予測キャッシュを用いたDNS名前解決時間短縮手法の研究

藤山 大響[1]　神屋 郁子[2]　下川 俊彦[1]

概要：インターネットサービスの利用には名前解決が必要である．現代の Web ページの構成では複数の外部ドメインを含むため, 名前解決時間の蓄積はユーザ体験を損なう要因となる．そこで本研究では, 名前解決時間短縮を目的とした「予測型 DNS キャッシュシステム」を開発した．本システムは, ユーザのアクセスパターンから「共起性」と「周期性」に着目した．「共起性」に統計分析,「周期性」に機械学習を適用し, 将来の問い合わせを予測してプリフェッチする．事前に収集した DNS クエリログを用いたシミュレーション評価の結果, 従来手法と比較してキャッシュヒット率の向上と名前解決時間の削減を確認し, 本システムの有効性を実証した．

キーワード：DNS キャッシュ, プリフェッチ, 機械学習

Research on a Method for Reducing DNS Name Resolution Time Using Predictive Caching

Fujiyama Hibiki[1]　Kamiya Yuko[2]　Shimokawa Toshihiko[1]

Abstract: Internet service usage requires name resolution. Modern web page structures include multiple external domains, making accumulated name resolution time a factor that degrades user experience. Therefore, this research developed a “Predictive DNS Caching System” aimed at reducing name resolution time. This system focuses on “co-occurrence” and “periodicity” derived from user access patterns. It applies statistical analysis to “co-occurrence” and machine learning to “periodicity” to predict and prefetch future queries. Simulation evaluations using pre-collected DNS query logs confirmed improved cache hit rates and reduced name resolution times compared to conventional methods, demonstrating the system’s effectiveness.

Keywords: DNS Cache, Prefetch, Machine Learning

1. はじめに

インターネット上のサービス利用には, 通信相手を特定するためにドメイン名を IP アドレスに変換する必要がある．ドメイン名を IP アドレスに変換する仕組みが DNS（Domain Name System）である．DNS がドメイン名を IP アドレスに変換する処理を「名前解決」と呼ぶ．

名前解決は, 階層化された複数の DNS サーバを順にたどって IP アドレスを特定する仕組みである．そのため, クライアントとサーバ間のネットワーク的な距離やサーバの応答性能に起因する名前解決の待機時間が発生しやすい．1 回の名前解決にかかる時間は, ネットワーク環境やキャッシュの状況によって数ミリ秒から数百ミリ秒に及ぶことがある．現代のインターネットサービスにおいては,Web ページの表現力が向上し, 画像や広告, 解析スクリプトなど, 多数の外部ドメインからリソースを読み込む構成が一般的である．そのため, ページ全体の描画が完了するまでに発生する累積した名前解決時間は無視できない規模となる．特に, 多数の外部ドメインからコンテンツを読み込む場合, 名前解決の遅延が累積することでページの表示が遅れ, ユー

[1] 九州産業大学理工学部情報科学科
Department of Information Science, Faculty of Science and Technology, Kyusyu Sangyo University, Fukuoka 813–8503, Japan
[2] 福岡女子大学国際文理学部環境科学科
Department of Environmental Sciences, Faculty of International Arts and Sciences, Fukuoka Women’s University, Fukuoka 813–8529, Japan

ザ体験を損なう要因となる.

本研究の目的は名前解決時間を短縮することである.

2. DNS と関連技術

DNS（Domain Name System）[1] とは, ドメイン名とそれに関連付けられた様々な情報（リソースレコード）を階層的に管理・運用するシステムである. インターネット上の通信において, このシステムはドメイン名を対応する IP アドレスへ変換するために利用される. 以下では,DNS サーバとキャッシュおよび関連技術における課題について述べる.

2.1 DNS サーバ

DNS サーバには役割の異なる 2 種類のサーバが存在する. 1 つは, ドメイン名と IP アドレスの情報を管理している権威 DNS サーバである. もう 1 つは, クライアントに代わり権威 DNS サーバに問い合わせ, 結果を応答するキャッシュ DNS サーバである. また, キャッシュ DNS サーバは, 問い合わせ結果を一時的に保持するキャッシュ機能を持つ. これにより, キャッシュ DNS サーバはクライアントへ高速に応答している.

