L-014

分散 Web システムにおける 異なる性能のキャッシュサーバを管理する機構の開発

Development of Cache Server Management Mechanism for Heterogeneous Specification in Distributed Web System

松田正也 [1] Masaya Matsuda
最所圭三 [1] Keizo Saisho

1 はじめに

Web サーバの負荷量に応じてクラウド上の仮想キャッシュサーバ (VC サーバ) 数を動的に増減させることで, 応答性を確保しつつ運用コストを低減する分散 Web システムの実現を目指している．本研究では，コストや性能を考慮して負荷量に応じて VC サーバを起動・停止し，それらにリクエストを振り分ける機能を持つ拡張ロードバランサを開発している [1]. 先行研究では，単一性能の VC サーバを用いた評価実験しか行っていなかった [2]．しかし，性能に応じて処理できるリクエスト量などが変化するため，異なる性能の VC サーバを利用できるように改良する必要がある．本稿では，実際のクラウド上で性能の異なる仮想サーバについて高負荷・低負荷時のふるまいを調査し，その結果にもとづいて開発した動的重み付けアルゴリズムおよびその評価について述べる．さらに，動的重み付けアルゴリズムを用いたオートスケールアルゴリズムの検討を行い，今後の展望について述べる．

2 分散 Web システムの概要

本研究で開発している分散 Web システムの構成を図 1 に示す．VC サーバはオリジンサーバのリソースのキャッシュを提供する．負荷量の監視及びキャッシュサーバの増減を拡張プログラムが行い, 実際のリクエストの振り分けはロードバランサの機能を用いて行う. 拡張プログラムの機能を以下に示す.

A 負荷監視機能: サーバの負荷量を監視する.
B キャッシュサーバ管理機能: 負荷量に応じて VC サーバの重み付け変更，起動・停止を行う.
C 振分先設定機能: アクセスの振り分け先を設定する.

本稿ではキャッシュサーバ管理機能について述べる.

3 クラウド上の仮想サーバの性能調査

サーバの性能毎の高負荷状態や低負荷状態と判断する基準の調査を行った．今回の調査では，提供する Web サービスとして DokuWiki を用いた．仮想サーバには，Microsoft Azure の仮想マシン DS1(1 コア 3.5GB), DS2_V2(2 コア 7GB), DS11_V2(2 コア 14GB), DS3_V2(4 コア 14GB), DS12_V2(4 コア 28GB) を用いた．仮想サーバにアクセスするクライアントには，D1(1 コア 3.5GB) を用い，Apache Bench

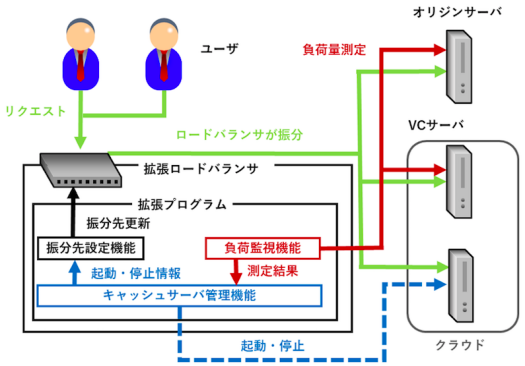

図 1 システムの構成

で負荷をかけた．仮想サーバ，クライアントともに西日本リージョンで構築し，同一プライベートネットワーク内に配置した．同時リクエスト数を 100 から 2,000 まで徐々に増やして，負荷量として Apache の稼働率, CPU 使用率，メモリ使用量，スループット，応答時間を計測した．Apache の稼働率は最大同時サービス数に対する現在の同時サービス数の割合を指す．予備実験により最大同時サービス数が 200 以上ではスループットにほぼ変化がなかったため，実験では全ての仮想サーバで 200 に設定した.

図 2 に DS3_V2 の実験結果を示す．負荷量は定期的に仮想サーバに問い合わせることで測定している．0 秒 ~150 秒では，スループットが非常に高い状態と非常に低い状態を繰り返している．これはクライアントからのアクセスを一気に処理することでスループットが増加し，その後アクセスが少なくなるためと考えられる．スループットが高い箇所では CPU 使用率も高くなっている．このことから，スループットの上下が激しい時は余力のある状態と考えられる．また，170 秒以降では稼働率がほぼ上限に達しているのに対し，CPU 使用率やスループットはあまり増加していない．メモリ使用率と負荷状況の関連性は見い出せなかった．他の性能のサーバでも同様の傾向が見られた．このことから以下の仮説を立てた.

1. 低負荷状態ではスループットが非常に高い状態と非常に低い状態が交互に発生する.
2. 高負荷状態でも CPU 使用率は上下する.
3. 高負荷状態では常に稼働率は上限に達している.

