M-036

独立分散型レイヤ 2 透過プロキシシステム上での ネットワークキャッシングによるサーバロードバランスの実現と評価

Implementation and Evaluation of Server Load-balancing with Network Caching Using the Independent and Distributed Layer-2 Transparent Proxy

桜打 彬夫 † Rick McGeer‡ 高田 秀志 §
Yoshio Sakurauchi Hideyuki Takada

1. はじめに

一般に，WEB のようなサーバクライアントシステムにおいて，オリジナルコンテンツを持つサーバのネットワーク付近では，クライアントのネットワーク付近に比べ，当該サーバに同じコンテンツを要求するリクエストが多く流れている．さらに，近年のネットワークトラフィックの爆発的な増加について，総務省のインターネット政策懇談会において，

- インターネットエクスチェンジとインターネットデータセンタやいわゆるキャッシュサーバの一体的な地方展開
- ネットワークの位置情報の活用による P2P アプリケーションの高度化

の 2 点が，官民一体となった実証実験を早急に行う必要があるものとして提言されている [1]．プロキシサーバ（以降，単に「プロキシ」と言う）は，ここでいうキャッシュサーバの一種である．

本稿では，この特徴と提言に着目し，プログラマブルスイッチ「OpenFlow」[2] による動的制御可能なレイヤ 2 透過プロキシシステムである「OpenProxy」[3] 上でネットワークキャッシングを実現することによる，サーバロードバランスを提案する．また，既存のロードバランサとの比較を行うと共に，ランダムネットワークを用いたシミュレーションによる評価を行い，その有用性を示す．

2. 既存のサーバロードバランス技術

現在のサーバロードバランス技術は，LVS（Linux Virtual Server）や専用機器などを用いて，密接に結びついた複数台のサーバに処理を分散させるものが主流である．しかし，このような技術は導入や保守が煩雑であり，同時に多数のリクエストが予想される大規模サイトのサーバなどへの適用に限られてきた．しかし，今日では，テレビやラジオなどのマスメディアだけでなく，様々な SNS（Social Networking Service）の影響で，サーバロードバランスが施されていないサーバへの突発的な多数のアクセスが発生することも珍しくない．

また，クライアントからの静的なコンテンツ取得要求に対する処理において，ロードバランサと CDN（Contents Delivery Network）の動作と目的は，ほぼ同じものとみなすことができる．プロキシシステムである OpenProxy を用いた提案手法と同様に，プロキシシステムを応用した CDN の 1 つに「Coral」[4] がある．Coral は，URL のホスト名のサフィックスに“.nyud.net”を付けることで利用できるサービスであり，特にサイトのトップページなどではユーザが明示的に URL を書き換える必要があるため，オリジナルコンテンツを持つサーバの負荷軽減の効果がユーザの行動に依存してしまうという問題がある．

