L-005

# ログ解析による Web サービスの稼働監視方式

## Web Service Monitoring Method by Analyzing Logs

砂田 英之†　　山田 耕一†
Hideyuki SUNADA　　Kouichi YAMADA

## １．はじめに

近年、システムの SOA 化に伴い、システムを構成する Web サービス間の依存関係が複雑化してきている。１つのサービス障害が複数の業務アプリケーションに影響を与える場合があり、障害時の原因箇所や影響範囲の特定を行うためにサービスレベルでの監視が求められている。

従来方式ではプラットフォームである SOAP 基盤にて JMX(Java Management eXtensions)のインタフェースを用いた監視や CBR(Content Based Routing)等の技術により、メッセージのバリデーションチェックによる監視が可能である。

しかし、他社の提供するサービスを監視することができない。または、既存システムの改修やルーティング定義の変更など、既存システムへの影響が課題となっていた。

本論文では、これらの課題に対して、サービス間の呼び出しを仲介する ESB(Enterprise Service Bus)のログを解析することで、障害を検出する方式について提案する。

## ２．従来技術

(1)SOAP 基盤による監視機能

以下に JMX にて監視できる代表的な項目を示す。[1]

表１　JMX による監視項目

| 監視項目 | 内容 |
|---|---|
| Info | 現在のスレッド数 |
| | スレッドのピーク数 |
| | スレッドの処理時間 |
| | スレッドの ID, 名称、状態 |
| | スタックトレース |
| | デッドロックの有無 |
| Property | 属性値のセット／ゲット |
| Operation | メソッド操作 |
| Notification | 境界値（最大、最小値） |
| | カウンタの一定以上の変化 |
| | 文字列の変更の監視 |

Web サービスを JMX 準拠にて作成していた場合、SOAP 基盤の機能と連携して上記の項目について計測を行うことができ、スレッドの処理時間や、デッドロックの有無などから障害発生の有無を推測することが可能である。

また、ルーティング定義を変更して SOAP メッセージのバリデーションを追加することができる。

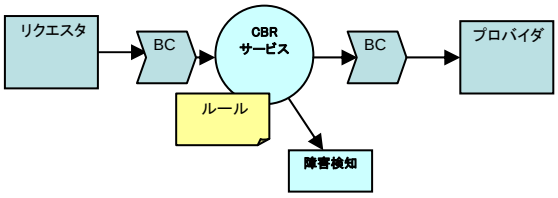


図１　障害検知処理野追加

(2) 従来方式における課題

従来方式では JMX による監視や、ルーティング情報を変更して監視機能を追加することで、サービスの稼働監視を可能にする方式が示されている。

しかし、これらの監視を実現するためには既存 Web サービスを JMX 対応に改修や、既存システムの設定を変更する必要があり、監視機能の導入の弊害となっている。

また、サービスには自社だけではなく他社の提供するプラットフォーム上で動作するサービスを利用する場合もあり、他社の提供サービスで障害が発生した場合には、本方式では監視することができず課題となっている。

## ３．解決策

(1)ESB ログの解析による検出

自社／他社のサービスを監視し、既存システムに対して影響を与えない方式として、サービスの呼び出しを仲介する ESB のログに着目した。

幾つかの ESB 製品を調査したところ、ログ情報には、SOAP メッセージを送受信した時刻、サービスの識別子、セッション情報、業務アプリ識別子、SOAP メッセージなどの情報が含まれることがわかった。

このことから、障害を検知するための障害検知ルールを取り決めて評価を行うとともに、プロセス定義を管理し、ログから算出できるサービスの実行順序とプロセス定義の処理フローとを比較することで、以下の障害を検出できることがわかった。

表２　ログ解析により検知できるサービス障害

| 種類 | 例 |
|---|---|
| タイミング不正 | 応答が遅い |
| 値の不正 | 境界値、フォーマット不正 |
| サービスのエラー（内部処理エラー） | エラーコード、メッセージなどを含む |
| シーケンスのエラー | プロセス定義から逸脱した呼び出しが発生した |

