

Профилирование пакета программ SpdRoot

Дидоренко Алексей Викторович
(стажёр-исследователь, ЛИТ ОИЯИ, didorenko@jinr.ru)

Войтишин Николай Николаевич
(к. ф. -м. н., ЛИТ ОИЯИ, nvoytish@jinr.ru)

Актуальность

Spin Physics Detector



International spin physics collaboration at the collider NICA

[General information](#) [Collaboration](#) [Presentations and publications](#) [Setup](#) [Internal access](#)

General Information

[SPD CDR & TDR](#)

[NEWS AND ANNOUNCEMENTS](#)

[UPCOMING CONFERENCES](#)

[CONTACTS](#)

[USEFUL LINKS](#)

Collaboration

[PARTICIPATING INSTITUTIONS](#)

[EXECUTIVE BOARD](#)

[TECHNICAL BOARD](#)

[PUBLICATION COMMITTEE](#)

[DOCUMENTS](#)

SPD Presentations

SPD Software

[SPD Software Wiki](#)

Monte Carlo simulation, event reconstruction for both simulated and real data, data analysis and visualization are planned to be performed by an object oriented C++ toolkit SPDroot. It is based on the FairRoot framework initially developed for the FAIR experiments at GSI Darmstadt and partially compatible with MPDroot and BM@Nroot software used at MPD and BM@N, respectively.

The SPD detector description for Monte Carlo simulation is based on the ROOT geometry while transportation of secondary particles through material of the setup and simulation of detector response is provided by GEANT4 code. The standard multipurpose generators like Pythia6 and Pythia8 as well as specialised generators can be used for simulation of primary nucleon-nucleon collision.

- [GIT Repository](#).

SpdRoot - пакет программ, который способен выполнять моделирование событий методом Монте-Карло, реконструкцию, анализ и визуализацию событий.

Утверждается, что реконструкция работает медленнее ожидаемых скоростей обработки события.

Актуальным вопросом этого проекта является поиск узких мест в исходном коде программы и дальнейшее повышение скорости обработки и эффективности использования вычислительных ресурсов.

Цель работы

Цель работы: найти узкие места процесса реконструкции событий в исходном коде SpdRoot.

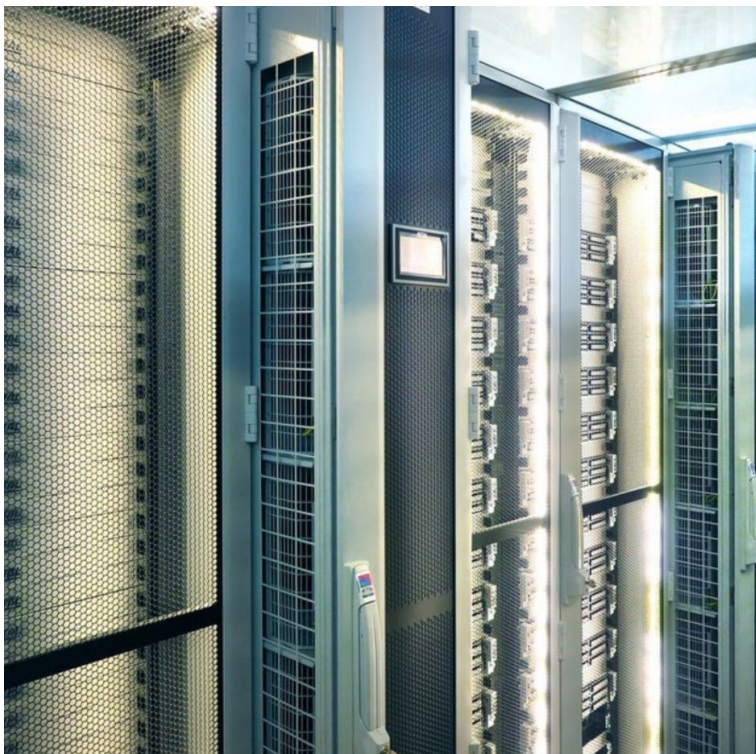
Задачи:

для функций реконструкции измерить:

- затрачиваемые ресурсы;
- время выполнения.

Изучить влияние типа поля на скорость реконструкции.

