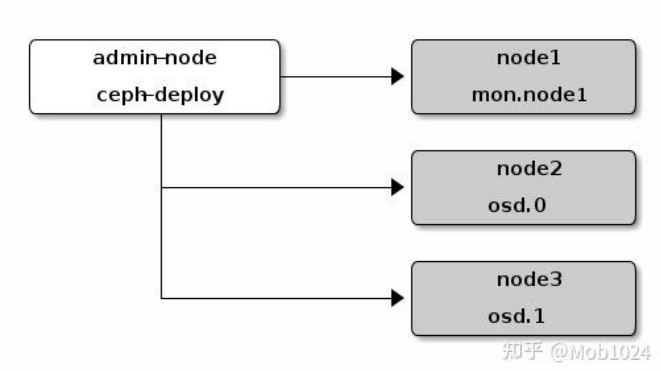
2021——x-RIG

Lab4

关键词：数据存储、集中式存储系统、分布式文件系统、 Ceph 分布式文件系统

Ceph 简介

随着信息技术的快速发展，互联网产生和处理的数据量迅猛增长。有研究表明，过去两年里产生的数据已经占世界数据总量的 90%。如何存储这些数据是存储系统相关研究的核心问题。传统上的集中式存储系统采用计算能力和 I/O 能力强悍的大型主机进行数据存储，然而目前在数据中心中存储的数据均为 PB 量级甚至更高，传统的集中式存储由于过高的设计和维护成本渐渐难以满足日益增长的存储需求。在目前的数据中心中，普遍采用大量的性价比更高的小型主机，通过分布式文件系统协同存储数据中心中的海量数据，常见的有 Hadoop 分布式文件系统(Hadoop Distributed File System，即 HDFS)和 Ceph 分布式文件系统等。



大概流程：先是烧录了树莓派镜像，然后安装 ceph-deploy 及其结点，进行多节点部署（3个）

大作业

关键词：嵌入式实时操作系统 FreeRTOS （更多介绍）、 seL4 capability

语言：Rust（主要使用语言）、C（FreeRTOS 源码语言、seL4 内核语言）

主要内容：先用 rust 语言重新实现了原本主要是以 C 语言实现的实时操作系统 FreeRTOS，并增加了 FreeRTOS 原本不具有的安全相关的措施——移植了 sel4 微内核操作系统中的 capability 机制(seL4 capability 详细介绍)

未完成的部分以及想法：由于 FreeRTOS 目前只支持单核处理器，从而考虑对 FreeRTOS 进行扩展，使其能支持多核处理器，从而提升性能上限。

难度：由于不了解 Rust 语言，对整个实验的难度难以进行客观评估，个人认为该实验实现的难点在于 capability 安全机制的原理理解以及移植过程。

附：About Capabilities

1. seL4 中的 capability 封装了对一个对象的引用和对该对象的访问权限
2. 以我们关心的 CAmkES 层面为例，一个 connector 被表示为一个 endpoint 对象，要调用该 connector 的 client component 就需要相应的 capability，其中既包含该对象的引用也包含访问权限

如上文所述，CAmkES 框架会实现这种对象的映射

3. seL4 共有 10 种对象，均需要通过 capability 来引用
4. object-oriented 对象优先
5. 内核层面检查 capabilities
6. 授权模式，用户之间的授权可随时通过摧毁 capability 以取消授权

2020——x-dontpanic

关键词：基于互联网的小型分布式文件系统、容器化技术、Erasure Code、基于 OpenVPN 的局域网、跨平台

项目简介：

本项目旨在实现可用性高的基于互联网网页的小型分布式文件系统。在已有的项目的基础上，希望实现容器化服务器端、多用户权限支持、更高效的文件传输、减轻中央服务器负担、提高文件安全性和可用性等优化，做出可用性高的“私人网盘”。

