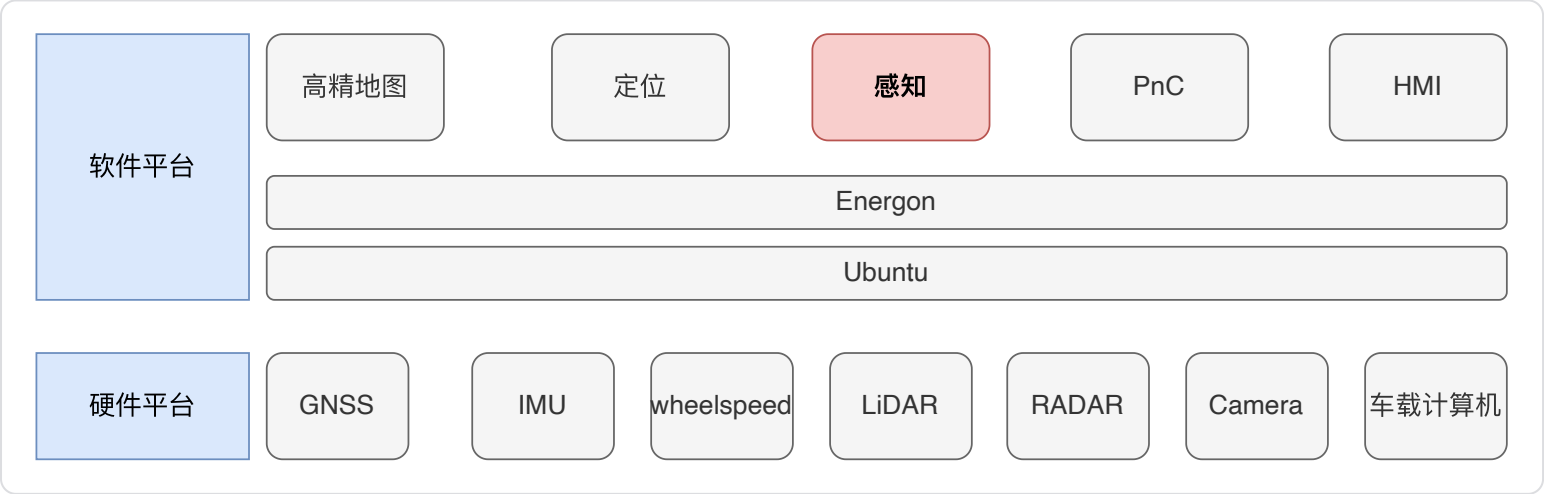
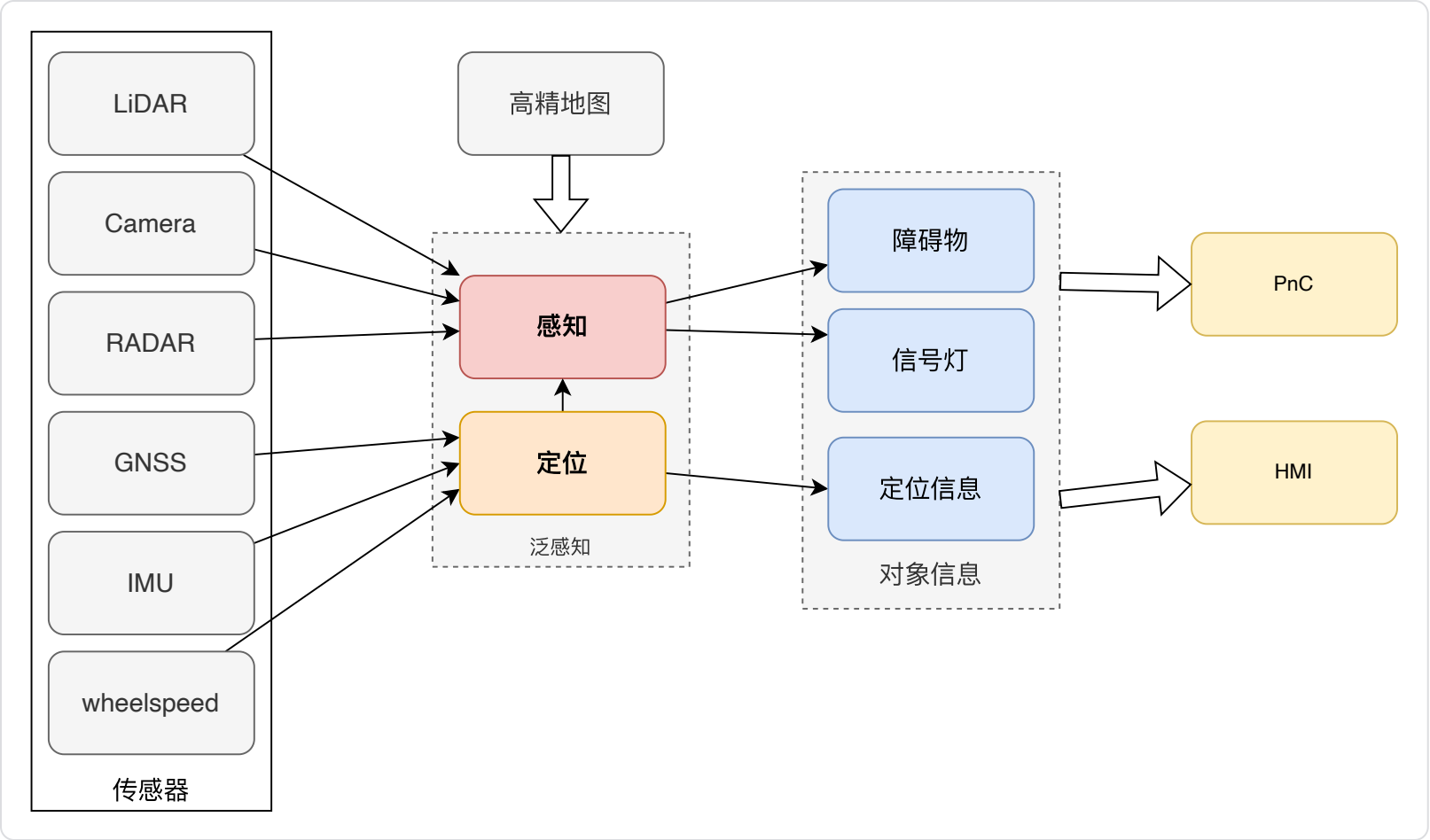
感知工程架构串讲-刘钦明

