マルチメディア通信と分散処理　97－27
コンピュータセキュリティ　8－27
（２０００．３．２２）

VI Architectureによる分散リアルタイム環境の構築

落合 真一、村山 和宏、山口 義一
三菱電機(株)

暗号処理、音声・ビデオデータ処理、センサデータ処理など大量データ、大量演算を必要とする処理は、従来多数のDSPを組み合せて信号処理を行っていた。現在、このような演算エンジンを高速なインターコネクトで接続した疎結合計算機クラスタにより実現し、信号処理/情報処理を融合することが求められている。本論文ではリアルタイムOS上にVirtual Interface (VI) Architectureを実装し、高帯域低遅延のデータ交換が可能な分散環境を構築することにより、リアルタイム処理に必要な応答時間保証の実現性について検証を行った。評価の結果、VIAのソフトウェアオーバヘッド削減の有効性を確認できたが、応答時間の保証に関しては新たな課題も抽出できた。

Evaluation for Distributed Real-time Environment on VI Architecture

Shinichi Ochiai, Kazuhiro Murayama, Yoshikazu Yamaguchi
Mitsubishi Electric Corporation

DSP is used for massive data operation, such as ciphering, voice and video processing, and sensor processing. Although recently clustering of computers is acquired for such fields to integrate data processing and information processing. High-speed networks and micro-processors can be composed into powerful parallel processing engine. To realize that, we have been designing distributed real-time data processing system using VI architecture. In this work, we apply VI on real-time operating system, and evaluate response time of distributed data exchange. We have confirmed effectiveness of VI, but there need be some extension to guarantee certain response time.

1. はじめに

暗号処理、音声・ビデオデータ処理、センサデータ処理は、従来多数のDSPを組み合せて処理を行っていた。しかし、汎用マイクロプロセッサが高性能化し、かつSIMD命令が実装されるようになったことから、汎用プロセッサの並列化による信号処理と情報処理の融合が目指されている。これにより、データ処理アルゴリズムの高度化、データ相関の抽出、処理の動的負荷分散などの要求に対応できる。

汎用ネットワークの高速化やPC技術の急速な進歩により、安価で高性能な計算機のクラスタ構築が可能になったことから、このような演算エンジンを疎結合計算機クラスタにより実現するアプローチがある。計算機クラスタでは、システム構成の柔軟性やスケーラビリティが実現できる。

本論文では、汎用の高速ネットワークとその通信技術としてVI (Virtual Interface) Architecture

を採用し、分散リアルタイム環境の構築、評価を行う。特にプロセッサ間データ交換の応答性に注目し、疎結合計算機クラスタで応答時間保証を必要とする並列データ処理エンジンの実現性を検討する。

2. 背景

2.1. VI Architecture

ネットワークハードウェア技術の急速な成長により、ギガビット、10 ギガビットクラスの高速ネットワークが実現されている。このような高速ネットワークでは、ソフトウェアオーバヘッドがボトルネック要因として大きく現われる。例えばギガビットネットワークを使用しても、汎用プロトコルである UDP では、アプリケーションは 300Mbit/秒から 400Mbit/秒程度の転送性能しか得られない。これは以下の問題による。

- データコピー：データ転送、保護がカーネルによって行われるため、ユーザ空間、カーネルバッファ間でデータコピーが行われること。
- トラップオーバヘッド：データ転送要求、完了通知などほとんどの処理がシステムコールや割込みなどのシステムトラップを必要とすること。

このような課題を解決する手段として、OS の関与無しにプロセスがネットワーク通信を行えるユーザレベル通信に関する研究が行われており、コーネル大学の U-Net[1]や UCB の AM、イリノイ大学の FM[2]、RWCP の PM[3]などがある。その中でマイクロソフト、コンパック、インテルは、ユーザレベル通信の研究成果を生かし、VI Architecture (VIA)[4]として標準仕様を公開した。VIA の特徴を以下に示す。

- ハードウェアサポート可能なプロトコル
- 仮想空間マッピングによる資源保護実現
- コネクションベース通信の提供

最近では VIA に準拠した製品が登場しており、VI をハードウェアでサポートした NIC では、アプリケーションがハードウェア性能の 6 割近くを利用できることが示されている[5]。

2.2. 本研究の目的

VIA のようなユーザレベル通信では、広帯域、低遅延の通信が可能であり、しかも OS 資源の競合を回避できるので、プロセッサ間データ交換において応答時間保証が可能と予想される。そこで、本研究ではリアルタイム OS 上に VIA を実装し、疎結合計算機クラスタ上での応答時間を評価することにより、分散リアルタイム環境構築の実現性を探ることを目的とする。

