IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

SJTU

新硬件驱动的分布式事务处理系统
—— 性能、功能、智能

陈 榕

并行与分布式系统研究所 上海交通大学

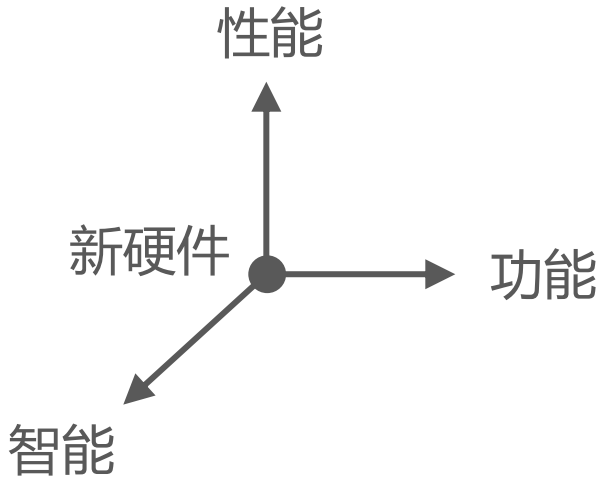
《大数据管理前沿技术论坛》

2022.11

研究合作者：魏星达、陈海波、臧斌宇以及IPADS分布式系统团队

提纲

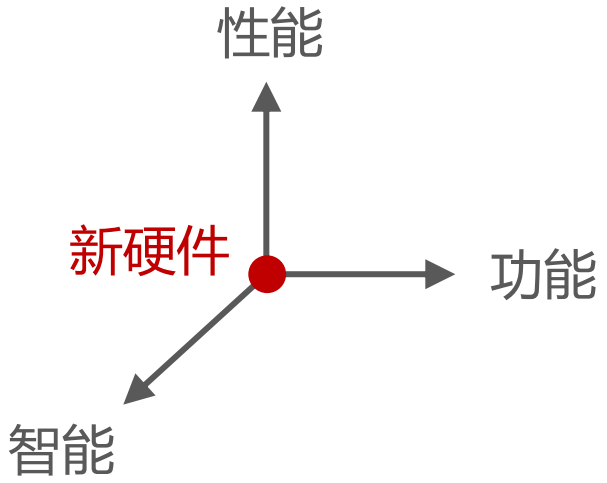
- 新硬件驱动分布式事务处理系统研究
- 方向 #1：性能
- 方向 #2：功能
- 方向 #3：智能



提纲

- 新硬件驱动分布式事务处理系统研究

- 方向 #1：性能
- 方向 #2：功能
- 方向 #3：智能



异构新硬件繁荣

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 / NPU
- 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

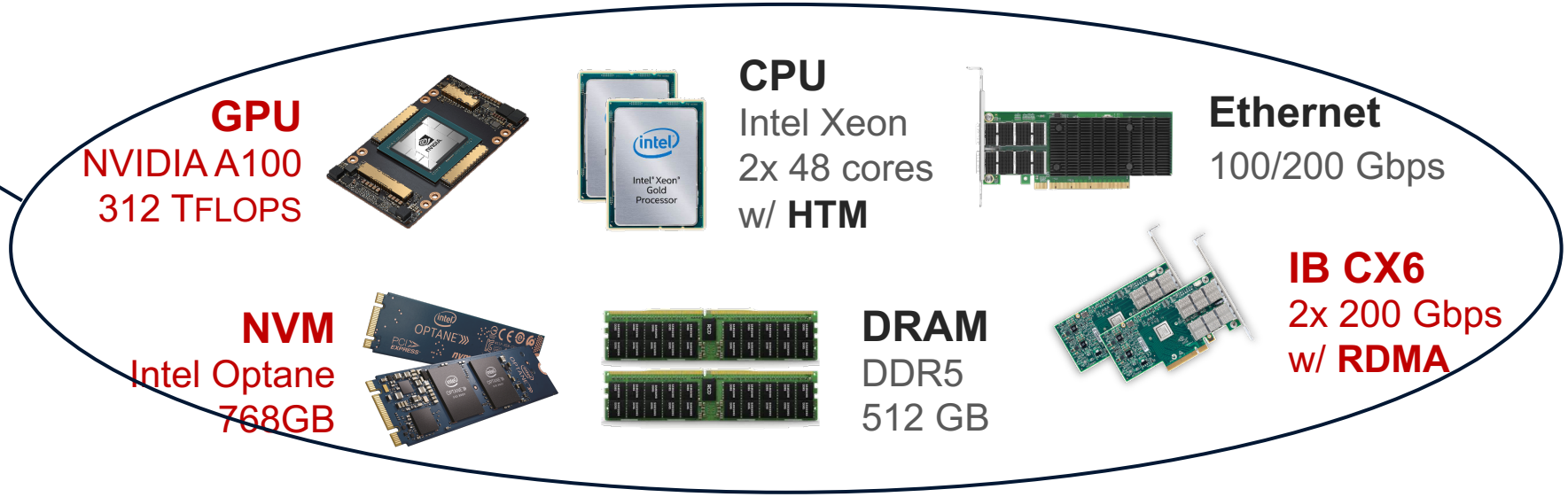
内存级、大容量

新功能

事务性执行

独立访存

数据持久化



基础系统研究：新硬件驱动

路径 #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、大数据分析、区块链
- RDMA（高带宽/低延迟）：高性能通信、Far Memory、Resource Disaggregation
- NVM（内存级/持久化）：文件系统、存储系统、数据库、系统缓存

路径 #2：新硬件赋能、软件功能硬件化

- HTM（事务性执行/ACI）：Transactions、并发控制、VM Introspection
- TEE/SGX（安全隔离）：可信计算、Sandbox、区块链共识、去中性化计算
- RDMA/SmartNIC（独立访存/计算）：负载均衡、高可用、内存管理、自动扩容

基础系统研究：新挑战

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

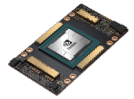
HTM

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

基础系统研究：新挑战

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

基础系统研究

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

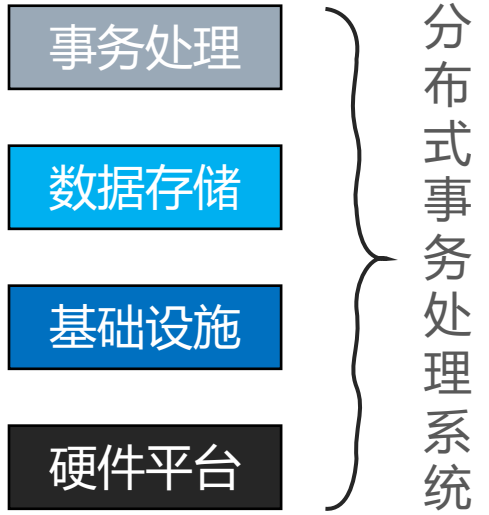
应用：分布式事务处理 (DTX)

分布式事务处理

大量关键应用的基础支撑

同时集合

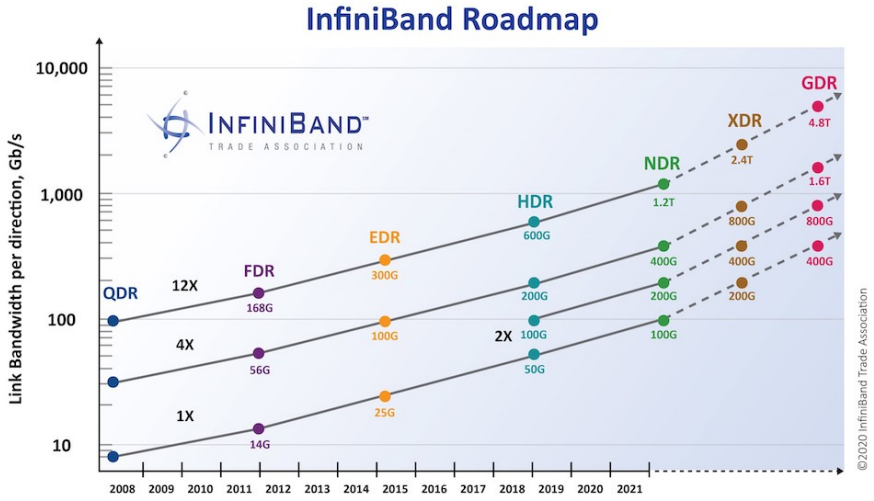
计算 / 存储 / 网络

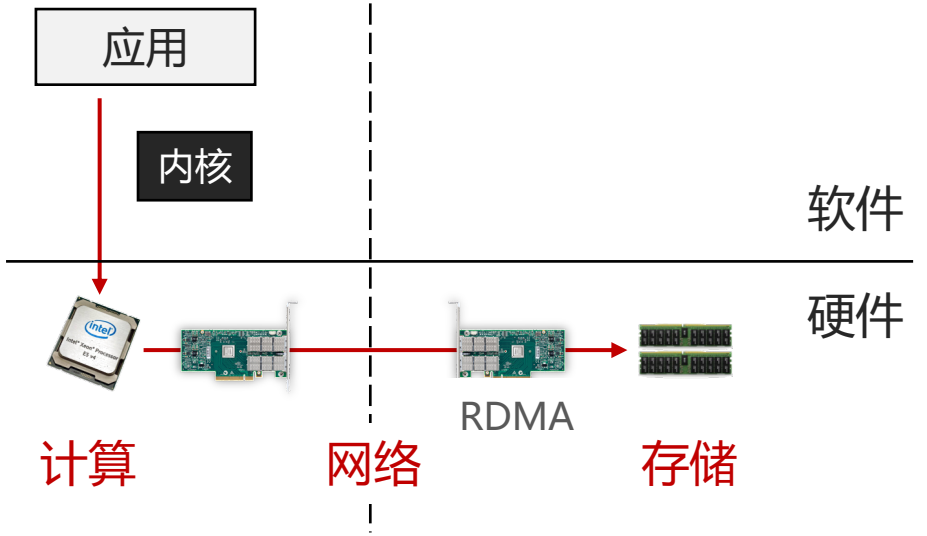


硬件：远程直接内存访问 (RDMA)

RDMA网络

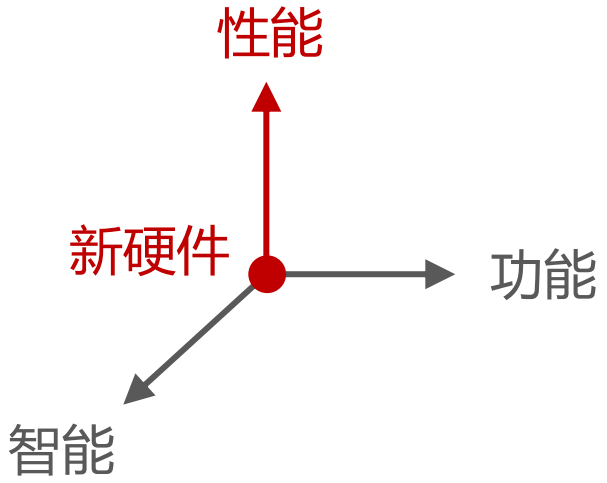
- **性能**：高带宽 e.g., > 400 Gbps
  低延迟 e.g., < 1 μs
- **功能**：跨节点直接内存读写
  CPU/kernel bypassing





提纲

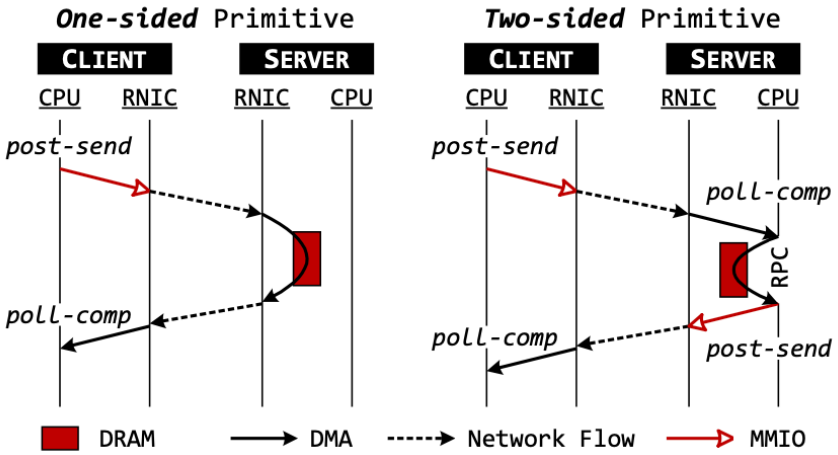
- 新硬件驱动分布式事务处理系统研究
- 方向 #1：性能
- 方向 #2：功能
- 方向 #3：智能



