INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Entwicklung von Adaptern zum Management von Hostvirtualisierungslösungen

Michael Hauser

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Entwicklung von Adaptern zum Management von Hostvirtualisierungslösungen

Michael Hauser

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller Prof. Dr. Heinz-Gerd Hegering (em.)
Betreuer:	Dr. Nils gentschen Felde Tobias Lindinger
Abgabetermin:	30. Juli 2009

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. Juli 2009

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Virtualisierung ermöglicht das Etablieren mehrerer virtueller Systeme auf einem einzigen physisch vorhandenen System. Mit dem zunehmenden Einsatz von Virtualisierung in Industrie und Forschung und der Vielzahl der zur Verfügung stehenden Virtualisierungslösungen muss auch ein effizientes Management virtueller IT Infrastrukturen möglich sein. Dazu liefert jeder Hersteller eigene Managementanwendungen für sein jeweiliges Produkt. Allerdings gibt es bis jetzt noch keine integrierte Plattform, mit der sich die Lösungen unterschiedlicher Anbieter von einem zentralen Punkt aus managen lassen. In einer Diplomarbeit wurde ein umfangreiches Konzept einer erweiterbaren Managementplattform für unterschiedliche Virtualisierungslösungen entworfen und in den Grundfunktionalitäten implementiert. Grundlage für die Kommunikation der Managementplattform mit den Virtualisierern bilden Adapter. Ziel dieser Arbeit ist es, einen Adapter zu entwickeln, der die Kommunikation zwischen der Managementplattform und einem VMware ESX Server ermöglicht. Dazu wird zunächst eine Einführung in die Managementarchitektur des VMware ESX Servers gegeben. Danach wird, aufbauend auf der prototypischen Implementierung aus der Diplomarbeit ein Adapter entworfen, der das Monitoring von Prozessor- und Arbeitsspeicherauslastung eines VMware ESX Servers ermöglicht. Hauptaufgaben bei der Realisierung des Adapters sind das Erzeugen der Nachrichten für die Managementplattform, die Erweiterung der Funktionalität der Managementplattform, die Anpassung der Nutzerschnittstelle und das Abrufen und Konvertieren der benötigten Informationen vom VMware ESX Server.

Inhaltsverzeichnis

1. Einleitung

Die Leistungsfähigkeit moderner Hardware und IT Infrastrukturen (Netze etc.) ist heute enorm. Noch vor wenigen Jahren waren Anwendungen, die heutzutage auf jedem Notebook ohne Probleme laufen nur auf Hochleistungsrechnern in Rechenzentren verfügbar. Durch diese hohe Leistungsfähigkeit ergeben sich aber auch Leistungsüberschüsse. Für den privaten Anwender mag das schön sein, für Unternehmen ist es schlichtweg Geldverschwendung, denn die teuer bezahlten Ressourcen werden nicht genutzt. Virtualisierung setzt genau an diesem Punkt an: auf einem physischen System werden mehrere virtuelle Systeme ausgeführt. So ist es möglich die Ressourcen eines Systems auf mehrere Instanzen von Betriebssystemen aufzuteilen. Insbesondere für Unternehmen ist somit die Virtualisierung ein effektiver Beitrag zur besseren Nutzung der vorhandenen Ressourcen. Auf dem Markt gibt es mittlerweile viele kommerzielle und auch kostenlos verfügbare Virtualisierungssysteme wie z.B. VMware ESX oder Xen Hypervisor.

1.1. Motivation und Ziel

Mit dem zunehmenden Einsatz solcher Technologien und der Vielzahl der zur Verfügung stehenden Virtualisierungslösungen muss auch ein effizientes Management virtueller IT Infrastrukturen möglich sein. Jeder Anbieter bietet zu seinem Produkt eine adäquate Lösung, jedoch gibt es bis jetzt noch keine Plattform, mit der sich die Lösungen unterschiedlicher Anbieter von einem zentralen Punkt aus managen lassen. In der Diplomarbeit von Florian Bittner an der LMU [Bit08] wurde ein umfangreiches Konzept einer erweiterbaren Managementplattform für unterschiedliche Virtualisierungslösungen entworfen und in den Grundfunktionalitäten implementiert. In der vorliegenden Arbeit wird, aufbauend auf der prototypischen Implementierung aus der Diplomarbeit, ein Adapter entwickelt, der die Anbindung an einen VMware ESX Server ermöglicht.

1.2. Aufbau der Arbeit

In Kapitel 2 werden zunächst einige Grundlagen behandelt: Es wird ein kurzer Überblick über Virtualisierung, den ESX Server des Herstellers VMware und die Managementplattform gegeben. Danach wird im Kapitel 3 das Konzept des Adapters dargestellt. Im Kapitel 4 werden die Implementierung und die Umsetzung des Adapters dokumentiert. Im letzten Kapitel werden die erreichten Ziele erläutert.

2. Grundlagen

In diesem Kapitel werden die Grundlagen der vorliegenden Arbeit dargestellt. Es wird zunächst eine kurze Einführung zum Thema Virtualisierung gegeben, danach wird die Managementplattform dargestellt und ein Überblick des zu managenden VMware ESX Servers gegeben.

2.1. Virtualisierung

Im wesentlichen wird bei der Virtualisierung eine Ressource von der darunterliegenden physischen Grundlage entkoppelt: So kann z.B. mit virtuellem Speicher einer Anwendung mehr Speicherplatz zur Verfügung gestellt werden, als in dem System wirklich installiert ist, indem dazu Auslagerungsdateien auf die Festplatten verschoben werden. Diese Technik ist auf sämtliche Arten von Hard- und Software übertragbar, wie Netze, Hintergrundspeicher, Prozessoren, Betriebssysteme, Anwendungen und andere Ressourcen: solche Virtualisierungstechniken legen eine Abstraktionsschicht zwischen die eigentlichen Ressourcen und die Anwendungen, die darauf zugreifen. Diese Virtualisierungstechniken werden vom Endbenutzer

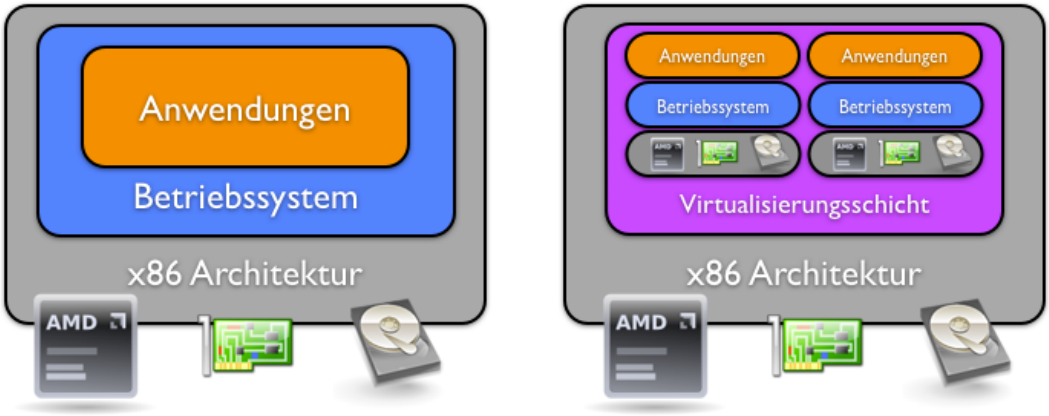

Abbildung 2.1.: Infrastrukturwandel durch Virtualisierung

nicht wahrgenommen und können so einfach in eine Hardwareumgebung eingebaut werden. Dadurch wandelt man seine Infrastruktur von einem starren Gebilde in ein flexibles Werkzeug, das die Ressourcen optimal verteilt und bestmöglich ausnutzt. Die Abbildung 2.1 zeigt den Unterschied in der Infrastruktur vor und nach dem Einsatz der Virtualisierung: Vor dem Einsatz der Virtualisierung sind die Ressourcen fest zugeteilt und eng aneinander gekoppelt, so läuft pro Rechner nur ein Betriebssystem und die ungenutzten Ressourcen liegen brach. Anders verhält es sich nach Einführung der Virtualisierung: Die Anwendungen und

Betriebssysteme werden von der darunterliegenden Hardware abgekoppelt und sind auf die verschiedenen Gastsysteme aufteilbar. Weiterführende Informationen und eine Einführung in die Grundlagen der Virtualisierung finden sich in [VMw06].

2.2. Managementplattform

In diesem Abschnitt soll ein kurzer Überblick der Managementplattform [Bit08] gegeben werden. Die Grundidee der Managementplattform besteht darin, unterschiedliche Virtualisierer von einer zentralen Stelle aus zu steuern und zu überwachen. An die Stelle unterschiedlicher Tools tritt die Managementplattform, die die Funktionalitäten der einzelnen Werkzeuge zusammenfasst.

2.2.1. Überblick

In der Abbildung 2.2 ist ein abstraktes Managementszenario dargestellt: Das Szenario besteht aus unterschiedlichen Virtualisierern. Sie werden jeweils durch eine eigene Management-Anwendung gesteuert und überwacht. Somit muss ein Manager drei unterschiedliche Systeme gleichzeitig über drei unterschiedliche Managementlösungen betreuen. Dies kann in einer pro-

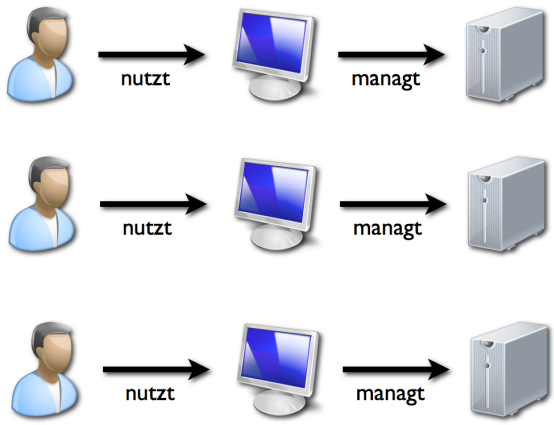

Abbildung 2.2.: Managementsituation vorher(vgl. [Bit08] S. 5)

totypischen Umgebung durchaus noch eine akzeptable und praktikable Lösung sein. Sobald man sich aber groß dimensionierte Lösungen im Produktiveinsatz bei Unternehmen vorstellt, wo es wichtig ist einen umfassenden Überblick über den Zustand aller Systeme zu haben, ist dies keine praktikable Lösung mehr. So muss man hier auf mehrere Werkzeuge zurückgreifen, die sich auf unterschiedlichen Rechnern befinden oder auch räumlich getrennt sind. Hier setzt die Managementplattform [Bit08] an: Sie bietet die Überwachung und Steuerung unterschiedlicher Systeme über einen zentralen Zugangspunkt. Nach dem Einsatz der Managementplattform [Bit08] bietet sich ein Managementszenario wie es in der Abbildung 2.3 dargestellt ist:

Wo bislang mehrere Werkzeuge auf unterschiedlichen Systemen zum Betreuen mehrerer Virtualisierungslösungen benötigt wurden, gibt es nun nur noch einen zentralen Zugangspunkt zum Management der unterschiedlichen Systeme. Die Steuerung der Virtualisierungshosts und der virtuellen Maschinen erfolgt mit Hilfe einer Weboberfläche in einem Browser.

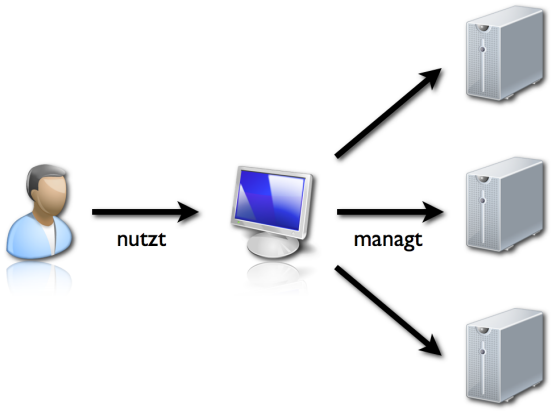

Abbildung 2.3.: Managementsituation nachher (vgl. [Bit08] S. 6)

Über diese Nutzerschnittstelle kann der Nutzer mit der Plattform und den angebundenen Systemen kommunizieren: Es werden Nutzereingaben entgegengenommen und an die Plattform weitergegeben. Genauso übernimmt die Weboberfläche die Darstellung der Statusinformationen aller zu managenden Systeme.

