大規模サーバ間の部品依存関係に基づく障害通知方式の提案

敷田幹文[†1]

近年，情報ネットワークは我々の社会生活に必要不可欠なインフラとなっており，サービスを提供するサーバにはきわめて高い信頼性が要求されている．ハードウェアや基本ソフトウェアの進歩によって，障害を回避する大規模・高信頼性サーバが構築可能となったが，発生した障害の復旧には管理者の作業が不可欠である．従来の運用管理ソフトウェアにより，各種サーバの障害情報を収集し，管理者へ通知を行うことが可能であるが，大規模なシステムでは膨大な量の情報が通知される．各管理者は一部のシステムのみを担当している場合でも，担当システムに関連のある他システムの情報を必要とすることが多い．つまり，複雑に依存し合う大規模システムでは情報のフィルタリングが困難であり，各自が担当するシステムに関する重要な通知の埋没化が問題となっている．本論文では，サーバの各部や様々なサービスの依存関係に注目し，個々の障害に応じて適切な管理者のみへ適切な情報のみを通知する障害情報通知方式を提案する．たとえば，ディスクドライブの故障時には，その管理者のみでなく，コンテンツを利用する Web アプリケーション管理者に，その依存の程度に合わせて通知をすることが可能となる．また，この方式を適用した例に基づき，本方式を用いた大規模サーバ群の障害管理支援の有効性に関する議論を行う．

A Method of Event Notice Based on Dependencies among Components of Large-scale Servers

Mikifumi Shikida[†1]

In some large organizations such as a large enterprise or a university, large-scale servers are designed to organize large information systems. Reliability of the large-scale servers is key factor of managing those systems. In these years, we have large-scale high availability servers. Administrators' works are essential to the restoration of trouble. It is possible that event notice is done to the administrators by integrated management tools. They receive enormous quantity of notices for large-scale systems. Because dependencies of large-scale systems are complicated, filtering of notices is difficult. This paper proposes new event notice method based on dependencies among components of servers and services. The system based on this method sends suitable notice to suitable administrators. We illustrate the method with a simulation on an example system and discuss its usefulness.

1. はじめに

近年，インターネットが広く普及し，情報ネットワークは我々の社会生活に必要不可欠なインフラとなっている．そのような状況下で，サービスを提供するサーバにはきわめて高い信頼性が要求されている．ハードウェアや基本ソフトウェアの進歩によって，障害を回避する大規模・高信頼性サーバが構築可能となった[1),2)]が，発生した障害の復旧には，故障部品の交換やソフトウェアの設定変更など，管理者の作業が不可欠である．

そこで，発生した障害をいかにして検出し，管理者に的確に通知するかという問題が研究されており，様々な障害管理ソフトウェア[3)–6)]が実用化されている．それらの運用管理ツールを利用することにより，組織内の様々なシステムの各部に関する情報が運用管理サーバに収集され，サーバ内で情報の内容を参照し，あらかじめ設定された条件に従って適切な担当者へ適切なメディアで通知を行う．

しかしながら，大規模なサーバを含むシステムでは膨大な量の情報が通知される．各管理者は一部のシステムのみを担当していたとしても，担当システムに関連のある他システムの情報を必要とすることも多い．複雑に依存し合う大規模サーバでは，情報のフィルタリングが困難であり，各自が担当するシステムに関する重要な通知の埋没化が問題[7)]なっている．

本論文では，大規模サーバの各部や様々なサービス

†1 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

の依存関係に注目し，個々の障害に応じて適切な管理者のみへ適切な情報のみを通知する障害情報通知方式を提案する．これにより，たとえば，ディスクドライブの故障時には，ストレージシステム管理者のみでなく，そのディスクにコンテンツを格納しているデータベース管理者や，そのデータベースを利用する Web アプリケーション管理者にも，それぞれの依存程度に応じた注意喚起の通知をすることが可能となる．また，本論文では，この方式をいくつかのシステムにおける障害に適用した例に基づき，本方式を用いた大規模サーバ群の障害管理支援の有効性に関する議論を行う．

以下，2 章で従来の障害管理について述べ，3 章では本論文の先行研究であるサーバの依存関係抽出法について述べる．4 章では本論文で我々が提案するサーバの依存関係を考慮する障害管理方式について説明し，5 章では本論文の方式の有効性に関する議論を行う．

2. 従来の障害管理

本章では，障害管理に関連する従来研究および実用化されているシステムについて述べる．

2.1 障害検出

近年，運用管理の大規模集中化が進んでおり，多数の機器に関して集中管理する方式が提案・実用化されている．特にネットワーク機器に関しては，組織内の各所に大量に分散設置され，各機器が持つ状態も複雑でないことから，SNMP を用いた統合管理ソフトウェア[8),9)] が広く普及している．

このような集中管理方式には，監視のためのネットワークトラフィックの増加という問題もあるが，階層化を行ったり，エージェントや分散オブジェクト技術を用いたりする研究[10)–12)] も行われている．