目录

什么是感知



感知在自动驾驶架构中，属于软件平台的一部分。



感知的作用是根据传感器的数据，获取车辆周围的对象信息。

输入：传感器数据，输出：对象信息。

感知输出的对象信息，主要提供给PnC，用于车的规划控制，也会提供给HMI，用于人机交互。

传感器	特点	作用
LiDAR（激光雷达）	• 精度高，可达厘米级 • 有效距离100m以内	• 主传感器，触发感知流程（频率10Hz / 周期100ms） • 识别障碍物

	• 输出数据是3D点云，数据量大	
Camera（摄像头）	• 能够识别颜色 • 分辨率高 • 受光照影响大 • 输出数据是2D的	• 识别信号灯 • 识别障碍物 • 与LiDAR数据融合互补
RADAR（毫米波雷达）	• 距离远，可达200m • 可以穿越障碍物 • 精度低	• 检测车辆速度 • 检测盲点 • 与LiDAR数据融合互补
GNSS（全球卫星导航系统）	• 能够获取全局定位 • 受天气影响大 • 精度有限	• 车辆自身定位
IMU（惯性测量单元）	• 频率高 • 精度高	• 车辆自身定位

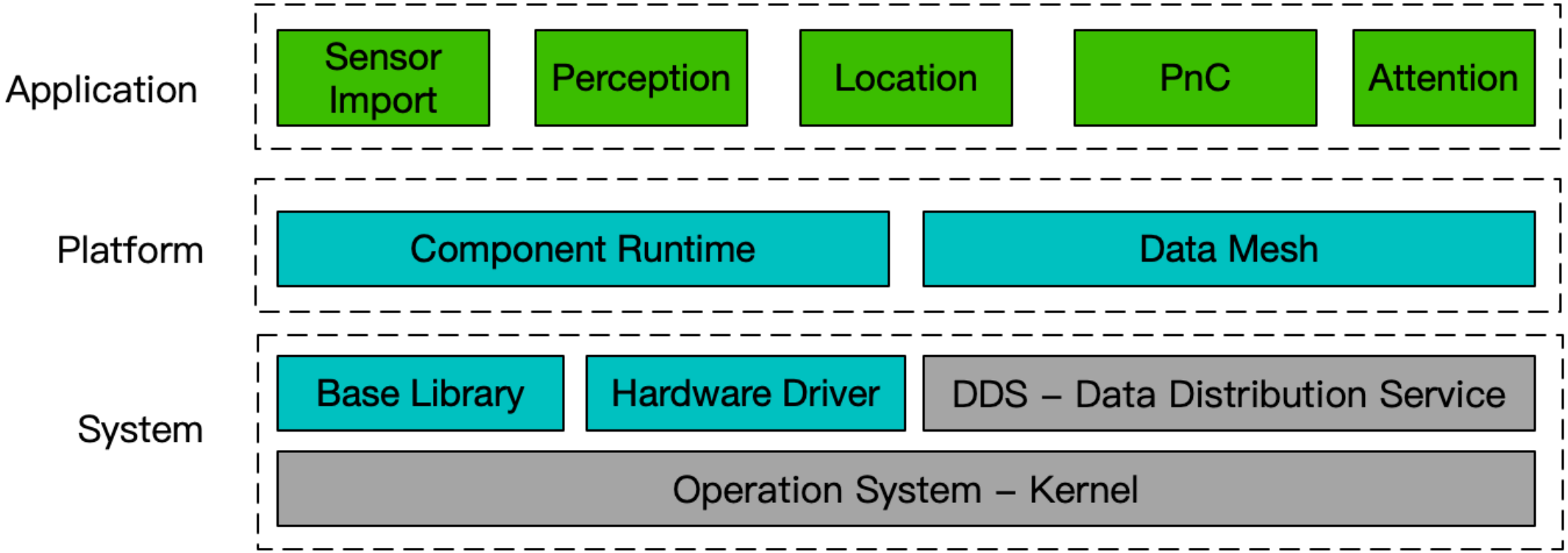
Energon框架

框架发展历史

ROS ==> Cybertron ==> **Energon**

框架	说明
ROS	开源框架Robot Operating System的缩写，致力于构建机器人应用。虽然叫OS但不是操作系统
Cybertron	百度自研框架，用于替代ROS。解决ROS的消息到达乱序，触发不及时，不能连续的事件触发等问题
Energon	百度自研框架，用于替代Cybertron。通过图引擎提升并发能力从而减小时延，通过DataMesh实现声明式的数据共享得数据流更清晰。是当前使用的框架

