電気学会研究会資料（通信研究会）
The Papers of Technical Meeting on "Communications", IEE Japan
2021.8.25-26

# エッジコンピューティングにおけるフィードバック遅延改善のためのキャッシュパラメータ決定手法の検討

伊藤　友輔*，郡司　理太（東京理科大学），木村　共孝（同志社大学）
平田　孝志（関西大学），村口　正弘（東京理科大学）

## A study on cache parameter determination method to improve feedback delay in edge computing

Yusuke Ito*, Rita Gunji (Tokyo University of Science), Tomotaka Kimura (Doshisha University)
Kouji Hirata (Kansai University), Masahiro Muraguchi (Tokyo University of Science)

Abstract

In this paper, we propose a cache parameter determination method to improve feedback delay in edge computing. Our method dynamically determines a cache parameter to adjust the weight of each metric based on popularity and generation time of feedback information stored in edge server's cache space. The simulation results show that the proposed scheme can improve feedback time.

**キーワード**：IoT，エッジコンピューティング，キャッシング，キャッシュパラメータ，生成時間
(IoT, edge computing, caching, cache parameter, generation time)

## 1. はじめに

近年，様々なモノがインターネットに接続する IoT (Internet of Things) 化が急速に進んでいる．これに伴い，エッジ側で生成されるリアルタイムデータをより短時間で処理する需要が高まっている．このため，ボトルネックとなるクラウドコンピューティングの代わりに，エッジコンピューティング[1]を適用するシステムモデルが提案されている．この環境では，ユーザからの要求により，エッジサーバはセンサー端末などから IoT 情報を収集・処理し，その後フィードバック情報を配信する．この際に，フィードバック情報はエッジサーバにキャッシュされ，今後，同一のフィードバック情報を近隣のユーザがリクエストした際に，そのキャッシュを利用して短時間でフィードバック情報を配信することが可能となる．このように，キャッシュの活用により，リアルタイム性のさらなる向上が期待できるが，エッジサーバのキャッシュ容量は制限されるため，適切にキャッシュすべき情報を選択することが重要である．

先行研究では，様々なキャッシング手法[2,3]が提案されているが，それらはキャッシュ判断の指標として人気度のみに着目したものであり，フィードバック生成時間まで考慮されていない．文献[4]では，エッジコンピューティング環境下におけるキャッシュ利用効率の向上を目的として，フィードバック情報の人気度だけでなく生成時間も考慮してキャッシュする情報を決定する手法が提案されている．このキャッシング手法では，人気度と生成時間の重視する割合をキャッシュパラメータによって調整し，以下の式(1)に従ってキャッシュ優先度 $P$ を算出する．

$$P = \begin{cases} \dfrac{T_g}{T_{gmax}}(1-w)+w & (T_r = 0) \\ \dfrac{T_g}{T_{gmax}}(1-w)+\dfrac{T_{rmin}}{T_r}w & (T_r > 0) \end{cases} \quad (1)$$

ここで，$T_g$ と $T_r$ はフィードバック情報の生成時間と参照時間，$T_{gmax}$ と $T_{rmin}$ は最大生成時間と最小参照時間，$w \in [0,1]$ はキャッシュパラメータとして定義される．

実環境では，サービス特性は多種多様である上に，ネットワークの輻輳状態やエッジサーバの処理状況，ユーザの要求傾向の変化は容易に発生する状況にあり，フィードバック情報の人気度や生成時間は動的に変化するが，文献[4]では静的にキャッシュパラメータを決定するため，状況変化に対応することは困難である．

本研究では，文献[4]の手法をベースとして，各サービスの人気度や生成時間の変化に対応し，よりフィードバック遅延を改善できるような動的なパラメータ $w$ の決定手法を提案する．加えて，シミュレーション評価を行うことで，提案手法によってパラメータを決定することにより，環境の変化に追従して，最悪ケースよりも平均フィードバック時間を短縮できることを示す．

