(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년10월01일
(11) 등록번호 10-1997816
(24) 등록일자 2019년07월02일

(51) 국제특허분류(Int. Cl.)
G06F 9/52 (2018.01) *G06F 9/46* (2006.01)
G06F 9/54 (2018.01)
(52) CPC특허분류
G06F 9/524 (2013.01)
G06F 9/46 (2013.01)
(21) 출원번호 10-2015-7014571
(22) 출원일자(국제) 2013년10월28일
심사청구일자 2018년01월19일
(85) 번역문제출일자 2015년06월01일
(65) 공개번호 10-2015-0122119
(43) 공개일자 2015년10월30일
(86) 국제출원번호 PCT/US2013/067108
(87) 국제공개번호 WO 2014/133595
국제공개일자 2014년09월04일
(30) 우선권주장
13/781,493 2013년02월28일 미국(US)
(56) 선행기술조사문헌
US20070118601 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
오라클 인터내셔날 코포레이션
미국, 캘리포니아 94065, 레드우드 쇼어스 엠에스 5오피7, 오라클 파크웨이 500
(72) 발명자
오텐코 올렉산드르
영국 위너쉬 버크셔 알쥐41 5제이비 배서스트 로드 24
(74) 대리인
박장원

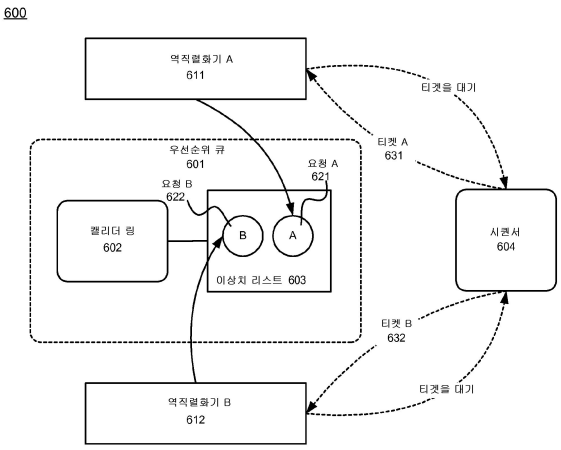
전체 청구항 수 : 총 20 항 심사관 : 임재우

(54) 발명의 명칭 **동시 우선순위 큐에서 시퀀서를 이용하는 시스템 및 방법**

(57) 요 약

시스템 및 방법은 동시 우선순위 큐를 지원할 수 있다. 상기 동시 우선순위 큐는 복수의 스레드들이 상기 우선순위 큐와 인터랙션할 수 있게 한다. 상기 우선순위 큐는 상기 우선순위 큐에서 하나 이상의 요청들에 대해 경합하는 복수의 스레드들을 검출 및 순서화하기 위해 시퀀서를 이용할 수 있다. 더욱이, 상기 우선순위 큐는 상기 복수의 스레드들 간의 경합을 감소시키는 동작을 한다.

대 표 도



(52) CPC특허분류
G06F 9/526 (2013.01)
G06F 9/546 (2013.01)

명 세 서

청구범위

청구항 1

시스템으로서,

하나 이상의 마이크로프로세서들;

복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하기 위한 우선순위 큐를 포함하며,

상기 우선순위 큐는 연속 패싱 기능을 갖는 동기화된 블록을 포함하고 그리고 상기 복수의 컨슈머들로부터의 상기 복수의 요청들의 비-차단(non-blocking) 수신을 위한 복수의 스레드들을 관리하고;

상기 우선순위 큐는 미리 구성된 시간보다 짧은 타겟 응답 시간을 갖는 상기 복수의 요청들의 제1 서브세트를 저장하도록 동작하는 캘린더 링 버퍼 메모리를 포함하는 캘린더 링 버퍼를 포함하고;

상기 우선순위 큐는 상기 미리 구성된 시간보다 긴 타겟 응답 시간을 갖는 상기 복수의 요청들의 제2 서브세트를 저장하도록 동작하는 이상치(outlier) 리스트 메모리를 포함하는 이상치 리스트를 포함하고;

상기 우선순위 큐는 상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출(first-in-first-out, FIFO) 순서를 시행하도록 동작하는 시퀀서를 포함하고; 그리고

상기 우선순위 큐는 상기 복수의 컨슈머들과 상기 복수의 요청들 간의 경합을 감소시키도록 동작하는 것을 특징으로 하는 시스템.

청구항 2

제1항에 있어서,

상기 복수의 요청들의 제1 서브세트 각각은 상기 복수의 요청들의 제1 서브세트 각각과 연관된 QoS(quality of service)의 값에 의존하는 위치에서 상기 캘린더 링 버퍼에 저장되는 것을 특징으로 하는 시스템.

청구항 3

제1항에 있어서,

상기 미리 구성된 시간은 2초인 것으로 특정되는 것을 특징으로 하는 동시 우선순위 큐를 지원하는 시스템.

청구항 4

제1항에 있어서,

상기 시퀀서는

동기화 메커니즘 또는 락 메커니즘을 요구하지 않고 상기 이상치 리스트에 저장된 상기 복수의 요청들 중 상기 제2 서브세트에 대해 선입선출 순서를 시행하기 위한 티켓 메커니즘을 구현하는 것을 특징으로 하는 시스템.

청구항 5

제1항에 있어서,

상기 우선순위 큐는 워크로드(workload)가 집약적(intensive)일 때 상기 캘린더 링 상의 경합을 감소시키기 위해 사용될 수 있는 가로채진 요청들의 리스트(list of stolen requests)를 포함하며, 상기 가로채진 요청들의 리스트는 동일한 캘린더 입력에 랜딩하는 하나 이상의 요청들을 포함하는 것을 특징으로 하는 시스템.

청구항 6

제1항에 있어서,

상기 우선순위 큐는 협동적인 동시실행을 지원하기 위해 이용되는 고속 레인을 포함하는 것을 특징으로 하는 시

스템.

청구항 7

제1항에 있어서, 상기 시퀀서는 상기 이상치 리스트에 저장된 상기 복수의 요청들 중 상기 제2 서브 세트에 대해 선입선출 순서를 시행하기 위한 티켓 메커니즘을 사용하는 것을 특징으로 하는 시스템.

청구항 8

제7항에 있어서,

상기 티켓 메커니즘은 컨슈머가 상기 우선순위 큐에 액세스를 시도할 때마다 증가되는 리더 카운트(reader count)를 유지하는 것을 특징으로 하는 동시 우선순위 큐를 지원하는 시스템.

청구항 9

제7항에 있어서,

상기 시퀀서는 상기 복수의 컨슈머들 중 하나의 컨슈머에게 티켓을 발행하도록 동작하고, 상기 티켓은 상기 컨슈머가 상기 우선순위 큐에 액세스하는 것을 허용하는 것을 특징으로 하는 시스템.

청구항 10

제7항에 있어서,

상기 티켓 메커니즘은 상기 복수의 컨슈머들 중 하나의 컨슈머가 상기 우선순위 큐에 액세스하려고 시도할 때 증가되는 리더 카운트(reader count)를 유지하고; 그리고 상기 리더 카운트는 상기 복수의 컨슈머들과 상기 복수의 요청들 사이의 경합을 검출하는데 사용되는 것을 특징으로 하는 시스템.

청구항 11

미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법으로서, 상기 방법은,

상기 복수의 컨슈머들로부터의 상기 복수의 요청들의 비-차단(non-blocking) 수신을 위한 복수의 스레드들을 관리하는 연속 패싱 기능을 갖는 동기화 블록을 사용하여 상기 복수의 컨슈머들로부터 상기 복수의 요청들을 수신하는 단계;

미리 구성된 시간보다 짧은 타겟 응답 시간을 갖는 상기 복수의 요청들의 제1 서브세트를 캘린더 링 버퍼에 저장하는 단계;

상기 미리 구성된 시간보다 긴 타겟 응답 시간을 갖는 상기 복수의 요청들의 제2 서브세트를 이상치(outlier) 리스트에 저장하는 단계;

우선순위 큐에 포함되는 시퀀서를 사용하여 상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출(first-in-first-out, FIFO) 순서를 시행하는 단계; 그리고

상기 복수의 컨슈머들과 상기 복수의 요청들 간의 경합을 감소시키는 단계를 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 12

제11항에 있어서,

상기 복수의 요청들의 제1 서브세트 각각과 연관된 QoS(quality of service)의 값에 의존하는 위치에서 상기 복수의 요청들의 제1 서브세트를 상기 캘린더 링 버퍼에 저장하는 단계를 더 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 13

제12항에 있어서,

상기 미리 구성된 시간은 2초로 특정되는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터

수신된 복수의 요청들을 우선순위화하는 방법.

청구항 14

제11항에 있어서,

동기화 메커니즘 또는 락 메커니즘을 요구하지 않고 상기 이상치 리스트에 저장된 상기 복수의 요청들 중 상기 제2 서브세트에 대해 선입선출 순서를 시행하기 위해 상기 시퀀서에서 티켓 메커니즘을 구현하는 단계를 더 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 15

제11항에 있어서,

워크로드(workload)가 집약적(intensive)일 때 상기 캘린더 링 상의 경합을 감소시키기 위해 사용될 수 있는 가로채진 요청들의 리스트(list of stolen requests)를 저장하는 단계를 더 포함하며, 상기 가로채진 요청들의 리스트는 동일한 캘린더 입력에 랜딩하는 하나 이상의 요청들을 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 16

제11항에 있어서,

협동적인 동시실행을 지원하는 고속 레인을 제공하는 단계를 더 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 17

제11항에 있어서,

상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브 세트에 대해 선입선출 순서를 시행하기 위해 상기 시퀀서에서 티켓 메커니즘을 구현하는 단계; 그리고

상기 티켓 메커니즘을 이용하여 컨슈머가 상기 우선순위 큐에 액세스를 시도할 때마다 증가되는 리더 카운트(reader count)를 유지하는 단계를 더 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 18

제11항에 있어서,

상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출 순서를 시행하기 위해 상기 시퀀서에서 티켓 메커니즘을 구현하는 단계; 그리고

상기 복수의 컨슈머들 중 하나의 컨슈머에게 티켓을 발행하는 단계를 더 포함하며,

상기 티켓은 상기 컨슈머가 상기 이상치 리스트에 액세스하는 것을 허용하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하는 방법.

청구항 19

제11항에 있어서,

상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출 순서를 시행하기 위해 상기 시퀀서에서 티켓 메커니즘을 구현하는 단계;

컨슈머가 상기 우선순위 큐에 액세스하려고 시도할 때마다 증가되는 리더 카운트(reader count)를 상기 티켓 메커니즘을 이용하여 유지하는 단계; 그리고

상기 복수의 컨슈머들과 상기 복수의 요청들 간의 경합을 검출하기 위해 상기 리더 카운트를 사용하는 단계를 더 포함하는 것을 특징으로 하는 미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선

순위화하는 방법.

청구항 20

미들웨어 머신 환경에서 복수의 컨슈머들로부터 수신된 복수의 요청들을 우선순위화하기 위한 명령어들을 저장하는 비 일시적인 컴퓨터 판독 가능 매체로서,

상기 명령어들은 컴퓨터에 의해 실행될 때 상기 컴퓨터로 하여금 단계들을 수행하게 하며, 상기 단계들은,

상기 복수의 컨슈머들로부터의 상기 복수의 요청들의 비-차단(non-blocking) 수신을 위한 복수의 스레드들을 관리하는 연속 패싱 기능을 갖는 동기화 블록을 사용하여 상기 복수의 컨슈머들로부터 상기 복수의 요청들을 수신하는 단계;

미리 구성된 시간보다 짧은 타겟 응답 시간을 갖는 상기 복수의 요청들의 제1 서브세트를 캘린더 링 버퍼에 저장하는 단계;

상기 미리 구성된 시간보다 긴 타겟 응답 시간을 갖는 상기 복수의 요청들의 제2 서브세트를 이상치(outlier) 리스트에 저장하는 단계;

시퀀서를 사용하여 상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출(first-in-first-out, FIFO) 순서를 시행하는 단계; 그리고

상기 복수의 컨슈머들과 상기 복수의 요청들 간의 경합을 감소시키는 단계를 포함하는 것을 특징으로 하는 비 일시적인 컴퓨터 판독 가능 매체.

