



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0113372
(43) 공개일자 2022년08월12일

(51) 국제특허분류(Int. Cl.)
G06F 12/02 (2018.01) G06F 11/30 (2006.01)
G06F 12/1018 (2016.01) G06F 9/32 (2018.01)
(52) CPC특허분류
G06F 12/0261 (2013.01)
G06F 11/3037 (2013.01)
(21) 출원번호 10-2022-7018294
(22) 출원일자(국제) 2020년11월13일
심사청구일자 없음
(85) 번역문제출일자 2022년05월30일
(86) 국제출원번호 PCT/US2020/060320
(87) 국제공개번호 WO 2021/126429
국제공개일자 2021년06월24일
(30) 우선권주장
16/724, 143 2019년12월20일 미국(US)

(71) 출원인
마이크로소프트 테크놀로지 라이선싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원
마이크로소프트 웨이
(72) 발명자
스티븐스 마오니 장
미국 워싱턴주 98052-6399 레드몬드 원 마이크로
소프트 웨이 마이크로소프트 테크놀로지
라이선싱, 엘엘씨
뒤수드 패트릭 헨리
미국 워싱턴주 98052-6399 레드몬드 원 마이크로
소프트 웨이 마이크로소프트 테크놀로지
라이선싱, 엘엘씨
(74) 대리인
김태홍, 김진희

전체 청구항 수 : 총 15 항

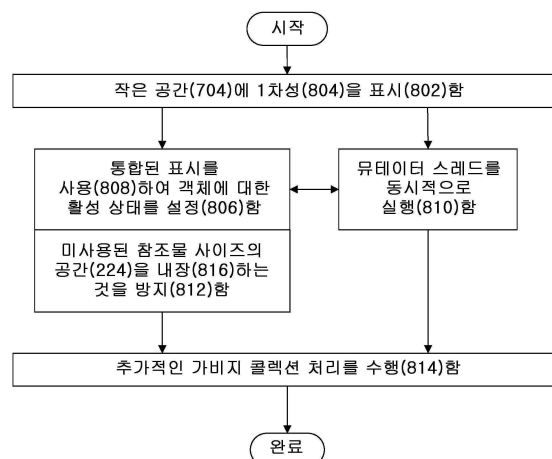
(54) 발명의 명칭 통합된 참조물 및 2차 객체 표시

(57) 요약

2차 객체를 효율적으로 관리하게 되면 가비지 콜렉션 동시성이 증가하고, 객체 저장 요구 사항은 감소된다. 2차 객체의 활성 상태 표시는 참조된 객체의 활성 상태 표시와 통합된다. 1차 객체가 아닌 객체에 참조물 사이즈의 2차 객체 식별자 필드 할당은 방지되며; 전용 비트는 객체 관계 테이블과 함께 1차 객체를 지정한다. 1차 객체는, 1차 객체가 활성 상태인 경우, 1차 객체의 참조된 객체는 아니면서, 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체를 갖는 객체이다. 활성 1차 객체의 임의의 참조된 객체도 여전히 활성 상태인 것으로 간주될 것이다. 참조된 객체를 표시하기 위한 코드 경로가 공유되어 보다 효율적인 2차 객체 표시를 가능하게 할 수 있다. 1차-2차 객체 관계는 종속 핸들에서 표현될 수 있고, 해시 테이블 또는 다른 데이터 구조에서 지정될 수 있다.

대표도 - 도8

메모리 관리 프로세스의 예



(52) CPC특허분류

G06F 12/0246 (2013.01)

G06F 12/0269 (2013.01)

G06F 12/1018 (2013.01)

G06F 9/321 (2013.01)

G06F 2212/1016 (2013.01)

G06F 2212/1044 (2013.01)

G06F 2212/1048 (2013.01)

명세서

청구범위

청구항 1

2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템으로서,

메모리와;

상기 메모리와 동작 가능하게 통신하는 프로세서

를 포함하고,

상기 프로세서는 (a) 다수의 객체에 내장된 전용 공간에, 상기 다수의 객체 중 각각의 객체가 1차 객체인지 여부를 표시하는 단계 - 각각의 전용 공간은 객체 참조물보다 작고, 1차 객체는, 상기 1차 객체가 활성 상태인 경우, 상기 1차 객체의 참조된 객체는 아니면서 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체를 갖는 객체임 -, 및 (b) 가비지 콜렉션 표시 동안 적어도 (i) 각 활성 객체 X의 모든 참조된 객체 및 (ii) 각 활성 객체 X의 모든 2차 객체 모두를 활성 상태로 표시함으로써 2차 객체를 참조된 객체로 취급하는 단계를 포함하는 메모리 관리 단계를 수행하도록 구성되고;

이로써 상기 시스템은 가비지 콜렉션 동안 참조된 객체의 표시와 2차 객체의 표시를 통합하여, 1차 객체가 아닌 객체에게 2차 객체에 대한 참조물을 보유하기 위한 참조물 사이즈의 공간을 내장할 것을 요구하지 않고, 추가 사용을 위한 메모리를 복구하는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 2

제1항에 있어서,

항목을 갖는 해시 테이블을 더 포함하고, 해시 테이블 항목은 1차 객체에 대한 1차 객체 참조물 및 2차 객체에 대한 2차 객체 참조물을 포함하여, 상기 2차 객체가 상기 1차 객체의 2차 객체임을 표시하는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 3

제1항에 있어서,

객체에 내장되어 상기 객체가 1차 객체인지 여부를 표시하는 각 전용 공간은 단일 비트로 구성되는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 4

제1항에 있어서,

상기 객체는 적어도 2개의 2차 객체를 갖는 1차 객체 Y를 포함하는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 5

제1항에 있어서,

상기 객체는 1차 객체가 아닌 객체 Z를 포함하고, 상기 객체 Z는 2차 객체를 갖지 않으며, 참조물도 내장하지 않는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 6

제1항에 있어서,

상기 객체는 1차 객체이기도 한 2차 객체 Q를 포함하는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 7

제1항에 있어서,

상기 시스템에서 적어도 부분적으로 실행된 프로그램의 객체의 최대 10퍼센트는 1차 객체가 되는 방식; 또는

상기 시스템에서 적어도 부분적으로 실행된 프로그램은 적어도 10,000개의 객체를 갖되, 각 객체 X는 해당 객체 X가 1차 객체인지 여부를 표시하기 위한 4개의 비트 이하의 각 내장된 전용 공간을 가지며, 그리고 각 객체 X에는 임의의 다른 객체를 해당 객체 X의 2차 객체로 식별하기 위한 4개의 비트보다 큰 어떠한 전용 공간도 내장되지 않는 방식

중 적어도 하나를 또한 특징으로 하는 것인, 2차 객체를 사용하여 메모리를 관리하도록 구성된 시스템.

청구항 8

2차 객체를 사용하여 메모리를 관리하기 위한 방법으로서,

다수의 객체에 내장된 전용 공간에, 상기 다수의 객체 중 각각의 객체가 1차 객체인지 여부를 자동으로 표시하는 단계 - 각각의 전용 공간은 객체 참조물보다 작고, 1차 객체는, 상기 1차 객체가 활성 상태인 경우, 상기 1차 객체의 참조된 객체는 아니면서 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체를 갖는 객체임 -;

가비지 콜렉션 표시 동안 적어도 (i) 각 활성 객체 X의 모든 참조된 객체 및 (ii) 각 활성 객체 X의 모든 2차 객체 모두를 활성 상태로 표시함으로써 2차 객체를 참조된 객체로 자동으로 취급하는 단계; 및

1차 객체가 아닌 객체에, 2차 객체에 대한 참조물을 보유하기 위한 임의의 참조물 사이즈의 공간을 내장하는 것을 자동으로 방지하는 단계

를 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 9

제8항에 있어서,

가비지 콜렉션 표시 동안,

적어도 활성 상태에 대해 개별 객체가 확인되는 지점까지 2차 객체 표시 및 참조된 객체 표시에 의해 공유되는 코드 경로를 따르는 단계; 또는

적어도 활성 상태에 대해 개별 객체를 확인하는 것을 지원하기 위해 객체 참조물이 열거되는 지점까지 2차 객체 표시 및 참조된 객체 표시에 의해 공유되는 코드 경로를 따르는 단계

중 적어도 하나를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 10

제8항에 있어서,

가비지 콜렉션 표시 동안,

비-1차 객체를 포함하는 핸들 테이블을 통해 루핑하여 2차 객체 핸들을 수집하는 것에 대한 의존성을 방지하는 단계; 또는

2차 객체 핸들을 수집하기 위해 활성 프로모션 카운터 값에 대한 의존성을 방지하는 단계

중 적어도 하나를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 11

제8항에 있어서,

객체를 생성하기 위한 요청의 결과로서 상기 객체에 대한 핸들을 리턴하기 전에, 상기 객체에 내장된 비트를 설정하여, 상기 객체가 1차 객체임을 표시하는 단계

를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 12

제8항에 있어서,

해시 테이블을 사용하여 객체가 1차 객체의 2차 객체임을 지정하는 단계를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 13

제8항에 있어서,

데이터 구조를 사용하여 객체 X 및 객체 Y가 각각 1차 객체 Z의 2차 객체임을 지정하는 단계; 및

객체 X의 각 참조된 객체를 표시하고 객체 X의 각 2차 객체를 표시함으로써, 2차 객체 X를 그 자체의 1차 객체로 취급하는 단계

중 적어도 하나를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 14

제8항에 있어서,

특정 1차 객체-2차 객체 관계의 적어도 하나의 객체를 대체하기 위한 커맨드를 수신하는 단계, 및 상기 커맨드를 수행하는 동안 1차 객체-2차 객체 관계를 지정하는 데이터 구조를 업데이트하는 단계

를 더 포함하는, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

청구항 15

제8항에 있어서,

상기 방법은 가비지 콜렉션 동안 참조된 객체를 표시하기 위해 참조된 객체 표시 코드 경로를 따르는 단계, 및 가비지 콜렉션 동안 2차 객체를 표시하기 위해 2차 객체 표시 코드 경로를 따르는 단계를 포함하고,

상기 참조된 객체 표시 코드 경로의 적어도 50%는, 라이브러리 코드를 포함하거나 라이브러리 코드를 제외하거나 또는 둘 모두인지 여부, 그리고 실행된 모든 경로 또는 모든 이용 가능한 경로 또는 최단 경로만을 따르는지 여부에 관계없이, 다음의 중첩 메트릭: 즉, 코드 라인의 수, 컴파일러에 의해 생성된 명령어의 수, 호출된 루틴의 수, 또는 스택 프레임의 수 중 적어도 하나에 따라, 상기 2차 객체 표시 코드 경로와 중첩하는 것인, 2차 객체를 사용하여 메모리를 관리하기 위한 방법.

발명의 설명

기술 분야

배경 기술

- [0001] 컴퓨터 프로그래밍에서, 가비지 콜렉션(garbage collection)(GC)은 자동 메모리 관리를 제공하는 데 도움을 준다. 해당 "가비지"는 프로그램의 사용을 위해 할당된 메모리 공간으로서, 해당 프로그램에 의해서는 더 이상 필요하지 않지만, 다른 프로그램의 사용을 위해서는 아직 이용 불가능한 메모리 공간이다. 가비지 콜렉터(garbage collector)는 이러한 메모리 구역을 식별하여 후속 재활당을 위해 재확보(reclamation)될 수 있도록 시도한다.
- [0002] 가비지 콜렉션은 프로그래머가 더 이상 필요로 하지 않는 데이터 객체를 수동으로 해제할 필요가 없도록 하여 프로그래밍이 보다 쉬워지게 할 수 있다. 가비지 콜렉션은 또한 일부 런타임 오류를 방지하는 데 도움을 줄 수 있으므로, 데이터의 무결성과 가용성을 개선하여 사이버 보안을 강화시킬 수 있다. 많은 프로그래밍 언어 또는 런타임은 가비지 콜렉션을 필요로 하거나, 또는 가비지 콜렉션을 옵션으로서 허용하고 있다. 일부 소프트웨어 개발 환경에서는 주어진 프로그램에서의 가비지 콜렉션과 수동 메모리 관리가 모두 허용되고 있다.
- [0003] 가비지 콜렉터가 자주 실행될 수 있거나 또는 예측할 수 없는 시간에 실행될 수 있거나 또는 두 경우 모두에 실

행될 수 있고, 사용자 프로그램 실행을 상당히 느리게 할 수 있기 때문에, 가비지 콜렉터의 효율적인 구현은 우선 순위가 높을 수 있다. 실제로, 대부분의 가비지 콜렉터는 때때로 사용자 프로그램이 가비지 콜렉터와 동시에 실행되는 것을 방지하는 “전체 멈춤(stop the world)” 접근 방식을 이용하고 있다. 가비지 콜렉터가 덜 자주 실행되도록 구성하거나 가비지 콜렉터에게 단축된 시간을 제공하여 자신의 작업을 완료하게 하는 것을 상상할 수 있다. 그러나, 가비지 콜렉션 효율을 증가시키지 않으면서도 가비지 콜렉션에 소요되는 시간의 양을 단순히 감소시키게 되면, 재확보되는 메모리의 양이 상당히 감소될 수 있고, 이는, 예를 들어, 휘발성 및 비휘발성 메모리 디바이스 사이에서 데이터를 교환하는 데 소요되는 시간을 증가시키거나 가용 메모리를 더 빠르게 소진시킴으로써 사용자 프로그램 성능을 저해할 수 있다.

발명의 내용

[0004] 본 문서에 기술된 일부 실시예는, 특히 각각의 1차 객체가 활성 상태인 경우에, 1차 객체의 참조된 객체는 아니더라도 활성 상태인(따라서 가비지가 아닌) 것으로 간주되는 2차 객체와 관련하여 가비지 콜렉션 동안 개선된 효율성을 제공한다. 일부 실시예는 참조된 객체의 활성 상태 표시(aliveness marking)를 2차 객체의 활성 상태 표시와 통합한다. 특히, 일부는 각 종류의 표시마다의 코드 경로를 중첩함으로써, 코드 공간과 복잡성을 줄이고 실행 시간을 줄인다. 일부 실시예는, 실제로 1차 객체가 아니어서 2차 객체가 없는 객체에서 참조물 사이즈의 2차 객체 식별자 필드의 할당을 방지함으로써 전체 메모리 사용 효율성을 개선한다. 다른 효율성도 또한 제공될 수 있다.

[0005] 2차 객체를 사용하여 메모리를 관리하도록 구성된 일부 실시예는 다수의 객체에 내장된 전용 공간을 가정하거나 제공한다. 각 전용 공간은 공간의 내장 객체가 1차 객체인지 여부를 표시한다. 각 전용 공간은 객체 참조물(object reference)보다 작고; 일부 실시예에서 전용 공간은 객체 내의 단일 비트이다. 1차 객체는, 1차 객체가 활성 상태인 경우, 1차 객체의 참조된 객체는 아니면서, 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체를 갖는 객체이다. 1차 객체의 임의의 참조된 객체도 또한 활성 상태인 것으로 간주될 것이다. 가비지 콜렉션 표시 동안, 일부 실시예는 적어도 (i) 각 활성 객체 X의 모든 참조된 객체 및 (ii) 각 활성 객체 X의 모든 2차 객체 모두를 활성 상태로 표시함으로써 2차 객체를 참조된 객체로 취급한다. 객체 X의 참조된 객체 Y는 X의 선언된 내장 필드에서 X가 가리키는 객체이다.

[0006] 본원의 교시 내용과 관련된 다른 기술적 활동 및 특징은 또한 본 기술 분야의 기술자에게는 명백할 것이다. 주어진 예는 단지 예시에 불과할 뿐이다. 이 개요는 청구된 발명의 요지의 핵심 특징 또는 필수 특징을 식별하기 위한 것도 아니고 청구된 발명의 요지의 범위를 제한하도록 사용되는 것도 아니다. 오히려, 본 개요는 일부의 기술적 개념을 단순화된 형태로 소개하기 위해 제공되며, 그 개념은 아래의 상세한 설명에서 추가로 기술된다. 신규한 사항은 적절히 이해되는 바와 같이 청구항으로 정의되며, 본 개요가 청구항과 충돌한다면, 청구항이 우선되어야 한다.

도면의 간단한 설명

[0007] 첨부된 도면을 참조하여 보다 구체적인 설명이 제공될 것이다. 이들 도면은 선택된 양태만 도시하므로 적용 범위나 영역을 완전히 결정하지는 않는다.

도 1은 컴퓨터 시스템을 전반적으로 나타내고 또한 구성된 저장 매체를 전반적으로 나타내는 블록 다이어그램이다.

도 2는 향상된 메모리 관리 기능을 갖는 일부 컴퓨터 시스템의 일부 양태를 나타내는 블록 다이어그램이다.

도 3은 가비지 콜렉션 동안 서로 다른 시점에서의 메모리 상태의 양태를 나타내는 다이어그램이다.

도 4는 가비지 콜렉션 스레드(garbage collection thread) 및 뮤테이터 스레드(mutator thread)의 동시적 실행(concurrent execution)을 나타내는 다이어그램이다.

도 5는 1차 객체, 2차 객체, 및 객체 참조물을 나타내는 다이어그램이다.

도 6은 객체들 간의 1차-2차 관계를 나타내는 해시 테이블을 나타내는 다이어그램이다.

도 7은 객체 인스턴스 데이터 구조의 비트를 나타내는 다이어그램이다.

도 8은 일부 메모리 관리 방법의 단계를 나타내는 플로우차트이다.

도 9는 일부 메모리 관리 방법의 단계를 추가로 나타내는 플로우차트이다.

발명을 실시하기 위한 구체적인 내용

[0008] 개관

[0009] 신규한 사항은 그 원천을 넘어 확장되지만, 신규한 사항의 원천을 이해하는 것은 신규한 사항을 보다 완전히 이해하는 데 도움을 줄 수 있다. 본 사례에서, 본원에 기술된 일부 교시 내용은 2차 객체와 관련된 동시적 또는 비동시적 가비지 콜렉션의 성능을 개선하도록 연구 중이었던 마이크로소프트의 혁신자가 직면한 기술적 과제에 의해 동기가 부여되었다. 혁신자는 2차 객체를 이용하여 동시적 가비지 콜렉션 메커니즘의 성능 조사를 수행했다. 이러한 성능 조사는 언젠가 나중에 공용 API를 통해 2차 객체 생성을 직접 사용할 수 있도록 하기 위해 수행된 예방 조치였다.

[0010] 일부 가비지 콜렉션 환경에서, 객체는, 예컨대, 종속 GC 핸들 타입을 사용하여, 확장된 참조물을 포함할 수 있다. 종속 핸들(dependent handle)은 1차 객체와 2차 객체를 가지며, 2차 객체는 1차 객체가 활성 상태인 경우에만 활성 상태이다(예외로 하는 조건은 다른 개체에 의해 활성 상태로 유지되는 경우이다). 이는, 일부 구현에 있어서, 표시(marking)에 대해 복잡성을 부가하게 되는데, 그 이유는 1차 객체에 대한 정보가 2차 객체에 관해서는 알지 못하기 때문일 것이다.

[0011] 이러한 2차 객체는 내부적으로만 사용되거나 또는 다른 방식으로 공용 API 액세스로부터 대체로 또는 완전히 은닉되어 사용되었기 때문에 상대적으로 드물었다. 종속 핸들 생성부를 공용 API로 만든 결과 2차 객체의 사용이 증가한다면, 본원에 제공된 교시 내용을 통해, 친숙한 가비지 콜렉션 접근 방식이 제공하지 않는 방식으로, 2차 객체의 가비지 콜렉션을 효율적으로 확장할 수 있을 것이다.

