(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2012 년 9 월 7 일 (07.09.2012)

WIPO | PCT

(10) 국제공개번호
WO 2012/118268 A2

(51) 국제특허분류:
G06F 9/44 (2006.01) *G06F 9/06* (2006.01)
G06F 9/38 (2006.01)

(21) 국제출원번호: PCT/KR2011/009483

(22) 국제출원일: 2011 년 12 월 8 일 (08.12.2011)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
10-2011-0017732 2011 년 2 월 28 일 (28.02.2011) KR

(71) 출원인 (US 을(를) 제외한 모든 지정국에 대하여): ㈜지노게임즈 (GINNO GAMES INC.) [KR/KR]; 서울특별시 강남구 삼성동 147-7 서경빌딩 5 층, 135-090 Seoul (KR).

(72) 발명자; 겸
(75) 발명자/출원인 (US 에 한하여): 김정택 (KIM, Jumg taek) [KR/KR]; 서울특별시 역삼 1 동 682-1 효신빌라 306 호, 135-081 Seoul (KR).

(74) 대리인: 심충섭 (SHIM, Choong sup); 서울특별시 강남구 논현동 13-7 유진빌딩 302 호, 135-811 Seoul (KR).

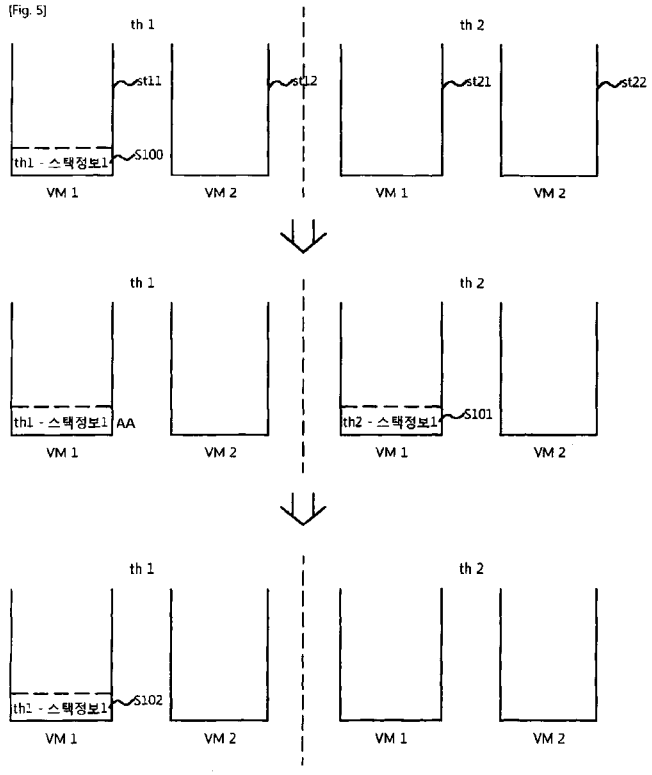
(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR),

[다음 쪽 계속]

(54) Title: MULTI-THREAD PROCESSING SYSTEM USING A MULTI-VIRTUAL MACHINE, AND METHOD THEREFOR

(54) 발명의 명칭 : 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 그 방법



(57) Abstract: The present invention relates to a multi-thread processing system using a multi-virtual machine, and to a method therefor. The method for multi-thread processing using the multi-virtual machine comprises the steps of: when a first function call is provided to a second actor by a first actor assigned in a first virtual machine for processing a first thread, inserting first stack information corresponding to the first function call into a first stack corresponding to the first thread and to the first virtual machine in order to provide the first function call; providing a second function call prior to the completion of the first function call in order for a third actor assigned in the first virtual machine to process a second thread; and inserting second stack information corresponding to the second function call into a second stack corresponding to the second thread and to the first virtual machine in order to provide the second function call.

(57) 요약서:

[다음 쪽 계속]

WO 2012/118268 A2

OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서 없이 공개하며 보고서 접수 후 이를 별도 공개함 (규칙 48.2(g))

멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 그 방법이 개시된다. 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 제 1 스레드(thread)를 처리하기 위해 제 1VM(virtual machine)에 할당된 제 1 액터가 제 2 액터로 제 1 펑션 콜(function call)을 하면, 상기 제 1 펑션 콜을 하기 위해 상기 제 1 펑션 콜에 대응되는 제 1 스택정보가 상기 제 1 스레드 및 상기 제 1VM 에 대응되는 제 1 스택에 삽입되는 단계, 상기 제 1VM 에 할당된 제 3 액터가 제 2 스레드를 처리하기 위해 상기 제 1 펑션 콜이 완료되기 전에 제 2 펑션 콜을 하는 단계, 및 상기 제 2 펑션 콜을 하기 위해 상기 제 2 펑션 콜에 대응되는 제 2 스택정보가 상기 제 2 스레드 및 상기 제 1VM 에 대응되는 제 2 스택에 삽입되는 단계를 포함한다.

명세서

발명의 명칭: 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 그 방법

기술분야

[1] 본 발명은 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 그 방법에 관한 것으로, 보다 상세하게는 멀티 스레드(multi-thread)를 지원하지 않는 버추얼 머신(virtual machine, 이하 'VM')을 이용하여 멀티 스레드를 처리할 수 있는 액터 시스템을 구현할 수 있는 시스템 및 그 제공방법에 관한 것이다.

[2]

배경기술

[3] 데이터 처리 시스템의 발달과 함께는 최근에는 멀티코어 환경이 널리 이용되고 있다. 또한, 수정 및 테스트가 용이한 장점 및 코딩의 간편성 등의 다양한 이유로 스크립트(script) 언어도 널리 사용되고 있다.

[4] 스크립트 언어 중에는 멀티 스레드를 지원하지 않는 스크립트 언어가 존재할 수 있다. 멀티 스레드를 사용하지 못하는 경우에는 멀티코어와 같은 환경을 제대로 활용할 수 없게 된다. 따라서, 멀티 스레드를 지원하지 않는 스크립트 언어만으로 또는 다른 언어와 연동하여 소정의 멀티코어 환경을 사용하는 시스템을 구현하기 위해서는 서로 독립적인 여러 개의 스크립트 VM을 사용하는 것이 필요하다.

[5] 한편, 대용량의 데이터 프로세싱을 요구하는 컴퓨팅 시스템을 구현하기 위해 병렬 컴퓨팅이 이용될 수 있다. 이러한 병렬 컴퓨팅의 일 예로 객체 기반의 시스템 예컨대, 액터 시스템(Actor System)에 대한 개념이 공지된 바 있다. 이러한 액터 시스템에 대한 개념 및 액터 시스템에서의 데이터 처리방법에 대해서는 본 출원인이 출원한 한국특허출원(출원번호 10-2009-0104500, "교착 상태의 방지를 위한 데이터 처리 방법 및 시스템", 이하 '선출원')에도 개시된 바 있다. 선출원에 기재된 기술적 사상 및 내용은 본 명세서의 레퍼런스로 포함될 수 있다.

[6] 액터 시스템은 대용량의 데이터 프로세싱을 수행하기 위해 멀티 스레딩을 지원하는 환경(예컨대, 멀티코어 환경)이 요구될 수 있다. 따라서, 전술한 바와 같이 스크립트 언어를 사용하여 이러한 액터 시스템을 구현하기 위해서는 복수의 스크립트 VM을 이용할 수 있는 것이 요구될 수 있다.

[7]

발명의 상세한 설명

기술적 과제

[8] 따라서, 본 발명이 이루고자 하는 기술적인 과제는 멀티 스레드를 지원하지 않는 스크립트 언어를 이용하여 객체 기반의 시스템(예컨대, 액터 시스템)을 구현할 수 있는 기술적 사상을 제공하는 것이다.

[9] 또한, 이러한 스크립터 언어를 이용한 객체 기반의 시스템(예컨대, 액터 시스템)이 멀티 스레딩을 수행할 수 있도록 복수의 VM을 이용하는 기술적 사상을 제공하는 것이다.

[10] 또한, 복수의 VM을 이용하는 경우, 서로 다른 VM에 할당된 객체(예컨대, 액터)간의 펑션 콜(function call)을 안정적으로 수행할 수 있는 기술적 사상을 제공하는 것이다.

[11]

과제 해결 수단

[12] 상기 기술적 과제를 달성하기 위한 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 제1스레드(thread)를 처리하기 위해 제1VM(virtual machine)에 할당된 제1액터가 제2액터로 제1펑션 콜(function call)을 하면, 상기 제1펑션 콜을 하기 위해 상기 제1펑션 콜에 대응되는 제1스택정보가 상기 제1스레드 및 상기 제1VM에 대응되는 제1스택에 삽입되는 단계, 상기 제1VM에 할당된 제3액터가 제2스레드를 처리하기 위해 상기 제1펑션 콜이 완료되기 전에 제2펑션 콜을 하는 단계, 및 상기 제2펑션 콜을 하기 위해 상기 제2펑션 콜에 대응되는 제2스택정보가 상기 제2스레드 및 상기 제1VM에 대응되는 제2스택에 삽입되는 단계를 포함한다.

[13] 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 상기 제1액터가 상기 제1펑션 콜을 하기 위해, 상기 제1VM의 락(Lock)이 해제되고, 상기 제2액터가 할당된 제2VM이 락 되는 단계를 더 포함할 수 있다.

[14] 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 상기 제2펑션 콜이 완료되기 전에 상기 제1펑션 콜이 완료되면, 상기 제1액터는 상기 제1스택정보에 상응하는 프로세싱을 수행하는 단계를 더 포함할 수 있다.

