情報処理学会論文誌 Vol. 50 No. 2 872–881 (Feb. 2009)

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

加地智彦[†1] 最所圭三[†2]

インターネット接続の一般化により，多様かつ高度なサービスがWebで提供されるようになってきた．サービスを提供するWebサーバへの要求は大きくなっているが，アクセス集中などにより，Webサーバの能力以上の要求が行われると，サーバの応答性能が著しく低下し，満足なサービスを受けられなくなったユーザが不満を持ち，サービス提供者の信用が大きく低下する．サーバの過負荷問題への対策は，サービス提供者の大きな課題となっている．過負荷問題の対策として，負荷分散技術が積極的に用いられている．しかしながら，複数台の計算機を準備，運用するコストは大きく，サービス需要の予測が困難なこともあいまって，十分に計算機を準備できず，負荷分散技術を効果的に利用できない場合がある．本論文では，このような状況により，サーバが過負荷状態に陥りそうなとき，ユーザからの初回アクセスを，到着順序を基準に受け入れるか拒否するか決定し，拒否されたユーザに対しては，「次回いつになればアクセスが可能になるか」を保証することによってユーザの不満を抑えるWebシステム「NAP-Web」を提案する．NAP-Webの実現により，計算機を追加しなくても，Webサーバの応答性能を保ちながら，ユーザの不満の少ない，サービスの継続が可能となる．NAP-Webの実現には，動的負荷制御，次回アクセス保証，次回アクセス補助の3つの機能が必要である．これら3つの機能のうち，動的負荷制御機能，次回アクセス保証機能を実現するためのWebアクセススケジューリング機構と次回アクセスチケット機構のプロトタイプを，Apacheモジュールmod_nap_webとして実装した．mod_nap_webを導入したWebサーバを評価した結果，次回アクセスを保証することによって，応答性能を保ちながらサービスを行うことが可能であることが分かった．

A Web System Promising Next Access to Control User's Complaint for Overloaded Server

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

As connection of the Internet is popular, various and sophisticated services are provided on the Web and demands to Web server are hard. If the number of requests exceeds capacity of Web server, response performance of the server drops markedly. When a server falls into overload, users can not receive the service, and they complain and their confidence to the service provider declines. Therefore, service providers are worried about server overload problem. They adopt server load balancing technique actively, as countermeasure to overload problem. However, the technique is not always effective. Since cost for operation of many computers is high and estimation of service demand is difficult, they might not supply enough computers. In this paper, a Web system "NAP-Web" is proposed. NAP-Web controls user's complaint using the following method. When server would fall into overload, the system determines whether it accepts access according to arrival order. It promises next access to the rejected access. NAP-Web can continue to service without additional computer, keeps it's response performance, and keeps user's complaint to a minimum. NAP-Web is comprised of three functions: dynamic load control, next access promising, and next access support. Access scheduling mechanism and next access ticket mechanism are designed for dynamic load control and next access promising, respectively. Two mechanisms are implemented as an apache module mod_nap_web. The result of evaluation of server with mod_nap_web shows it can avoid overload and promise next access.

1. はじめに

インターネット接続の一般化により，多様かつ高度なサービスがWebで提供されるようになってきた．サービスを提供するWebサーバへの要求は大きくなっているが，アクセス集中などにより，Webサーバの能力以上の要求が行われると，サーバは過負荷状態に陥り，応答性能が著しく低下する．サーバの過負荷問題への対策は，サービス提供者の大きな課題となっている．サーバが過負荷状態に陥ると，満足なサービスを受けられなくなったユーザが不満を持ち，サービス提供者の信用が大きく低下する．過負荷問題の対策として，負荷分散技術が積極的に用いられている．しかしながら，複数台の計算機を準備，運用するコストは大きく，サービス需要の予測が困難なこともあいまって，十分に計算機を準備できず，負荷分散技術を効果的に利用できない場合がある．たとえば，サービスを開始する前にユーザ数を把握することは困難であるため，サービス開始直後の需要予測は失敗することが多い．そのため，前評判の高いサービスでは，恒例行事のように開始時のサーバダウンが頻繁に発

†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
Faculty of Engineering, Kagawa University

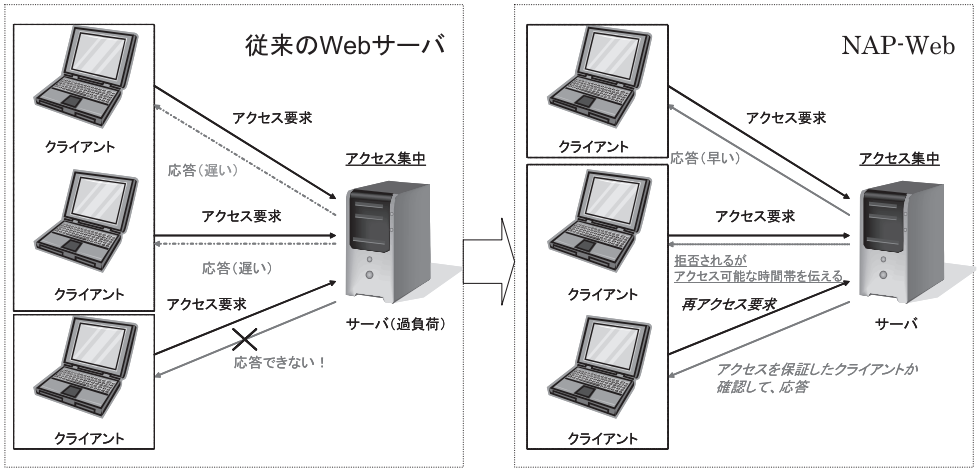

図 1　NAP-Web の概要
Fig. 1　Outline of NAP-Web.

