BDS：用于数据中心间数据复制的集中式近乎最佳的叠加网络

张玉超 蒋俊晨 至 徐
科大 芝加哥大学 清华大学
小辉否 马丁·里德 王海阳
清华大学 埃塞克斯大学 明尼苏达大学德卢斯
广耀 张苗 陈凯
百度 百度 科大

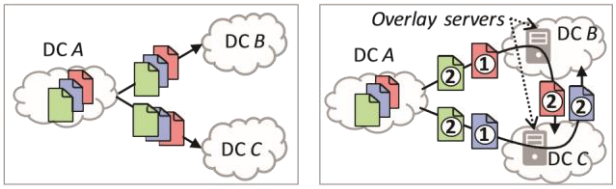
抽象

许多重要的云服务都需要将海量数据从一个数据中心（DC） 复制到多个 DC。虽然对 DC 间数据传输的性能已大为提高，但 pri 或解决方案不足以优化批量数据多播，因为它们无法探索服务器存储和转发数据的能力，以及地理分布式 DC 中存在的丰富的 DC 间覆盖路径。为了利用这些机会，我们提出了 BDS，一 *BDS* 个应用程序级多播覆盖网络，用于大规模 DC 数据复制。BDS 的核心是一个完全*集中的*体系结构，允许中央控制器维护中间服务器数据传递状态的最新全局视图，以便充分利用可用的叠加路径。为了快速响应网络动态和工作负载变化，BDS 通过将控制算法分离到叠加路径的选择和数据传输调度中来加快控制算法的速度，可以有效地优化每个算法。这使得 BDS 能够以在数万个叠加路径上进行数百个 TB 数据的多播规模，近乎实时地（例如每隔一秒）更新覆盖路由决策。在最大的在线服务提供商之一的试点部署表明，BDS 可以实现 3-5+加速比提供商的现有系统和几个众所周知的覆盖路由基线。

*EuroSys '18，2018 年 4 月 23 日至 26 日，葡萄牙*波尔图©2018 年计算机械协会。ACM ISBN 978-1-4503-5584-1/18/04…$15.00
https://doi.org/10.1145/3190508.3190519
ACM 参考格式：
张玉超、蒋俊臣、柯旭、倪晓辉、马丁·里德、王海阳、姚光、张苗、陈凯。2018. BDS：数据中心间集中式近乎最佳的叠加网络数据复制。在*欧元系统'18：第十三届欧洲系统会议 2018 年 4 月 23 日至 26 日，葡萄牙波尔图*。Acm， 纽约， 纽约， 美国， 14 页 https://doi.org/10.1145/3190508.3190519

1 介绍

对于大型在线服务提供商（如 Google、Facebook 和百度），一个重要的数据通信模式*是批量数据的直流*间多播—将大量数据（例如用户日志、网络搜索索引、照片共享、博客文章）复制到地理分布位置的多个 DC 中。我们对百度工作量的研究表明，直流间多播已经占到直流间流量的 91%（§2），这证实了其他大型在线服务提供商的流量模式[27，58]。随着全球部署更多的 DC，大量数据呈爆炸式增长，因此需要以频繁且高效的方式复制 DC 间流量。



(a) Direct replication over pair-wise inter-DC WANs (b) Replication leveraging overlay paths

图 1：一个简单的网络拓扑，说明覆盖路径如何及时减少直流间多播组合。

假设任何两个 PC 之间的 WAN 链路是 1GB/s，并且 *A* 希望向 *B*and 和 C 发送 3GB *数据*。将数据分别从 *A 发送到* to B *和* and *C* 需要 3 秒 （a），但同时使用叠加路径 *A*=*B*=*C*and 和 *A* =C=*B* 只需要 *B* 2 秒 （b）。T 他圈出的数字显示每个数据件发送的顺序。

虽然为提高 DC 网络性能（例如 [22、24、27、44、51、57]）作出了巨大努力，但重点是提高每对 DC 之间的广域网（WAN） 路径的性能。但是，这些以 WAN 为中心的方法不完整，因为它们无法利用跨地理分布式 PC 的丰富应用程序级覆盖路径，以及服务器存储和转发数据

的能力。如图 1 所示，通过作为中间点的多个叠加服务器并行发送数据，可以显著提高直流间多播的性能，从而规避直流网络中缓慢的 WAN 路径和性能瓶颈。我注意到，这些叠加路径应该是瓶颈*脱节*的;也就是说，它们不共享共同的瓶颈链路（例如，图 1 中的 *A*+*B*+*C*and 和 *A*+C+*B*），而且这种瓶颈不相交的叠加路径在地理分布的 DC 中大量 *B* 可用。

本文介绍了 *BDS，一个应用程序级多播*叠加网络，它将数据分割成细粒度单元，并通过瓶颈-非相交的叠加路径并行发送。这些路径是动态选择的，以响应网络条件的变化和每台服务器的数据传递状态。请注意，BDS 选择应用程序级覆盖路径，因此是对 WAN 性能的 n 和工作层优化的补充。虽然应用级多播叠加已应用于其他上下文（例如 [9、29、34、48]），但构建一个用于直流间多播流量带来了两个挑战。首先，由于每个 DC 都有数万台服务器，因此大量可能的叠加路径使得实时大规模更新叠加路由决策变得难以操作。先前的工作要么依赖于单个服务器的本地反应性决策 [23，26，41]，这导致缺乏全局信息导致次优，要么限制自己严格结构化（例如分层）拓扑 [37]，因为拓扑无法利用所有可能的叠加路径。其次，即使延迟敏感流量的延迟增加，也可能导致收入损失[56]，因此必须严格控制 DC 间大容量数据多播的带宽使用情况，以避免对其他延迟敏感流量产生负面影响。

为了应对这些挑战，BDS *全面集中了*-DC 间多播的调度和路由。与服务器必须保留某些本地决策才能实现理想的可伸缩性和对网络动态的响应的直觉相反，BDS 的集中式设计建立在两个经验观察 （§3）： （1） 虽然实时进行集中式决策是 1，但大多数多播数据传输至少持续几十秒，因此可以容忍稍微延迟的决策，以换取基于全局视图的近乎最佳的路由和调度。（2） 集中分配发送速率，可最大限度地减少直流间多播流量与延迟敏感流量之间的干扰。

使 BDS 实用的关键是如何在性能改动和请求动态到达时，近乎实时地（在几秒钟内）更新覆盖网络。BDS 通过将集中控制*分离为*两个优化问题（数据传输调度和覆盖单个数据传输的路由）来实现此目的。这种脱钩实现了亲 vv 可优化，同时，BDS 允许在一秒秒内更新覆盖网络路由和调度;在考虑大型在线服务提供商的工作负载时，这比共同解决路由和调度快四个数量级（例如，沿 10^4 个脱节叠加路径同时发送 $10^{\uparrow 5}$ 个数据块）。

我们实现了 BDS 的原型，并整合到百度中。我们在 10 个 DC 中部署了 BDS，并进行了 7 天 500 TB 数据传输的试点研究（每天 71 TB）。我们的实际实验表明，BDS 比×百度现有的名为"Gingko"的解决方案实现了 3-5°的加速，可以消除批量数据传输导致带宽消耗过大的事故。利用跟踪驱动仿真和 micro 基准测试，我们还表明：BDS 优于 CDN 中广泛使用的技术，BDS 可以通过一个通用服务器处理百度直流间多播流量的工作负载，BDS 能够处理各种故障方案。

我们的贡献总结如下：

- 描述百度在直流间批量数据多播的工作负载，以激发对应用级多播覆盖网络（§2）的需要。
- 呈现 BDS，一个应用程序级多播覆盖网络，通过 c ent 分管控制体系结构（§3，4，5） 实现近乎最佳的流完成时间。
- 通过在百度 （+6） 进行真实世界试点部署，展示 BDS 的实际优势。

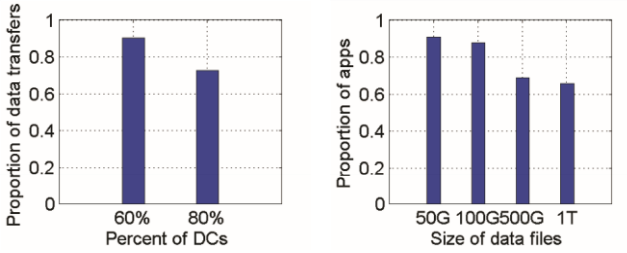
2 应用级直流间多播叠加的案例

我们首先为应用程序级多播 ov erlay 网络提供案例。我们首先在百度（一家全球规模的在线服务提供商）中描述直流间多播工作负载。然后，我们展示利用地理分布式 DC（§2.2）中提供的不相交的应用程序级覆盖路径，提高多播性能的机会。最后，我们考察了百度当前的直流间多播（Gingko）解决方案，并吸取真实事件的教训，为 BDS 的设计提供信息（§2.3）。这些发现是基于百度在 7 天内对 DDC 间流量的数据集。该数据集由 30 多个地理分布的 DC 之间的大约 1265 个多播传输组成。

应用程序类型	多播流量的百分比
所有应用程序	91.13%[1]
博客文章	91.0%
搜索索引	89.2%
脱机文件共享	98.18%
论坛帖子	98.08%
其他数据库同步	99.1%

