边缘服务器系统架构的研究与实践

2019-07-25 15:00

边缘服务器系统架构的研究与实践

吴 松
华中科技大学计算机学院
http://grid.hust.edu.cn/wusong

边缘计算

概念

边缘计算是在靠近数据源的边缘进行数据处理的新型计算范式

措施

将大部分的计算沉降在靠近设备的边缘，小部分连接远处的云端

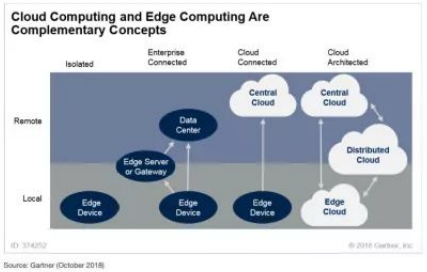
四大因素推动边缘计算发展


边缘数据处理
边缘系统架构


边缘数据处理
边缘系统架构

边缘计算发展态势

- **智能化助力**
  诸如物联网视频处理、智慧城市、智能家居、无人驾驶、VR/AR等智能化应用需求推动了物联网“**边缘+AI**”的趋势
- **大数据推动**
  据思科表示，到2020年，预计将有超过500亿台设备接入互联网，产生超过400ZB的数据；而这些数据将有45%需要被处理在边缘

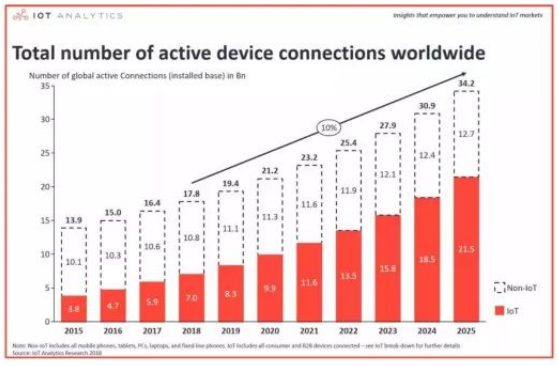


- **Gartner公布2019十大技术趋势，边缘计算位列其中，被视为未来科技发展重要趋势**
- **Gartner预测：2017年10%的企业数据产生在边缘，到2022年这一比例将超过50%**
- **2016年华为、沈阳自动化所、信通院、英特尔、ARM、软通动力等6家机构联合发起的边缘计算产业联盟（ECC）在北京正式成立，目前成员数量突破200+家**

边缘数据处理的需求与挑战

- **数据量大、数据速率快**

到2020年，预计将有超过500亿台设备接入互联网，产生超过400ZB的数据；而这些数据将有70%需要在边缘被处理【1】



【1】D. McAuley, R. Mortier, and J. Goulding. The dataware manifesto. In 2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011), pages 1–6, 2011.

边缘数据处理的需求与挑战

- **实时性要求高**

例如为可穿戴式摄像头提供的视频服务，响应时间需要在25ms至50ms之间；在某些工业系统检测、控制、执行的部分场景，实时性要求在10ms以内【1】

【1】S. Agarwal, M. Philipose, and P. Bahl, "Vision: The Case for Cellular Small Cells for Cloudlets," in Proceedings of the International Workshop on Mobile Cloud Computing & Services, 2014, pp. 1–5

