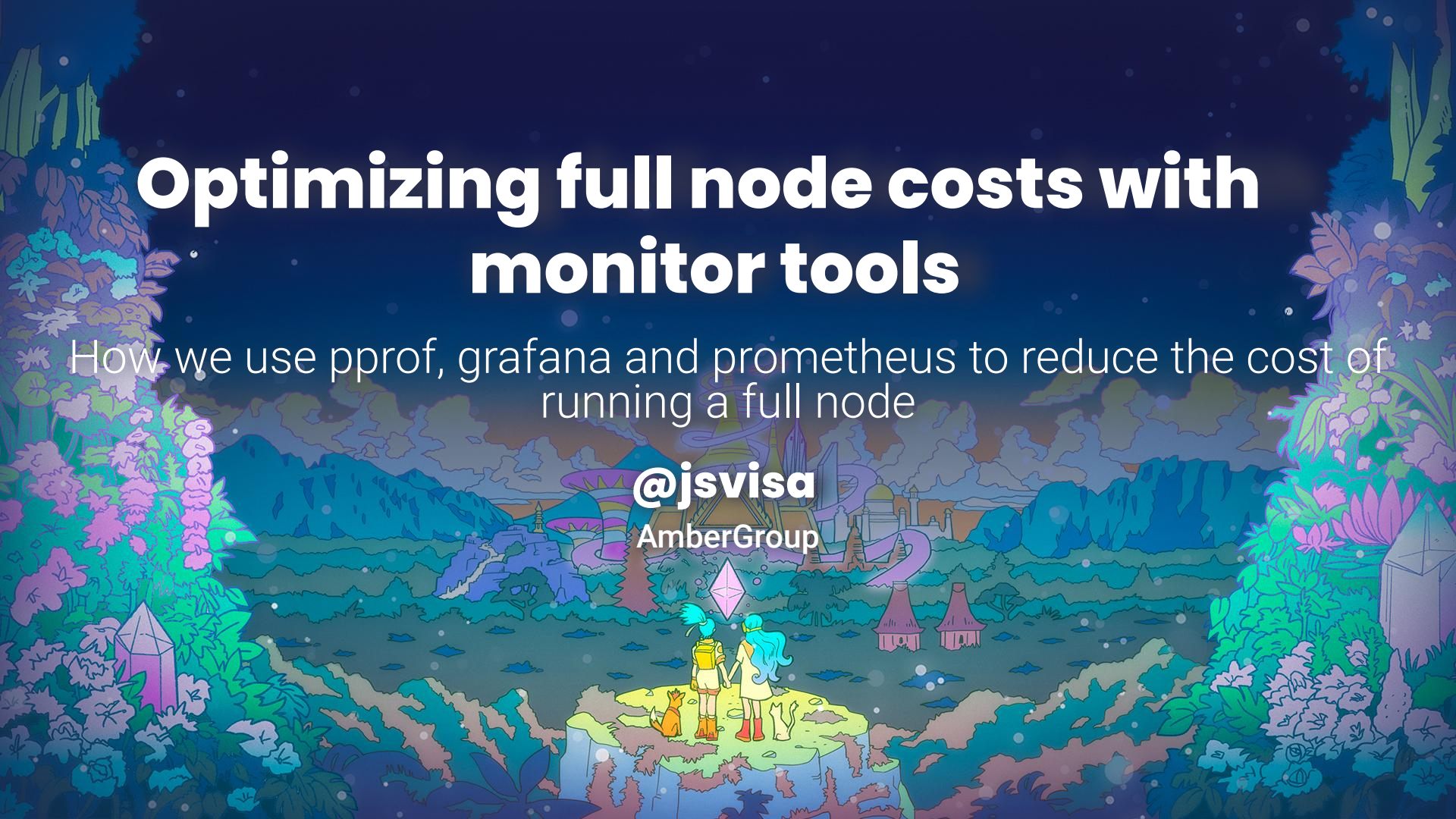
Optimizing full node costs with monitor tools

How we use pprof, grafana and prometheus to reduce the cost of running a full node

@jsvisa

AmberGroup



About me && Ambergroup

AmberGroup:

