std::core_dump()

The Newsletter of the C++ Alliance

Issue 1
January 2026

In this Issue

New Faces at C++ Alliance

Steve Gerbino
Working on Beast2
Background: Hands-on CTO with clang/LLVM build/CI expertise
Previous: CTO at Koinos/Respublica, co-founder of Coinos Group

Michael Vandeberg
Working on Beast2
Background: 10 years of C++ experience, extensive Boost usage
Previous: Sr. Engineer at Steemit, co-founder of Coinos Group

Jeremy Childers
Django Developer
Background: Django, Wagtail and GraphQL experience
Previous: Open source background with C++ and Boost experience

Vlad Serebrennikov
Working on our Clang Compiler initiative
Previous: Clang Maintainer at Bloomberg
Background: Deep compiler expertise, organized 7,000+ Clang bug tracker issues

The C++ Alliance is committed to improving the C++ ecosystem, and this month we're putting real resources into strengthening the Clang front-end. The issue we're addressing is fundamental: compiler front-ends are notoriously difficult to work on, and there's a critical shortage of engineers with front-end expertise. This creates bottlenecks across the entire C++ ecosystem - slower bug fixes, delayed feature adoption, and diagnostic quality that lags behind what developers need. By investing in Clang's front-end, through both talent development and improved workflows, we're working to break these bottlenecks and deliver tangible improvements in compile times, diagnostics, and standard feature support that benefit every C++ developer in the Clang ecosystem.

We're starting with Clang (rather than GCC) because it gives us the best place to prove and refine a repeatable workflow for delivering high-quality, upstream-ready work: we can iterate quickly in one public compiler ecosystem, measure visible outcomes (throughput and adoption), and then share what we learn more broadly, including with the GCC community.

With Vassil Vassilev, we're launching a Clang front-end talent pipeline in partnership with his Compiler Research Group: a structured, mentored program focused specifically on front-end work. The goal is to grow more capable compiler contributors and help address the broader shortage of front-end engineers, while producing practical improvements that can move upstream.

We're also launching, with Krystian Stasiowski as our chief scientist, a *human-agentic Clang initiative*: a GitHub-native development and review workflow, backed by fast build/CI infrastructure, where AI can help draft analysis, patches, tests, and revisions - but human compiler experts remain the decision-makers and only polished work is submitted upstream. In parallel, we're doing targeted "knowledge capture" from experienced Clang implementers so hard-won front-end expertise directly improves our execution. To help with this effort, we've added Vlad Serebrennikov to our team. Please join us in #cppa-clang!

Introducing Beast2: The Future of C++ Networking

by Vinnie Falco

After years of listening to developer feedback and building on the hard-won lessons of Boost.Asio, we're excited to share progress on Beast2 - a complete reimagining of C++ networking for the coroutine era.

Starting From What Developers Actually Write
Peter Dimov set the design direction with a simple question: what should networking code actually look like? The answer drives every architectural decision in Beast2:

```
auto [ec, n] = co_await sock.read_some(buf);
```

One line. No platform headers bleeding through. No completion handler boilerplate. Just coroutines doing what coroutines do best.

This isn't about abandoning what made Boost.Asio great, it's about distilling 25 years of field experience into something cleaner. Buffer sequences and executors remain concept-driven and extensible. What changes is the surface area developers interact with daily.

A Family of Focused Libraries
Beast2 is a family of composable components, each designed for direct use:

Capy
provides the execution foundation optimized for async I/O, with task<T>, thread_pool, strand, and coro_lock. Tasks automatically propagate executor context, guaranteeing correct affinity without manual bookkeeping. This library alone represents what we believe belongs in the C++ Standard Library.

Corosio
delivers coroutine-first portable networking where every operation returns an awaitable. It includes SSL stream wrappers for both WolfSSL and OpenSSL, with streams that are non-templated for ABI stability.

Http
implements sans-I/O HTTP/1.1 with strict RFC compliance, Express.js-style routing, automatic compression, and extensive middleware including multipart/form processing and cookie management.

Beast2 brings it together as a high-level HTTP and WebSocket server with multithreaded execution, while Boost.Burl provides a high-level HTTP client with coroutine ergonomics.

Technical Wins That Matter
ABI Stability: In Boost.Asio, changing SSL implementations means recompiling everything. Beast2's stream algorithms see io_stream& rather than templated types, link against WolfSSL, then relink against OpenSSL with no recompilation required.

Zero-Allocation Steady State: Performance-critical servers can't afford allocations in hot paths. Beast2's capy::task achieves this through coroutine frame reuse. The key insight: the coroutine frame allocation we can't avoid pays for all the type erasure we need.
No Configuration Macros: Want thread-safe contexts? Use io_context. Need maximum single-threaded performance? Use unsafe_io_context. One library, one object file, runtime configuration.

