Vol.2015-SLDM-170 No.6
Vol.2015-EMB-36 No.6
2015/3/6

仮想リリース遡及法の非周期タスクへの適用

田中 清史[1,a)]

概要：Earliest Deadline First（EDF）に基づくスケジューリングでは，より早期のデッドラインを持つジョブが高い優先順位となる．そこで，特定のタスクの優先順位を上げるためにはデッドラインの前倒しが有効である．スケジュール可能性を維持しつつこれを実現する方法として，著者は過去に**仮想リリース遡及法**を提案した．既提案の仮想リリース遡及法が周期タスクのジッタ削減を目的としていたのに対し，本稿では，非周期タスクに適用することにより，タスクの応答時間を短縮することを試みる．シミュレーション評価において，非周期タスクの平均応答時間を最大 14.8%短縮することを確認した．

Virtual Release Advancing Applied to Aperiodic Tasks

Kiyofumi Tanaka[1,a)]

Abstract: In Earliest Deadline First (EDF) scheduling, tasks with earlier deadlines have higher priorities. Therefore, making the deadlines earlier is effective to raise particular tasks' priorities. The author proposed *virtual release advancing* to obtain earlier deadlines for particular tasks while keeping schedulability. While the method originally aimed to reduce jitters of periodic tasks, this paper proposes the application of the method to aperiodic tasks and tries to shorten their response times. In the simulation-based evaluation, the virtual release advancing reduced the average response times of aperiodic tasks by up to 14.8%.

1. はじめに

リアルタイムスケジューリングアルゴリズムの一つである Earliest Deadline First（EDF）[1] では，より早期のデッドラインを持つジョブが優先して実行される．このため，システム内で重要度の高いタスクに対してデッドラインを早めることで，その応答時間およびジッタを短縮することが可能である．しかし，デッドラインを早めることにより EDF におけるスケジュール可能（Schedulable）条件 [2] を満たさなくなると，デッドラインミスが発生することになる．

著者は過去に，周期タスクと非周期タスクの混在セットに対するアルゴリズムである Total Bandwidth Server (TBS) [3] の基本原理に対し，タスク実行の早期終了の予測を導入し，タスクセット全体のスケジュール可能状態を維持しつつ，応答時間およびジッタを削減するいくつかの手法を提案してきた [4], [5], [6], [7], [8]．これらは全て，特定の対象（周期，あるいは非周期）タスクのデッドラインを前倒しすることにより，EDF スケジューリングにおける優先順位を上昇させ，当該タスクのリアルタイム性を向上させる手法である．このうち，文献 [6] で提案された仮想リリース遡及法（Virtual Release Advancing）は，タスクのリリース（起動要求）時刻を仮想的に前倒しすることにより，対応するデッドラインを早期化する方法である．文献 [6] では，周期タスクのジッタ削減を主目的とする手法として提案されたが，本稿において，非周期タスクに適用する場合の仮想リリース遡及法を提案し，本手法により非周期タスクの平均応答時間が短縮可能であることをシミュレーション評価により示す．

本稿は以下の構成をとる．2 節において提案方式が基本技術として利用する TBS の概要を紹介する．続いて，3 節において非周期タスクに対する仮想リリース遡及法を定義し，それを実現するアルゴリズムを示す．4 節でシミュレーションにより提案手法の評価を行い，最後に 5 節で本稿をまとめる．

[1] 北陸先端科学技術大学院大学
JAIST, Asahidai 1-1, Nomi, Ishikawa 923–1292, Japan
[a)] kiyofumi@jaist.ac.jp