[1] 香川大学

Copyright © 2017 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

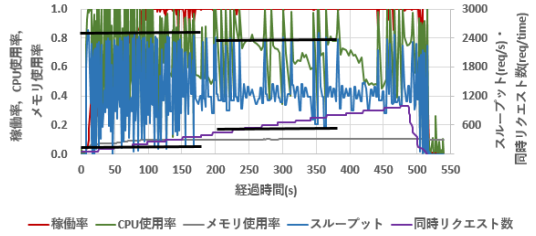

図 2　負荷条件調査実験結果

4. リクエスト数が少ない状態では，稼働率も CPU 使用率も低い．

表 1 に各性能における平均スループットを示す．

表 1　各性能における平均スループット

性能	DS1	DS2_V2	DS11_V2	DS3_V2	DS12_V2
平均	249.6	775.6	887.9	1297.8	1632.5

4　動的重み付けアルゴリズム

本節では，負荷量に応じて重み付けを変更する動的重み付けアルゴリズムを提案する．VC サーバには重み付けがされており，重みに比例した数のリクエストを振り分ける．このアルゴリズムでは，s 秒間の負荷状態を表す負荷変数 l を導入する．l は以下に示す状態に応じて高負荷状態では増加，低負荷状態では減少させる．

1. 高負荷状態
   (a) $OR >= 0.8$
   (b) $Th_{AVG} \times 0.8 < Th < Th_{AVG} \times 1.3$ かつ $CR >= 0.9$
2. 低負荷状態
   (a) $OR <= 0.3$ かつ $CR <= 0.5$
   (b) $Th < Th_{AVG}$ かつ $CR <= 0.4$

ここで，Th はスループット，Th_{AVG} は VC サーバの性能毎の平均スループット，OR は Apache の稼働率，CR は CPU の使用率を表す．それぞれ (a) と (b) の条件があるが，その条件毎に l を ± 1 の増減を行うため最大で ± 2 の増減がありうる．l が v 以上なら高負荷と判定し，$-v$ 以下なら低負荷と判定する．高負荷状態の VC サーバと低負荷状態の VC サーバが混在する場合は，高負荷状態のサーバの重みを w 減少し，低負荷状態のサーバの重みを w 増加する．w は振り分けの割合の変動量である．

性能毎に重みを固定で設定して行った実験 (実験 1)，性能毎に重みを設定した状態から重みを変更する実験 (実験 2)，均一な重みの状態から開始し重みを変更する実験 (実験 3) を行った．性能毎の重みは表 1 にもとづき，表 2 に示す値に設定した．実験 2，3 では，s を 10，v を 5，w を 5 に設定した．ロードバランサとして haproxy[3] を用い，アルゴリズムは重み付き最小接続数方式を使った．

表 3 にそれぞれの実験における平均スループットを示す．実験 2 が最もスループットが高くなったが，実験 1 との差は僅かであった．今回の実験では，重みを変更した後に次の変更を行うまでのマージンを設けなかったため，連続して重みの変更が発生し，低負荷だった VC サーバに一気に負荷が集中しスループットが低下したため期待したほど効果が無かった．また，実験 2，3 から事前に VC サーバの性能に合わせて重み付けをしておいた方が，スループットが高くなることが確認できた．

表 2　各性能毎の重み

性能	DS1	DS2_V2	DS11_V2	DS3_V2	DS12_V2
重み	10	31	36	52	65

表 3　各性能における平均スループット

実験種別	実験 1	実験 2	実験 3
平均	5514.4	5596.9	5105.4

5　異なる性能のサーバのためのオートスケールアルゴリズムの検討

動的重み付けアルゴリズムを活用したオートスケールアルゴリズムの検討を行った．スケールアウトアルゴリズムの動作手順を重み 1 当たりの性能 F_1，稼働中の VC サーバ群の処理限界負荷量 L_L，新しいサーバを起動するまでにかかる時間 S，S 後の負荷量 L_S，スケールアウトの閾値 Th_{high}，L_S を処理に必要な性能 P, ボトルネックの負荷の性質 T と，その T のコストが最も低くなるベンダー V_T を用いて以下に示す．なお T として CPU 使用率，データ通信量，ネットワーク I/O をなどを考えている．

1. 動的重み付けアルゴリズムで F_1 を測定
2. 負荷測定機能で L_L を測定
3. L_S を一定間隔で測定
4. 負荷状況から T を選択
5. $L = L_S/L_L >= Th_{high}$ を満たすとき以下を実行
   (a) V_T を選択
   (b) $P = (L - Th_{high}) \times L_L/F_1$ を計算
   (c) V_T の中から P 以上の重みを持つ VC サーバを起動する

スケールインアルゴリズムの動作手順を現在の負荷量 L_N，スケールインの閾値 Th_{low}，スケールイン後の負荷量 L_{LI} を用いて以下に示す．

1. $L_I = L_N/L_L <= Th_{low}$ を満たすとき以下を実行
   (a) $Th_{low} < L_N/L_{LI} < Th_{high}$ を満たす最もコストの大きい VC サーバを停止する．

6　おわりに

異なる性能のサーバを用いる動的重み付けアルゴリズム，それを利用したオートスケールアルゴリズムについて述べた．今後は提案したオートスケールアルゴリズムを評価し改良していく．今回の実験では CPU がボトルネックになっていたが，ディスク I/O やネットワーク I/O などがボトルネックになり得るサービスで評価を行う予定である．

参考文献

[1] 堀内晨彦, 最所圭三, ”クラウドに適した Web システムにおけるキャッシュサーバの負荷監視および負荷分散”, 信学技報, vol.114, no110, IN2014-20, pp.79-84, 2014
[2] 松田正也, 最所圭三, ”分散 Web システムにおけるスケーリングアルゴリズムの改良と評価”, 情報処理学会 第 79 回全国大会, 7T-03, p3-271, 2017.03
[3] haproxy, http://www.haproxy.org/

Copyright © 2017 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.