청구항 21

삭제

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 반대하지 않지만, 그 밖의 모든 것은 저작권으로 보호된다.

[0003] 기술분야

[0004] 본 발명은 일반적으로, 미들웨어와 같은 컴퓨터 시스템들 및 소프트웨어에 관한 것이며, 특히 미들웨어 머신 환경에서 큐를 지원하는 시스템들 및 방법들에 관한 것이다.

배 경 기 술

[0005] 수년의 기간에 걸쳐, 어떤 거대 조직이면 이 조직은 종종, 여러 가지 다양한 컴퓨터 하드웨어, 운영 체제들, 및 어플리케이션 소프트웨어를 망라하는 산개한 IT 인프라스트럭쳐(sprawling IT infrastructure)를 알게 된다. 비록, 이러한 인프라스트럭쳐의 각각의 개별적인 컴포넌트가 자체적으로 잘 엔지니어링되어 있고 잘 유지되어 있다 하더라도, 이러한 컴포넌트들을 상호연결하거나 또는 공통 리소스들을 공유하기를 시도할 때, 이 시도는 종종 어려운 관리 태스크(difficult administrative task)일 수 있다. 최근에, 조직들은 가상화 및 중앙집중식 스토리지와 같은 기술들에 관심을 돌렸으며, 더욱 최근에는, 공유 인프라스트럭쳐에 대한 토대를 제공할 수 있는 클라우드 컴퓨팅에 관심을 돌렸다. 그러나, 이러한 환경들에서 사용하기에 특히 적합한 올-인-원 플랫폼(all-in-one platform)들은 거의 존재하지 않는다. 이들이 본 발명의 실시예들이 해결하고자 하는 일반적인 영역들이다.

발명의 내용

[0006] 동시 우선순위 큐를 지원하는 시스템들 및 방법들이 제공된다. 상기 동시 우선순위 큐는 복수의 스레드들이 상기 우선순위 큐와 인터랙션(interation)할 수 있게 한다. 상기 우선순위 큐는 상기 우선순위 큐에서 하나 이상의 요청들에 대해 경합하는 복수의 스레드들을 검출 및 순서화(order)하기 위해 시퀀서를 사용할 수 있다. 더욱이, 상기 우선순위 큐는 복수의 스레드들 간의 경합을 감소시키는 동작을 한다.

[0007] 첨부 도면들에 비추어 본 명세서를 읽을 때, 다양한 실시예들의 다음의 상세한 설명으로부터 본 발명의 다른 목적들 및 장점들이 이 기술 분야의 통상의 지식을 가진 자에게 분명할 것이다.

도면의 간단한 설명

[0008] 도 1은 본 발명의 실시예에 따른 미들웨어 머신 환경(100)의 예를 도시한다.

도 2는 본 발명의 실시예에 따른 미들웨어 머신 플랫폼 또는 환경의 다른 예를 도시한다.

도 3은 본 발명의 다양한 실시예들에 따른 미들웨어 머신 환경에서 요청들을 처리(handling)하기 위해 우선순위 큐를 이용하는 예를 도시한다.

도 4는 미들웨어 머신 환경에서 비-차단 큐(non-blocking queue)를 지원하는 예를 도시한다.

도 5는 본 발명의 다양한 실시예들에 따른 동시 우선순위 큐(concurrent priority queue)를 지원하는 예를 도시한다.

도 6은 본 발명의 다양한 실시예들에 따른 이상치 리스트에 대한 선입선출(FIFO) 순서(order)를 보장하는 예를 도시한다.

도 7은 본 발명의 실시예들에 따른 동시 우선순위 큐에서 이상치 리스트 내로 복수의 요청들을 추가하기 위한 예시적인 인터랙티브 챠트를 도시한다.

도 8은 본 발명의 다양한 실시예들에 따른 동시 우선순위 큐의 서로 다른 컨슈머들 간의 경합을 검출하는 예를 도시한다.

도 9는 본 발명의 실시예에 따른, 경합이 검출될 때 희생자와 경합자 간의 인터랙션을 예시하는 예시적인 인터랙티브 챠트이다.

도 10은 본 발명의 실시예에 따른, 어떤 경합도 검출되지 않을 때 희생자와 경합자 간의 인터랙션을 예시하는 예시적인 인터랙티브 챠트이다.

도 11은 본 발명의 실시예에 따른 우선순위 큐에서 협동적인 동시실행을 지원하기 위한 예시적인 순서도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0009] 클러스터에서 작업 공유 먹싱(work sharing muxing)을 지원할 수 있는 시스템들 및 방법들이 본 명세서에 개시된다.

[0010] 도 1은 본 발명의 실시예에 따른 미들웨어 머신 환경(100)의 예를 도시한다. 도 1에 도시된 바와 같이, 각각의 미들웨어 머신 시스템(102)은 여러 미들웨어 머신 랙(rack) 컴포넌트들(104)을 포함하며, 이들 각각은 고성능 미들웨어 머신 하드웨어 노드들(106)(예컨대, 64-비트 프로세서들, 고성능 대용량 메모리, 및 리던던트 인피니밴드 및 이더넷 네트워킹) 및 미들웨어 머신 소프트웨어 환경(108)의 조합을 포함한다. 그 결과, 일단위 또는 월단위보다는 분단위로 프로비져닝되고(provisioned) 요구에 따라(on demand) 스케일링할 수 있는 완전한 어플리케이션 서버 환경이 이루어진다. 일 실시예에 따르면, 각각의 미들웨어 머신 시스템은 전체(full), 절반(half) 또는 1/4(quarter) 랙으로 또는 랙 컴포넌트들의 다른 구성으로 배치(deploy)될 수 있으며, 여러 미들웨어 머신 시스템들이 더 거대한 환경들을 생성하기 위해 인피니밴드를 다시 이용하여, 함께 결합될 수 있다. 각각의 미들웨어 머신 소프트웨어 환경은 여러 어플리케이션 서버 또는 다른 소프트웨어 인스턴스들로 프로비져닝될 수 있다. 예를 들어 도 1에 도시된 바와 같이, 어플리케이션 서버 인스턴스(109)는 가상 머신(116), 운영 체제(120), 가상화 계층(124) 및 어플리케이션 서버 계층(128)(예컨대, 서블릿(servlet)(132), EJB(134) 및 그리드링크(136) 컨테이너들을 포함하는 웹로직)을 포함할 수 있다. 다른 어플리케이션 서버 인스턴스(110)는 가상

머신(118), 운영 체제(122), 가상화 계층(126) 및 데이터 그리드 계층(140)(예컨대, 활성 캐시(active cache)(142)를 포함하는 응집(coherence))을 포함할 수 있다. 인스턴스들 각각은 서로와 통신할 수 있고 엑사로직 통합 팩(ExaLogic integration pack)과 같은 미들웨어 머신 통합 컴포넌트(150)을 이용하여 자신의 미들웨어 머신 하드웨어 노드 및 다른 노드들과 통신할 수 있는 바, 상기 미들웨어 머신 통합 컴포넌트는 하기에 더욱 상세히 기술되는 바와 같이, 인피니밴드 및 다른 특징들을 지원하는 것과 같은 여러 최적화 특징들을 자체적으로 제공한다.

[0011] 도 2는 본 발명의 실시예에 따른 미들웨어 머신 플랫폼 또는 환경의 다른 예를 도시한다. 도 2에 도시된 바와 같이, 각각의 어플리케이션 서버 인스턴스는 미들웨어 머신 환경 내의 전송기 및/또는 수신기(160, 161)로서 역할을 할 수 있다. 각각의 어플리케이션 서버 인스턴스는 또한, 먹서(muxer)(162, 163)와 관련되며, 상기 먹서는 어플리케이션 서버들이 인피니밴드 네트워크(164)를 통해 서로와 통신하게 할 수 있다. 도 2에 도시된 예에서, 어플리케이션 서버 인스턴스는 커널 공간(165), 사용자 공간(167) 및 어플리케이션 서버(예컨대, 웹로직 공간)(166)을 포함할 수 있고, 상기 어플리케이션 서버는 또한, 소켓 다이렉트 프로토콜(sockets direct protocol)(168), JVM(예컨대, JRockit/Hotspot 계층)(170), WLS 코어(172), 서블릿 컨테이너(174) 및 JSP 컴파일러(176)와 관련될 수 있다. 다른 예들에 따르면, 미들웨어 타입 소프트웨어의 다른 조합들이 포함될 수 있다. 다양한 실시예들에 따르면, 머신 통합 컴포넌트는 공유 인프라스트럭처에 대한 토대를 제공하고 상기 공유 인프라스트럭처 내의 성능을 향상시키기 위해, 제로 버퍼 카피들(Zero Buffer Copies), 스캐터/개더 I/O(Scatter/Gather I/O), T3 커넥션들, 레이지 역직렬화(Lazy Deserialization) 및 그리드링크 데이터소스와 같은 특징들(180)을 제공할 수 있다.

[0012] 우선순위 큐

[0013] 본 발명의 다양한 실시예들에 따르면, 동시 시스템은 적절한 서비스 수준 협약(SLA)으로 서비스를 제공하기 위해 유입 요청들을 우선순위화하는 데 우선순위 큐를 이용할 수 있다.

[0014] 도 3은 본 발명의 다양한 실시예에 따른 미들웨어 머신 환경에서 요청들을 처리하기 위해 우선순위 큐를 이용하는 예를 도시한다. 도 3에 도시된 바와 같이, 하나 이상의 스레드들, 예컨대 역직렬화 스레드(deserializing thread)들 A 내지 B(311 내지 312)은 하나 이상의 요청들(320)을 포함하는 유입 네트워크 트래픽(310)을 역직렬화할 수 있다. 역직렬화 스레드들 A 내지 B(311 내지 312)은 예컨대, add() 방법들을 이용하여 우선순위 큐(301)에 요청들(320)을 위치시킬 수 있다. 그 다음, 복수의 작업자 스레드들, 예컨대 작업자 스레드들 A 내지 C(321 내지 323)은 우선순위 큐(301)에 동시적으로 액세스할 수 있고, 예컨대, delete_min() 방법들을 이용하여 요청들(320)에 대한 권한을 주장할 수 있다.

[0015] 우선순위 큐(301)는 요구하는 동시실행 기준(demanding concurrency criteria)을 충족하도록 설계될 수 있어서, 경합자들 사이의 인터랙션(interaction)이 전체적으로 시스템의 스루풋(throughput)의 저하(degradation)를 야기하지 않는다. 추가적으로, 우선순위 큐(301)는 고정된 사이즈의 메모리 풋프린트(footprint)를 갖도록 구현될 수 있어서, JVM은 기초 요소(primitives)의 고정된 배열들 상에서 자신의 동작들을 양호하게 최적화할 수 있고, 실질적인 캐시 효율성을 달성할 수 있다.

[0016] 본 발명의 다양한 실시예들에 따르면, 우선순위 큐(301)는 캘린더 큐, 예컨대 웹로직 어플리케이션 서버에 제공된 캘린더 큐에 근거하여 구현될 수 있다. 상기 캘린더 큐는 복수의 버킷(bucket)들을 갖는 캘린더를 포함할 수 있고, 상기 버킷들 각각은 특별한 시간 슬라이스 내에 속하는 이벤트들을 저장할 수 있다. 예를 들어, 복수의 버킷들은 현재 시간과 타겟 서비스를 비교함으로써 저장 및 배열될 수 있다. 시간의 차이가 제1 바이트에 존재하면, 요청은 제1의 256개의 버킷들 내의 버킷에 저장될 수 있다. 특정 버킷이 요청을 실행하기 위해 타겟 시간의 실제 값을 이용하여 선택될 수 있다. 더욱이, 시간의 차이가 제2 바이트에 존재하면, 요청은 제2의 256개의 버킷들 내의 버킷에 저장될 수 있다.

