课程报告：操作系统创新与未来发展方向

叶睿宸

1 引言

操作系统（OS）作为计算机系统的核心软件层，历史上主要基于“单一整机服务器”假设：CPU、内存、存储等硬件资源紧耦合在整机内，操作系统负责管理该整机、用户空间和内核空间隔离、驱动、进程管理、内存管理、文件系统等。随着云计算、数据中心、硬件异构化、资源模块化、边缘计算、安全威胁激增等趋势的兴起，这些传统的 OS 假设正在面临挑战。

本报告基于三篇代表性的研究论文：

1. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation, OSDI 2018
2. DBOS: A DBMS-Oriented Operating System, VLDB 2022
3. FlexOS: Towards Flexible OS Isolation, ASPLOS 2022

通过深入研读这三篇论文，本报告旨在系统总结三篇论文的背景、设计、实现、评价、优缺点；对操作系统未来发展方向进行综合理解；并结合我在分布式系统与云计算研究领域的视角，展望了该领域操作系统的潜在发展路径。

2 论文综述

2.1 LegoOS

随着数据中心规模扩大、硬件异构化、资源利用与弹性需求增强，传统服务器整机（monolithic server）模式逐渐显现瓶颈：资源利用率低（例如内存、存储空闲）、硬件弹性差、故障域大。论文指出，为了改进弹性、资源复用、异构支持、故障隔离，数据中心可以将服务器拆分成“CPU、内存、存储”等模块化、网络可访问的硬件组件（即硬件资源解聚）——但现有操作系统并未为此做好准备。因此，作者提出设计一种新的 OS 模型：Splitkernel，并据此实现了 LegoOS，该系统的关键设计如下：

1. Splitkernel 模型：将传统 OS 的功能（如进程调度、内存管理、存储管理）拆分成多个 “监控器（monitors）”，每个监控器运行于一个硬件组件（例如处理器节点、内存节点、存储节点）上，功能分散但通过高速网络协作。
2. 硬件组件分离：论文中定义了 pComponent（处理器组件）、mComponent（内存组件）、sComponent（存储组件），每类组件上运行专门的监控器。操作系统视为对用户呈现一组“分布式服务器”而非一个整机。
3. 资源分配与故障处理：LegoOS 在资源分配层面负责将内存、存储等模块化组件分配给处理器组件，并在部分硬件失败时做迁移或重试，从而提高弹性和资源利用。
4. 兼容传统接口：尽管底层硬件解耦，LegoOS 努力兼容 Linux 的 syscall 接口，从而减少应用迁移负担。

该篇论文提出颇具前瞻性的 OS 模型，契合未来数据中心硬件模块化、解耦趋势。将 OS 设计与资源解耦硬件协同考虑，从而推动 OS 向数据中心规模与模块化适应。同时证明这种模型在模拟环境下具备可行性。

2.2 DBOS

随着云计算、数据中心规模化、应用态状态膨胀（PB 级数据）、分布式系统普及，传统 OS 架构（单机 OS+集群调度器+分布式文件系统+网络管理）显得日益繁复且难以统一管理。论文指出：为什么不将一个分布式事务型数据库管理系统（DBMS）作为 OS 的基础？因为 DBMS 本身具备高并发、分布式、高可用、状态管理、事务机制等特性。于是提出一种新的 OS 架构：数据库-导向操作系统（DBMS-Oriented OS）。该系统的关键设计如下：

1.将 OS 中的各种状态（进程、通信、任务、文件元数据、调度状态）抽象为数据库表（“everything-is-a-table”）。

2.所有 OS 操作（如调度、文件管理、IPC）通过事务机制访问这些表：状态变迁即事务，查询 / 更新通过数据库。

3.架构上，底层为高性能分布式内存 DBMS（如 VoltDB）加上微内核，上层为 OS 服务。OS 服务运行在数据库之上。

4.系统可横跨多节点或多机集群，从 OS 视角支持“集群资源管理”而不只是单机。

作者对 DBOS 的初步原型进行了评测，覆盖任务调度、文件系统、进程间通信（IPC）等 OS 服务。评测结果显示，基于 DBMS 构建的 OS 在这些服务上的性能可与传统系统（如 Linux+分布式文件系统）接近，同时在状态可分析性、统一管理、代码复杂度方面具备优势。

2.3 FlexOS

传统操作系统的隔离安全策略（如用户内核分离、虚拟机隔离、容器隔离）刻意在 OS 设计阶段就被锁定，而在部署后改变这些隔离机制往往需要重大重构。与此同时，硬件保护机制日益丰富（如 Intel MPK、ARM Morello/CHERI、TEE 等），安全威胁不断出现（如 Meltdown/Spectre），应用对隔离、性能需求多样。论文指出将隔离策略留到部署时选择，从而让 OS 隔离机制更加灵活。并基于此提出了 FlexOS: Towards Flexible OS Isolation。其关键的设计如下：

