学校编码：10384
学　　号：200215001

厦門大學

博　士　学　位　论　文

基于 eBPF 的异构计算系统下的 serverless 安全容器应用加速技术研究

Capital Reorganization: The Best Choice for A State-Owned Enterprise with Financial Crisis

熊若凡

指导教师姓名：张一鸣教授

专　业　名　称：　计算机技术

论文提交日期：　　年　月

论文答辩日期：　　年　月

学位授予日期：　　年　月

20　年　月

厦门大学学位论文原创性声明

本人呈交的学位论文是本人在导师指导下，独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合法律规范和《厦门大学研究生学术活动规范（试行）》。

另外，该学位论文为（　　）课题（组）的研究成果，获得（　　）课题（组）经费或实验室的资助，在（　　）实验室完成。（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明。）

本人声明该学位论文不存在剽窃、抄袭等学术不端行为，并愿意承担因学术不端行为所带来的一切后果和法律责任。

声明人　（签名）：

指导教师（签名）：

年　　月　　日

厦门大学学位论文著作权使用声明

本人同意厦门大学根据《中华人民共和国学位条例暂行实施办法》等规定保留和使用此学位论文，并向主管部门或其指定机构送交学位论文（包括纸质版和电子版），允许学位论文进入厦门大学图书馆及其数据库被查阅、借阅。本人同意厦门大学将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于：

（　　）1.经厦门大学保密委员会审查核定的涉密学位论文，于　　年　　月　　日解密，解密后适用上述授权。

（　　）2.不涉密，适用上述授权。

（请在以上相应括号内打“√”或填上相应内容。涉密学位论文应是已经厦门大学保密委员会审定过的学位论文，未经厦门大学保密委员会审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权。）

声明人（签名）：

年　　月　　日

学位论文答辩委员会名单

主席	赵 XX	厦门大学	教授	博士生导师
委员	刘 XX	厦门大学	教授	博士生导师
	杨 XX	中国 XXXX 科学院 XXXXXXX 研究所	研究员	博士生导师
	黄 XX	XXXX 大学	教授	博士生导师
	周 XX	XXXX 大学	副教授	硕士生导师
秘书	吴 XX	厦门大学	助理教授	-

答辩时间：2020 年 5 月 10 日上午

答辩地点：厦门大学海韵园行政 C-505 会议室

摘　要

论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。论文摘要的书写应力求精确、简明。切忌写成对论文书写内容进行提要的形式，尤其要避免“第 1 章……；第 2 章……；……”这种或类似的陈述方式

