在网缓存综述

孙纪涛[1)]

[1)](华中科技大学 计算机科学与技术学院, 武汉 430074)

摘 要 近年来，现代互联网服务如搜索、社交网络和电子商务对高性能的键值存储需求不断增加。这些服务中，每个网页的渲染往往需要大量的存储访问，因此系统运营商越来越依赖内存中的键值存储来满足庞大用户群体的吞吐量和延迟需求。然而，在面对数十亿用户的扩展时，动态且倾斜的工作负载成为扩展键值存储的主要挑战。本文首先分析了当前存储领域面临的几大问题，为了解决这一问题提出了在网缓存系统，文章简要介绍了可编程交换机的内容，详细地介绍了可编程交换机在缓存系统在读密集型工作负载、写密集型工作负载以及分布式共享内存系统的研究情况，最后分析给出在网缓存未来潜在的研究方向。

关键词 在网缓存；缓存一致性；分布式共享内存；可编程交换机

A Survey for In-network Caching

Jitao Sun[1)]

[1)](School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract In recent years, modern internet services such as search, social networks, and e-commerce have seen an increasing demand for high-performance key value storage. In these services, the rendering of each webpage often requires a large amount of storage access, so system operators are increasingly relying on in memory key value storage to meet the throughput and latency needs of a large user group. However, when facing the expansion of billions of users, dynamic and skewed workloads become the main challenge in expanding key value storage. This article first analyzes several major problems facing the current storage field, and proposes an in-network caching system to solve this problem. The article briefly introduces the content of programmable switches and provides a detailed introduction to the research status of programmable switches in caching systems under read intensive workloads, write dense workloads, and distributed shared memory systems. Finally, the article analyzes and provides potential research directions for on network caching in the future.

Key words In-network Caching; Cache consistency; Distributed shared memory; Programmable Switches

1 引言

现代互联网服务，如搜索、社交网络和电子商务，对高性能的键值存储有着重要依赖。即使是渲染单个网页也经常需要数百甚至数千次的存储访问。因此，随着这些服务扩展到数十亿用户，系统运营商越来越依赖内存中的键值存储来满足必要的吞吐量和延迟需求。在扩展键值存储时，一个主要挑战是应对倾斜的动态工作负载。热门项目比其他项目收到的查询要多得多，而且由于热门帖子、限时优惠和趋势事件，“热门项目”的集合会迅速变化。例如，在 Facebook 的 Memcached 部署中，

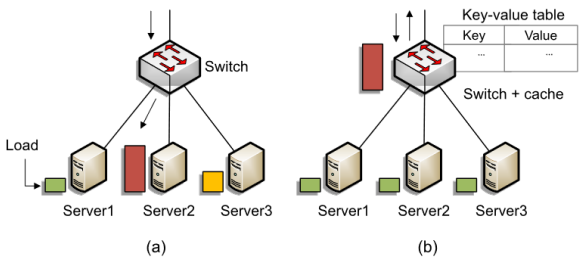

图 1 (a) 传统的基于交换机的缓存 (b) 基于可编程交换机的缓存

最终修改稿日期：2024年1月3日；孙纪涛，男，2002年生，硕士，计算机学会(CCF)会员（Q0950G）,主要研究领域为分布式系统、在网计算. E-mail: jitsun@hust.edu.cn

10%的项目占了 60-90%的查询量。这种倾斜可能导致严重的负载不平衡，从而导致显著的性能下降：服务器要么被过度利用，要么被利用不足，吞吐量降低，响应时间受到长尾延迟的影响。负载不平衡的问题对于高性能、内存中的键值存储尤为严重。图 1 显示了一个典型的倾斜键值存储系统，其中存储键值对象的服务器之间存在负载不平衡。这种系统的性能可能表现为降低的吞吐量和长延迟。例如，由于存储了热点项目并且被过度利用，服务器 2 可能会出现显著的延迟，而服务器 1 则未充分利用。传统的基于闪存和磁盘的键值存储可以使用快速内存缓存层（如 SwitchKV）来实现平衡，但服务器端的缓存对内存存储来说并不适用，因为缓存层和存储层之间的性能差异很小。作为替代方案，可以使用选择性复制，即将热门项目复制到其他存储节点。然而，选择性复制除了消耗更多硬件资源外，还需要复杂的数据移动、数据一致性和查询路由机制，这使系统设计变得复杂，并引入了额外开销。

最近，受软件定义网络（SDN）和数据中心网络快速变化的要求驱动，新型可编程网络设备类别的出现，例如可编程交换机和网络加速器。可编程交换机允许特定于应用程序的报头解析和定制匹配-动作规则，提供具有轻量级可编程转发平面的太比特数据包交换。网络加速器配备了可扩展的低功耗多核处理器和快速流量管理器，支持更大规模的数据平面计算以达到线速。它们共同提供了在网络中传输时进行数据包处理的能力，可以用于应用级别的计算。这些技术的提出，为在网缓存提供了巨大的应用前景。

2 问题与挑战

in-network computation (INC) 的关键思想是将一组计算操作从端服务器转移到可编程网络设备，以便实时以低延迟提供远程操作，减少网络流量，并使服务器进入低功耗模式，甚至关闭或移除服务器，从而实现节能和降低成本。