表 1：直流间多播（将数据从一个 DC 复制到许多 DC）主导了百度的直流流量。

[1] 总体多播流量共享是使用通过一个随机采样的 DC 的流量估算的，因为我们无法访问所有 DC 间流量的信息，但此数字与我们观察到的其他 DC 的流量一致。



（a） 多播（b）多播传输比例，用于传输百分比大于某些发展中国家阈 值。

图 2：直流间多播 （a） 注定为 DC 的很大一部分，（b） 具有较大的数据大小。

2.1 百度的直流间多播工作负载

直流间多播流量份额：表 1 显示 DC 间多播（将数据从一个 DC 复制到多 PCDC）作为所有直流间流量的一小部分。我们看到，DC 间多播主导了百度的整体 DC 流量（91.13%），以及单个应用程序类型的流量（89.2 至 99.1%）。DC 间多播流量相当于

直流间流量的主导地位凸显了*优化直流间多播性能的重要性*。DC 间多播的目的地在哪里？接下来，我们想知道这些传输是否被目的地到大部分（或只是少数）DCs，以及它们是否共享共同的目的地。图 2a 草图显示百度 DC 的分布百分比，该百度将转移到多播传输的目的地。我们看到，90% 的多播传输注定至少 60% 的 DC，70% 的 DC 被访问到 80% 以上的 DC。此外，我们发现源 DC 和目标 DC 集的多样性很大（此处未显示）。这些观察表明，预先配置所有可能的多播请求是站不住脚的;相反， *我们需要一个系统来自动路由和安排任何给定的 DC 间多播传输。*

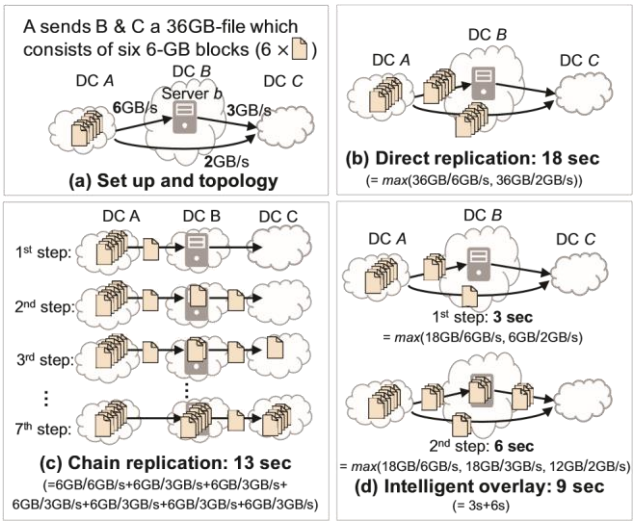


图 3：比较智能应用程序级叠加 （d） 与基线性能的示例：天真的应用程序级叠加 （c） 和无过度铺设 （b）。

直流间多播传输的大小：最后，图 2b 概述了直流间多播的数据大小的分布。我们看到，超过 60% 的多播传输，文件大小超过 1TB（90% 超过 50GB）。鉴于分配给每个多播的总 WAN 禁令 dwidth 按多个 Gb/s 的顺序排列，因此这些传输不是瞬态的，而是持久的，通常持续至少几十秒。因此，*任何优化多播流量的方案都必须动态地适应数据传输过程中的任何穿孔变化*。另一方面，这种时间持久性还意味着*多播流量可以容忍由集中控制机制（如 BDS* （§3））引起的少量延迟。

这些观测结果共同激励他采用系统化的方法优化直流间多播性能。

2.2 直流间应用级叠加的潜力

众所周知，通常可以使用应用程序级叠加 （13} 提供多播。在这里，我们显示，应用程序级叠加网络可以大大减少 DC 间 mult icast 完成时间（由时间定义，直到每个目标 DC 都有数据的完整副本）。请注意，应用程序级叠加不需要任何网络级支持，因此，与之前对 WAN 优化的工作一样，它非常适用。

应用程序级叠加网络的基本思想是沿瓶颈-脱节的叠加路径 *bottleneck-disjoint* [15] 分发流量，即两个路径不共享一个常见的瓶颈链路或中间服务器。在 DC 间传输的上下文中，两个叠加路径可以遍历不同的 DC 序列

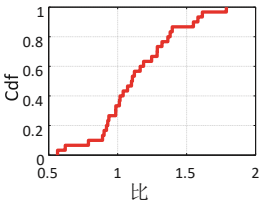


图 4：存在显著的性能差异

在我们网络中的直流覆盖路径中，表示大多数叠加路径对是瓶颈脱节。

（*类型 I*），或遍历相同目录序列的不同服务器序列 （类型 *II*），或两者的某种组合。接下来，我们使用 examples 来显示两种类型的叠加路径中可能出现的瓶颈-脱节覆盖路径，以及它们如何提高直流间多播性能。瓶颈-脱节叠加路径的示例：在图 1 中，我们已经看到两个 I 型叠加路径 and （*A=B=→C* 和 *A=C=B*） 是如何成为瓶颈脱节的，以及它是如何提高直流间多播的性能的。图 3 显示了类型 II 瓶颈-分离叠加路径（遍历相同的 DC 序列，但不同的服务器序列）的示例。假设我们通过两个瓶颈分离路径将 36GB 数据从 DC *A*to 复制到 *B*and to a server 和 to :

from to *C* C：（1） *A+C：* 从 *A* 到 *B* 到 *C* 使用具有 2GB/s 容量的 IP 层 WAN 路由，或 （2） *A= b=C：* 从 *A* 复制到 B 中的*服务器 b*，容量为 6GB/s，b 到 *C* 的 3GB/s 容量。数据被分成六个 6GB 块。我们考虑三种策略。（1） *直接复制*：如果 *A* 通过 *WAN 路径直接* and 将数据发送到 B 和 *C*（图 3（b），则完成时间是 18 秒。（2） *Simple chain replication* 简单链复制：应用程序级 overlay 路径的天真使用是通过服务器 *b* 发送块作为存储和中继点（图 3（c），完成时间是 13 秒（比没有叠加的要少 27%）。（3）*智能多播*叠加：图 3（d）通过同时沿两条路径选择 ily 发送块，在 9 秒内完成（比链复制少 30%，比直接复制少 50%），进一步提高了性能。

野生的瓶颈-非相交叠加路径：很难识别网络性能数据集中的所有瓶颈-非相交的 ove rlay 路径，因为它没有每个多播传输的每跳带宽信息。相反，我们观察到，如果两个叠加路径同时具有不同的端到端吞吐量，它们应该是瓶颈-d 是联合的。我们举了一个野外瓶颈-脱节覆盖路径的示例，该路径由*两个*叠加路径 A+*b*=*C*and 和 *A+C* 组成，其中从 DC *A* 到 DC C 的 WAN *路由*通过 DC *B*，b*b* 是 B 中的*服务器*（这两个路径是

在拓扑上与图 3 相同）。如果 $_{Bw}{}^{Bw}{}_{A=}{}^{A=}{}_{b=}{}^{C}$ C 6 = 1，则它们是瓶颈脱节（BW_p 表示路径的吞吐量

p）图 4 显示了数据集中$_{所有}{}^{可能}$值 $_{A、bC}$ 和 C 的 *BW* $BW_{A=}{}^{A=}{}_{b=}{}^{C}$ *C b* 的分布情况。我们可以看到，超过 95% 的 *A+ b=→C* 和 *A+C* 对具有不同的端到端吞吐量，即它们是瓶颈脱节。

2.3 现有解决方案的限制

实现和演示应用级覆盖网络的潜在改进存在一些复杂性。作为一阶近似值，我们可以简单地借用其他上下文中的多播叠加网络的现有技术。但百度的操作经验显示了这种方法的两个局限性，将在下面描述。

百度现有的解决方案：为了满足 DC 间数据复制快速增长的需要，百度几年前就部署了应用级覆盖网络"金科"。尽管经过多年的改进，Gingko 仍基于接收器驱动的分散式叠加多播协议，类似于其他叠加网络（如 CDN 和基于叠加的实时视频流 [9、47、55]）。基本 idea 是，当多个 DC 从源 DC 请求数据文件时，请求的数据将流回中间服务器的多个阶段，其中每个阶段的发送方选择由下一阶段的接收方以分散化 ed 方式驱动。限制 1：当地适应效率低下。现有的分散协议缺乏全局视图，因此受到次优调度和路由决策的影响。为了显示这一点，我们在 Bai du 的网络中向两个目标 DC 发送了一个 30GB 文件。每个 DC 都有 640 台服务器，每个服务器具有 20Mbps 的上传和下载带宽（在分配给生产流量中每个批量数据传输的带宽大小相同）。这个 30GB 的文件均匀地存储在所有这些 640 台服务器上。理想情况下，如果服务器为所有块选择最佳源，则完成时间为 $\frac{30\times1024}{640\times20Mbps\times60s/min}=41$ 分钟。但如图 5 所示，

目标 DC 中的服务器平均需要 195 分钟（4.75=最佳完成时间）才能接收数据，5% 的服务器甚至等待时间超过 250 分钟。此问题的关键原因是，单个服务器只能看到可用数据源的子集（即已下载文件一部分的服务器），因此无法利用所有可用的叠加路径来最大化吞吐量。即使叠加网络只是部分分散（例如 [23]），即使每台服务器都有全局视图，单个服务器的本地调整仍会创建覆盖路径上的潜在热点和拥塞，也可能发生这种次优性能。

