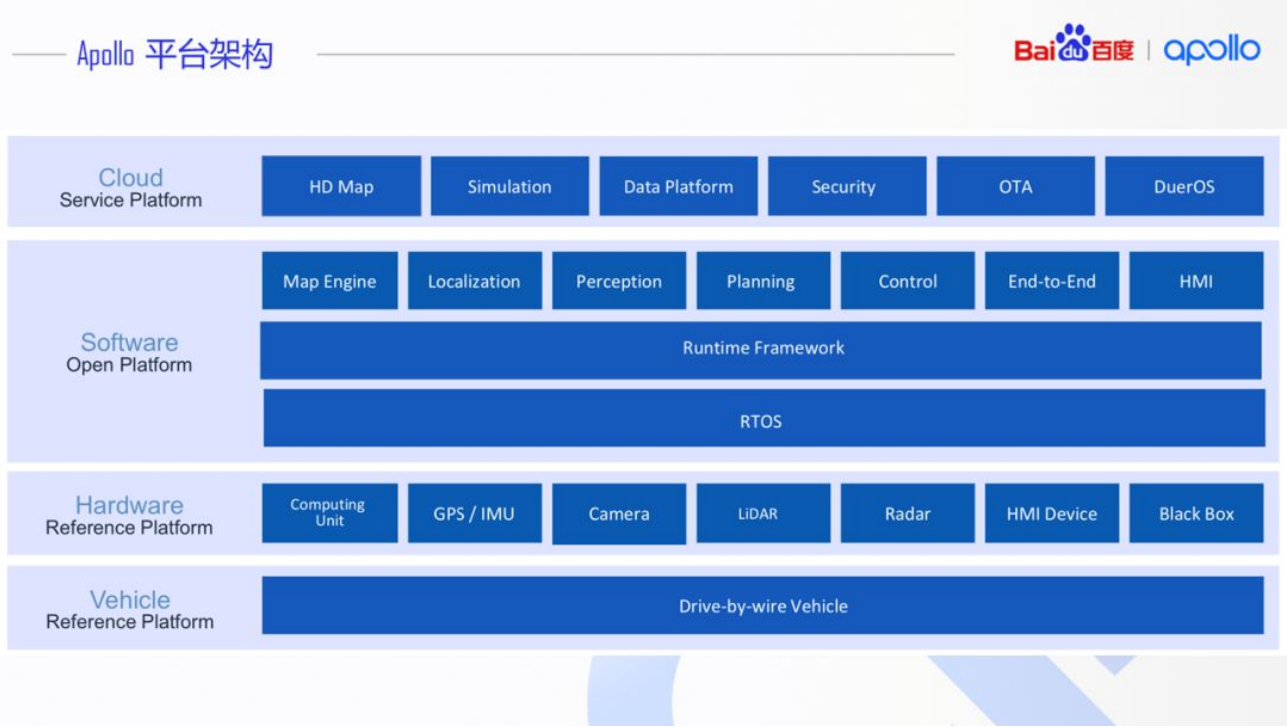
Apollo公开课丨自动驾驶专用计算框架探索和实践

对于自动驾驶框架，相信大家一定听过ROS。**ROS是一个机器人操作系统框架，用在自动驾驶领域还存在一些问题**，例如自动驾驶是专用领域，ROS是通用操作系统，ROS采用相对公平的调度方案，按照时间片分片，周而复始执行。对自动驾驶而言，用公平算法可能并不是特别合适。

