Vol.2009-IOT-5 No.4
2009/5/28

NAP-Webにおける次回アクセス補助機能

加地智彦[†1]　最所圭三[†2]

供給能力以上のアクセスが集中した際に次回アクセスを保証することでユーザの不満を抑えながらサービスを継続する Web システム NAP-Web の開発を行っている．NAP-Web では，次回アクセスが可能となる時間を予測する必要があるが，正確な予測は困難であるため，幾分かの余裕を加えた時間をクライアントに対して教えている．しかし，余裕をつけることにより，次回アクセス処理の効率が落ちてしまう．この効率を改善するために,「次回アクセス補助機能」を NAP-Web に持たせた．次回アクセス補助機能は，クライアントが教えられた時間になる前に，サーバの状態を調査し．サーバに空きがあれば次回アクセスすることをを許可する．本稿では，次回アクセス補助機能について述べた後，クライアントがサーバの状態を調査するタイミングを決定する方法について述べ，評価を行う．

Next Access Support Function for NAP-Web

Tomohiko Kaji[†1] and Keizo Saisho[†2]

NAP-Web is one of web access control systems for congested server. The system guaratees rejected client to be accepted next access. Though it tells when he can access again, it is difficult to predict right time. Therefore, the told time has margin. However, the margin worsen server utilization. Next access support function is introduced to improve the utilization. The function permits client to check server condition before the told time. If server can afford to reply, client can access again. This paper describes next access support function, the way to decide when client check server condition, and evaluation of them.

†1 香川大学大学院工学研究科
Graduate School of Engineering, Kagawa University
†2 香川大学工学部
Feculty of Engineering, Kagawa University

1. はじめに

インターネットユーザの増加は，サービスの高度化，多様化を招いている．WWW サービスを提供する Web サーバへの要求は年々高まっており，サービス提供者は自身のサービス基盤が需要に対して十分にスケールしているか常に考慮しなければならない．だが，Web サービスの需要予測は困難である．例えば，TV ニュースに Web サイトが紹介されたりすれば，突発的にアクセスが集中することがある．そのような際に,「現在混雑しています．しばらくお待ちください」といったメッセージを表示する Web サイトが多いが，いつまで待てばいいのか分からないため，ユーザの不満は大きくなる．また，いつまで待てばわからないのでユーザはサーバの混雑が解消していない状態でも，再アクセスを繰り返し，その対応のために余計にサーバの負荷が高まることがある．

我々は，このような場合にユーザに対して「いつまで待てばいいか」を知らせ，保証する Web システム「NAP-Web」の開発を行っている[1)]．NAP-Web は「動的負荷制御機能」,「次回アクセス保証機能」,「次回アクセス補助機能」の三つの機能により構成される.「動的負荷制御機能」はサーバの状態に応じて処理しきれないアクセスを拒絶する.「次回アクセス保証機能」は「動的負荷制御機能」によってアクセスを拒絶されたクライアントに対し，いつまで待てばサーバがアクセスを処理できるようになるかを予測し，その時間をチケットの形で伝える．チケットを持ったクライアントが次回アクセスを行った時，チケットの内容を見て，アクセスの可否を判断する．ここで,「次回アクセス保証機能」はいつまで待てばサーバがアクセスを処理できるようになるか（次回アクセス可能時間）を予測する必要があるが，正確な予測は困難である．予測が外れた場合，伝えた時間よりも実際の次回アクセス可能時間が遅れると，ユーザの不満は大きくなると考えられる．逆に，伝えた時間よりも次回アクセス可能な時間が早まった場合，ユーザの不満が大きくならないと推測できる．そのため，NAP-Web では幾分かの余裕を加えた時間を，次回アクセス可能時間としてクライアントに伝えることにした．しかし，この余裕の分，サーバの利用率は下がってしまう.「次回アクセス補助機能」は，サーバの利用率を向上させるために，クライアントからの次回アクセスを補助する機能である．本機能においては，クライアントはサーバから次回アクセス可能時間を伝えられると，伝えられた時間になる前に，サーバの状況を調べる．もし，サーバの状態が空いているようであれば，クライアントに対して次回アクセスを許可する．

