碩 士 學 位 論 文

차량용 인포테인먼트를 위한 경량 가상화 플랫폼

高麗大學校 融合소프트웨어專門大學院

임베디드소프트웨어學科

朴 惠 貞

2014 年 12 月

柳 燁 敎 授 指 導

碩 士 學 位 論 文

차량용 인포테인먼트를 위한 경량 가상화 플랫폼

이 論文을 工學 碩士學位 論文으로 提出함

2014 年 12 月

高麗大學校 融合소프트웨어專門大學院
임베디드소프트웨어學科

朴 惠 貞 (印)

朴惠貞의 工學 碩士學位 論文

審査를 完了함

2014 年 12 月

委員長 유 혁 (印)

委 員 최 진 영 (印)

委 員 민 성 기 (印)

요 약

최근 차량용 인포테인먼트 시스템은 스마트 폰, 태플릿 PC와 같은 개인 기기의 대중화와 인터넷에 연결된 커넥티드 카의 등장에 따라 제공하는 서비스를 다양화하고 있다. 그러나 이와 같은 서비스의 확장은 인포테인먼트 시스템을 불안정하고 보안에 취약한 시스템으로 만든다. 가상화는 가상 머신에 고립성을 만족하는 실행환경을 제공하기 때문에 신뢰성을 요구하는 자동차 시스템에서 인포테인먼트를 안전하게 통합하기 위해 이용될 수 있다. 그러나 가상화 플랫폼 구현을 위해 일반적으로 사용되는 하이퍼바이저 방식은 소프트웨어 계층의 추가로 인한 가상화 오버헤드를 갖는다. 이러한 오버헤드는 가상화 플랫폼에서 동작하는 자동차 소프트웨어의 실시간성을 훼손할 수 있기 때문에 차량용 가상화 플랫폼은 시스템에 고립성뿐만 아니라 경량성을 제공할 수 있도록 설계되어야 한다. 본 논문에서는 물리적 가상화 방식을 이용하여 경량성을 갖춘 차량용 가상화 플랫폼인 LET-V를 제시한다. 또한 하드웨어 지원 하이퍼바이저 방식인 Xen/ARM과 LET-V의 가상화 오버헤드를 비교하여 LET-V가 CPU와 I/O 연산에서 경량성을 가지는 것을 보인다.

목 차

그림 목차

표 목차

1. 서론

1.1 연구배경

오늘날 대부분의 자동차들은 인포테인먼트(Infotainment) 시스템을 포함하고 있다. 인포테인먼트 시스템은 운전자와 탑승자의 편의를 위하여 네비게이션, DVD 플레이어, 라디오, 블루투스 등의 서비스를 제공한다. 이에 더하여 최근 스마트폰이나 태블릿 PC와 같은 개인 기기의 대중화와 인터넷에 연결된 커넥티드 카의 등장은 인포테인먼트 시스템에서 제공하는 서비스를 다양화하고 있다[1]. 자동차에 장착 된 PC를 통한 각종 정보 검색 서비스, 음성으로 차량 내 기기를 제어하는 커맨드 서비스 또는 사용자 기기와 연계한 원격 차량 진단 및 제어 서비스 등이 인포테인먼트 시스템에서 제공 될 수 있다.

그러나 이와 같은 서비스의 확장은 인포테인먼트 시스템을 불안정하고 보안에 취약한 시스템으로 만든다. 다수의 I/O 장치나 잦은 사용자 대화로 인한 인포테인먼트 소프트웨어의 복잡성 증가는 의도치 않은 소프트웨어의 실패 가능성을 높인다. 또한 사용자 장치나 인터넷 연결을 통한 외부 데이터의 유입은 인포테인먼트 시스템을 악성코드의 유입으로 인한 악의적인 시스템 오류의 위험에 노출 시킨다. 반면 자동차 소프트웨어는 인간의 안전과 직접적으로 연결되기 때문에 다른 임베디드 시스템보다도 높은 신뢰성을 요구한다[2]. 따라서 자동차 시스템에서 확장된 인포테인먼트 서비스를 제공하기 위해서는 인포테인먼트 시스템의 오류 또는 공

격으로부터 자동차 시스템을 보호할 수 있는 신뢰성 있는 플랫폼의 제공이 필요하다.

이를 위한 새로운 자동차 소프트웨어 플랫폼으로 가상화가 제시되고 있다[3][4]. 가상화는 하나의 물리 하드웨어에서 다수의 가상 머신을 제공하며 이 위에서 실행되는 소프트웨어에 고립성을 보장한다. 따라서 가상화 플랫폼에서 인포테인먼트 시스템을 구현함으로써 신뢰성 있는 자동차 플랫폼을 제공할 수 있다.

