並列ダウンロードシステムによるデータ転送の高速化

著者　小林由人　　指導教員　高久有一

1. はじめに

インターネット上から大容量のデータをダウンロードする際に、ファイルを所有しているサーバ間とのインターネット環境が悪いと多くの時間を要してしまう。この問題を解決するために、別の場所にある同じファイルを所有するサーバを用いて、異なる回線から同時にファイルのダウンロードを行う。本研究では、以上のような処理でダウンロードのデータ転送の高速化を試みた。本研究で想定するシステムのイメージを図1に示す。

図1. システムのイメージ

2.システム概要

クライアントとサーバが1対多で通信を行い、データをダウンロードする。システム上のすべてのサーバは同じデータを所持している。1つのファイルを複数のブロックに分割する。次に、このブロックの取得を複数のサーバから並列に行う。この処理によってファイル取得の高速化を図る。ダウンロードのイメージを図2に示す。

また、クライアントがサーバの1台に接続し、他のサーバの情報を得てダウンロードを行うシステムとする。したがって、クライアントはシステム上のいずれか1台のサーバに関する情報でシステムを利用することが可能となる。本システムでは同じプログラムやファイルを所有しているサーバが様々な通信環境のもとに複数存在していることを前提としている。そのため、クライアントが初めに持っていたサーバとの通信環境が悪かった場合に、他の通信環境がよいサーバからのデータのダウンロードも行うことができ、より短い時間でのダウンロードが行うことができると考えられる。今回の研究では、各サーバのプログラムやファイルの同期は手動でコピーをして実現する。

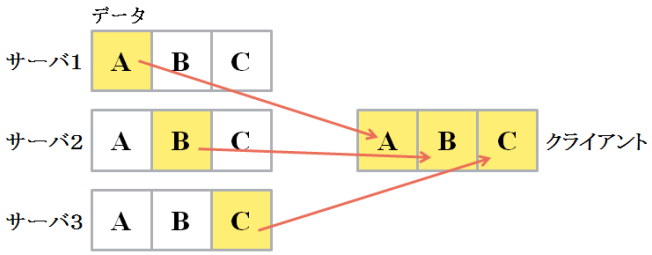

図2. ダウンロードのイメージ図

3. 技術概要

3. 1. Java

Sun Microsystems 社が開発したプログラミング言語である[1]。C 言語に似た表記法を採用しているが、既存の言語の欠点を踏まえて一から設計された言語であり、最初からオブジェクト指向性を備えている点が大きな特徴である。強力なセキュリティ機構や豊富なネットワーク関連の機能が標準で用意されており、ネットワーク環境で利用されることを強く意識した仕様になっている。また、Java で開発されたソフトウェアは特定の OS やマイクロプロッセッサに依存することなくどのようなプラットフォームでも動作する。

今回の研究ではネットワーク環境で利用されることを強く意識した仕様という点とどのようなプラットフォームでも動作するという点に着目し Java を用いることに決定した。また、ユーザが実際にシステムを使用するとなると GUI の実装が必要不可欠である。そこで Swing などの GUI 実装のためのライブラリも用意されている Java を用いる。

3. 2. スレッド

スレッドを用いることにより、1 つのプログラム上で複数の処理を同時に行うことができる。本研究のシステムでは、サーバが複数のクライアントと同時に通信を行う。そこで、クライアントと接続を確立するとスレッドを用いて処理を行う。また、クライアントは複数のサーバと同時に接続しデータをダウンロードするため、この際にもスレッドを用いて並列処理を行う。

4. システムを用いた通信速度の測定

ファイルを所有するサーバ間のインターネット環境が悪い場合に、データを分割して別の場所にある同じファイルを所有するサーバから同時にダウンロードすることで通信速度が速くなるのかを確かめる。Java ネットワークプログラミングを使用し、図 3 の各通信路で 1Gbyte のファイルのダウンロードに要する時間を測定する。

現在のインターネットの標準の通信速度は 1Gbit/sec 以上である。そこで、実際のインターネットの環境を再現するために通信速度が 1Gbit/sec の HUB を入れた場合である構成Ⅰと Ⅱで測定を行う。しかし、実際のインターネットでは通信速度が遅いところも存在する。そこで、通信速度が 100Mbit/sec の HUB を入れることで意図的に通信速度が遅くした構成Ⅲと Ⅳで測定を行う。また、2 台のサーバからダウンロードを行う際に、通信速度が遅いサーバから同時にダウンロードを行った場合を構成Ⅴ、速いサーバからダウンロードを行った場合を構成Ⅵとする。

