三代架构

- 背景
- 系统设计目标
- 挑战：
- 三代平台概要设计
- 模块设计
    - 统一共享数据
        - 数据采集
        - 数据使用
    - 车机交互
        - ASR接入，待融合两者设计
        - TTS接入
    - 融合中控
        - 流式交互抢跑
        - 端云一致性保障
    - 灰度实验室
    - E2E性能优化项
        - 流式抢跑，如上
        - nativeApi优化
- TODO：

背景

伴随语音创新业务爆炸式发展，急需升级三代语音架构，用以集成自研ASR、超高自由度、多人全双工、免唤醒、认知计算等创新业务能力

打造语音行业最领先、最智能、最快速的对话系统

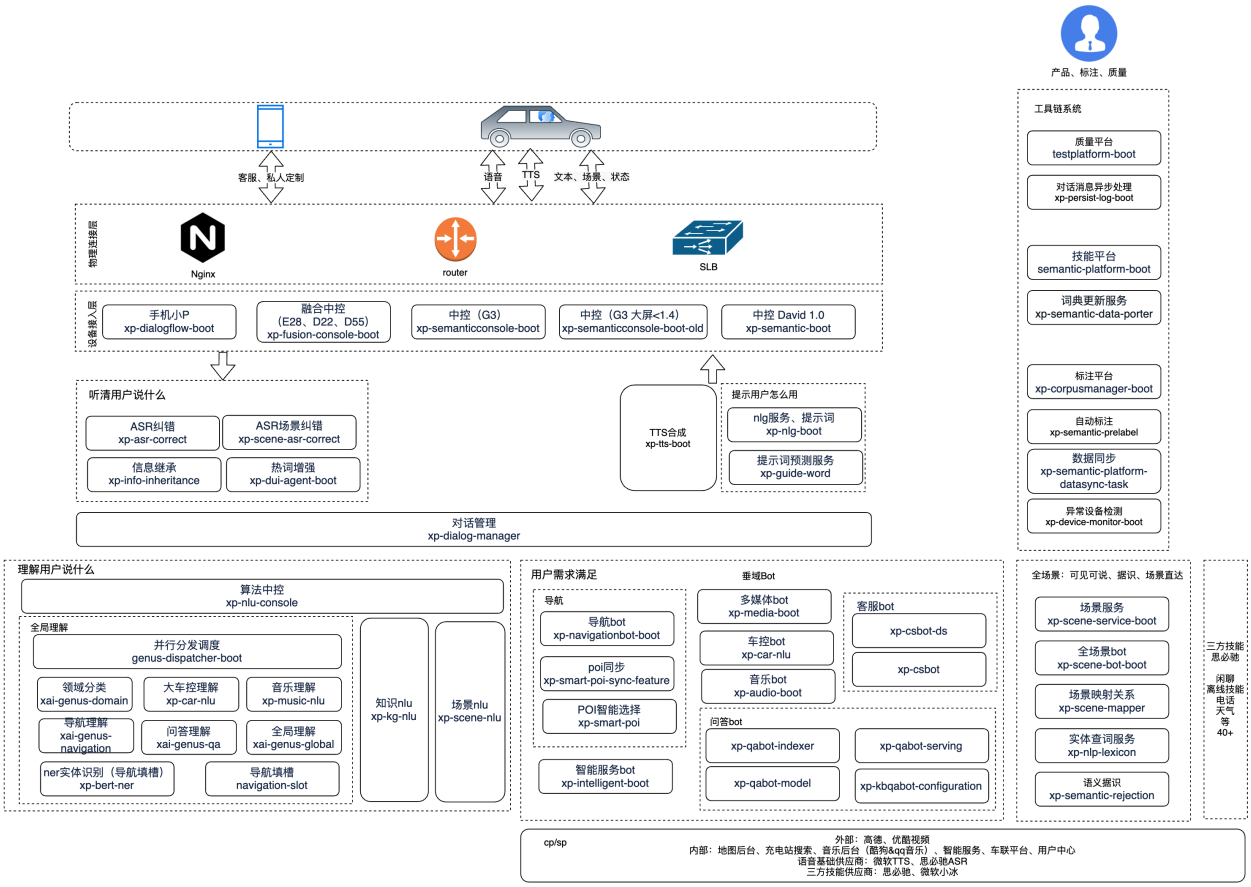
系统设计目标

- 高可用性：持续稳定生产系统，平滑过渡大创新落地，提供全流程灰度实验室、延迟管控、服务降级、限流熔断等基础能力
- 端云一致：保障在本地召回及弱网状态，云端状态与客户端状态保持最终一致，云端支持多阶段提交、回退、终止能力
- 实时感知：支持多业务数据信号实时采集诉求，强感知数据亚秒级返回，弱感知小时级别
- 更低延迟：性能要求E2E 500ms，最快小P，E38-三代性能专题-- E2E 500ms
- 轻量协议：解决未来150+微服务之间交互痛点，协议灵活可扩展
- 多车型适配：E38、E28a（22年中）、F30（规划中）车型隔离部署上线

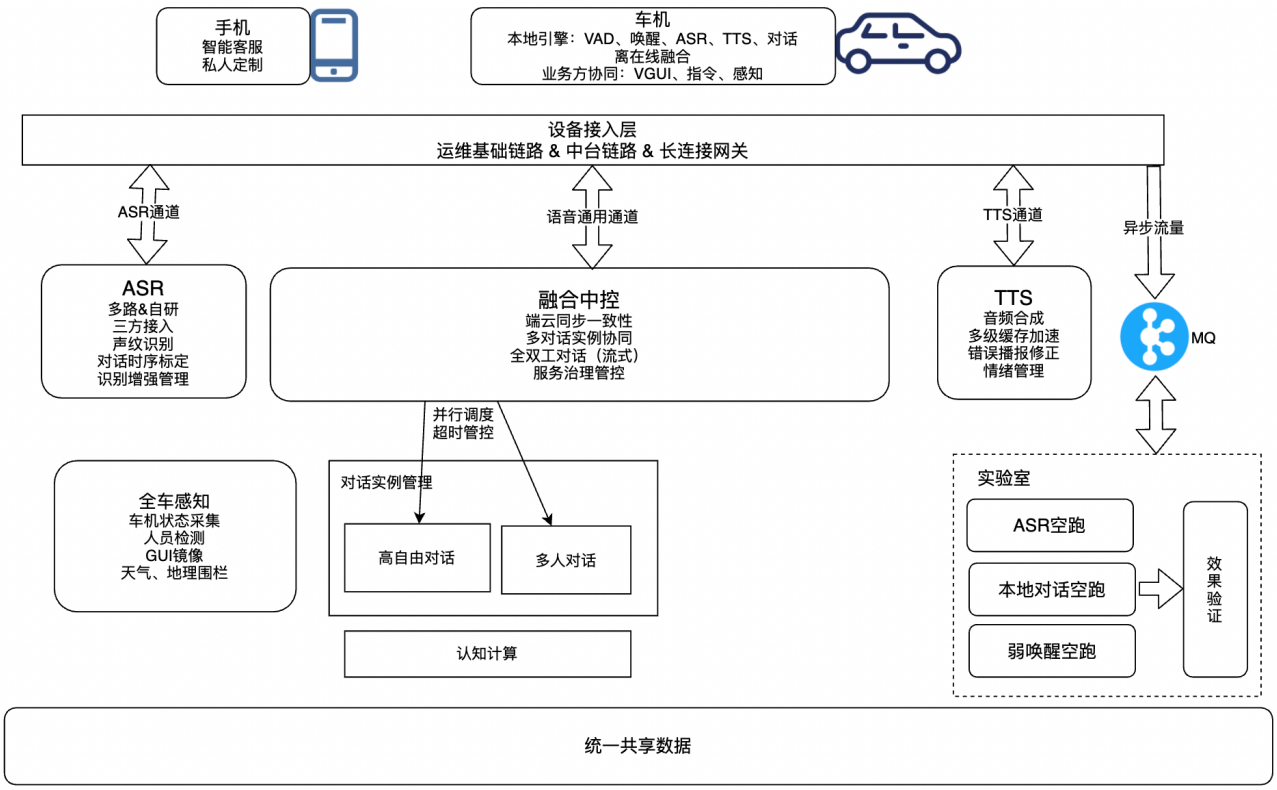
挑战：

- 多业务线交互复杂，调用关系错综复杂，如何简化交互？
- 多人与高自由度融合难度大，怎么保障存量业务前提下稳定交付大创新？
- 服务爆炸式增长，保障创新业务功能同时，如何保障生产质量，如何快速定位问题？待进一步设计

