RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

Rust China Conf 2020

Shenzhen, China

2020conf.rustcc.cn

Rust 作为汽车软件主语言的探索

The Exploration of Using Rust as a Main Language in Auto Industry

国汽智控软件研发总监 - 肖猛

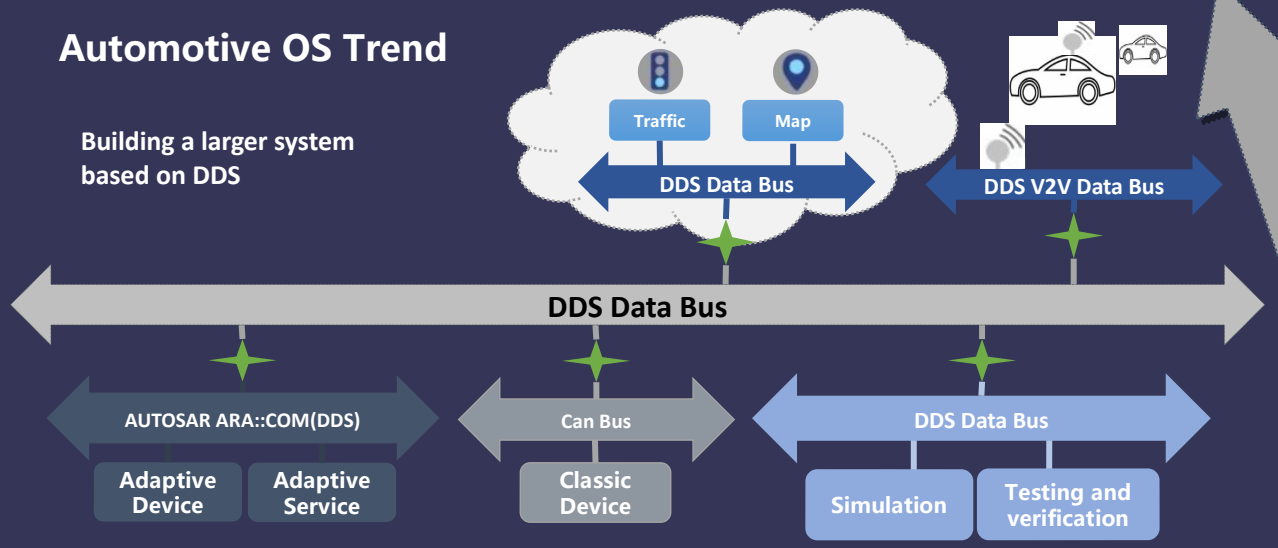
传统汽车软件的特殊之处

- 高实时性要求
  - *很多方面对软件要求硬实时，MCU 软件是主要的解决方案*
- 很强的硬件关联性
- 与汽车电子电器架构密切相关
- 功能安全要求
- 数据传输量少，但信号复杂

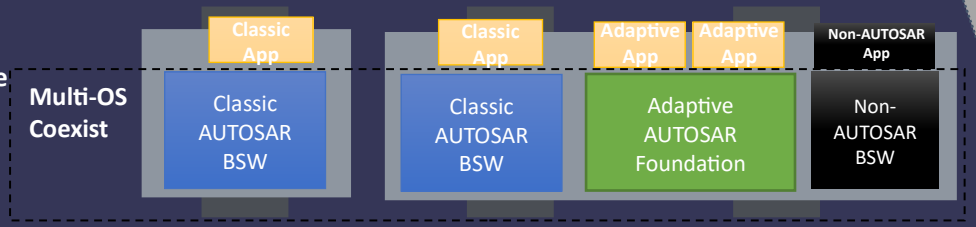
- 一次发布，很少升级
- 软件开发流程复杂而严格
- 要考虑开发、工厂刷写、4S 店维护的全生命周期管理
- Classic AutoSar 逐渐成为 ECU 开发的规范，但成本高企

