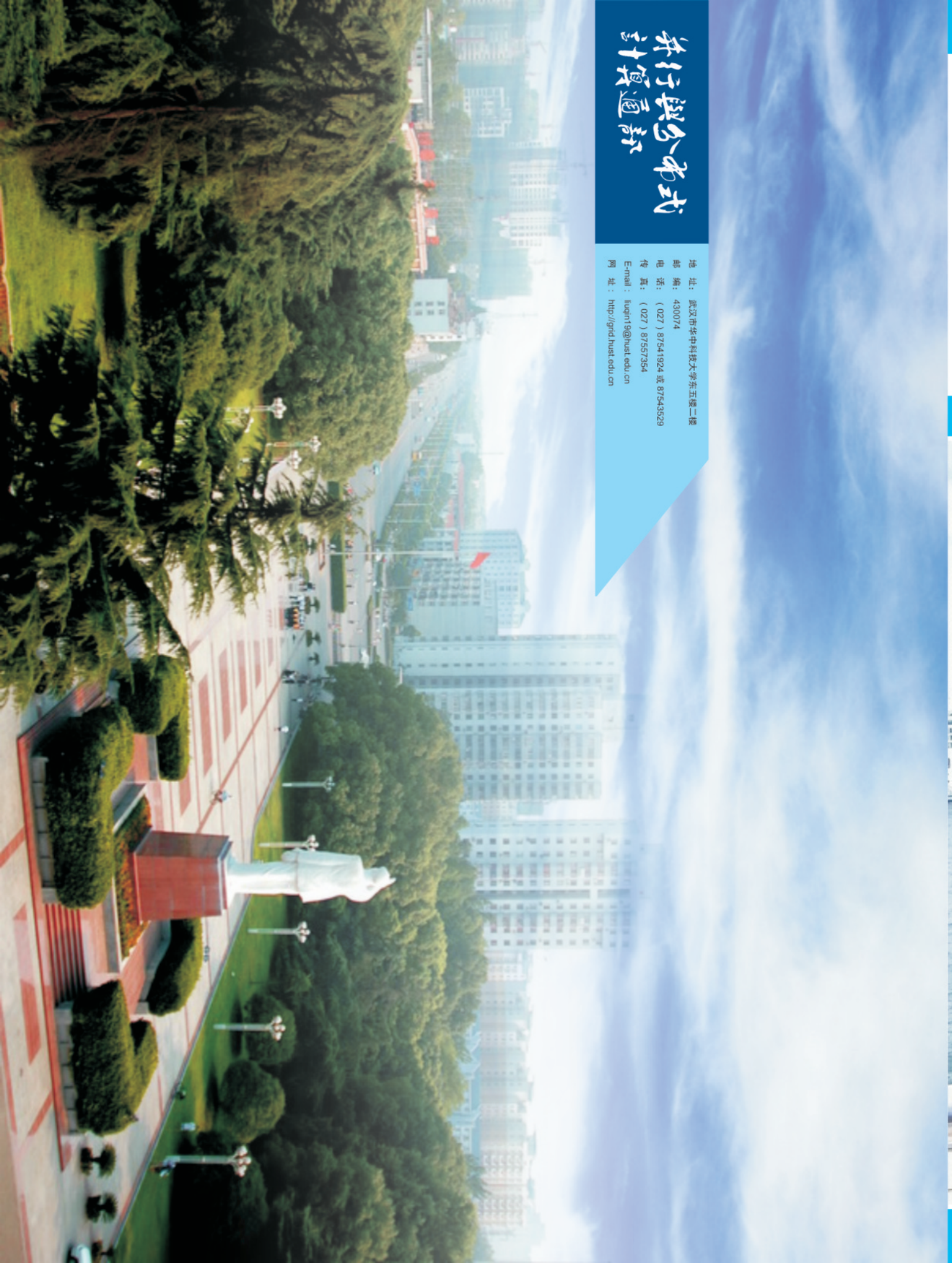


并行與分布式計算通訊

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
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訊

BING XING YU FEN BU SHI JI SUAN TONG XUN

2019年第1期　总第36期　2019年03月

封面人物：刘长鸣——关于解决科研问题的思考和分享

2019年度实验室各研究方向规划

Flink流计算引擎

追逐皇冠上的明珠

http://grid.hust.edu.cn

# 2019 年实验室新春联欢会

# 新平台、新起点、新目标
## ——2019年度实验室科研方向年度规划

2018年是不平凡的一年，实验室硕果累累，已成立大数据技术与系统国家工程研究中心。站在这样一个新平台上，实验室各研究方向在2019年将在以往研究基础上开启新的征程。实验室老师们齐心协力，通过广泛讨论分析，提出了各自研究领域拟在2019年攻克的主要挑战以及符合社会需求和技术发展趋势的研究内容。本期季刊将从云计算与移动计算、系统软件和体系结构、网络空间安全和大数据这四个实验室研究方向出发，向读者展示实验室在新一年的具体科研规划，希望实验室大家庭的同学们可以从中了解前沿和找到自己的定位及目标，更好地服务实验室及国家发展。

在云计算与移动计算方向，2019年将重点围绕边缘计算、区块链和数据中心这三个方面开展研究工作。在边缘计算方面，将着重研究面向边缘计算的超轻量级虚拟化关键技术、面向边缘计算的动态负载调度技术与系统和基于边缘计算的健康监测和疾病风险评估；在区块链方面，将着重研究适用于同构/异构区块链跨链交互的双引擎平台和基于区块链的医疗健康数据可追溯安全共享技术用于解决医患和医疗机构之间的纠纷，进一步维护医患的共同利益；在数据中心方面，将着重研究高效能云计算数据中心节能新技术和数据中心智能化网络服务架构，争取在扩大研究工作影响的同时将科研成果应用到重要云计算企业和实际的生态环境中。

在系统软件和体系结构方向，2019年将重点解决以大数据应用为代表的高时效处理需求给异构计算环境和异构内存管理带来的挑战。本研究组将在面向复杂大数据处理需求的数据流体系结构设计方面，开展硬件层面的灵活定制、执行层面的高效调度和通信层面的动态划分三个关键问题的研究，提出数据流体系结构的设计规范和设计方法。在面向大数据高时效处理操作系统方面，设计并实现面向异构内存的系统软件栈，基于RDMA的异构内存池化机制，开展云环境下特定需求下的虚拟机定制优化，基于效益的异构内存动态分配机制等研究工作。在面向图计算的异构计算平台及系统方面，考虑图计算异构并行低、高效表达难、动态划分复杂等多个方面的难题，研究异构体系结构下的图计算软硬件技术体系。

在网络空间安全方向，2019年将着重在云计算安全、区块链和软件漏洞检测方面开展研究。具体来说，面向容器化、SDN化的云环境中的传统安全问题以及新型安全问题，研究基于容器技术和SDN技术的灵活定制、高效执行的安全防御机制，并融合容器技术与SDN技术通过软件定义的方式定制安全防御机制。在面向物联网–边缘计算的大数据机密性保护方面，开展具有实用安全的加密搜索机制、灵活易扩展的混合可搜索加密机制以及高效通用的数据安全交换机制的相关研究。在区块链支撑技术方面，开展数字资产安全、基于零知识证明的区块链资产恢复、智能合约安全漏洞防护、智能合约去中心化审计这四个关键问题的研究。在软件漏洞检测方面，开展基于知识图谱的智能化源代码漏洞检测技术研究，研究多源漏洞数据的采集与分析、漏洞知识图谱的构建、智能化源代码漏洞检测这三个方面。

在大数据方向，2019年将重点从大数据基础理论、大数据处理支撑技术和大数据管理与分析技术这三个方面开展大数据相关研究。在大数据基础理论方面，将着重研究面向新型体系结构的低通信复杂度分布式算法；在大数据处理支撑技术将着重研究面向大数据中半结构化数据的加速解析、大规模图数据的表示与存储、异构图计算、大规模深度学习系统、高时效大数据流连接处理平台和HPC环境下的I/O调度优化；在大数据管理与分析技术方面将着重研究基于多视角的网络表征学习、大规模知识管理与搜索和抗隐私分析的大数据查询处理。

衷心祝愿实验室在新的一年圆满完成各项科研任务，站在大数据技术与系统国家工程研究中心这样一个新的平台上开启新的起点，取得更大的成就，服务国家战略需求！

张　宇

二零一九年三月

主　编：金　海
本期执行主编：张　宇
编　委：（按姓氏拼音排序）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、吕新桥、马晓静、羌卫中、邵志远、石宣化、王多强、吴　松、肖　江、谢　夏、徐　鹏、余　辰、袁　斌、袁平鹏、章　勤、张　宇、赵　峰、郑　龙、郑　然、邹德清
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

…… 1

## 热点

…… 3

## 封面人物

关于解决科研问题的思考和分享 …… 刘长鸣　9

## 专栏

云计算与移动计算组研究规划
…… 吴　松，刘方明，余　辰，陆　枫，肖　江，顾　琳　11
支撑大数据处理的高效体系结构和系统软件——体系结构与系统软件组科研规划
…… 廖小飞，刘海坤，蒋文斌，邵志远，郑　然，郑　龙，张　宇　14
新型计算模式下的关键安全理论和技术——网络空间安全组科研规划
…… 邹德清，徐　鹏，代炜琦，羌卫中，马晓静，袁　斌　18
大数据组研究规划
…… 石宣化，陈汉华，赵　峰，袁平鹏，华强胜，谢　夏，丁晓锋，黄　宏　23

## 声音

Linux 内核 CFS 调度器 …… 杨晓礼　28
浅析边缘计算 …… 邹俊峰　30
GAN-生成式对抗网络 …… 任　毅　32
moby && cli 源码分析 …… 陈　旭　33
基于 golang 的网络爬虫框架 …… 吴朱天　35
Flink 流计算引擎 …… 邹俊峰　39

## 动态

实验室举行 2019 年新春联欢晚会 …… 刘　芹　41
2018 年度实验室十大新闻揭晓 …… 刘　芹　42

## 推荐

ACES: Automatic Compartments for Embedded System …… 李晨晨　推荐　43
Through-Wall Human Pose Estimation Using Radio Signals
…… 李辉楚吴　推荐　45
FoggyCache: Cross-Device Approximate Computation Reuse …… 武　静　推荐　47
Kpart: A Hybrid Cache Partitioning-Sharing Technique
for Commodity Multicores …… 陈　鹏　推荐　48
Fine-grained Concept Linking using Neural Networks in Healthcare
…… 陈　林　推荐　50
Processing-in-Memory for Energy-efficient Neural Network Training:
A Heterogeneous Approach …… 刘　聪　推荐　52
T-Fuzz: fuzzing by program transformation …… 石　建　推荐　54
Improving Accuracy of Android Malware Detection
with Lightweight Contextual Awareness …… 吴月明　推荐　55
A Distributed Framework for Emerging AI Applications …… 吴文超　推荐　58
Neural Architectures for Named Entity Recognition …… 桂向宇　推荐　60
Data Tiering in Heterogeneous Memory Systems …… 王孝远　推荐　62

## 交流

追逐皇冠上的明珠 …… 刘　博　64

## 量子计算介绍与近期进展

（陈良宇 整理）

量子计算有两个非常不一样的模型。一个是状态模型，一个是量子退火[1]模型。量子计算基于量子比特（qubits），以实现Shor算法[2]分解一个2048比特的大整数为例，理论证明通用量子计算需要至少4000个逻辑量子比特。第一种模型做的量子计算机，现在还都只有两位数量级的物理量子比特，而理论上已经证明如果要有4000个逻辑比特，按照状态模型造的量子计算机需要有2亿个物理量子比特，这在目前是难以实现的。第二种模型做的量子计算机现在已经实现，已有厂家（如D—Wave）在卖了，最新型的已经有2000个工作比特，已非常接近实用，但量子退火模型只能做一种运算：模拟退火，并且不能证明比电子计算机快（量子退火（QA）是用于通过使用量子波动的过程在给定的一组候选解（候选状态）上找到给定目标函数的全局最小值的元启发式。）。

所以量子计算乐观的预测是：在不久的将来，第一种模型的量子计算可以作为电子计算机的加速器出现，在实际运算中得到了应用。

除了制造的问题，量子计算现在的理论证明都没有把I/O算在里面。如果把I/O开销计入，量子计算机并不比电子计算机快。I/O的问题是量子计算最重要的问题，比量子计算的算法还重要。

### 如何在现有的量子计算机上实现量子霸权

“量子霸权”（quantum computational supremacy）是指量子计算拥有的超越所有经典计算机的计算能力。

量子计算机在多项式时间内可以解决的问题类称为有界错误量子多项式时间复杂性类（Bounded-error Quantum Polynomial time, BQP）。而在证明P不等于NP之前，几乎没办法直接证明BQP严格真包含P（或者BPP，有界错误概率多项式时间复杂性类，Bounded-error Probabilistic Polynomial time）。所以必须依赖一些复杂性的假设，比如之前计算机科学家设计的RSA加密就是依赖整数质因数分解十分困难的假设。但是Shor算法可以在多项式时间内解决整数质因数分解问题，所以整数质因数分解问题之前一直是量子计算超越经典计算所依赖的例子。但因为Shor算法（分解2048比特的整数）需要超过4000个逻辑量子比特去解决，所以近期大家没有继续试图使用Shor算法来展示量子霸权，而是试图用50~100个物理比特的机器产生一个非常大的概率分布，即量子波色采样，然后证明这个概率分布在经典计算机上因为算力不够而无法产生。这个方案有两个好处，第一是其依赖的复杂性假设非常强，假设的是多项式时间层次（Polynomial Hierarchy, PH）从第三层开始不会坍缩。PH是计算机科学家认为极为不可能互相等价（即塌缩）的一个复杂性类层次，所以这个假设比整数分解的困难性假设更强。第二是这个方案对错误的容忍度比较强，不需要可容错的量子机器来实现，适合现阶段的NISQ系统。这里需要说明的是，（量子采样）这个计算任务本身是没有任何实用价值的，唯一的目的就是为量子计算机量身定制一类问题，展示其计算能力。

最近埃文·唐（Ewin Tang）等人提出了新的经典算法来求解用户推荐问题[3]以及矩，求逆问题[4]。他们的算法指数量级地缩短了求解的时间从复杂性角度上达到了和某类量子机器学习算法一样的速度，证明了量子算法在这些问题上没有优势。但是还有很多的量子算法与已知的经典算法相比有指数时间的优势，比如著名的Shor算法。埃文·唐的算法需要基于对输入矩阵的随机查询访问，而这类算法已经被证明

（如果应用到整数质因数分解问题时）和Shor算法相比是有指数量级的差别的，所以新出现的算法在整数质因数分解等问题上是无望超过量子算法的。另外，新算法在解矩阵求逆问题上对于所求解的矩阵需要低秩等假设。新的经典算法虽然达到了同样的速度，但是它只能求解低秩的矩阵，而量子算法，例如HHL算法[5]，没有低秩这一要求。

### 量子计算机的结构

量子计算机基本结构称为量子图灵机，和NP图灵机非常相似，区别在于它可以有负的概率以及最后只能根据概率随机取出一个结果写入纸带。这个模型不太直观，所以不经常被提起。姚期智教授在1993年证明了量子图灵机和简单的量子线路模型等价[6]。现在基本可以认为量子计算机就是一个可逆逻辑电路加上一些特殊的量子门。和冯·诺伊曼结构相比，量子计算机是没有外部储存设备的，其CPU就是内存。可以认为量子计算机直接在内存里进行运算。可以把每个量子比特看作一个2个经典比特的内存空间。这和经典的内存的区别在于，多个内存组合不是简单地相加而是张量积。n个量子比特储存的是$2^n$个经典比特的信息。在量子比特上进行量子门操作（物理上就是打激光、微波等操作）就等价于直接在内存空间里进行一个$2^n \times 2^n$的矩阵运算。每个量子计算过程都可以看作在内存空间里的初始值上做矩阵和内存向量的乘法，然后以一定的概率读出最后内存空间的部分信息。


[1] 杜卫林, 李斌, 田宇. 量子退火算法研究进展[J]. 计算机研究与发展,2008,45(9):1501-1508.

[2] Shor P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer[J]. SIAM J.Sci.Statist.Comput ,1994, 26(5): 1484-1509.

[3] Tang E. A quantum inspired classical algorithm for recommendation systems[OL]. (2018). https://arxiv.org/abs/1807.04271.

[4] Gilyen A, Lloyd S,Tang E. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension[OL].(2018). https://arxiv.org/abs/1811.04909.

[5] Harrow A W, Hassidim A, Lloyd S. Quantum algorihtm for solving linear systems of equations[OL]. (2010). https://arxiv.org/abs/0811.3171

[6] Yao A. Quantum circuit complexity[C]. Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science, 1993: 352-361.


## 云计算驱动十大战略技术变革

（谢晓然 整理）

2018年年末，Gartner发布了2019年十大战略技术发展趋势，分别为自主设备、增强分析、AI驱动的开发、边缘计算、沉浸式体验、区块链、智能空间、隐私与道德、量子计算。虽然这些技术表面看似乎毫不相关，但细究根源，提供计算力的云，是其背后最基础的连接点。从产业发展来看，2019年，传统企业的数字化转型、互联网的ToB化、数据中心互联网化等产业浪潮，以及AI、IoT、区块链等技术变革均将继续向前发展，层出不穷的创新技术将渐次融入科技浪潮中，转化成无数灵活可用的新产品，云计算作为信息科技领域的水电煤，其公共属性将愈发明显，在这场科技变革中将发挥更为关键的作用。

如果说云计算的上半场是以消费互联网为主导，那么以企业信息化改造为核心的下半场，其横跨了从互联网到移动互联网，最后到万物互联网，构成一张物物相连的超大规模互联网络，云计算作为连接的关键支撑，随着IoT日渐成熟，可发挥的作用将会愈发重要。可以预见的是，未来IoT连接需求将无处不在，基于云的IoT将成为云厂商们的下一个角力点。据IDC的预测，到2020年40%的物联网数据将会在网络边缘自动处理，到2025年这个比例会超过

50%。随着IoT市场的全面爆发，大量实时的需要交互的计算将在边缘节点完成，边缘计算能够有效避免数据向云端传输时所面临的带宽限制、数据泄露风险和时延问题，成为下一阶段的关键点。

尽管公有云已是公认的未来，但大部分的传统企业都不会直接走到公有云，这当中既有对数据主权、监管合规等方面的要求，也有自身业务转型难以一步到位的特征，所以现阶段很多企业更倾向于私有化的形式上云，而具备公有云与私有云两者优势的专有云，成为当前阶段的最佳选择。专有云，可以把它想象成是企业内部私有的“公有云”，将成熟的公有云平台缩小或内化了，其本质内核是经过规模化运营实践验证的公有云。

Gartner预测，到2021年，中国市场公有云和专有云的收入占比为4:6。这意味着在未来几年中，专有云将快速进入舞台中央，成为云计算下半场的主导力量。在可预见的未来，互联网企业上云结束，各大云厂商对于传统政企市场的争夺空前激烈，专有云之争将全面爆发，云计算也将在曲折中持续接近终极形态。

## 图卷积神经网络概述

（褚　磊　整理）

深度学习的快速发展推进了模式识别和数据挖掘领域的研究。目标检测、机器翻译、语音识别等许多机器学习任务曾高度依赖手工特征工程来提取信息特征集合，但多种端到端深度学习方式（即卷积神经网络、长短期记忆网络和自编码器）改变了这种状况。深度学习在多个领域的成功主要归功于计算资源的快速发展（如 GPU ）、大量训练数据的收集，还有深度学习从欧几里得数据（如图像、文本和视频）中提取潜在表征的有效性。例如卷积神经网络可以利用平移不变性、局部连通性和图像数据语义合成性，从而提取出与整个数据集共享的局部有意义的特征，用于各种图像分析任务。

尽管深度学习已经在欧几里得数据中取得了很大的成功，但真实世界中，许多重要的数据集都是以图或者网络的形式存在的，即从非欧几里得域生成的数据。例如，在电子商务领域，一个基于图的学习系统能够利用用户和产品之间的交互以实现高度精准的推荐。在化学领域，分子被建模为图，新药研发需要测定相关分子的生物活性。在论文引用网络中，论文之间通过引用关系互相连接，将它们分成不同的类别依赖于复杂的引用关系。

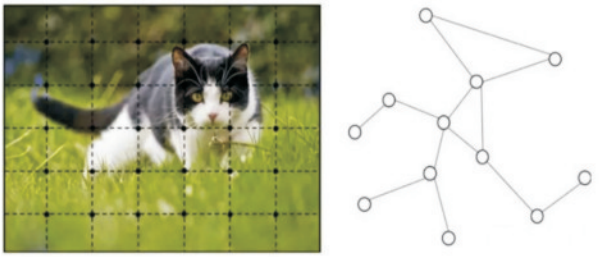


图 1 左：欧几里得数据　右：非欧几里得数据

图数据的复杂性对现有机器学习算法提出了重大挑战，因为图数据是不规则的。对于 graph level ，每张图大小不同、节点无序，对于 node level ，同一张图中的每个节点都有不同数目的邻近节点，使得一些在图像中容易计算的重要运算（如卷积）不能再直接应用于图结构数据。此外，现有机器学习算法的核心假设是实例彼此独立。然而，图数据中的每个实例都与周围的其它实例相关，含有一些复杂的连接信息，用于捕获数据之间的依赖关系，包括引用、朋友关系和相互作用。

最近，越来越多的研究开始将深度学习方法应用到图数据领域。受到深度学习领域进展的驱动，为了应对图数据的复杂性，重要运算

的泛化和定义，研究人员在设计图神经网络的架构时借鉴了卷积网络、循环网络和深度自编码器的思想。例如，图 2 展示了从二维卷积启发得到的图卷积。

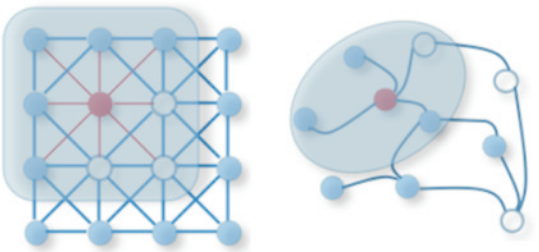


图 2 左：二维卷积　右：图卷积

图卷积网络主要分为 Spectral-based 和 Spatial-based（Non-spectral）两类。基于谱的方法依赖于拉普拉斯矩阵的特征分解。它有三个影响。首先，对图的任何扰动都会导致特征基的变化。其次，学习的过滤器是依赖于域的，这意味着它们不能应用于具有不同结构的图。第三，特征分解需要 O(N^3) 计算复杂度和 O(N^2) 内存。滤波器定位在空间域中，学习到的权重可以在图中的不同位置共享。然而，谱方法的一个常见缺点是它们需要加载整个图。基于空间的方法将图形卷积表示为来自邻居的特征信息的聚合。根据图卷积层的不同叠加方式，将空间方法分为基于递归的和基于组合两类。基于递归的方法试图获得节点的稳定状态，而基于组合的方法则试图包含更高阶的邻域信息。在每个层中，两种方法都必须在训练期间更新所有节点上的隐藏状态。但是，它的效率不高，因为它必须将所有中间状态存储到内存中。为了解决这个问题，提出了几种策略，包括针对基于组合方法的切割成子图再训练（如 W. Hamilton& Z. Ying 提出 GraphSage 算法，NIPS2017），以及针对基于递归方法的随机异步训练（如 H. Dai& Z. Kozareva 提出的 SSE 算法，ICML2018）。

## 隐私和机器学习

（方宏彪 整理）

1. 为什么需要隐私保护的机器学习算法?

机器学习算法从大量数据中学习并更新模型参数去编码数据间的关系。理想情况下，机器学习的参数应该去编码通用的信息（如吸烟的人更容易患肺癌），而不是学习训练样本中特定的事实(Jack患有肺癌)。但事实上，机器学习算法那并不能自动地忽略特定细节。因此机器学习模型在某种程度上会暴露关于训练样本的信息（如训练集中的数据的统计信息，某些样本是否存在于训练集中），隐私保护的机器学习算法企图解决这些问题。

2. 如何定义和保证隐私?

