Skyplane: Optimizing Transfer Cost and Throughput Using Cloud-Aware Overlays

摘要

Skyplane，是一种用于优化云和云之间的大规模数据传输的系统，将overlay网络引入公共云环境，实现了应用层的跨区域和跨云数据传输。这是将overlay网络的思想成功扩展到云计算环境的创新应用。它能提高传输吞吐量，同时考虑价格和性能之间的权衡。该系统的规划器使用线性规划来确定数据传输的最佳覆盖路径和资源分配。在一个云内，Skyplane的性能比公共云传输服务高出最多4.6倍，在云之间的性能提高最多5倍。总之，论文在overlay网络、网络优化、资源管理等多个方面进行了创新，成功将overlay网络引入云环境，解决了跨云大数据传输的关键难题。

关键词：云感知覆盖网络；成本与吞吐量权衡；混合整数线性规划

1 引言

随着云计算的发展，越来越多的应用开始在多个区域和多个云提供商之间传输大量的数据，比如为了遵守隐私法规、利用特定硬件或者避免被供应商“锁定”在其服务上。但是跨区域和跨云之间的数据传输往往比较缓慢，已经成为这些应用的性能瓶颈。

文章所要解决的问题就是如何优化云环境下大批量数据传输的成本和吞吐量。文章指出直接在网络层优化路由协议可能会对其他类型应用产生负面影响，而云提供商也没有足够的动机来优化到其他云提供商的数据传输。作者提出了一个应用层的解决方案Skyplane，来通过覆盖网络实现跨区域跨云环境下的数据高吞吐低延迟传输。

如下图所示，Skyplane利用云感知的网络覆盖层在应用层上进行价格和性能之间的权衡，通过间接路径传输数据，而不是依赖直接路径。这种方法可以提高数据传输的吞吐量和效率。其规划器使用混合整数线性规划来确定最佳的覆盖路径和资源分配，以满足用户对价格或性能的需求。通过测量云中数据传输的成本，Skyplane能以最少的额外成本减少数据的传输时间。其性能在同一云服务提供商中比官方提供的云传输服务高出最多4.6倍，在不同云服务之间的性能提高最多5倍。Skyplane 的成本和吞吐量都优于 RON，而其找到的覆盖路径的吞吐量比直接路径高出 4.7 倍，但成本开销仅为 14%。

该论文的研究问题具有实际应用价值。跨区域和跨云大数据传输是云计算中一个普遍存在的痛点，使用覆盖网络进行优化可以提供很好的解决方案，因此具有较高的应用价值。论文内容详实且包含开源代码。论文从系统与算法设计、部署测试等方面进行了详细论述，进行复现有足够理论依据和技术参考，开源代码为复现工作提供了极大的便利。

此外，复现本文对自身研究工作有帮助。该论文结合了网络、优化算法、云计算等多方面知识，与我的研究方向较为契合，还能在此过程中加强对overlay网络的理解，对提高自身研究能

力有一定的帮助。

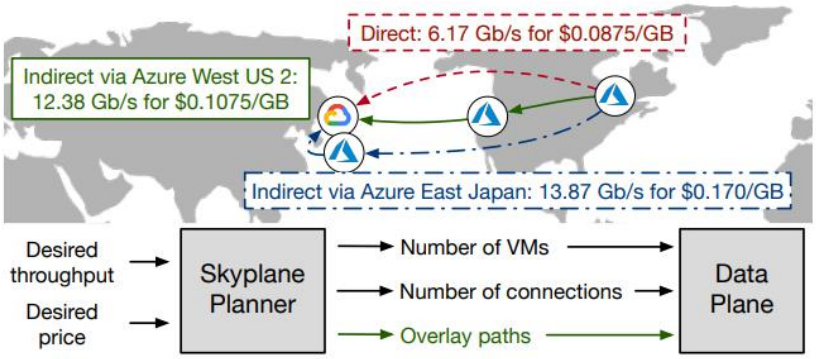


图 1　Skyplane 工作示意图

2　相关工作

Overlay网络作为当今互联网领域的一项前沿技术，扮演着连接和增强网络的关键角色。这种网络结构通过在底层网络之上创建一个覆盖层，提供了更高层次的抽象和灵活性，为各种应用和服务的开发创造了便利条件。然而，随着互联网的不断发展，Overlay网络的研究方向也愈发多元，涉及诸多关键问题。以下将对Overlay网络研究的主要方向进行概要分类和描述。

