大規模サーバにおける部品依存関係の動的抽出方式の提案

奥井 裕[†1] 敷田幹文[†2]

近年，大学や企業などの組織においてコンピュータシステムの利用が必要不可欠である．多重化技術や仮想化技術などによりシステムの大規模・複雑化が進み，内部構成が動的に変化するようになり，システム管理者がシステム全体の構成を把握することが困難になっている．その結果，障害発生時に障害の影響範囲が分からず，システムを利用するクライアントに応じた障害通知を行うことが難しくなっている．本論文では，サーバを構成する物理的・論理的な部品依存関係の動的抽出を行う方式を提案する．この方式によってシステム全体の最新の構成を把握することができ，障害通知に応用することでクライアントに応じた障害通知を可能にする．また，分散して依存関係情報を持つことにより耐故障性の実現および依存関係情報のアクセス制御を可能にする．

Proposal of Dynamic Extraction Method of Part Dependence in Large-scale Servers

Yutaka Okui[†1] and Mikifumi Shikida[†2]

In recent years, it is necessary and indispensable that the organization such as universities and the enterprises uses the computer system. A large scale and the complication of the system have advanced by the clustering technology and the virtualization technology, etc. And, the system administrator's understanding the composition of the entire system became difficult. As a result, when the trouble occurs in the server, the extent of the impact of the trouble cannot be found. And, it is difficult to notify the trouble corresponding to the client that uses the system. In this paper, we propose the method to extract physical and logical part dependence that composes the server dynamically. The latest composition of the entire system can be understood by this method. And, the trouble notification according to the client is enabled by applying this method to the trouble notification. Moreover, the achievement of the breakdown and the access control of dependence information are enabled with dependence information it distributes.

1. はじめに

近年，大学や企業などの組織においてコンピュータシステムの利用が必要不可欠となっており，Webやメール，NASなど多種多様なサービスを数多くのユーザが常時利用できる事が求められている．この要求に応える為，高性能，高可用性を実現する必要が出てきた．

従来は単体のサーバで提供していた複数の機能を，現在のシステムではそれぞれ独立したサーバに分離し，互いに連携することによってサービスを提供している．分離させることにより，ある機能を提供するソフトウェアが利用できるサーバのリソースが増加し，結果として高性能化を実現できる．また，サーバの重要度に応じてクラスタの構築やディスク・NICの多重化が行われ，加えて，近年のサーバの処理能力向上に伴い仮想化技術が導入されている．クラスタの構築や多重化により，あるサーバや内部のディスクやNICが故障した場合でも問題なくサービスを提供し続けることができる．また，サーバ上で動作する仮想マシンを動的に別のサーバに移動するライブマイグレーションを利用することにより，提供するサービスを停止することなくサーバのメンテナンスを行うことが可能となり，高可用性を実現している．このように，新しい構成や技術を導入することにより，前述の要求に応えることが可能になっている．

しかし，高性能化実現のためのサーバ連携によって，あるサービスを提供する為に必要なハードウェア・ソフトウェア部品を容易に探すことができなくなった．また，高可用性の実現のための多重化・クラスタ化に

†1 北陸先端科学技術大学院大学情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
†2 北陸先端科学技術大学院大学情報科学センター
Center for Information Science, Japan Advanced Institute of Science and Technology

より，利用されるディスクやサーバのIPアドレスが動的に変化するようになった．一方，仮想化技術によって物理的構成と論理的構成が一致しないだけでなく，さらに，ライブマイグレーションによりサーバと仮想マシンの対応関係が動的に変化するようになった．その結果，システム全体の構成把握がさらに困難なものとなった．

すなわち，このような大規模かつ複雑なシステムを把握する為に，システム管理者はシステム設計時の資料や各サービスの基本設定などの静的な情報だけでなく，多重化や仮想化技術などによって動的に変化する情報を入手する必要がある．しかし，静的な情報ならまだしも，システムの様々な箇所で動的に変化する情報の入手は難しく，システム全体の最新の構成の把握が困難になっている．システム全体の把握が困難になった結果，例えば，あるサーバやディスクが故障やサービスを提供するソフトウェアの不正終了などにより障害が発生した時，詳細な影響範囲を正確に把握することができず，システムを利用するクライアントに応じた分かりやすい障害通知を行うことができない事態となっている．