二代架构现有模块图，未来会2-3倍，甚至更多的扩张



三代平台概要设计

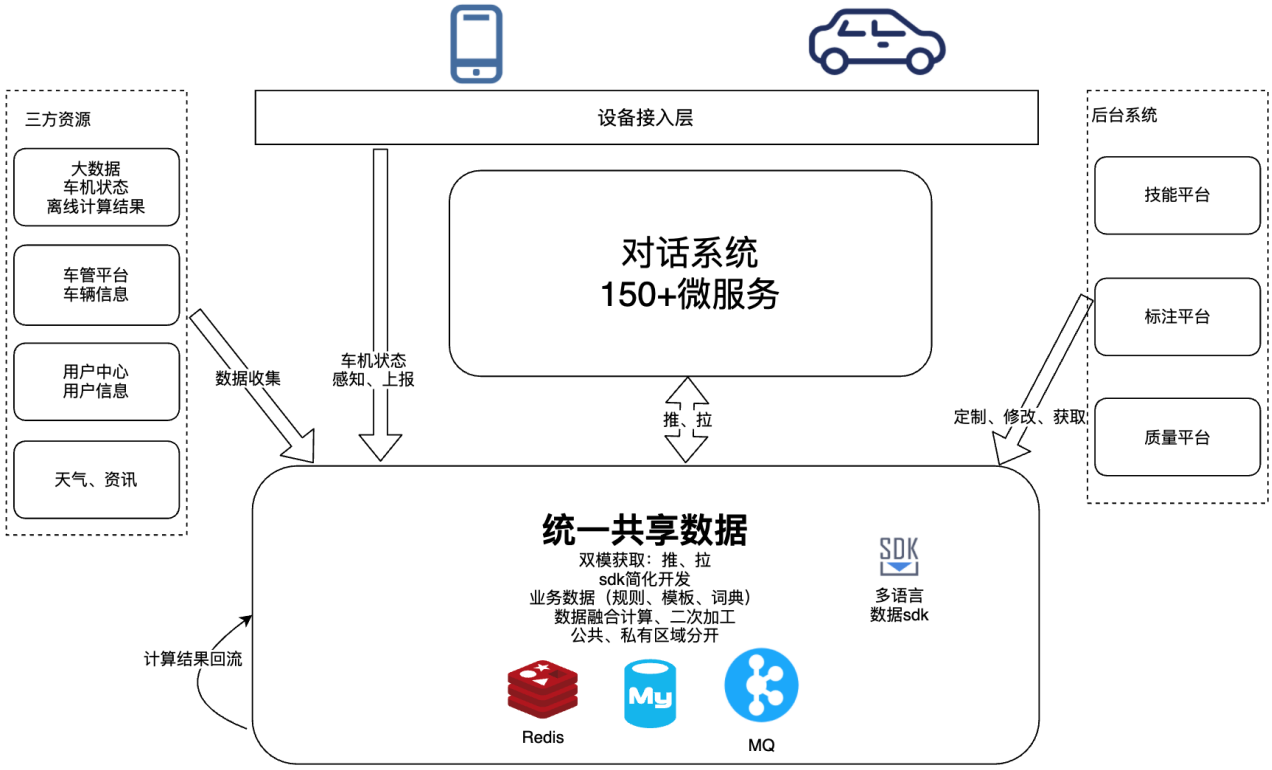


模块设计

统一共享数据

功能：提供底层数据共享，让所有业务拥有方便存储和快速获取数据能力，专注业务开发，降低业务依赖其他服务中间结果异步等待复杂度，充分信息共享

特性：推、拉模式结合，改变传统模式等待结果后显性强依赖调用，降低调用复杂度

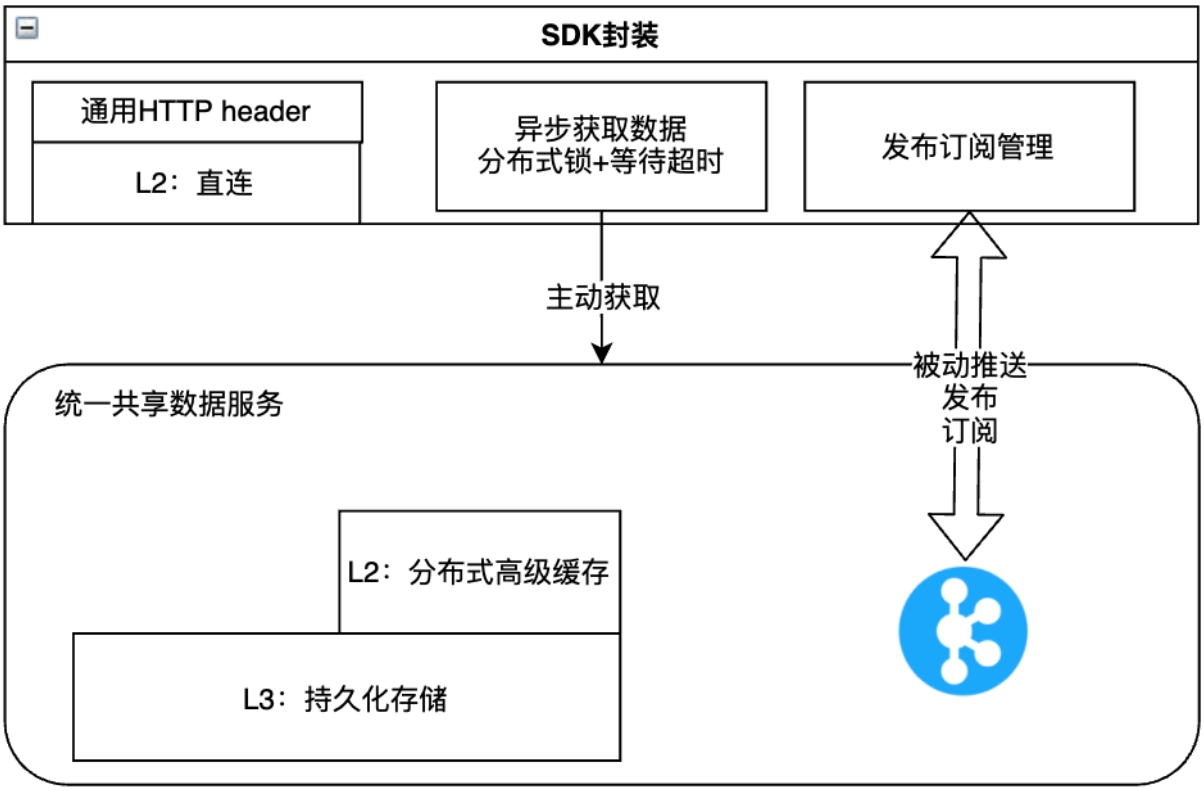


数据采集

数据资源	来源	实时性	缓存级别
实时车机状态 场景感知人数判断初步分析 当前播放歌曲 L2 当前目的地 L2 经纬度、电量 L2 说话语序 L2 播报语序 L2	语音app文本链路上报	亚秒级	
车辆配置 L1 用户基础信息 L1 订单信息 L3 用户收藏 L2 用户习惯数据 L3	大数据周期同步	准实时（小时）	
外部环境感知 L3 天气、路况 L3	通过API周期性感知	准实时（小时）	
规则数据			