[0012] 특히, 하나의 기술적 과제는 이러한 2차 객체 핸들의 콜렉션을 효율적이고 확장 가능하게 만드는 방법에 있다. 다른 기술적 과제는 1차-2차 객체 관계를 효율적이고 확장 가능한 방식으로 표현하는 방법에 있다. 한 가지 접근 방식은, 1차 객체 핸들(primary object handle), 2차 객체 핸들(secondary object handle), 및 비-1차 객체 핸들(non-primary object handle)을 포함하는 핸들 테이블 내에 2차 객체 핸들을 계속해서 나열하는 것일 것이다. 비-1차 객체는 2차 객체 핸들(따라서 해당 2차 객체의 기본 객체가 됨)을 포함할 공간을 갖는 객체이지만, 실제로 1차 객체가 아닌 객체이다(즉, 실제로는 어떠한 2차 객체도 갖지 않는다).

[0013] 그러나 이러한 핸들 테이블 접근 방식은 비효율적인 방식인데, 그 이유는 이러한 방식은 가비지 콜렉션 동안 많은 비-1차 객체를 통해 루핑을 수행하여 2차 객체 핸들을 수집하는 것을 포함하기 때문이다. 이는 또한 비-1차 객체가 추가되면 2차 객체 핸들을 수집하기 위해 수행되는 작업이 증가하기 때문에 충분히 확장되지 않는다. 모든 객체에 2차 객체 핸들을 위한 공간을 저장하는 것은 또한 비효율적인데, 그 이유는 많은 경우 해당 공간이 이용되지 않기 때문이다.

[0014] 본원에 교시된 일부 실시예는, 예컨대, 해시 테이블, 밸런싱된 트리, 또는 다른 데이터 구조에서, 1차 객체와 그의 모든 2차 객체 간의 매핑을 구축한다. 비-1차 객체에 대한 항목은 매핑 데이터 구조에 나타날 필요가 없으며, 이는 2차 객체를 식별하는 것에 관한 한 이러한 매핑 데이터 구조를 기존 핸들 테이블보다 효율적으로 만드는 데 도움을 준다.

[0015] 일부 실시예는, 가비지 콜렉션에서 객체가 포함하는 참조물과 객체가 가질 수 있는 임의의 2차 객체 모두를 표시하는 것을 자동으로 포함한다. 일부는 두 가지 종류의 표시 모두에 대해 중요한 코드 경로를 공유함으로써 효율성을 추가로 향상시킨다.

[0016] 추가적으로, 일부 실시예는 객체 내에 IS-PRIMARY 비트를 내장하며, 이러한 비트는, 객체가 1차 객체이고, 따라서 객체가 하나 이상의 2차 객체, 예컨대, 앞서 논의된 매핑에서 식별된 2차 객체를 갖는다는 것을 표시하도록 설정되어 있다. IS-PRIMARY 비트는 비-1차 객체, 즉, 2차 객체가 없는 객체를 표시하도록 클리어된다. 따라서, 이들 실시예는 일부 공간 효율성이 낮은 접근 방식이 그랬던 것처럼 각 객체에 추가 슬롯을 할당하는 것을 방지하고 있다.

[0017] 이들은 단지 예일 뿐이다. 이들 실시예의 다른 양태 및 다른 메모리 관리 실시예가 또한 여기에 설명된다.

[0018] 운영 환경

[0019] 도 1을 참조하면, 일 실시예에 대한 운영 환경(100)은 적어도 하나의 컴퓨터 시스템(102)을 포함한다. 컴퓨터 시스템(102)은 다중 프로세서 컴퓨터 시스템일 수도 있고 아닐 수도 있다. 운영 환경은 클라우드 내에서 클러스

터링, 클라이언트-서버 네트워킹, 및/또는 피어 투 피어 네트워킹될 수 있는 주어진 컴퓨터 시스템 내의 하나 이상의 머신을 포함할 수 있다. 개별 머신은 컴퓨터 시스템이고, 협력하는 머신 그룹도 또한 컴퓨터 시스템이다. 주어진 컴퓨터 시스템(102)은, 예컨대, 애플리케이션을 갖는 최종 사용자 용으로, 관리자 용으로, 서버로, 분산 처리 노드로, 및/또는 다른 방식으로 구성될 수 있다.

[0020] 인간 사용자(104)는 디스플레이, 키보드, 및 다른 주변 장치(106)를 사용함으로써, 입력된 텍스트, 터치, 음성, 움직임, 컴퓨터 비전, 제스처, 및/또는 다른 형태의 I/O를 통해, 컴퓨터 시스템(102)과 상호 작용할 수 있다. 스크린(126)은 착탈식 주변 장치(106)일 수 있거나 시스템(102)과 일체화된 부분일 수 있다. 사용자 인터페이스는 실시예와 하나 이상의 인간 사용자 간의 상호 작용을 지원할 수 있다. 사용자 인터페이스는 커맨드 라인 인터페이스(command line interface), 그래픽 사용자 인터페이스(graphical user interface)(GUI), 내추럴 사용자 인터페이스(natural user interface)(NUI), 음성 커맨드 인터페이스(voice command interface), 및/또는 다른 사용자 인터페이스(UI) 프리젠테이션을 포함할 수 있으며, 이러한 프리젠테이션은 별개의 옵션으로 제공될 수 있거나 통합될 수 있다.

[0021] 시스템 관리자, 네트워크 관리자, 클라우드 관리자, 보안 분석가 및 기타 보안 담당자, 운영 담당자, 개발자, 테스터, 엔지니어, 감사자, 및 최종 사용자는 각각 특정 타입의 사용자(104)이다. 한 명 이상의 사람을 대신하여 작용하는 자동화된 에이전트, 스크립트, 재생 소프트웨어, 디바이스 등은 또한, 예컨대, 시스템(102)의 테스트를 가능하게 하는 사용자(104)일 수도 있다. 저장 디바이스 및/또는 네트워킹 디바이스는 프로세서(110)로부터의 분리 가능성에 따라, 일부 실시예에서는 주변 장비로 간주되고 다른 실시예에서는 시스템(102)의 일부로 간주될 수 있다. 도 1에 도시되지 않은 다른 컴퓨터 시스템은, 예를 들어, 네트워크 인터페이스 장비를 통한 네트워크(108)로의 하나 이상의 연결을 사용하여 컴퓨터 시스템(102) 또는 다른 시스템 실시예와 기술적 방식으로 상호 작용할 수 있다.

[0022] 각각의 컴퓨터 시스템(102)은 적어도 하나의 프로세서(110)를 포함한다. 컴퓨터 시스템(102)은 다른 적합한 시스템과 마찬가지로 또한 하나 이상의 컴퓨터 판독 가능 저장 매체(112)를 포함한다. 저장 매체(112)는 상이한 물리적 타입일 수 있다. 저장 매체(112)는 휘발성 메모리, 비휘발성 메모리, 장소에 고정된 매체, 착탈식 매체, 자기 매체, 광학 매체, 솔리드 스테이트 매체, 및/또는 (단순히 전파된 신호 또는 단순한 에너지와는 대조되는) 다른 타입의 물리적 내구성 저장 매체일 수 있다. 특히, 휴대용(즉, 외장형) 하드 드라이브, CD, DVD, 메모리 스틱 또는 다른 착탈식 비휘발성 메모리 매체와 같은 구성된 저장 매체(114)는, 컴퓨터 시스템에 삽입되거나 설치될 때 기능적으로 컴퓨터 시스템의 기술적인 일부가 되어, 프로세서(110)와의 상호 작용을 위해 그리고 프로세서(110)에 의해 사용되기 위해 자신의 내용이 액세스될 수 있게 된다. 착탈식 구성의 저장 매체(114)는 컴퓨터 판독 가능 저장 매체(112)의 일 예이다. 컴퓨터 판독 가능 저장 매체(112)의 일부 다른 예는 내장 RAM, ROM, 하드 디스크, 및 사용자(104)에 의해 쉽게 착탈될 수 없는 다른 메모리 저장 디바이스를 포함한다. 현재 미국 특허 요구 사항을 준수하기 위해, 컴퓨터 판독 가능 매체나, 컴퓨터 판독 가능 저장 매체나, 컴퓨터 판독 가능 메모리 그 어느 것도 미국에서 계류 중이거나 특허 허여된 임의의 청구항 하에서의 신호 그 자체가 아니거나 단순한 에너지가 아니다.

[0023] 저장 매체(114)는 프로세서(110)에 의해 실행 가능한 이진 명령어(116)로 구성되며, "실행 가능"은, 예를 들어, 가상 머신에서 실행되는 머신 코드, 해석 가능 코드, 바이트 코드, 및/또는 코드를 포함하도록 본원에서 광범위한 의미로 사용된다. 저장 매체(114)는 또한 명령어(116)의 실행에 의한 기술적 효과를 위해 생성, 수정, 참조, 및/또는 사용되는 데이터(118)로 구성된다. 명령어(116) 및 데이터(118)는 이들이 상주하는 메모리 또는 다른 저장 매체(114)를 구성하며; 해당 메모리 또는 다른 컴퓨터 판독 가능 저장 매체가 주어진 컴퓨터 시스템의 기능부일 때, 명령어(116) 및 데이터(118)는 또한 해당 컴퓨터 시스템을 구성한다. 일부 실시예에서, 데이터(118)의 일부는 제품 특성, 인벤토리, 물리적 측정치, 설정치, 이미지, 판독치, 목표치, 볼륨 등과 같은 실세계 항목을 나타낸다. 이러한 데이터는 또한 백업, 복원, 커밋, 중단, 재포맷, 및/또는 다른 기술적 동작을 통해 변환된다.

[0024] 일 실시예가 컴퓨팅 디바이스(예컨대, 범용 컴퓨터, 서버, 또는 클러스터)에서 하나 이상의 프로세서에 의해 실행되는 소프트웨어 명령어로서 구현되는 것으로 설명될 수 있지만, 이러한 설명은 모든 가능한 실시예를 망라하는 것으로 의미하지는 않는다. 기술자는 동일하거나 유사한 기술적 효과를 제공하기 위해, 동일하거나 유사한 기능이 종종 전체적으로 또는 부분적으로 하드웨어 로직에서 직접 구현될 수도 있다는 것을 이해할 것이다. 소프트웨어 구현에 대해 대안적으로 또는 추가적으로, 본원에서 기술되는 기술적 기능은 적어도 부분적으로, 하나 이상의 하드웨어 로직 컴포넌트에 의해 수행될 수 있다. 예를 들어, 다른 구현예를 배제하지 않고, 실시예는 필드 프로그래머블 게이트 어레이(FPGA), 주문형 집적 회로(ASIC), 주문형 표준 제품(ASSP), 시스템 온 칩 컴포넌트

트(SOC), 복합 프로그래머블 로직 디바이스(CPLD), 및 유사한 컴포넌트와 같은 하드웨어 로직 컴포넌트(110, 128)를 포함할 수 있다. 실시예의 컴포넌트는, 예를 들어, 그의 입력, 출력, 및/또는 기술적 효과에 기반하여 상호 작용하는 기능 모듈로 그룹화될 수 있다.

[0025] 프로세서(110)(예컨대, CPU, ALU, FPU, TPU, 및/또는 GPU), 메모리/저장 매체(112), 및 디스플레이(126) 이외에도, 운영 환경은 또한 다른 하드웨어(128), 예를 들어, 배터리, 버스, 전력 공급 장치, 유선 및 무선 네트워크 인터페이스 카드를 포함할 수 있다. 명사 "스크린" 및 "디스플레이"는 본원에서 상호 교환가능하게 사용된다. 디스플레이(126)는 하나 이상의 터치 스크린, 펜 또는 태블릿으로부터의 입력에 응답하는 스크린, 또는 출력만을 위해 동작하는 스크린을 포함할 수 있다. 일부 실시예에서, 인간 사용자 I/O 디바이스(스크린, 키보드, 마우스, 태블릿, 마이크로폰, 스피커, 모션 센서 등)와 같은 주변 장치(106)는 하나 이상의 프로세서(110) 및 메모리와 동작 가능한 통신 상태로 존재할 것이다.

[0026] 일부 실시예에서, 시스템은 유선 및/또는 무선 네트워크(108)에 의해 연결된 다수의 컴퓨터를 포함한다. 네트워킹 인터페이스 장비(128)는, 예를 들어, 주어진 컴퓨터 시스템에 존재할 수 있는 패킷 교환 네트워크 인터페이스 카드, 무선 트랜시버, 또는 전화 네트워크 인터페이스와 같은 네트워크 컴포넌트를 사용하여, 네트워크(108)에 대한 액세스를 제공할 수 있다. 네트워킹 인터페이스 장비와 스위치 또는 라우터 또는 방화벽과 같은 다른 네트워크 컴포넌트의 가상화는 또한, 예컨대, 소프트웨어 정의된 네트워크 또는 샌드박스 또는 다른 보안 클라우드 컴퓨팅 환경에 존재할 수 있다. 일부 실시예에서, 하나 이상의 컴퓨터는 연결이 끊어지거나 다른 네트워크형 디바이스 또는 원격 클라우드에 간헐적으로만 연결되기 때문에 부분적으로 또는 완전히 "에어 갭(air gapped)"된다. 특히, 본원에서 교시된 메모리 관리 기능은 에어 갭 시스템(air gapped system)에 설치된 다음, 주기적으로 또는 때때로 착탈식 매체를 사용하여 업데이트될 수 있다. 주어진 실시예는 또한 직접 메모리 액세스, 착탈식 비휘발성 저장 매체, 또는 다른 정보 저장 검색 및/또는 전송 접근 방식을 통해 기술적 데이터 및/또는 기술적 명령어를 전달할 수 있다.

[0027] 기술자는 본원의 "운영 환경" 하에서 제시된 기술한 양태 및 다른 양태가 주어진 실시예의 일부를 형성할 수 있다는 것을 이해할 것이다. 이 문서의 제목은 특징을 실시예 특징 세트 및 비실시예 특징 세트에 엄격하게 분류하기 위한 것은 아니다.

[0028] 하나 이상의 항목이 도면에서 점선의 윤곽선 형태로 도시되거나 괄호 안에 나열되어, 이들 항목이 반드시 도시된 운영 환경 또는 모든 실시예의 일부는 아니지만 본원에서 논의되는 바와 같은 운영 환경 또는 일부 실시예의 항목과 상호 운용될 수 있음을 강조하고 있다. 임의의 도면이나 임의의 실시예에서 점선의 윤곽선 또는 괄호 형태 내에 있지 않은 항목이 반드시 요구되는 것은 아니다. 특히, 도 1은 편의를 위해 제공되며; 도 1에 항목이 포함되어 있다고 해서 해당 항목 또는 항목의 설명된 용도가 본 신규한 사항 이전에 알려졌다다는 의미는 아니다.

[0029] 시스템에 대한 추가 정보

[0030] 도 2는 일부 시스템(200)의 일부 양태를 나타내는 것으로, 이러한 시스템(200)은 2차 객체를 포함하는 메모리 관리를 위해 본원에 교시된 기능(202)의 적어도 일부와 함께 향상된 시스템(102)이다. 특히, 예시된 향상된 시스템(200)은 GC 스레드(206) 내에 또는 GC 표시 단계 코드(210) 또는 다른 위치 내에 설치된 가비지 콜렉터(garbage collector)(GC)(204)를 포함한다. 향상된 GC는 다음의 특성: 즉, GC가 참조물 표시와 2차 객체 표시를 통합하는 특성, GC가 객체가 1차 객체인지 여부를 표시하기 위해 객체에 내장된 참조물 사이즈의 필드에 의존하는 것을 방지하는 특성, 또는 GC가 비-1차 객체 핸들을 통해 루핑하여 2차 객체 핸들을 수집하는 것을 방지하는 특성 중 하나 이상을 가지고 있다. 이러한 기능(202) 특성은 객체에 대한 저장 요구 사항을 줄임으로써 저장 효율성(944)을 돕고, 동시적 가비지 콜렉션 및 비동시적 가비지 콜렉션 모두에서, GC 표시와 관련한 계산 요구 사항 및 계산 복잡성을 줄임으로써 계산 효율성(944)을 돕는다.

[0031] 뮤테이터 스레드(208)는 많은 경우에 사용자 스레드일 수 있고, 즉 사용자 어드레스 공간에서 실행되는 스레드일 수 있다. 그러나, 본원에서 교시된 것과 동일한 도구와 기법은 커널 어드레스 공간에서 실행되는 스레드에 대해 수행되는 가비지 콜렉션에 적용될 수 있다.

[0032] 예시된 시스템(200)의 메모리(112)는 힙(214) 내에 할당된 객체와 같은 다수의 객체(212)를 포함한다. 예시된 바와 같이, 객체(212)는 하나 이상의 2차 객체(218)를 갖는 1차 객체(216)일 수 있거나, 객체는 비-1차 객체(220)일 수 있다. 1차 객체 X의 2차 객체 Y는, Y가 그 자체의 2차 객체 Z를 가질 수 있다는 점에서, 그 자체로 1차 객체일 수도 있다. 1차 객체, 2차 객체, 및 비-1차 객체는 각각 활성 객체(222)일 수도 있거나 그렇지 않을 수도 있다. 임의의 1차-2차 객체 관계 또는 그러한 객체 관계의 부족에 관계없이, 1차 객체, 2차 객체, 및 비-1

차 객체는 각각 다른 객체(들)에 대한 0개 또는 그 이상의 참조물(224)을 가질 수도 있다. 비-1차 객체의 핸들(224)은 (적어도) 핸들 테이블(228)에서 유지될 수 있다. 핸들 테이블(228)은 데이터 구조 양태, 예를 들어, 어레이, 트리, 인덱스, 해시, 또는 포인터를 사용하여 구현될 수 있다.

[0033] 도 2에 나타난 바와 같이, 객체(212) 및 메모리(112)의 다른 부분은 가비지 콜렉션 표시 단계 동안 도달 가능(230)한 것이거나 도달 불가능(232)한 것일 수 있고, GC 표시 단계에 의해 표시(234)된 것이거나 미표시(236)된 것일 수 있고, 그리고 GC가 아닌 다른 소프트웨어에 의해 사용 중(238)인 것이거나, 그러한 소프트웨어에 의해 미사용 중(240)인 것일 수 있다. 사용 중인 것이 아니지만 재확보 가능한 것으로 아직 식별되지 않은 메모리는 플로팅 가비지(226)로 간주될 수 있다.

[0034] 언급된 바와 같이, 향상된 기능(202)은 참조된 객체 표시와 2차 객체 표시의 통합을 포함할 수 있다. 이러한 통합은 두 종류의 표시를 위한 공유 코드 경로에 의해 구현될 수 있다. 따라서, 도 2는 또한 코드 경로(246), 및 코드 경로들의 임의의 중첩(242)(공유)의 범위를 측정하기 위한 메트릭(244)을 도시하고 있다. 그러나, 통합은 표시를 호출하는 데 필요한 세부 정보가 부족하다는 점에서도 분명할 수 있다. 일부 실시예에서, 최상위 GC 표시 루틴에 대한 호출자는 종속 핸들 표시를 위한 명시적 요청의 세부 사항 없이, 단순히 해당 표시 루틴에게 참조물을 입수하여 표시하도록 요청한다. 이러한 통합 코드를 사용하면, 호출자는 종속 핸들 표시 또는 다른 2차 객체 표시를 인식할 필요가 없으며, 2차 객체 표시는 개별 객체의 레벨 다운을 제외하고는 특별한 처리를 필요로 하지 않는다.

[0035] 도 3 및 도 4는 가비지 콜렉션의 일부 양태를 추가로 나타낸 것이다. 이들 양태 중 일부는 익숙하거나, 익숙한 가비지 콜렉션 활동에서 발생하지만, 본원의 교시 내용과의 관련성은 본 신규한 사항의 가비지 콜렉션 표시를 이해한 후에만 분명해진다.