3. OpenProxy によるサーバロードバランス

前節のような現状をふまえ，サーバから独立しており，ユーザから透過的に利用可能なサーバロードバランスを提案する．

†立命館大学大学院 理工学研究科
‡Hewlett-Packard Laboratories
§立命館大学 情報理工学部

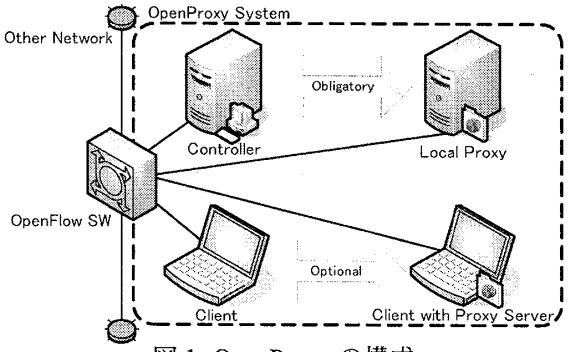

図 1: OpenProxy の構成

3.1 OpenProxy

まずはじめに，提案手法の要素技術である OpenProxy について説明する．OpenProxy は，プログラマブルスイッチ OpenFlow による動的制御可能なレイヤ 2 透過プロキシシステムである．図 1 に，OpenProxy の構成を示す．Local Proxy は，現状では Squid などの既存のプロキシ，Client（with Proxy Server）は，Windows PC などの既存の計算機を想定している．なお，Client with Proxy Server では，プロキシが動作しており，Local Proxy と同様に，リダイレクトされた他のクライアントからのリクエストを処理することができる．OpenFlow Switch は，OpenProxy の中心となるコンポーネントであり，Controller によって，その動作が制御される．具体的には，OpenFlow SW がリクエストを補足すると Controller にその通知を行い，Controller がどのようにリクエストを処理するかを決定し，OpenFlow SW に命令を出す．OpenFlow SW は命令に基づいて，通常のレイヤ 2 スイッチのように経路情報に基づいたリクエストの転送を行う他，NAT（Network Address Translation），あるいは NAPT（Network Address Port Translation）を用いて Local Proxy または Client with Proxy Server にリクエストのリダイレクトを行う．

OpenProxy は，レイヤ 2 で動作するため，リクエストが流れるネットワークの適切な位置に配置するだけで良く，クライアント側，あるいはサーバ側での設定は必要ない．さらに，OpenProxy は，Controller が制御する範囲において，サーバや他の OpenProxy と独立している．また，文献 [3] においては，iptables を用いた透過プロキシシステムとの性能評価を行い，一定の優位性が認められている．

3.2 サーバロードバランスの提供

先に述べたように，オリジナルコンテンツを持つサーバのネットワーク付近では，クライアントのネットワーク付近に比べ，当該サーバに同じコンテンツを要求するリクエストが多く流れている．そこで，サーバのネットワーク付近に OpenProxy を配置することで，キャッシュのヒット率が上がり，効率的な OpenProxy の運用ができると考えられる．これは，本来はサーバで処理されるリクエストを OpenProxy が処理しているという点で，サーバロードバランスの一種とみなすことができる．

OpenProxy の基本動作は，通常のレイヤ 2 スイッチと変わらない．すなわち，クライアントからサーバへの経路は，通常の IP の経路制御と同じである．つまり，クライアントからサーバへ至る経路の途中に OpenProxy を配置することで，オリジナルコンテンツを持つサーバをルート，一つの OpenProxy をノードとした，論理的な木構造の CDN が構築されることになる．また，サーバが複数台あることを考えれば，ネットワークに分散する OpenProxy がそれぞれのサーバについて同様の論理的な木構造の CDN を形成することになり，その効用が増すことが予想される．

4. ランダムネットワークを用いた評価

提案手法の有用性を確かめるために，ランダムネットワークを生成してサーバとクライアントをランダムに配置し，クライアントからサーバの能力を超えるリクエストを意図的に発生させ，リクエストがどのように処理されるのかの評価を行った．なお，今回は，簡単のため，一つのサーバに複数のクライアントからのリクエストが集中するものとした．

4.1 評価概要

今回のシュミレーションでは，サーバが属する上位のネットワークを表現するため，一つの AS(Autonomous System) を一つのノードとし，AS 内の通信は考慮しない．ノードには，

- 通常ノード 経路情報に基づいてサーバへのリクエスト転送のみを行う
- **OpenProxy** ノード 通常ノードの働きに加え，一定数のリクエストをプロキシにリダイレクトする

の 2 種類がある．

シュミレーションの手順は，以下の通りである．

1. 通常ノードと OpenProxy ノードを合わせて計 1024 ノード生成し，ランダムにノード間の接続を行う．接続にあたってのエッジ数は，1024 ノードの完全グラフを考えると 523,776 であるが，今回は 100 分の 1 の 5,238 とした．なお，全 1024 ノードは，最低一つのエッジで相互に接続されているものとする．
2. サーバとクライアントを，ラインダムネットワークのいずれかのノードにランダムに配置する．サーバの数は 1，クライアントの数はランダムとする．
3. 各クライアントから計 1024 のリクエストをランダムに発生させる．
4. リクエストは，IP ルーティングを模した最短経路のエッジを通ってサーバへ向かう．途中，OpenProxy ノードがあれば，その能力内においてプロキシにリダイレクトを行う．
5. サーバに到達したリクエストは，サーバの能力内において処理を行う．

