软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(9):3312−3330 [doi: 10.13328/j.cnki.jos.006389]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

S3ML: 一种安全的机器学习推理服务系统*

马俊明[1,3], 吴秉哲[2], 余超凡[4], 周爱辉[4], 巫锡斌[4], 陈向群[1,2,3]

[1](北京大学 软件与微电子学院, 北京 102600)

[2](北京大学 信息科学技术学院, 北京 100871)

[3](高可信软件技术教育部重点实验室 (北京大学), 北京 100871)

[4](蚂蚁集团, 浙江 杭州 310013)

通信作者: 陈向群, E-mail: cherry@pku.edu.cn

摘　要: 隐私保护问题在当今机器学习领域日益受到关注, 构建具备数据安全保障的机器学习服务系统变得越来越重要. 与此同时, 以英特尔 SGX 为代表的可信执行环境技术得到了日益广泛的使用来开发可信应用和系统. SGX 为开发者提供了基于硬件的名为飞地的安全容器来保障应用程序的机密性和完整性. 本文基于 SGX 提出了一种面向机器学习推理的安全服务系统 S3ML. S3ML 将机器学习模型运行在 SGX 飞地中以保护用户隐私. 为了构建一个实用的基于 SGX 的安全服务系统, S3ML 解决了来自两方面的挑战. 首先, 机器学习推理服务为了保证高可用性和可扩展性, 通常包含多个后端模型服务器实例. 当这些实例在 SGX 飞地内运行时, 需要新的系统架构和协议来同步证书及密钥, 以构建安全的分布式飞地集群. S3ML 设计了基于 SGX 认证机制的飞地配置服务, 来专门负责在客户端和模型服务器实例之间生成、持久化和分发证书及密钥. 这样 S3ML 可以复用现有的基础设施来对服务进行透明的负载均衡和故障转移, 以确保服务的高可用性和可扩展性. 其次, SGX 飞地运行在一个名为飞地页面缓存 (EPC) 的特殊内存区域, 该区域的大小有限, 由主机上的所有 SGX 飞地竞争, 运行在飞地中应用的性能因此易受到干扰. 为了满足机器学习推理服务的服务级别目标, 一方面 S3ML 使用轻量级的机器学习框架和模型来构建模型服务器以减少 EPC 消耗. 另一方面, 通过实验发现了使用 EPC 页交换吞吐量作为保障服务级别目标的间接监控指标是可行的. 基于该发现, S3ML 提出基于 EPC 页交换强度来控制服务的负载均衡和水平扩展活动. 基于 Kubernetes、TensorFlow Lite 和 Occlum 实现了 S3ML, 并在一系列模型上进行实验, 对 S3ML 的系统开销、可行性和有效性进行了评估.

关键词: 机器学习推理; 服务系统; SGX; 可信计算; 隐私保护

中图法分类号: TP311

中文引用格式: 马俊明, 吴秉哲, 余超凡, 周爱辉, 巫锡斌, 陈向群. S3ML: 一种安全的机器学习推理服务系统. 软件学报, 2022, 33(9): 3312–3330. http://www.jos.org.cn/1000-9825/6389.htm

英文引用格式: Ma JM, Wu BZ, Yu CF, Zhou AH, Wu XB, Chen XQ. S3ML: Secure Serving System for Machine Learning Inference. Ruan Jian Xue Bao/Journal of Software, 2022, 33(9): 3312–3330 (in Chinese). http://www.jos.org.cn/1000-9825/6389.htm

S3ML: Secure Serving System for Machine Learning Inference

MA Jun-Ming[1,3], WU Bing-Zhe[2], YU Chao-Fan[4], ZHOU Ai-Hui[4], WU Xi-Bin[4], CHEN Xiang-Qun[1,2,3]

[1](School of Software and Microelectronics, Peking University, Beijing 102600, China)

[2](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[3](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

[4](Ant Group, Hangzhou 310013, China)

* 基金项目: 国家重点研发计划 (2017YFE0123600)

收稿时间: 2021-03-23; 修改时间: 2021-04-29; 采用时间: 2021-05-30; jos 在线出版时间: 2022-06-15

Abstract: As the privacy-preserving problem gains increasing concerns in today's machine learning (ML) world, constructing an ML serving system with a data security guarantee becomes very important. Meanwhile, trusted execution environments (e.g., Intel SGX) have been widely used for developing trusted applications and systems. For instance, Intel SGX offers developers hardware-based secure containers (i.e., enclaves) to guarantee application confidentiality and integrity. This paper presents S3ML, an SGX-based secure serving system for ML inference. S3ML leverages Intel SGX to host ML models for users' privacy protection. To build a practical secure serving system, S3ML addresses several challenges to run model servers inside SGX enclaves. In order to ensure availability and scalability, a frontend ML inference service typically consists of many backend model server instances. When these instances are running inside SGX enclaves, new system architectures and protocols are in need to synchronize cryptographic certificates and keys to construct distributed secure enclave clusters. A dedicated module is designed, it is called attestation-based enclave configuration service in S3ML, responsible for generating, persisting, and distributing certificates and keys among clients and model server instances. The existing infrastructure can then be reused to do transparent load balancing and failover to ensure service high-availability and scalability. Besides, SGX enclaves rely on a special memory region called the enclave page cache (EPC), which has a limited size and is contended by a host's all enclaves. Therefore, the performance of SGX-based applications is vulnerable to EPC interferences. To satisfy the service-level objective (SLO) of ML inference services, S3ML first integrates lightweight ML framework/models to reduce EPC consumption. Furthermore, through offline analysis, it is found feasible to use EPC paging throughput as indirect monitoring metric to satisfy SLO. Based on this result, S3ML uses real-time EPC paging information to control service load balancing and scaling activities for SLO satisfaction. S3ML has been implemented based on Kubernetes, TensorFlow Lite, and Occlum. The system overhead, feasibility, and effectiveness of S3ML are demonstrated through extensive experiments on a series of popular ML models.

Key words: machine learning inference; serving system; SGX; trusted computing; privacy-preserving

机器学习近年来在图像分类、自然语言处理等诸多领域的成功, 带动了部署在数据中心中的机器学习推理服务数量的快速增长[1,2]. 在构建机器学习推理服务的过程中, 服务提供商首先使用大数据集离线训练机器学习模型, 然后将服务部署在云端, 进而为用户提供实时服务. 机器学习推理服务通常以 HTTP 或 RPC (远程过程调用) 接口的形式暴露给用户使用. 与其他在线服务一样, 延迟是服务提供商最关心的性能指标. 开发者通常通过服务级别目标 (service-level objective, SLO) 来量化地约束服务延迟. 典型的服务级别目标通常以尾延迟的形式来定义, 例如 99% 的请求延迟小于 500 ms[3].

服务级别目标的满足依赖于一个高可用、可扩展的底层服务系统, 其通常采用如图 1 所示的分布式系统架构. 在该图中, 服务前端接收来自用户客户端并发的请求, 并将这些请求分发到并行的后端模型服务器实例上. 每个实例都装载预先训练好的机器学习模型, 并在隔离的资源环境 (如 Docker 容器[4]) 中运行. 服务系统根据工作负载水平实时动态地调整实例数量, 以满足服务级别目标的要求[3]. 近年来, 面向机器学习推理的服务系统受到了越来越多研究者的关注[3,5,6]. 现有的这些工作关注于支持异构的多模型框架、降低服务运维成本、减少服务响应时间等目标. 与它们不同的是, 本文提出了一种安全的机器学习推理服务系统 S3ML (a secure serving system for machine learning inference), 旨在保护用户在使用推理服务时的隐私安全.

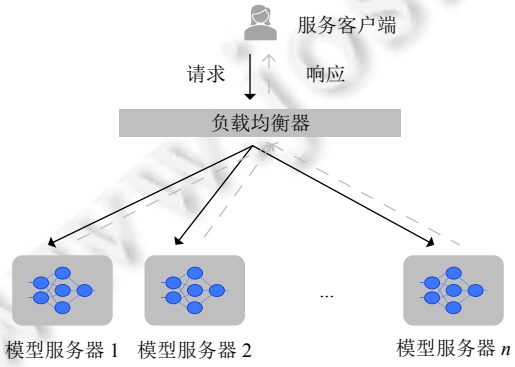

图 1 机器学习推理服务系统的简化系统架构

随着国内外监管环境对于个人隐私保护的日益重视[7], 机器学习服务系统的安全性受到越来越多研究者的关注. 特别是在当前, 众多机器学习服务都部署在公有云或私有云等公共基础设施之上. 用户需要将私有数据上传到云数据中心, 才能从服务提供商处获得推理结果. 但是, 在使用服务的过程中, 不能保证用户的数据不会被服务提供商、基础设施提供商或意外的恶意攻击者窃取或滥用. 因此, 设计并实现一种能够满足用户隐私保护需求的服务系统是至关重要, 特别是对于数据隐私十分重要的应用场景 (例如医疗与金融数据分析)[8].

可信执行环境 (trusted execution environment, TEE) 是解决服务使用者与提供者之间信任问题的一种主流方案. 英特尔软件保护扩展 (Intel software guard extensions, SGX)[9]是目前在服务器平台上被广泛使用的可信执行环境技术. 该技术用于保护运行在不受信任的服务器平台上的程序的机密性和完整性, 由英特尔于 2015 年在 Skylake 系列的 CPU 中首次推出. SGX 提供了名为飞地 (enclave) 的基于硬件的安全容器, 用于托管用户代码和数据, 防止其被飞地外不受信任的特权进程 (如虚拟机监视器和操作系统) 窃取或篡改. S3ML 基于 SGX 为用户在云平台上使用机器学习推理服务提供安全保证以及隐私保护.

但是, 直接使用 SGX 构建机器学习服务系统会导致服务性能的下降, 因此构建满足特定服务级别目标的安全机器学习服务仍然面临一些挑战. 第一个挑战来自于如何灵活地构建高可用、可扩展的飞地集群以保证服务的可靠性. 在使用 SGX 时, 为了保护通信中的秘密数据, 输入需要在客户端加密, 并传输到飞地内进行解密和计算. 因此, 客户端与飞地之间通常需要执行密钥交换协议 (例如 Diffie-Hellman 密钥交换协议[10]) 来建立安全信道. 对于服务系统而言, 这意味着客户端必须与每个模型服务器实例执行密钥交换协议, 这个过程会十分复杂. 此外, 这样做会导致每个后端实例生成自己独有的密钥. 在客户端加密的请求只能由持有相应解密密钥的实例来处理. 因此不能复用现有的软件设施 (例如负载均衡器) 对应用进行透明的负载均衡和故障转移. S3ML 提出通过构建基于 SGX 认证机制的飞地配置服务用于专门负责证书与密钥的生成、保存和分发. 该服务解耦了客户端与模型服务器实例之间的交互, 使二者通过与该服务交互来独立地同步证书和密钥. S3ML 提出了新的系统架构和协议设计, 以确保飞地配置服务本身也具备安全性和高可用性. 开发者能够通过该服务灵活地组建高可用、可扩展的飞地集群.

