2003－OS－94　(8)
2003／8／4

# IPMI規格に基づく管理保守系システムソフトウェア

岡家　豊[†]　木村　かず子[†]　石川　裕[††]

IPMI イニシアティブが規格している IPMI (Intelligent Platform Management Interface) は、ボード上の温度、電圧、冷却ファンなどを監視するハードウェア仕様である。IPMI 規格は、信頼性、可用性、保守性、管理性を備えたいわゆるディペンダブルシステムを実現するハードウェア支援技術である。ディペンダブルシステムを実現するには、故障診断ツール、監視通報ツールなどの管理保守系システムソフトウェアが必要である。このようなシステムソフトウェアを開発するためには、開発段階から想定している機器故障に対応する機能が仕様通りに動作しているか確認するテスト環境が必須である。

我々は、管理保守システムソフトウェアの開発と同時に機器故障を模擬するシミュレータも開発している。故障シミュレータは、IPMI 規格で定義されているセンサ情報やベンダ固有情報に基づいて人工的に情報を生成する。故障シミュレータは、Linux カーネル上の一プロセスとして稼働する User Mode Linux のカーネルモジュールとして実現される。

# The Development of Management System Software based on the IPMI Standard

Yutaka Okaie,[†] Kazuko Kimura[†] and Yutaka Ishikawa[††]

IPMI (Intelligent Platform Management Interface specification), defined by IPMI Initiative, is the specification of hardware that observes temperature, voltage, cooling fan, and so on. The IPMI specification supports hardware mechanisms that realizes a reliable, available, and manageable system so-called *dependable system*. To realize such a dependable system, it is necessary to develop maintenance system software, such as a diagnostic tool, watchdog and reporting tool, and so on. To develop such system software, a testing environment is required, which allows us to check whether functions corresponding to failures assumed on the developing stage work well.

We are developing the failure simulator of machinery and tools as well as the maintenance system software. Based on the sensor information and vendor proprietary information defined in IPMI, the failure simulator generates information artificially. The failure simulator is implemented as kernel modules of User Mode Linux that works as a process on the native Linux kernel.

## 1. はじめに

企業における基幹系システムを中心に高い信頼性、可用性、保守性、管理性を備えたシステムのニーズが高まり、様々な企業でそのようなシステムを実現するための取り組みが行なわれている。例えば、IBM 社における autonomic computing、Intel 社における modular computing、Microsoft 社の trustworth computing などである。本稿ではこのようなシステムを総称してディペンダブルシステムと呼ぶことにする。

ディペンダブルシステムを実現するシステムソフトウェアとして管理保守系システムソフトウェアが必要である。管理保守系システムソフトウェアには、故障診断ツール、監視・通報ツールなどが含まれる。例えば、システム障害が生じると監視通報ツールが自動的に遠隔地にある監視センタに通報する。監視センタでは遠隔操作によりシステム障害が生じたシステムの保守プロセッサを使用して故障診断ツールを動かし、障害を特定する。

従来、企業における基幹系システムで使われているサーバコンピュータでは、ハードウェアメーカ毎に独自の監視ハードウェア上で管理・保守ツールが開発されてきた。1998 年以来、IPMI イニシアティブ (Intel、DELL、Hewlett-Packard および NEC の 4 社) によって IPMI (Intelligent Platform Management Interface) と呼ばれる監視システムの規格化が行なわれ

† NEC ソフト株式会社
†† 東京大学

ている。本規格は、サーバの温度、電圧、冷却ファン、電源などを監視するハードウェア仕様を標準化している。最近の x86 系 CPU を搭載したサーバ系のボードには、IPMI 規格に準拠した監視プロセッサが搭載されている。