Die Managementplattform ist in [Bit08] vollständig modelliert worden. Die prototypische Implementierung der Plattform aus [Bit08] realisiert nur einen Teil der modellierten Funktionen. Von der Plattform werden aktuell nur das Hinzufügen und Überwachen von Ressourcen unterstützt. Einen Überblick über die funktionalen Anforderungen der Plattform und deren Umsetzung gibt die Tabelle 2.1.

Anforderung	**Prototyp**
Hinzufügen/Entfernen von Ressourcen	✓
Konfigurieren der Ressourcen	—
Monitoring der Ressourcen	✓
Dienstgüteüberwachung der Ressourcen	—
Zustandsüberwachung der Ressourcen	—
VM-Migration	—
VM-Steuerung	—
VM-Vorlagen	—
Klonen von VMs	—
Benutzer-/Gruppenverwaltung	—
Benachrichtigungen	—
Autodiscovery	—

Tabelle 2.1.: Funktionale Anforderungen und Prototyp (vgl. [Bit08] S.78)

2.2.2. Architektur

Die Plattform besteht aus verschiedenen Bausteinen. Sie stellen die Basisdienste für die Plattform zur Verfügung. Im Folgenden wird ein kurzer Überblick über die Architektur der Plattform und die Funktion der Bausteine gegeben. Abbildung 2.4 zeigt die Architektur der Plattform nach [Bit08].

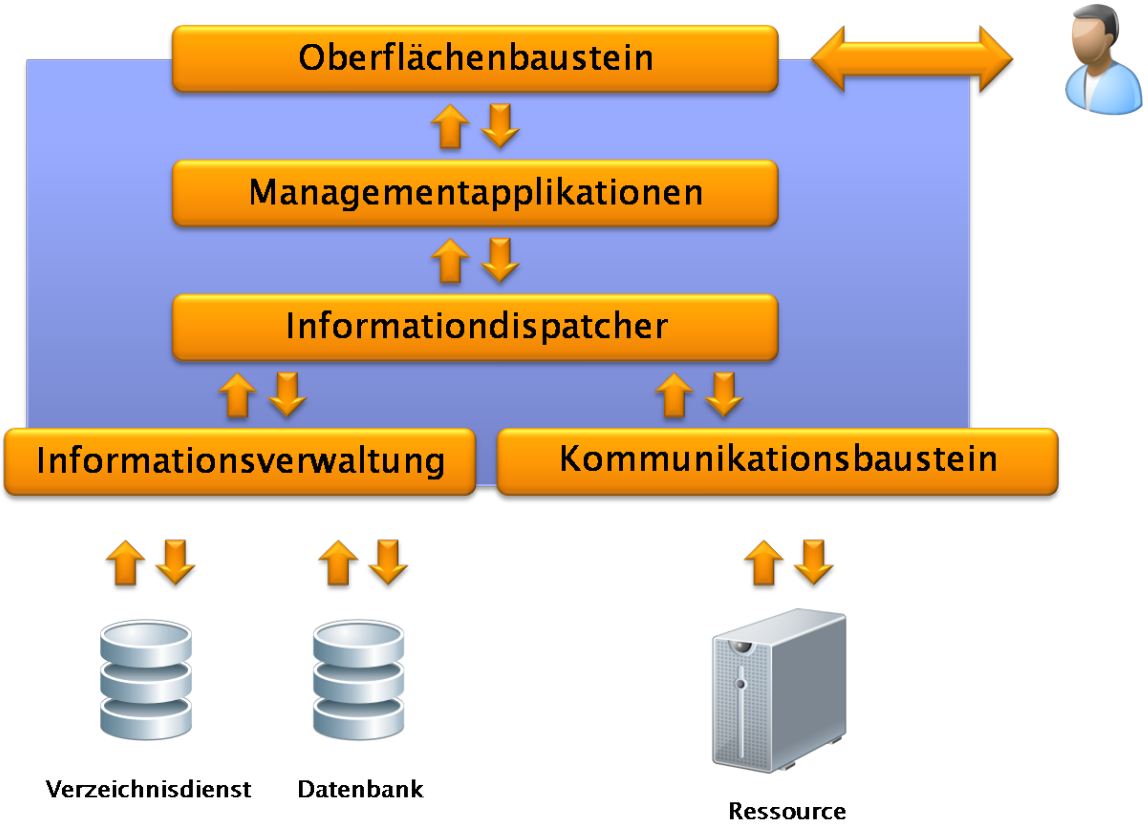

Abbildung 2.4.: Bausteine der Managementplattform

Der **Oberflächenbaustein** bietet Zugriff auf das System von außen. Hier werden alle relevanten Informationen dargestellt, die nötig sind um das System und die angebundenen Virtualisierer zu managen. Hier werden auch die Eingaben des Anwenders entgegengenommen und über die Plattform zu den entsprechenden Ressourcen geleitet. Um bei der Gestaltung der Oberfläche flexibel zu bleiben ist keine bestimmte Zugriffstechnik in der Plattform definiert.

Managementapplikationen: Sie stellen die eigentlichen Funktionalitäten der Managementplattform dar. Hier werden die gesammelten Informationen verarbeitet. Die wichtigsten Funktionalitäten dieses Bausteins sind im Wesentlichen das Überwachen der Leistung und die Konfiguration der angebundenen Ressourcen.

Der **Informationdispatcher** ist das Kernsystem. Er bildet die Schaltzentrale zwischen den Bausteinen Managementapplikationen, Kommunikationsbaustein und Informationsverwaltung. Hier werden Managementdaten empfangen und an die entsprechenden Bausteine weitergeleitet.

Ein weiterer Baustein ist die **Informationsverwaltung**. In diesem Teil der Plattform werden im Wesentlichen Managementdaten gespeichert. Über einen Adapter können verschiedene Systeme angebunden werden, etwa eine Datenbank zur Speicherung und Verwaltung der Managementdaten oder auch ein Verzeichnisdienst zur Authentifizierung der Nutzer.

Der **Kommunikationsbaustein** ist die Schnittstelle zwischen Plattform und Ressourcen oder auch anderen Managementplattformen. Über unterschiedliche Ressourcenadapter im

Kommunikationsbaustein können die einzelnen Ressourcen oder eine andere Managementplattform angebunden werden. In der folgenden Abbildung 2.5 ist der Kommunikationsbaustein zusammen mit den zu entwickelnden Ressourcenadaptern dargestellt. Der Adapter zum VMware ESX Server ist die Aufgabenstellung dieser Arbeit, ein Adapter zur Steuerung von XEN über die Virtualisierungs-API libvirt ist Fokus der Arbeit von [Mic09].

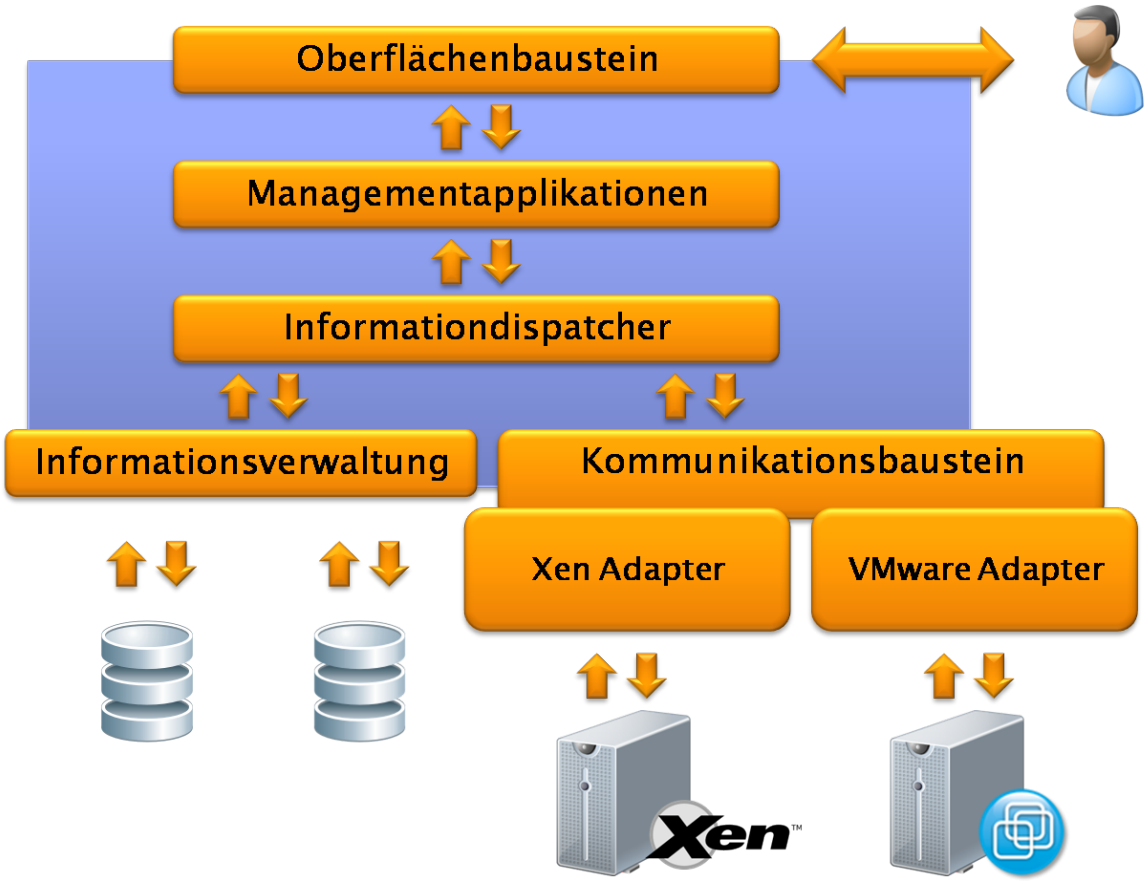

Abbildung 2.5.: Kommunikationsbaustein mit Ressourcenadaptern

2.2.3. Kommunikation

Die Kommunikation innerhalb der Plattform erfolgt über Nachrichten, die zwischen den Bausteinen gesendet werden. Wenn ein Ressourcenadapter eine Nachricht von der Plattform erhält, die aktuelle Performancewerte (z.B. Speicherauslastung des Hosts) anfordert, so wird diese Nachricht im Adapter angenommen und verarbeitet. Dann werden die neuen Informationen von der Ressource angefordert. Hat der Adapter die Informationen ermittelt, so verpackt er sie wieder in eine Nachricht und sendet diese an die Plattform zurück. Der Informationsdispatcher sorgt dann für die Weiterleitung der Nachricht an die zuständigen Stellen innerhalb der Managementplattform. Die Abbildung 2.6 zeigt den Ablauf der Kommunikation zwischen den Bausteinen.

