システムソフトウェアと
オペレーティング・システム 82−13
（1999. 8. 2）

UNIX OS におけるノード間高速通信

黒澤崇宏† 小川尚志† 刀野暢洋‡
Andreas Savva‡ 福井恵右‡ 岸本光弘†
† 株式会社富士通研究所
‡ 富士通株式会社

UNIX クラスタにおけるノード間通信を高速化するため，SCnet (Smart Cluster Network) を開発した．SCnet は System Area Network (SAN) である Synfinity-0 上に標準 SAN 通信方式として提案されている VI アーキテクチャを一部ソフトウェアエミュレーションすることで実現するものである．本稿では，特に SCnet の通信メモリ管理方式，データ通信方式，ネットワーク多重化方式の設計について述べ，ソフトウェアエミュレーションにより VI アーキテクチャを実現する手法を紹介する．また，実装した SCnet について，レイテンシとバンド幅について性能を測定し，0 バイト転送時のレイテンシは 42 μsec，128バイト転送時のバンド幅は 107MB/sec との結果を得た．

Fast Inter-node Communication on UNIX Systems

Takahiro Kurosawa† Naoshi Ogawa† Nobuhiro Tachino‡
Andreas Savva‡ Keisuke Fukui‡ Mitsuhiro Kishimoto†
† Fujitsu Laboratories Ltd.
‡ Fujitsu Limited.

Smart Cluster Network (SCnet) offers fast inter-node communication on UNIX systems connected using the Synfinity-0 System Area Network (SAN). SCnet implements the VI Architecture, a proposed standard of SAN connected systems that enjoys wide industry backing, by emulating some features of the VI Architecture in software. This paper describes the design of SCnet, especially the memory management mechanism for data buffers, communication protocols, and effective usage of multiple NICs, possibly connected to multiple networks. We present experimental results using a two node system. SCnet achieves latency of 42μsec for a 0 byte transfer, and bandwidth of 107MB/sec for a 128KB transfer.

1 はじめに

計算機（ノード）を複数接続し，クラスタ構成としたものは，可用性や性能向上を実現することが可能である．クラスタ構成による並列化で性能向上を図るためには，ノード間結合ネットワークのハードウェアの高速化や，その上に実装される通信ソフトウェアの高速化が要求さ

れる．

LAN 上で用いられている TCP/IP 通信では，プロトコル処理やシステムコールなどのソフトウェアオーバヘッドが大きく，高性能な通信ハードウェアを用いてもレイテンシの面での性能向上に限界がある．このため，近年ではクラスタを構成するノード間を従来の LAN を用いて接続するのではなく，より広帯域·低レイテンシの専用ネットワーク (総称して System Area Network (SAN) と呼ばれる) を用いて接続することが提案されている．SAN は低エラーレートで高速な通信が可能であるばかりでなく，ユーザプロセスから直接ハードウェアにアクセス可能な通信方式 (ユーザレベル通信) が利用可能である．これにより，システムコールのオーバヘッドを介在させることなく通信が可能であり，より高速な通信を実現することが可能である．

VI アーキテクチャ[1, 2, 3] は，Compaq, Intel, Microsoft により，SAN と SAN 上のユーザレベル通信の次期業界標準として提唱されている．我々は，VI アーキテクチャの UNIX への実装として，SCnet(Smart Cluster Network) を SAN である Synfinity-0(AP-Net とも呼ばれる) [4] の上に設計·実装した．本稿では，ソフトウェアによる VI アーキテクチャの実現法の提案とその評価を行う．

2 System Area Network

ここでは，VI アーキテクチャと Synfinity-0 のそれぞれの特徴について説明し，VI アーキテクチャを Synfinity-0 で実現する際にソフトウェアによるエミュレーションが必要となる機能についてまとめる．

2.1 VI アーキテクチャ

ここでは，VI アーキテクチャの概要について示すこととする．

VI アーキテクチャでは，コネクション指向の通信機構を提供する．アプリケーションはまず VI (Virtual Interface) と呼ばれる仮想的な通信エンドポイントのコンテキストを作成し，通信先の VI に対して接続してから通信を行う．

VI はハードウェア上に多数生成することができる．VI によりハードウェアが仮想化されるため，VI を使用して通信しているプロセスそれぞれが，あたかも NIC を占有して使用しているような状態で通信を実行することが可能である．従来の SAN のハードウェアでは一つまたは少数の通信エンドポイントしかサポートしていないものがほとんどであり，複数のアプリケーション間で NIC を共有して使用するには困難が伴った．そのため，多数の通信エンドポイントをハードウェアで実現するという構成は VI アーキテクチャの大きな長所と言える．

