-grpool

Gogoroutine2KBJavaC++4mCPU4G1024Go4*1024*1024/2=200wGo

goroutineGCgoroutinegoroutine/grpoolgoroutine100Wgoroutine100Wgoroutinegrpoolgoroutine

goroutine/CPUgoroutinegoroutineGogoroutineGo

Go

510010000

Pool

goroutinegoroutine

Job

FuncJobWorkerFunc

```
type Func func(ctx context.Context)
```

Worker

goroutineWorkerJobJob

```
import "github.com/gogf/gf/v2/os/grpool"
```

Content Menu

```
func Add(ctx context.Context, f Func) error
func AddWithRecover(ctx context.Context, userFunc Func, recoverFunc
RecoverFunc) error
func Jobs() int
func Size() int
func New(limit ...int) *Pool
    func (p *Pool) Add(ctx context.Context, f Func) error
    func (p *Pool) AddWithRecover(ctx context.Context, userFunc Func,
recoverFunc RecoverFunc) error
    func (p *Pool) Cap() int
    func (p *Pool) Close()
    func (p *Pool) IsClosed() bool
    func (p *Pool) Jobs() int
    func (p *Pool) Size() int
```

grpool.Newgoroutinelimitgoroutinegoroutine Size()goroutine Jobs()

grpoolgoroutinegoroutinegrpool.Add func()/

> ⚠ grpool2

goroutine100goroutine1000

```
package main

import (
         "context"
         "fmt"
         "github.com/gogf/gf/v2/os/gctx"
         "github.com/gogf/gf/v2/os/grpool"
         "github.com/gogf/gf/v2/os/gtimer"
         "time"
)

var (
    ctx = gctx.New()
)

func job(ctx context.Context) {
         time.Sleep(1*time.Second)
}

func main() {
         pool := grpool.New(100)
         for i := 0; i < 1000; i++ {
             pool.Add(ctx,job)
         }
         fmt.Println("worker:", pool.Size())
         fmt.Println(" jobs:", pool.Jobs())
         gtimer.SetInterval(ctx,time.Second, func(ctx context.Context) {
             fmt.Println("worker:", pool.Size())
             fmt.Println(" jobs:", pool.Jobs())
             fmt.Println()
         })

         select {}
}
```

sleep 1goroutinegtime.SetInterval1goroutine

```
package main

import (
         "context"
         "fmt"
         "github.com/gogf/gf/v2/os/gctx"
         "github.com/gogf/gf/v2/os/grpool"
         "sync"
)

var (
    ctx = gctx.New()
)

func main() {
         wg := sync.WaitGroup{}
         for i := 0; i < 10; i++ {
            wg.Add(1)
            grpool.Add(ctx,func(ctx context.Context) {
                      fmt.Println(i)
                      wg.Done()
            })
         }
         wg.Wait()
}
```

0-9

```
10
10
10
10
10
10
10
10
10
10
```

gogrpool/(goroutinei)ii10 iii

1)go

```
package main

import (
    "fmt"
    "sync"
)

func main() {
    wg := sync.WaitGroup{}
    for i := 0; i < 10; i++ {
        wg.Add(1)
        go func(v int){
            fmt.Println(v)
            wg.Done()
        }(i)
    }
    wg.Wait()
}
```

```
0
9
3
4
5
6
7
8
1
2
```

2)

```
package main

import (
         "context"
         "fmt"
         "github.com/gogf/gf/v2/os/gctx"
         "github.com/gogf/gf/v2/os/grpool"
         "sync"
)

var (
   ctx = gctx.New()
)

func main() {
         wg := sync.WaitGroup{}
         for i := 0; i < 10; i++ {
            wg.Add(1)
            v := i
            grpool.Add(ctx, func(ctx context.Context) {
                      fmt.Println(v)
                      wg.Done()
            })
         }
         wg.Wait()
}
```

```
9
0
1
2
3
4
5
6
7
8
```

grpoolfunc(ctx context.Context)ictxi

goroutineAddWithRecover

AddWithRecoveruserFuncpanicRecovery FuncRecovery FuncuserFuncpanic

```
package main

import (
        "context"
        "fmt"
        "github.com/gogf/gf/v2/container/garray"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/grpool"
        "time"
)

var (
        ctx = gctx.New()
)
func main() {
        array := garray.NewArray(true)
        grpool.AddWithRecover(ctx, func(ctx context.Context) {
                array.Append(1)
                array.Append(2)
                panic(1)
        }, func(err error) {
                array.Append(1)
        })
        grpool.AddWithRecover(ctx, func(ctx context.Context) {
                panic(1)
                array.Append(1)
        })
        time.Sleep(500 * time.Millisecond)
        fmt.Print(array.Len())
}
```

grpoolgoroutine

1)grpool

```
package main

import (
         "context"
         "fmt"
         "github.com/gogf/gf/v2/os/gctx"
         "github.com/gogf/gf/v2/os/grpool"
         "github.com/gogf/gf/v2/os/gtime"
         "sync"
         "time"
)

var (
   ctx = gctx.New()
)

func main() {
         start := gtime.TimestampMilli()
         wg := sync.WaitGroup{}
         for i := 0; i < 10000000; i++ {
            wg.Add(1)
            grpool.Add(ctx,func(ctx context.Context) {
                      time.Sleep(time.Millisecond)
                      wg.Done()
            })
         }
         wg.Wait()
         fmt.Println(grpool.Size())
         fmt.Println("time spent:", gtime.TimestampMilli() - start)
}
```

2)goroutine

```
package main

import (
         "fmt"
         "github.com/gogf/gf/v2/os/gtime"
         "sync"
         "time"
)


func main() {
         start := gtime.TimestampMilli()
         wg := sync.WaitGroup{}
         for i := 0; i < 10000000; i++ {
            wg.Add(1)
            go func() {
                      time.Sleep(time.Millisecond)
                      wg.Done()
            }()
         }
         wg.Wait()
         fmt.Println("time spent:", gtime.TimestampMilli() - start)
}
```

3)

3

```
grpool:
    goroutine count: 847313
     memory   spent: ~2.1 G
     time     spent: 37792 ms

goroutine:
    goroutine count: 1000W
    memory    spent: ~4.8 GB
    time      spent: 27085 ms
```

goroutineCPU