(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년06월04일
(11) 등록번호 10-2816189
(24) 등록일자 2025년05월29일

(51) 국제특허분류(Int. Cl.)
G06F 21/50 (2013.01) *G06F 9/4401* (2018.01)
(52) CPC특허분류
G06F 21/50 (2013.01)
G06F 9/4418 (2013.01)
(21) 출원번호 10-2022-0162182
(22) 출원일자 2022년11월29일
심사청구일자 2022년11월29일
(65) 공개번호 10-2024-0079358
(43) 공개일자 2024년06월05일
(56) 선행기술조사문헌
KR1020210157246 A*
(뒷면에 계속)
전체 청구항 수 : 총 2 항

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
박기웅
서울특별시 광진구 능동로17길 21, 304호(화양동)
최상훈
서울특별시 광진구 긴고랑로2길 38-2, 301호(중곡동)
(74) 대리인
양성보

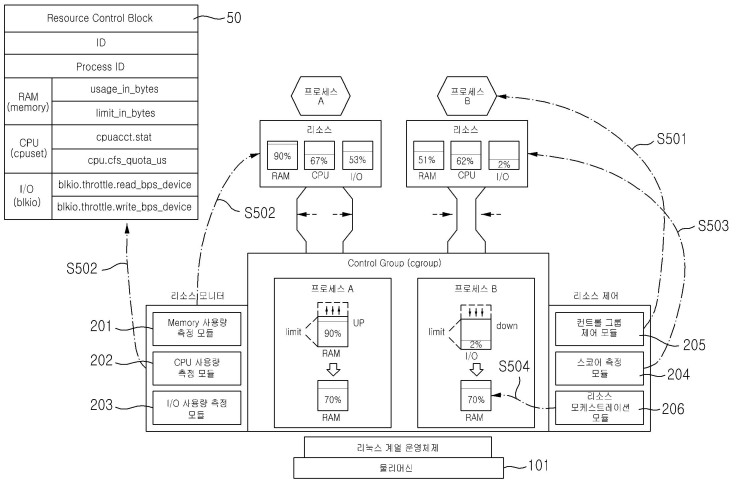
심사관 : 정성훈

(54) 발명의 명칭 **보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템**

(57) 요 약

보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템이 개시된다. 본 발명에 따른 프로세스 하이버네이션 시스템은 컴퓨터 운영체제(OS) 환경에서 구동되는 프로세스 중 타겟 프로세스를 설정하고, 상기 타겟 프로세스에 대해 리소스와 관련된 커널 기능인 컨트롤 그룹(control group)을 생성하고, 상기 컨트롤 그룹을 통해 상기 타겟 프로세스의 리소스 사용량을 모니터링하고, 상기 리소스 사용량에 기초하여 상기 타겟 프로세스의 리소스 사용 범위를 제한할 수 있다.

대 표 도



(56) 선행기술조사문헌
최상훈 외, "리눅스 운영체제에서 프로세스 자원 사용량을 효율적으로 오케스트레이션하기 위한 프레임워크"(2021.11.11.)*
KR100784595 B1
KR101219816 B1
JP2012252493 A
KR1020180020334 A
*는 심사관에 의하여 인용된 문헌

이 발명을 지원한 국가연구개발사업
과제고유번호 1711152732
과제번호 IITP-2021-0-01816-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 메타버스 자율트윈 핵심기술 연구
기 여 율 40/100
과제수행기관명 세종대학교 산학협력단
연구기간 2022.01.01 ~ 2022.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711161570
과제번호 NRF-2020R1A2C4002737
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 중견연계연구지원사업
연구과제명 IoT 침해사고 대응을 위한 지능형 분석 플랫폼 기술 연구
기 여 율 20/100
과제수행기관명 세종대학교 산학협력단
연구기간 2020.03.01 ~ 2023.02.28
이 발명을 지원한 국가연구개발사업
과제고유번호 1711174187
과제번호 001657941G0003072
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송기술국제공동연구(R&D)
연구과제명 랜섬웨어 침해사고 전주기적 능동대응을 위한 다각적 수집-분석-대응 플랫폼 개발
기 여 율 20/100
과제수행기관명 세종대학교 산학협력단
연구기간 2022.07.01 ~ 2024.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711093714
과제번호 2019-0-00426
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보보호핵심원천기술개발사업
연구과제명 IoT 기반 이식-침습형 고위험 의료장치를 위한 능동형 킬 스위치 및 바이오 마커 활용 방어 시스템 개발
기 여 율 20/100
과제수행기관명 국민대학교 산학협력단
연구기간 2019.04.01 ~ 2022.12.31

명 세 서

청구범위

청구항 1

컴퓨터 시스템으로 구현되는 프로세스 하이버네이션(process hibernation) 시스템에 있어서,

메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서는,

컴퓨터 운영체제(OS) 환경에서 구동되는 프로세스 중 타겟 프로세스를 설정하는 과정;

상기 타겟 프로세스에 대해 리소스와 관련된 커널 기능인 컨트롤 그룹(control group)을 생성하는 과정;

상기 컨트롤 그룹을 통해 상기 타겟 프로세스의 리소스 사용량을 모니터링하는 과정; 및

상기 리소스 사용량에 기초하여 상기 타겟 프로세스의 리소스 사용 범위를 제한하는 과정

을 처리하고,

상기 적어도 하나의 프로세서는,

오케스트레이션(orchestration) 시스템을 통해 리눅스(Linux) 운영체제 계열을 사용하는 물리머신에서 구동되는 모든 프로세스들의 리소스를 하나의 인터페이스를 통해 일괄적으로 제어하는 것으로,

상기 타겟 프로세스와 함께 해당 프로세스에 대한 제어 주기와 기준 스코어를 포함한 동적 제어 기준을 설정하고,

상기 타겟 프로세스에 대해 개별적인 컨트롤 그룹 파일시스템을 생성하여 상기 타겟 프로세스에 대한 리소스 제어 권한을 획득하고,

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스의 최대 메모리 사용량과 현재 메모리 사용량을 포함한 메모리 사용량을 수집하고,

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 사용한 CPU 사이클 수와 현재 설정된 CPU 할당 시간을 포함한 CPU 사용량을 수집하고,

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 송수신한 바이트 수와 현재 초당 처리 가능한 읽기/쓰기 바이트 수를 포함한 디스크 I/O 사용량을 수집하고,

상기 메모리 사용량과 상기 CPU 사용량 및 상기 디스크 I/O 사용량을 포함한 상기 리소스 사용량을 상기 타겟 프로세스에 대한 리소스 컨트롤 블록(resource control block) 구조체를 통해 기록 및 관리하고,

상기 리소스 컨트롤 블록 구조체에 기록된 상기 리소스 사용량을 이용하여 상기 타겟 프로세스에 대한 제어 스코어를 계산하고,

상기 제어 스코어가 상기 타겟 프로세스에 대해 설정된 상기 기준 스코어를 초과하는 경우 상기 타겟 프로세스를 제어 대상 리스트에 추가하고,

상기 제어 대상 리스트에 포함된 프로세스의 리소스 사용 범위를 제한하고,

상기 제어 스코어는 상기 타겟 프로세스의 리소스 허용 한도에 대한 현재 리소스 사용량의 비율, 및 상기 현재 리소스 사용량에 대한 이전 리소스 사용량의 증가 또는 감소 비율의 가중치 합계로 계산되는 것

