GopherCon 2025 Go Contributor Summit Notes

2025-08-26
https://go.dev/s/gc25-summit

Attendees:

- Madhav Jivrajani, madhav.jiv@gmail.com
- Jake Bailey (@jakebailey)
- Carlos Amedee (@cagedmantis)
- Filippo Valsorda (@FiloSottile) filippo@golang.org
- Roxy Light (@zombiezen)
- Keith Randall (@randall77)
- Rob Findley (@findleyr) rfindley@google.com
- Michael Pratt (@prattmic) mpratt@google.com
- Sean Liao (@seankhliao) sean@liao.dev
- PJ Malloy (@thepudds) thepudds1460@gmail.com
- Dmitri Shuralyov (@dmitshur) dmitshur@golang.org
- Many more…

Session 1

- Go dependencies and breaking changes in dependencies
    - Kubernetes project appreciates the new features in Go 1.23 like GODEBUG
    - Zap had a recent breaking change. Main repository (the import path) changed and introduced a diamond dependency.
    - Adding a "Deprecated:" comment can be finicky. If you don't make it a separate paragraph, it won't register as deprecated. Feels like there might be a "go vet" check possible, or possibly to relax the paragraph restriction.
    - What's the status of automating the creation of v2? go:inline exists. No automated tooling for creating the library. Might be the right tradeoff: much many more callsites than there is library code.
    - apidiff is still useful. Kubernetes depends on it in their CI.
    - When is the right time to move to a module to v1.0? When it's widely adopted. It's fine to make breaking changes before that. Good communication helps.
    - Would it make sense to include metadata in go.mod to forward to new import path.
    - logr vs. slog?
    - Would be nice to have gopls/modernize integration into golangci-lint.
    -
- Project automation
    - Dead bot detection

        - For externally contributed bots, it is hard to tell if they are down, and what to do about it
    - CL tests are getting slower (~15min to ~20 min in a year).
        - race builder is very slow. Now slower than longtest builders
        - Probably just stop running benchmarks on race builders
    - Is test sharding built in?
        - Sharding is done by looking at recent runtimes
        - Sharding would be easier if there were first-party ways to do it
        - Test json output is intended for external tooling to be able to parse test output
        - It would be nice for the test harness to do sharding
    - Original test target was 5 min. We're close to that on the fastest builders.
    - Script for running remote tests on different goos/goarch.
        - Remote test execution is mostly implemented as part of that
    - Most of the slow test friction is in CI, not in the IDE
    - There have been a lot of flakes lately
        - Have to redo trybots quite a bit
        - Would be nice to not block submits if a known flake happens
        - LUCI has support for attributing flakes to a CL
        - watchflakes just looks at logs (from LUCI)
        - It would be nice for watchflakes to auto-close issues that look like they have been fixed
        - It would be nice for watchflakes to detect that failures happen only on particular goos/goarchs (and then auto-assign them).

crpyto/tls profiles

- PREVIOUS DISCUSSION (GopherCon EU 25)
- TLS has ~15 different core configurations options
- Frank Demis – "TLS config should be 'state your goals' and then the library figures it out for you"
    - Configuration should follow "high-level" mandates
- We do not test all the combinations
    - BoGo gives high coverage, but not total coverage
- David Chase why not permute all the profiles?
    - [filippo] I'm not sure that there's a lot of value there
    - Roland Shoemaker there's a lot of TLS implementations – tuple of (go server, go client, 3p impl)
    - [filippo] Not as fast when testing independent implementations
- Roland Shoemaker BoGo takes approx 1 min to run
- [filippo] Real challenge is test matrix that would be highest fidelity in catching the issues
- [filippo] **Create profiles that evolve over time and can be selected by clients**
    - e.g. "CNSA2.0 Profile" or "256-bit Profile"
    - What should an API that does this look like?
    - What is the starter set of profiles?

