L-020

# 分散 Web システムにおけるキャッシュサーバ管理機能の試作と評価

Prototyping and Evaluation of an Cache Server Management Function for Distributed Web System

堀内晨彦＊，最所圭三＊

Akihiko Horiuchi，Keizo Saisho

## 1　はじめに

我々は，クラウド環境において負荷量に応じて動的に仮想キャッシュサーバ数を増減させることで，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，図 1 に示すように，キャッシュサーバと大元の Web サーバ (オリジンサーバ) の負荷状況を監視し，負荷量に応じてクラウド環境で提供される仮想キャッシュサーバを起動・停止させ，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．拡張ロードバランサは，稼働中のサーバの平均負荷量が上限を超えると仮想キャッシュサーバを起動し (スケールアウト)，下限を下回ると停止させている (スケールイン)．

これまで研究では，サーバの負荷監視機能および振分先設定機能の有効性を確認することを優先し，仮想キャッシュサーバを最大数まで起動した状態で，負荷量を変化させたときの振り分け対象になる仮想キャッシュサーバ数の変化，クライアントからのアクセス時間を評価する実験を行い，その有効性を確認している [1]．今回，キャッシュサーバ管理機能の実装を行い，その評価を行った．

本稿では，オートスケールに合わせて仮想キャッシュサーバを起動・停止させるための「キャッシュサーバ管理機能」およびその試作と評価について述べる．

## 2　分散 Web システムの概要

図 1 に示すように，拡張ロードバランサは次の 3 つの機能で構成されている．

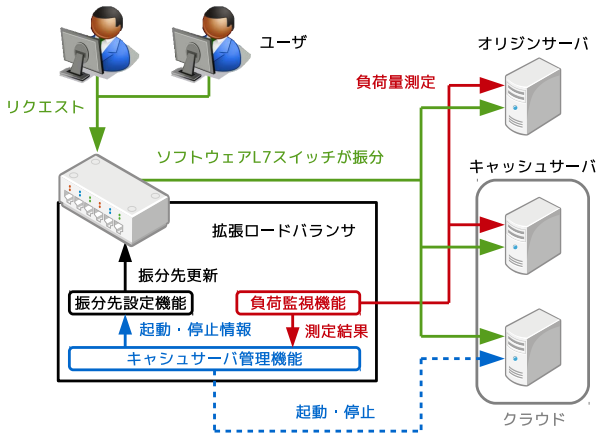


図 1　分散 Web システムの概要

＊ 香川大学，Kagawa University

- **負荷監視機能**
  オリジンサーバや仮想キャッシュサーバ群の負荷量を定期的に監視する機能である．Web サーバには Apache 2.4 を用いており，その最大処理数に対する現在の処理数の割合 (稼働率) を測定し，現在振分中の Web サーバの稼働率の平均値 (平均稼働率) を負荷量として用いる．
- **キャッシュサーバ管理機能**
  負荷監視機能が測定した負荷量に応じて必要な仮想キャッシュサーバの台数を決定し，それらの起動・停止を行う機能である．仮想キャッシュサーバの起動・停止には，OpenStack[3] などのクラウド基盤が提供する API を用いる予定である．
- **振分先更新機能**
  仮想キャッシュサーバの起動・停止に合わせて，ロードバランサの振分先の設定を更新する機能である．ロードバランサには Linux カーネル内に実装されている IPVS[2] を用いている．Web サーバの IP アドレスに対する振分の重みを IPVS の設定コマンドを用いて変更することで実現する．

## 3　キャッシュサーバ管理機能の設計および試作

本節では，前節で述べたキャッシュサーバ管理機能の設計および試作について述べる．キャッシュサーバ管理機能は，以下の手順で仮想キャッシュサーバの起動・停止を行う．

- **スケールアウト**
  負荷量がスケールアウトの閾値 ( $Th_{high}$ ) を上回った場合，新たな仮想キャッシュサーバを起動する．仮想キャッシュサーバがサービスを開始するまで待機した後，起動したことを振分先更新機能に通知し，その IP アドレスをロードバランサの振分先に追加する．
- **スケールイン**
  負荷量がスケールインの閾値 ( $Th_{low}$ ) を下回った場合，最後に起動した仮想キャッシュサーバを停止する．停止することを振分先更新機能に通知し，その IP アドレスをロードバランサの振分先から削除した後，仮想キャッシュサーバを停止する．

