Vol.2020-DPS-182 No.18
Vol.2020-CSEC-88 No.18
2020/3/12

RTTとState Machine Replicationの通信パターンに基づく広域SMRの応答時間最適なレプリカ配置決定手法とその応用

沼倉 正太[1,a)]　中村 純哉[1,b)]　大村 廉[1,c)]

概要：広域 SMR（State Machine Replication）はサービスの複製であるレプリカを地理的に離して配置することで，サービスの大規模災害に対する耐性を向上させるレプリケーションである．広域 SMR は世界各地にデータセンタ（以下，サイトと呼ぶ）を持つパブリッククラウドを利用することで容易に構築できる．レプリカを配置するサイトの組み合わせは応答性能に大きな影響を与えるため，サービス設計者が望む性能を実現するには数多くのレプリカ配置に対して応答性能を評価する必要がある．しかし，その評価には大変な手間がかかるという問題があった．本研究では，サイト間の RTT（Round Trip Time）を用いて広域 SMR のメッセージ通信を模擬することにより，各レプリカ配置の応答性能を高速かつ高精度で見積もる手法を提案する．この手法は，応答性能を見積もるために実際にレプリケーションを構築する必要がないため，一つのレプリカ配置の見積もりはわずか数秒で完了する．これらの特徴はパブリッククラウドを用いた評価実験によって，実際に確認する．また提案手法は，見積もり方法を変更することで，応答性能だけでなく耐故障性能や運用コスト性能，総合的な性能などのさまざまな評価尺度に拡張できることを示す．

キーワード：State Machine Replication，広域 SMR，レプリカ配置

Latency-optimal Replica Deployment Selection for Geographic State Machine Replication using RTTs and its Communication Patterns and its Application

Abstract: Geographic state machine replication (SMR) is a replication method in which replicas of a general service are located on multiple continents to improve the fault tolerance of the service. Nowadays, Geographic SMR is easily realized using public cloud services; SMR provides extraordinary resilience against catastrophic disasters. If some of the replicas fail, the service can be continued by the replicas in other sites (regions). The locations of replicas have a significant impact on the response time of the replication. Therefore, it is necessary to evaluate the response times of many replica deployments in order to realize the desired performance of a system integrator. However, the evaluation takes a lot of time and effort. In this paper, we propose a method to estimate the response time of each replica deployment quickly and accurately. The estimation is performed by simulating the message communication within SMR using the round trip times between sites. Because the method does not require building a replication for the estimation, so the estimation of one replica deployment completes in only a few seconds. These characteristics are shown by experimental evaluations using a public cloud service. We also show that the proposed method can be extended to various measures other than the response time, such as fault tolerance, operation cost, and overall performance, by changing the estimation method.

Keywords: State Machine Replication，Geographic SMR, replica deployment

[1] 豊橋技術科学大学
Toyohashi University of Technology
[a)] numakura.shota.nj@tut.jp
[b)] junya@imc.tut.ac.jp
[c)] ren@tut.jp

Vol.2020-DPS-182 No.18
Vol.2020-CSEC-88 No.18
2020/3/12

1. はじめに

クライアントサーバモデルで動作するサービスは，悪意を持ったユーザによる攻撃を受けるなど，何らかの原因によって意図しない挙動を示したり停止したりする．このような問題を防ぐためにサービスに耐故障性を備えることが重要となる．*State machine replication* (SMR) [1] はサービスを**レプリカ**と呼ばれる複数のサービスに複製することで耐故障性を向上させるレプリケーション手法である．SMR では**合意**と呼ばれる処理によって全レプリカの状態を一定に保つ．これにより，レプリカの一部が故障した場合でも他のレプリカによってサービスを継続できる．なかでも，レプリカを地理的に離して配置することでサービスの大規模災害に対する耐性を向上させる SMR は**広域** *SMR* と呼ばれ，世界各地にデータセンタ（以下，**サイト**と呼ぶ）を持つパブリッククラウドを利用することで容易に構築できる．

レプリカを配置するサイトの組み合わせは広域 SMR の応答性能に大きな影響を与えるため，サービス設計者が望む性能を実現するには数多くのレプリカ配置に対して応答性能を評価する必要がある．しかし，応答性能の評価には，サーバの用意，SMR の構築，応答性能の測定といった作業を数多くのレプリカ配置に対して行う必要があり，大変な手間がかかるという問題がある．このようなレプリカ配置の決定問題は，データレプリケーションを対象に一部行われているが [2]，SMR を対象とした研究は著者らの知る限りこれまで行われていない．

本論文では，サイト間の RTT（Round Trip Time）を用いて広域 SMR のメッセージ通信を模擬することにより，各レプリカ配置の応答性能を高速かつ高精度で見積もる手法を提案する．この手法では，まずあらかじめレプリカを配置予定のサイト間の RTT を計測する．次に，RTT の半分の時間をサイト間のメッセージ送信にかかる時間として，SMR プロトコルの通信パターンから SMR の合意処理にかかる時間を見積もる．この手法は，応答性能を見積もるために実際にレプリケーションを構築する必要がないため，一つのレプリカ配置の見積もりはわずか数秒で完了する．