2.1　路由与拓扑控制

Overlay网络中的路由与拓扑控制是关键的研究领域之一。在这方面，学者们关注如何设计更加智能、高效的路由算法，以确保数据能够快速、可靠地在Overlay网络中传输。这包括基于节点状态的拓扑构建方法，如何在网络拓扑发生变化时动态更新路由表等。对于大规模Overlay网络，通常采用分层和分布式的路由协议，以降低复杂性。

2.2　网络安全与隐私

Overlay网络的安全性和隐私保护是互联网研究领域的热点之一。研究者们致力于开发防御措施，以应对各种网络攻击，包括拒绝服务攻击、数据篡改、信息泄漏等。在这个方向上的工作还包括加密技术、身份认证方法，以及匿名性保护等，以确保Overlay网络的安全可靠。

2.3　分布式系统与协同计算

Overlay网络广泛应用于分布式系统和协同计算环境。在这一领域，研究者们关注Overlay网络在任务调度、资源管理、数据共享等方面的应用。如何在分布式环境中高效协同工作，提高整体系统的性能，是这个方向上研究的核心问题。

2.4　改善网络服务质量

Overlay网络在不同应用场景中具有广泛应用，包括P2P文件共享、视频流媒体、内容分发网

络等。在这方面，研究者关注如何定制Overlay网络以适应特定应用的需求。同时，为了提供更好的用户体验，研究者还致力于研究如何保障服务质量（QoS），确保网络能够满足各种应用的性能需求。

3 本文方法

3.1 云网络分析

使用 iperf3 测量吞吐量:作者在每个区域对之间建立 TCP 连接，使用 iperf3 工具发送数据，测量网络的实际吞吐量。此过程测试了 20 个 AWS 区域、24 个 Azure 区域和 27 个 GCP 区域之间的连接，总共生成了一个 5184 条链路的完整吞吐量矩阵。测量吞吐量耗费了大约 4000 美元的网络流出费用。

下图分析了基于测量结果的 GCP 和 Azure 网络延迟与吞吐量之间的关系。可以看出，无论 GCP 还是 Azure，云内部同区域之间的链路(intra-cloud routes)RTT 更低，代表距离更近;而跨云之间的链路(inter-cloud routes)RTT 更高。与 RTT 对应地，跨云链路的吞吐量也比较低，难以达到云内链路的水平。这是因为不同的云提供商对出口带宽有限制，例如 GCP 对所有出口流量限制为 7 Gbps，AWS 对单个实例的出口限制为 5 Gbps。这些结果验证了跨云链路确实是影响性能的瓶颈。