然而，将计算任务卸载到网络中存在三个挑战。

（1）尽管随着可编程网络基础设施相关的硬件资源随着时间的推移不断改进，但支持通用数据中心计算或服务（如键值存储）的计算能力有限，存储空间也很少。例如，Arista 7150S 交换机中的 Intel FlexPipe 芯片具有灵活的解析器和可定制的匹配-操作引擎，用于进行转发决策和控制流状态转换。交换机被编程为一组规则，然后在数据流经过交换机时对数据包头部进行数据驱动的修改。该交换机上的 9.5 MB 数据包缓冲内存并不用于存储非数据包数据；即使使用了，大部分仍然需要用于缓冲来自数十个端口的传入流量，以防拥塞，只留下有限的空间供其他用途。

（2）数据中心网络在端点之间提供了许多路径，网络组件故障很常见，因此在网络元素之间保持计算和状态的一致性是复杂的。

（3）需要一种简单而通用的计算抽象来支持广泛类别的数据中心应用，并可以轻松集成到应用逻辑中。

当前面临的网络缓存有很多问题和挑战，尤其是在分布式共享内存（DSM）系统的背景下。以下是总结得到的一些关键问题和挑战[5]：

1) 缓存一致性的复杂性和开销： 分布式缓存固有地涉及缓存一致性，这是复杂的并且会产生显著的开销。这种复杂性来自于为协调而进行的过多通信，如跟踪缓存副本的分布、使缓存失效以及串行化冲突请求。即使在一些系统中，写入比例略微增加也可能严重影响整体吞吐量。
2) 可编程交换机的局限性： 可编程交换机虽然提供了减少服务器之间协调和快速处理数据包等优点，但也存在一些限制。其中包括由于其受限的编程模型、较小的芯片上内存（通常为 10-20 MB）和处理故障状态的挑战，例如数据包丢失可能导致死锁等潜在错误状态。
3) 在快速网络环境中的性能影响： 缓存一致性可能会极大地限制系统性能，即使在快速网络的情况下也是如此。当在网络上共享更多数据时，这会导致明显的吞吐量降低和网络数据包增加。
4) 所有权迁移和数据包丢失处理： 为了解决交换机内存容量有限的问题，采用了所有权迁移和幂等操作等技术。所有权迁移基于缓存块的一致性流量在主代理和交换机之间动态转移，有助于有效地管理内存。幂等操作用于处理数据包丢失，确保操作是可重复的而不会产生不良影响。
5) 可扩展性和崩溃安全性的担忧： 可扩展性是一个关键问题，特别是在扩展到超过单个机柜规

模的系统时。解决缓存失效风暴以及在交换机、内存节点和阴影节点等组件之间保持崩溃安全性是重大挑战。确保在组件故障情况下的数据一致性和恢复需要复杂的机制，这可能使系统设计和芯片上内存使用变得复杂。

这些挑战凸显了在 DSM 系统中实现高效的网络缓存的复杂性。它们要求在充分发挥可编程交换机的能力的同时，平衡处理缓存一致性和系统可扩展性的固有限制和复杂性。

3 可编程交换机

现代商品交换机具有数十兆字节的芯片内存。由于许多内存键值存储目标是小值（例如，在 Facebook 的 Memcached 部署中，76%的读取查询的值小于 100 字节），芯片内存足够大，可以存储 O(NlogN)个项目进行负载平衡，同时仍然可以为传统网络功能留出足够的交换机资源。对于无法适应一个数据包的大项目，可以将项目分成较小的片段，并使用多个数据包检索它们。

传统交换机 ASIC 是固定功能的，因此添加新功能需要设计和制造新的 ASIC，这会产生巨大的资本和工程成本。然而，像 Barefoot Tofino 和 Cavium XPliant 这样的新一代可编程交换机 ASIC 使得网络内缓存成为可能并且可以立即部署。它们允许用户对交换机数据包处理流水线进行编程，而无需更换交换机 ASIC。具体来说，我们可以

1) 对交换机解析器进行编程，以识别自定义数据包格式（例如，包含用于键和值的自定义字段），

2) 对芯片内存进行编程，以存储自定义状态（例如，存储热门项目和查询统计信息），以及

3) 对交换机进行编程，以执行自定义操作（例如，将值从芯片内存复制到数据包中并检测重点访问者）。

3.1 PISA架构

PISA 是一个数据包处理模型，包括以下元素：可编程解析器、可编程匹配-操作流水线和可编程逆解析器，见图 2。可编程解析器允许程序员定义头部（根据自定义或标准协议）并对其进行解析。解析器可以表示为状态机。可编程匹配-操作流水线执行数据包头部和中间结果上的操作。单个匹配-操作阶段具有多个存储块（表、寄存器）和算术逻辑单元（ALU），允许同时进行查找和操作。由于一些操作结果可能需要用于进一步处理（例如，数据依

