文章编号: 1000-5641(2018)05-0030-11

面向新硬件的数据处理软件技术

涂云山, 储佳佳, 张 耀, 翁楚良

(华东师范大学 数据科学与工程学院, 上海 200062)

摘要: 近年来, 计算机硬件技术飞速发展, 取得了显著的进步, 一些高性能、低时延的新型硬件技术不断涌现, 如: 异构的处理器、可编程的高速网卡/交换机、易失/非易失的存储器等, 给传统的计算机体系结构和系统带来新的机遇和挑战. 然而, 在大数据处理中, 直接将传统的软件技术应用到新型硬件上很难发挥出硬件技术突破所带来的全部潜在性能. 因此, 这就促使我们重新思考传统的软件技术, 以便可以释放硬件进步带来的全部红利. 本文从计算、传输、存储三个方面讨论了面向新型硬件的数据处理软件技术, 梳理和分析了该领域中的相关工作, 总结概述已取得的进展, 分析存在的新问题和挑战, 从而为未来探索数据处理性能"天花板"的研究提供有价值的参考.

关键词: 新硬件; 数据处理; 软件栈

中图分类号: TP392 **文献标志码**: A **DOI:** 10.3969/j.issn.1000-5641.2018.05.003

Data processing software technology for new hardware

TU Yun-shan, CHU Jia-jia, ZHANG Yao, WENG Chu-liang

(*School of Data Science and Engineering, East China Normal University, Shanghai* 200062*, China*)

Abstract: The rapid development of computer hardware in recent years has brought profound technological progress. New hardware for high-performance and low-latency applications (e.g., heterogeneous processors, programmable high-speed NICs/switches, and volatile/non-volatile memory) have been emerging, which bring both new opportunities and challenges to traditional computer architectures and systems. In the case of big data processing, it is difficult to adapt traditional software technology directly onto new hardware, this makes realizing the full potential brought by breakthroughs in hardware technology challenging. Hence, a rethink of traditional software technology can help unlock the benefits brought by advancements in hardware technology. This paper reviews: data processing technology for new hardware from the perspective of computing, transmission, and storage; an analysis of related work in the field; a summary of progress made to date; and new problems and challenges that exist. The study also provides a valuable

收稿日期: 2018-07-04

基金项目: 国家自然科学基金(61732014, 61772204)

第一作者: 涂云山, 男, 硕士研究生, 研究方向为并行与分布式系统.
E-mail: yunshantu@stu.ecnu.edu.cn.

通信作者: 翁楚良, 男, 教授, 博士生导师, 研究方向为并行与分布式系统、新型存储与内存计算、系统虚拟化与系统安全. E-mail: clweng@dase.ecnu.edu.cn.

reference point for future research on exploring the performance ceiling of systems for data processing.

Keywords: new hardware; data processing; software stack

0 引　　言

近年来, 随着智能手机、物联网、人工智能等新兴技术的发展, 使得接入互联网的设备变得越来越多, 产生的数据量也越来越大, 现有的系统已经很难满足业务增长的需要, 促使我们对现有的系统进行改进升级, 以至于它们可以应对技术发展带来的挑战[1-3]. 与此同时, 计算机硬件和体系结构也在不断发展, 像多核/众核、GPGPU、FPGA 等与具体应用相关的异构处理器技术[4-10], 像 RDMA、可编程网卡、可编程交换机等高速定制化的新型网络技术[10-15], 以及像 NVDIMM、Intel® Optane™ 等高吞吐量低时延的新型存储技术[10,16-17]已逐渐成为新兴计算平台的主流技术. 当我们直接将这些高速的新型硬件运用到现有系统中时, 发现传统的软件技术已经很难将这些新型硬件的全部潜能给释放出来. 这是因为, 随着硬件技术的不断进步, 整体系统中硬件延迟大幅降低, 从而使传统系统中开销占比较低的软件开销迅速提升, 成为系统的主要瓶颈.

在处理器方面, 经过几十年的发展, 半导体的制造工艺已经达到了10 nm, 非常接近工艺的物理极限, 以至于摩尔定律很难再继续. 对数据处理应用来说, 硬件提升速度放缓, 与数据增长速度差距逐渐加大, 给处理器带来了新的挑战. 多核/众核、GPGPU、FPGA 等技术的不断发展给数据处理应用带来了新的机遇: 多核/众核技术可以使数据处理内部没有依赖关系的进程或线程真正地并行执行; GPGPU采用的单指令多线程(Single Instruction Multiple Threads, SIMT)可以使大量线程并行地处理数据; FPGA 采用硬件加速的方式使得每一条指令执行得更快, 同时也具有更低的时钟周期, 从而使它的功耗更低. 新兴的处理器技术使得加速数据处理成为可能, 但是它也给传统的软件技术带来新的挑战, 需要做出调整以适应处理器的变化.

在网络方面, 随着以太网技术的发展, 10 Gbps 以太网已经逐渐在数据中心普及, 有的甚至可达 40 Gbps. 此外, RDMA 的硬件价格也在不断降低, 逐渐从传统的高性能计算领域走向数据中心的普通集群. 在这些高速的网络硬件上, 基于内核的 TCP/IP 协议栈的软件开销所占比例已经很大, 变得不可被忽视. Simon Peter 等人[18]通过在内核和用户空间进程中记录时间戳的方式, 分析了网络软件栈的开销, 如表 1[18]所示. 从表中我们发现, 在 Linux 中, 基于内核的 TCP/IP 协议栈花费在报文处理上的时间约为 70%, 主要原因是各层参数的软件多路分解和安全性检查. 为了进一步释放高速网络带来的红利, 内核旁路、零拷贝、轮询等技术被新型网络软件栈广泛采用.

在存储方面, 新型非易失存储器(Non-Volatile Memory, NVM)的出现, 使得 I/O 瓶颈得到很大缓解. Intel® Optane™技术革新了内存和存储, 将存储时延降到了微秒级, 有效地填充了内存和外设之间的存储鸿沟. 与网络软件栈相同的是, 硬件技术的突破无形之中也使软件技术的落后凸显. 为了进一步说明这个问题, 我们使用 fio[19]和 blktrace[20]工具通过实验分析了存储软件各部分的开销, 如表 2 所示. 随着硬件技术的进步, 驱动和设备的访问时延从 3 791μs 显著降到 24 μs. 在 SATA HDD 中, 文件系统的开销占比仅有不到 2%, 而在 NVMe SSD 中, 这部分所占比例迅速攀升到 63%. 这些数据已经反映了改进构建在新硬件之上的存储软件系统的紧迫性. 因此, 提出了一些高效的策略, 如, 多队列、内核旁路、轮询、消除上

下文切换等, 用于探索存储性能的极限.

