인터페이스 부하방지를 위한 전가상화 기반 지능형 빌딩 시스템 구현

김오범*, 정광식, 손진곤**
한국방송통신대학교 평생대학원 정보과학과
ohbeomkim@gmail.com*, {kchung0825, jgshon}@knou.ac.kr

An Implementation of Full Virtualization based Intelligent Building System for Interface Overload Prevention

Oh Beom Kim*, Kwang Sik Chung, Jin Gon Shon
Dept of Computer Science, Graduate School,
Korea National Open University

요 약

지능형 빌딩 시스템은 빌딩을 운영하는 다양한 단위시스템 정보를 통합하여 빌딩 거주자에게 쾌적하고, 안전한 생활을 할 수 있도록 운영하는 시스템을 말한다. 하드웨어적인 방법은 단위시스템에 비례하게 비용이 증가하는 단점이 있다. 소프트웨어적인 방법은 하나의 서버에서 정보를 수집하기 때문에 단위시스템에 통신량 증가로 인한 오류가 발생하면 전체 시스템에 영향을 미치는 단점을 가지고 있다. 본 논문에서는 소프트웨어적인 방법의 단점을 개선하기 위해서 부하 및 오류를 격리화할 수 있는 가상화를 이용하여 통합 관리 시스템과 인터페이스 관리 시스템으로 운영한다. 시스템의 구조는 통합 관리 모듈, 가상 관리 모듈, 구성 관리 모듈, 그리고 단위시스템 관리 모듈로 분리되며 제안 시스템을 사용함으로써 인터페이스 관리 시스템에서 발생하는 오류 및 부하로 인한 전체 시스템 오류를 축소시킬 수 있다.

1. 서론

지능형 빌딩 시스템은 단순한 건축물의 공간 활용의 개념이 아닌 첨단 정보통신 서비스를 활용하여 쾌적한 환경을 제공함으로써 향후 변화대응에 유연성이 유지되어 고도사회의 요구에도 부흥할 수 있는 빌딩을 말한다. 이러한 지능형 빌딩 시스템을 운영하기 위해서는 빌딩에 설치된 단위시스템 정보를 하나의 통합된 정보로 구성하는 것이 매우 중요하다. 하지만 단위시스템은 각 설치된 기계를 제어하기 위해서 각 제조사마다 제어에 용의한 비개방형 프로토콜을 사용한다. 그렇기 때문에 각 단위시스템 간에 정보를 교환하기 쉽지 않은 문제점이 있다. 이러한 문제를 해결하기 위해서 BACnet, LonWorks, Modbus 등의 표준화된 개방형 프로토콜을 혼용하여 사용한다. 하지만 개방형 프로토콜은 관제점과 관련한 데이터만을 처리하므로 관제점이 아닌 데이터를 처리하기 힘들다. 그렇기 때문에 시스템간의 유기적으로 연동할 수 있는 지능형 빌딩 시스템을 구축하는 것이 효율적이다[1]. 단위시스템 정보를 통합하기 위해서는 하드웨어적인 구성 방법과 소프트웨어적 구성 방법이 있다. 하드웨어적 구성 방법은 단위시스템 정보를 수집하는 중간 Gateway를 두어 정보를 수집하고 수집된 정보를 메인서버에서 관리한다. 하드웨어적인 구성 방법의 장점은 중간 Gateway를 사용하기 때문에 정보수집에 부하를 축소할 수 있다. 단점은 통합 되어 구성된 정보량에 따라서 중간 Gateway의 수가 증가하기 때문에 발생하는 비용적인 문제와 관제점이 아닌 데이터를 처리하기 힘든 문제가 있다. 소프트웨어적인 구성 방법은 하드웨어적으로 구성하는 중간 Gateway를 하나의 메인서버에서 DLL 및 EXE로 대체하여 처리하는 방법을 말한다. 장점은 Gateway가 하드웨어가 아닌 소프트웨어로 구성하기 때문에 정보수집에 유연하게 대응할 수 있고 비용이 감소한다. 단점은 하나의 메인서버에서 정보수집과 단위시스템 통신을 하기 때문에 오류 및 부하가 발생하면 시스템 전체에 영향을 줄 수도 있다. 소프트웨어적 구성 방법은 서버에서 대량의 데이터를 통합할 때 발생하는 서버 부하 문제를 해결하기 위해서 물리적으로 단위시스템과 통신하는 인터페이스 모듈을 별도로 운영한다. 인터페이스 모듈을 분리 운영하게 되면 서버의 부하 및 오류를 격리화 할 수 있는 장점이 있지만 하드웨어 추가 비용및 관리 비용이 발생하는 단점이 있다. 본 논문에서는 소프트웨어적인 구성 방법에 단점을 보안할 수 있는 VIBS(Virtualization Intelligent Building System)을 제안한다. VIBS는 하드웨어적인 장점과 소프트웨어적인 장점을 동시에 사용할 수 있는 구조로 설계되어 있다. 하드웨어적인 장점인 오류 및 부하 격리화를 사용하기 위해 서버에서 발생하는 유휴자원을 통해 하드웨어적인 효과를 낼 수 있는 전가상화를 구성한다. 가상화를 구성하면 메인 운영체제에서 별도의