学者提出过很多隐私保护的数据分析方法，如经典的匿名化方法通过在分析之前移除敏感信息或者替换为随机值来保护隐私。但是在攻击者能够获取辅助信息的时候，匿名化方法效果将会下降。Dwork等人提出的差分隐私能从理论上提供严格的隐私保证。差分隐私通过引入随机化机制保证单个个体的存在与否无法对最终结果产生明显的影响。

3. 一种基于差分隐私的机器学习范式

Private Aggregation of Teacher Ensembles (PATE) 是Nicolas Papernot 和 Lan Goodfellow提出的一种新颖的满足隐私保护的机器学习框架。PATE框架由两个阶段组成：（1）在不相交的敏感数据上训练多个Teacher模型，出于隐私考虑，这些模型不公开，仅为Student的训练提供标签。（2）Student使用未标记的公开数据，借助Teachers模型进行标记，标记结果基于差分隐私机制进行聚合，最后只发布Student模型。PATE 对模型的选取没有要求，也不需要定制的训练过程，以半监督知识迁移的方式实现了隐私保护。

## 海力士、三星公布DDR5内存细节：最高7.5Gb/s，更省电

（卢浩迪 整理）

sk海力士在国际固态电路大会上公布了首个DDR5内存芯片的细节。同时三星也在此次会议上描阐释了基于低功耗规范的LPDDR5的SDRAM的实现。两家公司计划在2019年底之前发布DDR5产品，其中SK海力士专注于台式机，三星专注于移动设备。

DDR5内存提供了两倍于DDR4内存的带宽，同时电压更低，数据传输效率更高。作为制定DDR5内存标准的组织，原计划于2018年底完成标准制定，但事实上这项工作依旧在进行中，预计在今年年底有相关产品会出现。

此次会议中，SK Hynix的演讲主要集中在16Gb DDR5 SDRAM模块上，提到了其DDR5产品可以提供每针脚6.4Gb/s的速率，工作电压为1.1V，低于现在的DDR4内存，与LPDDR4相当。每个引脚的吞吐量高达6.4Gb/s，16Gb的容量，采用1纳米工艺制造，尺寸仅为76.22平方毫米。与此同时三星称其制造的10nm LPDDR5 SDRAM芯片仅消耗1.05V，同时提供高达7.5G/s的速度。与SK海力士今年计划出货的相比，这同样更高效，更强大。

而在2019年2月19日，JEDEC公布了LPDDR5更新标准，但依旧不是最终标准。根据标准其可以提供比LPDDR4高50%的I/O速率，达到6400MT/s。同时由于更低的工作电压，可以为智能手机、平板、笔记本电脑等便携式设备提供更高的性能和更长的续航时间。

三星和海力士都表明将在19年第四季度发布DDR5内存产品。而Cadence和Micron也已在2018年10月份表示将在19年年底生产DDR5内存。初期DDR5内存价格相较于DDR4内存更贵，大规模上市依旧需要时日。

## 遭遇黑客，5亿枚NEM币不翼而飞，加密货币交易所Coincheck损失5.23亿美金

（万 立 整理）

日本最大的加密货币交易所之一Coincheck，日前遭遇了一场飞来横祸。

Coincheck周五晚间召开新闻发布会上称，由于遭遇黑客攻击，平台全部（5.26亿）NEM币（新经币）被非法转移。根据估算，这批丢失的新经币价值5.23亿美金，这是历史上最大规模的数字货币盗窃案。

根据Coincheck的官方博客，自日本时间周六下午5点起，这家平台已经暂停了日元提现和接受新支付服务。

由于日本是全球最大的加密货币市场，这次事件对于整个币圈都是一场浩劫。上一次类似事件发生在日本是2014年，比特币交易所Mt. Gox因黑客攻击暂停线上交易，随即申请破产保护。

受这次事件影响，比特币下跌至接近1400美元。

新经币（New Economy Movement，缩写NEM）是一种点对点的虚拟货币，位列世界第九位虚拟货币。

在Coincheck的新闻发布会上，公司联合创始人大塚雄介表示，需要做好最坏的打算，即所有用户的资产有可能追不回来。

至于这次事件的发生原因，根据钛媒体报道，通常情况下，客户的资金都会被存放在“冷钱包”中，即秘钥进行离线保存，物理隔离以保证资金安全，但是新经币币种特殊，大塚雄介的解释是，因为“系统性困难”，新经币只能被保存在联网的“热钱包”中。这意味着，新经币时刻暴露在互联网中，面临着黑客

的攻击。

（链接：https://www.36kr.com/p/5116402）

## EOS用户遭受彩虹攻击，用户私钥被窃取

（王秦远 整理）

2018年EOS 账户遭受彩虹攻击，所谓“彩虹攻击”，即通过事先遍历所有常见的助记词组合并预先生成对应的公私钥，然后在网络中寻找已经被彩虹表记录碰撞出来的账户，一旦被彩虹表碰撞，则意味着攻击者拥有了对应账户的控制私钥，验证威胁账号安全。EOS中文治理社区EMAC，已经接到六起报告EOS丢失的事件，丢失的EOS数量从几十个到几十万个不等，直接造成上千万数字资产损失，而资产被盗的原因大多是开通账户注册过程中常见的小问题：比如，助记词选择，账户有效性核实等。

用户私钥的生成最好使用随机字符串进行生成，若允许用户自定义输入则必需输入足够复杂的助记词，私钥的使用必需恪守在本地使用的原则，不允许进行网络传输，对于私钥至少应该加密保存到本地，按照需求进行离线保存使用。

随后，针对可能遭受的“彩虹”攻击，PeckShield定制了一套危机解决方案：

1、披露因助记词使用问题导致的资产被盗漏洞细节，并呼吁EOS社区用户加强安全防范，避免使用空助记词或较弱的助记词组合；

2、启用EOSRescuer（peckshield.com/eosrescuer）公共查询服务，用户可一键查阅自己的账号是否存在安全风险；

3、将已经监测到的存在高安全风险的用户资产暂时转移到特定的安全账户，受影响用户可联系PeckShield进行认领。

为确保用户利益不受侵害，PeckShield会将上述安全账户的所有记录透明公开，并接受第三方媒体的监督。注：用户认领时需提供EOS账户所有权的证明，包括必要的交易记录等。

（链接：http://www.sohu.com/a/240507050_100142711）

## 斯坦福大学领导的工程团队推出了单芯片计算机(a computer-on-a-chip)的原型

（张 鑫 整理）

斯坦福大学领导的一个工程团队开发了一种单芯片计算机的原型，于2月19日在旧金山举行的国际固态电路会议上（the International Solid-State Circuits Conference）公布了这种单芯片计算机的原型。单芯片计算机原型的数据处理和存储电路的耗电量不到任何类似电子设备的十分之一。

斯坦福团队与法国CEA-LETI研究所的研究人员合作，将阻变式存储器（RRAM）作为单芯片计算机的关键元件。RRAM的特点是存储密度，与其他形式的内存相比，它可以在更少的空间内嵌入更多的数据，同时它的能源效率不会对有限的电力供应造成压力;它还可以在芯片休眠期间保留数据。

RRAM还有另一个重要优势。工程师可以直接在处理电路的顶部构建RRAM，以将数据存储和计算集成到单个芯片中。斯坦福大学的研究人员开创了将内存和处理整合到一个芯片中的概念，因为它比在现有情况下在不同芯片之间来回传递数据更快，更节能。法国的CEA-LETI团队负责将RRAM移植到硅处理器上。斯坦福大学的研究小组提高了RRAM的存储能力，并开发了一种算法来延长使用寿命，因此原型机应该能够运行10年。

# 关于解决科研问题的思考和分享

## 序言

非常荣幸能成为这一期的封面人物，能取得这样的成绩除了自身努力之外当然还离不开周围同学老师的鼓励和帮助，以及一些运气。借着这次机会和大家分享一下从日常工作到论文写作等一些经验，可能由于我所见所知甚少，而且我也是从自我的角度出发总结出的经验，所以无法得出一个非常全面完备的结论，大家仁者见仁智者见智，能够从中获得一些帮助或者启发也是很好的。

## 日常与敏捷编程

我在实验室最开始的做科研的体验有幸得益一个非常紧急的任务，为了赶deadline需要我一个多月内从阅读理解算法到完成项目。所以不同于每周组会的进展，我需要全力去推进这个项目，当时微软的老师就提出了类似于敏捷编程的概念，说到底实际上我认为就是“不断迭代，不要求把每一步做到完美，没有问题的时候就全速推进，有问题的时候就及时讨论问题”。这句话虽然说上去很简单，但是实际在用的时候让我收益良多，现在想来我的很多包括思考问题的习惯，甚至是生活的风格也深受这个影响。

由于目标十分清晰，我们就首先把最终目标划分成了几个子步骤，每个步骤前后衔接紧密，任务的科学划分需要对任务有深刻的理解，就像行军打仗规划路线图一样，体现了指挥官指挥的能力，如果任务划分的好，及时的正反馈能极大地激发行动力。并且前后阶段逻辑上紧密相连，使得每一步的意义都很清晰，这样即使遇到了技术上的困难无法跨过去，我们也能及时地分析原因，快速地得出替换方案。

有了这样成功的经验之后，我发现，这其实代表着高效解决问题的一个相对通用的方式。日常生活中（比如科研过程）经常会遇到没有头绪的情况，这个时候其实最有帮助的是，停下来思考，这一步的意义在哪，为什么一定要这么做，我其实真正想从这一步中得到什么，为什么只有这一种特定的公式或者方法？用别的方法行不行？这个方法和别的方法比较优缺点在哪？为了我想达到的最终目标，我该用哪个？当这些问题思考清楚之后，往往我就清楚下一步我该做什么了。当然有的解决方法相对耗时，这个时候就需要考虑时间成本。而且往往当这些事情想清楚之后，我也会充满干劲，因为这是我自己思考得出的成果，我非常乐于见到我思考的东西在现实中实现之后会变成什么样子，我又该如何修正它。

## 没人关心你具体做了什么

在论文写作的过程中，作为初学者的我经常犯的一个错误是，用老师的话说，就是写得像一篇实验报告，详细地描述了我做了什么，怎么做的。这点其实很容易理解，我们学生作为做实验的人，一来，大家本科的时候都做实验，写了无数篇实验报告，详细记录自己做实验的过程是一个非常自然的行为，二来，我们都习惯性地认为这个研究的重难点就在于我们

coding的某一部分代码，因为它解决了之前没有人解决了的问题。但是老师一句话非常一针见血地指出，那些reviewer又不是你的任课老师，他们根本不关心你做了什么，他们只关心他们读了你的文章之后，他们的知识有了哪些增长，文章所有部分都必须紧密围绕这点来写。因为要有知识增长，所以先得给那些不是这个领域的reviewer们有一个知识的background，这也是体现作者对这个领域认识是否恰当，以适当的角度介绍background可以加强本研究的价值和贡献。后续也是在这些background以及目前的state-of-the-art所存在的问题，以及我们是怎么解决这个问题的思路下写作的，而并不是吹嘘自己写出了多牛的代码。吹嘘的部分集中在experiment & result，那个地方才是说自己的代码有多牛的地方，其他地方应该全部是idea层面的推理和逻辑，和代码实现一点关系都没有。

## 神枪手假说

最后一点我认为很有裨益但不直接地产生作用的是，所谓的神枪手假说。即假设一个神枪手在一根杆子上每隔0.1米便开枪射出一个洞，那么生活在这根杆子上的生物中的一位科学家就会总结出他们的世界每隔0.1米就会出现一个洞，并把此当作“真理”。大家在学习物理的过程中都知道，除了公式推导之外，另外一个很重要的部分是记住很多常数的值，比如万有引力常数或者普朗克常量，我认为这点和神枪手假说中0.1米异曲同工。

以我拙见，我们身边的客观世界对我们来说都是“黑盒”，科学家们对这个黑盒进行了很多“黑盒测试”，得出了很多惊人的结论，增加了我们的知识，极大地便利了我们的生活，但这些结论往往都是对所观测到的一系列现象的总结，其正确性并不能证明这个结论就是写在万物背后的规则或者逻辑，也没有理论能做到完全正确。我们能做的，只是不断地以一个尽可能精简的函数去拟合或者描述我们所能观测到的一切，并把这个函数称为规则。从另外一个角度上来看，这个世界上本没有这么多规则，出于不同的目的，我们才有了这些人为的规则，从这个角度上来看一些很让人心存畏惧的问题时，这些问题也往往变得不那么让人感到恐怖。

## 尾声

如果说之前包括本科在内的求学生涯大多都是在学习知识，研究生阶段可以说是初次体验用所学知识解决严肃的实际问题的过程，同时这也是一个很好静下心来审视周围一切事物的机会，如果说本科毕业大家还是会尽量选择升学，研究生毕业后会渐渐面临各种各样人生的选择，如果能利用好这段时间仔细思考自己真正适合的职业和真正想过的生活，这也不失为一个很好的收获。

转眼硕士就要毕业了，在这几年间，认识了很棒的同学和老师，对我有不同程度的启迪。感谢实验室开放的大学术环境让我有机会接触到了不同的课题，开拓了我的眼界，也增加了我对领域系统的思考，祝各位老师事业有成，科研顺利，同学们顺利毕业，offer多多。

**刘长鸣**
2016级硕士
研究方向：多媒体安全，软件安全
Email：liuchangming@hust.edu.cn

# 云计算与移动计算组研究规划

吴　松，刘方明，余　辰，陆　枫，肖　江，顾　琳

本方向在2019年将重点围绕边缘计算、区块链和数据中心等方面开展研究工作。包括面向边缘计算的超轻量级虚拟化关键技术、动态负载调度技术和相关典型应用，面向区块链跨链交互的双引擎平台和基于区块链的医疗健康数据安全共享技术，高效能云计算数据中心调度/节能新技术和智能化网络架构新方法，等等。主要研究规划和重点关注的研究内容如下。

**（1）面向边缘计算的超轻量级虚拟化关键技术**

随着万物互联时代的到来和无线网络的普及，网络边缘的设备数量和产生的数据都在飞速增长。传统云计算在庞大的数据压力、延迟敏感型和低功耗要求等应用场景下都显现出了弊端。边缘计算（Edge Computing）在传统云计算发展受限的背景下产生，它将云计算的能力拓展至距离终端最近的边缘侧，并通过云边端的统一管理实现云计算服务的下沉。本方向将围绕边缘计算的核心系统、调度机制、典型应用开展研究工作。在面向边缘计算的核心系统方面，主要研究针对该场景的超轻量级虚拟化关键技术，首先研究基于边缘服务器的超轻量级特型内核构建和应用封装方法，通过结合库操作系统和硬件虚拟化技术，为应用构建定制化的特型内核，支持应用的高效部署和快速启动；其次研究基于超轻量级虚拟化的基础运行环境，构建统一的边缘服务器平台，并以此为基础构建标准化的人机物融合系统支撑环境。

**（2）面向边缘计算的动态负载调度技术与系统**

边缘计算是在靠近端设备或数据源头的一侧综合网络、计算、存储等核心能力为一体的开放平台，以实现低延迟和高性能的边缘计算应用。边缘计算应用的灵活性与移动性使其对各类资源的需求在时间与空间上动态变化，然而边缘节点的资源受限，需要对边缘节点上的负载（服务放置与资源缓存等）进行智能调度规划，以更好地满足各类边缘计算应用的服务需求。为此，将针对边缘计算场景和新兴应用设计动态负载调度机制和系统，从实时的边缘计算环境提取负载数据以不断更新智能调度模型和策略，满足边缘计算应用的资源需求和性能目标；将搭建边缘计算系统环境和新兴应用实例，并收集和分析实时的应用负载数据，在实际系统和真实数据集下验证所提出机制和系统的有效性和实用性，并与传统策略深入对比。

**（3）基于边缘计算的健康监测和疾病风险评估**

针对基于穿戴式设备的健康医疗应用，研究物联网环境下的健康监测和早期疾病风险评估与个性化干预技术，主要研究内容包括：1）面向穿戴式设备数据采集的高能效数据传输策略。传统边缘/终端设备的数据传输在安全性和可靠性无法满足现有医疗健康大数据应用的需要，拟研究传输能效高、隔离性强的边缘设备数据传输技术，通过5G、Wifi等异构网络的能耗特性，为应用构建定制化的数据传输策略，支持应用的高能效数据传输。2）基于深度学习的日常健康监测和早期疾病预警。基于机器学习或深度学习方法的疾病预测模型效果在很大程度上依赖于带明确标签的数据和代表性特征提取。依据迁移学习和强化学习等新一代深度学习方法可以降低对医护人员标注数据的要求。通过对患者的生理和病理情况实施跟踪从

而实施反馈调整，以优化多阶段决策的利弊权衡，从而实施疾病早期预警和个性化干预。

**（4）面向区块链跨链交互的双引擎平台**

区块链跨链技术可促进不同区块链之间的信息流通与协同操作，打破制约物联网发展的数据孤岛桎梏，构建丰富的区块链应用交互场景，对未来区块链生态系统建设起举足轻重的作用。研究小组拟提出适用于同构/异构区块链跨链交互的双引擎平台，具体包括：1）存储引擎，弹性融合NewSQL存储方式，提供面向多元化数据的交互式存储及查询服务支持，同时有效保障数据所有者的隐私安全及数据控制权；与2）网络引擎：构建直接高效交互网络管道，设计跨链通信协议及分层拓扑模型以优化网络资源，在保障通信安全的前提下提升通信效率及可扩展性，以支撑分布式高性能实时处理功能。

**（5）基于区块链的医疗健康数据可追溯安全共享**

在医疗健康数据的共享解决方案中，现有的中心化解决思路都无法避免数据被工作人员泄露的风险。区块链的提出为医疗敏感数据的安全共享带来契机。本方向将研究基于区块链存储RWD数据，不仅希望能在普适意义上得到公众和医疗机构等的安全共享认可，解决内部人员泄密问题，并且可以采取策略实现数据操作行为的监控，即实现数据的可追溯安全共享。这不仅能够有助于解决医患和医疗机构之间的纠纷，而且由于系统可信地记录了敏感数据的产生来源与贡献者，因此能更进一步维护医患的共同利益。

**（6）高效能云计算数据中心节能**

云计算数据中心业务持续增长，容器化、跨域化、液冷化成为主要趋势，计算能力和密度要求越来越高，需要在性能与节能各方面进一步突破：1）跨区域容器负载均衡技术：目前OpenStack官方项目Tricircle解决了不同区域之间OpenStack实例通信网络的自动化部署问题，为跨区域容器通信奠定了基础。然而，对于部署在跨区域容器中的应用而言，如何实现负载均衡机制成为了亟需解决的问题，具体需要填补的空白包括：在单一区域场景下，OpenStack官方负载均衡项目Octavia并不针对容器提供负载均衡服务；在跨区域场景下，Tricircle也未提供跨区域的容器负载均衡服务。为此，将拓展Tricircle和Octavia，通过OpenStack API设计和实现跨区域容器负载均衡机制，包括实现对负载均衡服务所需的计算、网络等相关资源的管理；实现根据特定负载均衡策略通过Virtual IP分发负载等功能；将上述功能和机制开源到OpenStack社区且通过社区评审认可。2）数据中心液冷系统的能效优化：愈发密集的服务器导致传统风冷难以满足散热冷却需求，而液冷技术具有能耗低、效率高的特点，逐渐成为数据中心制冷的首选方案。然而，液冷系统仍然存在诸多问题，如其制冷效率和功耗受制于供水温度、制冷缺乏细粒度控制、成本较高等。如何增强液冷系统在数据中心中的可用性、绿色节能、动态冷却能力，仍是目前数据中心能效优化的前沿方向。将针对数据中心负载的时空差异性，提出“温水”冷却策略，通过预测服务器负载，对液冷系统水温实现动态调控，并细粒度优化数据中心制冷能效。

**（7）数据中心智能化网络服务架构**

网络功能虚拟化通过虚拟化技术增加网络服务的灵活性并降低了部署和升级的开销，为快速部署和调度网络服务提供了可能。然而，网络服务的多样化、服务部署的灵活性和服务规模的扩大化同时也对网络服务的管理提出了严峻的挑战。网络资源和服务需求都具有的明显异构性，面对高度动态的网络服务需求和物理网络结构，当前网络服务管理还停留在基于

既定规则或者数学模型的静态架构，不能充分利用网络功能虚拟化的优势。为保障网络服务的服务质量和能效，如何智能化地对网络服务进行部署、调度、负载均衡管理，建立“感知–决策–执行”一体的网络服务智慧化管理，实现智能感知、自主决策、自动执行，并保证服务的灵活性和可扩展性是网络功能虚拟化技术普及过程中一个亟待攻克的关键和难点。本方向拟从网络功能智能化管理策略、动态环境下网络服务的智能调度算法、实时网络流负载均衡和数据包调度策略、和大规模网络服务下多智能体协作算法四方面开展工作，以达到合理分配网络资源、提升网络服务效能，降低整体网络服务成本的目的。

**（8）基于场景的模式识别与基于多源数据的分析推理**

通过普适计算环境下对场景的描述，重点研究基于多角度多维度的人脸识别与比对技术，研究基于行为特征的模式识别技术；研究对象标定与提取技术，为普适场景下的人物对象识别、跟踪、检索与行为、位置预测奠定应用基础；通过对多源数据在实际场景中的应用需求解析，研究多维多源数据的描述方法，研究机器学习方法/深度学习方法在多源数据分析中的应用；研究工业相结合的特殊领域中的数据分析方法；建立数据驱动的智能化策略生成机制，研究数据驱动的推理机制。

目前，围绕着上述研究目标，研究小组已经部署人力开展相关研究，系统研发也正在开展，一方面拟争取能在2019年的顶级学术会议和期刊上发表研究成果，扩大研究工作的影响，另一方面将努力抓住国家重点研发计划和国家自然科学基金的申请机遇，争取获得国家项目支持。同时，小组也会持续重视科研成果转化，将科研成果应用到重要云计算企业和实际的生产性环境中。

**吴　松**

教授、博导，实验室副主任

研究方向：云计算、虚拟化

Email：wusong@hust.edu.cn

**刘方明**

教授、博导

研究方向：云计算与数据中心、软件定义的网络SDN与虚拟化技术、绿色计算与通信、移动互联网。

Email：fmliu@hust.edu.cn

**余　辰**

教授、博导

研究方向：普适计算、移动互联网

Email：yuchen@hust.edu.cn

**陆　枫**

副教授

研究方向：高性能计算应用、移动互联网等方向的研究。

Email：lufeng@hust.edu.cn

**肖　江**

副教授、博导

研究方向：分布式计算与系统、大数据分析与处理、无线网络与移动计算

Email：jiangxiao@hust.edu.cn

**顾　琳**

讲师

研究方向：网络功能虚拟化、软件定义网络、云计算

Email：anheeno@mail.com

# 支撑大数据处理的高效体系结构和系统软件

## ——体系结构与系统软件组科研规划

廖小飞，刘海坤，蒋文斌，邵志远，郑　然，郑　龙，张　宇

## 1 背景与挑战

由于大数据处理正日益表现出数据规模的海量性、数据依赖的复杂性和内在的强并行性等显著特点，现有计算系统由于处理器架构、内存架构以及处理模式方面都存在着难以逾越的鸿沟，已不能高效支撑大数据处理需求。当前，各类新型处理/存储器件的出现为更好的支撑大数据处理提供了可能。一方面，新型加速器架构（FPGA以及ASIC芯片等）为大数据处理提供了面向各类大数据应用可定制的处理器设计可能，允许其充分减少数据访问开销，充分利用丰富的片内并行性；另一方面，新型存储器件技术（PCM等）快速发展，也为构建新的低延迟、高吞吐的大数据处理的内存读写环境提供了可能。然而，对于各类大数据处理应用，这些新型处理器件和存储器件在系统架构、运作机制和使用方式等方面仍然面临着巨大挑战。具体来说，在支持图计算和深度学习等典型大数据应用时，异构平台及其上层系统软件在编程模型、执行方式、数据放置和任务调度等方面分别面临着诸多挑战，以简单有效地管理和利用这些异构计算及存储资源。

**1）以大数据应用为代表的高时效处理需求给异构计算环境带来了挑战**

以CPU为代表的现代控制流体系结构遵循统一化设计理念，仍是当前大数据处理平台的主要载体，尤其在处理复杂计算任务方面具有难以撼动的地位。但是，以图计算、深度学习为代表的大数据市场迅猛增长，由于其体量庞大、类型复杂、处理模式多样的特征，无疑给计算机体系结构的处理能力提出了更高的要求，这种要求不仅仅体现在计算性能上，更是对能效比提出了要求，使得现代控制流体系结构越来越难以匹配多样化的数据处理需求。

FPGA和ASIC 等专用化处理设备因其并行度高、能耗低而备受青睐，考虑其难以胜任较复杂的计算任务，通常需要与CPU 协同使用，然而，这样的异构平台因频繁的数据通信加速效果并不理想。数据流模型以数据驱动指令执行，指令执行顺序完全取决于数据依赖关系和计算速度，天然解决同步通信开销大的难题。历经多年发展的数据流模型有成熟的理论体系支撑，有利于高时效处理架构的研发，而且定制软件体系结构相比定制硬件也明显具有更强的灵活性。

然而，设计基于数据流的高时效大数据处理环境，还须考虑到软硬件资源的复杂性。性能、成本、能效与擅长领域各异的计算和存储设备不断涌现，如各类新型的异构计算核心（如GPU、FPGA、MIC）与多层次新型存储器件（非易失性存储介质NVM、3D堆叠缓存）。这些丰富的硬件资源适合不同阶段的高效处理需求，何时使用何种资源进行何种任务的处理无疑将能有效提升大数据处理的性能。此外，资源的多层属性与数据流计算任务的细粒度诉求之间存在着矛盾，进一步为复杂软硬件资源的充分利用带来了的新挑战，制约了大规模数据流计算任务部署和运行的效率。如何充分考

虑新型体系结构异构计算核心和多层次存储环境的资源复杂性，亟待设计数据流计算的执行引擎和软件环境。

**2）以高时效大数据处理需求对异构内存管理带来的挑战**

现有计算系统在面向特定场景的数据处理加速、内存架构以及处理模式方面都不能高效支撑大数据处理高时效，低功耗的要求。当前，融合多核/众核/加速器等异构计算和新型存储器件的计算机体系结构为更好的支撑大数据处理提供了可能，同时也带来了巨大的挑战。

首先，NVM进一步放大了传统系统软件栈对数据存取层次过深，软件开销太大的问题。如何减少软件层在数据管理、数据存储、和数据传输过程中的开销，是高时效大数据处理系统面临的一个重要挑战。

其次，由于单节点服务器在空间尺寸上的扩展性不足，通过跨节点的内存资源虚拟化和共享可以有效提高内存资源的容量。为此，需要研究低延迟的异构内存池化关键技术，在网络拓扑和协议上实现可扩展和低延迟端到端传输支持，对本地内存和远程内存访问的调度优化、数据放置策略及数据一致性保障机制。

最后，数据中心应用的一个趋势是操作系统轻量化。一个核心问题是如何构建执行特定应用的虚拟机操作系统，减少操作系统中其它不相关功能组件造成的资源浪费和性能干扰。并且，如何在虚拟机启动时给出一些启发式信息，指导虚拟机在异构内存中的数据放置策略。

## 2 主要研究内容

基于上述分析，本研究组将继续延续过去几年制定的研究路线，在面向大数据处理的数据流体系结构、面向虚拟化场景的异构内存系统两个主要方面开展研究，并特别面向图计算需求，开展异构平台上的图计算系统的研究。

**1）在面向复杂大数据处理需求的数据流体系结构设计方面，开展硬件层面的灵活定制、执行层面的高效调度和通信层面的动态划分三个关键问题的研究，提出数据流体系结构的设计规范和设计方法。**

第一，开展面向大数据高效处理的数据流体系结构研究，主要包括数据流抽象机、数据流加速器、存算联动机制三个方面。在数据流抽象机方面，重点突破基于数据流的并行计算理论、数据流图抽象模型、基于数据流的高效存储模型、数据流抽象机的形式化验证系统；在数据流加速器设计方面，研究基于FPGA的数据流加速器，突破面向大数据处理的高并发流水结构、基于FPGA的高吞吐数据流处理模式等关键技术，重点研究多级流水功能模块间的关联关系和负载平衡机制，探讨定制化高并发流水模式的可重构方法，开展数据流加速器的仿真验证；在存算联动机制方面，聚合主机和加速器存储空间，实现多模式协同的组合分析与调度，引入新型存储器件（如High Bandwidth Memory），研究低通信延迟的数据管理与调度方法，实现高效一体化的集中管理与计算。

