

Comet における VIA 的インタフェース

小林 伸治、陣崎 明

新情報処理開発機構 並列分散システム富士通研究室
〒211-8588 川崎市中原区上小田中 4-1-1
E-mail: {koba,zinzin}@flab.fujitsu.co.jp

高速プロトコル処理エンジンを搭載するネットワークアダプタ、Comet をソフトウェアから利用する際のインタフェースとして VIA(Virtual Interface Architecture)を利用することを検討した。VIA は各ユーザプロセスが仮想的なネットワークインタフェースを専有できる方式であり、カーネルの介在無しにネットワークインタフェースが直接ユーザプロセス中のデータを DMA するため送受信のオーバーヘッドが少ない。Comet では TCP/IP を主要なプロトコルとしているため、VIA で TCP/IP を利用する方式について検討した。その結果、VIA をベースにして Comet 用の効率的なインタフェースを構築できる見通しが見ついた。

Comet、プロトコルエンジン、VIA、TCP/IP

Software interface for Comet based on VIA

Shinji Kobayashi, Akira Jinzaki

Parallel and Distributed Systems Fujitsu Laboratory, RWCP
1-1, Kamikodanaka 4-Chome, Nakahara-ku, Kawasaki, 211-8588 Japan
E-mail: {koba,zinzin}@flab.fujitsu.co.jp

We investigated to use the VIA(Virtual Interface Architecture) as a software interface of the network adapter Comet which is equipped with a fast protocol processing engine. In the VI Architecture, the overhead of sending and receiving is small because each user processes can have their own virtual network interfaces, and the network interface can directly access data in user processes using DMA without interrupting the kernel. As the TCP/IP is the most important protocol in Comet, we studied on using the TCP/IP in the VIA environment. As the result, we got the prospect that we can construct an efficient interface for Comet based on the VIA.

Comet, protocol engine, VIA, TCP/IP

1. はじめに

ATM 等の高速な広域網が利用できるようになり、広域分散した並列計算環境が現実的になってきた。並列計算においてはノード間の通信遅延や帯域が重視されるため、専用の通信プロトコルを設計して用いるのが常道である。しかし、昨今のインターネットの隆盛を見るまでもなく、世界的な分散環境で実用に堪えるプロトコルは現時点ではTCP/IP 以外に考えられない。

TCP/IP は重いプロトコルであるが、我々が開発している Comet ではプロトコル処理を Comet アダプタ上で行うため高速な処理が可能である。Comet を用いれば広域分散網としての TCP/IP ネットワーク上に並列分散計算環境を構築することが可能になる。

高速なネットワークインタフェースを設計する場合、それをソフトウェアからどのように効率的に利用できるかが大きな課題となる。特に、Comet のようにアダプタ上でプロトコル処理を行うアーキテクチャでどのようなソフトウェアインタフェースが有効であるかは確立されていない。

高速ネットワークインタフェースでよく利用される手法として、ネットワークインタフェースをユーザ空間に直接マップする方法がある。この方法はカーネルのオーバーヘッド無しにアダプタを利用できることが特長である。カーネルを介さずに複数のユーザプロセスから同一のインタフェースを利用する方法も各種提唱されている。なかでも、VIA は仮想化したネットワークインタフェースを各プロセスが専有できるという明確なモデルに基づいており、汎用性も高い。多くのベンダが支持していることもあり、今後の標準の1つとなる可能性がある。

そこで、本研究では Comet のソフトウェアインタフェースとして VIA を利用することができるかどうかを検討した。第2節では我々が開発している Comet について説明し、第3節では VIA のモデルを説明する。Comet では TCP/IP が非常に重要なプロトコルであるため、第4節では VIA と TCP/IP の関係に注目して検討を行い、第5節でその結果をまとめる。