[15] 상기 제1펑션 콜 또는 상기 제2펑션 콜 중 적어도 하나는 리드 콜(read call) 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 할 수 있다.

[16] 상기 기술적 과제를 해결하기 위한 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 소정의 스레드가 생성되면 상기 스레드에 VM별 스택을 할당하는 단계, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템에 의해 관리되는 소정의 제1액터가 상기 스레드를 처리하면서 소정의 펑션 콜을 하는 단계, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 상기 스레드의 상기 제1액터가 할당된 제1VM에 대응되는 스택을 판단하고, 판단된 스택에 상기 펑션 콜에 대응되는 스택정보를 삽입하는 단계를 포함한다.

[17] 상기 펑션 콜은 상기 제1VM과는 다른 제2VM에 할당된 제2액터로의 리드 콜 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 할 수 있다.

[18] 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 상기 펑션 콜을 처리하기 위해, 상기

제1VM의 락(Lock)을 해제하고, 제2VM을 락 하는 단계를 더 포함할 수 있다. 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 프로그램을 기록한 컴퓨터 판독 가능한 기록매체에 저장될 수 있다.

[19] 상기 기술적 과제를 해결하기 위한 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 복수의 VM들, 소정의 스레드가 생성되면 상기 스레드에 VM별 스택을 할당하기 위한 스택 모듈, 복수의 액터들(Actor)을 생성 및 관리하고 상기 복수의 액터들 각각을 상기 복수의 VM 중 어느 하나에 할당하기 위한 액터 모듈, 및 상기 복수의 액터들 중 어느 하나의 제1액터가 소정의 스레드를 처리하기 위해 펑션 콜을 하면, 상기 스레드 및 상기 제1액터가 할당된 제1VM에 대응되는 스택을 판단하고, 판단된 스택에 상기 펑션 콜에 대응되는 스택정보를 삽입하기 위한 제어모듈을 포함한다.

[20] 상기 펑션 콜은 상기 제1VM과는 다른 제2VM에 할당된 제2액터로의 리드 콜 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 할 수 있으며, 상기 제어모듈은 상기 펑션 콜을 처리하기 위해, 상기 제1VM의 락(Lock)을 해제하고, 제2VM을 락 할 수 있다.

[21]

발명의 효과

[22] 본 발명의 기술적 사상에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 그 방법은 액터 간의 펑션 콜(예컨대, 리드 콜)을 수행할 때 데드 락(dead lock)이 발생하지 않는 효과가 있다.

[23] 또한, 각각의 스레드에 VM별로 스택이 분리되어 있으므로, 멀티 스레드 수행에 의한 VM의 스택 에러가 발생하지 않을 수 있는 효과가 있다.

[24] 또한, 대기 상태의 액터의 VM은 락(Lock)이 해제될 수 있으므로, 시스템의 성능이 향상될 수 있는 효과가 있다.

[25]

도면의 간단한 설명

[26] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 간단한 설명이 제공된다.

[27] 도 1은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 서로 다른 VM에 할당된 액터 간에 통신을 위한 락 설정방법을 설명하기 위한 도면이다.

[28] 도 2는 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따른 락 설정방법을 이용하면서 VM 별로 하나의 스택을 사용하는 경우 발생할 수 있는 문제점을 설명하기 위한 도면이다.

[29] 도 3은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 서로 다른 VM에 할당된 액터 간에 통신을 수행하는 일 예를 설명하기 위한 도면이다.

[30] 도 4는 도 3에 도시된 바와 같은 통신을 수행할 때 종래의 방법에 따른 VM 스택의 변화를 설명하기 위한 도면이다.

[31] 도 5는 도 3에 도시된 바와 같은 통신을 수행할 때 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 스택의 변화를 설명하기 위한 도면이다.

[32] 도 6은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 서로 다른 VM에 할당된 액터 간에 통신을 수행하는 다른 이례를 나타내는 도면이다.

[33] 도 7은 도 6에 도시된 상황에서 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 스택이 변화하는 과정을 설명하기 위한 도면이다.

[34] 도 8은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템의 개략적인 구성을 나타내는 도면이다.

[35]

발명의 실시를 위한 형태

[36] 본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 바람직한 실시 예를 예시하는 첨부 도면 및 첨부 도면에 기재된 내용을 참조하여야만 한다.

[37] 또한, 본 명세서에 있어서는 어느 하나의 구성요소가 다른 구성요소로 데이터를 '전송'하는 경우에는 상기 구성요소는 상기 다른 구성요소로 직접 상기 데이터를 전송할 수도 있고, 적어도 하나의 또 다른 구성요소를 통하여 상기 데이터를 상기 다른 구성요소로 전송할 수도 있는 것을 의미한다.

[38] 반대로 어느 하나의 구성요소가 다른 구성요소로 데이터를 '직접 전송'하는 경우에는 상기 구성요소에서 다른 구성요소를 통하지 않고 상기 다른 구성요소로 상기 데이터가 전송되는 것을 의미한다.

[39] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시 예를 설명함으로써, 본 발명을 상세히 설명한다. 각 도면에 제시된 동일한 참조부호는 동일한 부재를 나타낸다.

[40] 도 1은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 서로 다른 VM에 할당된 액터 간에 통신을 위한 락 설정방법을 설명하기 위한 도면이다.

[41] 먼저 도 1a를 참조하면, 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 복수의 VM들(예컨대, VM 1, VM 2)을 제공한다. 상기 VM들(예컨대, VM 1, VM 2) 각각은 적어도 하나의 액터(예컨대, 10, 20)에 대응될 수 있다. 즉, 특정 액터가 생성될 때 마다, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 생성된 액터의 프로세싱을 담당할 VM을 지정할 수 있다. 소정의 VM(예컨대, VM1)은 복수의 액터에 대응될 수도 있다. 하지만, 동시에

복수의 액터들(또는 스레드)이 어느 하나의 VM을 사용할 수는 없다. 이는 상기 VM(예컨대, VM 1)이 멀티 스레드를 지원하지 않기 때문이다.

[42] 따라서, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 전술한 바와 같이 복수의 VM들(예컨대, VM 1, VM 2)을 이용하여 멀티 스레딩(multi-threading)을 지원하도록 구현될 수 있다.

[43] 한편, 각각의 VM들(예컨대, VM 1, VM 2)은 특정 시점에서 어느 하나의 액터(예컨대, 10 또는 20)만이 사용할 수 있다. 따라서, 도 1에 도시된 바와 같이 액터(10)가 VM 1을 사용하는 경우에는 다른 액터들 또는 다른 스레드가 상기 VM 1을 사용하지 못하도록 상기 VM 1을 락(Lock)할 수 있다.

[44] 이처럼 상기 액터(10)가 상기 VM 1을 락 한 후, 상기 VM 1을 이용해 소정의 동작 또는 기능을 수행하다가 다른 액터(예컨대, Actor B, 20)와 통신(communication)을 수행하여야 하는 경우가 있을 수 있다. 예컨대, 이벤트(event) 또는 리드(read) 호출(call) 등과 같은 다양한 펑션 콜(function call)이 그러한 경우일 수 있다.

[45] 상기와 같은 펑션 콜을 이용해 서로 다른 VM에 할당(대응)된 액터간에 통신을 수행하여야 하는 경우, 펑션을 콜하는 액터(예컨대, 10)은 콜을 당하는 액터(예컨대, 20)가 할당된 VM(예컨대, VM 2)의 락을 획득하여야 할 수 있다(S20). 이를 통해 상기 VM(예컨대, VM 2)가 상기 펑션 콜에 대응되는 동작을 수행하고, 그 동안에 다른 액터 또는 스레드가 상기 VM(예컨대, VM 2)를 사용하지 못하도록 할 수 있다. 이때 상기 펑션 콜을 하는 액터(예컨대, 10)가 자신의 VM(예컨대, VM 1)의 락을 계속 유지하고 있어야 할지 또는 락을 해제하여야 할지가 이슈(issue)가 될 수 있다.

[46] 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에서는 펑션 콜을 하는 액터(예컨대, 10)는 자신의 VM(예컨대, VM 1)의 락을 해제하고(S10), 펑션 콜을 받는 액터(예컨대, 20)의 VM(예컨대, VM 2)의 락을 획득하도록 할 수 있다(S20). 이처럼 소정의 펑션 콜을 위해 VM의 락을 해제하게 되면, 펑션 콜에 대한 응답이 올 때까지(S11), 상기 VM을 다른 액터들 또는 스레드가 이용할 수 있게 되므로, 시스템의 효율성이 높아지는 효과가 있다.

[47] 만약, 펑션 콜을 하는 액터(예컨대, 10)가 자신의 VM(예컨대, VM 1)의 락을 해제 하지 않는 경우에는 시스템의 효율성 뿐만 아니라, 데드 락(dead-lock)의 문제도 발생할 수 있다. 이는 도 1b를 참조하여 설명하도록 한다.

[48] 도 1b를 참조하면, 소정의 액터(10)는 자신의 VM(예컨대, VM 1)의 락을 유지한 상태에서(S30) 다른 액터(20)로 소정의 제1펑션 콜을 할 수 있다. 그러면, 상기 액터(20)는 자신의 VM(예컨대, VM 2)의 락을 획득할 수 있다(S40). 이때 상기 액터(20)가 상기 제1펑션 콜과 실질적으로 동시에 상기 액터(10)로 제2펑션 콜을 하거나, 상기 액터(20)가 자신의 VM(예컨대, VM 2)을 이용하여 소정의 동작을 수행하다가(S41), 상기 제1펑션 콜에 상응하는 응답을 상기 액터(10)로 출력하기

전에, 상기 액터(10)로 소정의 제2펑션 콜을 할 수 있다(S42). 그러면, 상기 액터(10)의 VM(예컨대, VM 1)은 액터(10)에 의해 락이 유지된 상태 즉, 상기 제1펑션 콜에 상응하는 응답을 받아야 하는 상황에서 다시 액터(20)에 의해 락이 획득되게 되어(S31) 두 액터가 서로 응답만을 기다리게 되는 데드 락 상태가 되게 된다.