关键词：形状记忆；聚氨酯；织物；合成；应用（一般选 3～8 个单词或专业术语，且中英文关键词必须对应。）

Abstract

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

Keywords: shape memory properties; polyurethane; textile; synthesis; application

目 录

摘 要 I

Abstract II

目 录 III

Table of Contents V

主要缩略词表 VI

主要符号表 VII

图索引 VIII

表索引 IX

第 1 章 绪论 1

1.1 研究背景 1

1.1.1 Serverless 冷启动问题的技术矛盾 1

1.1.2 现有解决方案及其局限性 2

1.2 研究意义与创新点 3

1.2.1 选题意义 3

1.2.2 研究目标和内容 4

1.3 国内外研究现状 5

1.4 本文主要内容和章节安排 6

1.5 Serverless 函数并发启动关键技术点 7

1.6 基于 eBPF 的 DPU 静态虚拟化 8

1.6.1 系统设计与实现 9

1.6.2 核心模块设计 10

1.6.3 关键流程设计 11

第 2 章 总结与展望 13

参考文献 14

附录 A 关于 XXX 的证明 15

附录 B　Maxwell Equations 16

致　谢 17

在学期间完成的相关学术成果 18

Table of Contents

Abstract-Chinese I

Abstract-English II

Table of Contents in Chinese III

Table of Contents in English V

List of Acronyms V

List of Symbols VI

List of Figures VIII

List of Tables IX

Chapter 1 Introduction 1

1.1 Research Background 1

1.2 Motivation 3

1.2 Research objectives and content 4

1.3 Research Progress Overview in Home and Abroad 5

1.4 Major Contents and Chapter Arrangement 6

References 14

Appendix A Proof of XXX 15

Acknowledgements 17

Relevant Academic Achievements Completed During the Academic Period 18

主要缩略词表

缩写	英文全称	中文含义

主要符号表

符号	中文含义

图索引

图 1.1 Overview 4
图 1.2 DPU 应用程序的调配和服务编排 8
图 1.3 eBPF 隔离的 DPU 调配和服务编排 10

表索引

第 1 章　绪论

1.1　研究背景

近年来，随着云计算技术的快速发展，无服务器计算（Serverless Computing）凭借其” 按需分配、自动弹性伸缩、零基础设施运维” 的特性，在微服务、事件驱动型应用、数据处理等领域广泛应用。其核心形态函数即服务（Function as a Service, FaaS）通过细粒度函数部署实现了业务逻辑与资源管理的解耦，首先，它可以帮助应用程序开发人员专注于核心逻辑，并将与基础架构相关的任务（例如自动缩放）留在无服务器平台上。其次，它采用了具有细粒度付费模式（例如 1MS [7]）的 pay-as-you-go 模型，以便用户可以节省未使用的计算资源的成本。第三，无服务器计算也使云提供商有益于他们可以更有效地管理资源。Serverless 的核心思想是将基础设施的管理责任交给云服务提供商，用户无需关心服务器的配置和管理，可以专注于业务逻辑的实现。

然而，在 Serverless 环境中，如何高效、安全地运行容器化应用仍然是一个亟待解决的关键问题。但冷启动延迟问题始终是制约其性能的关键瓶颈，尤其在突发请求和高并发场景下更为显著。

传统 Serverless 平台依赖容器化技术（如 Docker）实现资源隔离与执行环境封装。尽管容器较虚拟机轻量，但其底层依赖的 Linux 控制组（cgroups）机制在资源分配效率上仍存在不足：（1）cgroups 层级结构的资源协商延迟：在高并发场景下，cgroups 的 CPU、内存等子系统的协同配置效率低下，导致函数实例初始化过程中产生显著调度延迟 10；（2）静态资源预配置的局限性：现有策略难以动态适应函数负载特征，导致资源碎片化与利用率下降，这一问题在突发流量场景下被放大，形成性能与资源效率的矛盾 13。

1.1.1　Serverless 冷启动问题的技术矛盾

Serverless 架构的核心优势在于其按需分配资源的特性，即函数实例在无请求时缩容至零，而在请求到来时快速启动以处理任务。然而，这种动态伸缩机制带来了显著的冷启动延迟问题。冷启动是指函数实例从零状态到完全就绪所需的时间，主要包括容器初始化、依赖加载、代码执行环境准备等阶段。研究表明，冷启动延迟通常在数百毫秒到数秒之间，这对于实时性要求较高的应用场景（如在线交易、

实时数据处理）是不可接受的 716。

冷启动问题的根源在于 Serverless 平台依赖的容器化技术（如 Docker）。尽管容器较虚拟机轻量，但其底层依赖的 Linux 控制组（cgroups）机制在资源分配效率上仍存在不足：（1）cgroups 层级结构的资源协商延迟：在高并发场景下，cgroups 的 CPU、内存等子系统的协同配置效率低下，导致函数实例初始化过程中产生显著调度延迟；（2）静态资源预配置的局限性：现有策略难以动态适应函数负载特征，导致资源碎片化与利用率下降，这一问题在突发流量场景下被放大，形成性能与资源效率的矛盾 716。

