2004－HI－111　(12)
2004／11／12

システム運用管理におけるUNDO操作ユーザインタフェース

池上 輝哉, 加藤 清志, 中村 暢達, 平池 龍一
ＮＥＣ インターネットシステム研究所

ＩＴサービスは社会インフラの一つとなっており、たとえ障害が発生しても即座に復旧可能とする、あるいは顕在化させないといった高い可用性が要求されるケースが多い。著者らは、サービスの耐障害性向上の一つの方策として、オペレーターの操作ミスやアプリケーションバグ等に起因する障害が発生した際にも、過去のシステム状態への復旧を可能とするＵＮＤＯ機構を開発した。本稿では、このＵＮＤＯ機構において、アプリケーションの入出力データと処理データを記録し、障害が発生した際には、アプリケーションデータを記録時の状態に戻し、障害の原因となった入出力データの分析と除去を可能とするＵＩを提案する。

Undoable Management User Interface for System Reliability

Teruya IKEGAMI, Kiyoshi KATO, Nobutatsu NAKAMURA and Ryuichi HIRAIKE
Internet Systems Research Laboratories, NEC Corporation

Recently, system reliability is becoming important and computer systems should be recovered immediately when some failures occurred. We have developed the undo mechanism which enables to control states of the system by taking snapshots and web-queries, and allows operators to recover from their own mistakes or activated application bugs. This paper describes about the undoable management user interface which enables to restore snapshots and analyze web-queries that caused failures by utilizing the undo mechanism.

1. はじめに

今日の、ITサービスは無くてはならない情報インフラと化している。ITサービスへの要求レベルや規模、社会的重要性が大きくなればなるほど、障害が発生した場合に、関連するサービスを巻き込んだ大きな損害につながる恐れがあり、個々のサービスにおいて、より高いレベルの可用性が要求される。

サービス障害を引き起こす原因には、ソフトウェアの設計ミスや運用者の判断･操作ミスなど、これまで主にハードウェア領域において強化されてきた耐障害性の機能だけでは充分に対応できないものが多く含まれる。このような原因に対処するには、ソフトウェ

ア領域においても何らかの耐障害性機能を設け、かつハードウェア領域の耐障害性機能と密接に連携させることが必要となる[1]。

その中で、操作ミス防止や即時復旧のためのユーザインタフェース（UI）も重要な要素となる。しかしながら、これまでの運用管理業務におけるUIは、ある程度の専門的な知識と経験を備えた人間が用いることを想定しており、効率に重点がおかれることが多く、操作者を支援する機能についてはあまり考慮されていなかった。専門家からすれば、GUIより寧ろCUIが好ましいという意見もある[2]。特に、運用を開始してから長い時間が経過しているシステムでは、設計当初に比べ環境が大きく異なる場合もあり、操作者の技量のみに頼るやり方では判断ミスや操作ミスを避けられない。そういったミスを防ぐには、システムの状態を適切に提示することで、操作者が的確な判断を下せるようにするUIが極めて重要である。

著者らは、システム運用管理における耐障害性向上の一方策として、HTTP-GW（ゲートウェイ）機能を用いたシステム状態制御UIを提案する。HTTP-GW機能とは、管理対象とするシステムの状態を、仮想マシンのスナップショットと実行処理要求/応答のウェブクエリにより管理するものである。提案するUIは、本機能を活用し、過去のシステム状態へ戻り（UNDO）、アプリケーションを再度実行することで、復旧操作を開始する直前の状態とすることを可能とする。本稿では、システム運用管理におけるUNDO機能について考察した後、提案UIについて述べる。

2. システム運用管理におけるUNDO機能

高いサービス可用性を求められるシステムの運用においては、オペレーターの操作ミスやアプリケーションエラーに起因して何らかの障害が発生した場合にも、即時にシステム状態を修復できる機能が必要となる。

以下、システム状態管理の手法を整理し、システム運用管理の観点からみた、備えるべきUNDO機能について考察する。これらの考察を踏まえて、著者らが開発したHTTP-GW機能について述べる。