第 2 个挑战在于如何在 SGX 硬件限制下保证服务级别目标满足要求. 在现代数据中心中, 集群管理系统会将离线的批处理作业和在线的延迟敏感型服务协同调度以提高集群资源利用率[11]. 现有的服务系统[3,5,6]通常以 Docker 容器来封装和运行后端模型服务器. Docker 利用 Linux 上的 cgroups 机制[12]来限制容器内进程组能够使用到的硬件资源 (包括 CPU、内存、I/O 等) 配额, 从而减轻容器间的性能干扰. 对于不依赖 SGX 技术的服务, 给定一个 Docker 容器能够使用的资源配额, 可通过对模型服务器进行离线性能分析, 来预估其在生产环境运行时的性能. 但对于依赖 SGX 技术的服务而言, 其还需要一种被称之为飞地页面缓存 (enclave page cache, EPC) 的特殊类型资源. EPC 十分稀缺, 在 Linux 操作系统上可以通过页交换机制来实现超额分配. Taassori 等人的工作表明[13], EPC 页交换触发会导致应用程序出现显著的性能下降. 由于 EPC 被一台机器上所有的飞地竞争, EPC 页交换很容易被触发. 对机器学习推理服务而言, 这意味着延迟降低, 服务级别目标违反的可能性变高. 但现有的操作系统缺乏稳定和成熟的类 cgroups 机制来隔离不同容器对 EPC 的使用, 从而减轻性能干扰. 为了克服这个挑战, 一方面 S3ML 使用轻量级的机器学习框架和模型来构建模型服务器以减少 EPC 消耗. 另一方面, S3ML 提出了一种方法来定量描述 EPC 页交换吞吐量和飞地托管服务延迟之间的关系, 并进一步地使用这种关系来控制负载均衡 (load balancing) 和水平扩展 (horizontal scaling) 活动, 从而在存在 EPC 资源争用的情形下满足服务级别目标.

本文基于 Kubernetes[14]、TensorFlow Lite[15]以及 Occlum[16]实现了 S3ML 的原型系统, 并在一系列常用机器学习模型上进行实验来评估 S3ML 系统的性能. 本文的核心贡献如下.

(1) 本文基于 SGX 可信计算技术提出了一种安全的机器学习推理服务系统用于构建具备隐私保护特性的机器学习云端服务.

(2) 本文提出了新的系统架构和协议来进行证书与密钥管理, 从而使得开发者可以灵活地构建高可用、可扩展的 SGX 飞地集群.

(3) 本文提出了一种方法用于定量描述 SGX EPC 页交换吞吐量和飞地托管服务延迟之间的关系, 以改进现有的针对非安全处理器的负载均衡和水平扩展方法.

(4) 本文实现了一个工业级的原型系统, 并通过在一系列代表性的机器学习模型上进行实验对其系统开销、可行性和有效性进行评估.

1 SGX 概述

为了更好地理解 S3ML 的研究动机与系统设计, 本文首先对 SGX 进行简要介绍. 截至 2021 年 3 月, SGX 包括两个主要版本, 最初的版本通常被称为 SGX 1.0, 另一个版本具有增强的特性, 如飞地动态内存管理, 被称为 SGX 2.0. 由于 SGX 2.0 仅在有限 CPU 系列上支持, 且 SGX 1.0 已经包含本文所需的所有 SGX 核心的安全特性, 因此本文只讨论 SGX 1.0.

1.1 SGX 基础

SGX 是一种用于构建可信执行环境的新兴技术, 是一种基于硬件实现的在非可信平台上进行可信计算的解决方案. SGX 技术包括一组 CPU 指令扩展和专用硬件架构, 用于实现一种被称为飞地的安全容器. SGX 通过特殊机制保护运行在飞地中的代码以及数据的机密性与完整性. 飞地运行在特殊的加密内存里, 只有飞地内的程序运行在 CPU 时才会对相应的代码和数据进行解密. 因此, 即使攻击者有机会接触到硬件, 也只能窥探到加密后的信息. 在一个飞地进行初始化之前, 代码和数据驻留在主机的非可信区域. 同时, SGX 也无法保障飞地以外数据的安全性. 因此, 为了保护私密数据, 在使用 SGX 时, 数据应该在飞地外进行加密, 然后传输到飞地内解密再进行计算.

1.2 认证和密封

SGX 提供了认证机制, 其功能是让用户可以验证一段已知的程序确实是运行在一台支持 SGX 功能的计算机的飞地里的. 认证又包括本地认证 (local attestation) 和远程认证 (remote attestation). 本地认证使得一个飞地可以向运行在同一平台上的另一个飞地证明自己的身份. 远程认证使得一个飞地可以向远程第三方实体证明自己的身份. SGX 提供了飞地测度 (即 MRENCLAVE 和 MRSIGNER 两个值) 用于在认证过程中表明飞地的身份并验证飞地的完整性.

在飞地中运行的代码和数据是易失性的. 密封是 SGX 提供的一种机制, 用于将数据保存到持久化的非可信存储中, 例如硬盘. 在密封过程中, 应用使用密封密钥 (sealing key) 在飞地中加密数据. 密封密钥只能由该应用的飞地程序获得, 并且是从根密封密钥 (root sealing key) 衍生而来的. 根密封密钥则被硬件编码在 CPU 中, 并且每一个支持 SGX 的 CPU 的根密封密钥都是互不相同的, 这就决定了某一飞地在某台机器上密封的数据只能在该台机器上进行解密.

1.3 飞地页面缓存

飞地所驻留的特殊加密内存区域被称为飞地页面缓存. EPC 由专用硬件加密, 禁止所有来自非飞地内存的访问. EPC 以 4 KB 的粒度划分为页面进行内存分配. 飞地在初始化时申请它需要的所有 EPC 空间, 并且在分配之后不能动态修改. EPC 的总大小非常有限, 在一台计算机上, SGX 飞地应用程序可以使用的 EPC 空间是 128 MB. 在这其中, 只有约 93.5 MB 的大小可供用户程序和数据使用, 剩余空间预留用来存储 SGX 相关的元数据. 飞地在初始化时申请其所需要的所有 EPC 空间, 并且不能在后续进行动态更改. 在目前市场主流的 Linux 操作系统上, SGX 驱动支持通过页交换技术来实现 EPC 超配. 每台机器上的飞地可以申请超过128 MB 的内存使用. 当 EPC 中没有空闲页分配给飞地使用时, SGX 驱动程序会基于特定的缓存替换算法将 EPC 中的某些页面加密后放置到非加密的普通 DRAM 内存上去. 当这些换出的页面在重新被需要使用时又会被换入到 EPC 中. 因为换入换出页面时涉及到完整性校验以及数据加解密等高计算开销的过程, 因此会导致应用程序性能的显著降低. 由于 EPC 很小, 并且由运行在一台机器上的所有飞地竞争, 因此可能会频繁地触发页交换. Vaucher 等人[17]通过修改 SGX 驱动程序和集群调度器代码, 提出了防止因 EPC 超配而导致页交换的方法. 然而, 这种方法导致许多现实世界中会申请 93 MB 以上 EPC 空间的应用程序无法启动. 此外, 这种刻意禁止 EPC 超配的方法也会导致低效的资源管理. 例如, 对于一个服务而言, 即使在工作负载水平较低无法完全利用所有申请的 EPC 空间时, 也不能将剩余空间留给其他飞地使用. 与先前的工作不同, S3ML 允许 EPC 页交换活动发生, 并通过实验表明通过合理的策略本工作可以显著降低 EPC 页交换导致的程序性能下降.

2 系统设计

S3ML 的设计目标是构建一个安全、高可用、可扩展和低延迟的分布式服务系统, 在保证用户数据隐私安全的前提下, 能够运行在现代数据中心中并面向用户提供稳定的机器学习推理服务.

2.1 威胁模型

S3ML 建立在如下的威胁模型之上: 模型服务器和飞地配置服务在 SGX 飞地内运行. 用户使用机器学习推理服务过程中的输入输出数据是本文认为必须保护的敏感数据. 此外, 对于一些敏感的机器学习模型及其超参数, 也可以选择性地通过加密模型并在飞地内加载模型时解密来进行保护. 用户必须信任 S3ML 系统和 SGX. 对于 S3ML 的信任可以通过向用户公开源代码进行检测的方式来建立. S3ML 设计目标内所能抵抗的攻击者是可以控制主机硬件和包括虚拟机监视器、操作系统在内的特权软件的黑客. 即使遇到这种强力的黑客, S3ML 也可以保护用户在使用机器学习推理服务中的隐私数据不被泄露. 对于针对 SGX 自身硬件和系统的攻击, 例如拒绝服务攻击 (denial-of-service attacks)、侧信道攻击 (side-channel attacks)、隐蔽通道攻击 (covert-channel attacks) 等不在本文的讨论范围之内[18−20].

2.2 整体设计

图 2 展示了 S3ML 的系统架构和关键组件. 整个 S3ML 系统分为两部分, 即安全保证部分和服务级别目标保证部分. 前一部分由基于 SGX 认证机制的飞地配置服务和机器学习推理服务组成, 它们直接与客户端交互并确保系统安全. 为了讨论简洁起见, 图里对两个服务进行了简化, 每个服务只对应一个后端飞地实例. 后一部分是系统支持组件, 不与客户端通信, 负责分析、监控、负载均衡和水平扩展推理服务, 以确保机器学习推理服务能够满足服务级别目标.

