(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년04월01일
(11) 등록번호 10-1964392
(24) 등록일자 2019년03월26일

(51) 국제특허분류(Int. Cl.)
G06F 15/167 (2006.01)
(21) 출원번호 10-2014-7005377
(22) 출원일자(국제) 2012년09월18일
심사청구일자 2017년09월12일
(85) 번역문제출일자 2014년02월27일
(65) 공개번호 10-2014-0068909
(43) 공개일자 2014년06월09일
(86) 국제출원번호 PCT/US2012/055942
(87) 국제공개번호 WO 2013/048826
국제공개일자 2013년04월04일
(30) 우선권주장
13/414,593 2012년03월07일 미국(US)
61/541,051 2011년09월29일 미국(US)
(56) 선행기술조사문헌
KR1020010045288 A
KR1020060066341 A
US20110055493 A1
US7594234 B1

(73) 특허권자
오라클 인터내셔날 코포레이션
미국, 캘리포니아 94065, 레드우드 시티, 오라클 파크웨이 500
(72) 발명자
시엔 쉬강
중국 베이징 펭타이 디스트릭트 마 지아 푸 시 루 30번 11-3-401
리 샹동
중국 베이징 쉬안우 디스트릭트 창춘지에실리 9-5-1
(74) 대리인
박장원

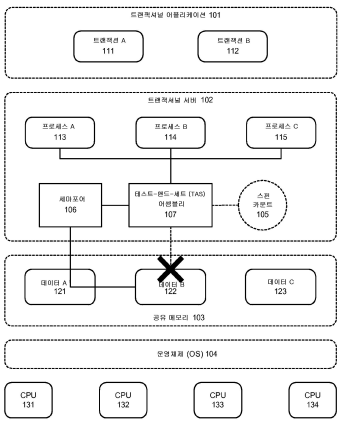
전체 청구항 수 : 총 21 항 심사관 : 윤혜숙

(54) 발명의 명칭 **트랜잭셔널 미들웨어 머신 환경에서 자가-튜닝 락킹 매커니즘을 지원하기 위한 시스템 및 방법**

(57) 요 약

락킹 메커니즘은 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위해 트랜잭셔널 미들웨어 시스템에 지원될 수 있다. 트랜잭셔널 미들웨어 머신 환경은 복수의 프로세서들 상에서 실행되는 운영 체제에 의해 제공되는 세마포어를 포함한다. 트랜잭셔널 미들웨어 머신 환경은 또한, 하나 이상의 프로세스들과 관련된 테스트-앤드-세트(TAS: test-and-set) 어셈블리 컴포넌트를 포함한다. 각각의 상기 프로세스는 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 동작한다. 추가적으로, 프로세스는 상기 TAS 컴포넌트가 다수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에, 세마포어 상에서 블록킹되고 공유 메모리 내의 데이터의 락킹의 해제를 기다리도록 동작한다.

대 표 도 - 도1



명 세 서

청구범위

청구항 1

트랜잭셔널 미들웨어 머신 환경(transactional middleware machine environment)에서 락킹 매커니즘(locking mechanism)을 지원하기 위한 시스템으로서,

복수의 프로세서들 상에서 실행되는 운영 체제에 의해 제공되는 세마포어(semaphore)와, 여기서 상기 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작하고;

하나 이상의 프로세스들과 관련된 TAS(test-and-set) 어셈블리 컴포넌트를 포함하여 구성되며, 여기서 각각의 상기 프로세스는 상기 TAS 어셈블리 컴포넌트를 사용하여 상기 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 동작하고,

여기서 상기 TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 상기 락킹을 획득하지 못한 이후에, 일 프로세스는 상기 세마포어를 이용하여 블록킹되고 상기 공유 메모리 내의 데이터 상의 락킹의 해제(release)를 기다리도록 동작하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 2

제1항에 있어서,

상기 락킹 메커니즘은 복수의 동시적인 트랜잭션들이 존재할 때 상기 공유 메모리 내의 트랜잭션 데이터를 보호하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 3

제1항에 있어서,

상기 락킹 메커니즘은 스핀 카운트(spin cout)를 사용하고, 상기 스핀 카운트는 허용된 TAS 동작들의 최대 라운드들의 특정 수인 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 4

제3항에 있어서,

상기 스핀 카운트는 메타데이터에 미리 구성되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 5

제3항에 있어서,

상기 스핀 카운트는 하드웨어 구성 및 어플리케이션 시나리오(application scenario)에 근거하여 동적으로 결정되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 6

제3항에 있어서,

상기 스핀 카운트는 특별한 프로세스를 사용하여 주기적으로 결정되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 7

제6항에 있어서,

상기 스핀 카운트는 알고리즘을 이용하여 동적으로 결정되고, 상기 알고리즘은,

이전 주기에서의 스핀 실패들(spin failures)의 수가 스핀 실패 한도(spin failure limit)를 초과하고 이전 주기에서의 CPU 휴지비(CPU idle ratio)가 CPU 휴지비 한도보다 낮으면 상기 스핀 카운트는 이전 주기의 스핀 카운트로부터 증가되고, 그리고

상기 CPU 휴지비가 CPU 휴지비 한도를 초과하면 상기 스핀 카운트는 이전 주기에서의 스핀 카운트로부터 감소됨을 특정하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 8

제7항에 있어서,

스핀 실패는 프로세스가 상기 TAS 동작을 특정 횟수 시도한 이후에 데이터 상의 락킹을 획득하지 못할 때 발생되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 9

제1항에 있어서,

상기 세마포어를 이용하여, 상기 프로세스는 락 오너(lock owner)가 기상(wake up)하고 상기 락킹을 해제할 때 상기 락킹을 획득하도록 동작하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 10

제3항에 있어서,

상기 스핀 카운트는 최적화된 값을 찾도록 수동으로 미세하게 튜닝되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하기 위한 시스템.

청구항 11

트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법으로서,

복수의 프로세서들 상에서 실행되는 운영 체제와 관련된 세마포어를 제공하는 단계와, 상기 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작하고;

하나 이상의 프로세스들 중 일 프로세스를 통해 TAS 어셈블리 컴포넌트를 사용하여 상기 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 하는 단계와;

상기 TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 상기 락킹을 획득하지 못한 이후에, 상기 세마포어를 이용하여 상기 프로세스가 블록킹되는 단계 및 상기 공유 메모리 내의 데이터 상의 락킹의 해제를 기다리는 단계를 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 12

제11항에 있어서,

복수의 동시적인 트랜잭션들이 존재할 때 상기 공유 메모리 내의 트랜잭션 데이터를 보호하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 13

제11항에 있어서,

상기 락킹 메커니즘이 스핀 카운트를 사용하게 하는 단계를 더 포함하고, 상기 스핀 카운트는 허용된 TAS 동작들의 최대 라운드들의 특정 수인 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 14

