1st CCF CHIP
中国计算机学会芯片大会

超异构处理器设计
技术论坛

面向异构硬件体系
系统软件支撑和优化技术

陈 榕

并行与分布式系统研究所
上海交通大学

2022.7.29

研究合作者：魏星达、谢夏婷、韩明聪、张汉泽、杨健邦、唐大海、宋小牛、陆放明、陈海波、以及IPADS团队、阿里巴巴达摩院等

IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

提纲

- 异构硬件体系与系统软件研究
- 研究路径 #1：异构硬件聚合
- 研究路径 #2：应用特征反哺

提纲

- **异构硬件体系与系统软件研究**
- 研究路径 #1：异构硬件聚合
- 研究路径 #2：应用特征反哺

异构硬件体系：新趋势

通用硬件体系发展受限，异构硬件体系带来新机会

- 算力：CPU + GPU / TPU / FPGA
- 网络：Ethernet + RDMA / DPU
- 存储：DRAM + NVM / NVMe SSD

高性能

高并发、大算力

高带宽、低延迟

内存级、大容量

新功能

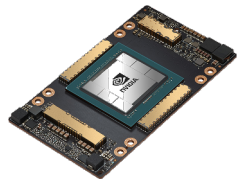
事务性执行

独立访存

数据持久化

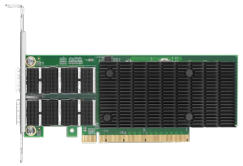
异构服务器

异构数据中心



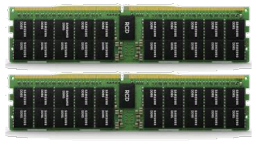
GPU
NVIDIA A100
312 TFLOPS

CPU
Intel Xeon
2x 48 cores
w/ **HTM**

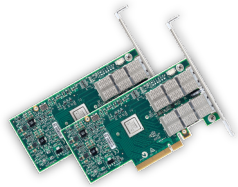


Ethenet
100/200 Gbps

NVM
Intel Optane
768GB



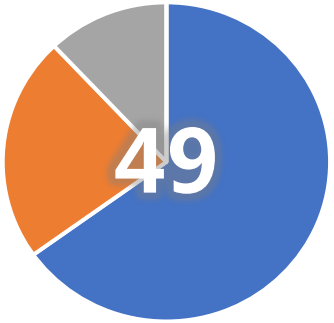
DRAM
DDR5
512 GB

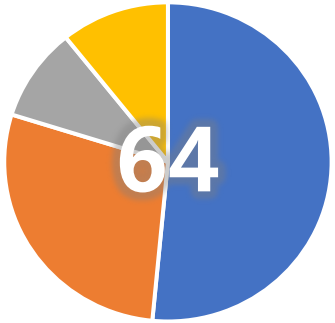


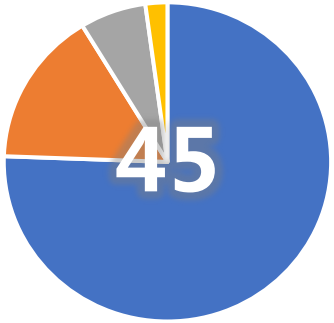
IB CX6
2x 200 Gbps
w/ **RDMA**

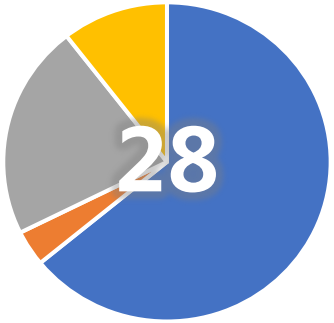
异构硬件 ⊕ 系统研究

计算机系统会议：OSDI、ATC、EuroSys、FAST @2022









CSRankings

Computer Science Rankings

▼ Operating systems

ACM SIGOPS, USENIX

OSDI

SOSP ← 双数年未召开

EuroSys

FAST

USENIX ATC

OSDI	ATC	EuroSys	FAST	
34.7%	**48.4%**	**24.4%**	**35.7%**	← **异构硬件相关论文占比**
0	7	1	3	网络：RDMA/SmartNIC/P4 . . .
6	6	3	6	存储：NVM/CSSD/NVMe . .
11	18	7	1	算力：GPU/TPU/TEE/RISC-v . .

系统软件研究：新角色

研究方向 #1：寻找目标场景、兑现性能提升

- GPU（高并发、大算力）：深度学习训练/推理、大数据处理、图计算
- RDMA（高带宽、低延迟）：高性能通信、Remote/Far memory
- NVM（内存级、持久化）：文件系统、存储系统、数据库、系统缓存

研究方向 #2：新硬件赋能、软件功能硬件化

- HTM（事务性执行/ACI）：Transactions/OLTP、并发控制、VM Introspection
- TEE/SGX（系统隔离）：可信计算、Sandbox、区块链共识、去中性化计算
- RDMA/SmartNIC（独立访存/计算）：Replication、负载均衡、pipelining

系统支撑面临挑战

硬件：瞄准单一需求

案例：

RDMA

提供低时延跨节点内存读写，但无法保证多个操作的原子性，以及写入数据的持久性

HTM

提供事务性读写，但仅限于单节点内且访存数量受限、操作不可被中断

NVM

提供数据持久性但受限在单节点内，且依赖于特定指令和不同访存粒度

GPU

跨节点传输依赖于CPU和系统内存，且并发任务调度难共享、难抢占

系统支撑面临挑战

应用：需求丰富多样

案例：

分布式事务

支持跨节点事务，支持并发内存读写操作的原子性、一致性、隔离性、以及持久性等

高可用日志

实现跨节点的日志备份(WAL)，支持系统高可用和日志数据的持久化

远程索引缓存

分布式有序存储在客户端的索引缓存需要在保证数据一致性的同时减少网络开销

实时智能任务

需要大算力硬件以提供计算加速，同时也需要快速任务抢占满足高时效需求

系统支撑面临挑战

硬件：瞄准单一需求 VS 应用：需求丰富多样

RDMA 

HTM 

NVM 

GPU 

使用场景受限

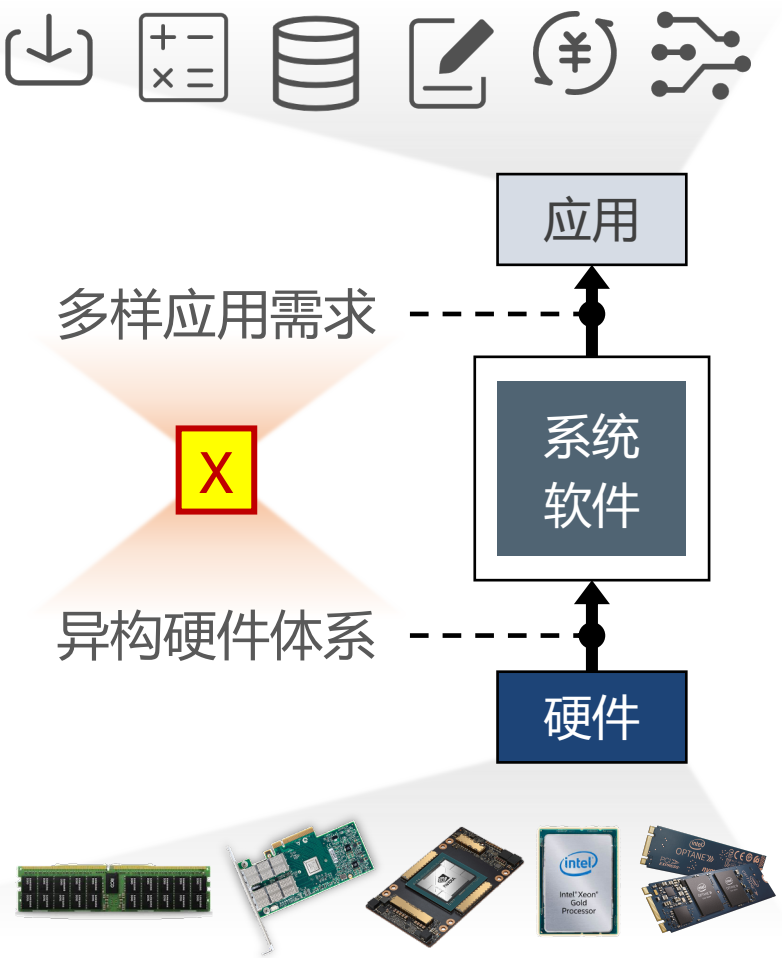
性能获益受损

- 分布式事务
- 高可用日志
- 远程索引缓存
- 实时智能任务

系统软件研究：新路径

基础系统软件栈：承上启下

- **难点**：**复合**需求、**互斥**能力、**不匹配**供需
- **目标**：聚焦异构硬件，拓展适用场景、提升优化效果

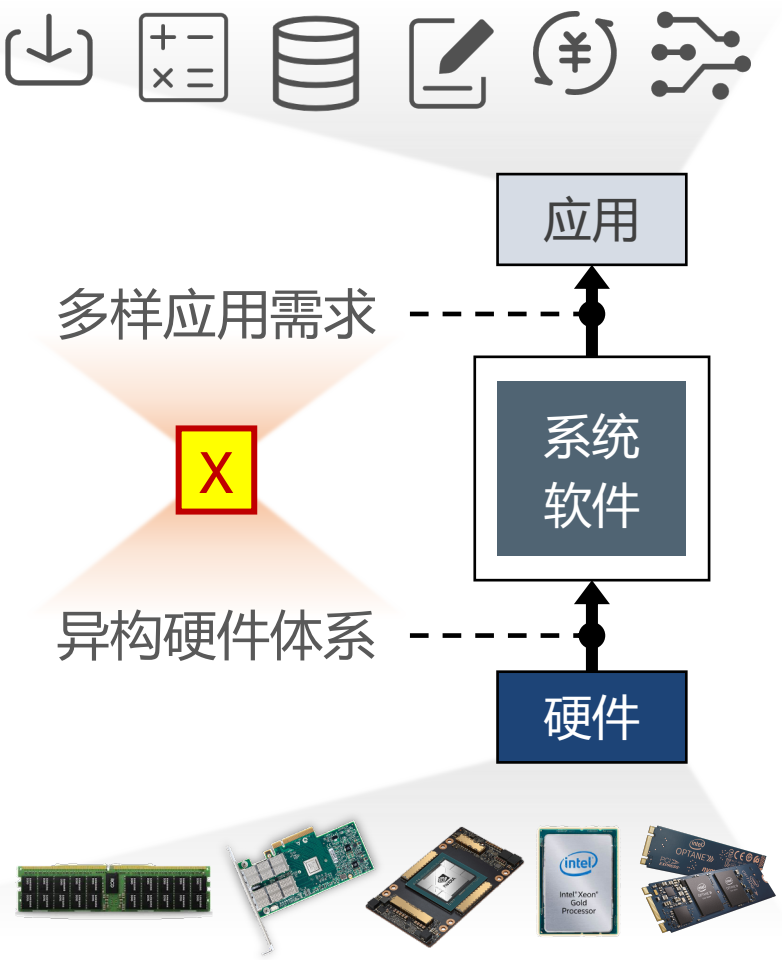


系统软件研究：新路径

基础系统软件栈：承上启下

- **难点**：**复合**需求、**互斥**能力、**不匹配**供需
- **目标**：聚焦异构硬件，拓展适用场景、提升优化效果

“路漫漫其修远兮，吾将上下而求索”



路径 #1：异构硬件聚合【向下求】

- **思路**：组合多种硬件特性，满足复合应用需求

路径 #2：应用特征反哺【向上求】

- **思路**：挖掘领域应用特征，突破传统设计限制

提纲

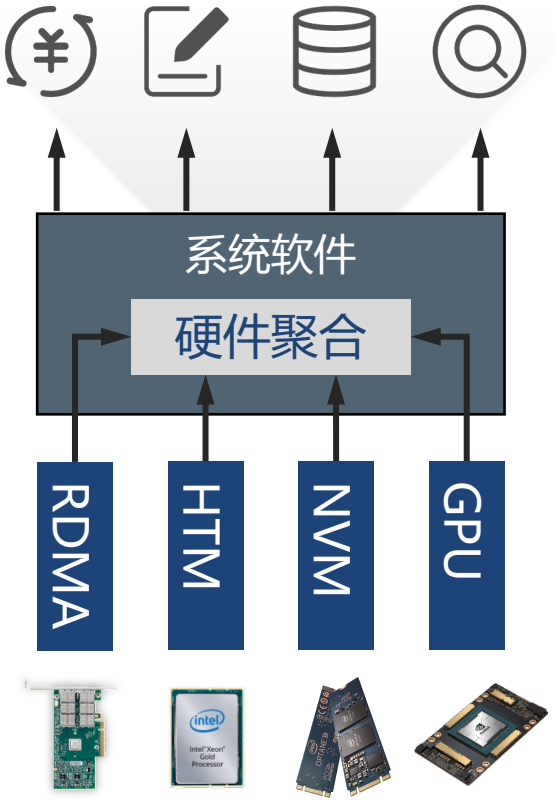
- 异构硬件体系与系统软件研究
- **研究路径 #1：异构硬件聚合**
- 研究路径 #2：应用特征反哺