評価実験では，一般的なパブリッククラウドである Amazon Elastic Compute Cloud（以降 Amazon EC2 と表記）を用いて，提案手法の見積精度と見積時間を評価する．見積精度の評価は，数千パターンのレプリカ配置について，実際に Amazon EC2 に構築して計測した応答時間と，提案手法による見積もり値を比較することで，提案手法が高精度に見積もりできることを示す．さらに，レプリカ数とレプリカを配置するサイト数が現実的な設定において，数十秒ですべてのレプリカ配置候補の応答性能を見積もれることを示す．

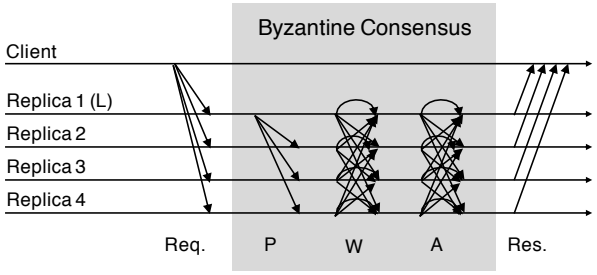

図 1 BFT-SMaRt [3] の通信パターン (L はリーダを，Req, P, W, A, Res はそれぞれ Request, Propose, Write, Accept, Response を意味する)

Fig. 1 The message transmission pattern for BFT-SMaRt [3]. Replica 1 is the leader replica and Req., P, W, A, and Res., indicate Request, Propose, Write, Accept, and Response messages, respectively.

最後に，提案手法は，見積もり方法を変更することで，応答性能だけでなく耐故障性能や運用コスト，総合的な性能などのさまざまな評価尺度に拡張できることを示す．

2. 背景

2.1 State Machine Replication

State Machine Replication (SMR) [1] はサービスを n 台のレプリカに複製することで，高々 f 台のレプリカの故障耐性を実現するレプリケーション手法である．SMR においてレプリケーション対象のサービスは状態機械として定義され，各レプリカは状態機械の複製を持つ．レプリカはクライアントからリクエストを受け取ると，リクエストの実行順序を決定する合意処理をレプリカ間で行い，決定した順序でリクエストを実行することで状態を更新する．その後，レプリカはリクエストの実行結果をレスポンスとしてクライアントに返す．合意によって，たとえ複数のクライアントからのリクエストが通信遅延によってレプリカごとに異なる順序で到着する場合でも，全レプリカのリクエスト実行順序が等しくなるため，各レプリカの状態を一致させることができる．

SMR の合意処理を実現するためのアルゴリズムは SMR プロトコルと呼ばれ，次の 2 つの要求を満たす必要がある．

- *Safety*: 故障していない全レプリカは，クライアントからのリクエストを同じ順序で実行する．
- *Liveness*: クライアントは最終的に送信したリクエストに対するレスポンスを受理する．

これまで多くの SMR プロトコルが提案されている [3–6]．

図 1 に代表的な SMR プロトコルである BFT-SMaRt [3] の合意処理時の通信パターンを示す．クライアント・レプリカ間及びレプリカ間を繋ぐ矢印はメッセージの通信を表しており，一度の合意に複数の通信が必要なことがわかる．特に広域 SMR では一般的な SMR と比較してレプリカ間の通信遅延が大きいため，通信時間がレイテンシの大部分

を占める．

SMR では故障の種類として，レプリカの動作が停止する**クラッシュ故障**と，レプリカの動作が予想不可能となる**ビザンチン故障**が主に扱われる．クラッシュ故障に耐性を持つ SMR は *CFT SMR*，ビザンチン故障に耐性を持つ SMR は *BFT SMR* と呼ばれる．CFT SMR の場合 $n \geq 2f+1$，BFT SMR の場合 $n \geq 3f+1$ の関係を満たす必要があることが理論的に証明されている [7]．以降，CFT SMR と BFT SMR を区別する必要がない場合は単に SMR と表記する．

2.2 関連研究

最適なレプリカ配置を決定する問題は，データレプリケーションの分野でこれまで多く行われている．Cook らは，シンプルな read-write ポリシー（データオブジェクトを読むときは 1 つのレプリカを参照する．データを書くときは，すべてのレプリカがあるノードと通信する）において，データの読み書きにかかる時間をコストとして定式化し，最適なレプリカ配置を決定することが NP 困難であることを証明した [2]．彼らはまた，同問題に対する近似アルゴリズムを提案した．彼らの対象はデータのレプリケーションであり，本論文で扱うレプリケーションとは異なる．しかしながら，彼らの論文で行われている定式化は本論文で扱う定式化と似ている．

Sen らは，データレプリケーションにおいて，数学モデルを用いてレプリカ配置の最適化問題を試みた過去の研究を包括的にまとめた [4]．そこでは，ファイル配置，プログラムとファイル配置，データベースセグメント配置，レプリカやミラー配置という 4 つの配置事例と，データ配置問題，クエリルーティング問題，サーバ位置問題，ユーザ割り当て問題の 4 つの問題に基づいて分類している．

一方で，広域 SMR におけるレプリカ配置の評価や最適化は，著者らの知る限りこれまでほとんど考慮されていない．与えられたレプリカ配置において，リーダと呼ばれる特別な振る舞いをするレプリカを変更することにより，応答性能を改善する広域 SMR プロトコル [6, 8] が一部存在するのみである．