Energon整体架构



最核心的两部分是数据网络和组件运行时

数据网格DataMesh

声明式的数据共享描述方案，对使用者屏蔽传输和底层存储细节。

概念：

Window：内部数据载体

Snapshot：对Window的访问器

Change：一次数据变更，用于更新DataMesh

DataHandle：用户通过DataHandle与DataMesh交互

读场景：

```C++
int Process(const TriggerInfo& trigger_info) noexcept override {
  uint64_t trigger_version = trigger_info.trigger_version; // 数据版本号，对于感知是LiDAR数据接收时间的纳秒级时间戳
  Accessor<Window<ConstLidarDataConstPtr>> lidar_data_accessor = lidar_data.Access(); // 申请accessor
  const Window<ConstLidarDataConstPtr>::Snapshot& snapshot = *lidar_data_accessor; // 通过Accessor重载的*获取snapshot
  int index = snapshot.GetVersionIndex(trigger_version); // 获取trigger_version在snapshot中对应的位置
  if (index == -1) {
    LOG_INFO << "hdmap get trigger version fail";
    return 1;
  }
  const ConstLidarDataConstPtr& latest_lidar_data = snapshot[index]; // 从snapshot拿到对应trigger_version的数据
  // const ConstLidarDataConstPtr& latest_lidar_data = snapshot.Last(); // 直接从snapshot取最新数据

  // 业务逻辑...
  return 0;
}

DECLARE_INTERFACE(
  DECLARE_INPUT(Window<ConstLidarDataConstPtr>, lidar_data) //输入类型的DataHandle
)
```

写场景：

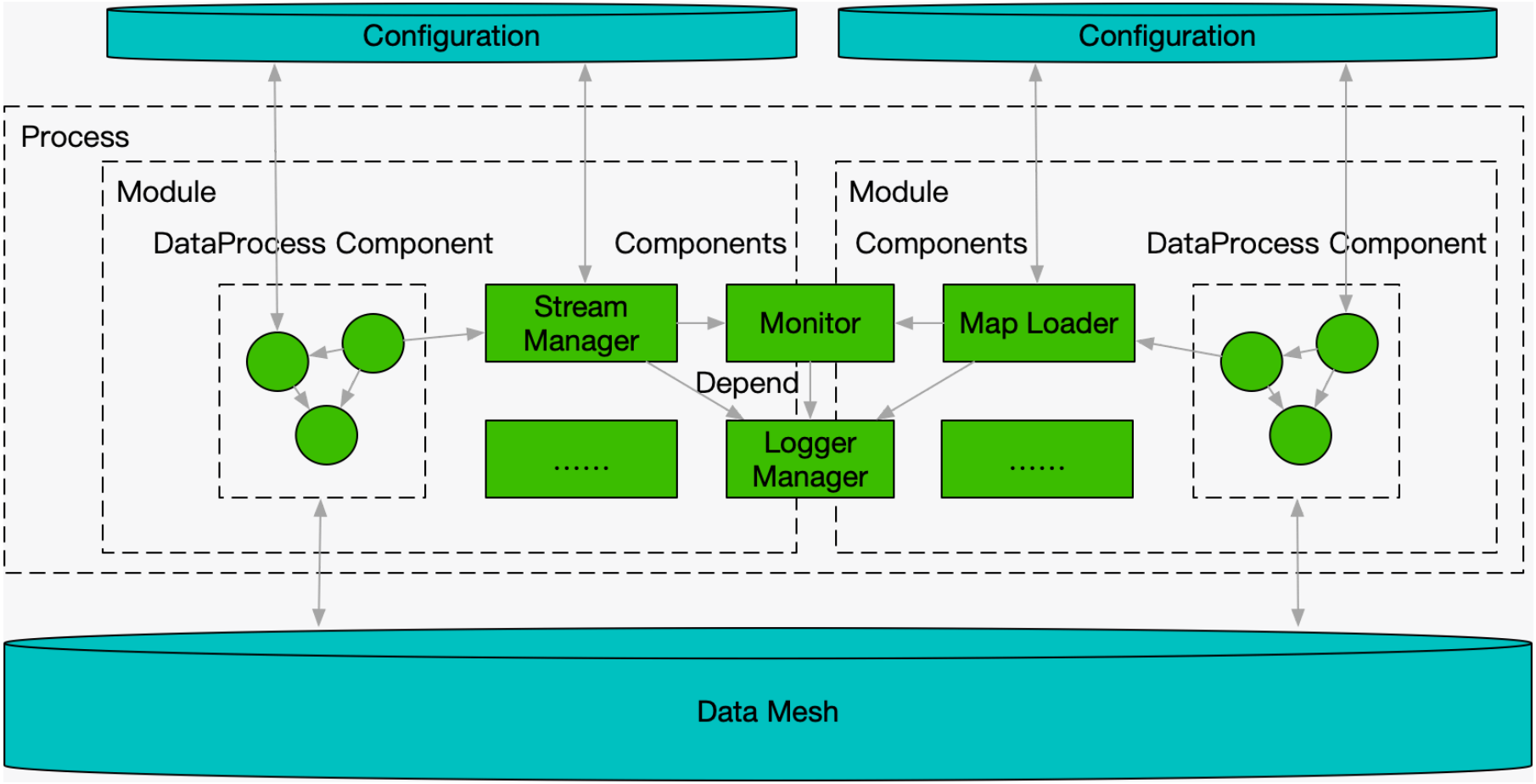
C++

```cpp
int Process(const TriggerInfo& trigger_info) noexcept override {
  uint64_t trigger_version = trigger_info.trigger_version;
  Change<Window<HdmapStructConstPtr>> hdmap_struct_change = hdmap_struct.AllocateChange(); //申请一次数据变更
  hdmap_struct_change.SetVersion(trigger_version); // 根据业务需求，设置数据对应的版本号
  HdmapStructConstPtr& const_hdmap_struct_frame = *hdmap_struct_change;
  // 业务逻辑...
  Policy policy{DISCARD_OUT_OF_ORDER}; // 提交策略：丢弃乱序数据（提交的version小于当前version）
  auto status = hdmap_struct.Commit(policy, ::std::move(hdmap_struct_change)); // 提交数据
  if (status != 0) {
    LOG_ERROR << "Failed to send HdmapStructPtr.";
    return 1;
  }
  return 0;
}

DECLARE_INTERFACE(
  DECLARE_OUTPUT(Window<HdmapStructConstPtr>, hdmap_struct) //输出类型的DataHandle
)
```

