Grouk

并发之痛

Thread，Goroutine，Actor

王渊命
Grouk
@jolestar

Grouk

个人介绍

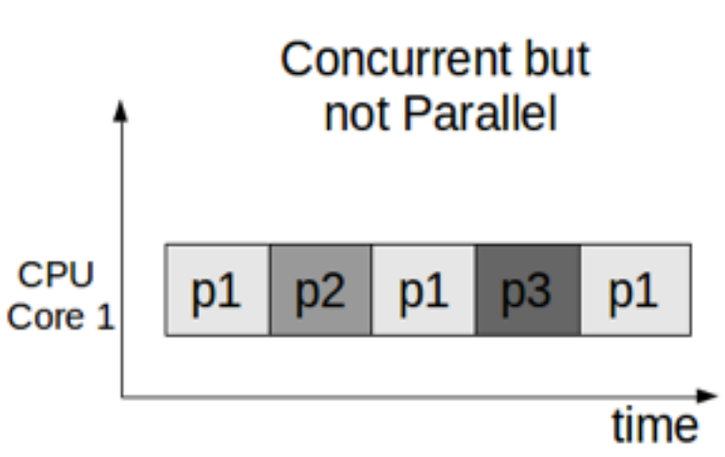
- 王渊命 jolestar Go语言”爱好者”
- 新浪微博
- 新浪微米
- Grouk

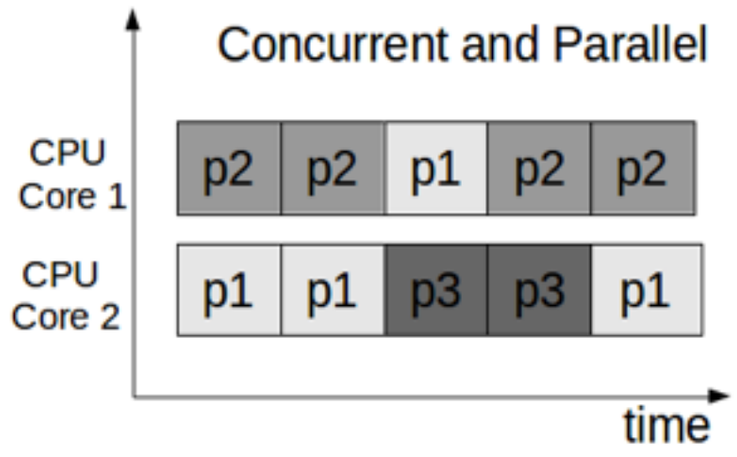
调查

- 多少人吃过并发程序的亏?
- 多少人是也同时会Java?

并发与并行

- concurrency
- parallelism





https://nikolaygrozev.wordpress.com/2015/07/14/overview-of-modern-concurrency-and-parallelism-concepts/

http://blog.golang.org/concurrency-is-not-parallelism

Grouk

为什么并发程序难?

We believe that writing correct **concurrent**, **fault-tolerant** and **scalable** applications is too hard. Most of the time it's because we are using the wrong tools and the wrong level of abstraction.

