(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2012년06월22일
(11) 등록번호 10-1159322
(24) 등록일자 2012년06월18일

(51) 국제특허분류(Int. Cl.)
G06F 15/16 (2006.01)
(21) 출원번호 10-2005-0052333
(22) 출원일자 2005년06월17일
심사청구일자 2010년06월10일
(65) 공개번호 10-2006-0048419
(43) 공개일자 2006년05월18일
(30) 우선권주장
10/871,134 2004년06월18일 미국(US)
(56) 선행기술조사문헌
KR1020040089583 A
KR1020050071358 A

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
로츠, 자콥 알
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
호웰, 조나단 알.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
도우설, 존 알.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
(74) 대리인
제일특허법인

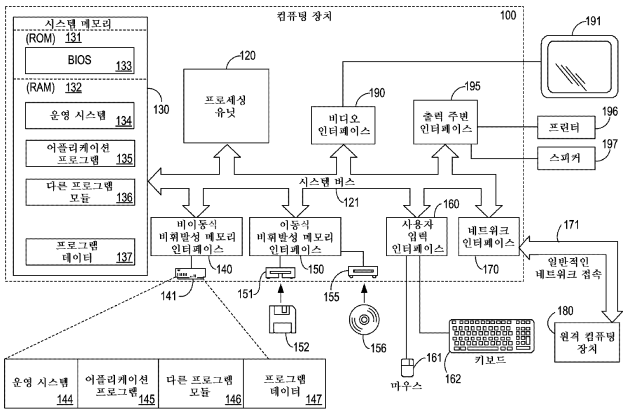
전체 청구항 수 : 총 21 항 심사관 : 지정훈

(54) 발명의 명칭 **오동작 허용 분산 컴퓨팅 시스템에서의 레플리카 세트의효율적인 변경**

(57) 요 약

분산 컴퓨팅 시스템은 일 세트의 컴퓨팅 장치를 사용하여 오동작 허용 방식으로 동작될 수 있다. 일 세트의 컴퓨팅 장치는 상태 머신의 동일한 레플리카(replica)를 구현하고 제안들을 선택함으로써 다수의 오동작을 허용할 수 있다. 레플리카를 호스팅(host)함으로써 분산 컴퓨팅 시스템에 참여하고 있는 일 세트의 컴퓨팅 장치는, 컴퓨팅 장치를 이 세트로부터 제거하거나 추가함으로써, 또는 참여하기 위한 특정 컴퓨팅 장치를 규정함으로써 수정될 수 있다. 이 세트에서 참여하고 있는 컴퓨팅 장치를 변경하는 것은 결점이 있는 장치를 동작가능한 장치로 교체하거나 또는 시스템 내의 리던던시(redundancy)의 양을 증가시킴으로써 오동작 허용을 증가시킨다.

대 표 도



특허청구의 범위

청구항 1

컴퓨팅 장치들의 제1 세트를 포함하는 고장 방지 분산 컴퓨팅 시스템(fault-tolerant distributed computing system) - 상기 제1 세트 내의 각각의 컴퓨팅 장치는 상태 머신의 레플리카(replica)를 실행하고, 상기 상태 머신 상에서 실행될 동작들은 상기 제1 세트의 리더(leader)에 의해 제안됨 - 에서, 컴퓨팅 장치들의 제2 세트 - 상기 제2 세트 내의 각각의 컴퓨팅 장치는 상기 상태 머신의 레플리카를 실행함 - 를 생성하는 방법에 있어서,

상기 제1 세트 내의 컴퓨팅 장치에 의해 상기 제2 세트를 생성하도록 요청하는 단계와,

상기 제1 세트 내의 정족수(quorum)의 장치들 중에서 상기 제2 세트를 포함하는 상기 컴퓨팅 장치들이 상기 상태 머신의 소정의 스텝(step) 이후에 상기 상태 머신을 실행하는 것에 동의하는 단계와,

상기 제1 세트의 상기 리더에 의해 상기 상태 머신의 상기 소정의 스텝에 이르기까지 수행될 널 동작들(null operation)의 개수를 제안하는 단계를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 방법은, 상기 방법이 시작되는 시기를 기술(describing)하는 정책(a policy)에 따라서 자동으로 개시되는, 방법.

청구항 3

제1항에 있어서,

상기 방법은 명백한 사용자 요청에 응답하여 개시되는, 방법.

청구항 4

제1항에 있어서,

상기 제2 세트 내의 하나 이상의 컴퓨팅 장치들에 의해 상기 제1 세트 내의 하나 이상의 컴퓨팅 장치들의 신뢰성(authenticity)을 증명(verifying)하는 단계를 더 포함하는 방법.

청구항 5

제1항에 있어서,

상기 리더에 의해 제안된 동작들은 상기 제1 세트 내의 정족수의 컴퓨팅 장치들이 상기 제안에 동의하는 경우에만 상기 상태 머신에 의해 수행되는, 방법.

청구항 6

복수의 복제된(replicated) 컴퓨팅 장치들을 포함하는 고장 방지 분산 컴퓨팅 시스템 - 상기 복제된 컴퓨팅 장치들의 제1 세트는 단계들 중 제1 시퀀스에서 상태 머신에 의해 수행될 동작들을 결정하고, 상기 복제된 컴퓨팅 장치들의 제2 세트는 상기 상태 머신에 의해 나중에 단계들 중 제2 시퀀스에서 수행될 동작들을 결정하며, 상기 제1 시퀀스 및 제2 시퀀스는 서로 배타적임 - 내의, 제1 복제된 컴퓨팅 장치 상에서 실행되기 위한 컴퓨터 실행가능 명령어들을 포함하는 컴퓨터 판독가능 저장 매체에 있어서,

상기 컴퓨터 실행가능 명령어들은,

상기 상태 머신의 단계들에서 동작들을 실행하기 위한 실행 모듈과,

상기 제1 시퀀스에서의 동작들을 결정하기 위하여 상기 제1 세트 내의 다른 복제된 컴퓨팅 장치들과 조정(coordinating)하기 위한 제1 동의 모듈(agreement module)에 의해 성능을 향상시키는, 컴퓨터 판독가능 저장 매체.

청구항 7

제6항에 있어서,

상기 컴퓨터 실행가능 명령어들은,

상기 제2 시퀀스에서의 동작들을 결정하기 위하여 상기 제2 세트 내의 다른 복제된 컴퓨팅 장치들과 조정하기 위한 제2 동의 모듈에 의해 더 성능을 향상시키는, 컴퓨터 판독가능 저장 매체.

청구항 8

제6항에 있어서,

상기 제1 동의 모듈은, 상기 실행 모듈이 상기 제1 시퀀스의 단계들을 실행한 이후에 파괴되는, 컴퓨터 판독가능 저장 매체.

청구항 9

제8항에 있어서,

상기 파괴된 제1 동의 모듈은 상기 복제된 컴퓨팅 장치들의 제2 세트에 관한 정보를 보유하는, 컴퓨터 판독가능 저장 매체.

청구항 10

제8항에 있어서,

상기 실행 모듈은 어떠한 동의 모듈도 상기 복제된 컴퓨팅 장치 상에 남아 있지 않은 경우 파괴되는, 컴퓨터 판독가능 저장 매체.

청구항 11

제8항에 있어서,

파괴되기 전에, 상기 제1 동의 모듈은, 상기 복제된 컴퓨팅 장치들의 제2 세트가 상기 상태 머신에 의해 수행될 동작들을 결정하는 것을 보증하기 위하여 상기 제2 세트에서 하나 이상의 복제된 컴퓨팅 장치들을 폴링(poll)하는, 컴퓨터 판독가능 저장 매체.

청구항 12

제11항에 있어서,

파괴되기 전에, 상기 제1 동의 모듈은, 상기 제2 세트 내의 정족수의 복제된 컴퓨팅 장치들이 상기 단계들 중 제1 시퀀스에서 최종 동작의 수행을 즉시 뒤따르는 상기 상태 머신의 상태를 안정한 저장소에 저장하는 것을 보증하는, 컴퓨터 판독가능 저장 매체.

청구항 13

제6항에 있어서,

상기 복제된 컴퓨팅 장치들의 제1 세트는 리더 장치를 포함하고, 상기 제1 시퀀스 내의 동작은, 상기 제1 세트 내의 정족수의 복제된 컴퓨팅 장치들이 상기 리더 장치에 의한 동작에 대한 제안에 동의하는 경우에만 상기 상태 머신에 의해 수행되는, 컴퓨터 판독가능 저장 매체.

청구항 14

복수의 복제된 컴퓨팅 장치를 포함하는 고장 방지 분산 컴퓨팅 시스템 - 상기 복제된 컴퓨팅 장치들의 제1 세트는 단계들 중 제1 시퀀스에서 상태 머신에 의해 수행될 동작들을 결정하고, 상기 복제된 컴퓨팅 장치들의 제2 세트는 단계들 중 제2 시퀀스에서 상기 상태 머신에 의해 수행될 동작들을 결정하며, 상기 제1 시퀀스 및 제2 시퀀스는 서로 배타적이며, 상기 단계들 중 제1 시퀀스는 상기 단계들 중 제2 시퀀스를 선행함 - 내의 복제된 컴퓨팅 장치에 있어서,

상기 상태 머신의 동작들을 실행하기 위한, 상기 복제된 컴퓨팅 장치와 연관된 저장 시설 상의 실행 모듈; 및

상기 제1 시퀀스에서의 동작들을 결정하기 위하여 상기 제1 세트 내의 다른 복제된 컴퓨팅 장치들과 조정하기 위한, 상기 복제된 컴퓨팅 장치와 연관된 저장 시설 상의 제1 동의 모듈을 포함하는 복제된 컴퓨팅 장치.

청구항 15

제14항에 있어서,

상기 제2 시퀀스에서의 동작들을 결정하기 위하여 상기 제2 세트 내의 다른 복제된 컴퓨팅 장치들과 조정하기 위한, 상기 복제된 컴퓨팅 장치와 연관된 저장 시설 상의 제2 동의 모듈을 더 포함하는 복제된 컴퓨팅 장치.

청구항 16

제15항에 있어서,

상기 실행 모듈은, 일 단계가 상기 제1 세트에 의해 결정될 때까지 널 동작들을 수행하고, 상기 결정된 단계 이후에는 상기 제2 시퀀스의 동작들을 수행하는, 복제된 컴퓨팅 장치.

청구항 17

제14항에 있어서,

상기 제1 동의 모듈은, 상기 실행 모듈이 상기 제1 시퀀스의 단계들을 실행한 이후에 파괴되는, 복제된 컴퓨팅 장치.

청구항 18

제17항에 있어서,

파괴되기 전에, 상기 제1 동의 모듈은, 상기 복제된 컴퓨팅 장치들의 제2 세트가 상기 상태 머신에 의해 수행될 동작들을 결정하는 것을 보증하기 위하여 상기 제2 세트에서 하나 이상의 복제된 컴퓨팅 장치들을 폴링(poll)하는, 복제된 컴퓨팅 장치.

청구항 19

제18항에 있어서,

파괴되기 전에, 상기 제1 동의 모듈은, 상기 제2 세트 내의 정족수의 복제된 컴퓨팅 장치들이 상기 단계들 중 제1 시퀀스에서 최종 동작의 수행을 즉시 뒤따르는 상기 상태 머신의 상태를 안정한 저장소에 저장하는 것을 보증하는, 복제된 컴퓨팅 장치.

청구항 20

제17항에 있어서,

상기 파괴된 제1 동의 모듈은 상기 복제된 컴퓨팅 장치들의 제2 세트에 관한 정보를 보유하는, 복제된 컴퓨팅 장치.

청구항 21

제17항에 있어서,

상기 실행 모듈은 어떠한 동의 모듈도 상기 복제된 컴퓨팅 장치 상에 남아 있지 않은 경우 파괴되는, 복제된 컴퓨팅 장치.

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0022] 본 발명은 일반적으로 분산 컴퓨팅에 관한 것이고, 보다 구체적으로는, 오동작 허용 분산 컴퓨팅에 관한 것이다.

[0023] 분산 시스템의 이점은 하나의 일체식 컴퓨팅 장치의 기능을 마비시키는 물리적인 어려움에 직면하더라도 계속적으로 동작할 수 있는 능력이다. 이러한 어려움은 연속된 정전, 혹독한 기후, 홍수, 테러 활동 등을 포함할 수 있다.

[0024] 개개의 멤버 컴퓨팅 장치들이 네트워크로부터 끊어지거나 전원이 꺼지거나, 시스템 오동작을 겪고, 또는 그렇지 않으면 이상하게 되는 증가된 위험을 보상하기 위해서는, 리던던시가 분산 컴퓨팅 시스템이 계속 동작하도록 하기 위하여 사용될 수 있다. 따라서, 임의의 하나의 컴퓨팅 장치에서 실행되는 프로세스나 저장된 정보는 여분으로 부가적인 컴퓨팅 장치에 저장되어, 심지어 컴퓨팅 장치 중 하나가 동작하지 않더라도 정보에 계속 액세스할 수 있게 된다.

[0025] 분산 컴퓨팅 시스템은 완벽한 리던던시를 실행할 수 있는데, 이러한 시스템 내의 모든 장치는 동일한 일을 수행하고 동일한 정보를 저장한다. 이러한 시스템은 심지어 장치 중 거의 절반이 동작하지 않더라도 사용자가 유용한 동작을 계속 실행하도록 할 수 있다. 또는, 이러한 시스템은 동일한 정보의 많은 복사품이 지리적 영역에 걸쳐 분포되도록 사용될 수 있다. 예를 들어, 다국적 기업은 세계적인 분산 컴퓨팅 시스템을 확립할 수 있다.