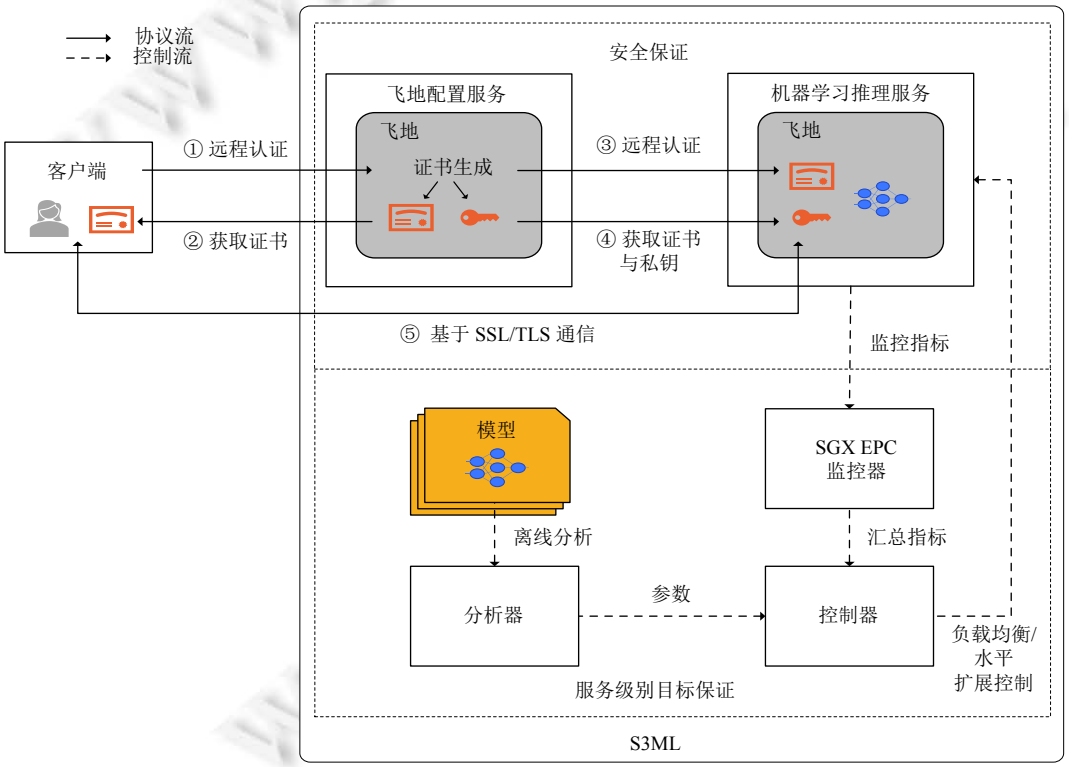

图 2 S3ML 的系统架构

在 S3ML 中, 机器学习推理服务为用户提供核心的推理功能. S3ML 将每一个模型服务器实例运行在 SGX 飞地中来确保推理服务的机密性与完整性. 因此, S3ML 想要实现保护用户数据安全的目标, 关键在于如何在客户端与 SGX 飞地内的模型服务器之间建立一个安全的信道. S3ML 使用 SSL/TLS 协议[21]来实现该安全信道的搭建. 选择该协议的理由如下: (1) 因为该协议在近 20 年内得到了广泛使用以及安全性检验并且可以保护 S3ML 遭受重放攻击. (2) 该协议形成了规范的标准, 得到几乎所有主流的 Web 和 RPC 服务器的支持. 基于 SSL/TLS 协议, 输入输出数据在传输过程中加密, 并且只能在客户端和飞地内部解密. SSL/TLS 安全信道的建立需要在服务器端持有一份证书和相应的私钥, 在客户端只需要持有证书即可. 使用 S3ML 中的服务与使用其他传统非 SGX 环境下的 SSL/TLS 服务 (例如 HTTPS 服务) 的方法基本相同, 唯一的区别在于: 在 S3ML 中, 每个模型服务器实例将在第一次启动时从飞地配置服务获得特定于服务的证书和私钥; 客户端在调用推理服务之前, 也需要向飞地配置服务请求相应服务的证书.

对于使用不同模型的不同服务, S3ML 将首先进行离线分析, 以了解在没有 EPC 干扰的理想环境下服务的相关运行时信息, 以及存在 EPC 干扰的环境下服务尾延迟与 EPC 页交换活动之间的关系. 这些分析结果将被用于运行时调整 S3ML 控制器中的参数. S3ML 周期性地监控模型服务器的状态以及每个集群节点上的 SGX 使用情况 (包括节点的 EPC 页交换吞吐量以及飞地使用 EPC 的大小). 这些指标将被汇总并发送给 S3ML 的控制器组件, 该组件根据预设的算法控制每个服务的负载均衡和水平扩展活动, 以确保服务级别目标得到满足.

2.3 证书与密钥同步协议

图 2 中的实线描述了 S3ML 中的证书与密钥同步协议, 该协议用于确保证书与密钥的分发是安全的. 整个协议的第 1 步是证书与密钥生成. S3ML 不信任由第三方证书颁发机构签发的数字证书, 而是由飞地配置服务负责在飞地内为每个服务生成自签名证书和私钥. 整个协议涉及两个阶段. 第 1 阶段的参与者是客户端和飞地配置服务. 当一个服务完成构建之后, 客户端首先与飞地配置服务通信, 并通过 SGX 远程认证机制验证飞地配置服务的身份. 客户端然后将机器学习推理服务模型服务器的飞地测度值发送给飞地配置服务请求生成推理服务的证书和私钥. 飞地配置服务完成了证书与私钥生成之后, 会把证书发送给客户端. 第 2 阶段的参与者是飞地配置服务和机器学习推理服务. 在每个模型服务器实例初始化时, 其会在飞地内生成一个短期使用的非对称加密公私钥对$<pk_{\text{inf}}, sk_{\text{inf}}>$. 模型服务器会向飞地配置服务发送包含公钥 pk_{inf} 的飞地远程认证报告, 飞地配置服务通过 SGX 远程认证机制验证模型服务器的身份, 并将用 pk_{inf} 加密的服务证书与私钥发送回给模型服务器. 模型服务器在飞地内用私钥 sk_{inf} 进行解密, 得到明文的证书与私钥. 执行完该协议后, 一个机器学习推理服务所有的模型服务器实例都获得了相同的证书与私钥, 客户端也获得了相应的证书. 然后, 客户端可以与机器学习推理服务建立起 SSL/TLS 安全信道.

在整个过程中, 服务的私钥只在 SGX 飞地内可见, 以保证机密性. 客户端验证飞地配置服务的完整性, 然后由飞地配置服务验证所有模型服务器飞地的完整性. 在 S3ML 中, 模型服务器飞地不需要反向验证飞地配置服务的身份. 原因在于: 如果飞地配置服务是伪造的, 模型服务器飞地就不能获得可以与客户端成功建立 SSL/TLS 信道的证书与私钥, 因此用户数据仍然不会泄露.

2.4 基于 SGX 认证机制的飞地配置服务

基于 SGX 认证机制的飞地配置服务是 S3ML 中的核心组件, 其本身也是一个运行在飞地里的集群服务. 飞地配置服务由多个后端服务器实例组成, 用于负载均衡和故障转移. 每个服务器实例运行在 SGX 飞地内, 确保其所生成的证书和密钥的安全性. 由于每个机器学习推理服务和客户端都需要在第一次调用之前与飞地配置服务进行交互, 因此飞地配置服务的可用性对于整个 S3ML 而言至关重要.

飞地配置服务是一个有状态的服务, 它的状态指的是其管理的所有其他服务的飞地测度值、SSL/TLS 证书与私钥. 飞地配置服务有状态的性质为实现其高可用的设计带来了更多的挑战. 首先, 飞地使用的 EPC 内存是易失性的, 需要对状态进行加密才能持久化地保存在非可信的外部高可用存储中. 其次, 需要在飞地配置服务的所有后端实例间实现状态同步. 最后, 还要防止一些极端的场景, 例如数据中心出现电源故障, 导致所有飞地中用来持久化状态的密钥丢失.

图 3 描述了飞地配置服务的设计. 当一个飞地配置服务的后端实例在初始化时, 它会首先判断自己是否是第 1 个启动的后端服务器实例. 这可以通过悲观的并发控制机制 (例如竞争分布式锁) 或乐观的并发控制机制 (例如 compare-and-swap, CAS 操作[22]) 来实现. 如果确认是飞地配置服务的第 1 个后端实例, 它将生成一个长期使用的对称加密密钥 sk_{storage} 用于加密飞地配置服务的状态. 同时, 该实例也会使用 SGX 的密封机制将 sk_{storage} 持久化保存到本机存储上. 之后, 该实例将启动通信服务器并注册到飞地配置服务的前端. 对于后续启动的飞地配置服务实例, 每个实例将首先确定本机存储上是否存在密封的 sk_{storage}. 如果存在的话, 可以通过 SGX 解封机制 (unsealing) 直接得到 sk_{storage}. 如果不存在的话, 实例将生成一个短期使用的非对称加密公私钥对<pk_{AECS}, sk_{AECS}>, 并将包含公钥 pk_{AECS} 的飞地远程认证报告发送给飞地配置服务. 注意, 因为此时飞地配置服务至少有一个后端实例已经启动了, 所以其可以提供正常的服务. 飞地配置服务通过 SGX 远程认证机制来验证该实例的身份, 并使用 pk_{AECS} 加密 sk_{storage} 发送给实例. 然后, 实例使用私钥 sk_{AECS} 解密得到明文的 sk_{storage}, 并使用 SGX 密封机制将其保存到本地存储. 最后, 该实例启动通信服务器, 并注册到飞地配置服务前端.

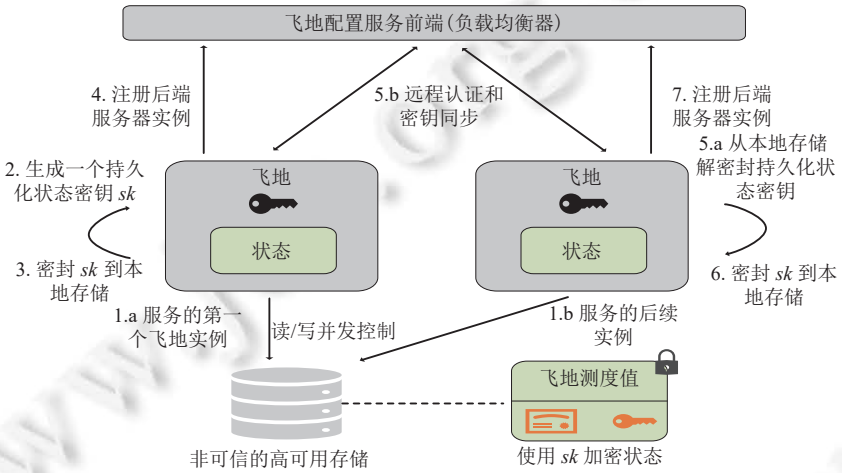

图 3 飞地配置服务的设计

在飞地配置服务完全启动之后, 每个后端实例都获得了相同的长期使用的对称加密密钥 sk_{storage}. sk_{storage} 用来加密飞地配置服务的状态, 并将密文存储在外部非可信的高可用存储中 (例如云对象存储) 用于持久化和状态同步. 如第 2.3 节所述, 客户端在调用机器学习推理服务之前会与飞地配置服务进行交互来生成服务证书与私钥, 这会导致飞地配置服务状态的创建. 飞地配置服务还为客户端提供了证书与私钥更新和删除等功能的接口, 客户端根据自身需求通过调用服务接口请求飞地配置服务完成相应功能. 这些功能的调用过程和证书与私钥的创建本质上无明显区别, 这里不再赘述. 负载均衡器会将客户端的请求分发到不同的后端实例, 因此这些实例需要在共享的非可信存储上进行状态同步. 由于涉及飞地配置服务的交互并不频繁, S3ML 当前使用乐观的并发控制策略来保证不同后端实例对共享存储读写操作中的数据一致性.

