計算機アーキテクチャ 144－27
（２００１．７．２７）

レスポンシブリンクを用いた 実時間処理用分散共有メモリの設計と実装

鈴木 誠　森実 裕司　小林 秀典　山崎 信行　安西 祐一郎

慶應義塾大学大学院 理工学研究科 開放環境科学専攻

E-mail: {makoto, morizane, kobahide, yamasaki, anzai}@ayu.ics.keio.ac.jp

パケットの優先度に応じた追い越し機能を持つレスポンシブリンクを用いて、実時間処理用のソフトウェア分散共有メモリ (Distributed Shared Memory) の設計及び実装を行う。分散実時間システムにおいては、異なるノードに存在するタスクが協調して実行する処理を、ある時刻までに終了させることが重要となる。しかし、ネットワークの負荷が動的に変動するような環境では、それらのタスクのデッドラインを保証することは困難である。本機構ではタスクの優先度をレスポンシブリンクのパケット優先度にマップする事でネットワークの動的な負荷の変動に対処する。また、HOME をベースとしたメモリモデルを採用することで、リモートアクセス時の遅延を一定となるようにする。このようにすることで、分散実時間システムにおいて優先度の高いタスクの分散実時間処理をサポートする。

Design and Implementation of a Distributed Shared Memory on *Responsive Link* for Real-Time System

Makoto Suzuki, Hiroshi Morizane, Hidenori Kobayashi, Nobuyuki Yamasaki, Yuichiro Anzai

School of Science for Open and Environmental Systems
Graduate School of Science and Technology, Keio University

In this paper, we design and implement distributed shared memory on *Responsive Link* for distributed real-time system. *Responsive Link* supports priority packet and high priority packets can get ahead of low one when conflict in switch. In distributed system, many tasks which is physically distributed communicate with each other. But predicting communication latency, which is important for real-time applications, is difficult especially where network traffic changes dynamically. In our system, we deal with these traffic by assigning the priority of threads to the packets. And adopting home based memory model supports fixed latency with the remote memory access. In this way, we try to support the execution of task which has high priority in distributed real-time environment.

1. はじめに

近年、ロボット制御、自動車制御など発達において分散実時間処理の役割は重要なものとなってきている。特にロボット技術においては、ヒューマノイドロボットの登場などによりその存在が我々にとって身近なものとなってきている。

単一プロセッサでのシステムと異なり、分散実時間システムでは各ノード間でデータの受け渡しが生じ、またそれが時間的に予測可能でなければならない。テレビ会議システムやファクトリーオートメーションのように、ネットワークの負荷が事前にある程度既知の場合には、帯域保証といった資源予約をベースとしたアプローチも有効である。しかし自立移動ロボットのように実世界と相互作用を持つシステムでは、あらゆる事象を想定しなければならず、ネットワークにかかる負荷を完全に予測することは困難である。

レスポンシブプロセッサ [1] は各種実時間システムの並列分散制御を目的として開発されたシステムオンチップ (SoC) である。レスポンシブプロセッサの最大の特徴は実時間通信をサポートするレスポンシブリンクを持つことである。レスポンシブリンクはデータラインとイベントラインの分離、パケットの優先度による追い越し等の機能により実時間通信をサポートする。

一方で、分散共有メモリに関する従来の研究では、

オーバーヘッドの削減を目的とした、メモリモデル、コンシステンシモデル等の研究が多く、実時間性に関する研究はほとんど行われていない。

本研究では、タスクの優先度をレスポンシブリンクのパケット優先度にマップする事で、ネットワークの負荷が動的に変動するような環境においても、時間の予測が可能な分散共有メモリの実現を試みる。

2. 背　　景

2.1 *Responsive Processor*

レスポンシブプロセッサは、各種パーソナルロボットや、ホームオートメーション、オフィスオートメーションなど、様々な並列分散実時間処理用途に必要な機能を1チップに集積した、システムオンチップ (SoC) である。レスポンシブプロセッサは以下のような構成を持つ。

- プロセッサコア – SPARC lite
- *Responsive Link* – 用途別の2系統のリンク
- コンピュータ用周辺
  SDRAM I/F, DMA, PCI, USB, タイマ/カウンタ, SIO, PIO, RTC
- 制御用周辺
  A/D 変換器, D/A 変換器, PWD 発生器, パルスカウンタ

