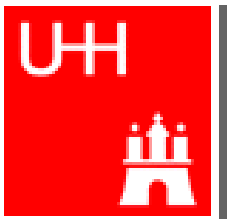

Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften
Fachbereich Informatik

Hausarbeit im Seminar:
Systemmonitoring unter Linux
Sommersemester 2010

Intelligent Platform Management Interface

Timme Katz

timme.katz@informatik.uni-hamburg.de
Studiengang M.Sc. Informatik
Matr.-Nr. 5800173
Fachsemester 4
Abgabe: 22.09.2010

Betreuer: Timo Minartz

Inhaltsverzeichnis

1 Einleitung

Diese Seminararbeit beschäftigt sich mit dem „Intelligent Platform Management Interface“, einer herstellerübergreifenden Spezifikation zur Überwachung und Verwaltung von Rechnern. Im folgenden Abschnitt wird eine kurze Motivation für das Thema dargestellt und ein Überblick über die Arbeit gegeben.

1.1 Motivation

Das Überwachen des eigenen Rechners unter dem Schreibtisch fällt meistens relativ leicht. Wenn das System ausgelastet ist, sind die Lüfter deutlich zu hören. Auch eine defekte Festplatte macht sich meistens akustisch bemerkbar. Außerdem hat der Anwender immer physikalischen Zugriff auf sein System. In Rechenzentren sind all diese Voraussetzungen nicht gegeben. Im Rechenzentrum ist es immer laut, der Anwender sitzt nicht direkt vor dem Server den er gerade benutzt und die Maschinen laufen physikalisch weitentfernt von den Anwendern oder Administratoren. Aus diesen Gründen sind andere Methoden zur Überwachung und Verwaltung der Systeme nötig, welche vor allem den Fernzugriff über ein Netzwerk unterstützen müssen. Gerade in großen heterogenen Umgebungen sollte eine Verwaltungssoftware eingesetzt werden, die hersteller- und betriebsystemübergreifend verfügbar ist.

1.2 Zielsetzung

Diese Arbeit beschäftigt sich mit dem „Intelligent Platform Management Interface“ (IPMI) im Kontext des Systemmonitoring. Ziel der Arbeit ist es einerseits einen Überblick über die interne Funktionsweise von IPMI und seine Möglichkeiten zu geben sowie andererseits den praktischen Einsatz an zwei Beispielen vorzustellen.

1.3 Aufbau der Arbeit

Nach der Einleitung in das Thema Systemmonitoring werden die technischen Grundlagen und die Funktionsweise von IPMI erläutert. Anschließend soll ein Überblick über die Bereiche Monitoring und Fernzugriff gegeben werden. Im Bereich des Monitoring wird genauer auf die Sensoren und das Abfragen der Sensorwerte eingegangen. Zum Thema Fernzugriff wird der Fokus auf „Serial over LAN“ (SOL) gelegt. Dieses Feature erlaubt den Zugriff auf die serielle Schnittstelle des überwachten Systems über das Netzwerk. Nach diesem Theoriehauptteil folgt ein kurzer Praxisteil, in dem zwei praktische Anwendungen von IPMI vorgestellt werden.

2 Theorie

Um IPMI praktisch einsetzen zu können, sollten zunächst die theoretischen Grundlagen und die Funktionsweise verstanden werden. Die folgenden Abschnitte geben einen Überblick darüber, welche Funktionen IPMI bietet und geben somit schon einen Ausblick auf Möglichkeiten des praktischen Einsatzes.

2.1 Grundlagen

Das „Intelligent Platform Management Interface“ ist eine Spezifikation von Funktionen und Protokollen zur Überwachung und Verwaltung von Rechnern. Über IPMI können die Sensoren eines Systems erkannt und deren Messwerte ausgelesen werden. Darüber hinaus bietet IPMI Funktionen im Bereich Power Control, also dem Ein-/Ausschalten und Neustarten von Systemen. IPMI wird hauptsächlich im Serverbereich eingesetzt, in dem meist kein direkter physischer Zugang zum Rechner besteht. Aus diesem Grund können IPMI Systeme auch aus der Ferne bedient werden.

Der Zugriff auf die IPMI Funktionalität erfolgt über spezielle Hardware, welche unabhängig von BIOS und Betriebssystem arbeitet. Die zentrale Komponente ist der „Baseboard Management Controller“ (BMC), welcher meistens in das Mainboard integriert ist. Einige Hersteller, z.B. Supermicro [1] und IBM [2], bieten auch Zusatzkarten zum Nachrüsten an. Diese Zusatzkarten sind meistens herstellerspezifisch, darum ist das Nachrüsten von IPMI nicht bei allen Systemen möglich. Der BMC ist ein kompletter „Mini-Rechner“ mit eigenem Prozessor, Speicher und Firmware.

2.1.1 Entstehung

Die Firmen Intel, HP, NEC und Dell haben 1998 die erste Version der IPMI Spezifikation veröffentlicht. Federführend für die Entwicklung ist Intel. Nach der Version 1.5 von 2001 wurde 2004 die aktuelle Version 2.0 veröffentlicht. In den Versionen 1.5 und 2.0 wurden vor allem die Netzwerkfunktionen wesentlich ausgebaut und verbessert.