    - How to set it? GODEBUG or config flags?
    - How overridable should they be?
    - Profiles should effectively "make things disappear" from the implementation
    - [david benoit] mlkem+go1.23 implicitly turns off options in a confusing manner
    - [filippo] new go versions may shift profiles that will change core configurations that could break (e.g. new cipher suite being implicitly bumped)
- David Chase Tim King's internal presentation about "what does the Go module version really mean?"
    - [filippo] it has improved in recent years; however, this is a digression.
- Roland Shoemaker If GODEBUG set and explicit API usage, it should fail – how?
    - David Chase in the runtime, if there's a risk of new changes causing niche failures, GODEBUG is used to revert behavior
    - [filippo] GODEBUG usage for this API would be 2-dimensional – (1) "bring me back" usage and (2) which version of the profile do you select
        - e.g. (1) GODEBUG=tls_go1.24
        - e.g. (2) GODEBUG=<override config>
    - David Chase Would you like another environment variable?
        - [filippo] There is no precedent for that?
        - Roland Shoemaker GODEBUG is overloaded
    - [derek parker] e.g. if a new environment variable, GOTLSPROFILE= would be much more clear
    - [filippo] there's so much GODEBUG infrastructure; we probably should not introduce a new runtime environment
    - David Chase What is the behavior that we desire?
    - Roland Shoemaker One idea that makes GODEBUG more "idiomatic" is that we could add a modern profile that is "restricted" and it would be the default; then a GODEBUG flag would revert the profile to an unrestricted state.
    - [derek parker] would it make sense to load system profiles on disk? does this elide GODEBUG?
        - [filippo] requires exposing all the beahviors of the API;
        - [derek parker] make it easier for forks to change defaults.go
            - [filippo] no backwards compatibility here
    - [filippo] GODEBUG for "time machine" , GODEBUG for "override" , GODEBUG=fip140 acts as a "priority override" with ONLY
        - Roland Shoemaker what are the failure modes for fip140 ONLY?
        - [filippo] good question for the fips140
    - [filippo] How to set it?
        - Roland Shoemaker "tls.SetProfile"
        - [filippo] GODEBUG is for "break-glass" situations
        - [filippo] TLS Config field that should disable other fields?
        - Roland Shoemaker Profile should set the maximal configuration; profiles should change the shape of the defaults.

        - David Chase would the profiles do anything other than disconnecting knobs? is there a notion of compatible set of restrictions?
        - Roland Shoemaker the profile designs a set of valid config options; the config itself is a subset of that; no profile combinations are valid.
-

- VS Code / IDE
    - VS Code version check not working in workspaces
        - Remove check for Go version from VS Code – just query gopls for each file
        - Workspace issues
            - Zero-config gopls has issues in large monorepo: some things work, but others don't
            - Gopls applies GOWORK=off, but other tools invoked from VS Code don't, leading to an inconsistent experience and confusing error messages
            - Should go.work files be ignored if you're in a module that isn't included in the go.work file
            - Go build error message could be improved
    - LSP wish list
        -

Session 2

post-quantum cryptography

- [derek] RHEL, a lot pressure for mldsa
    - [filippo] first concern should be does this make sense at this current time? there's many artificial pressures that could merit doing something less desirable like exposing 3P implementations in the standard library
    - Roland Shoemaker we have a post-quantum roadmap that says we desire a use-case in the real-world before we put PQ in the standard library
- Peter Weinberger why is it a good idea?
    - [flilippo] KEX is risk limiting idea; "i'm not a physicist" but I am not sure they will be able to do what they think
    - [filippo] I don't care about signatures now
    - Peter Weinberger internet doesn't move fast when it comes to crypto;
    -

compiler/runtime/cgo

- michael m: Slow cgo builds
    - matloob: Pathologically slow cases
    - iant: Problem inside google
    - cherry: inside google it's linking
    - iant: cgo builds are delaying later builds, building C++ code