组件运行时

模块化，配置化。数据计算，任务触发&调度



Process

实际启动的系统进程，通过启动器launcher执行，可以加载多个module。

对于感知，其进程为compute3d_sched，根据硬件配置的不同启动不同的module。

compute3d_sched进程启动（computing_node_hw50_pandar90_13_600.launch） YAML

```
launcher -c avar_reporter.yaml -c gpu_setup_component.yaml -c gpsbin_component.yaml -c dag_streaming_gnss_rtk.yaml -c dag_streaming_novatel.yaml -c localization_onboard_rt6.yaml -c localization_security_monitor.yaml -c localization_gnss_sins.yaml -c scene_manager_onboard.yaml -c crowdmap_passback.yaml -c onlinemap_onboard.yaml -c lidar_hesai90.yaml -c camera_13c_ads50_struct.yaml -c all_radar_conti430.yaml -c sensor/lidar_hesai90_preprocess_component.yaml -c sensor/gnss_preprocess_component.yaml -c sensor/imu_preprocess_component.yaml -c sensor/wheelspeed_preprocess_component.yaml -c perception/ads50_hesai90_setup.yaml -c perception/blindspot_processor.yaml -c sensor/camera_13c_preprocess_component.yaml -c sensor/radar_conti430_preprocess_component.yaml -c perception/traffic_light_component.yaml -c perception/traffic_fusion_component.yaml -c perception/camera_extrinsics_monitor_component.yaml -c static_transform.yaml -c statistics.yaml -c dag_metric_agent.yaml -c tc_trace.yaml -c compute3d_sched.yaml -p compute3d_sched
```

Module

组件模组，包含多个组件及配置

Component

最终一个进程由多个组件构成。

常见组件类型由以下几种：

通用组件

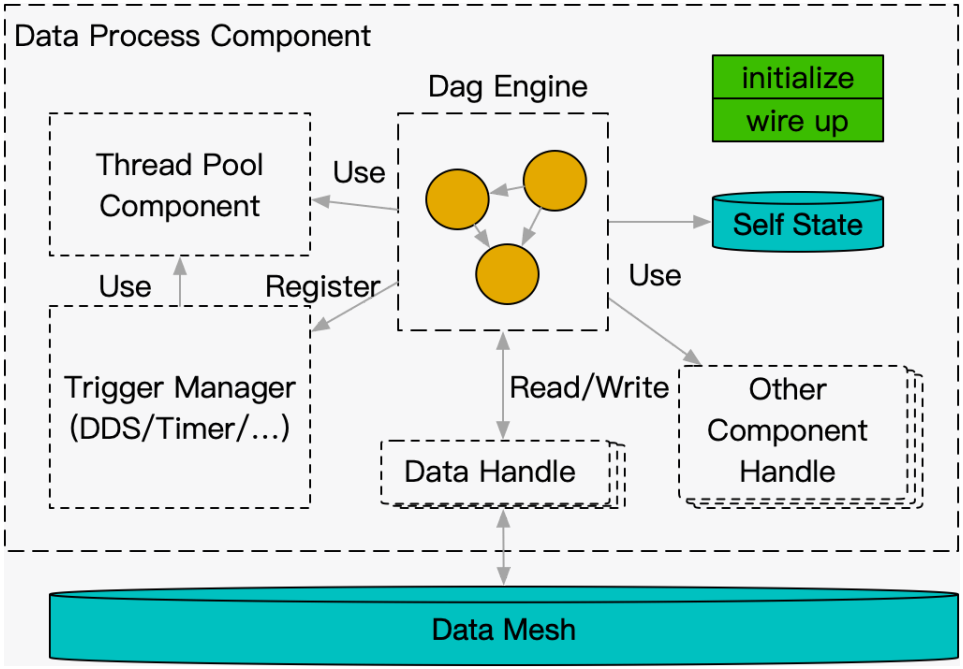
最简单的组件，一个普通的类即可注册为一个通用组件。一般用于非数据驱动的功能模块，也可用于既不接收消息也不发送消息的场景。