2.1. システム状態の管理

「システム状態」を構成する要素としては、サービスを提供するアプリケーションの動作状態と、アプリケーションが記録／参照するデータがある。このアプリケーション動作状態およびデータ、それぞれを管理する手法を挙げ、システム運用管理へ適用する場合の留意点を示す。

2.1.1. アプリケーション動作状態の管理

アプリケーション動作状態を管理する一般的な手法として、ジャーナル管理がある。ジャーナル管理は、動作状態を管理するために入出力データを記録するものである。

例えば、Microsoft 社の Web Application Stress Tool や、IBM 社の Synthetic Transaction Investigator では、システムのパフォーマンステストに用いる入力シーケンス（ウェブクエリやDBトランザクション）を記録することや、後から実行（再生）することが可能である。テスト時には、実際にインターネット上のウェブコンテンツにアクセスし、リンクを辿っていくといった操作の流れを、入力シーケンスとして記録する。次に、負荷をかける期間やクライアント-サーバ間の回線速度等のテスト実行環境を任意に設定した上で、記録した入力シーケンスを再生する。

入力シーケンスは、リストやXML形式のテキストとして提示され、テスト実行者は詳細情報の参照や編集が可能となる。また、テスト結果として、応答時間やクライアント数、データ総量、タイムアウトしたスレッド総数等、様々な情報を数値やグラフとして参照できる。

ジャーナル管理では、入力シーケンスの記録･編集および細やかな再生設定の手段と、再生結果の

詳細なレポート生成機能が求められる。運用管理業務では、これらの機能に加え、システムが正常に動作しているかを常に把握可能とすることが求められ、シーケンス実行中の、個々の入出力処理の成否についても、操作者が確実かつ素早く把握できるように情報を提示することが重要となる。

2.1.2. アプリケーションデータの管理

アプリケーションが記録／参照するデータを管理する一般的な手法として、バックアップ管理がある。バックアップ管理は、ある時点におけるアプリケーションデータを記録するものである。

例えば、Windows XP における「システム復元」機能がある。本機能により、OSに何らかの障害が発生した場合、システムの設定を事前に記録された状態（チェックポイント）へ復元できる。また、チェックポイントを記録した日時に加え、特定のプログラムがインストールされた日時についても表示される。これは特定のプログラムがインストールされる際にチェックポイントが自動生成されることによる副次的な情報であるが、例えば特定のプログラムをインストールした後にシステムに不具合が発生した場合に、読み出すチェックポイントを選択するための有用な情報となりうる。

バックアップ管理を用いてデータの復旧を行う場合、アプリケーションデータを過去の状態に戻してから、できるだけ復旧作業前のシステム状態に近づけるために、再度アプリケーションをインストールするといった作業が必要となることが多い。特に、個々の作業がシステム状態に大きな影響を与える可能性が高い運用管理業務では、チェックポイント記録時の前後に行われた作業を提示することが重要となる。更に、複数のチェックポイントを保持した状況で、適切なチェックポイントを選択する際に、チェックポイントに関する情報を一覧し、それぞれを比較できることが望ましい。

2.2. UNDO操作のための情報管理

前節で、システム状態を構成する要素として、アプリケーション動作状態とデータがあることを述べた。運用管理業務においては、この何れか一方だけを記録するだけでは不充分であり、両方を併せて記録することが重要となる。

以下、システム状態をUNDO可能とするために、管理すべき情報について述べる。

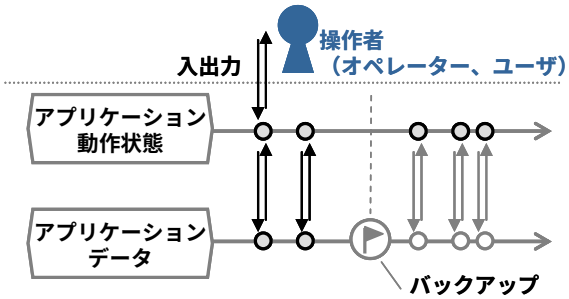


図 1. 動作状態とデータの関係

先ず、アプリケーションデータの管理について述べる。図1に示すように、例えば適当な周期毎にアプリケーションデータのバックアップをとれば、その時点へのデータ復旧は保証される。しかし、常にエンドユーザからのリクエストを処理する必要のあるシステムにおいては、単にデータを過去の状態に戻すだけでは、バックアップ以後に処理されたエンドユーザからのリクエストがデータに反映されていないことになる。換言すると、任意のシステム状態への復旧を可能とするために、アプリケーションデータだけでなく、アプリケーション動作状態も管理することが必要となる。