生している．負荷分散技術を用いても，見積りよりサービス需要が大きく，用意したサーバ環境で十分にサービスが提供できない場合，サービスの提供を受けることのできなかったユーザの不満や苦情は強く，サービス提供者の信用が大きく低下してしまう．

そのような状態に陥ったとき，ユーザからのアクセスを制限することで過負荷状態を避けつつ，サーバの応答性能を低下させないようにするが，アクセス拒絶時には，「しばらくの間お待ちください」とユーザに伝える Web システムもある．この場合，ユーザは具体的にどれくらい時間がたてばアクセスできるようになるのか分からないため，サーバが過負荷状態から脱していないときでも，再アクセスを行うことが多く，さらに負荷をサーバに与えてしまう．再アクセスを失敗することが続くならその不満は大きくなる一方である．

本論文では，サーバが過負荷状態に陥りそうなとき，ユーザからの初回アクセスを，到着順序を基準に受け入れるか拒否するかを決定し，拒否されたユーザに対しては，「次回いつになればアクセスが可能になるか」を保証することによってユーザの不満を抑えながら，サーバの応答性能を低下させない Web システム「NAP-Web」を提案する．

図 1 に NAP-Web の概要を示す．図 1 の左側が従来の Web サーバ，右側が NAP-Web を導入した Web サーバを示す．従来の Web サーバでは，過負荷状態が発生した際，クライアントへの応答が極端に遅くなる，もしくは，タイムアウトする．このような場合に，ユーザの感じる不満は，1) サービスが受けられたかどうか判断するまで，最悪タイムアウトまで待たなければならない，2) いつになればサービスが受けられるようになるかまったく分からない，の 2 点である．1) の不満を解決するためには，ある一定時間内にサービスが可能かどうかを判断し，可能でない場合は，即座にユーザにそのことを通知する機能が必要である．2) の不満を緩和するためには，サービス可能となる時間を予測し，その時間をユーザに伝え，伝えられた時間に行われる再アクセスに対してサービスを保証する機能が必要である．

本研究では，上記の 2 点を解決するために，以下の機能を NAP-Web に持たせることにした．サーバに対してユーザからのアクセスが集中した際に，システムの負荷状態に応じてアクセスの可否を決定する（動的負荷制御機能）．初回アクセスが失敗した場合，そのことをユーザに即座に伝えることで 1) の不満を解消する．動的負荷制御機能によってアクセスを拒絶されたユーザに対しては，次回いつになればアクセスが可能になるかを保証する（次回アクセス保証機能）ことで，2) の不満を緩和する．また，次回アクセスを保証するためには，サーバ側でいつになれば過負荷状態から脱することができるか予測する必要があるが正確な予測は困難である．このため，予測が外れた際の補助機能を用意する（次回アクセス補助機能）．たとえば，サーバがクライアントに伝えた時間帯よりも早く過負荷状態より脱

した場合，クライアントにそのことを伝えることが考えられる．

本論文では，これらの機能のうち，動的負荷制御機能と次回アクセス保証機能を実現するための，アクセススケジューリング機構と次回アクセスチケット機構について述べる．アクセススケジューリング機構は，サーバ上でのアクセス処理を時系列に順序づけ，処理済みアクセスの統計情報から，一定時間内に処理を完了できるアクセス量を推測し，その値に基づいて，アクセスをグループ化し，スケジューリングする．次回アクセスチケット機構は，動的負荷制御機能によってアクセスを拒絶されたユーザに対して，次回アクセスが可能な時間を予測し，その時間が書かれた仮想的なチケットを発行する．そのチケットを持ったユーザが書かれた時間に再アクセスを行うと，それを許可する．

以下，本論文では，2 章で過負荷問題の対応策として従来用いられている負荷分散技術と関連研究について触れ，3 章で本論文で提案している NAP-Web の概要を述べる．4 章でアクセススケジューリング機構と次回アクセスチケット機構の概要と設計について述べ，5 章でこれらの機構をプロトタイプ実装した Apache モジュール mod_nap_web について述べる．6 章で mod_nap_web を導入したサーバの評価について述べる．7 章でまとめと今後の課題について述べる．

2. 研 究 背 景

NAP-Web の概要を説明する前に，過負荷対策として利用されている負荷分散技術の概要とその問題点について述べる．また Web サーバについて行われている関連研究についても述べる．

