DNS を用いた分散 Web システムにおける負荷分散機構の改良と評価

16G473　森垣 航太 （最所研究室）

DNS を用いた分散 Web システムにおいて，サーバ間での負荷の偏りを防ぐ振分け一時停止機能と，可能な限り長い TTL を使用しつつ負荷の偏りを防ぐための可変式 TTL を用いた重み付けラウンドロビン機能の提案及び適用したシステムの評価について述べる．

1　はじめに

本研究室では，負荷量に応じてクラウド上のキャッシュサーバの台数を動的に増減することで，応答性を維持しつつ運用コストを削減する分散 Web システムを開発している．先行研究では，ロードバランサを用いた負荷分散機構が開発された [1]．ロードバランサを用いる場合，全てのリクエストの振分けを制御できるため柔軟な振分けが可能だが，リクエストが増加した際にロードバランサ自体がボトルネックになる可能性がある．そこで，本研究では DNS を用いた負荷分散機構を開発することにした [2]．本機構では DNS ラウンドロビン機能を用いてリクエストの振分けを行っているが，サーバ間での負荷の偏りが発生し応答性が低下してしまう場合があった．この問題を解決するために，負荷が集中するサーバへの振分けを一時的に停止する機能を実装した．さらに，可能な限り長い TTL を使用しつつ負荷の偏りを防ぐことを目的とした可変式 TTL を用いた重み付けラウンドロビン機能を実装した．本稿では，実装したこれらの機能及び評価について述べる．

2　DNS を用いた分散 Web システム

図 1 に DNS を用いた分散 Web システムを示す．本システムは管理サーバ，権威 DNS サーバ，オリジンサーバとクラウド上のキャッシュサーバで構成されている．管理サーバはオリジンサーバやキャッシュサーバの負荷監視，その結果に伴うキャッシュサーバの起動・停止と DNS ゾーン情報の更新を行う 3 つの機能を持つ．負荷量として稼働中のサーバの稼働率 (現在のリクエスト処理数/最大同時処理数) の平均を使用する．平均稼働率が閾値 Th_{high} を上回った場合にはキャッシュサーバを起動，閾値 Th_{low} を下回った場合には停止させる．また，各クライアントからのリクエストの振分けには DNS ラウンドロビンを用いる．

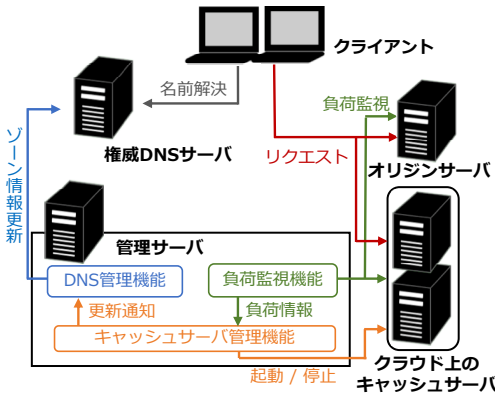

図 1: DNS を用いた分散 Web システム

3　DNS ラウンドロビンを用いた負荷分散機構の評価および問題点

前節で述べた DNS を用いた分散 Web システムの評価実験を行った．実験環境として，4 台のホストマシン上に仮想マシンとしてクライアント 12 台，ルート DNS サーバ 1 台，権威 DNS サーバ 1 台，DNS キャッシュサーバ 12 台，管理サーバ 1 台，オリジンサーバ及びキャッシュサーバ 10 台を構築した．クライアントからのリクエストを段階的に増加させ，オリジンサーバ及びキャッシュサーバへ負荷をかけていく実験を行った．

実験の結果，各サーバの稼働率がある程度上下している部分と極端に高いサーバと低いサーバに分かれている部分があり，サーバ間で長時間の負荷の偏りが発生していた．また，長時間稼働率の高い状態が続いたサーバの応答時間を確認したところ，稼働率が高い状態が続いた後，最大応答時間が 60 秒付近まで増加していた．それに伴い，その区間の平均応答時間も 6 秒付近まで増加していた．この値は同じ区間の全サーバでの平均応答時間の約 3 倍であった．これらのことから，DNS ラウンドロビンを用いた単純な振分けでは均等な負荷分散はできず，応答性が低下してしまう場合があることが確認できた．