さらに近年は，ネットワーク機器に加えてサーバの監視も同様に行うソフトウェア[13),14)] が増えている．しかし，サーバはネットワーク機器に比べて状態が複雑であり，詳細な情報を取得しなければ検出された障害の通知から状況の概略が理解できない[15)]．

2.2 障害通知の問題

統合管理ソフトウェアによって集中監視を行い，障害が検出されると，その状況を管理者に通知する必要がある．

なお，本論文での通知とは，電子メールやページャで即座に送信したり，警告ランプを点灯したりするだけでなく，障害情報を蓄積して管理者の画面で一覧が参照可能にする方法など，広い意味で用いている．

ネットワーク機器の場合は，監視対象のほとんどは物理層からネットワーク層までのレイヤであり，各機器の動作の類似性が高く，相互接続が目的であることからも，同一の管理者が組織内の大部分を担当する傾向にある．一方，サーバの場合は，ディスクドライブなどのハードウェアから Web 上のアプリケーションソフトウェアまで数多くのレイヤに分かれており，またメーカ間の差異が大きいため，ネットワーク機器の管理よりも 1 人の管理者による全体把握が困難である．従来の規模であれば少人数で把握していることによる利点が大きかったが，近年は大企業や大学のシステムが急速に大規模化しており，管理者の分業が進む傾向にある．

しかし，サーバの各レイヤは関係があり，依存し合うことでサービスが提供されている．この依存関係は同一サーバの複数レイヤのみでなく，複数のサーバ間にも存在している．たとえば，Web 上のアプリケーションが他のサーバ上のデータベースにアクセスしたり，データベースサーバのデータが巨大なストレージサーバの一部に格納されたりしていることもある．

このような管理体制で，従来の統合管理ソフトウェアによる障害通知を利用する際に，担当する部分システムに関する通知のみでなく，依存関係のある他システムの通知も受け取るべきであるが，複雑な依存関係に合わせた通知設定を行うことは困難である．またすべてに関して詳細情報を受け取ると最も重要な担当システムの情報が埋没化する．

したがって，多大な労力を要するシステム管理業務の遂行のためには複数人で作業を分担すべきであるが，分担した場合の弊害があるために分担が進まず，現状では一部の管理者に負荷が集中するという問題をかかえた組織も多い．

3. サーバの依存関係抽出法

本章では，次章で述べる方式の先行研究として我々がすでに提案しているサーバの依存関係抽出法[16)] について説明する．

近年，大規模なシステムを運用する組織においては，複雑なサーバ群を複数人の部署で集中管理する形態が増加している．このような組織のサーバ群は，ストレージサーバ，データベースサーバ，Web サーバ，各種アプリケーションサーバなどが，互いに依存関係を持つことが多い．その場合，一部のみを担当する管理者にとっては，全体システムの把握は困難であり，担当サーバ/サービスと依存関係があっても，さらにその先の依存関係先まで把握できない．

文献 16) で提案した方式は，各種サーバや周辺機器の設定情報を収集し，またシステム設計者が情報を補

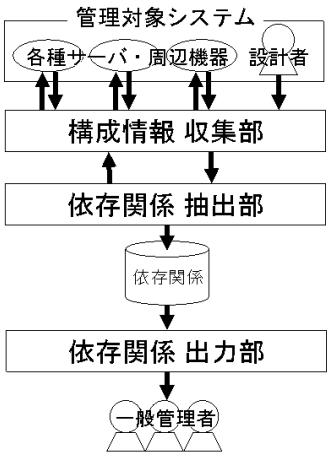

図 1 依存関係に基づく構成管理支援システムの概要[16)]
Fig. 1 The outline of configuration management system based on dependencies.

足する．それらの情報をもとに各部の依存関係をすべて抽出し，各管理者がシステム全体の構成を把握することの支援を行うものである．この方式を用いた構成管理支援システムの概要を**図 1** に示す．

たとえば，ディスクドライブなどの物理デバイス，RAID ディスクなどの仮想デバイスや HTTP プロトコルサーバなどの上位レイヤにおけるサービスなど，様々なオブジェクトに関して依存関係の調査を行う．調査には，それぞれのオブジェクトに関係する設定ファイル，管理コマンドの出力結果の解析などを行うため，システムの種類ごとに異なる処理が必要となる．また，アプリケーションのインストール場所など，場合によっては設計を行った管理者が与える情報をもとに行う．この調査アルゴリズムの詳細は文献 16) で述べている．

本方式を用いた構成管理支援の試作システムの出力結果を**図 2** に示す．これは，我々の組織で実際に稼動している高可用性クラスタストレージサーバの構成情報の一部を入力として与え，NFS クライアント上のディレクトリを指定して依存している全ディスクの一覧と，その一覧の 1 つの詳細を出力させた例である．

