毕业实训进度报告四

本周主要关注在实践进展上，理论进展仍处于 kubernetes in action 阅读完前 1-8 章的进度。

在实践进展方面，设计了整合 Monitor 和 Predict 模块的后端，其中后端主要通过 Flask 架设；前端基于 Vue 架构，撰写了 Pods，Deployments，Services 和 View 等 component。前后端采用 MVVM 分离式架构，前端监听端口 3000，后端监听端口 5000，前端页面通过 Vue 的 Created hook 对后端接口进行访问，后端返回 JSON 格式数据。这是基本 Monitor 模块的整体架构。

在周四与封彬彬学长交流后，基本 Monitor 模块增加同步时间功能，考虑到后续需要整合多个容器之间的相关性，需要容器的监控时间完全对齐，我重构了前端 Pods 组件，用户可以通过复选框选中多个容器，设置监控时间，对他们进行统一操作。这一数据发送到后端，后端分析每个容器后针对每个容器都创造一个监控线程，并把他们添加到整体监控数组中。该监控线程每隔 15s（可配置）访问 Prometheus metrics 并汇总这些 metrics，将他们以时序数据的格式存储到 InfluxDB 中。

预测模块的设计比监控模块更为复杂，因为预测模块需要动态的图表数据更新，在原来客户端以 HTTP 协议请求后端数据的基础上，需要后端主动式的推送正在监控中的容器的最新数据，HTTP 协议不适合实现这种双向沟通的架构。于是我查阅了相关资料，最终确定了在后端使用 Flask-socketio，在前端使用 vue-socket 使前后端能基于 Websocket 协议通讯。特别的是，该 websocket 长链接只在打开可视化图表页面的时候建立，在关闭可视化图表页面前断开，这样该长链接并不会影响 HTTP 协议消息的发送和收取。

同时我设计了每个监控预测线程在 websocket 协议消息中有独有的名称（包含容器和预测方法/监控方法），这样前端将只订阅特定容器特定监控方法的消息，后端可同时推送多个容器或同一容器多个预测方法的消息，而前端并不会把这些消息混为一谈，只听取自己订阅的消息。整体设计模式类似一个只能听不能发送消息的聊天室，消息由后端统一发送。