スーパーデータベースコンピュータ (SDC) におけるデータ流制御方式の評価

5H−6

平野聡 原田昌信 中村稔 鈴木和宏 喜連川優 高木幹雄
東京大学 生産技術研究所

1 概要

スーパーデータベースコンピュータ SDC は、プロセッサ4台と磁気ディスク装置2台を密に結合し「処理モジュール」とし、それらを高機能オメガ・ネットワークで疎に結合したハイブリッド・アーキテクチャをとる高並列関係データベースサーバである。SDC は「コンテナモデル」によりディスク、ネットワーク、プロセッサを関係付け、I/O 制御、バッファ管理、データ流制御、負荷分散、デッドロック防止などを実現している [1]。本論文では、デバイスへの負荷に応じてバッファの容量を適応的に制御することによりバッファを通過するデータ流を調整し、より少ないバッファ容量で実行時間の短縮を図る「適応化」の方式、アルゴリズム及びシミュレーションによる評価の結果について述べる。

シミュレーションの結果によれば、バッファの構成を適応的に決定するアルゴリズムは負荷変動が無い場合には若干のコストがかるもののほぼ最適なバッファ容量と実行時間を発見できること、大きな負荷変動がある場合にはバッファの容量を固定的に割り当てておくアルゴリズムと比較してより少ないバッファ容量で同等以上の実行時間を実現可能であることがわかった。

2 バッファの適応化アルゴリズム

デバイスには通常バッファを割り当てる。しかし、メモリの総量は限られているため、各ハードウェア構成要素（プロセッサ、ネットワーク、ディスク）にそれぞれ無限大のバッファを割り当てるわけにはいかない。従ってメモリを分割しそれぞれに適当量を割り当てる必要がある。本論文ではネットワークの出力バッファを例にとって、バッファの容量や構成を負荷に対して適応的に決定するアルゴリズムを示し、評価を行う。

SDC のコンテナモデルでは、デバイス間のデータ転送をコンテナと言う容器で行ない、コンテナの集合がバッファとなる。バッファが一杯あるいは空になるとバッファからの出力の効率が低下するため、コンテナモデルは以下のようにバッファを管理する。

プロセッサからネットワーク出力バッファ (以下バッファと略す) へ流入するコンテナ量がネットワークへ出てゆく流出量より多いと、バッファにコンテナが溜まってゆく (図1)。各時点でのコンテナ数 CP が中断点 SP に達すると、ディスク入力の中断をディスクマネージャに要請する。実際にディスク入力が止まり、バッファ内のコンテナ数が減り始めるまで時間遅れがあるため、CP は SP を超えて大きくなる。CP の最高到達点を高水位指標 HWM と呼ぶ。CP が低下してゆき、再開点 RP に達するとディスク入力を再開する。再開をディスクマネージャに要請してから実際にコンテナが流入してくるまで時間遅れがあるため、RP は 0 より高い点にある。CP の最低到達点を低水位指標 LWM と呼ぶ。BC はバッファの容量である。また、SP は RP より最低 MB コンテナ分大であるものとする。

適応化は次の方針で行なう。

Evaluation of the dataflow management of SDC, The Super Database Computer

S.Hirano, M.Harada, M.Nakamura, K.Suzuki, M.Kitsuregawa, M.Takagi

Institute of Industrial Science, University of Tokyo

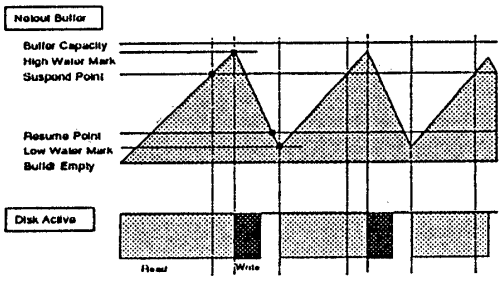

図 1: バッファの構成

SP − 非中断状態 かつ CP 上昇中 かつ $CP = SP$ であったら、それは中断点でディスク入力を中断状態とする。SP は初期値を $2/3BC$ とし、CP が再開点に達したら SP を下記アルゴリズムに従って決定する。