ストレージに関する依存関係のみでなく，物理デバイスからアプリケーションまで広範囲にわたる間接的な依存関係の検索も可能である．たとえば，**図 3** に示すようなオブジェクトの依存関係が把握できていると，ストレージサーバの管理者は，自分が管理するどのディスクドライブ内のデータがどのような重要サービスに利用されているのか知ることができる．また，Web 上のアプリケーション管理者も，自分が管理するアプリケーションのファイルがどのサーバのどの物理

```
ASM2:search>>> depend -v %DIR%is14e1%/home/i2008 %DISK%*%*
       CLASS:DIR HOST:is14e1 PATH:/home/i2008
100:3  CLASS:DISK HOST:fs90 DISKname:/dev/dsk/c3t0d1s2
66:7   CLASS:DISK HOST:fs8-dct10 DISKname:RG-01
76:8   CLASS:DISK HOST:fs8-dct10 DISKname:/dev/dsk/c1t0d1s2
70:14  CLASS:DISK HOST:fs8-dct10 DISKname:/dev/dsk/c1t0d7s2

ASM2:search>>> depend -vod 76:8
    CLASS:DIR HOST:is14e1 PATH:/home/i2008
<-- CLASS:DIR HOST:fs90 PATH:/home/fs5001 (nfs_mount)
<-- CLASS:DISK HOST:fs90 DISKname:/dev/dsk/c3t0d1s2 (ufs_mount)
<-- CLASS:DISK HOST:fs8-dct10 DISKname:RG-01 (SAN FibreChannel)
<-- CLASS:DISK HOST:fs8-dct10 DISKname:/dev/dsk/c1t0d1s2 (RAID5)
```

図 2 依存関係の出力例
Fig. 2 The example of output of dependencies.

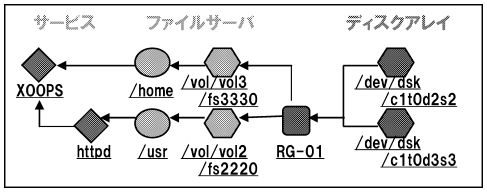

図 3 オブジェクトの依存関係例
Fig. 3 An example of dependencies among objects in servers.

資源を利用しているか容易に把握できる．すなわち，この方式によってレイヤを越えた依存関係の把握をも支援可能である．近年はストレージの仮想化も進んでおり，このような依存関係は通常把握が困難であった．

4. サーバの依存関係に基づく障害情報通知法

本章では，サーバや周辺機器の各部の依存関係に基づいて障害情報の通知を行う方式について述べる．

4.1 方式の概要

本方式は，3 章で述べた方式を基盤とし，これを障害情報の通知に応用するものである．サーバや周辺機器の各部の依存関係が把握できていれば，ある部分で発生した障害の影響が他のどの部分に波及するかを算出することができる．その算出結果に従って，各部分の担当管理者へ通知を行う．またその際，影響の程度に合わせた内容で通知する．

なお，本論文での障害とは，ハードウェアの故障やソフトウェア上のエラーが発生した場合のみでなく，異常かどうかにかかわらず，何らかの状態変化が発生した場合をすべて含むものとする．たとえば，夜間にデータのバックアップを行い，そのジョブの終了時にレポート通知を行うことがある．エラーと断定できるような事象が何も発生していなくても，たとえば，バックアップに通常よりも著しく長い時間を要した場合には確認が必要であるという理由で，担当する管理者はレポートに目を通す．このような通知に関しても

本方式の対象とする．

4.2 障害による影響の算出法

ここでは，システムのある部分において障害が発生した場合に，他の部分にどの程度の影響が及んでおり，その部分の管理者にどのように通知すべきか，その程度を算出する方法を述べる．なお，サーバ管理の現場ではシステム運用上の特殊事情があって変則的な設定を行いたい場合がありうるために，自由度の高い算出法としているが，通常の運用では次節で述べるように簡単化して利用する．

まず関連する用語を以下のように定義する．

オブジェクト サーバや周辺機器を構成する要素をオブジェクトと呼ぶ．ディスクドライブやテープドライブなどの物理的な要素，および仮想ディスク，ファイルシステム，ディレクトリ，FTP や HTTP などのサービスなどの論理的要素もすべてオブジェクトとして扱う．$n+1$ 個のオブジェクトが存在しているとき，これらを $V_0, \ldots, V_n$ と表記する．

依存関係 オブジェクト V_i がオブジェクト V_j に依存しているとき，**依存関係**があるという．また，この関係に関しては V_j は**依存元オブジェクト**，V_i は**依存先オブジェクト**と呼ぶ．依存関係は 1 対 1 とは限らず，たとえば 1 つの仮想ディスクが 2 つのディスクドライブのミラーとなっていることもあるが，ここでは簡単化のため，仮想ディスクのオブジェクトが 2 つのディスクドライブのオブジェクトそれぞれと独立に依存関係が存在しているとする．