本稿では,「次回アクセス補助機能」を実現するための仕組みについて述べた後，クライアントがサーバの状態を調べるタイミングを決定する方法を二つ提案する．一つは，静的な

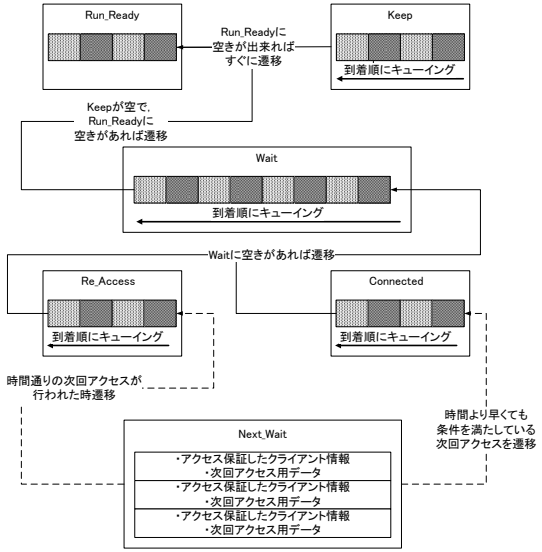

図 1 アクセスグループの関係
Fig. 1 Ralationship between Access group

モデルを用いてタイミングを決定する方法であり，もう一つは，サーバが過去に処理した統計情報を用いて動的にタイミングを決定する方法である．それぞれの方法について，シミュレーションを行った結果を報告する．

2. アクセススケジューリング機構

アクセススケジューリング機構は，クライアントからのアクセスをサーバの状態に応じて以下の 6 つにグループ分けし，順序制御を行うための機構である．6 つのアクセスグループの関係を**図 1** に示す．なおこれらのうち Run_Ready，Wait，Next_Wait，Re_Access については文献[1] で詳しく述べている．

Run_Ready：通常通り OS によって処理されるのを待っているアクセスグループである．このアクセスグループの処理が完了すると，後述の Wait キューの先頭からアクセスがこのグループに移動する．

Wait：Run_Ready からあふれたアクセスが入るアクセスグループである．このアクセスグループの最大数は，サーバの処理状態から指定秒数内に Run_Ready に移動できると推測される数に制限される．

Next_Wait：Wait からもあふれたアクセスが入るアクセスグループである．このグループに入ったアクセスは応答を一度拒絶されるが，いつになれば次回アクセスが可能になるかが書かれたチケットをクッキーの形で渡される．拒絶されたアクセスの情報はサーバ側で記録される．

Re_Access：チケットを持ったクライアントがチケットに書かれた時間に再アクセスを行った時に割り振られるグループである．このグループに入ったアクセスは到着順にキューイングされる．Wait キューに空きができると，先頭のアクセスが Wait キューの最後尾に移動する．

Keep：複数のファイルで構成されている Web ページを取得するためのアクセスグループである．Run_Ready に割り振られたアクセスが，ごく短期間に再アクセスを行うと Keep アクセスグループに割り振られる．Keep アクセスグループに割り振られたアクセスはキューを作り，. Wait より優先して処理される．

Connected：次回アクセス補助機能により，チケットに書かれた時間よりも早い時間に行われた再アクセスを処理するためのアクセスグループである．チケットに書かれた時間よりも早く行われた再アクセスが後述の条件を満たすと，このアクセスグループに振り分けられる．Connected アクセスグループは到着順にキューを作り，Wait に空きが出来ると先頭のアクセスが Wait キューの最後尾に移動する．

3. 次回アクセスチケット機構

次回アクセスチケット機構は，Next_Wait に割り振られたアクセスに対して，次回アクセスが可能な時間が書かれたチケットの配布と照合を行う機構である．チケットには，チケット ID とチケット開始時刻，チケット終了時刻，チケット番号，スロット番号が記述される．なお，チケット開始時刻にクライアントが再アクセスを行えるとは限らないため，チケットはそれぞれ有効期間を持つ．この有効期間の間に行われた再アクセスが Re_Access に割り振られることになる．また，この有効期間はサーバの予測どおりに処理が進まなかった時のために付け加えられる余裕時間となる．