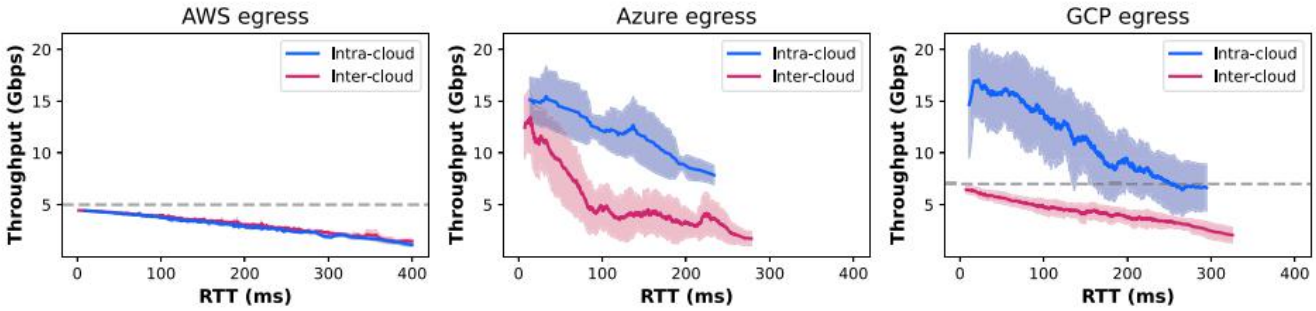


图 2 网络测量结果图

3.2 求解最佳方案

使用现成的线性规划(LP)求解器找到最优计划。作者将数据传输规划问题形式化为一个混合整数线性规划模型(MILP)，其中包括overlay路径、TCP连接数目、每个区域的VM数目等作为变量，并有吞吐量与价格的约束条件。然后使用成熟的商业MILP求解器Gurobi对其进行求解，仅需不到5秒就可以找到最优解。

为进一步加快求解，可以将MILP问题放宽为连续LP问题，允许变量取连续值。这个连续LP问题可以在多项式时间内用内点法算法求解，比MILP的求解时间更短。最后将LP解中连续的变量取整，效果与MILP解差异在1%以内。将规划问题建模为LP，是Skyplane的关键创新。这使得

它可以整体考虑路由、价格与资源配置，并快速获取最优数据传输方案。现成的LP求解器为此提供了支持。

作者使用流网络来表示叠加网络拓扑的思想，其中节点表示区域，边表示链路。文中提出一个如下图所示最小成本流量问题的线性规划来求解这个流网络的最优流量分配方案。

关于约束条件，文中列出了多个可以由用户自定义的限制，包括:

目标吞吐量:用户期望的数据传输吞吐量。

吞吐量网格限制:每个链路的最大容量。

TCP连接限制:每个实例最大TCP连接数。

虚拟机限制:每个区域最大实例数量。

出口带宽限制:每个实例的出口带宽上限。

此外，作者还考虑了成本信息，包括每个区域的网络出口费用和实例费用。

综合考虑所有的约束条件，构建一个线性规划模型来最小化成本的同时满足用户指定的吞吐量要求。

$$\begin{aligned} \arg\min_{\mathrm{F}} \quad & \langle C, \mathrm{F} \rangle \\ \text{subject to} \quad & \textstyle\sum_{(c,v)\in E} \mathrm{F}_{c,v} \geq \text{TPUT GOAL} \\ & \textstyle\sum_{(u,v)\in E} \mathrm{F}_{u,v} = \sum_{v,w} \mathrm{F}_{v,w} \quad \forall v \in V - \{s,t\} \\ & 0 \leq \mathrm{F} \leq \text{LIMIT}^{link} \end{aligned} \tag{1}$$

where s and t are the source and destination regions, $\text{LIMIT}^{link} \in \mathbb{R}_+^{|V|\times|V|}$ is the maximum capacity for each link and $C \in \mathbb{R}_+^{|V|\times|V|}$ is the cost per unit of bandwidth between regions. We use the same notation for matrix and vector inner products: $\langle C, \mathrm{F} \rangle = \sum_{u,v} C_{u,v} \mathrm{F}_{u,v}$.

Variables	
$\mathrm{F} \in \mathbb{R}_+^{\lvert V\rvert\times\lvert V\rvert}$	*Throughput grid*
$\mathrm{N} \in \mathbb{Z}_+^{\lvert V\rvert}$	*VMs per region*
$\mathrm{M} \in \mathbb{Z}_+^{\lvert V\rvert\times\lvert V\rvert}$	*TCP conn. per region*
Constraint: goal throughput	
$\text{TPUT GOAL} \in \mathbb{R}_+^{\lvert V\rvert\times\lvert V\rvert}$	*User's desired throughput*
Constants: provider limit	
$\text{LIMIT}^{link} \in \mathbb{R}_+^{\lvert V\rvert\times\lvert V\rvert}$	*Throughput grid limit*
$\text{LIMIT}^{conn} \in \mathbb{Z}_+^{\lvert V\rvert\times\lvert V\rvert}$	*TCP connection limit*
$\text{LIMIT}^{ingress} \in \mathbb{Z}_+^{\lvert V\rvert}$	*VM limit*
$\text{LIMIT}^{egress} \in \mathbb{Z}_+^{\lvert V\rvert}$	*Egress bandwidth limit*
Constants: provider cost	
$\text{COST}^{egress} \in \mathbb{R}_+^{\lvert V\rvert}$	*Egress cost ($/Gbit)*
$\text{COST}^{VM} \in \mathbb{R}_+^{\lvert V\rvert}$	*VM cost ($/s)*

Table 1: Symbol table for Skyplane's ILP formulation.

