并行與分布式計算通部

地　址：武汉市华中科技大学东五楼一楼
邮　编：430074
电　话：（027）87541924 或 87543529
传　真：（027）87557354
E-mail：liuqin19@hust.edu.cn
网　址：http://grid.hust.edu.cn

SCTS 服务计算技术与系统教育部重点实验室

CGCL 集群与网格计算湖北省重点实验室

并行與分布式計算通部

BING XING YU FEN BU SHI JI SUAN TONG XUN

2019年第4期　总第39期　2019年12月

http://grid.hust.edu.cn

2019年区块链与可信系统国际会议
2019 International Conference on Blockchain and Trustworthy Systems

2019中国计算机大会
CNCC2019

风采

2019年实验室学术成果展示

光阴似箭，岁月无痕，转眼2019年就要结束了。回首2019年，实验室全体师生用智慧和汗水收获了一批高水平学术成果。经统计，2019年实验室师生在实验室Top80和CCF A类国际学术会议上发表论文19篇，IEEE/ACM期刊上发表论文17篇，其他SCI期刊上发表论文22篇。

在ASPLOS 2019会议上，张宇等人针对迭代有向图处理系统面临的图顶点状态传递缓慢和冗余数据处理等问题，提出了一种以路径为基本并发处理单位的GPU图计算系统。在ISCA 2019会议上，姜炜祥等人针对数据中心温水冷却系统面临的负载不平衡等问题，提出一种细粒度制冷控制方法。在USENIX ATC 2019会议上，金盼盼等人针对互联网突发事件造成的后端服务器负载失衡问题，设计了一种动态按需组包卸载与批处理系统。在INFOCOM 2019会议上，陈姝彤等人针对边缘数据中心应急需求响应面临的时延和频繁开关问题，提出一种面向边缘数据中心的应急需求响应在线市场机制；高彬等人针对移动边缘计算架构下的用户服务质量问题，设计了一种在线优化方法；张启夏等人针对5G网络切片中虚拟网络功能部署面临的性能需求问题，提出一种自适应干扰感知的虚拟网络功能联合部署方案。在SC 2019会议上，赵进等人针对现有图计算框架中并发任务之间的冗余数据存储和访问问题，设计了一个高效的图存储系统。在ICDCS 2019会议上，顾琳等人针对如何协调虚拟网络功能实例的部署并在服务功能链间调度流量以实现网络效用最大化的问题，提出一种基于深度确定性策略梯度的算法；陈琼等人针对边缘环境下多任务迁移学习的有效性问题，设计了一种数据驱动的机器学习任务分配机制；戴小海等人针对比特币数据存储开销大的问题，设计了一种类似于拼图的数据减缩方案；胡蝶等人针对边缘环境中流处理应用的延迟问题，设计了一种应用于边缘服务器上的FPGA加速的通用分布式流处理系统；袁平鹏等人针对RDF图数据研究面临的问题，从存储、划分、计算框架和分析等四个方面分析研究现状及成果；王飞越等人针对Cuckoo过滤器的数据插入过程中负载不均导致的延时增加的问题，设计了一套面向大数据集合的插入、查询和删除等操作的高效算法；张凡等人针对数据的倾斜分布限制分布式流连接处理系统并行处理能力的问题，提出了一种高时效可扩展的流连接系统。在HPDC 2019会议上，黄航等人针对容器隔离性不足的问题，设计了一种容器有效资源的计算方法和动态调整机制。在IPDPS 2019会议上，樊浩等人针对不同特征的并发I/O负载产生的NCQ竞争问题，设计了一种NCQ感知的I/O调度框架；周舜杰等人针对数据倾斜给现有流连接系统数据划分带来的计算负载不均的问题，提出了一种高性能的分布式流连接系统。在MSST 2019会议上，黄卓等人针对容器镜像构建效率低下的问题，设计了一种容器镜像快速构建系统。在ASE 2019会议上，吴月明等人针对现有安卓恶意软件检测技术精确度不高和开销大等问题，提出一个基于社交网络分析的安卓市场级别的恶意软件扫描系统。

在TOCS期刊中，石宣化等人针对大数据处理系统托管执行环境的内存管理问题，设计了一种基于对象生命周期的内存管理系统。在TDSC期刊中，丁晓峰等人针对加密大数据的top-k查询问题，提出了一种多关键字top-k搜索方案。在TOS期刊中，张宇等人针对并发图分析任务面临的高额数据访问开销问题，提出了一种关联性感知的并发图处理系统。在TPDS期刊中，金海等人针对传统数据处理系统无法适应小而短的微批作业需求，提出了批流系统慢节点预调度机制；黄禹等人针对流数据集的高频键值随时间变化导致的流处理系统负载均衡问题，提出了一种新颖的负载均衡机制；华强胜等人针对当前流图划分难以并行化的问题，提出一种基于纯策略博弈的边图划分算法；张凡等人针对现有事件流通信策略分配中存在的计算或通信复杂度高的问题，设计了一种高效的事件流通信策略分配算法。在TACO 期刊中，王孝远等人针对目前的大页内存管理机制在异构内存管理中出现的DRAM容量和带宽资源浪费问题，提出一种新型的异构内存管理机制；周放等人针对内存数据持久化的性能问题，提出了一种基于硬件级检查点技术的高效内存数据持久化机制；蒋文斌等人针对深度学习系统GPU内存受限问题，提出一种基于层的内存复用与优化方法。在TSC期刊中，袁斌等人针对软件定义网络中数据层的流表溢出攻击问题，提出了一种服务质量可感知的peer-support策略。在TCC期刊中，姚德中等人针对移动云计算中最佳QoS服务提供商选择难问题，提出一种众包服务平台QoS保证机制；刘海坤等人针对云平台镜像管理复杂、存储开销大等问题，提出一种基于查询定义的虚拟机镜像自动定制机制。在TBD期刊中，丁晓峰等人针对大规模数据隐私策略的Skyline计算开销大的问题，提出了一种基于MapReduce的并行Skyline计算算法。在Proceedings of the IEEE期刊中，林立等人总结了异构多层次计算架构下计算迁移在任务划分、分配和执行等方面存在的挑战和解决方案。在IEEE Internet of Things Journal期刊中，邹德清等人针对云取证方案存在的粒度粗、拓展性差等问题，提出了一个结合连续内存镜像分析与动态污点追踪的隐私侵犯取证系统。在JSAC的期刊中，顾琳等人针对NFV服务的效用最大化问题，基于强化学习提出了一种深度确定性策略梯度的改进算法降低了服务的端到端时延和服务开销。

本期季刊将选取实验室2019年部分代表性学术论文进行内容简述。回首2019，我们硕果累累，上述高水平学术成果是对各位老师和同学辛勤付出的肯定。展望2020，让我们不忘初心，励志前行！

石宣化

2019年12月31日

主　编：金　海
本期执行主编：石宣化
编　委：陈汉华、代炜琦、丁晓锋、耿　聪、顾　琳、胡　侃、华强胜、黄　宏、蒋文斌、廖小飞、刘方明、刘海坤、刘英书、陆　枫、马晓静、羌卫中、邵志远、石宣化、王多强、吴　松、肖　江、谢　夏、徐　鹏、叶晨成、余　辰、袁　斌、袁平鹏、张　腾、张　宇、章　勤、赵　峰、郑　龙、郑　然、邹德清
（按姓氏拼音排序）
责任编辑：刘　芹

地　址：武汉市华中科技大学东五楼二楼
邮　编：430074
电　话：（027）87541924 或 87543529
传　真：（027）87557354
E-mail：liuqin19@hust.edu.cn
Homepage：http://grid.hust.edu.cn
（此刊仅供内部交流学习）

卷首语

热点

封面人物

专栏

声音

动态

推荐

交流

用线性代数语言实现图算法：GraphBLAS

（张晓辉 整理）

在大数据和人工智能时代，图算法被广泛应用于各个领域，包括社交网络、情报分析、物流优化等，这些算法可以检测到大量图中存在的微小模式。随着技术的进步，新的硬件层出不穷，GPU、TPU、FPGA令人眼花缭乱，在这些新的硬件平台实现图算法是一项艰巨的任务。

有研究者认为，当前通过线性代数的方法来实现图算法的技术已经足够成熟，可以用来发展一组标准的基本原语，将图算法的实现与日益复杂的硬件隔离，从而为开发者提供便利。受基本线性代数子程序（Basic Linear Algebra Subprograms, BLAS）启发，他们将这种标准命名为GraphBLAS。这些工作得到了许多机构的支持，包括麻省理工学院林肯实验室、劳伦斯伯克利国家实验室、太平洋西北国家实验室和桑迪亚国家实验室、IBM、英特尔、加州大学圣巴巴拉分校、加州大学伯克利分校、加州大学戴维斯分校、印第安纳大学等。

GrapBLAS背后的原理是，当用邻接矩阵来表示图时，可以用稀疏矩阵–向量乘法来表示广度优先搜索的计算过程。将线性代数计算中涉及的标量操作进行一般化来定义半环（semiring），进而扩展这些原语的范围，以支持更广泛的并行图算法。

2019年9月25日，GraphBLAS论坛正式发布了1.3.0版C语言实现的API（http://people.eecs.berkeley.edu/~aydin/GraphBLAS_API_C_v13.pdf）。目前，依据GraphBLAS开发实现的项目主要有：MPI/C++ Combinatorial BLAS (CombBLAS)、Java Graphulo、Matlab/Octave D4M、GPI (IBM) and IBM GraphBLAS、GraphPad (Intel)、GraphBLAS Template Library (SEI/Indiana)、SuiteSparse Graph BLAS (Texas A&M)、GraphBLAST (UC Davis and LBNL)。

谷歌用强化学习优化量子门控制策略

（王烨飞 整理）

实现量子计算机的最主要挑战就是其最基本的组成部分：量子比特。然而量子比特可以和任意携带能量、且足够近的物体进行相互作用，这些都可能造成量子比特状态无法预测的改变。更复杂的问题是，用来控制量子比特的工具也会带来很多挑战，例如超导回路等。这些电子控制的缺点是会产生白噪声，此外对于来自外部辐射源的干扰以及数模转换器，它们甚至会引入更多的随机误差，从而降低量子回路的性能。

量子计算机固然有很多独特的优势，同时量子计算机更需要严格控制容错、信息丢失等误差，很难手动模拟排除，那么我们为什么不用机器学习来学习并控制它们呢？

近日谷歌的研究者就提出了用深度强化学习极大提升量子计算的性能。谷歌在 Nature 合作期刊《npj Quantum Information》上发表了一篇论文，提出结合深度强化学习的方法来实现通用量子控制，在门控制的精确度和速度上有了数量级的提升，从而能够极大地提高量子计算机的计算能力。 文章提出了一种使用深度强化学习生成的新的量子控制框架，其中可以通过单个控制成本函数来概括量子可控制优化中的各类实际问题。与标准随机梯度下降的解决方案相比，该框架可将量子逻辑门的平均误差最多降低两个数量级，并且大幅降低了来自最优门生成的副本的门时间。

其创新之处在于对量子控制函数的改进以及提出的基于深度强化学习的高效优化方法。

在量子计算过程中准确地评估泄漏信息的常见做法是，一开始就模拟整个计算。然而，这并不利于达成构建大规模量子计算机的目的，因为量子计算机的优势就在于它们能够执行经典系统所无法执行的计算。谷歌研究人员通过使用改进后的物理模型，能够让通用的成本函数对逐渐增加的泄漏误差、控制边界条件

的违背情况、总的门时间和门保真度进行联合优化。

创建了新的量子控制成本函数后，下一步就是应用高效的优化工具将该函数最小化。经证实，现有的优化方法无法找到对于控制波动同样具有鲁棒性的令人满意的高保真度解决方案。相反地，谷歌研究人员则采用同步策略的深度强化学习（RL）方法，即置信域强化学习（Trusted-Region Rl），因为该方法在所有基准问题中均表现出良好的性能，对样本噪声具有固有的鲁棒性，并且能够优化有着数亿个控制参数的数百种高难度的控制问题。

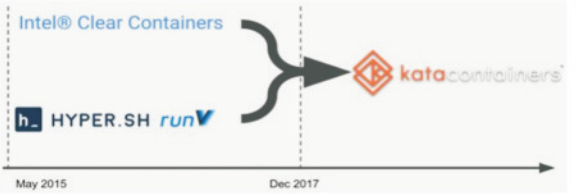
这项工作为使用近期量子设备来开展量子仿真、量子化学和量子霸权测试开启了更加广阔的应用空间，也凸显了使用创新性机器学习技术和能够利用通用量子控制方案的灵活性和附加计算能力的近期量子算法的重要性。进一步，该领域的研究者还需要做更多的实验来将机器学习技术（就比如说我们在这项工作中开发的技术）整合到实际的量子计算过程中，从而利用机器学习来充分提高量子计算机的计算能力。

Kata Containers

（王杰章 整理）

1 、 kata-containers 是什么

kata containers 是由 OpenStack 基金会管理，但独立于 OpenStack 项目之外的容器项目。它是一个可以使用容器镜像以超轻量级虚机的形式创建容器的运行时工具。 kata containers 整合了 Intel 的 Clear Containers 和 Hyper.sh 的 runV ，能够支持不同平台的硬件（ x86-64 ， arm 等），并符合 OCI (Open Container Initiative) 规范，同时还可以兼容 K8s 的 CRI （ Container Runtime Interface ）接口规范。目前项目包含几个配套组件，即 Runtime ， Agent ， Proxy ， Shim ， Kernel 等。目前 Kata Containers 的运行时还没有整合，即 Clear containers 和 runV 还在独立的组织内。



如果大家了解 Docker 技术，就会知道，真正启动 Docker 容器的命令工具是 RunC ，它是 OCI 运行时规范 (runtime-spec) 的默认实现。

Kata containers 其实跟 RunC 类似，也是一个符合 OCI 运行时规范的一种实现（即 Clear Container 和 runV 都符合 OCI 规范），不同之处是，它给每个容器（在 Docker 容器的角度）或每个 Pod （ k8s 的角度）增加了一个独立的 linux 内核（不共享宿主机的内核），使容器有更好的隔离性，安全性。

关于 OCI 的相关介绍，可以参见开发者社区中的另一篇文章： OCI 的前世今生基于 kata containers 创建的容器兼具虚机和容器的优点，在隔离性上，跟常规虚机类似，但在部署时 间和运行效率上却与常规容器相近。

2 、在容器生态系统中的位置

根据上面的内容，可能很多人还是不清楚 Kata Containers 到底是用在哪里的。接下来看一下 Kata containers 在容器生态系统中的位置。

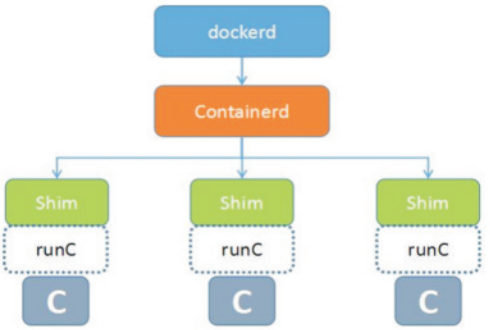
首先我觉得有必要介绍一下，什么是容器的运行时，很多情况下，大家都把 docker 或者 rkt 作为容器的运行时，而在 OCI 中，将 runC 称作运行时。因此，大家都很困惑到底什么是容器运行时。

容器运行时是一个相对的概念，比如，从 k8s 的角度看，直接创建容器的组件是 docker 或 containerd ，因此，将 docker 、 containerd 以及新加入的 CRI-O 作为容器运行时组件。

而在 docker 、 containerd 或 CRI-O 的角度看，真正启动容器的组件是 runC ，因此，docker 中将 runC 作为容器运行时工具，当然在 docker 中， runC 可以被替换，比如可以替

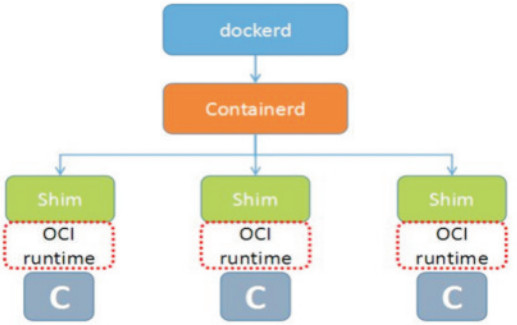
换为本文介绍的 kata containers（即 clear Container 或者 runV）。

接下来，通过几个结构图来说明下 kata containers 在 docker 以及 k8s 中的位置。首先看一下 docker 目前的组件关系图：



我们看到 runC 处于 docker 组件图的最底端，runC 下面就是容器。目前 docker 已经不是一个专一的容器管理组件，而真正的容器管理组件是 containerd，而 containerd 本身也不会直接跟操作系统交互去创建、删除容器，而是借助 runC 来对容器生命周期进行管理。因此这里可以讲 runC 作为容器运行时。

容器圈中，针对容器运行时指定了 OCI 规范，因此，实际上 Docker 的架构已经变成如下结构：



如红色虚线框内所示，也就是说，只要符合 OCI 规范的运行时工具，都可以被 docker（或者说是 containerd）使用。了解上面的内容，我想 kata containers 在容器的什么位置，应该就显而易见了。它符合 OCI 运行时规范，因此，可以作为 runC 的替代组件。下面是 k8s 组件的结构图：

注：CC 表示 clear containers，CC 和 runV 有一个虚线框圈起来，表示后续这两个组件会合并成 kata containers，目前可以分别使用。上图中给出了 k8s 分别将 docker、containerd 和 CRI-O 作为容器管理工具（以 k8s 角度的容器运行时）的组件关系图。k8s 为了能对接多种容器管理工具，抽象了 CRI 接口，每个容器管理工具只需实现接口即可，详细内容参见《我们还需要 docker 吗？》。每个容器管理工具的最后端直接操作容器的就是上一节介绍的符合 OCI 的容器运行时（以 docker 等的角度看），那 kata containers 就是符合 OCI 容器运行时的其中之一。根据以上分析，我们可以看出，kata containers 其实不会替换 docker，也不会替换 containerd，更不会替换 k8s，而跟他们是协作关系，只是其中的一个组件。

3、包含的组件及其功能介绍

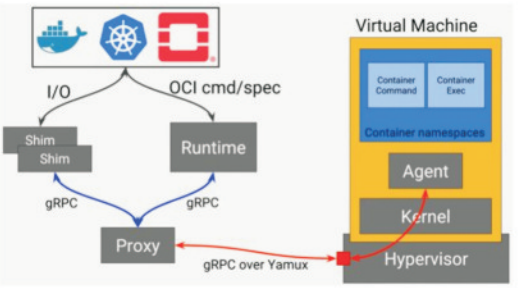
runtime：符合 OCI 规范的容器运行时命令工具，主要用来创建轻量级虚机，并通过 agent 控制虚拟机内容器的生命周期。目前 kata containers 还没有一个统一的运行时工具，用户可以选择 clear containers 和 runV 中的其中之一。

agent：运行在虚机中的一个运行时代理组件，主要用来执行 runtime 传给他的指令，在虚机内管理容器的生命周期。

shim：以对接 docker 为例，这里的 shim 相当于是 containerd-shim 的一个适配，用来处理容器进程的 stdio 和 signals。shim 可以将 containerd-shim 发来的数据流（如 stdin）传给 proxy，然后转交给 agent，也可以将 agent 经由 proxy 发过来的数据流（stdout/stderr）传给 containerdshim，同

时也可以传输 signal 。这里的 shim 跟上一节 k8s 结构图中的 shim 不是一个组件，上节提到的 shim 表示 containerd-shim proxy ：主要用来为 runtime 和 shim 分配访问 agent 的控制通道，以及路由 shim 实例和 agent 之间的 I/O 数据流。

kernel ： kernel 其实比较好理解，就是提供一个轻量化虚机的 linux 内核，根据不同的需要，提供几个内核选择，最小的内核仅有 4M 多。下图是 kata containers 官方给出的各组件之间的关系。详细原理还有待进一步研究。



4 、现状以及后续的计划

kata containers 项目刚起步，还没有完整的一套组件供用户使用，只是提出了一套完整的架构。但是从架构中看，跟 clear containers 目前采用的架构基本一致，包括架构图以及组件类型。因此，目 前针对 kata containers 的研究，可以通过 clear containers 入手。

目前， kata containers 项目中，已经增加了包括 agent ， shim ， proxy 等组件的代码库，但是还没有针对 runtime 的代码库。在后续的计划中， kata containers 将分为三步来实现 clear containers 和 runV 两个项目的融合：

首先，在 katacontainer 中兼容 clear containers 和 runV 两个运行时 整体采用 kata containers 的架构 clear containers 和 runV 两个运行时可以无缝对接 kata containers 的其他组件 用户可以在两个运行时之间来回切换

其次，将 clear containers 和 runV 合并为一个，形成 kata containers 自己的 runtime 最后，废除 clear containers 和 runV 的兼容。

大数据时代，你离“隐私被泄露”有多远?

（刘仁婉 整理）

大数据时代，你离个人隐私被泄露有多远？答案是，近在咫尺。

——前言

近日，有媒体报道称，某社交平台涉嫌监控用户日常沟通记录，其投放的广告内容也与用户隐私内容密切相关。平台很快否认了指责，但不论它的回应是否有脱罪之嫌，都至少向大众证实了一个事实：互联网用户的个人网络行为，普遍会受到外部渠道的“暗中观察”。

基于这种观察，用户的每一次网络活动，都将成为大数据的一部分，并以此为针对自己的“画像”提供素材。就这样，用户在不知不觉间，被精准圈入到了一个个标签人群中，等待接受“个性化信息”的投喂。

个人数据被泄露——由来已久且愈演愈烈

去年3月曝光的Facebook数据泄露事件中，有5000万用户的个人资料，被用来作为向其精准投放政治广告的重要参考，这些人占据着美国选民人数的四分之一。可以说，此举或许直接影响了此前的总统大选结果。同年还有多起大规模数据泄露事件发生——万豪发公告称旗下酒店喜达屋5亿房客信息被泄露; 社交平台陌陌的3000万用户数据在暗网被销售；问答网站鼻祖Quora的1亿用户数据被窃等。

然而，基于群体用户个人数据被整理之后的“画像”行为，其争议之处在于，它除了可能造成用户被迫接受指定信息外，精准的推送也为用户提供了实实在在的便利。这也正是很多人心甘情愿奉上个人数据的原因所在。几年前的一份《中国个人信息安全和隐私保护报告》显示，当被问及是否“愿意提供个人信息以获得更便利的服务享受”时，53%的人选择了“愿意”。在大数据技术被更加深入和广泛使用的今天，这一比例势必还在进一步扩大。在这种“一个愿打一个愿挨”的情况下，很多互联网平台也就更加有恃无恐的，不断试探着用户关于个人隐私的底线。

合法标准如何制定，相关规范又能从何做起？

既然被画像不可避免，那么衡量对个人数据的采集是否涉及侵害个人隐私的依据，就是这一行为上的度。因此，需要有一个严格而清晰的“标准”存在。但遗憾的是，由于每个人对隐私的理解都有所不同，加上不同领域的“画像”对数据采集的要求各异，因此“标准”就只能制定在一个相对宽泛和抽象的范围内。这也直接给相关平台提供了钻空子、打擦边球的灰色地带。

当然，在这一过程中，企业未必就是此类恶性事件的始作俑者，相反可能还会是直接损失最大的一方。比如在Facebook的数据泄露事件中，利用用户数据干预大选的是一家英国公司，但Facebook却不得不因自身的监管不力，吐下市值蒸发百亿，创始人扎克伯格被传唤至国会接受质询的苦果。

需要警惕的是，正如上述案例所示：大数据时代，对于个人数据的非法采集和泄露，除了直接关乎用户的个人隐私、财产和人身安全外，已经直接上升到了国家安全的范畴。

那么，如何在最大范围和程度上实现对个人数据的有效保护？

首先，在国家层面上，我国针对网络安全和个人数据安全的保护，早已有所行动。《中华人民共和国网络安全法》、《信息安全技术个人信息安全规范》等相关法律法规都已陆续出台。但随着个人数据安全保护形势的不断严峻，还需要立法和行政机构在实体和程序上，制定出台更加详尽的法规和条例，同时也需要各级监管部门加大执法力度和惩罚力度。

对于企业自身而言，除了要严格遵守采集用户个人数据的相关规定外，也要加强对第三方获取用户数据目的的审查和使用用户数据行为的监管。但由于一般企业的商业属性，令其在执行上述操作的自觉性上难以得到保障。加上目前国内掌握大规模用户数据的互联网企业，在经过前些年的千帆竞流后，已经愈发表现出在各自领域的“寡头”潜质。绝对的体量，企业和用户间悬殊的力量对比，会直接造成“店大欺客”现象的发生。想要一劳永逸地杜绝企业层面上大规模个人数据泄露的发生，历史上美国最高法院的一起案例，或许就具有很好的参考价值。在1911年，美最高法一纸判决将当时的标准石油公司，拆解成34家地区性石油公司，从而终结了美国石油寡头的垄断行为。这种遏制垄断行为的方法，对于目前国内的某些一家独大的互联网公司来说，可能同样适用。

此外，用户个人也要提高对第三方应用的警惕，加强个人信息安全保护意识，规范自己的网络行为。不过最后我们必须要明确的一点是，尽管大数据技术的应用带来了大规模个人数据泄露的隐患，但我们不能因噎废食般就此否定它的价值。技术本身没有对错之分，每一个技术的产生也都不带有原罪，我们要规范的不是技术，而是使用技术的人。

（链接：http://192.168.73.134/www.sohu.com/a/304681862_386566 ）

基于GPU的压缩图遍历

（杨　词　整理）

应用拥有大量内核和超大内存带宽的GPU 作为加速图形处理平台成为了热点。但是当图大小超过GPU内存时，GPU内存的不可拓展性以及CPU与GPU之间高成本的数据传输，限制了GPU图形处理的能力。目前在这个问题上有一些杰出的工作（核外策略，多GPU策略，分布式GPU策略等），这些工作展示了GPU处理大于内存图的可行性，但是仍存在GPU内存成本高等问题。

新加坡国立大学Kian-Lee Tan教授团队在SIGMOD 2019上发表的论文介绍了基于GPU的压缩图遍历方法（GCGT），使得GPU可以直接在内存遍历压缩图结构（CGR）。该方法分配线程独立处理压缩邻接表，提出两阶段遍历策略以消除线程发散，设计任务窃取机制处理线程间负载均衡问题，充分利用GPU的计算资源。此外，针对遵循幂律分布的现实世界图，该方法提出了

wrap-centric解码方法提升解码效率，使用残差分割遍历处理幂律图中的残差长序列，打破残差之间的顺序依赖性以进行并行处理，缓解负载不平衡问题。测试表明，与Ligra+、Gunrock等成熟图处理系统相比，GCGT具有良好的图压缩率，在社交、网络等各类图数据集上能取得合理的性能。

龙芯将于12月发布最新CPU

（张朝钦 整理）

据龙芯中科公众号消息，宣布其2019产品发布暨用户大会将于12月24日在国家会议中心开启，届时会推出龙芯新一代处理器架构产品。

龙芯称此次大会是龙芯面向全行业的年度盛会，旨在构建以龙芯CPU为核心的自主创新产业生态，分享龙芯近20年积累的技术与能力，展示基于龙芯平台的最新成果、解决方案及应用案例。

据推测，此次发布的全新架构处理器产品应该是龙芯3A4000处理器，此前有资料显示它依然是28nm工艺，频率从龙芯3A3000的1.5GHz提升到了2.0GHz，架构升级为GS464V，搭配的芯片组也升级到了龙芯7A2000，28nm工艺，并在安全及加密上做了加强，增加了安全模块，支持多种解密加密算法。龙芯3A4000较3A3000相比，除大幅优化提升计算模块外，更在芯片内部增加安全模块，支持多种加解密算法。从CPU层面确保真正的“底层安全”。