2.1 負荷分散技術

ユーザの増大による過負荷問題に対する最も有効な手段が負荷分散技術[1] である．負荷分散技術を用いた Web サービスでは，ユーザからのアクセスを複数台の計算機に振り分けることで，過負荷による応答時間の悪化やシステムダウンを防止する．DNS ラウンドロビンを用いた単純な手法，ロードバランサを用いたハードウェアによる負荷分散，クラスタリングソフトウェアによる負荷分散などの手法がある．複数台の計算機によりサービスを提供するため，その中の一部の計算機に障害が発生したとしても，他の計算機によりサービスを継続することが可能である．予想以上のアクセス集中により計算機資源が足りなくなったとしても，余剰の計算機を追加することが容易であるなどの特長を持つ．十分な性能を持つ計算機が非常に安価に手に入るようになった近年では，負荷分散技術を利用することが一見リーズナブルな手法であるように思われる．しかしながら，計算機を増やすことによる設備費，維持費，空調費，人件費などの管理運用コストは無視できず，サービス管理者はコストとアクセス量のバランス調整に頭を悩ませることになる．

Web サービスに必要な計算機資源を見積もることは困難である．たとえば，TV などのメディアで Web サービスが紹介されるなどの，外部のイベントによる突発的なアクセス集中を予測することは困難である．人気のあるコンテンツでユーザ数が予測以上に増加することにより，計算機資源の増強が追いつかない場合もある．逆に，アクセス量が予測を大きく下回る場合，計算機資源が無駄になってしまう．以上のように計算機資源の見積りは困難であり，サービス管理者は必要な予算の見積りにも悩むことになる．

このような問題に対して，外部の企業が提供する CDN（Contents Delivery Network）サービス[2] を利用する手法がある．CDN は大規模なコンテンツ配信用のミラーサーバネットワークであり，サービス管理者は需要を超えたアクセスを CDN 側にまわすことで，自社の Web システムが過負荷になることを回避できる．しかしながら，ミラーサーバであるために，更新が頻繁でないコンテンツ（HTML，PDF，JPG，MPG，ストリーミングなど）の配信には向いているが，更新が頻繁に行われるコンテンツ（主にプログラムにより動的に生成されるもの）の配信に関してはデータ整合性の面で向いていない．NAP-Web では，ソフトウェア的に過負荷問題への対応を行うため，多数のユーザに対して，高速な同時処理を行うことはできないが，CDN のような外部ネットワークを必要としない．また，1 つのサーバで運用できるため，更新が頻繁に行われるコンテンツでもデータ整合性の問題は発生しない．

2.2 関 連 研 究

過負荷問題を回避するために，負荷分散技術のほかにも様々な研究が行われている．

Web サーバの最大同時接続数や，持続的接続時間などのパラメータを動的にチューニングすることによって，Web 処理を効率化したり，性能保証を行ったりする手法がある．杉木ら[3),4)] は Autonomic Computing プロジェクトにおいて，サービス需要の変動に応じてサーバ自身がチューニングを行うシステムの開発を行っている．最大同時接続数や持続的接続時間の動的変更が提案されている．Menasce ら[5] は，QoS 値を応答時間，平均スループット，拒絶確率の増減量などから決定し，QoS 値を最良とする最大同時接続数の探索を行う手法を提案している．本研究でも初回アクセスに対する性能保証のために，最大同時接続数を動的に決定しているが，その枠に入らなかったアクセスに対して，次回アクセスを保証する点が，上記の研究とは異なっている．

また，Web 処理に特化したスケジューラを提案している研究もある．松沼ら[6] が提案す

る Session-aware Queue Scheduling は，Web アプリケーションにおいて，セッションレベルでの処理最適化を行うために，同時処理ページ数，同時処理セッション数を動的に決定するスケジューラである．複数のページが遷移して，1 つの目的を達成するセッション処理を効率化したり，セッション保護を行ったりすることを提案している．Tomcat 上の Java サーブレットコンテナとして実装されており，アプリケーション層において Web 処理をスケジューリングしている点が本研究と類似している．スラナワラッら[7] は，Web サーバのプロセスの特徴的な振舞いを検知し，そのプロセスを OS スケジューラの Ready キューに優先的に配置することで Web サービスを高速化する手法を提案している．OS のスケジューラそのものに手を入れている点が本研究とは異なる．上記のスケジューラに関する 2 つの研究は Web 処理の効率化によりサーバ性能を向上させることで，より多くのリクエストを処理することを目指している．本研究でも，アプリケーション層で Web 処理をスケジューリングするアクセススケジューラを提案しているが，サーバ性能の向上ではなく，次回アクセスを保証することを目的としている点がこれらの研究と異なる．

Web サーバの過負荷問題に対して，従来のクライアントサーバモデルでなく，P2P の利用を提案している研究もある．池嶋ら[8] は，Asagumo Web と呼ぶサービスインフラを提案している．P2P ネットワーク上で該当ファイルを一意に検出するための仕組みや，データ形式などについて提案を行っている．P2P を利用するため，1 度流出したデータの削除が困難である，データの整合性がとりにくい，などの P2P 特有の諸問題を有しているのに対して，本研究では従来の Web サーバの仕組みを利用しているため，そのような問題は発生しない．

3. NAP-Web の概要

2 章で述べたように，アクセス集中による Web サーバの過負荷対策には負荷分散技術が利用されているが，多くの計算機資源を必要とし，その見積りが困難である．我々は，計算機資源を十分に準備できない状況下でも，ユーザの不満をできる限り抑えながらサービスを継続することを目的とした，次回アクセスを保証する Web システム「NAP-Web」を提案する．NAP-Web が持つ機能は大きく分けて以下の 3 つである．