Neuerungen der Version 1.5 sind Serial/LAN Messaging and Alerting. Das bedeutet, dass das Senden von Benachrichtigung über Systemereignisse per serieller Schnittstelle oder über das Netzwerk möglich ist. Das „Platform Event Filtering“ erlaubt einen besseren Umgang mit großen Mengen von Systemnachrichten, da bestimmte Kategorien einfach ausgefiltert werden können. Außerdem wurden neue Sensor- und Event-Typen hinzugefügt.

[1] `http://www.supermicro.com/products/accessories/addon/SIM.cfm` Letzter Zugriff: 22.09.2010

[2] `http://www-947.ibm.com/support/entry/portal/docdisplay?brand=5000008&lndocid=MIGR-50116` Letzter Zugriff: 22.09.2010

In der Version 2.0 wurde die Authentifizierung und Vertraulichkeit der Kommunikation mit dem BMC verbessert. Die umfangreichste Änderung ist die Erweiterung des Netzwerkprotokolls um Nutzdaten (Payloads), wodurch nicht nur IPMI Nachrichten, sondern beliebige Daten übertragen werden können. Diese Funktion erlaubte die Einführung von SOL welches in Abschnitt 2.4.3 genauer erläutert wird.

2.1.2 Funktionen

IPMI eignet sich nicht nur zur Überwachung von Systemen, sondern auch zur Wiederherstellung nach einem Fehlerfall oder zur Inventarisierung. Zusammenfassend bietet IPMI Funktionen aus den folgenden fünf Bereichen der Systemverwaltung: **Monitoring, Recovery, Logging, Alerting, Inventory**.

Das Monitoring umfasst die Überwachung von Hardwaresensoren, wie Temperatur, Spannung, Lüfterdrehzahlen und Einbrucherkennung. Der Bereich Recovery umfasst Funktionen zum Ein-/Ausschalten und Neustarten des Systems im Fehlerfall. Außerdem werden wichtige Systemereignisse von der Logging Komponente zur späteren Auswertung gespeichert. Das Alerting umfasst die automatische Benachrichtigung der Administratoren über wichtige Systemereignisse, z.B. Warnungen von Hardwaresensoren. Der letzte Bereich, das Inventory beinhaltet das Sammeln von Informationen über die vorhandene Hardware, z.B. Seriennummern der Geräte und Ersatzteilnummern (Field Replaceable Units / FRU).

2.2 Architektur

Nach dieser kurzen Übersicht über die Funktionen, gibt es im Folgenden einen Überblick, aus welchen Komponenten ein IPMI System aufgebaut ist und wie diese miteinander interagieren.

Zunächst lassen sich die Komponenten von IPMI in drei Kategorien einteilen: Das Herzstück sind die Management Controller, es wird dabei zwischen dem zentralen Baseboard Management Controller (*BMC*) und den Satellite Controllern unterschieden. Die Kommunikation innerhalb von und mit IPMI kann über die drei Schnittstellen lokales System Interface, serielle Schnittstelle oder LAN abgewickelt werden. Als dritten Bereich gibt es noch die Informationsspeicher Sensor Data Records (*SDR*), Field Replaceable Units (*FRU*) und das System Event Log (*SEL*).

2.2.1 Schema

Die Abblidung 2.1 zeigt den schematischen Aufbau eines IPMI Systems. Die zentrale Schnittstelle ist der Baseboard Management Controller, es handelt sich hierbei um einen „Mini-Rechner“ mit eigener CPU, Speicher und Firmware. Der BMC arbeitet unabhängig vom Zustand des Betriebssystems, der Rechner muss lediglich mit Strom versorgt

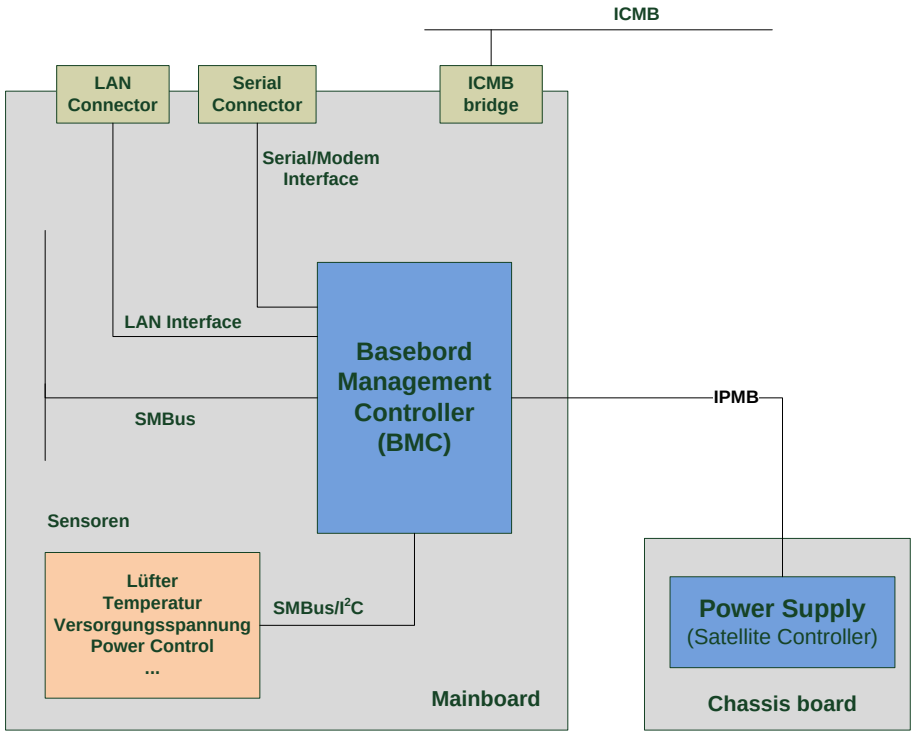

Abbildung 2.1: IPMI Schema (Eigene Zeichnung nach [IHND09])