2. Total Bandwidth Server

Total Bandwidth Server（TBS）は周期的なハードタスクと非周期的な非リアルタイムタスクの混在セットに対して，非周期タスクの実行を担当する（スケジューリング上の仮想的な）動的優先度サーバの一つであり，非周期タスクに対して短い応答時間を提供し，かつ軽いスケジューラ実装を可能としている [3], [9]．前提として，ハードタスクは周期的に起動され，周期と一致するデッドラインを持つ．一方，非リアルタイムタスクは非周期的に起動され，デッドラインは持たないが，起動（要求）されたときに，以下の式から求まる仮のデッドライン d_k が与えられる．

$$d_k = max(r_k, d_{k-1}) + \frac{C_k^{WCET}}{U_s} \quad (1)$$

ここで，k は k 番目の非周期タスクインスタンスであることを意味する．r_k は k 番目のインスタンスの起動要求（リリース）時刻，d_{k-1} は $k-1$ 番目（一つ前）のインスタンスのデッドライン，C_k^{WCET} は k 番目のインスタンスの最悪実行時間（Worst-Case Execution Time, WCET），および U_s は非周期タスクの実行を担当するサーバのための，プロセッサ使用率として表現されるバンド幅である．この式により，非周期タスクの起動要求発生毎に，そのインスタンスの実行に U_s のバンド幅を与えることになる．$max(r_k, d_{k-1})$ の項により，連続するインスタンス間でバンド幅が重ならないように調整している．非周期タスクのインスタンスが仮のデッドラインを与えられた後，全ての周期タスクと非周期タスクが EDF アルゴリズムによってスケジュールされる．U_p をハード（周期）タスクのプロセッサ使用率とすると，$U_p + U_s \leq 1$ でタスクセットがスケジュール可能であることが証明されている [3]．

TBS において，非周期タスクの WCET が過大見積もりされている状況下では，式 1 によってタスクインスタンスのデッドラインが過大に計算されることになる．したがって，当該インスタンスの実行が遅延され，応答時間が長くなる可能性がある．文献 [10] では，インスタンスの実行が終了したときに，実際に費やした実行時間によってそのタスクインスタンスのデッドラインを再び計算し，後続する非周期タスクインスタンスのデッドライン計算に反映させるリソース回収（resource reclaiming）を試みている．これにより，後続するインスタンスはデッドライン早期化の恩恵を受けて応答時間が改善することが期待できる．

リソース回収を伴う TBS では，非周期タスクインスタンスは以下の式によって仮のデッドラインが与えられる．

$$d'_k = \overline{r}_k + \frac{C_k^{WCET}}{U_s} \quad (2)$$

$\overline{r}_k$ は以下で計算される値である．

$$\overline{r}_k = max(r_k, \overline{d}_{k-1}, f_{k-1}) \quad (3)$$

すなわち，要求時刻と，前のインスタンスの再計算されたデッドライン（$\overline{d}_{k-1}$，後述），および前のインスタンスの終了時刻（f_{k-1}）のうち，最大のものがリリース時刻として選ばれる．（$\overline{d}_{k-1}$ が f_{k-1} よりも早くなることがありえる．これは，あくまでも $k-1$ 番目のインスタンスはリソース回収前の古いデッドラインで実行されたためである．）$k-1$ 番目の非周期タスクが終了したとき，実際に費やした実行時間（$\overline{C}_{k-1}$）を使用する以下の式によってデッドラインの再計算を行い，後続タスクのリリース時刻選択の式 3，続いて後続タスクのデッドライン計算の式 2 へ反映させる．

$$\overline{d}_{k-1} = \overline{r}_{k-1} + \frac{\overline{C}_{k-1}}{U_s} \quad (4)$$

本稿で提案する仮想リリース遡及法は，このリソース回収法と組み合わせることにより，より大きな効果を持つことになる．このことについては 3.2 節で詳しく述べる．

3. 仮想リリース遡及法

本節において，リアルタイム性向上の対象となるタスクに対して，デッドラインの早期化を実現する仮想リリース遡及法を提案する．文献 [6] において周期タスクのみに限定した場合の仮想リリース遡及法が示されたが，本節では周期タスクのみならず非周期タスクも適用対象とする一般化された仮想リリース遡及法を示す．

