通知によるコンテンツ一斉公開機構を用いたWWWクラスタシステム

西馬一郎[†,☆] 河合栄治[††] 知念賢一[†]
山口英[†] 山本平一[†]

WWWクラスタではすべてのサーバが同一のコンテンツを保持し，均一なサービスをしなければならない．しかし，サーバのコンテンツを新しく置き換える更新時，コンテンツを格納すると即時に公開され，各サーバで処理時間に差が存在するためクラスタ内にコンテンツが新旧混在する．そこで，本稿ではコンテンツ更新時におけるクラスタでのコンテンツの不一致を解消し，コンテンツの同一性を保持する機構を提案する．提案方式では公開に至るまでの処理を格納と公開処理に分離し，すべてのサーバが格納した後に一斉に公開する．この格納と公開処理は通知を中心に構成する．処理を分離することにより，公開に至るまでの処理の時間差を吸収することができる．このため，提案方式はサーバを集中配置したクラスタから転送に時間を要するような分散配置したクラスタまで適用できる．

The Design and Implementation of WWW Publishing Cluster with Notification-based Synchronization

Ichiro Nishiuma,[†,☆] Eiji Kawai,[††] Ken-ichi Chinen,[†]
Suguru Yamaguchi[†] and Heiichi Yamamoto[†]

A WWW cluster that consists of multiple servers should act as a single system and provide consistent information to clients. However, contents update on such a cluster often results in contents inconsistency. Because uploaded contents are immediately available to clients after their uploads, there exists a period when some servers have new contents and the others old. We propose a WWW publishing system that achieves strict synchronization of contents on a cluster. Our system divides the contents publishing process into two phases. On the first stage, an update event is notified each server and the servers retrieve the contents. On the second stage, the publishing time of the contents is notified and each server makes the new contents available to clients at the publishing time with synchronization. Since our system is highly scalable, it is applicable from a single cluster to geographically distributed clusters.

1. はじめに

WWW（World Wide Web）が社会基盤となり，様々なサービスが展開されるようになった．大規模なWWWサイトでは，複数のサーバにより構成するWWWクラスタが採用される．クラスタを構成することにより，サーバ1台あたりの負荷を軽減し，全体の処理能力の向上が図られる．クラスタは1つのシステムとして動作を保証し，均一なサービスをしなければならない．このため，すべてのサーバが同一のコンテンツを保持する必要がある．オークションサイトやチケットサイトなどではこの同一性が重要である．クライアントは正確な情報を得る機会があるにもかかわらず，機会を逸することがあるからである．このような現象は，クライアント側にとって，情報取得の機会に関する不平等と考える．大規模なWWWサイトではこのようなコンテンツの同一性の乱れによる不確実な動作は許されない．

コンテンツの同一性はクラスタ内のコンテンツを更新するときに損なわれる可能性が高い．コンテンツの更新とは，コンテンツ作成者が既存のコンテンツに変更を加えることである．更新したコンテンツをクライアントに公開できる状態にするため，すべてのサーバ

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†† 科学技術振興事業団さきがけ21
PRESTO, Japan Science and Technology Corporation
☆ 現在，日本経済新聞社情報技術本部
Presently with Information Technologies Bureau, Nihon Keizai Shimbun

に転送，格納処理し，更新を反映する．サーバではコンテンツを格納処理すると，コンテンツは即時に公開される．サーバで処理時間に差が生じ，すべてのサーバの格納処理が同時に終了しないと，クラスタ全体で公開するコンテンツが不一致となる．したがって，更新前の古いコンテンツと更新後の新しいコンテンツがクラスタ内に混在する時間が生じ，アクセスしたクライアントは新旧両方のコンテンツを取得する可能性がある．

そこで，本稿ではコンテンツが混在することを防ぐため，コンテンツを一斉に公開することにより，コンテンツの同一性を保持する機構を提案する．これをNPS（Notification-based Publish Synchronization）と呼ぶ．NPSでは公開に至るまでの処理過程を格納と公開に分離する．サーバは通知を受けコンテンツを格納する．このときコンテンツをただちに公開しない．次にサーバは公開時刻の通知を受け，公開時刻になるまで待機し，公開時刻に一斉公開する．

WWWサービスの一般化にともない，DNS（Domain Name System）やHTTP（Hypertext Transfer Protocol）リダイレクトを利用した機構，レイヤ4スイッチなど，各層にわたって負荷分散機構が開発されており，様々な用途に合わせて用いられる．コンテンツ配信時にはこれらの負荷分散機構との組合せを考慮しなければならない．最近普及しつつあるCDN（Content Distribution Network）では，負荷分散機構と協調してコンテンツの更新を制御するため，特別なプロトコルや名前空間を使用している．よって，特定の負荷分散機構に合わせてコンテンツの更新を制御しなければならず，組み合わせる負荷分散機構の選択の自由度は低い．一方NPSでは，負荷分散機構を操作せずに，コンテンツ供給側のみでコンテンツの更新を制御する．したがって，負荷分散機構の種類によらず，自由度は高い．