图 3　最佳方案求解方案图

3.3　计算传输成本

最小成本流量无法准确反映云环境下的数据传输成本。在Skyplane中传输的总成本包括两个部分:

1)出口成本:按出口流量计费，与带宽无关。可以通过单位时间内的流量乘以传输时间，求出总出口成本。

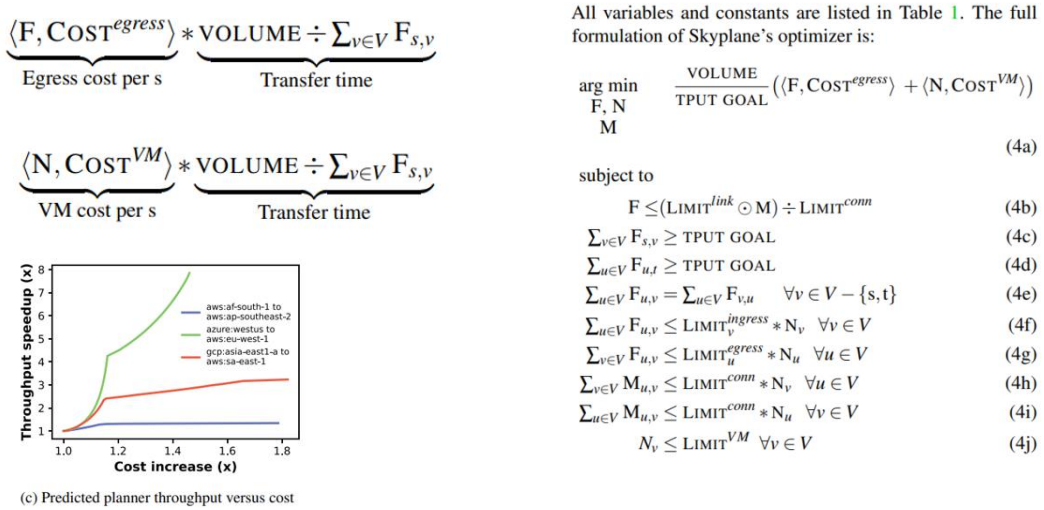
2)虚拟机成本:这个决策变量表示每个区域使用的虚拟机实例数量。随着实例数量的增加，总带宽会线性提升。所以实例的成本也需要考虑在内。

文中还讨论了使用并行TCP连接可以提高吞吐量，但并不是线性关系。为控制成本，文中将TCP连接数作为一个决策变量，受到实例数和连接数限制的约束。

对于最大化吞吐量的问题，文中通过枚举多个吞吐目标求出对应的最小成本解，形成帕累托

前沿曲线，从中选择最优解。

以上内容丰富了传输成本的计算，将其分为出口成本和实例成本两部分。同时也引入了TCP连接和实例数作为决策变量，受到相应服务限制的约束，扩大了线性规划模型表达的范围，以提供更优解。



(c) Predicted planner throughput versus cost

图 4　传输成本计算方案图

4　复现细节

4.1　与已有源代码对比

该项目是开源的，但在云服务商之间传递数据需要较多的云计算理论知识储备和得到相关服务商的支持，这一流程是繁杂且耗时的，所以本文着重于复现作者的实验结果和调整现有的参数以观察相关参数的调整会对实验结果产生何种影响，具体内容详见4.6节。

4.2　安装及配置Skyplane

首先安装 Skyplane，使用这行命令即可：$ pip install "skyplane[aws]"，之后要进行云凭据的设置。在安装完云提供商的 CLI 工具后，使用相应的命令行进行凭据配置，这里以GCP为例 $ gcloud auth application-default login，按提示进行身份验证即可，完成这些设置即可进行传输作业，但要注意跨云传输需要进行额外的配置，Skyplane以及云服务商都提供了相关教程这里不再赘述。