另一方面，现有安全隔离方案（如容器运行时、微虚拟机）虽提供强隔离性，但其引入的上下文切换、内核拦截等开销与 Serverless 函数的短生命周期特性冲突。近期研究表明，基于 **eBPF（Extended Berkeley Packet Filter）** 的轻量级内核可编程技术成为突破传统虚拟化架构的新方向 612。eBPF 通过沙盒化机制允许用户态程序安全执行内核级操作（如网络过滤、系统调用追踪），为构建” 零虚拟化开销” 的隔离机制提供了可能。例如，UCloud 的 Serverless 容器产品 Cube 通过 eBPF 实现高效服务发现，替代 kube-proxy，降低网络延迟 6；阿里云 SAE 则利用 eBPF 实现多语言无侵入式监控，减少数据在内核态与用户态间的拷贝开销 12。

值得注意的是，近年来国产 DPU（专用数据处理器）在数据中心网络加速领域取得显著进展。例如，云豹智能推出的 400Gbps DPU 芯片通过卸载网络与存储任务提升了算力集群效率，但其硬件级隔离能力仍存在不足。现有国产 DPU 多聚焦于网络带宽与计算卸载性能优化，而在多租户场景下，其基于传统虚拟化（如 SR-IOV）或软件定义网络（SDN）的隔离机制难以实现细粒度的安全边界，导致租户间资源争用与潜在的安全风险 412。这一问题在 Serverless 场景中尤为突出——函数实例的高密度部署需要硬件与软件协同的轻量级隔离机制，而传统 DPU 的隔离方案无法满足此类需求。

1.1.2 现有解决方案及其局限性

为缓解冷启动问题，业界提出了多种优化方案，主要包括：

预留实例：通过预启动一定数量的函数实例以减少冷启动时间，但这种方法削弱了 Serverless 按需付费的成本优势，且无法完全消除冷启动延迟 16。

容器快照技术：利用 CRIU（Checkpoint/Restore in Userspace）等技术捕获容器状态快照，在请求到来时快速恢复容器状态。然而，这种方法对内存和文件系统

的依赖较大，且难以完全兼容现有 Serverless 平台 16。

Init-Less 模式：通过将用户代码改写为无初始化依赖的模式，结合延迟恢复策略（Lazy-Restore）减少冷启动时间。这种方法虽有效，但需要对用户代码进行较大改造，增加了开发复杂度 16。

尽管上述方案在一定程度上缓解了冷启动问题，但仍存在以下局限性：

资源利用率与性能的权衡：预留实例和容器快照技术虽能减少冷启动时间，但增加了资源占用，降低了 Serverless 的成本优势 716。

兼容性与开发复杂度：现有优化方案往往需要对底层容器技术或用户代码进行改造，增加了技术复杂度和迁移成本 16。

1.2　研究意义与创新点

针对上述挑战，本研究提出双路径优化方案：

cgroups 资源调度优化：通过重构 cgroups 层级结构与动态调度算法，提升高并发场景下函数实例的启动吞吐量。此方向可借鉴华为云 FunctionGraph 的 Pod 池化技术与镜像快速分发策略，结合资源本地化缓存减少冷启动时间 1016。

基于 eBPF 的静态程序分析框架：通过静态分析函数行为特征，预构建最小特权执行上下文，规避传统虚拟化开销。该方案需结合 eBPF 在协议解析（如 HTTP/MySQL）与系统调用追踪（accept/read/write）中的高效性，实现安全性与性能的平衡 59。

两项工作的协同将推动 Serverless 平台向高效性、安全性与资源弹性兼备的架构演进，为云原生应用的性能优化提供新范式。

1.2.1　选题意义

本研究的主要目标是提升 Serverless 环境中容器应用的安全性和性能，尤其是在异构计算平台下。研究的意义可从以下几个方面进行阐述：

优化 cgroup 机制提升并发启动性能随着 Serverless 架构的广泛应用，容器化的应用需要在高度动态和弹性的环境中快速启动和运行。Linux 内核的 cgroup 机制在容器的资源管理和隔离方面发挥了重要作用，但在面对高并发容器启动时，现有 cgroup 的性能和可扩展性仍然不足。通过对 cgroup 机制进行优化，可以有效提升 Serverless 平台中容器的启动效率，增强平台在高并发环境下的稳定性和扩展能力，满足现代云计算平台对高性能和高并发的需求。

