(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0075830
(43) 공개일자 2015년07월06일

(51) 국제특허분류(Int. Cl.)
G06F 11/30 (2006.01) *G06F 11/34* (2006.01)
(21) 출원번호 10-2013-0164175
(22) 출원일자 2013년12월26일
심사청구일자 2013년12월26일

(71) 출원인
인천대학교 산학협력단
인천광역시 연수구 아카데미로 119, 11호 12 (송도동)
(72) 발명자
박문주
인천광역시 남구 석정로 165 (도화동, 인천대학교 컴퓨터공학과)
주윤덕
경기도 시흥시 조남길 5, 6동 203호 (목감동, 현대빌라)
(74) 대리인
특허법인충정

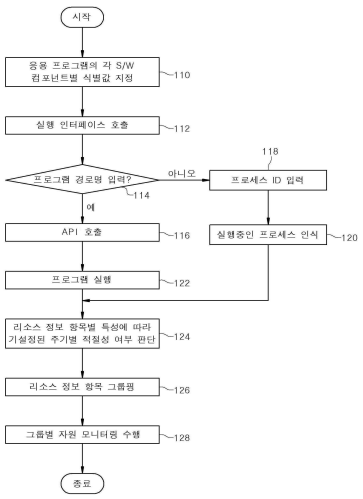
전체 청구항 수 : 총 12 항

(54) 발명의 명칭 **멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법 및 장치**

(57) 요 약

본 발명은 멀티코어 임베디드 리눅스 시스템을 위한 응용프로그램 분석 방법에 있어서, 실행 인터페이스를 통해 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는 과정과, 상기 실행 인터페이스를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하는 과정을 포함함을 특징으로 한다.

대 표 도 - 도1



명 세 서

청구범위

청구항 1

멀티코어 임베디드 리눅스 시스템을 위한 응용프로그램 분석 방법에 있어서,

실행 인터페이스를 통해 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는 과정과,

상기 실행 인터페이스를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하는 과정을 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 2

제1항에 있어서, 상기 자원 모니터링을 수행하는 과정은,

리소스(resource) 정보 항목별 특성에 따라 기설정된 주기별 적절성(suitability) 여부를 판단하여 적어도 둘 이상의 리소스 정보 항목으로 그룹핑을 수행하고, 리눅스 시스템에서 지원하는 모니터링을 위한 규칙에 따라 각 그룹별 리소스 정보 항목에 대한 모니터링을 수행함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 3

제2항에 있어서, 상기 자원 모니터링을 수행하는 과정은,

프로그램의 실행에 대응하는 정책(policy) 정보에 따라 상기 정책 정보의 기설정된 실행 항목별 사용량을 이용률 측정을 위한 자료구조에 누적하고, 상기 리소스 정보 항목별 특성에 따라 그룹핑된 그룹별 기설정된 주기를 상기 실행 항목별 이용률 측정에 적용하여 산출함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 4

제2항에 있어서, 상기 리소스(resource) 정보 항목별 특성에 따라 기설정된 주기는,

정적(static) 리소스(resource) 데이터로 설정된 항목에 대하여 프로그램 실행 중 1회 모니터링이 실시되는 제1 주기,

지피(jiffy) 단위로 발생하는 스레드의 코어 이주를 기반으로 CPU 정보에 대한 1 지피별 모니터링이 실시되는 제2 지피 주기,

상기 리소스 정보 항목별 모니터링의 오버헤드를 고려하여 해당 프로그램이 구동되는 단위인 태스크 스케줄링 데이터로 설정된 항목에 대하여 사용자로부터 정의된 주기로 모니터링이 실시되는 제3 주기를 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 5

제4항에 있어서,

상기 제1 주기는, 시스템의 CPU 거버너(governor), 현재 사용 가능한 코어 개수 및 총 메모리 용량을 나타내는 리소스 정보 항목에 설정되고,

상기 제2 지피 주기는, 프로세스의 처리기 사용량, 해당 지피에서의 CPU 주파수, 문맥전환 횟수를 나타내는 리소스 정보 항목에 설정되고,

상기 제3 주기는, 메모리 사용량, I/O 정보, GPU 사용량 및 기타 CPU 정보를 나타내는 리소스 항목에 설정되어

해당 주기별 리소스 정보 항목별 모니터링이 수행됨을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 6

제3항에 있어서, 상기 기설정된 실행 항목별 사용량은,

해당 프로그램의 코어별 이용율, GPU 이용율, 메모리 점유율 및 네트워크 송수신량을 포함하고,

상기 코어별 이용율은,

기설정된 주기별 프로세스의 스레드별 사용한 누적된 코어별 사용량을 해당 주기에 대한 총 CPU 타임으로 나눔으로써 산출되고,

상기 GPU 이용율은,

GPU의 전원 온/오프 상태를 gpu_power_state 변수에서 읽어 누적하고 기설정된 주기별로 백분율로 환산하여 산출되고,

상기 메모리 점유율 측정은,

proc 파일 시스템으로부터 메모리에 적재된 양에 대한 정보를 획득하여 메모리 측정을 위한 자료구조에 합산하여 바이트(byte) 단위로 누적한 후 다른 응용 프로그램과 공유하는 메모리에 대해서는 공유하는 프로세스의 수로 나누어 계산하고, 기설정된 간격별 메모리에 로드된 누적량의 평균을 전체 메모리 크기로 나눔으로써 산출됨을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법.