本稿では，NPSがコンテンツの同一性を保持できることを示す．疑似環境をLAN（Local Area Network）内に構築し，コンテンツ供給側で用いられる他の方式とNPSを比較し評価する．また，コンテンツの更新や管理にともなって発生する手間や帯域などを検討し，NPSの優位性を明らかにする．

2. コンテンツの不一致が起こる原因

本章では，クラスタ内のコンテンツが不一致となる原因を考察し，クラスタにおいてサーバがコンテンツを格納する既存の方式を比較する．

コンテンツ作成者が行ったコンテンツの更新は，クライアントへ公開するホスト（以下，公開ホスト）に反映される．公開ホストにコンテンツを格納すると，コンテンツは即時に公開される．クラスタでは，複数の公開ホストでこの格納処理を行う．公開ホスト間でこの処理時間に差が生じるため，公開のタイミングがずれる．処理時間の差を解消することは難しい．これは，各公開ホストの負荷やネットワークの遅延などが一様でないためである．このように，クラスタ内でコンテンツが不一致となる原因は2つ存在し，**即時公開**と**処理時間の差**である．

表1 コンテンツ格納方式における利点，欠点
Table 1 The pros and cons of the methods of storing contents.

	ローカルディスク	リモートディスク	キャッシュ
コンテンツの管理	×複雑	○容易	○容易
コンテンツ作成者の手間	×煩雑	○容易	○容易
供給側ネットワークの帯域	○狭	×広	○狭
クライアントへの応答時間	○短	×長	△中

次に，公開ホストがコンテンツを格納する方式を分類する．ローカルディスクやリモートディスクにコンテンツを格納する方式，またこれらの折衷案としてリモートディスクに格納したコンテンツをキャッシングする方式がある．**表1**にこれらの方式をクラスタに導入した際の比較を行った．以下にそれぞれの方式について述べる．

2.1 ローカルディスクの場合

公開ホスト上のWWWサーバは，ローカルディスクに格納したコンテンツを読み込み，クライアントへコンテンツを提供する．ローカルディスクは高速であるため，WWWサーバは高速にコンテンツを提供できる．しかし管理者は，クラスタ内のすべての公開ホストのコンテンツを管理しなければならない．

コンテンツ作成者がクラスタ内の全公開ホストのファイルシステムに対してコンテンツを転送すると，コンテンツの更新は即時に反映される（**図1**）．コンテンツ作成者はすべての公開ホストに転送するため，手間がかかり煩雑である．**図2**に公開ホスト間での転送，格納処理の時間差が原因でコンテンツの不一致が発生する様子を示す．これらの処理に要する時間差は，ネットワークの遅延や公開ホストの負荷が影響する．

2.2 リモートディスクの場合

公開ホストがリモートディスクに格納したコンテンツを利用してクライアントに応答する．**図3**のよう

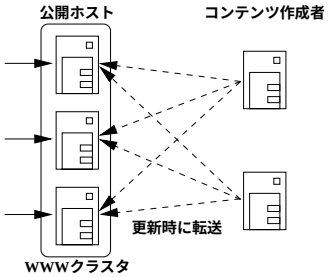

図 1 ローカルディスクに格納する場合のコンテンツの流れ
Fig. 1 The flow of contents stored on local disk.

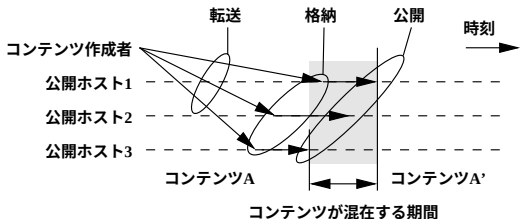

図 2 処理時間の差により発生するコンテンツの不一致
Fig. 2 Contents are mixed during uploading on servers.

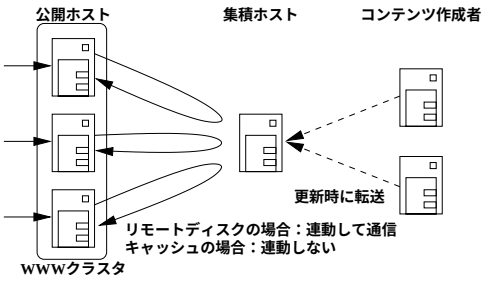

図 3 リモートディスクおよびキャッシュに格納する場合のコンテンツの流れ
Fig. 3 The flow of contents stored on remote disk or cache.

