CMS's Need for ROOT I/O

Christopher Jones *FNAL*

ROOT 6

Congratulations on releasing ROOT 6 on schedule!

Early access to ROOT 6 was very helpful to CMS

CMS is further along with integrating a new ROOT version than ever before

Especially notable since is a major version change

All standard CMS jobs can now run using ROOT 6

Can begin full validation

Still a few remaining issues before CMS can full adopt ROOT 6

Performance

Thread-safety

Other small issues

ROOT 6 Performance

Reconstruction Job Timing

Reads and writes ROOT files
ROOT 5: 3.16 seconds/event with startup of job taking 80 seconds
ROOT 6: 3.44 seconds/event with startup of job taking 179 seconds
9% slower per event

Simulation Job Timing

Only writes ROOT file
ROOT 5: 2.81 seconds/event with startup of job taking 84 seconds
ROOT 6: 2.92 seconds/event with startup of job taking 172 seconds
4% slower per event

Jobs are CPU intensive should spend little time in I/O layer

ROOT 6 Remaining Issues

Memory usage

Much larger than ROOT 5

No per event memory increases seen

All extra memory seems to be at startup

Have been informed it will be fixed in 6.02

Checksums

Values of checksums not yet stable

Should be fixed in 6.00.02

Bug in checksum calculation for which CMS has a ugly work around

Non-class types

CMS has own patch of ROOT 6 to implement Reflex ability to deal with non-class types

In discussion with ROOT team on how to get needed functionality

Bug fix for missing class version for base class issue

Philippe already has fix not yet incorporated in 5.38

Short-Term Threading Support

Categories of thread support

thread-unsafe: API can only be called by one thread at a time
thread-usable: independent objects can be modified on different threads
thread-safe: same object can be modified by different threads simultaneously

Need thread-usable I/O in ROOT 5 and 6

Read different TFiles on different threads safely
Write different TFIles on different threads safely

Need thread-usable filling of histograms in ROOT 5 & 6

Fill different histograms on different threads safely

Long-Term Threading Support

Parallel storage to TFile

- Parallel serialization of branches
- Parallel compression
- Do not block a thread when writing out to file

Parallel reading from TFile

- Parallel reading of different branches from same event
- Parallel reading of same branch from different events

CPU intensive tasks must be done on threads controlled by CMS

- Reason: grid sites will tell application how many cores the job can use
- Fine if task based is a non-default option
- Task based decomposition to fit with TBB

Threads used to decrease I/O latency can be separate from CMS

- As long as they have very small CPU utilization over the job lifetime

Want to minimize resources used

- limit extra file handles
- limit extra memory buffers

Wish List

Native endian

Faster

Reduced utilization of mutexes