表 1 网络软件栈开销

Tab. 1 Network software stack overhead

		接收器运行/μs	CPU空闲/μs
报文处理	进	1.26(37.6%)	1.24(20.0%)
	出	1.05(31.3%)	1.42(22.9%)
进程调度		0.17(5.0%)	2.40(38.8%)
拷贝	进	0.24(7.1%)	0.25(4.0%)
	出	0.44(13.2%)	0.55(8.9%)
内核交换	返回	0.10(2.9%)	0.20(3.3%)
	系统调用	0.10(2.9%)	0.13(2.1%)
总计		3.36	6.19

表 2 存储软件栈开销

Tab. 2 Storage software stack overhead

	SATA HDDs/μs	SAS SSDs/μs	NVMe SSDs/μs
文件系统	72(1.86%)	71(45.51%)	60(63.16%)
块设备层	11(0.28%)	10(6.41%)	11(11.58%)
驱动&设备	3 791(97.86%)	75(48.08%)	24(25.26%)
总计	3 874	156	95

可见, 传统的数据处理软件栈已经很难发挥出新硬件带来的全部潜能, 设计实现一套面向新硬件的数据处理软件栈迫在眉睫. 近年来, 大量研究工作也给出了许多不同的思想和方法, 用于优化新硬件下的数据处理软件栈. 与现有的综述论文[1-5,10,16-17]不同的是, 本文试图从计算、传输、存储等三个角度, 对面向新硬件的软件技术研究现状进行全面梳理, 分析该领域中已取得的进展以及存在的新问题和挑战, 从而为未来探索数据处理性能“天花板”的研究提供有价值的参考.

本文的余下内容安排如下: 第 1 节介绍新型计算机硬件的发展和趋势; 第2节梳理和分析基于异构处理器体系结构的软件优化技术; 第3节梳理和分析面向新型高速网络设备的软件优化技术; 第4节梳理和分析针对高吞吐量低时延存储设备的软件优化技术; 第5节讨论存在的新问题和挑战, 并给出总结.

1 新型硬件发展趋势

1.1 异构的处理器

中央处理器(CPU)一直以来被广泛认为是计算机的心脏, 是执行计算的主要硬件设备. 为了适应各式各样的应用需求, 它的功能更具一般化和通用性. 在四十多年的发展中, 从最初的 Intel 4004 到最近的 Intel Xeon Platinum 8180, 单核性能已提高了 35 000 多倍. 然而, 最近十年, 由于半导体制造工艺已经接近物理极限, 单核性能也很难再有较大的提升, 通用 CPU 逐渐开始转向多核/众核的方向. 例如, Intel Xeon Platinum 8180 单颗处理器就多达 28 个物理核. 通过增加核数的方式来提高并行度, 进一步提升了 CPU 的数据处理能力. 面对呈现指数级增长的海量数据, 仅通过增加 CPU 核数, 已很难满足应用对并行计算能力的需求, 于是, 混合 GPGPU 的异构处理器系统就应运而生. 但是阿姆达尔定律(Amdahl's Law)告诉我们, 当一个程序的可串行负载达到一定阈值(或可并行负载低到一定程度)时, 无论怎么增加 CPU 核数也不会带来很大的性能提升, 有时候甚至可能因为竞争资源导致性能下降. 为了进一步提升系统的整体性能, 引入 FPGA, 采用硬件方式加速数据处理成为新的趋势.

目前主流的数据中心服务器采用 CPU+GPGPU+FPGA 的异构处理器体系, 如图 1 所示, GPGPU 和 FPGA 通过 PCIe 总线与 CPU 相连, CPU 与 CPU 之间通过 QPI 连接. GPGPU是一种支持通用计算的图形处理器, 不仅拥有传统显卡的图形处理能力, 也具备类似 CPU 的通用计算能力. 目前, 最新 NVDIA TESLA P40 采用帕斯卡架构, 单精度浮点运算能力高达 12 TeraFLOPS, 整数运算能力高达 47 万亿次/s. 如果一台服务器配置了 8 块 TESLA P40, 则它的处理能力完全可以与 140 台 CPU 服务器的性能媲美. 然而, CPU和GPGPU的能耗也是一个不容忽视的问题. 因此, 也有些服务器搭载了主频更低、能耗更少的 FPGA 处理器, 通过硬件方式加速数据处理, 提高整体性能. 随着 OpenCL 的发展, FPGA的编程也变得越来越简单, 仅需要通过 C/C++添加适当的 pragma 就能实现 FPGA 的编程, 这也从某种程度上推动了 FPGA 进一步的发展.

Inter Xeon Platinum 8180　　NVIDIA TESLA P40　　Xilinx VC709 Evaluation boards

图 1　异构的处理器

Fig. 1　Heterogeneous processors

1.2　可编程的高速网络

遵循端到端原则的 TCP/IP 协议栈, 经过三十多年的实践检验, 被证明是一种高效的结构, 它既降低了因特网的复杂度也易于增加新的应用. 但是, 一些新兴应用所需要的网络服务, 例如, 服务质量保证、多播支持、移动性支持等, 无法仅由端系统提供, 需要中间的路由器/交换机提供支持. 与此同时, 处理器主频的发展脚步慢了下来, 开始转向多核和并行, 然而程序的可并行度又受到不可并行部分的占比所限制. 因此, 人们开始将 FPGA 与网卡集成, 与交换机集成. 这既能满足新兴应用的需要, 也可以卸载一部分 CPU 的负载到网络设备上, 进一步提高整个系统的数据处理能力.