此外，还可以使用配置命令调整 Skyplane ，Skyplane 配置选项总结包括命令行、传输并行度、网络、对象存储、回退到本机命令和实例配置六个方面，涵盖了各种设置。

4.3　单个云区域之间传输大文件

我比较了 Skyplane 和云数据传输工具（例如 AWS DataSync 和 Google Cloud Data Transfer）的性能。在两个 AWS 区域之间传输单个大文件。原文的结果为：总体而言，对于最

大的测试文件传输，Skyplane比 AWS DataSync快 113.4 倍。我获得的结果为118.1倍。获得的结果因测试区域、文件大小和使用的虚拟机数量而异。与原文结果（左图）相比，时间略长，分析结果得知有以下几种可能：两个服务器所处的区域与作者也可能存在不同；生成的随机文件虽然大小相同，但其内容等也可能影响传输速率；此外，所处的网络状况也会影响传输速率。

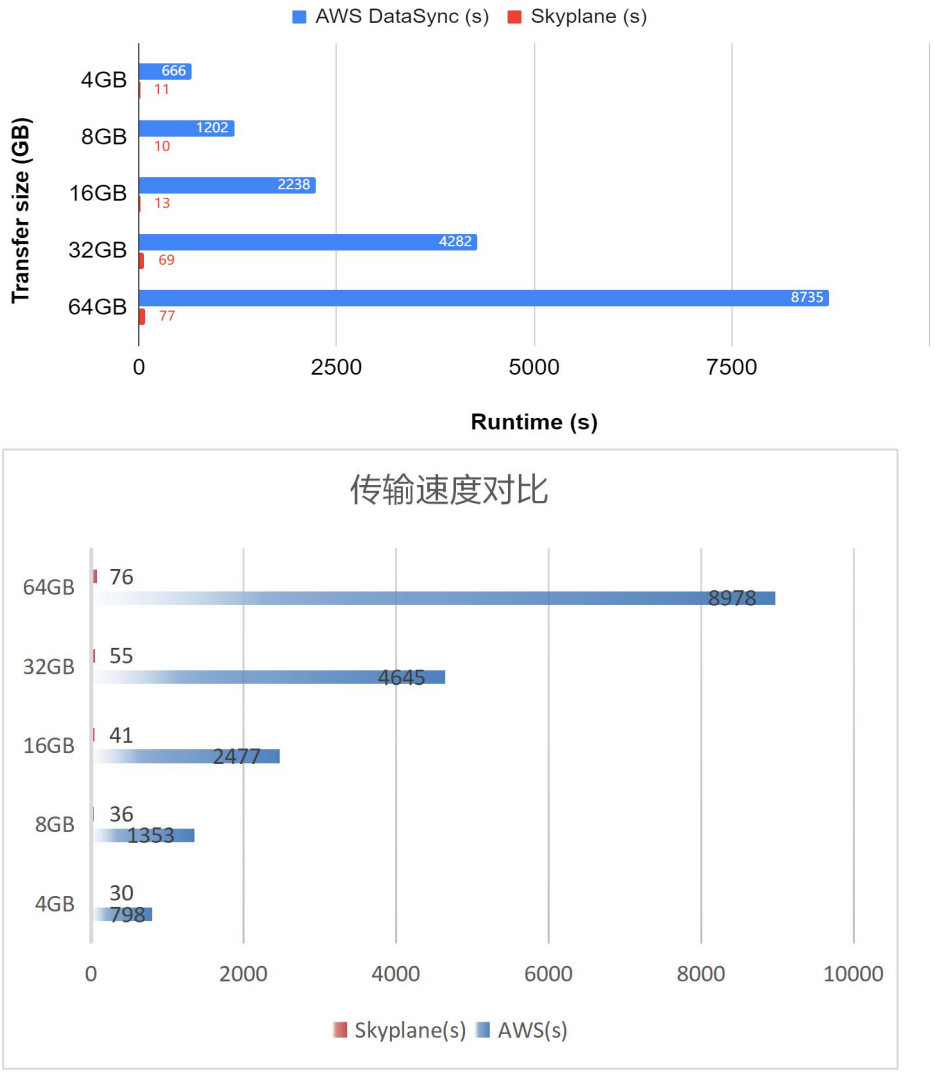


图 5　传输速度对比图

4.4　压缩成本比较

