複数サーバ環境での解決実績に基づいた システム障害解決支援機構の構築

笠井 貴之† 鷹野 孝典‡

†神奈川工科大学大学院工学研究科情報工学専攻 〒243-0292 神奈川県厚木市下荻野 1030
‡神奈川工科大学情報学部情報工学科 〒243-0292 神奈川県厚木市下荻野 1030
E-mail: †s1785009@cco.kanagawa-it.ac.jp, ‡takano@ic.kanagawa-it.ac.jp

あらまし システム運用において迅速な障害原因の特定・解決は重要である．障害に柔軟に対応できるシステム構成や障害検知の自動化が可能となっているが，障害原因特定は多くの場合に人間が行っており，解決までの試行錯誤を削減するための自動化支援が必要である．既存研究ではルールに基づいて障害原因を特定するものなどがあるが，障害解決の自動化機能は実現されていない．本研究では，システム障害の効率的な解決を目的とした，複数サーバ環境での解決実績に基づいたシステム障害解決支援機構を提案する．提案手法では，対象サーバで障害が発生した際に，障害知識ベースに問い合わせることで解決方法として得られるコマンド群を仮想検証環境上で自動実行することで有効性を検証する．この検証結果を解決実績とし，障害知識ベースにフィードバックすることで記録されたコマンド群に優先順位を付ける．このサイクルにより，障害知識ベースは原因特定・解決に有用であるコマンド群の解決実績情報を継続的に更新し，優先的に障害発生サーバに提供することができる．実験では障害環境を再現した実験環境を用いて提案手法の実現可能性を検証する．

キーワード システム障害，サーバ運用，仮想環境，障害解決，知識ベース

1. はじめに

ソフトウェアシステムの基盤部分は Amazon Web Services[1]や Google Cloud Platform[2]，Microsoft Azure[3]などのクラウドサービスを利用して構築することが一般的になっている．Web コンソールや REST API によってプログラマブルに制御可能なこれらのクラウドサービスを利用することでシステム管理者は大規模・複雑なシステム構成を迅速かつ柔軟に構築可能となった．また，システムの規模増大・複雑化にともなってシステム管理者の人的コストが飛躍的に増大したため，システム管理者は総合的な運用コストを抑えるために可能な限り運用の自動化を行う必要がある．例えばシステム障害の自動検知については Datadog[4]や New Relic[5]のような CPU 使用率やメモリ使用量などのサーバメトリクス監視サービスを導入することによって実現している．しかしながら，依然としてシステム障害の原因調査・解決は多くの場合人間が行っており，クラウドサービスの導入によるドメイン知識の増大にともなって経験豊富な管理者であっても障害解決まで多くの試行錯誤が必要となる．また，このような状況下では運用する対象システム構成の複雑化，規模の増大に比例して経験豊富な管理者に知識・経験が偏ってしまうことで特定人物に障害解決作業が依存してしまうことや人的コストも増大するという問題を抱えている．このようにシステム運用における迅速な障害原因の特定や解決は重要であり，システム障害の原因特定と解決の効率化のためには，人間の試行錯誤を削減するための自動化支援が必要になっている．既存研究ではルールに基づいて障害原因を特定するものなどがあるが，障害解決の自動化機能は実現されていない．

本研究では，システム障害の対象範囲は広く汎用的な議論が困難であるため， Web サービスにおいてユーザへのサービス提供が停止した状態を解決すべきシステム障害として扱う．これには大別して 2 つの難易度設定があると考えられる．1 つ目は設定ファイルやデータ格納ディレクトリ権限設定ミスなどの時間をかければ経験や知識が浅い状態でも解決できる単純なアプリケーション・ミドルウェアのデプロイ時の障害である．2 つ目は CPU やメモリなどのサーバリソースの異常消費など豊富な経験や専門知識を有する管理者がいなければ解決できない OS やネットワークレイヤの障害である．

本研究では，システム障害の効率的な解決を目的とした，複数サーバ環境での解決実績に基づいたシステム障害解決支援機構を提案する．提案手法では，対象サーバで障害が発生した際に，障害知識ベースに問い合わせることで解決方法として得られるコマンド群を仮想検証環境上で自動実行することで有効性を検証する．この検証結果を解決実績とし，障害知識ベースにフィードバックすることで記録されたコマンド群に優先順位を付ける．このサイクルにより，障害知識ベースは原因特定・解決に有用であるコマンド群の解決実績情報を継続的に更新し，優先的に障害発生サーバに