にクラスタとコンテンツ作成者の間に集積ホストを設け，クラスタは集積ホストのファイルシステムを共有する．コンテンツを 1 カ所で管理できるため，管理は容易である．コンテンツ作成者は集積ホストにのみ更新したコンテンツを転送すれば，即時に公開ホストに反映される．

この場合，公開ホストに到来するクライアントからの要求が連動して集積ホストに到達するため，公開ホストの台数分の要求が集中し，負荷や帯域が増大する面で欠点がある．また，公開ホスト・集積ホスト間の遅延時間が長くなると，クライアントへの応答に時間を要する．

2.3 キャッシュの場合

キャッシュ技術は頻繁に利用されるデータを直近の記憶領域に格納し再利用するものであり，ファイルシステムや WWW に採用されている．コンテンツを格納するファイルシステム上だけでなく，通信上のキャッシュも可能である．本稿の対象とする WWW でも，多くのキャッシュ技術が採用されている．

クラスタにキャッシュシステムを導入した場合，図 3 のように，コンテンツ作成者は集積ホストにのみ更新したコンテンツを転送すればよい．クライアントの要求に応じて，キャッシュシステムは集積ホストからコンテンツを取得，キャッシュに格納しクライアントに応答する．よって，クライアントに対する応答時間が短くなり，処理の集中や帯域を抑制できる．また，個々のキャッシュシステムは自律的に動作するため，コンテンツを管理する必要はない．管理者はコンテンツの複写や退避などの処理が省け，公開ホストの追加や撤去など公開ホストの台数の増減に対応しやすい．したがって，キャッシュシステムは WWW クラスタへの導入が容易である．

ただし，それぞれのキャッシュシステムは独立して動作するため，クラスタ内のコンテンツを一致させることはできない．さらに，集積ホストに反映されたコンテンツの更新を即時に反映することは難しい．このため，キャッシュシステムを制御する仕組みが必要となる．

3. コンテンツ一斉公開（NPS）の設計と実装

2 章で述べたコンテンツが不一致となる 2 つの要因のうち，処理時間の差を解消することは難しい．そこで，即時公開を回避することによって，新旧のコンテンツの混在を防ぐ．公開が処理時間の差に影響を受けないように，処理過程を格納と公開に分離する．公開ホストではコンテンツを即時に公開せず，すべての公開ホストがコンテンツを格納するまで待機し，公開時刻に従って公開する．すなわち，NPS では複数の公開ホストでコンテンツを一斉に公開することにより，コンテンツの同一性を保持する．

2.3 節に示したように，キャッシュシステムはクラスタへの導入が容易であるが，それぞれ自律的に動作するため，コンテンツの更新の制御が困難である．そこで NPS では，公開ホストのキャッシュシステムとそのコンテンツを制御するプログラムを集積ホストに設ける．これをコンテンツ制御サーバ（以下，制御サーバ）と呼ぶ．さらにキャッシュシステムに制御サーバの指示に従い公開時刻にコンテンツを公開する機能を設け，コンテンツの一斉公開を実現する．このとき，キャッシュシステムにリバースプロキシを用いる．リ

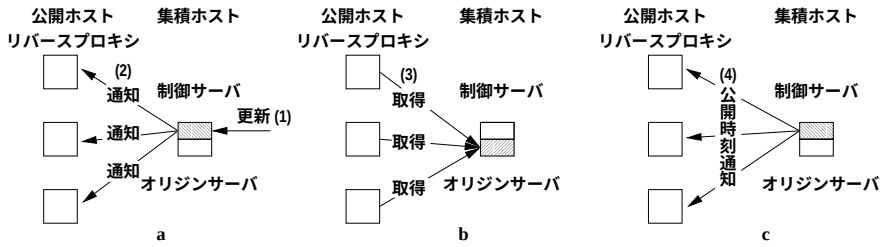

図 4 公開に至るまでのコンテンツの制御
Fig. 4 The control of updated contents.

バースプロキシとは，クライアントからの要求に対して WWW サーバの代わりに応答するシステムである．

NPS は，コンテンツの格納処理の時間差を吸収することができるため，WWW サイトにおけるサーバの構成や配置に依存しない．LAN 上に集中して公開ホストを配置したクラスタや，WAN（Wide Area Network）上にクラスタを分散して配置した分散クラスタに適用することができる．

3.1 コンテンツ制御サーバとリバースプロキシの通信

コンテンツ作成者から受理したコンテンツが公開ホストで公開されるまで，制御サーバが行うコンテンツの制御を**図 4** に示す．制御サーバは，図 4-a (1) にあるように，コンテンツ作成者から更新したコンテンツを受理する．図 4-a (2) にあるように，受理したコンテンツを取得するよう全リバースプロキシに要求を送る．リバースプロキシは要求を受けると，オリジンサーバからコンテンツを取得する（図 4-b）．取得して格納すると応答を制御サーバに返す．制御サーバは全リバースプロキシから応答があると，公開時刻を通知する（図 4-c）．公開時刻は，過去の公開時刻通知に要した時間の平均時間に，余裕を持たせるための一定時間を追加して算出する．以上の処理の流れを時間の経過とともに示すと，**図 5** となる．このようにして，制御サーバはコンテンツ作成者が更新したコンテンツを蓄積し，コンテンツの更新を制御する．