[0036] 도 3에 예시된 바와 같이, 가비지 콜렉션 코드는 프로그램 실행에 의해 연속적인 구역(302)으로 분할되는 메모리(112)의 세그먼트(300)에 대해 동작한다. 주어진 구역(302)은 도 3의 확인 표시 심볼로 표시된 바와 같이, 하나 이상의 뮤테이터 스레드에 의해 명확하게 사용 중인 것일 수 있으며, 따라서 재확보를 위한 접근이 제한된 것일 수 있다. 주어진 구역(302)은 또한 분명히 사용되지 않을 수 있으며, 따라서 도 3에서 X 표시로 나타난 바와 같이, 재확보 가능한 것일 수 있다. 구역(302)의 재확보 가능성은 또한 결정되지 않을 수 있다.

[0037] 가비지 콜렉션은 일반적으로 어드레스를 추적하거나 메모리 구역의 사용 또는 미사용 상태를 결정하고, 그에 따라 메모리 구역(또는 메모리 구역을 나타내는 데이터 구조)을 표시(308)하는 것을 포함한다. 일부 가비지 콜렉터는 표시된 구역을 발견된 제자리에 남기고, 그 후, 프로그램(120, 122, 124)은 재확보 가능한 구역의 메모리에 할당된다. 이 접근 방식은 소위 표시-소거 가비지 콜렉션 알고리즘(mark-sweep garbage collection algorithm)에서 사용된다. 다른 가비지 콜렉터는, 동일한 재확보 가능성 상태를 가진 다수의 구역을 함께 압축(306)하기 위해, 사용된 구역을 재배치하거나 재확보 가능한 구역을 재배치하거나 둘 모두를 수행한다. 표시-압축(mark-compaction)으로 알려진 이 접근 방식은 종종 표시-소거 접근 방식(mark-sweep approach)보다 큰 재확보 가능한 구역(304)을 생성하지만, 압축 결과와 매칭하도록 어드레스를 올바르게 업데이트해야 하기 때문에 더 복잡하다.

[0038] 도 3 및 도 4는 가비지 콜렉션의 단계 또는 서브 단계(402), 및 일부 동시성이 있을 때 동시성에 대한 이들의 관계를 나타내는 데 도움을 준다. 이들 도면은 표시 단계(308), 소거 단계(404), 및 압축 단계(306)를 포함하는 가비지 콜렉션 사이클의 일부 특정 단계를 도시한 것이다. 표시 단계(308) 동안, 도달 가능한 객체(230) 또는 다른 재확보 불가능한 메모리 구역은 제자리에 또는 이를 나타내는 데이터 구조 내에 표시된다. 소거 단계(404) 동안, 재확보 가능한 객체는, 예컨대, 재할당을 위해 풀(pool)에 배치됨으로써, 힙(heap)(214)으로부터 소거된다. 압축 단계(306) 동안, 재확보 가능한 구역이 함께 배치되며; 이것은 또한 사용된 구역을 함께 배치하는 것으로 간주될 수 있다. 주어진 가비지 콜렉터가 각 단계를 반드시 수행하는 것은 아니며, 예컨대, 때때로 압축 단계(306)는 생략된다.

[0039] 가비지 콜렉션에 대한 간단한 접근 방식은 가비지 콜렉터가 메모리 구역을 사용 중 또는 재확보 가능으로 표시(308)하는 동안 가비지 콜렉터를 제외한 모든 것을 단순히 일시 중단하는 것이다. 그러나, 이러한 완전한 "전체 멈춤(stop the world)" 접근 방식은 사용자 프로그램 성능을 너무 심각하게 저하시키기 때문에 대부분의 시스템에서는 허용되지 않고 있다. 따라서, 가비지 콜렉터 스레드(206) 및 임의의 뮤테이터 스레드(들)(208)의 동시적 실행을 허용함으로써 긴 "전체 멈춤(stop the world)" 서브 단계(406)를 제한하는 목표로 가비지 콜렉터가 설계될 수 있다. 표시 단계는 동시적 부분과 비동시적 부분으로 세분될 수 있다. 기능(202)의 사용은 표시 단계 동안 동시성을 증가시키는 경향이 있고, 따라서 가비지 콜렉션의 대상이 되는 사용자 코드의 성능을 향상시키는

경향이 있다. 동시성은 적어도 두 개의 스레드가 실행 시 적어도 일부 시간 동안 중첩하는 것을 포함하지만, 동시성은 두 스레드가 항상 함께 시작하거나 함께 일시 중단하거나 함께 중지할 것을 요구하지는 않는다.

[0040] 도 5는 객체(212)를 추가로 나타낸 것이다. 이 예에서, 객체 A는 2차 객체 C를 갖는 1차 객체(216)이다. 객체 A는 또한 f0으로 라벨링된 내장된 참조물(224)을 사용하여 참조된 객체(502)를 참조한다. 따라서, 도 5는 2개의 참조물(224), 즉 객체 A를 참조하는 포인터 변수(504) 내에 있는 참조물과, 객체 B를 참조하는 객체 A 내에 있는 참조물을 나타내고 있다.

[0041] 이 예에서, 포맷 정보(506)는 또한 객체 A에 내장된다. 포맷 정보 구조(506)는, 예를 들어, 객체 A의 필드 또는 다른 멤버의 타입 및 오프셋을 나타낼 수 있다. 기술자는, 도 5가 주어진 객체의 한 예지를 가리키는 화살표를 가지고 있지만, 이는 해당 객체를 식별하는 데 사용되는 어드레스가 해당 객체의 첫 번째 바이트의 어드레스임을 시사하며, 실제로 객체 지향 프로그래밍의 구현에는 객체에 할당된 구역(302) 내로의 0이 아닌 오프셋을 식별하는 어드레스를 사용할 수 있다는 것을 인식할 것이다. 예를 들어, 어드레스는 포맷 정보 또는 다른 메타 데이터를 포함하는 미리 정의된 헤더 섹션(506) 뒤에 오는 바이트를 가리킬 수 있다. 따라서 본원의 교시 내용은 기술자에 의해 조정될 수 있다.

[0042] 도 6은 1차-2차 객체 관계를 나타내는 해시 테이블(602)을 나타낸 것이다. 이 예에서, 각 1차-2차 객체 관계는 종속 핸들(606)의 일부이다. 이 예에서, 각 종속 핸들(606)은 해시 테이블(602) 내에 자체 항목(604)을 갖고, 항목은 1차 객체의 해시 코드에 따라 구성된다. 테이블(602)의 예시된 부분에서, 제1 항목은 객체 C가 객체 A의 2차 객체임을 나타내고, 제2 항목은 객체 D가 객체 C의 2차 객체임을 나타낸다. 일부 실시예는 1차 객체의 어드레스를 해시하고, 핸들을 함께 링크하며, 그 결과 1차 객체에 대한 쿼리는 해당 쿼리된 1차 객체의 모든 2차 객체를 빠르게 제공할 수 있게 된다. 압축 가비지 콜렉터에서, 일부 1차 객체 어드레스는 변경될 수도 있으며, 이후에 다시 해시될 것이다. 보다 일반적으로, 일부 실시예에서, 핸들은 1차 객체가 될 수 있는 모든 객체에서 전용 참조물 사이즈의 공간을 사용하는 대신, 1차-2차 객체 관계를 추적하는 데 사용된다.

[0043] 도 7은, 객체의 각 멤버의 데이터 타입 및 크기 또는 오프셋과 같은 것을 나타내는 데 사용될 수 있는 비트(700)를 포함하여, 객체의 포맷 정보 부분(506)으로서 구성된 메모리를 나타낸 것이다. 본원의 교시 내용에 따라, 친숙한 인스턴스 포맷 정보는, 객체가 1차 객체인지 여부를 표시하는 IS-PRIMARY 비트(702)를 포함하도록 향상될 수 있다. 객체가 1차 객체인 경우, 객체의 하나 이상의 2차 객체가, 예컨대, 해시 테이블(602)에서 식별된다. 도 7에서 IS-PRIMARY 비트(702)의 위치는 단지 예시일 뿐이며, 규범적이지는 않다.

[0044] 일부 실시예는 (일부 다른 접근 방식과는 달리) 모든 객체가 참조물을 보유하기에 충분히 큰 공간을 따로 두도록 요구하지 않고도 2차 객체를 사용할 수 있다는 이점을 제공한다(942). 실시예는 객체당 하나의 비트(IS-PRIMARY 비트)와 실제의 1차 및 2차 객체 식별자를 위한 공간을 따로 두고 있다. 이렇게 하면 객체(212)가 증식될 때 공간이 절약된다. 런타임 환경(100)은 이와 같이 무언가를 위해 사용할 수 있는 몇 개의 스페어 비트(spare bit)(700)를 가질 수 있으므로, 일단 본원의 교시 내용이 이해되면, 이러한 비트(702)를 구현하는 것은 기존 런타임의 코드베이스에 반드시 매우 파괴적인 것은 아니다.

[0045] 일부 실시예의 다른 이점은 2차 객체를 표시하는 것이 참조된 객체를 표시하는 것과 통합된다는 점이다. 이는, 기존 GC 코드에 대해 비교적 작은 중단 없이 다른 작업도 수행하게 함으로써, 기존 코드(210)를 표시에 활용한다.

[0046] 특히, 일부 실시예는 본원에서 교시된 2차 객체(218)를 사용하여 메모리를 관리(936)하도록 구성된 시스템(200) 또는 다른 시스템(102)과 같은 2차 객체 메모리 관리 기능 향상 시스템을 사용하거나 제공한다. 향상된 시스템(200)은 메모리(112), 및 메모리와 동작 가능하게 통신하는 프로세서(110)를 포함한다. 프로세서(110)는 메모리 관리 단계를 수행하도록 구성되고, 메모리 관리 단계는 (a) 다수의 객체(212)에 내장된 전용 공간(704)에, 다수의 객체 중 각각의 객체가 1차 객체(216)인지 여부를 표시하는 단계 - 각각의 전용 공간은 객체 참조물(224)보다 작고, 1차 객체는, 1차 객체가 활성 상태인 경우, 1차 객체의 참조된 객체는 아니면서 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체(218)를 갖는 객체임 -, 및 (b) 가비지 콜렉션 표시 동안 적어도 (i) 각 활성 객체 X의 모든 참조된 객체 및 (ii) 각 활성 객체 X의 모든 2차 객체 모두를 활성 상태로 표시함으로써 2차 객체(218)를 참조된 객체(502)로 취급하는 단계를 포함한다. 향상된 시스템(200)은 가비지 콜렉션 동안 참조된 객체의 표시와 2차 객체의 표시를 통합하여, 1차 객체가 아닌 객체(220)에게 2차 객체에 대한 참조물을 보유하기 위한 참조물 사이즈의 공간을 내장할 것을 요구하지 않고, 추가 사용을 위한 메모리를 복구한다.

- [0047] 일부 실시예는 항목(604)을 갖는 해시 테이블(602)을 더 포함한다. 각 해시 테이블 항목은 1차 객체(216)에 대한 1차 객체 참조물(224) 및 2차 객체(218)에 대한 2차 객체 참조물(224)을 포함하여, 2차 객체가 1차 객체의 2차 객체임을 표시한다. 다른 실시예에서, 1차-2차 객체 관계는 다른 데이터 구조를 사용하여 표현된다. 사용된 특정 데이터 구조에 관계없이, 이러한 표현은, 객체가 1차 객체로서 동작하는 경우에만 모든 객체에 공간을 예방 조치적으로 할당하는 비효율적인 접근 방식과는 달리, 실제로 1차-2차 객체 관계에 관련된 객체에만 유리하게 제한될 수 있다.
- [0048] 일부 실시예에서, 객체에 내장되어 해당 객체가 1차 객체인지 여부를 표시하기 위한 각 전용 공간(704)은 단일 비트로 구성된다. 기술자는, 이 비트가 서로 다른 구현예에서 다르게 라벨링될 수 있으며, 예컨대, 내부 프로토타입에서, 비트(702)는 "HAR" 비트("참조물로 추가된(Have Added References)" 비트)로 지칭되었지만, 도 7에서, 비트(702)는 "IS-PRIMARY" 비트로 라벨링된다는 것을 이해하고 있다.
- [0049] 라벨링에 관계없이, 실시예는 모든 객체에서 전용 2차 참조 공간의 비효율성을 방지할 수 있다. 이러한 방지를 통해, 1차 객체가 아닌 객체가 참조된 객체(502)에 대한 참조물(224)을 갖고 있는 경우에도 공간이 절약된다. 특히, 일부 실시예에서, 객체(212)는 1차 객체가 아닌 객체 Z(220)를 포함하고, 객체 Z는 2차 객체를 갖지 않으며, 또한 참조물도 내장하지 않는다.
- [0050] 일부 실시예에서, 1차 객체는 다수의 2차 객체를 갖는다. 따라서, 일부에서 객체(212)는 적어도 2개의 2차 객체(218)를 갖는 1차 객체 Y(216)를 포함한다.
- [0051] 일부 실시예에서, 2차 객체는 그 자체가 1차 객체일 수 있다. 따라서, 일부에서 객체(212)는 1차 객체(216)이기도 한 2차 객체 Q(218)를 포함한다.
- [0052] 생산 시나리오는 일부 실시예의 공간 절약 이점을 나타내는 데 도움을 준다. 일부에서, 향상된 시스템(200)에서 적어도 부분적으로 실행된 프로그램의 객체(212)의 최대 10퍼센트는 1차 객체(216)이다. 따라서, 적어도 90%는 포인터 필드와 같은 NULL 참조물 사이즈의 필드를 사용하는 대신, 1비트(702)만을 사용하여 1차 객체가 아님을 표시함으로써 효율성(944)에 기여한다. 이러한 절약은 해시 테이블 또는 다른 1차-2차 객체 관계 표현 데이터 구조(602)에 필요한 스토리지에 의해 일부 상쇄될 수 있지만, 모든 객체의 포인터 대신 모든 객체의 비트를 균형적으로 사용하면 공간이 절약된다. 기술자는, 이러한 최대 10퍼센트가 모든 실시예에 대한 절대적인 요구 사항이 아니라, 일부 상황에서 충족되는 예시적인 비율에 불과하다는 것을 이해할 것이다.
- [0053] 일부 시나리오에서, 향상된 시스템(200)에서 적어도 부분적으로 실행된 프로그램은 적어도 10,000개의 객체(212)를 가지며, 이들 객체 중 각 객체 X는 객체 X가 1차 객체인지 여부를 표시하기 위한 4 비트 이하의 각각의 내장된 전용 공간(704)을 갖는다. 각 객체 X에는 또한 임의의 다른 객체를 객체 X의 2차 객체로 식별하기 위한 4개의 비트보다 큰 어떠한 전용 공간도 내장되지 않는다.
- [0054] 다른 시스템 실시예는 또한 본원에서, 컴퓨팅 하드웨어에 대한 광범위한 논의에 의해 제공되는 설명된 프로세스 또는 구성된 매체의 시스템 버전으로서 직접 또는 도출 가능하게 기술된다.
- [0055] 특정 아키텍처 예가 도면에 도시되어 있지만, 실시예는 이러한 예를 벗어날 수 있다. 예를 들어, 서로 다른 도면에 표시된 항목은 일 실시예에서 함께 포함될 수 있거나, 도면에 표시된 항목은 생략될 수 있거나, 서로 다른 항목에 표시된 기능은 더 적은 수의 항목 또는 단일 항목으로 결합될 수 있거나, 항목의 이름이 변경되거나, 이들 항목이 서로에 다르게 연결될 수 있다.
- [0056] 본 개시 내용에서는 기술의 양태를 기술하는 데 도움을 주는 예가 제공되지만, 이 문서에 제공된 예는 이용 가능한 모든 실시예를 기술하지는 않는다. 주어진 실시예는, 예를 들어, 추가적이거나 서로 다른 기술적 특징, 메커니즘, 시퀀스, 데이터 구조, 또는 기능을 포함할 수 있고, 본원에 제공된 예로부터 달리 벗어날 수도 있다.
- [0057] **프로세스(일명, 방법)**
- [0058] 도 8은 본원에 교시된 바와 같은 시스템(200)과 같은 향상된 시스템, 또는 다른 기능 향상된 시스템에 의해 수행되거나 지원될 수 있는 방법(800)의 균을 예시한 것이다. 도 9는 본원에 교시된 신규한 기능을 갖는 시스템의 동작 동안 사용하기에 적합한 메모리 관리 방법(이는 또한 그 단어의 법적 의미에서 "프로세스"로 지칭될 수도 있음)을 추가로 예시한 것이다. 도 9는 도 8에 표시된 단계에 대한 몇 가지 개선 사항, 보완 사항, 또는 상황에 맞는 조치를 포함한다. 도 9는 도 8에 표시된 단계를 또한 포함한다. 도면에 도시되거나 개시된 기술적 프로세스는, 달리 표시되지 않는 한, 예컨대, 공간(704)을 관독하는 가비지 콜렉션 코드에 의해 그리고 공간(704)을 초기화하는 객체 생성자 코드에 의해, 자동으로 수행될 것이다. 프로세스는 또한 인간 관리자 또는 다른 인간에

의한 동작이 연루된 한도까지는 부분적으로는 자동으로 그리고 부분적으로는 수동으로 수행될 수 있으며, 예컨대, 일부 실시예에서 개발자는 가비지 콜렉션 사이클에 대한 상대적 빈도를 특정할 수 있다. 본원에서 신규 사항으로 고려된 프로세스는 전적으로 수동인 것은 아니다. 주어진 실시예에서, 프로세스의 0 개 또는 그 이상의 예시된 단계는 아마도 동작할 서로 다른 파라미터 또는 데이터로 반복될 수 있다. 일 실시예의 단계는 또한 도 8 및 도 9에 나열된 하향식 순서와는 다른 순서로 수행될 수도 있다. 이들 단계는 연속적으로, 부분적으로 중첩하는 방식으로, 또는 완전히 병렬로 수행될 수 있다. 특히, 프로세스 동안 수행되는 단계를 나타내기 위해 제어 플로우차트(800) 동작 항목 또는 제어 플로우차트(900) 동작 항목이 횡단되는 순서는 프로세스의 수행마다 달라질 수 있다. 차트의 횡단 순서는 또한 프로세스 실시예마다 달라질 수 있다. 이들 단계는 또한, 수행되는 프로세스가 동작 가능하고 적어도 하나의 청구항을 준수하는 경우, 생략, 결합, 이름 변경, 재그룹화될 수 있거나, 하나 이상의 머신 상에서 수행될 수 있거나, 또는 이와는 달리, 예시된 플로우에서 벗어날 수도 있다.

[0059] 일부 실시예는 2차 객체를 사용하여 컴퓨팅 시스템 메모리를 관리(936)하기 위한 방법을 사용하거나 제공하며, 상기 방법은 자동으로 수행되는 단계: 즉, 다수의 객체에 내장된 전용 공간(704)에, 다수의 객체 중 각각의 객체가 1차 객체인지 여부를 표시하는 단계(802) — 각각의 전용 공간은 객체 참조물보다 작고, 1차 객체는, 1차 객체가 활성 상태인 경우, 1차 객체의 참조된 객체는 아니면서 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체를 갖는 객체임 —; 가비지 콜렉션 표시(308) 동안 적어도 (i) 각 활성 객체 X의 모든 참조된 객체(502) 및 (ii) 각 활성 객체 X의 모든 2차 객체(218) 모두(808)를 활성 상태(222)로 표시(806)함으로써 2차 객체를 참조된 객체로 취급하는 단계(902); 및 1차 객체가 아닌 객체(220)에, 2차 객체(218)에 대한 참조물(224)을 보유하기 위한 임의의 참조물 사이즈의 공간을 내장(816)하는 것을 방지하는 단계(812)를 포함한다. 압축(306) 또는 소거(404)와 같은 추가적인 가비지 콜렉션 처리가 또한 수행(814)될 수 있다.

[0060] 일부 실시예에서, 가비지 콜렉션 표시의 적어도 일부는 활성 상태에 대해 적어도 개별 객체가 확인되는 지점까지 2차 객체 표시 및 참조된 객체 표시에 의해 공유되는 코드 경로(246)를 따른다(904). 예를 들어, 일부 실시예에서 동일한 코드 경로는 go_through_objects 매크로 호출 또는 개별 객체를 표시하는 가장 낮은 루틴(가장 세밀한 단위)의 다른 호출까지 이어진다. 이들 실시예에서, 이 입도를 초과하는 코드는 GC 표시와 관련하여 참조된 객체와 2차 객체를 구별하지 않는다.