我々は、IPMI 規格に準拠したサーバシステムと Linux カーネル上でディペンダブルシステムを実現するためのシステムソフトウェアの開発を行なっている。最初の取り組みとして、管理保守システムソフトウェアの開発を行なっている。開発する管理・保守システムは、被監視サーバ上の障害を検知すると監視センタに通報する機能だけでなく、故障の度合により自動的に冗長部品に切り替える機能やスタンバイコンピュータへのサービス移行機能などを提供する。

このようなシステムソフトウェアを開発するためには、開発段階から想定している機器故障に対応する機能が仕様通りに動作しているか確認するテスト環境が必須である。そこで、我々は、管理保守システムソフトウェアの開発と同時に機器故障を模擬するシミュレータも開発している。故障シミュレータは、IPMI 規格で定義されているセンサ情報やベンダ固有情報に基づいて人工的に情報を生成する。故障シミュレータは、Linux カーネル上の一プロセスとして稼働する User Mode Linux のカーネルモジュールとして実現される。

以下、本稿では、次節で IPMI 規格の概要を説明した後、第 4 節において、現在設計中の管理・保守系システムソフトウェアの概要を述べる。第 5 節では、IPMI 規格に基づく故障模擬シミュレータについて述べる。本研究開発に関連する事例を第 6 節で述べる。

## 2. IPMI 1.5 規格 [1)]

IPMI (Intelligent Platform Management Interface) は 1998 年に IPMI イニシアティブ (Intel、DELL、Hewlett-Packard および NEC の 4 社) によって定義された規格である。本規格はサーバの温度、電圧、冷却ファン、電源などを監視するインテリジェントなサーバハードウェアに対するメッセージングや共通インターフェースを定義したものである。ここで、インテリジェントなサーバハードウェアとは、自動監視、ロギング、自動復帰といったマネジメント機能がハードウェアもしくはファームウェアに直接、実装されているものである。

従来、サーバのハードウェア監視のための共通インタフェースは存在しなかったため、OS やハードウェアに依存して管理ソフトウェアを作成する必要があった。IPMI が規定する共通インタフェースにより、被監視対象マシンの OS やハードウェアの違いを意識しないで管理ソフトウェアの作成が可能となった。複数の OS やハードウェアが混在する環境下においてリモートサーバを統一的に監視するソフトを作成するのに、この点は大変重要である。

IPMI 規格のハードウェアの構成は図 1 のようになっている。以下の小節で構成要素について述べる。

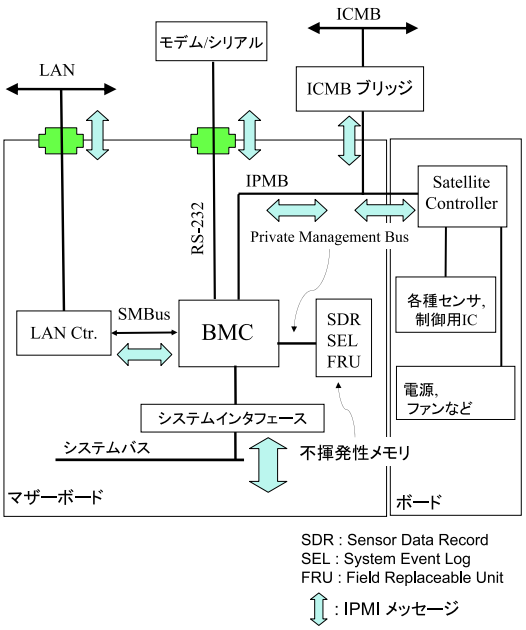


**図 1** IPMI のハードウェア構成

### 2.1 BMC、IPMB および ICMB

IPMI 規格に準拠したボード上には **BMC (Baseboard Management Contrpoller)** と呼ばれるコントローラがあり、ハードウェアと管理ソフトウェア間の標準インタフェースを提供する。

本規格では、増設されたボード上の BMC と接続するための専用の内部バス **IPMB (Intelligent Platform Management Bus)** についても規定されている。IPMB はシリアルバスで、同一筐体内のモジュール間を結んでいる。この時、マザーボード上の BMC と、増設ボード上の BMC を区別するために、後者を **Satellite Controller** と呼んでいる。