RDMA 101

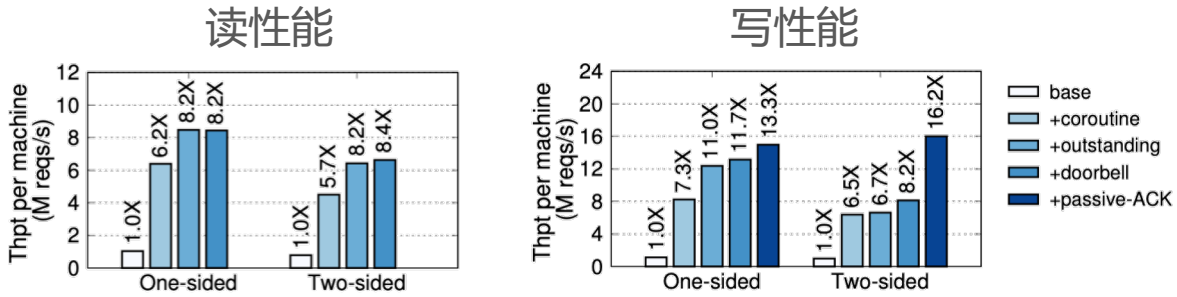
RDMA操作

- One-sided: READ | WRTIE | ATOMIC
- Two-sided: SEND | RECV



RDMA性能优化

- Coroutine @OSDI'16
- Oustanding Requests @SIGCOMM'14
- Doorbell Batching @ATC'16
- Passive ACK @OSDI'18



基于RDMA的分布式事务处理

RDMA+DTX 系统

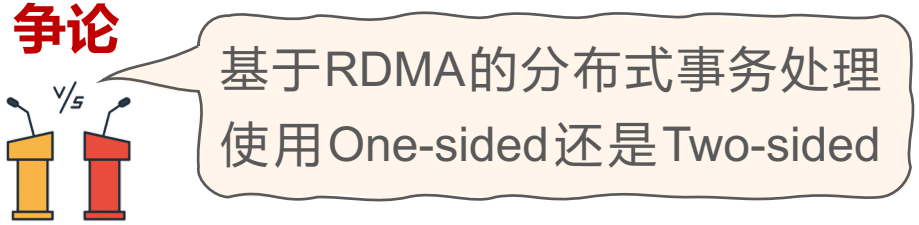
- One-sided系统代表：

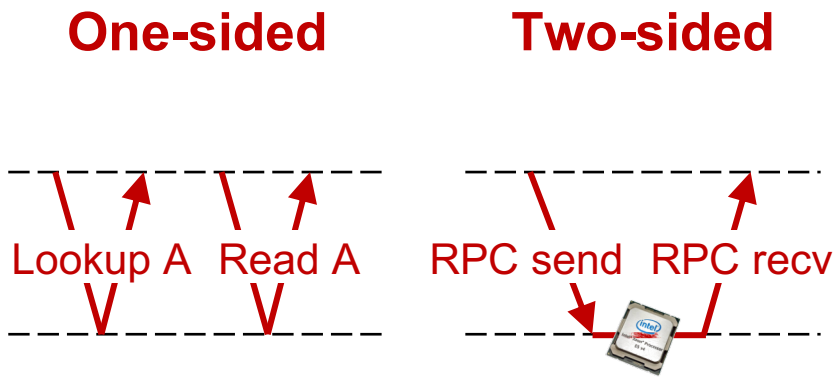
  DrTM @SOSP'15 FORD @FAST'22

- Two-sided系统代表：

  FaRM @SOSP'15 FaSST @OSDI'16

注：都采用相同的OCC协议实现





	One-sided	Two-sided
Throughput	✓	✗
#Round-trips	>=2	1
CPU util	low	high

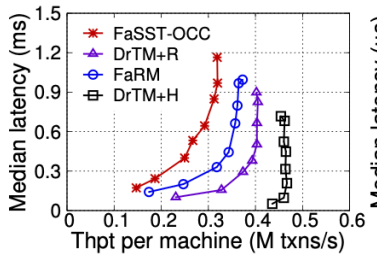
混合分布式事务处理系统

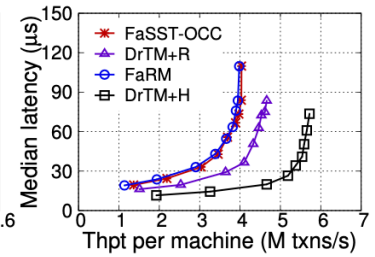
DrTM+H @OSDI'18：基于RDMA混合实现的DTX

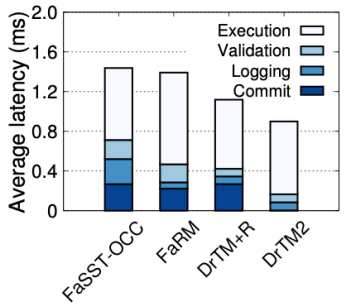
- 首个混合RDMA实现：one-sided ⊕ two-sided
- 解构并发协议，分阶段定制

  阶段：执行/E、检查/V、日志/L、提交/C
- 效果：相比最优单种类RDMA实现，

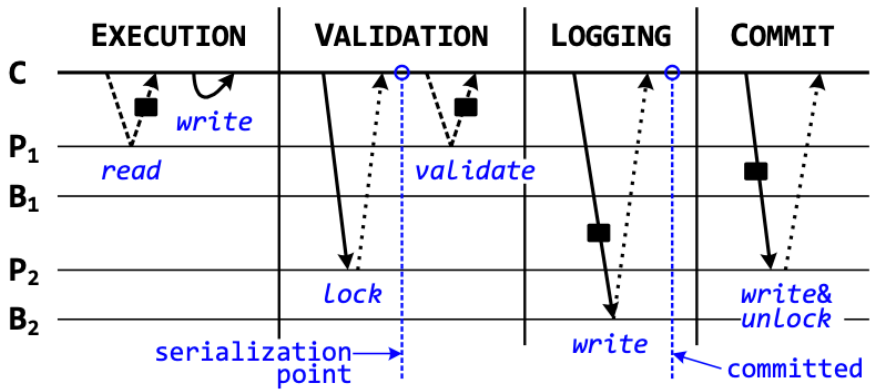
  提升200%吞吐量，降低65%时延







事务协议阶段



* 分布式事务协议各阶段都能使用两种RDMA实现

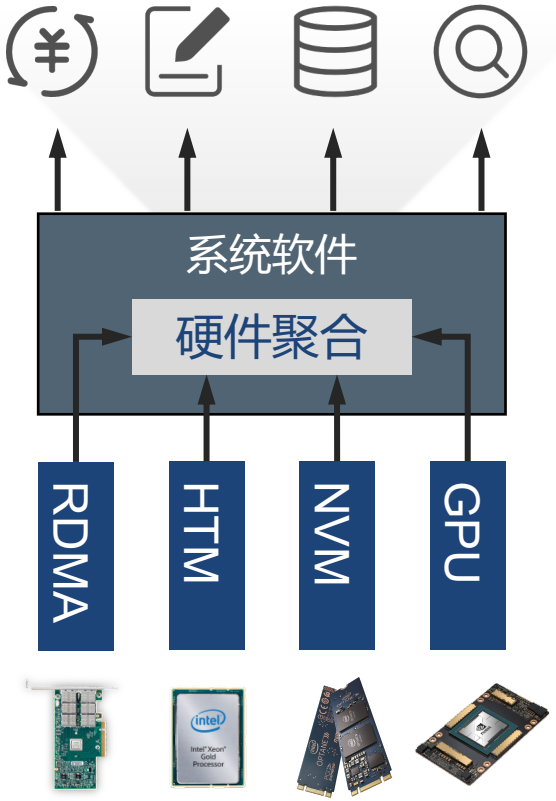
	RW-TX				RO-TX	
	E	V	L	C	R	V
FaRM	I	II+I	I	II	I	I
DrTM+R	I	I+I	I	I+I	I	I
FaSST	II	II	II	II	II	II
DrTM+H	I/II	I/II	I	I/II	I/II	I

异构数据中心：多种新硬件共存

挑战：组合多种硬件特性，实现“1+1>2”

- 新思路：以网络硬件RDMA为核心，将硬件事务性内存HTM、非易失性内存NVM等异构新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)



分布式事务处理

基于RDMA的事务处理研究

- 代表性工作：FaRM @SOSP'15 FaSST @OSDI'16 DrTM+H @OSDI'18
- 使用RDMA实现事务协议，如OCC @SOSP'15/EuroSys16
- 性能受损：提出versioning等机制确保RDMA访存一致性

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 @EuroSys'14 Leis et al. @OSDI'16
- 使用HTM实现事务协议，如OCC @EuroSys'14 TSO @ICDE'14
- 场景受限：简单事务逻辑，且仅限于单机事务执行

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

分布式事务处理：RDMA ⊕ HTM

HTM: Hardware Transaction Memory

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

Strong Atomicity

分布式事务处理：RDMA ⊕ HTM

HTM: Hardware Transaction Memory

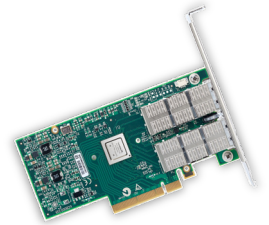
> *a non-transactional code will unconditionally abort a transaction when their accesses conflict*

Strong Atomicity

RDMA: Remote Direct Memory Access

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

Strong Consistency



分布式事务处理：RDMA ⊕ HTM

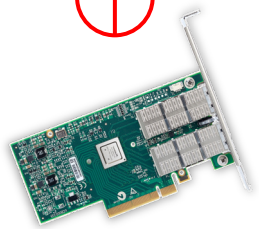
HTM: Hardware Transaction Memory

> *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

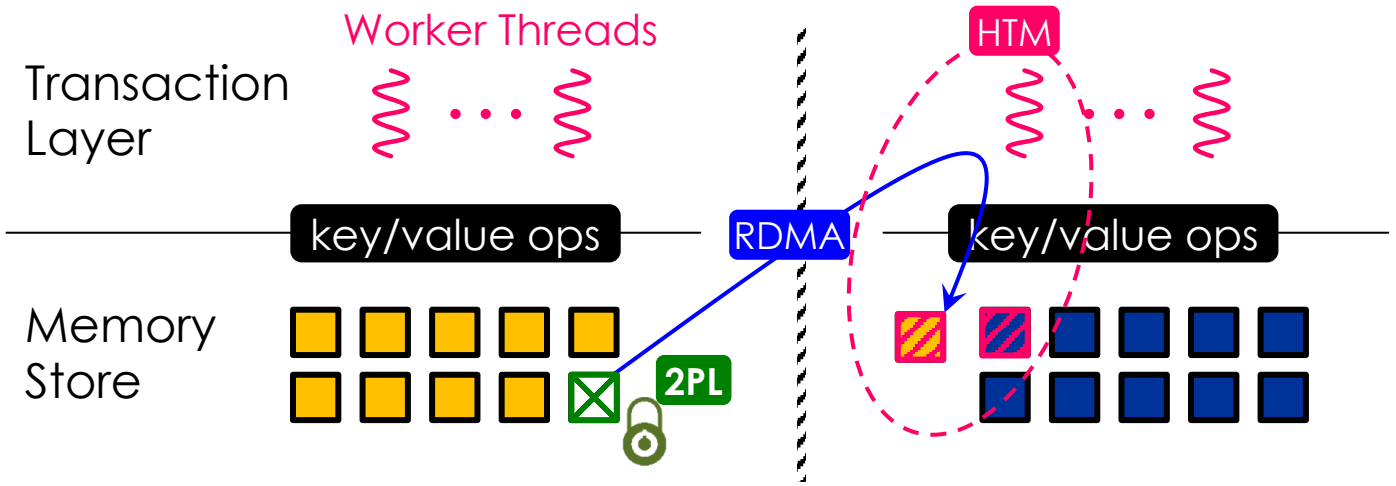
Distributed HTM

分布式硬件事务内存

分布式事务处理：RDMA ⊕ HTM

DrTM @SOSP'15：基于Distributed-HTM的分布式事务处理

- 面向RDMA+HTM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)



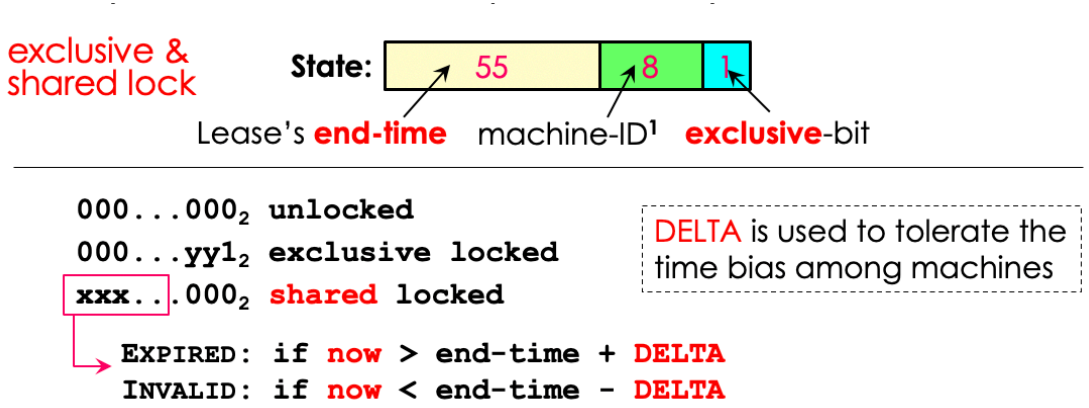
DrTM 并发控制方法

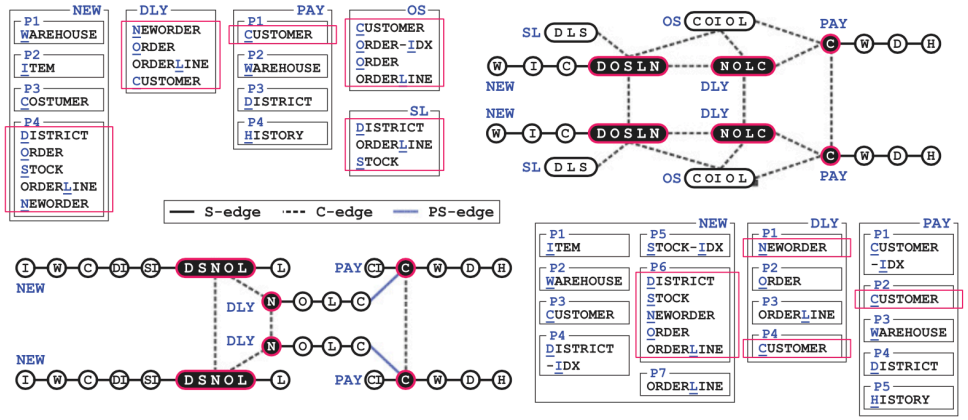
- **HTM**: L-TX vs. L-TX
- **RDMA**: L-TX vs. D-TX
- **2PL**: D-TX vs. D-TX