[49] 따라서, 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 도 1a에 도시된 바와 같이 펑션 콜을 하는 액터는 자신의 VM의 락을 해제하고 펑션 콜을 하도록 구현될 수 있다.

[50] 다시 도 1a를 참조하면, 상기 펑션 콜을 당한 액터(예컨대, 20)는 자신의 VM(예컨대, VM 2)의 락을 획득하고(S20), 상기 펑션 콜에 상응하는 동작을 수행할 수 있다(S21). 그리고, 동작이 완료되면, 다시 자신의 VM(예컨대, VM 2)의 락을 해제하고(S22) 상기 펑션 콜에 상응하는 응답(response)을 상기 액터(예컨대, 10)으로 전송할 수 있다. 이처럼 펑션 콜에 상응하는 응답 역시, 펑션 콜과 마친가지로 응답을 하는 액터(예컨대, 20)는 자신의 VM(예컨대, VM 2)의 락을 해제하고(S22), 응답을 받는 액터(예컨대, 10)가 자신의 VM(예컨대, VM 1)의 락을 획득(S12)하도록 할 수 있다. 그러면, 응답을 받은 액터(예컨대, 10)은 응답결과를 받고 소정의 동작을 수행할 수 있다.

[51] 한편, 도 1a에 도시된 바와 같이 서로 다른 VM 에 할당된 액터 간에 통신을 위한 펑션 콜을 할 때의 락 컨트롤 방법을 사용하게 될 때, VM 별로 하나의 스택을 이용하게 되면 예기치 못한 스택 에러(stack error)가 발생할 수 있다. 이는 도 2를 참조하여 설명하도록 한다.

[52] 도 2는 VM 별로 하나의 스택을 사용하는 경우 발생할 수 있는 문제점을 설명하기 위한 도면인데, 먼저 도 2a는 도 1a에서 설명한 바와 같이 본 발명의 실시 예에 따라 펑션 콜을 하는 액터는 자신의 VM의 락을 해제하도록 구현될 수 있다.

[53] 도 2a에 도시된 바와 같이 제1액터(10) 및 제3액터(30)는 VM 1에 할당되고, 제2액터(20)는 VM 2에 할당될 수 있다.

[54] 이러한 경우, 제1액터(10)는 자신의 VM(예컨대, VM 1)의 락을 가진 상태에서 소정의 동작을 수행하다가, 락을 해제하고(S50) 소정의 제1펑션 콜을 제2액터(20)로 할 수 있다. 그러면, 제2액터(20)는 자신의 VM(예컨대, VM 2)의 락을 획득하고(S60), 자신의 VM(예컨대, VM 2)을 이용해 상기 제1펑션 콜에 상응하는 동작을 수행하며, 상기 동작이 완료되면 락을 해제하고(S61), 상기 제1펑션 콜에 상응하는 응답을 상기 제1액터(10)로 전송할 수 있다(S54).

[55] 한편, 상기 제1액터(10)가 제1펑션 콜을 하고 상기 제1펑션 콜에 상응하는 응답을 받기까지는, 상기 제1액터(10)가 자신의 VM(예컨대, VM 1)의 락을 해제하였으므로 다른 액터(예컨대, 제3액터(30)) 또는 다른 스레드가 상기 VM(예컨대, VM 1)을 이용할 수 있다.

[56] 예컨대, 제3액터(30)는 상기 제1액터(10)가 자신의 락을 해제한 후(S50), 소정의

이벤트 호출(S51)에 의해 상기 VM(예컨대, VM 1)의 락을 획득하고(S52) 소정의 동작을 수행할 수 있다. 그리고, 상기 제3액터(30)가 상기 동작을 수행하다가 자신의 VM(예컨대, VM 1)의 락을 해제하고(S53) 소저의 제2펑션 콜을 다른 액터로 할 수 있다.

[57] 이러한 플로우로 VM 1에 할당된 액터들이 순차적으로 락을 획득하고, 펑션 콜을 수행하는 경우 VM 1에 대응되는 스택은 스택 에러가 발생할 수 있다. 이는 도 2b를 참조하여 설명한다.

[58] 도 2b를 참조하면, 제1액터(10)가 자신의 VM(예컨대, VM 1)의 락을 획득하여 소정의 동작을 수행하다가 제1펑션 콜을 하게 되면(S50), 상기 제1액터(10)는 자신이 수행하던 상기 동작에 대한 정보 즉, 제1스택정보를 스택(Stack_VM 1)에 삽입하게 된다. 상기 제1스택정보를 상기 제1펑션 콜에 대응되는 스택정보로 정의할 수 있다. 그리고, 상기 제1펑션 콜이 완료되면(즉, 응답을 수신하면), 상기 제1스택정보를 상기 스택(Stack_VM 1)에서 인출하여 원래 수행하던 상기 동작을 계속 수행할 수 있다.

[59] 하지만, 도 2a에 도시된 바와 같이 상기 제1펑션 콜이 완료되기 전에(S54), 상기 VM 1의 락을 제3액터(30)가 획득하고, 상기 제3액터(30)가 소정의 다른 VM에 존재하는 다른 액터로 제2펑션 콜을 하게 되면(S53), 상기 제3액터(30)가 할당된 상기 VM 1은 상기 제2펑션 콜에 대응되는 제2스택정보가 상기 스택(Stack_VM 1)에 삽입되게 된다. 이후, 상기 제1액터(10)가 상기 제1펑션 콜이 완료되어 다시 자신의 VM(예컨대, VM 1)의 락을 획득하게 되면, 상기 제1액터(10)는 자신의 VM(예컨대, VM 1)에 대응되는 스택(Stack_VM 1)의 탑(top)에 존재하는 스택정보(즉, 제2스택정보)를 인출하게 된다. 인출되는 스택정보(즉, 제2스택정보)는 상기 제3액터(30)가 수행하던 동작이므로, 결국, 제1액터(10)가 제3액터(30)가 수행하던 동작에 상응하는 스택정보를 꺼내게 되어 에러가 발생하게 된다.

[60] 이러한 스택 에러는 특정 VM에서 소정의 액터가 펑션 콜을 수행하고, 상기 펑션 콜이 완료되기 전에 상기 특정 VM에서 다른 액터가 또 다른 펑션 콜을 수행한 후, 처음 수행되었던 펑션 콜이 완료되는 상황이기만 하면 발생할 수 있다.

[61] 이처럼, 멀티 스레딩을 위해 단순히 복수의 VM만을 분리하여 제공하는 경우에는 정상적인 동작을 보장할 수 없게 될 수 있다. 즉, 일반적으로는 VM 별로 펑션 콜을 지원하기 위해 하나씩의 스택이 유지될 수 있는데, 복수의 VM 만을 분리하여 제공하고 VM별로 하나의 스택을 유지하는 경우에는 정상적인 동작을 보장할 수 없게 된다.

[62] 따라서, 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템 및 방법은 전술한 바와 같이 VM 별로 스택을 하나씩 유지하는 것이 아니라, 스레드별로 VM 각각에 대응되는 스택을 제공하여 이러한 문제점을 해결할 수 있는 기술적 사상을 제공한다.

[63] 도 3은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 서로 다른 VM에 할당된 액터 간에 통신을 수행하는 일 예를 설명하기 위한 도면이다.

[64] 도 3을 참조하면, 제1액터(10)는 자신의 VM(예컨대, VM 1)의 락을 이용하여 소정의 동작(데이터의 처리)을 수행하다가 다른 VM(예컨대, VM 2)에 할당된 제2액터(20)와 통신을 위해 소정의 펑션 콜을 수행할 수 있다. 상기 펑션 콜은 이벤트 콜, 리드 콜 등 서로 다른 VM에 할당된 액터에게 소정의 정보를 요구하거나, 소정의 액션을 요구하고 응답을 기다릴 수 있는 모든 형태의 메시지 콜을 의미할 수도 있다.

[65] 상기 제1액터(10)는 상기 펑션 콜을 위해 전술한 바와 같이 자신의 락을 해제하고(S100), 제2액터(20)가 자신의 VM(예컨대, VM 2)의 락을 획득하도록 할 수 있다. 그러면, 상기 제2액터(20)는 상기 펑션 콜에 상응하는 동작을 수행한 후, 자신의 락을 해제하고, 상기 펑션 콜에 상응하는 응답을 상기 제1액터(10)로 전송할 수 있다. 그러면, 상기 제1액터(10)는 다시 자신의 VM(예컨대, VM 1)의 락을 획득할 수 있다(S102). 이와 같이 상기 제1액터(10)가 상기 펑션 콜을 하고, 다시 상기 펑션 콜에 상응하는 응답을 받는 일련의 데이터 프로세싱 과정을 하나의 스레드(스레드 1)로 취급할 수 있다.

[66] 한편, 상기 제1액터(10)가 락을 해제하고(S100) 다시 락을 획득(S102)하기 전까지(즉, S101), 소정의 액터에 의해 상기 제1액터(10)의 VM(예컨대, VM 1)은 다른 액터에 의해 이용이 가능한 상태일 수 있다. 상기의 구간(S101)에서 소정의 액터에 의해 상기 VM이 이용되어 특정 동작 또는 데이터 프로세싱을 수행하는 과정을 또 다른 스레드(스레드 2)로 정의할 수 있다.