High-Level by Default
Unlike its predecessor, Beast2 is full-featured from the start:

```
srv.wwwroot.use("/", serve_static("C:\\Users\\Vinnie\\my-website"));
```

Express.js patterns meet C++ performance, with proper multithreaded execution built in.
The libraries are under active development and we welcome community feedback as the APIs stabilize.

New Boost Website Features

by Rob Beeston

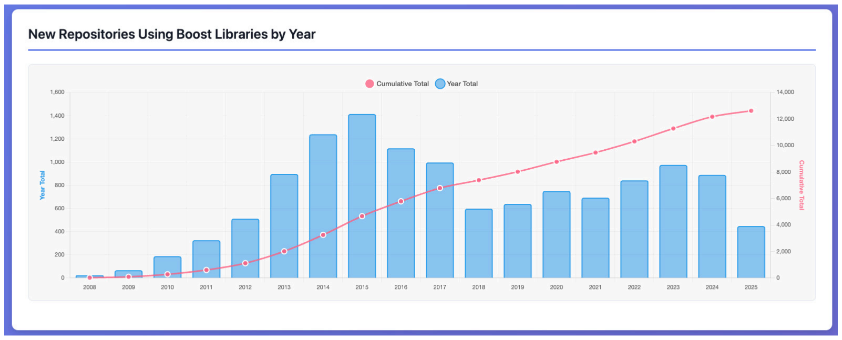
You may not know this, but the Boost website is already undergoing its next transformation. While the relaunch last year improved the legacy site, it was really just a step toward something much bigger. To move that vision forward, we've partnered with Metalab, a top-tier design firm known for thoughtful, developer-focused experiences.

Over the past few weeks, they've worked closely with us to shape a clean, modern, and community-centered design. We're now entering the engineering phase, where this vision will be built into the current site. The update introduces richer user profiles with space for your bio, achievements, and social links; a more robust publishing system for Boost news, events, and learning guides; and clearer pathways for discovering Boost and getting involved. Alongside new features like enhanced profiles and contribution badging, this is all about making Boost easier to explore, participate in, and grow with.

Watch this space...There's much more to come.

New Boost Library Usage Dashboard

by Will Pak



We have a versioned dashboard in development that tracks Boost usage across GitHub and lets you drill down from ecosystem trends to individual libraries and their external consumers.

Why it matters:

- Helps maintainers and leadership prioritize by showing which libraries are heavily used and which appear to be trending up/down over time.
- Adds context for "is Boost growing?" by looking at Boost's usage rate within established C++ repos (10+ stars), not just raw repo counts.
- Makes it easy to point to concrete examples via top-repo lists (by stars and by Boost usage intensity).

What we have done so far:

- Adoption over time (2008–2026), plus Boost usage rate within the quality-filtered C++ set.
- A sortable overview across 171+ libraries (repo count, total includes, recent vs. historical usage, and an activity score).
- Per-library pages with external consumer lists and internal dependency relationships.

What's next:

- Per-header stats (which specific headers/APIs drive usage).
- Better accounting for transitive usage (Boost pulled in indirectly).
- A lightweight, repeatable, period-based release report generator built on the same data.

Jan website traffic

102k
uniques

Total repositories using Boost libraries
(repos with 10+ stars)

13,733

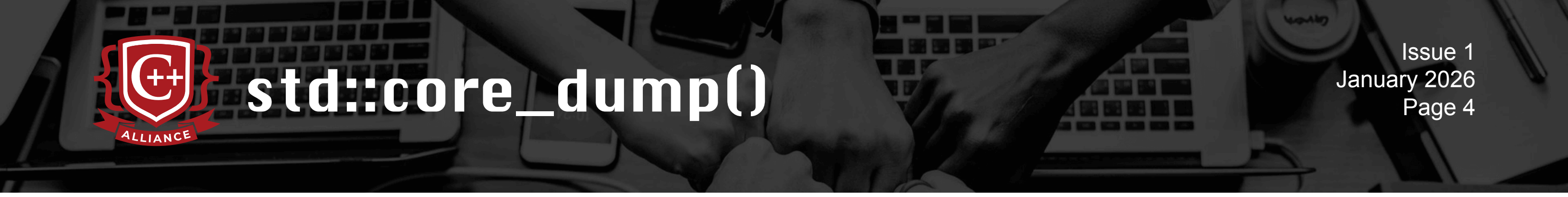
X followers

1,954

New Boost Testimonial Partners

https://hpx.stellar-group.org/

https://fil-c.org/



The C++ Conference

Major Plans for CppCon 2026

by Mark Cooper

The C++ Alliance is heading to CppCon 2026 with our most ambitious presence yet. This year, we're expanding our commitment to the C++ community with an enhanced sponsorship package, exciting announcements, and plenty of opportunities to connect with fellow developers.

The Boost Lounge
In addition to being a Gold sponsor, we're running the Boost Lounge. An open area featuring comfortable sofas and chairs, a few old school arcade games, complimentary barista service, and free freshly popped popcorn.

