



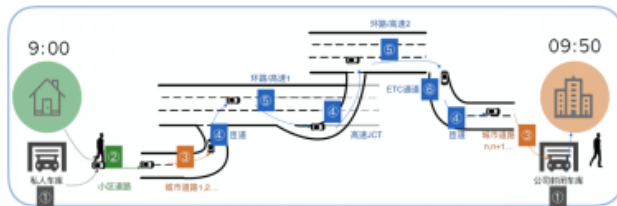
ANP3系统故障排查&&春季大扫除发布

ANP SYS | 辛建康

ANP3行业现状 - 快速过一下ANP3产品

【通勤】私家车主要行车场景之一，主要有以下特点：

1) 上下班拥堵路况；2) 路线和道路环境熟悉；3) 除基本城市道路外，北上广深等大城市通常会覆盖大段高速和环路道路



功能核心优势

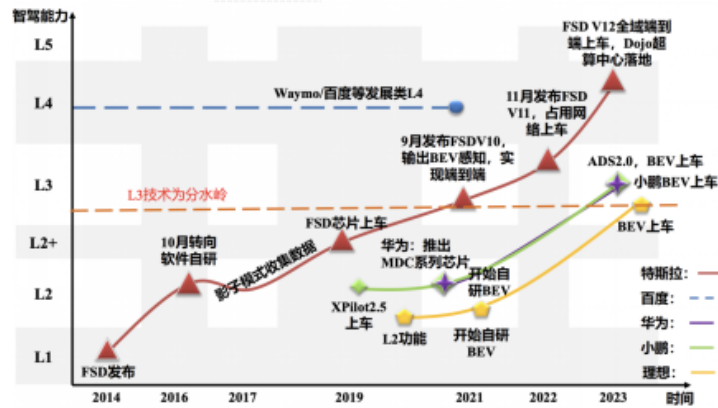
- 将传统的辅助驾驶升级为具备A到B能力的高级辅助驾驶；
- 通过人机界面，降低用户的认知难度和使用门槛；



产品形态

- 车机地图发起导航
- 车辆按照导航路径自动驾驶（人机共驾）
- 在能力不足的地方提醒接管

ANP3行业现状 – 大家都在”卷”，超级”卷”



- 国内某证券公司行业调研: 2022 年乘用车 NOA 标配前装搭载交付量为 21.2 万 辆, 今年 1-6 月交付 20.9 万辆, 已接近去年全年水平, 1-9 月交付量已达 37.7 万辆, 同比增长 **151.2%**, 渗透率接近 2.5%。
- ANP3三域融通: 快速集结, “开箱即用”, 22年10月正式开始融通, 500W+行代码, 数百个模块, 快速进入第一梯队; 我们很”**难**”, 也很”**强**”

ANP3系统架构 – 大型迭代不低于10轮, 小型迭代无数, 如何应对?

23年H1		23年Q3	23年Q4	24年
BEV上线	DriveOS 6050	纯视觉感知	OCC感知	轻图
ANP3三域融通	ANP3感知复用	AVP+ANP感知融通	Cyber迁移	感知能力提升
预处理异构计算解耦	...	SOP交付	预测、规划升级	...

► 系统的典型问题

- **硬件向:** 授时跳变、相机掉线、BGM联网、底盘丢心跳、底盘控制延时、网卡UDP流量上限等...
- **OS和CUDA:** CPU10卡死、CUDA Context竞争、系统卡顿等
- **中间件:** 跨中间件通讯、多运行时竞争、自研中间件稳定性等
- **业务层:** 随机越界、算法退出、临界区竞争卡顿、死锁、死循环、不合理计算量等
- **基础库/系统运维:** 故障检测、深度检测、容灾、故障恢复等

应对

- “工具”远比”人”靠谱
- 懂E2E、懂大数据、会巧用

ANP3系统故障Tips - 相信AI的力量

[illegible]