** Corresponding Author

운영체제를 관리할 수 있기 때문에 두개의 운영체제를 사용하여 통합 관리 시스템과 인터페이스 관리 시스템으로 분리 운영 한다. 인터페이스 관리 시스템을 분리 운영할 수 있기 때문에 인터페이스에서 발생하는 오류 및 부하로 인해 전체 시스템에 영향을 주지 않는다. VIBS는 가상화를 효율적으로 사용할 수 있는 모듈을 제공하여 두개로 분리된 시스템을 효과적으로 사용할 수 있게 설계되었다. 또한 서버에서 발생하는 유휴 자원을 사용하기 때문에 가용된 자원을 효율적으로 사용할 수 있는 장점을 가지고 있다.

2. 관련 연구

2.1 개방형 프로토콜

개방형 프로토콜은 지능형 빌딩 시스템을 운영하는 단위시스템 정보를 표준화하여 통신할 수 있도록 제공하는 표준화된 프로토콜을 말한다. 개방형 프로토콜의 대표적인 표준은 BACnet과 Modbus가 있다. BACnet은 물리 계층, 데이터 링크 계층, 네트워크 계층 및 응용 계층의 4계층의 구조를 지닌다. BACnet 응용 계층에서는 서로 다른 네트워크 장비간의 상호 운용성을 보장하기 위하여 빌딩 자동화 설비들 간에 교환되어야 할 다양한 정보들을 객체와 프로퍼티들로 표현하며, 이러한 객체와 프로퍼티들로 표현되는 정보들을 처리하기 위해 요구되는 다양한 기능들을 응용 계층 서비스를 정의하고 있다[2][3]. Modbus는 수많은 산업용 기기에 사용되는 표준통신 프로토콜 가운데 하나로서, RTU, IED와 감시/계측/제어 총괄시스템 사이의 상호 운영성을 확립하기 위하여 개발된 프로토콜이다[4][5].

2.2 가상화

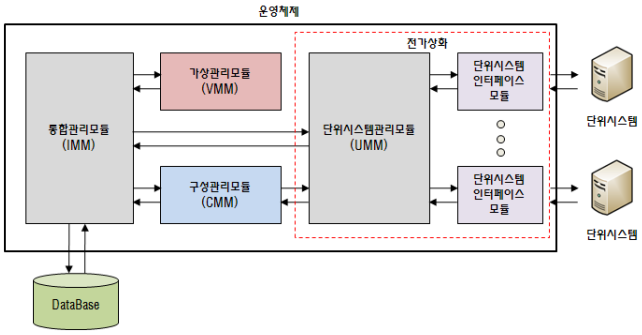
가상화 기술은 운영체제(OS)에 한정된 자원을 더 효율적으로 사용하기 위해서 발전한 기술로 VMM(Virtual Machine Monitor)을 통해서 실제 물리적인 컴퓨팅 자원과 사용자의 작업 영역을 분리하여 사용 할 수 있도록 하는 기술이다. 가상화 기술은 크게 전가상화(full-virtualization), 반가상화(para-virtualization)로 나눌 수 있다. 전가상화는 하드웨어를 시뮬레이션한 결과로 가상화를 하기 때문에 설치가 간편하며 사용하기 편리하다. 하지만 하드웨어 자체를 시뮬레이션하여 가상화를 진행하기 때문에 유휴자원 문제를 완전하게 해결하지는 못한다. 반가상화는 가상코드를 사용하여 가상화를 진행하기 때문에 CPU, 메모리를 자유롭게 가상화할 수 있다. 하지만 가상코드로 가상화를 진행하기 때문에 운영체제 자체를 재컴파일 해야 한다. 또한 가상화 구성이 복잡하고 특정 하드웨어 자원에서만 가상화를 지원한다[6][7].