제13항에 있어서,

메타데이터에 상기 스핀 카운트를 미리 구성하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 15

제13항에 있어서,

하드웨어 구성 및 어플리케이션 시나리오에 근거하여 상기 스핀 카운트를 동적으로 결정하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 16

제13항에 있어서,

특별한 프로세스를 사용하여 상기 스핀 카운트를 주기적으로 결정하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 17

제16항에 있어서,

상기 스핀 카운트는 알고리즘을 이용하여 동적으로 결정되고, 상기 알고리즘은

이전 주기에서의 스핀 실패들 수가 스핀 실패 한도를 초과하고 이전 주기에서의 CPU 휴지비가 CPU 휴지비 한도보다 낮으면 상기 스핀 카운트는 이전 주기의 스핀 카운트로부터 증가되고, 그리고

상기 CPU 휴지비가 CPU 휴지비 한도를 초과하면 상기 스핀 카운트는 이전 주기에서의 스핀 카운트로부터 감소됨을 특정하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 18

제17항에 있어서,

스핀 실패는 프로세스가 상기 TAS 동작을 특정 횟수 시도한 이후에 데이터 상의 락킹을 획득하지 못할 때 발생되는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 19

제11항에 있어서,

락 오너가 기상하고 상기 락킹을 해제할 때 상기 락킹을 획득하는 데 상기 세마포어를 사용하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 20

제13항에 있어서,

최적화된 값을 찾도록 상기 스핀 카운트를 수동으로 미세하게 튜닝하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 미들웨어 머신 환경에서 락킹 매커니즘을 지원하는 방법.

청구항 21

컴퓨터로 하여금 제11항 내지 20항 중 어느 한 항의 방법을 수행하도록 하는 프로그램을 기록한 컴퓨터 판독가능한 기록 매체.

청구항 22

삭제

발명의 설명

기 술 분 야

[0001] 저작권 공지

[0002] 본 명세서에서 개시된 부분은 저작권 보호를 받는 내용을 포함한다. 저작권자는 미국특허상표청의 특허 파일 또는 기록에 나타난 대로 본 특허 문서 또는 특허 개시내용을 어느 누군가가 팩시밀리 재생하는 것은 반대하지않지만, 그 밖의 모든 것은 저작권으로 보호된다.

[0003] 기술분야

[0004] 본 발명은 일반적으로, 미들웨어와 같은 컴퓨터 시스템들 및 소프트웨어에 관한 것이며, 특히 트랜잭셔널 미들웨어 머신 환경(transactional middleware machine environment)을 지원하는 것에 관한 것이다.

배 경 기 술

[0005] 트랜잭셔널 미들웨어 시스템 또는 트랜잭션 지향식 미들웨어(transaction oriented middleware)는 조직 내에서 다양한 트랜잭션들을 프로세스할 수 있는 기업 어플리케이션 서버들을 포함한다. 고성능 네트워크 및 멀티프로세서 컴퓨터와 같은 새로운 기술들의 발달에 따라, 트랜잭셔널 미들웨어의 성능을 더 향상시킬 필요가 있다. 이러한 것들은 대체로 본 발명의 실시예들이 해결하도록 의도된 영역들이다.

발명의 내용

[0006] 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위해 트랜잭셔널 미들웨어 시스템에 지원될 수 있는 락킹 메커니즘(lock mechanism)이 본 명세서에 개시된다. 트랜잭셔널 미들웨어 머신 환경은 복수의 프로세서들 상에서 실행되는 운영 체제를 포함하고, 이 프로세서들의 각각은 공유 메모리 내의 데이터에 액세스하도록 동작한다. 트랜잭셔널 미들웨어 머신 환경은 또한, 운영 체제에 의해 제공되는 세마포어(semaphore) 및, 하나 이상의 프로세스들과 관련된 테스트-앤드-세트(TAS: test-and-set) 어셈블리 컴포넌트를 포함한다. 각각의 프로세스는 상기 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행할 수 있다. 더욱이, 프로세스는 상기 TAS 컴포넌트가 특정 수의 TAS 동작들을 수행한 후 락킹을 획득하지 못한 이후에, 세마포어를 블록킹하고 공유 메모리 내의 데이터의 락킹의 해제(release)를 기다릴 수 있다.

도면의 간단한 설명

[0007] 도 1은 본 발명의 실시예에 따른 자가-튜닝 락킹 메커니즘을 지원하는 트랜잭셔널 미들웨어 머신 환경의 예시를 도시한다.

도 2는 본 발명의 실시예에 따른 트랜잭셔널 미들웨어 머신 환경에서 자가-튜닝 락킹 메커니즘을 지원하기 위한 예시적인 흐름도를 도시한다.

도 3은 본 발명의 실시예에 따른 트랜잭셔널 미들웨어 머신 환경에서 락킹 메커니즘을 지원하기 위한 시스템의 기능 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0008] 복수의 프로세서들을 가진 고속 머신들 및 고성능 네트워크 연결의 장점을 취할 수 있는 Tuxedo와 같은 트랜잭셔널 미들웨어 시스템을 지원하기 위한 시스템 및 방법이 본 명세서에 기술된다. 락킹 메커니즘은 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위해 트랜잭셔널 미들웨어 시스템에서 지원될 수 있다. 트랜잭셔널 미들웨어 머신 환경은 복수의 프로세서들 상에서 실행되는 운영 체제에 의해 제공되는 세마포어를 포함한다. 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스할 수 있다. 트랜잭셔널 미들웨어 머신 환경은 또한, 하나 이상의 프로세스들과 관련된 TAS 어셈블리 컴포넌트를 포함한다. 각각의 상기 프로세스는 상기 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 동작한다. 추가적으로, 프로세스는 상기 TAS 컴포넌트가 다수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에, 세마포어 상에서 블록킹되고 공유 메모리 내의 데이터의 락킹의 해제를 기다리도록 동작한다.

[0009] 본 발명의 실시예에 따르면, 시스템은, 신속하게 준비될 수 있고 수요에 따라 스케일링될 수 있는 대량의 병렬 인-메모리 그리드(massively parallel in-memory grid)를 포함하는 완전한 Java EE 어플리케이션 서버 컴플렉

스를 제공하기 위해 WebLogic Suite와 같은 어플리케이션 서버 또는 미들웨어 환경과 함께 고성능 하드웨어, 예컨대 64-비트 프로세서 기술, 고성능 대용량 메모리 및 리던던트 인피니밴드(redundant InfiniBand) 및 이더넷 네트워킹의 조합을 포함한다. 실시예에 따르면, 시스템은 전체, 절반 또는 1/4 랙(rack) 또는 다른 구성으로 배치될 수 있고, 상기 시스템은 어플리케이션 서버 그리드, 스토리지 영역 네트워크 및 인피니밴드(IB) 네트워크를 제공한다. 미들웨어 머신 소프트웨어는 예컨대, WebLogic Server, JRockit 또는 Hotspot JVM, Oracle Linux 또는 Solaris, 및 Oracle VM과 같은 어플리케이션 서버, 미들웨어 및 다른 기능을 제공할 수 있다. 실시예에 따르면, 시스템은 IB 네트워크를 통해 서로와 통신하는 복수의 컴퓨팅 노드들, IB 스위치 게이트웨이 및 스토리지 노드들 또는 유닛들을 포함할 수 있다. 랙 구성으로 구현될 때, 랙의 사용되지 않은 부분들은 비어있는 채로 있거나 필러(filler)들에 의해 채워질 수 있다.