[67] 도 3에서는 스레드 2가 처리되는 중에 소정의 펑션 콜이 수행될 수 있고, 스레드 2에서 수행된 상기 펑션 콜은 상기 스레드 1에서 수행된 펑션 콜이 완료되기 전에 완료된다고 가정할 수 있다.

[68] 그러면, VM 별로 하나의 스택이 할당된 경우, 상기 VM 1에 대응되는 스택(Stack_VM 1)의 변화는 도 4와 같을 수 있다.

[69] 도 4는 도 3에 도시된 바와 같은 통신을 수행할 때 종래의 방법에 따른 VM 스택의 변화를 설명하기 위한 도면이다.

[70] 도 3 및 도 4를 참조하면, 상기 VM 1에 하나의 스택(Stack_VM 1)이 할당되는 경우, 상기 제1액터(10)가 스레드 1을 수행하면서 제1펑션 콜을 하게 되면(S100), 상기 제1펑션 콜에 대응되는 스택정보(th1_스택정보 1)가 상기 스택(Stack_VM 1)에 삽입되게 된다. 그 후 소정의 액터가 스레드 2를 수행하면서(S101) 소정의 펑션 콜을 수행하게 되면, 상기 스택(Stack_VM 1)에는 상기 펑션 콜에 대응되는 소정의 스택정보(th2_스택정보 1)가 삽입되게 된다. 이 후, 상기 스레드 1에서 상기 제1액터(10)가 락을 획득하기 전(S102)의 소정의 시점에서, 상기 스레드 2의 펑션 콜이 완료되면(S101), 상기 스택(Stack_VM 1)에 삽입된 스택정보(th2_스택정보 1)는 상기 스택(Stack_VM 1)에서 인출되게 된다.

[71] 그러면, 상기 스택(Stack_VM 1)에는 상기 스레드 1의 제1펑션 콜에 대응되는 스택정보(th1_스택정보 1)가 가장 탑에 위치하게 되어, 상기 제1액터(10)가 락을 획득하면(S102) 상기 스택정보(th1_스택정보 1)를 정상적으로 이용하게 될 수 있다.

[72] 한편, 도 3에 도시된 바와 같은 상황에서 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 도 5에 도시된 바와 같은 방식을 이용할 수 있다.

[73] 도 5는 도 3에 도시된 바와 같은 통신을 수행할 때 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 스택의 변화를 설명하기 위한 도면이다.

[74] 도 3 및 도 5를 참조하면, 제1액터(10)가 스레드 1을 처리하면서, 제1펑션 콜을 수행하면(S100), 상기 제1펑션 콜에 대응되는 스택정보(th1_스택정보 1)가 상기 스레드 1에 할당되고 상기 제1액터(10)가 할당된 VM 1에 대응되는 스택(st11)에 삽입된다. 즉, 도 4에 도시된 바와는 달리, 소정의 스택정보(th1_스택정보 1)를 스택에 삽입하기 위해 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 스레드 및 VM에 대응되는 스택을 먼저 판단하고, 판단된 스택에 상기 스택정보(th1_스택정보 1)를 삽입할 수 있다.

[75] 그 후 소정의 액터가 스레드 2를 수행하면서(S101) 상기 VM 1을 이용하여 소정의 펑션 콜을 수행하게 되면, 상기 펑션 콜에 대응되는 소정의 스택정보(th2_스택정보 1)는 상기 스레드 2 및 상기 VM 1에 대응되는 스택(st21)에 삽입되게 된다. 이 후, 상기 스레드 1에서 상기 제1액터(10)가 락을 획득하기 전(S102)의 소정의 시점에서, 상기 스레드 2의 펑션 콜이 완료되면(S101), 상기 스택(st21)에 삽입된 스택정보(th2_스택정보 1)는 상기 스택(st21)에서 인출되게 된다.

[76] 그 후, 상기 제1액터(10)가 락을 획득하면(S102), 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 상기 스택정보(th1_스택정보 1)가 저장된 스택을 검색하고, 검색된 스택(St11)에서 상기 스택정보(th1_스택정보 1)를 인출할 수 있다.

[77] 한편, VM 별로 하나의 스택이 할당된 경우에는 스택 에러가 발생할 수 있는 상황에서, 본 발명의 기술적 사상에 따라 스택이 할당된 경우의 스택변화를 살펴보면 도 6 내지 도 7과 같을 수 있다.

[78] 도 6은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 서로 다른 VM에 할당된 액터 간에 통신을 수행하는 다른 일례를 나타내고, 도 7은 도 6에 도시된 상황에서 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 따라 스택이 변화하는 과정을 설명하기 위한 도면이다.

[79] 도 6 내지 도 7을 참조하면, 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 도 6에 도시된 바와 같이 제1액터(121) 및

제3액터(123)는 VM 1에 할당되고, 제2액터(122)는 VM 2에 할당될 수 있다.

[80] 이러한 경우, 제1액터(121)는 자신의 VM(예컨대, VM 1)의 락을 가진 상태에서 소정의 동작을 수행하다가, 락을 해제하고 소정의 제1펑션 콜을 제2액터(122)로 할 수 있다(S110). 그러면, 제2액터(122)는 자신의 VM(예컨대, VM 2)의 락을 획득하고, 자신의 VM(예컨대, VM 2)을 이용해 상기 제1펑션 콜에 상응하는 동작을 수행하며, 상기 동작이 완료되면 락을 해제하고, 상기 제1펑션 콜에 상응하는 응답을 상기 제1액터(121)로 전송할 수 있다(S54).

[81] 한편, 상기 제1액터(121)가 제1펑션 콜을 하고 상기 제1펑션 콜에 상응하는 응답을 받기 전에 소정의 제3액터(123)가 상기 VM 1의 락을 획득할 수 있다(S112). 상기 제3액터(123)는 소정의 이벤트 호출(111) 또는 펑션 콜에 의해 상기 VM 1의 락을 획득할 수 있다(S112).

[82] 그리고, 상기 VM 1의 락을 획득한 제3액터(123)는 소정의 동작을 수행하다가 자신의 VM(예컨대, VM 1)의 락을 해제하고(S113) 소저의 제2펑션 콜을 다른 액터로 할 수 있다. 상기 제2펑션 콜은 상기 제1펑션 콜이 완료되기 전 즉, 상기 제1펑션 콜에 대한 응답이 오기 전에 수행될 수 있다.

[83] 이 후, 상기 제1액터(121)는 상기 제2액터(122)로부터 상기 제1펑션 콜에 상응하는 응답을 수신하고, 다시 상기 제1액터(121) 자신의 VM(예컨대, VM 1)의 락을 획득할 수 있다.

[84] 여기서 상기 제1액터(121)가 제1펑션 콜을 수행하고, 다시 상기 제2액터(122)로부터 응답을 받는 일련의 과정이 하나의 스레드(스레드 1)일 수 있고, 상기 제3액터(123)가 상기 VM 1의 락을 획득하고 제2펑션 콜을 하는 일련의 과정이 또 하나의 스레드(스레드 2)일 수 있다.

[85] 이러한 경우, 도 2b에서 전술한 바와 같이 각각의 VM 별로 하나의 스택이 유지되는 경우에는 스택 에러가 발생할 수 있다.

[86] 하지만, 본 발명의 기술적 사상에 의하면, 이러한 경우에도 스택 에러를 방지할 수 있다.

[87] 도 7에 도시된 바와 같이 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법에 의하면, 각각의 스레드별로 VM들 각각에 대응되는 스택들(st11, st12, st21, st22)이 유지될 수 있다.

[88] 그러면, 상기 제1액터(121)가 제1펑션 콜을 수행하면(S110), 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 상기 제1펑션 콜에 상응하는 스레드(즉, 스레드 1) 및 VM(VM 1)에 대응되는 스택(st11)을 판단(검색)하고, 판단된 스택(st11)에 상기 제1펑션 콜에 대응되는 제1스택정보를 삽입할 수 있다.

[89] 이 후,상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 상기 제3액터(123)가 제2펑션 콜을 수행하면(S113), 상기 제2펑션 콜에 상응하는 스레드(즉, 스레드 2) 및 VM(즉, VM1)에 대응되는 스택(st21)을 판단할 수 있다. 그리고 판단된 스택(st21)에 상기 제2펑션 콜에 대응되는 제2스택정보를 삽입할

수 있다.

[90] 그 후, 상기 제1액터(121)가 다시 자신의 VM(예컨대, VM 1)의 락을 획득하면(S114), 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템은 현재의 스레드 및 VM에 대응되는 스택(st11)의 탑에 위치한 스택정보(즉, 제1스택정보)를 인출할 수 있다.

[91] 이처럼, 본 발명의 기술적 사상에 따르면, 스레드 각각이 VM 별 스택을 유지하게 되어 서로 다른 액터에 의해 펑션 콜이 교차되어 수행될 때 발생할 수 있는 스택 에러를 방지할 수 있다. 또한, 스택의 최대 개수는 스레드의 개수 × VM의 개수로 제한될 수 있다.

[92] 도 8은 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템의 개략적인 구성을 나타내는 도면이다.

[93] 도 8을 참조하면, 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)은 제어모듈(110), 액터모듈(120), 스택모듈(130), 및 복수의 VM들(140)을 포함한다. 상기 복수의 VM들(140)에는 적어도 VM 1(141) 및 VM N(142)가 포함될 수 있다.