## 2. 提案手法

本手法はフィードバック時間を評価するための指標に関してキャッシュ内のフィードバック情報同士を逐次的に比較していき，人気度と生成時間のどちらを重視すべきペアがより多いかを判断し，その結果に基づいてキャッシュパラメータ $w$ を決定する．以下では，まず 2・1 節でフィードバック情報同士の比較時に使用する指標について述べる．次に，2・2 節で具体的なキャッシュパラメータの決定方法について述べる．

〈2・1〉フィードバック時間に関する考察

ある一つのサービスのフィードバック時間について考察する．ここでは，サービスの要求間隔 $i$ とフィードバック生成時間 $g$ が常に一定である状況を前提として考える．

(1) フィードバック時間のパターン

キャッシュ機能がないものとすると，あるサービスの要求間隔 $i$ とフィードバック生成時間 $g$ の関係によってユーザリクエストのタイムスケジュールのパターンは以下の 4 通り存在する．ここでは，6 回分のリクエストの平均フィードバック時間について考える．

**パターン 1: $g < i$ の場合**

全リクエストが生成処理のトリガーとなり，それぞれ自身のリクエストによって開始された生成処理を待つことになるため，各リクエストのフィードバック時間は $g$ となる．したがって，平均フィードバック時間も $g$ になる．

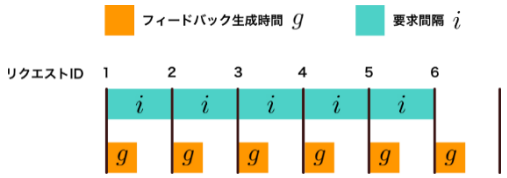


図 1 $g < i$ の場合

Fig. 1. $g < i$.

**パターン 2: $g = i$ の場合**

生成処理のトリガーとなるリクエストのフィードバック時間は $g$ となるが，それ以外のリクエストについては，既に実行中の処理が完了した後にまとめてフィードバックされるため，そのフィードバック時間は 0 となる．したがって，平均フィードバック時間は $\frac{g}{2}$ になる．生成処理終了と新規リクエスト到着のタイミングが同時である場合の動作については諸説あるが，今回は上記のように考える．

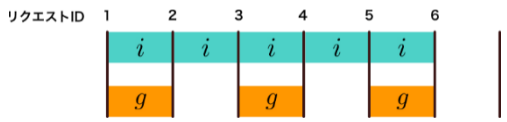


図 2 $g = i$ の場合

Fig. 2. $g = i$.

**パターン 3: $g > i$ ($g$ は $i$ の倍数でない) の場合**

生成処理のトリガーとなるリクエストのフィードバック時間は $g$ となるが，それ以外のリクエストについては，既に生成処理が開始されているため，フィードバック時間が $g$ よりも小さくなる．この場合の平均フィードバック時間は $\frac{2g-i}{2}$ になる．

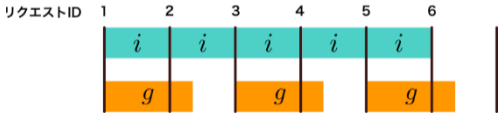


図 3 $g > i$ ($g$ は $i$ の倍数でない) の場合

Fig. 3. $g = i\ (i \nmid g)$.

**パターン 4: $g > i$ ($g$ は $i$ の倍数) の場合**

生成処理中に到着するリクエストの中に，生成処理終了のタイミングと同時になるものが存在する．つまり，パターン 1 と 2 を組み合わせた動作となる．したがって，これら 6 回分のリクエストに要する平均フィードバック時間は $\frac{2g-i}{3}$ になる．

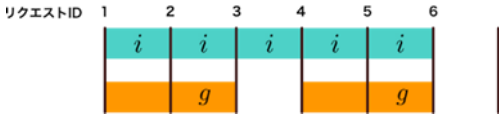


図 4 $g > i$ ($g$ は $i$ の倍数) の場合

Fig. 4. $g = i\ (i \nmid g)$.