4　振分け一時停止機能

前節で述べた問題を解決するために，過負荷サーバへのリクエスト振分けを一時的に停止させる機能を実装した．この機能では，稼働率が閾値 Th_{over} を上回ったサーバを過負荷状態と見なし，Th_{over} を下回るまでリクエストの振分けを一時停止させる．実装には DNS ソフトウェアである PowerDNS[3] の PipeBackend モジュールを用いた．このモジュールは，PowerDNS への名前解決の問い合わせに対してユーザが作成したプログラムを用いた回答内容の生成を可能とする．

実装した振分け一時停止機能の評価実験を行った．ケース A を本機能なし，ケース B を $Th_{over} = 0.6$，ケース C を $Th_{over} = 0.8$，ケース D を $Th_{over} = 1.0$ とし，各 5 回ずつ実験を行い，平均を比較する．その他の実験環境及び実験手順は前節と同様である．TTL60 秒での実験結果を表 1 に示す．コストはオリジンサーバ及びキャッシュサーバの稼働時間の合計である．平均応答時間，1 コストあたりの処理数共にケース D が最も良い結果となった．また，平均応答時間ではケース A が，1 コストあたりの処理数ではケース B が最も悪い結果となった．しかし，各ケースでの結果にそこまで大きな差がないことがわかる．また，各ケースでの応答時間を比較したところ，紙面の都合で結果の表は省略するが，本機能を用いることで 3 秒未満の短い応答時間の割合が増加し，3〜9 秒の応答時間の割合が減少してい

た．しかし，9〜60 秒の長い応答時間の割合も増加していた．これらの結果から，TTL に 60 秒を使用した場合では本機能を用いることで一部は改善されるが，悪化してしまう部分があることが確認できた．

表 1: 各ケースの実験結果 (TTL 60)

ケース	コスト	合計 応答時間(秒)	処理数	平均 応答時間(秒)	1コスト あたりの処理数
A	7571	627947	700772	0.90	92.56
B	7904	629728	725834	0.87	91.83
C	7752	635798	716271	0.89	92.39
D	7909	622597	739972	0.84	93.56

次に，TTL を 30 秒に設定し同様の実験を行った．実験結果を表 2 に示す．TTL60 秒の場合と違い，本機能を用いることで平均応答時間，1 コストあたりの処理数共に結果が良くなっており，改善されていることがわかる．また，応答時間に関しても，本機能を用いることで短い応答時間の割合が大きく増加し，長い応答時間の割合は同等以下になっていた．これらの結果から，使用する TTL が短くなるほど振分け一時停止機能による応答時間及びコストパフォーマンス改善の効果が大きいことが確認できた．また，本実験環境では閾値 Th_{over} の違いによる値の変化はあまり見られなかった．これは，頻繁にリクエスト振分けの一時停止が行われることで各サーバの稼働率が激しく上下してしまい，本実験での閾値の差程度では大きな影響はなかったためであると考えられる．

表 2: 各ケースの実験結果 (TTL 30)

ケース	コスト	合計 応答時間(秒)	処理数	平均 応答時間(秒)	1コスト あたりの処理数
A	7257	643281	661994	0.97	91.22
B	8046	600700	786708	0.76	97.77
C	8272	591389	804834	0.73	97.29
D	7576	608056	768810	0.79	101.48

