Simple Compact Tree (SCT): A μ-DST format for LHCb

Hans Dembinski for the MPIK Heidelberg group

Max-Planck-Institut für Kernphysik

SCT in a nutshell

SCT: μ-DST format independent of LHCb framework

- Fast
  - Reading FULL DST: 500 seconds
  - Reading SCT: 5 seconds **100 x faster**
- Compact size
  - FULL DST: 4 GB
  - SCT: 0.46 GB **9 x smaller**
- Simple
  - ROOT TTree with branches of PODs and arrays of PODs
  - No other libs or headers needed
- Self-describing
  - Documentation included in every file
  - No outdated, missing, or obscure documentation
- Extensible
  - New variables can be added without breaking user code
- Prunable
  - Skip over/remove variables you don't need

How we use it

- SCTs are used in min-bias studies where storing the full event is possible and useful
- Produce **one set** of **master SCTs**
  - Subset on lxplus, see /eos/project/l/lhcb/
- Users produce **"stripped" SCTs**
  - Branches may be removed to produce an even more compact SCT file (tool prune_sct.py is provided)
  - Reduced SCTs may fit on your laptop
- Avoids Grid usage after SCT generation
- Fast development cycle
- Use TTree::Scan and TTree::Draw for super-quick plots
- Code is stable

SCT structure

Store events in two-level hierarchy

- Plain variables
  - event: run number, event time, trigger bits...
  - detector: magnet polarity…
- Variable-length arrays of plain variables
  - vertices: x, y, z, …
  - tracks: x, y, z, px, py, pz, ...
  - same for MC particles and MC vertices

Are slowly-changing variables wasting space? No.

- Some variables equal for many events, e.g. run number, magnet polarity
- ROOT's run-length encoding compresses repeated values efficiently

Why variable-length arrays instead of std::vector?

- Serializing vector for each track variable introduces space overhead
- Storing standard vector of struct causes various problems

Excerpt from documentation

Branches:
evt_run : run number
evt_evnum : event number
evt_bxtype : bunch crossing type (see enum LHCb::ODIN::BXTypes)
evt_trtype : trigger type (see enum LHCb::ODIN::TriggerType)
evt_bunchid : bunch ID
evt_mag_pol : magnetic field is -1: pointing down, 1: pointing up, 0: unknown

trig_tck : trigger configuration key
trig_bits : trigger bits for this event, see HLT map below

tracks_* : track counters,
see https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbTrackingStrategies#TrackTypes
[…]
vtx_len : number of primary vertices in this event
vtx_[x,y,z] : position
vtx_[chi2,ndf] : fit quality
vtx_cov : covariance matrix for (x, y, z)
[…]

How to read SCT in user code

https://gitlab.cern.ch/hdembins/V0Analysis see DstConverter/reader_example

```
// filename points to some file
std::unique_ptr<TFile> f{TFile::Open(filename)};
TTree* tree = static_cast<TTree*>(f->Get("sct"));

#define REGISTER_VAR(type, name)             \
     type name = 0;                          \
     if (tree->GetBranch(#name))             \
        tree->SetBranchAddress(#name, &name)

#define REGISTER_ARRAY(type, name, size)     \
     type name[size];                        \
     if (tree->GetBranch(#name))             \
        tree->SetBranchAddress(#name, &name)

constexpr auto vtx_len_max = 256;

REGISTER_VAR(Int_t, evt_run);
// ...
REGISTER_VAR(Short_t, vtx_len);
REGISTER_ARRAY(float, vtx_x, vtx_len_max);
// ...
```

Open for collaborators

- Source code on Gitlab

  https://gitlab.cern.ch/hdembins/V0Analysis

  – DstConverter (DaVinci application) produces SCTs
- Check out README.md for installation instructions
- Participate: Request developer access via Gitlab website

Conclusion

- SCT is a μ-DST format for LHCb in use by MPIK and other groups
- Most branches are detector-independent, adapting SCTs to other CERN experiments is easy
- Users from other groups are welcome

Pros

- Fast
- Compact
- Simple
- Documented
- No LHCb framework

Cons

- No LHCb framework
  (using algorithms that expect LHCb::Track and friends is more cumbersome)

https://gitlab.cern.ch/hdembins/V0Analysis