News from the ROOT Team

D. Piparo (CERN, EP-SFT) for the ROOT team

ROOT

Data Analysis Framework

https://root.cern

ROOT Releases

- v6.14/6 to be announced this week
- v6-16-00-patches branch created today
  - v6-16-00-rc1 tag available for testing by experiments
- We fixed these blockers

  ROOT-9743, ROOT-9757, ROOT-9666, ROOT-9668, ROOT-9686, ROOT-9719, ROOT-9725

- We are working on a few remaining blockers:

  ROOT-9637, ROOT-9660, ROOT-9668, ROOT-9709

Implicit Parallelism
Removal of Task Interleaving

Nested Parallelism, Work Stealing

- ROOT adopts a task-based parallelism approach
- Crucial to keep all cores busy all the time, reducing imbalance
- Two behaviours which help there:
  - Nested parallelism: a task can launch smaller tasks and wait for them.
  - Work stealing: idle workers randomly steal tasks from other workers' queues
- Both present in ROOT

The issue: “Task Interleaving”

- A thread is idle since the task it is running is waiting for all subtasks to finish, all sub-items of work were stolen.
- A new item of work is started within the very same thread, from the very function which was “paused” for waiting the subtasks

  This requires items of work to be re-entrant!

- Hard to implement and hard to explain to users

Re-entrant: “it can be interrupted in the middle of its execution and then safely be called again ("re-entered") before its previous invocations complete execution.”

Example 1: RDataFrame

- ROOT flushes baskets on disk in parallel
  - This happen through the submission of several compression-tasks
- ROOT reads branches in parallel
  - TTree::GetEntry
- RDataFrame parallelises the processing over event clusters
  - 1 task per cluster

It can happen that the processing of a range of events (a cluster) is blocked because of nested parallelism. The processing of a range of events can start while the processing of a different range of events is idle, *within the very same thread*.

See issue reported at Atlas ASG meeting:
https://indico.cern.ch/event/767992/

Example 2: CMS Simulation

- Again ROOT's parallel flush baskets
- A task in this context can contain the simulation of an entire event
  - even ~1 minute long

The simulation of a new event could be started therewith blocking the writing a bunch of other events

See issue reported at the ROOT IO Workshop by CMS:
https://indico.cern.ch/event/715802

The Solution: Work Isolation

- In a nutshell: forbid stealing tasks spawned by *parents*
- Nested parallelism and work stealing continue to work, with a more limited scope
- This implies (among other things) that:
  - In RDF, the processing of a range of events can never be interrupted by the processing of another range.
  - In CMSSW, no G4 task (actually no fwk task), can be stolen by ROOT's tasks
- Directives for the user:
  - Express parallelism with ROOT primitives is always OK
  - Nest TBB parallelism within ROOT parallelism is OK provided that TBB parallelism is *isolated*
- See https://software.intel.com/en-us/node/684814

Consequences on Performance/Scaling

- Work isolation will be in ROOT 6.16 and 6.14.08
- Removing task interleaving allowed to simplify ROOT
  - E.g. RDF, TTreeProcessorMT
- No performance penalty could be measured. On the contrary…
- … Parallel analysis of large Totem datasets improved runtime by 30%
- … Simple Atlas SUSY studies improved runtime by 10%