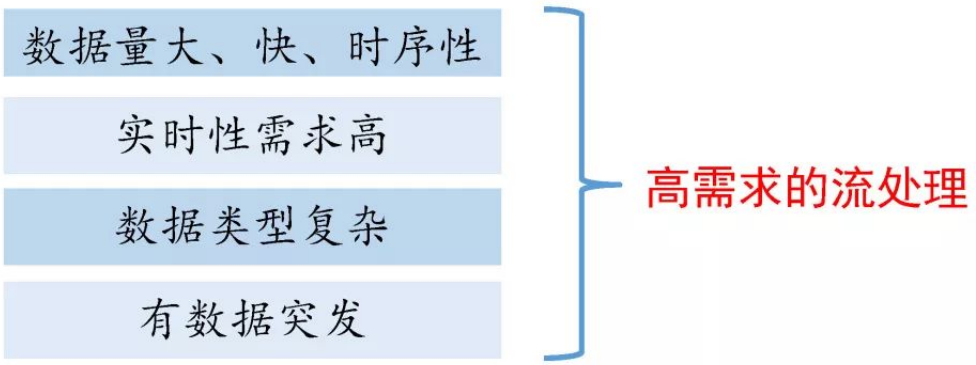
边缘数据处理的需求与挑战

- **边缘应用多样化**

边缘平台下应用的多样性（如视频处理、实时推荐、环境感知等），同时带来了对数据处理架构的通用性需求

- ✓自动驾驶
- ✓无人机
- ✓智能家居
- ✓VR/AR
- ✓智慧城市
- ✓……

边缘数据处理的本质



目前边缘流处理架构

- **基于边缘**：利用边缘设备或者小型边缘集群来进行数据处理
  - ✓ **边缘设备端**：增强边缘终端设备的流处理能力，如 F-Mstorm[1]
  - ✓ **边缘服务器**：利用边缘的小型服务器或者集群协作，如 Seagull[2]
- **云边结合**：将边缘设备或集群与云数据中心的资源相结合，提供统一的运行环境，协同进行流数据的处理。
  - ✓ **减少网络传输延迟**：对地理分布的云边协同优化，如 SpanEdge[3]
  - ✓ **加快数据处理与共享**：主要关注更好地实现操作符到物理节点的映射，如 Firework[4]

【1】Mengyuan Chao, Chen Yang, Yukun Zeng, Radu Stoleru:F-MStorm: Feedback-Based Online Distributed Mobile Stream Processing. SEC 2018: 273-285
【2】Roshan Bharath Das, Gabriele Di Bernardo, Henri E. Bal:Large Scale Stream Analytics Using a Resource-Constrained Edge. EDGE 2018: 135-139
【3】Hooman Peiro Sajjad, Ken Danniswara, Ahmad Al-Shishtawy, Vladimir Vlassov:SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers. SEC 2016: 168-178
【4】Quan Zhang, Qingyang Zhang, Weisong Shi, Hong Zhong:Firework: Data Processing and Sharing for Hybrid Cloud-Edge Analytics. IEEE Trans. Parallel Distrib. Syst. 29(9): 2004-2017 (2018)

典型工作

处理模式	代表论文	结论
基于边缘	F-MStorm: Feedback-based Online Distributed Mobile Stream Processing（**SEC'18**）	利用本地移动设备处理实时流数据，支持动态配置和调度
	Large Scale Stream Analytics using a Resource-constrained Edge（**EDGE'18**）	根据节点与传感器数据源的接近程度以及节点能够处理的处理量将任务分配给节点
云边结合	Firework: Data Processing and Sharing for Hybrid Cloud-Edge Analytics（**TPDS'18**）	通过虚拟共享视图和服务组合为IoE应用程序提供云边协同下的分布式数据处理和共享
	SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers（**SEC'16**）	通过跨中心和近边缘数据中心分发流处理应用程序，减少WAN连接带来的延迟