[0061] 추가 예로서, 다음의 의사 코드는 go_through_objects의 사용을 예시한다:

[0062] if (ObjA.ContainsPointers || ObjA.IsPrimary)

[0063] {

[0064] go_through_objects(Objects ObjA, fn mark)

[0065] {

[0066] while (notend())

[0067] {

[0068] Object r = GetNextReference();

[0069] mark(r);

[0070] {

[0071] }

[0072] }

[0073] 다른 예로서, 일부의 실시예에서 가비지 콜렉션 표시의 적어도 일부는 적어도 활성 상태에 대해 개별 객체를 확인하는 것(906)을 지원하기 위해 객체 참조물이 열거되는 지점까지 2차 객체 표시 및 참조된 객체 표시에 의해 공유되는 코드 경로(246)를 따른다(904). 예를 들어, 일부 실시예에서 동일한 코드 경로는 GetNextReference() 호출 또는 표시할 객체의 어드레스를 획득하는 가장 낮은 루틴(가장 세밀한 단위)의 다른 호출까지 이어진다.

[0074] 일부 실시예는 이전 접근 방식보다 빠르게 객체를 표시함으로써 더 나은 성능을 제공한다(942). 예를 들어, 일부 실시예는 비-1차 객체를 포함하는 핸들 테이블을 통해 루핑하여 2차 객체 핸들을 수집하는 것에 대한 의존성을 방지한다(916). 일부는 2차 객체 핸들을 수집하기 위해 활성 프로모션 카운터(promotion counter)(922)에 대

한 의존성을 방지한다(920).

- [0075] 일부 실시예는 객체를 생성하기 위한 요청의 결과로서 객체에 대한 핸들을 리턴(926)하기 전에, 객체에 내장된 비트(702)를 설정(924)하여, 객체가 1차 객체임을 표시한다.
- [0076] 일부 실시예는 해시 테이블을 사용(928)하여 객체가 1차 객체의 2차 객체임을 지정한다. 일부에서는 해시 테이블 또는 다른 데이터 구조는 다수의 2차 객체를 단일 1차 객체와 연관시킨다. 따라서, 일부는 데이터 구조(602)를 사용(928)하여 객체 X 및 객체 Y가 각각 1차 객체 Z의 2차 객체임을 지정한다. 일부 실시예는, 예를 들어, SetPrimary() 또는 SetSecondary() 호출에 따라 해시 테이블 또는 다른 데이터 구조를 업데이트하는 API를 포함한다. 일부 방법(900)은 특정 1차 객체-2차 객체 관계의 적어도 하나의 객체를 대체(912)하기 위한 커맨드를 수신하는 단계(910), 및 커맨드를 수행하는 동안 1차 객체-2차 객체 관계를 지정하는 데이터 구조를 업데이트하는 단계(914)를 포함한다.
- [0077] 일부 상황에서의 일부 실시예는 2차 객체를 1차 객체로 취급하고, 그에 따라 표시한다. 일부 방법에서, 2차 객체 X를 그 자체의 1차 객체로 취급하는 단계(930)는 객체 X의 각 참조된 객체를 표시하는 단계(308) 및 객체 X의 각 2차 객체를 표시하는 단계를 포함한다.
- [0078] **구성된 저장 매체**
- [0079] 일부 실시예는 구성된 컴퓨터 판독 가능 저장 매체(112)를 포함한다. 저장 매체(112)는 디스크(자기, 광학 또는 다른 것), RAM, EEPROM 또는 다른 ROM, 및/또는 특히 (단순한 전과 신호가 아닌) 컴퓨터 판독 가능한 저장 매체를 포함하는 다른 구성 가능한 메모리를 포함할 수 있다. 구성되는 저장 매체는 특히 CD, DVD 또는 플래시 메모리와 같은 착탈식 저장 매체(114)일 수 있다. 착탈 가능하거나 불가능할 수 있고 휘발성이거나 비휘발성일 수 있는 범용 메모리는, 착탈식 저장 매체(114)로부터 및/또는 네트워크 연결과 같은 다른 소스로부터 판독된 데이터(118) 및 명령어(116)의 형태로, 1차성을 표시하기 위해 전용되는 IS-PRIMARY 비트(702) 또는 다른 공간(704)과 같은 항목, 1차 객체(216), 2차 객체(218), 종속 핸들(606), 1차-2차 객체 관계 데이터 구조(602), 및 가비지 콜렉션(938) 동안 참조된 객체 표시와 2차 객체 표시를 통합하는 중첩하는 코드 경로(246)를 사용하는 실시예로 구성되어, 구성된 저장 매체를 형성할 수 있다. 구성된 저장 매체(112)는 컴퓨터 시스템(102)으로 하여금 본원에 개시된 바와 같이, 2차 객체를 이용하여 메모리를 관리하기 위한 기술적 프로세스 단계를 수행하게 할 수 있다. 따라서, 도면은 구성된 저장 매체 실시예 및 프로세스(일명, 방법) 실시예뿐만 아니라 시스템 실시예 및 프로세스 실시예를 예시하는 데 도움을 준다. 특히, 도 8 또는 도 9에 도시되거나 본원에서 교시되는 임의의 프로세스 단계는 저장 매체를 구성하여 구성된 저장 매체 실시예를 형성하는 것을 돕는 데 사용될 수 있다.
- [0080] 일부 실시예는 데이터(118) 및 명령어(116)로 구성된 컴퓨터 판독 가능 저장 매체(112, 114)를 사용하거나 제공하며, 이 저장 매체는 적어도 하나의 프로세서(110)에 의한 실행 시 컴퓨팅 시스템으로 하여금 2차 객체를 사용하여 메모리를 관리하기 위한 방법을 수행하게 한다. 이 방법은, 다수의 객체에 내장된 전용 1차성 공간(704)에, 다수의 객체 중 각각의 객체가 1차 객체(216)인지 여부를 표시하는 단계(802) — 각각의 전용 공간(704)은 객체 참조물(224)보다 작고, 1차 객체는, 1차 객체가 활성 상태인 경우, 1차 객체의 참조된 객체는 아니면서 가비지 콜렉션에 의해 활성 상태인 것으로 간주되는 적어도 하나의 2차 객체(218)를 갖는 객체임 —; 가비지 콜렉션 표시(308) 동안 적어도 (i) 각 활성 객체 X의 모든 참조된 객체(502) 및 (ii) 각 활성 객체 X의 모든 2차 객체(218) 모두를 활성 상태로 표시함으로써 2차 객체를 참조된 객체로 취급하는 단계(902); 및 1차 객체가 아닌 객체에, 2차 객체에 대한 참조물을 보유하기 위한 임의의 참조물 사이즈의 공간을 내장(816)하는 것을 방지하는 단계(812) — 각각의 참조물 사이즈의 공간은 적어도 32 비트의 사이즈임 —를 포함한다.
- [0081] 일부 실시예에서, 참조된 객체 표시와 2차 객체 표시의 통합은 실시예의 코드 활용 및 성능 속도 이점에 기여한다. 일부에서, 이 통합은 코드 경로를 중첩(242)하는 것을 포함한다. 코드 경로는 동일한 실행 경로를 따르는 한도까지는 중첩된다.
- [0082] 중첩(242)은 다양한 방식으로 측정될 수 있다. 예를 들어, 스택의 레벨에서 중첩이 측정될 수 있다. 개별 문자가 실행 경로를 따라 호출되는 각각의 메서드 또는 다른 루틴을 나타낸다고 가정한다. 그러면 경로 ABCDEFGKLM과 경로 ZYABCDEFGR은 ABCDEFG의 중첩을 갖는다. ABCDEFGKLM이 참조된 객체 표시 경로이고 ZYABCDEFGR이 2차 객체 표시 경로인 경우, ABCDEFG의 중첩은 참조된 객체 표시 경로의 70%이다. 대안적으로, 공유 코드 블록, 공유 코드 라인, 또는 공유 명령어와 관련한 중첩(242)이 측정될 수 있다. 달리 명시되지 않는 한, 이러한 측정(244) 중 임의의 것이 사용될 수 있으며, 중첩 백분율은 가장 큰 백분율이다. 보다 일반적으로, 향상된 GC가 표

시(308)할 때, "참조물을 입수하여 표시하는" 작업을 수행하는 공통 코드 경로가 있다. 종속 핸들 표시가 이 공통 코드에 통합되어 있기 때문에, 개별 객체 표시의 레벨을 초과하는 호출자는 참조된 객체 표시와 2차 객체 표시 간의 구별을 인식할 필요가 없으며, 이전에 향상되지 않은 시스템(102)에 기능(202)을 구현할 때 특별한 처리가 필요하지 않다.

[0083] 특히, 일부 실시예에서, 방법은 가비지 콜렉션 동안 참조된 객체를 표시하기 위해 참조된 객체 표시 코드 경로 246-R0를 따르는 단계(904)를 포함하고, 가비지 콜렉션 동안 2차 객체를 표시하기 위해 2차 객체 표시 코드 경로 246-S0를 따르는 단계(904)를 포함한다. 이들 중 일부에서, 참조된 객체 표시 코드 경로의 적어도 50%는 2차 객체 표시 코드 경로와 중첩한다(242). 중첩(242)은, 라이브러리 코드를 포함하거나 라이브러리 코드를 제외하거나 또는 둘 다인지 여부, 그리고 실행된 모든 경로 또는 모든 이용 가능한 경로 또는 최단 경로만을 따르는지 여부에 관계없이, 다음의 중첩 매트릭(244): 즉, 코드 라인의 수, 컴파일러에 의해 생성된 명령어의 수, 호출된 루틴의 수, 또는 스택 프레임의 수 중 적어도 하나에 따라 측정된다.

[0084] 일부 실시예에서, 통합된 참조물 및 2차 객체 표시는 종속 핸들 표시가 표시 단계(308)의 마지막 서브 단계에서 라기보다는 더 일찍 수행될 수 있기 때문에 향상된 유연성을 제공한다. 이러한 유연성이 없으면, 종속 핸들 표시는 객체를 활성 상태로 만들 수 있는 모든 서브 단계로부터 완전한 결과를 얻기 위해 마지막 서브 단계가 되어야 할 수도 있으며, 따라서 모든 활성 1차 객체의 2차 객체를 표시할 수 있다. 그러나, 이러한 경직성은 다른 서브 단계를 표시 단계(308)의 마지막 서브 단계로 만드는 것과 충돌하여, 잠재적으로 "마지막"으로 알려진 서브 단계의 다수의 인스턴스가 차례로 정리되도록 유도한다.

[0085] 일부 실시예에서, 참조된 객체 표시와 통합되는 2차 객체 표시를 위한 향상된 표시 방법은, 애플리케이션 프로그램의 실행과 동시에(810) 2차 객체의 가비지 콜렉션 표시를 수행함으로써, 전체 멈춤(stop the world) 조건(406)을 부과(932)하는 것을 방지한다. 실제로, 일부 실시예에서, 이러한 방법은 백그라운드 가비지 콜렉터(background garbage collector)의 일부로서 구현된다. 따라서, 이러한 방법은 동시적 가비지 콜렉션의 일부로서 수행된다(940). 다른 실시예에서, 이러한 방법은 비동시적 가비지 콜렉션의 일부로서 수행된다. 어떤 상황에서든, 이러한 방법은 하나 이상의 STW 서브 단계의 길이를 줄임으로써 성능을 향상시킬 수 있다.

[0086] 일부 실시예에서, 실시예가 1차 객체(216) 및 비-1차 객체 객체(220) 모두의 핸들을 포함하는 임의의 핸들 데이터 구조를 통해 루핑(918)하여 표시(308)를 위한 2차 객체(218) 핸들을 수집하는 것에 대한 의존성을 방지하기 때문에, 일부 이전 접근 방식에 비해 성능 속도가 향상되었다.

[0087] 추가 세부 사항, 예 및 관찰

[0088] 위의 논의에 대한 추가 지원이 아래에 제공된다. 편의를 위해, 이 추가 지원 자료는 다양한 제목 아래에 나타나고 있다. 그럼에도 불구하고, 이는 모두 고려되는 실시예에 대한 본 개시 내용의 통합되고 일체화된 논의 부분으로서 이해되도록 의도된다.

[0089] 동시적 종속 핸들 및 최종화 스캐닝

[0090] GC의 동시적 플레이버(concurrent flavor)인 백그라운드 가비지 콜렉션(background garbage collection)(BGC)은 STW 부분을 부분적으로 단축함으로써 대기 시간에 대해 최적화된다. 향상되지 않은 BGC의 한 플로우에는(STW에서 수행되는) Init 초기화 단계, 이에 후속되는 동시 표시 단계(STW는 아님), 이에 후속되는 Final Mark 최종 표시 단계(STW), 이에 후속되는 소거 단계를 포함한다. 이러한 BGC의 최종 표시 단계는 캐치업(catch up) 서브 단계, 종속 핸들 표시 서브 단계, 짧고 약한 NULLING 서브 단계, 최종화 스캐닝 서브 단계, 및 길고 약한 NULLING 서브 단계를 포함한다.

[0091] 이러한 BGC 예에서, 표시 작업의 벌크는 사용자 스레드와 동시에 수행되며; 소거는 모두 동시에 수행된다. Init 및 Final Mark는 STW로 수행되며 일반적으로 매우 짧다. Init는 BGC 사이클이 수행되어야 한다고 결정하고, 스택 루트를 채우기 위해 호출 스택을 수행한다(이 시점에서는 표시가 수행되지 않는다). Final Mark는 캐치업을 수행하여 메모리 관리 상태에 대한 정확한 그림을 입수하는 것으로 시작된다. 캐치업은 현재 루트를 입수하고 그것을 다시 표시하는 것을 포함하고; 대부분의 객체가 이미 표시되어 있기 때문에 이러한 표시 부분은 일반적으로 매우 빠르게 완료될 것이다. 캐치업은 또한 힙의 수정된 부분을 재방문하는 것을 포함한다. 대부분의 객체가 이미 표시되어 있으므로, 여기에서 수행할 작업이 많지 않은 경우가 많다(이 예는 동시 표시 동안 동시적으로 재방문하므로 STW 동안의 재방문도 일반적으로 매우 빠르다).

[0092] 이 시점에서 BGC는 다음의 임의의 구성 범주와 관련이 있는 것을 제외하고는 어떤 객체가 활성 상태에 있어야

하는지를 알고 있다.

- [0093] 최종화 가능한 객체: BGC는 최종화 가능한 객체를 검토하고, 비활성화될 것으로 GC가 확인한 객체와 그들이 보유하고 있는 객체를 홍보해야 한다. 이것은 최종화 스캐닝이라고 지칭된다. 최종화는, 예컨대, GC가 객체가 비활성화된 것으로 표시할 때 폴에 객체를 다시 추가함으로써(최종화기가 실행됨), 관리되는 객체의 수명을 관리하는 데 도움을 주기 위해 사용될 수 있다. (최종화기를 실행하기 위해) 비활성화될 것으로 확인된 최종화 가능한 객체와 그 자식 객체를 홍보하는 데 걸리는 시간은 상당할 수 있다.
- [0094] 종속 핸들: BGC는 종속 핸들(dependent handle)(DH)의 1차 객체 타겟을 검토하고, 활성 1차 객체의 경우 그의 2차 객체를 홍보해야 한다(그들을 활성 상태로 취급한다). 임의의 2차 객체로 인해 임의의 DH의 1차 객체가 활성 상태로 되면, BGC는 해당 DH에 대해 동일한 작업을 수행해야 한다. 이것은 DH 스캐닝이라고 지칭된다. DH는 편집 및 지속을 위한 코어 언어 런타임(Core Language Runtime)에서만 사용되었기 때문에 드물게 사용되었다. 그런 다음 DH의 일부 런타임은, 예컨대, xaml과 상호 작용하기 위한 종속 핸들의 사용을 위해, 추가되었다. 그런 다음 코어 언어 런타임은 ConditionalWeakTable을 노출했다. 이 시점에서, 일부 코드에서 DH를 사용하는 경우는 드물지 않으므로, 비향상된 코드를 사용하는 경우 DH로 인해 객체를 홍보하는 데 걸리는 시간이 또한 길어질 수 있다.
- [0095] 널링(NULLing): NULL은 비활성될 것으로 확인되면 짧고 약하거나 길고 약한 핸들의 타겟이 된다. 주목할 것은 이 예에서 DH는 길고 약한 시맨틱을 사용하므로 길고 약한 핸들로 널링될 것이라는 것이다(예컨대, 일 구현예에서는 둘 다 GCScan::GcWeakPtrScan에서 수행된다). 짧고/길고 약한 핸들 타겟 널링(NULLing)은 핸들이 많은 경우 시간이 좀 필요할 수 있지만 NULL 처리 자체는 빠르기 때문에 일반적으로 상당한 성능 문제가 예상되지는 않는다.
- [0096] 이러한 구성 범주는 일반적으로 애플리케이션(124)에서 선택 사항이다. 그러나, 이러한 구성 범주가 존재하는 경우, 이러한 예시적인 BGC는 STW로 수행되는 Final Mark 단계에서 이를 처리한다. 힙 사이즈가 커질수록, 코드가 임의의 또는 모든 이러한 구성 범주를 경우에 Final Mark가 더 오래 걸릴 가능성이 높아진다. 본원의 교시 내용은 가비지 콜렉션을 더 많이 동시적으로 수행함으로써, 예컨대, STW 서브 단계를 줄이거나 제거함으로써, 대기 시간을 줄이는 데 적용될 수 있다.
- [0097] 일부 레거시 코드는 종속 핸들 타입을 사용자 코드에 공개적으로 노출하지 않으며, 사용자 코드는 BCL API를 사용하여 간접적으로만 종속 핸들을 생성할 수 있다. 기본 클래스 라이브러리(Base Class Library)(BCL)는 런타임 바로 위에 구축된 가장 낮은 관리 코드 계층이다. 이 BCL API의 구현에는 종속 핸들을 할당하거나 해제할 수 있다. 런타임에 대해, BCL은 사용자 코드이지만, 최종 사용자 코드는 아닌데, 그 이유는 최종 사용자 코드가, 예컨대, 종속 핸들을 생성할 수 없는 방식으로, BCL이 런타임과 통신할 수 있기 때문이다. 어느 시점에서, 종속 핸들은 최종 사용자 코드에 공개적으로 노출될 수 있다. 본원에 교시된 기능(202)은, 이러한 노출이 종속 핸들의 사용을 증가시키는 한도까지는, 성능 이점을 제공(942)할 수 있다.
- [0098] 개선된 기능(202)으로 레거시 BGC를 향상시키는 방법을 고려할 때, 하나의 목표는 확장성이다. 일부 시나리오에서는 최종화를 위해 준비된 임의의 수의 객체가 있을 수 있거나 또는 임의의 수의 종속 핸들이 있을 수 있다. 성능 향상은 애플리케이션 코드와 동시적으로 또는 대부분 동시적으로 실행할 수 있는 경우 더 바람직할 수 있다.
- [0099] 일부 시나리오에서, 또 다른 고려 사항은 본 프로그램 시맨틱을 유지하는 방법에 있다. 일부 환경은, 사용자 코드가 짧은 핸들과 길고 약한 핸들을 모두 사용하여 동일한 객체를 가리키는 패턴을 포함하며, 코드는, 짧고 약한 핸들의 타겟이 널(null)이고 길고 약한 핸들의 타겟이 널(null)이 아닌 경우, 최종화기가 실행될 것이라는 것을 가정한다. 일부 실시예는 상황을 더 동시적으로 만들 경우에 이 동작을 유지한다. 일부에서는 DH의 길고 약한 시맨틱이 유지된다.
- [0100] 일부 시나리오에서, 또 다른 고려 사항은 DH 스캐닝을 더 빠르게 만드는 방법에 있다. 일부 레거시 코드에서는 DH가 추가 기능으로 간주되었기 때문에 DH 스캐닝이 별도의 단계로서 수행된다. 향상되지 않은 DH 스캐닝은 그다지 성능이 좋지 않은 단순한 알고리즘을 사용한다. 향상되지 않은 DH 스캐닝은 또한 표시 중 마지막 단계가 되도록 시도한다는 점에서, 나머지 BGC와 양호하게 통합되지는 않는다. 이는 DH 스캐닝 자체가 표시 오버플로우와 같은 다른 작업을 도입하여 표시 오버플로우 처리 및 DH 스캐닝을 인터리브하게 할 수 있기 때문에 어색하다. 다른 타입의 루트도 있는 경우, 루트도 또한 인터리빙을 시도하여 코드 플로우 및 디자인을 더욱 더 지저분하게 만들 수 있다.