提供することができる．

本研究の特徴と貢献は下記のようにまとめられる．

- 障害発生・解決サイクルを体系的に構築することにより，半自動的な障害知識ベースの学習機構を実現している点
- 過去の障害解決情報をコマンドとして形式化し，仮想検証環境を適用することで部分的に自動的な障害原因分析機能を実現している点
- 障害解決可能な知識を解決実績を考慮して評価することにより，実際的な解決方法を優先的に提供できる点
- 上記の処理により，システム管理者に対しては障害解決分析に有用な情報を提供し，意思決定のための支援ができる点

実験で対象とする障害はディレクトリツリーの変更をともなうものである．具体的にはソフトウェアのデプロイミスや設定ファイル等の権限ミスを起因とした障害を対象として，障害環境を再現した実験環境を用いて提案手法の実現可能性を検証する．

2. 関連研究

システム障害・異常検知についての研究が行われている．永井らは障害発生装置の種類・構成要素と障害原因の対応を定義しルールベースで障害原因の推論を行った[6]．アプリケーションレイヤでは中村が Web アプリケーションにおける障害予測の早期化・高精度化のために Web ページのアクセス時間に対して統計的な方式を用いたアクセス時間解析方式を提案している[7]．Holub，Wang らは RTCE(Run-Time Correlation Engine)という分散システムログを収集しエラーとの相関を分析するエンジンを提案し，クラウド環境での評価を行っている[8] [9]．Xu らはコンソールログの文字列を解析し障害時に発生するログ情報を抽出する手法を提案している[10][11]．Mirgorodskiy らは複数の運用管理プロセスにおけるログを比較することで問題のある運用管理プロセスを特定する手法を提案している[12]．Diao らは運用管理者が記述したトラブルチケットに含まれる文字列と障害の原因を関連付けたルールに基づいて原因特定を行う手法を提案している[13]．工藤らはより少ない手間と短い時間でルールが構築できるルール記述方式とルール構築方法，そしてそれを効率的に精度良く実行するための解析ルールのデータ形式と解析実行方式を提案している[14]．阿部らはイベントネットワークのようなマルチベンダ機器で構成される特殊な環境において syslog メッセージの総量による分析を行い異常検知する手法を提案している[15]．また，障害の再発防止についての研究も行われている．西野らは熟練管理者の障害対処操作対応をルール化してそのノウハウを運用者全員が共有することで再発防止を行った[16]．また，障害の回避手段の提示に関する研究も行われている．加藤らは基本的な障害対処ルールを記述しておき複雑な障害に対してはそれらを基に障害回避方法を自動生成し運用者への提示を行った[17]．

既存研究ではシステムの異常検知や障害環境情報の自動収集によって管理者が障害解決を行う際の支援を行っている．しかしながら，実際に障害を解決するための操作コマンドの生成・実行といったシステム障害解決の自動化支援機能は実現されていない．この機能を実現するためには以下の 3 つの課題点があると考えられる．

1 つ目は障害原因特定・解決に用いられる過去の経験（解決実績）が個人の経験に依存しており，それを普遍的に共有化する方法や活用する仕組みがないという点である．管理者の技量には個人差があり，また，常に同じ管理者がすべての障害解決を行うわけではないので，どうしても知識の偏りが存在する．だからこそ知識の共有化を行わなければならないのだが，マニュアル等では対応できる幅に限界がある．このため，過去の障害対応知識を有する障害知識ベースの構築が必要であると考えられる．

2 つ目は，障害解決という作業が形式化されていないため，実際の障害環境で作業を行わないとその解決方法が障害解決に有効であるか判断できないという点である．通常管理者が障害原因の特定やその解決を行う際のアプローチとして，自身の経験や知識を基に原因の推測や解決のための方法を考案し実行する．このとき用いられる管理者の知識や経験は個人に依存しており，マニュアル等である程度知識共有されていても，最終的には個人の技量に依存してしまう．また，考案した解決方法の有効性は実際に障害環境で試行しないと判断できず，その知識も得ることができない問題がある．このため障害解決という作業を明確に形式化し，障害環境とほとんど同じような環境で実際に実行することでその有効性を確認する必要があると考えられる．