- 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

動的負荷制御機能は，システムの状態に合わせて適切な同時処理アクセス数を動的に決定し，その数に基づいてアクセスの許可と拒絶を行うことで，システムが過負荷状態になることを防ぐ機能である．この機能により，アクセスが許可されたクライアントに対して一定の応答時間でサービスを行うことを保証する．この機能を実現するためには，どのような状態を過負荷状態と判定するか，また過負荷状態に陥らない適切な同時処理アクセス数をどのように決定するか，といった問題を解決しなければならない．

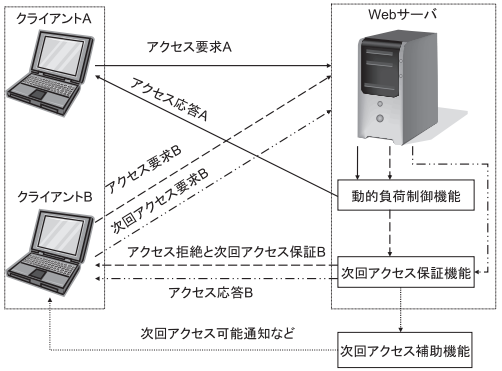

図 2　NAP-Web の機能
Fig. 2　Function of NAP-Web.

次回アクセス保証機能は，動的負荷制御機能によりアクセスを拒絶されたクライアントに対して，次回いつになればサーバが過負荷状態より脱し，アクセスが可能になるかを教え，かつそのアクセスに対するサービスを保証することで，ユーザの不満を緩和する機能である．サーバがいつになれば過負荷状態より脱するかを予測しユーザに伝え，その時間に再アクセスを行ったクライアントに対してアクセスを優先的に許可する機能が必要となる．

次回アクセス補助機能は，次回アクセス保証機能を補助する機能である．再アクセスが可能となる時間を完全に予測するのは困難であるため，予測が失敗した場合の補助を本機能で行う．次回アクセス保証機能により伝えられた時間帯よりも早くサーバが負荷状態より脱した場合にクライアントにそのことを伝え，そのクライアントからのアクセスは優先的に許可する．次回アクセス保証機能により伝えられた時間帯に自動的にサーバに対してクライアントが接続を行う機能をクライアントに持たせることもこの機能に含まれる．これらの機能を実現するためには，クライアント・サーバ間での連携が必要となる．

図 2 にこれらの機能の概要を示す．クライアント A，B からのアクセス要求 A，B は，

サーバの動的負荷制御機能によりアクセスを許可もしくは拒絶される．許可されたアクセス要求 A に関してはアクセス応答 A が返る．拒絶されたアクセス要求 B は次回アクセス保証機能により次回アクセス保証 B が与えられる．クライアント B がサーバに次回アクセスを保証された時間帯に次回アクセス要求 B を行うと，アクセス応答 B が返ってくる．次回アクセス補助機能により，サーバによって次回アクセスを保証された時間帯よりも早くサーバが過負荷より脱した場合は，次回アクセス可能通知が行われる．

NAP-Web は Web サーバデーモン「Apache」[9] で動作する Apache モジュールとして開発を行っている．すべての機能を Apache モジュールとして実装することにより，モジュールの追加を行うだけで当システムが利用できる．

NAP-Web の開発にあたっては，過負荷状態を定義する必要がある．本研究では，近年のネットワークの高速化を考慮し，過負荷状態をサーバ単体の能力不足により応答時間が悪化した状態と定義し，ネットワークの帯域不足や遅延による応答時間の悪化については考慮しない．

4. 概要と設計

本章では，NAP-Web の機能のうち，「動的負荷制御機能」と「次回アクセス保証機能」を実現するアクセススケジューリング機構と次回アクセスチケット機構について述べる．

4.1 概　　要

4.1.1 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，3 章で述べた NAP-Web の機能のうち，「動的負荷制御機能」を実現するものである．この機構は，クライアントからのアクセスを，以下に示す 4 つのアクセスグループに分けスケジューリングする．アクセスグループの関係を**図 3** に示す．

Run_Ready：OS によって通常どおりスケジューリングされ，処理が行われ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値は，サーバが実際に同時処理を行うアクセスの数（最大同時処理数）となる．この数が増加すると応答時間のばらつきが激しくなることが実験により分かっている[10]．応答時間のばらつきが激しくなると応答時間の予測が困難となるため，ばらつきが発生しない数に抑える必要がある．この数は，扱うコンテンツの種類やサーバの能力などの影響を受けるため，環境に応じて，設定しなければならない．

Wait：Run_Ready からあふれたアクセスが入るグループである．このグループ内のアクセスは到着順にキューイングされ，Run_Ready に空きができ次第，先頭のアクセスから

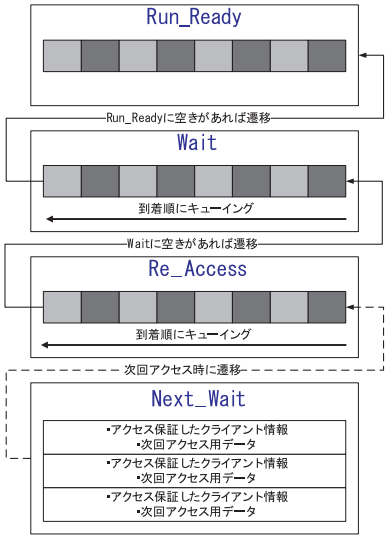

図 3　アクセスグループ
Fig. 3　Access group.

