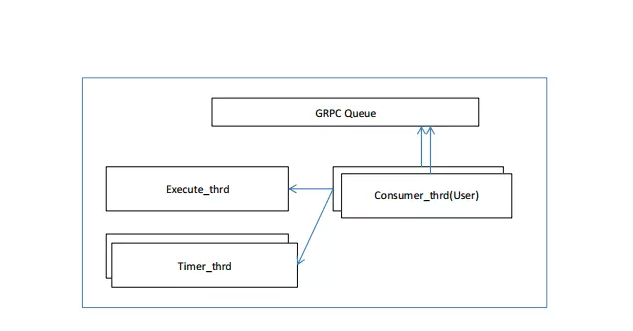
**GRPC调研**

[](https://www.jianshu.com/u/969b247d1c2d)

[巡南山的小钻风](https://www.jianshu.com/u/969b247d1c2d) 关注

2018.03.26 11:00 字数 780 阅读 159评论 3喜欢 0

Grpc 线程架构

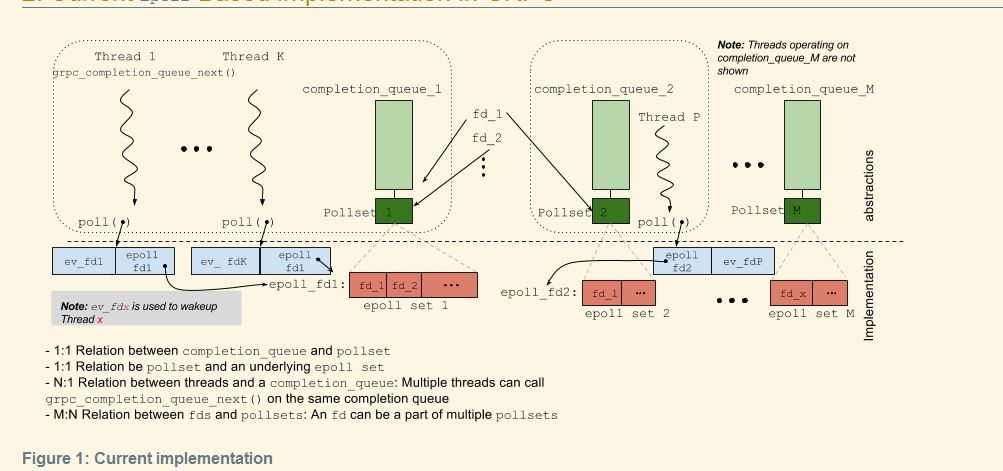


Execute\_thrd: 公有线程，启动时负责启动监听，IO，其他任务；

Timer\_thrd：负责执行指定的定时任务（非周期性任务）；

Consumer\_thrd：从Queue中取Event消费。

**网络模型**



Thread: 用户Consumer线程；

Completion\_queue: Grpc Queue，用来实现异步的事件队列；

Pollset: 每个queue有个对应的pollset，用来从底层收集IO事件；

Polling island : 是pollset的一个逻辑单位，可以包含多个pollset；

Epoll set: 底层os对应的epool set，OS内核负责收活动FD，底层epool的封装（windows giops封装）

**Transport Explainer**

在Grpc上的一个batch，由几个基本的stream ops组成。Batch是调度的基本单位。

Stream ops如下:

send\_initial\_metadata

Client: initate an RPC

Server: supply response headers

recv\_initial\_metadata

Client: get response headers

Server: accept an RPC

send\_message (zero or more) : send a data buffer

recv\_message (zero or more) : receive a data buffer

send\_trailing\_metadata

Client: half-close indicating that no more messages will be coming

Server: full-close providing final status for the RPC

recv\_trailing\_metadata: get final status for the RPC

Server extra: This op shouldn't actually be considered complete until the server has also sent trailing metadata to provide the other side with final status

cancel\_stream: Attempt to cancel an RPC

collect\_stats: Get stats

The fundamental responsibility of the transport is to transform between this internal format and an actual wire format, so the processing of these operations is largely transport-specific.

Transport负责内部格式和网络格式之间转换。

One or more of these ops are grouped into a batch. Applications can start all of a call's ops in a single batch, or they can split them up into multiple batches. Results of each batch are returned asynchronously via a completion queue.

每个call的ops可以放到一个batch中，如果一部实现，会放到多个batch中，每一个batch会返回completion queue一个Event。

Internally, we use callbacks to indicate completion. The surface layer creates a callback when starting a new batch and sends it down the filter stack along with the batch. The transport must invoke this callback when the batch is complete, and then the surface layer returns an event to the application via the completion queue. Each batch can have up to 3 callbacks:

recv\_initial\_metadata\_ready (called by the transport when the recv\_initial\_metadata op is complete)

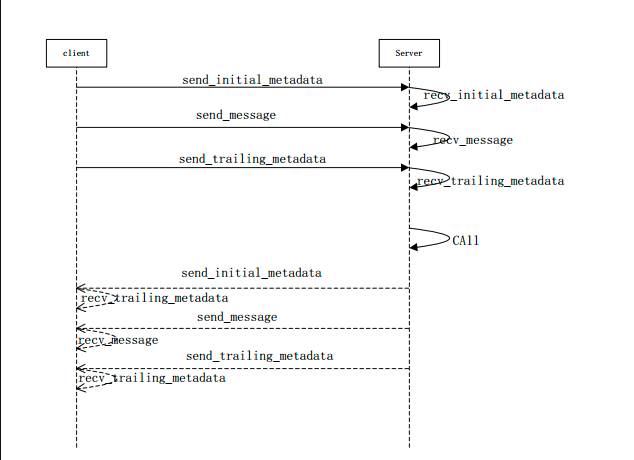
recv\_message\_ready (called by the transport when the recv\_message op is complete)

on\_complete (called by the transport when the entire batch is complete)

上面三个回调函数，肯定会被执行

一个batch可以分成这三个阶段recv\_initial\_metadata，recv\_message\_ready，on\_complete。

一个call的时序图：



Client send\_initial\_metadata: Initiate an RPC with a path (method) and authority

Server recv\_initial\_metadata: accept an RPC

Client send\_message: Supply the input proto for the RPC

Server recv\_message: Get the input proto from the RPC

Client send\_trailing\_metadata: This is a half-close indicating that the client will not be sending any more messages

Server recv\_trailing\_metadata: The server sees this from the client and knows that it will not get any more messages. This won't complete yet though, as described above.

Server send\_initial\_metadata, send\_message, send\_trailing\_metadata: A batch can contain multiple ops, and this batch provides the RPC response headers, response content, and status. Note that sending the trailing metadata will also complete the server's receive of trailing metadata.

Client recv\_initial\_metadata: The number of ops in one side of the batch has no relation with the number of ops on the other side of the batch. In this case, the client is just collecting the response headers.

Client recv\_message, recv\_trailing\_metadata: Get the data response and status

**模式处理**

服务器端异步模式框架，添加自定义Event Queue增加并发性能

