支线产品,6 代监控主体架构

版本:1.0

撰写时间:2024-3

更新日志

目录

说 6 代监控主体架构前先惯例性的铺垫思路

这里的设计思路并不是什么点子,想法,也不是灵光一现.而是从我切身生活与经历中总结而出的规律:每个时代的浪潮都是伴随着技术革命或则是政策趋势,今天是 AI 大模型时代.作为一个技术员跟上时代的方式就是学习当前的热点技术,掌握它,并且做出产品.剩下的,交给时间.如果只是学习当前热点技术,而不去尝试动手做出产品,这会是一种对世界和自我的轻视,尽管赶潮流有时遇到合适的人,会带来或多或少的财富,但惰性会长时间伴随着你,时间长了,成为混世魔王.

身处于浪潮之中,一定要去追浪潮,并且一定要做出产品.这是开发 6 代监控最底层逻辑.至于成功失败,财富的多少,这些东西在生命和时间面前不能勉强,需要客观条件达到,关键是在每个浪潮中做产品的过程:灵魂会净化,心灵会成长.经历几个浪潮以后,就一定会走上孜孜不倦做产品的道路.这里面的酸甜苦辣,如人饮水.

输赢

世界进步的规则总是先有技术,然后资金进来参与,当技术与资金发生相互作用时,会出现方向性的势力割据,这种势力割据可以直接碾压所有弱小资金以及成千上万的技术团队和技术型公司.唯一可以与之所抗衡的只会是另一帮发生了相互作用的技术+资金.技术完成创造,资金推动现实,这是天作之合,世界就是这样在运作.

6 代监控的赢面是项目交付,在完成交付前,只会输不会赢,在完成交付后,需要保持更新升级做好品质工作,用时间来挤进圈子,成为系统集成商的候选供应方.目前已经完成交付,后续的升级,需要实际项目推进.再做 2 单应该可以形成技术碾压,实现以旧换新.

理解监控市场

在技术圈里面,懂得监控市场的人是极少发言的,不懂的都在看热闹.监控是一个供需市场,在供应环节,设备供应主要是海康,大华,宇视,英飞拓,在需求环节,主要是系统集成商和设备供应方共同解决客户需求,也可以理解成,系统集成商与设备供应方既是竞争对手也相互依赖.而要做好解决客户需求这件事情,集成商和设备供应商是站在平等市场位置上的,最大的差别是手上是否拥有平台,这种平台就是 6 代监控.拥有平台的一方,在技术经验和业务深度的双重加持下,会成为市场中的赢家.

Pas 圈很多人从来没做过监控,都是凭直觉看到海康是上市公司,资本雄厚,技术研发队伍强大,其实在市场环节,海康与系统集成商是站在同一个位置在竞争,所有人都跟随 AI 的潮流,所有人都在处心积虑做自己的框架.6 代监控切进去根本不存在被巨头统治一说,就是存粹的是不是拥有自主平台,你拥有自主平台,做几个项目,你就展翅高飞.

开发 6 代监控面对的难度前所未有

6 代监控定位之初就是做成平台,做这件事,是以纯净的灵魂,和一颗赤子之心去孜孜不绝的追求技术的状态在进行.很多高水平的技术很看重钱,其实钱这种东西是外在的事物,像股票,炒币,投资,这些东西是无法被控制的.真正的能量都来自灵魂,我经常与人说净化灵魂,其实就是这个意思.做技术类的工作一旦脱离了灵魂,行为就会受外在事物影响,人的心态的就会发生变化,这是做不长的,这会破坏秩序,生出是非.

6 代监控的核心工作是模子,因为定位之初是做成平台,只有在模子出来以后,围绕模子日积月累,不断的添砖加瓦,保持更新,长时间运作下去,最后才可以完成平台.模子并不是凭空想象来做,也不是模拟,就是直接在堆满服务器的数据中心跑.根据回忆,这个模子大约在立项后的 3 个月左右写好的,然后 gpu 服务器这些设备的被陆续安装到位,是驻场安装,在现场住了一个月酒店,很多,设备匹配,交换机问题,拓扑问题,各种蛋疼.因为 idc 机房都是拿钱堆出来的,现场是甲方亲自出动解决,我是远程配合,一条战线,半夜 3 点都可以从床上爬起来这种配合.最耽误时间的事情是设备坏掉,设备不匹配,只能通过网购解决,这种事情会等好几天,如果遇到菜鸟和奸商网店会万马奔腾.

