Using TBB in ATLAS Liquid Argon Calibration Code

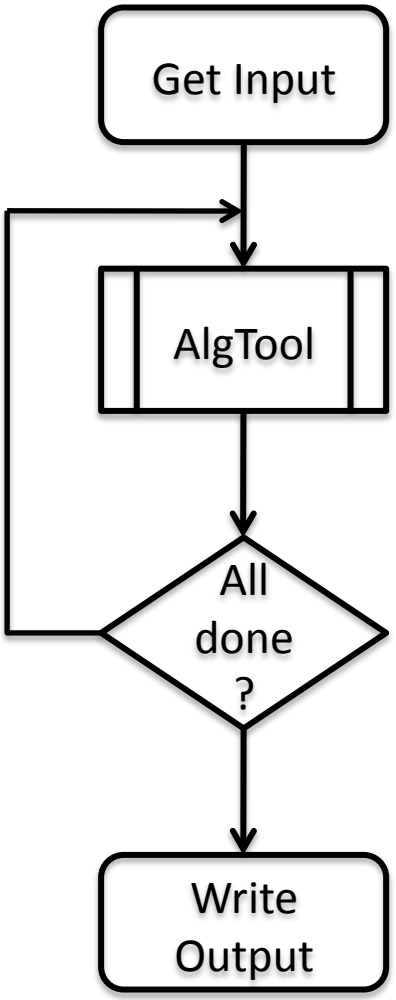
Walter Lampl
University of Arizona

Setting the scene

- Test/Example case: RTM parameter extraction
  - Fitting of electrical parameters using the Calibration Waveform
  - Needed to get Optimal Filtering weights used in the LAr ADC -> MeV step
- Started looking at this job because:
  - It is quite slow. Typically 3-4 CPU hours for a half-barrel in one gain
    - Complete calibration dataset for the EM calorimeter is sub-divided into 2 half-barrels, 2 end-caps and three gains => 12 jobs in total
  - It should be very suitable for Autovectorization because it handles large arrays. A LArWave is basically a vector<double> of size 768
    - Got ~40% speed-up
  - It should be also suitable for multithreading because cells are treated independently
- To avoid any misunderstanding: This code **does not run in reconstruction.** It runs on the calibration farm to produce constants used in reconstruction
  - Nevertheless based on the athena framework
  - Calibration taken ~ 1/week, update as needed ~ 1/month

Workflow: Algorithm & Tool

- Algorithm (*LArRTMParamsExtractor*) retrieves input container(s) (*LArCaliWaveContainer*)
- Loops over input container(s), gain, channel, DAC
- Calls AlgTool (*LArWFParamsTool::getParams*) on each CaliWave
  - This tool does the computational weight-lifting
  - **Each CaliWave is treated independently!**
- Fills output containers (*LArCaliPulseParamsComplete and LArDetCellParamsComplete*)
- **Note**: The algo has no event loop, all work is done in stop()
- **The problem:** LArWFParamsTool internally calls many private methods and passes data between them via member variable
  - Not thread-safe
  - Program flow obfuscated, calling methods in wrong order leads to garbage results



Steps done for multi-threading

- Make LArWaveParamsTool::getParams thread safe
  - Pass input/output data as arguments/return values
    - Mostly as const-reference to a struct holding many values
    - Relying on Return-Value-Optimization for bulky returned objects
  - The getParams method is now const (w/o cheating!)
- Create a class-in-a-class functional inside of LArWaveParamsTool
- Prepare a vector<struct> holding pointers to input and output objects
- Invoke LArWaveParamsTool::getParams once in serial mode to make sure all caches are filled
- Let tbb::parallel_for work on the vector<struct>

Worked out of the box!

Test Results

- Running on an ~empty lxplus6 node (Intel Xeon 2.27 GHz)
- `athena.py LArtauR_Auto_00214354_00214355_00214356_EB-EMBA.py`
  `17257.32s user 5.59s system 1448% cpu 19:51.64 total`
- Comparison: Same job on my slc5 desktop (Intel core duo 3.16 GHz) with athena 17.0.2.10:
  - `athena.py LArtauR_Auto_00214354_00214355_00214356_EB-EMBA.py`
    `13249.53s user 15.22s system 99% cpu 3:41:41.32 total`
- Output is identical
- Interesting detail: My job did not get killed by a lxplus CPU time limit

Playing a bit more with TBB components

- Add tbb::atomic<unsigned> as error counter
- About caches and lazy initialization:
  - The LAr cabling map is loaded from the conditions database on the first invocation
    - Common design-pattern use by many LAr/Calo condition-tools
  - That means in this case: Inside the threaded part
  - Without an explicit call to the cabling in the serial part prior to parallel_for the job crashes in an ugly way
- Adding a Mutex-lock during loading of cabling map solves this issue
  - Though I am a bit unsure about the bool m_init
    - Should it be atomic … or is this an overkill?
  - Less efficient than explicit initialization: The first thread talks to the database, all others have to wait

Conclusion/Summary

- Using TBB in athena is easier than I thought
  - All attempts to use TBB components worked out-of-the-box
- But this is not standard reconstruction. I know the involved code very well (no black boxes)
- I understand that for a multithreaded application the fact that it ran once doesn't mean it is error-free

Link to the code:

https://svnweb.cern.ch/trac/atlasusr/browser/wlampl/athena/TBB/LArCalorimeter/LArCalibUtils/LArCalibUtils/LArRTMParamExtractorTBB.h

https://svnweb.cern.ch/trac/atlasusr/browser/wlampl/athena/TBB/LArCalorimeter/LArCalibUtils/src/LArRTMParamExtractorTBB.cxx