アプリケーション連携環境におけるログ出力処理方法の開発事例とその考察

5S−01

曽田 克則　由良 俊介　西村 徹　深澤 広明　野瀬 昌禎
ＮＴＴ情報流通プラットフォーム研究所

1　はじめに

ログ情報は、アプリケーション開発時だけでなく、運用時においても重要な役割を果たす。アプリケーションの実行状況を正確な記録として残すため、運用中に障害が発生した場合、その障害発生箇所の特定や、障害発生原因の解析に欠かすことができない情報である。

一方、近年、企業再編やコスト削減のため、個別に開発したシステムを連携させて、統合的なサービスを提供するケースが増えつつある。このため、連携後のシステムにおいて障害が発生した場合に、どのシステムにおいて障害が発生したかを迅速に切り分けることが重要となっている。

本稿では、前述の個別に開発したシステムを連携させた環境(アプリケーション連携環境)において、障害発生時の切り分け時間を短縮するために、従来技術をベースにログ出力処理機能を開発し、同機能の運用上のガイドラインを規定し、有効性の評価を行った。

2　切り分けプロセスにおける問題点

障害発生時の切り分け作業は、一般的に、運用者、アプリケーション開発者によって、以下に示すプロセスで行われる。

(1)運用者用のコンソール画面等で障害の発生を確認する。
(2)テキストエディタを使用してログファイルを開く。
(3)ログ全体を概観し、障害発生前後の状況を把握する。
(4)エラーに関するログの検索を行う。
(5)エラーに関するログの内容から障害状況を把握し、障害箇所を特定する。
(6)上記プロセスで障害箇所が特定できない場合、エラーやＡＰＩ呼び出しに関するログなどの出力するログの種別の範囲やログの出力可否を切り替えて、障害状況を再調査する。

上記のうち、一次切り分けとして(1)〜(5)について運用者が行い、それでも障害箇所が特定できない場合は、アプリケーション開発者が(6)の作業を行う。

これらの各プロセスにおける問題としては、以下に示すものがある。

(1)障害の発生の確認が遅れた場合、障害箇所の特定までに時間がかかる。
(2)アプリケーション連携環境においては、ログファイルが複数あるため、ファイルを開く作業に手間取り、時間がかかる。
(3)アプリケーション連携環境においては、ログが大量に出力されるため、運転状態全体を把握するのに時間がかかる。
(4)アプリケーション連携環境においては、アプリケーション開発者に依存してログのフォーマットが異なるため、特定のログの検索に時間がかかる。
(5)ログの数や内容が不十分だと障害状況の把握、障害箇所の特定に時間がかかる。
(6)出力するログの種別の範囲やログの出力可否を切り替える場合、アプリケーションの修正が必要になり、修正作業に時間がかかる。

今回は、上記の中でも、特に切り分け時間に対する影響の大きい(2)〜(6)の問題に取り組んだ。

3　従来技術と本提案における改善点

前章の問題に対して、一般的なログ出力処理機能(syslog や Java の Logger 等)においては、次のような対処がなされている[1]。

- 問題点(2)、(4)、(6)に対しては、以下のように対処されている。アプリケーションからログの出力や制御の機能を切り離す。その上で、アプリケーション側では、切り離したログ出力処理機能に対してログの出力を要求する処理の埋め込み箇所であるログの出力ポイントと出力内容を指定する。また、ログ出力処理機能側では、ログの出力可否の制御や、ログのフォーマットに関する整形処理を行えるようにする。また、ログ出力先を集約できるようにする。
- 問題点(3)に対しては、障害を起こしやすい箇所や、障害による影響の大きい箇所など、プログラム内の特定の箇所に関するログを収集できるように、ログを出力する範囲(どのアプリケーション、さらにはアプリケーション内のどのモジュールのログを出力するか)が指定できるよう対処されている。

A development and study of the logging method on application integration environment
Katsunori Soda, Shunsuke Yura, Toru Nishimura, Hiroaki Fukasawa, Masayoshi Nose
NTT Information Sharing Platform Laboratories

しかし、問題点(5)については、従来技術で解決できていない。問題点(5)を詳細に見ると２つの問題があることがわかった。

(A)「通信エラー」や「サーバから応答がありません」などログメッセージの表現がアプリケーション開発者ごとに異なるため、運用者の理解の妨げとなって、障害状況の把握に時間を要する場合がある。

