P0433 Executors Issues Needing Resolution | P2219R0

Jared Hoberock

2020-09-15

Review of the unified executors proposal (P0443) has generated issues that need resolution. Further review will generate more.

Rather than resolve these in committee, let's use Github:

- Those with a strong opinion on the resolution of an issue should volunteer to submit a pull request and signal their commitment by assigning themselves to the issue on Github
- At least one principle author will review pull requests before merge
- Unassigned issues will eventually be resolved by a principle author

Issues Needing Resolution

1. Suggested: Explicitly specify the level of abstraction of the library
2. Suggested: Add a (Short!) rationale for design choices
3. Suggested: Add an explicit terms and design section
4. Consider moving 'Polymorphic wrapper' into a separate proposal
5. Define minimal API for Properties of any_executor (for example: prefer_only)
6. Align execution agent's definition with the one in the standard
7. Consider having the description of behavioral properties in one place use references to that
8. Describe when and how submit/execute/other similar methods use allocator
9. Investigate the possibility to simplify allocator_t property
10. Consider consolidation of schedulers/senders type requirements on the P0443 level
11. Investigate whether vendors can add their own properties via free functions but not member functions.
12. Introduce the "behavioral properties" term of art
13. Introduce the "Properties" term of art in P1393
14. Consider alternative naming for outstanding_work_t property and nested properties for that
15. Consider alternative naming for relationship_t property and nested properties for that
16. Provide stronger guarantees on outstanding_work.untracked
17. Investigate the right default for relationship_t property
18. Consider the third option for relationship_t property
19. Investigate the right default for outstanding_work_t property
20. Consider third option for outstanding_work_t property
21. Make sure that P0443 clearly states that receiver functions cannot be called until start is called
22. Investigate the necessity of operator== for static_thread_pool senders and for senders in general
23. Consider whether to add a submit member function to the static_thread_pool sender type.
24. Consider alternative name instead of schedule()
25. Remove 2.5.3.5
26. Consider adding is_always_equal trait for executors/schedulers/senders
27. Explore is operator!= should be explicitly listed
28. 2.5.3 Schedule() method should be in the synopsis
29. Change descriptive type C to something more specific
30. State the relation between the scheduler_type and executor_type
31. Scheduler of static_thread_pool should have the same set of properties as sender does
32. Consider adding the method that indicates that static_thread_pool is stopped
33. Add wording how 2.5.2.3 "Worker management" functions interact with each other

34. 2.5.2.3 Add attach_and_wait() method
35. Provide and example for untracked executor with no more work expected
36. Investigate why the wording has been changed and identify the right one
37. Consider better name for static_thread_pool
38. Define better term instead of function objects (work-item, task, etc.)
39. Should properties be usable with non-executors?
40. Non-movable properties
41. any_executor<P...>::target causes unused RTTI overhead
42. bulk_guarantee needs to be better specified
43. What does "established property" mean?
44. Why does executor-of-impl<E,F> require invocable<F&> and not simply invocable?
45. Review wording of any_executor member functions that require a valid target
46. Consider making `receiver_of<T,void>` synonymous with `receiver_of<T>`
47. new_thread_t needs stronger language
48. Replace use of "vectorized" with current language
49. Permit SIMD executors with floating point implications
50. Define what blocking means in the context of senders and receivers
51. Consider positioning `value_types`'s `Variant` parameter consistently with `error_types`'s `Variant` parameter
52. `sender_traits` specializations may instantiate `Tuple` and `Variant` templates with `void`
53. Questions about execution::execute and exceptions
54. Require the executor/scheduler concepts to be swappable
55. The scheduler/executor concepts have inconsistent requirements for copyability/movability
56. Don't require that sender operations do not introduce data races
57. Can execution::start() be made a member function instead of an ADL-based customisation-point?
58. Consider removing as-sender helper adapter from execution::schedule wording
59. Nothing in scheduler concept that requires the sender returned from schedule to be a void-sender
60. Resolve whether the basis-operation CPs should require noexceptness or be transparent to noexceptness and have the requirement enforced by the concept
61. as-operation::start() can potentially call set_error() on an already-completed receiver
62. Should set_value/set_error/set_done methods in as-receiver be rvalue-qualified
63. Harmonize `static_thread_pool::executor_type::bulk_execute` with `execution::bulk_execute`'s requirements