3. VIA による分散リアルタイム環境

本研究においてはターゲットとして汎用 PC を使用し、分散リアルタイム処理の検証環境を構築する。VIA NIC としては、VIA をハードウェア実装したギガネット社の cLAN を採用した。OS にはリンクス社リアルタイム OS、LynxOS を選択した。リアルタイム OS の中で LynxOS を選択した理由は VIA の特徴を生かすために仮想空間保護機能を持った OS が重要と判断したからである。ソフトウェア構成を図 1 に示す。

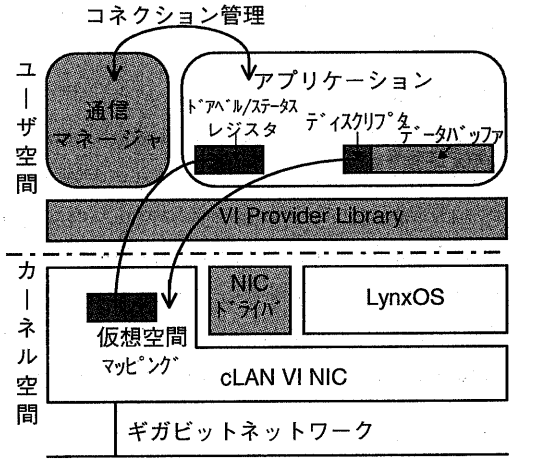

図 1: ソフトウェアの構成

今回、cLAN の VIA を使用するために LynxOS 上に次のソフトウェアを実装した。

(1) cLAN NIC ドライバ
(2) 通信マネージャ
(3) Virtual Interface Provider Library (VIPL)

VIA のためのハードウェア依存機能は全て NIC ドライバ内に実装し、OS 自体は変更していない。

NICドライバはNICの初期化、NICのドアベルレジスタやステータスレジスタのアプリケーションへのマッピング、転送データバッファのメモリ領域のNICへの登録などの処理を行う。通信マネージャは各ホストの通信マネージャ間で通信し、アプリケーションの通信コネクションの設立、解放の管理を行う。一旦通信コネクションが確立した後は、アプリケーションは通信マネージャやNICドライバによらず、自仮想空間内にマッピングされたレジスタを使い直接NICにデータ転送要求を発行する。ただし、今回の実装ではアプリケーションが処理完了をブロッキングにより待つ場合には、NICドライバへのシステムコールを使用し、割込みを受信している。VIPLはVIAの標準インタフェースを提供するライブラリである。VIPL内の並行処理はLynxOSの提供するマルチスレッド機能により実現した。

4. 評価

4.1. 測定環境

3章の開発により、評価環境を構築し、リアルタイム性能評価を行った。測定環境を表1に示す。

表1： 測定環境

ハードウェア	CPU: Pentium III 600MHz メモリ: SDRAM (100MHzバス) バス: PCIバス32ビット、33MHz
NIC	ギガネット社cLAN1000 物理転送性能1.25Gbit/秒、全二重 最大メッセージ長65Kバイト
スイッチ	ギガネット社cLAN5000
OS	LynxOS 3.0.1

また、測定のリファレンス環境として、ギガネット社が提供しているLinux用のVIAソフトウェアを使用し、同一ハードウェア上にLinuxカーネル2.2.9をインストールし比較性能測定を行った。

4.2. 基本性能

リアルタイムOS上のVIA実装の有効性を確認するために、帯域性能、応答遅延、CPU負荷などの基本性能の測定をし、Linuxの場合と比較を行った。VIAでは処理完了通知の受信方法としてポーリングとブロッキングのインタフェースを提供している。ポーリングを行った場合、ビジーウェイトにCPUが100%使用され、この間マルチタスク処理はできない。ブロッキングを行った場合は割込みにより処理完了待ちを行う。この2種類の方法を比較する。メッセージ長を変えた場合の帯域性能の測定結果を図2に示す。