异构硬件聚合

组合多种硬件特性，满足复合应用需求

- 以网络硬件RDMA为核心，将硬件事务性内存HTM、高并发处理单元GPU、非易失性内存NVM等异构硬件在原语层面进行聚合、创造“逻辑新硬件”克服单一功能局限性

硬件	原语 (primitive)
RDMA	read | write | cas | xadd | send | recv | …
HTM	xbegin | xend | xcommit | xabort | …
NVM	clwb | sfence | pcommit | nt-store | ...
GPU	load | store | … (on 128-512 bits, SIMD)



案例 #1-1：分布式事务处理

基于RDMA的事务处理研究

- 代表性工作：FaRM[@SOSP15/SIGMOD19]，FaSST[@OSDI16] 等
- 使用RDMA实现软件并发控制协议，包括OCC[@SOSP15/OSDI16] 等
- 提出versioning等机制确保RDMA访存一致性【性能受损】

RDMA

提供低时延跨节点内存读写，但无法保证多个操作的原子性

基于HTM的事务处理研究

- 代表性工作：DBX[@EuroSys14]，Leis et al.[@OSDI16] 等
- 使用HTM实现软件并发控制协议，包括OCC[@EuroSys14]、TSO[@ICDE14] 等
- 性能接近细粒锁实现，且仅限于单机内事务执行【功能受限】

HTM

提供事务性读写，但仅限于单节点内且访存数量受限、执行不可被中断

案例 #1-1：分布式事务处理

HTM: Hardware Transaction Memory

a non-transactional code will unconditionally abort a transaction when their accesses conflict

Strong Atomicity

案例 #1-1：分布式事务处理

HTM: Hardware Transaction Memory

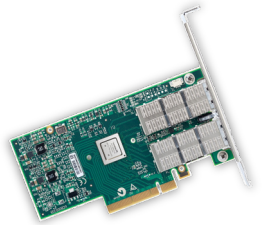
> *a non-transactional code will unconditionally abort a transaction when their accesses conflict*

Strong Atomicity

RDMA: Remote Direct Memory Access

> *one-sided RDMA operations are cache-coherent with local accesses*

Strong Consistency



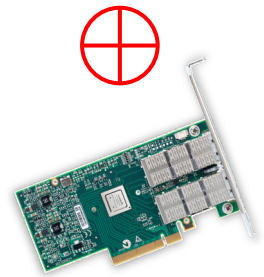
案例 #1-1：分布式事务处理

HTM: Hardware Transaction Memory

> *a non-transactional code will unconditionally abort a transaction when their accesses conflict*

RDMA: Remote Direct Memory Access

> *one-sided RDMA operations are cache-coherent with local accesses*



RDMA ops will abort conflicting **HTM** TX

Basis for **Distributed TX** processing

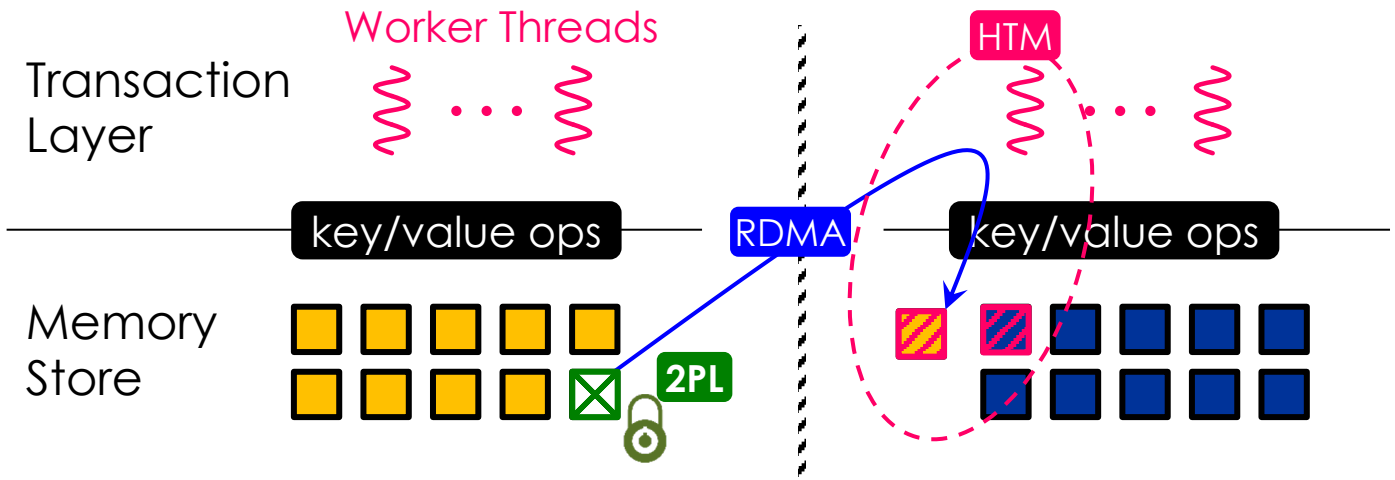
Distributed HTM

分布式硬件事务内存

案例 #1-1：分布式事务处理

DrTM：基于Distributed-HTM的分布式事务处理

- 基于RDMA的2PL协议：分布式事务(Distributed TX) → 单机事务(Local TX)
- 使用HTM直接实现单机事务处理
- 基于Distributed-HTM的高效分布式键值存储(DrTM-KV)



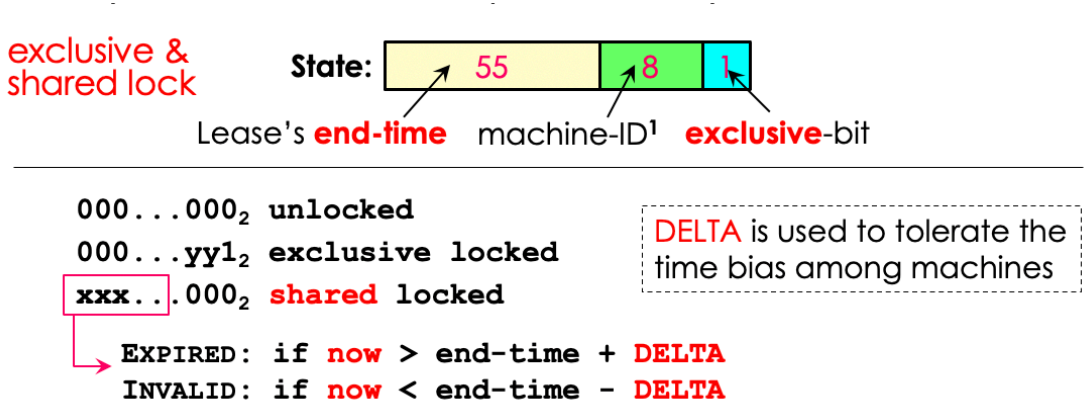
DrTM并发控制方法

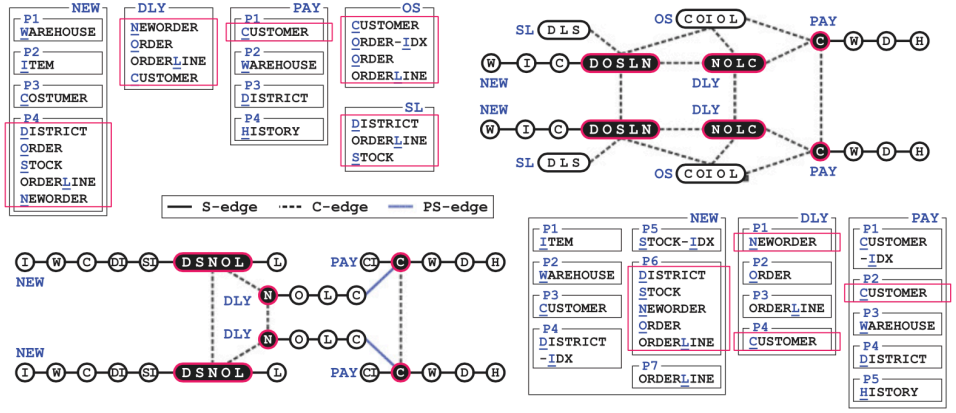
- **HTM**: L-TX vs. L-TX
- **RDMA**: L-TX vs. D-TX
- **2PL**: D-TX vs. D-TX

案例 #1-1：分布式事务处理

DrTM：基于Distributed HTM的分布式事务处理

- **挑战 #1**：基于RDMA有限硬件原语实现分布式读写锁
- **技术 #1**：采用基于租约(Lease)的RDMA分布式共享锁实现
- **挑战 #2**：使用访存数量受限的HTM直接实现事务处理
- **技术 #2**：采用 TX chopping 技术实现事务的按需切分

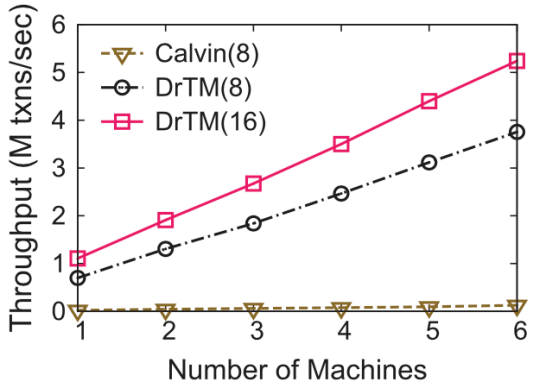


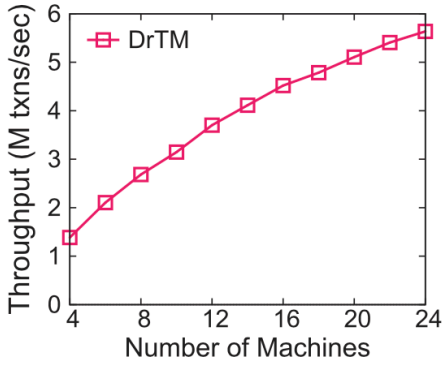


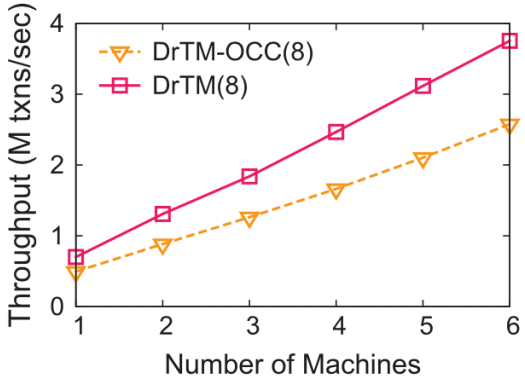
案例 #1-1：分布式事务处理

DrTM：基于Distributed HTM的分布式事务处理

- 6节点RDMA集群上，相比传统最优系统(Calvin)，吞吐量达到 5.52 百万TPC-C事务每秒，性能提升20倍，平均时延降低下降2个数量级
- 相比基于单纯RDMA的实现(OCC协议)，吞吐量进一步提升50%，以及更好的扩展性







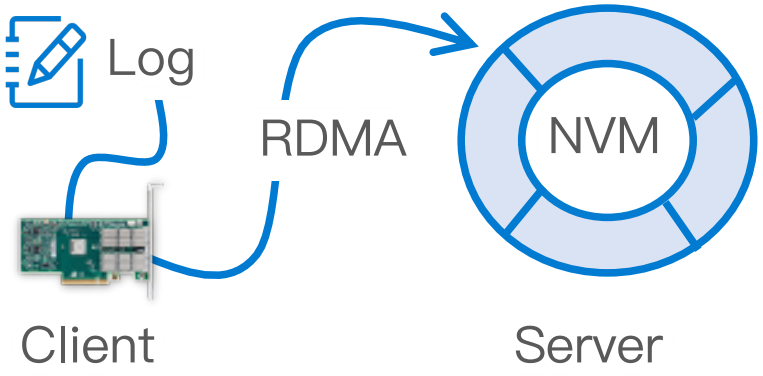
		w/ logging
Standard-mix (txns/sec)		3,243,135
New-order (txns/sec)		1,459,495
Latency (μs)	average	15.02
	50%	7.02
	90%	30.45
	99%	91.14
Capacity Abort Rate (%)		43.68
Fallback Path Rate (%)		14.80

