操作系统课程报告

当前主流操作系统大多存在“固定设计”的特点，比如一开始就确定了安全隔离策略、只能适配特定硬件，后续想修改或适配新场景就得大规模调整代码，很难满足现在多样化的需求 —— 比如有的应用要高安全，有的要高性能，还有新硬件不断出现，旧系统跟不上。

而从发展方向来看，操作系统正朝着更灵活、更适配分布式硬件、更易管理、更能平衡安全与性能的方向走。未来的系统不会再“一刀切”，而是能根据应用需求定制配置；能支持 CPU、内存、存储等硬件分散部署，提高资源利用率；会用更简单的方式管理系统状态，降低开发和维护难度；还能根据实际情况动态调整安全策略，既保证安全又不浪费性能。这些变化，都是为了更好地适配现在越来越复杂的计算环境，比如大数据、人工智能应用的需求。

三篇论文核心技术解析

一、《FlexOS: Towards Flexible OS Isolation》

（一）模块化组件拆分技术

FlexOS 最基础的技术是将传统操作系统的完整功能拆解为多个独立的细粒度组件，比如负责任务调度的组件、处理网络数据的组件、管理数据库连接的组件等。这些组件就像独立的积木，彼此间通过标准化的接口通信，没有紧密的绑定关系。这种设计打破了传统操作系统“牵一发而动全身”的困境 —— 修改某个组件时，不会影响其他功能的正常运行，比如升级网络组件的安全机制，无需改动调度组件的代码。

（二）可配置隔离机制

传统操作系统的隔离方式是固定的，而 FlexOS 实现了隔离策略的“按需选择”。它支持多种硬件和软件隔离技术的灵活切换，比如可以用 Intel MPK 技术实现轻量级内存隔离，也能通过 VM/EPT 技术构建更严格的虚拟机隔离环境。用户在部署系统时，只需通过简单配置就能决定：哪些组件需要严格隔离（比如处理支付数据的组件），哪些组件可以放松隔离以提升性能（比如普通的日志处理组件）。同时，它还提供了数据共享注解工具，明确标注哪些数据可以在组件间安全传递，避免隔离过度导致的功能受限。

（三）安全配置导航工具

由于组件和隔离方式的组合选项繁多，普通用户很难直接找到最优配置。FlexOS 为此设计了“部分安全排序”技术，它能像“智能筛选器”一样工作：用户只需输入性能预算（比如要求每秒处理 10 万次请求），工具就会自动从几十甚至上百种配置中，筛选出既满足性能要求、又能达到最高安全等级的几种方案。这就避免了用户手动逐一测试的麻烦，大大降低了灵活配置的使用门槛。

二、《LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation》

（一）三级组件化监控架构

LegoOS 针对硬件分散部署的场景，将操作系统功能拆分成三个核心监控器（类似“专属管理员”）：进程监控器专门管理 CPU 和扩展缓存，内存监控器负责内存的分配与地址映射，存储监控器提供文件存储服务。每个监控器直接运行在对应的硬件组件上，比如内存监控器就部署在独立的内存设备中，这样各个硬件的管理互不干扰，单个组件出问题不会影响整体系统运行。

（二）虚拟缓存与扩展缓存技术

为了解决硬件分散后的数据访问延迟问题，LegoOS 设计了独特的缓存机制。它在 CPU 所在的进程监控器中只保留基础缓存层级，把大部分内存管理工作交给内存监控器，通过“虚拟缓存”技术解决不同硬件组件的地址空间隔离问题。同时，专门引入“扩展缓存”（ExCache）作为中间层，介于 CPU 缓存和远程内存之间，自动把常用的数据存在这里。比如运行数据分析程序时，频繁调用的中间结果会保存在扩展缓存中，不用每次都去远程内存读取，大幅提升了速度。

（三）分布式内存与 RDMA 通信优化

LegoOS 采用两级内存管理方式：有一个“主内存监控器”负责粗粒度的内存分配，其他内存监控器处理具体的细节操作，这样多台设备的内存能被整合为一个“大内存池”，供应用灵活使用。在组件通信上，它基于 RDMA（高速网络技术）定制了通信框架，这种技术能让硬件组件之间直接传输数据，不用经过 CPU 中转。比如 CPU 向内存发送数据时，可直接通过 RDMA 传递，延迟比传统网络通信降低很多，保障了分布式硬件的高效协作。