3 つ目は障害中の環境で無作為に複数の解決方法を試行しても，それが障害解決に有効ではなく，さらなる障害を引き起こしてしまう可能性がある点である．1 つの障害原因に対して 1 つの解決方法だけであるとは限らず，複数の解決方法がある可能性があるため管理者は複数の解決手段を考案し，それらを実行する必要がある．しかしながら，それらは無作為に実行してもその障害を直接解決できるものと解決できないものとがある．さらに最悪の場合，新たな障害を発生させる可能性もある．そのため，試行する際には過去に実行したことがあり，有効性に関してある程度実績がある

ものを優先的に試行する必要があると考えられる．そのため，試行する複数の解決方法には実行優先順位を付加する必要があると考えられる．

以上のことからシステム障害解決の自動化支援機能の実現は困難であるといえる．

3. 提案方式

本研究では，システム障害の効率的な解決を目的とした，複数サーバ環境での解決実績に基づいたシステム障害解決支援機構を提案する．提案方式は以下の 3 つの特徴を持つ．

1. 障害発生・解決サイクルを体系的に構築することにより半自動的な障害知識ベースの学習機構を実現する
2. 過去の障害解決情報をコマンドとして形式化し仮想検証環境を適用することで（部分的に）自動的な障害原因分析機能を実現する
3. 障害解決可能な知識を解決実績を考慮して評価することにより，実際的な解決方法を優先的に提供する

提案方式では過去の障害知識を普遍的な知識として活用可能であり，共有可能なものとするために障害知識ベースを構築する．この障害知識ベースには過去に発生した障害のエラーログや管理者が実行したコマンド，人間が手動で解決できるものか否かを表すフラグ等が記録され，将来の障害に対応するための知識を記録する．障害知識ベースはプライマリ障害知識ベースと管理対象の複数サーバ上で稼働するセカンダリ障害知識ベースから構成される．

- プライマリ障害知識ベース：解決した過去の障害時のシステムログと解決のために実行したコマンド群を記録する．
- セカンダリ障害知識ベース：プライマリ障害知識ベースから得たコマンド群を仮想検証環境上で実環境に影響を与えずに実際に実行し，実行結果を保持する．

セカンダリ障害知識ベースは，管理対象サーバ上で障害が発生した際，プライマリ障害知識ベースから得たコマンド群について解決方法の有効性を検証する．この検証結果は解決実績としてプライマリ障害知識ベースにフィードバックされ，記録されたコマンド群の優先順位付けに用いられる．また，Union File System を用いて実環境ファイルシステム上に実環境に近い状態の仮想的なファイルシステムを構築し，この仮想検証環境上でファイルシステムの変更をともなうシステム障害の解決作業を実行する．これによって管理者はほぼ実環境と同じ環境で，実環境に影響を与えること無く障害解決に必要な調査・検証を行うことが可能となる．このサイクルにより，プライマリ障害知識ベースは原因特定・解決に有用であるコマンド群の解決実績情報を継続的に更新し，優先的に障害発生サーバに提供することができる．図 1 に提案方式のシステム概要図を示す．

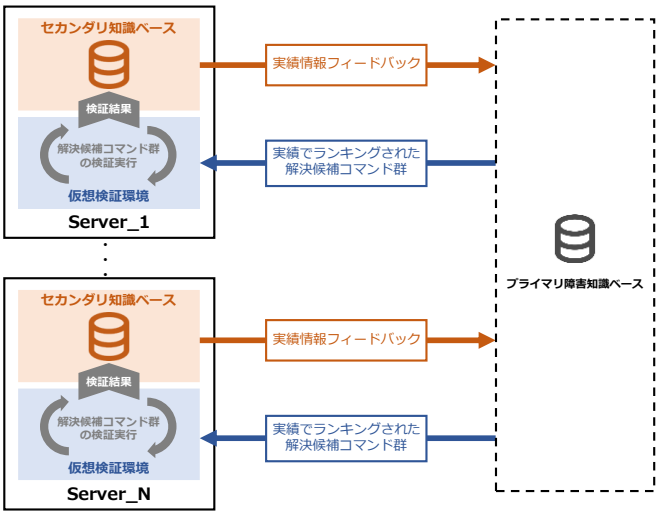

図 1 提案システムの概要

提案システムの実行手順は以下の通りである．

1. 過去のシステム障害時に出力されたシステムエラーログメッセージとその解決に使用されたコマンド群が記録されたプライマリ障害知識ベースを構築する．
2. 障害発生中のサーバ上に記録されたログメッセージをクエリにプライマリ障害知識ベース上で検索を行い，障害内容の解決に適した複数のコマンド群を優先度付きで抽出する．
3. 障害発生中の対象サーバ上で Union File System を用いて実環境ファイルシステム上に実環境に近い状態の仮想的なファイルシステムを構築する．
4. プライマリ障害知識ベースから検索結果として返ってきたコマンド群を自動的に仮想検証環境で実行する．
5. 検証結果をセカンダリ障害知識ベースに記録し，解決実績としてプライマリ障害知識ベースにフィードバックする．