차량 플랫폼을 위해 적용할 수 있는 가상화 기술은 크게 네 가지로 구분된다[3]. 가상화를 수행하는 VMM의 구현 위치에 따라서 물리적 가상화, 하이퍼바이저 가상화, 유저 모드 가상화로 나뉘며 하이퍼바이저 가상화는 하드웨어의 가상화 기술 지원 여부에 따라 다시 반가상화 하이퍼바이저 가상화, 하드웨어 지원 하이퍼바이저 가상화로 구분된다. 각 가상화 구현 방식에 따라 시스템에 다른 수준의 신뢰성이 제공된다. 가장 일반적으로 사용되는 방식은 하이퍼바이저 방식이다.

하이퍼바이저 방식에서 VMM은 하드웨어 바로 위에 소프트웨어로 구현되어 가상화를 위한 기능을 제공한다. VMM은 프로세서의 특권 계층에서 가상 머신의 모든 시스템 자원에 대한 접근을 관리 및 처리하기 때문에 높은 수준의 신뢰성을 제공할 수 있다. 그러나 새로운 하이퍼바이저 소프트웨어 계층의 추가로 인한 가상화 오버헤드는 일반적으로 실시간성을 요구하는 자동차 시스템에 또 다른 위협이 될 수 있다. 따라서 자동차를 위한 가상화 플랫폼은 신뢰성 있는 실행환경을 제공할 뿐만 아니라 가상화 오버헤드를 최소화하도록 설계 되어야 한다.

1.2 연구목표

본 논문에서는 차량용 인포테인먼트를 위한 경량 가상화 플랫폼으로 LET-V(Lightweight Virtualization Platform for Automotive Software)를 제안한다. LET-V는 물리적 가상화 방식으로 가상화 플랫폼을 구현하였다. 물리적 가상화 방식에서 VMM은 하드웨어에 구현된 분할(partitioning)에 기반하여 가상 머신에 고립된 실행 환경을 제공한다. 하드웨어에 의한 실행환경 분할이 올바르게 구현 되었다면 소프트웨어에 의한 고립성의 훼손이 발생하지 않으므로 물리적 가상화 방식은 하이퍼바이저 방식의 가상화에 비해 높은 신뢰성을 제공할 수 있다. 또한 물리적 가상화는 대부분의 VMM기능 구현에 하드웨어 분할을 이용하고 가상화 소프트웨어에는 최소한의 필요한 기능만을 제공하도록 한다. 따라서 하이퍼바이저 방식보다 낮은 가상화 오버헤드를 가진다.

LET-V 는 하드웨어 분할을 위해 ARM 프로세서의 TrustZone[5][6] 기술을 사용하였다. TrustZone은 ARM 프로세서가 제공하는 보안을 위한 SoC 기술로써 하나의 물리 프로세서를 분할하여 두 개의 가상 실행 공간을 제공한다. LET-V는 시큐어 월드(Secure world)와 노멀 월드(Normal world)로 불리는 각각의 실행 공간에서 게스트 운영체제가 낮은 가상화 오버헤드만으로 실행 될 수 있는 가상화 플랫폼을 구축하여 인포테인먼트와 자동차 시스템을 위한 실행 환경을 제공한다.

1.3 논문의 구성

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 ARM 환경에서 하이퍼바이저 가상화 방식으로 구현된 가상화 플랫폼을 소개한다. 3장에서는 TrustZone에서 제공하는 하드웨어 분할에 대해 설명하고 이를 이용한 LET-V의 가상화 플랫폼 디자인에 대해 설명한다. 4장에서는 실험을 통해 LET-V의 가상화로 인한 성능 오버헤드를 알아본다. 마지막으로, 5장에서는 결론을 기술한다.

2. 관련연구

이번 장에서는 ARM 프로세서에서 하이퍼바이저 가상화 방식으로 구현된 가상화 플랫폼을 소개한다. 각각의 가상화 플랫폼은 반가상화 하이퍼바이저 방식과 하드웨어 하이퍼바이저 방식으로 구현되었다.

2.1 Xen on ARM

Xen on ARM[7] 은 반가상화 하이퍼바이저 방식으로 구현된 가상화 플랫폼이다. 반가상화 하이퍼바이저 방식에서 VMM은 특권계층에서 구동하는 소프트웨어이며 하드웨어의 가상화 지원 기능을 이용하지 않기 때문에 모든 가상화 기능은 온전히 소프트웨어로 구현된다[3].