2.2 キャッシュとその課題

キャッシュに保持された情報には TTL（Time to Live）と呼ばれる情報が含まれる. これはキャッシュに情報を保持できる期間を示し,TTL の値が 0 になるとキャッシュから情報が破棄される. そのため, キャッシュに情報が存在しない際は権威 DNS サーバへの問い合わせ処理が必要である. これが名前解決に伴う待機時間を増大させる要因となっている.

2.3 関連技術と課題

本節では,2.2 で述べた名前解決に伴う待機時間の増大を解決するための既存技術について紹介する.

(1) **DNS Prefetch（Web）**[2]
ブラウザにおける DNS Prefetch は,HTML 内の画像やリンク先など, 将来的にアクセスが発生するドメイン名を事前に名前解決し, 遅延を削減する技術である. しかし, これには Web 制作者による HTML への明示的な記述が必要であり, 記述がないドメイン名に対しては事前に名前解決されない. また, 効果範囲はブラウザ内の閲覧行動に限定され,OS の更新やメールソフトによる通信など, ブラウザ外のドメイン名を事前に検知してプリフェッチできない.

(2) **DNS Prefetch（DNS サーバ）**[3]
クライアントへはキャッシュされている情報を即座に応答しつつ, バックグラウンドで権威 DNS サーバへ問い合わせ, キャッシュを最新の状態にする機能である. これは, キャッシュされている情報に対して問い合わせがあった際に,TTL の値が設定された閾値を下回った場合に機能する. このサーバ側の手法は, あくまで「既にキャッシュに存在する情報」の有効期間を更新する仕組みである. したがって, キャッシュに存在しない情報を事前に取得できない.

(3) **Running a Root Server Local to a Resolver**[4]
キャッシュ DNS サーバが上位の権威 DNS サーバの持つ情報をローカルに保持し, 自身への問い合わせ（ループバック通信）で完結させる手法である. これにより, 上位の権威 DNS サーバに対する名前解決の遅延を事実上ゼロにできる. しかし, この手法を利用するためには, 対象となる DNS サーバの全データを常に同期・保持し続ける必要がある. そのため, 管理コストやストレージ容量の制約から現実的ではない. したがって, クライアントが目的とするドメイン名の名前解決時間を短縮する決定的な解決策とはなりえない.

(4) **IP Anycast**[5]
共通のサービス用 IP アドレスを地理的に分散した複数のサーバ（ホスト）で共有する仕組みである. DNS の分野では, 権威 DNS サーバに加え,Google Public DNS や Cloudflare DNS に代表されるパブリック DNS においても, この技術が広く採用されている. クライアントがこれらのサーバに対してクエリを送信すると, ネットワーク経路的に最も近い場所に存在するサーバへとルーティングされる. この仕組みにより, 応答時間を短縮できる. しかし, 通信経路が最適化されたとしても, キャッシュミス発生時に権威 DNS サーバへ問い合わせるというプロセス自体はなくならない. したがって, 権威 DNS サーバの処理遅延や, 依然として残るネットワーク遅延の影響を完全に排除できない.

これらの既存技術の課題に対して, 本研究は過去のアクセスパターンから将来のクエリを動的に予測し, 権威 DNS サーバへの問い合わせプロセスを事前に完了させることで名前解決の時間を短縮する.

3. 予測型 DNS キャッシュシステム

目的達成のため, 本研究では「予測型 DNS キャッシュシステム」を開発した. 予測型 DNS キャッシュシステムとは, 将来名前解決されるドメイン名を予測し, 予めキャッシュすることで名前解決時間の短縮を図るシステムである.

3.1 要件定義

本システムの実現に必要な機能要件は 2 つある. 1 つ目は問い合わせ予測機能である. クライアントからクエリが送信される前に, 権威 DNS サーバへの問い合わせと結果のキャッシュを完了させる必要がある. よって, 本システムではクライアントから将来発生する問い合わせを事前に予

測する機能が求められる.

