(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0060227
(43) 공개일자 2018년06월07일

(51) 국제특허분류(Int. Cl.)
G06F 7/58 (2006.01)
(52) CPC특허분류
G06F 7/588 (2013.01)
(21) 출원번호 10-2016-0159478
(22) 출원일자 2016년11월28일
심사청구일자 없음

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
홍석희
서울특별시 노원구 덕릉로73길 28, 206동 2304호 (중계동, 양지대림아파트)
김성겸
서울특별시 송파구 삼학사로12길 8, 502호(석촌동)
(뒷면에 계속)
(74) 대리인
특허법인충정

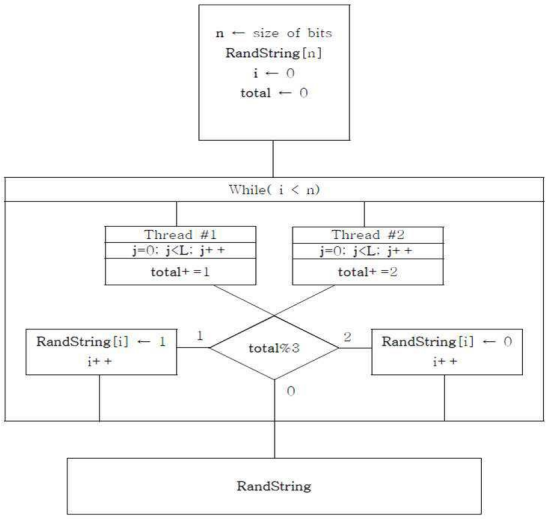
전체 청구항 수 : 총 1 항

(54) 발명의 명칭 **멀티코어 마이크로프로세서 기반 소프트웨어에서 발생하는 경쟁상태를 이용한 난수발생 기술**

(57) 요 약

현재 많은 시스템에서 채택하고 있는 멀티코어 시스템에서 멀티 스레드 애플리케이션을 통해 발생하는 비결정론적 상황인 데이터 경합(data-race)을 통해 발생하는 동시성 오류(concurrency error)에 의해 난수를 발생하는 애플리케이션 혹은 장치를 제공할 수 있다.

대 표 도 - 도3



(72) 발명자

박건민

서울특별시 강남구 개포로 516, 608동 1305호(개포동, 주공아파트)

이승준

서울특별시 성북구 안암로9가길 35, 301호 (안암동5가)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711026432
부처명	미래창조과학부
연구관리전문기관	정보통신기술진흥센터
연구사업명	대학ICT연구센터 육성지원사업
연구과제명	무결점 보안을 위한 지상/위성 양자통신 기술 개발
기 여 율	1/1
주관기관	고려대학교 산학협력단
연구기간	2016.01.01 ~ 2016.12.31

명 세 서

청구범위

청구항 1

난수를 생성하는 방법에 있어서,

멀티코어 시스템에서 멀티 스레드 애플리케이션을 통해 데이터 경합을 유도하는 제1 단계;

공유 메모리를 참조하여 상기 데이터 경합을 관측하는 제2 단계;

비결정적 상황에 대한 비트를 출력하여 상기 데이터 경합 결과를 가공하는 제3 단계; 및

상기 제1 내지 제3 단계를 반복하여 상기 난수를 생성하는 제4 단계; 를 포함하는 것을 특징으로 하는 난수 생성 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 멀티코어 마이크로프로세서 시스템 상에서 발생할 수 있는 비결정론적인 상황을 유도하여 관측(Detect)하고 가공하여 난수로 이용하는 기술에 관한 것이다.