- A leading digital finance company
- Providing comprehensive solutions for digital asset management
- Offering scalable, all-in-one Web3 infrastructure solutions
- Amber.ac has launched the BUIDL_QUESTS Web3 Innovation Challenge

Me: @jsvisa

- DA Leader of AmberGroup Web3 Security Team
- An independent ethereum ecosystem contributor(mainly on execution clients: go-ethereum, reth, erigon)

About me && Ambergroup

Me: @jsvisa my contributes on ethereum ecosystem from GitHub(2018–2024.09)

year	erigon	go-ethereum	prysm	reth	others	total merged PRs
2018	0	Closed: 12, Merged 31	0	0	0	31
2019	0	Merged: 1	0	0	0	1
2022	1	Closed: 2, Merged: 9	Merged: 2	0	Closed: 1	11
2023	4	Closed: 22, Merged: 72	Merged: 8	Merged: 4	Closed: 1, Merged: 4	88
2024	4	Closed: 1, Merged: 6	1	Closed: 13, Merged: 75	Closed: 4, Merged: 18	100

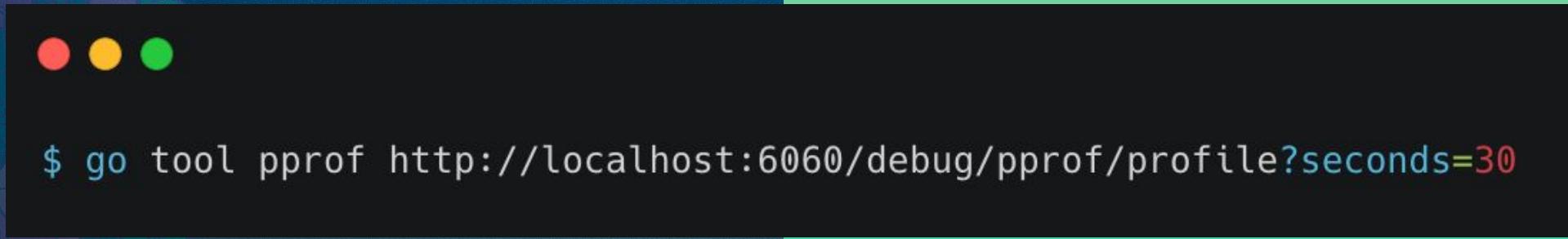
Here is one of the performance issue we found in execution client geth

Let’s find the performance issues

cadvisor dashboard in Grafana

Let’s pprof it

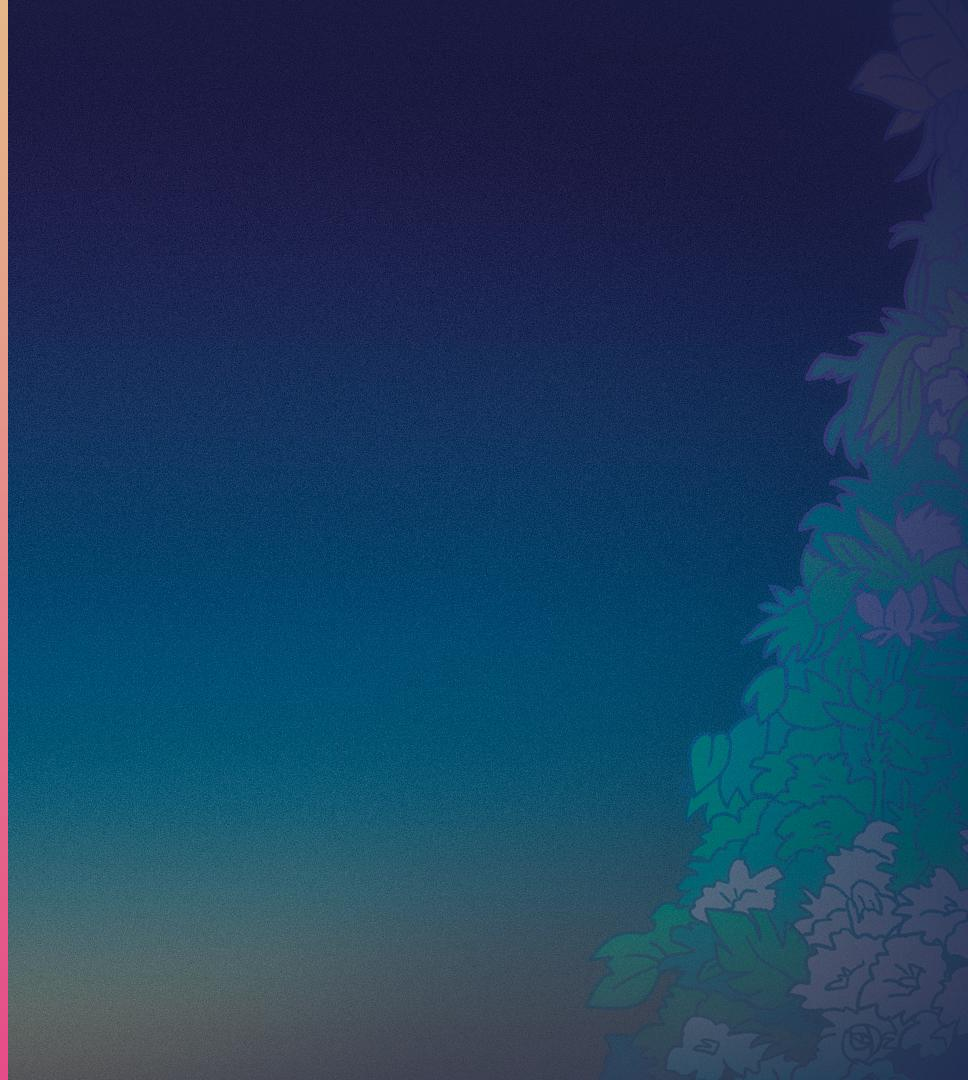
Let’s use the pprof tool to see where and how are the abnormals