龙芯是基于MIPS架构的国产处理器，现在已经被广泛运用于通用领域、嵌入式领域以及安全应用领域当中，其中作为嵌入式CPU被运用于工控设备的情况比较多，甚至还被安装进北斗卫星里面。

HyperFaaS：一种弹性serverless框架

（陶志恒 整理）

最近，一种称为serverless的新计算范式变得越来越流行。serverless将传统的应用程序模型分解为更细粒度的函数来提供服务，开发人员可以专注于功能逻辑，而完全无需担心服务器或虚拟机的管理。serverless仍处于起步阶段，现有的serverless计算基础架构开始呈现出一系列问题。最突出的问题是运行时开销，简单采用容器引擎（如Docker）带来了显著的容器启动开销和运行环境设置开销，使得serverless无法满足高并发的突发负载的延时和吞吐量要求。

NSDI上的文章<HyperFaaS: A Truly Elastic Serverless Computing Framework>提出一种称为HyperFaaS的新颖的serverless框架来解决这个问题。HyperFaaS的目标是：1）通过层级调度和容器共享最大化资源利用率和利用效率；2）通过临时资源扩展最小化高并发的突发函数的性能损失。

层级调度：容器外部有全局调度器GS（Global Scheduler），容器内有局部调度器ICS（In-container Scheduler）。ICS负责分发请求到函数实例，并监控函数的QoS（Quality-of-Service），周期性地与GS交换信息。当函数的QoS达到设定的水位时，ICS通知GS采用行动，过量的请求将由GS寻找可用容器后重新调度。这种层级调度可以有效地避免容器过载，从而显著提升响应延时。

多阶段自动扩容：为有效地处理突发负载，HyperFaaS设计了一种“突发–缓冲”方式的扩容策略。首先先扩展现有的容器，期望负载只是临时到来，持续时间极短；如果负载继续持续存在，请求将被GS调度到其他资源富裕的容器；如果现有的容器资源依然不足，GS会异步通过冷启动创建新容器，在创建完成后GS将请求重新调度到新容器。通过多阶段扩容，可以使得请求的延时曲线变得更加平滑。

容器共享：HyperFaaS采用资源复用的方式在多函数间实现容器共享，这是多阶段自动扩容的关键技术。一个容器中同时存在两个函数，每个函数都有两种状态：通用态和专用态，并且可以进行状态切换。容器负载较小时，一个函数装载代码后变为专用态提供服务；当突发请求到来，QoS超过容器设定水位时，GS寻找负载小的容器，将另一个通用态函数转变为专用态提供服务。

我的科研生活复盘

前　言

转眼已毕业一年有余，收到作为本期的封面人物分享经历的邀请，十分惊喜与荣幸。回顾在SCTS/CGCL实验室度过的三年，满是感恩与怀念。借此机会复盘一下研究生的三年，也向实验室的同学们分享一点经验。

快速进入角色

现在读研的理由有很多，已不仅仅是单纯的学术追求，有些人跟随大流，大家都读研那我也要读研；有些人为了能够找到更好的工作，拿到更高的工资而读研；有些人为了逃避步入社会而选择读研。不管是因为什么原因，既然选择了读研，首先需要做到的就是快速进入角色。不同于本科阶段的被动接受书面知识，研究生阶段需要主动发现问题，寻求知识，老师不再是知识的传授者而是研究方向的掌舵者。我们不能等着老师给分配课题，等着老师告诉我们去怎么做，而是需要在老师确定研究方向后，自主地阅读相关文献，思考可行且有意义的课题，然后与老师讨论，让老师把关。记住自己才是科研的主导者，如果不主动地积极思考探索，一味地等着被动的分配任务，那么科研之路只会举步维艰。

刚进入小组的时候，导师刘方明老师考虑到我的动手能力强，可以做一个系统类型的富有挑战性的课题。刘老师当时正在关注FPGA等异构硬件在分布式系统与网络领域中的应用，希望结合硬件性能加速和灵活可配置等特性去克服网络功能虚拟化的性能瓶颈。经过组会讨论后，大家都认可了这个研究课题的可行性。至此我算是进入了研究生的角色，开始大量阅读文献，分析当时工业界和学术界的相关研究现状。

有些事情躲不掉的

确定做FPGA之后，一个个的难题便接踵而至。读研之前我从没接触过FPGA，更没接触过硬件编程语言，既然选择了做FPGA这个方向，那就只能硬着头皮上。首先学习FPGA的工作原理，只有深刻理解FPGA的工作原理，深刻理解其优势，才能想出具体怎么将其与网络功能虚拟化相结合，想出好的idea。当时查阅了很多的英文网站、以及FPGA相关的论文，从无到有将FPGA理解透彻，最终与老师一起讨论出论文的idea。有了idea之后，接下来便是要实现相应的系统，用实验数据说话。FPGA不是通过软件语言（如C、Java）来编程，有很高的编程门槛，当时就只能自学硬件编程语言。将FPGA与网络功能软件相结合一起协同工作，需要编写相关设备驱动，组里的小伙伴们也都没有这方面的经验，又只能自学Linux设备驱动的编写。当完成驱动后，可以正常操作FPGA了，又出现了新的问题：使用FPGA给网络功能加速，需要选择基于什么实现的网络功能模型呢？最终基于高性能考虑，选择了DPDK，然后继续研究DPDK的源码并改写之前完成的驱动，移植到DPDK框架中。论文第一次投稿被拒，评审们指出实验环节的测试用例太少，这个是在预料之中，当时因为对FPGA不熟悉，在FPGA上开发代码的难度大、周期长，因此只实现了一种功能。收到评审意见后，增加更多的网络功能作为实验

用例成为了必须越过的坎，这时又和王秀秀学妹一起加班加点实现了另外两种网络功能。在克服了一个又一个的技术难点、经过与老师和组员们一次又一次的修改之后，最终论文被分布式系统领域的著名国际学术会议IEEE ICDCS 2018 会议录用。

回想这篇论文从无到有，直到最终发表，难题总是一个接一个出现，与其躲避，不如直面和解决问题。科研就是这样，时不时出现一个“惊喜”，如果我们被“惊喜”吓退而选择逃避，那么科研之路就真的“永无止境”。我也十分感恩研究FPGA这段经历，极大地锻炼了我学习新知识和面对新问题的能力。

做学问要耐得住寂寞

科研生活比想象的难度大，需要大量地阅读文献，需要绞尽脑汁地思考创新点，需要反复地进行实验，需要撰写出一篇英文论文，更需要逐字逐句地反复推敲修改。每一篇被高水平会议或期刊接收的论文背后，都少不了作者们奋斗的日夜。

与本科时丰富多彩的课余生活比起来，三年的研究生时光虽然略少色彩，但却有更加深刻的成长。那些东五楼大爷催促要锁门的唠叨，那些周末不回家去实验室看论文的日子，那些在走廊踱步思考如何论证观点的下午，那些听闻其他的小伙伴论文被录用后高兴而焦虑的时刻，还有那些因为做实验改论文而熬的夜，现在还历历在目。可能依旧没有想出好的idea，可能实验结果仍旧不太理想，可能投稿再一次被拒，但这些对于科研道路的成长是必不可少的，希望大家都能耐得住寂寞，不辜负自己的美好年华。

人人为我，我为人人

“人人为我，我为人人”是刘老师经常教育我们的话。一方面，论文能够被ICDCS会议录用不但有刘老师的悉心指导，自身的不断努力，更少不了组里小伙伴们的帮助。感谢组会中小伙伴们对我论文严格地“挑刺”，感谢大家不厌其烦地帮我理清思路、审阅论文和查错勘误，正是在大家的帮助下我才能够快速地成长。另一方面，我给其他组员们的论文“挑刺”时也锻炼了我发现问题和找出痛点的眼光，帮组员们搭建服务器和搭建实验环境也锻炼了我的动手能力，小组内互帮互助的良好氛围，既给科研生活带来了生机和温暖，也在无形之中提升了大家的能力。

尾　声

在实验室的三年是难忘而珍贵的三年，不仅仅是因为发表了一篇高水平学术论文和获得了硕士学位，更是因为有着刘老师认真负责的悉心指导，有着一群为了科研而共同努力奋斗的小伙伴们。虽然已经工作一年多，成为了一名平凡的码农，科研已渐渐离我远去，但是非常感谢刘老师领我在学术殿堂游览一番，一睹殿内的精彩。最后，祝愿实验室的小伙伴们都能拥有自己的精彩科研人生。

李肖瑶
2015级硕士研究生
研究方向：网络功能虚拟化与性能加速
Email：calmisi.lee@gmail.com

2019 年实验室学术成果一览表

序号	第一作者	文　章
1	张　宇	Yu Zhang, Xiaofei Liao, Hai Jin, Bingsheng He, Haikun Liu, and Lin Gu, "DiGraph: An Efficient Path-based Iterative Directed Graph Processing System on Multiple GPUs", In Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2019), April 13-17, 2019, Providence, RI, USA, pp.601-614
2	高　彬	Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu, "Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing", In Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019), April 29-May 2, 2019, Paris, France, pp. 1459-1467
3	张启夏	Qixia Zhang, Fangming Liu, and Chaobing Zeng, "Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices", In Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019), April 29-May 2, 2019, Paris, France, pp. 2449-2457
4	陈姝彤	Shutong Chen, Lei Jiao, Lin Wang and Fangming Liu, "An Online Market Mechanism for Edge Emergency Demand Response via Cloudlet Control", In Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019), April 29-May 2, 2019, Paris, France, pp. 2566-2574
5	姜炜祥	Weixiang Jiang, Ziyang Jia, Sirui Feng, Fangming Liu, and Hai Jin, "Fine-grained warm water cooling for improving datacenter economy", In Proceedings of the 46th International Symposium on Computer Architecture (ISCA 2019), June 22-26, 2019, New York, USA, pp.474-486
6	金盼盼	Panpan Jin, Jian Guo, Yikai Xiao, Rong Shi, Yipei Niu, Fangming Liu, Chen Qian, and Yang Wang, "Postman: rapidly mitigating bursty traffic by offloading packet processing", In Proceedings of the 2019 USENIX Annual Technical Conference (ATC 2019), July 10-12, 2019, Renton, WA,USA, pp.849-862
7	王飞越	Feiyue Wang, Hanhua Chen, Liangyi Liao, Fan Zhang, Hai Jin, "The Power of Better Choice: Reducing Relocations in Cuckoo Filter", In Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS 2019), July 7-10, 2019, Dallas, Texas, USA, pp.358-367
8	袁平鹏	Pingpeng Yuan, Longlong Lin, Zhijuan Kou, Ling Liu, and Hai Jin, "Big RDF Data Storage, Computation, and Analysis: A Strawman's Arguments", In Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS 2019), July 7-9, 2019, Dallas, Texas, USA, pp.1693-1703.
9	张　凡	Fan Zhang, Hanhua Chen, Hai Jin, "Simois: A Scalable Distributed Stream Join System with Skewed Workloads", In Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS 2019), July 7-10, 2019, Dallas, Texas, USA, pp.176-185
10	陈　琼	Qiong Chen, Zimu Zheng, Chuang Hu, Dan Wang and Fangming Liu, "Data-driven task allocation for multi-task transfer learning on the edge", In Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), July 7-9, 2019, Dallas, Texas, USA, pp. 1040-1050
11	顾　琳	Lin Gu, Deze Zeng, Wei Li, Song Guo, Albert Zomaya and Hai Jin, "Deep Reinforcement Learning based VNF Management in Geo-distributed Edge Computing", In Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), July 7-9, 2019, Dallas, Texas, USA, pp. 934-943

序号	第一作者	文　章
12	吴　松	Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, Jiang Xiao and Fei Chen, “When FPGA-accelerator meets Stream Data Processing in the Edge”, In Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), July 7-9, 2019, Dallas, Texas, USA, pp. 1818-1829
13	戴小海	Xiaohai Dai, Jiang Xiao, Wenhui Yang, Chaofan Wang and Hai Jin, “Jidar: A Jigsaw-like Data Reduction Approach Without Trust Assumptions for Bitcoin System”, In Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), July 7-9, 2019, Dallas, Texas, USA, pp. 1317-1326
14	黄　航	Hang Huang, Jia Rao, Song Wu, Hai Jin, Kun Suo, and Xiaofeng Wu, “Adaptive Resource Views for Containers”, In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC 2019), June 22-26, 2019, New York, NY, USA, pp.243-254
15	赵　进	Jin Zhao, Yu Zhang, Xiaofei Liao, Ligang He, Bingsheng He, Hai Jin, Haikun Liu, and Yicheng Chen, "GraphM: An efficient storage system for high throughput of concurrent graph processing", In Proceedings of the 31st International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2019), November 17-19, 2019, Denver, Colorado, USA,pp.401–412
16	周舜杰	Shunjie Zhou, Fan Zhang, Hanhua Chen, Hai Jin, Bing Bing Zhou, "FastJoin: A Skewness-Aware Distributed Stream Join System", In Proceedings of the 33th International Parallel and Distributed Processing Symposium (IPDPS 2019), May 20-24, 2019, Rio de Janeiro, Brazil, pp.1042-1052
17	樊　浩	Hao Fan, Song Wu, Shadi Ibrahim, Ximing Chen, Hai Jin, Jiang Xiao and Haibing Guan, “NCQ-Aware I/O Scheduling for Conventional Solid State Drives”, In Proceedings of the 33th International Parallel and Distributed Processing Symposium (IPDPS 2019), May 18-22, 2019, Rio de Janeiro, Brazil, pp.523-532
18	黄　卓	Huang, Zhuo, Song Wu, Song Jiang, and Hai Jin, “FastBuild: Accelerating Docker Image Building for Efficient Development and Deployment of Container”, In Proceedings of the 35th International Conference on Massive Storage Systems and Technology (MSST 2019), May 23-24, 2019, Santa Clara, California, USA, pp. 28-37
19	吴月明	Yueming Wu, XiaoDi Li, Deqing Zou, Wei Yang, Xin Zhang, and Hai Jin, “MalScan: Fast Market-Wide Mobile Malware Scanning by Social-Network Centrality Analysis”, In Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE 2019), November 12-14, 2019, San Diego, California, United States
20	石宣化	Xuanhua Shi, Zhixiang Ke, Yongluan Zhou, Hai Jin, Lu Lu, Xiong Zhang, Ligang He, Zhenyu Hu, Fei Wang,"Deca: a garbage collection optimizer for in-memory data processing", ACM Transactions on Computer Systems, vol.36(1), no. 3, pp.3:1-3:47,2019
21	张　宇	Yu Zhang, Jin Zhao, Xiaofei Liao, Hai Jin, Lin Gu, Haikun Liu, Bingsheng He, and Ligang He, "CGraph: A distributed storage and processing system for concurrent iterative graph analysis jobs". ACM Transactions on Storage, vol.15,no.2,pp.10:1-10:26, 2019
22	蒋文斌	Wenbin Jiang, Yang Ma, Bo Liu, Haikun Liu, Bing Bing Zhou, Jian Zhu, Song Wu, and Hai Jin, " Layup: Layer-adaptive and Multi-type Intermediate-oriented Memory Optimization for GPU-based CNNs",ACM Transactions on Architecture and Code Optimization, vol.16, no.4, pp.39:1-39:23, 2019
23	王孝远	Xiaoyuan Wang, Haikun Liu, Xiaofei Liao, Ji Chen, Hai Jin, Yu Zhang, Long Zheng, Bingsheng He, and Song Jiang. "Supporting Superpages and Lightweight Page Migration in Hybrid Memory Systems." ACM Transactions on Architecture and Code Optimization, vol.16(2), no. 11, pp. 11:1-11:26, 2019.

序号	第一作者	文　章
24	吴　松	Song Wu, Fang Zhou, Xiang Gao, Hai Jin, and Jinglei Ren, “Dual-Page Checkpointing: An Architectural Approach to Efficient Data Persistence for In-Memory Applications”, ACM Transactions on Architectures and Code Optimization, vol.15,no.4, pp.1-27, 2019
25	金　海	Hai Jin, Fei Chen, Song Wu, Yin Yao, Zhiyi Liu, Lin Gu, Yongluan Zhou, “Towards Low-Latency Batched Stream Processing by Pre-Scheduling”, IEEE Transactions on Parallel and Distributed Systems, vol.30, no.3, pp.710-722, 2019
26	华强胜	Qiang-Sheng Hua, Yangyang Li, Dongxiao Yu, and Hai Jin, “Quasi-Streaming Graph Partitioning: A Game Theoretical Approach”, IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 7, pp.1643-1656,2019
27	廖小飞	Xiaofei Liao, Yu Huang, Long Zheng, Hai Jin, “Efficient Time-Evolving Stream Processing at Scale”,IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 10, pp.2165-2178, 2019
28	张　凡	Fan Zhang, Hanhua Chen, Hai Jin, "Piggyback Game: Efficient Event Stream Dissemination in Online Social Network Systems", IEEE Transactions on Parallel and Distributed Systems, vol.30, no.3, pp.692-709, 2019
29	丁晓锋	Xiaofeng Ding, Peng Liu, and Hai Jin,"Privacy-Preserving Multi-Keyword Top-k Similarity Search over Encrypted Data",IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 2, pp.344-357,2019
30	丁晓锋	Xiaofeng Ding, Li Wang, Zhiyuan Shao, and Hai Jin, "Efficient Recommendation of De-Identification Policies Using MapReduce", IEEE Transactions on Big Data, Vol. 5, No. 3, pp.343-354, 2019
31	袁　斌	Bin Yuan, Deqing Zou, Shui Yu, Hai Jin, Weizhong Qiang, and Jinan Shen, “Defending against Flow Table Overloading Attack in Software-Defined Networks”, IEEE Transactions on Services Computing, vol.12, no.2, pp.231-246, 2019
32	刘海坤	Haikun Liu, Bingsheng He, Xiaofei Liao, Hai Jin, “Towards Declarative and Data-Centric Virtual Machine Image Management in IaaS Clouds”, IEEE Transactions on Cloud Computing, vol.7, no.4, pp.1124-1138, 2019
33	姚德中	Dezhong Yao, Chen Yu, Laurence T. Yang, Hai Jin,"Using Crowdsourcing to Provide QoS for Mobile Cloud Computing",IEEE Transactions on Cloud Computing,vol.7,no.2,pp.344-356,2019
34	邹德清	Deqing Zou, Jian Zhao, Weiming Li, Yueming Wu, Weizhong Qiang, Hai Jin, Ye Wu, and Yifei Yang, “A Multigranularity Forensics and Analysis Method on Privacy Leakage in Cloud Environment”, IEEE Internet of Things Journal, vol.6, no.2, pp.1484–1494, 2019
35	林　立	Li Lin, Xiaofei Liao, Hai Jin, and Peng Li,"Computation offloading toward edge computing" ,Proceedings of the IEEE, vol. 107, no. 8, pp: 1584-1607, 2019
36	金　海	Hai Jin, Yi Luo, Chenjing Gao, Xunzhu Tang, Pingpeng Yuan, “ComQA:Question Answering Over Knowledge Base via Semantic Matching”, IEEE Access 2019, vol.7, pp.75235-75246, 2019
37	林　立	Li Lin, Peng Li, Xiaofei Liao, Hai Jin, and Yu Zhang,"Echo: An edge-centric code offloading system with quality of service guarantee" ,IEEE Access, vol. 7, pp: 5905-5917, 2019
38	刘　伟	Wei Liu, Haikun Liu, Xiaofei Liao, Hai Jin, and Yu Zhang, "NGraph: Parallel Graph Processing in Hybrid Memory Systems", IEEE Access, vol. 7, pp.103517-103529,2019
39	李　珍	Zhen Li, Deqing Zou, Jing Tang, Zhihao Zhang, Mingqian Sun, and Hai Jin, “A Comparative Study of Deep Learning-Based Vulnerability Detection System”, IEEE Access, vol.7, pp. 103184–103197, 2019
40	羌卫中	Weizhong Qiang, Weifeng Li, Hai Jin, and Jayachander Surbiryala, “Mpchecker: Use-After-Free Vulnerabilities Protection Based on Multi-Level Pointers”, IEEE Access, vol.7, pp.45961-45977, 2019
41	金　海	Hai Jin, Chen Qian, Dongxiao Yu, Qiang-Sheng Hua, Xuanhua Shi, Xia Xie, “Parallel computation of hierarchical closeness centrality and applications”, World Wide Web: Internet and Web Information Systems,vol.22, no. 6, pp.3047-3064, 2019

序号	第一作者	文　章
42	丁晓锋	Xiaofeng Ding, Wanlu Yang, Kim-Kwang Raymond Choo, Xiaoli Wang, and Hai Jin, "Privacy Preserving Similarity Joins Using MapReduce",Information Sciences, vol. 493, pp.20-33,2019
43	陆　枫	Feng Lu, Lin Gu, Laurence Tianruo Yang, Liwen Shao, Hai Jin,"Mildip: An energy efficient code offloading framework in mobile cloudlets",Information Sciences,vol.513, pp.84-97,2019
44	范学鹏	Xuepeng Fan, Xiaofei Liao, Hai Jin,"FunctionFlow: coordinating parallel tasks",Frontiers Comput. Sci, vol.13,no.1, pp.73-85,2019
45	吕新桥	Xinqiao Lv, Wei Xiao, Yu Zhang, Xiaofei Liao, Hai Jin, Qiang-Sheng Hua,"An effective framework for asynchronous incremental graph processing",Frontiers Comput. Sci. vol.13,no.3, pp.539-551,2019
46	桂创意	Chuangyi Gui, Long Zheng, Bingsheng He, Cheng Liu, Xinyu Chen, Xiaofei Liao, and Hai Jin, "A Survey on Graph Processing Accelerators: Challenges and Opportunities", Journal of Computer Science and Technology, vol. 34, no. 2, pp. 339-371, 2019.
47	李陈希	Chenxi Li, Xiaofei Liao, Hai Jin,"Enhancing application performance via DAG-driven scheduling in task parallelism for cloud center",Peer-to-Peer Networking and Applications, vol.12,no.2,pp.381-391 ,2019
48	石　翔	Xiang Shi, Xiaofei Liao, Dayang Zheng, Hai Jin, Haikun Liu,"VMBKS: a shared memory cache system based on booting kernel in cloud",The Journal of Supercomputing, vol.75,no.1, pp.4-19, 2019
49	沈济南	Jinan Shen, Deqing Zou, Hai Jin, Bin Yuan, and Weiqi Dai, “A domain-divided configurable security model for cloud computing-based telecommunication services”, The Journal of Supercomputing, vol.75, no.1, pp.109–122, 2019
50	Saqib Qamar	Saqib Qamar, Hai Jin, Ran Zheng, Parvez Ahmad," Multi stream 3D hyper-densely connected network for multi modality isointense infant brain MRI segmentation",Multimedia Tools Applications,vol.78,no.18,pp. 25807-25828,2019
51	赵　峰	Feng Zhao, Yu Shen, Xiangyu Gui, Hai Jin, “ SDBPR: Social Distance-aware Bayesian Personalized Ranking for Recommendation", Future Generation Computer Systems, vol.95,no.6, pp. 372-381, 2019
52	陈　俊	Jun Chen, Chen Yu, Hai Jin, “Evaluation model for business sites planning based on online and offline datasets”, Future Generation Computer Systems,vol.91,no.1,pp.465-474,2019
53	顾　琳	Lin Gu, Jingjing Cai, Deze Zeng, Yu Zhang, Hai Jin, Weiqi Dai, “Energy efficient task allocation and energy scheduling in green energy powered edge computing”, Future Generation Computer Systems, vol.39, pp.89-99,2019
54	陆　枫	Feng Lu, Ziqian Shi, Lin Gu, Tianruo Yang, Hai Jin, “An Adaptive Multi-level Caching Strategy for Distributed Database System”, Future Generation Computer Systems, vol.97, pp.61-68,2019
55	邵志远	Zhiyuan Shao, Zhenjie Mei, Xiaofeng Ding, Hai Jin, “BlockGraphChi: Enabling Block Update in Out-of-Core Graph Processing”, International Journal of Parallel Programming, vol.47, no.4, pp.668-685, 2019
56	顾　琳	Lin Gu, Deze Zeng, Sheng Tao, Song Guo, Hai Jin, Albert Y. Zomaya," Fairness-Aware Dynamic Rate Control and Flow Scheduling for Network Utility Maximization in Network Service Chain", IEEE Journal on Selected Areas in Communications, vol.37, no.5, pp.1059-1071, 2019
57	周　放	Fang Zhou, Song Wu, Youchuang Jia, Xiang Gao, Hai Jin, Xiaofei Liao and Pingpeng Yuan, “VAIL: A Victim-Aware Cache Policy to Improve NVM Lifetime for Hybrid Memory System”, Parallel Computing, vol.87, pp.70-76, 2019
58	丁晓锋	Xiaofeng Ding, Yangling Ou, Jianhong Jia, Hai Jin, Jixue Liu, “Efficient subgraph search on large anonymized graphs”, Concurrency and Computation: Practice and Experience, vol. 31, no. 23, pp. 1-14, 2019

说明：文章按实验室Top 80、CCF A类国际学术会议论文、IEEE/ACM Transactions论文和其他SCI期刊排序。

GPU上基于路径的有向图处理系统

张　宇

文章发表在Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2019)上，ASPLOS是综合体系结构、编程语言和操作系统三个方向的计算机系统领域顶级会议，本届会议共收到350篇投稿，录用74篇，录用率约为21%。

内容概述

最近，大量迭代图算法被提出用于分析现实世界中的有向图（例如，蛋白质相互作用网络和社交网等）来服务于各种应用。然而，这些迭代有向图算法在分析大规模有向图时通常十分耗时。因此各应用希望能够减少这些算法的执行时间，以便能够获得实时结果。随着GPU加速器的快速进展，其相对于CPU具有高计算能力和高内存带宽，人们越来越关注在GPU上进行图分析。虽然现有基于GPU的图计算系统已被广泛研究和优化，但这些系统对迭代有向图的处理仍存在重大挑战。现有系统将顶点/边作为GPU核心上的基本并行处理单元，这使得各图顶点难以将其最新状态快速有效地传递给其邻居图顶点，从而导致迭代有向图处理任务面临着图顶点状态传递缓慢和冗余数据处理等问题。

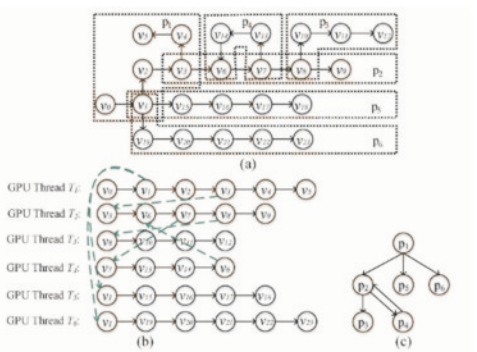


图1. 以路径为中心的执行模型解释

针对这样的问题，我们在GPU上实现了一种有效的基于路径的迭代有向图处理系统DiGraph。如图1所示，它将有向图表示为一组不相交的有向路径，并将该路径作为基本并行处理单元。具体来说，各GPU线程同时处理不同的路径。在每条路径上，每个GPU线程沿着该路径上图顶点的顺序在一轮图处理内依次处理每个图顶点，并且每个图顶点的最新状态被异步发送到其后继图顶点从而在同一轮图处理内更新它们状态，获得更快的顶点状态传递速度。为了减少重复处理路径的开销，它根据路径的依赖关系，为路径构造一个有向无环图（DAG），并根据DAG 的拓扑顺序将路径分配给GPU进行并行处理，其中DAG 的每个图顶点表示一个路径的子集。此外，我们也进一步在每个SM上设计了路径调度策略，根据路径对状态传递的重要度来安排路径处理顺序，使得一些重要路径能够得到有效处理，减少冗余处理开销。实验结果表明，与现有GPU图计算系统相比，本方法能提高迭代有向图算法的性能达3.54倍。

详细内容参见：

Yu Zhang, Xiaofei Liao, Hai Jin, Bingsheng He, Haikun Liu, and Lin Gu. DiGraph: An efficient path-based iterative directed graph processing system on multiple GPUs. In Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 601–614, 2019.