청구항 7

멀티코어 임베디스 리눅스 시스템을 위한 응용 프로그램 분석 장치에 있어서,

사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는 실행 인터페이스부와,

상기 실행 인터페이스부를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하도록 제어하는 제어부를 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

청구항 8

제7항에 있어서,

상기 제어부의 제어에 따라 리소스(resource) 정보 항목별 특성에 따라 기설정된 주기별 적절성(suitability) 여부를 판단하여 적어도 둘 이상의 리소스 정보 항목으로 그룹핑을 수행하고, 리눅스 시스템에서 지원하는 모니터링을 위한 규칙에 따라 각 그룹별 리소스 정보 항목에 대한 모니터링을 수행하는 모니터링부를 더 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

청구항 9

제7항에 있어서,

상기 제어부의 제어에 따라 프로그램의 실행에 대응하는 정책(policy) 정보에 따라 상기 정책 정보의 기설정된 실행 항목별 사용량을 이용률 측정을 위한 자료구조에 누적하고, 상기 리소스 정보 항목별 특성에 따라 그룹핑된 그룹별 기설정된 주기를 상기 실행 항목별 이용률 측정에 적용하여 산출하는 리소스 사용량 산출부를 더 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

청구항 10

제7항에 있어서,

상기 제어부의 제어 하에 정적(static) 리소스(resource) 데이터로 설정된 항목에 대하여 프로그램 실행 중 1회 모니터링이 실시되는 제1 주기,

지피(jiffy) 단위로 발생하는 스레드의 코어 이주를 기반으로 CPU 정보에 대한 1 지피별 모니터링이 실시되는 제2 지피 주기,

상기 리소스 정보 항목별 모니터링의 오버헤드를 고려하여 해당 프로그램이 구동되는 단위인 태스크 스케줄링 데이터로 설정된 항목에 대하여 사용자로부터 정의된 주기로 모니터링이 실시되는 제3 주기를 설정하는 주기 설정부를 더 포함함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

청구항 11

제7항에 있어서,

상기 제1 주기는, 시스템의 CPU 거버너(governor), 현재 사용 가능한 코어 개수 및 총 메모리 용량을 나타내는 리소스 정보 항목에 설정되고,

상기 제2 지피 주기는, 프로세스의 처리기 사용량, 해당 지피에서의 CPU 주파수, 문맥전환 횟수를 나타내는 리소스 정보 항목에 설정되고,

상기 제3 주기는, 메모리 사용량, I/O 정보, GPU 사용량 및 기타 CPU 정보를 나타내는 리소스 항목에 설정되어 해당 주기별 리소스 정보 항목별 모니터링을 수행함을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

청구항 12

제9항에 있어서, 상기 리소스 사용량 산출부는,

해당 프로그램의 코어별 이용율, GPU 이용율, 메모리 점유율 및 네트워크 송수신량을 산출하되,

상기 코어별 이용율은,

기설정된 주기별 프로세스의 스레드별 사용한 누적된 코어별 사용량을 해당 주기에 대한 총 CPU 타임으로 나눔으로써 산출되고,

상기 GPU 이용율은,

GPU의 전원 온/오프 상태를 gpu_power_state 변수에서 읽어 누적하고 기설정된 주기별로 백분율로 환산하여 산출되고,

상기 메모리 점유율 측정은,

proc 파일 시스템으로부터 메모리에 적재된 양에 대한 정보를 획득하여 메모리 측정을 위한 자료구조에 합산하여 바이트(byte) 단위로 누적한 후 다른 응용 프로그램과 공유하는 메모리에 대해서는 공유하는 프로세스의 수로 나누어 계산하고, 기설정된 간격별 메모리에 로드된 누적량의 평균을 전체 메모리 크기로 나눔으로써 산출됨을 특징으로 하는 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 멀티코어 통신 처리 서비스에 관한 것으로, 보다 상세하게는 멀티코어 임베디드 시스템상에서 동작하는 리눅스(Linux) 응용의 수행 특성을 분석할 수 있는 수행 분석 도구를 통해 기존의 리눅스 정책보다 성능 및 에너지 소모 측면에서 향상된 수행 기술에 관한 것이다.

배 경 기 술

[0002] 임베디드 시스템을 위한 프로세서의 성능이 높아짐에 따라, 스마트 폰, 태블릿 등을 중심으로 멀티코어 프로세서의 채택이 보편화 되고 있다. 사용자는 멀티코어 프로세서를 사용함으로써 병렬처리를 통한 성능 향상을 기대하나, 임베디드 시스템 상의 응용이 병렬화되어 있지 않다면 오히려 소프트웨어 복잡성을 높이고 시스템의 에너지 사용량을 증가시킬 가능성이 있다. 따라서 멀티코어 프로세서 기반의 임베디드 시스템에서는 멀티코어 프로세서의 처리 능력과 에너지 소모와의 균형을 유지하는 것이 매우 중요한 이슈이다.

[0003] 과거의 임베디드 시스템에서는 단일 처리기 시스템으로 구성되는 경우가 많았으며, 이에 따라 임베디드 시스템

을 위한 응용들은 병렬처리를 고려하지 않은 경우가 대부분이다. 따라서 기존의 응용들은 멀티코어 임베디드 시스템의 하드웨어를 충분히 활용하지 못할 가능성이 높다.