3.1. 障害知識ベースの構築

過去にシステム障害が発生したソフトウェアのシステムエラーログメッセージとその障害を解決するために管理者が用いたシステムコマンドの対応付けを記録する障害知識ベースを構築する．

障害知識ベースを構築する際には，全文検索エンジンに対して，障害発生時に自動または手動で収集した

エラーログをキー，その障害を解決するために実行したコマンドを値として自動または手動で記録する．検索時に，エラーログをクエリに用いて検索を行うことで，発生中の障害に似た過去の障害事例が検索可能となる．検索結果として返ってくるコマンド群を発生中の障害を解決する可能性のあるコマンド群とし，仮想検証環境上で実際に実行する．表 1 にプライマリ障害知識ベースのスキーマ，表 2 にセカンダリ障害知識ベースのスキーマを示す．

表 1 プライマリ障害知識ベースのスキーマ

field	description
eid	データ ID
type	障害の種類 （e.g. disk,network,memory）
success_number	成功実績
failure_number	失敗実績
owner	障害が発生したサーバ名
log_messages	障害発生時のエラーログメッセージ
resolve_cmd	管理者が解決のために実行したコマンド

表 2 セカンダリ障害知識ベースのスキーマ

field	description
flag_auto	自動的に解決できたか
flag_human	人間が手動で解決できたか

システム障害には多くの種類があり，大まかに分類するとシステム的に自動的に解決できるものと自動的に解決できなくとも人間が手動で作業すれば解決できるものが存在すると考えられる．提案システムにおいてこれらを区別するために，セカンダリ障害知識ベースではその障害が自動的に解決できたか，もしくは人間が手動で解決できたかという知識を記録する．これを用いる利点として，管理者に対して有効な解決方法を提示する際，より適当な情報を提供できるという点がある．自動解決できる障害に対しては，障害知識ベースに蓄積された知識から直接的な解決方法を管理者に提示する．もしくは自動解決できない問題に対しては，試行され解決に失敗した方法を提示する．障害知識ベースには失敗した方法に対応した障害原因も記録されているので，障害原因の候補から除外することが可能となり，この失敗知識も障害原因推測に活用できると考えられる．

3.2. 仮想検証環境の構築

実環境上のファイルシステムを変更せずファイルシステムの状態を保ったままの仮想検証環境の構築には Union File System を用いた．Union File System とは Readable/Writable な Upper ディレクトリツリーと Readonly な Lower ディレクトリツリーという 2 つをマージし 1 つのディレクトリツリーとして見せるファイルシステムであり，ある時点のディレクトリツリーのスナップショットを維持したままいつでも破棄可能な変更を行うことを可能とする．これを用いて障害発生時のディスクの状態を Readonly のレイヤとして作成し，その上に Readable/Writable なレイヤ（仮想検証環境）を作成し，その環境上でコマンド群を試行する．例として，ファイルの権限操作を行う場合の手順を図 2 に示す．Step1 では Readonly な不正権限ファイルに対して正常な権限へ戻すための操作を行う．Step2 では Readonly なファイルを編集するために中間層に操作対象のファイルのコピーを行う．この時点で Union File System をマウントしたディレクトリツリーでは中間層にコピーされたファイルが見えている．最後の Step3 では中間層にコピーされた操作対象ファイルに対して Step1 で行われた権限操作が行われる．これによって最終的にユーザから見えるのはディレクトリツリー内に存在する正常な権限を持ったファイルとなり，権限操作が透過的に行われたように見える．

提案方式で構築する仮想検証環境は，実環境のディレクトリツリーのみ複製するため，実行中のプロセスやサーバ外部のネットワーク上で発生した障害は再現できないという点が実環境と異なる．このためこの仮想検証環境で再現可能となる障害はディレクトリツリーの変更に起因するものに限られる．例えば，特定のソフトウェアのパッケージ依存問題や操作権限を原因とした障害に対しては有効であるが，メモリリークなどの実行中のプロセスに起因するもの，サーバ外のネットワーク上での障害は再現することができない．

