Muon Shield optimization for SHiP experiment as HTC MC task
GRID'2025

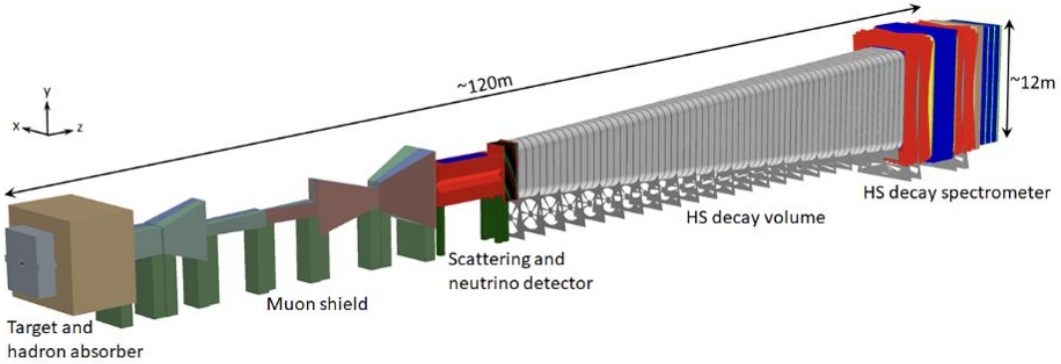
E. Kurbatov [1]
F. Ratnikov [1]

[1]HSE University

LAMBDA • HSE

Dubna, 2025

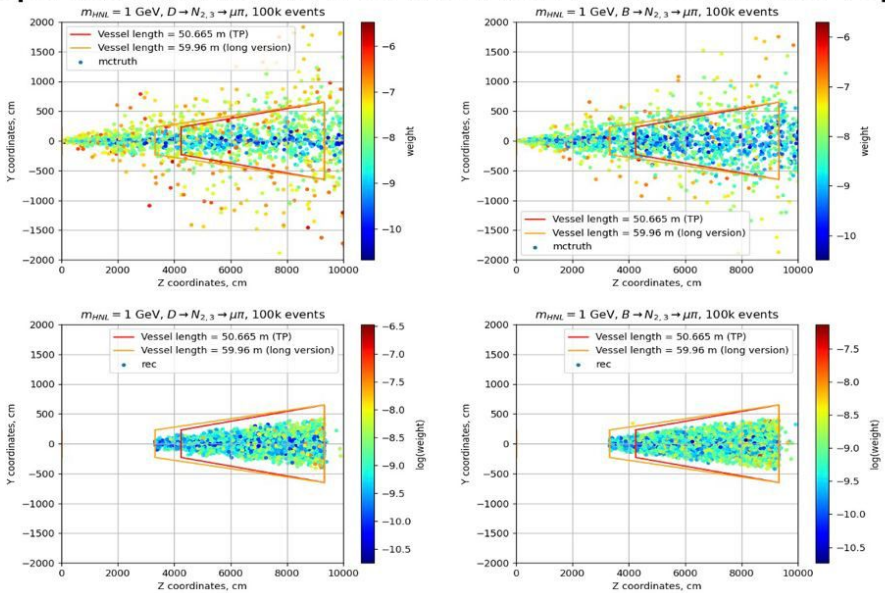
SHiP Experiment



- The Search for Hidden Particles (SHiP) - new experiment on SPS CERN
- Fixed target
- $2x10^{20}$ target-proton interactions per 5 years, $4x10^{13}$ protons per second