また、筐体同士を接続し、その間での BMC 情報取得を行うための外部バス **ICMB (Intelligent Chassis Management Bus)** も本規格で定められている。

### 2.2 IPMI インタフェース

BMC にアクセスするインタフェースとして、シリアルモデム、LAN、および CPU とのシステムインタ

フェースに対応するものが提供されている。次の小々節でこれらについて述べる。

これらのインタフェースを介して管理ソフトウェアと BMC 間で、IPMI コマンドと呼ばれる、IPMI で定義されている要求・応答型のメッセージのやりとりを行う。

#### 2.2.1 LAN インタフェース

LAN インタフェースにおいては、UDP 上のプロトコル RMCP (Remote Management Control Protocol) を用い、BMC へ直接アクセスを行う。この機能は IPMI over LAN と呼ばれる。

RMCP は DMTF (Distributed Management Task Force)[8] によって定められた形式である。RMCP を用いることで、システムのブート前や OS 不在時のメッセージ送受信も可能である。これは DMTF が別途規格した ASF (Alert Standard Forum) の機能によるものである。

#### 2.2.2 シリアルモデムインタフェース

本インタフェースでは、直接シリアル、もしくは外部のモデムを経由して BMC と IPMI メッセージをやりとりするのに以下 3 つのデータ形式が異なる接続モードが定められている。

(1) Basic Mode : 最低限のフレームで IPMI メッセージを包み込んで送受信するモード。3 つのモード中で性能的には一番良いが、シリアルアプリケーションを作成する際は、IPMI 規格を意識して作成しなければならない。

(2) PPP Mode : LAN インタフェースの場合と同様に RMCP を用いて IPMI メッセージを PPP で送受信するモード。LAN ベースのソフトウェアとの親和性が高いが、性能的には少し劣ってしまう。

(3) Terminal Mode : 入力した文字が そのまま IPMI メッセージとして BMC に送受信するモード。行指向のインタフェースを提供する。直接シリアル接続して使うことが想定されている。性能的には Basic Mode より悪く、機能的にも他のモードと比べ劣っている。

#### 2.2.3 システムインタフェース

キーボードインタフェースと同様に、I/O ポートに対する入出力命令を使って BMC と通信するインタフェース。代表的なものに KCS (Keyboard Controller Style) がある。

### 2.3 センサ情報取得

モニタされた温度、電圧、ファン回転数などの情報取得は、直接対象を監視しているハードウェアにアクセスして得るのではなく、第 2.2 節で述べた IPMI コマンドを通して取得する。このアプローチにより、監視対象のハードウェア構成に依存することなく管理ソフトウェアを開発することができる。

表 1 センサタイプの概要

| タイプ | 概要 |
|---|---|
| 温度 | CPU 温度 |
| 電圧 | ボード上の 5V 系、12V 系の電圧 |
| ファン | CPU ファンの回転数 |
| 筐体イントリュージョン | 筐体のドアが開閉、<br>LAN ケーブルの装脱着 |
| プロセッサ | 熱暴走等の異常 |
| メモリ | ECC エラー |
| 電源ボタン | 電源、リセット等のボタン押下 |
| LAN | NIC 異常 |
| バッテリ | バッテリ残量 |

### 2.4 Private Management Bus

図 1 に示すように、BMC には Private Management Bus 上に不揮発性メモリがつながっている。不揮発性メモリの中には、以下で述べる SDR、SDR Repository、SEL、および FRU 情報が格納されている。

Private Management Bus を通して、BMC と不揮発性メモリ間で IPMI メッセージのやりとりが可能なので、メインプロセッサ、BIOS および OS の状態とは独立に情報を取り出すことが可能である。