[94] 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)은 전술한 바와 같이 전술한 바와 같이 멀티 스레딩을 지원하지 않는 소정의 스크립트 언어 및/또는 소정의 다른 언어를 이용하여 소정의 애플리케이션(예컨대, MMOG(Massively Multiplayer Online Games) 등)을 제공하는 서버로 구현될 수 있다. 따라서, 상기 VM들(140)은 상기 스크립트 언어로 구현되는 소정의 가상 머신을 의미할 수 있다. 따라서, 본 발명의 기술적 사상에 따라 상기 스크립트 언어를 이용하면서도 멀티 스레딩이 필요한 소정의 환경(예컨대, 멀티코어 환경)에 적합한 시스템의 구현이 가능할 수 있다.

[95] 한편, 본 명세서에서 모듈이라 함은, 본 발명의 기술적 사상을 수행하기 위한 하드웨어 및 상기 하드웨어를 구동하기 위한 소프트웨어의 기능적, 구조적 결합을 의미할 수 있다. 예컨대, 상기 모듈은 소정의 코드와 상기 소정의 코드가 수행되기 위한 하드웨어 리소스(resource)의 논리적인 단위를 의미할 수 있으며, 반드시 물리적으로 연결된 코드를 의미하거나, 한 종류의 하드웨어를 의미하는 것은 아님은 본 발명의 기술분야의 평균적 전문가에게는 용이하게 추론될 수 있다. 따라서, 상기 모듈은 본 명세서에서 정의되는 기능을 수행하는 하드웨어 및 소프트웨어의 결합을 의미하며 특정 물리적 구성을 의미하는 것은 아니다.

[96] 또한, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)은 본 발명의 기술적 사상을 구현하기 위해 필요한 하드웨어 리소스(resource) 및/또는 소프트웨어를 구비한 논리적인 구성을 의미할 수 있으며, 반드시 하나의 물리적인 구성요소를 의미하거나 하나의 장치를 의미하는 것은 아니다. 즉, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)은 본 발명의 기술적 사상을 구현하기 위해 구비되는 하드웨어 및/또는 소프트웨어의 논리적인 결합을 의미할 수 있으며, 필요한 경우에는 서로 이격된 장치에 설치되어 각각의

기능을 수행함으로써 본 발명의 기술적 사상을 구현하기 위한 논리적인 구성들의 집합으로 구현될 수도 있다. 또한, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)은 본 발명의 기술적 사상을 구현하기 위한 각각의 기능 또는 역할별로 별도로 구현되는 구성들의 집합을 의미할 수도 있다.

[97] 예컨대, 상기 제어모듈(110), 상기 액터모듈(120), 상기 스택모듈(130), 및/또는 상기 VM들(140) 각각은 서로 다른 물리적 장치에 위치할 수도 있고, 동일한 물리적 장치에 위치할 수도 있다. 또한, 구현 예에 따라서는 상기 제어모듈(110), 상기 액터모듈(120), 상기 스택모듈(130), 및/또는 상기 VM들(140) 각각을 구성하는 소프트웨어 및/또는 하드웨어의 결합 역시 서로 다른 물리적 장치에 위치하고, 서로 다른 물리적 장치에 위치한 구성들이 서로 유기적으로 결합되어 각각의 상기 모듈들을 구현할 수도 있다.

[98] 상기 제어 모듈(110)은 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템(100)에 포함되는 다른 구성(예컨대, 상기 액터모듈(120), 상기 스택모듈(130), 및/또는 상기 VM들(140) 등)의 기능 및/또는 리소스(resource)를 제어할 수 있다.

[99] 또한, 상기 제어모듈(110)은 전술한 바와 같이 특정 액터들의 펑션 콜을 컨트롤 할 수 있다. 즉, 특정 액터의 펑션 콜에 의해 수행되는 소정의 VM의 락 해제 및/또는 락 획득을 제어할 수 있다.

[100] 상기 액터 모듈(120)은 복수의 액터들 각각을 생성 및 관리할 수 있다. 또한, 상기 액터 모듈(120)은 특정 액터가 생성되면, 생성된 액터를 상기 복수의 VM들(140) 중 어느 하나에 할당되도록 할 수 있다.

[101] 상기 액터 모듈(120)에 의해 생성 및 관리되는 액터들 각각은 선출원에도 개시된 바와 같이 능동적 객체로써 자신의 상태를 수정하거나, 다른 액터와 통신을 수행할 수 있고, 새로운 액터를 생성할 수도 있다. 또한, 자신이 수행하는 일련의 프로세싱 과정 즉, 스레드에 대한 정보를 가지고 있을 수도 있다.

[102] 상기 스택모듈(130)은 스레드 각각에 상기 복수의 VM들(140)에 대응되는 스택을 할당하고, 이를 유지할 수 있다. 따라서, 상기 스택모듈(130)에 의해 유지되는 스택은 스레드의 수 × VM의 수가 될 수 있다. 스택의 데이터 구조 및 이를 이용한 데이터 입출력은 전산학에서 널리 공지되어 있으므로 상세한 설명은 생략하도록 한다.

[103] 따라서, 상기 제어모듈(110)은 상기 액터 모듈(120)에 의해 생성 및 관리되는 소정의 액터가 펑션 콜을 하면, 상기 펑션 콜에 상응하는 스레드 및 VM에 대응되는 스택이 무엇인지를 판단하고, 판단된 스택에 상기 펑션 콜에 대응되는 스택정보를 삽입하도록 상기 스택모듈(130)을 제어할 수 있다.

[104] 한편, 상기 제어모듈(110)은 상기 펑션 콜이 수행되면, 펑션 콜을 한 액터 자신의 VM의 락을 해제하고, 펑션 콜을 당한 액터의 VM의 락을 획득하도록 제어할 수 있다. 이를 통해 도 1b에서 설명한 바와 같은 데드 락을 방지하고, 시스템의 성능을 향상시킬 수 있다.

[105] 본 발명의 실시 예에 따른 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 하드 디스크, 플로피 디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어, 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고 본 발명을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술분야의 프로그래머들에 의해 용이하게 추론될 수 있다.

[106] 본 발명은 도면에 도시된 일 실시 예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시 예가 가능하다는 점을 이해할 것이다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

[107]

산업상 이용가능성

[108] 본 발명은 객체 기반의 멀티 스체드 시스템에 적용될 수 있다.

[109]

[110]

청구범위

[청구항 1] 제1스레드(thread)를 처리하기 위해 제1VM(virtual machine)에 할당된 제1액터가 제2액터로 제1펑션 콜(function call)을 하면, 상기 제1펑션 콜을 하기 위해 상기 제1펑션 콜에 대응되는 제1스택정보가 상기 제1스레드 및 상기 제1VM에 대응되는 제1스택에 삽입되는 단계;
상기 제1VM에 할당된 제3액터가 제2스레드를 처리하기 위해 상기 제1펑션 콜이 완료되기 전에 제2펑션 콜을 하는 단계; 및
상기 제2펑션 콜을 하기 위해 상기 제2펑션 콜에 대응되는 제2스택정보가 상기 제2스레드 및 상기 제1VM에 대응되는 제2스택에 삽입되는 단계를 포함하는 멀티버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 2] 제 1항에 있어서, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은,
상기 제1액터가 상기 제1펑션 콜을 하기 위해, 상기 제1VM의 락(Lock)이 해제되고, 상기 제2액터가 할당된 제2VM이 락 되는 단계를 더 포함하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 3] 제 1항에 있어서, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은,
상기 제2펑션 콜이 완료되기 전에 상기 제1펑션 콜이 완료되면, 상기 제1액터는 상기 제1스택정보에 상응하는 프로세싱을 수행하는 단계를 더 포함하는 멀티버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 4] 제 1항에 있어서, 상기 제1펑션 콜 또는 상기 제2펑션 콜 중 적어도 하나는,
리드 콜(read call) 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 5] 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 소정의 스레드가 생성되면 상기 스레드에 VM별 스택을 할당하는 단계;
상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템에 의해 관리되는 소정의 제1액터가 상기 스레드를 처리하면서 소정의 펑션 콜을 하는 단계;
상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 상기 스레드의 상기 제1액터가 할당된 제1VM에 대응되는 스택을 판단하고, 판단된 스택에 상기 펑션 콜에 대응되는 스택정보를 삽입하는 단계를 포함하는 멀티 버추얼 머신을 이용한 멀티

쓰레드 처리방법.

[청구항 6] 제 5항에 있어서, 상기 펑션 콜은,
상기 제1VM과는 다른 제2VM에 할당된 제2액터로의 리드 콜 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 하는 멀티버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 7] 제 6항에 있어서, 상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법은,
상기 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템이 상기 펑션 콜을 처리하기 위해, 상기 제1VM의 락(Lock)을 해제하고, 제2VM을 락 하는 단계를 더 포함하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법.