TBS では基本的に，リリース（起動要求）時刻を起点として，最悪実行時間と使用可能バンド幅から求まる時間幅を加えることで絶対デッドライン時刻を得ている．このことから，リリース時刻が早まると，絶対デッドライン時刻がより早くなり，当該インスタンスが早期にスケジュールされる可能性が高くなる．

仮想リリース遡及法は，タスクの起動要求の発生時に，要求時刻がより過去であったと仮定し，より短いデッドラインを得る手法である．ただし，スケジュール可能性を維持するために，過去のスケジュールに影響を及ぼさない範囲で仮想リリース遡及を行う．これが可能となるのは，より早期のデッドラインを持つ他のタスクインスタンスが，対象インスタンスの起動要求発生時刻よりも以前にスケジュールされていた場合である．このような場合，たとえリリース時刻をより過去の時刻とみなしたとしても，実際のリリース時刻よりも過去のスケジュール結果は同じである．このように，過去のスケジュールが変更されない範囲でリリース時刻を仮想的に過去の時刻と見なすことにより，対応するデッドライン時刻が前倒しされ，対象タスクの応答性の向上が期待できる．

3.1 仮想リリース遡及法の例

図 1 に仮想リリース遡及法による応答時間短縮の例を示す．図では 2 つの周期タスク τ_1, τ_2 がそれぞれ周期 $T_1 = 12$, $T_2 = 10$，実行時間 $C_1 = 4$，$C_2 = 5$ を持つ．2 つの周期タ

Vol.2015-SLDM-170 No.6
Vol.2015-EMB-36 No.6
2015/3/6

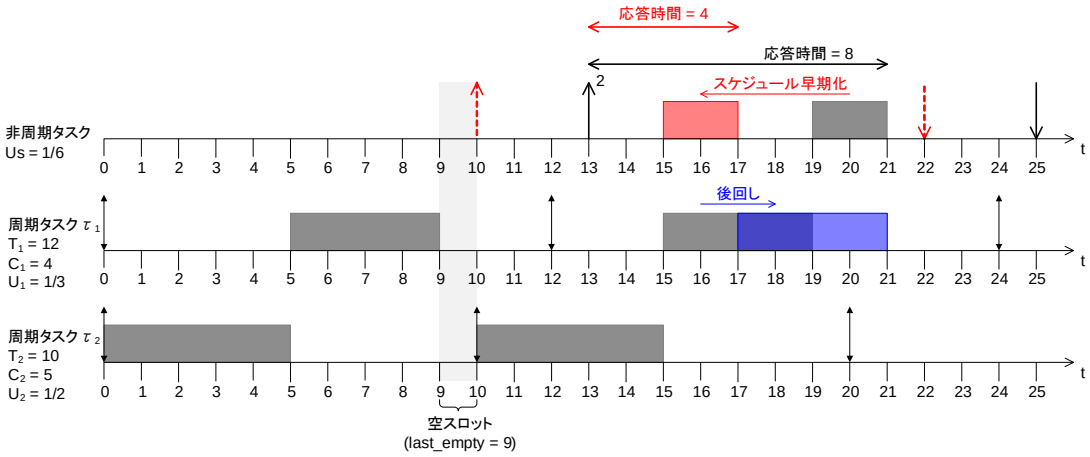

図 1 仮想リリース遡及の例（空スロットによる遡及限界）.

スクによるプロセッサ使用率は $U_p = 4/12 + 5/10 = 5/6$ となり，TBS のバンド幅は $U_s = 1 - 5/6 = 1/6$ となる．時刻 13 で，実行時間が 2 である非周期タスクの起動要求が発生し，TBS により計算されるデッドライン時刻は 25 となる．結果的に，当該非周期インスタンスは時刻 21 で実行を終了し，応答時間は $21 - 13 = 8$ となる．