限制 2：与延迟敏感流量的交互。现有的多播覆盖网络共享相同的 DC 间 WAN，具有对延迟敏感的流量。尽管使用标准的 QoS 技术，并且给最低 rity to 原始

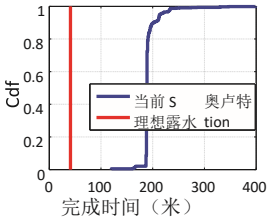


图 5：与理想解决方案相比，目标 DC 中不同服务器的实际流完成时间 CDF。

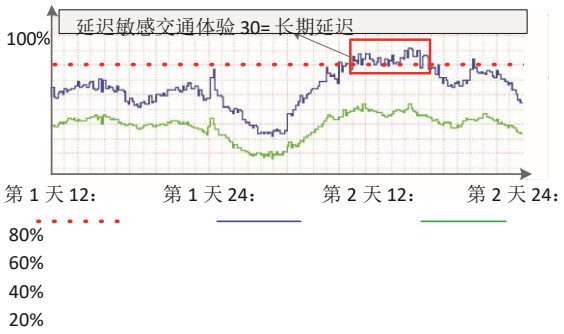


图 6：两天内直流间链路的利用率。第二天直流批量数据传输对延迟敏感流量造成严重的互不通。

批量数据传输，我们仍然看到批量数据多播请求的突发到达对延迟敏感流量的负面影响。图 6 显示了 INTERDC 链路在两天内的带宽利用率，在此期间，长达 6 小时的批量数据传输在第二天晚上 11：00 开始。蓝线表示传出带宽，绿线表示传入带宽。我们可以看到，批量数据传输导致链路利用率过高（即超过 80%的安全阈值），因此，对延迟敏感的在线流量经历了超过 30+ 延迟的膨胀。

2.4 主要观察

本节的主要观察结果如下：

- *直流间多播*占直流间流量的一小部分，在源宿率上具有很大的变异性，通常持续至少几十秒。
- *瓶颈-不相交的*叠加路径在地理分布的 DC 之间随处可见。
- 现有的解决方案，即依靠本地适应，可能会对*在线流量产生欠* and *佳的性能和负面影响*。

3 BDS 概述

为了优化直流间多播，同时最大限度地减少对延迟敏感流量的干扰，*我们提出了 BDS*，一个完全*集中*的接近最佳的应用程序级覆盖网络，用于直流间多播。在详细介绍 BDS 之前，我们

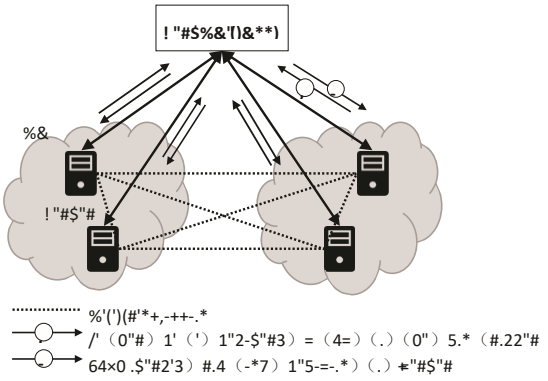


图 7：BDS 的集中设计。

首先强调其设计选择背后的直觉，以及使其实用的挑战。集中控制：对广面积覆盖网络的传统智慧在某种程度上依赖于 *local* 单个节点（或中继服务器）的本地调整，以实现理想的可扩展性和对网络动态的响应能力（例如，[9、23、35、41]），尽管由于缺乏全局视图或编排，导致性能欠佳。相比之下，BDS 采取明确的立场是，在直流间多播的设置中，集中控制广面积覆盖网络，仍然可实现近乎最佳的性能，是切实可行的。BDS 的设计与最近关于集中管理大规模分布式系统的其他工作相吻合，例如，[20] 在高级别上，BDS 使用集中式控制器，定期从所有服务器提取信息（例如，数据传递状态），更新有关路由叠加的决策，并将它们推送到服务器上本地运行的代理（图 7）。当控制器发生故障或无法访问时，BDS 将回退到分散控制方案，以确保性能正常下降，以适应本地自适应 （§5.3）。

BDS 的集中式设计由以下几个经验观察驱动：

1. *较大的 decision 空间：*直流间叠加路径的绝对数量（随着更多服务器充当叠加节点而呈指数级增长），使得单个服务器很难仅根据本地测量来探索所有可用的叠加路径。相比之下，我们可以通过保持所有服务器的数据交付状态的全局视图，并动态平衡各种数据块的可用性，从而显著提高叠加多播性能，这对于实现近乎最佳的穿孔（§4.3）至关重要。
2. *大数据大小：与延迟敏感型流量不同，延迟敏感流量在几*到 10 毫秒的时间尺度下持续，
   直流多播在更粗糙的时间尺度上持续。因此，BDS 可以容忍短暂的延迟（几秒钟），以便从维护数据传输全局视图并能够协调所有叠加服务器的集中式控制器获得更好的路由决策。
3. *严格的流量隔离：*如 §2.3 所述，DC 间多播必须避免热点，因为带宽使用过多会对延迟敏感流量的延迟产生负面影响，但如果没有覆盖服务器之间的任何协调，则很难防止此类情况。相比之下，通过以集中方式控制所有服务器的发送速率（§5），可以更简单地分配每个数据转送 r 的带宽。
4. *工程复杂性较低：*从概念上讲，集中式体系结构将控制复杂性移动到集中式控制器，使 BDS 易于实现，在每台服务器中本地运行的控制逻辑可以是无状态的，并且仅在新数据单元或控制消息到达时触发。

实现集中控制的关键：从本质上讲，BDS 的设计在产生小更新延迟来换回集中式系统带来的近乎最佳决策之间实现了权衡。因此，实现这种有利平衡的关键是近乎最佳但高效的叠加 r 外向算法，该算法可以近乎实时地更新决策。乍一看，这确实难以解决。对于百度规模的工作负载，集中式叠加路由算法必须从 10 个 4 台服务器中为 10 ^5 个数据块选择下一个跃点。当我们考虑通过这些服务器的可能叠加路径数量的增长以及更细粒度的块分区时，Thi s 的运行规模可能会呈指数级增长。在标准路由公式和线性编程中，通过探索如此大的决策空间（§6.3.4）来制定近乎最佳的解决方案可能完全不现实。下一节将介绍 BDS 如何应对这一挑战。

4 近乎最佳和高效的决策逻辑

BDS 的核心是一种集中式决策算法，它可近乎实时地定期大规模更新叠加路由决策。BDS 通过将控制逻辑 *decoupling* 分离为两个步骤（§4.2）： 叠加调度（即要发送哪些数据块 （§4.3） 和路由（即用于发送每个数据块的路径 （§4.4））在解决方案优化和接近实时更新之间实现有利的权衡，每个步骤都可以高效且近乎最佳地解决。 the control logic into two steps (

4.1 基本配方

我们首先提出叠加交通工程的问题。表 2 总结了关键符号。

BDS 中的叠加流量工程在空间和时间上均以精细粒度运行。要利用源和目标 DC 之间的许多叠加路径，BDS

变量 s	意义
B	所有任务的块集
B	块
	The size of block *b* Set of paths between a source and destination pair
P	特定路径
我	路径上的链接
c(*（I）*	链路 l 的*容量*
·*T*	计划周期
T_k	*k*- th 更新周期
wb， *sk*） （*T*	二进制：*如果 s* 被选为 T k 的 *b* 的目标 *T 服务器*
R 向上（*s*）	服务器的上传*容量*
R 向下 （*s*）	服务器的下载*容量*
fb， *pk*） （*T*	分配给在 *T k 的路径* p 上*发送 b 的带宽*

表 2：BDS 决策逻辑中使用的符号。

将每个数据文件拆分为多个数据块（例如 2MB）。为了应对网络条件和请求到达的变化，BDS 每+T*T* 更新覆盖流量工程的决策（默认情况下为 3 [2] 秒）。

现在，多播叠加路由问题可以表述如下：

输入：BDS 采用以下参数输入：所有数据块 Bto *s* 的集，每个块 *b s* 的大小 = ρ(*（b），*)从服务器*到 s* 的路径集，P $P_{s=,s,}$ 更新周期 =*T*，*并且*对于每个服务器的上传（resp. 下载） 容量 R_{up}(*（s）* （resp）。 $R_{向下}$（*s*）.请注意，每个路径 *p* 由多个链接 *I* 组成，每个链接由一对服务器或路由器定义。我们使用 *c*（*I*） 来表示链路 l 的容量 *I*。

,*s* destined to 输出：对于每个周期 *T* $s_{k,}$, block *块 b*、服务器 s 和路径 *p* ∈ P_{s+} ，*BDS* 返回为输出 2 元组 ∈ $\langle w^{(T_k)}, f^{(T_k)} \rangle$