通信のエンドポイントとなる VI は，Send Queue と Receive Queue と呼ばれるキューを持つ．これらのキューに Descriptor と呼ばれるデータ構造をエンキューし，さらに NIC 上のハードウェア資源 (ドアベルと呼ばれ，VI ごとに存在する) を用いて NIC に Descriptor が追加されたことを通知する．これらすべての処理はユーザプロセスが直接行い，システムコールを伴ったカーネルコードによる処理は行わない．Descriptor には送受信バッファのアドレスや，通信種別などを記述する．VI アーキテクチャがサポートする通信種別は，send/receive 型通信と RDMA 型通信に分けられる．

send/receive 型通信では，送信側の VI の Send Queue 上の Descriptor に記述された送信バッファのアドレスから，指定されたサイズのデータを受信側に送信することになる．受信側の VI では，Receive Queue 上の Descriptor に記述された受信バッファの領域に，受信データを書き込むことになる．一方，RDMA 通信では，RDMA 通信を発行する側の VI 上の Send Queue にある Descriptor にローカル側のデータバッファの情報とリモート側のデータバッファの情報を記述することで，直接メモリアドレス指定により通信を行う．

VI アーキテクチャでは，Descriptor やデータ通信用のバッファなど，VI NIC が参照するメモリの領域はあらかじめ VI NIC に登録してから利用する構成となっている．

2.2 Synfinity-0

我々は，UNIXクラスタの性能向上を目的として SCnetの実装を行った．SCnetのターゲットとなるクラスタのノードには CPU として UltraSPARC-II を搭載し，OS として Solaris2.6 を搭載したものとした．SAN は Synfinity-0を用いることとした．本節では，Synfinity-0について説明し，VI アーキテクチャで規定されたハードウェアとの機能の比較を行う．

Synfinity-0は，片方向通信のバンド幅 240MB/sec の性能を持つ SAN である．Synfinity-0の NIC は，send/receive 型通信と RDMA 型通信の両方をハードウェアでサポートしている．Synfinity-0における send/receive 型通信は，send により送信されたデータが受信側の NIC が管理するリングバッファに書き込まれるという通信モデルを採用している．RDMA 型通信については，RDMA write 通信 (put) と RDMA read 通信 (get) の両者をサポートしている．

Synfinity-0は VI アーキテクチャの仕様制定より以前に開発された SAN であるため，VI アーキテクチャの仕様を完全には満足していない．例えば，Synfinity-0は VI アーキテクチャと異なり，多数の通信エンドポイントはサポートしていない．VI のような，不特定多数の通信エンドポイントをサポートするためのハードウェア的な機構が存在せず，一定数の通信エンドポイントが通信を行うのに適した構成となっている．

NIC のメモリ保護機構についても，Synfinity-0と VI NIC では異なったモデルを採用している．Synfinity-0のメモリ保護機構は，NIC あたり 2 あるいは 3 個の通信エンドポイントのみについてメモリ保護を実現している．単一の NIC では，4 個以上の通信エンドポイント間のメモリ保護を行うことはできない．一方，VI アーキテクチャでは，ノード上に多数生成可能な VI ごとに異なるメモリ保護を行うことができるとしている．

2.3 エミュレーションが必要とされる機能

以上で述べた通り，Synfinity-0と VI アーキテクチャの間には機能の相異点が複数存在する．SCnetにおいては，Synfinity-0のハードウェアでは実現されていない以下の機能をソフトウェアエミュレーションにより実現する必要がある．

- メモリ保護
- データ通信

前述の通り，Synfinity-0にはメモリ保護機構は存在しているものの，VI アーキテクチャで規定されている多数の通信エンドポイントを考慮したメモリ保護機構ではない．このため，SCnetではメモリ保護機構をソフトウェアにより実現する必要がある．

VI アーキテクチャでは，Descriptor などのデータ構造は VI NIC のハードウェアが解釈し，要求されたデータ通信などの処理を実行する．しかし，Synfinity-0は Descriptor などの VI アーキテクチャのデータ構造を解釈しないハードウェアである．Synfinity-0で VI アーキテクチャのデータ通信を実現する際には，まず VI アーキテクチャのデータ構造の解釈をソフトウェアで実現する必要がある．

