进阶课程㉜丨Apollo ROS原理—1

知识点

敲黑板，本文需要学习的知识点有

节点　　障碍物

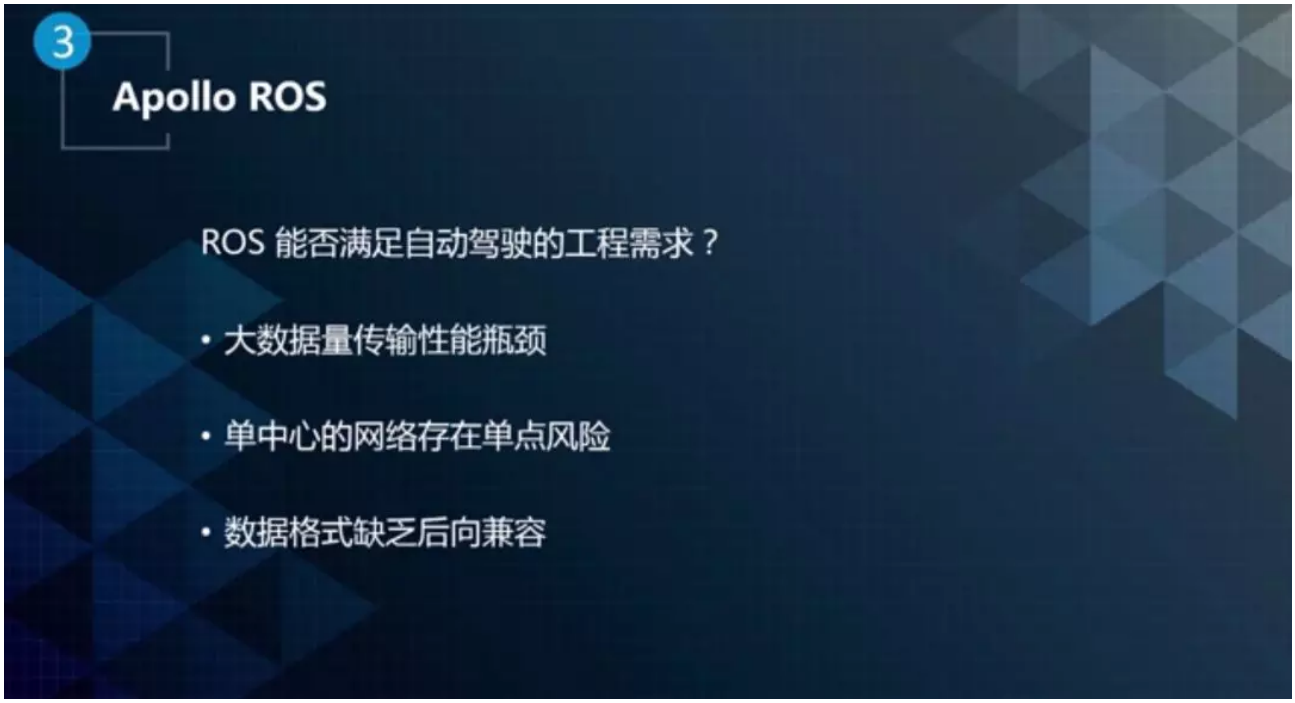
视觉定位　　共享内存

吞吐量　　拓扑结构

ROS在开发过程中，基于功能把整个自动驾驶系统分成多个模块，每个模块负责自己消息的**接收、处理、发布**。当模块需要联调时，通过框架可以把各个模块快速的集成到一起。

目前ROS仅适用于Apollo 3.0之前的版本，最新代码及功能还请参照Apollo 3.5及5.0版本。

以下，ENJOY



ROS是很多大学或者实验室进行探索性项目实验所采用的基本框架，但在实际的自动驾驶工程化需求面前，还有很多明显不足。

大数据传输性能瓶颈

实验性项目里面采用的Topic是Message，数据量是比较小的，可能只有几K或者最多1~2MHZ，但在实际自动驾驶场景里面数据量非常大。例如Lidar一帧数据大概是7M，一秒钟10帧，就会产生70M/S的流量；一个Camera按5M计算，四个Camera就是20M，如果是按10HZ计算一秒钟会产生200M左右的数据。ROS架构对大数据传输存在很大的性能瓶颈，一种直接后果是**时延非常高**，这在自动驾驶整个系统里面是非常危险的。

单中心的网络存在单点风险

中心化的网络存在明显的单点风险，整个ROS虽然是一个松耦合的架构，它包含一个节点管理器，节点管理器介入的时候，只是在节点建立通信之前有一个简单的拓扑映射，这种关系虽说极大程度释放了各个节点之间开发的耦合，但同时也带来了比较大的风险。如果Roscore存在一些故障退出，而节点之间使用了需要不定时的交互方式，像Service 、Parem进行数据交互的时候就会存在一定的风