現在の分散 Web システムでは Linux カーネルに標準的に搭載されている KVM ハイパーバイザを用いており，クラウド基盤の API を用いることができない．そこで，KVM を始めとする仮想化基盤を操作するための共通 API を提供する libvirt[4] ライブラリを用いて，キャッシュサーバ管理機能を試作することにした．試作したキャッシュサーバ管理機能の概要図を図 2 に示す．libvirt は各プログラミング言語向けに

提供されている API ライブラリと，ハイパーバイザ上で API からの操作を受けるデーモン (libvirtd) で構成されている．

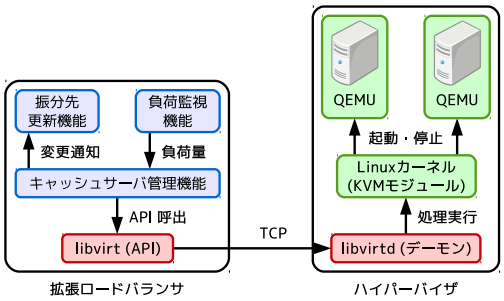


図 2　キャッシュサーバ管理機能の概要

キャッシュサーバ管理機能が libvirt を呼び出す手順は以下の通りである．

1. `NewVirConnection` でハイパーバイザの IP アドレスを指定し，libvirtd と TCP のコネクションを確立する．
2. `LookupDomainByName` で仮想キャッシュサーバのドメイン名を指定し，操作する仮想マシンを特定する．
3. 仮想キャッシュサーバを起動する場合は `Create`，停止する場合は `Shutdown` を実行する．
4. `CloseConnection` で libvirtd との TCP コネクションを切断する．

なお，ハイパーバイザの IP アドレスやハイパーバイザが管理する仮想キャッシュサーバのドメイン名は，図 3 に示すような拡張ロードバランサの設定として読み込んでいる．試作したキャッシュサーバ管理機能は，複数のハイパーバイザ上の仮想キャッシュサーバを用いることができる．

```
{
  "hvs": [                              // Hypervisors
    {
      "host": "192.168.11.20",          // HV's IP Address
      "vms": [                          // Virtual Machines
        {
          "name": "cache-server-1",     // VM's Domain Name
          "host": "192.168.11.21"       // VM's IP Address
        },
        {
          "name": "cache-server-2",     // VM's Domain Name
          "host": "192.168.11.22"       // VM's IP Address
        }
      ]
    }
  ]
}
```

図 3　拡張ロードバランサの設定例 (JSON 形式)

## 4　仮想マシンの起動・停止時間の調査

キャッシュサーバ管理機能が起動・停止を実行してから，仮想キャッシュサーバが実際にサービスを開始・終了するまでにかかる時間を調査した．仮想キャッシュサーバには CPU を 1 コア，メモリを 1GB 割り当て，OS は Ubuntu Server 14.04 を用いた．libvirt を用いて起動・停止の操作を実行してからアクセスが可能・不可能になるまでの時間を 10 回測定した結果を表 1 に示す．

表 1　仮想キャッシュサーバの起動・停止時間

| | 起動〜アクセス可能 | 停止〜アクセス不可能 |
|---|---|---|
| 1 | 9.724 | 3.004 |
| 2 | 9.726 | 3.031 |
| 3 | 9.512 | 3.049 |
| 4 | 9.556 | 3.003 |
| 5 | 9.265 | 3.029 |
| 6 | 9.681 | 3.067 |
| 7 | 9.432 | 3.021 |
| 8 | 9.494 | 299 |
| 9 | 9.440 | 296 |
| 10 | 9.767 | 3.050 |
| 平均 | 9.767 | 3.025 |

表 1 より，起動してからアクセス可能になるまでに約 10 秒，停止してからアクセス不可能になるまで約 3 秒かかっていることが分かる．しかし，実際に分散 Web システムに対してベンチマークツールである Apache Bench を用いて大量のアクセスをかけ，これらの時間で振分を開始・終了する予備実験を行ったところ，それぞれ 30 秒程度の時間を置かないとアクセスが中断する問題が発生した．これは，起動時にアクセスが可能になっても，他の多くのサービスプログラムを立ち上げ中であり，その影響を受けたと思われる．また，停止時には振分を終了した仮想キャッシュサーバがレスポンスを返却途中であると考えられる．そのため以降の評価実験では，仮想キャッシュサーバを起動してから振分を開始までの時間と，振分を終了してから仮想キャッシュサーバを停止するまでの時間を，どちらも 30 秒とした．