虽然 S3ML 是为机器学习推理服务实现的分布式系统, 但是飞地配置服务的高可用设计并不局限于这种使用场景. 飞地配置服务提供了一种通用的解决方案, 能够帮助 Web 服务或其他分布式计算框架构建安全、高可用、可扩展的飞地集群. 以有状态的分布式计算作业为例, 将其运行在 SGX 环境中的常规做法是分别将每个任务运行在一个独立的飞地中, 那么应用主程序与每个任务的通信仍然要进行密钥协商来保证安全[23]. 这种需求与 S3ML 是一致的, 仍然可以依赖本文所提出的飞地配置服务快速地建立应用主程序与每个任务间基于 SSL/TLS 的安全信道.

2.5 机器学习推理服务

S3ML 中的每个机器学习推理服务都由其服务前端和一系列的后端模型服务器实例所构成. 每个模型服务器

都被设计成自包含和无状态的. 自包含指的是每个后端实例只需要在初始化阶段与飞地配置服务通信以获得建立 SSL/TLS 的证书和私钥, 此后不需要外部依赖和存储来保存其状态, 就能独立地为每个请求提供服务. 自包含的性质避免了服务与未经认证的外部服务进行交互, 从而降低了数据泄漏的风险. 无状态特性使得服务具有更高的可用性与可扩展性, 即 S3ML 可以通过启动不同数量的实例来动态地改变服务容量. 这样, S3ML 可以直接复用现有的支持一致性算法 (例如 Paxos[24]和 Raft[25]) 的高可用中间件, 实现服务发现和服务健康检查, 通过将流量路由到健康的实例和重启崩溃的实例来保障服务的可用性.

本质上来说, 用于机器学习推理的模型服务器的实现与 S3ML 系统是解耦的. 模型服务器主要包含两个功能, 即通信功能和机器学习推理功能. 对于前者, S3ML 允许使用支持标准 SSL/TLS 协议的任何通信设施来构建通信服务器. 其实现不限于特定的应用层协议 (例如 HTTP 或 RPC) 和特定的软件 (例如 Apache[26]、Nginx[27]或 gRPC[28]). 对于后者, 近年来学术界和工业界有很多的工作关注于实现 SGX 环境下的机器学习训练或推理应用. 在当前 S3ML 原型系统的实现中, 我们基于一个流行的轻量级机器学习推理框架 TensorFlow Lite 和一个目前最先进的 SGX 库操作系统[29]Occlum 实现了在飞地中进行机器学习推理的功能. 需要注意的是, 提升单机上 SGX 飞地中机器学习推理应用的性能并不是本文的关注重点, 我们也可以基于这些与 S3ML 正交的相关工作 (如 TF Trusted[30]、Occlumency[31]和 Vessels[32]) 来构建机器学习推理服务的模型服务器.

2.6 推理服务离线分析

本小节首先定性地观察 EPC 页交换活动对机器学习推理服务延迟的影响. 具体而言, 本文使用如下的方法来探究这种影响: 第 1 步, 在一台支持 SGX 的机器上启动一个运行在飞地内的模型服务器, 然后持续向这个模型服务器发送推理请求. 第 2 步, 在同一台机器上的另一个飞地内运行一个干扰程序. 该程序会申请一块连续的 EPC 内存空间, 并以无限循环的方式访问该内存中的每一个页面. 该干扰程序的目的是确保分配给该程序的 EPC 页面能够因始终处于活跃状态而被驱动程序保留在 EPC 中. 这样, 模型服务器飞地仅能使用 EPC 中的剩余资源. 当剩余资源不足够模型服务器使用时, 便能刻意地制造出 EPC 页交换活动.

以图像分类任务的模型 MobileNetV1 (float)[33]为例. 图 4 显示了在上述过程中, 服务延迟、EPC 页换入吞吐量、EPC 页换出吞吐量、每个飞地占用 EPC 空间大小的变化趋势. 可以看到, 当模型服务器启动时, 所有的 EPC 空间都分配给了它使用. 这是因为 Occlum 启动时实际上会请求超过 93 MB 的 EPC, 尚未使用的空间会被预留用于启动其他用户空间进程. 由于 EPC 的分配是在飞地初始化时确定的, 因此模型服务器可能不会实际使用其分配到的所有 EPC 大小. 当干扰程序飞地在第 25 s 和第 80 s 启动时, 它立刻抢占了分配给模型服务器飞地的 EPC, 并将剩余的 EPC 留给了模型服务器. 这个观察符合实验预期, 确认了通过这种方式确实能够人为地制造 EPC 页交换活动. 在干扰程序飞地启动后, EPC 页面不断地被换入和换出, 导致推理服务的延迟显著上升. 在第 50 s 和第 106 s 时, 干扰程序飞地被终止, 其占用的 EPC 空间被释放, EPC 页交换吞吐量立刻下降, 推理服务的延迟也重新回到干扰程序飞地启动前的水平. 简而言之, 通过该实验定性地确认了 EPC 页交换对服务延迟有显著影响.

基于以上观察, 本文进一步定量地研究会触发违反服务级别目标的 EPC 页交换吞吐量的边界. 首先, 仍然是启动一个模型服务器飞地和一个干扰程序飞地. 然后, 客户端将持续向模型服务器发送请求. 通过配置干扰程序所请求 EPC 空间的大小来人为地控制 EPC 页交换吞吐量的大小. 图 5 显示了在不同的平均 EPC 页换入吞吐量下, MobileNetV1 (float) 模型为 2 000 个请求服务的 90% 分位点、95% 分位点以及 99% 分位点的延迟 (图中 P90、P95、P99 表示). 可以观察到, EPC 页换入吞吐量和服务延迟是正相关的, EPC 页换入吞吐量越高, 各分位点服务延迟越高. 给定一个机器学习推理服务的服务级别目标 (图中 SLO 表示), 可以确定触发服务级别目标违背的 EPC 页交换 (换入或换出) 吞吐量的边界. 请注意, 在此实验中, 被换入和换出的 EPC 页面几乎都是分配给模型服务器飞地使用的页面. 原因在于干扰程序飞地通过无限循环遍历页面来保持其申请的页面能一直驻留在 EPC 中. 在同时运行批处理作业和延迟敏感型服务的实际集群场景中, 监控器只能知道总的 EPC 页交换吞吐量是多少, 而无法获知其中有多少被交换的页面是属于模型服务器飞地的, 而正是这些页面的换入和换出对服务延迟产生了直接的影响. 但是, 使用本实验所获得的边界值作为监控服务级别目标违反的指标仍然是可行的. 因为这是一个更安全的

数值, 它基于一个更保守的假设来防止服务级别目标无法满足. 真实 EPC 干扰场景中的服务延迟可能会低于预期, 因为一些被交换的页面也可能属于服务飞地以外的其他程序飞地. 另外, 由于每个机器学习推理服务的飞地在二进制文件大小、模型大小、特征空间、计算复杂度和服务级别目标要求上都存在不同, 因此需要进行独立的离线分析来确定触发服务级别目标违反的 EPC 页交换吞吐量边界.

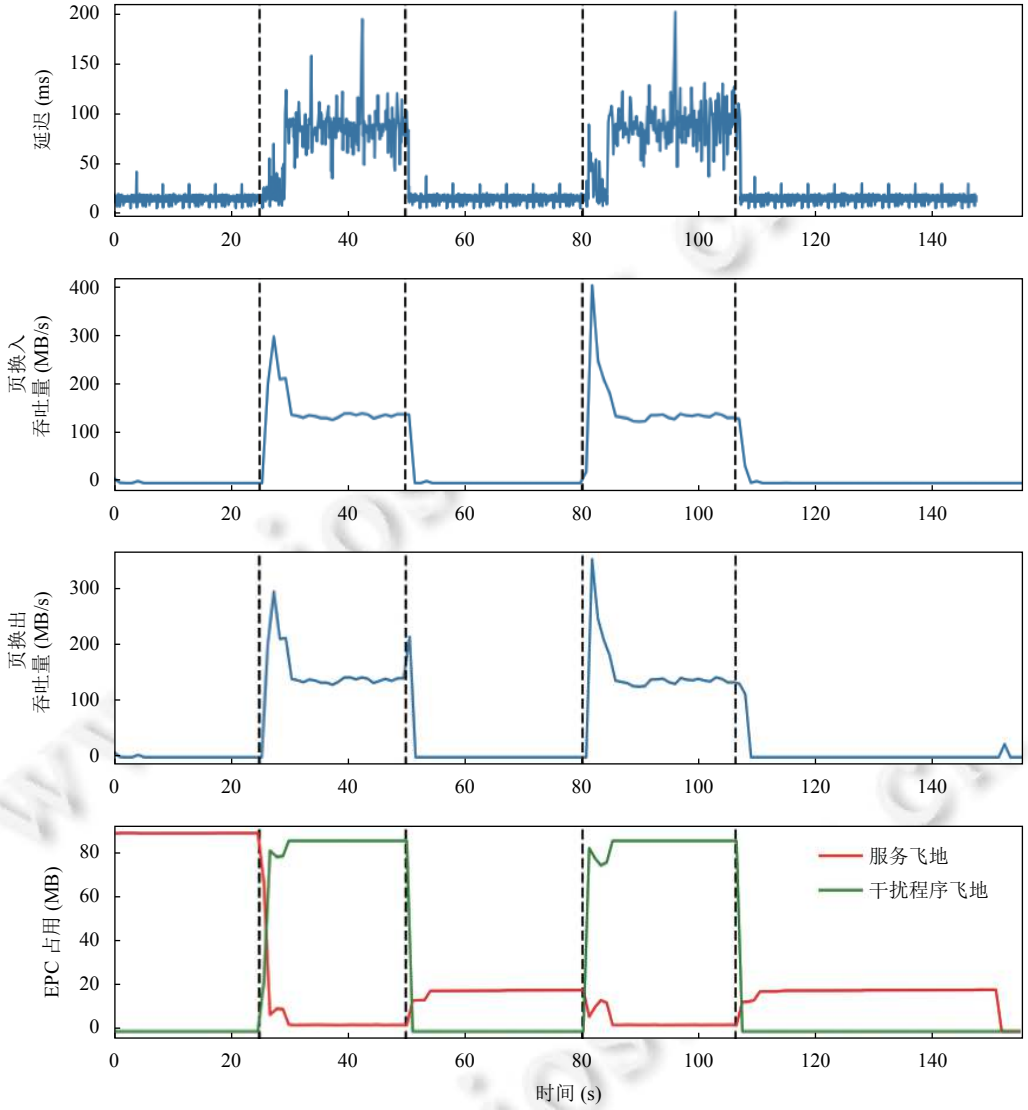

图 4 EPC 页交换活动对服务延迟的影响

2.7 负载均衡和水平扩展控制

离线分析的结果会进一步地发送给 S3ML 的控制器, 来控制负载均衡和水平扩展活动. S3ML 选择使用 EPC 页交换吞吐量作为服务延迟的间接指标, 而不是直接在负载均衡前端收集服务延迟数据是出于如下的原因: 由于 SSL/TLS 流量只能在飞地内解密, S3ML 负载均衡器无法获取应用层的服务延迟数据. 因此, S3ML 使用传输层的负载均衡机制 (转发 TCP/UDP 流量) 来分发请求. 这样做的另外一个好处在于: 由于机器学习推理服务的实现与 S3ML 是解耦的, 该方案可以实现透明地将流量分配给不同应用层实现的后端模型服务器实例 (例如 HTTP 服务器或 RPC 服务器), 而无需实现特定于应用程序的负载均衡器.

