Gaudi TBB migration proposal

B. Wynne

07/04/2021

Gaudi TBB migration

tbb::task_scheduler_init
- Used to set the overall size of the thread pool for an MT job
- tbb::global_control sets a maximum, but does not override (apparent) internal limit of N_threads <= N_cores

tbb::task
- Each algorithm's execute() method wrapped in a task class for parallel execution, then enqueued

Both tbb::task_scheduler_init and tbb::task are marked deprecated

tbb::task_arena can replace both components, and is supported in oneAPI

Gaudi ThreadPoolSvc will own a single tbb::task_arena, with number of threads set on construction

Algorithms executed as std::functions passed to the arena

Extensive use of TBB data structures (e.g. tbb::concurrent_queue) is unaffected

Quirks and future developments

Gaudi ThreadPoolSvc finalises all threads (i.e. TLS) by using a boost::barrier
- Relies on knowing exactly how many threads were used
- Charles reported TBB sometimes using extra threads and idling others: slides
- In my limited testing, did observe threads used > threads requested in some jobs, implying some threads not finalised
- Minor issue: sometimes fewer threads are used than requested, until finalise

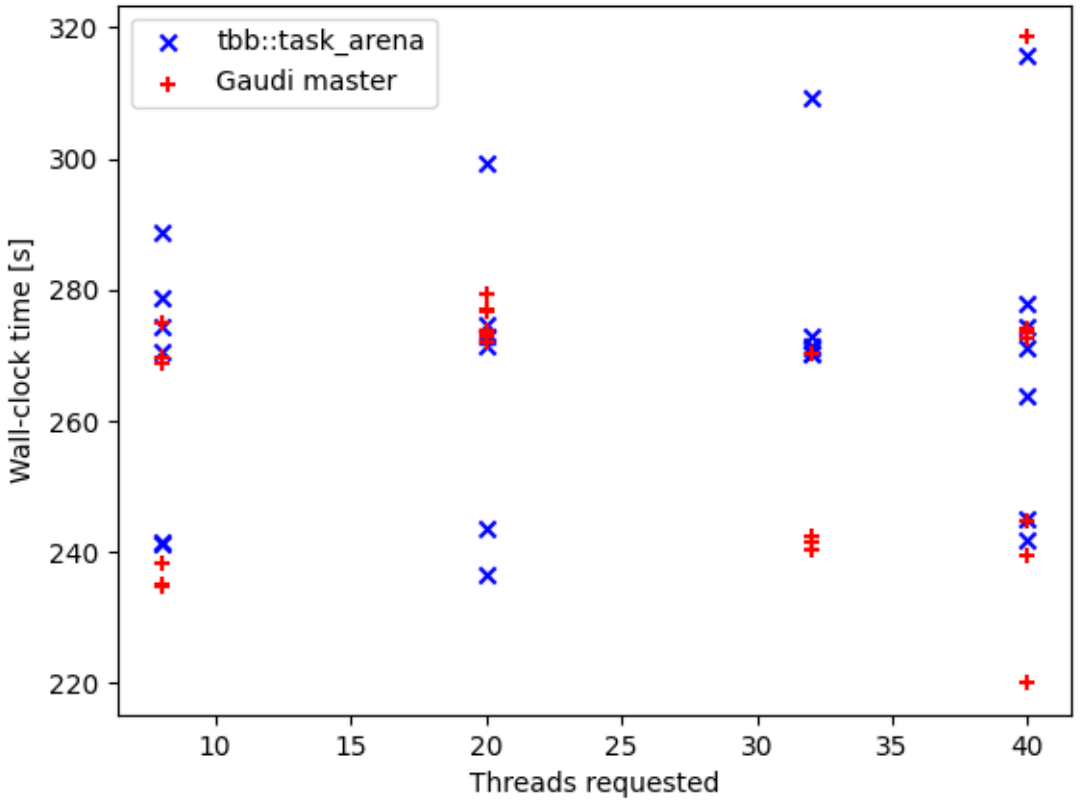
CMS makes extensive use of tbb::task_group
- At the top level there's still a single arena to set thread number
- Task groups used for smarter recycling of threads within a single event for CPU cache reuse (ask Chris), potentially this could be added to Gaudi
- tbb::task_arena allows us to oversubscribe CPU, but do we actually need this?