Automotive OS Trend

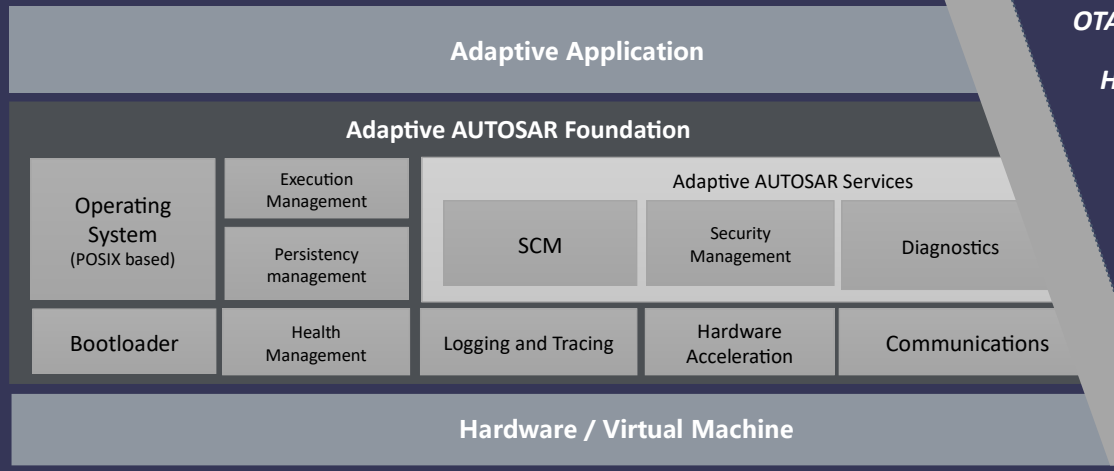
Building a larger system based on DDS



Multiple systems coexist, Classic AUTOSAR supports high safety and high real-time, and Adaptive AUTOSAR supports high-performance computing for parallel processing.



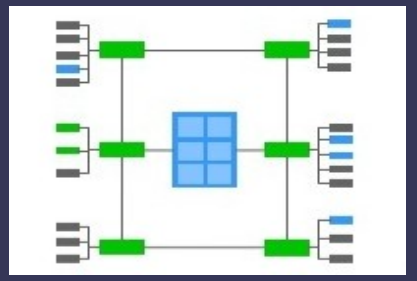
AUTOSAR Adaptive
Support complex intelligent systems with greater flexibility



Evolution of E/E Architectures

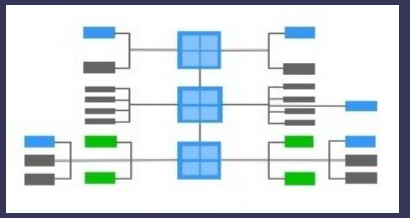
Future

- *IP/Ethernet communication*
- *Centralized application/functions*
- *Computing power for AI*
- *Anything anywhere(sensor/actors)*



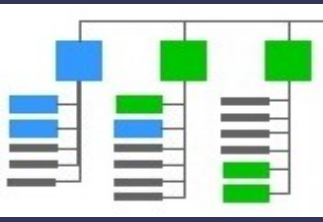
Tomorrow

- *Central computing nodes*
- *Mix of signal based and service orientated communication*
- *Partly centralized functions*
- *Software upgradability*



Today

- *Signal Based communication*
- *System of ECUs*
- *Predictable communication*
- *Function orientated topology*