(2) フィードバック時間の一般化

以下では，それぞれのパターンにおける各リクエストのフィードバック時間を順に並べた数列について考える．図 5 より，これらのフィードバック時間列は周期性を持つことがわかる．$g - ki\ (k \geq 0)$ が正となる最大の整数 $k$ を用いて 1 周期は $k+1$ と求めることができ，$\left\lfloor \frac{g}{i} \right\rfloor + 1$ 回を一つのまとまりとして，各リクエストのフィードバック時間が繰り返される．

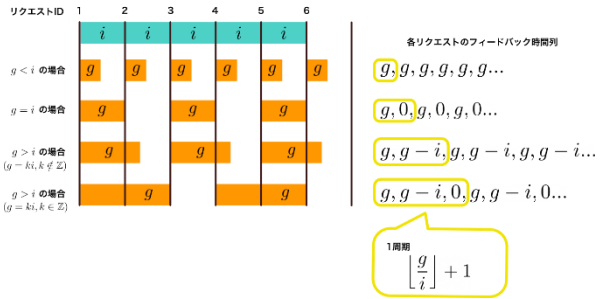


図 5 各パターンにおけるフィードバック時間列

Fig. 5. Patterns of feedback time series.

次に，この周期性を利用して，総リクエスト回数 $R$ のときの累計フィードバック時間 $F$ を求める．$R = T\left(\left\lfloor \frac{g}{i} \right\rfloor + 1\right) + r$ を満たすように周期数 $T$ と残りのリクエスト数 $r$

を定めると，$F$ は次式のように表される．

$$F = \left\{ \sum_{k=0}^{\left\lfloor \frac{g}{i} \right\rfloor} (g - ki) \right\} T + \sum_{k=0}^{r-1} (g - ki) \tag{2}$$

実際の環境では，総リクエスト数 $R$ が無限大とみなすことができるほど非常に大きい数になると考えられるため，$T$ の値が非常に大きくなり，残りの項の影響をほとんど無視することができる．したがって，最終的に累計フィードバック時間 $F$ は次式のようになると考えられる．

$$F \approx \left\{ \sum_{k=0}^{\left\lfloor \frac{g}{i} \right\rfloor} (g - ki) \right\} T \tag{3}$$

さらに，1 周期分のリクエスト数を $f = \left\lfloor \frac{g}{i} \right\rfloor + 1$ として，式(3)の 1 周期分のフィードバック時間を変形すると，以下のようになる．

$$\sum_{k=0}^{f-1} (g - ki) = gf - i\frac{(f-1)f}{2} = f\left\{ g - \frac{i(f-1)}{2} \right\} \tag{4}$$

基本的にサーバは長期間稼働するものであるため，式(4)の 1 周期分のフィードバック時間が最も全体のフィードバック時間に影響を与えるということがいえる．したがって，サービス間のフィードバック時間を比較する際の指標として，式(4)の値を利用することができると考えられる．

式の簡単化のため，1 周期分のリクエスト数 $f$ が実数となることを許容し，$f = \frac{g}{i} + 1$ として，式(4)を以下のように変形する．

$$f\left\{ g - \frac{i(f-1)}{2} \right\} = \frac{g(g+i)}{2i} \tag{5}$$

以降では，式(5)から得られる値をサービス間のフィードバック時間を比較する際の指標として使用する．

〈2･2〉キャッシュパラメータ決定手法

(1) 2 つのサービスを比較する場合

ある 2 つのサービスを比較した時に，どちらをキャッシュする方がよりフィードバック遅延を改善できるか考える場合，$\frac{g(g+i)}{2i}$ の大小関係を比較し，値が大きい方をキャッシュすれば良いと判断することができる．キャッシュパラメータ $w$ については，2 つのサービスのうち，キャッシュすべきサービスの要求間隔が小さい場合は人気度重視ペアとして 1，そうでない場合は生成時間重視ペアとして 0 に設定すれば良い．

(2) 複数サービスを比較する場合

次に，総数 $N$ の複数サービスに拡張した場合を考える．この場合，${}_N\mathrm{C}_2$通りの全ペアに対して，$w$ を 0 と 1 のどちらに設定するべきかを判定し，それらの結果の平均値を適切なキャッシュパラメータの近似解 $w^*$ として採用する．$w^*$ は式(6)として表される．

