2U-7

実時間分散制御システムの永続化に関する研究 *

西野 弘毅
電気通信大学 情報通信工学専攻†

高田 昌之
電気通信大学 情報基盤センター‡

1 研究の背景

近年では、様々な産業分野で自動化、IT 化が進んだことにより、計算機システムが大規模化かつ専門化しソフトウェア開発コストの増大が問題となっている。その解決策の一つとして、最初から大規模なシステムを作るのではなく、単純な機能を持ったソフトウェアをネットワークによって繋ぎ合わせ協調させることにより全体として大きなシステムを構築する手法がある。このソフトウェアを繋ぎ合わせ協調動作させるシステムを分散システムという。

この分散システムのうち、機器制御での協調動作を行うために実時間性を重視したものが、高田研究室によって開発された GlueLogic である。

GlueLogic は、複数のエージェントと呼ばれる単純な機能を持ったソフトウェアと協調動作をするための共有情報を扱う一つのサーバによって構成される。

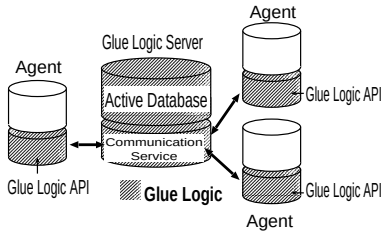

図 1: GlueLogic

そして、GlueLogic は、機器制御に十分な実時間性を確保するために、乏しい計算機資源上での動作のため最低限必要な機能を厳選している。

しかし、GlueLogic を介したシステムが自動化工場の外の資材管理、資材調達など GlueLogic の設計の想定範囲の外にも広がると問題が出てきた。

それは、GlueLogic の管理する共有情報が、GlueLogic サーバのメモリ上にのみにあり、永続化すなわち複数のシステムの実行を跨いで情報を保持することができていないことである。例えば、システムが異常終了した場合に全ての共有情報が失われてしまい復旧が困難となり、かといって個々のエージェントで情報を永続化すると、情報が複数の計算機に散らばってしまう。

このため、GlueLogic が永続化機能を保持することが必要となった。

2 研究の目的

このような背景から、GlueLogic の扱う共有情報の永続化を行うことにより、GlueLogic をより広い領域で有用に活用できるようにすることを本研究の目的とする。

2.1 研究の要件

本研究において満されるべき要件を、永続化機能の追加により機器制御に十分な実時間性を損なわないことと、これまでに開発された GlueLogic を用いたシステムのソフトウェアリソースを無駄にしないため、既存のエージェントに修正を加えずに永続化機能を利用できるようにすることの 2 点とした。

2.2 永続化へのアプローチ

上記の二つの要件を満して GlueLogic を永続化するためには、二つの考慮すべき事項がある。

それは、まず GlueLogic にどのようにして新しい機能を追加するのか、次に永続化機能そのものをいかに実時間性を保てるように実現するのかである。

3 GlueLogic への機能追加

まず、GlueLogic への機能追加方法について考える。

これまで GlueLogic に何らかの機能を追加する場合は汎用エージェントとして実装を行い、システム全体の実時間性に影響を与えないためサーバには手を加えない方針を取ってきた。

しかし予備実験により、エージェントとして永続化機能を追加しても、通信のオーバーヘッドにより永続化した情報の整合性が保証できないことが判明した。

そのため、新たな方法としてサーバの機能をスレッドごとに分けてマルチスレッド化を行い、永続化機能をそのサーバ内のスレッドとして実装を行うこととした。

3.1 サーバのマルチスレッド化

マルチスレッド化したサーバの構成を図 2 に示す。

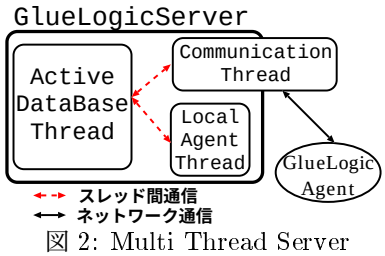

図 2: Multi Thread Server

*Research for Perpetuation of Real-time Distributed Control System

†Hiroki NISHINO, The University of Electro-Communications, Department of Information & Communication Engineering

‡Masayuki TAKATA, The University of Electro-Communications, Information Technology Center

まず、サーバ内部に共有情報を管理するデータベーススレッドがある。このスレッドが既存のサーバであるかのように、他のスレッドはスレッド間通信により共有情報を参照できる。

このようにサーバの内部で共有情報を使って協調動作を行うスレッドを、ローカルエージェントと呼ぶ。

ローカルエージェントの一種に通信エージェントがある。この通信エージェントがサーバ外部の GlueLogic エージェントとサーバ内のデータベーススレッドの橋渡を行うことにより、外部からは既存のサーバと同様に扱うことができ、データベーススレッドから見ても外部のエージェントでもローカルエージェントでも同じものとして扱うことができる。

