DNSラウンドロビンの際に起きる負荷の偏りを軽減する方法

秋田谷駿一[1,a)]　串田 高幸[1]

概要：クライアントからのリクエストをサーバに均等に転送する方式のロードバランシングを DNS サーバを用いて実現した DNS ラウンドロビンが存在する．DNS ラウンドロビンは他のロードバランサーと比べ安価で導入しやすいという特徴がある．しかし, その反面負荷の偏りが起こる問題がある．本研究では, 負荷の状態によってゾーン情報を更新し少ない順に並び替えることで負荷の偏りを軽減を図る．

1. はじめに

1.1 背景

インターネットでは, 0 から 255 までの数値を 4 つ組み合わせる (IPv4 の場合) ことで表現されるため人間には覚えられない. そこで個々のコンピュータ名を識別するためにドメイン名が必要である [1] . ドメイン名は, インターネットの発展とともに膨大な数となってきたためそれを管理・運用するシステムの必要性が出てきた．そのため開発されたのが, DNS である．正式名称は, Domain Name System という．DNS は, 世界中に存在するサーバをつなげて構築されているデータベースのためこのサーバにアクセスすることでドメイン名から IP アドレスを検索したりメールアドレスからサーバを特定することが可能になっている．DNS は, 電話帳のように表されることが多く個々のコンピュータへの通信の際に参照することで通信したい相手を見つけることができる．DNS の役割を担うサーバには, キャッシュサーバと権威サーバが存在する．キャッシュサーバの役割は, クライアントから集めたクエリを一定期間保存することで DNS の通信がネットワーク上に飛び交ったり権威サーバに高い負荷をかけないためのものである．権威サーバの役割は, キャッシュサーバからくる名前解決要求に対して返答を返すサーバである．キャッシュサーバはキャッシュに存在しない名前解決要求を権威サーバに送る. 基本的に, ユーザがアクセスするサーバは, キャッシュサーバである．権威サーバは構築したシステムの通信に関わる情報を持っているためこのサーバが落ちるとシステムは, 動作しない．この権威サーバを狙った攻撃が存在している．代表的なものだと DDOS 攻撃がある．[2] これは, ランダムで存在しない名前の DNS サーバへのリクエストを大量に送りつけるものである. キャッシュサーバは, 知らない情報のため権威サーバへ問い合わせをすることで権威サーバへ負荷をかけるものだ．DNS を悪用した DNS キャッシュポイズニングという攻撃も存在する．これは, DNS の脆弱性を利用して偽の情報を DNS に記憶させることでアクセスしたいページと別なページを表示するユーザに対した攻撃である．現在のネットワークトラフィック量は, 急速に成長しておりネットワークの混雑, サーバの過負荷は深刻な問題となっている [3]. この問題を解決するためにロードバランサを用いてロードバランシングしている．ロードバランシングによりシステムのスループットが最大になるように様々な処理を分散させることが可能になっている [4]. 負荷分散という基本機能を用いたロードバランサ製品が登場したのは, 1996 年のことだ．その後, まもなくしてインターネットをビジネスの主とした企業が多く誕生した．2004 年には, インターネットビジネスが根付いてきており web サイトを通じて大きな売上を稼ぐ企業が増加してきたことでサイトのレスポンスタイムがそのまま売上に直結するようになった．また, それと同時に可用性も求められるようになりどこかのサーバがダウンしても他のサーバで補えるようにすることも重要な要素の一つとなった．しかし, ロードバランサは非常に高価なものである．大手の企業であれば導入することは可能だろうが, 新しく企業を立ち上げた際やインターネットビジネスが中心ではない中小企業では, 導入することが困難である．ロードバランシングには, スタティックとダイナミックの 2 種類が存在している [5]. スタティックなロードバランシングは, 事前に定義されている処理能力, メモリとストレージ容量, 通信性能に基づいて負

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0118004

荷の割当て, トラフィックを変化させる分散方式であり一般的には動的な変化を考慮していない [6]. スタティックなものは安定した環境で, 高い生産性を実現することが可能だが柔軟性がなく対応できないという欠点がある [6]. ダイナミックなロードバランシングは, 負荷分散対象のサーバの状態によって負荷の割当て, トラフィックを変化させる分散方式になっている [7]. ダイナミックなものは柔軟性が高く臨機応変な対応が可能. しかし, 環境によって大きく左右されるため非効率でオーバヘッドが発生してしまう. これによりパフォーマンスの低下を引き起こす事がある [6].