2 つ目は, プリフェッチ実行機能である. 本論文では, 予測に基づきユーザのアクセスより早いタイミングで能動的にキャッシュすることを「DNS プリフェッチ（以下, プリフェッチ)」と呼ぶ. クライアントからの要求よりも早いタイミングで名前解決処理を完了できれば, 権威 DNS サーバへの問い合わせに起因する待機時間を解消できる. よって, 本システムでは予測されたドメイン名をプリフェッチし, クライアントからの問い合わせが発生する前にキャッシュヒット可能な状態を確立する機能が求められる.

3.2 アクセスパターン

ユーザによるアクセスパターンは, ユーザの行動変化に伴い常に変動する. しかし, ユーザの Web ブラウジングやアプリケーションの動作は完全なランダムでなく, 一連の行動パターンに基づいていることが多い. そこで, 本研究ではユーザのアクセスパターンに見られる 2 種類の規則性に着目した. なお, このアクセスパターンを利用するためには DNS クエリログを使用する. DNS クエリログとは, 問い合わせとその応答が記録されたログのことである.

3.3 共起性

共起性とは, 特定のドメイン名への問い合わせに依存して別のドメイン名への問い合わせが発生する規則性である. 具体例としては,Web ページを開いた際に埋め込まれている画像などが配置された CDN への自動的なリクエスト発生や, ポータルサイトや検索エンジンの結果ページから特定のサービスや Web サイトへユーザがリンクをたどるなどが挙げられる.

3.4 周期性

周期性とは, 特定の時刻や曜日の条件に依存して特定のドメイン名への問い合わせが発生する規則性である. 具体例としては, 毎朝業務開始時にアクセスされるポータルサイトや, バックグラウンドで実行されるシステムの更新プロセスなどが挙げられる.

3.5 予測手法

これらの規則性を捉えるために, それぞれに適した予測手法を適用した. 共起性には統計的分析を適用する. 過去の DNS クエリログから「ドメイン名 A の出現後にドメイン名 B が出現する条件付き確率」を算出し, 確率の高いドメイン名を予測する.

周期性にはアクセスパターンの傾向を学習・予測可能な機械学習を適用する. DNS クエリログのタイムスタンプとドメイン名の出現関係に着目し, 個々のドメイン名に対する問い合わせ発生確率を算出する予測モデルを構築する.

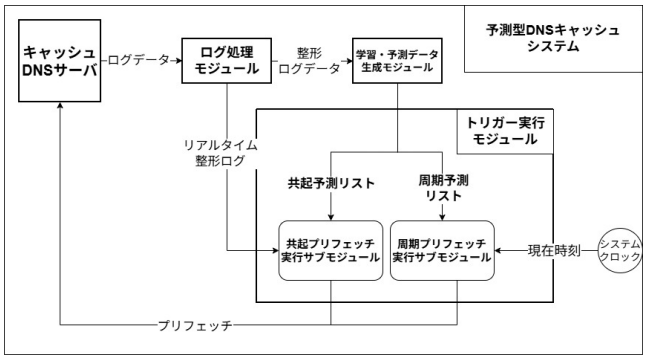

図 1 全体構成図
Fig. 1 Overall configuration diagram

3.6 システムの構成

本システムの全体構成図を図 1 に示す. 本システムは, キャッシュ DNS サーバに加え, ログデータの収集・加工をする「ログ処理モジュール」, 規則性を学習して予測データを生成する「学習・予測データ生成モジュール」, 実際にプリフェッチする「トリガー実行モジュール」の 3 要素によって構成される.

3.7 ログ処理モジュール

キャッシュ DNS サーバから出力される DNS クエリログを収集し, 分析可能な形式に正規化した整形ログを出力する. この整形ログは, 後段の学習・予測データモジュールおよびトリガー実行モジュール内の共起プリフェッチ実行サブモジュールの入力データとなる.