そこで，本論文ではクラスタ化されたサーバや多重化されたディスク，様々なサービスやそれらを利用するクライアントなどを含むシステム全体に存在する依存関係を動的に抽出することにより，最新の構成を把握する方式を提案する．また，抽出した依存関係情報を予め決められた領域ごとに分散して保持することにより，アクセス制御を実現する．この方式を利用することにより，静的な情報と動的な情報を半自動的に入手してシステム全体の最新の構成を把握することができる為，システム管理者の負担を軽減することができる．また，提案する方式を障害通知に応用することにより，クライアントに応じた分かりやすい障害通知が可能となる．

以下，2章で関連研究を述べ，3章で先行研究について説明した後，4章で提案方式について述べ，5章でその動作例を示し，6章で考察を行い，7章でまとめを述べる．

2. 関連研究

文献[1)-3)]は国際的な共通情報モデル(CIM)によってシステム全体をモデル化することにより構成を把握し，障害発生時の影響範囲を特定する研究である．これらの研究ではシステムのモデル化を手動で行っている．小規模かつ単純なシステムではモデル化は容易であるが，大規模かつ複雑なシステムをモデル化するのは現実的ではない．また，システムの基本設計情報や各計算機の設定ファイルなどの静的な情報を基にモデル化を行う為，設定ファイルの変更などシステムの最新情報が反映されず実際のシステムとモデルが異なる可能性があり，障害発生時に正しく影響範囲が特定できない場合がある．

表1 クラスの分類例

クラス	該当情報	主属性	属性
Host	ホスト情報	ホスト名	OSの種類 バージョン
File	記憶装置上の データ (ファイル)	ホスト名 パス	
Dir	記憶装置上の データ (ディレクトリ)	ホスト名 パス	
Disk	物理的 記憶装置	ホスト名 ディスク名 (仮想ディスク)	ソフト名 FS
Service	提供する サービス	ホスト名 サービス名 ポート	ソフト名 バージョン

文献[4)]はシステムに存在するサーバやクライアント上に知識を持たせたエージェントを配置し，各エージェントが自律的に連携し障害の検出および通知を行う研究である．エージェントに持たせる知識は管理者が書く必要がある点に注目しなければいけない．小規模かつ単純なシステムならば，エージェント数が少なく連携の為の知識の記述は容易であり，管理者の負担にならない．しかし，大規模かつ複雑なシステムになると，エージェント数が爆発的に増加し，記述する知識も膨大・複雑になり管理者に大きな負担をかけてしまう．

3. サーバの依存関係抽出

本章では，本論文の先行研究として我々が提案をしたサーバの依存関係抽出法[5)]について説明する．

先行研究では，サーバを構成するディスクやファイル，サービスなどの要素を物理的・論理的な部品として扱い，それらの情報をクラスで分類してオブジェクトという独自の単位で表現する．クラスの分類例を表1に示す．オブジェクトを用いることにより，ディスクや仮想ディスク等の下位レイヤからアプリケーションやサービス等の上位レイヤを同じ概念で扱うことができるようになる．また，オブジェクト間には依存関係が存在する．例えば，ファイルオブジェクトはファイルの実体が保存されているディスクオブジェクトやファイル共有の為のサービスオブジェクトに依存していると言える．オブジェクトを用いているので，サーバだけでなくサーバ間，サーバ・クライアント間に存在する依存関係を扱うことができる．

オブジェクトの依存関係情報は，サーバ上に存在する設定ファイルや管理コマンドの実行結果などの情報を入手して解析したり，管理者が入力することにより抽出を行う．オブジェクトの種類ごとにどの情報を解