### 2.5 SDR

SDR (Sensor Data Record) は BMC または Satellite Controller が持つセンサ情報である。

センサ情報としては、ボード上のセンサ数、各センサが何を監視しているか (電圧、温度、回転数、筐体カバーの開閉、メモリエラーなど)、スレシュホールドを持つか、監視対象に異常が発生した時に生成するイベントの情報がある。

例えば、1.2 V の電圧センサの SDR には以下のものが含まれている。

- SDR (IPMI) の バージョン : 1.5
- 上限値 : 1.31
- 下限値 : 1.08
- 単位 : V

IPMI 規格で定められているセンサのタイプは全部で 41 種類ある。ここでは、主だったタイプの概要を表 1 に示す。

### 2.6 SDR Repository

システム内に存在するであろう Satellite Controller、センサ、およびデバイスの情報は BMC の **SDR Repository** として保存される。

実際は、SDR Repository は過去のある時点での上記情報を保持している。SDR Repository を利用することで、システム内に存在していることが想定されて

いたデバイスが存在しない、もしくは認識されていないことを検知可能である。

SDR の持つセンサ情報は、現在のセンサに対するものであり、あるデバイスがシステム起動時に外れていたとしても、それを検出できない。

### 2.7 SEL とイベントメッセージ

BMC および Satellite Controller は、各々の SDR のセンサ情報に基づいてデバイスを監視している。デバイスに異常 (e.g. CPU 温度がスレシュホールドを越えた、電源が落ちている) が発生した時、その旨を伝える**イベントメッセージ**と呼ばれるメッセージを生成し、BMC に送る。

この時に発生した異常のことを**イベント**と呼び、そのログは **SEL (System Event Log)** と呼ばれる。

ログとして書き込まれる情報には以下のものが含まれる。

(1) イベント発生時刻
(2) イベント内容

イベント内容の例としては、CPU ファンの回転数が SDR の上限値を越えたり、筐体カバーが開いているなどのシステムの異常が挙げられる。

### 2.8 FRU 情報

**FRU 情報**はシリアル、パーツ、モデルの番号および、センサタイプの情報である。この情報は、故障が発生した際、どのデバイスで発生したかものかを特定し、デバイスの交換をする際に利用される保守管理情報である。

### 2.9 PEF

あるイベントが発生した旨が BMC に伝えられた時に、何らかのアクションを実行させる仕組みを **PEF (Platform Event Filtering)** と呼んでいる。

アクションとしては、システム停止、再起動、NMI (Non-Maskable Interrupt) および、アラート通知が指定できる。

例えば、第 2.5 節で挙げた例において、電圧が SDR で規定されるスレシュホールドを越えた場合、アクションが PEF により定義される。

### 2.10 アラート通知機能

故障などのイベントが発生した時に、その旨を**アラート**として通知する機能がある。本規格ではアラート通知に LAN またはシリアルモデムを経由するものが定められている。

#### 2.10.1 LAN アラート

LAN を経由してのアラート通知は SNMP Trap を含む UDP のデータグラムを生成して行われる。この時のデータフォーマット **PET (Platform Event Trap format)** も規定されている。