[0026] 그러나, 분산 컴퓨팅 시스템은 이러한 시스템을 포함하는 개개의 장치들이 동일한 순서로 동일한 동작을 실행하는 것을 적절히 보증해야 한다는 복잡성 때문에 유지하는데 어려움이 있을 수 있다. 종종 발생하는 이러한 어려운 일을 촉진하기 위해서는, 상태 머신 접근법(state machine approach)이 개개의 장치들 사이의 활동을 조정하기 위해 종종 사용된다. 상태 머신은 응답/상태 쌍을 각각의 명령/상태 쌍에 연결시키는 일 세트의 상태, 일 세트의 명령, 일 세트의 응답 및 클라이언트 명령에 의해 설명될 수 있다. 상태 머신은 그 상태를 변경하여 응답을 생성함으로써 명령을 실행할 수 있다. 따라서, 상태 머신은 막 실행하려고 하는 행동 및 현 상태에 의해 완벽하게 설명될 수 있다.

[0027] 그러므로, 상태 머신의 현 상태는 명령이 실행된 순서, 그때 이후로 실행된 명령 및 이전 상태에 달려있다. 2 이상의 상태 머신 사이의 동기화를 유지하기 위해, 공통의 초기 상태가 확립될 수 있고, 초기 상태로 시작하는 각각의 상태 머신은 동일한 순서로 동일한 명령을 실행할 수 있다. 따라서, 하나의 상태 머신을 다른 상태 머신과 동기화시키기 위해서는, 다른 상태 머신에 의해 실행되는 명령이 결정될 필요가 있다. 그러므로, 동기화 문제는 실행될 명령의 순서를 결정하는 문제가 되고, 보다 구체적으로는, 주어진 단계에 대하여 실행되는 특정 명령을 결정하는 문제가 된다.

[0028] 주어진 단계에 대하여 어떤 명령이 실행될 것인가를 결정하기 위한 하나의 메커니즘이 팩서스 알고리즘(Paxos algorithm)으로 알려져 있다. 팩서스 알고리즘에서, 개개의 장치 중 어느 하나는 리더(leader)로서 동작하여 시스템 내의 모든 장치들에 의해 실행되기 위한 주어진 클라이언트 명령을 제안하려고 할 수 있다. 이러한 모든 제안은 더 쉽게 제안들을 트랙킹(track)하기 위해 제안 번호를 첨부하여 보내질 수 있다. 이러한 제안 번호는 장치들이 실행할 명령에 동의하려는 특정 단계와 관련될 필요는 없다. 최초, 리더는 리더가 제출하려고 하는 제안에 대한 제안 번호를 제시할 수 있다. 이후 나머지 장치들 각각은 그들이 찬성하려는 최종 제안 또는 그들이 어떠한 제안에도 찬성하지 않으려는 지시를 갖고 있는 리더의 제안 번호의 제시에 응답할 수 있

다. 다양한 응답을 통하여 리더가 장치들에 의해 찬성되는 다른 제안들을 알지 못한다면, 리더는 이전 메시지에서 제시된 제안 번호를 사용하여 주어진 클라이언트 명령이 장치들에 의해 실행되고 있다고 제안할 수 있다. 이 상태에서, 각각의 장치는 이러한 동작에 찬성할지 또는 거부할지를 결정한다. 다른 리더의 더 높은 제안 번호의 제시에 응답했다면 장치는 이러한 동작을 단지 거부해야 한다. 정족수로 알려진 장치의 충분한 수가 이러한 제안에 찬성한다면, 제안된 동작은 동의된 것이고, 각각의 장치는 그 동작을 실행하여 결과를 전송할 수 있다. 이러한 방법으로 장치들 각각은 모든 장치들 사이에서 동일한 상태를 유지하면서 동일한 순서로 동작을 실행할 수 있다.

[0029] 일반적으로, 팩서스 알고리즘은 두 상태로 고려될 수 있는데, 초기 상태에서는, 상술한 바와 같이, 리더가 장치들에 의해 찬성된 이전 제안을 알 수 있고, 두번째 상태에서는, 리더가 실행하기 위한 클라이언트 명령을 제안할 수 있다. 일단 리더가 이전 제안을 알게 되면, 리더는 초기 상태를 계속 반복할 필요가 없다. 대신, 리더는 여러 단계에서 분산 컴퓨팅 시스템에 의해 실행될 수 있는 일련의 클라이언트 명령을 제안하면서 두번째 상태를 계속 반복할 수 있다. 이러한 방법으로, 각각의 단계에 대하여 분산 컴퓨팅 시스템에 의해 실행되는 각각의 클라이언트 명령이 팩서스 알고리즘의 한 경우라고 고려될 수 있는 반면, 리더는 다음 단계에 대하여 다른 클라이언트 명령을 제안하기 전에 주어진 단계에 대하여 제안된 클라이언트 명령에 장치들이 표결하기를 기다릴 필요가 없다.

[0030] 전체적으로 분산 컴퓨팅 시스템은 상태 머신으로 모델링될 수 있다. 따라서, 완벽한 리던던시를 구현하는 분산 컴퓨팅 시스템은 전체 시스템의 상태를 복제하는 장치 각각을 가질 수 있어, 각 장치가 동일한 상태 머신 자체의 "레플리카(replica)"를 호스팅한다. 이 시스템은 각각의 레플리카가 동일한 상태를 유지하기를 요구한다. 어떤 레플리카가 하나의 클라이언트 명령이 실행되었다고 믿는 반면, 제2 그룹의 레플리카는 다른 클라이언트 명령이 실행되었다고 믿는다면, 전체 시스템은 더 이상 하나의 상태 머신으로서 동작하지 않는다. 이러한 상황을 피하기 위하여, 다수의 레플리카는 일반적으로 시스템에 의해 실행되도록 제안된 클라이언트 명령을 선택하도록 요구받을 수 있다. 각각 다수인 레플리카의 임의의 두 그룹이 적어도 하나의 레플리카를 공유하기 때문에, 각각 다수의 레플리카를 포함하는 두 그룹이 상이하게 제안된 클라이언트 명령을 선택하지 못하도록 적어도 하나의 공통된 레플리카에 의존하는 팩서스 알고리즘과 같은 메커니즘이 구현될 수 있다.

발명이 이루고자 하는 기술적 과제

[0031] 그러나, 연장된 기간 동안 동작하기로 기대되는 시스템은 영원히 은퇴하거나 그렇지 않으면 영원히 동작하지 않게 되는 장치를 처리해야만 한다. 그러므로, 시스템은 시스템의 미래의 오동작에 대한 허용치가 초기 수준으로 회복되도록 어떤 장치들을 교환하여 다른 것과 교체하는 수단(즉, 상태 머신 레플리카 세트를 변경하는 것)을 제공하여야 한다. 그렇지 않으면, 일단 정족수의 오동작이 발생하면, 시스템은 더 이상 오동작을 허용하지 않는다. 동작하지 않는 장치들을 교체하는 것은 새로운 오동작을 허용하기 위한 오동작 허용치를 회복시킨다. 부가적으로, 시스템은 새로운 장치의 추가가 허용가능한 장치의 오동작의 수를 증가시키도록 하여 더 많은 오동작 허용치를 제공해야 한다. 원래의 팩서스 알고리즘은 시스템이 프로토콜에 참여하고 있는 상태 머신 레플리카 세트를 어떻게 변경할지를 결정하는 단순한 설명을 제공한다. 그러나, 원래의 팩서스 알고리즘에 의해 제기된 접근법은 레플리카 세트에서 변경하는 동안 일어날 수 있는 다양한 특별 상황을 설명하지는 못한다.

[0032] 분산 컴퓨팅 시스템은 일 세트의 컴퓨팅 장치를 사용하여 오동작 허용 방식으로 동작할 수 있다. 일 세트의 컴퓨팅 장치는 팩서스 알고리즘을 사용하여 상태 머신의 동일한 레플리카를 구현하여 제안을 선택함으로써 다수의 오동작을 허용할 수 있다. 본 발명의 실시예들은 분산 컴퓨팅 시스템에 참여하고 있는 일 세트의 컴퓨팅 장치가 세트로부터 컴퓨팅 장치를 제거 또는 추가하거나, 참여하고 있는 특정 컴퓨팅 장치를 규정함으로써 수정되도록 한다. 세트에 참여하고 있는 컴퓨팅 장치를 변경하는 것은 결함이 있는 장치를 동작가능한 장치로 교체하거나 시스템 내의 리던던시의 양을 증가시킴으로써 오동작 허용치를 증가시킨다.

발명의 구성 및 작용

[0033] 그러므로, 본 발명의 일 실시예에서, 제1 세트의 컴퓨팅 장치는 일 시퀀스의 단계가 장치들 중에 복제되고 있는 상태 머신에 의해 실행되는지를 결정하는 것을 담당한다. 제2 세트의 장치는 제1 세트로부터의 담당을 제거하고 상태 머신에 의해 실행될 일 시퀀스의 단계에서 결정하는 단계에 대한 담당을 맡음으로써 규정될 수 있다. 제1 세트가 담당하는 단계의 범위는 제2 세트가 담당하는 단계의 범위와 상호 배타적이다.

[0034] 본 발명의 다른 실시예에서, 상태 머신에 의해 실행될 일 시퀀스의 단계를 결정하는 것을 담당하는 제1 세트

의 컴퓨팅 장치의 리더는 컴퓨팅 장치의 새로운 세트가 그 세트를 계승하도록 선택되었음을 안다. 새로운 세트는 일 시퀀스의 단계를 결정하는 담당이 특정 단계에서 시작하는 것으로 가정하고, 제1 세트의 리더는 널 동작(null operation)이 특정 단계 이전 단계에서 실행될 단계로서 결정되도록 한다.

[0035] 본 발명의 또 다른 실시예에서, 동일한 상태 머신의 레플리카를 행하는 컴퓨팅 장치들은 실행 모듈 및 하나 이상의 동의 모듈을 호스팅한다. 각 장치의 실행 모듈은 순차적으로 복제된 상태 머신의 단계들을 실행한다. 각 장치는 멤버인 레플리카의 각 세트에 대응하는 동의 모듈을 갖는다. 동일한 머신 상의 다른 레플리카 세트의 동의 모듈은 각각 일 시퀀스의 단계들에 대한 선택된 동작을 제공하기 위해 그 머신 상의 하나의 실행 모듈과 통신한다.

[0036] 비록 여기에서의 설명은 주로 분산 컴퓨팅 시스템에서의 컴퓨팅 장치의 동작에 초점을 맞추고 있지만, 이러한 설명이 개별 프로세서나 개별 메모리 공간에서와 같은 단일 컴퓨팅 장치에서 구동하는 프로세스들에도 동등하게 적용됨을 이해할 것이다. 따라서, 추가적인 실시예들은 다중 프로세서가 물리적으로 하나 이상의 컴퓨팅 장치에 위치하는 다중 프로세서 환경, 그리고 다중 가상 머신이 하나 이상의 컴퓨팅 장치에 의해 실행되는 다중 가상 머신 환경에서 수정된 팩서스 알고리즘의 동작을 포함한다. 본 발명의 부가적인 특징 및 이점은 다음의 첨부된 도면을 참조하여 설명되는 예시적인 실시예들의 상세한 설명으로부터 분명해질 것이다.

[0037] 첨부된 청구의 범위가 본 발명의 특징을 설명하고 있지만, 주로 본 발명의 목적 및 이점과 함께 본 발명은 첨부된 도면을 참조하여 다음의 상세한 설명으로부터 가장 잘 이해될 수 있다.

[0038] 분산 컴퓨팅 시스템은 시스템에 참여하기에 충분한 프로세서 및 저장 능력을 갖는 다수의 개별 퍼스널 컴퓨팅 장치, 서버 컴퓨팅 장치, 또는 다른 장치들을 포함할 수 있다. 분산 컴퓨팅 시스템은 구성 컴퓨팅 장치들이 상당히 증가된 프로세싱 능력 및 저장 공간을 제공하거나, 또는 다수의 장치들이 동일한 정보에 대하여 액세스하도록 하는 리던던시를 구현하는 능력을 모을 수 있다. 따라서, 분산 컴퓨팅 시스템에 대한 하나의 공통된 사용은 공통 네트워크에 부착된 다수의 다른 퍼스널 컴퓨팅 장치들의 저장 공간 및 프로세싱 능력을 모으는 것이다. 이러한 분산 컴퓨팅 시스템은 어떤 장치가 현재 시스템의 일부이고, 주어진 일 세트의 정보가 어떤 장치에 저장되는지와 같은 시스템에 관한 정보를 보유할 수 있다. 이러한 정보는 장치들이 그 능력 및 저장 공간을 모으는데 필요할 수 있고, 결과적으로, 각 장치는 복제품을 포함할 수 있다. 시스템의 장치들 중의 정보의 동기화는 이하 설명될 상태 머신 접근법을 통하여 촉진될 수 있다.

[0039] 또는, 점차 분산 컴퓨팅 시스템에 대한 공통된 사용은 다양한 형태의 정보에 대한 중앙 저장소로서 동작할 수 있는 네트워크 서버의 사용이다. 이러한 분산 시스템은 시스템의 구성 장치들 모두에 중앙 저장소를 복제하려고 하므로, 중앙 저장소와 통신하려고 하는 모든 클라이언트는 통신할 편리하고 효율적인 장치를 찾을 수 있다. 게다가, 시스템의 분산 성질 때문에, 정전, 홍수, 정치적 폭동 등과 같은 로컬 이벤트가 전체 시스템이 계속 적절하게 동작하여 정보에의 액세스 및 다른 서비스를 클라이언트에게 제공하도록 하면서 소수의 컴퓨팅 장치에만 영향을 미칠 수 있다.

[0040] 이러한 분산 컴퓨팅 시스템은 상태 머신으로서 고려될 수 있고, 머신의 미래의 상태는 취해질 현 상태 및 동작에 의해 규정된다. 분산 컴퓨팅 시스템의 각 구성 장치는 이후 독립적으로 전체 시스템의 상태 머신을 실행한다. 상태 머신 접근법은 비동기적으로 구현될 수 있어, 구성 장치들에 걸쳐 정확한 동시성이 유지될 필요는 없고, 장치들 사이의 동기화가 모든 장치에 대한 초기 상태를 설정한 후 동일한 순서로 동일한 기능을 실행하도록 함으로써 달성될 수 있다. 동기화를 유지하기 위한 공통의 방법은 분산 컴퓨팅 시스템의 구성 장치들이 모두 그 기능을 실행하기 전에 다음 기능에 동의하고 실행된 기능의 리스트를 유지하도록 하는 것이다. 이러한 방법으로, 모든 장치가 동일한 상태를 가질 수 있고, 장치가 동작하지 않는다면 단지 실행한 마지막 기능을 결정하고, 리스트로부터 마지막 기능 이래로 동의된 기능을 인식하여 그 기능을 실행할 필요가 있다.