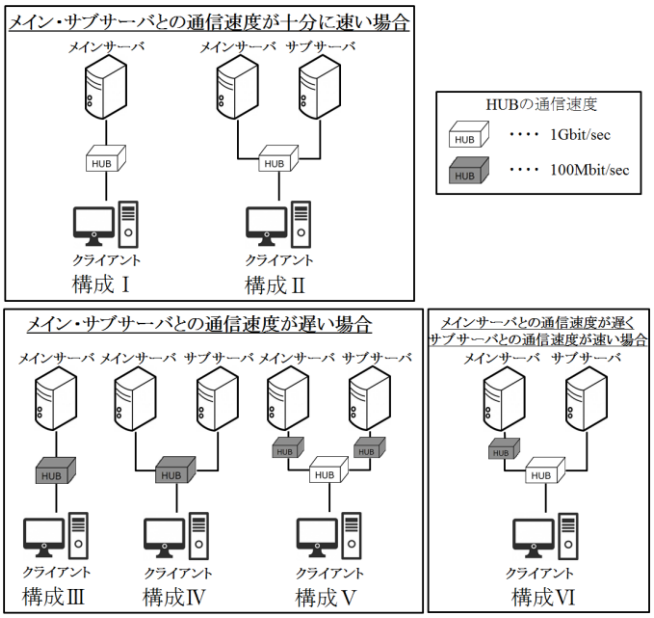

図 3. 通信路の構成

5. 測定結果

各構成における測定結果を表 1、図 4 に示す。ダウンロード前と後のデータは Linux の diff コマンドを使用して同じものであることを確認した。

まず、通信速度が速い場合の構成Ⅰと Ⅱを比較すると、1 台からダウンロードをした場合の方の通信速度が速いという結果になった。

次に、構成Ⅲと Ⅳで測定を行った場合のデータの通信速度は 100Mbit/sec に近いことがわかる。よって、この通信路の通信速度の上限に達していると考えられる。構成Ⅴと Ⅵを比較することで、最初にダウンロード要求を行ったサーバ間の通信速度が遅い場合に、異なる場所にあるサーバから同時にダウンロードを行うことで処理時間が短くなることがわかる。

表 1. 測定結果

構成	処理時間(sec)	速度(Mbit/sec)
Ⅰ	9.2	918.8
Ⅱ	10.0	753.3
Ⅲ	82.1	91.8
Ⅳ	80.3	93.9
Ⅴ	41.6	181.1
Ⅵ	44.1	170.8

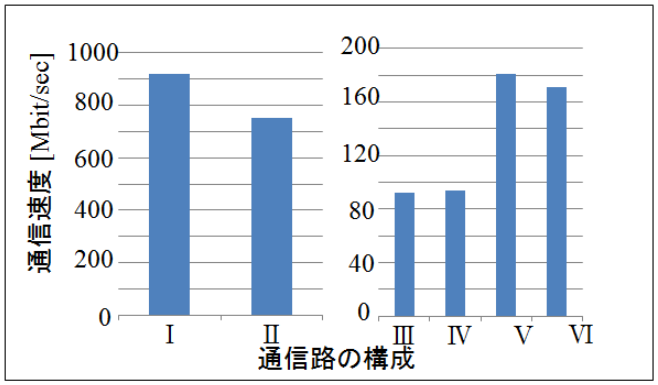

図 4. 各通信路の測定結果

6. まとめ

今回作成したシステムを使用することで、ファイルを所有しているサーバ間とのインターネット環境が悪い場合に、異なる回線から同時にファイルのダウンロードを行うことでデータ転送速度の高速化を実現することができた。しかし、実際のインターネット上にはさらに通信速度が遅いところがあると考えられる。そこで、プログラム上で意図的にデータの転送速度を遅くして、通信速度が 100Mbit/sec 以下の場合でも同じように測定を行わなければならない。

今後の課題として、サーバ間の通信速度に合わせて、ダウンロードするデータの大きさの割合を調整ができるようにする。このことにより、さらに通信速度を速くすることができると考えられる。また、GUI の実装など実際に使用する際に、ユーザが使いやすいようにする。

[参考文献]

[1] 初めての Java 入門
http://www1.bbiq.jp/takeharu/