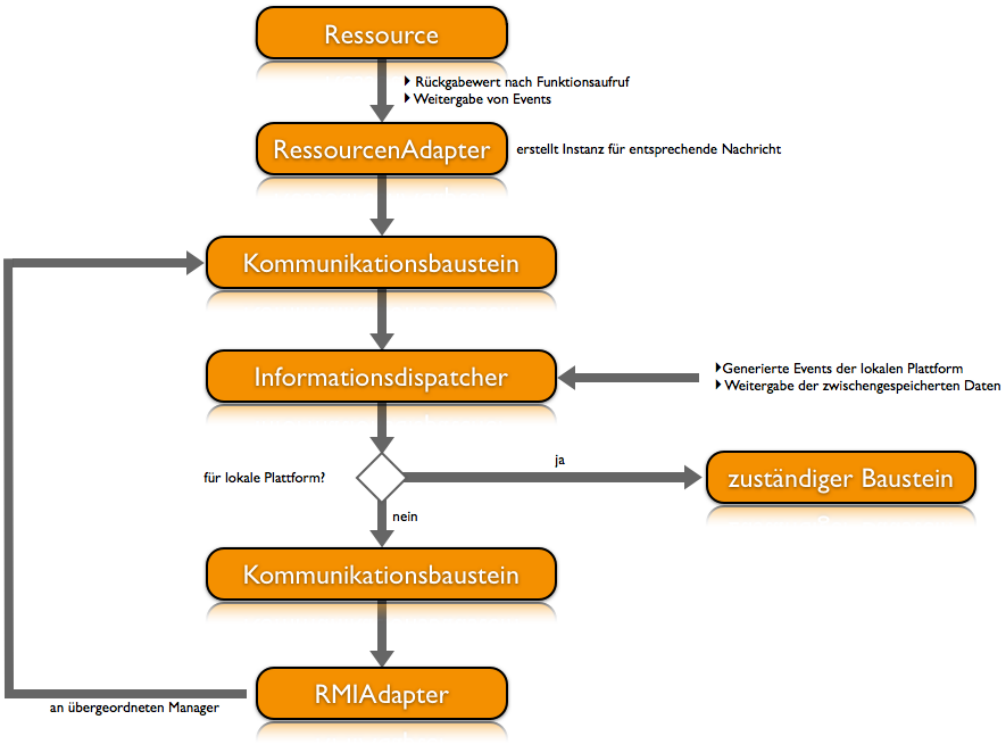

Abbildung 2.6.: Kommunikationsablauf in der Plattform (vgl. [Bit08] S. 56)

2.2.4. Aufbau des Projektnetzes

Die Plattform wurde in einem Projektnetz des Munich Network Management Teams (MNM-Team) am Institut für Informatik (IFI) an der Ludwig-Maximilian-Universität München (LMU) installiert. In diesem befindet sich ein Managementhost (projekt3), auf dem die Plattform installiert ist. Außerdem dient der Rechner als Router ins Institutsnetz. Als Virtualisierungshosts sind vier Systeme im Projektnetz installiert: Zwei XEN Systeme, ein VMware ESX Server und ein OpenVZ. Die Topologie des Netzes ist in Abbildung 2.7 dargestellt.

2.3. Virtualisierer

Ziel dieses Fortgeschrittenenpraktikums ist die Kommunikation der Managementplattform mit einem VMware ESX Server zu ermöglichen. VMware stellt verschiedene Programmierschnittstellen zum Management seiner Produkte zur Verfügung. In diesem Abschnitt wird zunächst eine kurze Einführung in den VMware ESX Server gegeben. Danach werden die Managementmöglichkeiten für den VMware ESX Server dargestellt und die Auswahl für eine Managementlösung anhand von Anforderungen definiert.

2.3.1. Einführung in VMware ESX

VMware ESX ist eine Virtualisierungsebene, die Prozessor, Speicher und Netzwerkressourcen in mehrere virtuelle Maschinen aufteilen kann. VMware ESX hat sich in Produktivumgebun-

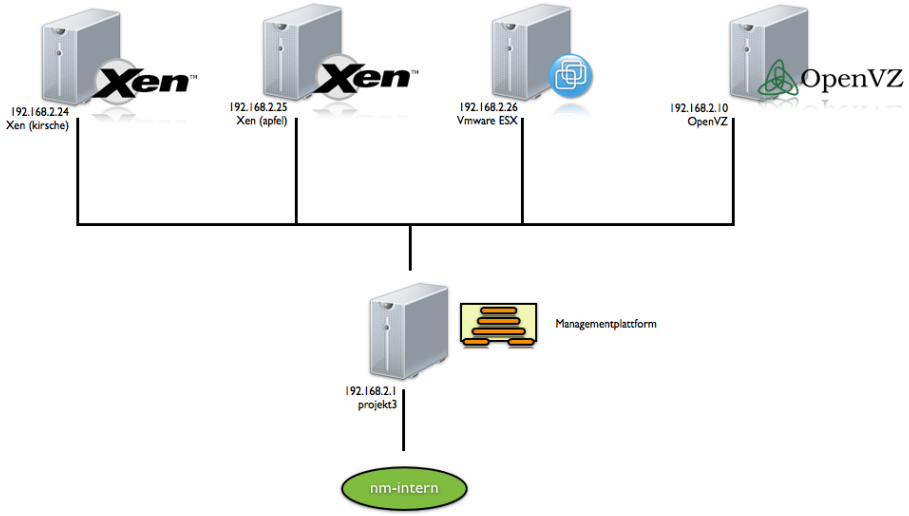

Abbildung 2.7.: Aufbau des Projektnetzes

gen bewährt und ist auf die Bedürfnisse großer IT-Umgebungen in Unternehmen zugeschnitten. Dabei wird VMware ESX direkt auf der Serverhardware installiert und fügt dort eine Virtualisierungsebene zwischen der Hardware und dem Betriebssystem ein (siehe auch Abbildung 2.1). Dies ermöglicht die Ausnutzung der Server Hardware durch mehrere virtuelle Maschinen. VMware nutzt dazu die sogenannte Bare-Metal-Architektur, die eine fast originalgetreue Serverperformance und Zuverlässigkeit ermöglicht. Weitere Informationen zu VMware ESX findet man unter [VMw09d].

2.3.2. Möglichkeiten zum Management von VMware ESX

Für den ESX Server stellt VMware ein sogenanntes „VMware Infrastructure object model“ zur Verfügung. Dieses Modell besteht im Wesentlichen aus serverseitigen Datenstrukturen für die Verwaltung, Überwachung, Konfiguration und weitere Managementaufgaben im gesamten Life-Cycle der Infrastrukturkomponenten. Es wird auf allen ESX Servern von VMware mitgeliefert. Allerdings erhält man die volle Funktionalität - insbesondere in Bezug auf die Verwaltung mehrerer Hosts - nur in Verbindung mit dem Virtual Center, einer proprietären Managementlösung von VMware. So ist beispielsweise das Migrieren von virtuellen Maschinen zwischen zwei Hosts oder das Verwalten ganzer Host-Cluster nur von Virtual Center unterstützt. Für Entwickler bietet VMware die Möglichkeit auf die serverseitigen Managementobjekte über die VMware Infrastructure API (VI API) zuzugreifen. Die VI API ist ein sprachneutrales Interface auf den „Server-Sided Management Layer“ der VMware Produkte.

Über den Managed Object Browser (MOB), der auf ESX Servern mitgeliefert wird, kann man die Managementobjekte erkunden, die auf dem Server liegen. Der Zugriff auf den MOB ist mit jedem Webbrowser möglich. Dazu muss man mit dem Browser die Adresse des Managementhosts aufrufen. Die URL zum Aufruf des MOB lautet:

```
https://<hostname.yourcompany.com>/mob
```

Nach Eingabe von Username und Passwort kann man nun die MOB Objekte des ESX Servers mit dem Browser durchsuchen. Die Abbildung 2.8 zeigt ein Beispiel für solch ein aufgerufenes serverseitiges Managementobjekt. Man sieht hier exemplarisch die Zusammenfassung der Eigenschaften einer virtuellen Maschine. Es werden Name, Typ und Wert der Eigenschaft angezeigt.

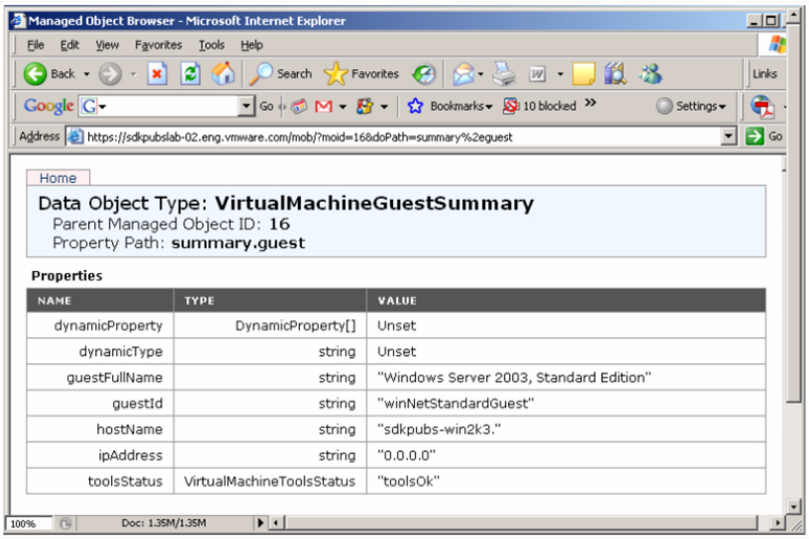

Abbildung 2.8.: Screenshot des MOB

Alle Managementobjekte befinden sich auf dem Server, sie werden nur durch Referenzieren an Client-Anwendungen weitergegeben. Somit muss jede Anwendung immer erst eine Verbindung zu dem Host aufbauen, danach den richtigen „Pfad“ zum gesuchten Objekt finden und dessen Referenz ermitteln. Der Client kann dann die Werte auslesen oder auch manipulieren. Dies geschieht durch Methoden, die ebenfalls auf dem Server liegen und mit der Referenz des Managementobjekts aufgerufen werden. Für das Entwickeln eigener Managementanwendungen stellt VMware drei unterschiedliche Programmierschnittstellen zur Verfügung: Das Virtual Infrastructure SDK (VI SDK), das Virtual Infrastructure Perl Toolkit (VI Perl Toolkit) und eine Virtual Infrastructure CIM/SMASH API (VI CIM). VI SDK und VI Perl Toolkit greifen über das VI API auf die serverseitigen Managementobjekte des Virtualisierungshosts zu. Anders das VI CIM, es greift auf eine Schnittstelle (CIM Object Manager, CIMOM) zu, die im Wesentlichen zum Managen der Serverhardware genutzt werden kann. Die Abbildung 2.9 zeigt eine Übersicht der angebotenen Programmierschnittstellen. Im Folgenden werden die unterschiedlichen Programmierschnittstellen kurz beschrieben bevor die Auswahl einer Lösung begründet wird.

Virtual Infrastructure SDK (VI SDK)

Im VI SDK werden alle Komponenten zur Verfügung gestellt, die man benötigt, um mit der VI API eigene Managementanwendungen in Java oder .NET (C #) zu entwickeln. Das sind insbesondere die WSDL Files, die benötigt werden, um von einer Client-Anwendung aus mit den serverseitigen Managementobjekten zu kommunizieren. Weiterhin mitgeliefert ist auch noch Beispielcode in Java und C# sowie Batch-Dateien und Shell-Skripte um die Codegeneration aus Java und C# zu vereinfachen. Das VI SDK benötigt außerdem noch Apache Axis

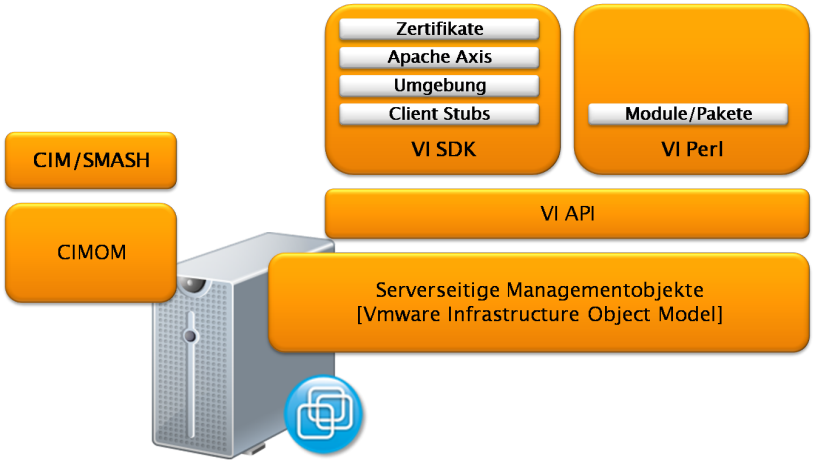

Abbildung 2.9.: VMware Programmierschnittstellen

in der Version 1.4. Will man mit dem ESX Server über HTTPS kommunizieren, so muss man zusätzlich noch Zertifikate installieren und für deren Verteilung bei den Clients sorgen. Insgesamt ist die Konfiguration eines Client Systems für das Arbeiten mit dem VI SDK aufwändig: Neben dem SDK selbst muss Apache Axis in Version 1.4 für die Kommunikation über WebServices installiert werden. Weiterhin müssen insgesamt 25 Umgebungsvariablen zu unterschiedlichen Bibliotheken gesetzt werden und die entsprechenden Serverzertifikate in den Java-Keystore importiert werden. Anschließend müssen noch die Client Stubs für die WSDL Kommunikation generiert werden und danach alle Java Archive damit neu kompiliert werden. Das Client-Setup für das SDK ist mit hohem Aufwand verbunden und muss zudem für jede Änderung des Clients angepasst werden (JDK Version, Apache Axis Version, Zertifikate). Weitere Informationen zum VI SDK findet man in [VMw08a], [VMw08d] und [VMw08h].