[0041] 서버로서 동작하는 분산 컴퓨팅 시스템은 다국적 회사용 중앙 데이터베이스 또는 인기있는 월드와이드웹 사이트와 같은 다양한 세트의 클라이언트에게 상당한 양의 정보를 제공하는데 특히 유용할 수 있다. 이러한 상황에서, 상당한 수의 클라이언트가 서버로서 동작하는 분산 컴퓨팅 시스템으로부터 정보를 요청할 수 있다. 다수의 장치들에 걸쳐 서버 기능을 구현함으로써, 서버는 전체적으로 증가된 리던던시 때문에 훨씬 덜 오동작하는 경향이 있다.

[0042] 구성 컴퓨팅 장치가 실행할 다음의 기능에 동의할 수 있는 하나의 메커니즘은 팩서스 알고리즘으로 알려져 있다. 팩서스 알고리즘에서, 이하 상술하는 바와 같이, 임의의 장치는 리더로 동작하여 분산 컴퓨팅 시스템 내의 다른 장치들에 제안 번호에 대한 제시를 전송할 수 있다. 다른 장치는 장치가 이미 찬성한 큰 제안 번호

를 갖는 제안의 지시 혹은 장치가 이전 제안에 찬성하지 않는 지시에 응답할 수 있다. 일단 리더는 다른 장치로부터 응답을 받으면, 제안할 기능을 선택하여 제안된 기능에 대한 찬성을 요청할 수 있다. 각 장치는 요청된 찬성 이전의 일정 시점에 더 높은 제안 번호에 대한 제시에 응답하지 않았다면 제안에 찬성할 것이다. 정족수의 장치가 이 제안에 찬성하면 그 제안은 받아들여지고, 리더는 모든 장치들에게 동의된 기능을 실행하라고 요청하는 메시지를 전송할 수 있다.

[0043] 그러나, 콘센서스(consensus) 알고리즘과 같이, 팩서스 알고리즘은 오동작을 허용하기 위하여 상당히 많은 수의 컴퓨팅 장치를 요구한다. 특히, F개의 오동작을 허용하기 위하여, 이 알고리즘은 분산 컴퓨팅 시스템이 적어도 2F+1개의 컴퓨팅 장치를 포함하도록 요구한다. 이러한 장치들 중, 단지 대다수만이 명령을 선택하여 시스템의 적절한 동작을 계속할 필요가 있다. 나머지 장치들은 명령을 선택하여 시스템을 구동하는 장치들 중 어느 하나가 오동작하지 않는다면 사용되지 않을 수 있다.

[0044] 그러나, 컴퓨팅 장치가 영원히 동작하지 않는다면, 팩서스 알고리즘이 F개의 추가적인 오동작을 허용할 수 있도록 하기 위하여, 이 장치를 2F+1개의 장치에서 제거하여 새로운 장치로 교체하는 것이 바람직하다.

[0045] **분산 컴퓨팅 환경**

[0046] 도면으로 돌아가서, 여기서 같은 참조 번호는 같은 구성요소를 나타내는데, 본 발명은 도 1에 도시된 예시적인 분산 컴퓨팅 시스템(10)과 같은 분산 컴퓨팅 시스템에 의해 구현될 수 있도록 도시된다. 설명의 편의를 위하여, 본 발명은 도 1에 도시된 바와 같이 상호 접속되는 컴퓨팅 장치(11~15)를 포함하는 시스템(10)과 같은 분산 컴퓨팅 시스템을 참조하여 설명될 것이다. 그 기술분야의 통상의 지식을 가진 자라면, 본 발명이 분산 컴퓨팅 환경에 적용가능하고, 설명할 목적으로 간단히 표현된 도 1의 예시적인 분산 컴퓨팅 시스템에 의해 제한되지 않음을 이해할 수 있을 것이다.

[0047] 또한, 도 1은 본 발명이 임의의 수의 클라이언트 컴퓨팅 장치를 갖는 환경에서 동작할 수 있지만, 하나의 클라이언트 컴퓨팅 장치(20)를 도시하고 있다. 클라이언트 컴퓨팅 장치(20)는 분산 컴퓨팅 시스템(10)으로 일반적인 통신 접속을 갖는 것으로 도시된다. 그 기술분야의 통상의 지식을 가진 자에게 알려져 있듯이, 이러한 통신 접속은 임의의 통신 매체 및 프로토콜을 사용하여, 클라이언트 컴퓨팅 장치(20)가 분산 컴퓨팅 시스템(10) 내의 하나 이상의 컴퓨팅 장치들과 통신하도록 할 수 있다.

[0048] 부가적으로는, 도 1은 분산 컴퓨팅 시스템(10)의 일부로서 도시되지는 않았지만 또한 시스템(10)으로의 일반적인 통신 접속을 유지하고 있는 컴퓨팅 장치(30, 31)를 도시하고 있다. 상술한 바와 같이, 통신 접속은 임의의 통신 매체 및 프로토콜을 사용하여 컴퓨팅 장치(30, 31)가 분산 컴퓨팅 시스템(10) 내의 하나 이상의 컴퓨팅 장치들과 통신하도록 할 수 있다. 이하 보다 상세히 설명되는 바와 같이, 컴퓨팅 장치(30, 31)는 시스템(10)의 일부가 아니지만 시스템(10)에 의해 실행되는 행위의 결과를 알 수 있다.

[0049] 비록 요구되는 것은 아니지만, 본 발명은 컴퓨팅 장치에 의해 실행되는 프로그램 모듈과 같은 컴퓨터 실행가능 명령어의 일반적인 내용으로 설명될 것이다. 일반적으로, 프로그램 모듈은 특정 태스크를 실행하거나 특정 앱스트랙트(abstract) 데이터 타입을 구현하는 루틴(routine), 프로그램, 오브젝트, 컴포넌트, 데이터 구조 등을 포함한다. 게다가, 그 기술분야의 통상의 지식을 가진 자는 본 발명이 휴대용 장치, 다중 프로세서 시스템, 마이크로프로세서 기반 또는 프로그램 가능한 가전제품, 네트워크 PC, 미니 컴퓨터, 메인프레임 컴퓨터 등을 포함하는 많은 다양한 컴퓨팅 장치로 실행될 수 있음을 이해할 것이다. 상술한 바와 같이, 본 발명은 또한 분산 컴퓨팅 시스템(10)과 같은 분산 컴퓨팅 환경에서도 실행될 수 있는데, 분산 컴퓨터 환경에서 태스크는 통신 네트워크를 통하여 연결되는 원격 프로세싱 장치에 의해 실행된다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 저장 장치에 위치될 수 있다.

[0050] 도 2로 돌아가서, 본 발명이 구현될 수 있는 예시적인 컴퓨팅 장치(100)가 도시된다. 컴퓨팅 장치(100)는 단지 적절한 컴퓨팅 장치의 일 예이고, 본 발명의 기능이나 사용 범위에 관한 어떠한 제한도 제기하려 하지 않는다. 예를 들어, 예시적인 컴퓨팅 장치(100)는 도 1에 도시된 컴퓨팅 장치(11~15, 20, 30~31)의 어느 하나를 정확히 나타내려는 것은 아니다. 예시적인 컴퓨팅 장치(100)는 다수의 컴퓨팅 장치에 부속된 바와 같이, 하나의 물리적 컴퓨팅 구조가 이하 설명될 동작을 실행하도록 하는 메모리 분할, 가상 머신, 다중 프로세서, 또는 유사한 프로그래밍 기술과 같은 하나 이상의 컴퓨팅 장치를 구현할 수 있다. 게다가, 컴퓨팅 장치(100)는 도 2에 도시된 주변장치들의 어느 하나 또는 임의의 조합에 관한 의존 또는 요구를 갖는 것으로 해석되어서는 안된다.

[0051] 본 발명은 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터 실행가능 명령어의 일반적인 내용으로 설명

될 수 있다. 일반적으로 프로그램 모듈은 특정 태스크를 실행하거나 특정 앱스트랙트 데이터 타입을 구현하는 루틴, 프로그램, 오브젝트, 컴포넌트, 데이터 구조 등을 포함한다. 분산 컴퓨팅 환경에서, 태스크는 통신 네트워크를 통해 연결되는 원격 프로세싱 장치에 의해 실행될 수 있다. 분산 컴퓨팅 환경에서 프로그램 모듈은 메모리 저장 장치를 포함하는 로컬 및 원격 컴퓨터 저장 매체에 위치될 수 있다.

[0052] 컴퓨팅 장치(100)의 컴포넌트는 시스템 메모리를 포함하는 다양한 시스템 컴포넌트들을 프로세싱 유닛(120)에 연결시키는 시스템 버스(121), 프로세싱 유닛(120) 및 시스템 메모리(130)를 포함할 수 있지만, 이것에 제한되지는 않는다. 시스템 버스(121)는 다양한 버스 아키텍처 중 어느 하나를 사용하는 메모리 버스나 메모리 컨트롤러, 주변 버스, 로컬 버스를 포함하는 몇몇 타입의 버스 구조 중 어느 하나일 수 있다. 예를 들어, 여기에 제한되는 것은 아니지만, 이러한 아키텍처는 ISA(Industry Standard Architecture) 버스, MCA(Micro Channel Architecture) 버스, EISA(Enhanced Industry Standard Architecture) 버스, VESA(Video Electronics Standard Associate) 로컬 버스, 및 메자닌 버스(Mezzanine bus)로도 알려진 PCI(Peripheral Component Interconnect) 버스를 포함한다. 게다가 프로세싱 유닛(120)은 하나 이상의 물리적인 프로세서를 포함할 수 있다.

[0053] 컴퓨팅 장치(100)는 일반적으로 다양한 컴퓨터 판독가능 매체를 포함한다. 컴퓨터 판독가능 매체는 컴퓨팅 장치(100)에 의해 액세스될 수 있는 임의의 사용가능한 메체일 수 있고 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체를 포함한다. 예를 들어, 여기에 제한되는 것은 아니지만, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈이나 다른 데이터와 같은 정보의 저장을 위한 기술이나 임의의 방법으로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 저장 매체는 컴퓨팅 장치(100)에 의해 액세스될 수 있고 희망하는 정보를 저장하는데 사용될 수 있는 RAM, ROM, EEPROM, 플래시 메모리나 다른 메모리 기술, CD-ROM, DVD나 다른 광 디스크 저장매체, 자기 카세트, 자기 테이프, 자기 디스크 저장매체나 다른 자기 저장 장치, 또는 임의의 다른 매체를 포함하지만 여기에 제한되지는 않는다. 통신 매체는 일반적으로 반송파나 다른 전송 메커니즘과 같은 변조된 데이터 신호로 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈이나 다른 데이터를 구현하고, 임의의 정보 전달 매체를 포함한다. "변조된 데이터 신호"라는 용어는 하나 이상의 특성 세트를 가지거나 신호 내의 정보를 인코딩하는 방식으로 변경되는 신호를 의미한다. 예를 들어, 여기에 제한되는 것은 아니지만, 통신 매체는 유선 네트워크나 직접 유선 접속과 같은 유선 매체 및 음향, RF, 적외선 및 다른 무선 매체와 같은 무선 매체을 포함한다. 위에 나열한 것의 어느 것의 조합도 컴퓨터 판독가능 매체의 범위 내에 포함되어야 한다.

[0054] 시스템 메모리(130)는 ROM(131), RAM(132)과 같은 휘발성 및/또는 비휘발성 메모리의 형태로 컴퓨터 저장 매체를 포함한다. 가동 개시 동안과 같은 컴퓨터(110) 내의 요소들 간의 정보를 전송하는 것을 돕는 기본 루틴을 포함하는 기본 입출력 시스템(BIOS; 133)은 일반적으로 ROM(131)에 저장된다. RAM(132)은 일반적으로 프로세싱 유닛(120)에 즉시 액세스 가능하고/거나 프로세싱 유닛에 의해 현재 동작되는 프로그램 모듈 및/또는 데이터를 포함한다. 예를 들어, 여기에 제한되는 것은 아니지만, 도 2는 운영 시스템(134), 어플리케이션 프로그램(135), 다른 프로그램 모듈(136), 및 프로그램 데이터(137)를 도시한다.