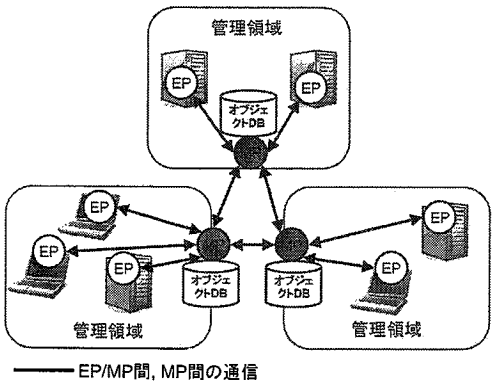

図 1　提案方式の構成

析するかは予め決まっている．

最初のオブジェクトは管理者によって与えられる．与えられたオブジェクトの依存関係情報を抽出することにより，依存先オブジェクトを新たに発見することができる．発見したオブジェクトに対して再び依存関係情報を抽出する処理を行うことにより，連鎖的にシステム全体の構成を半自動的に把握することができる．

しかし，先行研究ではシステムの構成に変化が少ない環境を対象としており，一度抽出した依存関係情報をある程度の期間使い続けることを前提としている．その為，システムの構成が変化した時に管理者が依存関係情報の再抽出を行わなければいけない．故に，定期的に依存関係情報が変化する環境では，抽出した依存関係情報とシステムの構成が異なるという問題が生じてしまう．

4. 依存関係の動的抽出方式

本章では，前章で述べた先行研究の問題点を解決すべく，システム全体の依存関係情報を動的に抽出する方式について説明する．

4.1 提案方式の概要と構成

先行研究では，システムに存在するサーバに問い合わせて情報を得ることにより，依存関係情報の抽出を行っている．サーバの構成が変更になりシステムの依存関係情報が変化しても再抽出を行うまで構成の変化を知ることはできない．その為，定期的に依存関係情報の再抽出を行うか，管理者が明示的に再抽出を指示するしかなかった．前者は依存関係に変化がない時も再抽出を行う為にコストがかかり，後者は管理者が再抽出の指示を忘れると最新の構成と依存関係情報から推測される構成が異なる事態が生じる．

そこで，本論文では，大学や企業などの組織におけるコンピュータシステムに存在する依存関係情報を動的に抽出し，システムの最新の構成を把握する方式を提案する．提案方式の構成を図 1 に示す．

提案方式では，組織のコンピュータシステムを研究科，研究室，部署，事業所，データセンターなどの単位で分割する．この分割されたシステムが存在する領域を管理領域と呼ぶ．管理領域には複数のサーバ，クライアントなどの計算機が存在しており，すべての計算機上で依存関係抽出プロセス (EP: Extract Process) を動作させる．EP によって各計算機内部の依存関係の動的抽出が行われる．また，各管理領域に管理サーバを設置し，依存関係管理プロセス (MP: Management Process) を動作させる．MP は自身が所属する管理領域で動作する EP の管理を行う．管理領域毎に EP が動的抽出した依存関係情報を MP が持つオブジェクト DB で管理する事により，システム全体の最新の構成を把握することができる．次節より，EP と MP について述べる．

4.2 依存関係抽出プロセス

EP はサーバやクライアントなどの計算機上で動作するプログラムである．EP が動作する計算機内部に存在する依存関係情報の動的抽出を行う．MP に抽出した情報を送信する一方，MP からオブジェクト抽出要求の受信を行う．EP は所属する管理領域の MP のみと通信を行い，他の管理領域の MP, 管理領域内外の EP とは通信を行わない．

また，依存関係情報の抽出は EP が動作する計算機内部だけあり，他の計算機のオブジェクトに対して依存関係を持っている場合は，それ以上抽出を行うことができない．EP は 3 つの要素から構成されている．

1. 未抽出オブジェクトキュー
2. 抽出部
3. 監視部

未抽出オブジェクトキューには，依存関係情報の抽出が行われていないオブジェクトや依存先が変化した可能性があるオブジェクトが格納される．

抽出部では，オブジェクトキューからオブジェクトを 1 つデキューし，依存関係情報の抽出を行う．抽出は先行研究[5] の方式と同様に，設定ファイルや管理コマンドの実行結果などの情報を解析することにより行う．この操作により得た依存関係情報や依存先特定に使用した情報などを MP に送信する．この作業を繰り返すことにより依存関係の抽出を行うことができる．

