タスク並列処理ソフトウェア開発支援ツール

4N−5

助村俊一† 加藤昭史† 小池秀耀†

大規模で複雑なソフトウェアシステムの並列処理を制御するため、各プログラムやモジュールの処理(タスク)の実行順序や通信タイミングをビジュアルに記述し、実行時の論理チェックや実行タイミングをシミュレーションするツールを開発している。そのプロトタイプについて報告する。

Development Tool for Parallel Task Prosessing Software

SHUNICHI SUKEMURA,† AKIFUMI KATO † and HIDEAKI KOIKE†

We develop a tool to control parallel prosessing of large-scale complicated software system. So we treat a program and module as task, and describe order of execution and communication timing of task in visual using tool. And tool can do logical check in execution and simulate execution timing of task.

1. はじめに

プログラムが複雑に関係しあっている大規模ソフトウェアシステムで並列処理を行う場合、個々のプログラムが非同期に並列で動いているためプログラムの実行順序を把握したり、デッドロック等の実行時の矛盾のチェックをすることが難しい。非同期な並列プロセスの記述として Petri Net[1] が知られており、Petri Net を用いた矛盾チェックなどの研究が多くされているが、Petri Net では、データの流れを中心に記述し、静的なネットワークしか記述できない。そこで、我々は、処理の単位(プログラムやモジュール等)をタスクとしてとらえ、タスクの処理の流れを中心に非同期で並列に動作するタスクを記述する方法を開発している。記述には2種類の記述がある。一つは言語としてテキストで厳密に表現する記述と、グラフとしてユーザに分かりやすく表現する記述である。

これらの記述を用いて、プログラムの実行時間、プログラム同士の実行順序や実行タイミングプログラム間のデータの受け渡しの条件やタイミング・通信時間を記述し、シミュレーションを行い、複雑なソフトウェアシステムの並列処理の動作を確認したり、実行順序の最適配置、論理チェック(デッドロック等の矛盾の発見)を行うことを可能とする(図1参照)。

以下で、タスクの並列動作を記述するために必要な記述項目について述べ、その記述項目のグラフ記述、テキスト記述について説明し、それらの記述を操作するツールについて述べる。

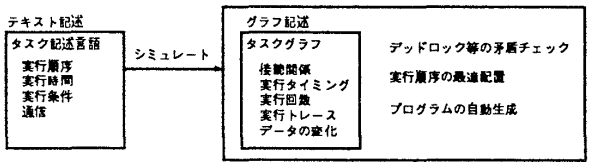

図1 概念図

2. 記述項目

複数のタスクが非同期に並列で動いている様子の表現は、タスクの定義(タスク)と、動的なタスクの実行(実行)、タスク間のデータ通信の記述(通信)に分類される。表1に分類別の記述項目を示す。

タスクは静的に存在し、他のタスクから起動されることで処理をはじめる。タスクは名前と処理時間、処理内容のデータから構成される。名前はタスクの識別に用い、処理時間はタスク実行のシミュレーションのために用いる。処理内容は、処理の内容をテキストで記述し、処理順序や処理タイミングをシミュレーションするために用いる。

タスク間のデータの受け渡しには通信を用いる。通信では送るデータの種類・容量と通信時間を指定する。通信のモデルとしては以下のようなモデルを想定している。送信と受信にはそれぞれ FIFO のバッファがあり、送信時には FIFO のバッファへのコピーを行い、受信時には FIFO のバッファからのコピーを行う。FIFO のバッファは溢れることがないものとして扱う。送信のバッファから受信のバッファへコピーが行われる時間が

† (株)富士総合研究所
FUJI RESEARCH INSTITUTE Corporation

表1 記述項目

分類	項目
タスク	タスク 処理内容 処理時間 変数の定義・設定
実行	タスクの起動 条件分岐
通信	通信 同期 送信 (ノン)ブロッキング受信 転送データ・時間

通信時間である。

送信時にはデータ受信のチェックはしない。受信は、ブロッキング受信とノンブロッキング受信があり。ブロッキング受信では、自分宛のデータが来るまで処理をブロックする。

同期のための表現も導入した。ここで言う同期とは処理の同期であり、同期ポイントを持つすべてのタスクが同期ポイントに達したときに、次の処理に進むという意味である。

3. グラフ記述

上述の記述項目をグラフとして表現する。

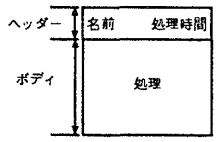

図2 タスク

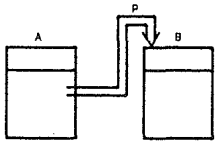

図3 タスクの起動

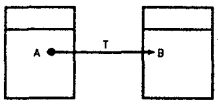

図4 通信

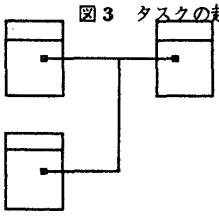

図5 同期

タスクは、四角で表現し (図2参照)、ヘッダとボディーから構成され、ヘッダーにはタスクの名前と処理時間が表示され、ボディーには処理の内容が書かれる。処理の内容はテキストで記述する。

タスクの起動には、二重線の矢印を用いる (図3参照)。矢印の先のタスク (B) が起動される。矢印の上に書かれている P で、プロセッサやホスト名などを指定する。

タスク間の通信には矢印を用いる (図4参照)。タスク A からタスク B にデータを送信する。●はデータを表す。T は通信時間を示す。

同期は、黒い四角で表す (図5参照)。同期同士を線で結ぶ。複数のタスクの動作の同期をとる。全てのタスクが同期を実行しない限り、次に進まない。

4. テキスト記述

テキストでの表現として言語を設計した。特徴的な、言語仕様について説明する。

send 文::= ラベル ":" "send" "(" データ郡 ")" "to" "(" 送信先郡 ")".

送信先::= 送信先ラベル名 "of" タスク名 |"sender of" ラベル名 |"parent".

receive 文::= ラベル名 ":" "receive" "(" データ郡 ")" "from" "(" 送信元郡 ")".

送信元::= 送信元ラベル名 "of" タスク名 |"parent" .

自分にデータを送ってきた相手に対して結果等のデータを送りたい場合がある。この場合、自分にデータを送ってくる相手は実行時にしか分からないため、記述が難しかった。そこで、送信先の指定に「自分自身にデータを送ってきた相手」という指定が記述できるようにした。具体的には「sender of ラベル名」であり、この意味はラベル名に相当する receive 文の位置にデータを送ってきた相手という意味になる。また、自分を起動した相手にデータを送ったり、受けたりしたい場合がある。そのための記述として「parent」を導入した。これは、自分を起動したタスクを意味している。

5. ツー ル

ツールでは、テキストを読んでグラフに表示し、実行をシミュレートする。シミュレート結果として、タスクの実行順序や実行タイミングの表示、論理チェックなどを行う。

6. ま と め

複雑に関係している非同期な並列プロセスの実行順序を記述するためのグラフ表現とテキスト表現について説明した。今後は、ツールを用いて実際のシステムを記述し、有効性を検討する。

謝辞 本研究は、科学技術庁の平成10年度科学振興調整費総合研究による「人間系の特性を考慮した大規模・複雑システムのモデル化、解析、制御、設計に関する統合研究」の一環として富士総合研究所が科学技術庁から委託を受けて実施したものである。

参 考 文 献

1) : High-level Petri Nets - Concepts, Definitions and Graphical Notation (1997).