[0055] 컴퓨팅 장치(100)는 또한 다른 이동식/비이동식 휘발성/비휘발성 컴퓨터 저장 매체를 포함할 수 있다. 예를 들어, 도 2는 비이동식, 비휘발성 자기 매체로부터 판독하거나 기록하는 하드디스크 드라이브(141), 이동식 비휘발성 자기 디스크(152)로부터 판독하거나 기록하는 자기 디스크 드라이브(151), 및 CD-ROM이나 다른 광 매체와 같은 이동식 비휘발성 광 디스크(156)로부터 판독하거나 기록하는 광 디스크 드라이브(155)를 도시한다. 예시적인 운영 환경에서 사용될 수 있는 다른 이동식/비휘발성, 휘발성/비휘발성 컴퓨터 저장 매체는 자기 테이프 카세트, 플래시 메모리 카드, DVD, 디지털 비디오 테이프, 고체 상태 RAM, 고체 상태 ROM 등을 포함하지만, 여기에 제한되지는 않는다. 하드 디스크 드라이브(141)는 일반적으로 인터페이스(140)와 같은 비이동식 메모리 인터페이스를 통하여 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151)와 광 디스크 드라이브(155)는 일반적으로 인터페이스(150)와 같은 이동식 메모리 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0056] 상술되고 도 2에 도시된 드라이브 및 이와 관련된 컴퓨터 저장 매체는 컴퓨팅 장치(100)에 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 및 다른 데이터의 저장을 제공한다. 도 2에서, 예를 들어, 하드 디스크 드라이브(141)는 운영 시스템(144), 어플리케이션 프로그램(145), 다른 프로그램 모듈(146) 및 프로그램 데이터(147)를 저장하는 것으로 도시된다. 이러한 컴포넌트는 운영 시스템(134), 어플리케이션 프로그램(135), 다른 프로그램 모듈(136) 및 프로그램 데이터(137)와 동일하거나 혹은 다를 수 있음을 명심해라. 운영 시스

템(144), 어플리케이션 프로그램(145), 다른 프로그램 모듈(146), 및 프로그램 데이터(147)는 최소한 그들이 다른 복제품임을 도시하기 위하여 여기서는 다른 수로 주어진다. 사용자는 일반적으로 마우스, 트랙볼 또는 터치패드로 일컬어지는 포인팅 장치(161)와 키보드(162)와 같은 입력 장치를 통하여 컴퓨팅 장치(100)에 정보 및 명령어를 입력할 수 있다. 다른 입력 장치들(도시되지 않음)은 마이크, 조이스틱, 게임패드, 위성접시, 스캐너 등을 포함할 수 있다. 이러한 그리고 다른 입력 장치들은 종종 시스템 버스에 결합되는 사용자 입력 인터페이스(160)를 통해 프로세싱 유닛(120)에 접속되지만, 병렬 포트, 게임 포트 또는 USB(Universal Serial Bus)와 같은 버스 구조 및 다른 인터페이스에 의해 접속될 수 있다. 모니터(191) 또는 다른 타입의 디스플레이 장치가 또한 비디오 인터페이스(190)와 같은 인터페이스를 통해 시스템 버스(121)에 접속된다. 모니터 이외에도 컴퓨터는 또한 출력 주변 인터페이스(195)를 통해 접속될 수 있는 스피커(197) 및 프린터(196)과 같은 다른 주변 출력 장치를 포함할 수 있다.

[0057] 컴퓨팅 장치(100)는 도 1에 도시된 것과 같은 네트워크 환경에서 하나 이상의 원격 컴퓨터에 논리적으로 접속함으로써 동작할 수 있다. 도 2는 원격 컴퓨팅 장치(180)로의 일반적인 네트워크 접속(171)을 도시한다. 도 1에 도시된 네트워크 접속 및 일반적인 네트워크 접속(171)은 LAN, WAN, 무선 네트워크, 토큰-링 프로토콜, 이더넷 프로토콜에 국한되는 네트워크, 또는 월드와이드웹(WWW)이나 인터넷을 포함하는 무선 네트워크, 또는 다른 논리적 물리적 네트워크를 포함하는 다양한 여러 타입의 네트워크 및 네트워크 접속 중 어느 하나일 수 있다.

[0058] 컴퓨팅 장치(100)가 네트워크 환경에서 사용되는 경우, 컴퓨팅 장치(100)는 유선이나 무선 네트워크 인터페이스 카드, 모뎀 또는 유사 네트워크 장치일 수 있는 네트워크 인터페이스나 어댑터(170)를 통해 일반적인 네트워크 접속(171)에 접속된다. 네트워크 환경에서, 컴퓨팅 장치(100) 또는 그 부분들에 관하여 표현된 프로그램 모듈은 원격 메모리 저장 장치에 저장될 수 있다. 도시된 네트워크 접속은 예시적이고, 컴퓨터 사이의 통신 링크를 확립하는 다른 수단이 사용될 수 있음이 이해될 것이다.

[0059] 다음의 설명에서, 본 발명은 달리 지시되지 않는다면 하나 이상의 컴퓨팅 장치에 의해 실행되는 동작의 심볼 표시 및 행위를 참조하여 설명될 것이다. 이처럼 때때로 컴퓨터 실행가능한 것으로 간주되는 이러한 행위 및 동작이 구조화된 형태로 데이터를 나타내는 전기 신호의 컴퓨팅 장치의 프로세싱 유닛에 의한 조작을 포함하고 있음을 이해할 것이다. 이러한 조작은 데이터를 변환하거나 컴퓨팅 장치의 메모리 시스템의 위치에 유지하는데, 이는 그 기술분야의 통상의 지식을 가진 자에 의해 잘 이해될 수 있는 방식으로 컴퓨팅 장치의 동작을 재구성하거나 또는 그렇지 않으면 변경한다. 데이터가 유지되는 데이터 구조는 데이터의 포맷에 의해 규정되는 특정 성질을 갖는 메모리의 물리적 위치이다. 그러나, 본 발명이 이전 내용에서 설명되었지만, 여기서 설명된 다양한 동작 및 행위가 또한 하드웨어로 구현될 수 있음을 그 기술분야의 통상의 지식을 가진 자가 이해하듯이 여기에 제한됨을 의미하지 않는다.

[0060] **개요**

[0061] 본 발명에 따르면, 분산 컴퓨팅 시스템은 장치들이 시스템으로부터 제거되거나 추가되도록 하면서 오동작 허용 알고리즘을 구현할 수 있다. 이하에서 더 자세히 설명될 팩서스 알고리즘은 컴퓨팅 장치의 수의 2배 이상이 사용된다면 오동작의 수를 허용할 수 있는 분산 컴퓨팅 시스템을 구현하는 메커니즘을 제공할 수 있다. 수학적으로, 변수 "F"가 허용될 수 있는 오동작의 수를 나타낸다면, 적어도 2F+1개의 컴퓨팅 장치들의 분산 컴퓨팅 시스템이 상태 머신의 동일 레플리카를 행함으로써 팩서스 알고리즘을 구현할 수 있다. 이러한 2F+1개의 컴퓨팅 장치 중에서, 적어도 그들 다수의 집합, 즉 적어도 F+1개의 장치들의 집합은 정족수일 수 있고 오동작 허용 방식으로 제안들을 선택할 수 있다.

[0062] 어떠한 컴퓨팅 장치도 오동작하지 않는다면, 2F+1개의 컴퓨팅 장치들의 집합은 팩서스 알고리즘에 따라 상태 머신을 구현할 수 있다. 컴퓨팅 장치 중 하나 이상이 동작하지 않는다면, 나머지 F개의 컴퓨팅 장치 중 하나 이상은 시스템이 팩서스 알고리즘을 계속 사용하게 하여 오동작 허용 방식으로 동작하도록 하는데 사용될 수 있다. 그러나, 컴퓨팅 장치가 영원히 동작하지 않는다면, 2F+1개의 컴퓨팅 장치의 집합으로부터 제거하거나, 또는 2F+1개의 컴퓨팅 장치의 집합에 새로운 컴퓨팅 장치를 추가하는 것이 바람직할 수 있다.

[0063] 본 발명에 의해 고려되는 하나의 실시예에 따르면, 컴퓨팅 장치의 초기 세트가 오동작 허용 분산 컴퓨팅 시스템을 제공하는데 사용된다. 컴퓨팅 장치 중 하나 이상이 오동작하는 경우, 장치는 초기 세트로부터 제거될 수 있고, 새로운 컴퓨팅 장치가 세트에 추가될 수 있다. 결과적으로, 오동작 허용 시스템의 동작은 오동작 허용 분산 컴퓨팅 시스템이 동작하는 컴퓨팅 장치의 세트를 수정할 수 있다.

[0064] **상태 머신**

[0065] 도 1에 도시된 분산 시스템(10)과 같은 분산 환경에서, 비동기적으로 동작하는 장치들 사이의 조정은 어려운 일일 수 있다. 이러한 어려움을 설명하는 하나의 메커니즘은, 기능의 실행이 상태 머신을 하나의 상태에서 다른 상태로 이동시키는 상태 머신의 견지에서 분산 컴퓨팅 시스템을 모델링하는 것이다. 상태 머신은 응답/상태 쌍을 각각 명령/상태 쌍과 링크시키는 일 세트의 상태, 일 세트의 명령, 일 세트의 응답 및 기능을 참조하여 설명될 수 있다. 상태 머신의 클라이언트는 상태 머신이 기능을 실행하라고 요청하는 명령을 발행할 수 있다. 기능은 상태 머신의 상태를 변경하여 응답을 생성한다.

[0066] 분산 컴퓨팅 시스템을 포함하는 개개의 장치들은 각각 시스템의 상태 머신을 실행할 수 있다. 그러므로, 장치들은 초기 상태를 결정하여 그때부터 동일한 순서로 동일한 기능을 실행함으로써 조정될 수 있다. 장치는 단지 장치가 실행한 최종 기능을 결정하고, 다른 장치에 의해 실행된 기능의 순서화된 리스트로 그 기능을 위치시키며, 이후 장치가 아직 실행하지 않은 순서화된 리스트로부터 기능을 실행하도록 함으로써 동기화될 수 있다. 이러한 상태 머신 접근법은 1978년 7월 출판된 "The Communications of the ACM, Volume 21, Number 7"에서 레즐리 램포트(Leslie Lamport)에 의해 "Time, Clocks and the Ordering of Events in a Distributed System"라는 제목으로 최초 제안되었으며, 그 내용은 여기에 그대로 본 명세서의 일부로 포함되었다.

[0067] **팩서스 알고리즘**

[0068] 상태 머신 접근법을 사용함으로써, 도 1에 도시된 분산 컴퓨팅 시스템(10)의 구성 장치들(11~15)의 동기화가 실행될 기능 및 실행 순서에 동의함으로써 달성될 수 있다. 실행될 기능에 동의하기 위한 하나의 방법은 팩서스 알고리즘으로 알려져 있다. 팩서스 알고리즘은 시스템(10)이 사전 경고 없이도 장치가 동작을 멈출 수 있는 오동작에 직면하더라도 적절히 동작할 수 있도록 한다. 팩서스 알고리즘은 적어도 정족수의 장치들이 시스템이 전체적으로 기능을 실행하기 전에 기능에 동의하기를 요구한다. 팩서스 알고리즘에 따라, 정족수는 간단한 다수일 수 있으나, 시스템의 특정 요구사항에 따라 그보다 더 많은 장치들을 포함할 수 있다. 그러나, 임의의 정족수가 적어도 하나의 장치를 공통으로 할만큼 정족수가 충분히 클 수 있다고 정의하자. 팩서스 알고리즘의 자세한 설명은 1998년 5월 출판된 "ACM Transactions on Computer Systems, Volume 16, Number 2"의 페이지 133~169에서 레즐리 램포트에 의해 "The Part-Time Parliament"라는 제목의 논문과, 2001년 12월 출판된 "ACM SIGACT News, Volume 32, Number 4"의 페이지 18~25에서 레즐리 램포트에 의해 "Paxos Made Simple"라는 제목의 논문에서 발견될 수 있는데, 그 내용이 어느 부분을 배타하지 않고 개시하더라도, 그 내용은 여기에 그대로 본 명세서의 일부로 포함된다.

[0069] 명확해지도록 몇몇 새로운 또는 대안적인 용어 및 개념들이 이제 본 발명의 실시예에 사용된 바와 같은 팩서스 알고리즘을 설명하는데 제시된다. 도 3을 참조하면, 4개의 장치(301~304)가 도시된다. 장치(301~303) 각각은 공통 상태 머신의 레플리카(305)를 행하여 "레플리카 세트"를 형성한다. 장치(304)는 레플리카 세트에 있지 않는다. 각 레플리카(305)는 두개의 논리 모듈, 동의 모듈(AM; 306), 및 실행 모듈(EM; 307)을 구동한다. 이하 설명된 방식으로, AM은 상태 머신에서 각 슬롯(또는 "단계")에 대한 동작을 선택하기 위해 서로를 조정하는데, 여기서 슬롯 "n"은 각 상태 머신 레플리카가 실행해야 하는 n번째 동작에 대한 논리 컨테이너이다. 예를 들어, 슬롯(310)은 실행될 동작 "f"를 포함하고, 슬롯(311)은 동작 "t" 등을 포함한다. EM(307)은 배열된 AM(306)으로부터 동작 선택을 알고 있다.

[0070] 팩서스 알고리즘은 하나의 레플리카를 "리더"로 하고 단지 리더가 슬롯에 대하여 선택될 동작을 제안하도록 함으로써, 어느 두 AM이 결코 동일한 슬롯에 대하여 다른 동작을 선택하지 않도록 보증한다. 그러나, 리더가 오동작할 수도 있고, 그래서 새로운 리더를 선출하기 위한 프로토콜이 바람직하다. 선출 프로토콜을 통해, 새로운 리더는 이전 리더의 행동에 관하여 충분히 배우게 되고, 이는 대립하는 선택을 하지 않게 한다.

[0071] 리더의 "임기(term of office)"는 "뷰(view)"로 불리고, 각 뷰는 뷰 식별자(view identifier(ID))에 의해 고유하게 식별되는데, 뷰 식별자는 고전적인 팩서스 표현에 사용되는 "제안 번호"와 동등하다. 다시 말하여, 각각의 특정 뷰에 대하여, 하나의 리더가 그 뷰동안 상태 머신에 의해 실행될 모든 동작을 제안한다. 각각의 레플리카는 현재 뷰의 트랙을 유지하고, 그 뷰의 리더로부터만 동작 제안을 수용할 것이다. 뷰 ID는 뷰 카운터와 초기화 레플리카 식별자로 구성되고, 따라서, 다수의 레플리카가 동시에 뷰를 초기화한다면, 서로 다른 뷰를 개시할 것이다. 도 3의 예에서, 컴퓨팅 장치 C(303)는 디스플레이되는 뷰의 리더이고, 장치 B(302)는 초기화된 뷰이고, 뷰 ID는 번호 8이다.