チケットの開始時刻はスロットと呼ぶ概念を用いて決定される．スロットとは，次回アクセスを処理するための時間を一定の間隔で区切ったものである．この間隔をスロット期間と呼び，サーバ管理者が決定する．スロット期間は，そのスロットに属する 1 番最初のチケットの開始時刻から，1 番最後のチケットの終了時刻までとなる．

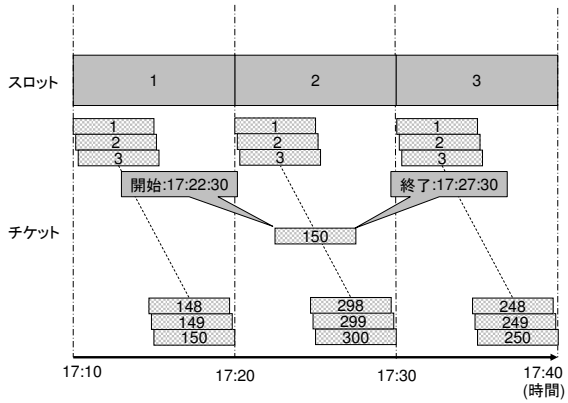

図 2 スロットの説明
Fig. 2 Explaination of Slot.

サーバの処理状態から，1 スロットあたりに処理が可能なアクセスの数を予測し，その数をスロットで配布するチケットの最大枚数とする．チケット最大枚数を計算する際，チケット有効期間は余裕時間として扱うので，その分をスロット期間から引いておく．そのスロットの最大枚数を計算する時のサーバの平均処理時間を $Avg(T_{proc})$，スロット期間を T_{Slot}，チケット有効期間を T_{Ticket} とした時，m 番目のスロットのチケット枚数 $Max(Slot_m)$ は，式 1 で表される．

$$Max(Slot_m) = (T_{Slot} - T_{Ticket})/Avg(T_{proc}) \quad (1)$$

m 番目のスロットに属する k 番目のチケットの開始時刻 $T_{Ticket_{m,k},start}$ は，m 番目のスロットの開始時刻 $T_{Slot_m,start}$ を用いて，式 2 で表される．

$$T_{Ticket_{m,k},start} = T_{Slot_m,start} + (k \times (T_{Slot} - T_{Ticket}))/Max(Slot_m) \quad (2)$$

また，m 番目のスロットに属する k 番目のチケットの終了時刻 $T_{Ticket_{m,k},end}$ は式 3 で表される．

$$T_{Ticket_{m,k},end} = T_{Ticket_{m,k},start} + T_{Ticket} \quad (3)$$

図 2 を用いて，チケット発行の例を説明する．ここでは，スロット期間 T_{Slot} は 10 分間，チケット有効期間 T_{Ticket} を 5 分間としている．スロット番号 2 番の開始時刻 $T_{Slot_2,start}$ を 17:20:00 とすると，スロットで一番最初に発行されるチケットの開始時刻 $T_{Ticket_{2,1},start}$ は 17:20:00，終了時刻 $T_{Ticket_{2,1},end}$ は 17:25:00 となる．サーバの平均処理時間 $Avg(T_{proc})$ を 1 秒とすると，スロットで処理できるチケット最大枚数 $Max(Slot_2)$ は 300 枚となり，

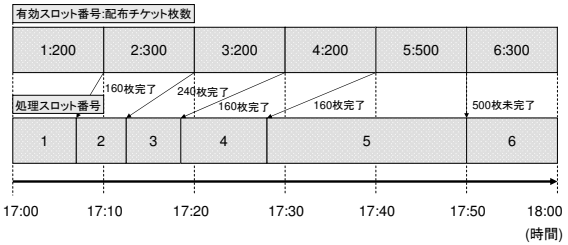

図 3 有効スロット番号と処理スロット番号
Fig. 3 Valid and Processig slot number

150 枚目のチケットの開始時刻 $T_{Ticket_{2,150},start}$ は 17:22:30，$T_{Ticket_{2,150},end}$ は 17:27:30 となる．

