AsagumoWeb: P2P 技術を用いた Web システム

AsagumoWeb: A Web System using P2P Technologies

池嶋 俊 † 阿部 洋丈 †† 加藤 和彦 ††,†††

Syun IKEJIMA† Hirotake ABE†† Kazuhiko KATO††,†††

† 筑波大学 情報学類

†University of Tsukuba, College of Information Sciences

†† 科学技術振興機構

††Japan Science and Technology Agency

††† 筑波大学大学院システム情報工学研究科

†††University of Tsukuba, Graduate School of Systems and Information Engineering

{ikeji,habe,kato}@osss.cs.tsukuba.ac.jp

P2P 技術を使用した Web システムを構築する AsagumoWeb を提案する。AsagumoWeb は P2P テクノロジーを使用して作成した分散データストレージを使う事によってサーバー負荷の集中を避け、高い availability を提供できる。本研究では、この AsagumoWeb を設計するにあたり、静的なページ及び動的なページを実装できるデータモデルおよび通信方式について説明する。

1 はじめに

近年、Web の利用者はますます増加している。Web の利用者が増えるに従って、サーバー負荷の増加が問題となってきている。現実世界行われている、店舗、窓口等のサービスは、物理的な制約から、サービス対象が限定されている。しかし、Web システムでは、世界中の人が同一の Web サービスを利用する事が可能である。サービス利用者数を予測する事は困難であり、既存の Web システムではサーバーがネットワーク/CPU 負荷に耐えられなくなる事もしばしばある。特に、掲示板サイトなどのコミュニティサイトでは、その傾向が顕著であり、なるべく低コストで負荷が分散されるシステムが望まれる。

本論文では、P2P 技術を利用したサーバーレス Web システムを提案する。既存の Web システムでは、クライアント・サーバーモデルを使用したシステムが用いられてきた。本提案システムでは、サーバーを廃し、既存の Web システムではクライアントとて用いられているユーザーのコンピュータをノードとし、ノード間だけでネットワークを構築する。

本提案システムは以下のような 3 つの特徴を持つ。

第 1 に本システムでは、Web アプリケーションは各ノード上で実行される。既存の Web システムでは、Web アプリケーションはサーバーで実行されるが、本提案システムでは Web アプリケーションへの処理要求はサーバーに送られるのではなく、各ノードで処理される。本システムのノード集合は、全てのノードから参照可能な単一のデータストレージを持つ。各ノードは処理に必要な Web アプリケーションをデータストレージから取り出し実行する。また、Web アプリケーションが要求する処理対象データも同じデータストレージから読み込まれる。Web アプリケーションが複数のコンピュータ上で実行されるモデルは Web サーバーをクラスタで構成する場合の手法で用いられているが、自律的に必要な Web アプリケーションを入手する点が本システムと異なる。

第 2 に、本システムでは、CPU 負荷が分散される。本システムでは、Web アプリケーションは、各ノードで実行され、1 台のサーバー上で実行される訳ではないため、CPU 負荷が集中する事はない。

第 3 に、本システムでは、ネットワーク負荷が自動的に分散される。通常の Web システムでは、クライアントからサーバーへの要求がネットワーク越しに行われるため、ネットワーク負荷が発生する。一方、本システムでは、ノードしか存在せず、ノードがクライアントとしてもサーバーとしても動作する。そのため、クライアントからサーバーへの要求はネットワークを経由して行われない。また、ノード間の通信も集中させず分散させる事が可能である。

負荷を分散させる既存手法としては、ネットワーク負荷分散手法として CDN(Content Delivery Network)[1][2] や Web キャッシングシステム [3] [4] [5] があり、CPU 負荷分散システムとしてサーバーをクラスによって構成する Web アプリケーションサーバーの分散などがある。それらの既存手法と比較して、本システムでは P2P 技術を使用した事で通常の Web システムでは発生しない問題が発生する。それは、ノード数が多く全てのノードに対しての処理を行えないために、保証できる事が少ない事である。そのため、本システムでは通信販売やオンラインバンキングなど、高度な保証が必要なサービスは対象とせず、コミュニティサイトなどの一時的に一部のデータにアクセスできない状況になっても損害が発生しないシステムを対象としている。