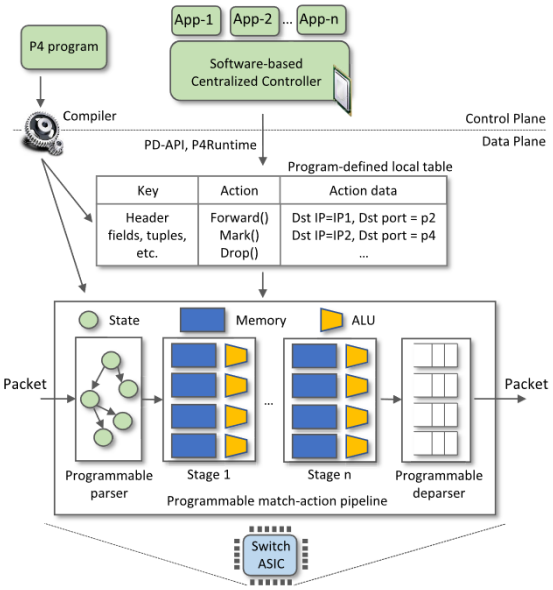

图 2 基于 PISA 的数据平面及其与控制平面的交互

赖关系），阶段被按顺序排列。可编程逆解析器将数据包头部重新组装并对其进行串行化以进行传输。PISA 设备与协议无关。

在图 2 中，P4 程序定义了用于查找操作的键的格式。键可以使用数据包头部信息来形成。控制平面使用键填充表项，并提供操作数据。键用于匹配数据包信息（例如，目标 IP 地址），而操作数据用于执行操作（例如，输出端口）。

3.2 可编程交换机的特征

表 1 P4 可编程交换机与固定功能交换机的对比

特征	可编程交换机	固定功能交换机
吞吐量	6.4Tb/s	6.4Tb/s
支持 100G 端口数	64	64
最大转发速率	4.8B pps	4.2 B pps
最大支持 25G/10G 端口数	256/258	128/130
可编程性	支持（P4）	不支持
功耗	每端口 4.2W	每端口 4.9W
大规模 NAT	支持（100k）	不支持
大规模有状态的访问控制列表	支持（100k）	不支持
大规模隧道	支持（192k）	不支持
数据包缓冲区	统一	分段
LAG/ECMP	灵活可编程性	哈希种子
ECMP	256 个路径	128 个路径
遥感	线速率统计	采样

延迟	400ns 内	450ns 内

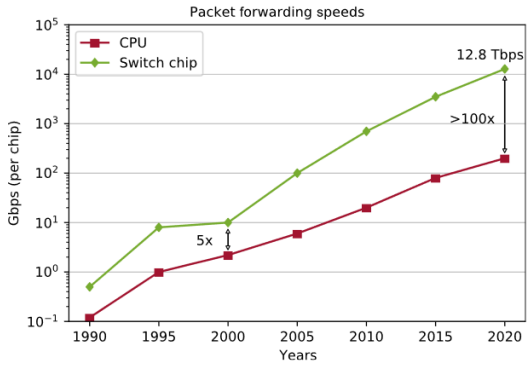

图 3 通用 CPU 和交换机芯片的数据包转发速度的演变

可编程交换机的主要特征包括：

1) 灵活性：程序员可以在显著较短的时间内设计、测试和采用新的协议和功能。
2) 自上而下的设计：几十年来，网络行业一直采用自下而上的方法运作。固定功能的 ASIC 位于底层，并向顶部的程序员强制执行可用的协议和功能。而有了可编程交换机，程序员自定义 ASIC 中的协议和功能。
3) 可见性：可编程交换机提供对网络行为更大的可见性。INT 是一个从数据平面收集和检索信息的框架的示例，无需控制平面的干预。
4) 降低了复杂性：固定功能的交换机集成了大量协议的超集。这些协议消耗资源，并增加了处理逻辑的复杂性，这是在硅芯片中硬编码的。而可编程交换机使程序员有选择地实施仅需的协议。
5) 差异化：程序员实施的定制协议或功能无需与芯片制造商共享。
6) 性能增强：可编程交换机不引入性能损失。相反，它们可能比固定功能的交换机产生更好的性能。表 1 显示了可编程交换机和固定功能交换机的比较。与通用 CPU 相比，ASIC 在交换速度上仍更快，并且如图 3 所示，这一差距正在不断扩大。

交换机性能的提升依赖于多维度的并行性，如下所述：

1) 不同阶段的并行性：管道的每个阶段一次处理一个数据包。在图 2 中，阶段的数量为 n。在入口和出口的管道上，实现可能具有超过 10 个阶段。虽然增加阶段可以提高并行性，但它们会在芯片上消耗更多的面积，增加功耗和延迟。
2) 阶段内的并行性：ASIC 每个阶段包含多个匹配-动作单元。在匹配阶段，表可用于并行查找。在图 2 中，每个阶段有四个匹配（蓝色部分）可以同时发生。一个 ALU 在头部字段上执行一个操作，实现对所有字段的并行操作。每个阶段存在数百个匹配-动作单元，整个管道中则有数千个。由于 ALU 执行简单的操作并使用简单的精简指令集计算机（RISC）类型指令集，因此它们可以以最小成本在硅中实现。
3) 非常长指令字：在给定时钟周期内发出的指令集可以被视为具有多个操作的一条大指令，称为非常长指令字（VLIW）。 VLIW 是由匹配表的输出形成的。每个阶段在一个数据包上执行一个 VLIW，阶段内的每个动作单元执行一个操作。因此，对于给定的数据包，每个字段每个阶段应用一个操作。
4) 管道上的并行性：交换芯片可能包含多个管道，也称为流水线，每个芯片一个。 PISA 设备上的管道类似于通用 CPU 上的核心。例如，包含两个和四个管道的芯片。每个管道与其他管道隔离，独立处理数据包。管道可以实现相同的功能或不同的功能。