2. Comet 概略

ネットワークの高速化に伴い、プロトコル処理のオーバーヘッドがネットワークの実効速度を制限するようになってきている。Comet はプロトコル処理の一部をネットワークアダプタカード上のハードウェアで行うことにより通信の高速化を目指したシステムである[1]。試作した Comet アダプタはプロトコル処理を行うエンジンを持ち、PMC(PCI Mezzanine Card)規格のネットワークカードを2枚搭載できる PCI(Peripheral Component Interconnect)規格のカードである[2](図1)。現在はプロトコル処理を Comet アダプタ上の汎用プロセッサでソフトウェアエミュレートしているが、今後 FPGA や ASIC にしていく予定である。

Comet アダプタはプロトコル処理をハードウェアで行う高速なネットワークカードとして用いることができる。TCP/IP のような重いプロトコルも Comet アダプタ上で高速に処理できる。汎用プロセッサによるエミュレーションをしている現在の試作でも、IPカプセル化して送信する処理を約 40 μ 秒で行える[3]。また、経路制御も Comet アダプタ上で実行することで高速なルータを実現できる。Comet による超高速インターネットルータの構築も我々の目標の1つである[4]。

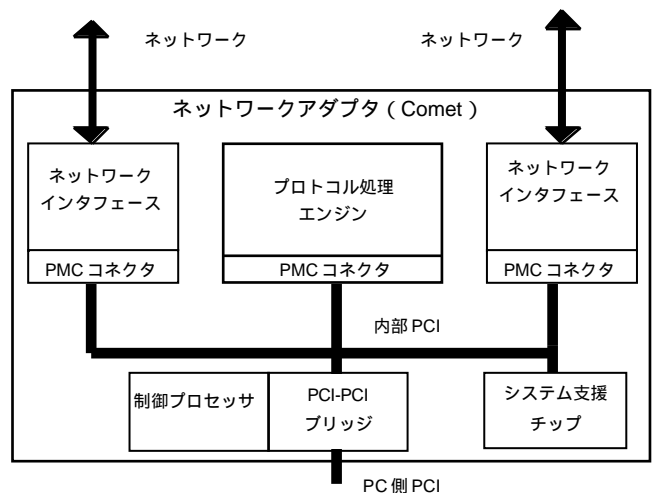


図1: Comet アダプタ構成

3. VIA 概略

VIA(Virtual Interface Architecture)は Compaq、Intel、Microsoft の3社が協同で開発しているネットワークイ

インタフェースのアーキテクチャであり、並列計算等のユーザプロセスから高速にネットワークインタフェースを使用することを目標にしている。仕様書の 1.0 版 [5] が公開されており、誰でも入手可能である。

VIA ではネットワークインタフェースカード(NIC)が複数のコンテキストを持てるようになっており、カーネル内のドライバと協調してこれらを仮想的なインタフェース(VI)として複数のユーザプロセスに提供する。各ユーザプロセスは VI を専有することができる(図 2)。

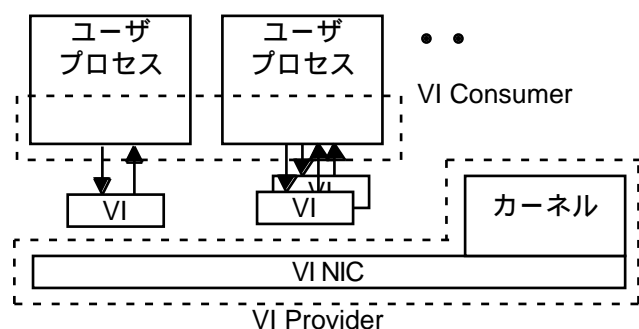


図 2: VIA のモデル

VIA の NIC とカーネル内ドライバとを併せて VI Provider と呼び、VI Provider が提供する VI を利用するユーザプロセス内のライブラリまたはアプリケーションを VI Consumer と呼ぶ。VI Provider と VI Consumer は各 VI が持つキューを介してコマンドの発行、結果の受取等を行う。データは NIC がユーザプロセスに直接 DMA することで送受信する。コマンドの発行からデータの送受信までカーネルの介在無しに行えるため低遅延の通信を実現できることが特長である(図 3)。