また，Synfinity-0は固定された数の通信エンドポイントしかサポートしないため，SCnetではソフトウェアにより多数の通信エンドポイントを効率良くサポートすることが求められる．また，VI アーキテクチャのデータ通信をソフトウェアでエミュレートする場合には，データ通信に伴って機能するメモリ保護機構も実現する必要がある．

3 ソフトウェアエミュレーションによる実現方式

3.1 SCnetの全体構成

SCnetの全体構成を，図 1 に示す．

SCnetは，VIPL と呼ばれるライブラリと Kernel Agent と呼ばれるデバイスドライバ部分か

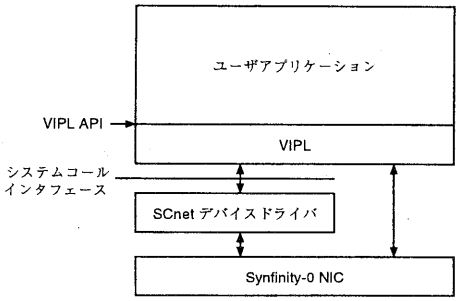

図 1: SCnet の全体構成図

ら構成される．ドライバは以下に挙げるモジュールにより構成され，それぞれのモジュールが相互に作用することで動作する．

Send Queue/Receive Queue 管理 Send Queue や Receive Queue にエンキューされた Descriptor を処理し，データ通信管理モジュールにデータ通信の指示を行う．

メモリ登録管理 ユーザプロセスからのメモリ登録/登録解除要求を受け，ページの物理メモリへの固定/固定解除を実行する．また，ユーザプロセス空間アドレス，カーネル空間アドレス，I/O バス空間アドレスの変換も行う．

コネクション管理 VI のコネクション開設・切断の管理を行う．

データ通信管理 データの送受信を受け，NIC に対して通信要求を発行する．また，NIC からの割り込みを受けて通信完了処理を実行する．

3.2 メモリ管理方式

VI アーキテクチャでは，通信に使用するメモリをあらかじめ VI NIC に登録することで，VI NIC による不正なメモリアクセスが検出可能となり，メモリ保護が実現できる．メモリ登録の際には，ユーザプロセス上の仮想アドレス空間と，VI NIC から参照される I/O バスのメモリアドレス空間の変換に利用する情報も登録しておく必要がある．この情報に基づいて VI NIC はユーザプロセスから指定されたメモリ領域を I/O バスのメモリアドレス空間に変換してアクセスすることになる．

VI NIC はメモリ登録後はそのメモリページが登録された位置に存在すると仮定して動作するため，メモリの登録を行う際には，デバイスドライバは対象となるページを物理メモリに固定し，ページングされないようにする必要がある．よって，SCnet のデバイスドライバはメモリ領域の管理を実現する必要がある．なお，物理メモリやメモリ保護に利用される資源は有限であるため，デバイスドライバや OS は登録可能なページ数の総量を制限する必要がある．

VI アーキテクチャでは，通信の際の不正なメモリアクセスがあった場合に NIC が誤りを検出してメモリの破壊を未然に防ぐこととなっている．しかし Synfinity-0 は VI NIC と異なり，2 あるいは 3 個の通信エンドポイントのみを考慮したメモリ保護モデルを採用している．よって，メモリアクセスの正当性チェックはハードウェアを用いずに全てドライバで実装することとした．

図 2 に示す通り，VI アーキテクチャで提案されているメモリ管理のためのデータ構造は，一つのメモリページあたりに一個必要としている．これは，I/O バスのメモリアドレス空間が物理メモリアドレス空間と同一であることを前提としているため，ユーザプロセス上の複数の連続ページにわたるメモリ領域は物理メモリのページでは連続したページと仮定できないからである．

SCnet のターゲットである SPARC アーキテクチャ上の Solaris では I/O バスのメモリアドレス空間は物理メモリと同一ではなく，一種の仮想アドレス空間として実現されている．このため，ユーザプロセス上の複数の連続ページにわたるメモリ領域は I/O バスのメモリアドレス空間でも連続したページとして参照可能となる．よって，SCnet ではこの特性を利用し，図 3 に示すように，複数の連続ページにわたるメモリ領域の場合にも一つのメモリ管理のデータ