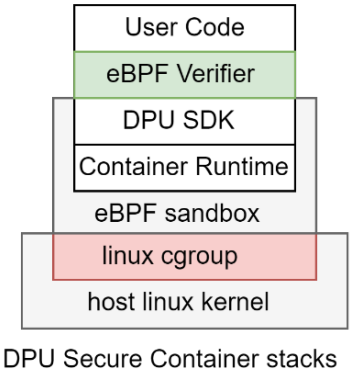

图 1.1 Overview

保证 DPU 资源的安全隔离在 Serverless 环境中，DPU 作为一种硬件加速单元，可以显著提升计算效率。然而，DPU 作为共享资源在多租户环境中的使用，容易面临安全性和资源隔离的问题。如何保证不同租户在使用 DPU 时的资源隔离性，防止不同租户之间相互干扰，成为了一个亟待解决的难题。本研究通过引入类似 eBPF verifier 的静态程序验证技术，能够在运行时前对 DPU 访问代码进行验证，确保多租户环境下对 DPU 的安全访问，从而避免内存泄露、数据篡改等安全风险。

推动 Serverless 平台在异构计算环境中的应用发展目前，Serverless 架构的研究主要集中在通用的计算资源和虚拟化技术上，针对异构计算资源（如 DPU、GPU 等）的支持尚处于起步阶段。本研究不仅将研究如何在 Serverless 架构中高效、安全地使用 DPU 加速计算，还将探索如何使 Serverless 平台更好地适应异构计算环境，为异构计算资源的全面融合提供理论依据和技术支持。

促进 Serverless 技术的商业应用和发展随着云计算技术的不断发展，Serverless 架构在互联网企业和各类业务应用中的应用越来越广泛。通过提升 Serverless 平台在高并发、大规模容器管理和异构计算资源安全管理方面的能力，本研究的成果不仅具有重要的学术价值，同时对促进 Serverless 技术在商业应用中的推广和发展具有重要意义。

1.2.2 研究目标和内容

本论文的主要研究目标是通过优化 Linux 内核 cgroup 机制，提高 Serverless 环境下容器的并发启动性能；通过引入静态程序验证技术，保证 Serverless 平台中 DPU 资源的多租户安全隔离。具体内容包括：

优化 cgroup 机制：研究并优化 Linux cgroup 在高并发、大规模容器启动场景

中的性能瓶颈，提高其在 Serverless 环境中的可扩展性和响应速度。eBPF 验证技术的应用：通过引入 eBPF verifier 等静态验证技术，确保 Serverless 平台中异构计算资源（如 DPU）的安全隔离，避免不同租户间的资源竞争和安全问题。通过以上研究，本论文旨在为异构计算平台下的 Serverless 架构提供高效、安全的容器应用加速技术，为下一代云计算平台的发展做出贡献。

1.3　国内外研究现状

国外学术界与工业界针对 Serverless 冷启动问题已展开多方面探索，主要集中在容器技术优化、虚拟化层轻量化与资源调度创新三个方向：

容器与镜像加速：微软提出基于并行化镜像构建的 FAST-BUILD 技术，通过文件级缓存复用机制减少容器构建时间 2；AWS 采用分布式多级缓存的镜像分发策略，结合 Block 存储优化，缩短镜像拉取时延 2。此外，MIT 团队开发的 RemoteFork 利用 RDMA 共享内存技术，将跨节点实例启动的数据拷贝延迟降至微秒级 3。轻量级虚拟化：Google 的 gVisor 通过用户态内核模拟实现安全容器隔离，启动时间压缩至毫秒级 2；斯坦福 Catalyzer 项目针对安全容器进行深度裁剪，结合预加载机制进一步降低初始化开销 2。非容器运行时技术：剑桥大学提出基于 WebAssembly（WASM）的 Faasm 框架，以进程级隔离替代容器虚拟化，在内存安全性与启动速度上取得平衡 3。尽管上述技术显著优化了冷启动的部分环节，但仍存在以下局限：