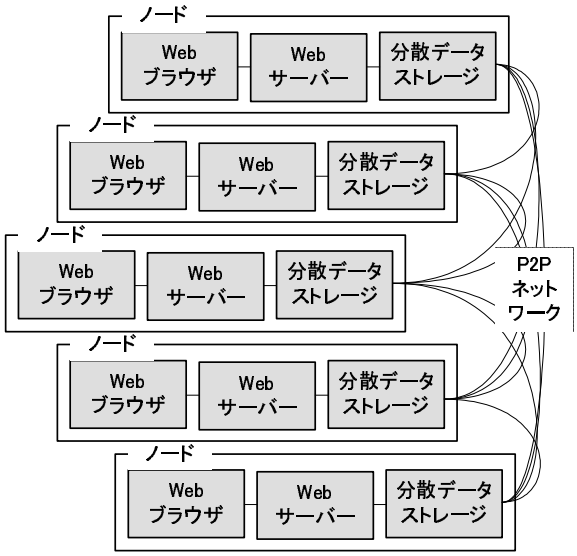

図 1: システムの外観

2 提案手法

2.1 概観

本提案システムのシステム全体を図 1 に示す。システムは複数のノードから構成されており、各ノードは P2P ネットワークで接続されている。各ノードは、通信を行いデータ交換している。通信方法は後述する。

システム内の全てのノードは Web ブラウザ、Web サーバー、分散データストレージの 3 つのモジュールから構成されており、ユーザーが操作するコンピュータ内にインストールされる。Web ブラウザモジュールはユーザーとのインターフェースとなる。Web サーバーモジュールは Web アプリケーションを実行する。分散データストレージモジュールは、他のノードと自律的に P2P ネットワークを構成する。

ユーザーからのアクセス要求は次のように処理される。まず、ユーザーは Web ブラウザを操作する。Web ブラウザから HTTP によって Web サーバーに要求が行われる。Web サーバーは要求に応じて、分散データストレージから Web アプリケーションを取り出し、実行させる。Web アプリケーションは分散データストレージとデータをやりとりしながら、処理を行い、結果を Web サーバー、Web ブラウザに返す。

2.2 設計上の課題

本システムを実装するにあたり、次のような 3 つの課題がある。

第一に、ファイル書き込みに関する排他制御の問題がある。本システムでは、同じ Web アプリケーションが 2 ノード以上のノード上で実行される。そのため、データの変更の排他制御に関して問題が発生する。全てのノードに対して通知を行い、ロック処理を行う場合、全てのノードと通信が行える状況でないとサービスを継続する事ができないため、可用性が下ってしまう。しかし、排他処理を行わない場合、お互いに通信できない状態にある複数のノード上で整合性の取れないデータに書き変えられてしまう可能性がある。

第二に、データ取得の保証の問題がある。データを持つ全てのノードがネットワークから切断されている場合、そのデータを他のノードは入手できない。分散データストレージでは、このような事を防ぐ事ができないため、全てのデータを入手できる事が保証されない。

第三に、データ ID の問題がある。Web システムでは、URI によって要求がなされる。そのため、データストレージ上でも URI に対応するデータを取り出せる ID が必要である。一般的な Web システムでは、ユーザーが指定したファイル名などを ID として URI が決定され、ファイルシステムから取り出される。しかし、分散されたデータストレージの場合、ファイル名である ID が衝突する可能性がある。これは排他処理と同じ原因による問題であるが。クライアントサイドまで影響がある可能性がある点が異なる。

2.3 本システムのアプローチ

本システムでは、2.2 で述べた問題に対して以下のようなアプローチを取る。

まず、データの排他制御に関しては、データ変更を許さない事で解決する。データは追加のみ許し、データの変更はデータの追加で表わす。

データ取得の保証に関しては、アプリケーションの記述時に工夫を行う事によって行う。本システム上で動作するアプリケーションは、入手できないデータが存在する事がある事を考慮に入れて設計を行い、取得できないデータが存在していても、取得できたデータを元に動作するように設計する。

