

# 動的適応可能な分散ファイルシステムの提案

孫 静涛<sup>†1</sup> 佐藤一郎<sup>†2</sup>

分散ファイルシステムにおいて、負荷により分散ファイルシステムのアーキテクチャを動的適応させる手法を提案する。本提案手法は分散ファイルシステムをクライアント・サーバ方式または Peer-to-Peer 方式の双方向に変化させることができる。

## Dynamic Adaptation in Distributed File Systems

JINGTAO SUN<sup>†1</sup> ICHIRO SATOH<sup>†2</sup>

This paper proposed dynamically adaptive distributed file architecture according to the load balancing based on distributed file system. The proposed can be changed in both directions of the Peer-to-Peer architecture or Client/Server architecture.

### 1. はじめに

クラウドコンピューティングをはじめとして、分散システム上にデータやファイルを保持することが多くなっている。その基礎となる分散ファイルシステムが数多く登場している。分散ファイルシステムは多数のコンピュータから様々な目的に利用されるが、いずれも固定的なアーキテクチャであり、利用状況に最適化されているとは限らない。そこで、利用状況や負荷に応じて、そのアーキテクチャを含めて変更できるようにすることで、最適化を実現していく。本研究では分散ファイルシステムがクライアント・サーバ方式または Peer-to-Peer 方式であることを考慮して、両方式を含めて動的適応できる分散ファイルシステムの実現方法を提案する。

例えば、サーバが過負荷になる場合に、サーバの負荷を軽減するために、実行しているプログラムを他のコンピュータに移動することにより、サーバ負荷を軽減することが有効だと思う。また、クライアント側の負荷があがる時に、クライアント側に移動させたプログラムを再びにサーバ側に戻ることによって、双方向に変換可能な分散ファイルシステムについて具体的に論じる。

以下の論文構成を述べておく。第2章は既存の分散ファイルシステムと負荷分散について説明する。第3章は、本提案手法の相転換条件と提案メカニズムについて説明する。

第4章はソフトウェア移動を前提として、本提案手法の応用する可能な具体例を示す。第5章はまとめと今後展望について述べる。

### 2. 既存研究

提案手法の適応性を実現するため、我々は従来の分散ファイルシステムの構成及び分散ファイルシステムの負荷分散対処について、調べていた。

#### (1) 分散ファイルシステム

分散ファイルシステム NFS (Network File System) ローカルに接続されたストレージをネットワークに介してリモートのコンピュータにデータの共有するために設計されたものである。近年では、GFS(Google File System)[1]やHDFS(Hadoop File System)[2]やGlusterFS[3]などがあげられる。既存分散ファイルシステムは三種類に分類できる。

- 集中管理式分散ファイルシステム。
- Peer-to-Peer 分散管理式分散ファイルシステム。
- 分散ハッシュテーブル管理式分散ファイルシステム

集中管理式というのは集中管理サーバを持ち、被管理者側のデータを統一管理する分散ファイルシステムである。集中管理サーバにより、被管理者の面倒をみてくれるので、管理しやすい、その反面、集中管理サーバの処理が遅れたら、システム全体が遅れるので、システムのネックにもなりやすい。体表例として、GFS や HDFS などがある。Peer-to-Peer 分散管理式というのは、集中管理式と違い、集中管理サーバをもたず、その代わりに個々のコンピュータは

<sup>†1</sup>(国立)総合研究大学院大学 / 国立情報学研究所  
The Graduate University for Advanced Studies /  
National Institute of Informatics

<sup>†2</sup>(国立)総合研究大学院大学 / 国立情報学研究所  
The Graduate University for Advanced Studies /  
National Institute of Informatics

対等関係を持ち、対等なコンピュータ同士自身はデータの管理とデータの共有二つの機能を持つ分散式ファイルシステムのことである。例えば、楽天による LeoFS などがある。

