课程报告：三篇新颖设计的操作系统论文 REVIEW

刘天豪 22551212

1. LEGOOS: A DISSEMINATED, DISTRIBUTED OS FOR HARDWARE RESOURCE DISAGGREGATION

1.1 核心思想：像乐高一样拼装硬件的操作系统。

传统的裸金属服务器的硬件基本都是固定的，每个节点都捆绑了特定数量的CPU、内存、储存能力。如果业务能完整利用硬件，显然可以最大化利用率/收益。但是，实际上的任务，计算/内存/储存需要的能力区别都很大。举个熟悉的例子，如果运行科学计算软件，计算能力基本是来多少用多少，永远不会嫌多，但是内存使用就相对低，储存需求就更低了。而如果运行数据库，那么内存、储存需要的压力就非常巨大，而CPU相对纯计算密集型任务就少得多。所以这样，有一部分资源就会被空置，导致浪费。单体模式在硬件弹性、故障处理和对异构硬件的支持方面也面临瓶颈。毕竟如果一个节点挂掉，需要进机房拆机子换硬件，才能重新上线。作者将问题归纳成:Inefficient resource utilization（资源利用率低下），Poor hardware elasticity（硬件弹性差），Coarse failure domain（故障域粗糙），Bad support for heterogeneity（异构支持性差）. 现在的一些容器化技术可能能解决这个问题，但是如果可以直接往别的机子上调度，就能非常快地响应故障。

作者团队为了解决这个问题。将 CPU、内存、存储等硬件转变为独立的、通过网络连接的组件，并提出了一个叫Splitkernel的操作系统模型来管理分布式的硬件。同时，他们最大胆的一点是，把计算机内几乎最高性能的一条总线——CPU-内存总线割开了，这就带来了极大的灵活性。他们使用了海量的100G IB卡来做互联，个人认为这样成本会很高，但是还是能解决一些问题的。

为了适配硬件的分离，作者对操作系统也进行了分离：

核心想法：将传统操作系统的功能分解为多个松散耦合的"监视器"(Monitors)。每个监视器在一个特定的硬件组件上运行，并管理该组件 。例如，CPU 监视器、内存监视器、GPU 监视器等 。所有监视器全部通过网卡传递消息。Splitkernel放弃支持跨组件的缓存一致性。OS本身只有两个任务：协调跨组件资源分配，以及处理组件故障。

1.2 解决方案

LegoOS 针对三种硬件组件： **pComponent（处理器）**、**mComponent（内存）**和 **sComponent (存储)** 。实现了三种监视器：进程监视器、内存监视器和存储监视器 。向用户提供的只有vNode，一个vNode跨越多个p/m/s Component. 同时，作者实现了Linux syscall接口，让Linux应用直接能跑上去。

最大的技术挑战来自于内存-cpu。LegoOS 将所有内存硬件功能（如页表、TLB）都移至 mComponent 。pComponent 只保留缓存 。由于 pComponent 无法访问物理地址，LegoOS 将其所有级别的缓存组织为"虚拟缓存"（Virtual Caches，即虚拟索引和虚拟标记）。为了解决网络访问内存的性能问题，LegoOS 提出在 pComponent 上保留少量（例如 4GB）的本地 DRAM 。这部分 DRAM 不作为主内存，而是被组织为末级缓存之下的一个"扩展缓存"（ExCache）。

LegoOS使用一个两级分布式的虚拟内存空间管理。高层（vRegions）：每个进程的虚拟地址空间被分成粗粒度的、固定大小的"vRegions"（例如 1GB）。一个"home mComponent"负责管理该进程的 vRegion 分配（即哪个 vRegion 由哪个 mComponent 拥有）。低层（vma trees）：每个 vRegion 的所有者 mComponent 负责管理该区域内的精细化虚拟内存区域。

1.3 性能分析

LegoOS 相比于拥有足够内存的单体 Linux，性能仅下降 1.3x 至 1.7x 。使用本地 SSD 或网络交换的内存受限 Linux 相比，LegoOS 的性能是其 0.8x 至 3.2x（即在某些情况下更快）。

1.4 个人觉得的提升空间

以IB、RDMA连起来的硬件总有许多限制，一些延迟高度敏感的应用可能会因为巨量的访存延迟出现问题（比如一个例子Simulink，它会把CPU的0核锁频率，来达到精确的时间步运行）。这属于比较本源的问题，解决比较困难，如果CXL被用作通信，可能会有一定的提升。

2. DBOS: A DBMS-ORIENTED OPERATING SYSTEM

2.1 特色想法

像管理数据库一样管理操作系统。作者认为传统 OS 仅为单节点设计 ，导致集群管理必须依赖于像 Kubernetes 这样的外部软件层 。这种分层架构集成度很差 ，使得监控、调试、安全和数据溯源等关键功能难以实现，往往需要定制化的"附加"方案 。作者认为，Unix 的"Everything is a file"模型已经过时 ，无法满足现代应用对异构硬件、大规模并行调度和数据溯源的需求 。