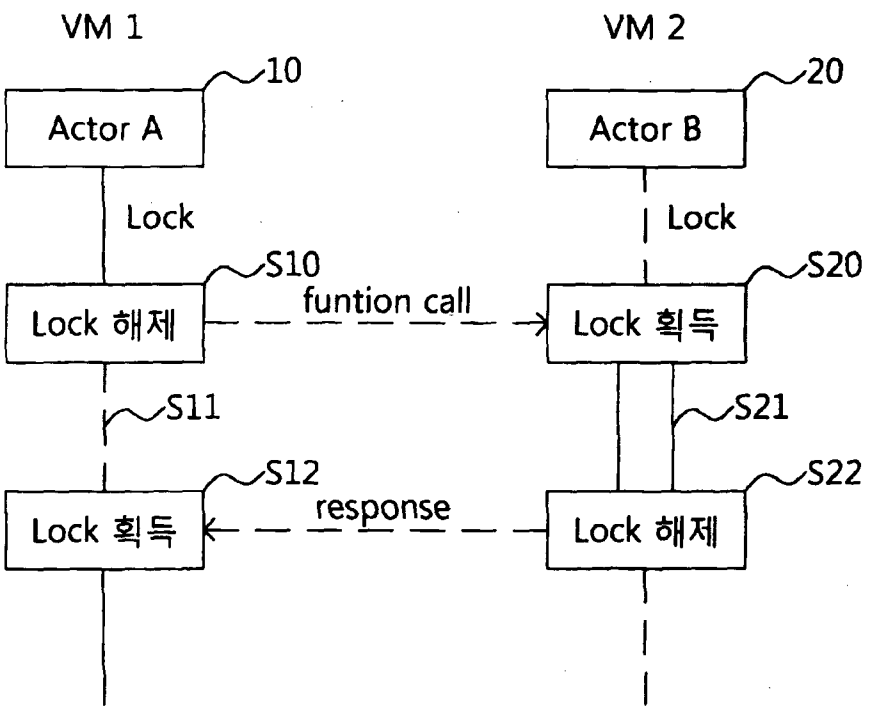
[청구항 8] 제 1항 내지 제 7항 중 어느 한 항에 기재된 방법을 수행하기 위한 프로그램을 기록한 컴퓨터 판독 가능한 기록매체.

[청구항 9] 복수의 VM들;
소정의 스레드가 생성되면 상기 스레드에 VM별 스택을 할당하기 위한 스택 모듈;
복수의 액터들(Actor)을 생성 및 관리하고 상기 복수의 액터들 각각을 상기 복수의 VM들 중 어느 하나에 할당하기 위한 액터 모듈; 및
상기 복수의 액터들 중 어느 하나의 제1액터가 소정의 스레드를 처리하기 위해 펑션 콜을 하면, 상기 스레드 및 상기 제1액터가 할당된 제1VM에 대응되는 스택을 판단하고, 판단된 스택에 상기 펑션 콜에 대응되는 스택정보를 삽입하기 위한 제어모듈을 포함하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템.

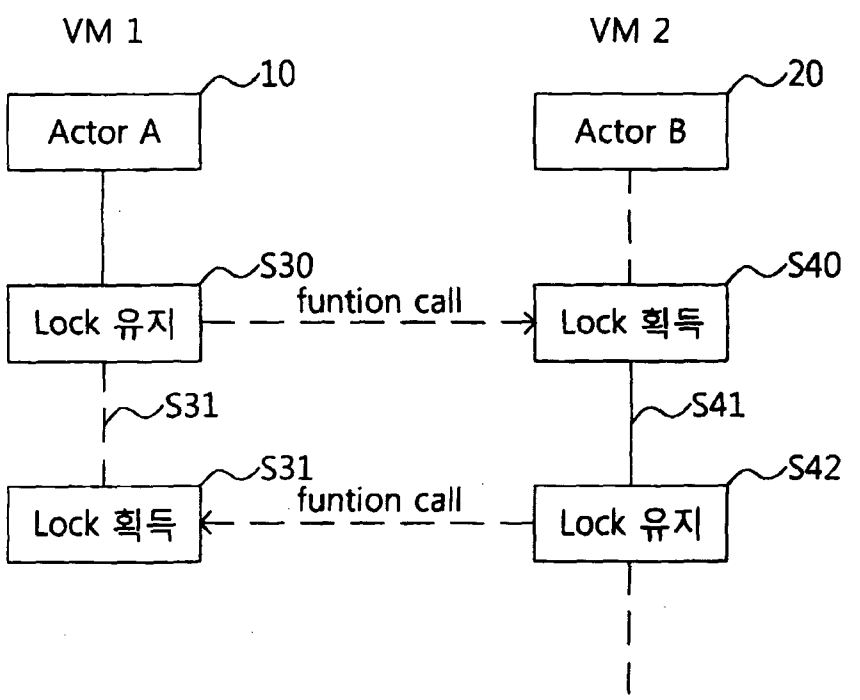
[청구항 10] 제 9항에 있어서, 상기 펑션 콜은,
상기 제1VM과는 다른 제2VM에 할당된 제2액터로의 리드 콜 또는 이벤트 콜 중 적어도 하나인 것을 특징으로 하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리방법.

[청구항 11] 제 10항에 있어서, 상기 제어모듈은,
상기 펑션 콜을 처리하기 위해, 상기 제1VM의 락(Lock)을 해제하고, 제2VM을 락 하는 멀티 버추얼 머신을 이용한 멀티 쓰레드 처리시스템.

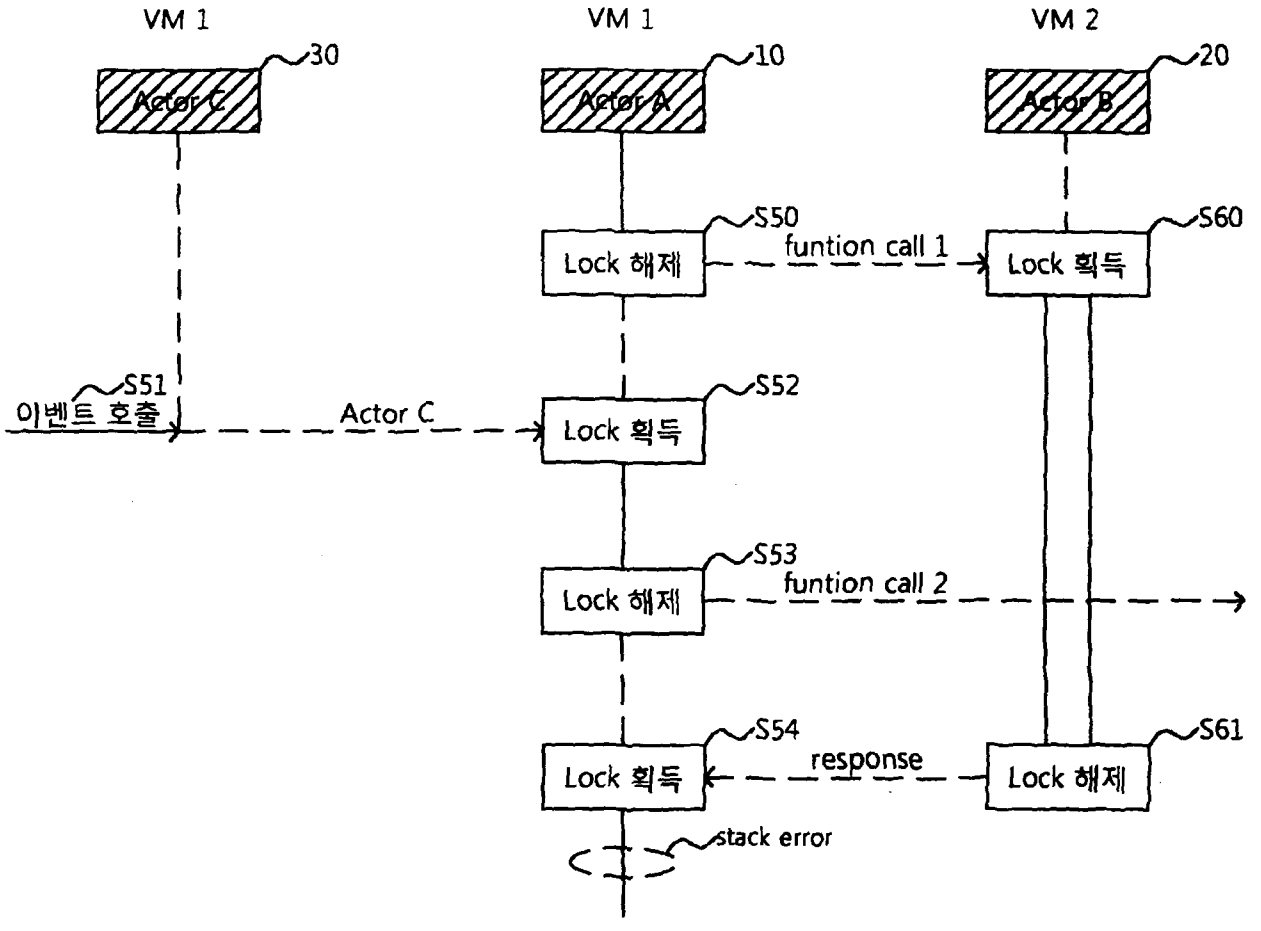
[Fig. 1a]


VM 1
VM 2
10
20
Actor A
Actor B
Lock
Lock
S10
S20
Lock 해제
funtion call
Lock 획득
S11
S21
S12
S22
Lock 획득
response
Lock 해제

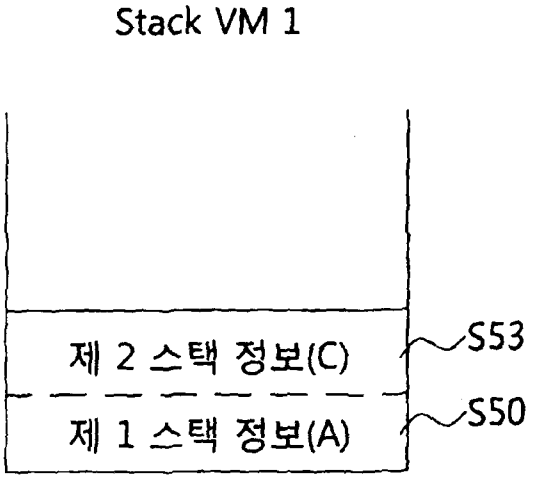
[Fig. 1b]