[0004] 반면, 새로 개발되고 있는 응용들은 멀티코어 프로세서나 GPGPU와 같은 현대 임베디드 시스템의 고성능 하드웨어를 고려하여 만들어졌을 수 있다. 이러한 응용의 특성에 따라 시스템에서의 해당 응용의 성능은 달라질 것이다.

[0005] 그러나 운영체제가 응용의 특성을 알고 있지 못하다면 그 특성에 적합한 처리를 하지 못하여 비효율성이 발생하게 되고, 결과적으로 성능의 저하나 고에너지 소모가 발생할 수 있다.

[0006] 멀티코어 임베디드 시스템에서는 응용의 특성에 따른 운영체제의 서비스가 필요하며, 이를 위해서는 응용의 특성을 파악하기 위한 방법이 필요하다. 현재 멀티코어 임베디드 시스템 소프트웨어 개발을 위해 사용할 수 있는 분석도구는 제한적이다.

[0007] 예를 들어, 리눅스 시스템을 위한 기존의 자원 측정 도구로 리눅스에 기본 탑재된 top과 그의 개선된 htop[1]을 대표적으로 들 수 있는데, 이들은 특정 응용에 대한 CPU 코어별 이용률만 확인할 수 없고 전체 응용을 대상으로 한 코어별 이용률만 확인 가능하다. 그 외에는 JTAG 인터페이스를 이용한 하드웨어 디버거를 이용하는 방법들이 있는데, 이러한 장비들은 매우 고가이며 디버깅용 JTAG 인터페이스가 노출되어 있지 않은 경우 사용할 수 없는 단점이 있다

발명의 내용

해결하려는 과제

[0008] 따라서 본 발명은 멀티코어 임베디드 시스템상에서 수행하는 응용의 특성을 분석하기 위한 것으로, 호환성 있는 사용을 위해 운영체제의 수정 없이 분석 도구의 결과를 활용하여 응용의 성능 및 에너지 효율을 향상시키기 위한 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법 및 장치를 제공하고자 한다.

과제의 해결 수단

[0009] 본 발명의 일 견지에 따르면, 멀티코어 임베디드 리눅스 시스템을 위한 응용프로그램 분석 방법에 있어서, 실행 인터페이스를 통해 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는 과정과, 상기 실행 인터페이스를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하는 과정을 포함함을 특징으로 한다.

[0010] 본 발명의 다른 견지에 따르면, 멀티코어 임베디스 리눅스 시스템을 위한 응용 프로그램 분석 장치에 있어서, 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는 실행 인터페이스부와, 상기 실행 인터페이스부를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하도록 제어하는 제어부를 포함함을 특징으로 한다.

발명의 효과

[0011] 본 발명은 멀티코어 임베디드 시스템상에서 동작하는 리눅스(Linux) 응용의 수행 특성을 분석할 수 있는 수행 분석 도구를 통해 기존의 리눅스 정책보다 성능 및 에너지 소모 측면에서 향상된 효과를 발휘한다.

도면의 간단한 설명

[0012] 도 1은 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법에 관한 전체 흐름도.

도 2는 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법이 적용된 요구 거버너와 성능 거버너가 사용되었을 때의 각 CPU 코어의 이용률을 보인 그래프.

도 3a는 기존 리눅스 시스템에서의 벤치마크 결과를 보인 테이블.

도 3b는 본 발명의 분석도구의 결과가 활용된 Firefox 수행 분석 결과를 보인 테이블.

도 4a는 본 발명이 적용된 분석도구의 테스트 결과를 바탕으로 벤치마크를 수행했을 때의 SunSpider 벤치마크 수행결과를 보인 테이블.

도 4b는 본 발명이 적용된 분석도구의 테스트 결과를 바탕으로 벤치마크를 수행했을 때의 Firefox 벤치마크 수행결과를 보인 테이블.

도 5는 본 발명이 적용된 분석도구의 테스트 결과를 바탕으로 벤치마크를 수행했을 때의 Firefox 벤치마크 수행결과에 따른 캐시미스에 대한 테이블.

도 6은 본 발명의 일 실시 예에 따른 멀티코어 임베디스 리눅스 시스템을 위한 응용 프로그램 분석에 관한 상세 블록도.

발명을 실시하기 위한 구체적인 내용

[0013] 이하 본 발명에 따른 바람직한 실시 예를 첨부한 도면을 참조하여 상세히 설명한다. 하기 설명에서는 구체적인 구성 소자 등과 같은 특정 사항들이 나타나고 있는데 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐 이러한 특정 사항들이 본 발명의 범위 내에서 소정의 변형이나 혹은 변경이 이루어질 수 있음은 이 기술 분야에서 통상의 지식을 가진 자에게는 자명하다 할 것이다.

[0014] 본 발명은 멀티코어 통신 처리 서비스에 관한 것으로, 더욱 상세하게는 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 경로명 혹은 프로세스 ID를 입력받아, 상기 경로명에 대응하는 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 리소스(resource) 정보 항목별 특성에 따라 기설정된 주기별 적절성(suitability) 여부를 판단하여 그룹핑을 수행하고, 리눅스 시스템에서 지원하는 모니터링을 위한 규칙에 따라 각 그룹별 리소스 정보 항목에 대한 모니터링을 수행함으로써 멀티코어 임베디드 시스템상에서 동작하는 리눅스(Linux) 응용의 수행 특성을 분석할 수 있는 수행 분석 도구를 통해 기존의 리눅스 정책보다 성능 및 에너지 소모 측면에서 향상된 수행 기술을 제공하고자 한다.

