計算機アーキテクチャ　128－12
ハイパフォーマンス
コンピューティング　70－12
（１９９８．３．５）

ワークステーションクラスタのための通信／同期コントローラ

早川 潔　　関口 智嗣

電子技術総合研究所
〒305-8568 茨城県 つくば市 梅園 1-1-4
E-mail: hayakawa@etl.go.jp

概　要　ワークステーションクラスタを使用して効率良い並列処理を行うためには，同期や通信のオーバーヘッドを低減することが不可欠である．特に，同期処理では，同期処理の要求が送られてから応答が帰ってくるまでのレイテンシを短くすることが問題となる．しかし，ワークステーションクラスタに使用されるネットワークは，情報量のスループットのみに重点がおかれている．よって，そのネットワークを使用して同期を処理する場合，同期のレイテンシが長くなってしまう可能性が高い．そこで，本稿では，汎用ネットワークとは別にレイテンシ重視の同期ネットワークとスループット重視の通信ネットワークを設け，そのネットワークを使用して同期／通信処理を可能にするコントローラを検討する．

The Communication/Synchronization Controller for Workstation Cluster

Kiyoshi Hayakawa　　Satoshi Sekiguchi

Electrotechnical Laboratory
1-1-4 Umezono,Tsukuba Ibaraki 305-8568, Japan
E-mail: hayakawa@etl.go.jp

Abstract　It is necessary to reduce overheads of synchronizations and communications to perform efficient parallel computation with workstation cluster. Specially, in the synchronizations, the latency which is period between sending the synchronization request and return the response is the most critical issue. But, the network which is used by workstation cluster allows PEs to communite data with high throughput(less care of the latency) . Therefore, when PEs synchronize thurough the network, it is difficult to reduce the synchronization latency. In this paper, we describe the communication/synchronization controller which allows PEs to synchronize with shorter latency using the synchronization network and communicate large amount of data using the communication network.

1 はじめに

数十台のワークステーションをATMなどの高速ネットワークで結合したワークステーションクラスタを構成することが可能になっている.このワークステーションクラスタ上で,効率良い並列処理を行うためには,高スループットなデータ転送,低レイテイシな同期処理が不可欠である.

通信や同期の処理をTCP/IPを利用して実現した場合,高スループットなデータ転送や低レイテンシな同期処理を行うことが難しい.この問題を解決するために,TCP/IPを介さず,デバイスドライバを作成し,直接ネットワークハードウェアを制御し,データ通信や同期処理を行う研究がなされている[5].また,Myrinetで接続したワークステーションクラスタでの通信ライブラリが開発されている[6].

本コントローラも高スループットなデータ転送や低レイテンシな同期処理を実現することを目的とする.本コントローラでは,高スループット・低レイテンシを実現するために,データ転送のリンクと同期処理のリンクを分け,データ転送のリンクでは,大きなパケットを流すことにより高スループットを実現し,同期処理のリンクでは,小さなパケット流すことにより,低レイテイシを実現する.

本稿では,本コントローラのハードウェア構成および本コントローラ上で行われる同期／通信の処理について説明する.

2 通信／同期コントローラ

2.1 ワークステーションクラスタ構成

本コントローラを実装したワークステーションクラスタを図1に示す.各ノードには,PCIバスが搭載されており,このバスに通信/同期コントローラを実装する.高速な通信/同期を実現するために,Ethernetなどのネットワークとは別に,通信/同期コントローラのためのネットワーク（Sync/Comm Network）を設ける.Sync/Comm Networkのトポロジは,2次元トーラスとする.

2.2 通信/同期コントローラ構成

通信/同期コントローラのハードウェア構成を図2に示す. 通信/同期コントローラは,Responsive Processor（RP）[1]と集中･分散Barrier Controller(集中・分散BC)からなる.RPは,ノードとPCIバスを介して接続され,集中BCはLocal I/Fに接続されている分散BCを介して各RPと接続されている.集中・分散BCはバリア型同期機構を実装するために設ける.