하이퍼바이저 가상화에서 게스트 운영체제는 하이퍼바이저가 하드웨어 자원에 대한 유일한 제어권을 가질 수 있도록 하기 위해 비특권 실행 모드에서 동작 되어야 한다. 따라서 게스트 운영체제는 ARM에서 제공하는 단 하나의 비특권 모드인 유저 모드(User mode)에서 유저 프로세스와 함께 실행된다. Xen on ARM 은 유저 모드에서 동작하는 게스트 운영체제를 유저 프로세스로부터 보호하기 위해 가상 특권모드, 도메인 관리 등의 기법을 도입하였다. 또한 게스트 운영체제의 올바른 실행을 위해 하이퍼바이저는 민감 명령어에 대한 처리 및 예외 처리 기능을 제공한다. 이와 같은 게스트 운영체제와 유저 프로세스를 분리하기 위한 기능 이외에도 페이지 관리 메커니즘과 같이 각 가상 머신에 고립된 실행환경을 제공하

기 위한 메커니즘이 하이퍼바이저에서 제공된다.

Xen on ARM과 같은 반가상화 하이퍼바이저 방식에서 가상화 기능은 모두 소프트웨어로 구현되어 있기 때문에 하드웨어를 이용하는 가상화 방식에 비해 가상화 오버헤드가 크다.

2.2 Xen/ARM

Xen/ARM[8]은 하드웨어 지원 하이퍼바이저 방식으로 구현된 가상화 플랫폼이다. VMM 소프트웨어를 위해 하드웨어에서 제공하는 가상화 지원 기술이 활용된다[3]. Xen/ARM에서는 VMM 소프트웨어의 구현을 위해 ARM Cortex-A15[9]부터 제공되는 가상화 확장 기능(Virtualization Extension)이 이용된다.

새롭게 도입된 하이퍼바이저 모드(hypervisor mode)에는 하이퍼바이저가 실행되어 게스트 운영체제가 유저 모드가 아닌 원래의 관리자 모드(Supervisor mode)에서 동작할 수 있도록 한다. 또한 2 단계 변환(2-stage translation)을 지원하는 MMU를 통해 가상 머신과 하이퍼바이저의 메모리 공간에 대한 보호 메커니즘을 제공한다. 마지막으로 가상 타이머(virtual timer)와 가상 GIC(virtual GIC)를 통해 가상 머신에 하이퍼바이저를 거치지 않고 제어할 수 있는 타이머와 인터럽트 CPU 인터페이스를 제공한다.

이와 같이 하드웨어에서 가상화 기술을 지원하는 경우 하이퍼바이저는 보다 간단하게 구현될 수 있으며 하드웨어에 VMM의 기능이 일부 구현 되므로 반가상화

하이퍼바이저 방식에 비해 낮은 가상화 오버헤드로 구현될 수 있다. 그러나 하드웨어의 가상화 기능을 반드시 요구하기 때문에 이식성에 문제가 있고 하드웨어의 가상화 기능이 게스트 운영체제의 실행환경을 완전하게 고립시켜 주지는 않는다.

3. 설계

이번 장에서는 LET-V의 전체 구성을 설명하고 가상화 소프트웨어인 모니터가 게스트 운영체제의 고립성을 만족시키기 위해 어떤 기능을 제공하는 지 살펴본다. 또한 TrustZone 이 지원하는 하드웨어 분할이 가상화 플랫폼을 구축하기 위해 어떻게 사용되었는지 설명한다.

3.1 TrustZone

물리적 가상화 방식으로 차량용 가상화 플랫폼을 구현하기 위해서는 하드웨어에서 실행 환경에 대한 분할 기법을 제공해야 한다. LET-V는 ARM 프로세서의 TrustZone 분할 기법에 기반하여 가상화 환경을 구축한다.

TrustZone은 ARM 프로세서의 보안 강화를 목적으로 하는 SoC 기술이다. TrustZone은 하나의 물리 프로세서에 분리된 두 개의 실행 공간을 제공하여 프로세서에서 보안을 필요로 하는 프로세스가 동작하는 경우 다른 프로세스의 실행 공간과 분리된 안전한 공간에서 프로세스가 실행되도록 한다. 이는 실패하면 안되거나 중요한 정보를 가지는 작업이 시스템의 다른 부분에 의해 영향 받지 않게 하기 위한 하드웨어적 고립 기법이다. TrustZone은 이러한 하드웨어 고립을 위해 다음과 같은 특성을 가진다.

