结项报告

一、项目信息

1. 项目名称

基于gazelle实现资源可视化

2. 方案描述

- 为 `gazellectl` 集成 `-m` 命令，支持带有时延参数，可实时查看 `gazelle` 大页内存使用情况
- 在 `gazelle` 内部，按照 `mbuf`、`rpc`、以及 `fixed`(固定内存)三种情况划分大页内存使用情况
- 按照 `lstack` 线程、`socket` 划分协议栈中 `mbuf` 的使用情况，并显示给用户

3. 时间规划

1）准备阶段（6.08-7.08）

- 阅读 `gazelle` 源码
- 学习 `gazelle` 中使用相关工具
  - `DPDK` 内存管理
  - `LwIp` 网络协议栈

2）预备阶段（7.08-7.20）

- 源码编译 `openeuler lwip`
- 以 `mysql`、`redis` 作为项目背景，运行调试 `gazelle`

3）编码阶段（7.20-9.20）

- 实现 `gazelle` 相关内存统计
- 将内存统计集成到 `gazellectl` 命令中

二、项目进度

1. 已完成工作

- 完成 `gazelle` 大页内存使用计算和统计
- 将内存统计结果集成到 `gazellectl` 命令中

2. 遇到的问题及解决方案

在参与 gazelle 项目之前，我对于网络在系统层面的认识还停留在 posix 接口中的 socket 和一些 socket option，对于整个网卡中数据包在内存的传递，以及网络协议栈对数据包的处理还一无所知。在阅读完 gazelle 源码以后，看到在网卡从接收到流量，从接收队列到 memory ring 再到最终传回 socket，还有数据通过缓冲区被用户发送到网卡时，我对于整个网络数据包的硬件流程有了更加清晰的了解。同时 gazelle 中 io 多路复用的实现、rpc 消息的处理也让我对这些编程概念有了更深层次的理解和认识。

在整个项目开发过程中，与我而言最难的部分在 lwip 中找到 socket 和 netconn 的使用逻辑，以及其中 send_ring、recv_ring、acceptmbox 还有 recvmbox 、sndqueue 等缓冲区中 mbuf 的传递和复制。整个网络的收发包流程以及 dpdk 的内存结构设计逻辑和 api 接口功能，前者导师给了我详细的指导，后者为此我在 github 中找到了 dpdk 的源码深入阅读，为之后内存计算提供了许多思路。

在此，也非常感谢 openeuler 社区以及杨宸导师给予我的此次参与开源项目的机会。感谢导师在完成项目过程中给予的帮助和指导，同时还感谢社区其他成员对代码的 review 以及提出的修改意见，让我对于编码规范和编码思路都有了更深刻的理解和认识。

3. 后续工作安排

- 等待项目 pr 合并
- 修复可能出现的 bug
- 后续可根据实际需要继续完善项目