2003－OS－94　(7)
2003／8／4

次世代高性能計算機アーキテクチャ上のシステムソフトウェア開発環境

石川　裕[†]　住元真司[††]　岡家豊[†††]
久門耕一[††]　木村かず子[†††]

我々が想定する次世代高性能アーキテクチャは、コモディティハードウェアとして一方向 10Gbps 以上の性能を有する PCI Express I/O と 10 Gbps Ethernet との組合せで、高さ 3U の匡体内に 10 枚以上のプロセッサボードが格納され、IPMI 規格に準拠した保守管理プロセッサが搭載されたブレードサーバである。そのようなアーキテクチャ上で、基幹業務向けに使用されていた並列コンピュータ同様の高信頼システムを実現するシステムソフトウェアの開発環境を開発している。

開発環境として、IPMI 規格に準拠したボード上のセンサ情報を人工的に変更することにより故障模擬を行なうツール、高信頼高性能通信機構を開発する際に使用する通信路故障模擬装置を開発すると共に、高信頼システムソフトウェアにおける核となる保守管理ツール、リモートメモリアクセス機構の開発を進めている。

A System Software Development Environment for Next Generation High Performance Computer Architectures

Yutaka Ishikawa,[†] Shinji Sumimoto,[††] Yutaka Okaie,[†††]
Kouichi Kumon[††] and Kazuko Kimura[†††]

The next generation high performance architecture, which we target, consists of tens CPU boards in a 3 U hight chassis called a blade server. Each CPU board contains a PCI Express I/O, whose speed is more than 10 Gbps, and 10 Gbps Ethernet links. The board also contains a maintenance processor based on the IPMI standard. We are developing a dependable system software development environment for such an architecture so that a parallel computer for business applications will be replaced with this system.

The dependable system software development environment consists of i) a failure system simulator, which artificially generates a failure sensor information specified in the IPMI standard, ii) a failure communication simulator, which artificially produces communication errors, iii) a basic management tool for a dependable system, and iv) a remote memory access facility for high communication link.

1. はじめに

コンピュータを高速 LAN で接続したクラスタ環境が、HPC (High Performance Computing) あるいは HA (High Avaialability) の分野で使われてきた。HPC の分野では、高性能通信ライブラリによる並列化、あるいはパラメータサーチのようなジョブの間で通信を必要としない大量ジョブの並列実行に使われている。HA の分野では、ファイルオーバやジョブの負荷分散の形態でクラスタ環境が使われている。

一方、基幹系アプリケーションでは、大量のジョブを実行するために、大容量のディスクや Storage Area Network につながった共有メモリ型並列コンピュータが使われてきた。クラスタは一部のアプリケーションで利用されるに留まり、限定的な利用でしかない。

近年、I/O バスと LAN の高速化技術は目覚しく発展している。今後数年で、ハードウェア的には、あるホスト上のメモリから他のホスト上のメモリへの LAN 経由のデータ転送幅は、I/O バスのバンド幅を考慮しても 10 Gbps で転送することが可能となる。CPU とメモリ間の転送が 6.4GByte/sec であることを考えると、CPU とメモリ間転送の 1/6 の転送能力を有するインターコネクトネットワーク

† 東京大学
†† 富士通研究所
††† NEC ソフト

がハードウエア的には実現されることになる。

また、高密度実装化技術の進歩により、今後、高さ 3 U☆の匡体に 10 個以上の高性能 I/O を持つ CPU ボードが搭載された、いわゆるブレードサーバが市場に出回るであろう。

このようなハードウェア環境で、我々は、従来のクラスタで扱っていた疎結合な利用ではなく、比較的密結合な利用、すなわち従来の共有メモリ型並列コンピュータ上で実行されてきた基幹系アプリケーションが利用可能となると考えている。すなわち共有型並列コンピュータを置き換えるシステムとして、単一並列コンピュータイメージを提供するブレードサーバシステムが実現可能であると考えている。

しかし、このような環境を実現するためには、次世代高速通信ハードウエア上のネットワークプロトコル、オペレーティングシステム、運用ツール、等多岐に渡るシステムソフトウエアを開発しなければならない。このために、我々は、まず、そのようなシステムソフトウエア開発に必要な開発環境に関する研究開発を行なっている。

