探讨Rust中的Trait

苏林

今天公开课内容

1、介绍Trait的四种作用
2、通过事例来入门Trait
3、Trait静态分发——没有开销的抽象
4、Trait动态分发——有运行时开销(寻址过程)
5、性能测试

介绍Trait的四种作用

1、接口(interface)
2、类型标记 Copy、Sized, 通过类型标记编译器可以识别一些固定类型的行为.
3、泛型限定、入参 和 出参 作为泛型的限制.
4、抽象类型(trait object) 相当于动态语言调用. 静态调用, 动态调用|动态分发.

通过一个简单的完整例子来了解一下Trait

```rust
trait Animal {
    // Static method signature; `Self` refers to the implementor type.
    fn new(name: &'static str) -> Self;

    // Instance method signatures; these will return a string.
    fn name(&self) -> &'static str;
    fn noise(&self) -> &'static str;

    // Traits can provide default method definitions.
    fn talk(&self) {
        println!("{} says {}", self.name(), self.noise());
    }
}
```

通过一个简单的完整例子来了解一下Trait

```
15  struct Cat {
16      name: &'static str,
17      age: i32
18  }
19
20  struct Dog {
21      name: &'static str
22  }
```

通过一个简单的完整例子来了解一下Trait

```rust
impl Animal for Cat {
    // `Self` is the implementor type: `Cat`.
    fn new(name: &'static str) -> Cat {
        Cat { name: name, age: 1 }
    }

    fn name(&self) -> &'static str {
        self.name
    }

    fn noise(&self) -> &'static str {
        "Meowww"
    }

    // Default trait methods can be overridden.
    fn talk(&self) {
        // For example, we can add some quiet contemplation.
        println!("{} pauses briefly... {}", self.name, self.noise());
    }
}
```

静态分发

静态分发是通过泛型来实现的。Rust 编译器会使用具体的类型来生成非泛型版本的函数（monomorphization）。

静态分发是在编译期完成的，也就是说，静态分发的前提是要求能够在编译期便确定要调用的函数版本。

思考: rust单态化，会不会使编译出来的结果(文件)变大?

动态分发

动态分发发生在运行时。

Rust 中的动态分发 是通过所谓的“trait object”来实现的，Rust 会在运行时根据 trait object 中的指针来调用相应版本的函数。

trait object 的创建：指针（如，引用、智能指针）+ dyn 关键字 + 要实现的 trait（一个或多个，trait 之间使用 + 连接），如，&dyn Trait、Box<dyn Trait>。其意思是：只要一个类型实现了指定的 trait(s)，即满足该约束。

零成本抽象

性能测试

QA环节

加群一起交流Rust & Datafuse

Datafuse Rust语言中文社区 知数堂 zhishuteng.com