[0072] 리더는 동작을 제안한 최종 슬롯을 기억한다. 클라이언트가 새로운 요청을 보내는 경우, PROPOSED 메시지를 모든 레플리카에 보냄으로써 이 요청이 최종 슬롯 다음의 슬롯에서의 동작이라고 제안한다. 복제가 현재 뷰의 리더로부터 PROPOSED 메시지를 받는 경우, 로컬 로그(log)에 메시지를 기록함으로써 준비한다. 이후, 리

더에게 PREPARED 메시지를 보낸다. 리더가 정족수(일반적으로 그 자체를 포함함)로부터 PREPARED 메시지를 받는 경우, 리더는 모든 레플리카에게 CHOSEN 메시지를 보냄으로써 동작을 선택한다. 다시 말하여, 리더는 어떠한 대립하는 동작도 선택될 수 없어 어떠한 EM도 동작을 실행할 수 있음을 약속한다.

[0073] 리더가 동작을 제안한 후, 이 동작이 실행되기 전에 네트워크 및 프로세싱 지연이 있다. 이 동안 리더는 파이프라인을 통하여 프로세스의 다양한 단계의 작업을 중첩시키기 위하여 더 많은 동작을 제안할 수 있다.

[0074] 본 발명의 일 실시예에서, 각각의 상태 머신은, 알고리즘이 레플리카의 로그가 무한정 성장하는 것을 막도록 안정한 저장 상태의 주기적인 복제를 한다. 이러한 복제는 "체크 포인트"로 불리고, 체크 포인트의 슬롯은 복제가 되기 전에 실행된 최종 동작의 슬롯으로 정의된다. 레플리카의 디스크 상의 최종 체크 포인트의 슬롯은 "로컬 안정점"으로 불린다.

[0075] 주기적인 메시지를 통하여, 리더는 레플리카의 로컬 안정점의 트랙을 유지하고, q번째로 높은 로컬 안정점인 집합적인 안정점을 알려주는데, 여기서 q는 정족수의 크기(예를 들어, F+1)이다. 이러한 값의 중요성은, 정족수의 레플리카가 점차 기능적이기 때문에, 점차 첫번째 c 동작을 반영하는 사용가능한 체크 포인트가 항상 있을 것이고, 여기서 c는 집합적인 안정점이다. 일부 레플리카는 집합적인 안정점 이하로 슬롯 내 모든 동작의 결과를 반영하는 상태를 제공할 수 있기 때문에, 이러한 슬롯에 대한 동작 선택은 기억될 필요가 없다. 따라서, 본 발명의 일 실시예에서, 레플리카는 "로그 절단(log truncation)"으로 불리는 프로세스에서 로그들이 무한정 성장하는 것을 막는 이러한 슬롯들에 대하여 PREPARE 엔트리를 버린다. 레플리카의 로그가 고정 또는 가변의 최대 로그 크기를 초과하지 못하도록 하기 위하여, 리더는 집합적인 안정점이 그 슬롯의 최대 로그 크기 내에 있을 때까지 동작을 제안하지 않는다.

[0076] 대립하는 동작들을 제안하지 않는 것을 보증하도록 새로운 리더가 선출되어 이전 뷰에 관하여 충분히 배우는 프로세스는 "뷰 변경"으로 불린다. 개요로서, 임의의 레플리카가 이러한 개요의 범위 밖의 수단에 의해 리더가 오동작했을 지도 모른다는 사실을 안다면, 레플리카는 뷰 변경을 초기화할 수 있다. 레플리카는 현 뷰 ID보다 더 큰 새로운 뷰 ID를 선택하여 모든 레플리카들에게 VIEW-CHANGE 메시지를 전달한다. VIEW-CHANGE 메시지를 받은 레플리카는 이미 더 높은 번호의 뷰가 없다면 이 새로운 뷰에 진입한다. 레플리카는 로그에서 준비 엔트리를 표현하는 VC-REPLY 메시지를 가지고 개시자(initiator)에 응답한다. 일단 뷰 변경 개시자가 정족수로부터 VC-REPLY 메시지를 갖는다면, 새로운 뷰에 대한 리더를 선택하여 이 리더에게 이 메시지를 제공한다. 새로운 리더는 대립하는 제안을 하지 않는 것을 보증하기 위하여, 이전 뷰에 관하여 충분히 알도록 이 메시지를 사용한다. 뷰 변경 프로세스는 앞서 인용된 램포트의 참조문헌에서 보다 충실히 설명된다. 도 3의 예에서, 장치 B(302)는 뷰 번호 8에 대한 뷰 변경을 개시한다.

[0077] 본 발명의 일 실시예에서, 주기적인 메시지를 통하여, 리더는 얼마나 많은 연속 동작이 선택되었는지를 각 레플리카에게 말한다. 따라서, AM은 EM에 의해 필요로 하는 다음 동작이 선택되었음을 들을 수 있다. 그러나, 리더가 그 동작에 대하여 PROPOSED 및 CHOSEN 메시지를 보낸 경우 오프라인이었기 때문에, AM은 이 슬롯에 대하여 어떤 동작이 선택되었는지 모를 수 있다. 이 경우에, AM은 동료 AM에게 그들 중 어느 하나가 최근에 선택된 동작의 캐시(cache)에 이 동작을 갖는지를 묻는다. AM은 로그로부터 동작을 포함하는 최근에 선택된 동작의 캐시를 유지하지만, 부가적으로 로그로부터 국지적으로 잘려진 동작을 포함할 수 있다. AM이 캐시에 동작을 갖는다면, AM은 완벽한 상태 이전을 실행하기 위하여 요청하는 AM에 대한 필요성을 방지하면서 요청하는 동료 AM에게 동작을 제공한다. 어떠한 동료도 이러한 동작을 제공할 수 없고, AM이 로그 절단 때문에 리더가 동작을 버린다는 사실을 안다면, 레플리카의 EM은 진행될 수 없다. 이러한 점에서, 더 최신의 레플리카로부터 상태를 가져온다.

[0078] **<u>레플리카 세트의 변경을 사용하여 수정된 팩서스 알고리즘</u>**

[0079] 위의 설명으로부터 알 수 있듯이, 컴퓨팅 장치 각각이 사용하고 있는 공통 상태 머신의 레플리카를 호스팅함으로써 알고리즘의 경우에 참여할 필요가 없을 수 있다. 다시 말하여, 어떤 장치들은 처음부터 도 3의 예에서 장치 D(304)와 같이 사용하고 있는 상태 머신 레플리카의 멤버가 아닐 수 있다. 예를 들어, 분산 시스템으로부터 제거되거나 또는 영원히 기능하지 않는 장치를 교체하기 위하여 이러한 장치를 레플리카 세트에 추가하는 것이 바람직하다. 레플리카 세트를 변경함으로써, 컴퓨팅 장치는 팩서스 알고리즘에 의한 고려로부터 제거되거나 추가될 수 있다. 따라서, 제거된 장치는 정족수가 취해진 2F+1개의 장치에서 카운팅되지 않는다. 본 발명의 실시예들은 레플리카 세트로부터 컴퓨팅 장치(더 정확히는 그들의 상태 머신)의 제거 및 추가를 촉진한다.

[0080] 레플리카 세트를 변경하는 접근법의 일반적인 설명은 다음과 같다. 상태 머신은 그 상태의 일부로서 서비스

를 담당하는 레플리카 세트를 포함한다. α 동작 이후에 레플리카 세트의 변경이 영향을 미치도록 다수의 펜딩(pending) 동작을 유지하기 위한 일정한 파이프라인 깊이 "α"가 있다. 다시 말하여, 레플리카 세트가 n번째 슬롯에서 변경되면, 새로운 레플리카 세트는 n+α개 이상의 슬롯에서만 동작들을 선택함으로써 α개의 펜딩 동작들을 허용한다. α가 파이프라인 동작의 깊이를 제한하기 때문에 큰 α가 바람직하다. 리더는 그 동작에 대하여 담당하는 레플리카 세트를 알 때까지는 동작 n+α를 제안할 수 없고, 동작 n을 실행할 때까지는 이것을 모를 수 있다.

[0081] 본 발명의 실시예들은 각각의 뷰가 특정 레플리카 세트가 되도록, 즉 고정된 레플리카 세트로만 간주되도록 고전적인 팩서스 모델을 수정한다. 이러한 방식으로, 레플리카 세트는 서로 간섭하지 않고, 뷰들은 고정된 레플리카 세트로 간주되기 때문에 더 관리하기 단순하다. 뷰들이 특정 레플리카 세트가 되는 것을 보증하기 위하여, 본 발명의 실시예는 뷰 ID 표현을 확장하여 이하 설명될 "시기(epoch)" 값을 포함하게 된다. 각 장치는 시기에 의해 라벨링된 개별적인 동의 모듈(AM)을 각 레플리카 세트에 대하여 구동하고, 각 레플리카 세트는 현재 활동하고 있는 멤버이다. 장치가 다수의 레플리카 세트에 있다면, 다수의 AM이 각각 그 자신의 시기로, 그러므로 그 자신의 뷰로 독립적으로 그리고 동시에 동작한다.

[0082] 확장된 뷰 ID의 일 예가 도 4a를 참조하여 도시된다. 레플리카 세트(430)는 컴퓨팅 장치 A(401), B(402), C(403)로부터 레플리카(405)를 포함하도록 도시된다. 도 3의 예에서와 같이, 컴퓨팅 장치 B(402)는 장치 C(403)를 리더로서 갖고 뷰 ID 번호 8을 갖는 뷰를 개시한다. 그러나, 레플리카 세트의 시기 값을 지시하는 추가적인 필드(field)가 그 뷰에 대한 각 장치의 AM(406)에 대하여 저장된다. 이 예에서, 레플리카 세트(430)에 대한 시기 값은 번호 3이다.

[0083] 레플리카 세트 특정 뷰를 갖는 레플리카 세트의 변경의 사용의 예는, 본 발명의 일 실시예에서 사용된 바와 같이, 이제 도 4a 및 4b를 참조하여 설명된다. 레플리카 세트는 연속하여 생성되고 R_i로 라벨링되는데, 여기서 인덱스 i는 레플리카 세트의 시기이고, R_0는 초기 레플리카 세트이다. 주어진 레플리카 세트에 대하여, 세트 내의 각 머신은 대응하는 AM을 갖는다. 도 4a에서, 레플리카 세트 내의 각 머신은 레플리카 세트 R_3(430) 내의 멤버쉽에 대응하는 AM을 갖는다. 이러한 AM 각각은 시기 3이다. 도 4a에서 AM(406) 각각은 뷰 <시기→3, 뷰→0, 개시자→nil>에서 시작한다. 시간이 진행됨에 따라, 뷰 변경이 발생하고, 이러한 뷰 ID가 진행된다. 예를 들어, 제1 뷰 변경이 레플리카 C에 의해 개시된다면, 새로운 뷰 ID는 <3, 1, C>일 것이다. 도 4a에서, 몇번의 뷰 변경이 발생하고, 뷰 ID는 <3, 8, B>이다. 도 4b에서 EM이 새로운 레플리카 세트 R_4(440)을 생성하는 동작을 실행하는 경우, 새로운 레플리카 세트의 멤버들에게 각각 적절한 시기에 AM을 개시하라고 알려준다. 어떤 머신이 R_3(430)과 R_4(440)에 있다면, 이는 두개의 독립적인 AM(406, 441)을 구동하게 할 것이다. 새로운 AM은 뷰<4, 0, nil>에서 시작할 것이다. 이러한 뷰 ID는 뷰 ID R_3에 독립적이고, 그 뷰를 독립적으로 선행할 것이다. 레플리카 세트 R_4(440)를 갖는 AM은 리더로서 레플리카 세트 R_4(440) 내의 어떤 AM을 선출한다. 이러한 리더는 레플리카 세트 R_3(430)의 리더로서 동일한 머신에 있을 필요는 없다. 도 4b에서, 뷰 변경은 AM(440)을 선행하여 뷰 ID<4, 1, A>로 진행한다.

[0084] 고정된 레플리카 세트로 간주함으로써, 본 발명의 실시예에서 뷰들을 관리하기 쉽다. 예를 들어, 뷰 변경이 레플리카 세트 내의 모든 AM인 잘 규정된 세트의 참여자를 포함한다. 이는 뷰 변경 개시자가 VIEW-CHANGE 메시지를 보내는 AM이고, 이들 대부분으로부터의 VC-REPLY 메시지는 진행할 다음 리더에 대하여 충분한 정족수의 레플리카를 구성한다. 또한, 이는 AM이 뷰 변경 개시를 담당하고, 리더가 되기에 적격일 때 분명하다. 파괴될 때까지 이를 담당하고 갖도록 생성된다. 부가적으로, 리더는 주기적인 메시지를 어떤 AM과 교환할 지 정확히 안다.

[0085] 도 5로 관심을 돌리면, 레플리카 세트 특정 뷰를 촉진하기 위하여, 동일한 머신 상의 상이한 레플리카의 AM 각각은 선택된 동작을 주도록 그 머신 상의 하나의 EM과 통신한다. EM은 수정되지 않은 팩서스 알고리즘에서와 같이 슬롯 순서로 선택된 동작을 실행한다. 어느 두개의 AM도 동일한 슬롯을 담당할 수 없기 때문에, 각각의 동작은 하나의 잘 규정된 AM으로부터 나온다. 도 5에서, 슬롯(500)은 레플리카 세트 R_4를 생성한 동작을 포함하는 슬롯이고, 슬롯(515)은 레플리카 세트 R_5를 생성한 동작을 포함하는 슬롯이다. 고정 파이프라인 깊이는 α=10이다. 그래서 레플리카 세트 R_4는 슬롯(510)(r_4=레플리가 세트 생성 슬롯(500+α))에서 슬롯(524)(r_5-1)까지 동작들을 선택하는 것을 담당한다. 리더가 담당하지 않는 동작을 제안하지 않기 때문에, 레플리카 세트는 담당하지 않는 동작을 선택하지는 않는다. 리더는 배열된 EM이 동작 n-α를 실행할 때까지 동

작 n을 제안하지 않고, 이러한 실행의 결과는 리더의 레플리카 세트가 슬롯 n을 담당하고 있음을 규정짓는다. 따라서, EM이 그 세트가 슬롯 r_m과 그 이후를 담당하도록 하는 레플리카 세트 R_m을 생성하는 동작을 실행한다면, 레플리카 세트 R_{m-1}과 그 이전에 대한 AM은 슬롯 r_m과 그 이상에 대하여는 제안을 하지 않을 것이다.