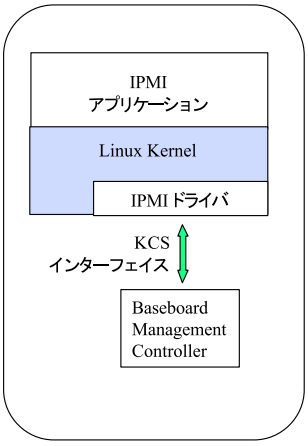


**図 2** IPMI アプリケーション

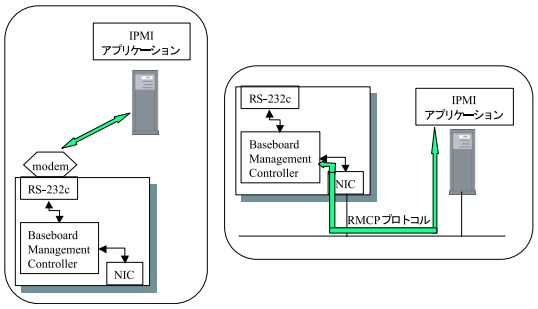


**図 3** IPMI アプリケーション

#### 2.10.2 シリアルモデムアラート

シリアルモデム経由のアラート通知機能は大別して 2 つある。1 つは PPP でリモートの LAN に接続し、LAN アラートと同様に SNMP Trap のフォーマットで通知するもの、2 つは外部モデム、またはシリアルモデムに接続し、指定されたページャに通知するものが定められている。

### 2.11 Watchdog timer

IPMI で定義されている Watchdog timer は BIOS や OS 上で動作するアプリケーション上で使用できる。このタイマが切れた時に、システム停止、再起動などの動作を指定できる。

この時、タイマが切れたというイベントが発生するので、第 2.10 節で述べた仕組みを用い、その旨を通知することも可能となっている。

## 3. IPMI アプリケーション

図 2 および図 3 に示す通り、IPMI を利用した管理保守ソフトウェア（以降 IPMI アプリケーションと呼ぶことにする）は大きくシステムインターフェイスとネットワーク経由の 2 通りの使用形態がある。

### 3.1 システムインターフェイス

図 2 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、CPU 故障等で OS がダウンすると、IPMI アプリケーションも動かないので、管理保守ソフトウェアとしての役割を果たすことができない。

主に通常のハードウェア監視およびログ情報の格納などを実現する場合に利用される他、ネットワーク故障で外部との接続が不可能な時の対応に利用される。

IPMI アプリケーションが、そのアプリケーションを実行しているハードウェアから IPMI アラート通知を受け取るのは BMC からの IRQ 割り込みが使われる。

### 3.2 ネットワーク経由

図 3 では、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。CPU や主記憶の故障により監視対象の OS がダウンしても BMC は独立して動作し、障害情報を IPMI アプリケーションに通報する。IPMI アプリケーションは、障害情報を入手すると、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。

## 4. 管理保守系システムソフトウェア

### 4.1 管理保守ソフトウェアの使用例

現在設計中の管理保守ソフトウェアの機能概要を例示する。

#### 4.1.1 自動負荷分散運転機能

例えば、複数台のコンピュータで http 要求を負荷分散して処理している状況を考える。負荷分散を行なっているコンピュータ上の自動負荷分散運転機能は、http 要求が少ない時には、コンピュータの大部分を ASF プロトコルで停止させる。要求が増えてくると、ASF プロトコルで停止させていたコンピュータを起動させ、http 要求の負荷分散を行なう。

#### 4.1.2 電源投入シーケンサ

多数の大規模サーバを有するセンタでは、サーバ全部を同時に電源投入すると電源投入時に生じるピーク電流が大きくなり許容電流容量を越えることがある。これを回避するために、ASF プロトコルのブートコマンドを使用して、電源投入を制御することが可能である。

#### 4.1.3 一元監視機能

様々な用途のコンピュータ群の状態を一元監視するためには、ユーザの要求に応じて、表示項目の選択、表示方法、操作機能をカスタマイズしたい。たとえば、全てのコンピュータの匡体イントリュージョンだけを

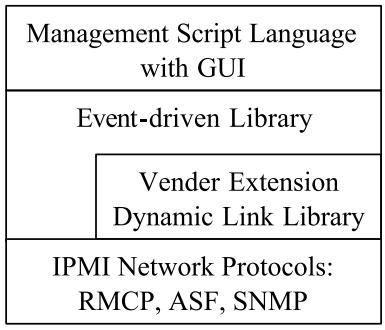


**図 4** 管理・保守ソフトウェアアーキテクチャ

監視したり、全てのコンピュータの温度を監視したり、と目的によって表示する項目を決めたい場合がある。監視対象のコンピュータ群の中から選択して表示したいという要求もあるだろう。