存在的问题

- **仅依赖于边缘的流处理**
  - 性能会受到边缘环境下有限资源的制约
- **云边协同的流数据处理**
  - 不可避免地引入了数据传输的网络延迟

能否将硬件加速器引入边缘环境下的通用流处理架构？

解决方案——FPGA VS GPU

FPGA是加速边缘流数据处理的理想选择

- ◆丰富的并行性
- ◆极低的功耗
- ◆在数据中心广泛应用

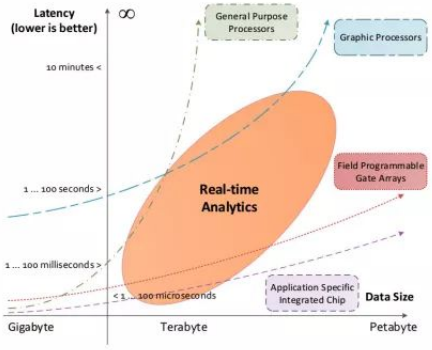
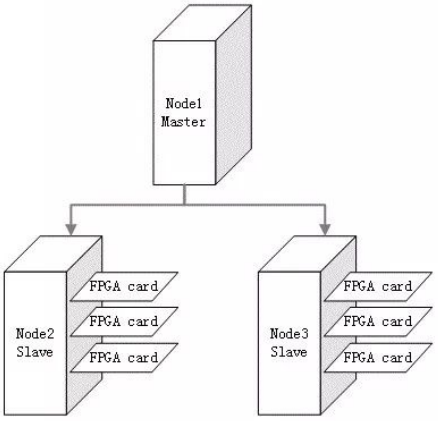


Fig. 1: Envisioned acceleration technology outlook.

【1】右图来源：Hardware Acceleration Landscape for Distributed Real-time Analytics: Virtues and Limitations. In ICDCS'17

边缘集群引入FPGA加速器的挑战

- **挑战一：**如何轻量级集成和管理边缘集群中的FPGA资源？
- **挑战二：**如何进行任务调度满足边缘集群资源受限场景？
- **挑战三：**如何减少FPGA-JVM之间数据传输开销，降低处理延迟？
- **挑战四：**如何让用户开发多样化FPGA加速流应用程序变得简单？



挑战一——如何轻量化集成管理

设计并实现了轻量级的FPGA集成

问题背景

- 云环境下数据处理系统的FPGA集成依赖于资源调度框架（例如Yarn）
- 边缘环境下资源受限，云环境下的FPGA集成方式造成过重的系统部署和资源使用负担

解决思路

- 边缘节点的轻量级FPGA管理器，为流处理应用提供FPGA加速器服务（FPGA-as-a-Service@Edge）
- 可插拔组件，使得FPGA资源的启用/停用变得更加灵活
- 仅需FPGA基础环境（例如Altera SDK for OpenCL），部署和运行不需要外部框架/库支持

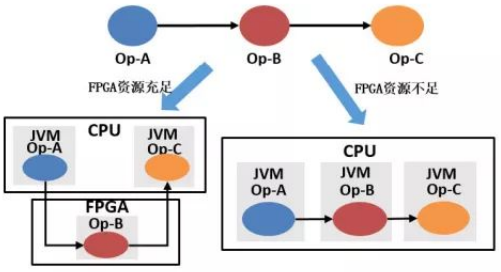
Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, etc., "When FPGA-accelerator meets Stream Data Processing in the Edge". ICDCS 2019, Dallas, Texas, USA

挑战二——资源受限环境下的任务调度

设计并实现自适应的CPU-FPGA调度策略

问题背景

- 边缘服务器的资源受限，可部署的FPGA资源也受到限制
- 在FPGA资源不足的场景下，应用程序的执行可能失效



解决思路

- 自适应的加速器任务调度策略，实现FPGA任务到CPU的自动反向卸载
- 缺省使用加速器优先调度算法，在FPGA资源不足时自动将加速器任务进行转化并调度到CPU上去执行

Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, etc., "When FPGA-accelerator meets Stream Data Processing in the Edge". ICDCS 2019, Dallas, Texas, USA