题外话：AI在知识垂向和广度上做的很好，但在知识关联和逻辑上稍差，这也是现在唯一人类还有点用处的地方…

[illegible]

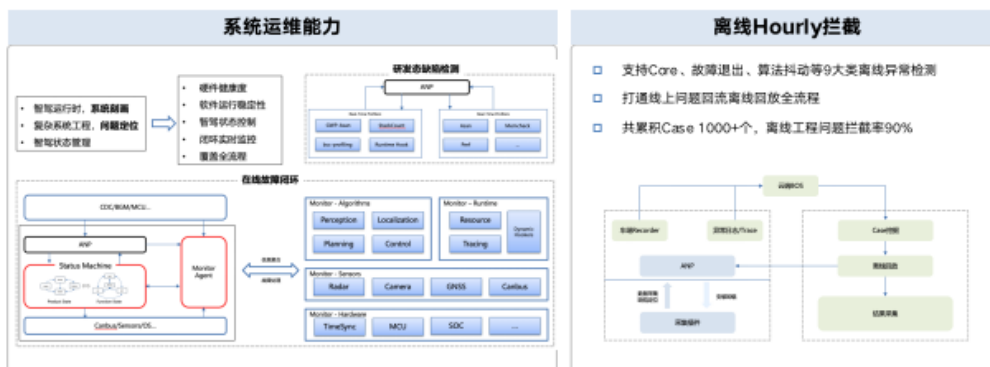
主 題	環境教育	主 題
活動目標	透過影片欣賞及討論，認識水資源的有限性，並了解水污染對環境造成的影響。	透過影片欣賞及討論，認識水資源的有限性，並了解水污染對環境造成的影響。
活動內容	1. 影片欣賞：播放「水資源」影片。 2. 討論：影片中的水資源問題，你認為最嚴重的是什麼？ 3. 小組討論：根據影片內容，設計一個水資源保護計劃。	1. 影片欣賞：播放「水資源」影片。 2. 討論：影片中的水資源問題，你認為最嚴重的是什麼？ 3. 小組討論：根據影片內容，設計一個水資源保護計劃。
活動時間	45分鐘	45分鐘
活動地點	學校禮堂	學校禮堂
活動負責人	陳國治	陳國治

[illegible]

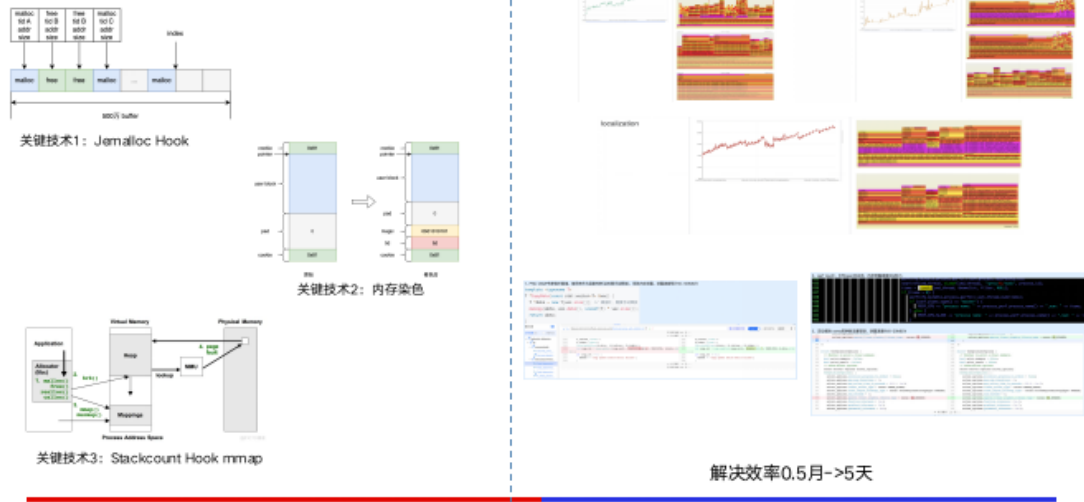
1	姓名	王德江	男
2	身份证号	360101198001010011	3601-01-80
3	手机号	13907010000	1390-0701-0000
4	电子邮箱	13907010000@163.com	1390-0701-0000
5	地址	江西省南昌市西湖区	3601-01-00
6	职业	教师	3601-01-00
7	备注	无	3601-01-00