Стек технологий



```
laxddid@ncx104 ~]$ perf

usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]

The most commonly used perf commands are:
annotate      Read perf.data (created by perf record) and display annotated code
archive       Create archive with object files with build-ids found in perf.data file
bench         General framework for benchmark suites
buildid-cache Manage build-id cache.
buildid-list  List the buildids in a perf.data file
c2c           Shared Data C2C/HITM Analyzer.
config        Get and set variables in a configuration file.
data          Data file related processing
diff          Read perf.data files and display the differential profile
evlist        List the event names in a perf.data file
ftrace        simple wrapper for kernel's ftrace functionality
inject        Filter to augment the events stream with additional information
kallsyms      Searches running kernel for symbols
kmem          Tool to trace/measure kernel memory properties
kvm           Tool to trace/measure kvm guest os
list          List all symbolic event types
lock          Analyze lock events
mem           Profile memory accesses
record        Run a command and record its profile into perf.data
report        Read perf.data (created by perf record) and display the profile
sched         Tool to trace/measure scheduler properties (latencies)
script        Read perf.data (created by perf record) and display trace output
stat          Run a command and gather performance counter statistics
test          Runs sanity tests.
timechart     Tool to visualize total system behavior during a workload
top           System profiling tool.
version       display the version of perf binary
probe         Define new dynamic tracepoints
trace         strace inspired tool

See 'perf help COMMAND' for more information on a specific command.
```



Метод поиска - профилирование

Профилирование используется для наблюдения за выполнением программы для сбора данных о различных аспектах, таких как:

- время выполнения;
- затрачиваемые ресурсы.

Цель профилирования — найти узкие места или области, где программа может быть оптимизирована для повышения ее эффективности и производительности.



perf как инструмент для анализа производительности ПО

perf - инструмент профилирования, который предназначен для систем на базе Linux. Преимущества:

- простой интерфейс командной строки
- богатый функционал.

Для анализа производительности SpdRoot использовались такие perf - команды как:

- perf record
- perf report
- perf probe

```
[alxdid@ncx104 ~]$ perf

usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]

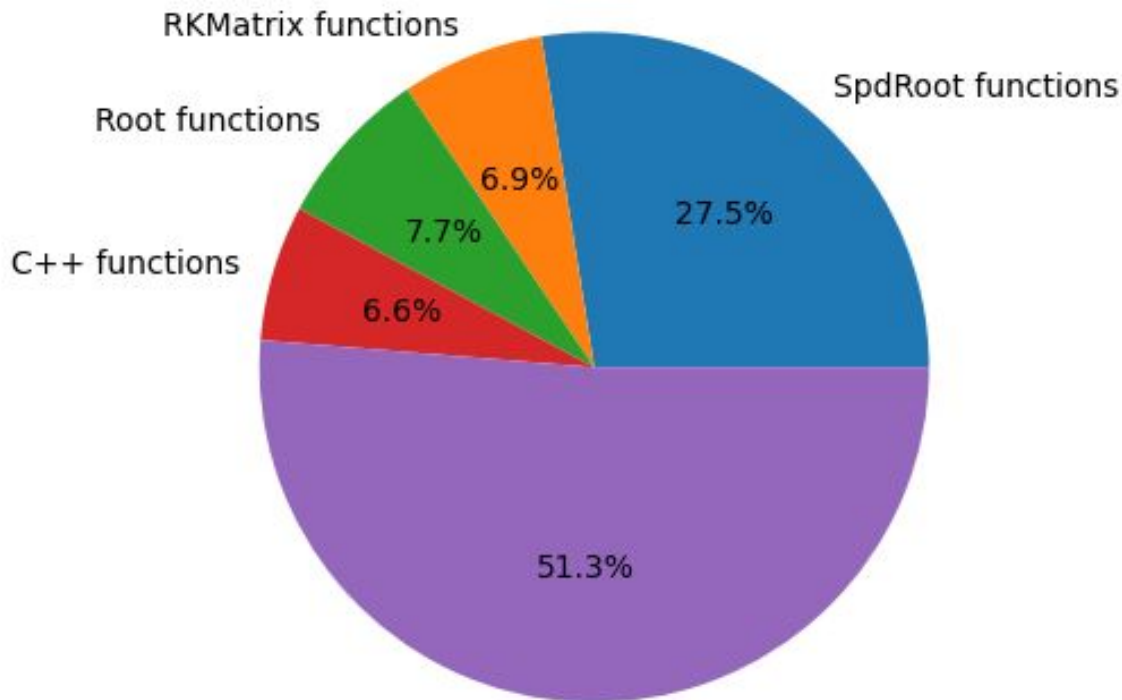
The most commonly used perf commands are:
  annotate  Read perf.data (created by perf record) and display annotated code
  archive  Create archive with object files with build-ids found in perf.data file
  bench    General framework for benchmark suites
  buildid-cache  Manage build-id cache.
  buildid-list  List the buildids in a perf.data file
  c2c      Shared Data C2C/HITM Analyzer.
  config   Get and set variables in a configuration file.
  data     Data file related processing
  diff     Read perf.data files and display the differential profile
  evlist   List the event names in a perf.data file
  ftrace   simple wrapper for kernel's ftrace functionality
  inject   Filter to augment the events stream with additional information
  kallsyms Searches running kernel for symbols
  kmem     Tool to trace/measure kernel memory properties
  kvm     Tool to trace/measure kvm guest os
  list     List all symbolic event types
  lock    Analyze lock events
  mem     Profile memory accesses
  record   Run a command and record its profile into perf.data
  report   Read perf.data (created by perf record) and display the profile
  sched   Tool to trace/measure scheduler properties (latencies)
  script  Read perf.data (created by perf record) and display trace output
  stat    Run a command and gather performance counter statistics
  test    Runs sanity tests.
  timechart  Tool to visualize total system behavior during a workload
  top     System profiling tool.
  version display the version of perf binary
  probe   Define new dynamic tracepoints
  trace   strace inspired tool

See 'perf help COMMAND' for more information on a specific command.
```