- [0101] 본원의 교시 내용은 향상되지 않은 DH 스캐닝의 두 결함을 모두 다루고 있다. 본원의 교시 내용은 DH 스캐닝을 처리하고 이를 더욱 동시적으로 만드는 향상된 방법을 기술하고 있다. 다음 관찰을 염두에 두면, 기술자는 향상된 DH 스캐닝을 기존 표시 코드로 폴딩하는 방법을 알 수 있다.
- [0102] 첫째, DH 또는 다른 1차-2차 객체 관계는 1차 객체가 활성 상태인 것으로 간주되는 경우 그의 2차 객체가 활성 상태로 간주되어야 함을 나타낸다. 이러한 수명 고려 사항은 1차 객체가 포함하는 임의의 참조물의 경우와 동일하며; 객체가 표시되면(활성 상태인 것으로 간주되면), 그 객체가 포함하고 있는 모든 참조물이 표시되어야 한다. 그리고 이 객체가 DH 또는 다수의 DH의 1차 객체인 경우, 그의 해당 제2 객체도 표시되어야 한다.
- [0103] 둘째, GC가 객체를 표시할 경우, 객체가 포함하는 참조물을 입수하여 표시한다. 일부 레거시 코드에서, 이것은 `go_through_object`로 지칭되는 루틴에서 수행되며, 이러한 루틴은 이러한 객체가 포함하는 참조물을 찾을 위치를 GC에 알려주는 정보를 디코딩한다. 향상된 코드가 이 객체가 해당 제2 객체를 갖고 있음을 알고 있는 경우, 이들 2차 객체를 표시 목적을 위한 참조물로서 통합할 수도 있으므로, 다른 구현예에서 `go_through_object` 또는 유사한 객체 관계 추적 루틴도 이러한 통합된 참조물을 표시할 수 있을 것이다. 2차 객체는 표시 이외의 목적을 위한 참조된 객체가 되지 않으며, 예컨대, 디버거는 단순히 1차 객체에 내장된 어드레스를 따라가는 것으로는 1차 객체에서 2차 객체로 추적할 수는 없다.
- [0104] 일부 실시예는 IRM 기능(202)을 포함하며, 여기서 IRM은 통합된 참조물 표시(Integrated Reference Marking)에 대한 약어이다. IRM 기능(202)은 참조된 객체 표시와 2차 객체 표시를 통합하기 때문에, "통합된 참조물 및 2차 객체 표시"라고 지칭될 수도 있다. IRM 기능(202)에 따라, 표시(308)는 DH에 의해 추가된 통합된 참조물을 인식하고 있다. `go_through_object`를 호출하는 코드 경로는 IRM 기능(202)이 추가될 때 변경되지 않은 상태로 유지될 수 있는데, 그 이유는 IRM이 각 객체가 포함하는 모든 참조물(기존 참조물 또는 2차 객체에 대한 참조물)을 자동으로 포함할 것이기 때문이다. 이것은 이전의 개별 DH 스캐닝보다 훨씬 더 양호한 솔루션이며, 기존의 참조 객체 스캐닝의 동시 표시를 활용함으로써 DH 스캐닝을 동시적으로 만드는 것이 더 용이하다.
- [0105] IRM 기능(202)의 일 양태는 어느 2차 객체가 1차 객체에 대응하는지를 인식하는 성능 기준 방식에 있다. 향상된 코드는 2차 객체에 대해 객체를 확인하는 시점을 인식해야 한다. 이것은 IS-PRIMARY 비트를 사용하고 확인함으로써 달성된다. 이러한 비트는 하나 이상의 2차 객체가 존재하고 있음을 나타내도록 설정되고; 주어진 1차 객체는 다수의 2차 객체를 가질 수 있다. 일 구현예에서, IS-PRIMARY 비트는 64-비트 시스템의 동기 블록(syncblock)에 내장된다. 이것은 32-비트 어드레스 공간에 대한 구현을 배제하지 않지만, 64-비트 시스템이 32-비트 시스템보다 훨씬 더 큰 힙을 가질 수 있기 때문에 64-비트 시스템 상에서 상대적인 성능 이득이 훨씬 더 클 수 있다.
- [0106] IRM 기능(202)을 갖는 향상된 코드는 또한 IS-PRIMARY 비트가 설정되어 있는 객체에 대한 2차 객체를 입수할 수 있어야 한다. 일부 실시예는 DH에 대한 1차 객체인 객체들에 대한 해시 테이블(602)을 포함한다. 불안정한 해시로 인한 문제를 방지하기 위해, 일부 실시예는 1차 객체에 대한 해시 코드를 해시 테이블 키로 사용한다. 이 테이블(602)은 DH 연산으로 구축되어야 하며, 예컨대, DH에 대해 1차 객체가 설정되는 경우, 그러한 1차 객체는 2차 객체를 값으로 하여(또는 이러한 1차 객체를 갖는 다수의 2차 객체가 존재하는 경우 값들 중 하나의 값으로 하여) 1차 해시 테이블에 삽입되어야 한다. DH의 1차 객체가 변경되면, 이전 1차 객체가 해시 테이블에서 제거되고 새로운 1차 객체가 삽입될 것이다. BGC가 또한 이러한 해시 테이블을 주시하고 있는 경우 매우 정밀한 동기화를 방지하기 위해, 일부 실시예에서는 BGC가 잠금을 취하고(STW), 그리고 잠금을 해제하기 전에 잠시 동안 작동하도록 할 수 있다.
- [0107] 이러한 방식으로 DH 스캐닝은 일반 표시로 폴딩될 수도 있다. 그러나, 일부 시나리오에서 한 가지 추가 고려 사항은 수정 사항이 재방문 코드에 의해 추적되도록 DH에 이러한 수정 사항을 레코딩하는 방법에 있다. 레거시 코드에서, 힙 수정 사항(heap modification)은 때때로 기입 감시 테이블(write watch table)에 수정 사항을 레코딩하는 기입 장벽(write barrier)을 통해 추적된다. DH 표시에 대한 유사한 효과를 달성하기 위해, 일부 실시예는 DH의 1차 객체에 대한 수정 사항을 레코딩하므로, 표시 코드는 그러한 수정된 2차 객체를 재방문하게 될 것이다. 핸들 테이블의 경우, 코드는 이미 핸들 테이블 기입 장벽(예컨대, `HandleWriteBarrier` 또는 다른 환경의 유사한 장벽)을 호출하여 수명을 조정할 수 있다. 향상된 코드는 이 기회를 이용하여, 기입 장벽 코드를 사용함으로써 기입 감시 테이블에 1차 객체를 레코딩할 수도 있다.
- [0108] 이와는 별개로, 일부 실시예는 Final Mark에서 별도의 DH 스캐닝을 생략한다. 1차 객체의 2차 객체에 대한 임의의 수정 사항이 있는 경우, 이는 핸들 테이블 기입 장벽을 통해 반영되어 1차 객체는 자연스럽게 재방문될 것이다. (DH의 1차 객체를 수정하거나 새로운 DH를 할당함으로써) 새로운 1차 객체가 추가되면, 가비지 컬렉션이

실행 엔진을 일시 중단하기 전에 그러한 1차 객체의 2차 객체가 구축되었을 것이다. STW에서, 가비지 콜렉터는 1차 객체에 대한 2차 객체의 관계에 액세스할 수 있으므로, 1차 객체가 표시되면, 가비지 콜렉터는 어느 객체가 또한 표시되어야 하는지를 결정할 수 있다.

[0109] 향상된 BGC의 한 플로우는 Init 초기화 단계(STW), 이에 후속되는 통합된 표시가 있는 동시적인 표시 단계(STW는 아님), 이에 후속되는 캐치업 및 짧고 약한 널링(NULLING)을 또한 수행하는 통합된 표시가 있는 Final Mark 최종 표시 단계(STW), 이에 후속되는 통합된 표시가 있는 동시적 최종화 스캐닝 단계(STW는 아님), 이에 후속되는 길고 약한 널링 단계(STW), 이에 후속되는 소거 단계를 포함한다. 이 향상된 BGC 워크플로우는 위에서 설명한 비향상된 BGC의 워크플로우보다 STW 시간이 더 짧은 경향이 있으므로, 애플리케이션 성능을 향상시킬 것이다.

[0110] 기술적 특성

[0111] 본원에 설명되는 실시예의 기술적 특성은 본 기술 분야의 통상의 기술자에게는 분명할 것이며, 또한 광범위한 사려 깊은 독자에게는 여러 방식으로 분명할 것이다. 일부 실시예는 컴퓨터 시스템에서 비트를 설정하거나 소거하는 것, 포인터를 추적하는 것, 메모리를 할당하는 것, 해시 테이블을 생성 또는 업데이트하는 것, 및 메모리를 관리하는 것과 같은 기술적 활동을 다루며, 이들 각각은 컴퓨팅 기술에 깊이 뿌리를 둔 활동이다. 논의된 기술적 메커니즘 중 일부는, 예컨대, 내장된 비트, 오타 정보 구조, 객체, 객체 참조물, 해시 테이블, 핸들, 및 코드의 예를 포함하고 있다. 논의된 기술적 효과 중 일부는, 예컨대, 비-1차 객체에 대한 감소된 저장 요구 사항, 가비지 콜렉션 동안 2차 객체 표시와 참조된 객체 표시의 통합, 증가된 동시성, 감소된 STW 시간, 및 가비지 콜렉션 동안의 특정 비효율성의 방식을 포함하고 있다. 따라서, 순수한 정신적 프로세스는 분명히 배제되어 있다. 교시 내용의 기술적 특성에 기반한 다른 이점도 또한 제공된 설명으로부터 기술자에게는 분명할 것이다.

[0112] 본원에서 설명되는 일부 실시예는 일부 사람에 의해서는 보다 넓은 맥락으로 보여질 수도 있다. 예를 들어, 어떤 것이 사용 중인지 여부, 사이즈, 수집, 관리, 및 관계와 같은 개념은 특정 실시예와 관련된 것으로 간주될 수 있다. 그러나, 본원에서 추상적인 아이디어에 대한 배타적 권리가 고려되고 있다는 것은 광범위한 맥락의 가용성을 따르는 것이 아니며, 그러한 권리가 고려되지도 않는다. 오히려, 본 개시 내용은 애플리케이션 소프트웨어와 가비지 콜렉션 동시성을 증가시키는 방법과 같은 특정 기술적 문제를 완전히 또는 부분적으로 해결하는 적절하게 특정된 실시예를 제공하는 데 초점을 맞추고 있다. 사용 상태, 사이즈, 수집, 관리, 또는 관계와 관련된 다른 구성된 저장 매체, 시스템, 및 프로세스는 본 범위를 벗어난 것이다. 따라서, 모호함, 단순한 추상성, 기술적 특성의 부족, 및 수반되는 증명 문제는 또한 본 개시 내용의 적절한 이해 하에서 방지된다.

[0113] 추가 조합 및 변형

[0114] 코드, 데이터 구조, 로직, 컴포넌트, 통신, 및/또는 이들의 기능적 등가물에 대한 이들의 조합 중 임의의 것은 또한 전술한 시스템 및 그 변형 중 임의의 것과 조합될 수 있다. 프로세스는 동작 가능한 임의의 서브 세트 또는 조합 또는 시퀀스로 본원에 설명된 임의의 단계를 포함할 수 있다. 각 변형은 단독으로 또는 임의의 하나 이상의 다른 변형과 함께 발생할 수 있다. 각 변형은 임의의 프로세스에서 발생할 수 있으며 각 프로세스는 임의의 하나 이상의 다른 프로세스와 조합될 수 있다. 변형을 포함하여 각 프로세스 또는 프로세스의 조합은 위에서 설명한 임의의 구성된 저장 매체 조합 및 변형과 조합될 수 있다.

[0115] 보다 일반적으로, 기술자는 본 개시 내용의 모든 부분 또는 그 임의의 특정 세부 사항이 실시 가능성, 상세한 설명 또는 최상의 모드와 같은 법적 기준을 충족시키기 위해 반드시 필요한 것은 아니라는 것을 인식할 것이다. 또한, 실시예는 본원에 설명된 특정 동기 부여 예, 프로그래밍 언어 또는 라이브러리 또는 런타임, 소프트웨어 프로세스, 개발 도구, 식별자, 데이터 구조, 데이터 구성, 표기법, 제어 플로우, 의사 코드, 명명 규칙, 또는 다른 구현 선택 사항으로 제한되지는 않는다. 심지어는 본 신규한 사항의 소유자로부터의 임의의 다른 특허 개시 내용과의 명백한 충돌은 본 특허 개시 내용에 제시되는 청구항을 해석하는 데 아무런 역할을 하지 못한다.

[0116] 두문자어, 약어, 이름, 및 심볼

[0117] 일부의 두문자어, 약어, 이름, 및 심볼이 아래에 정의되어 있다. 다른 것은 본원의 다른 곳에서 정의되거나, 또는 기술자에 의해 이해되기 위해 본원에서 정의를 요구하지는 않는다.

[0118] ALU: 산술 및 논리 유닛

[0119] API: 애플리케이션 프로그램 인터페이스

- [0120] BIOS: 기본 입력/출력 시스템
- [0121] CD: 콤팩트 디스크
- [0122] CPU: 중앙 처리 유닛
- [0123] DVD: 디지털 다기능 디스크 또는 디지털 비디오 디스크
- [0124] FPGA: 필드 프로그래머블 게이트 어레이
- [0125] FPU: 부동 소수점 처리 유닛
- [0126] GPU: 그래픽 처리 유닛
- [0127] GUI: 그래픽 사용자 인터페이스
- [0128] IaaS 또는 IAAS: 서비스로서의 인프라
- [0129] IoT: 사물 인터넷
- [0130] LAN: 근거리 네트워크
- [0131] OS: 운영 체제
- [0132] PaaS 또는 PAAS: 서비스로서의 플랫폼
- [0133] RAM: 랜덤 액세스 메모리
- [0134] ROM: 판독 전용 메모리
- [0135] TPU: 텐서 처리 유닛
- [0136] UEFI: 확장 가능한 통합 펌웨어 인터페이스
- [0137] WAN: 광역 네트워크
- [0138] **일부 추가 용어**
- [0139] 본원에서는 도면에 도시된 것과 같은 예시적인 실시예가 참조되고, 이를 설명하기 위해 본원에서는 특정 언어가 사용된다. 그러나, 본원에 도시된 특징의 변경 및 추가 수정, 그리고 관련 기술(들)의 기술자에게 발생하고 본 개시 내용을 소유하는 본원의 특정 실시예에 의해 예시된 추상적 원리에 대한 추가적인 기술적 적용은 청구항의 범위 내에서 고려되어야 한다.
- [0140] 용어의 의미는 본 개시 내용에서 명확해지므로, 청구항은 이러한 명확한 설명에 주의하면서 관독되어야 한다. 특정 예가 제공되지만, 관련 기술(들)의 기술자는 다른 예가 또한 사용된 용어의 의미 및 하나 이상의 청구항의 범위 내에 속할 수 있다는 것을 이해할 것이다. 본원에서의 용어는 일반적인 사용(특히 비 기술적 사용)에서, 또는 특정 산업의 사용에서, 또는 특정 사전 또는 사전 세트에서 갖는 것과 전적으로 동일한 의미를 가질 필요는 없다. 폭넓은 용어를 나타내는 데 도움이 되도록 다양한 구문과 함께 참조 번호가 사용될 수 있다. 주어진 본문 부분에서 참조 번호가 누락되었다고해서 반드시 도면의 내용이 본문에서 논의되고 있지 않다는 것을 의미하는 것은 아니다. 발명자는 구체적인 선택된 사전 편찬에 대한 권리를 주장하고 행사한다. 인용된 용어는 명시적으로 정의되지만, 용어가 따옴표 표시를 사용하지 않고도 묵시적으로 정의될 수 있다. 본원의 용어는 상세한 설명 및/또는 적용 파일의 다른 곳에서 명시적으로 또는 묵시적으로 정의될 수 있다.
- [0141] 본원에서 사용되는 바와 같이, "컴퓨터 시스템"(일명, "컴퓨팅 시스템")은, 예를 들어, 하나 이상의 서버, 마더보드, 처리 노드, 랩톱, 태블릿, 퍼스널 컴퓨터(휴대용 또는 비휴대용), 퍼스널 디지털 보조 장치, 스마트폰, 스마트워치, 스마트밴드, 셀룰러폰 또는 모바일폰, 적어도 하나의 프로세서 및 메모리를 갖는 다른 모바일 디바이스, 비디오 게임 시스템, 증강 현실 시스템, 홀로그래픽 프로젝션 시스템, 텔레비전, 웨어러블 컴퓨팅 시스템, 및/또는 적어도 부분적으로 명령어에 의해 제어되는 하나 이상의 프로세서를 제공하는 다른 디바이스(들)를 포함할 수 있다. 명령어는 메모리 및/또는 특수 회로부에 있는 펌웨어 또는 다른 소프트웨어의 형태일 수 있다.
- [0142] "다중 스레드" 컴퓨터 시스템은 다수의 실행 스레드를 지원하는 컴퓨터 시스템이다. "스레드"라는 용어는 아마도 동기화로 스케줄링할 수 있거나 스케줄링되는 코드를 포함하는 것으로 이해되어야 한다. 스레드는 본 개시

내용 외부에서, 예를 들어, "태스크", "프로세스" 또는 "코루틴"과 같은 다른 이름으로 알려질 수도 있다. 그러나, 본원에서는 스레드가 프로세스 내부의 실행 경로를 정의한다는 점에서 스레드와 프로세스를 구별하고 있다. 또한, 프로세스의 스레드들은 주어진 어드레스 공간을 공유하지만, 서로 다른 프로세스는 서로 다른 제각기의 어드레스 공간을 갖는다. 프로세스의 스레드는 병렬로, 순차적으로, 또는 병렬 실행과 순차 실행(예컨대, 시간 분할)의 조합으로 실행될 수 있다.

[0143] "프로세서"는 동시적 다중 스레딩 구현예(simultaneous multithreading implementation)에서의 코어와 같은 스레드 처리 유닛이다. 프로세서는 하드웨어를 포함한다. 주어진 칩은 하나 이상의 프로세서를 보유할 수 있다. 프로세서는 범용일 수 있거나, 또는 벡터 처리, 그래픽 처리, 신호 처리, 부동 소수점 산술 처리, 암호화, I/O 처리, 머신 학습 등과 같은 특정 용도를 위해 조정될 수 있다.

[0144] "커널"은 운영 체제, 하이퍼바이저, 가상 머신, BIOS 또는 UEFI 코드, 및 유사한 하드웨어 인터페이스 소프트웨어를 포함한다.

[0145] "코드"는 프로세서 명령어, 데이터(상수, 변수, 및 데이터 구조를 포함함), 또는 명령어 및 데이터 모두를 의미한다. "코드" 및 "소프트웨어"는 본원에서 상호 교환 가능하게 사용된다. 실행 가능한 코드, 해석된 코드, 및 펌웨어는 코드의 몇 가지 예이다.