[0017] 컨슈머가 예컨대, 작업자 스레드들 A 내지 C(321 내지 323) 중 가장 빠른 요청을 실행하도록 구성된 일 작업자 스레드를 통해 다음 요청을 제거할 것을 시도하면, 시스템은 비어있지 않은 제1 버킷에 대해 캘린더를 스캔할 수 있다. 이 버킷이 제1의 256개의 버킷들 중 하나가 아니면, 캘린더 큐는 루프를 이용하고 상기 제1의 256개의 버킷들 쪽으로 "한 레벨 아래의" 버킷들에 요청들을 이동시키는 방법을 프로모션(promotion)할 수 있다. 결국, 일부 요청들이 제1의 256개의 버킷들 내의 하나 이상의 버킷들로 프로모션될 수 있고, 컨슈머는 요청에 대한 권한을 주장하고 이에 따라 진행할 수 있다.

[0018] 상기 프로모션 프로세스는 시스템의 전반적인 성능에 영향을 가질 수 있는 로그 비용(logarithmic cost)을 수반

할 수 있다. 추가적으로, 캘린더 큐에 대한 다른 설계들이 존재할 수 있고, 이의 성능은 "O(1) add, O(logN) delete_min"과 "O(logN) add, O(1) delete_min" 사이의 선택에 제한될 수 있다.

[0019] 도 4는 미들웨어 머신 환경에서 비-차단 큐를 지원하는 예를 도시한다. 도 4에 도시된 바와 같이, 복수의 컨슈머들, 예컨대 컨슈머들 A 내지 B(411 내지 412)은 미들웨어 머신 환경(400)에서 우선순위 큐(401)에 동시에 액세스할 수 있다. 우선순위 큐(401)는 비-차단 큐로서 구현될 수 있고, 요청 관리자(402)를 통해 액세스될 수 있다.

[0020] 스레드 풀(403)을 관리하는 요청 관리자(402)는 서로 다른 요청들과 서로 다른 스레드들을 관련시키기 위한 별개의 로직을 가질 수 있다. 예를 들어, 요청 관리자(402)는 락 메커니즘(lock mechanism)을 이용하여, 동기화된 스테이트먼트(synchronized statement) 또는 동기화된 블록(410)에서 우선순위 큐(401)에 대해 호출들을 래핑(wrapping)함으로써 모든 스레드 풀 방법 호출들을 직렬화할 수 있다.

[0021] 따라서, 우선순위 큐(401)에서의 동작들은 상기 직렬화가 비-차단 우선순위 큐(401) 밖에서 행해지기 때문에 단일-스레드 설계(single-threaded design)에 의해 제한될 수 있다.

[0022] 동시 우선순위 큐

[0023] 도 5는 본 발명의 다양한 실시예들에 따른 동시 우선순위 큐를 지원하는 예를 도시한다. 도 5에 도시된 바와 같이, 복수의 컨슈머들, 예컨대 컨슈머 A 내지 C(511 내지 513)는 미들웨어 머신 환경(500)에서 동시 우선순위 큐(501)에 동시적으로 액세스할 수 있다.

[0024] 동시 우선순위 큐(501)는 유입 요청들을 우선순위화 및 저장할 수 있는 캘린더, 예컨대 캘린더 링(502)을 포함할 수 있다. 사이즈가 제한된 캘린더 링(502)은 미리구성된 시간 제한 내에 타겟 응답 시간을 가지는 요청들을 저장하도록 구성될 수 있다. 캘린더 링(502) 내에서, 요청이 링 버퍼 내에서 상기 요청의 서비스 품질(QoS)(예컨대, 타겟 서비스 시간)과 매치되는 위치에 바로(directly) 저장 또는 위치될 수 있다.

[0025] 따라서, 시스템은 캘린더 큐의 메모리 풋프린트를 변경함이 없이 요청들에 대해 훨씬 저렴한 룩업을 달성할 수 있다. 더욱이, 시스템은 캘린더 큐의 delete_min 동작의 로그 복잡도(logarithmic complexity)를 거의 선형 캐시 효율적 검색(linear cache efficient search)으로 감소시키면서도, O(1) 동작들로서 캘린더 큐에 요소들을 추가하는 것을 유지할 수 있다.

[0026] 추가적으로, 미리 구성된 시간 제한보다 긴 타겟 서비스 시간을 갖는 요청이 이상치(outlier)들의 리스트, 예컨대 이상치 리스트(504)에 추가될 수 있다. 이 요청들의 스케줄링이 시간에 중대성이 있는(time critical) 것이 아닐 수 있기 때문에, 시스템은 분류된 이상치들의 리스트(504)로의 더 느린 추가를 허용한다. 더욱이, 동시적인 우선순위 큐(501)는 동일한 QoS를 갖는 이상치 리스트에 대해 선입선출(FIFO) 순서(order)를 집행(enforce)하도록 시퀀서(sequencer), 예컨대 outliers_seq를 이용할 수 있다.

[0027] 예를 들어, 캘린더 링(502)은 2초 미만의 타겟 응답 시간(또는 QoS)을 갖는 요청들을 저장하도록 구성될 수 있는 바, 그 이유는 2초 초과의 QoS를 갖는 요청들은 드문 것으로 고려될 수 있기 때문이다. 더욱이, 2초 미만의 QoS를 갖는 요청들은 QoS와 매치되는 캘린더 링(502)에 위치될 수 있고, 2초 초과의 QoS를 갖는 요청들은 이상치들의 리스트(504)에 위치될 수 있다.

[0028] 도 4에 도시된 캘린더 큐와 달리, 요청 관리자(510)는 동기화된 스테이트먼트에 캘린더 큐(501)에 대한 매 호출을 삽입할 필요가 없다. 연속-패싱(507)을 지원하는 동기화된 블록(506)은 동시 우선순위 큐(501)의 범위 내에서 구현될 수 있다. 컨슈머들, 예컨대 컨슈머들 A 내지 C(511 내지 513)는 동시 우선순위 큐(501) 외부로부터 스레드 풀(520)에 액세스하는 것은 필요로 하지 않는다.

[0029] 연속-패싱을 이용하여, 시스템은 비-차단하는 것으로부터 차단하는 것으로 캘린더 큐(504)를 변형할 수 있다. 연속-패싱(507)은 컨슈머들 A 내지 C(511 내지 513)이 스레드 풀(520)에서 유휴 작업자(idle workder)들 또는 스레드들(530)을 관리할 수 있게 하며, 따라서 스레드 풀(520)에서 대기할 수 있는 스레드들(530)이 재사용될 수 있다.

[0030] 추가적으로, 동시 우선순위 큐(501)는, 동시 우선순위 큐(501)가 경합을 검출할 수 있게 하는 시퀀서(503)를 포함할 수 있고, 협력적인 동시실행을 지원하기 위해 고속 레인(505)을 이용할 수 있다. 따라서, 동시 우선순위 큐(501)는 락(lock)들이 경합에 관한 지식을 노출하는 것을 필요로 함이 없이 경합을 인지하고 있고 적절하게 처리할 수 있다.

[0031] 이상치 리스트에 대한 FIFO 유지

[0032] 도 6은 본 발명의 다양한 실시예들에 따른 이상치 리스트에 대한 선입선출(FIFO) 순서를 보장하는 예를 도시한다. 도 6에 도시된 바와 같이, 미들웨어 머신 환경(600)에서의 우선순위 큐(601)는 캘린더 링(602) 및 이상치 리스트(603)를 포함할 수 있다.

[0033] 서로 다른 스레드들 상의 복수의 호출자들, 예컨대 역직렬화기 A 내지 B(611 내지 612)은 우선순위 큐(601)에 동시에 액세스하는 것을 시도할 수 있다. 우선순위 큐(601)는 이상치 리스트(603)에 대한 선입선출(FIFO) 순서를 보장하기 위해 티켓 메커니즘에 근거한 시퀀서(604)를 사용할 수 있다.

[0034] 예를 들어, 호출자, 예컨대 역직렬화기 A(611)가 이상치 리스트(603)에 요청 A(621)을 추가할 것을 허용받기 전에, 역직렬화기 A(611)는 먼저, 티켓을 요청하는 메시지를 시퀀서(604)에 전송할 수 있다. 시퀀서(604)는 어떤 경합도 존재하지 않는 경우 역직렬화기 A(611)에 티켓, 예컨대 티켓 A(631)을 발행할 수 있다.

[0035] 더욱이, 다른 호출자, 예컨대 역직렬화기 B(612)는 이상치 리스트(603) 내로 다른 요청, 예컨대 요청 B(622)를 추가하는 것을 시도할 수 있다. 시퀀서(604)는 역직렬화기 B(612)로부터 티켓에 대한 다른 요청을 수신할 수 있다. 이 시간에, 역직렬화기 A(611)가 이상치 리스트(603) 내로 요청 A(621)을 추가하기 때문에, 시퀀서(604)는 역직렬화기 B(612)를 차단할 수 있다.

[0036] 역직렬화기 A(611)가 이상치 리스트(603) 내로 요청 A(621)을 추가하는 것을 완료한 후에, 역직렬화기 A(611)는 시퀀서(604)를 전진시킬 수 있다. 그 다음, 시퀀서(604)는 역직렬화기 B(612)에 티켓, 예컨대 티켓 B(632)을 발행할 수 있는 바, 상기 티켓은 역직렬화기 B(612)가 이상치 리스트(603) 내로 요청 B(622)를 요청 A(621) 후에 추가할 수 있게 한다.

[0037] 도 7은 본 발명의 실시예에 따른 동시 우선순의 큐에서 이상치 리스트 내로 복수의 요청들을 추가하기 위한 예시적인 인터랙티브 챠트를 도시한다. 도 7에 도시된 바와 같이, 시퀀서(710)는 우선순의 큐에 동시에 액세스하는 것을 시도할 수 있는 복수의 호출자들 또는 스레드들, 예컨대 역직렬화기들 A 내지 B(711 내지 722)를 편성(coordinate)할 수 있다.

[0038] 단계(701)에서, 역직렬화기 A(711)는 시퀀서(710)로부터 티켓을 획득하는 것을 시도할 수 있다. 단계(702)에서, 시퀀서(710)는 티켓 메커니즘을 체크할 수 있다. 이것이 성공하면, 우선순위 큐에 액세스하는 것을 시도하는 어떤 다른 호출자도 존재하지 않는다. 그렇지 않으면, 역직렬화기 A(711)는 다른 호출자가 이를 릴리즈할 때까지 차단될 수 있다.

[0039] 그 다음, 단계(703)에서, 다른 호출자, 역직렬화기 B(712)는 이상치 리스트에 요청을 성공적으로 추가할 수 있고, 역직렬화기 A(711)는 차단된다. 단계(704)에서, 역직렬화기 B는 시퀀서(710)를 진전시키는 것을 시도할 수 있다.

[0040] 단계(705)에서, 시퀀서(710)가 진전하라는 메시지를 수신한 후, 시퀀서(710)는 티켓을 생성하여, 대기중인 역직렬화기 A(711)에 발행할 수 있다. 결론적으로, 단계(706)에서, 역직렬화기 A(711)는 티켓을 수신하며, 단계(707)에서, 역직렬화기 A(711)는 이상치 리스트 내로 다른 요청을 추가하는 것을 진행할 수 있다.

[0041] 따라서, 시퀀서(710)를 이용하여, 이상치 리스트의 FIFO 순서는 동기화 또는 락 메커니즘을 구현하는 것을 필요로 함이 없이 보존(preserve)될 수 있다.