一方，当該非周期タスクの起動要求が図中の赤色上向き破線矢印のように時刻 10 で発生したと仮定すると，デッドラインは時刻 22（赤色下向き破線矢印）となる．これは τ_1 の 2 つ目のインスタンスのデッドラインよりも早いため，τ_1 のインスタンスよりも優先して非周期インスタンスがスケジュールされ，結果的に時刻 17 で終了し，応答時間は $17 - 13 = 4$ となる．この際，実際の起動要求時刻である時刻 13 よりも過去のスケジュールは，本手法により変更されてはならないことに注意されたい．したがって，本手法により早期化されるデッドライン時刻は，時刻 10 から時刻 13 の間に実行されていたインスタンスのデッドラインよりも早くなってはならない．図中では，時刻 10 から時刻 13 の間は τ_2 のインスタンスが実行されており，このデッドラインは 20 であるため，非周期インスタンスの前倒しされたデッドライン時刻 22 よりも早いため，時刻 10 から時刻 13 の区間のスケジュールに影響を及ぼしていない．

3.2 仮想リリース遡及法の定義

仮想リリース時刻を過去のどの時点まで前倒し可能であるかを定義する．前倒しの限界点として 3 つの要因がある．それらは，前インスタンスのデッドライン，空スロット，および既使用最遅デッドラインである．

(1) 前インスタンスのデッドライン

サーバ内の前インスタンスのデッドラインを越えて仮想リリース時刻を前倒しすることは，TBS のスケジューラビリティの観点から不可能であるため，第一の限界要因となる．これは前述の通り，式 1 の max 項からも明らかである．この限界点検査を実現するためには前インスタンスのデッドライン時刻を記憶する必要があるが，TBS の実現において既に記憶されているものを使用可能である．ここで，2 節で述べたリソース回収法を併用することにより，前インスタンスのデッドラインがより過去の時刻になるように再計算されるため，仮想リリース遡及の適用可能性が拡大する．

(2) 空スロット

いかなるインスタンスも実行されなかった時間スロット（空スロット）を越えて仮想リリース時刻を前倒しすることは，このスロットで当該インスタンスが実行されなければならなかったことを意味し，過去のスケジュールを変更することになる．したがって空スロットは第二の限界要因である．図 1 はこの空スロット限界要因が適用される例である．この検査を実現するために，最後の空スロット番号を記憶する必要がある．

(3) 既使用最遅デッドライン

過去のスロットのうち空スロット以外は，何れかのタスクのインスタンスが実行されている．各スロットはそこで実行されたインスタンスの絶対デッドラインが関連付けられる．これをスロットの既使用デッドラインとする．ここで，あるスロットに着目し，そのスロットを対象インスタンスの仮想リリース時刻が前倒しによって越えることが可能かどうかを判断する状況を想定する．仮想リリース時刻がそのスロットを越えた場合，それに応じて前倒しされるデッドラインが当該スロットの既使用デッドラインよりも早くなる場合は，そのスロットは対象インスタンスが実行されていなければならない．これは過去のスケジュールを変化させることになる．したがって，この条件は第三の限界要因となる．より正確には，遡及プロセスの過程で計算されるデッドラインは，仮想リリースの前倒しで越えていく全スロットの既使用デッドラインよりも早くなることは許可されない．この検査を実現するため

には，過去の各スロットの既使用デッドラインと，遡及プロセスにおいて越えていくスロット群の既使用デッドラインの最大値を記憶する必要がある．

図 2 に第三の既使用最遅デッドライン要因による前倒しの限界点の例を示す．対象非周期タスクに対して時刻 5 で起動要求が発生する．TBS によってデッドライン計算およびスケジュールを行った場合は，デッドライン時刻は 11 となり，応答時間は 3 となる．一方，仮想リリース遡及を適用する場合を考える．スロット 4（時刻 4 と 5 の間のスロット）は周期タスク τ_2 のインスタンスが実行されており，このスロットに関連付けられる既使用デッドラインは 8 である．（配列 dl[] に過去の各スロットの既使用デッドラインが記憶されている．）ここで，この値が既使用最遅デッドラインとして max_dl に格納される．この場合，デッドライン時刻 11 は max_dl よりも大きいため，リリース時刻を仮想的に 1 スロット分過去に移動させることが可能である．この遡及プロセスが，仮想リリース時刻が 2，デッドライン時刻が 8 となるまで繰り返される．更なる遡及は不可能である．何故なら，デッドライン時刻が 7 となり，max_dl よりも小さくなるためである．したがってデッドライン時刻が 8 となったときに遡及プロセスが終了する．この図の例では，仮想リリース遡及により応答時間が短縮され，2 となっている．