HNL sensitivity as a function of decay volume position

Spatial distribution of the HNL vertices. Color represents weight

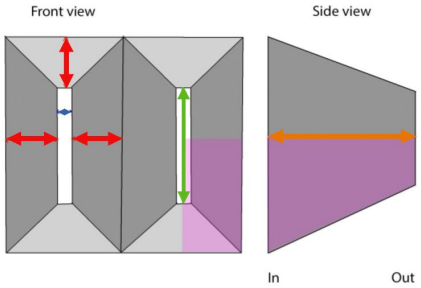


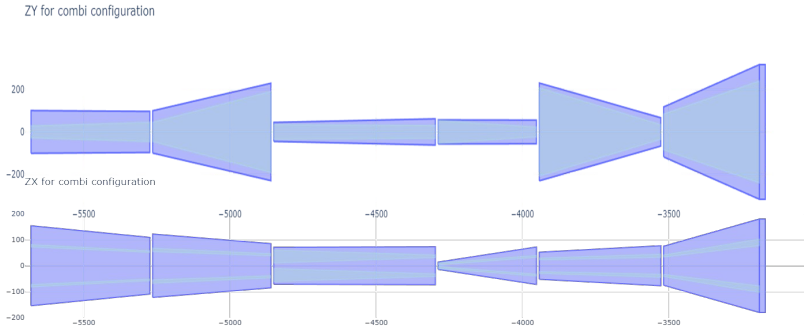
Color distribution strongly depends on gamma!

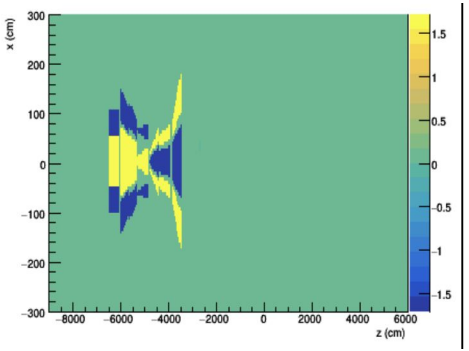
$$w = \frac{(L_2 - L_1)}{\beta\gamma c\tau} \; e^{-\frac{L_S}{\beta\gamma c\tau}}$$

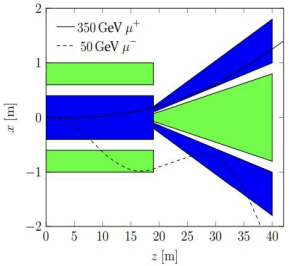
Closer to the target – higher sensitivity!

Muon Shield





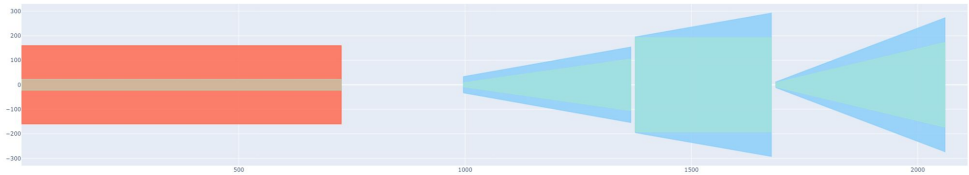


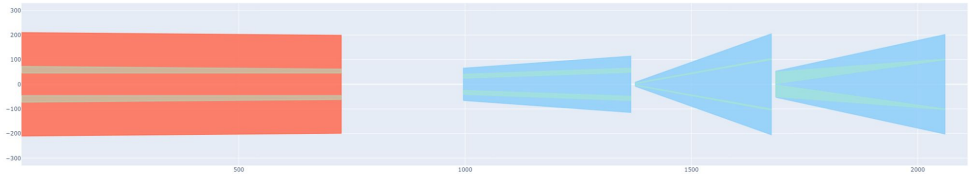


- One of the key components is the muon shield—a magnetic deflection system approximately 30 meters long, with a magnetic field of 1.7 T.
- Up to 42 parameters are required to define the muon shield's configuration.
- The shielding effectiveness remains a topic of debate and discussion among several research groups.
- The expected muon flux suppression is approximately 6 orders of magnitude, reducing the initial 10^{11} muons per spill.
- The process of muons traversing the shielding is highly stochastic.

SC-Muon Shield

- **Proposal**: Replace the first three magnets of the shield with a single short superconducting (SC) magnet (~5 T) to maintain the same integrated magnetic field strength.
- Optimization cycle (excluding FCN) remained unchanged.
- The number of parameters was reduced to 29 (with the gap between the warm and SC sections now treated as a parameter).