을 특징으로 하는 프로세스 하이버네이션 시스템.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

컴퓨터 시스템에서 수행되는 프로세스 하이버네이션(process hibernation) 방법에 있어서,

상기 컴퓨터 시스템은 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고,

상기 프로세스 하이버네이션 방법은,

상기 적어도 하나의 프로세서의 의해, 컴퓨터 운영체제(OS) 환경에서 구동되는 프로세스 중 타겟 프로세스를 설정하는 단계;

상기 적어도 하나의 프로세서의 의해, 상기 타겟 프로세스에 대해 리소스와 관련된 커널 기능인 컨트롤 그룹(control group)을 생성하는 단계;

상기 적어도 하나의 프로세서의 의해, 상기 컨트롤 그룹을 통해 상기 타겟 프로세스의 리소스 사용량을 모니터링하는 단계; 및

상기 적어도 하나의 프로세서의 의해, 상기 리소스 사용량에 기초하여 상기 타겟 프로세스의 리소스 사용 범위를 제한하는 단계

를 포함하고,

상기 프로세스 하이버네이션 방법은,

오케스트레이션(orchestration) 시스템을 통해 리눅스(Linux) 운영체제 계열을 사용하는 물리머신에서 구동되는 모든 프로세스들의 리소스를 하나의 인터페이스를 통해 일괄적으로 제어하는 것으로,

상기 설정하는 단계는,

상기 타겟 프로세스와 함께 해당 프로세스에 대한 제어 주기와 기준 스코어를 포함한 동적 제어 기준을 설정하고,

상기 생성하는 단계는,

상기 타겟 프로세스에 대해 개별적인 컨트롤 그룹 파일시스템을 생성하여 상기 타겟 프로세스에 대한 리소스 제

어 권한을 획득하고,

상기 모니터링하는 단계는,

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스의 최대 메모리 사용량과 현재 메모리 사용량을 포함한 메모리 사용량을 수집하는 단계;

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 사용한 CPU 사이클 수와 현재 설정된 CPU 할당 시간을 포함한 CPU 사용량을 수집하는 단계; 및

상기 제어 주기를 단위로 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 송수신한 바이트 수와 현재 초당 처리 가능한 읽기/쓰기 바이트 수를 포함한 디스크 I/O 사용량을 수집하는 단계

를 포함하고,

상기 제한하는 단계는,

상기 메모리 사용량과 상기 CPU 사용량 및 상기 디스크 I/O 사용량을 포함한 상기 리소스 사용량을 상기 타겟 프로세스에 대한 리소스 컨트롤 블록(resource control block) 구조체를 통해 기록 및 관리하는 단계;

상기 리소스 컨트롤 블록 구조체에 기록된 상기 리소스 사용량을 이용하여 상기 타겟 프로세스에 대한 제어 스코어를 계산하는 단계;

상기 제어 스코어가 상기 타겟 프로세스에 대해 설정된 상기 기준 스코어를 초과하는 경우 상기 타겟 프로세스를 제어 대상 리스트에 추가하는 단계; 및

상기 제어 대상 리스트에 포함된 프로세스의 리소스 사용 범위를 제한하는 단계

를 포함하고,

상기 제어 스코어는 상기 타겟 프로세스의 리소스 허용 한도에 대한 현재 리소스 사용량의 비율, 및 상기 현재 리소스 사용량에 대한 이전 리소스 사용량의 증가 또는 감소 비율의 가중치 합계로 계산되는 것

을 특징으로 하는 프로세스 하이버네이션 방법.

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

발명의 설명

기 술 분 야

[0001] 아래의 설명은 리눅스(Linux) 운영체제 환경에서 프로세스 리소스 사용량을 오케스트레이션(orchestration)하는 기술에 관한 것이다.

배 경 기 술

[0002] 컴퓨팅 시스템에서는 컴퓨팅 시스템에 대한 다양한 공격 전략과 공격 기법을 탐지하고 대응하기 위해 여러 종류의 보안 솔루션들이 한 시스템 내에서 복합적으로 구동된다. 컴퓨팅 시스템에서 보안 공격을 탐지하고 대응하기 위한 여러 종류의 보안 솔루션을 보여준다. 이처럼 컴퓨팅 시스템을 보호하기 위해 시스템 내에서 다수의 보안 프로세스들이 복합적으로 구동되는 상황에서 각 프로세스는 컴퓨팅 시스템의 리소스를 지속적으로 점유하며 실행 상태를 유지한다.

[0003] 하지만, 컴퓨팅 시스템에서는 항상 모든 유형의 보안 앱이 지속적으로 구동될 필요가 없다. 컴퓨팅 시스템을 사용하는 목적과 상황에 따라 구동될 필요가 있는 보안 앱과 구동될 필요가 없는 보안 앱은 달라질 수 있다. 예를 들면, 특정 상황에서는 모니터링과 행위 분석(behavior analysis)과 관련된 보안 프로세스의 동작은 필요하지만, 패치(Patch)나 암호화(Encryption)와 관련된 보안 프로세스는 필요로 하지 않을 수 있다.

[0004] 그럼에도 불구하고 컴퓨팅 시스템에서 구동되고 있는 보안 프로세스 중 컴퓨팅 시스템 이용 목적에 필요하지 않은 프로세스들도 컴퓨팅 리소스를 점유하며 백그라운드에서 실행 상태를 유지한다. 가용성과 데이터 보호가 매우 중요한 컴퓨팅 시스템에서는 시스템 보호를 위해 시스템에서 구동되는 보안 프로세스의 수가 더욱 많아질 것이고, 이로 인한 비효율성 문제는 더 심각해질 것이다.

[0005] 따라서, 지속적으로 컴퓨팅 리소스를 점유하며 실행 상태를 유지하는 다수의 보안 애플리케이션에 대한 적절한 제어 기법이 필요하다.

[0006] 하지만, 보안 앱을 제어하는 것에는 여러 가지 어려움이 따른다. 먼저, 보안 프로세스는 공격자 또는 사용자에 의해 임의로 프로세스가 종료되는 것을 방지하기 위한 자가 보호 메커니즘을 가지고 있는 경우가 많다. 이러한 프로세스의 자가 보호 메커니즘은 운영체제가 제공하는 특정한 기능을 이용하거나, 프로세스가 종료되었는지 감시하고 있다가 프로세스가 종료되었을 시 빠르게 해당 프로세스를 다시 실행시켜 실행 상태를 유지하는 Keep-alive 프로세스 유지하는 방법을 이용하는 등 다양한 방법들을 활용하여 구현될 수 있다.

[0007] 이와 같이, 보안 프로세스는 다양한 프로세스 자가 보호 메커니즘을 가진 경우가 많기 때문에 여러 방식으로 구현된 각각의 자가 보호 메커니즘에 대하여 개별적인 프로세스 제어 방법을 고안하는 것은 매우 어려운 일이다.