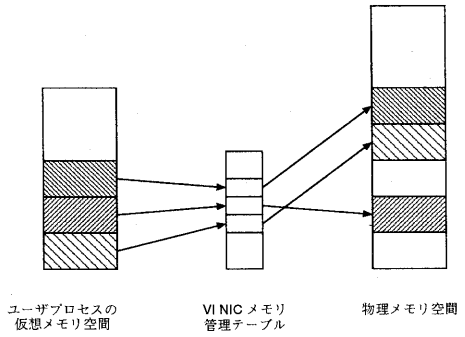

図 2: VI アーキテクチャのメモリ保護機構

構造を用意するという構成とした．これにより，メモリ管理のために必要となるデータ領域の量を削減することもできる．

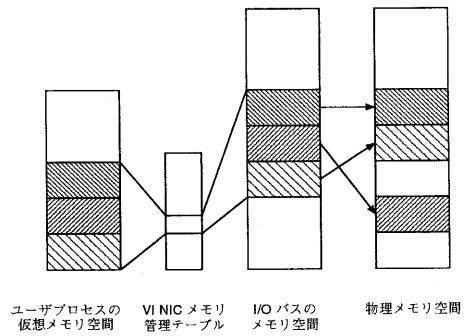

図 3: SCnet で実装したメモリ保護機構

3.3 通信方式

SCnet では，ユーザプロセスの通信要求にかかる時間を極力抑えるために，通信要求をまずドライバ内の待機キューにエンキューしておき，非同期に NIC に通信を発行するという構成とした．キュー構成を採用したことで，NIC が通信処理のための要求を受理できない状態にあったとしても，ユーザプロセスは通信要求を発行可能となるまで待機する必要がなく，要求を待機キューにエンキューした時点で処理を完了することを可能とした．NIC は通信処理を完了後に割り込みにより処理の完了を通知するため，ドライバは割り込みを契機としてキューの処理を再開することが可能である．このように SCnet では，ユーザプロセスからの通信要求にかかるコストを削減することを可能としている．

先に述べた通り，Synfinity-0 の通信モデルと VI アーキテクチャの通信モデルは異なり，Synfinity-0 の通信プリミティブと VI アーキテクチャの通信は一対一に対応していない．また，SCnet では通信用のメモリの保護を NIC のハードウェアではなく，ドライバにより実装する必要がある．以上のことから，我々は性能向上を図るために，Synfinity-0 の通信プリミティブを場合によっては複数組み合わせることとした．以下でその詳細について示す．

SCnet ではサイズの小さなデータを送信する場合には，図 4 に示すように，送信すべきデータをドライバ内の送信バッファにコピーし，そのデータに宛先 VI やデータサイズなどの情報をヘッダとして付与して send により送信することとした．受信のドライバは，受信バッファに到着したデータのヘッダを解析して送信データや宛先 VI をチェックし，対応する受信データバッファの領域にデータをコピーする．ここでは，この通信方式を copy-send-copy 通信と呼ぶことにする．copy-send-copy 通信は，メモリのコピーにかかる時間がデータ送信の際のレイテンシに対して無視できるほど小さい場合に有効な通信方式である．

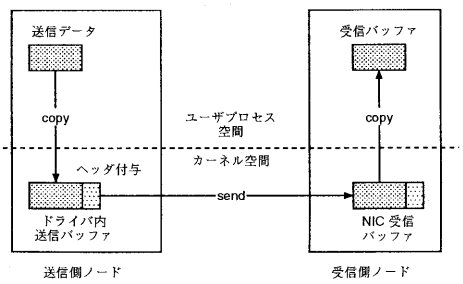

図 4: copy-send-copy 通信

サイズの大きなデータを送信する際には，copy-send-copy通信では効率の良い通信は行えない可能性がある．それは，通信のレイテンシに対してメモリのコピーにかかる時間が無視できないほど大きくなる場合において，メモリコピーにかかる時間がバンド幅性能に悪影響を与えるからである．さらに，Synfinity-0のNICの受信バッファはリングバッファ形式で管理されているため，受信バッファを圧迫するほど大きなサイズを送受信することは多数のVIによる通信をサポートする際のスケーラビリティを失なわせる要因となりえる．

そこで，データサイズの大きい場合の通信については，メモリ間コピー回数の多いcopy-send-copy通信ではなく，メモリコピーを行わない通信方式を採用することにした．この方式は，図5に示すように，まず最初に送信データの情報を受信ノードにsendを用いて送信する．受信ノードではデータの正当性をチェックした後，getにより送信データを受信バッファに取得する．この通信方式をsend-get通信と呼ぶことにする．send-get通信を用いることで，サイズの大きなデータについては，メモリ間のコピーによる性能劣化を防ぐことができる．