[0086] 레플리카 세트 특정 뷰를 보다 촉진하기 위해서, 본 발명의 실시예들은 리더가 새로운 레플리카 세트가 선택된 것을 안다면 제안 파이프라인을 비우도록 한다. 이러한 리더들은 "레임덕(lame duck)" 리더로 알려져 있다. 제안 파이프라인을 비우는 것은 다음과 같은 시나리오에서 유용하다. 레플리카 세트 R_{m-1}의 리더가 r_m-1을 담당하는 최종 동작을 제안하고, 클라이언트를 새로운 레플리카 세트로 다시 인도하기를 시작한다. 그러면 부서져서 이후 뷰 변경은 모든 예전 리더의 최근 제안을 잃게 되고, 새로운 리더가 제안한 최종 슬롯이 r_m-1 미만이 된다. 모든 활동하는 클라이언트가 새로운 레플리카 세트로 다시 인도되어졌기 때문에, 예전 레플리카 세트의 새로운 주요 부분으로의 요청을 하지 않을 것이고, 어떠한 동작도 동작 r_m-1에 대하여 제안되지 않을 것이다. 이는 슬롯 r_m 그 이상의 동작만이 선택되기 때문에 상태 머신이 더 이상 진행되지 않고, 동작 r_m-1이 실행될 때까지 실행될 수 없다. 이러한 시나리오를 피하기 위하여, 본 발명의 실시예들은 다음을 고려한다. 리더가 다른 레플리카 세트가 동작 r_m에서 동작이 r_m-1 미만이라고 제안한 최종 슬롯을 맡을 것임을 아는 경우, 제안된 최종 슬롯과 r_m에 사이의 각 슬롯에 대하여 널 동작을 제안한다. 이는 제안된 최종 슬롯이 적어도 머신이 실행한 동작의 수(적어도 r_m-α)이기 때문에 적어도 α-1개의 널 동작을 요구한다. 파이프라인을 비우는 이러한 프로세스는 상태 머신의 실행이 점차 슬롯 r_m에 도달하는 것을 보증한다.

[0087] 레플리카 세트 특정 뷰를 보다 촉진하기 위하여, 본 발명의 실시예들은 예전의 불필요한 AM을 버린다. 일단 R_m의 집합적인 안정점이 적어도 r_m-1이 된다면, R_m은 "확립"된다. 다시 말하여, 정족수의 멤버들이 개시되었고, 집합적인 안정점의 정의에 의해 점차 그들 중 하나가 체크 포인트에 적어도 슬롯 r_m-1을 제공할 수 있을 것이다. 이는 이전 레플리카 세트(즉, R_0, R_1, ..., R_{m-1})가 선택을 담당하는 r_m, 즉 모든 동작 미만의 동작에 관한 어떠한 정보에 대한 필요성을 방지한다. 따라서, R_m이 확립되는 경우, 모든 이전의 레플리카 세트는 "죽은", 즉 더 이상 필요하지 않게 된다. 죽은 AM을 갖고 있다고 아는 임의의 머신은 죽은 AM을 그들을 로그를 포함하여 파괴할 수 있다. 머신이 단지 구동하고 있는 AM을 파괴하면, 또한 EM도 파괴할 수도 있다. 따라서, 이는 더 이상 서비스에서 활동적인 참여자가 아니다.

[0088] 레플리카 세트 특정 뷰를 보다 촉진하기 위하여, 본 발명의 실시예들에서, 머신은 단지 파괴된 AM에 대한 비휘발성 상태의 작은 양, 즉 파괴된 로컬 AM("MaxDefunct") 사이의 가장 높은 시기와, 이것($R_{MaxDefunct+1}$)을 계승하는 시기에 대응하는 레플리카 세트를 보유한다. 이는 파고된 AM과의 통신을 개시하려고 하는 원격 AM이나 임의의 클라이언트에게 이것들을 제공한다. 파괴된 AM에 관하여 들은 클라이언트는 계승하는 레플리카 세트와 통신을 개시하고, 죽은 동료의 소식을 들은 AM은 그 자체를 파괴하기를 안다.

[0089] 본 발명의 실시예들은 계승자 레플리카 세트가 확립되는 것을 보증하기 위하여, 레플리카 세트의 AM에 대한 메커니즘을 제공하고, AM이 이러한 확립에 대하여 점차 알게 되어, 그들은 파괴될 수 있다. 레프리카 세트 R_{m-1}의 AM이 EM이 동작 rm-1, 즉 이러한 AM의 레플리카 세트가 담당하는 최종 동작을 실행했음을 아는 경우, 새로운 레플리카 세트 내의 각 AM에게 IS-SUCCESSOR-UP-TO-DATE 메시지를 주기적으로 보내는 쓰레드(thread)를 생성한다. 이 메시지는 수신자의 로컬 안정점이 적어도 r_m-1인지를 묻고, 그렇지 않다면, 그 슬롯이나 그 다음에 체크 포인트를 제공한다. 이미 긍정적으로 응답한 AM에 이러한 메시지를 보내지는 않는다. 점차, 정족수의 새로운 레플리카 세트로부터 긍적적인 응답을 받게 되어 R_m이 확립된 것을 알게 된다. 이러한 점에서, 죽어서 그 자체를 파괴하는 것을 안다.

[0090] R_{m-1} 내의 모든 AM이 IS-SUCCESSOR-UP-TO-DATE 메시지를 보내기 때문에, 점차 그들 중 하나는 R_m을 확립한다. IS-SUCCESSOR-UP-TO-DATE 메시지를 보내는 것의 담당은 단지 R_{m-1}의 리더에게만 할당될 수 있다. 또는, IS-SUCCESSOR-UP-TO-DATE 메시지를 보내는 것의 담당은 Rm-1의 리더에게만 보다는 오히려 모든 AM에게 할당된다. 이는 예전 레플리카 세트의 리더가 뒤에 있다면 새로운 레플리카가 곧 초기 체크 포인트를 획득하도록 한다. 또한, 리더에 전체 부담을 주기보다는 예전 레플리카 세트의 AM들 중에 체크 포인트를 제공하는 로드를 분산시킨다. 모든 AM에의 담당의 할당에 대한 추가적인 이유는 이하 설명될 인증 체인을 사용한 최적화와 관계된

다.

[0091] EM은 다음의 시나리오에서 IS-SUCCESSOR-UP-TO-DATE 메시지 내에 내재된 체크 포인트를 제공하는 것을 사용한다. EM이 배열된 AM으로부터 다음 동작을 획득할 필요가 있다고 하면, 어떠한 배열된 AM도 이러한 동작을 담당하지는 않는다. 예를 들어, 도 6에서, 장치 C(403) 상의 EM이 가령 R_4에 의해 관리되는 슬롯으로부터의 동작을 필요로 한다면, 머신이 R_4의 일부가 아니기 때문에 문의할 AM을 가지지 못한다. 이러한 경우에, EM은 R_5에 대응하는 AM의 경우에 필요한 번호 이하의 최하위 동작 번호를 담당하는 AM을 문의한다. 이 AM은 EM이 필요로 하는 동작을 제공할 수 없으며, 사실 r_5-1 이전의 어떠한 동작도 제공할 수 없음을 깨닫는다. 그래서, 레플리카 세트가 담당하지 않는 모든 동작의 결과를 포함하는 체크 포인트, 즉 r_5-1 또는 이후에서의 체크 포인트를 어디에서 획득하는지 EM에게 말한다. 동일 시기의 동료 AM 중 하나가 국지적으로 이러한 체크 포인트를 갖는다면, 동료 머신 상의 EM과 접촉하도록 EM에게 말한다. 그렇지 않다면, 선행하는 레플리카 세트 내의 일부 레플리카로부터 최근의 IS-SUCCESSOR-UP-TO-DATE 메시지를 받아, EM이 이 레플리카로부터 체크 포인트를 획득하도록 한다.

[0092] 단지 IS-SUCCESSOR-UP-TO-DATE 메시지가 새로운 AM을 생성할 수 있다면, 새로운 AM은 EM이 동작 r_m-1을 실행하기까지 시작할 수 없다. AM이 더 빨리 시작하도록 하는 또 다른 최적화로서, 본 발명의 실시예들은 EM이 세트를 생성하는 동작 r_m-α를 실행할 때 JOIN 메시지를 각각의 새로운 레플리카에 보내도록 한다. JOIN 메시지를 받는 머신은 새로운 레플리카 세트에 대한 AM을 개시할 것이다. JOIN 메시지는 단지 최적화이기 때문에 승인되거나 재전송되지 않는다.

[0093] 본 발명의 실시예들은 나중에 확립된 레플리카 세트에 관한 "계승자 정보"를 생성하여 사용한다. EM이 레플리카 세트 R_m을 생성하는 동작을 실행할 때, 레플리카 세트 R_{m-1}에 대한 로컬 AM에 의해 사용될 계승자 정보를 생성한다. 이 정보는 계승자가 담당하는 초기 동작 번호인 r_m과, 계승하는 레플리카의 본질(identity)을 포함한다. AM이 r_m을 필요로 하고, 그래서 AM이 리더라면 AM은 r_m 이상의 동작을 제안하지 않을 것이다. AM은 계승하는 레플리카의 본질을 필요로 하고, 그래서 나중에 IS-SUCCESSOR-UP-TO-DATE 메시지를 보낼 수 있다.

[0094] 레플리카 세트 R_m을 생성하는 동작 r_m-α를 실행하기 보다, EM은 그 동작을 건너뛰도록 하는 상태 전이를 받을 수 있다. 따라서, 상태 인출을 완성한 후에, EM은 배열된 AM의 어느 하나에 대한 계승자 정보를 이제 아는지를 체크하여 만약 그러하다면 그들에게 계승자 정보를 배분한다. 이를 달성하기 위하여, 상태는 심지어 예전 레플리카 세트에 대하여 r_m과 R_m을 포함한다. 상태 머신은 이 정보를 무한정 보유할 수 있다. 또는, 상태 머신은 점차 이 정보를 다음의 방식으로 버릴 수도 있다. 리더의 레플리카 세트의 집합적인 안정점이 그 슬롯의 최대 로그 크기 내에 있을 때까지 리더가 슬롯에 대한 동작을 제안하지 않을 것을 상기하자. 따라서, 상태 머신이 r_m-1로부터 떨어진 최대 로그 크기인 슬롯에서 동작을 실행하는 경우, 상태 머신은 레플리카 세트 R_m의 집합적인 안정점이 적어도 r_m-1임을 추정할 수 있고, 따라서, m-1 이하의 시기를 갖는 레플리카 세트가 죽는다. 그래서, 상태 머신은 죽은 것으로 아는 가장 최근의 레플리카 세트의 번호만을 기억하면서 m-1 이하의 시기를 갖는 레플리카 세트에 대한 정보를 버릴 수 있다. 따라서, 상태 머신은 계승자 정보를 임의의 AM에 제공할 수 없을 수 있지만, 이러한 AM에 대하여 EM은 그 자신을 파괴하라고 그들에게 말할 수 있다.

[0095] 본 발명의 실시예들은 레플리카 세트 내의 장치들이 정당함을 나타내기 위하여 "인증 체인"을 사용한다. 새로운 레플리카 세트의 일부임을 나타내는 메시지를 수신하는 머신은 보낸 사람이 거짓말을 하고 있지 않음을 확인할 필요가 있을 수 있다. 정상적으로 신뢰는 오동작으로 멈추는 콘센서스(consensus)의 문제는 아니다. 그러나, 레플리카 세트가 변경될 때, 이러한 메시지를 교환하는 머신은 동일한 오동작으로 멈추는 콘센서스 그룹에 있지 않다. 초기 레플리카 세트에 있는 머신은 서비스 인증, 즉 초기 레플리카 세트로 규정하는 믿을만한 인증기관(CA)에 의해 서명된 인증을 통하여 그곳에 속해 있음을 안다. CA가 레플리카 세트 변경 동안에 오프라인일 수 있기 때문에, 본 발명의 실시예는 새로운 레플리카 세트를 증명하기 위한 레플리카를 사용한다.

[0096] 인증 체인은 인증의 시퀀스이다. 각각의 인증은 임의의 레플리카 세트를 증명한다. 체인에서 최초 인증은 초기 레플리카 세트 R_0를 증명하는 서비스 인증이다. 다른 각각의 인증은 선행 레플리카 세트의 멤버에 의해 서명된다. 도 7을 참조하면, 인증 체인(700)의 일 예가 도시된다. 인증기관(예를 들어, 믿을만한 제3자)은 레플리카 세트 R_0가 장치 A, B, C를 구성하고 있다는 인증(710)을 발행한다. 레플리카 세트 R_0의 멤버인 장치

A는 레플리카 세트 R_1이 인증(711)에서 장치 A, B, D를 구성하고 있다고 인증한다. 장치 D는 레플리카 세트 R_2가 장치 A, D, E를 구성하고 있다는 인증(712)를 발행한다.

[0097] 머신이 특정 레플리카 세트에 있음을 보여주는 인증을 받으면, 머신은 이것이 그렇다고 확신한다. 새로운 AM은 이것을 원래의 체인으로서 저장한다. R_0의 AM에 대하여, 서비스 인증 자체는 원래 체인이다. AM이 계승자 레플리카 세트의 멤버에게 AM을 생성하라고 말할 필요가 있을 경우, 새로운 레플리카 세트를 표현하는 인증을 서명하고, 이것을 원래 체인을 부착하여 새로운 AM의 원래 체인으로서 이것을 보낸다.