監視部では，EP が動作する計算機内部に存在する依存関係の変化および動作プロセスの監視を行う．抽出部において，依存関係抽出時に依存先特定に使用した情報を MP に送信していた．この依存先特定情報を監視することにより，依存関係の変化を検出することができる．例えば，設定ファイル F を解析することにより，オブジェクト A がオブジェクト B に依存している事が明らかになったとする．オブジェクト A, B 間の依存関係は設定ファイル F に依存している．設定ファイル F のタイムスタンプや内容が変化した場合，オブジェクト A の依存先が変化した可能性があると

```
変数:
obj: オブジェクトを格納
obj_que: 未抽出オブジェクトキュー
obj_db: オブジェクト DB (MP 上に存在)
関数:
enq(obj){obj_que に obj をエンキュー }
deq(){obj_que からオブジェクトをデキュー }
create_obj(info){ 引数の情報を基にオブジェクトを生成 }
extract(obj){
 obj の種類，主属性，属性を基に依存関係情報抽出
 obj_db に依存関係情報を格納
 if obj に依存先が存在 {
  enq(create_obj(依存先情報))
 }
}
recv() {MP からオブジェクトの抽出要求を受信 }
check_dependence() { 依存関係の変化を監視 }
main() {
 connect(所属する管理領域の MP)
 thread_start(recv)
 thread_start(check_dependence)
 loop {
  while ((obj=deq_obj()) != なし) {
   extract(obj)
  }
  obj_que が空の間ブロック
 }
}
```

図 2 EP の動作アルゴリズム

分かる．よって，オブジェクト A の依存関係情報を再抽出することにより，依存関係の変化を追うことができる．また，EP が動作する計算機のプロセスを監視することにより，新しいオブジェクトを見つけることができる．

3 つの要素の動作をアルゴリズムで表現したものを図 2 に示す．なお，図中の obj_que，extract()，check_dependence() はそれぞれ，未抽出オブジェクトキュー，抽出部，監視部に対応している．

4.3 依存関係管理プロセス

MP は各管理領域の管理サーバ上で動作するプログラムである．管理サーバが所属する管理領域に存在する EP の管理を行う．EP と通信を行い，オブジェクトと依存関係情報の受信，オブジェクトの抽出要求の送信等を行う．管理領域外の EP とは通信は行わない．また，他の MP と通信を行いオブジェクト抽出要求などを送受信する．MP は以下の要素から構成されている．

1. オブジェクト DB
2. 抽出要求部

```
変数:
obj: オブジェクトを保持
obj_db: オブジェクト DB
accept() {EP/MP からの接続を待ち受ける }
check_db {
 loop {
  if obj_db に未抽出オブジェクトあり
   if 管理領域内で抽出可能
    該当 EP へ抽出要求を送信
   else
    他 MP へ抽出要求を送信
 }
}
関数:
recv_ep() {
 loop {
  EP からオブジェクト，依存関係情報を受信するまでブロック
  受信したオブジェクト，依存関係情報を obj_db に格納
 }
}
recv_mp() {
 while(obj = recv_mp_obj()) {
  if obj = 管理領域内で依存先抽出可能
   該当 EP に obj の抽出要求を送信
  else
   他 MP に obj の抽出要求を送信
 }
}
main() {
 thread_start(accept)
 thread_start(check_db)
 thread_start(recv_ep)
 thread_start(recv_mp)
 全スレッドが終了するまでブロック
}
```

図 3 MP の動作アルゴリズム

3. 通信部
4. アクセス制御部
5. 端末部

オブジェクト DB には，MP の管理領域におけるオブジェクトと依存関係情報を格納する．1 つのレコードにオブジェクトの種類，主属性，属性，依存先オブジェクト，依存先特定情報を格納する．オブジェクトの依存関係情報の抽出が行われていない場合，依存先オブジェクト，依存先特定情報は空欄になる．

抽出要求部では，オブジェクト DB の監視を行うことで依存関係情報の抽出が行われていないオブジェクトを見つけ，EP もしくは MP に対してオブジェクトの抽出要求を送信する．送信先はオブジェクトの依存

