CRYSTAL BALL: Checking Predictions Against the Record

Document Number:	P4047R0
Date:	2026-05-01
Intent:	Inform
Audience:	WG21

Reply-to:	Vinnie Falco vinnie.falco@gmail.com

Table of Contents

Abstract

This paper collects dated, public, falsifiable predictions about `std::execution` from proponents and critics alike, and checks each against the record.

Revision History

R0: May 2026 (pre-Brno mailing)

- Initial revision.

---

1. Disclosure

The author provides information and serves at the pleasure of the committee.

This paper examines the published record. That effort requires re-examining consequential papers, including papers written by people the author respects.

The author has papers before the committee proposing coroutine-based I/O (P4003R3) and analyzing `std::execution` (P2583R4, P4007R3, P4014R2). Every prediction and outcome cited below is sourced to public committee records, published blog posts, or recorded conference talks.

The author is the intelligence of record. P4023R0 [1] (Directions Group, "Strategic Direction for AI in C++") establishes that "the ultimate responsibility for accuracy, logic, and normative quality rests entirely with the human author." This paper follows that principle. AI tools assist with research, compilation, and drafting. The author curates, verifies, and takes responsibility for every claim.

This paper uses AI at every stage. Prediction discovery, outcome verification, scorecard organization, and paper drafting were all AI-assisted. Every stage involves machine output. P4023R0 identifies research, summarization, and consistency checking as permitted uses of AI in the committee process [1]. This paper's use of AI falls within that scope.

Human curation is required at every stage. AI can misidentify predictions, misclassify outcomes, or miss relevant public statements. The author verified every prediction-outcome pair against the cited source.

Outcome classifications reflect the author's assessment of the cited evidence. The scorecard is one systematic reading of the public record. Each entry cites its source. Readers who disagree with a classification can verify against the cited source and reach their own conclusion.

This paper asks for nothing.

2. Method

A prediction qualifies if it is (a) dated, (b) public, and (c) falsifiable - a claim about a future outcome that can be checked. Outcomes are graded: **Confirmed**, **Unconfirmed**, **Partially Confirmed**, or **Overtaken by Events**. Both sides miss. Both sides hit. The table includes all of them.

3. The Predictions

Timeline

#	Prediction	Source	Date	Outcome
T1	P2300 unlikely for C++23 given size of the proposal	Adelstein Lelbach, CppCon 2021 [2]	2021	Confirmed
T2	"merge most of the foundational bits [P2300] early in the C++26 cycle"	Voutilainen P0592R5 [3]	2022	Confirmed
T3	"not going to be in a timeframe conducive to inclusion in C++23" (re: language changes P2300 needs)	P2469R0 [4]	2021-10	Confirmed
T4	Networking TS should ship for C++23	P2469R0 [4]	2021-10	Unconfirmed - neither Net TS nor any networking shipped
T5	Stop Networking TS for C++23, focus on P2300	Voutilainen P2464R0 [5]	2021	Confirmed - Net TS stopped; P2300 proceeded

Safety and Correctness

#	Prediction	Source	Date	Outcome
S1	Stack overflow from reentrant completion without executors is "user-hostile default behaviour"	P2469R0 [4]	2021-10	Confirmed - Müller P3801R0 [6] (2025), Kühl P3796R0 [7]
S2	P2300 design "insufficiently paranoid to be suited to internet-facing software development"	P2480R0 [8]	2021-10	Partially confirmed - networking deferred to C++29
S3	`set_error` channel confusion - scheduling errors vs. I/O errors	P2469R0 [4]	2021-10	Confirmed - still unresolved
S4	"P2300 will likely require language changes to work effectively" - tail calls, trampolines	P2469R0 [4]	2021-10	Confirmed - P2583R4 [9] documents symmetric transfer gap (2026)
S5	Tail call concerns documented before P2300	Nishanov P1362R0 [10]	2019	Confirmed - still open

Customization Mechanism

#	Prediction	Source	Date	Outcome
C1	`tag_invoke` as the customization mechanism for sender algorithms	P2300R0-R5, P2403R0 [11]	2020-2022	Unconfirmed - abandoned; replaced by `transform_sender` (P2999R3 [12])
C2	`transform_sender` as the customization mechanism	P2999R3 [12]	2024	Unconfirmed - P3303R1 [13] "Fixing," P3718R0 [14] "Fixing Again," then P3826R5 [15] proposes removal
C3	"the customization mechanism has seen a fair bit of recent churn" - remove for C++26, restore in C++29	Niebler P3826R0 [15]	2025-10	Pending - proposed

Universality and Scope

#	Prediction	Source	Date	Outcome
U1	">90% of all async code in the future should be coroutines"	Niebler, "Structured Concurrency" [16]	2020-11	Shifted - by 2024: "returning a sender is a great choice" [17]
U2	"senders are like iterators when STL emerged" - universal adoption predicted	Kühl, quoted in P2300R3-R10	2021-25	Unconfirmed - no equivalent adoption curve
U3	SG1 "no consensus" that one grand unified model is needed (4-9-5-5-1)	P2453R0 [18]	2021-09	Confirmed - poll happened; direction proceeded regardless
U4	"Networking should be based on sender/receiver model" - weak consensus (17-11-10-4-6)	P2453R0 [18]	2021-10	Unconfirmed - no sender-based networking exists; deferred to C++29
U5	Production library chose awaitables over senders for usability	Prokoptsev, CppCon 2024 [19]	2024	Confirmed
U6	"C++26 will *still* not have a standard coroutine task type"	Niebler P3826R0 [15]	2025-10	Pending

