トレースログを用いた並行処理システムの動作解析*

6J－1

高橋 理・石岡卓也†

三菱電機(株)産業システム研究所‡

1 はじめに

鉄道や電力・プラントなどの大規模な監視制御システムでは、複数のタスクが並行して処理を行っている。このようなシステムでは、データや制御の流れが複数ある上に、外部からのタイミングに依存する挙動が存在するため、処理の流れを把握することが大変難しく、試験や保守に多大な労力を要する。

一方、このようなシステムでは、障害発生時や性能評価に備えて、処理履歴を動作ログとして、逐次出力している。このログファイルに出力されるログの量は、1日あたり数万行を越えることもある。

従って、障害発生時や性能評価の際に、これらの膨大なログを人間の手により解析し、状態把握を行うことは極めて困難である。そこで、ログの自動解析により、システムの保守・試験作業を支援するために必要な情報を提供する手法について検討した。

2 ログファイルの可視化

ログは、出力時間を示す時刻項とログの本文とからなる行単位の出力情報である。ログ本文には、イベントの種別やデータの内容、処理先・処理元タスクなどに関する情報が自由書式で出力される。また、一つの処理を複数のタスクに分散して行うことが多く、単位処理(各タスクでの処理)が完了するたびにログを出力するため、一つの処理パターンが複数のログから構成されることもある。

図1に示すログファイルの可視化[1]は、ログの内容(処理タスク・内容など)に応じて異なる図形や色を用いることにより、処理の内容に関する理解を視覚的に助けるものである。この可視化により、システムを構成する複雑な処理パターンを分類したり、動的な処理の遷移を把握することができる。

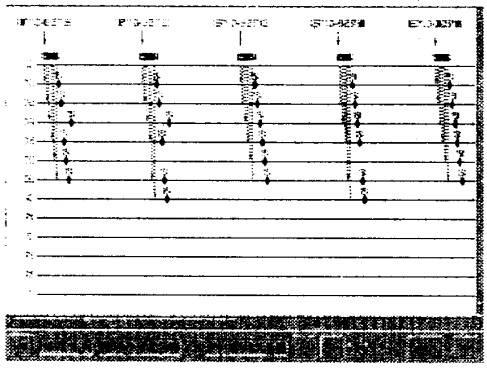

図1. ログファイルの可視化例

3 処理パターンの検索

次に、ログファイルから繰り返し出力されるログ列(処理パターン)をデータ圧縮アルゴリズムを利用して抽出することにより、実質的なログの量を減らすことを試みる。また、抽出したログ列を他のログと区別して表示・出力することで、システムの処理構成が明確になり、ログ解析の効率は一層向上する。

3.1 データ圧縮手法

データ圧縮アルゴリズムの一つである辞書的圧縮法LZ78[2]は、入力データを文字列として読み込み、辞書に登録された文字列に一致するかどうか調べる。そして、最大長の一致文字列の辞書登録番号(インデックス)をその文字列の代わりに出力し、次の1文字を追加した文字列を辞書に新規登録するものである。

この手続きは、出現頻度の高い文字列に対して、さらに長い文字列を辞書登録することにより、できるだけ長い文字列を発見しようとする処理で、高い圧縮率を実現することができる。

3.2 処理パターン検索アルゴリズム

本研究では、行単位で出力されるログを1つの文字と考える。そして、データ圧縮手法LZ78におけ

*Analysis of concurrent processes with classifying periodic log patterns

†Satoru Takahashi and Takuya Ishioka

‡Industrial Electronics and Systems Lab. Mitsubishi Electric Corporation

る辞書作成方式を利用することにより、複数の連続したログを1つのログ列として辞書登録し、出力頻度の高い処理パターン(ログ列)を以下の手順で抽出する。

3.2.1 辞書作成

入力ログファイルの先頭ログを起点とし、起点からの連続する複数のログを辞書に登録する。

まず、起点からの連続するログが、すでに辞書に登録されているログ列のいずれかと一致するかを判別する。今、辞書の i 番目にあるログ列について、登録されているログの数を n_i 個とし、各ログを時刻項(出力時間)および本文の組合せ $(T_j, A_j)(1 \leq j \leq n_i)$ として表現する。一方、入力ログファイルの起点となるログ (T_1', A_1') からの連続した n_i 個のログを $(T_j', A_j')(1 \leq j \leq n_i)$ とする。このとき、式(1)を満たすならば、n_i 個の入力ログは、辞書の i 番目に登録されているログ列に一致するとみなす。