VI Perl Toolkit

Um den Konfigurationsaufwand, der bei Nutzung des VI SDK auf der Client-Seite entsteht zu minimieren wird von VMware das VI Perl Toolkit zur Verfügung gestellt: Es bietet zahlreiche Managementaufgaben in einer VMware Infrastruktur Umgebung und arbeitet ebenfalls mit der VI API zusammen. Die im Toolkit mitgelieferten Skripte verbinden sich mit dem Host, suchen und finden die geforderten serverseitigen Managementobjekte und empfangen oder ändern deren Werte. Der Vorteil des VI Perl Toolkit ist die wesentlich einfachere Konfiguration des Client-Systems: Nach dem Bezug des VI Perl Toolkit müssen nur noch wenige Pakete ergänzt werden und schon kann das Toolkit installiert werden. Es müssen keine Stubs generiert werden, es müssen keine Umgebungsvariablen gesetzt werden, es müssen keine Zertifikate verwaltet werden. Weitere Informationen zum VI Perl Toolkit findet man unter [VMw08b], [VMw08c] und [VMw08e].

CIM/SMASH API

Ab Version 3.5 des ESX Servers wird ein CIM Object Manager (CIMOM) mitgeliefert. CIMOM bietet Server Monitoring Funktionen, die mit dem SMASH (Systems Management

Architecture for Server Hardware) Standard kompatibel sind. Das VMware CIM/SMASH API bietet folgende Funktionalitäten: Systemressourcen auflisten, Zustand der Systeme überwachen und Abschalten von Host Systemen zur Wartung. Das CIM/SMASH API hat einen reinen Fokus auf den Host und dessen Sensoren. Das Managen von virtuellen Maschinen ist nicht möglich. Weitere Informationen zum CIM/SMASH API sind unter [VMw09b] und [VMw09a] zu finden.

2.3.3. Anforderungen an eine Schnittstelle

VMware bietet insgesamt drei Schnittstellen zum Managen eines ESX Servers an. VI SDK und VI Perl Toolkit greifen auf die VI API zu, das CIM/SMASH API dient lediglich zum Überwachen der Serverhardware und wird deswegen hier nicht weiter betrachtet. Bevor man mit der Entwicklung eines Adapters beginnt, gilt es zunächst die Anforderungen an die Schnittstelle zu definieren. In [HAN99] (Seite 299 bis 302) sind wesentliche Kriterien für die Auswahl einer Managementplattform genannt. Da der im Rahmen dieser Arbeit entwickelte Adapter die Managementplattform in Ihrer Funktionalität unterstützt, sind diese Kriterien auch als Auswahlkriterien für eine Managementschnittstelle heranzuziehen. Wichtige Kriterien in diesem Zusammenhang sind:

- Flexibilität: Der Adapter muss flexibel sein und an die beim Betreiber eingesetzte Technologie anpassbar sein.
- Integrationsfähigkeit: Der Adapter darf keinen eigenen Datenbestand aufbauen oder nutzen.
- Sicherheit: Entsprechende Mechanismen sollen eine sichere Kommunikation ermöglichen.
- Rentabilität: Die für den Einsatz anfallenden Kosten müssen den erzielbaren Nutzen aufwiegen.
- Zukunftsträchtigkeit: Bei der Auswahl einer Schnittstelle ist abzuwägen, ob die Technologie sich auf dem Markt durchsetzen und halten wird.
- Modularität: Eine Austauschbarkeit der Komponenten soll die Wartbarkeit verbessern.
- Hard- und Softwarevoraussetzungen des Systems: Die Anforderungen an das Trägersystem bestimmen den Einsatz einer Managementlösung.
- Produktverfügbarkeit
- Release-Planungen und Produktankündigungen
- Erfahrungsberichte und Referenzen
- Dokumentationsgüte
- Kundenunterstützung

Flexibilität: Sowohl das VI SDK als auch das VI Perl Toolkit haben gewisse Mindestvoraussetzungen an das System: Beim VI SDK sind das eine Java oder .NET Entwicklungsumgebung und Apache Axis. Das VI Perl Toolkit hingegen läuft auf jedem Endsystem, das Perl unterstützt. Bei der Installation des VI Perl Toolkit fällt weniger Konfigurationsarbeit an, es ist schneller installiert und einfacher zu nutzen als das VI SDK.

Integrationsfähigkeit: VI SDK und VI Perl Toolkit greifen nur auf die Managementobjekte des ESX Servers zu und liefern deren Werte aus. Es wird kein eigener Datenbestand aufgebaut. Beide Schnittstellen unterstützen damit gleichermaßen die Forderung nach dem Vermeiden eigener Datenbestände.

Sicherheit: VI SDK und VI Perl Toolkit kommunizieren über HTTPS mit dem ESX Server. Beim VI Perl Toolkit wird HTTPS standardmäßig ohne zusätzlichen Konfigurationsaufwand genutzt. Möchte man das VI SDK über HTTPS nutzen, so muss man die HTTPS-Zertifikate vom ESX Server exportieren und manuell in die Konfiguration des VI SDK einbinden. Beide Lösungen unterstützen die HTTPS-Kommunikation, allerdings ist die Nutzung und Einrichtung mit dem VI Perl Toolkit unkomplizierter.

Rentabilität: Kostenbetrachtungen wurden in dieser Arbeit grundsätzlich nicht betrachtet. Allerdings entsteht ein großer Teil der Kosten durch Personaleinsatz: Wenn man die aufwändigere Konfiguration des VI SDK auf die damit verbundenen Personalkosten umlegt, so ergibt sich hier ein Vorteil für das VI Perl Toolkit, da es schneller und einfacher einzurichten und in Betrieb zu nehmen ist.

Zukunftsträchtigkeit: VI SDK nutzt Java oder .NET, VI Perl Toolkit nutzt Perl als Programmiersprache. Beide Technologien sind mittlerweile etabliert und blicken auf eine längere Geschichte zurück.

Modularität: Hat man sich für eine Managementschnittstelle entschieden, so kann man diese natürlich mit entsprechendem Aufwand gegen die andere Lösung austauschen. Sollte es aber mit einer der beiden Lösungen irgendwelche Probleme geben, die fordern, dass man sich für eine andere Managementschnittstelle entscheiden muss, so ist man davon abhängig, ob von VMware eine Alternative zur Verfügung gestellt wird. Momentan gibt es neben den von VMware selbst angebotenen Managementschnittstellen keine alternativen Lösungen.

Hard- und Softwarevoraussetzungen: Wie schon in den vorherigen Abschnitten erwähnt, hat das VI Perl Toolkit geringere Anforderungen an das Endsystem. Es muss keine zusätzliche Software (VI SDK: Apache Axis) installiert werden.

Produktverfügbarkeit: Beide Schnittstellen werden von VMware zum Download bereitgestellt und sind über die Entwicklerseiten von VMware verfügbar.

Releaseplanungen: Die letzte Version des VI Perl Toolkit war vom Juli 2008, davor gab es regelmäßige Aktualisierungen des VI Perl Toolkit. Das VI SDK hingegen wurde seit Dezember 2007 nicht mehr weiterentwickelt und es gab auch keine Ankündigung seitens VMware, ob das Produkt weiterentwickelt wird.

Erfahrungsberichte: VMware gibt in der Dokumentation des VI SDK zu, dass es einen sehr hohen Overhead bei der Systemkonfiguration verursacht und rät daher zur Verwendung des VI Perl Toolkit (vgl. [VMw08a], Seite 8). In den Entwicklerforen gibt es eine ähnliche Tendenz, was den Einsatz der Managementschnittstellen angeht. Außerdem fällt auf, dass es im Internet zusätzliche APIs gibt, die die eigentliche Nutzung des VI SDK vereinfachen sollen (vgl. [vJAC09]).

Dokumentationsgüte: Beide Lösungen sind gut dokumentiert. Es existieren jeweils Installations- und Programmierhandbücher zu den Schnittstellen. Die Dokumentationen werden von VMware in englischer Sprache herausgegeben und aktualisiert. Daneben gibt es eine aktive Entwicklercommunity, in der Fragen meist schnell und zielführend beantwortet werden.

Kundenunterstützung: Neben den Handbüchern und der Entwicklercommunity gibt es keine weitere Unterstützung durch VMware.

Die Tabelle 2.2 zeigt eine Zusammenstellung und Bewertung der eben aufgeführten Auswahlkriterien aus [HAN99].

	VI SDK	**VI Perl Toolkit**
Flexibilität	o	+
Integrationsfähigkeit	++	++
Sicherheit	o	++
Rentabilität	-	+
Zukunftsträchtigkeit	+	++
Modularität	o	o
Hard- und Softwarevoraussetzungen	-	+
Produktverfügbarkeit	+	+
Release-Planung	-	+
Erfahrungsberichte	-	+
Dokumentationsgüte	+	+
Kundenunterstützung	o	o

Tabelle 2.2.: Übersicht der Anforderungen

2.3.4. Auswahl einer Möglichkeit

Nach der Betrachtung des VI SDK und des VI Perl Toolkit und der Analyse der jeweiligen Installations- und Entwicklerhandbücher ist die Entscheidung für das VI Perl Toolkit gefallen. Der Hauptgrund ist die wesentlich einfachere Einrichtung der Clientsysteme bei der Installation des VI Perl Toolkit: Im Grunde reicht ein System mit einer Perl Installation und wenigen Zusatzpaketen und Modulen. Die umständliche Konfiguration von Umgebungsvariablen, die Installation von Apache Axis, das Generieren von Client-Stubs, das Verwalten und das Einbinden der Serverzertifikate entfällt bei der Installation des VI Perl Toolkit. Details zur Installation des VI Perl Toolkit finden sich in Kapitel 4.1 dieser Arbeit. Weiterhin erfolgt die Kommunikation der Perl Skripte mit dem ESX Server standardmäßig über HTTPS und ist somit ohne Zusatzaufwand bei der Konfiguration gesichert. Außerdem war zum Zeitpunkt

des Adapterentwurfs nur eine aktuelle Version des VI Perl Toolkit verfügbar, während die letzte Version des VI SDK seit Dezember 2007 nicht weiterentwickelt wurde.

Das CIM/SMASH API deckt einen anderen Anforderungsbereich ab, es kann lediglich die Hardware des Servers überwacht und gemanagt werden. Ein Monitoring von virtuellen Maschinen, die auf dem Host laufen, wird durch CIM/SMASH nicht unterstützt. Daher wurde das CIM/SMASH API für die Auswahl einer konkreten Schnittstelle auch nicht betrachtet.

3. Konzept des Adapters

In diesem Abschnitt wird das Konzept des Adapters dargestellt. Zunächst wird auf die Funktionalität des Adapters definiert. Nach einer Betrachtung der Schnittstellen werden die Architektur des Adapters und die Kommunikation mit der Managementplattform erläutert. Im letzten Abschnitt dieses Kapitels werden die nötigen syntaktischen und semantischen Konvertierungen erklärt.