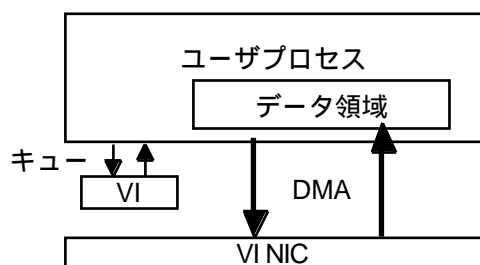


図 3: VIA におけるデータの流れ

NIC がユーザプロセス空間中のデータを直接 DMA するため、VIA では送受信の対象となるメモリ領域はあらかじめ登録しておく必要がある。

VIA の通信はコネクション指向である。すなわち、各 VI は通信先 VI に接続してから通信を行う。接続は 1 対 1 であり、ブロードキャスト通信等は直接行えない。

VIA の仕様では VI Provider と VI Consumer との間の API は規定していないが、サンプル API が付録として記述されている。Intel はこれを拡張して VIPL (VI Provider Library) という API を Intel Implementation Guide [6] の一部として公開しており、これが事実上の標準 API と言える。

ネットワークインタフェースを仮想化してユーザプロセスに直接提供するというアイデアは VIA に始まったものではない。U-Net [7] や Virtual Network Transport Protocols [8] を挙げるができる。VIA は仕様を定義して公開し、ベンダに広く採用を呼びかけている点でこれらの研究プロジェクトとは異なる。多くのベンダが賛同していることから、VIA は今後のネットワークアーキテクチャの一標準となる可能性を持っている。

Comet を高速ネットワークアダプタとして利用する場合もソフトウェアからどう利用するかということは重要な問題である。ユーザ空間に直接仮想的なインタフェースを提供する方式はオーバーヘッドが少ないため有望な方式である。そこで、Comet のソフトウェアインタフェースとして VIA を利用できるかどうか検討した。

4. VIA における TCP/IP の利用

VIA は元々 SAN (System Area Network) での高速ネットワークをターゲットとして開発されたため、TCP/IP のことはあまり考慮されていない。しかし、最初に述べた通り、Comet のソフトウェアインタフェースでは TCP/IP が利用できることが非常に重要である。そこで VIA における TCP/IP の利用方法を検討した。

VIA で TCP/IP を利用する方法は次の 2 つに分けることができる。

- VIA 同士の通信に TCP/IP を利用 (方式 1)
- TCP/IP が使える口を VIA に用意 (方式 2)

前者は VI NIC 間のネットワークとして Myrinet 等のネットワークメディアと同じレベルに TCP/IP を位置付ける方式で、他の TCP/IP ノードとの通信は考慮しな