3. 提案手法

本研究では，サイト間の通信時間に着目し，RTT と SMR の通信パターンに基づき広域 SMR のレイテンシを見積もる方法を提案する．RTT はサイト間の通信時間を把握するために使用し，SMR の通信パターンは各通信の発生タイミングを把握するために使用する．レイテンシの見積もりは，SMR の通信を RTT によって模擬することで行う．ここで，SMR で発生する各通信は 2 点間の片道の通信時間であり，RTT は 2 点間の往復の通信時間である．そのため，模擬の際には RTT の半分の値を使用する．

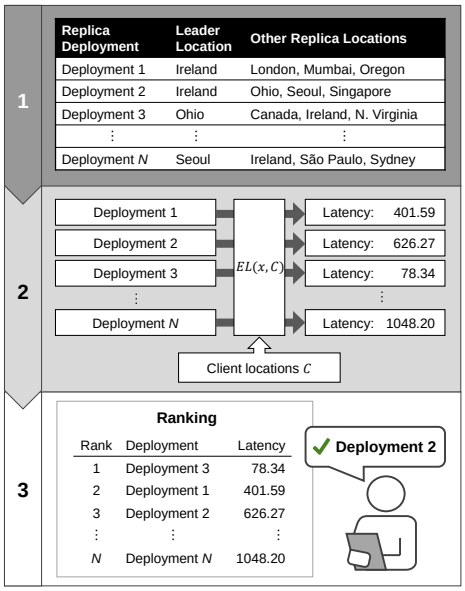

図 2 提案手法の概要
Fig. 2 Overview of the proposed method

また，実際の広域 SMR でクライアントがレプリケーションに対して行う通信は，書き込み操作と読み取り操作の 2 種類が存在し，それぞれ SMR の通信パターンが異なる．そこで，まず 2 つの操作ごとに見積もり値を計算し，その後，それぞれの操作の予想される発生割合を考慮し最終的なレイテンシを求める．予想される発生割合はサービス設計者ごとに決定する．

3.1 概要

図 2 に提案手法の全体像を示す．提案手法は，次の順序ですべてのレプリカ配置を評価し，最適なレプリカ配置を求める．

(1) まずすべての可能なレプリカ配置の集合 DC を，配置候補となるサイトの集合 SC[*1] とレプリカ数 n に基づき作成する．ここで各レプリカ配置は，リーダレプリカとその他のレプリカのペアで表される[*2]．
(2) 次に，各レプリカ配置 $x \in DC$ のレイテンシを，提案する見積もり方法によって求める．見積もり方法の詳細は 3.2 節で述べる．
(3) DC の要素を，見積もったレイテンシに基づき並び替え，ランキングとして出力する．

この結果，応答性能が最適なレプリカ配置がランキング 1 位として出力される．

[*1] 例えば Amazon EC2 にレプリケーションを構築する場合，Oregon, Paris, São Paulo, Tokyo 等のリージョンが，この集合の要素となる．
[*2] これは，リーダレプリカによって合意を行う SMR プロトコル [9–12] を想定している．もしリーダが存在しない SMR プロトコル [13–15] を利用する場合，各レプリカ配置は n 台のレプリカを要素に持つ集合となる．

3.2 応答性能の見積もり方法

本節では，レプリカ配置のレイテンシを RTT と SMR プロトコルの合意処理時の通信パターンから見積もる関数 $EL(x, C)$ について述べる．この関数は，レプリカ配置 x とクライアント位置の集合 C を引数として受け取る．また次式のように，各クライアントの見積もり値を関数 EL_c によって計算した後，それらの平均値を最終的な見積もり値として返す．

$$EL(x, C) = \sum_{c \in C} EL_c(x, c)/|C|, \tag{1}$$

以降では，各クライアントの見積もり値を計算する EL_c について詳細を述べる．EL_c は，クライアントが送信するリクエストの種類を読み取り操作と書き込み操作に分け，それぞれの操作のレイテンシを計算し，全体のリクエスト数に占める各操作の割合に基づいて最終的なレイテンシを決定する．これは，読み取り操作と書き込み操作によって，SMR プトロコルの通信パターンが異なるためである．書き込み操作の割合を p_w，読み取り操作の割合を p_r．書き込み操作時のレイテンシを EL_{c_w}, 読み取り操作時のレイテンシを EL_{c_r} とすると，EL_c は次の式で表せる．

$$EL_c(x, c) = EL_{c_w}(x, c) \times p_w + EL_{c_r}(x, c) \times p_r \tag{2}$$

ここで，$p_w + p_r = 1$ かつ $0 \leq p_w, p_r \leq 1$ である．

3.2.1 書き込み操作のレイテンシの見積もり

EL_{c_w} は，BFT-SMaRt のメッセージパターンを図 1 のように Request 部，Propose 部，Write 部，Accept 部，Response 部の 5 つのフェーズに分け，各レプリカがそれぞれのフェーズでメッセージ送受信するタイミングを RTT に基づいて Request 部から Response 部まで順に計算することで，書き込みリクエストのレイテンシを計算する．以降, 各フェーズにおける経過時刻を S_{req}, S_{wrt}, S_{acc}, S_{pro}, S_{res} と表記し，特定のレプリカ r_i における時刻は S^i_{pro} のように，上付き文字を追加することで表記する．また，レプリカまたはクライアント a と b の間の RTT の半分の時間を $\mathrm{RTT_h}(a, b)$ と表記する．