以上の機能を1チップに集積している。

レスポンシブプロセッサの最大の特徴は、実時間通信をサポートするレスポンシブリンクを持つことである。以下にレスポンシブリンクの特徴について説明する。

2.2 *Responsive Link*

レスポンシブリンクは、実時間通信をサポートするために、データのデッドラインに対する要求の厳しさの違いによって使用するラインを分離している。一般に、画像や音声などのデータは、データ容量が大きく、通信においてはバンド幅を保証しスループットの向上が望まれるが、システム内のイベント伝達のようなデータはレイテンシの保証が望まれる。そこで、レスポンシブリンクでは、1ラインをデータ伝達用 (Data Link) とイベント伝達用 (Event Link) の2系統に分離し (図1)、それぞれに以下のような特性をもたせることで、以上の2つの要求を満たすよう実装されている。

- Data Link
  - ソフトリアルタイム通信対象
  - パケットサイズ：64 byte 固定
- Event Link
  - ハードリアルタイム通信対象
  - パケットサイズ：16 byte 固定

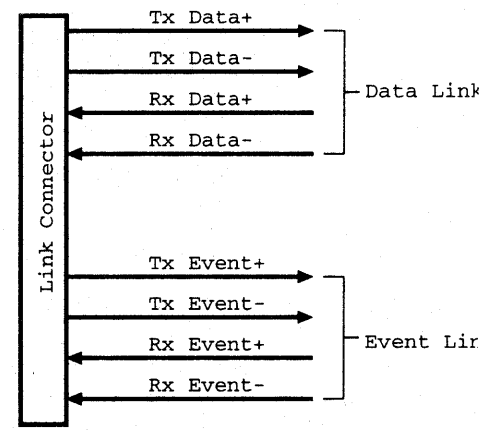

図1　Responsive Link Interface

Data Link と Event Link は独立したルーティングが可能である。レスポンシブリンクはカットスルー型のスイッチを採用することで、制御信号などの高いリアクティブ性の要求に対処している。さらに通信パケットに優先度を与えて、スイッチでの衝突の際にパケットの優先度による追い越しが可能である。リンクの優先度はリンクの宛先アドレス内に含まれる。そのために優先度ごとに違う経路を設定するといったことも可能となっている。また、その優先度はホップ毎にノード内で付け替えが可能である。リンクの接続は point-to-point で、トポロジーフリーな接続形態でネットワークをの構築が可能である。

2.3 *μ-PULSER*

μ-PULSER[2] は、本研究室で開発されてきたロボット用実時間オペレーティングシステムである。*μ-PULSER* はマルチタスク、マルチスレッド計算モデルのに基づき、柔軟性及び拡張性の高いマイクロカーネルにより実現されている。更にロボット制御において要求されるリアクティブ性を実現するために、ハードウェア割り込み、ソフトウェア割り込み、スレッド間同期、コンテキスト切り替え機構を統合し、透過的かつ高速に行えるようにしたコンカレントインタラプト機構 (CI 機構) を持つ。また、並列分散実時間システムに必要な実時間スケジューリング、リアクティブスケジューリング、モジュール間の通信機構等の機能を提供している。

3. 関連研究

分散実時間システムにおいて、メモリをベースとした通信機構に関する研究には、Reflective Memory[3]、RT-CRM(Real-Time Channel-Based Reflective Memory)[4] などがある。

SCRANet[5] は、ハードウェアベースの Reflective Memory であり、全ての共有データを一定時間ごとに置き換える。これらのハードウェアベースの reflective-memory システムではシステム全体で使用できる共有メモリのサイズが限られるので、スケーラビリティに問題がある。

RT-CRM は、センサ-アクチュエータ型の分散実時間システムを対象としたシステムであり、センサノード (Writer) とアクチュエータノード (Reader) の間にチャネルを張って、帯域保証をした片方向通信をサポートする。しかし、ロボットシステムのように、負荷が動的に変化し予測不可能な環境では、使用する帯域が予測不可能であるため適していない。

