SR2001における並列トレーサ/モニタ機能

2H－3　吉松　祐三子*, 岩嵜　正明*, 山口　理栄**, 佐竹　昌之**, 吉池　真悟***

（株）日立製作所　*システム開発研究所, **ソフトウェア開発本部
***日立ソフトウェアエンジニアリング（株）

1. はじめに

超並列計算機では、性能スケーラビリティが求められる。特定ノードへの負荷の集中はプログラム実行性能を大きく低下させるため、CPUやネットワークの負荷状況をモニタリングする必要がある。また、ソフトウェア開発時には、OSダウン時の原因解析や、ノード間データ転送でのメッセージ欠落・同期不良のデバッグ機能が必要となる。そこで、超並列計算機SR2001 [1]では、デバッグ及び性能チューニングツールとして、並列トレーサ/モニタ機能を提供する。

並列トレーサ/モニタにおいて、データ量はノード数に比例して増加する。また、トレーサ/モニタ実行によるユーザプログラムへの影響を最小限にする必要がある。本稿では、SR2001における並列トレーサ/モニタ機能およびデータ収集方式について述べる。

2. 並列トレーサ/モニタの概要

2.1 並列トレーサ/モニタの構成

並列トレーサ/モニタの構成を図1 に示す。

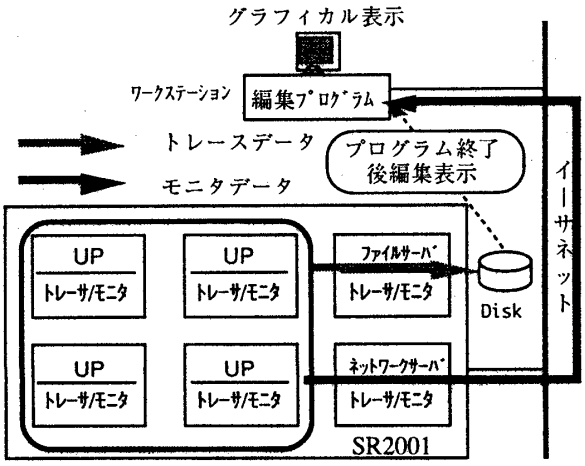

図1. 並列トレーサ/モニタの構成

各ノード上には、ノードの動作状況やそこで発生するイベントを収集するトレーサ/モニタが動作する。得られた情報の一部は、SR2001とイーサネットでつながれたワークステーション上に集め、編集・リアルタイム表示することでシステムの動作状況のチェックを可能としている。（モニタ機能）

また、実行状況や不良の原因を詳細に時間をかけてチェックするために、得られた情報は、一旦SR2001上のディスクに蓄え、実行後に解析する機能を持つ。（トレース機能）

2.2 並列トレーサ/モニタの機能概要

2.2.1 トレース情報収集機能

各ノードのトレーサでは下記の３種類のデータを収集する。

(1) イベント情報

スレッド・スイッチやシステム・コール発行等のイベントのトレースを発行時間順に採取する。これにより、OS内の動作状況を知ることができる。

(2) 通信情報

ノード間の送信パケットと受信パケットのメッセージ・ログを記録する。ユーザは、このメッセージ・ログを利用して、任意のノード間でのメッセージ交換における同期不良、メッセージ欠落等のデバッグを行うことができる。

(3) システムログ情報

I/Oエラー情報、デバイスの起動ログ等のシステム内のエラー及び動作シーケンス情報を記録する。OSダウン後、リモート・メモリダンプ機能によってトレースバッファに格納されたシステムログ情報を参照することで、OSダウンの原因解析を行う。本情報は、イベント情報や通信情報と異なり、ファイル出力の対象とせず、各ノード内のメモリ上に最新状態だけを残す。

2.2.2 モニタ情報収集機能

各ノードのモニタは、統計的なサンプリング情報、ハードウェアカウンタ情報等（以下、モニタ情報という）を収集する。モニタ情報には、下記の４種類がある。

(1) CPU稼働状況

各ノードのCPU稼働率、仮想メモリ使用状況等の情報。これにより、各ノードの負荷バランス状況等を得る。

(2) I/O稼働状況

各ノードのネットワークを除く入出力機器稼働

The Parallel Tracer and Monitor For SR2001
Yumiko YOSHIMATSU, Masaaki IWASAKI,
Rie YAMAGUCHI, Masayuki SATAKE, Shingo YOSHIIKE
Systems Development Laboratory and Software Development Center, Hitachi, Ltd., Hitachi Software Engineering Co., Ltd.

