エラー原因の診断機能のモジュール化とワークフローによる設定手順の簡略化

平尾 真斗[1]　小山 智之[2]　串田 高幸[1]

概要：サーバーの監視は, システムを運用するために必要不可欠である．そのためサーバーの管理者は, 運用しているサーバーに合わせた監視方法を実装する．サーバーを監視する手法は, 既存のソフトウェアを用いる方法と自作の監視ソフトウェアを用いる方法がある．自作の監視ソフトウェアは, 監視に必要な IP アドレス, ドメイン, URL の値がハードコーディングされている．しかし, IP アドレス, ドメイン, URL の値をハードコーディングしている場合, 監視対象の追加や新しい監視方法を追加する際にプログラムされているソースコード書き換えが必要になる．監視するプログラムのソースコードが複数ある場合, ソースコードを一つ一つ直していくのは手間である．この解決策として監視のソフトウェアに対する監視機能のモジュール化を提案する．実験では，監視対象のサーバーが移行することを想定する．監視対象のサーバーが移行された場合，そのサーバーの IP アドレスは変更される．監視するプログラムのソースコードをハードコーディングしている場合と提案の設定ファイルを用いる場合では，この変更に対してプログラムのファイル内容の記述を変更し，保存する必要がある．実験評価では, 実験をもとに得られたそれぞれの保存回数を比較した．実験の結果, 本提案の設定ファイルを用いる方法では, 監視するプログラムのソースコードをハードコーディングしているファイルに比べてファイルの保存回数を 2 回減らすことができた．

1. はじめに

背景

アプリケーションやサービスの増大に伴いサーバーの数も増えている [1]．サーバーの監視は，稼働しているサーバーを継続的に監視し異常が見つかった際にサーバー管理者に通知するために行われる [2]．サーバーに対する監視を行わない場合, 不具合が起きても早期に検知することができない．異常個所を手作業で探すことは，非効率である．そのような点からサーバーの監視は, システムの運用を効率的に行う上で不可欠である [3]．サーバー管理者は, 機器に合わせた監視方法を用いることでサーバーの異常に迅速に気づくことができる．例えば，Web サーバーの監視には, HTTP リクエストを送信して HTTP レスポンスの結果を監視する外形監視が用いられる．監視方法の実装は以下の 2 種類の方法が用いられる．

(1) 既存のソフトウェアを用いる方法
(2) 自作の監視ソフトウェアを用いる方法

(1) の既存の監視ソフトウェアを用いる方法としては, Zabbix を用いる方法がある．Zabbix とは, オープンソース監視ソリューションの 1 つであり, ネットワークサービス, サーバー, ネットワークハードウェアの監視および追跡ができる [4]．Zabbix を用いる場合, 監視を行いたいマシンに Zabbix のエージェントをインストールして監視を行う．

(2) の自作の監視ソフトウェアを用いる方法では, 監視を行う環境と対象に合わせて監視ソフトウェアを構成する．しかし, 自作の監視ソフトウェアは, 監視対象を特定するために必要な IP アドレス, ドメイン, URL の値がプログラムのソースコード内にハードコーディングされている．

プログラム 1　ハードコーディングの例

```
1 import subprocess
2 subprocess.run(["ping","192.168.100.143"])
```

ハードコーディングとは, 変えられるようにするべき変数をプログラム内のソースコードに直接書くことである．ハードコーディングの例をプログラム 1 に示す．プログラム 1 は，死活監視を実行する．使用言語は Python である．1 行目は, Linux の subprocess をインポートしている．2 行目は, subprocess を用いて死活監視を行う．2 行目の 192.168.100.143 は, Web サーバーの IP アドレスである．

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学大学院バイオ・情報メディア研究科
〒 192–0982 東京都八王子市片倉町 1404-1

この場合, プログラム内のソースコードに Web サーバーの IP アドレスの値が入っておりハードコーディングされている．ハードコーディングで値を固定している場合, システムの仕様変更やのちのメンテナンス性に問題がある [5]. 例えば, プログラム 1 のようなハードコーディングされているプログラムのソースコードでは, 監視対象を変える際にプログラム内のソースコードの IP アドレスを変える必要がある [6].

