学士論文 2020年度 (令和2年度)

ShadowLB: A Transparently Accelerated Load Balancer

慶應義塾大学 環境情報学部
橘 直雪

学士論文要旨 - 2020 年度 (令和 2 年度)

ShadowLB: A Transparently Accelerated Load Balancer

近年，ロードバランサーはハードウェアアプライアンスのものからソフトウェアのものへ変遷しつつある．これにより，データセンターは物理スペースや経済的なコスト問題を解決した．これらの高速ソフトウェアロードバランサにはカーネルバイパス系パケット処理フレームワーク等を利用した高速データプレーン技術を用いたパケット処理性能の向上が図られているが，これらの高速データプレーン技術は既存のコントロールプレーンの API を利用することができず，開発者はデータプレーンよりもコントロールプレーンの開発にコストを掛けているのが実情である．

そこで本研究では，既存のコントロールプレーン機能を流用しつつ，データプレーンのみ透過的に高速化させる機構,shadowLB を設計，実装した．shadowLB は既存のコントロールプレーンのデファクトスタンダードである ipvs の API を用いながら，テールレイテンシにおいて X 倍の性能を評価にて示した．

キーワード:
1. Load Balancer，2. XDP，3. 負荷分散

慶應義塾大学 環境情報学部
橘 直雪

Abstract of Bachelor's Thesis - Academic Year 2020

ShadowLB: A Transparently Accelerated Load Balancer

In recent years,load balancers have been shifting from hardware appliances to software. In this way, data centers have solved the physical space and economic cost problems. These high-speed software load balancers are designed to improve packet processing performance by using high-speed data plane technology that utilizes kernel bypass packet processing frameworks, etc.

However, these high-speed data plane technologies cannot utilize existing control plane APIs, and developers are spending more resources on control plane development than on data plane.

In this study, we designed and implemented shadowLB, a mechanism that transparently accelerates only the data plane while reusing the existing control plane functions. We have shown that shadowLB can achieve X times performance in tail latency while using the ipvs API, which is the de facto standard for existing control planes.

Keywords :
1. Data center network, 2. Network operation, 3. IPv6 transition mechanism

Keio University Faculty of Environment and Information Studies
Naoyuki Tachibana

目 次

図 目 次

表 目 次

第1章　序論

本章では本研究の背景と全体の構成について記述する．

1.1　ロードバランサを取り巻く環境

1.1.1　大規模ネットワークサービスの台頭

スマートフォンやタブレットの普及，また Wi-Fi スポットの増加などに伴い，インターネットのトラフィックは近年急速に増加している [2]．それに伴い，インターネットサービスを展開する際には，大量のトラフィックを処理するために，ロードバランサーを設置して複数のサーバにトラフィックを分散するというアプローチが一般的に取られている．

1.1.2　ロードバランサーのソフトウェア化

従来，ロードバランサは F5 Networks[1], Cirtix[2]等を代表とするハードウェアアプライアンス製品が一般的であった．しかし，ハードウェアロードバランサは処理能力は優れているものの，以下のような課題を抱えていた．

- **物理スペースの圧迫**
  ルータ，スイッチ，サーバと同様，1U 以上の設置スペースを必要とする．
- **経済的コスト**
  データセンター向けネットワーク製品は高価であるとともに，メンテナンス費用，電力等様々なコストが発生する
- **冗長化が困難**
  物理アプライアンスであることから，ネットワークトポロジに組み込むまでに最短でも 1〜2 日は必要とする．そのため，急増するネットワークトラフィックに即座に対応できない．

このようなオペレーター，ユーザー双方にとってコストが高いハードウェアロードバランサーに代わり，Google[3]の Magrev[3] や Microsoft[4]の Ananta[4] といったように，大規模

[1] https://www.f5.com/ja_jp
[2] https://www.citrix.com/ja-jp/
[3] https://www.google.com/
[4] https://www.microsoft.com/ja-jp

なサービスを提供している企業はロードバランサを自社技術を用いてソフトウェア化し始めた．各企業の高速ロードバランサの詳細に関しては，2.2 章にて説明する．

1.1.3 データプレーン処理の高速化

1.1.2 章で例示したソフトウェアロードバランサは，ハードウェアロードバランサと比較して劣る単体の処理性能を，汎用サーバーとソフトウェアデータプレーンを複数構成し，スケールアウトすることによって補っている．また，単体の集積効率をできるだけ向上させるため，DPDK[5],Netmap[6] 等のカーネルバイパス技術を用いている．これにより，特定のネットワーク処理に特化しない Linux サーバにおいても，高速なパケット処理を実現している．高速データプレーン処理技術の詳細については，2.3 章にて説明する．