3.3 P4语言

P4（Programming Protocol-independent Packet Processors）是一种用于定义网络设备数据包处理逻辑的编程语言。它旨在提供对网络设备（如交换机和路由器）数据包处理过程的更大控制权和可编程性。以下是 P4 语言的一些关键特点和方面的详细介绍：

1) 协议无关性（Protocol Independence）：P4 语言的一个主要目标是使网络设备对协议无关，即不受特定网络协议的限制。程序员可以自定义协议的解析、处理和生成，使设备能够适应不同的通信标准。
2) 可编程性（Programmability）：P4 允许程序员在网络设备中定义数据包处理逻辑，包括解析数据包头部、执行各种操作以及生成新的数据包头部。这使得网络设备可以根据特定的应用需求进行灵活配置和适应性调整。
3) 可重配置性（Reconfigurability）：P4 支持在实际运行中对设备的解析器和处理逻辑进行重新定义。这使得网络设备能够适应网络的变化，例如支持新的通信协议或适应新的应用场景。
4) 目标独立性（Target Independence）：P4 语言使

程序员能够编写与底层硬件无关的代码。编译器负责将 P4 程序转换为特定目标硬件（如 ASIC、FPGA、NIC 等）的二进制代码。

5) 层次化结构（Hierarchical Structure）：P4 程序通常由解析器（Parser）、处理器（Control）、计算单元（Compute）、逆解析器（Deparser）等模块组成，形成层次化结构。这些模块协同工作以完成对数据包的处理。

6) 多维度的并行性（Multi-dimensional Parallelism）：P4 支持在不同的阶段和模块中实现并行处理，以提高网络设备的性能。

7) P4_16 版本：P4_16 是 P4 语言的一个较新版本，于 2016 年发布。它扩展了 P4_14 的能力，支持更广泛的底层目标，包括 ASIC、FPGA 和 NIC 等。这使得 P4_16 更加适用于各种网络设备和场景。

4 研究现状分析

在内存中的键值存储的网络内缓存领域，内存缓存被证明对于基于闪存和磁盘的键值存储系统是一种有效的策略。这是因为 DRAM 的速度相对于 SSD 和 HDD 而言快了数个数量级。然而，随着键值存储系统本身被迁移到主内存，内存缓存逐渐失去其性能优势，变得不再切实可行。考虑一个拥有 128 台服务器的内存中键值存储系统：如果每台服务器提供 1000 万次每秒的查询（QPS），整个存储系统可以达到 1.28 亿 QPS 的吞吐量，而缓存层需要提供相应的吞吐量以保持平衡。将缓存层融入网络中成为平衡内存中键值存储系统的自然解决方案。目前，对在网缓存的研究主要集中在读密集型在网缓存、写密集型在网缓存以及分布式共享内存在网缓存，下面我们将依次介绍各个领域的研究现状。

4.1 读密集型在网缓存

在高性能内存中键值存储系统中，负载不平衡的问题尤为严峻。传统基于闪存和磁盘的键值存储系统可以通过快速的内存缓存层（如 SwitchKV [1]）实现平衡，然而对于内存中的存储系统，基于服务器的缓存并不能很好地解决这一问题，因为缓存层与存储层之间的性能差异微乎其微（见图 4）。作为替代方案，可以使用选择性复制，即将热点数据复制到额外的存储节点。然而，除了消耗更多的硬件资源外，选择性复制还需要复杂的数据移动、数据一致性和查询路由机制，这使得系统设计变得复杂并引入了额外的开销。

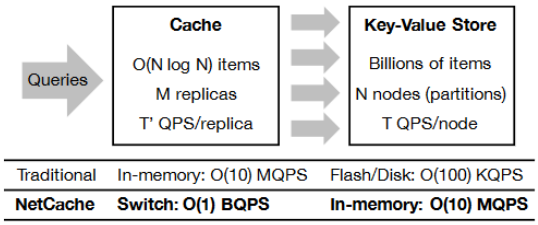

图 4 为了提供有效的负载平衡，缓存节点只需要缓存 O(NlogN) 项，但需要比存储节点快几个数量级 (T '>> T)