挑战三——降低JVM-FPGA数据传输延迟

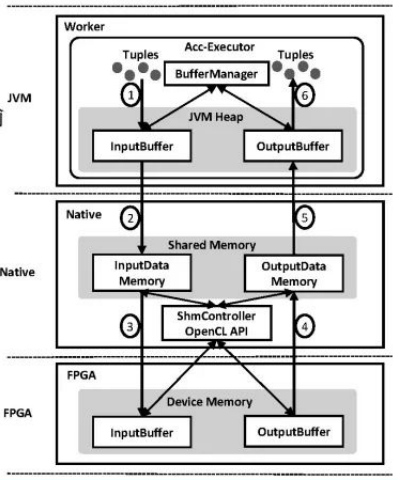
设计针对DSP系统的JVM-FPGA数据传输机制

问题背景

- 边缘环境下应用的数据处理实时性要求极高
- 数据处理的DSP系统普遍采用Java实现
- FPGA的引入不可避免地引入了JVM-FPGA数据传输延迟

解决思路

- 数据批量传输：缓存一定批量的数据同时传输给FPGA，提升数据传输带宽和吞吐量，减少频繁多次传输产生的数据移动开销
- 数据传输流水线：将数据传输的多个步骤设计成多级流水线，减少数据拷贝和移动时间



Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, etc., "When FPGA-accelerator meets Stream Data Processing in the Edge". ICDCS 2019, Dallas, Texas, USA

挑战四——提高易用性

设计并实现了用户友好的编程接口

问题背景

- FPGA的引入使得流处理应用的开发复杂化
- 用户需要处理与FPGA相关的复杂编程细节，降低了流系统的编程易用性

解决思路

- **设计了简单易用的应用拓扑编程接口**
  - 在原有系统（Storm）的编程接口基础上，增加了新的操作符类型抽象，用户可通过实现这种新操作符方便使用FPGA加速

Listing 1. IAccBolt.java

```
public interface IAccBolt extends
    Serializable{
       void accPrepare(Map stormConf,
            TopologyContext context,
            OutputCollector collector);
       void accExecute(Tuple input);
       void accCleanup();
}
```

Listing 2. Vector addition kernel

```
__kernel void vector_mult(
       __global const float *x,
       __global const float *y,
       __global float *restrict z)
{
       int index = get_global_id(0);
       z[index] = x[index] + y[index];
}
```

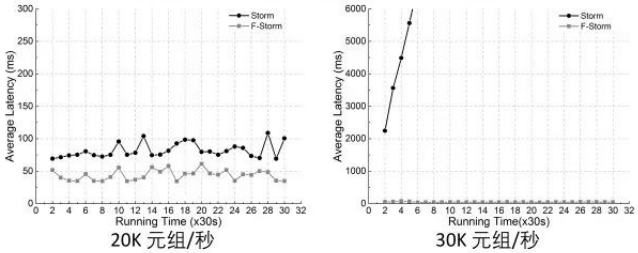
- **简化了FPGA加速器任务编程，降低了开发门槛，提升了开发效率**
  - 隐藏了与FPGA相关的加速器任务初始化、启动和停止、数据传输细节
  - 用户只需要致力于实现能在FPGA上高效执行的kernel

Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, etc., "When FPGA-accelerator meets Stream Data Processing in the Edge". ICDCS 2019, Dallas, Texas, USA