数据处理组件（DataProcessComponent）

支持实际的计算逻辑，支持对DataMesh的访问。

分为消息驱动和时间驱动两类。

DAG组件(Data Process Component)



继承于数据处理组件，引入了图引擎（ GraphEngine ），支持对处理函数进一步进行配置化的DAG组网。

图的每一个顶点是一个算子，对应一个processor类。

DAG是感知模块里最常用的组件。

单调递增Dag组件

一种互斥的单调递增的Dag组件的代理。代理里面的对象池持有一个Dag组件对象，并且会丢弃乱序的帧。

主要功能：

- 通过大小为1的对象池，保证线程安全；
- 丢弃乱序的帧，保证单调递增

典型应用：TrackerProcessor

DataMesh数据同步组件

GetRecentDagProcessor：弱同步。找不到直接返回

AlignDagProcessor：强同步，带超时时间。超时时间内反复遍历查找，直到找到数据。超时后返回超时

CommitDagProcessor：用于向DataMesh发送数据

感知数据流

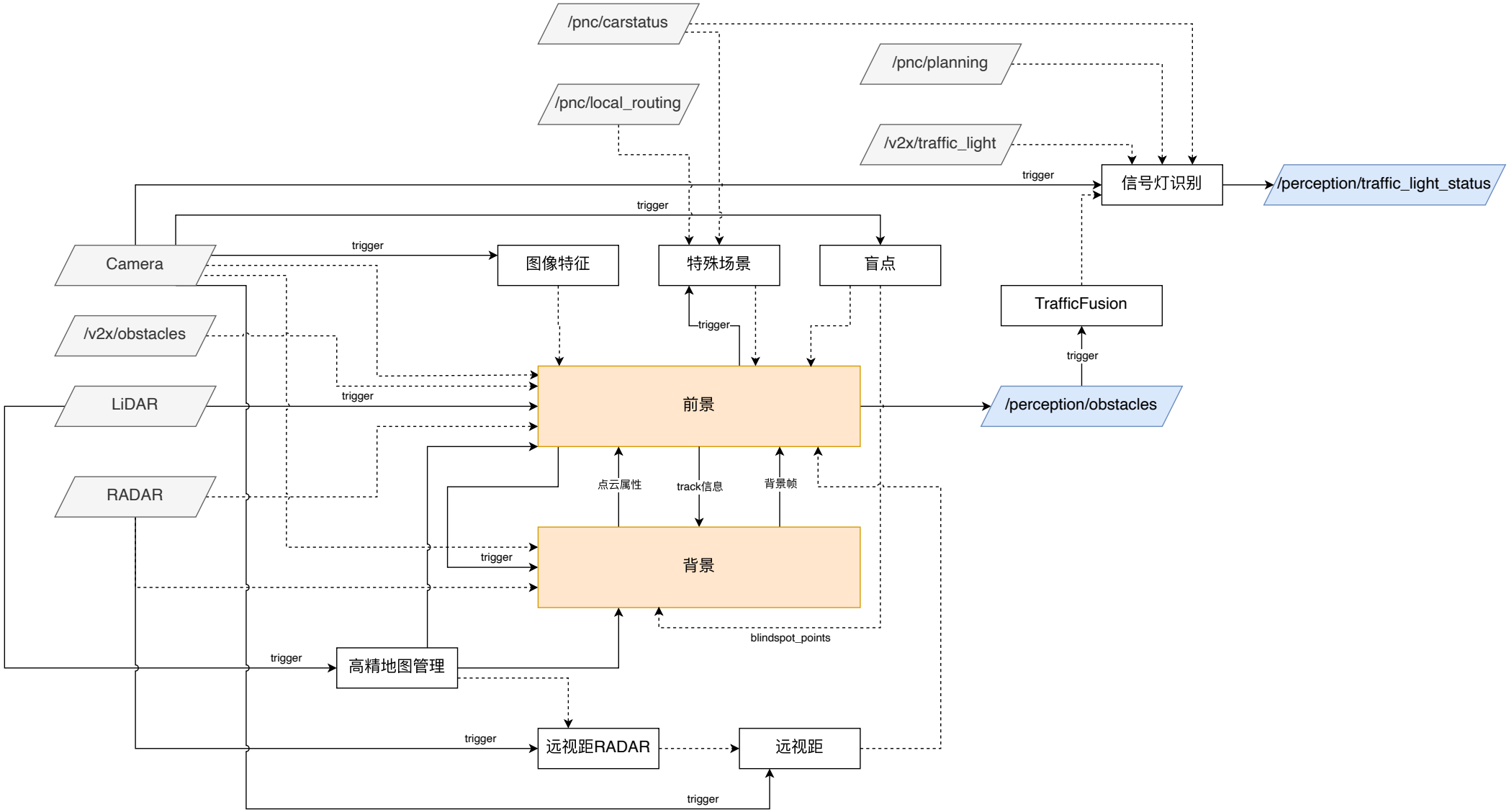
组件

以ads50_hesai90为例，从ads50_hesai90_setup.yaml可知由以下组件构成：

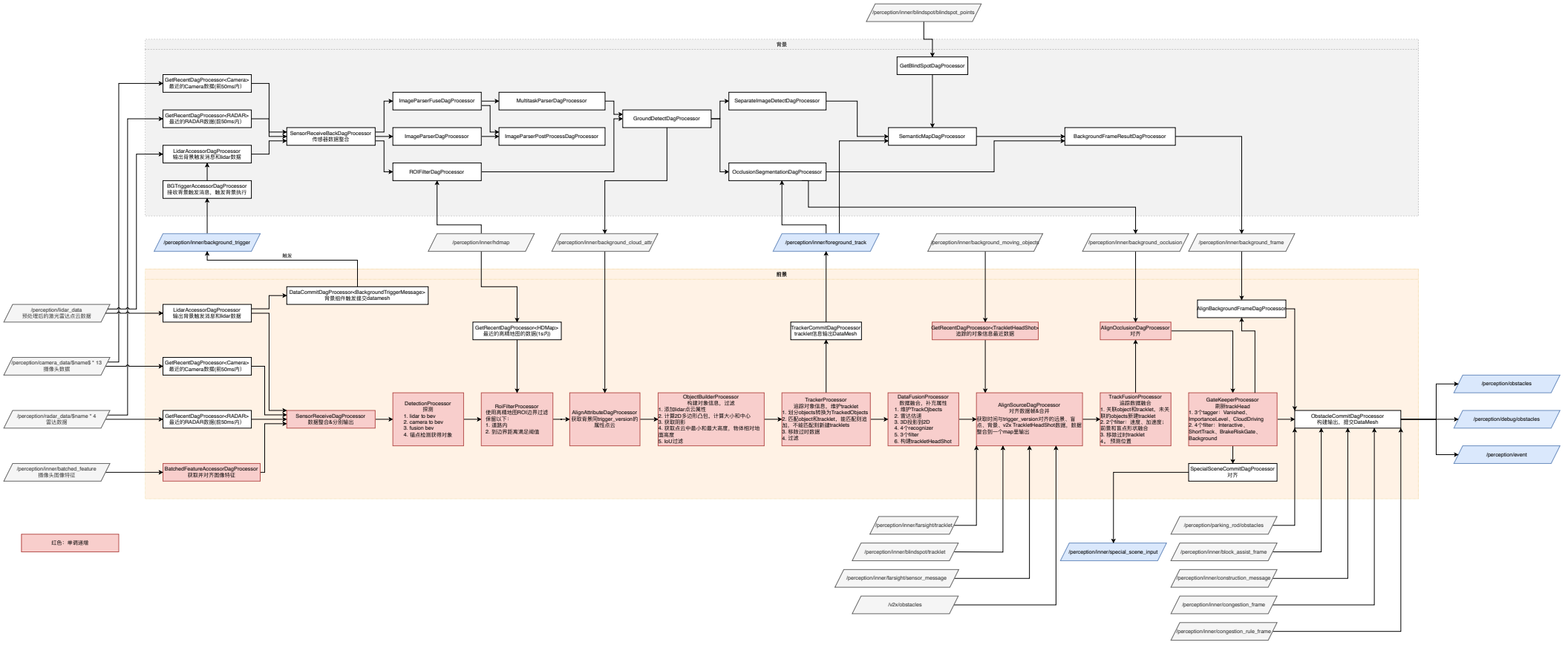
组件	yaml配置	类型	说明
ForegroundComponent	foreground_hesai90_processor.yaml	DagComponent	前景信息处理，主要
inference_gflag	foreground_hesai90_processor.yaml	GflagLoader	截入inference_eng
BackgroundComponent	background_hesai90_processor.yaml	DagComponent	主流程：背景信息处
SpecialSceneComponent	special_scene_hesai90_processor.yaml	DagComponent	特殊场景（堵车、放
BlindSpotComponent	blindspot_processor.yaml	DagComponent	盲区感知
FarSightComponent	farsight_processor.yaml	DagComponent	远视距
HdmapManagerComponent	hdmap_manager_component.yaml	DataProcessComponent	使用LiDAR点云查询
hdmap_gflag	hdmap_manager_component.yaml	GflagLoader	解析并载入高精地图
$camera_sensor_id$_image_feature_service_component	image_feature_service_hesai90_processor.yaml	DagComponent	camera图像特征提
FeatureBatchComponent	image_feature_service_hesai90_processor.yaml	DagComponent	camera图像特征对