4. 次回アクセス補助機能を実現する仕組み

図 2 を見て分かるとおり，もし予測どおりにサーバの処理が進み，かつ，全てのチケットを持った再アクセスがチケットの開始時刻どおりに行われた場合，次のスロットのチケットが配布されていたとしても，チケット有効期間の間，サーバは全く次回アクセスの処理を行わないことになり，サーバの利用率が低下してしまう．この無駄を無くすために，チケットの開始時刻より早く再アクセスが行われたとしても，サーバの状態が空いているのならば，そのアクセスの処理を許可する仕組みである次回アクセス補助機能を用意した．

次回アクセス補助機能を実現するためには，サーバ側とクライアント側の両方で対応が必要となる．サーバ側は，伝えた時刻よりも早く次回アクセスが行われた時，サーバの状態に応じて，そのアクセスを処理する．クライアント側は，サーバの状態を確認し，負荷状態でなければ次回アクセスを行う．この機能を持たないクライアントのため，時刻通りに次回アクセスが行われた場合も処理を行えるようにしておかなければならない．クライアント側のこの昨日はプラグインの形で開発する予定である．

以下，チケットを持っている再アクセスをどのようにサーバ側で受け入れるのか説明する．

4.1 有効スロット番号と処理スロット番号

サーバの状態を把握するために，有効スロット番号と処理スロット番号を用意する．有効スロット番号と処理スロット番号の概要を**図 3** に示す．有効スロット番号とは，サーバが予測どおりに処理を進めているときに，処理を行っているスロット番号である．図 3 では，スロットを生成し，チケットを発行していた場合，時計が 17:10 から 17:20 の間，スロット

番号 2 番が有効スロット番号となる．処理スロット番号とは，サーバが実際に処理を進めているスロット番号であり，サーバの処理が予測よりも早く進んでいる場合は，有効スロット番号よりも早く進む．処理スロット番号が進む条件は，スロットで配布したチケットのうち，何割かの次回アクセスが完了することである．例えば，80%の次回アクセスの処理が完了した時をその条件とすると，図 3 の例であればスロット番号 1 番のチケットのうち 160 枚が処理できた時，処理スロット番号は 2 番に進む．有効スロット番号と処理スロット番号を比較することで，サーバの状態を把握することができる．有効スロット番号と処理スロット番号が等しい時，サーバの処理は予測どおりに進んでいることになるし，有効スロット番号が処理スロット番号より小さいとき，サーバでの処理は予測より早く進んでいる．なお，処理スロット番号を進める条件である次回アクセスの処理が完了した割合は，サーバの運用データから得た，クライアントが実際に次回アクセスを行う確率とするのが望ましい．

4.2 Connected への接続条件

2 章で述べたとおり，次回アクセス補助機能により，伝えた時間よりも早く次回アクセスが行われた時，そのアクセスを処理するための Connected アクセスグループを用意した．Connected アクセスグループはチケットの開始時刻の前に行われた次回アクセスが，1）有効スロット番号と処理スロット番号が等しく，チケットのスロット番号が処理スロット番号より等しい，2）処理スロット番号が有効スロット番号と等しく，チケットのスロット番号が処理スロット番号より小さい，のどちらかの条件を満たし，かつ．

$$Q_{limit} - Q_{count} \geq M \quad (4)$$

の条件を満たした時に振り分けられる．このとき，Q_{limit} はキューの最大値，Q_{count} は現在キューに入っているアクセスの数，M はサーバ管理者がサービスの状態に応じて決定するキューの余裕分である．式 4 により，次回アクセス補助機能を用いないクライアントが，チケットに書かれたとおりに次回アクセスを行った場合の処理を保証する．

