分割アクセスによる大容量ファイル取得の高速化

01T268　藤原信孝（最所研究室）

本研究は、ミラーサーバに同時にアクセスして、大容量ファイルに対してのデータ転送の効率および信頼性の向上を目指すことである。この機能を実現する際の課題や方策について議論する。

1．はじめに

現在インターネットは、世界的に広く普及し、重要なコミュニケーションツールとなっている。しかしながら近年、インターネット人口の増加・高速回線の普及に伴うネットワーク負荷の増大、ワームの被害やDoS攻撃などによる外的要因、サーバメンテナンスなどの内的要因などにより、サービスが提供できない、あるいは大幅に低下するという状況が発生している。　これに対し、現在特にアクセスの多いサーバでは、同様のサービスを提供するサーバを複数設置し、それらにアクセスを分散させるためにミラーリングを行っている。

我々の研究室では、このようなミラーサーバ群にアクセスすることによる応答時間の短縮、および信頼性の向上に関する研究を行っている。また、HTTP/1.1[1]に備わっている、持続的接続機能・パイプライン機能を利用し、より高効率のプロキシ機構を開発してきた。これらの通信形態は、これまでの研究[2]の結果、若干の問題点が残ってはいるものの効果的であることが判明している。

そこで本研究では、これまでの研究では分割接続を利用しての、ミラーサーバへの同時アクセスによる大容量のファイルリクエストに対する取得速度の向上を目的とした通信システムの開発を行う。

2．通信形態

プロキシプログラムの動作の流れを図1に示す。

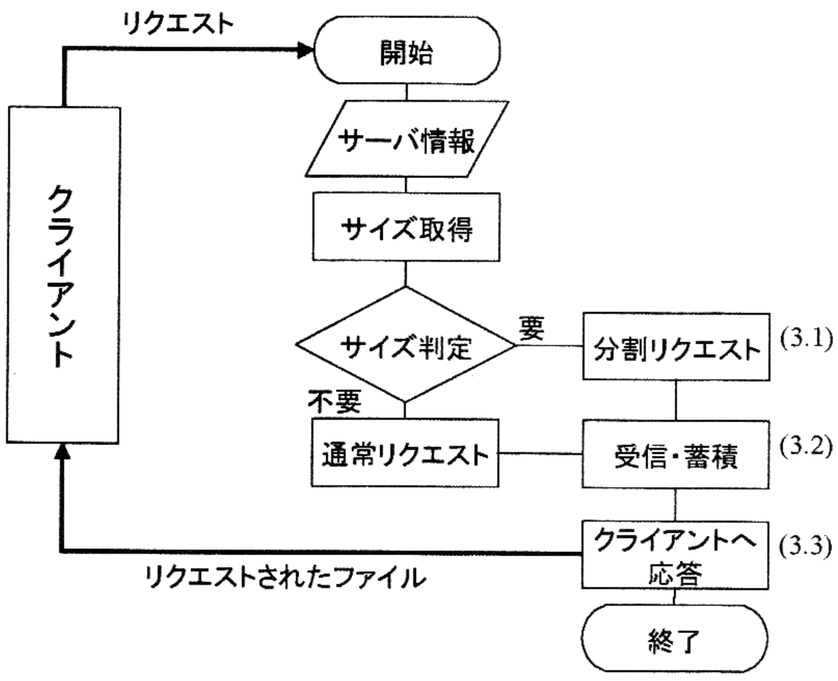

図1．プロキシ動作のフローチャート

プロキシプログラムは、クライアントからリクエストを受けることで起動する。このときリクエストとサーバ情報が与えられるものとする。

起動後、対象のサイズを取得し、そのサイズを基準値と比べる。リクエストされたファイルのサイズが小さい場合は分割する必要がないか、逆に分割、再構成の手間が増えるだけでメリットがないためである。