3. VIBS 구조 및 구현

본 논문에서 제안하는 시스템은 지능형 빌딩 시스템을 효율적으로 운영하기 위한 가상화 기반 지능형 빌딩 시스템(VIBS)이다. VIBS는 서버에서 발생하는 유휴자원을 사용하여 전가상화를 구성하고 단위시스템 정보를 관리하는 모듈과 단위시스템과 통신하는 모듈을 분리할 수 있는 환경에서 실행된다.

3.1 VIBS 구조

VIBS 구조(그림3-1)는 단위시스템 정보를 통합적으로 관리하는 통합 관리 모듈(IMM), 전가상 환경을 자동적으로 통합 관리 모듈에서 이용할 수 있는 가상 관리 모듈(VMM), 단위시스템과 통신하는 인터페이스 모듈을 관리하는 단위시스템 관리 모듈(UMM), 그리고 마지막으로 통합 관리 모듈과 단위시스템 관리 모듈 간에 정보를 교환할 수 있는 구성 관리 모듈(CMM)의 구조로 구성되어 있다.



(그림 3-1) VIBS 시스템 구조

3.1.1 통합 관리 모듈(IMM)

통합 관리 모듈은 단위시스템 정보를 통합하여 운영할 수 있도록 제공하는 모듈이다. 통합 관리 모듈의 구조는 클라이언트 관리, 단위시스템 구성 관리, 연동 관리, 단위시스템 정보 관리로 구분되어 있다. 지능형 빌딩시스템을 관리하는 운영자는 단위시스템 정보를 운영자가 원하는 형태로 그래픽으로 구성하여 서버와 통신을 통해서 정보를 확인할 수 있다. 또한 단위시스템에서 발생하는 비상알람 정보를 그래픽을 통해 확인할 수 있다.

클라이언트 관리는 서버와 메시지를 통신하는 클라이언트 정보를 관리한다. 서버에 접속한 클라이언트는 서버에 단위시스템 정보를 요청할 권한 여부를 확인 받아야 한다. 클라이언트가 권한이 없는 사용자일 경우는 접속을 거부하고 사용 권한이 있을 경우는 사용 승인 메시지를 송신한다. 승인된 클라이언트는 단위시스템 정보를 설정된 초단위로 서버에 요청하게 된다. 요청된 단위시스템 정보는 단위시스템 정보 관리를 통해서 가공하여 요청한 클라이언트에 송신하게 된다.

단위시스템 구성 관리는 단위시스템과 통신하는 인터페이스 정보를 XML형태로 구성하여 관리한다. XML구성은 인터페이스 간에 통신을 하기 위한 고유 ID, 인터페이스 프로토콜종류, 통신 프로토콜 타입, 통신 요청 주기, 이벤트 저장 폴더 위치를 저장하여 CMM을 통해 UMM에 업로드 된다.

단위시스템 정보관리는 DataBase, File, XML 등 다양한 오브젝트 형태로 구성되는 단위시스템 정보를 하나의 통합된 구조로 변경하여 관리 한다. 단위시스템 정보관리는 서버 번호, 시스템 번호, 통신 디바이스 번호, 카운팅 번호를 조합하여 하나의 유일한 ID로 가공하여 단위시스

템 오브젝트 설정 값과 함께 DataBase에 저장된다. 최초 서버 구동시 DataBase에 저장된 단위시스템 정보를 선택하여 테이블 구조로 단위시스템 정보를 구조화여 저장한다. UMM으로부터 단위시스템 상태정보가 수신되면 구성된 단위시스템 테이블 정보에 상태 값을 저장한다.

연동 관리는 지능형빌딩에 거주하는 사람들에 쾌적한 환경을 만들어주기 위해서 단위시스템 간에 연동을 관리한다. 단위시스템 정보들은 서로 다르게 운영되기 때문에 한쪽에서 비상알람 발생할 때 다른 단위시스템에 연동하기 힘든 구조로 되어 있다. 이러한 연동하기 힘든 부분을 소프트웨어적으로 연동 로직을 구성하여 동작하는 모듈이다.