$T_p{}^k$ f_b(, *bb*，*ss bb*，*p*，*p* , in which *其中* $w^{(b}{}_b{}^{T,}$ $_s{}^{k)}$ 表示是否选择了服务器*s 由于块*b 的目标服务器*在* $T_{k,}$ f b （， T p k ） 表示分配多少带宽发送块 b 在路径 *p* 在 $T_{k,}$, and 和 $f_b{}^{(,}$, $T^{T}{}_p{}^{k)}$ = 0 表示*路径 p* 未选择发送块 *b* 在 T_k 中。

约束：

- 路径 *p* 上的分配带宽不得超过*任何*链路 l 的容量 *p*，以及源服务器 R_{up}（*s*） *的上传*容量和目标服务器 *R*(*向下*（ss_ ）的下载容量。

$$f_b{}^{(T_k)}{}_{,p} \le min\left(min_{l\in p}c(l), q_{b,s'}^{(T_k)} \cdot R_{up}(s'), w_{b,s}^{(T_k)} \cdot R_{向下}(s)\right)$$

对于\b， p ∈ $P_{s\_,\ ,s}$ （1）

其中 $q^{(T_k)}_{b\ ,s}$ $= 1 - \prod_{i<k}(1 - w_{b,s}^{(T_i)})$ 表示在循环 T *k* 之前是否选择*服务器 s 作为块 b 的目标 k*。

- 对于所有路径，链路的汇总分配带宽不应大于其容量 *c*（*I*）。。

$$c(l) \ge \sum_{b\in B} f_{b,p}^{(T_k)}, \text{for } \forall l \in p \qquad (2)$$

- 在每个 cy cle 中选择发送的所有块必须在 +T 中完成其*传输*，也就是说，

$$\sum_{b\in \mathbb{B}} w_{b,s}^{(T_k)} \cdot \rho(b) \le \sum_{p\in \mathbb{P}} \sum_{b\in \mathbb{B}} f_{b,p}^{(T_k)} \cdot \Delta T, \text{for } \forall \quad Tk \qquad (3)$$

- 最后，所有模块必须在所有周期结束时传输。

$$N$$

$$\in k{=}1p\ Tpk) \ (4)\ bB\ ,b(b) \le \sum\sum\sum f\rho\in(\backslash$$

$$(\ b\ ) = \text{----------}$$

$$\text{----------}$$

$$\text{----------}\ P\ b \in B$$

目标：我们希望最大限度地减少传输所有数据块所需的周期数。也就是说，我们返回为最小整数 N，*N* 上面的约束具有可行的解决方案。

不幸的是，这种提法在实践中对两个儿子来说是难以解决的。首先，它是超线性和混合整数，因此计算开销随着更多潜在源服务器和数据块而呈指数级增长。其次，要找到最小整数 *N*，我们需要检查问题的可行性，为不同的值 *N*。.

[2]我们使用固定间隔为 3 秒，因为它足够长，BDS 可以根据百度的工作负载来更新决策，并且足够短，可以适应典型的性能变化，而不会对批量数据传输的完成时间产生明显影响。更多德泰 ls 在 # 6 中

4.2 分离调度和路由

在高层次上，BDS 背后的关键见解是将上述配方分成两个步骤：一个调度 *scheduling* 步骤，用于选择每个周期要传输的子块集

（即 w w_b〔^{T}, s^k〕），后跟*路由*步骤，该步骤选择路径并分配带宽以传输所选

块（即 f f_b〔·, ^{T}p$_k$^{k}〕。

这种脱钩显著减少了集中式控制器的计算过度广告。由于调度步骤选择块的子集，并且在后续路由步骤中只考虑这些选定的块，因此搜索空间将显著减少。从数学上讲，通过将 scheduling 步进与问题公式分离，路由步骤减少到混合整数 LP 问题，虽然不能立即处理，但可以通过标准技术来解决。接下来，我们将介绍每个步骤的更多详细信息。

4.3 调度

调度步骤 s 选择块的子集

在每个周期中转移，即 w〔 b_b^T, s^k〕。

调度（选择块的子集）的关键就是确保后续数据传输能够以最有效的方式完成。灵感来自

在 BitTorrent [14] 中，BD S 采用一种简单而高效的选择数据块的方法：对于每个周期，BDS 只需选择重复量最少的块子集。换句话说，BDS 通过在每个循环 e 中选择一组块而不是单个块的副本来概括最稀有的先方法。

4.4 路由

计划步骤选择在每个时隙（*w*〔$^{b}b^T$·, s^k〕中传输的块集后，路由步骤决定路径并分配带宽以传输所选块（即 f_b〔·, $^T p^k$〕）为了缩短传输完成时间，BDS 可最大化每个周期 T k 的吞吐量（传输的总数据 $T_{量}$）。

$$max \sum_{p\in Pb\in B} \sum f_{b(,Tpk)} \quad (5)$$

这当然是一个近似值，因为贪婪地在一个周期中最大化吞吐量可能会导致数据可用性不理想，并降低下一个周期的最大可访问吞吐量。但在实践中，我们发现，这种近似值会导致与基线的近似值性能提高，部分原因是上一节中描述的调度步骤会自动平衡块的可用性，因此，由于过去周期中贪婪的路由 de ccision 导致的数据可用性不足（例如，块的不足），这种情况很少发生。

此公式与 [4.1]的约束本质上是一个整数多商品流（MCF） 算法，已知为 NP-完成 [19]。若要在实践中解决此问题，标准近似值假定每个数据文件可以无限小地在源和目标之间的一组可能路径上同时拆分和传输。BDS 的实际路由步骤与此近似值非常相似，因为 BDS 还将数据拆分为数以万计的细粒度数据块（虽然不是极小），并且可以通过 MCF 问题中常用的标准线性编程（LP） 松弛来有效解决[18， 39]。

但是，当无限拆分任务时，块的麻木 r 将变得相当大，计算时间将难以忍受。BDS 采用两种应对策略：（1） 将块与相同的源和目标对组合，以减少问题大小（详见§5.1）；（2） 它使用完全证明的 im-多分时近似方案（FPTAS）[17]来优化原始问题的双重问题，并找出一个+-最佳的解决方案。这两种策略进一步缩短了集中算法的运行时间。

5 系统设计

本文介绍了 BDS 的系统设计和设计。

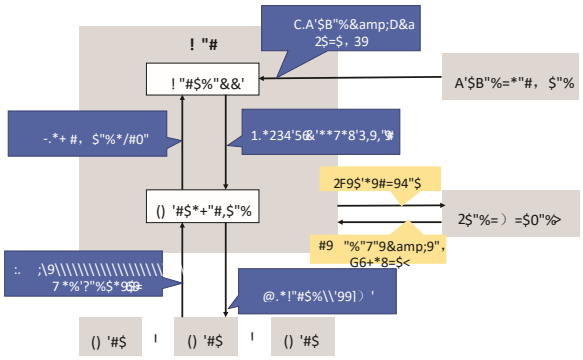


图 8：BDS 集中控制的接口。

5.1 BDS 的集中控制

BDS 会定期（默认情况下，每三秒一次）以集中方式更新路由和计划决策。图 8 概述了每个三秒钟周期 e 中的工作流。

(1) 它从 Agent *开始*，在每台服务器上运行本地，检查本地状态，包括数据块传递状态（哪些块已到达，哪些块尚未解决）、服务器可用性和磁盘故障等。

(2) 然后，这些统计信息被包装在*控制消息中*，然后通过称为代理监视器*的*高效消息传递层发送到集中式 BDS *控制器*。

(3) BDS 控制器还从网络监视器接收网络级统计信息（对延迟敏感的流量的带宽消耗和每个 DC 间链路上的带宽*消耗*）。

(4) 在收到来自所有代理和网络监视器的更新时，BDS 控制器运行集中决策算法 （§4） 以计算新的调度和路由决策，并将新决策与上一个决策之间

的差异发送到每个服务器本地代理通过代理监视器消息传递层。

(5) 最后，代理为每个数据传输分配带宽，并根据 Controll er 的路由和调度决策执行实际数据传输。

BDS 使用两个附加优化来提高工作流的效率。

- *块合并*。为了减少计算规模并实现更高效的传输，BDS 将具有相同源和目标块合并到一个子任务中。其好处是双重的：（1）它显著减少了每个 scheduling 周期中挂起的块数，从而降低了集中决策逻辑的计算成本;（2） 它减少了服务器之间的并行 TCP 连接数，否则可能会降低链路利用率和性能降低。
- *非阻塞更新*。为了避免被控制器的决策逻辑阻止，每个本地代理在控制器运行集中决策逻辑时保持正在进行的数据传输活动。同样，主计长在重新计算决策时推测数据交付状态的变化，并使用这些推测的数据分离状态作为集中逻辑的输入来考虑这一点。

5.2 动态带宽分离

为了保证跨

DC 大容量数据多播和延迟敏感流量，BDS 网络监视器监控每个 DC 内部/DC 内部链路上所有延迟敏感流的总带宽使用情况，并动态分配每个直流间多播传输的带宽。为了使延迟敏感流免受突发批量数据传输的负面影响，BDS 使用 80% 的链路利用率作为"安全阈值"，即批量数据传输的总带宽消耗不能超过任何 lin k 上链路容量的 80%，并动态决定每次数据传输的发送速率。