cgroup 并发瓶颈未解：现有研究多聚焦于容器运行时优化，但 cgroup 在高并发场景下的资源分配效率问题仍缺乏针对性设计 2；虚拟化层冗余开销依存：即使轻量级容器（如 Kata Containers）启动时间已优化至亚毫秒级，其内核加载与资源分配仍占用总延迟的 30% 以上 3。2. 国内研究动态国内企业与研究机构在 Serverless 冷启动优化中侧重于镜像本地化与资源预配置：

镜像快速分发：阿里云提出基于树结构的跨节点镜像分发方案 FaasNet，通过并行传输与增量更新技术将镜像分发效率提升 40% 2；华为云 FunctionGraph 采用 Pod 池化技术，通过预创建特化实例跳过镜像传输阶段，实现毫秒级实例启动 2 4。运行时环境加速：字节跳动 ByteFaaS 平台引入本地代码缓存机制，结合动态链接库预加载技术，将 Python 函数启动时间降低 33% 3；华为云通过共享内存加速与依赖包预解压技术，将冷启动延迟压缩至 10ms 以内 4。智能资源调度：腾讯云提出基于负载预测的动态缓存算法，通过机器学习模型预估实例需求，冷启动发生率降低 60% 3。然而，国内研究仍面临挑战：

cgroup 性能优化不足：现有方案依赖资源池化缓解压力，但未解决 cgroup 自身在高并发创建时的锁竞争与调度效率问题 2；静态程序分析缺位：eBPF 技术多用于网络性能优化（如 Cilium），尚未与函数代码特征分析结合以替代虚拟化层 4。3. 研究空白与突破方向当前国内外研究呈现两大共性局限：

cgroup 并发控制机制僵化：传统 cgroup 框架的任务调度策略难以应对突发性高密度函数实例创建需求，导致资源分配成为冷启动链路的性能瓶颈 2 3；虚拟化层与函数逻辑割裂：现有优化多从“加速容器启动”角度切入，但未深入挖掘函数代码特性以绕过虚拟化层 3 4。针对上述问题，本研究提出：

cgroup 动态批量操作机制：借鉴无锁队列与异步资源分配策略，优化高并发场景下的控制平面性能（如单节点支持 500+ 实例/秒的创建速率）2；eBPF 驱动的函数静态分析：通过提取函数系统调用依赖与内存访问模式，预构建轻量化运行环境，实现无虚拟化层的“按需隔离”3 4。这一方向突破了传统优化路径，为 Serverless 冷启动问题提供了底层资源管理革新与函数逻辑感知的双重解决方案，有望推动边缘计算与实时 AI 推理等场景的规模化应用。

1.4 本文主要内容和章节安排

1.5　Serverless 函数并发启动关键技术点

在本节中，我们分析了容器运行时在容器启动期间执行的与 cgroup 相关的操作，以及这些操作在 Linux 内核中的实现。我们以 crun 为例，这些分析同样适用于其他流行的容器运行时，如 runc。这些操作如算法 1 所示，大致可以分为三个步骤。第一步是在第 2 至 6 行启用指定的控制器并创建 cgroup。第二步是在第 8 行将子进程生成到已创建的 cgroup 中。最后一步是在第 10 行将资源限制值写入 cgroup 目录下相应的控制器接口文件中。

具体来说，Linux 内核中 cgroup 创建的实现如第 18 至 25 行所示。首先是分配并初始化一个 cgroup 结构体，并在 cgroupfs 的指定路径下创建一个新目录。接下来是在 cgroup 目录下创建 cgroup 核心文件。然后是为启用的控制器结构体分配内存，最后一步是创建相应的控制器接口文件。