三、《DBOS: A DBMS-oriented Operating System》

（一）“一切皆表”的状态管理技术

DBOS 颠覆了传统操作系统“一切皆文件”的理念，把系统里所有的状态数据都用数据库的“表”来存储 —— 比如任务调度信息存在“任务表”里，硬件资源状态存在“资源表”里，进程间的消息存在“消息表”里。这种方式让所有数据都有统一的结构，就像把杂乱的文件整理到规范的表格中，无论是查询某个任务的运行状态，还是统计资源使用情况，都能通过简单的 SQL 语句快速完成，不用再处理复杂的底层文件格式。

（二）数据库原生的 OS 服务实现

DBOS 的核心服务全都是基于数据库构建的。比如任务调度，就是通过查询“工作者表”找到空闲的计算资源，再往“任务表”里插入任务记录，整个过程就是简单的数据库操作，还能通过修改 SQL 逻辑轻松切换调度策略（比如优先分配给本地资源，或优先分配给空闲资源）。进程间通信（IPC）则通过往“消息表”里插入和读取数据实现，数据库的事务特性能自动保证消息不丢失、不重复，不用额外设计复杂的可靠性机制。文件系统也一样，文件的元数据（比如名称、大小、存储位置）存在数据库表中，查询和管理都比传统文件系统更高效。

（三）事务驱动的容错与调试技术

由于所有操作都基于数据库事务，DBOS 天然具备强大的容错能力。事务的“ACID 特性”能保证：要么操作完全成功，要么失败后回滚到之前的状态，不会出现数据错乱。比如系统升级时如果出现问题，只需将数据库回滚到升级前的状态，系统就能恢复正常。在调试方面，因为所有操作日志都存在表中，开发者可以通过 SQL 查询完整的操作轨迹，轻松定位那些时隐时现的“疑难 bug”，不用再拼接零散的日志文件。

基于三篇目标论文的操作系统未来发展方向理解

结合《FlexOS: Towards Flexible OS Isolation》《LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation》《DBOS: A DBMS-oriented Operating System》三篇论文的核心创新，操作系统未来将围绕“适配多样化需求、贴合硬件新形态、简化系统管理”三大核心方向发展：

一、灵活性

受 FlexOS 启发，未来操作系统将告别“一刀切”的固定架构，转向“延迟决策”模式。通过模块化拆分 OS 功能（如调度、网络、存储组件），用户可在编译 / 部署阶段，按需选择组件隔离方式、数据共享策略及安全硬化手段，无需在设计阶段固定核心规则。

二、分布式

参考 LegoOS 的 splitkernel 架构，未来 OS 将原生支持硬件“分散部署”。CPU、内存、存储等硬件不再绑定为单体服务器，而是独立成网络连接的组件，由专属监控器（进程、内存、存储监控器）本地管理。

三、数据驱动与简化管理

沿 DBOS“一切皆表”的思路，未来 OS 将用分布式数据库替代传统底层数据结构管理系统状态。任务调度、IPC、文件系统等核心服务，均通过数据库表操作实现。这种方式能统一系统状态管理，降低跨组件协作难度，让调试、监控通过简单查询完成，大幅简化 OS 开发与运维。

多模态大模型领域操作系统的未来发展方向

我的研究方向是多模态大模型，具有数据量大，算力要求强等特点，我认为未来的操作系统为了适配这项任务，会朝着如下方向发展

一、异构资源及其协同调度

多模态大模型需 CPU、GPU、存储、高速网络等协同工作，未来 OS 会借鉴模块化理念，将这些硬件抽象为独立资源池，配专属“管理者”精准管控。比如为 GPU 设计算监控器优化显存分配，为多模态数据设处理监控器适配文本、图像等不同需求。同时 OS 会打通与模型框架的接口，获取模型结构、训练进度等信息，动态调整资源分配 —— 像根据数据加载或计算峰值，自动调度空闲 GPU 或优化网络传输，让硬件效能最大化。

二、多模态大数据的高效管理

面对 TB 级甚至 PB 级的异构数据，OS 会优化存储与传输策略：文本用高压缩存

储，图像视频用支持随机访问的块存储，热点数据靠多级缓存加速读取。还会实现内存与显存的统一管理，支持数据预取和异步传输，隐藏传输延迟，同时整合多节点内存形成“大资源池”，解决大模型参数放不下的问题。针对实时推理场景，集成流处理引擎，边接收数据边预处理，直接传至 GPU 计算，减少等待时间。