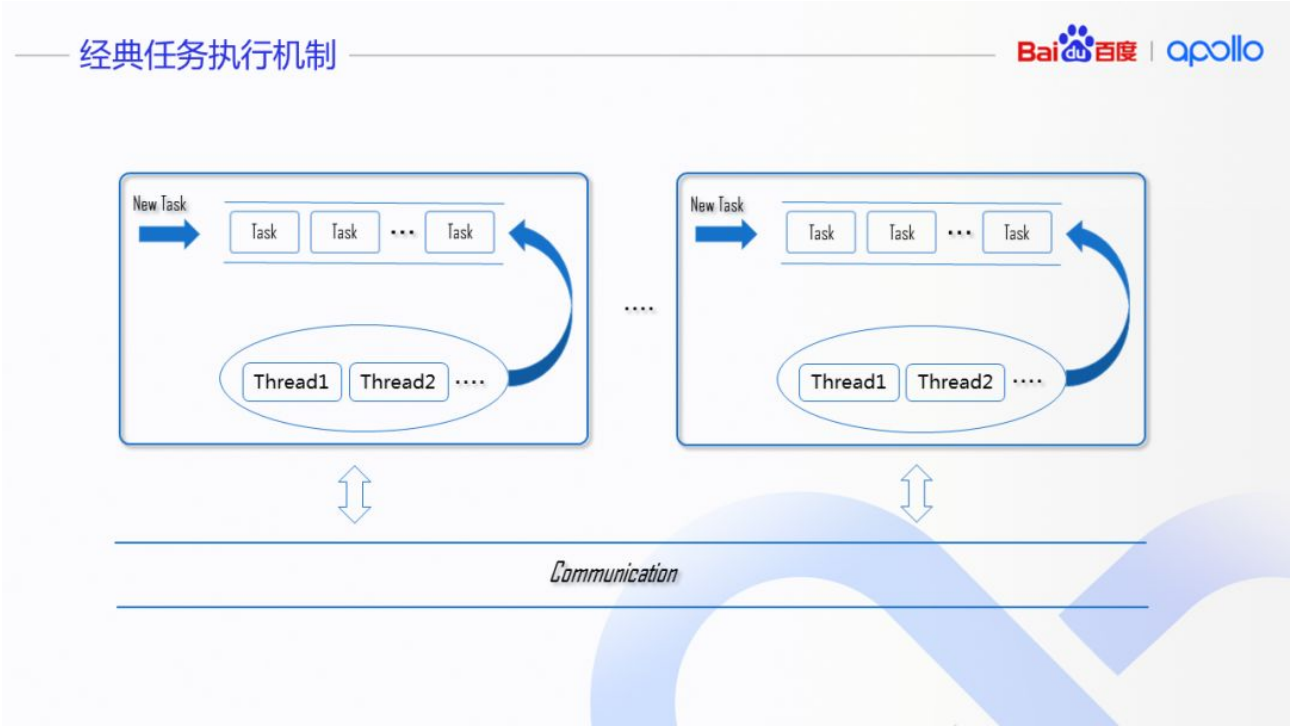
因此，我们认为自动驾驶应该有一个针对这种应用场景的调度框架，这就是我们研发自己的自动驾驶调度系统的原因。下面将和大家分享**百度的自动驾驶专用计算框架Cyber RT**。



▲Apollo平台架构

如图所示，Apollo 平台架构包括四个部分，最上面是**云端服务平台**，中间是**软件开源平台**，往下是**硬件参考平台**，最底层是**车辆参考平台**。

为了了解中间的软件框架，这里简单介绍一下Apollo平台架构。参考车辆更多是线控汽车，通过信号控制汽车的油门、刹车、方向等。硬件平台主要包含车载的计算单元，相当于人的大脑，以及各类传感器，用于感知周围环境。中间软件平台的中间部分是框架，负责协调软件层各个模块的任务调度。最上面的云端服务包括一些算法，车上软件升级OTA、地图以及其他云端数据服务。