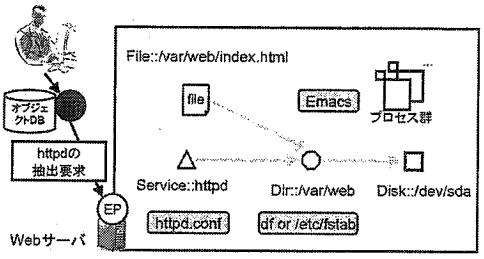

図 4　依存関係の動的抽出例

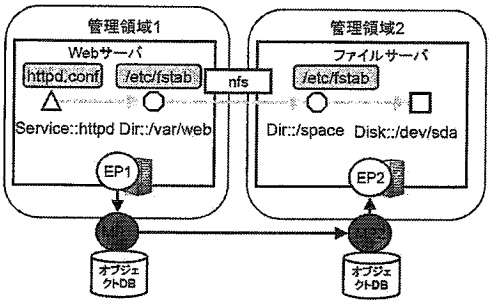

図 5　管理領域を超えた依存関係の動的抽出

関係情報の抽出が管理領域内で行うことができるかどうかによって決まる．

通信部では，EP と通信を行いオブジェクトや依存関係情報の受信を行う．また，MP と通信を行いオブジェクトの抽出要求を送受信する．

アクセス制御部では，管理者の権限などに基づき，MP が保持するオブジェクト DB へのアクセス制御を行う．

端末部では，オブジェクトや依存関係情報の入力および閲覧を行う．この部を通じてオブジェクト DB にオブジェクトや依存関係情報の入力を行う．

5 つの要素の一部をアルゴリズムで表現したものを図 3 に示す．

5. 依存関係の動的抽出例

本章では，オブジェクト間の依存関係の動的抽出例について述べる．

5.1 EP 内部での動的抽出例

ここでは，MP からオブジェクトと抽出要求を受け取った EP の動作について，図 4 を用いて説明を行う．

まず，MP が EP に対して httpd オブジェクトの依存関係の抽出要求を送信する．この要求を受け取った EP は，抽出アルゴリズムに従い httpd オブジェクトの依存関係情報の抽出を行う．オブジェクトの種類，主属性，属性によってどの情報を解析するかは予め決まっている．今回の例では，httpd.conf という設定ファイルを解析することにより，依存関係の抽出を行うことができる．解析の結果，httpd オブジェクトは/var/web というディレクトリを参照しており，依存しているということが分かる．抽出後，httpd オブジェクトの依存先特定情報として httpd.conf を記録する．

その後，/var/web を表すディレクトリオブジェクトを作成し，同じく依存先の抽出を行う．df コマンドの実行結果もしくは/etc/fstab を解析することにより，/dev/sda というディスクに依存していることが分かる．先ほどと同様に，df コマンドの実行結果や/etc/fstab を依存先特定情報として記録する．今回の例では，これ以上依存関係情報の抽出を行うことができないので抽出を終える．

ここで，Web サーバ上で Emacs を起動して/var/web/index.html というファイルをオープンしたとする．EP はプロセスを監視しているので，どのプロセスがどのファイルをオープンしたか把握することができる．この場合，Emacs が開いたファイルはディレクトリオブジェクト/var/web に依存していることが分かるので，新しくファイルオブジェクト/var/web/index.html を作り，依存関係情報を抽出する．このようにして，依存関係の動的抽出を行うことができる．httpd.conf が更新されて httpd オブジェクトが別のディレクトリを参照した場合や/var/web が違うディスクにマウントされた時でも，動的に依存関係情報を抽出することができる．

5.2 管理領域を超えた依存関係の動的抽出例

ここでは，管理領域を超えたオブジェクト間の依存関係情報の動的抽出について図 5 を用いて説明を行う．

図 5 は 2 つの管理領域が存在し，一方の管理領域に存在する Web サーバが，もう一方の管理領域に存在するファイルサーバのスペースを nfs マウントしている様子を表している．

ここで，EP1 に httpd オブジェクトの依存関係の抽出要求を出すと EP1 は Web サーバ内部の依存関係情報の抽出を行う．そして，ディレクトリ/var/web がファイルサーバの/space を nfs マウントしている事を把握する．しかし，EP1 は自身が動作する計算機内部に存在する依存関係情報しか抽出を行わない．