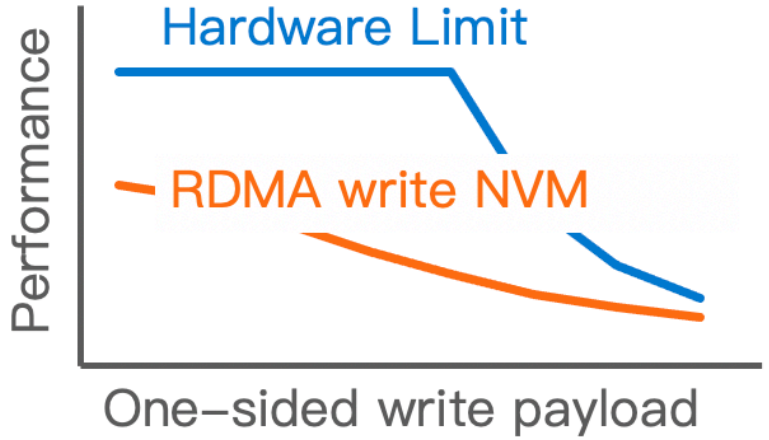
研究成果发表在 SOSP、EuroSys、ACM TOCS等系统领域标志性会议和期刊

案例 #1-2：高可用系统日志

基于RDMA+NVM实现高可用系统日志

- 代表性工作： Octopus[@ATC17]，Orion[@FAST19], AsymNVM[@ASPLOS20] 等
- 利用RDMA和NVM优化分布式系统高可用性，如文件系统，数据库等
- 仅考虑了RDMA+NVM系统正确性，未完全释放性能红利【性能受损】





RDMA

提供低时延跨节点内存读写，但无法保证多个操作的原子性和持久化等

NVM

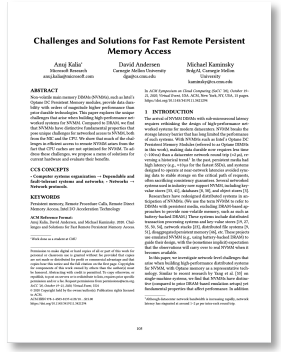
提供数据持久性但受限在单节点内，且依赖于特定指令和不同访存粒度

案例 #1-2：高可用系统日志

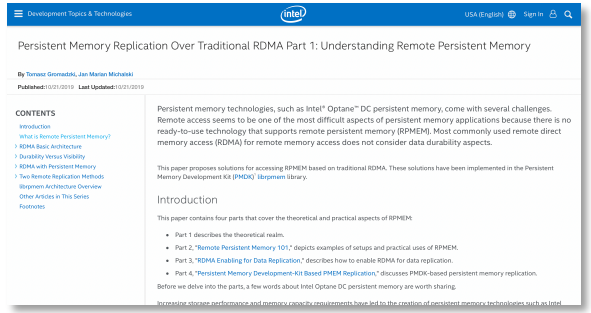
量产型 NVM 硬件特性 (e.g., Intel Optane PM)

- 与DRAM，以及各类模拟器的结果有较大差异
- NVM性能研究： Yang et al.@FAST20 ，Kalia et al.@SOCC20 等
- Intel 白皮书中仅给出了RDMA操作在NVM上的持久化方法
- RDMA与NVM交互方式复杂，存在许多性能陷阱

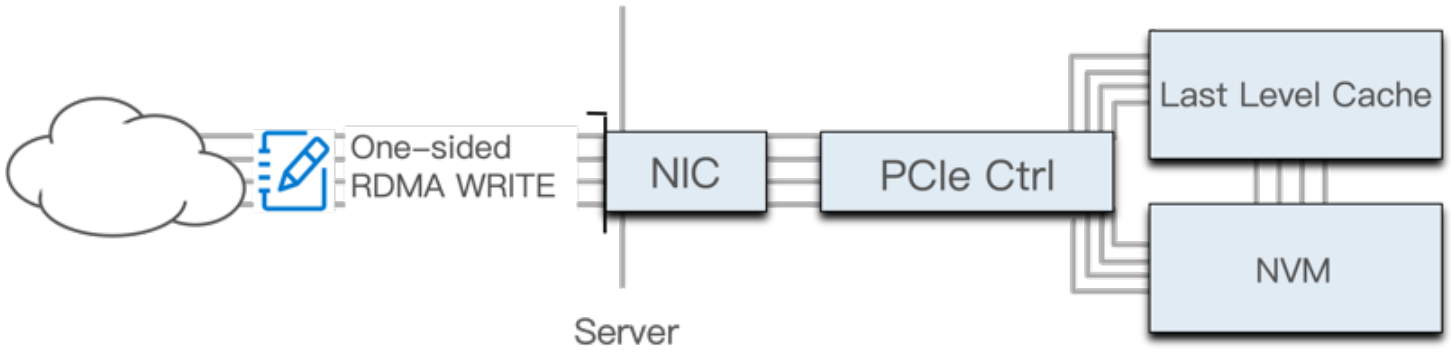
An Empirical Guide to the Behavior and Use of Scalable Persistent Memory



Challenges and Solutions for Fast Remote Persistent Memory Access



Persistent Memory Replication Over Traditional RDMA Part 1: Understanding Remote Persistent Memory



案例 #1-2：高可用系统日志

系统性研究 RDMA+NVM 的性能及优化方法

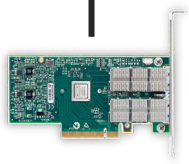
- 总结三方面优化建议【9 Hints】，并提出优化新方法
- 在现有RDMA+NVM系统上验证优化建议和方法有效性

  文件系统：Octopus@ATC17、数据库：DrTM+H@OSDI2018

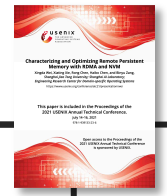
Optimization Hints

study

RDMA



NVM



optimizing

Octopus

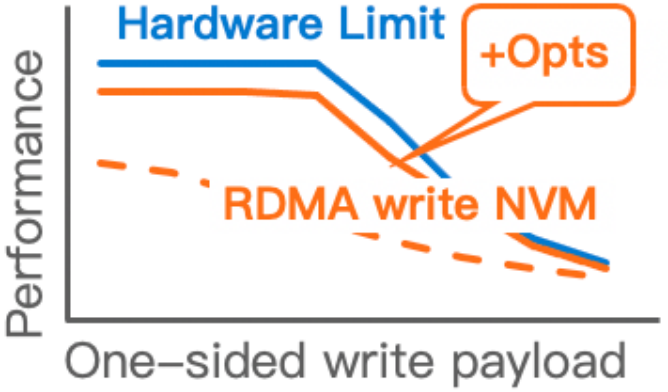
DrTM+H

案例 #1-2：高可用系统日志

系统性研究 RDMA+NVM 的性能及优化方法

- 总结三方面优化建议【9 Hints】，并提出优化新方法
- 在现有RDMA+NVM系统上验证优化建议和方法有效性

  文件系统：Octopus@ATC17、数据库：DrTM+H@OSDI2018



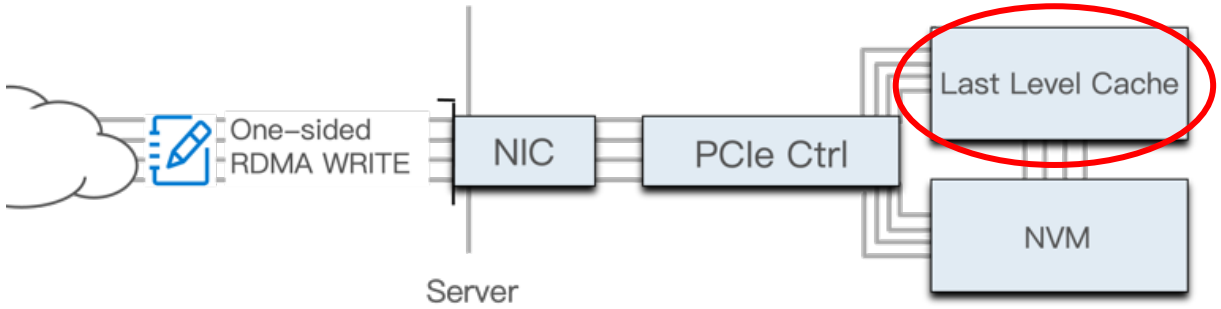
Optimization Hints
- Configuration (H1-H3)
- Access pattern (H4-H8)
- RDMA-aware (H9)

RDMA-NVM 优化建议 (H1-H9)
H1. 对于大数据传输使用NT-store
H2. 避免使用跨socket访问
H3. 避免以小于xpline粒度的访问
H4. 减少并发CPU NVM访问
...