    - cherry: so Google's internal build graph could be better
- michael k: purego?
    - felix: symbol conflicts with Rust-written libraries, using purego for that
    - cherry: is anyone interested in using it? or just calling C? harder to do on some platforms vs. others.
    - iant: who cares about this?
    - felix: want to ship a shared library in C++, and worried that with cgo it'll be difficult in CI
    - cherry: filippo said people who want that just want to call sqlite
    - jake: avoiding C and we would like to use a file watching library, but cgo is painful. easier on windows
    - cherry: eli once mentioned something related to LLM models (local models)
    - jake: another use-case: Microsoft still wants to do its own thing for crypto, and we need to build OpenSSL – this would simplify that as well
    - derek: same use-case for OpenSSL at RedHat, but our plan is to switch to the upstream FIPS stuff
    - felix: do we have a tracking issue?
    - AI: create a tracking issue
- cherry: new topic: upcoming changes affecting delve
    - cherry: randomized base heap address
    - derek: nothing I'm worried about
    - cherry: re: SIMD, now we'll have magic vector types. we probably need some way to represent this in delve
    - david: it's not clear what SIMD users want to see in their debugger
    - jake: isn't delve smart enough to see constant bit patterns?
    - derek: yes
    - keith: one change where when we're loading a slice into registers we're going to represent it as 4 words (base+offset instead of just an address)
    - iant: why?
    - keith: avoids having to deal with past-the-end pointers when slicing. right now you have to be careful that you don't slice to the end of the backing store and point to the next object.
    - david: ran into this a lot in SIMD performance early, and I don't think this CL is submitted but it's out there. there's a rewrite hack that can help a lot of things.
    - keith: let's talk.
    - piotr: will this be in DWARF.
    - keith: yes…?
    - cherry: yes, but the tricky part in the compiler is that we have a different representation in different circumstances.
    - frank: I've seen a lot of cgo libraries that depend on slice shape
    - keith: the in-memory representation is the same, it's just in registers
    - iant: could it be expressed as a sum in DWARF?
    - derek: been working on making sure more things have location lists.
    - piotr: I think this would be the first use of DWARF expressions.

- cherry: measuring go performance changes downstream
    - See https://perf.golang.org/dashboard/

Go packages driver

Go in the enterprise

- Telemetry and monitoring - general devx
- Breaking changes
    - Upgrading go versions and go modules aggressively is causing an issue
    - go/release tool as a way to detect and inform of breaking changes
    - K8s community is an example that that makes highly frictional changes
        - K8s flips v1 to v0 to avoid breaking changes. Pulling a dep from K8s vs something else like argo can be fully incompatible
    - Why are these ecosystems doing this?
        - Some of this is bad design (i.e. should have been `internal` package bug wasn't build that way)
        - Some of it is that the release process would be too difficult
        - Some libraries have strong compatibility guarantees (e.g. k8s/utils is good), we should have more of these
    - If companies begin funding these projects and start cutting funds when projects break compatibility guarantees, that might be a strong enough stick to prevent this in the future
- Upgrades
    - Can the Go team provide guidance about upgrading dependencies for libraries vs services
        - i.e. libraries should be the minimum version but services may want the maximum version
    - Can this be specified in the go.mod?
    - Can Renovate/Dependabot be configured for this?
    - It would be nice to have an open source version of Google's TAP where projects can run the tests of all of the open source projects that would be impacted in order to understand what would break
        - How can teams understand the closed source impact
- golang.org/x go version
    - golang.org/x sometimes frequently upgrades to the newest version of Go and forces downstream dependencies to upgrade their Go version sooner than they prefer to.
-