しかし、既存の分散ファイルシステムは利用状況が変化とともに、アーキテクチャ自体を適応できない。

## (2) 負荷分散

既存集中管理分散ファイルシステムでは、集中管理サーバの負荷が問題となるが、それを軽減させる方法としては下記がある。

- キャッシュの粒度の設計
- 帯域幅の可変性[4]
- レプリカファイルの可用性

近年ファイルのサイズが大きくなり、そのファイルは一旦保存されたら、他のサーバに移動するのはコストがかかる。

既存研究の適応性を実現する研究方法としては、

- パラメータなどの変更
- 事前予測不可能なソフトウェアの書き換え
- 予測可能なソフトウェア書き換え

などの手法が提案されている。2つ目の例として遺伝アルゴリズム[5]、3つ目の例とポリシーベースの書き換えなどがある。本研究では分散ファイルシステムを構成するプログラム機能の一部をコンピュータ間で移動させることで適応化させて、負荷を分散させる方法を狙っている。

## 3. 提案手法

本研究では、ソフトウェア移動を用いた動的適応について説明する。

分散システムアーキテクチャはクライアント・サーバモデルと Peer-to-Peer モデルがある。クライアント・サーバモデルは、クライアントとサーバの機能を分離することを目的で、提案されたコンピュータネットワークのアーキテクチャである。

また、Peer-to-Peer モデルは、多数のコンピュータ同士は対等関係を持ち、対等者同士はお互いに情報の提供と情報の共有を同時に実行できるという目的で提案されたコンピュータネットワークアーキテクチャである。

両方式は、システムの構成はもちろん、利用状況によって、向き不向きの場合がある。どちらかのアーキテクチャには環境変化に応じて、適応できない。

そこで図1に示したように我々はシステムの利用状況に応じてクライアント・サーバ方式または Peer-to-Peer 方式の優位な方式に、動的かつ自律的に双方向に変化できるア

ーキテクチャを提案する。

そこで、我々は、サーバ側に保存されているファイルの場所を変わずに、モバイルエージェントを用いて、サーバ側が実行している機能をクライアント側に送る。

そうすると、サーバ側のリクエスト回数が増え、過負荷になるときに、クライアントがリクエストする機能はサーバ側で実行するのではなく、リクエストしたクライアントは処理することによって、サーバ負荷を軽減することは有効だと思う。

例えば、サーバ側が実行している一部のコードをクライアントに移動することで、このクライアントがサーバの処理の一部肩代わりし、さらに Peer-to-Peer 方式の中の Peer に相当することもできるようになる。逆に Peer-to-Peer 方式においては、特定の Peer はファイルの提供を頻繁に求められる場合がある。その Peer をファイル提供の専用にする、つまりクライアント・サーバ方式に移行させることを考える。

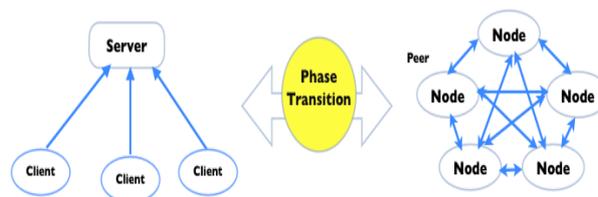


図 1: 相転換アーキテクチャ

### 3.1 動的適応に変換条件

動的適応が必要な事例について紹介するとともに、その適応内容及び適応条件を考察する。

#### 3.1.1 計算リソースの利用状況が違い

マルチメディアコンテンツの配信では一時的にサービスの利用者からコンテンツ配信サーバにアクセスを集中することがある。このときに、配信を行うサーバの負荷が上がるのが想定される。負荷の状況に応じて、サーバ側の処理の一部をクライアント側に任せることで負荷をさげられるはずである。ここでは配信サーバのように、一時的に多数のクライアントから集中管理サーバにリクエストを送るときに、サーバの負荷を減らすことは動的適応の条件となる。

#### 3.1.2 ネットワークの状況

計算リソースの利用状況により、急増なアクセスが起きるが、それはネットワークの不安定やネットワークの輻輳につながり、クライアントまたはサーバ間の受送信に遅延と中断が起こることも想定される。これも動的適応の条件の一つになる。例えば、ここでの通信遅延に関する測定方法としては、ping や traceroute などの測定ツールを使用する。それによって、クライアント側から定期的に集中管

