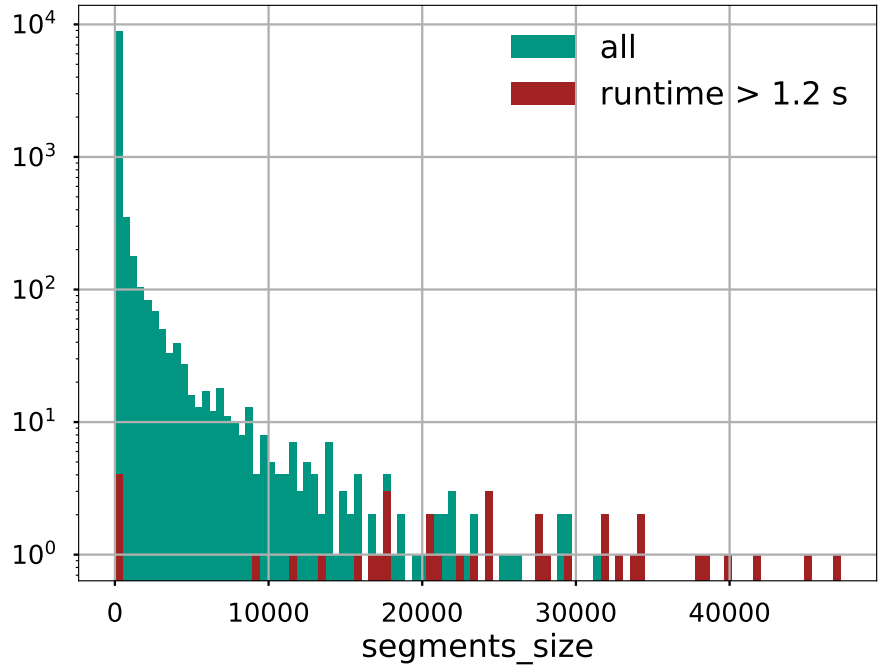
$\Rightarrow$ Intruduced cuts on these qualtities, after evaluating them vor Bhabha and $\Upsilon(4S)$ events.

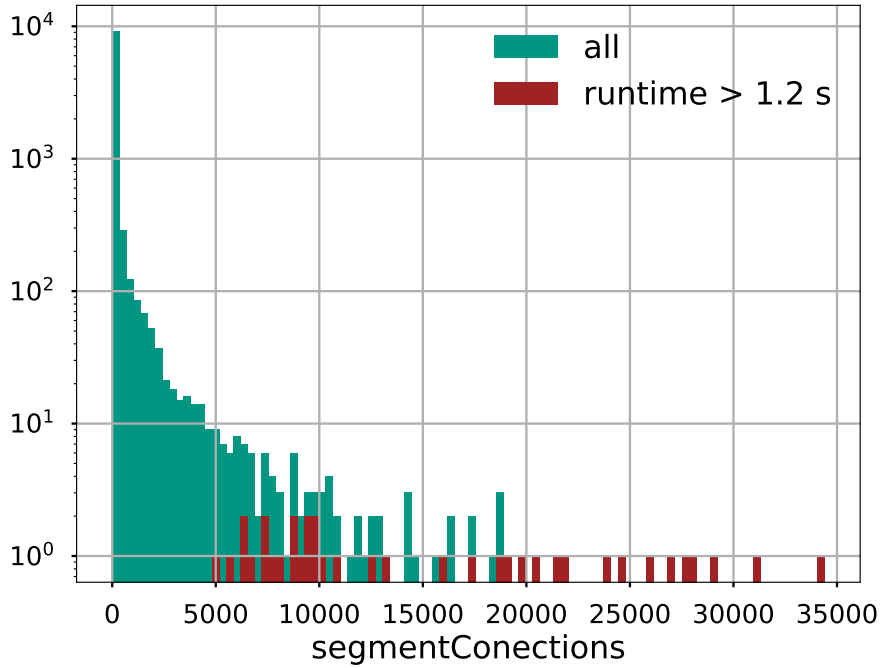
Cut on number of Track-Nodes, will be removed again...