3.3 仮想リリース遡及法のアルゴリズム

Algorithm 1 に仮想リリース遡及法のアルゴリズムを示す．アルゴリズム内で，r_k は k 番目の非周期インスタンスの実際の起動要求（リリース）時刻，vr_k は同インスタンスの仮想リリース時刻，d_k は同インスタンスのデッドライン時刻，d_{k-1} は $k-1$ 番目の非周期インスタンスのデッドライン時刻，C_k は k 番目の非周期インスタンスの実行時間，および U_s は TBS のバンド幅を意味する．（ここでは C_k として最悪実行時間を使用することにより TBS との組み合わせを想定するが，代わりに予測実行時間を使用することにより，適応型 TBS[4], [5] と組み合わせることも可能である．）

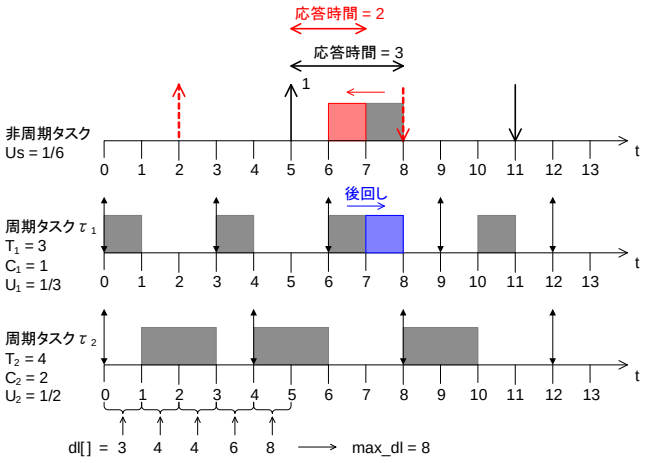

図 2 仮想リリース遡及の例（既使用最遅デッドラインによる遡及限界）．

Algorithm 1 Virtual Release Advancing

```
1:  max_dl ⇐ 0 /* Initialization of max_dl */
2:  vr_k ⇐ r_k     /* Initialization of vr_k */
3:  while TRUE do
4:    /* Reaching the limit of d_{k-1} */
5:    if vr_k ≤ d_{k-1} then
6:      vr_k ⇐ d_{k-1}
7:      d_k ⇐ vr_k + C_k/U_s
8:      break
9:    end if
10:   d_k ⇐ vr_k + C_k/U_s
11:   /* Reaching the limit of empty slot */
12:   if vr_k = last_empty + 1 then
13:     break
14:   end if
15:   if max_dl < dl[vr_k − 1] then
16:     max_dl ⇐ dl[vr_k − 1]
17:   end if
18:   /* Reaching the limit of prev-used max deadline */
19:   if d_k ≤ max_dl then
20:     break
21:   else
22:     /* Release advancing by one time slot */
23:     vr_k ⇐ vr_k − 1
24:   end if
25: end while
```

その他の変数として，$last\_empty$ は最後の空スロットの番号を保持し，配列 dl については，各要素 $dl[i]$ がスロット i の既使用デッドラインを保持する．また，max_dl がリリース遡及プロセスの中での既使用最遅デッドラインを保持する．

アルゴリズムの 1〜2 行目において，max_dl と vr_k の初期化を行い，続いて 5〜9 行目において，第一の前インスタンスのデッドライン要因の検査を行う．仮想リリース時刻が前インスタンスのデッドラインと同時刻か，あるいは早くなった場合は，仮想リリース時刻に d_{k-1} をセットし，対応するデッドラインを計算し，アルゴリズムを終了する．