第二，开展面向大数据高效处理的数据流执行引擎和软件环境研究，主要包括数据流图实时动态映射、数据流加速器多级存储与管理、编程与适配环境三个方面。在数据流图实时动态映射方面，探索平衡任务适配度及计算单元实时负载的自适应任务映射方法，研究数据流任务间的数据依赖关系，实现大数据应用数据流图关键路径识别及瓶颈任务优先加速策略。在数据流加速器多级存储与管理方面，研究加速器片上存储的数据替换及预取策略，实现数据访问特征敏感的数据自动迁移机制，平衡应用在生命周期内对数据流加速器存储结构的访问压力；在编程与适配环境，研究用户友好的数据流软件编程与适配环境，设计标准库的统一中间层抽象，实现用户程序对异构计算单元的自动适配。

第三，开展基于数据流的大数据高效处理

的分布式通信优化方法研究，主要包括数据流图时空估计模型、数据流图分布式动态划分、分布式高通量任务调度三个方面。在数据流图时空估计模型，研究时间维度的执行依赖分析模型与时间运行估计模型，空间维度的数据传输模型与消息交互模型；在数据流图分布式动态划分方面，研究应用、数据、资源利用和网络拓扑感知的分布式数据流图动态划分策略，研究相应低复杂度分布式数据流在线任务动态划分策略；在分布式高通量任务调度，利用数据流体系结构乱序执行的特点，研究子数据流图间的独立任务分布式调度策略。

**2）在面向大数据高时效处理操作系统方面，设计并实现面向异构内存的系统软件栈，基于RDMA的异构内存池化机制，开展云环境下特定需求下的虚拟机定制优化，基于效益的异构内存动态分配机制等研究工作。**

第一，在工业界非易失性存储介质硬件量产后，完善系统软件栈和应用开发接口，开发面向大数据处理的低延迟、高吞吐操作系统。实现操作系统对异构内存的访存调度和数据管理机制，研究多通道异构内存系统中的数据布局策略，平衡多个内存通道的带宽利用，提高异构内存子系统的内存带宽利用率。支持细粒度对象在异构内存中的静态放置和动态迁移：包括离线程序访存信息收集和特征分析，基于对象的异构内存分配器，基于性能和能耗效用的数据对象放置模型，以及热点对象数据的在线迁移机制；异构内存中基于大页的内存管理机制：在异构内存中使用大页可以提高地址转换效率，但和细粒度的内存管理（如页迁移）相冲突，课题将研究同时支持大页和细粒度页迁移的方法，支持连续地址空间中多粒度大页动态构建，大页中的热点数据的轻量级迁移机制等。

第二，异构内存环境下的轻量级虚拟机定制优化。基于LibOS的思想，以库的形式提供操作系统中的某个功能服务，例如网络功能，根据所要运行的应用，选取一个可用于构建操作系统的最小库集合，随后，通过这些库和相应配置文件组成unikernels，构建轻量级的虚拟机。此外，还要充分考虑在异构内存环境下虚拟机不同介质的内存分配对于虚拟机性能影响，通过静态分析，指导虚拟机在启动时把频繁访问的代码段和数据段放置到DRAM，其他数据放置在NVM，减小NFV虚拟机内页面的迁移开销。

第三，基于RDMA的异构内存资源的池化机制和跨节点共享技术。考虑到DRAM和NVM内存资源延迟的差异性，在异构内存网络结构下，应用数据可以有多种放置方式。应用程序的内存分配需要根据对访问延迟的敏感性，对数据进行更加精细的划分，综合考虑内存介质的非易失性、耐久性和能耗特征，网络负载均衡，数据并行性等因素，设计更加灵活的数据放置策略。同时，在RDMA访存机制中，对哈希表的遍历会带来较大的时延，在异构内存共享池中，数据流向复杂，标记众多，key-value数据结构的设计和RDMA友好的hash表设计，将是一个重要的研究问题。需要设计权衡访问延迟（跳数少）和内存数据可靠性（防止全局目录服务器单点失效）的网络拓扑结构，并且具有一定的可扩展性。

**3）在面向图计算的异构计算平台及系统方面，考虑图计算异构并行低、高效表达难、动态划分复杂等多个方面的难题，研究异构体系结构下的图计算软硬件技术体系。**

第一，研究基于FPGA的图计算加速器。设计高效的缓存模型，以及基于数据流的任务调度机制、低延迟通讯与同步架构；具体突破面向图数据的数据缓存和替换模型，高吞吐的多级流水线架构，低延迟的数据流调度机制，多流水间的异步同步机制等关键技术，实现高性能功耗比的FPGA 图计算加速器。

第二，研究图计算运行时资源管理和调

度。构建异构计算部件性能与能耗模型，以及计算任务执行开销的评价方法；提出计算任务的重组织方法、任务的动态调度框架、基于任务效能评价的调度模型、以及图结构特征敏感的调度优化策略。突破任务分解和任务封装、基于任务效能的调度方案、基于图结构特征的计算任务调度与优化机制等关键技术。

第三，研究异构体系结构下的图计算编程环境和高层次综合技术。设计用户简洁、底层透明的图计算支持库和编程框架，提出图计算应用编译的普适性优化机制，设计专属硬件模板，探索硬件模板设计空间的优化搜索机制和硬件模板参数的最优解；构建异构计算平台下易用性和高效性兼顾的编程模型，资源抽象表达与控制能力平衡的运行时技术体系。

第四，研究分布式环境下的图划分和计算方法。突破异构分布式计算环境中面向图计算的通信效率提升、平衡计算和通信的图数据划分、计算扩展性和效率等关键问题，研发分布式环境下图计算的高效执行引擎，从而取得子图高效划分与子图高速执行之间的平衡，实现大规模图的高效能计算。

第五，开展典型图计算应用示范。细化电力图分析、平安医保分析及阿里网上购物行为分析的需求，抽象子图模式匹配计算、聚类计算等重要图计算共性操作，开发典型的图计算系统原型集成上述关键技术，完成医保反欺诈、风险控制、状态分析等场景示范。

## 3 小结

目前，围绕着上述目标和任务，研究小组已经部署人力开展上述研究。小组已经在数据流体系结构方面，图计算机及系统软件、面向虚拟化场景的异构内存系统等方面开展了相关的研究工作。本年度研究方向的部署亟待长期坚持和努力，并通过若干重要应用的示范来展现设计优势。

**廖小飞**
教授、博导，实验室副主任
研究方向：系统软件和系统结构
Email：xfliao@hust.edu.cn

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

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

**邵志远**
副教授
研究方向：图计算
Email：zyshao@hust.edu.cn

**郑　然**
副教授
研究方向：分布式计算、高性能计算应用
E-mail：zhraner@hust.edu.cn

**郑　龙**
博士后
研究方向：可重构计算机体系结构及其运行环境
Email：longzh@hust.edu.cn

**张　宇**
博士后
研究方向：大数据处理、运行时系统及其优化、编程模型
Email：zhyu@hust.edu.cn

# 新型计算模式下的关键安全理论和技术
## ——网络空间安全组科研规划

邹德清，徐　鹏，代炜琦，羌卫中，马晓静，袁　斌

## 1 背景与挑战

平台容器化、服务边缘化、数据中心SDN化是云计算的三个方面的发展趋势，容器、边缘计算、软件定义网络成为新型的云计算模式所依赖的三种基础设施，但是，也带来了安全方面的挑战。首先，以灵活、轻量化为特点的容器技术，因其弱隔离性、高动态性导致了新的安全隐患；其次，SDN的单点故障、错误容忍、多租户隔离等安全问题也亟需解决；最后，边缘计算也存在终端资源受限，难以执行必要的隐私保护算法等问题。

区块链技术作为目前最有希望解决互联网信息可信与价值传输安全这两个问题的技术，它的应用与发展可能会带来一系列的变革，给网络空间环境带来一系列的挑战与发展机会，相关的安全研究、安全防护、安全利用方法也变得重要起来。但因其不可篡改性也导致了由智能合约安全漏洞或丢失密钥等疏忽带来的资产损失难以被追回的问题。

漏洞是各种网络空间安全事件发起的根源，软件系统作为网络空间的一个重要组成部分，其本身的漏洞给网络空间带来了严重的安全威胁。随着软件的开源化趋势成为主流，开源软件已成为软件开发的重要来源，开源软件的模块或组件中一旦存在有意或无意引入的漏洞，这些漏洞将随着开源软件的使用而扩散，呈现雪崩式的传播特点。代码漏洞影响深远，亟需从源头全面开展深层次、大规模、智能化的软件漏洞检测，尽可能早且准确地检测软件漏洞，以便及时修补，更好地保障软件安全。

**1）云平台容器化、SDN化后的高灵活性与高动态性为云环境中传统的安全问题带了新的挑战**

随着容器技术与SDN技术在云环境中的广泛应用，云环境中的服务架构、网络模式也从传统的紧耦合模式中解放出来。其中松耦合的微服务架构与解耦后的网络功能极大地满足了云环境的在易扩展性与通用性等方面的需求。与此同时，容器、微服务架构、SDN等轻量级技术的快速融合与应用带来的高动态性和灵活性不仅满足了云环境性能的需求，还为解决传统安全安全带了新的可能。与传统基于虚拟机的云环境相比，基于容器的云环境（以下简称容器云）在虚拟化技术方面更加轻量，同时在服务扩展方面也更加灵活。这使得容器云中对于资源的分配与使用更加细粒度。

松耦合的特性为容器云中服务的扩展、调度、编排带来了极大的便利，但同时也在一定程度上增加了云环境的复杂性。这种复杂性进一步地制约了防御方案的可行性。如何充分利用容器云中服务架构的灵活性与动态性、容器技术的轻量级特征来高效的设计针对传统安全问题的解决方案是我们研究的重点。同时，在使用SDN动态解决网络安全问题时，SDN的动态转发策略也可能导致网络功能策略失效，不同网络功能的策略的互相冲突，最终引起可信问题。所以如何在基于SDN的云网络中

对网络功能的策略进行可信验证，解决网络功能与SDN设备之间、跨网络功能之间的策略冲突的问题是一个重要的研究议题。

**2）物联网—边缘计算为代表的低能耗、高时效的需求给传统密码防御手段带来了挑战**

随着物联网与云计算融合的不断深入，工业界逐渐迈向了大数据后云时代。海量的物联网设备无时无刻不在产生数据，而物联网应用对服务的响应时间和数据的隐私性都会有更高的要求。传统的云计算模式已经不再适用于物联网应用的低能耗、高时效、高安全需求。这些在网络边缘产生的大数据迫切需要由网络边缘的节点去存储、分析和处理。这种新的计算范式和存储模型被定义成边缘计算并得到了工业界和学术界的广泛关注和研究。各大云计算巨头包括阿里巴巴、亚马逊和微软等都已在边缘计算上进行战略布局。

无论是物联网还是边缘计算，大数据的安全性都是无法回避的核心问题之一。大数据安全甚至与国家网络空间安全密切相关。大数据具有数量庞大、处理迅速、类型多样、高价值等特点，数据在集聚过程中由量变产生质变，集聚程度越高，其重要性和价值性就越高，安全保障的任务也会越来越重。密码技术是保障大数据安全最可靠、最经济、最有效的手段，应积极推进密码技术应用，切实保障大数据安全。而传统密码技术已不能完全使用于新兴的大数据领域，云计算、物联网、边缘计算等新技术的快速发展，对大数据的收集、处理和应用提出了新的安全要求，亟需一种新的密码技术来保护大数据安全。

**3）区块链的去中心化特征带来了安全挑战**

首先，区块链的智能合约漏洞给区块链账户资产安全带来了挑战。智能合约代码简短但逻辑复杂，而且相比传统软件，其更多的偏向于逻辑漏洞，难于指定判断此种漏洞的规则。并且，智能合约的生存环境的复杂性，使分析合约威胁因素艰难。而且智能合约一旦部署上链便无法修改，导致链上漏洞难以修复。执行合约的矿工与合约安全性也不具有利益相关性，相反，他们可能是恶意的，试图破坏合约安全。因此无论是链下还是链上保护智能合约安全都存在很大的挑战。

其次，区块链去中心化特征给资产安全管理带来了挑战。在区块链中，公私钥、地址是用户的唯一标识，用户通过私钥对资产进行操作，一旦丢失，将难以恢复对应资产，蒙受相应损失。为此，需要研究一种基于零知识证明的区块链中的资产恢复方法，它能够根据用户的需求事先在区块链中隐藏关键的恢复信息（譬如恢复人的公钥列表等），当秘钥丢失之后，通过零知识证明的方法，在不泄露关键恢复信息的前提下，证明用户对于资产的所有权，将用户资产从原有地址转移到新地址，实现资产恢复，并保证恢复过程中的安全性和过程可验证。

**4）以多源、复杂、关联为特征的开源软件为软件漏洞检测提出了新挑战**

传统基于模式的漏洞检测方法依赖专家手工定义漏洞规则或特征来描述漏洞，该方法不仅主观性强，而且难以同时达到较低的误报率和漏报率。最近提出的基于深度学习的漏洞检测，不需要专家手工定义特征，可以自动生成漏洞模式，但目前主要利用简单标记的单一漏洞代码数据，尚无法用于多源、复杂、关联数据的环境，漏洞检测效果仍有待提升。如何利用大量多源已知源代码漏洞数据，通过知识表征与关联，借助人工智能技术，减少漏洞检测的误报和漏报，成为未来软件漏洞检测的一个发展趋势。

并且，现有的漏洞数据库尚不能提供足够的信息满足漏洞检测的需求，研究人员往往需

要依据需求自行构建数据库。知识图谱作为一种揭示实体之间关系的语义网络，能够对现实世界的概念及其相互关系进行形式化描述，已广泛应用于智能搜索、智能问答、个性化推荐等领域。大量源代码漏洞数据蕴含了丰富的漏洞知识，基于这些漏洞知识构建知识图谱，能够容纳来自不同类型资源的漏洞知识实体以及实体之间的关系，为利用丰富的漏洞知识提高源代码漏洞检测的有效性提供了可能。现有的漏洞知识图谱研究能够从已有的结构化、半结构化、非结构化数据库中抽取部分漏洞相关本体以及本体之间的关系，但现有的漏洞知识图谱远非完善，仅能够提供较为简单的漏洞信息检索及相关应用，无法进行漏洞检测。

## 2 主要研究内容

基于上述分析，本研究组将在云计算安全、区块链、软件漏洞检测等方面开展研究。

**1）在面向容器化、SDN化的云环境中的传统安全问题以及新型安全问题方面，研究基于容器技术和SDN技术的灵活定制、高效执行的安全防御机制设计，并融合容器技术与SDN技术通过软件定义的方式定制安全防御机制。**

第一，开展面向云计算的容器安全关键技术研究，主要包括容器云内生安全、基于容器云的防御策略、容器云安全防御框架等三个方面。在容器云内生安全方面，重点突破因轻量级虚拟化而造成的容器间的弱隔离问题，探讨并定制在不损耗容器性能的前提下加强容器间隔离的方法；在基于容器云的防御策略方面，基于容器轻量级的特征以及容器云的灵活性、易扩展性，探索云环境中传统安全问题在容器云中的解决方案，实现依托于容器编排、调度机制的高效防御机制。在容器云安全防御框架方面，为容器云提供统一的安全策略部署接口，探讨基于容器云特性防御方案的标准化，实现高效一体化的安全策略部署方案。

第二，开展面向云计算的SDN安全关键技术研究，主要包括SDN安全基础架构、SDN安全管理以及SDN安全可视化等三个方面。在SDN安全基础架构方面，为云网络资源的统一安全管理提供支撑，包括：管理协议、状态收集机制以及安全控制机制，它是云网络软件定义安全的基础；在SDN安全管理方面，基于底层架构的访问接口收集的全局安全状态信息，开展大数据安全分析与决策,并提供对云计算网络资源的自动化/基于反馈/用户定制的多种安全响应；在SDN安全可视化方面，主要研究面向用户服务区域的安全呈现、网络异常重现以及面向系统异常行为的重现，为用户提供全面、直观的相关安全状态信息。

第三，开展融合SDN技术的容器云网络的安全性与可靠性研究，主要包括容器云网络错误容忍、容器联动的网络防御资源调度以及容器云网络的主动防御三个方面。在容器云网络错误容忍方面，重点针对容器网络复杂多变的高动态特性，重点研究在网络设备出现异常（例如故障或被攻破）的情况下，保证网络的响应时间以及服务不中断。在容器联动的网络防御资源调度方面，针对基于容器的安全防御方案（如针对DDoS的动态资源调度），重点研究基于实时的网络状态进行快速的联动网络资源调度方案。在容器云网络的主动防御方面，针对已知或可能的网络攻击，研究基于网络资源调度的主动防御方案，针对不同攻击的攻击特性与时间间隔，设计不同模式的网络路线动态规划与主动迁移方案，使攻击无效化，保证容器云网络与服务的安全。

**2）在面向物联网–边缘计算的大数据机密性保护方面，开展具有实用安全的加密搜索机制、灵活易扩展的混合可搜索加密机制以及高效通用的数据安全交换机制三个关键问题的研究。**

第一，研究实用安全的可搜索公钥加密及

其语义安全性基本模型。该研究目标是构建本研究内容核心理论的重要基础与起点，因此需要深入剖析大数据存储角度可搜索公钥加密存在的安全性问题、需要归纳导致这些问题最本质的原因或因素、需要从实用安全角度研究形成新一代可搜索公钥加密及其语义安全性基本模型的充分与必要条件。在此基础上，需研究可搜索公钥加密在实现多样化检索和提高检索效率方面的核心思想与方法、需要分析多功能性、高检索效率与实用安全性之间可能存在的矛盾、需要找到兼顾多功能性、高检索效率与实用安全性的方法。

第二，研究面向边缘计算的混合可搜索加密。随着物联网的快速发展，大数据的边缘存储将会变得越来越普及。该研究既能填补学术界在该领域的研究空白，也能为大数据在边缘计算或者分布式环境下的安全存储提供解决方案。针对该研究目标，需要着重分析大数据在边缘计算下面临的数据存储的安全威胁模型，研究可搜索加密算法的解构理论，结合特定的可搜索公钥加密和可搜索对称加密方案设计灵活易扩展的混合加密方案。

第三，研究面向边缘计算跨异构密码下的通用代理重加密。在大规模的网络下，边缘节点往往需要协同工作来提供服务，节点各自使用自己的密码系统来保护数据，此时需要一种通用密码手段去支持异构密码体制下不同节点的数据交换。针对该研究目标，需要着重分析大数据在边缘计算下面临的数据交换的安全威胁模型，结合加密共享技术理论，设计适用于异构密码体制的通用代理重加密方案。

**3）在区块链支撑技术方面，开展数字资产安全、基于零知识证明的区块链资产恢复、智能合约安全漏洞防护、智能合约去中心化审计等四个关键问题的研究。**

第一，研究区块链数字资产安全。区块链数字资产的安全性完全取决于私钥的安全性，私钥一旦丢失，用户将失去对资产的控制权，并且永远无法恢复。传统第三方认证机构能够实现用户身份认证，但存在中心化伪造的问题。为此，研究设计一种基于矿工挖矿流程的去中心化身份认证机制，解决传统第三方服务的安全性问题，能够在无密钥的情况下认证用户真实身份，并且能抵抗黑客或合谋攻击，实现密钥丢失后的资产恢复。

第二，研究基于零知识证明的区块链资产恢复。在区块链中，公私钥、地址是用户的唯一标识，用户通过私钥对资产进行操作，一旦丢失，将难以恢复对应资产，蒙受相应损失。为此，研究一种基于零知识证明的区块链中的资产恢复方法，它能够根据用户的需求事先在区块链中隐藏关键的恢复信息（譬如恢复人公钥列表等），当秘钥丢失之后，通过零知识证明的方法，在不泄露关键恢复信息的前提下，证明用户对于资产的所有权，将用户资产从原有地址转移到新地址，实现资产恢复，保证恢复过程中的安全性和过程可验证。

第三，研究智能合约安全漏洞防护。智能合约一旦部署上链便无法修改，即使发现漏洞也无法直接修复。但由于前期安全测试的不完备性，已部署的合约难免会存在漏洞。为实现在不修改智能合约代码的情况下，保护其免受漏洞利用，研究智能合约动态漏洞防护机制。即在交易真正被执行前，检测其是否会触发脆弱合约的漏洞，若会则阻止这种交易执行，否则让其正常执行。

第四，研究智能合约去中心化审计。智能合约漏洞造成的资产损失难以被追回，且当前的合约测试方法过于本地化、中心化，难以跳出思维定式，发现潜藏逻辑漏洞的难度较高。为此，研究提出一个基于双向锚定的侧链技术的去中心化审计平台，将合约率先在侧链上发布，待合约调度频次达到预设阈值，由参与测试用户共识无误后新合约上主链。合约发

布者根据方法中提出的能抵御多方合谋与bug拦截的激励模型提供奖励金用于吸引矿工与广大公链用户参与到侧链上的合约测试。实现合约测试去中心化、自治的同时，提高新合约的公信力。

**4）在软件漏洞检测方面，开展基于知识图谱的智能化源代码漏洞检测技术研究，研究包括多源漏洞数据的采集与分析、漏洞知识图谱的构建、智能化源代码漏洞检测等三个方面。**

第一、研究多源漏洞数据的采集与分析。针对云计算、大数据环境中数据复杂、异构，漏洞数据具有多维以及漏洞代码难收集等特点，研究漏洞基础信息与软件系统信息的获取与特征分析。主要包括漏洞信息智能提取技术、漏洞代码特征提取和漏洞分析方法匹配三方面。

第二、研究漏洞知识图谱的构建。针对漏洞数据的多层次、复杂等造成的多源漏洞知识分析的高难度等问题，研究多源漏洞知识图谱的构建，并维护知识图谱的信息积累与更新。主要包括漏洞知识表示、多源漏洞知识融合、漏洞知识更新、漏洞知识推理四个方面。

第三、研究基于知识图谱的智能化源代码漏洞检测技术。漏洞知识图谱包含了丰富的漏洞信息，但在漏洞检测方面，未见基于知识图谱的源代码漏洞检测技术研究，而且，软件规模越来越庞大，人力无法及时有效地分析处理其中隐藏的漏洞，研究基于知识图谱的智能化源代码漏洞检测技术，具体研究内容包括：漏洞分析过程智能化推荐、基于多元属性匹配的相似漏洞检测、基于聚类算法的漏洞模式推导、基于深度学习的智能化漏洞检测。

## 3 小结

目前，围绕着上述目标和任务，依托重点研发计划、自然科学基金等项目，本研究组已经开展的相关方面的深入研究，同时，面向企业需求，将相关研究成果通过企业委托开发的方式进行应用和推广。

**邹德清**
教授、博导
研究方向：云安全，软件定义安全
Email：deqingzou@hust.edu.cn

**徐　鹏**
副教授、博导
研究方向：密码学、数据安全
Email：xupeng@hust.edu.cn

**代炜琦**
讲师
研究方向：区块链、云安全、可信计算、虚拟化安全
Email：wqdai@hust.edu.cn

**羌卫中**
副教授
研究方向：虚拟化安全、云安全
Email：wzqiang@hust.edu.cn

**马晓静**
副教授、博导
研究方向：多媒体安全、密文数据处理
Email：lindahust@hust.edu.cn

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

# 大数据组研究规划

石宣化，陈汉华，赵　峰，袁平鹏，华强胜，谢　夏，丁晓锋，黄　宏

本组在2019年的研究重点是大数据基础理论（如面向新型体系结构的低通信复杂度分布式算法）、大数据处理支撑技术（如异构图计算、大规模深度学习、HPC I/O调度等）、大数据管理与分析技术（如大规模图数据的表示与存储、大规模知识管理与搜索）等。主要的研究规划和关注的研究内容如下：

**（1）面向新型体系结构的低通信复杂度分布式算法研究**

CPU时钟频率和内存带宽以及网络带宽的发展差距使得主存和缓存之间的数据移动（称为垂直通信）以及处理机之间的数据移动（称为水平通信）相对CPU运算具有更高的延迟和能耗代价，因此设计低通信复杂度（并行垂直和水平通信总开销）算法不仅可以提高吞吐率还可以降低能耗，这在大数据处理背景下尤为重要。非易失内存（NVRAM，譬如相变存储PCM和NVDIMM）由于其具有字节寻址、高存储密度以及读写不对称等特性使得它可以替代传统内存和外存，从而导致计算机体系结构发生变化。和传统垂直通信中只考虑降低读写的总次数不同，由于NVRAM中写比读具有更高的延迟和能耗，因此还要进一步降低写的总次数。

大图计算，譬如求解图的直径、半径，三角形计数，介数中心度求解、K-核分解等是大数据处理的重要分支。国际上高性能计算领域已经开展以线性代数方法来求解大图计算问题的研究，典型的项目是一个由MIT、Berkeley、CMU等美国名校以及Lawrence Berkeley等国家实验室以及Intel、NVDIA等企业参与的GraphBLAS。用代数方法来求解图计算问题的一个主要原因是并行数值计算相对比较成熟，但是图计算主要为稀疏矩阵计算。此外，如前所述，以NVRAM为基础的新型体系结构也对算法设计带来了重大挑战。本项目将以一系列大图计算问题为例，一方面通过把图计算问题转换成相应线性代数求解问题，从而在基于NVRAM的新型体系结构下对这些大图计算问题设计低通信复杂度分布式算法，另一方面还需要证明相应的通信复杂度下界。此外，本项目也对基础线性代数计算问题，譬如各种矩阵分解和Krylov子空间计算方法设计可证明的低通信复杂度分布式算法。