3.8 学習・予測データ生成モジュール

蓄積されたログデータをバッチ処理によって分析し, 予測モデルの構築や統計分析に必要なデータを生成するモジュールである. 3.5 で述べた予測手法に基づき, 最終的な予測結果として以下の 2 種類のリストを生成する. 1 つは共起予測リストである. あるドメイン名の問い合わせ後に, 高い確率で続いて問い合わせが発生すると予測されるドメイン名を対応付けたリストである. もう 1 つは周期予測リストである. プリフェッチを実行すべき時刻と, そのタイミングで問い合わせ発生が予測される対象ドメイン名を対応付けたリストである.

3.9 トリガー実行モジュール

生成したリストを参照してプリフェッチを実行するモジュールである. 動作トリガーに応じて,2 つのサブモジュールが機能する. 1 つ目は共起プリフェッチ実行サブモジュールである. ログ処理モジュールから出力される整形ログを監視し, 共起予測リストにあるドメイン名（トリガー）が検出された際に, 対応するドメイン名をプリフェッチする. もう 1 つが周期プリフェッチ実行サブモジュールである. 現在時刻を監視し, 周期予測リストで指定された時刻に到

達した際, 対象ドメイン名をプリフェッチする.

4. 予測型 DNS キャッシュシステムの実装

本システムはコンテナ仮想化技術である Docker[6] を採用し, キャッシュ DNS サーバと各モジュールを独立したコンテナとして実装した. また, 複数のコンテナを管理するためのツールとして Docker Compose を使用した. これにより, 各コンテナの起動順序, ネットワーク接続, およびボリュームの設定を YAML 形式のコードとして定義・管理できる.

4.1 Unbound コンテナ

キャッシュ DNS サーバには Unbound[7] を採用し, ポート 53（UDP/TCP）でリクエストを受け付ける. DNS クエリログの出力には, ディスク I/O 負荷の低いバイナリ形式である dnstap を採用した. dnstap とは DNS ソフトウェアのための構造化されたバイナリログである. Unbound コンテナとログ収集コンテナの間では,dnstap 用ソケットファイルを共有して通信する.

4.2 ログ収集コンテナ

ログ収集コンテナは, ログ処理モジュールのログ収集機能を担う. Unbound から出力される dnstap 形式のバイナリストリームを受信し, 分析しやすい CSV 形式に変換・保存する. 実装には go-dnscollector[8] を採用した. go-dnscollector とは DNS サーバからの DNS クエリと応答をキャプチャし, クリーンなデータを送信する軽量ツールである. ログ収集コンテナは,Unbound コンテナと共有されたボリューム上のソケットファイルを介してデータを受信する. その後, 変換後のデータを解析用ボリュームへ CSV ファイルとして書き出す.

4.3 Python コンテナ

Python[9] コンテナは, 学習・予測データ生成モジュールとトリガー実行モジュールの処理を担う Python 環境である. コンテナ内で各 Python スクリプトを実行し, 予測データの生成やプリフェッチの実行, ログの監視処理をする. ログ処理モジュールのログ整形処理も本コンテナで実行される. また, 機械学習には Microsoft が開発した勾配ブースティング決定木ベースの機械学習アルゴリズムである LightGBM[10] を使用し, 予測モデルを構築する.

5. 評価

本研究では, 予測型 DNS キャッシュシステムの有効性を検証するために, 評価指標を定義する. さらに, 評価指標を算出するための手段として, 本評価では収集済みの DNS クエリログを用いてシミュレーションする. 過去に蓄積された DNS クエリログを使用することで, 学習データ（過去）と評価データ（未来）を区分し, 評価用データを「予測すべき正解データ」として扱うことができる. これにより, プリフェッチしたドメイン名が, その後のトラフィックにおいて実際にクエリされたかをシミュレーションでき, 明確に判定することが可能になる. 本研究で使用するログデータは著者らが所属する研究室で収集したものである. 学習データを 2025 年 10 月 26 日から 11 月 26 日, 評価データを 12 月 4 日とすることで予測性能を検証した. また,11 月 27 日から 12 月 3 日のデータを予測モデルが過学習しないよう学習回数を調整する検証データとして使用した.

評価方法は, シミュレーションするための評価スクリプトを作成し, 予測すべき正解データ（評価データ）と予測型 DNS キャッシュシステムによって生成された共起予測リストと周期予測リストを入力として性能結果を出力する.