第一の要因検査をパスした場合は，10 行目でデッドライン計算を行う．続いて 12〜14 行目において第二の空スロット要因の検査が行われる．仮想リリース時刻が最後の空スロットの終わりの時刻と一致する場合は，アルゴリズムを終了する．

続いて，15〜17 行目において max_dl と一つ前のスロットの既使用デッドラインを比較し，必要があれば既使用最遅デッドラインを更新する．19〜20 行目で第三の既使用最遅デッドライン要因が検査される．現時点でのデッドラインが max_dl と等しいか，早くなった場合は，遡及アルゴリズムを終了する．

全ての検査をパスした場合は，23 行目で 1 スロット分の遡及を行い，ループを繰り返す．

なお，文献 [6] で示された仮想リリース遡及アルゴリズムは周期タスクに特化したものであったが，本節におけるアルゴリズムは一般化されたものであり，周期タスク，非周期タスクのどちらに対しても適用可能である．

3.4 仮想リリース遡及法のスケジュール可能性

仮想リリース遡及法では，対象インスタンスが前倒しされた仮想リリース時刻にリリースされたものと仮定するが，過去のスケジュールを変化させることはない．言い換えると，当該インスタンスが実際に仮想リリース時刻でリリースされた場合でも，TBS によって結果として同一のスケジュールとなる．アルゴリズムにおいて特に第一の前インスタンスのデッドライン要因検査を行うことにより，TBS のデッドライン計算方法に順守しているため，TBS のスケジューラビリティの性質が保存される．すなわち，全タスクのプロセッサ使用率の合計が 100%以下であれば，タスクセットはスケジュール可能である．

3.5 仮想リリース遡及法の実装の複雑さと実行オーバヘッド

Algorithm 1 において，ループの繰り返し毎に 7 行目あるいは 10 行目で除算（C_k/U_s）が実行される．アルゴリズムの実行オーバヘッドを抑えるために，この除算は事前に計算しておき，ループ繰り返し時にはその結果値を使用することが有効である．

記憶オーバヘッドとして，$last\_empty$ と max_dl は単一データであるため問題とはならないが，配列 dl のサイズ（要素数）を考慮する必要がある．システム稼働時間を通して全てのスロットを配列要素として用意することは非現実的であるため，アルゴリズムは遡及スロット数，すなわちループ繰り返し回数に上限を儲け，dl サイズを最大遡及スロット数とする必要がある．最大遡及スロット数による応答時間短縮効果への影響については 4 節において評価する．

4. 評価

本節において，仮想リリース遡及法を TBS および，著者が過去に提案した適応型 TBS[5] と組み合わせた方式をシミュレーションにより評価する．タスクセットとして，プロセッサ使用率の合計（U_p）が 60%〜95%（5%毎）となる周期タスク群と，プロセッサ使用率が約 2%となる非周期タスク群を混在させた．また，非周期タスクを担当する（適応型）TBS のバンド幅を $U_s = 1 - U_p$ とした．

タスクセット内の周期タスク，非周期タスクの各パラメータは次のように生成した．

- 周期タスクの周期：平均 100 ティックとする指数分布乱数．
- 周期タスクの最悪実行時間：平均 10 ティックとする指数分布乱数．
- 周期タスクの実際の実行時間：最悪実行時間と等しい．
- 非周期タスクの到着間隔：1000 ティックあたり平均 1.25 回とするポアソン分布乱数．
- 非周期タスクの最悪実行時間：平均 8 ティックとする指数分布乱数．
- 非周期タスクの実際の実行時間：平均 4 ティックとする指数分布乱数．ただし，最悪実行時間を上限とする．

