## [# Rust异步网络编程](https://www.cnblogs.com/liufu627/p/12562660.html)

Posted on 2020-03-24 22:29  [liufu627](https://www.cnblogs.com/liufu627/)  阅读(1910)  评论(0)  [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=12562660)  [收藏](javascript:void(0))  [举报](javascript:void(0))

# Rust异步网络编程

Rust的高性能异步网络编程模式目前是基于mio和futures这两个库构建的生态。

Tokio则连接这2个库构建了一个异步非阻塞事件驱动编程平台。

# 什么是 mio,futures,tokio

## 1- Mio

Mio是Rust的轻量级快速低级IO库，专注于非阻塞API,事件通知以及用于构建高性能IO应用程序的其他有用实用程序.

## 特征

快速 - 相当于OS设施级别的最小开销（epoll，kqueue等..）

非阻塞TCP，UDP。

由epoll，kqueue和IOCP支持的I/O事件通知队列。

运行时零分配

平台特定扩展。

平台支持

Linux

OS X

Windows

FreeBSD

NetBSD

Solaris

Android

iOS

Fuchsia (experimental)

## 2- futures

Rust中的零成本异步编程库,Futures可在没有标准库的情况下工作，例如在裸机环境中。

提供了许多用于编写异步代码的核心抽象：

> Future是由异步计算产生的单一最终值。一些编程语言（例如JavaScript）将此概念称为“promise”。

> Streams表示异步生成的一系列值。

> Sinks支持异步写入数据。

> Executors负责运行异步任务。

还包含异步I/O和跨任务通信的抽象。

所有这些是任务系统的基础，它是轻量级线程(协程)的一种形式。使用Future，Streams和Sinks构建大型异步计算，然后将其生成作为独立完成的任务运行，但不阻塞运行它们的线程。

## 3- Tokio

Tokio : Rust编程语言的异步运行时,提供异步事件驱动平台，构建快速，可靠和轻量级网络应用。利用Rust的所有权和并发模型确保线程安全

> 基于多线程，工作窃取的任务调度程序。

> 一个反应器操基于作系统的事件队列（epoll的，kqueue的，IOCP等）的支持。

> 异步TCP和UDP套接字。

这些组件提供构建异步应用程序所需的运行时组件。

## 快速

Tokio构建于Rust之上，提供极快的性能，使其成为高性能服务器应用程序的理想选择。

1. 零成本抽象

与完全手工编写的等效系统相比，Tokio的运行时模型不会增加任何开销。

使用Tokio构建的并发应用程序是开箱即用的。Tokio提供了针对异步网络工作负载调整的多线程，工作窃取任务调度程序。

2. 非阻塞I/O

Tokio由操作系统提供的非阻塞，事件I/O堆栈提供支持。

## 可靠

虽然Tokio无法阻止所有错误，但它的目的是最小化它们。Tokio在运送关键任务应用程序时带来了安心。

1. 所有权和类型系统

Tokio利用Rust的类型系统来提供难以滥用的API。

2.  Backpressure

Backpressure开箱即用，无需使用任何复杂的API。

3. 取消

Rust的所有权模型允许Tokio自动检测何时不再需要计算。Tokio将自动取消它而无需用户调用cancel函数。

## 轻量级

Tokio可以很好地扩展，而不会增加应用程序的开销，使其能够在资源受限的环境中茁壮成长。

1. 没有垃圾收集器

因为Tokio使用Rust，所以不包括垃圾收集器或其他语言运行时。

2. 模块化

Tokio是一个小组件的集合。用户可以选择最适合手头应用的部件，而无需支付未使用功能的成本。