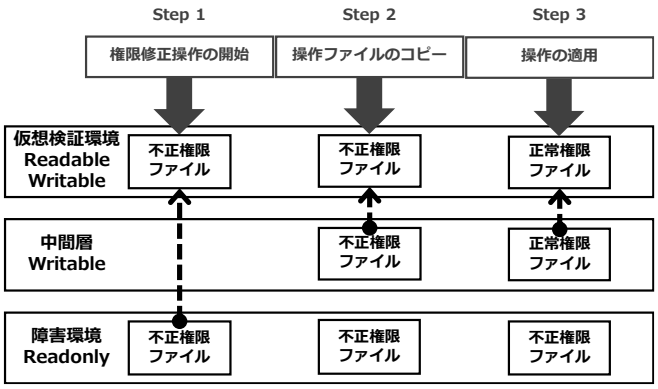

図 2 仮想検証環境の概要

3.3. 障害知識ベースからのコマンド抽出

解決に効果のあるコマンドを障害知識ベースから取得する際は，発生した障害のエラーログメッセージ *errMsg* をクエリとして障害知識ベースに入力し検索を行う．クエリとして入力した障害エラーログ，およ

び全ての過去の障害エラーログ $pastErrMsg_x$ との類似度は以下の類似度スコア計算式を用いて計算される．

$$score_{sim}(q,d) = coord(q,d) \times queryNorm(q) \times \sum_{t\ in\ q} \left(tf(t\ in\ d) \times idf(t)^2 \times boost(t) \times norm(t,d)\right)$$

類似度スコア $score_{sim}(errMsg,\ pastErrMsg_x)$の高い順に検索結果として，コマンド候補群を抽出する．次に早期に障害解決可能なコマンドを実行するために，エラーログによる類似度スコアだけでなく，障害知識ベースに記録された各解決候補コマンドの実績スコアによって仮想検証環境上で実行する際の優先順位を付ける．解決候補コマンドを $resolveCandCmd_x$，そのコマンドの成功実績を $successPerf_x$，失敗実績を $failurePerf_x$ としたとき実績スコア $score_{perf}$ は以下の式で求められる．

$$score_{perf}(resolveCandCmd_x) = successPerf_x - failurePerf_x$$

発生した障害 $failure_x$ に対して抽出された解決候補コマンド $resolveCandCmd_x$ の類似度スコア $score_{sim}$ および実績スコア $score_{perf}$ の高い順に検索結果として，仮想検証環境において実行するコマンド候補群を返す．

3.4. 仮想検証環境での実行

障害知識ベースから検索したコマンド群は，そのまま実行するだけで障害解決できるものと解決には至らないもののその過程に貢献できるものが混在する．コマンド 1 つごとに仮想検証環境を構築し，実際に実行することを自動的に繰り返すことで，どのコマンドが障害解決に効果があるかを検証する．

3.5. 解決実績のフィードバック

仮想検証環境上での各コマンドの実行結果は各サーバ環境における解決実績としてセカンダリ障害知識ベースに記録される．また，今後類似する障害が発生した場合優先的に問い合わせ結果となるようにプライマリ障害知識ベースに解決実績をフィードバックする．プライマリ障害知識ベースにエラーログおよび実行したコマンド内容とその実行が成功した場合には成功実績 *success* に+1 点，失敗した場合は失敗実績 *fail* に+1 点というように継続的に解決実績を更新する．このサイクルにより，障害知識ベースは原因特定・解決に有用であるコマンド群の解決実績情報を継続的に更新し，優先的に障害発生サーバに提供することができる．解決実績のフィードバックサイクルの流れを図 3 に示す．

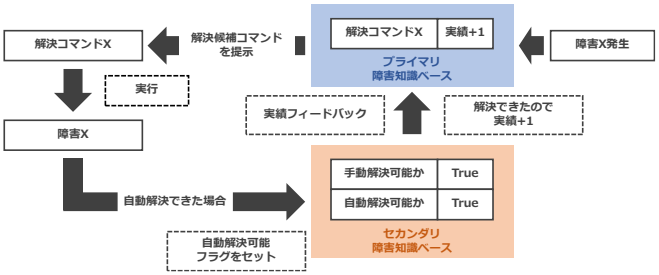

図 3 解決実績フィードバックサイクルの流れ

4. 実験

4.1. 実験目的

本実験では，提案方式を障害環境を再現した実験環境上で実行し，以下の点を確認することでその実現可能性を確認する．

- 対象サーバ上での解決実績を障害知識ベースへ継続的にフィードバックすること．
- 障害知識ベースにフィードバックされた解決実績を基に推薦コマンド群に優先順位が反映されること．