———— Akka

Grouk

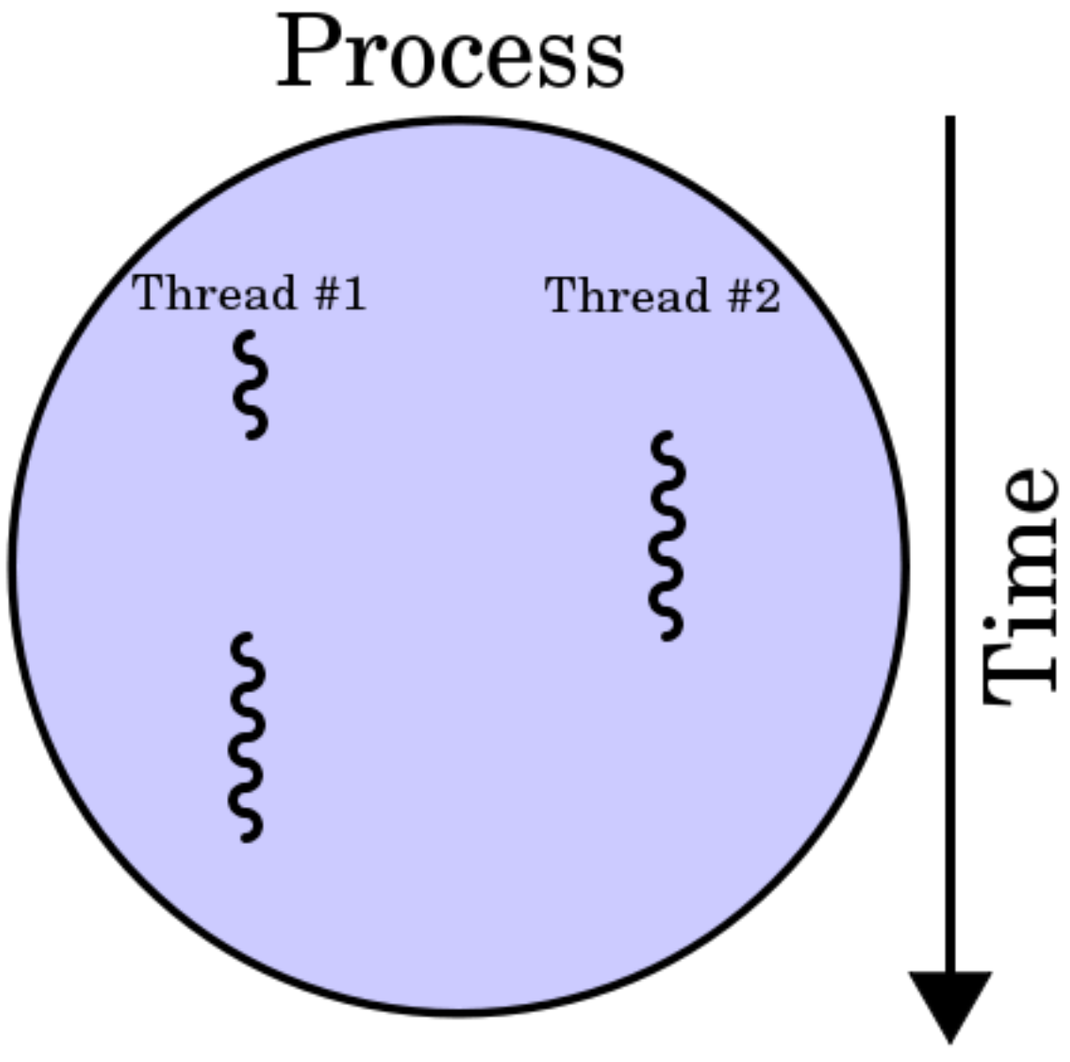
正确的工具和抽象?

从头来看

- 面向过程 数据 + func
- 面向对象 对象（数据 + func）
- 谁来执行func？ cpu — 进程/线程

线程（Thread）

- 系统内核
- 调度器
- 资源共享（同一进程下）



https://en.wikipedia.org/wiki/Thread_(computing)

线程之痛-复杂性

- 竞态条件（race conditions）

- 依赖关系以及执行顺序

线程之痛-复杂性

- Mutex(Lock)　(Go sync, Java concurrent)
- semaphore
- volatile
- compare-and-swap　(Atomic)

线程之痛-应该用多少线程

给孩子喂多少饭合适?

给孩子喂多少饭合适?

- 孩子不吃了就好了
- 孩子吃饱了就好了
- 逐渐增量，长期观察，然后计算一个平均值
- 孩子吃吐了就别喂了
- 没控制好边界，把孩子给给撑坏了

Grouk

教孩子说话，自己吃饭，自管理

计算机不会自己说话，如何自管理?

应该有多少线程?

