创新点:

一、

平台相较于之前已有的系统，通过多种手段提升了系统的并发量，可以在实际生产生活中更好的应对高并发场景。此外我们的检测平台相较于传统系统功能更加丰富，数据更加丰富，界面更加美观，以多种图表来展示检测结果的数据状况，提升了用户体验。最后,我们系统基于强大的Spring Security框架和流行的jwt token生成方案实现了平台的安全模块,平台支持标准的RBAC权限模型,同时兼顾了xss、csrf等攻击的防护,在生产中我们的平台安全性有确定的保障。

1.配置服务器增大内置工作线程数量,提升服务器的处理能力;同时我们通过定制客户端和服务器合适的keepalive值,以合理的长连接方式减少了短时间内频繁的客户端与服务器的请求连接,节省了服务器的资源

2.前端Vue+后端SpringBoot+算法Flask的架构,在基于前后端绝对分离的基础上实现了动静分离,将静态资源放nginx,视频资源放cdn,提升了资源的加载速度,提升了系统的运行效率。同时,我们现有的架构保证了模块与模块之间的松耦合,提升了系统的可扩展度,无论后期我们何时需要改进业务,都可以如鱼得水。

3.基于内存数据库Redis实现了平台的缓存模块。将频繁访问的数据预加redis，既减少了系统与数据库的频繁交互又加快了数据的访问速度。

4.在处理视频检测结果的时候使用了消息中间件rabbitmq，通过异步的方式处理我们检测的结果,避免了服务器的阻塞等待,提升了系统的并发度。

由于在实际生产环境中,并发过大往往会伴随同一时刻内流量过大，而过大的流量会压垮系统,为保证我们平台的可用性,这里也采用了可靠的限流措施。

5.基于令牌桶算法,使用RateLimiter实现了系统的流量限制。令牌桶的算法即提前预加桶中固定数量的令牌，请求过来的时候只有获取到令牌的请求才可以继续访问，否则系统即拒绝当前请求，保证系统的可用性。

6.由于过多的多线程使用会导致频繁的上下文切换以及为保证线程安全使用锁的性能消耗，因此使用java的线程池机制并发的处理固定数量的请求，同时将不能及时处理的请求以队列这种数据结构进行缓存。整体将多余的请求排队处理，缓解了大流量对系统的压力。

二、