作者在这个基准测试中测量了 Skyplane 解压缩算法的影响，该算法节省了大量的数据出口费用。此外，Skyplane不收取任何服务费。总体而言，在传输未压缩的 220GB 英文维基百科转储时，Skyplane 比 AWS DataSync 便宜 6.2 倍。我还是使用随机生成的64GB文件进行传输，AWS DataSync成本为$1.781，Skyplane成本为$0.307，我的测试结果为Skyplane 比 AWS DataSync 便宜 5.8倍，至于rsync，我未进行测试。

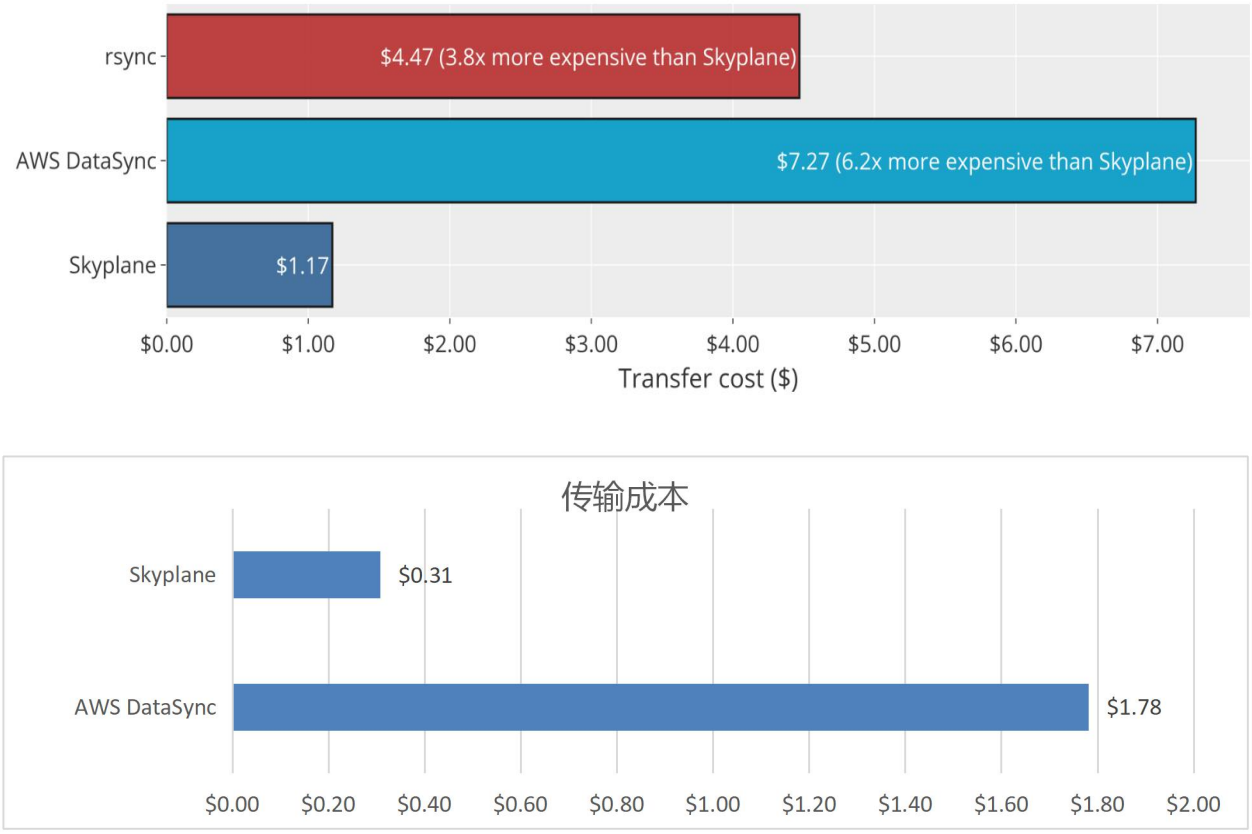


图 6　传输成本对比图

4.5　AWS 上的 ImageNet 传输

此测试中，作者测量了 Skyplane API 的传输速度和普遍支持。为了传输 70 GB 的假 imagenet，Skyplane 支持跨 AWS、GCP 和 Azure 传输。对于下面选定的传输区域对，它会在大约 25 秒内完成传输。但是，AWS DataSync 仅支持传入和传出 AWS 服务的数据传输，而且速度很慢。由于条件所限，我仅在AWS与GCP之间进行了数据传输测试。结果如右下图，可以看到结果和原文所述差别较小，较好的复现了原文实验。