数据使用

SDK能力细分

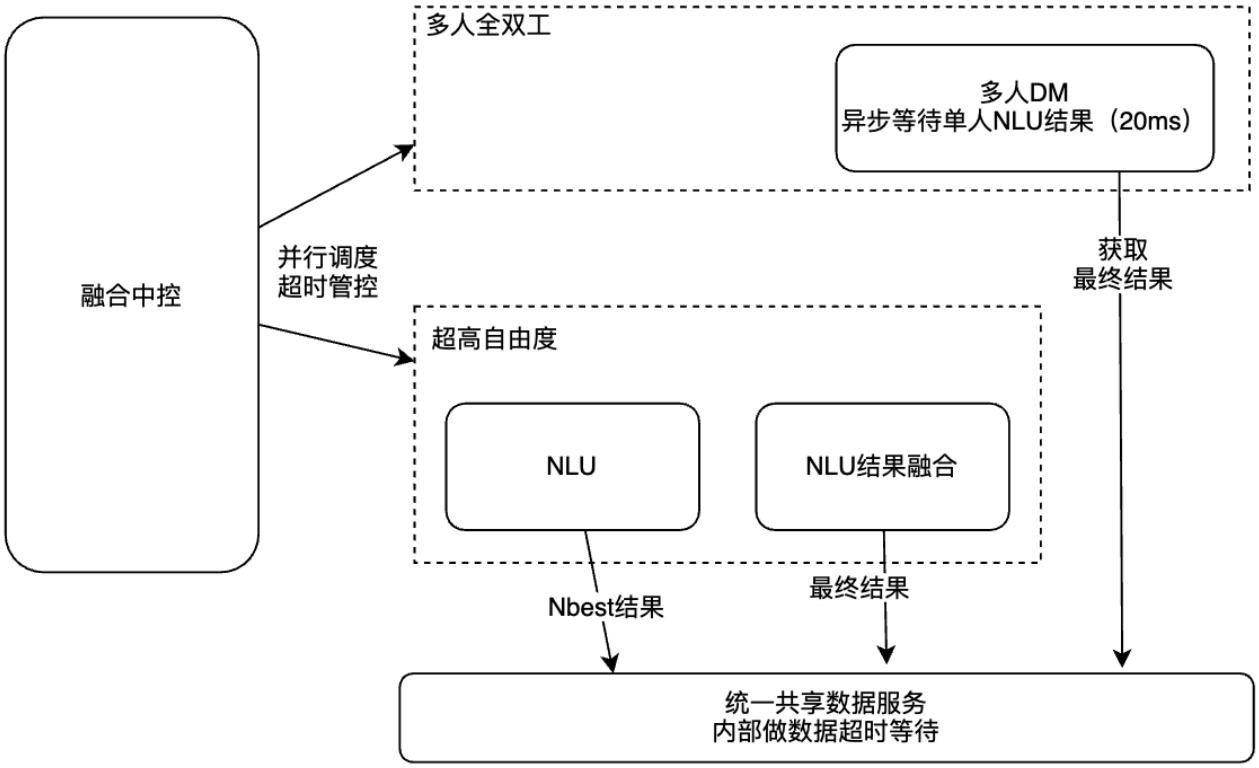


使用场景介绍

- 主动拉数据：

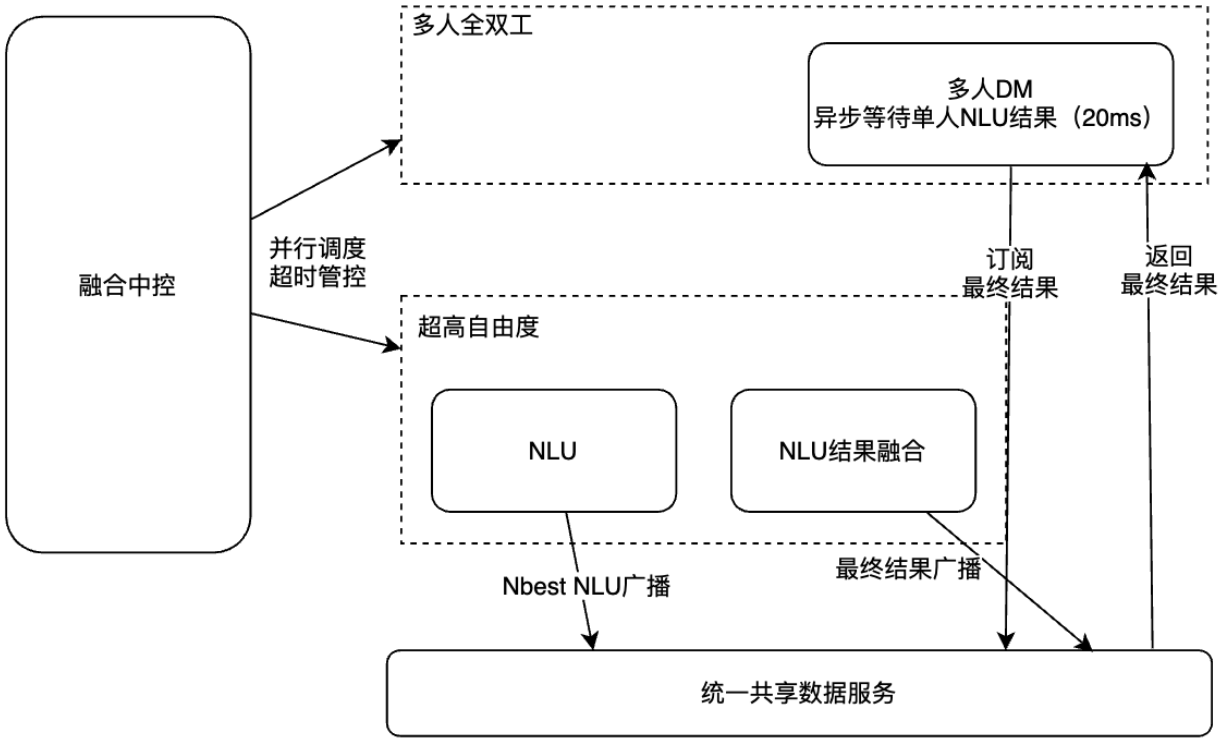
获取非实时数据，车机基本信息、用户基本信息、用户对话历史等

- 业务逻辑强依赖，同步获取结果



- 非强依赖异步获取结果：

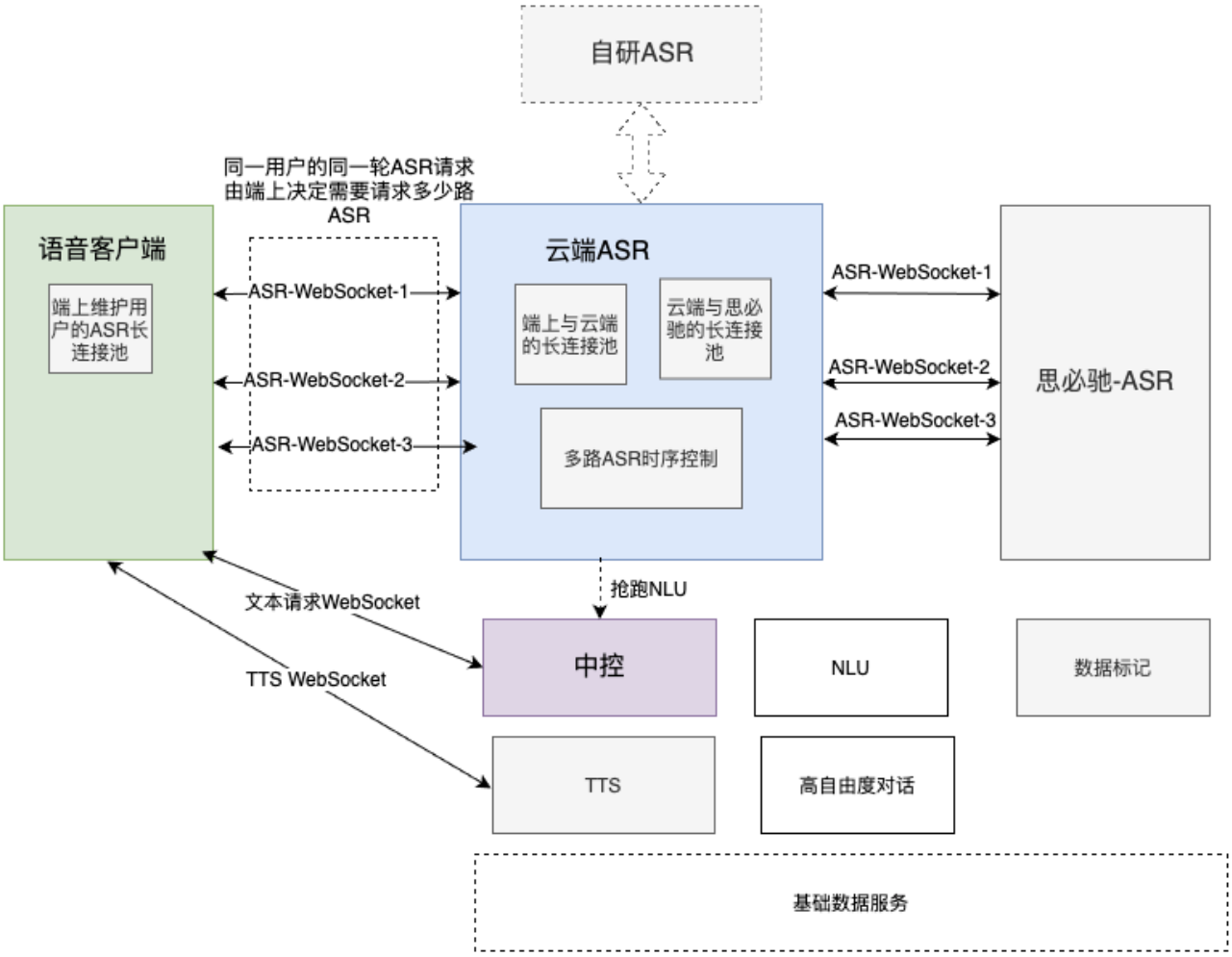
分布式锁等待获取其他模块中间结果，多人依赖单人NLU最终结果，则采用分布式等待锁方式，异步等待结果，如规定时间内无返回则超时



