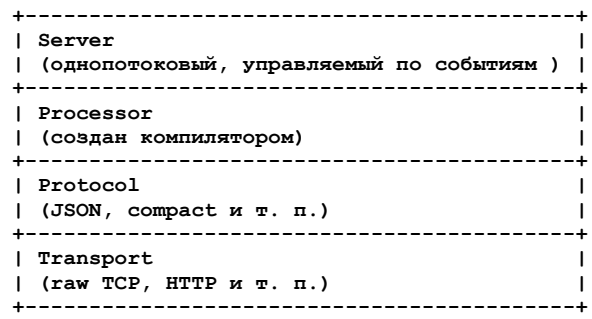
Сетевой стек thrift

Оригинал

Простое представление сетевого стека Apache Thrift приведено на рисунке



Transport

Транспортный уровень обеспечивает простую абстракцию чтения-записи из сети или в сеть. Это позволяет Thrift отвязать нижележащий (базовый) транспорт от остальной части системы (например, сериализации и десериализации). Интерфейс **Transport** обеспечивает ряд методов (неполный список):

- open;
- close;
- read;
- write;
- flush.

В дополнение к интерфейсу **Transport** Thrift использует интерфейс **ServerTransport** для восприятия и создания примитивов транспортных объектов. Этот интерфейс применяется в основном на серверной стороне для создания новых транспортных объектов при входящих соединениях. Интерфейс **ServerTransport** поддерживает методы:

- open;
- listen;
- accept;
- close.

Имеется несколько вариантов транспорта для большинства поддерживаемых в Thrift языков:

- file - чтение и запись дисковых файлов;
- http - взаимодействие по протоколу http.

Protocol

Абстракция протокола обеспечивает механизм отображения хранящихся в памяти структур данных в «проводной» формат (wire-format). Иными словами, протокол определяет использование типами данных нижележащего транспорта для кодирования и декодирования себя. Таким образом, реализация протокола управляет схемой кодирования и отвечает за сериализацию и десериализацию. Примеры протоколов включают JSON, XML, plain text, compact binary.

Функции интерфейса **Protocol** указаны ниже.

```
writeMessageBegin(name, type, seq)
writeMessageEnd()
writeStructBegin(name)
writeStructEnd()
writeFieldBegin(name, type, id)
writeFieldEnd()
writeFieldStop()
writeMapBegin(ktype, vtype, size)
writeMapEnd()
writeListBegin(etype, size)
writeListEnd()
writeSetBegin(etype, size)
writeSetEnd()
writeBool(bool)
writeByte(byte)
writeI16(i16)
writeI32(i32)
writeI64(i64)
writeDouble(double)
writeString(string)

name, type, seq = readMessageBegin()
                  readMessageEnd()
name = readStructBegin()
```

```
       readStructEnd()
name, type, id = readFieldBegin()
                 readFieldEnd()
k, v, size = readMapBegin()
             readMapEnd()
etype, size = readListBegin()
              readListEnd()
etype, size = readSetBegin()
              readSetEnd()
bool = readBool()
byte = readByte()
i16 = readI16()
i32 = readI32()
i64 = readI64()
double = readDouble()
string = readString()
```

Протоколы Thrift являются потоковыми и не требуют явного кадрирования. Например, не требуется знать размер строки или число элементов списка перед началом его преобразования в последовательность (сериализации). Протоколы, доступные для большинства поддерживаемых Thrift языков, включают:

- binary - достаточно простое двоичное кодирование, где размер и тип поля представляются байтами, за которыми следуют значения полей;
- compact - см. THRIFT-110;
- json.

Processor

Уровень Processor инкапсулирует чтение данных из входного потока и запись в выходной поток. Потоки на входе и выходе представляются объектами уровня Protocol. Интерфейс Processor крайне прост и имеет вид

```
interface TProcessor {
    bool process(TProtocol in, TProtocol out) throws TException
}
```

Реализации процессора для конкретных серверов создаются компилятором. Процессов, по существу, считывает данные из «провода» (с помощью протокола ввода), передаёт их обработчику (реализуется пользователем) и записывает отклик в «провод» (с помощью протокола вывода).

Server

Сервер объединяет перечисленные выше функции:

- создание транспорта;
- создание протоколов ввода-вывода для транспорта;
- создание процессора на основе протоколов ввода-вывода;
- ожидание входящих вызовов и передача их процессору.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru