RUST 调研报告

一、简介

Rust 最初是由 Mozilla 研究院的 Graydon Hoare 设计创造，然后在 Dave Herman, Brendan Eich 以及很多其他人的贡献下逐步完善的。Rust 的设计者们通过在研发 Servo 网站浏览器布局引擎过程中积累的经验优化了 Rust 语言和 Rust 编译器。

从发布起，Rust 受到了广泛的关注，连续七年在 Stack Overflow 开发者调查的“最受喜爱编程语言”评选项目中折取桂冠。开发者对于 Rust 语言的关注度也在逐步提升（如下图）

Fig 3. Interest in Rust programming language over time
(May 2012-May 2022, generated from google trends) [18].

二、优势

1、Safety(language)：

一种变成语言的安全性主要归结于其防止和发现问题（如缓冲区越界）的能力，这类问题又主要归结于内存管理。

（1）Rust 使用**所有权系统**来决定内存的分配和释放，提高了运行的表现。简单来说：当一个变量被声明时，内存被分配；当变量不在范围内时，内存则被释放（好像变量的范围是内存的所有者）。通过禁止编程者直接手动管理内存，防止了 use after 和 double free 等漏洞。所有权系统将内存和变量的生命周期直接联系起来。

```
fn make_vec()
{
    // owned by make_vec's scope
    let mut vec = Vec::new();

    vec.push(0);
    vec.push(1);
    // scope ends, `vec` is destroyed
}
```

（2）内存在同一时间只能有一个所有者。这会导致函数之间无法共享参数的问题，这个问题通过一个**借阅系统**来实现，其规则如下：只能存在一个可变的参数或者多个不可变的参数。两种不能同时实现，即意味着最多只能有一线程修改内存且没有其他线程读取该内存。

（3）rust 还存在许多其他的安全特性，如：缓冲区访问的自动边界检查

2、Performance：

（1）Rust 使用零损耗抽象的概念，在不限制性能的同时简化了语言。例如：

A. 单形态化：允许用户创建通用化的函数，并在编译时转化为具体类型的函数。

B. 标准库函数：不需要重新创建通用类型，提供了性能和互用性更好的库。

（2）rust 没有内存垃圾收集器。垃圾收集器：

A. 垃圾收集器增加了内存和 cpu 的使用
B. 垃圾处理器很难控制，可能会导致意外的暂停
下图反映了六种语言的性能：

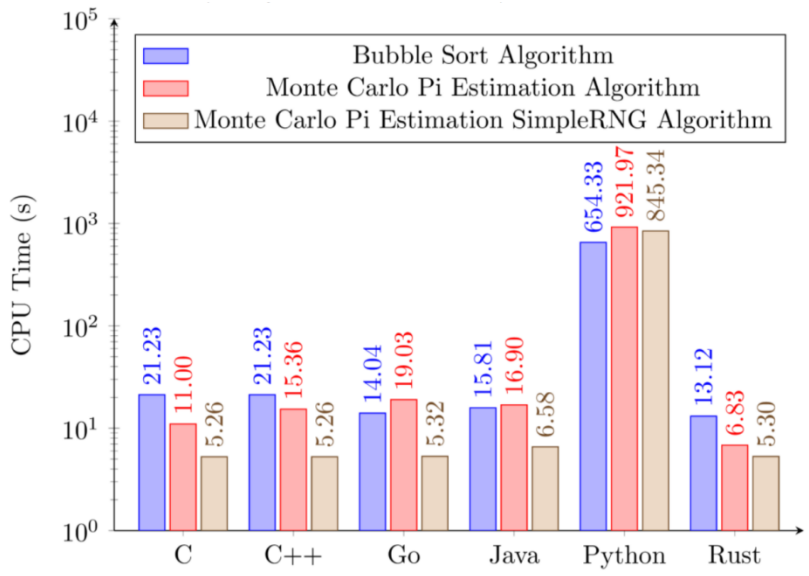

Fig 1. Average CPU time benchmark results [5].

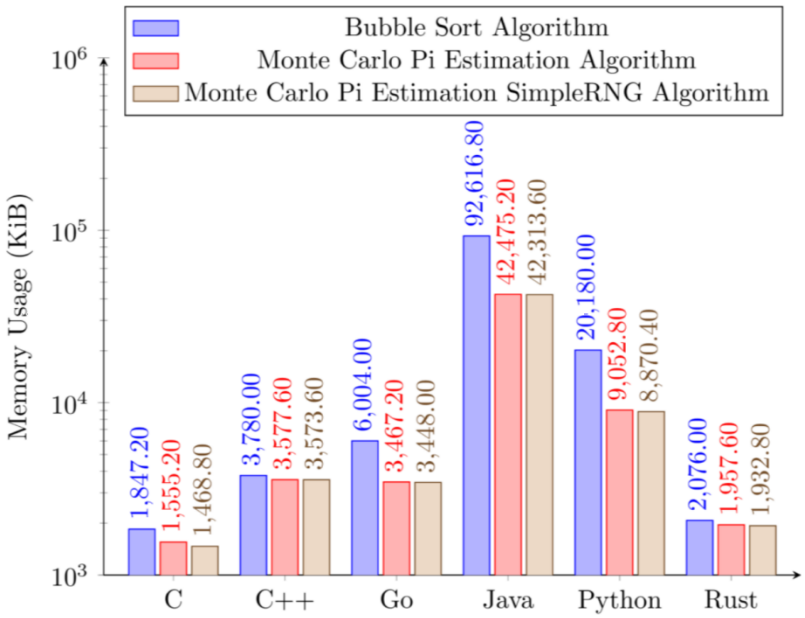

Fig 2. Average memory usage benchmark results [5].

3、Others
（1）隔离（isolation）零拷贝的软件故障隔离：
Rust 支持软件故障隔离（SFI）比任何主流语言都要低的开销。SFI 封装
软件中不受信任的扩展，且不依赖于硬件地址空间。
（2）分析（analysis）有效的静态信息流分析
Rust 实现精确高效的静态信息流量控制（IFC）。IFC 确保不受信任的模块不会通过未经授权的渠道泄露敏感数据,从而为其提供安全保障。

（3）自动化（automation）自动检查点

三、总结

综上。Rust 语言相较于传统 C 语言，有上述的诸多特性和优势，因此通过 rust 语言对 FreeRTOS 进行重写，将对其性能和安全性有巨大优化。

四、参考文献

[1]William Bugden, Ayman Alahmar. Rust: The programming language for safety and performance. *arXiv preprint arXiv:2206.05503, 2022*

[2]Abhiram Balasubramanian, Marek S Baranowski, Anton Burtsev, Aurojit Panda, Zvonimir Rakamarić, Leonid Ryzhyk .*Proceedings of the 16th workshop on hot topics in operating systems, 156-161, 2017*