4.2. 実験環境

実験環境では OS として Linux (Ubuntu16.04)を使用した 3 台の管理対象サーバと 1 台の障害知識ベースを VirtualBox 上に仮想マシンとして構築した．管理対象サーバでは，仮想検証環境の構築のために Linux 上の Union File System である OverlayFS を使用する．障害知識ベースの構築には Elasticsearch を使用する

実験において障害知識ベースとして使用するデータを 20 件用意した．障害内容とその原因，解決に用いるコマンドの一部を表 3 に示す．また実験環境上で擬似的に障害を発生させるための障害原因コマンド一覧，その障害が解決したことを確認するための解決確認コマンド一覧，および障害時に発生したエラーログメッセージを，それぞれ表 4，表 5，表 6 に示す．

実現可能性の検証のために，表 7 のコマンドによって引き起こされる障害(実験データにおける eid=7)を想定した実験を行う．提案方式で実行される障害知識ベースへの問い合わせに用いる疑似障害によって出力されたエラーログメッセージを表 8 に示す．

表 3 実験に使用する障害データ

eid	障害内容 / 障害原因 障害解決コマンド	
1	https を用いるリポジトリからパッケージをダウンロードできない	パッケージマネージャが https 対応していない
	sudo apt install -y apt-transport-https	
2	Elasticsearch が起動しない	動作するために必要な JDK がインストールされていない
	sudo apt install -y openjdk-8-jdk	
3	プロキシ下でパッケージがダウンロードできない	http_proxy 環境変数の設定がされていない
	①echo '#!/bin/bash' > /etc/profile.d/proxy.bash ②echo export ¥ http_proxy=http://ccproxy.kanagawa-it.ac.jp:10080' ¥ >> /etc/profile.d/proxy.bash ③echo export 'https_proxy=$http_proxy' ¥ >> /etc/profile.d/proxy.bash ④echo ¥ Defaults env_keep+=¥"http_proxy https_proxy¥" ¥ >> /etc/sudoers	
4	MySQL が起動しない	データベースディレクトリ内での操作権限不足
	sudo chmod 755 /var/lib/mysql	
5	MySQL が起動しない	データベースディレクトリ内での操作権限不足
	sudo chown mysql.mysql /var/lib/mysql	
6	プロキシ下でパッケージがダウンロードできない	apt でのプロキシ環境下用の設定がされていない
	①echo 'Acquire::http::proxy ¥"http://ccproxya.kanagawa-it.ac.jp:10080/¥";' > /etc/apt/apt.conf.d/10proxy ②echo 'Acquire::https::proxy ¥"http://ccproxya.kanagawa-it.ac.jp:10080/¥";' >> /etc/apt/apt.conf.d/10proxy	
7	MySQL が起動しない	データベースディレクトリ内での操作権限不足
	①sudo chown mysql.mysql /var/lib/mysql ②sudo chmod 755 /var/lib/mysql	

表 4 障害原因コマンド

eid	障害原因コマンド
1	sudo apt remove -y apt-transport-https
2	sudo apt remove -y openjdk-8-jdk
3	①unset http_proxy ②unset https_proxy
4	sudo chmod 000 -R /var/lib/mysql
5	sudo chown root.root /var/lib/mysql
6	find /etc/apt/apt.conf.d/ -type f ¥ | xargs -I{} sed -i -r '/Acquire::https?::proxy.+/d' {}
7	①sudo chown root.root /var/lib/mysql ②sudo chmod 000 /var/lib/mysql

表 5 解決確認コマンド

eid	解決確認コマンド
1	sudo apt update
2	sudo systemctl start elasticsearch.service
3	sudo apt update
4	sudo systemctl restart mysql.service
5	sudo systemctl restart mysql.service
6	sudo apt update
7	sudo systemctl restart mysql.service