このようなカスタマイズを可能とするスクリプト言語を提供する。ユーザはスクリプト言語を用いて、ユーザ毎に監視ツールをカスタマイズすることが可能となる。

#### 4.1.4 異常診断機能

監視しているサーバで、Watchdog リセットで OS がダウンした場合、その原因を解明するためには次のような手順がとられる。

(1) SEL を解析し、異常を示すイベントが発生していたか調べる。異常が見つかれば、IPMI の FRU 情報から交換すべき部品を特定する。
(2) 診断システムを立ち上げて、ハードウェアが故障していないか調べる。

管理保守ソフトウェアは、このような手順を自動化する。

#### 4.1.5 異常回復機能

最近の PC サーバでは、冗長部品を持たせることにより信頼性向上を図っている。例えば、2 系統の LAN を有するサーバにおいて、一方の LAN が故障した場合を考える。管理保守ソフトウェアは、LAN 故障イベントを取得して、他方の LAN に切り替えると同時に、管理者に通報する。

#### 4.1.6 異常時サービス移動

例えば、WEB サーバを考える。WEB コンテンツは高信頼ファイルサーバで管理されている、あるいは、WEB コンテンツは代替コンピュータ上にコピーされている、とする。サービス中の WEB サーバの CPU 温度異常が検出されると、管理保守ソフトウェアは、管理者に通報すると共に、代替コンピュータの電源を投入し、WEB サーバ機能を代替コンピュータに行なわせる。

### 4.2 ソフトウェアアーキテクチャ

第 4.1 節で述べた機能を実現するソフトウェアを実現するために、図 4 に示すような管理保守系ソフトウェアアーキテクチャを設計中である。

#### 4.2.1 IPMI Protocol 層

IPMI Protocol 層は、IPMI で規格化されたシステムインターフェイスおよびシリアル・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 Protocol 層は、同時に複数の被監視マシンとの通信機構も提供される。

#### 4.2.2 イベント駆動ライブラリ

IPMI Network Protocol 層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。保守管理系ソフトウェアと BMC との間では、要求・返答型メッセージと BMC から通報される障害報告の 2 種類のインターフェイスがある。BMC からの通報は非同期イベントである。このため、イベント駆動プログラミングスタイルを提供するのが自然だと考えている。

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。このようなライブラリでは、例えば、ベンダ依存センサにおけるセンサ情報に含まれるコードが、どのような意味なのか解説する機能が提供される。

#### 4.2.3 管理スクリプト言語

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、オブジェクト指向型スクリプト言語で、GUI 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。

GUI 記述に関しては、以下のような機能を管理スクリプト言語に導入する。

(1) 表示コンピュータの選択機能
多数のコンピュータ機器を管理する場合に、管理対象コンピュータ毎の表示、監視センサ情報に関する全ての管理対象コンピュータの状態を表示する、など、ユーザの要求に合わせた選択機能を提供する。

(2) 表示するセンサ情報の選択機能
GUI ウインドウのトップに、常に表示しておきたいセンサ情報、必要に応じて表示したい情報、などはユーザの目的によって異なる。これらをカスタマイズするのに必要な機能を提供する。