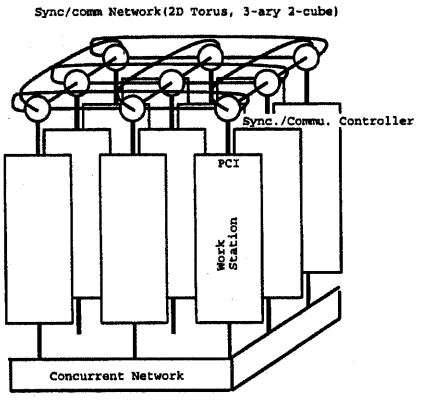

図 1: ワークステーションクラスタの構成 (9 ノード時)

2.2.1 Responsive Processorの構成

RPは,並列分散リアルタイム処理のために開発されたプロセッサである[1].RP内には,RISC Core（SPARC lite）,高速リンク,USB･SIO･PCI･SDRAMの各I/F, タイマなどが実装されている.

RP内のLink（Responsive Link）は,大量データ転送を可能にするためのデータパケット用Linkとハードリアルタイムを実現するためのイベントパケット用Linkに分かれている.リンク速度は,データ／イベントともに最高100Mbpsである.

3 同期処理

本コントローラでサポートする同期処理は,RPのみで行われる同期処理と分散・集中BCが行う同期処理にわかれる.RPのみで行われる同期処理は,主に排他制御であり,分散・集中BCで行われる同期処理は,バリア型[3]の同期処理ある.

3.1 RPのみで行われる同期処理

RPのみで行う同期処理は,条件同期,普通のバリア,mutex,Non-Collision Lock(NCL),カウンティ

ングセマフォとする.各同期命令は,デバイスファイルの入出力命令を利用する.

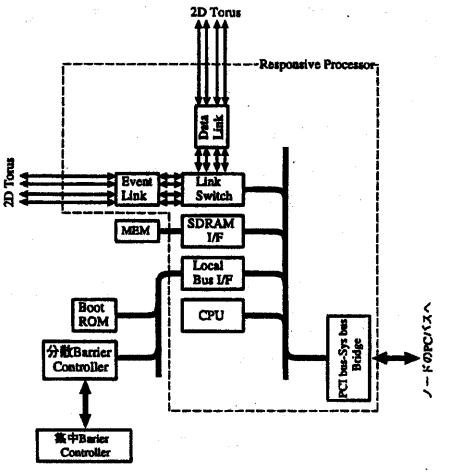

図 2: 通信同期コントローラのハードウェア構成

3.1.1 イベントパケット

RP のみで行われる同期処理では,Responsive Link のイベントリンクを利用する.ハードリアルタイムを実現するために,イベントリンクを流れるパケット（リンクパケット）は,必ず 1 ホップを 3 μ s 以内で通過することが可能である.また,イベントパケットのパケットサイズは,16Byte と比較的小さいので,レイテンシを短くできる.よって,イベントパケットを利用して同期を行うことにする.

同期処理で使用するイベントパケットのパケットフォーマットを図3に示す.

3.2 同期処理の動作

本節では,本コントローラの特徴であるバリア同期の通信方法,NCL の同期方法について説明する.

3.2.1 バリア同期

バリア同期には,バリア同期プロセスの実行開始時刻をある程度合わせる同期と合わせないバリア同期を用意する.

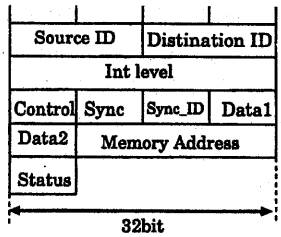

図 3: イベントパケットフォーマット

プロセスの実行開始時刻をある程度合わせるバリア同期 この同期方法は,RP のイベントパケットが必ず 1 ホップ 3 μ s で通過できるという特徴を利用する.

この同期は,Master/Slave 方式で同期通信を行い,同期成立を通知するとき,Master は,各 Slave までのホップ数を計算し,一番遠い Slave にパケットが届くまでの待ち時間を各 Slave に知らせる.

プロセスの実行開始時刻を合わせないバリア同期 プロセスの実行開始時刻を合わせないバリア同期のための通信アルゴリズムとして,次の 3 つのアルゴリズムを用意する.