4. 設計及び実装

実時間処理用分散共有メモリを実現するために、分散実時間処理用オペレーティングシステム *μ-PULSER* に以下の機能を追加する。まず、リモートアクセスを検知するための MMU のトラップハンドラ。更にページリクエストの処理やキャッシュページの管理、変更したメモリの書き戻しといった処理を行う DSM マネージャ。そしてメモリの同期をとるためのロックマネージャ。また、それらのマネージャへのアクセスを行うライブラリ関数を実装する。DSM マネージャとロックマネージャは、カーネルレベルのスレッドとして実装する。

4.1 メモリモデル

多くのページベースの分散共有メモリシステムが COMA 型のメモリモデルを採用している。COMA 型のメモリモデルではページミスが起きたときに、そのページのオーナーを探索する必要がある。このようなメモリモデルではオーナーとは最後に書き込みを行ったプロセスである。オーナーの探索アルゴリズムには集中管理型と分散管理型があり、集中管理型の場合はページ管理者への問い合わせとなる。この場合ページ管理者がボトルネックとなりやすい。分散管理型の場合、ページのオーナーと思われるノードに問い合わせを行い、更にそのノードがオーナーでなかったら、その次のノードに問い合わせるというチェーンをたどることになる。ここでのオーバーヘッドは予測不可能となり、実時間システムには適切でない。

そこで本研究では HOME をベースとした NUMA のようなメモリモデルを採用する。HOME をベースとしたメモリモデルでは、共有されるページは固定したホームノードを持ち、リモートの共有ページをアクセスしたときはそれをキャッシュとして扱う。また、ページミスを起こした時にどのノードに取りに行くかが既知となるので、ここでのオーバーヘッドは予測可能である。

4.1.1 リード時の動作

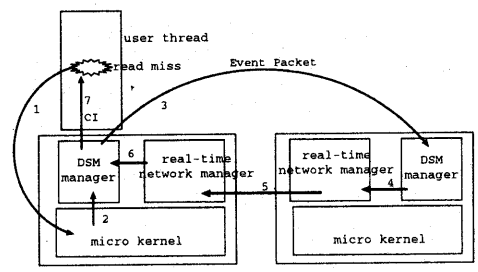

図2 リードミス時の動作

ページが所属するノードは 32bit 仮想アドレス空間の上位 8 ビットを用いて決定する。リードミスが起こると図 2のように動作する。まず、リードミスが起こるとハードウェア MMU により検知されトラップされる。トラップハンドラによりそのページがリモートかどうか、更にリモートならばキャッシュされているかどうかが調べられる。ページがリモートで更にキャッシュもされていなければローカルの DSM マネージャにページミスを知らせるメッセージを出す。DSM マネージャはそのページのホームノードの DSM マネージャに対してイベントリンクを用いてページリクエストを出す。ページリクエストを受け取った DSM マネージャはページを検索し、そのノードのリアルタイムネットワークマネージャに対して送信リクエストを出す。その際、ページミスを起こしたスレッドの優先度をリンクの優先度にマッピングしたアドレスを渡す。ページを受け取った DSM マネージャはそのページを Read Only として MMU にセットし CI 信号を用いてリードミスを起こしたスレッドを起床させる。

4.1.2 ライト時の動作

ライト時も、ローカルにページが無い場合、はリードミスのときと同じようにそのページの HOME にページリクエストを行なう。ページが到着すると DSM マネージャは、Read/Write の状態で MMU にセットする。また、DSM マネージャは、そのページの Twin

を生成し管理する。ライト時、ローカルにRead Onlyの状態でページが存在する時は、MMUによりトラップが起こり、トラップルーチンが、書き込もうとするページのTwinを作成し、Read Only状態からRead/Write状態に変える。

Twinページを作成し、書き込みを反映させる時に差分をとり、書き込みをした箇所だけHOMEに伝える。multiple writeプロトコルを採用することにより、false sharingによるオーバーヘッドを削減することが可能となる。

4.2 優先度のマッピング

レスポンシブリンクは4つの優先度を持つ。スレッドの優先度をレスポンシブリンクのパケットの優先度にマッピングすることにより、優先度の高いスレッドが要求するページは、ネットワークの負荷が高くても優先して送られてくることになる。