배 경 기 술

[0002] **난수**

[0003] 난수(random number)는 생성되기 전에는 예측할 수 없는 수를 뜻한다. 예를 들어 어떤 수가 {0, …, 2^{n-1}} 사이의 난수라고 가정하면, 이 수는 2^{-n}보다 높은 확률로 예측할 수 없다. 또한 m개의 난수가 있다면, 다른 m-1개의 난수를 모두 알고 있다 하더라도, 나머지 m번째 수를 2^{-n}보다 큰 확률로 예측할 수 없다. 난수는 암호학적으로 대칭키 암호 알고리즘의 비밀키, 스트림 암호 알고리즘의 초기화벡터, 비대칭키 암호 알고리즘 RSA의 큰 소수등을 생성할 때 사용된다. 이와 같이 암호 알고리즘이나 프로토콜등은 난수의 사용이 가능하다는 가정 하에서 설계된다. 그러므로 난수를 생성하는 과정의 안전성에 결함이 있다면 이는 암호 알고리즘 자체의 안전성에 영향을 미친다.

[0005] **비결정론적 난수발생기**

[0006] 비결정론적 난수 발생기는 물리적 현상의 임의성을 이용하는 것이다. 이와 같은 발생기는 참난수와 마찬가지로 매우 높은 엔트로피를 가진다는 장점이 있는 반면, 높은 엔트로피의 잡음을 얻는 것이 쉽지 않고 다양한 플랫폼에 적용하기가 어렵다는 단점이 있다.

[0008] **멀티코어 프로세서**

[0009] 멀티코어 CPU는 두 개 이상의 독립코어를 단일 집적회로로 이루어진 하나의 패키지로 통합한 것이다. 칩 레벨 멀티프로세서(CMP)라고도 한다. 현재 주로 사용되는 멀티 코어 시스템 구조는 동종(Homogeneous)의 다수 코어를 사용하여 공유메모리를 갖추고 있는 시스템으로서, 각 코어들은 운영체제(operating system, OS)의 도움을 받아 서로 다른 명령을 각각 처리할 수 있다. 도 1은 듀얼 CPU 코어 칩의 예시적인 도면이다.

[0011] **데이터 경합(data-race)**

[0012] 데이터 경합은 어떤 공유 데이터가 두 스레드 이상에게서 동시에 접근되는 상황을 말한다. 이때, 코어들에서 스레드가 데이터 경합을 할 경우 어떤 코어가 우선시 되어서 공유 데이터에 접근할지 알 수 없으며, 회로의 공정상 변이 및 여러 환경요인에 의해 결정된다.

발명의 내용

해결하려는 과제

[0013] 본 발명이 해결하고자 하는 과제는 현재 많은 시스템에서 채택하고 있는 멀티코어 시스템에서 멀티 스레드 애플리케이션을 통해 발생하는 비결정론적 상황인 데이터 경합(data-race)을 통해 발생하는 동시성 오류(concurrency error)에 의해 난수를 발생하는 애플리케이션 혹은 장치를 제공하는 것이다.

발명의 효과

[0014] 본 발명에 의하면, 현재 저전력, 고사양의 CPU를 요구하는 소형 디바이스들에서도 멀티코어 시스템을 채택한다면 생길 수 있는 비결정론적인 상황을 손쉽게 관측하여 난수를 생성할 수 있다. 이는 암호 알고리즘 등에 사용할 비밀키의 안전성을 높일 수 있는 효과를 볼 수 있다.

도면의 간단한 설명

[0015] 도 1은 공유메모리를 갖는 듀얼 CPU 코어 칩의 예시적인 도면이다.

도 2는 데이터 경합의 유도를 도시하는 예시적인 도면이다.

도 3은 본 발명의 실시예에 따른 난수 발생 알고리즘을 설명하기 위한 도면이다.

도 4a 및 도 4b는 생성된 난수가 난수성을 만족한다는 것을 설명하기 위한 그래프이다.

도 5는 난수의 분산 정도를 나타내는 테스트 결과를 도시하는 도면이다.