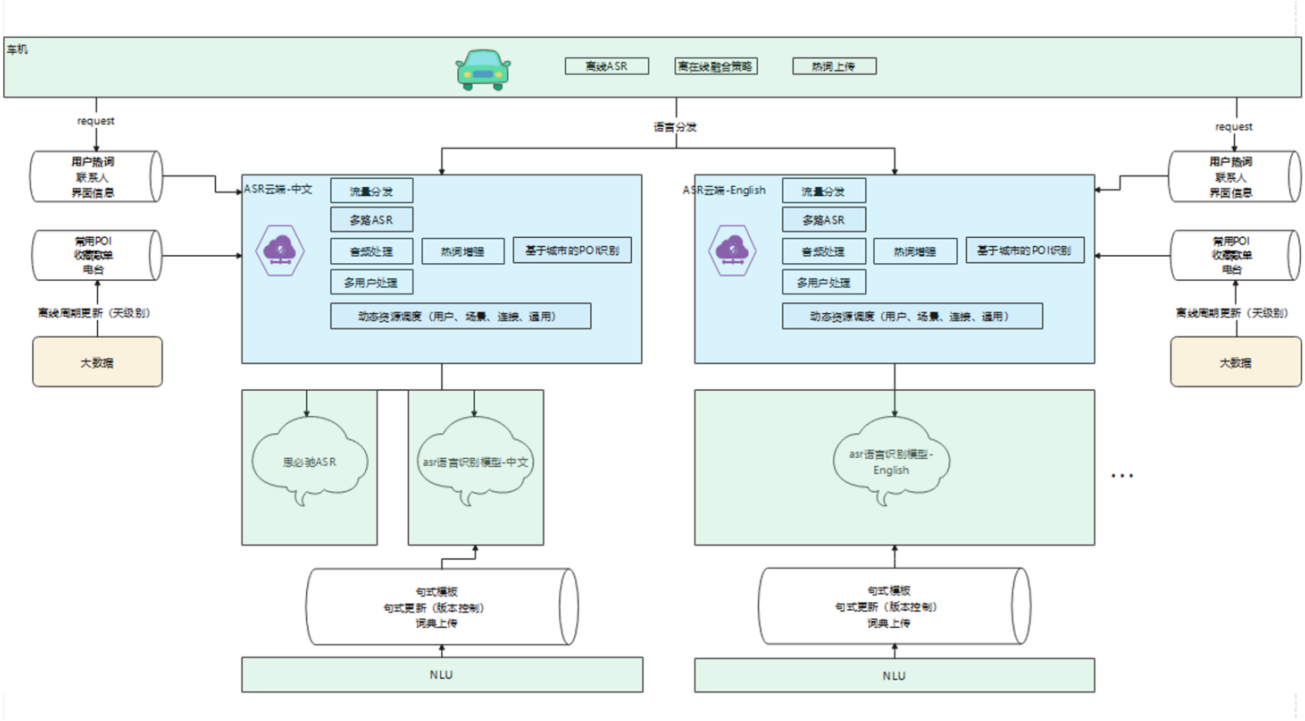
车机交互

ASR接入，待融合两者设计

多路ASR设计文档，优先E28落地收集数据



自研ASR的云端服务设计文档，将ASR算法能力工程化落地

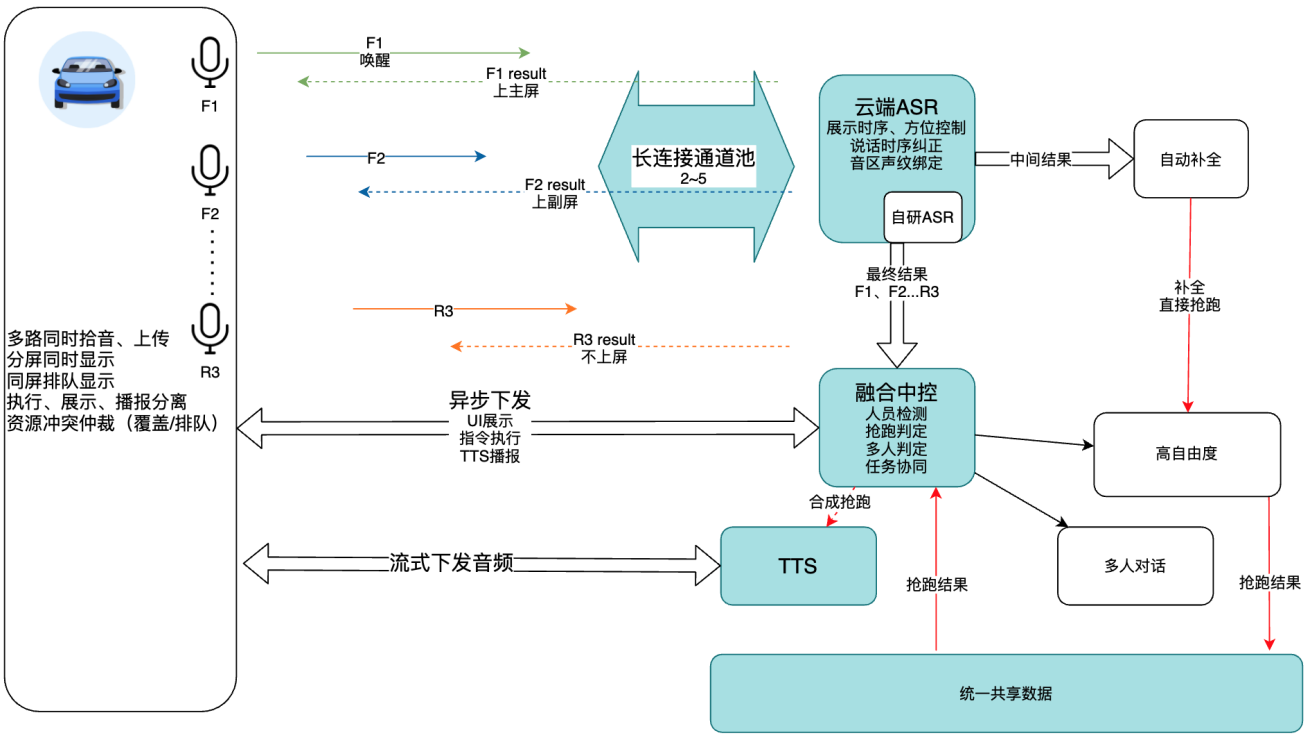


TTS接入

云端tts设计方案-微软，情绪管理设计待完成。

融合中控

- **流式交互抢跑**



补全直接抢跑炒近路，不需要经过asr服务，融合中控，最终由融合中控进行抢跑判定

抢跑判定方式：

中间结果	补全结果	ASR最终结果	匹配结果
明天天	明天天气怎么样 weather	明天天气怎么样 weather（tomorrow）	字符串匹配 补全命中
		明天天气好不好 weather（tomorrow） 需要确认占比后是否可行？？ 李晨延 NLU相同时，结果是否相同？？	NLU全匹配 补全命中
		明天天气挺好的 chat	意图匹配失败 补全失败

- **端云一致性保障**

离线处理

弱网处理

云端处理状态：

流式补全抢跑实例、ASR结果实例

客户端真正执行结果上报：各个业务做融合，

时序处理？

灰度实验室

通过异步流量分发机制，解耦线上生产业务，可准实时的进行新业务灰度验证功能

多渠道效果验证

- 监控大盘业务埋点验证
- 结果发送异步消息队列后标注平台效果评估
- 大数据日志埋点比对

需要增强能力：

增加HTTP流量调度，上下文查询能力

E2E性能优化项

流式抢跑，如上

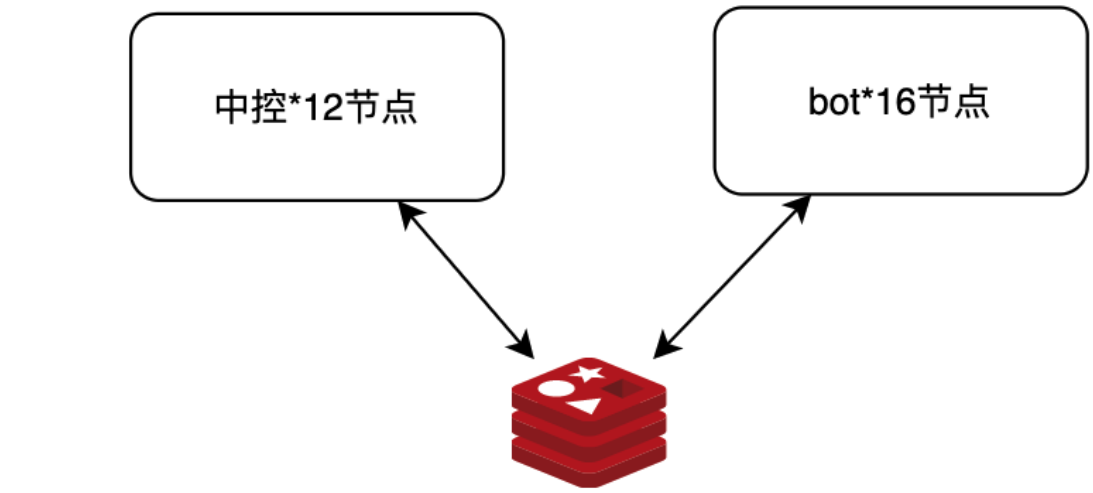
nativeApi优化

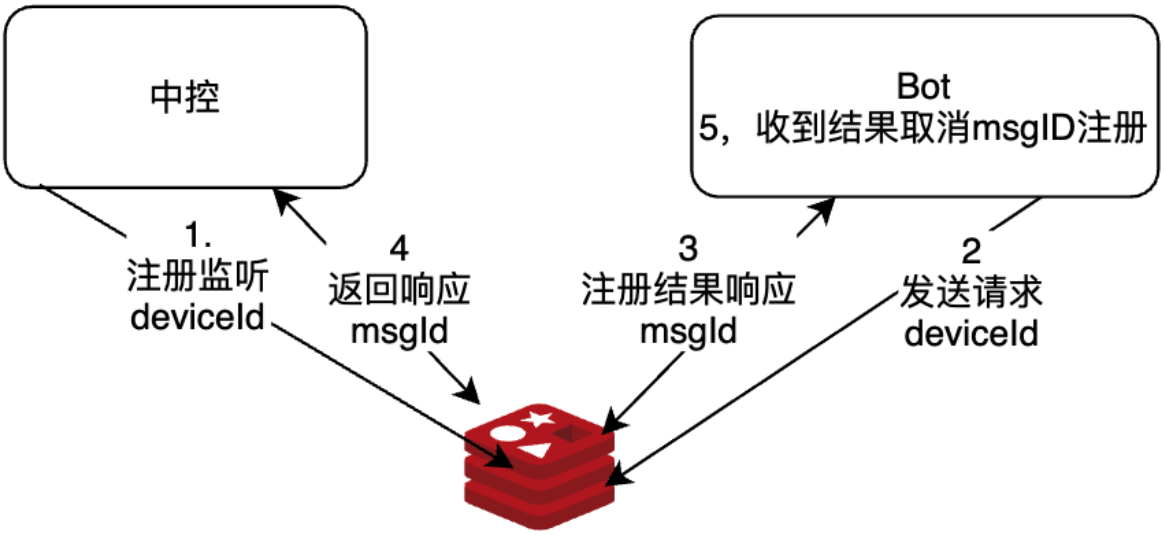
当前问题：

1. 目前广播机制会极大影响整体性能以及后续横向扩容
2. 一次请求访问慢，目前越来越多的依赖nativeApi

解决方案：

1. 通过统一共享数据，减少nativeApi调用
2. 一次nativeApi，会进行两次广播，nativeApiReq中控12个节点会收到，nativeApiResp业务bot16个节点会收到，横向扩容难，升级为动态发布订阅真正点对点消息协议





共享数据定义：

用户基础信息

uid			

`user_id`, `vin`, `device_id`, `is_online`, `last_login_time`, `last_logout_time`, `user_config` 倾听时长、体验改进计划、全场景开关、自定义唤醒词 `speech_version`, `dui_version_suffix`, `dui_device_name`, `car_type`, `car_platform`			

TODO：

- [ ] 多车型多微服务可测性、上线敏捷性
- [ ] 服务爆炸式增长，保障创新业务功能同时，如何保障生产质量，如何快速定位问题?
- [ ] 连接层的安全认证体系，正式review，需上会
- [ ] 用户请求响应冲突问题 杨如栋
- [ ] Java & python 支持粒度摸底 杨如栋
- [ ] 时效性、读写权限（单写多读）写走接口、读可直连，读写性能指标 读 1ms 写3ms
- [ ] key设计 ns+uid+topic
- [ ] SDK header自动传递机制可行性 张岩 杨如栋 拉通中台、python，类似java threadLocal
- [ ] 细化核心共享数据API
- [ ] 细化header、L2、L3数据结构