分布式事务处理：RDMA ⊕ HTM

DrTM @SOSP'15：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**： 使用访存量受限的HTM实现复杂的事务处理
- **技术 #2**： 采用 TX Chopping 技术实现事务的按需切分

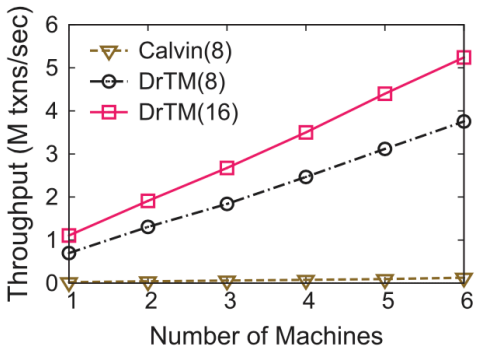




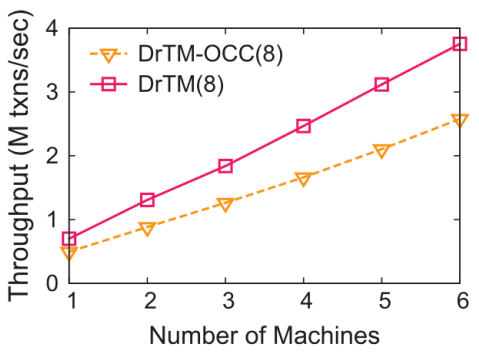
分布式事务处理：RDMA ⊕ HTM

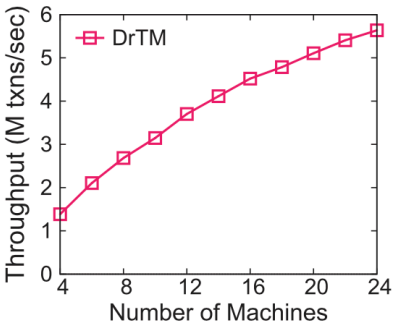
DrTM @SOSP'15：基于Distributed-HTM的分布式事务处理

- 6节点RDMA集群上，TPC-C吞吐量达到 5.52 百万事务每秒，相比传统最优系统(Calvin)性能提升20倍，平均时延降低下降2个数量级
- 相比仅使用RDMA实现的系统，吞吐量进一步提升50%，并具有更好的扩展性



		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

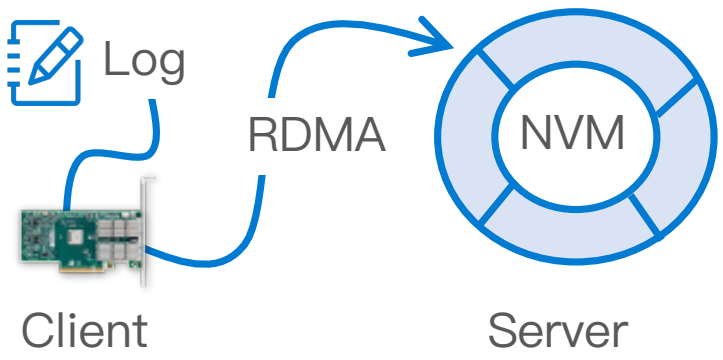


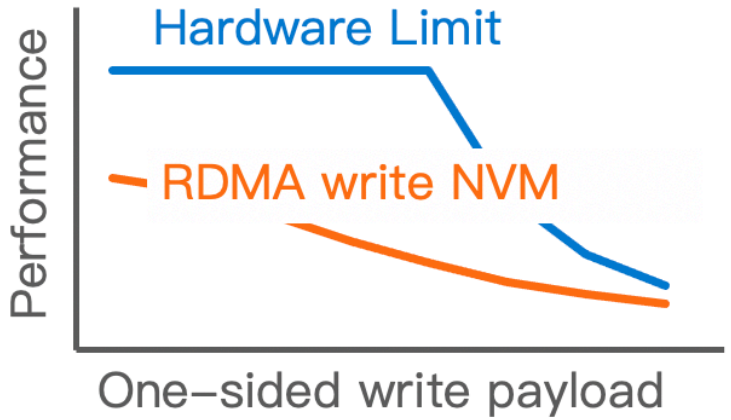


高可用事务日志：RDMA ⊕ NVM

基于RDMA+NVM实现高可用事务日志

- 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
- 代表性工作： Octopus @ATC'17 Orion @FAST'19 AsymNVM @ASPLOS'20
- 性能受损：仅考虑了RDMA+NVM系统正确性，协作性能差





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

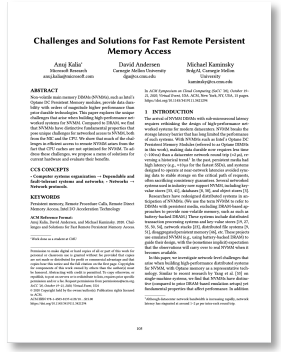
NVM

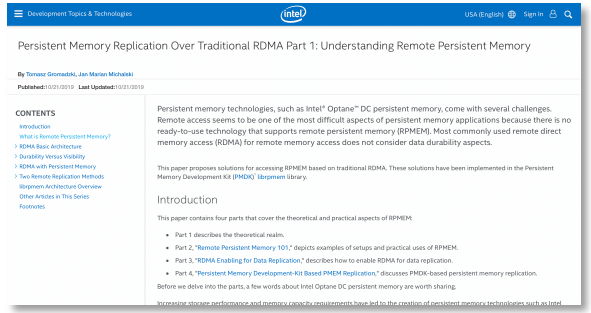
提供数据持久性但受限在单节点内，且依赖于特定指令和不同访存粒度

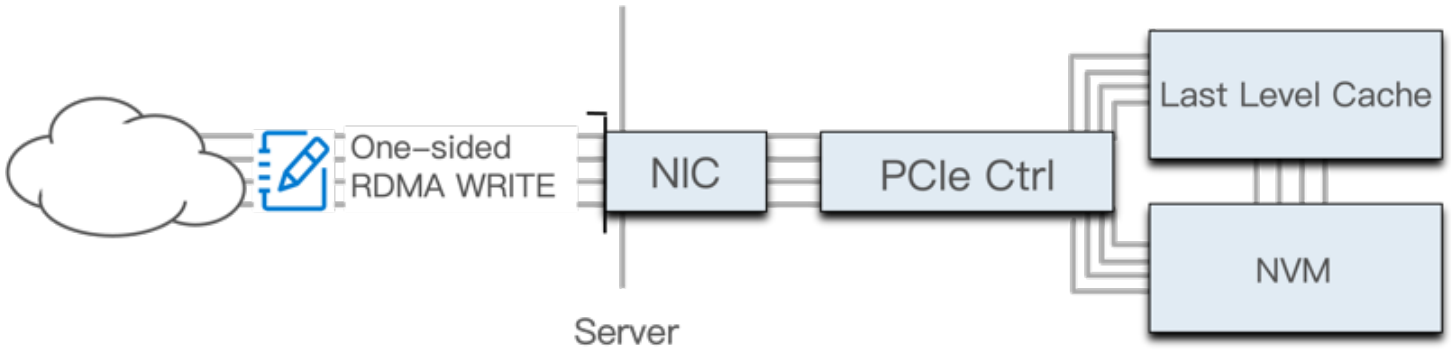
高可用事务日志：RDMA ⊕ NVM

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. @FAST'20 ，Kalia et al. @SOCC'20
- 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







高可用事务日志：RDMA ⊕ NVM

系统性研究 RDMA+NVM 的性能及优化方法 @ATC'21

- 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

  事务处理：DrTM+H @OSDI'18　文件系统：Octopus @ATC'17

Optimization Hints

study

RDMA ⊕ NVM

optimizing

Octopus

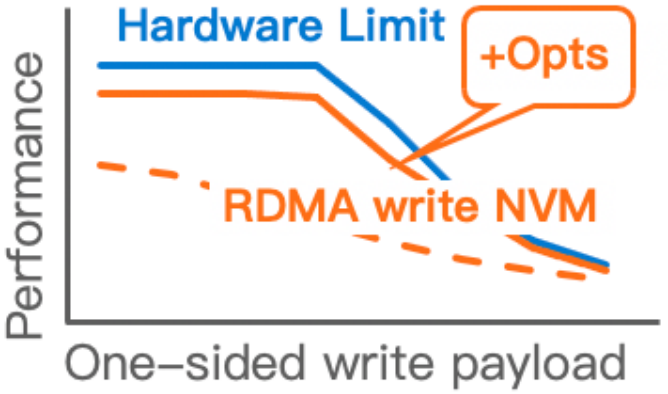
DrTM+H

高可用事务日志：RDMA ⊕ NVM

系统性研究 RDMA+NVM 的性能及优化方法 @ATC'21

- 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

  事务处理：DrTM+H @OSDI'18　文件系统：Octopus @ATC'17



Optimization Hints

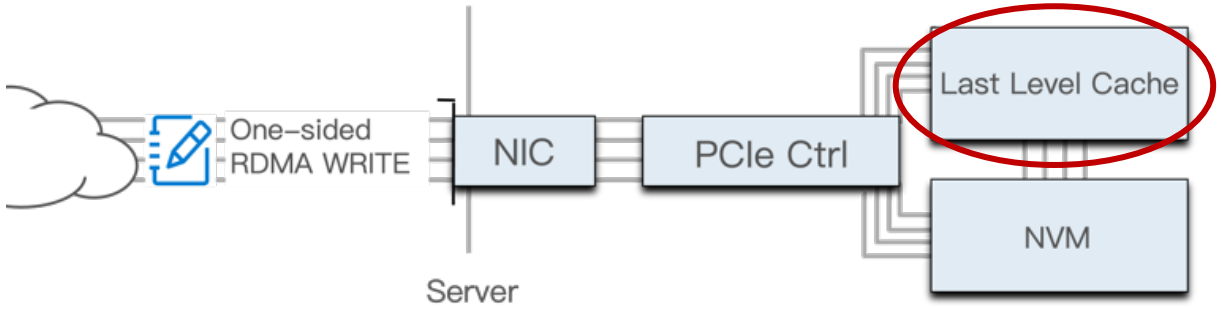
- Configuration (H1-H3)
- Access pattern (H4-H8)
- Persistent operation (H9)

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
...