발명의 내용

해결하려는 과제

[0008] 보안 앱의 리소스 할당량을 현재 리소스 사용량보다 적게 설정할 시 프로세스가 리소스 부족으로 강제 종료되는 현상을 방지하고 스코어 기반의 리소스 사용량 제어를 통해 보안 앱의 리소스 사용량을 최소화하는 방법을 제공한다.

[0009] 리눅스 운영체제 계열을 사용하는 물리머신에서 구동되는 모든 프로세스들의 리소스를 하나의 인터페이스를 통해 일괄적으로 제어 가능한 오케스트레이션 시스템을 제공한다.

과제의 해결 수단

[0010] 컴퓨터 시스템으로 구현되는 프로세스 하이버네이션(process hibernation) 시스템에 있어서, 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, 컴퓨터 운영체제(OS) 환경에서 구동되는 프로세스 중 타겟 프로세스를 설정하는 과정; 상기 타겟 프로세스에 대해 리소스와 관련된 커널 기능인 컨트롤 그룹(control group)을 생성하는 과정; 상기 컨트롤 그룹을 통해 상기 타겟 프로세스의 리소스 사용량을 모니터링하는 과정; 및 상기 리소스 사용량에 기초하여 상기 타겟 프로세스의 리소스 사용 범위를 제한하는 과정을 처리하는 프로세스 하이버네이션 시스템을 제공한다.

[0011] 일 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 타겟 프로세스와 함께 해당 프로세스에 대한 동적 제어 기준을 설정하고, 상기 리소스 사용량이 상기 동적 제어 기준을 벗어나면 상기 타겟 프로세스의 리소스 사용 범위를 제한할 수 있다.

[0012] 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 타겟 프로세스에 대해 개별적인 컨트롤 그룹 파일시스템을 생성하여 상기 타겟 프로세스에 대한 리소스 제어 권한을 획득할 수 있다.

[0013] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스의 메모리 사용량과 CPU 사용량 및 디스크 I/O 사용량 중 적어도 하나를 수집할 수 있다.

[0014] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스의 최대 메모리 사용량과 현재 메모리 사용량을 수집하고, 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 사용한 CPU 사이클 수와 현재 설정된 CPU 할당 시간을 수집하고, 상기 컨트롤 그룹의 서브시스템을 통해 상기 타겟 프로세스가 송수신한 바이트 수와 현재 초당 처리 가능한 읽기/쓰기 바이트 수를 수집할 수 있다.

[0015] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 리소스 사용량을 상기 타겟 프로세스에 대한 리소스 컨트롤 블록(resource control block) 구조체를 통해 기록 및 관리할 수 있다.

[0016] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 리소스 사용량이 설정 기준을 초과하는 경우 상기 타겟 프로세스를 제어 대상 리스트에 추가하고, 상기 제어 대상 리스트에 포함된 프로세스의 경우 리소스 사용 범위를 제한할 수 있다.

[0017] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 리소스 사용량을 이용하여 상기 타겟 프로세스에 대한 제어 스코어를 계산하고, 상기 제어 스코어가 상기 타겟 프로세스에 대해 설정된 기준 스코어를 초과하는 경우 상기 타겟 프로세스의 리소스 사용 범위를 제한할 수 있다.

[0018] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 상기 타겟 프로세스의 리소스 허용 한도에 대한 현재 리소스 사용량 비율과 상기 현재 리소스 사용량에 대한 이전 사용량 비율을 이용하여 상기 제어 스코어를 계산할 수 있다.

[0019] 컴퓨터 시스템에서 수행되는 프로세스 하이버네이션(process hibernation) 방법에 있어서, 상기 컴퓨터 시스템은 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 프로세스 하이버네이션 방법은, 상기 적어도 하나의 프로세서의 의해, 컴퓨터 운영체제(OS) 환경에서 구동되는 프로세스 중 타겟 프로세스를 설정하는 단계; 상기 적어도 하나의 프로세서의 의해, 상기 타겟 프로세스에 대해 리소스와 관련된 커널 기능인 컨트롤 그룹(control group)을 생성하는 단계; 상기 적어도 하나의 프로세서의 의해, 상기 컨트롤 그룹을 통해 상기 타겟 프로세스의 리소스 사용량을 모니터링하는 단계; 및 상기 적어도 하나의 프로세서의 의해, 상기 리소스 사용량에 기초하여 상기 타겟 프로세스의 리소스 사용 범위를 제한하는 단계를 포함하는 프로세스 하이버네이션 방법을 제공한다.

발명의 효과

[0020] 본 발명의 실시예에 따르면, 리눅스 운영체제 환경에서 구동 중인 다수의 보안 앱의 리소스를 하나의 인터페이스로 제어할 수 있는 시스템을 제공한다. 리눅스 운영체제 환경에서 구동 중인 프로세스의 CPU, 메모리, I/O, 네트워크 등 리소스 사용량을 모니터링하여 현재 사용량에 비해 불필요하게 리소스가 많이 할당되었을 때 프로세스가 사용할 수 있는 리소스의 최대 사용량을 제한하는 방법을 통해 불필요한 리소스 소모를 방지하고 사용자가 사용 가능한 리소스 자원을 최대로 확보할 수 있다.

[0021] 본 발명의 실시예에 따르면, 클라우드 플랫폼 환경에서 구동 중인 인스턴스의 리소스를 효율적으로 기록할 수 있는 시스템을 제공한다. 가상화 환경에서 구동되는 인스턴스(프로세스)들의 메모리를 수집하는 과정에서 발생하는 메모리 데이터의 중복성을 최소화할 수 있는 방법을 통해 메모리 수집 연산의 속도를 가속하고 불필요하게 낭비될 수 있는 스토리지 공간을 확보할 수 있다.

도면의 간단한 설명

[0022] 도 1은 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션 시스템이 적용되는 리눅스 운영체제 환경 예시를 도시한 것이다.

도 2는 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템의 내부 모듈을 도시한 것이다.

도 3은 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 과정의 일례를 도시한 순서도이다.

도 4는 본 발명의 일 실시예에 있어서 리눅스 환경에서의 구동 중인 프로세스들의 리소스 사용량을 모니터링하는 과정 예시를 도시한 것이다.

도 5는 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템에서 리소

스 사용량을 모니터링하고 오케스트레이션 하는 과정 예시를 도시한 것이다.

도 6은 본 발명의 일 실시예에 있어서 프로세스 하이버네이션 시스템을 구현하기 위한 컴퓨터 시스템의 예를 도시한 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0023] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0025] 본 발명의 실시예들은 리눅스 운영체제 환경에서 프로세스 리소스 사용량을 오케스트레이션하는 기술에 관한 것이다.

[0026] 본 명세서에서 구체적으로 개시되는 것들을 포함하는 실시예들은 보안 앱의 리소스 할당량을 현재 리소스 사용량보다 적게 설정할 시 프로세스가 리소스 부족으로 강제 종료되는 현상을 방지하고 스코어 기반의 리소스 사용량 제어를 통해 보안 앱의 리소스 사용량을 최소화할 수 있다.

[0027] 본 명세서에서 구체적으로 개시되는 것들을 포함하는 실시예들은 오케스트레이션 시스템을 통해 리눅스 운영체제 계열을 사용하는 물리머신에서 구동되는 모든 프로세스들의 리소스를 하나의 인터페이스를 통해 일괄적으로 제어할 수 있다.