Communication bandwidth

OTA/Dynamic deployment

Hypervisor/Virtualization

Functional complexity

Sensors/Data Volume

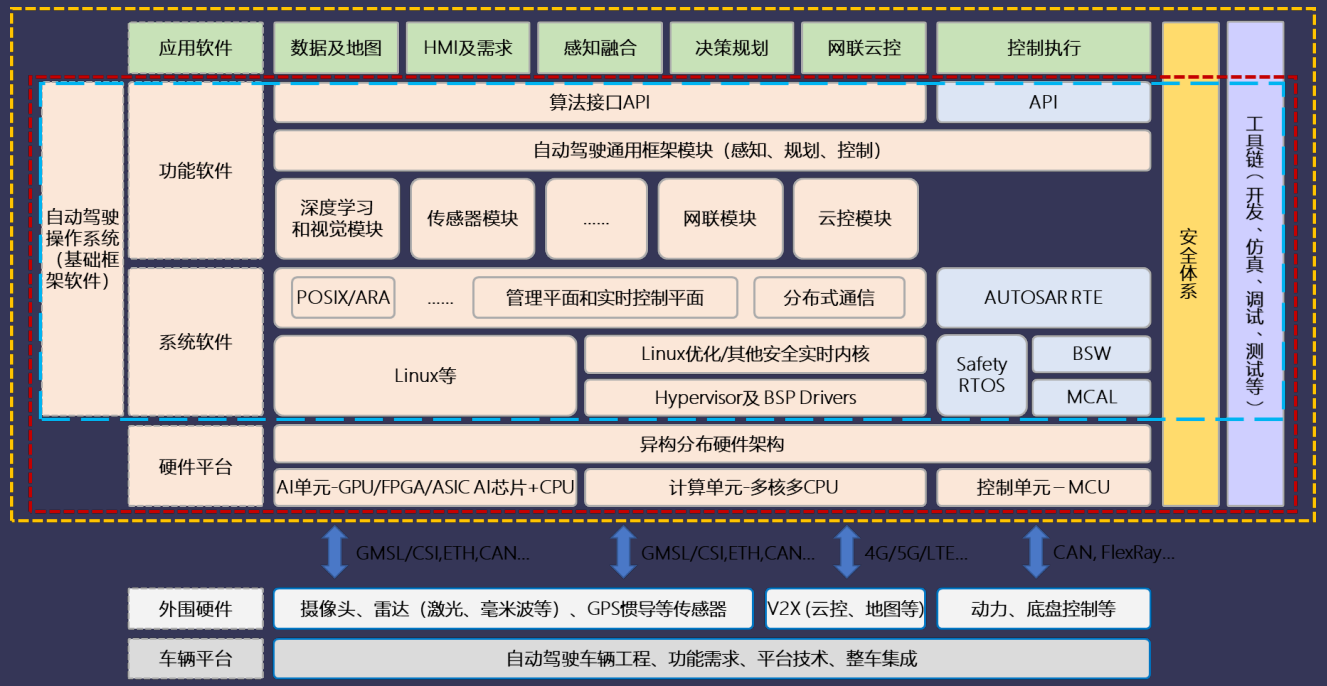
information security

Parallel Computing

Computing power

Functional safety

现代汽车软件的特殊之处



- 分布式异构系统
- 感知算法带来极高的算力要求
- 自动驾驶、娱乐系统都带来大量的数据传输要求
- 自动驾驶应用带来更大的功能复杂度
- 汽车随时在线（互联网）
- 更复杂信息安全和功能安全要求