- Master/Slave（任意台数）
- Shuffle Exchange（2〜16 台用）
- 変形 Shuffle Exchange（17 台〜36 台）

以下に,変形 Shuffle Exchange アルゴリズムの手順について示す.

1. write 命令を実行し,条件ビットが配置されているノード ID（ID）,同期の種類（sync）,同期処理の ID（sync_ID),を RP の特定のメモリに書き込む.書き込みが終了した後,write 命令を終了し,スリープする.

2. RP は書き込みを検出し,同期命令を解読する.偶数列ノードの RP は,図 4では右隣のノードの RP,奇数列ノードの RP は左隣のノードに同期ポイントに到達したことを通知する（図 4（1）参照）.

3. 2. の送受信を行った RP のうち偶数列ノードの RP は 2. の手順が完了したことを左隣の RP に通知する.その通知を受けとった RP は,左

隣ノードのRPに右隣から通知が届いたことを知らせる．また，奇数列ノードのRPは2.の手順が完了したことを右隣のRPに通知する．その通知を受けとったRPは，右隣ノードのRPに左隣から通知が届いたことを知らせる（図4（2）参照）．

4. 2.・3.の通知を送受信し終えたRPのうち，偶数行のRPは，下のノードに上記の操作が完了したことを通知し，奇数行のRPは，上のノードに上記操作の完了を通知する（図4（3）参照）．

5. 2.・3.・4.の通知を送受信し終えたRPのうち，偶数行のRPは，上のノードのRPに上記の手順全てが完了したことを通知する．その通知を受けとったRPは，上のノードのRPに下から届いた通知を知らせる．また，奇数行のRPは，下のノードのRPに上記の手順全てが完了したことを通知する．その通知を受けとったRPは，下のノードのRPに下から届いた通知を知らせる（図4（4）参照）．

6. 2.・3.・4.・5.の通知を全て受けとったRPは，スリープをしているプロセスをレジュームさせる割り込みをかける．

3.2.2 Non-collision Lock(NCL)

Lock操作では，ロック変数アクセス時に，プロセス同士が競合してしまう場合がある．それによって，ネットワークの負荷が増大する可能性がある．そこで，ロック獲得を試みたプロセスを，試みた時刻が早いもの順に，ロックを獲得させる操作—Non-Collision Lock(NCL）—を用意する.NCLは，基本的には，MCS[2]と同じ動作を行うが，待ち時間の実現方法が異なる．

NCLにおけるロック獲得の手順を以下に示す．

1. 準備として，ロック獲得を試みたプロセスの順番を示すFIFOを，クリティカルセクションをもつノードのRPのローカルメモリに用意する．

2. クリティカルセクションにアクセスしたいプロセスは，RPに対して,NCL操作のロック獲得を示すWrite命令を実行し，プロセスをスリープさせる．

3. NCL操作のロック獲得要求を示すWrite命令を受けとったRPは，クリティカルセクションがあるノードのRPにNCL操作のロック獲得要求を示すイベントパケットを送出する．

4. イベントパケットを受けとったRPは，ノード番号・Sync_IDをFIFOに投入する．もし，投入前にFIFOが空の場合,FIFOにノード番号・Sync_IDを投入した後,FIFOの先頭のノードにロックを獲得したことを示す応答イベントパケットを送出する．

