
# Gaudi: migrating from TBB Task API

## ATLAS viewpoint

Illya Shapoval
Lawrence Berkeley National Laboratory

HSF Software Project Discussion on Multithreading, 6 May 2020

# Instances of deprecated TBB Task API usage in Gaudi

| | Deprecated TBB Task API features | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | task class | task allocation | explicit task destruction | task recycling | synchro-nization | task context | task affinity | empty task | task list | priorities |
| **AvalancheSchedulerSvc** [ATLAS, ...] | | | | | | | | | | |
| **ThreadPoolSvc** [ATLAS, ...] | | | | | | | | | | |
| **SerialTaskQueue** [?] | | | | | | | | | | |

**Common runtime pattern of usage:**

1. Allocate memory for a task object of class *T* derived from the *tbb::task* class
2. Schedule the task with *tbb::task::enqueue(T&)*

# Migrating to *tbb::task_group*

- *tbb::task_group* - “the higher among the lower” abstraction
  - required non-blocking semantics provided by *tbb::task_group::run(Func&&)*
- Advantages and disadvantages:
  - functors and lambda expressions as tasks, instead of deriving from *tbb::task*
  - cancelation of spawned tasks
  - more compact code
  - *~tbb::task_group()* can throw and thus violate the noexcept contract of Gaudi components
  - *tbb::task_group::run* may put more strain on task stealing mode (scalability concerns?)

# Migrating to *tbb::task_group*

- *tbb::task_group* - “the higher among the lower” abstraction
  - required non-blocking semantics provided by *tbb::task_group::run(Func&&)*
- Advantages and disadvantages:
  - functors and lambda expressions as tasks, instead of deriving from *tbb::task*
  - cancelation of spawned tasks
  - more compact code
  - *~tbb::task_group()* can throw and thus violate the noexcept contract of Gaudi components
  - *tbb::task_group::run* may put more strain on task stealing mode (scalability concerns?)
- The approach already implemented in MR 1067 (note changes stat: +114, -226)
  - observing 1.5% improvement in scheduling overhead at 18 threads in one of the ATLAS scenarios

# Migrating to *tbb::task_group*

- *tbb::task_group* - “the higher among the lower” abstraction
  - required non-blocking semantics provided by *tbb::task_group::run(Func&&)*
- Advantages and disadvantages:
  - functors and lambda expressions as tasks, instead of deriving from *tbb::task*
  - cancelation of spawned tasks
  - more compact code
  - *~tbb::task_group()* can throw and thus violate the noexcept contract of Gaudi components
  - *tbb::task_group::run* may put more strain on task stealing mode (scalability concerns?)
- The approach already implemented in MR 1067 (note changes stat: +114, -226)
  - observing 1.5% improvement in scheduling overhead at 18 threads in one of the ATLAS scenarios

But, we are studying more advanced runtime systems:

- see the proposal for Gaudi explaining advantages of HPX.