Session 3

WebAuthn and Passkeys

- [filippo] Passkey API addresses much higher risk issue that's relevant
- [filippo] API opinions
    - Ideally, high-level API that with the ease of password hashes
    - Passkeys starts with database table and FK to (cred, metadata, …)
    - Passkeys should be a blob in the database
- [filippo] WebAuthn defines the set of APIs to obtain signatures from the platform that binds keys for a given client that are not phishable
    - Need a list of pre-loaded random values that can be expired
    - Hard parts API wise are lookups per user
    - API has two touchpoints with the browser
    - Multi-stage API?
- [Takashi] Should the browser provide common interface for many authentication methods?
- [dmitri] Should the backend just wrap the frontend API? e.g. using WebAssembly
    - [filippo] Should there be a companion component?
    - Roland Shoemaker more people using Go Web assembly would justify this
    - [filippo] Client would need to shipped as JS not Go
- [takashi] digital credentials API that chrome uses to tie to hardware
    - [filippo] alternative to cookies, not passwords
- [filippo] It should be a high-level "passkey" package if anything
- [nealpatel] UX for native vs. web
-

How to make Go project approachable

- [nonametag] The proposal process is unclear — when someone publishes a proposal, it's unclear where in the queue it is. When will it be considered? Lots of frustration. I did some proposals where they were reviewed immediately and some were never looked at. It's a major pain.
    - [alan] Things are often quite small and simple, but there's just so many. It ends up being haphazard.
    - [austin] We are thinking of ways to make the queue more transparent. It's a big bottleneck. A thing that has helped — we get an hour a week. We split the proposals into low / high controversy queues. We spend the first ½ hour on low controversy. If we exhaust, we go to high controversy. It's a system of queues.
    - [nonametag] Can we consider splitting queues by workstream?
    - [austin] We are considering work queues / task forces as an experiment. We have them for jsonv2 and some others. We also want subcommittees which are more specific than the subteams.
    - [nonametag] At companies like Uber / Meta, we have our own build systems. Typical Go users who use their own build systems run into some different problems.

- [pjmalloy] Enforcement of conventions are sometimes inconsistent, even among Go team members. An external team member creating an issue along with their CL is more work externally, but there's a lot more traffic on the issue tracker. Few people are looking at Gerrit. Should we be enforcing issues more, or is that just more noise for the Go team?
    - [pjmalloy] It says it won't be accepted without an issue. Getting more things into the issue tracker is helpful for the Go team — you want more people to jump in. Even if it might be obvious for us, having the issue helps "give the bug a URL". We want people to be able to find things and tell us how we broke their esoteric use case.
    - [iant] Sometimes there are truly trivial cases which just don't need an issue. If you change a comment, there's no point. If there's behavior change, maybe you do.
    - [keith] Sometimes even a large optimization that's straightforward and changes no semantics, maybe that doesn't need an issue either.
    - [pjmalloy] In the standard library, we still sometimes see bug issues with true behavior change with no issue. Maybe it still depends on where you're working. Agree that not everything does need an issue, even optimizations, etc.
    - [jakebailey] Does having issues help when writing release notes? Is there value there?
    - [pjmalloy] The more issues there are, the more we get people commenting.
        - [iant] Not necessarily a good thing. More comments can add a lot of noise.
    - [alan] It's generally quite simple to create an issue. It's not a super high barrier. With a CL, it's opaque.
    - [michaelm] I get lots of CLs where there is a single line commit message. It's easy to get lost in the weeks. With an issue there's a different context — it encourages people to talk.
    - [marc] The issue tracker should be the place to talk about the problem — Gerrit should be the place to write the solution.
    - [michaelm] It's generally easier for new contributors to get started if they begin with an issue. It's a lot harder if we jump right to the CL.
- [unknown] Can the GopherBot open a request automatically if there is a CL with no issue?
    - [pjmalloy] It already points it out if there isn't one. It started out apologetic, but now it's more assertive. This does drive compliance.
    - [sean] Some of those messages are too long.
    - [jakebailey] Let's not automatically open issues.
    - [austin] It's probably not useful to fully automate that.
    - [sean] Can we make the message attached to the commit dropdown itself, rather than commenting on the CL?
    - [pjmalloy & michaelm] It's quite hard to reverse engineer what a CL is doing if there's no issue talking about what they were trying to do.
- [jakebailey] I don't have try bot permissions. We can only run the try bots on our CLs — not somebody else's. Maybe I just need permissions.