そこで、次に、アプリケーション動作状態の管理において、管理対象とするべきアプリケーション入出力データについて述べる。

図 2 に、アプリケーションへの入力データによって引き起こされる障害の典型的な例を示す。オペレーターが、ネットワーク設定等で誤った設定を入力してしまった場合、結果としてシステム誤作動やサービス

停止につながる恐れがある。また、開発段階では発見されなかったバグが特定の入力シーケンスによって活性化することも考えられる。特に、複数のエンドユーザからのリクエストを一括して処理する必要があるアプリケーションでは、その開発段階で入力シーケンスを網羅的に想定することは困難なため、結果的に、稀な条件下でのみ活性化するようなバグが残ることは避けられない。

従って、システム管理を行うオペレーターの操作と、サービスを享受するエンドユーザのリクエストの両方を、アプリケーションの入出力データとして管理するべきである。

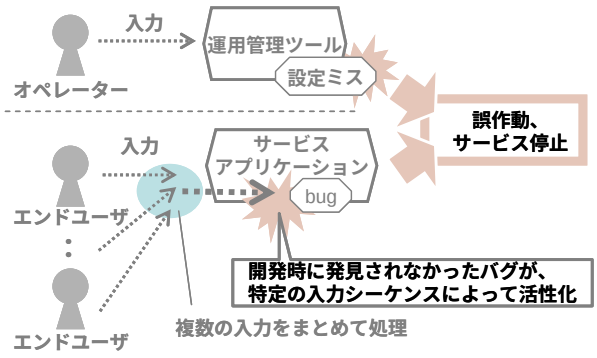


図 2　アプリケーションへの入力による障害例

2.3. HTTP-GW機能

UNDO機構を設計する際に、アプリケーションの入出力の意味を解析すれば、例えばアプリケーションデータに影響を与えない入力は重要でないとみなして記録しないといった処理も可能となる [3]。しかし、現実には様々なサービスが存在し、それらに対応する個々のアプリケーションの入出力の意味を全て解析することは非常に困難である。従って、アプリケーションに特化した機構を必要としない、汎用的な入出力データ管理を考えることが望ましい。

そこで著者らは、Web/AP/DBの三層構造モデルを対象としたHTTP-GW機能を開発した[4]。図 3 に本機能の概要を示す。

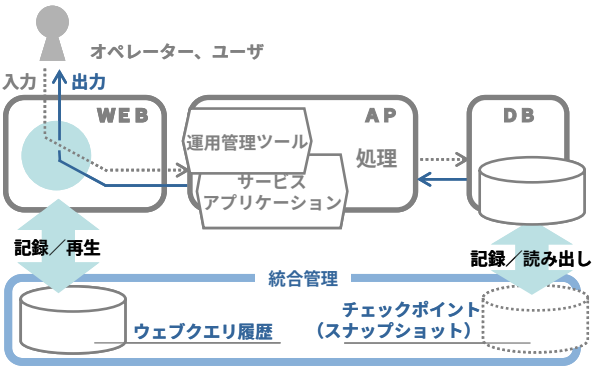


図3　HTTP-GW機能概要

アプリケーション入出力インタフェースのプロトコルとして一般的に使われているHTTPを対象とし、オペレーターやユーザからの処理要求および応答をウェブクエリ履歴として記録し、後から再生可能とする。更にアプリケーションデータを、仮想マシンのスナップショット（実行環境のメモリイメージを二次記憶へ書き出す）により、チェックポイントとして記録する。チェックポイントを記録する時点で、最後に処理が終了したクエリの情報をあわせて記録することで、チェックポイント読み出し実行後に、再生処理を開始するクエリを特定し、任意のシステム状態へ戻すことが可能となる。