[0042] 서로 다른 컨슈머들 간의 경합 검출 및 처리

[0043] 도 8은 본 발명의 다양한 실시예들에 따른 동시 우선순위 큐의 서로 다른 컨슈머들 간의 경합을 검출하는 예를 도시한다. 도 8에 도시된 바와 같이, 서로 다른 컨슈머들, 예컨대 컨슈머들 A 내지 B(811 내지 812)는 미들웨어 머신 환경(800)에서 우선순위 큐(801)에 동시에 액세스할 수 있다. 우선순위 큐(801)는 캘린더 링(802), 이상치 리스트(803) 및 고속 레인(804)을 포함할 수 있다.

[0044] 각각의 컨슈머는 동시성(concurrency)을 더 감소시키기 위해 가로채진 요청들의 리스트(list of stolen requests)의 장점을 취할 수 있다. 예를 들어, 컨슈머 A(811)는 컨슈머 A(811)에 대한 로컬 리스트로서 나타날 수 있는 가로채진 요청들의 리스트(807)를 사용할 수 있다. 추가적으로, 컨슈머들 A 내지 B(811 내지 812) 각각은 자신이 우선순위 큐(801)에 액세스하도록 허용되기 전에 티켓에 대해 요청하는 메시지를 시퀀서(805)에 전송할 수 있다.

[0045] 시퀀서(805)는 시퀀서(805)로부터 티켓을 요청한 리더(reader)들의 총 수의 현재 카운트인 리더 카운트(806)를

유지할 수 있다. 시퀀서(805)는 티켓 요청을 수신할 때마다 리더 카운트(806)를 증가시킬 수 있다. 더욱이, 이 리더 카운트(806)는 경합을 검출하기 위해 사용될 수 있다.

[0046] 예를 들어, 컨슈머 A(811)는 자신이 우선순위 큐(801)에 액세스하도록 허용되기 전에 시퀀서(805)로부터 티켓, 예컨대 티켓 A(831)을 획득할 수 있다. 그 다음, 다른 컨슈머 B(812)는 시퀀서(805)에 티켓, 예컨대 티켓 B(832)에 대한 요청을 전송할 수 있다. 시퀀서(805)는 리더 카운트(806)를 증가시킬 수 있고, 컨슈머 A(811)가 자신의 동작을 완료할 때까지 컨슈머 B(812)를 차단할 수 있다.

[0047] 한편, 컨슈머 A(811)는 시퀀서(805)에서의 현재 리더 카운트(806), t가 자신이 갖고 있는 티켓 번호를 초과함(t > 티켓 번호)을 검출할 때, 컨슈머 B(812)로부터 경합을 검출할 수 있다. 그 다음, 컨슈머 A(811)는 고속 레인(804)에 요청을 배치하고 이를 컨슈머 B(812)에 의해 소모될 수 있게 함으로써 경합을 감소시킬 수 있는 바, 이는 협력적인 동시실행 전략(cooperative concurrency strategy)으로 지칭될 수 있다.

[0048] 따라서, 시퀀서(805)를 이용하여, 우선순위 큐는 락 또는 동기화 메커니즘을 필요로 함이 없이, 컨슈머들이 캘린더 링(802)에 추가적으로 고속 레인(804) 및 가로채진 요청들의 리스트(804)에 액세스할 수 있게 함으로써, 복수의 컨슈머들 간의 경합을 처리하기 위해 최적화될 수 있다.

[0049] 도 9는 본 발명의 실시예에 따른, 경합이 검출될 때 희생자와 경합자 간의 인터랙션을 도시하는 예시적인 인터랙티브 챠트이다. 도 9에 도시된 바와 같이, 시퀀서(920)는 동시 우선순위 큐의 서로 다른 컨슈머들, 예컨대 희생자(922)와 경합자(921) 간의 경합들을 검출 및 처리하기 위해 사용될 수 있다.

[0050] 단계(901)에서, 경합자(921)는 자신이 티켓을 기다리고 있음을 나타내는 메시지를 시퀀서(920)에 전송할 수 있다. 희생자(922)가 우선순위 큐에 현재 액세스하고 있기 때문에, 단계(902)에서 시퀀서(920)는 경합자(921)를 차단하고, 리더 카운트, 리더들을 증가시킬 수 있다.

[0051] 단계들(903 내지 905)에서, 희생자(922)는 동시 우선순위 큐로부터 요청을 픽업(pick up)하고, 현재 리더들에 대해 시퀀서(92)를 체킹할 수 있다(단계들(903 내지 904)은 또한, 비-시간적 천이(non-temporal transition)로서 지칭될 수 있으며, 이는 차후의 섹션에서 논의될 것이다).

[0052] 그 다음, 단계(906)에서, 희생자(922)는 획득된 리더 카운트, t와 티켓 번호를 비교할 수 있는 바, 상기 티켓 번호는 자신이 갖고 있는 티켓 번호이다. 시퀀서(920)가 리더 카운트를 이미 증가시켰기 때문에, 희생자(922)는 t > 티켓 번호임을 발견할 때 경합을 검출할 수 있다.

[0053] 후속적으로, 단계들(907 내지 908)에서, 희생자(922)는 고속 레인에 요청을 배치하고, 희생자(922)가 단계(909)에서 리더 시퀀스를 진전시키는 것을 시도하기 전에 고속 레인 내의 요청 카운트, fastLane_w를 갱신할 수 있다(단계들(907 내지 908)은 또한, 비-시간적 천이(914)로서 지칭될 수 있으며, 이는 차후의 섹션에서 논의될 것이다).

[0054] 리더 시퀀스를 진전시키기 위해 희생자(922)로부터 메시지를 수신한 후, 단계(910)에서 시퀀서(920)는 경합자(921)에 티켓을 발행하는 것을 진행할 수 있고, 이 티켓은 경합자(921)를 릴리즈시킨다.

[0055] 마지막으로, 단계(911)에서, 경합자(921)는 티켓을 수신할 수 있고, 단계(912)에서, 경합자(921)는 고속 레인으로부터 요청에 대한 권한을 주장(claim)하는 것을 진행할 수 있다.

[0056] 도 10은 본 발명의 실시예에 따른, 어떤 경합도 검출되지 않을 때 희생자와 경합자 사이의 인터랙션을 도시하는 예시적인 인터랙티브 챠트이다. 도 10에 도시된 바와 같이, 희생자(1022)는 우선순위 큐에 액세스하는 동시적인 특성(concurrent nature)으로 인해, 경합자(1021)로부터 경합을 항상 검출할 수 있는 것은 아니다.

[0057] 단계들(1001 내지 1003)에서, 희생자(1022)는 동시 우선순위 큐로부터 요청을 픽업할 수 있고, 희생자(1022)는 현재의 리더 카운트, 리더들에 대해 시퀀서(1020)를 체크할 수 있다(이 단계들(1001 내지 1002)은 또한, 비-시간적 천이로서 지칭될 수 있으며, 이는 차후의 섹션에서 논의될 것이다).

[0058] 더욱이, 단계(1004)에서, 시퀀서(1020)는 경합자(1021)로부터 티켓에 대한 요청을 수신할 수 있고, 단계(1005)에서, 시퀀서(1020)는 리더 카운트, 리더들을 증가시킬 수 있다.

[0059] 단계(1006)에서, 희생자(1022)는 경합자(1021)로부터 경합을 검출하지 못할 수 있는 바, 그 이유는 경합자(1021)가 현재의 리더들을 희생자(1022)에게 리턴한 후 시퀀서(1020)가 리더 카운트를 증가시켰기 때문이다.

[0060] 단계(1007)에서, 희생자(1022)는 요청에 대한 권한을 주장하고 리더 시퀀서를 진전시킬 수 있고, 상기 리더 시

퀀서는 또한, 단계(1008)에서, 경합자(1021)에게 티켓을 발행할 있다.

[0061] 그 다음, 단계(1009)에서, 경합자(1021)는 티켓을 수신할 수 있고, 단계(1010)에서, 경합자(1021)는 우선순위 큐, 예컨대 고속 레인으로부터의 요청에 대한 권한을 주장하는 것을 진행할 수 있다.

[0062] 도 11은 본 발명의 실시예에 따른 우선순위 큐에서 협력적인 동시실행을 지원하기 위한 예시적인 순서도를 도시한다. 도 11에 도시된 바와 같이, 단계(1101)에서, 시스템은 복수의 스레드들이 동시 우선순위 큐와 인터랙션할 수 있게 한다. 그 다음, 단계(1102)에서, 시스템은 우선순위 큐 내의 하나 이상의 요청들에 대해 경합하는 복수의 스레드들을 검출하기 위해 시퀀서를 사용할 수 있다. 더욱이, 단계(1103)에서, 시스템은 우선순위 큐에서 복수의 스레드들 간의 경합을 감소시킬 수 있다.

[0063] 시간적 그리고 비-시간적 천이들

[0064] 본 발명의 실시예에 따르면, 동시 프로그래밍 모델은 시간적 천이들 및 비-시간적 천이들과 같은 서로 다른 종류의 천이들을 지원할 수 있다.

[0065] 일반적인 프로그램 순서 단계들을 포함하는 비-시간적 천이 단계들은 컴파일러 및 프로세서에 의해 자유롭게 재순서화(reorder)될 수 있다. 예를 들어, 도 9는 비-시간적 천이 단계들(913 및 914)을 포함하고, 도 10은 비-시간적 천이 단계(1011)를 포함한다. 이들 비-시간적 천이 단계들 각각 내에서, 재순서화가 프로그램 순서에 의해 의도되는 로직을 깨지않음이 보장될 수 있다.

[0066] 한편, 또한 프로그램 순서로 나타날 수 있는 시간적 천이들에 대한 재순서화에 관한 제약들이 존재한다. 시간적 천이들은 동시 알고리즘들에서의 신중한 설계 선택들을 포함할 수 있고, 시간적 천이들은 컴파일러 및 프로세서 장벽(barrier)들로서 구현될 수 있다. 예를 들어, delete_min에서 시간적 천이는 어떤 데이터 로드들, 예컨대 fastLane_w를 판독하는 것이 데이터 로드들, 예컨대 fastLane_r의 로드를 앞서갈 수 없음을 나타낼 수 있다.

[0067] 시간적 천이들 및 비-시간적 천이들의 사용은 일관된 상태 변이(mutation of the states)를 할 수 있게 하기 위해 호출자가 동시 알고리즘의 다른 부분들의 진행에 관한 정보를 수집할 수 있게 한다.

[0068] 도 9에 도시된 예에서, 희생자(922)가 단계(906)에서 경합자를 검출한 후, 리더들은 경합자들이 더 도착할 때 변화할 수 있다. 희생자(922)가 리더들의 정확한 값을 알지 못할 수 있더라도, 오직 리더 카운트가 증가할 수 있다는 사실은 여전하다. 따라서, 고속 레인 링에 요청을 배치시키는 것이 안전하다.

[0069] 한편, 도 10에 도시된 예에서, 희생자(1022)가 단계(1006)에서 어떤 경합자도 존재하지 않음(리더들 == 티켓 번호)을 결정한 후, 리더들의 참 값(true vale)은 실제로 이전에 리턴된 것보다 클 수 있는 바, 그 이유는 경합자(1021)가 도착하기 때문이다. 이러한 상황은 드문 것으로 추정될 수 있고, 리더들의 정확한 값을 결정하기 위한 비용은 이를 아는 것으로부터의 이득(gain)보다 클 수 있다. 추가적으로, 잘못된 추측을 하는 것의 비용은 낮은 것으로 고려될 수 있는 바, 그 이유는 희생자(1022)가 단계(1007)에서 리더 시퀀스를 진전시키는 것을 거의 바로 진행하고, 이는 어떤 경합자들을 매우 빨리 차단해제할 수 있기 때문이다.