課題

課題は, サーバーの異常を特定する機能をハードコーディングしている場合, 設定の変更には, プログラム内のソースコードの変更が必要になる．監視を行うプログラムのソースコードが複数ある場合, ソースコードを一つ一つ直していくのは手間である．例えば, GCP から AWS にサーバーを移行したときに監視対象の IP アドレスが変わるため, プログラム内のソースコードを変更する必要がある．そのためサーバーの管理者は, 展開しているサーバーに合わせて再度コーディングする必要がある．

各章の概要

2 章では, 関連する既存研究を紹介する．3 章では, 提案手法について説明する．4 章では, 実装と実験方法について説明する．5 章では, 評価及び分析の手法を説明する．6 章では, 提案手法の議論を行う．最後に 7 章でまとめを行う．

2. 関連研究

Giljong Yoo らは, 機能ベースでアスペクト指向プログラミングを使用した監視方法を提案している [7]．この手法では, 監視プロセスは，ユーザー側の値入力違反のチェックと監視後の監視機器の状態異常チェックに分けられる．まず, 監視実行前に渡される引数のチェックを行う．ユーザーが実際に入力した値に対してチェックを行い入力値に誤りがある場合に報告を行う．異常がない場合, 監視を実行し状態図を確認の結果をもとに異常状態を確認する．研究では, アスペクト指向プログラミングを用いた各機能のスコープ制約の構成と各機能に基づく状態図とモジュール機能関係リストを活用した障害検知を用いることで, システム全体の異常を特定している．しかしこの提案では, 多環境で使われることを考慮していない．したがって環境が固定されてしまう．

Wenxian Zeng らは, SNMP に基づくネットワークサーバー監視システムの設計と実装を提案している [8]．SNMP は, 監視機器の状態を要求する監視マネージャーと応答を返す監視エージェントに分かれている．監視エージェントは，監視マネージャーから情報要求に対し, MIB と呼ばれる機器情報の集合体を送る．これにより監視マネージャーは, 監視エージェントの状態を確認することができる．この提案では, 以下の 5 つのモジュールを用いて SNMP の異常検知を実現している．

(1) データ取得モジュール

SNMP のやり取りによって得られた MIB の値を取得する．

(2) データ分析モジュール

取得した MIB の値をユーザーが読みやすいデータ形式に変換し, データベースに保存する．

(3) データベースモジュール

サーバーのリスト, データベースの読み取りと書き込みの応答を格納し, 他のモジュールに統一されたデータベースインターフェイスを提供する．

(4) 故障診断モジュール

警告処理, 故障検出, 故障位置特定を行う．

(5) 統計表示モジュール

データベースに記録されたデータを処理し, 結果をグラフ形式で表示する．

しかし, この提案を多環境の監視に適応した場合, SNMP を用いて監視を行うため SNMP を用いていない箇所に異常が出ていた場合検知ができない.

Wenxian Zeng らは, SNMP に基づくサーバー監視システムの設計と実装を提案している [9]．この手法では, システムをデータ層, サービス層, 機能層の 3 つの層で設計している．データ層では, 監視マネージャーと監視エージェント間の通信を担当し，MIB の情報を要求し, 設定する。また, この層は, サービス層に収集したデータを送る役割がある．データの収集では, リアルタイム収集とタイミング収集がある．

(1) リアルタイム収集では, 監視エージェントから送られてきたリアルタイムな情報を収集し, 上位層に送信する．
(2) タイミング収集では, ある時間間隔で定期的に情報を収集し, 上位層に送信する．

サービス層は, 収集したデータを機能層に送信もしくは, データベースに格納する．機能層は, データ層とサービス層から送られてきた MIB の情報をもとに, システム管理者へのアラートとグラフの表示を行う．また, MIB のカスタマイズを行うことによって拡張性の高いグラフを作成することができる．しかし, この提案では, 何秒おきに監視を行うか, アラートの回数, タイムアウトの時間を, システム管理者が変えられるようにしていない.

3. 提案

提案方式

本稿では, サーバー監視のソフトウェアに対する監視機能のモジュール化を提案する．図 1 は, 実行ソフトウェアがシステム管理者からの設定ファイルの値をもとに監視を実行するまでの流れを示している．

システム管理者は, 実行ソフトウェアが監視に必要な設

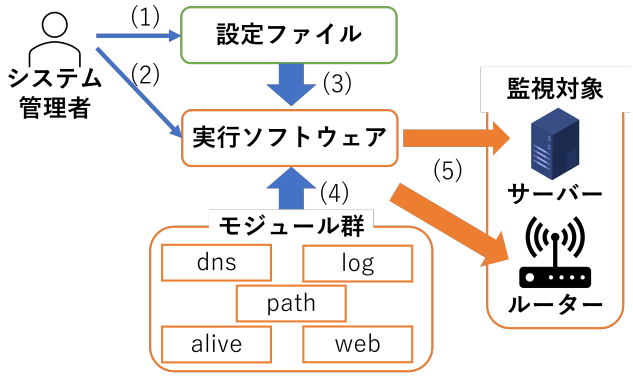

図 1 提案方式

定を設定ファイルの記載を用いて行う．設定ファイルには，監視を行う際のモジュールの記述，監視パラメータ，監視条件を記述する．実行ソフトウェアは，設定ファイルのモジュールの記述，監視パラメータ，監視条件をもとに監視を行う．設定ファイルにモジュールの記述，監視パラメータ，監視条件の記載することでプログラムのソースコードに直接値を書くハードコーディングを防ぐ．監視対象は，実行ソフトウェアに現在の機器の状態を送る．実際にシステム管理者が監視を行うまでの流れを説明していく．