먼저 TrustZone은 CPU를 시큐어 월드(Secure world)와 노멀 월드(Normal

world)로 불리는 두 개의 실행 환경으로 구분한다. 프로세서는 SCR(Security Control Register)의 NS(Non-secure) 비트의 값에 따라 월드를 구분한다. 또한 TrustZone의 두 월드는 ARM의 CPU 모드와 수직적이므로 두 월드는 모두 ARM 아키텍처에 의해 정의된 7개의 CPU모드(User, FIQ, IRQ, Supervisor, Abort, Undef, System)를 가진다.

두 월드의 관리를 위해 TrustZone에는 7개의 CPU 모드 외에 모니터 모드(monitor mode)라는 새로운 실행 모드가 추가되었다. 모니터 모드는 시큐어 월드에 있으며 이 모드에서 실행되는 모니터 소프트웨어는 관리자 모드(Supervisor mode)보다 높은 권한에서 실행된다.

각 월드는 모니터 권한을 필요로 하는 작업을 처리하기 위해 SMC(Secure Monitor Call)을 통해 모니터 소프트웨어에 이를 요청할 수 있다. SMC는 모니터 모드로 진입하기 위한 새로운 명령어이다. SMC는 특권 모드에서만 실행되며 만약에 특권모드에서 동작하는 소프트웨어가 SMC를 호출하면 이것은 트랩을 발생하고 모니터 모드에서 실행되는 소프트웨어에 의해 요청에 맞는 적절한 처리 과정이 수행된다.

TrustZone은 메모리 공간에 대한 분할 메커니즘도 제공한다. TZASC(TrustZone Address Space Controller) 설정을 통해 각 월드의 메모리 영역을 지정하고 접근 권한을 설정할 수 있다. 따라서 각 월드는 자신의 메모리 영역을 다른 월드로부터 보호하거나 원한다면 공유 메모리 영역으로 설정할 수도

있다.

I/O 장치 또한 월드에 따라 할당될 수 있다. TZPC(TrustZone Protection Controller)는 각 월드에 연관된 모든 I/O 장치의 소유권을 정의한다. TZPC의 값에 따라 각 장치가 할당된 월드를 알 수 있다.

마지막으로 각 월드는 인터럽트를 분할하여 받는다. FIQ 인터럽트는 시큐어 월드로 IRQ 인터럽트는 노멀 월드로 전달되며 각 인터럽트에 대한 월드 설정은 GIC(General Interrupt Controller)의 보안 비트(Security Bit) 값으로 설정할 수 있다.

3.2 LET-V 구조

그림 1은 LET-V의 구조를 보여준다. LET-V는 모니터, TZ-RTOS, TZ-GPOS, 통신 장치로 구성된다. 모니터는 시큐어 월드의 모니터 모드에서 동작하며 두 개의 게스트 운영체제를 운영하는 가상화 환경을 제공하는 데 핵심적인 역할을 수행한다. 게스트 운영체제인 TZ-RTOS와 TZ-GPOS는 TrustZone 실행 환경에 맞추어 수정된 게스트 운영체제이다. LET-V 가상화 환경에서 동작하기 위해 게스트 운영체제는 몇 가지 수정이 필요하다. 먼저 TZASC의해 구분된 월드의 메모리 영역에 맞추어 물리 메모리 맵이 수정된다. 다음으로 시큐어 월드 운영체제의 경우 FIQ를 인터럽트로 받아 처리할 수 있도록 인터럽트 핸들러가 수정된다. 마지막으로 각 운영체제에 월드 간 통신 장치가 추가된다. 월드 간 통신장치

는 두 월드의 통신을 수행하는 가상의 디바이스이다.

모니터는 분할 관리, 인터럽트 관리, 전환 관리, 통신 관리의 네 가지 기능을 갖는다. 분할 관리는 시스템 부팅 시 수행되는 월드 초기화 루틴이다. TZASC 에 의한 메모리 분할, TZPC에 의한 장치 분할, GIC 에 의한 인터럽트 분할 등에 대한 설정을 이 때 수행한다. 인터럽트 관리는 발생한 FIQ 또는 IRQ 인터럽트를 적절한 월드로 전달해 주는 역할을 한다. 전환 관리는 월드 스위칭을 위한 루틴이다. 적절한 조건 발생시 현재 월드의 컨텍스트를 저장하고 다른 월드의 컨텍스트를 복구하여 월드를 전환시키는 역할을 한다. 마지막으로 통신 관리는 월드간 통신을 위한 IPI(Inter Processor Interrupt)를 관리한다.

이어지는 장에서는 LET-V의 구성 요소와 TrustZone의 하드웨어 분할이 CPU, 메모리, I/O 에 대해 어떻게 가상화를 지원하는지 알아본다.