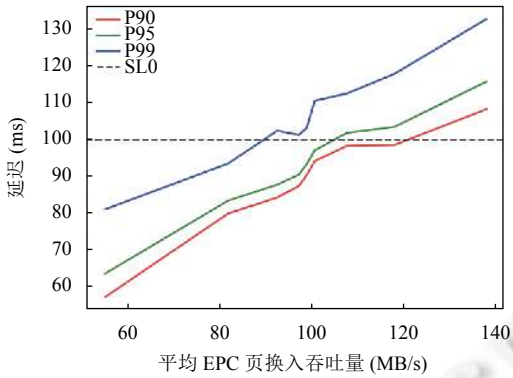

图 5 EPC 页交换吞吐量与机器学习推理服务延迟间的定量关系

在负载均衡和水平扩展控制上, S3ML 利用实时 EPC 监控信息来对现有方法进行改进, 使其具有 SGX 可感知的能力, 以减轻 EPC 页交换活动对服务延迟的影响. 具体来说, S3ML 不直接决定如何将请求从前端分发到后端, 而是控制后端模型服务器实例是否有资格服务请求. S3ML 对负载均衡和水平扩展的控制都遵循基于阈值的被动式控制方法. 算法 1 描述了 S3ML 中负载均衡的控制逻辑, S3ML 周期性地监控每个飞地所在集群节点上的 EPC 活动. 如果一个节点的 EPC 页交换吞吐量 (换入和换出) 连续数个周期高于某一阈值 (服务级别目标违反边界的某一百分比, 例如 70%)(第 6 行), 那么 S3ML 会暂停从服务前端到后端模型服务器实例的网络流量 (第 7 行). 当 S3ML 检测到节点上不再有干扰程序飞地 (除系统架构自带飞地和模型服务器实例飞地以外的其他飞地) 运行时, S3ML 会恢复从服务前端到该实例的网络流量, 使其能够再次为客户端服务 (第 10–12 行). 网络流量的开关通过设置实例负载均衡的权重来实现 (0 为关, 1 为开). 对于水平扩展, 面向无状态服务的算法通常使用平均 CPU 利用率作为调整实例数量的监控指标. S3ML 的不同之处在于, 其只考虑那些实际提供服务的实例 (即负载均衡权流量未被暂停) 的 CPU 利用率. 这样就可以根据 S3ML 实时的服务容量来实现水平扩展. 注意, 本文的创新点在于将 EPC 监控信息引入到负载均衡和水平扩展控制中, 使二者能够具备 SGX 可感知的特性. 对于负载均衡和水平扩展算法本身的创新不是本文的关注点.

算法 1. S3ML 中负载均衡的控制逻辑.

Require: service backend endpoints, cluster nodes, paging threshold, N

Ensure: load balancing decisions

```
1. function LoadBalanceControl (endpoints, nodes, TH, N)
2.   while true do
3.     endpoints.updateCounter(nodes, TH)
4.     for endpoint ∈ endpoints do
5.       if endpoint.weight == 1 then
6.         if endpoint.counter > N then
7.           endpoint.weight ← 0
8.         end if
9.       else
10.        if endpoint.node.runningEnclaves = Ø then
11.          endpoint.weight ← 1
12.        end if
13.      end if
```

14. **end for**
15. **end while**
16. **end function**

3 系统实现

本文基于 Kubernetes 实现了 S3ML. Kubernetes 为开发者提供了一系列实用的抽象和功能来构建和维护高可用、可扩展的服务. 本文基于 gRPC 和 Occlum 开发了 SGX 环境下的飞地配置服务和机器学习推理服务. 推理功能基于 TensorFlow Lite 实现. S3ML 中的服务都以 Kubernetes Service[34]的形式暴露给用户和开发者使用.

S3ML 中的监控功能基于当前 Kubernetes 生态系统中流行的开源监控框架 Prometheus[35]实现. Prometheus 通过运行在集群每个节点上名为 node exporter 的守护进程周期性地采集每个节点上的硬件资源使用信息, 并将它们汇总在一起. 但是, 当前的 node exporter 不支持收集关于 SGX EPC 页交换的信息. 此处存在的另一个问题在于英特尔官方提供的 SGX 驱动程序并不提供 EPC 资源使用的相关信息, 但 Fortanix 开发的开源 SGX 驱动程序[36]为 S3ML 提供了所需要的 EPC 使用信息, 包括总换入和换出的 EPC 页数、每个运行飞地所占用的 EPC 页数等. 基于该驱动程序, S3ML 在 Prometheus node exporter 中实现了一个针对 EPC 资源的信息收集器, 通过读取 /proc/sgx_enclavaes 和/proc/sgx_stats 两个文件来采集 EPC 资源的使用情况.

S3ML 中 SGX 可感知的负载均衡控制是在 Kubernetes 原生 IPVS[37]模式负载均衡功能的基础上扩展而来的. IPVS 模式负载均衡是当前 Kubernetes 中最先进的传输层负载均衡方案, 其通过每个集群节点上运行的一个名为 kube-proxy 的组件来实现. Kube-proxy 通过设置一系列的 IPVS 转发规则, 将网络流量从服务前端分发到不同的后端模型服务器实例. Kube-proxy 目前支持 6 种 IPVS 调度算法, 包括轮询 (round-robin, RR)、最少连接数 (least-connection, LC)、最短预期延迟 (shortest expected delay, SED) 等[38]. S3ML 选择 SED 作为基础负载均衡算法, 通过动态改变模型服务器实例的 IPVS 权重来实现 SGX 可感知的负载均衡控制. S3ML 实现了一个自定义的 Kubernetes 控制器, 它会周期性地从 Prometheus 获取每个节点的 EPC 监控信息, 并从 Kubernetes API Server 获知每个服务的后端实例所在的具体节点, 建立起服务后端实例与集群节点的映射关系, 然后控制器会按照第 2.7 节中的控制逻辑进行工作. 本文还扩展了 kube-proxy 的功能, 使其具备了更新每个节点上 IPVS 权重的能力. 在水平扩展方面, Kubernetes 提供了面向服务的水平扩展功能[39], 开发者可以设置自定义的监控指标来实现服务的水平扩展, 从而达到根据用户负载水平动态地调整服务容量的目的. S3ML 对机器学习推理服务的水平扩展控制基于该功能实现.

4 实验评估

本文通过一系列有代表性的机器学习模型来评估 S3ML 在系统性能、负载均衡和可扩展性上的表现. 所有的实验都在阿里云上完成. 首先, 本文通过比较模型服务器在是否使用 SSL/TLS 通信以及是否运行在 SGX 环境中的性能差异来研究 S3ML 所引入的性能开销. 之后, 本文将 S3ML 的 SGX 可感知的负载均衡方法与 Kubernetes 原生负载均衡算法进行比较. 最后, 本文对 S3ML 的可扩展性进行评估.

4.1 实验设置

4.1.1 实验平台

我们使用 3 台阿里云 ecs.ebmhfg5.2xlarge 裸金属服务器实例构建了一个包含 3 节点的 Kubernetes 集群. Kubernetes 的主节点被配置为允许在其上调度任务 (即 Pod). 每个实例都有 8 个 CPU 核、32 GB 内存以及支持 SGX 功能的 Intel Xeon E3 1240 V6 (Skylake) 处理器. 所有实例都通过高带宽内部网络连接. 表 1 概述了评估实验中所使用的软件环境.

4.1.2 机器学习推理任务和模型

我们使用图像分类任务来评估 S3ML. 图像分类是最具代表性的机器学习任务之一, 它的应用场景非常广泛,

近年来取得了很多突破. 在图像分类任务中, 输入图像和输出结果被视为需要保护的敏感数据. 表 2 列出了本实验中使用的 4 种模型, 即 MobileNetV1 (float)[33]、MobileNetV1 (quantized)[40]、EfficientNetLite (float)[41]和 EfficientNetLite (quantized). 这 4 种模型涵盖了不同范围的模型大小和服务延迟, 使本实验能更泛化地评估 S3ML. MobileNetV1 和 EfficientNetLite 都是在近几年内被提出的具有代表性的新型图像分类模型. 它们基于 ImageNet (ILSVRC-2012-CLS)[42]数据集进行训练, 并针对小型内存设备进行了专门的优化. Float 表示是浮点模型, quantized 表示是量化模型. 本实验所使用的输入数据 (将要被分类的图片) 也是从 ImageNet 数据集中提取的.

表 1 S3ML 评估实验中的软件环境

软件名称	版本
CentOS	7.2
Linux Kernel	3.10.0-514.6.2.el7.x86_64
Kubernetes	1.18.5
Docker	19.03.12
gRPC	1.26.0
TensorFlow Lite	1.15.0
Occlum	0.15.0

表 2 S3ML 评估实验中的模型信息

模型名称	类型	大小	TensorFlow Hub标签
MobileNetV1	浮点型	1.8 MB	mobilenet_v1_0.25_128
MobileNetV1	量化型	485.61 KB	mobilenet_v1_0.25_128_quantized
EfficientNetLite	浮点型	17.72 MB	efficientnet/lite0/fp32
EfficientNetLite	量化型	5.18 MB	efficientnet/lite0/int8

4.2 系统开销

4.2.1 实验设置

在该实验中, 我们设置了 4 种模型服务器的运行环境. (1) 不使用 SSL/TLS 通信的 Linux 环境; (2) 使用 SSL/TLS 通信的 Linux 环境; (3) 不使用 SSL/TLS 通信的 Occlum 环境; (4) 使用 SSL/TLS 通信的 Occlum 环境. Linux 环境意味着推理服务的模型服务器不运行在 SGX 飞地中, 而 Occlum 环境则意味着模型服务器运行在 SGX 飞地中. 最后一种实验环境是 S3ML 中机器学习推理服务实际运行的环境, 另外 3 种设置是作为对比的基准设置.