い(図 4)。後者は他の TCP/IP ノードと通信するために Comet を高速な TCP/IP プロトコル処理機構として利用するための方式である(図 5)。

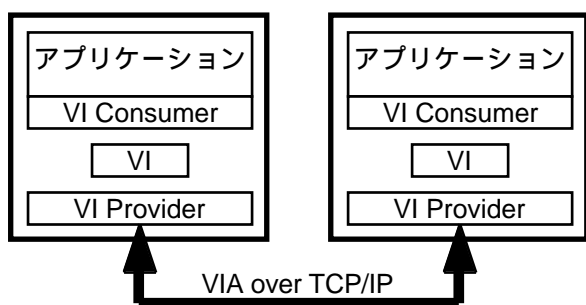


図 4: VIA 同士の通信に TCP/IP を利用する方式

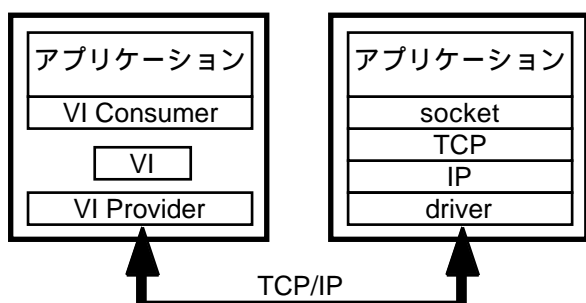


図 5: TCP/IP が使える口を VIA に用意する方式

4.1. 方式 1

VI間の通信に TCP/IP を利用する方式 1 では TCP/IP のプロトコル処理をどこで行うかという問題がある。TCP/IP のプロトコル処理はカーネルで行うのが一般的であるが、VI 間の通信にカーネル内のプロトコルスタックを利用しては VIA の特長が生かせない。先に述べた通り、Comet では TCP/IP のプロトコル処理を Comet アダプタ上で行うため、VIA の特長を損なわずに TCP/IP を利用できる(図 6)。

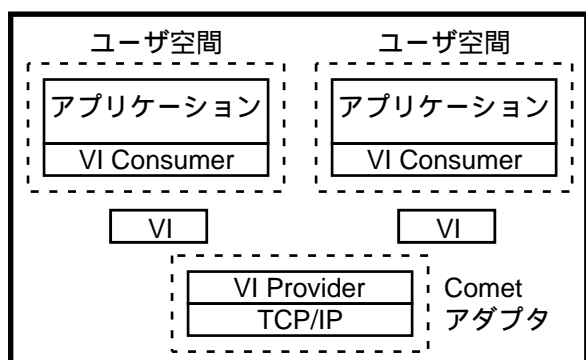


図 6: Comet での方式 1 の実現

方式 1 ではアプリケーションから受け取ったデータに VI 間の通信に必要な情報を付加して IP パケット化する。方式 1 を実現するためには、VIA で要求される通信の特性を TCP/IP が満たすことができるかどうかを検証する必要がある。

通信の特性として重要なものは、通信の信頼性である。VIA では通信の信頼性を 3 レベル定義している。各レベルで必要とされる主要な機能を表 1 に、TCP/IP の各プロトコルが提供する信頼性を表 2 に示す。

表 1: VIA で要求される信頼性

信頼性のレベル	Unreliable	Reliable Delivery	Reliable Reception
データ誤りの検出	要	要	要
重複配送しない保証	要	要	要
データ配送の保証	不要	要	要
データの順序性保証	不要	要	要
送信要求の完了	送信後	送信後	受信後

表 2: TCP/IP が提供する信頼性

プロトコルの種類	IP	UDP	TCP
データ誤りの検出	無	有	有
重複配送しない保証	無	無	有
データ配送の保証	無	無	有
データの順序性保証	無	無	有

Unreliable ではデータ誤りの検出と重複配送しないことを保証する必要がある。IP 層はどちらも保証しないので、上位層で保証する必要がある。トランスポート層として TCP を採用すれば両方とも保証できるが、少々オーバースペックである。UDP ではデータエラーの検出を行えるが、重複配送の可能性がある。以上を考慮すると、Unreliable のレベルでは UDP を使い、シーケンス番号等を付加することで重複配送を避けるのが適当であると考えられる。

Reliable Delivery と Reliable Reception は送信要求が完了するタイミング以外は同じである。データ配送の保

証(あるデータは必ず 1 度 配送される)、順序性の保証は TCP を採用することで確保できる。Reliable Reception では送信要求が完了した時点でリモートノードでの受信も完了している必要があるが、これは受信完了のアクノリッジを返すようにすれば実現できる。すなわち、TCP を利用すれば Reliable Delivery と Reliable Reception で必要とされる機能を簡単に実現できる。

通信相手を特定するためにはアドレスが必要である。VIA におけるネットワークアドレスはホストアドレスと識別子(discriminator)からなり、どちらもバイト単位で長さを指定できる。このため、IP アドレスをそのまま VIA でのアドレスとして使用できる。現在の IPv4 のみならず、128bit のアドレス長を持つ IPv6 にも対応可能である。識別子も TCP や UDP のポート番号に対応付けることができるが、Intel Implementation Guide では識別子は 16bytes 指定できるようにすることを推奨している。これを満たすためにはポート番号とは独立に管理する必要があるが、それは特に問題とはならない。