従来のシステムでは、ハードウエアは故障しないものと仮定してソフトウエアが開発されてきた。しかし、多数のプロセッサやディスクが接続されるシステムにおいては、構成要素の一部であるプロセッサやディスクが動かなくなっても他の正常なハードウエアを使用して、単一並列コンピュータとして動作させることが可能となる高信頼システムソフトウエアが必要となる。このような高信頼システムソフトウェア開発においては、様々なハードウエア故障を系統的に模擬し、開発したソフトウエアが耐故障性を有する高信頼性を保っているかどうかを検証するための開発支援環境が必須である。

また、高信頼システムソフトウェアを早期に実現するための、基盤ソフトウェアの整備が必須である。我々は、開発ツール、基盤ソフトウェアモジュールとして、それぞれ、以下のような研究開発を行なっている。

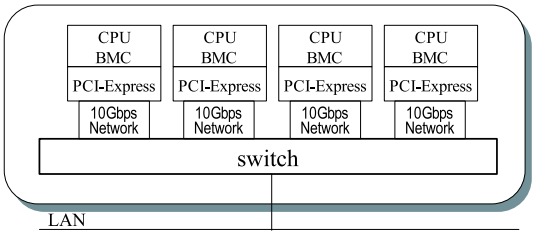
(1) 開発ツール

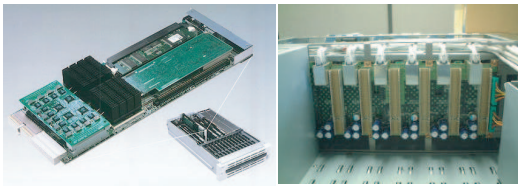
ネットワークにつながった PC 環境上でのシステム開発は非効率的であり、また、ハードウエア故障を模擬するためのツールが必要となる。我々は、User Mode Linux 上にハードウェア故障を模擬する開発ツールを開発している。

(2) 基盤ソフトウエアモジュール

システム構築の基盤となるソフトウエアモジュールの早期提供が不可欠である。ここでは、保守・監視系ツールとリモートメモリアクセス通信機構の 2 つを開発している。

☆ 1U は 約 4.45cm





写真は既存ブレードサーバの例です

図 1 想定アーキテクチャ

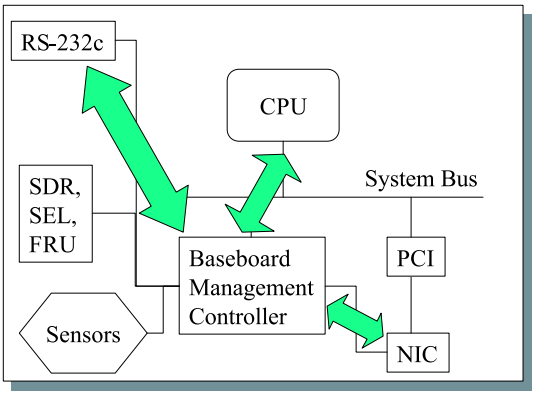


図 2 IPMI の概要

本稿では、これら次世代高性能計算機アーキテクチャ上で実現される高信頼システムソフトウェアを開発するための開発環境に関する研究について紹介する。次節では、我々が想定する次世代高性能計算機アーキテクチャについて概観する。第 3 節において故障模擬シミュレータの概要について述べる。そして、第 4 節において、管理・保守ソフトウェアアーキテクチャの概要、および第 5 節において、リモート通信機構の概要を述べる。

2. 想定システム

2.1 想定アーキテクチャ

我々は、次世代高性能計算機アーキテクチャとして、今後 5 年間のプロセッサ、メモリ、I/O バス、

LAN の技術を踏まえ、一つの匡体に格納されるいわゆるブレードサーバを想定している。図 1 に示す通り、I/O としては PCI Express[1] による一方向 10Gbps 以上の性能、LAN としてはプロトコル処理をネットワークインターフェイスカード側（以降 NIC と呼ぶ）で行なう 10 Gbps Ethernet が実用化されているだろう。

従来サーバコンピュータや専用並列コンピュータでは、各社独自の保守管理プロセッサが搭載されていた。1998 年以来、Intel 社を中心にボードや匡体の保守管理機能の仕様として IPMI(Intelligent Platform Management Interface Specification)[3] が策定されている。既に、Intel x86 系のサーバ PC には、IPMI 準拠の管理プロセッサが搭載されている。我々が想定する次世代高性能計算機アーキテクチャには、IPMI 準拠の管理プロセッサが標準で搭載されていると想定している。