理サーバに遅延測定パケットを送信し、遅延時間を測る。転送速度に影響を与える遅延時間だと判断すれば、動的適応を自律的に行う。

図 2 に適応条件を示す。

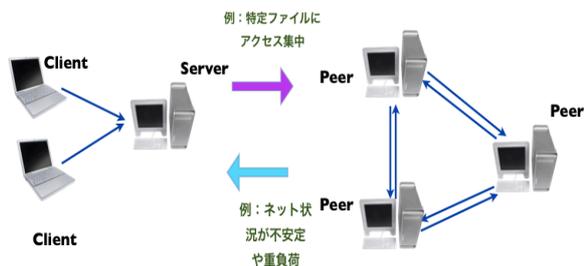


図 2: 相転換条件

### 3.2 ソフトウェア移動

動的適応を実現する方法として、動作ポリシーやパラメータを変える方法や、遺伝アルゴリズムに代表されるようにソフトウェアそのものを変更する方法などが提案されている。前者は適応可能範囲が限定され、後者は適応性のために大量の計算リソースが必要となることと、適応内容が事前に予測できないことから、システムの信頼性などを保証できないという問題がある。そこで分散システム上においてソフトウェアの実行位置を変えることによる動的適応を実現する。

例えば図 3 のように監視サーバは通信ツールを用いて、ファイルサーバの CPU の使用状況やネットワークのトラフィックなどを監視している。クライアントからファイルサーバへのリクエストが増え、ファイルサーバの処理能力は低負荷状態から負荷状態になる前に、監視サーバからファイルサーバにファイルサーバの現在の状況をお知らせして、ファイルサーバ側の一部のソフトウェア機能をクライアント側に移動させ、クライアントがその機能を実行する。

また図 4 のようにクライアント側のアクティビティモニタ(赤い円)からクライアントの CPU, システムメモリ, ネットワークの受送信情報を監視する。クライアント側は低負荷から負荷状態になる前に、監視結果をクライアント自身にお知らせ、クライアント側が実行しているソフトウェア機能をサーバ側に戻して、サーバ側がその処理を実行する。

そして、ソフトウェア機能の移動にはモバイルエージェント技術を用い、移動対象となる機能はモバイルエージェントとして実装されていることを前提にする。

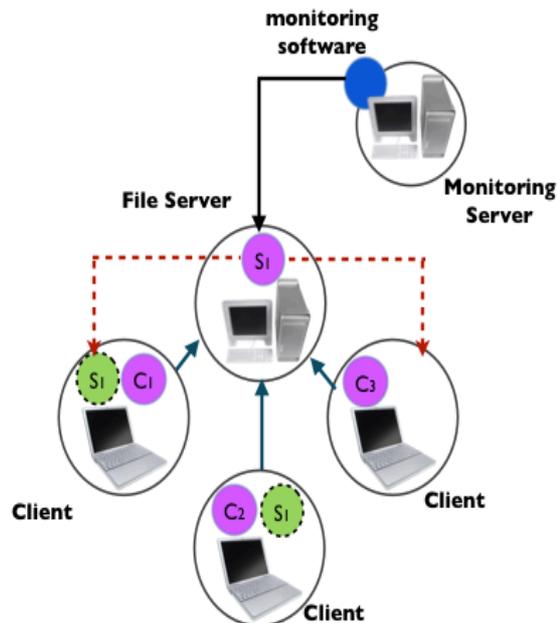


図 3: サーバ側のソフトウェア機能の移動

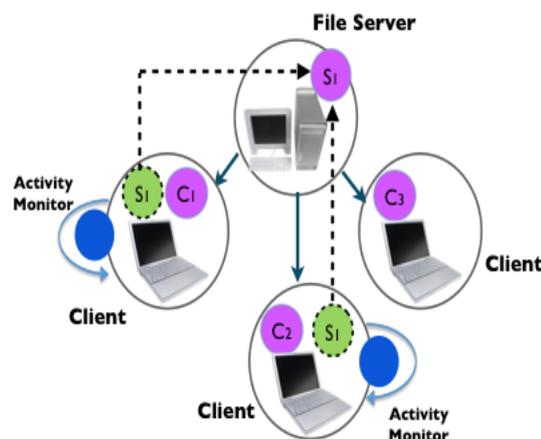


図 4: クライアント側のソフトウェア機能の移動

### 3.3 ソフトウェア移動の実現方法

ここからソフトウェア移動の実現方法について説明する。

図 5 の赤線に示したように既存集中管理式分散ファイルシステムでは、クライアントから集中管理サーバにリクエストを送る回数が増えれば、集中管理サーバ側の処理が遅くなり、過負荷を生じることがある。