4.2.2 实验结果

图 6 报告了在 4 种不同环境设置下服务 10 000 个推理请求时, 4 种机器学习模型的平均推理时间以及服务器的平均通信时间. 每个子图从左到右, 每个条形分别代表模型服务器运行在没有 SSL/TLS 的 Linux、有 SSL/TLS 的 Linux、没有 SSL/TLS 的 Occlum 以及有 SSL/TLS 的 Occlum 环境中. 蓝色条形表示用于推理的时间, 橙色条形表示客户端和服务器通信的时间. 可以发现, 与在 Linux 中运行相比, 在 Occlum 中运行模型服务器会引入额外的推理和通信开销. 具体来说, 推理时间会有 1.09–3.77 倍的降低, 具体降低程度取决于使用的是什么模型. 而通信开销则与模型无关, 取决于输入和输出数据. 在不使用 SSL/TLS 的设置中, 通信时间的降低为 4.64–5.78 倍, 而在使用 SSL/TLS 的设置中, 通信时间的降低为 3.90–4.48 倍. 这是由于传输过程中存在额外的数据加密和解密, 使用 SSL/TLS 比明文通信更耗时. 但与总响应时间 (包括推理和通信) 相比, 使用 SSL/TLS 所带来的额外开销可以忽略不计, 尤其是对于推理较慢的模型. 总之, S3ML 通过在运行于 SGX 飞地的模型服务器和客户端之间建立 SSL/TLS 安全信道来提供安全的机器学习推理服务. 对于 4 种流行的机器学习模型, 服务的响应时间在几十毫秒到几百毫秒之间, 这种性能是可以接受的.

4.3 负载均衡

4.3.1 实验设置

我们在本实验中评估 S3ML 中 SGX 可感知的负载均衡控制是否能够减轻来自同一节点上的其他飞地的干扰. 本实验在 Kubernetes 集群上创建了一个有 3 个后端模型服务器实例的机器学习推理服务. 每个模型服务器实例运行在一个集群节点上. 客户端向推理服务发送请求, 请求的到达时间遵循泊松分布. 对于不同的模型, 通过配置泊松分布的均值来控制发送请求的速率. 在每轮实验中, 客户端持续发送 10 min 的服务请求. 在 2–4 min 和 6–8 min 两个时间段内, 人为地在某个节点上启动一个运行在 SGX 飞地内的批处理任务作为干扰任务. 由于目前没

有公开可用的 SGX 负载数据集进行评估, 这里实际运行的是 Stress-SGX[43]程序, 将其作为干扰批处理任务. Stress-SGX 是一个开源的 SGX 测试程序. 为了验证 SGX 可感知的负载均衡控制能够正确地处理不同程度的 EPC 干扰活动, 本实验对每个模型和每种负载均衡算法进行了两次测试. 通过为 Stress-SGX 程序分配不同大小的 EPC 空间, 以在每次实验中模拟高度干扰和低度干扰两种情况. 高度干扰被定义为 Stress-SGX 程序将会持续导致 EPC 页交换吞吐量超过控制阈值的情况. 反之, 低度干扰则不会持续导致 EPC 页交换吞吐量超过控制阈值. 此外, 由于 EPC 干扰可能会导致系统服务容量的下降. 本实验通过合理地设置客户端的请求速率, 可以保证即使 EPC 干扰导致部分模型服务器实例无法提供服务能力, 剩余的模型服务器实例也足以服务全部的请求.

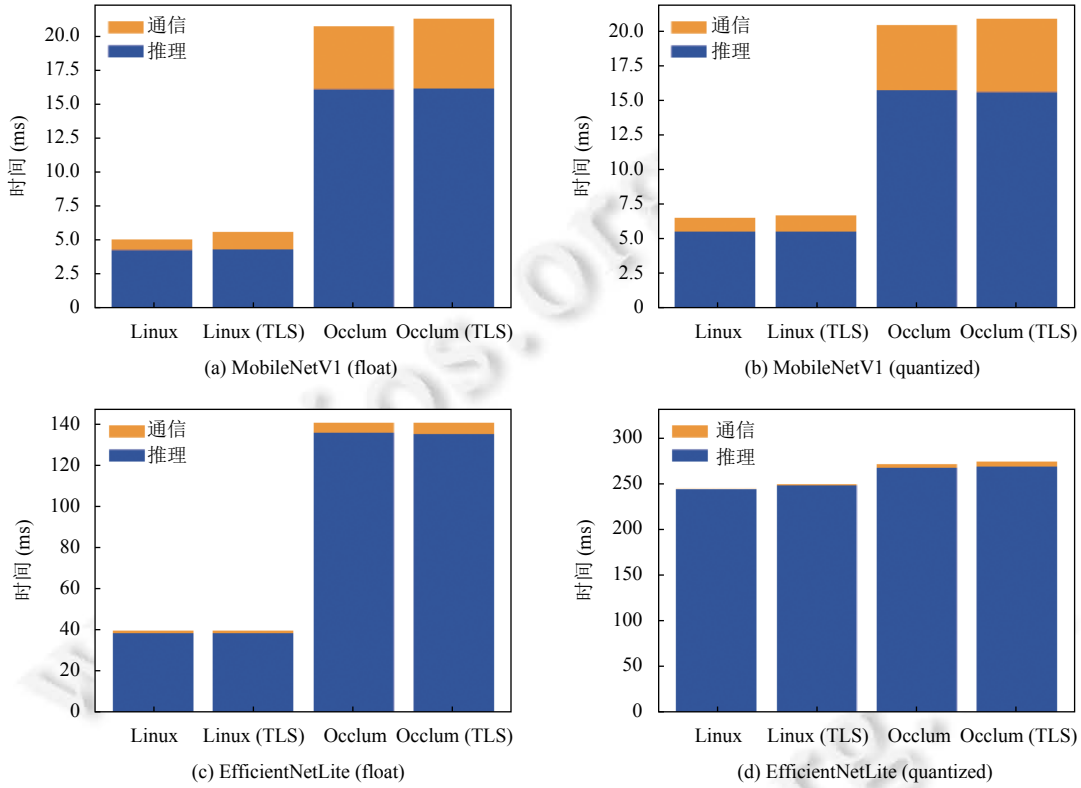

图 6 S3ML 系统开销评估实验结果

4.3.2 参数设置

对于 SGX 可感知的负载均衡控制, EPC 监控信息被设置为每秒采集一次, 连续监控周期数被设置为 5, 触发负载均衡权重更新的阈值百分比被设置为 70%. 因为机器学习推理服务对延迟较为敏感, 为了尽可能地达到监控的实时性, 监控周期参数为 Promethus 中的最小监控粒度. 连续监控周期参数是根据第 2.6 节离线分析中 EPC 页交换活动变化反应到服务延迟变化的时间差来设置的. 对于负载均衡权重更新的阈值参数, 该值设置过低会降低控制逻辑的灵敏度, 设置过高会增加服务级别目标违背的风险. 当前该参数是在进行多次尝试后所设置的经验值. 服务延迟计算的是从模型服务器接收请求到返回响应之间的时间. 本实验中使用 99% 分位点的尾延迟作为服务级别目标的指标. 对于 MobileNetV1 (float)、MobileNetV1 (quantized)、EfficientNetLite (float) 和 EfficientNetLite (quantized) 这 4 个模型, 其服务级别目标分别设置为 100 ms、100 ms、500 ms 和 600 ms.

4.3.3 基准设置

本实验将 SGX 可感知的负载均衡与 3 种 Kubernetes 原生的负载均衡算法比较. 这 3 种算法的描述如下.

• 轮询 (RR): 下一个到达的请求将被分发到下一个服务器.

• 最少连接数 (LC): 下一个到达的请求将被分发到具有最少活动连接数的服务器.

• 最短预期延迟 (SED): 下一个到达的请求将被分发到预期延迟最短的服务器.

4.3.4 实验结果

表 3 报告了在高度和低度干扰设置下使用 4 种不同负载均衡算法的 4 种模型的 99% 分位点的服务延迟. 在发生 EPC 高度干扰的情况下, 在 3 种原生的 Kubernetes 负载均衡算法中, 轮询算法的性能最差. 这其中的原因在于: 当节点上发生了高度的 EPC 干扰, 位于同一节点上的模型服务器的推理延迟将会显著增加. 然而, 轮询算法仍然会持续不断地将到达的请求分发到该模型服务器. 这样导致的结果是模型服务器的服务能力无法承载越来越多的请求, 使得更多的请求延迟上升, 最终导致不能满足服务级别目标. 相比之下, 最短预期延迟算法在 3 种原生算法中具有最佳性能, 因为它总是尽可能地将请求分发给有着预期低延迟的服务器. S3ML 的 SGX 可感知负载均衡在所有的 4 种负载均衡算法中获得了最佳性能, 并且它是唯一一个能够满足 4 个模型所有服务级别目标的方法. 具体来说, 对于每个模型, 与其他 3 种负载均衡算法相比, SGX 可感知的负载均衡将 99% 分位点的服务延迟降低了 64.47%–95.59%、52.31%–95.15%、42.63%–92.89% 以及 20.48%–88.88%. 在 EPC 低度干扰设置下, 可以观察到, 4 种负载均衡算法都不会导致服务级别目标的违反, 并且性能没有显著差异. 这个结果表明, S3ML 的 SGX 可感知的负载均衡控制可以有效地处理不同强度的 EPC 页交换活动所导致的性能干扰.

表 3 S3ML 负载均衡评估实验结果

干扰程度	模型名称	负载均衡算法	99%分位点延迟 (ms)	模型名称	负载均衡算法	99%分位点延迟 (ms)	干扰程度	模型名称	负载均衡算法	99%分位点延迟 (ms)	模型名称	负载均衡算法	99%分位点延迟 (ms)
高	Mobile NetV1 (float)	RR	815.36	Efficient NetLite (float)	RR	3 059.76	低	Mobile NetV1 (float)	RR	**25.20**	Efficient NetLite (float)	RR	188.69
		LC	683.63		LC	527.17			LC	32.50		LC	163.17
		SED	101.11		SED	379.09			SED	30.46		SED	**157.05**
		S3ML	**35.92**		S3ML	**217.50**			S3ML	30.55		S3ML	158.89
	Mobile NetV1 (quantized)	RR	733.13	Efficient NetLite (quantized)	RR	3 863.17		Mobile NetV1 (quantized)	RR	**25.53**	Efficient NetLite (quantized)	RR	394.88
		LC	568.85		LC	552.40			LC	31.44		LC	393.61
		SED	74.61		SED	540.10			SED	29.08		SED	**391.14**
		S3ML	**35.58**		S3ML	**429.48**			S3ML	29.74		S3ML	395.13

4.4 可扩展性

4.4.1 实验设置

我们在本实验中评估 S3ML 中机器学习推理服务的可扩展性. 在实验中, 我们每次使用不同数量的模型服务器实例来为客户端提供服务. 仍然是对 4 种模型进行实验, 每个实例都被调度到一个单独的集群节点上.