BDS 动态带宽分离的主要优点是，它以集中方式高效分配带宽，在传输层中类似 [27]。在动态网络环境存在的情况下，赋予在线延迟敏感流量更高优先级的传统技术（例如[27]）仍可能造成带宽浪费或性能干扰 [49]。相比之下，BDS 动态监控延迟敏感应用程序的聚合带宽，并计算 DC 间多播使用的残余带宽，同时满足安全链路利用率阈值。最后，请注意，BDS 优化了应用程序级叠加，因此与提高 WAN 性能和公平性的网络层技术相辅相成 [10、25、32、40]。

5.3 容错

接下来，我们将介绍 BDS 如何处理以下故障。

1. *控制器故障*：BDS 控制器被复制 [28]：如果主控制器发生故障，将选择另一个代表 lica 作为新控制器。如果所有控制器副本都不可用（例如，在 DC 和控制器之间的网络分区），则服务器中运行的代理将回退到当前分散的叠加协议，作为默认值，以确保性能正常下降。
2. *服务器故障*：如果服务器上的代理仍然能够工作，它将在下一个周期向代理监视器报告故障状态（例如服务器崩溃、磁盘故障等）。否则，选择此服务器作为数据源的服务器将报告代理监视器的不可用。在这两种情况下，控制器将在下一个周期中从潜在数据源中删除该服务器。
3. *DC 之间的网络分区：如果*网络分区发生在 DC 之间，则位于与控制器同一分区中的 DC 将工作与以前相同，而分离的 DC 将回退到分散的覆盖网络。

5.4 实施和部署

我们已经实现了 BDS，并部署在 B aidu 的 10 个地理分布 DC 的 67 台服务器上。BDS 使用 3621 行 golang 代码实现 [1]。

控制器的复制在三个不同的地理位置动物园管理员服务器上实现。代理 Monitor 使用 HTTP POST 在控制器和服务器之间发送控制消息。BDS 使用 wget 进行每次数据传输，并在每次数据传输中设置-限制速率字段来强制分配带宽。在每个服务器中运行的代理使用 Linux Traffic 控制 （tc） 来强制限制 DC 间多播流量的总带宽使用情况。

BDS 可与任何直流间通信模式无缝集成。所有应用程序需要做的就是调用由三个步骤组成的 BDS API：（1）向 BDS 提供源 DC、目标 DC、中间服务器和指向 th e 批量数据的指针;（3） 向 BDS 提供批量数据;（3） 向 BDS 提供批量数据。（3） 提供 BDS，提供源 DC、目标 DC、中间服务器和指向 the 批量数据的指针;（2） 在所有中间服务器上安装代理;（3）最后，设置数据传输的开始时间。然后 BDS 将在指定时间开始数据分发。我们推测，我们的实施也应当适用于其他公司。

BDS 有几个参数，这些参数要么由百度管理员设置，要么通过评估结果进行实证设置。这些参数包括：为延迟敏感流量保留的带宽 （20%）、数据块大小 （2MB） 和更新周期长度 （3 秒）。

6 评价

使用百度 DC 中的试点部署、跟踪驱动仿真和微台标记的组合，我们显示：

1. BDS 完成直流间多播 3-5+比百度现有解决方案快，以及行业中其他基线（+6.1）。
2. BDS 显著减少了批量数据多播流量和延迟敏感流量（§6.2） 之间的干扰事件。

3. BDS 可以扩展到大型在线服务提供商的流量需求，容忍各种故障，并实现接近最佳流量完成时间（§6.3）。

6.1 与现有解决方案的性能改进

*6.1.1 方法。*基线：我们将 BDS 与三个现有解决方案进行比较：银杏（百度现有的分散式直流间多播战略 y）、子弹 [26]和 Akamai 的叠加网络 [9]（多播实时视频的集中式策略）。

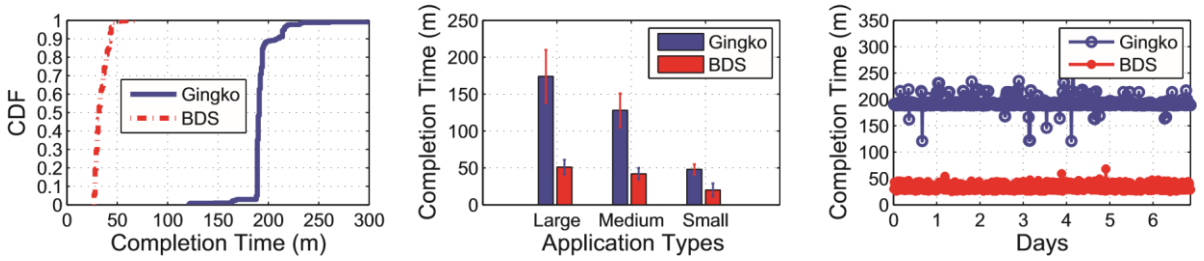
试点部署：我们选择多个不同数据大小的服务，并运行 A/B 测试，其中我们运行 BDS，而不是百度的默认解决方案 Gingko 在随机选择的几天内相同的时间。

跟踪驱动仿真：作为实际流量的试点部署的补充，我们还使用跟踪驱动仿真对 BDS 进行更大规模的评估。我们使用与 BDS 相同的拓扑、服务器数量和链路容量模拟其他两个叠加多播技术，并按与试验部署相同的时间顺序重播直流间多播数据请求。

*6.1.2 BDS vs. 金科。*我们首先评估一个需要将 70 TB 数据从一个源 DC 分发到 10 个目标 DC 的服务上的 BDS 和 Gingko。 图 9a 显示了每个目标服务器上完成时间的累积分发函数 （CDF）。我们可以看到，BDS 的中位完成时间是 35 分钟，比 Gingko 快 5°，大多数 DC 需要 190 分钟才能获取数据。

为了概括这一发现，我们选取了三个数据量为大、中、小的应用程序，并比较了图 9b 中每个应用程序的完成时间（和标准偏差）BDS 和 Gingko 的平均值（和标准偏差）。我们看到 BDS 始终优于金科，并且性能差异较小。我们还看到 BDS 在数据大小较大的应用程序中有更大的改进。最后，图 9c 显示了 BDS 和金科在一个随机选择的应用程序中平均共吸时间的时间序列，我们看到 BDS 一直优于光科 4+。.

6.1.3 BDS vs. other overlay multicast techniques. 表 3 将 BDS 与其他两个基线（项目符号和 Akamai 的叠加网络，使用跟踪驱动仿真。我们在三个设置中显示结果。在基线评估中，我们将 10TB 数据从一个 DC 发送到 11 个 DC，每个 DC 有 100 台服务器，uplo 广告和下载链接容量设置为 20MB。在大规模评估中，我们发送



（a） 完成时间的分配。 （b） 按应用程序类型进行比较。 （c） 按完成时间比较。

图 9：[BDS 与 Gingko（百度现有解决方案）] 是试点部署的结果。

同一个 DC 之间的 100TB 数据，每个 DC 都有 1000 台服务器。在速率限制评估中，设置与基线实验中的设置相同，但 Server 上传和下载速率限制设置为 5MB 除外。我们看到，BDS 在基准设置× 中完成时间比子弹和 Akamai 短 3 倍，在大规模× 和小型带宽设置中完成时间短 4 倍，这证实了

解决 方案	基线	大型	费率限制
子弹	28 米	82 米	171 米
Akamai	25 米	87 米	138 米
Bds	9.41 米	20.33 米	38.25 米

表 3： [BDS 与项目符号 [26]，Akamai [9] 跟踪驱动仿真中三个解决方案的完成时间。

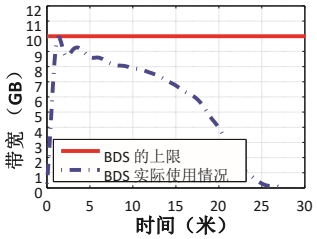


图 10：带宽分离的有效性。

在 §6.1.2 中发现，当数据大小较大时，BDS 有更大的改进。

6.2 协调带宽分配的好处

为了减少大容量数据多播流量和延迟敏感流量之间的负干扰发生，百度根据链路容量与延迟敏感流量的历史峰值带宽之间的差异，在每个链路上设置批量数据传输的

带宽限制。如果 BDS 能够将大容量数据多播流量保持在带宽限制内，那么我们可以最大限度地减少冲突。为了测试 BDS 在保持批量数据多播流量和延迟敏感流量之间的区分方面的有效性，我们对批量数据传输设置了 10GB/s 的带宽限制。图 10 显示了一个 DC 间链路上 BDS 的实际带宽使用情况。我们可以看到，在 BDS 中，批量数据使用的实际带宽始终低于 10 GB/s。

6.3 微观基准

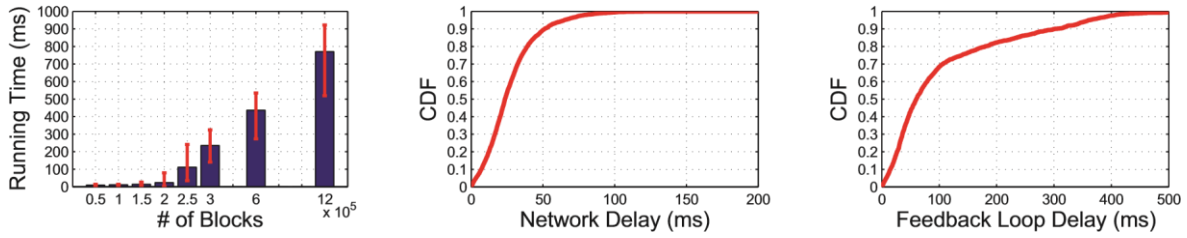
接下来，我们使用微基准测试来沿三个指标对 BDS 进行扩展：（1） 集中控制的可扩展性;（3） 集中控制的可扩展性;（3） 集中控制的可扩展性;（3） 集中控制的可扩展性;（3） 集中控制的可扩展性;（2） 容错;（3） BDS 参数的优值。