3.1. Funktionalität des Adapters

Der Adapter unterstützt folgende Funktionalitäten:

- Verbindung zum Host
- Auslesen des Hostnamens
- Auslesen des Host-Softwaretyps
- CPU-Auslastung des Hosts
- Arbeitsspeicherauslastung des Hosts
- CPU-Auslastung der Virtuellen Maschine
- Arbeitsspeicherauslastung der Virtuellen Maschine

3.2. Schnittstellen

Der Adapter ist das Bindeglied zwischen der Managementplattform und dem VMware ESX Server. Dazu muss der Adapter die angebotenen Informationen des ESX Servers auslesen und für die Plattform aufbereiten. Die Abbildung 3.1 veranschaulicht die Schnittstellen des Adapters und der Managementplattform und zeigt den Informationsfluss zwischen den Komponenten. Auf der linken Seite der Abbildung ist die Managementplattform mit den von der Plattform genutzten Performancedaten dargestellt. Auf der rechten Seite der Abbildung steht der ESX Server mit seinen Managementinformationen. Zwischen Managementplattform und ESX Server befindet sich der VMware Adapter und das VI Perl Toolkit. Der VMware Adapter nimmt eine Nachricht von der Managementplattform entgegen und verarbeitet sie, um die entsprechende Operation auszuführen und die benötigten Informationen vom ESX Server abzuholen. Dazu schickt der VMware Adapter ein Perl Script aus dem VI Perl Toolkit ab und wartet auf die entsprechende Antwort mit den angeforderten Performancedaten des ESX Servers. Nachdem der VMware Adapter die Daten erhalten hat, extrahiert er diese aus der Antwort des ESX Servers und führt semantische und syntaktische Anpassungen der Werte durch, um die Performancedaten in das für die Managementplattform passende Format umzuwandeln. Die Berechnung und Konvertierung dieser Werte ist in Kapitel 3.5

dokumentiert. Nach der Umwandlung werden die Performancedaten wieder in eine Nachricht geschrieben und an die Managementplattform gesendet. Zur Verdeutlichung des grundsätzlichen Ablaufs hier exemplarisch die Abfolge bei Anforderung eines HOST CPU Updates durch die Managementplattform:

1. Managementplattform verschickt Nachricht mit HOST CPU Update
2. VMware Adapter empfängt und verarbeitet Nachricht
3. VMware Adapter löst Absenden des Perl Scripts für Hostinformationen aus
4. Perl Script fordert die Werte HOST CPU Available und HOST CPU Used an
5. ESX Server liefert Performancedaten an VMware Adapter
6. VMware Adapter berechnet aus den Werten die prozentuale Auslastung
7. VMware Adapter sendet Nachricht mit dem neuen HOST CPU Wert an die Plattform
8. Plattform nimmt Nachricht entgegen und verarbeitet enthaltene Information

Dieser grundsätzliche Ablauf ist in der Abbildung 3.1 auch durch den logischen Informationsfluss gekennzeichnet: Die Anforderung geht immer von der Managementplattform aus, VMware Adapter und VI Perl Toolkit lösen die eigentliche Informationsanforderung am ESX Server aus. Nach Anpassung und Berechnung der Werte wird vom VMware Adapter eine Nachricht an die Plattform zurückgesendet.

3.3. Architektur

Der Adapter zur Anbindung an den ESX Server ist Bestandteil des Kommunikationsbausteins der Managementplattform. Der Adapter selbst bekommt seine Managementinformationen aus zwei weiteren Klassen, die die Abfrage der Werte vom ESX Server realisieren und eine Anpassung der gelieferten Werte vornehmen. Abbildung 3.2 zeigt den Aufbau der Klassen des VMware Adapters.

Die Klasse ESX_Hostinfo liefert die Grundinformationen über das System, also den Namen des Virtualisierungshosts und den Typ des Hosts (Softwareversion) an den Adapter. Außerdem werden die Performancedaten (CPU- und Speicherauslastung) des Hosts von dieser Klasse ermittelt und an den Adapter weitergegeben. In der Klasse erfolgt auch die Berechnung der Werte aus den Rohdaten des Hosts.

Die Klasse ESX_Vminfo ist für die Performancedaten der virtuellen Maschinen zuständig. Sie berechnet die Performancewerte einer virtuellen Maschine und liefert diese an den VMware Adapter.

In der Klasse Vmwareadapter werden die Infos der beiden Klassen ESX_Hostinfo und ESX_Vminfo an die Managementplattform übergeben und weiterverarbeitet.

3.4. Kommunikation

Die beiden Klassen ESX_Hostinfo und ESX_Vminfo liefern die benötigten Informationen an den VMware Adapter, der diese an die Plattform weiterreicht. Jede dieser Klassen sendet ein

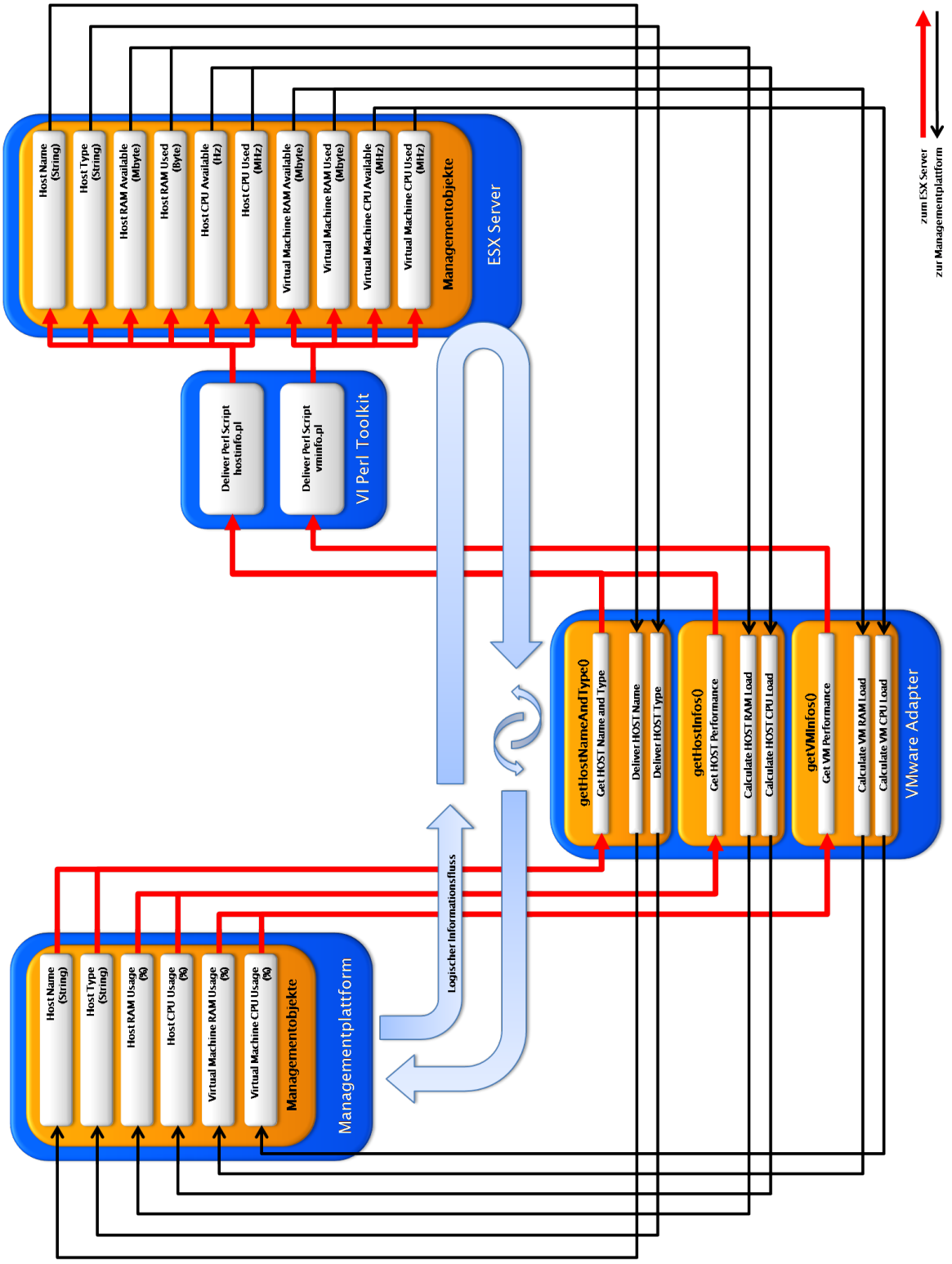

Abbildung 3.1.: Schnittstelle von Plattform und ESX Server

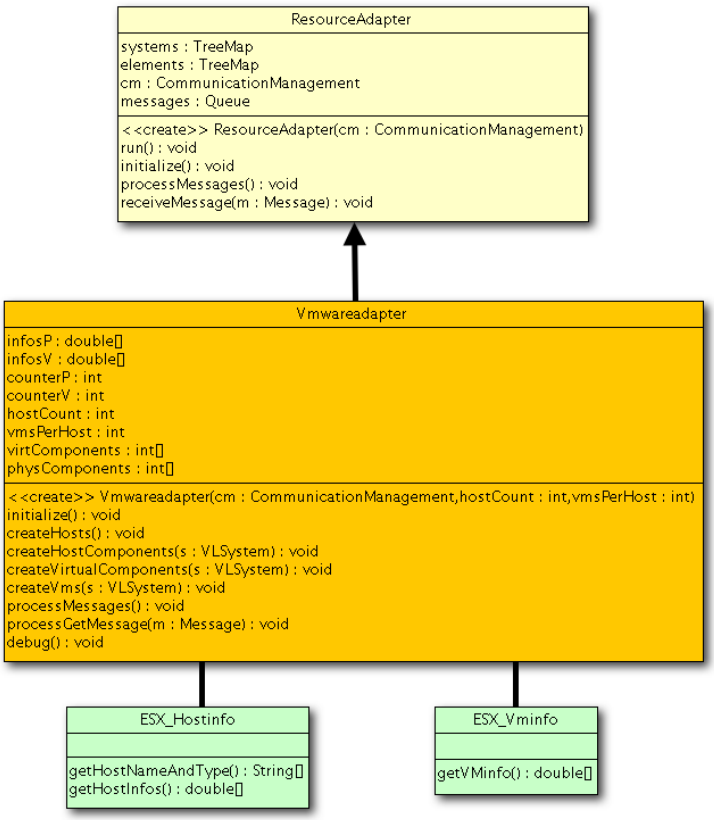

Abbildung 3.2.: Klassendiagramm des VMware Adapters

perl Skript mit der Anforderung der gewünschten Daten an den ESX Server. Die resultierende Antwort des ESX Servers wird in der die Anfrage absendenden Klasse (ESX_Hostinfo oder ESX_Vminfo) geparst und verarbeitet (vgl. auch Abbildung 3.1). Die Performancedaten von Host oder virtueller Maschine werden berechnet und dann als Array an den VMware Adapter weitergegeben. In der Klasse Vmwareadapter selbst erfolgt keine Berechnung der Daten mehr. Hier werden die Werte lediglich an die Plattform weitergegeben. Durch die Trennung von Berechnung und Datenweitergabe wird die eigentliche Ermittlung der Daten vor dem Adapter verborgen. So kann die zu Grunde liegende Werteberechnung unabhängig vom Adapter geändert und angepasst werden, ohne, dass man den Adapter, der die Kommunikation mit der Plattform sicherstellt, verändern muss. Die Klasse Vmwareadapter benötigt lediglich einen double bzw. String Wert als Eingabe von den informationsliefernden Klassen (ESX_Hostinfo oder ESX_Vminfo). Die Berechnung und Ermittlung der Daten ist dem VMware Adapter verborgen.

3.5. Semantische und syntaktische Konvertierung

Die vom ESX Server gelieferten Werte für die Performanz des Servers müssen in den beiden informationsliefernden Klassen (ESX_Hostinfo und ESX_Vminfo) noch für die Plattform konvertiert werden. Aus den gelieferten Rohdaten wird die Prozentuale Auslastung der Systeme ermittelt. Folgende Daten über die Performanz benötigt die Managementplattform vom ESX Server:

- CPU-Auslastung Host in Prozent
- RAM-Auslastung Host in Prozent
- CPU-Auslastung der virtuellen Maschine in Prozent
- RAM-Auslastung der virtuellen Maschine in Prozent

Folgende Werte werden zur Berechnung der Auslastung vom ESX Server geliefert:

i	**Counter** C_i	**Ressource**	**Einheit** E_i	**Faktor**$F_(i)$
1	CPU Available	ESX Host	MHz	·1000000
2	CPU Usage	ESX Host	Hz	1
3	RAM Available	ESX Host	MByte	÷1024
4	RAM Usage	ESX Host	byte	· 1024
5	CPU Available	VM	MHz	1
6	CPU Usage	VM	MHz	1
7	RAM Available	VM	MByte	1
8	RAM Usage	VM	MByte	1

Tabelle 3.1.: Vom ESX Server gelieferte Daten

Die prozentualen Werte werden durch Berechnung nach folgender Formel ermittelt:

$$PerformacePercentage(i) = 100 \cdot \frac{Z(i+1) \cdot F(i+1)}{Z(i) \cdot F(i)}$$

wobei i mod 2 immer null sein muss.