为什么 Rust 语言适合开发汽车软件

- Rust 作为系统级语言，是 C/C++ 的最佳替代者
- Rust 用于 MCU 开发，一样能达到硬实时要求。同时提供完美的内存安全机制。

  试想一下：使用经过编译器内联优化的闭包，处理中断请求

- C++ 11 带来了更多特性，但是历史包袱太重。建一个 C++11 的团队，还不如建一个 Rust 的团队更靠谱

目前 Rust 在汽车领域的应用现状

- 几乎没听说量产汽车使用
- 有 Rust 方案支持 ROS2 开发

实际项目 - 某自动驾驶应用中间件软件

三个月实现原型

软件架构会跨越多个异构平台：

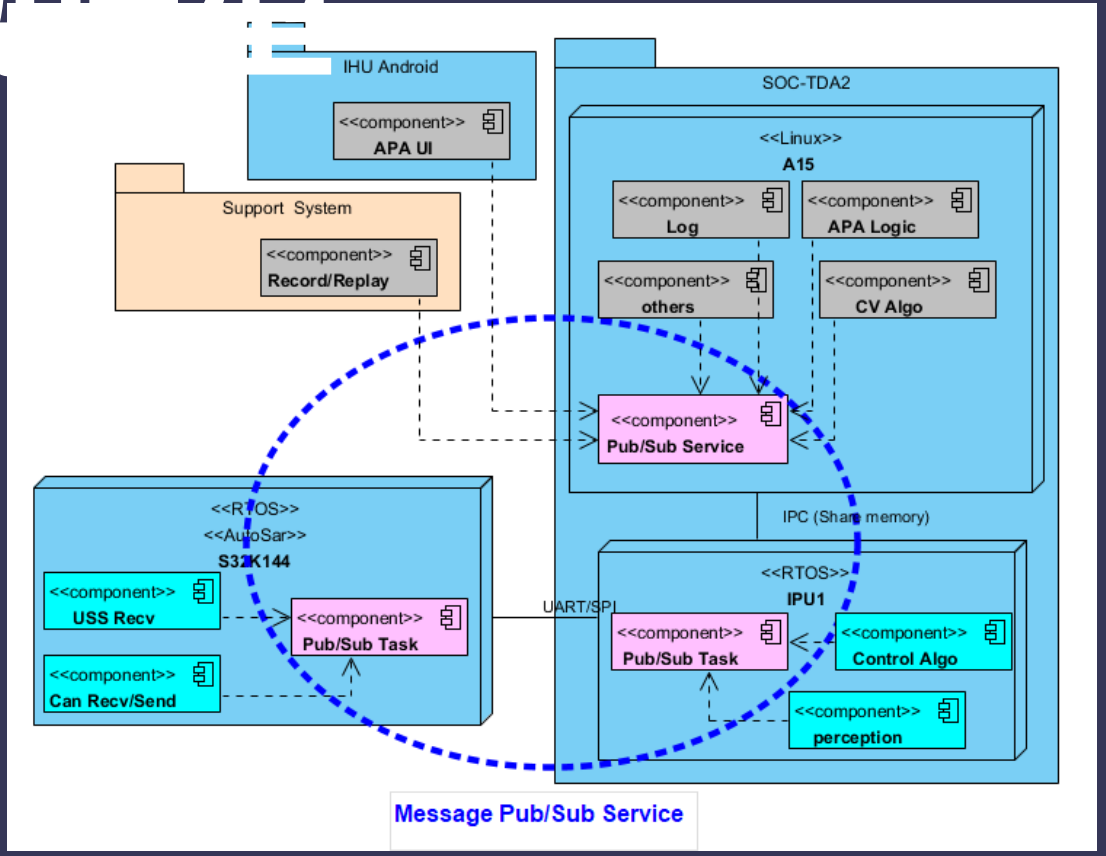
- TDA2 SOC 内的 Cortex A15 运行 Linux
- TDA2 SOC 内的 M4 是 RTOS,
- MCU 上运行 AutoSar 系统或裸机系统。
- IHU 上运行 Android 系统

我们要求：

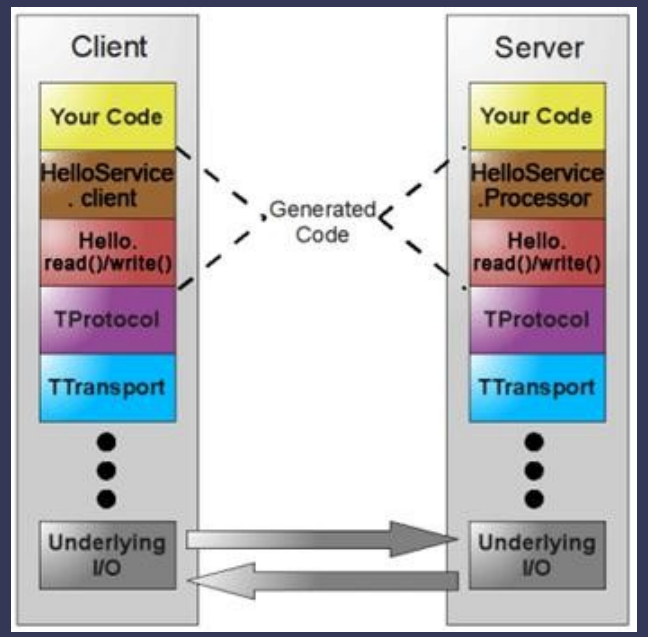
- 运行在不同硬件和 OS 上的各个软件子系统之间需要相互通信，一个子系统发出的消息，会有多个其它子系统需要接收。
- 能够支持 C/C++/Python/Java 等多种开发语言
- 能够解除各子系统之间的直接依赖关系。
- 各子系统之间的通讯协议能够使用一致的方式进行定义。
- 数据序列化和底层的通讯机制对应用逻辑透明。
- 数据传输的质量 (QoS) 会有不同的需求
- 高度的稳定性和数据传输效率
- 运行在 MCU/RTOS 上的软件 通过 26262 认证 (功能安全)
- …… more