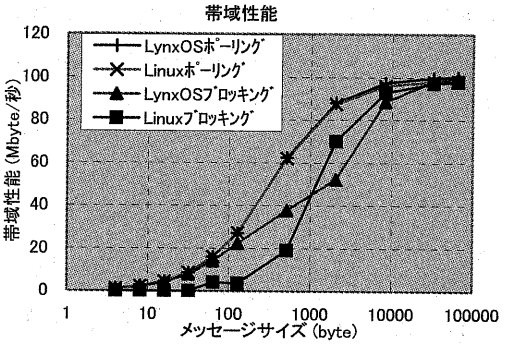

図2: メッセージサイズによる帯域性能

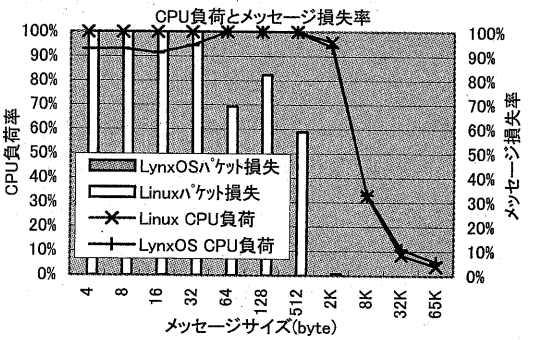

図3: メッセージサイズによるCPU負荷

今回の測定環境においてはピーク帯域性能として100MByte/秒 (800Mbit/秒)が達成されることを確認した。これはcLAN物理転送性能の64%の値である。ポーリングを行った場合、OSに対する依存性はほとんどなくなるため、LynxOSとLinuxでほぼ同じ性能が得られる。ブロッキングを使用した場合は、割込み処理のOSの特性が現われる。リアルタイムOSであるLynxOSは割込みが頻発する小さいメッセージサイズにおいて良い性能を示す。ブロッキング時のCPU負荷率

を図3に示す。CPU負荷率の測定方法は、通信処理と同時に低い優先度でバックグラウンド処理を実行し、通信を行っていない場合に対する実行性能の割合から算出している。LynxOSもLinuxも送信処理のCPU負荷はほぼ同等である。これは送信処理のOS依存性がほとんどないためである。しかし、受信側では、ブロッキングの場合小さいメッセージサイズでは割込み間隔が短くなるために割込み処理が間にあわず、Linuxでは受信ウィンドウが不足しほとんどのメッセージが失われる。それに対し、LynxOSは同じ条件でも追従できている。また4バイトのメッセージの応答時間(one way)を見ると表2のようになる。

表2: 4バイトメッセージの応答時間(one way)

	ポーリング	ブロッキング
LynxOS	7.37 μ秒	14.36 μ秒
Linux	7.34 μ秒	15.10 μ秒

これまでの結果と同様にポーリング時にはOSによる性能差はないが、ブロッキングの場合はLynxOSの応答時間のほうが短いことを示している。以上のようにブロッキングを使用した場合、リアルタイムOSの有効性が確認できた。さらにVIAの性能について、以下のように考察する。

(1) ポーリングとブロッキングの比較

ブロッキングを使用した場合、ポーリングに比べて、システムコール、コンテキストスイッチ、割込み応答のオーバヘッドが増加する。LynxOSの場合、約7μ秒の応答時間の増加となる。

(2) 小さいメッセージサイズの通信

図2の帯域性能からもわかるように、VIAではメッセージサイズが小さいときには、その効果が発揮できない。これはディスクリプタ処理がオーバヘッドとして顕著に現われることと、10 μ秒オーダ間隔で生じる完了処理でCPU処理がボトルネックになるためである。図3が示すように1KByte以下のメッセージの連続転送ではCPU負荷が100%となる。VIAではメッセージサイズが大きいほどCPU負荷が小さくなる。

4.3. ボトルネック要因

複数の同時通信を行うことにより現状環境のボトルネック要因を探った。その結果を表3に示す。

表3：複数通信時の帯域変化（単位：MByte/秒）

	コネクションA	コネクションB	帯域計
(1) 1対1通信 A、B送信	49.51	49.51	99.02
(2) 1対1通信 A:送信、B:受信	48.47	48.73	97.20
(3) 1対2通信 A:送信、B:受信	31.35	68.54	99.89
(4) 1対1通信 A: 32KB、B: 128B	4.21	21.82	26.03

((1)～(3)のメッセージサイズはA、B共32KB)

(1)は2本のコネクションA、Bが同方向でデータ送信した場合、(2)は2本のコネクションが送受逆方向の場合、(3)は2本のコネクションを別のホストに対して張った場合、(4)は2本のコネクションでメッセージサイズを変えた場合で、それぞれ測定を行った。(1)～(3)では、どの場合も2本のコネクションの帯域性能の合計値は約100MByte/秒であり、1本の通信コネクションの場合とほぼ同じとなる。今回の測定環境ではネットワーク物理層は全二重のスイッチであるので、(2)、(3)では帯域が拡大するはずであるが同方向通信と同じ性能しか得られない。4.2章の測定結果より32KByteメッセージサイズのCPU負荷は11%と低いので、CPU処理がボトルネックとなっているわけではない。したがって、PCIバスからメモリへのデータ転送がボトルネックとなり、帯域性能が向上しないものと予想する。さらに、(4)の小さいメッセージと組合せた状況では、メッセージサイズの大きい通信がメッセージサイズの小さい通信に阻害されて、帯域が大きく減少することがわかる。したがって、小さいメッセージの通信も性能ボトルネックの要因となり得る。