张　宇
副教授
研究方向：大数据系统软件、运行时优化
Email：zhang_yu9068@163.com

移动边缘计算中的网络选择和服务放置协同优化

高　彬，刘方明

文章“Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing”发表在第38届IEEE国际计算机通信会议（INFOCOM）上。该会议是IEEE在通信网络领域的旗舰级会议，由IEEE Communication Society主办。本届INFOCOM共收到投稿1464篇，接受288篇，录用率19.7%。

内容概述：

近年来，5G和WiFi等无线接入技术为爆炸式增长的移动设备和移动应用提供了网络接入支持，以访问部署在远程中心云中的服务。然而，该模式下用户与对应服务之间的端到端时延较高，特别是对于自动驾驶、无人机、虚拟现实等新兴的时延敏感型应用，现有的云计算模式难以满足日益严苛的时延响应需求。为了满足此类延迟敏感的移动应用性能要求，如图1所示的移动边缘计算可作为云计算基础设施的有效扩展，进一步将计算和存储资源从核心网络部署到用户附近的网络边缘，每个边缘云都可视为一个微数据中心，同时每个边缘云都会部署网络接入点，从而使得移动用户可以通过选择就近的网络接入点来访问和获取移动边缘云服务，以减少终端设备到云服务的延迟。

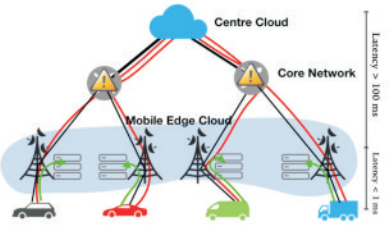


图1 移动边缘计算与云计算的互补

由于移动应用位置随时间动态变化，移动边缘计算架构下的分布式计算服务过程需要回答下列问题：首先是选择车辆周边的哪个网络接入点对服务进行访问？其次，在移动过程中是否需要将服务从上一时刻所在的位置迁移到其他边缘云？如果服务需要被迁移，那么何时进行迁移？应该迁移到哪个位置？本文通过系统建模和在线优化方法回答移动边缘计算的上述基础问题。

本文针对用户服务质量的三种时延进行了建模：第一部分是用户在网络接入点中等待的排队延迟；第二部分是通信延迟；第三部分是对服务进行动态迁移所产生的迁移开销，即当系统决定对服务的放置位置进行调整时所产生的迁移时延。我们将上述问题形式化为长期优化问题，解决此类长期优化问题的主要挑战包括：首先，该问题是NP难问题，不存在多项式时间内的求解方法；其次，长期优化问题的准确求解需要提前获取系统未来的状态信息，而这类信息无法被准确获取或预测；最后，模型中联合优化的部分是复杂的纯整数规划问题，且变量之间相互耦合，尚不存在准确求解的方法。为此，本文设计了一种在线优化方法OLSA 来克服以上难点：首先将某一时刻的总延迟分为非迁移时延和迁移时延，其中非迁移时延仅与系统的当前状态有关，而迁移时延与系统过去的状态有关；OLSA在线分析和比较非迁移时延与迁移时延之间的关系，据此来高效确定系统中用户的网络接入点选择和服务放置决策。基于真实的数据集的实验结果表明仅对网络接入点选择或服务放置进行优化的基准方法无法达到全局最优，而所提出的OLSA方法在大多数情况下的总时延最低。

详细内容参见：

Bin Gao, Zhi Zhou, Fangming Liu* and Fei Xu, "Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing", In Proceedings of IEEE INFOCOM, April 29-May 3, 2019, Paris, France, pp. 1459-1467.

高　彬
2017级硕士研究生
研究方向：移动边缘计算
Email：gaobin.me@gmail.com

刘方明
教授、博导
研究方向：分布式系统与网络
Email：fmliu@hust.edu.cn

面向5G网络切片的自适应干扰感知虚拟网络功能部署

张启夏，刘方明

文章“Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices”被IEEE INFOCOM 2019收录。IEEE INFOCOM是计算机网络与通讯领域的旗舰级会议，会议投稿总数为1464篇，最终录用288篇，录用率仅为19.7%

内容概述：

不同于“一刀切”的4G演进分组核心网，5G网络有望成为一种支持多种垂直应用的新型网络——以灵活的按需组网和高效的服务部署来满足日益增长的服务需求。利用网络功能虚拟化和软件定义网络等新兴技术提出的网络切片，通过在同一物理网络基础设施上分离出多个虚拟的端到端的逻辑网络，为构建服务定制化的5G网络提供了技术支持。每个网络切片从无线接入网、承载网再到核心网上进行逻辑隔离，并根据服务需求部署特定的虚拟网络功能，从而为不同5G场景实现虚拟化的“专用网络”。图1描述了一个5G网络切片架构图，包括三个典型的5G应用场景。

然而，在5G网络切片中部署虚拟网络功能还面临着如下三点挑战：首先，由于5G网络切片是一个端到端的逻辑虚拟网络，应考虑在边缘服务器和核心服务器中联合部署虚拟网络功能；第二，由于不同5G场景的性能需求不同，因此需要一种自适应的服务部署方案；第三，如无人驾驶、4K/8K高清视频等5G场景对吞吐量和延迟等性能要求特别严格，而把虚拟网络功能部署在同一服务器上时会产生性能干扰，从而带来性能下降，如何量化和降低这种性能干扰也是需要考虑的实际因素。

为解决上述三点挑战，本文提出了一种在边缘服务器和核心服务器中联合部署虚拟网络功能的解决方案，能有效量化和降低性能干扰，并自适应且高效地在不同的5G网络切片中部署服务。首先，在建模时，综合考虑了5G网络切片中的边缘服务器和核心服务器、CPU和内存等多维度资源、虚拟网络功能转发拓扑图等，进而通过供给需求模型来量化虚拟网络功能之间的性能干扰，最终设计和实现了一种自适应干扰感知的启发式方法AIA，来在5G 网络切片中自动、高效地部署虚拟网络功能。实验表明，相比传统部署方案，AIA能在无人驾驶和4K/8K高清视频这两个5G应用场景中将吞吐量分别提升20.11%和24.21%。

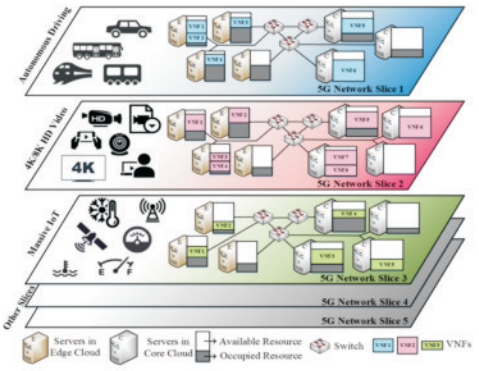


图1 5G网络切片架构图

详细内容参见：

Qixia Zhang, Fangming Liu, and Chaobing Zeng, “Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices”, In Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019), April 29-May 2, 2019, Paris, France, pp. 2449-2457

张启夏

2016级博士研究生

研究方向：网络功能虚拟化

Email：zhangqixia427@hust.edu.cn

刘方明

教授、博导

研究方向：分布式系统与网络

Email：fmliu@hust.edu.cn

边缘数据中心应急需求响应的在线市场机制

陈姝彤，刘方明

文章“An Online Market Mechanism for Edge Emergency Demand Response via Cloudlet Control”被IEEE INFOCOM 2019收录。IEEE INFOCOM是计算机网络与通信领域的旗舰级会议，在国际上享有盛誉和重要影响力，会议投稿总数为1464篇，最终录用288篇，录用率仅为19.7%。

内容概述：

在应急需求响应期间，数据中心等大型用电设施必须将对电网供电量的需求降低到规定数值以下，否则电力系统的稳定性会遭受到严重威胁。边缘数据中心（IaaS cloudlet）作为实现边缘计算的重要基础设施之一，拥有广泛分布的特点和灵活调度负载及控制cloudlet开关的能力，非常适合于配合应急需求响应来协助维护电力系统的稳定性。但边缘数据中心应急需求响应的实现面临着两方面的挑战：一方面，IaaS服务的租户对边缘应用的响应时间往往有很高的要求，动态放置可能使传输时延增加，影响IaaS租户的服务质量；另一方面，边缘数据中心cloudlet的频繁开关会造成大量开销。

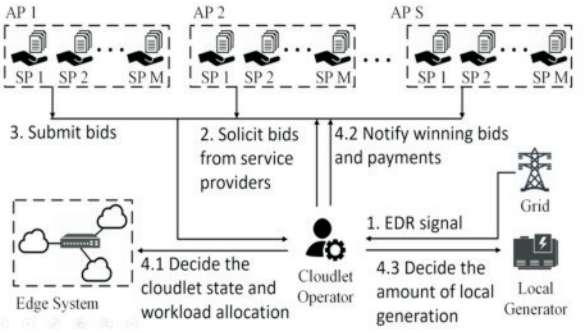


图1. 边缘数据中心应急需求响应的在线市场机制

围绕上述问题，论文首次提出了面向边缘数据中心的应急需求响应在线市场机制（如图1）。该机制通过对IaaS租户进行适当的经济补偿，激励IaaS租户暂时放宽对传输时延的需求；并且，提出了高效的负载放置策略以保证租户的服务质量满足时延要求；同时，设计了在线算法来控制cloudlet的开关时机，在无法预知未来负载信息和应急需求响应要求的情况下，对当前开关状态做出尽可能最优的决策，以减少应急需求响应期间的开销。该机制的目标是最大化应急需求响应期间的全局效益（social welfare）。

文章基于真实的应急需求响应事件，对所提出的机制性能进行了测试，验证了该机制的高效性。实验发现，文章提出的机制几乎完全避免了不必要的开启开销，并且在面对不同的cloudlet开启开销时，均能在应急需求响应期间获得最佳的全局效益。

详细内容参见：

Shutong Chen, Lei Jiao, L. Wang and Fangming Liu, “An Online Market Mechanism for Edge Emergency Demand Response via Cloudlet Control”, In Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019), April 29-May 2, 2019, Paris, France, pp. 2566-2574

陈姝彤
2015级博士研究生
研究方向：数据中心与绿色计算、边缘计算
Email：shutongchen@hust.edu.cn

刘方明
教授、博导
研究方向：分布式系统与网络
Email：fmliu@hust.edu.cn

基于IT负载感知的高效能数据中心混合水冷系统

姜炜祥

该成果“Fine-grained Warm Water Cooling for Improving Datacenter Economy”发表在计算机系统结构领域的顶级国际学术会议之一ACM/IEEE ISCA 2019（The 46th International Symposium on Computer Architecture）。ISCA创办于1973年，今年在全球共收到365篇投稿，仅录用62篇，录用率约为16.9%。

内容概述：

近年来，新型的水冷系统越来越受到数据中心运营商的关注。与传统风冷系统相比，水冷系统具有更高的制冷效率，不但可以提高数据中心内服务器的密度，更重要的是水冷系统可以通过自然蒸发进行散热，从而有效降低制冷系统的能耗，提高数据中心能效。论文通过调研与测量发现：

（1）当前的水冷系统制冷策略非常保守，采用的制冷水温较低，因此存在大量冗余的制冷能力，而通过提高水温可以有效减少水冷系统的能耗，但是提高水温在面临突发负载时存在制冷失败的风险；

（2）当前水冷系统统由于技术限制，大都采用集中式、粗粒度的制冷控制方法，导致水冷系统在应对局部热点时制冷效率低下。

围绕上述问题，本文提出采用温水制冷来减少制冷能耗，并针对温水制冷带来的制冷失败风险，设计了与半导体制冷片协同的混合水冷系统，以及自适应数据中心负载变化的细粒度制冷控制方法：当数据中心中服务器温度普遍较低时，只采用半导体制冷片应对局部热点，而仅当大量服务器温度普遍较高时，才启用制冷机降低水温。该方法能够为数据中心服务器提供精确制冷，避免集中粗放式的制冷能耗浪费。通过原型系统实现和真实数据中心负载的实验验证，所提出的混合水冷系统在对CPU制冷时，可将局部能效比指标PUE将至1.04~1.05，大幅降低了数据中心的能耗成本。

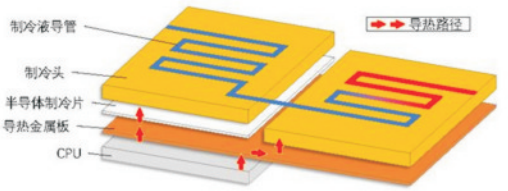


图1（a） 混合水冷结构设计

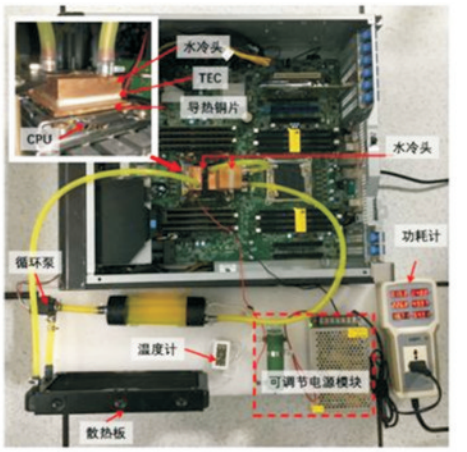


图1（b） 混合水冷系统原型

详细内容参见：

Weixiang Jiang, Ziyang Jia, Sirui Feng, Fangming Liu, and Hai Jin, “Fine-grained warm water cooling for improving datacenter economy”, In Proceedings of the 46th International Symposium on Computer Architecture (ISCA 2019), June 22-26, 2019, New York, USA, pp.474-486

姜炜祥
2014级博士研究生
研究方向：数据中心能耗优化
Email：wxjiang@hust.edu.cn

PostMan：快速缓解突发流量的动态按需组包卸载与批处理系统

金盼盼，刘方明

文章“PostMan: Rapidly Mitigating Bursty Traffic by Offloading Packet Processing”被 USENIX Annual Technical Conference 2019收录。ATC 是系统领域的顶级国际学术会议之一。2019年收到356篇投稿，录用71篇，录用率为20%。

内容概述：

互联网突发事件（如热点新闻、促销季等）会产生难以预测的激增流量，实际测量表明这类突发流量往往生成大量的小数据包。当前的分布式系统及网络协议栈在处理这些小数据包时会产生巨大开销，给后端服务器带来严重的负载均衡问题。解决负载均衡问题的传统方法之一是进行热数据迁移，进而将流量从过载服务器分发至其他服务器。然而，这一方案耗时久、开销大，会进一步降低过载服务器的性能。如何快速高效地处理互联网突发事件所引发的海量小数据包，从而迅速缓解负载均衡问题以降低延迟并提升吞吐率，成为了分布式系统亟待解决的难题。

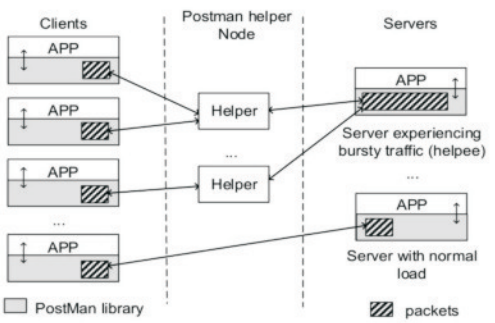


图1 PostMan系统架构图

为了解决该痛点问题，本文设计了快速缓解突发流量的动态按需组包卸载与批处理系统PostMan，可作为热数据迁移的应急互补方案。如图1所示，PostMan在客户端与服务器之间部署少量的中间辅助节点，具有三方面优势：（1）PostMan具有高效性。PostMan实现了基于DPDK 和mTCP的用户态协议栈，以进行高效的按需组包与批处理，即当服务器过载时，辅助节点将发送给同一服务器的大量小数据包先组装成大数据包，然后再转发给该服务器，从而将小数据包的处理开销从过载服务器卸载到辅助节点上；（2）PostMan具有高可扩展性和高容错性。PostMan所部署的中间件辅助节点采用无状态设计，因此辅助节点规模可以任意扩展，而且PostMan能够在不同辅助节点之间进行无缝流量迁移，并在迁移过程中避免数据包乱序的隐患；（3）PostMan具有成本合理性。针对难以预测的流量波动，PostMan能够仅在检测到突发流量时才按需启动或关闭辅助节点，以最大限度减少应急所需的额外成本。

基于Memcached和Paxos的实验结果表明，PostMan可帮助这些重要服务在数百毫秒内缓解突发流量，而传统的热数据迁移方案则需要数十秒之久。具体而言，PostMan能够分别将Memcached和Paxos的吞吐率迅速提高3.3倍和2.8倍。

详细内容参见：

Panpan Jin, Jian Guo, Yikai Xiao, Rong Shi, Yipei Niu, Fangming Liu, Chen Qian, Yang Wang,“PostMan: Rapidly Mitigating Bursty Traffic by Offloading Packet Processing”, In Proceedings of the 2019 USENIX Annual Technical Conference (ATC), July 10-12, Renton, USA, pp. 849-862.

金盼盼
2017级硕士研究生
研究方向：分布式系统与网络
Email：panpanjin@hust.edu.cn

刘方明
教授、博导
研究方向：分布式系统与网络
Email：fmliu@hust.edu.cn

面向大数据集合高效表示的优选Cuckoo过滤器

王飞越

文章发表在第39届发表于国际分布式计算系统大会（ICDCS'19）上，该会议是分布式计算系统领域的重要国际学术会议，本年度录用率为19.6%。

内容概述：

许多大数据应用需要基于有限的内存空间对大数据集合进行高效的表示，并支撑快速的插入、查询、删除等操作。为了满足以上需求，近年来许多近似集合成员表示技术和结构被提出。这些近似集合成员表示结构使用布尔标签或指纹来代替原始的数据进行存储，提升了集合存储和计算效率，并被广泛应用在各种大数据系统当中。在当前各种近似集合成员表示结构中，Cuckoo过滤器是一种空间和查询时间均高效的数据结构。然而，通过理论和实验分析，发现了Cuckoo过滤器中因为其数据随机插入机制导致其在插入过程中存储负载不均，插入重定位操作频繁，延时显著延长的问题。

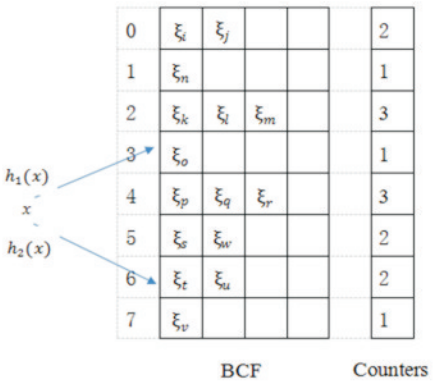


图1 BCF的结构与插入过程

为了有效降低Cuckoo过滤器的重定位次数，提高大数据集合管理的时效性，设计了一种基于插入位置优选的集合表示结构—优选Cuckoo过滤器BCF，并设计了一套高效的算法对大数据集合进行插入、查询和删除等操作。图1显示了BCF的数据结构。BCF基于排队论中的"The Power of Two Choices"的原理，在大数据集合动态插入的过程中，通过简单选择元素插入的位置来尽量保证各个存储位置已插入元素个数的均衡。经证明，与随机插入策略相比，该插入策略在插入同样多元素的情况下，任一存储位置已存储的指纹数的最高值期望会指数级下降。因此，该插入策略能够有效延迟Cuckoo过滤器中满载存储位置出现的时间，因而解决频繁触发重定位导致长插入延时的问题。

结合实际系统大数据集的实验测试结果证明了BCF可以有效均衡不同存储位置的负载分布，延后重定位产生的时间，并显著地提升大数据集合的插入效率。相比于当前使用随机插入策略的Cuckoo过滤器，BCF将平均重定位次数降低了35%，平均插入时间降低了25%。同时，将BCF应用到骚扰电话拦截和智能手机缓存系统中，也显著提升了骚扰电话拦截系统的平均吞吐率和手机缓存系统的效率。

详细内容参见：

Feiyue Wang, Hanhua Chen, Liangyi Liao, Fan Zhang, Hai Jin, "The Power of Better Choice: Reducing Relocations in Cuckoo Filter." Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, July 7-9, pp.358-367, 2019

王飞越
2016级硕士研究生
研究方向：大数据处理
Email：wangfy@hust.edu.cn

RDF图数据的存储、计算和分析总结及展望

林隆龙

文章发表在IEEE 2019 International Conference on Distributed Computing Systems上。ICDCS是分布式计算与系统领域享有盛誉和重要影响力的顶级国际学术会议。ICDCS 2019于2019年7月7日至7月10日在美国德克萨斯州达拉斯举行。它收到了接近1000篇稿件，最终录用131篇文章。

内容概述：

本文给出了RDF存储，计算和分析的研究框架，如图1所示。本文主要从存储、划分、计算框架和分析等四个方面分析研究现状及研究成果，分享我们对RDF大图处理在新型RDF应用程序以及新计算和存储设备的不断涌现情况下所面临挑战的看法。

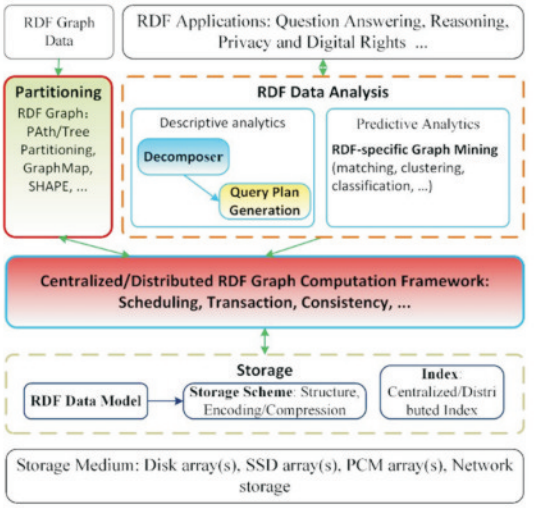


图1 RDF大图研究框架

首先，在RDF数据的存储方面，我们分析了已有的RDF数据模型及存储结构优缺点，给出了一种标记的RDF超图模型。依据此模型，将RDF图表达成行为实体（包括Subject和Object）、列为谓词的矩阵。矩阵对应的值为1或0，指示相关三元组是否存在对应实体。由于该矩阵极为稀疏，针对此矩阵的存储设计了以块为单位的存储结构及函数索引技术。但RDF大图存储对于大规模写操作及数据类型支持仍存在一些挑战。

其次，划分是并行分布式处理RDF大图的前提。目前普遍采用哈希划分。但哈希划分导致RDF大图完整性破坏，从而使得后续RDF大图处理性能变差。我们提出了路径划分及语义哈希划分等提升了划分的局部性及均衡性。但划分与应用场景相关，设计适应面广且性能优的RDF图划分方法依然是一挑战。

第三，处理RDF大图可以采用集中式，也可以采用分布式。我们设计了集中式处理系统，采用多端队列来解决访问冲突问题。在分布式框架里，我们采用已有的分布式框架，如Hadoop、Spark等构建。针对RDF大图处理中存在的特有问题，我们设计了任务划分及调度方法。

最后，RDF数据的价值在于应用。分析RDF数据是RDF数据应用的基础。数据分析包括描述性，查询及预测性分析等。但它们都需要查询RDF大图。我们给出了RDF大图分析问题定义任务。然后针对此问题，给出了两阶段动态求解方法。在分布式环境下，我们给出了SHAPE方法。针对预测性分析中的聚类问题，我们提出了SI-Cluster及VEPathCluster算法。尽管如此，我们认为在RDF大图分析方面，需要着力发展通用高效的机器学习模型及算法来解决RDF大图分析问题。

详细内容参见：

Pingpeng Yuan, Longlong Lin, Zhijuan Kou, Ling Liu and Hai Jin, Big RDF Data Storage, Computation, and Analysis: A Strawman's Arguments, Proceedings of IEEE 2019 International Conference on Distributed Computing Systems (ICDCS 2019), July 7-10, 2019, Dallas, Texas, USA, pp.1693-1703

林隆龙

2017级博士研究生

研究方向：社交网络分析、图算法、时态图数据的社区挖掘。

Email：longlonglin@hust.edu.cn

高时效大数据流连接处理系统Simois

张　凡

文章发表在第39届国际分布式计算系统大会（ICDCS'19）上，该会议是分布式计算系统领域的重要国际学术会议，本年度录用率为19.6%。

内容概述：

大数据时代的应用系统往往需要对实时产生的大数据进行高效的流处理，并快速响应用户的请求。现有的流连接处理系统建立在大规模分布式集群上，利用大量的计算单元，并配合高效数据分发策略，来提升系统性能和可扩展能力。然而，现实应用系统中普遍存在数据倾斜分布的规律，少数键值在数据流中占据着非常大的比例。这种特征限制了分布式流连接处理系统的并行处理能力，尤其是处理热门数据键值的计算单元严重制约系统整体吞吐率。通过对商业网约车的真实数据进行分析，发现乘客的订单流和司机的行车轨迹流中的数据在地理位置分布上均存在高倾斜性。在这种高倾斜分布的数据下，对极少键值对应数据的连接计算带来了系统主要负载，并造成了系统计算节点负载极度不均。

为此，研发了高时效可扩展的流连接系统Simois。Simois采用轻量的数据结构，对导致系统主要工作负载的少量键值进行实时的识别，并采用高效的区分调度策略：对高负载键值进行轮询划分，以使得其带来的连接计算负载均分到所有节点上；对非高负载键值，采用哈希划分的策略，避免大量冗余通信和计算开销，保证系统在存储和计算上的可扩展性。

Simois的系统架构如图1所示。其中，高负载键值预测模块是Simois系统的设计中的核心挑战。为了满足实时流处理的低时空复杂度需求，Simois设计了一种新颖的轻量指数计数器，对每一个到来的流数据键值以常数级时间开销进行计数，基于概率地快速预估流数据的当前到来频率；同时，将两条流的频率乘积转化为对数相加问题，快速滤选出潜在带来高负载的键值。通过使用真实网约车系统的大数据集对Simois系统进行了性能评估，结果表明，相对于当前国际上性能最优的大数据流连接处理系统，Simois将系统总体吞吐率提升了52%，将平均处理延时降低了37%。

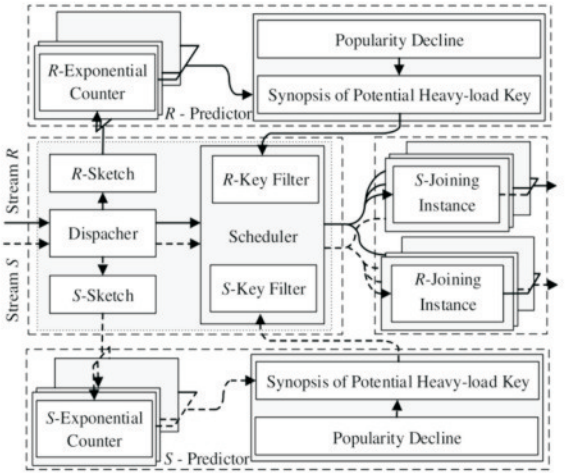


图1 Simois系统架构

详细内容参见：

Fan Zhang, Hanhua Chen, Hai Jin, "Simois: A Scalable Distributed Stream Join System with Skewed Workloads," Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, July 7-9, pp.176-185, 2019

张　凡
2014级博士研究生
研究方向：大数据流处理
Email：zhangf@hust.edu.cn

边缘环境下数据驱动的机器学习任务分配机制

陈　琼，刘方明

文章发表在The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)。ICDCS是分布式计算与系统领域享有盛誉和重要影响力的顶级国际学术会议，本届ICDCS录用率仅为19.6%。

内容概述：

在边缘环境下，由于监测部署成本高、测量可靠性低等原因，因此数据稀缺性成为一个普遍难题。迁移学习非常适合解决此类问题，其基本思想是通过任务之间共享知识来解决数据量不足的任务训练问题，其学习过程如图1所示。

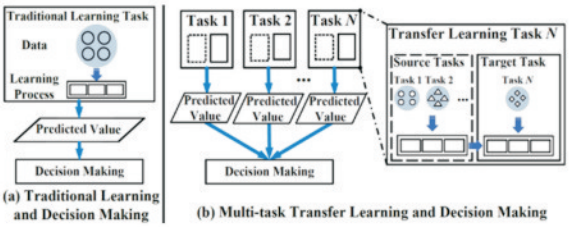


图1　多任务迁移学习示意图

然而，目前迁移学习系统对于资源受限的边缘设备来说过于复杂，原因在于：首先，机器学习模型本身就是计算和通信密集型的；其次，为了避免模型过时、且利用最新数据，需要对每一个任务从头开始反复训练。在这种场景下，面对计算复杂性的挑战，需要解决如何提高多任务迁移学习在边缘环境下的有效性。

通过对真实边缘环境下多任务数据集的分析，本文定义了任务重要性：即执行该任务与不执行该任务之间的最终决策性能（如节能效果）差异。实验结果如图2所示，任务重要性呈明显的长尾分布。因此，本文发现在多任务迁移学习中并非所有的任务都必须执行，而是只有少数任务是最重要的。

基于以上发现，本文的核心思想是：在执行时间限制范围内，选择执行少数重要性高的机器学习任务，从而在保持较高预测准确性的同时提升计算效率。据此，设计了一种数据驱动的机器学习任务分配机制，包含：1）基于聚类强化学习（CRL）的通用流程：首先利用聚类算法在历史环境中获取当前相似环境，然后进行基于CRL模型的任务分配；2）基于支持向量机（SVM）的局部流程：首先基于领域知识的特征工程获取特征向量，然后进行基于SVM模型的任务分配。

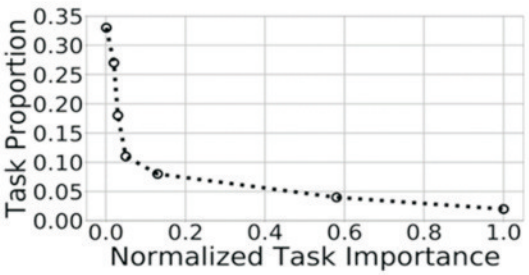


图2　任务重要性的长尾分布

基于真实数据集的实验结果表明，所提出的数据驱动方法相比领域内传统的多任务迁移学习机制，减少了69%以上的处理时间。

详细内容参见：

Qiong Chen, Zimu Zheng, Chuang Hu, Dan Wang, and Fangming Liu*, “Data-driven Task Allocation for Multi-task Transfer Learning on the Edge”, in Proc. of IEEE ICDCS, 7-9 July 2019, Dallas, Texas, USA, pp. 1040-1050.