**（2）面向大数据中半结构化数据的加速解析研究**

JSON是一种非常流行的数据格式。它被用于大多数Web应用程序（如Twitter和Facebook）。JSON简单、灵活，具有很强的表达能力。它在应用中作为数据交换格式越来越流行，促使人们对运行JSON数据的分析查询越来越感兴趣。为了满足这种不断增长的需求，许多数据分析引擎（例如Apache Spark、Drill、Storm、和Jaql）都天然支持JSON数据源。然而由于Json数据的多结构性，不能得到现有数据库很好的支持，因此数据分析的过程往往是直接利用Spark等框架在原始数据上进行分析。查询原始数据的一个关键瓶颈是解析这些原始数据本身。特别是对于一些可读性比较强的半结构化

数据如Json，解析的过程是非常昂贵的，因为它们依赖于基于状态机的算法，该算法每输入一字节执行一系列指令，用户获取Json中的某一个字段需要将整个数据都完整的解析出来。为解决Json数据解析代价高昂的问题拟开展如下研究：1）研究用户使用Json数据的行为特征，对用户的常见使用特征进行总结；2）充分利用现在处理器并行处理的能力，探究使用SIMD技术加速Json解析的可能；3）根据用户对于半结构化数据的行为特征设计高效地自动缓存策略；4）深究Json字段的列存可能性，以及针对Json数据索引结构，以实现解析过程中的谓词下推。

**（3）HPC环境下的I/O调度优化**

随着多种新型硬件被引入下一代超算系统，新型超算中心的异构化趋势愈加明显，如何高效合理地利用新型硬件的优势来打破I/O壁垒成为当下亟待解决的问题。I/O壁垒通常指高性能计算中心中，计算性能与I/O性能不匹配的问题，该问题在数据密集型应用普及的背景下愈加严重。

打破I/O壁垒需要重新考虑与设计I/O与计算关系。为此，拟提出使用新型硬件（FPGA）加速I/O模块的研究方向。FPGA在并行加速以及能耗上有着较大优势，可依此开展以下研究1）将文件系统各模块解耦合，挖掘硬件化潜力；2）设计高效合理的文件系统与FPGA的协作模式，避免不必要的开销；3）分析传统HPC应用在新硬件环境下的I/O行为，探索优化点；4）延续以往研究，提高多应用场景下服务质量保障的精度，结合Burst Buffer特性对超额I/O需求进行优化调度。

**（4）异构图计算**

图计算是以图的数据结构来抽象描述事物之间的关联性，并且在该数据结构上进行计算分析的模式。由于图能够很好的描述事物之间的关联性，且在结构和语义等方面具有很强的表达能力，所以使用图计算能够挖掘事物之间潜在不易洞察的行为和联系。然而，现实生活中的图数据不仅数据规模大，而且具有一定的不规则性，存在着跳跃式访存、数据依赖性强、计算/通信比例低等特点。由于图计算的特点对通用CPU计算系统带来了巨大挑战。

由于专用加速器往往具有丰富的带宽资源、超高的并行能力及极低的数据传输延迟等技术优势，研究如何利用诸如GPU、MIC、FPGA等加速器来设计异构图计算系统加速图计算是当前的研究热点。2019年本组在围绕异构图计算系统方面拟展开以下研究：1）研究基于异构体系结构（以GPU为例）下的高效图着色算法，为异构图计算任务管理、数据划分提供基础支撑；2）研究基于异构体系结构（以GPU为例）上动态图的存储策略，支持异构图的快速更新。

**（5）大规模深度学习系统研究**

深度学习在图像识别，目标跟踪，自动驾驶，机器翻译等领域取得卓越性能同时，其耗时的训练和推测过程也给各种加速器带来存储压力过大，计算效率底下，功耗高的问题。随着新的网络架构层出不穷，深度学习朝着动态，异构，大模型发展，特别是应用场景从传统的图像识别领域迁移到大数据处理领域，处理数据对象从单个图片，视频到大规模图数据结构（社交网络，蛋白质分子图，知识图谱等），缺乏灵活高效，易于使用的系统支撑已经成为主要瓶颈。基于此。在2019年。我们将做以下的工作。

a. 针对于处理大规模图数据的深度学习系

统设计。

着力解决深度学习应用于大规模图数据系统短板，现有的深度学习系统普遍缺乏对大规模图数据处理的支持，比如消息传播等，而传统的图处理系统无法直观表达深度学习的特有的梯度传播和更新问题。我们将图计算框架引入深度学习系统，提出新的编程处理模型，高效的资源调度架构，数据资源组织方式。实现对于大规模图数据的高效处理。从而支持大规模图嵌入，知识挖掘，个性化推荐等应用场景。

b. 针对于大模型的深度学习系统的内存，计算瓶颈优化。

深度学习训练过程中存在大量密集计算，同时模型和中间结果给加速器（比如GPU, FPGA, ASIC等）带来巨大内存压力。基于此，我们将研究。1）研究透明高效的内存管理机制。从编译层出发对深度学习的访存进行深度优化，实现框架和应用无关，自动的加速器内存管理。2）研究适用多核架构的复杂网络的调度，并行策略。针对复杂，异构神经网络，比如循环神经网络，动态神经网络，树形神经网络，对抗神经网络（GAN）等，提出高效batch 机制，路径选择和kernel优化等。

**（6）高时效大数据流连接处理平台**

云计算、物联网等新兴技术产业的飞速发展促使了大数据的爆发，并给数据处理系统带来了巨大的压力。在大数据时代，每时每刻有大量的系统和用户，以极快的速度产生各式各样的数据。越来越多的应用需要具备对实时产生的大数据流进行快速处理的能力，以获取其价值。然而，实际应用中大规模实时流处理系统面临数据泛在多源、关联稀疏、倾斜性高、时序性强等特征，带来时空开销大、资源利用率低、系统扩展性差等严峻挑战，研发高时效可扩展数据流连接系统成为亟待解决的课题，具有重要的学术价值和应用价值。为了解决泛在多源数据流的动态性、倾斜性、时序性等特征给实时流连接处理系统带来的扩展性差、资源利用率低、时效性难以保证的问题，本方向未来一年拟重点开展如下研究：1）可扩展多路流连接处理模型及系统结构；2）面向高倾斜数据的分布式流连接负载均衡机制；3）面向复杂事件处理的时序流数据分布式计算引擎；4）基于FPGA的流连接并行处理系统。

**（7）巨量图的计算与学习**

图是表达复杂关系和知识的有效方式之一。而图的形式多样，比如无标签的简单图、带标签的属性图以及时态图, 同时图数据也在不断的增长。因此，如何高效的对巨量多模态的图数据进行存储、处理和学习是本组研究的重点。具体的探索如下：1）研究自适应而高效的图划分框架。综合图的拓扑结构、语义信息、均衡局部性、数据冗余等多种因素，研究高效的大图划分方法；2）研究大图数据的存储模型和索引机制；3）研究适应新型体系结构下的并行分布式的调度方案，比如海量主存、多级缓存、异构众核、GPU等下的图处理任务调度策略。4）研究图数据划分感知的查询分解方法，适应大规模并行分布处理的大图查询计划生成方法；5）研究时态图的相关问题，比如时态图的稠密子图的挖掘，最短路径和可达性的查询等；6）研究基于知识图谱的智能问答技术，比如自然语言问题语义理解，面向智能问答的知识表示和问题生成等。

**（8）基于多视角的网络表征学习**

随着网络规模的不断发展和壮大，对网络

的表示学习也成为网络研究的一大热点。良好的网络表征方法能有效提高网络后续任务的执行能力和执行效率。目前对网络的表征学习多基于单视角展开，然而，网络中的丰富信息量并不仅仅只存在于单一视角之中，多视角的引入必将提高网络的表示能力，因此，本研究将基于多视角的网络表征学习展开。传统的多视角学习往往只关注多视角中的某一方面，如不同视角的耦合信息，或者不同视角的独特信息。本研究将充分考虑这两方面的信息，并对其进行合理的选择。具体拟展开以下研究：1）多视角耦合信息与独特信息的选择决策；2）基于多视角的网络表征学习方法的研究；3）多视角网络中冗余信息的有效处理。

**（9）抗隐私分析的大数据查询处理**

随着互联网和物联网技术的蓬勃发展，在诸如城市交通、医学研究、航空航天、地质气候检测等数据密集型的应用领域，数据规模庞大、类型繁多且高速增长，引起前所未有的数据存储、查询与管理的复杂难题。大数据是当前学术界和产业界的研究热点,正影响着人们日常生活方式、工作习惯及思考模式，但目前大数据在收集、存储和处理过程中面临着诸多安全风险。大数据所导致的隐私泄露为用户带来严重困扰,虚假数据将导致错误或无效的大数据分析结果，这就对大数据管理平台提出了全新的挑战。为了满足各种隐私制约型数据查询与处理的高效性，需要解决传统数据库查询处理方法不能应对面向隐私保护、大数据日益复杂的查询处理需求，以及满足基本的数据查询、数据分析及较为复杂的针对二维或高维数据查询处理等。基于以上诸多问题以及要求，拟开展以下研究：1）建立面向隐私保护的大数据分布、存储、检索与访问的分布式体系架构，研究满足隐私保护需求的分布式文件系统；2）设计支持并行查询的隐私预算分配策略，以及满足隐私预算分配的并行任务调度方法；3）研究在流数据处理模式下的隐私保护模型，以及满足包括差分隐私等各种标准的查询算法；4）研究面向隐私保护的深度学习算法及优化方法，及面向隐私保护的新型神经网络结构；5）研究支持隐私参数设置的高效并行查询语言及用户编程模型，以及使用SQL 语言进行隐私制约型查询的表达方法。

**（10）大规模知识管理与搜索**

数据和内容是大数据网络的核心，针对大数据数据体量大、种类多、价值密度低等特点所导致的数据利用率低等问题，如何有效地对多源异构的大规模数据进行高效的存储管理，并准确获取我们所需的价值数据具有重要的研究意义，也是大数据网络信息管理的核心价值所在。有鉴于此，我们主要研究：（1）大数据环境下的知识管理，从知识生成、存储、组织、自治、导航、检索等层次出发，研究数据驱动的知识发现基本理论和关键技术。如：研究基于多源异构大数据的处理转化机制，构建海量知识库的高效抽取功能框架，并实现多知识库的统一映射转化技术；研究知识图谱的构建技术，研究针对元数据、知识实体及语义关联信息的自动抽取技术，智能构建知识图谱；研究知识图谱的弹性存储模型，支持大规模知识库的增量式更新，研究弹性存储模型以及访问策略等。（2）大数据的实时搜索，检索用户所需要的信息，理解数据空间和网络空间之间的交互，对其进行可靠管理、准确分析和全面理解，并提供个性化的服务，主要内容包括深层网络搜索、社交网络搜索、泛在网络信息搜索、数据流链路预测等。（3）大数据的智能推

演技术，推理并挖掘出网络大数据背后隐含的关联信息，提供数据的智能化服务，主要内容包括构建基于嵌入式知识图谱的知识库表示模型，研究嵌入式知识图谱表示的关系推演算法，知识图网络匹配算法、知识图网络关联分析等技术。

**（11）大规模图数据的表示与存储**

作为数据挖掘领域经典算法之一k近邻算法（K Nearest Neighbor）在实际应用中有着重要的地位，传统的knn算法面临着负样本影响分类准确性的问题，通过对抗学习网络提高knn算法在存在负样本条件下的准确性具有重大意义。对大规模图数据进行计算，往往需要耗费大量的硬件资源和运行时间。图摘要作为对原始图的一种简洁表示，在图的存储压缩、加速查询、交互式分析、可视化等方面有着重要的作用，能够减少存储空间并且降低查询时间。随着社交平台的发展，网络结构的数据越来越受人们的关注，同时对网络的分析也面临着和对高维数据挖掘一样的挑战——维度灾难。本方向未来一年拟重点开展如下研究：（1）利用张量的多维关联特点，研究基于布尔张量分解的属性图摘要（Graph summarization）算法。（2）研究k近邻算法在对抗样本存在的情况下的鲁棒性研究，保证分类的准确性。（3）研究网络表示学习（Network Emebdding）算法，即对复杂网络中的节点用低维向量表示。（4）研究对医疗数据的获取与分析。

**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

**陈汉华**
教授，博导
研究方向：大数据处理系统、分布式计算系统
Email：chen@hust.edu.cn

**赵　峰**
教授，博导
研究方向：信息检索、数据挖掘与分布式计算等
Email：zhaof@hust.edu.cn

**袁平鹏**
教授，博导
研究方向：图数据库、图处理系统、分布式计算等
Email：ppyuan@hust.edu.cn

**华强胜**
副教授，博导
研究方向：算法与体系结构协同设计，大数据算法与理论，并行与分布式计算
Email：qshua@hust.edu.cn

**谢　夏**
副教授
研究方向：海量数据挖掘，并行计算
Email：shelicy@hust.edu.cn

**丁晓锋**
副教授，博导
研究方向：大数据管理系统，隐私保护，深度学习
Email：xfding@hust.edu.cn

**黄　宏**
讲师
研究方向：社会计算，大数据分析与挖掘
Email：honghuang@hust.edu.cn

# Linux内核CFS调度器

（杨晓礼 https://www.cnblogs.com/XiaoliBoy/p/10410686.html）

Linux内核作为一个通用的操作系统（OS），需要兼顾各种各样类型的进程，包括实时进程、交互式进程、批处理进程等。而调度器（Scheduler）作为OS的核心组件——CPU时间的管理器，主要负责选择某些就绪的进程来执行。不同的调度器根据不同的方法挑选出最适合运行的进程。目前，在Linux内核中支持的调度器有CFS调度器、Realtime调度器、Deadline调度器和Idle调度器 。本篇将简单介绍CFS调度器的设计原理。

CFS（完全公平调度器）实现的主要思想是维护为任务提供处理器时间方面的平衡（公平性），这意味着应给进程分配相当数量的处理器。分给某个任务的时间失去平衡时（意味着一个或多个任务相对于其他任务而言未被给予相当数量的时间），应给失去平衡的任务分配时间，让其执行。

CFS通过虚拟运行时间（vruntime）来实现平衡，维护提供给某个任务的时间量。进程的虚拟时间是指实际运行时间相对于权重为0的进程的比例值。在CFS调度器中有一个计算虚拟时间的核心函数calc_delta_fair()，它的计算公式为：

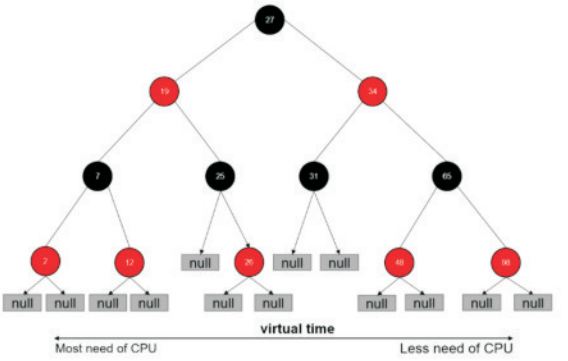
vruntime = 实际运行时间*1024 / 进程权重

因此，进程按照各自不同的速率在物理时钟节拍内前进，优先级高则权重大，其虚拟时钟比真实时钟跑得慢，但获得比较多的运行时间；反之，优先级低则权重小，其虚拟时钟比真实时钟跑得快，反而获得比较少的运行时间。CFS调度器总是选择虚拟时钟跑得慢的进程来运行，从而让每个调度实体（sche_entity）的虚拟运行时间互相追赶，进而实现进程调度上的平衡。

CFS调度器没有将进程维护在运行队列中，而是维护了一个以虚拟运行时间为顺序的红黑树。红黑树的主要特点有：

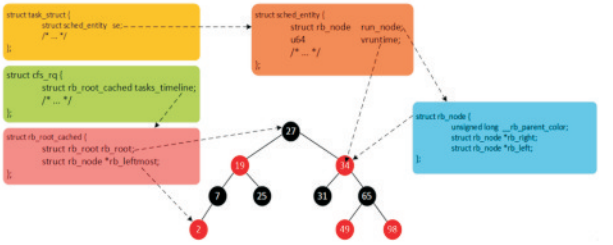
1. 自平衡，树上没有一条路径会比其他路径长出俩倍。

2. O(log n) 时间复杂度，能够在树上进行快速高效地插入或删除进程。




如图所示，进程存储在以vruntime排序的红黑树中，对处理器需求最多的任务（vruntime最低）存储在树的左侧，处理器需求最少的进程（vruntime最高）存储在树的右侧。为了保证公平性，调度器每次选取红黑树最左端的进程进行调度。

CFS的内部原理大致为如图所示：




Linux内的所有任务都由称为 task_struct 的

任务结构表示，它位于调度的最顶端。该结构（在./linux/include/linux/sched.h）完整地描述了任务并包括了任务的当前状态、其堆栈、进程标识、优先级（静态和动态）等等。

```
struct task_struct{
    ...
    volatile long state;
    void *stack;
    unsigned int flags;
    int prio;
    int static_prio;
    int normal_prio;
    struct sche_entity se;
    ...
};
```

但是，由于不是所有任务都是可运行的，所以在task_struct中不会发现任何与CFS相关的字段。因此，需要通过一个名为 sched_entity 的新结构来跟踪调度信息。

```
struct sched_entity{
    ...
    struct load_weight load;
    struct rb_node run_node;
    struct list_head group_node;
    u64 vruntime;
    ...
};
```

sched_entity包含负载权重、各种统计数据以及vruntime（任务运行的虚拟时间量，并作为红黑树的索引）。同时，sched_entity还包含红黑树的节点rb_node。

```
struct rb_node{
    unsigned long __rb_parent_color;
    struct rb_node *rb_right;
    struct rb_node *rb_left;
};
```

红黑树的每个节点都由 rb_node 表示，它只包含子引用和父对象的颜色。红黑树的叶子不包含信息，但是内部节点代表一个或多个可运行的任务。红黑树的根通过rb_root_cached结构中的rb_root引用，而该结构同时包含了红黑树的最左节点rb_leftmost的指针。

```
struct rb_root_cached{
    struct rb_root rb_root;
    struct rb_node *rb_leftmost;
};
```

在运行过程中，__schedule()（在./kernel/sched/core.c中）是CFS调度器的核心函数，其作用是让调度器选择和切换到一个合适的进程运行。

在时钟周期开始时，调度器调用__schedule()函数来开始调度的运行。

然后，__schedule()函数调用pick_next_task()让进程调度器从就绪队列中选择一个最合适的进程next，即红黑树最左边的节点。

接着，通过context_switch()切换到新的地址空间，从而保证next进程运行。

在时钟周期结束时，调度器调用entity_tick()函数来更新进程负载、进程状态以及vruntime（当前vruntime + 该时钟周期内运行的时间）。

最后，将该进程的虚拟时间与就绪队列红黑树中最左边的调度实体的虚拟时间做比较，如果小于坐左边的时间，则不用触发调度，继续调度当前调度实体。否则，则表明最左边的调度实体更需要调度。因此，调度器将当前调度实体放回红黑树，并选择红黑树中最左边的调度实体作为next在下一个时钟周期进行调度。

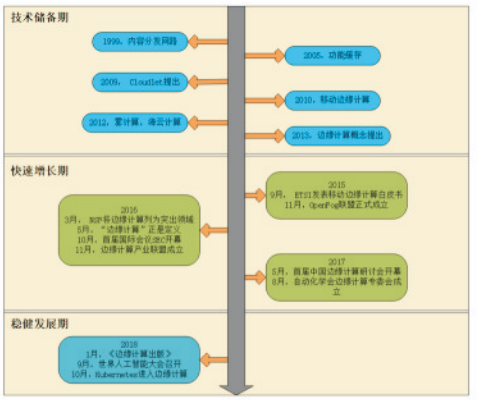
通过以上的结构和调度方式，Linux内核保证了操作系统中进程调度的公平性。

# 浅析边缘计算

（邹俊峰 https://user.qzone.qq.com/441919123/blog/1551168751）

近年来，随着万物互联时代的快速到来和无线网络的普及，网络边缘的设备数量和产生的数据都快速增长。根据 IDC 预测，到2020 年全球数据总量将大于40泽字节（zettabyte，ZB），而物联网产生数据的45%都将在网络边缘处理。在这种情形下，以云计算模型为核心的集中式处理模式将无法高效处理边缘设备产生的数据。集中式处理模型将所有数据通过网络传输到云计算中心，利用云计算中心超强的计算能力来集中式解决计算和存储问题，这使得云服务能够创造出较高的经济效益。但是在万物互联的背景下，传统云计算有几个不足：实时性不够；带宽不足；能耗较大；不利于数据安全和隐私。为了解决以上问题，面向边缘设备所产生海量数据计算的边缘计算模型应运而生。

边缘计算是在网络边缘执行计算的一种新型计算模型，边缘计算操作的对象包括来自于云服务的下行数据和来自于万物互联服务的上行数据，而边缘计算的边缘是指从数据源到云计算中心路径之间的任意计算和网络资源，是一个连续的系统。边缘计算模型和云计算模型并不是取代的关系，而是相辅相成的关系，边缘计算需要云计算中心强大的计算能力和海量存储的支持，而云计算中心也需要边缘计算中边缘设备对海量数据及隐私边缘计算模型具有几个明显的优点：首先，在网络边缘处理大量临时数据，不再全部上传云端，这极大的减轻了网络带宽和数据中心功耗的压力。其次，在靠近数据生产者处做数据处理，不需要通过网络请求云计算中心的响应，大大减少了系统延迟，增强了服务响应能力。最后边缘计算将用户隐私数据不再上传，而是存储在网络边缘设备上，减少了网络数据泄露的风险，保护了用户数据安全和隐私。得益于这些优势，边缘计算近年来得到了迅速发展。总的来说，边缘计算的发展期可分为三个时期：技术储备期、快速增长期和稳健发展期。三个时期中的关键事件及其时间表由下图给出。




## 边缘计算核心问题与技术

负载均衡。边缘计算中，边缘设备产生大量的数据，同时边缘服务器提供了大量的服务。因此，根据边缘服务器、以及网络状况，如何动态的对这些数据进行调度至合适的计算服务提供者，将是边缘计算中的核心问题。

隔离技术是支撑边缘计算稳健发展的重要研究技术，边缘设备需要通过有效的隔离技术来保证服务的可靠性和服务质量。隔离技术需要考虑两方面，一是计算资源的隔离，即应用程序间不能相互干扰；二是数据的隔离，即不同应用程序应具有不同的访问权限。在云计算场景下，由于某一应用程序的崩溃可能带来整个系统的不稳定，造成严重的后果，而在边缘计算下，这一情况变得更加复杂。例如在自动驾驶操作系统中，既需要支持车载娱乐满足用户需求，又需要同时运行自动驾驶任务满足汽车本身驾驶需求，此时，如果车载娱乐的任务干扰了自动驾驶任务，或者影响了整个操作系

统的性能，将会引起严重后果，对生命财产安全造成直接损失。隔离技术同时需要考虑第三方程序对用户隐私数据的访问权限问题，例如，车载娱乐程序不应该被允许访问汽车控制总线数据等。目前在云计算场景下主要使用VM 虚拟机和Docker 容器技术等方式保证资源隔离。边缘计算可汲取云计算发展的经验，研究适合边缘计算场景下的隔离技术。在云平台上普遍应用的 Docker技术可以实现应用在基于OS级虚拟化的隔离环境中运行，Docker的存储驱动程序采用容器内分层镜像的结构，使得应用程序可以作为一个容器快速打包和发布，从而保证了应用程序间的隔离性。

全新的体系结构。无论是如高性能计算一类传统的计算场景，还是如边缘计算一类的新兴计算场景，未来的体系结构应该是通用处理器和异构计算硬件并存的模式。异构硬件牺牲了部分通用计算能力，使用专用加速单元减小了某一类或多类负载的执行时间，并且显著提高了性能功耗比。边缘计算平台通常针对某一类特定的计算场景设计，处理的负载类型较为固定，故目前有很多前沿工作针对特定的计算场景设计边缘计算平台的体系结构。

## 边缘计算典型应用

### 智能网联车和自动驾驶

随着机器视觉，深度学习和传感器等技术的发展，汽车的功能不再局限于传统的出行和运输工具，而是逐渐变为一个智能的，互联的计算系统，我们称这样新型的汽车为智能网联车（Connected and Autonomous Vehicles，CAVs）。智能网联车的出现催生出了一系列的新的应用场景，例如自动驾驶，车联网以及智能交通。Intel在2016 年的报告指出，一辆自动驾驶车辆一天产生的数据为4TB，这些数据无法全部上传至云端处理，需要在边缘节点（汽车）中存储和计算。自动驾驶计算场景无疑是目前最热的研究方向之一，围绕此场景有经典的自动驾驶算法评测数据集KITTI，还有针对不同自动驾驶阶段的经典的视觉算法。在工业界目前有许多针对CAVs场景推出的计算平台，例如NVIDIA DRIVE PX2和 Xilinx Zynq UltraScale + ZCU106等。同时，学术界有许多前沿工作也开始探索CAVs场景下的边缘计算平台的系统设计。Liu等将自动驾驶分为传感（Sensing），感知（Perception）和决策（Decision-making）三个处理阶段，并比较三个阶段在不同异构硬件上的执行效果，由此总结除了自动驾驶任务与执行硬件之间的匹配规则。Lin等对比了感知阶段三个核心应用，定位（Localization），识别（Object Detection）和追踪（Object Tracking）在 GPUs，FPGAs和ASICs不同组合运行的时延和功耗，指导研究人员设计端到端的自动驾驶计算平台。除了硬件系统结构设计，还有一类研究推出完整的软件栈帮助研究人员实现自动驾驶系统，例如百度的Apollo和日本早稻田大学的Autoware。

### 智能家居

随着物联网技术的发展，智能家居系统得到进一步的发展，其利用大量的物联网设备（如温湿度传感器、安防系统、照明系统）实时监控家庭内部状态，接受外部控制命令并最终完成对家居环境的调控，以提升家居安全性、便利性、舒适性。Berg Insight 的调查报告显示，欧美和北美洲的智能家居数据将在2019年达到 6800万。然而，随着智能家居设备的越来越多，且这些设备通常都是异构的，如何管理这些异构设备将会是一个亟待解决的问题，如设备的命名、数据的命名以及设备的智能化联动。并且，由于家庭数据的隐私性，用户并不总是愿意将数据上传至云端进行处理，尤其是一些家庭内部视频数据。而边缘计算可以将计算（家庭数据处理）推送至家庭内部网关，减少家庭数据的外流，从而降低数据外泄的可能性，提升系统的隐私性。工业界的一些企业已经注意到这一点，例如亚马逊的Echo、三星的SmartThings和谷歌