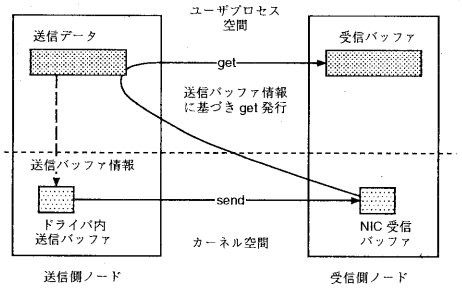

図 5: send-get 通信

このように，SCnetでは，copy-send-copy通信とsend-get通信を通信データサイズにより使い分けることで，最適な通信性能を得ることができる仕組みとなっている．

3.4 多重ネットワークサポート

一般的に，複数のネットワークを負荷分散して同時に利用することで，全体としてのバンド幅性能を向上させることが可能である．また，一つのネットワークに故障が生じた場合，そのネットワークを切り離して他のネットワークを利用することで，ネットワークの故障を隠蔽することができ，可用性の向上を図ることも可能である．SCnetでもこのような手法を利用することで負荷分散による性能向上と故障隠蔽による可用性の向上を図っている．ここでは，SCnetの負荷分散機構と可用性の向上の実現手法を紹介する．

UNIXでは，同種のデバイスがシステム上に複数存在する場合，デバイスのインスタンスごとにデバイスファイルを作成して用いることが通常である．しかし，SCnetでは敢えてこのようなデバイス管理手法を用いないことにした．それは以下のような理由による．

- NICごとにデバイスファイルを作成した場合，アプリケーションは各NICを識別した上で選択することになる．この状況で負荷分散を可能とするためには，ユーザプロセスはシステム上にNICの数と各デバイスファイルごとの負荷を知る必要がある．

- システム上に複数あるNICのうちの一つが故障した場合，新規にNICを使用するユーザプロセスは故障したNICを識別し，他の故障していないNICを利用する機構が必要である．また，故障したNICを使用していたユーザプロセスは，その故障以降は通信不能となり，結果的に可用性は実現できない．

そこで，SCnetでは図6に示すように，NICが複数システム上に存在する場合も，デバイスファイルはシステムに一つだけしか存在しない構成とした．これにより，アプリケーション単位で負荷分散を意識する必要がなくなり，統一的な負荷分散が可能となる．また，可用性の向上についても，一部のNICが故障している場合

でも，利用可能な NIC の識別をアプリケーションが行う必要はない．また，そればかりでなく，NIC を使用して通信中のプロセスに対しても，ドライバ内部で使用する NIC を切り替えることで透過的な故障の隠蔽も実現可能である．

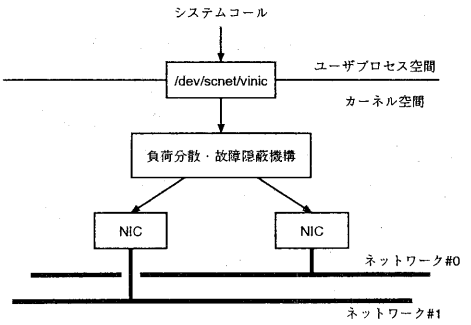

図 6: SCnet の多重ネットワークサポート

CPU	UltraSPARC-II 360MHz
CPU 数	2 個/ノード
SAN	Synfinity-0
主記憶容量	512MB/ノード
ノード数	2 (対向通信)

表 1: 性能測定環境

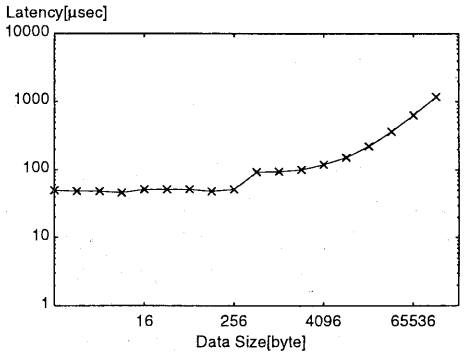

図 7: レイテンシの測定結果

4 性能測定・考察

以上のような設計に基づき実装した SCnet の性能測定を行った．測定環境を表1 に示す．測定は，1 バイトから 128 KB のデータサイズにおける通信のレイテンシとバンド幅について，VI アーキテクチャの性能測定ツールとして公開されている Intel Performance Suite を用いて行った．