図 2 に示す通り、IPMI は BMC(Baseboard Management Controller) により実現される。BMC は、IPMI が規定するボード上の電圧、CPU の FAN 回転数、CPU の温度、メモリエラー、匡体カバーの開閉、等のセンサー情報を管理している。電圧、回転数、温度などスレッシホールドをもつようなデータに対しては、上限値、下限値を設定できる。スレッシホールドを越えるような状況が発生すると障害を通知する機能がある。また、Watchdog timer によりオペレーティングシステムが正常動作しているかどうかを監視する機能もある。

CPU は BMC と KCS と呼ばれるインターフェイスで情報取得できる。また、IPMI 1.5 からは、CPU を介さず LAN やシリアル、モデム経由で情報を取得することが可能である。これらインターフェイスでは、UDP パケットを用いて直接 BMC と情報の授受が可能であり、また、遠隔によるシステム停止、再起動なども可能となっている。

2.2 想定アーキテクチャ上のシステムソフトウェア

第 2.1 節で述べたアーキテクチャ上において、図 3 に示すような管理面、運用面において単一システムイメージを提供する Linux 環境によるディペンダブルシステム構築を考えている。各 CPU 上では Linux カーネルが稼働し、ユーザプロセスに対しては単一の TCP/IP アドレス、単一 I/O 空間を提供するとともにプロセス移送を可能とする。このようなシステム上で保守管理プロセッサである BMC から得られるハードウェア障害レポートを受けて、保守管理者への自動通報、代替プロセッサによるファイルオーバ機能など、ディペンダブルシステムで必要とされる信頼性、可用性、管理性を実現する。

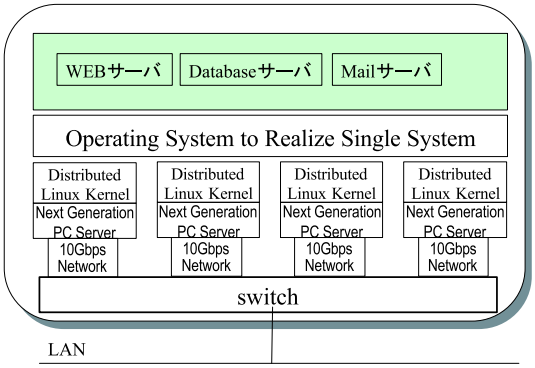


図 3 ソフトウェアアーキテクチャ

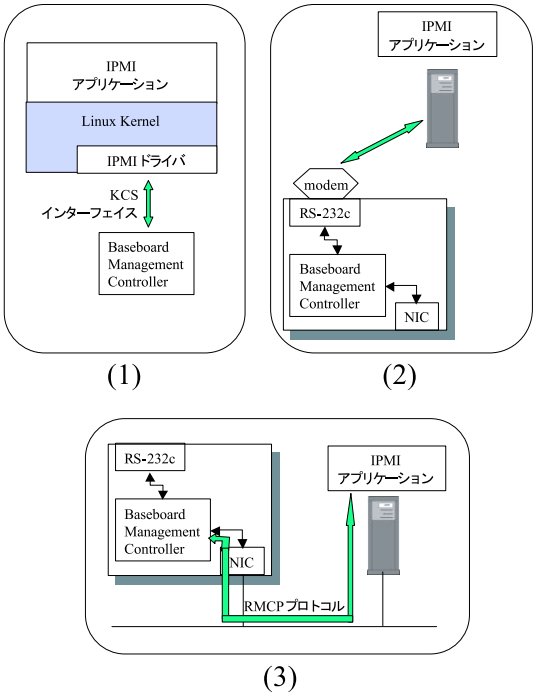


図 4 IPMI アプリケーション

保守管理プロセッサ BMC を制御するドライバである IPMI ドライバは、Red Hat Linux 8 以降のリリースにバンドルされている。IPMI ドライバにより、ボード搭載の BMC と KCS インターフェイスで通信が可能となる。図 4 に示す通り、IPMI を利用したソフトウェア（以降 IPMI アプリケーションと呼ぶことにする）は大きく 3 通りの実現方法が考えられる。

図 4(1) では、IPMI アプリケーションは、実行しているハードウェア環境を監視している。この

場合、OS がダウンすると、IPMI アプリケーションもダウンしてしまう。図 4 の (2) および (3) は、それぞれ、シリアル・モデム経由、ネットワーク経由で BMC と直接通信している IPMI アプリケーションの例である。監視対象の OS がダウンしても BMC は独立して動作しているので、障害情報を取得し、BMC に対して再起動などの遠隔操作を実現することが可能となる。