- 内存（线程的栈空间）
  Java的栈空间（64位VM）默认是1024k

- 调度成本（context-switch）
  个人电脑线程切换成本大约 6000纳秒
  （非严格测试）

- CPU利用率

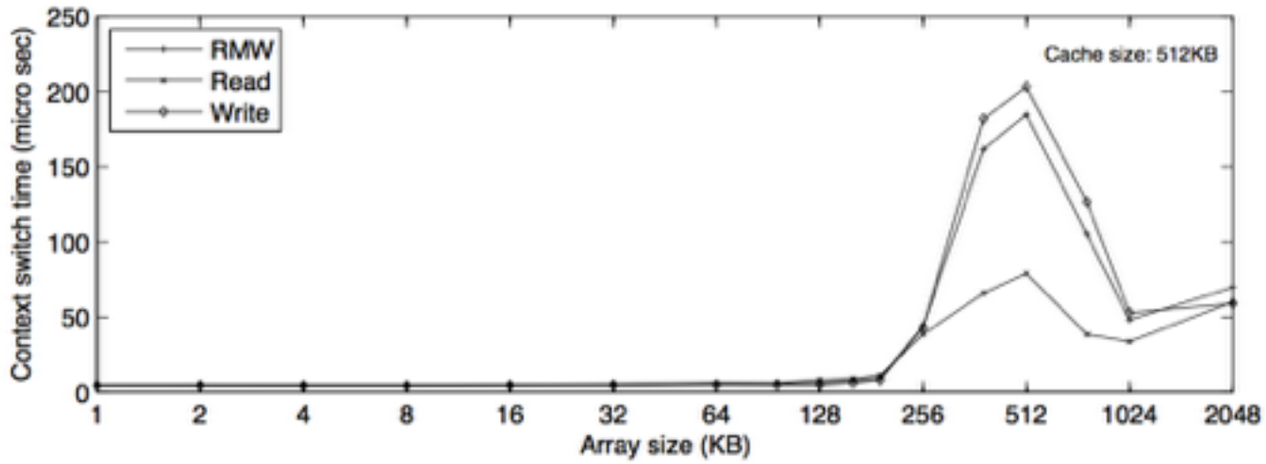


Figure 1: The effect of data size on the cost of the context switch

Quantifying The Cost of Context Switch - www.cs.rochester.edu

CPU利用率

100毫秒的时间片，该线程实际使用运算15毫秒，等待网络80毫秒，解析和处理结果返回5毫秒，我们有4核CPU，应该多少线程合适?

$$100/(15+5)*4=20$$

应该有多少线程?