当服务器到位以后,甲方终于解放,终于可以回家陪老婆,我的事情也开始了:就是跑模子,初期各种小问题多如牛毛,许多问题一直拖了大半年才得到解决,诸如谜之丢包,谜之短瘫,谜之断线.在模子方式下解决这些小问题都是业余在搞,只要不是长时间瘫痪,宕机这种级别,都是慢慢解决,想到办法试一下,解决不了,就想其它办法.其实这些小问题都不算难点.

真正的难点是**磁盘阵列,隐藏 bug,推进落地给项目开新功能,生产工艺,修正建模体系,最后是持续 2 年的长跑**.当这些事情全部堆在一起是前所未有的难度.

磁盘阵列

阵列系统最初我根本不想碰它,我做 6 代监控项目之初就是打算把目标找出来给结果然后完成交付,但随着建模数据源的需求越来越强烈,已经到了必须要有丰富数据源的地步了.而 NVR 子系统全是离散的,大约 20-50 路 2k 使用一组 NVR,在 idc 机房 NVR 系统有一大堆设备,采集数据源需要登入各个设备,我尝试过使用 openapi 来采集 NVR 数据,偏偏这些设备又是不同厂商的组合,openapi 这条路,最后直接被否决.因为监控是基建工作,基建会分阶段进行,各个阶段都会有不同的基建工程标准.

这时候,我考虑的办法就是让甲方辛苦一点,每次人肉的登入 NVR,一路一路的采集.虽然累,效率低下,但好歹能采集出数据.因为我要下手做磁盘阵列去替代 NVR,这种工作等同于把传统监控好几年的技术积累全部躺完,其代价远远高于甲方的工作负担,看在钱的面子上甲方会忍受这种辛苦,这完全没问题.事实上,甲方真的专门花了好几天时间去人肉采集.

下手做磁盘阵列的真正原因是人肉采集策略给出后几个月,我们突然意识到:做 AI 项目不光是结果,AI 项目它给人带来的直观感受也很重要,甚至说,这种感受并不亚于 AI 的识别结果,因为很多识别结果根本没用,例如像人数统计,人流量统计,折腾半天,就为得到几个数字,这些数字真有这么重要吗?灵魂提问!

AI 视觉和磁盘阵列的关系

这里插入一段技术讲解:AI 是一种数据,而 NVR 里面装的是视频,就是 MP4,如果要在 mp4 播放时,给人画上框框,提示这个人是干什么,给道路画上记号,给汽车画上方向,这时候,就需要从 mp4 成千上万的帧里面找到可以 AI 匹配的那一帧,这种匹配是时间戳匹配,需要精确到毫秒.NVR 里面视频时间是监控头时间,或则是中心化时间,中心化时间是指小群集统一时间,在 NVR 系统,可以统一化配置一批监控头时间.如果要在 NVR 基础上计算出视频与 AI 毫秒级时间戳,难度很高,除非 NVR 厂商自己开发.如果要解决 AI 视觉,必须有替代 NVR 的系统,这就是磁盘阵列.

回到主题

替代 NVR 的磁盘阵列本身并不简单,有一定难度,如果急功近利,或则轻视技术,这会是一件很难完成的工作,要完成这件工作必须有非常充足准备,一个是整体战斗策略层面,另一个则是需要有指标上的科学依据,这是一个理论在前实践在后的前置性预测.因为磁盘阵列会牵扯到:编码重建,传输优化,存储策略,缓存管理,硬件 IO 优化,这是整个传统监控领域最核心的一个技术环节.

编码重建

视频的编码重建环节使用了 2 个机柜的 gpu 服务器做这件事,同时也对整个 AI 端进行大规模更新,过去的 AI 端只会干视频解码+识别+输出识别结果,现在需要干视频解码->视频编码重建->识别->输出识别结果.看似只加了编码重建环节,但其实这几个步骤加在一起可以把 gpu 服务器的北桥彻底榨干:首先解码都是 1080p,2k 不等,单台服务器的负载大约在 50-80 路,从服务器接收到视频数据,然后把视频数据解码成光栅,这一步几乎已经吃掉接近 20g 的北桥带宽,而在 AI 的识别环节,这 20g 的光栅数据不是一次性传递给 gpu 做识别计算,这是反复的传递:这种传递通常是先传递一次整图光栅,单张光栅大约在 8-14M,单路监控头每秒 25 帧,每台服务器会负载 50-80 路的监控,AI 在识别完成后,大都会是一个框框或则是其它的数据,这些框框和数据会被流程抠图出来做第二次识别处理,这是非线性流

程,北桥带宽在这一流程中几乎被耗尽,cpu 在数据传递上的计算消耗其实很大:当 cpu 中 90%的核心算力被用掉以后,剩下 10%的算力会带不动高流量,表现出的症状是内存数据往 gpu 做 copy 传递时发生卡顿.