RP − 中断状態 かつ CP 下降中 かつ $CP = RP$ であったら、それは再開点で中断状態を解除する。RP は初期値を $1/3BC$ とし、CP が中断点に達したら、RP を下記アルゴリズムに従って決定する。

再開点では前回の再開点からの HWM が確定しているので、その実績を基に次回に期待される HWM を推測し、SP 及び BC を決定する。前回の HWM が BC に近すぎた場合には BC を追加し、SP と BC との距離を大きくすることにより、次回に期待される HWM が BC から遠ざかるようにする。また、前回の HWM が BC から遠すぎた場合には BC を減少、SP と BC との距離を小さくすることにより、次回に期待される HWM が BC に近づくようにする。

アルゴリズム 1: BC を可変にしない場合

$$\delta = BC - HWM - HM$$

$$SP := \begin{cases} SP + \delta, & \text{if } RP + BC < SP + \delta \\ RP + MB, & \text{otherwise} \end{cases} \quad (1)$$

アルゴリズム 2: BC を可変にする場合

$$\delta = BC - HWM - HM$$

$$BC := BC + RP + MB - (SP + \delta) \quad (2)$$

$$SP := RP + MB$$

ここで、HM (High Margin) は期待される HWM と BC との間にとる緩衝用バッファであり、データ流入量が期待よりも多かった場合に $HWM = BC$ となることを予防する役割を果たす。RP も SP と同様に決定する。

3 負荷が一定の場合における適応化アルゴリズムの評価と改良

図2にコンテナ発生源が定常的にコンテナを発生する場合においてアルゴリズム 2 を使用したシミュレーションの結果を示す。シミュレータは、source、buffer、sink、timebase からなり、それぞれ、ディスクとプロセッサ、適応化アルゴリズムを備えたバッファ、ネットワークをシミュレートする。timebase は各モジュールにクロックを供給する。

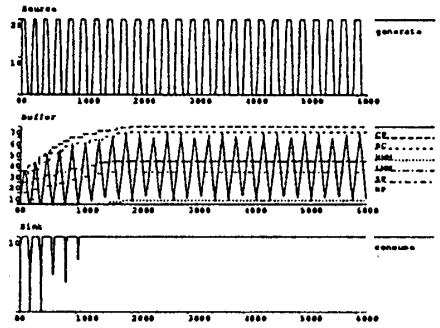

図 2: アルゴリズム 2(負荷変動なし)

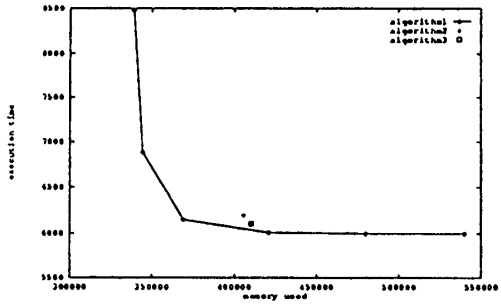

図 3: メモリ使用量対実行時間 (負荷変動なし)

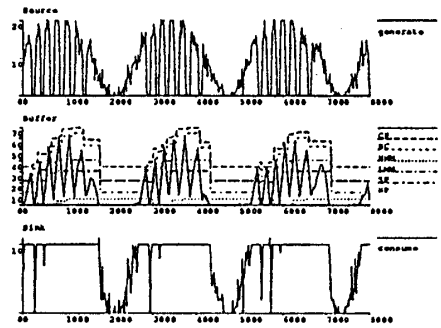

図 4: アルゴリズム 4(負荷変動あり)