The motivation for this cut probably came when the script I used was still buggy and I just fine-tuned it now, to work again.



Exploding Combinatorics – Segments

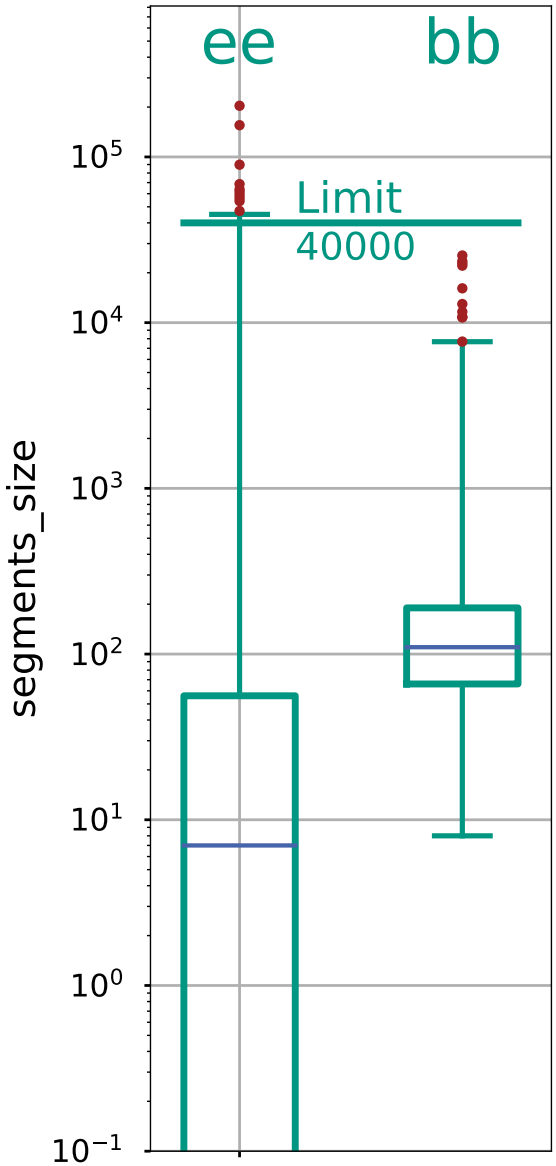


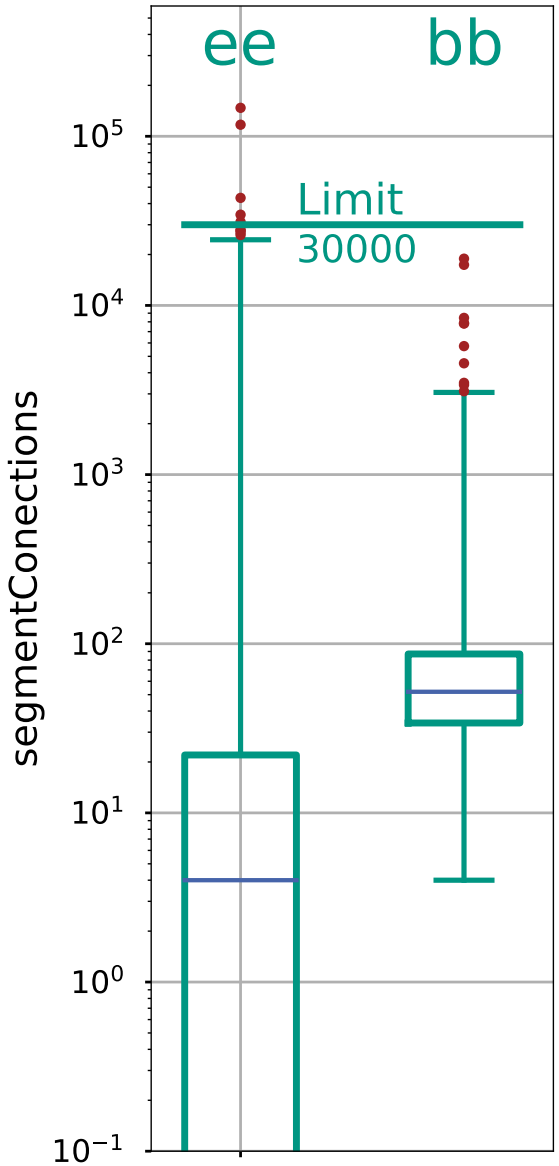