信頼性を要求するシステムソフトウェアを開発するにあたっては、開発段階においてハードウェア故障を模擬する装置により機能テストを行なっていく必要がある。また、保守管理プロセッサから上がる障害レポートに応じた基本管理ツールの早期開発が重要であると考えている。

さらに、プロセス移送等では高速通信機構を必要とする。ホスト側でプロトコル処理をしていては高速通信機構を実現できない。NIC 側で遠隔メモリアクセス (Remote Direct Memory Access: RDMA) 機能を有するシステムを構築する。

3. 故障模擬システムの概要

故障模擬システムとして、IPMI レベルでの故障模擬システムとネットワークレベルでの故障模擬システムを構築する。

3.1 IPMI レベル故障模擬システム

IPMI レベル故障模擬システムは、User Mode Linux(UML) を利用する。UML は Linux カーネル（以降 Native カーネル）上のユーザプロセスとして起動される。図 5 に示す通り、IPMI ドライバの KCS インターフェイスとの通信部分を変更し、故障模擬モジュールとの通信に変更する。故障模擬モジュールは、仮想的に BMC と同様の機能を提供する。すなわち、BMC がセンサしている CPU FAN 回転数、CPU 温度等のセンサ情報の値を人工的に生成し IPMI ドライバに伝える。

図 5 に示されている User Interface プロセスは Native カーネル上で動き、故障模擬モジュールと通信を行なうユーザインターフェイスを実現する。ユーザインターフェイスを介して、故障模擬モジュールが人工的に生成するセンサ情報を自由に変更することが可能となる。

図 6 は、ネットワーク経由で監視するツールを開発する時に使用される形態である。UML 上では、IPMI LAN 機能を模擬するプロセスを起動する。IPMI LAN Emulator プロセスは、IPMI が規定する UDP 上のプロトコルである RMCP プロトコルを処理する。開発ツールは Native カーネル上のプロセスとして起動される。このプロセスは、UML 上の IPMI LAN Emulator プロセスと RMCP プロトコルで通信する。これにより、開発ツールは仮想 BMC と RMCP プロトコルで通信しているのと等価となる。

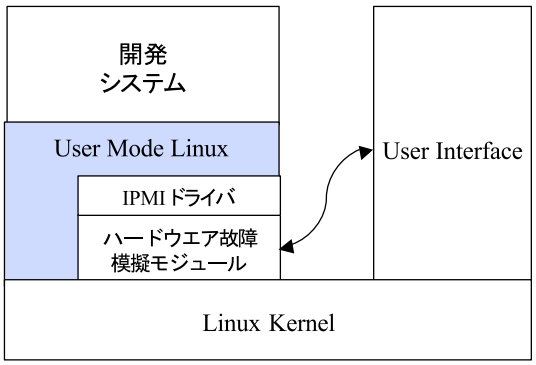


図 5 IPMI レベル故障模擬システム

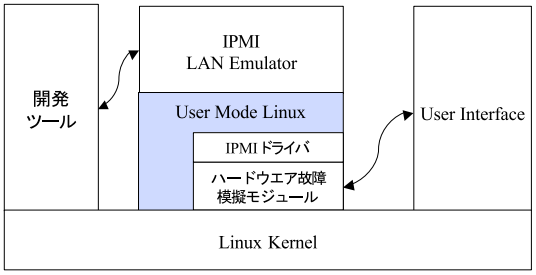


図 6 IPMI レベル故障模擬システム 1

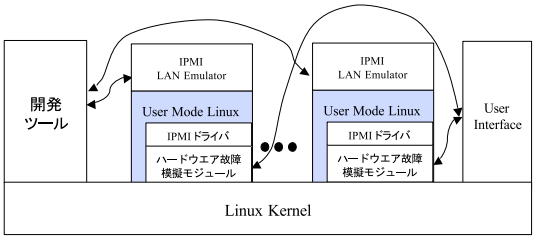


図 7 分散 IPMI レベル故障模擬システム 2

図 7 に示す通り、複数 UML を立ち上げて、それぞれで、故障模擬モジュールを含めた IPMI ドライバを使用することにより、ターゲットアーキテクチャ環境を模擬する。