险。如果是分布式系统， Roscore只存在于一台机器上，Roscore如果出现故障，两台机器之间通信就处于一个不可信的状态。

数据格式缺乏后向兼容

ROS是基于Message的分发和订阅的消息通讯框架，使用Message需要提前设置Message包含哪些类型的数据。把这个模块放到一个更复杂的系统里面的时候，要格外注意Message之间的数据兼容。

我们根据实际的场景需求，在定义的Obstacle信息里面加一段文字，那么相应的下游所有订阅此Obstacle的节点都要去做对应的适配，同时基于之前的Message所录制的一些实验数据，想在新的框架下使用也都需要一个批量的转化。ROS现有的数据格式缺少后向兼容，此问题在Apollo ROS里面得到解决。

02

Apollo ROS对ROS的改进

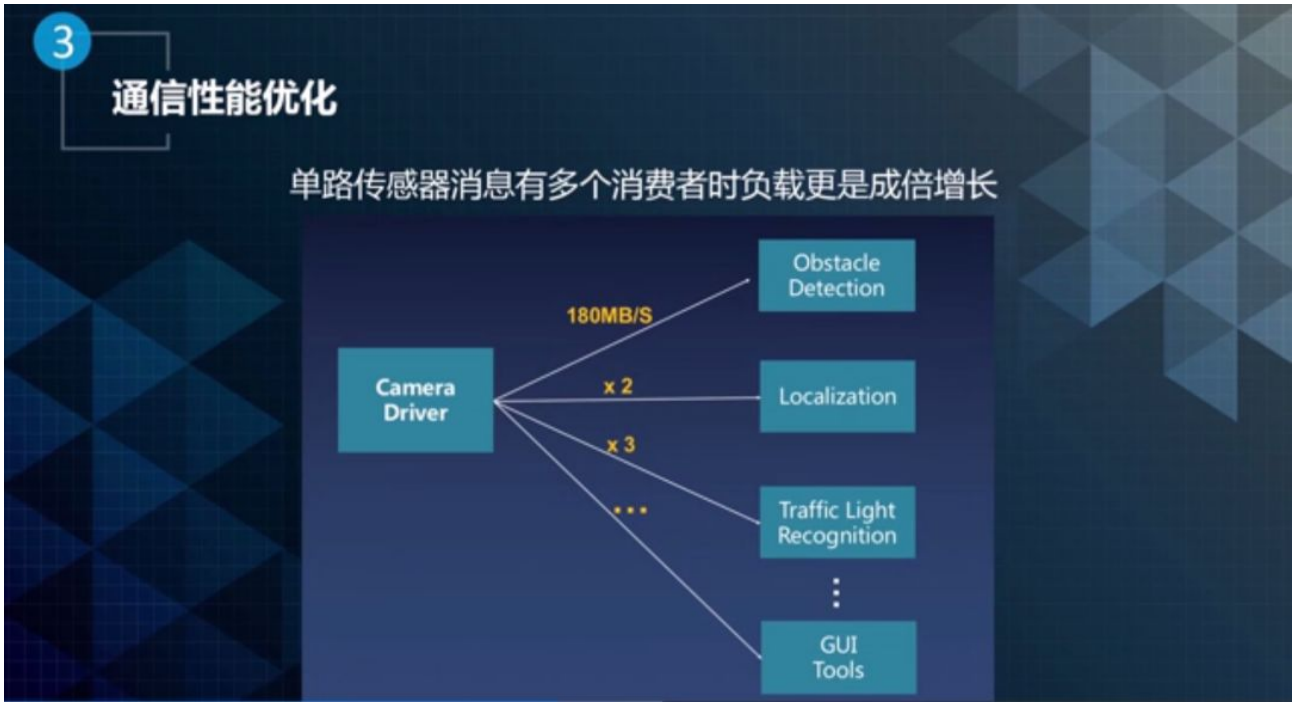
III 通信性能优化

首先，我们看看为什么要进行通信优化，主要有以下几点原因：

- **自动驾驶大量使用传感器引发很大的传输带宽需求。**

自动驾驶使用大量的传感器，这些传感器的数据量非常庞大。大量数据在目前ROS的通讯架构里面会带来比较高的延迟或是丢帧。节点之间通信是一帧一帧进行的，如果上一帧消息高延迟时，下一帧消息的发送就需要等待。ROS提供了这种消息丢弃的机制，如果等待时间长会丢弃一些数据，数据丢弃在实际自动驾驶系统中会造成比较大的风险。