[0010] "Sun Oracle Exalogic" 또는 "Exalogic"으로 본 명세서에서 지칭되는 본 발명의 실시예에 따르면, 시스템은 Oracle Middleware SW suite 또는 Weblogic과 같은 미들웨어 또는 어플리케이션 서버 소프트웨어를 호스팅하기 위한 배치가 용이한 해법(easy-to-deploy solution)이다. 본 명세서에 기술된 바와 같이, 실시예에 따르면, 시스템은 하나 이상의 서버들, 스토리지 유닛들, 스토리지 네트워킹을 위한 IB 페브릭 및 미들웨어 어플리케이션을 호스팅하는 데 요구되는 모든 다른 컴포넌트들을 포함하는 "박스 내의 그리드"이다. 상당한 성능이 예컨대, 리얼 어플리케이션 클러스터(Real Application Cluster)들 및 엑사로직 오픈 스토리지(Exalogic Open storage)를 이용하여 대량의 병렬 그리드 구조를 활용함으로써 모든 타입의 미들웨어 어플리케이션들에 제공될 수 있다. 시스템은 선형적 I/O 확장성(scalability)을 포함하는 개선된 성능을 제공하고, 사용 및 관리하기에 간단하며, 작업상 중대한(mission-critical) 가용성 및 신뢰성을 제공한다.

[0011] 본 발명의 실시예에 따르면, Tuxedo는 고성능, 분산형 비즈니스 어플리케이션들의 구성, 실행 및 관리(administration)를 할 수 있는 소프트웨어 모듈들의 세트이며, 다수의 다단(multi-tier) 어플리케이션 개발 툴들에 의해 트랜잭셔널 미들웨어로서 사용되었다. Tuxedo는 분산형 컴퓨팅 환경들에서 분산형 트랜잭션 프로세싱을 관리하는 데 사용될 수 있는 미들웨어 플랫폼이다. 이는, 기업의 레거시 어플리케이션들을 언락킹하고 상기 어플리케이션들을 서비스 지향식 구조로 확장하면서도 비제한적 확장성 및 표준-기반의 상호운용성을 제공하기 위한 검증된 플랫폼이다.

[0012] 본 발명의 실시예에 따르면, Tuxedo 시스템과 같은 트랜잭셔널 미들웨어 시스템은 엑사로직 미들웨어 머신과 같은 복수의 프로세서들을 가진 고속 머신들 및 인피니밴드(IB) 네트워크와 같은 고성능 네트워크 연결의 장점을 취할 수 있다.

[0013] 자가-튜닝 락킹 메커니즘

[0014] 본 발명의 실시예에 따르면, 자가-튜닝 락킹 메커니즘은 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위해 트랜잭셔널 미들웨어 시스템에서 지원될 수 있다. 자가-튜닝 락킹 메커니즘을 사용하여, 트랜잭셔널 미들웨어 머신 환경은 대량의 동시적인 트랜잭션들을 갖는 어플리케이션들과 같은 트랜잭셔널 어플리케이션 시나리오들에서 상당한 스루풋 향상을 달성할 수 있다.

[0015] 도 1은 본 발명의 실시예에 따른 자가-튜닝 라킹 메커니즘을 지원하는 트랜잭셔널 미들웨어 머신 환경의 예시를 도시한다. 도 1에 도시된 바와 같이, 트랜잭셔널 미들웨어 머신은 운영 체제(OS)(104)를 지원하는 복수의 CPU들(131 내지 134) 및 다양한 트랜잭셔널 데이터(121 내지 123)를 포함하는 공유 메모리(103)를 포함한다. 복수의 동시적인 트랜잭션들(111 내지 112)을 갖는 트랜잭셔널 어플리케이션(101)은 트랜잭셔널 서버(102)에서 복수의 프로세스들(113 내지 115) 상에서 실행될 수 있고, 상기 프로세스들 각각은 효과적인 락킹 메커니즘을 구현하기 위해 원자적 TAS(테스트-앤드-세트) 어셈블리(107)를 사용할 수 있다. 락킹 메커니즘은 동시적인 트랜잭션들이 존재할 때 공유 메모리에서 트랜잭션 데이터를 보호할 수 있다. 추가적으로, 트랜잭션 어플리케이션에서의 프로세스는 필요한 경우, 데이터(122) 상에서 락킹을 획득하도록 OS에 의해 제공되는 세마포어 메커니즘(106)을 사용할 수 있다.

[0016] 일 실시예에 따르면, 프로세스가 데이터(122) 상에서 락킹을 얻고자 할 때, 프로세스는 다수의 라운드들에 대해 TAS 동작을 수행할 수 있다. 시스템은 허용된 TAS 동작의 라운드들의 수인 타겟 스핀 카운트(target spin count)를 특정할 수 있다. 상기 타겟 스핀 카운트는 미리 구성될 수 있거나 동적으로 결정될 수 있다.

[0017] 타겟 스핀 카운트에 도달되기 전에 락킹이 이용가능해지면, 프로세스는 OS에 의해 제공되는 세마포어 메커니즘보다 훨씬 적은 비용으로 락킹을 획득할 수 있다. 한편, 락킹이 이 주기 동안 이용가능하지 않으면, 프로세스는 세마포어를 블록킹하고 락 오너(rock ower)가 기상해서 락킹을 해제시킬때가지 기다리도록 구성될 수 있다.

[0018] 본 발명의 실시예에 따르면, 타겟 스핀 카운트 값은 하드웨어 구성 및 어플리케이션 시나리오의 맥락에서 결정될 수 있다. 사용자들은 최적화된 값을 찾기위해서 상기 스핀 카운트 값을 수동으로 미세하게 튜닝할 수 있다. 상기 결정은 락킹을 얻기 위한 CPU 사용과 시간 사이에 트레이드오프(trade-off)가 존재하기 때문에 일부 상황들에서 명백하지 않을 수 있다. 예를 들어, 사용자들은 보다 짧은 시간 프레임에서 락킹을 얻기 위해서 보다 많은 TAS 동작들을 수행하는 데 보다 많은 CPU 전력을 소모해야 할 수 있다. 따라서, 모든 경우에 명백한 최적화된 타겟 스핀 카운트 값은 존재하지 않을 수 있다.