1.基于 LibOS（库 OS）模型，实现 OS 各组成部分为可组合模块。FlexOS 将这些模块以 fine-grained 组件形式组织，组件之间可根据配置划分到不同“隔离域（compartments）”。

2.用户可在部署时选择每个组件使用何种隔离机制（如 MPK、EPT/VM、CHERI）、以及数据共享策略、软件硬化等级。组件之间通信被视为“call gates +共享数据”机制。

3.提出一种“设计空间探索”机制：系统可评估多个隔离配置（例如论文中对 Redis 做了 80 种配置探索），并基于性能预算选出“最安全的几种”。

FlexOS 的提出为 OS 隔离机制提供了极大的灵活性，支持“隔离策略”从设计时转移到部署时。为 OS 安全研究提供了模块化、可配置、探索设计空间的方法。并且增强了 OS 的适应性：不同应用场景、硬件平台、性能 / 安全要求可裁剪隔离机制。

表 1：三篇论文核心思想对比

特性	LegoOS	DBOS	FlexOS
核心理念	硬件资源解耦与池化	操作系统即数据库服务	可灵活配置的安全隔离
要解决的问题	资源利用率低、扩展不灵活	分布式应用状态管理复杂	安全机制僵化，性能与安全难以兼顾
关键创新	分离式内核、资源池化	所有 OS 状态存入数据库，事务化操作	部署时可配置的隔离机制
架构视角	硬件资源中心化	数据状态中心化	安全信任域中心化

如表 1 所示，这三篇论文虽然分别聚焦不同维度，但它们在整体趋势上具有高度的共通性。三篇都从传统操作系统假设（单机整机、固定硬件资源、本地内存、单一隔离机制）出发，指出这些假设在现代云、数据中心、边缘、异构环境下已弱化。并且共同暗示未来 OS 不再单一维度优化，而是“资源管理 + 状态管理 + 安全隔离”三维并重。总的来看，这三篇工作合起来描绘了操作系统未来可能演进的三个维度：硬件资源结构、软件架构模型、隔离配置。它们相互补充，构成一个较完整的未来 OS 演化蓝图。

3 操作系统未来发展

基于上述三篇论文，以及我对当前硬件和软件趋势的观察，我认为操作系统未来主要将在以下几个方向发展：

3.1 硬件资源解耦

随着网络速率、RDMA、可编程互连的发展，CPU、内存、存储、加速器（GPU、TPU、DPU）可能被模块化、远程访问、组合部署。操作系统需支持对这些分离硬件资源的管理：资源分配、远程内存访问、故障迁移、弹性伸缩、异构支持。LegoOS 是典型代表。在未来 OS 必须假设“内存可能在远程节点”“存储可能网络附加”“加速器可能脱离主机”。因此，OS 的调度、通信、内存管理、故障恢复等都将重新设计。

3.2 状态与服务数据化（数据库化）

在大规模分布式环境中，系统状态（任务、调度、资源、通信、日志）规模巨大、分布广、需要治理与分析。DBOS 提出将 OS 状态作为“表 + 事务”处理。这预示着 OS 将变得更像一个“服务平台”而非单一内核：状态统一管理、事务保证、一致性支持、可查询可分析。未来 OS 可能内建状态数据库、提供查询接口、支持自治监控、可视化运维。

3.3 灵活与自适应隔离

FlexOS 的模块化隔离机制应对了硬件漏洞（如 Meltdown）和新保护机制（如 CHERI）的挑战。未来操作系统需要支持动态配置的隔离策略，以适应多样化的安全和性能需求。虽然隔离（用户内核、容器、虚拟机、TPM/TEE）曾是固定设计，但硬件、软件环境迅速变化。未来 OS 将提供多个隔离机制（MPK、VM/EPT、CHERI、TEE 等）作为模块，可基于应用需求、硬件能力、性能预算进行选择、切换。隔离成为一个设计空间而非固定决策。

3.4 云、边缘、多节点环境作为常态

单机服务器不再唯一运行场景。OS 要跨节点、跨域、跨云、跨边缘协作。其职责从管理单机资源扩展为管理资源池、节点之间通信、迁移、故障恢复、跨节点调度。OS 将更加分布式、可迁移、自治。

3.5 模块化且可配置

操作系统将越来越模块化。不同应用场景（高性能服务、实时系统、边缘设备、IoT、安全关键系统）对 OS 的要求不同。因此，OS 构建时可能按需裁剪、部署时配置、运行时切换。组件化、可升级、可定制成为趋势。

3.6 可观察性、治理、分析能力增强

随着系统规模扩大、异构化加剧、故障复杂化，OS 自身需具备更强的可观察性、审计能力、日志/状态分析、自动诊断。状态数据化有助于此。OS 不再“黑箱”运作，而是可被监控、可重构、可优化的系统平台。