werden damit der BMC einsatzfähig ist. Die Management Controller sind die Schnittstelle zwischen Anwender und den einzelnen Sensoren im System. Eine IPMI Implementierung benötigt nicht zwingend weitere Satellite Controller, kann diese aber zusätzlich zum BMC noch einsetzen. Diese könnten z.B. zur Überwachung und Kontrolle von Komponenten dienen, die nicht direkt auf dem Mainboard integriert sind. Abbildung 2.1 zeigt beispielsweise einen Satellite Controller im Netzteil des Rechners.

Neben den Management Controllern gibt es noch eine Vielzahl von Sensoren, z.B. für Lüfterdrehzahlen, Temperatur, Versorgungsspannung und Festplatten. Die Sensoren werden entweder per SMBus oder über I^2C abgefragt, auf beide Protokolle wird in Abschnitt 2.2.4 eingegangen. Die Management Controller kommunizieren untereinander über ein I^2C ähnliches Protokoll, den „Intelligent Platform Management Bus“ (*IPMB*). Da der IPMB für kurze Leitungen innerhalb eines Systems ausgelegt ist, sieht die IPMI Spezifikation noch den „Intelligent Chassis Management Bus“ vor. Dieser wird über eine ICMB Bridge angebunden und kann für die Anbindung von Komponenten in externen Gehäusen genutzt werden, welche keine volle IPMI Implementierung mit eigenem BMC besitzen.

Für die externe Kommunikation nutzt IPMI meistens die im System vorhandenen seriellen und Netzwerkschnittstellen mit. Es existieren aber auch IPMI Implementierungen, z.B. von Supermicro, in denen der BMC eine dedizierte LAN Schnittstelle besitzt.

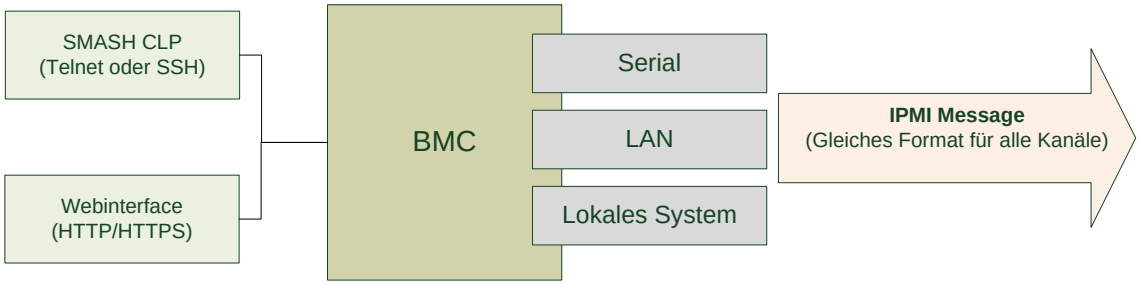

Abbildung 2.2: Kommunikationsschnittstellen des BMC

2.2.2 Nachrichten und Schnittstellen

Der BMC kann über verschiedene Schnittstellen Nachrichten verschicken, einige Möglichkeiten werden in Abbildung 2.2 dargestellt. In der IPMI Spezifikation sind nur die auf der rechten Seite gezeigten Schnittstellen zum Versenden von IPMI Nachrichten definiert. Eine IPMI Nachricht hat ein festes Format und ist für alle Übertragungsmedien (lokal, seriell oder LAN) identisch. Darüber hinaus haben viele Hersteller von IPMI Systemen noch weitere Optionen in ihre Produkte integriert, die eine Verbindung nicht nur über IPMI Nachrichten, sondern auch über andere Protokolle zulassen. Abbildung 2.2 zeigt auf der linken Seite zwei Beispiele dafür. Einerseits gibt es ein Webinterface welches über HTTP oder HTTPS erreichbar ist und Zugriff auf die Sensorwerte, die Power Control Funktionen und die Konfiguration des BMC erlaubt. Andererseits gibt es noch den SMASH (Systems Management Architecture for Server Hardware) CLP, welcher über Telnet oder SSH erreichbar ist. SMASH ist eine Kommandozeilenerweiterung zu IPMI, die direkt auf dem BMC ausgeführt wird und eine einheitliche Oberfläche in heterogenen Umgebungen bieten soll.

