7H−1

スーパーデータベースコンピュータ第二版 (SDC2) におけるシステムソフトウェアの構成

中村 稔 田村 孝之 喜連川 優 高木 幹雄
東京大学生産技術研究所

1 概要

我々は現在スーパーデータベースコンピュータ第二版(SDC2)を開発中である. 本システムはスーパーデータベースコンピュータ第一版 (SDC1) に対して各モジュールの処理性能の向上と高機能オメガネットワークによる多モジュールの相互接続を行なった機構をとっており現在その評価を進めている. 本論文ではまず,SDC2 のアーキテクチャーを示す. 次に SDC2 上でのシステムソフトウェアの構成について述べる. さらに SDC2 におけるデータ流の制御とデッドロックの回避方法について述べる.

2 SDC2 の構成

SDC2 の構成を図 1 に示す.

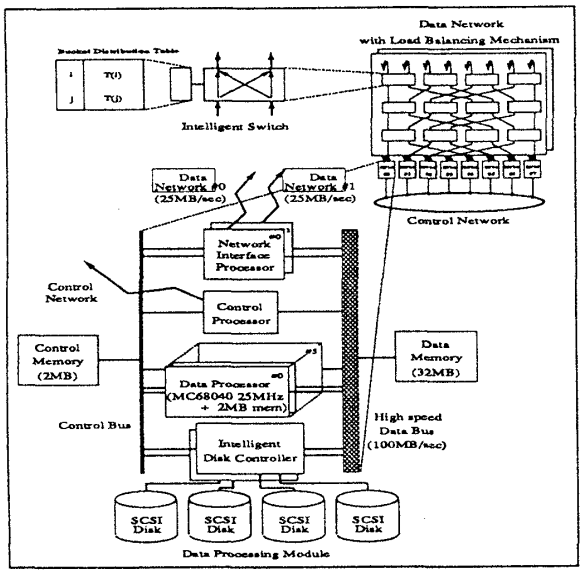

図 1 SDC2 の構成

SDC2 は最大でプロセッサ 7 台と磁気ディスク装置 4 台を密に結合した処理モジュール並びに複数の処理モジュールを疎に結合する高機能オメガネットワーク [1] からなるハイブリッドアーキテクチャをとる. この構成では密結合の利点である軽い通信コストによる高速性とモジュール数の増減によるスケーラビリティが同時に得られる.

SDC2 は SDC 第一版 [2, 3] に対して以下のような特徴を持つ.

- 処理モジュールの性能向上と小型化
- ディスクコントローラおよびネットワークコントローラにそれぞれ専用の制御プロセッサを用い, Control Processor の負荷を軽減
- データネットワークの二重化と性能向上並びに, 機能の追加
- ハードウェア/ソフトウェアの可変長タプルへの対応

SDC1 と SDC2 との比較を以下に示す.

	SDC1	SDC2
Processor	68020/20MHz × 4	68040/25MHz × 7
Data Memory	8MB	32MB
Data bus band width	40MB/sec	100MB/sec
Disk drive	2 台 /module	4 台 /module
Data Network	10MB/sec	25MB/sec
Number of modules	2	8

3 SDC2 のシステムソフトウェアの構成

SDC2 におけるソフトウェアの構成を図 2 に示す.

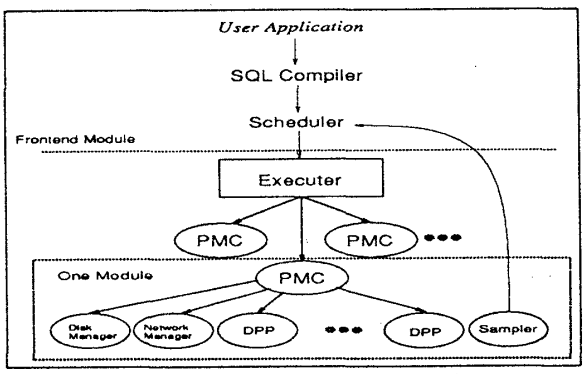

図 2 モジュール内のソフトウェアの構成

SDC2 では, モジュール上のプロセス群とフロントエンド上の SQL コンパイラ, スケジューラーによって, 以下のように処理が進行する.

1. アプリケーションから SQL の形で発行された問い合わせはフロントエンド上でコンパイルされ静的最適化が施されたのち Scheduler に渡される.
2. Scheduler では実行中の処理やファイルの配置, Sampler からの情報などを元に動的最適化を行ない適当なモジュール群に問い合わせを発行する.
   Sampler はモジュール内の各プロセッサの稼働率やメモリーの消費率に関するデータを収集し定期的にフロントエンドモジュールに報告する. このプロセスは実際のデータ処理

Structure of system software of the Super Database Computer version II (SDC2)
M.Nakamura, T.Tamura, M.Kitsuregawa and M.Takagi
Institute of Industrial Science, University of Tokyo

には関与しないが, 処理の動的最適化の際のヒント情報としてフロントエンド上で利用される.