4. Implementierung

Dieses Kapitel dokumentiert die Implementierung des Adapters. Nachdem grundlegende Vorbereitungen und Anpassungen an den Systemen beschrieben werden, wird die Installation des VI Perl Toolkit erklärt und ein Überblick über die wichtigsten Punkte bei der Programmierung des Adapters gegeben.

4.1. Grundlegende Vorbereitungen

4.1.1. Vorbereitungen auf dem Virtualisierungshost

Auf dem ESX Server sind nach der Standardinstallation keine weiteren Anpassungen nötig, um mit dem VI Perl Toolkit zu arbeiten. Der ESX Server benötigt im Wesentlichen nur eine korrekte Netzkonfiguration, um über das Netz über eine korrekte IP-Adresse erreichbar zu sein. Man kann zum Testen der Netzkonfiguration mit dem ESX Server eine Verbindung aufbauen, indem man auf den MOB des ESX Servers zugreift. Der Zugriff auf den MOB ist in Kapitel 2.3.2 beschrieben. Sollte der MOB nicht aufrufbar sein, so muss die Konfiguration des Servers überprüft werden.

4.1.2. Vorbereitungen auf dem Managementhost

Auf dem Managementhost sind nur kleine Vorbereitungen nötig. Es müssen einige Bibliotheken, Pakete und Perl-Module installiert sein, bevor man das VI Perl Toolkit installieren kann. Zum Bezug der fehlenden Daten muss der Managementhost über eine Verbindung in das Internet verfügen.

Benötigte Bibliotheken und Pakete

Für die Installation des VI Perl Toolkit müssen folgende Pakete auf dem System installiert sein:

- Perl 5.8.8
- Linker utility (binutils)
- GNU C Libraries (glibc oder libc6)
- XML DOM/SAX Libraries (libxml12)
- Perl Documentation (perl-doc)
- Perl URI-library (liburi-perl)
- OpenSSL-Library (libssl-dev)

Die Pakete können auf jeder Linux Distribution mit den entsprechenden Befehlen (apt, rpm oder vergleichbar) installiert werden und können unter Umständen auch schon Bestandteil einer Linux Distribution sein.

Bezug benötigter Perl-Module

Je nach Installations- und Distributionstyp werden weiterhin noch einige Perl-Module benötigt. Diese sind leicht über die Perl CPAN Shell zu installieren. CPAN ist das Comprehensive Perl Archive Network - ein weltweit gespiegeltes Online-Repository, in dem Perl-Module und Dokumentationen verfügbar sind. Mittlerweile ist CPAN in der Perl-Entwickler Welt zum defacto-Standard geworden. Nähere Informationen zu CPAN gibt es auf den Webseiten des CPAN unter [Hie09] Die CPAN Shell zum Installieren und verwalten von Modulen ruft man mit folgendem Befehl auf:

```
perl -MCPAN -e shell
```

Nach Eingabe dieses Befehl befindet man sich in der CPAN Shell. Nun kann man nach Eingabe von

```
install <Modulname>
```

die fehlenden Perl-Module installieren. Folgende Perl-Module werden benötigt:

- Crypt-SSLeay (0.51) [Crypt::SSLeay]
- Data-Dumper (2.102) [Data::Dumper]
- MethodMaker (2.0.8) [Class::MethodMaker]
- XML-LibXML (1.60) [XML::LibXML]
- XML-LibXML (1.60) [XML::LibXML]
- libwww-perl (5.805) [LWP]

Die Module werden mit dem obigen Befehl installiert. Nach der Installation der fehlenden Pakete verlässt man die CPAN Shell über den Befehl

```
install <Modulname>
```

exit

Bezug des VI Perl Toolkits

Für diesen Adapter wird das VI Perl Toolkit in der Version 1.6 verwendet. Es ist über die VMware Webseite zu beziehen ([VMw08i]). Für den Bezug des VI Perl Toolkit muss man sich mit einer gültigen E-Mail Adresse bei der Entwickler-Community von VMware registrieren. Neben dem Bezug der Software bekommt man von der Entwickler-Community Unterstützung bei Fragen zu den VMware Produkten. Die Community ist im Internet über [VMw09c] erreichbar, spezielle Unterstützung zum VI Perl Toolkit findet man unter [VMw08f]. Nach dem Download des VI Perl Toolkit kann man das Produkt installieren, falls alle oben genannten Pakete auf dem Host eingerichtet sind. Das tar.gz File enthält neben den RedHat und Ubuntu Versionen des VI Perl Toolkit auch den Quellcode, den man für jede Linux-Distribution mit den üblichen Befehlen make und make install kompilieren kann. Nach dem Kompilieren ruft man die Installationsroutine des VI Perl Toolkit mit folgendem Befehl auf:

```
<Installationsverzeichnis des VI Perl Toolkit>/perl vmware-install.pl
```

Die Routine startet und überprüft die Systemkonfiguration. Sollten noch Pakete oder Module fehlen, so wird dies in der festgestellt und entsprechend gemeldet. Die fehlenden Pakete müssen dann noch manuell installiert werden, bevor man die Installationsroutine noch einmal aufruft. Nachdem die Installationsroutine durchlaufen ist, bekommt man eine Meldung über die erfolgreiche Installation und Informationen zum Entfernen des VI Perl Toolkit vom System. Nun kann man die Installation des VI Perl Toolkit mit folgendem Befehl überprüfen:

```
<Verzeichnis des VI Perl Toolkit>/apps/general/
perl connect.pl --server <IP-Adresse des ESX Servers>
```

Nach Eingabe von Username und Passwort erhält man folgende Ausgabe:

```
Connection Successful
Server Time : 2009-06-02T14:09:19.005378Z
```

Für weitergehende Informationen zur Installation wird von VMware ein Installationshandbuch bereitgestellt([VMw08c]).

4.2. Programmierung des Adapters

Der Adapter nutzt verschiedene Perl Skripte um die benötigten Werte vom ESX Server anzufordern. Die Antwort des Servers enthält die gefragten Werte, die im Adapter konvertiert werden. Nach dieser Formatanpassung werden die von der Plattform benötigten Werte aus den konvertierten ESX-Daten berechnet und an den Adapter weitergegeben. Dieser leitet die Daten dann an die Plattform weiter.

4.3. Perl-Skripte

Das VI Perl Toolkit liefert schon eine Vielzahl von Scripts für die gängigsten Anwendungen zum Managen des ESX Servers mit. Zum Auslesen von Performancedaten und Konfiguration von Host und virtuellen Maschinen kann auf die im Toolkit mitgelieferten Skripte zurückgegriffen werden. Die Tabelle 4.1 zeigt die Funktionalität einiger mitgelieferter Perl Scripts.

Bei allen Skripten müssen verschiedene Parameter beim Aufruf mitgegeben werden, um das Script an den richtigen Virtualisierungshost zu senden. Hier ein Beispielaufruf:

```
perl hostinfo.pl
        --server 192.168.2.26
        --fields hostname,software
        --username <username>
        --password <password>
```

In obigem Beispiel werden die Adresse des Servers über den Parameter --server 192.168.2.26 abgefordert. Die Parameter --fields hostname,software fordern den Namen des Hosts und die auf dem Host installierte Software (ESX Server Version) an. Die Felder <username> und <password> dienen dem Authentisieren des Nutzers und müssen immer mit gesendet werden, um die Managementinformationen vor unberechtigtem Zugriff zu schützen. Zum Entwickeln eigener Skripte ist eine API vorhanden. Die API und weiterführende Informationen zum Umgang mit den Perl Scripts und deren Programmierung erhält man unter [VMw08b], [VMw08c], [VMw08e] und [VMw08g].

Script	Funktion
connect.pl	Connects to and disconnects from a host.
dsbrowse.pl	Browses datastores and lists their attributes.
extractlog.pl	Extracts the vmware.log for any virtual machine.
fileaccess.pl	Performs put and get operations on datastore and configuration files.
guestinfo.pl	Lists and customizes attributes of a guest OS on a virtual machine.
hostdiagnostics.pl	Extracts the specified log from the host of the virtual center.
hostevacuate.pl	Migrates all virtual machines from one host to another.
hostinfo.pl	Displays the processor, network and memory attributes of the hosts.
hostops.pl	Performs host operations: add standalone, disconnect, reconnect, enter and exit maintenance mode, reboot, shutdown host, add host, remove host, move host into folder cluster.
load_session.pl	Loads a saved session to a host.
save_session.pl	Connects to a host and saves session state in a file.
sharesmanager.pl	Displays or modifies shares for memory, cpu, and disk for specified virtual machines.
snapshotmanager.pl	Captures the state of one or more virtual machines into a snapshot.
vdiskcreate.pl	Creates a new virtual disk on a virtual machine.
vidiscovery.pl	Displays the hierarchy of managed virtual entities.
viperformance.pl	Retrieves performance counters from a host.
viversion.pl	Displays all information about the product.
vmclone.pl	Clones a virtual machine while customizing cloned virtual machine and guest OS.
vmcontrol.pl	Operates power-on, power-off, suspend, and reset on the virtual machine; operates reboot, shutdown, and standby on the guest OS.
vmcreate.pl	Creates virtual machines according to an XML specification file.
vminfo.pl	Lists the properties of the virtual machines.
vmmigrate.pl	Migrates virtual machines within the current host, or to a different host.
vmreconfig.pl	Reconfigures a virtual machine.
vmregister.pl	Registers and unregisters a virtual machine.
vmsnapshot.pl	Creates snapshots of virtual machines.
vmtemplate.pl	Converts a virtual machine to a template and template back to a virtual machine.

Tabelle 4.1.: Im VI Perl Toolkit mitgelieferte Perl Skripte (vgl. [VMw08e])

4.3.1. Aufruf der Perl-Skripte in Java

Java kann zwar viele Skriptsprachen direkt ansteuern, jedoch ist die Skriptsprache Perl nicht direkt unterstützt. Daher wird hier der Umweg über ein runtime.exec() und ein gepuffertes Einlesen der daraus resultierenden Konsolenausgabe gegangen. Nach dem Absetzen der Skripte werden die Rückgaben des Servers geparst und anschließend in Java konvertiert und für die Weitergabe an die Plattform aufbereitet. Mit dem Befehl

```
String Command = "perl /root/vmware-viperl-distrib/apps/host/hostinfo.pl
        --server 192.168.2.26
        --fields hostname,software
        --username <username>
        --password <password>";
```

wird das aufzurufende Script in den String Command geschrieben. Danach kann dieses Script über folgenden Code:

```
Runtime runtime = Runtime.getRuntime();
Process process = runtime.exec(Command);
```

aus der Java Anwendung heraus abgesetzt werden. Die Antwort des Servers auf dieses Script wird auf die Konsolenausgabe geleitet und muss dann für die Weiterverarbeitung in Java geparst werden. Dies wird durch einen Buffered Reader realisiert, der die Konsolenausgabe in einen String überführt:

```
InputStream is = process.getInputStream();
InputStreamReader isr = new InputStreamReader(is);
BufferedReader br = new BufferedReader(isr);
String line;
StringBuffer temp = new StringBuffer();
while ((line = br.readLine()) != null) {
        temp.append(line);
        temp.append("\n");
}
//StringBuffer in String umwandeln
String Ausgabe = temp.toString();
```

Mit entsprechenden String-Operationen werden die vom ESX Server in der Antwort gelieferten Werte extrahiert und berechnet. Die Konvertierung und Berechnung der Werte ist in Kapitel 3.5 beschreiben.