Let's pprof it

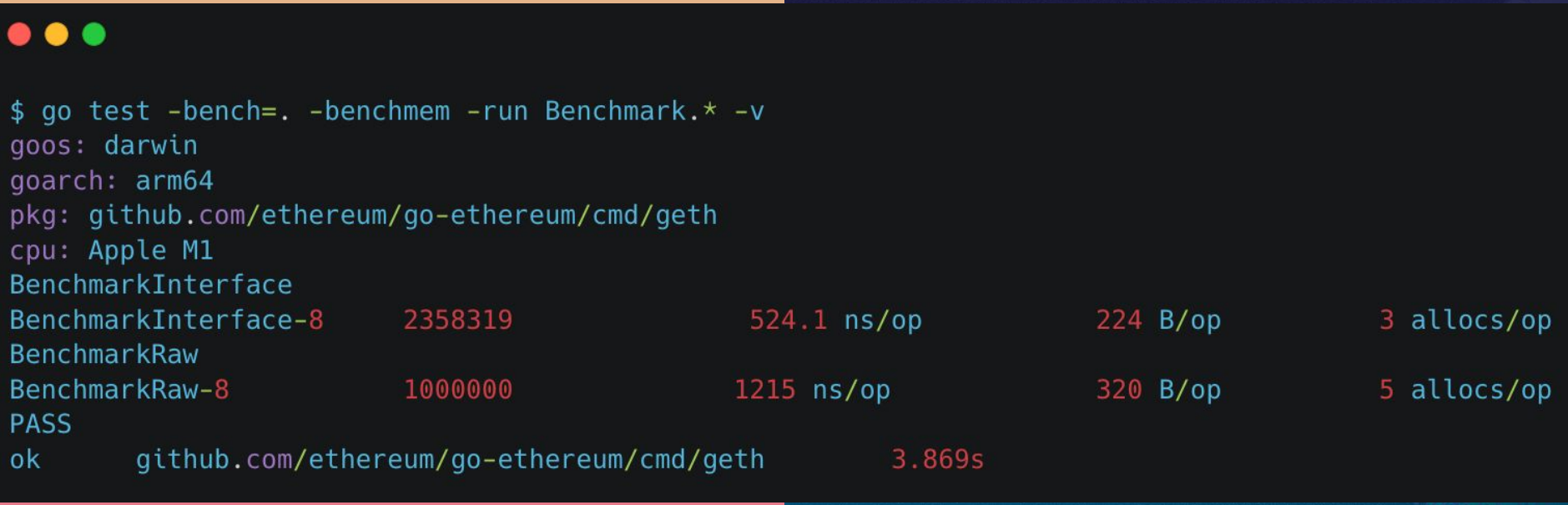
What’s the problem?