[0070] 더욱이, 도 8에 도시된 예에서, 컨슈머 A(811)에 의해 고속 레인(804) 내로 요청, 예컨대 요청 A(821)을 추가하는 것은 비-시간적 천이들로 구현될 수 있고, 컨슈머 B(812)에 의해 고속 레인(804)으로부터 요청 A(821)에 대한 권한을 주장하는 것은 시간적 천이들로 구현될 수 있다.

[0071] 이 예에서, 경합자, 컨슈머 B(812)는 요청들이 실제로 고속 레인 링(804)에 저장되기 전에 fastLane_w의 값을 목격(observe)할 수 있다. 여기서, 희생자, 컨슈머 A(811)가 리더 시퀀서를 진전시킨 후에, 단 하나의 경합자만이 고속 레인(804)에 액세스하는 것이 허용된다. 그 다음, 고속 레인(804)에 액세스하는, 목격될 수 있는 요청들의 수보다 많은 경합자들은 존재하지 않을 수 있다.

[0072] 따라서, 경합자들(811)이 고속 레인(804)에 대한 요청 카운트, fastLane_w가 고속 레인(804)의 실제 채워진 부분보다 앞서 있음을 관찰할 수 있더라도, 경합자들은 오직, 각각의 경합자에 의해 한 번씩만 갱신되는 인덱스, fastLane_r를 이용하여 값들에 액세스할 수 있다. 다시 말해, 희생자, 컨슈머 A(811)는 릴리즈될 수 있는 경합자들의 최대 수를 제어함으로써 고속 레인(804)에서 요청들의 액세스성(accessibility)을 제어할 수 있는 바, 그 이유는 충분한 경합자들이 희생자(812)에 의해 릴리즈될 때에만 fastLane_r의 값이 fastLane_w에 도달할 수 있기 때문이다.

[0073] 일부 실시예들에서, 상기 언급된 방법들 중 하나를 구현하기 위한 컴퓨터 프로그램이 제공된다. 본 발명의 실시예에 따르면, 컴퓨터 프로그램은 시스템으로 하여금 단계들을 수행하게 하는 바, 상기 단계들은 복수의 스레드

들이 동시 우선순위 큐와 인터랙션할 수 있게 하는 단계와, 시퀀서를 통해, 우선순위 큐에서 하나 이상의 요청들에 대해 경합하는 복수의 스레드들을 검출하는 단계와, 우선순위 큐를 통해, 복수의 스레드들 간의 경합을 감소시키는 단계를 포함한다.

[0074] 일 실시예에서, 동시 우선순위 큐를 지원하기 위한 시스템이 제공된다. 시스템은 복수의 스레드들이 동시 우선순위 큐와 인터랙션할 수 있게 하기 위한 수단과, 시퀀서를 통해, 우선순위 큐에서 하나 이상의 요청들에 대해 경합하는 복수의 스레드들을 검출하기 위한 수단과, 우선순위 큐를 통해, 복수의 스레드들 간의 경합을 감소시키기 위한 수단을 포함한다.

[0075] 일 실시예에서, 시스템은 또한, 우선순의 큐에 캘린더 링을 포함시키기 위한 수단을 포함하고, 상기 캘린더 링은 미리 구성된 서비스 품질(QoS)보다 짧은 타겟 응답 시간을 갖는 다양한 요청들을 저장하는 동작을 한다.

[0076] 일 실시예에서, 시스템은 또한, 미리 구성된 QoS를 약 2초로 특정하기 위한 수단을 포함한다.

[0077] 일 실시예에서, 시스템은 또한, 고속 스캐닝을 위해 캘린더 링과 동기화하여 갱신되는 비트 맵을 이용하기 위한 수단을 포함한다.

[0078] 일 실시예에서, 시스템은 또한, 우선순위 큐와 가로채진 요청들의 리스트를 관련시키기 위한 수단을 포함하며, 상기 가로채진 요청들의 리스트는 워크로드가 집약적일 때 캘린더 링 상의 경합을 감소시키기 위해 사용될 수 있고, 상기 가로채진 요청들의 리스트는 동일한 캘린더 입력에 랜딩하는 하나 이상의 요청들을 포함한다.

[0079] 일 실시예에서, 시스템은 또한, 우선순위 큐에 협동적인 동시실행을 지원하기 위해 사용될 수 있는 고속 레인을 포함하기 위한 수단을 포함한다.

[0080] 일 실시예에서, 시스템은 또한, 시퀀서를 통해 복수의 스레드들을 순서화할 수 있는 티켓 메커니즘을 이용하기 위한 수단을 포함한다.

[0081] 일 실시예에서, 시스템은 또한, 티켓 매커니즘을 통해, 컨슈머가 우선순위 큐에 액세스하는 것을 시도할 때마다 증가되는 리더 카운트를 유지하기 위한 수단을 포함한다.

[0082] 일 실시예에서, 시스템은 또한, 시퀀서를 통해, 경합 스레드에 티켓을 발행하기 위한 수단을 포함하며, 티켓은 상기 경합 스레드가 우선순위 큐에 액세스할 수 있게 한다.

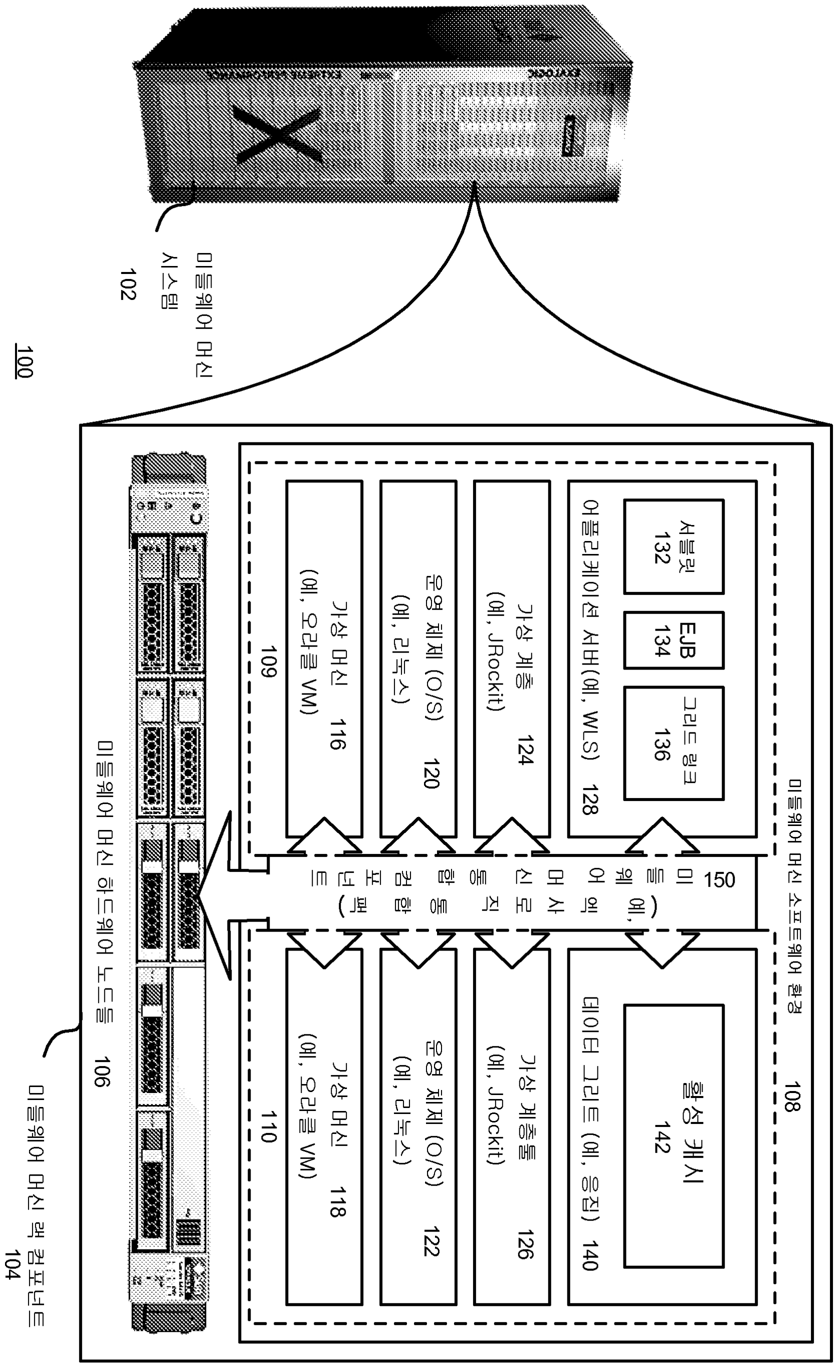
[0083] 본 발명은 본 발명의 교시들에 따라 프로그램된 하나 이상의 프로세서들, 메모리 및/또는 컴퓨터 판독가능 스토리지 매체를 포함하는 하나 이상의 종래의 범용 또는 특수 디지털 컴퓨터, 컴퓨팅 디바이스, 머신 또는 마이크로프로세서를 이용하여 통상적으로 구현될 수 있다. 적절한 소프트웨어 코딩은 소프트웨어 기술 분야의 숙련자들에게 분명할 바와 같이, 본 발명의 교시들에 근거하여 숙련된 프로그래머들에 의해 쉽게 준비될 수 있다.

[0084] 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.

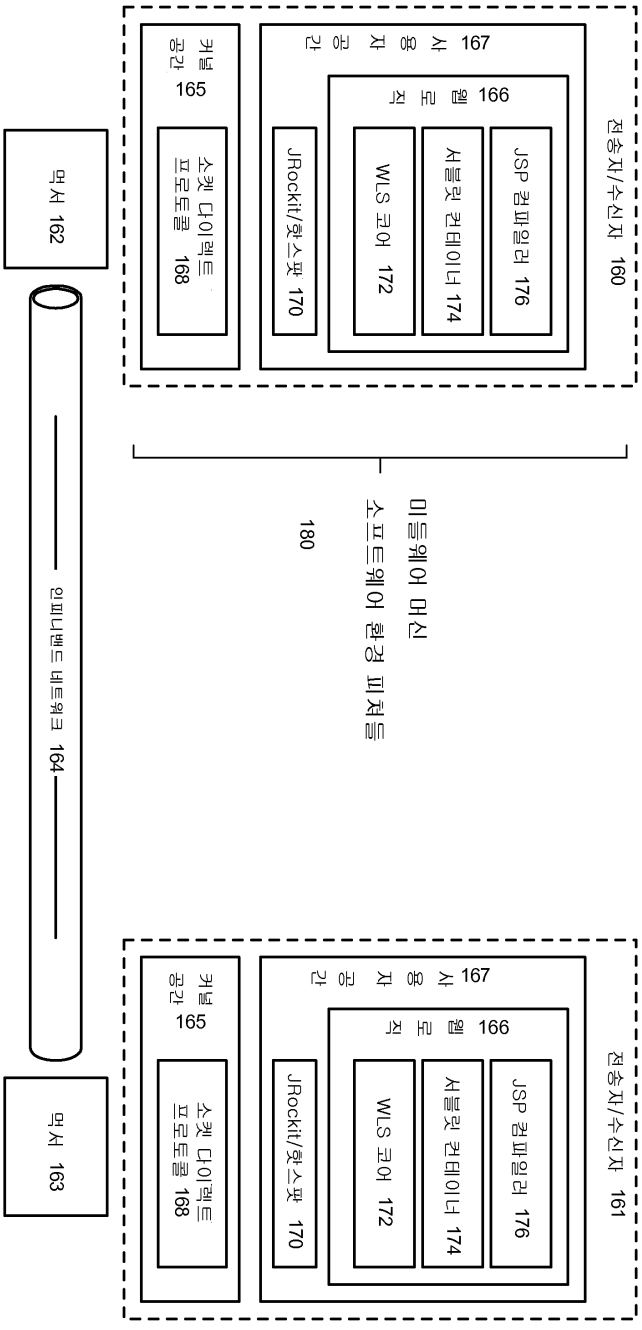
[0085] 본 발명의 상기 상세한 설명은 예시 및 설명을 위해 제공되었다. 본 설명은 완전한 것이거나 또는 정확히 개시된 형태들로만 본 발명을 제한하고자 의도된 것이 아니다. 많은 수정들 및 변형들이 이 기술분야의 숙련자에게 분명할 것이다. 위 실시예들은 본 발명의 원리 및 이의 실용적 응용을 가장 잘 설명하기 위해 선택 및 기술되었으며, 그럼으로써 이 기술분야의 숙련자들은 본 발명에 대한 다양한 실시예들 및 고려되는 특별한 사용에 적합한 다양한 수정들을 이해할 수 있다. 본 발명의 범위는 다음의 특허 청구 범위 및 이의 균등물에 의해 한정되어야 함이 의도된다.