5 可変式 TTL を用いた重み付けラウンドロビン

前節において，短い TTL を設定した上で振分け一時停止機能を用いることでサーバ間での負荷の偏りが緩和され，応答性が改善することを確認した．しかし，短い TTL を使用する場合，名前解決の問い合わせ頻度が増えてしまうため，DNS サーバへの負荷の増加に繋がる．サーバのスケールアウトやリクエストの振分け一時停止が行われる際には短い TTL を使用することで，各クライアントへの振分け先更新の反映を早くしたい．これらの処理が行われるのはサーバが高負荷状態のときであり，それ以外では短い TTL を使用する必要性は低い．そのため，状況に応じて必要とされる TTL は異なると言える．そこで，負荷の低いサーバへは通常の TTL を，負荷の高いサーバへは負荷状態に応じて短くした TTL を設定し，選択した IP アドレスの中で最も短い TTL を全体の TTL として使用する機能を実装した．また，重み付けラウンドロビンの考え方を導入し，負荷に応じた重みの決定と名前解決の問い合わせの回答に使用する IP アドレス数を制限することで，負荷の低いサーバが選択される確率を高くし，必要以上に短い TTL が使用されないようにした．

実装した機能において，回答に使用する IP アドレス数の違いによる影響を調べるため実験を行った．IP アドレス数が 1,2,4,8 個の場合をそれぞれケース A〜D とした．表 3 に各ケースの実験結果を示す．平均応答時間，1 コストあたりの処理数共にケース A が最も悪い結果，ケース C が最も良い結果となっていることがわかる．応答時間についても同様にケース C が最も短い応答時間の割合が大きかった．また，平均 TTL 値はケース A で 56.5 秒，ケース B で 51.6 秒，ケース C で 42.4 秒，ケース D で 33.3 秒となった．これらの結果から，IP アドレス数が多いほど使用される TTL が短くなるため，振分け先更新の反映が早くなり応答性やコストパフォーマンスが良くなるが，多すぎる場合は負荷の高いサーバも選択されやすくなるため，結果が悪くなってしまうことが確認できた．また，本実験環境では IP アドレス数を 4 個に設定することで最も良い結果が出た．

表 3: 各ケースの実験結果 (IP アドレス数変更)

ケース	コスト	合計 応答時間(秒)	処理数	平均 応答時間(秒)	1コスト あたりの処理数
A	7103	478190	452622	1.06	63.72
B	7754	411042	524793	0.78	67.68
C	8192	343047	624244	0.55	76.20
D	8109	366767	582246	0.63	71.80

次に，本機能を用いる場合の最も良い結果と，固定の TTL で全サーバの IP アドレスを返す場合の結果の比較を行った．固定の場合の TTL は，本機能を用いた場合のケース C の結果 (42.4 秒) とほぼ同じにするため，40 秒に設定した．実験結果を表 4 に示す．平均応答時間は 0.05 秒と大きな違いはないが，1 コストあたりの処理数では約 8 の差があり，本機能を用いることで少ないコストで多くのリクエストを処理できるようになったと言える．この結果から，本機能を用いることで平均 TTL 値が大きい場合でも応答性やコストパフォーマンスが改善されることを確認できた．

表 4: 各ケースの実験結果 (機能あり/なし)

ケース	コスト	合計 応答時間(秒)	処理数	平均 応答時間(秒)	1コスト あたりの処理数
機能あり	8192	343047	624244	0.55	76.20
機能なし	8374	345536	572148	0.60	68.33

6 まとめ

過負荷サーバへの振分けを一時的に停止させる振分け一時停止機能を導入することで，短い TTL を使用する場合であればサーバ間での負荷の偏りが緩和され，応答性が改善されることを確認した．また，負荷に応じて TTL と重みを決定する可変式 TTL を用いた重み付けラウンドロビンを導入することで，従来よりも長い TTL を使用しつつ応答性及びコストパフォーマンスを改善できることを確認した．

今後の課題として，重みと TTL を決定するアルゴリズムの改良，サーバの稼働台数に応じてオートスケールの閾値を自動決定するアルゴリズムの検討などが考えられる．

参考文献

[1] A. Horiuchi, and K. Saisho, “Prototyping and Evaluation of Virtual Cache Server Management Function for Distributed Web System”, The 2015 International Conference on Computational Science and Computational Intelligence (CSCI’15), pp.324-329, 2015

[2] 森垣航太，“分散 Web システムにおける DNS を用いた負荷分散機構の実装と評価”，香川大学，卒業論文，2016

[3] PowerDNS, `https://www.powerdns.com/`