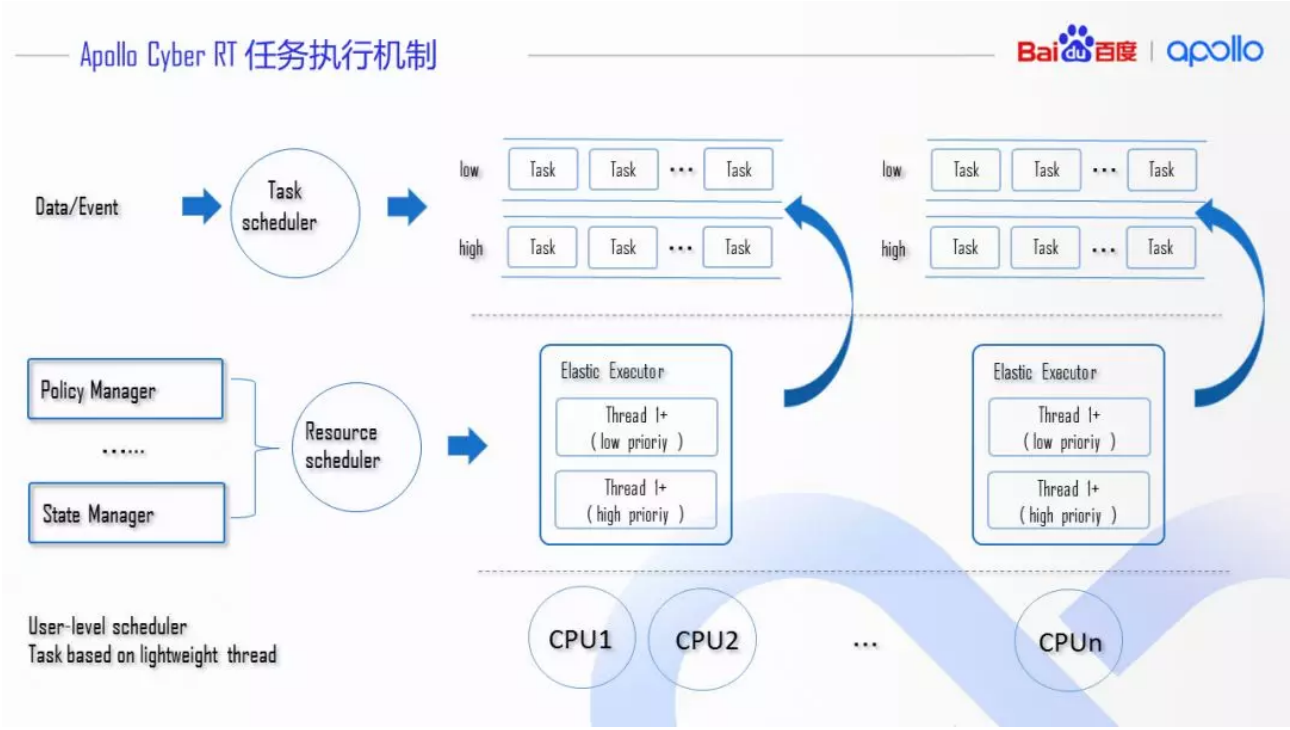
▲ROS经典任务执行机制

这个是ROS的经典任务执行模式，左右两边分别是一个node，每个node可能起一个线程或多个线程。ROS程序会给每个Message一个**callback（回调）函数**，每次收到这个消息即可得到执行机会，可处理收到的数据或做一些其他动作。

这个模式中的通讯环节由ROS服务器处理，订阅的某个消息来了之后就会触发**callback函数**。ROS服务器通用性比较强，来任务就处理，节点之间谁先处理，谁后处理，完全看系统内核调度，目前基本上使用公平调度策略。这种模式导致大部分时间花在任务的挑选、选择、上下文不停的切换过程中，导致服务器性能下降厉害，系统很难达到较好状态。

▲自动驾驶面临着的挑战

另外，自动驾驶面临着很多的挑战，如图所示，传感器数据量巨大，车身配备了很多的摄像头，每个摄像头的分辨率都比较高，每秒产生的数据量可能高达1个多GB。其次，自动驾驶涉及的任务错综复杂，需要快速感知和决策，各种任务都需要障碍物消息等等，公平的调度机制可能不合适。第三，自动驾驶中关键任务和非关键任务的调度需要有所区别，例如，有些场景车突然出现，这时应该执行关键任务，赶紧减速避让，而不是处理非关键的数据处理任务。



▲Cyber RT任务执行机制

根据自动驾驶特点，我们在软件层面做了一些尝试，提出了自动驾驶专用计算框架Cyber RT，其任务执行机制如上图所示。

首先，该**计算框架采用集中式控制机制**，随着汽车工业发展，硬件系统越来越智能，处理能力越来越强，可以将软件层面的各种算法、任务都放到硬件层的计算单元处理，不需要为每个模块配置一个单片机，增加控制的难度。