当加入了编码重建环节后,只要在 AI 端启动流程,80 路监控因为传递带宽和 cpu 算力不够用,出现内存无限暴增现象:gpu 服务器与云服务器不同,gpu 服务器工作与物理硬件,暴增型的内存一旦用完物理内存,只有通过 ipmi/bmc 来重启服务器,系统日志甚至写入"系统资源耗尽"这类崩溃事件都会时有时无,这对 idc 来说会是一场致命的灾难.

cpu 传递带宽被耗尽的解决办法

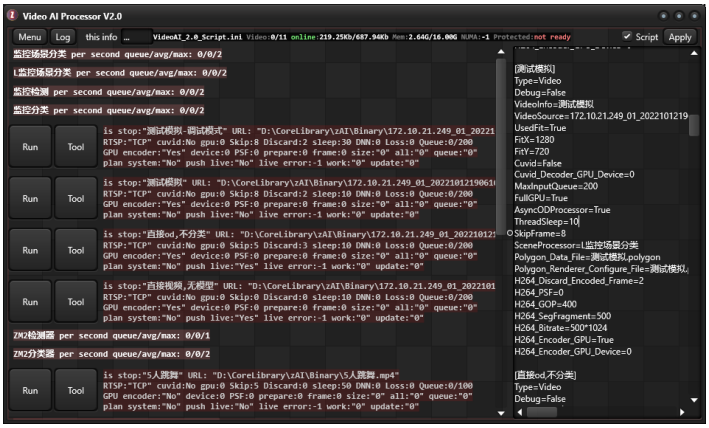
第一步是手动操作,一路一路的单独启动摄像头,每启动一路都要间隔 10 来秒,用于观察 cpu,北桥,gpu,内存,这些硬件设备的消耗状态.得到结果:在编码重建并入以前,单台 gpu 服务器极限可以负载 50-80 路,编码重建加入以后,单台 gpu 服务器的负载极限只能到 20 路左右,并且确定问题,打消了 gpu 不够用,监控头的帧率,分辨率配置这些无故猜测,问题就是出在 cpu 数据传递,前面的数据出现卡顿,导致了后面的数据一直等,也可以理解成数据输入量超出流水线的处理能力,流程进入数据积累状态,最终发生崩溃.

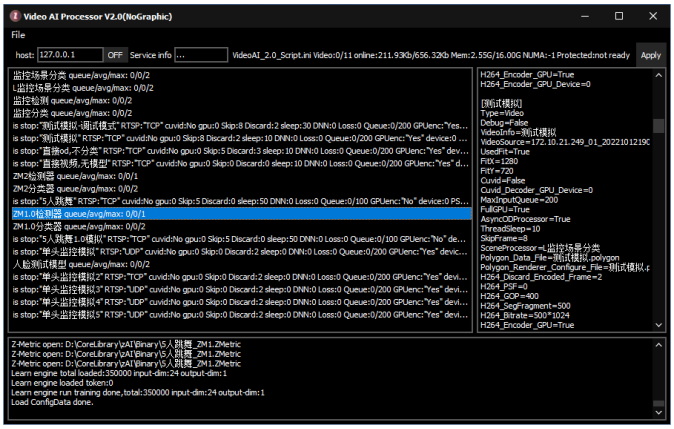
第二步是降低分辨率,把原本 1080p,2k 这类高分,一律砍成 720p,并且再把帧率从 25fps 砍成 10fps,这样操作下来,原本 80 路在解码环节每秒 24g 左右的带宽指标降低到 8G 左右,同时在 AI 识别环节也给出了跳帧识别机制,每一路监控从每秒 20 次识别输入变成 4-8 次识别输入,综合下来以后,原本 cpu 中的每个 numa 消耗从 90%也降低到了 50%以下.

第三步是再次并入编码重建,这一次,80 路支持编码重建的监控被运行起来了.机柜中每服务器都插了 4 张 gpu,唯一的问题是 gpu 的算力消耗有点超出预估,80 路帧率为 10 的 720p 监控视频在编码重建消耗了 30%的 gpu 算力,预估为 20%以下,这会影响 AI 识别环节算力开销,仍然小概率会出现内存无限暴增现象.

第四步是给出进程保护系统,这就是平时大家在社区所看到的带有 numa 和亲和性的保护系统,只要 AI 服务器出现内存暴增,那就重启那个 AI 服务进程,如果达到物理内存临界点,那就重启整台服务器.

下图为 AI 端的两个版本,分别是运行于 windows server(desktop experiment)和 windows server console.
整个解决办法环节都是通过配置 AI 端的脚本来干.具体细节会在 AP 服务器文档详细说明.