陈　琼
2017级硕士研究生
研究方向：云计算、边缘计算与应用机器学习
Email：qiongchen@hust.edu.cn

刘方明
教授、博导
研究方向：分布式系统与网络
Email：fmliu@hust.edu.cn

基于强化学习的VNF实例部署和流量调度机制

顾　琳

文章发表在The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)。ICDCS是分布式计算与系统领域享有盛誉和重要影响力的顶级国际学术会议，本届ICDCS录用率仅为19.6%。

内容概述：

对于网络服务提供商而言，轻量级服务是一种高效提供服务的方法，而网络功能虚拟化（Network Function Virtualization, NFV）的出现使其成为可能。通过网络功能虚拟化技术，网络服务提供商可以以特定顺序灵活组合多个虚拟化网络功能（Virtualized Network Functions, VNF）进而构成服务功能链（Service Function Chain, SFC），将传统的基于硬件的网络功能软件化为VNF，实现了网络的集中控制，有效解决了部署网络服务对定制化硬件的依赖，在促进网络的开放性、创新性、灵活性和可扩展性等方面具有巨大的潜力。为提供高效低耗的NFV服务，如何协调VNF实例的部署并在服务功能链之间调度流量以实现网络效用最大化（Network Utility Maximization, NUM）是NFV领域中一个不可忽略的痛点和难点。鉴于传统基于模型的优化方法常受到一些约束条件的限制或基于理想化的假设，本文在深度强化学习的基础下提出了一种基于深度确定性策略梯度（Deep Deterministic Policy Gradients, DDPG）的算法。

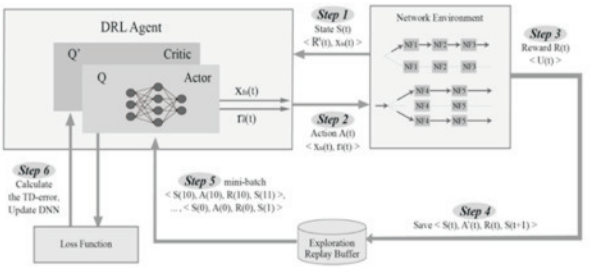


图1 强化学习示意图

针对网络效用最大化问题，我们重新设计了探索方法，并发明了双重经验回放缓冲区结构，并通过实验证明了基于DDPG的改进强化学习算法的高效性。改进版DDPG (cDDPG)算法，主要有：

1）偏差探索：针对在基于正态分布的探索中一些较优的决策可能会被忽略的情况，我们提出了偏差探索，通过对一些特定的状态指定相应的决策进行探索，确保这些较优的决策被更多的探索与学习。

2）双重经验回放库：上面提到的第二个问题是由于在经验回放库中无法区分地存储所有样本而引起的。因此，我们提出了一种由全局回放库和基线回放库组成的双重经验回放库结构。全局回放库与通用DDPG算法设计中的经验回放库相同，而基线回放库特别用于存储效用良好的样本。

3）根据两个经验回放库的不同内容和特性，我们还重新设计了经验取样和替换策略，在不同的训练阶段改变从全局回放库和基线回放库抽取不同比例的样本来训练神经网络，通过这种方式来确保一些潜在的效用良好的样本以更高的权重被选取来做神经网络的训练。

详细内容参见：

Lin Gu, Deze Zeng, Wei Li, Song Guo, Albert Y. Zomaya, Hai Jin, “Deep Reinforcement Learning based VNF Management in Geo-distributed Edge Computing”, in Proc. of IEEE ICDCS, 7-9 July 2019, Dallas, Texas, USA, pp. 934-943.

顾　琳
讲　师
研究方向：分布式系统与网络
Email：lingu@hust.edu.cn

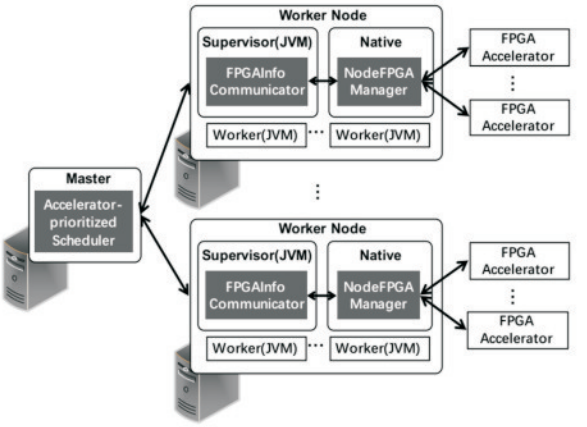
FPGA加速边缘环境下的流数据处理

胡　蝶

文章发表国际学术会议International Conference on Distributed Computing Systems 2019。ICDCS会议是分布式计算系统领域中历史最悠久的会议，主要关注分布式计算系统的各个方面，涵盖了云计算、分布式大数据系统、分布式操作系统和中间件、物联网、边缘计算等领域，近年论文录用率在20%左右。

内容概述：

如今，流数据处理应用程序代表着边缘计算场景下的“killer”应用程序，它将计算任务运行在靠近数据源的边缘，可提升数据处理的实时性。近年来这方面的研究工作主要集中在引入轻量级分布式流处理系统到边缘环境中，以及在边缘服务器和云数据中心之间划分和调度计算任务。但是由于边缘服务器的计算能力有限，当前已有研究成果无法实现流数据应用程序所需的超低延迟。通过在边缘服务器中引入FPGA并将其集成到DSP 系统中，我们能够在边缘基础架构中实现更低延迟的流数据处理。设计并实现了F-Storm – 应用于边缘服务器上的FPGA加速的通用分布式流处理系统。F-Storm将基于PCIe的FPGA集成到边缘环境的流处理系统中，并提供加速器服务给每一个运行在此系统上的流数据应用程序。



系统的总体架构图如图一所示，我们在Storm系统的基础上实现了F-Storm，首先我们设计并实现了NodeFPGAManager组件，实现了基于JVM运行的大数据系统与基于C/C++/OpenCL的FPGA的轻量级集成，降低部署和运行成本；与此同时，我们还实现了加速器优先的调度策略，资源受限的边缘环境下自适应的CPU-FPGA混合调度；另外，我们利用数据批量传输和数据传输流水线技术，减少了JVM与FPGA之间的数据传输开销，从而使得FPGA的使用变得更加高效，程序延迟能够降低；最后，我们还提供了用户友好的编程接口，提升用户开发流处理应用程序的效率。

实验表明，与Storm相比，F-Storm将矩阵乘法和grep应用的延迟减少了36%和75%，并分别获得了1.4倍和2.1倍的吞吐量改进，证明了将FPGA应用于边缘环境下的流数据处理的加速的可行性和潜力。

详细内容参见：

S. Wu et al., "When FPGA-Accelerator Meets Stream Data Processing in the Edge," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 2019, pp. 1818-1829. doi: 10.1109/ICDCS.2019.00180

胡　蝶
2016级硕士研究生
研究方向：FPGA流计算加速
Email：hudie2016@hust.edu.cn

Jidar: A Jigsaw–like Data Reduction Approach without Trust Assumptions for Bitcoin System

肖　江，戴小海

该成果发表在The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), 是实验室分布式系统组在区块链领域的研究成果。ICDCS是分布式系统领域的重要会议之一, 专注分布式相关理论和技术，本届ICDCS录用率仅为19.6%。

内容概述：

因其去中心化、可追溯和无需信任等特点，比特币近年来吸引了各界的广泛关注。但持续增长的比特币存储开销，使得每个用户都存储一份完整的账本数据变得越来越难。特别是对只进行了少量交易且只拥有有限存储空间的用户而言，存储完整的比特币数据对他们来说不公平也不现实。已经有一些工作基于提出的新假设以重新设计系统协议，从而降低存储开销。较为人知的一个方案是轻节点，其依赖于全节点存储所有的数据，并及时将与其相关的数据返回。但是，轻节点方案引入了永久性数据丢失的风险，因为和某轻节点相关的数据可能会在后期被所有全节点裁减掉。另一个方案将节点组织成若干个合作组，合作组中的每个节点只需存储一部分数据，整个合作组存储完整的账本数据。但是，该方案基于合作组中的节点互相信任的假设，在现实中很难达成。

设计与实现

针对比特币数据存储开销大的问题，本文设计了一种类似于拼图的数据减缩方案。该方案无需任何额外的信任假设，且依赖于以下对于比特币交易系统的理解：

1. 比特币中的大多数用户只关心他们自己的数据，而对其他人的数据并不关心

2. 用户没有义务帮忙存储其他用户的数据，因为他们并不能直接从数据存储中获利，这可能会导致数据保存者（如全节点）在后期不负责任的删减。

3. 系统的优化不应该破坏其原有的优点，如去中心化、可追溯性和无需信任等。

该方案中的每个节点都需存储每个区块中的区块头，但只需存储区块体中和自己相关的交易和对应的梅克尔分支。如图1所示，当账户A接收到新的区块k+1时，因为该区块中包含的交易Tx-1和账户A相关，因此账户A需要保留Tx-1和对应的梅克尔分支。相反，当账户接收到区块k+2时，由于该区块中的所有交易都和账户A无关，因此账户A只保留k+2的区块头信息。

因为不存在任何节点保存了所有的数据，新交易的有效性验证必须依赖于交易发起者发送的证明信息。新交易的有效性验证主要是对交易的input进行验证，包括两部分：1）input是之前的某个有效output (existence proof)；2）该input没有被花费过 (unspent proof)。Existence proof依赖于交易发起者提供的梅克尔分支作为交易证明的一部分。Unspent proof依赖于我们对区块头进行的修改，我们在区块头中添加了一个布隆过滤器。一个区块中所有的input用于构建该布隆过滤器。通过将input与之后的每个区块头中的布隆过滤器进行比较，得到该input没有被花费过。

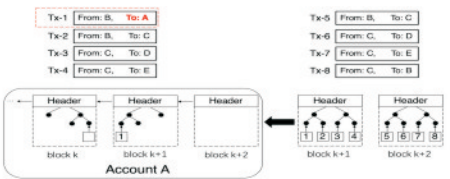


图1 接收并存储新区块的示意图

详细内容参见：

Xiaohai Dai, Jiang Xiao, Wenhui Yang, Chaofan Wang, and Hai Jin Jidar: A Jigsaw-like Data Reduction Approach without Trust Assumptions for Bitcoin System[C]// 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019.
DOI: 10.1109/ICDCS.2019.00132

肖　江
副教授
研究方向：分布式系统，区块链数据管理
Email：jiangxiao@hust.edu.cn

戴小海
2017级博士研究生
研究方向：区块链可扩展性
Email：daixh@hust.edu.cn

容器资源视图隔离

黄　航

文章发表在 ACM HPDC'2019上，该会议2019年接收有效投稿数为106，其中22篇被接受为长文口头报告，录用率约为21%。该会议为CCF B类会议，是由美国计算机学会组织的并行与分布式计算领域重要学术会议。

内容概述：

随着操作系统级虚拟化的高速发展，容器已经广泛应用于云环境下的软件交付、分发、部署中，成为了虚拟机的可行替代品。在传统的虚拟机中，客户机操作系统必须运行在虚拟硬件上，且每个客户机都有自己的操作系统及内核。但是在容器中，应用可以直接访问主机的物理资源，并且多个容器间共享一个主机内核。相较于虚拟机，容器不需要额外的虚拟硬件层，这避免了不必要的虚拟化开销，但也带来了隔离性不足的问题。当前，隔离性问题给容器资源管理带来巨大的挑战。缺乏硬件抽象层，主机系统总是将物理资源总量暴露给每个容器，而实际上每个容器只能使用有限的系统资源，这使得基于系统资源视图来进行资源管理的运行时（OpenMp等）以及编程语言（Java等）受到严重影响。

本文的工作主要关注CPU与Memory资源。其设计目标是消除容器中应用观察到的资源可用量与实际资源可用量间的语义鸿沟，并且构建弹性应用去适应动态变化的资源可用量。主要贡献包括以下四点。

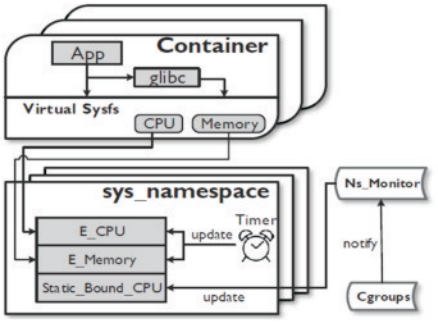


图1 系统架构

首先，为了处理容器中的语义鸿沟，作者设计实现了容器间相互隔离的实时资源视图，该资源视图不断更新以实时反映容器的有效资源。如图1所示，系统主要包含3个部分：1）sys_namespace，维护容器的有效资源量（CPU和内存）；2）虚拟sysfs，提供连接sys_namespace和应用程序的接口；3）Ns_Monitor，追踪Cgroups的改变并更新相对应的sys_namespace。

其次，设计了容器有效CPU资源的计算方法和动态调整机制。有效的CPU资源而不是大量竞争中的CPU资源可以让应用高效调整其并行度。

第三，设计了容器有效内存资源的计算方法和动态调整机制。本文采取启发式的算法，通过预测增加容器有效内存后对主机系统内存的影响，来逐步地增加容器的有效内存。当监控到主机系统内存紧张，启动全局内存回收。

第四，设计并验证了容器中弹性应用的构建方法。通过JVM和OpenMP两个案例，本文说明了如何在容器中构建弹性应用，通过容器私有资源视图提供的有效CPU和内存资源信息来动态调整应用配置，从而在资源动态竞争的容器环境下保持和提高应用性能。

通过测试表明该系统能够显著提升容器中应用的性能，可提升Java垃圾回收性能高达80%。

详细内容参见：

Hang Huang, Jia Rao, Song Wu, Hai Jin, Kun Suo, Xiaofeng Wu. Adaptive Resource Views for Containers. In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing 2019 Jun 17 (pp. 243-254). ACM.

黄　航
2017级博士研究生
研究方向：容器技术
Email：huanghang@hust.edu.cn

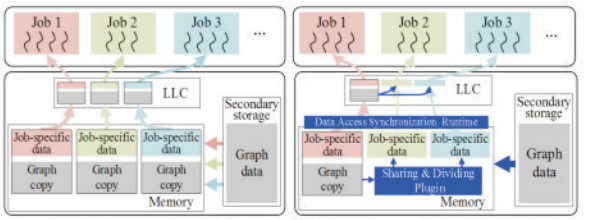
面向并发图计算任务的存储系统

赵　进

文章发表于Proceedings of the 2019 International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2019)上，SC是高性能计算领域的顶级国际会议，今年共收到344篇投稿，录用78篇，录用率约为22.7%。

内容概述：

随着实际应用中对图分析的需求快速增加，同一底层图上往往并发地运行着大量的图计算任务。然而，现有图计算框架的存储系统主要为有效地服务单个任务而设计，使得并发图计算任务以如图1(a)所示的模式执行，各任务独自地访问相同的图结构数据，这造成了存储资源（内存，缓存等）和数据访问通道的冗余开销，最终导致现有图计算框架在处理并发任务时吞吐量较低。然而，由于并发任务通常反复地遍历同一底层图，因此这些任务之间的数据访问存在很强的相似性。一方面，在遍历过程中，同一底层图的大部分比例可以由多个并发任务共享，称为空间相似性。另一方面，在一段时间内，相同的图数据可能分别地被不同的并发任务访问，称为时间相似性。



(a) 现有图处理系统执行并发图计算任务　(b) 现有系统与GraphM结合来执行并发图计算任务

图1 并行图计算任务的执行模式

基于以上发现，本文设计了一个高效的存储系统，它可以轻松地嵌入到与现有的图计算框架中并充分利用并发图计算任务之间的数据访问相似性，通过有效地降低数据访问和存储开销来提高现有图计算框架在处理并发任务时的吞吐量，同时减少用户的编程负担。并发任务在该系统的执行模式如图1(b)所示。在该存储系统中，本文设计了一个Share-Synchronize机制来挖掘并发运行任务之间的数据访问相似性。详细地说，图结构数据首先需要与任务特征数据进行解耦合以便其能够被并发任务共享，而任务特征数据由各任务单独地维护。同时，共享图结构分区(partition)需要通过本文设计的图标记算法来逻辑地划分为适应缓存大小的块(chunk)。然后，该存储系统按统一的顺序将图结构块流到内存/缓存中，并使得并发任务以细粒度同步的方式并发地处理这些图数据块，从而使得并发任务对共享图数据的访问规则化。通过这种方式，内存/缓存中只需维护一个图构数据的副本就能服务多个并发图计算任务，并且数据访问成本由这些并发任务均摊。更重要的是，当该图存储系统嵌入到现有图计算框架时，这些框架仍然按照它们的执行模型运行，这是因为该图存储系统透明地规则化了在现有框架上执行的并发任务的遍历路径。此外，本文还设计了一个图分区的调度策略来最大化加载到内存中的图分区的利用率。实验表明，我们设计的图存储系统能够使得现有图计算框架上执行的并发图计算任务的吞吐量提高1.73倍到13倍。

详细内容参见：

Jin Zhao, Yu Zhang, Xiaofei Liao, Ligang He, Bingsheng He, Hai Jin, Haikun Liu, and Yicheng Chen. GraphM: An efficient storage system for high throughput of concurrent graph processing. In Proceedings of the 2019 International Conference for High Performance Computing, Networking, Storage and Analysis, pages 401–412, 2019.

赵　进

2017级博士研究生

研究方向：图计算

Email：zjin@hust.edu.cn

高时效大数据流连接处理系统FastJoin

周舜杰

文章发表在第33届国际并行和分布式处理系统大会（IPDPS'19）上，该会议是并行和分布式计算领域的重要国际学术会议，今年收到投稿372篇，录用103篇，录用率27.7%。

内容概述：

大数据时代，许多实时数据分析与处理系统需要对多条流数据进行快速和精准的连接处理。高效的流连接处理系统需要满足以下几个基本需求：1）低延时和高吞吐；2）内存高效使用和可扩展性；3）流连接结果的完整性。为了满足以上需求，分布式流连接系统需要探究高效的流数据划分和调度策略。现有的流连接系统主要使用两种数据划分策略：随机划分和基于连接键值的哈希划分策略。对于关联稀疏的真实大数据流而言，随机策略会引起大量不必要的连接计算和复制通信开销。哈希划分策略将相同键值的两条流数据均映射到相同的连接计算单元上，这样可以避免了冗余的计算和通信开销。然而，现实的大数据流往往存在数据的高倾斜的特征，简单的哈希映射方法会导致不同计算单元上分配的元组个数差异巨大，带来严重的连接计算负载不均，制约系统的整体吞吐性能。

为了解决以上问题，研发了高性能分布式流连接系统FastJoin。FastJoin的设计思想是实时地对不同流连接计算单元的负载不均情况进行预估，一旦检测到节点负载不均的情况产生，即时通知当前最重负载的节点向最轻负载的节点进行负载迁移，直到两个节点达到均衡。

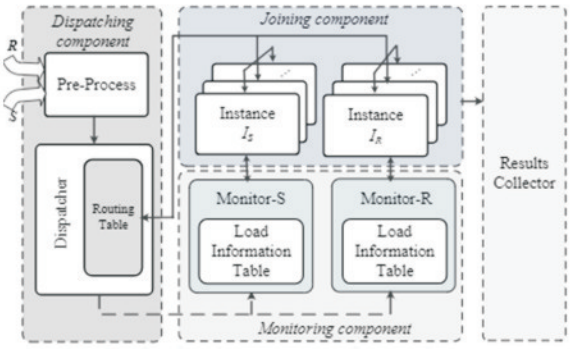


图1 FastJoin系统架构

FastJoin的系统架构如图1所示。系统通过使用二部图连接的思路来保证连接的完整性和存储的可扩展性，并设置独立的负载监测模块，对流数据实时统计计算，以对每个连接计算单元实时计算负载情况进行有效预估。当负载不均发生，系统开始迁移键值时，为了避免大量数据迁移导致当前计算负载最低的节点负载激增，同时极小化对数据迁移产生的开销，系统需要有效地筛选迁移的键值集合。最优化迁移键值的选择是一个NP难的问题，FastJoin设计了一种轻量的重要键值筛选技术，并基于贪心算法和模拟退火的启发式优化算法进行了系统算法实现。通过使用真实系统的大规模数据对FastJoin系统的性能进行了评估，测试结果表明，相对于目前最新流连接系统，FastJoin将系统总体吞吐率提高了31.7%，将平均处理延时降低了15.3%。

详细内容参见：

Shunjie Zhou, Fan Zhang, Hanhua Chen, Hai Jin, Bing Bing Zhou, "FastJoin: A Skewness-Aware Distributed Stream Join System." Proceedings of the 33rd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, May 20-24, pp.1042-1052, 2019.

周舜杰
2016级硕士研究生
研究方向：大数据处理
Email：zhoushunjie@hust.edu.cn

基于SSD设备层任务队列状态感知的公平性I/O调度算法

樊 浩

该成果“NCQ-Aware I/O Scheduling for Conventional Solid State Drives”发表在B类国际学术会议International Parallel & Distributed Processing Symposium（IPDPS’19）上。IPDPS会议是分布式和并行计算领域的重要国际会议，主要关注分布式和并行计算相关的算法、编程模型、系统软件以及体系结构设计。

内容概述：

SSD内部多通道高并发的特性为SSD带来了高性能，并且随着SSD成本的下降，越来越多的主机使用SSD加速数据密集型应用。但是在并发场景中，例如云环境中多虚拟机共享主机存储时，由于主机无法有效地感知SSD内部的多通道以及GC等操作，SSD内部来自不同虚拟机的请求无法被公平地调度。

目前基于SSD的公平性的研究主要集中在两方面：1. 通过修改SSD的硬件特性来提高公平性。这些方法对公平性提高明显，但是需要修改硬件接口，不利于SSD的广泛部署。2. 设计SSD友好的I/O调度算法。虽然这些I/O调度算法有一定的效果，但是这些方法都忽略了设备层任务队列对于SSD性能的影响。

我们设计了一系列实验，观察NCQ对公平性的影响。实验显示，不同特征的I/O负载并发时，可能会产生NCQ竞争。侵略性强的负载（负载的请求块大、请求数目多）会过量的占用NCQ，这使得侵略性低的请求很难进入NCQ，从而导致侵略性低的负载无法被及时地发送到SSD。同时，调度算法内的请求合并可以提升总体带宽，但是合并会减少负载的请求数目从而改变负载的侵略性。另外，调度算法使用anticipation机制提升公平性。当调度算法中低侵略性负载的请求被全部派发到SSD，调度算法会暂时停止向NCQ发送请求，并等待低侵略性负载的请求到达调度算法，来提高低侵略性负载被调度的机会。这一机制降低了NCQ的利用率，从而减低了SSD的并行度和性能。

基于以上发现，为了同时提高SSD在并发环境中的使用率和公平性，我们设计了NCQ感知的I/O调度框架，NASS。其工作原理如图1所示。首先，NASS通过一个负载评估模型，根据负载的一些重要参数实时地评估负载。然后，动态地控制负载的请求派发来缓解NCQ竞争并提高NCQ的利用率。

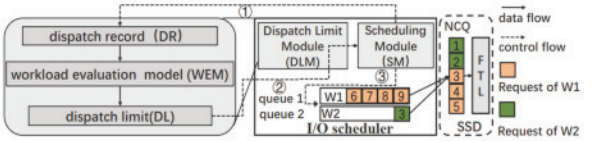


图1 NASS的总体架构图

我们将NASS集成到了各种主流的I/O调度器，包括CFQ、BFQ、FIOPS、FlashFQ。我们使用Filebench和HiBench生成的真实并发I/O负载对NASS进行了综合测试。NASS在不同调度算法下可以提升11-23%的公平性和9-29%的SSD设备利用率。NASS非常适用于复杂的云计算存储环境，这使得该项技术具有较好的应用前景。

详细内容参见：

Hao Fan, Song Wu, Shadi Ibrahim, Ximing Chen, Hai Jin, Jiang Xiao, and Haibing Guan. NCQ-Aware IO Scheduling for Conventional Solid State Drives-revise In Proceeding of IPDSP’19.