一般来讲，支持发布订阅的中间件是解决这类需求的典型方案。

但是获取一个支持我们这种异构平台的发布订阅实现并不是一件容易的事情。



基于 Rust/Thrift 实现的中间件架构



利用 Thrift 提供的底层 RPC 机制

MCU 侧开发全套 thrift 运行库 / 代码生成插件
及
发布订阅中间件

SOC 侧开发使用 Rust 开发发布订阅中间件
使用
Thrift Rust 运行库

Thrift Storm micro(pub/sub)

Topics Management | Message forward | QoS

Support multi-transport

Message Bridge

桥接

Message Bridge On Linux

Thrift Storm Linux(pub/sub)

Topics Management | Message forward | QoS

Support multi-transport

Thrift pure C
on TI RTOS/MCU

UART/SPI Transport | TI IPC Transport | Memory Transport

Runtime Lib

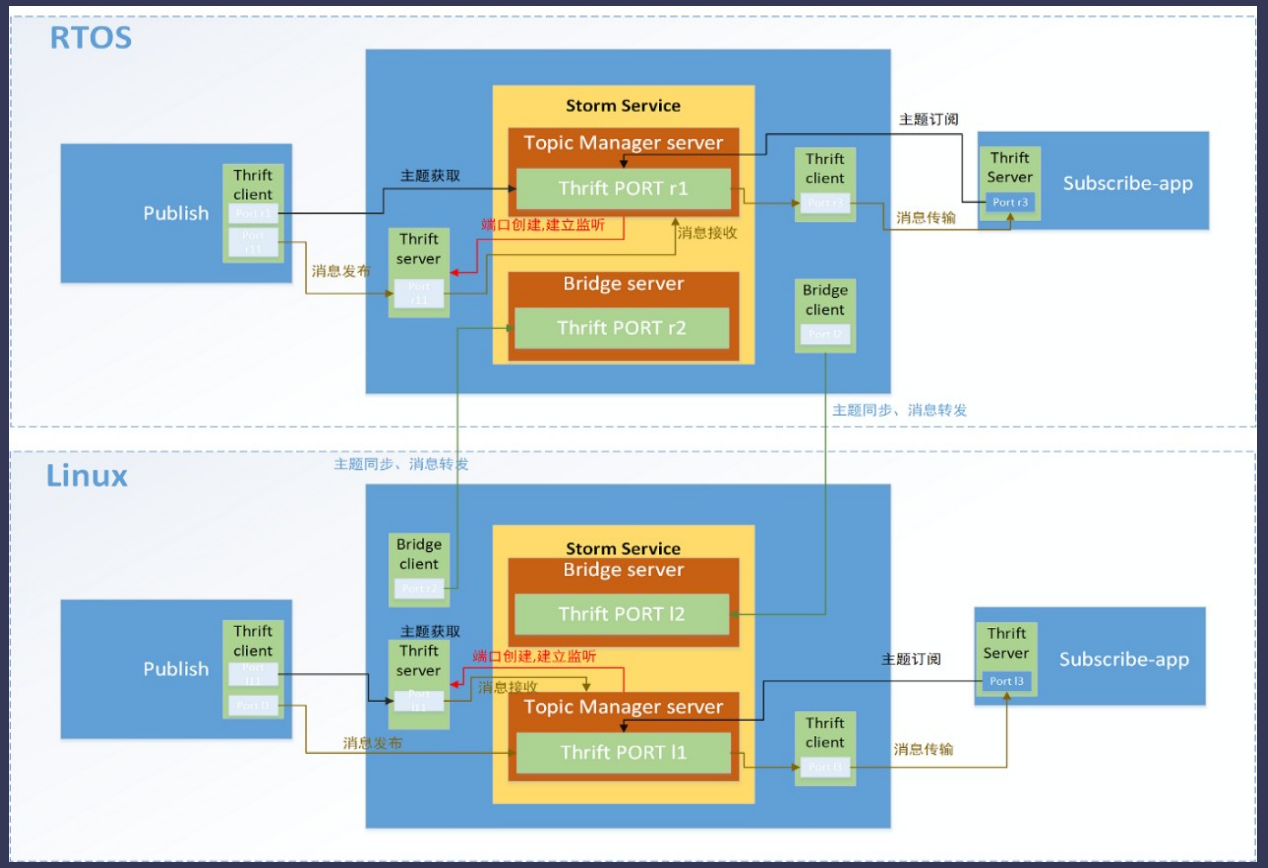
Thrift Rust
Runtime on Linux