完成编码重建后的视频片段可以有精确时间搓,并且视频体积可控,视频质量可控,这些都是阵列系统最需要的功能.

传输优化

AP 服务器=AI Processor 服务器,这个服务器是作用就是部署在机柜里面的主力服务器,通常一台插了 4-8 卡 GPU 的物理服务器可以跑 2-4 个 AP 服务器进程,这些 AP 进程以不同的 NUMA+亲和性来部署.在单台 GPU 服务器可以带 80 路监控的条件下,每路监控每秒会有 4-8 次 AI 识别输入,80 路每秒就会有 600 条 AI 识别结果往阵列中心发送,同时,大约每 30-90 秒会有一个视频剪辑片段,这就是 NVR 的 MP4 数据,这种数据在 6 代监控是走的 h264 裸码流,数据体积大约是 30-200M 之间,视变码率而定,如果画面中有许多人在移动,或则受车灯光照这些因素影响,运动向量的

变码就会很大,这也是 30-200M 悬殊差原因.这里取一个中间值,每 60 秒 80M,80 路就是每分钟会有 4.8G 左右的视频数据传输.

现在给 GPU 服务器的数据传输做一个综合性的计算:AI 数据+视频数据=GPU 服务器每秒上传在 100M 左右,这种流量已经接近并且达到千兆交换机/路由器的极限,真实情况是,因为监控大多数时间画面都是静止,80 路走 720p 的 ap 服务器出口流量会在 50M/s 左右,这种流量是 365 天+24h 持续的,细节部分会涉及到多层交换机的 truck 口,以及光口,电口,电缆,光纤,这些基建环节的工作,拓扑网络是 6 代监控比较难搞的一个环节.6 代监控的服务器都是按机柜运作,单独电口无法带动多台 ap 服务器传输.

在传输优化环节,主要使用 ZNet 的 CompleteBuffer 机制,大家可以参考下面的开源项目,在 2022-2023 期间,ZNet 针对数据中心的万兆网络做了很多深度优化,这些优化都围绕 CompleteBuffer 机制来干.
https://github.com/PassByYou888/ZNet

存储策略和 IO 优化

存储策略和 IO 优化是相辅相成的.6 代监控的磁盘阵列主要处理"AI+编码重建视频"两种数据,而存储技术主要使用 ZDB2 数据引擎,这是用于设计专用数据引擎的体系,解决磁盘阵列这种需求最合适的做法就是设计专用数据引擎.

使用 ZDB2 设计出的所有数据引擎都是离散模型,一个数据库会由 n 个小库组成,这 n 个小库就是 n 个独立以 zdb2 格式存储的文件,在数据引擎工作期间,会吃满 n 条线程,其中每条线程负责各自 zdb2 存储文件读写.这里需要注意:如果数据以连续不断的方式输入进来,zdb2 工作期间会吃满 n 条线程,因此 n 值是与磁盘阵列+cpu 核心是对应关系,这导致 6 代监控阵列服务器是有 cpu 核心数量要求的.

磁盘阵列和 zdb2 的关系是大文件 api:在磁盘阵列中,磁盘可以被映射成系统目录或则是驱动器,这些目录与驱动器以路径方式来表示,因此大文件 api 通过指明路径从而得到磁盘阵列的存储资源.

6 代监控阵列存储策略:关闭即复位

6 代监控默认阵列策略,只要关闭数据中心的阵列服务器,所有的数据都会被重置,这种模型的优点是快速启动和关闭,缺点是一旦重启所有数据都会丢失.在数据中心以无参数直启服务器时,默认以该策略运作.

6 代监控阵列存储策略:读写分离

也叫 IO 分离,既写入时不会发生物理 IO 写入,而是仿真写入到临时内存,在读取数据时仿真写入的数据是有效的,只有使用 flush 机制才会将仿真写的临时内存同步到磁盘中.其中 flush 有等待和异步两种模型,等待模型是等物理 IO 返回则表示完成物理写入,异步是直接 flush 到操作系统的写入缓存,如果缓存容量可以容纳本次 flush 的数据体量,flush 会立即返回,否则 flush 会等待缓存队列,这种等待不是等物理写入而是等缓存空闲.

读写分离机制具备抗断电能力,阵列服务器可以直拔电源,数据只会丢失最后一次 flush 的时差,时差丢失长度大约在在 1-5 分钟间.读写分离机制对磁盘和内存有硬件的要求,同时也要求深入理解操作系统的阵列工作原理.满足这两点,可以做到用单阵列服务器替代 10 台 NVR.