# GAN–生成式对抗网络

（任　毅　整理）

神经网络是一种函数的逼近。在数学中我们可以把函数当作是一种机器，当你向函数中传递一些参数的时候，函数会给你相应的生成一个或一组数字。

基于以上的想法和启发，如果我们能够在生活中找到这样的函数，那么我们就会得到想要的结果。但是，目前大家往往不能或尚未想明白如何将生活中发生的一些事件或者现象用一系列的加减乘除的公式写出来，这个时候，人们就运用到了逼近的思想，用海量的参数的神经网络来逼近和拟合我们想要的数学函数，得到我们想要的结果。同时，万能逼近定理表明，一个具有足够多参数，足够多的神经元的网络是可以实现各种各样的函数拟合的。

利用神经网络这个工具以后，我们就可以搭建起足够巨大的网络结构，调整各个参数来完成函数的逼近，而不用显式的去表达这个函数公式。

此时，我们只是拟合出了这个函数的输入和输出，但是我们并不知道函数本身的精确数学表达式，因此也不知道我们拟合的效果到底有多好或者多差，这个时候就需要损失函数来做出判断，gan就是一种无需显式的数学表达式，同时又能逼近神经网络损失函数的一个方法。

GAN由判别器D和生成器G组成，其D和G自身就是另一种神经网络。它们不是显式损失函数，而只是神经网络表示的近似。判别器的任务是区分真实数据分布与生成数据分布。

用监督方式训练判别器时，标签可随意使用，所以采用二元交叉熵等显式损失函数训练判别器就很简单。但由于判别器是生成器的损失函数，这代表判别器的二元交叉熵损失函数的累积梯度同样会被用于更新生成器网络。

通过传统损失函数与神经网络的集成，GAN使将神经网络作为损失函数来训练另一神经网络成为可能。两个神经网络间的巧妙交互使得深度神经网络能够解决一些先前无法完成的任务（如生成逼真图像）。将 GAN 本质上视为一种学得的损失函数，发现GAN的巨大的简洁和力量。

接上页

的GoogleHome， 均可作为智能家居的控制中心。然而，这些设备，仍然需要一些额外的网络服务，如各种识别服务，不能完全依靠自身去处理数据，从而导致仍存在一定的隐私泄露隐 患。微软和苹果分别提出了HomeOS和HomeKit，其作为智能家居的框架，可以方便用户对设备进行控制，但是仍然缺少一些具体的工作。

## 展望

边缘计算经过近几年的技术储备，已经得到了来自国内外政府、学术界和工业界的广泛重视和一致认可，现在到了全面开花结果并带来经济效益的时候。得益于网络、隔离技术、体系结构、操作系统、算法执行框架、数据处理平台以及安全隐私这七个关键技术的快速发展，边缘计算技术已经走向成熟，并在许多应用场景下发挥作用。发现边缘计算目前已经在公共安全中的实时数据处理和自动驾驶中取得了成功的经验，并迅速扩展到虚拟现实、智能家居等场景，在未来几年我们将会看到边缘计算在工业物联网和智慧城市等领域取得更多成功的例子，为产业升级提供助力。边缘计算自正式提出以来，不过短短几年，就已经取得了爆发性的增长，按照这个趋势继续进行下去，边缘计算将产生更大的外溢效果，成为各行各业的粘合剂和智能产业发展的催化剂，促进整个工业体系的升级转型。

# moby && cli源码分析

（陈　旭 https://sabertazimi.github.io/cgcl-blog/ ）

Moby由一下部分组成：

1. 一个容器化的后台组件库（比如，一个低层次的构建器、日志工具、数据卷管理、网络、镜像管理、containerd、SwarmKit……）

2. 一个用于将这些组件组装成一个独立容器系统，并协助构建、测试和部署这些组装产物的框架。

3. 一个名为Moby Origin的参考组装品，它是Docker容器系统的开放基础，同时也是使用来自Moby库或其它项目的不同组件的容器系统示例。

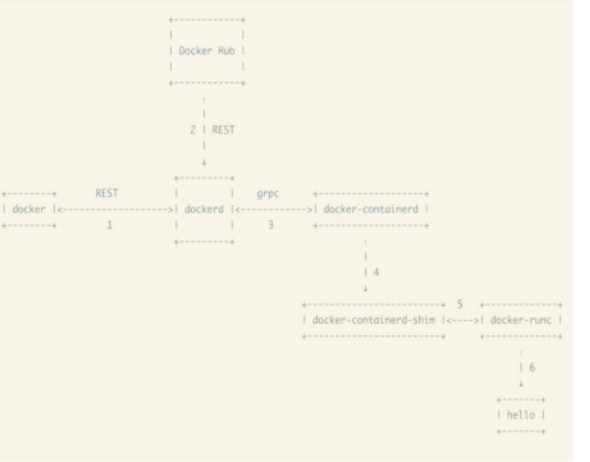
Moby用于组装专门的容器系统，无需重新设计轮子。它提供了十几个标准组件，并将它们组装成定制平台的框架

学习dockerd支持的参数，以及这些参数背后的功能是如何实现的。重点关注网络这部分，参考自己动手写docker这本书。

先弄清楚初始化过程中到底起了哪些服务，如果有需要在具体的了解这些功能是如何实现的，这些实现的代码在什么地方。重点分析其中的部分代码。

## 容器的创建

在使用docker run命令进行容器创建时，会经历一下步骤，moby项目中的dockerd（dockerd进程就是docker的服务器端）起到为docker命令行服务的作用，docker和dockerd之间使用REST方式进行通信。




## docker client 过程1---> dockerd

在docker cli项目的$GOPATH/github.com/docker/cli/cli/command/container/run.go文件中，是docker run命令的实现文件。当输入docker run hello-world后其中的函数runRun(dockerCli, cmd.Flags(), &opts, copts)会进行容器的下载创建运行工作，参数opts是一些运行时的选项；copts是创建容器的一些配置参数。

runRun()函数完成了以下工作：

1. 将配置文件中的环境参数转化为键值对形式

2. 使用转换后的环境参数创建新的ListOpts，供创建容器的时候使用。

3. 将传入的hello-world，创建容器的参数，docker client所在平台的相关信息存储在containerConfig中，也就是即将需要启动容器的配置信息。containerConfig中包含了Config和

HostConfig和NetworkingConfig。

4. 一些验证工作，确保上述三个步骤的正确运行

Config结构体包含关于容器的配置数据。它应该只保存关于容器的便携式信息。这里，“便携式”意味着“独立于我们正在运行的主机”。HostConfig中应显示非便携信息。添加到此结构的所有字段都必须标记为临时，以保持从旧的V1兼容性配置中获得可预测的哈希值。

HostConfig结构体包含关于容器不可移植配置结构。这里，“非便携式”意味着“依赖于我们正在运行的主机”。便携式信息应该出现在Config中。

NetworkingConfig表示容器的每个接口的网络配置承载在docker run和docker network connect命令中指定的网络配置.

当上述四个工作完成后，调用runContainer (dockerCli, ropts, copts, containerConfig)，这个函数主要做了一下工作：

1. 根据给定的参数进行相关的配置开启

2. 创建ctx, cancelFun := context.WithCancel (context.Background()) 是管理整个处理流程的信息传递链路

3. 调用createResponse, err := createContainer (ctx, dockerCli, containerConfig, &opts.createOptions)，此函数完成一下工作：

>根据配置文件中的hostConfig.ContainerIDFile信息创建容器目录。我们在日常创建容器时，经常会出现下列错误提示Container ID file found, make sure the other container isn't running or delete就是来源于这里。

>给镜像打上tag，trusted标签。

>response, err=dockerCli.Client().ContainerCreate()通过docker client向docker daemon发送创建容器所需的相关信息，如果没有错误发生并且本地没有找到相关镜像，调用pullImage()函数，在这个函数中client发出pull镜像请求。

>其具体实现在$GOPATH/github.com/docker/cli/vendor/github.com/docker/docker/client/container_create.go文件中，通过客户端构造了一个post请求发送到moby项目中的路由中，cli.post(ctx, "/containers/create", query, body, nil)就是其具体发送实现。在moby项目$GOPATH/github.com/docker/docker/api/server/router/container/container.go文件中，定义了一系列可能接受到的路由，其中容器创建路由为router.NewPostRoute ("/containers/ create", r.post ContainersCreate)正好和客户端发出的请求想对应，这是下载镜像的任务就交给dockerd来进行。具体的分析在后面。

4. client.ContainerStart启动容器，此函数发送一个请求给docker daemon请求启动容器，请求指令为cli.post(ctx, "/containers/"+containerID +"/start", query, nil, nil)

5. 判断启动容器是是否开启了tty模式，进行相应的设置。

6. 如果有错误，返回错误代码return cli. StatusError{StatusCode: status}

>如果容器启动失败，出现“找不到”/“没有这样的”错误，则返回127

>如果容器启动失败，出现“拒绝许可”错误，则返回126

>返回125表示守护程序故障

上述这些工作都是docker client所完成的。

# 基于golang的网络爬虫框架

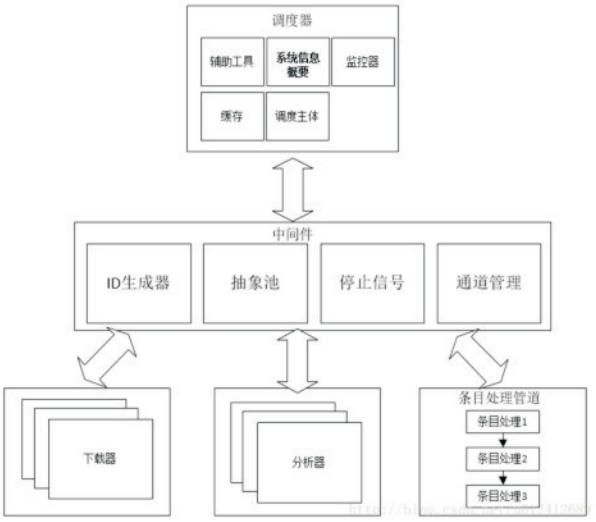
（吴朱天 https://blog.csdn.net/u012412689/article/details/79552702）

## 一、介绍

这是一个用go语言实现的网络爬虫框架，本框架的核心在于可定制和可扩展，用户可以根据自己的需要定制各个模块，同时，也给出了一个实现demo供参考。Go语言的初学者也可以通过这个项目熟悉go语言的各种特性，尤其是并发编程。

## 二、项目框架介绍

该网络爬虫主要由五个部分组成：调度器、中间件、下载器、分析器、条目处理器。下面分别介绍每部分的功能。




**调度器：**对整个爬虫框架进行调度，所有消息都要经过调度器，因此同时肩负着缓存和产生系统信息概要的功能。主要包括辅助工具、系统信息概要、缓存、调度主体四个部分；

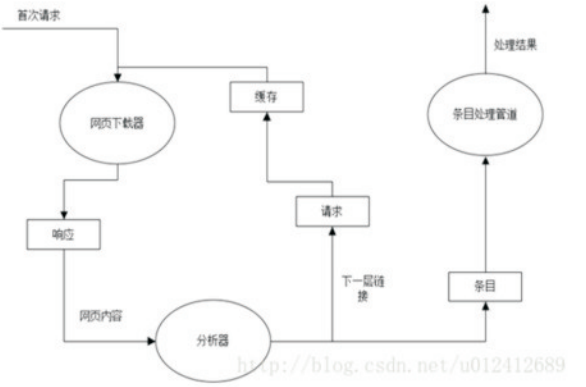
**中间件：**承上启下，辅助调度器调度整个系统，主要包括ID生成器、抽象池、停止信号、通道管理；

**下载器：**接受输入请求，转化为http请求，下载相应的网页，供分析器分析。设计为多线程，可以同时下载多个页面，并且通过实现一个抽象池来管理线程数量；

**分析器：**接受作为输入的相应，并且还原为HTTP相应，同时分析生成网页中新的HTTP请求（网页中链接）以及要进一步分析的条目。设计为多线程，可以同时处理多个相应，并且通过实现一个抽象池来管理线程数量；

**条目处理管道：**包括多个条目处理器，接受分析器传递的条目，最后给出处理结果。可以根据自己的需要定制分析方式，后文中将给出一个条目处理管道demo。

## 三、数据流




输入首次请求的网站之后，下载响应网站形成响应数据传送给分析器，分析器分离其中的链接和要处理的条目，链接形成下一

层请求给调度器缓存起来，条目发送给条目处理管道进一步处理，最后给出最终处理结果。

## 四、各模块接口设计

### 1、调度器

（1）调度器主体

主要用于启动和停止整个系统，并且从中获取一些系统运行的状态。

```
type Scheduler interface {
    // 开启调度器。
    // 调用该方法会使调度器创建和初始化各个组件。在此之后，调度器会激活爬取流程的执行。
    // 参数channelArgs代表通道参数的容器。
    // 参数poolBaseArgs代表池基本参数的容器。
    // 参数crawlDepth代表了需要被爬取的网页的最大深度值。深度大于此值的网页会被忽略。
    // 参数httpClientGenerator代表的是被用来生成HTTP客户端的函数。
    // 参数respParsers的值应为分析器所需的被用来解析HTTP响应的函数的序列。
    // 参数itemProcessors的值应为需要被置入条目处理管道中的条目处理器的序列。
    // 参数firstHttpReq即代表首次请求。调度器会以此为起始点开始执行爬取流程。
    Start (channelArgs base.ChannelArgs, pool BaseArgsbase. PoolBaseArgs,crawlDepth uint32, httpClient Generator Gen HttpClient,respParsers[]anlz.ParseResponse, itemProcessors[] ipl.ProcessItem, firstHttpReq *http.Request) (err error)
    // 调用该方法会停止调度器的运行。所有处理模块执行的流程都会被中止。
    Stop() bool
    // 判断调度器是否正在运行。
    Running() bool
    // 获得错误通道。调度器以及各个处理模块运行过程中出现的所有错误都会被发送到该通道。
    // 若该方法的结果值为nil，则说明错误通道不可用或调度器已被停止。
    ErrorChan() <-chan error
    // 判断所有处理模块是否都处于空闲状态。
    Idle() bool
    // 获取摘要信息。
    Summary(prefix string) SchedSummary
}
```

（2）系统信息摘要

通过此接口获取系统的整体情况

```
type SchedSummary interface {
    String() string // 获得摘要信息的一般表示。
    Detail() string // 获取摘要信息的详细表示。
    Same(other SchedSummary) bool // 判断是否与另一份摘要信息相同。
}
```

（3）缓存

一般来说一个页面会有多个链接，这样会导致深一层的请求远多于下载器的下载线程，所以需要先把一部分请求缓存起来，等到下载线程空闲再发送出去。

```
type requestCache interface {
    // 将请求放入请求缓存。
    put(req *base.Request) bool
    // 从请求缓存获取最早被放入且仍在其中的请求。
    get() *base.Request
    // 获得请求缓存的容量。
    capacity() int
    // 获得请求缓存的实时长度，即：其中的请求的即时数量。
    length() int
    // 关闭请求缓存。
    close()
    // 获取请求缓存的摘要信息。
    summary() string
}
```

（4）监控器

监控整个系统的运行，在所有线程都结束之后及时返回信号，以便于调度器主体终止整个系统。

```
    // scheduler代表作为监控目标的调度器。
    // intervalNs代表检查间隔时间，单位：纳秒。
```

// maxIdleCount代表最大空闲计数。

// autoStop被用来指示该方法是否在调度器空闲一段时间（即持续空闲时间，由intervalNs * maxIdleCount得出）之后自行停止调度器。

// detailSummary被用来表示是否需要详细的摘要信息。

// record代表日志记录函数。

func Monitoring(scheduler sched.Scheduler, intervalNs time.Duration, maxIdleCount uint, autoStop bool, detailSummary bool, record Record) <-chan uint64

**2、中间件**

（1）ID生成器

生成一个系统中唯一的ID，保证下载器池和分析器池的唯一性。

```
type IdGenerator interface {
    GetUint32() uint32 // 获得一个uint32类型的ID。
}
```

（2）抽象池

从下载器池和分析器池中抽象出来的票池，方便控制goroutine的数量。其中，Entity可以理解为上一个接口中的唯一ID实体。

```
type Entity interface {
    Id() uint32
}

type Pool interface {
    Take() (Entity, error)
    Return(entity Entity) error
    Total() uint32
    Used() uint32
}
```

（3）停止信号

由于本框架是多线程的，所以需要一个信号来关闭整个系统。

```
type StopSign interface {
    // 置位停止信号。相当于发出停止信号。
    // 如果先前已发出过停止信号，那么该方法会返回false。
    Sign() bool
    // 判断停止信号是否已被发出。
    Signed() bool
    // 重置停止信号。相当于收回停止信号，并清除所有的停止信号处理记录。
    Reset()
    // 处理停止信号。
    // 参数code应该代表停止信号处理方的代号。该代号会出现在停止信号的处理记录中。
    Deal(code string)
    // 获取某一个停止信号处理方的处理计数。该处理计数会从相应的停止信号处理记录中获得。
    DealCount(code string) uint32
    // 获取停止信号被处理的总计数。
    DealTotal() uint32
    // 获取摘要信息。其中应该包含所有的停止信号处理记录。
    Summary() string
}
```

（4）通道管理

由于本系统是多线程的，所以将通道管理放在一起方便管理，分别管理请求通道、响应通道、条目处理通道、错误通道。

```
type ChannelManager interface {
    // 初始化通道管理器。
    // 参数channelArgs代表通道参数的容器。
    // 参数reset指明是否重新初始化通道管理器。
    Init(channelArgs base.ChannelArgs, reset bool) bool
    // 关闭通道管理器。
    Close() bool
    // 获取请求传输通道。
    ReqChan() (chan base.Request, error)
    // 获取响应传输通道。
    RespChan() (chan base.Response, error)
    // 获取条目传输通道。
    ItemChan() (chan base.Item, error)
    // 获取错误传输通道。
    ErrorChan() (chan error, error)
    // 获取通道管理器的状态。
    Status() ChannelManagerStatus
    // 获取摘要信息。
```

```
    Summary() string
}
```

**3、下载器**

用于根据请求下载响应的网页。

```
type PageDownloader interface {
    Id() uint32 // 获得ID。
    Download(req base.Request) (*base.Response, error)
// 根据请求下载网页并返回响应。
}
```

**4、分析器**

获取下载器下载的内容，进一步分析，剥离出下一层请求和需要处理的条目。

```
type Analyzer interface {
    Id() uint32 // 获得ID。
    Analyze(respParsers []ParseResponse, resp base.Response)
([]base.Data, []error) // 根据规则分析响应并返回请求和
条目。
}
```

**5、条目处理管道**

分别处理每一个条目，并且通过管道发送错误信息。

```
type ItemPipeline interface {
    // 发送条目。
    Send(item base.Item) []error
    // FailFast方法会返回一个布尔值。该值表示当前
的条目处理管道是否是快速失败的。
    // 这里的快速失败是指：只要对某个条目的处理
流程在某一个步骤上出错，
    // 那么条目处理管道就会忽略掉后续的所有处理
步骤并报告错误。
    FailFast() bool
    // 设置是否快速失败。
    SetFailFast(failFast bool)
    // 获得已发送、已接受和已处理的条目的计数值。
    // 更确切地说，作为结果值的切片总会有三个元
素值。这三个值会分别代表前述的三个计数。
    Count() []uint64
    // 获取正在被处理的条目的数量。
    ProcessingNumber() uint64
    // 获取摘要信息。
    Summary() string
}
```

**6、其他基础结构**

```
type Data interface {
    Valid() bool // 数据是否有效。
}

// 请求。
type Request struct {
    httpReq *http.Request // HTTP请求的指针值。
    depth uint32 // 请求的深度。
}

// 响应。
type Response struct {
    httpResp *http.Response
    depth uint32 //响应的深度
}
```

## 五、具体实现

各接口的具体实现见我的github: https://github.com/hustfoam/WebCrawler

## 六、定制demo示例

主要需要实现响应解析函数以及条目处理器函数，其中条目处理函数可以是一系列处理方式，具体见我的github：https:// github. com / hustfoam / WebCrawler / blob / master / demo / demo.go

## 七、扩展

本框架的初级版可以定制响应解析函数和条目处理函数，根据自己的需求实现。如果有需要的话，下载代码之后更改demo中的ParseResponse、ProcessItem 函数即可定制。更进一步的，可以跟据自己的需求实现上述框架中的接口，改变数据处理方式。

## 参考文献

《go语言编程实战》

https://github.com/hustfoam/WebCrawler

# Flink流计算引擎

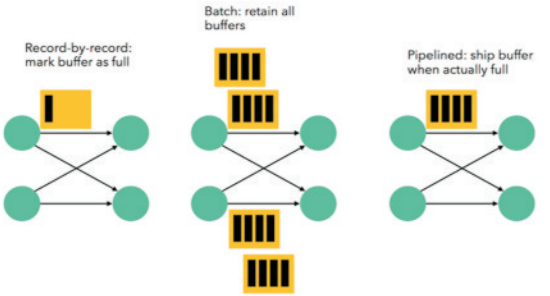
（周文翔 https://blog.csdn.net/Joker_Chou/article/details/87802254）

伴随着海量增长的数据，数字化时代的未来感扑面而至。不论是结绳记事的小数据时代，还是我们正在经历的大数据时代，计算的边界正在被无限拓宽，而数据的价值再也难以被计算。时下，谈及大数据，不得不提到热门的下一代大数据计算引擎Apache Flink（以下简称Flink）。

## 统一的批处理与流处理系统

在大数据处理领域，批处理任务与流处理任务一般被认为是两种不同的任务。一个大数据项目一般会被设计为只能处理其中一种任务，例如Apache Storm、Apache Smaza只支持流处理任务，而Aapche MapReduce、Apache Tez、Apache Spark只支持批处理任务。Spark Streaming是Apache Spark之上支持流处理任务的子系统，看似一个特例，实则不然——Spark Streaming采用了一种micro-batch的架构，即把输入的数据流切分成细粒度的batch，并为每一个batch数据提交一个批处理的Spark任务。所以Spark Streaming本质上还是基于Spark批处理系统对流式数据进行处理，和Apache Storm、Apache Smaza等完全流式的数据处理方式完全不同。目前同时支持流处理和批处理的计算引擎，只有两种选择：一个是Apache Spark，一个是Apache Flink。通过其灵活的执行引擎，Flink能够同时支持批处理任务与流处理任务。

在执行引擎这一层，流处理系统与批处理系统最大不同在于节点间的数据传输方式。对于一个流处理系统，其节点间数据传输的标准模型是：当一条数据被处理完成后，序列化到缓存中，然后立刻通过网络传输到下一个节点，由下一个节点继续处理。而对于一个批处理系统，其节点间数据传输的标准模型是：当一条数据被处理完成后，序列化到缓存中；并不会立刻通过网络传输到下一个节点；当缓存写满，就持久化到本地硬盘上；当所有数据都被处理完成后，才开始将处理后的数据通过网络传输到下一个节点。这两种数据传输模式是两个极端，对应的是流处理系统对低延迟的要求和批处理系统对高吞吐量的要求。Flink的执行引擎采用了一种十分灵活的方式，同时支持了这两种数据传输模型。Flink以固定的缓存块为单位进行网络数据传输，用户可以通过缓存块超时值指定缓存块的传输时机。如果缓存块的超时值为0，则Flink的数据传输方式类似上文所提到流处理系统的标准模型，此时系统可以获得最低的处理延迟。如果缓存块的超时值为无限大，则Flink的数据传输方式类似上文所提到批处理系统的标准模型，此时系统可以获得最高的吞吐量，同时缓存块的超时值也可以设置为0到无限大之间的任意值。缓存块的超时阈值越小，则Flink流处理执行引擎的数据处理延迟越低，但吞吐量也会降低，反之亦然。通过调整缓存块的超时阈值，用户可根据需求灵活地权衡系统延迟和吞吐量。下图为Flink执行引擎数据传输模式。




在统一的流式执行引擎基础上，Flink同时支持了流计算和批处理，并对性能（延迟、吞吐量等）有所保障。相对于其他原生的流处理

与批处理系统，并没有因为统一执行引擎而受到影响，从而大幅度减轻了用户安装、部署、监控、维护等带来的成本。

## Flink的技术栈

Flink核心是一个流式的数据流执行引擎，其针对数据流的分布式计算提供了数据分布、数据通信以及容错机制等功能。基于流执行引擎，Flink提供了诸多更高抽象层的API以便用户编写分布式任务：

1. DataSet API，对静态数据进行批处理操作，将静态数据抽象成分布式的数据集，用户可以方便地使用Flink提供的各种操作符对分布式数据集进行处理，支持Java、Scala和Python。

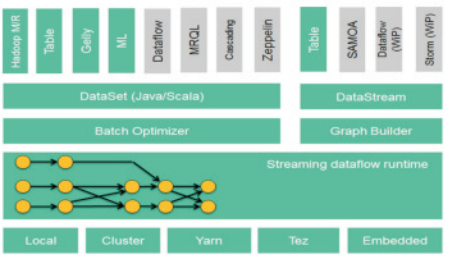
2. DataStream API，对数据流进行流处理操作，将流式的数据抽象成分布式的数据流，用户可以方便地对分布式数据流进行各种操作，支持Java和Scala。

3. Table API，对结构化数据进行查询操作，将结构化数据抽象成关系表，并通过类SQL的DSL对关系表进行各种查询操作，支持Java和Scala。

此外，Flink还针对特定的应用领域提供了领域库，例如Flink ML，Flink的机器学习库，提供了机器学习Pipelines API并实现了多种机器学习算法；还有Gelly，Flink的图计算库，提供了图计算的相关API及多种图计算算法实现。

另外，Flink也可以方便地和Hadoop生态圈中其他项目集成，例如Flink可以读取存储在HDFS或HBase中的静态数据，以Kafka作为流式的数据源，直接重用MapReduce或Storm代码，或是通过YARN申请集群资源等。