高可用事务日志：RDMA ⊕ NVM

RDMA+NVM 优化建议

- **H5**：关闭 DDIO 以绕过处理器缓存

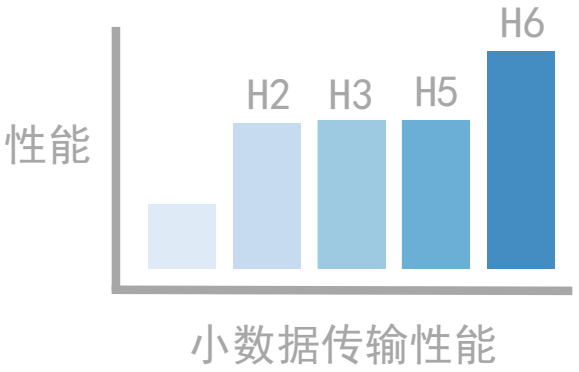


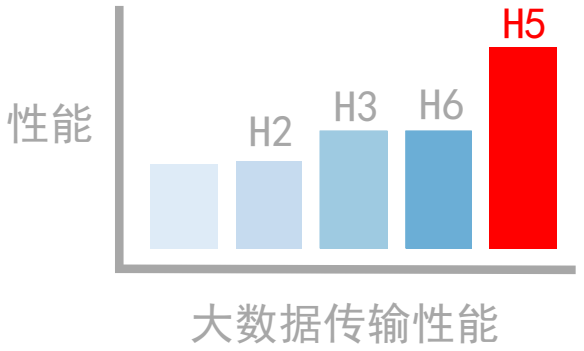
NVM 硬件特性

- 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

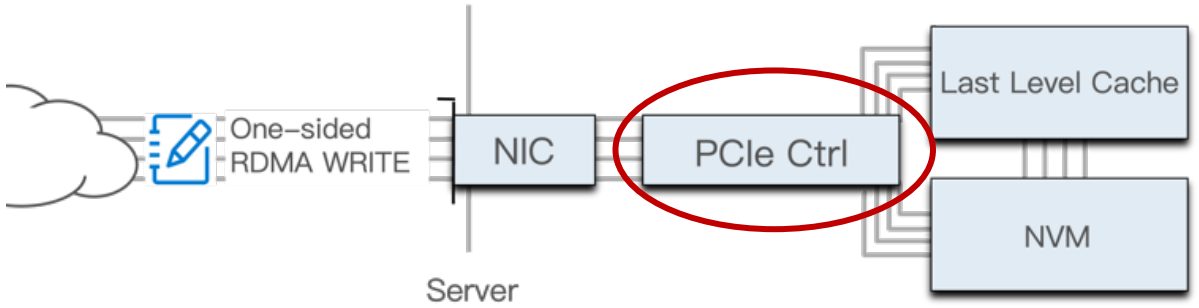




高可用事务日志：RDMA ⊕ NVM

RDMA+NVM 优化建议

- **H6**：按 PCIe 最小传输粒度访问 NVM

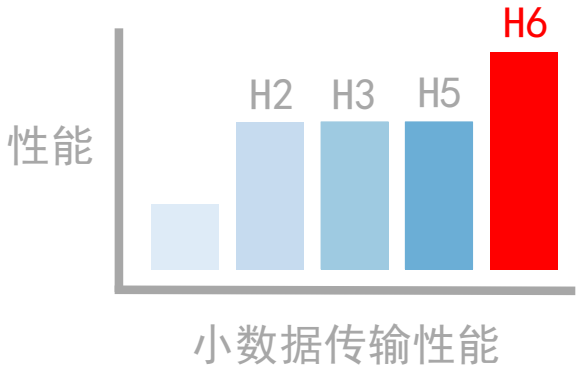


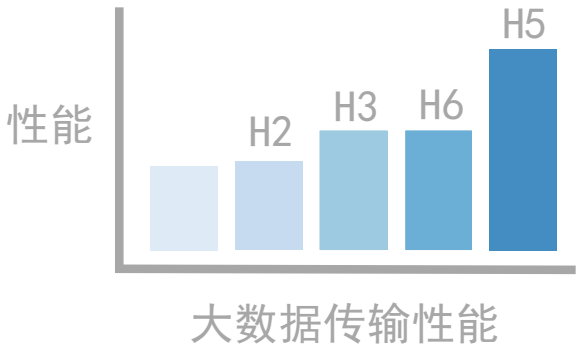
NVM 硬件特性

- 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

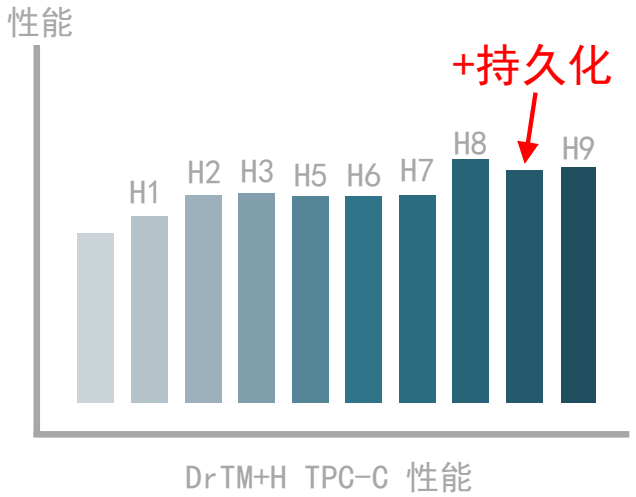




高可用事务日志：RDMA ⊕ NVM

利用建议优化 RDMA+NVM 系统

- 分布式事务处理 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



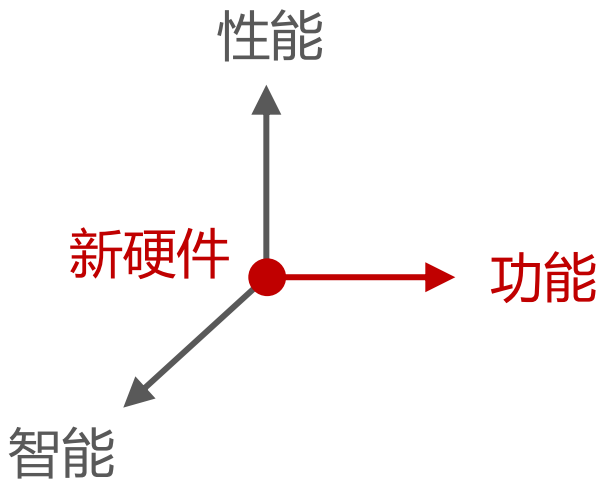
RDMA-NVM 优化建议 (H1-H9)
H1. 对于大数据传输使用NT-store
H2. 避免使用跨socket访问
H3. 避免以小于xpline粒度的访问
H4. 减少并发CPU NVM访问
...

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

提纲

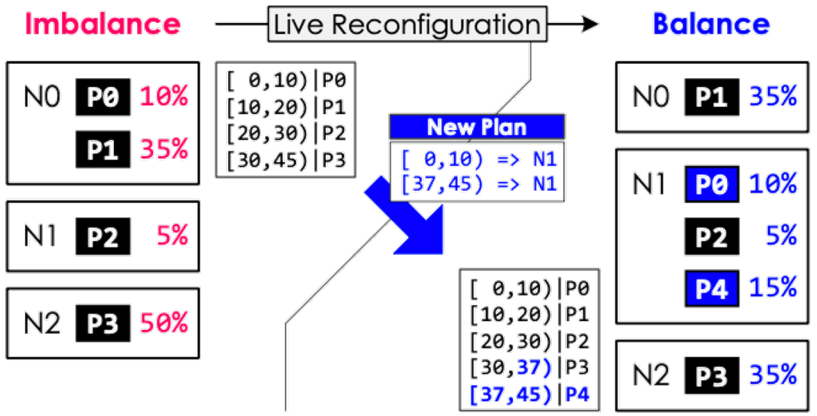
- 新硬件驱动分布式事务处理系统研究
- #1：性能
- #2：功能
- #3：智能



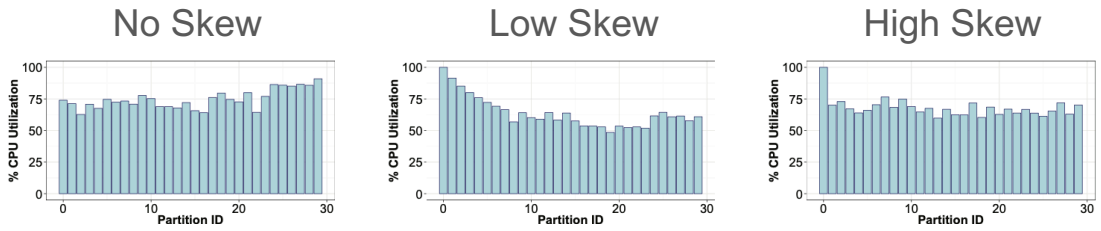
功能 #1：动态重配置

分布式事务处理面临负载均衡问题

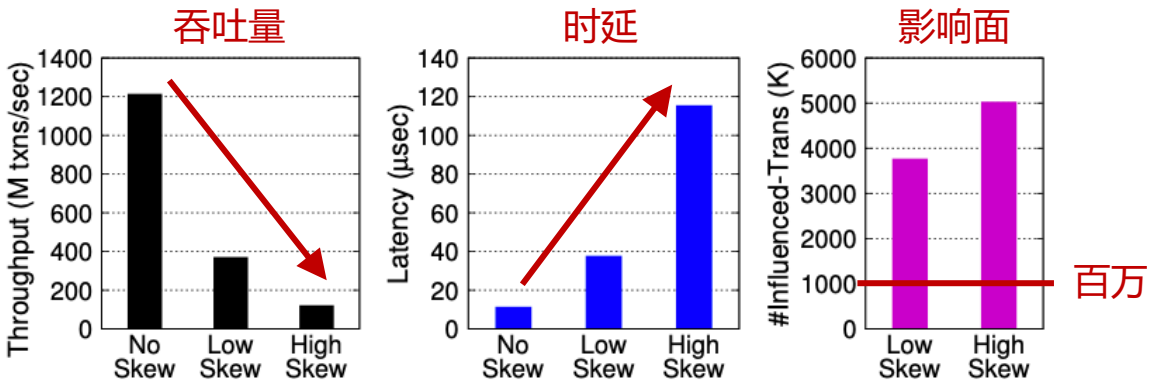
- 负载不均 → 性能受损严重、影响百万事务
- 动态重配置：数据在线迁移



- P0 在线迁移到 N1
- P3 分裂成 P3 和 P4，P4 在线迁移到 N1



来自 E-Store @VLDB'14

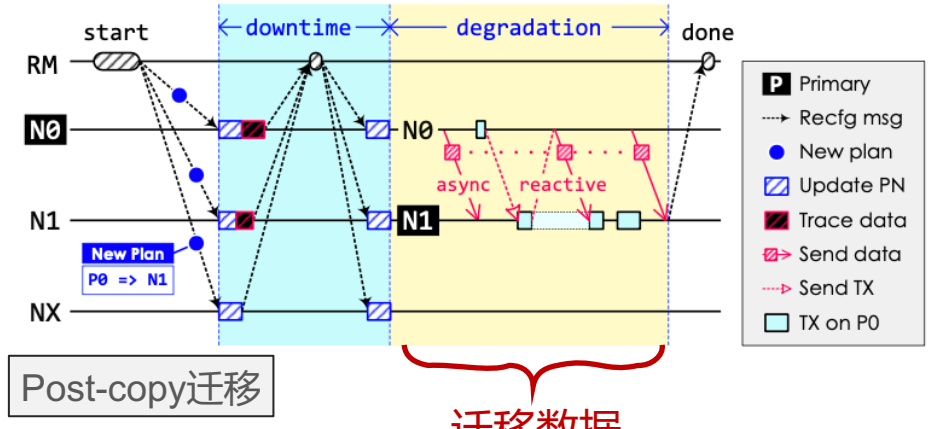


测试结果来自RDMA事务处理系统 DrTM+R@EuroSys'16

功能 #1：动态重配置

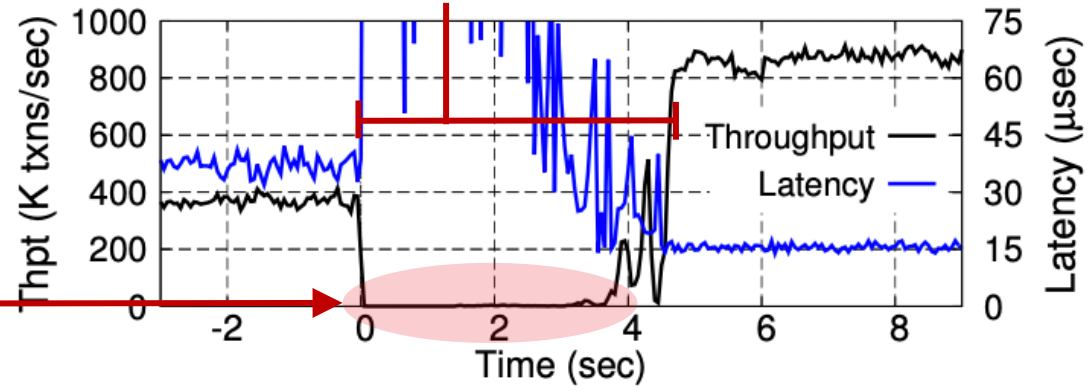
分布式事务处理系统的数据在线迁移

- Post-copy @SIGMOD15：先迁负载、后迁数据
- 性能受损时间长达数秒、且存在停止服务
  - 按需迁移数据延迟高
  - 数据迁移加剧系统过载