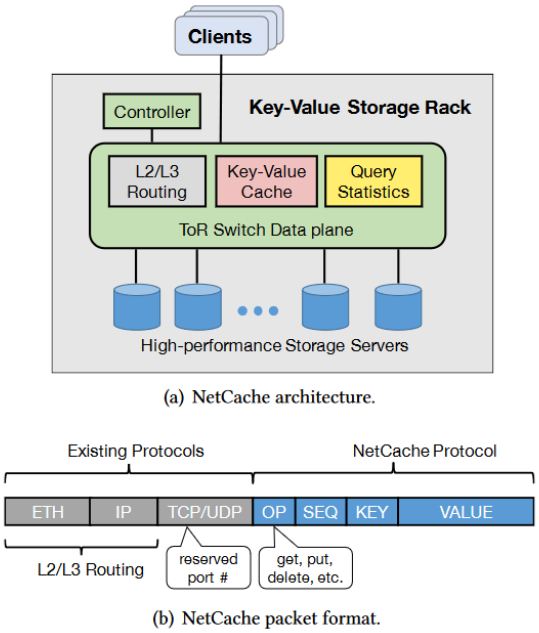

图 5 NetCache 概览

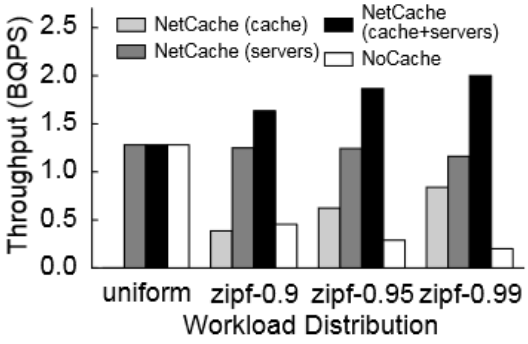

图 6 NetCache 中几个系统的吞吐量随倾斜度的变化情况

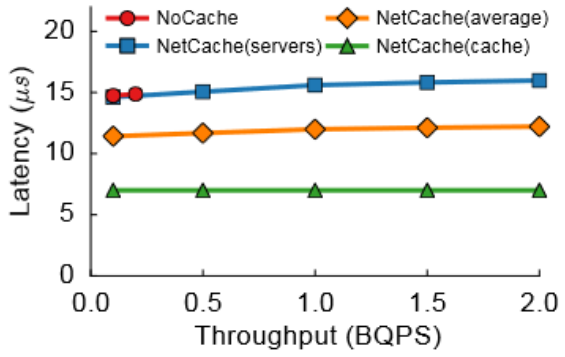

图 7 NetCache 中几个系统的延迟随吞吐率的变化情况

为解决上述问题，论文[2]介绍了 NetCache，它是一种新一代可编程交换机与高效缓存数据在网络中的键值存储架构（见图 5），充分利用了可编程交换机的强大性能和灵活性。传统缓存与存储服务器相比，交换机在数据输入输出方面经过了优化，性能提升了数个数量级，因此它们是构建高性能内存键值存储的理想位置。传统缓存通常需要高的缓存命中率（>90%）来处理大部分查询，而 NetCache 使用交换机作为负载均衡缓存，其缓存命中率较中等（<50%）。负载均衡缓存可以为热门数据提供服务，并使服务器上的其余负载更加均匀。尽管交换机的芯片内存有限，但 NetCache 利用了理论结果，即缓存 O(NlogN) 个项目足以平衡整个散列分区关键-值存储集群中的 N 存储服务器（或 CPU 核心）的负载，而不管系统中存储的项目数量。总体而言，尽管工作负载不均匀，整个系统能够保证高的聚合吞吐量和低的延迟。NetCache 的核心是一个数据包处理管线，用于检测、索引、存储和提供键值项。我们使用匹配-操作表来对传输包头中携带的键进行分类，并使用可编程交换机中的芯片内存实现的寄存器数组来存储值。它利用现代交换机 ASIC 的多流水线、多级结构，高效地索引和打包可变长度的值，以适应有限的交换机表和内存资源。

图 6,7 展示了，对于高性能内存中键值存储，NetCache 将吞吐量提高了 3-10 倍，并将高达 40% 的查询延迟降低了 50%。这篇论文也第一次证明了，复杂的应用程序级功能（例如网络内缓存）可以在可编程交换机上以线速率运行。

IncBricks [4]支持使用可编程的网络中间盒在网络中进行缓存。这些网络内缓存或键值存储系统需要保持网络硬件和服务器中的数据一致。它们采用类似的基于服务器的机制来解决这个问题。例如，在 IncBricks 中，服务器记录共享者列表，并在客户端发送 SET 或 DELETE 请求时向网络加速器发出失效命令。它与 NetCache 有所不同，IncBricks 还在底层使用了硬件加速器，而 NetCache 则是纯软件实现。

4.2 写密集型在网缓存

键值存储系统已广泛部署在现代数据中心，用于管理结构化数据（以记录为单位）以支持数据密

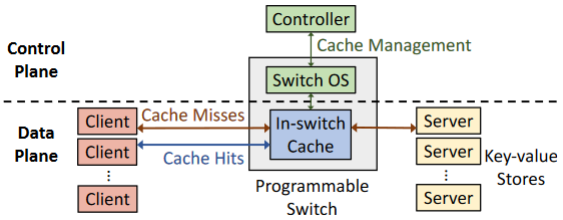

图 8 FarReach 架构

集型应用，例如社交网络、网络索引和电子商务。实际的键值存储工作负载传统上以读为主（例如，Facebook 的 Memcached 中读写比可达 30:1）。近期对生产键-值存储的实地研究显示写密集型工作负载的主导地位；例如，Twitter 的 Twemcache 集群中有超过 20%的写入次数多于读取次数，而 Facebook 的 RocksDB 生产中的人工智能/机器学习服务有 92.5%的读修改写操作。此外，写密集型工作负载是倾斜的；例如，在 Twitter 的 Twemcache 集群中，25%的频繁访问（即热点）记录占据了写入工作负载的主导地位。在数据中心为键值存储实现高写入性能是一项具有挑战性的任务。从客户端到键值存储服务器发出的写请求往往因交换机到服务器的传输和服务器端处理而遭受长往返时间（RTT）。特别是，如果服务器过载，I/O 请求将具有长排队延迟甚至可能被丢弃。此外，在跨足多个服务器的分布式键值存储中，由于倾斜工作负载下对热点记录的大量请求，部分服务器可能成为瓶颈，从而导致负载不平衡。