F-Storm系统性能评估

延迟

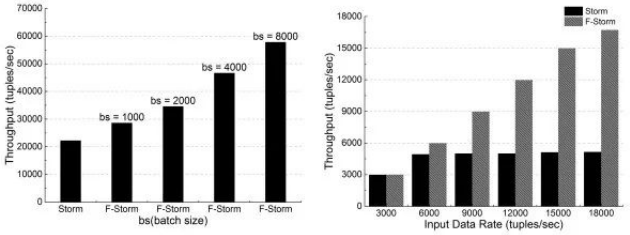
Grep应用在不同数据输入速率下的数据处理延迟



20K 元组/秒　　30K 元组/秒

- 与Storm相比，F-Storm显著减小了数据处理延迟，并且在面临高速数据流时仍能保持较低的延时，可更好地应对数据突发

吞吐量



Grep应用的吞吐量（不同的批量大小下）　　Grep应用的吞吐量（不同的数据输入速率下）

- 与Storm对比，F-Storm获得了显著的吞吐量提升（2.1x-3.4x），处理的数据速率越高时，F-Storm的吞吐量优势更加明显

启示

- 通过F-Storm的设计、实现和评估证明了**在边缘服务器中使用FPGA加速流数据应用**的重要性和可行性
- **未来的研究方向**
  - **提供通用的FPGA加速器服务**：如何将FPGA加速器的使用与特定DSP系统分离，以一套FPGA加速器服务支持不同的DSP系统
  - **优化流应用操作符的部署**：在边缘计算环境中，计算资源例如边缘服务器或者FPGA可能是地理上分布的，如何根据计算资源的类型和位置优化流应用操作符的布局，以实现较低的延迟
  - **优化FPGA-FPGA数据传输**：如果不同FPGA板上的某些任务需要交换数据，可以直接通过FPGA-FPGA路径传输数据，而不是FPGA-CPU-FPGA路径，这样可以节省大量的数据传输时间

Song Wu, Die Hu, Shadi Ibrahim, Hai Jin, etc., “When FPGA-accelerator meets Stream Data Processing in the Edge”. ICDCS 2019, Dallas, Texas, USA

研究点

研究点

边缘资源碎片化问题

终端资源碎片化

- 底层硬件多样化
- 资源能力参差不齐
- 运行在其上的系统软件呈现专用的特点

问题

- 端和端、云和端之间业务逻辑的部署和管理十分复杂

【1】https://www.huawei.com/minisite/liteos/cn/

边缘设备发展新趋势

平台化趋势

- 终端设备资源能力越来越强，呈现出平台化趋势，例如树莓派、英伟达jetson系列以及物联网多传感器套件

在新趋势下，如何探索便于边缘设备管理的新模式？

虚拟化为边缘赋能

边缘虚拟化的需求和挑战

◆**更快的引导速度**

实时计算服务（像VR/AR、无人驾驶）场景下需要启动时间控制在ms级【1】

◆**更少的资源开销**

边缘下的资源受限，对运行环境占用内存需求资源在MB级

◆**更高效的弹性部署**

在边缘节点之间提供弹性拓展来适应边缘应用场景的高峰和低谷

◆**兼容云生态**

边缘下很多场景是在利用云边协同能力，兼容云生态也至关重要【2】




【1】S. Agarwal, M. Philipose, and P. Bahl, "Vision: The Case for Cellular Small Cells for Cloudlets," in Proceedings of the International Workshop on Mobile Cloud Computing & Services, 2014, pp. 1–5.
【2】Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation. In IEEE Access, 2017

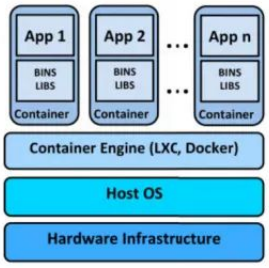
边缘虚拟化

研究点		代表成果	特点
容器	轻量化趋势	Docker Container Deployment in Fog Computing Infrastructures. **(EDGE'18)**	**通过消除容器在边缘部署期间产生的不必要延迟，将容器部署时间大大减少**
		Extend Cloud to Edge with KubeEdge.**(SEC'18)**	**对Kubernetes做轻量化研究加快容器在边缘场景的编排**
		Container-Based Cloud Platform for Mobile Computation Offloading **(IPDPS' 17)**	**使用容器将移动端的工作负载Offloading到云上**
Unikernel	向边缘场景应用	RaDiCS: Distributed Computing Service over Raspberry Pis with Unikernels. **(MobiSys'18)**	**提出了基于树莓派集群环境的分布式数据处理框架**
		FADES: Fine-Grained Edge Offloading with Unikernels **(HotConNet@SIGCOMM '17)**	**通过Unikernel实现了对边缘任务的细粒度卸载，加强了云边协作**
		Jitsu: Just-In-Time Summoning of Unikernels （**NSDI'15**)	**将Unikernel移植到新的Xen/ARMv7架构，构建了在资源受限环境下的云平台**