もう 1 つの重要な通信特性は遅延である。特に、VIA が主なターゲットとしている並列計算アプリケーションでは重要な問題である。どの程度の遅延まで許容されるかはアプリケーションの種類等によって異なる。一般的なインターネットでは遅延が数百 m 秒になることもあるが、これでは実用になるアプリケーションは限られてしまう。

Comet では汎用プロセッサによるエミュレーションでも IP カプセル化の処理を 40 μ 秒程度で行える。今後 ASIC 化した場合には 5 μ 秒程度に短縮できる見込みである。すなわち、Comet を利用した超高速インターネットルータは 1 ホップあたり 5 μ 秒程度で処理できる。このくらいになるとほとんどの場合距離による伝送遅延の方が支配的になる。利用するメディアにもよるが、数十～数百 Km 以内ならば数 m 秒の遅延となり、多くのアプリケーションを実用的に実行できる。

4.2. 方式 2

先に述べたように、TCP/IP のプロトコルスタックは

通常カーネル内に実装される。1 つのネットワークアダプタを複数のプロセスから利用するために調停する必要があるからである。しかし、VIA のように複数の仮想的なインタフェースが提供されて各プロセスがそれらを専有できるようになれば TCP/IP プロトコルスタックをカーネル内に実装する必要はなくなる。すなわち、各プロセス毎に TCP/IP プロトコルスタックを持つ図 7 のような構成が可能になる。

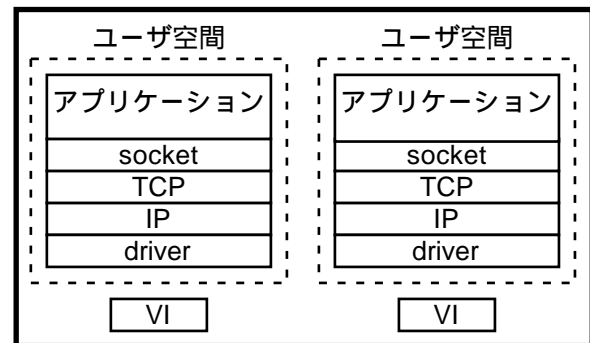


図 7: プロセス毎の TCP/IP プロトコルスタック

このような構成で TCP/IP プロトコルスタックをユーザレベルのライブラリとして実装し、従来のドライバに相当する部分に VI の機構を持たせることもできる。しかし、この方式では TCP/IP プロトコル処理のオーバーヘッドを削減することはできない。

Comet では TCP/IP のプロトコル処理も Comet アダプタ上で行う予定である。この場合、ドライバ部分ではなく socket 層に VI の機構を持たせることになる(図 8)。