(1) システム管理者が設定ファイルに監視に必要なパラメータや条件を選択していることを示している．
(2) システム管理者が実行ソフトウェアの実行を行っていることを示している．
(3) 実行ソフトウェアが設定ファイルから受け取ったモジュール，各パラメータの設定，監視条件の情報を受け取っていることを示している．
(4) 実行ソフトウェアは，設定ファイルの情報をもとにモジュールを選択していく．
(5) 実行ソフトウェアは，監視対象に向けて監視を行う．

設定ファイルでの既存のモジュール選択と監視条件の選択

プログラム 2 設定ファイル

```
1 module: #ここのモジュールの設定を記述
2   dns: #名前解決
3     dnscount: 3 #名前解決を何回行うか
4     domainname: manato-log #ドメイン名
5 conditions: |- #条件記載部
```

既存のモジュールの選択と監視パラメータは，設定ファイルに記載する．モジュールの構造は設定ファイル内のmodule の下にモジュールの名前がある．その下にモジュールにわたされる各パラメータがある．記載の仕方はプログラム 2 のように行う．例えば，名前解決を行いたい場合，2 行目にある dns モジュールの下のパラメータを選択する．

dns モジュールのパラメータは，3 行目から 4 行目で示している．3 行目の dnscount は，名前解決を一回につき何回行うかを示す．4 行目の domain は，名前解決時に使用されるドメインである．もし dns モジュールを使用しない場合，module 内の dns と各パラメータをコメントアウトもしくは削除すればよい．監視条件の記載は，module 部の下にある conditions 部で行う．ここでどのモジュールを使用するかやどの条件で監視を組み合わせるかを記載する．既存の監視モジュールは，以下の 5 つである．

(1) alive モジュール
(2) web モジュール
(3) path モジュール
(4) dns モジュール
(5) log モジュール

alive モジュール

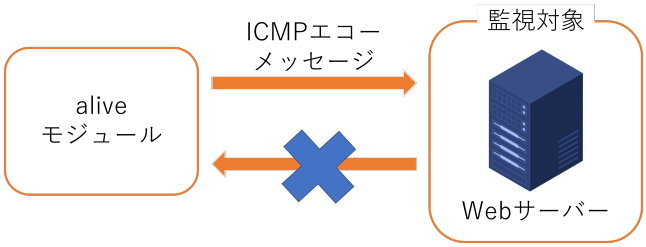

図 2 alive モジュール

alive モジュールでは，死活監視を行う．具体的な監視方法については図 2 で示す．死活監視では，Web サーバーに対して ICMP エコーメッセージを送る．ICMP エコーメッセージの応答がない場合，異常と判定を行う．コマンドは，Ping コマンドを使用する．

web モジュール

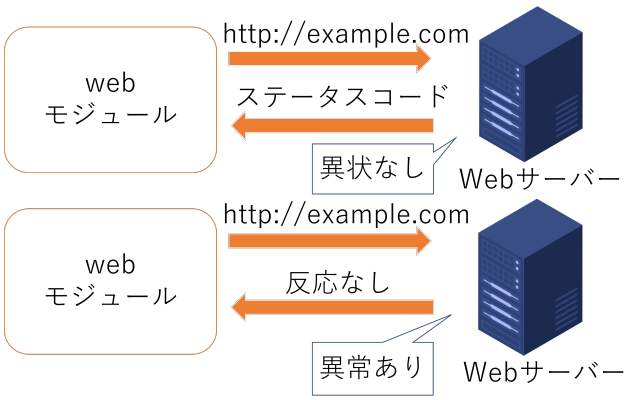

図 3 web モジュール

web モジュールでは，外形監視を行う．具体的な方法については，図 3 で示す．Web サーバーは，HTTP リクエストに対して HTML，CSS，JavaScript を提供する．外形監視では，対象サーバーに対して HTTP リクエストを送り