影响时间：4.7秒

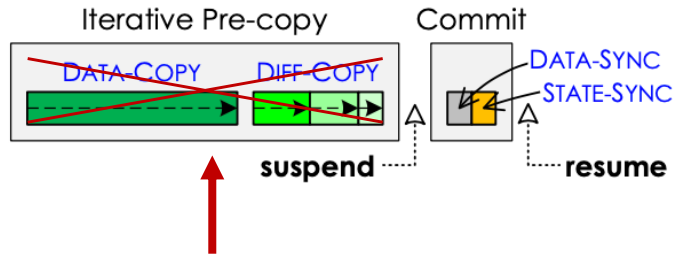
停止服务：THPT=0



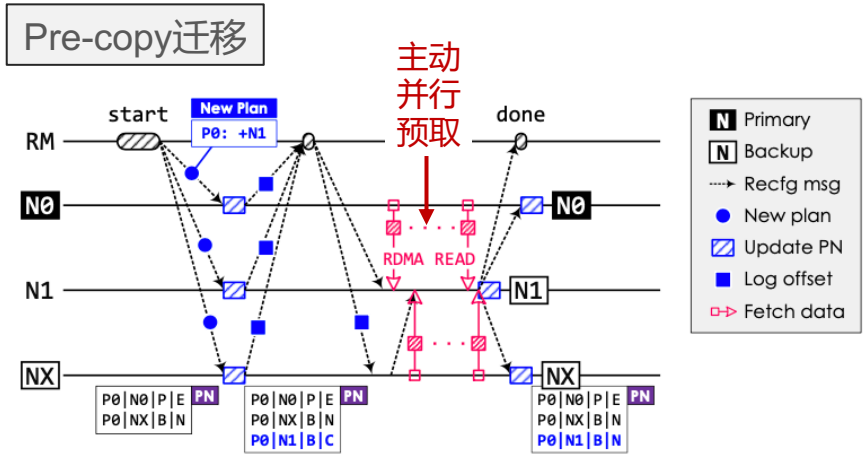
功能 #1：动态重配置

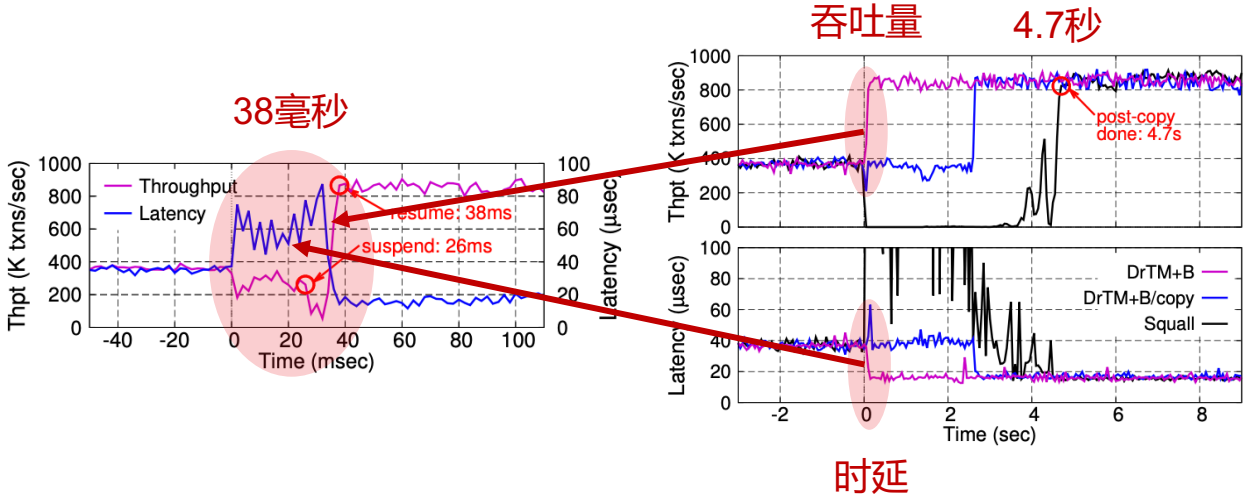
DrTM+B @ATC17：基于HA的RDMA动态重配置技术

- 分布式高可用技术(HA)提供 “一致性多副本”
- **技术**：预拷贝(Pre-copy)、RDMA异步主动并行预取
- **效果**：重配置过程性能损失降低十倍、受损时间缩短百倍



复用HA副本，跳过迭代拷贝





功能 #2：快速自动扩容

在线事务处理 ⊕ Serverless Computing

- 利用基础设施提供的自动扩容能力
- 代表性工作：FaRMv2 @SIGMOD'19 Beldi @OSDI'20 Heus et al. @DEBS'21
- Serverless数据库：PolarDB / AWS Aurora / Azure SQL / CockroachDB Serverless
- 优点：快速响应负载变化、降低运营成本

Azure SQL Database serverless

Article • 10/29/2022 • 18 minutes to read • 14 contributors

Applies to: Azure SQL Database

Serverless is a compute tier for single databases in Azure SQL Database that automatically sc
compute based on workload demand and bills for the amount of compute used per second.
serverless compute tier also automatically pauses databases during inactive periods when or
storage is billed and automatically resumes databases when activity returns.

Amazon Aurora Serverless

注册 Aurora Serverless v2（预览版） 开始使用 Amazon Aurora

Amazon Aurora Serverless 是 Amazon Aurora 的一种按需自动扩展配置版本。Amazon Aurora Serverless 会根据应用程序的需求自动启动
库实例，即可在云中运行数据库。

Serverless Databases on MongoDB

Deploy a serverless database to meet variable application demand with minimal configuration and a reliable backend you can trust.

CockroachDB serverless is generally available & more product updates

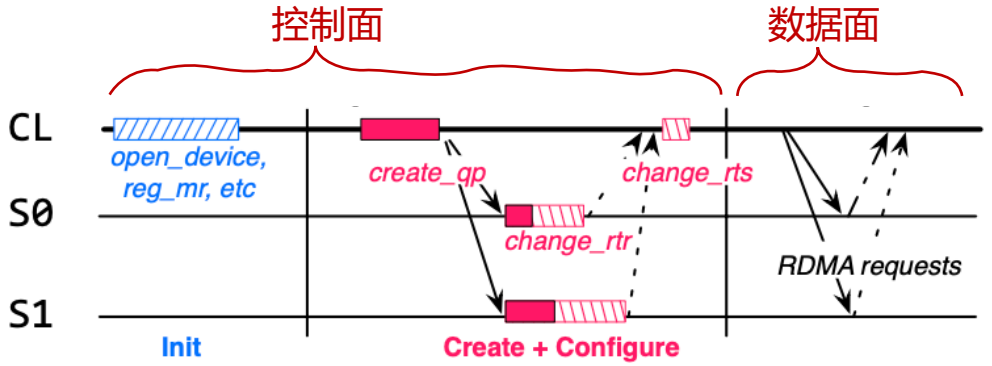
Written by Nate Stewart on September 21, 2022

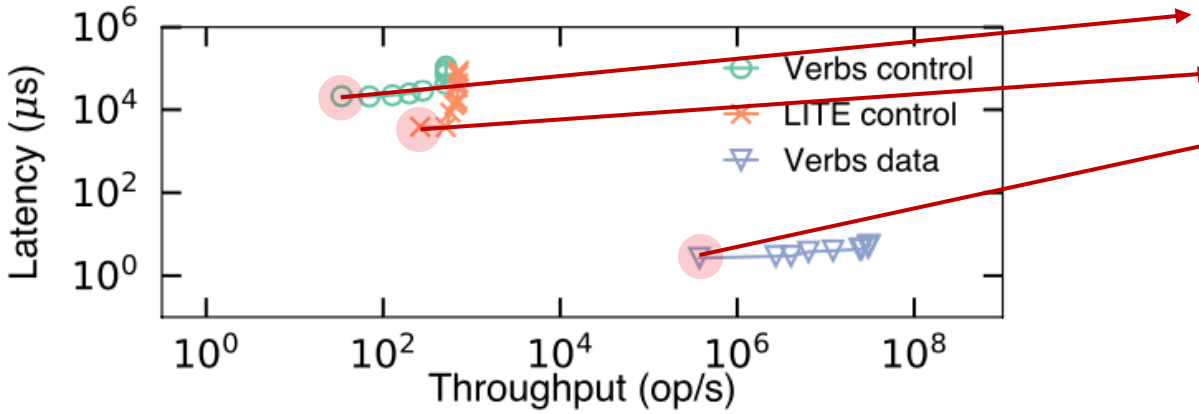
功能 #2：快速自动扩容

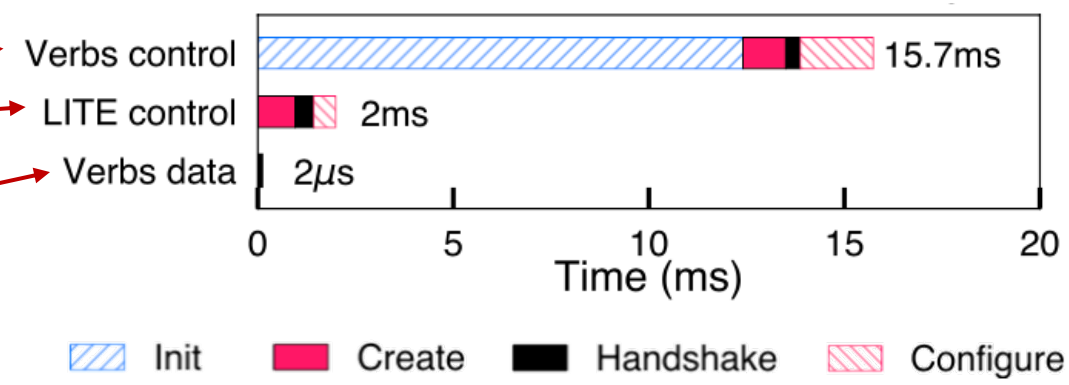
RDMA事务处理在自动扩容时面临的挑战 (#1)

- **挑战**：RDMA控制面 (Control Plane) 开销大
- 创建并连接 RCQP 需要数毫秒 >> 数据传输延迟

注：RCQP = Reliable Connected Queue Pair



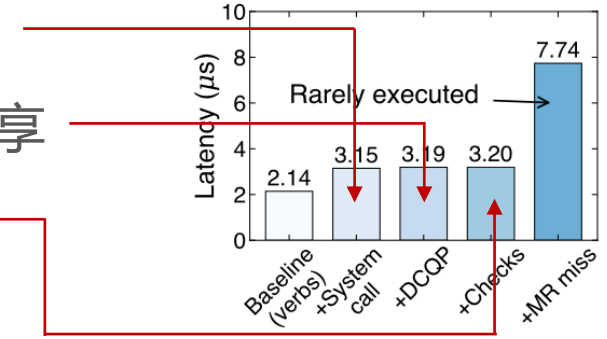


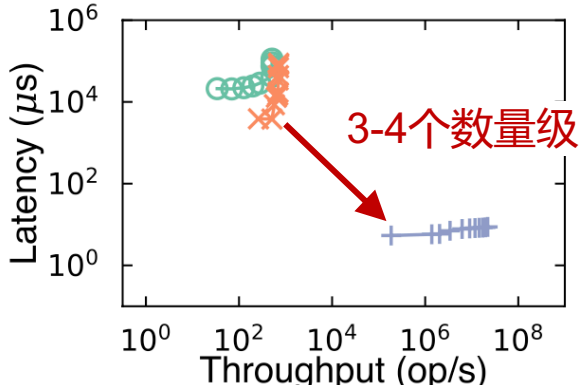


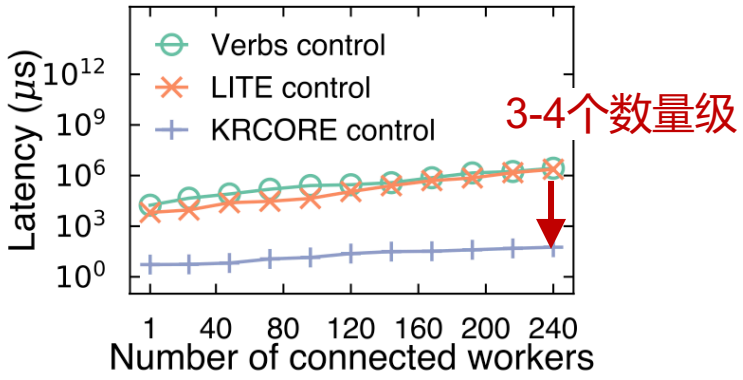
功能 #2：快速自动扩容

KRCore @ATC22：微秒级、高可扩展RDMA控制面

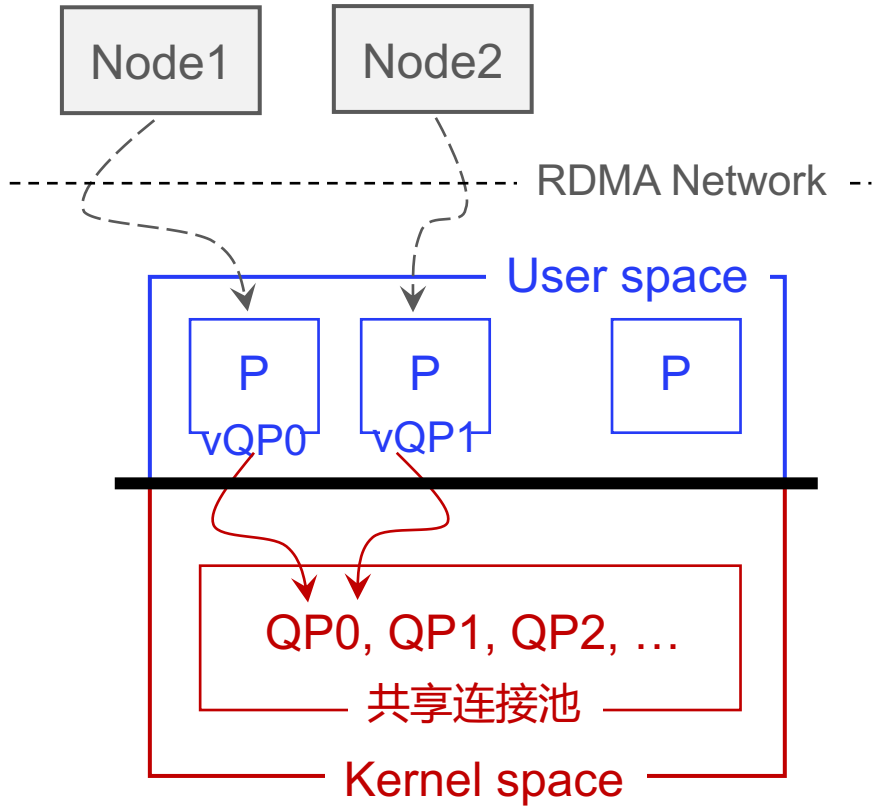
- 内核级RCQP池：安全共享
- DCT QP特性支持：多机共享
- RCQP虚拟化：节省资源