[0015] 이하, 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법에 대해 자세히 살펴보기로 한다.

[0016] 도 1은 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법에 관한 전체 흐름도이다.

[0017] 도 1을 참조하면, 110 과정에서는 응용 프로그램은 단일 프로세스 하에서 프로세스를 관리하는 운영체제 모듈이 상기 응용 프로그램의 각 S/W 컴포넌트별로 구분할 수 있는 식별값을 지정한다.

[0018] 112 과정에서는 실행 인터페이스를 호출하고, 114 과정에서는 상기 실행 인터페이스를 통해 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 입력 여부를 체크한다.

[0019] 이때, 상기 경로명은 리눅스 시스템상에 저장된 리소스의 위치를 식별하기 위한 것으로, 파일 또는 디렉토리의 위치를 식별하는 문자숫자식 기호의 시퀀스이고, 상기 경로명이 파일명에 의해 특정되면 현재의 작업 디렉토리에서 파일을 검색하고, 파일이 상이한 디렉토리에 존재하면, 파일 검색을 위한 경로가 특정된다.

[0020] 상기 114 과정에서의 체크 결과, 경로명이 입력된 경우, 116 과정으로 이동하여 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출한다. 상기 API(application programming interface)는 프로그램 내에서 실행을 위해 특정 서브루틴에 연결을 제공하는 함수를 호출하는 것으로 구현되는 것으로, 이러한 API의 호출을 통해 122 과정에서는 해당 프로그램을 실행한다.

[0021] 이러한 응용 프로그램은 리눅스 운영체제 하에서 프로세스/스레드 관리를 위한 PCB(process control block) 자료구조(구조체)에서와 같이, 하나의 프로세스 내에 다수의 스레드로 구성되어 실행되는 것으로, 시스템 라이브러리(system library), S/W 플랫폼(platform), 3rd party library, 어플리케이션 등의 다양한 S/W 컴포넌트들이 단일 실행파일로 링크되어 단일 프로세스로 실행될 수 있다.

[0022] 124 과정에서는 상기 실행 인터페이스를 통해 경로명 입력에 따라 상기 경로명에 대응하는 응용 프로그램에 대한 자원 모니터링 수행을 위해, 리소스(resource) 정보 항목별 특성에 따라 기설정된 주기별 적절성(suitability) 여부를 판단한다.

[0023] 이후, 126 과정에서 적어도 둘 이상의 리소스 정보 항목으로 그룹핑을 수행하고, 리눅스 시스템에서 지원하는 모니터링을 위한 규칙에 따라 128 과정에서는 각 그룹별 리소스 정보 항목에 대한 모니터링을 수행한다.

[0024] 이러한 경로명을 입력받는 경우에는 본 발명이 적용된 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석의 제어를 위한 제어부(예컨대, 응용 프로그램 분석 도구)에서 직접 응용을 실행시켜 시작과 동시에 모니터링을 시작할 수 있다. 상기 제어부는 리눅스가 제공하는 시스템 정보를 이용하여 프로그램의 수행 특성을 조사한다.

[0025] 이를 기반으로 상기 자원 모니터링 수행은, 프로그램별 실행에 대응하는 정책(policy) 정보에 따라 상기 정책 정보의 기설정된 실행 항목별 사용량을 이용률 측정을 위한 자료구조에 누적하고, 상기 리소스 정보 항목별 특성에 따라 그룹핑된 그룹별 기설정된 주기를 상기 실행 항목별 이용률 측정에 적용하여 산출한다.

[0026] 더욱 상세하게는, 상기 리소스(resource) 정보 항목별 특성에 따라 정적(static) 리소스(resource) 데이터로 설정된 항목에 대하여 프로그램 실행 중 1회 모니터링이 실시되는 제1 주기, 지피(jiffy) 단위로 발생하는 스레드의 코어 이주를 기반으로 CPU 정보에 대한 1 지피별 모니터링이 실시되는 제2 지피 주기, 상기 리소스 정보 항목별 모니터링의 오버헤드를 고려하여 해당 프로그램이 구동되는 단위인 태스크 스케줄링 데이터로 설정된 항목에 대하여 사용자로부터 정의된 주기로 모니터링이 실시되는 제3 주기를 설정한 후, 상기 제1 주기로 설정된 1) 시스템의 CPU 거버너(governor)/sys/devices/system/cpu/cpu0/scaling_governor, 2) 현재 사용 가능한 코어 개수/sys/devices/system/cpu/possible, 3) 총 메모리 용량/proc/meminfo 데이터들을 모니터링한다.

[0027] 그리고, 제2 주기로 설정된 1) 프로세스의 처리기 사용량/프로세스의 스레드가 사용한 코어 및 이용량 /proc/[pid]/task/[tid]/stat, 각 코어의 사용시간 /proc/stat 2) 해당 지피에서의 CPU 주파수 /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freu 및 3) 문맥전환 횟수 데이터들을 모니터링한다. 스레드의 코어 이주(migration)가 지피 단위로 발생하므로 1 지피 마다 CPU 정보를 조사한다. [pid]는 수행 프로세스의 PID 번호를, [tid]는 스레드의 TID 번호를 나타낸다.