5. クライアントの振る舞い

次回アクセス補助機能において，クライアント側は教えられた時間よりも早くサーバの状態を調べに行く必要がある．ここで問題となるのは，どのタイミングで状態を調べに行くか，である．あまり頻繁に状態を調べに行けば，その行為自体がサーバの負荷になってしまうし，逆に状態を調べに行く頻度を減らしすぎると，サーバの手が空いた時に上手く再アクセスを行わせることができなくなってしまう．そこで，本稿ではクライアントがサーバの状態を調べに行くタイミングを決定する以下の二つの方法を提案する．

(1)　残り時間の半分ごとに調べに行く（静的な）方法
(2)　統計情報を用いて調べに行く（動的な）方法

(1) の方法では，まずチケットを発行されると，チケットに書かれた時刻までの半分の時間がたったら，サーバの状態を調べに行く．もし Connected に振り分けられれば，そのまま次回アクセスが処理されるのを待ち，振り分けられなければ，またその時刻からチケットに書かれた時刻までの半分の時間がたったらサーバの状態を調べに行くことを繰り返す．(2) の方法では，サーバは処理スロット番号が進むたびに，予測したスロット期間と実際のスロット期間の差分を記録し，その統計値を取る．この情報はチケットと共に渡される．そして，クライアントは処理が予測より早くなっているとき（つまり処理スロット番号が有効スロット番号より大きくなった時），その統計値を用いてサーバに状態を聞きに行くタイミングを決定する．処理が予測どおりに進んでいる場合は 1 と同じように，残り時間の半分が過ぎたタイミング毎に調べに行く．例えば，スロットの期間がそれぞれ 5 分で，スロット 1 からスロット 3 が予測よりも平均 2 分早く処理されていて，現在スロット 4 の次回アクセスが処理されていたとする．このタイミングでスロット 7 のチケットが発行された時，スロット 7 のアクセスが渡されるチケットに書かれた待機時間は 15 分〜20 分の間になるが，1 つのスロットあたり 2 分早く進んでいるので，クライアントは 3 スロット分の 6 分早い 9 分〜14 分経つとサーバの状態を聞きに行く．この時，処理スロット番号が 7 番まで進んでいて，4.2 節の条件を満たしていれば，Connected に振り分けられる．もし Connected に振り分けられる条件を満たしていなければ，チケットに書かれた時刻までの残り時間の半分を待つことを繰り返す．もし，処理スロット番号が 5 番か 6 番であれば，サーバはまた統計情報をクライアントに与える．クライアントはその統計情報を用いて待機時間を決定しなおす．

6. 評　　価

次回アクセス補助機能のうち，サーバ側の対応部分を NAP-Web に追加した．本章では，5 章で述べたクライアントがサーバの状態を調べに行くタイミングを決定する二つの方法を評価する．二つの方法をシミュレートするスクリプトを用意し，サーバに対してアクセスを行わせる．各アクセスはサーバにおかれた掲示板 CGI に 10KB の文字列を書き込む．このアクセスを行うスレッドを (1)1000 個同時に起動し，各スレッドは 50 回アクセスを繰り返すスクリプトと，(2)1 分間に 50 個ずつ起動するのを 20 分間続け，各スレッドは実験を開始してから 20 分経つまでアクセスを繰り返すスクリプトを用意した．(最後に生成されたスレッドは 1 分間だけ実行されることになる）実験に使用するサーバは Core2Duo E8500,

表 1 実験パラメータ
Table 1 Experimental Parameter

データ項目	設定値
Run_Ready の閾値	10
Wait+Re_Access 滞在期間 (秒)	4
Connected 滞在時間 (秒)	10
スロット期間 (秒)	30
チケット有効期間 (秒)	30
処理スロット移行割合 (%)	80