语言：Java、WebAssembly

主要内容：该项目是基于前人的项目进行完善和改进，将原来中心化的数据传输改为去中心化，以解决网络拥堵、带宽有限、安全隐患等问题；使用容器化技术，增强了可移植性，能够在多种不同运行环境下正常工作，同时保证了安全性；实现以角色为基础的访问控制；客户端实现跨平台以最大程度地利用碎片化的设备存储；使用 RS Code 编码算法实现纠删码技术对文件进行冗余，比副本备份节约了 3 倍于文件大小的空间；OpenVPN 建立虚拟局域网

难度评估：本项目由于是在前人项目的基础上改进与优化，因此整体任务量偏大，难度也偏高，所需知识面较广（涉及编码技术、容器化技术、分布式、基于 OpenVPN 的局域网、数据库相关知识等）。

附：**多用户权限支持 —— RBAC 介绍**

以角色为基础的访问控制（Role-based access control，RBAC），是一种较新且广为使用的访问控制机制。不同于其他的访问控制直接赋予使用者权限，RBAC 将权限赋予角色。在一个组织中，根据不同的职责产生不同的角色，执行某项操作的权限被赋予对应的角色。组织成员通过被赋予不同的角色，从而取得执行某系统功能的权限。对于批量的用户权限调整，只需调整用户关联的角色权限，无需对每一个用户都进行权限调整，既提升效率，又降低了出现漏调的概率。

OpenVPN 建立虚拟局域网

数据连接直接建立在用户和存储设备之间还会产生一个问题：寻址。中央服务器通常有公网 IP，但是寻常的存储设备在 NAT 之后。两个在 NAT 背后的设备没有办法直接通信。采用 OpenVPN 构建虚拟的局域网，使得当存储设备和用户不在同一个局域网内时，也能够进行寻址。

工作原理：

OpenVPN 服务器一般需要配置一个虚拟 IP 地址池和一个自用的静态虚拟 IP 地址（静态地址和地址池必须在同一个子网中），然后为每一个成功建立 SSL 连接的客户端动态分配一个虚拟 IP 地址池中未分配的地址。这样，物理网络中的客户端和 OpenVPN 服务器就连接成一个虚拟网络上的星型结构局域网，OpenVPN 服务器成为每个客户端在虚拟网络上的网关。OpenVPN 服务器同时提供对客户端虚拟网卡的路由管理。当客户端对 OpenVPN 服务器后端

的应用服务器的任何访问时，数据包都会经过路由流经虚拟网卡，OpenVPN 程序在虚拟网卡上截获数据 IP 报文，然后使用 SSL 协议将这些 IP 报文封装起来，再经过物理网卡发送出去。OpenVPN 的服务器和客户端在虚拟网卡之上建立起一个虚拟的局域网络，这个虚拟的局域网对系统的用户来说是透明的。为了充分利用直接进行数据连接的传输效率，可以对以下情况分类处理：

1. 存储设备已经有公网 IP，直接访问
2. 存储设备和用户在一个物理局域网内，直接访问
3. 1、2 外的其他情况，采用 OpenVPN 的虚拟局域网。

2019——x-Erasure-Code-Improvement-based-on-ceph

关键词：ceph、纠删码、SIMD、ceph 日志

项目简介：

在 ceph 的平台上对纠删码插件进行改进，使用柯西矩阵进行编码，使用 intel 的 SIMD 指令集机型矩阵计算的加速，对于数据进行增量的方式修改，对于不同的数据进行分层的管理，达到更好的容错性和可用性。使用日志的方式达到分布式中的一致性和速度方面的提升。

主要内容：先是进行 ceph 虚拟机搭建和 ceph 物理机搭建（附有脚本）；然后部署 Ceph 存储集群；然后利用一个实现纠删码的 C 语言仓库——Jerasue 库，应用于 ceph 纠删码实现；并且使用 SIMD 指令集加速 RS 编解码过程；最后是对实现的 ceph 进行测试。