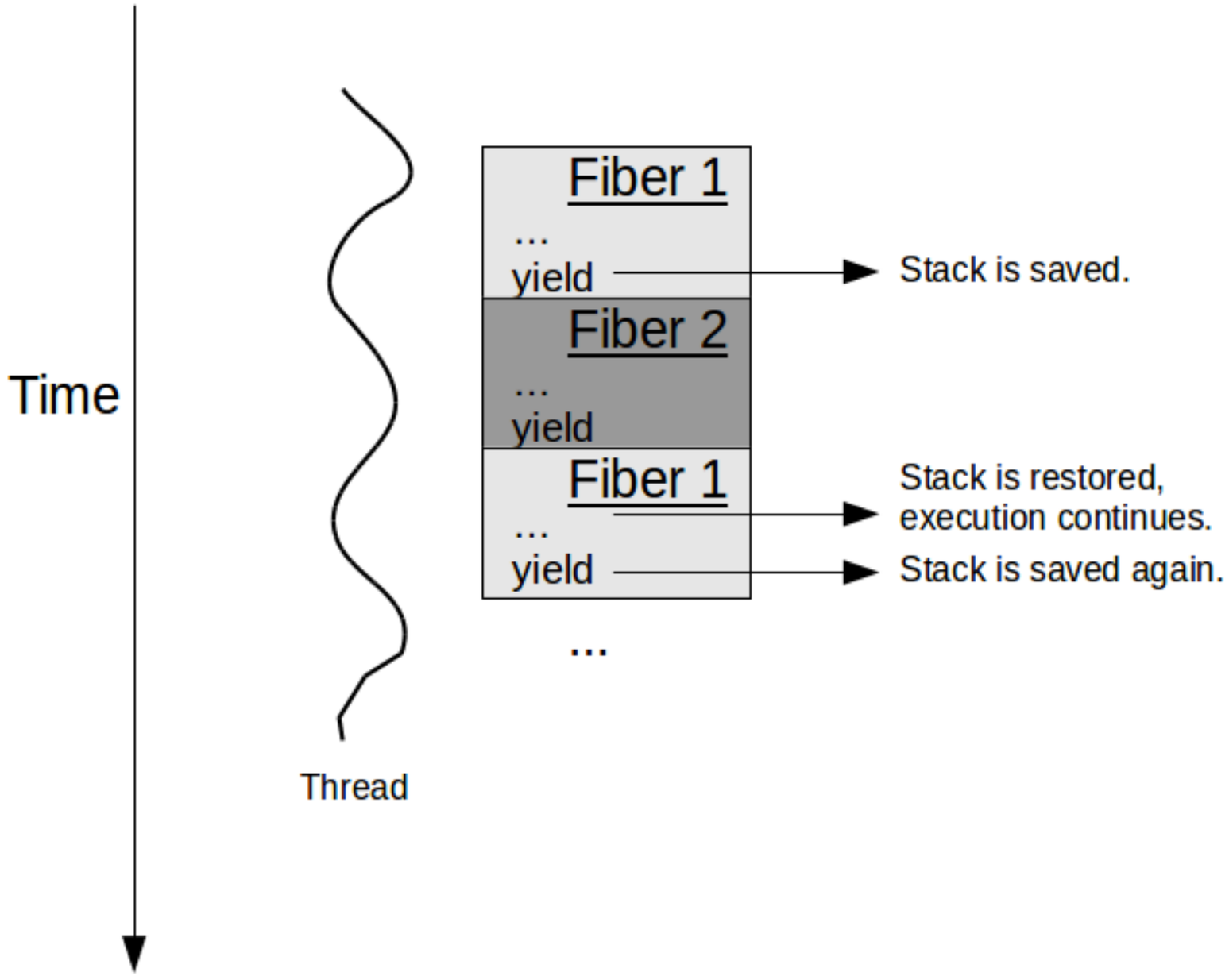
- 线程的成本较高（内存，调度）不可能大规模创建
- 应该由语言或者框架动态解决这个问题

线程池模式

- 一定程度上控制了线程的数量，实现了复用，降低了线程的使用成本
- 没有解决数量的问题
- 不同的任务可能有不同的并发需求，为了避免互相影响可能需要多个线程池

陈力就列，不能者止

让阻塞的代码片段让出位置



How?