5. ロック獲得要求を送出したRPは,NCLによるロック獲得応答を受信した場合，該当プロセスのスリープを解除する．

6. スリープを解除させられたプロセスは，クリティカルセクションを実行する．

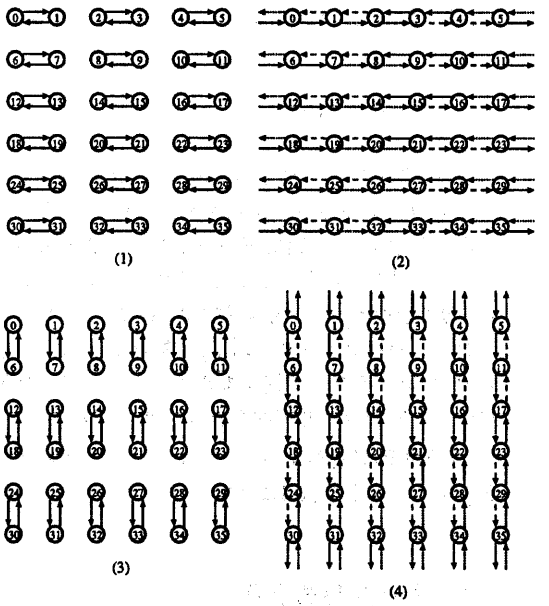

図4: プロセスの実行開始時刻を合わせないバリア同期におけるプロセッサ間通信

NCLにおけるロック解放の手順を以下に示す．

1. クリティカルセクションのアクセスを終了したプロセスは，RPに対して,NCL操作のロック解放を示すWrite命令を実行し，次の命令に移行する．

2. NCL 操作のロック解放を示す Write 命令を受けとった RP は, クリティカルセクションのあるノードに NCL 操作のロック解放要求を示すイベントパケットを送出する.

3. イベントパケットを受けとった RP は,FIFO を 1 つ前に移動し（FIFO の先頭は消去）,FIFO の先頭になったプロセスにロック獲得を通知する NCL の応答パケットを送信する. また, ロック解放要求を送信した RP にロック解放応答を送信する.

3.3 分散・集中 Barrier Controller を使用した同期処理

図 5に分散 Barrier Controller 周辺の構成図を示す.

このコントローラを使用した同期命令の待ち状態処理は, 以下の手順で行われる.

1. プロセスは,Write 命令を用いて,PCI バスを介して Local Bus につながっている分散 Barrier Controller にデータを書き込み, スリープ状態に移行する（図 5参照）.

2. 分散 Barrire Contorller が Local Bus の Address 出力をデコードし, Adress に対応した同期処理を行う.

3. 同期が成立したら,Local バスを介して,RP 内の割り込みコントローラ（IPC）へ割り込み信号を送出する.

4. IPC が CPU Core に割り込みをかけ,CPU Core が PCI を介して,PE に割り込みをかける.

5. 割り込みがかかった PE は, スリープしているプロセスをレジュームさせる.

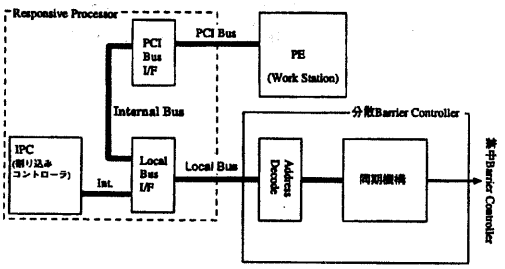

図 5: 分散 Barrier Controller 周辺の構成図

分散・集中 Barrier Controller を利用して, ハードウェアで処理させる同期は, 重複可バリア [4] と RBCQ-1 同期機構である. 本節では,RBCQ-1 同期機構について提案する.

3.3.1 RBCQ-1 同期機構

従来のバリア領域を設定できる同期機構では, 入口同期コードと出口同期コードの 2 種類の同期コードが必要となる.RBCQ 同期機構 [7] では,1 種類の同期コードでバリア領域を表すことにより同期を行う. また, 同期要求信号から応答信号を生成するロジックの段数が比較的少ないので, 高速な動作が可能である.

しかし,RBCQ 同期機構では, 同期に参加するか否かを示す情報（バリア参加情報）をバリアキューの格納しておかなければならない. このため, 他の同期機構に比べてハードウェアコストが大きくなってしまう. そこで, バリアキューのかわりに, 参加しない同期の回数をカウンタするカウンタを追加することにより, ハードウェアコストを少なくした同期機構として,RBCQ-1 同期機構を搭載する.

図 6にハードウェア構成を示す. ノードは, 同期コードを実行する際,PE Wait 信号を送出し,PE Go 信号が返ってくるまで, 待ち状態となる.

PE Wait 信号送出時に, (そのノードが同期に参加しない回数＋1) を Counter に送出する.

RBCQ 同期同期機構では, 以下の処理が行われる.

- キューの先頭 (BQ-Head) のバリア参加情報とプロセッサが送出する PE Wait 信号との AND を PE Go 信号とする. また,PE Wait 信号がアクティブ (High) かつカウンターの値が 0 の場合,Set_d の値をカウンタにセットする.