樊 浩
2014级博士研究生
研究方向：I/O虚拟化
Email：u201014452@hust.edu.cn

容器镜像高效构建方法研究

黄　卓

文章发表在35th International Conference on Massive Storage Systems and Technology(MSST'19)，会议主要关注分布式存储等方面。2019年收到77篇投稿，录用21篇，录取率约为27%。

内容概述：

容器，例如Docker，已经变得越来越流行。它是一种轻量级的内核虚拟化技术，被广泛使用来替代传统虚拟机。因为Docker容器与主机共享系统内核，所以其用于虚拟化的namespace和用于资源限制的Cgroup的初始化过程很快。为了支持灵活的镜像定制，Docker采用了一种在客户端构建镜像的机制。用户可以通过运行Dockerfile，包含用于构建镜像的指令的脚本）从提供高度可定制的镜像构建。

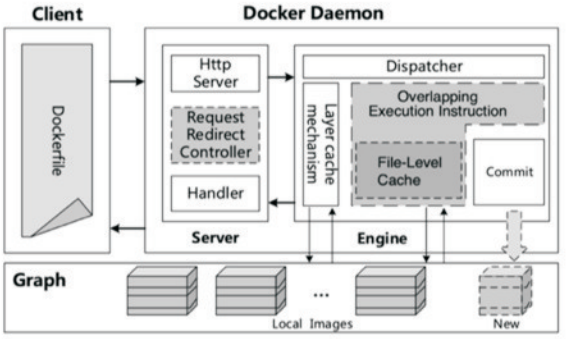


图1. FastBuild架构

在本地服务器上执行Dockerfile中的指令时，通常需要下载大量文件，并且这些文件必须从Internet上的远程服务器中检索。在许多情况下，应用需要在容器启动之前先构建镜像，而检索文件是镜像构建效率低下的主要原因，容器镜像的构建制约了容器的快速启动。

我们进行了大量的镜像数据分析，显示在构建镜像期间，存在大量可重复使用的输入文件，这保证了使用本地缓存来提供最近访问的文件，以避免不必要的远程文件下载。尽管这种方法具有提高镜像构建效率的巨大潜力，但其设计仍面临着巨大挑战。我们需要找到一种方法，在不改变容器镜像格式以及内容的前提下，利用本地缓存的文件数据。因为Dockerfile指令的执行位于容器实例中，Docker无法感知生成的文件请求，所以FastBuild必须非侵入式地获取文件请求以用于缓存文件，并在检测到命中后将请求重定向到缓存。FastBuild从执行Dockerfile指令的容器实例的network namespce中截取到相应的文件请求，做到了不改变容器组织结构和数据内容。在此基础上还并行化的执行dockerfile指令，将每一条指令的execution阶段和commit阶段进行重叠，进一步加速了镜像的构建。

最后，通过从Docker Hub获得的Dockerfile实验表明，我们的系统可以将构建速度提高多达10倍，将下载的数据减少72%。

详细内容参见：

Zhuo Huang, Song Wu, Song Jiang, and Hai Jin, "FastBuild: Accelerating Docker Image Building for Efficient Development and Deployment of Containers", in Proceedings of 35th IEEE Symposium on Massive Storage Systems and Technologies (MSST'19), Santa Clara, CA, May, 2019.

黄　卓
2016级博士研究生
研究方向：容器性能优化
Email：huangzhuo@hust.edu.cn

基于社交网络中心性分析的安卓市场级别恶意软件扫描系统

吴月明

文章发表在Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)上。ASE是国际公认的软件工程领域顶级学术会议之一，是CCF A类会议。2019年收到435篇投稿，录用91篇，录用率约为21%。

内容概述：

针对安卓应用市场的恶意软件扫描需要具备快速且精确两个条件，现有安卓恶意软件检测技术可以分为基于语法特征和基于语义特征两种。然而，基于语法特征的检测技术效率高，但因缺乏应用的语义信息而导致精确度不够高；基于语义特征的检测技术精确度高，但一般程序分析开销较大，从而导致扩展性并不高。因此，为了解决上述问题，我们提出了一个基于社交网络分析的安卓市场级别的恶意软件扫描系统（MalScan）。我们将应用的函数调用图视为复杂的社交网络，并对其进行社交网络分析，挖掘出其中敏感API的中心性，然后以此作为语义特征来检测恶意软件。

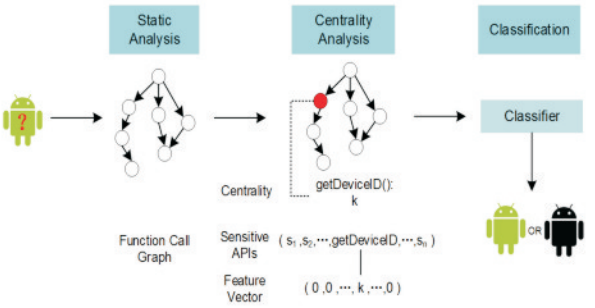


图1 MalScan系统框架图

如图1所示，MalScan主要分为3个阶段：1）静态分析阶段，该阶段的主要目的是为了得到安卓应用的函数调用图；2）中心性分析阶段，在得到应用的函数调用图之后，针对图中的敏感API节点，我们采用社交网络分析提取出这些敏感API的中心性；3）分类阶段，这些敏感API的中心性会作为特征构建特征向量，并输入至机器学习模型中进行训练，最后将训练好的模型用来检测和分类新的安卓应用。

实验结果表明，相对于MaMaDroid和Drebin（MaMaDroid是发表于2017年NDSS上的一篇文章，Drebin是发表于2014年NDSS上的一篇文章），在检测精度方面，MalScan可以检测到更多的恶意软件；在检测效率方面，MalScan的平均开销只需0.7秒，比MaMaDroid和Drebin快100多倍。另外，以GooglePlay市场为例子的调研结果表明，MalScan在扫描GooglePlay中的应用时，平均开销最快可以达到1秒左右，如此高的效率也验证了MalScan适用于市场级别的恶意软件扫描。

详细内容参见：

Yueming Wu, XiaoDi Li, Deqing Zou, Wei Yang, Xin Zhang, and Hai Jin, “MalScan: Fast Market-Wide Mobile Malware Scanning by Social-Network Centrality Analysis”, In Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), November 12-14, 2019, San Diego, California, United States

吴月明
2016级博士研究生
研究方向：恶意软件检测，漏洞检测
Email：wuyueming@hust.edu.cn

内存计算垃圾回收优化器

胡振宇，石宣化

文章发表在ACM Transactions on Computer Systems(TOCS)上。该期刊每年出版4期，每期录用论文3篇左右，近五年的平均影响因子是4.404，主要关注计算机领域系统设计、规范、实现、行为和使用等方面的研究。

内容概述：

大数据处理系统近十年来发展迅猛，比如Apache Hadoop、Apache Spark，能基于普通的硬件以横向扩展的方式处理海量的数据。这些大数据处理系统大多使用带有托管执行环境的高级语言（如Java、Scala、C#等）开发来降低分布式环境下的部署和调试的难度。然而托管环境提供内建的高级功能，比如自动内存管理和并发模型，方便了用户的开发，却使得其对象模型的底层实现非常复杂，带来额外的内存和中央处理器（CPU）开销。以JVM虚拟机为例，当堆中有大量对象存活时，GC会带来严重的CPU开销和内存膨胀问题。

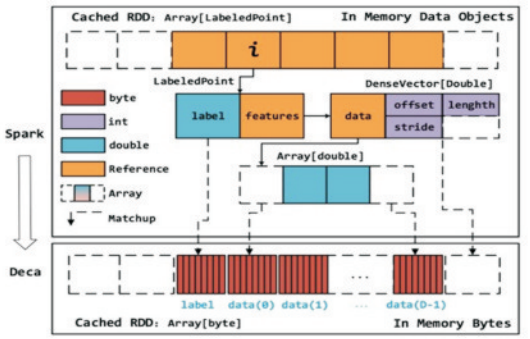


图1 对象拆解示意图

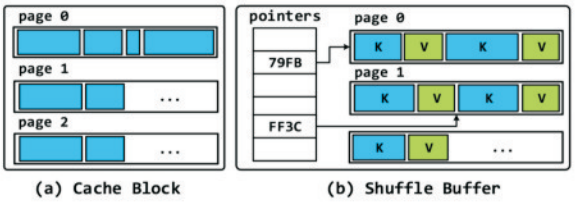


图2 Deca内存布局

我们提出Deca，该系统从用户自定义类型（UDT）和用户自定义方法（UDF）入手，将分布式数据处理系统中的数据对象的生命周期与数据容器关联，一方面剥离了对象中关于锁、状态等附加信息，完全拆解对象，解决了内存膨胀问题(如图一所示)，另一方面通过生命周期的统一管理，将对象拆解到字节数组中，跳过JVM的内存管理，消除垃圾收集的CPU开销(如图二所示)。

使用生成和真实数据集进行的广泛实验研究表明，与原生Spark相比，Deca能够将垃圾收集时间减少99.9%，内存消耗减少46.6%，缓存开销减少23.4%；在无数据溢出时，实现执行加速1.2x-22.7x，在数据溢出的情况下，实现16x-41.6x的加速。

详细内容参见：

Xuanhua Shi, Zhixiang Ke, Yongluan Zhou, Hai Jin, Lu Lu, Xiong Zhang, Ligang He, Zhenyu Hu, and Fei Wang. "Deca: a garbage collection optimizer for in-memory data processing." ACM Transactions on Computer Systems (TOCS) 36, no. 1 (2019): 3.

胡振宇
2017级硕士研究生
研究方向：内存计算
Email：cszhenyuhu@hust.edu.cn

石宣化
教授，博导
研究方向：云计算与大数据处理、异构并行计算等
Email：xhshi@hust.edu.cn

关联性感知的并发图处理系统

张　宇

本文发表在TOS 2019 (ACM Transactions on Storage)。ACM TOS是计算机体系结构、并行与分布计算、存储系统等领域的最新的研究进展和技术，是中国计算机学会（CCF）推荐的A类顶级期刊。ACM TOS每年出刊4期，每期8篇论文左右，2019年影响因子为1.322。

内容概述：

随着图计算的快速发展，图计算平台通常需要高效处理大量并发图分析任务。虽然现有解决方案已广泛研究和优化单图分析任务的执行，然而它们由于缓存干扰和内存墙等问题对于并发图分析任务的执行面临高额的数据访问开销，导致系统吞吐量低。我们观察到并发图分析任务的数据访问中存在大量时空关联性。基于此分析观察结果，本文因此提出了一种关联性感知的并发图处理系统，以充分利用这些数据访问关联性，使并发图分析任务能够高效地处理缓存/内存中共享图结构数据，通过有效降低平均数据访问开销提供更高的系统吞吐量。

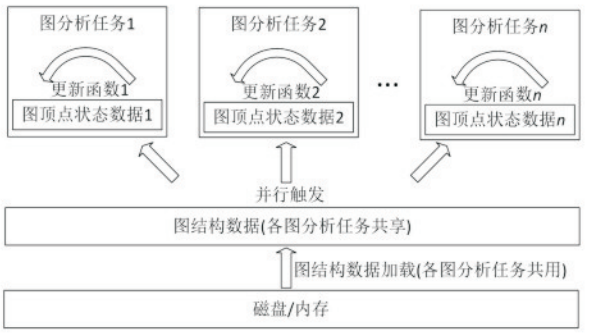


图1 基于LTP的多任务执行模型解释

具体来说，如图1所示，本系统采用基于LTP（Load，Trigger，Push）的多任务执行模型支持多图分析任务的有效执行。LTP模型将图数据分解为图结构数据和图顶点状态数据。各图分析任务都会拥有一个私有图顶点状态数据集，而图结构数据为各图分析任务所共享。然后，LTP模型将根据各图分析任务的数据访问和处理需要，反复载入各任务共享的待处理的图结构数据，触发各图分析任务根据此图结构数据和各自的更新函数（用于完成各算法目的），更新各自私有的图顶点状态直到收敛。这样，它允许各图分析任务利用它们之间的数据访问空间/时间关联性，充分共享图结构数据及其访问，提高系统吞吐率。

在LTP模型中，各图分析任务分别拥有私有空间存储各算法自身图顶点状态数据，而图结构数据作为全局数据被它们所共享。因此，它保证了执行过程的正确性。由于对同一个图进行分析的图算法们需要遍历它们共享的图结构数据以对各图顶点进行相应处理达到各自目标，因此它们存在数据访问空间/时间关联性。LTP模型将最需要处理的图划分块加载到内存，从而能利用时间关联性减少数据访问量。此外，LTP模型以块为单位并行触发各算法的更新函数对此块进行处理，以利用各任务数据访问空间关联性，尽量让它们共用图结构数据（实践显示其占到了整个图分析任务所需访问数据的大部分）访问。实验结果表明，与现有解决方案相比，本方法能显著提高并发图分析任务的吞吐量。

详细内容参见：

Yu Zhang, Jin Zhao, Xiaofei Liao, Hai Jin, Lin Gu, Haikun Liu, Bingsheng He, and Ligang He. CGraph: A distributed storage and processing system for concurrent iterative graph analysis jobs. ACM Transactions on Storage, 15(2):10:1–10:26, 2019.

张　宇
副教授
研究方向：大数据系统软件、运行时优化
Email：zhang_yu9068@163.com

深度学习系统内存重用及优化方法研究

蒋文斌，马　阳

文章发表在ACM Transactions on Architecture and Code Optimization上。该期刊每季度出版一期，每期录用论文10篇左右，2018年影响因子是1.166，主要关注面向嵌入式和通用系统的计算机架构、编程模型、编译器和操作系统等方面的研究。

内容概述：

在深度学习中，GPU通常用于加速深度神经网络的训练过程，然而GPU有限的物理内存意味着它很难训练大规模的深度神经网络模型。现有的内存优化方法主要包括重新计算方法和CPU-GPU迁移方法，它们对神经网络中的所有层应用单一的优化方式，不能实现理想的训练性能。因此我们提出一种基于层的内存复用和优化方法Layup，以进一步节约GPU显存消耗。我们的方案主要基于以下两个观察：第一，不同层对于计算资源和内存空间需求并不总是保持一致，因此需要设计以层为粒度的优化方案以实现更小的性能开销；第二，在神经网络的训练过程中会产生多种类型的中间数据，为了实现更好的内存效率，需要考虑中间数据的内存消耗问题。

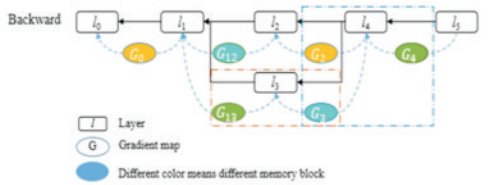


图1　（a）梯度图内存复用策略

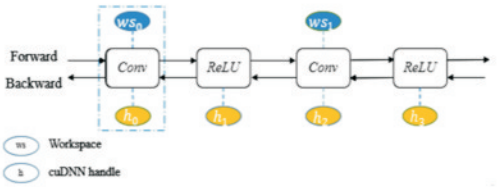


图1　（b）卷积工作空间和cuDNN句柄的内存复用策略

Layup方法包含两种策略：（1）层自适应策略选择机制，为神经网络中不同的层选择合适的内存优化方式，降低训练时的性能开销。结合GPU异步执行的特性，通过分析神经网络中不同层的迁移开销和重新计算开销，将神经网络中的层划分为计算敏感型和迁移敏感型两类，使用CPU-GPU迁移方法优化计算敏感型层的特征图，使用重新计算方法优化迁移敏感型层的特征图，同时使用流水线并行的方式将数据传输过程和计算过程重叠起来。（2）基于多种中间数据的内存复用策略，使用滑动窗口的方式复用梯度图的内存空间，如图1（a）所示，并基于神经网络逐层计算的特性，逐层复用卷积工作空间和cuDNN句柄数据，如图1（b）所示，以进一步降低深度神经网络训练时的内存使用量。

实验表明，Layup方法可以显著降低深度神经网络训练时的内存消耗，同时保持较低的性能开销。训练时的内存消耗量最高减少92%左右，同时，在所测试的神经网络模型上的性能开销平均仅为12%。特别地，在内存容量为12GB的GPU上可以训练高达2500层的ResNet模型（批量大小为16），与SuperNeurons方法相比提升约30%，进一步扩展了单个GPU上神经网络模型的规模。

详细内容参见：

Wenbin Jiang, Yang Ma, Bo Liu, Haikun Liu, Bingbing Zhou, Jian Zhu, Song Wu, and Hai Jin,?Layup: Layer-Adaptive and Multi-Type Intermediate-Oriented Memory Optimization for GPU-Based CNNs, ACM Transactions on Architecture and Code Optimization

蒋文斌
教　授
研究方向：深度学习、分布式计算
Email：wenbinjiang@hust.edu.cn

马　阳
2016级硕士研究生
研究方向：深度学习系统优化技术
Email：mayangmails@qq.com

Supporting Superpages and Lightweight Page Migration in Hybrid Memory Systems

王孝远

文章发表在国际学术期刊ACM Transactions on Architecture and Code Optimization（TACO）上。该期刊每季度出版一期，每期10篇论文左右，2018年影响因子1.131，主要关注面向嵌入式和通用系统的计算机架构、编程模型和操作系统等方面的研究。该论文成果将在体系结构领域国际旗舰会议HiPEAC'20进行宣读交流。

内容概述：

应用日益增长的内存容量需求对计算机系统设计带来了巨大的挑战，如何高效的提升和利用内存容量成为了当代计算机系统亟待解决的两个重要问题。

针对内存容量提升方面，由DRAM与新型非易失存储介质（NVM）组成的异构内存因为其良好的兼容性和扩展性，受到了学术界和工业界的广泛关注。同时，为了保证内存系统的高效性，现有工作通常采用迁移的方式对异构内存的页面进行管理。具体说来，异构内存系统需要周期性的把较慢的非易失内存介质中的热数据（频繁访问的数据）迁移到DRAM中，同时把DRAM中的冷数据（非频繁访问的数据）迁移到非易失内存介质中，从而提高系统性能与降低能耗。

在大内存的高效利用方面，当代计算机系统通常使用大页来降低大内存系统中的地址转换开销。具体说来，系统通常都在软硬件两方面支持大页，比如说X86-64系统中支持4 KB、2 MB和1 GB三种分页模式，并且在处理器硬件方面设置多粒度分离式TLB以支持多粒度页面映射。在这种情况下，一个虚拟地址可以被分离式多粒度TLB并行查找，从而缩短查询开销。尽管分离式TLB实现简单、性能良好，但是这些都是基于操作系统的配合前提下。

然而，大页的使用妨碍了由DRAM和非易失内存（NVM）组成的异构内存关键技术的使用，比如细粒度的页面迁移机制。由于绝大多数的内存访问都分布在一个大页的小部分区域中，此时若以大页粒度（比如2 MB）进行页面迁移，那么迁移操作将会带来巨大的性能开销（甚至会掩盖页面迁移带来的性能收益），例如DRAM容量资源和带宽资源的巨大浪费。由此可知，异构内存中大页的使用存在一个两难的境地：轻量级页面迁移会削弱甚至抵消使用大页的性能收益。

因此我们提出了一种新型内存管理机制——Rainbow。Rainbow使用不同的粒度对管理DRAM和NVM，并采用分离式TLB机制显著加快了DRAM和NVM的地址转换速度。Rainbow在内存控制器中使用位图来标识页面的迁移状态，从而在逻辑上保证了大页的完整性。Rainbow通过采用轻量级的页面监测机制，显著的减少页面监测的开销。此外，我们设计了一套DRAM页面重映射机制，显著降低了地址转换开销。实验证明，与不支持大页的策略相比，Rainbow可以显著的降低负载的地址转换开销，同时提升多达2.9倍（平均43%）的系统性能。除此之外，Rainbow可以显著的降低系统能耗。

详细内容参见：

Xiaoyuan Wang, Haikun Liu, Xiaofei Liao, Ji Chen, Hai Jin, Yu Zhang, Long Zheng, Bingsheng He, and Song Jiang. 2019. Supporting Superpages and Lightweight Page Migration in Hybrid Memory Systems. ACM Trans. Archit. Code Optim. 16, 2, Article 11 (April 2019), 26 pages.

王孝远

2014级博士研究生

研究方向：内存计算，操作系统

Email：xiaoyuanw@hust.edu.cn

基于双页检查点的高效数据持久化系统

周　放

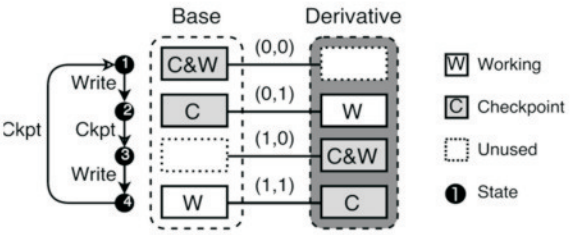
文章发表在ACM Transactions on Architecture and Code Optimization上。该期刊每季度出版一期，每期10篇论文左右，2018年影响因子是1.313，主要关注面向嵌入式和通用系统的计算机架构、编程模型、编译器和操作系统等方面的研究。

内容概述：

由于存储设备缓慢的I/O性能，数据持久化问题一直是制约应用性能的一个重要的瓶颈。新型非易失性存储NVM技术的成熟为实现高性能数据持久化带来了曙光。我们提出了一种基于硬件级检查点技术的高效内存数据持久化机制：首先，我们提出了新的内存数据一致性恢复机制，在内存中周期性地建立内存数据检查点，在系统出错之后可快速恢复到上一个检查点位置，保证应用的不间断执行。其次，提出了新的架构设计方案，利用硬件技术建立内存检查点，保证内存数据的Crash Consistency，消除软件中的一致性保障开销，提供高性能运行环境。最后，提出了创新性的双页检查点机制，突破了现有系统设计中的Tradeoff制约，同时实现了最小化的NVM写次数与最少的元数据记录开销。

双页映射机制，其核心思路是将一个逻辑页映射为两个物理NVM页，基页（Base Page）与派生页（Derivative Page）。我们对每页中所有Cacheline设置两个状态位记录其存储状态。一位代表检查点数据的存储位置；另一位为脏位。通过这种方式，避免了为每个脏Cacheline记录大量元数据信息。上图1中，表示一个Cacheline在所有运行情况下的四种数据存储状态以及对应的元数据记录方式。

双页映射机制在内存控制器中保存一个页级的映射表记录基页与派生页的映射关系，并在每一项页级映射信息后，按顺序存储该页中64个Cachelines的2个状态位。通过重复利用页级映射信息，避免了对每一个脏Cacheline进行大量的元数据记录，实现了最小的元数据记录量。每个NVM访存操作能够在控制器中快速地查找与访问基页与派生页中Cachelines的状态信息。



测试结果证明，与最新硬件检查点技术ThyNVM和PageOverlay相比，NVM写次数减少34%，元数据记录量减少9倍，性能提升1.28倍。与最新NVM软件Transaction Lib，NVML和DudeTM相比，性能提升13.6倍。

详细内容参见：

Song Wu, Fang Zhou, Xiang Gao, Hai Jin, Jinglei Ren. Dual-Page Checkpointing: An Architectural Approach to Efficient Data Persistence for In-Memory Applications. ACM Trans. Archit. Code Optim., Vol. 15, No. 4, Article 57.
DOI: https://doi.org/10.1145/3291057

周　放
2014级博士研究生
研究方向：异构内存系统管理
Email：flame@hust.edu.cn

面向低延迟的批流系统慢节点预调度机制

陈　飞

文章发表在IEEE Transactions on Parallel and Distributed Systems上。该期刊每月出版一期，每期录用论文20篇左右，2019年影响因子是3.971，主要关注并行与分布式架构、并行与分布式算法、并行与分布式计算应用以及并行与分布式系统软件等方面的研究。

内容概述：

批流系统将连续的流计算离散化为一系列小时间粒度上的微批作业，并将这些作业周期性提交给批处理引擎处理。典型的代表如Spark Streaming。当前的集群环境下，由于不同批次的硬件更新、资源共享、负载数据倾斜等原因，慢节点在批流系统中普遍存在，严重影响系统性能。

传统的批处理引擎处理慢节点的技术是为大而长的批处理作业而设计，是一种反应式的策略，在批流系统中可以归结为post-scheduling方法。这类方法不能很好地适应小而短的微批作业的处理需求，要么无法快速准确检测或预测慢节点（批处理模块可能已经将数据分发到批处理队列中）；要么无法及时重新调度慢节点上的任务（处理模块可能已经开始处理数据）。在任务执行过程中，执行重新调度不可避免地会增加任务执行时间；另外，由于数据已经分发出去，重新调度要求重新分发数据，引起昂贵的内部开销。

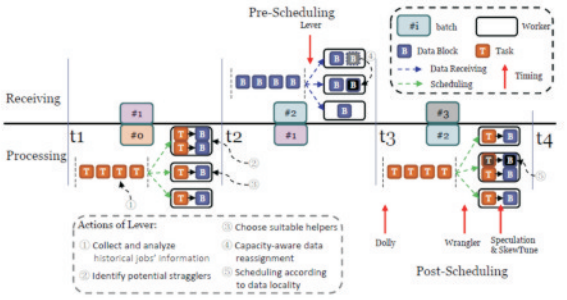


图1 Pre-scheduling 处理流程

针对上述问题，我们提出了pre-scheduling的策略Lever，如图1所示，能够准确预测慢节点并及时地做出调度决策以最小化处理延迟。首先，考虑到微批作业的周期性和慢结点局部性，我们可以通过分析历史作业的运行时信息结合负载的变化识别下一个批次的潜在慢节点；其次，引入ILC（Iterative Learning Control）模型通过多批次的迭代评估节点的计算能力；最后，针对潜在的慢节点，选择合适的帮助者节点，在Batching Module中按照各帮助者节点计算能力提前对潜在的慢节点任务进行划分和分配，在最后任务调度运行时，就可以有效的避免慢结点的产生。

我们在pre-scheduling策略部署在面向Spark的第三方开源系统Spark-Lever中，是目前Spark社区中由大陆高校贡献的唯一流处理开源项目。测试结果表明：与Spark Streaming相比，Lever能够减少30.72%~42.19%的job完成时间。

详细内容参见：

Hai Jin, Fei Chen, Song Wu, Yin Yao, Zhiyi Liu, Lin Gu, Yongluan Zhou, "Towards Low-Latency Batched Stream Processing by Pre-Scheduling", IEEE Transactions on Parallel and Distributed Systems, vol.30, no.3, pp.710-722, 2019

陈　飞
2013级博士研究生
研究方向：大规模分布式流计算系统
Email：fei_chen_2013@hust.edu.cn

基于博弈论方法的准流图划分

欧阳李原，华强胜

文章发表在IEEE Transactions on Parallel and Distributed Systems上。该期刊每月出版一期，每期录用论文20篇左右，影响因子为3.402，主要关注并行与分布式架构、并行与分布式算法、并行与分布式计算应用以及并行与分布式系统软件等方面的研究。

内容概述：

在大图上开展图计算，图划分是一个至关重要的先序步骤。已有的图划分模型包括离线图划分和流图划分。在传统的流图划分模型中，当前边需要根据先前到达边的划分块信息帮助当前边选择最佳划分块。因此，在传统流图的划分模型中很难并行地开展划分任务。另一方面，离线图划分模型在划分过程中需要知道图的全局信息，很难适用于大规模图。

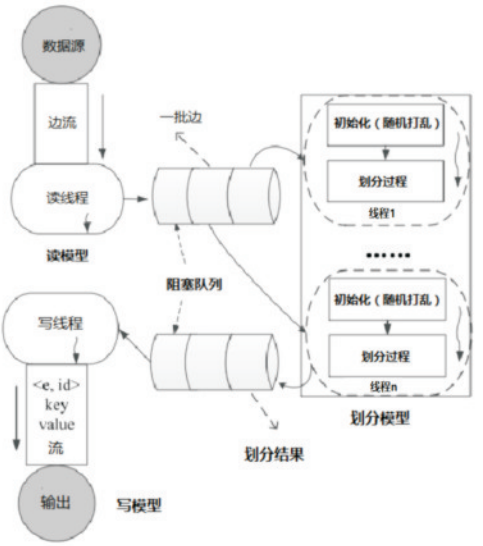


图1 准流模图划分模型

针对现有图划分模型的不足，我们提出了一种准流图划分模型，并基于该模型提出了一种基于纯策略博弈的边图划分算法。具体地，我们的准流图模型采用的是边划分模式，划分的是图中所有边的集合。将读入的边流分为很多批次，批次内每条边被视为博弈的玩家，每条边的划分块选择被视为其策略，从而将原图的划分问题分解为一系列寻找纳什均衡的过程。每个批次的边选择其最佳划分块时，只依赖于该批次内其它边的划分块选择，所以各个批次内的博弈过程可以并行的寻找纳什均衡，具体过程如图1所示。首先通过设计适当的个体代价函数和社会福利函数构造一个博弈过程，并证明了该博弈过程是一个确切势博弈，从而一定存在纯策略纳什均衡。当每批中博弈过程达到纳什均衡时，这批划分任务就完成。

通过实验论证，在多个真实图数据集和随机图数据集上，基于博弈的划分算法和已有的流图划分算法相比，顶点平均备份数指标和边数标准差均有显著改进。

详细内容参见：

Qiang-Sheng Hua, Yangyang Li, Dongxiao Yu, and Hai Jin, “Quasi-Streaming Graph Partitioning: A Game Theoretical Approach”, IEEE Transactions on Parallel and Distributed Systems, Vol. 30, No. 7, July 2019, pp.1643-1656.