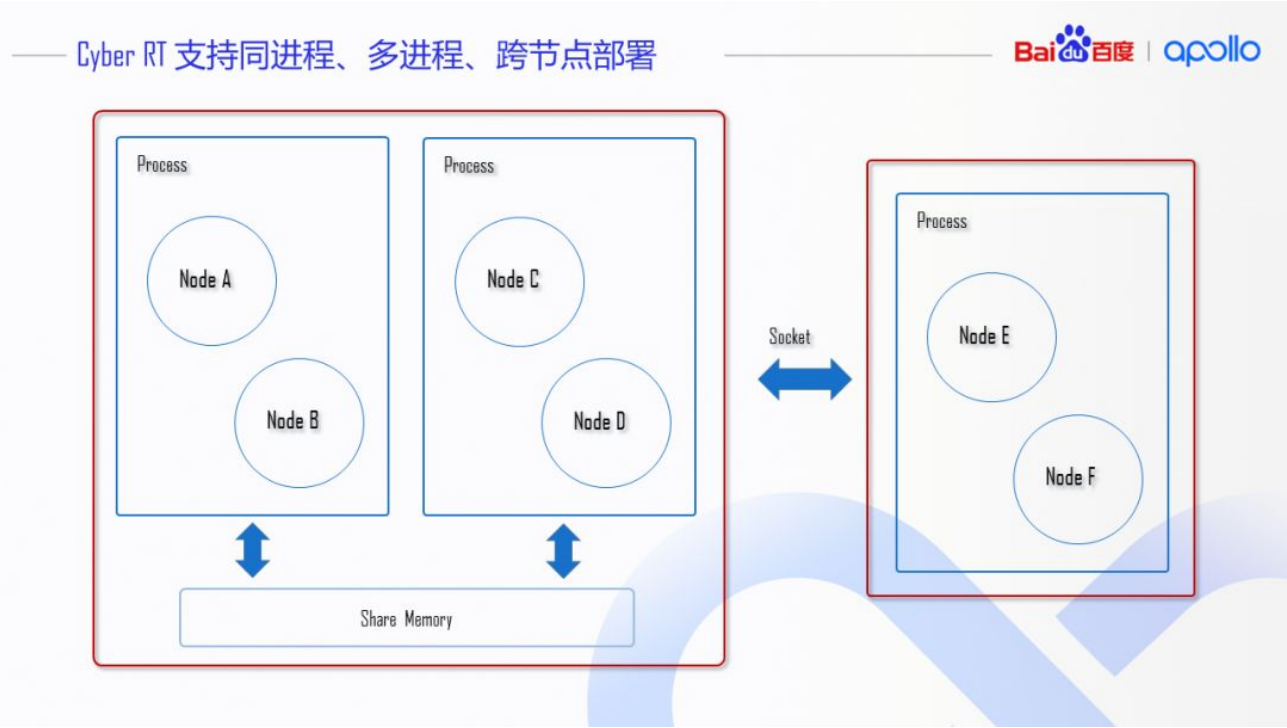
中间部分是**弹性执行器**，它改变了之前分散式调度的方法，采用集中式协调运行的策略。比如根据任务属性考虑两个任务是否应该分配到相近CPU或者同一个CPU执行。

在资源分配方面，我们开源了一些**资源划分算法**，它会划分一些组，比如可以把某个任务放在一个处理器上，或者给某组任务设置一个任务池，分配到几个CPU执行。可以通过设置任务优先级定义任务的执行顺序。例如实时任务的优先级比较高，会抢占低优先级任务的资源。我们希望把任务弹性做得更好一点，在资源有限的条件下，让系统运行更好，最重要资源分配给最需要任务。当然可以通过增加资源来应对这种情况，但是从技术架构人员的角度出发，考虑的永远是资源处于不够用的状态，如

何合理使用资源，让资源利用率最高，成本、功耗等最小，这就是为什么要打造一个专业的调度框架。

其实，这种框架并不好做，**内核调度**发展到今天历经了无数次的迭代更新和大量杰出开发者不停的改进，而自动驾驶远比通用系统复杂。**服务器的调度**关注的是切换代价足够小，不涉及过多业务逻辑。而自动驾驶涉及了很多业务逻辑，要将其看得更加重要。那么，是不是可以在内核中开发调度模块，因为内核里面调度策略有很多种，是不是可以在里面加一个，让内核和业务逻辑有个交互，上面业务逻辑告诉内核现在处于什么状态。

这种方式存在几个问题，第一是缺乏灵活性，把调度做到内核里，如果要使用别的系统怎么办？第二是切换代价比较大。这也是把框架从内核空间搬到任务空间的原因。



▲Cyber RT支持同进程、多进程、跨节点部署

下面我们讨论一下调度不同内核的关系以及Task载体。首先我们希望能对任务做更多控制，因此使用了协程，协程是属于用户空间的一个轻量级进程，它切换比较快，不需要CPU上下来回倒。第二是可控比较多，这种功能对技术要求比较高。目前我们开放了Cyber RT的两个调度策略，一个是**经典策略**，另一个是**精细配置**，结合车的整体情况去调优。

上面说的都是很复杂东西，但是使用起来比较容易。下面我们就介绍如何使用Cyber RT做一个组件。

▲使用Cyber RT做一个组件

如图所示，Component这个组件有两个**关键函数**，第一个是**init（初始化）**，做独立组件配置信息或者组织一些需要传递的信息。第二是**proc函数**，其实就是相当于ROS的拷贝。上半部分是核心逻辑，之后得为逻辑表达准备DAG图，写一个DAG文件描述，包括component组件名字、类和Confpath等信息。比如模型文件，receiver表示收到信息，sender表示这个组件发什么消息。

在编译完新的组件类，准备好DAG文件之后，还要准备一个 **launch 文件**。