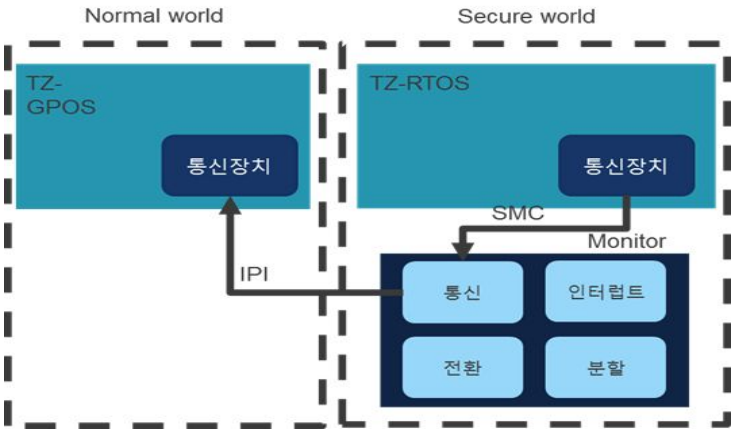

그림 1. LET-V 구조

3.3 CPU 가상화

LET-V에서 각 월드의 CPU 컨텍스트는 모니터의 전환 관리 기능과 TrustZone의 코프로세서 분할에 의해 고립된다. CPU의 모든 범용 레지스터 및 상태 레지스터는 월드 전환 시에 월드 컨텍스트를 위한 모니터의 메모리 영역에 저장 및 복원된다. 모니터의 메모리 영역은 게스트 운영체제에 의해 접근이나 변경될 수 없으므로 모니터에 의해 저장된 CPU 컨텍스트는 다음 월드 전환 시에 안전하게 복구된다. 또한 TrustZone은 시스템 제어를 위한 CP15 코프로세서에 대해 월드에 따라 별도의 레지스터 값을 유지한다. 월드가 전환될 때 코프로세서의 값도 자동으로 변하기 때문에 게스트 운영체제는 다른 월드의 코프로세서 값을 바꿀 수 없다. 따라서 게스트 운영체제에서 민감 명령어의 수행으로 프로세서의 상태가 변경되더라도 다른 월드의 게스트 운영체제의 상태에 영향을 미치지 않는다.

CPU 스케줄링은 시큐어 월드 인터럽트와 SMC에 의해 수행된다. 모니터의 전환 관리는 프로세서가 노멀 월드에 있는 동안 시큐어 월드 인터럽트인 FIQ가 발생하거나 시큐어 월드에서 SMC 호출을 통해 전환 관리에 월드 전환을 요청할 때 프로세서의 월드를 전환한다. LET-V의 이 같은 CPU 스케줄링은 시큐어 월드에 CPU자원에 대한 더 높은 우선순위를 준다.

3.4 메모리 가상화

TrustZone은 각 월드에 독립적인 가상 주소 공간을 유지할 수 있도록 한다. TTBR(Translation Table Base Register)은 MMU가 참조하는 페이지 테이블의 베이스 주소를 저장하는 CP15 레지스터이다. TrustZone에서 각 월드는 서로 다른 TTBR(Translation Table Base Register) 및 가상 메모리 설정 레지스터를 가진다. 프로세서의 월드가 바뀌면 레지스터 값들이 자동으로 전환되기 때문에 각 월드는 가상의 MMU를 할당 받은 것과 마찬가지로 가상 주소 공간을 유지할 수 있다. 이것은 하이퍼바이저 가상화 방식에서 게스트 운영체제에 가상 주소 공간을 제공하기 위해 주소의 2단계 변환을 사용하는 것 보다 효율적이다.

TrustZone은 각 월드의 메모리 영역을 서로 고립시키기 위해 TZASC를 제공한다. TZASC값을 설정하는 것으로 메모리 영역의 분할 및 접근 권한을 설정할 수 있다. 게스트 운영체제가 접근이 허용되지 않은 영역에 접근할 경우 발생하는 ABORT는 모니터에 의해 처리된다. 이로 인해 게스트 운영체제는 다른 시스템 구성 요소의 메모리 읽기 또는 쓰기에 의한 정보의 유출 또는 시스템 오류로부터 보호될 수 있다. 이는 가상 머신에서 동작하는 소프트웨어의 오류나 악의적인 접근에 의한 시스템 오류를 방지한다.

3.5 I/O 가상화

TrustZone에서 장치는 TZPC의 설정을 통해 시큐어 월드 또는 노멀 월드에 할당 될 수 있다. 장치가 할당 되면 게스트 운영체제는 자신의 I/O 장치에 대한 연

산을 직접 디바이스 드라이버를 통해 수행한다. 이를 통해 LET-V에서 수행되는 게스트 운영체제는 가상화를 하지 않은 환경에서 운영체제의 I/O 연산과 거의 비슷한 성능을 낼 수 있다.