4.4. 定応答性評価

これまでの測定結果をもとに、高優先度周期通信の応答時間保証性を検証する。ここではリアルタイム性を必要とするデータ交換処理を行っている間に、低優先度の負荷処理を実行した場合のリ

アルタイムデータ交換の最悪応答時間の変動を測定する。4.3 節の結果より、応答時間を悪化させる要因として、以下の処理を負荷処理として測定を行った。

(1) **メモリバス負荷**: キャッシュにヒットしないメモリコピーを連続実行
(2) **PCI バス負荷**: 100Mbps イーサネットのネットワークバースト転送を連続実行
(3) **小メッセージの転送**: cLAN 上で 1KB メッセージ長のデータを連続送信

リアルタイムデータ交換はシステム内で最も高い優先度を設定し、定周期で固定サイズのデータを送信する。データ送信からそれに対する返信(4 バイト)を受信するまでの時間を応答時間とした。これを 10 万回実行し、応答時間保証性を見る。本環境における LynxOS 自体のコンテキストスイッチ時間は 2 μ秒、割込み遅延は 6 μ秒、システムコールオーバヘッドは 1 μ秒以下である。10KByte のリアルタイムデータ交換を行った場合の測定結果を表 4 に示す。ソフトリアルタイム応答性を判断するために、99%の測定結果が入る応答時間を表中に示した。同時に結果のヒストグラムを図 4 に示す。

表 4: 負荷による応答時間変化 (単位: μ秒)

	最小値	99%値	最悪値
(0)無負荷	140	150	170
(1)ﾒﾓﾘ負荷	170	220	240
(2)PCIﾊﾞｽ負荷	140	240	320
(3a)32KB 通信	220	280	360
(3b)1KB 通信	160	680	750

(リアルタイム通信メッセージサイズ 10KB)

図 4 中では無負荷時の最小値 140 μ秒が縦棒実線で示されており、矢印で示す範囲が応答時間の分布する領域である。メモリ負荷、PCI バス負荷により、応答時間が悪化することがわかる。しかし、この場合では無負荷時の最良の応答時間 140 μ秒に対し、最悪値はその 3 倍以内に収まっている。これにより、応答時間時間保証の設計が可能と判断する。それに対し、1KB サイズのメッセージ交換をバックグラウンド処理として行っている場合は、処理の優先度設定にも係わらずリアルタイム通信が阻害され、応答時間が大きく悪化する。最悪値が無負荷時の 6 倍以上の値になることに加えて、ほとんどの応答時間が無負荷時の 4 倍以上となっている。小さいサイズのメッセージ通信の競合は応答時間の最も大きな遅延要因となることがわかった。このような状況では応答時間保証のためには対処が必要である。