欧阳李原
2018级硕士研究生
研究方向：分布式算法
Email：oyliy@hust.edu.cn

华强胜
副教授，博导
研究方向：分布式算法
Email：qshua@ hust.edu.cn

大规模时变流处理系统负载均衡机制

黄　禹

文章发表在“Efficient Time-Evolving Stream Processing at Scale”发表于IEEE Transactions on Parallel and Distributed Systems (TPDS) 2019上。IEEE TPDS 主要关注并行分布式计算算法、并行分布式应用、并行分布计算结构和并行分布计算软件等领域最新的研究进展和技术。IEEE TPDS每年出刊12期，每期15篇论文左右，2018年影响影子 3.402。

内容概述：

分布式流处理在解决许多现实问题中扮演着重要角色，如欺诈检测、实时建议及热度分析等，生成流数据的应用程序无处不在。现实世界中的流数据集通常表现出有趣的特征：它们固有的高频键值（热键）通常会随着时间的推移而演变，在某个时间间隔内某个高频键值可能在下一时间间隔内变为低频（非热键）。有效地处理这些随时间变化的流数据集对流处理的负载均衡和可扩展性有着至关重要的影响。

基于真实数据中键值的倾斜分布，将热键分配更多的工作者从而保证负载均衡；将非热键分配少量工作者，减少数据备份数量，从而降低内存开销保证系统的可扩展性。我们提出了一种新颖的负载均衡机制，该机制可以通过最近的热键识别和工作者分配来保证高效的分布式流处理。这项工作的关键观察是，随时间变化的流数据的键在时间间隔的有限距离内可能具有偏斜的分布。因此，我们提出了一种基于时间片的近期热键识别技术，它具有专门的时间片内频率计数和时间片间热度衰减。

由于需要对每个数据元进行分类，整个处理过程涉及大量的近期热键识别操作。 因此，这些识别操作应该是轻量级的，以便整个分布式流系统可以充分发挥其在负载平衡和可扩展性方面的优势。我们观察到，尽管键值的频率随时间变化，但有界范围内随时间变化的流数据集的键值表现出偏斜的幂律分布，其中一小部分键值占多数负载 。因此，这允许通过使用分层处理在有限的时间间隔内实现实时负载平衡。我们提出了一种基于时间片的方法来准确识别最新的热键。在时间片内，对键值的出现次数进行计数，仅存储最频繁使用的键值数以保持较低的内存开销。在时间片间，使用时间感知方法，对键值的频数进行衰减。

其次，处理随时间变化的流数据集可能还需要及时进行调整，以保证负载平衡。更糟糕的是，异构资源可能进一步加剧该问题。为了找到合适的工作者，服务器必须在每次负载分配时频繁地从所有工作者收集状态信息，而这需要大量的通信开销。因此很难做出有效的工作者分配决策保持实时负载平衡。为了确保工作人员分配的效率，我们发现了流处理操作的同质性，即在相同工作者中，不同批次元组的处理时间差异可以忽略不计。因此，我们进一步提出了一种启发式方法，以一种更有效的方式来推断获取远程工作者的状态信息。

通过在128个节点的群集上评估分组方法， 我们在Apache Storm上的实际部署表明，相比于当前国际上先进的流数据分组方案，其平均延迟降低了87.12%，并且能够在极小（低于4%）的内存开销下，为分布式流数据系统提供和基于轮询分组方案接近的延迟和吞吐量。

详细内容参见：

Xiaofei Liao, Yu Huang, Long Zheng, Hai Jin, “Efficient Time-Evolving Stream Processing at Scale”,IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 30, no. 10, pp. 2165-2178, 2019

黄　禹
2016级博士研究生
研究方向：存内计算、图计算
Email：yuh@hust.edu.cn

面向社交网络大数据的事件流通信优化机制

张　凡

文章发表在IEEE Transactions on Parallel and Distributed Systems上。该期刊每月出版一期，每期录用论文20篇左右，2019年影响因子是3.402，主要关注并行与分布式架构、并行与分布式算法、并行与分布式计算应用以及并行与分布式系统软件等方面的研究。

内容概述：

在线社交网络是人们分享和获取信息的主要途径之一。数以亿计的用户在社交网络上每天发布与浏览各种实时的事件信息。在线社交网络系统为了保护用户隐私，常常限制用户的浏览范围在自己或好友分享的事件之内；另一方面，也常常需要为用户提供个性化的展示结果。因此，与传统网页采用的公共视图不同，在线社交络系统需要为每个用户建立独立视图来保存该用户及其关注对象关联的数据。由于数据量庞大，用户数据通常基于一致性哈希的划分方法存储在不同服务器内，并基于内存形成相应的视图。每当有新的事件被发布，为了保证用户浏览的高时效性，系统需要及时对相关联的所有视图进行更新。图1展示了社交网络系统的事件流通信过程。由于大量关联复杂的活跃用户频繁产生和获取新事件，社交网络系统中为了更新视图而产生的跨服务器通信开销是其主要的工作负载。

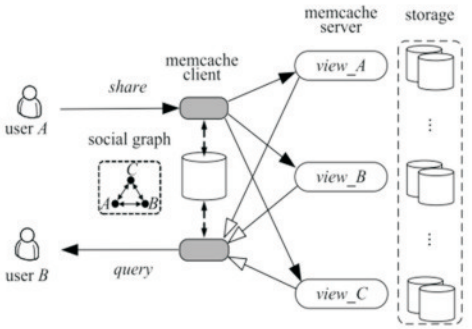


图1 社交网络系统事件流通信示意

社交网络中事件流通信有两种基本机制：一种是将用户新发布的事件流即时推送到所有好友视图中，保证用户浏览事件时的本地性，然而对于社交网络中大量“僵尸”用户而言，这种机制会产生大量不必要的通信开销；另一种将事件仅存放于本地，当用户请求时从所有好友视图中按需拉取，这种机制在用户频繁拉取时会带来显著的网络通信开销与延时。为了降低过高通信开销，一种基于“搭便车”思想，即借由公共好友视图来进行事件流通信的策略被提出。然而，现有基于组合优化的事件流通信策略分配算法，或具有极高的计算复杂度，或难以高效利用“搭便车”结构来充分降低通信开销。

本研究探究了高效的事件流通信机制，即为每一对具有好友关系的用户对选择一种高效的事件流通信策略：即时推送、按需拉取、或者“搭便车”。不同用户的不同浏览频率、公共好友的数量等因素都会为不同的通信策略带来不同的收益，因此策略的选择本质上是一种博弈。通过博弈分析方法，我们设计了一种高效的事件流通信策略分配算法，在快速求解的同时保证整体通信开销接近最优值。我们进一步设计了多种并行化方法来提升了算法的执行效率。基于此，我们搭建了在线社交网络事件流通信原型系统。通过使用Twitter和Flickr系统的真实数据进行实验，结果表明我们的方法相比现有策略显著降低了计算时间，同时大幅度降低了全局的事件流通信开销。

详细内容参见：

Fan Zhang, Hanhua Chen, Hai Jin, "Piggyback Game: Efficient Event Stream Dissemination in Online Social Network Systems." IEEE Transactions on Parallel and Distributed Systems, 30(3): 692-709, 2019

张　凡
2014级博士研究生
研究方向：大数据流处理
Email：zhangf@hust.edu.cn

面向加密数据的隐私保护多关键字Top-k相似性搜索

丁晓锋

文章发表在IEEE Transactions on Dependable and Secure Computing上。TDSC属于CCF A类期刊，2019年影响因子为6.404，是计算机及网络安全、可信计算等研究领域中顶级刊物，主要关注计算机及网络安全、可靠性领域最新的研究进展和技术。该期刊每年出版6期，共收录不到90篇文章。

内容概述：

云计算为个人和企业提供了巨大的计算能力和可扩展的存储容量来支持医疗保健和科学研究等领域中的各种大数据应用，因此越来越多的数据所有者将数据外包到云服务器上以方便数据管理和挖掘。但是，数据所有者的敏感信息被泄露或者共享给云中部分不受信任的第三方将会造成隐私威胁。目前广泛使用的数据隐私保护技术是将外包到云服务器的数据提前进行加密处理，但是这会降低数据效用并使许多传统的数据分析方式例如基于关键字的top-k文档检索不再实用。

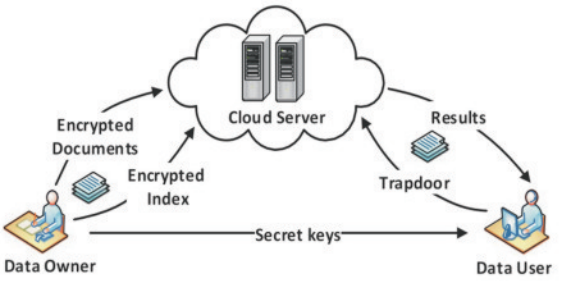


图1 面向外包加密数据的系统模型

本研究针对大数据加密抗隐私泄露提出了一种多关键字top-k搜索理念，并尝试为此问题确定有效且安全的解决方案。之前方法中随机遍历算法可以通过使用不同的密钥来改变访问路径和两个相同查询的搜索结果，从而实现数据安全性。另一方面，因为云服务器可以通过比较和分析其访问路径和搜索结果来链接两个查询，EGMTS在查询不可链接性保护方面存在缺陷。综合以上，我们提出了一种随机群组多关键词top-k搜索方案（RGMTS），它吸收了随机遍历算法安全性和EGMTS高效性的优点，并提供了比EGMTS更多的数据安全性。具体来说，出于对查询数据的隐私考虑，我们构造了一个特殊的基于树的索引结构，并设计了一个随机遍历算法，使得即便相同的查询在索引上也能产生不同的访问路径，并且还可以在隐私保护效果更强的情形下保持查询的准确性不变。此外，为了提高查询效率，我们提出了一种基于分区思想的群组多关键字top-k搜索方案，在这里我们为所有文档构造出一组基于树的索引。最后，我们将这些方法以一种有效且安全的方式结合在一起来解决我们提出的top-k相似性搜索问题。

经过在实际数据集上进行大量实验，最终实验结果证明相较于目前最新的方法，我们提出的方法可以明显提高隐私泄露的防御能力，可扩展性和查询处理的时间效率。

详细内容参见：

Xiaofeng Ding, Peng Liu, and Hai Jin, “Privacy-Preserving Multi-Keyword Top-k Similarity Search over Encrypted Data”, IEEE Transactions on Dependable and Secure Computing, Vol. 16, No. 2, March/April 2019, pp.344-357.

丁晓锋
副教授，博导
研究方向：隐私保护，查询处理
Email：xfding@hust.edu.cn

基于MapReduce的去标识化策略推荐

丁晓锋

文章发表在IEEE Transactions on Big Data上。该期刊每年出版4期，每期录用论文12篇左右，主要关注大数据分析、大数据可视化、大数据策展和管理、大数据的安全与隐私等方面的研究。

内容概述：

大数据时代，各个群体间交换和发布数据变得越来越重要。为了保证数据发布中的隐私安全，隐私策略对数据进行抽象描述，用一个大区间去替换数据属性的某个具体值。例如：将医疗数据表中年龄21替换成[20, 30]这一区间。Skyline是一种有效的平衡隐私保护和数据可用性的方法。该方法要求筛选出一系列“感兴趣”的策略，这些“感兴趣”的策略指的是那些不被其他策略“支配”的策略。被筛选掉的策略在隐私保护和数据可用性两方面都不比留下的策略好。但是随着数据取值范围的扩大，隐私策略数量会呈指数级增长。因此，大规模数据隐私策略的 Skyline 计算仍然是一个重大的挑战。

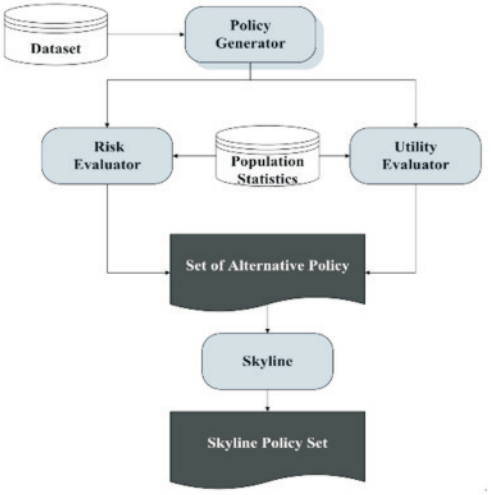


图1 去标识策略推荐框架

在本文中，我们提出了基于MapReduce的并行算法SKY-FILTER-MR。如图1所示，首先，将 MapReduce 编程模型应用到传统的隐私策略 Skyline 计算可以显著提高算法的效率，这样就可以有效地处理大规模数据隐私策略 Skyline 计算。其次，近似的Skyline是在Skyline的基础上设置一个控制计算精度的参数。它要求筛选掉的策略在隐私保护和数据可用性两个方面都不比Skyline策略集中的策略在一定的参数区间好，这样筛选掉的策略更多，将近似的Skyline应用到策略过滤可以大大降低备选策略集的规模，从而缩短了备选策略集Skyline计算的时间。同时可以通过参数设置来调节隐私保护和数据可用性的平衡。

一般而言，我们的近似方案可以保证所生成策略的风险和效用成本在一定范围内是最优的，并能够无缝调整以权衡并接近最佳的保证，从而降低风险和提高数据实用性。通过大量的实验和分析，SKY-FILTER-MR在最好的情况下，备选策略集规模下降了732倍，同时执行效率提高了4倍。SKY-FILTER-MR算法具有良好的扩展性，可以有效地进行大规模数据隐私策略Skyline计算。近似的Skyline参数对算法效率影响明显，参数越大，执行效率越高。在保证计算精度的前提下，SKY-FILTER-MR 降低了备选策略集的规模，提高了算法的执行效率。

详细内容参见：

Xiaofeng Di, Li Wang, Zhiyuan Shao, Hai Jin. Efficient Recommendation of De-identification Policies using MapReduce. IEEE Trans. on Big Data, 2019, 5(3): 343-354

丁晓锋
副教授，博导
研究方向：隐私保护，查询处理
Email：xfding@hust.edu.cn

软件定义网络中流表溢出攻击的缓解机制

袁 斌

文章发表在IEEE Transactions on Services Computing(TSC 2019)上。该期刊每季度出版一期，每期录用论文17篇左右，2018年影响因子是5.707。TSC是服务计算研究领域的顶级期刊，也是中国计算机学会(CCF)推荐的B类期刊，主要关注服务计算的理论与方法、云计算相关技术、物联网、Web服务等相关研究。

内容概述：

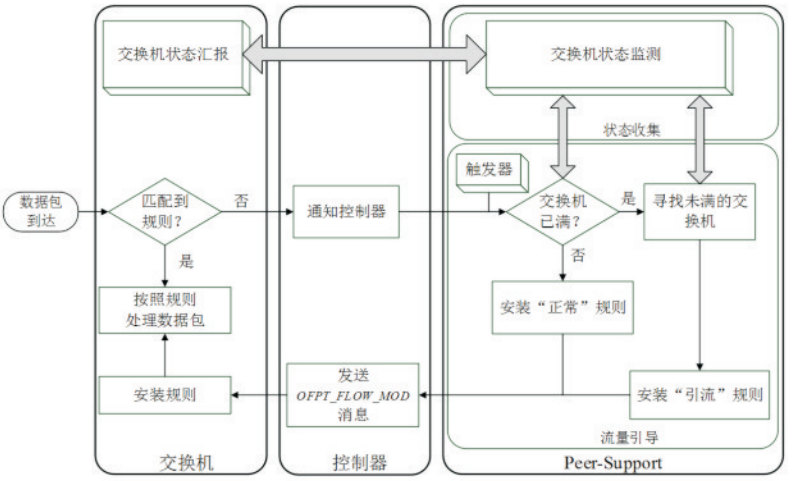


图1 peer-support策略的工作流程

软件定义网络作为一种新型的网络架构，将传统网络划分成数据面、控制面与应用面，使得网络的管理变得更加高效与灵活。数据层作为软件定义网络的基础设施层，是网络系统正常运行的基础支撑，在整个软件定义网络框架中十分重要。数据层中的流表溢出攻击是DDoS攻击在软件定义网络中的变体，该攻击利用交换机空间有限的缺陷，通过伪造数据流的方式耗尽交换机资源，使得交换机不可用。

为解决上述问题，提出了服务质量可感知的peer-support策略，该策略的工作流程如图1所示。交换机对收到的数据包进行处理规则的匹配，如果发现了匹配的规则，则按照流规则处理该数据包，否则交换机发送消息给控制器。控制器收到该消息后，首先检查对应交换机的流表空间使用情况。如果该交换机的流表空间没有被完全消耗，则按照正常的网络策略下发处理规则到交换机中以处理该数据包，否则控制器将寻找最合适的协助交换机，并安装流量引导规则，把目标交换机的网络数据包引导到该协助交换机上进行处理。针对peer-support策略的有效性和效率，分别测试了网络系统在有无该策略下以及在不同参数设定下的保持时间。

实验结果表明，应用peer-support策略后，软件定义网络系统的保持时间有显著提升，提升的平均倍数约等于交换机的个数。在不同的攻击速率下，只要网络中有足够的交换机，系统的保持时间总能超过攻击的平均时长，这意味着peer-support策略能够成功抵御攻击。

详细内容参见：

Bin Yuan, Deqing Zou, Shui Yu, Hai Jin, Weizhong Qiang, and Jinan Shen, “Defending against Flow Table Overloading Attack in Software-Defined Networks”, IEEE Transactions on Services Computing, vol.12, no.2, pp.231-246, 2019

袁 斌
博士后
研究方向：软件定义网络安全，物联网安全，云安全
Email：yuanbin@hust.edu.cn

查询定义的虚拟机镜像自动定制机制

刘海坤

文章发表在IEEE Transactions on Cloud Computing（TCC）上，该期刊每年出版四期，每期录用论文20篇左右。2018年最新影响影子5.967，主要关注云计算领域的理论、算法、系统、应用和性能。

内容概述：

随着越来越多的应用迁移到公有云环境下，云端的虚拟机镜像管理越来越成为了一个重要的挑战。首先，服务器提供商一般只会提供有限数量的虚拟机镜像模板，用户仍然需要自己配置和维护应用需要执行环境，不能简化用户应用在云端的部署流程。其次，通用的虚拟机镜像文件往往附带大量用户应用不相干的系统组件，使得虚拟机镜像文件过于臃肿，使得虚拟机启动部署的时延较大，并且浪费内存和存储资源。最后，虚拟机镜像管理操作如虚拟机镜像克隆、衍生、缓存机制等使得虚拟机镜像之间的关系非常复杂，并且虚拟镜像之间往往存在巨大的冗余信息。

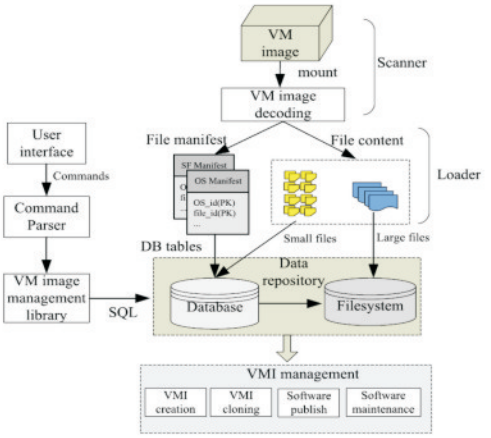


图1 虚拟机镜像文件解耦和重组

针对以上挑战，我们设计并开发了一个基于查询定义的虚拟机镜像自动定制和管理工具Hemera。如图1所示，在云服务提供商端通过对虚拟机镜像内部文件进行解耦，并通过数据库和文件系统进行组织和管理，从而可以方便地进行数据去冗余操作。同时，在云终端用户层，提供用户友好的基于SQL查询语言的虚拟机镜像配置接口，可实现用户自定义的组件化软件自动安装和配置。

在虚拟机启动时，虚拟机通过一个伪文件系统镜像（包含操作系统、应用程序的文件列表信息）来获取需要加载的内核镜像和用户程序运行时库，而并不需要真正生成一个虚拟机镜像文件，如图2所示。虚拟机镜像文件仍分散在包含数据库和文件系统的数据仓库中，这样可以保证虚拟机按用户应用程序需要的最小镜像进行启动，大幅缩短了虚拟机的启动时间。

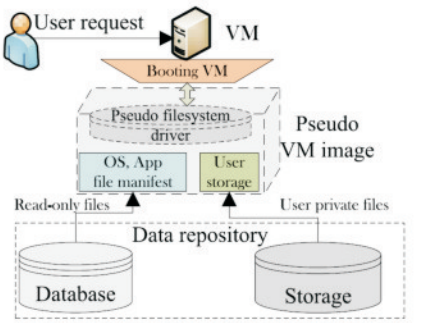


图2 虚拟机从伪虚拟机镜像启动

实验研究表明，与传统的虚拟机镜像管理模式相比，Hemera可以减小虚拟机的镜像文件大小达6.9倍，虚拟机克隆和软件安装操作比传统的虚拟机镜像管理模式快一个数量级。

详细内容参见：

Haikun Liu, Bingsheng He, Xiaofei Liao, Hai Jin, “Towards Declarative and Data-Centric Virtual Machine Image Management in IaaS Clouds”, IEEE Transactions on Cloud Computing, vol.7, no.4, pp.1124-1138, 2019

刘海坤
副教授
研究方向：云计算，虚拟化，内存计算
Email：hkliu@hust.edu.cn

基于众包的移动云服务QoS保障机制

姚德中

文章发表在IEEE Transactions on Cloud Computing上。该期刊每月出版一期，2019年期刊影响因子为5.967，主要关注云计算中的创新研究思想，应用结果和案例研究，重点关注与理论、算法、系统、应用和性能相关的关键技术问题。

内容概述：

移动云计算希望利用分布式在线计算资源使得移动设备更加强大，对于一组具有相同功能的候选服务，QoS 将在服务的组合和选择中起到至关重要的作用，所以只有当云服务具有可靠的 QoS 时，用户才能信任移动云计算。

但由于移动计算环境复杂且不可预测，缺乏对环境的了解和低效的服务发现削弱了用户对于服务的体验质量，所以通常使用基于上下文感知的方法来获取用户的环境信息，来决策选择可靠云服务，保障云服务QoS服务质量。但由于用户存在移动性，云服务提供商和网络环境的信息也在不断变化，用户无法及时选择到合适的云服务，而高频率和持续的网络环境感知则是低效率和高能耗的。

所以为了实现了在移动设备上提供QoS保证的可靠云服务支持平台，必须要明确两个问题：1）如何高效地获得上下文信息；2）如何让决策器的决策更加有效。

研究小组希望所有用户都能参与解决这些复杂问题，用户可以分享在不同的上下文中选择合适移动云服务的经验，共同完成全局环境信息。这种全局的上下文感知方法较本地感知能收集更多的服务提供商和网络环境的知识，且利用收集的信息可以更有效地实现智能云服务选择问题。同时，用户的选择结果可以用于新的更新，因此其他用户不需要继续了解环境，当他们到达新的地方时，也能很快发现合适的服务，从而避免持续检测带来的能耗增加。对此研究小组设计了一个移动云服务质量框架，利用基于众包的服务质量适配器来适应不同类型的移动应用，如图1所示。在运行过程中，移动用户的上下文感知器会定时感知环境信息，并将这些上下文信息注册到众包平台，众包平台将会维护上下文数据库存储上下文环境和可用云提供商之间的关系。

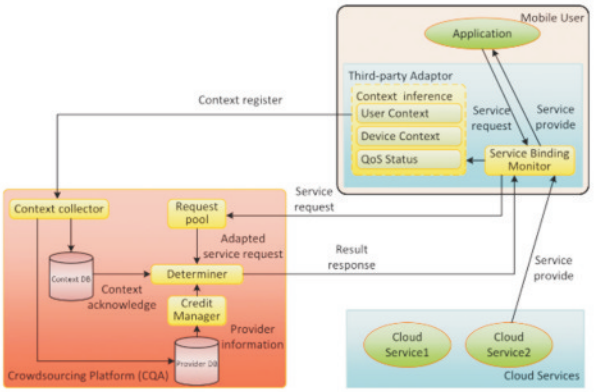


图1 移动云服务质量框架

实验结果表明，相对于单纯的高频率和持续的网络环境上下文感知，基于众包框架下的服务发现时间有缩短49%和并且在移动中的服务发现稳定性得到提高。

详细内容参见：

Dezhong Yao, Chen Yu, Laurence T. Yang, Hai Jin, "Using Crowdsourcing to Provide QoS for Mobile Cloud Computing", IEEE Transactions on Cloud Computing, vol.7, no.2, pp.344-356,2019

姚德中

副教授

研究方向：移动数据挖掘、物联网、边缘计算

Email：dyao@hust.edu.cn

云环境下隐私侵犯行为监控及取证系统

吴月明

文章发表在IEEE Internet of Things Journal 2019上。该期刊每两个月一期，每期发表论文73篇左右，是物联网领域的顶级期刊，近3年平均JCR影响因子为6.735，是JCR 1区Top期刊。

内容概述：

云平台下的隐私侵犯行为具有泄露痕迹易销毁、隐私数据难追踪等特点，已有的云取证方案存在粒度粗、拓展性差等问题，且忽略了隐私侵犯细节与行为特征，无法对隐私侵犯行为进行细粒度刻画。针对上述问题，提出了一个结合连续内存镜像分析与动态污点追踪的隐私侵犯取证系统。如图1所示：第一步，从网络流量中截获可疑应用和异常流量；第二步，通过VMI技术获取目标虚拟机软件环境配置信息，依据该信息自动化生成模拟虚拟机环境，并将可疑应用或异常流量导入该环境执行；第三步，使用连续内存镜像技术对执行过程进行粗粒度取证分析；第四步，隐私信息标记为污点，通过动态污点追踪技术对执行过程进行细粒度取证分析。

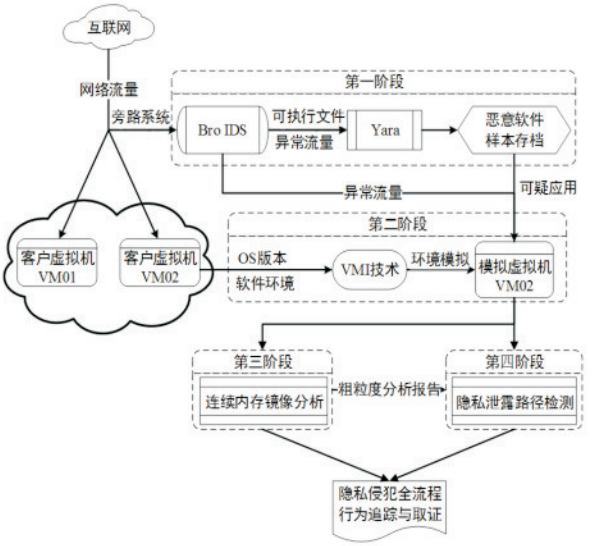


图1 云环境下隐私侵犯取证系统框架图

测试结果表明系统能够有效检测到由恶意软件或漏洞攻击导致的隐私侵犯行为，能够监控键盘记录、敏感文件和敏感内存的隐私侵犯行为，并获取详细隐私侵犯路径。通过与知名反病毒引擎腾讯Habo和KingKong进行实验对比分析，在sids、devicetree、injections、privileges方面，KingKong和Habo很少甚至没有检测到这些对象，而连续内存镜像模块则覆盖到了。针对dlls、processes 的检测，连续内存镜像分析模块比KingKong和Habo覆盖的范围要更多更广。另外，系统还对WannaCrypt 恶意软件进行了监控分析。检测结果显示样本将目标文件进行改名，增加.WNCRY后缀，然后读取敏感文件并进行加密操作。