目前, Netronome 公司的 Agilio SmartNICs[21]可以在通用的 x86 商用服务器上使用, 并且可以支持不同速率的以太网, 例如, 10GbE、25GbE、40GbE以及50GbE. 这种带有计算能力的网卡不仅给自身带来了更大的灵活性, 也缩短了传统的网卡到内存再到 CPU 的数据处理路径, 仅仅需要网卡与内存之间的交互就足够了. 用户可编程的交换机如雨后春笋般不断涌现, 例如, Barefoot Tofino[22], 一个 6.5Tb/s(65x100GbE 或 260x25GbE)全可编程的以太网交换机, Cavium Xpliant 系列[23], 是面向数据中心的可编程的以太网交换机, 可以适用于 1 G/10 G/25 G/100 G 网络. 这些可编程的网络技术使得数据中心依靠传统的中间件来解决的问题, 如: 负载均衡、地址翻译、探查 DDoS 攻击、设备迁移, 可以高效地被折叠进可编程的网络中, 进一步改善了整体的应用性能, 同时, 也使得网络变得更加灵活、应用具体化.

1.3　新型非易失存储

基于 DRAM 的易失性存储器的访问时延在几十纳秒左右, 而传统的机械硬盘的访问时延在 1-10 毫秒之间, 以至于内存和外设之间的差异达到几十万倍, 形成了很大的鸿沟. 近些年, 高速发展的闪存技术仅仅只能将外设的时延降低到几百微秒, 并不能填补内存和外设的速度差异. 一些新型的非易失性存储器(NVM)开始出现, 寻求使用新型替代存储介质来突破 I/O 瓶颈, 典型的存储介质有: 相变存储器(PCM), 利用硫属化物材料在“无定形相(高阻态)”和“结晶相(低阻态)”两种状态间阻值的变化进行数据存储; 自旋矩传输磁存储器(STT-RAM), 利用磁隧穿结(MTJ)来存储数据的工艺, 在 MTJ 中, 隧穿绝缘体薄层被置于两层强磁性介质中; 电阻式存

储器(RRAM), 利用阻值变化进行数据存储的技术. 不同存储器之间的性能对比, 如表 3[24-28]所示.

表 3 不同存储器的性能对比

Tab. 3 Performance comparison of different memory/storage technologies

	PCM	STT-RAM	RRAM	SRAM	DRAM	Flash	HDD
读时延/ns	48	10	116	10	55	25 000	3 000 000
写时延/ns	150	10	145	10	55	200 000	3 000 000
能耗	中	中/低	低	非常低	中	高	非常高
使用寿命/次	10^8	10^{12}	10^6	10^{15}	$> 10^{15}$	10^5	$> 10^{15}$
非易失性	是	是	是	否	否	是	是

同时, 英特尔和美光也在芯片的制造工艺方面做出了革命性的创新, 研发出了新型的 3D 堆叠技术, 将传统的二维芯片制造推到了三维空间. 这种堆叠技术很好地解决了内存控制器瓶颈问题, 降低了访问时延, 提高了设备的带宽. Intel® Optane™系列产品[29]就是这项技术市场化的产物, 它有两种不同规格, 分别是, Optane 内存和 Optane 固态硬盘. 它们将设备的访问时延降到了十微秒量级, 很好地填补了内存和外设之间的存储鸿沟. 同时, 这些先进的技术也给构建在硬件之上的数据处理软件带来新的机遇和挑战, 促使整个软件栈作出调整来适应高速的存储硬件.

2 新型计算技术

2.1 多核

佐治亚理工学院的研究团队针对多核系统提出一种可拓展的定序原语——ORDO[30], 改善了现有的并发控制中采用原子指令解决高速缓存行冲突问题的可拓展性. 在多核、多 CPU 的服务器上, 定序需要使用非常昂贵的原子指令, 这也很大程度上限制多核结构的并行性. 因此, ORDO 采用了一个全局恒定的同步硬件时钟, 并且它也是一个对多核架构友好和具有不确定性窗口的时钟. 此外, 作者也通过使用他们所提出的 ORDO 原语, 做了大量的实验, 验证基于四种不同架构(Intel Xeon、Xeon Phi、AMD 和 ARM)的算法和系统软件(例如, RLU、OCC、Hekaton、TL2 和进程分叉)的可拓展性是否得到改善. 实验结果表明, 这些不同的算法和系统软件均得到了很好的改善.

麻省理工学院的研究团队提出了一种新颖的文件系统设计——ScaleFS[31], 它采用与每个 CPU 核绑定的操作日志, 解耦了在内存和磁盘中的文件系统. 这种设计使得高度并发地访问内存中共享的数据结构成为可能, 这是因为并行的存/取操作没有缓存冲突, 可以并行执行, 因此, 可以使文件系统的性能得到完美地线性扩展. ScaleFS 在每个 CPU 核对应的日志中记录相应的操作, 以便它们可以被延迟, 直到 fsync 系统调用才更新到磁盘中. fsync 系统调用将会合并每个 CPU 核的日志, 并将这些操作应用到磁盘上. ScaleFS 通过使用基于时间戳的线性化点、计算操作的依赖关系和吸收操作等技术, 既保证了良好的执行合并性能, 又确保了最终操作的正确性.

除了上述 ORDO 和 ScaleFS 在高速缓存行方面的优化以外, 随着内存计算技术的兴起, 也有许多的研究者尝试在 NUMA 架构方面做出努力, 改善访问本地和远端内存不对称的问题, 进一步提高系统的整体性能, 如, 内存分配与调度[32]、并行查询评估[33]等. 通过上述这些工作, 我们发现目前基于多核体系结构的软件优化技术, 主要采用减少软件系统中数据冲突方法, 并提高阿姆达尔定律中可并行部分的占比, 从而实现更高的并行性, 达到加速上层数据处理应用的目的.

2.2 GPGPU

由于 OLAP 中的任务执行模式非常符合 SIMT 特征, 因此, 在 OLAP 系统中实现数据并行计算是较为容易的. 大量的数据可以被切分成多个数据块, 将数据块作为并行执行的基本单位. 所以 OLAP 的场景适合 GPGPU 的处理模型, 开发者可以将多个数据块分配到多个不同线程块上并行执行. 随着一台服务器中 GPU 数量的不断增加, PCIe 的带宽成为了新的瓶颈. 英伟达公司研发出新型 NVLink 技术[34], 这也是 GPU数据库与 PCIe 传输带宽瓶颈的一个解决方案. 在CPU/GPGPU 的异构体系结构中设计任务调度的框架, 以及设计更为高效的算法来优化数据操作是当前研究的重点.