Характеристики в случае поля 1/8 общего объёма.

```
SpdFieldMap1_8 *MagField = new SpdFieldMap1_8("full_map");
MagField->InitData("field_full1_8.bin");
SpdRegion *reg = MagField->CreateFieldRegion("box");
reg->SetBoxRegion(-330, 330, -330, 330, -386, 386); //
(X,Y,Z)_(min,max), cm
run->SetField(MagField);
```

Характеристики в случае поля $1/8$ общего объема. Затрачиваемые ресурсы



$\leq 0.5\%$ overhead reconstruction functions

SpdRoot functions (27.5%) top CPU users

6.97% **genfit::RKTrackRep::RKPropagate** - старый алгоритм (2013) часть GENFIT. Экстраполяция методом Рунге-Кутты, много операций с матрицами.

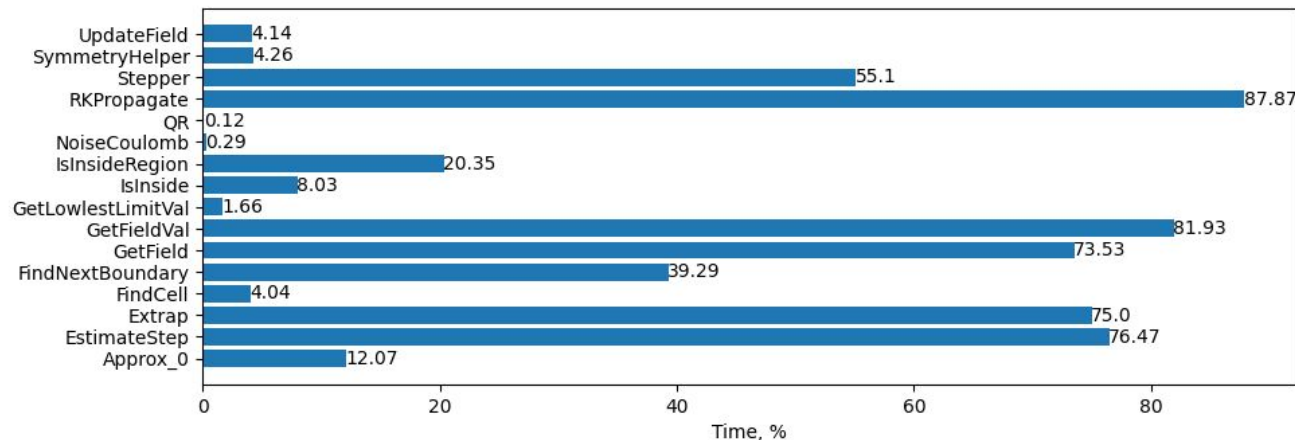
5.60% **SpdFieldMap1_8::Approx_0** - функция состоит из перемножения и сложения векторов и матриц

2.14% **SpdFieldMap1_8::FindCell** - поиск ячейки в поле которой принадлежит точка