详细内容参见：

Deqing Zou, Jian Zhao, Weiming Li, Yueming Wu, Weizhong Qiang, Hai Jin, Ye Wu, and Yifei Yang, “A Multigranularity Forensics and Analysis Method on Privacy Leakage in Cloud Environment”, IEEE Internet of Things Journal, vol.6, no.2, pp.1484–1494, 2019

吴月明
2016级博士研究生
研究方向：恶意软件检测，漏洞检测
Email：wuyueming@hust.edu.cn

Computation Offloading Toward Edge Computing

林　立

文章发表在Proceedings of the IEEE期刊上。该期刊是IEEE的权威期刊，每年出版12期，每期录用论文15篇左右，2018年影响因子10.694，主要收录电子、电气工程以及计算机科学方面的综述文章，为中国计算机学会推荐A类期刊。

内容概述：

Computation offloading—计算迁移，指计算在不同设备之间进行“卸载”。典型的计算迁移形态为“云端融合”方式：即移动设备将自身部分计算迁移至云端执行，以达到移动端节能和提升计算执行速度的目的。然而由于云数据中心与终端设备距离较远，导致网络延迟较大。因此，这种方式不适应于当前一些对于延迟要求较高的应用，如：实时视频分析、车联网应用等。边缘计算的出现，将计算由云推向网络边缘，大大降低了网络延迟，同时也有效减少了数据中心的网络负载。边缘计算和云计算的结合形成一种异构的多层次计算架构，而计算迁移的流动也随之发生变化，如图1所示。

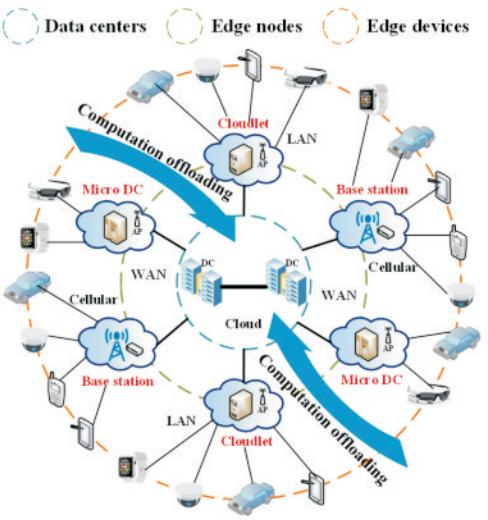


图1 云边端融合计算的体系结构

本文首先分析了在多层体系架构下复杂异构网络中计算迁移执行时间和能耗的收益，进而阐述了计算迁移应用于边缘计算过程中，在应用程序任务划分、任务分配以及任务执行等方面存在的挑战和解决方案。在引入边缘节点之后，计算迁移的这几个环节在体系结构、任务划分方式、调度策略、资源隔离粒度等方面都发生了深刻的变化。此外，由于容器以及无服务器架构等技术的发展，计算迁移的任务执行方式也发生了改变。

论文随后总结了计算迁移技术在实时视频分析、智能物联网、智能车联网应用以及云游戏等四个场景中如何被应用。典型的方式是一种云边端的计算融合。例如，在视频分析应用中，视频数据被传输至边缘节点实施即时分析并进行领域相关的应用，而分析的结果和元数据进一步传输至数据中心进行存储和大数据分析。

最后，论文探索了计算迁移未来的机遇和发展方向。首先，在编程模型方面，为了适配边缘计算和云计算融合的多层体系架构，将出现通用的或者针对某一特定领域的编程模型（如：云边融合的分布式深度学习模型训练）。其次，若计算迁移形成一种服务（Offloading as a Service），边缘节点的资源管理以及移动设备跨节点的计算迁移服务维持等都存在挑战。此外，由于边缘节点松耦合的管理方式并且包含较多的隐私信息（如位置信息，网络上下文等），在安全和隐私保护方面也存在诸多的挑战。

详细内容参见：

Li Lin, Xiaofei Liao, Hai Jin, and Peng Li. "Computation offloading toward edge computing." Proceedings of the IEEE, vol. 107, no. 8, pp: 1584-1607, 2019.

林　立
2012级博士研究生
研究方向：云端融合计算、边缘计算
Email：llin@hust.edu.cn

面向网络功能虚拟化的服务公平性和服务效用优化机制

顾　琳

文章发表在IEEE Journal on Selected Areas in Communications上。该期刊每月出版一期，每期录用论文20篇左右，2019年影响因子是9.302，主要关计算机注网络和通信方面的研究。

内容概述：

对于网络服务提供商而言，轻量级服务是一种高效提供服务的方法，而网络功能虚拟化（Network Function Virtualization, NFV）的出现使其成为可能。通过网络功能虚拟化技术，网络服务提供商可以以特定顺序组合多个虚拟化网络功能（Virtualized Network Functions, VNF）进而构成网络服务链（Network Service Chain, NSC），将传统的基于硬件的网络功能软件化为VNF，有效解决了部署网络服务对定制化硬件的依赖，在促进网络的开放性、创新性、灵活性和可扩展性等方面具有巨大的潜力。

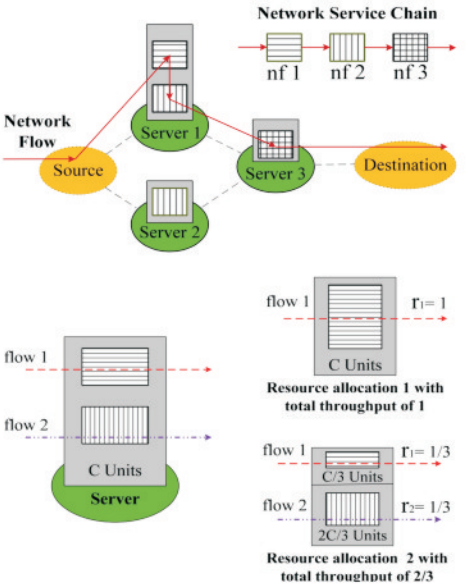


图1 服务链的多跳动态队列和服务公平性问题

然而，当对服务链进行实时调度时，面临以下三个难点：1）网络服务是多个网络功能的多跳队列，任一个网络功能都有可能成为服务链的瓶颈。2）网络流通过不同网络功能处理后，其数据量会有缩放，会产生流量爆发并增加服务时延。3）网络功能会被多个网络服务共享，独立的对各个服务进行优化会导致服务的不公平，如图1所示。

因此，一个实时的NFV调度器需要解决三个问题：1）每个VNF应该分配多少资源？2）网络流数据包是否需要被分流并转发到几个VNF上进行处理？3）如何平衡每个VNF上的吞吐量和服务的公平性？为了解决上述问题，本文研究了NFV网络下的公平感知的动态接入控制和网络流调度问题。首先，分析网络服务的服务功能之间的关联性和顺序性，建立了多跳动态队列的网络模型，提出网络功能为元单位的细粒度网络包分发策略。其次，基于建立网络模型，使用李雅普诺夫框架分析求解模型，提出了一种综合考虑网络吞吐率和公平性的网络流动态调度算法，通过最优性分析证明该算法能够在网络性能和稳定性间达到[O(1/V)，O(V)]的权衡。最后，设计了一个基于DPDK作为I/O框架的NFV网络流处理模拟系统，并将提出的动态调度算法应用到该网络流处理系统中。实验表明相比传统的处理模式，该调度算法支持下的网络系统的性能能够达到最优，并有效降低了27%的服务开销。

详细内容参见：

Lin Gu, Deze Zeng, Sheng Tao, Song Guo, Hai Jin, Albert Y. Zomaya," Fairness-Aware Dynamic Rate Control and Flow Scheduling for Network Utility Maximization in Network Service Chain",IEEE Journal on Selected Areas in Communications, vol.37, no.5, pp.1059-1071, 2019.

顾　琳
讲　师
研究方向：分布式系统
Email：lingu@hust.edu.cn

Risc-v介绍及工具链实践

（彭　平　https://blog.csdn.net/ALLap97/article/details/103365502）

RISC-V简介：

RISC-V是由加州大学伯克利分校于2010年开发的全新的指令集架构，它不像X86，ARM一样，使用需要巨额的授权费，而是以BSD许可证发布，完全开源，免费开放。同时在设计上吸取了X86, ARM等指令集的经验，并且没有历史包袱，如今备受关注。

加州大学伯克利分校在2015年成立非盈利组织RISC-V基金会，该基金会旨在聚合全球创新力量共同构建开放、合作的软硬件社区，打造RISC-V生态系统，构建硬件版的“Linux”。近年来，谷歌、高通、IBM、英伟达、NXP、西部数据、Microsemi、中科院计算所、麻省理工学院等100多个企业和研究机构先后加入了RISC-V基金会。

而国内自中美贸易战以来，中兴及华为的事件也提醒着我们自主芯片行业的重要性，RISC-V这个全球开放的架构没有知识产权的限制，对中国而言是一个很好的机会。近几年来，中国RISC-V产业联盟和中国开放指令生态系统(RISC-V)联盟相继成立，RiSC-V在国内也在不断铺开发展。

RISC-V特点：

事实上RISC-V并不是第一个免费或发放的指令集架构，在这之前也有诸如openRISC及SPARC等开放的指令集架构。但为什么RISC-V吸引了业界这么多关注呢？RISC-V架构在设计上的各种优点才真正使之得到了众多专业人士的青睐好评和众多商业公司的相继加盟，进而成为至今为止最具备革命性意义的开放处理器架构。

1. 简洁

对于目前主流的x86与ARM架构，经过几十年的发展，其架构文档长达几百数千页。其中一个主要的原因是因为其架构的发展的过程也伴随了现代处理器架构技术的不断发展成熟。同时作为商用的架构，为了能够保持架构的向后兼容性，其不得不保留许多过时的定义或是当时为满足特殊场景下的一些设定，从而使得整体冗长。

而近几年推出的RISC-V架构，则具备了后发优势。由于计算机体系结构经过多年的发展已经相对成熟，因此新的RISC-V架构在设计上能够规避一些指令集架构设计上的问题，并且没有背负向后兼容的历史包袱，因此整体上显得相对精简。

目前的“RISC-V架构文档”分为“指令集文档”和“特权架构文档”，两者在篇幅上不过一两百页，熟悉体系结构的工程师仅需一至两天便可将其通读。虽然“RISC-V的架构文档”还在不断地丰富，但是相比“x86的架构文档”与“ARM的架构文档”，RISC-V的篇幅可以说是极其短小精悍。

2. 模块化

与以往增量拓展的ISA不同，RISC-V是模块化的。它的核心是一个名为RV32I的基础ISA，运行一个完整的软件栈。RV32I是固定的，永远不会改变。这为编译器编写者，操作系统开发人员和汇编语言程序员提供了稳定的目标。模块化来源于可选的标准扩展，根据应用程序的需要，硬件可以包含或不包含这些扩展。这种模块化特性使得RISC-V具有了袖珍化、低能耗的特点，而这对于嵌入式应用可能至关重要。以下为其指令集的组成：

- 基本整数指令集

RV32I：基本整数指令集，可被一个或者多个可选指令集扩展进行增强，不能被重新定

义。包含了整数计算指令、整数load、整数store和控制流指令。32位地址空间与整数指令，支持32个通用寄存器

RV32E：嵌入式系统设计的RV32I的简化版本，将通用整数寄存器数量减少到16个。

RV64I、RV128I：将整数寄存器和支持的用户地址空间扩展到64/128位

• 标准扩展指令集

M：标准整数乘法和除法扩展，增加了对保存在整数寄存器中的值进行乘法和除法的指令。

A：标准原子指令扩展，增加了对存储器进行原子的读、修改和写操作的指令，以支持处理器见的同步。包括内存原子操作+加载保留/条件存储

F：标准单精度浮点指令，增加了浮点寄存器、单精度计算指令、单精度load和store指令。

D：标准双精度浮点指令，扩展了浮点寄存器，并增加了双精度计算指令、load和 store 指令。

C：压缩指令：解决代码体积相对较大的问题，指令编码长度为16位，将一部分普通最常用的32位指令中的信息进行压缩重排，每条短指令必须和一条标准的32 位RISC-V 指令一一对应。

V（增加矢量操作）、B、J、P……

3. 易于编译/链接

由于寄存器中的数据访问起来要比存储器中的快得多，为了获得良好的性能，变量应该尽量存放在寄存器而不是内存中，但同时也要注意避免频繁地保存和恢复寄存器，因为它们同样会访问内存。这件事在有许多寄存器的时候变得更加容易，x86-32 只有 8 个寄存器，ARM-32 有 16 个寄存器，而RISC-V 有32 个整型寄存器。毫无疑问，有了更多的寄存器，编译器和汇编程序员的工作会更加轻松。

Risc-v工具链实践

介绍了这么多，决定在Linux上配置一下Risc-v相关的工具链来具体实践一下。

首先下载Riscv-tools

```
git clone https://github.com/riscv/riscv-tools.git
cd riscv-tools
git submodule update --init --recursive
```

Riscv-tools由以下几个部分组成：

1. riscv-isa-sim：包含Spike, 一个RISC-V ISA的模拟器

2. riscv-opcodes：包含了标准的RISC-V指令操作码及控制状态寄存器的枚举

3. Riscv-pk：包含一个RISC-V代理内核（pk）及用于连接RISC-V系统的引导加载程序（bbl）

4. Riscv-openocd：提供片上编程及调试的支持

5. Riscv-tests：一些指令级别的测试

然后下载并编译RISC-V与c、c++语言的交叉编译工具riscv-gnu-toolchain。

```
cd riscv-tools
git clone--recursive https://github.com/riscv/riscv-gnu-toolchain
# 设置环境变量
export RISCV=".your.path./riscv-tools/riscv-gnu-toolchain"
export PATH=$PATH:$RISCV/bin
#默认生成64位的编译工具：riscv64-unknown-elf-***
./configure --prefix=$RISCV
make -j4
#附带生成32位的编译工具：riscv32-unknown-elf-***
./configure –prefix=$RISCV –with-arch=rv32gc –with- abi=ilp32d
make -j4
```

此时得到编译工具riscv64-unknown-elf-gcc，能够编译 c 文件，但生成的二进制文件还无法正确运行。

```
riscv64-unknown-elf-gcc helloworld.c -o helloworld
```

最后在riscv-tools目录下运行build.sh编译整个工具链。

得到整个工具链后便可以通过工具运行上一步生成的二进制文件了

```
spike pk helloworld
# 输出
bbl loader
Hello World!
```

内核OverlayFS——文件读写

（杨晓礼 https://arkingc.github.io/2017/12/26/2017-12-26-linux-code-overlayfs-read_write/）

内核版本：4.4.68

文件（一般文件或目录）的读写和文件的创建删除不同，文件的创建删除作用于inode结构，需要用到inode操作。而文件的读写则是作用于file结构，因此需要用到file操作

通常，**读写文件之前需要先打开这个文件**，通过open系统调用，传入表示文件的路径字符串，随后内核会根据路径字符串进行路径查找，最终得到一个表示文件的file结构，并使之与当前进程相关联，之后的读写便会根据这个file结构来进行，这个结构的f_op字段是一个file操作的指针，file操作包含了文件读写函数的指针。OverlayFS中文件的读写也不例外，但是在OverlayFS中，一般文件的读写和目录的读写又有区别：

一般文件的读写：在OverlayFS中，文件来源于upper层或lower层。在打开文件时，OverlayFS实际上打开的是upper层或者lower层的文件，因此随后发生的读写也就是对upper层或lower层的文件。

目录的读写：与一般文件不同的是，目录的读写相对复杂。因为对于一个同时存在upper层和lower层的目录，在对该目录进行搜索时，OverlayFS会对upper层和lower层的目录分别进行搜索，然后做一些合并处理。但是对于同时存在于upper层或lower层的文件来说，由于“上层覆盖下层”，因此只需要读取upper层中文件就行了。

正因如此，**对于目录的读取，OverlayFS定义了自己的file操作，而对于一般文件的读写，OverlayFS不需要定义自己的操作**

获取 file 操作

文件的读写函数由file操作中的函数指针来索引，因此要了解文件读写到底调用了什么函数就需要弄清楚文件何时、如何获得自己的file操作。而文件的file操作又由file结构的f_op字段指向，所以可以看看在路径查找过程中，获取表示文件的file结构时，如何为f_op字段赋值

在文件打开过程中，解析完路径字符串时，会调用到vfs_open()函数：

```
int vfs_open(const struct path *path, struct file *file,
        const struct cred *cred)
{
    struct inode *inode = vfs_select_inode (path->dentry,
file->f_flags);
    if (IS_ERR(inode))
        return PTR_ERR(inode);
    file->f_path = *path;
    return do_dentry_open(file, inode, NULL, cred);
}
```

这个函数会选取一个inode，设置file的f_path字段，最后调用函数do_dentry_open()，这个函数会进一步设置file的其它字段，包括f_op，最后执行文件打开操作：

```
static int do_dentry_open(struct file *f,
                struct inode *inode,
                int (*open)(struct inode *, struct file *),
                const struct cred *cred)
{
    ...
    f->f_inode = inode;
    f->f_mapping = inode->i_mapping;
    ...
    f->f_op = fops_get(inode->i_fop);
```

```
        ...
        if (!open)
                open = f->f_op->open;
        if (open) {
                error = open(inode, f);
                if (error)
                        goto cleanup_all;
        }
        ...
}
```

从代码中可以看出，**file结构的f_op字段使用inode的i_fop字段进行赋值。因此文件的inode决定了file操作。**通过函数vfs_open()发现，传入的inode由vfs_select_inode()决定：

```
static inline struct inode *vfs_select_inode (struct dentry *dentry,
                                        unsigned open_flags)
{
        struct inode *inode = d_inode(dentry); //dentry->d_inode
        if (inode && unlikely(dentry->d_flags & DCACHE_OP_SELECT_INODE))
                inode = dentry->d_op->d_select_ inode (dentry, open_flags);
        return inode;
}
```

dentry表示路径字符串最后一个分量的目录项，也就是OverlayFS中，所需打开的文件的目录项。对于OverlayFS中的目录项，函数的if条件成立，因此会进一步调用dentry操作的d_ select_ inode()函数，OverlayFS实现了对应的函数：

```
struct inode *ovl_d_select_inode(struct dentry *dentry, unsigned file_flags)
{
        int err;
        struct path realpath;
        enum ovl_path_type type;
        if (d_is_dir(dentry))
                return d_backing_inode(dentry);
        //如果不是一个dir，则会执行后面的代码
        type = ovl_path_real(dentry, &realpath); //获取真实路径
        if (ovl_open_need_copy_up(file_flags, type, realpath.dentry)) {
                err = ovl_want_write(dentry);
                if (err)
                        return ERR_PTR(err);
                if (file_flags & O_TRUNC)
                        err = ovl_copy_up_truncate(dentry);
                else
                        err = ovl_copy_up(dentry);
                ovl_drop_write(dentry);
                if (err)
                        return ERR_PTR(err);
                ovl_path_upper(dentry, &realpath);
        }
        if (realpath.dentry->d_flags & DCACHE _OP_SELECT_INODE)
                return  realpath.dentry->d_op->d_select_inode (realpath.dentry, file_flags);
        return d_backing_inode(realpath.dentry);
}
```

通过这个函数，我们可以发现，对于目录和一般文件，返回的inode不同：

目录：如果是一个目录，则直接返回dentry->d_inode。因为dentry是OverlayFS中所要打开文件的目录项，所以其指向的inode为OverlayFS下，所要打开文件对应的inode

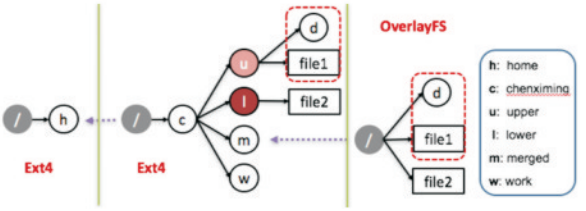
一般文件：如果是一个一般文件，则会根据打开标志以及对应upper或lower层文件的目录项决定是否要执行copy-up操作。比方说，如果是一个lower层的一般文件，以读写方式打开，则会将这个文件拷贝至upper层。如果是以只读方式打开，则不需要。然后根据最终upper层或lower层相应文件的目录项的d_flags字段决定是否需要进一步调用d_select_inode()函数。如果不需要则返回upper层或lower层相应文件的inode

分析到这已经可以得出结论了，对于一个目录，函数返回OverlayFS中该目录的inode，然后

根据inode的i_fop字段来决定文件的读写操作。对于一个一般文件，返回的并不是OverlayFS中文件的inode，通常情况下，返回的是upper或lower层对应文件的inode

一般文件读写

假设要对OverlayFS下的file1进行读写，在文件打开过程中，会得到一个表示file1的file结构，函数vfs_select_inode()会返回upper目录下file1的inode，然后根据这个inode设置file结构的f_inode字段，根据inode的i_fop字段来定义file操作，因此，实际上打开的是upper目录下的file1文件。由于upper目录下的file1是ext4下的一个文件，因此file操作由ext4实现，今后读写OverlayFS中的file1，实际上是对upper目录中的file1进行读写



目录读写

假设要对OverlayFS下的目录d进行读写，在文件打开过程中，会得到一个表示d的file结构。和一般文件不同，函数vfs_select_inode()返回的inode就是OverlayFS下目录d对应的inode。接着会根据这个inode设置file结构的f_inode字段，根据inode的i_fop字段来定义file操作，因此，实际上打开的文件就是OverlayFS下的目录d，而不是upper目录下的目录d。今后读写OverlayFS中的目录d并不会转化为读写upper下目录d，相应的file操作由OverlayFS实现

通过OverlayFS的inode分配函数也可以看出，如果不是为目录分配inode，则不会设置inode的i_fop字段。因为实际的读写操作作用于upper层或lower层的相应文件，因此不需要OverlayFS中文件的inode。但是为目录分配inode时，会设置i_fop字段，在目录读写时，为file结构的f_op字段赋值：

```
struct inode *ovl_new_inode(struct super_ block *sb,
                umode_t mode,
                struct ovl_entry *oe)
{
    struct inode *inode;
    inode = new_inode(sb);//在sb下分配一个inode
    if (!inode)
        return NULL;
    inode->i_ino = get_next_ino();//设置inode号
    inode->i_mode = mode;//设置mode
    inode->i_flags |= S_NOATIME | S_NOCMTIME;
//设置flag
    mode &= S_IFMT;
    //根据inode类型设置inode操作
    switch (mode) {
    case S_IFDIR:
        inode->i_private = oe;
        inode->i_op = &ovl_dir_inode_operations;
        inode->i_fop = &ovl_dir_operations;
        break;
    case S_IFLNK:
        inode->i_op=&ovl_symlink_inode _operations;
        break;
    case S_IFREG:
    case S_IFSOCK:
    case S_IFBLK:
    case S_IFCHR:
    case S_IFIFO:
        inode->i_op = &ovl_file_inode_operations;
        break;
    default:
        WARN(1, "illegal file type: %i\n", mode);
        iput(inode);
        inode = NULL;
    }
    return inode;
}
```

实验室师生参加2019中国计算机大会（CNCC 2019）

刘　芹

10月19日，为期三天的CNCC2019在温婉的江南水乡、历史文化名城苏州画上了圆满的句号。作为已连续多年积极参加CNCC的参与单位，实验室师生300余人参加了本次会议。

本届大会由中国计算机学会主办，苏州工业园区管委会、苏州大学联合承办，以“智能+引领社会发展”为主题，包括15场特邀报告、79场前沿技术论坛、22场特色活动及100多项科技成果展览展示，400多位国内外计算机领域知名专家、企业界权威人士到会演讲，参会规模超过8000人，达到历史最高。

CCF第十二届会员代表大会暨理事会换届选举会议于10月20日圆满落下帷幕，实验室主任金海教授当选副理事长。

颠覆性技术是关乎国家竞争力和国际地位的重大课题。10月17日下午，由实验室主任金海教授主持的“CCF会士论坛——颠覆性技术创新”成为CNCC前沿技术论坛及特色活动中最独特的一道风景。四位CCF会士：中科院计算所研究员、中国科学院大学徐志伟教授、浙江大学之江实验室副主任鲍虎军教授、清华大学计算机系舒继武教授和北京交通大学于剑教授从计算系统、虚拟现实、内存计算和图灵测试等方面阐述了他们对各自领域颠覆性技术的思考和展望。金海教授对四位CCF会士的报告作了高度评价和精彩点评。

实验室陆枫副教授于10月19日下午在“面向新工科教育的实践教学改革与探索”论坛中作了“基于智能分析的并行计算系统能力培养MOOP实践”的报告，分享了面向并行计算思维与系统能力培养的MOOP实践平台建设成果和经验，以实践环境工业化、实践项目挑战化、实践评估智能化推进实践教学改革的发展。

会议期间，实验室师生除参加大会主题论坛、听取特邀报告外，还根据自己关注的研究方向，分别选择了人工智能、网络安全、云计算、大数据、边缘计算等方面的计算机前沿科学技术主题，参与分论坛，博士生陈聃、张信民等与各领域专家学者、企业界人士积极交流沟通，并参观了多家IT企业的最新应用成果展示。

通过本次CNCC2019参会，实验室师生收获颇丰，不仅深刻了解了国内外计算机领域最新进展和宏观发展趋势，学习了学术界、产业界最前沿的科研成果，还通过与专家学者和企业界人士交流沟通，领略了真正的学术精神，拓宽了学术视野，为今后的科研工作明确了方向。

CNCC2019大会后，CCF第十二届会员代表大会于10月19-20日召开。来自全国各地的会员代表、理事候选人共计450余人出席了大会。会员代表以无记名投票的方式，选举产生了CCF第十二届理事会理事长、副理事长、理事以及新一届监事会监事长和监事。最终包括我实验室主任金海教授在内的3位候选人当选副理事长，另2位分别为清华大学胡事民教授、微软亚洲研究院副院长周明博士。

换届选举不是简单地产生新一届理事会，而是一次完善学会治理架构和制度的重要契机，是将有很高专业水平，有管理能力和对CCF高度认同的会员选举组成理事会的大事。

刘　芹
主要负责实验室宣传、科研项目管理等办公室行政事务
Email：liuqin19@hust.edu.cn

RFAcc: A 3D ReRAM Associative Array based Random Forest Accelerator

李会则 推荐

本次我推荐的是匹兹堡大学杨俊教授团队在ICS 2019上发表的一篇文章，文章提出了使用基于新兴的非易失存储器ReRAM阵列来设计更低延时，更低能耗的随机森林加速器。

随机森林在解决分类和预测问题上应用广泛，但是训练一个随机森林往往需要很长的时间，尤其是在基于CPU的平台上训练一个大规模随机森林更是可能要花费几天时间。训练过程缓慢受到以下几点影响：1、密集的内存访问和大量的比较操作；2、工作线程有限且存在大量的错误预测。而在基于GPU的平台上训练随机森林同样存在着内在的分支低性能等问题。而由于FPGA内存容量受限，FPGA同样无法提供理想的加速性能。基于现有的问题和最近很热门的PIM结构，他们提出了一种加速随机森林训练的专用结构：RFAcc，为了实现这种全新的专用结构，他们设计了一个基于3D ReCAM阵列（图一所示）的关联比较器，如图二所示，这个比较器可以将寄存器里的一个数与一个向量中的每一个元素按照从最高位到最低位的顺序依次比较大小以达到极高并行度实现关联比较的目的。之后他们利用提出的3D关联比较器，构建了RFAcc用来加速随机森林的训练。