分割する場合は、右に進む。リクエストの部分では、後述するそれぞれの方式に即して始点と終点を計算し、割り当てられたサーバへリクエストする。

リクエストの応答として、ファイルの本体が送られてくると、それを蓄積する段階に移り、蓄積が決められたある段階まで達すると、応答としてクライアントに送り出す。

ここで、応答を得るまでに経過する時間は、サーバの能力や、負荷の状況によって異なる場合が普通である。本研究では、いかにサーバの能力を見極めるかが要点の1つとなる。

3．要求技術の検討

前章図1に対応する分割接続の各段階において考慮すべき項目について、具体的な方式を述べる。

3.1 分割方式

プロキシの中心的部分であり、全体の構成を左右する。サーバ数分割、定数分割、固定サイズ分割が考えられる。

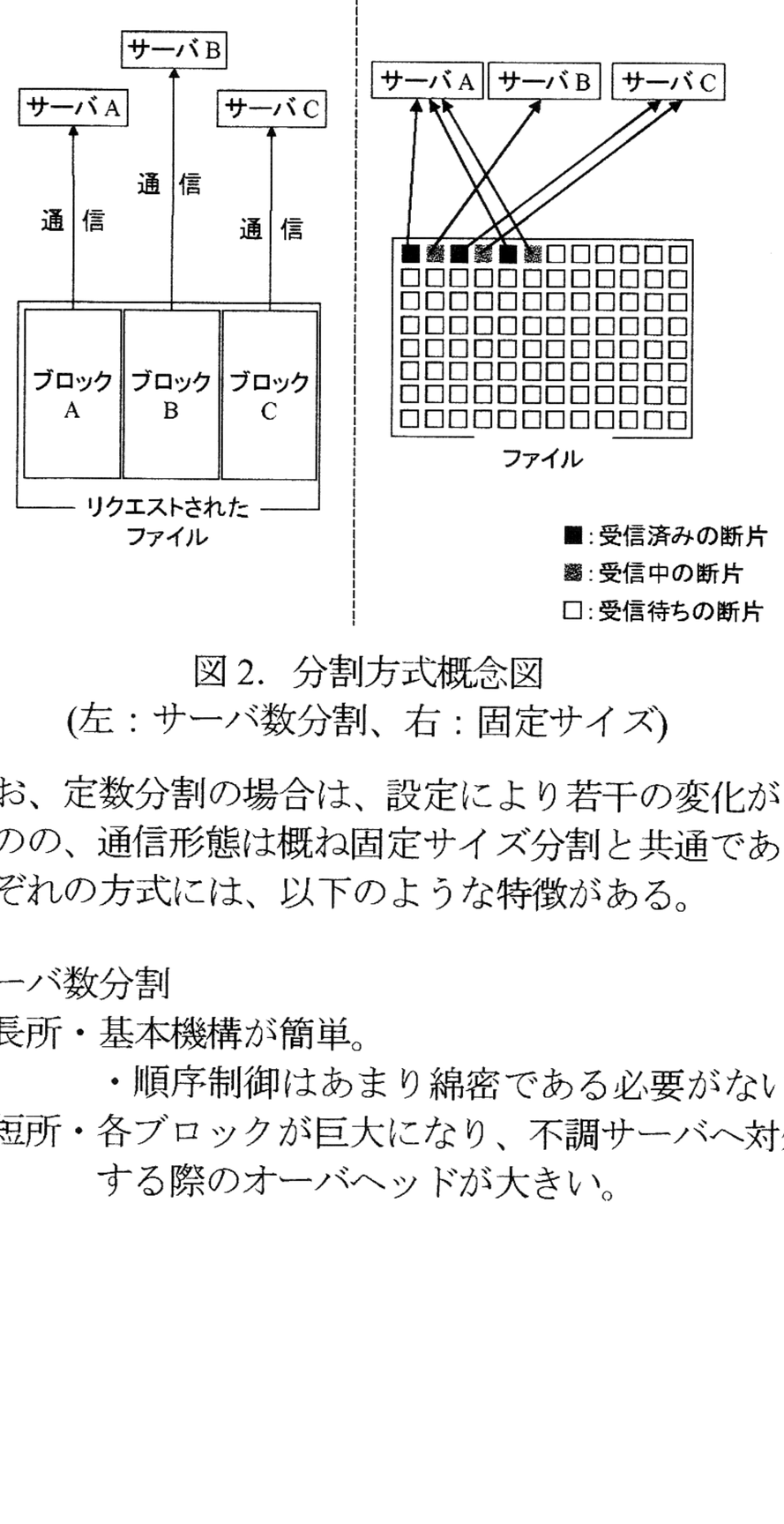

図2．分割方式概念図
(左：サーバ数分割、右：固定サイズ)