GDB[35]是一个基于 GPGPU 开发的完整的查询处理系统, 包含了一系列基于 GPGPU 架构的高度优化的操作原语和关系算子, 并设计了代价估算模型来估算 GPGPU 上的查询执行时间. 马格德堡大学的研究团队提出了一个针对列存储数据库设计的混合查询执行引擎——HyPE[36], 能够综合 CPU 和 GPGPU 的特点, 生成混合式查询计划, 并基于代价估算模型对查询计划进行优化. 在异构处理器查询执行时间的估算中, HyPE采用基于学习的自调整决策模型, 在输入数据集和结果执行时间之间找到关联, 并采用统计方法进行学习. 当有足够的观察对象时, 统计方法将会拟合出一个近似的时间估算函数. 麻省理工学院的研究人员开发的 MapD[37] 是成功商业化并更具实用性的新型大规模并行数据库, 它利用 LLVM 编译框架将 SQL 语句编译成 GPU 原生代码加速计算. 除了上述关于查询引擎的工作, 还有基于现有的大数据分析和数据库平台进行改进和优化的工作, 如 Spark-GPU[38]、PG-Strom[39]等.

OLAP 型负载关注在大量数据情况下的单条查询的分析效率, 这与 SIMT 的数据并行执行模型非常契合, 使得 GPGPU 在 OLAP 应用中可以很好地并行执行. 相反地, OLTP 型的负载要求在短时间内处理大量事务, 对处理的并行度以及吞吐量的要求更高. 因此, GPGPU 单指令多数据流的方式不适合 OLTP 应用. 为了改善这种不适应性, 南洋理工大学和香港中文大学提出并实现了一个面向 OLTP 应用的 GPGPU 事务执行引擎 GPUTx[40]. GPUTx 将多个事务聚集到一个块上, 作为一个任务执行, 借此获得更高的吞吐量. 块执行模式只支持预先定义的事务类型, 需要给出明确的执行策略. 开发者根据块中的事务是否按其时间戳顺序递增执行来判断一个块事务模型是否执行成功.

2.3 FPGA

现阶段, 在数据处理领域, 使用 FPGA 加速计算主要是针对某些特定的场景和算法. David Sidler 等人[41]探讨了如何在数据中心中优化和改进基于 FPGA 的 TCP/IP 协议栈, 以及极大程度地减少尾延迟问题. Kaan Kara 等人[9]通过使用 FPGA 加速基于哈希的数据分片, 将 FPGA 芯片作为协处理器放置在其中的一个插槽上, 采用 QPI 技术与 CPU 相连, 首先将带分布的数据加载到高速缓存行中, 然后在 FPGA 上执行分片写回操作. David Sidler 等人[8]采用一个基于Intel Xeon+FPGA 的混合架构加速传统数据库 MonetDB 中的模式匹配查询, 首先在 CPU 上进行作业调度和分配, 然后在 FPGA 上实现了一个状态机, 执行具体的状态转换和正则匹配.

除了上述这些特定的工作以外, 也有研究者尝试在 FPGA 或 FPGA+CPU 中实现一个完整的软件系统. Zsolt Istvan 等人[7]基于FPGA实现了一个就近的键值数据存储引擎Caribou. 它采用了 Cuckoo 哈希表的方式存储键值对, 使用基于位图的方式分配和管理 BRAM 和 DRAM 的内存, 在 FPGA 上采用执行流水和数据并行的两种策略加速查询操作. 张铁赢等人[11]充分利用异构计算的体系结构进行软硬协同设计, 实现了一个兼容 MySQL 的关系型数据库系统 X-DB. X-DB 采用 FPGA 的流水线和 Intel QAT 技术, 实现数据的解压缩. IBM 的数据仓库分析系统 PureData[42], 其前身是 Netezza, 采用 FPGA 和多核相结合的协同处理技术加速计算, 同时这项技术也逐渐被商用 DB2 所采用[43]. 我们相信, 随着计算机体系结构的发展, 会有越来

越多的系统采用 FPGA 来加速数据处理, 进一步减少整个软件栈的开销.

3 新型网络技术

3.1 数据面与控制面

在传统系统内核中, 将数据面和控制面混在一起, 导致了许多不必要的软件开销, 如, 系统调用、内存拷贝等. 随着高速网络的发展, 数据面和控制面分离成为一种新的趋势, 因此, 有许多研究者在此方面做出努力和尝试. 华盛顿大学和苏黎世联邦理工学院研究团队提出了操作系统仅作为控制面的方案 Arrakis[18], 将传统的 I/O 数据路径从内核中隔离出来, 提供一个用户级的库与硬件设备直接交互, 访问控制和硬件配置等工作仍然保留在内核之中, 这样提高了系统的安全性. 斯坦福大学和洛桑联邦理工学院研究团队提出了一个高吞吐量和低延迟并且数据面和控制面分离的操作系统 IX[44], 使用硬件虚拟化技术将内核的管理和调度功能(控制面)与网络处理(数据面)分开, 提供了一个零拷贝、原生的编程接口, 使得应用可以直接操作 I/O 数据.

Intel 工程师们也开发出了一套用户态数据面套件 DPDK[45], 它提供了高效灵活的包处理解决方案. 在 DPDK 中, 开源社区的工程师们实现了一套多核和大内存页的框架, 以及一整套轮询模式的驱动和无锁的环形缓存. 韩国科学技术院和普林斯顿研究团队, 也针对多核系统实现了一套可拓展的用户态 TCP 协议栈——mTCP[46], 采用不共享文件描述符和批处理的方式进行报文收发, 同时也提供一套支持类似于传统内核接口的编程模型, 方便了上层应用. 这些将数据面和控制面分离, 以及将传统基于内核的数据处理迁移到用户空间的研究工作, 缓和了传统内核模式下上下文切换、内存拷贝、中断等带来的软件开销, 使得高速网络的潜能得到进一步释放.