    - [russ] There's an internal security barrier possibly.
- [jakebailey] Sometimes my CLs get sent to contributors who are not terribly active. Does the owners file need to be updated?
    - [austin] It's easy to get tragedy of the commons here. Nobody is working on this — but if we had a system where CLs could get assigned to queues owned by groups of people, that might be better. Right now it just gets assigned to a handful of people and everybody thinks the other people will do it.
    - [jakebailey] We automatically assign the opener, or people involved with an issue as reviewers if it gets linked to the CL. Oftentimes, a specific person owns an issue — it might not be a team.

Fuzzing/code coverage

- Getting runtime code coverage? Teams at Uber are getting coverage data by hitting a debug endpoint. Go 1.21 made a change with regards to atomic mode. Is it possible to run "set" mode? We have problems with race detection?
    - What is atomic coverage? It uses sync/atomic counters for each line.
    - This is how fuzzing works too.
    - "Set" coverage is zero or one rather than accumulating counters.
    - Biggest reason for runtime code coverage is dead-code detection.
    - Might be worth it to creating tooling specifically around dead-code detection.
    - Might be worth a proposal to have tooling like PGO to canary coverage then don't instrument known-live code.
    - Is stack coverage (like in telemetry) useful?
    - Use cases
        - I want how a particular segment of code is being used
        - Search across all code to find dead code
- Friction with Fuzzing
    - Unfamiliar so people don't know to reach tool
    - Making decisions around where server compute lives (even getting load tests running is tricky)
    - There's not an option to run more than one fuzz test in the same
    - Using generative AI for mutation testing?
    - Fuzzing was dropped after its MVP phase (which is where we're at today)
- Code coverage for integration testing
    - Created a stub for running `go build` with the correct flags and configure for coverage directory
    - For system testing, ideally server processes could have a debug endpoint that coverage could be pulled
    - Code cover profile has some friction with VSCode.

Formal verification and feasibility of Go compiler and runtime

    - Q: what benefits would exist to formally verify parts of the Go compiler or the runtime? Is this something that we see a benefit in doing? Are there parts of the compiler/runtime that would benefit from verification (sans the cost of it).
    - Ex of formal verification in a Go project: etcd/raft - TLA+ traces are logged out at runtime and those traces are verified against a parent model to try and better verify runtime behaviour
    - Maybe trace logging via an -x flag?
    - Log traces for the compiler or the runtime and then the model itself needs to be maintained by the user.

Session 4

FIPS

- [filippo] policy enforcement is still an open issue
    - if enabling this mode, you may still use non-FIPS compliant configs without any warning or breakage
    - recommendation:
    - some users desire a way to turn off non-FIPS things e.g. GODEBUG=fip140-only
    - feedback about this "only" mode: things that negotiate algorithms (e.g. SSH) needs to know ahead of the time otherwise it will select the wrong algorithms
    - major issue: when combining mlkem,x25519
        - using x25519 alone: bad
        - using it part of hybrid: good
    - websockets use SHA1 in a non-security context for checksums
        - these codepaths should not error despite the non-FIPS mode
    - AWS uses md5 checksums should not "explode" in FIPS-mode
    - [rsc] examples of desiring "exploding errors"
        - [filippo] redis uses SHA1 for cache keys
    - [rsc] "WithFIPSOff" — same semantics as the "constant-time" crypto thing
- [filippo] e.g. instantiate AEAD and then not using it (like pre-filling map)
    - in this case, it explodes before it should
- [rsc] extra work for enforcement should be ok
- [frank] extra work is okay for ensuring this boundary
- [redhat team] static analysis and runtime enforcement
    - having both is valuable
    - [rsc] govulncheck would serve the former
- [rsc] what are the core needs
    - [filippo] Bubble behavior for hybrids
    - [filippo] GCM for SSH (opinions)
        - [rsc] not the end of the world to add this as a public API
- Roland Shoemaker different nonce set for one of our x/ aead implementations
    - [filippo] testing interface of mlkem that does not use system randomness for encapsulation
    - [filippo] that should likely be in testing/cryptotest

