用Rust写Protobuf扩展

溪塔科技 宁志伟

Protobuf

- Google 出品的序列化框架
- 语言无关
- 体积小
- 速度快
- 扩展性好
- 与gRPC搭配
- 支持的语言多

CITA-Cloud

- CITA-Cloud 是一个以区块链技术为基础，融合云原生技术的柔性集成开放平台。
- 微服务
- 组件可替换
- 模块解耦
- 开源
- https://github.com/cita-cloud
- https://cita-cloud-docs.readthedocs.io/zh_CN/latest/

CITA-Cloud中的Protobuf

- 微服务间的gRPC接口
- 接口参数以及核心数据结构

- 抽象且通用-建模语言

协议分解

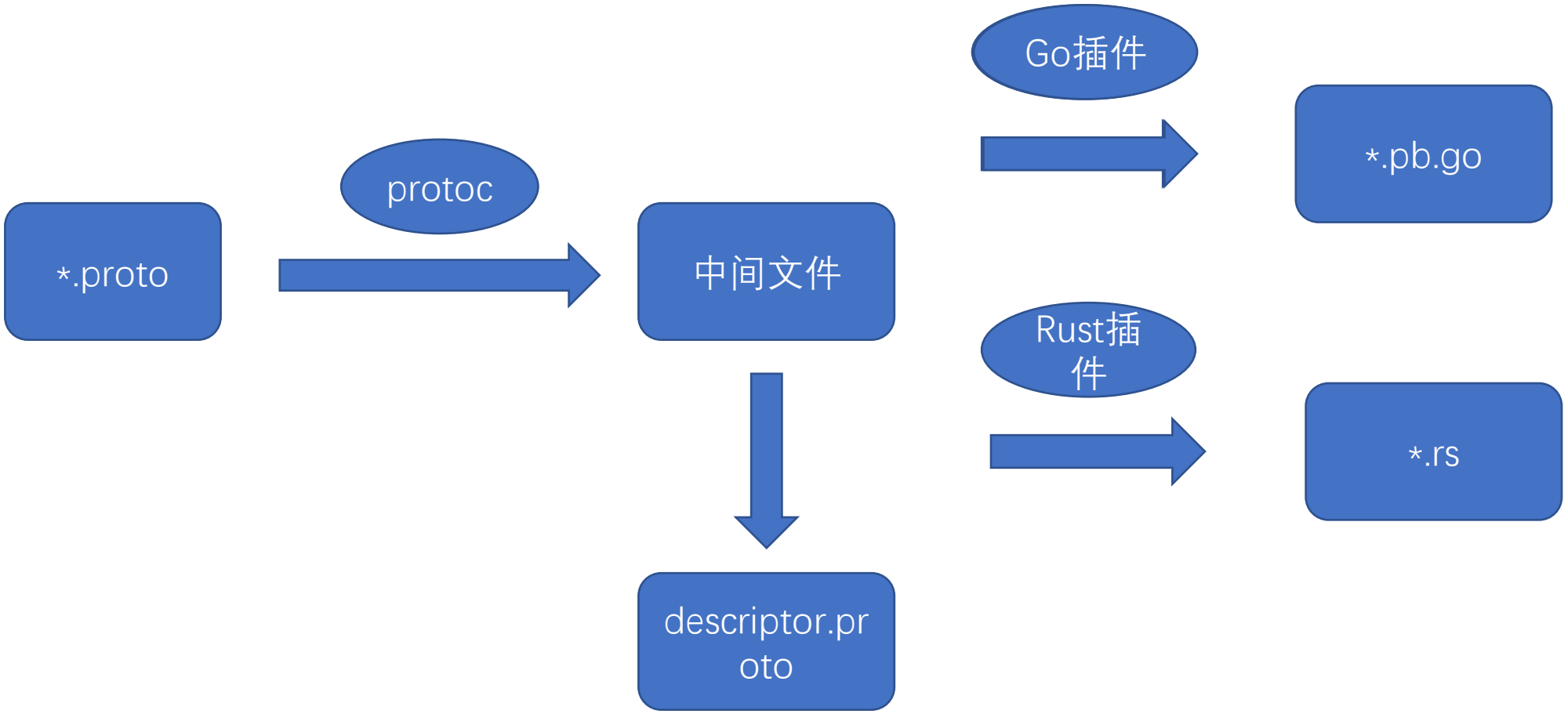
- Nonce——去重协议
- Valid_until_block——超时协议
- Transactions_root——轻节点协议