3.2 远程直接数据存取

斯坦福的研究团队[47-51] 提出了一个基于 Infiniband 高速网络的新型内存键值存储系统 RAMCloud, 它使用基于日志结构的内存管理策略改善高使用率情况下内存分配与回收效率, 将读操作时延降到 5 μs, 写时延降到了 15 μs. 此外, 他们也在 RAMCloud 系统中实现了辅助索引, 为访问一个键值对提供了更多的索引方式. 同时, 他们还在系统中实现了线性化, 提高了系统的一致性, 使得其可以满足更多更复杂的应用场景. 与 RAMCloud 类似, FaRM[52]和 MICA[53]都采用了 RDMA 技术优化数据传输, 提高系统的整体性能. 但是, 不同的是, FaRM 和 MICA 更加关注系统的整体吞吐量, 在此方面做了许多优化工作, 使得 RDMA 高吞吐量的特性进一步得到体现.

除了上述这些 NoSQL 系统, 传统的关系型数据库系统也开始使用 RDMA 技术. 阿里巴巴数据库团队使用了基于以太网的 RDMA 和 NVMe SSD 技术实现了关系型数据库 PolarDB[54], 它借助于 RDMA 提升网络交互的效率, 利用 NVMe SSD 来提升单机 I/O 的性能, 将计算层和存储层分离. 此外, 也有一些研究者使用 RDMA 技术优化查询处理[55]、事务处理[56]等操作.

3.3 可计算的网络

随着 CPU 处理能力的提速趋于平稳, 一些研究者利用两端的 RDMA 技术去加速通信. 然而, 当引入 RDMA 以后, CPU 的处理能力又成为了新的瓶颈, 限制了数据处理应用的整体性能. 因此, 中国科学技术大学和微软研究团队提出了一个与可编程网卡相结合的内存键值存储系统KV-Direct[15]. 它将一部分的 CPU 负载卸载到可编程的网卡上, 缩短数据处理的路径, 进一步提高系统的整体性能. 此外, 约翰霍普金斯大学和 Barefoot 网络的研究团队, 使用可编程的网络实现了一个驻留在交换机的缓存 NetCache[14], 使得读数据的路径变得更短, 提高了系统的读性能. NetCache 的核心是一个数据包处理流水线, 利用现代可编程交换机在交换机数据面中有效地进行探测、索引、缓存, 同时, 也保证了缓存一致性.

也有一些工作将可计算的网络与非易失性存储器相结合, 如, PASTE[13]. PASTE 是由日本电气公司欧洲实验室和比萨大学的研究团队共同研发, 是一种新的 NVMM 网络编程接口, 支持标准的网络协议, 如 TCP 和 UDP. 同时, 它也将可编程网卡与 NVMM 紧密结合, 一旦数据经过网卡的 DMA 到达主机内存, 就将被永久驻留, 不需要再次复制. 此外, 微软 Azure 云也在开始使用基于 FPGA 的定制可编程网卡 SmartNIC[12]来加速云上的数据处理应用.

4　新型存储技术

4.1　一致性与编程模型

非易失性内存的一致性问题源于缓存到内存的乱序写, 以及 NVM 中数据的部分写. 因为缓存中的数据被刷入内存时是乱序执行的, 在故障重启后的 NVM 中, 可能存在只写了一半的数据或非连续的数据. 为了解决 NVM 的一致性问题, 学术界做出了很多努力.

NV-Heap[57]致力于提供安全的接口来构建持久化对象, 它保证了指针安全, 提供了一组简单的原语, 例如, 持久性对象、专用指针类型、内存分配器, 来保证一致性. 不同于 NV-Heap 基于对象的存储, Mnemosyne[58]支持多字长的事务. Mnemosyne 提出了一些编程接口来创建、管理非易失性内存, 以及保证故障发生时数据的一致性. 它提供了原语用于直接更新持久化数据, 并通过轻量的事务机制保证数据的一致性更新. HEAPO[59]设计了本地的持久化堆结构以及相应的调用接口, 利用 undo 日志和事务机制来保证写入操作的一致性. PMDK(Persistent Memory Development Kit)[60]提供了一系列的库来简化基于存储级内存的编程工作. SAP 的研究人员尝试将 NVM 应用到商用 HANA 数据库中, 他们针对如何判断 NVM 中数据的有效与否、系统重启后指针重定向以及 NVM 中无效数据的垃圾回收等问题给出了合适的解决策略[61].

4.2　精简的机制

传统的数据处理机制面向磁盘等慢速设备设计, 常常通过一些空间换时间、冗余操作来换取系统的一致性、持久性或高性能性. 当 NVM 技术被引入后, 这些原本有效的软件机制的开销也变得越来越不可忽略, 逐渐成为新的性能瓶颈.

基于磁盘的存储系统通常采用预写式日志(Write-Ahead Logging, WAL)来保证数据的持久性, 提升写操作的效率. WAL 机制保证了在事务被提交之前相关的事务日志已经被安全地持久化, 防止因为断电或其他系统故障导致数据丢失. 因为磁盘存在随机写和顺序写速度差较大的问题, 通过WAL机制将随机写数据转化为顺序写日志, 极大地提升了写操作的执行效率. 与此同时, WAL 机制同样带来了数据冗余、写入放大以及资源消耗等问题[62-63]. 考虑到 NVM 的读写速度比磁盘快了一个数量级, 并且随机写和顺序写差异较小, 存储系统可以采取就地更新的方式, 在写操作执行期间, 不写 redo 日志, 而是直接将更新后的数据写入NVM [64-67].

Joy Arulraj 等人[63]讨论并分析了就地更新、复制更新以及基于日志的更新方式三种情况下的系统恢复机制. 因为 NVM 的非易失性, 系统重启后, 所有的更新数据仍在NVM中, 不需要经过回放日志以及构造内存表的阶段, 大大减少了系统恢复的时间. 也正因为 NVM 的非易失性, 导致NVM 中存在大量无效的未提交数据, 可以通过能感知 NVM 的分配器[60]来回收无效空间.

I/O 速度显著提升, 使得网络交互的代价逐渐成为数据复制过程中的新瓶颈. 主要的改善策略包括减少网络交互的次数、引入快速的网络硬件以及减少不必要的拷贝开销. Mojim[68]使用了一套高度优化的网络栈以及复制策略来减少数据复制过程中的开销. 它使用两层结构, 第一层的节点之间保持强一致性, 并通过 RDMA 进行消息传递. 第一层和第二层节点之间的数据同步具有多种模式, 可以在可用性、一致性以及性能之间权衡选择.