도면

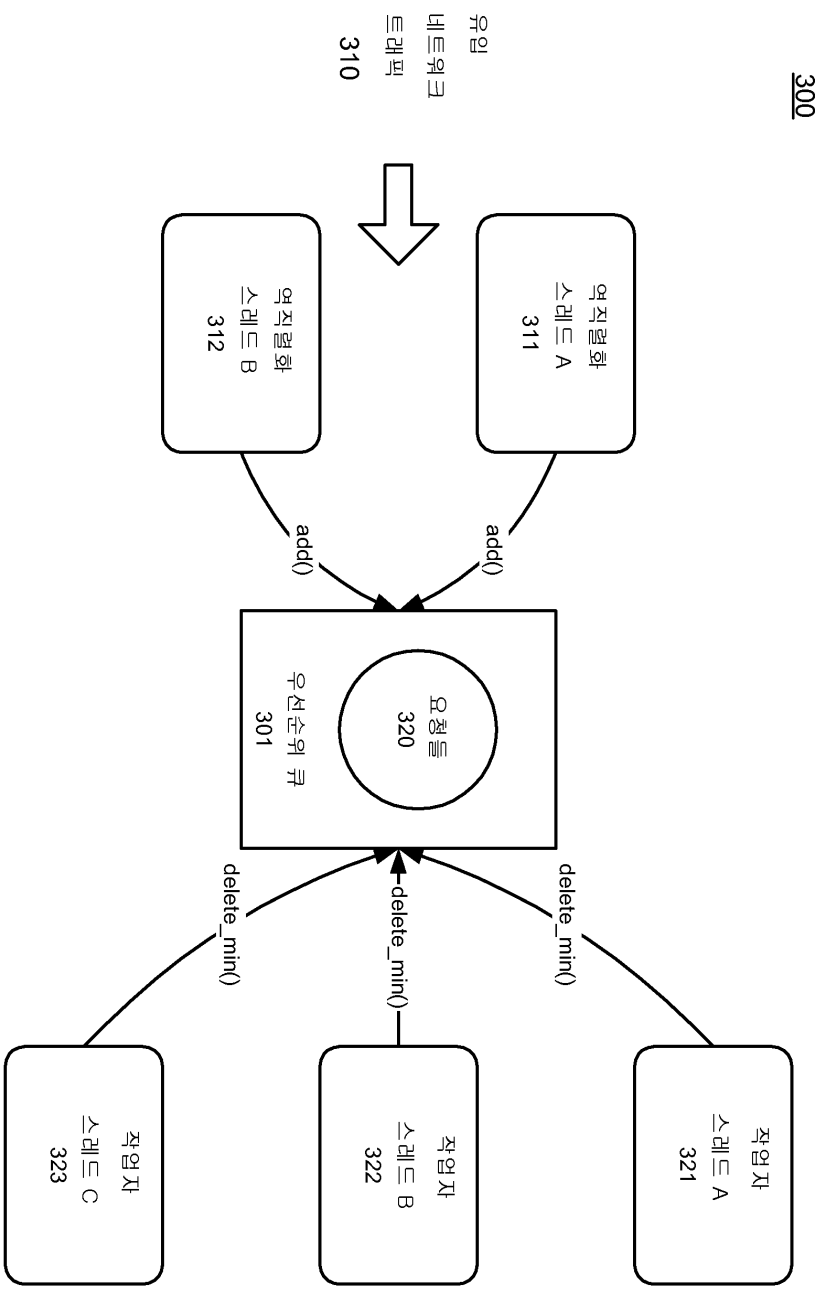
도면1


100
미들웨어 머신 시스템
102
미들웨어 머신 랙 컴포넌트
104
미들웨어 머신 하드웨어 노드들
106
미들웨어 머신 소프트웨어 환경
108
109
110
서블릿
132
EJB
134
그리드 링크
136
어플리케이션 서버(예, WLS) 128
가상 계층 124
(예, JRockit)
운영 체제 (O/S) 120
(예, 리눅스)
가상 머신 116
(예, 오라클 VM)
150
미들웨어 머신 통합 컴포넌트
(예, 엑사로직 통합 팩)
활성 캐시
142
데이터 그리드 (예, 응집) 140
가상 계층들 126
(예, JRockit)
운영 체제 (O/S) 122
(예, 리눅스)
가상 머신 118
(예, 오라클 VM)

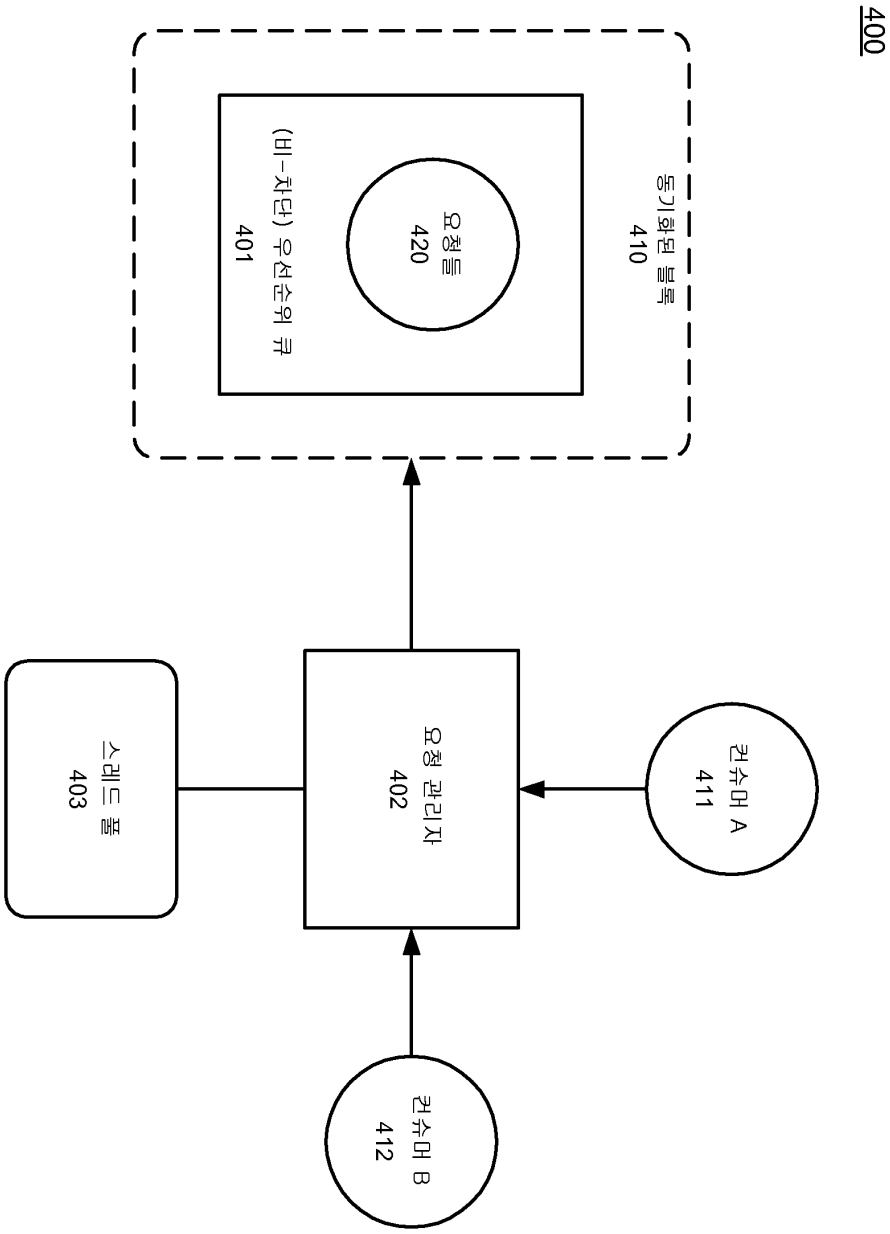
도면2


전송자/수신자 160
사용자 공간 167
웹 로직 166
JSP 컴파일러 176
서블릿 컨테이너 174
WLS 코어 172
JRockit/핫스팟 170
커널 공간 165
소켓 다이렉트 프로토콜 168
미들웨어 머신 소프트웨어 환경 피쳐들 180
인피니밴드 네트워크 164
먹서 162
먹서 163
전송자/수신자 161
사용자 공간 167
웹 로직 166
JSP 컴파일러 176
서블릿 컨테이너 174
WLS 코어 172
JRockit/핫스팟 170
커널 공간 165
소켓 다이렉트 프로토콜 168

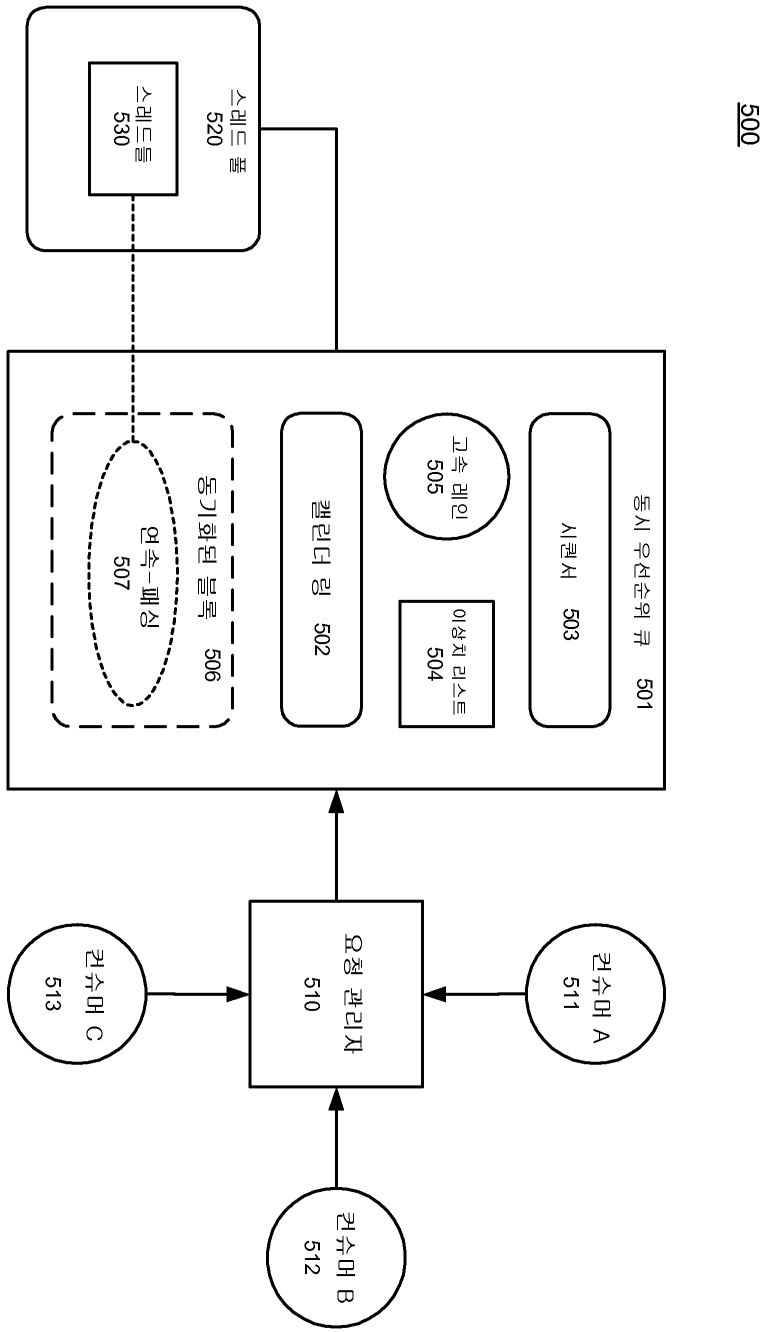
도면3


300
유입
네트워크
트래픽
310
역직렬화
스레드 A
311
역직렬화
스레드 B
312
add()
add()
요청들
320
우선순위 큐
301
delete_min()
delete_min()
delete_min()
작업자
스레드 A
321
작업자
스레드 B
322
작업자
스레드 C
323