- 用户自定义协议的框架

```
5   message BlockHeader {
6       // hash of previous BlockHeader
7       bytes prevhash = 1;
8       uint64 timestamp = 2;
9       uint64 height = 3;
10      bytes transactions_root = 4;
11      bytes proposer = 5;
12  }
13
14  message Transaction {
15      uint32 version = 1;
16      // 1. length is 20 bytes for ev
17      // 2. if executor is multi-vm,
18      bytes to = 2;
19      // length is less than 128
20      string nonce = 3;
21      uint64 quota = 4;
22      uint64 valid_until_block = 5;
```

- https://github.com/cita-cloud/cita_cloud_proto

Protobuf扩展



Protobuf扩展

```
message Foo {
  // ...
  extensions 100 to 199;
}
```

```
extend Foo {
  optional int32 bar = 126;
}
```

```
message FieldOptions {
  // The ctype option instructs the C++ code gener
  // representation of the field than it normally
  // options below.  This option is not yet impleme
  // release -- sorry, we'll try to include it in
  optional CType ctype = 1 [default = STRING];
  enum CType {

  // Clients can define custom options
  extensions 1000 to max;

  reserved 4;  // removed jtype
}
```

```
syntax = "proto3";

package main;

import "google/protobuf/descriptor.proto";

extend google.protobuf.MessageOptions {
  optional string my_option = 51234;
}

extend google.protobuf.FieldOptions {
    string default_string = 50000;
    int32 default_int = 50001;
}

message TestMessage {
    option (my_option) = "hello world";
    string name = 1 [(default_string) = "gopher"];
    string no_option = 2;
    int32 age = 3 [(default_int) = 10];
}
```

Rust相关库

- Dropbox-pb-jelly
  - Python的codegen，拼Rust代码字符串
- rust-protobuf
  - 支持扩展和gRPC
  - 有些封装，但是还是在拼字符串
- Prost
  - 不支持扩展
  - Tonic-build支持gRPC
  - 使用derive宏简化生成的Rust代码
  - 使用quote库优雅的生成代码

例子

```rust
#[derive(Clone, PartialEq, ::prost::Message)]
pub struct BlockHeader {
    /// hash of previous BlockHeader
    #[prost(bytes, tag = "1")]
    pub prevhash: std::vec::Vec<u8>,
    #[prost(uint64, tag = "2")]
    pub timestamp: u64,
    #[prost(uint64, tag = "3")]
    pub height: u64,
    #[prost(bytes, tag = "4")]
    pub transactions_root: std::vec::Vec<u8>,
    #[prost(bytes, tag = "5")]
    pub proposer: std::vec::Vec<u8>,
}
```

```rust
#[derive(PartialEq,Clone,Default)]
pub struct Proof {
    // message fields
    pub content: ::std::vec::Vec<u8>,
    pub field_type: ProofType,
    // special fields
    pub unknown_fields: ::protobuf::UnknownFields,
    pub cached_size: ::protobuf::CachedSize,
}

impl<'a> ::std::default::Default for &'a Proof {
    fn default() -> &'a Proof {
        <Proof as ::protobuf::Message>::default_instance()
    }
}

impl Proof {
    pub fn new() -> Proof {
        ::std::default::Default::default()
    }

    // bytes content = 1;


    pub fn get_content(&self) -> &[u8] {
        &self.content
    }
    pub fn clear_content(&mut self) {
        self.content.clear();
    }
```

例子

```
def gen_closed_enum(self, name: Text, enum_variants: List[Tuple[int, EnumValueDescriptorProto]], scl: Sourc

    # Generate a closed enum
    self.write_comments(self.source_code_info_by_scl.get(tuple(scl)))
    if self.derive_serde:
        self.write(
            "#[derive(Clone, Copy, PartialEq, Eq, PartialOrd, Ord, Debug, Hash, Deserialize, Serialize)]"
        )
    else:
        self.write(
            "#[derive(Clone, Copy, PartialEq, Eq, PartialOrd, Ord, Debug, Hash)]"
        )
    self.write("#[repr(i32)]")
```

```
let transport = generate_transport(&server_service, &server_trait, &path);

quote! {
    /// Generated server implementations.
    pub mod #server_mod {
        #![allow(unused_variables, dead_code, missing_docs)]
        use tonic::codegen::*;

        #generated_trait

        #service_doc
        #[derive(Debug)]
        pub struct #server_service<T: #server_trait> {
            inner: _Inner<T>,
        }
```

Demo

- https://github.com/rink1969/proto_desc_printer

Q&A

溪塔科技小助手

扫一扫上面的二维码图案，加我微信