Condition prototype and beyond

Hadrien Grasland

LAL – Orsay

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

Prototype status

- Condition handling prototype delivered on time for MS41
- Performance benchmarks added, allow…
  - Detailed performance analysis
  - Testing of specific usage scenarios
  - Regression checking
- Prototype presented to the LHCb experiment[1]
  - Some interesting feedback and optimization suggestions
  - Overall, no major objection to this design
- Path towards DD4Hep integration was discussed

[1] https://indico.cern.ch/event/598284/contributions/2417507/attachments/1392777/2122282/Condition_Upgrade.pdf

Current prototype performance

Measurements on a Xeon E5-1620 v3 @ 3.50GHz:

	µs	ns	
Treuse + Tstart,0	5,4	5 400,0	→ Minimal event scheduling delay (when conditions are ready)
Tbuild,0 + Tload,0	12,3	12 300,0	→ Minimal condition slot creation delay (for 1 condition)
Talg-init,0	1,0	1 000,0	→ Minimal ConditionAlg scheduler startup delay
Tstart,1	0,0	-0,1	→ Extra event scheduling delay per condition (**negligible**)
Tbuild,1+Twrite	0.3	332.0	→ Condition creation delay
Tread	0,0	9,8	→ Condition readout delay
Talg,init,1	0.1	71.8	→ ConditionAlg scheduling delay

Analysis:

- Small event scheduling overhead when conditions do not change (couple of µs, comparable to scheduling a TBB task!)
- Condition reads are extremely cheap (overhead is barely measurable)
- ConditionSlot creation and condition writes can be more expensive
  - Not a concern for LHCb, impact must be evaluated for ATLAS

Next steps for the prototype

- Performance evaluation for ATLAS use cases
  - Need some orders of magnitude: Amount of conditions? HLT input rate? Amount of HLT nodes?
  - To be discussed with Andrea Formica, Walter Lampl...
- Need to plan for Gaudi integration
  - Salient issue: (lack of) asynchronous IO in Gaudi

Asynchronous IO

- Basic principle: CPUs should not wait for IO devices
  - Start an IO operation, synchronize only when needed
  - Powerful interface for this: C++ Concurrency TS futures (as implemented by Boost.Thread, HPX, Just::Thread…)
  - Many possibilities: blocking, polling, continuations…
- Diverging opinions on async IO integration into Gaudi
  - Blocking Algorithms modeling IO *operations*?
  - Asynchronous Services modeling IO *resources*?
- Major consequences on condition handling, IO strategy should be decided before prototype is integrated

Questions? Comments?

Condition prototype @ https://gitlab.cern.ch/hgraslan/conditions-prototype