장치의 인터럽트는 GIC의 인터럽트 보안 비트를 설정하여 월드에 할당 한다. 장치 인터럽트는 시큐어 월드에 할당되면 FIQ를 노멀 월드에 할당되면 IRQ를 발생한다. 모니터의 인터럽트 관리 기능에 의해 장치 인터럽트는 할당된 월드로 전달되어 처리된다.

3.6 월드 간 통신

LET-V에서는 공유 메모리를 통한 월드 간 통신 기법을 제공한다. 월드 간 통신을 구현하기 위해 게스트 운영체제에는 공유 메모리에 입출력을 위한 디바이스 드라이버가 추가된다. 통신을 할 때 게스트 운영체제는 디바이스 드라이버를 통해 보내고자 하는 데이터를 공유 메모리에 쓰기 한다. 쓰기가 완료되면 게스트 운영체제는 SMC를 통하여 모니터의 통신 관리 기능을 호출한다. 통신 관리 기능은 다른 월드의 게스트 운영체제에 IPI(Inter Processor Interrupt)를 보내 공유 메모리에 데이터가 입력되었음을 알린다. IPI를 받은 게스트 운영체제는 통신 드라이버를 통하여 공유 메모리로부터 데이터를 읽어 들이며 필요하다면 다시 SMC를 호출하여 상대 게스트 운영체제에 응답을 보낸다. 따라서 게스트 운영체제간의 통신을 제공하기 위해서는 게스트 운영체제의 수정이 필요하다.

4. 실험

이번 장에서는 실험을 통하여 LET-V의 가상화 오버헤드를 측정한다. 물리적 가상화 방법으로 구현된 LET-V와 하드웨어 지원 하이퍼바이저 방식으로 구현된 Xen/ARM의 성능을 비교하여 LET-V의 경량성을 보인다.

4.1 실험 환경

LET-V 성능 평가를 위해 Insiginal Arndale 보드를 사용했다. Arndale 보드는 Samsung Exynos5250 프로세서를 사용하며 Exynos5250은 1.7GHz ARM Cortex-A15 아키텍처에 기반하고 있다. ARM Cortex-A15 아키텍처는 LET-V에서 이용하는 ARM TrustZone Security Extension과 Xen/ARM에서 이용하는 Virtualization Extension을 모두 지원하기 때문에 두 가상화 플랫폼의 성능을 비교하기에 적합하다.

LET-V 가상화 환경에서 시큐어 월드와 노멀 월드에는 각각 AUTOSAR2.0 실시간 운영체제와 Linux 3.9.0 범용 운영체제가 게스트 운영체제로 동작하도록 구현되었다. 한편, Xen/ARM 가상화 환경은 게스트 운영체제로 DomU에 Linux 3.9.0 이 구동하며 제어 도메인인 Dom0에서 마찬가지로 Linux 3.9.0이 구동하도록 구현되었다.

LET-V와 Xen/ARM의 가상화 오버헤드를 측정하기 위해 LET-V의 노멀 월드 Linux 게스트 운영체제와 Xen/ARM 의 DomU Linux 게스트 운영체제를 가상화

를 하지 않은 Native Linux를 기준으로 하여 비교하였다. 크로스 컴파일 된 Lmbench 벤치마크가 성능 평가를 위해 사용되었으며 Xen/ARM Linux의 루트 파일 시스템은 Buildroot를 사용하여 생성되었다.

4.2 코드 크기

먼저 코드 크기 면에서 LET-V의 경량성을 보이기 위해 LET-V와 Xen/ARM의 코드 라인 수와 이미지 크기를 비교하였다. 표 1 은 그 결과를 나타낸다. 코드 라인 수와 이미지 크기 모두 LET-V가 Xen/ARM에 비해 10배 이상 작은 크기를 보인다. 이것은 LET-V가 가상화 기능을 구현할 때 TrustZone의 기능을 최대한 이용하여 소프트웨어 부분을 최소화했기 때문이다. 이처럼 작은 코드 크기는 실행 시 메모리의 foot print를 감소시키고 가상화 소프트웨어에 대한 코드 검증 비용과 노력을 절감할 수 있다는 장점을 가진다.