†三菱電機株式会社　情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

(2)実現方式

ESB のログ情報を解析し、障害を検知するための構成を以下に示す。

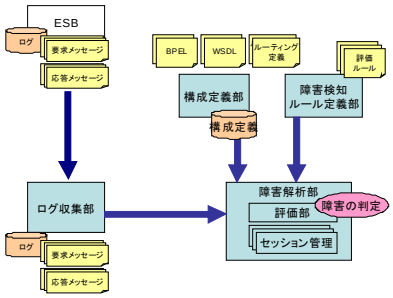


図２　システム構成

ESB からログ情報を収集するログ収集部と、プロセス定義、ルーティング定義などから Web サービスの構成情報を管理する構成定義部、サービス障害を検知するためのルールを設定するルール定義部、そしてログ情報、構成情報、障害検知ルールからサービス障害の発生有無を評価する障害解析部から構成される。障害解析部は、プロセス定義を実行するセッション毎に呼び出し順序や SOAP メッセージ情報を管理するセッション管理部と、ルールの評価を行う評価部とから構成される。

以下に、障害検知の処理の中心となるセッション管理と評価部で実行する処理を示す。

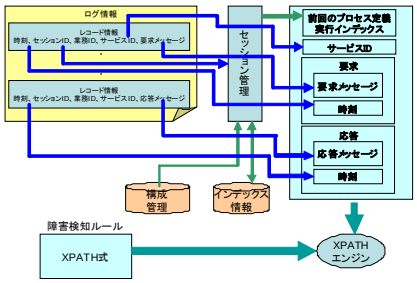


図３　障害検知処理

収集後のログをレコード単位で処理し、セッション情報から当該セッション管理部にレコードの情報を渡す。セッション管理ではサービス ID 毎に時刻、メッセージの情報、前回実行したプロセス定義のインデックス情報を合成した XML 形式のメタデータを作成する。障害検知ルールを本メタデータに対する XPATH 式として予め定義し、XPATH エンジンによる評価にて障害の有無を判定する。インデックス情報は、障害検知の評価が完了後、サービス ID と構成定義の情報から更新する。

障害検知ルールとして、例えば以下のような定義となる。

```
／応答／戻り値　＞　１００
※記載は簡略化
```

SOAP メッセージの形式は WSDL(Web Services Description Language)により規定されており、サービス仕様に基づきバリデーションルールを XPATH 式にて記述する。

## 4. 評価

システムの SOA 化に伴いサービスレベルでの監視が求められていたが、導入時の既存システムへの影響や、他社の提供するサービスの監視が課題となっていた。

本方式では、サービス呼び出しを仲介する ESB のログを解析するアプローチにより、既存システムの設定変更を不要にするとともに、他社の提供するサービスの稼働監視も実現することができた。

また、インデックス情報、時刻、SOAP メッセージ情報を保持するメタデータを構築し XPATH 式による評価を行うことで、従来方式と同等の値の不正（要求値／応答値のバリデーション）、タイミングのエラー、サービスのエラーの検知だけでなく、シーケンスのエラーといった障害にも対応できるようになった。複数条件を組み合わせた判定も可能であり、新たなルールを追加する際には XPATH 評価関数を拡張することで対応することができ、拡張性についても保持することができた。

| 項目 | 内容 | 障害検知の利用可否 |
|---|---|---|
| Info | 現在のスレッド数 | × |
| | スレッドのピーク数 | × |
| | スレッドの処理時間 | ○ |
| | スレッドのID,名称、状態 | × |
| | スタックトレース | × |
| | デッドロックの有無 | ○ |
| Property | 属性値のセット／ゲット | × |
| Operation | メソッド操作 | × |
| Notification | カウンタの一定以上の変化 | ○ |
| | 境界値（最大、最小値） | ○ |
| | 文字列の変更の監視 | ○ |

| 障害 | 本方式の対応 |
|---|---|
| タイミングのエラー | ○ |
| 値の不正 | ○ |
| サービスのエラー（内部処理エラー） | ○ |
| シーケンスエラー | ○ |

図４　従来方式との比較

原因箇所や影響範囲の特定については、検出したサービス障害の箇所に対して、発生順による重み付けの評価により原因箇所を特定する方式[2]や、構成情報からサービス呼び出しの依存関係を求め影響範囲を抽出する方式などが示されており、これらの技術と連携することで解決可能となる。

## 5. 今後の課題

本論文にて、ログを解析してサービス障害を検知することで稼働監視を行う方式について示した。

しかし、サービス障害の発生原因としては、サービス本体の問題の他にネットワークや H/W など下位レイヤの障害に起因する場合がある。

今後は、下位レイヤとサービスとの依存関係を管理するとともに、監視の連携を行い原因箇所を求める方式について検討を進める予定である。

## 参考文献

[1]JMX テクノロジのチュートリアル,
http://java.sun.com/j2se /1.5.0/ja/docs/ja/guide/jmx/tutorial/
[2] 山田耕一,”障害箇所特定の方式",第 73 回情報処理学会全国大会講演論文集(2011)