応答がない場合, 異常と判定する．外形監視コマンドには, curl コマンドを利用する．このコマンドは, HTTP リクエストを Web サーバーに送るコマンドであり, 応答内容には，HTTP リクエストの応答に対するステータスコードの内容が含まれる．

path モジュール

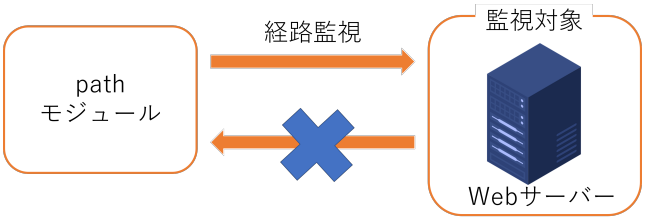

図 4　path モジュール

path モジュールでは, 監視サーバーと対象サーバー間の経路監視をする．具体的な方法は, 図 4 に示す．経路監視では, Web サーバーに対して ICMP エコーメッセージを送る．ICMP エコーメッセージが中間機器か Web サーバーに対して届いている場合, 正常と判断し届いていない場合異常と判断する．コマンドは, Ping -R コマンドを使用している．これにより中継機器の状態も監視できる．

dns モジュール

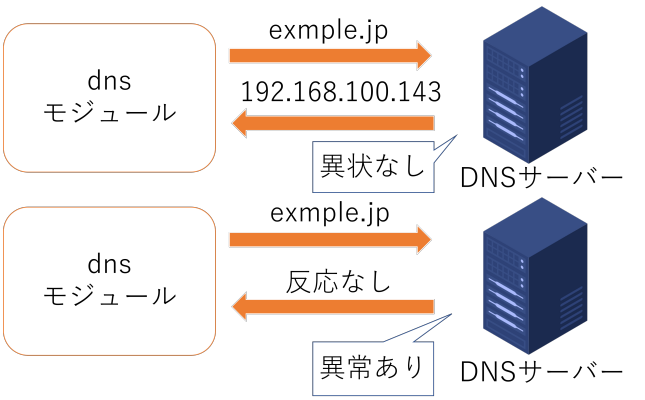

図 5　dns モジュール

dns モジュールでは, DNS サーバーに対して名前解決を行うことを想定する．図 5 に dns モジュールの具体例を示す．DNS サーバーは, URL 内に含まれるドメインを IP アドレスに変換する役割を持っている．管理者が設定したドメインに対して IP アドレスを返せるかどうかで異常判定をする．ドメインに対して IP アドレスが変換されている場合, 正常と判断し IP アドレスに変換されないか DNS サーバーにアクセスできない場合は, 異常と判断する．コマンドは，nslookup コマンドを使用する．

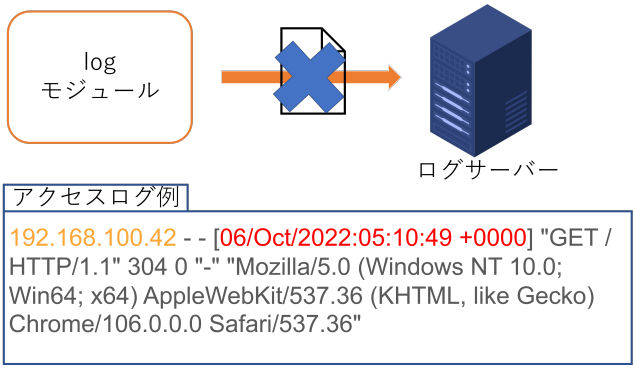

図 6　log モジュール

log モジュール

log モジュールでは, ログの件数を監視する．log モジュールの監視方法とログの例は，図 6 に示す．ログサーバーは, ログを保存する．監視サーバーは，ログサーバーに保存されているログの件数を監視しログ更新が行われない場合, 異常と判断する．ログとは, コンピュータの利用状況やデータ通信など履歴や情報の記録である．ログは, システムやサーバーの種類によって種類が異なる．例えば, Web サーバーにアクセスがあった際に記録されるのがアクセスログである．アクセスログ内には，Web サーバーに対してアクセスを行った端末の IP アドレスや, ステータスコード, アクセスしてきた端末の情報が記録される．