- ChatGPT

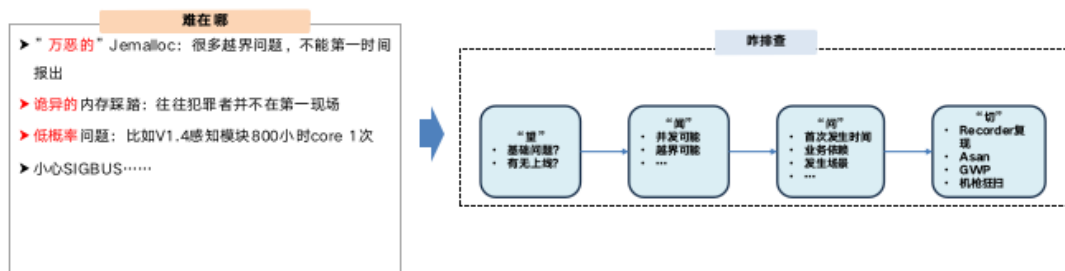
ANP3系统故障排查 - 我们有什么



内存泄露排查 - 线程监控、大内存函数级实时监控



工程可用性 - Core、越界、踩踏等



► Tips

- ✓ 大部分问题, 基本在望的阶段, 就被拦住了, 比如…空指针、abort
- ✓ 少量的问题, 需要再进一步看一下, 现在依赖人工, 但未来希望能AI替代
- ✓ 需要相信大数据的力量, 从Sugar、实时回传推算, 首次问题发生版本、场景、模块可能运行到哪等
- ✓ 95%以上问题, 离线Asan都能扫出来; 5%问题, 需要依赖GWP-Asan(比如800小时复现1次的诡异地问题)

Core、越界、踩踏 - Why GWP

```
开始时间: 2024-02-04 14:42:43
问题号: slave
版本号: 8.3.38.63
产出分支: master
问题场景: 运行非预期出现 ["localization"]
测试任务: 主链开发测试
日志下链接: http://ank-snpchain.bj.bcebos.com/data_recycle_daemon/20240204/ank_run_error_17
0228983/slowdown_JDLU677BP2NSNP002I37/ank_run_error_170228983/JDLU677BP2NSNP002I37_2
024020244442_slave.rar?authorization=bce-auth-v%2F0cfca%254d95c4e2b835295f75955262
9f9924-02-0470t%3A4B%3AS4Z2NF-%2FHost%2F833712T2e4B9e562942e6bf023c12c56e02262
98995ecbb101444fabac8756cf
trace info: *** SIGSEGV (Si0x0F5027e1e8) received by PID 213481 (TID 214575) from PID -18764344
55; stack trace: ***
@   0xfffff6ad4af0    _kamel_rt_sigreturn=0
@   0xfffff52723f0    adu:localization:lite:SimSolverMultiLane::calc_lane_offset()+0x10
@   0xfffff5280f98    adu:localization:lite:SimSolverMultiLane::collect_mixend_mixed_lines()+0xa8b8
@   0xfffff5282348    adu:localization:lite:SimSolverMultiLane::collect_mixed_matching_lines()+0xb
48
@   0xfffff528708c    adu:localization:lite:SimSolverMultiLane::calc_on_all_lines()+0xc2cd
@   0xfffff528ba74    adu:localization:lite:SimSolverMultiLane::compute_opt_pose()+0x2bc4
@   0xfffff57510fc    adu:localization:fusion:MapmatchFuseUrban::fuse_ego_lane()
+--[ANP-9067642] https://fusion.baidu.com/#JDLU677BP2NSNP002I37[B.3.38.63][2024-02-04 14:42:43].com
```