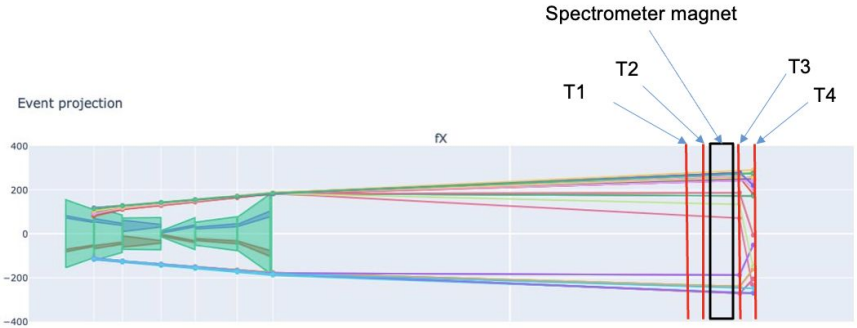


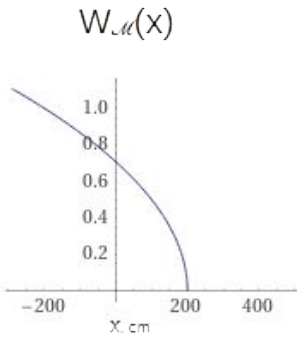


Loss Function

$$(1 + e^{\frac{10 \cdot (M - M^*)}{M^*}}) * (1 + \Sigma W_\mu)$$

M - Muon Shield weight
M* - Constant
W_μ - Nonlinear function of the x-coordinate for a muon in T1





The dedicated muon sample is used for MC:

- Generation stage weights have been removed.
- The sample size has been reduced to the minimum required for stable results.

New Loss Function

Initial objective: the primary loss function was designed to optimize muon flux through the tracking station.

Additional requirements introduced later:

Minimizing flux through the SND, a penalty term was added to the base loss function to regulate SND-bound muons.

Minimizing flux through the veto detector (surrounding t decay volume):

A separate monitoring loss function was implemented (r direct optimization).

Optimization was performed exclusively using the augmented function from point 1.

$F = W * (T_{flux} + SND_{flux} + 1)$

$W = a_1 * W_{sc} + a_2 * W_{warm}$

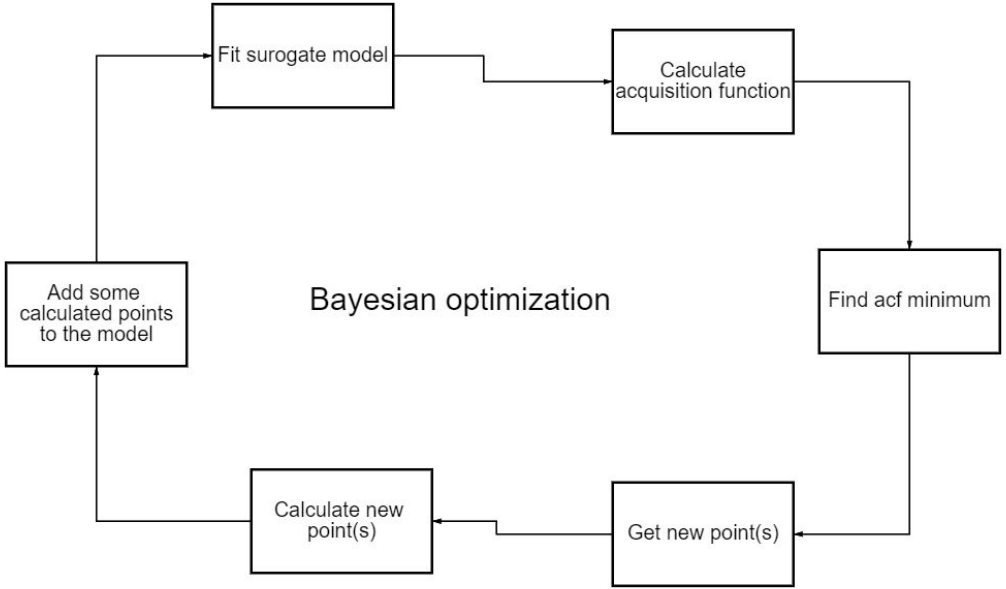
$T_{flux} = a_3 * \max(0, (T - T_{good}))$

