Atitit 提升性能 提升线程数 协程 Quasar go goroutine

## Go的goroutine

## Fiber与jdbc对问题

IntStream**.**range**(**0**,** 100000**).**forEach**(** **new** Fiber**(()** **->** **{**

System**.**out**.**println**(**"This is a fiber not thread"**);**

Fiber**.**sleep**(...);** *// NOT Thread.sleep() !!!*  **}).**start**()** **);**

以上这个例子我们可以看到,创建了100k个协程, 打印输出,然后sleep() 如果把Fiber换成Thread类,这段代码几乎是无法运行的。

看上去它只需把Thread替换为Fiber, 就可以方便地从线程切换到协程来支持高并发。遗憾的是,这样的协程库,它只能**部分支持**在协程中调用阻塞API, 因为它有个前置条件:**在协程里,所有调用的阻塞API必须换成协程的版本。**

仔细再看刚才的例子,我们除了在创建协程时把new Thread()换成了new Fiber(), 还把Thread.sleep()换成了Fiber.sleep()才实现了真正协程的功能。

由于这个限制, 我们不能随心所欲地调用阻塞API,而是必须先找到对应的API的协程版本,并替换之, 所以, 我们需要使用类似于FiberSocket,FiberServerSocket等相应的类。此外,由于无法把三方库(比如JDBC)内部的阻塞API实现替换掉, 所以就产生了另一个问题: **协程库无法支持调用有阻塞API的三方库。**

鉴于以上调研, 我们需要考虑在三方库和协程库之间做一定的取舍。

# Quasar prblm

[quasar] ERROR: while transforming jdk/internal/loader/URLClassPath

版本问题，，提升jdk为jdk8就ok ，，quasar版本为0.7.10

# **monitor**

可以通过JMX监控fiber的运行状态，work queue的堆积,fiber的数量，调度延迟等

## **FJP worker运行时如果有疑似blocking,会有WARNING hogging the CPU or blocking a thread**

you can disable the warning by setting a system property with "-Dco.paralleluniverse.fibers.detectRunawayFibers=false

## Quasar最主要的贡献就是提供了轻量级线程的实现，叫做fiber(纤程)。

Fiber的功能和使用类似Thread, API接口也类似，所以使用起来没有违和感，但是它们不是被操作系统管理的，它们是由一个或者多个ForkJoinPool调度。一个idle fiber只占用400字节内存，切换的时候占用更少的CPU，你的应用中可以有上百万的fiber，显然Thread做不到这一点。这一点和Go的goroutine类似。

以上两条是选择fiber还是thread的判断条件，主要还是看任务是I/O blocking相关还是CPU相关。幸运地是，fiber API使用和thread使用类似，所以代码略微修改久就可以兼容。

Fiber特别适合替换哪些异步回调的代码。使用FiberAsync异步回调很简单，而且性能很好，扩展性也更高。

似Thread, fiber也是用Fiber类表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | new Fiber<V>() {  @Override  protected V run() throws SuspendExecution, InterruptedException {  // your code  }  }.start(); |

与Thread类似，但也有些不同。Fiber可以有一个返回值,类型为泛型V，也可以为空Void。run也可以抛出异常InterruptedException。

你可以传递SuspendableRunnable 或 SuspendableCallable 给Fiber的构造函数:

|  |  |
| --- | --- |
| 1  2  3  4  5 | new Fiber<Void>(new SuspendableRunnable() {  public void run() throws SuspendExecution, InterruptedException {  // your code  }  }).start(); |

甚至你可以调用Fiber的join方法等待它完成，调用get方法得到它的结果。

Fiber继承Strand类。Strand类代表一个Fiber或者Thread，提供了一些底层的方法。

fiber中的ThreadLocal是fiber local的。InheritableThreadLocal继承父fiber的值。

Fiber、SuspendableRunnable 、SuspendableCallable 的run方法会抛出SuspendExecution异常。但这并不是真正意义的异常，而是fiber内部工作的机制，通过这个异常暂停因block而需要暂停的fiber。

任何在Fiber中运行的方法，需要声明这个异常(或者标记@Suspendable),都被称为suspendable method。

反射调用通常都被认为是suspendable, Java8 lambda 也被认为是suspendable。不应该将类构造函数或类初始化器标记为suspendable。

synchronized语句块或者方法会阻塞操作系统线程，所以它们不应该标记为suspendable。Blocking线程调用默认也不被quasar允许。但是这两种情况都可以被quasar处理，你需要在Quasar javaagent中分别加上m和b参数，或者ant任务中加上allowMonitors和allowBlocking属性。

### **性能对比**

刘小溪在CSDN上写了一篇关于Quasar的文章：[次时代Java编程（一）：Java里的协程](http://geek.csdn.net/news/detail/71824" \t "https://colobu.com/2016/07/14/Java-Fiber-Quasar/_blank)，写的挺好，建议读者读一读。

它参考[Skynet](https://github.com/atemerev/skynet" \t "https://colobu.com/2016/07/14/Java-Fiber-Quasar/_blank)的测试写了代码进行对比，这个测试是并发执行整数的累加：  
测试结果是Golang花了261毫秒，Quasar花了612毫秒。其实结果还不错，但是文中指出这个测试没有发挥Quasar的性能。因为quasar的性能主要在于阻塞代码的调度上。  
虽然文中加入了排序的功能，显示Java要比Golang要好，但是我觉得这又陷入了另外一种错误的比较， Java的排序算法使用TimSort，排序效果相当好，Go的排序效果显然比不上Java的实现，所以最后的测试主要测试排序算法上。 真正要体现Quasar的性能还是测试在有阻塞的情况下fiber的调度性能。

虽然使用Go可以解决这个问题，但是对于系统A的改造比较大，还增加了系统的复杂性。Netty性能好，改动量还可以接受，但是不妨看一下这个场景，系统的问题是由http阻塞引起。

这正是Quasar fiber适合的场景，如果一个Fiber被阻塞，它可以暂时放弃线程，以便线程可以用来执行其它的Fiber。虽然整个集成系统的吞吐率依然很低，这是无法避免的，但是系统的吞吐率确很高。

综上所述，如果想彻底改造系统A,则可以使用Go库重写，或者使用Netty + Rx的方式去处理，都能达到比较好的效果。如果想改动比较小，可以考虑使用quasar替换线程对代码进行维护。

# 协程的方式更多用来做阻塞密集型（比如 I/O）的操作，计算密集型的还是使用线程更加合理。