4.3 キャッシュページの管理

ページの転送単位、つまりキャッシュとして扱うページの大きさは1KBとした。レスポンシブリンクによる通信はDPM(Dual Port Memory)を通して行なわれるが、1回のDMAによるバースト転送でリンクに流すことの出来る容量に制限があるためである。ページ単位を1 KBとすることで、1回のDMA転送でページを転送することができ、ページ転送に伴うオーバーヘッドを削減できる。

キャッシュページも通常のローカルのページと同様に、MMUによりアクセス保護を行う。また、タスクごとにキャッシュページ管理用の構造体を持たせ、キャッシュしたページの仮想アドレス、ページの属性、物理ページナンバー、Twinページの物理ページナンバー、TLBのエントリナンバーなどを保持する。リードミス時やRead Onlyページから Read Writeページに変更するとき、また、ページの無効化要求がきたときにこの構造体が参照される。

4.4 一貫性プロトコル

本研究では、ホームベースのRelease Consistencyプロトコルを実装する。

Release時には、Releaseを行おうとしているスレッドのアドレス空間において、書き込みが行なわれたページを検索し、そのTwinページと比較し差分をとる。その後、差分はそれぞれのHOMEに送られ、HOMEのメモリに変更が書き込まれる。それぞれのHOMEに書き込みを行なった後、ロックマネージャにunlockリクエストを出し、Releaseは完了する。

Acquire時は、まず、これから入ろうとするクリティカルセクションに対応するロックをロックマネージャに要求する。ロックマネージャは、リクエストを受け取ると、リクエストを出したスレッドの優先度にしたがってキューイングする。ロックを獲得するとそのロックの以前のオーナーと比較され、その結果とともにロックを獲得したことをそのロックを要求しているスレッドに知らせる。そのロックの以前の所有者が異なる時、キャッシュページは無効化される。

4.5 ロック管理機構

ロックは、集中管理型とする。分散管理型のアルゴリズムでは、スケーラビリティーの点では有効であると考えられるが、通信に伴うオーバーヘッドが増加し、また時間的な予測可能性も低下すると思われるからである。

ロックマネージャはリクエストを受け取ると、要求されているロックが現在使用されているかどうかを調べる。使用されていなければ、ロックを要求しているスレッドに対しロックを獲得したことを知らせるメッセージを送る。その際、そのロックの以前のオーナーと比較しその結果も合わせて送る。ロックが使用中であったら優先度に従ってwaitキューに入れられる。優先度が同じ場合はFIFOとする。

4.6 プログラミングインターフェイス

プログラミングインターフェイスとして以下の関数を実装する。

- void DSM_Init(void)
- void DSM_Exit(void)
- void DSM_Lock(unsigned int lockid)
- void DSM_Unlock(unsigned int lockid)
- void DSM_Acquire(unsined int lockid)
- void DSM_Release(unsined int lockid)
- void DSM_UnCache(void)
- void DSM_WriteBack(void)

DSM_Init() は、初期化関数でありそのタスク用の構造体を用意する。DSM_Exit() はそのタスクが使用していたキャッシュページを解放し、またそのタスク用の構造体の領域も開放する。DSM_Lock(), DSM_UnLock() は、直接ロックマネージャと通信し、ロックの獲得、解放を行う。これらの関数は通常DSM_Acquire(), DSM_Release() 関数の中で用いられる。DSM_Acquire() はロックの獲得を行い、ロックマネージャからの返信を受ける。そのロックの以前のオーナーを調べ同じでなければDSM_UnCache() 関数により、キャッシュページの無効化を行う。DSM_Release() はDSM_WriteBack() 関数により、書き込みを行ったページの差分を取り、ページのホームノードに送り書

き込みを反映させる。

DSM_UnCache(), DSM_WriteBack() は、本来 DSM_Acquire(), DSM_Release() の中で用いられるもので、単独で用いるとメモリの一貫性を無くしてしまうこともある。しかし、分散実時間システムにおいては、周期的にセンサデータを読む場合などにキャッシュページをリフレッシュしたい等の要求が考えられる。そこでユーザがリモートページのキャッシュを管理できるようこれらの関数を用意した。

5. 評　　価

本論文が提案する実時間処理用分散共有メモリの評価を行う。性能は、ネットワークにトラフィックがあった場合に、タスクの優先度を考慮し、そのトラフィックよりも高い優先度のタスクがページアクセスを行った時と、タスクの優先度を考慮せずにトラフィックと同じ優先度でページアクセスを行った時のオーバーヘッドの比較を行う。