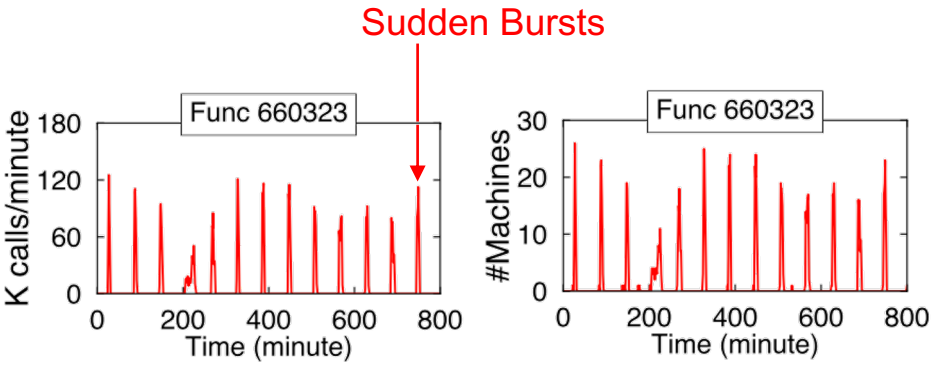
KRCore 系统架构



功能 #2：快速自动扩容

RDMA事务处理在自动扩容时面临的挑战 (#2)

- 计算资源的初始化从offline走向online
- **挑战**：跨节点启动慢 (冷启动)，占响应时间90%以上



自动扩容延迟分析

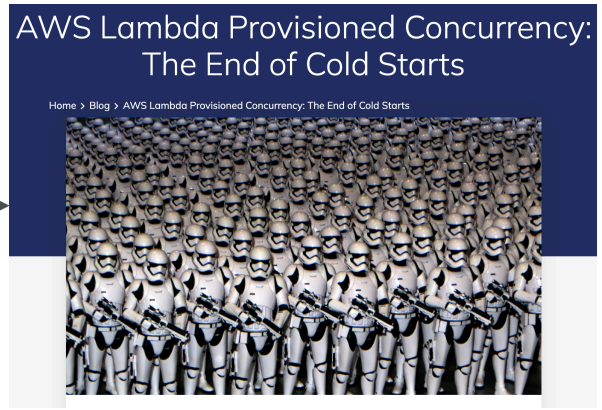
时间

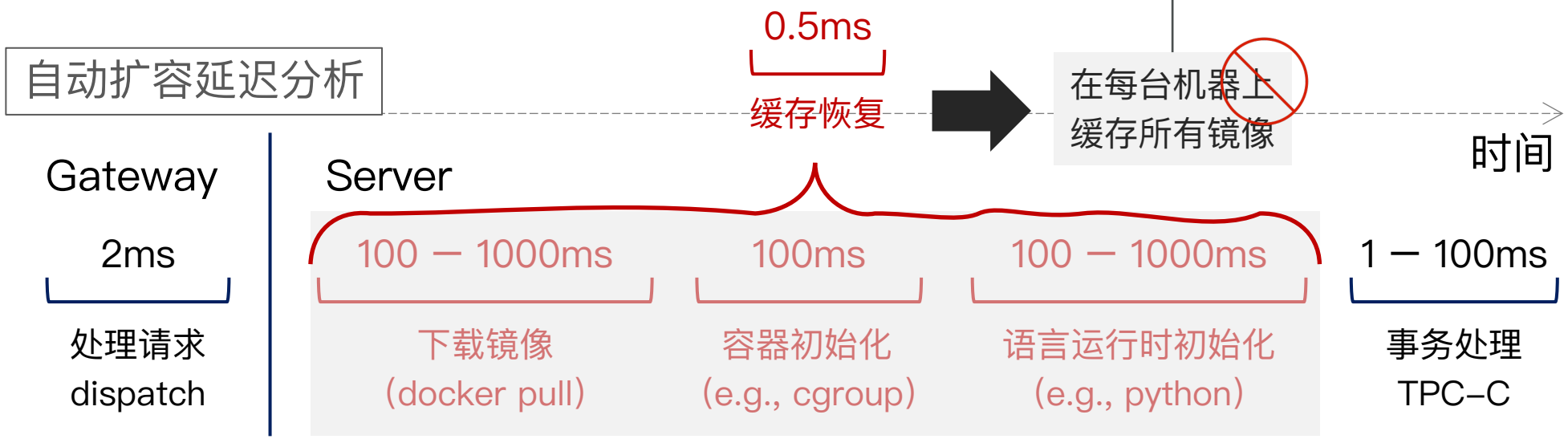
Gateway	Server			
2ms	100 — 1000ms	100ms	100 — 1000ms	1 — 100ms
处理请求 dispatch	下载镜像 (docker pull)	容器初始化 (e.g., cgroup)	语言运行时初始化 (e.g., python)	事务处理 TPC-C

功能 #2：快速自动扩容

RDMA事务处理在自动扩容时面临的挑战 (#2)

- 计算资源的初始化从offline走向online
- **挑战**：跨节点启动慢 (冷启动)，占响应时间90%以上
- 现有基于缓存的热启动方法需要占用大量资源





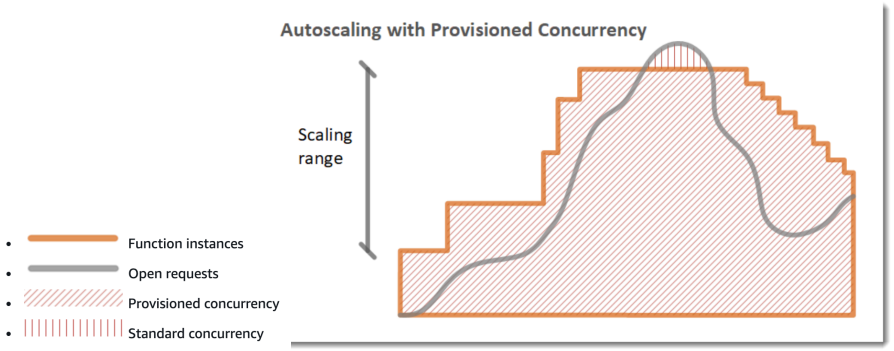
功能 #2：快速自动扩容

RDMA事务处理在自动扩容时面临的挑战 (#2)

- 计算资源的初始化从offline走向online
- **挑战**：跨节点启动慢 (冷启动)，占响应时间90%以上
- 现有基于缓存的热启动方法需要占用大量资源

预置并发：AWS Lambda Provisioned Concurrency

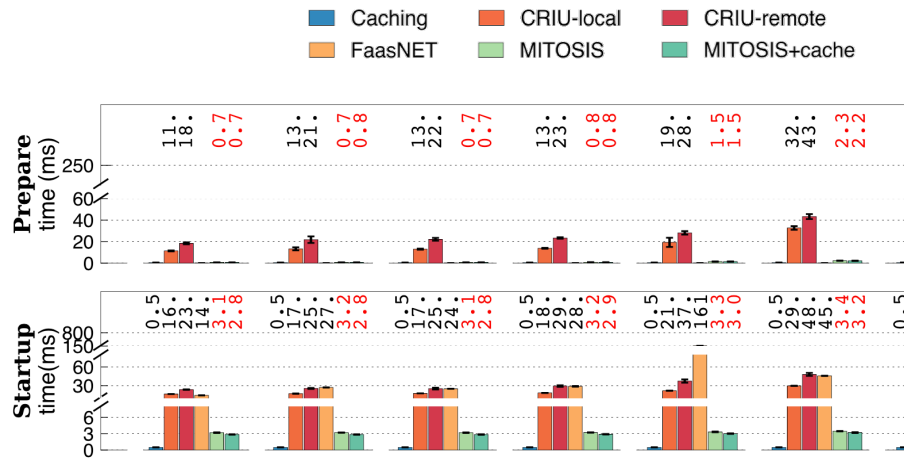
- 额外付费：$0.05 per GB-hour for 1 provisioned concurrency and 1 function (1 version)
- 不支持并行恢复，Provisioned Concurrency = 并发请求数
- 难以准确预估需要的缓存数量 (Provisioned Concurrency)



功能 #2：快速自动扩容

MITOSIS @OSDI'23：基于RDMA的实时自动扩容

- 借鉴内核经典设计：Fork → RDMA-based Remote Fork
- 基于RDMA实现跨节点内存映射、Page fault、CoW
- 支持跨节点、并发热启动 (~ 3*ms*)，且无需预置资源
- 首次实现：1秒内自动扩容至10,000

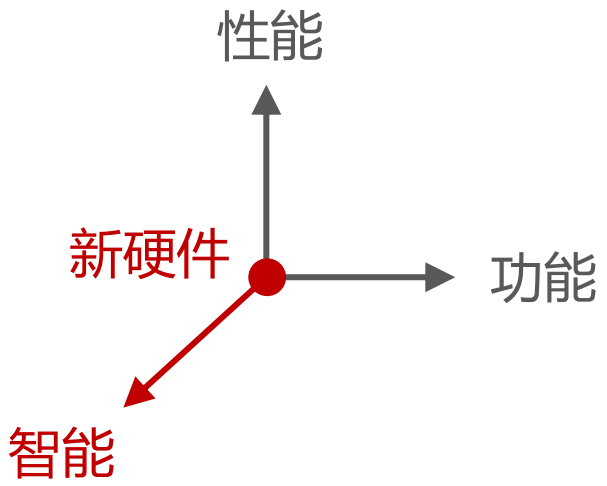


	Coldstart [9, 116]	Caching [63, 120, 93, 100, 119]	Fork [37, 17, 36]	Checkpoint/Restore [117, 37, 114, 20]	Remote fork MITOSIS
Local startup performance	Very slow $(100\,ms)$	Very fast $(< 1\,ms)$	Fast $(1\,ms)$	Medium $(5\,ms)$	Fast $(1\,ms)$
Remote startup performance	Very slow $(1,000\,ms)$	N/A	N/A	Slow $(24\,ms)$	Fast $(3\,ms)$
Overall resource provisioning	$O(1)$	$O(n)$	$O(m)$	$O(1)$	$O(1)$

在(m)台机器上执行1个任务的(n)个并发请求

提纲

- 新硬件驱动分布式事务处理系统研究
- #1：性能
- #2：功能
- #3：智能

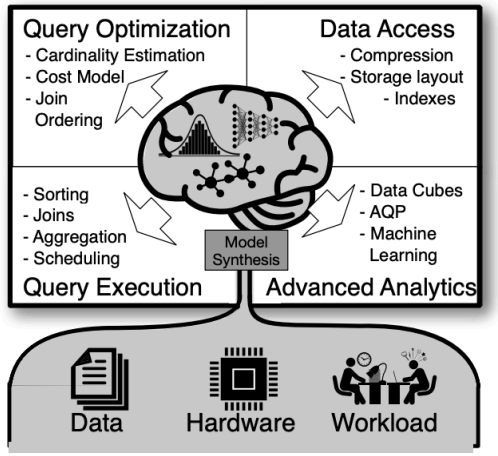


人工智能赋能数据管理：AI ⊕ DB

AI for DB and DB for AI

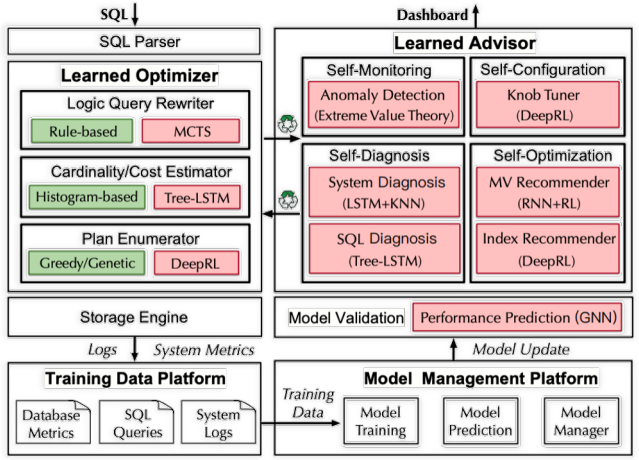
- Data Configuration
- Query Optimization
- Database Design
- Database Monitoring
- Database Diagnosis
- Training Data Generation
- . . .