1.2 スタティックな負荷分散方式について

スタティックな分散方式に,DNS round robin と呼ばれる DNS サーバを用いて 1 つのドメイン名に複数の IP を割り当てて順番に問い合わせをして応答するという負荷分散の手法が存在する [8]. ユーザは, サーバにアクセスをしたいと思ったら, まず DNS にリプライを送ることで, DNS は登録されているサーバのアクセス方法をユーザに DNS リクエストとして返す. ユーザは, それをもとにサーバにアクセスすることが可能. 図 1 に DNS ラウンドロビンの概要を示す.

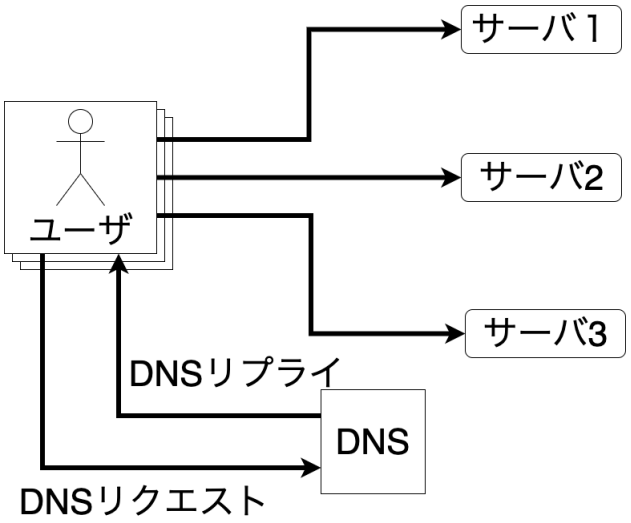

図 1 DNS ラウンドロビン

1.3 ダイナミックな負荷分散方式について

ダイナミックな分散方式に, Dynamic Ratio と呼ばれるものがある. これは, 動的比率を用いるものである. CPU やメモリの使用率が低いサーバに転送する負荷分散の方式だ.[9] 基本的な動作は先程述べた DNS ラウンドロビンと変わらないが, サーバに比率をつけることで, 順番に負荷を割り当てていたところを次の順番のサーバに負荷がかかっているときは比率の大きいものに次の処理を割り振ることで効率的な負荷分散を実現している. 図 2 に Dynamic Ratio の概要を示す.