边缘虚拟化的核心——轻量级虚拟化

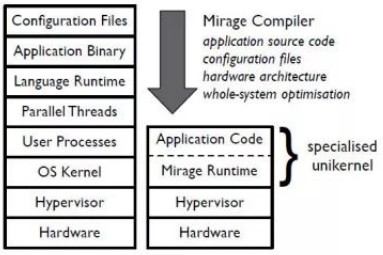
■ **容器特点**

- ✓ 引导速度较快
- ✓ 资源利用率高
- ✓ 部署灵活
- ✓ 生态环境好
- ✓ **依然太“重”**



■ **Unikernel特点**

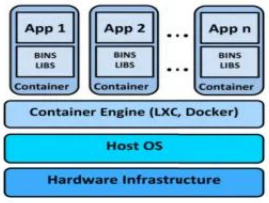
- ✓ 引导速度快
- ✓ 资源利用率很高
- ✓ 安全性更好
- ✓ **不够灵活**



边缘虚拟化的核心——轻量级虚拟化

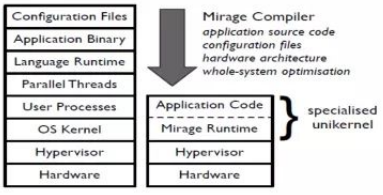
■ **容器特点**

- ✓ 引导速度较快
- ✓ 资源利用率高
- ✓ 部署灵活
- ✓ 生态环境好
- ✓ **依然太“重”**



■ **Unikernel特点**

- ✓ 引导速度快
- ✓ 资源利用率很高
- ✓ 安全性更好
- ✓ **不够灵活**



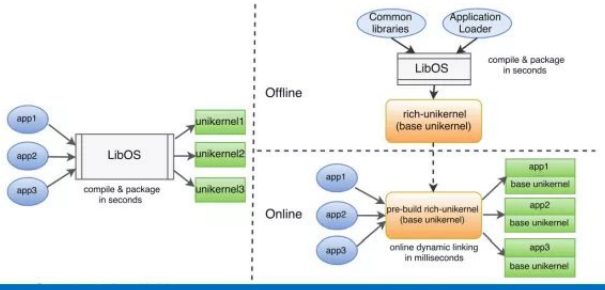
边缘Unikernel的技术挑战

◆**挑战一**：Unikernel一旦构建之后是不可变的，重新构建新的Unikernel需要花费数秒编译时间，难以满足边缘应用的多样性、灵活性、实时性需求

◆**挑战二**：现有的Unikernel普遍基于Linux内核代码构建，不能提供Android等移动端操作系统支持，难以满足Mobile offloading场景需求

挑战一：如何使unikernel更具灵活性

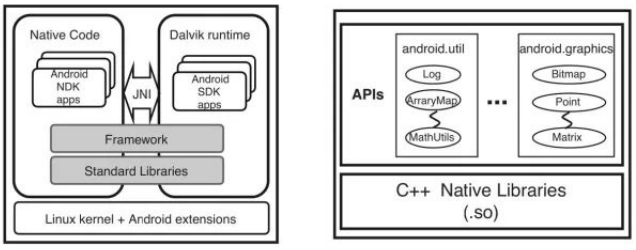
- **提出和设计了Rich-unikernel模型**
  - 为某一类应用定制一个公共的、通用的base unikernel，包含某一系列应用所需代码库的最小集
  - 将应用代码动态加载到base unikernel中直接运行，在不需要重新构建unikernel情况下实现应用代码的快速unikernelize



Song Wu, Chao Mei, Hai Jin, Duoqiang Wang, "Android Unikernel: Gearing mobile code offloading towards edge computing". Future Generation Comp. Syst. 86: 694-703 (2018)
开源系统：https://github.com/cgcl-codes/Libdroid