[0028] 또한, 제3 주기로 설정된 1) 메모리 사용량/proc/[pid]/task/[tid]/statm, 2)I/O 정보; read/write 시스템 호출 수 및 데이터 양/proc/[pid]/task/[tid]/io, 네트워크 송수신량/proc/self/dev, 3) GPU 사용량 및 4) 기타 CPU 정보; 스레드들의 코어 이주 횟수/proc/[pid]/task/sched, 스레드의 코어 친화도(affinity) /proc/[pid]/task/[tid]/status 데이터들을 모니터링 한다. 이는 기타 정보로서 시스템상 모니터링 오버헤드를 고려하여 100ms 이상의 간격으로 조사한다.

[0029] 또한, 상기 기설정된 실행 항목별 사용량은, 해당 프로그램의 코어별 이용율, GPU 이용율, 메모리 점유율 및 네트워크 송수신량을 포함한다.

[0030] 상기 코어별 이용율은, 기설정된 주기별 프로세스의 스레드별 사용한 누적된 코어별 사용량을 해당 주기에 대한 총 CPU 타임으로 나눔으로써 산출되는 것으로, /proc 파일 시스템은 프로세스마다 각각의 스레드가 CPU를 사용한 시간이 지피 단위로 저장되어 이를 이용하여 매 지피마다 각 스레드가 사용한 CPU 시간과 어떤 코어에서 CPU 시간을 소모했는 지에 대한 정보를 읽어 코어별 사용량을 저장하기 위한 자료구조에 누적한다. 타겟 시스템에서 1 지피는 10ms로 설정되어 측정간격마다 프로세스의 스레드들이 사용한 누적된 코어별 사용량을 해당 간격에 대한 총 CPU time으로 나눔으로써 프로세스의 코어별 이용률을 구한다.

[0031] 이때, 프로세스가 자식 프로세스를 만드는 경우에는 부모 프로세스의 PID, 즉 PPID를 조사하여 CPU 사용량에 포함시킨다. /proc/[pid]/stat 파일에서 PPID를 찾을 수 있다. 이를 위해서 분석 대상 프로세스의 PID보다 큰 숫자의 PID를 가지는 프로세스에 대해 PPID를 검색해야 하는 오버헤드가 발생한다.

[0032] 상기 GPU 이용율은, GPU의 전원 온/오프 상태를 gpu_power_state 변수에서 읽어 누적하고 기설정된 주기별로 백분율로 환산하여 산출된다.

[0033] 상기 메모리 점유율 측정은, proc 파일 시스템으로부터 메모리에 적재된 양에 대한 정보를 획득하여 메모리 측정을 위한 자료구조에 합산하여 바이트(byte) 단위로 누적한 후 다른 응용 프로그램과 공유하는 메모리에 대해서는 공유하는 프로세스의 수로 나누어 계산하고, 기설정된 간격별 메모리에 로드된 누적량의 평균을 전체 메모리 크기로 나눔으로써 산출된다.

[0034] 상기 네트워크 송수신량 측정은, /proc 파일 시스템에 누적되는 송수신량을 측정 간격단위로 읽어 온 후, 이전 값과의 차를 네트워크 사용량 측정을 위한 자료구조에 누적한다.

[0035] 한편, 114 과정에서의 체크 결과, 프로그램 경로 명 입력이 아닌 경우 118 과정으로 이동하여 프로세스 ID 입력을 확인하고, 120 과정에서 현재 실행 중인 적어도 하나의 프로세스를 인식하고, 인식된 프로세스에 대하여 124 과정으로 이동하여 이후의 동작을 수행한다.

[0036] 여기서, 도 2를 참조하면, 도 2는 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법이 적용된 요구 거버너와 성능 거버너가 사용되었을 때의 각 CPU 코어의 이용률을 그래프화하여 도시한 것으로, Firefox는 약 30개 이상의 스레드를 생성하지만, 실험 결과 동시에 수행되는 최대 스레드 수는 3개였다. 따라서 Firefox는 4개 코어를 가진 CPU를 모두 활용하지는 못한다.

[0037] 이때, 도 3a에 도시된 테이블은 각각의 거버너가 사용되었을 때 에너지 소모와 성능을 보였다. SunSpider 벤치마크에서의 성능은 처리시간이므로, 시간이 작을수록 성능이 높은 것이다. 따라서 에너지 효율은 에너지 소모와 성능의 곱(Joule-second)에 비례한다. 요구 거버너에서는 약 5153.3 J·s, 성능 거버너에서는 3347.4 J·s의 Joule-second를 가지며, 따라서 성능 거버너가 소요되는 에너지에 비해 더 높은 성능을 보이고 있다.

[0038] 도 3b의 테이블은 응용 프로그램 분석도구로 분석된 Firefox의 수행 내용을 보인 것으로, 동시에 수행되는 스레드의 최대 수는 3개였지만, 실제로 최대 CPU 이용률은 주파수에 관계없이 100% 이하임을 볼 수 있으며, 따라서 현재 버전의 Firefox의 병렬성은 멀티코어 시스템을 활용하기에 충분하지 않다. 시스템의 코어 수는 4개이나, 시스템 전체의 최대 CPU 이용률도 평균 100% 이하, 최대 213% 이하임을 볼 수 있다. 최고 속도로 수행하는 성능 거버너에서도 전체 코어를 활용하지 못하고 있다. 따라서 이 응용은 멀티코어 CPU를 충분히 활용하지 못하고 있으며, 실제로는 하나의 코어에서 수행해도 동등한 성능을 보일 것으로 예상된다. 오히려 많은 코어를 사용함으로써 코어 이주의 오버헤드가 발생하고, 이로 인한 캐시의 성능 저하가 있었을 것이다. 또한 측정 결과, ARM용의 Firefox에서는 GPU를 사용하지 않고 있었다.

