5D-05

階層型システムの無停止機能更新を実現する運用手順自動生成手法の提案

上野 優† 関口 敦二† 安家 武† 阿比留 健一†

†株式会社富士通研究所

1 はじめに

情報処理システムの機能更新は，運用系，待機系，および切り替え装置を組み合わせることで，サービスを止めずに行うことができる．ところが，こうした無停止機能更新は，更新手順の順序関係を考慮する必要があるため，システムの管理者にとって，その手順決定の負担が問題となっている．そこで本稿では，無停止機能更新を実現する運用手順自動生成手法を提案する．

2 無停止機能更新の手順生成とその課題

本稿が対象とする無停止機能更新，先行研究およびその課題について述べる．

2.1 無停止機能更新

図 1 に，文献 [1] で述べられている，サービス無停止を可能にするシステムと，機能更新手順を示す．

図 1 のような階層型システムにおいて，機能更新中にサービスが停止する場合を 4 つ挙げる．第 1 に，切り替え装置 (LB) がユーザアクセスを処理できなくなった場合，第 2 に，ユーザ-LB 間，LB-APP 間，APP-DB 間のネットワーク接続性が失われた場合，第 3 に，LB が接続するアプリケーション（運用系 APP）が機能停止し，ユーザアクセスを受け取れなかった場合，第 4 に，アプリケーションとそれが利用する DB のスキーマが，不一致の場合である．

本稿では，図 1 の手順①～③のような手順考案時に問題となる，第 3，第 4 の場合に着目し，手順生成のアイデアを示す．

2.2 先行研究とその課題

クラウドの発展に伴い，リソース最適化をねらったサーバ構成の変更が頻繁に行われるようになったため，

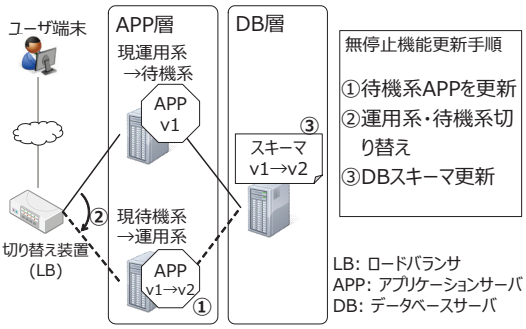

図 1 運用系・待機系を組み合わせたシステムと，無停止機能更新手順

管理者にとって，構成変更に合わせた手順を考案する工数の削減が課題となっている．ところが，以下に述べるように，システム全体の整合性・無停止性を満たす，具体的な手順を提示する仕組みは提案されていない．

ネットワーク装置，OS などの設定値間の依存関係の把握を容易にし，依存関係の見落としを防ぐことで運用の判断を支援するフレームワークが提案されている [2]．しかし，設定値を変更する具体的な操作が，依存関係を満たすかどうかを判断する仕組みは述べられておらず，運用手順を生成することはできない．

また，分散システムの機能更新において，ノード間の通信インタフェースに不整合を起こさない手順を算出，提示する方法が提案されている [3]．しかし，2 ノード間の整合性・無停止性を判断する仕組みは述べられているが，3 ノード以上が関わる，システム全体としての整合性・無停止性を判断する仕組みは述べられておらず，図 1 のような機能更新手順を提示することはできない．

3 無停止機能更新手順の自動生成

無停止判定条件

生成する手順がサービス無停止であるかを判定する条件を以下に示す．

条件 1 ユーザアクセスが振り分けられている APP サーバの機能を停止しない

条件 2 アプリケーションと DB スキーマをともに更新する場合，アプリケーションを先に更新する

Proposal of a method to generate the operational procedure that supports zero-down-time upgrade of multitier systems

UENO, Masaru†, SEKIGUCHI, Atsuji†, YASUIE, Takeshi†and ABIRU, Kenichi†

†FUJITSU LABORATORIES LTD.