3.1.2 가상 관리 모듈(VMM)

전가상화를 이용하여 분리시킨 Guest OS는 독립적인 하나의 운영체제로 운영되기 때문에 메인 운영체제에서 수동적으로 사용자가 관리해야 한다. 사용자가 수동으로 관리하게 되면 각 운영체제에 설치된 IMM과 UMM간에 동기화를 진행할 수 없기 때문에 전체 시스템에 문제를 일으킬 수 있다. 이러한 문제를 해결해주는 것이 VMM이다. VMWare를 사용하여 가상화를 구성하면 가상화 정보를 파일로 변경하여 하드에 저장된다. 저장된 가상파일은 사용자가 선택하여 가상화를 실행할 수 있다. VMWare는 프로세스에서 가상화를 관리할 수 있도록 API를 제공한다. VMM은 VMWare에서 제공하는 API를 사용하여 Guest OS를 관리한다. API를 사용하게 되면 가상파일을 프로세스가 점유하여 컨트롤이 가능하기 때문에 자동으로 가상파일을 ON, OFF할 수 있다. 또한 Guest OS에 설치된 실행 파일들을 자유롭게 관리할 수 있다.

3.1.3 구성 관리 모듈(CMM)

구성 관리 모듈은 IMM과 UMM간에 표준화된 XML 파일을 업로드 및 다운로드하는 서비스를 제공한다. XML은 데이터를 일반 텍스트 형식으로 저장하는 수단이며 따라서 어떤 컴퓨터에서도 데이터를 읽는 것이 가능하다. UMM에서 실행되는 단위시스템과 통신하는 인터페이스 모듈은 다양한 프로토콜을 관리해야 하기 때문에 프로토콜 구성할 때 개발 언어가 변경 될 수 있다. 이때 특정 파일에 정보를 저장하게 되면 표준화하기 어렵기 때문에 표준화된 XML을 사용하여 파일 폴더에 저장한다.

3.1.4 단위시스템 관리 모듈(UMM)

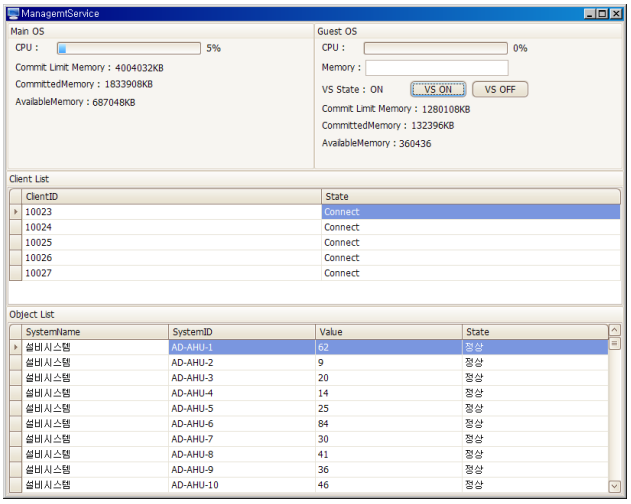
단위시스템 관리 모듈은 단위시스템과 인터페이스하기 위해서 개발된 통신 모듈을 관리하는 일을 수행한다. 인터페이스를 구성하기 위해서는 단일 프로그램으로 구현하거나 DLL로 구성하여 메인 프로그램에서 로드하여 사용한다. VIBS는 가상화를 사용하여 인터페이스를 관리하기 때문에 EXE 및 DLL를 사용할 경우 관리가 어렵다. 또한 EXE및 DLL에서 오류가 발생할 때 인터페이스 자체적으로 로그는 정의하지만 운영체제 오류로 종료되었을 경우는 확인할 수가 없다. 이러한 문제를 해결하고 관리의 효율을 높이기 위해서 UGS에서는 통신 모듈을 Windows 서비스로 개발하여 운영체제가 오류 상태를 로그 테이블에 저장할 수 있도록 구성관리 한다.