[0098] 본 발명의 실시예들은 원래 체인을 짧게 유지하는 것을 돕는 두가지 선택적인 최적화를 사용한다. 첫째, 하나의 레플리카 세트가 어떤 인증에 의해 증명되고, 이 레플리카 세트가 그 체인의 후속 인증의 서명자를 포함하면, 둘 사이의 체인에서 모든 매개하는 인증들이 제거될 수 있다. 둘째, AM이 다른 전임자로부터 더 짧은 체인을 받는다면 원래 체인을 대체할 수 있다. 예를 들어, 도 7을 다시 참조하면, 장치 E는 장치 D로부터 체인(700)을 받으면, 이후 장치 A는 최종 인증(712)을 서명하는 것을 제외하고는 체인(700)과 동일한 체인을 보내며, 장치 E는 더 짧은 체인(720)을 생성하기 위하여 불필요한 중간 인증(711)을 제거할 수 있으며, 이 더 짧은 체인(720)으로 원래 체인(700)을 교체할 수 있다. 전임자로부터 더 짧은 인증 체인을 받을 수 있는 기회는 상술한 바와 같이 단지 리더가 아닌 레플리카 세트 내의 모든 AM이 IS-SUCCESSOR-UP-TO-DATE 메시지를 보내기 때문에 가능하다.

[0099] 본 발명의 실시예들은 상태 머신이 레플리카 세트를 변경할 수 있는 인터페이스를 제공한다. 동작을 실행하고 있는 상태 머신은 새로운 레플리카 세트에 포함되기 위한 머신의 어레이를 규정하는 change_replicas() 기능을 불러낸다.

[0100] 본 발명의 실시예들은 레플리카 세트가 언제 변경되는지를 결정하는 방법을 사용한다. 어떤 실시예에서는, 관리자는 클라이언트가 레플리카 세트 변경 요청을 하도록 함으로써 레플리카 세트를 명확히 관리한다. 상태 머신은 관리자의 증거가 부족함을 요청하는 것을 무시한다. 관리자는, 예를 들어, 머신을 영원히 은퇴시키거나, 또는 추가적인 오동작 허용을 위한 새로운 머신을 배치하기 위하여 명백한 요청을 사용할 수 있다.

[0101] 어떤 실시예에서는, 레플리카 세트 변경은 관리자에게 요청없이 끊임없는 머신 오동작에 대처하기 위하여 자동적으로 실행된다. 각각의 EM은 규정된 간격으로, 예를 들어, 매 5분마다 자신이 살아있음을 보고하는 서비스에 대한 요청을 특정 클라이언트에게 보낸다. 상태 머신은 각 EM이 자신이 살아있음을 보고한 마지막 시간을 구성하는 여분의 상태를 계속 알고 있는다. 이 상태는 각 레플리카 상의 로컬 클럭에 의해서가 아니라 동작에 의해 완전히 결정되고, 그래서 각 동작은 리더에 의해 할당된 타임스탬프(timestamp)를 포함하고, 각 동작은 임의의 "시간"에서 실행된다. 주어진 시간에서 동작을 실행하는 것에 대한 추가적인 상세한 설명은 2002년 12월, 메사추세추, 보스톤의 아디야(Adya) 등의 "FARSITE: Federated, available and reliable storage for an incompletely trusted environment" 5번째 OSDI, 페이지 1~14에서 설명되고, 그 내용은 여기서 어느 부분에 대한 배제없이 개시하는 모든 요소에 대하여 전적으로 본 명세서의 일부로 포함된다. 주어진 간격, 예를 들어, 매 시간마다, 상태 머신은 레플리카 중에서 가장 오래된 최종적으로 살아있는 시간이 1시간 이상인지를 체크한다. 만약 그렇다고 한다면, 대응하는 장치가 영원히 동작하지 않는 것으로 가정하고, 레플리카 세트에서 그것을 교체하기 위하여 change_replicas() 기능을 호출한다.

[0102] 상태 머신은 또한 결정적으로 교체될 새로운 장치를 선택한다. 주기적으로 레플리카가 될 수 있는 장치들은 특별한 클라이언트 요청을 서비스로 보낸다. 이러한 요청은 보낸 사람이 오동작으로 멈춘 서비스에 참여하도록 권한을 부여하는 인가된 기관으로부터의 인증을 포함한다. 상태는 머신들이 이러한 요청을 언제 마지막으로 했는지 잠재적인 레플리카로부터 타임스탬프로의 맵핑을 포함한다. 새로운 머신을 선택하기 위하여, 상태 머신은 타임스탬프가 충분히 최근인 머신들 사이의 의사 랜덤이지만 결정적인 선택을 한다.

[0103] 본 발명의 원칙이 적용될 수 있는 가능한 많은 실시예들의 견지에서, 도면을 참조하여 여기서 설명된 실시예들은 단지 예시적이고 본 발명의 범위를 제한하는 것이 아님을 명심해야 한다. 예를 들어, 그 기술분야의 통상의 지식을 가진 자는 소프트웨어로 도시된 예시적인 실시예들의 일부 요소들이 하드웨어로 구현될 수 있고, 그 반대도 가능하며, 또는 예시적인 실시예들이 본 발명의 정신으로부터 벗어남이 없이 그 배치 및 상세한 설명에서 수정될 수 있음을 인식할 것이다. 그러므로, 여기서 설명된 바와 같이, 본 발명은 다음의 청구의 범위 및 그 균등물의 범위 내에 들 수 있는 모든 실시예들을 고려한다.

발명의 효과

[0104] 본 발명은 오동작 허용 분산 컴퓨팅 시스템에 참여하고 있는 일 세트의 컴퓨팅 장치가 팩서스 알고리즘을 사용하여 상태 머신의 동일한 레플리카를 구현하여 제안들을 선택하도록 함으로써, 상기 세트로부터 컴퓨팅 장치를 제거하거나 추가하는 것이 가능하게 되어, 레플리카 세트를 효율적으로 변경할 수 있게 된다.

도면의 간단한 설명

[0001] 도 1은 본 발명의 일 실시예가 구현될 수 있는 예시적인 분산 컴퓨팅 시스템을 일반적으로 나타내는 블록도.

[0002] 도 2는 본 발명의 일 실시예가 구현될 수 있는 예시적인 컴퓨팅 장치를 일반적으로 나타내는 블록도.

[0003] 도 3은 본 발명의 일 실시예가 구현될 수 있는 예시적인 세트의 복제된 상태 머신(state machine)을 일반적으로 나타내는 블록도.

[0004] 도 4a는 본 발명의 일 실시예가 구현될 수 있는 예시적인 세트의 복제된 상태 머신을 일반적으로 나타내는 블록도.

[0005] 도 4b는 본 발명의 일 실시예가 구현될 수 있는 예시적인 세트의 복제된 상태 머신을 일반적으로 나타내는 블록도.

[0006] 도 5는 본 발명의 일 실시예에 따른 예시적인 세트의 복제된 상태 머신의 동작 슬롯을 일반적으로 나타내는 블록도.

[0007] 도 6은 본 발명의 일 실시예가 구현될 수 있는 예시적인 세트의 복제된 상태 머신을 일반적으로 나타내는 블록도.

[0008] 도 7은 본 발명의 일 실시예에 따른 예시적인 체인의 레플리카 세트 인증을 일반적으로 나타내는 블록도.