Flink的技术栈如下所示：




## Flink的状态管理

Flink最区别于其他流计算引擎的，其实就是状态管理。什么是状态？例如开发一套流计算的系统或者任务做数据处理，可能经常要对数据进行统计，如Sum,Count,Min,Max,这些值是需要存储的。因为要不断更新，这些值或者变量就可以理解为一种状态。如果数据源是在读取Kafka, RocketMQ，可能要记录读取到什么位置，并记录Offset，这些Offset变量都是要计算的状态。

Flink提供了内置的状态管理，可以把这些状态存储在Flink内部，而不需要把它存储在外部系统。这样做的好处是第一降低了计算引擎对外部系统的依赖以及部署，使运维更加简单；第二，对性能带来了极大的提升：如果通过外部去访问，如Redis, HBase它一定是通过网络及RPC。如果通过Flink内部去访问，它只通过自身的进程去访问这些变量。同时Flink会定期将这些状态做Checkpoint持久化，把Checkpoint存储到一个分布式的持久化系统中，比如HDFS。这样的话，当Flink的任务出现任何故障时，它都会从最近的一次Checkpoint 将整个流的状态进行恢复，然后继续运行它的流处理。对用户没有任何数据上的影响。

那么Flink是如何做到在Checkpoint恢复过程中没有任何数据的丢失和数据的冗余？来保证精准计算的？这其中原因是Flink利用了一套非常经典的Chandy-Lamport算法，它的核心思想是把这个流计算看成一个流式的拓扑，定期从这个拓扑的头部Source点开始插入特殊的Barries，从上游开始不断的向下游广播这个Barries。每一个节点收到所有的Barries,会将State 做一次Snapshot，当每个节点都做完Snapshot之后，整个拓扑就算完整的做完了一次Checkpoint。接下来不管出现任何故障，都会从最近的Checkpoint进行恢复。Flink利用这套经典的算法，保证了强一致性的语义。这也是Flink与其他无状态流计算引擎的核心区别。

# 实验室举行 2019年新春联欢晚会

刘　芹

吉犬腾欢辞旧岁，金猪报喜迎新春。2月24日晚，2019新春联欢晚会隆重举行。实验室师生近三百人欢聚一堂，辞旧迎新，喜迎新春。

实验室主任金海教授为本场晚会致新春贺词，对实验室全体师生在2018年做出的努力与贡献表示由衷的感谢，祝福即将毕业的2016级硕士前程似锦、未来可期，同时希望全体师生在2019年继续保持求真务实、团结协作的理念，再创辉煌。

联欢晚会在欢快喜庆的开场舞《舞蹈串烧》中拉开帷幕。大合唱《东五群侠传》以振奋人心的音符、铿锵有力的节奏展示了实验室学生的团队协作与开拓进取精神。《荧光舞》以酷炫的荧光与律动的舞蹈，展现了同学们的科技创意与青春活力。诗朗诵《2018你过的好吗？》以轻松诙谐的语言诠释了同学们的乐观坚强，给大家留下了深刻的印象。《影子戏》、《钢琴独奏》分别让大家领略到中国传统民间艺术与西方音乐文化的魅力。《歌曲串烧》、《草原上升起的太阳》、《易燃易爆炸》、《龙的传人》以优美动听的歌声，赢得了现场观众的阵阵掌声。小品《剁椒鱼头》、《人生自古谁无死》以幽默夸张的表演，令人捧腹大笑。

晚会节目丰富多样，精彩纷呈，其间穿插游戏、抽奖、最佳论文颁奖等，十大新闻评选颁奖更是掀起一波又一波的高潮。2018年度实验室十大新闻评选：吴松获得一等奖，刘方明和陈意诚获得二等奖，刘海坤等7名老师同学获得三等奖。

最后，联欢晚会在《至少还有你》的精彩合唱中圆满落下帷幕。

晚会结束后，经实验室全体师生网络投票，郭凤琳获得最佳节目主持人奖，武静、王兰心、刘聪分别获得最佳女演员一、二、三等奖，陈聃、张照博、聂廷宝分别获得最佳男演员一、二、三等奖，东五群侠传、荧光舞、影子戏分别获得最佳节目一、二、三等奖。

2019年，实验室将继续秉承“明德厚学、求是创新”的华中大精神，坚持“育才明德、开放竞争”的原则，不断开拓进取。

通讯员：刘　芹

刘　芹
主要负责实验室宣传等办公室事务
Email：liuqin19@hust.edu.cn

# 2018年度实验室十大新闻揭晓

刘　芹

2019年2月24日晚，在一年一度的实验室新春联欢晚会上，2018年度实验室十大新闻揭晓，“实验室获得2018年度教育部自然科学一等奖，这是实验室第一次获得教育部自然科学奖”等十条新闻入选：

1、实验室获得2018年度教育部自然科学一等奖，这是实验室第一次获得教育部自然科学奖。

2、实验室主任金海教授当选2019年度IEEE Fellow，他是我校在IEEE Computer Society中当选的第一位Fellow。

3、廖小飞教授获得国家自然科学基金杰出青年基金项目、入选中青年科技创新领军人才。

4、邹德清教授获中央网信办颁发的2018国家网络安全优秀教师奖。

5、廖小飞教授荣升外事处处长、吴松教授荣升计算机学院副院长、邹德清教授兼任计算机科学与技术学院、网络空间安全学院副院长。

6、实验室牵头的首项国家重点研发计划项目“面向图计算的通用计算机技术与系统”正式获批并启动。

7、实验室金海教授主持的国家自然科学基金重点项目“面向大数据高效处理的体系结构和关键技术”获批。

8、实验室获批成立大数据技术与系统国家地方联合工程研究中心，填补实验室国家级科研平台建设的空白。

9、实验室2018年到账科研经费突破2000万，创历史新高。

10、2018年实验室出版专著四本，具体包括：金海教授、石宣化教授共同撰写的《大数据处理》；金海教授、徐鹏副教授、邹德清教授共同编写的《中国网络空间安全前沿科技发展报告2018》；邹德清教授、代炜琦博士、金海教授共同编写的《云服务安全》；徐鹏副教授、金海教授共同编写的《云数据安全》。

通讯员：刘　芹

刘　芹
主要负责实验室宣传等办公室事务
Email：liuqin19@hust.edu.cn

# ACES: Automatic Compartments for Embedded System

李晨晨　推荐

ACES: Automatic Compartments for Embedded Systems是2018年USENIX会议上录取的一篇文章，该会议于2018年8月在美国举办。

随着移动设备的普及和大数据时代的飞速发展，物联网（IoT）也成为生活中必不可少的一部分。物联网的扩散给数据连接提升了一个新的高度。这种模式具有很大的潜力。但是同时又使得我们的生活暴露在更多的网络攻击之中。EPOXY将DEP，多样性和堆栈保护的策略能够被用在裸机系统中，但是该方法没有解决特权分离和进程隔离的问题。因此攻击者可以通过申请另一个数据的权限进而获得使用关键数据的权限。MINION使用例如编译器和动态分析来推断线程级别的分区，同时根据操作系统上下文来切换或者更改隔间。因为该方法使用的是固定的方法来确定隔间，开发者无法自由的确定安全边界，同时也存在着不同的系统的异构问题。ARM的Mbed uVisor方法是一个区域性的ARM Cortex-M系列设备的平台，它通过创建隔离专区的逻辑应用程序限制对数据的访问以及代码子集的外围设备。但是，它的不足之处在于需要开发者手动分区数据和管理所有隔间之间的通信，且隔间都是限制为单个线程的。因此只能实现粗粒度的隔间分配。本文提出了一种新的方法----ACES。这种方法通过创建隔间的方式，将最小特权原则应用于裸机系统中，同时通过静态分析的方法自动创建隔间，使得每个隔间都与外设或者数据相关联。ACES提供了写和程序区域之间的控制流完整性，也让开发人员能够通过分区策略在性能和安全性之间进行权衡和取舍，让设计人员拥有更高的自由度。省去了重构的问题

所谓隔间，就是一组能够同时访问和获取的内存区域以及他们之间被授权的数据流，其被定义为隔离的代码区域以及其可访问的数据，外围设备和允许的控制流转移。每条指令都属于一个隔间，而数据和外围设备可能从多个隔间访问。因此，ACES的隔间是以代码为中心的，而不是以线程为中心的。如图1所示显示了几个隔间，从图中可以看出，隔间A能够允许访问代码区域X并读取和写外设1，数据区域1和3.而从b可以看出，隔间A也可以从Foo过度调用Baz进入隔间C。除此之外，其他所有的隔间过渡都是不被允许的 。本文使用程序依赖性图和初始区域图来进行隔间的创建。

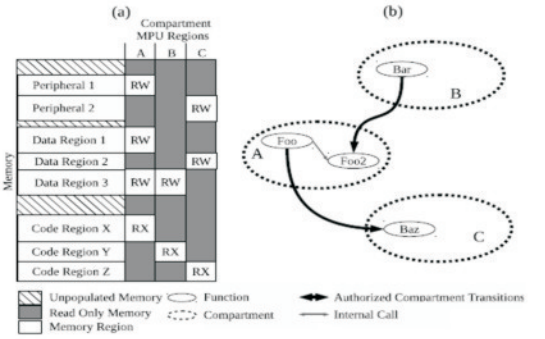



图1 隔间实例

隔间的创建是通过程序依赖性图（PDG）和初始区域图来进行的。程序依赖性图捕获所

有的控制流，全局数据和外围设备的依赖关系，虽然根据研究表明，精确的程序依赖性图的创建时不可行的，但是本文通过静态分析的方法根据代码之间的关系建立初步的PDG图。但是程序的关系性依赖图有时候并不能满足资源限制和硬件的约束，例如，根据PDG图产生隔间的话，数据区域可能会比MPU数量更多，且由每个区域引起的区域或者性能的开销过高，这不符合系统的硬件约束。因此，本文继续分析，将PDG图进一步简化至初始区域图。初始区域图通过对PDG图中的相同类型的顶点进行合并来进行简化。通过结合代码定点来合并包含的函数和相关边。合并代码定点可能会将数据或者外设暴露给合并函数，但是这样做可以提高性能，因为更多的功能位于同一隔间，使得系统进行隔间转换的开销减少。数据定点的合并采用联合其包含的全局变量来进行合并。合并两个数据定点增加了可以使用的代码量。但是外设定点的合并与前两者不同，外设的地址是固定在硬件之中的。因此外设是独立存在的，由代码区域独立决定访问哪个外设。因此，外设定点不需要进行合并。根据这些合并原则进行合并。针对代码区域的分析之后，运用PDG已经建立了初始区域图。但是隔离策略只定义了代码，全局变量和外围设备应该分成几部分，但是没有进行安全性的划分和优化。例如，若A, B数据均在隔间A内，那么攻击者就能够轻松的通过访问数据A来达到访问数据B的权限，这是不被允许的。因此需要对初始区域图进行安全策略的优化，安全策略就是根据访问的内存区域和外部设备，将访问相同外设和内存区域的进行合并，使得系统不用频繁的进行隔间的转化，降低了系统的开销，同时也提升了安全性。最后一步是将得到的优化后的程序区域图进行降维，这是为了确保ACES产生的隔间都能满足MPU的硬件性质。ACES的最后一个要素是堆栈保护MPU保护的系统起始地址和大小，这意味着MPU的保护是粗粒度的，很难精确保护小数据区域和不能轻易搬迁的区域，比如堆栈。为了克服这些限制，本文采用微模拟器，微模拟器通过捕捉故障原因来阻止访问，然后在软件中模拟所有结果写指令，更新存储器，寄存器和处理器标志。实验证明，本文提出的ACES在安全性和性能上都有很大的提高。

为了评估ACES的有效性，本文比较了Na¨ve文件名，优化文件名和外围设备划分政策。分别从运行时间开销和内存区域开销进行分析。运行时间开销如图2所示：

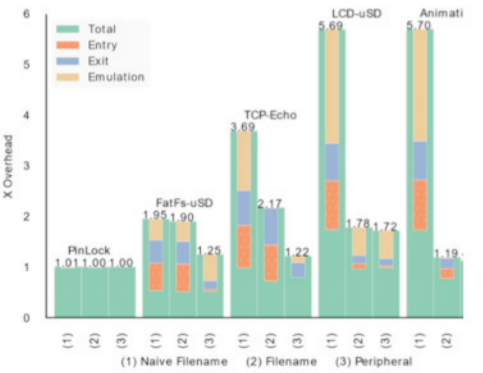



图2 运行时间开销

图中明确的展示了隔间进入，退出和指令仿真三个部分在总时间开销中的所占比例。从图中可以看出，LCD-uSD和Animation总的时间开销是最大的，且在其中占据主要时间的还是仿真模拟产生的时间。同理内存区域的开销也证明了本文提出的方法的可行性。

李晨晨
2018级博士生
研究方向：区块链
Email：D201880868@hust.edu.cn

# Through-Wall Human Pose Estimation Using Radio Signals

李辉楚吴　推荐

本次我推荐的是麻省理工学院计算机科学与人工智能实验室（MIT CSAIL）的Dina Katabi团队，在CVPR2018上发表的一篇文章。在过去10年里，Katabi教授一直在研究如何让计算机看到墙后的事物。

人体姿态估计是计算机视觉中的一类重要任务，常被应用于监视，活动识别，游戏等。该问题被定义为生成由关节和躯干上的关键点组成的2D形式的骨架以描述目标的姿态。虽然，基于图像的解决办法已经在这个任务上取得了长足的进步，但是这些基于相机的工作都存在一个共同的难题，即无法准确的识别被遮挡部分的姿态。过去的解决办法是根据目标未被遮挡的部分，推断其被遮挡部分的姿态。然而，由于人体的姿态是多变的，这种推断方法很容易出错。此外，当目标被墙壁完全遮挡时，这种方法变得不可行。不同于可见光很容易被墙壁和不透明物体阻挡，Wi-Fi频率范围内的射频（RF）信号可以穿过这类遮挡物。由于RF信号会被人体反射，因此接收到的反射信号中隐含有反射面与收发装置的位置关系，从而提供了利用RF信号估计人体姿态的机会。过去，已有很多基于无线信号的相关工作，可是这些工作都比较粗糙，仅能识别出身体的某一部分动作，而且无法用于多人的环境。使用无线信号来产生类似于现有技术的计算机视觉系统所实现的详细和准确的姿势描述，仍然是一个艰难的挑战。

本文提出了一种基于无线信号的姿势估计方法RF-Pose，这是一个利用神经网络的系统解析无线信号并提取准确的2D人体姿势。即使人们被遮挡或在墙后，RF-Pose也能准确的估计其姿态。图 1为RF-Pose的一个结果演示。

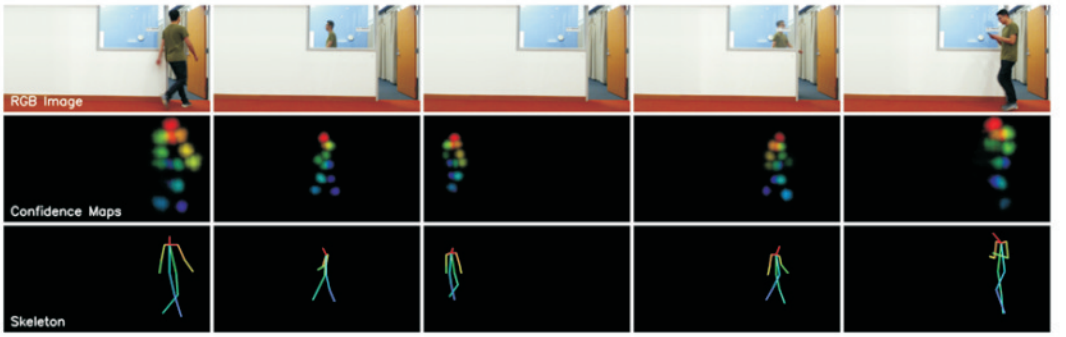
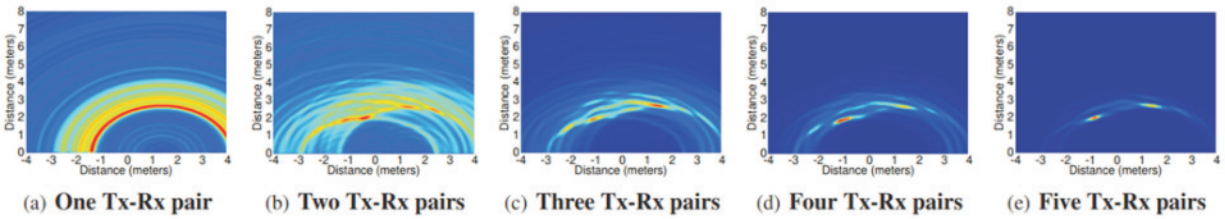



图 1. 该图显示了单人场景下的测试示例

它展示了RF-Pose跟踪目标进入房间，甚至当他完全被遮挡在墙后时的表现。RGB Image：由与无线电传感器共同定位的相机捕获的图像。Confidence Maps：仅由RF信号提取出的关键点置信图。Skeleton：从Confidence Maps解析的骨架表明即使存在完全遮挡，RF-Pose也

可以使用RF信号来估计人体姿势。

RF-Pose使用的是Katabi研究小组自行设计的RF设备。该设备的以极低的功率（功率比Wi-Fi低1000倍）发送固定模式的调频信号，由于室内环境布置的影响，在一段时间内，接收端会陆续收到经过不同路径反射的相同模式的调频信号，再根据各反射信号的强度推算反射面至接收天线的可能位置，最后按照图 2 (a)画出距离–能量热力图。由于RF-Pose的设备使用多个接收天线，通过简单的叠加就能够得到如图 2 (e)所示的教为准确的2D结果。




(a) One Tx-Rx pair (b) Two Tx-Rx pairs (c) Three Tx-Rx pairs (d) Four Tx-Rx pairs (e) Five Tx-Rx pairs

图 2. 不同天线数对目标位置判断的影响

图 2仅仅解决了目标位置的估计，实现姿态估计还需要目标各关键点的高度信息。此外，RF-Pose还需要解决如何根据位置和高度信息生成关键点置信度映射图的问题。因此，RF-Pose设计了一个跨模式的师生网络模型，如图 3 所示，该网络模型使用同步图像的视觉信息帮助RF-Pose训练由RF信号生成关键点置信图的学生网络模型。

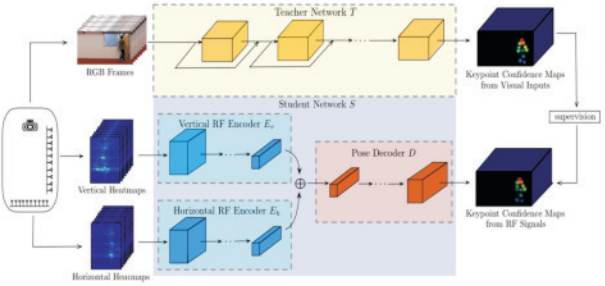



图 3. 用于RF-Pose的师生网络模型

在训练阶段，RF-Pose利用两组天线分别提取水平和垂直的热力图，使用两个10层的编码器分别对两组热力图进行编码，然后再通过一个4层的解码器对组合后的编码进行解码得到关键点置信度映射图。同时，在与RF设备相近的位置，设置了一个照相机同步对目标进行拍照。RF-Pose使用了一个由卡耐基梅隆大学团队在CVPR 2017上发布的姿态估计模型OpenPose 作为教师网络，来监督学生网络训练。完成训练后，RF-Pose将不再需要视觉信号，仅通过RF信号，即可对目标的姿态做出准确的估计。

在实验部分，RF-Pose以OpenPose为基准，在50个不同的环境下进行了长达50个小时的对比实验。实验的结果表明，在正常情况下，RF-Pose与OpenPose的表现相似。在目标被墙壁遮挡的情况下，OpenPose无法估计目标姿态，而RF-Pose的表现仅稍逊于目标被挡住的情况。

李辉楚吴
2018级博士
研究方向：智能感知
Email：credolee@hust.edu.cn

# FoggyCache: Cross-Device Approximate Computation Reuse

武 静 推荐

“FoggyCache: Cross-Device Approximate Computation Reuse”是发表在移动计算和网络领域会议MobiCom 2018的一篇文章。文章主要研究在移动计算和物联网场景中，实现不同端设备计算结果重用的问题。

移动计算和物联网场景下的应用需要完成高实时性的交互任务和计算密集型的识别任务，端设备有限的计算资源和电量资源无法满足上述需求，大量任务需要卸载到服务器端进行处理。“远端卸载”有效解决了端设备资源不足的问题，但是应用响应时间会增大。现有工作通常从应用层面出发，确认任务的处理“位置”，即在本地完成处理还是在服务器端完成处理；然而，该文从设备层面出发，通过分析输入数据的相似性，实现不同设备间计算结果的重用，避免重复计算，进而缩短应用的响应时间和设备的电量消耗。

上述“跨设备”的计算结果重用存在以下两方面的挑战：1）不同设备的输入数据不是精确相等，而是时间、空间及语义方面的相似，且数据分布会随时间的变化而变化，系统需要基于输入数据的特点迅速检索出“最相似”的若干条记录（包括输入数据和对应的计算结果）。2）输入数据的相似并不能保证重用结果的准确。例如，不同设备上相似的输入可能对应不同的输出，盲目使用某一结果作为重用结果可能会导致识别错误。

为了应对上述挑战，作者提出了根据数据分布进行自适应调整的相似度比较算法A-LSH（Adaptive Local Sensitive Hashing）和具有高准确度的重用结果选择算法H-kNN（Homogenized k Nearest Neighbors）。其中，A-LSH 先使用kNN（k Nearest Neighbors）求得“最相似”的k条记录，并计算其距离，然后求解距离的中位数和95分位数，最后将两者中的较大值设置为A-LSH的核心参数局部敏感因子。A-LSH继承了LSH（Local Sensitive Hashing）对高维数据快速检索的优点，同时克服了LSH无法感知数据分布特征的缺点。H-kNN充分利用聚类结果的统计特征来提高重用结果的准确度。作者将上述算法封装成服务FoggyCache，详细设计如图1 所示。FoggyCache 是典型的服务器–客户端模式。运行在端设备上的FoggyCache客户端会拦截所有“远端卸载”请求，使其先确认本地A-LSH缓存是否有可重用的结果。若有，直接返回该结果；若无，将该请求转发至对应的服务器。FoggyCache服务端负责收集计算结果，并将相应内容同步到各FoggyCache客户端。

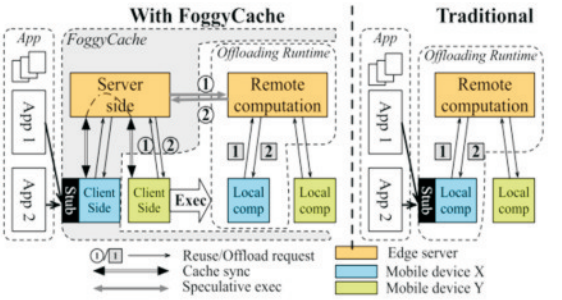


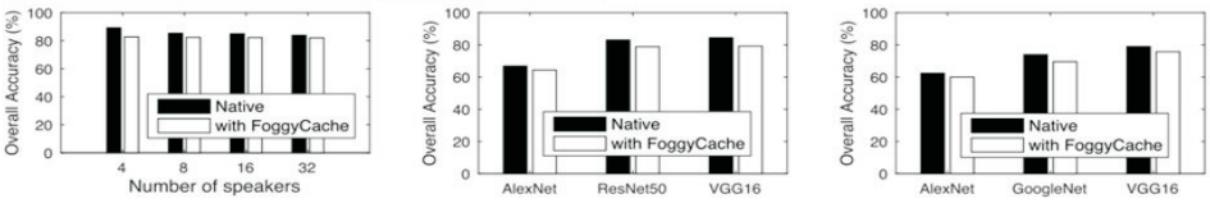

图1 FoggyCache的系统架构

# Kpart: A Hybrid Cache Partitioning-Sharing Technique for Commodity Multicores

陈　聃　推荐

“Kpart: A Hybrid Cache Partitioning-Sharing Technique for Commodity Multicores”是被IEEE 高性能计算架构会议（HPCA 2018）录用的一篇文章，该会议于2018年2月在奥地利维也纳召开。KPart提出了一种混合缓存分区共享技术，将应用程序分组到集群中，然后根据这些集群划分缓存大小。KPart依赖于一种新的技术评估应用程序在共享缓存时所遭受的性能损失来构建集群，其中集群的个数KPart会通过详细的分析信息来选择最佳值以提高整体性能。

接上页




(a) Speech identification　(b) Landmark recognition　(c) Plant recognition

图2 部署FoggyCache前后识别精度的对比

表1 部署FoggyCache前后应用响应时间和端设备电量消耗对比

| Workload | | | Latency (ms) | | | | Energy (mJ) | |
|---|---|---|---|---|---|---|---|---|
| Application | Description | | Offl. (w/o) | Offl. (w/) | Without | With | Without | With |
| Speaker Identification | Number of speakers | 4 | 28.1 | ***8.4*** | 13.1 | ***4.2*** | 30.4 | ***9.8*** |
| | | 8 | 28.1 | 11.8 | 13.1 | 5.5 | 30.4 | 13.3 |
| | | 16 | 28.1 | 12.1 | 13.1 | 5.9 | 30.4 | 13.7 |
| | | 32 | 28.1 | 13.2 | 13.1 | 6.4 | 30.4 | 15.0 |
| Landmark Detection | Types of neural network | AlexNet [46] | 24.6 | 16.3 | 37.1 | 19.5 | 365.9 | 39.5 |
| | | ResNet50 [37] | 32.8 | 17.7 | 102.4 | 27.9 | 1315 | 110.7 |
| | | VGG16 [73] | 53.8 | ***21.4*** | 269.6 | ***57.3*** | 3132 | ***246.9*** |
| | Video feed (campus) w/ VGG16 | | 53.8 | ***12.0*** | 269.6 | ***25.4*** | 3132 | ***114.2*** |
| Plant Recognition | Types of neural network | AlexNet | 24.6 | 16.6 | 37.1 | 21.4 | 316.8 | 113.9 |
| | | GoogleNet [76] | 29.2 | 17.9 | 65.3 | 32.2 | 817.4 | 236.8 |
| | | VGG16 | 53.8 | ***27.9*** | 269.6 | ***99.8*** | 3132 | ***901.4*** |
| | Video feed (grocery) w/ VGG16 | | 53.8 | ***16.5*** | 269.6 | ***30.8*** | 3132 | ***131.1*** |