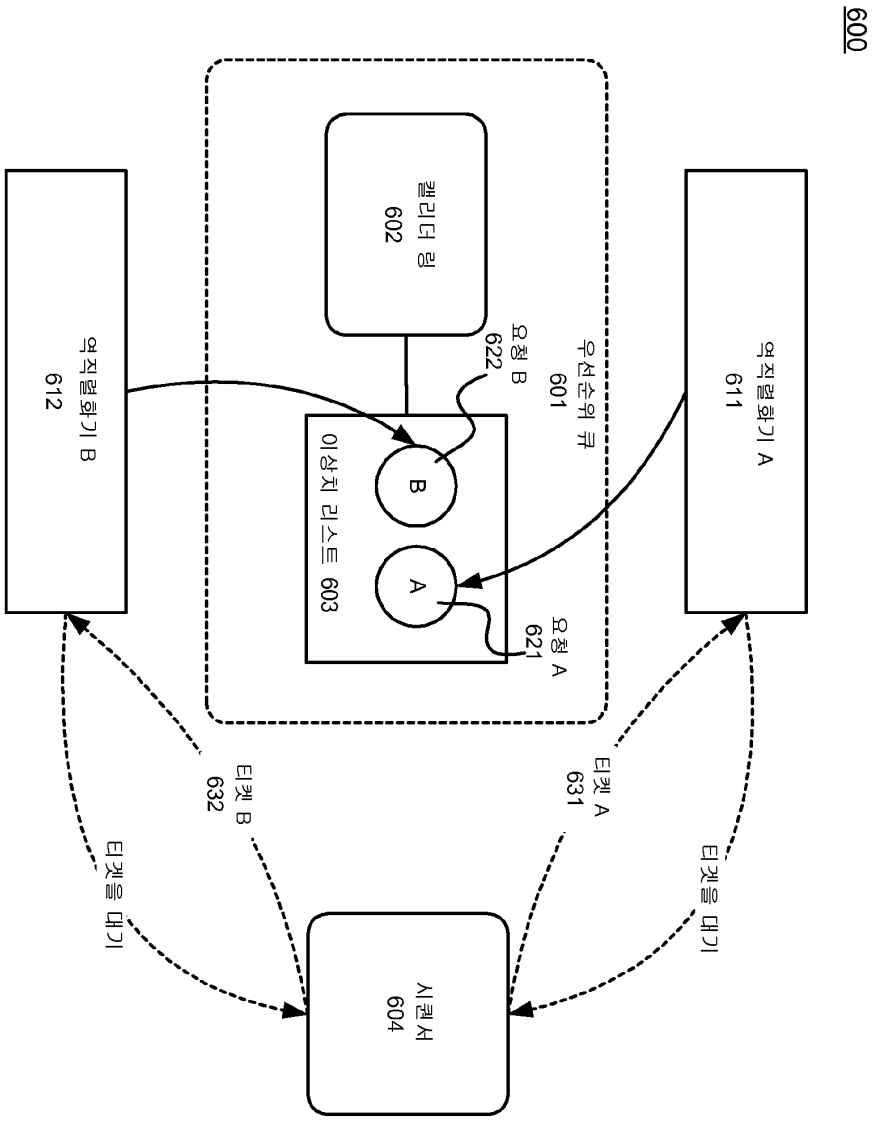
도면4


400
동기화된 블록
410
(비-차단) 우선순위 큐
401
요청들
420
요청 관리자
402
스레드 풀
403
컨슈머 A
411
컨슈머 B
412

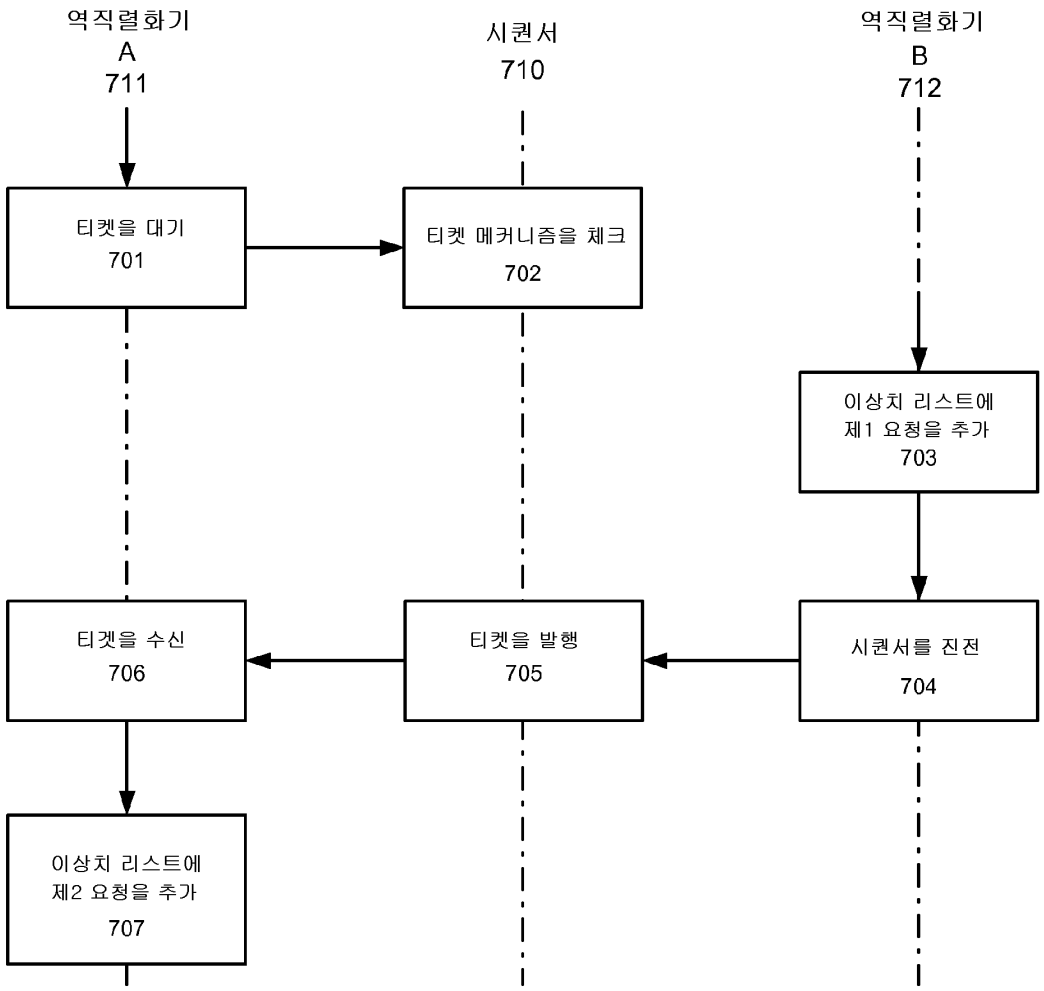
도면5


500
스레드 풀
520
스레드들
530
동시 우선순위 큐 501
시퀀서 503
고속 레인
505
이상치 리스트
504
캘린더 링 502
동기화된 블록 506
연속-패싱
507
요청 관리자
510
컨슈머 A
511
컨슈머 B
512
컨슈머 C
513

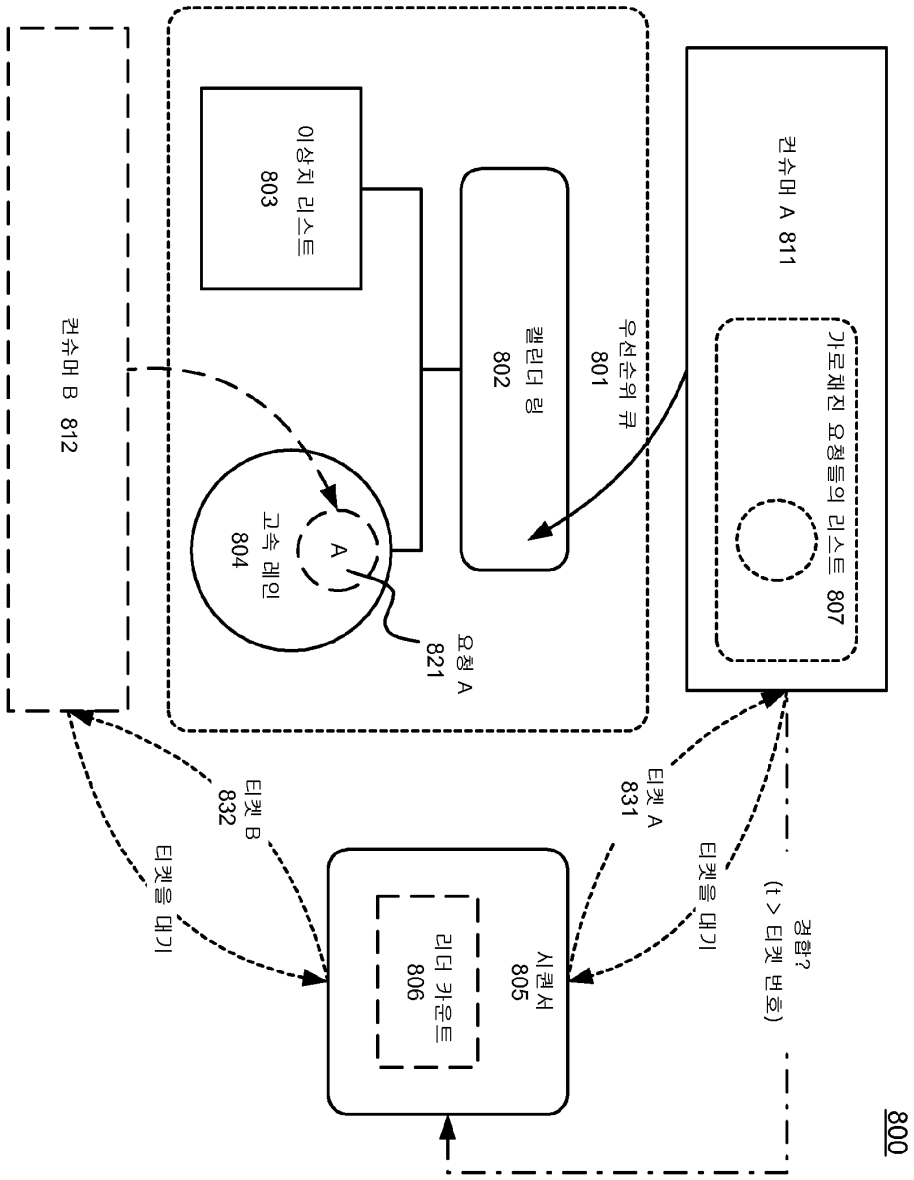
도면6


600
역직렬화기 A
611
역직렬화기 B
612
캘리더 링
602
우선순위 큐
601
요청 B
622
B
A
이상치 리스트 603
요청 A
621
티켓 A
631
티켓 B
632
티켓을 대기
티켓을 대기
시퀀서
604