使用读写分离策略以后,会延长阵列服务器的启动时间,具体延长时间根据对阵列工作原理掌握程度而定.

6 代监控阵列存储策略:热备+冷备

备份需要区分 zdb2 数据引擎的备份和操作系统的磁盘阵列备份,无论哪种备份模型都不适用于 6 代监控.当数据规模达到>100TB 体量后,还原等同于数据中心长时间瘫痪,大数据还原时长会从 10 小时到 7 天不等.其次是监控数据即巨大又廉价,不是金融类数据,因此任何备份机制都不适用于监控.如果不小心在数据中心敲入热备命令,只有等第二天备份完成后再来删除,并且当天数据中心 IO 性能会下降为 50%,传导给子系统以后,所有来 AP 服务求的写入,监控客户端的视频回放,响应速度都会下降 50%.
读写分离是专门针对 6 代监控的阵列系统所设计的机制,读写分离机制也可以用于替代传统备份功能.

6 代监控的阵列数据类型:AI 识别结果数据

这种数据的典型特征就是数量庞大,磁盘空间占用极少.因此所有的识别结果都暂存在内存中,磁盘阵列环节主要作用是将内存数据以差分方式同步到物理磁盘中.这样干是因为数据条目庞大,如果每个条目读取都跑去触发 io 这会很耗时,故直接走内存路线.

AI 识别结果大约每 1000 万条数据会消耗 60G 物理内存+30G 磁盘空间.

6 代监控的阵列数据类型:视频编码重建数据

编码重建的数据特征是数量不多,单条数据的体积高达 30-200M,以中间值计算,2 个机柜服务器的视频编码重建大约在 50-80G/每分钟,因此主要使用阵列作为主要存储方式,物理内存只用于暂存时间,长度,帧率,编码,监控头这些关键信息.800 路监控的单日数据规模大约会在 50-85TB 左右,这时候,如果需要查询和下载监控数据,在物理阵列层面,依靠堆内存无法减缓 hdd 寻道开销.在另一方面,基于 ZDB2 体系所设计出的数据引擎都是滚动形式的存储,当监控的数据规模达到临界,会启用删头增尾机制,这种滚动机制的性能硬伤也会是寻道开销环节.

优化编码重建数据的效率关键在于物理阵列的部署,以相对较低的成本,搭建出高性能的阵列系统,实现替代 NVR 的分布式模型,等同于在技术层面占据了制高点,在解决了建模数据来源,图形,各种外围 app 的加持下可以实现对传统的以旧换新.

对磁盘阵列总结

磁盘阵列服务器+ap 服务器是整个 6 代监控最核心的底座部分,外围的大屏幕,监控室,识别目标报警,人脸验证,手机平台移动端,web 界面端,erp 生产端,这些环节可以无限扩展升级,从项目整体来看,外围属于跑量的业务级开发,使用资金+时间来推进.

替代 NVR 的磁盘阵列系统在内部是堆结构算法运行的,这也是整个 6 代监控系统中最繁琐的技术环节之一,系统内部复杂度仅次于 ap 服务器.

在下手做磁盘阵列前,做过一次关于替代 nvr 的技术可行性探索,这些探索属于前期的调查+准备工作,如果没有这些前期的验证铺路,磁盘阵列是无法并入到 6 代监控并应用的.在 2023 年我已经将这些探索工作开源,提交到 git.
https://github.com/PassByYou888/zMonitor_3rd_Core

