**项目总结报告**

日期：9/5/2018

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 6 | 项目名称 | 慧眼识踪 |
| 编程语言 | Html,css,javascript,java,python | 开发平台和框架 | 平台：macOS High Sierra, Windows 10,CentOS 7,ubuntu 16  框架：react,springboot,tensorflow |

|  |  |
| --- | --- |
| **软件需求特性** | |
| 1.是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。  原先的需求中，“在开发板进行开发”一项由于条件限制，经组内讨论后取消，改为实现API管理、监控、日志管理等运维方面的改进并已实现；同时原计划中的响应时间、容量等性能需求，由于服务器性能的限制未能完全实现。  此外，所有立项时的需求均已实现。  2.采用哪种架构风格？哪些设计模式？  系统采用B/S架构。应用部署采用微服务架构，以远程过程调用替代进程内方法调用。   1. 在运维管理相关的docker镜像部署时，采用代理模式，使用caddy作反向代理，提供简单身份认证机制。 2. 前端页面使用了React框架，整体为MVVM模式。 3. 前端在后台管理界面，使用了备忘录模式，编辑修改一条数据之前，将原本的数据备份，若取消编辑，则恢复为原来的数据。 4. 视频直播采用的是中介者模式。Nginx服务器作为视频中转服务器，是摄像头与网页的中介者，不同摄像头经过中转服务器暴露统一的直播接口。 5. Prometheus监控系统中使用适配器模式，我们的服务各不相同，各自很难输出Prometheus需要的数据格式，通过配置对应的data exporter作为适配器，使得不同的服务都能输出Prometheus格式的数据。   3.技术方案有哪些亮点？   1. 使用微服务架构，每个服务开发之间相互独立，只需暴露远程调用接口即可互相通信。将每个服务打包成docker镜像，集群部署、负载均衡十分快捷方便。 2. 使用Rancher作为容器管理平台，自动为集群内每台主机安装基础设施。提供cattle、swarm、kubernetes等容器编排、调度框架使用。自带应用商店，可从应用商店快速部署流行应用。自带身份认证机制。 3. 使用Nginx-rtmp视频中转服务器，而不是简单地本地摄像头调用，更符合实际应用场景。 4. 使用Prometheus+Alert Manager+Grafana的监控系统，不仅仅提供对主机、容器、服务使用情况的监控，而且提供美观的可视化，配置了Discord和企业微信的自动报警机制。一旦服务挂掉或者CPU、内存占用率过高就发出警报。 5. 使用Elasticsearch+Fluentd+Kibana的日志管理系统，不仅仅提供对于服务、容器的日志存储、管理与查询功能，同样提供美观的可视化。 6. 使用Rancher-pipeline快速实现CI/CD。   4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试？  测试过程中对系统的每一功能性需求均做了单元测试，并针对各种功能的不同组合做了集成测试，同时使用了外部工具进行了压力测试以测试性能，进行了兼容性测试等非功能性测试以检验非功能性需求。 | |
| **项目组成员对项目的贡献度（%）** | |
| 注：总计100%  金瑞洋：20%（监管项目进度、调度任务、编写文档、协助识别技术）  王见思：30%（项目架构设计，前后端开发与运维）  宋逸凡：25%（前端开发、协助识别技术）  李翌珺：25%（主管识别技术） | |
| **软件度量** | |
| 软件代码行数（不包括注解行、空行和复用代码）： | 3993 |
| 复用他人代码行数： | 6705 |
| 类的个数： | 28 |

|  |
| --- |
| **经验、教训和建议** |
| 1.着手开始做一个项目之前应对项目各方面做好充分了解，包括但不限于所需资源与已有资源、相关技术细节、具体需求等。  2.开发时应注重效率，每次工作都要有所进展，并将精力集中在最核心最迫切的需求上，避免拖沓。小型项目尤为如此。  3.各个成员之间应分工明确，通过组会确认进度和计划。不同成员合并各自的工作成果时应尽可能使接口简洁明了，避免衍生出过多的额外工作。  4.充分利用已有的开源代码和工具，如无必要则避免“造轮子”。  5.硬件上的性能限制使开发受到了不小的阻力，有钱的话还是要用好一点的硬件设施。  6.做好服务监控与日志管理，有利于尽早发现可能的问题。  7.架构应结合实际硬件条件与项目规模进行设计。 |