[0028] 본 발명의 실시예들에 따른 프로세스 하이버네이션 시스템은 적어도 하나의 컴퓨터 장치에 의해 구현될 수 있으며, 본 발명의 실시예들에 따른 프로세스 하이버네이션 방법은 프로세스 하이버네이션 시스템에 포함되는 적어도 하나의 컴퓨터 장치를 통해 수행될 수 있다. 이때, 컴퓨터 장치에는 본 발명의 일실시예에 따른 컴퓨터 프로그램이 설치 및 구동될 수 있고, 컴퓨터 장치는 구동된 컴퓨터 프로그램의 제어에 따라 본 발명의 실시예들에 따른 프로세스 하이버네이션 방법을 수행할 수 있다. 상술한 컴퓨터 프로그램은 컴퓨터 장치와 결합되어 프로세스 하이버네이션 방법을 컴퓨터에 실행시키기 위해 컴퓨터 판독 가능한 기록매체에 저장될 수 있다.

[0029] 도 1은 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션 시스템이 적용되는 리눅스 운영체제 환경 예시를 도시한 것이다. 도 1은 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템이 적용되는 환경을 나타내고 있다.

[0030] 도 1을 참조하면, 본 발명에 따른 프로세스 하이버네이션 시스템은 물리적인 자원과 리눅스 계열 운영체제가 구동되고 있는 물리머신(101), 그리고 물리머신(101)을 사용하는 사용자(102)가 존재하는 환경에서 적용될 수 있다.

[0031] 본 발명에 따른 프로세스 하이버네이션 시스템은 사용자(102)의 PC 환경에서 구동되는 보안 앱을 포함한 백그라운드 프로세스의 리소스 사용량을 최소화하여 주요 프로세스의 연산 처리량을 극대화시킬 수 있다.

[0032] 본 발명에 따른 프로세스 하이버네이션 시스템은 다음의 절차를 통해 이루어진다.

[0033] 1. 타겟 보안 프로세스를 선정하고 제어 기준 룰을 지정하는 과정

[0034] 2. 리소스 사용량을 모니터링하는 과정

[0035] 3. 프로세스 리소스 컨트롤 블록(RCB)을 업데이트 하는 과정

[0036] 4. 리소스 사용량에 대한 스코어 측정 과정

[0037] 5. 리소스 제어 대상을 선정하는 과정

[0038] 6. 리소스 사용량을 제어하는 과정

[0039] 도 2는 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템의 내부 모듈을 도시한 것이다.

[0040] 도 2를 참조하면, 본 발명에 따른 프로세스 하이버네이션 시스템은 프로세스의 리소스들을 모니터링하기 위한 모듈들(201, 202, 203), 그리고 프로세스의 리소스들을 제어하기 위한 모듈들(204, 205, 206)로 구성될 수 있다.

[0041] 프로세스의 리소스를 모니터링하는 모듈은 다수의 프로세스가 구동되는 환경에서 프로세스의 리소스(Memory, CPU, I/O, Network) 사용량을 측정하는 역할을 하는 것으로, 상세하게 프로세스의 메모리 사용량을 실시간으로 수집하는 메모리 사용량 측정 모듈(201), 프로세스의 CPU 사용량을 실시간으로 수집하는 CPU 사용량 측정 모듈

(202), 프로세스의 디스크 I/O 사용량을 실시간으로 수집하는 I/O 사용량 측정 모듈(203)을 포함할 수 있다.

[0042] 그리고, 프로세스의 리소스를 제어하는 모듈은 리소스 사용량을 스코어로 변환하는 역할, 프로세스 컨트롤 그룹을 관리하는 역할, 프로세스의 리소스를 오케스트레이션하는 역할을 하는 것으로, 상세하게 프로세스의 리소스 사용량에 대한 스코어를 측정하는 스코어 측정 모듈(204), 타겟 프로세스를 컨트롤 그룹(control group, 이하, 'Cgroup'이라 칭함)으로 관리하는 컨트롤 그룹 제어 모듈(205), 프로세스의 리소스를 오케스트레이션하는 리소스 오케스트레이션 모듈(206)을 포함할 수 있다.

[0043] 메모리 사용량 측정 모듈(201)은 구동 중인 프로세스의 메모리 사용량을 측정하는데 사용된다. 이때, 메모리의 사용량은 Cgroup의 메모리 서브시스템 중 usage_in_bytes 컨트롤 제어파일을 통해 확인한다. Cgroup은 다양한 유형의 리소스(메모리, CPU 등)를 제한하고 모니터링할 수 있는 계층적 그룹으로 프로세스를 구성할 수 있도록 하는 커널 기능을 의미한다. Cgroup의 네임스페이스(namespace)는 /proc/[pid]/Cgroup와 /proc/[pid]/mountinfo를 통해 표시되는 프로세스의 Cgroup의 범위를 가상화할 수 있다. 각 Cgroup에 한 개 이상의 프로세스가 소속될 수 있고, Cgroup에 허용할 리소스를 사용자가 원하는 만큼 할당할 수 있다. 또한, Cgroup을 통해 해당 Cgroup 내의 프로세스가 점유하고 있는 리소스에 대한 정보를 모니터링 할 수 있다. 메모리 사용량 측정 모듈(201)은 memory.memsw.usage_in_bytes를 통해 Cgroup에서 프로세스에 의해 사용되는 현재 메모리 사용량과 스왑 영역 사용량의 합계를 모니터링 할 수 있다.

[0044] CPU 사용량 측정 모듈(202)은 구동 중인 프로세스가 사용중인 CPU 정보를 측정하는데 사용된다. 이때, CPU 사용률 정보는 Cgroup의 cpuacct 서브시스템 중 cpuacct.stat 컨트롤 제어파일을 통해 확인할 수 있다. CPU 사용량 측정 모듈(202)은 cpuacct.stat를 통해 사용자 모드와 시스템(커널) 모드에서 Cgroup의 작업과 자식 작업이 사용한 CPU 사이클 수(USER_HZ로 시스템에 정의된 단위를 사용)를 모니터링할 수 있다.

[0045] I/O 사용량 측정 모듈(203)은 구동 중인 프로세스의 디스크 I/O 사용량을 수집하는데 사용된다. 이때, Cgroup의 blkio 서브시스템 중 blkio.io_service_bytes 컨트롤 제어파일을 통해 확인할 수 있다. I/O 사용량 측정 모듈(203)은 blkio.io_service_bytes를 통해서 Cgroup에 의해 특정 장치 간에 전송되는 바이트 수를 모니터링할 수 있다. 이때, 항목은 major, minor, operation, bytes의 4개 필드로 구성된다. Major와 minor는 리눅스 할당 장치에 지정된 장치 유형과 노드 번호를 의미하고, operation은 작업 유형(read, write, sync, async 등)을 나타내며, bytes는 전송된 바이트 수를 나타낸다.

[0046] 스코어 측정 모듈(204)은 리소스의 사용량을 스코어로 계산하는데 사용된다.