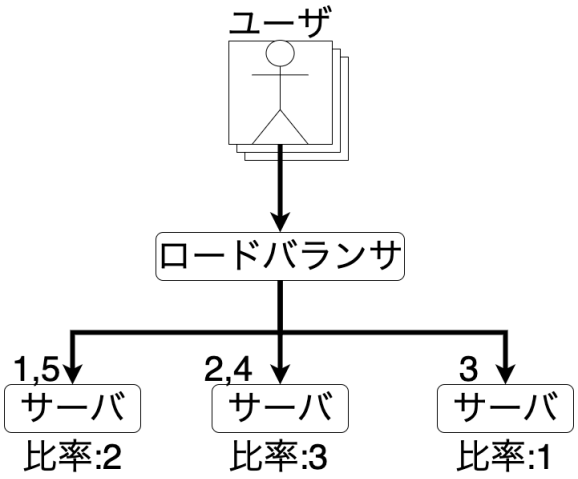

図 2 Dymic ratio

1.4 課題

DNS ラウンドロビンでは負荷が偏るという問題点が存在する [8]. 登録されているサーバに上から順にリクエストの割り振りを行うためヘビーユーザからのアクセスがあった場合割り当てられたサーバは他のサーバに比べ負荷が高い状態になることから偏りが起きている. 他にも, DNS ラウンドロビンには分散先サーバの障害が検知できないという問題点も存在する. DNS ラウンドロビンでは, 分散先サーバの状態が取得できないためこのような問題が存在する. 他のロードバランサよりも安価に導入できることから負荷の偏りを減らすことができれば高価なロードバランサを導入できない企業においても効果が見込める. 図 3 に, 課題の負荷の偏りを図式化したものを表す. 赤色になっている部分が負荷が大きい状態のサーバとなっている. 青色は負荷があまりかかっていない状態のサーバになっている.

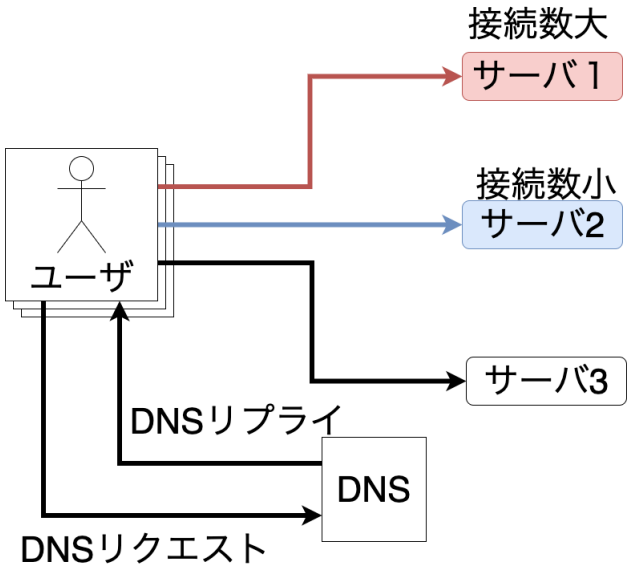

図 3 課題

2. 関連研究

Radojevic は CLBDM と呼ばれるアルゴリズムの提案をした [10]. これは DNS ラウンドロビンの改良でアプリケーション層でのセッションスイッチングを実現している. この研究では, 接続時間を計算しその時間がしきい値を超えると接続を終了するというものとなっている. 終了した場合は, タスクを別なノードに転送する. そうすることで, 負荷のばらつきを軽減されている. しかし, 特定の状況下でしか起こらない負荷の偏りのため一般的に起きる問題ではないとされている. さらに, 入力を取るのではなくサーバ負荷をネットワークパラメータのみでフォワーディングを行えていないことが課題とされている [8]. 重み付き最小接続スケジューリングのアルゴリズムを用いて行った研究では, サーバー負荷アルゴリズムは, ラウンドロビンアルゴリズムや接続数アルゴリズムに比べて CPU 使用率が高く, サーバー負荷アルゴリズムを使用した場合,8 つの HTTP 間で負荷に大きなばらつきが生じるという課題がある [11]. ミラーサーバシステムに用いる DNS ベースの分散方法の研究では, 各ミラーサーバ候補の「適応率」を取得する. この「適応率」は, クライアントがミラーサーバを選択する際の指標とされている. 通常の DNS ラウンドロビンシステムよりも平均転送速度が向上することが確認されている [12]. しかし, この研究では転送速度は改善されているが偏りは改善されていない. XU Zongyu らの研究 [13] では, サーキュラーリストと最後に選択されたサーバへのポインタを用いてディスパッチの決定を行う. リクエストが到着したとき, 各サーバの負荷の総和を計算し, i 番目のサーバが最大でなければ, 新しいリクエストをそのサーバに割り当てる. しかし, i 番目のサーバが最大でなければ, 新しいリクエストは i+1 番目のサーバに割り当てられるというアルゴリズムだ. 評価では, WEB サーバノードはすべて同じ性能で同じコンテンツを持ち, 動的・静的ページの両方を処理できることを仮定している. 通常のアルゴリズムの場合の最大負荷は, 558.34 となり最小負荷は, 477.30 となっている. 負荷の範囲は, 81.04. 負荷分散は 606.43 となっている. この研究でのアルゴリズムを使用した場合は, 最大負荷が 525.26 となり最小負荷 502.34 となっている. 負荷範囲は 58.13 を削減し, 28.27 負荷の軽減はできているが台数が少ないので台数を増やしたらどのような結果になるかわからない.