そこで，EP1 は抽出した依存関係情報と共に未抽出であるファイルサーバのオブジェクト情報を MP1 に送信する．未抽出のオブジェクトを受け取った MP1 は自身の管理領域にファイルサーバが存在するか確認する．今回の例では，MP1 の管理領域にファイルサーバは存在しないので，MP2 にファイルサーバのオブジェクトの抽出要求を送信する．抽出要求を受信した MP2 は管理領域にファイルサーバが存在するか確認する．その後，EP2 にオブジェクトの抽出要求を出すことにより，内部で依存関係が抽出される．

このように，EP と MP が連携することにより，管理領域を超えた依存関係情報の動的抽出を行うことができる．

6. 考　　察

この章では，提案方式を 2 つの観点から考察を行う．

6.1 障害通知への応用

従来は大規模・複雑なシステムの最新の構成の把握が困難であった為，システム内部のサーバに何らかの障害が発生した場合に詳細な影響範囲が分からなかった．

その為，サーバの障害の影響を受ける可能性がある全クライアントに対して通知を行う必要があった．例えば，障害発生時にファイルサーバ上のファイルをオープンしている 20 人のクライアントに影響が及ぶが，その 20 人だけでなく使う可能性がある他の 1000 人に通知するということである．

提案方式により動的に抽出した依存関係情報を障害通知へ利用する事により，サーバで発生した障害の影響がどのクライアントのどのアプリケーションにまで影響が及ぶか把握することができる．そして，クライアントに対して「ファイルサーバで障害が発生し，あなたが word で開いている memo.doc に影響が出ます.」などのユーザにとって分かりやすい障害通知を実現することができる．

6.2 依存関係情報の分散管理

従来研究では，抽出した依存関係情報を 1 つのシステムで集中管理しており，システムがダウンすると依存関係情報を利用することができないという問題点や，抽出したシステム全体の依存関係情報は管理者の権限に関わらず見ることができるという問題点があった．

提案方式では，各 MP が管理する管理領域ごとに依存関係情報を持ち分散管理を行うことによって耐故障性を実現している．また，MP が持つ依存関係情報を他の MP に見せるかどうかを選択することができるので，管理者の権限に応じた依存関係情報へのアクセス制御を実現することができる．

7. おわりに

本論文では，大規模かつ複雑なシステムに存在する依存関係を動的に抽出する方式を提案した．

本方式を用いることにより，クラスタ構成のサーバやディスクの多重化，仮想化技術などにより動的に変化するシステム全体の最新の構成を把握することができるようになった．そして，把握した構成を障害通知に応用することでクライアントに分かりやすい障害通知を実現することができ，また，抽出した依存関係情報を分散して持つことにより，依存関係情報へのアクセス制御が可能となった．

従来は管理者がシステムに関する静的な情報だけでなく動的に変化する情報を入手した上で全体の構成を把握する必要があった．しかし，本方式を用いることにより，情報の入手および構成の把握が半自動化される為，管理者にかける負担を軽減することができる．これにより，今まで以上に大規模・複雑なシステムを構築し，少人数の管理者や高度なスキルを持たない管理者でも運用することが可能となる．

今後は，提案する方式を実装したシステムの試作を行い，実運用されているシステム上で有効性の検証を行う予定である．また，現在はある組織内におけるシステムを対象としているが，組織を超えて依存関係の抽出を行う発展が考えられる．

参 考 文 献

1) 細川武彦，森　信胤，虎渡昌史：運用管理システムにおける監視対象モデルを用いた障害影響範囲分析の課題と解決策，情報処理学会 IOT 研究報告，Vol.2005, No.33, pp.277–282 (2005).
2) 酒井将人，石川　裕：CIM を用いた障害検知システム，情報処理学会 OS 研究報告，Vol.2006, No.86, pp.125–132 (2006).
3) 田中　智，宇和田弘美：CIM 知識ベースを活用したシステム運用管理のインテリジェント化，NRI 技術創発，Vol.3, No.3, pp.78–91 (2004).
4) 今野　将，吉村智志，羽鳥秀明，岩谷幸雄，阿部　亨，木下哲男：能動化された状態情報に基づくネットワーク管理支援方式，情報処理学会論文誌，Vol.46, No.2, pp.493–505 (2005).
5) 森　一，敷田幹文：サーバの依存関係を考慮したシステム構成管理の支援法，情報処理学会論文誌，Vol.46, No.4, pp.940–948 (2005).