在实验部分，作者用三个不同场景下的真实数据测试了FoggyCache的性能。综合图2及表1可知，FoggyCache在精度损失低于5%的情况下，将响应时间缩短了50%到88%，电量消耗减少至部署前的3%及20%。

武　静
2018级博士
研究方向：边缘计算、AR
Email：wujinghust@hust.edu.cn

信息可以离线收集，也可以通过一种新的分析机制在低开销的情况下在线收集。

KPart提出了当前缓存存在的问题：目前系统实现了way-partitioning，但是提供的分区数量有限，而且经常会影响性能。如图1，对每个应用程序分区，每个应用程序都映射到不同的分区，这种分区方式带来的问题是相联性极差，结果导致比在未分区基础上的吞吐量下降了3.8%。

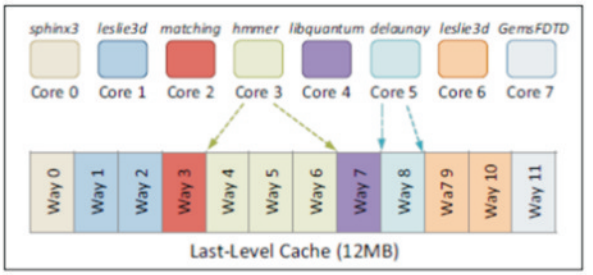



图1 当前系统的分区划分

KPart通过对应用程序进行集群划分，然后再对集群划分缓存大小来实现混合缓存分区共享技术，在Intel Broadwell-D系统上KPart平均提高了24%的吞吐量。为了实现对应用程序进行集群划分，KPart使用的是Distance metric方法，如图2，是根据将应用程序组合共享一定的缓存大小的缺失率和应用程序分别独立占用这个缓存大小的一部分的总缺失率进行比较，判断这组应用程序适不适合划分到同一个集群，一个集群内的应用程序是共享相同的缓存大小，当这组应用程序共享缺失率与所有应用程序独立占有总缺失率相差太多时，就不适合将其放入到一个集群。

实现对集群划分缓存大小分配，KPart采用的还是原有的UCP's Lookahead 算法，KPart提出可以将每个集群的缺失曲线输入到 Lookahead

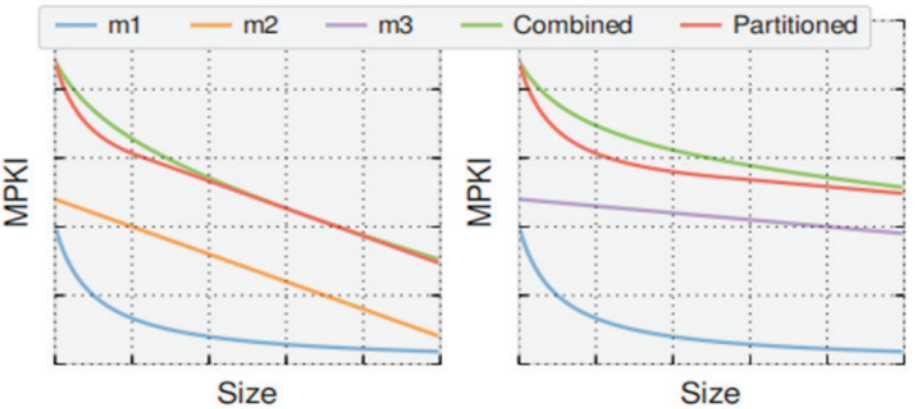



图2 Distance metric 方法

算法中，以找到在整个系统中最大限度地减少总缓存缺失的缓存大小，而在这里，KPart没有使用每个集群的缺失曲线，而是重新定义了speedup曲线，定义如下：

$$speedup(S) = \frac{IPC(S)}{IPC(C/N)}$$

S是集群分配到的缓存大小，C是整个缓存大小，N是应用程序的数量。当使用speedup曲线相对于缺失曲线，实验评估可以观察到平均高出5%的加权加速比。

陈 聃
2018级博士研究生
研究方向：异构内存、缓存管理机制
Email：cdhust@hust.edu.cn

# Fine-grained Concept Linking using Neural Networks in Healthcare

陈 林 推荐

“Fine-grained Concept Linking using Neural Networks in Healthcare”是数据库领域顶级学术会议SIGMOD 2018录用的一篇文章。本届会议于2018年6月在美国德克萨斯州休斯敦市召开。

近些年，包括电子健康记录，医学图像在内的松耦合医疗数据数量迅猛增长，跨领域医疗数据分析需要有关医学概念的数据集成，在现实生活中这些概念通常是由文本片段所表示。因此，要想利用好医疗数据，一个关键的步骤就是将这些文本片段连接到与之对应的概念上去。但是，医院数据库中的文本片段大多是经过很多年由不同的临床医生录入进去的，他们之间存在着各种不同的书写风格和书写标准，其中首字母缩写，单词缩写，简写现象非常普遍。这样就可能会导致医生所写的文本片段和与之对应的标准概念描述大相径庭。这种医疗文本片段和对应的标准概念描述之间严重的单词矛盾使得文本片段的概念连接变得异常困难。另一方面，由本体所组成的概念经常会出现语义重叠，这进一步加大了概念连接的难度。

正是因为医疗数据有着巨大的应用价值，有关医疗数据中的概念连接已经有了很多研究结果。现有概念连接方法主要分为三类：词典库方法，基于机器学习的方法，联合方法。对于词典库方法，在一个大规模医疗数据库中，要严格包含所有单词到术语和术语到概念的对应组是非常困难的。对于基于机器学习的方法，手写特征只在概念数量比较少的情形下有效，却不适用于大概念数据集，因为这中间还有很多新的因素需要考虑。

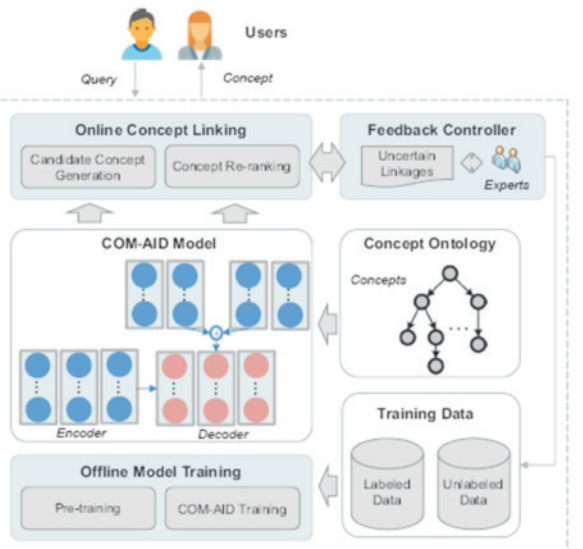



图1 神经概念连接框架

为了解决这些问题，本文提出了一种新颖的神经概念连接方法（Neural Concept Linking，NCL），不同于依靠字典或者一些手写特征，NCL利用各种从知识库中抽取出来的概念描述和由本体提供的概念层来训练已设计的组合注意力编码–解码神经网络。这个神经网络为每一个概念合成了一个数值向量。这些合成向量捕捉概念标准描述到各种别名连接过程中的语义翻译，并且使概念间的层次关系具

体化。基于这些合成向量，神经网络可以将一个概念翻译成任意一个查询，并且评价翻译质量。特别地，这个神经网络利用一个定制的翻译过程来计算从一个概念生成文本片段的概率。在NCL 中，神经网络是离线训练的，在这个离线查询过程中，NCL首先利用一个轻量级关键字匹配器来选择一些概念，然后根据由神经网络计算出来的翻译概率将这些概念排序，排在第一的概念就作为概念连接结果返回给用户。

在整个NCL框架中，最关键的就是组合注意力编码解码（composite attentional encode-decode，COM-AID）模型。图2表示了COM-AID的网络结构。由图可知，COM-AID将概念$c$ 和查询$q$作为输入，然后得到一个从概念$c$产生$q$ 的概率。一方面，左边的概念编码器将文本片段作为输入并利用基于RNN的神经网络将每一个单词编码到隐层状态$h_t^c$。另一方面，右边的解码器将生成的$h_t^c$与被视为一串单词的查询$q$作为输入，然后借助于文本结构注意力机制来评价$h_t^c$解码到$q$的效果。这里的文本机构注意力机制指的是：1）解码器关注与$h_t^c$更相关的部分以此来捕获文本片段的核心含义；2）解码器通过利用本体结构来关注有关$c$的概念。

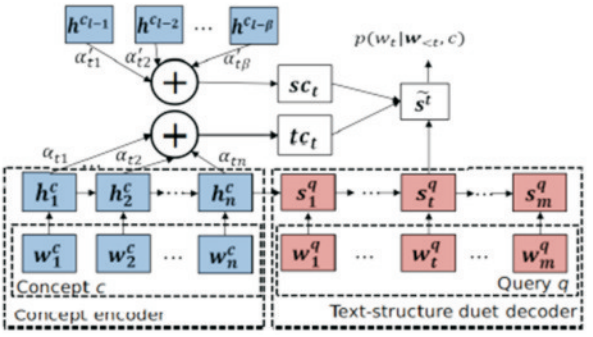



图2 COM-AID模型的网络结构

本文第一次将基于神经网络的方法用于医疗数据概念连接，实现了现实中简要且嘈杂的文本片段与本体中细粒度知识库概念的连接。借助于设计好的COM-AID神经网络，概念连接通过一个翻译过程来完成，在这里COM-AID利用注意力机制同时考虑了翻译过程中的文本和结构背景，这样就同时解决了一个概念有多种文本片段描述和一个文本片段可能匹配多种概念的这两个问题。文章最后通过使用两个真实的医院数据集来进行实验研究，实验结果表明NCL产生了准确的概念连接结果并且明显优于目前主流的基于词典，基于词嵌入，基于逻辑回归等其他方法。总体说来，NCL能具有如此优秀的性能主要是因为以下三个原因：第一，鉴于它强大的端对端语义翻译性能，编码–解码过程以某种方式消除了表面字符串之间的单词矛盾；第二，医疗数据集包含对一个概念的各种描述，因此可以通过自翻译来捕捉概念的核心含义；第三，组合注意力机制在解码过程中增加了更多的信息，提高了连接结果的准确度。

陈　林
2018级博士生
研究方向：深度学习，隐私保护
Email：chenl@hust.edu.cn

# Processing-in-Memory for Energy-efficient Neural Network Training: A Heterogeneous Approach

刘 聪 推荐

“Processing-in-Memory for Energy-efficient Neural Network Training: A Heterogeneous Approach”是被微体系架构会议（MICRO2018）录用的一篇文章，该会议于2018年10月在日本福冈召开。文章提出了一种异构内存处理（PIM）系统的软硬件协同设计方案来进行高效的神经网络（NN）训练。硬件设计结合了数百个固定功能的算术单元和基于ARM的可编程核在一个3D芯片堆叠存储器的逻辑层，形成一个附加在CPU 上的异构PIM体系结构；软件设计提供了一个编程模型和一个运行库系统，用于跨CPU和异构PIM提供的计算资源编写、卸载和调度各种神经网络训练操作。

神经网络模型使用越来越多的参数和数据集，训练这种复杂的模型需要巨大的计算和内存资源、能量和时间，在训练NN时，一个关键的能耗和性能瓶颈是系统中的数据移动，而内存处理（PIM）技术被认为是解决各种应用中的数据移动挑战的一种非常有效的解决方案。

本文的目标是设计一个基于PIM的NN训练加速系统，该系统能够有效地加速编写在广泛使用的机器学习框架上的未经修改的训练模型。为了实现目标，作者提出了一个软硬件协同设计的异构PIM框架。设计由三个部分组

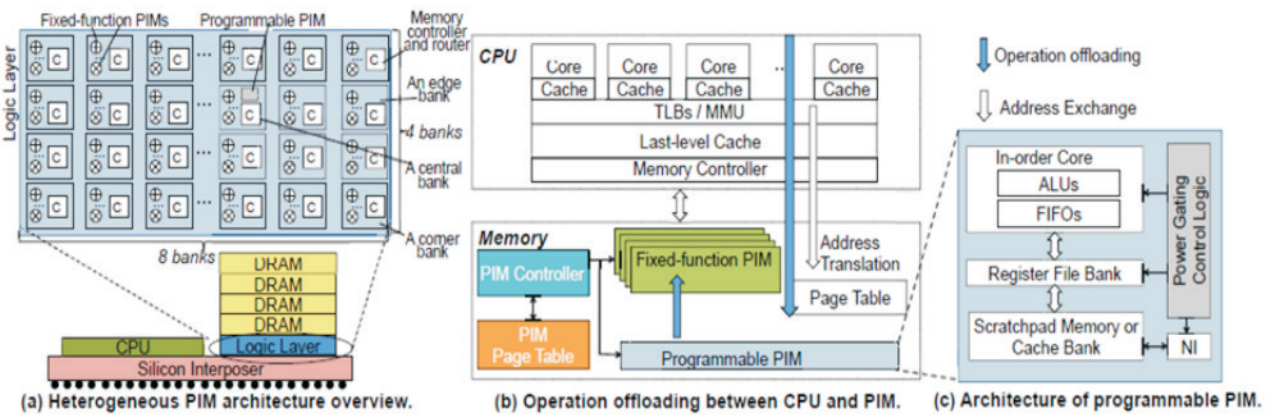



图1 异构PIM的体系结构概述

成：首先，采用了一种异构的PIM体系结构，它将固定功能逻辑和可编程核集成在三维堆叠主存中；其次，扩展OpenCL编程模型以解决编程挑战。该编程模型将主机CPU和异构PIM映射到OpenCL的平台模型上，并扩展OpenCL的执行和内存模型，以实现高效的运行库调度；最后，提出一个运行库系统，它最大限度地提高了PIM硬件利用率和NN操作级并行性。

本文遵循软硬件协同设计原则。采用的异构的PIM体系结构包括一个可编程的PIM，它是一个ARM核，还包括大规模的固定功能PIM，它是分布在所有内存块上的加法器和乘法器。虽然此设计可以用于各种3D芯片堆叠存储设备，但本文以32组内存堆栈为例（如图1（a）），其中bank是堆栈中的垂直切片。由于平台模型包括多种类型的异构设备，这种平台模型是由NN训练操作的特点驱动的。所以本文的OpenCL编程模型增加了（1）递归内核调用，使PIM之间的内核调用支持复杂的NN操作（例如Conv2DBackpropFilter）；（2）操作流水线，以提高有限并行性的小型NN操作的硬件利用率；（3）扩展了内存模型，以支持主机处理器和加速器之间共享的单个全局内存。运行库系统负责对固定功能的PIM、可编程PIM和CPU的调度操作。为了减少NN训练时间，运行库力求最大限度地利用PIM和CPU来优化系统吞吐量。

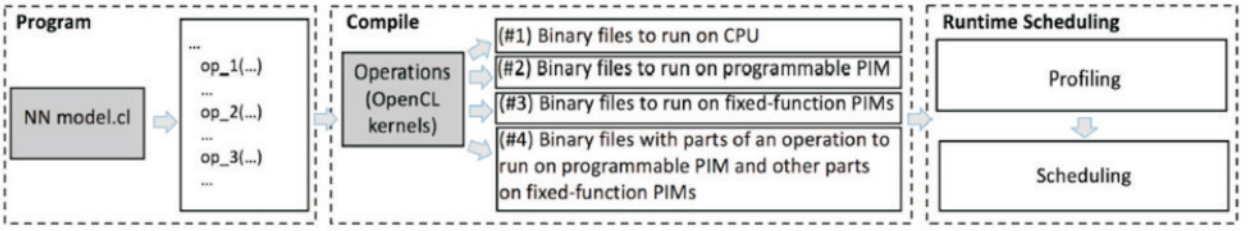



图2 本文的软件框架设计进行神经网络训练的过程

为了评估本文的设计，作者选取了5种训练模型VGG-19，AlexNet，Deep Convolutional Generative Adversarial Networks（DCGAN），ResNet-50，Inception-v3，并比较了5种配置：分别在CPU、GPU、可编程PIM、固定功能PIM 上执行所有训练操作；在设计的异构PIM上执行操作。为了更好地评估论文提出的模型，分别就执行时间、能耗、敏感性研究、软件影响等多个方面进行了讨论。通过各种NN训练负载，证实了本文提出的异构PIM在性能和能源效率方面都取得了显著的改善。

刘　聪
2018级博士研究生
研究方向：内存计算
Email：liu_cong0321@163.com

# T-Fuzz: fuzzing by program transformation

石 建 推荐

“T-Fuzz: fuzzing by program transformation”是国际顶级安全会议Security and Privacy 2018的录用论文。这篇文章提出了基于程序变换的模糊测试方法T-Fuzz。T-Fuzz通过移除程序中复杂的sanity check来提高模糊测试的覆盖率，使得模糊测试能够执行被复杂sanity check保护的代码，从而发现隐藏更深的漏洞。

## 背 景

模糊测试是当今最有效的漏洞挖掘方法。模糊测试方法通过随机产生的输入测试程序，发现其中的bug。但模糊测试却存在代码覆盖率低的问题，因为程序中存在复杂的sanity check，例如，对魔数、校检和的检查，随机产生的输入无法满足sanity check，所以无法发现隐藏在程序较深的执行路径中的bug。为了解决代码覆盖率的问题，已有的方法依靠不精确的启发式方法或者复杂的输入变异技术绕过sanity check，但其作用有限。

作者对santiy check进行了一些探讨和研究，sanity check可以被分为两类：NCC（Non-Critical Check）和CC（Critical Check）。NCC 是那些用来过滤数据的check，例如对魔数的检查；CC是严重影响程序功能和逻辑的check，例如，TCP包分析程序中对长度的检查。NCC可以在不产生误报的情况下被去掉，去掉NCC后产生的bug能在原程序中复现，但是去掉CC，可能引入误报，这些bug不能在原程序中复现。

## 方 案

为了改进模糊测试的效果，这篇文章提出了T-Fuzz，图1展示了T-Fuzz主要的组成部分和工作流。T-Fuzz使用AFL作为其模糊测试工具，其设计思路主要分为三部分：1）发现NCC，2）程序变换，3）过滤误报并报告bug。

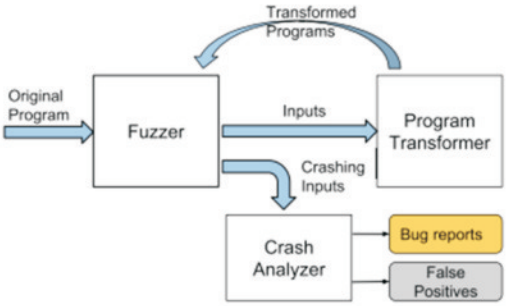



图1 T-Fuzz

**1）发现NCC：**

发现NCC是T-Fuzz中最重要的一步。为了性能的原因，作者过近似的认为模糊测试工具产生的所有输入都绕不过去的check都是NCC。当模糊测试工具被卡住时，T-Fuzz利用一个轻量级、低准确性的动态追踪方法检测NCC：Sanity check会被编译为条件跳转指令，在CFG（control flow graph）中表现为基本块S指出的T和F两条分支。所有输入都绕不过去的check意味着模糊测试产生的所有输入在此处都执行T的分支或者都执行F的分支。

**2）程序变换：**

T-Fuzz对程序进行变换，移除检测到的NCC，方式是将NCC替换为相反的条件跳转。因为这种方式没有改变变换后程序中基本块的地址，在变换后的程序中的追踪可以直接映射到原程序中，所以这种方式保留了原程序的结

# Improving Accuracy of Android Malware Detection with Lightweight Contextual Awareness

吴月明 推荐

“Improving Accuracy of Android Malware Detection with Lightweight Contextual Awareness”是国际顶级会议ACSAC'18收录的一篇论文。本文通过分析APP中的入口点函数和敏感API之间是否存在着可达路径来进行频数分析，并以此构建特征向量来进行恶意软件检测。

现有context-based的恶意软件检测方法包含了太多的context信息，尽管现有工作已经验证了它们在检测恶意软件上的有效性。但是，过多的context信息会导致运行开销过大，使得它们不具有扩展性。因此，本文提出了一个疑问：在这么多的context信息中，这些context信

接上页

构同时保留了获取路径约束所需要的信息。这种程序变换的好处是减小了崩溃分析时原程序和变换后程序的不同所带来的复杂度。

**3）过滤误报并报告bug：**

因为移除的NCC中可能包含在原程序中有重要作用的check，所以对变换后程序的模糊测试可能会产生误报。T-Fuzz的崩溃分析器过滤误报的方式是验证出现在变换后程序中的bug是否同样出现在原程序中。T-Fuzz追踪变换后程序中输入到达崩溃点的路径约束，该路径约束同样代表了原程序的路径约束。路径约束的可满足行表明了这个崩溃是不是误报，如果路径约束可以在约束求解器中得出答案，则T-Fuzz 报告该bug。

T-Fuzz从一个不同的角度来解决模糊测试中覆盖率的问题，与传统的变异输入的思路不同，T-Fuzz通过变换目标程序实现扩大覆盖率的目标。然而，这种方式也存在一些问题：1）过滤误报的方式依然还需要依靠路径约束求解，相当于把符号执行的过程延后。这种方式绕过了复杂检查，可能发现隐藏更深的漏洞，但计算复杂度并没有降低。2）存在一个sanity check就需要做一次程序变换，所以T-fuzz面临变换版本爆炸的问题。如果sanity check很多，则需要生成不同的变换程序，T-Fuzz对变换后的不同版本程序分别做模糊测试。

## 测　试

T-Fuzz在 DARPA Cyber Grand Challenge（CGC）数据集、LAVA-M和4个真实程序（pngfix，tiffinfo，magick和pdftohtml）上进行了测试，T-Fuzz与现有技术相比覆盖了更多的代码而且发现了更多的bug。在CGC数据集中，T-Fuzz在166个二进制程序中发现了bug，而Driller和AFL分别发现了121个和105个。

石　建
2015级博士
研究方向：软件安全
Email：shijianwh@hust.edu.cn

息各自在区分是不是恶意软件时效果是不是不一样？有哪些context信息在区分是不是恶意软件时效果很好？基于这个疑问，本文做了一个全面的调研实验，利用AppContext（AppContext是发表于ICSE2015的一篇文章）来提取不同的context信息。本文一共收集了54969个敏感行为，每一个行为都包含着不同的context信息。最后本文采用了岭回归来对每一个context信息进行重要性分析。实验结果如图1所示，从图中可以看到，entry points在区分是不是敏感行为时占有很大的比重。基于这个发现，本文提出了一个轻量级的安卓恶意软件系统，PikaDroid。

| Method Type | Activity % (Entrypoints) | Receiver % (Entrypoints) | Intent_params% | Service % (Entrypoints) | Database % | Device % | UI % (Entrypoints) | Network % |
|---|---|---|---|---|---|---|---|---|
| CLASS_LOADING | **47** | **8** | 3 | **13** | 4 | 7 | **12** | 1 |
| REFLECTION_METHOD | **39** | **5** | 10 | **3** | 7 | 9 | **8** | 10 |
| ACCESS_COARSE_LOCATION | **22** | **0** | 20 | **11** | 1 | 15 | **3** | 16 |
| ACCESS_NETWORK_STATE | **27** | **2** | 6 | **15** | 4 | 13 | **11** | 14 |
| ACCESS_WIFI_STATE | **34** | **9** | 9 | **21** | 0 | 7 | **3** | 9 |
| BROADCAST_STICKY | **43** | **2** | 2 | **30** | 0 | 0 | **10** | 0 |
| CAMERA | **68** | **0** | 0 | **0** | 0 | 0 | **20** | 0 |
| CHANGE_WIFI_STATE | **21** | **12** | 9 | **33** | 0 | 1 | **11** | 4 |
| DISABLE_KEYGUARD | **21** | **0** | 0 | **79** | 0 | 0 | **0** | 0 |
| GET_ACCOUNTS | **59** | **1** | 4 | **7** | 0 | 0 | **16** | 0 |
| INTERNET | **43** | **6** | 5 | **12** | 2 | 6 | **13** | 2 |
| READ_PHONE_STATE | **21** | **2** | 4 | **35** | 5 | 16 | **5** | 6 |
| RECORD_AUDIO | **32** | **1** | 0 | **63** | 0 | 0 | **3** | 0 |
| RESTART_PACKAGES | **51** | **5** | 3 | **6** | 1 | 3 | **12** | 0 |
| VIBRATE | **43** | **4** | 6 | **20** | 1 | 2 | **12** | 3 |
| WAKE_LOCK | **44** | **4** | 4 | **13** | 3 | 3 | **16** | 1 |
| WRITE_SETTINGS | **53** | **1** | 8 | **4** | 4 | 1 | **12** | 3 |

**Table 1: A detailed report of the findings in our case study on contextual factors. This table describes the relation between sensitive API categories and contextual factors. Each row represents a API category and each column represents a Context Category, and the value represents feature importance. The feature importance represents the predictive power a feature has in a learning problem (higher scores are better).**

图1 不同context信息的重要性分析结果

PikaDroid的整理框架图如图2所示，总共分为2个模块：静态分析模块和学习模块。在静态分析模块中，总共分为3个小阶段，分别为：调用图的提取和入口点函数的发现，可达性分析，频率分析。本文采用静态分析工具WALA进行调用图的提取，在提取出APP的调用图之后，再利用WALA从调用图中挖掘出所有的入口点函数。在找到调用图中的所有入口点函数之后，PikaDroid会进行可达性分析，分析这些入口点函数和敏感API调用之间是否存在着可达路径，如果存在着可达路径，那么就将这个入口点函数和敏感API调用记录为一个pair（entrypoint, targetAPI）。提取出所有APP的所有入口点函数和敏感API调用之间的可达pairs之后，PikaDroid再对这些pairs进行频数分析来分配不同的权重。比如，入口点函数Service.onStart()到达API调用TelephonyManager. getDeviceID()这个pair在恶意软件集中出现的次数是在正常软件中出现的11.05倍，因此这个pair 分配的权重就是11.05。另一个方面，如果我们只考虑敏感API在恶意软件和正常软件中的频数，并以此分配权重的话，TelephonyManager. getDeviceID()的权重只有0.401，那么在检测恶意软件时会造成误报。