[0009] <도면의 주요 부분에 대한 부호의 설명>

[0010] 10: 분산 컴퓨팅 시스템

[0011] 11, 12, 13, 14, 15, 30, 31: 컴퓨팅 장치

[0012] 20: 클라이언트 컴퓨팅 장치

[0013] 100: 컴퓨팅 장치

[0014] 120: 프로세싱 유닛

[0015] 130: 시스템 메모리

[0016] 140: 비이동식 비휘발성 메모리 인터페이스

[0017] 150: 이동식 비휘발성 메모리 인터페이스

[0018] 160: 사용자 입력 인터페이스

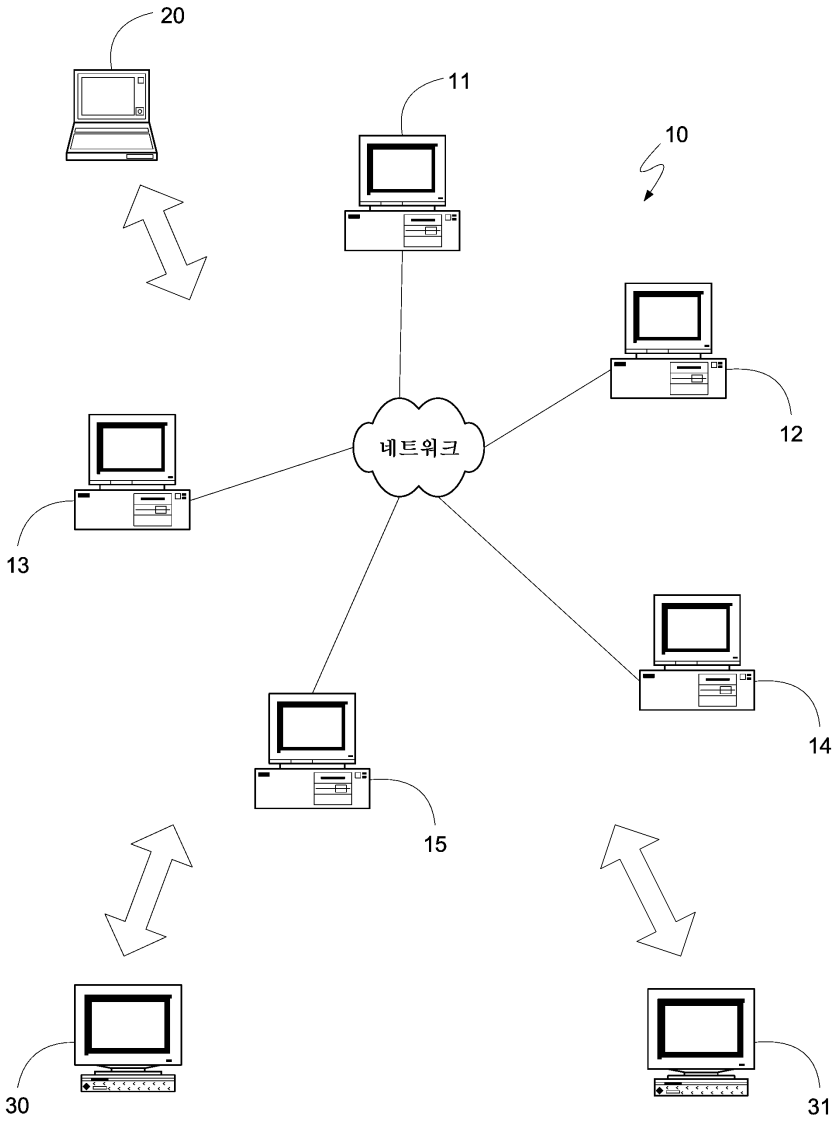
[0019] 170: 네트워크 인터페이스

[0020] 180: 원격 컴퓨팅 장치

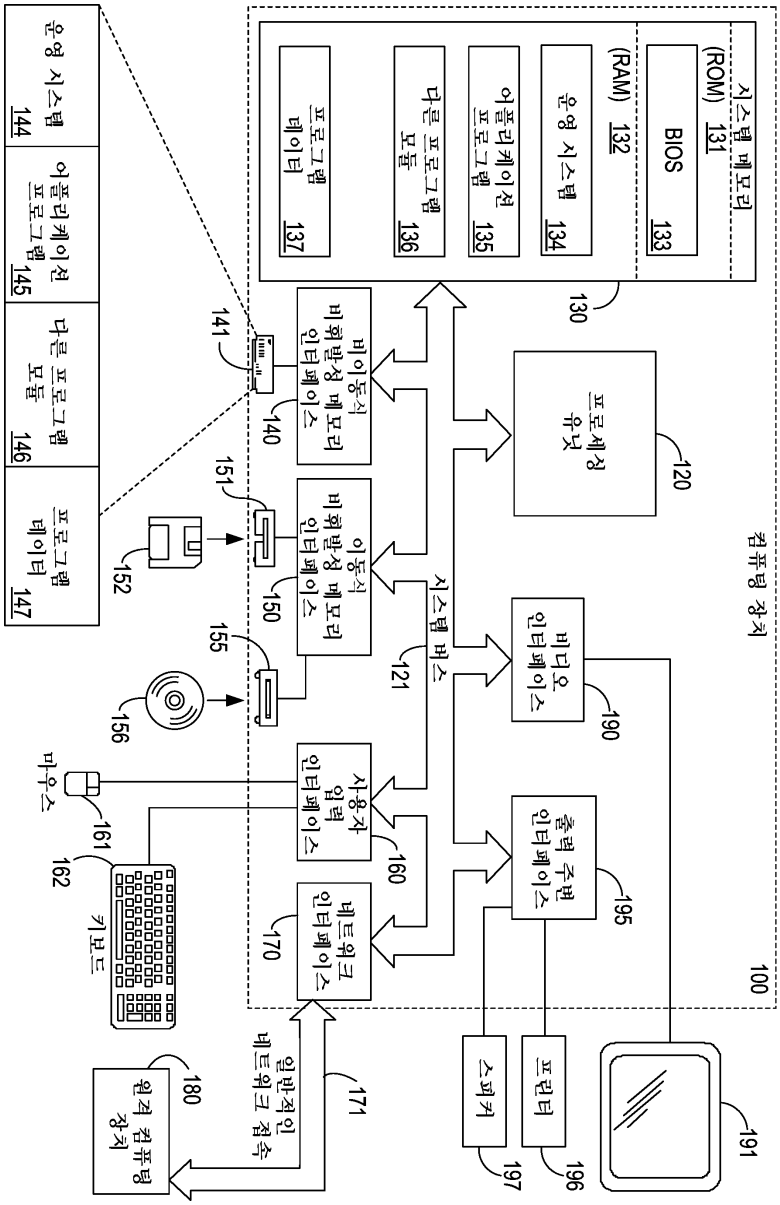
[0021] 190: 비디오 인터페이스

도면

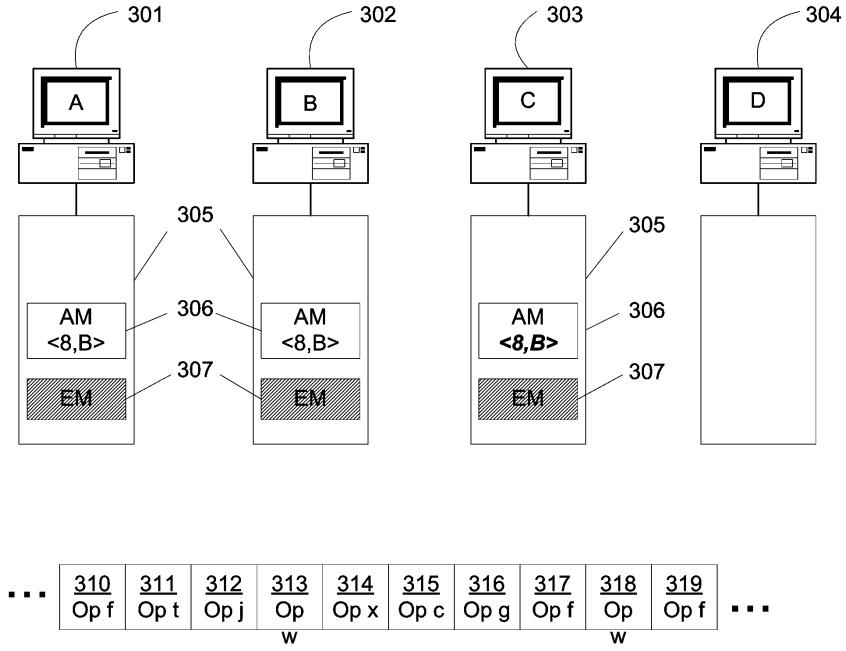
도면1



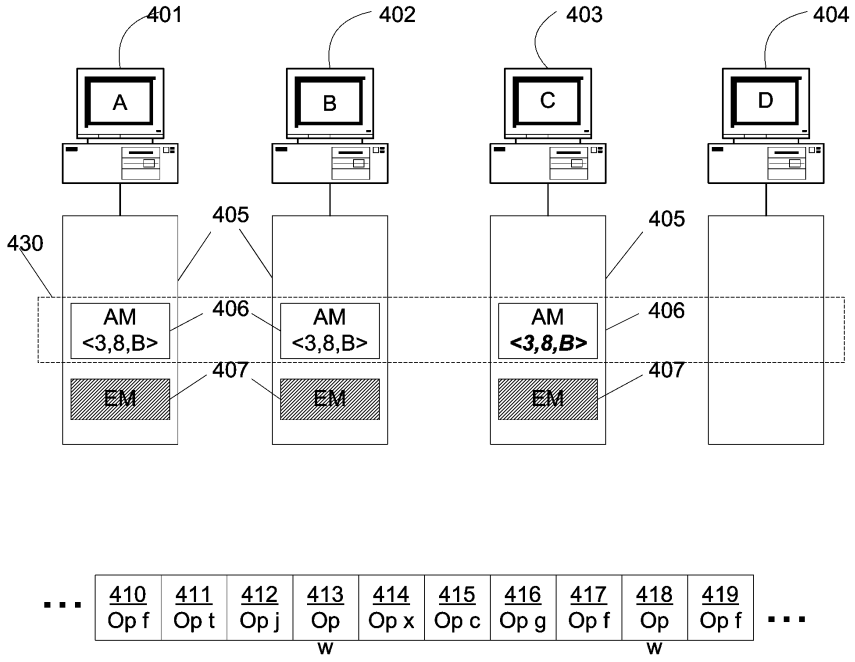
도면2


컴퓨팅 장치
100
시스템 메모리
(ROM) 131
BIOS 133
(RAM) 132
운영 시스템 134
어플리케이션 프로그램 135
다른 프로그램 모듈 136
프로그램 데이터 137
130
프로세싱 유닛
120
시스템 버스
121
비휘발성 메모리 인터페이스
140
141
이동식 비휘발성 메모리 인터페이스
150
151
152
155
156
비디오 인터페이스
190
출력 주변 인터페이스
195
사용자 입력 인터페이스
160
161
162
마우스
키보드
네트워크 인터페이스
170
171
일반적인 네트워크 접속
원격 컴퓨팅 장치
180
모니터
191
프린터
196
스피커
197
운영 시스템
144
어플리케이션 프로그램
145
다른 프로그램 모듈
146
프로그램 데이터
147

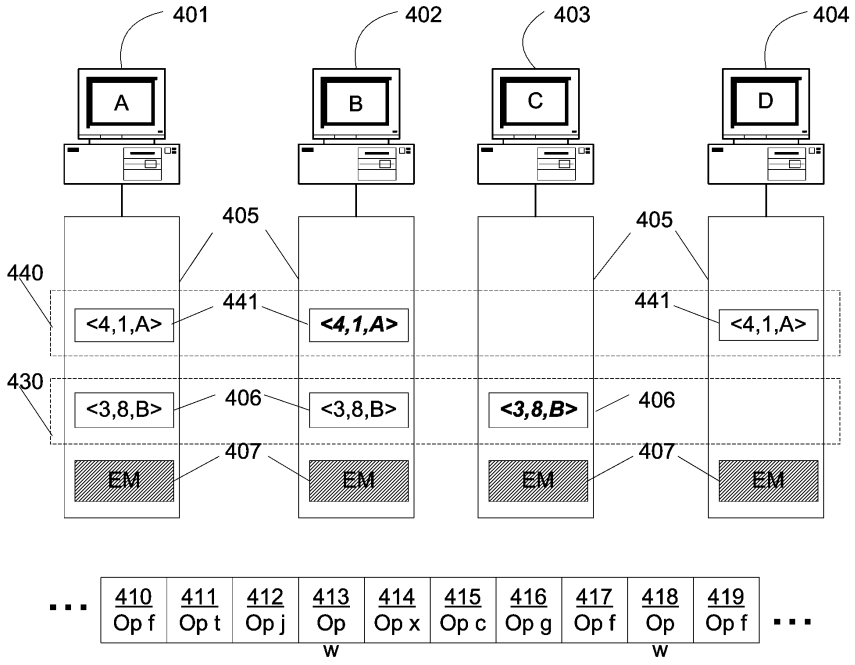
도면3


301
302
303
304
A
B
C
D
305
306
307
AM
<8,B>
EM
310 Op f
311 Op t
312 Op j
313 Op w
314 Op x
315 Op c
316 Op g
317 Op f
318 Op w
319 Op f

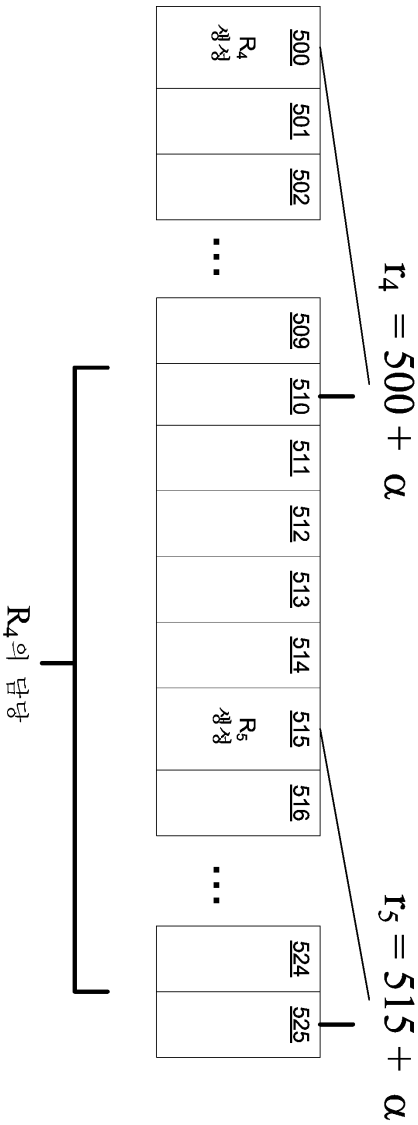
도면4a


401
402
403
404
A
B
C
D
405
406
407
430
AM
<3,8,B>
EM
410 Op f
411 Op t
412 Op j
413 Op w
414 Op x
415 Op c
416 Op g
417 Op f
418 Op w
419 Op f

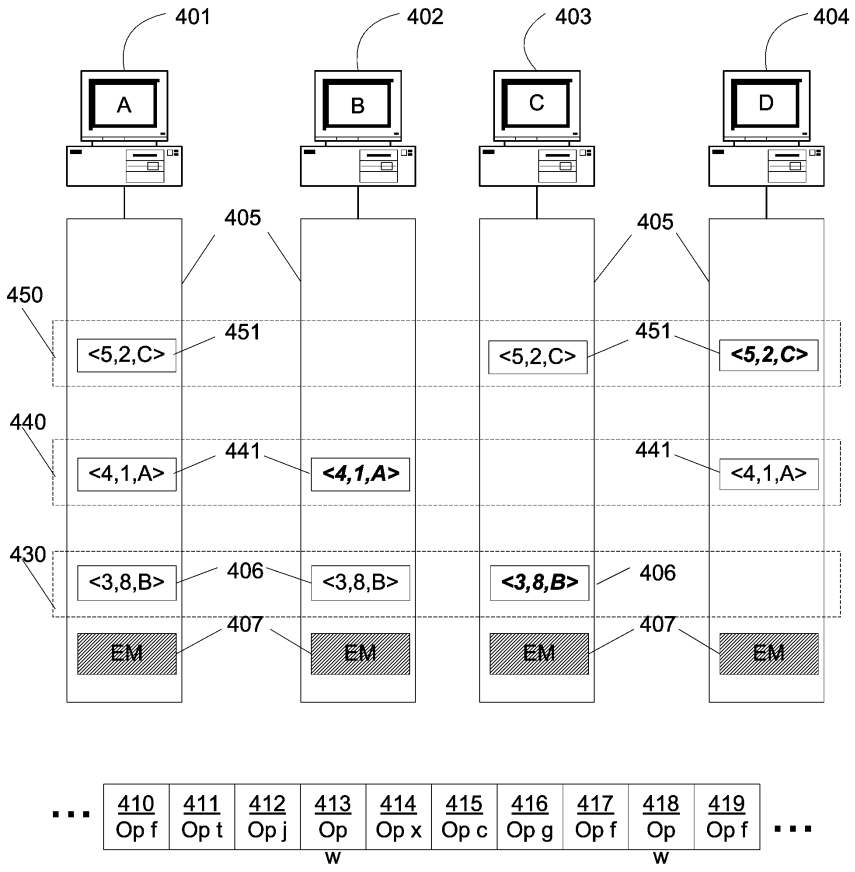
도면4b


401
402
403
404
A
B
C
D
405
405
440
441
441
<4,1,A>
<4,1,A>
<4,1,A>
430
406
406
406
<3,8,B>
<3,8,B>
<3,8,B>
407
407
407
EM
EM
EM
EM
410 Op f
411 Op t
412 Op j
413 Op w
414 Op x
415 Op c
416 Op g
417 Op f
418 Op w
419 Op f

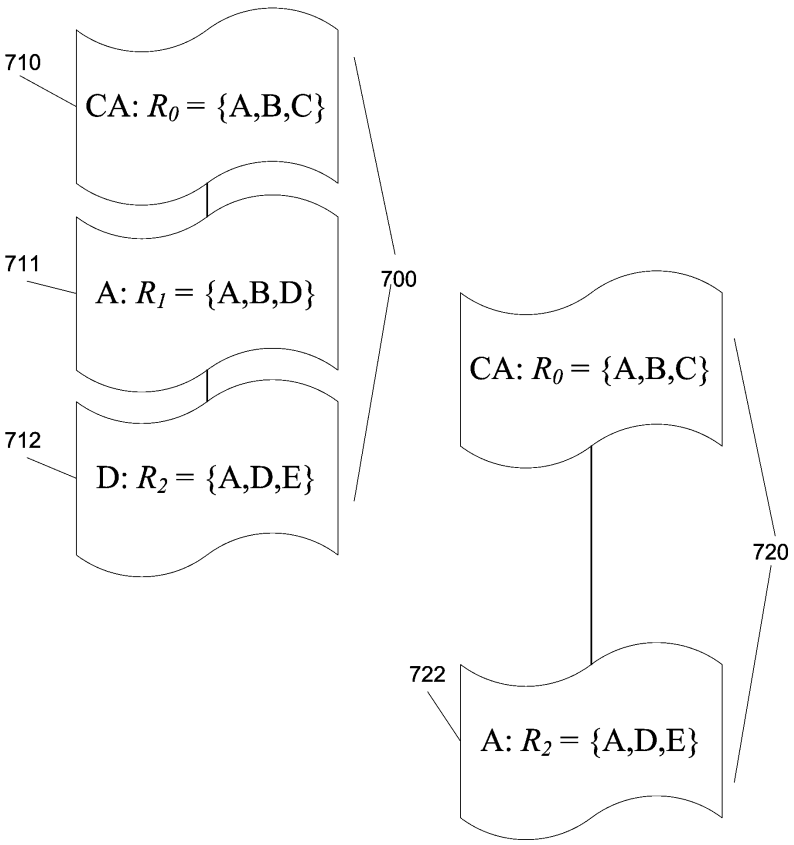
도면5


500
501
502
509
510
511
512
513
514
515
516
524
525
R4 생성
R5 생성
r4 = 500 + α
r5 = 515 + α
R4의 담당

도면6


401
402
403
404
A
B
C
D
405
405
450
451
451
<5,2,C>
<5,2,C>
<5,2,C>
440
441
441
<4,1,A>
<4,1,A>
<4,1,A>
430
406
406
<3,8,B>
<3,8,B>
<3,8,B>
407
407
EM
EM
EM
EM
410 Op f
411 Op t
412 Op j
413 Op w
414 Op x
415 Op c
416 Op g
417 Op f
418 Op w
419 Op f

도면7


710
CA: R_0 = {A,B,C}
711
A: R_1 = {A,B,D}
712
D: R_2 = {A,D,E}
700
CA: R_0 = {A,B,C}
720
722
A: R_2 = {A,D,E}