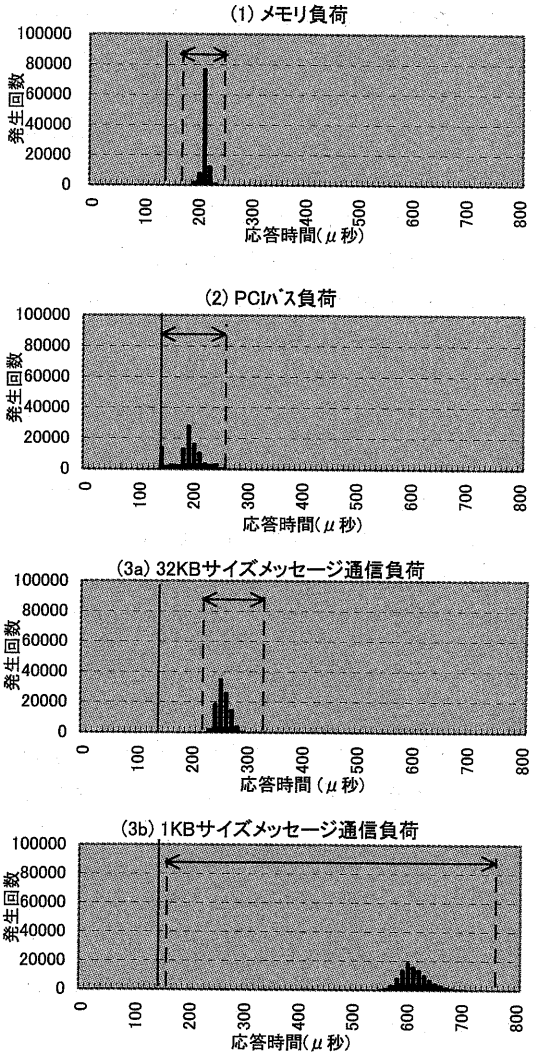

図 4: 負荷による応答時間変化

5. 考察・課題抽出

VIA を使った通信によりプロセッサ間通信のソ

フトウェアボトルネックを解消し、高速ネットワークのハードウェア性能を引出すことができることは確認できた。しかし、分散リアルタイム環境における応答時間保証という点に対しては課題が残されている。それについて考察を行う。

(1) VIAによる分散環境の応答時間保証

VIA をハードウェア実装したギガビットネットワークでは、アプリケーションが直接ハードウェア性能を利用することができる。同時に、分散環境における精度の高い応答時間保証も可能になる。従来のOSが関与するデータ交換では、カーネル内バッファ管理、ドライバ処理、共有管理データが競合要因となり、応答時間の最悪値を大きくしていたが、VIAによる通信ではそれらの要因を排除できる。今回の評価環境では、1KByte、10KByte、100KByte各サイズのデータ交換において、有負荷時の測定結果のばらつきは最小応答時間の2倍以内に収まっている。

それに対し、VI NICで小さいメッセージとの競合が発生した場合、応答時間が大きく悪化する。この原因は以下の点にある。

a) 割込み頻発による優先度制御の阻害
割込みは最高優先で処理することや、割込みによりコンテキストスイッチが頻発し、オーバヘッドが増大する。

b) ディスクリプタ処理による帯域の減少
VIA では各メッセージに付けられるディスクリプタに対するソフトウェア、ハードウェアの処理が性能を決定する。小さいメッセージサイズでは、相対的にディスクリプタに対する処理が増大し、帯域が減少する。

上記の問題を解決し、VIA上の通信の応答時間保証を向上させるために、優先度による割込み頻度制御や、帯域制御機構の導入について方式検討が必要と考える。

(2) ハードウェアボトルネック

今回の測定により、疎結合計算機クラスタを実現するための高速ネットワークはホスト内のIOバス、メモリバスを飽和させる可能性があることが明確になった。今回の測定結果からも、非競合時の帯域性能のボトルネックは、CPU やネットワークではなく、PCIバスからメモリへのデータ転送にあると予想できる。今後の高速なネットワークの性能を有効にするには、ホスト単体のアーキテクチャも変えることが必要である。従来のバス型のIO接続ではなく、スイッチのようなIO接続が必要となる。

6. まとめ

リアルタイムOS上でのVIAの評価の結果、プロセッサ間データ交換においてソフトウェアオーバヘッドを削減するVIAの有効性を確認できた。VIA を使い分散リアルタイム環境を構築することにより、スケーラビリティのある並列データ処理エンジンを実現することが可能である。しかし、応答時間保証を実現には、割込みの頻発や小メッセージによる競合を回避することが重要であることが判明した。今後それらの方式について検討を進める予定である。

参考文献

[1] T. von Eicken, A. Basu, et. al, "U-Net: A User-level Network Interface of Parallel and Distributed Computing", Proc. of the 15th ACM Symposium of Operating systems Principles, vol. 29, (no. 5), Dec. 1995

[2] S. Pakin, V. Karamcheti, A. A. Chien, "Fast messages: efficient portable communication for workstation clusters and MPPs", IEEE Concurrency, vol. 5, (no.2), April-June 1997

[3] H. Tezuka, A. Hori, Y. Ishikawa et. al, "PM: An Operating System Coordinated High Performance Communication Library", In Peter Sloot Bob Hertzberger, editor, High - Performance Computing and Networking, vol. 1225 of Lecture Notes in Computer Science, April 1997

[4] "Intel Virtual Interface (VI) Architecture Developer's Guide Revision 1.0", Intel, http://develpper.intel.com/design/servers/vi/technology/technology.htm, Sep., 1998

[5] E. Speight, H. Abdel-Shafi, J. K. Bennett, "Realizing the Performance Potential of the Virtual Interface Architecture", Proc. of the 13th ACM International Conference of Supercomputing, 1999