Domain Viability

#	Prediction	Source	Date	Outcome
D1	Sender/receiver deployed in shipping Facebook products, "monthly users number in the billions"	Niebler P2470R0 [20]	2021	Confirmed
D2	NVIDIA "fully invested in P2300... we plan to ship in production sometime next year"	NVIDIA statement in P2470R0 [20]	2021	Confirmed - nvexec shipped
D3	Bloomberg experimentation with sender/receiver	P2470R0 [20]	2021	Confirmed

Networking

#	Prediction	Source	Date	Outcome
N1	"The C++ Standard can't access the Internet. This is embarrassing and long overdue."	P2469R0 [4]	2021-10	Confirmed - still true
N2	"We don't have a composable approach proposal [for networking] at this point"	Voutilainen P0592R5 [3]	2022	Confirmed - still true
N3	Chair's note: "In the short term, this poll result doesn't mean much. We don't have a paper in hand that proposes networking based on the P2300 model."	P2453R0 [18]	2021-10	Confirmed - still no such paper

Implementation Maturity

#	Prediction	Source	Date	Outcome
M1	"many algorithms clearly labeled 'Not yet implemented' in libunifex"	P2469R0 [4]	2021-10	Confirmed - libunifex archived; 50+ post-approval items in C++26 WD [21]
M2	P2300 task type concerns: stack overflow, allocator, error handling	Müller P3801R0 [6], Kühl P3796R1 [7]	2025	Confirmed - 16 open issues documented by task author

4. Scorecard

Category	Predictions	Confirmed	Unconfirmed	Shifted / Partial	Pending
Timeline	5	4	1	0	0
Safety / Correctness	5	4	0	1	0
Customization Mechanism	3	0	2	0	1
Universality / Scope	6	2	2	1	1
Domain Viability	3	3	0	0	0
Networking	3	3	0	0	0
Implementation Maturity	2	2	0	0	0
Total	**27**	**18**	**5**	**2**	**2**

5. Conclusion

Predictive	Not Predictive
Safety and correctness concerns	Universality claims
Domain viability (where deployed)	Customization mechanism design
Networking gap	Timeline estimates (all sides)
Implementation maturity concerns	

References

[1] Michael Garland, Andrew McKenney, Roger Orr, Bjarne Stroustrup, Daveed Vandevoorde, Michael Wong. P4023R0. "Strategic Direction for AI in C++: Governance, and Ecosystem" (Directions Group, 2026).

[2] Bryce Adelstein Lelbach. "C++ Standard Parallelism." CppCon 2021

[3] Ville Voutilainen. P0592R5. "To boldly suggest an overall plan for C++26." WG21

[4] Christopher Kohlhoff, Jamie Allsop, Vinnie Falco, Richard Hodges, Klemens Morgenstern. P2469R0. "Response to P2464: The Networking TS is baked, P2300 Sender/Receiver is not." WG21, 2021

[5] Ville Voutilainen. P2464R0. "Ruminations on networking and executors." WG21, 2021

[6] Jonathan Müller. P3801R0. "Concerns about the design of std::execution::task." WG21, 2025

[7] Dietmar Kühl. P3796R1. "Coroutine Task Issues." WG21, 2025

[8] Christopher Kohlhoff, Jamie Allsop, Klemens Morgenstern. P2480R0. "Response to P2471: "NetTS, Asio, and Sender library design comparison" - corrected and expanded" WG21, 2021

[9] Mungo Gill, Vinnie Falco. P2583R4. WG21, 2026

[10] Gor Nishanov. P1362R0. "Incremental Approach: Coroutine TS + Core Coroutines." WG21, 2019

[11] Michael Garland, et al. P2403R0. "Presentation on P2300 - std::execution." WG21

[12] Eric Niebler. P2999R3. "Sender Algorithm Customization." WG21, 2024

[13] Eric Niebler. P3303R1. "Fixing Lazy Sender Algorithm Customization." WG21, 2024

[14] Eric Niebler. P3718R0. "Fixing Lazy Sender Algorithm Customization, Again." WG21, 2025

[15] Eric Niebler. P3826R5. "Fix Sender Algorithm Customization" WG21, 2025

[16] Eric Niebler. "Structured Concurrency." 2020

[17] Eric Niebler. "What are Senders Good For, Anyway?" 2024

[18] Bryce Adelstein Lelbach. P2453R0. "2021 October Library Evolution Poll Outcomes." WG21, 2022

[19] Dmitry Prokoptsev. "C++ Coroutines and Structured Concurrency in Practice." CppCon 2024

[20] Eric Niebler. P2470R0. "Slides for presentation of P2300R2: std::execution (sender/receiver)" WG21, 2021

[21] LWG unresolved prioritized list

[22] P2300R10 - "`std::execution`" (Eric Niebler, Michał Dominiak, Georgy Evtushenko, Lewis Baker, Lucian Radu Teodorescu, Lee Howes, Kirk Shoop, Michael Garland, Bryce Adelstein Lelbach, 2024).