[0146] 본원에서 "프로그램"은 애플리케이션, 커널, 드라이버, 인터럽트 핸들러, 펌웨어, 상태 머신, 라이브러리, 및 프로그래머(또한 개발자로 지칭되기도 함)가 작성하고/하거나 자동으로 생성되는 다른 코드를 포함하도록 광범위하게 사용된다.

[0147] "루틴"은 일반적으로, 루틴이 호출된 프로그램 실행 지점 직후에 명령어에 대한 제어권을 리턴하는 호출 가능한 코드 조각이다. 사용된 용어에 따라, 다른 곳에서 때때로 "함수"와 "프로시저"가 구별되고; 즉, 함수는 일반적으로 값을 리턴하지만 프로시저는 그렇지 않다. 본원에서 사용되는 "루틴"은 함수 및 프로시저 모두를 포함한다. 루틴은 값(예컨대, $\sin(x)$)을 리턴하는 코드를 가질 수 있거나 또는 값(예컨대, void 함수)을 제공하지도 않으면서 간단히 리턴할 수 있다.

[0148] "참조된 객체" 대 "2차 객체"에 대해, 기술자는, 객체에 대한 일반 참조물이, 예컨대, 포인터 필드로서, 소스 코드에서 명시적으로 선언될 수 있고, 종속 핸들도 소스 코드에서 명시적으로 선언될 수 있음을 인정할 것이다. 그러나, 어떻게든 관련된 객체의 이러한 식별은, 일부 실시예에서, 그 용도 및 저장 요구 사항에서 상이하다. 다른 객체를 참조하는 객체 A의 고정 참조물(regular reference)은 객체 A에 참조물 사이즈의 필드를 할당하는 것을 포함하며, 이는 일부 다른 객체 B를 반드시 항상 식별하는 것은 아니다. 코드는 A가 일부 다른 객체 B를 절대로 참조하지 않는 경우에도, 객체 A에 있는 해당 필드의 저장 비용을 항상 발생시킨다. 대조적으로, 종속 핸들은 객체 A와 다른 객체 B 사이의 관계를 수립한다(즉, A가 활성 상태라면, B가 활성 상태라고 간주한다). 객체 A에 대한 종속 핸들은, 특정 객체 A와 특정 객체 B 사이에 실제 관계가 있을 경우, 참조물 사이즈의 필드만을 사용하여 객체 B를 식별한다.

[0149] 본원에서 "참조물", "어드레스", "포인터", 및 "핸들"이라는 용어는 스토리지에서 적어도 참조물 사이즈인 객체 식별자를 논의할 때 사용되고, 예컨대, N-비트 가상 어드레스 공간에서 적어도 N개의 비트가 된다. 더 중요한 것은 각각이 IS-PRIMARY 비트보다 크다는 것이다.

[0150] 객체와 관련하여 본원에서 사용되는 "포함"은 "내장"보다 더 광범위하게 사용된다. "포함"은 참조물, 종속 핸들, 또는 연속 할당을 통한 포함을 포함하지만, "내장"은 연속 할당을 필요로 한다. 따라서, 객체 A가 객체 B에 대한 포인터를 가지고 있는 경우, 객체 A는 객체 B를 포함하고, 포인터는 객체 A에 내장되지만 객체 B 자체는 객체 A에 내장되지 않는다. 이러한 구별은 (a) 종속 객체가 1차 객체로 사용되는 경우 모든 객체에 참조물 사이즈의 필드를 내장시킴으로써 종속 객체를 지원하는 것과, (b) 단일 비트(IS-PRIMARY 비트)만을 내장하고 실제로 1차 객체인 객체에 대해서만 (해시 테이블 내의) 참조물 사이즈의 필드를 사용함으로써 종속 객체를 지원하는 것 사이의 차이를 존중하고 있다.

[0151] "클라우드"는 측정된 주문형 서비스에 탄력적으로 이용 가능한 컴퓨팅, 저장, 및 네트워킹을 위한 풀링된 리소스를 의미한다. 클라우드는 사설, 공용, 커뮤니티, 또는 하이브리드일 수 있으며, 클라우드 서비스는 서비스로서의 인프라(IaaS), 서비스로서의 플랫폼(PaaS), 서비스로서의 소프트웨어(SaaS), 또는 다른 서비스의 형태로 제공될 수 있다. 달리 언급되지 않는 한, 파일로부터의 판독 또는 파일로의 기입에 대한 임의의 논의는 로컬 파일의 판독/기입, 또는 클라우드 네트워크 또는 다른 네트워크일 수 있는 네트워크를 통한 판독/기입, 또는 둘 모두(로컬 및 네트워크형 판독/기입)를 수행하는 것을 포함한다.

- [0152] "IoT" 또는 "사물 인터넷"은 어드레스 가능한 내장된 컴퓨팅 노드의 임의의 네트워크형 집합체를 의미한다. 이러한 노드는 본원에 정의된 컴퓨터 시스템의 예이지만, 다음의 특성: (a) 사람이 관독 가능한 로컬 디스플레이가 존재하지 않음; (b) 로컬 키보드가 존재하지 않음; (c) 1차 입력 소스는 비언어적 데이터 소스를 추적하는 센서임; (d) 로컬 회전 디스크 스토리지 - RAM 칩 또는 ROM 칩이 유일한 로컬 메모리를 제공함 -가 존재하지 않음; (e) CD 또는 DVD 드라이브가 존재하지 않음; (f) 가전 제품 또는 가구 설비에 내장된 것; (g) 이식 또는 착용 가능한 의료 디바이스에 내장된 것; (h) 차량에 내장된 것; (i) 프로세스 자동화 제어 시스템에 내장된 것; 또는 (j) 환경 모니터링, 도시 인프라 모니터링, 산업 장비 모니터링, 에너지 사용 모니터링, 인간 또는 동물 건강 모니터링, 물리적 보안, 또는 물리적 운송 시스템 모니터링 중 하나에 초점을 맞춘 설계물 중 적어도 2개를 또한 갖는다. IoT 스토리지는 클라우드를 통한, 다른 네트워크를 통한, 또는 직접 로컬 액세스 시도를 통해 무단 액세스의 타겟일 수 있다.
- [0153] 계산 리소스에 대한 "액세스"는, 리소스를 관독하거나, 수정하거나, 기입하고, 실행하거나, 또는 활용하기 위한 권한 또는 다른 능력의 사용을 포함한다. 시도된 액세스는 실제 액세스와는 명시적으로 구별될 수 있지만, 유사격자에 의해 "시도된" 것이 아닌 "액세스"는 시도된 액세스와 실제로 수행되거나 제공된 액세스를 모두 포함한다.
- [0154] 본원에서 사용되는 바와 같이, "포함하는"은 달리 언급되지 않는 한 추가 요소를 허용한다(즉, 포함은 구비를 의미한다).
- [0155] "최적화"는 반드시 완벽하지는 않지만 개선하는 것을 의미한다. 예를 들어, 최적화된 프로그램이나 알고리즘에서 추가적인 개선을 수행하는 것이 가능할 수 있다.
- [0156] 본원에서의 "프로세스"는 때때로 컴퓨팅 과학 기술의 용어로서 사용되며, 그러한 기술적 의미에서, 예를 들어, 코루틴, 스레드, 태스크, 인터럽트 핸들러, 애플리케이션 프로세스, 커널 프로세스, 프로시저, 또는 객체 메소드를 포함할 수도 있거나 이로 지칭될 수도 있는, 계산 리소스 사용자를 포괄한다. 실제로, "프로세스"는 (각각 Microsoft Corporation, Linus Torvalds의 표시인) Windows® 작업 관리자, Linux® ps, 또는 다른 운영 체제 환경에서의 유사한 유틸리티와 같은 시스템 유틸리티에 의해 식별되는 계산 엔티티이다. 본원에서의 "프로세스"는 또한, 예컨대, 시스템 청구항 또는 제품(구성된 저장 매체) 청구항과 대조되는 프로세스 청구항을 설명할 때, 특허법의 기술 용어로 사용된다. 유사하게, 본원에서의 "메소드"는 때때로 컴퓨팅 과학 기술의 기술 용어(일종의 "루틴")로서 사용되며, 또한 "방법"은 특허법의 기술 용어("프로세스")로도 사용된다. 특허법의 의미에서 "프로세스" 및 "방법"은 여기에서 상호 교환적으로 사용된다. 기술자는 특정 사례에서 어떤 의미가 의도되는지를 이해할 것이고, 주어진 청구항의 프로세스 또는 방법(특허법 의미에서)은 때때로 하나 이상의 프로세스 또는 메소드(컴퓨팅 과학의 의미에서)를 사용하여 구현될 수 있음을 또한 이해할 것이다.
- [0157] "자동으로"는 자동화가 없는 것과 대조되는 것으로, 자동화(예컨대, 본원에서 논의되는 특정 동작 및 기술적 효과를 위해 소프트웨어에 의해 구성된 범용 컴퓨팅 하드웨어)를 사용하는 것을 의미한다. 특히, "자동으로" 수행되는 단계는 인간에 의해 시작되거나 인간에 의해 대화식으로 안내될 수 있지만, 종이 상의 손으로 또는 사람의 마음으로 수행되지는 않는다. 따라서 기술적 상호 작용이 제공되지 않고서는 실현될 수 없는 하나 이상의 기술적 효과를 얻기 위해 자동 단계가 머신으로 수행된다. 자동으로 수행되는 단계는 예방 조치로 수행된 적어도 하나의 작업을 포함하는 것으로 추정된다.
- [0158] 기술자는 기술적 효과라는 것이 기술적 실시예의 추정 목적임을 이해하고 있다. 예를 들어, 계산이 실시예에 포함되고 일부 계산이 기술적 컴포넌트 없이 (예컨대, 종이와 연필로 또는 심지어는 정신적 단계로) 수행될 수도 있다는 단순한 사실만으로는 기술적 효과의 존재를 제거하지 못하거나 실시예의 구체적인 기술적 특성을 변경하지 못한다. 객체의 표시, 참조된 객체의 식별, 2차 객체의 식별, 및 본원에서 논의되는 다른 많은 동작과 같은 메모리 관리 동작은 본질적으로 디지털인 것으로 이해된다. 인간의 마음은 CPU 또는 다른 프로세서, 또는 RAM 또는 다른 디지털 스토리지와 직접 인터페이스하여 본원에 교시되는 메모리 관리 단계를 수행하기 위해 필요한 데이터를 관독하고 기입할 수 없다. 이것은 모두 본 개시 내용을 고려하여 본 기술 분야의 기술자에 의해 잘 이해될 것이다.
- [0159] "계산적으로"는 마찬가지로 컴퓨팅 디바이스(적어도 프로세서와 메모리)가 사용되는 것을 의미하며, 단순한 인간의 생각 또는 단순한 인간 행동만으로 결과를 얻는 것을 배제한다. 예를 들어, 종이와 연필로 산술을 수행하는 것은 본원에서 이해되는 것과 같이 계산적으로 산술을 수행하는 것이 아니다. 계산 결과는 더 빠르고, 더 광범위하고, 더 깊고, 더 정확하고, 더 일관되고, 더 포괄적이고/이거나, 인간 단독의 수행 범위를 넘어서는 기술적 효과를 제공한다. "계산 단계"는 계산적으로 수행되는 단계이다. "자동으로" 또는 "계산적으로"는 반드시 "

즉시"를 의미하지는 않는다. "계산적으로" 및 "자동으로"는 본원에서 상호 교환가능하게 사용된다.

- [0160] "예방 조치로"는 사용자의 직접적인 요청이 없음을 의미한다. 실제로, 사용자는 단계의 결과가 사용자에게 제시 될 때까지는 실시예에 의한 예방 조치적 단계가 가능했다는 것을 심지어는 깨닫지 못할 수도 있다. 달리 언급된 경우를 제외하고, 본원에 설명되는 모든 계산 및/또는 자동 단계는 또한 예방 조치적으로 수행될 수도 있다.
- [0161] 본 문서 전반에 걸쳐, 선택적인 복수의 "(들)"의 사용은 표시된 특징이 하나 이상 존재한다는 것을 의미한다. 예를 들어, "프로세서(들)"는 "하나 이상의 프로세서" 또는 동등하게 "적어도 하나의 프로세서"를 의미한다.
- [0162] 미국 법률 및 관행의 목적을 위해, 청구항 또는 다른 곳에서 본원의 "단계"라는 단어를 사용하는 것은 수단 플러스 기능(means-plus-function), 단계 플러스 기능(step-plus-function), 또는 35 United State Code 섹션 112 제6 단락/섹션 112(f)의 청구항 해석을 호출하기 위한 것이 아니다. 그 효과에 대한 모든 추정은 이로써 명시적으로 반박된다.
- [0163] 미국 법률 및 관행의 목적을 위해, 청구항은 "수단"이라는 문구를 사용하지 않는 한 수단 플러스 기능 해석을 호출하기 위한 것은 아니다. 수단 플러스 기능 언어로 해석되도록 의도된 청구항 언어는 "수단"이라는 문구를 사용하여 그 의도를 명시적으로 언급할 것이다. 수단 플러스 기능 해석이 적용되는 경우, "수단"의 사용 및/또는 법원의 청구항 언어의 법적 구성에 관계없이, 주어진 명사 또는 주어진 동사에 대해 본 명세서에서 언급된 수단은 청구항 언어로 연결되고, 그리고 다음: 즉, 도면의 블록 다이어그램에서 동일한 블록 내에 출현한 것, 동일하거나 유사한 이름으로 표시된 것, 동일한 참조 번호로 표시된 것, 임의의 도면에 도시된 기능적 관계, 본 개시 내용의 본문에 언급된 기능적 관계 중 임의의 것에 의해 본원에서 함께 연결된 것으로 이해되어야 한다. 예를 들어, 청구항 한정 사항이 "zac 위젯"을 인용하고 해당 청구항 한정 사항이 수단 플러스 기능 해석의 대상이 된 경우, 최소한 "zac 위젯"을 언급하는 모든 도면 블록, 단락, 또는 예 내의 명세서의 모든 곳에서 식별되거나, 또는 zac 위젯에 할당된 임의의 참조 번호로 함께 묶이거나, 또는 zac 위젯의 구조 또는 동작과 기능적 관계가 있는 것으로 개시되는 모든 구조는 zac 위젯의 적용에서 식별된 구조의 일부로 간주될 것이고, zac 위젯 구조에 대한 균등물 세트를 정의하는 데 도움을 줄 것이다.
- [0164] 기술자는, 이러한 혁신적인 개시 내용이 다양한 데이터 값 및 데이터 구조를 논의하고, 그러한 항목이 메모리(RAM, 디스크 등)에 상주하여 메모리를 구성한다는 것을 인식할 것이다. 기술자는 또한, 이 혁신적인 개시 내용이 주어진 구현예에서 실행 가능 코드로 구현되어야 하는 다양한 알고리즘 단계를 논의하고, 그러한 코드가 또한 메모리에 상주하며, 이러한 코드가 이러한 코드를 실행하는 임의의 범용 프로세서를 효과적으로 구성함으로써 이러한 프로세서를 범용 프로세서에서 기능적으로 특수 목적 하드웨어인 특수 목적 프로세서로 변환하는 것을 인식할 것이다.
- [0165] 따라서, 기술자는 (a) 청구항에서 열거되는 메모리, 및 (b) 청구항에서 열거되는 데이터 구조 또는 데이터 값 또는 코드를 비중첩 항목으로 취급하는 실수를 범하지 않을 것이다. 데이터 구조, 데이터 값 및 코드는, 심지어는 청구항이 언급된 각각의 모든 데이터 구조, 데이터 값 또는 코드 조각에 대한 해당 상주성(residency)을 명시적으로 언급하지 않는 경우에도, 메모리에 상주하는 것으로 이해된다. 따라서, 그러한 상주성에 대한 명시적인 설명은 불필요하다. 그러나, 그러한 것이 또한 금지되지는 않으며, 다른 모든 데이터 값과 데이터 구조 및 코드를 상주성에서 배제하지 않고, 강조를 위한 하나 또는 두 개의 선택적 설명이 있을 수 있다. 마찬가지로, 청구항에 열거되는 코드 기능은 해당 구성 품질이 청구항에 명시적으로 열거되어 있는지 여부에 관계없이 프로세서를 구성하는 것으로 이해된다.
- [0166] 본 문서 전체에 걸쳐, 달리 명시적으로 언급되지 않는 한, 프로세스 내의 단계에 대한 임의의 언급은, 해당 단계가 이해 당사자에 의해 직접 수행될 수 있고/있거나 개입 메커니즘 및/또는 개입 엔티티를 통해 해당 당사자에 의해 간접적으로 수행될 수 있고, 여전히 단계의 범위 내에 있는 것으로 간주한다. 즉, 직접 수행이 명시적으로 언급된 요구 사항이 아니라면, 이해 당사자에 의한 단계의 직접 수행이 요구될 필요는 없다. 예를 들어, 목적지 또는 다른 대상과 관련하여 할당하는 것, 확인하는 것, 소거하는 것, 내장하는 것, 열거하는 것, 실행하는 것, 추적하는 것, 나타내는 것, 루핑하는 것, 관리하는 것, 표시하는 것, 측정하는 것, 수행하는 것, 제공하는 것, 설정하는 것, 지정하는 것, 대체하는 것, 취급하는 것, 업데이트하는 것, 또는 사용하는 것(및 할당하는, 할당되는, 확인하는, 확인되는 등)과 같은 이해 당사자에 의한 동작을 포함하는 단계는, 본 문서에서 열거되지만, 여전히 이해 당사자에 의해 직접 수행되는 것으로 이해될 수 있는 임의의 동작을 포함하여, 일부의 다른 당사자에 의한, 전술한 또는 포워딩하는 것, 복사하는 것, 업로드하는 것, 다운로드하는 것, 인코딩하는 것, 디코딩하는 것, 압축하는 것, 압축 해제하는 것, 암호화하는 것, 암호 해독하는 것, 인증하는 것, 호출하는 것 등과 같은 개입 동작을 포함할 수 있다.