なお、定数分割の場合は、設定により若干の変化があるものの、通信形態は概ね固定サイズ分割と共通である。それぞれの方式には、以下のような特徴がある。

サーバ数分割
- 長所・基本機構が簡単。
- 順序制御はあまり綿密である必要がない。
- 短所・各ブロックが巨大になり、不調サーバへ対処する際のオーバヘッドが大きい。

・最も遅いサーバがプロキシ全体の性能に及ぼす影響が大きい。

固定サイズ分割

長所・不調サーバへの対処によるオーバヘッドが小さい。
・速度差の大きいサーバ群に対しても、性能が低下しにくい。

短所・ファイルサイズとブロックサイズの調整をする必要がある。
・配置制御が必要、かつ膨大。

3.2 取得データの蓄積

取得したデータブロックを蓄積する方法であり、事実上の再構成方法である。スライディングウィンドウと穴埋め式が考えられる。概念図を図 3.に示す。

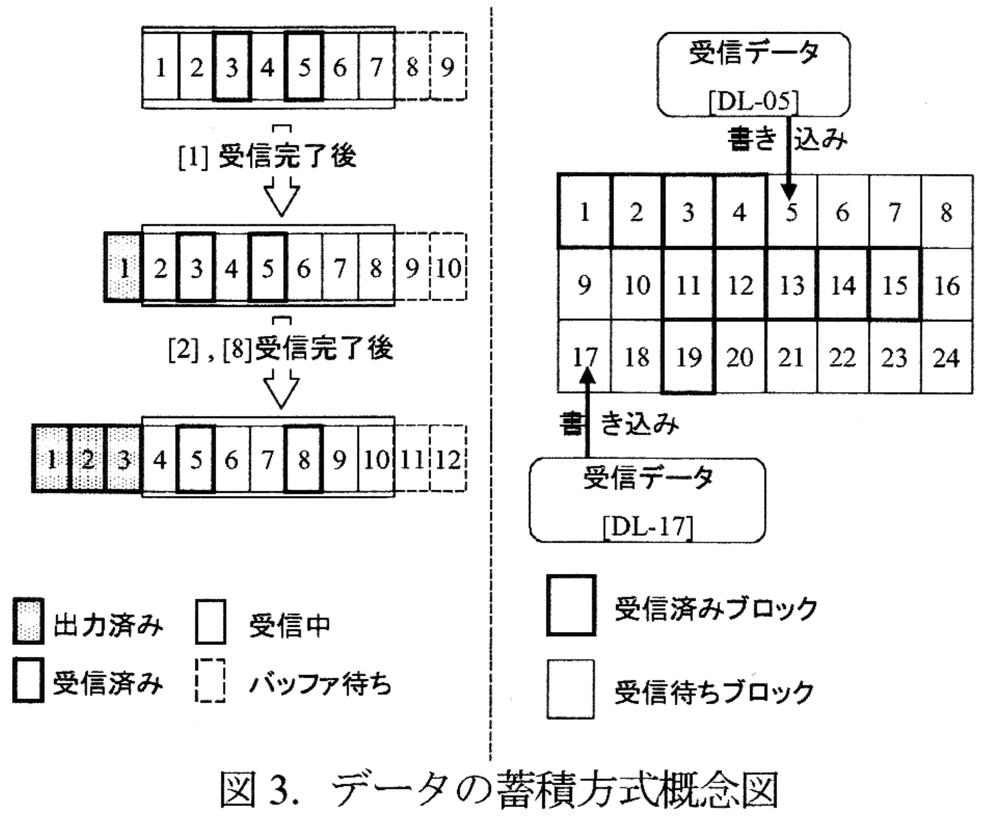

図 3. データの蓄積方式概念図
(左：スライディングウィンドウ、右：穴埋め式)

スライディングウィンドウは、バッファサイズが小さくストリームデータに向いており、穴埋め式は、低速のサーバから影響を受けにくく全体を一括で取得するデータに向いている。