(B)アプリケーションに埋め込まれるログの出力ポイントが統一されていないため、アプリケーションによっては重要度の高いログが出力されない場合があり、障害箇所の特定に支障をきたす。

今回、上述の問題を解決するため、以下の改善を行った。

(A)に対するアプローチとして、起こりうる主要なエラーを端的に表すキーワードを、エラーに関するログのメッセージの接頭辞として付与するよう規定し、それによる障害状況の把握を可能にする。本事例の環境は、複数のサーバおよびデータベースからなるネットワーク構成であるため、特に運転の継続に影響の大きいと考えられる①通信エラー(Communication_Error)、②DB コネクションエラー(DBConnection_Error)、③ファイル I/O エラー(FileIO_Error)、④前記以外の致命的なエラー(Fatal_Error)について、接頭辞を規定する。

(B)に対するアプローチとして、重要度の高いログの出力ポイントを統一的に規定することで、アプリケーションごとに異なるログの出力ポイントの差をなくし、障害状況の把握を容易にする。出力ポイントの指定にあたってはエラー解析に有効な例外発生箇所とその前後の処理の流れがわかるように、プログラム内の①例外処理部の先頭(エラーログ)、②各メソッドの先頭と最後(イベントログ)、③複数の処理で構成されるサービスの完了箇所(サービスログ)に、各状況を示すログを出力するように規定する。

4　本提案のログ出力処理機能とログ管理規定

前章の従来技術と本提案を組み合わせて、本稿の主題であるアプリケーション連携環境における切り分け時間の短縮がなされたかどうかを検証するため、従来技術をベースにしたログ出力処理機能を開発し、同機能の運用上のガイドラインとしてログ管理規定を定めた。その機能と規定について、以下に示す。

4．1　ログ出力処理機能

複数のアプリケーションからのログ出力要求を受け付け、アプリケーションから渡されるログ情報を基に、指定された条件による出力制御とログのフォーマットに関する整形処理を行い、指定されたファイルに出力する。

4．2　ログ管理規定

ログ出力処理機能を利用するアプリケーションが守るべきルールとして、前章で示した本提案のログメッセージの接頭辞とログの出力ポイントの規定に加えて、障害箇所の特定に有効と考えられる以下の従来の規定を適用した。

- ログ出力処理機能に対してアプリケーション内のどこでログの出力要求があったかを示すログの出力箇所の名称を識別子として規定した。具体的には Java のパッケージ名付きクラス名を識別子とした。

5　実験および結果と考察

今回の実験環境に、前章の機能および規定を上記環境に適用し、有効性を評価した。評価においては、運用者、およびアプリケーション開発者に対するヒアリング形式で行った。その内容は以下の通りである。

- 接頭辞の付与により、障害状況の把握が容易になり、切り分けにかかる時間が短縮された。しかし、ログメッセージの記述内容が十分でないログが一部存在したため、障害状況の把握に手間取る場合があった。これは、ログメッセージの記述内容についての考え方がアプリケーション開発者によって異なるためである。
- 運用者が参照する一次切り分け用のログについては、出力ポイントの不足は無かった。しかし、アプリケーション開発者が参照するデバッグ用ログ(「DB 接続開始」など)については、アプリケーション開発者独自の判断でログの出力ポイントを決めているため、ログ情報が不足し障害箇所の特定が出来ない場合があった。これはデバッグ用ログの出力ポイントの指定に関して、アプリケーション開発者の自由度が高すぎることに起因していると考えられる。

6　おわりに

本稿では、アプリケーション連携環境でのログ出力処理方法について、従来技術と改善したログ管理規定を組み合わた場合の有効性の評価について述べた。今回の結果では、切り分け時間の短縮について一定の効果が見られたが、ログメッセージの記述内容の不十分さや、デバッグ用ログの不足などの課題があることがわかった。今後は、ログメッセージの記述内容に関する規定、デバッグ用ログの出力ポイントに関する規定の検討を行いたい。

参考文献

[1] Graham Hamilton : Java Logging APIs DRAFT0.55 http://jcp.org/aboutJava/communityprocess/review/jsr047/spec.pdf