[0047] 컨트롤 그룹 제어 모듈(205)은 프로세스의 리소스 사용량을 제어하기 위해 프로세스 전용 Cgroup 파일시스템을 관리하는데 있어 사용된다.

[0048] 리소스 오케스트레이션 모듈(206)은 프로세스의 리소스 최대 사용량을 제어하는데 있어서 사용된다. 리소스의 제한은 Cgroup의 서브시스템 제어를 통해 관리한다.

[0049] 도 3은 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 과정의 일례를 도시한 순서도이다.

[0050] 도 3을 참조하면, 본 발명에 따른 프로세스 하이버네이션 시스템은 현재 리눅스 운영체제 환경에서 구동되는 다양한 프로세스 중 리소스의 사용량을 동적으로 제어할 제어 대상 프로세스인 타겟 프로세스 명을 설정할 수 있고 해당 프로세스에 대한 제어 주기와 동적 제어의 기준 스코어를 포함한 제어 기준 룰을 설정한다(S301).

[0051] 프로세스 하이버네이션 시스템은 지정된 제어 대상 프로세스에 대해 개별적인 Cgroup 파일시스템을 생성하고 리소스 제어 권한을 획득한다(S302).

[0052] 이후, 프로세스 하이버네이션 시스템은 프로세스 별 할당된 Cgroup의 서브시스템을 모니터링하여 리소스를 수집한다(S303 내지 S305). 메모리 사용량의 경우 Cgroup의 memory 서브시스템 중 usage_in_bytes 컨트롤 제어파일 읽기 연산을 통해 수집한다(S303). CPU의 사용량의 경우 Cgroup의 cpuacct 서브시스템 중 cpuacct.stat 컨트롤 제어파일 읽기 연산을 통해 수집한다(S304). I/O의 사용량의 경우 Cgroup의 blkio 서브시스템 중 blkio.io_service_bytes 컨트롤 제어파일 읽기 연산을 통해 수집한다(S305).

[0053] 이때, 프로세스 하이버네이션 시스템은 상기한 단계(S303 내지 S305)에서 수집된 리소스 사용량 정보를 리소스 컨트롤 블록(RCB) 구조체를 통해 관리할 수 있다(S306).

[0054] 다음, 프로세스 하이버네이션 시스템은 리소스 컨트롤 블록(RCB) 구조체에 업데이트된 리소스 사용량에 대한 스

코어를 계산한다(S307).

[0055] 프로세스 하이버네이션 시스템은 단계(S307)에서 계산된 스코어가 사용자에 의해 지정된 기준 스코어를 초과하는지 검사한다(S308).

[0056] 프로세스 하이버네이션 시스템은 타겟 프로세스의 리소스 사용량에 대한 스코어가 기준 스코어를 초과하지 않는 경우 리소스 사용량 수집 과정(S303 내지 S305)을 반복한다.

[0057] 한편, 프로세스 하이버네이션 시스템은 타겟 프로세스의 리소스 사용량이 사용자에 의해 설정된 동적 제어 기준을 벗어나면, 일례로 타겟 프로세스의 리소스 사용량에 대한 스코어가 기준 스코어를 초과하는 경우 해당 프로세스를 제어 대상 리스트에 추가한다(S309).

[0058] 프로세스 하이버네이션 시스템은 제어 대상 리스트에 포함된 프로세스에 대해 최대 리소스 사용 범위를 제한하여 해당 프로세스의 리소스 사용량이 사용자가 지정한 스코어를 초과하지 않도록 설정한다(S310).

[0059] 도 4는 본 발명의 일 실시예에 있어서 리눅스 환경에서의 구동 중인 프로세스들의 리소스 사용량을 모니터링하는 과정 예시를 도시한 것이다.

[0060] 리눅스 운영체제 환경에서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템이 구동 중인 프로세스의 리소스 사용량을 수집하는 과정은 다음과 같다.

[0061] 도 4를 참조하면, 본 발명에 따른 프로세스 하이버네이션 시스템은 리눅스 운영체제 환경에서 구동 중인 프로세스들 중 사용자가 지정한 프로세스 리스트 정보를 바탕으로 미리 지정된 제어 주기를 단위로 리소스 사용량을 수집한다(S401 내지 S403). 메모리 사용량 측정 모듈(201)은 타겟 프로세스의 최대 메모리 사용량과 현재 사용량을 수집한다(S401). CPU 사용량 측정 모듈(202)은 타겟 프로세스가 사용한 CPU 사이클 수와 현재 설정된 CPU 할당 시간을 수집한다(S402). I/O 사용량 측정 모듈(203)은 타겟 프로세스가 지정된 디바이스에 송수신한 바이트 수와 현재 초당 처리할 수 있는 읽기/쓰기 바이트 수를 수집한다(S403).

[0062] 도 5는 본 발명의 일 실시예에 있어서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템에서 리소스 사용량을 모니터링하고 오케스트레이션 하는 과정 예시를 도시한 것이다.

[0063] 리눅스 운영체제 환경에서 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템을 통해 프로세스의 리소스가 제어되는 과정은 다음과 같다.

[0064] 도 5를 참조하면, 본 발명에 따른 프로세스 하이버네이션 시스템은 컨트롤 그룹 제어 모듈(205)를 통해 제어 대상 프로세스에 대해 개별적인 Cgroup 파일시스템을 생성하고 리소스 제어 권한을 획득한다(S501).

[0065] 이후, 프로세스 하이버네이션 시스템은 메모리 사용량 측정 모듈(201), CPU 사용량 측정 모듈(202), I/O 사용량 측정 모듈(203)을 통해 타겟 프로세스의 리소스 사용량을 수집한다(S502). 수집된 리소스 정보는 리소스 컨트롤 블록 구조체(50)를 통해 기록 및 관리될 수 있다.

[0066] 프로세스 하이버네이션 시스템은 스코어 측정 모듈(204)을 통해 리소스 컨트를 블록 구조체(50)에 삽입된 리소스 사용량 정보를 분석하여 스코어를 측정한다(S503). 스코어 측정 모듈(204)은 타겟 프로세스의 리소스 사용량을 점수화하여 스코어로 변환할 수 있다. 일례로, 스코어 측정 모듈(204)은 리소스 컨트를 블록 구조체(50)에 저장된 정보를 기반으로 타겟 프로세스의 리소스 사용량에 따른 프로세스 제어 스코어를 계산하는 것으로, 허용 한도에 대한 현재 리소스 사용량의 비율, 및 현재 리소스 사용량에 대한 이전 사용량의 증가 또는 감소된 리소스 사용량의 비율이라는 두 가지 요소의 가중치 합계로 프로세스 제어 스코어를 계산할 수 있다.

[0067] 예를 들어, 프로세스 제어 스코어는 수학식 1과 같이 정의할 수 있다.