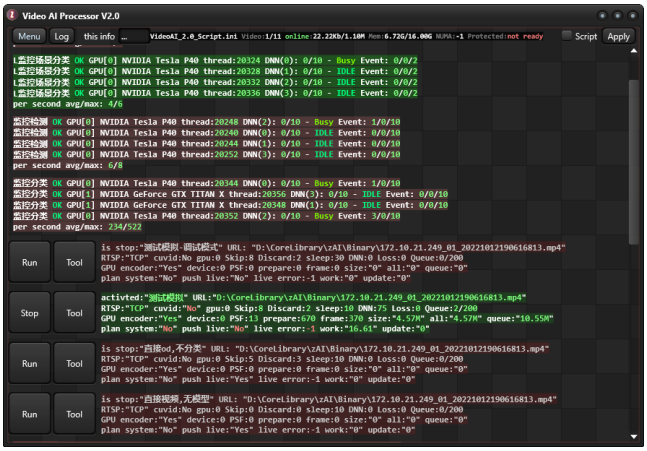
隐藏 bug

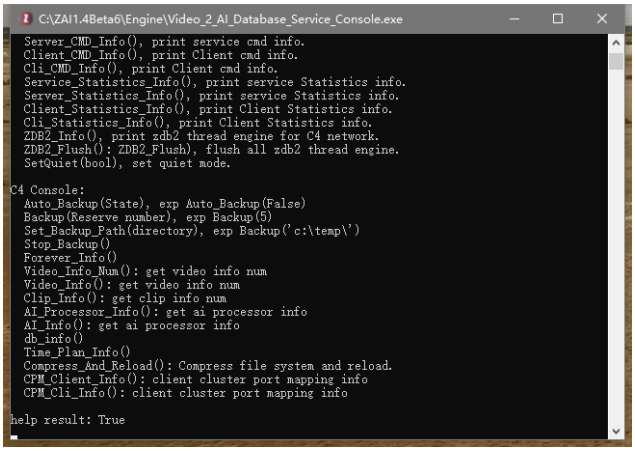
6 代监控的推进方向是走的堆算法和结构,修复 bug 的支持技术体系是整个项目中的一部分.

解决隐藏 bug 的前置工作是寻找隐藏 bug,这里需要说明一下,在内存平均使用规模>100GB 且长时间处于运行状态的单进程服务器程序中,也就是处于运营状态的后台系统,寻找 bug 就是分析运行状态,外部分析可以使用进程管理器,跟踪目标进程的内存数值,cpu 数值,句柄数值,线程数值,然后结合发生问题的时间来推断服务器正在干什么事情,例如,正在转存来自 ap 服务器的视频编码重建发生问题,正在查询某些时间段的数据发生问题,做这件事需要非常的了解服务器的程序内部设计,也就是需要达到可以对这些服务器做二次开发的程度.依靠知识,是最好的 bug 修复工具.其次就是内部分析工具.

内部分析工具是把服务器运行中的某些实例,变量,以状态的方式输出来,这些状态是外部工具,例如进程管理器无法探测出的状态.整个 6 代监控中几乎所有的服务器都有详尽的运行状态分析工具,这些工具大都以命令行形式来使用.大型程序,或则,高复杂度流程,这里不会区分服务器或则是 app,只要程序规模大了,都会有内部分析工具一说.例如在磁盘阵列系统所有的分析工具都可以使用 Help 命令来查找,而在 ap 这类工具中,是直接把流程内部状态画到 app 的 ui 上面.

下列左图为 ap 服务器和数据中心,在 ap 服务器中没有花哨的操作界面,就是存粹的把 gpu 和每一路监控运行状态直接画出来,这样干,既能帮助发现 bug,也能清楚整个 idc 的拓扑网络状态.因为更新状态频率很高,所以 ap 服务器选择使用 dx 路线的 ui,也就是 fmx 框架.在设计思路层面,这种含有大量内核数字化的信息的 app 就是带有内部分析工具机制的做法.下列右图为数据中心的 help 界面,这是走 console window 的内部状态做法,通过命令行敲入来达到内部状态分析的作用.





内部分析工具在许多大型流程,操作系统,网络游戏,手机游戏,游戏引擎,都有各自的呈现方式,会不拘一格,主要作用就是用于分析隐藏 bug,这是一种现代化项目做法.

推进落地给项目开新功能

Z-AI 的开发运作是先做建模,亦或,准好建模的 gpu 和数据,在这时候,会针对目标需求给出一个 runtime 系统,也就是落地交付的 runtime 系统.6 代监控的 runtime 是分阶段进行的,首先是核心完结,核心是磁盘阵列(数据中心),AP(跑 AI 识别的服务器群).当核心完结以后,才是替代 ie+activex 的监控前端,AI 可视化,web 查询接口,subscribe 识别结果上报,hls 直播,erp 自动化,微信小程序,web 业务化,等等这类外围的 app.

这里需要明确理解,6 代监控是一种项目,在 6 代监控中 AI 只能算一项地基,更多的会是围绕 AI 的一系列 app 来支撑应用交付.把这件事做到位会需要花钱.认为业务第一,轻视技术,不把开发外围视作重大战役,极大概率会死在开发过程中.这是一种对生命和时间的浪费,并且也会被 600585 视作耻辱级交往.