データ ID の衝突の問題については、各データにユーザーが決めたファイル名を用いるのではなく、ユニークな ID を割り当てる事によって解決する。

3 設計

3.1 データモデル

本システムの分散データストレージでは、次のようなモデルでデータを保存する。

本提案では、Web ページの 1 ページを構成するデータをページと呼ぶ。本提案方式では、データはページ単位よりもさらに小さい単位でデータストレージに保存される。この小さい単位をブロックと呼ぶ。

1 つのブロックはただ 1 つのページに属しており、ページには複数のブロックが属している。(図 2)

各ページおよびブロックは、UUID(Universally Unique IDentifier) を ID として管理される。UUID は、ネットワークインターフェースに付与されている固有識別子とブロック作成時刻を組み合わせる事で実現可能である。LAN カードの MAC アドレスと時刻により生成された UUID は確実にユニークである事が保証される。これにより、異なるページに同じ ID が付けられてしまう事を避けられ、各ページはこの ID を元にしたユニークな URI で一意に選択する事ができる。

各ブロックには、そのブロックがどのページに所属するかを示すために、所属するページの ID が含まれている。ページには、ページに含まれているブロックのデータは含まず、あるページのブロックを列挙する場合は、ブロック全体の集合の中から、ページの ID に所属するブロックのリストを抽出する。

データの変更の衝突を避けるために、データストレージ上に保存された各ブロックは、書き換えを禁止し、ブロックの追加のみを許すこととする。ページの変更に相当する操作は、追記内容をアプリケーションにおいて適切に解釈することによって実現する。

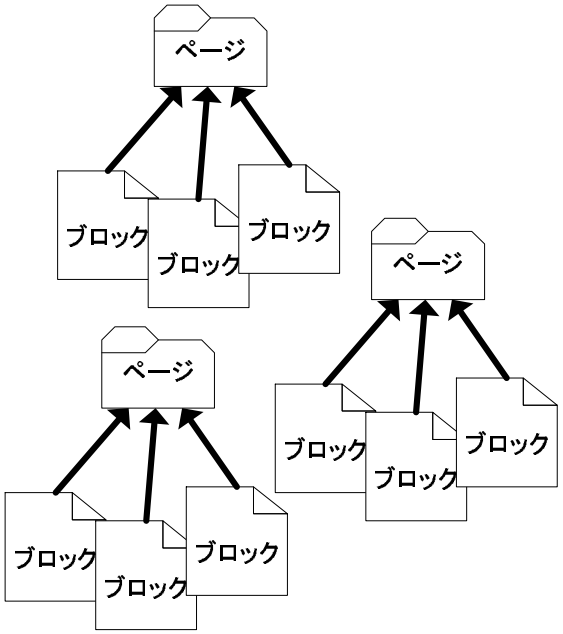

図 2: ページとブロックの関係

3.2 Web アプリケーション

本システム上で動作する Web アプリケーションは、他のデータと同様にデータストレージに格納されている。アプリケーションはユーザーからの要求に応じて、データストレージから取り出され実行される。

Web アプリケーションは要求に応じて、データの表示またはデータの変更を行う。

データの表示を行う場合、データを HTML に変換する必要がある。データストレージ内では、ページはブロックの集合である。Web アプリケーションは現在要求されているページ内のブロックを集め、表示を行う。(図 3) この時、全てのブロックを集められる事は保証できないため、それを留意してアプリケーションを作成する必要がある。例えば、掲示板ページでは、全ての掲示板ページ内のブロックが集められる事は保証できない。集められないブロック内の書き込みは表示しない処理を行う必要がある。