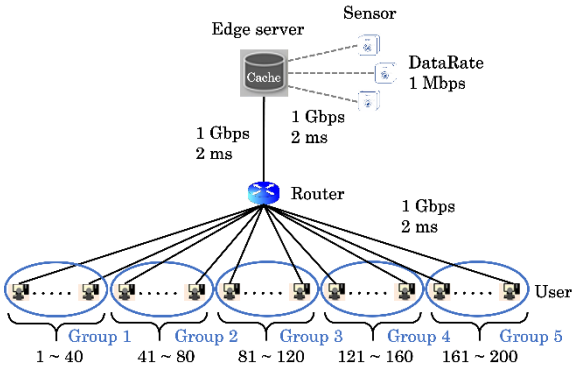


図 6 シミュレーションモデル

Fig. 6. Simulation model

$$w^* = \begin{cases} 1.0 & (N < 2) \\ \dfrac{S_g \times 0 + S_r \times 1}{{}_N\mathrm{C}_2} = \dfrac{S_r}{{}_N\mathrm{C}_2} & (N \geq 2) \end{cases} \tag{6}$$

ここで，$S_g$ は生成時間を重視すべきペア総数，$S_r$ は参照時間を重視すべきペア総数として定義される．式(6)は LRU に従って並べたキャッシュ優先順位から，フィードバック時間を考慮した指標に従って並べた順位へ並び替える時の隣接要素の交換回数を適切なパラメータの近似解として利用することを意味する．また，$S_r$ は数列の転倒数として考えることができ，BIT (Binary Indexed Tree) [5]を用いることで，現実的な計算時間で算出可能である．

表 1 フィードバック情報の詳細情報

Table 1. Details of feedback information.

| 人気度 | 低 <----------> 高 | | | | |
|---|---|---|---|---|---|
| グループ ID | 1 | 2 | 3 | 4 | 5 |
| フィードバック ID | 1~20 | 21~40 | 41~60 | 61~80 | 81~100 |
| 要求間隔 [s] | 10.3 | 2.8 | 1.41 | 0.93 | 0.7 |
| IoT 情報量 [MB] | 0.01 | 3.5 | 1~5 | 3.5 | 0.01~0.5 |

## 4. シミュレーション評価

提案法の有効性を調査するために，シミュレーションによって，静的にパラメータを決定する場合と提案手法によってパラメータを動的に調整する場合の比較評価を行う．

〈4･1〉シミュレーション環境

本シミュレーションでは，図 6 のトポロジを用いて評価を行う．この環境では，200 人のユーザを 40 人ずつ 5 グループに分け，各グループのユーザは 20 サービスの中からランダムに一つ選択し，エッジサーバに要求する．エッジサーバは 5 台のセンサーデバイスから IoT 情報を取得し，そのデータを基に処理を行うことで，フィードバック情報を生成する．各ユーザグループの要求間隔と要求先サービスの IoT 情報量は表 1 に示す．エッジサーバのキャッシュ

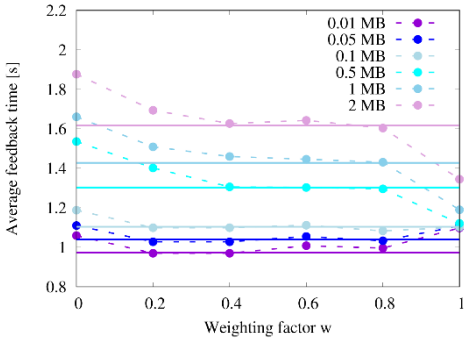


図 7　最も人気度が高いサービスの IoT 情報量による影響

Fig. 7. Effect of most popular data size.

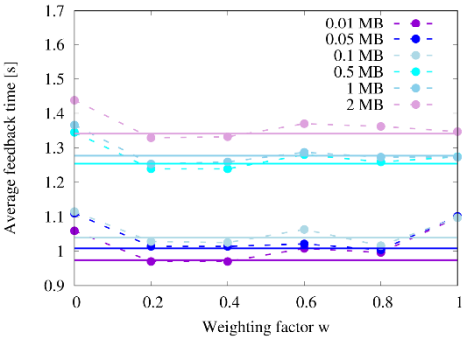


図 8　最も人気度が低いサービスの IoT 情報量による影響

Fig. 8. Effect of most unpopular data size.