[0039] 이를 기반으로 다음과 같은 설정을 하여 Firefox를 이용한 SunSpider 벤치마크 테스트를 실시하였다.

[0040] 최대 시스템 CPU 이용률이 213% 이므로, On-line 코어의 수는 2개로 한다. 나머지 2개 코어는 hot-plugging 기능을 이용하여 오프라인에서 설정한다.

[0041] 응용의 최대 CPU 이용률이 100% 이하이므로, Firefox 응용은 CPU1 코어에 고정 할당하여 코어이주를 허용하지 않는다.

[0042] 도 4a에 도시된 테이블은 위와 같은 설정으로 벤치마크를 수행했을 때의 SunSpider 벤치마크 수행 결과를 보여준다. 도 4a의 테이블에서와 같이 한 개의 코어에 고정하여 수행시킨 결과 더 적은 에너지 소모로 더 높은 성능을 얻을 수 있다. Joule-second로 계산해 보면, 요구 거버너를 사용한 경우는 약 4592.6 J·s, 성능 거버너를 사용한 경우는 약 3142.9 J·s로 각각 약 10.88%와 약 6.11%가 감소하여 소모 에너지 대비 더 높은 성능을 보임을 알 수 있다.

[0043] 도 4b에 도시된 테이블은 분석 결과가 적용된 시험에서 수행된 Firefox의 벤치마크 테스트를 분석도구로 분석한 결과를 보여준다.

[0044] 도 4b에 도시된 바와 같이, 프로그램의 수행 코어를 고정하여 스레드의 이주를 제거함으로써 전체 시스템의 CPU 이용률은 오히려 낮아졌으며, 메모리 사용량도 감소하였다. 또한 이주를 하지 않음으로써 캐시미스가 감소함으로 인한 성능 향상을 예상할 수 있는데, 실제 측정 결과는 도 5에 도시된 테이블과 같다.

[0045] 캐시미스는 프로그램만의 수치를 측정할 수 없어 전체 시스템의 수치를 측정하였다. 캐시미스율을 비교하지 않은 이유는, 두 실행에서의 수행 시간 자체가 달라 시스템의 전체 수행 명령어의 수가 다르기 때문이다. 캐시미스는 프로그램만의 수치를 측정할 수 없어 전체 시스템의 수치를 측정하였다. 캐시미스율을 비교하지 않은 이유는, 두 실행에서의 수행 시간 자체가 달라 시스템의 전체 수행 명령어의 수가 다르기 때문이다. 캐시미스 회수를 비교하면, 수행 코어를 고정했을 때 DCache load/store, ICache에서 각각 17.85%, 19.12%, 6.97%의 miss가 감소했음을 알 수 있다.

[0046] 이상에서는, 도 1을 참조하여 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로

그램 분석 방법에 대해 살펴보았다.

[0047] 이하, 본 발명의 일 실시 예에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 장치에 대해 도 6을 참조하여 살펴보기로 한다.

[0048] 도 6은 본 발명의 일 실시 예에 따른 멀티코어 임베디스 리눅스 시스템을 위한 응용 프로그램 분석에 관한 상세 블록도이다.

[0049] 도 6을 참조하면, 본 발명이 적용된 장치(600)은 실행 인터페이스부(610), 시스템 메모리(612), 제어부(616), 정책 정보부(618), 리소스 사용량 산출부(620), 주기 설정부(622) 및 모니터링부(624)를 포함한다.

[0050] 상기 실행 인터페이스부(610)은 사용자로부터 모니터링하고자 하는 프로그램의 위치 기반 경로 탐색을 위한 기설정된 타입의 경로명 혹은 프로세스 ID를 입력받는다.

[0051] 상기 시스템 메모리(612)는 휘발성(RAM 등), 비휘발성(ROM, 플래쉬 메모리 등), 또는 2개의 조합일 수 있다. 시스템 메모리(612)는 일반적으로 구동 시스템(614), 하나 이상의 프로그램 모듈을 포함하고, 프로그램 데이터를 포함할 수 있다.

[0052] 상기 제어부(616)는 상기 실행 인터페이스부(610)를 통해 경로명 및 프로세스 ID 중 어느 하나의 입력 여부에 따라 상기 경로명에 대응하는 프로그램의 API를 순차적으로 호출하여 해당 프로그램을 실행하거나 혹은 리눅스 시스템에서 현재 실행 중인 적어도 하나의 프로세스에 대한 자원 모니터링을 수행하여 응용 프로그램에 대한 자원 모니터링을 수행하도록 제어한다.

[0053] 상기 모니터링부(624)는 제어부(616)의 제어에 따라 리소스(resource) 정보 항목별 특성에 따라 기설정된 주기별 적절성(suitability) 여부를 판단하여 적어도 둘 이상의 리소스 정보 항목으로 그룹핑을 수행하고, 리눅스 시스템에서 지원하는 모니터링을 위한 규칙에 따라 각 그룹별 리소스 정보 항목에 대한 모니터링을 수행한다.

