<03/18/23>

Deep dive into runtime features provided by Go1.22

Takuma Shibuya

CyberAgent Inc.

sivchari

- Takuma Shibuya
  - sivchari
- CIU
  - AKE
- Go Next Experts
- Go Conference Host

SECTION ONE

runtimeパッケージの変更点

SECTION ONE

runtime/metrics

- 4つのメトリクスが追加 (#63340)
  - /sched/pauses/stopping/gc:seconds
  - /sched/pauses/stopping/other:seconds
  - /sched/pauses/total/gc:seconds
  - /sched/pauses/total/other:seconds
- /runtime/proc.go
  - GCMark or GCTerm以外はOtherとしてMetricが加算される
  - /gc/pauses:secondsはDeprecated
  - /sync/mutex/wait/total:secondsにinternal lockも含まれる

SECTION TWO

runtime/pprof

- mutexのプロファイル記録が変更
  - 100個のgoroutineが10 millisecずつ遅延
  - 10 millisecでの記録ではなく1secとして記録される
  - より正確にボトルネックの度合いを表現
- Internal mutex lockはruntime._LostContendedRuntimeLockで報告される
- Darwinプラットフォームの CPUプロファイルにメモリマップが入り、逆アセンブルができるようになった

SECTION THREE

runtime/trace

- runtime/traceの抱えていたいくつかの課題が解決された
- ほとんどのプラットフォームで os.clockを使うようになった
- traceがパーティショニングされるようになったためストリーマブルな処理が可能になった
- traceにsystem callの継続時間やスレッド周りの情報が入った

SECTION THREE

runtime/trace

- 実験的にこれらを利用した trace pacakgeであるexp/traceが公開された
  - 現時点では1.22でしか動かない
  - GOEXPERIMENT=noexectracer2で古い実装に戻せる
- trace実行と終了のレイテンシが大幅に減少
- GCのMarkphaseの間に開始と終了ができるようになった
- https://blog.felixge.de/reducing-gos-execution-tracer-overhead-with-frame-pointer-unwinding/

SECTION THREE

runtime/trace

- G, M, Pの中でPに紐づけたバッファにイベントを書き込む
  - HACKING.md
  - GやMに比べて一般的に Pは少ないためバッファの数を最小にできる
    - Preemption/IO Wait
- 同期を考えるとSTWなどの際にPsを利用することが有用である

SECTION THREE

traceを触ってみる

- https://go.dev/blog/execution-traces-2024
  - Design Doc
- Go1.21までのtraceの問題点
  - 高いオーバーヘッド
  - 適切にスケールできず分析するにはでかすぎる
  - 特定の悪い挙動を補足する時にいつトレースを開始すればいいかが不明瞭
  - 実行トレースの解析パッケージがなかったためプログラム上での分析が困難

SECTION THREE

まとめ

- runtime周りの複数の改善によってメトリクスまわりなどに改修が入った
- pprofのmutextプロファイルは個人的にとても嬉しい
- Trace周りの変更は G,M,Pの取扱が絡むため深い理解にはHACKING.mdが必要そう
- FlightRecorderによりコード内部で G,M,Pが扱えるようになった
  - e.g. Middleware/GMPを扱えるのでそれを使ったツール
    - goroutine scheduler simulatorつくれるかやってみる
  - とはいえ使いどころはしっかりと考えよう