作者提出了一个激进的策略：将一切化为表格（Everything is a table)， 所有操作系统的状态——无论是进程、文件、调度决策还是网络消息——都作为行存储在高性能的、多节点的、事务性的 DBMS 表格中 。这种设计的最大优势在于，操作系统最复杂的功能（如事务、高可用性和多节点支持）只需在底层的 DBMS 中实现一次，所有上层服务即可免费获益 。基于这一架构，传统的操作系统服务被极大地简化了 。例如，一个复杂的任务调度器可以被实现为对 `Task` 表和 `Worker` 表的几行 SQL 查询 。更重要的是，由于所有 OS 状态本质上都是可查询的数据 ，因此监控、分析和数据溯源不再是难题，而是变成了对 OS 状态的简单 SQL 查询 。

2.2 实现方法

作者在论文中实现了一个名为 **DBOS-straw** 的原型系统 。这是一个验证其核心思想（即操作系统可以构建在数据库之上）的第一阶段原型 。这个原型基于 Linux（作为 Level 1）和 VoltDB（一个高性能的内存事务型 DBMS，作为 Level 2）构建 。

任务调度器：调度器不再是一个复杂的 C 程序，而是被实现为在 DBMS 中运行的SQL 存储过程 。所有调度所需的状态都存储在数据库表中，一个简单的 FIFO 调度决策被实现为一个事务。只要修改一行SQL，就可以切换成另一种模式的调度器。

IPC：进程间通信不再是基于套接字（socket）的握手，而是通过对 `Message` 表的读写操作来完成 。由于 IPC 构建在 DBMS 之上，它自动获得了传统 IPC 难以实现的强大功能，例如：事务性、高可用性（通过表复制）和恰好一次（exactly-once）语义（通过在接收后删除消息）。

文件系统：作者实现了两个文件系统（本地化版本和并行版本），其所有元数据和数据块都存储在 DBMS 表格中。DBMS先天对文件性质敏感，分析即查询文件。

2.3 个人体会

在操作系统实验课上，我们粗浅了解了一些内存管理，其实就是需要管理海量的结构体，如果用DBMS管理这些结构体，也会是一种可能可以尝试的操作。

3. FLEXOS: TOWARDS FLEXIBLEOS ISOLATION

3.1 特色想法

作者指出，现代操作系统在"设计时"就被**刚性地锁定**在一种特定的安全和隔离策略中 。例如，一个操作系统要么被设计为单体内核（如 Linux），要么是微内核（如 SeL4），要么是单地址空间 OS 。不同的应用程序具有截然不同的安全和性能需求 。一个"一刀切"的 OS 设计无法在所有场景下都达到最佳 。现有的硬件保护机制可能会被漏洞（如 Meltdown）攻破 。在僵化的设计下，切换到替代的保护机制需要巨大的工程重构努力 。而新硬件也在不断加入新的隔离机制。

作者想要：把OS的安全和隔离策略选择从设计时候，推迟到编译/部署时候。

为了实现这一目标，作者提出了 **FlexOS**，这是一个基于"库操作系统"（LibOS）模型构建的新型模块化操作系统 。LibOS 传统上被用于为特定应用定制*性能*；而 FlexOS 将这个理念扩展到了*安全维度* 。

3.2 实现方法

作者基于 Unikraft 库操作系统构建了 FlexOS 的原型 。其解决方案包含两个关键部分：一个灵活的 API 和一个构建时工具链。FlexOS 的源代码是"隔离机制无关"的。它使用一套抽象的 API 来标记需要隔离的边界，在编译时，用户提供一个配置文件来指定所需的安全策略。FlexOS 的工具链会读取此配置，并执行源码到源码的转换。作者演示了在Intel MPK、EPT上的效果。

3.3 启示

安全策略是一个比较复杂的问题。如果OS有这样的灵活性，应用可以探索安全和性能的最好平衡能力，并且如果一个底层出问题了，也可以快速更新来弥补。这样可能可以修复Apple Silicon M系列的侧信道漏洞等问题，并且不带来很大的安全开销。如果硬件商埋后门，那也可以通过OS来一定程度解决。

4. 与自己的研究方向关联

我的研究方向是后量子密码学（PQC）、密码算法加速。现在PQC其实没有被特别广泛应用，一个原因就是开销太大，如果上专用加速设备，则会面临很严重的浪费问题，因为不一定有这么多加密业务，虽然FPGA/ASIC实现非常快。如果有LegoOS的思想，我加入一个cComponent，专门做加密解密调用，这个monitor可以被一个大集群共享，来达到快速处理加密解密的目的。而且这种操作天然提供了一个非常强悍的安全边界，只有加密部分管加密。FlexOS的灵活安全性也可以为PQC库部署做一个很好的启发，如果涉及密码的部分，可以配置一个最强的隔离舱，即使系统其他部分被攻破，读取和篡改私钥也是非常困难的。同时，如果有大批量加密解密需求，就开一个最高性能的，将跨域请求转换成零拷贝的，跑得最快的。这些思想无疑将帮助无处不在的密码部分与操作系统更好地协同工作。