然而，现有的基于交换机的缓存方法主要针对读密集型工作负载，并采用直写缓存策略。因此，在写密集型工作负载下，与无缓存相比，它们并未提供写入性能的提升。为了解决这个问题，论文[3]提出了 FarReach，一个快速、可用和可靠的基于交换机的写回缓存框架，旨在改善在写密集型工作负载下键值存储的 I/O 性能。如图 8，FarReach 通过精心设计控制平面和数据平面，实现了将缓存管理

卸载到控制平面中的集中式控制器，同时通过轻量级的控制平面管理实现了高性能的数据平面。它包括以下设计特性：(i) 快速的缓存入场，将热点记录引入到交换机缓存中，而不阻塞数据平面的 I/O 流量；(ii) 可用的缓存驱逐，确保从交换机缓存中驱逐的最新记录仍然可供读取请求使用；以及(iii) 可靠的快照生成和零丢失的崩溃恢复，用于在交换机故障期间防范数据丢失。

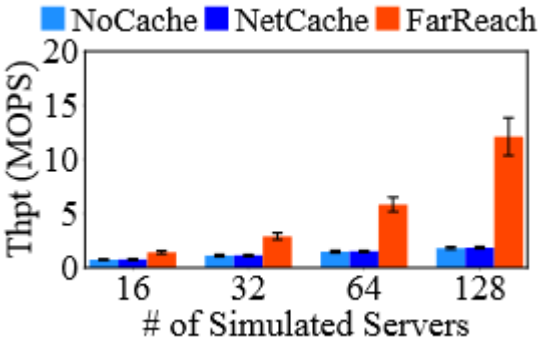

图 9 FarReach 吞吐量性能测试

回写缓存策略虽然能够带来写密集型工作负载下的性能提升，但也为缓存管理带来了新的挑战。论文[3]列出了以下三个挑战：

1) 性能挑战。由于可编程交换机具有受限的流水线编程模型和有限的硬件资源，因此有必要将交换机级别的缓存管理卸载到一个集中式控制器，而交换机仅在写回策略下在数据平面中更新缓存记录。然而，由于控制器到交换机的高延迟，在可编程交换机中，控制平面处理速度远远慢于数据平面处理，从而成为 I/O 性能的瓶颈。
2) 可用性挑战。在写回缓存下，缓存准入和驱逐算法都需要在控制平面和数据平面之间进行仔细协调，以便正确地维护在交换机缓存或服务器存储中的最新记录；否则，过时的记录可能会返回给客户端。在直写缓存中不存在这样的问题，因为它总是在服务器端保持最新的记录。在可编程交换机中，可用性问题更具挑战性，因为控制器需要同时管理缓存和服务器更新，但会产生很高的开销。此外，控制器不在数据平面的数据包转发路径上，无法了解数据平面中传输的数据包的情况。
3) 可靠性挑战。在回写缓存中，最新记录可能仅保留在交换机缓存中，并且可能会延迟更新到服务器端存储。如果交换机崩溃，所有最新的记录将丢失。在直写缓存中，再次不存在这样的问题，因为最新的记录可以持久地保留在服务器端存储中。

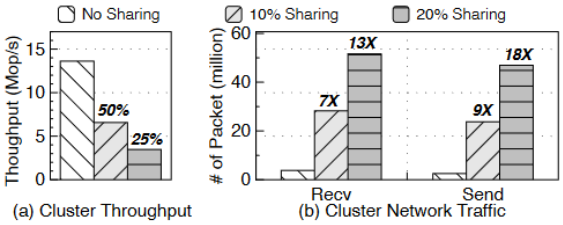

图 10 缓存一致性带来的影响

FarReach 精心设计了三个方案来解决上述三个挑战：(i) 针对性能挑战，FarReach 支持非阻塞的缓存入场，以便将热点记录引入到交换机缓存中，从而实现高写入性能。它还确保在可编程交换机的多流水线设置下的缓存入场中的原子性。(ii) 针对可用性挑战，FarReach 确保从交换机缓存中驱逐的任何最新记录仍然对客户端可用。(iii) 针对可靠性挑战，FarReach 在交换机故障期间提供数据丢失保护。它使用一种崩溃一致性的快照生成算法来制作交换机缓存状态的快照。它还确保在多流水线设置下生成快照的原子性。它进一步将快照生成与上游备份耦合，以实现零丢失的崩溃恢复。