本機能では、ウェブクエリ履歴において、個々のクエリの詳細情報を参照することや、再生時に処理するか否かを設定することが可能である。また再生時には、記録した際のクエリの入力タイミングを再現することや、クエリを一つ処理する毎に再生を一時停止してシステム状態を確認することが可能である。このような柔軟なクエリ管理と、アプリケーションデータの記録･読み出しを可能とするチェックポイント管理とをあわせて備えることにより、障害発生時にチェックポイントまでシステム状態を戻すだけでなく、障害の原因となったクエリの発見や除去が可能となる。

3. UNDO操作ユーザインタフェース

著者らは、HTTP-GW機能を活用し、システム状態のUNDO操作を可能とするUIを開発した。

以下、本UIの特徴を述べた後、障害対処時の本UIの操作例を示す。

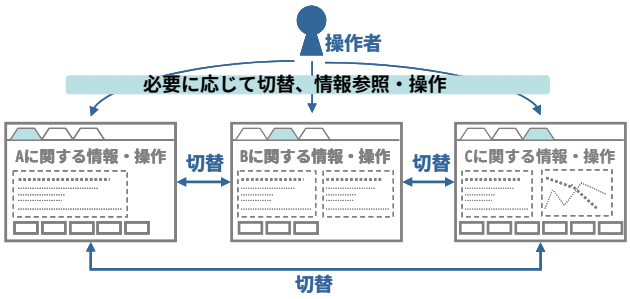
3.1. 情報の階層化

システムを運用管理する際には、オペレーターが、管理対象とするシステムに関する大量の情報を常に把握していることが重要となる。

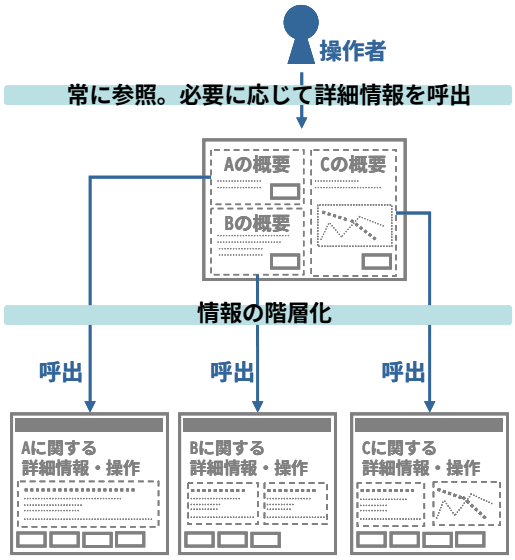
様々な情報を提供する手段として、マルチウインドウやタブによる画面切り替えが考えられる（図 4 (a)）。各々のウインドウやタブに操作目的や情報カテゴリを振り分けることで、目的指向あるいはオブジェクト指向の分かり易いUIが実現できる。しかし、ある情報を参照している際には他の情報が参照できない、あるいは参照し難くなるため、参照すべき情報を隠蔽してしまい見落としを招く恐れがある。

従って、システム運用管理においては、画面の切り替えを必要としないように、一画面上で重要な情報をまとめて把握できる画面を備えるべきである。そこで、情報の参照方法として、重要な情報の概要を把握でき、主要な操作を実行できるような主となる一画面があり、必要な場合にのみ詳細情報の参照画面を呼び出す形態（図4 (b)）とした。

このような階層的な情報の参照を行うUIを設計する際には、常に提供する情報と詳細情報とを切り分ける、情報･機能の階層化が重要となる。本UIでは、HTTP-GW機能の動作状態とウェブクエリ処理状況、最新のチェックポイントに関する情報に加えて、ウェブクエリ処理を制御する主要な操作機能を、基本画面に配置した。また、個々のウェブクエリの詳細な情報や、保有する全てのチェックポイントの一覧および読み出し･削除機能については、個別の詳細画面を設け、必要な場合に呼び出し可能とした。