- 异步回调（NodeJS）
- GreenThread

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

callback hell

GreenThread

- 用户空间
- 语言或者框架层调度
- 更小的栈空间允许创建大量实例（百万级别）

Grouk

几个概念

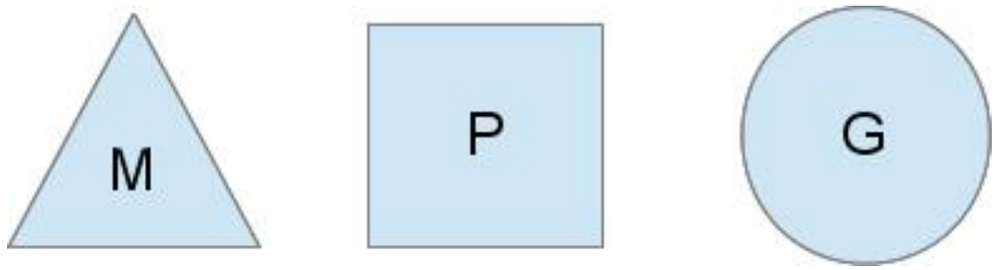
- Continuation
- Coroutine
- Fiber

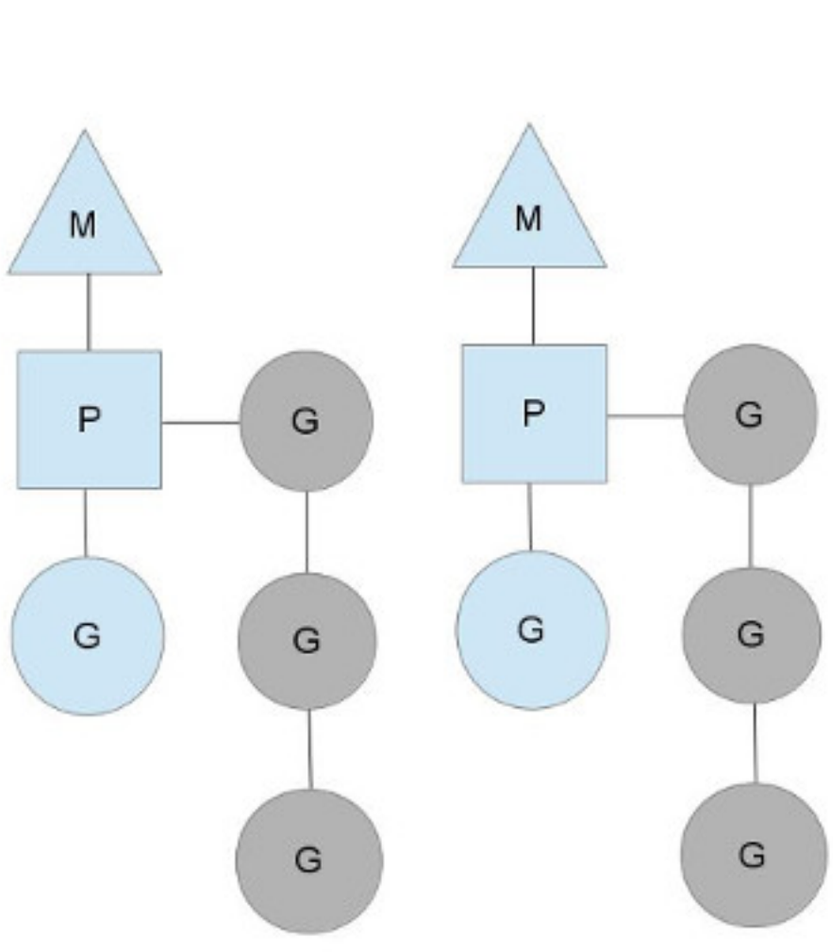
Grouk

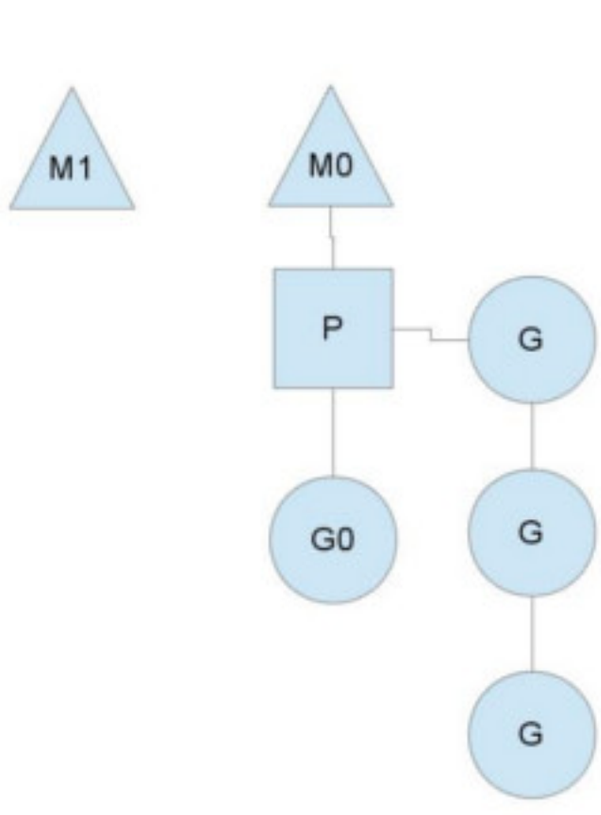
Goroutine

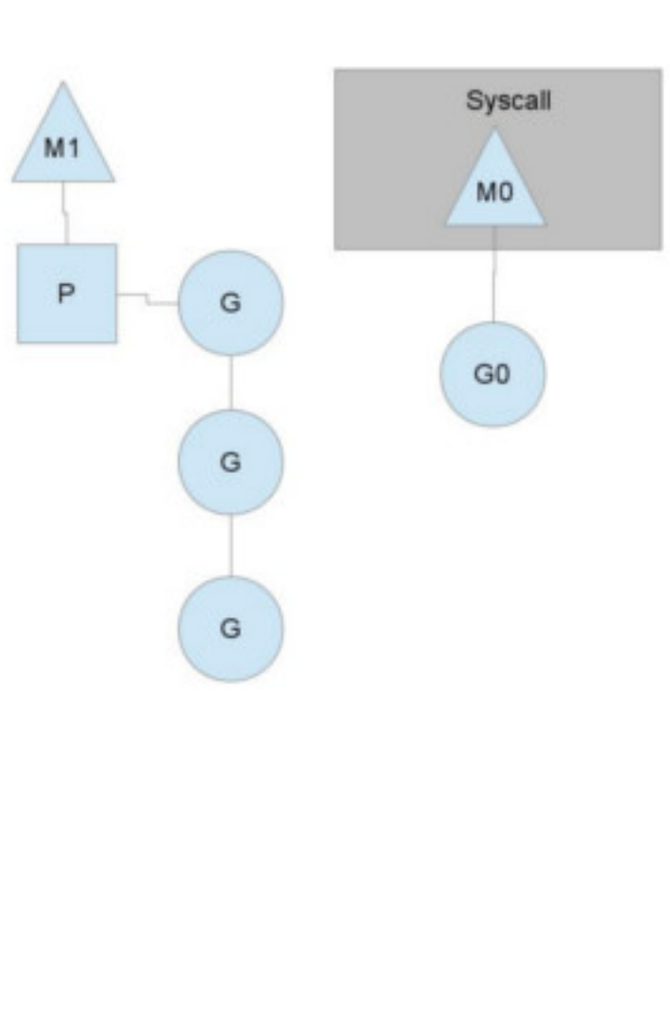
- Coroutine
- 调度器
- Channel

Goroutine — 调度器









https://morsmachine.dk/go-scheduler

Grouk

Goroutine — Channel

Do not communicate by sharing memory; instead, share memory by communicating.

- SynchronousQueue
- ArrayBlockQueue

Goroutine 银弹?

- 其他资源出现瓶颈如何处理？ (Goroutine池，需要多少个Goroutine?)
- 互联网在线应用的响应时间限制
- 依然把难题扔给调用方

Actor

- Processing – actor可以做计算的，不需要调用方控制
- Storage – actor可以保存状态
- Communication –actor之间可以通过发送消息通讯

Actor

- 发送消息给其他的Actor
- 创建其他的Actor
- 接受并处理消息，修改自己的状态

Grouk

Actor

- Actor独立更新
- 无缝弥合本地和远程调用
- 容错
- 扩展 分布式

Grouk

Actor

- Erlang OTP
- Akka (Scala,Java)
- Quasar (Java)

Actor 与 Goroutine

- 抽象层面不一样
- 解决的问题范围不一样

结论

革命尚未成功

同志任需努力

构想:在Goroutine上实现Actor?

- 分布式
- self-manage
- 和容器集群融合

Grouk

Q&A

王渊命
@jolestar
https://grouk.com