手順 1 において，生成する 1024 ノードのうち，OpenProxy ノードの数を 0 から 1024 まで 4 ずつ変化させ，それぞれについて 1000 回の試行を行う．

また，サーバが同時に処理できるリクエスト数を表す同時接続数，サーバが待たせておけるリクエスト数を表す保留コネクション数は，Apatch 2.2 のデフォルト値を用いて，それぞれ 256，511 とした．さらに，OpenProxy ノードがリダイレクトできるリクエスト数は 128 とした．

4.2 結果と考察

各試行について，発生させた 1024 リクエストがどのように処理されたかを調べ，平均をとったものを図 2 に示す．リクエストは，サーバで処理された（Served），サーバで待たされた（Waited），サーバで拒否された（Rejected），OpenProxy で処理された（Redirected）のいずれかに当てはまる．

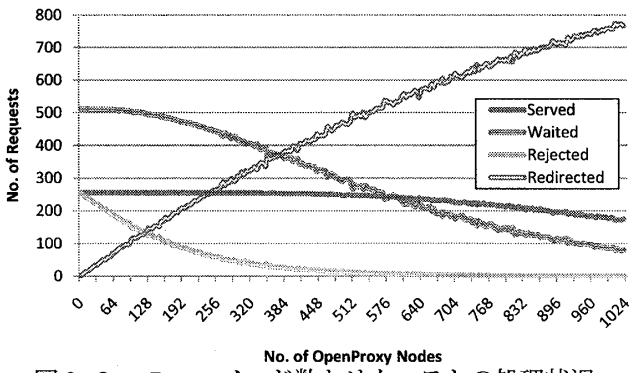

図 2: OpenProxy ノード数とリクエストの処理状況

まず，OpenProxy ノードの数が増えるにつれて，Waited と Rejected がすぐに減少しているのに対して，Served は 500 前後になるまで減少していない．これは，リクエストが十分な OpenProxy ノードを経由しなかったために，サーバにまでリクエストが到達したためだと考えられる．また，Rejected については，OpenProxy ノードがリニアに増加するのに対して，リニアよりも早く減少している．Rejected は，サーバに到着しているリクエスト数がサーバの能力を既に超えているにもかかわらず，サーバに対してコンテンツの要求をしたリクエストである．すなわち，Rejected の減少はサーバロードバランスが実現されたことを表す．

以上の結果から，ネットワーク上のサーバやクライアントの位置に関係なく，OpenProxy をネットワークに配置することは，サーバロードバランスの実現につながることが示されたと言える．さらに，例えばデータセンタを擁する AS の付近に OpenProxy を集中して配置するなど，恣意的に OpenProxy を配置することで，より少ない OpenProxy でより効果的なサーバロードバランスを提供できるものと考えられる．

5. おわりに

本稿では，OpenProxy 上でネットワークキャッシングを実現することによる，サーバロードバランスを提案した．また，ランダムネットワークを用いたシミュレーションによる評価を行い，その有用性を示した．今後は，より現実の WEB に近いネットワークトポロジを用い，通信路の帯域なども考慮した評価を行っていくとともに，OpenProxy をどのように動作させるかのチューニング，さらにネットワークトラフィックの削減について研究を進めていく予定である．

参考文献

[1] インターネット政策懇談会：インターネット政策懇談会 報告書, Technical report, 総務省 (2009).

[2] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow: enabling innovation in campus networks, *ACM Computer Communication Review*, Vol. 38, No. 2, pp. 69–74 (2008).

[3] 桜打彬夫, McGeer, R., 高田秀志：OpenProxy: プログラマブルスイッチ「OpenFlow」による動的制御可能なレイヤ 2 透過プロキシシステム, 情報処理学会 第 72 回全国大会, 2ZC-6 (2010).

[4] Freedman, M. J., Freudenthal, E. and Mazieres, D.: Democratizing Content Publication with Coral, in *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)* (2004).