[0054] 상기 리소스 사용량 산출부(620)는 제어부(616)의 제어 하에 프로그램의 실행에 대응하는 정책(policy) 정보부의 정책 정보에 따라 상기 정책 정보의 기설정된 실행 항목별 사용량을 이용률 측정을 위한 자료구조에 누적하고, 상기 리소스 정보 항목별 특성에 따라 그룹핑된 그룹별 기설정된 주기를 상기 실행 항목별 이용률 측정에 적용하여 산출한다.

[0055] 상기 주기 설정부(622)는 상기 제어부(622)의 제어 하에 정적(static) 리소스(resource) 데이터로 설정된 항목에 대하여 프로그램 실행 중 1회 모니터링이 실시되는 제1 주기, 지피(jiffy) 단위로 발생하는 스레드의 코어 이주를 기반으로 CPU 정보에 대한 1 지피별 모니터링이 실시되는 제2 지피 주기 및 상기 리소스 정보 항목별 모니터링의 오버헤드를 고려하여 해당 프로그램이 구동되는 단위인 태스크 스케줄링 데이터로 설정된 항목에 대하여 사용자로부터 정의된 주기로 모니터링이 실시되는 제3 주기를 각각 설정한다.

[0056] 여기서, 상기 제1 주기는, 시스템의 CPU 거버너(governor), 현재 사용 가능한 코어 개수 및 총 메모리 용량을 나타내는 리소스 정보 항목에 설정되고, 상기 제2 지피 주기는, 프로세스의 처리기 사용량, 해당 지피에서의 CPU 주파수, 문맥전환 횟수를 나타내는 리소스 정보 항목에 설정되고, 상기 제3 주기는, 메모리 사용량, I/O 정보, GPU 사용량 및 기타 CPU 정보를 나타내는 리소스 항목에 설정된다.

[0057] 이어서, 상기 리소스 사용량 산출부(620)는 해당 프로그램의 코어별 이용율, GPU 이용율, 메모리 점유율 및 네트워크 송수신량을 산출하되, 상기 코어별 이용율은, 기설정된 주기별 프로세스의 스레드별 사용한 누적된 코어별 사용량을 해당 주기에 대한 총 CPU 타임으로 나눔으로써 산출되고, 상기 GPU 이용율은, GPU의 전원 온/오프 상태를 gpu_power_state 변수에서 읽어 누적하고 기설정된 주기별로 백분율로 환산하여 산출되고, 상기 메모리 점유율 측정은, proc 파일 시스템으로부터 메모리에 적재된 양에 대한 정보를 획득하여 메모리 측정을 위한 자료구조에 합산하여 바이트(byte) 단위로 누적한 후 다른 응용 프로그램과 공유하는 메모리에 대해서는 공유하는 프로세스의 수로 나누어 계산하고, 기설정된 간격별 메모리에 로드된 누적량의 평균을 전체 메모리 크기로 나눔으로써 산출된다.

[0058] 상기와 같이 본 발명에 따른 멀티코어 임베디드 리눅스 시스템을 위한 응용 프로그램 분석 방법 및 장치에 관한 동작이 이루어질 수 있으며, 한편 상기한 본 발명의 설명에서는 구체적인 실시 예에 관해 설명하였으나 여러 가지 변형이 본 발명의 범위를 벗어나지 않고 실시될 수 있다. 따라서 본 발명의 범위는 설명된 실시 예에 의하여 정할 것이 아니고 청구범위와 청구범위의 균등한 것에 의하여 정하여져야 할 것이다.

References

[1] An interactive process viewer for Linux, [Online]. Available: http://htop.sourceforge.net (2013, Sep 5)

[2] Y. Ju and M. Park, A Resource Usage Monitoring Tool for Analysis of Application Execution Patterns on Multi-core Embedded Systems, *Proc. of the 39th KIISE Fall Conference*, vol.39, no.2(A), pp.171-173, 2012. (in Korean)

[3] M. Weiser, B. Welch, A.J. Demer, and S. Shenker, Scheduling for Reduced CPU Energy, *Proc. of the 1st USENIX Conf. on Operating Systems Design and Implementation*, pp.13-23, 1994.

[4] SunSpider JavaScript Benchmark, [Online]. Available: http://www.webkit.org/perf/sunspider-0.9/sunspider-driver.html (2013, Sep. 5)

[5] V. Pallipadi and A. Starikovskiy, The Ondemand Governor, *Proc. of the Linux Symposium*, vol.2, pp.215-230, 2006.

[0059]

부호의 설명

[0060]