高自由度对话对接节奏 7月

接口协议: 流式、多人多路、取消回滚、结果反馈 2021-7-17

共享数据梳理高自由度对话使用场景以及数据，拉会讨论 2021-7-17

与高自由度联调：mock数据 2021-7-28

- 背景
- 系统设计目标
- 挑战：
- 三代平台概要设计
- 模块设计
    - 统一共享数据
        - 数据采集
        - 数据使用
    - 车机交互
        - ASR接入，待融合两者设计
        - TTS接入
    - 融合中控
        - 流式交互抢跑
        - 端云一致性保障
    - 灰度实验室
    - E2E性能优化项
        - 流式抢跑，如上
        - nativeApi优化
- TODO：

背景

伴随语音创新业务爆炸式发展，急需升级三代语音架构，用以集成自研ASR、超高自由度、多人全双工、免唤醒、认知计算等创新业务能力

打造语音行业最领先、最智能、最快速的对话系统

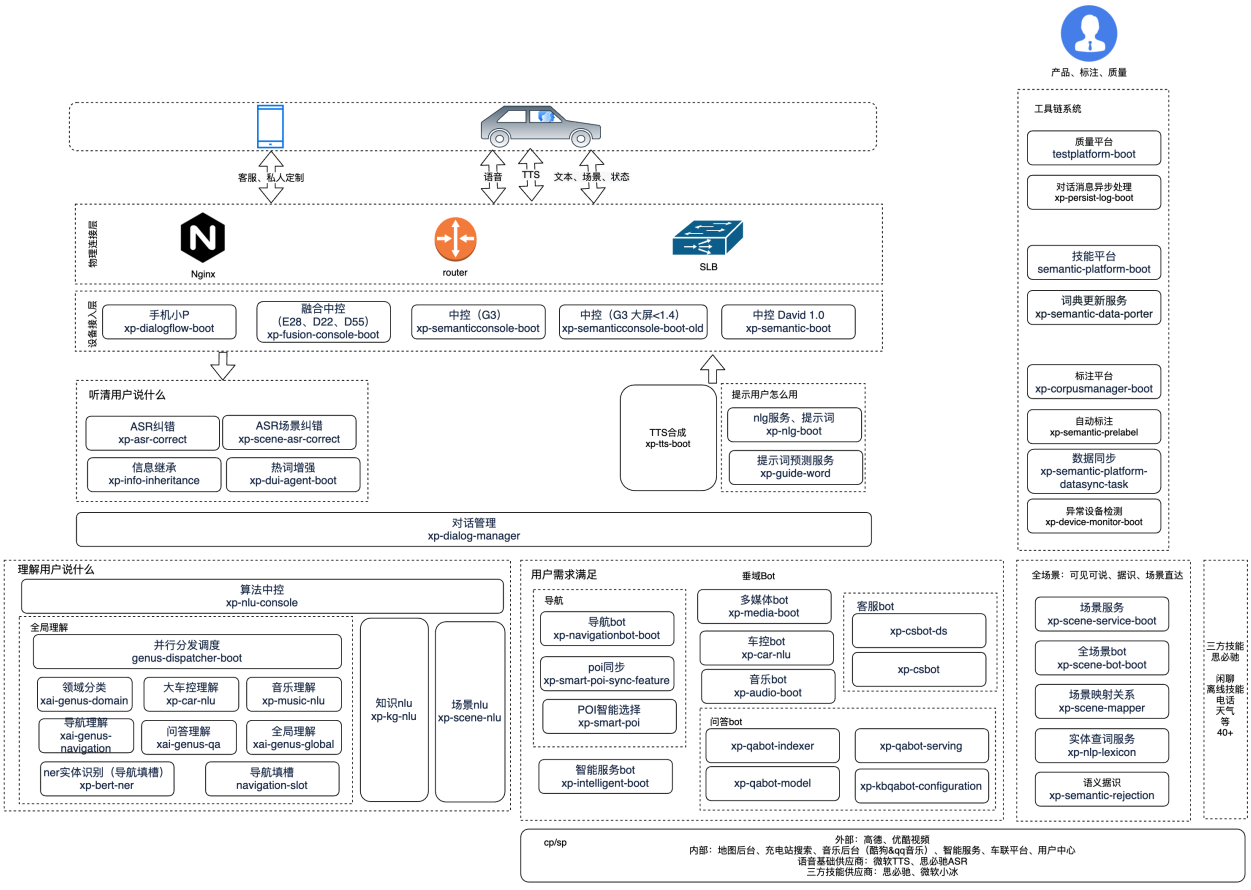
系统设计目标

- 高可用性：持续稳定生产系统，平滑过渡大创新落地，提供全流程灰度实验室、延迟管控、服务降级、限流熔断等基础能力
- 端云一致：保障在本地召回及弱网状态，云端状态与客户端状态保持最终一致，云端支持多阶段提交、回退、终止能力
- 实时感知：支持多业务数据信号实时采集诉求，强感知数据亚秒级返回，弱感知小时级别
- 更低延迟：性能要求E2E 500ms，最快小P，E38-三代性能专题-- E2E 500ms
- 轻量协议：解决未来150+微服务之间交互痛点，协议灵活可扩展
- 多车型适配：E38、E28a（22年中）、F30（规划中）车型隔离部署上线

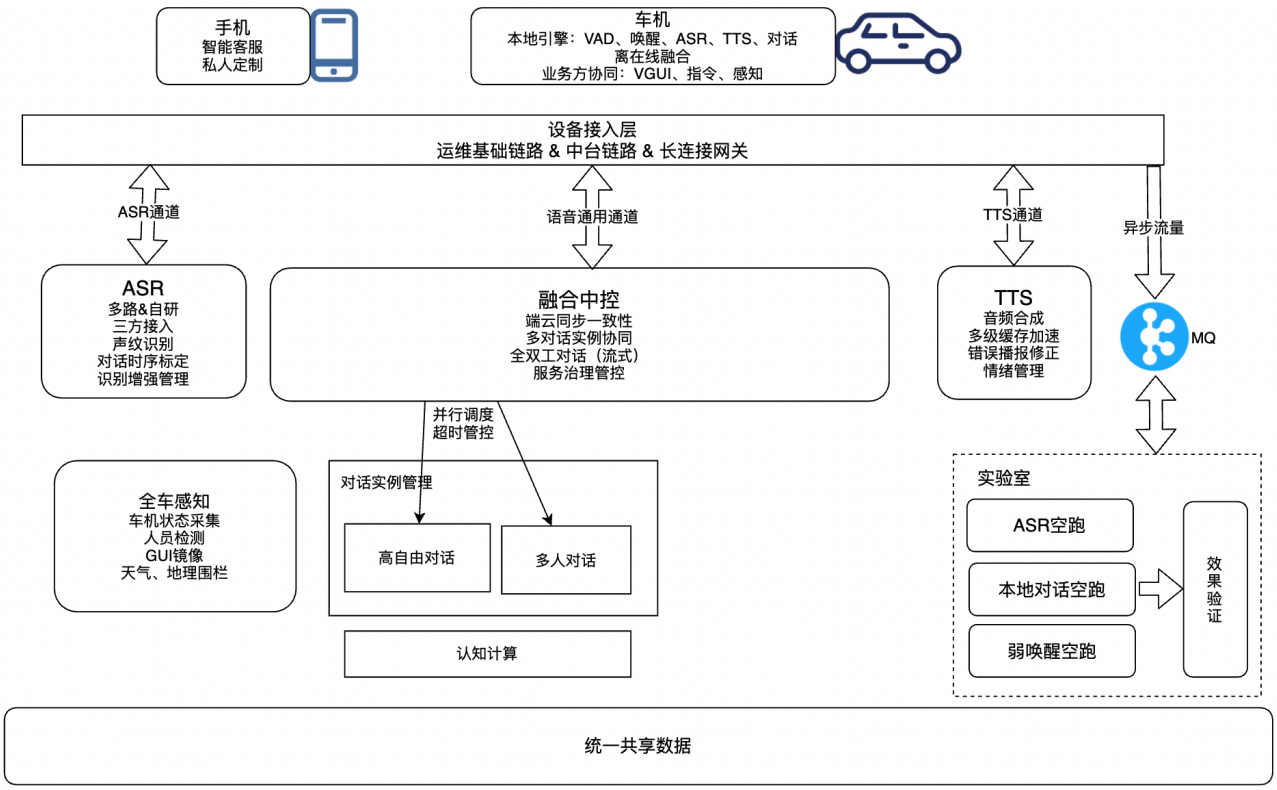
挑战：

- 多业务线交互复杂，调用关系错综复杂，如何简化交互？
- 多人与高自由度融合难度大，怎么保障存量业务前提下稳定交付大创新？
- 服务爆炸式增长，保障创新业务功能同时，如何保障生产质量，如何快速定位问题？待进一步设计