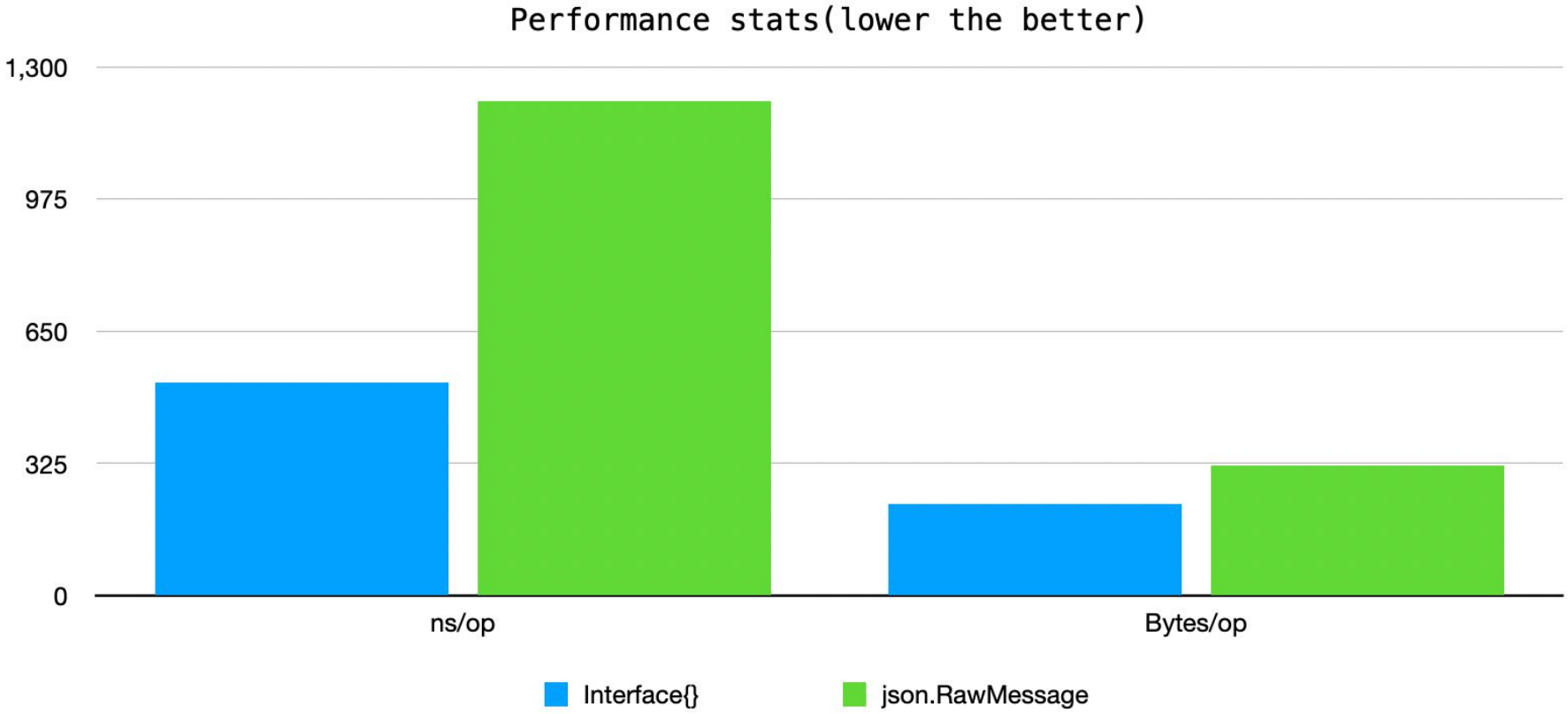
```go
package main

import (
    "encoding/json"
    "testing"
)

type rawResult struct {
    ID     string          `json:"subscription"`
    Result json.RawMessage `json:"result,omitempty"`
}

type interfaceResult struct {
    ID     string      `json:"subscription"`
    Result interface{} `json:"result,omitempty"`
}

type rawMessage struct {
    Version string          `json:"jsonrpc,omitempty"`
    ID      json.RawMessage `json:"id,omitempty"`
    Method  string          `json:"method,omitempty"`
    Params  json.RawMessage `json:"params,omitempty"`
    Result  json.RawMessage `json:"result,omitempty"`
}

type interfaceMessage struct {
    Version string          `json:"jsonrpc,omitempty"`
    ID      json.RawMessage `json:"id,omitempty"`
    Method  string          `json:"method,omitempty"`
    Params  interface{}     `json:"params,omitempty"`
    Result  interface{}     `json:"result,omitempty"`
}
```



```go
package main

import (
    "encoding/json"
    "testing"
)

type rawResult struct {
    ID     string          `json:"subscription"`
    Result json.RawMessage `json:"result,omitempty"`
}

type interfaceResult struct {
    ID     string      `json:"subscription"`
    Result interface{} `json:"result,omitempty"`
}

type rawMessage struct {
    Version string          `json:"jsonrpc,omitempty"`
    ID      json.RawMessage `json:"id,omitempty"`
    Method  string          `json:"method,omitempty"`
    Params  json.RawMessage `json:"params,omitempty"`
    Result  json.RawMessage `json:"result,omitempty"`
}

type interfaceMessage struct {
    Version string          `json:"jsonrpc,omitempty"`
    ID      json.RawMessage `json:"id,omitempty"`
    Method  string          `json:"method,omitempty"`
    Params  interface{}     `json:"params,omitempty"`
    Result  interface{}     `json:"result,omitempty"`
}
```

```go
type Header struct {
    Number int
}

func BenchmarkInterface(b *testing.B) {
    header := Header{Number: 1}
    for i := 0; i < b.N; i++ {
        params := interfaceResult{ID: "id", Result: header}
        msg := interfaceMessage{
            Version: "2.0",
            Method:  "rpc_test",
            Params:  params,
        }
        json.Marshal(msg)
    }
}

func BenchmarkRaw(b *testing.B) {
    header := Header{Number: 1}
    for i := 0; i < b.N; i++ {
        enc, _ := json.Marshal(header)
        params, _ := json.Marshal(&rawResult{ID: "id", Result: enc})
        msg := rawMessage{
            Version: "2.0",
            Method:  "rpc_test",
            Params:  params,
        }
        json.Marshal(msg)
    }
}
```