データの変更を行う場合、Web アプリケーションはユーザーからのデータをブロックに変換し、ページに入れる処理を行う。

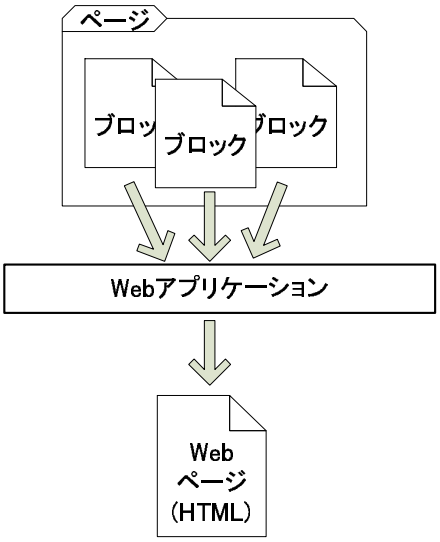

図 3: Web アプリケーションの動作

3.3 分散データストレージ

分散データストレージを設計するにあたって、データをどのようにコピーするかの戦略は 2 つ考えられる。

第一に、データの先行取得を行う戦略がある。これは、全ての構成ノードが全てのデータを持つ戦略である。新しいブロックが保存されたノードはそのブロックを全てのノードに転送し、全てのノードを最新に維持する。各ノードが全てのデータを持つ事により、既存の Web システム以上の性能を出す事が可能であると思われる。しかし、この手法は必要な物理ストレージの量が多く必要になるという問題がある。

第二の戦略として、データの遅延取得の戦略がある。これは、必要に応じてブロックを他のノードから取得するという戦略で、新しいブロックを保存されたノードはそれを保持しておき、他のノードがそのブロックを必要応じて取得するという戦略である。この戦略の場合、データストレージの総量は少なくて済むが、データ取得に通信が必要であるため、時間が遅くなると考えらえる。各ブロックは UUID が付加されているため、必要なブロックはネットワーク上で一意に決定できる。この戦略を取る場合 Chord[6]、Kademlia[7] などの DHT(Distributed Hash Table) を利用する事により、データ取得時間を短くする事ができる。各ブロックには UUID が設定されているため、DHT を応用する事は簡単である。しかし、この戦略では、ブロックを公開しているノードがダウンすると、他のノードがそのデータを持たないために、そのブロックを取得する事ができなくなるという問題がある。

この 2 つの戦略は、それぞれに利点・欠点がある。この 2 つの戦略を使い分け、データを入手しやすくするために第一の戦略を使い、必要ストレージ量を少なくするために第二の戦略を用いる。つまり、ブロックがネットワーク上から消失しないように、複数のノードに配り、その他のノードには必要に応じでブロックを取得するという方針を取る事が有効だと考えられる。複製は多い方がネットワークから消失する可能性が減り、必要物理ストレージが多く必要である。最適な複製の数とどのノードに複製数は今後の研究課題である。

本システムのモデルではブロックは変更不可であり、同じ ID のブロックは同じ内容を持つ事が保証できるため、分散ストレージはブロック内のデータの同期を行う必要はない。また、ファイル名ではなく、UUID を ID として使用するため、ID が衝突を回避する必要もない。

4 おわりに

本提案では、P2P 技術を利用した Web システムを提案した。Web システムに P2P 技術を利用する事により、ネットワーク負荷、CPU 負荷が分散される新しい Web アプリケーション開発の基盤を作成する事が可能になる。

今後の展望としては、本システムの有用性の定量的評価、ID をわざと衝突させるような悪意あるノードがある場合への対処、匿名での情報を発信するための手法の確立、情報を発信した者を特定するための手法の確立などがある。

参考文献

[1] Akamai, http://www.akamai.com

[2] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In Proceedings of the 11th International World Wide Web Conference, pp. 252-262. IEEE, May 2002.

[3] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address flash crowds. In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002), pp. 203-213, Cambridge,MA, USA, March 2002.

[4] I. Ari, B. Hong, E. L.Miller, S.A.Brandt and D.E.Long. Managing Flash Crowds on the Internet. Proc. MASCOTS 2003.

[5] S. lyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. Proc. PODC2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-topeer lookup service for Internet applications. In Proc. ACM SIGCOMM, Aug. 2001.

[7] MAYMOUNKOV, P., AND MAZIERES, D. Kademlia: A peer-topeer information system based on the xor metric. In Proceedings of (IPTPS) (2002).