依存度 オブジェクト V_i が V_j に依存しているとき，その程度を**依存度**と呼び，$D_{i,j}$ と表記する．オブジェクト V_i が動作するために V_j が必須であれば $D_{i,j}=1$ とする．たとえば，ミラー構成の仮想ディスクのように，1 つのディスクドライブが故障しても動作可能であれば $D_{i,j}<1$ と考える．依存度は 0 より大きく 1 以下の値であるが，各依存関係における依存度の実際の値は，たとえばホットスペアディスクの有無などによって管理者が個別に設定する．また，個別の依存度とは別に，各依存関係が障害の影響をどの程度伝えるべきかをシステム全体に関して規定する値を**伝播指数**と呼び，p と表記する．

重要度 各サーバは何らかの必要性があって設置し運用されているはずであるが，その重要性には差異がある．たとえば，組織外へ公開している HTTP サービスや管理者のみが試験的に利用しているサービスなど，1 台のサーバ上でもオブジェクトによって重要性は異なる．その程度をそのオブジェクトの**重要度**と呼び，オブジェクト V_i の重要度を I_i（ただし $1 \geq I_i \geq 0$）と表記する．

障害の重大度 あるオブジェクトで何らかの状態変化が発生した場合，それがそのオブジェクトの動作に致命的な故障であるのか，異常とはいえないが状態を通知する必要があるのか，その程度には差がある．このような値をその障害の**重大度**と呼び，S と表記する．多くのイベント管理ソフトウェアでは，重大度を 4 から 8 程度のレベルに分けているが，ここでは $1 \geq S > 0$ と一般化する．

影響度 あるオブジェクトで発生した障害が，別のオブジェクト V_i の動作に影響を及ぼすとき，その影響の程度を**影響度**と呼ぶ．影響度は E_i と表記し，$1 \geq E_i \geq 0$ とする．

通知レベル 障害が発生し，あるオブジェクト V_i の影響度が $E_i > 0$ となった場合はそのオブジェクトを担当する管理者に通知すべきであるが，どの程度の事象として通知すべきかを表す値を**通知レベル**と呼ぶ．通知レベルは N_i と表記し，$1 \geq N_i \geq 0$ とする．従来の統合管理ソフトウェアでは，通知レベルは障害の重大度と同じ（すなわち，$N_i = S$）であり，その通知を行うかどうかや通知メディアを選択するのみのものが多い．

ここで，$n+1$ 個のオブジェクトが存在し，V_i が V_{i-1} に依存している（$n \geq i > 0$）とし，これら以外の依存関係がないと仮定する．オブジェクト V_0 で重大度 S の障害が発生した場合，各オブジェクトの影響度を次のように定義する．

$$E_i = \begin{cases} S & (i=0) \\ E_{i-1}(D_{i,i-1})^p & (n \geq i > 0) \end{cases} \quad (1)$$

依存関係に分岐があった場合でも，すべての依存先オブジェクトの影響度を式 (1) と同様に算出する．

ここで，式 (1) の漸化式を解くと式 (2) のようになる．すなわち，この式は，オブジェクト V_n への障害の影響度は，障害発生箇所 V_0 の重大度 S を途中の各依存度に従って徐々に減らした値であることを意味している．

$$E_n = S\prod_{i=1}^{n}(D_{i,i-1})^p \quad (2)$$

影響度の算出例を述べる．単純に 100%依存する関係であれば $D_{i,i-1}=1$ なので，$E_n = S{\times}1{\times}1{\times}\cdots{\times}1 = S$ となるが，RAID などで冗長化した箇所がたとえば $D=0.5$ ならば，$E_n = S{\times}1{\times}1{\times}0.5{\times}1{\times}\cdots{\times}1$ となる．また，伝播指数 p は全体の依存度を一律に変更するものであり，たとえば極端な例として，「RAID によっ

て影響が減る効果を 2 倍にする」場合には，$p = 2$ とすることで $E_n = S \times 1 \times 1 \times 0.5^2 \times 1 \times \cdots \times 1 = 0.5^2 S$ となる．したがって，p を 1 より大きな値にすると影響が減る程度が強まり，1 より小さな値にすると影響が減る程度が弱まって，より遠くのオブジェクトまで大きな影響が及ぶことになる．

以上のように，障害が発生したオブジェクトから依存関係に従って，間接的に依存しているすべてのオブジェクトの影響度を決定することができる．また，障害オブジェクトから到達できないオブジェクトの影響度は 0 であるとする．なお，本論文では依存関係には循環がない，すなわち依存関係は非循環有向グラフ（DAG）であると仮定している．

以上のようにして求めた影響度をもとに，各オブジェクトの管理者への通知レベルは次の式で定める．

$$N_i = E_i I_i \quad (n \geq i \geq 0) \tag{3}$$

4.3 通常運用時の値の決め方

前節で述べた影響度や通知レベルの算出法は，様々なポリシの組織に適用できるように一般性を持たせた式で説明したが，実際には，多くのシステム管理の現場でこのような自由度は不要である．それぞれの値の決め方は次のように考える．