まず，リーダレプリカ l がクライアントからリクエストを受信するタイミング S_{req} は，次のように表される．

$$S_{req} = \mathrm{RTT_h}(c, l) \tag{3}$$

リーダレプリカ l はリクエストを受信すると，そのリクエストを各レプリカに Propose メッセージとして送信する．従って，レプリカ r_i が Propose メッセージを受信するタイミング S^i_{pro} は，次のように表される．

$$S^i_{pro} = S_{req} + \mathrm{RTT_h}(l, r_i) \tag{4}$$

レプリカは Propose メッセージを受信すると，Write メッセージを全レプリカにブロードキャストする．その後，レプリカは過半数（$\lceil (n+1)/2 \rceil$）のレプリカから同じ内容の Write メッセージを受信したとき，Write メッセージを受理する．レプリカ r_i が Write メッセージを受理するタイミング S^i_{wrt} は，レプリカ r_i がレプリカ r_j から送信された Write メッセージを受信するタイミングに基づいて計算される．

$$S^i_{wrt} = \mathrm{find}(T^i_{wrt}, \lceil (n+1)/2 \rceil) \tag{5}$$

ここで，$t_{wrt}(r_i, r_j) = S^j_{pro} + \mathrm{RTT_h}(r_j, r_i)$, $T^i_{wrt} = \{t \mid t_{wrt}(r_i, r_j), 0 \leq j < n\}$ であり，$\mathrm{find}(S, k)$ は，集合 S から k 番目に小さい要素を返す関数である．

Accept メッセージは，Write メッセージと同じ方法で送信される．したがって，$t_{acc}(r_i, r_j) = S^i_{wrt} + \mathrm{RTT_h}(r_j, r_i)$ と定義すると，S^i_{acc} は次のようになる．

$$S^i_{acc} = \mathrm{find}(T^i_{acc}, \lceil (n+1)/2 \rceil) \tag{6}$$

ここで，$T^i_{acc} = \{t \mid t_{acc}(r_i, r_j), 0 \leq j < n\}$ である．

最後に, レプリカは過半数の Accept メッセージを受信すると，リクエストを実行し，実行結果をレスポンスとしてクライアントへ送信する．クライアントは $n - f$ 個の同じ内容のメッセージを異なるレプリカから受信するとその結果を受理する．したがって，最終的に $EL_c(x, c)$ は次のように計算できる．

$$EL_{c_w}(x, c) = S_{res} = \mathrm{find}(T_{res}, n - f) \tag{7}$$

ここで，$T_{res} = \{t \mid S^i_{acc} + \mathrm{RTT_h}(r_i, c), 0 \leq i < n\}$ である．

3.2.2 読み取り操作のレイテンシの見積もり

EL_{c_r} が x と c を入力として読み取り操作のレイテンシを計算する方法を説明する．クライアントはリクエストを全レプリカに対して送信すると，各レプリカはただちにレスポンスをクライアントに返す．クライアントは $n - f$ 個の同じ内容のメッセージを異なるレプリカから受信するとその結果を受理する．したがって，$EL_{c_r}(x, c)$ は次のようになる．

$$EL_{c_r}(x, c) = \mathrm{find}(T_{read}, n - f) \tag{8}$$

ここで，$T_{read} = \{t \mid \mathrm{RTT_h}(c, r_i) + \mathrm{RTT_h}(r_i, c), 0 \leq i < n\}$ である．

4. 評価

この節では，提案手法の有効性を評価する．まず，4.1 節では，RTT($\mathrm{RTT_h}$) を用いてレプリカ配置の応答性能を評価することの妥当性を確認する．次に 4.2 節ではパブリッククラウド上に実際にレプリケーションを構築してレイテンシを計測し，提案手法によるレイテンシの見積もり精度を評価する．最後に 4.3 節では，レイテンシの見積もりにかかる時間について述べる．

すべての実験は，代表的なパブリッククラウドである Amazon EC2 で行う．レプリカを配置可能なサイトの集合

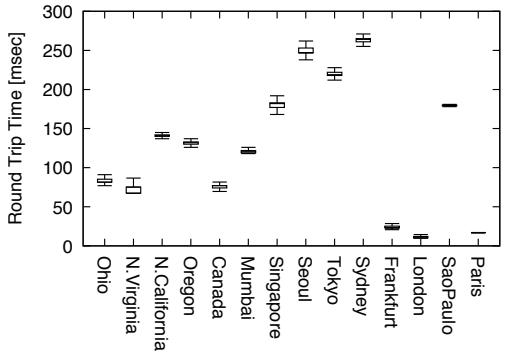

図 3 期間 A における Ireland からその他のリージョンへの RTT の変化

Fig. 3 Distribution of RTT from Ireland to each region during term A.

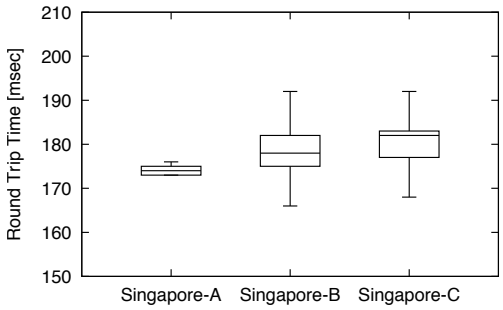

図 4 期間 A，B，C における Ireland と Singapore 間の RTT の変化

Fig. 4 Variation of RTT from Ireland to Singapore during terms A,B, and C.