610: 실행 인터페이스부 612: 시스템 메모리

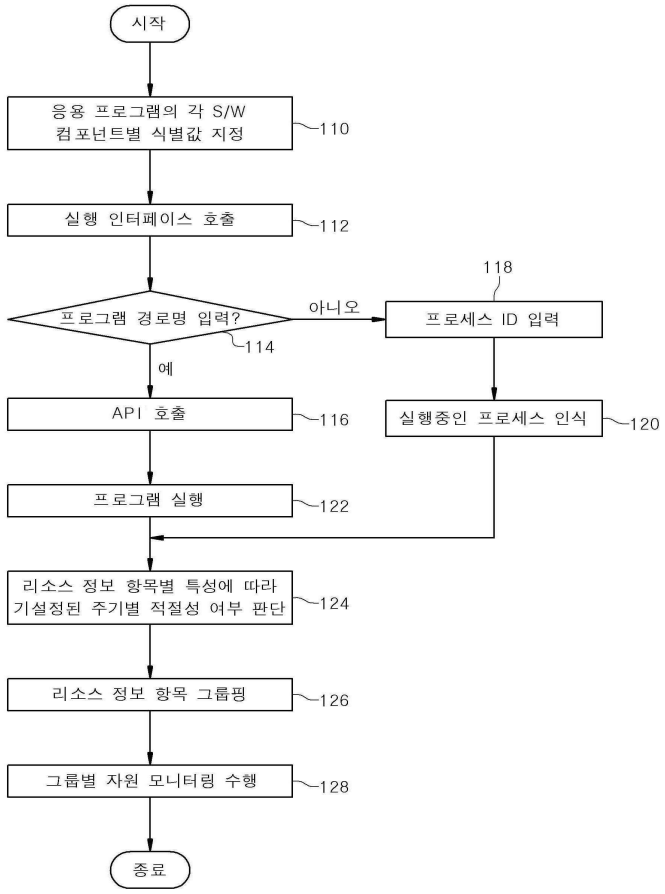
616: 제어부 618: 정책 정보부

620: 리소스 사용량 산출부 622: 주기 설정부

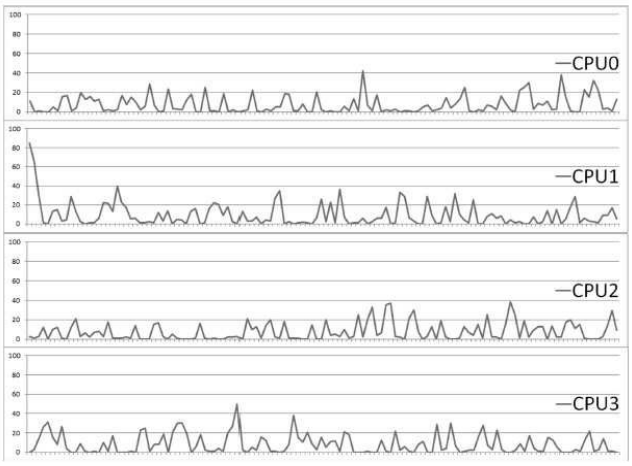
624: 모니터링부

도면

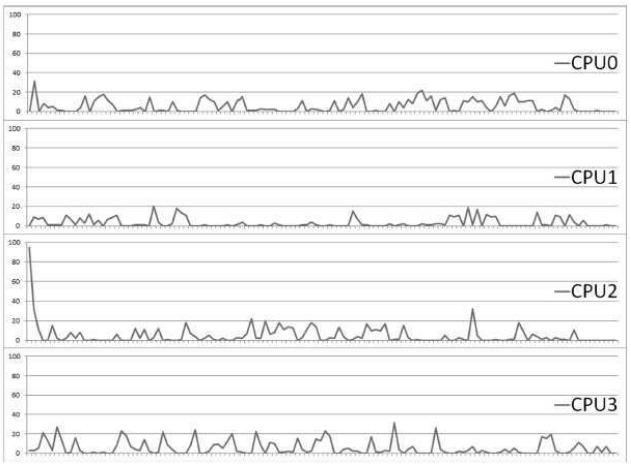
도면1


시작
응용 프로그램의 각 S/W 컴포넌트별 식별값 지정
110
실행 인터페이스 호출
112
프로그램 경로명 입력?
114
아니오
118
프로세스 ID 입력
예
API 호출
116
실행중인 프로세스 인식
120
프로그램 실행
122
리소스 정보 항목별 특성에 따라 기설정된 주기별 적절성 여부 판단
124
리소스 정보 항목 그룹핑
126
그룹별 자원 모니터링 수행
128
종료

도면2a


CPU0
CPU1
CPU2
CPU3

도면2b



도면3a

Governor	Ondemand	Performance
Energy consumption	2849.4 J	3041.4 J
SunSpider benchmark result	1808.6 ms	1100.6 ms

도면3b

Governor	Ondemand	Performance
Average CPU utilization	23.9%	13.6%
Max. CPU utilization	99.0%	99.0%
Average CPU utilization of the system	82.8%	40.6%
Max. CPU utilization of the system	170.0%	213.0%
Average memory uasage	8.72%	8.67%
Max. memory usage	11.07%	10.48%
Migration	2,603회	1,860회
file system read	0 KB	0 KB
file system write	10,406KB	10,304KB
Network RX	3619.4KB	3566.2KB
Network TX	255.6KB	205.4KB

도면4a

Governor	Ondemand	Performance
Energy consumption	2823.1 J (-0.92%)	2985.8 J (-1.83%)
SunSpider benchmark result	1626.8 ms (-10.05%)	1052.6 ms (-4.36%)

도면4b

Governor	Ondemand	Performance
Average CPU utilization	21.5%	13.8%
Max. CPU utilization	87.1%	96.0%
Average CPU utilization of the system	78.9%	72.3%
Max. CPU utilization of the system	136.7%	169.3%
Average memory uasage	8.56%	8.30%
Max. memory usage	10.59%	10.23%
Migration	0회	0회
file system read	0 KB	0 KB
file system write	13,032KB	11,048KB
Network RX	3615.8KB	3706.9KB
Network TX	248.9KB	224.8KB

도면5

		Original	Changed
DCache	load miss	253,207,079	208,018,640
	store miss	251,326,538	203,283,456
ICache miss		661,448,749	615,377,284

도면6