二代架构现有模块图，未来会2-3倍，甚至更多的扩张



三代平台概要设计

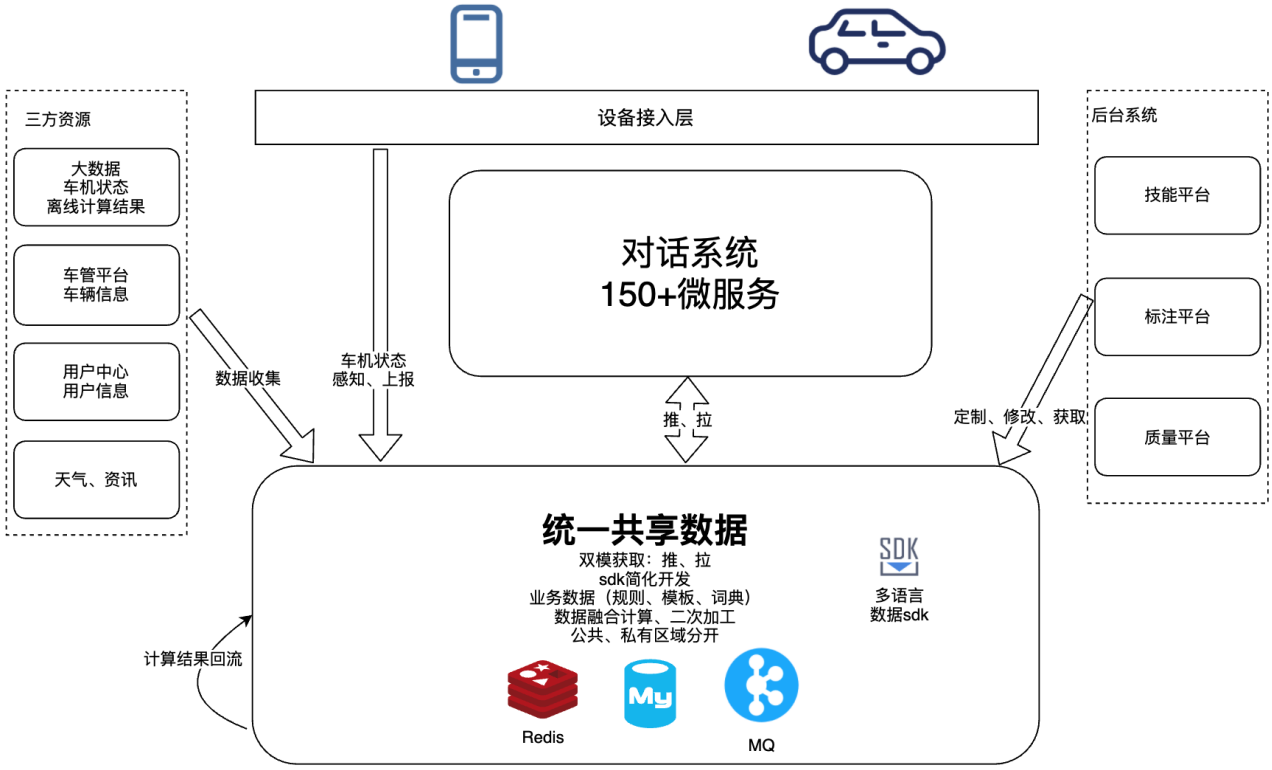


模块设计

统一共享数据

功能：提供底层数据共享，让所有业务拥有方便存储和快速获取数据能力，专注业务开发，降低业务依赖其他服务中间结果异步等待复杂度，充分信息共享

特性：推、拉模式结合，改变传统模式等待结果后显性强依赖调用，降低调用复杂度

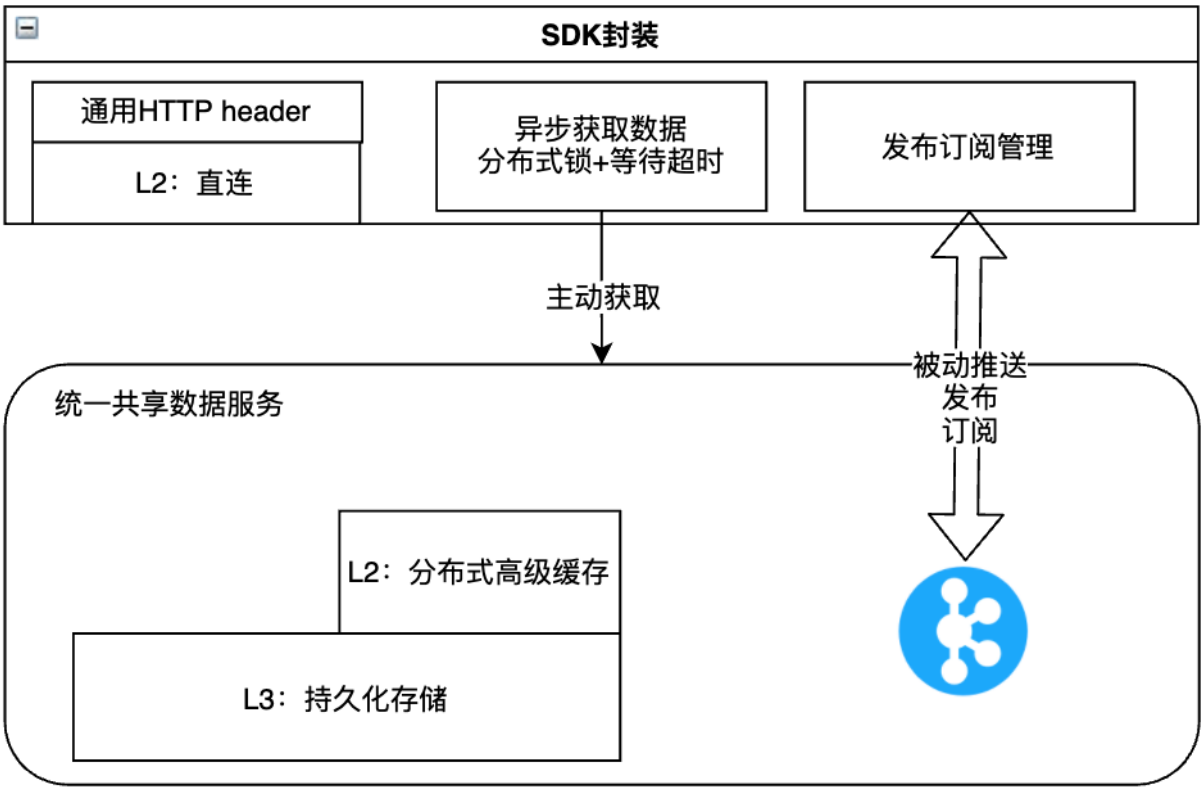


数据采集

数据资源	来源	实时性	缓存级别
实时车机状态 场景感知人数判断初步分析 当前播放歌曲 L2 当前目的地 L2 经纬度、电量 L2 说话语序 L2 播报语序 L2	语音app文本链路上报	亚秒级	
车辆配置 L1 用户基础信息 L1 订单信息 L3 用户收藏 L2 用户习惯数据 L3	大数据周期同步	准实时（小时）	
外部环境感知 L3 天气、路况 L3	通过API周期性感知	准实时（小时）	
规则数据			

