



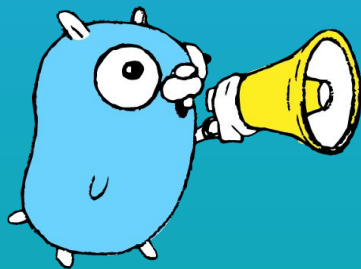
02/26/2025

# What's GOCACHEPROG ?

sivchari

CyberAgent

The Go gopher was designed by [Renée French](#).



自己紹介

---

GOCACHEPROGとは

---

GOCACHEPROGの挙動

---

GOCACHEPROGを触ってみる

---

GOCACHEPROGユースケース

---

まとめ

---

“



# 自己紹介



”



- **sivchari**
  - **X/GitHub @sivchari**
- **CIU**
  - **AKE (Astro Kubernetes Engine)**
- **CyberAgent Go Next Experts**
- **Go Conference メインオーガナイザー**



“



# GOCACHEPROGとは

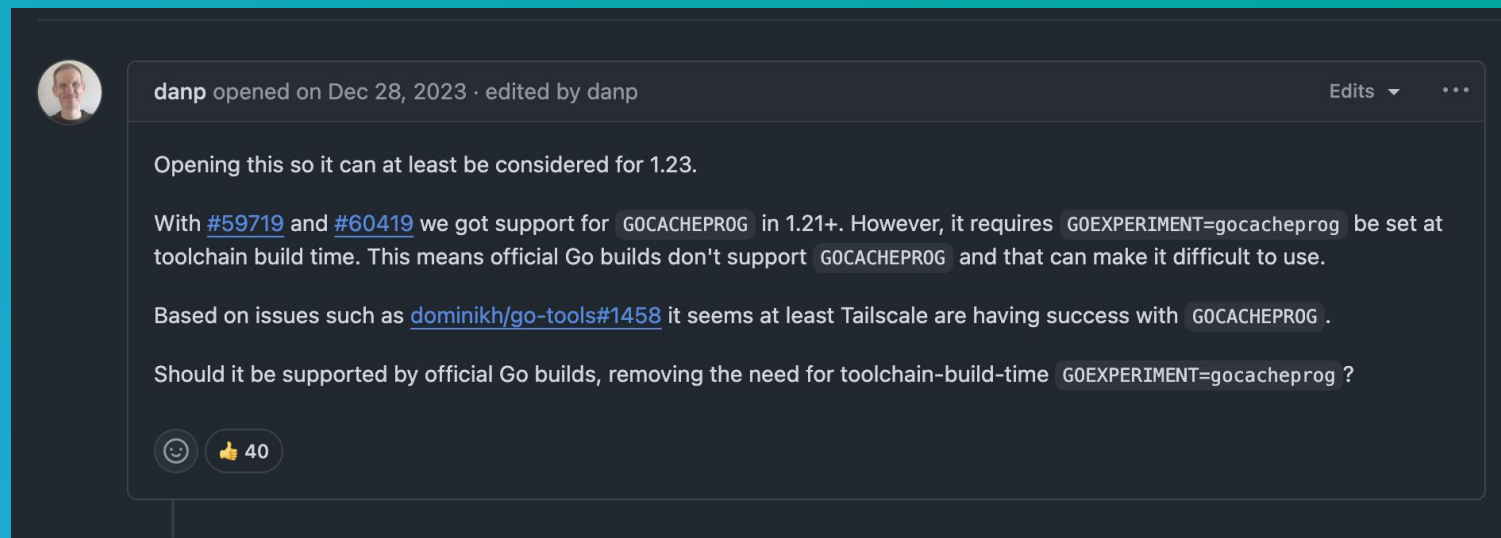


”