5.1 評価指標の定義

本評価では, システムの性能を評価するため, 以下の指標を定義する. また, 各数式における変数は以下の通りとする.

- □ N_{total}：全クエリ数
- □ N_{legacy}：既存キャッシュ（TTL 内）によるヒット数
- □ $N_{\text{prefetch\_hit}}$：予測型 DNS キャッシュシステムのプリフェッチによるヒット数
- □ N_{miss}：プリフェッチを行ったが（または行われず）ミスとなった回数
- □ $N_{\text{generated}}$：システムが発行したプリフェッチの総数
- □ N_{timing}：予測ドメインは正解であったが,TTL 切れによりヒットしなかった回数

(1) **キャッシュヒット率**

全クエリに対してキャッシュによる応答ができた割合である. ログに記録された既存のキャッシュ機構のみによるキャッシュヒット率と, 本システムによる最終的なキャッシュヒット率を比較し, その上昇幅を評価する.

$$\text{キャッシュヒット率} = \frac{N_{\text{legacy}}}{N_{\text{total}}} \times 100 \tag{1}$$

$$\text{本システムのヒット率} = \frac{N_{\text{legacy}} + N_{\text{prefetch\_hit}}}{N_{\text{total}}} \times 100 \tag{2}$$

(2) **総解決短縮時間**

プリフェッチが成功したことで削減された待機時間の総和である. 各クエリ i において, プリフェッチを行わなかった場合に権威 DNS サーバへの問い合わせにかかるはずだった時間 $t^{(i)}_{\text{response}}$ を積算して算出する.

$$\text{総解決短縮時間} = \sum_{i \in \text{Hits}} t^{(i)}_{\text{response}} \tag{3}$$

(3) **適合率**

システムが生成・発行したプリフェッチの総数のうち, 実際にキャッシュヒットに繋がった（役に立った）割合である. この値が高いほど無駄なプリフェッチが少なく, 効率的に予測が行われていることを示す.

$$\text{適合率} = \frac{N_{\text{prefetch\_hit}}}{N_{\text{generated}}} \times 100 \quad (4)$$

(4) **カバー率**

既存のキャッシュ機構ではミスとなっていたクエリのうち, 本システムによってキャッシュヒットできた割合である. 本研究におけるカバー率は, 全クエリ数ではなく「プリフェッチによるヒット数と実際のミス数の和（$N_{\text{prefetch\_hit}} + N_{\text{miss}}$）」を分母とする.

$$\text{カバー率} = \frac{N_{\text{prefetch\_hit}}}{N_{\text{prefetch\_hit}} + N_{\text{miss}}} \times 100 \quad (5)$$

(5) **タイミングミスマッチ率**

予測したドメインへのアクセスは実際に発生したが, プリフェッチのタイミングが早すぎたために TTL 有効期限切れとなり, キャッシュヒットに至らなかった割合である. 発行したプリフェッチ総数に対する割合として算出する. タイミングミスマッチ率が高いほど TTL 期限後に予測ドメイン名へのアクセスが多く, 低いほど期限内のアクセスが多い, もしくはアクセスがなかったことを意味する. なお, 本指標は 100%を超える場合がある. これは,1 回のプリフェッチ発行（分母）に対して, 有効期限が切れた後に同一ドメイン名へのアクセスが複数発生（分子）し, それらが全てタイミングミスマッチとして計上されるためである. 例えば, タイミングミスマッチ率が 200%であれば,1 回のプリフェッチに対して平均 2 回の期限切れ後アクセスが発生したことを示す.

$$\text{タイミングミスマッチ率} = \frac{N_{\text{timing}}}{N_{\text{generated}}} \times 100 \quad (6)$$

(6) **クエリ量増加率**

プリフェッチのために追加で発生した DNS クエリの量が, 元のユーザの DNS クエリの量に対してどの程度の割合かを示す指標である. これは高速化に対するネットワークコストを表す.

$$\text{クエリ量増加率} = \frac{N_{\text{generated}}}{N_{\text{total}}} \times 100 \quad (7)$$