Socket Transport | TI IPC Transport | Share MEM Transport

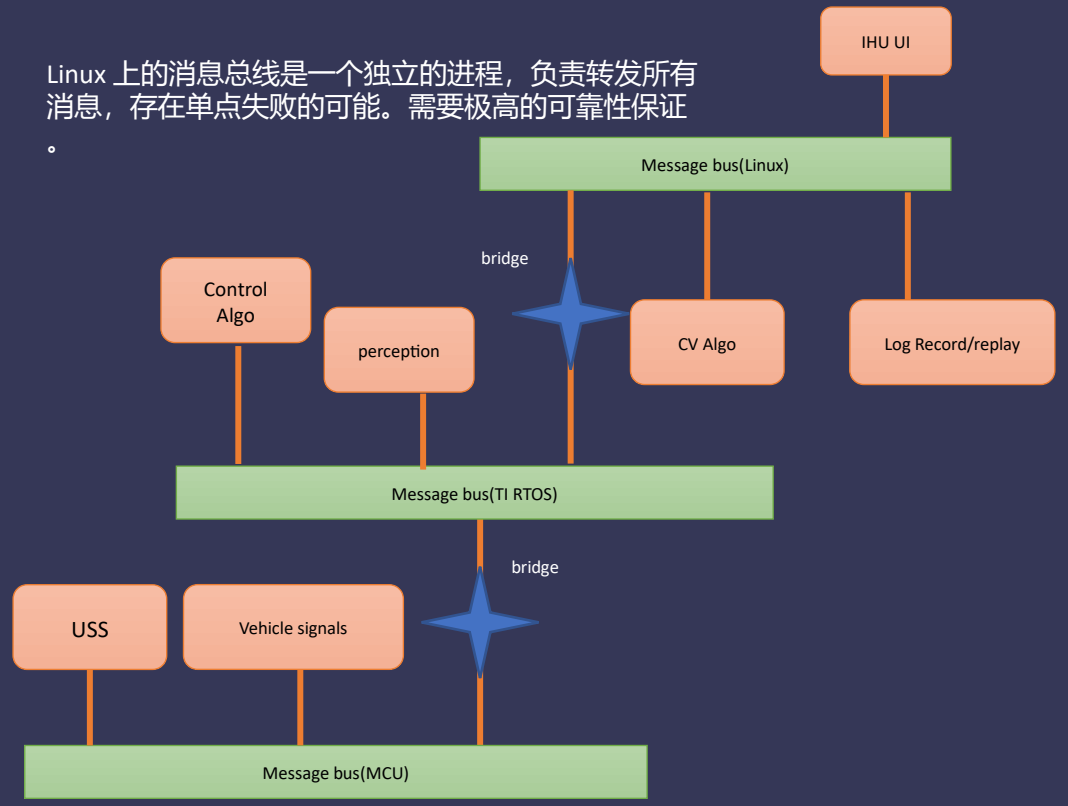
code generate plugin

Tools

基于 Rust/Thrift 实现的中间件架构



Linux 上的消息总线是一个独立的进程，负责转发所有消息，存在单点失败的可能。需要极高的可靠性保证。



每一个独立的发布订阅服务就是一个消息总线，不同总线之间的消息由 bridge 进行转发。

Rust + Actor 带来高可靠性和高性能

I Love

Rust

- 内存安全的系统级语言
- 零开销抽象

Linux 上的发布订阅中心服务使用 Rust Actor 库 actix

Actor

- supervision 特性提供高可靠性
- Actor 模式为并发而生

同一系统中不同进程间用共享内存比 socket 提高 10 倍的通讯速度。

系统中其他进程用 C/C++，测试用 python

Thrift

- 跨语言支持让异构支持、测试更便捷
- 共享内存 Transport 提供高速通信

进一步的改进方向

- 去中心化的分布式发布订阅实现
- 将 Thrift 的底层与 DDS 集成
- 支持更多的 QoS 能力
- MCU 软件用 Rust 重写

Q&A

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