4 AI 大语言模型研究领域的操作系统发展展望

作为 AI 大语言模型领域的研究人员，我认为未来操作系统的发展将深度围绕大模型训练与推理的特化需求展开，具体体现在以下方向：

4.1 面向大模型的异构计算资源调度优化

未来操作系统需成为异构计算的智能调度中枢。大模型训练需要协同调度 CPU、GPU、NPU、DPU 等多元算力，而传统操作系统对 GPU、NPU 的精细化管理能力不足。下一代操作系统应实现：

1. 动态资源切片：将 GPU 显存和计算单元按训练/推理任务动态分区，支持多任务并行执行而不互斥。

2. 跨节点算力池化：通过操作系统级的虚拟化技术，将分布式算力节点抽象为统一资源池，降低分布式训练的资源调度开销。

3. 功耗感知调度：根据模型结构和硬件状态自动优化计算路径，提升能效比（如选择 TensorCore 或 ShaderCore 执行特定算子）。

4.2 端云协同的 AI 原生架构

大模型落地需平衡计算效率与隐私保护，端云协同将成为操作系统的核心能力。

1. 端侧轻量化推理引擎：操作系统需内置模型压缩与加速框架（如支持 INT4/INT8 量化、自适应算子优化），使百亿参数模型可在手机、边缘设备稳定运行。

2. 云边端任务编排：操作系统需支持训练与推理任务的动态分流。例如，敏感数据在端侧处理，非敏感任务卸载至云端，并通过统一接口管理数据流与模型版本。

3. 跨平台模型部署标准：建立类似 ONNX 的开放模型格式标准，使操作系统能无缝适配不同硬件后端（如华为昇腾、寒武纪等国产芯片）。

4.3 智能体（Agent）与原生的操作系统交互范式

操作系统将从“工具”演进为主动服务的智能体：

1. 意图预测与资源预分配：通过分析用户行为序列，操作系统可预加载模型资源、分配算力，降低推理延迟（如统信 UOS 的意图判断模块已优化 AI 任务响应时间）。

2. 自治运维与故障预测：集成 AI 运维智能体，实时检测 GPU 故障、显存泄漏等问题，并实现自修复。

3. 自然语言交互接口：用户可通过自然语言指令直接调度系统资源（如“为训练任务预留 50% GPU 显存”），降低分布式系统管理门槛。

4.4 开源生态与标准化建设

大模型时代的操作系统竞争本质是生态竞争，需通过开源策略打破技术壁垒：

1. 统一适配框架：推动类似龙蜥社区“MCP 适配标准”的接口规范，减少 AI 芯片与操作系统的适配成本。

2. 开源数据集与工具链：操作系统需深度集成 MLOps 工具（如模型版本管理、数据流水线），并通过开源社区吸引开发者贡献优化组件（如 DeepSeek 通过开源快速构建生态）。

3. 国产软硬件全栈协同：操作系统需与国产芯片（如昇腾、龙芯）、框架（如 PaddlePaddle）形成闭环，降低对 CUDA 生态的依赖。

4.5 安全与隐私保护的原生设计

大模型涉及敏感数据，操作系统需提供”硬件级可信执行环境”与”数据全生命周期保护”：

1. 联邦学习与差分隐私支持：在操作系统内核层实现加密计算区，支持模型训练时不暴露原始数据。

2. 模型知识产权保护：通过安全启动、模型水印等技术，防止部署的模型被非法提取或篡改。

5 结语

LegoOS、DBOS 和 FlexOS 展示了操作系统设计在资源解耦、数据中心化和灵活隔离方面的突破性进展，预示着操作系统向模块化、分布式和专用化的未来发展。趋势虽然清晰，但现实仍存在若干挑战，远程资源访问引入延迟、带宽瓶颈、故障域扩大；OS 要兼顾高性能、低延迟、资源节约，不能引入过多抽象层次开销；向状态数据库化迁移可能牺牲实时性或引入事务开销；隔离机制可调虽好，但配置复杂、工具不成熟、开发门槛高等。

抛开可以预见的曲折的未来发展道路，在 AI 大模型领域，这些创新也启发了对解耦硬件、数据驱动管理和安全隔离的深入思考，在该领域操作系统未来所承载的角色将比传统更加关键，它不仅负责管理单机资源，还必须协调异构加速器、显存/内存、网络带宽、跨节点资源；不仅提供任务调度，还要支持大规模状态管理、可分析监控；不仅要隔离与保护，还要灵活可配置；不仅为通用服务提供支撑，还要为大模型这一特殊负载提供定制化支持。

参考文献

[1] Shan Y, Huang Y, Chen Y, et al. {LegoOS}: A disseminated, distributed {OS} for hardware resource disaggregation[C]//13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). 2018: 69-87.

[2] Skiadopoulos A, Li Q, Kraft P, et al. DBOS: A dbms-oriented operating system[J]. 2021.

[3] Lefeuvre H, Bădoiu V A, Jung A, et al. FlexOS: towards flexible OS isolation[C]//Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022: 467-482.