(a)　タブによる並列的な情報参照



(b)　階層的な情報参照

図 4　情報参照時の画面遷移の形態

3.2. 表示画面と操作方法

図 5 に本UIの基本画面を示す。HTTP-GW機能を制御するために必要十分な情報を、ステータス情報表示部、チェックポイント情報表示部、クエリ情報表示部および操作メニュー部に配し、常に同じ場所に表示するようにした。

画面下のフレーム部には既存の運用管理ツール画面が表示されている。平常時のシステム運用管理はこのツールで行い、何らかの問題が発生した場合やテスト実行を行いたい場合には、フレーム上部の本UIを用いるといった利用シーンを想定している。

以下、基本画面の各情報表示部および個別の画面として呼び出されるチェックポイント一覧画面について述べる。

図 5　基本画面構成

・　**ステータス情報表示部**

HTTP-GW機能の動作状態を表示する。状態に対応したアイコンと説明文を表示しており、操作者は常にシステムの状態を把握することが可能となる。

・　**操作メニュー部**

本UIでは、「再生」だけでなく、クエリ処理制御やチェックポイント管理の操作に「コマ送り」や「巻き戻し」、「記録」といった音楽・映像編集で用いられるメタファを対応付けた操作メニューボタンを設けた。

これにより、クエリ実行（再生）やリストア（巻き戻し）、チェックポイント保存（記録）等のクエリ制御操作をワンボタンで実行可能としている。

・　**クエリ情報表示部**

クエリに関する情報として、処理予定および処理済みクエリのリストを表示する（図 6）。リストに表示する項目としては、要求日時や要求元、要求先のファイルタイプ（拡張子により、画像/操作/ページのいずれかに分類）、応答結果を備える。また、応答に失敗したクエリと、その原因と考えられる操作要求のクエリを強調する配色を用いる。

クエリ再生中は、処理を終えたクエリが、処理予定のリストから処理済みのリストに移行していくことになる。また、クエリ処理を停止している状態において、リスト上の任意のクエリを選択し、より詳細な情報を参照することや、再生時に実行しないように設定することが可能である。

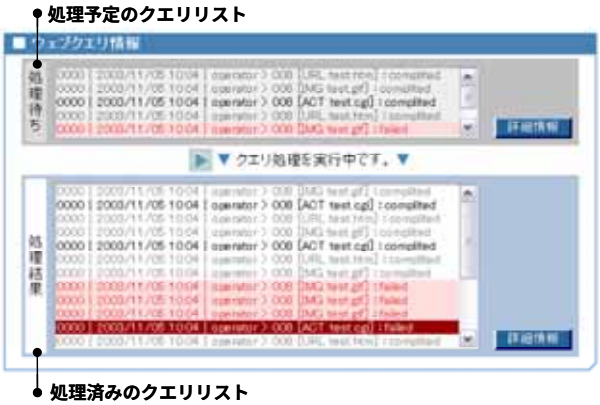


図 6　クエリ情報表示部

・　**チェックポイント情報表示部**

クエリ情報表示部では、一連のクエリ情報をリストにより表示するが、一度に表示できる情報量（行数）には制限があり、全てのクエリ情報をまとめて概観する事はできない。例えば、短期間にまとめて大量のクエリが処理された場合には、一度も画面に表示されることなくリスト後方に流れてしまうクエリが発生し、その結果、処理に失敗したクエリがあっても、操作者が気付かないといったことも考えられる。また、HTTP-GW機能は、ジャーナル管理とバックアップ管理の両方の機能を備えるもので、そのUIは、それぞれの特徴を違和感なく統合する必要がある。

そこで、同じ時間軸上の概念である、チェックポイントとウェブクエリ処理状況に関する情報をあわせて表示するチェックポイント表示部を設けた（図 7）。記録しているウェブクエリ全体を表す「棒」の中で、チェックポイント位置と再生位置を示すことで、現在のクエリ処理状態や、リストア時にどこまで戻ることになるのか、応答に失敗したクエリは無いかといったことが直観的に把握できるようにした。