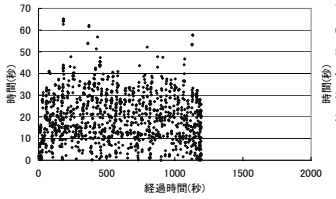

図 4 (1)α
Fig. 4 (1)α

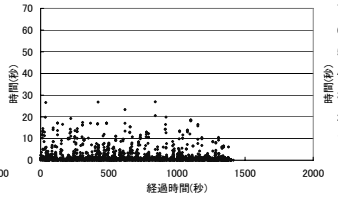

図 5 (1)β
Fig. 5 (1)β

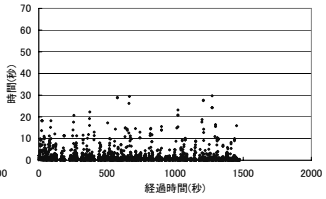

図 6 (1)γ
Fig. 6 (1)γ

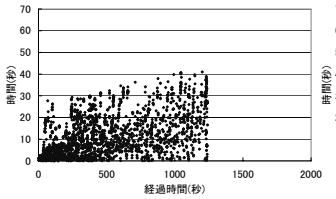

図 7 (2)α
Fig. 7 (2)α

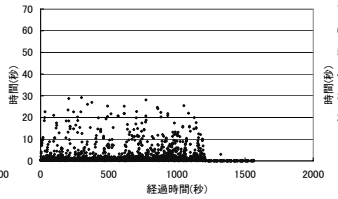

図 8 (2)β
Fig. 8 (2)β

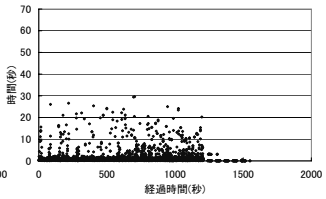

図 9 (2)γ
Fig. 9 (2)γ

表 2 統計値
Table 2 Statistics

	(1)			(2)		
	α	β	γ	α	β	γ
平均応答時間 (秒)	21.43	0.33	0.35	11.05	0.38	0.37
最大応答時間 (秒)	65.68	26.43	29.73	52.63	29.18	29.58
スループット (アクセス/秒)	39.84	35.33	33.83	43.87	46.16	46.19
リクエスト成功率 (%)	95.06	99.9	99.6	98.2	99.9	99.7

メモリ 2GB の環境である．以下，NAP-Web を利用していない状態の実験を α，残り時間の半分ごとに調査する方法を用いた実験を β，統計状態を用いて調査する時間を決定する方法を用いた実験を γ とする．これらと記号 (1)，(2) と組み合わせて，(1) のスクリプトを用いた α の実験を (1)α，(2) のスクリプトを用いた β の実験を (2)β のように表す．なお，この実験で使用した NAP-Web のパラメータを**表 1** に示す．(1)α から (2)γ までの実験結果のうち，応答時間についてのグラフを**図 4** から**図 9** に示す．これらのグラフの横軸は経過時間，縦軸は応答時間を示す．また，**表 2** に応答時間についての統計値を示す．

まず，(1) に関する図 4 と**図 5**，**図 6** の 3 つのグラフを見比べても分かるとおり，急激にアクセスが集中した時，応答時間が極端に延びる傾向にあったのが，NAP-Web を導入することによりアクセスが分散し，応答時間が平滑化されている．このことは，表 2 で (1)α の平均応答時間が，21.43 秒なのに対し，(1)β と (1)γ の平均応答時間がそれぞれ 0.33 秒，0.35 秒と短くなっていることからも分かる．また，次第にアクセスが集中する (2) の場合でも同様に，**図 7** から図 9 と表 2 をそれぞれ比較すると平均応答時間が短くなっていることが分かる．

次に，次回アクセスについて評価する．NAP-Web 導入後の実験である (1)β，(1)γ，(2)β，(2)γ のそれぞれの待機時間の結果を**図 10** から**図 13** に示す．また，次回アクセスに関する各データの統計値を**表 3** に示す．ここでは次回アクセスをするための待ち時間とその処理時間の和を待機時間と呼ぶことにする．また，縮小時間とはチケットを配布されたアクセスが次回アクセス補助機能によってどれだけ早く次回アクセスが可能になったかを示し,，指示時間とは，チケットの開始時刻とクライアントがチケットを受け取った時刻との差とする．スロット間隔とは，あるスロットの最後のチケットを持った次回アクセスが完了してから，その次のスロットの最初のチケットを持った次回アクセスが完了するまでの時間である．この間隔が短いほど，次回アクセスの処理が無駄なく出来ていると言える．そして，リトライ数とは Connected に振り分けられるまでのクライアントがサーバの状態を調査する回数である．この回数が少ないほど良いタイミングでサーバの状態を調査できていると言える．表 3 より，待機時間の平均は γ の方が短くなる傾向があるが，図 10 と**図 11**，**図 12** と図 13 を比較すると，γ の方が待機時間のばらつきが大きい．また，スロット間隔の平均値と最大値の差が大きいこと，リトライ数の分散が大きいことなどからも，上手くタイミングを決定できている時とそうでない時の差が β よりも大きいことがわかる．