	코드 라인 수	이미지 크기
LET-V	1K	69KB
Xen/ARM	110K	880KB

표 1. LET-V 와 Xen/ARM 코드 크기 비교

4.3 CPU 연산

그림 2는 가상화 환경에서 정수 연산에 대한 지연시간을 Native Linux와 비교한 결과이다. 정수 연산 명령은 범용 레지스터가 이용되며 게스트 운영체제가 연

산 명령을 수행할 때 VMM의 개입을 필요로 하지 않는다. 때문에 LET-V와 Xen/ARM 모두 Native와 비교해서 지연시간에 큰 차이를 보이지 않는다.

그림 3은 가상화 환경에서 부동 소수점 연산에 대한 지연시간을 Native와 비교한 결과이다. 정수 연산 지연시간과 다르게 Xen/ARM의 지연시간은 Native에 비해 2~17배 증가했다. 이는 ARM 프로세서에서 제공하는 VFP(Vector Floating Point) 아키텍처에 의한 것이다. VFP 는 부동소수점 연산을 위해 자체적인 레지스터를 가진다. Xen/ARM 에서는 부동 소수점 연산 시 VFP 아키텍처를 사용하는 대신 하이퍼바이저를 통해 소프트웨어적으로 에뮬레이트하여 처리한다. 반면에 LET-V에서는 VFP 아키텍처를 노멀 월드에 할당하여 게스트 운영체제가 VFP를 그대로 사용할 수 있도록 한다.

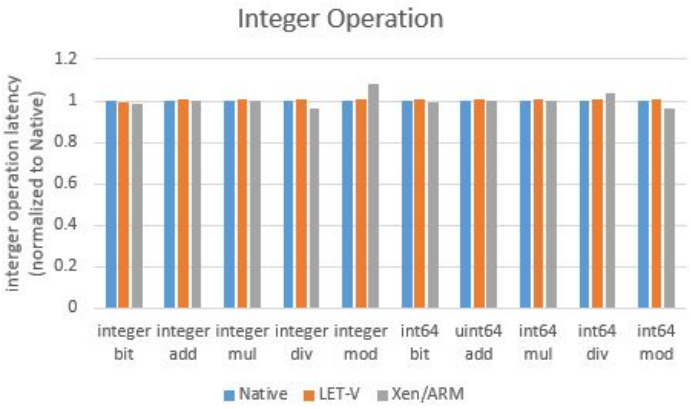

그림 2. 정수 연산 지연시간

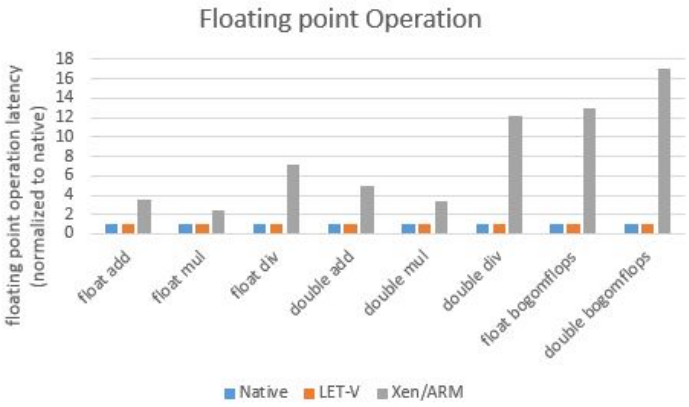

그림 3. 부동 소수점 연산 지연시간

4.4 I/O 연산

LET-V와 Xen/ARM의 I/O 장치에 가상화를 평가하기 위해 파일 연산을 수행하여 루트 파일 시스템을 가지는 SD카드 장치에 대한 I/O 가상화 오버헤드를 측정했다.

그림 4는 0K에서 10K까지 파일 사이즈를 변경하면서 파일 생성과 삭제 연산에 대한 지연시간을 측정한 결과이다. LET-V가 Native와 거의 같은 정도의 지연시간이 소요된 반면 Xen/ARM은 적게는 30%에서 많게는 80%까지 지연시간의 증가를 보였다. 이것은 Xen/ARM의 분리 드라이버 모델에 의한 것이다. Xen에서 게스트 운영체제는 I/O 연산을 수행하기 위해 입출력 관리 도메인인 Dom0에 요청을 보내고 Dom0에서 장치 연산을 처리한 뒤 다시 Dom0로부터 응답을 받아 장치 연산을 완료하게 된다. 이와 같은 Dom0 도메인과의 상호작용이 입출력 연산의 오

버헤드가 된다. 반면에 LET-V는 게스트 운영체제가 직접 장치 드라이버를 통한 입출력 연산을 수행하므로 Native와 비슷한 입출력 성능을 보일 수 있다. 그림 5는 각 가상화 환경에서 파일 읽기와 복사 연산의 대역폭을 비교한 것이다. Xen/ARM이 Native에 비하여 20%의 파일 복사 오버헤드를 갖는다.