評価には、レスポンシブプロセッサ PCI 評価ボード (図 3) を用いる。ボードは表 1のような動作速度に設定した。

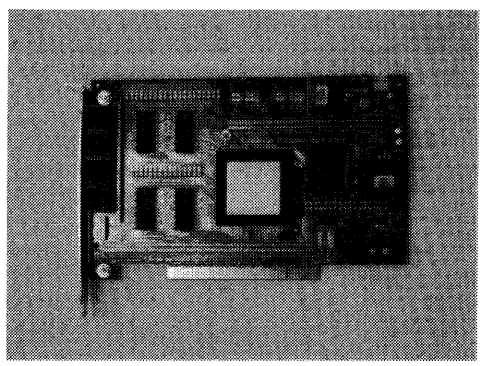

図 3　*Responsive Processor* PCI 評価ボード

表 1　評価における *Responsive Processor* の動作速度

CPU core speed	100 MHz
Responsive Link speed	6 Mbaud

なお、評価は 2 ノードを用いて行った。図 4に、ネットワークにトラフィックが無い状態でのページアクセスに伴うオーバーヘッドと、ネットワークに負荷をかけた状態での、スレッドの優先度の違いによるオーバーヘッドの比較を示す。

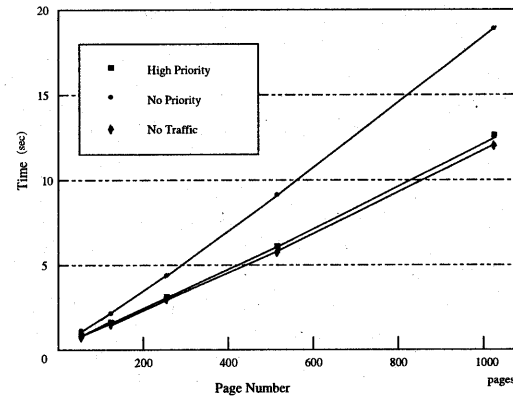

図 4　スレッド優先度の違いによるページ転送時間

トラフィックのパケットの優先度よりも、ページ転送のパケットに高い優先度を与えることで、トラフィックが無い場合と比べてほぼ同じオーバーヘッドで実現できていることが分かる。逆に、優先度を考慮せずにページ転送を行うと、トラフィックの影響を大幅に受けていることが確認できる。

測定結果より、ページ転送に伴うオーバーヘッドがかなり大きくなっていることが分かるが、これは安定性を重視してリンクの通信速度を一番遅く設定しているからである。本来の通信速度を用いればかなり改善できると思われる。また、オーバーヘッドを大きくしている原因として、*μ-PULSER* のネットワーク管理機構などが考えられるため、今後改善していく必要がある。

6. 結　　論

本論文では、分散共有メモリにおいて、ページ転送、書き戻しの際にスレッドの優先度をレスポンシブリンクの優先度にマップする事で、優先度の低いトラフィックに影響されずにページアクセスが行えることを示した。また、HOME をベースとしたメモリモデルを採用することで、リモートへのメモリ参照を固定のオーバーヘッドで実現させた。

今後の課題としては、通信に伴うオーバーヘッドを削減し、ノード数を増やして評価をとることなどが挙げられる。

参 考 文 献

1) Yamasaki, N. and Matsui, T.: A Functionally Distributed Responsive Micro Controller for Distributed Real-time Processing, in *Proceedings of the 1997 IEEE/RSJ International Con-*

ference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97, pp. 793–798 (1997).
2) 奥埜貢士：μ-PULSER における実時間通信機構の設計と実装, Master's thesis, 慶應義塾大学 (1999).
3) Jovanovic, M. and Milutinovic, V.: An Overview of Reflective Memory Ssystems, in *IEEE Concurrency Vol. 7, No. 2, April-June 1999* (1999).
4) Shen, C. and Mizunuma, I.: RT-CRM: Real-Time Channel-Based Reflective Memory, in *IEEE Transactions on Computers Vol.49, No.11, Novenber 2000* (1997).
5) Reflective Memory Nnetwork (1996), VME Microsystems Int'l Corp.