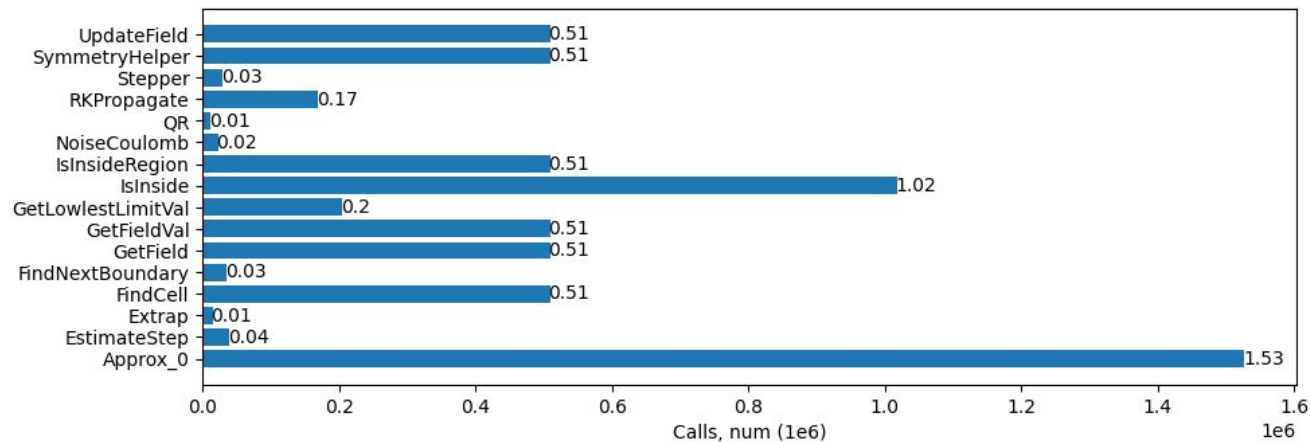
1.61% **SpdFieldMap1_8::GetField** - возвращает значение поля в точке

1.52% **SpdBoxRegion::IsInside** - проверка находится ли точка внутри области вокруг точки (r,z)

Характеристики в случае поля 1/8 общего объема. Время выполнения



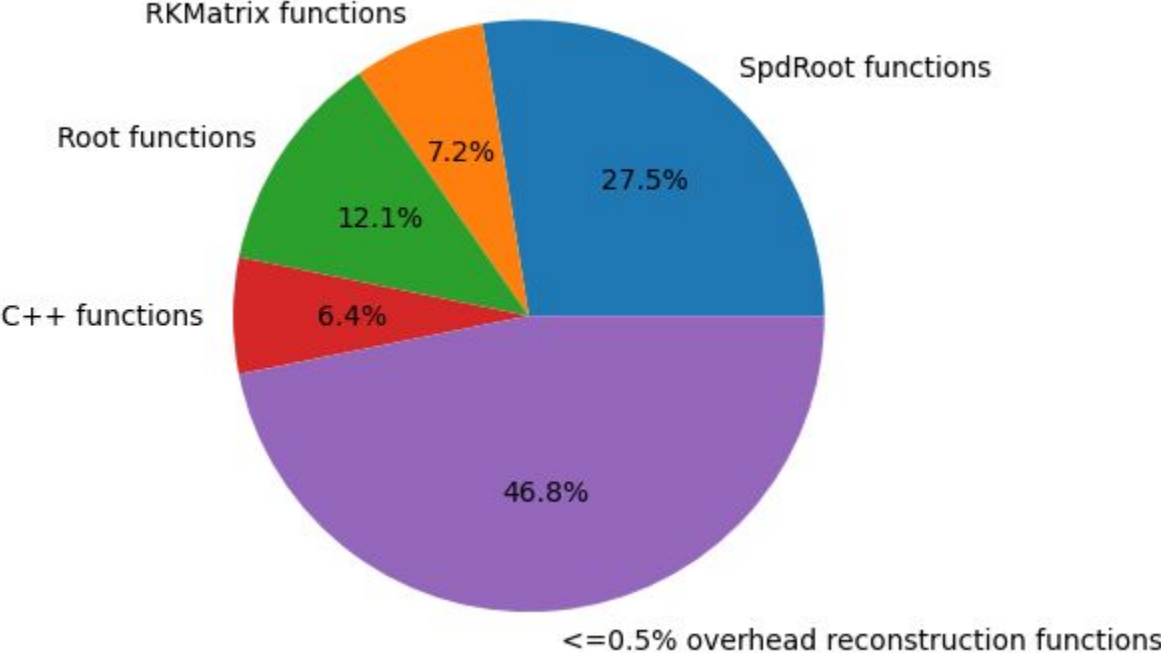
Total time = 124,0379 sec
(с учётом затрат на perf)