[0068] [수학식 1]

[0069] $$score = \left(\frac{x_n}{y}\right) + \left((1 - \left(\frac{x_{n-1}}{x_n}\right)) \times 2\right)$$

[0070] 여기서, x는 리소스 사용량을 나타내고, y는 리소스 허용 한도를 나타낸다

[0071] 프로세스 하이버네이션 시스템은 측정된 스코어가 사용자에 의해 지정된 기준 스코어를 초과할 경우 리소스 오케스트레이션 모듈(206)을 통해 타겟 프로세스의 리소스 최대 사용량을 제한한다(S504). 리소스 오케스트레이션 모듈(206)은 스코어 측정 모듈(204)을 통해 계산된 프로세스 제어 스코어를 기준으로 타겟 프로세스에 대한 리

소스 제한을 동적으로 조정할 수 있다. 다시 말해, 프로세스 하이버네이션 시스템은 사용자가 제어할 프로세스에 대해 별도의 공간을 Cgroup에 할당하고 각각의 Cgroup을 액세스하여 최대 리소스 사용량을 동적으로 제어할 수 있다.

[0072] 예를 들어, 리소스 오케스트레이션 모듈(206)은 프로세스 제어 스코어가 1.0을 초과하면 타겟 프로세스에 대한 리소스 제한을 확장하고 프로세스 제어 스코어가 0.5 미만이면 타겟 프로세스에 대한 리소스 제한을 축소할 수 있다. 이에 따라, 리소스 부족으로 인해 프로세스가 우발적으로 종료되는 것을 방지함과 아울러, 불필요한 리소스 할당을 차단하여 프로세스의 리소스 사용을 최소화할 수 있다.

[0073] 도 6은 본 발명의 일실시예에 따른 컴퓨터 시스템의 예를 도시한 블록도이다. 본 발명에 따른 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템은 도 6과 같이 구성된 컴퓨터 시스템(600)에 의해 구현될 수 있다.

[0074] 도 6에 도시된 바와 같이 컴퓨터 시스템(600)은 본 발명의 실시예들에 따른 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 방법을 실행하기 위한 구성요소로서, 메모리(610), 프로세서(620), 통신 인터페이스(630) 그리고 입출력 인터페이스(640)를 포함할 수 있다.

[0075] 메모리(610)는 컴퓨터에서 판독 가능한 기록매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 여기서 ROM과 디스크 드라이브와 같은 비소멸성 대용량 기록장치는 메모리(610)와는 구분되는 별도의 영구 저장 장치로서 컴퓨터 시스템(600)에 포함될 수도 있다. 또한, 메모리(610)에는 운영체제와 적어도 하나의 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 메모리(610)와는 별도의 컴퓨터에서 판독 가능한 기록매체로부터 메모리(610)로 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록매체를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록매체가 아닌 통신 인터페이스(630)를 통해 메모리(610)에 로딩될 수도 있다. 예를 들어, 소프트웨어 구성요소들은 네트워크(660)를 통해 수신되는 파일들에 의해 설치되는 컴퓨터 프로그램에 기반하여 컴퓨터 시스템(600)의 메모리(610)에 로딩될 수 있다.

[0076] 프로세서(620)는 기본적인 산술, 로직 및 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(610) 또는 통신 인터페이스(630)에 의해 프로세서(620)로 제공될 수 있다. 예를 들어 프로세서(620)는 메모리(610)와 같은 기록 장치에 저장된 프로그램 코드에 따라 수신되는 명령을 실행하도록 구성될 수 있다.

[0077] 통신 인터페이스(630)는 네트워크(660)를 통해 컴퓨터 시스템(600)이 다른 장치와 서로 통신하기 위한 기능을 제공할 수 있다. 일례로, 컴퓨터 시스템(600)의 프로세서(620)가 메모리(610)와 같은 기록 장치에 저장된 프로그램 코드에 따라 생성한 요청이나 명령, 데이터, 파일 등이 통신 인터페이스(630)의 제어에 따라 네트워크(660)를 통해 다른 장치들로 전달될 수 있다. 역으로, 다른 장치로부터의 신호나 명령, 데이터, 파일 등이 네트워크(660)를 거쳐 컴퓨터 시스템(600)의 통신 인터페이스(630)를 통해 컴퓨터 시스템(600)으로 수신될 수 있다. 통신 인터페이스(630)를 통해 수신된 신호나 명령, 데이터 등은 프로세서(620)나 메모리(610)로 전달될 수 있고, 파일 등은 컴퓨터 시스템(600)이 더 포함할 수 있는 저장 매체(상술한 영구 저장 장치)로 저장될 수 있다.

[0078] 통신 방식은 제한되지 않으며, 네트워크(660)가 포함할 수 있는 통신망(일례로, 이동통신망, 유선 인터넷, 무선 인터넷, 방송망)을 활용하는 통신 방식뿐만 아니라 기기들 간의 근거리 유선/무선 통신 역시 포함될 수 있다. 예를 들어, 네트워크(660)는, PAN(personal area network), LAN(local area network), CAN(campus area network), MAN(metropolitan area network), WAN(wide area network), BBN(broadband network), 인터넷 등의 네트워크 중 하나 이상의 임의의 네트워크를 포함할 수 있다. 또한, 네트워크(660)는 버스 네트워크, 스타 네트워크, 링 네트워크, 메쉬 네트워크, 스타-버스 네트워크, 트리 또는 계층적(hierarchical) 네트워크 등을 포함하는 네트워크 토폴로지 중 임의의 하나 이상을 포함할 수 있으나, 이에 제한되지 않는다.

[0079] 입출력 인터페이스(640)는 입출력 장치(650)와의 인터페이스를 위한 수단일 수 있다. 예를 들어, 입력 장치는 마이크, 키보드, 카메라 또는 마우스 등의 장치를, 그리고 출력 장치는 디스플레이, 스피커와 같은 장치를 포함할 수 있다. 다른 예로 입출력 인터페이스(640)는 터치스크린과 같이 입력과 출력을 위한 기능이 하나로 통합된 장치와의 인터페이스를 위한 수단일 수도 있다. 입출력 장치(650)는 컴퓨터 시스템(600)과 하나의 장치로 구성될 수도 있다.

[0080] 또한, 다른 실시예들에서 컴퓨터 시스템(600)은 도 6의 구성요소들보다 더 적은 혹은 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 컴퓨터 시스템(600)은 상술한 입출력 장치(650) 중 적어도 일부를 포함하도록 구현되거나 또는 트랜시버(transceiver), 각종 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.

[0081] 이처럼 본 발명의 실시예들에 따르면, 사용자의 PC 환경(리눅스 계열 운영체제)에서 구동되는 보안 앱을 포함한 백그라운드 프로세스로 인해 소모되고 있는 리소스를 분석하고 제어함으로써 불필요한 리소스 소모를 최소화함과 동시에 주요 프로세스가 더 많은 리소스를 할당받음에 따라 연산 처리량이 극대화될 수 있다.

[0082] 본 발명에 따른 보안 앱 오케스트레이션을 위한 프로세스 하이버네이션 시스템은 리소스 사용이 제한적인 초소형 임베디드 시스템, 대규모의 서비스가 유기적으로 연결되어 구동되는 클라우드 컴퓨팅 분야 등에 적용될 수 있다.