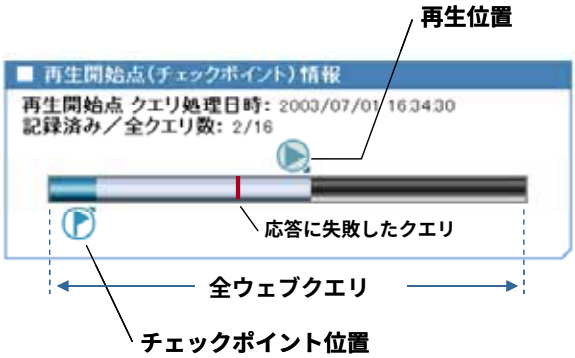


図 7　チェックポイント情報表示部

・　**チェックポイント一覧画面**

保有しているチェックポイントを記録日時順に表示する（図 8）。操作者は、任意のチェックポイント読み出しおよび削除を実行できる。

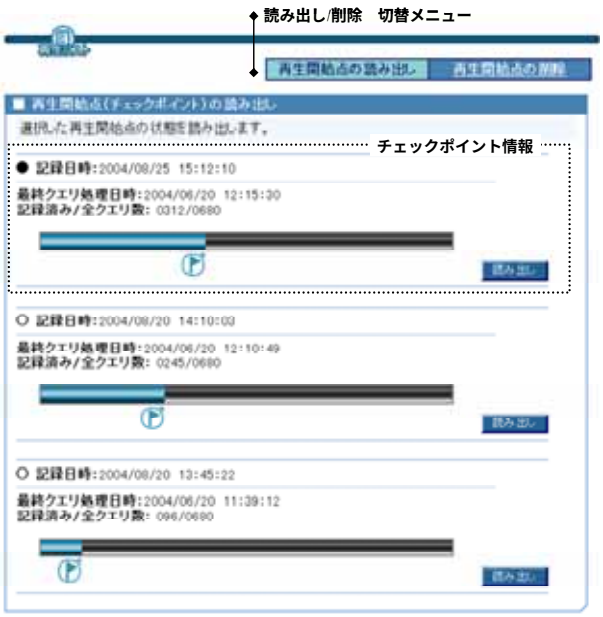


図 8　チェックポイント一覧画面

各チェックポイントにおいて、どこまでクエリ処理が進んでいるかを、基本画面におけるチェックポイント情報と同様の「棒」を用いて並べることで、単に記録日時やクエリ数などの数値を示すよりも、直観的に各チェックポイントのクエリ処理状況を比較することができる。結果として、読み出しや削除を行うチェックポイントを適切に選択できる。

3.3　障害対処時の操作例

オペレーターの操作ミスや、ある種の入力シーケンスによるアプリケーションバグの活性化により、システムに問題が生じた場面を想定し、本UIを用いて障害対処を行う際の操作の流れを示す（図 9）。