- **单路传感器消息有多个消费者时负载成倍增长。**

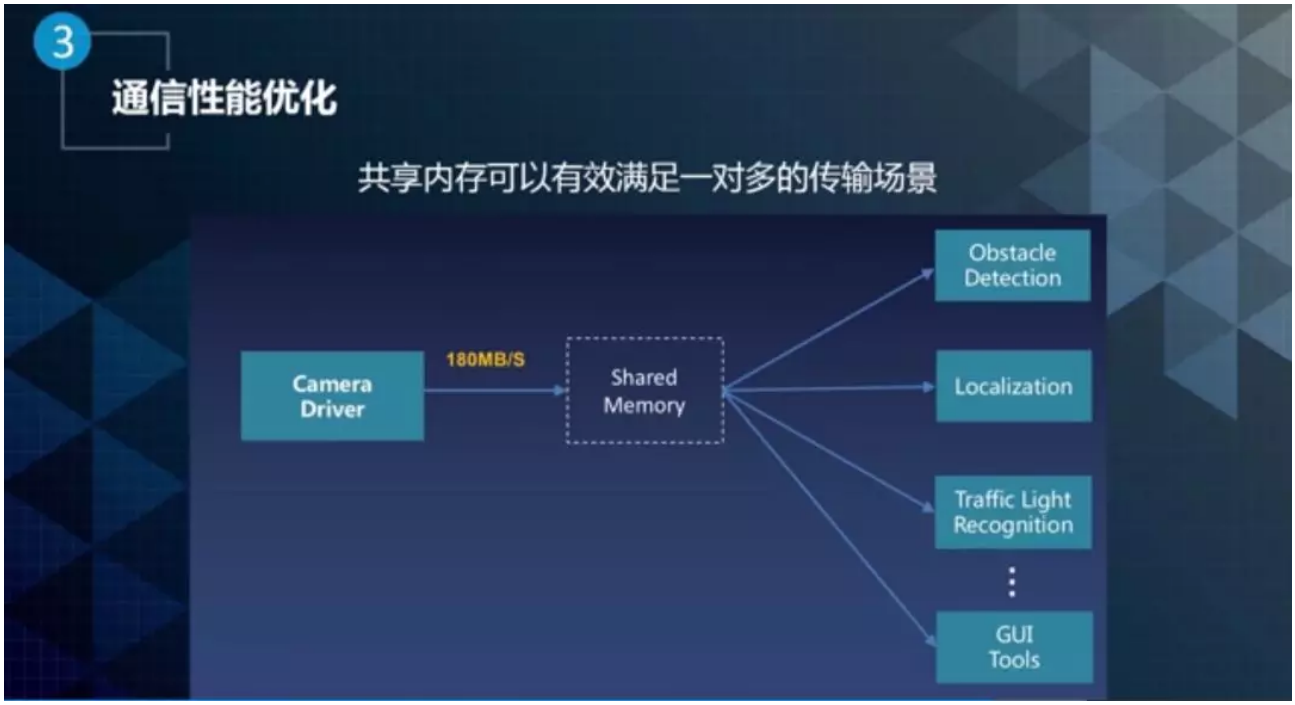


自动驾驶系统发送传感器数据是一对一进行的。例如Lidar向自动驾驶系统发送数据时，如果只有一个订阅节点，传输的数据量是7M乘以10HZ，也就是70MB/S。自动驾驶系统是一个比较复杂的拓扑结构，一个传感器数据可能会有很多的下游订阅节点。例如感知的障碍物检测、通过视觉定位的模块、用红绿灯识别等都会订阅Camera信息。在单点的情况下是一对一，如果是一对多，传输的数据会被复制多次，造成网络负载成倍增加。