依存度 $D_{i,j}$ 3 章で述べた方式により依存関係があると判断されているオブジェクト間の依存の程度であるので，すべての依存度が $D_{i,j} = 1$ と考える．ただし，RAID 構成の仮想ディスクのように，動作に 100%必要ではない場合は 0.5 など 1 より小さな値を設定する．その値の大きさは RAID の構成や管理のポリシによるが，このような箇所は全オブジェクトのごく一部である．

伝播指数 p 依存度は各依存関係での個別の設定であるが，大規模なシステムを複数人で管理している場合に，算出される影響度の大きさを全体的に調整したいことがありうる．そのような配慮が不要であれば通常は $p = 1$ とする．式 (2) において，$D_{i,j}$ の p 乗を用いており，依存度 $D_{i,j} = 1$ の場合には p の影響はない．すなわち，上記の RAID 構成などの理由で依存度を 1 より小さく設定した箇所に関して，全体的に変化させる場合のパラメータである．

重要度 I_i 従来の障害通知ツールでは，サーバの各部位に対して障害発生時に通知を行うかどうかのみを設定することも多い．そのような単純な運用を行うのであれば，通知が必要なオブジェクトでは $I_i = 1$ とし，通知が不要なオブジェクトでは $I_i = 0$ とする．影響が大きい場合のみ通知が必要という運用を行う場合には中間の値を設定することで式 (3) に従って通知の加減が可能である．

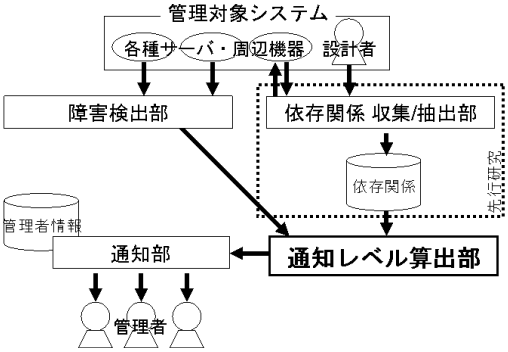

図 4 依存関係に基づく障害通知支援システムの概要
Fig. 4 The outline of event notice system based on dependencies.

重大度 S 従来の syslog や統合運用管理ソフトウェアでは，重大度を 4 から 8 程度のレベルに分けている．このレベルを 0 から 1 の範囲に単純にマッピングして入力として与える．

以上の理由から，単純な運用を考えるのであれば $p = 1$ とし，また RAID などがない限りすべて $D_{i,j} = 1$ とするため，その場合の式 (2) は次のように単純化される．

$$E_n = S \tag{4}$$

すなわち，障害が発生すると，そのオブジェクトから依存関係があるすべてのオブジェクトで障害の重大度がそのまま影響の大きさとして伝わり，各オブジェクトの管理者に通知を行うことを意味する．このような単純な運用であっても，まったく依存関係がない箇所の管理者への通知を防ぐことが可能であり，無駄な通知を減らして重要な通知の埋没化を抑止する．

4.4 管理者への障害通知システム

前節で述べた方式に基づいて，あるオブジェクトにて障害が発生した際に，管理者への通知を行うシステムが構成できる．このシステムは**図 4** に概要を示すように，依存関係抽出部，障害検出部，通知レベル算出部，通知部の 4 つの部分からなる．

依存関係抽出部 各サーバや周辺機器を構成する全オブジェクトの依存関係を 3 章の方式に基づいて抽出する．

障害検出部 管理対象の各サーバを監視し，管理サーバに知らせる．この部分には既存の様々な障害検出方式が利用できる．

通知レベル算出部 前節で述べた算出法に基づいて，障害の影響を受ける各オブジェクトの通知レベルを算出する．

通知部 あらかじめ設定されている各オブジェクトを

表 1 影響度と通知レベルの算出結果
Table 1 The result of notice level.

クラス	ホスト	主属性	重要度	影響度	通知レベル
Disk	C	/dev/dsk/c0t0d0s7	1.0	0.900	0.900
Directory	C	/export	0.0	0.900	0.000
Directory	B	/usr/local/www	0.0	0.900	0.000
Service	B	HTTP:8080	0.5	0.900	0.450
Directory	A	/www	0.0	0.630	0.000
Service	A	HTTP:80	1.0	0.630	0.630

左側：図 6 に基づく設定
右側：重大度 0.9 の障害がホスト C のディスクで発生した場合の算出結果

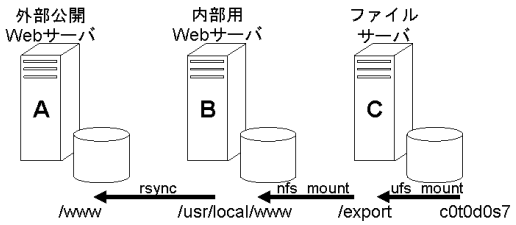

図 5 Web サーバの構成例
Fig. 5 The example for web servers.