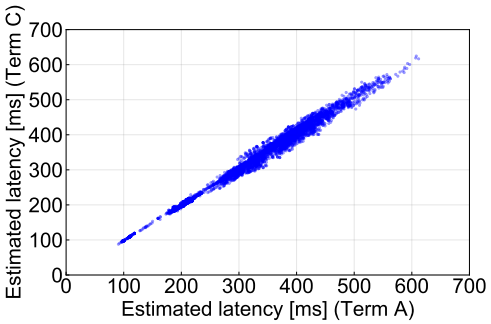

図 5 期間 A と期間 C のランキングの差 (C = Multiple)

Fig. 5 Ranking difference between Terms A and C (C = Multiple).

SC として，Amazon EC2 の 15 リージョンを利用する（すなわち $|SC| = 15$）．レプリカおよびクライアントを動作させるインスタンスには t2.micro を利用し，OS には Ubuntu Server 16.04 64bit を使用する．

4.1 RTT の有効性

ここでは，時間経過による RTT の変動を調べることで，レイテンシの見積もりに RTT を用いることが妥当であるかを評価する．また，見積もったレイテンシが，どの程度の期間有効であるかについても評価する．

4.1.1 実験方法

RTT の計測には `ping` コマンドを用いる．計測の準備として，各リージョンに 2 つのインスタンスを配置する．2 つのインスタンスの内 1 つは `ping` コマンドの実行用，もう片方は `ping` コマンドの宛先用である．各リージョンでは，実行用のインスタンスから，その他のリージョンの宛先用インスタンスに対して，2 秒おきに `ping` コマンドを実行し，RTT を計測する．計測は次に示す 3 つの期間で行った（時刻は UTC 表記）．

- 期間 A: 2018 年 3 月 7 日 19:27〜22:13
- 期間 B: 2019 年 1 月 11 日 11:14〜1 月 28 日 3:41
- 期間 C: 2019 年 4 月 15 日 15:48〜4 月 23 日 11:15

4.1.2 結果と考察

まず，図 3 に期間 C における RTT の変化を箱ひげ図として示す．ここでは Ireland の結果のみを紹介する．RTT はどのリージョンについても変動が見られるが，その変動は小さい．最も変動の大きい Ireland – Singapore 間では，RTT の平均値は 180.3 ms，標準偏差は，24.1 ms だった．

次に，計測期間が変化することによる RTT の変動について調べる．図 3 において最も変動が大きかった Ireland – Singapore 間について，期間 A，B，C の変動の推移を図 4 に示す．平均値はそれぞれ，期間 A が 175.3 ms，期間 B が 179.8 ms，期間 C が 180.3 ms だった．期間 A と期間 C では約 13 ヶ月の時間差があり，その間に RTT は 5 ms 上昇したことがわかる．一見僅かな差であるが，このような変化がすべてのリージョン間で起こると，提案手法を用いて求めたレイテンシは大きく変化する可能性が高い．

次に，計測期間の違いがレイテンシの見積もり結果に与える影響を調査するため，期間 A と期間 C のそれぞれの RTT の値を用いてランキングを作成した．結果を図 5 に示す．調査時のクライアント配置は Multiple（4.2 節で述べる），$p_w = 1$，$p_r = 0$ とした．結果から，特に期間 A の 250〜550ms にかけて，期間 A と期間 C の見積もり値に大きな差が生じていることがわかる．最も見積もり値の変化が大きかったレプリカ配置は Seoul（リーダ），São Paulo，Sydney，Tokyo であり，見積もり値は期間 A で 408.1ms，期間 C で 447.2ms であった．この結果から，RTT の計測期間の違い（及び RTT の違い）は提案手法によって見積もったレイテンシに大きく影響を及ぼすことが分かった．

以上の結果から，パブリッククラウドにおいて RTT の変動は短期間（数時間〜数日程度）では十分に小さく，通信時間に基づいてレプリカ配置を評価することは十分有効であることが確認できた．一方で，レプリカ配置の決定から 1 年間などの長期間が経過すると，各リージョン間の RTT は変化し，当初決定したレプリカ配置が最適ではなくなる．これは，最適なレプリカ配置を保つには定期的なレプリカの再配置が必要になることを示唆する．

4.2 見積もり精度

ここでは，可能なすべてのレプリカ配置について，実際に Amazon EC2 上にレプリケーションを構築して計測したレイテンシと，提案手法によって見積もったレイテンシを比較することで，提案手法の有効性を検証する．

4.2.1 実験方法

State machine replication は，オープンソースの SMR ライブラリ BFT-SMaRt [3] バージョン 1.1[*3]を用いて Amazon EC2 に構築する．BFT SMR を想定し，レプリカ数は $n = 4$，許容する故障台数 $f = 1$ とする．その他は，BFT-SMaRt の標準設定を用いる．同じリージョン上に複数のレプリカを配置することは考えない．レプリカを配置するサイトの組み合わせが同じでも，リーダレプリカのサイトが異なる場合は，異なるレプリカ配置とみなす．したがって，$|DC| = |SC| \times {}_{|SC|-1}\mathrm{C}_{n-1} = 5,460$ となる．