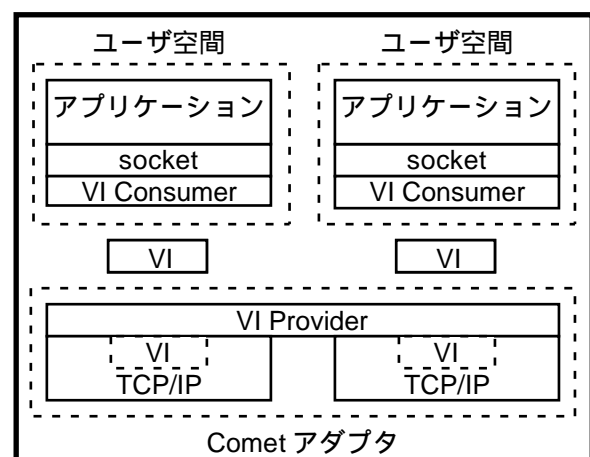


図 8: Comet での方式 2 の実現

このモデルでは socket と VI を 1 対 1 に対応付けること

ができる。元来 socket は通信インタフェースの抽象化のために導入されたものであり、仮想的なネットワークアダプタである VI と対応付けることは自然である。

VIA のモデルでは通信は VI 間で行われるため、ユーザプロセス側の VI の通信相手となる VI が必要である。この場合、通信相手の VI は Comet アダプタ内に存在する。別の言い方をすれば、VI としての通信は Comet アダプタ内で終端し、VIA による通信と TCP/IP による通信を Comet が相互接続していることになる。方式 1 とは異なり、ネットワーク上には VIA 固有の情報は流れない。アプリケーションから受けたデータを単純に TCP/IP カプセル化したデータが流れるだけである。

上位層に socket の API を提供するような VI Consumer を作成する上で一番問題になるのはデータバッファの管理である。VIA では送受信に使うデータ領域は登録しておく必要があるのに対して、socket では任意のデータ領域を送受信することができる。このため、socket に対する write や read が発生した時点でその領域を登録するか、あらかじめ登録してある領域にコピーする必要がある。これらのコストを完全に削減するためには socket の API を拡張してデータ領域の登録、解除を行えるようにする必要がある。高速性が最重要課題であるアプリケーションはこれらの拡張 API を使用してプログラムのチューニングを行うべきであろう。しかし、通常は通信に使用される領域は限られているため、一度登録した領域はできる限り登録したままにしておく等の工夫をすることで、socket の API を変更しなくても十分効率的な VI Consumer を作成できる。

4.3. 方式 1 と方式 2 の混在

VIA の枠組みを拡張すれば方式 1 と方式 2 を混在させて実現することができる。VIA が提供する VI に属性を持たせ、VI 同士で通信を行うための VI や、TCP/IP ノードと通信を行うための VI などをユーザプロセス (VI Consumer) が指定できるようにするのである。この場合の実現方法としては、1 つの VI Provider が複数種類の VI を提供する方式と、VI の種類だけ VI Provider を用意しておく方式が考えられる。

両方式を実装することにより、Comet アダプタのイン

タフェースを VIA をベースにした方式で統一しながら、並列計算等のアプリケーションにも TCP/IP のアプリケーションにも効率的なソフトウェアインタフェースを実現できる。

5. 結論

プロトコル処理をハードウェアで行う Comet アダプタ用のソフトウェアインタフェースとして VIA が使えるかどうかを検討した。その結果、VIA を一部拡張することで並列計算等のアプリケーションを広域分散環境で実行するにも、TCP/IP のアプリケーションにも有効なインタフェースを構築できそうであるという結論に達した。今後は詳細な設計を行うとともに Comet アダプタ上で実装を行い、TCP/IP ノードとの接続性の確認や性能評価を行っていく予定である。

参考文献

- [1] 陣崎、中村、村井：並列ネットワークサーバ Comet のアーキテクチャとその応用、SWoPP'98 CPSY, 1998 年 8 月
- [2] 河合、下國、都筑、竹原、陣崎：Comet のハードウェア試作と性能評価、SWoPP'98 CPSY, 1998 年 8 月
- [3] 古賀、陣崎：Comet による IEEE1394 を利用した計算機ネットワークの構築、SWoPP'98 CPSY, 1998 年 8 月
- [4] 水野、陣崎：Comet による広域分散共有メモリの提案とその評価、SWoPP'98 CPSY, 1998 年 8 月
- [5] “Virtual Interface Architecture Specification Version 1.0”, <http://www.viarch.org/>
- [6] “Intel Virtual Interface (VI) Architecture Implementation Guide Draft Revision 0.95”, http://developer.intel.com/design/servers/vi/developer/ia_imp_guide.htm
- [7] M. Welsh, A. Basu, T. von Eicken “Incorporating Memory Management into User-Level Network Interfaces”, Hot Interconnects V, Aug. 1997
- [8] B. N. Chun, A. M. Mainwaring, D. E. Culler “Virtual Network Transport Protocols for Myrinet”, Hot Interconnects V, Aug. 1997