案例 #1-2：高可用系统日志

RDMA+NVM 优化建议

- **H5**：关闭 DDIO 以绕过处理器缓存

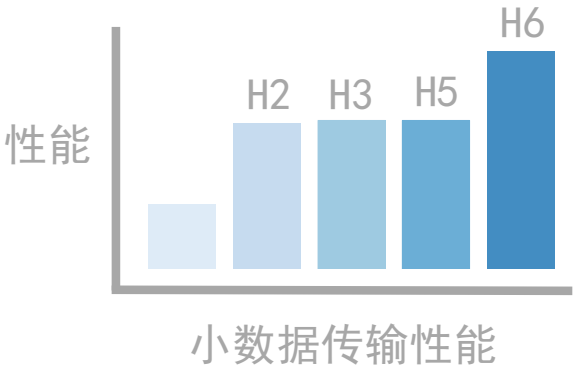


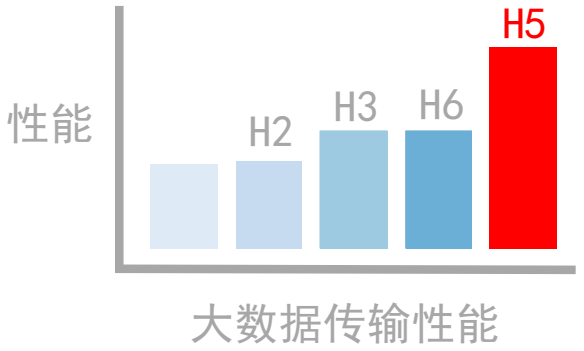
NVM 硬件特性

- 性能：顺序写 >> 随机写

默认 RDMA+NVM 行为

- DDIO 会优先将 RDMA 写入缓存
- 缓存以随机方式写会 NVM

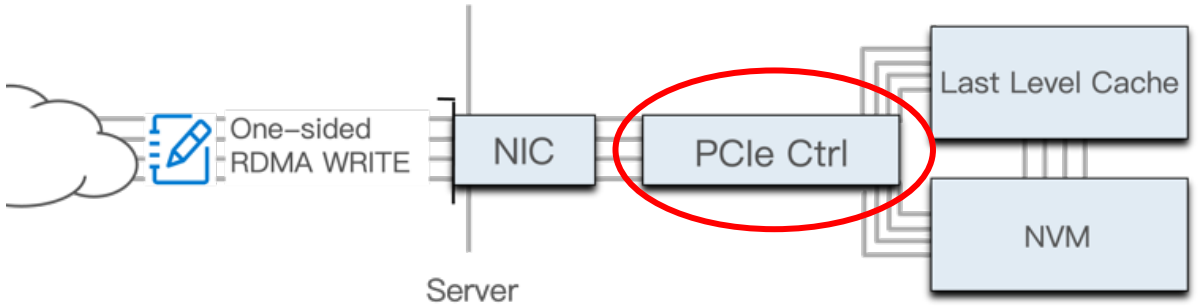




案例 #1-2：高可用系统日志

RDMA+NVM 优化建议

- **H6**：按 PCIe 最小传输粒度访问 NVM

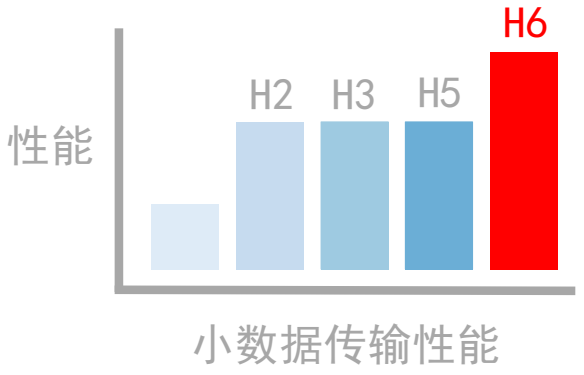


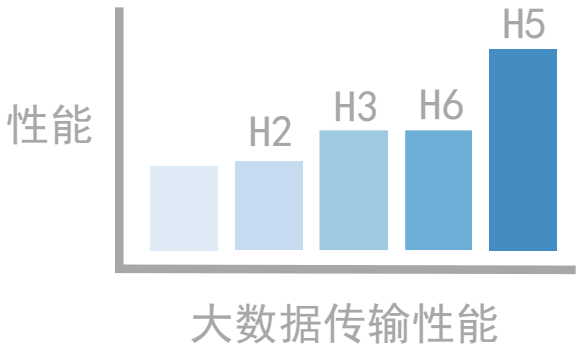
NVM 硬件特性

- NVM 读请求严重影响写请求

默认 RDMA+NVM 行为

- RDMA 请求不满足 PCIe 最小粒度
- PCIe 将写变成先读再写

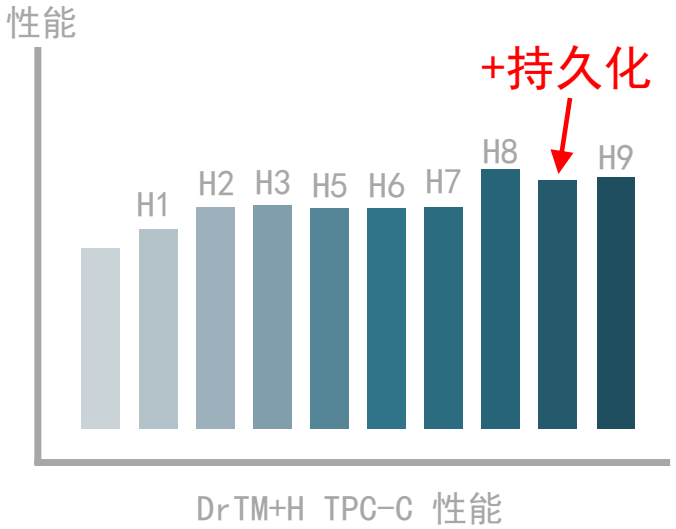




案例 #1-2：高可用系统日志

利用建议优化 RDMA+NVM 系统

- 分布式数据库 DrTM+H[@OSDI18]：**1.44X** @TPC-C 测试
- 分布式文件系统 Octopus[@ATC17]：**2.40X** @Data I/O 测试



RDMA-NVM 优化建议 (H1-H9)
H1. 对于大数据传输使用NT-store
H2. 避免使用跨socket访问
H3. 避免以小于xpline粒度的访问
H4. 减少并发CPU NVM访问
...
...

DrTM+H系统优化

1. 优化系统配置 H2,H5
2. 优化日志结构 H3, H6-H8
3. 优化日志持久化写入 H1, H9

研究成果发表在 Usenix ATC 等系统领域标志性会议

提纲

- 异构硬件体系与系统软件研究
- 研究路径 #1：异构硬件聚合
- **研究路径 #2：应用特征反哺**

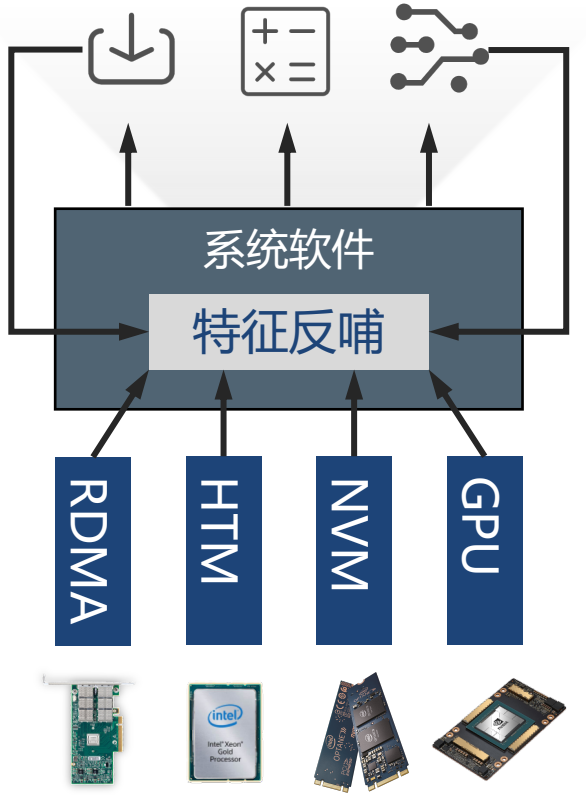
应用特征反哺

挖掘领域应用特征，突破传统设计限制

- 应用需求丰富多样，且往往具有复合性，对于利用异构硬件特性进行完美适配提出了巨大挑战，需要通过挖掘领域应用特征反哺系统软件设计，并结合方法创新与技术突破

远程索引缓存
- 吞吐量
- 利用率
- 存储开销

实时智能任务
- 大算力
- 强实时
- 高效率



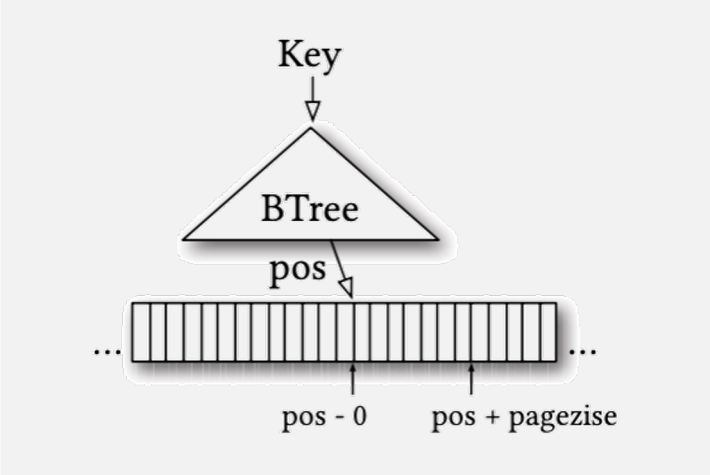
案例 #2-1：远程索引缓存

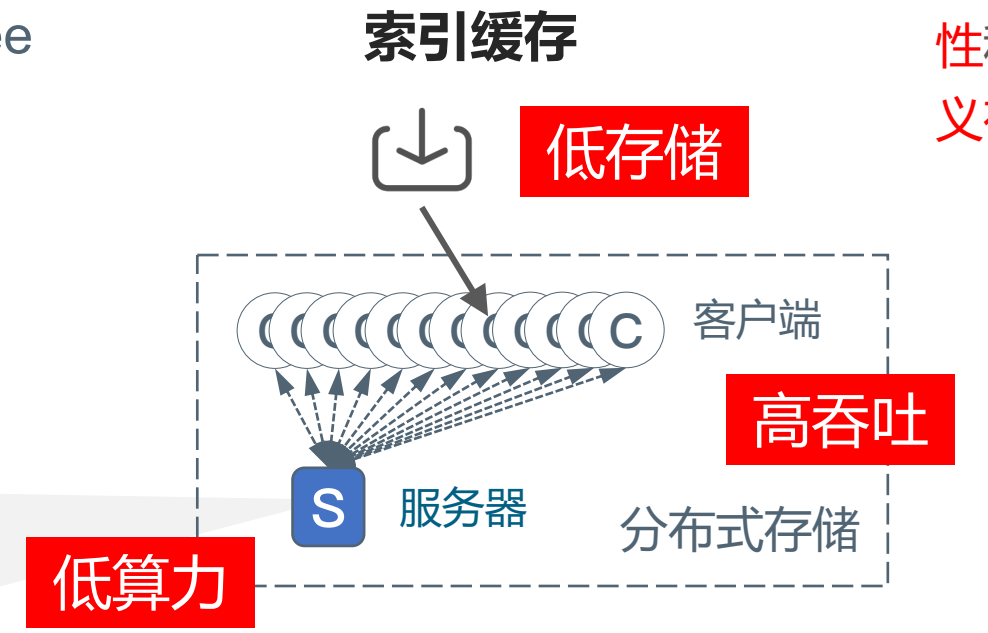
键值存储 (key-value store)

- 分布式应用的存储基础：OLTP/OLAP、文件系统、深度学习训练等
- 键值模型 (KV)

Model (key) → pos　// e.g., B+ Tree

KVS (pos) → value





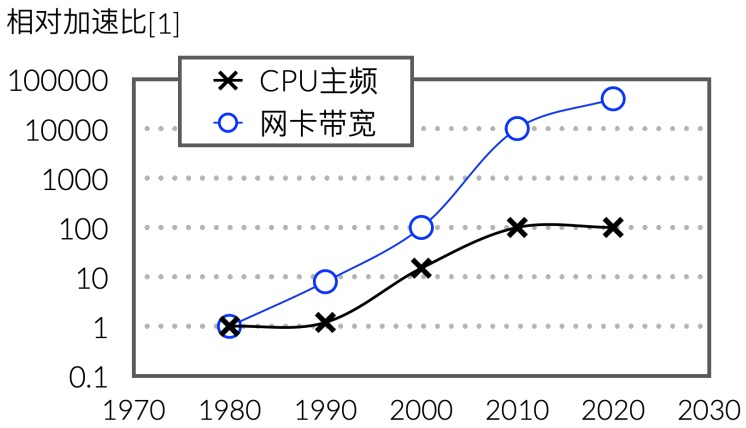
RDMA

提供低时延跨节点内存读写，但无法保证多操作的原子性和一致性，且语义有限：R/W/C/X

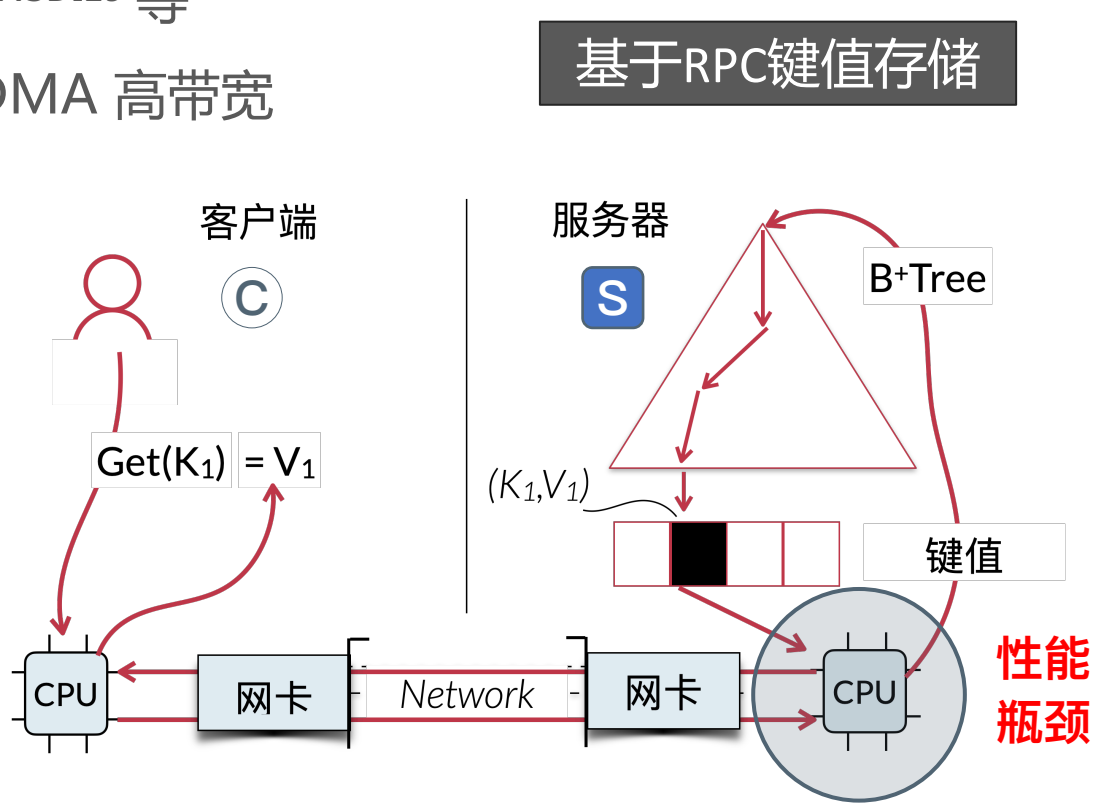
案例 #2-1：远程索引缓存

基于 Two-sided RDMA 键值存储设计

- 利用 RDMA (Send/Recv) 加速 RPC，eRPC[@NSDI19] 等
- 服务器 CPU 成为性能瓶颈，无法充分利用 RDMA 高带宽
- CPU 频率增长落后于 RDMA 网卡带宽增长