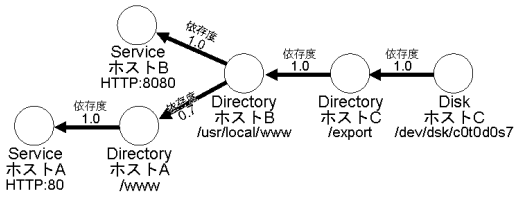

図 6 各オブジェクトの依存関係
Fig. 6 The dependencies in the example.

担当する管理者に対して，算出された通知レベルをもとに，適切な内容の通知を行う．

4.5 運 用 例

ここでは，サーバシステムの例を用いて，これまでに述べた方式を用いて管理者へ通知を行うシミュレーションについて述べる．

図 5 に Web サービスを提供するシステムの構成例を示す．このシステムでは，すべてのコンテンツはファイルサーバであるホスト C に格納されており，組織内向け Web サーバであるホスト B は，ホスト C のディレクトリを NFS で共有している．また，組織外にサービスしている Web サーバ A はホスト B のコンテンツを定期的にリモートコピーしている．この場合の各オブジェクトの依存関係を 3 章の方式で抽出すると**図 6** のグラフが得られる．この各依存関係について依存度を図 6 上の数値のとおり設定する．ホスト A のディレクトリ/www はコピーであり，コピー元の障害によってただちに停止するわけではないが，長期的視点ではデータの更新が受けられないので依存度を 0.7 としている．

各オブジェクトに通知が必要かどうかという観点で**表 1** のとおりに重要度を設定する．ここで，ホスト C のディスク c0t0d0s7 で重大度 0.9 の障害が発生した場合の，各オブジェクトの影響度と通知レベルは表 1 右側のとおりに算出される．ただし，ここでの伝播指数は $p = 1$ とする．なお，主属性[16] とはそのオブジェクトを識別するための属性である．

表 1 の結果をもとに具体的にどのような通知を行うかは，システムの実装や管理者のポリシに基づく設定によって定められる．たとえば，管理者が通知レベルを 5 段階に分けて，「0.0〜0.2 通知しない，0.2〜0.4 デバッグ情報，0.4〜0.6 存在の通知，0.6〜0.8 警告，0.8〜1.0 重大なエラー」と設定していた場合には，これらの通知レベルの値から，次のような通知が行われる．ホスト C のディスク管理者には 0.9 という通知レベルで障害の詳細情報が送られるが，ホスト B の HTTP サービスは組織内向けの重要性の低いサービスであるため管理者への通知は障害が存在することのみを伝える程度（0.45）となる．一方，ホスト A の HTTP サービスは組織外へ公開している重要なサービスであるため，影響は致命的ではないにもかかわらず，注意を喚起するようにホスト B の場合よりも詳しい警告通知（0.63）が得られるように設定できた．

なお，算出した通知レベルに基づいて実際にどのように通知するかは，各現場によって異なり，また 1 つの部署でも各管理者によって異なるため，運用時に設定を行う．従来の syslog やその他近年の統合管理ソフトウェアなどでも，各情報を重大度付きで扱って統合管理しているが，各重大度でどの管理者にどのように通知するかは運用時にシステムに設定しているのと同様である．

5. 議 論

本章では，提案する障害通知方式の有用性に関する議論を行う．

5.1 各管理者への個別通知

従来の障害管理ソフトウェアの多くは，あらかじめルールを記述し，これに適合する障害が発生した場合に指定されたアクションを実行して通知を行う．したがって，各管理者の要求に合わせて細かくルールを設定することによって，どのような要求にも正確に適合させることが原理的には可能である．たとえば，あるディスクドライブの故障時に通知する管理者として，ディスク管理者のみでなく，そのディスク内のデータを利用するサーバやアプリケーションの管理者も通知先として記述すればよい．

しかし，近年はサーバ類を大規模に集中管理する傾向にあり，組織内データセンタでは巨大なサーバ群を管理している．我々の大学においても，高可用性サーバなどの大型システムが十数式あり，全学に配置している小型のサーバ類まで数えるならば監視対象のサーバは百数十システムになる．いくつかの障害検出ソフトウェアを用いて障害情報を収集しているが，数千本のディスクドライブなど，オブジェクト数は 1 万を超える．それらのサーバのすべての部品に対して影響を受ける管理者を判別し，個別にアクションのルールを設定することは非現実的である．

さらに，近年の複雑なサーバは様々なレイヤに分かれており，またそれに対応して管理者も異なる場合が多いため，レイヤを越えた依存関係に合わせてルールを設定することはきわめて困難である．たとえば，ストレージサーバの場合，単にディスクドライブをマウントするのではなく，RAID 構成のディスクアレイが一般的であり，さらにその仮想ディスクを複数のディスクアレイにまたがって仮想化したファイルシステムを利用する技術も普及しつつある．このように何段階にもレイヤ分けされたストレージをアプリケーションから利用する場合，担当するコンテンツが物理的にどこに格納されているかをサービス管理者が把握することはきわめて困難である．低いレイヤの情報を隠蔽することが仮想化やレイヤ分けの目的の 1 つではあるが，障害などの特殊な状況下で情報が必要となったときに大きな問題となる．