3.2 通信レベル故障模擬システム

高信頼ネットワークプロトコル開発を目的として、ネットワークカード内にパケットの喪失、CRC エラーを人工的に発生させる機構および性能チューニングのために必要となる統計情報を取得する機構を実装する。NIC 上にプロトコル処理を行なう

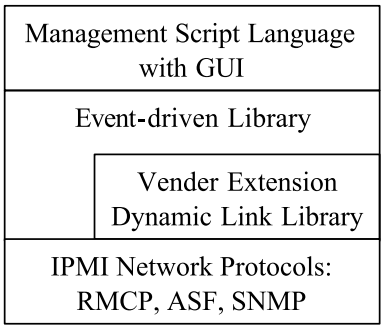


図 8 管理・保守ソフトウェアアーキテクチャ

ファームウェアと 10 Gbps Ethernet 物理層との間に FPGA を組み込む。FPGA において、パケットの喪失、CRC エラーを人工的に発生させるとともに統計情報取得機構を実装する。

4. 管理・保守ソフトウェアアーキテクチャの概要

図 8 に、管理・保守ソフトウェアアーキテクチャを示す。IPMI Network Protocol 層は、IPMI が規格化したシリアル・モデムや LAN 経由で使用できるプロトコルを実現する。プロトコルには、RMCP、ASF、SNMP が含まれる。複数の BMC が搭載された機材が管理できるように、IPMI Network Protocol 層は、同時に複数の被監視マシンと通信が出来るような構造となっている。

IPMI Network Protocol 層の上にイベント駆動型プログラミングを支援するイベント駆動ライブラリが定義される。BMC 側から上がる障害通知は非同期イベントであるために、イベント駆動プログラミングを提供するのが自然だからである。さらに、IPMI に対する要求メッセージ組み立て機構、応答メッセージの解釈機構などが提供される。

IPMI ではベンダ依存のセンサ情報を定義することが出来る。イベント駆動ライブラリにおいて、ベンダ依存センサ情報を扱うためのライブラリを動的にリンクする機能を提供する。

イベント駆動ライブラリ層の上に管理スクリプト言語が実現される。管理スクリプト言語は、GUI インターフェイス記述、障害イベントが生じた場合の対応方法の記述に利用される。

管理・保守ソフトウェアアーキテクチャの詳細は、論文 [7] で述べる。

5. リモートメモリアクセス機構

従来の TCP/IP やソケットインターフェイスでは send/receive 型のインターフェイスのみが提供されている。ユーザアプリケーションはメッセージを受信するために receive システムコールを発行し、カーネル内の受信バッファに溜ったデータを取り出す操作を行なう。ユーザアプリケーションの受信操作が滞ると受信バッファがオーバフローし、通信性能が劣化する。

遠隔メモリアクセス (Remote Direct Memory Access: RDMA) を提供するために、IP プロトコルの拡張やソケットレイヤにおける RDMA 機能を提供する API の策定などが行なわれている。RDMA 機能が提供されると、ユーザアプリケーションが陽にメッセージ受信をする必要がなくなり、また、カーネル内では受信バッファを経由せずに直接ユーザメモリに格納することが可能となり、受信バッファオーバフローに伴う通信性能の劣化が生じなくなる。

10Gbps 級のネットワークでは、ホスト側でプロトコル処理を行なっていてはネットワークの物理性能が出せない。なぜならば、パケットの到着の度に割り込みをあげるようでは、1 パケット 1,500 バイトとして、連続してパケットが到着すると 1.2 マイクロ秒毎に割り込みが生じることになる。1Gbps Ethernet では、割り込みのタイミングを遅らせる機構が導入されている。割り込みの低減に伴いホストの負荷は減るが、通信遅延が増大してしまう。

このような理由により、NIC 上での RDMA 処理を行なう機構を研究開発する。詳しくは、論文 [8] で述べる。

6. 関連技術

6.1 テスト技術

高信頼システムソフトウェアを開発するための環境として、我々はテスト環境を重視し、そのためのツール開発を行なっている。近年、ソフトウェア開発段階でのテスト基盤が注目されている。例えば、2002 年、米国 NIST の報告書 [2] では、「ソフトウェアのバグやエラーにより、米国経済は年 595 億ドル（7 兆円）の損害を被っている」と述べられ、さらに、「全てのバグはとれないが、テスト基盤を向上し開発段階の初期にソフトウェアの不具合を直せば、1/3 にあたる 222 億ドル（2.6 兆円）を削減することができるだろ」としている。