도 6은 NIST 제공의 ENT 프로그램 결과를 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0016] **데이터 경합(data-race)의 유도**

[0017] 연산에 있어서 동일한 개수의 Instruction을 갖지만 다른 역할을 하는 스레드 A, B를 멀티코어 시스템의 각기 다른 물리적인 코어에서 실행 할 수 있도록 생성한다. 이 스레드 A, B는 공유메모리의 동일한 부분을 참조 하고 있다. 스레드 A, B가 공유메모리에 여기서 물리적으로 다르지만 동일한 연산속도를 가진 코어는 공유메모리에 자신의 연산 결과 값을 쓰는 과정에서 스레드 A, B중 하나의 명령이 무시된다. 여기서 두 코어는 서로 동형(Homogeneous)이고, 스레드 A B는 서로 동일한 Instruction수를 가지고 있기 때문에 어느 코어의 명령이 우선이 될지 실행하기 전에는 알 수 없다. (도 2 참조)

[0018] 운영체제(Operating System, OS)의 경우, 스케줄러가 스레드들의 배치를 하기 때문에 데이터 경합을 유도하기 위해 스레드 A,B의 명령을 각각 동일한 횟수만큼 하도록 한다.

[0020] **데이터 경합(data-race)의 관측**

[0021] 도 2에서 볼 수 있듯이 데이터 경합 이후에 공유메모리를 참조하여 데이터 경합의 결과를 알 수 있다.

[0023] **데이터 경합(data-race)결과의 가공**

[0024] 데이터 경합이후에 공유데이터를 가공하여 비결정적인 상황에 대해서 각각 50:50의 확률로 0,1의 비트를 출력한다.

[0026] **난수의 생성**

[0027] 사용자가 원하는 비트의 크기만큼 데이터 경합(data-race)의 유도, 관측, 가공을 반복하여 원하는 비트 크기의 난수를 생성하도록 한다.

[0029] 본 발명의 일 실시예에 따른 난수 발생 알고리즘은 도 3에 도시된 바와 같다.

[0030] While 구문 내에서 스레드 #1와 스레드 #2가 다른 작업을 L번 반복하면서 경합 상태를 유도한다. 이에 따라서 공유데이터 total에 각각 L번을 참조하는 과정에서 데이터 경합(data-race)를 유도한다. 스레드 #1와 스레드 #2가 모두 종료된수 공유데이터 total%3의 값을 계산하여 1이면 비트 1을 2이면 비트 0을 출력한다. 만약, total%3의 값이 0일 경우에는 비트를 출력하지 않고 데이터 경합(data-race)를 다시 유도하여 결과 데이터를 가공한다.

[0032] **공유 데이터 가공과 난수성에 만족하는 비트 출력의 증명**

[0033] X : Thread #1 가 1 ≡ 1 (mod 3)를 더하는 횟수 (x∈{0, 1, …, L})

[0034] Y : Thread #2 가 2 ≡ 2 (mod 3)를 더하는 횟수 (y∈{0, 1, …, L})

[0035] P(X)=P(Y)

[0037] 스레드 #1와 #2가 공유데이터에 자신의 작업의 영향을 미칠 확률은 동일하다. 이곳에서 동종(Homogeneous)의 멀티코어 환경이 필요하다. 각각의 스레드가 동종(Homogeneous)의 멀티코어에 배치되어 실행되면, 동일한 확률로 데이터 경합(data-race)에서 우선된다. 따라서 P(X)=P(Y)이다.

[0038] 여기서 확률변수 Z=X-Y(≡X+2Y=공유데이터 결과값 mod3)로 정의하면

[0039] P(Z)=P(-Z) (z∈{-L,…, -1, 0, 1, …, L})

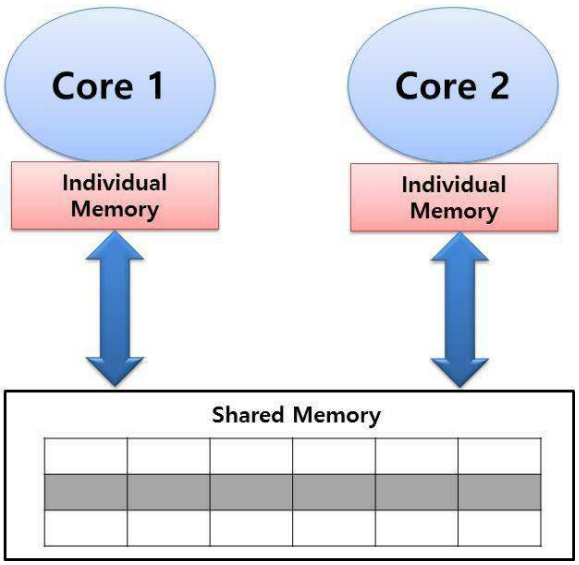
[0040] 이다.(즉, P(z=0)기준으로 확률 분포가 좌우대칭이다.)