        - [rsc] should think about whether or not they should separated out for the sake of clients importing them
        - [filippo] don't see that many more being added that merits splitting
- [rsc] of all the crypto warts, this implementation of policy enforcement seems low on the list
- Roland Shoemaker what are the semantics for goroutines spawned inside the boundary bubble?
    - [rsc] should share the same semantics as the constant-time thing
    - [rsc] bubble discussed above needs to use glocal storage; new goroutines do not inherit it and that should be fine
- [rsc] fips140-enforced, fips140-without-enfrocement
- [redhat team] With the OpenSSL fork, 3P SHA256 SIMD pass differently implemented structures that emulate signature algo structures
    - vendor libraries can use non-compliant crypto that Go implementtations will acept it
    - [rsc] they're unsafely digging into memory to get the digest structures?
    - [rsc] there exists a way for us to catch this now

MCP

- https://github.com/modelcontextprotocol/go-sdk
- There's an MCP server for gopls (un-advertised)
    - Exposes info about go build and go workspace
    - Kinda messy right now, LLMs might over-call tools and exhaust tools
    - Good thing: can provide docs
- Are we planning to test with other LLM providers? Is it just claude right now?
    - Gemini is much worse than Claude in many cases
    - Tool call is controlled by the system prompt - this is the problem with poor tool calling like in gemini
    - System prompt does not fix it completely still
    - Gemini does a bad job of using MCP tools
- Most users are using claude overwhelmingly, gemini is cheaper but still not great
- How does Go position itself in the MCP SDK "market"? Distinctly better than typescript in terms of UX
    - Having a remote MCP is highly desirable - Go is excellent for that
        - Also great for commands, ease of builds and cross compilation
    - Need to do more promotion for Go
        - Most MCP servers is npx abc.
        - People use that because momentum and npm automatically updating to the latest version
        - Distribution is the issue - discussions around using existing toolchains to distribute Go MCP binaries
    - Remote MCP server hosting platforoms like cloudflare, vercel etc only support typescript

        - Cloudflare heavily relies on V8 isolates: https://developers.cloudflare.com/workers/reference/how-workers-works/#isolates
        - Properties like that can send a strong signal to support remote Go MCP hosting
        - If platforms support Go MCP hosting, that is a big driver for adoption
    - V1.0 is on track for the go mcp sdk
- Goal is for the SDK to be contributor maintained.

Proposal Process

- [chris] I personally want to be more involved in the proposal process. The weekly announcement thing is the best thing the Go team ever did for transparency. We often used those as discussion starters at my meetups. We have to limit the human tendency to keep talking and talking forever about things that have already been addressed.
    - [austin] Once a proposal that has hundreds of comments — many people won't read those comments. Experimenting with LLMs to get summaries out of that stack of comments.
    - [chris] Those end up using too many words to say too little.
    - [mpratt] You can often tell it something and ask if that's been said, rather than rely on it for the summary.
- [alan] We've talked about federalizing things into committees. Rejection power should live in those committees before things get sent up for a higher review.
    - [austin] We have task forces for jsonv2 and collections — they have a goal and termination criteria.
    - [chris] I was involved in jsonv2 from the beginning and on the video calls that Joe ran. I have some helpful feedback on how some of those things ran and what was useful. This idea of subgroups is useful; there are some that I'm interested in and it's a more productive use of my team to be there. For other stuff like crypto, I have no opinion.
- [iant] There should be a committee that says no — we need to be able to say no to things quickly. It's difficult to say no to things as a lone IC.
    - [pjmalloy] That can also assist with spreading the load of writing out that paragraph closing the proposal.
    - [alan] We should all feel empowered to give a vociferous opinion of "this sounds like a bad idea" — if Robert Griesemer gives an opinion like that, maybe it's not worth looking deeper.
- [chris] What sort of feedback did you get on the error syntax ban?
    - [iant] It's been great — Sean closes lots of issues.
    - [austin] We don't have statistically unbiased opinions.
- [ian] Has an upcoming proposal — not sure if his feedback on the proposal process would be valuable, but happy to provide notes.
    - [chris] Often, the first time somebody does something on a team, that's the best time for feedback — you can only collect that one time.
    - [austin] The proposal process should not be intimidating.
- [sean] The proposal committee has been rejecting less proposals lately.