3. Scheduler からのコマンドを受けて Executer は処理を実行するモジュール群の PMC にコマンドを送る. また処理の内容に応じて適宜モジュール間の同期をとる.
4. PMC はコマンドの内容に応じて Diskman, Netman, DPP の起動を行ない, 同期をとりながら処理を進める. また, モジュール間の同期が必要になった場合には Executer に同期の要求を出す.
5. DPP はディスクおよびデータネットワーク からのデータを実際に処理し処理結果を再びディスクまたはデータネットワークに出力する.

4 SDC2 におけるデータ流制御

SDC ではデータメモリを固定長のページに分割しモジュール内でのデータ交換はすべてこのページを受け渡すことで行なわれる (図 3). ページの受渡しはコントロールメモリ上のバッファ構造体を通じて行なわれる. ページ内には複数のタプルが格納され, ページ長を越えなければタプルの長さに制限はない. また, DNET を用いたモジュール間のデータ交換はタプルを単位として行なわれ, ページからタプルへの変換およびその逆変換は DNET インターフェース上で行なわれる.

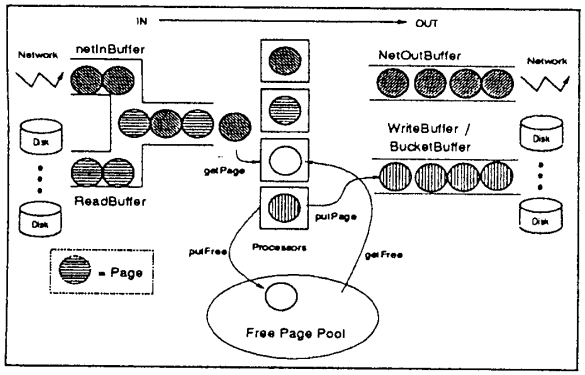

図 3 モジュール内のデータの流れ

SDC 第一版ではバッファ毎にフリーページをあらかじめ割り当てていたのに対し SDC2 では モジュール内に単一のフリーページプールを用意しすべてのバッファーはこのフリーページプールを共有するようにした. その結果, モジュール内のフリーページが一元的に管理できるようになり, データメモリを効率良く利用できるようになった.

SDC2 で用いられるバッファ構造体はページが流れるパイプとみなすことができる, バッファの両端にはそれぞれ, ページの生成者, 消費者となるプロセスが一つ以上存在する. バッファは別々のプロセッサ上のプロセスをつなぐ役割を果たし, 同時にデータのフローコントロールとデッドロックの回避を行なうポイントでもある. バッファへのページの供給と取り出しはそれぞれ, putPage(), getPage() ライブラリ関数によって行なわれる. それぞれのバッファには同時に格納できるページの最大数が指定でき, putPage の際に最大ページ数を越えていた場合は, putPage を呼び出したプロセスはブロックされ, ページが消費されて最大値を下回った時点で処理が再開される.

5 デッドロックの回避

SDC2 におけるデッドロックはフリーページが消費し尽くされることで発生する. SDC2 ではページの生成者は同時にフリーページの消費者であり逆にページの消費者はフリーページの生成者であるため, デッドロックを回避するためにはフリーページがなくなる前にページの生成を中断し消費者によってページを解放させればよい.

SDC2 ではディスクがデータの最終的な生成者であり消費者であるが, 同時には生成者か消費者かの一方にしかなれない. このためデッドロックを回避する場合の基本的戦略はディスクからのデータ読み出しを中断して出力結果または中間ファイルをディスクに書き出すことである.

フリーページプールにはデッドロック回避を開始するページ数 (lowWaterMark) と中断した処理を再開するページ数 (highWaterMark) があらかじめ指定されている. Diskman はデータの読みだしにおいてフリーページを獲得する際にフリーページの残量が lowWaterMark を下回るとディスクからの読み出しを中断する. さらに, モジュール内の各バッファが保持しているページの量を調べ, その結果に応じて以下のようにしてデッドロック回避を行なう.

- **writeBuffer に highWaterMark 以上のページがある場合**
  - writeBuffer の内容をディスクの出力ファイルに書き出す.
- **bucketBuffer のページの合計が highWaterMark を越える場合**
  - bucketBuffer の内容をディスクの中間ファイルに書き出す.
- **writeBuffer および bucketBuffer の合計が highWaterMark を越えない場合**
  - これは他のモジュールのデータ消費が遅いために netOutBuffer 上のページが消費されないために発生する. この場合 Diskman はフリーページが highWaterMark を上回るまでアイドル状態となる.

6 まとめ

以上, SDC2 のシステムソフトウェアの構成とデーター流制御ならびにデッドロック回避の方法について述べた. 現在 SDC2 の試作と評価を進めている.

参考文献

[1] 田村, 中村, 喜連川, 高木「スーパーデータベースコンピュータ (SDC2) におけるデータネットワーク系の実装」SWoPP '93

[2] 平野, 原田, 中村, 相場, 鈴木, 楊, 喜連川, 高木「スーパーデータベースコンピュータ (SDC) におけるモジュール群制御方式と 2 モジュール SDC の試作・評価 」並列処理シンポジウム, 1991

[3] 平野、原田、中村、小川、楊、喜連川、高木「スーパーデータベースコンピュータ SDC のアーキテクチャ」並列処理シンポジウム, 1990