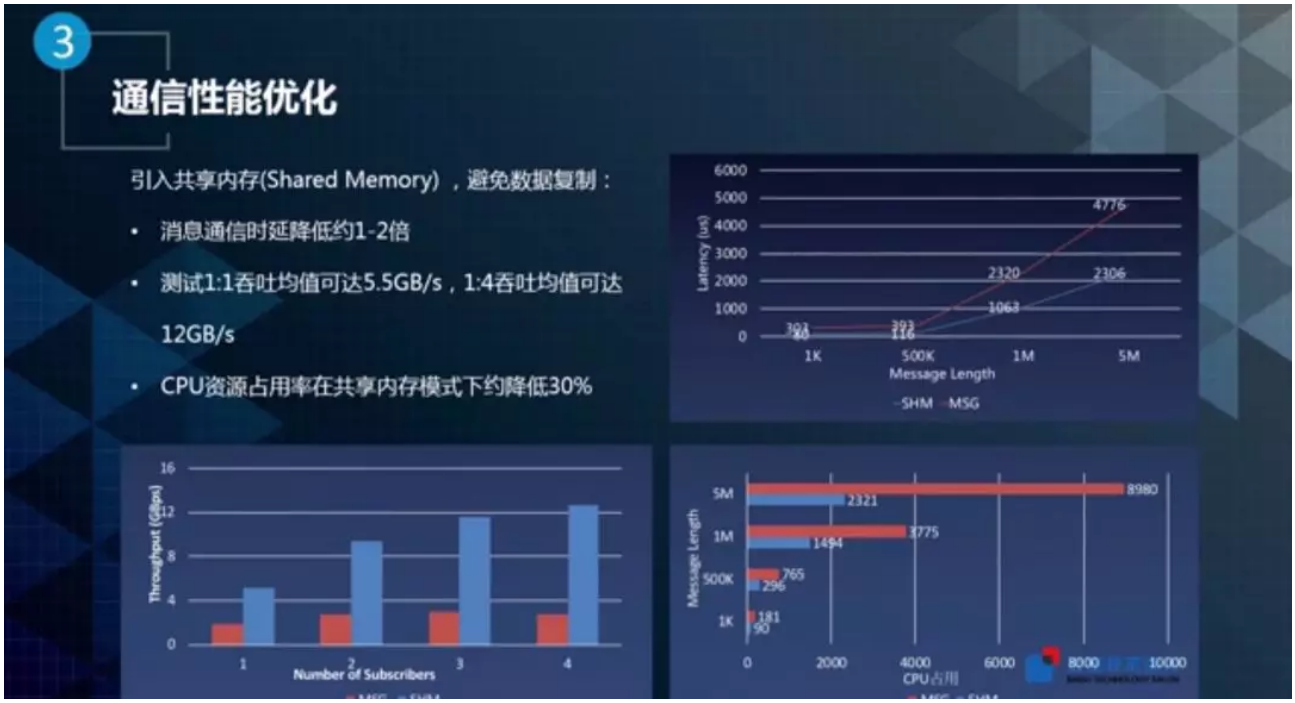
针对这一问题，Apollo ROS做了一个基于共享内存的通信机制减少数据的复制次数，从而提升这种通信模式的效率。

如上图所示，左侧是**ROS原生的通讯框架**，一个数据从发送方到接收方经历四次数据复制。第一次是从节点到用户内存的数据复制，第二次是从发送方到内核的数据复制，第三次是经过TCP连接，从内核再向接收节点用户态空间的复制，第四次是接收节点拿到这个信息之后，通过反序列化把信息取出来组成一个结构变化的信息。

右侧是**Apollo ROS优化后的框架**，它基于共享内存改进，可以减少两次数据拷贝。第一次是发送节点把消息序列化成流式数据，第二次是接收节点直接从共享内存里面取相应的消息指针，把共享内存消息取出来进行反序列化成结构化信息进行使用。减少了从用户到内核态以及从内核态到用户的两次数据拷贝。



对于有多个订阅节点的情况，例如Camera下游会有很多订阅节点，如果是三个节点，会有三条通信链路，分别是四次的内存拷贝，也就是12次数据拷贝。而在基于共享内存的通信方式下，每一条链路内存拷贝的次数只需要两次，三条链路只需要六次。



上图是一个有优化的Apollo ROS Benchmark效果展示，用Apollo ROS替代原生ROS Socket的通信方式之后，从实际的路测数据表现情况来看，性能提升非常明显。以上三张图分别从三个指标对比两者的性能。

消息通信延时

如图右上角所示：随着消息逐渐增大，基于共享内存通信延时比基于原声ROS Socket的通信延时降低一半。以5M数据为例，传送一帧5M大小的数据，基于ROS Socket大概需要四毫秒左右的时间，基于共享内存通信只需要两毫秒左右。

吞吐量

如图左下角所示：整个自动驾驶系统的网络拓扑结构非常复杂，数据流向的拓扑结构也比较复杂。在一些极端的情况下，整机数据量会增加。在一些多车道，路面状况比较复杂，车辆较多的情况下，感知和Planning模块，或者和其它模块之间的数据流就会成倍增加，所以在测试一些极端情况下，系统吞吐量也是自动驾驶需要考虑的一个重要方面。

如上图所示，在吞吐量测试1：1情况下，整机性能可以达到5.5GB每秒的速度，如果是1：4，性能提升会更明显。

CPU资源占用率

如图右下角所示：CPU资源占用率在共享内存通信情况下降低约30%， 主要是因为减少了多次内存复制。