## 5　キャッシュサーバ管理機能の評価

試作したキャッシュサーバ管理機能を組み込んだ拡張ロードバランサの評価するための実験を行った．図 4 に示すように，拡張ロードバランサ 1 台，仮想キャッシュサーバ 9 台，クライアント 9 台を用いた．分散 Web システムに適したキャッシュ機構が開発中のため，仮想キャッシュサーバの代わりにミラーサーバとし，アクセス先には DokuWiki[5] を用いた．各クライアントからは Apache Bench を用いてアクセスを行い，同時アクセス数は 100 とする (最大同時アクセス数 900)．

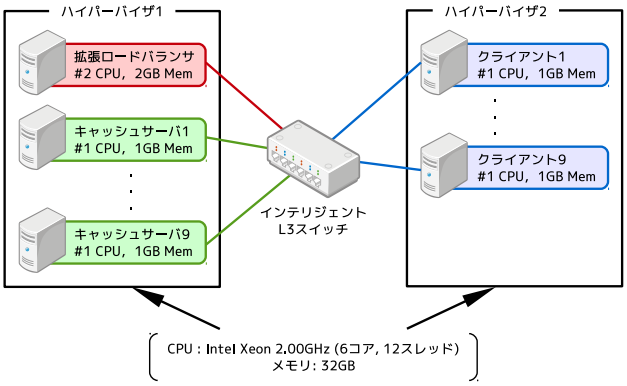


図 4　分散 Web システムの実験環境

実験手順は下記の通りである．ロードバランサに対するアクセスを段階的に増減させ，その際の負荷量や仮想キャッシュサーバの稼動台数を調査する．

1. ロードバランサに対するアクセスが無い状態で開始
2. 30 秒ごとにアクセスするクライアントを 1 台追加
3. 全クライアントがアクセス中の状態で 60 秒維持
4. 30 秒ごとにアクセスするクライアントを 1 台削除
5. ロードバランサに対するアクセスが無くなると終了

キャッシュサーバ管理機能を無効にし振分先の切替のみを行った場合の実験結果を図 5 に，キャッシュサーバ管理機能を有効にした場合の実験結果を図 6 に示す．図 5 では，同時アクセス数の増加に応じて平均稼働率も上昇し，スケールアウトの閾値である $Th_{high} = 0.8$ を超えると即座に仮想キャッシュサーバが振分先に追加され，負荷の解消により平均稼働率が下降している．ここでは仮想キャッシュサーバは常に起動しており，起動から振分開始までにかかる時間を考慮しないため，平均稼働率が 1.0 に達することはなかった．しかし，図 6 ではキャッシュサーバ管理機能を用いて仮想キャッシュサーバの起動を行っているため，平均稼働率が $Th_{high}$ を超えてから負荷が解消するまでに時間がかかっていることが分かる．そのため平均稼働率が 1.0 に達している部分もあり，最大アクセス時の稼動台数も図 5 の 7 台と比較してまだ 5 台しか起動していない．

次に，キャッシュサーバ管理機能を有効にした場合の各 Web サーバの稼働率を図 7 に示す．図 7 より，仮想キャッシュサーバの起動から振分開始までに時間がかかることによって，既に振分中の Web サーバにアクセスが集中し負荷がかかっていることが分かる．さらに，新たな仮想キャッシュサーバへの振分を開始しても，既に振り分けられたアクセスを処理し切るまでは Web サーバの負荷が解消されないという問題もある．これはクライアントの応答時間の増加など，サービスの品質を低下させる可能性があり，解決する必要がある．