モジュールの追加

モジュールの追加を行いたい場合, 設定ファイル内のモジュールの追加とモジュールの作成を行う．図 7 にモジュールの追加とモジュール作成の流れを示していく．

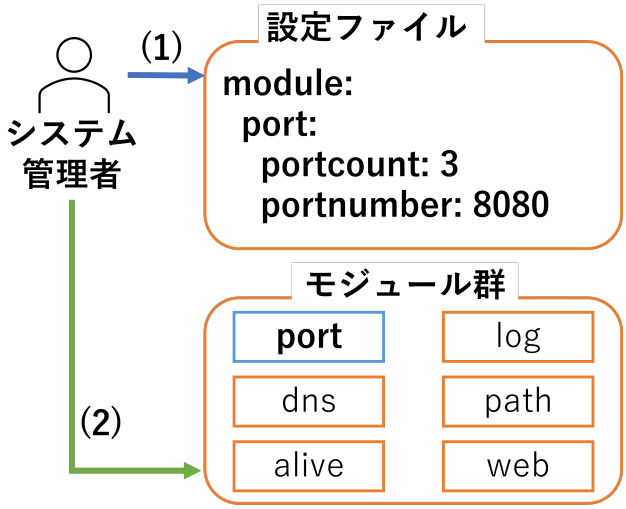

図 7　モジュールの追加

(1) システム管理者は，新しい監視モジュールである port モジュールを設定ファイルに追加する．その際, port モジュールの下には，監視に必要なパラメータを選択する．module の下にある port は, port モジュールを示している．portcount は, ポート監視を何回行うのか

を示している．portnumber は，ポート番号を示している．port モジュールは，監視対象サーバーの port を監視する．

(2) port モジュールを作成する．port モジュールには，設定ファイル内で port モジュールに記載した各パラメータが引数として渡されていくように記載していく．

ユースケース・シナリオ

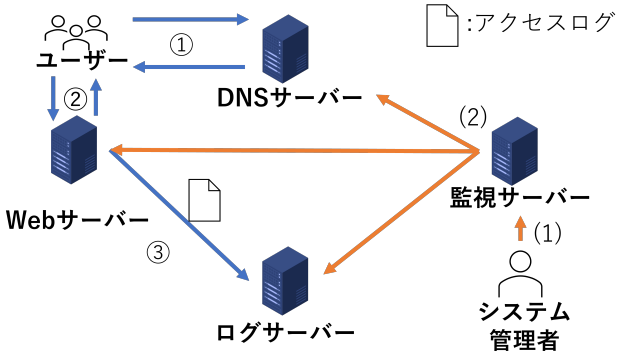

図 8 ユースケース

本ユースケースシナリオでは，ファッション EC サイトを想定している．EC サイトのシステム管理者が監視サーバーを用いて異常個所の特定を行う．ユースケースシナリオを図 8 に示す．ユーザーは，ファッション EC サイトを使い服を購入しようとしている．サーバーは，DNS サーバー，Web サーバー，ログサーバー，監視サーバーがある．①から③は，ユーザーが EC サイトのサーバーにアクセスする流れを示している．

①は，ユーザーが Web サイトの閲覧に必要な IP アドレスを DNS サーバーに問い合わせることを示している．DNS サーバーは，URL に含まれるドメインを IP アドレスに変換する役割がある．

②は，ユーザーが①で取得した IP アドレスをもとに Web サーバーにアクセスを行っていることを示している．Web サーバーは，ユーザーのアクセスに対して HTML，CSS，JavaScript を返す．

③は，Web サーバーがアクセスログをログサーバーに送ることを示している．監視方法は，サーバーによって異なる．例えば，Web サーバーに対する監視方法は，外形監視を用いる．(1) と (2) では，システム管理者が監視サーバーに対してどのような監視方法を選択するかを示している．

(1) システム管理者は，監視サーバーに対して監視対象，監視タイミング，監視方法の選択を行う．

(2) 監視サーバーは，システム管理者が設定した監視条件をもとにサーバーに対する監視を行う．

4. 実装

図 9 に設定ファイルからモジュール群の選択の流れを示す．構成は，大きく設定ファイル，実行ソフトウェア，モジュール群の 3 つに分けられる．このうち監視を行う際のモジュールの記述，監視パラメータ，監視条件を記載する設定ファイルは，プログラミング言語である YAML で実装されている．設定ファイルのから渡されたパラメータをもとに監視モジュールの選択と監視を行う実行ソフトウェアとモジュール群は，プログラミング言語である Python で実装されている．各構成要素の説明を (1) から (6) で説明していく．