1.2 本研究の着目する課題

カーネルバイパスをはじめとする高速パケットフレームワークを用いてデータプレーンを高速化したソフトウェアロードバランサーは，高いパケット処理性能の一方で, いずれもコントロールプレーンに変更を加えなければならないという課題を持っている．例えば GLB や Katran は，独自の CLI 機構やデータプレーン API を実装している [7]．ここで問題になるのは，既存のソフトウェア資産を一切活用できないことである．運用する環境にも依存するが，ヘルスチェックエージェントや CLI，Kubernetes[5],OS といったオーケストレータの仕組みにも手を加える必要がある．その結果，データプレーンよりコントロールプレーンの開発コストが嵩んでいるのが実情である [8]

1.3 本研究の目的

本研究においては，ソフトウェアロードバランサーの開発にあたり既存のコントロールプレーンを流用しつつ，データプレーンのみを透過的に高速化できる仕組みを提案する．提案手法により，エンジニアはコントロールプレーンの開発について意識することなく，データプレーンの高速化に注力できるようになる．

1.4 本論文の貢献

本論文では，ソフトウェアロードバランサーのうち，Linux カーネルに取り込まれており，デファクトスタンダード的な立ち位置にある IPVS[9] のコントロールプレーンを利用しつつ，データプレーンを透過的に高速化させる機構，shadowLB を提案し，実装を行った． shadowLB を使用することによって，既存コントロールプレーンを流用しつつ，ipvs+Linux サーバのみの構成と比較してテールレイテンシにおいて X%の向上が見られ

[5] https://kubernetes.io/ja/

た．また本研究では，ソフトウェアロードバランサーを実装するに当たり，コントロールプレーン，データプレーン双方のアーキテクチャを意識することなく設定を抽象化できるモデルを提唱する．これにより，既存の様々なソフトウェア資産を利用したロードバランサの開発への貢献が期待できる．

1.5 本論文の構成

本論文の構成を以下に示す．

第 2 章では，高速なソフトウェアロードバランサの例を紹介するとともに，データプレーン高速化に欠かせないパケット処理フレームワーク等について説明する．

第 3 章では，ソフトウェアロードバランサが直面しているコントロールプレーンの再利用性について論ずる．

第 4 章では，本研究において提案する shadowLB に求められることと，それを実現する手法について論ずる．

第 5 章では，shadowLB のコントロールプレーン-データプレーンをつなぐ中間レイヤの設計と第 6 章でも評価実験に用いる PoC(Proof of Concept) の具体的な実装について述べる．

第 6 章では，第 3 章で述べた課題に対して，本提案手法が有用であることを検証するための実証実験の概要及び具体的なシナリオについて述べ，結果を考察する．

第 7 章では，ロードバランサー以外のネットワーク機器にて実施されているソフトウェア資産の再利用を目的とした取り組みについて紹介する．

第 8 章では，本研究のまとめと本研究の展望について検討する．

第2章　ソフトウェアロードバランサ

本章ではソフトウェアロードバランサについて説明した上で既存のソフトウェアロードバランサについて紹介し，またソフトウェアロードバランサを論ずる上で必要不可欠な高速データプレーンの技術について論ずる．

2.1　ソフトウェアロードバランサ開発の背景

2.1.1　大規模パブリッククラウドの現状

近年のパブリッククラウドにおいては，ホスティングサービス上にホストされているサービスにとどまらず，自社サービスもデプロイされているケースが多い上に，そのトラフィック量は年々増加しており [1], 基礎となるインフラストラクチャに多大な負荷をかけている．

また，自社サービスをホストするデータセンターのネットワークはサービス同士の連携によってトラフィックが増大する．その結果，データセンター内部のサーバー間通信量も年々増加しており [10]，パブリッククラウドに求められるトラフィック処理要件はますます厳しくなっている．

2.1.2　ロードバランサーに求められる要件

2.2　高速なソフトウェアロードバランサ

本セクションでは，大規模サービスを運用している各企業のソフトウェアロードバランサについて紹介する．

2.2.1　Ananta

Ananta は，Microsoft Azure で採用されているソフトウェアロードバランサである．Ananta は一般的なアプライアンス型ロードバランサーがフェイルオーバー構成でスケールアップするアプローチになるのと対象的に，分散型構成でスケールアウトが容易な点が特徴的である．