Run_Ready に移動する．通常の Web サーバでは最大同時処理数からあふれたアクセスはサーバプロセスに渡されず，TCP コネクションキューで待たされることになるが，そのようなアクセスもサーバプロセスに渡してしまうことで，クライアントからのアクセスが到着してから，サーバが応答を返すまでの時間をサーバ側で知ることができる．また，到着順にキューイングされることにより，それぞれのアクセスの待ち時間のばらつきが吸収される．このようにして得た応答時間の過去の統計により，応答時間の予測をより正確に行うことができる．

Next_Wait：Wait からもあふれたアクセスが入るグループである．このグループに入ったアクセスはいったん処理を拒絶される．このとき，アクセスを行ったクライアントに対しては，後述する次回アクセスチケット機構によって，次回いつになればアクセスが可能であるかが書かれた仮想的なチケットが発行される．チケット発行の履歴はサーバに残される．

Re_Access：Next_Wait に割り振られたアクセスを行ったクライアントが，次回アクセスチケット機構によって，チケットに書かれた時間帯に再アクセスを行った場合に割り振られる．到着順にキューイングされ，Wait に空きができ次第，先頭のアクセスから Wait に移

動する．なお，再アクセスを行った際に，Run_Ready もしくは Wait に空きがあった場合は Re_Access には割り振られず，直接 Run_Ready もしくは Wait に割り振られる．なお，応答予測の失敗などにより，サーバ管理者によって指定された時間内に処理が完了できないほどの再アクセスが集中すると，Re_Access に割り振ることのできないアクセスが現れるが，その場合は，再び Next_Wait にアクセスを割り振る処理を行う．当然，再々アクセスが必要な場合は，ユーザの不満が大きくなることが予想されるため，再アクセスが集中しないような仕組みが必要である．

なお，Wait の最大値と Re_Access の最大値の和は，サーバ管理者によって指定された時間内に処理を完了できるだけのアクセス数に制限される．このアクセス数は，負荷状況に応じて，動的に決定される．この制限を行うことによって，Run_Ready，Wait，Re_Access に割り振られたアクセスが一定時間内に処理を完了できるようにする．

また，現在の処理状況に応じて，Wait と Re_Access の割合を変更することで，次回アクセスの処理の優先度を変更できる．

4.1.2　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．これにより，NAP-Web における「次回アクセス保証機能」を実現する．

本機構は，アクセススケジューリング機構によって，Next_Wait に割り振られたアクセスに対して，仮想的なチケットを配布する．このチケットには，アクセスを識別するユニークなチケット ID と，次回アクセスが可能な時間帯（開始時刻と終了時刻）が記述される．この機能を実現するために，次回アクセスが可能な時間帯の導出が必要となる．次回アクセスが可能な時間帯はユーザが再アクセスを忘れずに行うと期待できる範囲の時間であるべきだが，この範囲が大きすぎると，再アクセスを処理するための計算機資源を長い間待機させなければならないため，ある程度の大きさで制限する必要がある．

また，上述のチケットを持ったアクセスが来た際，そのチケットが有効であるかを判定する必要がある．このため，チケットを発行した履歴をサーバ側が記憶しておかなければならない．

4.2　設　　計

本節では，上述の 2 つの機構の設計について述べる．

4.2.1　アクセススケジューリング機構の設計

Apache の典型的な処理サイクル[11] を**図 4** に示す．アクセススケジューリング機構は，

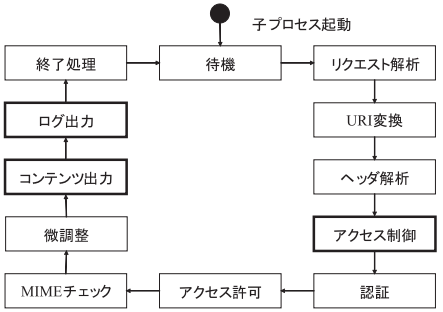

図 4　Apache の処理サイクル
Fig. 4　Process cycle of Apache.

図 4 のアクセス制御ステップと，ログ出力ステップで動作する．

クライアントから新しいアクセスがあると，まずはアクセス制御ステップまで Apache の通常処理が行われる．アクセス制御ステップで，アクセススケジューリング機構の処理が開始され，以下のように動作する．

(1)　処理済みアクセスの統計から，指定時間内に処理が完了できるだけの Wait と Re_Access の上限値を動的に決定する．

(2)　Run_Ready の現在の数をチェックし，上限値以下であればアクセスを Run_Ready に振り分け，通常の Apache の処理手順でログ出力ステップまで処理する．