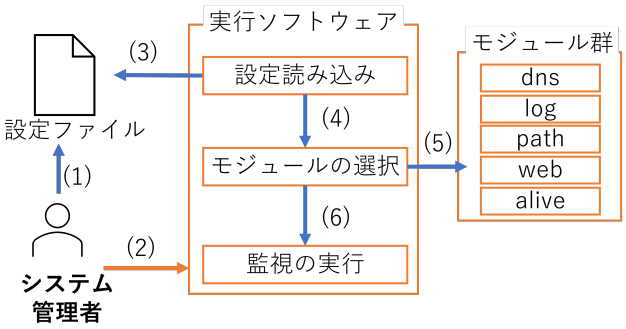

図 9 設定ファイルからモジュール群の選択の流れ

(1) システム管理者が，設定ファイルにモジュールの記述，監視パラメータ，監視条件が記述を行う．

(2) システム管理者は，実行ソフトウェアを実行する．

(3) 実行ソフトウェアが設定ファイルのモジュール，監視パラメータ，監視条件を読み取る．

(4) 実行ソフトウェアが設定ファイルをもとにモジュールの選択を行う．

(5) 実行ソフトウェアが監視パラメータをモジュールに入れていく．

(6) 実行ソフトウェアが監視を実行する．

設定ファイル

システム管理者が監視モジュールの選択，パラメータの設定，監視条件の記載を行う設定ファイルはプログラム 3 のようになっている．設定ファイルは，大きく 2 つに分かれている．1 行目から 19 行目は，module 部である．ここでは，モジュールとモジュールのパラメータを記述する．各モジュールは，2 行目のように記載を行う．モジュールに対応する監視パラメータは，3 行目から 4 行目のように記載を行う．例えば，dns モジュールを使う場合，2 行目のように dns 書き，3 行目と 4 行目で dns モジュールのパラメータを記載する．dns モジュール内の dnscount は，名前解決を何回行うかを示している．domainname は，ドメインを示している．

21 行目から 23 行目は，conditions 部である．ここでは，監視条件の記載を行う．システム管理者は，監視条件を Python の文法で記述する．このようにすることで複数の条件を記述できるようにしている．モジュールの呼び出し方は，22 行目のように modulelist[番号].main(**list[番号])

の形式で行う．監視モジュールは，この番号を使い分けることで呼び出しを行う．番号は，0 から始まり dns モジュールは，0 番目，web モジュールは，1 番目，log モジュールは，2 番目，alive モジュールは，3 番目，path モジュールは，4 番目のように順番となっている．プログラム 3 の 22 行目から 23 行目では，実際に dns モジュールと web モジュールを呼び出している．最初に dns モジュールが呼び出され結果が失敗だった場合，web モジュールが呼び出される．

プログラム 3　設定ファイル

```
1  module: #このモジュールの設定を記述
2    dns: #名前解決
3      dnscount: 3 #名前解決を何回行うか
4      domainname: manato-log #ドメイン名
5    web: #外形監視
6      webcount: 3 #外形監視を何回行うか
7      weburl: http://192.168.100.143/
       #外形監視用URL
8    log: #ログ監視
9      logcount: 3 #ログ監視を何回行うか
10     logpath: /var/log/access.log
       #ログファイルのパス
11   alive: #死活監視
12     alivecount: 3 #死活監視を何回行うか
13     aliveserver_ipaddr:
       #死活監視を行いたいIPアドレス
14     - 192.168.100.143 #IPアドレス
15   path: #経路監視
16     pathcount: 3 #経路監視を何回行うか
17     pathserveripaddr:
       #経路監視を行いたいIPアドレス
18     - 192.168.100.143 #IPアドレス
19     - 192.168.150.10
20
21 conditions: |- #条件記載部
22   if modulelist[0].main(**list[0]) ==
     "失敗":
23       modulelist[1].main(**list[1])
```

5.　実験と分析

実験環境

実験環境を図 10 に示す．実験環境では，2 つのサーバーを使用する．Web サーバーは，HTTP リクエストに対して応答を行う．Web サーバーは，Nginx を使用している．Nginx は，オープンソースで軽量かつ高性能なサーバーとして，その高性能と拡張の容易さから近年広く利用されている [10]．監視サーバーには，提案の設定ファイルとモジュールが入っている．モジュールは，web モジュール，alive モジュール，path モジュール，log モジュール，dns モジュールが入っている状態とする．

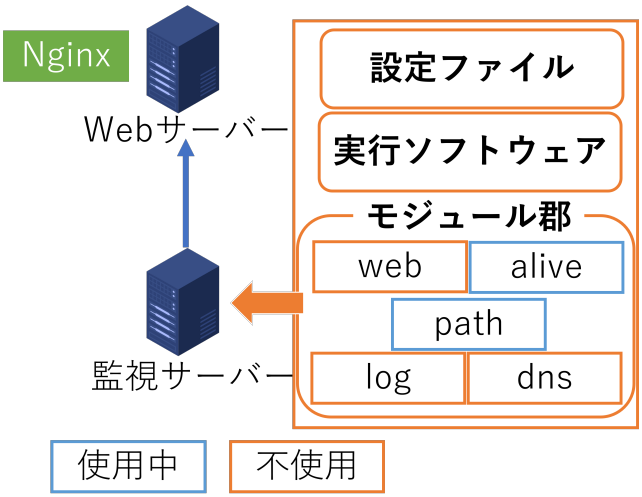

図 10　実験環境

実験シナリオ

シナリオは，EC サイトで Web サーバーの移行を行うことを想定している．Web サーバーの移行前には，監視サーバーは，Web サーバーに対して死活監視と経路監視を行っていた．Web サーバーの移行を行うと IP アドレスが変わる．そのため死活監視と経路監視に必要な IP アドレスの値を変える必要がある．