[0083] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 어플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0084] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

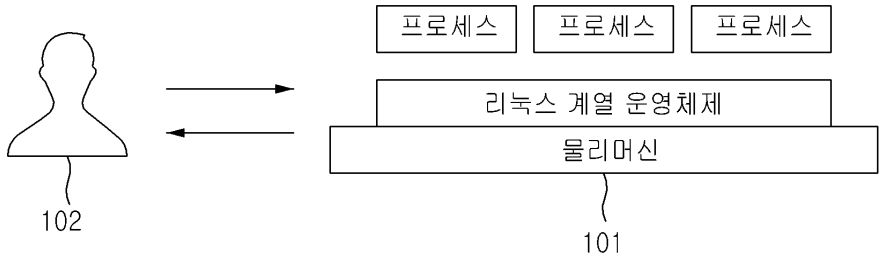
[0085] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 이때, 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수 개의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 어플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다.

[0086] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

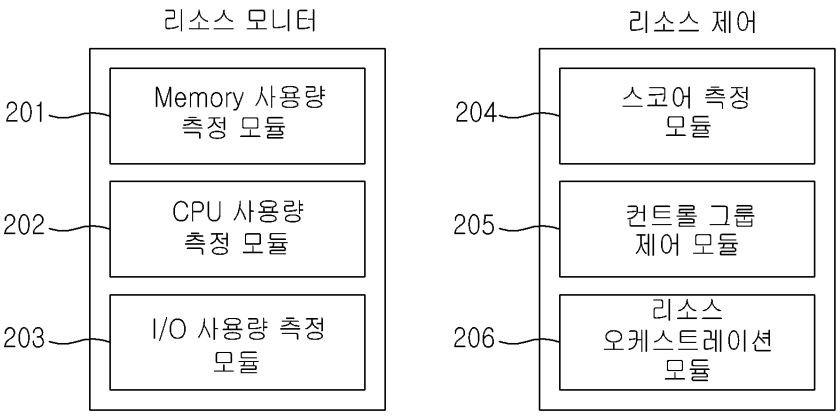
[0087] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

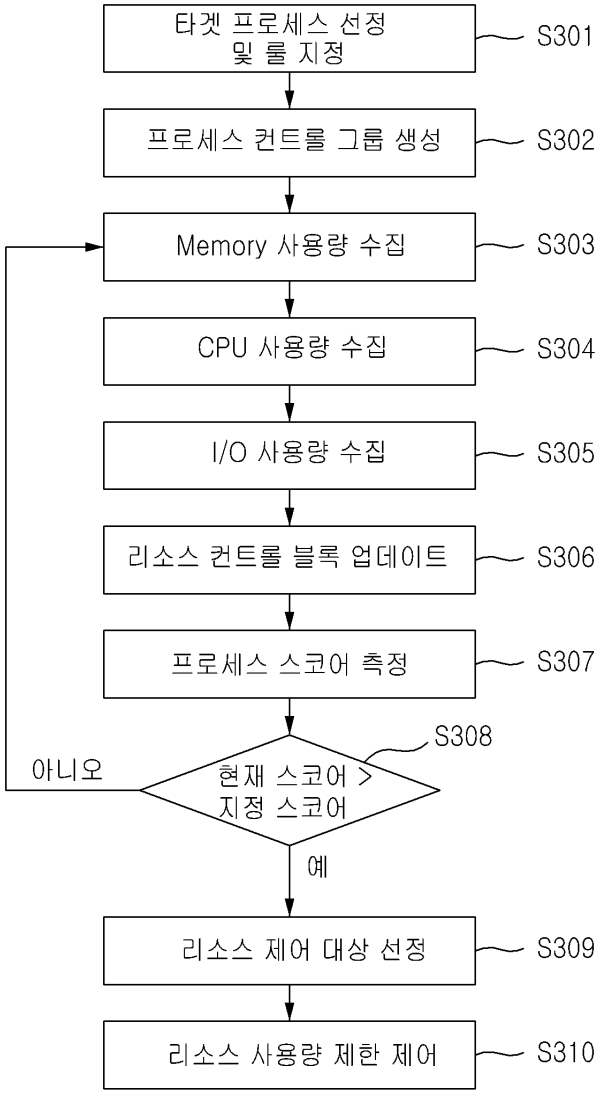
도면1


프로세스
프로세스
프로세스
리눅스 계열 운영체제
물리머신
102
101

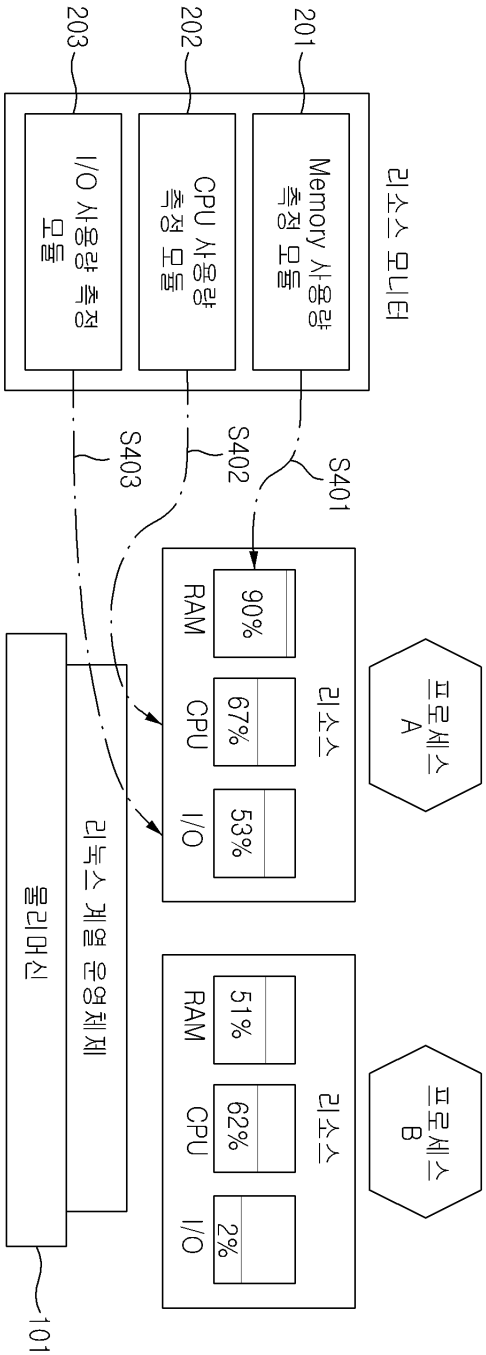
도면2


리소스 모니터
Memory 사용량 측정 모듈
CPU 사용량 측정 모듈
I/O 사용량 측정 모듈
201
202
203
리소스 제어
스코어 측정 모듈
컨트롤 그룹 제어 모듈
리소스 오케스트레이션 모듈
204
205
206