The lounge will be our home base for the week. A place where people can unwind, catch up and exchange ideas.

Capturing Developer Stories
In the lounge, we'll also be conducting developer interviews, gathering testimonials and creating video content that showcases the experiences of those who contribute to Boost as well as the C++ ecosystem overall.

If you have a story to tell, please contact me and we'll schedule you in.

C++ Documentary Premiere
We'll be hosting the premiere of a new documentary showcasing the C++ ecosystem.

This is a passion project The C++ Alliance has been developing over the past year, interviewing notable figures in the community, and providing a record of one of the most impactful technologies of our time.

Boost Update Session
Harry Bott will lead a session covering the latest developments from the Boost community. Helping the community understand what Boost has built since last year, and what's to come.

Conference T-Shirts
Once again, Rob Beeston will be designing the official conference t-shirts. Can't wait to see what he comes up with this year!

Call for Presenters
Consider doing a talk at CppCon 2026! We'd be happy to help you submit a proposal. Whether it's a technical deep-dive, project update, or lessons learned - it's a great opportunity to connect with the community. Reach out to Mark Cooper and we'll start the process.

CppCon 2026 is a significant opportunity to strengthen connections across the C++ community and share the work we've been doing at the Alliance. We'll be sharing more details in the coming months.

Boost Library Updates

The Boost community continues to advance C++ development with significant library updates and milestones. Here's a roundup of recent progress across several key libraries.

Boost.Decimal Accepted for 1.91

Author: Matt Borland (co-authored with Chris Kormanyos)

Boost.Decimal has been officially accepted into Boost following a successful formal review. The library provides a ground-up implementation of IEEE 754 and ISO/IEC DTR 24733 Decimal Floating Point numbers, addressing the well-known representation errors that occur with binary floating point types in applications where rounding errors are impactful, such as financial calculations.

Boost.Multi Accepted for March 5 Review

Author: Alfredo Correa

Boost.Multi, a modern successor to Boost.MultiArray, has been scheduled for formal review on March 5. The library provides multidimensional array abstractions for C++ with a clean, intuitive interface that works seamlessly with the C++ Standard Library. It offers flexible memory layout options and supports both owning arrays and reference views.

Boost.MultiIndex Refactored for 1.91

Author/Maintainer: Joaquín M. López Muñoz

The venerable Boost.MultiIndex library has undergone refactoring for the upcoming 1.91 release. This library enables the construction of containers maintaining multiple indices with different sorting and access semantics, providing powerful data structure capabilities borrowed from relational database concepts.

Boost Library Updates

(continued...)

Boost.Container and Boost.Interprocess new features

Author/Maintainer: Ion Gaztañaga

The Boost team actively maintains classic libraries like Container and Interprocess. Recent releases add comprehensive overalignment support for memory allocators and pool-based classes, extend uses-allocator construction, and incorporate C++20/C++23 API improvements to basic_string and other utilities. The maintainers have also deployed numerous bug fixes and introduced new container options for greater portability across platforms.

Boost.GIL Maintenance Restarted

Maintainers: Stefan Seefeld & Mateusz Łoskot

Active maintenance has resumed on Boost.GIL (Generic Image Library), the C++14 header-only library that abstracts image representations from algorithms. GIL allows developers to write generic code that works across various image types while maintaining performance comparable to hand-optimized implementations for specific formats.

Systems & CI Highlights

by Sam Darwin

Here's a quick rundown of what's new.

Documentation & Previews

A long-awaited contribution to isomorphic-git enabling submodule support has finally been merged and released. This paves the way for improved Antora integration. We've also created a new antora-downloads-extension that automatically retries UI bundle downloads, helping eliminate frustrating build failures in the Boost Superproject.

PR preview comments have been streamlined: instead of flooding inboxes with new comments on every build, Jenkins now updates existing comments with fresh timestamps so developers can easily verify successful rebuilds.

Boost.org Website

We fine-tuned the horizontal pod autoscaler to respond more rapidly during traffic spikes, cleaned up redirect chains to send visitors directly to www.boost.org, and resolved a server crash caused by PDF parsing issues. The team also successfully published Boost 1.90.0, working closely with release managers to address website update challenges.

Release Tools & Infrastructure

A notable addition: "nodocs" versions of Boost releases are now generated and uploaded to archives.boost.io. Teams looking to speed up CI builds can pull these lighter packages. The release workstation was also migrated from GCP to AWS, reducing upload times to S3, with auto-scaling configured via GitHub Actions for release periods.

CI Updates

New container images are available for VS2026 (Windows) and Ubuntu 25.10. We've added documentation build testing to the boostorg/container repository and created a Fil-C GitHub Actions demo for teams interested in exploring memory-safe C/C++ compilation.

Mailman3

Work is underway on a streamlined "Accept and Unmoderate" button for list administrators—though the upstream proposal is still under discussion.