数据使用

SDK能力细分

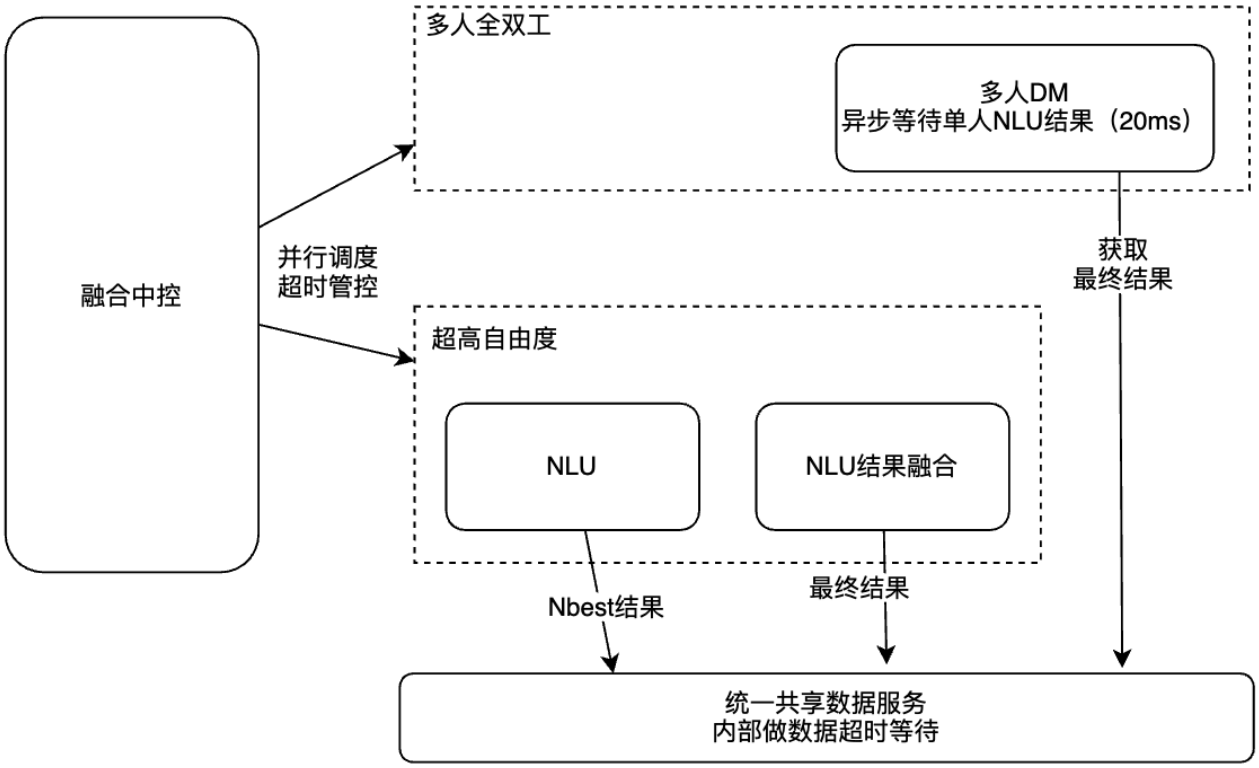


使用场景介绍

- 主动拉数据：

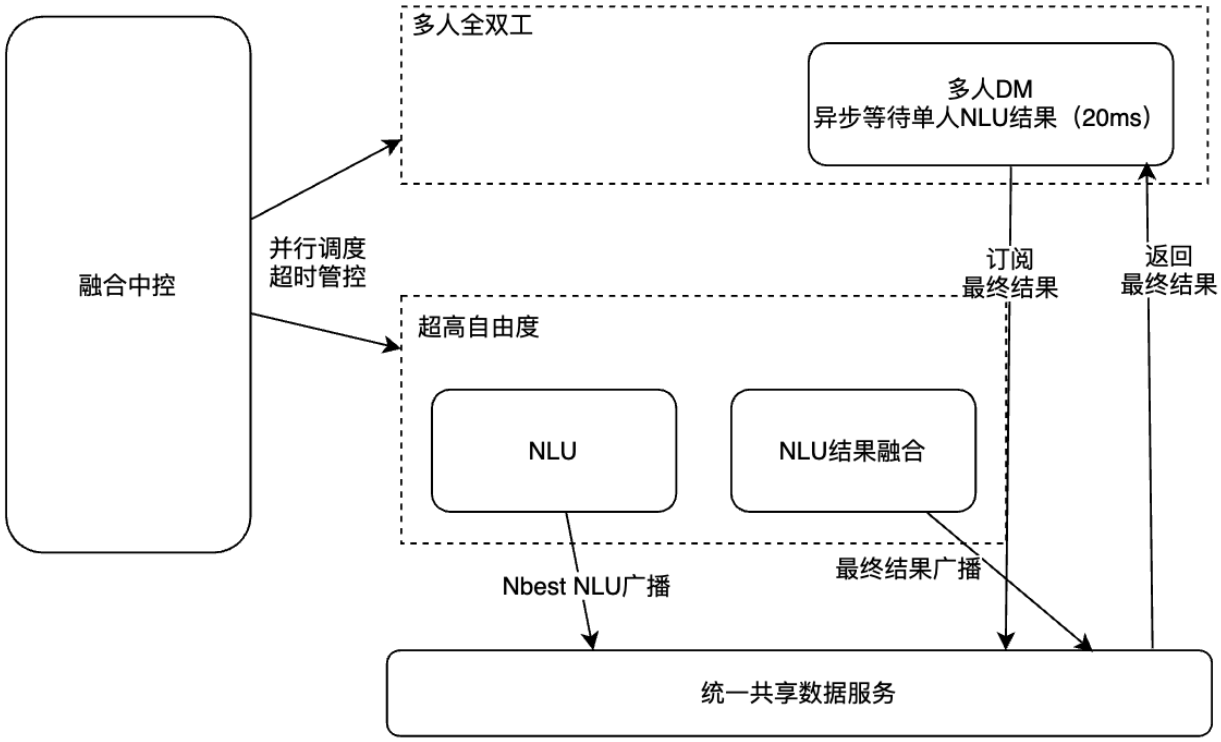
获取非实时数据，车机基本信息、用户基本信息、用户对话历史等

- 业务逻辑强依赖，同步获取结果



- 非强依赖异步获取结果：

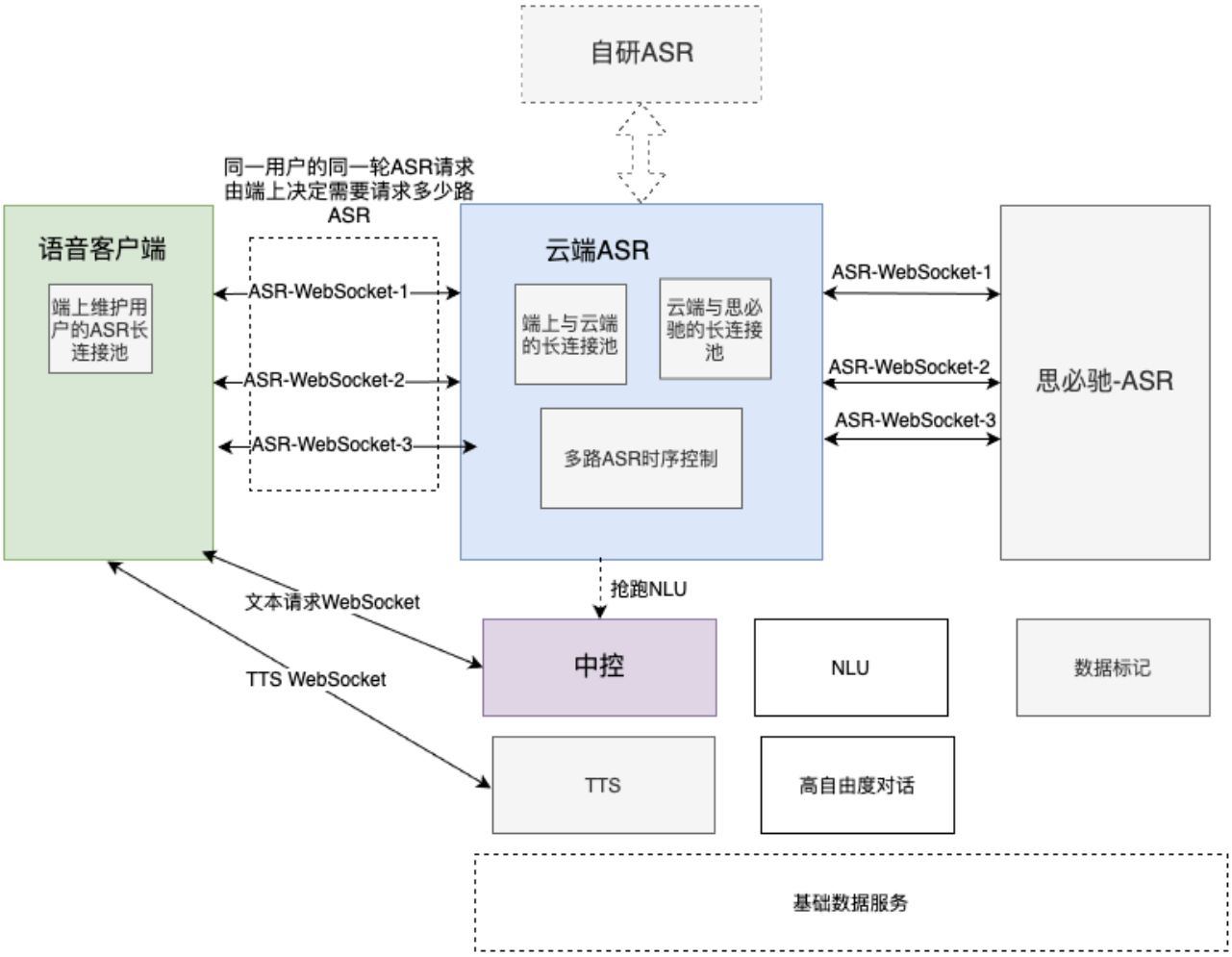
分布式锁等待获取其他模块中间结果，多人依赖单人NLU最终结果，则采用分布式等待锁方式，异步等待结果，如规定时间内无返回则超时