- これと同時に,BQ-Head に格納されているバリア参加情報内で,PE Go 信号が送出されたプロセッサのビットのみを, 同期成立検出回路がクリアする.

- BQ-Head の内容が全て 0 になった時点で, 全ノードの Counter の値から 1 を引く.Counter の値が 0 になったノードの BQ-Head のみを 1 にセットする.

なお,Counter の初期値を決定するため, プログラムの最初で, 全てのノードが参加する同期を行う.

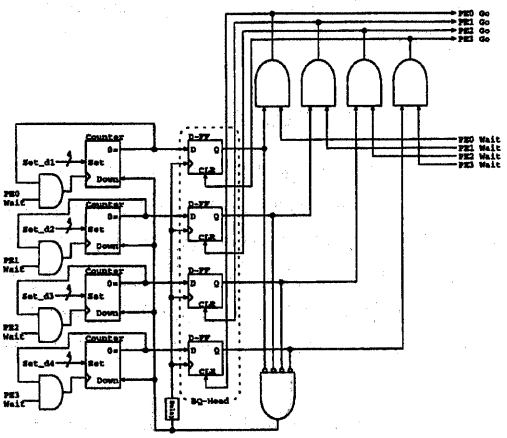

図 6: RBCQ-1 のハードウェア構成（PE 4 台時）

4 メッセージパッシング通信処理

各 PE 間でデータ通信を行うために, 準備として, ローカルメモリの一部を送信用のバッファと受信用のバッファに割り当てる (複数の送受信バッファを設定可能にする) . データ転送は, データリンクを使用する.

メッセージパッシングの手順を以下に示す.

1. 送信側ノードは,RP のローカルメモリに送信バッファを作成し, 送信データを指定されたバッファに一時格納した後, 送信命令を RP に送る.
2. 送信命令を受信した RP は, レスポンシブリンクのデータパケットを使用して, 指定された送信バッファのデータを送信する.
3. 受信側のノードは,RP の受信バッファを検査し, データが届いたことを示す検査結果が返って来た場合, 受信側のノードは, 受信バッファにあるデータを受けとる.

5 おわりに

データ転送では高スループットなデータ転送を行い, 同期処理では低レイテンシな同期処理を可能にする通信／同期コントローラについて説明した. 本コントローラを使用することにより, 粒度の細かくより密に協調した並列処理を効率良く行うことが可能になると考える. 今後は, このコントローラをアルファワークステーションクラスタ「etlwiz」に実装し, 初期性能評価を行う予定である.

参考文献

[1] 山崎信行, 松井俊浩, “機並列分散リアルタイム制御用レスポンシブ・プロセッサの設計” 第 15 回日本ロボット学会学術講演会, No.3, pp.1027-1028, 1997.

[2] John M. Mellor Crummey and Michael L. Scott, ” Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors” ACM Trans. on Comp. Sys., Vol.9, No.1, pp.21-65, Feb.1991.

[3] 高木 浩光, 有田 隆也, 曽和 将容, “細粒度並列実行を支援する種々の静的順序制御方式の定量的評価” 並列処理シンポジウム *JSPP'91* 論文集,pp.269-276, May.1991.

[4] 高木 浩光, 有田 隆也, 曽和 将容, “重複可能なバリア型同期のためのスケジューリングアルゴリズムとその性能” 電子情報通信学会技術研究報告 CPSY91-15,pp.91-98,1991.

[5] 松本 尚, 平木 敬, “100BaseTX によるメモリベース通信の性能評価” 電子情報通信学会技術研究報告 CPSY97-50, pp.109-116, 1997.

[6] 手塚 宏史, 堀 敦史, 石川 裕, “ワークステーションクラスタ用通信ライブラリ” 並列処理シンポジウム JSPP'96 論文集, pp.41-48, 1996.

[7] 早川 潔, 本多 弘樹, “RBCQ 同期機構およびその同期方式の提案と性能評価”, 並列処理シンポジウム *JSPP'97* 論文集, pp.45-52, May.1997.