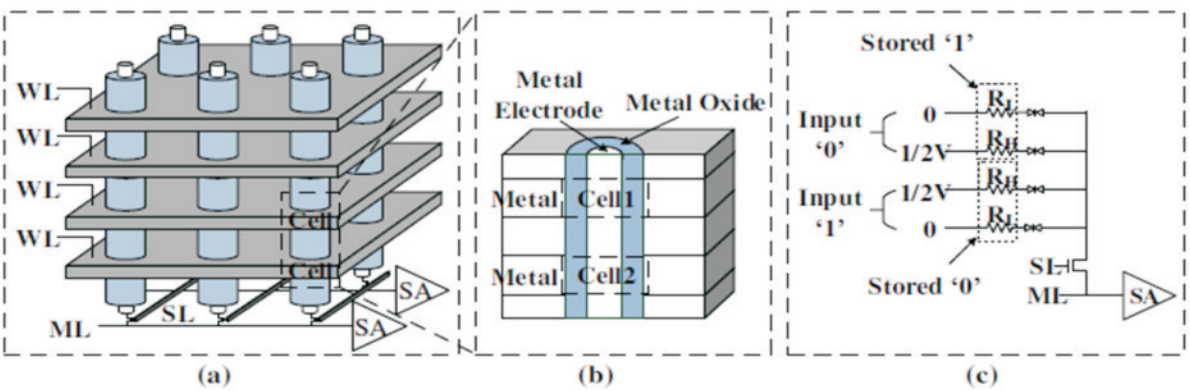


图1 3D ReCAM

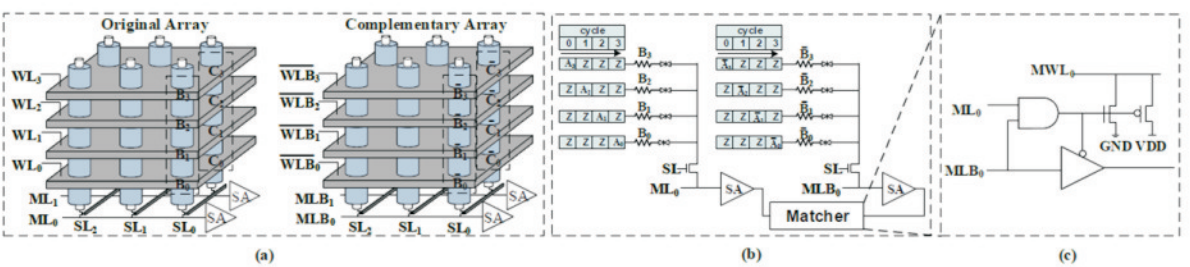
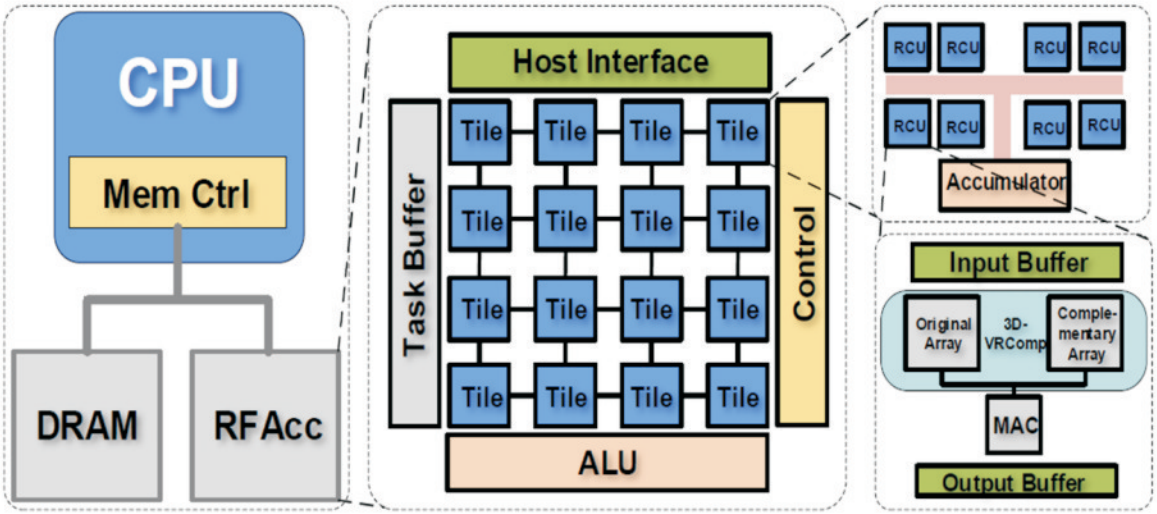


图2 关联比较器的设计



图三 系统整体设计

在设计中面临了一些问题，于是他们又提出了3种优化策略来最大化系统性能，首先他们采用一元编码来提高比特级比较并行度，其次他们提出了一种流水线设计来提高比较输出，最后他们同时训练多个决策树节点来最大化利用RFAcc的资源。

系统整体的设计图显示在图三。整个流程为CPU将配置文件传输给RFAcc以进行训练，RFAcc完成所有训练后再将训练后的森林返回给主机CPU。整个RFAcc作为主机系统的内存来进行工作，每个RFAcc芯片都包含一个任务寄存器，一个ALU和一个控制单元。系统一次只训练一棵树，但是多个节点可以同时训练这棵树从而使系统系统最佳。任务寄存器记录了正在被训练的树的信息。ALU是用来做计算以决定某个特征值对的划分是否合理。上文提到的关联比较单元（RCU）是系统的基本组成单元，多个RCU组成了一个内存Tile而多个内存Tile就组成了RFAcc

文中还详细描述了训练随机森林的7个步骤：任务寄存器初始化；选择RCU；初始化RCU；进行比较；对每个子树的标签进行计数；计算Gini以及最后的初始化子树节点，由于篇幅原因，具体的操作在这里不做赘述。

在实现部分，他们设计了四种配置的RFAcc分别与基于CPU和GPU的随机森林加速器进行对比，采用RAPL技术对CPU能耗进行估计同时采用Nvidia提供的模拟器来估计GPU能耗，而RFAcc的响应时间和能耗是采用一种周期精确的方法进行估计的，所有的3D ReCAM阵列的参数都是采用NVSim进行模拟获得的，在数据集的选取方面他们选取了多种规模多种类型的数据集使实验更加科学。实验结果表明RFAcc在与基于CPU和GPU的加速器对比中，可以达到8564和16850的加速比，同时可以有4到5个数量级的能量节省。

李会则
2017级博士研究生
研究方向：基于新型内存的内存计算
Email：huizeli@hust.edu.cn

DECAF++: Elastic Whole-System Dynamic Taint Analysis

赵浩钧 推荐

“DECAF++: Elastic Whole-System Dynamic Taint Analysis”是国际高水平会议RAID在2019年收录的一篇论文，该文在之前已有的工作基础上，提出了一种弹性的全系统动态污点分析的方法，在保证精度和准确度的同时，尽可能地减少污点分析的开销，大幅度提高了Decaf的性能，使其在动态污点分析方面有更好的表现。

动态污点分析是在程序运行过程中，以运行时的机器指令流为依据，对系统内不信任的数据进行信息流跟踪，以监控系统安全的状态。其本质是信息流传播的记录。

动态污点分析的主要过程由三个阶段组成，如图1所示。1）污点标记：识别污染的来源进行标记。如果数据来源是不可信的，那么将其记为被污染的（Tainted）。2）污点传播：根据正确的传播规则，将污点数据随着程序的运行进行污点的标记及传递（或清除）。3）攻击检测（策略及规则）：制定合理的规则与策略，通过这些规则和策略，检测出程序运行中受到的各种攻击。

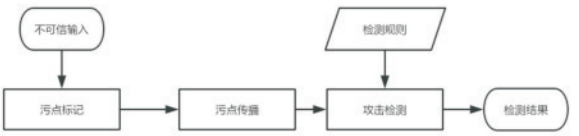


图1 动态污点分析的主要过程

在进行污点分析的时候，通常借助shadow memory来存储污点信息，即维护一段存储区域，以二级树结构形式映射物理内存，记录对应内存位置的污点信息。

本文系统地分析了动态污点分析的开销，以动态污点分析工具Decaf为例，确定了污点状态检查与污点传播是开销的两个主要来源，针对这两个方面提出了一种弹性的全系统动态污点分析方法，并落地实现了Decaf++并投入测试使用。

在已有的工作中，大部分都着眼于硬件加速的方法提高分析效率，因此这些优化往往受限于硬件的架构与功能。本文采取纯软件的方法，调整分析模式实现性能提高。针对污点传播开销与污点状态检查开销提出了弹性污点传播和弹性污点状态检查。

弹性污点传播：如果正在进行污点分析的操作并不改变任何污染状态值，则可以对这些指令跳过污点分析，从而减少因为污点传播判断而产生的开销。Decaf++根据是否发生污点传播定义了两种检测模式：跟踪模式与检查模式，如图2所示。当系统中不存在污点数据的时候以检查模式运行，当存在来自污染源的输入的时候，切换到跟踪模式进行判定,直到污染传播收敛，shadow memory全部清零（即系统中不再存在污点数据）的时候再切换回检查模式执行。即执行模式取决于内存是否被污染。

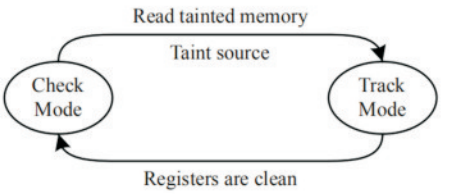


图2 Decaf++检测模式切换

对于两个模式的切换，Decaf++也做了细腻

的优化。由于监控寄存器的开销较大，需要在对每条指令检测前都要检查寄存器污染状态，开销巨大。所以Decaf++在检查模式下，只监控污染源（检查模式下系统内不会产生污点，而只可能由外部传入）；在跟踪模式下，较少的去检查寄存器（频繁的检查而获得确切的污点收敛时间来停止检测的时间开销远大于少量的检查且容忍短时间的无污染检测开销），一旦发生了污点的产生或收敛便进行模式切换。

弹性污点状态检查：在污点分析的时候，不可避免的要与shadow memory进行交互，通常对每次内存操作都要发起污点状态的检查，访问shadow memory。这就带了来了大量的时间花费。设想当进行一个较大内存地址的检查时，若较大内存集合中不包含污点信息，那么就可以安全的跳过对该集合每个地址进行检查。Decaf++在TLB中加载物理页面时扫描物理页面，决定是否进一步检查各个内存地址，如图3所示。通过修改QEMU的TLB填充逻辑，如果页面包含受污染的字节，则通过一些TLB控制位为页面设置shadow memory handler，之后无论何时访问到此页面，Decaf++都会将请求重定向到shadow memory handler。

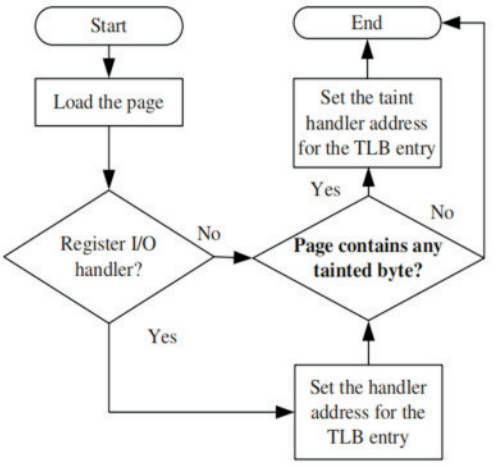


图3 Decaf++扫描物理页面流程

同样在污点状态检查的处理中也有一个优化的细节在于：对内存加载与内存存储操作的处理有细微区别。

Decaf++修改了底层QEMU的内存加载存储操作，如图4所示，当页面TLB控制位包含shadow memory handler的时候存储器加载器操作的地址转换过程导致TLB未命中，跳转到TLB miss处理，此时若控制位指示调用shadow memory处理程序则执行操作，并从shadow memory中加载相应地址的污点状态。如果TLB-check命中，即会调用shadow memory handler，所以根据是否处在跟踪模式选择加载0为污染状态或不加载污染状态。

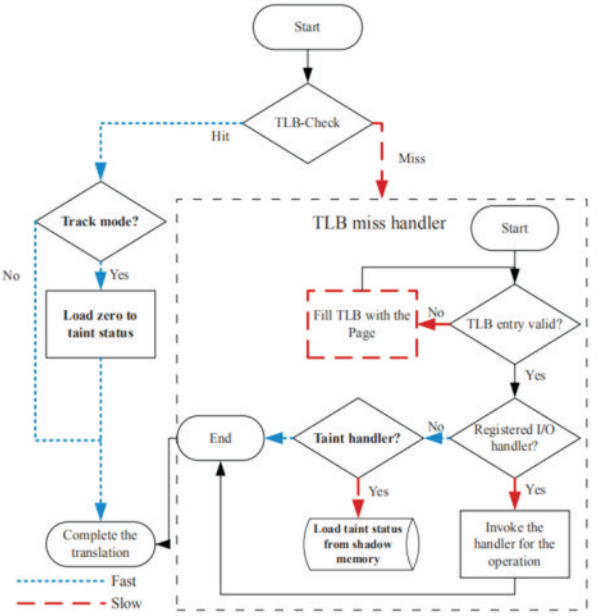


图4 Decaf++中底层QEMU的内存加载操作

对于内存存储操作，与加载相比有些许的不同，如图5所示。即要先判断存储操作的源操作数的污染状态——当源操作数受污染时，无论是否有shadow memory handler都要去更新shadow memory，并且要更新TLB控制位为新的shadow memory handler注册；当源操作数无污染的时候，与加载操作的处理相似。

Real-time Distributed Co-Movement Pattern Detection on Streaming Trajectories

古 文 推荐

“Real-time Distributed Co-Movement Pattern Detection on Streaming Trajectories”是被计算机国际顶级会议VLDB 2019录用的一篇文章。随着具有定位功能的移动设备的广泛部署，越来越多的轨迹数据可以被收集。这些数据可以实现共同运动模式检测，这在诸如轨迹压缩和未来运动预测之类的应用中很重要。现有的共同运动模式检测研究通常考虑历史数据，因此提出了离线算法。但是，诸如未来运动预测之类的应用程序需要对流轨迹进行实时处理。因此，本文研究了流轨迹上的实时分布式共同运动模式检测。

现有的离线方法假定在所有数据均可用处理时开始，然而，在流处理中，无界数据是实时到达的，这使得模式检测具有挑战性。为此，本文提出了一个基于Apache Flink的框架，该框架旨在进行高效的分布式流数据处理。

为了支持在流轨迹上进行有效的共同运动模式检测，必须解决三个挑战。第一个挑战，如何实时处理大规模的流轨迹？为了解决这个问题，本文利用Flink进行分布式流处理。Flink利用流水线式数据传输来实现低延迟和高吞吐量。

接上页

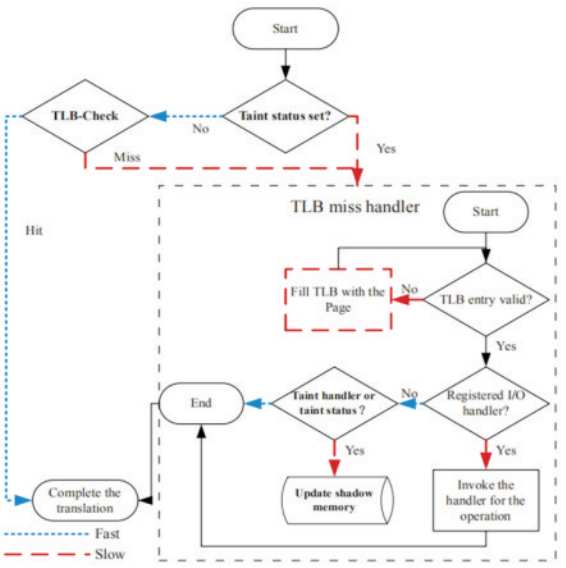
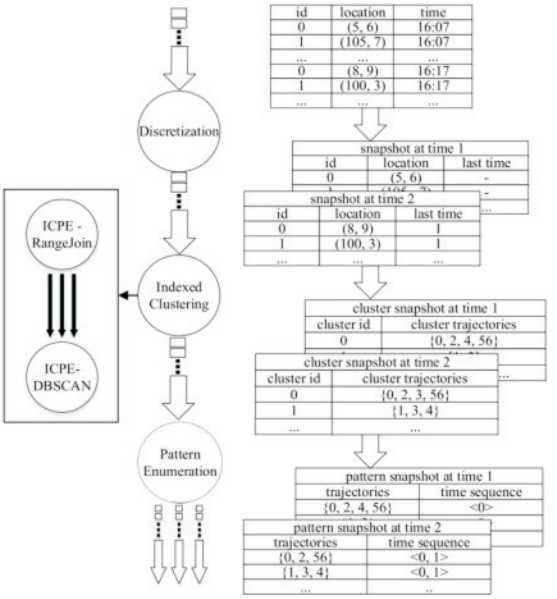


图4 Decaf++中底层QEMU的内存存储操作

最后，文章测试了Decaf++的性能，在这里不再赘述，感兴趣的读者可以去RAID2019论文集中查找该篇论文。同时建议阅读“Make It Work, Make It Right, Make It Fast:Building a Platform-Neutral Whole-System Dynamic Binary Analysis Platform”了解其他Decaf的相关工作，以便明晰Decaf++的改进与优化。个人实测过Decaf++，相较于Decaf确实有一定的性能提升。当然还有部分工作可以深入探讨，比如如何降低模式转换时产生的开销（虽然很小），以及在弹性污点传播的跟踪模式下，check shadow memory的频次如何调到一个最合适的大小，来保证既不会因频次过多而浪费时间也不会因频次过少而产生过长时间的无污染检查。

赵浩钧
2019级博士研究生
研究方向：云环境下隐私取证
Email：d201980936@hust.edu.cn

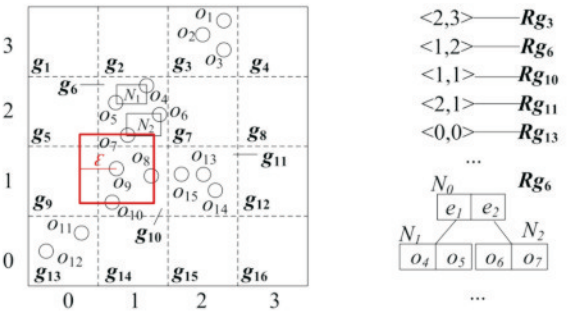
第二个挑战，如何有效地对流轨迹进行聚类？为了解决这个问题，本文采用了两层GR-index，它使用网格索引作为全局索引，并使用R树作为网格单元的局部索引。基于GR-index，在初始聚类步骤中使用了范围连接，通过在构建GR-index时执行范围查询，而不是在索引构造之后执行查询，可以加快连接处理。挑战三：如何减少模式枚举的指数级成本？为了解决这个问题，本文提出了一种简单而有效的基于id的分区方法。



图一：实时轨迹流模式检测框架ICPE

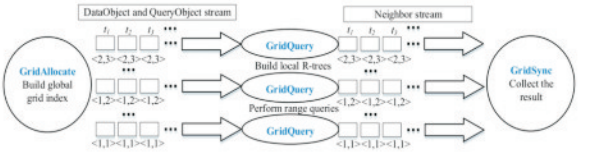
本文提出一种实时轨迹流模式检测框架ICPE (Indexed Clustering and Pattern Enumeration)，首先，它使用窗口操作将流式轨迹转换为快照，每个时间片生成该时间片的轨迹点分布快照，其次，ICPE基于Range Join和DBSCAN进行基于索引的聚类操作，当新快照到达时，ICPE将检测一起移动的轨迹，以获得聚类快照。最后，当每个群集快照到来时，ICPE利用模式枚举获取所有共同移动模式。

本文设计了三种算法来计算基于GR-tree的范围查询，即GridAllocate、GridQuery和GridSync。GridAllocate构建全局网格索引（即，计算每个位置的键），以将每个快照划分为不相交的网格单元，并将其转换为数据对象（特定网格单元中包含的位置）和查询对象（其范围区域与此网格单元相交）。对于每个网格单元，GridQuery方法将为数据对象构建本地R树，并为每个查询对象执行范围查询。最后，GridSync收集所有网格单元的结果。在此，为每个快照建立一个GR-index，并在查询后将其删除，因此不需要维护索引。



图二：GR-index

本文评估了该框架的效率和可扩展性，并与替代方法进行比较。所有实验都是在由11个节点组成的群集上进行的，其中一个节点充当主节点，其余节点充当从节点。

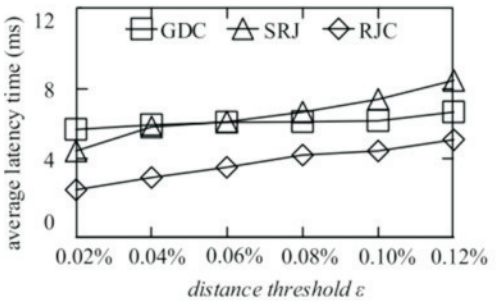


图三：ICPE范围查询

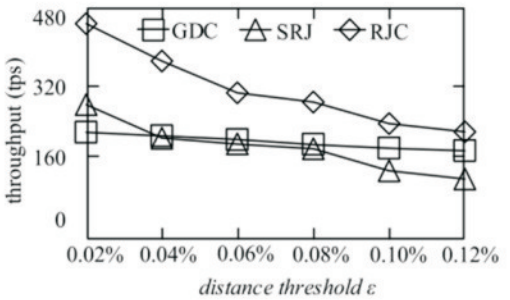
由于这是首次对流轨迹上的实时分布式共同运动模式检测的研究，因此没有基准方法。所以，本文根据历史轨迹调整了最新的分布式共同运动模式检测方法，用作基准方法。

实验结果如图四，RJC比SRJ具有更好的延迟和吞吐量，此外，RJC的性能优于GDC。这是因为GDC使用小值来划分数据空间，从而导致分区过多。最后，由于逐渐累积导致更大的

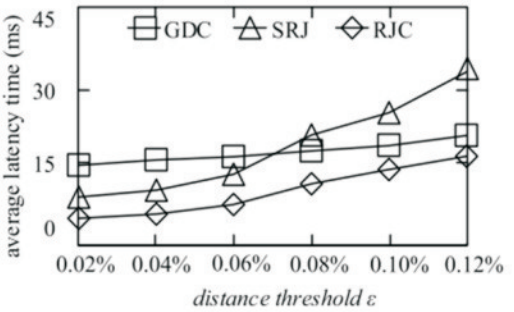
搜索空间，延迟不断增加，并且吞吐量减少。



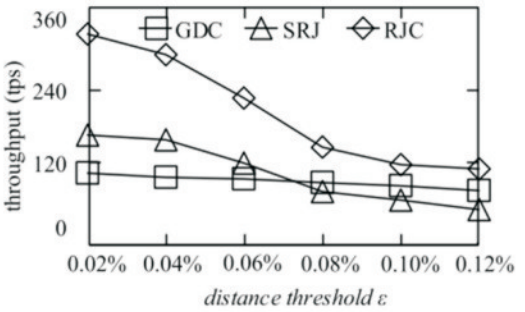
(a) Geolife



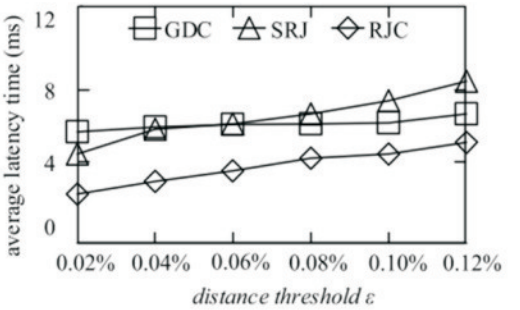
(b) Geolife



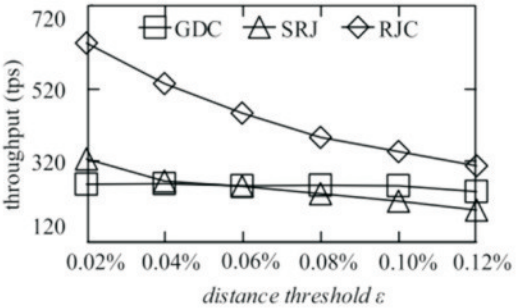
(c) Taxi



(d) Taxi



(e) Brinkhoff



(f) Brinkhoff

图四：算法性能对比

总结：本文研究了实时分布式共同运动流轨迹上的模式检测，这在将来的运动预测，轨迹压缩和各种基于位置的服务中很有用。本文设计了一个基于Flink的框架，称为ICPE。该框架包括两个阶段，即聚类和模式枚举。使用真实和合成数据集进行的大量实验表明，所提出的框架及其组成的数据结构和算法是有效且可扩展的。可以看出，ICPE可以实现低延迟和高吞吐量，因此，它可以支持对流轨迹进行实时共同运动模式检测。

古　文
2018级博士研究生
研究方向：大数据，分布式流处理
Email：guwenly@hust.edu.cn

大胆思考，严谨求证

彭　轩

不知不觉中就在华科度过了七年时光，今年已是研究生生涯的第四年。在实验室对石老师印象最深的一句话是“要从系统的角度去思考问题”，但是刚开始的自己确实不知道怎样思考才是从系统的角度去看问题，只能是去阅读系统顶会的论文找感觉。而石老师一直对我们秉承着高要求，敦促着我们要做高水平的系统研究，那时候OSDI/SOSP大概就是我心中不可触碰的两座圣杯。

转博之后通过石老师有幸去了MSRA系统组杨凡研究员下面做实习生，这段期间主要是参与肖文聪学长主导的深度学习中GPU 调度的工作，大概持续了8个月，最后论文成功发表在了OSDI'18上，这次系统顶会的工作经验也给自己后来的研究提供了许多宝贵的经验。

这份工作完成后偶然看到一篇关于深度学习集群中job failure的survey，其中一个问题是算法研究人员的模型在GPU上训练时经常会遇到显存不够的问题，于是便调研了下这个方向的一些研究工作，觉得目前的工作都做得有所欠缺，并且现有的深度学习系统对这一块的支持十分有限，和石老师以及mentor讨论后写了一份proposal，并抄送给了相关的研究员和系统组大佬周礼栋副院长，他们也都对这个问题挺感兴趣并提出了一些想法与意见。

回到实验室之后和组里的同学分享了自己的这个研究课题，石老师说这是一个很有趣的研究课题，十分鼓励我并集结了一群优秀的学生来一起往下做。这期间对自己能力的考验不仅仅来自于解决具体问题，同时也是自己第一次带领一群小伙伴来做研究，需要给每个人分配合适的任务并管理好大家的进度。其中最困难的一段时间应该是，在讨论重计算这个技术应该以一个怎样的设计放进深度学习的GPU内存管理系统才是合适的，最开始的意见是从编译的角度来思考这个问题，但是编译这一块自己的基础十分薄弱，在和编译领域大牛慎溪鹏老师简单讨论后仍然没有思路。于是自己从编译这个圈子跳出来，重新审视重计算到底需要考虑系统的哪些方面才能做好，最后觉得用Spark RDD的容错技术在运行时来做重计算更加合适。在经历了一年多的努力之后，论文被ASPLOS'20录用，在此也非常感谢三位小伙伴的辛苦工作和老师们的悉心指导。

写在最后算是自己对研究的一些感触吧。首先研究是灵活的，因为问题是自己来定义的，所以当我们觉得研究走不下去的时候不妨跳出自己画的那个“圈子”，重新来思考问题本身；另一方面，对于系统研究需要有严密的逻辑，因为系统研究不像算法类的有着公式推导，它全靠人的智慧与经验来思考系统设计的严谨性与完整性，这一点使得严密的逻辑思维在系统研究中显得格外重要。

彭　轩
2017级博士研究生
研究方向：分布式深度学习系统平台
Email：piecesix@hust.edu.cn