3.3 クライアントへの応答時期

リクエストされたファイルの到着通知を、どの段階で行うかのタイミングである。全完了時、先頭部到着時が挙げられる。概念図を図 4.に示す。

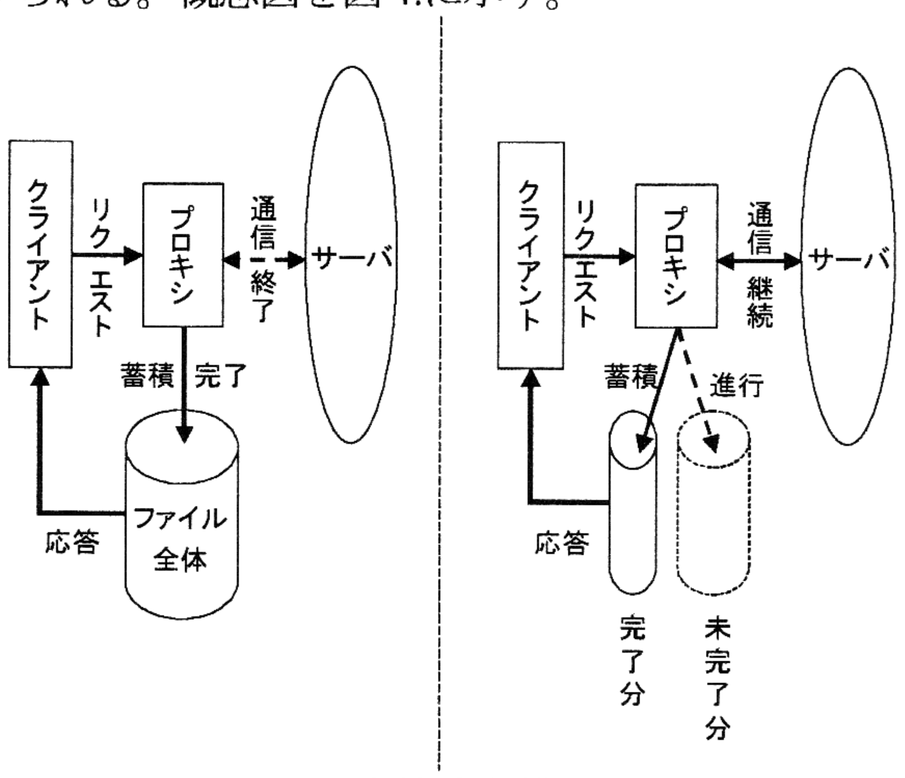

図 4. 応答時期模式図
(左：全完了時、右：先頭部到着時)

対象とするファイルの違いから、蓄積方法と関係が深く、全完了時は穴埋め式と、先頭部到着はスライディングウィンドウと相性がよい。

3.4 応答の遅いサーバへの対処

プロキシの応答速度向上のための技術である。以下の 3 方式が考えられる。

完全切捨て：対象サーバをサーバリストから削除し、以降一切通信しない。
ダミー通信：リクエスト送信を停止し、ダミーのリクエストで、状況を観察する。
並行処理 ：別のサーバへ同一部分を予備としてリクエストする。

完全切捨てでは、サーバの影響を完全に遮断でき、ダミー通信が、サーバ、ネットワーク資源面で優れ、ファイル取得の確実性では並行処理が優れている。

4. 実装の状況

以上、分割転送方式のそれぞれの段階における手順について、その特性を議論してきた。現在、これらの方式を実現するためのプリミティブを作成中である。

プロトタイプ作成は、開発環境を Windows + Borland C と定め、POSIX スレッドを用いたプロキシプログラムの開発を行っている。しかし、現在のところ、実験を行えるまでには開発が進んでいない。

現時点での問題点として、

- 各々の子プロセスについて、通信の終了を即座に察知できないこと
- 保存する際にファイルが破損すること

が挙げられる。

5. まとめ

分割接続の各段階における各方式は、対象とするファイルや各方式間の相性によって、実用的、または実現可能な組み合わせは限られる。以下に一例を示す。

表 1. 方式組み合わせの例

	分割方式	蓄積方式	応答時期
方式 1	サーバ数	穴埋め	先頭部
方式 2	固定サイズ	スライディング	先頭部
方式 3	固定サイズ	穴埋め	全完了

しかし、プロキシプログラムの開発が遅れており、これらの有効性についての実験を行うまで至っていない。

文献

[1] IETF, "Request for Comments: 2616 Hypertext Transfer Protocol-HTTP/1.1", June 1999.

[2] 藤澤弘, “複数のミラーカーバへの分散アクセスによる効率的なアクセス機構に関する研究”, 香川大学大学院信頼性情報システム工学専攻 修士論文, 2004.