4.4.2 实验结果

图 7 给出了具有不同实例数的推理服务的 99% 分位点的延迟和吞吐量. 可以看到, 对于 4 个模型, 服务尾延迟在运行单个和多个实例的场景下没有显著差异, 而服务吞吐量与实例数量间则呈线性关系. 与运行单个实例相比, 运行 3 个实例的模型服务器的吞吐量分别提高了 3.01 倍、2.98 倍、3.03 倍和 2.94 倍. 这个实验结果验证了 S3ML 对于机器学习推理服务仍然具备良好的线性可扩展性.

5 相关工作

本文从机器学习推理服务系统和 SGX 相关应用这两个方面对相关工作进行调研.

5.1 机器学习推理服务系统

Clipper[5]是一个低延迟的采用分层架构设计的服务系统. Clipper 通过将服务系统分为模型选择层和模型抽象层, 实现了在一个系统中可以集成不同的机器学习框架 (如 Spark[44]、TensorFlow[45]). 此外, Clipper 在缓存、批大小等方面进行优化来减少推理延迟并提高推理服务的吞吐量和准确性. MArk[3]和 Swayam[46]是基于云的服务系

统, 旨在实现服务级别目标的同时能够降低成本开销. ParM[6]提出了一种新的基于学习的方法, 用于在服务系统中实现类似于云存储中纠删码技术所能达到的容错能力. TensorFlow-Serving[47]是一个工业级的机器学习推理服务系统, 有着灵活的模型版本管理能力以及良好的性能. Rafiki[48]同时提供机器学习训练和推理服务, 它通过在线集成学习来在推理延迟和推理准确性之间取得平衡. S3ML 与现有的服务系统是正交的. S3ML 的关注重点是使用英特尔 SGX 来为用户提供具备隐私保护能力的机器学习推理服务. 此外, S3ML 提出了 SGX 可感知的服务级别目标保障方法, 为用户提供低延迟的服务.

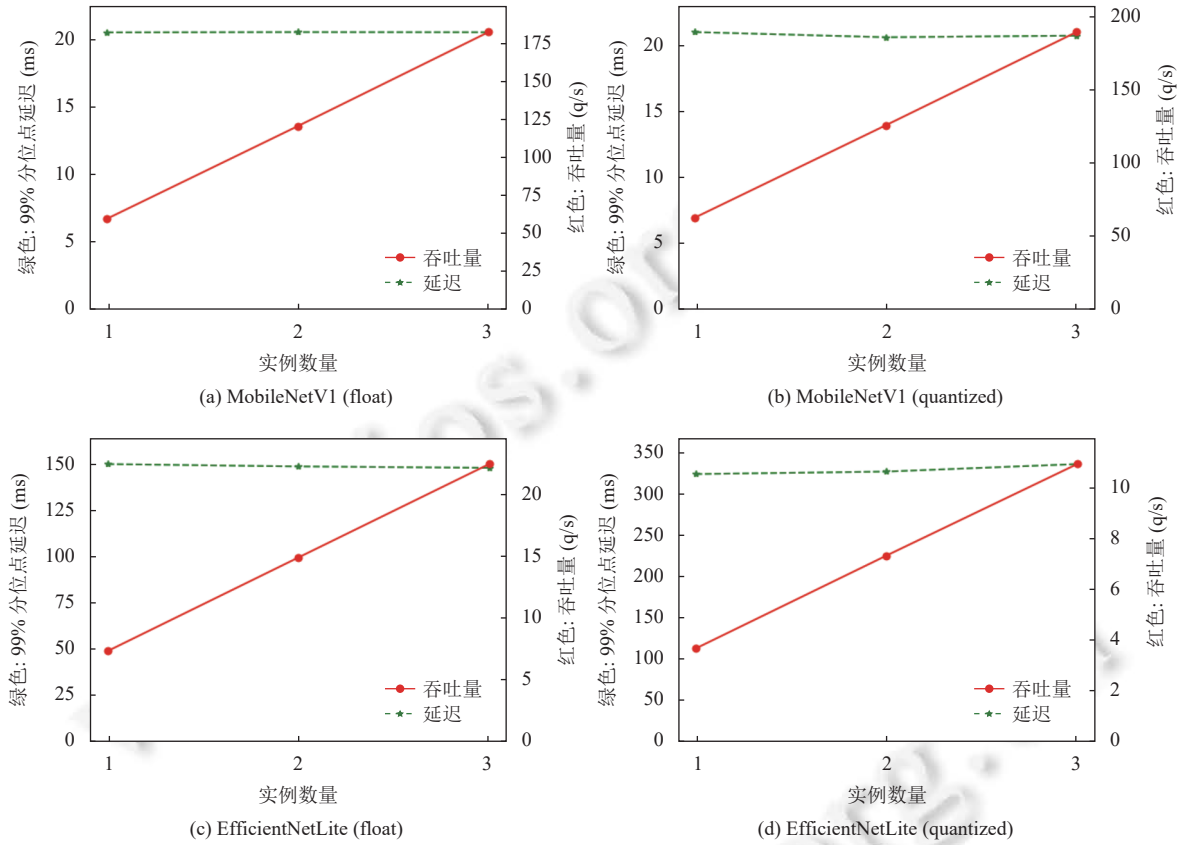

图 7 S3ML 可扩展性评估实验结果

5.2 SGX 相关应用

近年来, 已经出现一系列的工作使用可信执行环境技术构建安全的数据处理系统[23,49]、存储系统[50,51]以及函数加密系统[52]. 董春涛等人[53]和王娟等人[54]对这些工作进行了综述性的介绍. 与这些工作相比, S3ML 有着不同的系统设计和证书与密钥同步协议. 在机器学习领域, TensorSCONE[55]基于 SCONE[56]将 TensorFlow 运行在了 SGX 飞地内部, 实现了一个安全的机器学习系统. SLALOM[57]将深度神经网络的计算分为可信和非可信分部分, 实现了基于 SGX 的安全和高性能的深度神经网络计算. 这些工作主要集中在使用 SGX 进行安全的机器学习训练, 而 S3ML 则侧重于在构建安全的机器学习推理服务. 此外, 还存在部分基于可信执行环境的机器学习推理系统. VanNostrand 等人[58]主要针对移动设备, 基于 ARM 的 TrustZone[59]实现了面向深度神经网络模型的安全推理功能. TF Trusted[30]基于 Asylo[60]和 TensorFlow Lite 实现了在 SGX 飞地内进行机器学习推理的功能. Occlumency[31]基于机器学习框架 Caffe[61]实现了 SGX 环境下的具备隐私保护能力的深度学习推理功能. Vessels[32]则基于 Darknet[62]框架在 SGX 中实现了类似的功能. 这些工作与 S3ML 是正交的, 它们的关注点是在单机上提升机器学

习推理框架运行在可信执行环境中的性能, 而 S3ML 则是关注于构建工业级的安全分布式推理服务系统. 这些工作也可以在 S3ML 中使用来实现模型服务器的推理功能.

6 总结与未来工作

本文提出了一种安全的机器学习推理服务系统 S3ML. S3ML 旨在解决用户使用机器学习推理服务时的隐私保护问题. S3ML 利用英特尔 SGX 保护用户数据的机密性和完整性. S3ML 通过构建安全的证书与密钥管理服务(飞地配置服务), 帮助开发者轻松灵活地构建安全的模型服务器飞地集群, 进一步形成高可用和可扩展的机器学习推理服务. 此外, 本文实验分析了使用 EPC 页交换吞吐量作为保障服务级别目标的间接监控指标的可行性, 并提出了在现代数据中心中不同运行程序之间存在 EPC 争用的情况下, 通过使用实时 SGX EPC 监控信息来控制机器学习推理服务的负载均衡与水平扩展活动, 以满足用户服务级别目标的要求. 通过在一系列流行模型上进行的实验, 本文对 S3ML 的系统开销、可行性以及有效性进行了评估.

本文所提出的系统目前已经部分部署在了真实的工业生产环境中并能稳定地工作, 但仍然存在一些不足: S3ML 目前只适配了有限的机器学习框架并只支持轻量级的机器学习模型. 未来我们将结合在单机上使用 SGX 进行安全机器学习推理的前沿工作以及英特尔 SGX 技术的最新进展, 来进一步扩充系统功能、提升系统性能, 构建更加完善的安全机器学习推理服务系统.

References:

[1] AWS AI services. 2021. https://aws.amazon.com/machine-learning/ai-services/

[2] Google AI and machine learning products. 2021. https://cloud.google.com/products/ai

[3] Zhang CL, Yu MC, Wang W, Yan F. MArk: Exploiting cloud services for cost-effective, SLO-aware machine learning inference serving. In: Proc. of the 2019 USENIX Conf. on Usenix Annual Technical Conf. Renton: USENIX Association, 2019. 1049–1062.

[4] Merkel D. Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014, 239: 2. [doi: 10.1097/01.NND.0000320699.47006.a3]

[5] Crankshaw D, Wang X, Zhou G, Franklin MJ, Gonzalez JE, Stoica I. Clipper: A low-latency online prediction serving system. In: Proc. of the 14th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2017. 613–627.

[6] Kosaian J, Rashmi KV, Venkataraman S. Parity models: Erasure-coded resilience for prediction serving systems. In: Proc. of the 27th ACM Symp. on Operating Systems Principles. Huntsville: ACM, 2019. 30–46. [doi: 10.1145/3341301.3359654]

[7] General data protection regulation (GDPR). https://gdpr-info.eu/

[8] Tan ZW, Zhang LF. Survey on privacy preserving techniques for machine learning. Ruan Jian Xue Bao/Journal of Software, 2020, 31(7): 2127–2156 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6052.htm [doi: 10.13328/j.cnki.jos.006052]

[9] Costan V, Devadas S. Intel SGX explained. IACR Cryptology ePrint Archive, 2016: 86.