4.3.2. Workaround zur Erhöhung der verfügbaren Werte für den History Graph

Die Managementplattform greift auf die Messwerte der letzten 60 Sekunden zu und stellt diese als Graph in der Benutzerschnittstelle dar. Sollten innerhalb der letzten 60 Sekunden weniger als zwei Messwerte zur Verfügung stellen, wird von der Plattform eine Fehlermeldung erzeugt. Bei längeren Testläufen wurde die Anzahl der Messwerte beobachtet. Je nach Last von ESX Server und Projektnetz waren innerhalb der letzten 60 Sekunden 4 bis 16 Messwerte bei der Plattform registriert. Um die Zuverlässigkeit der Managementplattform zu erhöhen wurde ein Workaround zur künstlichen Erhöhung der Messwerte eingefügt: Dazu werden die Messwerte nicht direkt an die Plattform übergeben, sondern in einem Array zwischengespeichert.

4.4. Erweiterung der Managementplattform

An der Managementplattform müssen einige Änderungen durchgeführt werden, um den Adapter und seine Funktionalität zu unterstützen. Im Wesentlichen wurden zwei Methoden zum Auslesen der aktuellen Arbeitsspeicherauslastung hinzugefügt und zusätzliche Messages und Operationen für die Auslastung des Arbeitsspeichers ergänzt. In Benutzeroberfläche wurden eine neue grafische Darstellung der Arbeitsspeicher- und Prozessorauslastung eingefügt. Dazu wurde die Bibliothek JpGraph genutzt (Bezug und Dokumentation unter [Con09]). Alle Änderungen an der Managementplattform sind im Anhang, Abschnitt A.1 dokumentiert.

4.5. Registrierung des Adapters

Um den Adapter an der Plattform zu registrieren sind einige Änderungen nötig: Es muss eine zusätzliche Schaltfläche für den VMware Adapter in der Nutzeroberfläche einfügt werden und eine Funktion zum Aufruf des Adapters hinterlegt werden. Dazu waren kleinere Anpassungen am Aussehen der Nutzeroberfläche nötig. Weiterhin müssen in allen Bausteinen der Plattform entsprechende Methoden zum Handling der Messages des neuen Adapters hinzugefügt werden. Eine umfangreiche Übersicht mit allen angepassten Dateien befindet findet sich im Anhang, Abschnitt A.2.

5. Zusammenfassung

5.1. Erreichte Ziele

In dieser Arbeit wurde ein Adapter entwickelt, der die Kommunikation zwischen der in [Bit08] entwickelten Managementplattform und einem VMware ESX Server ermöglicht. Das Hinzufügen einer Ressource (ESX Server) zur Plattform und das Monitoring von Prozessor- und Arbeitsspeicherauslastung des hinzugefügten ESX Servers und der virtuellen Maschinen auf dem ESX Server sind möglich. Die Abbildung 5.1 zeigt die Benutzeroberfläche der Managementplattform nach der Erweiterung durch den im Rahmen dieser Arbeit entwickelten Adapter.

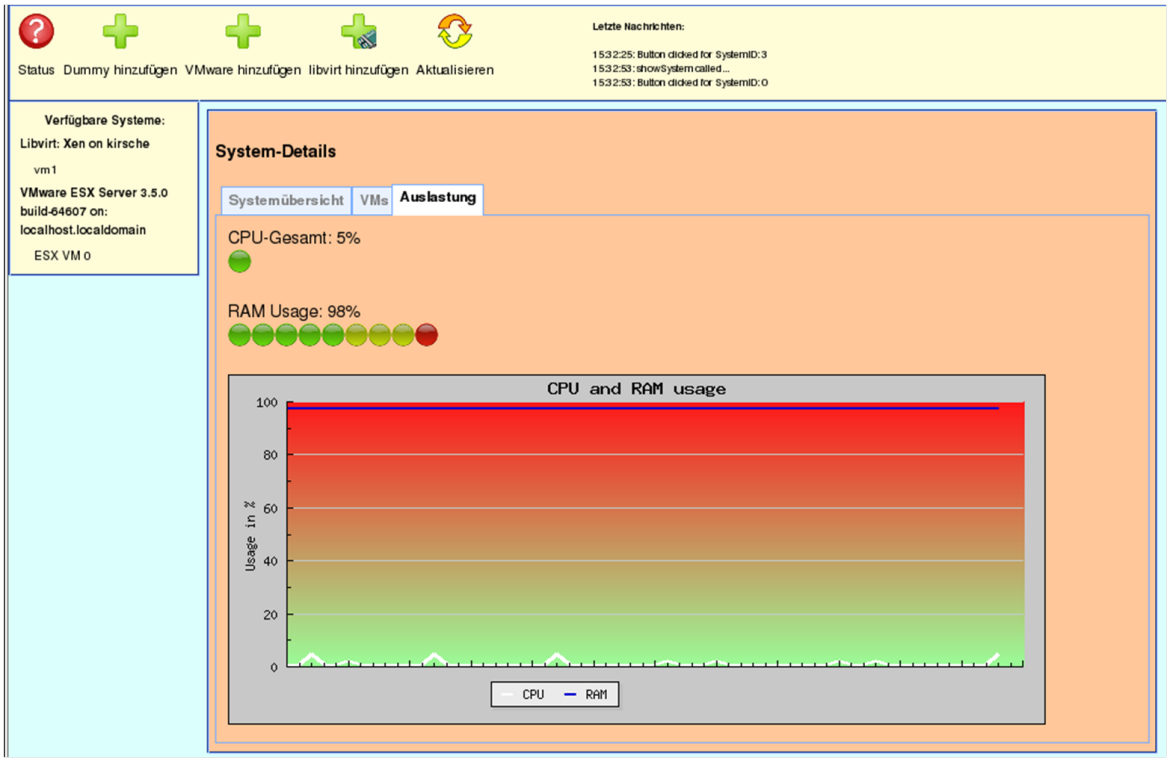

Abbildung 5.1.: Nutzeroberfläche der Managementplattform mit CPU- und RAM-Auslastung

5.2. Einschränkungen

Neben den durch die prototypische Implementierung der Plattform bedingten Einschränkungen in der Funktionalität der Plattform besteht ein wesentliches Problem bei der aktuellen

Umsetzung in der Anzahl der Nachrichten, die von der Plattform verwaltet werden und über das Netz geschickt werden. Für jeden Host und jede virtuelle Maschinen werden aktuell drei Nachrichten verwaltet: Eine für die Ressource selbst und je eine Nachricht für die CPU- und Arbeitsspeicherauslastung. Mit jeder zusätzlichen Monitoringfunktion und Ressource kommt also jeweils eine weitere Nachricht hinzu. Genauso ist das nachträgliche Anpassen der Anzahl der Virtuellen Maschinen pro Host nicht dynamisch Möglich, sondern nur durch einen Neustart der Adapter, da nach dem Hinzufügen der Ressourcen an der Plattform keine weiteren Mechanismen zur Anpassung der Ressourcenkonfiguration implementiert sind (Discoverymechanismus).

5.3. Ausblick

Weitere Schritte für die Entwicklung der Plattform sind neben dem beseitigen der eben genannten Einschränkungen auch die Erweiterung um eine Datenbankanbindung in der z.B. History-Daten der Plattform effizient verwaltet und ausgewertet werden können. Ebenfalls wichtig sind eine Datenbank zur Benutzerverwaltung und ein entsprechendes System mit unterschiedlichen Nutzergruppen und Berechtigungen. Der in dieser Arbeit entwickelte Adapter erweitert den Prototypen aus [Bit08] um Monitoringfunktionen für Hosts und virtuelle Maschinen; Im Rahmen weiterer Arbeiten ist insbesondere eine Erweiterung der Managementplattform um Steuerungsfunktionen für die Ressourcen (z.B. virtuelle Maschine starten/stoppen) denkbar.

A. Anhang

A.1. Anpassungen an der Managementplattform

/opt/vlmanagement/Code/src/WebUI/protected/pages/Home.php

In der Methode showSystem($id) muss die Ampeldarstellung für die Arbeitsspeicherauslastung eingefügt werden. Je nach Auslastung wird ein grünes, gelbes oder rotes Licht angezeigt. Analog zur Darstellung für die Prozessorauslastung wird folgender Code hinzugefügt:

```
if($this->soap->systemHasElems($id, "Ram")){
        $ram = $this->soap->getRam($id);
        $ramUsage = $this->soap->getRamUsage($ram->id);
        $content = $content."RAM Usage: ".$ramUsage."%";

        $ramUsageDurch10 = $ramUsage/10;
        $usagePicRam = $this->getUsagePic($ramUsageDurch10);
        $content = $content."    ".$usagePicRam."<br>";
}
```

/opt/vlmanagement/Code/src/WebUI/protected/pages/Home.php

Die Historygraphen für Arbeitsspeicher- und Prozessorauslastung sollen in einem gemeinsamen Graphen angezeigt werden. Der Graph wird mit Hilfe der Bibliothek JpGraph ([Con09]) erzeugt:

```
function sumPic($cpuHist, $ramHist, $picfile){
        $prod1=$cpuHist; //Daten für CPU-History
        $prod2=$ramHist; //Daten für RAM-History
        $graph = new Graph(700,300, "auto");
        $graph->SetScale('linlin',0,100);
        $graph->img->SetMargin(50,20,20,50);
        $graph->SetBackgroundGradient('red:1.1','green:1.6',GRAD_HOR,
            BGRAD_PLOT);
        $graph->ygrid->SetColor("azure3");
        $graph->title->Set("CPU and RAM usage");

        $lineplot1=new LinePlot($prod1);
        $lineplot1->SetColor("white");
        $lineplot1->SetWeight(2);
        $lineplot2=new LinePlot($prod2);
        $lineplot2->SetColor("mediumblue");
        $lineplot2->SetWeight(2);

        $graph->yaxis->title->Set("Usage in %");
        $lineplot1->SetLegend("CPU");
```

```
        $lineplot2->SetLegend("RAM");
        $graph->legend->SetLayout(LEGEND_HOR);
        $graph->legend->Pos(0.4,0.95,"center","bottom");
        $graph->xaxis->HideLabels();

        $graph->Add($lineplot1);
        $graph->Add($lineplot2);
        $graph->Stroke($picfile);
}
```

Für die Darstellung in der Weboberfläche muss der neue Graph noch in der Methode showSystem($id) hinzugefügt werden:

```
$sumPic = $this->sumPic($this->soap->getSystemCpuHistory($id), $this->
    soap->getSystemRamHistory($id), "images/".session_id()."-".$id."-
    systemSumGraph.png");
$content = $content."<img src='images/".session_id()."-".$id."-
    systemSumGraph.png#".time()."'/>";
```

/opt/vlmanagement/Code/src/WebUI/protected/modules/soap.php

Zwei neue Methoden für die aktuelle Auslastung und für die History des Arbeitsspeichers kommen hinzu:

```
public function getSystemRamHistory($systemId){
        return (array) $this->client->getSystemRamHistory($systemId)->
            item;
}

public function getSystemRamUsage($sysId){
        return $this->client->getSystemRamUsage($sysId);
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/access/SoapAdapter.java

Zwei neue Methoden für die aktuelle Auslastung und für die History des Arbeitsspeichers kommen hinzu:

```
@WebMethod(operationName="getSystemRamHistory")
public int[] getSystemRamHistory(int systemId){
        return this.am.pi.getSystemRamHistory(systemId);
}

@WebMethod(operationName="getSystemRamUsage")
public int getSystemRamUsage(int systemId){
        return this.am.pi.getSystemRamUsage(systemId);
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/application/PlatformInterface.java

Zwei neue Methoden für die aktuelle Auslastung und für die History des Arbeitsspeichers kommen hinzu:

```
public int[] getSystemRamHistory(int systemId){
        return this.cm.getSystemRamHistory(systemId);
}

public int getSystemRamUsage(int systemId){
        return this.cm.getSystemRamUsage(systemId);
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/application/ConfigurationManager.java

Zwei neue Methoden für die aktuelle Auslastung und für die History des Arbeitsspeichers kommen hinzu:

```
public int[] getSystemRamHistory(int systemId){
        return this.am.getInformationManagement().getSystemRamHistory(
            systemId);
}

public int getSystemRamUsage(int systemId){
        return this.am.getInformationManagement().getSystemRamUsage(
            systemId);
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/information/InformationManagement.java