The **Number of Segments** (left) is the quantity which was already used to cut on.

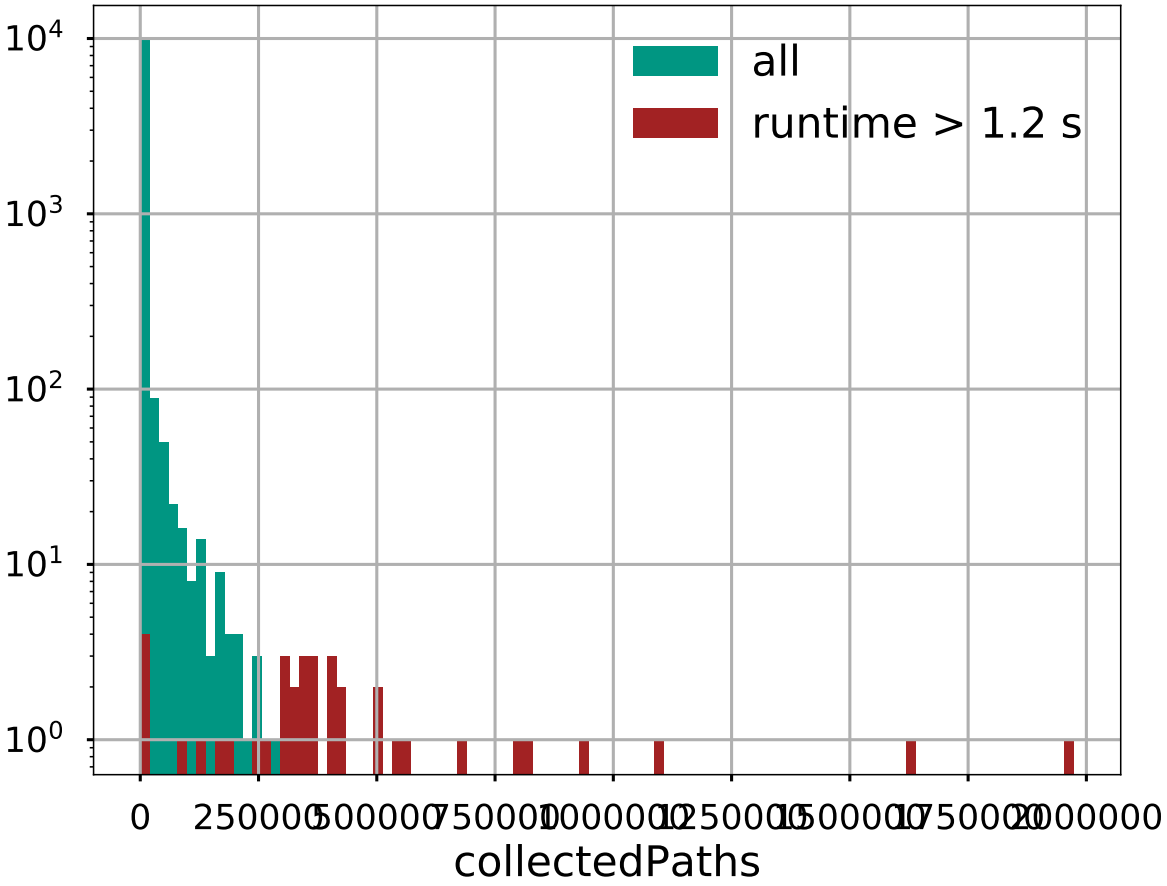
The **Number of Connections among Segments** made during the 3rd Step of the SegmentNetworkProducer also qualifies as a quantity for a cut!