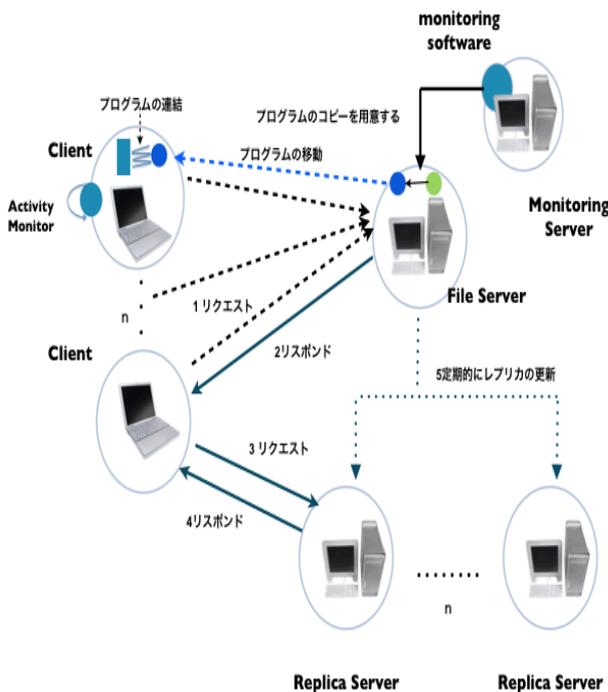


図 5:マスターサーバを生じる負荷

そこで,本研究は,様々な機能を持つサーバ側の中から,クライアント側に最適化しやすい機能をクライアントに移動することで,環境変化に応じる動的適応する.例えば,クライアント側に応じてフレームレートを変更する場合や,通信量を減らすためにファイルサイズを圧縮する場合などが想定される.

研究対象となる機能を実現するためには,

まず,図5に示したように,監視サーバからファイルサーバに負荷状況をお知らせしたときに,ファイルサーバ側が実行しているプログラムそのものの複製品を作り,レプリカとしてサーバ側に保存しておく.つまり,図5に示したように緑色円はサーバ側のオリジナルのプログラム,青い円はレプリカである.

次にファイルサーバ側のオリジナルプログラムは一時的に休止し,ロックをかけておく.監視サーバから再び再開指示までサーバ側にて機能しない.また,その青い円に示したようにレプリカはクライアント側に送ってもらう.クライアント側はそのレプリカを受け取った後に,図5に示したようにクライアント側の元機能とファイルサーバから受け取ったレプリカは異なったプログラムなので,ここで,二つのプログラムをマッチングする.そのマッチング方法として,以下の二つを挙げる.

一つ目は実行しているコードを Byte コードに変更による適応性を対処する.

二つ目は動的なメソッド呼び出し方式である.例えば, CORBA(Common Object Request Broker Architecture)や

EJB(Enterprise JavaBeans)など分散オブジェクト技術があり.いずれにプログラムをコンポーネント化して,ネットワークを経由で,他のプログラムからメソッドを呼び出せる.

Byte コード変更方式はサーバとクライアントの両側に行っているコードを変更しないと行けなく実行中システムのコストが大きというデメリットがある.動的なメソッド呼び出し方法とはプログラムコードをその機能や呼び出し方の情報と共に一種のカプセル化し,コンピュータネットワークを経由して他のプログラムからメソッド呼び出せるので,プログラミング言語やプラットフォームに依存しないメリットがある.

前者による負荷をあげるために,本研究は後者を用いる.なお,後者は呼び出し側に明示的な記述が必要となるが,分散ファイルシステムの処理自体を制限するものではない.