语言：C（主要）、C++

难度：整体难度适中，实现难点应该在 ceph 搭建以及 SIMD 指令集对编码进行加速这两个方面。

附：纠删码简介

分布式系统需要在硬件失效等故障发生后仍然能继续提供服务。就数据而言，HDFS 采用每份数据 3 副本的方式，保证某些数据损失之后仍能继续使用。数据的容错除了副本还有另一种做法，就是把丢失的数据计算出来。这就是纠删码（erasure coding，EC）的思想了。它将数据分割成片段，把冗余数据块扩展、编码，并将其存储在不同的位置，比如磁盘、存储节点或者其它地理位置。

SIMD

最近几代的 CPU，尤其是 GPU，需要数据并行代码才能充分发挥效率。数据并行性要求对不同的输入数据应用相同的操作序列。因此，CPU 和 GPU 可以减少指令解码和调度所需的硬件，而更多的算术和逻辑单元可以同时执行相同的指令。在 CPU 架构上，这是通过 SIMD 寄存器和指令实现的。单个 SIMD 寄存器可以存储 n 个值，单个 SIMD 指令可以对这些值执行 n 个操作。在 GPU 架构中，n 个线程以完美的同步方式运行，由一个指令解码器/调度程序提供。每个线程都有本地内存和一个给定的索引，用于计算内存中加载和存储的偏移量。

当前 C++编译器可以将标量代码自动转换为 SIMD 指令（自动矢量化）。然而，编译器必须重构当开发人员在 C++中编写纯标量实现时丢失的算法的固有属性。因此，C++编译器无法将任何给定的代码向量化为其最有效的数据并行变量。尤其是跨越多个函数甚至翻译单元的大型数据并行循环，通常不会转换为有效的 SIMD 代码。

SIMD 优点

以加法指令为例，单指令单数据（SISD）的 CPU 对加法指令译码后，执行部件先访问内存，取得第一个操作数；之后再一次访问内存，取得第二个操作数；随后才能进行求和运算。而在 SIMD 型的 CPU 中，指令译码后几个执行部件同时访问内存，一次性获得所有操作数进行运算。这个特点使 SIMD 特别适合于多媒体应用等数据密集型运算。

SIMD 与纠删码的结合

伽罗瓦菲尔德算法构成了 Reed-Solomon 和其他擦除编码技术的基础，以保护存储系统免受故障的影响。伽罗瓦场算法的大多数实现都依赖于乘法表或离散对数来执行此操作。但是，128 位指令（如英特尔的流式 SIMD 扩展）的出现使我们能够更快地执行伽罗瓦场算法。

日志记录——osd 日志

Ceph OSD 日志也是一种事务日志，它是基于文件系统的 OSD 的关键组成部分，提供存储系统所需的一致性保证。 Ceph OSD 使用日志有两个原因：速度及一致性。

> 速度： 日志使得 Ceph OSD Daemon 进程能够快速的提交小 IO。Ceph 将小的随机 IO 顺序的写入日志，让后端文件-系统有更多时间来合并 IO，提升系统的突发负载能力。然而，这可能带来剧烈的性能抖动，表现为一段时间内的高速写入，而之后一段时间内没有任何写入直至文件系统追上日志的进度。
>
> 一致性：Ceph OSD Daemon 进程需要一个文件系统接口来保证多个操作的原子性。Ceph OSD Daemon 进程提交操作描述到日志并将操作应用到文件系统。这使得能够原子的更新一个对象（如：pg metadata）。在 filestore 配置的[min_sync_interval max_sync_interval]间隔范围内，Ceph OSD Daemon 进程会停止写操作,并同步文件系统与日志，建立同步点[译者注：此时会记录已同步的最大 Seq 号]，以便 Ceph OSD Daemon 清除日志项、重用日志空间。在故障发生后，Ceph OSD Daemon 从最后一个同步操作点开始重放日志。