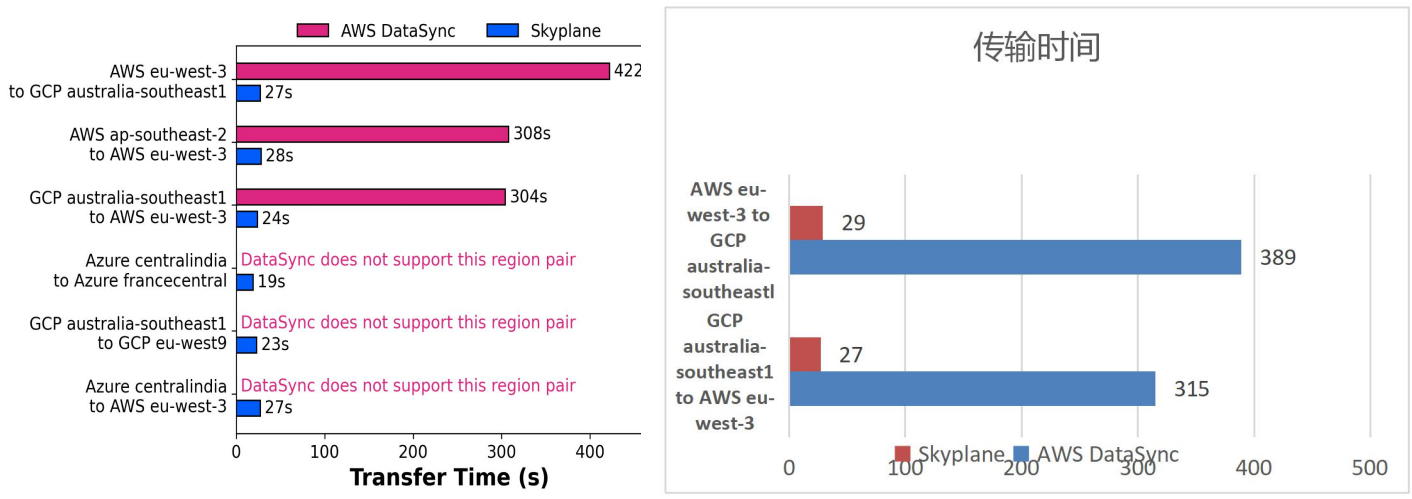


图 7　跨区域传输速度对比图

4.6 创新点

Skyplane利用并行虚拟机以高速率传输数据。默认情况下，每个SkyplaneVM使用32个vCPU，此数量的虚拟机可在AWS上提供高达5Gbps的网络吞吐量。于是我试验了每个区域使用8个虚拟机，即请求256个vCPU。我以作图的方式比较了区域VM数目（VCPU）增加给传输带来的速度增益。在AWSEC2控制台中，要增加Skyplane的vCPU配额，需进入相应区域，选择配额选项卡，搜索“运行按需标准实例”，选中并点击“请求增加配额”，在表单中指定所需的vCPU分配，等待AWS审核通过即可。