$SND_{flux} = a_4 * \max(0, (S - S_{good}))$

a_i – tunable parameters, $T(a_i)$ – muon flux through tracking stations (equivalent to previous definition)

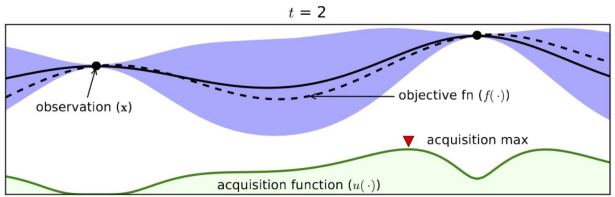
$S(a_i)$ – Muon flux through SND (equivalent to previous definition)
T□ᵢ□ / S□ᵢ□ – "Good enough" flux values

Bayesian optimization



Global Bayesian Optimization of a "Black Box" System delivers excellent results when the following conditions are met:

- Low dimensionality (<10 parameters)
- Continuous loss function (smooth behavior)
- Simple parameter space topology (no extreme nonlinearities or discontinuities)
- No gradient information available for the loss function



And if we consider non-gaussian prior?

Our algorithm (WU-GO) is fundamentally an LCB (Lower Confidence Bound) approach, but with a custom uncertainty estimator replacing the predictive posterior variance.

Key Features:

- Balances exploitation-exploration within the LCB framework.
- Accounts for response stochasticity, similar to EGO (Efficient Global Optimization).

Algorithm 1 Wasserstein Uncertainty Global Optimisation (WU-GO)

Input: Ground truths $\mathcal{M}$, generator G, grid $\tilde{\Theta}$, parameter κ
Output: Optimal configuration $\hat{\theta}^*$

while stopping criteria are not met **do**
  Fit G on $\mathcal{M}$
  Approximate f: $\hat{f}(\theta) = \mathbb{E}[G(\theta)] \approx \frac{1}{n}\sum_{j=1}^{n} x_j, x_j \sim G(\theta)$
  Estimate $\sigma_{\mathbb{W}}$: $\hat{\sigma}_{\mathbb{W}}(\theta) = \min_{\mu \in \mathcal{M}} D(\mu, G(\theta))$
  Predict $\hat{\theta} = \arg\min_{\theta \in \tilde{\Theta}} \{\hat{f}(\theta) - \kappa \cdot \hat{\sigma}_{\mathbb{W}}(\theta)\}$
  Call simulator for $\hat{\theta} : \hat{\mu}$
  $\mathcal{M} = \mathcal{M} \cup \{\hat{\mu}\}$
end while

https://github.com/hse-cs/waggon

And if we want to achieve a local minimum?