[0019] 사용자 레벨 세마포어 구현의 일 단점은 트랜잭셔널 어플리케이션이 특정 머신 타입에 따라 실시간으로 타겟 스핀 카운트를 동적으로 조정하지 못할 수 있다는 점이다. 일반적으로, 사용자 레벨 세마포어는 단지, 정적으로 구성된 타겟 스핀 카운트 값을 사용하고, 사용자들은 실험 실시들에 의해서만 타겟 스핀 카운트 값을 수동으로 조정할 수 있다. 최적의 스핀 카운트 값이 머신에 종속적이고 모든 플랫폼들에 대하여 하나가 전체에 피팅되는(one-fit-in-all) 값이 존재하지 않기 때문에, 보다 적절한 접근법은 타겟 스핀 카운트 값을 동적으로 그리고 실시간으로 계산하기 위한 메커니즘을 채용하는 것이다.

[0020] 본 발명의 실시예에 따르면, 타겟 스핀 카운트 값은 공유 메모리에 저장될 수 있다. Tuxedo 데몬 프로세스와 같은 특별한 프로세스는 이전 주기에 수집된 동작 정보에 따라 스핀 카운트 값을 주기적으로 변경시킬 수 있다. 예를 들어, Tuxedo 데몬은 디폴트로 5초 마다 한 번씩 타겟 스핀 카운트 값을 갱신할 수 있다.

[0021] 일 실시예에 따르면, 알고리즘은 타겟 스핀 카운트 값을 구성하는 데 사용될 수 있다. 알고리즘은 CPU 휴지비(idle ratio)가 낮거나 또는 너무 많은 TAS 동작들이 락킹을 얻지 못하고 시스템이 세마포어로 스위칭되었으면, 타겟 스핀 카운트 값을 증가시킬 수 있다. 더욱이, 알고리즘은 CPU 휴지비가 너무 높으면 타겟 스핀 카운트 값을 감소시킬 수 있다.

[0022] 도 2는 본 발명의 실시예에 따른 트랜잭셔널 미들웨어 머신 환경에서 자가-튜닝 락킹 메커니즘을 지원하기 위한 예시적인 흐름도를 도시한다. 도 2에 도시된 바와 같이, 단계(201)에서, 시스템은 복수의 프로세서들 상에서 실행되는 운영 체제와 관련된 세마포어를 제공할 수 있고, 여기서 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작한다. 그 다음, 단계(202)에서, 하나 이상의 프로세스들 중 일 프로세스는 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행할 수 있다. 최종적으로, 단계(203)에서, 프로세스는 TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에, 세마포어 상에서 블록킹되고 공유 메모리 내의 데이터 상의 락킹의 해제를 기다릴 수 있다.

[0023] Tuxedo로 SPINCOUNT 값을 구성

[0024] 본 발명의 실시예에 따르면, Tuxedo 구성 파일에서의 SPINCOUNT 파라미터와 같은 메타데이터는 타겟 스핀 카운트를 특정하는 데 사용될 수 있다. SPINCOUNT는 정적 구성 값(static configured value) 또는 동적 구성 값일 수 있다.

[0025] 예를 들어, Tuxedo는 시스템 레벨 세마포어 상에서 블록킹되기 전에 게시판 락킹 대기들이 몇 번 스핀(spinning)하는지 결정하도록 정적으로 설정된 값의 SPINCOUNT를 사용할 수 있다. 이 알고리즘의 단점은 최적 값의 SPINCOUNT가 CPU량, 작업부하, 게시판(BB) 락킹의 대기 수 등과 같은 많은 동적 인자들에 의존적이기 때문에 사용자에 의해 설정된 값이 특정 플랫폼 상에서 최적 값의 SPINCOUNT가 아니라는 점이다.

[0026] 추가적으로, Tuxedo는 런타임 환경을 고려하면서도 SPINCOUNT 값을 동적으로 튜닝할 수 있다. 알고리즘은 SPINCOUNT 파라미터에 대한 적절한 값을 결정하는 데 사용될 수 있다. 시스템은 너무 많은 TAS 동작들이 이전 주기에서 실패했고, 시스템이 세마포어로 스위칭되었으며 충분한 CPU 휴지비가 존재했다면, SPINCOUNT를 증가시킬 수 있다. 한편, 시스템은 CPU 휴지비가 너무 높았다면 SPINCOUNT를 감소시킬 수 있다.

[0027] 상기 알고리즘은 CPU 사용 예컨대, CPU 휴지비 및 스핀 실패의 비(예컨대, 락킹을 얻기 위한 매 10000번의 동작들에서의 스핀 실패의 비)에 근거한다. 스핀 실패의 비는 락킹들이 TAS 동작을 통하는 대신 세마포어를 통해 몇 번이나 획득되는지를 나타낼 수 있다.

[0028] 프로세스는 상기 프로세스가 활성 상태에 있을 때 스핀 모드에 있을 수 있다. 프로세스는 상기 프로세스가 다수 번 TAS 동작들을 수행하는 것을 시도한 이후에 락킹을 획득하지 못하면 세마포어 상에서 블록킹될 수 있고, 이는 스핀 실패라고 지칭된다. 프로세스는 스핀 실패가 발생되기 이전에 락킹을 얻기 위해서 TAS 동작을 또다시 수행하길 시도할 수 있다. 구성가능한 파라미터, 예컨대 SPINCOUNT는 호출되고 수행될 TAS 동작의 라운드들의

수를 특정하는 데 사용될 수 있다.

[0029] 본 발명의 실시예에 따르면, 최소 휴지 CPU율 및 스핀 실패율(SPIN failed rate)이 구성 파일 내의 메타데이터를 사용하여 정의될 수 있다. 예를 들어, Tuxedo 구성 파일에서, 최소 휴지 CPU율을 정의하기 위한 MINIDLECPU 파라미터가 존재할 수 있고, 상기 파라미터의 값은 1 내지 100의 범위를 가지며 디폴트로 20을 가진다. 추가적으로, 스핀 실패율을 정의하기 위한 FACTOR 파라미터가 존재할 수 있고, 상기 파라미터의 값은 1 내지 10000의 범위를 가지며 디폴트로 1000을 가진다.

[0030] 최소 휴지 CPU율 및 스핀 실패율이 주어질 때, 시스템은 각각의 스캔 유닛 내의 SPINCOUNT를 튜닝할 수 있다. 그 다음, 시스템은 스핀 실패율이 너무 높거나(예컨대, 스핀 실패율이 (1/FACTOR*1.1)보다 크게 설정 되거나) 또는 휴지 CPU 시간이 충분하면(예컨대, 휴지율 > MINIDLECPU% +0.05), SPINCOUNT를 증가시킬 수 있으며, 새로운 SPINCOUNT = 오래된 SPINCOUNT + 오래된 SPINCOUNT * (cpu_idletime/cpu_usertime)이고, 최대 SPINCOUNT는 10,000,000로 설정될 수 있다. 이와는 달리, 시스템은 휴지 CPU 비가 너무 낮으면(예컨대, 휴지율 < MINIDLECPU% - 0.05), SPINCOUNT를 감소시킬 수 있으며, 새로운 SPINCOUNT = 오래된 SPINCOUNT/4이고, 최소 SPINCOUNT는 50000으로 설정된다.