实验结果显示，增加的虚拟机（vCPU）使传输时间平均缩短了7倍左右。

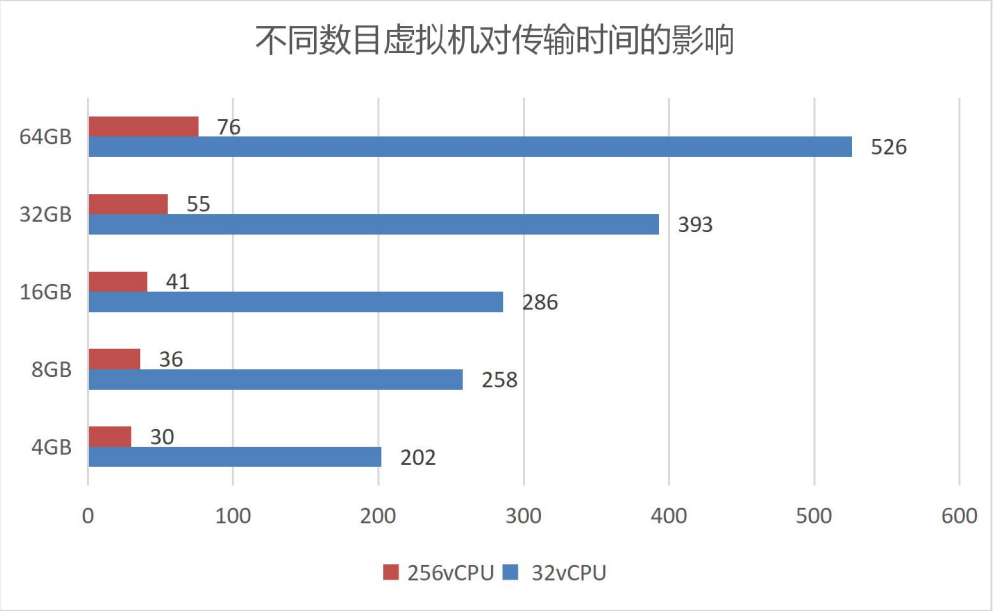


图 8 不同数目虚拟机条件下传输时间对比图

5 总结与展望

本文探讨了如何利用云感知覆盖网络在云区域之间高效传输数据。我们的主要观点是，可以将覆盖网络的原理应用到云环境中，以确定高质量的网络路径，从而实现快速传输。不过，将覆盖网络的原理应用于云环境需要考虑云资源的定价，尤其是与网络带宽相关的出口费用。在进行批量数据传输时，Skyplane可以在性能和成本之间进行权衡。它的工作原理是接受用户或应用程序提供的性能约束，并求解混合整数线性程序（MILP）以获得最佳数据传输计划。

与直接路径相比，Skyplane选择的最优覆盖路径可以实现2-5倍的吞吐量提升，并且不会导致成本大幅增长。在有些情况下还可以减少成本支出，且Skyplane支持不同云提供商之间的跨云数据传输。经过实验验证，Skyplane在不同区域、不同数据集下可以较为稳定的复现性能指标。我以作图的方式比较了传输速度、压缩成本、和不同数目虚拟机对传输速度的影响。该论文结合了网络、优化算法、云计算等多方面知识，与我的研究方向较为契合，在此过程中我加强了对overlay网络的理解，对提高自身研究能力有一定的帮助。

参考文献

[1] Jain P, Kumar S, Wooders S, et al. Skyplane: Optimizing Transfer Cost and Throughput Using {Cloud-Aware} Overlays[C]//20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). 2023: 1375-1389.

[2] Amazon Web Services. EC2 on-demand instance pricing. https://aws.amazon.com/ec2/pricing/on-demand/, 2022.

[3] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines[C]//Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. 2005: 273-286.

[4] Kumar A, Jain S, Naik U, et al. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. 2015: 1-14.

[5] Lagar-Cavilla H A, Whitney J A, Bryant R, et al. Snowflock: Virtual machine cloning as a first-class cloud primitive[J]. ACM Transactions on Computer Systems (TOCS), 2011, 29(1): 1-45.

[6] Lai F, Chowdhury M, Madhyastha H. To Relay or Not to Relay for {Inter-Cloud} Transfers?[C]//10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18). 2018.

[7] Zhang Y, Jiang J, Xu K, et al. BDS: A centralized near-optimal overlay network for inter-datacenter data replication[C]//Proceedings of the Thirteenth EuroSys Conference. 2018: 1-14.

[8] Tseng S H, Agarwal S, Agarwal R, et al. {CodedBulk}:{Inter-Datacenter} Bulk Transfers using Network Coding[C]//18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21). 2021: 15-28.