(3) センサ情報の表示形式選択機能
数値データ表示法には、数値、グラフ、等様々な形式がある。これらを選択できる機能を提供する。

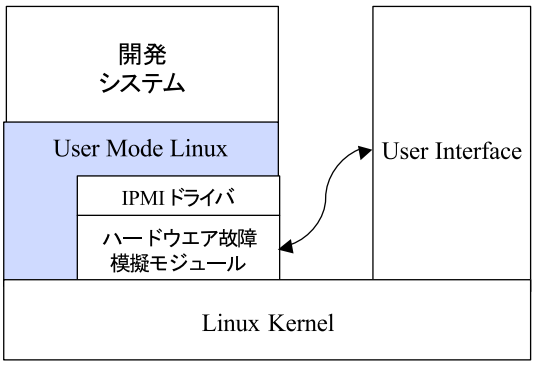


**図 5** IPMI レベル故障模擬システム

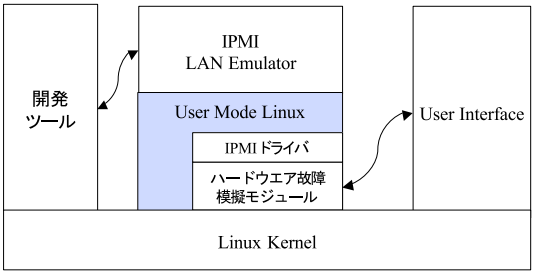


**図 6** IPMI レベル故障模擬システム 1

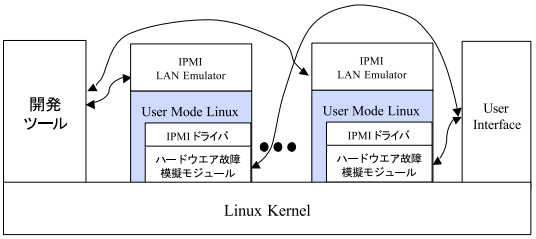


**図 7** 分散 IPMI レベル故障模擬システム 2

## 5. IPMI に基づく故障模擬シミュレータ

本節では IPMI に基づいた故障模擬シミュレータについて述べる。管理ソフトウェアの評価を行う際、実際に故障を発生させての動作検証を行うには、ハードウェアの改造等が必要となり、そのコストは甚大である。本シミュレータを用いることで、開発期間およびコストの削減が可能となるだけでなく、信頼できるシステムソフトウェアが早期に提供可能となる。

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。

### 5.1 システムインターフェイス

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。

また、IPMI で定義されている不揮発性メモリに格納される、SDR、SEL、FRU 情報を管理する。UML プロセス終了後も不揮発性メモリが保存されるように、不揮発性メモリの内容は、Native Linux 上のファイルとして実現される。

図 5 に示されている開発システムとは、開発する IPMI アプリケーションである。この使用例では、IPMI アプリケーションは、IPMI ドライバを使用して IPMI の情報を得る。

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。また、不揮発性メモリ管理も可能である。

### 5.2 ネットワーク経由

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。開発ツールは仮想 BMC と RMCP プロトコルで通信していることを意味する。

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。

## 6. 関 連 技 術

### 6.1 故障模擬システム

論文 [7] では、障害発生時のマイクロカーネル OS の信頼性を評価する方法として、システムに様々な障害を注入し、マイクロカーネル OS の振る舞いを解析するための支援ツール MAFALDA (Microkernel Assessment by Fault injection AnaLysis and Design Aid) を使ったものについて述べられている。

このツールを使い、システムコールで使われるパラメータや、メモリの状態を人工的に変更することで、カーネルの各構成要素に対する障害を発生させている。また、障害に対する Wrapper を作成することで、発生した障害が上位のソフトウェアに影響を及ぼさないような仕組みを与えると同時に、障害検知を行っている。

現段階で我々が提案する故障模擬システムでは IPMI 規格で定められている故障 (イベント) のみを対象としているが、上記のような障害や故障の模擬も必要か検討している。

### 6.2 監視ソフトウェア

IPMI を用いた監視ソフトウェアの商用化例としては、NEC 製の ESMPRO[5][6] がある。これは、Windows 上で稼働するソフトウェアである。ESMPRO 自体は KCS インタフェースを用いて実装されている。IPMI over LAN および IPMI over Serial を用いて実装された同社製品 MWA (Management Workstation Application) と連携させることで、リモート監視機能を実現している。