[0031] 다음의 리스팅 1은 SPINTUNING 구성에 대한 Tuxedo 예이다.

```
*RESOURCES
IPCKEY          123456
DOMAINID        simpapp
MASTER          ALLEN
MAXACCESSERS 10
MAXSERVERS    5
MAXSERVICES   10
MODEL           SHM
LDBAL           N
OPTIONS         EXALOGIC,SPINTUNING

*MACHINES
ALLENHOST     LMID="ALLEN"
APPDIR="/home/allen/Workspace/Tuxedo11gR1PS2/simpdir"
TUXCONFIG="/home/allen/Workspace/Tuxedo11gR1PS2/simpdir/tuxconfig"
TUXDIR="/home/allen/Software/OraHome/tuxedo11gR1PS2"
SPINTUNING_TARGET=1000
SPINTUNING_MINIDLECPU=20
*GROUPS
GROUP1
LMID=ALLEN GRPNO=1 OPENINFO=NONE

*SERVERS
DEFAULT:
CLOPT="-A"
simpserv  SRVGRP=GROUP1 SRVID=1

*SERVICES
TOUPPER
```

[0032]

[0033] 리스팅 1

[0034] 상기 예에서 도시된 바와 같이, 구성 파일의 *MACHINES 섹션은 튜닝 타겟을 구성하기 위해 사용되는 속성 "SPINTUNING_TARGET"을 포함한다. "SPINTUNING_TARGET"의 값은 0보다 크거나 동일하고 예컨대, "10000"보다 작거나 동일할 수 있는 숫자이다. 0의 값은 바이너리로 구축된 값이 사용된다는 것을 나타낸다. 빌트-인 값은 1000이다. "SPINTUNING_TARGET"의 디폴트 값은 0이다.

[0035] 상기 예에서, SPINTUNING_TARGET의 값은 게시판이 매 1000번의 락킹마다 시스템 세마포어를 통해 락킹되는 것이

최대 한 번 존재함을 의미한다. 시스템은 SPINCOUNT의 값을 증가시킬 수 있고, 많은 CPU가 SPINTUNING_TARGET의 큰 값을 충족시키는 데 소모된다. 속성은 옵션 SPINTUNING이 특정될 때에만 비제로(nonzero) 값으로 설정될 수 있다.

[0036] 상술된 예에 도시된 바와 같이, 구성 파일의 *MACHINES 섹션은 휴지 CPU율을 특정하기 위해 사용되는 속성 "SPINTUNING_MINIDLECPU"을 포함한다. 휴지 CPU율은 적절한 SPINCOUNT를 동적으로 찾는 데 시스템에 의해 사용될 수 있다. SPINCOUNT가 커질수록, 시스템이 사용하는 CPU도 많아진다. 사용자는 너무 많은 CPU를 소모하는 것을 회피하도록 "SPINTUNING_MINIDLECPU"를 통해 최소 휴지 CPU율을 설정할 수 있다.

[0037] "SPINTUNING_MINIDLECPU"의 값은 숫자 및 퍼센티지로 존재한다. 이는 "0"보다 크거나 같거나 또는 "100"보다 작거나 같을 수 있다. 0의 값은 제공된 바이너리로 구축되는 값이 사용되어야 함을 나타낸다. 예를 들어, Tuxedo에서, 빌트-인 값은 20으로 설정될 수 있고, "SPINTUNING_ MINIDLECPU"의 디폴트 값은 0이다. 속성은 옵션 SPINTUNING 이 특정될 때에만 비제로 값으로 설정될 수 있다.

[0038] 일부 실시예들에 따르면, 도 3은 상술된 바와 같은 발명들의 원리들에 따라 구성된 트랜잭셔널 미들웨어 머신 환경에서 락킹 메커니즘을 지원하기 위한 시스템(1000)의 기능 블록도를 도시한다. 시스템(1000)의 기능 블록들은 본 발명의 원리들을 시행하도록 하드웨어, 소프트웨어 또는 하드웨어 및 소프트웨어의 조합에 의해 구현될 수 있다. 도 3에 기술된 기능 블록들이 상술된 바와 같은 본 발명의 원리들을 구현하도록 결합되거나 또는 서브-블록들로 분리될 수 있음이 이 기술분야의 숙련자들에게 이해될 것이다. 그러므로, 본 명세서에서의 설명은 본 명세서에 기술된 기능 블록들의 어떤 가능한 결합 또는 분리 또는 추가의 정의를 지지할 수 있다.

[0039] 도 3에 도시된 바와 같이, 트랜잭셔널 미들웨어 머신 환경에서 락킹 메커니즘을 지원하기 위한 시스템(1000)은 제공 유닛(1100) 및 TAS 어셈블리 컴포넌트(1200)를 포함한다. 제공 유닛(1100)은 복수의 프로세서들 상에서 실행되는 운영 체제와 관련된 세마포어를 제공하고, 여기서 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작한다. TAS 어셈블리 컴포넌트(1200)는 하나 이상의 프로세스들과 관련되고, 각각의 상기 프로세스는 TAS 어셈블리 컴포넌트(1200)를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 동작한다. 프로세스는 TAS 컴포넌트(1200)가 특정 수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에 세마포어 상에서 블록킹되고 공유 메모리 내의 데이터 상의 락킹의 해제를 기다리도록 동작할 수 있다.

[0040] 일부 실시예들에서, 시스템(1000)은 복수의 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위한 보호 유닛(1300)을 더 포함한다.

[0041] 일부 실시예들에서, 시스템(1000)은 락킹 메커니즘으로 하여금 스핀 카운트를 사용하도록 하기 위한 허용 유닛(1400)을 더 포함하고, 상기 스핀 카운트는 허용된 TAS 동작들의 최대 라운드들의 특정 수이다.

[0042] 일부 실시예들에서, 시스템(1000)은 메타데이터에 스핀 카운트를 미리 구성하기 위한 사전구성 유닛(preconfiguring unit)(1500)을 더 포함한다.

[0043] 일부 실시예들에서, 시스템(1000)은 하드웨어 구성 및 어플리케이션 시나리오에 근거하여 스핀 카운트를 동적으로 결정하기 위한 동적 결정 유닛(dynamically determining unit)(1600)을 더 포함한다.

[0044] 일부 실시예들에서, 시스템(1000)은 특별한 프로세스를 사용하여 스핀 카운트를 주기적으로 결정하기 위한 주기적 결정 유닛(periodically determining unit)(1700)을 더 포함한다.

[0045] 일부 실시예들에서, 스핀 카운트는 알고리즘을 사용하여 동적으로 결정되고, 상기 알고리즘은, 이전 주기에서 스핀 실패의 수가 스핀 실패 한도(spin failure limit)를 초과하고 이전 주기에서 CPU 휴지비가 CPU 휴지비 한도보다 작으면 스핀 카운트가 이전 주기의 스핀 카운트로부터 증가되고, CPU 휴지비가 CPU 휴지비 한도를 초과하면 스핀 카운트가 이전 주기의 스핀 카운트로부터 감소됨을 특정한다.