3.2 リバースプロキシの設計と実装

リバースプロキシは制御サーバに指定された公開時刻に従い，公開時刻にコンテンツを公開できる状態にする．すべてのリバースプロキシが公開時刻に公開するため，クラスタでコンテンツを一斉に公開できる．リバースプロキシとして動作する Chamomile[1]に，制御サーバに従いコンテンツを公開する機能を新たに実装した．

更新の通信に失敗する場合を想定し，リバースプロキシは一定期間保持したコンテンツについてオリジンサーバにリクエストを発行して，コンテンツが更新されたかどうか確認する．これにより，失敗によるコンテンツの欠落を回復できる．

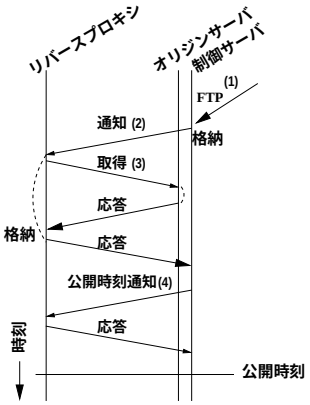

図 5 コンテンツの公開に至るまでの通信の流れ
Fig. 5 Message flow on contents publish.

3.3 コンテンツ制御サーバとリバースプロキシの通信プロトコル

制御サーバ Dnotify とリバースプロキシの通信とそれぞれが扱うプロトコルを**図 6** に示す．制御サーバはコンテンツ作成者からコンテンツを受理し，集積ホストのファイルシステムに格納する．このとき用いるプロトコルは FTP（File Transfer Protocol）[2]である．これはコンテンツ作成者は FTP クライアントプログラムを用いてファイルを転送することが多いからである．

リバースプロキシは，クライアントからの要求と制御サーバからの要求を扱う．これらはともに HTTP[3]であり，両者を識別するために制御サーバからの HTTP 要求を拡張し，X-Notify や X-Public ヘッダを新たに設けた．時刻の表現に関しては RFC1123[4]の形式を用いた．制御サーバが更新を通知する際の要求と公開時刻を通知する際の要求を以下に示す．

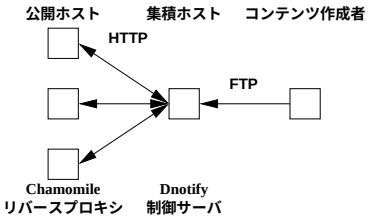

図 6 制御サーバのプロトコル
Fig. 6 The protocols used by the control server.

更新の通知

```
GET /index.html HTTP/1.0
X-Notify:
```

公開時刻の通知

```
GET /index.html HTTP/1.0
X-Public:  Mon, 19 Jul 2001 19:39:24 GMT
```

3.4 更新したコンテンツの扱い

制御サーバがコンテンツを受理してから公開するまで，コンテンツの転送と格納は，一時的なファイルを設けて行う．これは既存のプログラムとの整合性をとり，手動でのファイル操作を想定しているためである．NPS では 1 つのファイル名に対して中身の異なるファイルを同時に 2 つ持つことが必要であるが，一般的なファイルシステムでは実現できない．そこで制御サーバは，コンテンツを受理すると一時的なファイル名で保存する．公開時刻になると制御サーバとリバースプロキシはその一時的なファイル名を公開するファイル名に置き換える．

3.5 コンテンツの識別方法

NPS では，全リバースプロキシと制御サーバの間でコンテンツの中身の違いや一致を検出する必要がある．更新する前後のコンテンツは名前が同じであるため，その中身で識別しなければならない．多くの場合ファイルサイズや最終更新時刻などで識別できるが，本研究では厳密さを期するため，HTTP/1.1 に記載されている ETag（EntityTags）を用いた．NPS では，ETag はホストの OS や時刻に依存せずコンテンツの中身のみに依存させるためハッシュ関数を用いて決定される．

4. コンテンツ一斉公開（NPS）の評価

本章では，既存の代表的な更新を伝搬する方式と比較することで NPS を評価する．評価のための指標を設け，疑似環境を LAN 内に構築し実験を行った．LAN 上に集中的に公開ホストを配置したクラスタ環境，WAN 上に分散してクラスタを配置した分散ネットワーク環境を構築した．

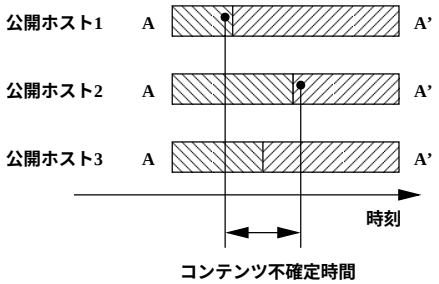

図 7 コンテンツ不確定時間
Fig. 7 Contents inconsistency time.