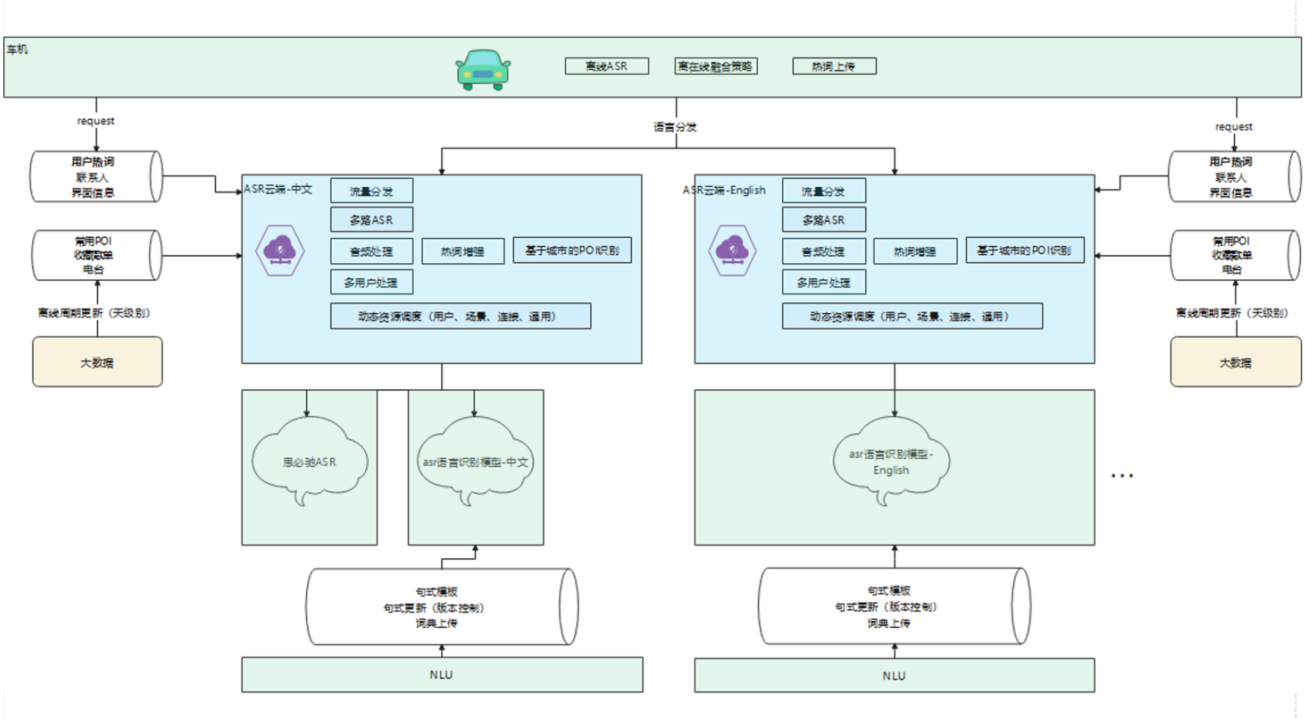
车机交互

ASR接入，待融合两者设计

多路ASR设计文档，优先E28落地收集数据



自研ASR的云端服务设计文档，将ASR算法能力工程化落地

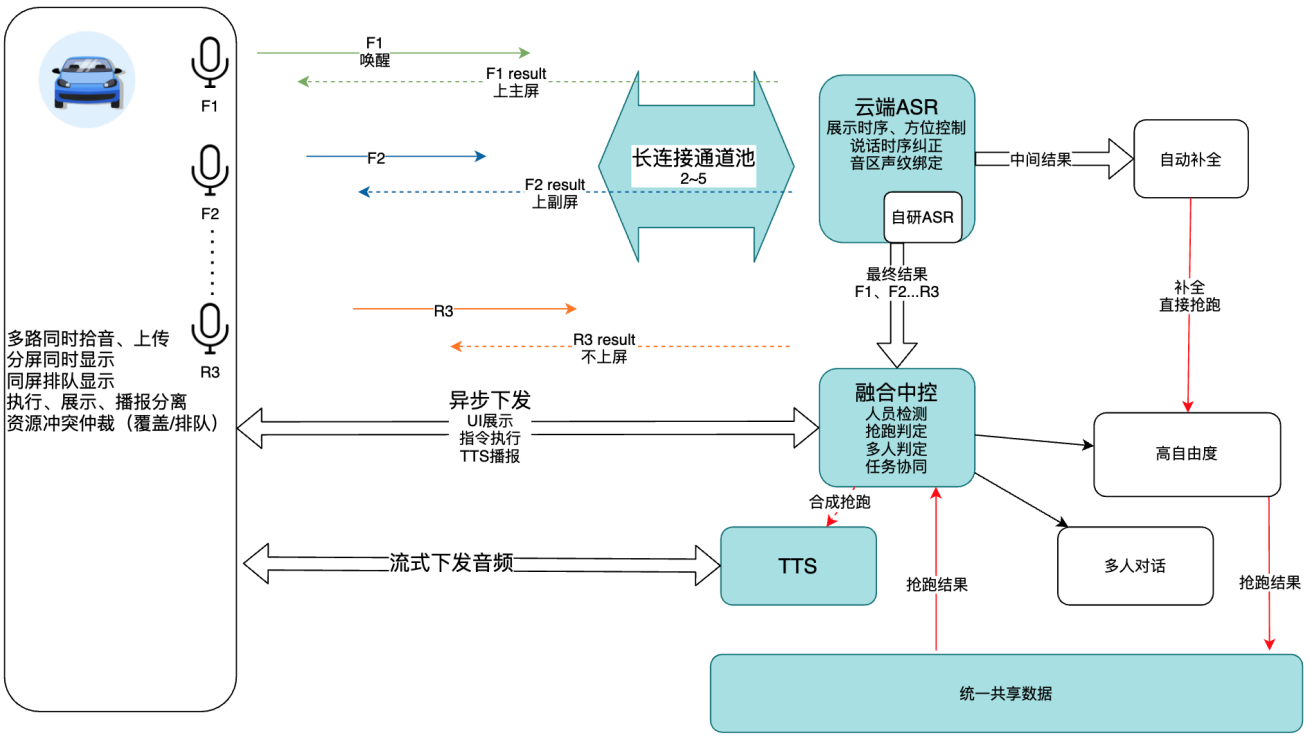


TTS接入

云端tts设计方案-微软，情绪管理设计待完成。

融合中控

- **流式交互抢跑**



补全直接抢跑炒近路，不需要经过asr服务，融合中控，最终由融合中控进行抢跑判定

抢跑判定方式：

中间结果	补全结果	ASR最终结果	匹配结果
明天天	明天天气怎么样 weather	明天天气怎么样 weather（tomorrow）	字符串匹配 补全命中
		明天天气好不好 weather（tomorrow） 需要确认占比后是否可行？？ 李晨延 NLU相同时，结果是否相同？？	NLU全匹配 补全命中
		明天天气挺好的 chat	意图匹配失败 补全失败

- **端云一致性保障**

离线处理

弱网处理

云端处理状态：

流式补全抢跑实例、ASR结果实例

客户端真正执行结果上报：各个业务做融合，

时序处理？

灰度实验室

通过异步流量分发机制，解耦线上生产业务，可准实时的进行新业务灰度验证功能

多渠道效果验证

- 监控大盘业务埋点验证
- 结果发送异步消息队列后标注平台效果评估
- 大数据日志埋点比对

需要增强能力：

增加HTTP流量调度，上下文查询能力

E2E性能优化项

流式抢跑，如上

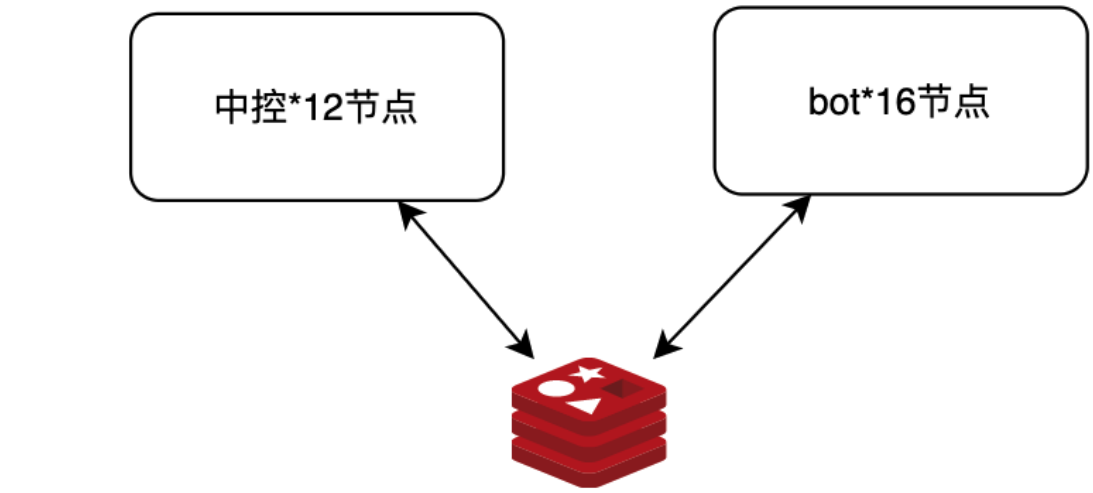
nativeApi优化

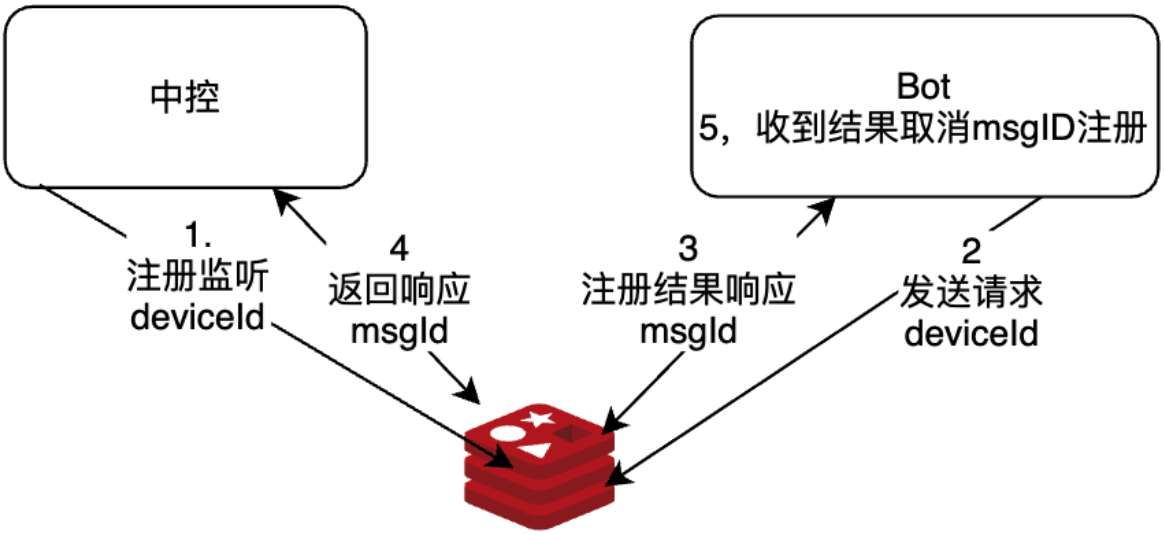
当前问题：

1. 目前广播机制会极大影响整体性能以及后续横向扩容
2. 一次请求访问慢，目前越来越多的依赖nativeApi

解决方案：

1. 通过统一共享数据，减少nativeApi调用
2. 一次nativeApi，会进行两次广播，nativeApiReq中控12个节点会收到，nativeApiResp业务bot16个节点会收到，横向扩容难，升级为动态发布订阅真正点对点消息协议





共享数据定义：

用户基础信息

uid			

`user_id`, `vin`, `device_id`, `is_online`, `last_login_time`, `last_logout_time`, `user_config` 倾听时长、体验改进计划、全场景开关、自定义唤醒词 `speech_version`, `dui_version_suffix`, `dui_device_name`, `car_type`, `car_platform`			

TODO：

- [ ] 多车型多微服务可测性、上线敏捷性
- [ ] 服务爆炸式增长，保障创新业务功能同时，如何保障生产质量，如何快速定位问题?
- [ ] 连接层的安全认证体系，正式review，需上会
- [ ] 用户请求响应冲突问题 杨如栋
- [ ] Java & python 支持粒度摸底 杨如栋
- [ ] 时效性、读写权限（单写多读）写走接口、读可直连，读写性能指标 读 1ms 写3ms
- [ ] key设计 ns+uid+topic
- [ ] SDK header自动传递机制可行性 张岩 杨如栋 拉通中台、python，类似java threadLocal
- [ ] 细化核心共享数据API
- [ ] 细化header、L2、L3数据结构

高自由度对话对接节奏 7月

接口协议: 流式、多人多路、取消回滚、结果反馈 2021-7-17

共享数据梳理高自由度对话使用场景以及数据，拉会讨论 2021-7-17

与高自由度联调：mock数据 2021-7-28