レプリカ同様，クライアントも Amazon EC2 上に配置されると仮定する．クライアントの位置と数の違いが見積もり結果に及ぼす影響を調査するため，クライアント配置として地理的に離れた Ireland, Sydney, N. Virginia のリージョンに 1 台のみ存在する配置（3 種類）と，各リージョンにクライアントが複数台存在する場合（Ireland に 10 台，Sydney に 3 台，N.Virginia に 5 台それぞれ存在するする場合）の合計 4 種類を用意した．以降，複数のリージョンにクライアントが存在する配置を Multiple と表記する．

レイテンシは，BFT-SMaRt に含まれるサンプルプログラム `LatencyClient` 及び `LatencyServer` を用いて計測する．`LatencyClient` は，定期的にリクエストをレプリケーションに送信し，レイテンシを計測する．`LatencyServer` は単に合意処理を行うだけの何の機能も提供しないサービスである．リクエストおよびレスポンスのデータサイズは，どちらも 1,024 Byte とする．書き込み操作の割合 p_w は 100%，読み取り操作の割合 p_r は 0%とした．`LatencyClient` はリクエストを，2 秒間隔で 50 回送信する．計測した値は上位 10%（5 個）と下位 10%（5 個）を外れ値として扱い，それ以外の値（合計 40 個）の平均値を，レプリケーションのレイテンシとする．レイテンシの見積もりには，4.1 節の期間 C の間に計測された RTT の平均値を使用する．

4.2.2 結果と考察

図 6 にクライアント配置 Ireland における，提案手法で見積もったレイテンシと実際にレイテンシを計測した値（実測値）の散布図を示す．図中の各点はレプリカ配置を表し，横軸は見積もったレイテンシ，縦軸はレイテンシの計測による実測値をそれぞれ表す．表 1 にはすべてのクライアント配置における平均二乗誤差平方根（RMSE）を示す．RMSE は，提案手法による見積結果と実測結果が完全に一致した場合を理想（すなわち $y = x$）として計算した．

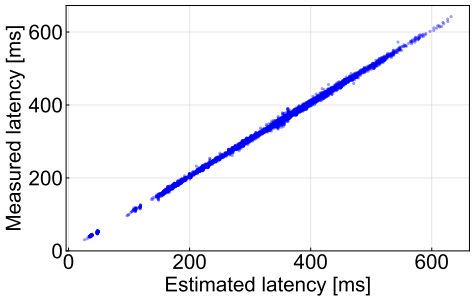

図 6 実測値と見積もり値の散布図 (C = Ireland)
Fig. 6 Scatter plot of measured latency and estimated latency (C = Ireland).

表 1 クライアント配置ごとの RMSE の一覧
Table 1 RMSE for each client deployment.

クライアント配置	RMSE
Ireland	6.8
N. Virginia	8.2
Sydney	8.0
Multiple	22.3

クライアント配置に着目すると，単一クライアントの結果よりも Multiple の方が RMSE の値が大きくなった．その原因の一つに，Multiple のスコアの算出方法は単一クライアントの結果を用いていることが考えられる．単一のクライアントの場合，実測値とスコアを単に比較し RMSE を計算するが，Multiple の場合は，比較する値を単一クライアントの結果の重み付けした値によって行う．その結果，重み付けによって Multiple の RMSE が単一クライアントの結果と比べて大きく変化した．

実験から，提案手法は様々なクライアント配置において，一貫して高い精度でレイテンシを見積もれることを確認できた．

4.3 見積もり時間

評価実験の最後では，提案手法の見積もりにかかる時間を評価する．なおレイテンシの見積もりには，一般的な PC[*4]を使用した．

まず，レプリカを配置するサイトの候補の数 $|SC|$ がレイテンシの見積もり時間へ及ぼす影響を調査する．$|SC|$ は，15，20，25，30 と変化させ，レプリカの数 $n = 4$ とした．表 2 に，$|SC|$ を変化させた時の全レプリカ配置候補のレイテンシ見積もりにかかった時間 t, レプリカ配置の総数 $|DC|$，1 レプリカ配置あたりの計算時間 $t/|DC|$ を示す．結果は，サイト数の増加に伴いレイテンシ見積もりにかか

[*3] `https://github.com/bft-smart/library/releases/tag/v1.1-beta`

[*4] CPU Intel Core i5 7400 3.0GHz，メモリ 8GB，Windows 10 Home 64-bit，Python 3.6.5.

表 2 $|SC|$ がランキング作成時間に与える影響（$n = 4$）
Table 2 Effect of $|SC|$ on ranking calculation time ($n = 4$).

$\|SC\|$	計算時間 t [sec]	$\|DC\|$	$t/\|DC\|$ [msec]
15	12.9	5,460	2.36
20	39.5	19,380	2.04
25	110.5	50,600	2.18
30	227.3	109,620	2.07

表 3 n がランキング作成時間に与える影響（$|SC| = 15$）
Table 3 Effect of n on ranking calculation time ($|SC| = 15$).

n	計算時間 t [sec]	$\|DC\|$	$t/\|DC\|$ [msec]
4	12.9	5,460	2.36
7	429.1	45,045	9.53
10	792.1	30,030	26.38
13	77.7	1,365	56.90