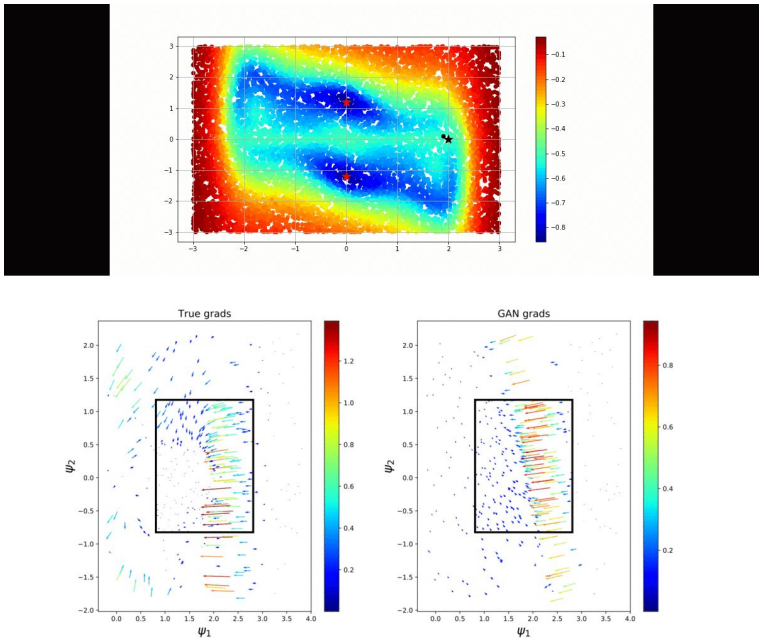
Challenges in High-Dimensional Optimization - Curse of dimensionality: global minima become exponentially harder to locate as parameter space grows.

Gradient descent is effective for local optimization but fails due to pathological curvature (e.g., ravines, saddle points) in the objective function and inability to backpropagate gradients through MC simulations (black-box stochasticity).

Proposed Solution - after known high-quality initial guess near the global minimum is found:

Train a generative surrogate model (e.g., neural network) to approximate simulation outputs.

Use the surrogate to enable gradient-based refinement from the initial point.



https://arxiv.org/abs/2002.04632

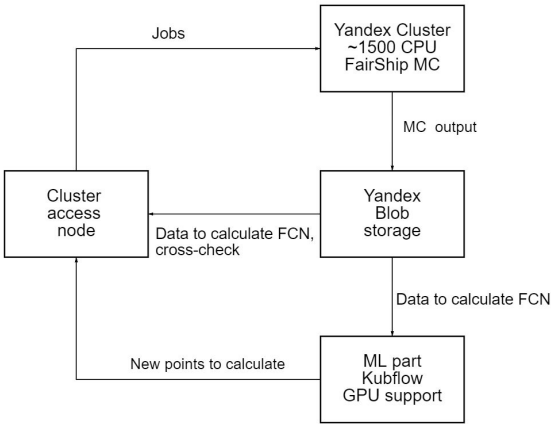
Computing setup

The research was supported by **Yandex.Cloud**, really - my deep personal thanks!

- 54 Nodes
- ~ 58 GB RAM per Node
- ~ 30 CPU per Node
- Not that much storage
- No any orchestration provided

3 different control approach was used during several years of the experiments:

- Wonderland (failed to find any actual information now)
- Disneyland:
  - Go language based docker orchestrator
  - https://github.com/skygrid/disneyland/
  - MC results was uploaded to Cern EOS for the further processing
- Kubernetes:
  - actual setup
  - S3 Yandex Blob storage as an output for the further processing



Simulation task

To calculate a single point for the optimization we need:

- Generate new Shield parameters and pass them to the control Node
- Wait for resources to be free, and run #NTasks to calculate the total #NEvents from the MC sample(NEvents/NTasks per task)
- Wait all the tasks are completed successfully, re-run task if anything failed (if the number of re-runs is not too much, mark as “failed” otherwise)
- Process outputs, calculate FCN and uncertainties, run additional set of tasks if the statistical uncertainty is not satisfying
- Add new calculated point to surrogate and start a new iteration

The time to obtain a single iteration FCN value depends on:

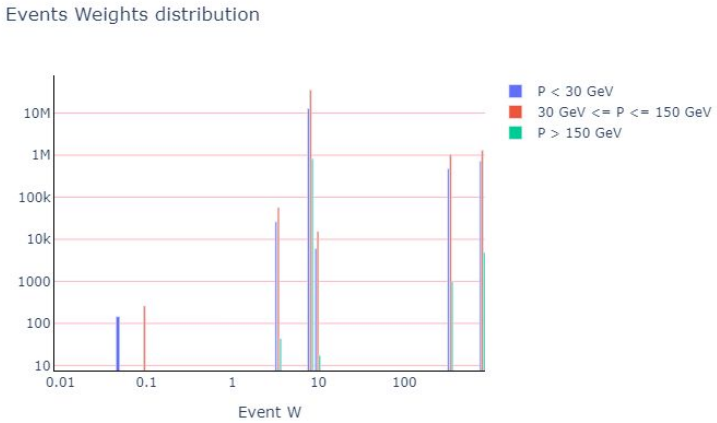
- total events amount to simulate (k*NEvents, typical k value is 1 actually, more simulation only for good points).
- NTasks - the number of jobs to which the initial MC sample is splitted. More tasks - less events per task to calculate, but there is some fixed time to start/process task, no need to split to. Also RAM is consumed by every task and actually this is the bottleneck.

Simulation sample

The number of events in the initial sample critically determines the simulation runtime. There are 2 main samples:

- Full sample: 68GB of data, is stored locally on the every node, matches the one spill, is used to estimate the detector performance characteristics
- Optimisation sample: 50MB of data, stored inside the docker image, sampled from the full one, big enough to make the optimization results statistically stable in most cases

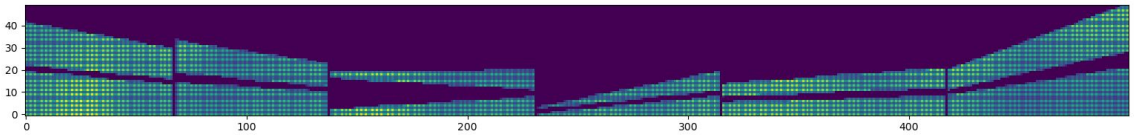
During the investigation on the statistical uncertainties in optimization instabilities the problem of generation-step weight in the initial event sample was found. The optimization sample was created with “weights unrolling” approach, this allowed to reduce the size of the sample by 2 orders of magnitude while the initial number of events was kept. This approach was added to the standard FairSHiP generator code as an option.



RAM issues

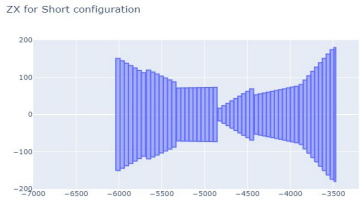
The performance of the FairSHiP builds was controlled with Intel VTune software. Still the RAM amount on Node was the main bottleneck in the configuration used to run MC. After one of the simulation software update the RAM consumption was doubled. After short investigation the cause was found - migration from flat geant fields to the field maps! The field maps were stored in the RAM as key-value pairs, where 3-dimensional coordinates were used as keys. To reduce the RAM consumption 2 ideas was implemented asap:

- Keep only ¼ of field maps, use the detector symmetry to recalculate the other parts.
- Remove keys, keep just the field values in some organized order as an array, store step sizes and overall field dimensions and center position - this allow to calculate array index from 3d-coordinate on-the-fly.

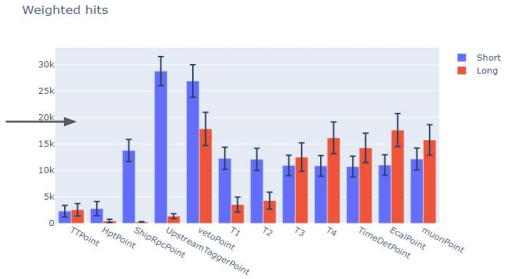


Optimization Stability issues

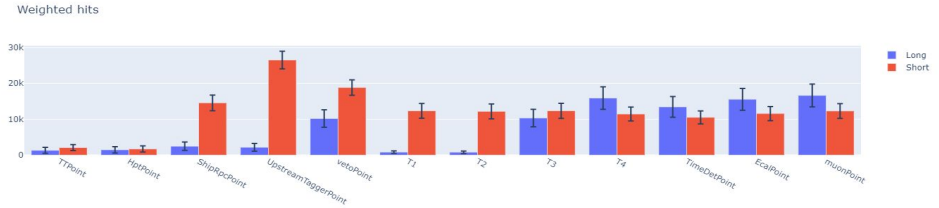
Even with all hacks on previous slides still the optimization result stability could be improved with synthetic field maps and some geometry fluctuations.