Gaudi is intended to support internal multithreading in algorithms
- In practise this is not used widely / at all
- Testing with tbb::parallel_for suggests that it behaves as expected within the new task arena, does not require a public Gaudi interface to the arena
- Another potential use for task groups?

We use std::threads in some places as well…
- Gaudi scheduler
- Message service (logging)

Performance

Gaudi master and the proposed tbb::task_arena MR have similar performance on a less-than-perfect test machine
- were some other people using it, and I think it limits jobs to 8 of 32 cores
- basically this just needs better testing

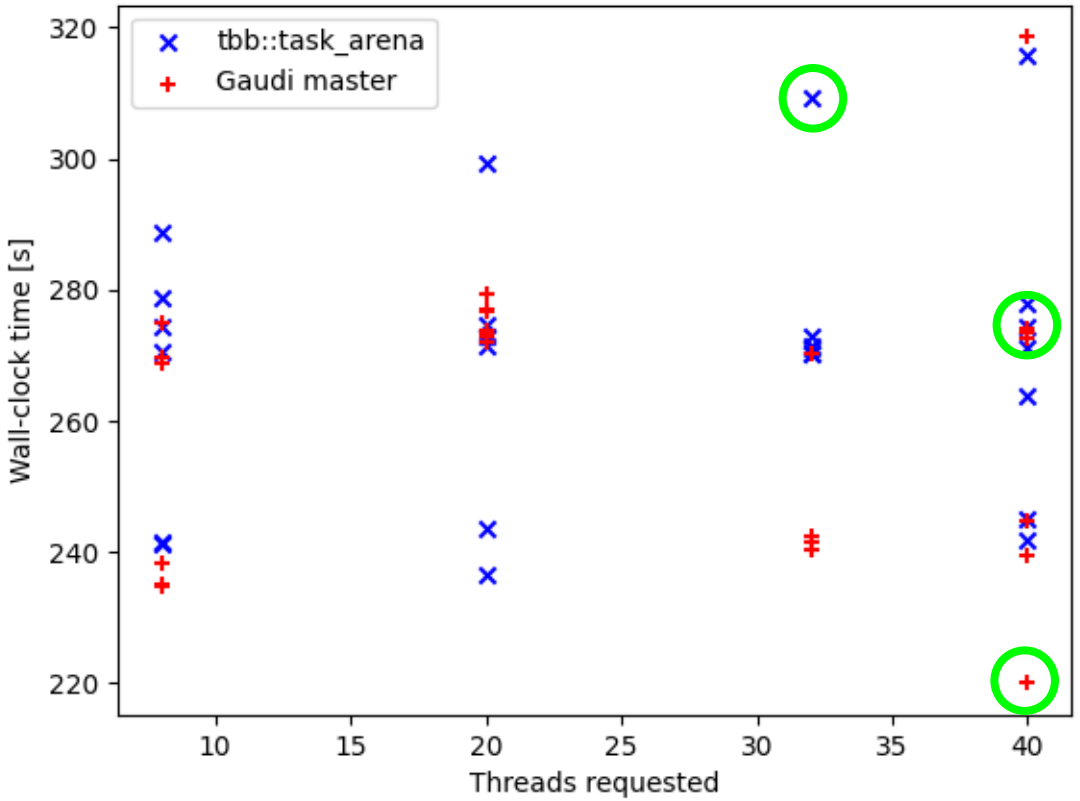


Arena mean 270 sec, std dev 20
Master mean 261 sec, std dev 21

AtlasMCRecoScenario test, 100 events, N slots == N threads

Performance

Gaudi master and the proposed tbb::task_arena MR have similar performance on a less-than-perfect test machine
- were some other people using it, and I think it limits jobs to 8 of 32 cores
- basically this just needs better testing



Three jobs in this set used more threads than requested, but there's no clear pattern – occurred in both code branches, faster and slower jobs