2.2.3 Lokales System Interface

Die IPMI Spezifikation (Seite 17 [IHND09]) sieht für den lokalen Zugriff auf den BMC vier verschiedene Schnittstellen vor: Keyboard Controller Style (KCS), System Management Interface Chip (SMIC), Block Transfer (BT) und SMBus System Interface (SSIF). Die KCS Schnittstelle basiert auf einem Intel 8742 Mikrocontroller und nutzt einen per-byte handshake. SMIC bietet eine Alternative zu KCS für Mikrocontroller die keine eingebaute Unterstützung für KCS besitzen. Bei SMIC handelt es sich um eine einfache drei PIN Schnittstelle, welche kostengünstig z.B. als FPGA umgesetzt werden kann. Die BT Schnittstelle bietet eine bessere Performance als KCS und SMIC. Das SSIF ist für einfache sehr kostengünstige BMC, die per SMBus an das System angeschlossen sind geeignet. Allerdings ist die Geschwindigkeit deutlich geringer als bei den anderen drei Schnittstellen.

Unter Linux/Unix wird der Zugriff über die oben genannten Schnittstellen über spe-

zielle Kernelmodule realisiert. Im Linux Umfeld wird hauptsächlich OpenIPMI [3] eingesetzt. Im BSD Bereich hat sich FreeIPMI [4] durchgesetzt.

2.2.4 Kommunikationsprotokolle

Die IPMI Spezifikation verwendet zur Kommunikation der MCs und Sensoren die Protokolle SMBus, I^2C und IPMB. Der Vorteil dieser Protokolle ist, dass sie einfach und kostengünstig zu implementieren sind. Darüber hinaus sind SMBus und I^2C in vielen Rechnern schon für IPMI ähnliche Aufgaben im Einsatz.

Der I^2C ist ein weit verbreiterter einfacher 2-Draht Bus. Er wurde von Phillips ursprünglich für die Verwendung in Fernsehgeräten entworfen. Die Bandbreite ist relativ gering, aber für das Übertragen von Sensorwerten u.ä. vollkommen ausreichend. Zum Einsatz kommt I^2C beispielsweise beim Auslesen von SPD EEPROMS, welche z.B. FRU Nummern enthalten, beim Übertragen von Monitor Einstellungen (DDC) sowie zum Ein-/Ausschalten des Netzteils. (Nach [NXP07]).

Der SMBus ist eine Untermenge des I^2C Buses und unter bestimmten Voraussetzungen auch mit diesem kompatibel. Die Ziele des SMBus sind Robustheit und Interoperabilität. Um diese zu erreichen, stellt er striktere Anforderungen an das Protokoll und die elektrischen Parameter als der I^2C Bus. Der SMBus fordert das Senden von ACK/NACK Antworten, d.h. im Gegensatz zum I^2C kann die Lebendigkeit von Bauteilen getestet werden, da diese Bestätigungen senden müssen. Bei I^2C können Bauteile bestimmte Anfragen ignorieren und müssen keine Antworten senden, wenn sie z.B. gerade mit Echtzeitaufgaben beschäftigt sind. (Nach [Wik10]).

Der „Intelligent Platform Management Bus“ ist ein I^2C basierter serieller Bus. Er dient zur Verbindung von MCs und BMC, der Abfrage von Sensoren und allgemein zur Kommunikation innerhalb eines Gehäuses.

Der „Intelligent Chassis Management Bus“ (ICMB) wird bei der Kommunikation zwischen verschiedenen Gehäusen eingesetzt, z.B. zur Abfrage von Sensoren in einem externen Bandlaufwerk.

2.3 Sensoren

Die IPMI Spezifikation sieht beim Zugriff auf die Sensoren eine zusätzliche Abstraktionsschicht vor. Das heißt, dass kein direkter Zugriff auf die Sensor Hardware erfolgt. Die im System vorhandenen Sensoren werden über die „Sensor Data Records“ (SDR) beschrieben. Der SDR ist einerseits ein Verzeichnis aller im System vorhandenen Sensoren und andererseits auch Informationsquelle für die exakte Beschreibung der einzelnen Sensoren. Um dem System neue Sensoren hinzuzufügen oder alte zu entfernen muss der SDR angepasst werden.

[3] http://openipmi.sourceforge.net/
[4] http://www.gnu.org/software/freeipmi/

Die IPMI Spezifikation definiert die zwei Typen von Sensoren, Discrete Sensors und Threshold Sensors. Discrete Sensors können 15 unterschiedliche Status annehmen. Die Digital Sensors sind keine Klasse für sich, sondern lediglich Discrete Sensors die nur zwei Zustände annehmen können. Threshold Sensors vergleichen ihre gemessenen Werte mit verschiedenen Schwellenwerten und geben sowohl den Messwert als auch einen Discrete Threshold Comparison Status zurück ([Fis10a]). In Abschnitt 2.3.2 werden diese Schwellen intensiver betrachtet.

Die Sensoren liefern ihre Werte nicht direkt in der gewünschten für den Menschen verständlichen Einheit, sondern einen Sensor RAW Wert. Dieser muss mit Hilfe einer Funktion umgerechnet werden. Sensoren bei denen diese Umrechnungsfunktion konstant ist, heißen *linear*. Sensoren bei denen die Umrechnungsfunktion vom Zeitpunkt der Messung abhängt, gehören zur Kategorie *non-linear*. Dazu gehören z.B. Temperatursensoren bei denen sich die elektrische Leitfähigkeit bei verschiedenen Temperaturen unterscheidet und die darum unterschiedliche RAW Werte liefern.