4.1 コンテンツ不確定時間

NPS を評価するための指標として，コンテンツ不確定時間を新たに定義する．コンテンツ不確定時間は，クラスタ内で新旧両方のコンテンツが混在する期間である．クラスタ内で最初にコンテンツの置換が確認されたホストの直前の時刻，そして最後に置換が確認されたホストの時刻の差をコンテンツ不確定時間とする．これを図で示すと，**図 7** のようになる．この場合，クラスタ内で最初にコンテンツが置換したのが公開ホスト 1，最後に置換したのが公開ホスト 2 である．クラスタ内のコンテンツの変化を調べるために，クライアントがそのコンテンツに対して一定のリクエスト間隔で要求を発行する．変化を検知するには最低 2 回の要求が必要であるから，コンテンツ不確定時間はリクエスト間隔以上の値となる．

4.2 実験 1：集中配置したクラスタにおける評価

NPS を代表的な更新の伝搬方式と比較し，評価する．比較の対象に，コンテンツを転送することにより更新の伝搬を行う rsync[5]，分散ファイルシステムである NFS（Network File System）と InterMezzo[6]，キャッシュシステムとして Squid[7] を用いた．ただし，NFS と Squid の場合，集積ホストのコンテンツを即時に反映させるため，自身のキャッシュ機能を無効にした．

4.2.1 実 験 方 法

実験環境は，**図 8** に示すように，3 台の公開ホストと集積ホスト，コンテンツ作成・転送のためのホストで構成される．コンテンツ作成者が集積ホストに対してコンテンツを転送する．各方式に沿って作成者によるコンテンツの更新は，公開ホストに反映される．クライアントが公開ホストに対してリクエストを発行し，公開ホスト上の WWW サーバのログファイルを調べ，コンテンツ不確定時間を算出する．**表 2** に各マシンの仕様を，**表 3** に各マシンで動作するアプリケーション

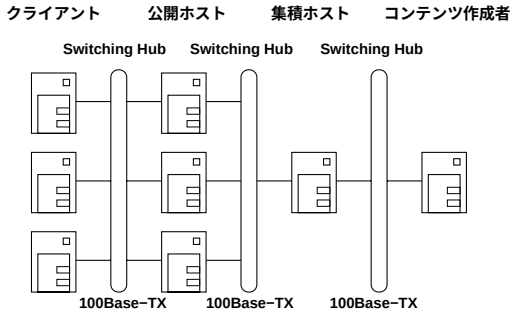

図 8 実験 1 の構成図
Fig. 8 The topology of testbed.

表 2 実験 1 に用いたマシンの仕様
Table 2 Machines used in the experiment.

	OS	CPU	Memory
クライアント	FreeBSD 4.3	PentiumIII 860 MHz	128MB
公開ホスト	Linux 2.4.4	PentiumII 450 MHz	256MB
集積ホスト	Linux 2.4.4	PentiumIII 860 MHz x2	512 MB

表 3 実験 1 に用いたアプリケーションの仕様
Table 3 Applications used in the experiment.

	公開ホスト	集積ホスト
rsync	Apache-1.3.20	
NFS	Apache-1.3.20	
InterMezzo	Apache-1.3.20, InterMezzo	InterMezzo
Squid	Squid-2.4	Apache-1.3.20
NPS	Chamomile-1.1	Dnotify, Apache-1.3.20

をまとめた．なお，集積ホストとクラスタ間で時刻は NTP（Network Time Protocol）[8]により同期する．

コンテンツ作成者はファイルサイズの異なる同一名のコンテンツを交互に集積ホストに転送することにより，コンテンツの更新を公開ホストに伝搬する．クライアントはこのコンテンツを取得する要求を公開ホストに対して発行する．クライアントプログラムには http_load[9]を用いる．このとき，1 秒間に発行するリクエストの件数（リクエストレート）を変化させる．

4.2.2 結　　果

コンテンツ不確定時間の累積度数分布を**図 9** に示した．このときのリクエストレートは秒間 300 件である．コンテンツ不確定時間を 80%tile 値で比較すると，rsync は 200 ms，NFS と InterMezzo は 100 ms，Squid は 17 ms，NPS は 6 ms である．NPS はコンテンツ不確定時間を大幅に低減する．Squid の場合，公開ホストに対するリクエストがすべて集積ホストに中継されるため，集積ホストに 3 台分のリクエストが集

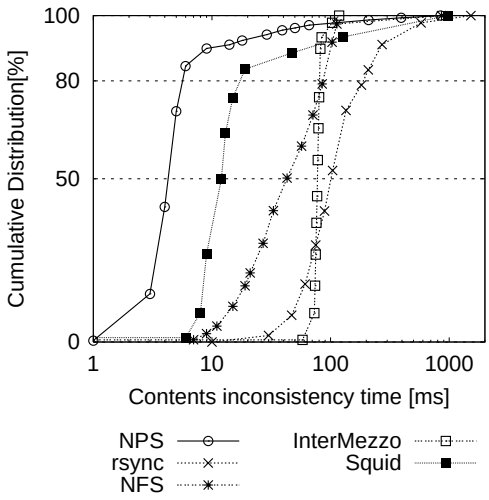

図 9 コンテンツ不確定時間の分布
Fig. 9 Cumulative distribution of contents inconsistency time.