来自 StRoM @Eurosys20



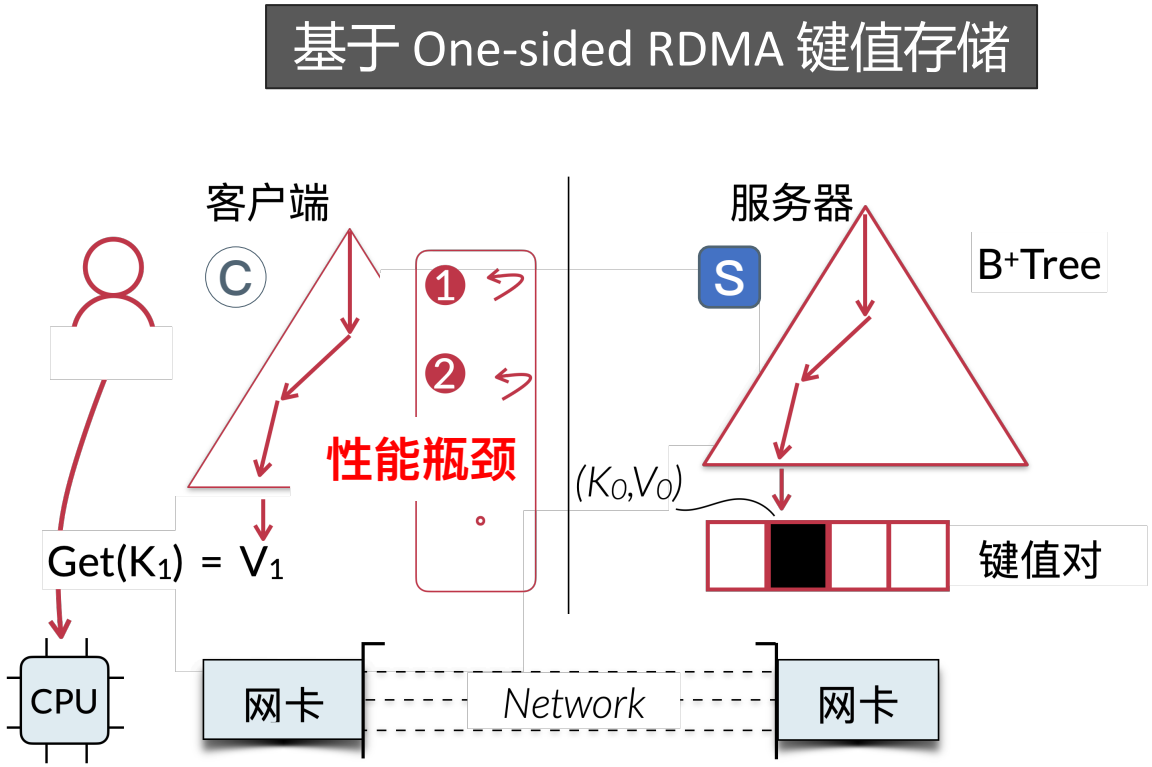
案例 #2-1：远程索引缓存

基于 One-sided RDMA 键值存储设计

- one-sided RDMA 语义有限：Read/Write
- 单次 RDMA操作仅能读取一层B⁺树节点
- 遍历B⁺树索引导致大量额外网络操作

分布式存储性能直接受限于网络操作数

- 1次键值操作需要$O(\log(n))$次网络操作
- 100M键值数据，1次get需要7次RDMA

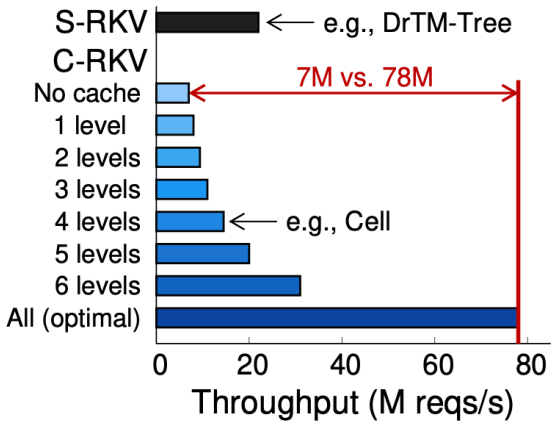


案例 #2-1：远程索引缓存

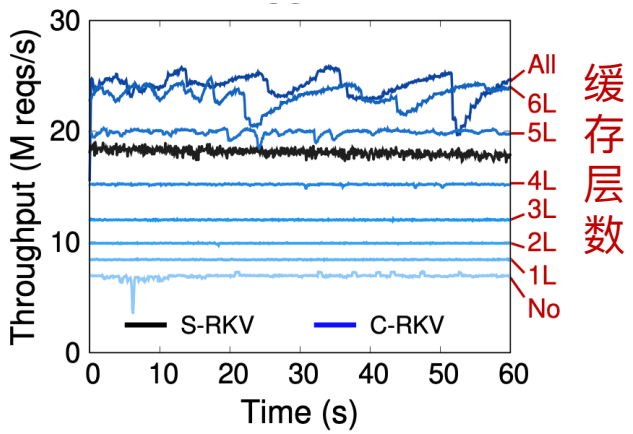
索引缓存技术：客户度缓存 (同构) 索引，降低遍历索引开销

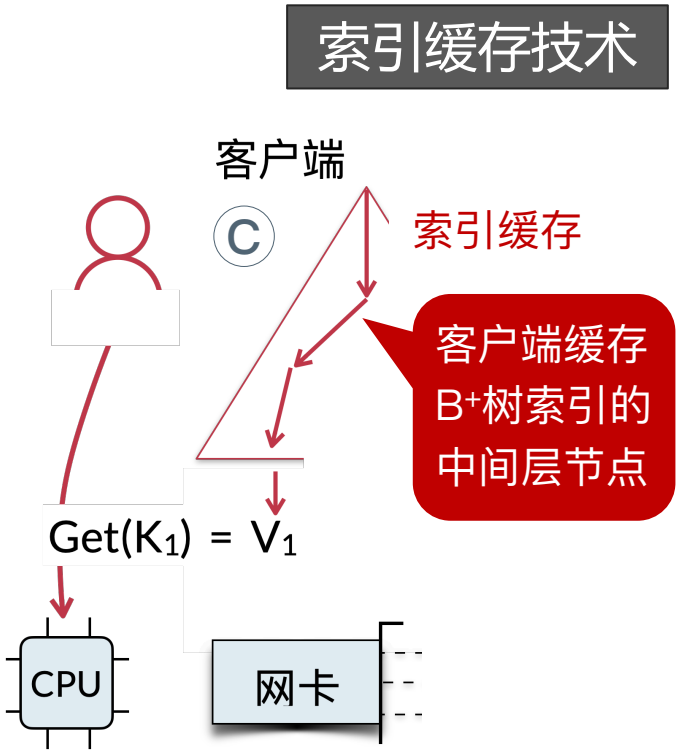
- 代表性工作：DrTM-KV[@SOSP15]、CELL[@ATC16]、FaRMv2[@SIGMOD19] 等
- 有序索引 (B⁺树) 不适合索引缓存j技术：
  1. 整树缓存客户端**内存开销巨大** (100M键值需要～650MB)
  2. 动态场景 (插入/删除) 造成缓存频繁失效，**性能颠簸**严重

静态负载



动态负载





案例 #2-1：远程索引缓存

① 基于 Two-sided RDMA 键值存储

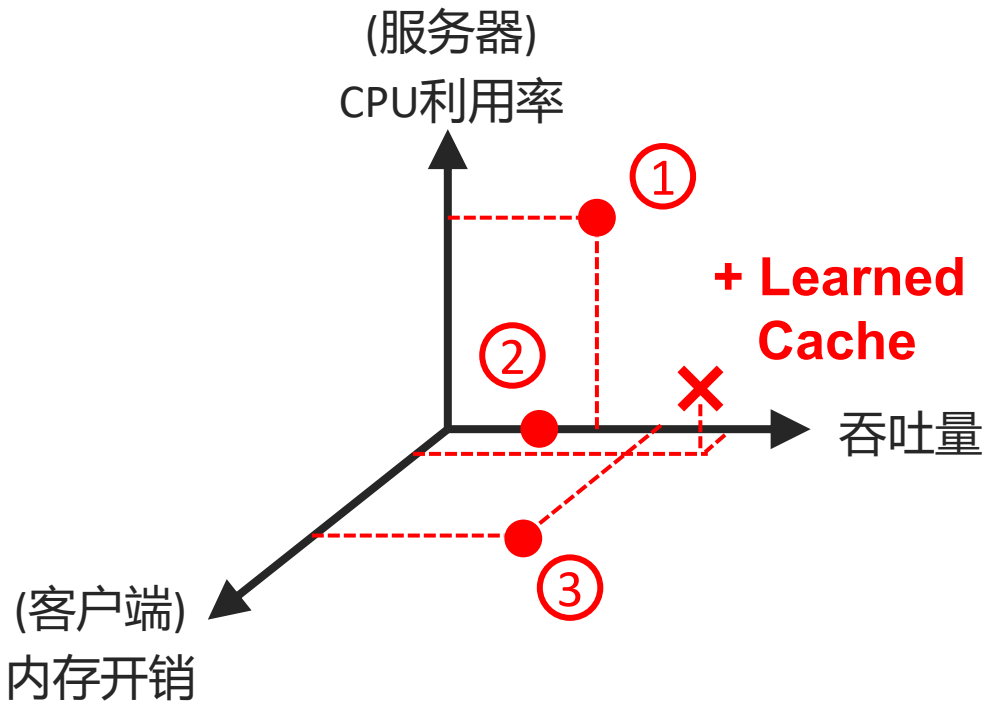
- 性能有限，且受限于服务器CPU

② 基于 One-sided RDMA 键值存储

- 性能有限，浪费RDMA网络资源

③ (B⁺树结构) 索引缓存技术

- 客户端内存开销大，动态负载性能颠簸

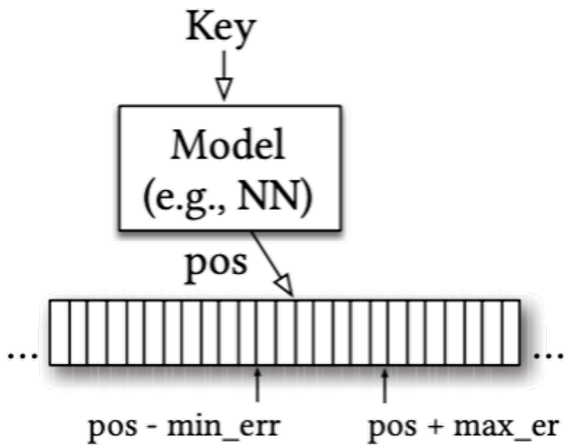


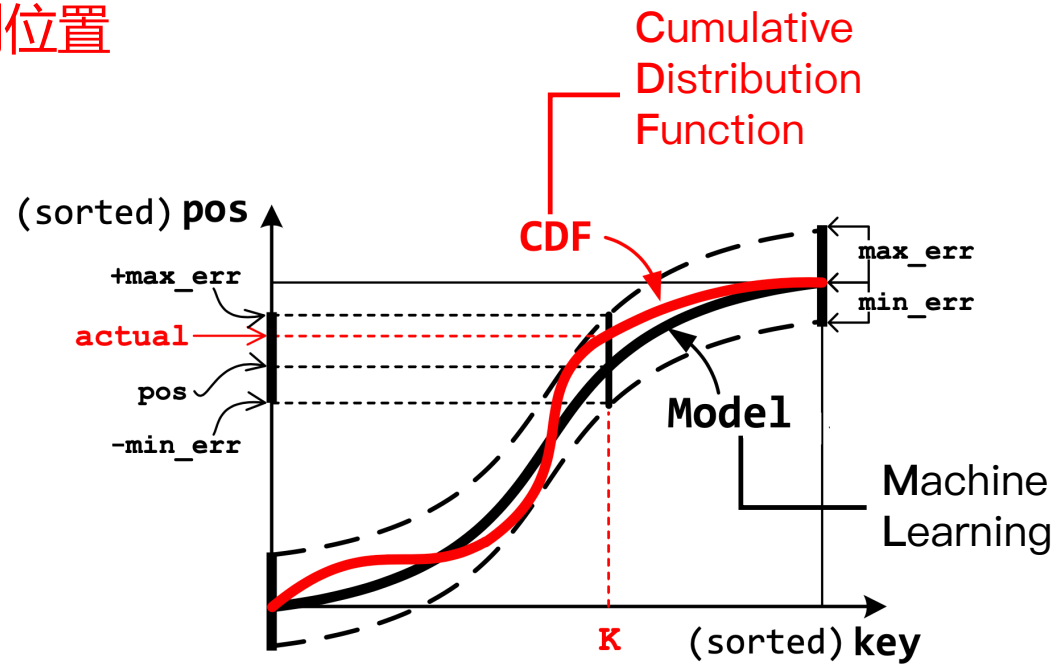
案例 #2-1：远程索引缓存

有序数据索引的特征

- **(键-位置) CDF分布**可以通过**ML模型训练逼近** (Learned Index@SIGMOD18)
- 使用机器学习 (ML) 模型 (代替B+树结构) 预测位置