4.3 轻量化的系统

除了精简不必要的软件机制, 相较于快速的非易失性存储设备, 存储软件栈本身存在过于厚重的问题. 一些研究工作致力于对存储栈进行优化, 使其更加轻量化.

SPDK[69]是由 Intel 开发的一系列库来帮助用户实现高效的存储应用. NVMe 驱动程序是其中的一个库, 它提供了一种较为直接的、零拷贝的数据传输方式. NVMeDirect[70]作为一种用户态的存储框架, 为应用提供了直接的操作接口来访问 NVMe 设备. 不同于 SPDK, NVMeDirect 提供了更加灵活的队列管理机制, 使得不同应用可以选择不同的队列调度以及完成方式. LightNVM[71]是一个位于内核空间的 Open-Channel SSD 子系统, 提供了一系列的操作接口使得主机可以直接地访问底层的 Open-Channel SSD. 还有一些工作通过绕过 Linux 内核[72]、使用轮询的方式替代中断方式[73]、消除不必要的上下文切换[74-76]来减少存储栈的开销.

作为一个用户态的日志结构的文件系统, NOVA[77]通过为每一个 inode 提供独立的日志来提升并行性, 并且将用户操作的文件数据从日志中分离来缩小日志大小, 进而减少垃圾回收的代价. Aerie[78]是一个面向存储级内存设计的文件系统, 它绕过了 Linux 内核, 极大地减少了存储栈中文件系统层的开销. DevFS[79]创造性地将文件系统从操作系统中搬到存储设备中, 使得文件系统和存储设备之间的访问路径更短. 但是另一方面, 缺少主机依托、CPU 资源受限、设备内存容量小等问题使得存储设备层面的文件系统仍然存在诸多使用限制.

5 结 束 语

随着新型硬件技术的发展, 构建在其之上的传统软件技术已经变得不再适用, 很难发挥出硬件变革带来的全部性能. 为此, 传统的软件系统需要做出改进以适应高速的计算、网络以及存储设备. 在计算方面, 将数据处理软件与异构计算的体系结构相结合, 根据计算任务的特征, 将任务的不同部分放到恰当的处理器上, 以达到充分发挥出异构计算的性能. 在网络方面, 高速和可编程的网络技术潜移默化地改变上层的软件系统, 使网络与应用的关系变得更加紧密, 从而减少了一些不必要的开销, 甚至可以探索应用具体的网络计算技术. 在存储方面, 新型非易失存储器的出现, 改变着现有系统的存储结构, 给上层应用带来许多可能, 同时也带来像一致性、持久性、安全性等诸多挑战. 未来的研究工作, 将基于学界、工业界已有的分布式数据处理软件系统, 探索与新型硬件技术相结合、软硬件协同设计等技术, 降低软件不适应引入的额外开销. 总而言之, 硬件技术的不断变革提升了数据处理系统的性能, 相应地, 软件技术需要针对新型硬件的特性做出适当的改变, 将硬件潜在性能发挥到极致.

[参 考 文 献]

[1] 陆游游, 舒继武. 闪存存储系统综述[J]. 计算机研究与发展, 2013, 50(1): 49-59.
[2] 程学旗, 靳小龙, 王元卓, 等. 大数据系统和分析技术综述[J]. 软件学报, 2014(9): 1889-1908.
[3] 孟小峰, 慈祥. 大数据管理: 概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-169.
[4] 李星, 吕方, 刘颖, 等. 关于多核/众核系统可扩展性趋势的探讨[C]//2014全国高性能计算学术年会. 2014.
[5] 周旭. 面向多核/众核体系结构的确定性并行关键技术研究[D]. 长沙: 国防科学技术大学, 2013.
[6] HARRIS M, HARRIS M, PURCELL T, et al. GPGPU: General purpose computation on graphics hardware[C]// ACM SIGGRAPH 2004 Course Notes. ACM, 2004: 33.
[7] ISTVÁN Z, SIDLER D, ALONSO G. Caribou: Intelligent distributed storage[J]. Proceedings of the Vldb Endowment, 2017, 10(11): 1202-1213.
[8] SIDLER D, ISTVÁN Z, OWAIDA M, et al. Accelerating pattern matching queries in hybrid CPU-FPGA architectures[C]// ACM International Conference. ACM, 2017: 403-415.
[9] KARA K, GICEVA J, ALONSO G. FPGA-based data partitioning[C]// ACM International Conference on Management of Data. ACM, 2017: 433-445.
[10] 中国计算机学会. CCF 2016-2017中国计算机科学技术发展报告[M]. 北京: 机械工业出版社, 2017.

