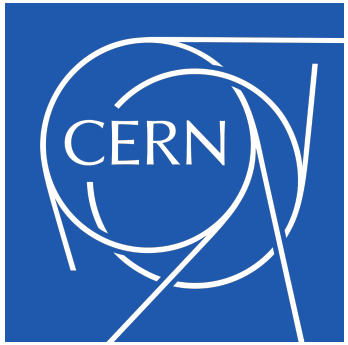

Profile-guided optimization

Speeding up LHCb software through compilation optimization

Oscar Buon

June 19, 2023

Supervisor: Sebastien Ponce

Table of contents

CERN

Intergovernmental organization on France-Switzerland border.
About 15,000 people working at CERN.
Large Hadron Collider (LHC): particle collider of 27 km.

LHCb

One of the 4 main experiments installed on the LHC.
More than 1,200 people working for the collaboration.

Computing

Analyzes data from the detector. Several TB/s.
The computing group has to maintain software with several million lines of code which some are 30 years old. Moreover the code is mainly written by non-software engineers.

Context

Worked on a stack with one main CMake:

`https://gitlab.cern.ch/clemenci/lhcb-super-project-template/`

Target: `x86_64_v3-centos7-gcc11+detdesc-opt+g`

Graphviz for printing graphs

Graphviz in CMake to create dependency graphs:

- CMake arg: `--graphviz=path/to/files.dot`
- Convert to SVG:
  `dot -Tsvg -o path/to/file.svg path/to/file.dot`
- Make arg: `CMAKEARGS="--graphviz=path/to/files.dot"`
- Setting `GRAPHVIZ_EXTERNAL_LIBS` to `FALSE` is usefull.

Static libraries

`gaudi_add_library` ⇒ `gaudi_add_static_library`
Replace `SHARED` by `STATIC` in `add_library` function call.

Static modules

By using previous static libraries. Need to add:

- `-Wl,--whole-archive` link option to the target and link them to the final executable.
- `-Wl,--whole-archive` and `-Wl,--allow-multiple-definition` link option to the executable.
- the global link option `-Wl,--export-dynamic`

Without these options modules cannot be initialized.
There may still be bugs with the functors that can't find some symbols.

The standard method to run test with python is not working. Need to run directly the executable with a json options file:

```
build.x86_64_v3-centos7-gcc11+detdesc-opt+g/run
./build.x86_64_v3-centos7-gcc11+detdesc-opt+
g/Gaudi/Gaudi/Gaudi_static options.json
```

Profile-guided optimization

Principle:

- Compiling the programm
- Running it to create profiles (counters)
- Recompiling the programm with the profiles

Improvements:

- Inlining
- Block ordering
- Register allocation
- Virtual call speculation
- Dead code separation

Link-time optimization

Allow the linker to perform optimizations that take account of all translation units.

Final building pipeline

- Compiling the programm with `-fprofile-generate`
- Running it to create profiles
- Recompiling the programm with `-flto -fprofile-use -fprofile-correction`

Results

Test: `hlt2_pp_thor` (6×1000 events)

Optimization	Acceleration	Confidence interval (2σ)
LTO	0.17%	$\pm$1.12%
LTO & PGO	6.74%	$\pm$1.44%
Static LTO	0.87%	$\pm$0.60%
Static LTO & PGO	6.88%	$\pm$0.83%

Conclusion

- Using LTO & PGO makes the programm running faster.
- Using static libraries and modules doesn't seem to be usefull and leads to some bugs.