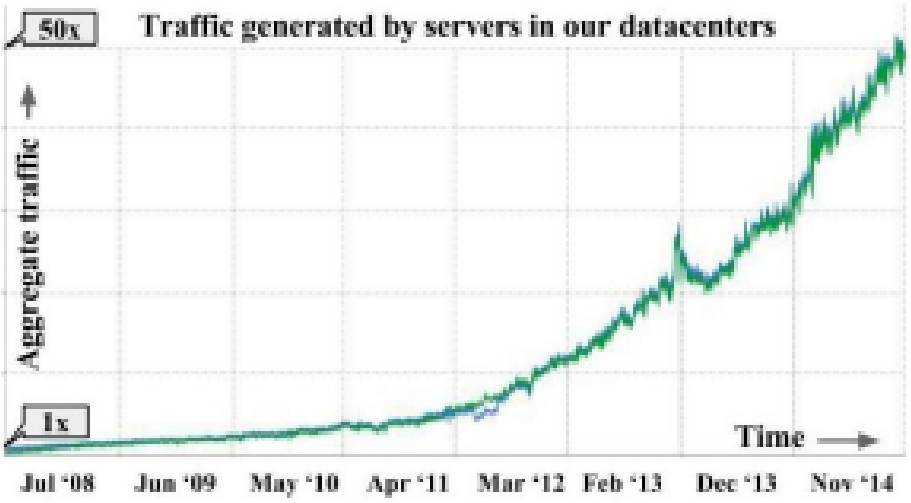

図 2.1: Aggregate server traffic in our datacenter fleet. [1] より引用

2.2.2 Magrev

2.2.3 GLB

2.2.4 Katran

2.3 高速データプレーン処理を支える技術

本セクションでは，ロードバランサの性能向上に必要不可欠な高速データプレーン技術について紹介する．

2.3.1 ソフトウェア技術

Kernel Bypass 系フレームワーク

Anti-kernel Bypass 系フレームワーク

独自 OS

2.3.2 ハードウェア技術

Fixed ASIC

Programmable ASIC

SmartNIC

第3章　データプレーン高速化に伴う課題

本章では 2.3 で述べたような技術を利用した際に生じる課題について述べる.

3.1　高速パケット処理フレームワークの課題

DPDK,

3.2　コントロールプレーン資産の再利用不可能性の弊害

コントロールプレーン資産を

第4章　shadowLB

本章では，3 章にて述べた問題点を解決するための基盤，shadowLB を実現手法の設計に関して論じる

4.1　概要

第5章　実装

本章では，第4章で述べた提案システムの設計と実装について述べる．

5.1　中間レイヤの設計

第6章　評価

本章では，第 4 章及び第 5 章で設計・実装に関して述べた本提案手法に関して，第 3.2 節で指摘したコントロールプレーンの再利用性の課題に対して有効性を示しつつ，データプレーンが高速化できていることを評価する．

6.1　評価要件

第7章　ネットワーク抽象化の関連技術

7.1　概要

本研究ではソフトウェアロードバランサにおけるコントロールプレーンの再利用性の向上を目指す．本章では，ロードバランサー以外のルータ，スイッチ，サーバといったネットワーク機器において行われているコントロールプレーンの再利用に向けた取り組みを紹介する．

第8章　結論

本章では，本研究の総括と今後の課題を示す.

8.1　本研究のまとめ

謝辞

参考文献

[1] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google ' s datacenter network. In *Sigcomm '15*, 2015.

[2] Cisco Systems. Cisco annual internet report（2018〜2023 年）ホワイトペーパー. `https://www.cisco.com/c/ja_jp/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020.

[3] Daniel E. Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. Maglev: A fast and reliable software network load balancer. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 523–535, Santa Clara, CA, 2016.

[4] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, et al. Ananta: Cloud scale load balancing. *ACM SIGCOMM Computer Communication Review*, 43(4):207–218, 2013.

[5] DPDK Intel. Data plane development kit, 2014.

[6] Luigi Rizzo. Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.

[7] GitHub. Github load balancer director. `https://github.com/osrg/gobgp/blob/master/docs/sources/dynamic-neighbor.md`, 2019.

[8] Hiroki Shiokura. Rapid evolution challenge at line's cloud. `https://speakerdeck.com/line_developers/rapid-evolution-challenge-at-lines-cloud`, 2020.

[9] linux vs.org. Ipvs. `http://linux-vs.org/software/ipvs.html`, 2016.

[10] Alexey Andreyev. Introducing data center fabric, the next-generation facebook data center network. `https:`

//engineering.fb.com/2014/11/14/production-engineering/
introducing-data-center-fabric-the-next-generation-facebook-data-center-network/,
2014.