3.2 VIBS 구현

VIBS는 크게 메인 서버와 인터페이스 서버로 분리될 수 있다. 메인 서버는 IMM, CMM, VMM을 사용하여 연결된 클라이언트를 관리한다. 인터페이스서버는 단위시스템과 통신하는 UMM을 사용하여 인터페이스를 관리한다. 개발에 사용된 언어 및 API는 .NET 2.0 C#, VMWare 6.5, VMWare API 1.6.2이다.

3.2.1 메인 서버 구현 화면

메인 서버는 인터페이스 서버 및 클라이언트를 관리한다. (그림 3-2)는 메인 서버 구현 화면이다. 초기에 실행할 때 vmx파일을 로드하여 가상화를 구동시킨다. 구동 후 로그인을 통하여 인터페이스 서버를 구동하여 Guest OS를 감시한다.

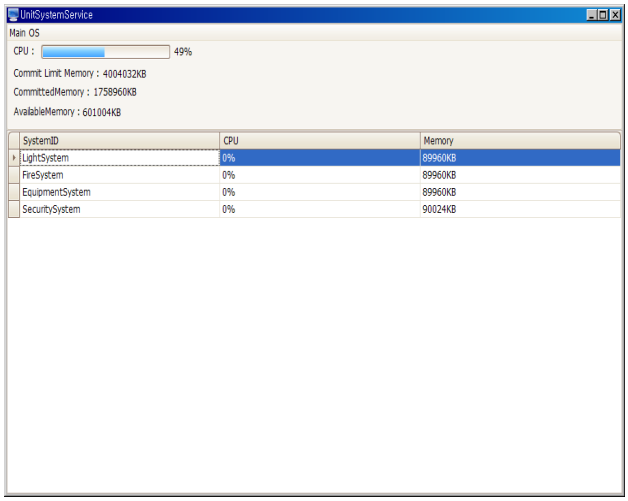


(그림 3-2) 메인 서버 구현 화면

메인 서버인 Main OS 상태 값은 현재 실행 중인 운영체제의 CPU 사용량, 한도 메모리, 전체 메모리, 사용 메모리를 감시한다. 감시된 결과는 KB단위로 표출된다. 메인 서버와 연결된 인터페이스 서버는 Guest OS의 상태를 감시하여 메인 서버에서 보이는 동일한 정보를 활용하여 Guest OS를 감시한다. 또한 가상 환경으로 구성된 Guest OS를 VMM을 이용하여 On, Off를 자유롭게 조정한다. 만약 인터페이스 서버 부하량이 일정 간격으로 증가하게 되면 인터페이스 서버를 다시 시작하여 전체 부하를 축소시킨다. 또한 메인 서버에 연결된 클라이언트 서버에 접속하기 위해서는 초기 통신 패킷을 정확하게 일치해야 연결을 시키고 아닐 경우는 접속을 해제한다. 메인 서버는 DataBase에서 저장된 단위시스템 오브젝트 정보를 테이블 화하여 가상 메모리에 가지고 있고, 인터페이스 서버에서 단위시스템 정보가 수신되면 단위시스템 오브젝트를 리스트 화하여 요청 클라이언트에 송신한다.

3.2.2 인터페이스 서버 구현 화면

인터페이스 서버는 가상화된 Guest OS에 설치된 단위 시스템과 통신하는 인터페이스를 관리한다. (그림 3-3)는 인터페이스 서버 구현 화면이다.



(그림 3-3) 인터페이스 서버 구현 화면

인터페이스 서버는 현재 Guest OS의 CPU 및 Memory상태를 체크하여 메인 서버에 송신한다. 서버는 단위시스템과 통신하기 위한 인터페이스 서비스를 관리한다. 서비스 관리 대상 목록을 현재 실행되고 있는 서비스의 CPU 사용률 및 Memory 사용률을 주기적으로 확인하여 리스트에 표출한다. CPU 사용률이 일정 시간 동안 지속적으로 증가할 경우는 판단하여 실행되고 있는 서비스를 다시 시작한다. 이와 같이 인터페이스 부하를 감시하여 시스템 전체에 발생하는 부하 문제를 해결한다.