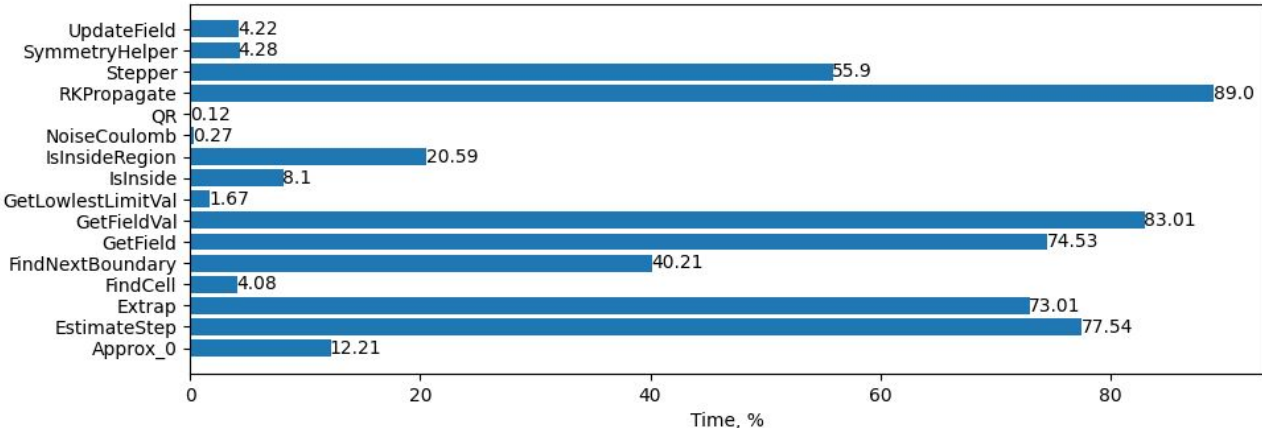
Характеристики в случае константного поля

```
SpdConstField* MagField = new SpdConstField();  
MagField->SetField(0., 0., 10.0); // kG  
SpdRegion* reg = 0;  
reg = MagField->CreateFieldRegion("tube");  
reg->SetTubeRegion(0, 174, -246, 246); // (R,Z)_(min,max),  
cm  
run->SetField(MagField);
```

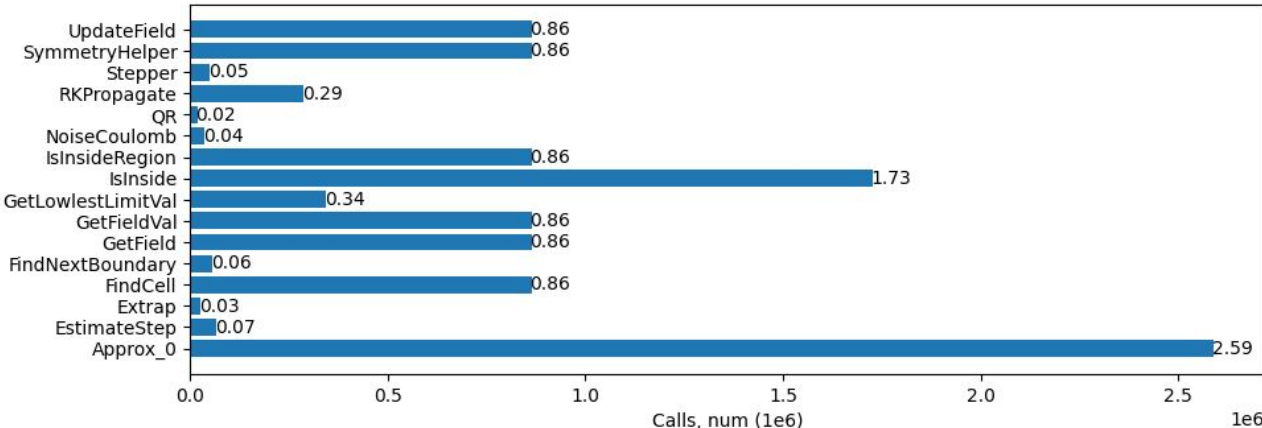
Характеристики в случае константного поля. Затрачиваемые ресурсы



Характеристики в случае константного поля. Время выполнения



Total time = 205,3638 sec
(с учётом затрат на perf)



Заключение

- Разные виды полей оказывают влияние на скорость реконструкции, но проценты времени работы функций достаточно близки в разных случаях.
- Наблюдаются различия в количестве вызовов функций.
- Root - функции занимают больше ресурсов в случае константного поля.

Планы:

- Получить больше статистических данных на большем количестве событий и на других видах магнитных полей.
- Удостовериться, что параметры поля влияют на скорость реконструкции.

Спасибо за внимание!