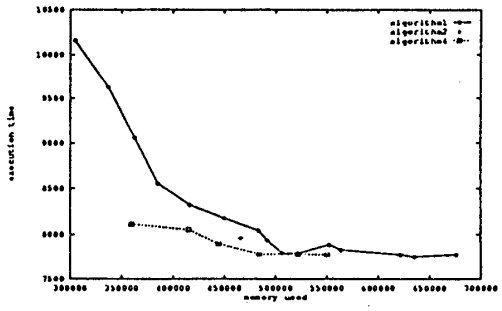

図 5: メモリ使用量対実行時間 (負荷変動あり)

図は階段状の曲線を有する前半部分と定常的な後半部分とからなっていることがわかる。定常的な後半部分では、CP は 0 と BC の間を移動し、0 や BC と重なることはない。このような状態では Sink の曲線は一定値をとり、Sink は最大性能でコンテナを消費することができる。一方の階段状の前半部分では、最適な SP,RP,BC を探索して試行錯誤をしながら段々と各曲線が移動してゆく。ポイントが適切でないうちは Sink のコンテナ消費が落ち込むことがある。これは実行時間の延長を意味する。

収束を加速するには HWM、LWM を推測する精度を上げることと δ の値域を広げることが必要である。HWM について、前回の SP が小さすぎて、$HWM = BC$ となっている状況考えてみると、次回の HWM はこの HWM を基準に推測しているため、実際の値よりも小さくなっている。すなわち、再びバッファフルが発生する可能性が高い。そこで、CP 曲線を外挿して「真の HWM」を推測し、次回に期待される HWM に反映する。LWM の推測も同様である。

以上の改良を施した適応化アルゴリズム 3 の実行時間は、改良前が 6197 クロックなのに対し、改良後が 6105 クロックと、オーバーヘッドは約半分になった。(理論上の最短実行時間は 6000 クロックである。)

アルゴリズムの妥当性を適応化をせぬ場合と比較した結果を図 3に示す。横軸はバッファの総使用量 (各クロックでのバッファの割当量×実行時間のクロック数)、縦軸は実行時間である。直線でプロットしてあるデータはアルゴリズム 1 によりバッファを固定的に割り当てた場合、独立してプロットしてある 2 点は適応化する場合で、真の LWM,HWM を推測しないアルゴリズム 2 と推測するアルゴリズム 3 である。この結果から、適応化のためのコストが若干懸かってはいるものの、最短に近い実行時間と適度なバッファの大きさが得られていると言える。

4 負荷が変動する場合における適応化アルゴリズムの評価と改良

次に、流入するデータ量が変動する場合について評価を行う。負荷変動のモデルとして source が発生するコンテナ数は時間に対して単振動をする発生確率を有するものとする。

データ量一定の時と同様に、妥当性の検討を行う。図 5にバッファを固定的に割り当ててバッファの大きさが動的に変化しないモード (アルゴリズム 1) との比較結果を示す。アルゴリズム 4 は $CP = 0$ が続くと BC の初期値を再設定する。

アルゴリズム 2 およびアルゴリズム 4 による曲線はアルゴリズム 1 による曲線よりも原点寄りに分布しており、より少ないバッファ量で同等以上の実行時間の短縮を実現していることがわかる。また、アルゴリズム 1 ではバッファ量が小さくなると急激に実行時間が長くなるのに対して、アルゴリズム 4 では BC の初期値はそれほど実行時間に影響を与えない。

5 まとめ

スーパーデータベースコンピュータ (SDC) において、負荷に応じてバッファの容量を適応的に制御することにより、より少ないバッファ容量で実行時間の短縮を図る方式を示した。シミュレーションにより、適応化アルゴリズムは負荷変動が無い場合には若干のコストがかるもののほぼ最適なバッファ容量と実行時間を発見できること、大きな負荷変動がある場合にはバッファの容量を固定的に割り当てておくアルゴリズムと比較してより少ないバッファ容量で同等以上の実行時間を実現可能であることを確認した。

参考文献

[1] 平野, 原田, 中村, 相場, 鈴木, 喜連川, 高木. スーパーデータベースコンピュータ sdc に於けるデータ流制御方式. 情報処理学会第 *43* 回全国大会, 1991.