前述の我々の大学のシステムでは従来の統合運用管理ソフトウェアを用いているが，以上のような理由から個別のルール設定はほとんど行えず，すべての障害情報を約 10 人の管理者の全員に対して通知する設定を行っている．その結果，正常時のレポートのみでも 1 日に数十通のメールを受け取り，障害が多い場合には数百通のメールを受け取る．そのため，各管理者にとって重要な通知が埋没化していることが大きな問題となっている．以上のことから，従来の方式は，**原理的には細かくアクション設定することが可能でも，現実的には大規模・複雑なシステムでの運用は不可能**であるといえる．

これに対し，本論文で提案する方式に基づく障害通知システムでは，各管理者が自分の担当システムを認識していることは当然であり，それらに関して現在でも管理ソフトウェアを操作しているので，それらの各オブジェクトのみに関して設定することは比較的容易と考える．たとえば，ディスクドライブが千本あるサーバにおいて，コンテンツが異なって影響先が多岐にわたる場合であっても，ドライブの交換担当者が同一であれば，1 回の設定で十分である．

すなわち，各管理者は各自が担当し把握している箇所のみの設定で，関連する他の担当者のシステムまで含む個別設定を全オブジェクトに別々に行った場合と同等となり，さらに依存の程度に適合する内容の通知を受けることが可能である．

5.2 システム構成変更

様々なサーバ群を集中管理する組織では，システムの部分的構成変更も頻繁にありうる．これは新しいシステムを導入した場合のみでなく，たとえばディスク内のデータを整理してより適切な領域へ移動したり，設定は変わらなくても試験運用から本稼動へ移行し，オブジェクトの重要度が変わったりする場合もある．

ここで，従来の運用管理ソフトウェアを用いて，前節で述べたように全オブジェクトへの個別設定が行えたと仮定する．しかし，構成変更が発生した際に，そのオブジェクトが依存しているすべてのオブジェクトに関するルール設定を修正する必要がある．これは設定変更の量的な問題のみでなく，管理区分を越えた設定変更が発生するという問題をも生じる．

一方，本論文の方式を用いると，その構成変更を担当して詳細を把握している管理者がそのシステムに対応する設定のみを修正すればよい．

ただし，本方式が利用している依存関係抽出法は静的な依存関係のみを対象としている[16]．したがって，システムの構成変更が行われるたびに依存関係の再抽出を行う必要があるが，変更に関連する全オブジェクトの設定を修正する必要がないため，これと比較すれば小さなコストであると考える．

たとえば，千本以上のディスクドライブを持つファイルサーバにおいて，あるアプリケーションのデータ領域を増やすために新たなファイルシステムを割り当てた場合，従来方式では，そのファイルシステムがある RAID グループに含まれている全ディスクドライ

ブを調べて，通知先を変更する必要がある．これに対して，提案方式では，OS のマウント情報などを収集して依存関係の再抽出が自動で行えるため，すなわち，通知に関する設定変更をまったく行わずに，通知先は構成変更に自動追従する．

5.3 管理者の階層

大規模なサーバ群を運用する組織では，運用管理部門に複数の管理者が所属していることが多い．管理者の業務支援という観点から管理者の階層について議論する．

複数の管理者がいるとき，各管理者のスキルは一定でない場合が多い．システムの全体的設計を行ったり，全システムを把握したりしている**上級管理者**や，経験が浅く，運用マニュアルに従って担当システムに関する日常業務を行う**初級管理者**がいる．また，両者の中間的スキルを持つ**中級管理者**もいる．

初級管理者は，運用マニュアルに沿う作業は担当できるが，経験が浅いために複雑な障害に対応できない．複雑な依存関係を持つサーバ群では，障害の発生箇所と原因がまったく異なることも多いためである．たとえば，ディスクドライブの故障に関係して RAID5 の再構築を行っているためディスクの性能が低下し，データベースアクセスに時間がかかり，Web アプリケーションでタイムアウトが発生するという場合を想定する．著者の経験では，アプリケーションを担当する初級管理者は症状の出ている箇所と異なる箇所の広い可能性を検討しない傾向にあり，自サーバ内で原因を探そうとして，原因究明に多くの時間を必要とする可能性が高い．

このような状況では，初級管理者は中級管理者や上級管理者に相談することとなる．その結果，障害の解析は可能となるが対応に時間がかかる．またこの方法を繰り返しているのでは上級管理者が個々の障害対応に携わることになり，階層の上位に位置する上級管理者に負担が集中する．