12	RadarPerceptionComponent	farsight_radar_processor.yaml	DagComponent	RADAR数据处理，
13	SceneManagerDagProcessor	scene_manager_processor.yaml	DagComponent	设置场景模式
14	HmiObstaclesDagProcessor	hmi_obstacles_processor.yaml	DagComponent	障碍物人机界面处理
15	PointCloudQualityComponent	point_cloud_quality_hesai90_processor.yaml	DagComponent	LiDAR点云信号质量
16	ImageQualityComponent	image_quality_processor.yaml	DagComponent	Camera图像信号质
17	ParkingRodComponent	parking_rod_processor.yaml	DagComponent	停车场拦杆识别
18	$sensor_name$_quality_assess_component	radar_quality_processor.yaml	DagComponent	RADAR质量评估
19	roi_serve_onboard	roi_serve_onboard.yaml	普通	?
20	ROI_gflag	roi_serve_onboard.yaml	GflagLoader	?
21	roi_client_onboard	roi_client_onboard.yaml	普通	?

整体数据流


/pnc/carstatus
/pnc/planning
/pnc/local_routing
/v2x/traffic_light
信号灯识别
/perception/traffic_light_status
trigger
Camera
图像特征
特殊场景
盲点
TrafficFusion
/v2x/obstacles
LiDAR
前景
/perception/obstacles
RADAR
点云属性
track信息
背景帧
背景
blindspot_points
高精地图管理
远视距RADAR
远视距

前景数据流


/perception/inner/blindspot/blindspot_points
GetBlindSpotDagProcessor
背景
GetRecentDagProcessor<Camera>
最近的Camera数据(前50ms内)
GetRecentDagProcessor<RADAR>
最近的RADAR数据(前50ms内)
LidarAccessorDagProcessor
输出背景触发消息和lidar数据
BGTriggerAccessorDagProcessor
接收背景触发消息，触发背景执行
SensorReceiveBackDagProcessor
传感器数据整合
ImageParserFuseDagProcessor
ImageParserDagProcessor
ROIFilterDagProcessor
MultitaskParserDagProcessor
ImageParserPostProcessDagProcessor
GroundDetectDagProcessor
SeparateImageDetectDagProcessor
OcclusionSegmentationDagProcessor
SemanticMapDagProcessor
BackgroundFrameResultDagProcessor
/perception/inner/background_trigger
/perception/inner/hdmap
/perception/inner/background_cloud_attr
/perception/inner/foreground_track
/perception/inner/background_moving_objects
/perception/inner/background_occlusion
/perception/inner/background_frame
触发
前景
/perception/lidar_data
预处理后的激光雷达点云数据
/perception/camera_data/$name$ * 13
摄像头数据
/perception/radar_data/$name * 4
雷达数据
/perception/inner/batched_feature
摄像头图像特征
LidarAccessorDagProcessor
输出背景触发消息和lidar数据
GetRecentDagProcessor<Camera>
最近的Camera数据(前50ms内)
GetRecentDagProcessor<RADAR>
最近的RADAR数据(前50ms内)
BatchedFeatureAccessorDagProcessor
获取并对齐图像特征
DataCommitDagProcessor<BackgroundTriggerMessage>
背景组件触发提交datamesh
SensorReceiveDagProcessor
数据整合&分别输出
GetRecentDagProcessor<HDMap>
最近的高精地图的数据(-s内)
DetectionProcessor
投影
1. lidar to bev
2. camera to bev
3. fusion bev
4. 锚点检测获得对象
RoiFilterProcessor
使用高精地图ROI边界过滤
规则以下：
1. 道路内
2. 到边界距离满足阈值
AlignAttributeDagProcessor
获取背景同trigger_version的
属性点云
ObjectBuilderProcessor
构建对象信息，过滤
1. 添加lidar点云属性
2. 计算2D多边形凸包，计算大小和中心
3. 获取朝向
4. 获取点云中最小和最大高度，物体相对地
面高度
5. IoU过滤
TrackerCommitDagProcessor
tracklet信息输出DataMesh
TrackerProcessor
追踪对象信息，维护tracklet
1. 划分objects将换为TrackedObjects
2. 匹配object和tracklet，提匹配轨迹
加，不能匹配则新建tracklets
3. 移除过时数据
4. 过滤
GetRecentDagProcessor<TrackletHeadShot>
追踪的对象信息最近数据
DataFusionProcessor
数据融合，补充属性
1. 维护trackObjects
2. 雷达估速
3. 3D投影到2D
4. 4个reacognizer
5. 3个filter
6. 构建trackletHeadShot
AlignSourceDagProcessor
对齐数据帧&合并
获取时间与trigger_version对齐的运景，盲
点，背景，v2x TrackletHeadShot数据，数据
整合到一个map里输出
AlignOcclusionDagProcessor
对齐
TrackFusionProcessor
追踪数据融合
1. 关联object和tracklet，未关
联的objects新建tracklet
2. 2个filter：速度，加速度；
前景和盲点形状融合
3. 移除过时tracklet
4。预测位置
AlignBackgroundFrameDagProcessor
GateKeeperProcessor
刷新trackHead
1. 3个tagger：Vanished、
ImportanceLevel、CloudDriving
2. 4个filter：Interactive、
ShortTrack、BrakeRiskGate、
Background
SpecialSceneCommitDagProcessor
对齐
ObstacleCommitDagProcessor
构建输出，提交DataMesh
/perception/obstacles
/perception/debug/obstacles
/perception/event
/perception/inner/farsight/tracklet
/perception/inner/blindspot/tracklet
/perception/inner/farsight/sensor_message
/v2x/obstacles
/perception/inner/special_scene_input
/perception/parking_rod/obstacles
/perception/inner/block_assist_frame
/perception/inner/construction_message
/perception/inner/congestion_frame
/perception/inner/congestion_rule_frame
红色：单调递增