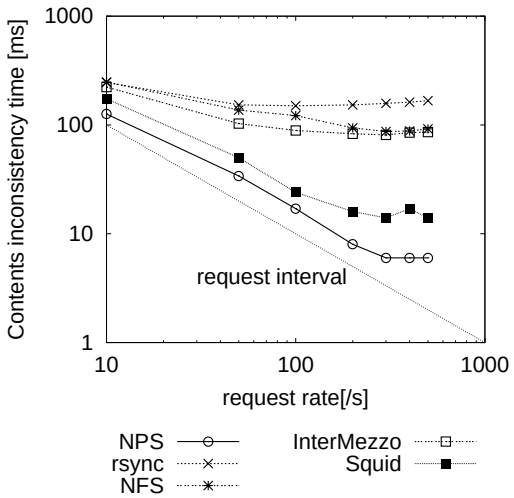

図 10 コンテンツ不確定時間の 80%tile 値とリクエストレートの関係
Fig. 10 Contents inconsistency time versus request rate.

中する．よって Squid の場合はコンテンツ不確定時間は短いが，集積ホストに与える負荷が大きくなる．NPS の場合も集積ホストからコンテンツを取得するが，更新を伝搬するときのみであるため，Squid に比べて集積ホストに与える負荷は無視できるほど小さい．

次に，クライアントの発行するリクエストレートを変化させた．リクエストレートが大きいほど検出できるコンテンツ不確定時間は短くなる．**図 10** にコンテンツ不確定時間の 80%tile 値とリクエストレートの関係を示す．点線部分はリクエストの時間間隔をプロットしたもので，検出できるコンテンツ不確定時間の最低値である．リクエストレートが小さいときは，5 方

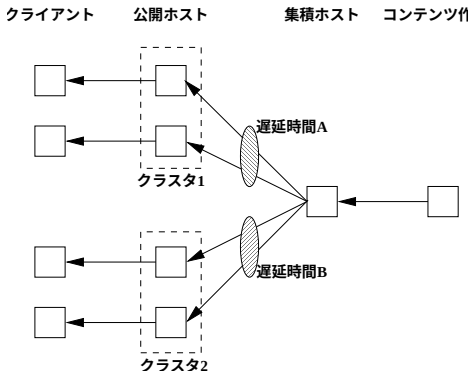

図 11 クラスタどうし異なる遅延を含む実験
Fig. 11 The network environment with several delays.

式ともコンテンツ不確定時間は 200〜300 ms 程度である．リクエストレートを増加させると NPS，Squid の場合は直線に沿ってコンテンツ不確定時間は減少し，NPS で 6 ms，Squid で 17 ms である．リクエストレートが毎秒 200 件以上では，一定値を示しており，これ以上リクエストレートを増やしても公開ホストが過負荷にならない限り変化しない．また，リクエストレートを公開ホストに対する負荷と見なすと，NPS では負荷が高い状態において，コンテンツの同一性を保持することができた．

4.3 実験 2：分散配置したクラスタにおける評価

本節では WAN 上にクラスタを分散配置する環境を想定し，集積ホスト，公開ホスト間に遅延発生装置を挿入する．まず，遅延時間を増加させ，更新を伝搬する他方式と NPS で遅延に対する特性を調べることにより比較を行った．次に 2 組のクラスタを分散配置する環境を構築し，コンテンツ不確定時間を測定することにより他方式と比較を行った．

4.3.1 実 験 方 法

遅延発生装置には FreeBSD に組み込まれた Dummynet[10] を用い，図 8 に示すように，集積ホスト・公開ホスト間に遅延発生装置を挿入する．これにより，集積ホスト・公開ホスト間の通信は遅延発生装置を介することになる．

2 組のクラスタにおける実験では，**図 11** のように，2 台で構成するクラスタを 2 組用意した．**表 4** に示すように遅延時間 A，B のパターンを 2 通り設けた．一方は国内のみ，他方は国内と海外にクラスタを配置する状況を想定している．実験 1 と同様にクライアントがリクエストを発行し，公開ホストの WWW サーバのログを調べることにより，コンテンツ不確定時間を算出した．

表 4 遅延時間の組合せ
Table 4 The combination of delay A and B.