実験方法

実験方法の流れを図 11 に示す．実験では，EC サイトで Web サーバーの移行を行うことを想定する．その際，監視サーバー内に事前に入っている提案の設定ファイルとモジュールを使用する．実験では，既存モジュールである alive モジュールと path モジュールを使用する．また，比較対象としてハードコーディングしている場合のファイルと実行ファイルを用意した．ハードコーディングしている場合の Ping.py は，死活監視を行うファイルであり，Path.py は，経路監視を行うファイルである．各ファイルには，監

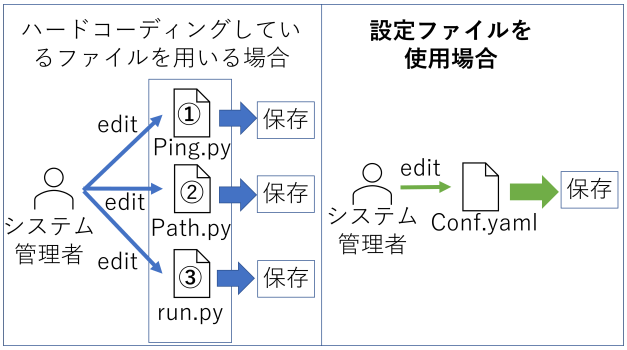

図 11　実験方法

視に必要なパラメータの記載を行う．例えば，死活監視と経路監視を行うファイルを使用する場合，監視回数と Web サーバーの IP アドレスが必要となる．run.py は，実行ソフトウェアを示している．実行ソフトウェアでは，監視条件の記載を行う．設定ファイルを使用する場合の Conf.yaml は，提案の設定ファイルを示している．設定ファイルでは，モジュールに必要な引数の指定と監視条件の記載を行う．ハードコーディングしているファイルを用いる場合の①から③ は，システム管理者がファイルの書き込みと保存を行うファイルの順番を示している．実験方法は，監視機能をハードコーディングしているファイルを使用する場合と提案の設定ファイルを記述してモジュールを使用する場合のファイルの保存回数を比較する．

実験評価と分析

実験の結果は，図 12 となった．縦軸は，ファイルの保存

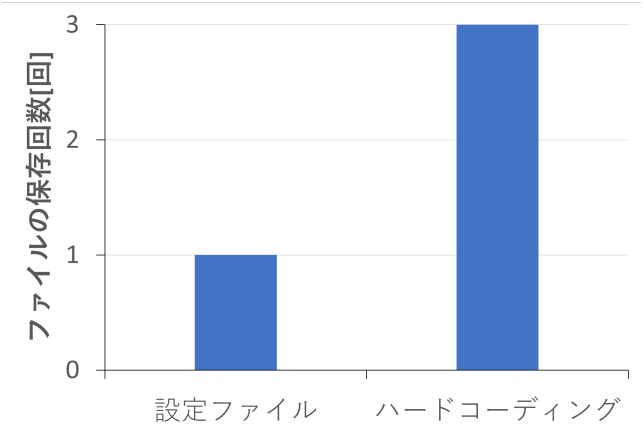

図 12 実験結果

回数を示しており単位は，回である．横軸の設定ファイルは，提案の設定ファイルを示している．ハードコーディングは，監視機能をハードコーディングしているファイルを示している．監視機能をハードコーディングしているファイルを用いる場合は，ファイルの保存が 3 回となった．一方，設定ファイルを用いる場合は，1 回となった．監視機能をハードコーディングしているファイルを用いる場合にくらべて設定ファイルを用いる場合では，ファイルの保存の回数を 2 回下げることができた．また設定ファイルを用いる場合のほうがハードコーディングしているファイルを用いる場合に比べてファイルの保存の回数が少ない理由として，モジュールの選択，パラメータの記載，監視条件を一つの設定ファイルにまとめたことが挙げられる．一方，監視機能をハードコーディングしているファイルを用いる場合，本実験のように新しい監視の追加や環境の変更が起きた際，各モジュールに対する保存と実行ソフトウェアに対する保存が必要となる．そのため保存の回数が多くなる．