清除理解误区

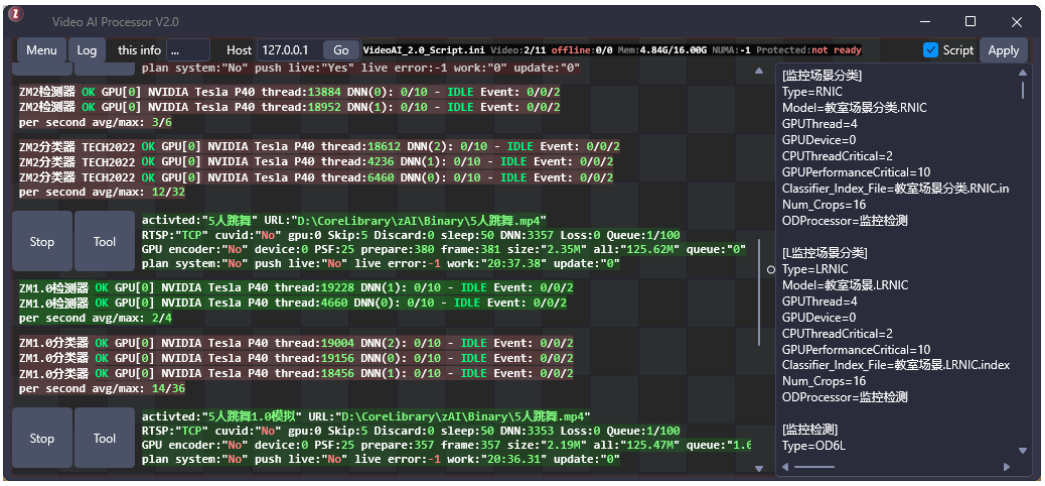
在技术圈中有许多框架,这些框架会对开发 app 起到提速的作用,但框架这种东西都是针对非常有普适性的目标需求做的.很多领域项目立项前,框架会被筛选掉,凡是有框架支持可以廉价和快速开发的领域,会被避开,或则走廉价外包路线.开发 6 代的外围 app 切勿与急速+高效率挂钩以跑量思维去对待,需要当成重大战役,以面对冷酷级对手去对待.只有这种状态才可以做出可以交付的产品.

生产工艺

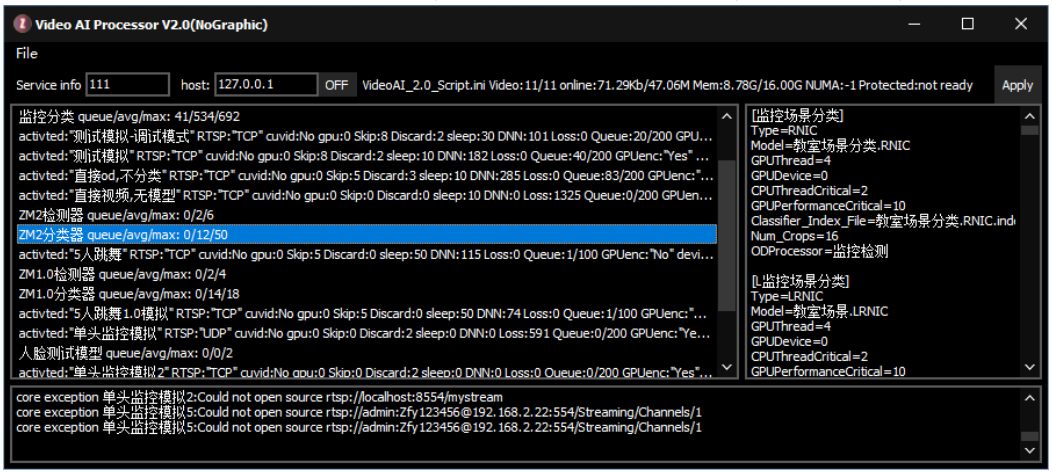
本小节是指:从筛选样本,标注,训练,测试,把模型并入 6 代监控,整个操作上的流程就是生产工艺.6 代监控是完全贯通流程,既有驱动模型的业务流程,也有稳定可靠的建模体系和外围 app 接口.

生产工艺是从 AI->业务的雪球体系,这一步是工艺级,也是整个技术栈最核心的工作,目前是闭环完结状态.

接入 AI 模型的环节主要是 AP 服务器,这种服务器目前有 2 个版本,下图为带 ui 的 windows 版本,使用 dx 作为图形状态呈现,8 卡 gpu 这类服务器使用该版本,右边的脚本就是模型脚本接口,由自动化工具生成脚本,具体细节会在 ap 文档详细说明.



下图为 no ui 的 windows core 版本,工作于定制硬件服务器,例如 AI 盒子,小型服务器



设计生产工艺的目标是面向人类的,从 AI 建模到接入监控跑业务需要有一套符合人类工作习惯的流程.

生产工艺不是一步到位,这是雪球模式:直接现场先让项目运作起来,然后,生产工艺不断的升级,传导给系统集成员和外围 app 开发员,系统集成员不断的反复建模+摸索最高识别率+最优化的服务器配置(换 gpu,换 cpu,换主板,换交换机,配 ap 参数),外围开发员反复去增加和修改外围 app.