	A	B
遅延パターン 1	10 ms	30 ms
遅延パターン 2	10 ms	100 ms

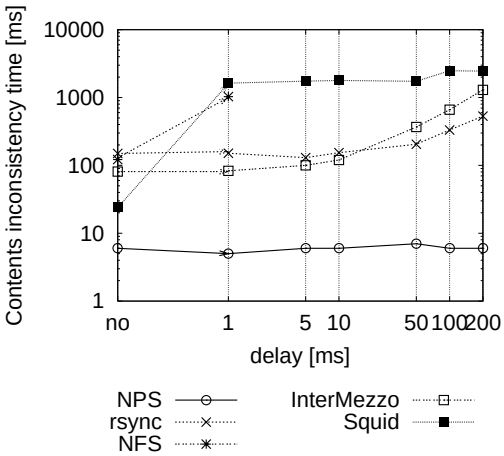

図 12 コンテンツ不確定時間と遅延時間の関係
Fig. 12 Contents inconsistency time versus delay.

4.3.2 結 果

各方式の遅延特性を調べた結果を示す．遅延時間とコンテンツ不確定時間の 80%tile 値の関係を**図 12** に示す．このときのクライアントからのリクエストレートは毎秒 300 件である．最左端の値は，遅延発生装置を挿入していないとき（実験 1）に得られた値である．他方式では遅延時間の増加とともにコンテンツ不確定時間が長くなった．一方，NPS では遅延時間にかかわらずコンテンツ不確定時間は一定で 6 ms である．NPS はコンテンツ更新の伝搬に関わる処理の時間差を吸収する機構であるため，遅延時間に影響を受けず，コンテンツの同一性を保持した．NFS は遅延時間が 1 ms より長くなると動作しなかった．これは集積ホスト，公開ホスト間の通信に時間を要するため，連続的に到来するクライアントからの要求に応答できないためだと考えられる．

次に 2 組のクラスタにおける更新の伝搬実験の結果を示す．この実験では，rsync と NFS は動作しなかったため対象から除いた．rsync では変化を検知するので処理に時間を要し，さらに遅延時間の影響を受けるものと考えられる．NFS では，集積ホストと公開ホスト間の遅延時間が長くなると通信に時間を要し，クライアントからの要求に応答できず，タイムアウトする．**図 13** に遅延パターン 2 のときのコンテンツ不確

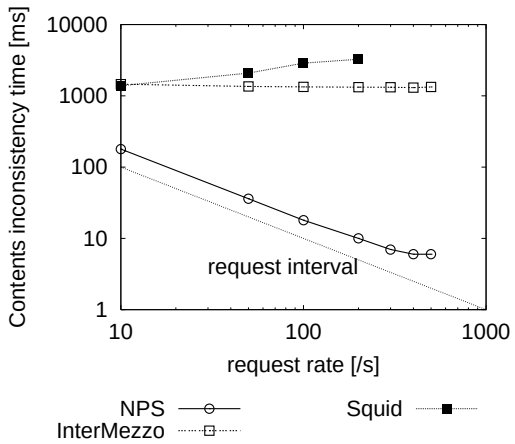

図 13 遅延パターン 2 のときのコンテンツ不確定時間とリクエストレートの関係
Fig. 13 Contents inconsistency time versus request rate.

定時間の 80%tile 値とリクエストレートの関係を示す．Squid では，毎秒 200 件より大きいリクエストレートでは動作せず，測定できなかった．クライアントからの要求が増加すると，この要求に連動して集積ホストから取得する処理が滞り，クライアントに応答できなくなる．NPS は，集中配置した場合（実験 1）と同じ値となり，遅延時間の影響を受けない．また，遅延パターン 1 においても，図 13 と同様のグラフを得た．NPS は遅延時間が長く，コンテンツ更新の伝搬に時間を要しても一斉に公開するため，コンテンツの同一性を保持することができる．

4.4 考　　察

NPS では，他方式に比べてコンテンツ不確定時間を大幅に削減することができた．ただしコンテンツ不確定時間の評価には，80%tile 値を採用しているため原理的な最小値であるリクエストの間隔より大きくなる．さらに，NTP による時刻同期の精度や NTP により修正した時刻を保持する OS の時刻の精度が，コンテンツ不確定時間に対して影響し，リクエスト間隔より大きい値が得られた．測定精度を向上させるために，リクエストレートを大きくして測定可能な最低値を小さくする必要がある．また，専用の OS やハードウェアを導入し時刻の精度を向上する必要がある．

NPS において，公開時刻を通知する通信が，遅延などの影響で公開時刻以降にリバースプロキシに到着する可能性がある．この場合，公開時刻を経過しているため，一斉公開はできずコンテンツの同一性を保持できない．このため，ある程度の余裕を持って公開時刻を決定する必要がある．ただし広域ネットワーク環境においては，ネットワークの遅延は一定でなく変化する．一方で長い余裕を与えると更新の伝搬に長い時間を要する．したがって，状況に順応する万能な公開時刻決定方法はない．このため，運用環境の状況を考慮して公開時刻を決定する必要がある．