上級管理者が持つ設計情報を依存関係抽出システムへ与えることにより，上級管理者の持つノウハウの一部を初級管理者が参照可能になる[16]．しかしながら，初級管理者は広い可能性を検討しない傾向にあるため，障害通知を受け取った際に構成管理システム内の設計情報をすぐに参照しない可能性が高い．本論文の方式では，障害通知システムが構成情報を直接参照して通知を行うことによって，障害の原因究明時間が短縮できる．

また，障害通知は管理者ごとにその影響度に応じた内容にすることが可能であるため，エラーメッセージなどの通知に含まれる情報の意味を理解する前に，自分への影響の大きさをただちに把握することができるため，初級管理者の障害管理業務の負荷が軽減される．

6. おわりに

本論文では，サーバ本体や周辺機器の各部に注目し，それらの間の依存関係に基づいて，適切な管理者のみに適切な内容で，障害などのイベント情報の通知を行うための方式の提案を行った．

近年は大規模で複雑なサーバ群を複数の管理者で管理する傾向にあり，それらのサーバはハードウェアからサービスアプリケーションまで様々なレイヤの各部が複雑に依存し合う構成となっている．そのため，従来の方式で原理的には障害通知可能であっても，現実的には大規模な設定が不可能で，膨大な量の障害情報が整理できず，管理業務の負担を増加させていた．本論文で提案する方式は，すでに抽出された依存関係を用い，単純な式により障害通知レベルを決定しているが，この方式によって実際のシステム運用管理の現場で利用できる可能性を見出したといえる．

サーバを構成するハードウェアやソフトウェア自身の信頼性が向上しても，それらを管理する管理者の業務を支援できなければ，全体としてのサービスの信頼性は向上しない．サーバが今後さらに大規模複雑化する社会において，本論文で提案した障害通知方式はきわめて有用である．

本論文の方式は複数の障害を別々に扱う．1 カ所の故障から複数の障害情報が発生した場合には，依存関係に基づいて関連付けを行い，それら複数の情報を集約することが今後の課題である．

参考文献

1) 敷田幹文，井口　寧，三輪信介，丹　康雄，松澤照男：大規模高可用性サーバの設計と運用，情報処理学会分散システム/インターネット運用技術シンポジウム，pp.57–62 (2001).
2) SUN Microsystems, Inc.: SUN Cluster2.2 System Administration Guide (1999).
3)（株）日立製作所：JP1 Version 8 (2007). http://www.hitachi.co.jp/Prod/comp/soft1/jp1/
4)（株）日立製作所：JP1 Version6 JP1/Base マニュアル (2001).
5) 富士通（株）：SystemWalker/CentricMGR Version 13.2 (2007). http://systemwalker.fujitsu.com/jp/
6) サン・マイクロシステムズ（株）：Sun Management Center 3.0 ソフトウェアユーザーマニュア

ル (2001).
7) 敷田幹文，井口 寧，藤枝和宏，松澤照男：高可用性システム統合監視機構の提案，情報処理学会分散システム/インターネット運用技術シンポジウム，pp.51–56 (2002).
8)（株）日立製作所：統合ネットワーク管理システム ネットワークノードマネージャネットワーク管理ガイド (2000).
9) 日本ヒューレットパッカード（株）：HP OpenView. http://www1.jpn.hp.com/products/software/management/openview/
10) 長田智和，谷口祐治，玉城史朗：大規模分散ネットワーク運用管理システムの提案，情報処理学会分散システム/インターネット運用技術報告 DSM-20，pp.31–36 (2000).
11) 知念眞也，長田智和，谷口祐治，玉城史朗：分散オブジェクト技術を用いたネットワーク監視システムの設計，情報処理学会分散システム/インターネット運用技術報告 DSM-20，pp.37–42 (2000).
12) 三好 優，釜洞健太郎，朴 容震，浦野義頼，富永英義：モバイルエージェントによる大規模・分散形ネットワーク管理法の一提案，情報処理学会マルチメディア通信と分散処理報告 DPS-97 (2000).
13)（株）日立製作所：統合ネットワーク管理システム サーバシステム管理 第 2 版 (2001).
14)（株）日立製作所:JP1 Version 8 統合管理 (2007). http://www.hitachi.co.jp/Prod/comp/soft1/jp1/product/merits/int/
15) 清水雅司，敷田幹文：モバイルエージェント技術を用いたサーバ監視システムの設計，情報処理学会分散システム/インターネット運用技術報告 DSM-27，pp.13–18 (2002).
16) 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).

(平成 19 年 6 月 11 日受付)
(平成 19 年 12 月 4 日採録)

敷田 幹文（正会員）

1965 年生．1995 年東京工業大学大学院理工学研究科情報工学専攻博士後期課程修了．博士（工学）．同年北陸先端科学技術大学院大学情報科学センター助手．2001 年同助教授．大規模分散システム，グループウェアに関する研究に従事．ACM，電子情報通信学会，日本ソフトウェア科学会各会員．