3. 提案

本研究では, 分散先サーバの送受信量をもとに登録されている A レコードの順番を並び替えるプログラムを用いた DNS ラウンドロビンの手法を提案する. 図 4 に提案の概要を示す.

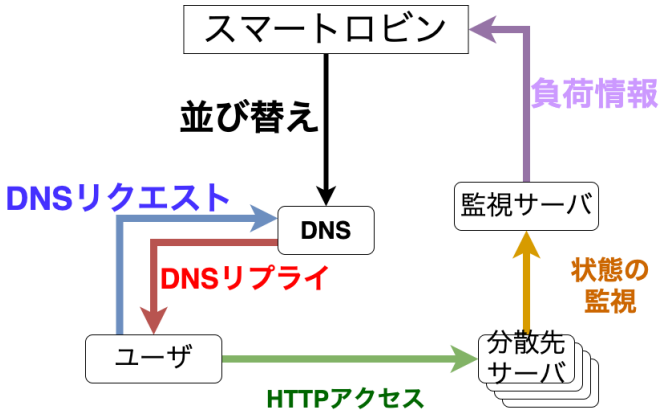

図 4 DNS ラウンドロビン

提案では, DNS サーバとは別に監視サーバを用意して監視サーバが分散先サーバのネットワークの送受信量を取得し, 負荷の低い順にレコードの中身を書き換えることで負荷の偏りを軽減する. ユーザはまずサイトにアクセスをしたいと思ったら DNS サーバにアクセス先の IP アドレスを取得するために DNS リクエストを送る. DNS が登録されているゾーン情報を確認をする. その際に, 監視サーバがゾーン情報に登録されているサーバのネットワークの送受信量を DNS サーバ内にあるソフトであるスマートロビンに送ることでスマートロビンは, ゾーン情報の書き換えを行う. ユーザから送られた DNS リクエストを受け取り DNS はゾーン情報を参照し DNS リプライを返す. そのリプライは, 登録されているゾーン情報を上から返す. ユーザはリプライに書いてある IP アドレスにアクセスすることで負荷が分散できる. 今回の提案では,DNS ラウンドロビンの静的な負荷分散では負荷の状態を考えずに上から順に分散していることから偏りが生じていたがゾーン情報を書き換えることで上から順に分散をしても偏ることはない.

3.1 設計

本研究では, 図 5 にあるように DNS サーバに登録されている A レコードの情報を分散先サーバから得たネットワークの送受信量をもとに負荷が低い順に並べ替える. 登録されているレコードの情報を 6 分毎に更新し, 分散先サーバである HTTP サーバの送受信量のデータを取得する. スマートロビンは, Zabbix から得た送受信量をもとに DNS の持つゾーン情報を書き換える. データに登録されている, IP アドレスは内部 IP アドレスを参照しており外部からのアクセスはできないようになっている.