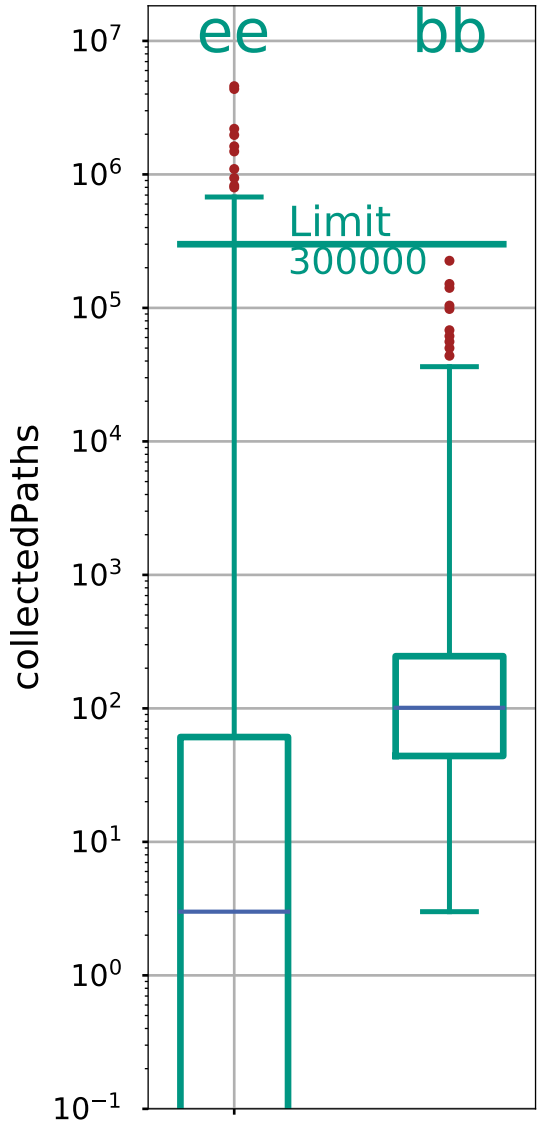
Exploding Combinatorics – Segments





Exploding Combinatorics – Collected Paths





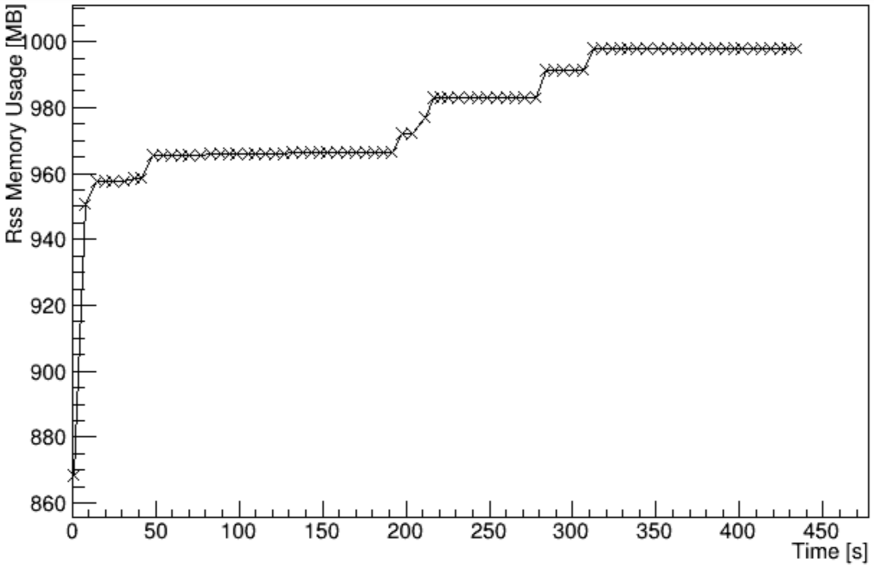
The **Number of Collected Paths** allows for a very efficient cut, but can not catch every problematic case.

