分散システムのための DNS キャッシュヒット率及び応答速度の改善手法の提案

上田 義明 †

† 株式会社プレイド／放送大学

1 はじめに

近年のシステムの運用では，一般にリソースの効率的な利用や負荷分散，高速なスケールアウトの実現のために，複数のノードで構成された分散環境を用いる．

ドメイン・ネーム・システム (DNS) はドメインネームと IP アドレスを関連付けるグローバルな分散データベースであり，現在のほぼすべてのインターネットにおいて利用されており，必要不可欠なシステムとなっている．

DNS はクライアントとして用いられるスタブリゾルバと，権威サーバへの問い合わせを行うフルサービスリゾルバ，DNS 問い合わせの仲介を目的とした DNS フォワーダーで構成されることが多い．都度フルサービスリゾルバへ問い合わせを行うことは，ネットワーク遅延の観点から効率的ではないため，一般に DNS の応答データをキャッシュすることで効率的な DNS の運用を行う．

しかし，分散環境下の DNS では，ノード間で DNS のキャッシュデータを共有することができず，キャッシュヒット率の低下を引き起こす問題があった．これは Web のクローリングやブラウジング，E メール送信，IP アドレスからホスト名の逆引きなど，大量の DNS 問い合わせが発生する環境において応答速度の低下を引き起こす原因となる．

本研究では，同一のアプリケーションの名前解決には共通する問い合わせが多くあることに着目し，各ノードで解決した DNS キャッシュを非同期で共有し，可用性や応答速度を落とさずにキャッシュヒット率を改善する手法を提案する．本手法を用いた PoC の性能を評価し，分散環境における DNS キャッシュヒット率の改善と DNS 問い合わせの応答速度の改善を確認した．

2 従来手法

従来の手法で分散環境下で DNS のキャッシュヒット率を高めるには 2 つの構成が考えられる．ひとつは DNS クライアントであるスタブリゾルバの部分で KVS などの共有キャッシュストレージを用いた構成である．もうひとつはスタブリゾルバが問い合わせる DNS サーバーを，フルサービスリゾルバではなく，キャッシュを目的とした，DNS フォワーダー（キャッシュDNS サーバー）を間に挟むことでキャッシュヒット率を向上させた構成である．しかし，いずれの手法であっても，キャッシュの取得時にネットワーク経由の問い合わせによる遅延が発生する．

3 提案手法

図 1 に提案手法の概要を示す．提案手法では各ノードにスタブリゾルバを配置し，メッセージングモデルの Publish-Subscribe(Pub/Sub) 方式を用いて，DNS レコードデータを非同期で共有をすることで，応答速度や可用性への影響なく，キャッシュヒット率を向上させる．

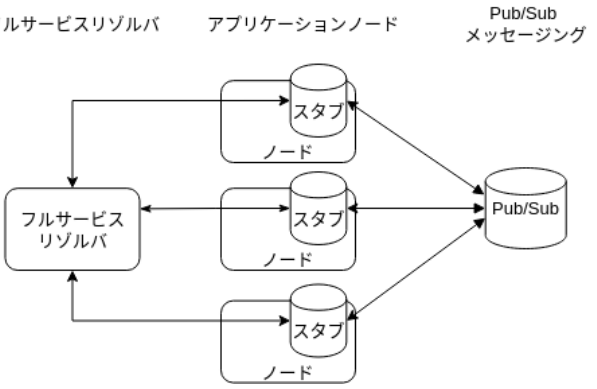

図 1: 提案手法の概要

4 評価

提案手法を評価するため，プロトタイプ実装をし，以下 4 手法を比較した．

- A: 共有キャッシュを用いない従来手法
- B: 共有キャッシュを用いる従来手法
- C: キャッシュDNS サーバーを用いる従来手法
- D: 提案手法

図 1 のスタブリゾルバには CoreDNS[1] を用いた．Pub/Sub 方式のメッセージングには Redis[2] を用いた．A には CoreDNS と CoreDNS 標準のプラグインの cache[3] を用いた．B には CoreDNS 標準のプラグインの cache と Redis を用いた分散キャッシュを実現する CoreDNS のプラグインの redisc[4] を用いた．C の DNS キャッシュサーバーは Dnsmasq[5] を用いた．† D の提案手法は

A Proposal of DNS Cache Hit Rate and Response Speed Improvement Technique for Distributed Systems