  MLmodel (key) → pos // e.g., LR、NN

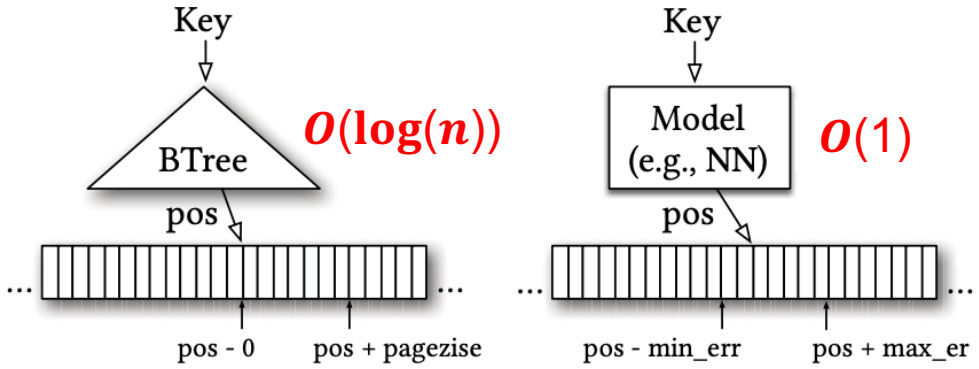


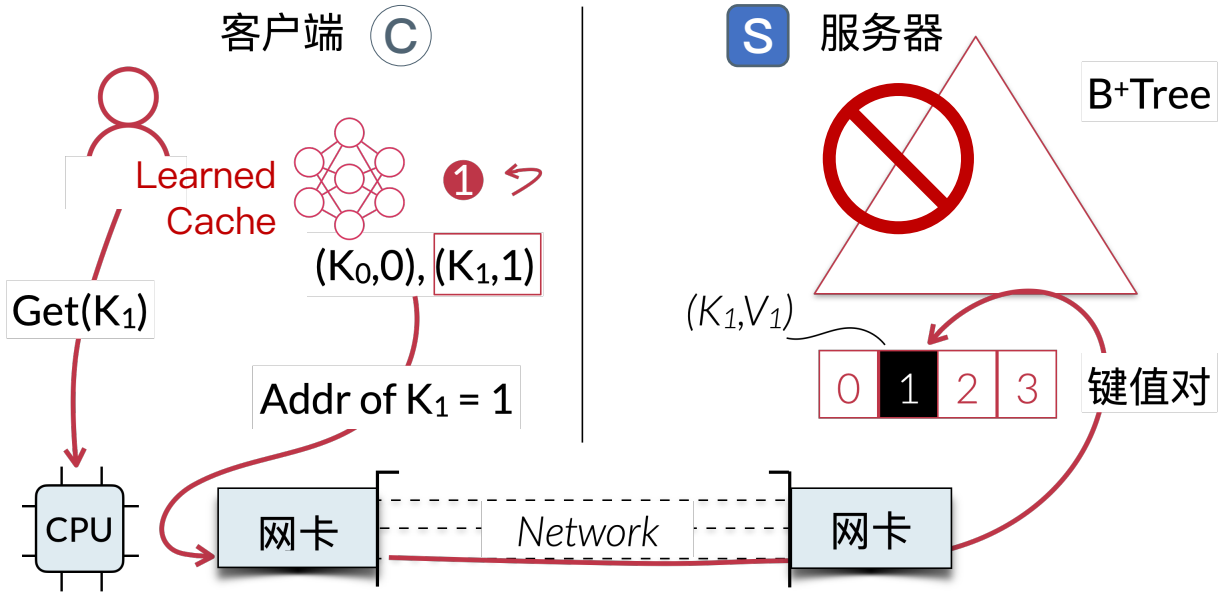


案例 #2-1：远程索引缓存

创新思路：使用机器学习 (ML) 模型构建索引缓存

- ML模型缓存优点：占用空间极少，无需整树遍历 → “用计算换网络”
- RDMA网络操作数：$O(\log(n))$ → $O(1)$
- 低内存开销 + 高查询效率

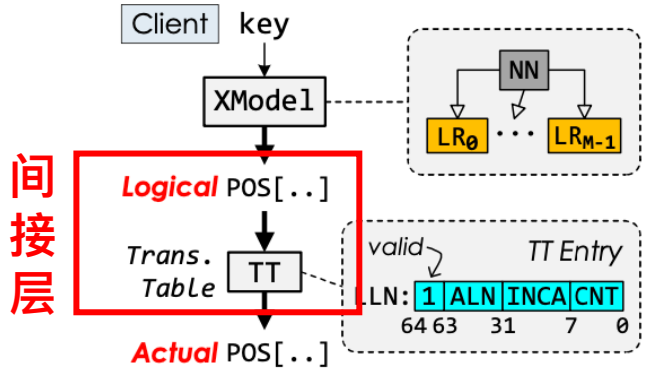


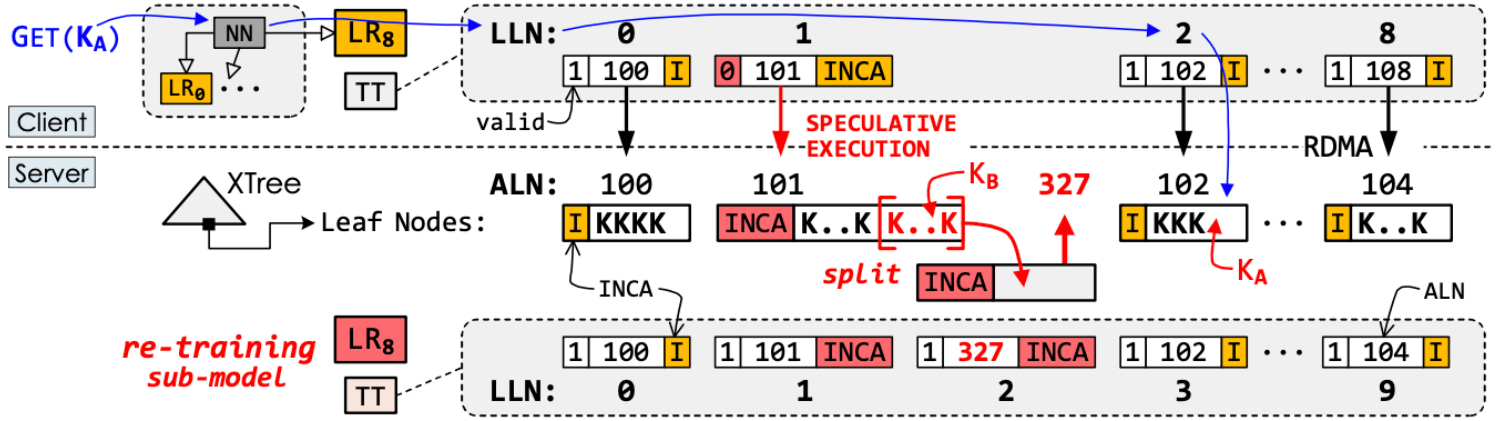


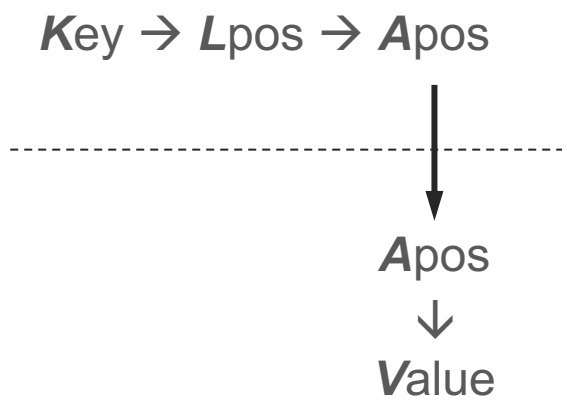
案例 #2-1：远程索引缓存

创新思路：使用机器学习 (ML) 模型构建索引缓存

- **挑战**：如何降低智能索引更新 (重训练) 开销？
- **技术**：引入间接层 (逻辑地址) → 解耦“**索引更新**”和“**模型训练**”
  + 客户端**按需**更新缓存



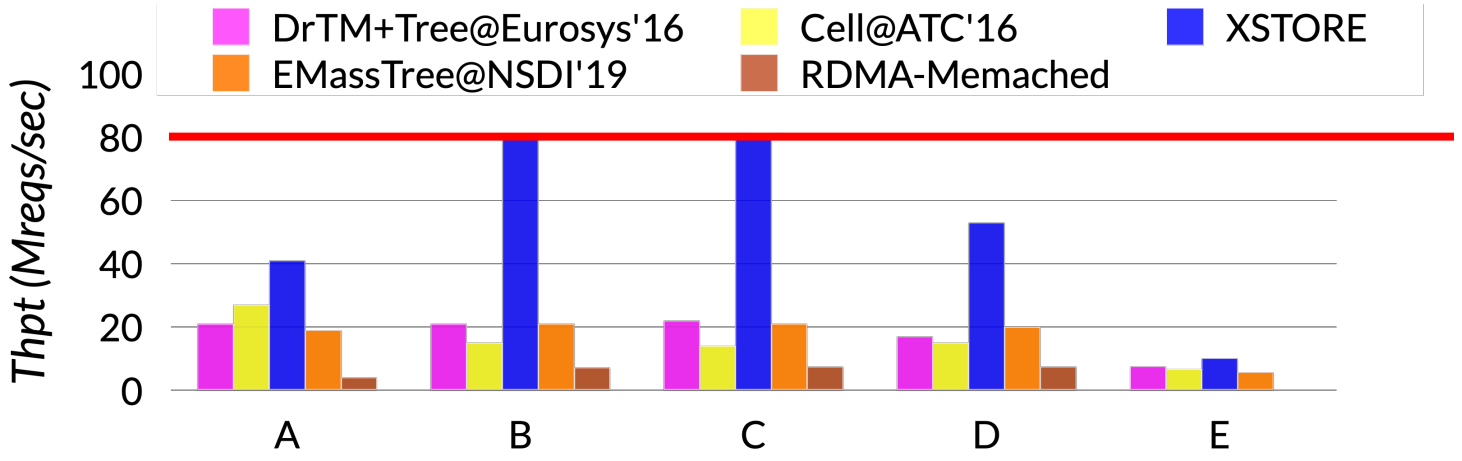




案例 #2-1：远程索引缓存

XStore：基于智能索引缓存的分布式有序键值存储系统

- **YCSB A-E**：100 M 键值对 (8B key+8B value)，Uniform / Zipfan 分布



Workload	A	B	C	D	E
R(S)/(U/I)	50:50	90:10	100:0	95:5	95:5

RDMA 网卡性能极限

只读负载：80M reqs/s
性能提升 3.7 ~ 5.9X

动态负载：53M reqs/s
性能提升 2.7 ~ 3.5X

研究成果发表在 Usenix OSDI、ACM TOS 等系统领域标志性会议和期刊

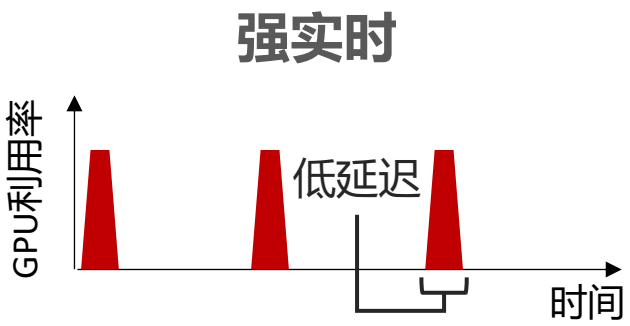
案例 #2-2：实时智能任务

深度神经网络推理服务 (DNN inference serving)

- 大量不同类型任务共存，例如，智能车场景
  - **强实时 (RT)**：障碍物识别、交通灯/标示/手势识别、. . .
    非实时 (BE)：路线规划、个性化推荐、智能助手、. . .
- 多样应用需求：实时性大算力、可控资源共享等
- 硬件平台：+GPU (AMD Radeon Instinct MI50、NVIDIA RTX 3080)