(3)　Run_Ready が一杯であれば，Wait の数をチェックする．Wait が一杯でなければ，Wait のキューの最後尾にアクセスを入れる．

(4)　Wait が一杯のときに，次回アクセス用のチケットを持っていれば，次回アクセスチケット機構によりその内容がチェックされる．チェックされた結果，チケットが有効なものであれば，Re_Access のキューに割り振られる．チケットが有効でなければ，その旨を伝えるページが渡された後，アクセスが拒絶される．

(5)　Wait が一杯であり，チケットを持っていなければ，アクセスを Next_Wait に入れる．Next_Wait に割り振られたアクセスは，後述する次回アクセスチケット機構によって，次回アクセスのためのチケットが発行される．

Run_Ready のアクセスの処理は，ログ処理ステップまで進むと，Wait の数を見る．Wait が空でなければ，Wait の先頭のアクセスを Run_Ready に移動する．この際，Wait に空きができるので，Re_Access が空でなければ，Re_Access の先頭アクセスを Wait に移動さ

せる．

4.2.2　次回アクセスチケット機構の設計

次回アクセスチケット機構は，図 4 のアクセス制御ステップとコンテンツ出力ステップで動作する．アクセススケジューリング機構によって，Next_Wait に割り振られたアクセスは，次回アクセスチケット機構が動作するコンテンツ出力ステップまで処理が移動する．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の情報から，チケットを発行する．チケットには，チケット ID，次回アクセスが可能となる開始時刻，およびチケットの終了時刻が書かれる．このチケット ID と本来のリクエスト内容，チケット開始時刻，チケット有効期間，チケット終了時刻は，サーバ側に記録される．チケット ID を発行した後，このアクセスが要求していたリクエストは拒絶される．ユーザは，本来要求していたリクエスト応答の代わりに，サーバが過負荷状態であることと，次回アクセスのためのチケットが発行されたことを知らされる．Web サーバの処理は，この後，アクセススケジューリング機構のログ処理ステップに移動する．

なお，本機構では，チケットの発行管理と，再アクセス間隔の分散のために，スロットという概念を導入している．スロットとは，次回アクセスが行われる時間をチケットの有効期間ごとに区切り，その期間で配布できるチケットの枚数を管理するものである．各スロットは，そのスロットの開始時刻と期間，その期間で発行できるチケットの最大数，すでに発行したチケットの数を持つ．最初のスロットの開始時刻は，Wait 内の最後のアクセスの完了予測時刻となる．それ以降のスロットの開始時刻は，直前のスロットの終了時間となる．スロットで発行できるチケットの最大数は，そのスロットの期間内に十分な余裕を持って処理を行えるであろうアクセスの数である．発行したチケットの数が発行できる最大数と等しくなったときは，次のスロットに順番を移す．チケットが新しく発行されるとき，スロットの開始時刻と現在までに発行された枚数，チケットの有効期間をもとに，チケットの開始時刻を決定する．このようにして，チケットの開始時刻を分散させることで，再アクセスが集中しないようにしている．

上記のコンテンツ出力ステップで渡されたチケットを持って，次回アクセスが行われると，アクセス制御ステップで，次回アクセスチケット機構が動作し，記録していたチケット ID とアクセスが持っているチケットの ID の照合を行う．チケットが正規のものであり，かつ有効期間と判定されると，アクセスはアクセススケジューリング機構によって Re_Access に割り振られる．

5. 実　　装

アクセススケジューリング機構と次回アクセスチケット機構のプロトタイプを Apache モジュール mod_nap_web として実装した．

5.1　アクセススケジューリング機構の実装

Wait 上限値の決定：本プロトタイプでは，Wait の上限値を，回帰直線を用いて，動的に決定している．最小二乗法を用いて，直近の処理済アクセスが Wait に割り振られた際の Wait キューのサイズと，応答時間の組をパラメータとして回帰直線を求める．アクセスが完了すべき最悪の時間として設定した時間とこの回帰直線が交わるときのキューサイズが，Wait の上限値となる．

アクセススケジューリングの実現：Wait から Run_Ready への遷移には，シグナルを用いている．Wait と Re_Access に割り振られたアクセスは，シグナル待機状態にしている．Wait の先頭アクセスが，処理が完了した Run_Ready のアクセスからシグナルを受信すると，処理を再開する．このようにして，Wait から Run_Ready への遷移を行っている．Re_Access から Wait への遷移は単純に Re_Access のキューから Wait のキューへの移動で実現される．

次回アクセスチケット機構への遷移：Next_Wait から次回アクセスチケット機構へ処理を移動する際には，Apache に用意されているエラーページへの自動遷移機能を利用している．具体的には，Next_Wait のアクセスに対しては HTTP エラーの 1 つである Service Unavailable(503) を発行する．すると，Apache はこのエラーに対応する URL へアクセスを内部リダイレクトさせる．リダイレクト先の URL で次回アクセスチケット機構が動き，チケットの配布が行われる．

プロトタイプの制限事項：負荷状況による Wait と Re_Access の割合変更は実装しておらず，Wait の上限値の動的変更のみ実現している．Re_Access の上限値は，静的に決定している．

5.2　次回アクセスチケット機構の実装

チケット ID の決定法：本プロトタイプでは，チケットの ID を，アクセスの到着時間と送信元 IP アドレスをもとに，MD5 ハッシュを用いて生成し，クライアントに HTTP1.1 で規定される cookie として渡す．この cookie の有効期間は，チケットの有効期間と同一である．

