调研任务：

（一）任务分工如下 图

- x-selvm
- x-orz
- x-qwq

（二）需要调研的内容

++ 对每个项目，需要记录：

1. 项目简介：用尽可能通俗的语言介绍项目的内容、目的、效果
2. 关键词：涉及的领域，使用的主要知识，语言等
3. 难度评价：这一评价的主观程度较高，建议从"需要的知识广度"、"复杂性"等角度详细记录下，方便其他队友了解
4. 其他：在研究过程中，发现的其他有用信息

++ 文档的最后，需要写下：

1. 自己感兴趣的领域
2. 要想研究的课题，设计思路

（三）DDL 3 月 6 日周日中午 12：00 之前，上传调研文档到群中。 阅读队友的文档。

周日下午 4：00，在西图研讨室进行线下交流 晚上可约饭

x-orz

介绍

极度省流版： 用 Uigniter（用于管理 Firecracker 集群）+Firecracker（虚拟机）+OSv（Unikernel 一种，微操作系统）做了一个服务器，做到了模块化设计（模块化运行应用程序以满足服务端需求）。

> Unikernal 可以理解为仅保留程序依赖的包的操作系统，做到了极度简洁，启动快，特别小

以下是文档内的项目介绍

> 随着云计算的兴起，越来越多的人们使用云平台来运行自己的服务与应用。为了提高资源的有效利用，云提供商常常会使用虚拟化技术将同一台物理机划分给不同的云租户。Unikernel是继Docker之后的又一种虚拟化技术，Unikernel 是将应用代码及其依赖与 libOS 一同打包的单一功能的单地址空间镜像，它具有轻量，快速，结构简单，安全，不可变基础设施等等特性，这些特性也使得 unikernel 非常适合于微服务（microservice）以及物联网（IoT）等等的架构。

我们参考研究了工业控制系统的结构。其大体的工作流程是在interface部分利用传感器等采集信号，然后通过Information Processing部分进行信息的处理，最后在intelligence部分对系统进行智能控制。我们的项目选取了其中的信息处理部分的一小部分，将应用进行解耦与模块化。将相对独立的功能封装进 Unikernel 运行，来发挥 Unikernel 快速，安全，轻量的优点，满足相应需求。

代码结构

Language	files	blank	comment	code
JavaScript	17	283	77	1368
Markdown	12	694	0	1184
Python	3	55	29	195
C	1	5	3	94
Bourne Shell	4	15	8	47
Pug	2	6	0	25
JSON	1	0	0	21
make	1	3	0	7
SUM:	41	1061	117	2941

还有go语言(项目中用了submodule 即uigniter的代码不在这仓库里)

难度评估

首先是干了啥：

> 尝试各种Unikernel实现，进行选择 编写模块化应用并在本机运行测试 打包Unikernel，在Unikernel之间实现网络通信 编写底层Unikernel管理单元Uigniter 加入auto scaling特性 进行性能测试

总结：

1. 安装OSv firecracker 并配置环境
2. 团队使用了Go语言实现了一个轻量的OSv管理工具：uigniter
3. 编写模块化应用并打包
4. 感觉有点难度

备注：

项目文档末尾附有OSv和firecracker的说明 包括安装 打包镜像 后者启动前者 打包模块的过程说明，所以如果有兴趣接手的话难度相对较低

x-selvm

介绍

> x-seLVM项目旨在利用Lua虚拟机，使seL4内核对平台的依赖转化为Lua虚拟机对平台的依赖，从而扩大seL4微内核的平台适用范围。主要的技术为将内核的部分接口进行改写，并将其注册到Lua程序中，实现运行在Lua虚拟机上的seL4内核。

> 本项目的目标是要提高seL4的平台适应能力，而Lua虚拟机几乎可以在所有的平台上运行，并且提供了对C语言的简单接口，因此计划将seL4的API进行部分改写，将API方法注册到Lua程序中，通过Lua程序

> 在虚拟机上的执行来实现跑在Lua虚拟机上的seL4内核，从而实现可以跑在各个平台上的seL4内核。

省流：将seL4的API进行部分改写，将API方法注册到Lua程序中，通过Lua程序在虚拟机上的执行来实现跑在Lua虚拟机上的seL4内核，从而实现可以跑在各个平台上的seL4内核。

使用语言

- RUST

  由于项目包含了大量依赖项和软件本身的代码 统计图一乐

  RUST语言一共写了4558行

  反转了 都是x-qwq项目的

- LUA语言

  一种小型脚本语言 很容易被C/C++语言调用

  统计中代码量非常小

  此外该项目重写了大量c语言函数以便让lua脚本调用

  注：该项目的LUA改写内核未能成功链接，没有通过测试

难度评估

1. 学习成本上要学RUST语言和LUA脚本语言
2. 要学习一点seL4项目
3. 对源码编译能力要求高（链接各路文件）、
4. 设计上有一定难度
5. 反转了 该项目基于x-qwq项目 难度略低

备注

该项目未能走到最后 有兴趣可以接手并优化

x-qwq

项目简介

> 我们的项目是基于 *Rust* 改造的 *seL4* 微内核。
>
> L4 是一种微内核构架的操作系统内核。seL4 是 L4 微内核系列的一种，经 Haskell 形式化验证并实现为 C。
>
> 我们计划将它改写为 Rust，进一步提升安全性。

省流：用RUST重写seL4微内核以提高其安全性

难度评估

1. 学习RUST语言 与c/c++类似
2. 学习seL4操作系统 熟悉其源代码（总共就8700行）
3. 总共改写了12个文件 共4558行
4. 总结：难度感觉中等