GPU

提供高并发大算力
但并发任务调度难
抢占、难共享

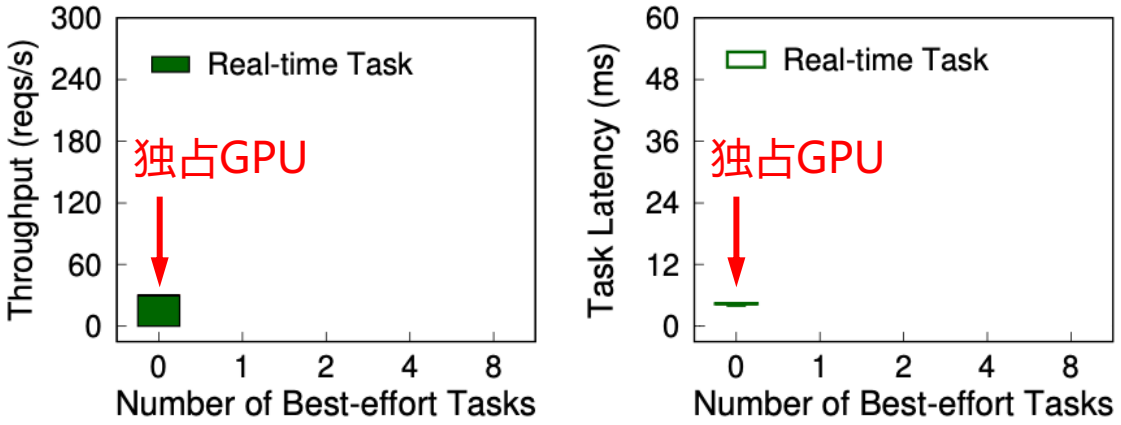


案例 #2-2：实时智能任务

独占式执行【主流方式】

✓ 无需抢占，RT任务立刻执行，不受干扰

× 成本高 (2x GPU)，GPU利用率低



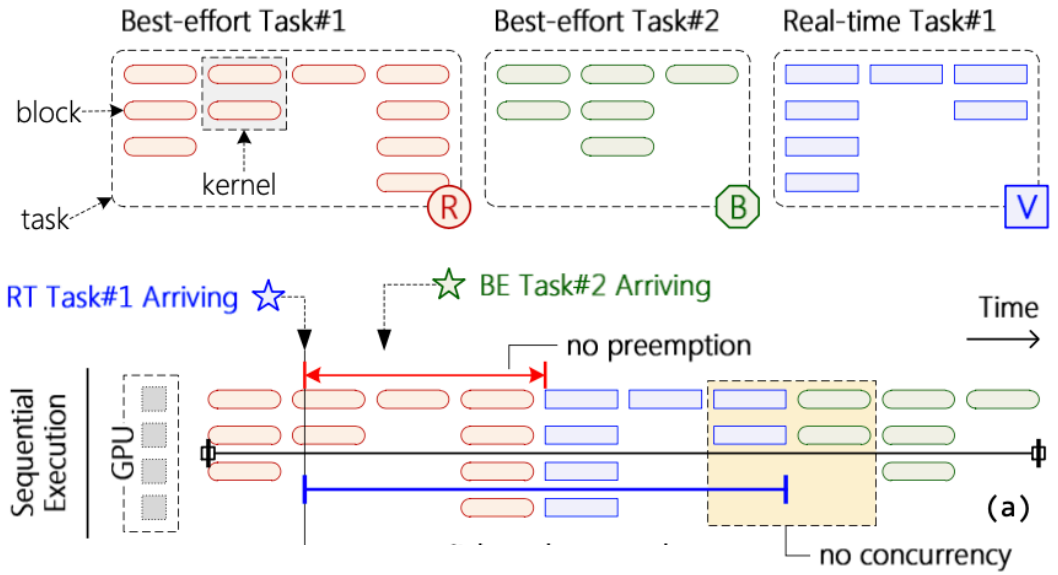
案例 #2-2：实时智能任务

串行执行 (Sequential Execution)

- 代表性工作：Clockwork[@OSDI20] 等
- 等待执行中任务完成

✓ RT任务执行不受干扰

× (端到端)时延长，GPU利用率低



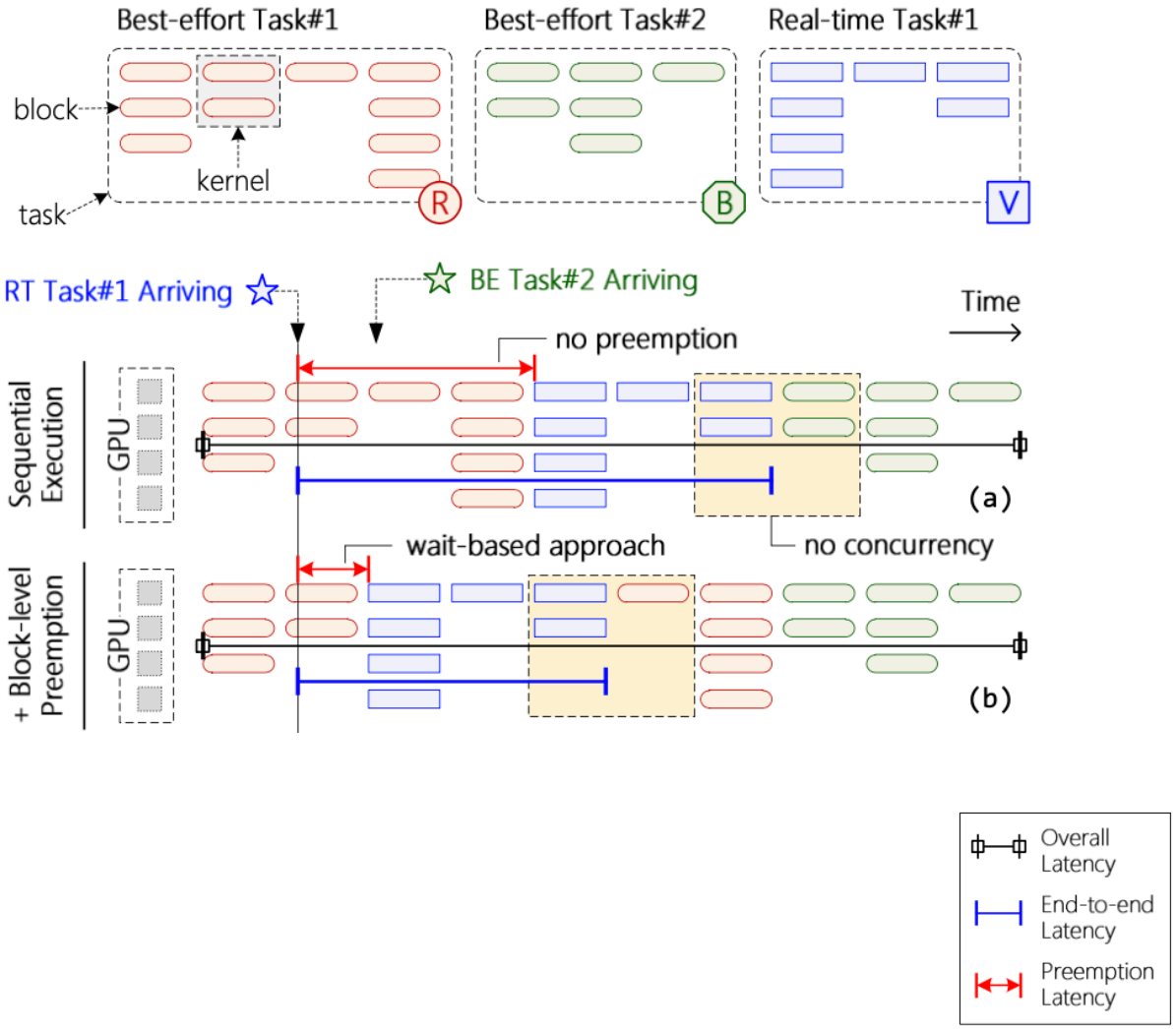
案例 #2-2：实时智能任务

被动抢占 (Block-level Preemption)

- 代表性工作： EffiSha[@PPoPP17] 等
- 等待执行中kernel完成

✓ 抢占延迟缩短

× (端到端)时延仍较长、GPU利用率低



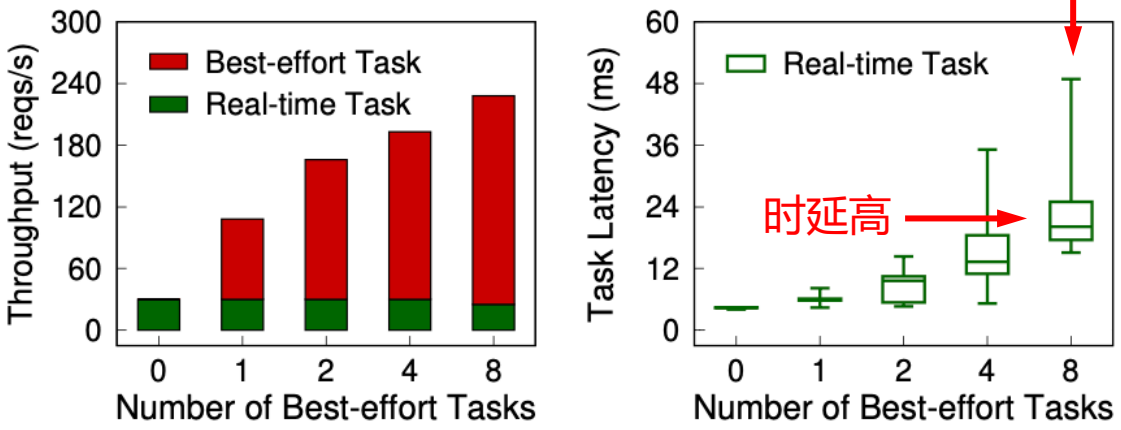
案例 #2-2：实时智能任务

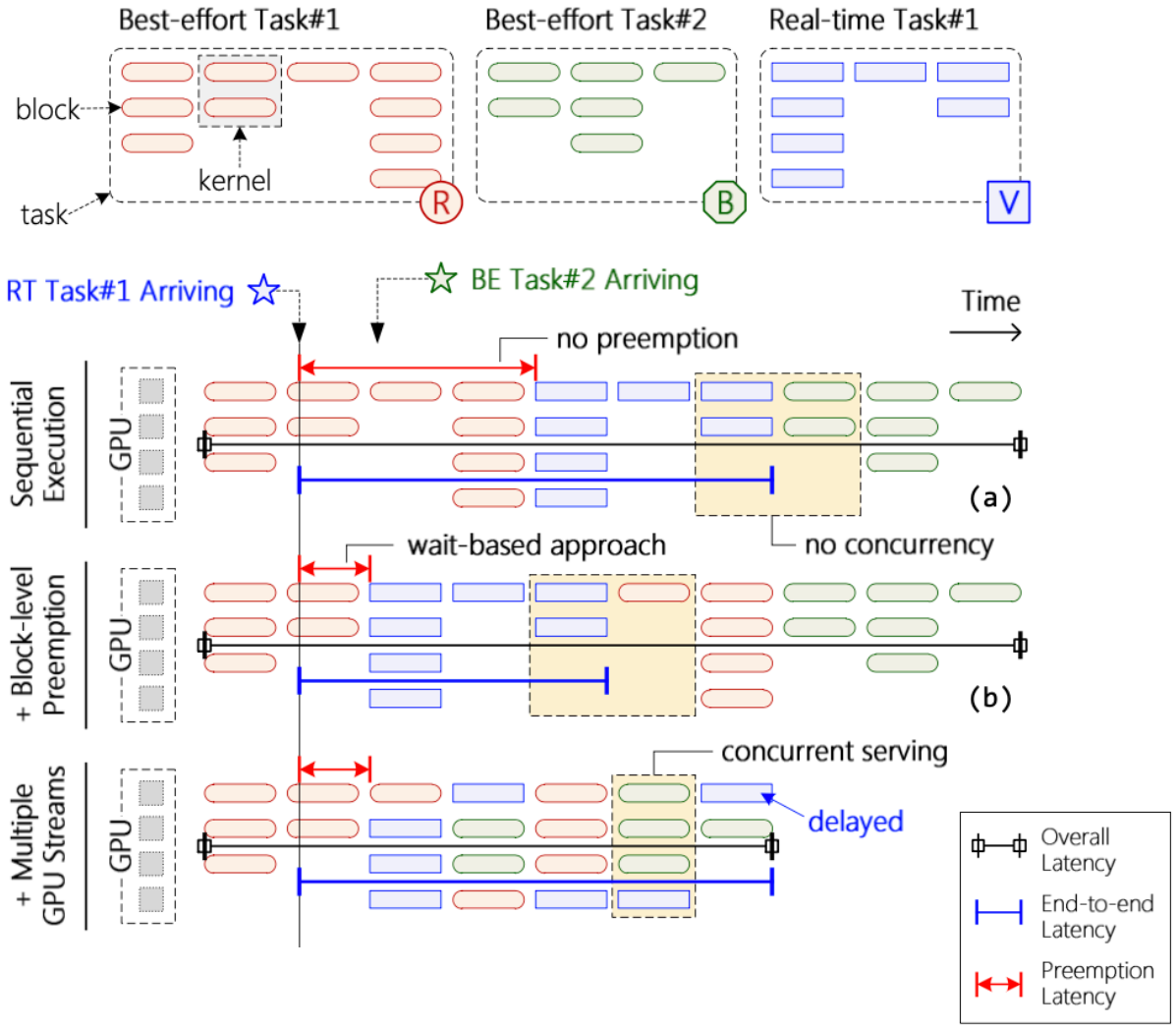
共享执行 (GPU Streams)