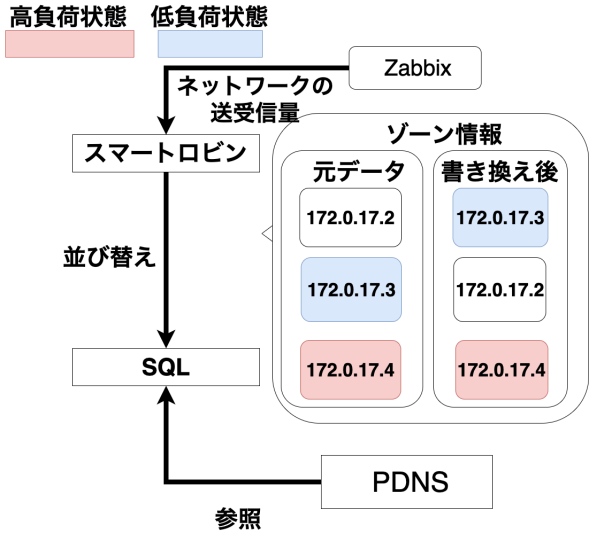

図 5　ソフトウェア図

図 6 にネットワークの送受信が大きいものをレコードの最初に置くフローチャートを示す.

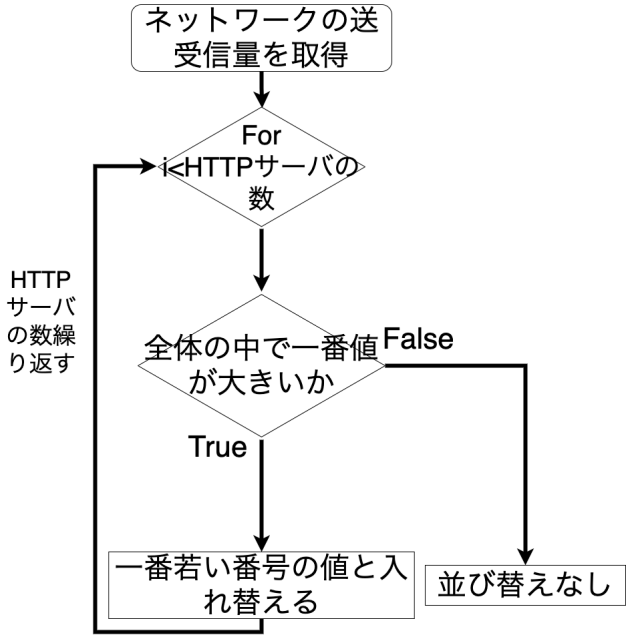

図 6　一番大きい値を見つけるフローチャート

図 7 にネットワークの送受信量が小さいものをレコードの最後におくプログラムのフローチャートを示す. ここで言っている大きな番号というのは, 登録したサーバの順番だ.

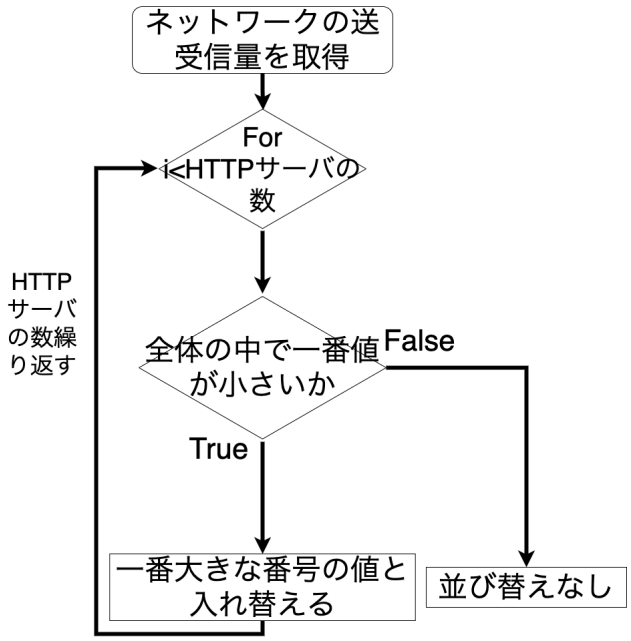

図 7　一番小さい値を見つけるフローチャート

図 8 にネットワークの送受信量を DNS サーバが取得する方法を示す.