VM 1
VM 2
10
20
Actor A
Actor B
S30
S40
Lock 유지
funtion call
Lock 획득
S31
S41
S31
S42
Lock 획득
funtion call
Lock 유지

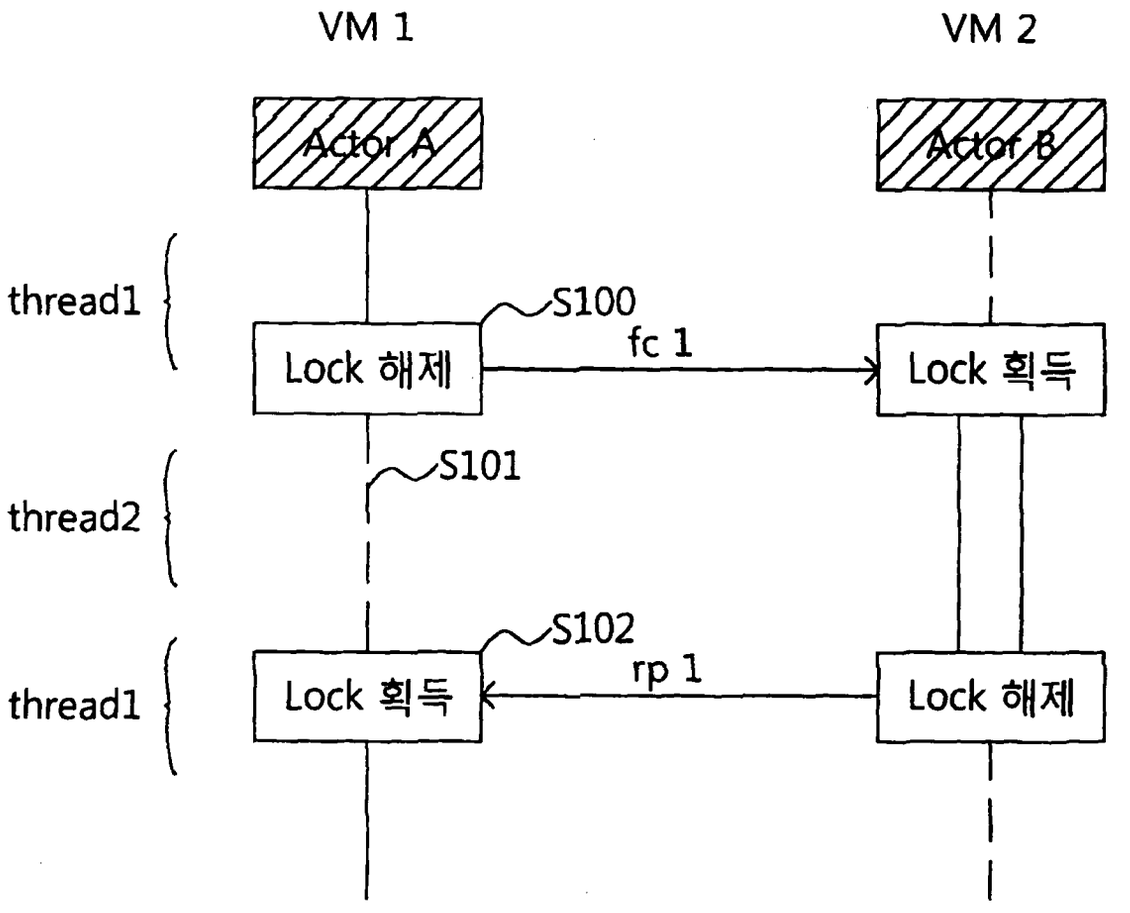
[Fig. 2a]


VM 1
VM 1
VM 2
30
10
20
Actor C
Actor A
Actor B
S50
S60
Lock 해제
funtion call 1
Lock 획득
S51
S52
이벤트 호출
Actor C
Lock 획득
S53
Lock 해제
funtion call 2
S54
S61
Lock 획득
response
Lock 해제
stack error

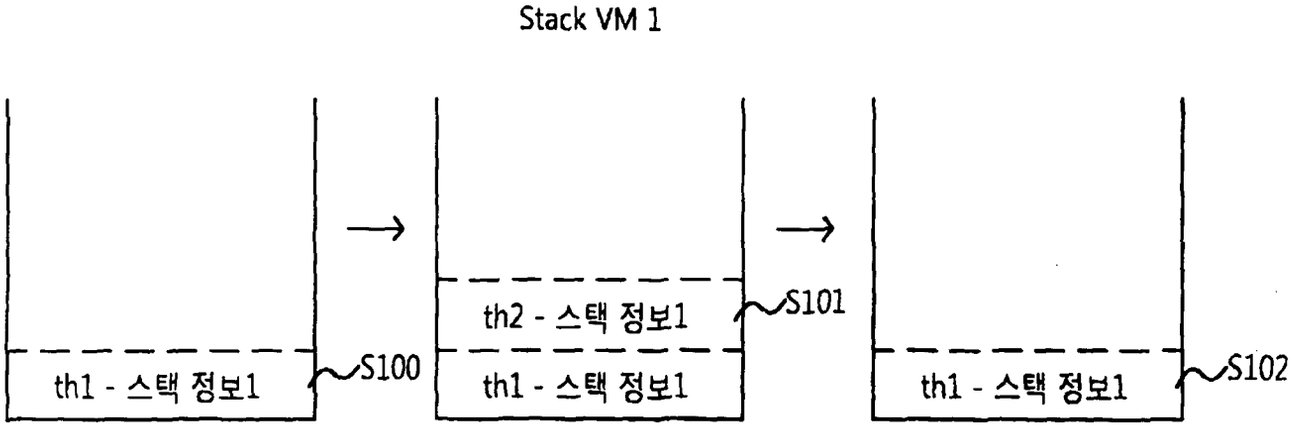
[Fig. 2b]


Stack VM 1
제 2 스택 정보(C)
S53
제 1 스택 정보(A)
S50

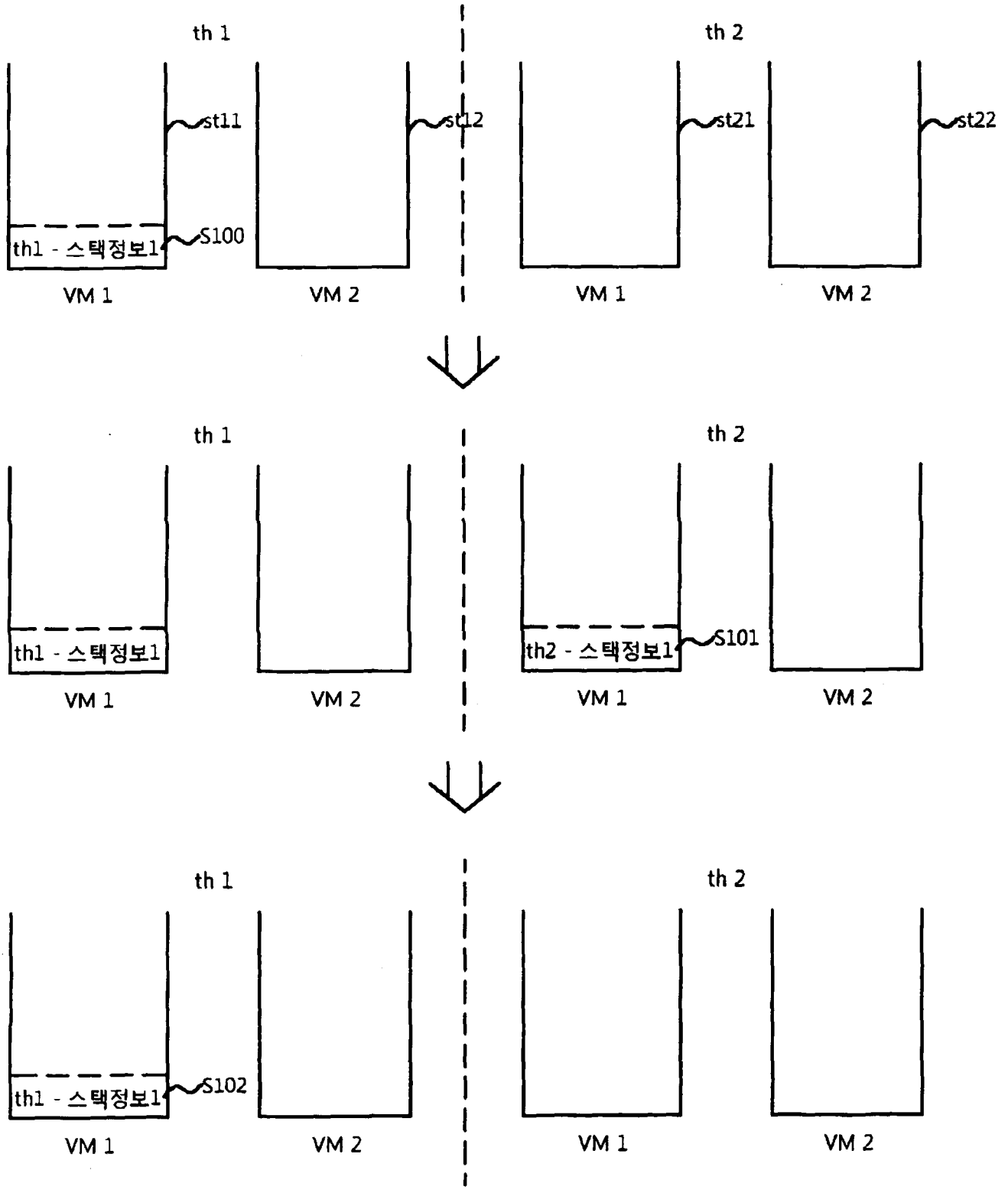
[Fig. 3]