表 6 エラーログメッセージ

eid	エラーログメッセージ
1	E: The method driver /usr/lib/apt/methods/https could not be found. N: Is the package apt-transport-https installed? E: Failed to fetch https://artifacts.elastic.co/packages/6.x/apt/dists/stable/InRelease ...
2	Nov 28 05:56:17 ubuntu-xenial elasticsearch[11810]: could not find java; set JAVA_HOME or ensure java is in PATH
3	W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial/InRelease Could not resolve 'archive.ubuntu.com' W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial-updates/InRelease Could not resolve 'archive.ubuntu.com' W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial-backports/InRelease Could not resolve 'archive.ubuntu.com' ...
4	2019-01-08T16:23:12.907580Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable 2019-01-08T16:23:12.907658Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable 2019-01-08T16:23:12.907675Z 0 [ERROR] InnoDB: Plugin initialization aborted with error Generic error ...
5	2019-01-08T16:23:12.907580Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable 2019-01-08T16:23:12.907658Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be

	writable 2019-01-08T16:23:12.907675Z 0 [ERROR] InnoDB: Plugin initialization aborted with error Generic error ...
6	W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial/InRelease Could not resolve 'archive.ubuntu.com' W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial-updates/InRelease Could not resolve 'archive.ubuntu.com' W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/xenial-backports/InRelease Could not resolve 'archive.ubuntu.com' ...
7	2019-01-08T16:23:12.907580Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable 2019-01-08T16:23:12.907658Z 0 [ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable 2019-01-08T16:23:12.907675Z 0 [ERROR] InnoDB: Plugin initialization aborted with error Generic error ...

表 7 実験で実行した障害コマンド

①sudo chown root.root /var/lib/mysql ②sudo chmod 000 /var/lib/mysql

表 8 疑似障害により出力された エラーログメッセージ

[ERROR] InnoDB: The innodb_system data file 'ibdata1' must be writable

4.3. 実験方法

障害知識ベース上にある既知の障害を対象として，表 4 の障害原因コマンドを実行することにより，擬似的に障害を発生させる．障害が発生している状態で仮想検証環境の構築を行うことで障害環境を再現し，提案方式を用いて表 3 の内容とダミーデータを含む 20 件のデータを障害知識ベースとして構築し障害解決を行う．また，障害解決作業のあと表 5 の解決確認コマンドを実行することで，障害が解決されたかの判定を行いその結果を解決実績として障害知識ベースにフィードバックする．このとき障害知識ベース内の解決実績部分が初期値から更新されていることを確認することで，解決実績が障害知識ベースへフィードバックされていることを確認する．また，障害知識ベースへの問い合わせ結果が解決実績によって優先順位が付けられていることを確認する．

4.4. 実験結果

障害原因コマンドを実行したあと，提案方式を適用する．提案方式適用前の検索結果を表 9 に示す．また，提案方式を適用したあとの問い合わせ結果を表 10 に示す．

表 9 フィードバック前の障害知識ベース に対する問い合わせ結果

順位	eid	成功実績	失敗実績	検索スコア
1	14	0	0	11.22
2	4	0	0	7.93
3	6	0	0	7.93
4	20	0	0	7.93
5	3	0	0	5.86
6	11	0	0	5.86
7	17	0	0	0.36
8	0	0	0	0.34
9	9	0	0	0.34
10	19	0	0	0.34

表 10 フィードバック後の問い合わせ結果

順位	eid	成功実績	失敗実績	検索スコア
1	6	1	0	7.93
2	14	0	1	11.22
3	4	0	1	7.93
4	20	0	1	7.93
5	3	0	1	5.86
6	11	0	1	5.86
7	17	0	1	0.36
8	0	0	1	0.34
9	9	0	1	0.34
10	19	0	1	0.34

4.5. 実験考察

実験結果より表 9 と表 10 の比較から，障害知識ベースから提示された解決候補コマンド群の成功実績および失敗実績の値が更新されていることがわかる．これは，障害発生サーバ上において，仮想検証環境における解決候補コマンドの実行結果を解決実績情報として障害知識ベースへフィードバックしたためである．

また，実験結果よりフィードバック前後で eid=14 と eid=6 の順位が変動していることがわかる．eid=14 の順位が低下した理由として，失敗実績が更新されたためであると考えられる．提案方式適用前は表 9 より検索結果順位として eid=14 が 1 位として返ってきているため eid=14 が最初に検証実行されるが，検証結果として解決に失敗すると失敗実績が更新される．反対に eid=6 の順位が向上した理由として検証環境での障害解決に成功し成功実績が更新されたためである．この

ことから障害知識ベースへの問い合わせ結果が解決実績によって優先順位をつけることが可能であると確認できた．

これらのことから，提案方式によって障害知識ベースに記録されている解決候補コマンド群が現在の障害を解決可能かどうか検証し，その結果を解決実績情報として障害知識ベースにフィードバックするサイクルを継続的に行うことで，過去の実績から，より有効であるとされる解決候補コマンドを管理者に優先的に提案することが可能であると考えられる．

