**产品功能需求**

**术语解释：**

**系统架构：**“客户端-服务器”功能架构、“发布-订阅”通信架构

**负载模型：**周期性、非周期性，强时限，弱时限混合任务。任务载体可以是线程或进程。任务不确定性来自来共享资源、任务并行

**调度资源：**处理器包括GPU、CPU、DLA。 资源包括cuda stream、内存、互斥锁等其他共享资源。

**调度策略：**两级，由callback调度算法和CPU调度算法共同决定

**系统环境：**POSIX.1

**功能划分：**

**1.配置管理：**生成各种相应的初始化配置，并提供必要的分发功能

**2.工作流调度器：**（1）动态确定每个自动驾驶相关任务的激活时间，比如说离线标定、在线标定、建图等任务；（2）动态确定任务中模型的选择、缩放、量化、剪枝、近似等。

**3. 硬件资源服务：**周期性收集系统内的现有硬件资源，提供给数据流调度器。全面的反映系统各种参数的变化。

**4. 数据流调度器：**整个实时系统的核心模块。自动驾驶系统由数个**数据流链**（data flow chain）构成。每个数据流链代表一个自动驾驶相关的**任务**，由多个具有**数据依赖**关系的数据处理模块组成DAG的形式 （A -> B 代表模块A的输出为模块B的输入）。调度器以最小化多链端到端时延为优化目标，通过和线程管理模块以及CPU调度模块互动来完成该优化目标。

**5. 进程/线程管理：**线程管理中的相关机制进行改进

**6. CPU调度器：**不会对Linux/RTOS现有的内核调度器进行大的调整（基于PREEMPT补丁）

**7.日志管理：**提供系统运行的必要数据的连续时间反映，使系统的运行可以得到追踪和审计。

**8.通信管理：**将AutoSAR通信模块中的API与DDS的API建立关系。

**工作流程：**

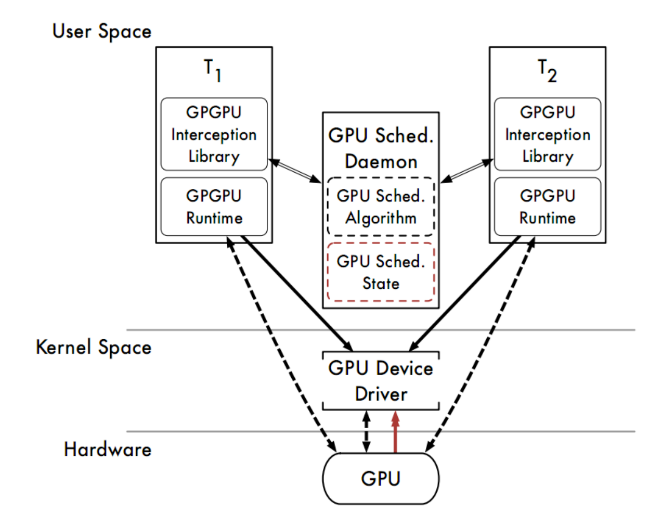
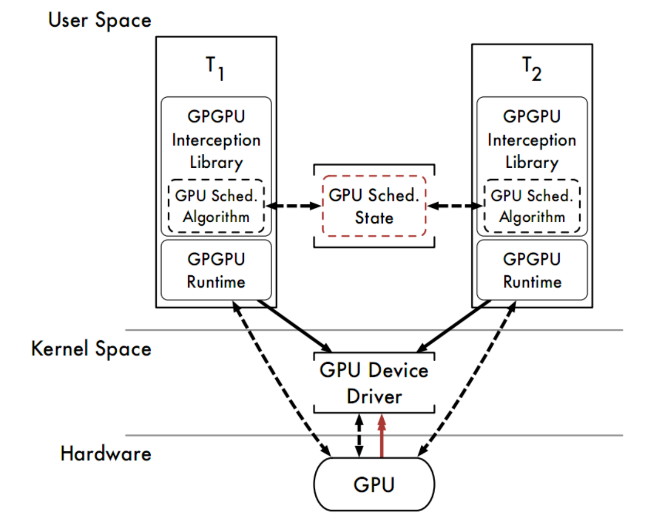
1.用户指定任务的DAG图通过**配置管理**分发到各个模块，DAG图的方向由数据流向确定。DAG图中的节点需要指定资源、优先级。

2.DAG图将被**工作流调度器**进一步划分为CPU、GPU执行单元（线程/进程）作为最小调度单位。每个节点通过**硬件资源服务**进行最坏时间估计得到对应的执行时间信息。

3.APP绑定到AUTOSAR运行时框架，**数据流调度器**通过对AutoSAR或pthread库进行覆盖，或显式调用我们的库函数API进行调度。运行时信息由**硬件资源服务**进行统计。

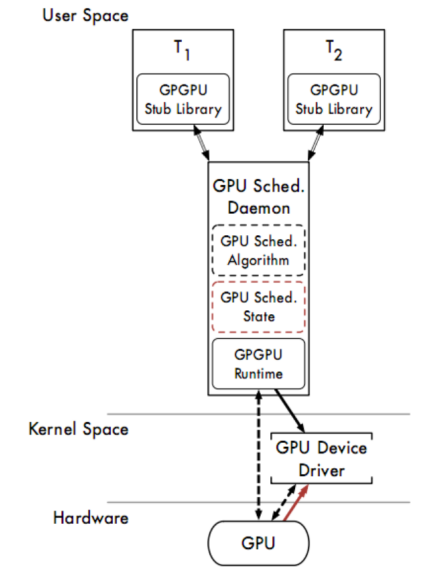
4.APP注册任务之后，不能再要求获取新的资源。APP的生命周期通过执行管理（EM）进行管理。运行数据被**日志管理**记录，并持久化。