将一个进程附加到 cgroup 可以通过将进程 ID 写入 cgroup 目录下的 cgroup.procs 文件来完成。为了在创建时将新的子进程附加到指定的 cgroup，Linux 提供了 clone3 系统调用。

将进程移动到 cgroup 的过程需要查找或创建一个 css_set，对控制器执行 can_fork 检查以确保资源限制不被超出，最后将进程附加到控制器。

删除 cgroup 与创建 cgroup 相反。它涉及解除 cgroup 控制器的链接，将控制器离线，并销毁控制器。在 cgroup 删除过程中，锁的持有时间要短得多，因为在持有锁时只需使控制器不再被引用。其余耗时的操作是异步且无锁地完成的。因此，与创建相比，cgroup 删除对 cgroup 可扩展性的影响较小。

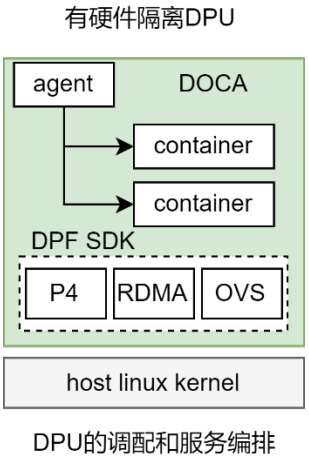

图 1.2 DPU 应用程序的调配和服务编排

1.6 基于 eBPF 的 DPU 静态虚拟化

云计算技术对异构需求越来越高，传统架构存在着处理能力与数据量增长不匹配、资源利用不足、安全风险等问题。DPU 功能 serverless 化变得非常常见。例如英伟达推出的 DOCA 平台框架（DPF）是一个 DPU 程序编排框架，帮助创建 BlueField 加速的云软件平台，使得 DPU 可以在 K8s 环境中被直接使用。同时已有研究人员开始研究异构计算系统（CPU-GPU-DPU）上的 serverless 系统（Serverless Computing on Heterogeneous Computers ASPLOS’22）。

DPU 容器（DPU container）通常是指基于 DPU（数据处理单元）技术，使用容器化技术来部署和管理的计算环境。简单来说，它结合了 DPU 的硬件加速能力和容器的灵活性，通常用于高性能计算、网络加速、存储加速等场景。Nvidia DPU 具有硬件级隔离：NVIDIA BlueField DPU 通过硬件加速引擎和专用处理资源，为每个租户提供独立的计算、存储和网络资源，确保租户之间的资源隔离。在云原生计算平台中，DPU 通过卸载和加速基础设施功能，支持多租户环境下的高性能计算。DOCA 编程框架（DPF）可以实现多租户的 DPU 服务的调配和编排，通过容器化的方式将 Kubernetes 控制平面功能扩展到 DPU，使管理员能够直接在 BlueField DPU 上部署和卸载 NVIDIA DOCA 服务和基于 DOCA 的第三方服务。DPF 配备了专门用于无缝集成的 SDK，提供了一个一致的模块化工具包，例如 OVS，RDMA 开发工具包。

目前国内的 DPU（数据处理单元）在硬件级隔离方面的能力相对有限，要实现完全的硬件级隔离（特别是在安全性、可靠性、性能等方面的完全隔离）仍需要进一步的技术发展和硬件支持，尤其是是在 K8s 等云应用场景下。

1. 虚拟化和资源隔离：许多 DPU 提供了硬件加速的虚拟化支持，尤其是在网络、存储和计算资源的隔离方面。例如，DPUs 如浪潮的“云脉”DPU 或者华为的“昇腾”DPU，已经具备一定的隔离能力，能够在同一物理硬件上支持多个虚拟机或容器的运行，并为其提供网络加速、存储加速等服务。但这种隔离更多是针对资源访问的虚拟化层面的，而非完全的硬件隔离。

2. 安全性与可信计算：要实现完全的硬件级隔离，还需要依赖于更强的安全特性，比如硬件级的加密、完整性保护、物理隔离等机制。目前一些 DPU 已经支持基于硬件的安全功能（如加密引擎），但对于极高要求的安全场景（如政府、金融等领域），硬件级的完全隔离通常需要结合特殊的硬件设计和安全协议，甚至依赖于可信执行环境（TEE）技术。