5.2 評価結果

本節では,5.1 で示した指標に基づき, シミュレーションによって得られた評価結果を述べる. 使用する評価データに含まれる総クエリ数は 135,473 件であり, 待機時間の総和である総名前解決時間は 445.5 秒である. 評価結果を表 1 に示す. 表中の各列は左から順に「評価指標」,「周期予測のみ適用した結果」,「共起予測のみ適用した結果」, および周期予測と共起予測を適用した「全体の結果」を示している. また, 一番右の列（既存）は比較基準として本システムによるプリフェッチを適用せず, 既存のキャッシュ機構のみを利用した場合の測定値である. 評価の結果, 予測型 DNS キャッシュシステム導入によるキャッシュヒット率の増加が確認された. また, 総解決短縮時間において全体で約 31 秒の短縮効果を示した.

表 1 評価結果
Table 1 Evaluation results

評価指標	周期のみ	共起のみ	全体	既存
キャッシュヒット率	55.7%	60.1%	60.3%	55.5%
適合率	11.6%	35.1%	32.0%	-
カバー率	0.5%	10.4%	10.8%	-
総解決短縮時間	1.2 sec	29.7 sec	30.7 sec	-
タイミングミスマッチ率	210.3%	123.0%	118.0%	-
クエリ量増加率	1.9%	13.2%	15.0%	-

表 2 確信度 40%における評価結果
Table 2 Evaluation results at 40% confidence level

評価指標	周期のみ	共起のみ	全体	既存
キャッシュヒット率	55.7%	67.7%	67.8%	55.5%
適合率	11.6%	37.6%	36.1%	-
カバー率	0.5%	27.4%	27.7%	-
総解決短縮時間	1.2 sec	87.3 sec	88.0 sec	-
タイミングミスマッチ率	210.3%	60.9%	60.7%	-
クエリ量増加率	1.9%	32.4%	34.1%	-

また, 共起予測リストにおける確信度（予測されたドメイン名に付与されたそのドメイン名が出現する条件付き確率）の閾値を 40%とした場合の評価結果を表 2 に示す. この場合, 予測されたドメイン名の確信度が 40%以上であればプリフェッチを実行する. 表 1 と比較すると, 共起予測によるキャッシュヒット率や総解決短縮時間が大きく向上していることが確認された. 一方で, クエリ量においても大きく増加していることが確認された.

さらに, 確信度の閾値を 80%とした場合, 表 1 と比較すると共起予測によるキャッシュヒット率や総解決短縮時間が低下していることが確認された. 確信度 80%における評価結果を表 3 に示す.

周期予測リストの予測スコア（予測されるドメイン名の出現確率）についても閾値を 40%と 80%とした場合について評価した. 表 1 と比較したがどちらも結果に変化は見られなかった.

表 3　確信度 80%における評価結果
Table 3　Evaluation results at 80% confidence level

評価指標	周期のみ	共起のみ	全体	既存
キャッシュヒット率	55.7%	57.3%	57.5%	55.5%
適合率	11.6%	24.1%	21.4%	-
カバー率	0.5%	4.0%	4.4%	-
総解決短縮時間	1.2 sec	14.0 sec	15.2 sec	-
タイミングミスマッチ率	210.3%	184.0%	169.5%	-
クエリ量増加率	1.9%	7.3%	9.2%	-

5.3　考察

評価シミュレーションによる結果についての考察を述べる．本シミュレーションの結果より, 名前解決による待機時間の短縮が確認された．さらに, 確信度の閾値設定がシステムの性能に大きなトレードオフをもたらすことが明らかになった．また, 予測スコアの閾値の変化が結果に影響を与えなかった点について, その要因を考察する．