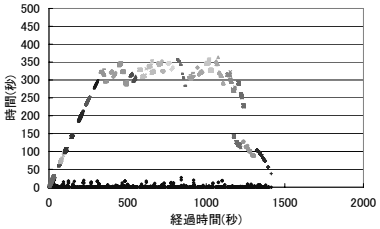

図 10 (1)β（待機時間含む）
Fig. 10 (1)β (with Waiting time)

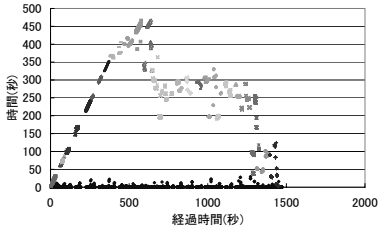

図 11 (1)γ（待機時間含む）
Fig. 11 (1)γ (with Waiting time)

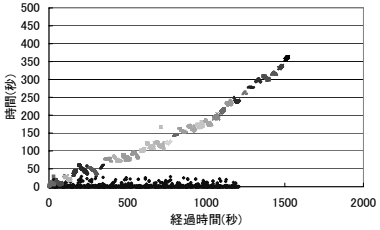

図 12 (2)β（待機時間含む）
Fig. 12 (2)β (with Waiting time)

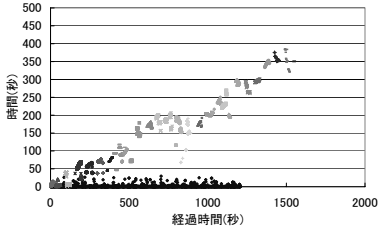

図 13 (2)γ（待機時間含む）
Fig. 13 (2)γ (with Waiting time)

表 3 次回アクセスの統計値
Table 3 Statistics of Next Access

	(1)		(2)	
	β	γ	β	γ
平均待機時間 (秒)	254.95	235.06	145.48	110.72
平均縮小時間 (秒)	97.05	31.30	145.88	20.72
平均指示時間 (秒)	352.04	266.36	291.36	131.44
発生回数	3398	3049	5311	3108
リトライ数平均	1.84	1.58	1.00	1.19
リトライ数分散	0.42	0.56	0.03	0.83
リトライ数合計	6255	4825	5315	3728
平均スロット間隔 (秒)	5.51	5.03	13.45	8.69
最大スロット間隔 (秒)	39.132	50.59	16.31	45.35

7. まとめと今後の課題

NAP-Web における次回アクセス補助機能の仕組みについて述べた．そして，次回アクセス補助機能で必要なクライアントがサーバに状態を調査しに行くタイミングを決定する二つの方法を提案した．二つの方法について現在まで評価した結果，残り時間の半分ごとに調査する方法の方が安定して良いタイミングを決定する傾向があることが分かった．しかしながら，条件を変えた時にこの傾向が現れるかの評価を行えていない．例えば，様々なコンテンツに対してアクセスを行う場合についても評価を行わなければならない．また，統計データを利用する方法についてはそのばらつきが少なくなるように洗練する必要がある．今後，これらの課題を解決し，より効率的な次回アクセスを可能にする仕組みを構築したい．

参 考 文 献

1) 加地智彦，最所圭三：過負荷時のユーザの不満を抑えるために次回アクセスを保証する Web システム，情報処理学会論文誌，Vol.50，No.2，pp.872-881(2009)