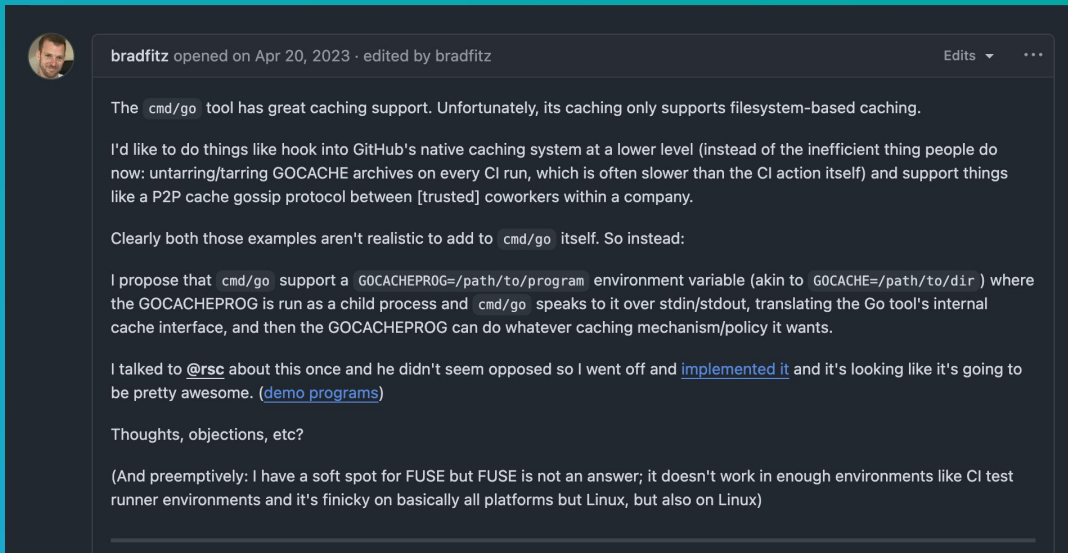
## GOCACHEPROGとは

- Go1.24からデフォルトからの機能



## GOCACHEPROGとは

- Go1.21から実はGOEXPERIMENTで導入されていた
  - リリースノートへの記載はなし



## GOCACHEPROGとは

---

- リリースノート
  - <https://tip.golang.org/doc/go1.24#gocacheprog>

The `cmd/go` internal binary and test caching mechanism can now be implemented by child processes implementing a JSON protocol between the `cmd/go` tool and the child process named by the `GOCACHEPROG` environment variable. This was previously behind a `GOEXPERIMENT`. For protocol details, see the documentation.

`cmd/go`の内部バイナリ、テストキャッシュの機構は `cmd/go` ツールと `GOCACHEPROG` で指定された子プロセス間の `JSON protocol` の実装により、子プロセスにより実装できるようになりました。以前は `GOEXPERIMENT` により隠されており、詳細はドキュメントを参照してください。



## GOCACHEPROGとは

---

- ドキュメント(cacheprog package)
  - <https://pkg.go.dev/cmd/go/internal/cacheprog@master>
  - デフォルトでは既存のファイルシステムによる内部で持っているキャッシュ機構を利用する
  - **cacheprogにGOCACHEPROGで定義する必要のあるAPIが存在する**
    - **This API is internal :(**

“



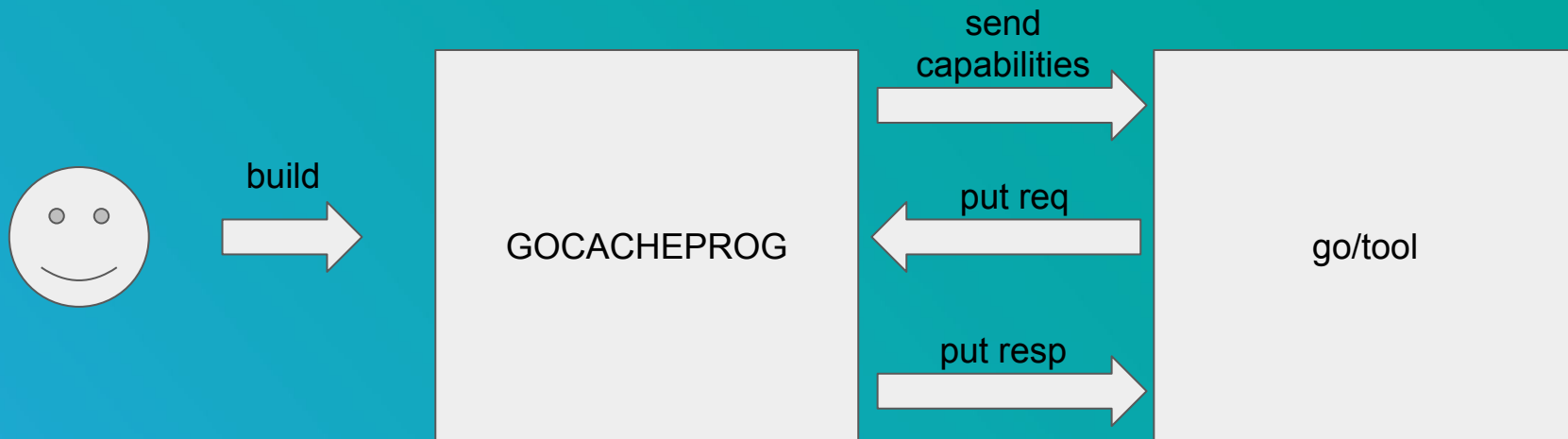
# GOCACHEPROGの挙動



”



## GOCACHEPROGの挙動



## GOCACHEPROGの挙動

---

- **send capabilities**の書き込みから開始される
  - 現在は **Put, Get, Close**
  - 起動するプロセスが何をできるかを送る(MCPみたい)
    - **Put**を消すと絶対にキャッシュがヒットしない
  - ユーザー主体での拡張は現状できない

## GOCACHEPROGの挙動

---

- それぞれの**Capability**には制約がある
  - **Put**
    - **Action ID**をキーとして**Output ID**と**Body**を保存する必要がある
    - 終了までローカルのファイルシステムにも書き込み、**DiskPath**として保持する必要がある
  - **Get**
    - キャッシュがない場合**Miss**を**true**にする
  - **Close**
    - プログラムを終了させる