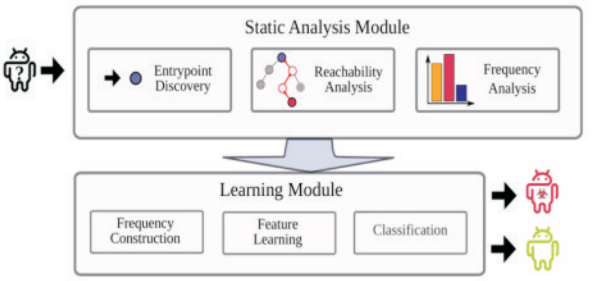



图2 PikaDroid系统框架图

将所有的pairs都分配好权重之后，

PikaDroid会将这些pairs的权重作为特征，当给定一个新的APP来检测时，PikaDroid会首先通过静态分析提取调用图，然后挖掘入口点函数并通过可达性分析找到这个APP的所有入口点函数到敏感API调用的pairs，然后将这些pairs分配前面已经计算出来的权重，并以此构建特征向量进行恶意软件分类。

为了验证PikaDroid的有效性，本文将PikaDroid和MaMaDroid进行了一个实验比较（MaMaDroid是NDSS2017的一篇文章）。实验一共分析了23631个APP，并将这些APP根据其开发时间分为了3个时间段：2010-2012，2013-2015和2016-2018。实验结果如表1所示，从图中可以看到，PikaDroid在F1值上全都优于MaMaDroid，这也验证了PikaDroid在检测安卓恶意软件上的有效性。

表1：PikaDroid和MaMaDroid的实验结果对比

| Dataset | | Accuracy (%) | | Precision (%) | | F1 Score (%) | | Recall (%) | | FP Rating | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Samples | Time Period | Pika | MaMa | Pika | MaMa | Pika | MaMa | Pika | MaMa | Pika | MaMa |
| Malware vs. Benign | 2010-12 | **97.61** | 94.66 | **98.29** | 97.13 | **97.65** | 94.64 | **97.02** | 92.28 | **1.76** | 2.87 |
| | 2013-15 | **98.43** | 97.55 | **99.24** | 97.55 | **97.89** | 96.70 | **96.57** | 95.87 | **0.44** | 1.45 |
| | 2016-18 | **97.56** | 96.53 | 96.90 | 97.08 | **96.07** | 94.27 | **95.24** | 91.61 | 1.38 | 1.25 |
| | 2010-18 | **98.00** | 95.67 | **98.52** | 95.81 | **97.41** | 94.58 | **96.33** | 93.38 | **0.96** | 2.77 |
| Grayware vs. Benign | 2010-12 | **98.15** | 95.68 | **97.98** | 97.85 | **96.74** | 92.02 | **95.53** | 86.84 | 0.79 | 0.77 |
| | 2013-15 | **96.91** | 92.63 | **97.80** | 91.09 | **94.04** | 85.45 | **90.57** | 80.47 | **0.75** | 2.90 |
| | 2010-15 | **96.83** | 93.22 | **96.92** | 94.86 | **94.15** | 86.78 | **91.54** | 79.97 | **1.12** | 1.67 |
| (Mal-&Grayware) vs. Benign | 2010-12 | 96.99 | 97.55 | **97.87** | 97.55 | **97.44** | 96.70 | **97.03** | 95.87 | 3.06 | 1.45 |
| | 2013-15 | **96.54** | 93.57 | **97.59** | 94.60 | **96.44** | 93.39 | **95.32** | 92.21 | **2.28** | 5.10 |
| | 2010-18 | **97.79** | 92.63 | **98.27** | 93.90 | **97.47** | 92.12 | **96.70** | 90.40 | **1.34** | 5.33 |

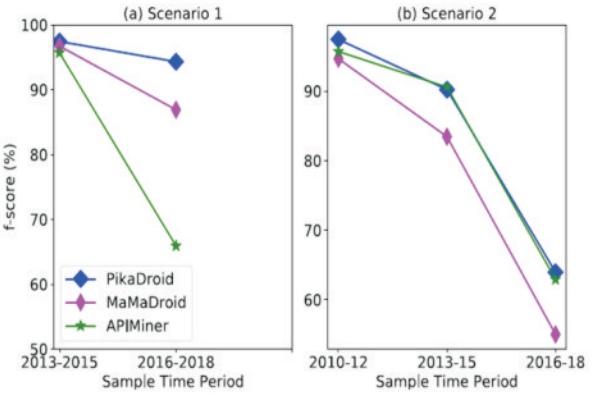



图3：PikaDroid、DroidAPIMiner和MaMaDroid在检测新APP时的实验结果

此外，本文还选取了另外一个工具DroidAPIMiner，并针对DroidAPIMiner和MaMaDroid进行了第二个实验：通过训练一个旧的数据集得到一个训练模型，并用这个模型来检测新的不同时间段的APP。实验结果如图3所示，从图中可以看到，PikaDroid在检测新的时间段的恶意软件时效果要优于MaMaDroid和DroidAPIMiner，这也验证了PikaDroid在针对安卓APP的进化上具有更强的鲁棒性。

本文提出了PikaDroid，一个新的安卓恶意软件检测系统。通过分析APP中入口点函数和敏感API之间的可达性挖掘不同的pairs，并利用频数分析针对这些pairs分配相应的权重。当给定一个新的需要检测的APP时，通过分析该APP是否包含某些pairs并以这些pairs的权重构建特征向量进行恶意软件检测。

吴月明
2016级博士
研究方向：安卓恶意软件检测，漏洞检测
Email：wuyueming21@163.com

# A Distributed Framework for Emerging AI Applications

吴文超　推荐

“A Distributed Framework for Emerging AI Applications”是发表在OSDI 2018上的一篇论文。OSDI和SOSP作为系统领域会议皇冠上“明珠”，在学术和工业界有着巨大的影响力。作者来自UC Berkeley，许多优秀的开源分布式系统如spark都曾诞生于此。

这篇文章主要指出AI应用，特别是强化学习应用对于系统设计带来的新的挑战，作者提出支持任务并行和actor模型结合的动态图计算模型，利用共享存储实现中心化的状态控制，使用自底向上的任务调度机制，实现了灵活，可拓展，低调度延迟的系统。实验表明，该系统支持多种强化学习应用，实现了亚秒级别的远端任务调用延迟，达到了180万每秒的任务调度。接下来我们将以问题为导向来解读这篇文章的设计思想。

**问题1：强化学习的主要的特征是什么？为系统设计带来哪些挑战？**

强化学习相对于传统的机器学习，需要与环境不断进行交互，根据环境的状态，评估策略，得到当前环境下的动作，通过对于实施动作后的新状态和奖励来不断更新策略。主要具有以下几个特征，第一：得到好的策略需要对于环境进行大量模拟。第二：强化学习计算图异构并且动态演化，不同模拟运行时间从几微秒到数分钟不等。第三，强化学习应用特别是机器人控制，自动驾驶等需要快速做出响应。相应的，设计强化学习系统需要满足条件和挑战是：第一：灵活性，需要支持并发异构的任务，满足功能的多样性，例如机器人需要对于环境中多个传感器做不同计算，不同的仿真轨迹时间长短不一，需要资源类型各异（CPU，GPU）。之前BSP模型是对于不同数据分区实施相同计算，无法满足强化学习要求。第二：支持动态图计算。在训练策略阶段，可以等待部分仿真轨迹完成即开始更新策略，之后开启新的并行仿真过程，如图1所示，这种强化学习需求导致计算图发生动态改变，执行顺序无法预测。第三点：系统应该支持高并发任务调度，由于实时响应需求，调度本身不应该成为性能的瓶颈。第四：系统应该具备透明的容错功能，以及具备对于外部的库（比如各种模拟器）进行actor封装。

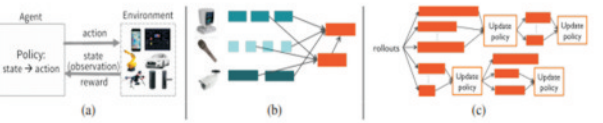



图1 强化学习任务计算特点图

**问题2：RAY提供哪些编程模型和接口？如何解决问题1的挑战？**

提供无状态，没有副作用的远端函数调用，返回feature，ray.get()阻塞直到到所有feature完成。ray.wait()在输入feature子集完成后即可返回，实现对于并发，动态异构的任务的支持。

远端函数可以指定资源需求（如GPU），调度器在任务执行时会自动高效管理和调度，满足任务异构资源需求。远端函数可调用其他的远端函数，一个feature可以作为其他的远端函数输入，这样实现灵活嵌套。RAY通过Actor的编程模实现了对有状态的组件（比如模拟器）无缝支持。

**问题3：RAY的架构是如何？**

如图2所示，RAY应用层主要包含3个主要的组件：driver，执行用户程序的进程，Worker：执行远端函数无状态进程，Actor：有状态的进程。

系统层主要包含3个组价：全局的控制存储，分布式调度器，分布式对象存储，全局控制存储负责元数据和控制状态信息，通过中心化存储方式，使得其他的组件变得无状态，简化了容错（出错的组件重启只需要从中心化的存储里面读取最新的状态），使得其他组件可以很容易地水平拓展。分布式调度器采用了自底向上调度方式，由全局调度器和每一个节点局部调度器组成，任务生成之后，局部调度器首先负责调度任务在本地执行，如果本地节点过载，或者本地的资源无法满足任务（比如没有GPU），任务才会发送到全局调度器，全局调度器通过心跳收集每一个节点最新的负载，资源信息，根据这些信息决定任务分配的节点。当全局的调度器成为瓶颈时候，可以通过多个副本并行调度。内存分布式存储，通过共享内存实现对象存储，如果一个任务执行过程中的输入不在本地，那么数据首先会被复制到本地存储，然后进行处理，处理结果的元数据信息保存在全局控制存储里面。类似于之前的spark，Ray通过血统+checkpoint机制实现错误的恢复，需要注意的是对于无状态的计算很容易恢复，而对于有状态的actor，在构造计算的时候，相同的actor不同调用，系统自动加入了有状态的边依赖，从而保证运行正确性。

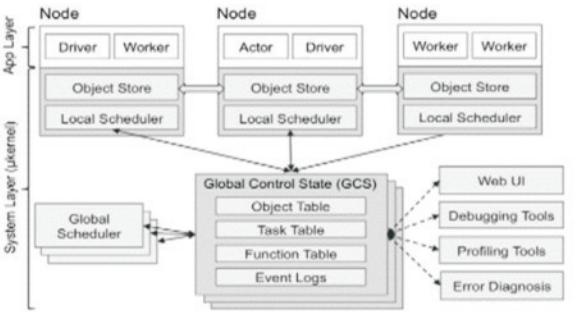



图2 RAY的系统架构

**问题4：系统的性能如何？**

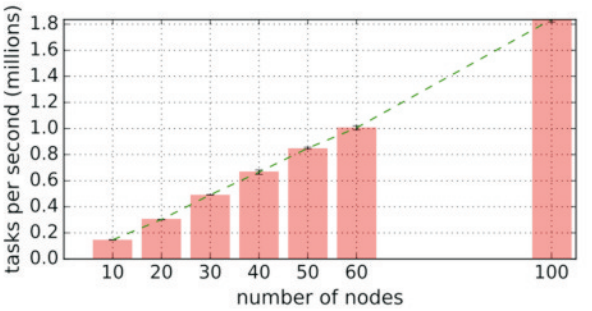



图3 RAY的吞吐率随着节点增加结果

图3显示了RAY在水平拓展能力，随着节点的增加，系统吞吐率呈现了线性的增长，当节点达到100时，RAY实现了每一秒1800000任务的吞吐率。

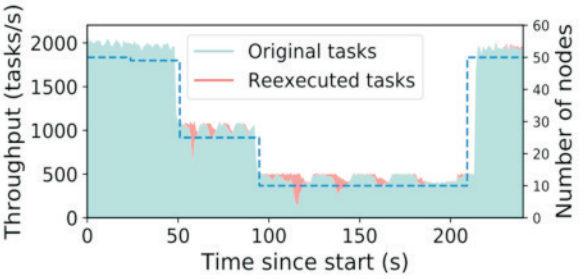



图4 RAY出现的节点失效的恢复

# Neural Architectures for Named Entity Recognition

桂向宇　推荐

“Neural Architectures for Named Entity Recognition”是人工智能领域顶级国际会议ACL2016录用的一篇文章。本文由Carnegie Mellon University与Pompeu Fabra University的NLP团队合作完成。该文章将神经网络引入到传统的命名实体识别（NER）任务中，在不使用外部先验知识库的条件下，使英文的NER的F1指标达到了与使用外部知识库相当的水平，并且遥遥领先于基于支持向量机等传统的识别方法。除了基于词向量的方法，论文还将句法处理中的transition model和神经网络相结合，利用堆结构，构建了shift-reduce的神经网络模型。该模型的优势在于它可以将name entity作为一个整体表示；而在传统的基于word embedding的方法中，一个实体不存在独立的表示，而是作为一些列词向量的组合。

Long Short-Term Memory (LSTM) 单元是递归神经网络RNN的一种特殊形式，它利用门控开关机制控制信息在神经网络中的保留和去除。随着信息的传递，有价值的信息被附加上的较高的权重，从而得到保留；而无价值信息在传播过程中将逐渐减弱甚至消失。LSTM较好的解决了传统RNN单元在长程依赖方面的不足，广泛应用于序列标注等任务中。命名实体识别任务也被看作是序列标注问题——对给定

接上页

图4显示了RAY从节点失效后的回复过程，节点在25秒，50, 100秒被人为关闭，在210秒后系统系统恢复完毕。

总结：总体而言，本文第一个提出了通用的强化学习平台RAY，极大地方便了AI开发者，因为上层Python里面实现动态，异构的任务是很难的。其系统创新性工作并不多，除了任务特性利用和编程模型设计，自低向上的调度方式适配强化学习任务动态本地生成的特点，避免全局调度瓶颈问题，容错技术沿用了spark里面的血统机制。虽然如此，个人认为，RAY应用不仅仅于强化学习，对于一般数据流处理，图处理，都很好的应用和借鉴意义。目前系统开源且很粗糙，其性能优化也将是未来的一个研究点。

吴文超
2016级博士生
研究方向：深度学习系统优化，内存计算
Email：wcwu@hust.edu.cn

的文字序列进行标注，使用BIO或者BIOES的标注方式。其中，B代表实体的begin，I代表inner，O代表out，E代表end，S代表single entity。通过该类标注，可以将序列中的实体抽取出来。

然而，处理序列标注问题的经典方法有条件随机场（CRF）等基于概率图模型的方法，这些方法均把序列看作有向图或无向图，根据图的依赖关系定义一定的特征并训练，从全局最优的角度找到合适的标注方式，这些方法在语音识别中也发挥了较大作用。另外基于支持向量机（SVM）的方法同样需要定义一系列语言特征，从而学习出最优模型。这些方法无不依赖于人工定义的特征质量，而这些特征的定义需要一定的语言学功底，比如前后缀关系、字符大小写特征、关键词词库等。

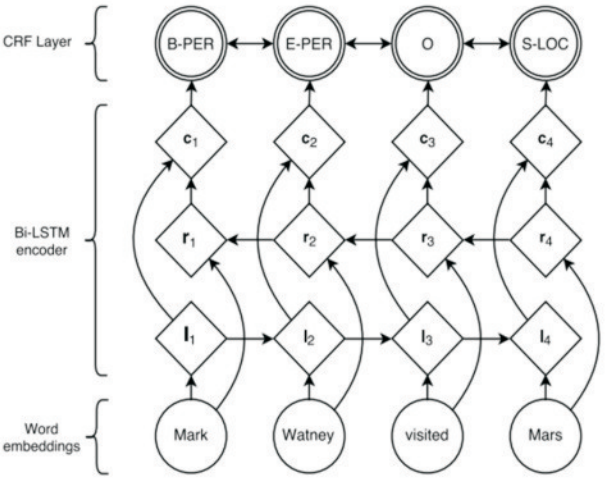



图1 BiLSTM-CRF 神经网络结构

一般认为神经网络是一个较好的特征提取器，它可以较为方便的提取出数据中的特征，减轻了特征工程的工作。论文将神经网络与传统概率模型相结合，形成了LSTM-CRF神经网络，如图1所示。

底层是词向量层，通过对提前训练好的词向量进行查找，将每个单词替换成向量形式。其中较为有名的词向量训练方法有word2vec，Glove 方法，近期 Facebook 提出了 FastText 方法。词向量的分布式表示能较好的表示词语的语义及前后位信息，有助于提高任务的精度。Bi-LSTM层作为特征提取器提取出语句中的信息。双向神经网络有助于减少对近邻信息的依赖偏移。CRF层定义了状态转移特征矩阵，避免了从“O”到“I”这种标注的出现。

本文在CoNLL2002和2003的数据集上进行实验，包括四种语言的NER任务，该模型均取得了最佳或接近最佳的成绩。这表明LSTM-CRF模型具有较好的跨语言识别的泛化能力。其中的S-LSTM模型尽管没有达到最好成绩，但是它将transition-based model和现有的模型进行融合，十分具有新意。同时，该模型对entity的向量化表示带来了启发，如何进行实体的向量化表示成为后续工作的关键。目前该文章已经被引用500多次，是实体识别的经典之作。

桂向宇
2017级博士生
研究方向：知识图谱
Email：guixy@hust.edu.cn

# Data Tiering in Heterogeneous Memory Systems

王孝远　推荐

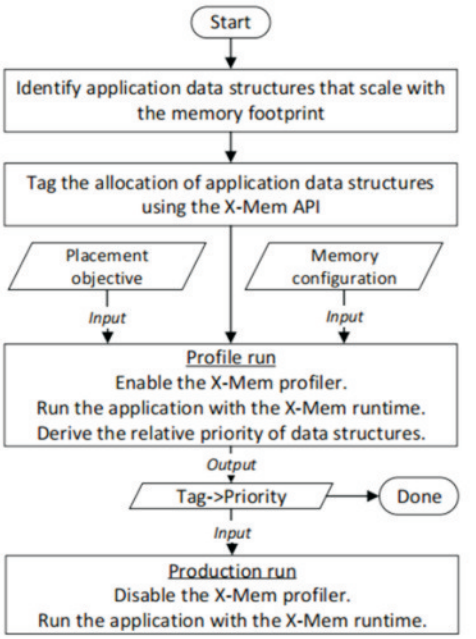
“Data Tiering in Heterogeneous Memory Systems”是体系结构领域的国际顶级会议EuroSys 2016录用的一篇文章，该会议于2016年4月在英国伦敦举行。

数据中心应用程序数据集呈指数型增长，由于主存容量大小，把工作负载分放到磁盘中降低了程序的性能，这通常是难以接受的。面对这个挑战，工业界提出了大容量，性能和花费介于DRAM和辅存之间的非易失存储技术。NVM的性能如下表所示：

| Parameter | DDR-DRAM | NVM |
|---|---|---|
| Capacity per CPU | 100s of GBs | Terabytes |
| Read Latency | 1× | 2× to 4× |
| Write bandwidth | 1× | $\frac{1}{8}$× to $\frac{1}{4}$× |
| Estimated cost | 5× | 1× |
| Endurance | $10^{16}$ | $10^{6}$ to $10^{8}$ |

数据中心运营商和用户关心使用NVM代替DRAM是否会降低程序的性能和带来较大的开销。回答这个问题对于实现和采用这种系统至关重要。

这篇文章针对平型架构下的混合内存系统，提出了一种数据分类和放置策略，来更好的在不同类型的存储介质中分配应用程序数据结构，已达到程序性能的提升。首先设计和实现了一个内存管理系统X-Men，然后利用分析工具自动化识别对数据结构访问模式，自动化映射内存区域到内存类型来优化性能，具体介绍如下：




## X –Men设计

X -Men可以通过把数据结构自动放置到合适的内存类型上来提高程序性能。为了达到这个目的，它必须要能区分属于不同数据结构的对象。 X-Men为程序员提供了拓展的memory allocation接口，可以显式的为每个数据结构在分配时指定一个标签，从而为运行时提供必要的信息把每个对象映射到所属的数据结构。其流程图：

程序开始运行前的准备工作包括两部分：（1）为每个数据结构附加上标签。（2）自动分类数据结构，分析应用程序，得到放置每个数据结构到DRAM的收益，该部分的输出是一

个tag-to-priority集。

X-Men为用户提供了一些API。分配程序除了增加了一个数据分类器（即tag）类似于标准的allocator API。Xmem-priority使应用可以显示的指定每个数据结构的优先值，优先值高的先被放置到DRAM中去。

```
void* xmalloc(int tag, size_t size);
void xfree(size);
void xmem_priority(int tag, int prio);
```

## 内存访问时间模型

X-Mem支持自动分配虚拟内存区域到实际的物理内存类型来优化程序性能。最佳的放置策略是基于下面讨论的一个内存访问时间模型。

程序平均内存访问时间为：$A = \frac{\sum_{i=1}^{n} C_i \times S_i}{\sum_{i=1}^{n} C_i}$

其中N为程序中的数据结构个数，Si为访问数据结构的延迟，Ci为访问数据结构i的次数。

现在超标量处理器中，访问延迟不等于内存访问时间。处理器把多个内存请求放在指令流中，利用乱序执行并行运行可以隐藏一部分访问延迟。因此，对于一些特定的访存模式内存访问有效延迟比实际物理内存访问延迟要低。

下表是对内存访存模式的一个分类：

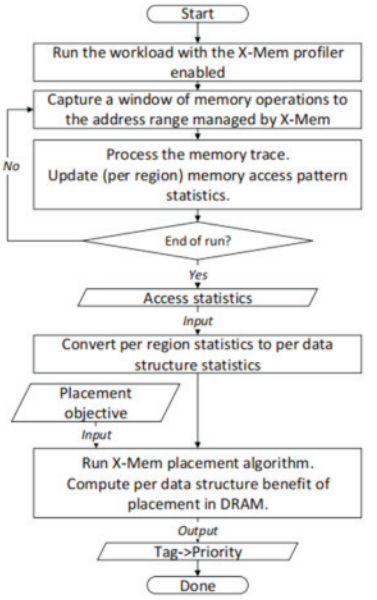
| | Dependent | Independent |
|---|---|---|
| Sequential | NA | Streaming |
| Non-sequential | Pointer chasing | Random |

## 分析工具

首先 使用一个分析器去确定每个数据结构不同访存模式的频度。然后，在批量运行时，利用分析数据去维护一个按优先级排列内存区域链表，来指导数据放置到DRAM中去。

对于每个数据结构，分析工具把访存分为random，pointer和streaming中的一种，并且统计每种类型的数量。如果第一个访存地址中的值等于第二个访存的地址，则为指针类型。通过一个滑动窗口来检测流式访问。并且统计总的访存数量，除去这两种模式就是随机访问的数量。

在运行时，尽可能多的把优先级链表中的内存区域映射到DRAM中，然后把剩余的区域放置到慢速的NVM中去。




最后文章考虑了三种类型的典型应用程序：高性能图计算框架GraphMat，内存数据库VoltSDB，key-value存储MemC3来对系统进行评估。利用文章提出的数据放置策略，程序性能可以提高22%到76%，同时，较纯DRAM内存系统，仅仅带来了13%到40%的性能降低。

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

# 追逐皇冠上的明珠

刘　博

如果用金灿灿的皇冠来比喻研究者们对科学的憧憬和向往，那么皇冠上那颗独一无二又光彩照人的明珠，则是我们每一个学术人心目中永远的梦想。

回想2015年进入实验室以来这近4年的时光，那么短又那么长。短是因为目睹着优秀文献迅雷般的出现与更替交迭，感受到同窗好友们一一毕业时不舍的友情，长是因为体会到工程实验的复杂与漫长周期，经历了论文撰写与修改投稿时的辗转难待。

刚刚提起笔时那股澎湃的心动感，渐渐被回顾往事自己却碌碌无为的羞愧所掩盖，我决定记录下自己成长的体会，希望能对后来的学弟学妹抛砖引玉，发挥余热。

## 那么短，时不我待

当我选择踏入深度学习系统研究这个领域时，就被其深深的吸引住。从硕士时做一些简单的算法改进，一大步跨越到开始深入思考如何提升系统的资源利用率，以及计算和通信等性能，我如同一个婴孩般蹒跚学步的前行。回想当时每一次和陆路师兄等同门的交流讨论，都觉得时间那么短，宛如饥渴的旅人汲取着他们身上的智慧和经验。

当我初到武汉加入实验室这个大家庭，很多生活和工作的细节都让我有些忙乱不知所措。这时候周围伙伴好友都伸出了温暖的双手，给我许多关怀，每次周末和徐慧等同门的聚餐闲聊，平抚了许多思乡孤单的小情绪。每个毕业季，当依依告别即将毕业的小伙伴，则是我最不舍的时刻，只嫌相聚相知总是那么短。

## 那么长，此日可待

当我选择从事科研学术这条道路，难免遇到各种困难。当理论设想思路纷乱时，当实验结果不如预期时，当彻夜修改提交时，当相关工作先于自己问世时，当论文投稿被拒时，这些个心痛难捱的时刻，都会让我觉得未来如此漫长而黑暗。每到这时，我总会鼓励自己坚持下去，始终相信美好的事情即将发生。

当我在完成基本研究工作的同时，经常会陷入自我怀疑的泥潭，是否idea不够fundamental，是否实验结果不够competitive，是否论文level达不到顶级会议期刊的要求。特别是在强者如林的实验室，更是常常怀疑自己天资愚钝不适合科研。每到这时，我也总是告诫自己，继续积累，耐心等待。

## 追求皇冠上的明珠，是对自我的严格要求

当我开展自己的研究时，常常会因为种种困难和原因，而不自觉地降低对文章质量的追求，企图通过达到毕业标准来模糊普通跟优秀的界限。每到这时我会告诫自己，学术道路道阻且长，只有对自我有着严格的要求和管理，才能够最终有幸摘得皇冠上的明珠。

刘　博
2015级博士生
研究方向：深度学习系统
Email：80937591@qq.com