3.DPU 与 CPU 的协同工作：目前的 DPU 在硬件级的完全隔离方面，可能仍然受到 CPU 和内存共享、I/O 访问控制等因素的制约。尤其是在一些复杂的并发计算和高性能计算场景下，虽然 DPU 可以通过硬件支持高效的网络处理和计算加速，但要完全独立于 CPU 的安全和资源调度依然是一个技术难点。

目前国内的这种不具备完全硬件隔离能力的 DPU 无法支持租户之间的资源隔离要求，阻碍了 DPU 容器化的应用。ebpf 借助 ebpf verfier 实现了一个安全的沙箱，可以防止 ebpf 代码以多种方式来损害内核（DOS 攻击、信息窃取攻击等）。Verifier 可以在一定程度上对容器内的代码实现安全性隔离（todo：需要找一些 case 分析）。同时，eBPF verifier 运行在程序加载阶段，不会给函数容器启动添加额外影响。通过 ebpf verifier 实现 ebpf sandbox 可以在软件层面对用户代码实现安全保障作用。加入 ebpf sandbox 后的 DPU 容器化模型如下图所示：

1. 硬件虚拟化与资源分区 SR-IOV（单根 I/O 虚拟化）DPU 的物理网卡通过 SR-IOV 虚拟化为多个虚拟功能（VF），每个租户的虚拟机（VM）或容器独占一个 VF，实现网络流量的直接硬件隔离，减少延迟并提升吞吐量。

硬件资源分片 DPU 的计算核心（如 Arm CPU、加速引擎）和内存资源通过硬件分区分片，为不同租户分配独立的计算单元和内存空间，防止资源争用。

1.6.1　系统设计与实现

1.6.1.1　总体架构设计

本方案采用分层架构（如图 X-1 所示），核心模块包括静态验证器、白名单管理器、资源隔离引擎和回退控制器。系统工作流程如下：

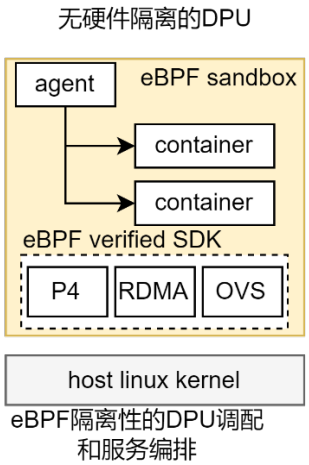

图 1.3 eBPF 隔离的 DPU 调配和服务编排

1. 用户提交函数代码后，触发静态验证器进行内存安全与行为合规性检查。2. 验证通过的函数以普通进程形式运行，由资源隔离引擎限制其权限与资源使用。3. 验证失败的函数触发回退控制器，启动容器实例执行，同时记录失败原因供后续优化。4. 白名单管理器动态维护安全函数列表，支持按需扩展与权限分级。

1.6.2 核心模块设计

1.6.2.1 静态验证器

目标：确保函数代码无内存越界、非法控制流与未授权系统调用。实现方案：

中间表示（IR）生成：基于 LLVM 将用户代码转换为 IR，消除语言特性差异（如指针运算）。

安全规则检查：

内存安全：验证所有内存访问边界（如数组索引、缓冲区大小）。

控制流完整性：禁止跳转至动态计算地址（如函数指针需静态绑定）。

系统调用过滤：通过符号执行检测代码是否仅调用白名单函数。

形式化验证增强：集成 Z3 求解器，对循环与分支路径进行数学证明。约束：强制使用预定义内存池（禁止 malloc/free），支持 Rust 等内存安全语言优先通过验证。