監視対象やアラート通報先の設定、電源管理、および IPMI 情報取得がリモートから GUI で行える。また、MWA と BIOS でリモートから対象ホストの画面を操作することが可能なリモートコンソール機能や、リモートのデバイスを、あたかも対象のデバイスであるかのように扱えるリモートドライブ機能が提供されている。

PC サーバ用の監視ソフトウェアの商用化例としては、HP 製 Insight Manager や Linux NetworX 製の独自ハードウェア ice box およびその管理ソフトウェア Clusterworx[9] がある。

Insight Manager は Windows 上で動作する Web アプリケーションである。SNMP と Desktop Management Interface (DMTF によって策定された PC や周辺機器を管理するための標準インタフェース) を利用した管理ソフトウェアである。

Clusterworx は、様々な OS 上で稼働させるために Java で記述されている Web アプリケーションである。CPU 温度、電源状態など、ハードウェアの情報取得、および OS の稼働状況の取得も行える。また、クラスタ全体におけるディスクイメージを一定に保つのに、独自のディスククローニング機能を提供している。

クラスタ監視ソフトウェアとして、カリフォルニア大学バークレ校が開発した ganglia[10] がある。被監視サーバ上の、システムの情報を採取するプログラ

ムと、監視サーバ上の採取された情報をまとめるプログラムが連携して動作する。クラスタの状態をブラウザから閲覧できる。また、ホストの詳細な状態取得やパラメータ設定を行う Linux コマンドも提供されている。

### 6.3 ディペンダブルシステム

ディペンダブルシステムに向けて、各社様々な取り組みが行なわれている。例えば、IBM 社における autonomic computing、Intel 社における modular computing、Microsoft 社の trustworth computing などがある。我々が知る限り技術詳細が公開されていないので、各社の取り組みの技術的背景は不明である。Intel 社による技術説明資料では、IPMI 技術を中核に、冗長部品との組合せで、Intel 社が掲げる modular computing を実現しようとしている。

### 6.4 資源情報管理

論文[2]で述べたように、DMTF (Distributed Management Task Force)[8] 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。CIMOM では、SNMP、IPMI などから取得可能な機器情報を CIM データベースサーバに登録、あるいは、CIMOM 経由で機器に対して要求を行なうことを目的としてプロバイダと呼ばれる API を定義している。

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。実際、IPMI が規定する情報は IPMI プロバイダとして定義され、CIMOM 上でアクセスできる枠組は整っている。しかし、現在、我々が必要とするアラート通知機能を含めた CIMOM、IPMI プロバイダの実装は、我々が知る限り、存在しない。

我々は CIM を含めた管理体系を主眼として研究開発を進めているのではなく、ディペンダブルシステムのための管理保守ソフトウェア構築が主眼である。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。

## 7. おわりに

本論文では、IPMI 規格に準拠したディペンダブルシステムを実現するために必要となる保守管理系システムソフトウェアに関する我々のアプローチについて述べた後、当該ソフトウェア開発時に必要となる故障シミュレータの概要を述べた。


## 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および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) Intel, Hewlett-Packard, NEC, and Dell, "-IPMI - Intelligent Platform Management Interface Specification, V1.5," 2002.
2) 石川、住元、久門、木村、岡家、「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」、情報処理学会研究報告 03-OS-12 (SWOPP03)、 情報処理学会、2003.
3) 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.
4) Hewlett-Packard、「Insight Manager 7 User Guide」、2002.
5) NEC、「White Paper NEC Express 5800 サーバ リモートマネジメント機能」2000.
6) NEC、「MWA Ver.3.10.xx ファーストステップガイド」第 46 版, 2002.
7) Jean Arlat, Jean-Charles Fabre, Manuel Rodriguez and Frederic Salles, "Dependability of COTS Microkerlnel-Based Systems", IEEE Transactions on Computers, Vol. 51, No. 2, 2002
8) http://www.dmtf.org
9) http://www.linuxnetworx.com/products/
10) http://ganglia.sourceforge.net