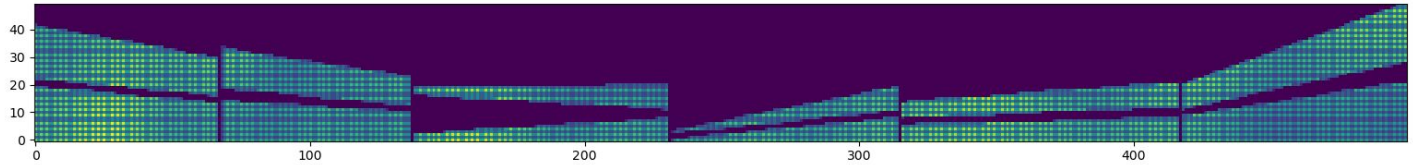


With steps

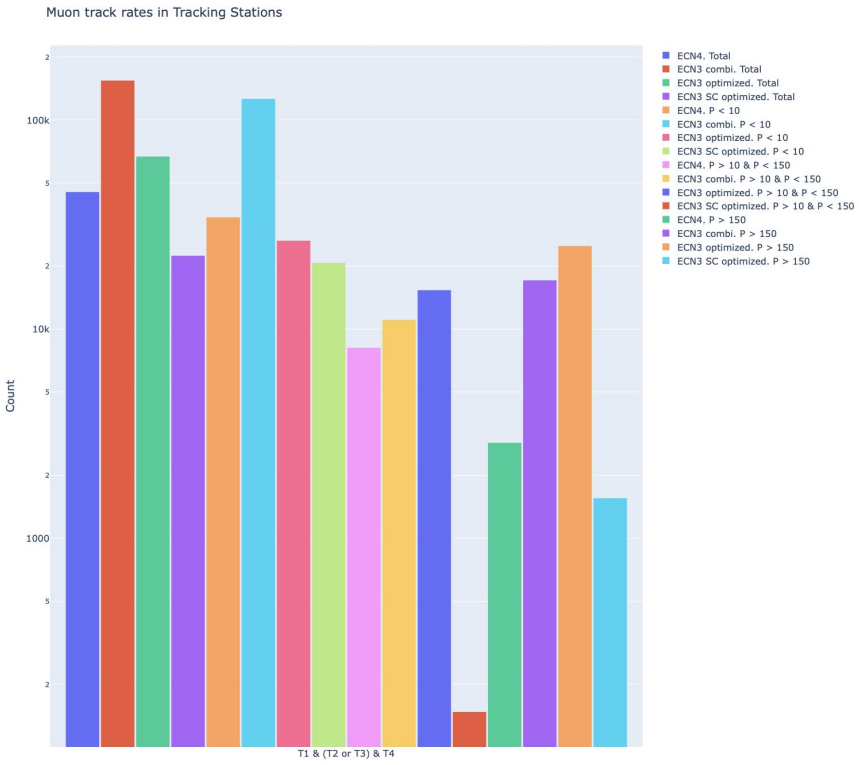


Without steps





Optimization results



Shield	Rate [muons/spill]	Length [m]
ECN4	45k	35
ECN3 Combi	160k	30
ECN3 Optimized	67k	30
ECN3 SC Optimized	22k	20

Conclusions

- FairSHiP software stack is pretty complex and some additional efforts was needed to make a convenient docker image updates possible
- Thousand of mc task should be run simultaneously to achieve the acceptable speed of optimization process
- The MC software is profiled to provide the balanced load on resources avaible
- The optimization pipeline works smooth and provided a good results for several optimization tasks already