6. 議論

本稿では，サーバー監視のソフトウェアに対する監視機能のモジュール化を提案した．この提案で新たにモジュールの追加を行う際，ローカル上で管理している．しかしローカル上でプログラムのソースコードを管理している場合，管理する際にローカル上にあるプログラムのソースコードのみが最新版となってしまう．そのため GitHub 上でプログラムのソースコードをアップロードしておけば最新版のソースコードをほかのユーザーが再利用したりモジュールの共同開発ができる．

実験では，監視モジュールをハードコーディングしている場合と提案の設定ファイルを用いる場合の保存の回数を比較した．結果的に，提案の設定ファイルを用いる方法のほうがハードコーディングしているファイルを用いる場合に比べて保存の回数が 2 回少なかった．しかしハードコーディングしているプログラムのソースコードのファイルが一つのファイルにまとめた場合，保存の回数は，1 回となる．そのため設定ファイルを用いる場合と保存の回数が変わらない．新しい評価方法として書きこまれたプログラムのソースコードをコンパイルするのにかかる時間を計測する．今回作成したモジュール群と設定ファイルは，プログラム言語として Python を用いているがコンパイルが必要になる C 言語では，プログラムのソースコードを変更するたびにコンパイルが必要になる．C 言語でプログラムのソースコードを書いていた場合，提案の設定ファイルを用いればコンパイル数を 0 にできる．よってハードコーディングしているソースコードのファイルを 1 つにまとめた場合よりもコンパイルにかかる時間を評価できる．

7. おわりに

課題は，異常の原因を特定する機能をハードコーディングしている場合，設定の変更には，プログラム内のソースコードの変更が必要になることである．提案では，サーバー監視のソフトウェアに対する監視機能のモジュール化を提案した．実験では，監視モジュールと監視条件の追加を提案の設定ファイルを用いて行った．実行ソフトウェアは，設定ファイルを読み取り監視モジュールの選択と監視対象に対する監視を行う．実験評価では，死活監視と経路監視の追加を本稿で示した提案の設定ファイルで追加した場合と監視機能をハードコーディングしているファイルで追加した場合で比較した．

謝辞 本テクニカルレポートの執筆にあたりご助言を賜りました東京工科大学大学院バイオ・情報メディアコンピュータサイエンス専攻の大野 有樹さん，東京工科大学コンピュータサイエンス学部の増田 和範さんに感謝申し上げます．

参考文献

[1] Kim, S.-T., Park, H.-J. and Kim, Y.-C.: The load monitoring of Web server using mobile agent, *2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479)*, Vol. 5, pp. 89–94 vol.5 (online), DOI: 10.1109/ICII.2001.983500 (2001).

[2] Renita, J. and Elizabeth, N. E.: Network's server monitoring and analysis using Nagios, *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 1904–1909 (online), DOI: 10.1109/WiSPNET.2017.8300092 (2017).

[3] Wang, Z., Wang, Y., Shao, G. and Guo, Z.: Research and Development of Monitoring System for Network Servers, *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–3 (online), DOI: 10.1109/WiCom.2008.801 (2008).

[4] Mardiyono, A., Sholihah, W. and Hakim, F.: Mobile-based Network Monitoring System Using Zabbix and Telegram, *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, pp. 473–477 (online), DOI: 10.1109/IC2IE50715.2020.9274582 (2020).

[5] Dustin, E.: *Effective software testing: 50 specific ways to improve your testing*, Pearson Education India (2003).

[6] Agrawal, D., Giles, J., Lee, K.-W., Voruganti, K. and Filali-Adib, K.: Policy-based validation of SAN configuration, *Proceedings. Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004. POLICY 2004.*, pp. 77–86 (online), DOI: 10.1109/POLICY.2004.1309152 (2004).

[7] Yoo, G. and Lee, E.: Monitoring methodology using Aspect Oriented Programming in functional based system, *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, Vol. 1, pp. 783–786 (2010).

[8] Wang, Z.-q., Wang, Y. and Shao, G.-q.: Research and Design of Network Servers Monitoring System Based on SNMP, *2009 First International Workshop on Education Technology and Computer Science*, Vol. 3, pp. 857–860 (online), DOI: 10.1109/ETCS.2009.728 (2009).

[9] Zeng, W. and Wang, Y.: Design and Implementation of Server Monitoring System Based on SNMP, *2009 International Joint Conference on Artificial Intelligence*, pp. 680–682 (online), DOI: 10.1109/JCAI.2009.34 (2009).

[10] Qin, E., Wang, Y., Yuan, L. and Zhong, Y.: Research on Nginx Dynamic Load Balancing Algorithm, *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 620–624 (online), DOI: 10.1109/ICMTMA50254.2020.00138 (2020).