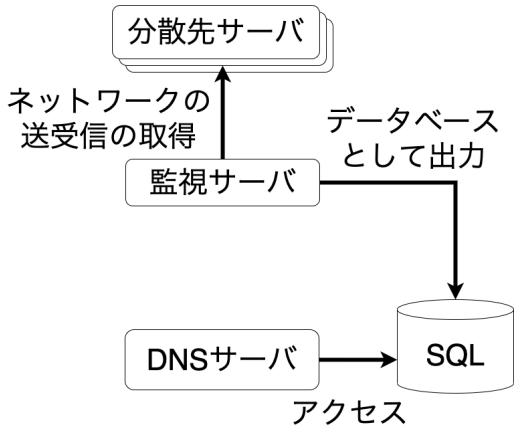

図 8　負荷情報取得する方法

4.　実装と実験環境

4.1　実装

本研究で使用した DNS サーバ, DNS, 監視サーバ, 分散先サーバについて記す. DNS サーバは, cent OS7 上で docker を動かすことで DNS を動作させている. DNS は PowerDNS Community, Bert Hubert より提供されている PowerDNS[14] を docker 上で動かしている. 監視サーバは, Zabbix 社より提供されている Zabbix[15] を docker 上で動かしている. 分散先サーバは, DNS サーバ内で docker 上で動かしている. http サーバの中には, 4K 動画を置くことで送受信量を多くすることで意図的に負荷の偏りを起こしている.

4.2 実験環境

図 9 に実験の環境を示す.

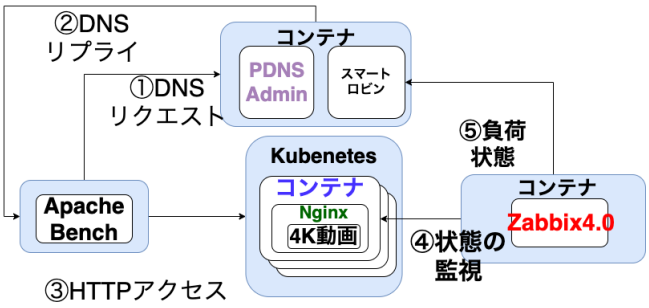

図 9 構成図

DNS サーバで使用した cent OS7 は安定性を重視して選択した. DNS で使用した PowerDNS は, PowerDNSAdmin を使用することでバックエンドにある mysql を web インターフェスで管理できることから選択した. また, 同じ DNS である BIND よりも高速に動作することも理由として挙げられる. 監視サーバで使用した Zabbix は, 収集したデータをすべてデータベースに保存できることから選択した. Zabbix を監視サーバとして別のマシンとして配置した理由としては, DNS サーバが落ちた際でも監視を続けることができ, ネットワークの送受信量の送信を行っているため送信ができなかった際に DNS サーバに問題があると判断できるため配置した.

5. 評価と分析

通常の DNS ラウンドロビンとスマートロビンを組み込んだ DNS ラウンドロビンを行った際に起きる負荷の偏りを Container にかかったネットワークの送受信量をグラフ化して示す.

図 10 にあるように送受信量を分布で表す.

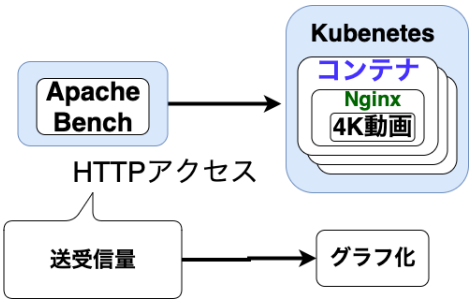

図 10 評価の方法

CPU に負荷をかけて Zabbix で監視したグラフを図 11 に示す.