A Learned DB System



来自 SageDB @CIDR'19

An Autonomous DB System



来自 openGauss @VLDB'21

Thanks to https://github.com/TsinghuaDatabaseGroup/AIDB

For more info, pls. see surveys/tutorials in above github repo

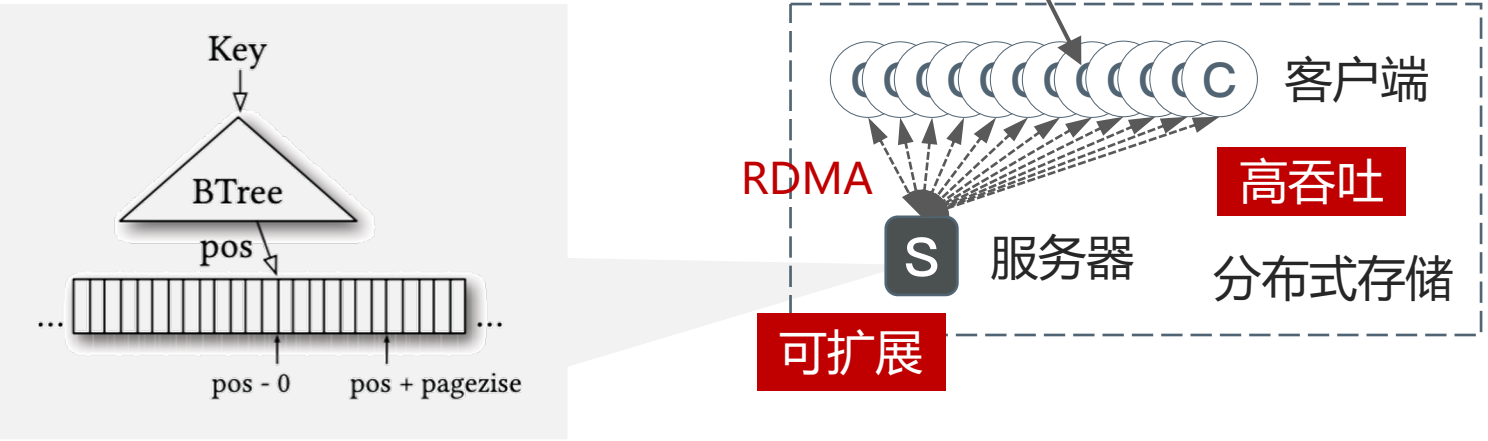
分布式事务处理的存储基础

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)：高可扩展、接口简单

Model (key) → pos　// e.g., B+ Tree

KVS (pos) → value



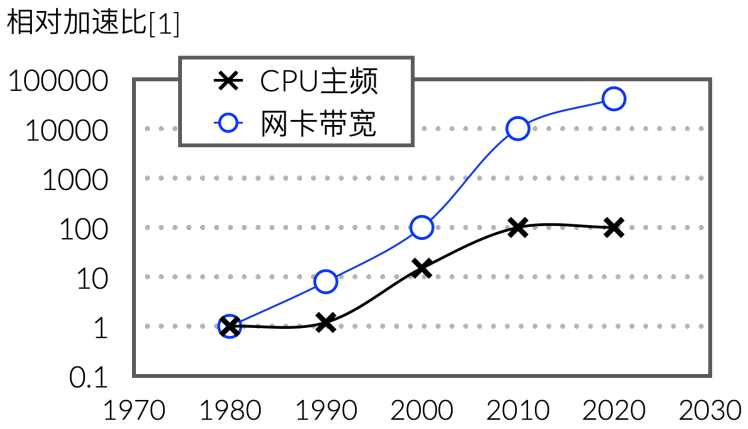
RDMA

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

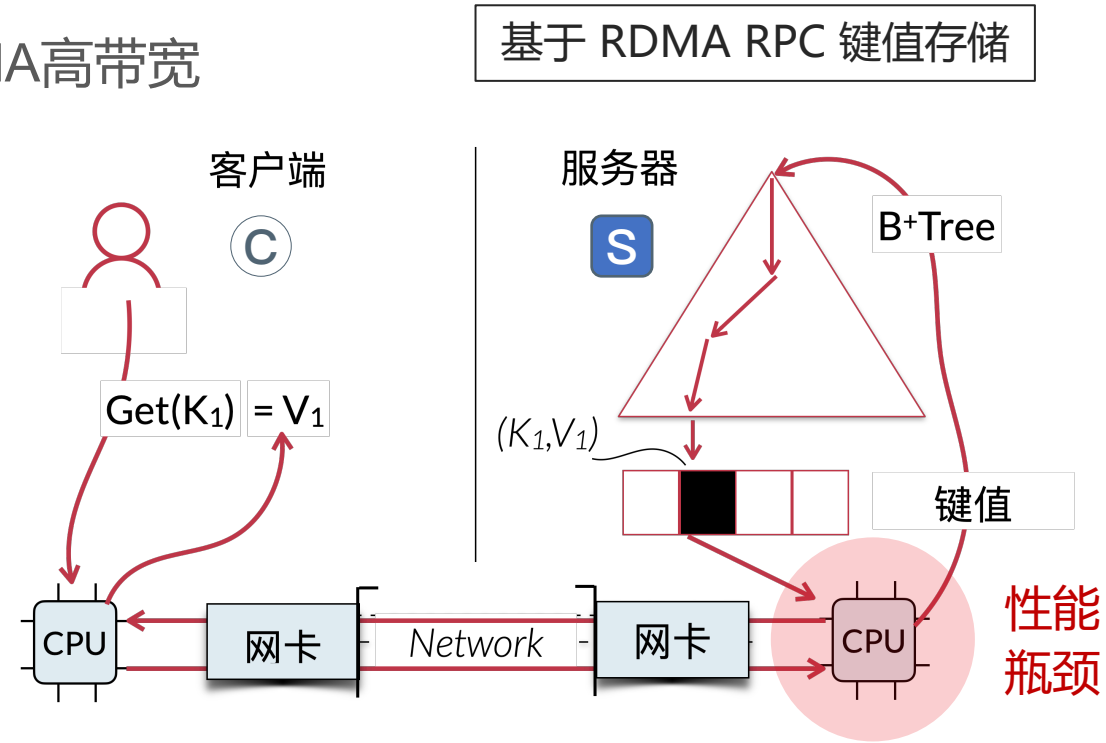
基于RDMA的分布式键值存储

基于 Two-sided RDMA 键值存储设计

- 利用RDMA SEND/RECV加速RPC，eRPC @NSD'I19 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



基于RDMA的分布式键值存储

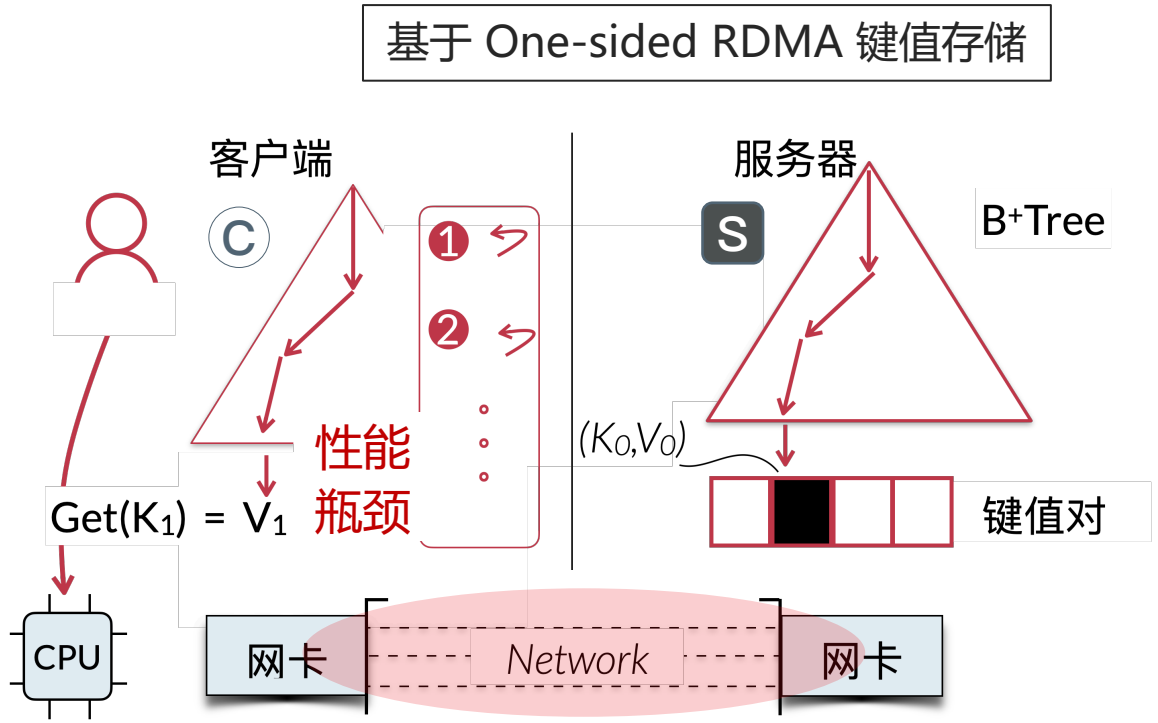
基于 One-sided RDMA 键值存储设计

- one-sided RDMA 语义有限：READ/WRTIE
- 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 7x RDMA

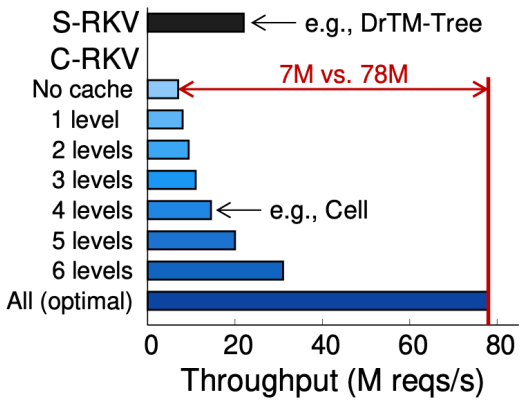


基于RDMA的分布式键值存储

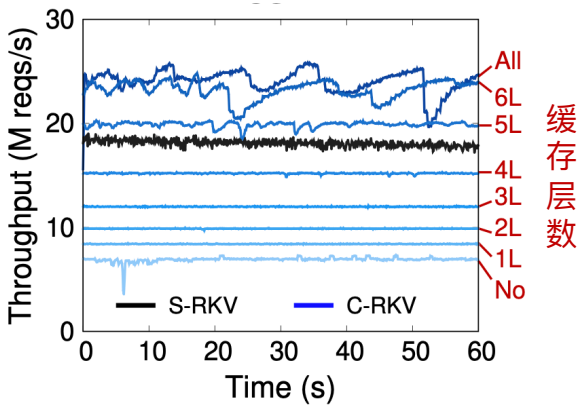
索引缓存：客户度缓存 (同构) 索引，降低遍历索引开销

- 代表性工作：DrTM-KV @SOSP'15、CELL @ATC'16、FaRMv2 @SIGMOD'19
- 有序索引 (B⁺树) 不适合缓存：
  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