$ go test -bench=. -benchmem -run Benchmark.* -v
goos: darwin
goarch: arm64
pkg: github.com/ethereum/go-ethereum/cmd/geth
cpu: Apple M1
BenchmarkInterface
BenchmarkInterface-8 2358319 524.1 ns/op 224 B/op 3 allocs/op
BenchmarkRaw
BenchmarkRaw-8 1000000 1215 ns/op 320 B/op 5 allocs/op
PASS
ok github.com/ethereum/go-ethereum/cmd/geth 3.869s


Performance stats(lower the better)
1,300
975
650
325
0
ns/op
Bytes/op
Interface{}
json.RawMessage

PR: ethereum/go-ethereum#28328

27 rpc/subscription.go

```
@@ -105,7 +105,7 @@ type Notifier struct {
105 105
106 106      mu            sync.Mutex
107 107      sub           *Subscription
108     -    buffer        []json.RawMessage
    108 +    buffer        []any
109 109      callReturned bool
110 110      activated     bool
111 111  }
@@ -129,12 +129,7 @@ func (n *Notifier) CreateSubscription() *Subscription {
129 129
130 130  // Notify sends a notification to the client with the given data as payload.
131 131  // If an error occurs the RPC connection is closed and the error is returned.
132     - func (n *Notifier) Notify(id ID, data interface{}) error {
133     -     enc, err := json.Marshal(data)
134     -     if err != nil {
135     -         return err
136     -     }
137     -
    132 + func (n *Notifier) Notify(id ID, data any) error {
138 133      n.mu.Lock()
139 134      defer n.mu.Unlock()
140 135
@@ -144,9 +139,9 @@ func (n *Notifier) Notify(id ID, data interface{}) error {
144 139          panic("Notify with wrong ID")
145 140      }
146 141      if n.activated {
147     -         return n.send(n.sub, enc)
    142 +         return n.send(n.sub, data)
148 143      }
```


Thank you!
@jsvisa
Ambergroup