†Yoshiaki UEDA

†PLAID, Inc. / The Open University of Japan

CoreDNS 標準のプラグインの cache と独自で実装した Pub/Sub のプラグインを用いた．

評価には Web のクローラーをシミュレーションしたアプリケーションを作成し，クローリング後の CoreDNS のメトリクスをキャッシュヒット率と応答速度の観点から評価した．クローリングのアプリケーションは DataForSEO 社の日本での Web サイト上位 1000 件のドメイン [6] に対して，Headless Chromium[7] を用いて各ノードがドメイン直下のルートのみのレンダリングを行った．クローリングの対象となるドメインは，Redis にリスト型で保存し，各ノード上のアプリケーションが POP コマンドを用いて重複なくクローリングを行った．CoreDNS のメトリクスは Prometheus プラグインを用いて HTTP 経由でメトリクスの取得を行った．

表 1 の実験環境を 20 ノード用意し，ノード上にスタブリゾルバを構築した上でクローラーを実行し評価を行った．Pub/Sub や rediscのデータストアとなる Redis や，DNS キャッシュサーバーは表 1 と同様のものを，別途構築した．クローリングのキューの管理に用いる Redis はキャッシュのデータストアとは別に表 1 と同様のものを，別途構築した．評価に用いる値は各構成で 10 回の実行を行い，中央値を用いた．

表 1: 実験環境

環境	Google Cloud Platform
サービス	Compute Engine
シリーズ	N2 シリーズ カスタムマシンタイプ
CPU	Intel Cascade lake* vCPU * 4
メモリ	2GB
OS	Linux Ubuntu 20.04 LTS 64-bit
ゾーン	asia-northeast1-b

4.1 キャッシュヒット率

評価には CoreDNS のメトリクスの coredns_dns_requests_total を coredns_cache_hits_total で割ることでキャッシュヒット率を求める．B では rediscの coredns_redisc_hits_total をキャッシュヒットの値に加算しキャッシュヒット率を計算した．C では Dnsmasq のメトリクス取得のクエリを発行しメトリクスを取得した．D の提案手法では，独自でキャッシュヒット率のメトリクスを実装し，これをキャッシュヒットの値に加算しキャッシュヒット率を計算した．

実験の結果を表 2 に示す．

*実験日時点では，Google Cloud Platform の N2 シリーズでは，Cascade lake と Ice lake のみの提供となっており，本実験では CPU プラットフォームを Cascade lake 以降と指定することで，実験環境の CPU を Cascade lake のみに限定し実験を行った．

表 2: キャッシュヒット率の実験結果

CoreDNS + cache プラグイン	38.93%
CoreDNS + cache プラグイン + redisc	86.45%
DNS キャッシュサーバー	80.27%
CoreDNS + cache プラグイン + 提案手法	76.34 %

4.2 応答速度

評価にはスタブリゾルバの CoreDNS のメトリクスの coredns_dns_request_duration_seconds_sum を取得し，全ノードの値を合算し比較した．実験の結果を表 3 に示す．

表 3: 応答速度の実験結果

CoreDNS + cache プラグイン	1034.600 秒
CoreDNS + cache プラグイン + redisc	779.482 秒
DNS キャッシュサーバー	504.728 秒
CoreDNS + cache プラグイン + 提案手法	388.369 秒

5 おわりに

本論文では DNS レコードのデータを Pub/Sub を用いて非同期で共有する手法を提案した．実験では共有キャッシュを用いない従来手法と比較しキャッシュヒット率は 36 ポイントの向上，応答速度では 646 秒の高速化を確認した．共有キャッシュを用いる従来手法と比較しキャッシュヒット率は 11 ポイント低下したが，応答速度では 391 秒の高速化を確認した．キャッシュDNS サーバーを用いる従来手法と比較しキャッシュヒット率は 5 ポイント低下したが，応答速度では 116 秒の高速化を確認した．

DNS 問い合わせが多い分散環境では，本手法を用いることで DNS の構成を大きく変えることなく，キャッシュヒット率と応答速度の改善を確認した．

参考文献

[1] CoreDNS. https://coredns.io/.

[2] Redis. https://redis.io/.

[3] cache. https://coredns.io/plugins/cache/.

[4] redisc. https://coredns.io/explugins/redisc/.

[5] Dnsmasq. https://thekelleys.org.uk/dnsmasq/doc.html.

[6] Top 1000 websites. https://dataforseo.com/top-1000-websites.

[7] Headless chromium. https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md.

†CoreDNS は性能上の問題で実験で用いるクエリにすべて応答することができず，本実験では DNS キャッシュサーバーは Dnsmasq を用いた．