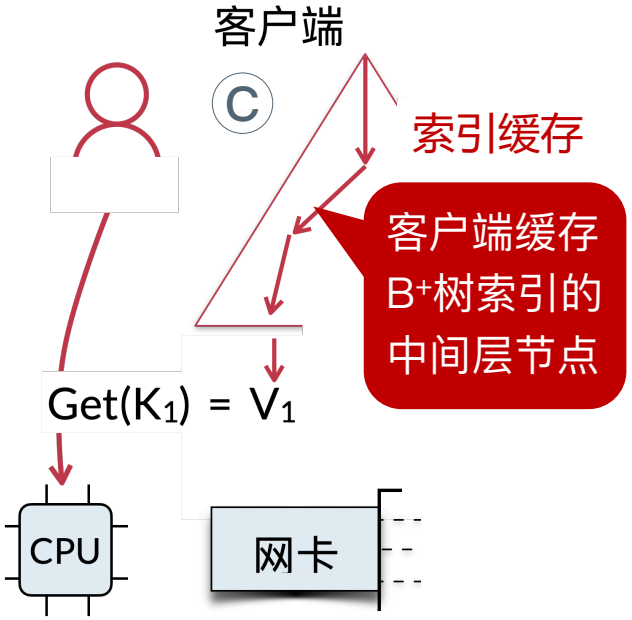
静态负载



动态负载



索引缓存技术



基于RDMA的分布式键值存储

① 基于 Two-sided RDMA 键值存储

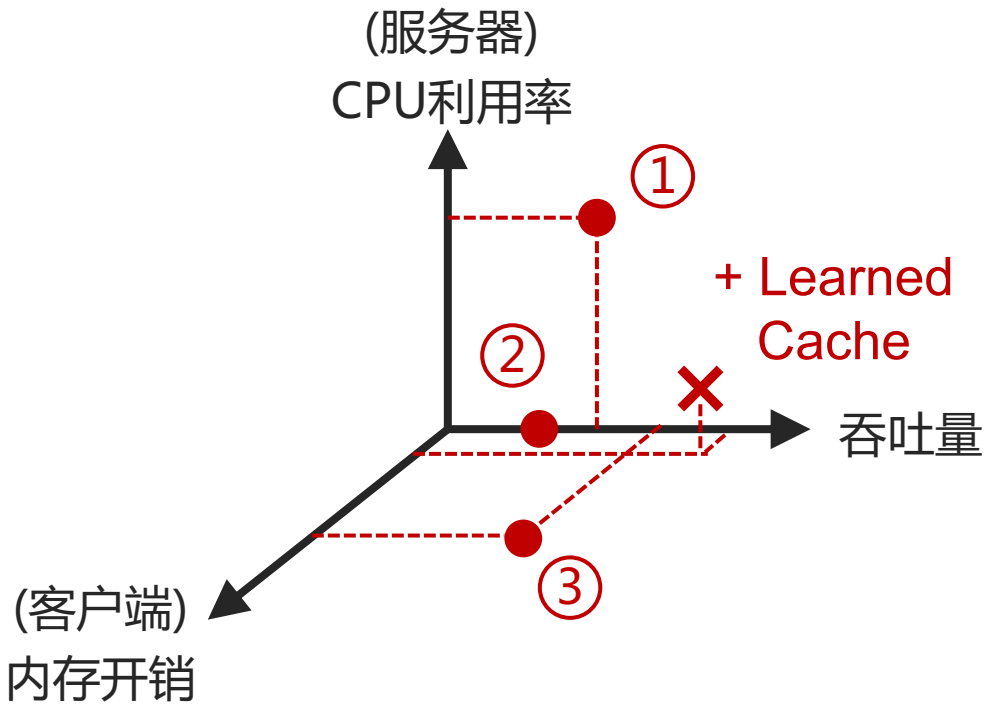
- 性能有限，且受限于服务器CPU

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

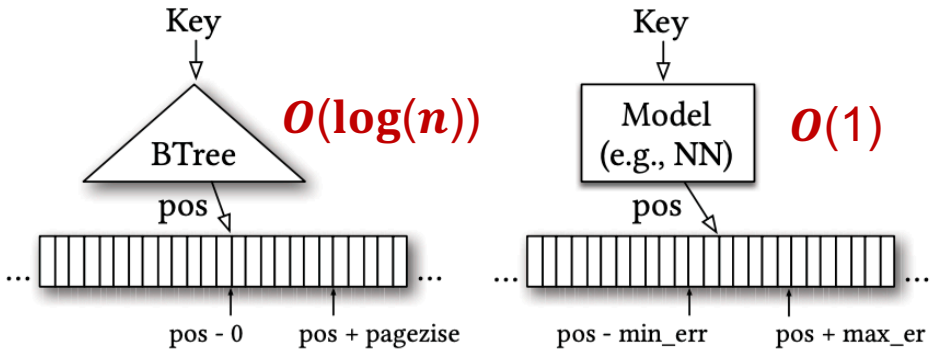


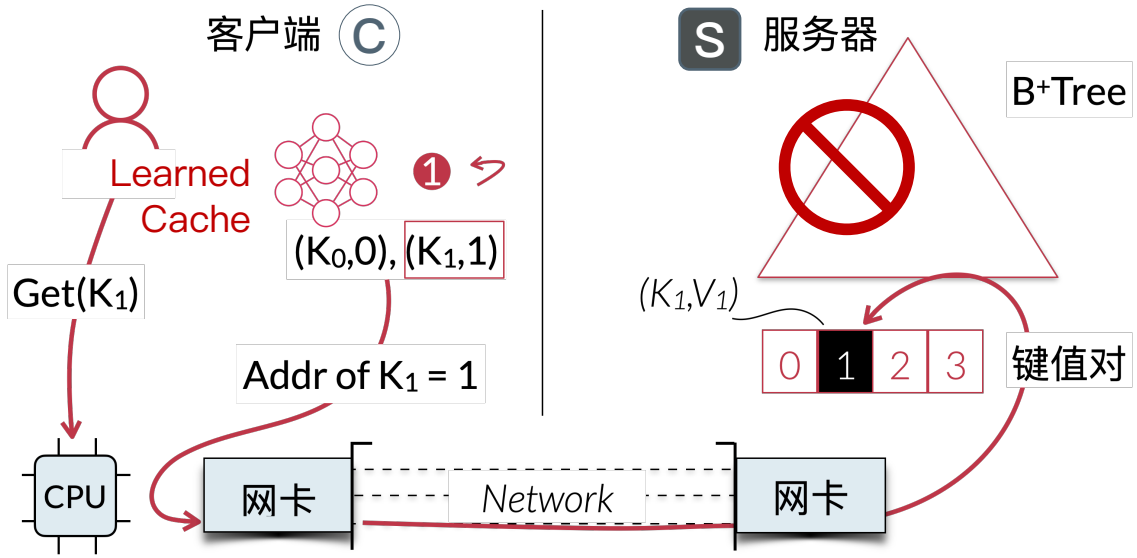
RDMA键值存储的智能缓存

Learned (RDMA) Cache：使用ML模型构建索引缓存

- ML模型缓存优点：占用空间少，无需整树遍历
- RDMA网络操作数：$O(\log(n)) \rightarrow O(1)$
- 优点：低内存开销、高查询效率

“计算 ⟷ 网络+访存”



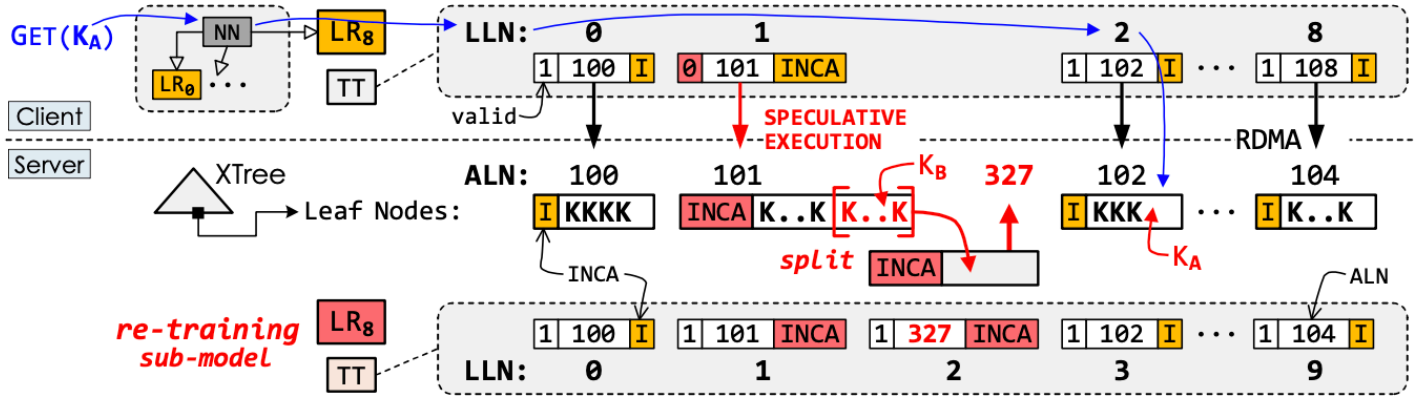


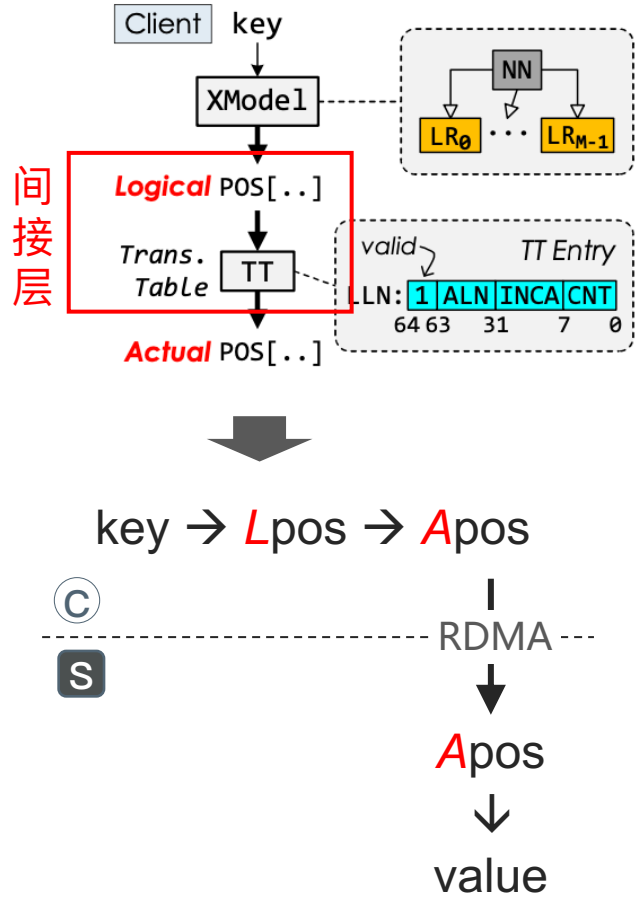
RDMA键值存储的智能缓存

Learned (RDMA) Cache：使用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**：引入逻辑地址 → 解耦索引更新和模型训练

  + 客户端按需更新缓存

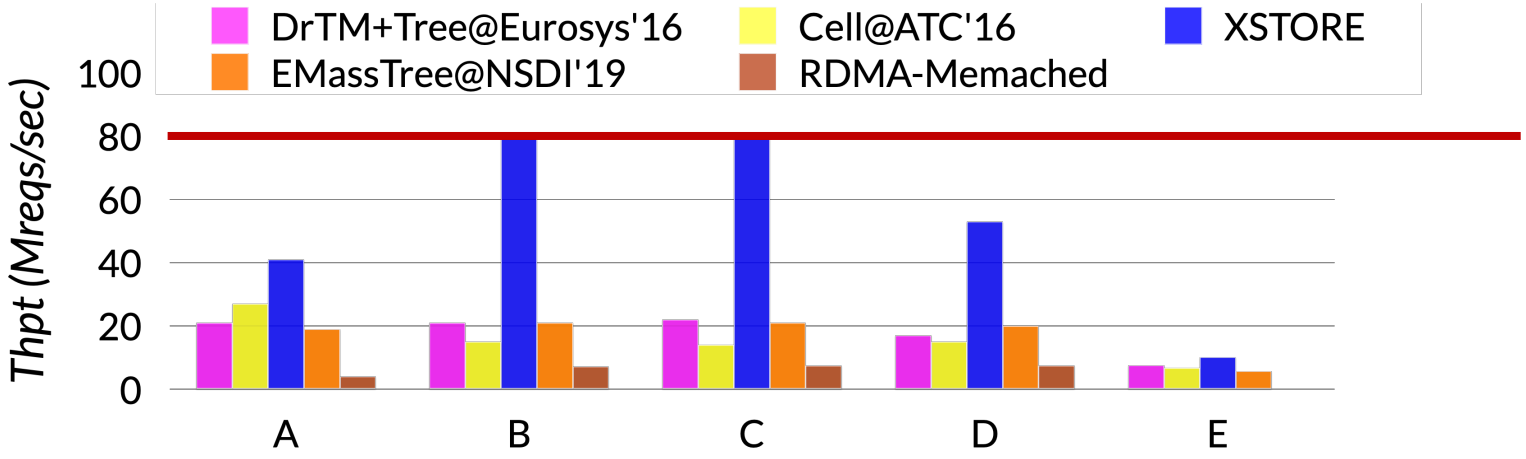




RDMA键值存储的智能缓存

XStore @OSDI'20：基于智能索引缓存的分布式键值存储系统

- YCSB A~E：100 M 键值对 (8B key+8B value)，Uniform / Zipfan 分布
- 分布式事务处理系统 DrTM+H 吞吐量提升2.27X



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

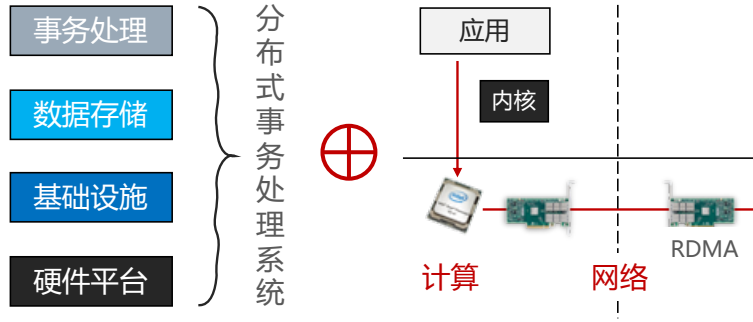
小结 & 感谢

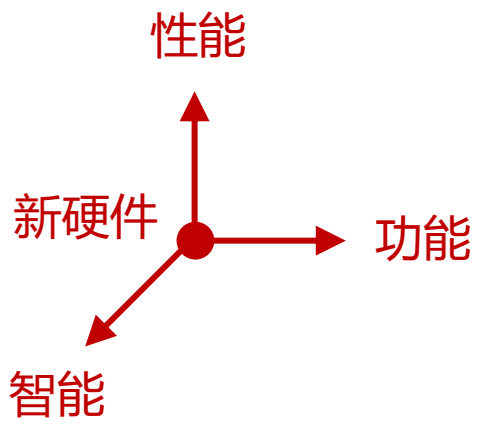
新硬件繁荣，为大数据管理带来机遇

硬件能力单一与应用需求多样间的矛盾造成系统性挑战

新硬件驱动研究案例：“分布式事务处理” 与 “RDMA网络”

从性能、功能、智能三个方面进行了尝试，取得了一些效果





IPADS INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

ipads.se.sjtu.edu.cn

研究工作详情参见：https://ipads.se.sjtu.edu.cn/~rchen