[0046] 일부 실시예들에서, 스핀 실패는 프로세스가 특정 횟수 TAS 동작을 시도한 이후에 데이터 상의 락킹을 획득하지 못할 때 발생된다.

[0047] 일부 실시예들에서, 시스템(1000)은 락 오너가 기상해서 락킹을 해제할 때 락킹을 획득하는 데 세마포어를 사용하기 위한 사용 유닛(1800)을 더 포함한다.

[0048] 일부 실시예들에서, 시스템(1000)은 최적화된 값을 찾도록 스핀 카운트를 수동으로 미세하게 튜닝하기 위한 미세 튜닝 유닛(1900)을 더 포함한다.

[0049] 다른 실시예들은 트랜잭셔널 미들웨어 머신 환경에서 락킹 메커니즘을 지원하기 위한 장치를 포함하는 바, 상기 장치는 복수의 프로세서들 상에서 실행되는 운영 체제와 관련된 세마포어를 제공하기 위한 수단과, 여기서 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작하고, 하나 이상의 프로세스들 중 일 프로세스를 통해 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위해 하나 이상의 TAS 동작들을 수행하기 위한 수단과, TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에 프로세스를 통해 세마포어 상에서 블록킹하고 공유 메모리 내의 데이터 상의 락킹의 해제를 기다리기 위한 수단을 포함하여 구성된다.

[0050] 다른 실시예에서, 상기 장치는 복수의 동시적인 트랜잭션들이 존재할 때, 공유 메모리 내의 트랜잭션 데이터를 보호하기 위한 수단을 더 포함하여 구성된다.

[0051] 다른 실시예에서, 상기 장치는 락킹 메커니즘이 스핀 카운트를 사용하도록 하기 위한 수단을 더 포함하여 구성되고, 상기 스핀 카운트는 허용된 TAS 동작들의 최대 라운드들의 특정 수이다.

[0052] 다른 실시예에서, 상기 장치는 메타데이터에 스핀 카운트를 미리 구성하기 위한 수단을 더 포함하여 구성된다.

[0053] 다른 실시예에서, 상기 장치는 하드웨어 구성 및 어플리케이션 시나리오에 근거하여 스핀 카운트를 동적으로 결정하기 위한 수단을 더 포함하여 구성된다.

[0054] 다른 실시예에서, 상기 장치는 특별한 프로세스를 사용하여 스핀 카운트를 주기적으로 결정하기 위한 수단을 더 포함하여 구성된다.

[0055] 일 실시예에서, 장치를 더 포함하고, 여기서 스핀 카운트는 알고리즘을 이용하여 동적으로 결정되고, 상기 알고리즘은, 이전 주기에서 스핀 실패의 수가 스핀 실패 한도를 초과하고 이전 주기에서 CPU 휴지비가 CPU 휴지비 한도보다 작을 때 스핀 카운트가 이전 주기의 스핀 카운트로부터 증가되고, CPU 휴지비가 CPU 휴지비 한도를 초과할 때 스핀 카운트가 이전 주기의 스핀 카운트로부터 감소됨을 특정한다.

[0056] 다른 실시예에서, 상기 장치는 스핀 실패가 프로세스가 특정 횟수 TAS 동작을 시도한 이후에 데이터 상의 락킹을 획득하지 못할 때 발생되는 것을 특징으로 한다.

[0057] 다른 실시예에서, 상기 상치는 락 오너가 기상하고 락킹을 해제할 때 락킹을 획득하는 데 세마포어를 사용하기 위한 수단을 더 포함한다.

[0058] 다른 실시예에서, 상기 장치는 최적화된 값을 찾는 데 스핀 카운트를 수동으로 미세하게 튜닝하기 위한 수단을 더 포함한다.

[0059] 그리고, 또다른 실시예는 트랜잭셔널 미들웨어 머신 환경에서 락킹 메커니즘을 지원하기 위한 시스템을 포함하고, 상기 시스템은 복수의 프로세서들 상에서 실행되는 운영 체제와 관련된 세마포어를 제공하기 위한 유닛과, 여기서 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작하고, 하나 이상의 프로세스들과 관련된 TAS 어셈블리 컴포넌트를 포함하여 구성되고, 여기서 각각의 상기 프로세스는 TAS 어셈블리 컴포넌트를 사용하여 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위한 하나 이상의 TAS 동작들을 수행하도록 동작하고, 여기서 프로세스는 TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 락킹을 획득하지 못한 이후에 세마포어 상에서 블록킹되고 공유 메모리 내의 데이터 상의 락킹의 해제를 기다리도록 동작한다.

[0060] 다른 실시예에서, 상기 시스템은 복수의 동시적인 트랜잭션들이 존재할 때 공유 메모리 내의 트랜잭션 데이터를 보호하기 위한 유닛을 더 포함하여 구성된다.

[0061] 다른 실시예에서, 상기 시스템은 락킹 메커니즘이 스핀 카운트를 사용하도록 하기 위한 유닛을 더 포함하여 구성되고, 상기 스핀 카운트는 허용된 TAS 동작들의 최대 라운드들의 특정 수이다.

[0062] 다른 실시예에서, 상기 시스템은 메타데이터 내의 스핀 카운트를 미리 구성하기 위한 유닛을 더 포함하여 구성된다.

[0063] 다른 실시예에서, 상기 시스템은 하드웨어 구성 및 어플리케이션 시나리오에 근거하여 스핀 카운트를 동적으로 결정하기 위한 유닛을 더 포함하여 구성된다.

[0064] 다른 실시예에서, 상기 시스템은 특별한 프로세스를 사용하여 스핀 카운트를 주기적으로 결정하기 위한 유닛을 더 포함하여 구성된다.

[0065] 다른 실시예는 시스템을 포함하고, 여기서 스핀 카운트는 알고리즘을 이용하여 동적으로 결정되고, 상기 알고리

즘은, 이전 주기에서 스핀 실패의 수가 스핀 실패 한도를 초과하고 이전 주기에서 CPU 휴지비가 CPU 휴지비 한도보다 작을 때 스핀 카운트가 이전 주기의 스핀 카운트로부터 증가되고, CPU 휴지비가 CPU 휴지비 한도를 초과할 때 스핀 카운트가 이전 주기의 스핀 카운트로부터 감소됨을 특정한다.

[0066] 다른 실시예는 시스템을 포함하고, 여기서 스핀 실패는 프로세스가 특정 횟수 TAS 동작을 시도한 이후에 데이터 상의 락킹을 얻지 못할 때 발생된다.

[0067] 다른 실시예는 시스템을 포함하고, 상기 시스템은 락 오너가 기상하고 락킹을 해제할 때 락킹을 획득하는 데 세마포어를 사용하기 위한 유닛을 더 포함하여 구성된다.