スロット発行チケット枚数の決定：各スロットで発行できるチケットの最大数は，スロッ

トの有効期間を直近の処理済みアクセスの平均応答時間で割った数としている．スロットの有効期間は静的に決定している．

6. 評　　価

アクセススケジューリング機構と次回アクセスチケット機構により，次回アクセス保証が可能となったかどうか，そして次回アクセスを行うことにより，サーバにアクセスが集中しても，サービス応答時間を一定時間内に収めることができるかどうかを確認するために，2 つの機構をプロトタイプ実装した Apache モジュール mod_nap_web を Apache 上に適用して，評価実験を行った．Web サーバとして使用する計算機は，Xeon2.4 GHz × 2，メモリ 1 GB を積んだ PC である．OS として Linux 2.6.23.15，Web サーバデーモンとして Apache2.2.8 を用いている．

mod_nap_web を適用する前と，適用した後の Web サーバに対して，負荷実験を行った．Web サーバ上に置かれた掲示板プログラム ASKA BBS [12] に 1 KB の文字列データを投稿する処理を 50 回繰り返すプロセスを 1 台あたり 200 個同時に実行する PC を 3 台用いて，サーバに負荷をかける．したがって，本実験は 600 台のクライアントがいっせいに 1 台の Web サーバに対してアクセスを行うことをシミュレートしていることになる．mod_nap_web 適用後の Web サーバから，アクセスが拒絶され，チケットが配布された場合は，各プロセスはチケットに書かれた時刻まで sleep してから次回アクセスを行う．mod_nap_web の設定として，Run_Ready の最大値は 10，Wait で待たされる最大の秒数を 4 秒，統計に用いる処理済みアクセス数を 100 としている．Wait の最大値は Wait で待たされる最大の秒数が 4 秒になるよう動的に決定される．アクセスチケット発行機構におけるスロットの期間は 5 分間となっている．クライアントは次回アクセスが必要な場合，サーバによって渡されたチケットの開始時刻に次回アクセスを行っている．チケットの有効期間は 5 分間としているが，今回の実験の目的は，次回アクセス保証が可能かどうかと，応答時間を一定に保てるかどうかを確認することであり，チケットの有効期間に関しての評価は行っていない．

mod_nap_web 適用前の実験結果のグラフを**図 5**，適用後の実験結果のグラフを**図 6** に示す．両グラフとも，横軸が経過時間を表し，縦軸が応答時間を表す．また，適用前と適用後の応答時間の統計をとったものを**表 1** に示す．

Web サーバに対してアクセスが集中すると，図 5 のように 1 秒以下の応答時間で済むアクセスとそれ以外の応答時間が極端に延びるアクセスに分かれる．応答時間が極端に延びるアクセスは，計算機資源が不足していたため，処理が後回しになってしまったアクセスであ

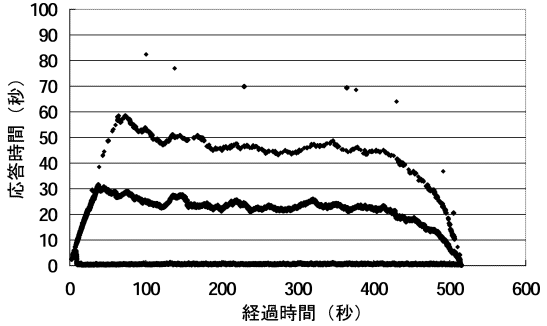

図 5　mod_nap_web 適用前
Fig. 5　Before applying of mod_nap_web.

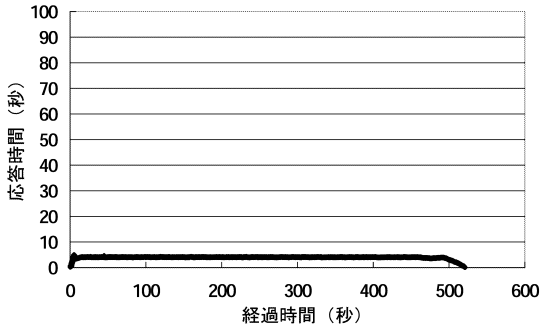

図 6　mod_nap_web 適用後
Fig. 6　After applying of mod_nap_web.

表 1　応答時間の統計値（単位：秒）
Table 1　Statistics for response time.

	適用前	適用後	適用後（含待機時間）
平均値	9.31	3.92	8.88
分散	11.61	0.51	16.91
最大値	82.40	4.98	76.17

表 2　待機時間の統計値（単位：秒）
Table 2　Statistics for wait time.