容量は 30 MB，シミュレーション時間は 180 s，試行回数は 30 回としている．

**〈4･2〉評価指標**

本評価では，全ユーザの要求に対するフィードバック時間の和を全要求回数で除算した平均フィードバック時間を評価指標として使用する．

**〈4･3〉比較対象**

キャッシュパラメータ $w$ を 0.0~1.0 の範囲内の値で静的に設定する場合と，提案手法を用いて動的にパラメータを調整する場合を比較評価する．

## 5.　シミュレーション結果

最も人気度が高いもしく低いサービスの IoT 情報量を変化させた場合の評価結果を図 7 と図 8 に示す．点線部は静的に各 $w$ を決定した場合，実線部は提案手法を用いた場合の平均フィードバック時間を表す．

点線部に注目すると，最も人気度が高いサービスの IoT 情報量の変化とともに，平均フィードバック時間が最小値付近の値となるパラメータ $w$ の値も変化するが，最も人気度が低いサービスの IoT 情報量を変化させた場合は最小値付近となる $w$ の値が特に変化しないことがわかる．

実線部に注目すると，提案手法を利用することで，適切なパラメータ $w$ の変化にある程度追従することができる．人気度が高いサービスの情報量を変化させた場合は最悪ケースよりも 10 %程度，人気度が低いサービスの IoT 情報量を変化させた場合は 6~10%程度フィードバック遅延を改善できることを確認できる．

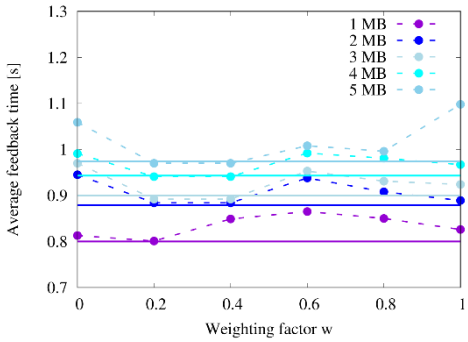


図 9　IoT 情報量の最大値による影響

Fig. 9. Effect of maximum data size.

一方で，最も人気度が高いサービスの IoT 情報量が大きくなった場合など，最小値付近の値に $w$ を調整できない状況になりうることも確認できる．今回はキャッシュ処理が実行される毎にパラメータ調整を行っているため，無駄な調整も含まれていたこと等がこの理由として考えられる．

IoT 情報量の最大値を変化させた場合の評価結果を図 9 に示す．図 7 と図 8 の場合と同様に，提案手法は適切なパラメータ $w$ の変化にある程度追従することができる．

## 6.　まとめ

本稿では，フィードバック生成時間を考慮したキャッシング手法において，適切なキャッシュパラメータ $w$ が変化する状況下であっても，ある程度その変化に追従してパラメータを調整できる決定手法を提案した．また，シミュレーション結果により，提案手法を用いることで，最悪ケースと比較してフィードバック遅延を改善できることを示した．今後の課題として，パラメータ調整頻度による本提案手法の性能への影響の評価などが挙げられる．

## 文　　献


（1） W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

（2） X. Wei, et al., "Similarity-aware popularity-based caching in wireless edge computing," Proc. ACM CF2020, pp. 257-260, May 2020.

（3） P. Shu and Q. Du, “Group behavior-based collaborative caching for mo-bile edge computing,” *Proc. IEEE ITNEC2020*, 7 pages, June 2020.

（4） Y. Ito, Y. Wada, T. Kimura, K. Hirata, and M. Muraguchi, "A caching scheme based on popularity and generation time of feedback information in edge computing," Proc. IEEE ICCE-TW2020, Sept. 2020.

（5） P. Fenwick, "A new data structure for cumulative frequency tables," *Software: Practice and experience*, vol. 24, no. 3, pp. 327-336, Mar. 1994.