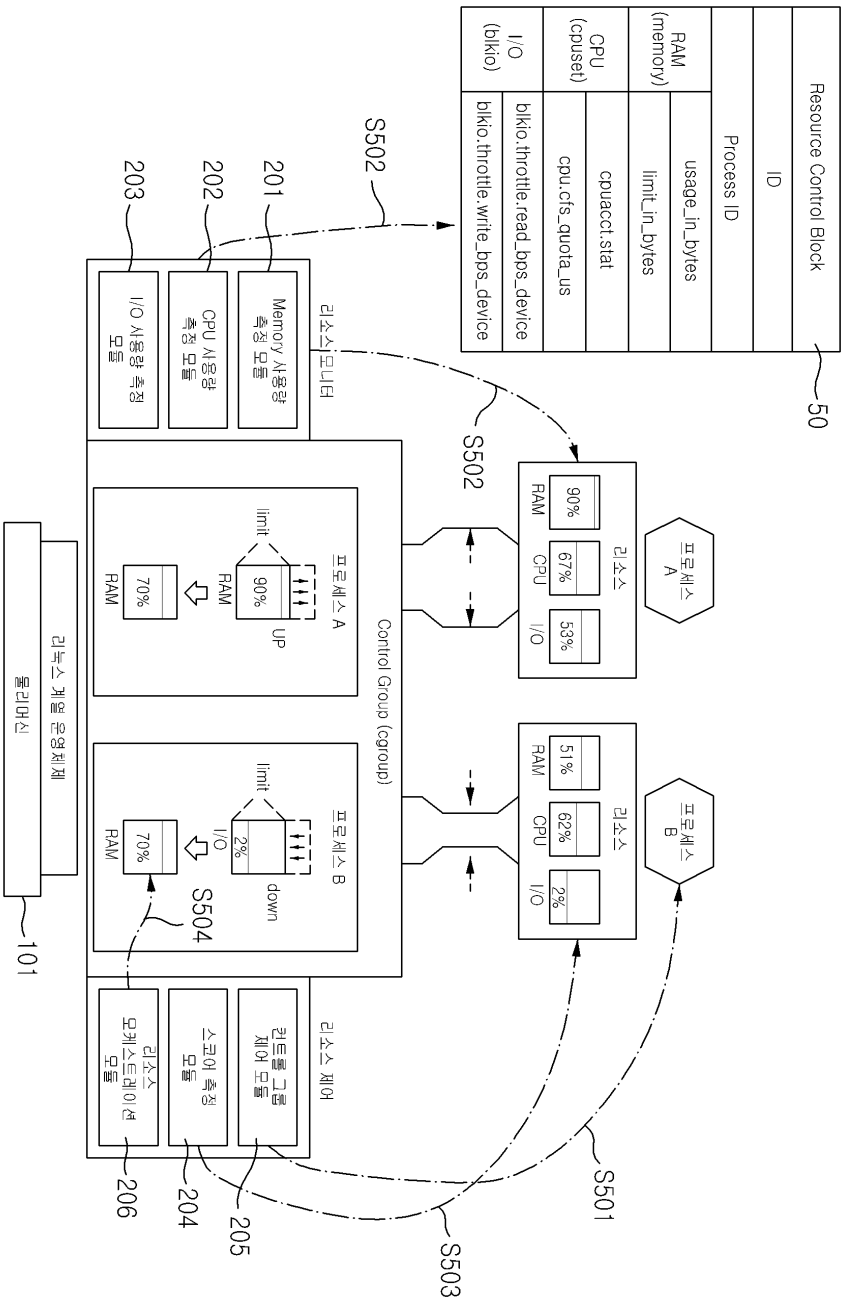
도면3


타겟 프로세스 선정
및 룰 지정
S301
프로세스 컨트롤 그룹 생성
S302
Memory 사용량 수집
S303
CPU 사용량 수집
S304
I/O 사용량 수집
S305
리소스 컨트롤 블록 업데이트
S306
프로세스 스코어 측정
S307
S308
현재 스코어 >
지정 스코어
아니오
예
리소스 제어 대상 선정
S309
리소스 사용량 제한 제어
S310

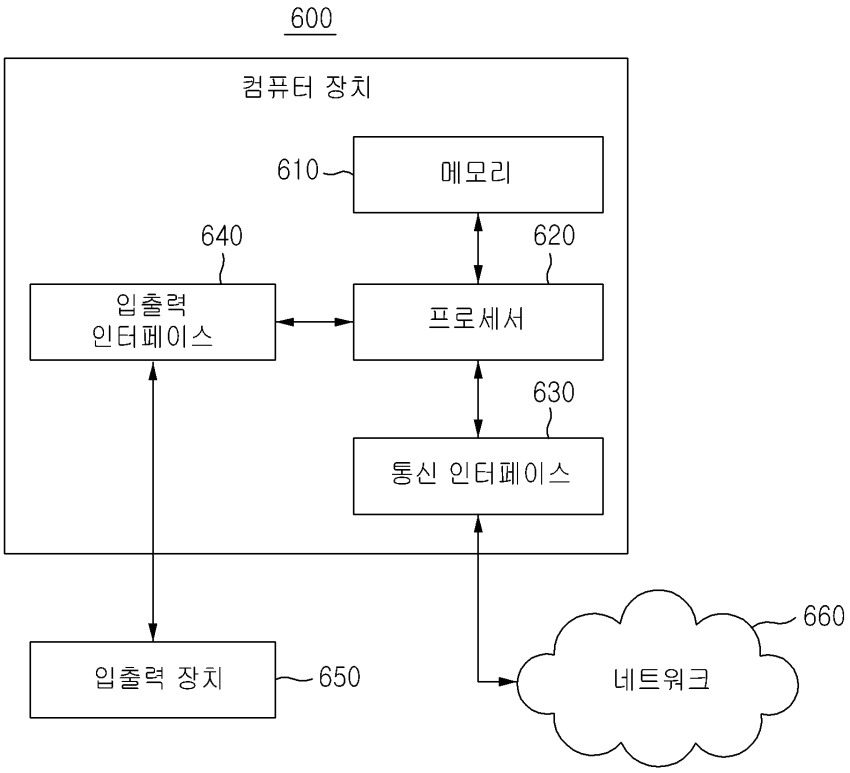
도면4


리소스 모니터
Memory 사용량 측정 모듈
CPU 사용량 측정 모듈
I/O 사용량 측정 모듈
201
202
203
S401
S402
S403
프로세스 A
리소스
RAM
90%
CPU
67%
I/O
53%
프로세스 B
리소스
RAM
51%
CPU
62%
I/O
2%
리눅스 계열 운영체제
물리머신
101

도면5


Resource Control Block
ID
Process ID
RAM (memory)
usage_in_bytes
limit_in_bytes
CPU (cpuset)
cpuacct.stat
cpu.cfs_quota_us
I/O (blkio)
blkio.throttle.read_bps_device
blkio.throttle.write_bps_device
50
S502
S502
201
202
203
Memory 사용량 측정 모듈
CPU 사용량 측정 모듈
I/O 사용량 측정 모듈
리소스 모니터
Control Group (cgroup)
프로세스 A
limit
RAM
90%
UP
RAM
70%
프로세스 B
limit
I/O
2%
down
RAM
70%
S504
리소스
RAM
90%
CPU
67%
I/O
53%
프로세스 A
리소스
RAM
51%
CPU
62%
I/O
2%
프로세스 B
리소스 제어
리소스 모니터링 모듈
스코어 측정 모듈
컨트롤 그룹 제어 모듈
206
204
205
S501
S503
리눅스 계열 운영체제
물리머신
101

도면6


600
컴퓨터 장치
610
메모리
640
620
입출력
인터페이스
프로세서
630
통신 인터페이스
660
입출력 장치
650
네트워크