完结后的生产工艺形态:系统集成员不再面对反复调教服务器设备,只需要确保设备和网络正常工作,软件和脚本环节已经进入自动化时代,并且常用的设备已形成系统集成经验,外围 app 开发员不再需要自己分析 AI 数据,只需要自己做个 http 服务器来接收分析结果,直接处理业务环节就行,很多专业外围例如,监控员系统,数据员系统,AI 业务自动化分析系统,基本已经完结了.后续文档会有详细使用讲解.

生产工艺是靠堆钱滚雪球做出来的.

主体架构总结

本文所描述主体架构不是完结一个框架,然后用框架做监控,而是讲解推进 AI 项目从开发到最终达成交付的方法.

6 代监控的开发是多种技术体系同步推进模式,在 AI 建模环节有建模工具,训练工具,测试工具,当 AI 建模通过测试会面临接入 AI 模型到监控,最后是用户端的视觉呈现,以及把 AI 识别结果给业务化,这一步也就是外围 app.

推进整个项目的进度需要在一个平台上做,平台就是数据中心+AP,在前沿圈,这种开发模式叫模子,当模子被开发出来,需要技术做到高层化建筑支持,剩下的都交给时间和金钱,通过不断的滚流程,不断的增加外围 app,不断的对模子升级改进,当推进项目达到一个指标,会交付变现完结合同.对于监控产品化,这会是一件长时间的工作.

在主体架构后续文档,会对外围体系如实做出全面而详细的描述,会依照项目实物来撰写.

已经完结的外围

6 代监控的外围非常庞大,未来这些外围的研发一律围绕主架够思路运作.黑标表示完结,红标表示待完结.

- **AI 建模端**:这就是 Z-AI 的整个建模体系,包括数据采集工艺,标注工艺,接入模型到监控的工艺,这套体系非常庞大,目前已完结.
- **单机 AI 监控仿真系统**:把 AI 并入到 6 代监控会是一件繁琐的工作,需要同时修改数十台 GPU 服务器的运行脚本,因此在部署 AI 模型前需要走一次仿真模拟,目前已经完结.
- **监控数据采集端**:做 AI 建模会需要现场数据,大规模数据采集是一件很有必要的功能,单次的采集数据规模会>100GB,并且采集频率很高.目前已完结.
- **专用历史回放系统**:目前使用监控数据采集端来顶替回放功能,就是下载过程比较繁琐,而专用回放系统,目前在补完状态.
- **监控员端**:也就是管理中心端,支持室内大屏幕,目前已完结.
- **序列化行为算法 app**:这一环节目前是一个外围 app,同于统计目标时间范围内的 AI 识别结果,目前已完结.
- **序列化行为 web 接口**:这一环节目前基于 ics http 框架给了一个 get 查询接口,功能与行为算法 app 相同,目前已完结.
- **监控转内网和外网的直播子系统**:可以支持外网手机观看,目前已完结.
- **IP 桥系统**:桥系统是搭建在摄像头 IP 到用户端之间的通讯桥,等同于 NVR 的 RTSP 转发服务,桥系统也等同于拓扑网络的一致化接口,例如在 6 代监控项目履行中,遇到一些棘手问题,可以网购 AI 盒子解决,而接入方法是通过 IP 桥.目前已完结.
- **自动化数据配置系统**:摄像头端会成百上千,这些摄像头大都通过 IP+命名进行数据化配置,网络分流,GPU 群集化负载,系统集成调教,这些繁琐工作是无法单独开展的,因此在 6 代监控走自动化工艺,通过一张 excel 表生成大型监控运作的脚本.目前已经完结.
- **时间分载系统**:用于定时开关摄像头群,主要几百路监控同时运行的数据量过大,浪费数据中心资源,这时候给出时间计划,在数据中心统一化的调度摄像头群.目前已完结.
- **人脸识别**:6 代监控中的人脸系统目前还在补完状态,目前我们是以靠购买人脸机实现的交付.
- **Subscribe 系统**:把 AI 监控业务傻瓜化的系统,也是 AI 盒子替代系统,目前在补完状态.
- **各种外围系统**:人员进出入管理(需要人脸),车辆进出入管理(买盒子),考勤管理,ERP 管理,各种 web,微信小程序,这些用户自己做.
- **策略层技术支持**:语音 speech 模型,GPT 模型,NLP 模型,SAM 模型,这些前沿模型都需要 GPU 服务器,6 代监控的后台全是 GPU 服务器,这些可以集成给外围开发使用.目前在补完状态.

全文完.
2024-4