る時間が大きく増加した．これはレプリカ配置の総 $|DC|$ が，サイト数の増加に伴って増加するためである．

次に，レプリカ数 n が見積もり計算に与える影響を調査する．レプリカの数 n を 4，7，10，13 と変化させ，サイト候補数 $|SC|$ は 15 とした．表 3 にレプリカ数 n を変化させた時の t，$|DC|$，$t/|DC|$ を示す．t とレプリカ配置の総数 $|DC|$ を比べると，それぞれが最大となる n は同じではない．レプリカ配置候補すべてのレイテンシ見積もりにかかった時間 t は $n = 10$ で最大になるが，一方 $|DC|$ は $n = 8$ で最大になる．これはレプリカ数 n の増加に伴い，1 レプリカ配置あたりの計算時間 $t/|DC|$ が増加するためである．

最後に，調査した見積もり計算時間に基づき，レプリカ数 n とサイト数 $|SC|$ が大きい場合のレイテンシ見積もり計算時間を見積もった．その結果，あるサイト数におけるレイテンシの計算時間はレプリカ数がサイト数のおよそ半分の時，最大となることがわかった．例えば，レプリカ数が 16，サイト数が 30 の場合，t は約 3 年となる．そのような場合は，各レプリカ配置の見積もり計算を並列化することや，見積もり計算前に何らかの指標に基づいてレプリカ配置を除外する必要がある．

実験から，レプリカ数やサイト数が極端に大きくない場合において，数百秒ですべてのレプリカ配置のレイテンシを見積もれることがわかった．これは現実的な計算時間であり，本提案手法を用いたレプリカ配置の決定は十分有効であるといえる．

5. 提案手法の拡張

ここまで，提案手法によって応答性能を見積もる方法を述べた．一方で，提案手法は見積もり方法を変更することで，応答性能以外の評価指標に基づいて最適なレプリカ配置を決定することができる．ここでは耐故障性能，運用コスト性能，そして複数の性能を評価する場合の性能の見積もり方法について述べる．さらに，見積もり方法ごとに，4 節と同様の条件でランキングを作成し，その結果を述べる．

5.1 耐故障性能

広域 SMR において耐故障性の高いレプリカ配置は，耐故障性の低いレプリカ配置と比べて，任意の 2 つのレプリカ間の距離が離れている．そこで，本節ではレプリカが配置されるサイト間の距離の平均を見積もり値として計算することで，耐故障性能を評価する方法を述べる．

見積もりでは，レプリカ間の距離がより均等に離れている場合に耐故障性能が良いと判断できるよう，平均値の計算に調和平均を用いる．調和平均は，相加平均よりも，小さな値が平均値の計算結果により強く影響を及ぼす特徴がある．調和平均を用いることで，レプリカ間の距離が均等に離れている場合に見積もり値の値が大きくなり，耐故障性が高いと判断される．そのため，調和平均を用いてスコアを計算することは，相加平均を用いたスコアの計算よりも耐故障性を評価する計算方法として適している．サイト間の距離の計算には，地球を楕円で近似する Vincenty 法を用いる．距離の計算はこの他にも，球面三角法や，ヒュベニ (Hubeny) の公式等を用いる方法がある．

この見積もり方法でランキングを作成したところ，最も良いレプリカ配置は Mumbai, Oregon, São Paulo, Sydney で，見積もり値は 12041.42 km だった．このレプリカ配置は，どのサイト間も 1 万 km 以上離れており，大規模災害に対する耐性が高いことがわかる．一方，最下位のレプリカ配置は Frankfurt, Ireland, London, Paris で，見積もり値は 548.57km だった．

5.2 運用コスト性能

本節では，SMR の運用にかかる費用を見積もることで，レプリカ配置における運用コスト性能を評価する方法を説明する．ここでは，運用コスト性能を Amazon EC2 の 1 時間あたりの利用料とし，利用料は 1 時間あたりの Amazon EC2 のインスタンス料金（アメリカドル）と，1 時間あたりのデータ転送料金（アメリカドル）で計算できると仮定する．Amazon EC2 ではリージョンごとに利用料金が定められており，リージョンごとの 1 時間あたりのインスタンス料金と 1GB あたりのデータ転送料金は Web ページ上[*5]で確認できる．この情報から，レプリカを配置するサイトごとに運用コストの見積もり値を計算する．

まず，1 時間あたりのインスタンス料金を，インスタンスタイプから見積もる．次に，1 時間あたりのデータ転送料金を，1 時間あたりに発生する SMR プロトコルの合意回数，1 回の合意で送信されるリクエストやレスポンスの

[*5] https://aws.amazon.com/jp/ec2/pricing/on-demand/

データサイズ，SMR プロトコルの通信パターンから見積もる．最後に，レプリカを配置するサイトごとに求めた 1 時間あたりのインスタンス料金と 1 時間あたりのデータ転送料金を合計し，1 時間あたりの見積もり値とする．

この見積もり方法でランキングを作成したところ，ランキング最上位のレプリカ配置は，Ohio（リーダ）, Oregon, Sydney, N. Virginia と，N. Virginia（リーダ）, Ohio, Oregon, Sydney の 2 つで，見積もり値は 0.1441 アメリカドルだった．一方で最下位のレプリカ配置は São Paulo（リーダ）, Singapore, Sydney, Tokyo であり，見積もり値は 0.3369 アメリカドルと，最上位と比べて 2.33 倍も運用コストが高い．