2.3.1 Sensortypen

Durch das Konzept der Sensor Data Records erlaubt die IPMI Spezifikation quasi beliebige Sensoren. Es gibt eine Reihe vordefinierter Sensoren, die allerdings durch die Hersteller um OEM Sensoren ergänzt werden können. Der Hersteller muss dann lediglich einen SDR mit entsprechenden Daten anlegen.

Die folgende Liste zeigt einige typische Sensoren:

- Versorgungsspannung (Netzteil)
- Netzteil (Existenz, Fehlfunktion, Redundanz)
- CPU (Temperatur, Spannung)
- Temperatur (Festplatte, Gehäuse)
- Lüfter (Anwesenheit, Umdrehung/Minute)
- Speicher (Versorgungsspannung)
- Festplatten (Existenz, Fehler)

Das für diese Arbeit benutzte Testsystem [5] mit einem Supermicro BMC bietet die in Tabelle 2.1 aufgelisteten Sensoren. Eine Nachfrage beim Support des Herstellers hat ergeben, dass das Netzteil keine Sensoren für die aktuelle Versorgungsspannung und die aufgenommene Leistung besitzt, wie sie beispielsweise ein Intel SR2500 [6] besitzt.

[5] Mainboard: Supermicro X8DTL-iF // R1.1

[6] `http://www.thomas-krenn.com/de/wiki/IPMI_Sensoren#Beispiel_2HE_Intel_Dual-CPU_SR2500_Server` Letzter Zugriff 24.09.2010

Lüfter	Spannung	Temperatur	Sonstige
FAN 1	CPU1 Vcore	CPU1 Temp	Chassis Intrusion
FAN 2	CPU2 Vcore	CPU2 Temp	PS Status
FAN 3	+1.5 V	System Temp	
FAN 4	+5 V		
FAN 5	+5VSB		
FAN 6	+12 V		
	-12 V		
	+3.3VCC		
	+3.3VSB		
	VBAT		

Tabelle 2.1: Sensoren des Testsystems mit Supermicro BMC

2.3.2 Sensor Grenzwerte: Beispiel Lüfter

Die Sensoren enthalten in ihrem SDR bestimmte Schwellenwerte bei deren Über- oder Unterschreitung Events generiert werden können. Abbildung 2.3 zeigt den fiktiven Verlauf eines Sensors für die Lüfterdrehzahl mit den möglichen Grenzen. Sowohl für die obere, als auch für die untere Schranke gibt es die Grenzen non-critical, critical und non-recoverable. Das erreichen der non-critical Grenze ist für den Betrieb noch nicht gefährlich, aber ein Anzeichen für anormales Verhalten des überwachten Bauteils. Bei Erreichen der critical Grenze befindet sich das Bauteil in einem gefährdeten Zustand und es sollten Gegenmaßnahmen eingeleitet werden. Bei Überschreiten der non-recoverable Grenze ist das Bauteil wahrscheinlich defekt und muss ausgetauscht werden (Seite 198 [IHND09]).

2.3.3 Was Sensoren noch können...

Über das einfache Ausgeben von Messwerten und das Erzeugen von Benachrichtigungen haben Sensoren noch weitere praktische Eigenschaften. So können mit „Entity Association Records“ einzelne Sensoren zu Gruppen zusammengefasst werden. Das eignet sich besonders für redundante Netzteile, welche zu einer „Power Unit“ zusammengefasst werden können. In diesem Fall kann der BMC nicht nur ein „Power failure“, sondern auch ein „Redundancy lost“ Event erzeugen. Der Administrator weiß dann, dass sein Server noch in Betrieb ist, er aber zeitnah ein Netzteil austauschen muss.

Außerdem können Sensoren noch mit „Field Replaceable Unit“ (FRU) Nummern verbunden werden. Die FRU ist eine Ersatzteilnummer. Im Falle eines durch einen Sensor gemeldeten defektes kann das passende Ersatzteil direkt beim Hersteller bestellt werden.

2.4 Fernzugriff

Die IPMI Spezifikation sieht neben den lokalen Schnittstellen auch den Fernzugriff über das Netzwerk vor. Über die Netzwerkschnittstellen können nicht nur Sensoren ausge-