**方案选型：**

三种方案选型：



|  |  |  |  |
| --- | --- | --- | --- |
|  | 协作式调度（守护进程） | 协作式调度（状态） | 中心化调度 |
| 优点 | 中心化方法实现简单 | 没有IPC开销，没有单独进程需要调度 | 模型运行在一个Context里可以分strem复用 |
| 缺点 | 存在一定 IPC 开销，调度进程本身会被CPU调度器调度 | 多个任务之间需要协同 | 数据需要跨进程传输 进程本身会被调度 |



|  |  |
| --- | --- |
| 事件触发调度器 | 时间触发调度器 |
| 调度算法设计复杂 | 调度算法设计简单 |
| 适用周期、偶发、非周期任务 | 适用周期性任务 |
| 动态适应性强 | 动态适应性弱 |

|  |  |  |
| --- | --- | --- |
|  | **Callback** | **WaitSet** |
| **机制** | **中间件异步监听** | **用户同步等待** |
| 吞吐量 | 低 | 高 |
| 开销 | 不需要切换 | Wait、read线程切换 |
| CPU占用 | 高 | 低 |
| 副作用 | 数据到达受callback函数处理时间影响 | 无 |
| 调度器 | 轮询处理callback group | 控制权在用户手中 |

AutoSAR中的通信服务分别为客户端和服务器proxy和skeleton生成代码，在客户端的应用程序实例化 绑定到在服务器端运行的服务实例的 代理。每个代理一次只能绑定到一个服务实例。

运用DDS解除AUTOSAR代理和服务实例耦合的方法：每个ara::com服务实例在以 service ID命名的特定分区上发布数据，每个代理使用以它们绑定的service ID命名的分区订阅数据。

|  |  |  |  |
| --- | --- | --- | --- |
| **AutoSAR API** | **events** | **methods** | **fields** |
| AutoSAR API作用 | 通知客户端应用程序在服务器端触发的事件 | 公开客户端应用程序可以调用的远程过程 | 提供客户端应用程序可以使用远程getter和setter修改的数据值 |
| 映射到DDS | Regular topic | Request topic&Reply topic | Getter/Setter对应Request topic&Reply topic，Notifier对应Regular topic |

**功能详细描述**

1. **配置管理：**

功能1.1：配置输入

**功能参数**，说明任务内部属性，是否可抢占，重要性，是否是可选执行。

**互联参数**，说明任务如何和其他任务依赖（launch kernel等）。依赖以DAG图呈现，共享资源（锁）对执行次序的影响不在此范围内。依赖可以是直接数据依赖，也可以是时间依赖（两个任务完成时间之间的约束）。不同的互联参数可以以AND/OR 的形式并联。

**资源参数**，说明资源需求，包括stream数量，共享资源占用标志位，GPU个数、CPU个数、DLA队列长度

**时间参数，**包括周期、截止期，等定时约束和行为。

功能1.2：静态指定，主动分派给其他模块

**2.工作流调度器：**

功能2.1：接收**配置管理**的配置，进一步分割DAG（按CPU-GPU）。

功能2.2：根据**硬件资源服务**的评估结果，进一步填充DAG中未完善的信息

功能2.3：根据**硬件资源服务**的资源监控信息，动态确定相关任务的激活时间

功能2.4：最终结果分发给**数据流调度器**

**3. 硬件资源服务：**

功能3.1：对任务进行最坏时间估计，具备稳定测量任务的99th 时延的能力

功能3.2：周期性主动监控系统内的现有硬件资源。包括GPU(Tegra stat)利用率, CPU利用率、 吞吐率、Cache 命中率、memory 利用率、bus、memory带宽

功能3.3：硬件资源服务需要长期运行，需要维持较低的开销(CPU占用率 < 1%)

功能3.3：对任务的计算开销、传输开销进行建模

功能3.4：对偶发任务的到达时间抖动分布进行建模

**4. 数据流调度器：**

功能4.1：接收配置信息，进行可调度性分析

功能4.2：执行调度。调度器为事件触发。

功能4.3：异常管理，当调度性分析失败，或任务执行超过deadline等异常情况，通知**日志**

功能4.4：生成一个可用于可视化的调度表

功能4.5：没有优先级反转，避免等待轮询点，回调执行期间可以被更高优先级的回调抢占

**模块**

说明：依赖于通信管理模块的实现。需要考虑序列化/反序列化，发布-订阅、请求-响应模式，跨SoC通信开销、时间同步精度影响。零拷贝依赖具体实现。

**5. 进程/线程管理：**

功能5.1：线程、进程绑定CPU、GPU（静态）

功能5.2：管理线程、进程的状态。

功能5.3：管理线程池任务队列的状态

**6. CPU调度器：**

功能6.1： 任务绑定CPU

功能6.2： 任务出让CPU

功能6.3： 设置调度参数

功能6.4： 设置调度策略

功能6.5： 获取进程的实时状态信息

**7.日志管理：**

功能7.1：异常报警功能

功能7.2：数据日志持久化

1. **通信管理：**

功能8.1：在服务实例上建立分区来支持数据订阅与发布

功能8.2：events、methods、fields API转换为DDS中topic操作

**未包含的功能：**

网络安全、数据确定性、跨平台特性、跨语言特性、功能安全\故障处理、确定性编程模型

**指标：**

最大、平均延缓（正值）

延迟（可能是正值、负值）

一定窗口内约束错过次数

**产品工作计划：**

1 熟悉 Drive OS 、熟悉 Orin 平台的特性

2 需要调用的TensorRT/CUDA 库API列表、AUTOSAR 线程/进程管理方面的API

3 特殊的硬件的支持，例如NVMEDIA、ISP、NVENC、NVDEC、NVsciSync NvSciIpc  等，需要进一步评估后决定