[0068] 그리고 또 다른 실시예는 시스템을 포함하고, 상기 시스템은 최적화된 값을 찾도록 스핀 카운트를 수동으로 미세하게 튜닝하기 위한 유닛을 더 포함하여 구성된다.

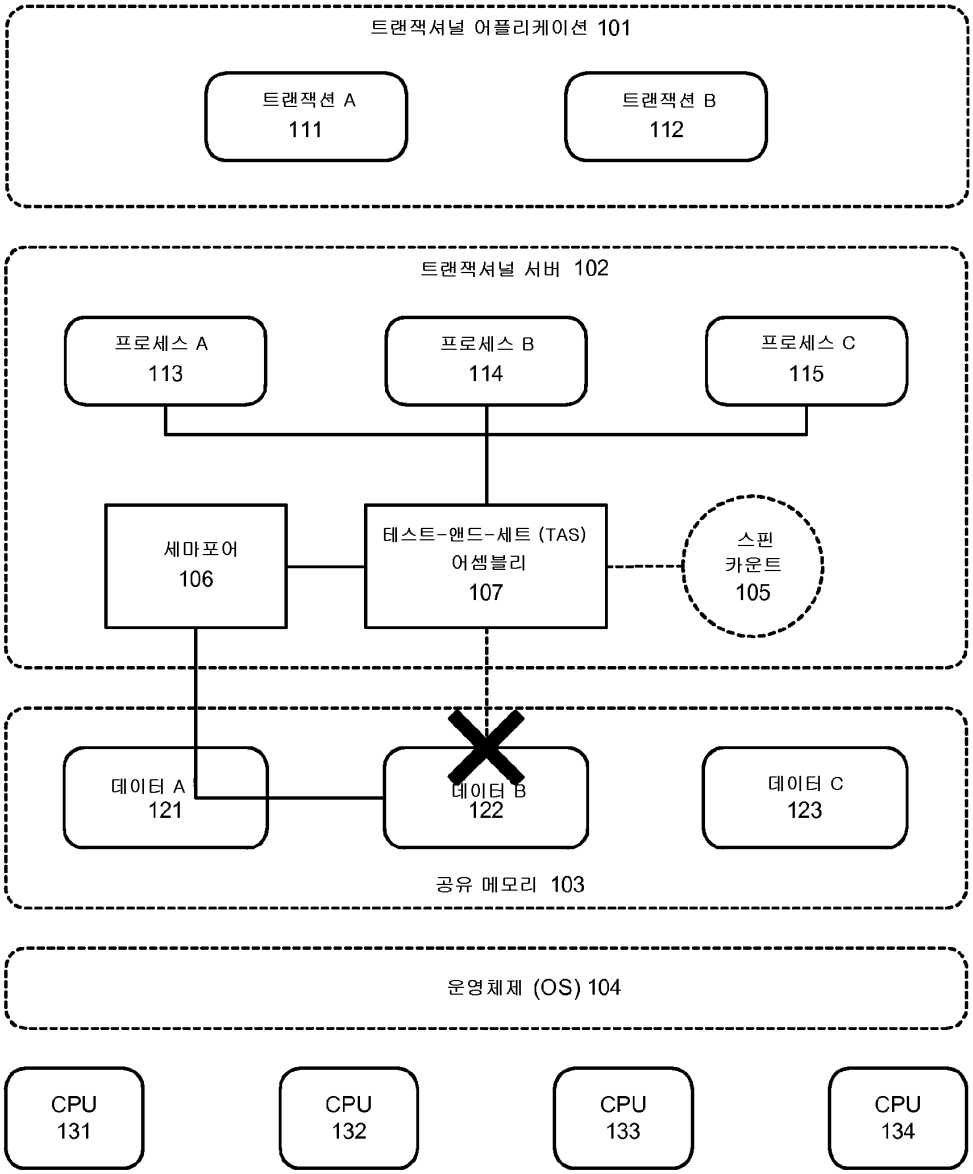
[0069] 본 발명은 본 발명의 교시들에 따라 프로그램된 하나 이상의 프로세서들, 메모리 및/또는 컴퓨터 판독가능 스토리지 매체를 포함하는 하나 이상의 종래의 범용 또는 특수 디지털 컴퓨터, 컴퓨팅 디바이스, 머신 또는 마이크로프로세서를 이용하여 통상적으로 구현될 수 있다. 적절한 소프트웨어 코딩은 소프트웨어 기술 분야의 숙련자에게 분명할 바와 같이, 본 발명의 교시들에 근거하여 숙련된 프로그래머들에 의해 쉽게 준비될 수 있다.

[0070] 일부 실시예들에서, 본 발명은 본 발명의 프로세스들 중 어느 것을 수행하도록 컴퓨터를 프로그램하는 데 사용될 수 있는/명령어들이 저장된 스토리지 매체 또는 컴퓨터 판독가능 매체(들)인 컴퓨터 프로그램 제품을 포함한다. 스토리지 매체는 이들로만 한정되는 것은 아니지만, 플로피 디스크(disk)들, 광학 디스크(disc)들, DVD, CD-ROM들, 마이크로드라이브 및 자기-광학 디스크(disk)들을 포함하는 어떤 타입의 디스크, ROM들, RAM들, EPROM들, EEPROM들, DRAM들, VRAM들, 플래시 메모리 디바이스들, 자기 또는 광학 카드들, (분자 메모리 IC들을 포함하는)나노시스템들 또는, 명령어들 및/또는 데이터를 저장하기에 적절한 어떤 타입의 매체 또는 디바이스를 포함할 수 있다.