[0041] 도 4a에서 검은 점은 Z mod 3의 값이 0이, 파란점은 1, 빨간점은 1이 될 확률 값들이 된다. 이것들을 모으면, 도 4b와 같이 Z=total mod3인 확률 값에서 P(total≡1 mod 3)=P(total≡2 mod 3)이라고 증명할 수 있다.

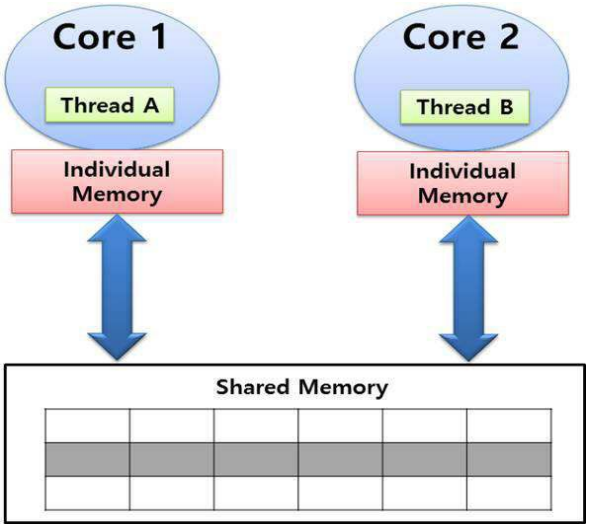
[0042] 도 5는 난수의 분산 정도를 나타내는 테스트 결과이며, 도 6은 NIST 제공의 ENT 프로그램 결과를 나타내는 도면이다.