VM 1
VM 2
Actor A
Actor B
thread1
S100
Lock 해제
fc 1
Lock 획득
thread2
S101
thread1
S102
Lock 획득
rp 1
Lock 해제

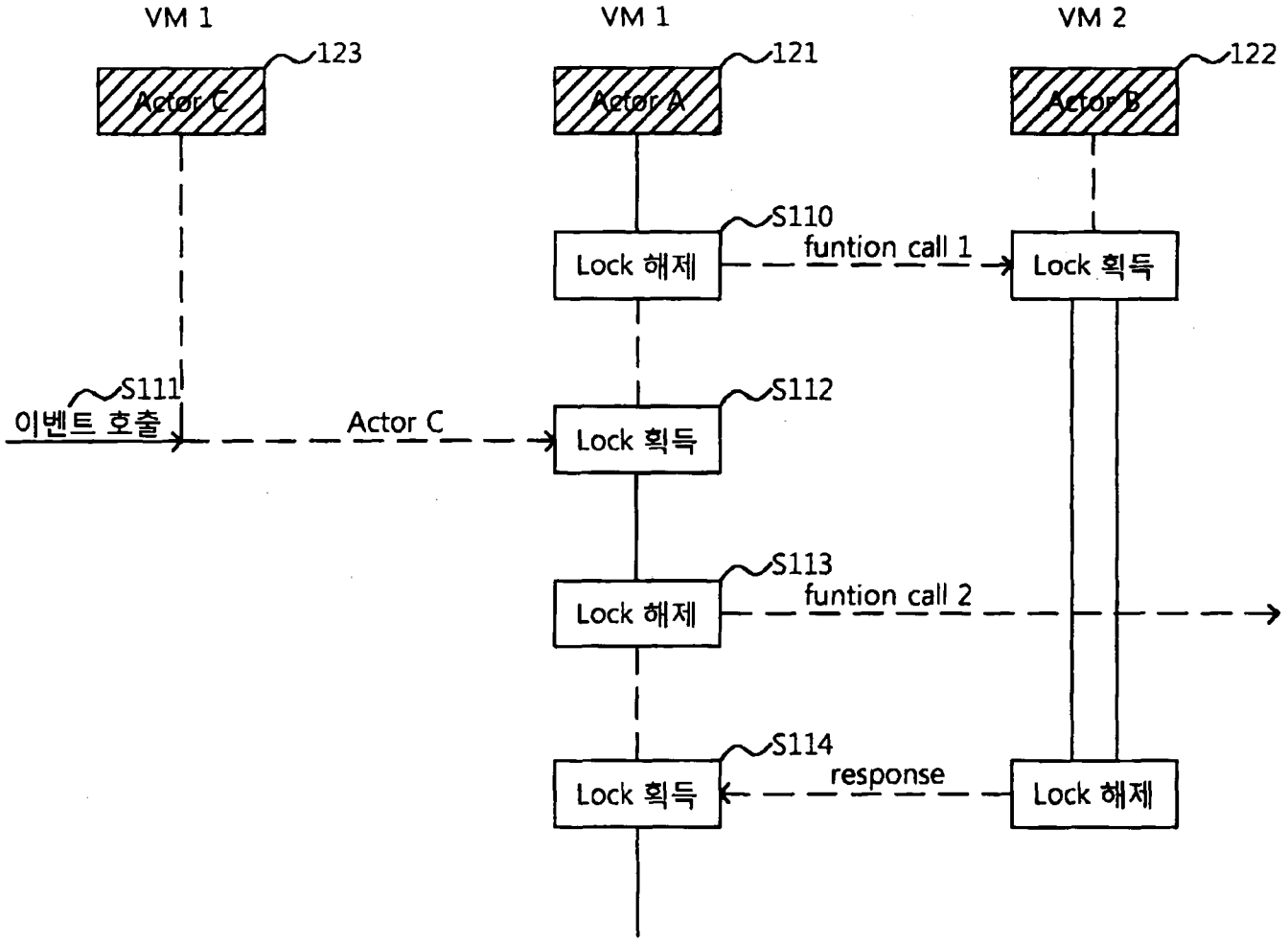
[Fig. 4]


Stack VM 1
th1 - 스택 정보1
S100
th2 - 스택 정보1
S101
th1 - 스택 정보1
th1 - 스택 정보1
S102

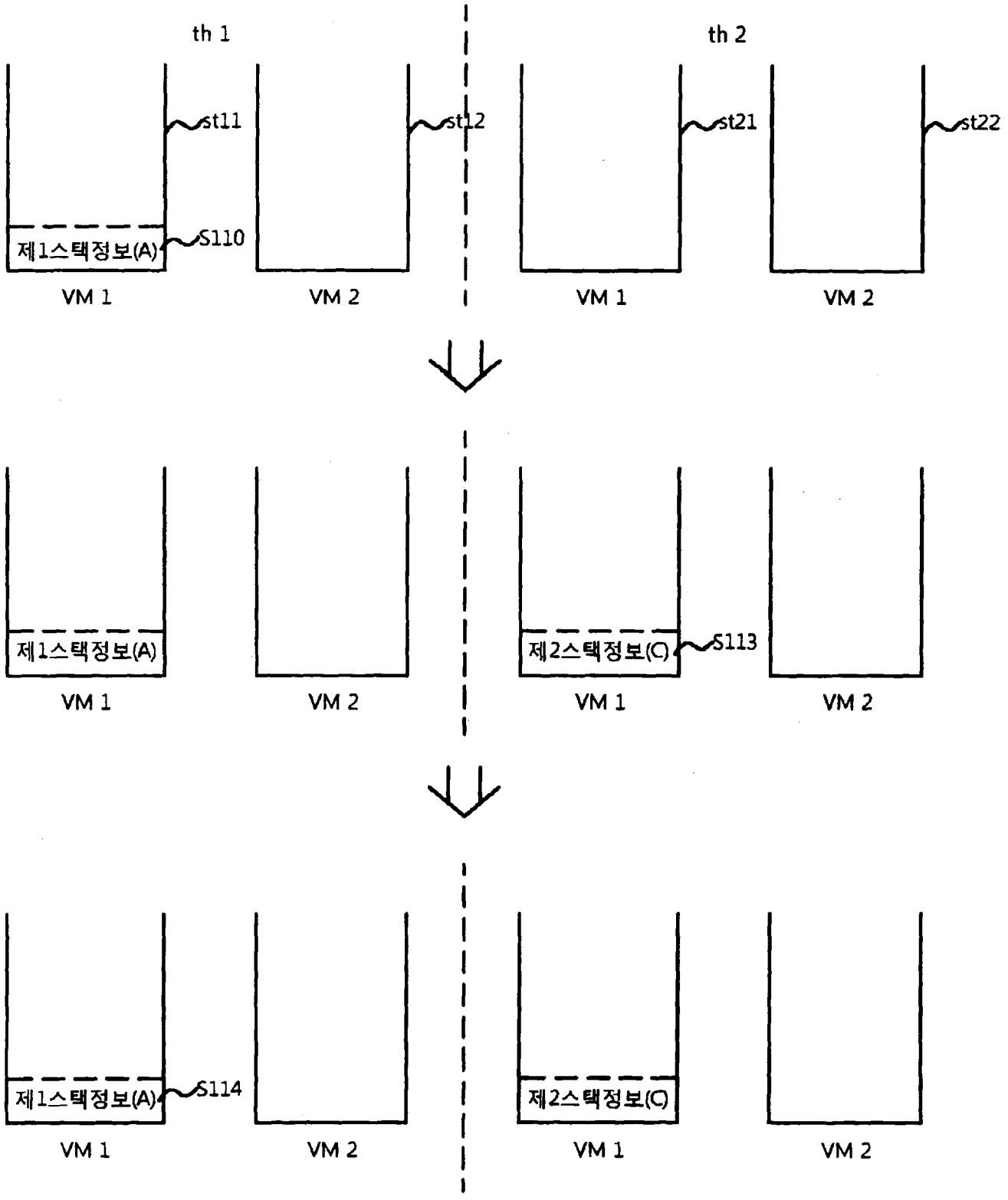
[Fig. 5]


th 1
th 2
st11
st12
st21
st22
th1 - 스택정보1
S100
VM 1
VM 2
VM 1
VM 2
th 1
th 2
th1 - 스택정보1
th2 - 스택정보1
S101
VM 1
VM 2
VM 1
VM 2
th 1
th 2
th1 - 스택정보1
S102
VM 1
VM 2
VM 1
VM 2

[Fig. 6]


VM 1
VM 1
VM 2
123
121
122
Actor C
Actor A
Actor B
S110
Lock 해제
funtion call 1
Lock 획득
S111
이벤트 호출
Actor C
S112
Lock 획득
S113
Lock 해제
funtion call 2
S114
Lock 획득
response
Lock 해제

[Fig. 7]


th 1
th 2
st11
st12
st21
st22
제1스택정보(A)
S110
VM 1
VM 2
VM 1
VM 2
제1스택정보(A)
제2스택정보(C)
S113
VM 1
VM 2
VM 1
VM 2
제1스택정보(A)
S114
제2스택정보(C)
VM 1
VM 2
VM 1
VM 2

[Fig. 8]