[11] 张铁赢, 黄贵, 章颖强, 等. X-DB: 软硬一体的新型数据库系统[J]. 计算机研究与发展, 2018, 55(2): 319-326.
[12] FIRESTONE D, PUTNAM A, MUNDKUR S, et al. Azure accelerated networking: SmartNICs in the public cloud[C]// 15th USENIX Symposium on Networked Systems Design and Implementation. USENIX Association, 2018.
[13] HONDA M, LETTIERI G, EGGERT L, et al. PASTE: A network programming interface for non-volatile main memory[C]//15th USENIX Symposium on Networked Systems Design and Implementation. USENIX Association, 2018.
[14] JIN X, LI X, ZHANG H, et al. NetCache: Balancing key-value stores with fast in-network caching[C]// Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017: 121-136.
[15] LI B, RUAN Z, XIAO W, et al. KV-Direct: High-performance in-memory key-value store with programmable NIC[C]// Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017: 137-152.
[16] AVNI H, 王鹏. 面向数据库的持久化事务内存[J]. 计算机研究与发展, 2018, 55(2): 305-318.
[17] 王健. 存储新篇章, 详解英特尔傲腾内存[J]. 电脑爱好者, 2017(11): 88-92.
[18] PETER S, LI J, ZHANG I, et al. Arrakis: The operating system is the control plane[J]. ACM Transactions on Computer Systems, 2015, 33(4): 1-30.
[19] Fio. [EB/OL]. [2018-07-02]. http://git.kernel.dk/?p=fio.git;a=summary.
[20] Block I/O Layer Tracing (blktrace). [EB/OL]. [2018-07-02]. https://git.kernel.org/pub/scm/linux/kernel/git/axboe/ blktrace.git.
[21] Netronome Agilio SmartNICs. [EB/OL]. [2018-07-02]. https://www.netronome.com/products/smartnic/overview.
[22] Barefoot Tofino. [EB/OL]. [2018-07-02]. https://www.barefootnetworks.com/products/brief-tofino/.
[23] Cavium Xpliant Family. [EB/OL]. [2018-07-02]. https://www.cavium.com/xpliant-ethernet-switch-product-family.html.
[24] XIE Y. Modeling, architecture, and applications for emerging memory technologies[J]. IEEE Design & Test of Computers, 2011, 28(1): 44-51.
[25] XIE Y. Future memory and interconnect technologies[C]// Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013. IEEE, 2013: 964-969.
[26] BEZ R, PIROVANO A. Non-volatile memory technologies: Emerging concepts and new materials[J]. Materials Science in Semiconductor Processing, 2004, 7(4/6): 349-355.
[27] MEENA J S, SZE S M, CHAND U, et al. Overview of emerging nonvolatile memory technologies[J]. Nanoscale Research Letters, 2014, 9(1): 526.
[28] YU S, CHEN P Y. Emerging memory technologies: Recent trends and prospects[J]. IEEE Solid-State Circuits Magazine, 2016, 8(2): 43-56.
[29] Intel Optane Technology. [EB/OL]. [2018-07-02]. https://www.intel.cn/content/www/cn/zh/architecture-and-technology/intel-optane-technology.html.
[30] KASHYAP S, MIN C, KIM K, et al. A scalable ordering primitive for multicore machines[C]// The Thirteenth EuroSys Conference. 2018: 1-15.
[31] BHAT S S, EQBAL R, CLEMENTS A T, et al. Scaling a file system to many cores using an operation log[C]// Symposium on Operating Systems Principles. ACM, 2017: 69-86.
[32] DREBES A, POP A, HEYDEMANN K, et al. NUMA-aware scheduling and memory allocation for data-flow task-parallel applications[J]. ACM Sigplan Notices, 2016, 51(8): 1-2.
[33] LEIS V, BONCZ P, KEMPER A, et al. Morsel-driven parallelism: A NUMA-aware query evaluation framework for the many-core age[C]// Proceedings of SIGMOD'14. ACM, 2014: 743-754.
[34] NVIDIA NVLink. [EB/OL]. [2018-07-02]. https://www.nvidia.com/en-us/data-center/nvlink/.
[35] HE B, LU M, YANG K, et al. Relational query coprocessing on graphics processors[J]. ACM Transactions on Database Systems, 2009, 34(4): 1-39.
[36] BRESS S, SAAKE G. Why it is time for a HyPE: A hybrid query processing engine for efficient GPU coprocessing in DBMS[J]. Proceedings of the Vldb Endowment, 2013, 6(12): 1398-1403.
[37] MapD. [EB/OL]. [2018-07-02]. https://www.mapd.com/.
[38] SALMI M F. Processing Big Data in Main Memory and on GPU[D]. Columbus, USA: The Ohio State University, 2016.
[39] MILETIĆ V, KOVAČIĆ B, LENKOVIĆ K. PG-Strom: Application of parallel programming technology NVIDIA CUDA on PostgreSQL database management system[C]// Razvoj Poslovnih I Informatičkih Sustava Case. 2013.
[40] HE B, YU J X. High-throughput transaction executions on graphics processors[J]. Proceedings of the Vldb Endowment, 2011, 4(5): 314-325.
[41] SIDLER D, ISTVÁN Z, ALONSO G. Low-latency TCP/IP stack for data center applications[C]// International Conference on Field Programmable Logic and Applications. IEEE, 2016: 1-4.
[42] FRANCISCO P. IBM Puredata System for Analytics Architecture[R]. IBM Redbooks, 2014.

[43] Netezza. [EB/OL]. [2018-06-25]. https://www.ibm.com/analytics/netezza/.

[44] BELAY A, PREKAS G, KLIMOVIC A, et al. IX: A protected dataplane operating system for high throughput and low latency[C]// Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2014: 49-65.

[45] Data Plane Development Kit. [EB/OL]. [2018-07-02]. https://dpdk.org/.

[46] JEONG E Y, WOO S, JAMSHED M, et al. mTCP: A highly scalable user-level TCP stack for multicore systems[C]// Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2014.

[47] OUSTERHOUT J, GOPALAN A, GUPTA A, et al. The RAMCloud Storage System[J]. ACM Transactions on Computer Systems, 2015, 33(3): 7.

[48] RUMBLE S M, KEJRIWAL A, OUSTERHOUT J. Log-structured memory for DRAM-based storage[C]// Usenix Conference on File and Storage Technologies. USENIX Association, 2014: 1-16.

[49] LEE C, PARK S J, KEJRIWAL A, et al. Implementing linearizability at large scale and low latency[C]//Proceedings of SOSP'15. ACM, 2015: 71-86.

[50] KEJRIWAL A, GOPALAN A, GUPTA A, et al. SLIK: Scalable low-latency indexes for a key-value store[C]// Usenix Conference on Usenix Technical Conference. USENIX Association, 2016: 57-70.

[51] ONGARO D, RUMBLE S M, STUTSMAN R, et al. Fast crash recovery in RAMCloud[C]// ACM Symposium on Operating Systems Principles. ACM, 2011: 29-41.

[52] NARAYANAN D, HODSON O, CASTRO M. FaRM: Fast remote memory[C]// Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014: 401-414.

[53] LIM H, HAN D, ANDERSEN D G, et al. MICA: A holistic approach to fast in-memory key-value storage[C]// Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014: 429-444.

[54] PolarDB. [EB/OL]. [2018-07-02]. https://help.aliyun.com/product/58609.html.

[55] SHI J, YAO Y, CHEN R, et al. Fast and concurrent RDF queries with RDMA-based distributed graph exploration[C]// Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2016: 317-332.

[56] WEI X, SHI J, CHEN Y, et al. Fast in-memory transaction processing using RDMA and HTM[C]// Symposium on Operating Systems Principles. ACM, 2015: 87-104.