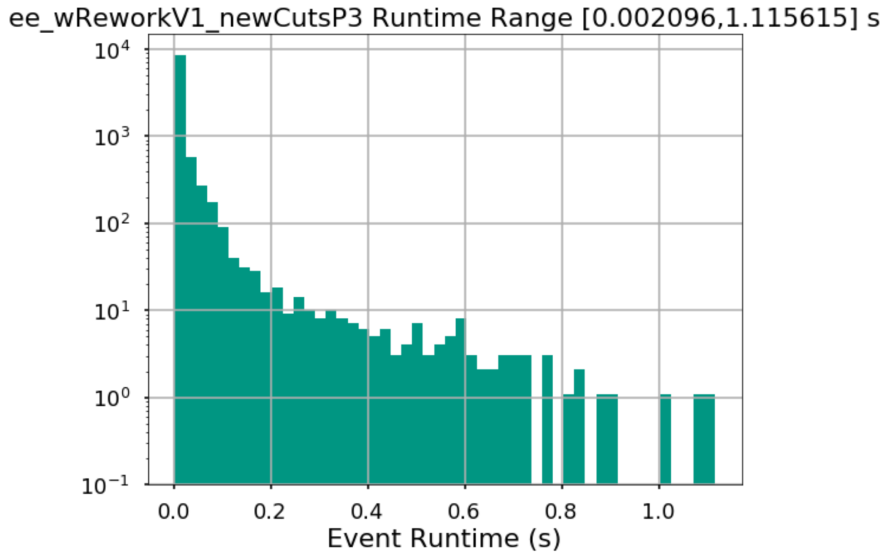
The Cuts

The combined application of the cuts

(Number of TrackNodes $<$ 3000)
Number of TrackNode Connections $<$ 8000
Number of Segments $<$ 40000
Number of Segment Connections $<$ 30000
Number of Collected Paths $<$ 300000

reduces the runtime and the memory consumption for Bhabha events.





Refactoring of VXDTF2

I also started to refactor the code related to the SegmentNetwork creation and the Path Finding to

- improve readability and maintainability;
- optimize the used algorithms;
- identify possible memory leaks or the reason for the ever rising memory consumption;
- reduce the number of necessary cuts to one by defining the Families already during the SegmentNetwork creation.

Refactoring of VXDTF2

The first changes made in the scope of the refactoring are already taken into account in the shown results.

An evaluation of the performance improvement due to these changes can be made using $\Upsilon(4S)$ events as these are not affected by the cuts.

Mean Module Runtime in ms	before Rework	after Rework
SegmentNetworkProducer	2.6	1.5
TrackFinderVXDCellOMat	2.3	1.7

Figures of Merit

The Figures of Merit remain unchanged for $\Upsilon(4S)$ events, as expected.

	bb_noRework_phase3	bb_wReworkV1_newCutsP3_phase3	bb_wReworkV1_newCuts_phase3	bb_wReworkV1_noCuts_phase3
Finding efficiency (primary)	95.72	95.72	95.72	95.72
Finding efficiency	92.91	92.91	92.91	92.91
Fake rate	5.98	5.98	5.98	5.98
Mean Number of Fakes	0.67	0.67	0.67	0.67
Clone rate	2.01	2.01	2.01	2.01
Hit efficiency (primary)	94.96	94.96	94.96	94.96
Hit efficiency	94.30	94.31	94.31	94.31
Hit purity (primary)	98.72	98.73	98.73	98.73
Hit purity	98.55	98.55	98.55	98.55