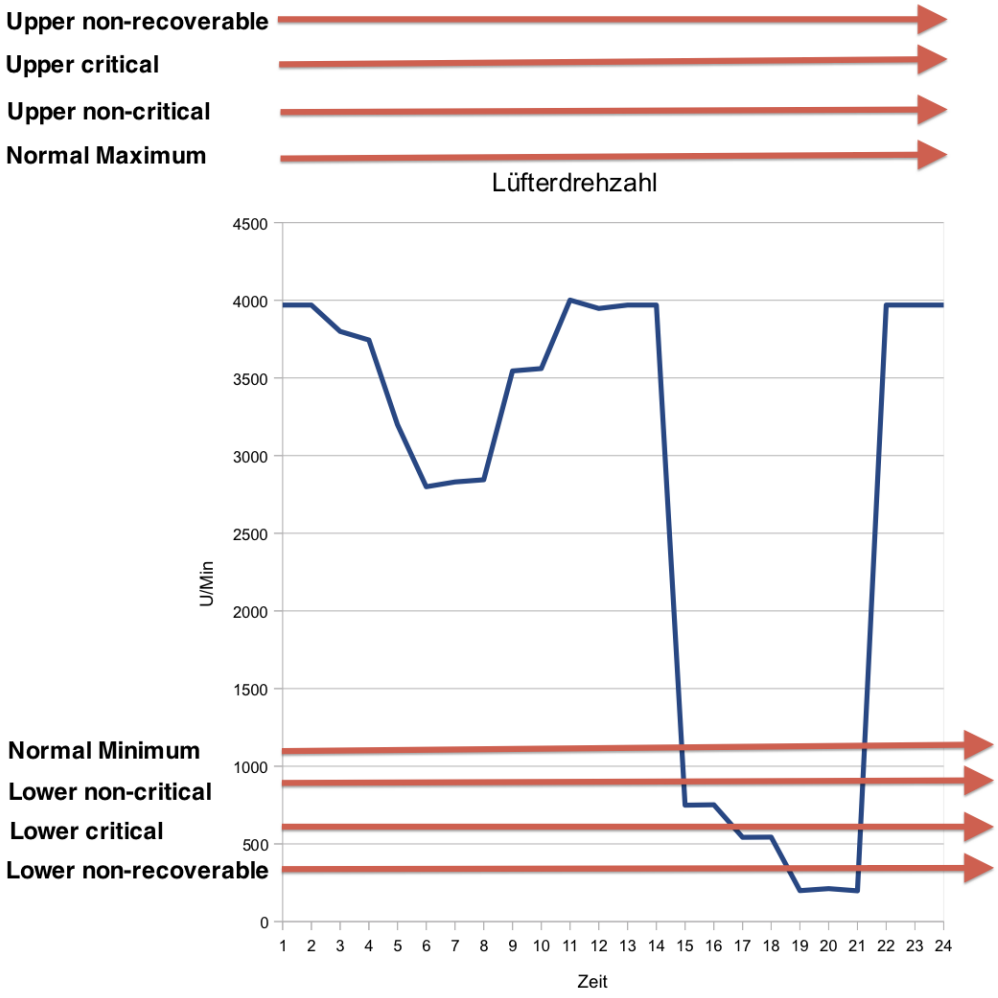

Abbildung 2.3: Fiktiver Werteverlauf einer Lüfterdrehzahl.

lesen und Warnungen versendet werden. Das System kann darüber auch neu gestartet werden und ein Zugriff auf die serielle Schnittstelle ist möglich (siehe SOL in Abschnitt 2.4.3).

2.4.1 Power Control (Betriebssystem unabhängig)

Das IPMI Powermanagement erlaubt die in Tabelle 2.2 aufgelisteten Operationen. Die mit einem Stern markierten Operationen sind in der Spezifikation als optional gekennzeichnet. Mit Power Up kann das System ganz normal gestartet werden, so als würde man den physikalischen Einschalter am Server drücken. Mit Power Down fährt der Server in den S4 (Hibernate) oder S5 (Soft-Off-Modus) herunter. Zu beachten ist, dass dies nicht unbedingt ein kontrolliertes Herunterfahren des Betriebsystems impliziert. Die Power Cycle Operation startet den Rechner nur neu, wenn er bereits eingeschaltet ist. Im ausgeschalteten Zustand hat der Befehl keinen Effekt. Wenn das Betriebsystem des Rechners eingefroren ist und er nicht mehr reagiert, kann er mit einem Hard Reset neugestartet werden. Nach Ausführen eines Soft Shutdown wird das System durch die Simulation eines ACPI „fatal overtemperature“ Events heruntergefahren.

Power Up	Einschalten
Power Down	soft off S4/S5 power state
Power Cycle*	Neustarten wenn Rechner läuft
Hard Reset	Neustarten in jedem Fall
Soft Shutdown*	Herunterfahren über ACPI

Tabelle 2.2: IPMI Power Control Operationen (*: optional)

2.4.2 Netzwerkzugriff

In der Version 1.5 wurde IPMI um Netzwerk Messaging und Alerting erweitert. Das heisst, IPMI Nachrichten und Alarme können nicht nur lokal sondern auch über das Netzwerk versendet werden. Für das Versenden von Nachrichten wird das „Remote Management Control Protocol“ (RMCP) eingesetzt. Es basiert auf UDP und erlaubt das Übertragen von IPMI Nachrichten über das Netzwerk. Mit Version 2.0 der Spezifikation wurde RMCP+ eingeführt, welches Verbesserungen in den Bereichen Authentifizierung und Vertraulichkeit der Nachrichten bietet. Erst mit RMCP+ ist das Verschlüsseln des IPMI Datenverkehrs möglich. Außerdem erlaubt RMCP+ den Transport von Nutzdaten innerhalb von IPMI Nachrichten. Dadurch ist es möglich Funktionen wie Serial over LAN und KVM über IPMI zu realisieren, welche mit Version 1.5 noch nicht oder nur mit proprietären Herstellererweiterungen möglich waren.

IPv6

Die aktuelle Version 4 des Internet Protokolls wird in absehbarer Zeit durch die neue Version 6 abgelöst werden. Gründe hierfür sind der beschränkte Adressbereich und die Größe der Routing Tabellen, welche der immer weiter wachsenden Zahl an Geräten nicht mehr gewachsen sind.