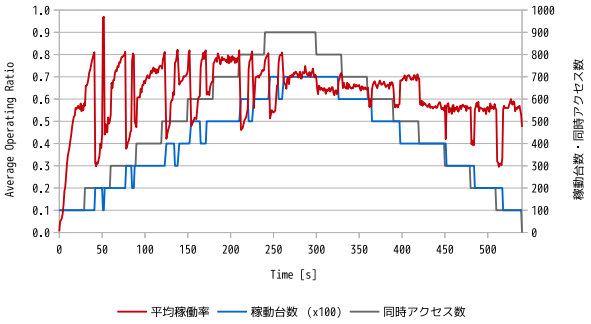


図 5　キャッシュサーバ管理機能無効時の実験結果

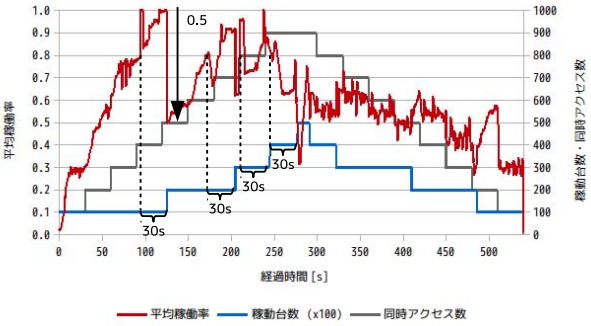


図 6　キャッシュサーバ管理機能有効時の実験結果

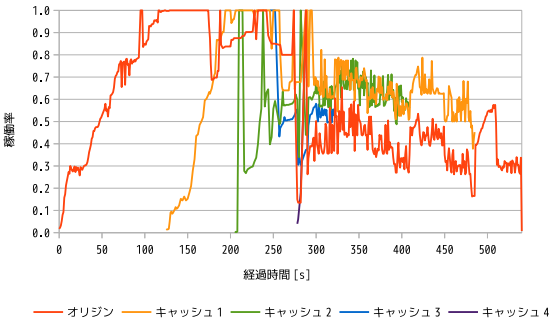


図 7　各 Web サーバの平均稼働率

## 6　増加率によるキャッシュサーバ複数台起動

前節で述べた仮想キャッシュサーバの起動時間による負荷を解消する方法を考える．例えば，アクセス数の増加を予測して事前に仮想キャッシュサーバを起動しておくことで，負荷量の増加時に即座に振分を開始する方法が考えられる．しかし，アクセス数の増加傾向を予測することが困難であったり，予測した通りに負荷量が増加せずに起動した仮想キャッシュサーバが無駄になったりという問題が考えられる．そこで，負荷量の増加率に基づいてスケールアウト時に起動する仮想キャッシュサーバの台数を決定することで，この問題を解決できると考えた．

### 6.1　増加率による起動台数決定アルゴリズム

システム全体の負荷量の増加割合を「増加率 (IR: Increase Ratio)」とする．負荷監視機能では，1 秒間隔で各 Web サーバの稼働率 (OR: Operating Ratio) を測定しており，移動平均を求めるために最新 10 個の平均稼働率 (AVGOR: Average OR) を保存している．そこで，保存した平均稼働率を用いて増加率を求めることにした．9 秒前の平均稼働率 $AVGOR_{n-9}$ に対する最新の平均稼働率 $AVGOR_n$ の割合を増加率 $IR$ として，式 1 で求める．

$$IR = AVGOR_{n-9}/AVGOR_n \tag{1}$$

また，増加率に応じて，起動する仮想キャッシュサーバの台数 $N$ は式 2 で求める．増加率が 0 以下の場合でも，平均稼働率が閾値 $Th_{high}$ を超えていればスケールアウトする必要があるため，起動する仮想キャッシュサーバは 1 台とする．増加率が 0 より大きい場合は，増加率を切り上げた値を起動する仮想キャッシュサーバの台数とする．

$$N = \begin{cases} 1 & (\ IR \leq 0\ ) \\ \lceil IR \rceil & \end{cases} \tag{2}$$

### 6.2 複数台起動時の評価

前節で述べた増加率に応じて起動する仮想キャッシュサーバの台数を決定する仕組みをキャッシュサーバ管理機能に実装し，従来の 1 台ずつ起動する場合と比較するための評価実験を行った．実験環境・実験手順は第 5 節の場合と同じである．また，停止する仮想キャッシュサーバの台数は常に 1 台である．

増加率に応じて起動する仮想キャッシュサーバの台数を決定した場合の実験結果を図 8 に示す．100 秒付近と 200 秒付近で，2 台の仮想キャッシュサーバが同時に起動 (振分開始) していることが分かる．図 6 では，最初の仮想キャッシュサーバ起動時の負荷量の減少幅は約 0.5 である．それに対して図 8 では，最初の仮想キャッシュサーバ起動時の負荷量の減少幅は約 0.65 と大きくなっている．仮想キャッシュサーバを複数台同時に起動することで，振分開始時により負荷を解消できていることが分かる．しかし，仮想キャッシュサーバを 2 台起動した直後に 1 台は停止しており，これは負荷量が $Th_{low}$ を下回ったためと思われる．