*6.3.1 可扩展性。*控制器运行时间：由于控制器需要决定每个数据块的调度和路由，控制逻辑的运行时间自然会随着块数而扩展。图 11a 将运行时间作为块总数的函数显示。我们可以看到，集中式 BDS 控制器可以在 800ms 内更新调度和路由决策，并设置 $10^{\uparrow 6}$ 个模块。从这个角度看，在百度的 DC 中，同时未完成的数据块的最大数量约为×3×105，BDS 可以在 300 毫秒内完成决策的更新。[5]

网络延迟：BDS 在 DC 间网络中工作，因此 DC 之间的网络延迟是算法更新过程中的关键因素。我们记录了 5000 个请求的网络延迟，并在图 11b 中显示 CDF。我们可以看到，90% 的网络延迟低于 50*ms*，平均值约为 25*ms*，这不到决策更新周期（3 秒）的 1%。

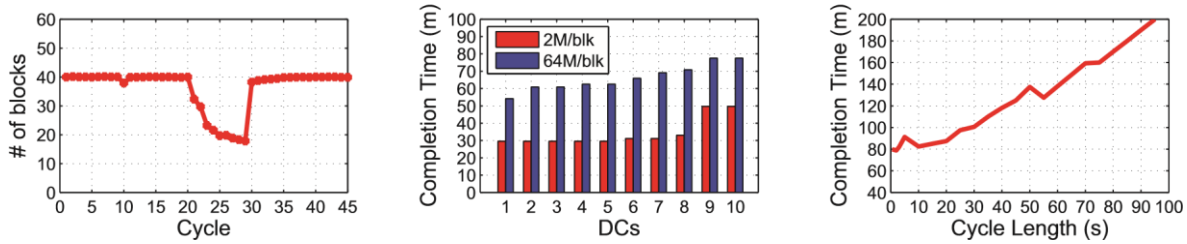
反馈循环延迟：对于集中式算法，小反馈循环延迟对于算法的可伸缩性至关重要。在 BDS 中，此反馈循环包含几个过程：状态从代理更新到控制器、运行集中式算法以及从控制器更新到代理的决策更新。我们测量谁 Ie 进程的延迟，如图 11c 的 CDF 所示，并发现在大多数情况下（超过 80%），反馈循环延迟低于 200*ms*。因此，我们声称 BDS 演示了足够短的延迟，并且能够扩展到更大的系统。

*6.3.2 容错。*在这里，我们检查以下故障方案对每个周期下载的块数的影响。在循环 0 到 9 期间，BDS 照常工作，一个代理在第 10 个周期中发生故障。控制器在第 20 个周期中发生故障，并在第 30 周期中恢复。图 12a 显示了每个周期下载块的平均数量。我们发现代理故障的轻微影响只持续一个周期，系统在第 11 个周期中恢复。当控制器不可用时，BDS 会回回默认的 dec 分段叠加协议，从而导致性能正常下降。随着控制器的恢复，性能在第 30 周期恢复。



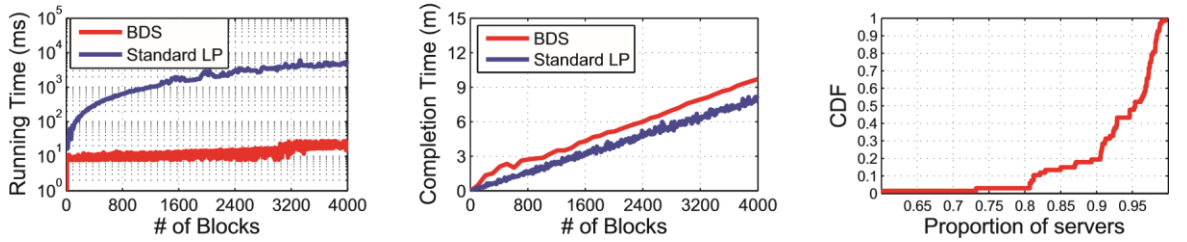
(a) The controller running time. (b) The inter-DC network delay. (c) Feedback loop delay.

图 11：[系统可伸缩性] 测量 （a） 控制器运行时间，（b） 网络延迟，（c） 反馈环路延迟。



（a） 在故障下每个周期的不同区块下平均下载次数 （b） 不同 （c）不同区块下的完成时间。

图 12：BDS 的（a）容错，（b）对不同块尺寸的敏感性，以及（c）不同的周期长度。



（a）将运行算法的减少 （b） BDS 的接近最佳程度为 （c） BDS 的块下载时间比标准 LP. 标准 LP 小比例。

图 13：关于（a） 算法运行时间减少，（b） 接近最佳，（c） 叠加效果的 [深入分析]
传输。

6.3.3 选择关键参数的值。块大小：在 BDS 中，大容量数据文件被拆分为块，可以在瓶颈不相交的路径上传输。但这引入了调度效率和计算开销之间的权衡。因此，我们使用不同的块大小（2MB 和 64MB）进行两个系列的实验。图 12b 显示 2MB/块方案中的完成时间比 64MB/块方案中的完成时间短 1.5 到 2 倍。但是，此优化引入了更长的控制器运行时间，如图 11a 所示。我们通过平衡两个注意事项来调整块大小：（1） 完成时间的限制，以及 （2） 控制器的操作开销。

更新周期长度：由于网络环境的任何更改都可能更改最佳的叠加路由决策，BDS 会通过定期调整路由方案来响应不断变化的网络条件。为了测试调整频率，我们设置了与 0 不同的周期长度。对于相同的 *s* 批量数据传输，5s 到 95*s*，图 12c 显示了完成时间。Smaller 周期长度导致完成时间更短，但当周期长度小于 3 s 时，好处会*减少*。这是因为更新过于频繁会引入更大的开销：（1） 从代理到控制器的信息收集，（2） t 他执行集中式算法，（3） 重新建立新的 TCP 连接。因此，考虑到调整粒度和相应的开销，我们最终选择 3s 作为默认周期长度。

6.3.4 深入分析。算法运行时间优化：BDS 分离调度和路由，可显著降低计算复杂性。为了清楚地显示优化，我们测量了 BDS 和标准 LP 解决方案下的算法运行时间。对于 standard LP 实验，我们使用 MATLAB [5] 上的 linprog 库，如果算法不收敛，则设置迭代数 （10^6） *linprog* 的上限，并记录 CPU 时间作为块数的函数。图 13a 显示 BDS 的运行时间低于 25*ms*，而标准 LP 的运行时间在只有 4000 *个块*时快速增长到 4 s。BDS 比现成的 LP 解算器快得多。BDS 的近乎最佳性：为了测量近乎最佳性，我们评估标准 LP 和 BDS 下的数据传输完成时间：2 个 DB、4 台服务器、20MB 用于服务器上传/下载速率。我们将块数从 1 到 4000，LP 解算器无法在合理的时间内完成。图 13b 显示了 BDS 的接近最佳性。不相交叠加路径的好处：

§2.2 揭示了应用程序级覆盖网络上不相交路径的好处。为了探索潜在的 benefit，我们记录从原始源下载的块数与块总数的比率，CDF 如图 13c 所示。对于大约 90% 的服务器，该比例不到 20%，这意味着超过 80%的块在不相交的路径上与其他 DC 进行 wnload，这显示了多播覆盖网络的巨大潜力。

总之，原型试验部署和 BDS 的跟踪驱动仿真都显示比现有解决方案快 3-5 + ，具有× 良好的性能和可靠性，以及近乎最佳的调度结果。

7 相关工作

在这里，我们将讨论一些与 BDS 相关的代表性工作，分为三类。

叠加网络控制。叠加网络为各种应用释放出巨大潜力，尤其是数据传输应用。代表性网络包括对等 （P2P） 网络和内容交付网络
（CDN）。P2P 架构已经通过许多应用程序验证，例如实时流式处理系统（酷流 [55]、Joost [2]、PPStream [4]、UUSee [6]）、视频点播 （VoD） 应用程序 （OceanStore [3]）、分布式哈希表 [42]以及最近比特币 [16]。但是，基于 P2P 原理的自组织系统具有长时间的收敛性。CDN 分布式服务相对于最终用户在空间上提供高可用性和性能（例如，减少页面加载时间），为许多应用程序提供服务，如多媒体 [59] 和实时流 [47]。