挑战二：如何支持移动端代码

- 设计和实现了Android Unikernel系统
  - 重构传统安卓系统为独立的Libdroid模块
  - 保留安卓特性相关库和API，如Bitmap，Logger
  - 去除安卓系统中硬件相关驱动，如USB，蓝牙，WIFI



(a) Android system (b) Refactored Android libraries

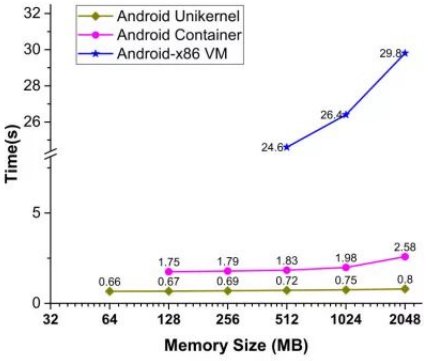
Song Wu, Chao Mei, Hai Jin, Duoqiang Wang, "Android Unikernel: Gearing mobile code offloading towards edge computing". Future Generation Comp. Syst. 86: 694-703 (2018)
开源系统：https://github.com/cgcl-codes/Libdroid

评测：系统开销

类型	内存占用	镜像体积	启动能量消耗
Andorid-x86 VM	516MB	1.4GB	334.2J
Cloud Android Container	128MB	1.02GB	49.4J
Android Unikernel	52MB	70MB	15.6J

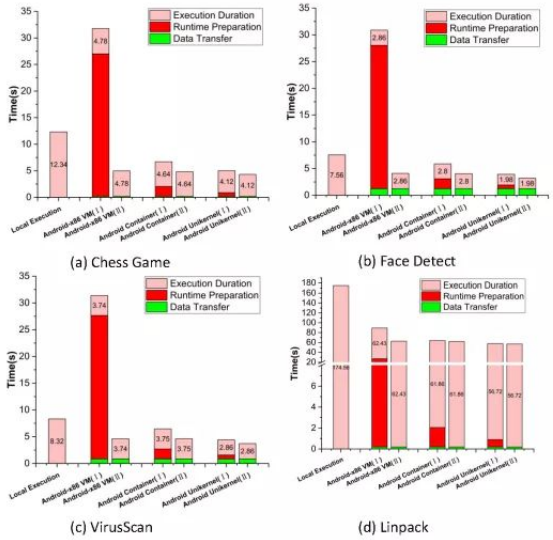
- Android Unikernel相比于传统Android虚拟机和Android容器
  - 内存占用分别减少10x，2.5x
  - 镜像体积分别减少20x，15.6x
  - 启动能量消耗分别减少21x，3.2x

启动时间



- Android Unikernel相比于Container启动时间减少2.6x，相比于虚拟机减少34x-37x

评测：实际应用性能



- 在冷启动和非冷启动下，Android Unikernel返回结果时间比本地运行平均加速60%
- 对于ChessGame和FaceDetect等实时性强的应用，VM比本地运行表现更差，因为其启动延迟长
- 在运行时间上Android Unikernel表现更好，因为没有上下文切换和系统调用，代码运行效率更好

【1】Song Wu, Chao Mei, Hai Jin, Duoqiang Wang:Android Unikernel: Gearing mobile code offloading towards edge computing. Future Generation Comp. Syst. 86: 694-703 (2018)

总结

边缘计算的兴起是计算模式的又一次变革，同时也给传统云环境下的系统架构带来了新的挑战，我们针对边缘场景下的数据处理架构和边缘虚拟化方法进行了深入分析与研究

●边缘数据处理

➢F-Storm：针对传统边缘流数据处理模式的问题，提出并实现了基于FPGA加速的边缘流处理方案

●边缘系统软件

➢Android Unikernel：针对Unikernel在边缘环境下的灵活性和兼容性问题，提出了新的Unikernel模型与系统实现

容器隔离机制-HPDC'19, 容器镜像管理-MSST'19, 容器I/O优化-IPDPS'19
http://grid.hust.edu.cn/wusong