原线上core: 只知道哪个函数挂了, 但不确定是那段内存飞了

- Use-after-free
- Buffer-underflow
- Buffer-overflow
- Double-free
- free-invalid-address

```

    #get the user after the login
    def login(self):
        #get the user name and password from the user
        username = input("Enter your username: ")
        password = input("Enter your password: ")

        #check if the user is already in the database
        if self.check_username(username):
            #check if the password is correct
            if self.check_password(username, password):
                #login successful
                print("Login successful!")
                return True
            else:
                #password incorrect
                print("Password incorrect!")
                return False
        else:
            #user not found
            print("User not found!")
            return False

    #check if the user is already in the database
    def check_username(self, username):
        #connect to the database
        conn = self.connect_to_db()
        cursor = conn.cursor()

        #check if the user is already in the database
        cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
        result = cursor.fetchone()

        #if the user is found, return True
        if result:
            return True
        else:
            return False

    #check if the password is correct
    def check_password(self, username, password):
        #connect to the database
        conn = self.connect_to_db()
        cursor = conn.cursor()

        #check if the password is correct
        cursor.execute("SELECT * FROM users WHERE username = %s AND password = %s", (username, password,))
        result = cursor.fetchone()

        #if the password is correct, return True
        if result:
            return True
        else:
            return False

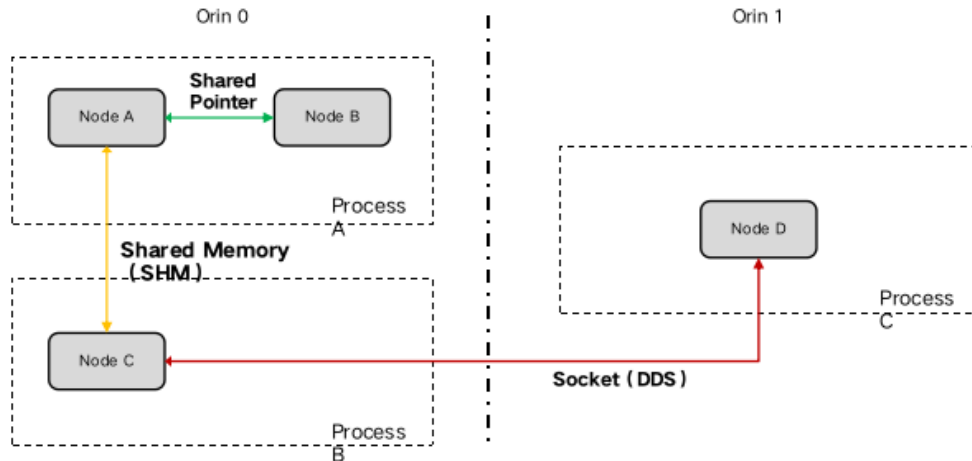
    #connect to the database
    def connect_to_db(self):
        #connect to the database
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            database="users"
        )
        return conn