状況情報。これにより、I/O性能やI/Oの負荷バランス状況等を得る。

(3) ネットワーク稼働状況

ネットワークのトラフィック情報。これにより、通信性能や通信の負荷バランス情報を得る。

(4) ハードウェア動作状況

プロセサ性能、メモリインタフェース性能情報。これによりFLOPS値、キャッシュヒット率を得る。

3. データ収集方式

3.1 トレースデータ収集方式における課題

並列トレーサ/モニタではノード数に比例して大量のデータがデータ収集ノードに送信される。このため、データ収集方式が重要な課題となる。以下、並列トレーサにおけるデータ収集方式に関する課題を示す。

(1) データ収集処理の分散化

並列トレーサでは、トレースデータがノード数に比例して増加する。このため、各ノードで採取したトレース情報を単一ノードでデータ収集を行うと、データ収集による負荷が集中する。従って、収集側の処理の分散化が必要である。

(2) データ転送方式

各ノードでは大量のトレースデータの送受信が発生する。従って、被トレースプログラムに対する影響を最小限とするデータ転送方式が課題となる。

(3) トレース・バッファ管理方式

トレースデータの転送回数を減らすためにはバッファリングが必要である。しかし、トレースすべきデータの発生頻度が高くなると、オーバフローが発生する。そこで、オーバフロー時の処理を検討する必要がある。

3.2 課題の解決方式

(1) データ収集ノードの分散化

データ収集処理の集中を防ぐため、図2 に示す構成で処理の分散化を図る。トレースデータを 1 カ所のディスクに格納するのではなく、複数の入出力ノード上に収集プログラム(サブマスタ)を動作させ、複数ディスクに分散して格納する。

収集時には、各サブマスタに対していくつかのノードを対応させ、対応したノードのトレーサ(スレーブ)からのトレースデータのみを各サブマスタが収集格納する。トレースデータの読み出し時には、サブマスタ全体を制御するマスタトレーサがそれぞれからデータを集め、一つのトレースデータに変換する。

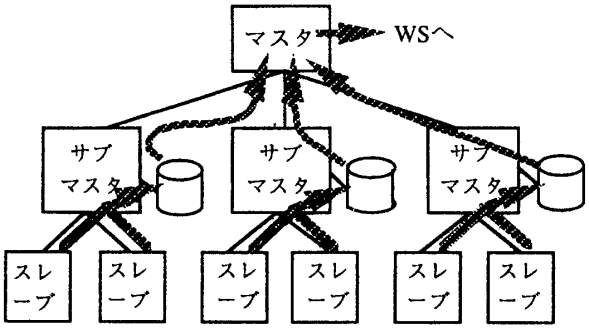

図2 並列トレーサの構成概略

割り当てるサブマスタの数や各ノードとの対応関係は、採取対象ノードの数やネットワーク上の幾何学的位置関係を用いて決定する。これにより、収集負荷を一定以内に押さえながら、効率よく多ノードからのトレースデータの採取を可能とする。

(2) 高速プロセッサ間通信機能の採用

SR2001のトレーサでは、スレーブからサブマスタへのデータ転送には、リモートメモリへの直接データ転送機能[2]を用いる。本方式は、再送等のプロトコル処理を必要としないため、高速転送が可能であり、通常通信を使った場合に比べ、転送処理時間を 1 桁近く削減できる。従って、被トレースプログラムに対する影響が小さい。

また、トレース・バッファのダブルバッファリング制御を行い、バッファオーバフローの可能性を削減している。さらに、それでも発生するバッファオーバフローに関しては、以下のエラー検出、回復処理を行うことで、例外的な状況にも対応可能とする。

(1) 各トレースレコードの前後に同じシーケンス番号を格納し、受信時に 2 つの番号の一致をチェックすることで、受信バッファへのデータの上書きをチェックする。

(2) トレース・バッファ両面がFULLになったままで、送信が完了していない場合、送信が完了するまでトレースレコードを破棄し、データを破棄したことを示す喪失レコードを送信する。

4. おわりに

並列トレーサのデータ収集方式について検討を行い、その結果、被トレースプログラムに対する影響の小さい並列トレーサ機能を実現した。

参考文献

[1] 西門,他: SR2001 OSの開発コンセプト 情報処理学会第50回全国大会 (1995)

[2] 薗田,他: SR2001における高速プロセッサ間通信機能 情報処理学会第50回全国大会 (1995)