- [0167] 데이터 또는 명령어를 참조할 때마다, 이러한 항목은 컴퓨터 판독 가능 메모리 및/또는 컴퓨터 판독 가능 저장 매체를 구성하여, 특정 제조품으로 변환하는 것으로, 이는, 예를 들어, 단순히 종이에 존재하는 것, 사람의 마음 속에 존재하는 것, 또는 단순한 신호로서 와이어를 통해 전파되는 것과는 대조되는 것으로 이해된다. 미국에서 특허 보호의 목적 상, 메모리 또는 다른 컴퓨터 판독 가능 저장 매체는 In re Nuijten 사건의 미국 특허 상 표청(USPTO) 해석 하에서 특허 가능한 대상의 범위를 벗어난 전파 신호나 반송파 또는 단순한 에너지가 아니다. 미국에서 어떠한 청구항도 신호 자체 또는 단순한 에너지를 포함하지는 않으며, 본 개시 내용의 관점에서 달리 주장하는 청구항 해석은 그 자체로 불합리한 것이다. 미국 이외의 지역에서 특허 허여된 청구항에서 달리 명시되지 않는 한 청구항은 신호 자체 또는 단순한 에너지를 포함하지 않는다.
- [0168] 또한, 본원의 다른 곳에서 명백하게 반대되는 내용이 있음에도 불구하고, (a) 한편의 컴퓨터 판독 가능 저장 매체 및 컴퓨터 판독 가능 메모리와, (b) 다른 한편의 전송 매체(또한 신호 매체로 지칭되기도 함) 간의 명확한 차이가 이해되어야 한다. 전송 매체는 전파 신호 또는 반송파 컴퓨터 판독 가능 매체이다. 이와는 대조적으로, 컴퓨터 판독 가능 저장 매체 및 컴퓨터 판독 가능 메모리는 전파 신호 또는 반송파 컴퓨터 판독 가능 매체가 아니다. 청구항에서 달리 명시적으로 언급되지 않는 한, "컴퓨터 판독 가능 매체"는 전파 신호 자체도 아니고 단순히 에너지도 아닌, 컴퓨터 판독 가능 저장 매체를 의미한다.
- [0169] 본원의 "실시예"는 예이다. 용어 "실시예"는 "본 발명"과 상호 교환가능하지는 않다. 실시예는 (결과가 이용 가능하다면) 다른 실시예를 생성하기 위해 양태를 자유롭게 공유하거나 차용할 수 있으며, 비록 양태의 결과적인 조합이 본원에서 그 자체로 명시적으로 설명되지 않더라도 그러할 수 있다. 각각의 그리고 모든 허용된 조합이 명시적이고 개별적으로 설명되도록 요구하는 것은 본 기술 분야의 기술자에게는 불필요하며, 특허 명세서가 해당 분야의 기술자인 독자를 위해 작성된다는 것을 인식하는 정책에 반하는 것이다. 심지어는 적은 수의 조합 가능한 특징으로부터 발생하는 가능한 조합의 수에 관한 공식적인 조합 계산 및 비공식적인 공통 직관은 또한, 본원에 설명된 양태에 대해 다수의 양태 조합이 존재한다는 것을 나타낼 것이다. 따라서, 각각의 모든 조합에 대한 명시적인 인용을 요구하는 것은 특허 명세서가 간결하고 독자가 관련 기술 분야에 대해 잘 알고 있어야 한다는 정책에 반하는 것이다.
- [0170] **참조 번호의 리스트**
- [0171] 다음의 리스트는 편의를 위해 그리고 도면을 지원하기 위해 그리고 다수의 항목을 참조하여 신규한 사항을 설명하는 명세서 본문의 일부로서 제공된다. 그럼에도 불구하고 여기에 나열되지 않은 항목도 주어진 실시예의 일부일 수 있다. 본문의 보다 나은 가독성을 위해, 주어진 참조 번호는 본문에서 참조된 항목의 전체가 아닌 일부의 열거를 옆에 열거된다. 주어진 항목의 다른 예 또는 다른 인스턴스를 참조하여 동일한 참조 번호가 사용될 수 있다. 참조 번호의 리스트는 다음과 같다:
- [0172] 100: 운영 환경으로서, 이는 또한 컴퓨팅 환경이라고도 지칭되기도 한다.
- [0173] 102: 컴퓨터 시스템으로서, 이는 또한 계산 시스템 또는 컴퓨팅 시스템이라고도 지칭되기도 한다.
- [0174] 104: 사용자로서, 예컨대, 소프트웨어 개발자이다.
- [0175] 106: 주변 장치이다.
- [0176] 108: 네트워크로서, 이는 일반적으로, 예컨대, LAN, WAN, 소프트웨어 정의 네트워크, 클라우드, 및 다른 유선 또는 무선 네트워크를 포함한다.
- [0177] 110: 프로세서이다.
- [0178] 112: 컴퓨터 판독 가능 저장 매체로서, 예컨대, RAM, 하드 디스크이다.
- [0179] 114: 착탈식 구성의 컴퓨터 판독 가능 저장 매체이다.
- [0180] 116: 프로세서로 실행 가능한 명령어이며; 착탈식 저장 매체 상에 존재하거나 또는 다른 메모리(휘발성 또는 비휘발성 또는 둘 모두) 내에 존재할 수 있다.
- [0181] 118: 데이터이다.
- [0182] 120: 커널(들)로서, 예컨대, 운영 체제(들), BIOS, UEFI, 디바이스 드라이버이다.
- [0183] 122: 도구로서, 예컨대, 안티바이러스 소프트웨어, 방화벽, 패킷 스니퍼 소프트웨어, 침입 탐지 시스템, 침입 방지 시스템, 다른 사이버 보안 도구, 디버거, 프로파일러, 컴파일러, 인터프리터, 디컴파일러, 어셈블러, 디스

어셈블러, 소스 코드 편집기, 자동 완성 소프트웨어, 시뮬레이터, 퍼저, 저장소 액세스 도구, 버전 제어 도구, 최적화기, 협업 도구, 다른 소프트웨어 개발 도구 및 도구 세트(예컨대, 통합 개발 환경을 포함), 하드웨어 개발 도구 및 도구 세트, 진단 루틴 등이다.

- [0184] 124: 애플리케이션으로서, 예컨대, 워드 프로세서, 웹 브라우저, 스프레드 시트, 게임, 이메일 도구, 커맨드이다.
- [0185] 126: 디스플레이 스크린으로서, 이는 또한 "디스플레이"라고 지칭되기도 한다.
- [0186] 128: 참조 번호 106, 108, 110, 112, 114와 달리 연관되지 않은 컴퓨팅 하드웨어이다.
- [0187] 200: 기능(202)의 일부 또는 전부를 포함하여, 가비지 콜렉션 메모리 관리 기능에 의해 향상된 시스템이다.
- [0188] 202: 본원에 기술된 임의의 형태의 가비지 콜렉션 메모리 관리 기능이다.
- [0189] 204: 가비지 콜렉터로서, 이는 "GC" 또는 "가비지 콜렉터 코드"라고 지칭되기도 한다.
- [0190] 206: GC 스레드로서, 즉 적어도 가비지 콜렉터 코드에 주로 전용되는 (명령어의 50%를 초과하는) 스레드이다.
- [0191] 208: 뮤테이터 스레드로서, 즉, 적어도 가비지 콜렉터 코드 이외의 코드에 주로 전용되는 (명령어의 50%를 초과하는) 스레드이며; 뮤테이터 스레드는 종종 사용자 스레드이지만 또한 커널 스레드일 수도 있으며, 용어의 "뮤테이터"는 뮤테이터 스레드가 가비지 콜렉션 중에 사용되는 어드레스를 변경할 수 있다는 것을 나타낸다.
- [0192] 210: 가비지 콜렉터 표시 단계 코드이다.
- [0193] 212: 객체로서, 예컨대, 객체 지향 프로그래밍 패러다임을 구현하는 프로그램에 할당된 객체 데이터 구조이며; 이는, 예컨대, 클래스, 구조체, 공용체, 멤버, 필드, 생성자, 소멸자를 사용하여, 구현될 수 있다.
- [0194] 214: 메모리의 힙 부분으로서, 이는 종종 간단히 "힙"이라고 지칭되기도 한다.
- [0195] 216: 1차 객체로서, 즉 적어도 하나의 2차 객체를 갖는 객체이다.
- [0196] 218: 2차 객체로서, 즉 그의 1차 객체가 활성 상태인 경우 활성 상태로 표시되어야 하는 객체이며; 2차 객체는 그의 1차 객체의 참조물 필드에 어드레스를 내장하고 있지 않은 다른 메커니즘에 의해 식별되고, 예컨대, 2차 객체는 1차-2차 객체 관계의 해시 테이블에서 식별될 수 있다.
- [0197] 220: 비-1차 객체로서, 즉 2차 객체를 갖고 있지 않고, 임의의 참조된 객체를 가질 수도 있거나 갖지 않을 수도 있는 객체이다.
- [0198] 222: 활성 객체로서, 즉 가비지 콜렉터에 의해 활성 상태인 것으로(아직 사용 중인 것으로) 표시된 객체이다.
- [0199] 224: 객체 참조물로서, 즉 다른 객체를 식별하는 객체에 내장된 값이며; 참조물은 또한 "포인터" 또는 "객체 어드레스" 또는 "링크" 또는 "핸들"이라고 지칭될 수 있고; 224는 또한 참조물 필드를 나타내며; 참조물 필드는 일반적으로 참조된 객체를 포함하는 임의의 어드레스 공간 내에 있는 어드레스만큼의 비트를 가지며, 예컨대, 32-비트 어드레스 공간에서의 최소 32개의 비트 및 64-비트 어드레스 공간에서의 최소 64개의 비트를 갖지만, 실제 참조물 자체는 다소 작을 수 있으며, 예컨대, 16개의 비트 또는 24개의 비트 또는 48개의 비트일 수 있다.
- [0200] 226: 플로팅 가비지로서, 예컨대, 사용 중인 것이 아니지만 재확보 용으로 아직 표시되지 않은 메모리이다.
- [0201] 228: 핸들 테이블이다.
- [0202] 230: 도달 가능한 메모리로서, 즉 할당된 메모리 부분을 연결하는 힙 루트 또는 다른 데이터 구조를 통해 도달 가능한 메모리이다.
- [0203] 232: 도달 불가능한 메모리로서, 즉 도달할 수 없는 메모리이다.
- [0204] 234: 표시된 메모리이다.
- [0205] 236: 표시되지 않은 메모리이다.
- [0206] 238: 사용 중인 메모리이다.
- [0207] 240: 사용 중이지 않은 메모리이다.
- [0208] 242: 실행 동안의 코드 경로 중첩이며; 하나 이상의 중첩 메트릭(244)을 사용하여 측정될 수 있으며; 숫자 242

는 또한, 예컨대, 실행 중첩을 생성하는 프로그램 설계 및 구성을 통해, 둘 이상의 코드 경로를 중첩하는 동작을 나타낸다.

- [0209] 244: 코드 중첩 메트릭으로서, 예컨대, 2개(또는 그 이상) 코드 경로가 각각의 코드 라인의 수, 컴파일러에 의해 생성된 명령어의 수, 호출된 루틴의 수, 또는 스택 프레임의 수와 관련하여 중첩되는 범위이다.
- [0210] 246: 실행 동안 코드를 통해 취해진 경로이며; 특히, 246-R0는 참조된 객체 표시 코드 경로를 나타내고, 246-S0는 2차 객체 표시 코드 경로를 나타낸다.
- [0211] 300: 메모리 세그먼트로서, 예컨대, 가비지 콜렉션에 의해 적어도 부분적으로 관리되는 영역, 페이지, 블록, 또는 다른 메모리 구역이다.
- [0212] 302: 할당된 메모리의 연속적인 구역으로서, 이는 사용 중이거나 재확보 불가능한 것(도 3에서 $\sqrt{}$ 로 표시됨)으로 알려지거나 또는 메모리의 연속적인 구역을 할당한 프로그램에 의해 더 이상 사용 중이지 않는 것으로 알려지므로 재확보 가능한 것(도 3에서 X로 표시됨)으로 알려진 것이다.
- [0213] 304: 압축으로 발생하는 재확보 가능한 메모리의 연속적인 구역이다.
- [0214] 306: 압축 단계로서, 예컨대, 하나 이상의 재확보 가능한 구역을 재배치하여 더 큰 연속적인 재확보 가능한 구역을 생성하는 단계이다.
- [0215] 308: 표시 단계이며; 이는 메모리 구역을 (i) 사용 중이거나 재확보 불가능한 것으로 알려지거나 (ii) 재확보 가능한 것으로 알려진 것으로 표시하기 위해 가비지 콜렉션 중에 수행되는 동작이다.
- [0216] 400: 일반적으로 스레드이다.
- [0217] 402: 가비지 콜렉션의 단계 및 서브 단계이다.
- [0218] 404: 가비지 콜렉션의 소거 단계이다.
- [0219] 406: 가비지 콜렉션의 전체 멈춤(stop-the-world) 서브 단계이며; 이는 또한 "STW"라고 지칭되기도 한다.
- [0220] 502: 참조된 객체이다.
- [0221] 504: 객체에 대한 포인터를 보유하도록 선언되거나 사용된 변수이다.
- [0222] 506: 객체에 전용된 저장 공간에 내장된 포맷 정보이다.
- [0223] 602: 1차-2차 객체 관계를 나타내는 해시 테이블 또는 다른 데이터 구조이며; 602는 또한 1차-2차 객체 관계를 나타낸다.
- [0224] 604: 데이터 구조(602) 내의 항목이다.
- [0225] 606: 종속 핸들로서, 즉 객체들 간의 1차-2차 객체 관계를 식별하는 핸들이며; 종속 핸들은 1차 객체 및 1차 객체와 연관된 2차 객체를 가지며, 종속 핸들을 이용하여 이러한 방식으로 두 객체를 연관시키면 객체들 간에 가비지 콜렉션 관계가 형성되며; 1차 객체가 활성 상태이면, 2차 객체도 또한 활성 상태로 간주될 것이다.
- [0226] 700: 비트이다(즉, 용량이 하나의 이진 숫자인 메모리 위치이다).
- [0227] 702: IS-PRIMARY 비트로서, 즉 해당 비트의 값은 해당 비트가 내장된 객체가 1차 객체인지 여부를 표시한다.
- [0228] 704: 2차 객체 메모리 관리 전용인 객체의 내부의 공간; 공간(704)은 하나 이상의 비트를 포함할 수 있고, 특히 (구현예에 따라) IS-PRIMARY 비트(702)를 포함하거나 이로 구성될 수 있고; 예컨대, 추가된 참조물의 최소 생성을 나타내기 위해 다른 비트를 포함할 수 있다.
- [0229] 800: 플로우차트이며; 800은 또한 도 8의 플로우차트에 의해 예시되거나 이와 일관되는 메모리 관리 방법을 나타낸다.
- [0230] 802: 예컨대, IS-PRIMARY 비트(702)를 설정하거나 소거함으로써, 객체가 1차 객체인지 여부를 표시한다.
- [0231] 804: 1차성으로서, 예컨대, 객체가 1차 객체인지 여부를 표시한다.
- [0232] 806: 객체 활성 상태를 설정한다.
- [0233] 808: 통합된 참조물 및 2차 객체 표시를 사용하며; 808은 또한 이러한 종류의 표시의 통합을 나타낸다.

- [0234] 810: 동시적으로 실행한다.
- [0235] 812: 단지 1차 객체인 경우에만 모든 객체에 참조물 사이즈의 공간을 내장하는 것을 방지한다.
- [0236] 814: 일반적으로 가비지 콜렉션 처리를 수행한다.
- [0237] 816: 객체에 비트 변수 또는 다른 데이터 구조를 내장한다.
- [0238] 900: 플로우차트이며; 900은 또한 도 9의 플로우차트(도 8의 단계를 통합함)에 의해 예시되거나 이와 일관되는 메모리 관리 방법을 나타낸다.
- [0239] 902: 2차 객체를 표시(308)를 위한 참조된 객체로 취급한다.
- [0240] 904: 코드 경로를 따른다.
- [0241] 906: 활성 상태에 대해 객체를 확인한다.
- [0242] 908: 객체 참조물 또는 어드레스 또는 핸들을 열거한다.
- [0243] 910: 커맨드를 수신한다.
- [0244] 912: 하나의 객체를 다른 객체로 대체한다.
- [0245] 914: 데이터 구조를 업데이트한다.
- [0246] 916: 전체 테이블을 통한, 즉 표시를 위해 객체를 열거(908)할 때 1차 객체 및 비-1차 객체를 모두 포함하는 테이블을 통한 루핑에 대한 의존성을 방지한다.
- [0247] 918: 전체 테이블(1차 객체 및 비-1차 객체를 모두 포함하는 테이블)을 통해 루핑한다.
- [0248] 920: 표시를 위해 객체를 열거(908)할 때 프로모션 카운터에 대한 의존성을 방지한다.
- [0249] 922: 프로모션 카운터, 예컨대, 얼마나 많은 2차 객체가 활성 상태로 표시되도록 유지되는지에 대한 또는 얼마나 많은 2차 객체가 표시 동안 방문되도록 유지되는지에 대한 카운터이다.
- [0250] 924: IS-PRIMARY 비트를 설정하여, 객체가 1차 객체임을 표시한다.
- [0251] 926: 예컨대, 객체 생성자의 리턴 값으로서, 객체 핸들을 리턴한다.
- [0252] 928: 예컨대, 1차-2차 객체 관계를 나타내는 데이터 구조(602)에서, 2차 객체를 지정한다.
- [0253] 930: 2차 객체를 그 자체의 1차 객체로 취급한다.
- [0254] 932: 전체 멈춤(stop-the-world) 조건을 부과하는 것을 방지한다.
- [0255] 934: 전체 멈춤(stop the world) 조건을 부과한다.
- [0256] 936: 컴퓨팅 시스템에서 메모리를 관리한다.
- [0257] 938: 메모리 관리의 일부로서 가비지 콜렉션을 수행한다.
- [0258] 940: 동시적 가비지 콜렉션의 일부로서 수행(실행)한다.
- [0259] 942: 적어도 하나의 효율성(944)을 제공한다.
- [0260] 944: 효율성, 예컨대, 보다 빠른 표시, 객체에 대한 보다 낮은 저장 요구 사항, 보다 간단한 코드 설계이다.
- [0261] 946: 일부 다른 참조 번호가 할당되지 않은 본 개시 내용에서 논의된 임의의 단계이다.
- [0262] **결론**
- [0263] 요컨대, 본원의 교시 내용은 관리되는 메모리(112)를 사용하는 애플리케이션(124)의 성능을 향상시키는 다양한 컴퓨팅 시스템(102) 메모리 관리 기능(202)을 제공한다. 2차 객체(218)를 효율적으로 관리(936)하게 되면, 가비지 콜렉션 동시성(810)이 증가하고, 객체(212) 저장 요구 사항이 감소된다. 2차 객체(218)의 활성 상태 표시(308)는 참조된 객체(502)의 활성 상태 표시(308)와 통합(808)된다. 1차 객체가 아닌 객체(220)에서 참조물 사이즈의 2차 객체 식별자 필드(도 5의 224) 할당은 방지되고(812); 전용 비트(702)는 1차 객체(216)를 지정한다. 1차 객체(216)는 적어도 하나의 2차 객체(218)를 갖는 객체(212)이고; 2차 객체(218)는 1차 객체(216)가 활성

상태인 경우 가비지 콜렉션(938)에 의해 활성 상태인 것으로 간주된다(806). 1차 객체 X의 2차 객체(218)는 일반적으로 동일한 1차 객체 X의 참조된 객체(502)도 아니다. 활성 1차 객체(216)의 임의의 참조된 객체(502)도 여전히 활성 상태인 것으로 간주될 것이다(806). 참조된 객체(502)를 표시(308)하기 위한 코드 경로 246-R0는 2차 객체(218)를 표시(308)하기 위한 코드 경로 246-S0와 공유되어 보다 효율적인(944) 2차 객체 표시(308)를 허용할 수 있다. 1차-2차 객체 관계(602)는 종속 핸들(606)에서 표현될 수 있고, 해시 테이블(602) 또는 다른 데이터 구조에서 지정(928)될 수 있다.

[0264] 실시예는 또한 일반 데이터 보호 규정(General Data Protection Regulation)(GDPR)과 같은 테스트되고 적절한 보안 제어 및 개인 정보 제어를 자체적으로 포함하거나 이로부터 이익을 얻는 것으로 이해되며, 예컨대, 소프트웨어 내로 멀웨어를 주입하거나 활성화하는 것을 통해 컴퓨팅 시스템의 오용을 방지하는 데 도움을 주도록 적절한 조치가 취해져야 한다는 것으로 이해된다. 본원에 교시된 도구 및 기법의 사용은 이러한 제어권의 사용과 호환 가능하다.

[0265] 비록 Microsoft 기술이 동기를 부여하는 일부 예에서 사용되지만, 본원의 교시 내용은 Microsoft에 의해 제공되거나 관리되는 기술에서의 사용으로 제한되지는 않는다. 예를 들어, 적절한 라이선스 하에서, 본 교시 내용은 다른 클라우드 서비스 제공자 또는 다른 개발 환경 제공자 또는 다른 런타임 제공자에 의해 제공되는 소프트웨어 또는 서비스로 구현될 수 있다.