```

GWP-Asan: 明确给出写飞内存; 正常路测就可以打开

参考：<https://mp.weixin.qq.com/s/xipHtjHPVlyFQ6W-1HfUQQ>

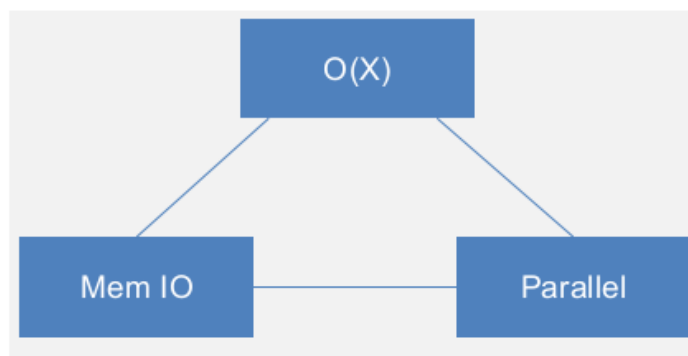
工程可用性 - 卡死、消息堵塞等



卡死排查思路

- ❖ 排core: /opt/log/anp/stdout
- ❖ 排自身: 工程, 有无锁、网络等待、超长sleep、IO等待等; 业务, 是否满足触发条件、上下游依赖等
- ❖ 排系统: CPU、内存、网络、IO、系统异常进程五要素
- ❖ 排对端: 是否收到对端消息; 对端是否发送了消息;
- ❖ 排中间件: 线程池打满、拓扑掉线、消息阻塞、抖动丢帧等
- ❖ 排外部依赖: BGM(授时、CDC转发); MCU(底盘、radar、gnss)等
- ❖ 系统: 未来会为全域增加ANR检测能力

性能问题排查 - 启动延时、端到端延时



排查思路: 资源观测->找工具->Profile->分析->优化->...

启动延时、端到端延时 - 常用工具

- strace: **启动延时**排查利器, 可以深入到每一个SO加载耗时
- sys ebf profiling: **CPU性能分析**, 函数级火焰图, 实车闭环采集
- jemalloc: **内存Profile**, 支持实车闭环采集
- 系统线程内存、CPU分析
- 进程级**流量**采集
- ...

其它

- 活用JIDU VP群公告文档, 大部分路测问题(联网、授时、地图问题)等, 都可以找到答案
 - 多报明确问题, 少报一些现象(比如DV黑屏了, 谁来看下), 提高问题排查效率
 - 相信工具、相信大数据
 - ...
-

推荐知识库

- ❖ [ANP工程Best Practice](#)
 - ❖ [ANP典型Core分析](#)
 - ❖ [定位问题分析宝典](#)
 - ❖ [ANP出车手册\(强烈建议多看JIDU VP群\)](#)
 - ❖ [系统性能指标观测](#)
 - ❖ [线程级内存分析](#)
 - ❖ [线程级CPU分析](#)
 - ❖ ...
-

24年春季大扫除 - 预告

- ❖ Rule1：重点清理可用性类问题
- ❖ Rule2：对潜在问题坚决清理
- ❖ Rule3：部分引入AI扫描能力(对接中)

• Q1 清理规则(初步)

- | | |
|---------------------------------------|--|
| ○ for 循环中不允许出现 <code>size - 1</code> | ○ non-void 函数先写返回值 |
| ○ 指针都要检查有效性 | ○ 不要在头文件里写 <code>using namespace xxx</code> 或 <code>using xxx</code> (别名除外); |
| ○ 不要在一个函数中返回地址 | ○ 当一个类必需持有自身类的时候, 禁止使用 <code>std::shared_ptr</code> ; |
| ○ 不要相信 <code>vector</code> 中元素的指针 | ○ 提前申请 <code>vector</code> 的空间 |
| ○ 递归函数一定要加层深限制 | ○ 对于 <code>vector</code> , 能用 <code>emplace_back</code> , 就用 <code>emplace_back</code> ; |
| ○ if 语句中, 右值放在 <code>==</code> 的左边 | ○ 能用乘法, 就不要用除法。 |
| ○ if 语句中, 即使只有一行, 也要加 <code>{}</code> | ○ 类内提前开辟内存 |
| ○ 浮点数比较, 要考虑精度 | ○ 禁用 <code>Malloc</code> 、 <code>New</code> |
| ○ 即使再不合理的代码, 也不要 <code>CHECK</code> | ○ 禁用 <code>CHECK</code> |
| ○ 默认使用 <code>std::abs</code> | ○ 日志、write消息削峰 |

感谢~