{ueno.masaru,sekia,yasuie.takeshi,abiru}@jp.fujitsu.com

```
// 条件 1 を満たす場合 true，満たさない場合 false を返す
stateA := 始点に対応する状態オブジェクト
stateB := 終点に対応する状態オブジェクト
if stateA と stateB の差分が APP サーバの属性値である
    app := 属性値の異なる APP サーバ
    if app の前段の LB の振り分け先サーバが，app である
        return false

return true
```

図 2　判定条件 1

```
// 条件 2 を満たす場合 true，満たさない場合 false を返す
stateInit := 探索の開始点（初期状態）に対応するオブジェクト
stateA := 始点に対応する状態オブジェクト
stateB := 終点に対応する状態オブジェクト
if stateA と stateB の差分が DB サーバの属性値である
    db := 属性値が異なる DB サーバ
    app := db に接続されている APP サーバ
    if app の前段の LB の振り分け先サーバが，app である
        if app のバージョンが，stateInit に等しい
            return false

return true
```

図 3　判定条件 2

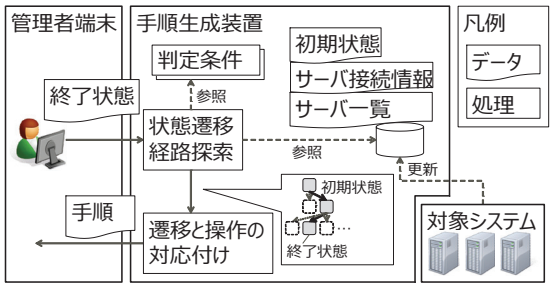

図 4　提案手法の全体像

これらは文献 [1] の記述から抽出・作成したものであり，次に述べるグラフ探索時に用いる．図 2，図 3 に，判定条件 1，2 を表す擬似コードを示す．

手順自動生成の方法

機能更新手順の自動生成の考え方を述べる．まず，システムを構成するサーバの属性値の組を，システムの状態ととらえ，状態遷移グラフの頂点と見なす．すると，サーバに対する操作は，状態遷移グラフの頂点から別の頂点への有向辺ととらえられる．そして初期状態と終了状態を表す，サーバの属性値の組を与え，初期状態から終了状態に至る最短経路を幅優先探索し，その経路に対応した操作を並べることで，手順を得る．この経路を探索する時，図 2，図 3 の判定条件がともに `true` となる辺のみを辿るため，無停止条件 1，2 を満たす手順を得る．

図 4 に本提案の全体像を，図 5 に本提案で得られる手順と，手順実行中のシステムの状態遷移を示す．図 5 の操作 1，2，3 を順に並べたものが，本提案で得られる無停止機能更新手順である．ここで，例えば操作 1 は，指定の版 (`v2`) のアプリケーションを対象サーバ (`APP02`) に配備し，旧プロセスを停止，新プロセスを起動する操作である．同様に，操作 2，3 は，それぞれのサーバ役割に応じて処理を行い，結果的にサーバの，ひいてはシステムの状態を変更するものである．

サーバID	LB01	APP01	APP02	DB01
サーバ役割	LB	APP	APP	DB
属性名	backend	version	version	schema_version
初期状態	APP01	v1	-	10
	↓	↓	v2	↓
	APP02	↓	↓	↓
	↓	↓	↓	20
終了状態	APP02	v1	v2	20

操作1　操作2　操作3

図 5　得られる手順と状態遷移

4　議論

従来の手順生成手法 [3] では，システム全体の整合性・無停止性を満たす具体的な機能更新手順を得るためには，隣り合う階層間の依存関係を満たすように操作の順序関係を入力する必要があった．例えば，図 5 の 4 つの状態に 4 つずつ含まれる，計 16 の属性値の遷移を把握した上で，順序関係を考慮する必要があった．一方，提案手法は，2 つの判定条件を満たす無停止機能更新手順を，グラフ探索により生成するものであり，管理者は，図 5 の例では，終了状態に含まれる 4 つの属性値のみを入力すればよく，工数削減効果を得る．さらに，提案手法は，図 1 にウェブサーバ層を追加した場合や，各階層のサーバ数が増減した場合にも，2 つの判定条件をそのままに適用できるため，構成が変化した場合の手順生成を容易にし，工数削減効果を得る．

5　まとめ

階層型システムの機能更新において，隣接する階層のサーバ設定値が満たすべき 2 条件を定義し，それらを満たす無停止機能更新手順を生成する手法を提案した．

今後の課題として，本手法の適用範囲を精査し，より汎用的なシステムやオペレーションに対する支援を検討することが挙げられる．

参考文献

[1] Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education, 2010.
[2] 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.
[3] 内田直樹. 高度 IN 分散システムにおけるプログラム更新手順の検討. 電子情報通信学会技術研究報告. SSE, 交換システム, Vol. 96, No. 251, pp. 43–48, 1996.