レイテンシは，2 ノード間の通信において，一方のノードから送信したデータが他方のノードで受信されるまでの時間である．バンド幅については，2 ノード間の通信において，通信されるデータのサイズを通信にかかった時間で割ったものをバンド幅とした．

性能の測定結果を 図7, 8 に示す．

レイテンシの測定により，SCnet を用いた通信の 1 バイト転送時のレイテンシは 49 μ sec であることがわかった．また，図中には示されていないが，0 バイト転送時のレイテンシは 42 μ sec であった．図 7 において，256 バイト以下のデータ通信でレイテンシがほぼ一定の値を示す理由は，ユーザプロセスから渡されるデータ構造の正当性チェックやアクセスの正当性チェックなどの，データサイズに依存しないコストが影響しているからであると推測される．VI アーキテクチャの仕様上，これらのチェック処理は本来 NIC ハードウェアが行うものとされるので，ハードウェアにより VI アーキテクチャを実装することで，よりレイテンシを削減することができるものと思われる．なお，データサイズが 256 バイトから 512 バイトに移る際にレイテンシの値が比較的大きく変動しているのは，通信方式が copy-send-copy 通信から send-get 通信に変化したことが影響している．

バンド幅の測定結果では，128KB (SCnet の最大転送サイズ) 転送時のバンド幅が 852M-bit/sec (107MB/sec) であることが測定された．この測定結果はハードウェアの最大バンド幅である 240 MB/sec の半分未満である．原因は，send-get 通信方式が send と get という複数の通信プリミティブにより実現されているため，

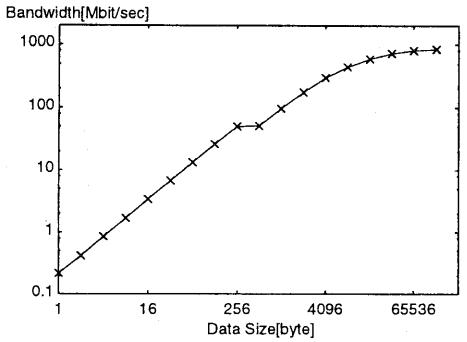

図 8: バンド幅の測定結果

send のレイテンシがバンド幅に影響を与えていることが考えられる．また，SCnetでは OS の制約により最大転送サイズを 128KB より大きいサイズにすることができないため，ハードウェアの限界性能を引き出すことができるサイズでの転送ができないことも挙げられる．

5 関連研究

M-VIA [5] は，Linux 上での VI アーキテクチャの実装であり，ネットワークハードウェアとして Fast Ethernet と Gigabit Ethernet を利用し，SCnetと同様にVI アーキテクチャの機能をソフトウェアでエミュレートすることで実現している．

6 まとめ

本稿では，UNIX上でのノード間高速通信を実現する SCnetの設計について説明を行った．SCnetは代表的な SAN の通信機構である VI アーキテクチャを Synfinity-0上でソフトウェアエミュレーションにより実現するものである．

VI アーキテクチャのソフトウェアエミュレーションによる実現手法として，特に，メモリ管理方式，通信方式，多重ネットワークサポートについて取り上げた．通信方式では，通信されるデータサイズにより通信プリミティブを使い分けることで性能を確保している．また，多重化されたネットワークをサポートすることで，高可用性と複数の VI 通信におけるスケーラビリティの確保を実現している．このようにSCnetでは Synfinity-0の持つハードウェア性能を劣化させることなく VI アーキテクチャを実現する工夫を行った．

また，実装した SCnetについて性能測定を行った．その結果，0 バイト通信時のレイテンシでは 42 μ sec であった．128KB 通信時のバンド幅では107MB/sec であった．

今後，さらなる性能向上を目的として，ハードウェアにより VI アーキテクチャをサポートした SAN を開発中である．

参考文献

[1] Compaq Computer Corp., Intel Corp., and Microsoft Corp. *Virtual Interface Architecture Specification*, December 1997. Revision 1.0.

[2] Intel Corp. *Intel Virtual Interface Architecture Developer's Guide*, September 1998. Revision 1.0.

[3] Dave Dunning et al. The virtual interface architecture. *IEEE Micro*, pages 66–76, March/April 1998.

[4] O. Shiraki, M. Nagatsuka, T. Horie, Y. Koyanagi, T. Shimizu, and H. Ishihata. AP-Net advanced high-performance network for scalable parallel server. In *Hot-Interconnects*, 1996.

[5] M-VIA Home Page. Available at http://www.nersc.gov/research/FTG/via/.