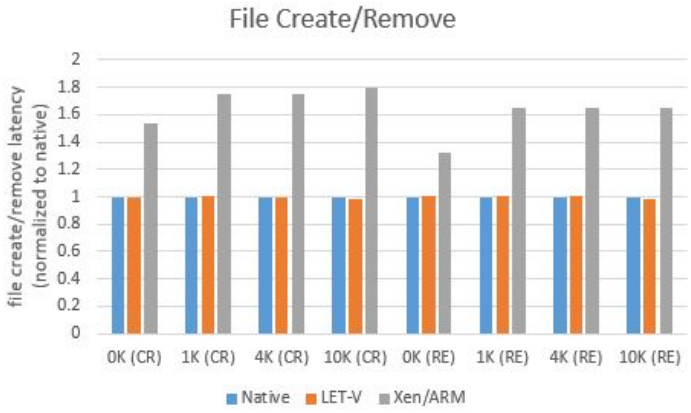

그림 4. 파일 생성 및 삭제 지연 시간

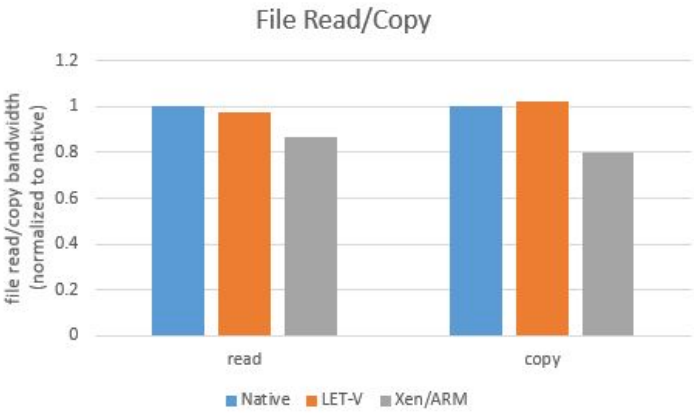

그림 5. 파일 읽기 및 복사 대역폭

5. 결론

본 논문에서는 인포테인먼트 시스템을 자동차 시스템에 안전하고 효율적으로 통합하기 위해 LET-V 경량 가상화 플랫폼을 설계하고 구현하였다. ARM TrustZone에서 제공하는 하드웨어 분할을 이용하여 물리적 가상화 기법으로 가상화 플랫폼을 구현하였다. 가상화 소프트웨어인 모니터에는 최소한의 기능만을 구현하여 가상화 오버헤드를 최소화했다. 구현된 가상화 플랫폼과 Xen/ARM을 비교하여 LET-V가 CPU 연산에서 I/O 연산에서 Xen/ARM보다 낮은 가상화 오버헤드를 가지는 것을 보였다.

참고 문헌

[1] Akira Y. Toyota H. Akiko H. Hirohisa M. Haruhidko S, “Car Information System for Added Value in Connected Cars”, Hitachi Review, Vol. 63, No. 2, Feburary 2014, pp 128 - 132, 2014.

[2] ISO, C. D. 26262, Road vehicles - Functional safety. ISO Standard, 2011.

[3] Pelzl, J., Wolf, M., & Wollinger, T. “Virtualization technologies for cars” Technical Report 2008, escrypt GmbH-Embedded Security. 2008.

[4] A. Groll, J. Holle, C. Ruland, M. Wolf, T. Wollinger, and F. Zweers, "Oversee a secure and open communication and runtime platform for innovative automotive applications," presented at the 7th ESCAR Embedded Security in Cars Conference, Düsseldorf, 2009.

[5] P. Wilson, A. Frey, T. Mihm, D. Kershaw, and T. Alves, "Implementing Embedded Security on Dual-Virtual-CPU Systems," Design Test of Computers, IEEE, vol. 24, pp. 582-591, 2007.

[6] T. Alves and D. Felton, "TrustZone: Integrated Hardware and Software Security---Enabling Trusted Computing in Embedded Systems. White paper, ARM, July 2004," ed.

[7] Hwang, J. Y., Suh, S. B., Heo, S. K., Park, C. J., Ryu, J. M., Park, S. Y., & Kim, C. R. (2008, January). Xen on ARM: System virtualization using

Xen hypervisor for ARM-based secure mobile phones. In Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE (pp. 257-261). IEEE.

[8] (2013). Xen ARM with Virtualization Extensions/Arndale Website.

[9] ARMLtd. ARMCortex-A15 Technical ReferenceManual ARM DDI 0438C, Sept. 2011.