我们简要介绍了评估部分中的两个基线：（1） 项目符号 [26]，它使地理分布节点能够自行组织成叠加网格。具体地说，每个节点使用 RanSub [43] 将汇总票证信息分发到其他节点，并接收来自其发送对等体的不相交数据。BDS 和 Bullet 之间的 mai n 区别在于控制方案，即 BDS 是具有数据传输状态全局视图的集中式方法，而 Bullet 是分散式方案，每个节点在本地做出决策。（2） Akamai 设计了一个 3 层叠加网络，用于提供实时流[9]，其中源将其流转发到反射器，反射器将传出流发送到舞台接收器。Akamai 和 BDS 之间有两个主要区别。首先，Akamai 采用 3 层拓扑，其中 edge 服务器从其父反射器接收数据，而 BDS 则通过更细粒度分配成功探索更大的搜索空间，而不受三个粗粒度层的限制。其次，在 Akamai 中，数据的接收顺序必须是连续的，因为它是为实时流式处理应用程序设计的。但是，BDS 中没有这样的要求，副作用是 BDS 必须决定将最佳传输顺序作为额外的工作。

数据传输和速率控制。DC 级 transport 协议的速率控制在数据传输中起着重要的作用。DCTCP [8]、PDQ [21]、CONGA [7]、DCQCN [60] 和 TIMELY [33] 都是经典协议，在传输效率上明显提高。一些拥塞控制协议，如基于信用的 ExpressPass [11] 和负载平衡协议，如爱马仕 [53]，可以通过改进速率控制进一步缩短流量完成时间。在此基础上，最近提出的 Numfabric [36] 和 Domino [46] 进一步探讨了 centralized TCP 在加快数据传输和提高直流吞吐量方面的潜力。对于某些扩展而言，共流调度 [12， 52] 与多播叠加调度在数据并行性方面有一些相似之处。但是，这项工作侧重于流级问题，而 BDS 是设计在应用程序级。

集中式交通工程。交通工程（TE） 长期以来一直是一个热门的研究课题，许多现有的研究[10、25、32、40、45、50、54]都说明了可伸缩性、异质性等的挑战，特别是在 DC 间层面。代表的 TE 系统包括谷歌的 B4[24]和微软的 SWAN[22]。B4 采用 SDN [30] 和 OpenFlow [31， 38] 来管理单个交换机，d 在路径上部署自定义策略。SWAN 是另一个在线流量工程平台，其软件驱动的 WAN 实现了高网络利用率。

总体而言，应用程序级多播覆盖网络对于数据传输 in DC 内部 WAN 至关重要。用户日志、搜索引擎索引和数据库等应用程序将从批量数据多播中获益匪浅。此外，这些优势与以前的广域网优化是正交的，进一步提高了直流间应用程序性能。

8 康克· 卢森

DC 间多播对于全球级在线服务提供商的性能至关重要，但之前专注于优化 WAN 性能的努力是不够的。本文介绍了 BDS，一种应用级多播覆盖网络，它极大地提高了直流间批量数据多播的性能。BDS 展示了完全集中的多播叠加网络的可行性和实际优势，该网络选择覆盖路径，以近乎最佳但有效的方式安排数据传输。BDS 集中式设计的关键见解是，基于全球视图做出稍微延迟的决策可以带来显著的好处，这超过了集中化的成本。我们相信，加快集中算法的发布调度和路由的洞察力可以推广，以激发其他集中式控制平台，其中集中决策在成本和收益之间取得良好的平衡。

承认Ts

这项工作部分得到香港 RGC ECS 的支持：
26200014、CRF-C703615G、科大 PDF 基金、中国 973 计划（2014CB340300）、中国国家自然基金会（61472212）、欧盟玛丽居里行动皇冠（ FP7-PEOPLE-2013-IRSES-610524）。我们要感谢我们的牧羊人保罗·罗曼诺和匿名的 EuroSys 评论者的宝贵反馈。

引用

[1] 转到编程语言。https://golang.org.
[2] 乔斯特 http://www.joost.com/.
[3] 海洋商店。http://oceanstore.cs.berkeley.edu/.

[4] Ppstream 。http://www.ppstream.com/。
[5] 解 决 线 性 编 程 问 题 - matlab linprog 。 https://cn.mathworks.com/help/optim/ug/linprog.html?s _ 蒂特尔。[6] 乌塞。http://www.uusee.com/.

[7] Alizadeh， M.， Edsall， T.， Dharmapurikar， S.， Vaidyanathan， R.， Chu， K.， F ingerhut， A.， Lam， V. T.， Matus， F.， PanAN， R.和 Yadav， N. CONGA： 数据中心分布式拥塞感知负载平衡。在 *ACM SIGCOMM* （2014）， 第 503~514 页。

[8] Alizadeh， M.， Greenberg， A.， Maltz， D. A.， Padhye， J.， Patel， P.， Prabhakar， B.， Sengupta， S. S. SridharanS.在 *ACM SIGCOMM* （2010）， 第 63~74 页。

[9] 恩德雷耶夫， K.， Maggs， B. M.， MEYerson， A. 和 Sitaraman， R. K. 设计用于流的叠加多播网络。*SPAA* （2013）， 149~158.

[10] C 母鸡， Y.，一个 lspaugh， S. 和 Katz， R. H. 从不同的生产工作负载中设计 MapReduce 的见解。 技术代表， 加州大学伯克利分校电气工程和计算机科学系，2012 年。

[11] Cho， I.， Jang， K. H.and 和 HAn， D. 信用计划延迟绑定拥塞控制 for 数据中心。在 *ACM SIGCOMM* （2017）， 第 239~252 页。

[12] C 豪杜里， M 奥沙拉夫 Stoica，和 and Eecs，I. 协调流： 集群网络的应用程序层抽象。*热点*（2012）。

[13] Chu， Y.- HH.， Rao， S. G.和 Zhang， H .端系统多播的案例。在 *ACM SIGMETRICS* （2000）， 第 28 卷， ACM， 第 1~12 页.

[14] Cohen， B. 激励措施在位方面建立稳健性.*Proc P 经济学研讨会*（2003 年），1+1。

[15] Datta， A. K. 和 Sen， R. K. 1 近似算法，用于瓶颈 disjoint 路径匹配。*信息处理信件 55*， 1 （1995）， 41~44。

[16] Eyal， 即 Gencer， A. E.， Sirer， e. G.and 和 V Ar Renesse， R. 比特币-NG：可扩展的区块链协议。在 *NSDI* （2016） 中。

[17] Fleischer， L. K.近似于商品数量的零分多商品流。*SIDMA*（2000）， 505-520.[18] Garg， N. 和 and Koenemann， J. 多商品流和其他分数包装问题的更快、更简单的算法。*SIAM 计算杂志 37*， 2 （2007）， 630~652.

[19] Garg， N.， V 阿齐拉尼， V.五 and 、Y 安纳卡基斯，M.原始近似算法，用于树中积分流和多切。 *算法 18*， 1 （1997）， 3~20。

[20] Gog， I.， Schwarzkopf， M.， G 离开， A.， Watson， R. N. M. 和 Hand， S. 公司：快速、集中的集群大规模调度。在 *OSDI*（萨凡纳，佐治亚州，2016 年），USENIX 协会，第 99~115 页。

[21] Hong， C. Y.， Caesar， M.and 和 Godfrey， P. B.以先发制人的调度快速完成。"

[22] anduri WattenhoferHong， C.-Y.， Kandula，S.， Mahajan， R.， Zhang， M.， Gill， V.在 *ACM SIGCOMM*（*2013* 年），第 15~26 页。

[23] Huang， T. Y.， Johari， R.， Mckeown， N.， Trunnell， M.和 and Watson， M.基于缓冲区的速率适应方法：来自大型视频流服务的证据。 *SIGCOMM* （2014）， 187-198.

[24] Jain， S.， Kumar，A.， Mandal， S.， Ong， et alJ.， Poutievski， L.， Singh， A.， Venkata，S.， Wanderer，J.， Zhou，J.， Zhu，M.， B4：使用全球部署的软件定义的 WAN 体验。在 *ACM SIGCOMM*（*2013* 年），第 43 卷，第 3~14 页。

[25] Kavulya， S.， Tan， J.， Gandhi， R.， 和 and N 阿拉西姆汉， P.从生产图中分析轨迹，可以分析 cluster。在 *CCGrid* （2010）， IEEE，第 94~103 页。

[26] Kostic=， D.， Rodriguez， A.， A.lbrecht，J.和 Vahdat，A. 子弹：使用叠加网格高带宽数据传播.在 *ACM* SOSP（2003 年）中，第 37 卷，ACM，第 282~297 页。

[27] K et alumar，A.， Jain，S.， Naik，U.， Raghuraman，A.， Kasinadhuni，N.， Zermeno，E. C.， Gunn，C. S.， Ai，J.， Carlin，B.， Amarandei-stavila，M. -SBwE：用于 WAN 分布式计算的灵活分层带宽分配。在 *ACM SIGCOMM* （2015）， 第 1~14 页。

[28] Lamport，L.兼职的帕利亚门特。*ACM TOCS 16*， 2 （1998），133~169.

[29] Liebeherr， J.， Nahas， M.and 和 Si， W. 应用层多播与德劳奈三角测量叠加.*IEEE JSAC 200*， 8 （2002）， 1472~1488.

[30] McKeown， N. 软件定义的网络.*INFOCOM 主题演讲*17，2（2009），30~32。 30–32.