In der IPMI Spezifikation ([IHND09]) findet IPv6 noch keine Erwähnung, da die IPMI Kommunikation auf UDP aufsetzt ist eine Migration auf IPv6 theoretisch kein Problem. Das für diese Arbeit verwendete Testsystem unterstütz schon den Einsatz von IPv6. Wie dem Quelltext zu entnehmen ist, unterstütz die aktuelle Version 2.0.18 von OpenIPMI IPv6. Die auf dem Testsystem installierte Version konnte allerdings keine IPv6 Verbindung zum BMC herstellen.

2.4.3 Serial over LAN

„Serial over LAN“ erlaubt den Zugriff auf die serielle Schnittstelle des Rechners über den BMC, also unabhängig vom Betriebssystem. Wenn das Betriebsystem nicht mehr über das Netzwerk erreichbar ist, z.B. wegen falscher Netzwerkeinstellungen oder einer zu restriktiven Firewall, ist dennoch ein Fernzugriff über SOL möglich. Der Administrator hat mit SOL also noch eine Fallback Lösung für die Verwaltung seiner Server. Um im Fehlerfall über SOL Zugriff auf den Server zu bekommen, sind einige Vorarbeiten am BIOS und Betriebsystem nötig, damit diese auch über eine serielle Schnittstelle ansprechbar sind. Im Folgenden soll dies am Beispiel von Linux gezeigt werden.

Um einen Rechner vollständig per SOL verwalten zu können sind Anpassungen in drei Bereichen notwendig: Es müssen BIOS, Bootloader und Betriebsystem angepasst werden. Das BIOS eines IPMI fähigen Systems bietet meistens eine Menüoption mit der das BIOS zusätzlich zum Bildschirm auch auf einer seriellen Schnittstelle ausgegeben wird. Dabei muss darauf geachtet werden, dass nicht die physikalische serielle Schnittstelle ausgewählt wird. Stattdessen muss die mit SOL markierte virtuelle Schnittstelle benutzt werden, da nur diese von IPMI auf das Netzwerk umgeleitet wird.

Anschliessend muss der Bootloader, in diesem Fall Grub, wie in Listing 2.1 angepasst werden. Die erste Zeile konfiguriert die Parameter der seriellen Schnittstelle. Dabei ist zu beachten, dass Grub die Schnittstelle mit Eins beginnend nummeriert, wogegen Linux mit Null beginend nummeriert. Die zweite Zeile sorgt dafür, dass der Bootloader im normalen Terminal und der seriellen Schnittstelle ausgegeben wird. Danach wird mit der dritten Zeile dafür gesorgt, dass die Boot Meldungen des Linux Kernels ebenfalls auf der seriellen Schnittstelle ausgegeben werden.

```
1  serial --unit=1 ---speed=19200 --word=8 --parity=no --stop
2  terminal --timeout=5 serial console
3  kernel /boot/vmlinuz [...] console=tty0 console=ttyS1,19200n8r
```

Listing 2.1: Bootloader

Nachdem das Betriebsystem gebootet ist, muss der Administrator sich noch anmelden können, weshalb die /etc/inittab um die Zeile aus Listing 2.2 ergänzt werden muss. Dadurch wird eine Login Shell auf der seriellen Schnittstelle gestartet.

```
1 s0:2345:respawn:/sbin/agetty 19200 ttyS0 vt100-nav
```

Listing 2.2: Linux Konsole auf serieller Schnittstelle

Damit ist das Betriebsystem für die Benutzung von SOL vorbereitet, anschließend kann mit OpenIPMI der Zugriff getestet werden (siehe Listing 2.3).

```
1 ipmitool -I lanplus -H intel-ipmi  -U admin sol activate
```

Listing 2.3: ipmitool

Allgemein ist zu beachten, dass sowohl für den BMC, das BIOS, Grub und Linux die selben Parameter und die selbe Schnittstellennummer, unter Berücksichtigung der unterschiedlichen Nummerierung, genutzt wird. Die Nummer der per SOL umgeleiteten seriellen Schnittstelle kann dem Handbuch des Servers oder der Einstellung im BIOS entnommen werden.

3 Praxis

Der folgende Abschnitt zeigt zwei Beispiele wie IPMI praktisch eingesetzt werden kann. Es gibt die Möglichkeit IPMI fähige Geräte mit Hilfe von Nagios zu überwachen. Außerdem wird ein Paper vorgestellt, welches den Einsatz von IPMI im HPC Bereich zeigt.

3.1 Nagios IPMI Plugin

Nagios ist ein umfassendes Netzwerküberwachungs Tool, welches durch Plugins einfach erweiterbar ist. Nagios kennt zwei verschiedene Arten von Checks. Die Host-Checks prüfen, ob ein Gerät über das Netzwerk erreichbar ist. Mit Hilfe der Service-Checks können dann die Dienste des jeweiligen zu überwachenden Gerätes auf ihre Funktionsfähigkeit geprüft werden. Aufgrund der großen Verbreitung von Nagios existieren eine Vielzahl von Plugins für viele verschiedenen Server Dienste.