6.2 CIM

DMTF (Distributed Management Task Force) は、コンピュータ部品からプロセスの状態まで、あらゆるコンピュータ資源の情報をオブジェクト指向データベース化しようと標準化を進めている。

DMTF で規格化しているデータベーススキーマは CIM (Common Information Model) と呼ばれ、CIM データベースサーバは CIMOM (CIM Object Manager) と呼ばれる [4]。

CIMOM はプロバイダと呼ばれるインターフェイスを持ち、機器情報や OS からの情報をデータベースに登録する枠組を提供している。プロバイダには、SNMP や IPMI が想定されている。

CIMOM の実装としては、SNIA CIMOM, openPegasus, openWBEM などがオープンソースとして入手可能である。これらは、問い合わせ型のインターフェイスしか提供していない。また、プロバイダモジュールは、CIMOM プロセス内に動的にリンクされて実行される形式をとり、非同期に情報を収集してデータベースに登録する枠組は提供されていない。また、分散環境を想定した複数の CIMOM 同士での交信や、分散された多数の機器データを管理する枠組は提供されていない。これら機能の実装は現在行なわれている状況である。

CIMOM を利用する利点は、様々なセンサー情報を得るために、個々に規格化されている API を実装することなく、統一 API でデータを得ることが出来ることである。しかし、現状の CIMOM では、我々が考えている保守ツールを実現するための機能が実現されていない。CIMOM の枠組を拡張していくアプローチもある。しかし、我々の関心は保守・監視ツールによる高信頼システムの実現であるために、CIMOM そのものを拡張するアプローチをとらない。将来的には、CIMOM とのインターフェイスを持ち、IPMI から得られるセンサー情報以外の情報を利用した環境を考えている。

6.3 openIPMI

IPMI 上のツールとして、openIPMI がある [6]。Linux カーネル用ドライバとドライバを使ったユーザレベルライブラリとツールがある。

ユーザレベルライブラリでは、イベントドリブン型のインターフェイスを提供している。各センサ情報を表示する簡単なツールが提供されているに過ぎない。

6.4 openSSI

シングルシステムイメージを提供する Linux カーネルとして OpenSSI[5] がある。OpenSSI では socket を持たないプロセスのプロセス移送が実現されている。現在、OpenSSI の機能をベースにしたシステム構築を検討している。

7. おわりに

本稿では、次世代超高速通信ハードウエアを使用した基幹業務向け並列コンピュータを実現するために必要なシステムソフトウェア開発環境の構想について述べた。次世代ハードウェアとして、PCI Express と 10 Gbps Ethernet でつながった PC サーバで IPMI 準拠の管理プロセッサが搭載されたブレードサーバ環境を想定している。高信頼システムソフトウェアを実現するために、開発段階から機能検証を可能とするような故障模擬シミュレータを開発し、システムソフトウェアの試験に使用する。

論文 [7] では、IPMI 規格に基づく故障シミュレータについて詳しく報告する。論文 [8] では、10Gb Ethernet 上の通信プロトコル作成支援技術について詳しく報告する。

平成 15 年度末までには、最初のプロトタイプシステムを構築し、試験運用を行なう予定でいる。

謝　辞

本研究の一部は、文部科学省「eSociety 基盤ソフトウェアの総合開発」の委託を受けた東京大学石川研究室、東京大学石川研究室と富士通研究所、東京大学石川研究室と NEC ソフトとの共同研究契約に基づいて行なわれた。

参　考　文　献

1) PCI Express, http://www.pcisig.com/specifications/pciexpress
2) NIST, “The Economi Impacts of Inadequate Infrastructure for Software Testing,” RTI Project Number 7007.011, 2002.
3) Intel, Hewlett-Packard, NEC, and Dell, “-IPMI - Intelligent Platform Management Interface Specification, V1.5,” 2002.
4) CIM and CIMOM, http://www.dmtf.org
5) openSSI, http://www.openssi.org
6) openIPMI, http://openipmi.sourceforge.net
7) 岡家、木村、石川、「IPMI 規格に基づく管理保守系システムソフトウェア」、情報処理学会研究報告 03-OS-12 (SWOPP03)、 情報処理学会、2003.
8) 住元、久門、石川、「10Gb Ethernet 上の通信プロトコル作成支援技術」、情報処理学会研究報告 03-OS-12 (SWOPP03)、 情報処理学会、2003.