U_p 毎に 10 通りの周期タスクセットと 10 通りの非周期タスクセットを生成し，$10 \times 10 = 100$ 個の混在タスクセットに対して，観測時間を 100,000 ティックとしてシミュレーションを行った．

図 3 に非周期タスクの平均応答時間を，オリジナルの TBS による結果を 1 として正規化した値として示す．オリジナルの TBS を “TBS”，オリジナルの TBS に仮想リリース遡及法を組み合わせたものを “TBS+VRA”，適応型 TBS を “ATBS”，適応型 TBS に仮想リリース遡及法を組み合わせたものを “ATBS+VRA” と記す．また，VRA に付随する括弧内の数字は，仮想リリース遡及アルゴリズムのループ回数の上限（“Inf” は無限）を意味する．

図において，オリジナルの TBS に対して仮想リリース遡及法を組み合わせることにより，平均応答時間が短縮されることが確認できる．$U_p = 90\%$のときに，最大 14.8%の短縮に成功している．アルゴリズムのループ回数に上限を設けることにより，改善率は上限 80 のときに 14.2%，上限 20 のときに 8.50%に下がっている．

表 1 に TBS+VRA のシミュレーションにおける最大遡及スロット数と，アルゴリズムの平均ループ実行回数を示す．U_p が高いときはループ回数の上限を設けることにより平均ループ実行回数が削減されていることがわかる．

適応型 TBS は TBS に対し，$U_p = 95\%$のときに最大 58.9%の改善率となっている．適応型 TBS に対して仮想リリース遡及法を組み合わせた場合は，更なる改善はわずかであり，$U_p = 95\%$のときに TBS に対して 61.0%の改善率となっている．適応型 TBS によって既に十分に短いデッドラインが計算されるため，特に既使用最遅デッドライン

表 1 最大遡及スロット数と平均ループ実行回数（TBS+VRA）．

U_p (%)	最大遡及スロット数			平均ループ実行回数		
	上限無	上限 80	上限 20	上限無	上限 80	上限 20
60	17	17	17	1.26	1.26	1.26
65	31	31	20	1.38	1.38	1.38
70	32	32	20	1.72	1.72	1.71
75	41	41	20	1.93	1.93	1.92
80	77	77	20	2.81	2.81	2.65
85	121	80	20	4.53	4.52	3.72
90	175	80	20	9.15	8.85	5.86
95	354	80	20	23.95	17.49	7.91

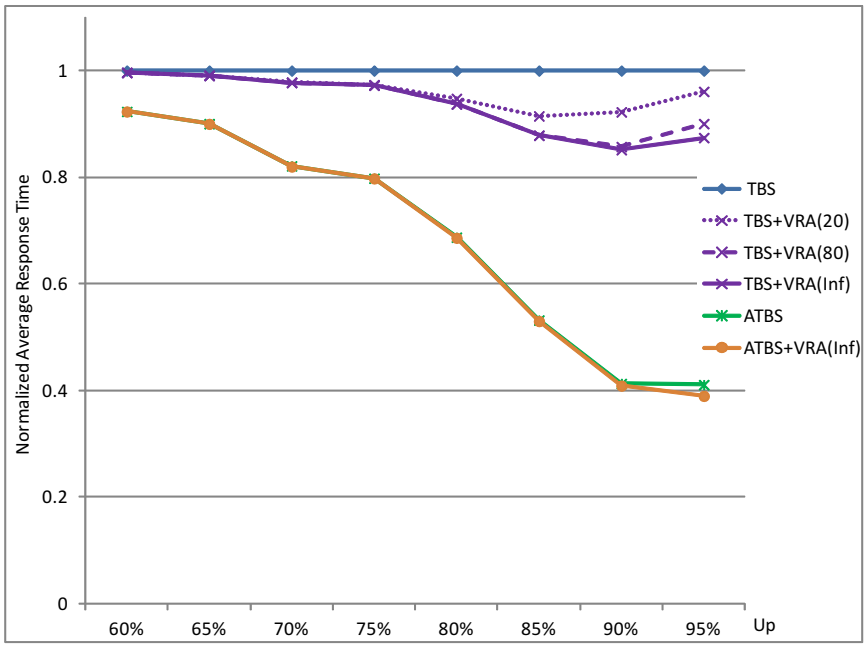

図 3 平均応答時間.