1.6.2.2 白名单管理器

目标：提供最小化、场景化的安全函数集合。实现方案：

分层白名单：

基础类：数学运算、字符串处理（如 memcpy 需带长度参数）。

I/O 类：受限文件读写（仅限临时目录）、网络请求（过滤目标 IP）。

平台服务类：访问数据库、消息队列的 SDK 封装。

动态审核机制：

社区投票新增高危函数（如 execve），默认隔离至 WebAssembly 沙盒执行。

自动监控函数使用频率与安全事件，动态降级或移除风险项。

1.6.2.3　资源隔离引擎

目标：在普通进程模式下实现近似容器的隔离性。实现方案：

轻量级隔离：

命名空间（Namespace）：隔离进程视图（PID、网络、文件系统）。

cgroups：限制 CPU、内存、磁盘 IO 配额。

安全加固：

seccomp 策略：仅允许白名单内的系统调用（如禁止 ptrace）。

SELinux 上下文：为进程分配最小权限标签（如禁止写入/proc）。

硬件辅助：可选启用 Intel SGX 加密敏感数据内存区域。

1.6.3　关键流程设计

1.6.3.1　函数提交与验证流程

用户上传函数代码及依赖声明。

静态验证器执行多阶段检查：

阶段 1（语法分析）：检测禁用语言特性（如内联汇编）。

阶段 2（符号执行）：遍历所有代码路径，标记潜在越界访问。

阶段 3（形式化证明）：对不确定路径调用 Z3 验证安全性。

生成验证报告，通过则编译为可执行文件，否则触发回退流程。

1.6.3.2　进程/容器混合调度流程

资源调度器根据验证结果分配执行环境：

进程模式：直接调用 fork() 启动，附加 cgroup、seccomp 策略。

容器模式：从预热池选取镜像，通过 runc 启动实例。

资源监控器实时收集 CPU、内存指标，超限时强制终止进程或扩容容器。

第 2 章　总结与展望

本文采用……。（结论作为学位论文正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。）

参考文献

附录 A　关于 XXX 的证明

附录相关内容…

附录 B　Maxwell Equations

因为在柱坐标系下，$\overline{\overline{\mu}}$ 是对角的，所以 Maxwell 方程组中电场 **E** 的旋度

所以 **H** 的各个分量可以写为：

$$H_r = \frac{1}{\mathbf{i}\omega\mu_r}\frac{1}{r}\frac{\partial E_z}{\partial\theta} \tag{B-1a}$$

$$H_\theta = -\frac{1}{\mathbf{i}\omega\mu_\theta}\frac{\partial E_z}{\partial r} \tag{B-1b}$$

同样地，在柱坐标系下，$\overline{\overline{\varepsilon}}$ 是对角的，所以 Maxwell 方程组中磁场 **H** 的旋度

$$\nabla\times\mathbf{H} = -\mathbf{i}\omega\mathbf{D} \tag{B-2a}$$

$$\left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta}\right]\hat{\mathbf{z}} = -\mathbf{i}\omega\overline{\overline{\varepsilon}}\mathbf{E} = -\mathbf{i}\omega\varepsilon_z E_z\hat{\mathbf{z}} \tag{B-2b}$$

$$\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta} = -\mathbf{i}\omega\varepsilon_z E_z \tag{B-2c}$$

由此我们可以得到关于 E_z 的波函数方程：

$$\frac{1}{\mu_\theta\varepsilon_z}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial E_z}{\partial r}\right) + \frac{1}{\mu_r\varepsilon_z}\frac{1}{r^2}\frac{\partial^2 E_z}{\partial\theta^2} + \omega^2 E_z = 0 \tag{B-3}$$

致　谢

本论文的工作是在导师……。

在学期间完成的相关学术成果

学术论文：

[1] 高凌. 交联型与线形水性聚氨酯的形状记忆性能比较 [J]. 化工进展, 2006, 532 － 535.（核心期刊）

[2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究 [J]. 中国机械工程, 2005, 16(14):1289-1291.

专利：

[3] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A[P]. 2005-03-30.

[4] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102[P]. (美国发明专利申请号.)