Tofino 交换机测试台上的实验表明（见图 9），在倾斜的写入密集型工作负载下，FarReach 的吞吐量比最先进的交换机内缓存方法提高了 6.6 倍。

在 AI 技术发展生机勃勃的时代，FarReach 为写密集型工作负载下在网缓存的性能研究带来了新的启发。

4.2 分布式共享内存在网缓存

分布式共享内存（DSM）在早期（大约上世纪 90 年代初）曾因提供统一的全局内存抽象而短暂流行。然而，由于在低速网络上性能不佳，它后来未能吸引更多的关注。随着高性能网络技术的不断进步（例如 RDMA），网络带宽现在可以迅速增至 100Gbps 甚至 200Gbps，而延迟降至不到 2 微秒，这促使人们重新关注 DSM。研究人员抓住这个机会，系统地重新思考 DSM 的设计，并在学术界和工业界取得了一系列的成功。例如，微软开发了基于 RDMA 的快速 DSM 系统 FaRM；在 FaRM 的基础上，工程师们构建了键值存储系统、分布式事务引擎，以及一个名为 A1 的图数据库。

然而，分布式缓存总是伴随着一致性问题，不

得不考虑其复杂性和开销。具体而言，一致性引入了大量的协调通信，例如跟踪缓存副本的分布、使缓存无效以及对冲突请求进行串行化，从而严重影响了整体吞吐量。例如，即使在基于 RDMA 的 ccNUMA 中，通过将写入比例从 0 增加到 5%，引入轻微的缓存一致性也可能使性能降低 50%。

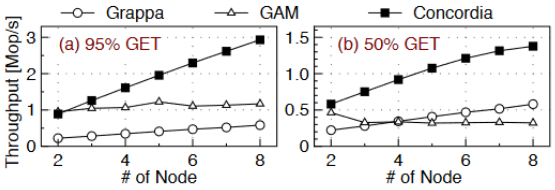

图 11 (a) 读密集型工作负载吞吐量 (b) 写密集型工作负载吞吐量

为了了解缓存一致性的性能影响，我们使用微基准测试评估 GAM [6]，这是一种由基于 RDMA 的先进 DSM 支持的基于目录的协议。在这个基准测试中，8 个节点的集群中的每个节点启动四个线程，对全局内存进行 8 字节的写入/读取操作，写入比例为 50%。在这里，我们将共享比定义为访问共享数据的操作百分比。通过将比例从 0（即无共享）变化到 20%，我们可以看到在图 10(a)中，吞吐量下降了 75%。此外，我们收集了所有节点接收和发送的数据包（图 10(b)）。随着更多数据的共享，由于缓存一致性协议中昂贵的分布式通信，网络上的数据包数量呈激增趋势（最多增加到 18 倍）。从基准测试中，我们得出结论即使在快速网络下，现有的缓存一致性协议也会严重限制系统性能。

为了解决上述问题，论文 [2] 介绍了 CONCORDIA，一种具有网络内缓存一致性的高性能机架级分布式共享内存管理系统。CONCORDIA 的核心是 FLOWCC（in-Flow Cache Coherence），这是一种由交换机和服务器共同支持的缓存一致性协议。在 FLOWCC 中，交换机作为数据平面，串行化冲突的一致性请求并将它们组播到目的地，从而减少了服务器之间的协调。另一方面，服务器充当控制平面，执行状态转换并向交换机发送相应的更新。CONCORDIA 利用芯片上的内存存储缓存块的元数据，并实现了读写锁以进行并发控制。CONCORDIA 的关键思想是：

1) 数据平面和控制平面的分离。在 CONCORDIA 中，交换机只做它擅长的事情，即作为数据平面路由数据包；它通过网络锁组播缓存一致性请求并串行化冲突请求（锁定可以被看作是控制冲突请求的路由路径）。相比之下，作为控制平面的服务器执行状态转换并向交换机发送相应的更新。这种分离简化了交换机的数据平面设计，以克服其受限的表达能力。

2) 统一基于交换机和服务器的一致性处理。由于交换机的芯片上内存容量有限，CONCORDIA 让交换机仅管理热数据的一致性，并将冷数据的处理交给服务器。根据缓存块引起的一致性流量，CONCORDIA 在服务器和交换机之间动态迁移所有权。

3) 最小化对交换机状态的修改次数。对于每个修改交换机状态的操作，即存储在交换机寄存器数组中的值，还必须处理相应的数据包丢失情况，这会消耗交换机的宝贵硬件资源并使系统设计变得更加复杂。

图 11 的评估表明，对于读密集型工作负载，当节点超过 2 个时，CONCORDIA 的性能优于 Grappa 3.9 倍到 5 倍，GAM 的性能优于 GAM 1.2 倍到 2.5 倍。对于写入密集型工作负载，CONCORDIA 的性能优于 GAM 1.2 至 4.2 倍，优于 Grappa 2.3 倍至 2.6 倍。

分布式存储系统的任务是在负载巨大且不可预测的情况下提供快速、可预测的性能。CONCORDIA 为在网缓存在分布式共享内存方面的研究开辟了新的思路。

5 未来研究方向

目前随着研究人员对在网缓存研究得越来越深入，更多的可能性被逐渐挖掘出来。以下是在网缓存方向未来潜在的研究方向：

1) 高级缓存管理技术：针对在网缓存系统中，探索如何处理多样化和动态工作负载。这包括开发智能算法来预测数据项的重要性，从而优化缓存命中率和降低延迟。例如，使用机器学习技术来预测哪些数据最有可能被请求，以及何时清除缓存中的旧数据。
2) 可扩展性研究：在大型和复杂网络环境中评估和增强在网缓存和计算框架的可扩展性。这可能涉及到新的架构设计，以支持更多的数据流量和更复杂的计算任务，同时保持高效的性能和低延迟。
3) 与新兴数据中心技术的集成优化：研究如何将在网缓存技术与软件定义网络、云计算以及其