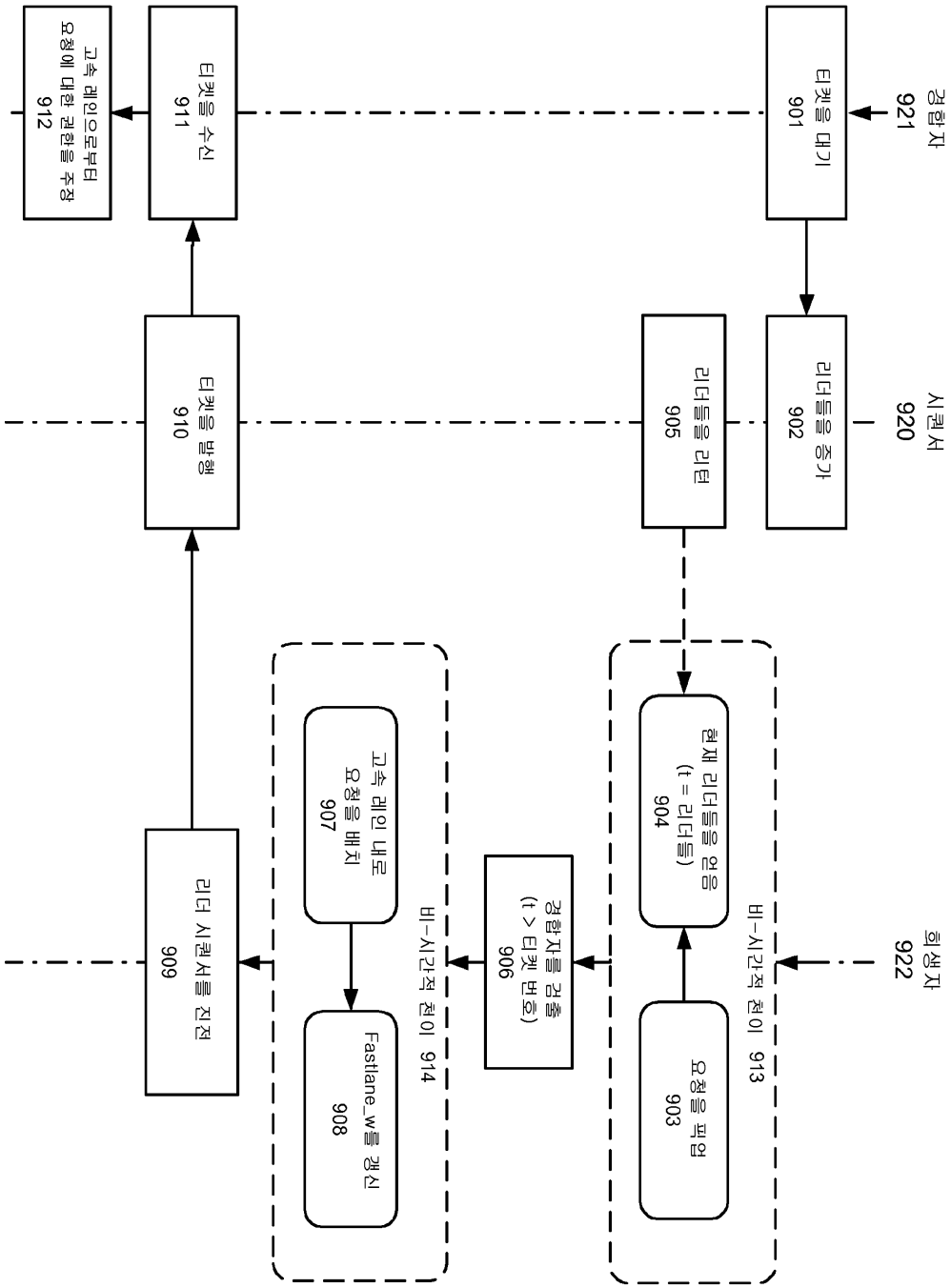
도면7


역직렬화기
A
711
시퀀서
710
역직렬화기
B
712
티켓을 대기
701
티켓 메커니즘을 체크
702
이상치 리스트에
제1 요청을 추가
703
티켓을 수신
706
티켓을 발행
705
시퀀서를 진전
704
이상치 리스트에
제2 요청을 추가
707

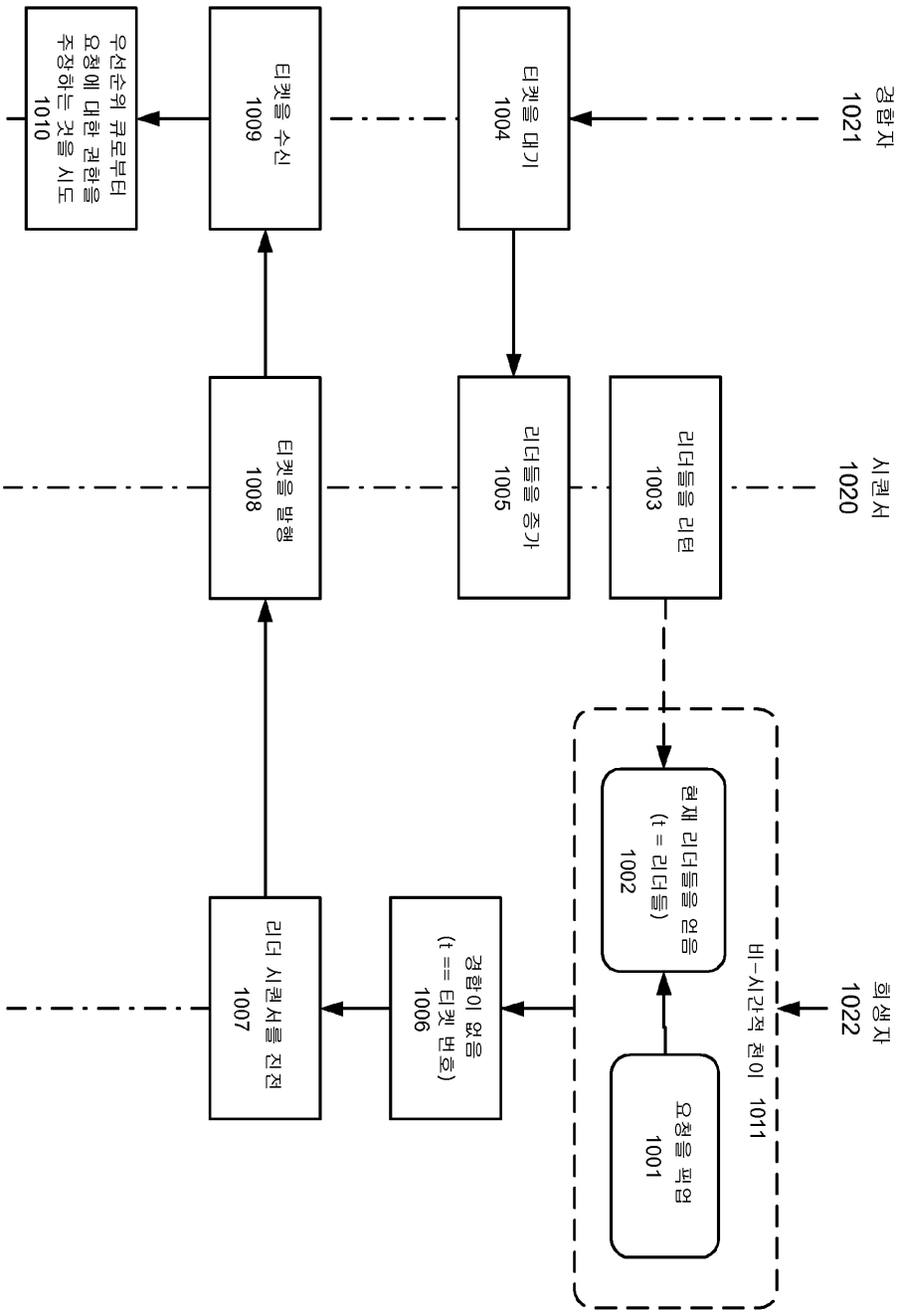
도면8


컨슈머 A 811
가로채진 요청들의 리스트 807
우선순위 큐 801
캘린더 링 802
이상치 리스트 803
고속 레인 804
A
요청 A 821
컨슈머 B 812
티켓 A 831
티켓 B 832
티켓을 대기
티켓을 대기
시퀀서 805
리더 카운트 806
경합? (t > 티켓 번호)
800

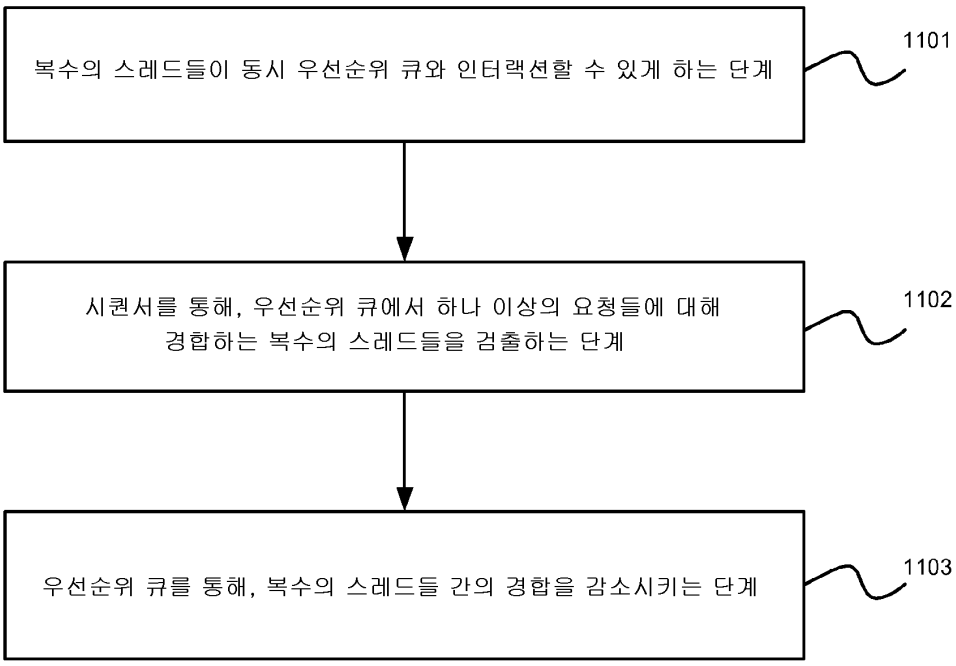
도면9


경합자
921
시퀀서
920
희생자
922
티켓을 대기
901
리더들을 증가
902
현재 리더들을 얻음
(t = 리더들)
904
요청을 픽업
903
비-시간적 천이 913
경합자를 검출
(t > 티켓 번호)
906
고속 레인 내로
요청을 배치
907
Fastlane_w를 갱신
908
비-시간적 천이 914
리더 시퀀서를 진전
909
티켓을 발행
910
리더들을 리턴
905
티켓을 수신
911
고속 레인으로부터
요청에 대한 권한을 주장
912

도면10


경합자
1021
시퀀서
1020
희생자
1022
비-시간적 천이 1011
요청을 픽업
1001
현재 리더들을 얻음
(t = 리더들)
1002
리더들을 리턴
1003
리더들을 증가
1005
티켓을 대기
1004
경합이 없음
(t == 티켓 번호)
1006
티켓을 발행
1008
티켓을 수신
1009
리더 시퀀서를 진전
1007
우선순위 큐로부터
요청에 대한 권한을
주장하는 것을 시도
1010

도면11



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 제11항 10번째 줄

【변경전】

시퀀서를 사용하여 상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출(first-in-first-out, FIFO) 순서를 시행하는 단계

【변경후】

우선순위 큐에 포함되는 시퀀서를 사용하여 상기 이상치 리스트에 저장된 상기 복수의 요청들의 상기 제2 서브세트에 대해 선입선출(first-in-first-out, FIFO) 순서를 시행하는 단계