[10] Diffie W, Hellman ME. New directions in cryptography. IEEE Trans. on Information Theory, 1976, 22(6): 644–654. [doi: 10.1109/TIT.1976.1055638]

[11] Lo D, Cheng LQ, Govindaraju R, Ranganathan P, Kozyrakis C. Heracles: Improving resource efficiency at scale. In: Proc. of the 42nd Annual Int'l Symp. on Computer Architecture. Portland: ACM, 2015. 450–462. [doi: 10.1145/2749469.2749475]

[12] Linux control groups. https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

[13] Taassori M, Shafiee A, Balasubramonian R. VAULT: Reducing paging overheads in SGX with efficient integrity verification structures. In: Proc. of the 23rd Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Williamsburg: ACM, 2018. 665–678. [doi: 10.1145/3173162.3177155]

[14] Burns B, Grant B, Oppenheimer D, Brewer E, Wilkes J. Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade. Queue, 2016, 14(1): 70–93. [doi: 10.1145/2898442.2898444]

[15] TensorFlow Lite. https://tensorflow.google.cn/lite

[16] Shen YR, Tian HL, Chen Y, Chen K, Wang RJ, Xu Y, Xia YB, Yan SM. Occlum: Secure and efficient multitasking inside a single enclave of intel SGX. In: Proc. of the 25th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Lausanne: ACM, 2020. 955–970. [doi: 10.1145/3373376.3378469]

[17] Vaucher S, Pires R, Felber P, Pasin M, Schiavoni V, Fetzer C. SGX-aware container orchestration for heterogeneous clusters. In: Proc. of the 38th Int'l Conf. on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018. 730–741. [doi: 10.1109/ICDCS.2018.00076]

[18] Wang WH, Chen GX, Pan XR, Zhang YQ, Wang XF, Bindschaedler V, Tang HX, Gunter CA. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 2421–2434. [doi: 10.1145/3133956.3134038]

[19] Chen GX, Chen SC, Xiao Y, Zhang YQ, Lin ZQ, Lai TH. SgxPectre: Stealing intel secrets from SGX enclaves via speculative execution. In: Proc. of the 2019 IEEE European Symp. on Security and Privacy. Stockholm: IEEE, 2019. 142–157. [doi: 10.1109/EuroSP.2019.00020]

[20] Van Bulck J, Minkin M, Weisse O, Genkin D, Kasikci B, Piessens F, Silberstein M, Wenisch TF, Yarom Y, Strackx R. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: Proc. of the 27th USENIX Conf. on Security Symp. Baltimore: USENIX Association, 2018. 991–1008.

[21] Rescorla E. The transport layer security (TLS) protocol version 1.3. RFC 8446, 2018. 1–160.

[22] Herlihy M. Wait-free synchronization. ACM Trans. on Programming Languages and Systems, 1991, 13(1): 124–149. [doi: 10.1145/114005.102808]

[23] Schuster F, Costa M, Fournet C, Gkantsidis C, Peinado M, Mainar-Ruiz G, Russinovich M. VC3: Trustworthy data analytics in the cloud using SGX. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 38–54. [doi: 10.1109/SP.2015.10]

[24] Lamport L. The part-time parliament. ACM Trans. on Computer Systems, 1998, 16(2): 133–169. [doi: 10.1145/279227.279229]

[25] Ongaro D, Ousterhout JK. In search of an understandable consensus algorithm. In: Proc. of the 2014 USENIX Annual Technical Conf. Philadelphia: USENIX, 2014. 305–319.

[26] Apache HTTP server. https://httpd.apache.org/

[27] Nginx. Microservices March 2022: Kubernetes networking. https://www.nginx.com/

[28] gRPC. A high performance, open source universal RPC framework. 2021. https://grpc.io/

[29] Engler DR, Kaashoek MF, O'Toole J. Exokernel: An operating system architecture for application-level resource management. In: Proc. of the 15th ACM Symp. on Operating Systems Principles. Copper Mountain: ACM, 1995. 251–266. [doi: 10.1145/224056.224076]

[30] TF trusted. https://github.com/capeprivacy/tf-trusted

[31] Lee T, Lin ZQ, Pushp S, Li CH, Liu YX, Lee YK, Xu FY, Xu CR, Zhang LT, Song JH. Occlumency: Privacy-preserving remote deep-learning inference using SGX. In: Proc. of the 25th Annual Int'l Conf. on Mobile Computing and Networking. Los Cabos Mexico: ACM, 2019. 46. [doi: 10.1145/3300061.3345447]

[32] Kim K, Kim CH, Rhee JJ, Yu X, Chen HF, Tian D, Lee B. Vessels: Efficient and scalable deep learning prediction on trusted processors. In: Proc. of the 11th ACM Symp. on Cloud Computing. Virtual: ACM, 2020. 462–476. [doi: 10.1145/3419111.3421282]

[33] Howard AG, Zhu ML, Chen B, Kalenichenko D, Wang WJ, Weyand T, Andreetto M, Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv: 1704.04861, 2017.

[34] Kubernetes service. 2021. https://kubernetes.io/docs/concepts/services-networking/service/

[35] Prometheus. From metrics to insight. 2021. https://prometheus.io/

[36] Fortanix. Linux-sgx driver. https://github.com/fortanix/linux-sgx-driver

[37] Linux virtual server. http://www.linuxvirtualserver.org/

[38] Scheduling algorithms in LVS. 2021. http://www.linuxvirtualserver.org/docs/scheduling.html

[39] Kubernetes. Horizontal pod autoscaling. 2021. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[40] Jacob B, Kligys S, Chen B, Zhu ML, Tang M, Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 2704–2713. [doi: 10.1109/CVPR.2018.00286]

[41] Tan MX, Le QV. EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 6105–6114.

[42] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang ZH, Karpathy A, Khosla A, Bernstein M, Berg AC, Li FF. ImageNet large scale visual recognition challenge. Int'l Journal of Computer Vision, 2015, 115(3): 211–252. [doi: 10.1007/s11263-015-0816-y]

[43] Vaucher S, Schiavoni V, Felber P. Short paper: Stress-SGX: Load and stress your enclaves for fun and profit. In: Proc. of the 6th Int'l Conf. on Networked Systems. Essaouira: Springer, 2018. 358–363. [doi: 10.1007/978-3-030-05529-5_24]

[44] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012. 15–28.

[45] Abadi M, Barham P, Chen JM, et al. TensorFlow: A system for large-scale machine learning. In: Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation. Savannah: USENIX, 2016. 265–283.

[46] Gujarati A, Elnikety S, He YX, McKinley KS, Brandenburg BB. Swayam: Distributed autoscaling to meet SLAs of machine learning inference services with resource efficiency. In: Proc. of the 18th ACM/IFIP/USENIX Middleware Conf. Las Vegas: ACM, 2017. 109–120. [doi: 10.1145/3135974.3135993]

[47] Olston C, Fiedel N, Gorovoy K, Harmsen J, Lao L, Li FW, Rajashekhar V, Ramesh S, Soyke J. TensorFlow-serving: Flexible, high-performance ML serving. In: Proc. of the 31st Conf. on Neural Information Processing Systems. Long Beach, 2017.

[48] Wang W, Gao JY, Zhang MH, Wang S, Chen G, Ng TK, Ooi BC, Shao J, Reyad M. Rafiki: Machine learning as an analytics service system. In: Proc. of the VLDB Endowment, 2018, 12(2): 128–140. [doi: 10.14778/3282495.3282499] [doi: 10.14778/3282495.3282499]

[49] Zheng WT, Dave A, Beekman JG, Popa RA, Gonzalez JE, Stoica I. Opaque: An oblivious and encrypted distributed analytics platform. In: Proc. of the 14th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX, 2017. 283–298.

[50] Priebe C, Vaswani K, Costa M. EnclaveDB: A secure database using SGX. In: Proc. of the 2018 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2018. 264–278. [doi: 10.1109/SP.2018.00025]

[51] Krahn R, Trach B, Vahldiek-Oberwagner A, Knauth T, Bhatotia P, Fetzer C. Pesos: Policy enhanced secure object store. In: Proc. of the 13th EuroSys Conf. Porto: ACM, 2018. 25. [doi: 10.1145/3190508.3190518]

[52] Fisch B, Vinayagamurthy D, Boneh D, Gorbunov S. IRON: Functional encryption using intel SGX. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 765–782. [doi: 10.1145/3133956.3134106]

[53] Dong CT, Shen QN, Luo W, Wu PF, Wu ZH. Research progress of SGX application supporting techniques. Ruan Jian Xue Bao/Journal of Software, 2021, 32(1): 137–166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6095.htm [doi: 10.13328/j.cnki.jos.006095]

[54] Wang J, Fan CY, Cheng YQ, Zhao B, Wei T, Yan F, Zhang HG, Ma J. Analysis and research on SGX technology. Ruan Jian Xue Bao/Journal of Software, 2018, 29(9): 2778−2798 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5594.htm [doi: 10.13328/j.cnki.jos.005594]

[55] Kunkel R, Quoc DL, Gregor F, Arnautov S, Bhatotia P, Fetzer C. TensorSCONE: A secure TensorFlow framework using intel SGX. arXiv: 1902.04413, 2019.

[56] Arnautov S, Trach B, Gregor F, Knauth T, Martin A, Priebe C, Lind J, Muthukumaran D, O'Keeffe D, Stillwell M, Goltzsche D, Eyers DM, Kapitza R, Pietzuch PR, Fetzer C. SCONE: Secure linux containers with intel SGX. In: Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation. Savannah: USENIX, 2016. 689–703.

[57] Tramèr F, Boneh D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview. net, 2019.

[58] VanNostrand PM, Kyriazis I, Cheng M, Guo T, Walls RJ. Confidential deep learning: Executing proprietary models on untrusted devices. arXiv: 1908.10730, 2019.

[59] ARM Holdings. Building a secure system using TrustZone technology. Technical Report, ARM Ltd., 2009.

[60] Asylo. https://github.com/google/asylo.

[61] Jia YQ, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. Caffe: Convolutional architecture for fast feature embedding. In: Proc. of the 22nd ACM Int'l Conf. on Multimedia. Orlando: ACM, 2014. 675–678. [doi: 10.1145/2647868.2654889]

[62] Joseph R. Darknet: Open source neural networks in C, 2013–2016. http://pjreddie.com/darknet/

附中文参考文献:

[8] 谭作文, 张连福. 机器学习隐私保护研究综述. 软件学报, 2020, 31(7): 2127–2156. http://www.jos.org.cn/1000-9825/6052.htm [doi: 10.13328/j.cnki.jos.006052]

[53] 董春涛, 沈晴霓, 罗武, 吴鹏飞, 吴中海. SGX应用支持技术研究进展. 软件学报, 2021, 32(1): 137-166. http://www.jos.org.cn/1000-9825/6095.htm [doi: 10.13328/j.cnki.jos.006095]

[54] 王鹃, 樊成阳, 程越强, 赵波, 韦韬, 严飞, 张焕国, 马婧. SGX技术的分析和研究. 软件学报, 2018, 29(9): 2778−2798. http://www.jos.org.cn/1000-9825/5594.htm [doi: 10.13328/j.cnki.jos.005594]

马俊明(1994－), 男, 博士, 主要研究领域为云计算, 分布式系统.

周爱辉(1990－), 男, 学士, 主要研究领域为隐私计算.

吴秉哲(1994－), 男, 博士, CCF 学生会员, 主要研究领域为机器学习, 隐私计算.

巫锡斌(1989－), 男, 硕士, 主要研究领域为机器学习系统, 隐私计算, 云原生.

余超凡(1990－), 男, 硕士, 主要研究领域为隐私计算.

陈向群(1961－), 女, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为系统软件, 软件工程.