3.2 マルチスレッド化の利点

マルチスレッド化を行い永続化機能をローカルエージェントとして追加することにより、スレッド間通信はメモリを共有するのでネットワーク通信と比べて共有情報に高速で確実にアクセス可能なため、ログの整合性を保証することができる。

さらに、各機能が並列に動作するため、通信やファイルアクセスなどの機能がボトルネックとなるのを防ぐことができ、実時間性を保つことができる。

これらにより、既存の機能の追加方法では通信のオーバーヘッドや実時間性への影響のため実現できなかった機能を追加することが可能となった。

4 永続化機能と実時間性

永続化において、永続化した情報間の整合性が取れていることは必須の要件である。

しかし、永続化の主要な方法であるダンプは、整合性のため一度に情報全体をダンプする必要がありその処理の間は情報の変更ができないので、情報量が増え処理の時間が増加すると実時間性に影響を与えてしまう。

そこで、ダンプの整合性について考えると、整合性をとる必要があるのは情報間に関連がある場合だけであることが解る。

そのため、共有情報内で関連のある情報をグループとしてまとめることにより、この問題に対処することができる。

4.1 共有情報のグループ化

共有情報のグループ化を行うことにより、結び付きの強い共有情報を明示することができる。

これにより、異なるグループの共有情報の間では整合性を考慮する必要が無くなるため、グループを個別に永続化を行うことができるようになる。

そして、グループごとに永続化の可否や強度を変えることにより、全体としての処理を減らすことができ、実時間性を保つことができる。

5 性能評価

サーバのマルチスレッド化、および永続化機能追加による実時間性への影響を評価するため、

1. 既存の GlueLogic サーバの C 言語版
2. 既存の GlueLogic サーバの Perl 版
3. Java により実装したマルチスレッド化サーバ

の 3 種類のサーバと、3 のマルチスレッド化サーバで永続化機能を最大の強度 (全ての共有情報のログを取る) の場合の 4 つの場合のサーバの 1 秒当りの動作サイクルを、計測エージェントの数を変えながら測定し、比較した。

5.1 結果

図 3 に結果を示す。

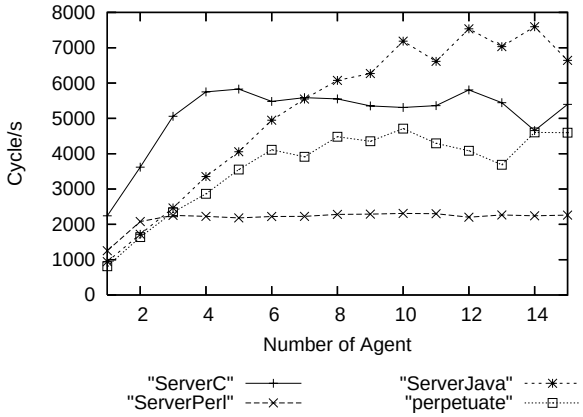

図 3: サーバの動作サイクルの比較

既存の GlueLogic サーバは、Perl 版は 2200[1/s]、C 版は 6000[1/s] 程度で動作サイクルが限界に達することが判明した。

マルチスレッド化サーバはエージェント数が少ない場合は既存のサーバに動作サイクルが劣るが、分散制御システムとして重要なエージェント数が多い場合では既存のサーバに勝り、さらに既存のサーバと比べて動作サイクルの限界が高いことが判明した。

永続化機能の結果は、エージェント数が少ない場合はマルチスレッド化サーバとほぼ変わらない動作サイクルであり、エージェント数が増加するにしたがって永続化すべき情報が増えるためマルチスレッド化サーバに動作サイクルは離されていくが、既存の C 版のサーバに近い動作サイクルを記録し、十分な実時間性を持つことが判明した。

6 結論

GlueLogic のサーバをマルチスレッド化し、そのスレッドとして永続化機能を実装した。そして、既存の GlueLogic サーバとの比較実験により、マルチスレッド化サーバおよび永続化機能が十分な実時間性を持つことを確認した。

今後の課題は、C 版でのマルチスレッド化サーバの実装やローカルエージェントを動的にロードする機能の追加などが考えられる。

参考文献

[1] 高田昌之: “GlueLogic の機能拡張について”, CQ 出版,interface 5月号, 1998

[2] Andrew S. Tanenbaum、Maarten van Steen: “分散システム—原理とパラダイム”, ピアソンエデュケーション , 2003