5.4　確信度とクエリ量の関係

表 3 に示す通り, 確信度の閾値を 80%と高くした場合, クエリ量の増加率は約 9.2%と低く抑えられたものの, キャッシュヒット率の向上は約 2.0 ポイントと限定的であった．対照的に閾値を 40%まで緩和した場合（表 2）ではキャッシュヒット率が約 12.3 ポイントと大幅に向上している．この要因として, ログ収集環境（研究室ネットワーク）におけるトラフィックの特性が挙げられる．本環境では, システムによる定期的な自動アクセスが多く, 共起予測リストのうち確信度が高いものはこれら特定の「人気ドメイン」や「定期実行タスク」によって占有されていたと考えられる．これらは予測しやすい反面, キャッシュ効果による時間短縮は限定的になりやすい．一方で, 人間によるブラウジング活動は不規則であり, 統計的に算出される確信度は相対的に低くなる傾向がある．閾値を 40%まで引き下げることで, こうした「人間による不確実だが価値のあるアクセス」を予測対象に含めることができ, 結果としてキャッシュヒット率や名前解決時間の短縮効果が大きく改善したと考える．ただし, これにはクエリ量増加というコストを伴うため, 運用環境に応じた調整が必要である．

5.5　予測スコアと特徴量生成上の制約

予測スコアの閾値を 40%から 80%に変化させても評価結果に差異が見られず, 表 1 と同様の結果となった．周期予測リストを確認したところ, 生成された予測タスクのスコアの大部分が約 0.97 以上という高い値に集中していることが判明した．これは, 予測モデルを作成するための特徴量生成段階における計算リソース（メモリ）の制約により, 頻度の低いドメイン名を前処理の段階でフィルタリング（削除）したことに起因する．結果として, 最終的な周期予測リストには極めて発生確率の高いドメイン名のみが残ることとなり, 閾値を 40%から 80%の範囲で操作しても, 対象となるタスクの集合が変化しなかった．このことから, メモリの制約を解消し, より広範囲なスコア分布を持つタスクを含めることで, 閾値調整によるキャッシュヒット制御が可能になると考える．

6.　まとめ

本研究では, 名前解決時間の短縮を目的とし, ドメイン名の予測およびプリフェッチをする予測型 DNS キャッシュシステムを開発した．開発した予測型 DNS キャッシュシステムの有用性を検証するために, 事前に収集した DNS クエリログを用いてシミュレーションによる評価をした．その結果, 本システムのキャッシュヒット率は従来手法を上回り, その向上が確認された．さらに, 総解決短縮時間の算出結果は, 本来であれば権威 DNS サーバへの問い合わせにかかる処理を, プリフェッチによってキャッシュからの応答に置換できたことを示している．よって, 本目的は達成できたと言える．

今後の課題は, 特徴量生成ロジックの改善が挙げられる．頻度の低いドメイン名情報を削除した結果, 周期予測リストに予測スコアが高いドメイン名しか残らない現象が起きた．これらを削除しない効率的な計算ロジックを実装できれば, 周期予測におけるプリフェッチ効果の向上ができると考える．

参考文献

[1] 渡邉結衣, 佐藤新太, 藤原和典：DNS がよくわかる教科書, SB クリエイティブ株式会社 (2018).
[2] MDN contributors.：dns-prefetch の使用 (online), 入手先 〈https://developer.mozilla.org/ja/docs/Web/Performance/Guides/dns-prefetch〉
[3] NLnet Labs.: *unbound.conf(5)* (online), 入手先 〈https://unbound.docs.nlnetlabs.nl/en/latest/manpages/unbound.conf.html〉
[4] Warren K. and Paul H.: *RFC 8806 Running a Root Server Local to a Resolver* (online), 入手先 〈https://www.rfc-editor.org/rfc/rfc8806〉
[5] 株式会社日本レジストリサービス (JPRS).：IP Anycast (online), 入手先 〈https://jprs.jp/glossary/index.php?ID=0108〉
[6] Docker, Inc.: *Compose file reference* (online), 入手先 〈https://docs.docker.com/reference/compose-file/〉
[7] NLnet Labs.: *Unbound by NLnet Labs* (online), 入手先 〈https://unbound.docs.nlnetlabs.nl/en/latest/〉
[8] Denis Machard.: *DNS-collector* (online), 入手先 〈https://github.com/dmachard/DNS-collector〉
[9] Python Software Foundation.: *Beginners Guide* (online), 入手先 〈https://wiki.python.org/moin/BeginnersGuide/Overview〉
[10] Microsoft Corporation.: *Welcome to LightGBM's documentation!* (online), 入手先 〈https://lightgbm.readthedocs.io/en/stable/〉