代码库和编译集成

感知代码库

代码库	说明
baidu/adu-perception/perception-future	感知主功能库
baidu/adu-perception/common	公共基础库
baidu/adu-perception/gpu-sched	GPU相关基础库
baidu/adu-perception/sensor	传感器基础数据结构
baidu/adu-perception/sensor-onboard	传感器计算框架接口库
baidu/adu-perception/model	感知模型库

perception-future是感知主功能库，编译产出为libadu-perception-future.so。

baidu/adu-lab/integration是负责整体集成的代码库，会将框架、定位、感知、PnC等全部所需的代码库全部编译，并下载编译产出。

baidu/adu-lab/integration会对各代码库的编译产出进行组装，放到对应的bin、include、lib、dag、launch、conf等目录。

通过bin/start_all_computer.bash完成整体的启动，包含自动驾驶主进程（compute3d_sched、planning_sched等），以及各种监控进程（monitor_gpu.bash、monitor_buddy_info.bash等）。

其中启动自动驾驶主进程是通过cyber_launch进行的，先根据hw、lidar、camera、radar等硬件配置找到launch目录下对应的launch文件，然后通过cyber_lauch start computing_node_xxxx.launch进行启动。

</> YAML

```yaml
<cybertron>
    ...
    <module>
        <name>compute3d_sched</name>
        <type>energon</type>
        <process_name>compute3d_sched</process_name>
        <cgroup>app/compute3d_sched</cgroup>
        <exception_handler>respawn</exception_handler>
        <dag_conf>avar_reporter.yaml</dag_conf>
        <dag_conf>gpu_setup_component.yaml</dag_conf>
        <dag_conf>gpsbin_component.yaml</dag_conf>
        <dag_conf>dag_streaming_gnss_rtk.yaml</dag_conf>
        <dag_conf>dag_streaming_novatel.yaml</dag_conf>
        <dag_conf>localization_onboard_rt6.yaml</dag_conf>
        <dag_conf>localization_security_monitor.yaml</dag_conf>
        <dag_conf>localization_gnss_sins.yaml</dag_conf>
        <dag_conf>scene_manager_onboard.yaml</dag_conf>
        <dag_conf>crowdmap_passback.yaml</dag_conf>
        <dag_conf>onlinemap_onboard.yaml</dag_conf>
        <dag_conf>lidar_hesai90.yaml</dag_conf>
        <dag_conf>camera_13c_ads50_struct.yaml</dag_conf>
        <dag_conf>all_radar_conti430.yaml</dag_conf>
        <dag_conf>sensor/lidar_hesai90_preprocess_component.yaml</dag_conf>
        <dag_conf>sensor/gnss_preprocess_component.yaml</dag_conf>
        <dag_conf>sensor/imu_preprocess_component.yaml</dag_conf>
```

```
        <dag_conf>sensor/wheelspeed_preprocess_component.yaml</dag_conf>
        <dag_conf>perception/ads50_hesai90_setup.yaml</dag_conf>
        <dag_conf>sensor/camera_13c_preprocess_component.yaml</dag_conf>
        <dag_conf>sensor/radar_conti430_preprocess_component.yaml</dag_conf>
        <dag_conf>perception/traffic_light_component.yaml</dag_conf>
        <dag_conf>perception/traffic_fusion_component.yaml</dag_conf>
        <dag_conf>perception/camera_extrinsics_monitor_component.yaml</dag_conf>
        <dag_conf>static_transform.yaml</dag_conf>
        <dag_conf>statistics.yaml</dag_conf>
        <dag_conf>dag_metric_agent.yaml</dag_conf>
        <dag_conf>tc_trace.yaml</dag_conf>
        <dag_conf>compute3d_sched.yaml</dag_conf>
        <dag_conf>patrol_in_3d.yaml</dag_conf>
        <dag_conf>monitor_imu_extrinsic_onboard.yaml</dag_conf>
    </module>
```

参考文档

- Energon指南
- Cybertron指南 主页
- 泛感知运行基础依赖
- 感知新架构Future数据流

串讲问题

1. 主车的位置，速度是感知来实现的吗?

2. energon框架用trigger version驱动DagComponent的一次新的调度，使用 trigger_version获取对应的Data和用snapshot.last获取到的Data是同一个数据吗

3. 向DataMesh提交什么情况下会乱序?

4. 前景只有LidarData是必须数据吗？如果缺少某路，比如ImageFeature缺少，输出结果会有什么影响?

5. 图像特征，从ImageFeatureProcessor给出来的图像特征是怎么表达，图像特征是如何被前景消费和使用的。

6. DetectionProcessor是如何进行障碍物探测的。前景和背景各策略核心计算模块的计算过程，影响算力和耗时的核心指标都有哪些?