[57] COBURN J, CAULFIELD A M, AKEL A, et al. NV-Heaps: Making persistent objects fast and safe with next-generation, non-volatile memories[J]. ACM Sigplan Notices, 2011, 46(3): 105-118.

[58] VOLOS H, TACK A J, Swift M M. Mnemosyne: Lightweight persistent memory[J]. ACM SIGARCH Computer Architecture News, 2011, 39(1): 91-104.

[59] HWANG T, JUNG J, WON Y. Heapo: Heap-based persistent object store[J]. ACM Transactions on Storage (TOS), 2015, 11(1): 3.

[60] Persistent Memory Development Kit. [EB/OL]. [2018-07-02]. http://pmem.io/pmdk/.

[61] ANDREI M, LEMKE C, RADESTOCK G, et al. SAP HANA adoption of non-volatile memory[J]. Proceedings of the VLDB Endowment, 2017, 10(12): 1754-1765.

[62] OGLEARI M A, MILLER E L, ZHAO J. Steal but no force: Efficient hardware Undo+Redo Logging for persistent memory systems[C]// 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2018: 336-349.

[63] ARULRAJ J, PAVLO A, DULLOOR S R. Let's talk about storage & recovery methods for non-volatile memory database systems[C]//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 707-722.

[64] AGRAWAL R, JAGADISH H V. Recovery algorithms for database machines with non-volatile main memory[C]// International Workshop on Database Machines. Berlin: Springer, 1989: 269-285.

[65] GAO S, XU J, HE B, et al. PCMLogging: Reducing transaction logging overhead with PCM[C]// Proceedings of the 20th ACM International Conference on Information and Knowledge Management. ACM, 2011: 2401-2404.

[66] OUKID I, BOOSS D, LEHNER W, et al. SOFORT: A hybrid SCM-DRAM storage engine for fast data recovery[C]// Proceedings of the Tenth International Workshop on Data Management on New Hardware. ACM, 2014: 8.

[67] ARULRAJ J, PERRON M, PAVLO A. Write-behind logging[J]. Proceedings of the VLDB Endowment, 2016, 10(4): 337-348.

[68] ZHANG Y Y, YANG J, MEMARIPOUR A, et al. Mojim: A reliable and highly-available non-volatile memory system[J]. ACM SIGARCH Computer Architecture News, 2015, 43(1): 3-18.

[69] Storage Performance Development Kit. [EB/OL]. [2018-07-02]. http://www.spdk.io.

(下转第 78 页)

[7] LANG H, FUNKE F, BONCZ P A, et al. Data blocks: Hybrid OLTP and OLAP on compressed storage using both vectorization and compilation [C]// International Conference on Management of Data. New York: ACM, 2016: 311-326.
[8] RAMNARAYAN J, MOZAFARI B, WALE S, et al. SnappyData:A hybrid transactional analytical store built on spark [C]// International Conference on Management of Data. New York: ACM, 2016: 2153-2156.
[9] LEE J, HAN W S, NA H J, et al. Parallel replication across formats for scaling out mixed OLTP/OLAP workloads in main-memory databases [J]. The VLDB Journal, 2018, 27(3): 421-444.
[10] SQream. SQream SQream DB [DB/OL]. [2018-06-16]. https://sqream.com/.
[11] ROOT C, MOSTAK T. MapD: A GPU-powered big data analytics and visualization platform [C]// Proceeding of the SIGGRAPH '16 ACM SIGGRAPH 2016 Talks. New York: ACM, 2016: Article No 73. DOI:10.1145/2897839.2927468.
[12] 阳振坤. OceanBase 关系数据库架构 [J]. 华东师范大学学报 (自然科学版), 2014(5): 141-148, 163.
[13] 黄贵, 庄明强. OceanBase 分布式存储引擎 [J]. 华东师范大学学报 (自然科学版), 2014(5): 164-172.
[14] GOOGLE. Google Snappy [DB/OL]. [2018-06-16]. https://github.com/google/snappy.
[15] SALOMON D. Data Compression: The Complete Reference [M]. New York: Springer-Verlag Inc, 2000.
[16] APACHE. Apache parquet [DB/OL]. [2018-06-16]. http://parquet.apache.org/.

(责任编辑: 李 艺)

(上接第 40 页)

[70] KIM H J, LEE Y S, KIM J S. NVMeDirect: A User-space I/O framework for application-specific optimization on NVMe SSDs[C]// Proceedings of HotStorage. 2016.
[71] BJØRLING M, GONZÁLEZ J, BONNET P. LightNVM: The linux open-channel SSD subsystem[C]// Proceedings of FAST'17. USENIX Association. 2017: 359-374.
[72] CAULFIELD A M, MOLLOV T I, EISNER L A, et al. Providing safe, user space access to fast, solid state disks[J]. ACM SIGARCH Computer Architecture News, 2012, 40(1): 387-400.
[73] YANG J, MINTURN D B, HADY F. When poll is better than interrupt[C]// Proceedings of FAST'12. USENIX Association. 2012.
[74] LI C, DING C, SHEN K. Quantifying the cost of context switch[C]// Proceedings of the 2007 Workshop on Experimental Computer Science. ACM, 2007: 2.
[75] SHIN W, CHEN Q, OH M, et al. OS I/O path optimizations for flash solid-state drives[C]// USENIX Annual Technical Conference. 2014: 483-488.
[76] YU Y J, SHIN D I, SHIN W, et al. Optimizing the block I/O subsystem for fast storage devices[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 6.
[77] XU J, SWANSON S. NOVA: A log-structured file system for hybrid volatile/non-volatile main memories [C]// Proceedings of FAST'16. USENIX Association. 2016: 323-338.
[78] VOLOS H, NALLI S, PANNEERSELVAM S, et al. Aerie: Flexible file-system interfaces to storage-class memory[C]//Proceedings of the Ninth European Conference on Computer Systems. ACM, 2014: 14.
[79] KANNAN S, ARPACI-DUSSEAU A C, ARPACI-DUSSEAU R H, et al. Designing a true direct-access file system with DevFS[C]//Proceedings of the 16th USENIX Conference on File and Storage Technologies. 2018: 241-256.

(责任编辑: 林 磊)