### 3.4 移動による増えた機能と欠けた機能の対処

ソフトウェア移動ことによって,クライアント側とサーバ側でソフトウェア機能の追加が起きるが,新たに追加された処理を呼び出す場合は,動的なメソッド呼び出しによる疎結合を利用する.なお移動元はソフトウェアが残っているためにそれを呼び出すことで対処できる.

### 3.5 ソフトウェア移動による動的適応の欠点

ソフトウェア移動はデータ移動と比べると移動対象が小さく,ネットワークへの負荷は小さいが,実行中ソフトウェアを他のコンピュータに転送・実行するには,移動元ではプログラム中のヒープ領域などのデータを整列化(直列化),移動先ではその逆変換などの計算的負荷も生じる.その他,輻輳しているネットワークには向かない.なお,移動先と移動元のデータ一貫性問題が生じる,本研究は今後,データ一貫性についても対応する予定である.

## 4. 動的適応可能なアーキテクチャの応用

ここでは提案手法を用いて,既存分散ファイルシステムに適応可能な例題について説明する.

### 4.1 計算リリースの状況

計算リリースの状況に応じた動的適応について,想定される事例をあげる.

#### 4.1.1 メディアコンテンツ配信

図6に示したように,コンテンツ配信では配信側となるサーバは,受信側となるクライアント端末,例えばパソコンや,スマートフォンや,タブレット端末に応じて,画面サイズや画像圧縮方式,フレームレートを最適化[6]したのちに配信している.

ただし,一部の端末には最適化したコンテンツを用意できるとは限らず,配信時に最適化していることもあり,その最適化処理はサーバの負荷となることは少なくない.一

方でクライアント端末は高性能化が進んでおり、計算リソースは余っていることが多い。そこでは配信サーバの負荷が高まったときはクライアント側に最適化処理の一部を移行させる。

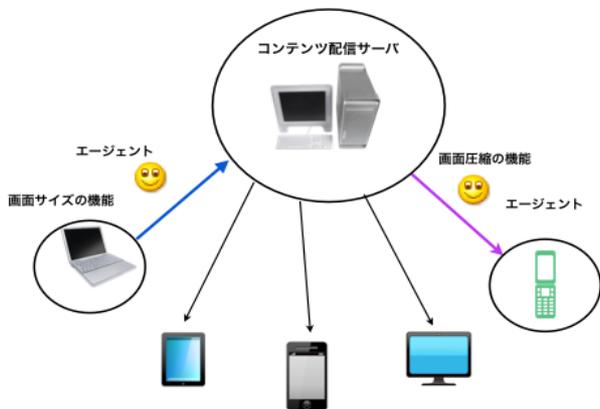


図 6:コンテンツ配信への応用

## 4.2 ネットワークの状況

計算リソースだけでなく、ネットワーク状況に応じた動的適応も有用となる。想定される事例を挙げる。

### 4.2.1 ファイルの圧縮

ネットワークが輻輳している場合、それに対処する方法として、通信ノードは通信回数を減らす方法とデータ量を減らす方法の二つがある。前者に関しては、例えば二つのノードにおいて通信が頻繁に行われる場合、片方のノードの処理の一部をソフトウェア移動により、もう片方のノードに移動させることで、ノード間通信をノード内通信に変えることで輻輳を減らせる。例えば Unix の `grep` コマンドのようにファイルを一行毎にパターンマッチングする場合、対象となるファイルが別のコンピュータにあると、ファイルを一行毎に読み込んでその各行内容を通信する必要があるが、`grep` 相当のプログラムをファイルがあるサーバに移動させれば通信量を減らせる。

次にデータ圧縮と適応性を議論する。一般にデータを送信前に圧縮したのちに送信し、受信側で再びも元のデータに戻すことによりデータ量を減らせるが、本研究のソフトウェア移動による適応性では、図 7 のようにデータ圧縮機能を送信側、圧縮データの解凍機能を受信側に動的に配置することで、動的にデータ圧縮を追加することができる。このとき送信側または受信側には動的に（非）圧縮プログラムを配置できるので、コンテンツ種類や送信または受信側の処理能力に応じて適切な圧縮アルゴリズムを選ぶこともできる。