도면

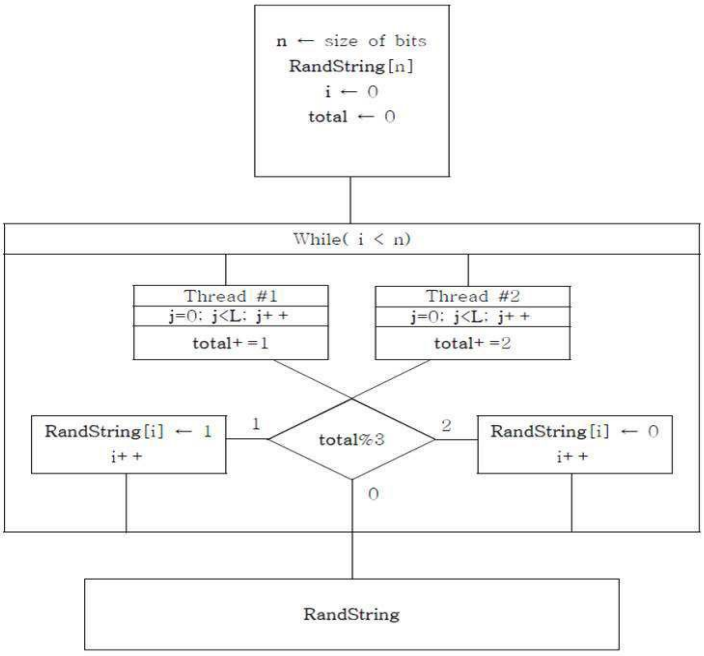
도면1



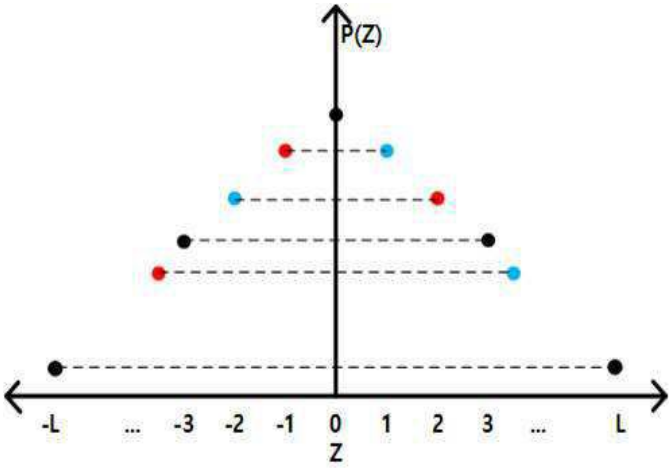
도면2


Core 1
Thread A
Individual Memory
Core 2
Thread B
Individual Memory
Shared Memory

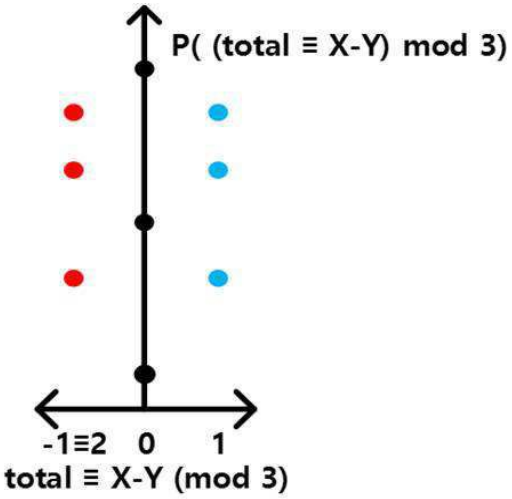
도면3


n ← size of bits
RandString[n]
i ← 0
total ← 0
While(i < n)
Thread #1
j=0; j<L; j++
total+=1
Thread #2
j=0; j<L; j++
total+=2
RandString[i] ← 1
i++
1
total%3
2
RandString[i] ← 0
i++
0
RandString

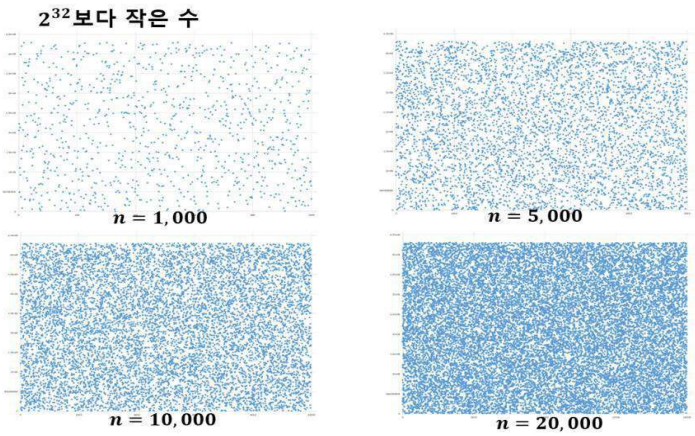
도면4a



도면4b



도면5



도면6

2^8보다 작은 수 5,000개

```
Entropy = 7.966875 bits per byte.

Optimum compression would reduce the size
of this 5000 byte file by 0 percent.

Chi square distribution for 5000 samples is 227.42, and randomly
would exceed this value 89.23 percent of the times.

Arithmetic mean value of data bytes is 127.7006 (127.5 = random).
Monte Carlo value for Pi is 3.116446579 (error 0.80 percent).
Serial correlation coefficient is 0.000431 (totally uncorrelated = 0.0).
```

2^8보다 작은 수 40,000개

```
Entropy = 7.995171 bits per byte.

Optimum compression would reduce the size
of this 40000 byte file by 0 percent.

Chi square distribution for 40000 samples is 267.90, and randomly
would exceed this value 27.71 percent of the times.

Arithmetic mean value of data bytes is 127.6607 (127.5 = random).
Monte Carlo value for Pi is 3.129912991 (error 0.37 percent).
Serial correlation coefficient is -0.000359 (totally uncorrelated = 0.0).
```