Das Nagios IPMI Plugin (check_ipmi_sensor) wird aktiv von Herrn Werner Fischer (Thomas-Krenn AG) betreut [Fis10b]. Durch das Plugin können die IPMI Sensoren des Systems überwacht und mit NagiosGrapher [1] im zeitlichen Verlauf dargestellt werden.

Das Plugin basiert auf OpenIPMI und kann entweder lokal oder über das Netzwerk mit dem BMC kommunizieren. Das „ipmitool“ des OpenIPMI Projekts enthält in der aktuellen Version 2.0.18 noch einen Fehler bei der Verarbeitung des Status von diskreten Sensoren. Wie auf der Mailingliste des Nagios Plugins [2] diskutiert wurde, zeigt das „ipmitool“ den Status OK obwohl der Sensor einen Fehler meldet. Da das Nagios Plugin nur die Statusspalte auswertet, wird keine Warnung gemeldet, obwohl ein Fehler vorliegt. Das im BSD Umfeld verbreitete FreeIPMI Projekt hat dieses Problem nicht. Dort kann das Programm „ipmimonitoring“ umfassender konfiguriert werden, so dass auch Fehler von diskreten Sensoren korrekt verarbeitet werden. Aus diesem Grund wird eine zweite Version des Nagios Plugins entwickelt, die auf FreeIPMI basiert.

Für die Überwachung kann check_ipmi_sensor entweder alle oder nur bestimmte Sensortypen abfragen. Dadurch kann der Administrator die Informationsflut gut bewältigen. Außerdem lassen sich mehrere Service-Checks für verschiedene Sensortypen anlegen. Das hat den Vorteil, dass aus den Nagios Warnungen direkt hervorgeht bei welchen Sensoren ein Problem aufgetreten ist.

[1] `http://www.netways.de/de/produkte/nagios_addons/nagiosgrapher/` Letzer Zugriff: 24.09.2010

[2] `http://www.mail-archive.com/ipmitool-devel@lists.sourceforge.net/msg01472.html` Letzter Zugriff: 22.09.2010

3.2 Proactive Fault tolerance for HPC

In ihrem Paper [NMES07] stellen die Autoren ein System vor in dem IPMI nicht zur Behandlung von bereits aufgetretenen Fehlern benutzt wird, sondern dazu präventive Massnahmen zur Fehlervermeidung einzuleiten.

Große Parallelrechnerumgebungen setzen auf eine Vielzahl an Rechenknoten. Das im Paper vorgestellte System basiert auf einer mit XEN virtualisierten Rechnerumgebung. Im Gegensatz zu den meisten Systemen wird kein reaktiver Ansatz genutzt, bei dem defekte Knoten nach einem Ausfall getauscht werden. Sondern ein proaktiver, bei dem der „Gesundheitszustand“ der Knoten durch IPMI überwacht wird. Das Monitoringsystem versucht dann an Hand von bestimmten Sensorwerten den Zustand der Rechner zu bewerten. Wenn ein Knoten mit hoher Wahrscheinlichkeit ausfallen wird, werden die virtuellen Maschinen auf andere gesunde Knoten migriert. Der potentiell defekte Knoten kann dann untersucht werden und anschließend wieder in Betrieb gehen. Dadurch werden „Downtimes“ durch abgestürzte Knoten bestmöglich vermieden.

4 Fazit

Die vorliegende Arbeit hat zunächst einen Einblick in den Aufbau von IPMI gegeben. Es wurde der BMC als zentrale Komponente vorgestellt. Außerdem wurden die Nachrichtenkanäle und Kommunikationsprotokolle eines IPMI Systems vorgestellt. Nach der Betrachtung der verschiedenen Sensortypen und deren Verwendung wurde der Bereich des Fernzugriffs untersucht. Hierbei lag der Fokus auf der SOL Funktionalität.

Zusammenfassend kann festgestellt werden, dass IPMI umfassende Funktionen im Bereich der Systemverwaltung. Es eignet sich für die Bereiche Monitoring, Recovery, Logging, Alerting und Inventory bietet. Ein großer Vorteil ist, dass es herstellerübergreifend verfügbar und betriebssystemunabhängig ist, was es gerade für heterogene Umgebungen interessant macht.

Ein Nachteil ist, dass es zusätzliche Hardware benötigt und ältere Server nicht unbedingt aufgerüstet werden können.

Literaturverzeichnis

[Fis10a] FISCHER, Werner: Alles über IPMI. In: *Linux Technical Review* (2010)

[Fis10b] FISCHER, Werner: *Nagios IPMI Plugin*. http://www.thomas-krenn.com/de/wiki/IPMI_Sensor_Monitoring_Plugin, 2010

[IHND09] INTEL ; HEWLETT-PACKARD ; NEC ; DELL: *Intelligent Platform Management Interface Specification Second Generation*. 06 2009

[NMES07] NAGARAJAN, Arun B. ; MUELLER, Frank ; ENGELMANN, Christian ; SCOTT, Stephen L.: *Proactive Fault Tolerance for HPC with Xen Virtualization*. ISC 2007, 06 2007

[NXP07] NXP: *I2C-bus specification and user manual (Rev. 03)*. 06 2007

[Wik10] WIKIPEDIA: *System Management Bus*. 06 2010