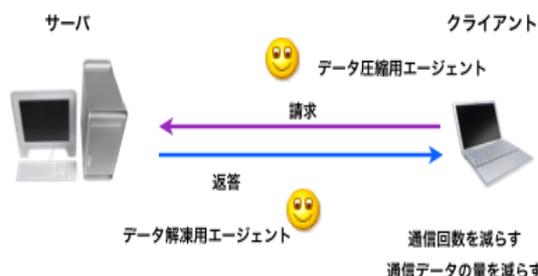
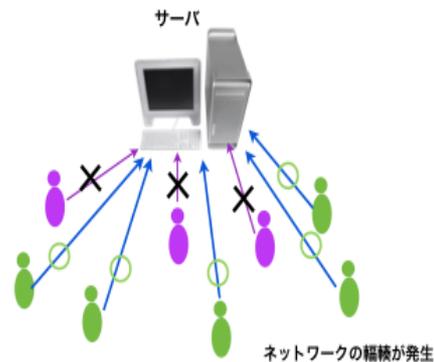


図 7: エージェントコード移動

## 5. おわりに

本稿では、動的適応可能分散ファイルシステムの動的適応条件、ソフトウェア移動原理、ソフトウェア移動の実現手法と提案手法を用いる適応可能事例を概説した。これはモバイルエージェントを用いて、分散システムを構成するコンピュータの機能を変えることにより、例えばクライアント・サーバまたは Peer-to-Peer 方式に変えることができる。

今後は、我々は提案した動的適応方式で既存の分散ファイルシステムに導入して、評価実験を行う予定である。その後、ソフトウェア移動によって生じたデータ一貫性の実現方法も検討・導入する予定である。また本提案手法では、分散ファイルシステムに限らず、データベース[8]やセンサーネットワーク[9]への動的適応にも有効な方法となる。

**謝辞** 提案手法の可用性について議論して頂いたアドバイザー中島震教授、また、貴重な意見をくださった友人たちに感謝致します。

### 参考文献

- 1) Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google File System." (2003). <http://office.microsoft.com/ja-jp/word-help/CH010097020.aspx>
- 2) Borthakur, Dhruba. "HDFS architecture guide." Hadoop Apache Project. <http://hadoop.apache>.

- org/common/docs/current/hdfs\_design.pdf (2008).
- 3) Whitepaper Gluster File System Architecture.  
[ftp://ftp.pku.edu.cn/open/filesystem/GlusterFS/doc/Gluster\\_Architecture.pdf](ftp://ftp.pku.edu.cn/open/filesystem/GlusterFS/doc/Gluster_Architecture.pdf)
  - 4) Atkin, Benjamin, and Kenneth P. Birman. "Network-aware adaptation techniques for mobile file systems." *Network Computing and Applications*, 2006. NCA 2006. Fifth IEEE International Symposium on. IEEE, 2006.
  - 5) Houck, Christopher R., Jeffery A. Joines, and Michael G. Kay. "A genetic algorithm for function optimization: a Matlab implementation." NCSU-IE TR 95.09 (1995).
  - 6) Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. "The akamai network: a platform for high-performance internet applications." *ACM SIGOPS Operating Systems Review* 44.3 (2010): 2-19.
  - 7) Satoh, Ichiro. "A mobile agent-based framework for active networks." *Systems, Man, and Cybernetics*, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on. Vol. 2. IEEE, 1999.
  - 8) Corbett, James C., et al. "Spanner: Google's globally-distributed database." To appear in *Proceedings of OSDI (2012)*: 1.
  - 9) Umezawa, Takeshi, Ichiro Satoh, and Yuichiro Anzai. "A mobile agent-based framework for configurable sensor networks." *Mobile Agents for Telecommunication Applications*. Springer Berlin Heidelberg, 2002. 128-139.
  - 10) Garlan, David, et al. "Rainbow: Architecture-based self-adaptation with reusable infrastructure." *Computer* 37.10 (2004): 46-54.
  - 11) André, Françoise, Erwan Daubert, and Guillaume Gauvrit. "Distribution and self-adaptation of a framework for dynamic adaptation of services." *ICIW 2011, The Sixth International Conference on Internet and Web Applications and Services*. 2011.