4 성능 평가

성능 평가는 소프트웨어적인 방법에서 가장 많이 발생하는 오류에 따른 서버 부하를 측정하였다. 성능 평가에 사용된 자원은 Duo CPU E8400 @3.00Hz, Memory 2GB이다. 소프트웨어적인 지능형 빌딩 시스템을 구성할 경우 가장 많이 발생하는 것이 통신에서 발생할 수 있는 메모리 누수 오류이다. 메모리 누수 오류는 프로세스에 할당된 2GB의 가상메모리 이상을 사용할 경우 프로세스가 종료되는 것을 말한다. 측정 방법은 1초 간격으로 10240Bytes의 메모리를 누적하는 인터페이스 모듈을 각 DLL, EXE, VIBS에서 실행하여 메모리 누수 오류가 프로세스에 미치는 영향을 테스트 하였다. <표4-1> 성능평가 테이블이다.

<표 4-1> 메모리 누수 오류 테스트

	DLL	EXE	VIBS
전체 할당 메모리	증가	증가	동일
최고 할당 메모리	누적	누적	동일
사용가능 전체 메모리	축소	축소	동일
인터페이스 모듈 가상 메모리	증가	증가	증가
메인 모듈 상태	종료	종료	동일

테스트에서 DLL 및 EXE에서 누수가 발생할 경우 프로세스 및 메인 모듈에 영향을 준다. 그러나 제안 시스템은 메모리는 누적되어 증가하나 메인 모듈에는 영향을 주지 않는다. 메모리 누수로 인해서 오류가 지속적으로 발생할 경우는 최고할당 메모리가 누적되어 한도 메모리 보다 클 경우 운영체제에 문제가 발생한다. 이러한 문제를 해결하는 방법은 운영체제를 다시 시작하는 것인데 하나의 운영체제만을 사용할 경우 다시 시작하게 되면 메인 모듈도 종료되는 단점을 있는데 반해 VIBS는 가상화만을 다시 시작하면 되기 때문에 메인 모듈에는 아무런 영향을 주지 않는다. 이처럼 VIBS는 메모리 오류가 발생할 때에는 유연하게 대처할 수 있는 장점을 가지고 있다.

5 결론

본 논문에서는 지능형 빌딩 시스템을 구성하는 소프트웨어적인 방법을 보완하는 시스템을 구현하였다. 지능형 빌딩 시스템을 소프트웨어적으로 구성하기 위해서는 메인 모듈과 인터페이스 모듈로 구분하는데 단위시스템과 통신하는 인터페이스 모듈을 DLL 및 EXE로 구성하는 것이 가장 많이 사용된다. 인터페이스 모듈을 DLL 및 EXE로 구성하여 운영할 경우 단일 운영체제에서 운영되기 때문에 인터페이스 모듈에서 발생할 수 있는 오류가 메인 모듈에 영향을 줄 수 있다. VIBS는 이러한 문제를 해결하기 위해서 가상화를 사용하여 모듈을 분리 운영하기 때문에 하드웨어적으로 인터페이스 모듈을 분리 운영했을 때와 같은 오류를 격리화하는 장점을 가지고 있다. 향후 연구에서는 가상화를 직접 관리하기 때문에 발생하는 메모리 사용량의 증가를 축소할 수 있는 경량화된 시스템 및 지능형 빌딩 시스템이 다수의 군으로 구성될 경우 발생하는 복잡성을 축소할 수 있는 VIBS 프레임워크를 연구할 계획이다.

참고 문헌

[1] 최경재, “초고층 빌딩 적용 통합자동제어 솔루션”, 대한설비공학회, 대한설비공학회 2009년 하계학술발표대회, 2009. 6, pp.1020-1024

[2] 박태진, “BACnet을 위한 네트워크 관리시스템의 설계 및 구현”, 대한전기학회, 전기학회논문지, 제58권 제11호, 2009. 11, pp.2253-2260

[3] http://www.bacnet.org/WG/index.html

[4] 송재구, 정성모, 김석수, 김태훈, 강동주, 김석주, “Modbus 기반 SCADA 해킹 테스트 시스템 설계”, 한국정보기술학회, 한국정보기술학회논문지, 제7권 제5호 2009. 10, pp.183-190

[5] Introduction to MODBUS, Technical Tutorial, Dec, 2002

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Wareld, ″Xen and the Art of Virtualization″, ACM in SOSP′03, 2003, pp 164-175

[7] VMWare, Inc. ″Understanding Full Virtualization, Paravirtualization, and Hardware Assist″, http://www.vmare.com, 2007