他新兴技术结合起来。目标是优化数据流，减少延迟，并提高整体网络效率。

4) 更广泛应用领域的缓存策略探索: 扩展在网缓存技术在不同应用领域的应用，如机器学习、视频流处理和物联网。这需要开发适用于这些领域特定需求的缓存策略和算法。
5) 增强的容错和恢复机制: 开发更强大的容错和恢复策略，以确保在网络中断或硬件故障的情况下，缓存数据的完整性和可用性不受影响。
6) 新计算原语的开发: 探索和开发适用于在网缓存的新计算原语，以支持更广泛的数据操作和处理任务。这可能包括在网络层面执行更复杂的数据处理和分析任务。
7) 对数据中心能效和运营成本的影响研究: 分析和优化在网缓存对数据中心整体能效和运营成本的影响。这可能涉及到新的能效优化策略和成本效益分析。
8) 安全性研究: 针对在网缓存系统的安全性进行深入研究，包括数据加密、访问控制和防御网络攻击的策略。这是确保数据隐私和完整性的关键领域。

6 结论

本文主要探讨了现代互联网服务中键值存储需求日益增长的背景下，在网缓存系统的重要性和挑战。文中分析了当前存储领域面临的问题，特别是在面对动态且倾斜的工作负载时的挑战。文献详细介绍了可编程交换机在缓存系统中的应用，尤其关注了读密集型和写密集型工作负载以及分布式共享内存系统。最后，文章还探讨了在网缓存的未来研究方向，包括缓存一致性、分布式共享内存以及可编程交换机的进一步发展。整体上，本文提供了一个对在网缓存技术及其在现代网络服务中应用的全面概述。

参 考 文 献

[1] Xiaozhou Li, Raghav Sethi, Michael Kaminsky, David G. Andersen, and Michael J. Freedman. 2016. Be Fast, Cheap and in Control with SwitchKV. In USENIX NSDI.

[2] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soul´ e, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. NetCache: balancing key-value stores with fast in-network caching. In Proc. of ACM SOSP, 2017.

[3] Siyuan Sheng, Huancheng Puyang, Qun Huang, Lu Tang, and Patrick P. C. Lee"FarReach:Write-back Caching in Programmable Switches" USENIX ATC 2023.

[4] Ming Liu, Liang Luo, Jacob Nelson, Luis Ceze, Arvind Krishnamurthy, and Kishore Atreya. IncBricks: Toward In-Network Computation with an In-Network Cache. In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '17, pages 795–809, New York, NY, USA, 2017. ACM.

[5] Jialin Li, Jacob Nelson, Ellis Michael, Xin Jin, and Dan R. K. Ports. 2020. Pegasus: Tolerating skewed workloads in distributed storage with in-network coherence directories. In Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI'20). USENIX Association, USA, Article 22, 387–406.

[6] Qingchao Cai, Wentian Guo, Hao Zhang, Divyakant Agrawal, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, Yong Meng Teo, and Sheng Wang. Efficient Distributed Memory Management with RDMA and Caching. Proc. VLDB Endow., 11(11):1604–1617, July 2018.

附录

课堂汇报记录：

问题 1：工作负载中的 skewness 是什么原因导致的？

工作负载中的不均衡（skewness）通常是由于数据或任务分布不均匀而引起的。主要可能是以下几个方面的原因

1) 热点数据： 在工作负载中，某些特定的数据项可能频繁地被访问，而其他数据项相对较少被访问。这种热点数据会导致对特定资源的高度竞争，从而引发不均衡。
2) 任务分布不均匀： 如果任务分布不均匀，某些任务可能比其他任务更加密集或耗时。例如，在分布式系统中，一些节点可能需要处理的请求比其他节点更多，导致负载不均衡。
3) 数据倾斜： 数据倾斜指的是数据在存储或处理中的分布不均匀。例如，在关系数据库中，某些表的某一列可能包含大量相同的值，导致查询时某些节点负担更重。

问题 2：为什么需要采用可编程交换机？

在现代网络中，面临着越来越复杂的网络环境，包括大规模数据中心、软件定义网络（SDN）、云计算等。可编程交换机能够更好地应对这些复杂性，为网络提供更强大的处理和控制能力。可编程交换机允许对数据包进行更加细粒度的处理和控制，从而能够实现网络的优化和性能提升。通过在交换机中实现定制化的处理逻辑，可以更好地满足特定应用对网络性能的要求。

问题 3：为什么写请求数占比会更多？

在实时数据处理、日志记录、在线事务处理系统（如银行和电子商务系统）等场景中，经常需要频繁更新和记录数据，从而导致写请求比读请求更多。

在人工智能（AI）训练过程中，写请求可能比读请求多，因为训练模型涉及大量数据的持续更新和调整。训练过程中，算法不断通过迭代调整权重和参数以优化模型性能。这些调整在每次迭代中都需要被记录下来，因此产生大量写操作。此外，生成的中间数据、日志记录以及模型状态的保存也是写操作的一部分。而读操作主要发生在数据集的加载阶段，一旦数据加载完成，重复读取的频率通常会低于模型参数和状态的写入频率。