    - [austin] It's unclear if we're rejecting more on intake or if the bar has shifted.
    - [chris] After we had the forward compatibility stuff with Go 1.21, it opened up the door to more complicated changes. Before then, there was a lot more strictness — we had just had generics. Now there are bigger changes to the language and ecosystem than we would have entertained before. There is a change in the willingness to be more adventurous.
    - [austin] For a combination of reasons, that's where we had to be. We've solved a lot of the technical problems that used to block us.
    - [chris] It's not a bad thing, but we have to be careful about getting too crazy. The short function proposal for instance does not seem to bring much value. We seem to forget all the change this will impose and the books that will go out of date. Part of Go's value was its conservative stability. Maybe we need to slow down again.
    - [austin] No good answer to that.
- [nick] Can subcommittees still say yes to proposals unilaterally?
    - [austin] They can probably make a recommendation that will be considered strongly.
- [pjmalloy] If you read the proposal document right now, it sounds like it's going right to the proposal committee. Mentions the 256-bit proposal with Keith / Ian where they were frustrated that their proposal seemingly was closed without review.
    - [austin] I think it says they should go to the committee once it reaches consensus — we never have consensus.
    - [iant] The IETF says rough consensus and running code.
- [chris] How many dormant proposals do we have that are lost in the process?
    - [austin] There are about 900 of those.
    - [alan] I've taken a look at them sorted by emojis and it's about 90% shutting them down.
    - [pjmalloy] There's a big waiting room by design to avoid livelock.
    - [austin] We have a similar issue on the other side — about 250 proposals that are accepted but not implemented. Many are incompatible with each other. We are always reviewing on top of the language as it is now.
    - [chris] If there's no plan for implementation, it seems like a challenge to accept it.
    - [austin] Oftentimes, the person making the proposal is probably the best person to implement it — but because the proposal process takes so long, they oftentimes have just moved on by then.

<u>WebAssembly</u>

- What is WebAssembly?
    - WebAssembly is closer to an AST, not an IR. There's only structured control flow.
    - It's like assembly but vaguely memory safe for sandboxing reasons.
    - Stack-based
    - Offers SIMD

    - The Go compiler can target WebAssembly for output. However, this is a complex target because of the structure required.
    - WASM does have some system calls like allocating memory.
    - Designed for one-pass machine code translation but can't generally benefit from feedback-driven optimizations.
- WASI: WebAssembly System Interface
    - Gives something close to POSIX
- Size concerns
    - Go runtime is largely fixed size, so maybe 10% of your payload.
- Who is the target customer for this?
    - Browsers, CDN compute
    - When TinyGo isn't an option because you need Go libraries that require features that TinyGo don't have
    - Maybe plugin architectures, but generally, the performance of such usages is generally prohibitive.
- There's finally some concurrency. But WASM threads != WASI threads. There are atomics now. But it doesn't help the Go runtime because the concurrency happens outside the WebAssembly sandbox.
- JSPI: JavaScript promise interface
    - Bi-directional asynchronous await. Await on host promises and await on WebAssembly promises.
- Go team met with the WebAssembly working group with regards to stack switching. They seem to be focused on JSPI rather than Go's concurrency.
- What about JVM as a target?
    - Not a good match for Go or C
    - It is heavily geared toward Java
    - Doesn't have interior pointers
    - JVM garbage collectors assume that the stack will be small and mostly insignificant. This is not the case for Go programs.
- Go's compilation process to WebAssembly
    - Invents a program counter and virtual method call
    - This introduces a huge performance cost