- 代表性工作：TensorRT@NVIDIA 等
- 并发执行所有任务，无抢占

✓ GPU利用率高，整体吞吐量高

✕ RT任务性能差，且波动大





案例 #2-2：实时智能任务

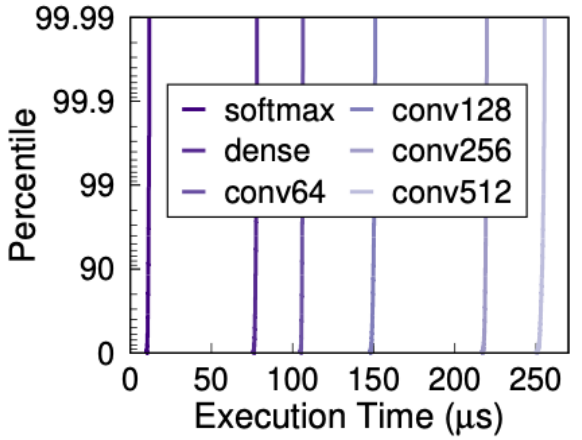
DNN推理任务关键特征

- **执行幂等性 (Idempotence)**：DNN模型中的GPU内核通常只包括线性代数计算，没有执行副作用，因而DNN推理任务的执行可以从**被中断的内核之前的任何内核恢复**，且不会改变推理结果

  注：已验证Apache TVM测试套件中的11个DNN模型所有320个GPU内核都具有幂等性！

- **延迟可预测 (Latency predictability)**：在GPU上独占式执行时，DNN推理中的GPU内核执行时间**稳定且是可预测**的

```
# device codes
__global__ void conv_relu(in, weight, out):
1    sum = 0;
2    for i in range(0,3)
3       for j in range(0,3)
4          sum += in[..] × weight[..]
5    out[..] = ReLU(sum)
__global__ void dense(in, weight, bias, out):
6    sum = 0;
7    for i in range(0,512)
8       sum += in[..] × weight[..]
9    out[..] = sum + bias[..]
```

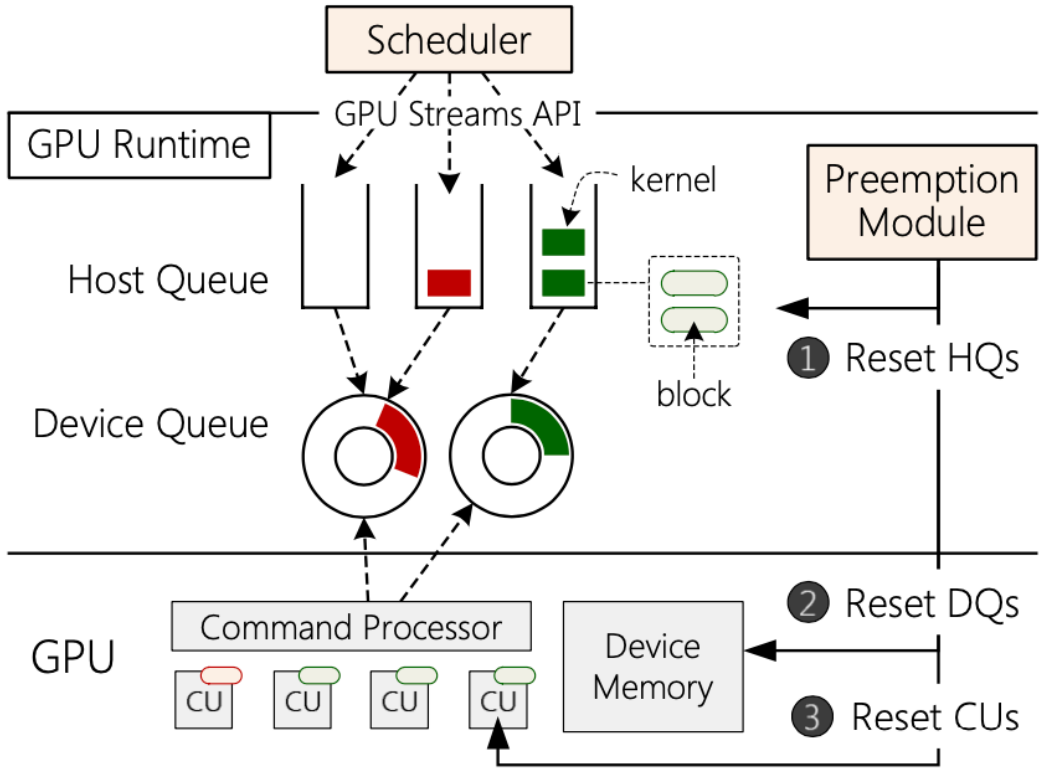


案例 #2-2：实时智能任务

创新思路 #1：基于重置的实时任务抢占

1. 利用GPU内核幂等性，即时重置GPU
2. 精准恢复被抢占任务的GPU内核

- **挑战**：如何即时重置GPU？
- **技术**：“软硬协同，动静结合”
  - ❶ GPU运行时重置任务队列 (HQs)
  - ❷ 编译器设置，GPU跳过执行 (DQs)
  - ❸ GPU驱动硬件重置执行单元 (CUs)



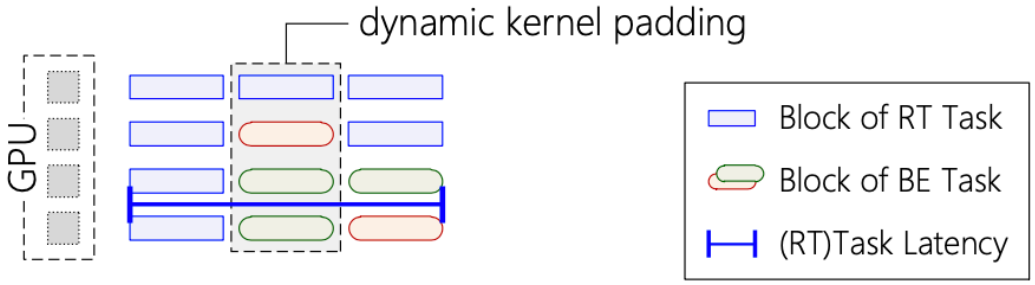
案例 #2-2：实时智能任务

创新思路 #2：动态内核可控填充

1. 仅使用RT内核剩余GPU资源执行非RT内核
2. 利用GPU内核时延可预测性避免影响RT内核

- **挑战**：如何实现动态可控填充？
- **技术**：代理内核 (Proxy kernel)

  + 高效GPU函数指针 (Function pointer)



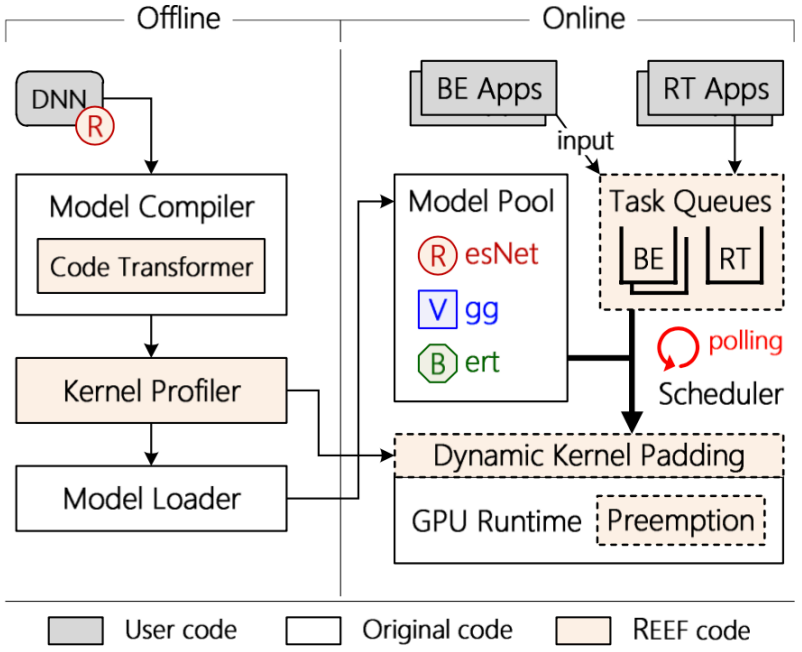

```
# device codes
__device__ void dense(in, weight, bias, out): ...

__global__ void dkp(rt_kern, rt_args,
                    be_kerns, be_argss):
1   ncus = rt_kern.ncus  # number of CUs
2   if (cu_id() < ncus) then
3      rt_kern(rt_args)  # run RT/kernel
4   else
5      ncus += be_kerns[i=0].ncus
6      while (cu_id() >= ncus)
7         ncus += be_kerns[++i].ncus
8      be_kerns[i](be_argss[i])  # run BE/kernel

# host codes
void inference(...):
    # set the real-time kernel w/ its args (e.g., dense)
9   rt_kern, rt_args = ...
    # select a set of best-effort kernels w/ their args
10  be_kerns, be_argss = kern_select(rt_kern)
11  dkp <<<..>>> (rt_kern, rt_args, be_kerns, be_argss)
12  ... # launch other dynamic padded kernels
```

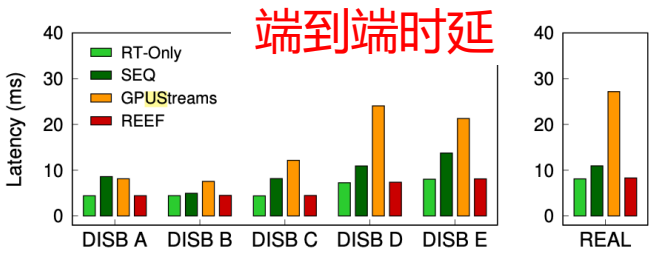
案例 #2-2：实时智能任务

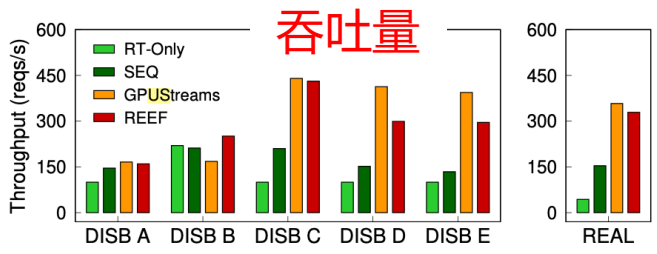
REEF：强实时、高并发DNN推理服务系统

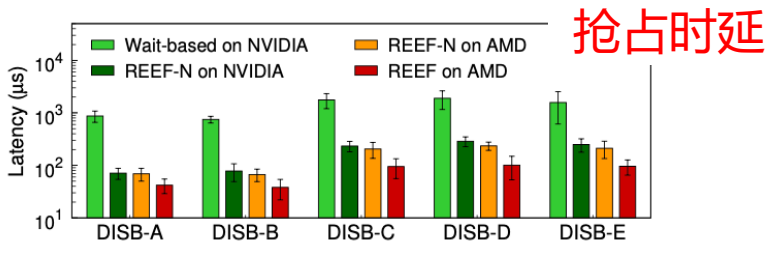
- 扩展 GPU 编译器、运行时、驱动等
- 同样支持闭源GPU，如 NVIDIA GPU
- **实时任务抢占**： < 40μs，降低抢占延迟15+倍
- **动态并发执行**：7.7X 吞吐量提升，< 2%时延下降

研究成果发表在 OSDI 等系统领域标志性会议









小节

1st CCF CHIP 中国计算机学会芯片大会

IPADS INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

ipads.se.sjtu.edu.cn

异构硬件成为发展新趋势，系统软件扮演重要新角色

硬件能力单一性与应用需求多样性间的矛盾带来系统性挑战

系统软件研究需要探索“经典”新路径——“上下求索”？

在“异构硬件聚合”和“应用特征反哺” 进行了尝试，并取得了一些成果

感谢

演讲人：陈 榕 @IPADS/上海交通大学
https://ipads.se.sjtu.edu.cn/~rchen