5.3 総合性能

最後に，応答性能，耐故障性能，運用コスト性能を評価する各見積もり方法を組み合わせて，総合的な性能を評価する方法を説明する．この方法ではまず，性能ごとにすべてのレプリカ配置の見積もり値を求める．次に，性能ごとに求めた見積もり値の最小値と最大値に基づき，見積もり値を 0〜1 で正規化する．このとき，見積もり値の小さいものが良いレプリカ配置になるように，必要であれば値の正負を反転させる．最後に，レプリカ配置ごとに各性能の正規化された見積もり値を合計し，総合性能を評価する見積もり値とする．

この見積もり方法でランキングを作成したところ，最上位のレプリカ配置は Oregon（リーダ）, Paris, Sydney, Tokyo で，見積もり値は 0.7623 だった．なお，このレプリカ配置は，応答性能を表すレイテンシの見積もり値が 320.7ms，耐故障性能を表すレプリカ間の調和平均の見積もり値が 9706.69km，運用コスト性能を表す 1 時間あたりの運用コストの見積もり値が 0.1670 アメリカドルであり，各性能のバランスがとれていることがわかる．一方で，最下位のレプリカ配置は São Paulo（リーダ）, London, Paris, Sydney で，見積もり値は 2.4938 だった．なお，このレプリカ配置はレイテンシの見積もり値が 616.3ms，レプリカ間の調和平均の見積もり値が 1801.91km，運用コストの見積もり地が 0.2703 アメリカドルだった．

6. おわりに

本研究では，広域 State machine replication によるサービス構築に利用可能な，応答時間最適なレプリカ配置決定手法を提案した．広域 SMR はパブリッククラウドの発達によって容易に構築できるようになったものの，サービス設計者が望む最適なレプリカ配置の実現は，性能の把握に時間と手間がかかり，レプリカ配置の候補数が膨大であるため難しい．本研究では，応答性能として，レイテンシを SMR の通信パターンとあらかじめ測定した RTT を用いて見積もる手法を提案した．レイテンシの見積もりは，SMR プロトコルの通信パターンに基づき，RTT の半分の値で通信を模擬することで行う．実験により，RTT を用いたレイテンシ見積もりが妥当であり，見積もり精度は高く，レプリカ数とサイト数が少ない状況では高速で見積もりが完了することを示した．さらに，見積もり方法を変更することで応答性能以外に，耐故障性能，運用コスト性能，総合的な性能の評価に拡張できることを示した．

謝辞 本研究は JSPS 科研費 18K18029 の助成を受けて実施されました．

参考文献

[1] Schneider, F. B.: Implementing fault-tolerant services using the state machine approach: a tutorial, *ACM Computing Surveys*, Vol. 22, No. 4, pp. 299–319 (1990).
[2] Cook, S. A., Pachl, J. and Pressman, I. S.: The optimal location of replicas in a network using a READ-ONE-WRITE-ALL policy, *Distributed Computing*, Vol. 15, No. 1, pp. 57–66 (2002).
[3] Bessani, A., Sousa, J. a. and Alchieri, E. E. P.: State Machine Replication for the Masses with BFT-SMaRt, *DSN 2014*, pp. 355–362 (2014).
[4] Sen, G., Krishnamoorthy, M., Rangaraj, N. and Narayanan, V.: Facility location models to locate data in information networks: a literature review, *Ann. of Operations Research*, Vol. 246, No. 1-2, pp. 313–348 (2015).
[5] Sousa, J. and Bessani, A.: Separating the WHEAT from the Chaff: An Empirical Design for Geo-Replicated State Machines, *SRDS 2015*, pp. 146–155 (2015).
[6] Eischer, M. and Distler, T.: Latency-Aware Leader Selection for Geo-Replicated Byzantine Fault-Tolerant Systems, *DSN-W 2018*, pp. 140–145 (2018).
[7] Lamport, L.: Lower bounds for asynchronous consensus, *Distributed Computing*, Vol. 19, No. 2, pp. 104–125 (2006).
[8] Liu, S. and Vukolic, M.: Leader Set Selection for Low-Latency Geo-Replicated State Machine, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 28, No. 7, pp. 1933–1946 (2017).
[9] Lamport, L.: The part-time parliament, *ACM Trans. on Computer Systems*, Vol. 16, No. 2, pp. 133–169 (1998).
[10] Castro, M. and Liskov, B.: Practical byzantine fault tolerance and proactive recovery, *ACM Transactions on Computer Systems*, Vol. 20, No. 4, pp. 398–461 (2002).
[11] Sousa, J. and Bessani, A.: From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation, *EDCC 2012*, pp. 37–48 (2012).
[12] Kotla, R., Alvisi, L., Dahlin, M., Clement, A. and Wong, E.: Zyzzyva: speculative byzantine fault tolerance, *SOSP 2007*, pp. 45–58 (2007).
[13] Moniz, H., Neves, N. F., Correia, M. and Verissimo, P.: RITAS: Services for Randomized Intrusion Tolerance, *IEEE Trans. on Dependable and Secure Computing*, Vol. 8, No. 1, pp. 122–136 (2011).
[14] Cachin, C., Kursawe, K., Petzold, F. and Shoup, V.: Secure and Efficient Asynchronous Broadcast Protocols, *CRYPTO 2001*, pp. 524–541 (2001).
[15] Nakamura, J., Araragi, T., Masuyama, S. and Masuzawa, T.: Efficient Randomized Byzantine Fault-Tolerant Replication based on Special Valued Coin Tossing, *IEICE Trans. on Information and Systems*, Vol. E97-D, No. 2, pp. 231–244 (2014).