[0266] 특정 실시예가 본원에서 프로세스로서, 구성된 저장 매체로서, 또는 시스템으로서 명시적으로 예시되고 설명되었지만, 한 타입의 실시예에 대한 논의는 일반적으로 다른 실시예 타입으로 확장된다는 것으로 이해될 것이다. 예를 들어, 도 8 및 도 9와 관련된 프로세스의 설명은 또한 구성된 저장 매체를 기술하는 데 도움을 되며, 다른 도면과 관련하여 논의된 것과 같은 시스템 및 제품의 기술적 효과 및 동작을 기술하는 데 도움을 준다. 일 실시예로부터의 제한 사항이 반드시 다른 실시예로 관독되는 것은 아니다. 특히, 프로세스는 구성된 메모리와 같은 시스템 또는 제품을 논의하는 동안 제시된 데이터 구조 및 배열로 반드시 제한되지는 않는다.

[0267] 기술자는 구현 세부 사항이 특정 임계치, 비교치, 샘플 필드, 특정 종류의 런타임 또는 프로그래밍 언어 또는 아키텍처, 특정 스크립트 또는 다른 작업, 및 특정 컴퓨팅 환경과 같은 특정 코드와 관련될 수 있으며, 따라서 모든 실시예에 나타날 필요는 없는 것으로 이해할 것이다. 기술자는 또한 세부 사항을 논의하는 데 사용되는 프로그램 식별자 및 일부 다른 용어가 구현예에 특정되므로, 모든 실시예와 관련될 필요는 없다는 것을 이해할 것이다. 그럼에도 불구하고, 반드시 본원에 제시되어야 하는 것은 아니지만, 이러한 세부 사항은 컨텍스트를 제공함으로써 일부 독자에게 도움이 될 수 있고/있거나 본원에서 논의된 기술의 가능한 많은 구현예 중 몇 가지를 설명할 수 있다.

[0268] 청구되거나 청구 가능한 모든 실시예를 예시하지만 포괄적이지 않은 기술적 프로세스, 기술적 효과, 기술적 메커니즘, 및 기술적 세부 사항을 포함하여, 본원에 제공된 항목에 상당한 주의를 기울이면, 기술자는, 본 개시 내용 및 본원에 기술된 실시예가 기술 분야를 벗어난 주제나, 또는 주요하거나 본래의 원인 또는 동기와 같은 그 자체에 대한 아이디어나, 또는 단순한 결과 그 자체나, 또는 정신적 과정이나 정신적 단계나, 또는 비즈니스 방법이나 널리 퍼진 경제적 관행이나, 또는 인간 활동을 조직하는 단순한 방법이나, 또는 자연 법칙 그 자체나, 또는 자연적으로 발생하는 사물이나 과정이나, 또는 생물이나 생물의 일부나, 또는 수학적 공식 그 자체나, 또는 격리된 소프트웨어 그 자체나, 또는 단순히 기존의 컴퓨터나, 또는 완전히 지각할 수 없는 모든 것이나 추상적인 아이디어 그 자체나, 또는 중요하지 않은 사후 솔루션 활동이나, 또는 특정되지 않은 장치에서 완전히 구현된 임의의 방법이나, 또는 유용하고 구체적인 결과를 생성하지 못하는 임의의 방법이나, 또는 모든 사용 분야의 임의의 선점이나, 또는 해당 보호가 요구되거나 허가를 받거나 시행 중인 관할 구역의 법률에 따라 특허 보호를 받을 자격이 없는 임의의 다른 대상에 관한 것이 아니라는 것을 이해할 것이다.

[0269] 본원에서 일부 특징 X를 갖는 실시예에 대한 언급 및 일부 특징 Y를 갖는 실시예에 대한 본원의 다른 부분에 대한 언급은 본원에 명시적으로 언급되어 있지 않는 한 특징 X 및 특징 Y를 모두 갖는 본 개시 내용의 실시예로부터 배제되지 않는다. 모든 가능한 부정적 청구항 제한 사항은, 실시예의 일부로 언급된 임의의 특징이 또한 다른 실시예에 포함되지 않도록 명시적으로 제거될 수 있다는 점에서, 본 개시 내용의 범위 내에 속하며, 비록 그러한 특정 제외가 본원의 임의의 예에서 제공되지 않더라도 그러하다. 용어 "실시예"는 단지 본원에서 적용 가능한 법률에 따른 방식으로 적용되는 본원의 교시 내용의 "프로세스, 시스템, 제조물, 구성된 컴퓨터 판독 가능 저장 매체, 및/또는 다른 예"의 보다 편리한 형태로서 사용되는 것에 불과하다. 따라서, 주어진 "실시예"는 적어도 하나의 청구항을 따르는 한 본원에 개시된 특징의 임의의 조합을 포함할 수 있다.

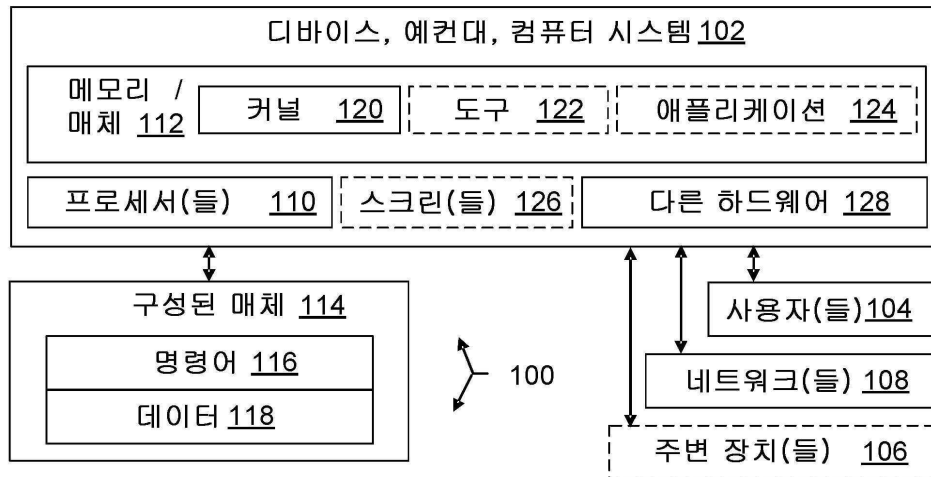
[0270] 도면에 도시된 모든 항목이 모든 실시예에 존재할 필요는 없다. 반대로, 실시예는 도면에 명시적으로 도시되지

많은 항목(들)을 포함할 수 있다. 일부 가능성이 본원의 특정 예에 의해 본문 및 도면에 설명되었지만, 실시예는 이러한 예에서 벗어날 수 있다. 예를 들어, 예의 특정 기술적 효과 또는 기술적 특징은 생략되거나, 이름이 바뀌거나, 다르게 그룹화되거나, 반복되거나, 하드웨어 및/또는 소프트웨어로 다르게 인스턴스화될 수 있거나, 둘 이상의 예에 나타나는 효과 또는 특징의 혼합일 수 있다. 일부 실시예에서 하나의 위치에 도시된 기능은 또한 다른 위치에 제공될 수 있고; 기술자는 전체적으로 볼 때 상호 작용하는 모듈의 집합체에서 원하는 기술적 효과를 반드시 생략하지 않고도, 주어진 구현예에서 다양한 방식으로 기능 모듈을 정의할 수 있음을 인식한다. 개별 단계는 공간 제한으로 인해 또는 편의상 도면의 단일 박스에 함께 표시될 수 있지만, 그럼에도 불구하고 개별적으로 수행할 수 있으며, 예컨대, 주어진 방법의 수행에서 하나의 단계는 다른 하나의 단계 없이도 수행될 수 있다.

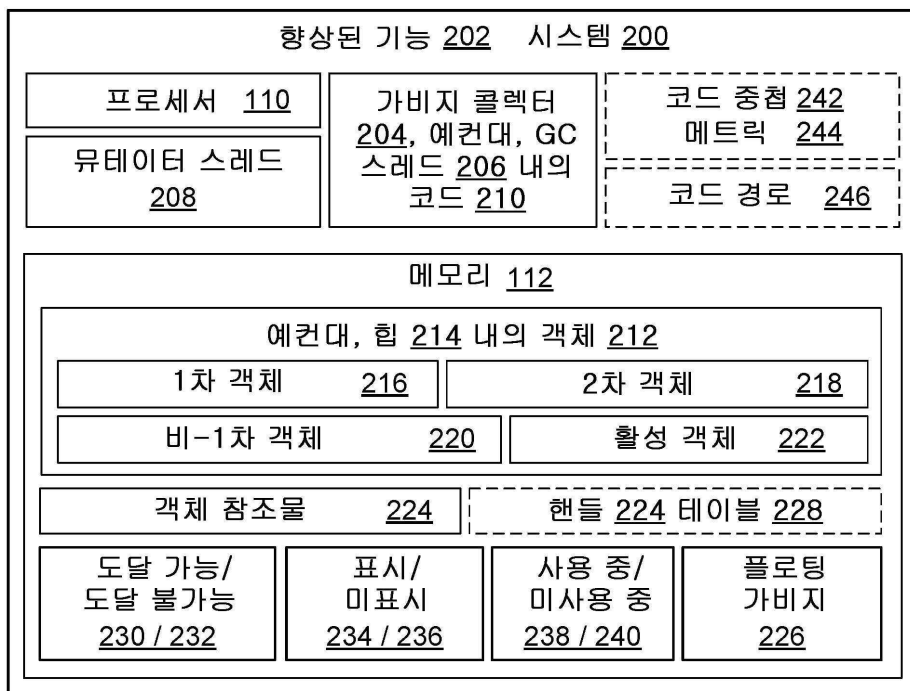
- [0271] 도면 전체에 걸쳐 참조 번호로 도면에 대한 참조가 행해졌다. 도면 또는 본문에서 주어진 참조 번호와 관련된 문구에서의 명백한 불일치는 단순히 해당 번호로 참조되는 범위를 넓히는 것으로 이해되어야 한다. 주어진 참조 번호의 서로 다른 예는 동일한 참조 번호가 사용되더라도 서로 다른 실시예를 지칭할 수 있다. 유사하게, 주어진 참조 번호는 동사, 명사, 및/또는 각각의 대응 인스턴스를 지칭하기 위해 사용될 수 있으며, 예컨대, 프로세서(110)는 명령어를 실행함으로써 명령어를 처리(110)할 수 있다.
- [0272] 본원에서 사용되는 바와 같이, 단수 형태의 용어는 하나 이상의 표시된 항목 또는 단계를 포함한다. 특히, 청구항에서 항목에 대한 참조는 일반적으로 적어도 하나의 이러한 항목이 존재함을 의미하고, 단계에 대한 참조는 단계의 적어도 하나의 인스턴스가 수행됨을 의미한다. 유사하게, 단수 형태는 문법적 오류 또는 오해를 피하기 위해 문맥이 허용하는 경우 복수 형태의 가능성을 포함하는 것으로 이해되어야 한다.
- [0273] 제목은 단지 편의만을 위한 것이며; 주어진 주제에 대한 정보를 제목이 해당 주제를 나타내는 섹션 외부에서 찾을 수도 있다.
- [0274] 출원되는 모든 청구항 및 요약서는 명세서의 일부이다.
- [0275] 본원에 사용된 임의의 용어가 산업 표준을 의미하거나 언급하는 한, 그리고 적용 가능한 법률이 그러한 표준의 특정 버전의 식별을 요구하는 한, 본 개시 내용은 적용 가능한 특허법에 따라 본 개시 내용의 가장 빠른 우선일을 기준으로, 최소한 초안 형태로 발표된 해당 표준의 최신 버전을 참조하는 것으로 이해되어야 한다(보다 더 최신인 것이 있다면 최종 형태가 우선한다).
- [0276] 예시적인 실시예가 도면에 도시되고 위에서 설명되었지만, 본 기술 분야의 통상의 기술자에게는 청구항에 제시된 원리 및 개념을 벗어나지 않고 다양한 수정이 행해질 수 있으며, 이러한 수정은 전체 추상적 개념을 포괄할 필요가 없다는 것이 분명할 것이다. 비록 발명이 구조적 특징 및/또는 절차적 동작에 특정된 언어로 기술되었지만, 첨부된 청구항에 정의된 발명은 반드시 청구항에 앞서 기술된 특정 특징 또는 동작에 반드시 제한되는 것은 아니라는 것을 이해해야 한다. 주어진 정의 또는 예에서 확인된 모든 수단 또는 양태 또는 기술적 효과가 모든 실시예에 존재하거나 활용될 필요는 없다. 오히려, 기술된 특정 특징 및 동작 및 효과는 청구항을 구현할 때 고려를 위한 예로서 개시된다.
- [0277] 전체 추상적인 사상을 포괄하는 데에는 미치지 못하지만 청구항의 동등성의 의미와 범위 내에 있는 모든 변경은 법이 허용하는 범위 내에서 최대한 수용되어야 한다.

도면

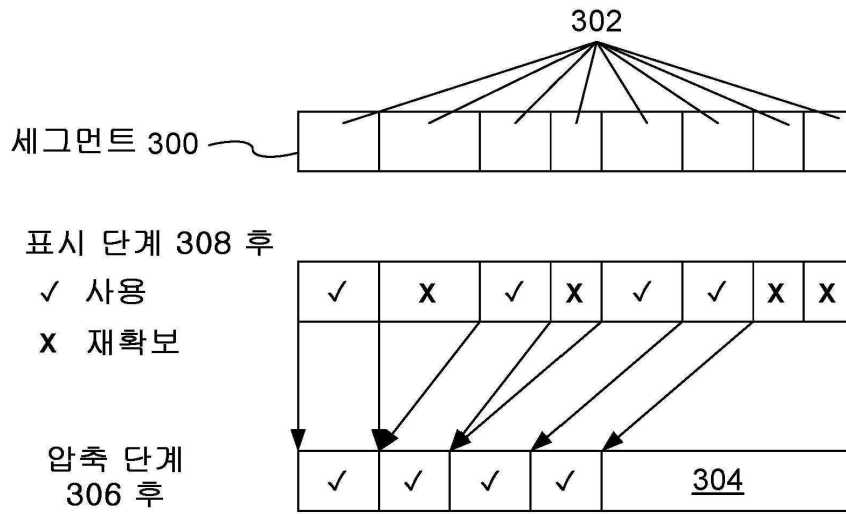
도면1



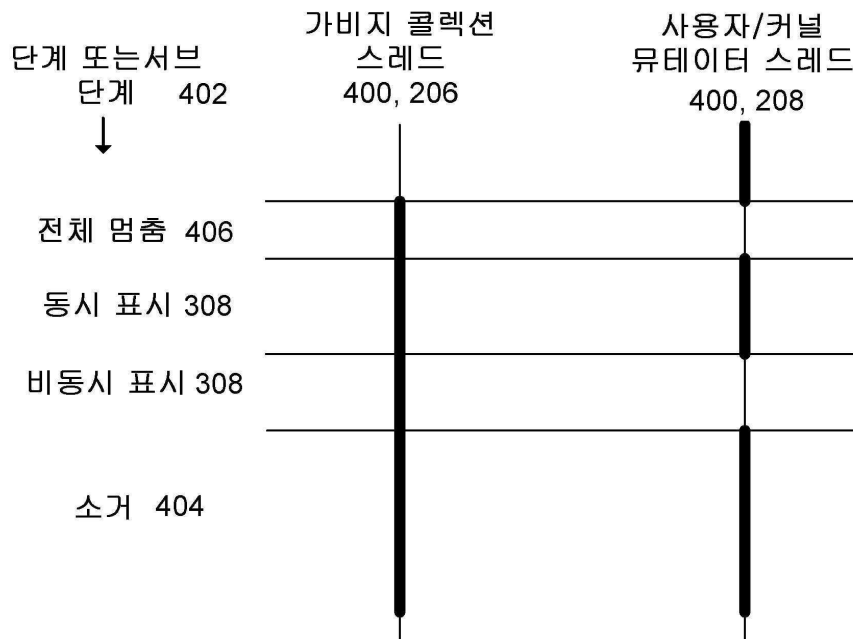
도면2



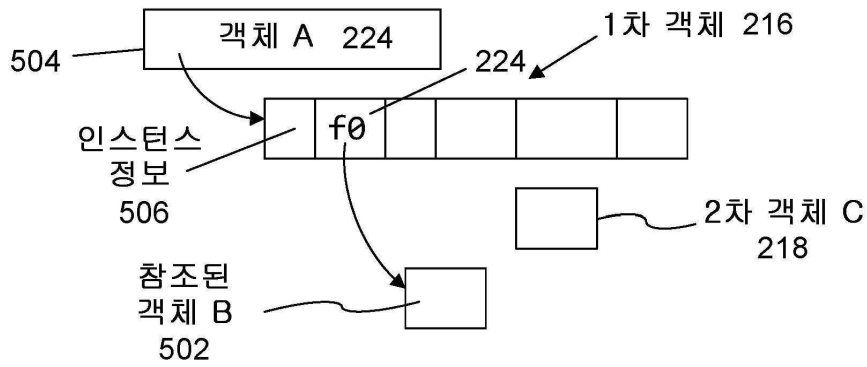
도면3



도면4



도면5



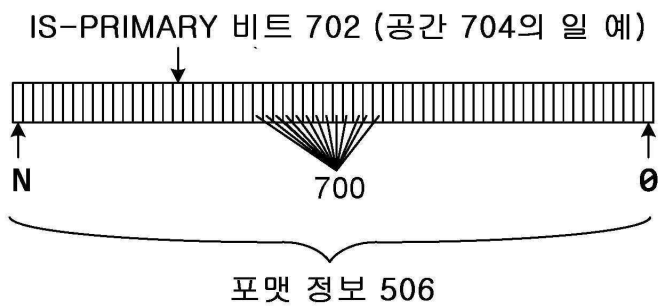
도면6

해시 테이블 602

종속 핸들 606	1차 객체 216 어드레스	2차 객체 218 어드레스
DH00	OBJ_A	OBJ_C
DH01	OBJ_C	OBJ_D

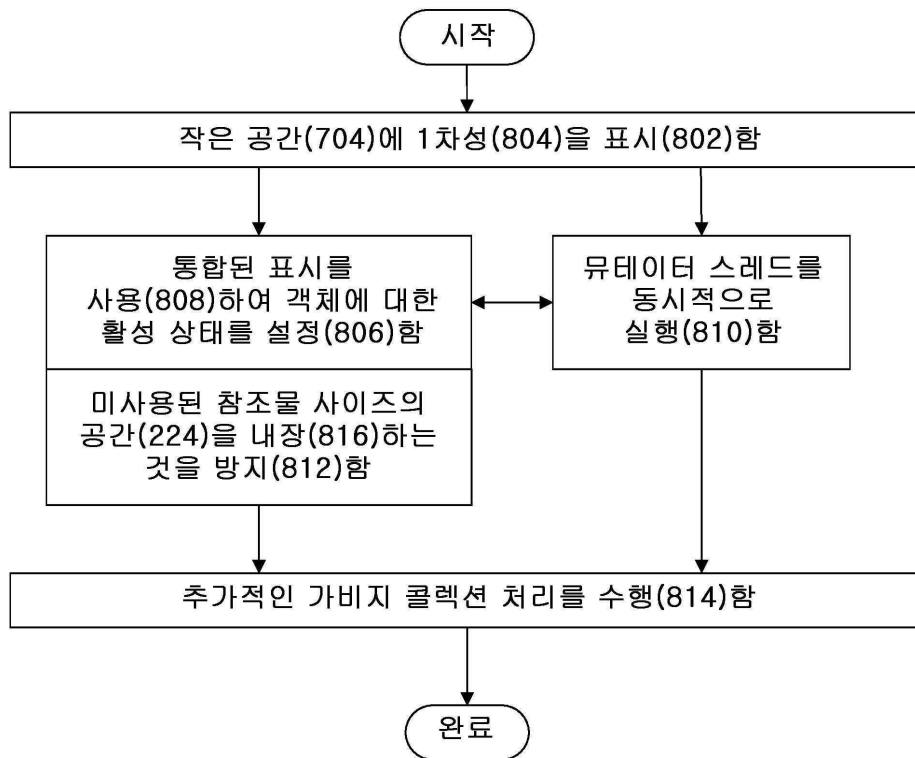
604

도면7



도면8

메모리 관리 프로세스의 예



도면9