Session 5

<u>Memory Profiling</u>

- [Felix] topic about go tool trace
    - For those who use it a lot, it's clear that it can be better; however, not what to improve?
    - Large traces are difficult to visualize
    - Tried to write a new viewer from scratch, focused on performance
    - Feature creep impacts performance

    - Usability problems exist as well
- Michael Knyszek
    - Most of these bespoke profilers serve a specific use-case, not a general one
    - It will have gaps when generalizing
- [Felix] translating wire format to JSON and then other formats and uses will degrade performance
    - Room for fun side projects: (1) use the tooing (2) find the pain points
    - Firefox profiler is good next target
    - Should be web-based viewer by default?
    - Browser APIs are fast-enough to render
    - Worst-case is tight-loop, goroutines pinging back and forth creating many small rectangles (render scaling issue)
- [Nicholas] Portability is extremely desirable for me, especially in offline situations
- Michael Knyszek go trace tool – scales more readily to large traces
- [Felix] p-centric view vs. g-centric view
- [Rhys] what if an RPC server with 1M goroutines?
    - Michael Knyszek we do have the "anything to do with"
    - [Rhys] the default heuristics group too liberally (e.g. net/http)
- Michael Knyszek ideal: many primitives (e.g. "lane" or "sub-lanes") in a singular HTTP client where you can attach information to those lanes
    - Simple API: browser doesn't maintain state or concept of what's going on
    - Instead, the go tool itself precomputes what's interesting and why
    - The browser acts as a specialized drawer
- [Felix] Javascript is capable of managing this in its own memory
    - Do you want to do this?
    - A good viewer should limit what goroutines you want to see
    - Algorithms to group this is non-trivial
- [Nicholas, datadog] Memory/Block/Mutex profilers use too much memory
    - one record per unique stack trace for the lifetime of the program
    - this is bigger deal for webservers or longer running jobs
    - recursive parsing / data structures
    - 100s MB is normal in this niche application
    - [Felix] small subsets experience this, but it is painful
    - on/off for these things would be good
    - Michael Knyszek there's a proposal for dropping data and not starting the collection at the beginning of the program
- Alan Donovan why not add more metadata for the frontend to consume?
    - Michael Knyszek unfortunately we cannot make go tool pprof changes since we do not own the tooling
- Cherry Mui if designing a new library or API for the purpose of viewers, consider the LLM integration and how agents might be useful in suggesting next points of investigation
- [Rhys] what if the go tool pprof could have a two way implementation

Debug Info

- New debug/gosym package to add information about inlined functions.
- Delve has patches for working with stripped binaries. Forked govulncheck's fork of debug/gosym.
- [Piotr] Want the overhead low for small lookups.
- [Piotr] We try to find the moduledata by searching for structures that look correct. It works surprisingly well.
- Every inlined function has a PC. Do we expose that?
    - [khr] We'd rather not expose those. We'd like to get rid of the NOPs.
    - [prattmic] From an API perspective, it would be nice to just have a struct with a pointer to the parent.
- debug/elf, debug/macho, etc have no higher level interface. So there is a lot of duplicate code doing the same thing. It would be nice to have a general interface.
    - [prattmic] A new API would unfortunately have to duplicate most existing methods. Not a high priority because it just enables a nice refactoring. debug/gosym/v2 gives new information you can't get right now.
- [Piotr] More information in DWARF?
    - Loclists are sometimes incorrect
    - [Derek] I've been working on SSA improvements to make it more aware of return parameter liveness. It may be difficult to plumb the data where it is needed. For named and unnamed return values.
- [Piotr] DWARF prologue starts before or after stack setup depending on GOARCH.
    - [Derek] Seems like a bug?
    - [Piotr] Want for compatibility with probes.

SIMD

Toolchain versions