Zwei neue Methoden für die aktuelle Auslastung und für die History des Arbeitsspeichers kommen hinzu:

```
public int[] getSystemRamHistory(int systemId){
        Calendar cal = new GregorianCalendar(TimeZone.getTimeZone("ECT")
            );
        int now = (int) java.lang.System.currentTimeMillis();
        mib.VLSystem s = (mib.VLSystem) this.elements.get(systemId);
        Collection c = s.getStat(now-60000, now).values();
        int[] list = new int[c.size()];
        int i=0;
        for(Object o: c){
                list[i] = ((SystemData) o).getRamUsage();
    i++;
        }
        return list;
}
```

```
public int getSystemRamUsage(int systemId){
        mib.VLSystem sys = (mib.VLSystem) this.elements.get(systemId);
        SystemData sd = (SystemData) sys.getLastStat();
        return sd.getRamUsage();
}
```

/opt/vlmanagement/Code/src/WebUI/protected/modules/vlmib/HostPerf.php

Auch hier muss die Arbeitsspeicherauslastung hinzugefügt werden. Folgende Zeile wird eingefügt:

```
public $ramUsage;
```

/opt/vlmanagement/Code/src/WebUI/wsdl.xml

Hier kommen die Messages für die Verwaltung der Arbeitsspeicher-Informationen hinzu:

```
<message name="getSystemRamUsage">
        <part name="arg0" type="xsd:int"></part>
</message>

<operation name="getSystemRamHistory">
        <soap:operation soapAction=""></soap:operation>
        <input>
                <soap:body use="literal" namespace="http://access/">
                        </soap:body>
        </input>
        <output>
                <soap:body use="literal" namespace="http://access/">
                        </soap:body>
        </output>
</operation>

<operation name="getSystemRamUsage" parameterOrder="arg0">
        <input message="tns:getSystemRamUsage"></input>
        <output message="tns:getSystemRamUsageResponse"></output>
</operation>

<operation name="getSystemRamHistory" parameterOrder="arg0">
        <input message="tns:getSystemRamHistory"></input>
        <output message="tns:getSystemRamHistoryResponse"></output>
</operation>

<message name="getSystemRamHistoryResponse">
        <part xmlns:ns4="http://jaxb.dev.java.net/array" name="return"
            type="ns4:intArray"></part>
</message>

<message name="getSystemRamHistory">
        <part name="arg0" type="xsd:int"></part>
</message>
```

```
<operation name="getSystemRamUsage">
        <soap:operation soapAction=""></soap:operation>
        <input>
                <soap:body use="literal" namespace="http://access/">
                        </soap:body>
        </input>
        <output>
                <soap:body use="literal" namespace="http://access/">
                        </soap:body>
        </output>
</operation>

<xs:complexType name="hostPerf">
        <xs:sequence>
                <xs:element name="cpuUsage" type="xs:float">
                        </xs:element>
                <xs:element name="id" type="xs:int"></xs:element>
                <xs:element name="name" type="xs:string" minOccurs="0">
                        </xs:element>
                <xs:element name="vmCount" type="xs:int"></xs:element>
        <!-- Beginn neue Zeile -->
                <xs:element name="ramUsage" type="xs:float">
                        </xs:element>
        <!-- Ende neue Zeile -->
        </xs:sequence>
</xs:complexType>
```

A.2. Anpassungen zum Registrieren des Adapters

/opt/vlmanagement/Code/src/WebUI/protected/pages/Home.page

Erzeugen eines neuen Buttons in der Weboberfläche zum Hinzufügen des Adapters. Dazu wurde der Code für den vorhandenen Button „Dummy hinzufügen“ dupliziert (button_add_dummy) und umbenannt. Im ursprünglichen Code der Weboberfläche wurde die Beschriftung des Buttons („VMware hinzufügen“) hinzugefügt.

/opt/vlmanagement/Code/src/WebUI/vlgui.css

Im Stylesheet für die Weboberfläche wird die Größe und Position für das Nachrichten-Feld (Style-Element #messages) angepasst: Die Positionierung ändert sich von left :380px auf left :500px und die Breite des Nachrichten-Felds ändert sich von width:500px auf width:800px.

/opt/vlmanagement/Code/src/WebUI/protected/pages/Home.php

In der Funktion button_clicked($sender, $param) wird eine neuer Case für den neu erzeugten Button eingefügt:

```
case "button_add_vmware":
        $this->soap->addVmwareHost(1,1);
        $this->CallbackClient->hide("ajax-loader");
        break;
```

Mit den Argumenten (1,1) bei $this->soap->addVmwareHost(1,1); wird die Anzahl der Hosts und der darauf laufenden virtuellen Maschinen angegeben.

/opt/vlmanagement/Code/src/Managementplattform2/src/access/SoapAdapter.java

Hier muss die neue Methode addVmwareHost hinzugefügt werden. Dazu wurde der Code der Methode addDummyHost dupliziert und entsprechend angepasst. Folgender Code wurde hinzugefügt:

```
@WebMethod(operationName="addVmwareHost")
public void addVmwareHost(int hosts, int vms){
    Logger.getLogger("").log(Level.INFO, "addVmwareHost(" +
                    String.valueOf(hosts) + "," + String.valueOf(
                        hosts) + ")");
    this.am.addVmwareHost(hosts, vms);
}
```

/opt/vlmanagement/Code/src/WebUI/protected/modules/soap.php

Hier muss die neue Methode ebenfalls hinzugefügt werden. Dazu wurden folgende Zeilen eingefügt:

```
public function addVmwareHost($hosts, $vms){
   $this->client->addVmwareHost($hosts, $vms);
 }
```

/opt/vlmanagement/Code/src/Managementplattform2/src/message/add/AddVmwareHost.java

Es muss eine neue Klasse erzeugt werden, die Nachrichten vom Typ AddVmwareHost für die Plattform erzeugt:

```
package message.add;
public class AddVmwareHost extends AddMessage{
    private int hostCount;
    private int maxVmsPerHost;

    public AddVmwareHost(int hosts, int vms){
        super();
        this.hostCount = hosts;
        this.maxVmsPerHost = vms;
    }

    public int getHostCount(){
        return this.hostCount;
    }

    public int getVmsPerHost(){
        return this.maxVmsPerHost;
    }
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/communication/Communicationmanagement.java

Nachrichten vom Typ AddVmwareHost müssen von der Plattform erkannt werden und es muss der Adapter gestartet werden. Dazu muss die Datei durch Einfügen eines zusätzlichen Falls in die Methode processAddMessage geändert werden. Folgender Code wird eingefügt:

```
else if(m instanceof AddVmwareHost){
    AddVmwareHost nm = (AddVmwareHost) m;
    Vmwareadapter da;
    Thread a = new Thread(da = new Vmwareadapter(this, nm.getHostCount()
        , nm.getVmsPerHost()));
    this.resourceAdapters.add(da);
    a.start();
}
```

/opt/vlmanagement/Code/src/Managementplattform2/src/application/PlatformInterface.java

Es wird zusätzlich folgende Methode eingefügt:

```
public void addVmwareHost(int hosts, int vms){
    this.cm.addVmwareHost(hosts, vms);
    }
```

/opt/vlmanagement/Code/src/Managementplattform2/src/application/ConfigurationManager.java

Es wird zusätzlich folgende Methode eingefügt:

```
public void addVmwareHost(int hosts, int vms){
    this.am.receiveFromApp(new AddVmwareHost(hosts, vms));
    }
```

/opt/vlmanagement/Code/src/Managementplattform2/src/access/AccessManager.java

Es wird zusätzlich folgende Methode eingefügt:

```
public void addVmwareHost(int hosts, int vms){
    this.pi.addVmwareHost(hosts, vms);
    }
```

/opt/vlmanagement/Code/src/WebUI/wsdl.xml

Auch hier müssen die neuen Messages hinterlegt werden. Dies geschieht durch Einfügen der folgenden Zeilen:

```
<message name="addVmwareHost">
        <part name="arg0" type="xsd:int"></part>
        <part name="arg1" type="xsd:int"></part>
</message>

<message name="addVmwareHostResponse"></message>

<operation name="addVmwareHost" parameterOrder="arg0 arg1">
        <input message="tns:addVmwareHost"></input>
        <output message="tns:addVmwareHostResponse"></output>
</operation>

<operation name="addVmwareHost">
        <soap:operation soapAction=""></soap:operation>
        <input>
                <soap:body use="literal" namespace="http://access/">
                </soap:body>
        </input>
        <output>
                <soap:body use="literal" namespace="http://access/">
                </soap:body>
        </output>
</operation>
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Bit08] Bittner, Florian: *Eine erweiterbare Managementplattform für Hostvirualisierungslösungen*, Oktober 2008.

[Con09] Consulting, Aditus: *JpGraph - PHP Graph Creating Library*, 2009. `http://www.aditus.nu/jpgraph/index.php`.

[HAN99] Hegering, Heinz-Gerd, Sebastian Abeck und Bernhard Neumair: *Integriertes Management vernetzter Systeme – Konzepte, Architekturen und deren betrieblicher Einsatz*. dpunkt Verlag, Heidelberg, 1999.

[Hie09] Hietaniemi, Jarkko: *CPAN Comprehensive Perl Archive Network Website*, 2009. `http://www.cpan.org/`.

[Mic09] Michelmann, Marcel: *Entwicklung von Adaptern zum Management von Hostvirtualisierungslösungen*, Juli 2009.

[vJAC09] Java API Community, VMware VI (vSphere): *VMware VI (vSphere) Java API FAQ*, 2009. `http://vijava.sourceforge.net/faq.php`.

[VMw06] VMware, Inc.: *Virtualization Overview*, 2006. `http://www.vmware.com/pdf/virtualization.pdf`.

[VMw08a] VMware, Inc.: *Developers Setup Guide, VMware Infrastructure SDK 2.5*, 2008. `http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/visdk25setupguide.pdf`.

[VMw08b] VMware, Inc.: *Installation Guide, VMware Infrastructure Perl Toolkit 1.6*, 2008. `http://www.vmware.com/support/developer/viperltoolkit/viperl16/doc/viperl_install.pdf`.

[VMw08c] VMware, Inc.: *Programming Guide, VMware Infrastructure Perl Toolkit 1.6*, 2008. `http://www.vmware.com/support/developer/viperltoolkit/viperl16/doc/viperl_proggd.pdf`.

[VMw08d] VMware, Inc.: *Programming Guide, VMware Infrastructure SDK 2.5*, 2008. `http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/visdk25programmingguide.pdf`.

[VMw08e] VMware, Inc.: *VI Perl Toolkit Utility Applications Reference*, 2008. `http://www.vmware.com/support/developer/viperltoolkit/viperl16/doc/perl_toolkit_utilities_idx.html`.

[VMw08f] VMware, Inc.: *VMware Developer Forums - vSphere SDK for Perl*, 2008. `http://communities.vmware.com/community/developer/vsphere_sdk_perl`.

[VMw08g] VMware, Inc.: *VMware Infrastructure (VI) API Reference Documentation*, 2008. http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/ReferenceGuide/index.html.

[VMw08h] VMware, Inc.: *VMware SDK Documentation Resources*, 2008. http://www.vmware.com/support/developer/vc-sdk/.

[VMw08i] VMware, Inc.: *vSphere SDK for Perl*, 2008. http://www.vmware.com/support/developer/viperltoolkit/.

[VMw09a] VMware, Inc.: *CIM SMASH/Server Management API Programming Guide*, 2009. http://www.vmware.com/support/developer/cim-sdk/smash/u2/ga/CIM_SMASH_PG.pdf.

[VMw09b] VMware, Inc.: *VMware CIM SMASH/Server Management API Support Resources*, 2009. http://www.vmware.com/support/developer/cim-sdk/smash/u2/ga/index.html.

[VMw09c] VMware, Inc.: *VMware Deverloper Community Website*, 2009. http://communities.vmware.com/community/developer/.

[VMw09d] VMware, Inc.: *VMware ESX 3.5*, 2009. http://www.vmware.com/files/de/pdf/esx_datasheet_de.pdf.