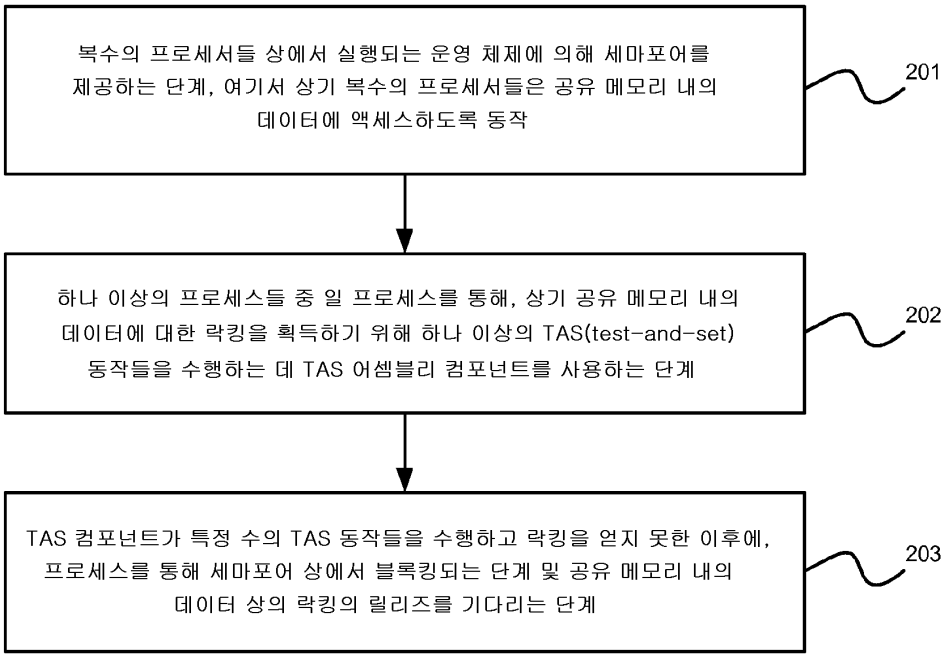
[0071] 본 발명의 상술된 상세한 설명은 예시 및 설명을 위해 제공되었다. 이는 완전한 것으로 또는 개시된 정확한 형태들에 본 발명을 제한하고자 의도된 것이 아니다. 많은 수정들 및 변형들이 이 기술분야의 숙련자에게 분명할 것이다. 실시예들은 본 발명의 원리 및 이의 실용적 응용을 가장 잘 설명하기 위해 선택 및 기술되었으며, 그럼으로써 이 기술분야의 숙련자들이 고려되는 특별한 사용에 적합한 다양한 실시예들에 대한 그리고 다양한 수정들을 포함하는 본 발명을 이해하게 할 수 있다. 본 발명의 범위는 다음의 특허 청구 범위 및 이의 균등물에 의해 정의되어야 함이 의도된다.

도면

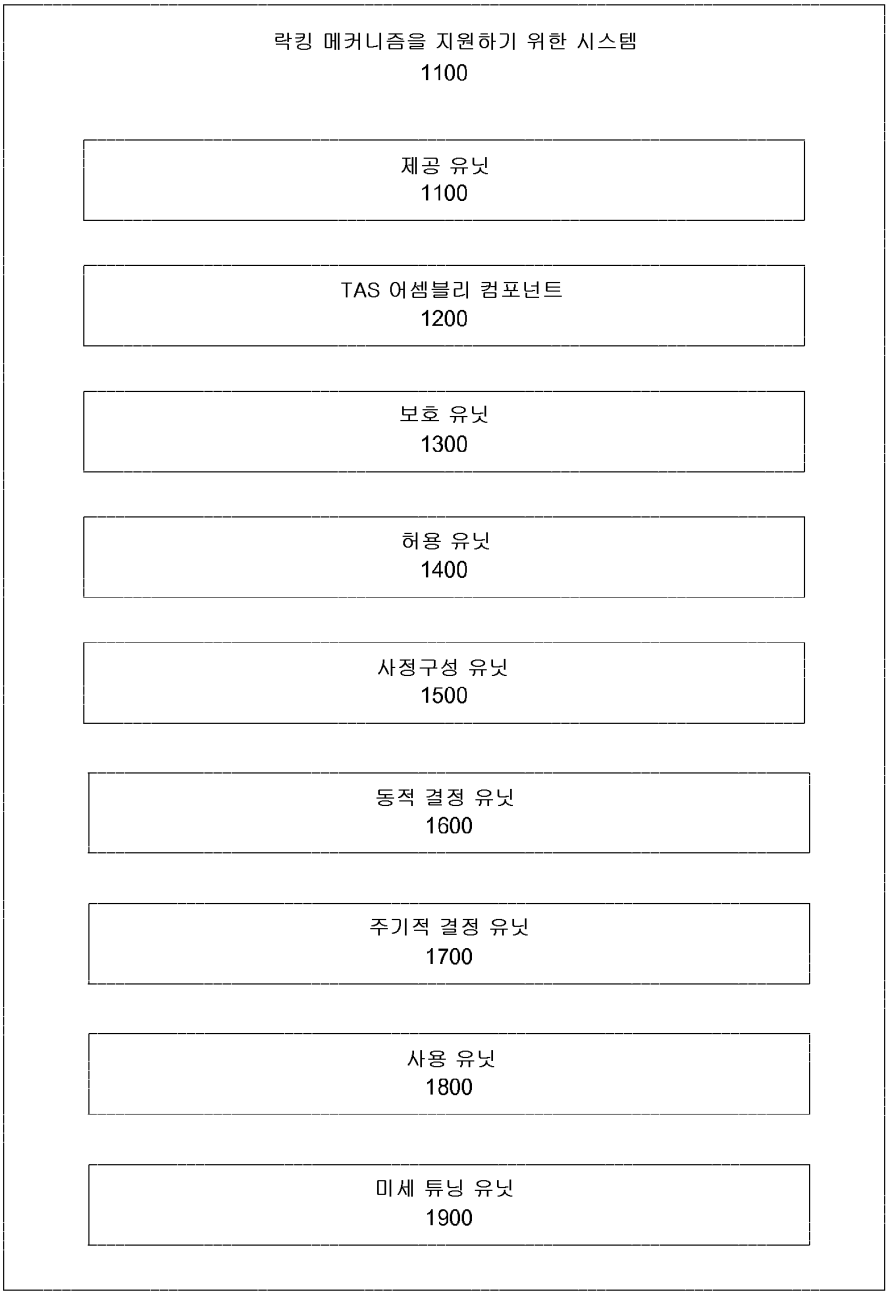
도면1


트랜잭셔널 어플리케이션 101
트랜잭션 A
111
트랜잭션 B
112
트랜잭셔널 서버 102
프로세스 A
113
프로세스 B
114
프로세스 C
115
세마포어
106
테스트-앤드-세트 (TAS)
어셈블리
107
스핀
카운트
105
데이터 A
121
데이터 B
122
데이터 C
123
공유 메모리 103
운영체제 (OS) 104
CPU
131
CPU
132
CPU
133
CPU
134

도면2


복수의 프로세서들 상에서 실행되는 운영 체제에 의해 세마포어를 제공하는 단계, 여기서 상기 복수의 프로세서들은 공유 메모리 내의 데이터에 액세스하도록 동작
201
하나 이상의 프로세스들 중 일 프로세스를 통해, 상기 공유 메모리 내의 데이터에 대한 락킹을 획득하기 위해 하나 이상의 TAS(test-and-set) 동작들을 수행하는 데 TAS 어셈블리 컴포넌트를 사용하는 단계
202
TAS 컴포넌트가 특정 수의 TAS 동작들을 수행하고 락킹을 얻지 못한 이후에, 프로세스를 통해 세마포어 상에서 블록킹되는 단계 및 공유 메모리 내의 데이터 상의 락킹의 릴리즈를 기다리는 단계
203

도면3


락킹 메커니즘을 지원하기 위한 시스템
1100
제공 유닛
1100
TAS 어셈블리 컴포넌트
1200
보호 유닛
1300
허용 유닛
1400
사정구성 유닛
1500
동적 결정 유닛
1600
주기적 결정 유닛
1700
사용 유닛
1800
미세 튜닝 유닛
1900