“



GOCACHEPROGを触ってみる

”



## GOCACHEPROGを触ってみる

---

- リポジトリはすでにいくつかある
  - <https://github.com/search?q=GOCACHEPROG&type=repositories>
- **bradfitz**さんのサンプルはgoroutine周りのハンドリングが失敗しているため修正版を試してみる
  - <https://github.com/or-shachar/go-tool-cache>

## GOCACHEPROGを触ってみる

---

- `git clone git@github.com:or-shachar/go-tool-cache.git`
- `go build -C cmd/go-cacher`
- `GOCACHEPROG="./cmd/go-cacher/go-cacher --verbose" go install std`
  - 1回目はキャッシュがないため時間がかかる
  - 2回目はキャッシュが当たっていることがわかる



“



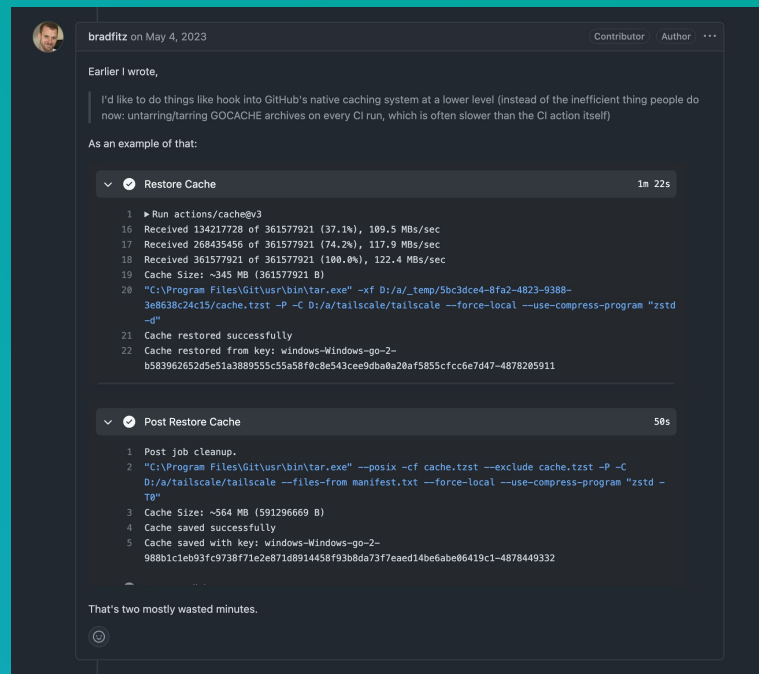
# GOCACHEPROGのユースケース

”



## GOCACHEPROGのユースケース

- CIの速度改善
  - **taring/untarringをやめる**
  - 部分的な**cache hit**
    - **build cacheはpackageごと**
- メトリクス
  - ビルドのキャッシュヒット率
- HTTP Serverによるキャッシュ共有



The screenshot shows a GitHub Actions workflow log for a user named bradfitz. The log is divided into two main sections: 'Restore Cache' and 'Post Restore Cache'.

**Restore Cache (1m 22s):**

```
1 ▶ Run actions/cache@v3
16 Received 134217728 of 361577921 (37.1%), 189.5 MBs/sec
17 Received 268435456 of 361577921 (74.2%), 117.9 MBs/sec
18 Received 361577921 of 361577921 (100.0%), 122.4 MBs/sec
19 Cache Size: ~345 MB (361577921 B)
20 "C:\Program Files\Git\usr\bin\tar.exe" -xf D:/a/_temp/5bc3dce4-8fa2-4823-9388-3e8638c24c15/cache.tzst -P -C D:/a/tailscale/tailscale --force-local --use-compress-program "zstd -d"
21 Cache restored successfully
22 Cache restored from key: windows-Windows-go-2-b583962652d5e51a388955c55a58f8c8e543cee9dba0a28af5855cfc667d47-4878205911
```

**Post Restore Cache (58s):**

```
1 Post job cleanup.
2 "C:\Program Files\Git\usr\bin\tar.exe" --posix -cf cache.tzst --exclude cache.tzst -P -C D:/a/tailscale/tailscale --files-from manifest.txt --force-local --use-compress-program "zstd -T0"
3 Cache Size: ~564 MB (591296669 B)
4 Cache saved successfully
5 Cache saved with key: windows-Windows-go-2-988b1c1eb93fc9738f71e2e871d8914458f93b8da73f7eae14be6abe06419c1-4878449332
```

At the bottom of the log, it says: "That's two mostly wasted minutes."

“



まとめ

”



## Summary

---

- **GOCACHEPROG**によりキャッシュの拡張が可能になった
- サブプロセスに委譲するような機構はプラグ化しやすく他言語、**WASI**などとの連携もできるようになりそう
- **API**定義を公開しているのでぜひご利用ください
  - **<https://github.com/sivchari/gocacheprog-api>**