[31] McKeown， N.， Anderson， T.， B 阿拉克里希南， H.， Parulkar，G.， Peterson， L.， Rexford， J.， Shenker， S.和 Turner， J. Openflow： 支持校园网络创新。*ACM SIGCOMM 38*， 2 （2008），69~74。

[32] Mishra， A. K.， Hellerstein， J. L.， Cirne， W. 和 and Das， C. R. 实现云后端工作负载的特征：来自 google 计算群集的见解。*ACM SIGMETRICS 每 37*， 4 （2010）， 34×41.

[33] Mittal， R.， Lam，V. T.， D UKKIPATIukkipati， N.， Blem， E.， Wassel， H.，
G 霍巴迪， M.， Vahdat， A.， Wang， Y.， WEtherall， D. 和 Zats， D. 及时：数据中心基于 RTT 的拥塞控制。在 *ACM SIGCOMM*（2015）， 第 537-550 页。

[34] Mokhtarian， K. 和 and Jacobsen， H. A. 网格覆盖的最小延迟多播算法。*IEEE/ACM 吨 23*， 3 （2015）， 973~986.

[35] aylor，D.，Han，D.，Seshan ZhangMukerjee， M. K.， Hong， J.， J.iang，J.，N 在 *ACM SIGCOMM* （2014）， 第 44 卷， ACM， 第 343~344 页.

[36] Nagaraj， K.， Bharadia，D.， Mao， H.， Chinchali， S.， Alizadeh， M.， 和 Katti，S. Numfabric：数据中心的快速灵活的带宽分配。在 *ACM SIGCOMM* （2016）， 第 188~201 页。

[37] Nygren， E.， Sitaraman， R. K. 和 and Sun， J.*Akamai 网络：高性能互联网应用的平台*。ACM， 2010.

[38] O 笔 F 低。开流规范。http://archive.openflow.org/wp/文档。

[39] Reed， M. J. 信息中心网络的交通工程.在 *IEEE ICC* （2012）， 第 2660~2665 页。

[40] Reiss， C.， T UMANOVumanov， A.， Ganger，G. R.， Katz， R. H.， 和 Kozuch， M. A.异质性，大规模云的动态性：谷歌跟踪分析。在 *SoCC* （2012）， ACM， 第 7.

[41] Repantis， T.， Smith， S.， S.mith 和 and Wein， J. 扩展 akamai 网络的监视基础结构。*Acm Sigops 操作系统评论 44*， 3 （2010），20–26.

[42] Rhea， henkertoica YS.， Godfrey， B.， Karp， B.， Kubiatowicz， J.， Ratnasamy， , SS.， S.S.， I.和 Yu， H. Opendht： 公共 dht 服务及其用途.在 *ACM SIGCOMM（2005* 年），第 35 卷，第 73~84 页。

[43] hirud Vahdat, A. Using random subsetRodriguez， A.， A.， a., J., B 在《*美国国际经济*国际服务系统》（2003 年）中，第 19-19 页。

[44] Savage， S.， Collins， A.， Hoffman， E.， Snell， J.和 and 恩德森，T.互联网路径选择的端到端效果。在 *ACM SIGCOMM（1999* 年）中，第 29 卷，第 289~299 页。

[45] Sharma， B.， Chudnovsky， V.， Hellerstein， J. L.， Rifaat， R. 和 Das， C. R. 建模和综合谷歌计算集群中的任务放置约束。在 *SoCC* （2011）， ACM， 第 3.

[46] Sivaraman， A.， Cheung， A.， Budiu， M.， Kim， C.， Alizadeh， M.， B 阿拉克里希南， H.， Varghese， G.， McKEown， N. and l icking， S. 数据包事务：线路速率交换机的高级别编程。在 *ACM SIGCOMM* （2016）， 第 15~28 页。

[47] S 里帕尼库尔柴， K.， M Aggs，B. 和 Z 挂， H.互联网实时流工作负载分析。在 *IMC* （2004）， ACM， 第 41-54 页.

[48] Wang， F.， Xiong， Y. 和 and Liu， J. mTreebone： 用于应用层实时视频多播的混合树/网格覆盖。在 *ICDCS（2007* 年），第 49。

[49] Wang， H.， Li， T.， Shea， R.， Ma， X.， Wang， F.， Liu， J.and 和 Xu， K.*IEEE/ACM 吨* （2017）。

[50] W 伊尔克斯， J. 更多谷歌集群数据。 http://googleresearch.blogspot.com/2011/11/， 2011.

[51] Zhang， H.， Chen， K.， Bai， W.， Han， D.， Tian， C.， Wang， H.， Guan， H.and 和 Zhang， M. 保证数据中心传输的最后期限。在 *EuroSys* （2015）， ACM， 第 20.

[52] Zhang， H.， Chen， L.， Yi， B.， Chen， K.， Geng， Y. 和 and Geng， Y. CODA： 在黑暗中自动识别和调度共流。在 *ACM SIGCOMM* （2016）， 第 160~173 页。

[53] Zhang， H.， Zhang， J.， Bai， W.， Chen， K. 和 and ChowDhury， M. 弹性数据中心负载巴尔在野外跳舞。在 *ACM SIGCOMM* （2017）， 第 253~266 页。

[54] Zhang， Q.， Hellerstein， J. L.和 and Boutaba， R. 在谷歌的计算集群中描述任务使用形状。在*拉迪斯*（2011 年）。

[55] Zhang， X.， Liu， J.， Li， B. 和 and Yum， Y.-S.酷流/DoNet：用于点对点实时媒体流的数据-dr iven 覆盖网络。在 *INFOCOM*（2005）， 第 3 卷， IEEE， 第 2102-2111 页。

[56] Z 挂， Y.， Li， Y.， Xu，英国， Wang， D.， Li， M.， Cao， X.和 Liang, Q.一种通信感知容器重新分发方法，用于高性能 vnfs。在 *IEEE ICDCS 2017* （2017）， IEEE， 第 1555~1564 页.

[57] Zhang， Y.， Xu， K.， Wang， H.， Li， Q.， Li， T.， 和 Cao， X. 快速公平：基于云的 Serv 冰链的延迟优化。*IEEE 网络*（2017）。

[58] Zhang， Y.， Xu， K .， Yao， G .， Zhang， M. 和 and Nie， X. Piebridge： 跨 dr 大规模大型数据传输调度系统 .在 *ACM SIGCOMM* （2016）， ACM， 第 553~554 页.

[59] Zhu， W.， Luo， C.， Wang， J.， 和 Li， S. 多媒体云竞争 uting .*IEEE 信号处理杂志 28*， 3 （2011）， 59~69 。

[60] Z 胡，Y.， Eran，H.， Firestone，D.， Guo，C.， L IPSHTEYNipshteyn，M.，
L 铁， Y.， Padhye， J.， Raindel， S.， Yahia， M. H. 和 and Zhang， M. 大规模 RDMA 部署的拥塞控制。*ACM SIGCOMM 45*， 5 （2015），523~536.

9 附录

假设我们要向 *m 目标 DC* 发送 N 个数据块。
在不丧失一般性的情况下，我们考虑两种情况：

- A（平衡）：每个 *N* 块都有 *k* 个重复;
- B（不平衡）：一半块各有 k_{k1} 个双面，另一半有 k_2 双面，k $k_1 < k_2$，（kk_1+ k_2）/2 = k。

请注意，$m > k$，否则，多播已经完成。接下来，我们证明在简化的设置中，BDS 在 A 中的完成时间严格小于 B。

为了计算 BDS 的完成时间，我们现在做出一些假设（这对于我们的结论来说不是关键）：（1） 所有服务器具有相同的上传（重新下载）带宽 *R* 向上（resp.$R_{向下}$），（2） 没有两个双面共享同一个源 （resp. destination）服务器，因此每个块的上传（resp. 下载）带宽为 R_{up}（resp.$R_{向下}$）。现在，我们可以在以下两种情况下编写完成时间：

$$t_A = \frac{V}{min\{c(l), \frac{kR_{up}}{m-k}, \frac{kR_{down}}{m-k}\}}$$

$$t_B = \frac{五}{min\{c(l), \frac{k_1R_{up}}{m-k_1}, \frac{k_2R_{up}}{m-k_2}, \frac{k_1R_{down}}{m-k_1}, \frac{k_2R_{down}}{m-k_2}\}} \quad (6)$$

其中 V 表示未传输块的总大小，
$V = N(m-k)\rho(b) = \frac{N}{2}(m-k_1)\rho(b) + \frac{N}{2}(m-k_2)\rho(b)$. 在百度的生产系统中，DC 间链路容量) c（I）*I*) 比单个服务器的上传/下载容量高几个数量级，因此我们可以安全地从*方程中的*分母中排除 *cc*（I）。*I* 最后，如果我们表示*最小*=*R*and *t 向上*，*RR 向下*= *R*，则 t_A = （m={kRk）V 和 t_B = m（mm=k_1kR_1）V 。

我们可以显示 that （m=kRk）V 是 k 的单调递减*函数*：

$$\frac{d}{dk}\frac{(m-k)Vd}{kR} = \frac{(m--k)^2N\rho(b)}{dk\ kR} = \frac{否\rho(b)}{R}\frac{m^2}{(1=\ k_2)} < 0 \quad (7)$$

现在，由于 $k > k_1$. 我们有 $t_A < t_B$。