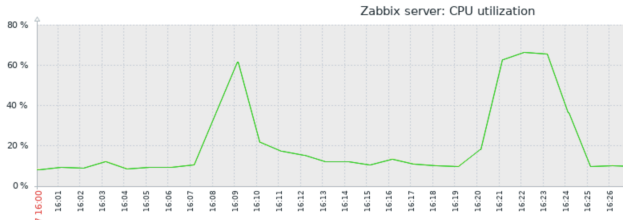

図 11 CPU に負荷をかけた際の監視

6. 議論

今回 docker 上ですべてのシステムを動かしていたが docker 上で動かすとコンテナを落とすことでデータが保存されていないという問題は解決できなかった. 本実験では, 接続台数が少なく実際の環境とは違うため多くの台数で行うとどのような結果になるのかわからない. また, 分散先サーバは同一のスペックを想定しているが実際の環境ではサーバのスペックは異なるため実用的ではないといえる. 本実験では A レコードのみの登録を行っているが, 実際の環境では他にも NS レコードや MX レコードといったレコードが存在している. このことから, A レコード以外の情報が登録されていた際の処理についても考えなければ日常的に使用することは難しいと言える.

7. おわりに

本件研究では, DNS ラウンドロビンを用いて負荷分散をする際に, 負荷が偏ることを課題とした. 負荷を割り振る際に, 分散先サーバの性能や負荷の状態を鑑みずに登録されている順番に割り振ることで負荷が偏る問題が起こる. 負荷の状態を考慮して負荷を割り振る必要がある. 監視サーバを用い負荷の状態を取得することで登録されている順番を入れ替え負荷の低いものを上位に, 負荷の高いものを下位に置くことで負荷を均等にする手法を提案した. 本実験では, ネットワークの送受信量を負荷の状態とした.

参考文献

[1] Venkataraman, S.: The distinctive domain of entrepreneurship research, *Seminal ideas for the next twenty-five years of advances*, Emerald Publishing Limited (2019).

[2] Mirkovic, J. and Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms, *ACM SIGCOMM Computer Communication Review*, Vol. 34, No. 2, pp. 39–53 (2004).

[3] Shang, Z., Chen, W., Ma, Q. and Wu, B.: Design and implementation of server cluster dynamic load balancing based on OpenFlow, *2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013)*, IEEE, pp. 691–697 (2013).

[4] Lin, F. C. H. and Keller, R. M.: The gradient model load balancing method, *IEEE transactions on software engineering*, No. 1, pp. 32–38 (1987).

[5] Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors, *Journal of parallel and distributed computing*, Vol. 7, No. 2, pp. 279–301 (1989).
[6] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M. and Al-Jaroodi, J.: A survey of load balancing in cloud computing: Challenges and algorithms, *2012 second symposium on network cloud computing and applications*, IEEE, pp. 137–142 (2012).
[7] Cardellini, V., Colajanni, M. and Yu, P. S.: Dynamic load balancing on web-server systems, *IEEE Internet computing*, Vol. 3, No. 3, pp. 28–39 (1999).
[8] Sotomayor, B., Montero, R. S., Llorente, I. M. and Foster, I.: Virtual infrastructure management in private and hybrid clouds, *IEEE Internet computing*, Vol. 13, No. 5, pp. 14–22 (2009).
[9] Harchol-Balter, M. and Downey, A. B.: Exploiting process lifetime distributions for dynamic load balancing, *ACM Transactions on Computer Systems (TOCS)*, Vol. 15, No. 3, pp. 253–285 (1997).
[10] Radojević, B. and Žagar, M.: Analysis of issues with load balancing algorithms in hosted (cloud) environments, *2011 proceedings of the 34th international convention MIPRO*, IEEE, pp. 416–420 (2011).
[11] Mustafa, M. and Ibrahim, A. M.: Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency, *vol*, Vol. 1, pp. 25–29 (2017).
[12] Yokota, H., Kimura, S. and Ebihara, Y.: A proposal of DNS-based adaptive load balancing method for mirror server systems and its implementation, *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.*, Vol. 2, IEEE, pp. 208–213 (2004).
[13] Bryhni, H., Klovning, E. and Kure, O.: A comparison of load balancing techniques for scalable web servers, *IEEE network*, Vol. 14, No. 4, pp. 58–64 (2000).
[14] Klein, A.: PowerDNS recursor DNS cache poisoning, *Trusteer, February-March* (2008).
[15] Olups, R.: *Zabbix 1.8 network monitoring*, Packt Publishing Ltd (2010).