平均値	58.7
分散	14.7
最大値	72

る．サーバが過負荷状態になると，このようなアクセスが増加し，十分な応答速度でサービスを行えなくなる．これに対して，mod_nap_web を適用すると，図 6 に示すように，そのような極端に応答時間が延びるアクセスが現れなくなり，かつ許可されたアクセスに対しては平均 3.92 秒，最悪でも 4.98 秒の時間でサービスが行えるようになった．

また，次回アクセスが発生した場合の待機時間を含めてサービスを受けるのに必要な時間という観点から比較した場合でも，表 1 に示すように，適用前の平均 9.31 秒，最大 82.4 秒に対して，適用後は平均 8.88 秒，最大 76.17 秒と若干の改善が見られる．

ユーザ側からの見え方を検証するために，アクセスを拒絶された際，クライアントが次回アクセスのために待つ時間を調べた．その結果を**表 2** に示す．表 2 から，アクセスを拒絶されたとしても平均 58.7 秒，最悪 72 秒待てば，次回アクセスが可能であることが分かる．

なお，本実験では次回アクセスが必要となったのは全 30,000 アクセス中 2,532 アクセスであり，それらはすべて成功した．

7. まとめと今後の課題

次回アクセスを保証する Web システム NAP-Web の概要と，NAP-Web のプロトタイプ実装である mod_nap_web について述べた．mod_nap_web はアクセススケジューリング機構と次回アクセスチケットにより構成されている．mod_nap_web を適用することによって，Web サーバに対してアクセスが集中した際，過負荷状態を避けながら，次回アクセス保証を行うことが可能である．ただし，実際に Web サーバ上に本機構を導入して運用するには，解決すべき問題点が残っている．まず，本システムを実現するために必要としている次回アクセス補助機能を実現しなければならない．この実現のためには，クライアントに機能を組み込む手法と，クライアント・サーバ連携のための通信方法について検討する必要がある．クライアントに機能を組み込む手法に関しては，ブラウザのみで完結させる場合と，プラグインを追加する場合を検討している．また，クライアント・サーバ連携のための通信方法として，server-push 技術を用いることを検討している．server-push 技術として ajax-polling や Comet，piggyback などがある．これらの手法について，今後検討を行っていく．その他の課題として，セッション通信への対応，応答時間が長くなる大容量ファイルのダウンロードへの対応，負荷が極端に異なるアクセスが混在した場合の対応，適切なチケット有効期間の決定，チケット時間を無視して連続アクセスを行うユーザ，もしくはチケットが配布されても次回アクセスを行わないユーザへの対応，十分な処理性能を発揮するためのチューニング，などがあげられる．今後，セッション通信の優先処理，チケット発行機構での次回アクセス率の適用などを実装し，実運用を可能にしていく．

参 考 文 献

1) Tony, B.（著）鍋島公章，上谷　一，横山晴庸（訳）：サーバ負荷分散技術，オライリージャパン (2001).
2) 田代秀一，西角直樹，西木健哉ほか：インターネットコンテンツ配送技術の最新動向，情報処理，Vol.42, No.11, pp.1082–1091 (2001).
3) 杉木章義，河野健二：応答時間の分散を利用したウェブサーバ接続数の自動設定，情報処理学会研究報告，2006-OS-102, pp.37–44 (2006).
4) 杉木章義，河野健二：性能パラメータの自動調整による Web サーバの性能向上，情報処理学会研究報告，2005-OS-99, pp.29–36 (2005).
5) Menasce, D.A. and Bannai, M.N.: On the Use of Performance Models to Design Self-Managing Computer Systems, *Proc. 2003 Computer Measurement Group Conf.* (2003).
6) 松沼正浩，光来健一，日比野秀章ほか：過負荷時の Web アプリケーションの性能を改善する Session-aware Queue Scheduling，日本ソフトウェア科学会論文誌「コンピュータソフトウェア」，Vol.23, No.2, pp.199–210 (2006).
7) スラナワラッ・スカンヤ，谷口秀夫：WWW サーバにおけるサービスの処理内容を考慮したプロセススケジュール法，情報処理学会論文誌，Vol.40, No.6, pp.2510–2522 (1999).
8) 池嶋　俊，阿部洋丈，加藤和彦：Asagumo Web：P2P 技術を用いた Web システム，日本ソフトウェア科学会第 22 回大会 (2005).
9) Apache Software Foundation. http://www.apache.org/
10) 加地智彦，最所圭三：Web 処理を効率化するアクセススケジューリング機構について，情報処理学会研究報告，2007-OS-104, pp.75–80 (2007).
11) 小山浩之：Web エンジニアのための Apache モジュールプログラミングガイド，技術評論社 (2003).
12) KENT WEB. http://www.kent-web.com/

(平成 20 年 3 月 9 日受付)
(平成 20 年 11 月 5 日採録)

加地　智彦（学生会員）

1982 年生．2005 年香川大学工学部信頼性情報システム工学科卒業．2007 年同大学大学院工学研究科信頼性情報システム工学専攻博士前期課程修了．現在，同大学院工学研究科博士後期課程在学．システムソフトウェア，インターネット応用システムに興味を持つ．

最所　圭三（正会員）

1959 年生．1982 年九州大学工学部情報工学科卒業．1984 年同大学大学院工学研究科修士課程修了．同年同大学工学部助手．1991 年同大学工学部講師．1993 年同大学大型計算機センター助教授．1994 年奈良先端科学技術大学院大学情報科学研究科助教授．2000 年香川大学工学部教授，現在に至る．工学博士．システムソフトウェア，インターネット応用，並列/分散処理，高信頼性システム等の研究に従事．1998 年情報処理学会全国大会大会優秀賞受賞．IEEE Computer Society，電子情報通信学会，日本ソフトウェア科学会各会員．