しかしながら，提案方式で自動解決できる障害はディレクトリツリーに関係するものに限られるためこれ以外のネットワークやプロセスに起因する障害に対応すること，また解決実績情報は解決できたかどうかの二値判定であるので部分的な障害解決に関する実績を反映することが今後の課題である．

5. まとめ

本研究では，システム障害の効率的な解決を目的とした，複数サーバ環境での解決実績情報に基づいたシステム障害解決支援機構を提案し，その実現可能性について検証した．

提案手法では，過去の障害知識を記録する障害知識ベースに現在の障害内容を問い合わせることで解決方法として得られるコマンド群を仮想検証環境上で自動実行し，その検証結果を解決実績として障害知識ベースにフィードバックすることで記録されたコマンド群に優先順位を付ける．このサイクルにより，障害知識ベースは原因特定・解決に有用であるコマンド群の解決実績情報を継続的に更新し，優先的に障害発生サーバに提供することができる．実験では Linux 上に構築した障害環境を再現した実験環境を用いて，障害知識ベースに記録された解決候補コマンド群を仮想検証環境上で実行し，その解決実績情報が障害知識ベースへ反映されることで検索結果の優先順位に影響を与えることが可能であると確認できた．

今後の課題としてネットワークやプロセス起因の障害への対応，部分的な障害解決実績の反映，より実際のサーバ稼働環境に近いデータによる評価が考えられる．

参 考 文 献

[1] “Amazon Web Services”, https://aws.amazon.com, 参照 Jan. 2019.

[2] “Google Cloud Platform”, https://cloud.google.com, 参照 Jan. 2019.

[3] “Microsoft Azure”, https://azure.microsoft.com, 参照 Jan. 2019.

[4] “Datadog”, https://www.datadoghq.com/, 参照 Jan. 2019.

[5] “New Relic”, https://newrelic.com/, 参照 Jan. 2019.

[6] 永井崇之，名倉正剛，”迅速な危機回復を目的とする大規模環境向け障害原因解析システム”, 情報処理学会論文誌，Vol.54，No.3，pp.1109-1119，2013

[7] 中村友洋，”Web アプリケーションの障害を予測する“アクセス時間解析方式”の提案”，情報処理学会論文誌コンピューティングシステム（ACS），Vol.47，No. SIG12(ACS15)，pp. 349-357，2006

[8] Holub，Viliam，et al.，”Run-time correlation engine for system monitoring and testing”，Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session，pp.9-18，2009

[9] Wang, Miao, et al.，”Scalable Run-Time Correlation Engine for Monitoring in a Cloud Computing Environment”，Engineering of Computer-Based Systems，IEEE International Conference on the(ECBS)，pp.29-38，2010

[10] Xu，Wei，et al.，”Online system problem detection by mining patterns of console logs”，Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on. IEEE，pp.588-597，2009

[11] Xu，Wei，et al.，”Detecting large-scale system problems by mining console logs”，Proceedings of the 27th International Conference on Machine Learning (ICML-10)，pp. 37-46，2010

[12] Mirgorodskiy，Alexander V.，et al.，”Problem diagnosis in large-scale computing environments”，Proceedings of the 2006 ACM/IEEE conference on Supercomputing，p.88，2006

[13] Diao，Yixin，et al.，”Rule-based problem classification in it service management”，2009 IEEE International Conference on Cloud Computing. IEEE，pp.221-228，2009

[14] 工藤裕，森村知弘，菅内公徳，増石哲也，薦田憲久，”障害原因解析のためのルール構築方法と解析実行方式”，電気学会論文誌 C（電子・情報・システム部門誌），Vol.132，No.10，pp.1689-1697，2012

[15] 阿部博，敷田幹文，篠田陽一，”イベントネットワークにおける syslog を用いた異常検知手法の提案と実データを用いた評価”，情報処理学会論文誌，Vol.59，No.3，pp. 1006-1015，2018

[16] 西野博之，坂下幸徳，敷田幹文，“大規模データセンターにおける運用ノウハウ共有による障害再発防止方式の提案”，インターネットと運用技術シンポジウム 2013 論文集，Vol.2013，pp.87-94，2013

[17] 加藤裕，敷田幹文，“障害予測における最適な障害回避手段の提示法”，インターネットと運用技術シンポジウム 2012 論文集，Vol. 2012，pp.110-116，2012