の条件によりリリース遡及が行われ難いことが理由である.

表 2 に ATBS+VRA のシミュレーションにおける最大遡及スロット数と，アルゴリズムの平均ループ実行回数を示す．表 1 と比較し，最大遡及スロット数および平均ループ実行回数の値が小さいことがわかる．このため，適応型 TBS に対する改善が小さくなっている.

5. おわりに

本稿では，非周期タスクの応答性の向上を目的とした仮想リリース遡及法を提案した．本手法は，タスクの起動要求が，実際よりも過去に発生したものと仮定し，対応するデッドラインを前倒しすることで優先度を向上させ，結果的に応答時間を短縮させるものである．同時に他の周期タスクのスケジュール可能性に影響を及ぼさないことが特徴である．シミュレーションによる評価では，平均応答時間が最大 14.8%短縮された．また，アルゴリズムのループ実行回数の上限値にしたがって平均応答時間短縮の効果が変化することが確認された．今後はアルゴリズム実行の詳細なオーバヘッドについての調査と，実アプリケーションに対する評価を行っていく予定である.

表 2 最大遡及スロット数と平均ループ実行回数（ATBS+VRA）.

U_p (%)	最大遡及スロット数			平均ループ実行回数		
	上限無	上限 80	上限 20	上限無	上限 80	上限 20
60	1	1	1	1.06	1.06	1.06
65	2	2	2	1.12	1.12	1.12
70	3	3	3	1.12	1.12	1.12
75	2	2	2	1.12	1.12	1.12
80	4	4	4	1.19	1.19	1.19
85	7	7	7	1.31	1.31	1.31
90	16	16	16	1.85	1.85	1.85
95	66	66	20	4.49	4.49	4.11

参考文献

[1] Liu, C.L., Layland, J.W.: *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*, Journal of the Association for Computing Machinery, Vol.20, No.1, pp.46–61 (1973).
[2] Buttazzo, G.C.: *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Third Edition, Springer, (2011).
[3] Spuri, M., Buttazzo, G.C.: *Efficient Aperiodic Service under Earliest Deadline First Scheduling*, Proc. of IEEE Real-Time Systems Symposium, pp.2–11, IEEE Computer Society, San Juan (1994).
[4] Tanaka, K., *Adaptive Total Bandwidth Server: Using Predictive Execution Time*, Proc. of 4th IFIP TC 10 International Embedded Systems Symposium (IESS), Springer, pp.250–261, Paderborn (2013).
[5] 田中 清史：適応型スケジューリングによる平均応答時間の短縮法 — 実行時間見積り方法の影響 —，組込みシステムシンポジウム 2013（ESS2013）論文集，pp.87–94，東京 (2013).
[6] 田中 清史：周期タスクのジッタを軽減するリアルタイムスケジューリング，組込みシステムシンポジウム 2014（ESS2014）論文集，pp.36–45，東京 (2014).
[7] Tanaka, K.: *Adaptive EDF: Using Predictive Execution Time*, ACM SIGBED Review, Vol.10, No.4, pp.41–44 (2013).
[8] Tanaka, K.: *Improvement of Adaptive EDF*, ACM SIGBED Review, Vol.11, No.3, pp.40–43 (2014).
[9] Spuri, M., Buttazzo, G.: *Scheduling Aperiodic Tasks in Dynamic Priority Systems*, The Journal of Real-Time Systems, Vol.10, No.2, pp.179–210 (1996).
[10] Spuri, M., Buttazzo, G., Sensini, F.: *Robust Aperiodic Scheduling under Dynamic Priority Systems*, Proc. of IEEE Real-Time Systems Symposium, pp.”210–219, Pisa (1995).