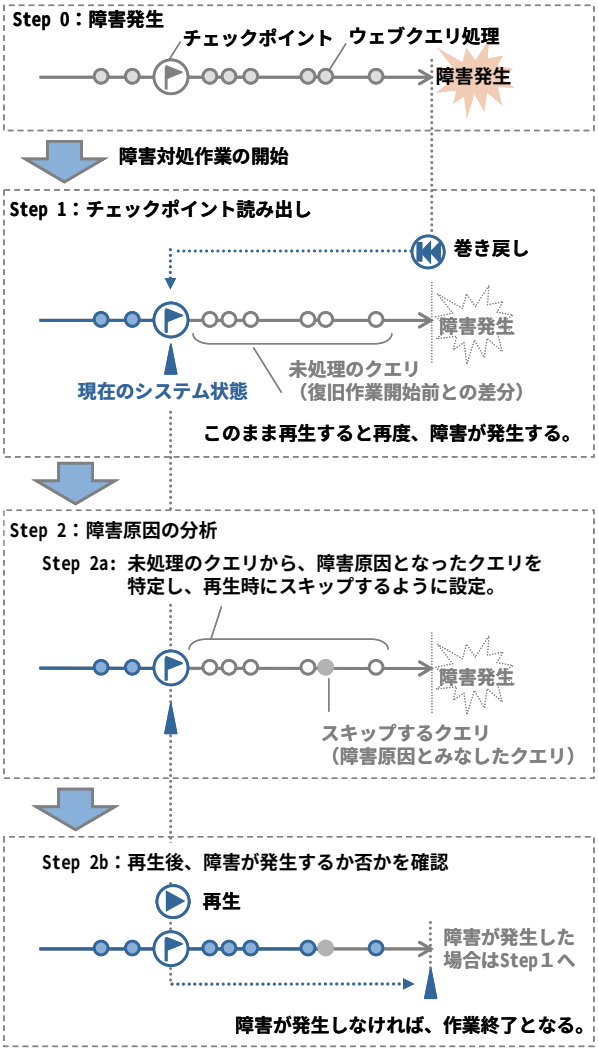


図 9　障害対処時の操作例

・　**STEP 1:チェックポイントの読み出し**

障害への対処を行う操作者は、障害発生前の正常に動作していた時点で記録したチェックポイントを読み出す（巻き戻し）。

単純にチェックポイントを読み出しただけでは、チェックポイント以後のクエリの分だけ、障害対処を開

始した時点のシステム状態との間に差分が存在することになる。ここから、チェックポイント以後のクエリを処理（再生）することで、障害対処を開始する直前のシステム状態に戻すことが可能となる。しかし、先に発生していた障害が再度発生することとなってしまう。

・ **STEP 2：障害原因の分析**

チェックポイントを読み出し、以後のクエリを再生しても、障害が発生しないようにするため、障害の原因を取り除く必要がある。そこで操作者は、チェックポイント以後の処理予定のクエリ内容から障害原因の分析を行う。分析作業としては、個々のクエリの内容を参照し、障害原因となった可能性のあるクエリを洗い出す。疑わしいクエリを再生時にスキップするように設定した上で再生し、同様の障害が発生するか否かを確認する。障害が発生した場合、他のクエリに原因があることになるため、再度チェックポイントまで復帰して同様の作業を継続する。再生速度やコマ送り数を調整することで、より効率的に分析作業を進めることが可能となる。

障害の原因となったクエリを特定できれば、このクエリを再生時にスキップすることで、障害の発生を回避した上で、障害対処作業を開始する直前のシステム状態に戻すことができる。

4. おわりに

本稿では、システム運用管理において、アプリケーション動作状態とデータをあわせて管理することで、たとえ障害が発生しても過去のシステム状態に戻り、障害の発生を回避した上で、障害が発生する直前のシステム状態に戻すことを可能とするＨＴＴＰ-ＧＷ機能およびＵＮＤＯ操作ＵＩを提案した。本ＵＩでは、ジャーナル管理とバックアップ管理それぞれの機能を、音楽・映像編集で用いられるメタファによって統一した操作体系を実現し、チェックポイントの読み出し（巻き戻し）や再生速度の変更といった操作を簡単に実行できるようにした。

今後は、本ＵＩについて、操作の間違え難さや学習容易性、操作効率などを評価するため、実際の運用管理現場での試用を予定している。

参考文献

[1] 中村,加藤,平池、"耐障害に向けたサービスアプリケーション多重実行方式の提案"、情報処理学会第 66 回全国大会、2004

[2] Rob Barrett, Paul Maglio, Eser Kandogan, "Usable Autonomic Computing Systems: the Administrator's Perspective", Proceeding of ICAC'04, 2004

[3] Brown, A. and D. A. Patterson, "Undo for Operators: Building an Undoable E-mail Store", Proceedings of the 2003 USENIX Annual Technical Conference, 2003

[4] 中村,加藤,平池、"自律運用管理におけるサービス指向多重実行方式"、並列/分散/協調処理に関するサマーワークショップ（SWoPP)、2004