4.5 運 用 実 験

NPS を第 83 回全国高等学校野球選手権大会の WWW サイトで運用し，現実の環境において有効性を確認するために評価した[11]．この実環境による運用の構成は，公開ホスト 3 台による集中配置したクラスタである．集積ホストと公開ホスト間の遅延を考慮し，過去の通信の平均時間の 1.2 倍の時間を見積もり公開時刻を決定した．更新したコンテンツについて，制御サーバが指定した公開時刻以降，クラスタ内のすべてのリバースプロキシが同一のコンテンツを提供しているかを調べた．その結果，1 日 1 万件の更新機会に対してほぼ 100%に近い確率で同一性の保持に成功した．なお，大会期間中の瞬間最大のアクセスは秒間 4500 件であった．NPS は現実のシステムで高負荷な状態においてもクライアントからの要求に応答しつつ，コンテンツの同一性を保持することができた．

5. ま と め

本研究では，負荷分散機構に頼らずコンテンツ供給側で WWW クラスタ内のコンテンツの同一性を保持することを目指した．既存の方式では，コンテンツの更新をクラスタに反映させる際に，コンテンツを格納すると即時にコンテンツは公開される．また各サーバでの処理時間に差が存在する．これらにより，クラスタ内に新旧のコンテンツが混在し，不一致となる．

そこで，NPS ではコンテンツを公開するまでの処理を格納と公開に分離し，コンテンツの公開をクラスタ内で揃えた．これにより，コンテンツの同一性を保持した．また，NPS ではコンテンツの格納処理時間の差を吸収することができ，負荷分散機構を操作する必要がない．

既存の代表的なコンテンツ更新を伝搬する方式と比較することで，NPS を評価した．疑似環境を LAN 内に構築し，公開ホストを集中的に配置したクラスタ，分散配置したクラスタに適用し，どちらの環境においても NPS はコンテンツの同一性を保持できることを確認した．NPS は WWW サイトにおいて，構成や配置にかかわらず適用可能である．

今後は，利用者の閲覧にあわせてページ単位のコンテンツ制御を行うことを計画している．

謝辞 本稿で提案した NPS を第 83 回全国高等学校野球選手権大会の WWW サイトに適用し，運用実験

させていただきました．貴重な実験の機会を与えて下さった関係組織の皆様，およびアクセスいただいた視聴者の皆様に深く感謝いたします．

参 考 文 献

1) Chamomile. URL: http://iplab.aist-nara.ac.jp/∼eiji-ka/chamomile/index.html
2) Postel, J. and Reynolds, J.K.: File Transfer Protocol, RFC 959 (Oct. 1985).
3) Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol — HTTP/1.1, RFC 2616 (June 1999).
4) Braden, R.T.: Requirements for Internet hosts — application and support, RFC 1123 (Oct. 1989).
5) rsync. URL: http://rsync.samba.org/
6) InterMezzo File System. URL: http://www.inter-mezzo.org/
7) Squid Internet Object Cache. URL: http://www.squid-cache.org/
8) Mills, D.L.: Network Time Protocol (NTP), RFC 958 (Sept. 1985).
9) http_load multiprocessing http test client. URL: http://www.acme.com/software/http_load/
10) Rizzo, L.: Dummynet: A Simple Approach to the Evaluation of Network Protocols, *ACM Computer Communications Review* (Jan.1997).
11) 西馬一郎，河合栄治，知念賢一，吉田豊一，香取啓志，山口 英：WWW クラスタにおける同期を考慮したコンテンツ更新機構の設計と実装．インターネットコンファレンス 2001 論文集，pp.131–140. 財団法人インターネット協会 (Nov. 2001).

(平成 14 年 3 月 18 日受付)
(平成 14 年 9 月 5 日採録)

西馬 一郎

2000 年大阪大学工学部応用自然科学科卒業．2002 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．同年，日本経済新聞社入社．

河合 栄治（正会員）

1996 年京都大学理学部数学科卒業．1998 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2001 年同博士後期課程修了．2000 年科学技術振興事業団さきがけ 21 研究員．工学博士．

知念 賢一（正会員）

1991 年琉球大学工学部電気工学科卒業．1995 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．1998 年同博士後期課程修了．同年，同大学情報科学研究科助手．工学博士．

山口 英（正会員）

1986 年大阪大学基礎工学部情報科卒業．1988 年同大学院修士課程修了．1990 年大阪大学情報処理教育センター助手．1992 年奈良先端科学技術大学院大学情報科学センター助手，同助教授．1993 年同大学情報科学研究科助教授．2001 年同教授．工学博士．

山本 平一

1963 年大阪大学工学部卒業．1965 年同大学院修士課程修了．同年，日本電信電話公社電気通信研究所入所．1990〜1992 年 NTT 理事・無線システム研究所所長．1992 年奈良先端科学技術大学院大学情報科学研究科教授．1994〜1996 年同研究科長．1997〜1998 年同大学副学長．1999 年同学情報科学研究科教授．工学博士．