$$\begin{array}{ll} \text{入力ログ} & (T_1', A_1')(T_2', A_2') \cdots (T_{n_i}', A_{n_i}') \cdots \\ \text{辞書登録ログ列} & (T_1, A_1)(T_2, A_2) \cdots (T_{n_i}, A_{n_i}) \end{array}$$

$$\forall j(1 \leq j \leq n_i) \rightarrow \begin{cases} A_j' = A_j \\ |(T_j' - T_1') - (T_j - T_1)| < \varepsilon \end{cases} \quad (1)$$

すべての辞書登録ログ列のうち、入力ログファイルの起点から読み込んだログに一致するものがある場合、該当ログ列の中で最もログの数が多いもののログ数を $n_{\max}$ とする。このとき、入力ログファイルの起点から $n_{\max}$ 個のログ列(辞書登録済)に、その次のログ $(T_{n_{\max}+1}', A_{n_{\max}+1}')$ を加えた $n_{\max}+1$ 個のログを辞書に新規登録する。

入力ログファイルの起点ログが、いずれの辞書登録済ログ列とも一致しない場合には、その起点ログ自身を辞書に登録する。

新規登録したログ列に含まれるログの数の分だけ起点を進め、次の辞書登録を行う。

3.2.2 ログ列の評価

入力ログファイルを末尾まで読み込み、辞書登録が終了した後、式(1)に従ってログファイルを再検索し、各辞書登録ログ列の出現頻度を計算する。

そして、各ログ列を出現頻度やログの数などから得点化する。辞書に登録されたログ列はログファイル中のログの並びにすぎないが、繰り返し現れるログ列ほど一定の処理パターンを構成していると考えられる。一方、ログの数が少ないログ列ほど出現頻度は高くなるが、処理が未完結になっている場合が多い。そこで、出現頻度とログ数とのトレードオフを考慮した評価を行う必要がある。

3.2.3 選択・後処理

得点最大のログ列を検索結果(処理パターン)として出力する。また、この処理パターンを辞書やログファイルから削除した上で、次の処理パターンを検索するために 3.2.1 に戻る。

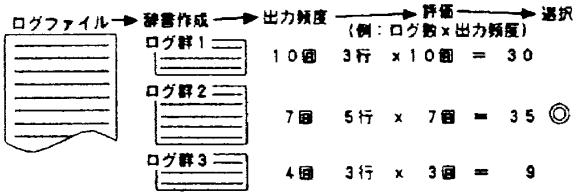

図 2. 処理パターンの検索イメージ

4 おわりに

大規模システムから処理履歴として出力されるログの量は膨大で、人間の手による解析は困難である。そこで、本報告では、ログファイルを自動解析して、システムを構成する処理パターンを検索する手法について検討した。

今後は、処理パターン検索手法についての詳細検討を行うと共に、実規模のシステムから出力されるログファイルを解析することにより、全体像の把握や障害改修に役立て、本手法の有効性を検証したい。

参考文献

[1] 高橋、石岡: トレースログ解析による大規模ソフトウェアシステムの可視化と状態解析, 第42回システム制御情報学会講演論文集, pp. 475–476、1998

[2] Ziv, J. and Lempel, A.: *Compression of individual sequences via variable-rate coding*, IEEE Transactions on Information Theory, Vol. 24, No. 5, pp. 530–536, 1978