また，図 8 において，350 秒以降は仮想キャッシュサーバの停止が行われていないことが分かる．現在のキャッシュサーバ管理機能の実装では，起動・停止を 1 台ずつ行うようにしているため，1 度操作に失敗すると以降の操作が行われないといった問題がある．そのため，仮想キャッシュサーバの稼動台数を正確に把握する機能が必要である．

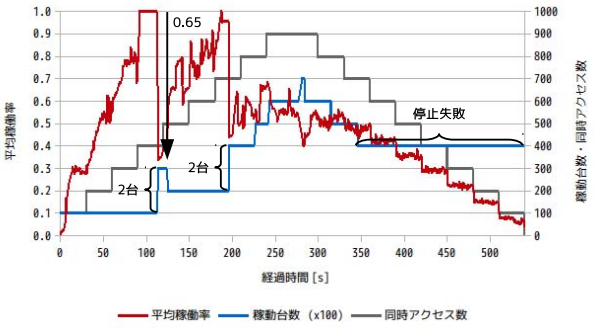


図 8　キャッシュサーバ複数起動時の実験結果

## 7 おわりに

以上，分散 Web システムにおいてオートスケールに合わせて仮想キャッシュサーバの起動・停止を行うキャッシュサーバ管理機能の試作と評価について述べた．仮想化管理用の共通 API を提供する libvirt を用いて機能の試作を行い，負荷量に応じて仮想キャッシュサーバの起動・停止が行えることを確認した．しかし，仮想キャッシュサーバの起動から振分開始までに 30 秒程度かかるため，その間に負荷が生じてしまった．そこで，増加率に応じて起動する仮想キャッシュサーバの台数を決定することにした．仮想キャッシュサーバを複数台同時に起動することで，振分開始時により負荷を解消できることが確認できた．

今後の課題として，次のようなものがある．

- **キャッシュサーバ管理機能の改良**
  仮想キャッシュサーバを起動・停止してから何秒程度待機すれば振分を開始・終了できるのかを調査する．また，仮想キャッシュサーバの稼動台数を正確に把握する機能を実装する．
- **スケールアウトの閾値の評価**
  現在は 0.8 で決め打ちしている $Th_{high}$ を変更した場合の評価を行い，最適な値を調査する．
- **現実に即した実験シナリオ**
  急激にアクセスを増加・減少させた場合の評価を行う．また，実際にサービスを行っている Web サーバのログを用いて実験を行う．
- **ハイパーバイザにおける仮想キャッシュサーバの稼動台数**
  図 5 と図 6 を比較すると，図 5 の方が実験開始直後の負荷量の上昇が急であることが分かる．これはハイパーバイザ上で 9 台全ての Web サーバが稼動していることによって，1 台当たりに割り当てらるマシンパワーが少なくなっていることが原因であると考えられる．そのため，複数台のハイパーバイザ上で Web サーバを稼動させた場合の評価を行う．
- **ロードバランサの振分アルゴリズムの改良**
  新たに仮想キャッシュサーバを起動した場合でも，既に Web サーバに振り分けられたアクセスを再分配することはできない．そこで，振分先のサーバの負荷によっては，仮想キャッシュサーバの起動を待ってから振り分けるなどの改良を行う．
- **プライベートクラウド環境での実験**
  現在の KVM による仮想環境を，OpenStack などを用いたプライベートクラウド環境に移行し実験を行う．

**謝辞** 本研究は JSPS 科研費 25330082 の助成を受けた．

### 参考文献

[1] 堀内晨彦，最所圭三，”移動平均による分散 Web システムにおける振分アルゴリズムの改良”，情報処理学会第 77 回全国大会講演論文集，2014，pp.3-171〜3-172
[2] IPVS Software - Advanced Layer-4 Switching, http://www.linuxvirtualserver.org/software/ipvs.html
[3] OpenStack Open Source Cloud Computing Software, http://www.openstack.org
[4] libvirt: The virtualization API, http://libvirt.org
[5] DokuWiki, https://www.dokuwiki.org/