(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년12월17일
(11) 등록번호 10-2340296
(24) 등록일자 2021년12월13일

(51) 국제특허분류(Int. Cl.)
G06F 9/46 (2006.01) *H04L 29/08* (2006.01)
(52) CPC특허분류
G06F 9/467 (2013.01)
H04L 67/10 (2013.01)
(21) 출원번호 10-2016-7032798
(22) 출원일자(국제) 2015년03월27일
심사청구일자 2020년01월20일
(85) 번역문제출일자 2016년11월23일
(65) 공개번호 10-2016-0147909
(43) 공개일자 2016년12월23일
(86) 국제출원번호 PCT/US2015/023120
(87) 국제공개번호 WO 2015/167724
국제공개일자 2015년11월05일
(30) 우선권주장
61/985,135 2014년04월28일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
US7284018 B1*
(뒷면에 계속)
전체 청구항 수 : 총 20 항

(73) 특허권자
오라클 인터내셔날 코포레이션
미국, 캘리포니아 94065, 레드우드 쇼어스 엠에스 5오피7, 오라클 파크웨이 500
(72) 발명자
쉔 쉬강
중국 베이징 100193 하이디안 디스트릭트 중관춘 소프트웨어 파크 빌딩 24번 오라클 빌딩
장 칭셍
중국 베이징 100193 하이디안 디스트릭트 중관춘 소프트웨어 파크 빌딩 24번 오라클 빌딩
(뒷면에 계속)
(74) 대리인
박장원

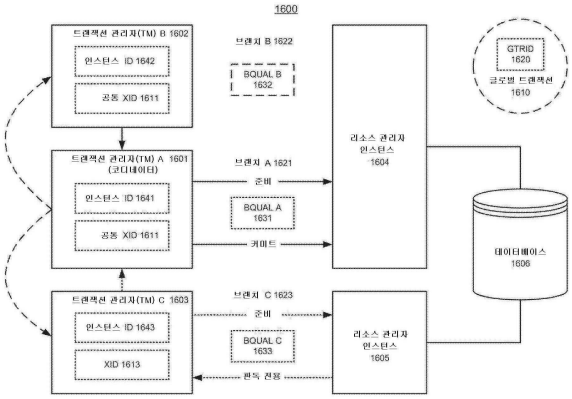
심사관 : 유진태

(54) 발명의 명칭 **트랜잭셔널 환경에서 리소스 관리자(RM) 인스턴스 인지에 기초하여 공통 트랜잭션 식별자(XID) 최적화 및 트랜잭션 친화성을 지원하기 위한 시스템 및 방법**

(57) 요 약

시스템 및 방법이 트랜잭션 환경에서 트랜잭션 프로세싱을 지원할 수 있다. 글로벌 트랜잭션에 대한 코디네이터가 상기 트랜잭셔널 환경에서 상기 글로벌 트랜잭션의 하나 이상의 참여자들에게 공통 트랜잭션 식별자(common transaction identifier) 및 리소스 관리자 인스턴스에 대한 정보를 전파하는 동작을 한다. 상기 코디네이터는 상기 코디네이터와 상기 리소스 관리자 인스턴스를 공유하는 상기 하나 이상의 참여자들이 상기 공통 트랜잭션 식별자를 이용하게 하며, 하나의 트랜잭션 브랜치를 이용하여 상기 리소스 관리자 인스턴스를 공유하는 상기 하나 이상의 참여자들에 대한 상기 글로벌 트랜잭션을 프로세싱할 수 있다.

대 표 도



(72) 발명자

리틀 토드 제이.

미국 일리노이즈 60067-6675 팔라틴 웨스트 일리노이즈 애비뉴 1155

진 용슌

중국 베이징 100193 하이디안 디스트릭트 중관춘 소프트웨어 파크 빌딩 24번 오라클 빌딩

(56) 선행기술조사문헌

US20030036919 A1*

US20130246368 A1

JP2004295272 A

JP2007011550 A

US07284018 B1*

*는 심사관에 의하여 인용된 문헌

(30) 우선권주장

14/587,468 2014년12월31일 미국(US)

14/587,474 2014년12월31일 미국(US)

14/587,476 2014년12월31일 미국(US)

명 세 서

청구범위

청구항 1

제1 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제1 그룹과 제2 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제2 그룹을 포함하는 복수의 트랜잭셔널 서버들과, 데이터베이스에 대한 액세스를 관리하기 위한 복수의 리소스 관리자 인스턴스들을 포함하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법으로서,

상기 제1 트랜잭션 관리자 및 상기 제2 트랜잭션 관리자를 통해 상기 복수의 리소스 관리자 인스턴스들의 공유 리소스 관리자 인스턴스로 향하는(directed) 글로벌 트랜잭션을 상기 복수의 트랜잭셔널 서버들로부터 수신하는 단계;

상기 글로벌 트랜잭션에 대한 코디네이터(coordinator)로서 트랜잭셔널 서버들의 제1 그룹의 제1 트랜잭션 관리자를 사용하는 단계;

상기 글로벌 트랜잭션에 대한 코디네이터를 통해, 상기 공유 리소스 관리자 인스턴스에 대한 공통 트랜잭션 식별자 및 정보를 트랜잭셔널 환경에서 상기 글로벌 트랜잭션에 참여하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자 모두에게 전파하는 단계;

상기 공유 리소스 관리자 인스턴스를 공유하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자가 공통 트랜잭션 식별자를 이용하게 하는 단계; 그리고

하나의 트랜잭션 브랜치를 사용하는 공유 리소스 관리자 인스턴스 및 상기 코디네이터를 통해, 상기 글로벌 트랜잭션을 프로세싱하는 단계를 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 2

제1항에 있어서, 상기 데이터베이스는 클러스터화된 데이터베이스인 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 3

제1항에 있어서, 상기 방법은,

트랜잭셔널 서버들의 제1 그룹에서 제1 트랜잭션 관리자에 의해 상기 글로벌 트랜잭션에 할당된 트랜잭션 식별자를 상기 공통 트랜잭션 식별자로서 이용하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 4

제3항에 있어서, 상기 데이터베이스는 클러스터화된 데이터베이스인 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 5

제1항에 있어서, 상기 방법은,

상기 코디네이터를 통해, 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자에 대한 상기 글로벌 트랜잭션을 커미트(commit)하거나 롤백(roll back)하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 6

제1항에 있어서, 상기 방법은,

상기 공유 리소스 관리자 인스턴스를 공유하지 않는 다른 참여자에 대한 상기 글로벌 트랜잭션을 프로세싱하기 위해 다른 트랜잭션 브랜치를 이용하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 7

제1항에 있어서, 상기 방법은,

상기 글로벌 트랜잭션에 관여되는 모든 참여자들이 상기 리소스 관리자 인스턴스를 공유할 때 상기 공유 리소스 관리자 인스턴스 상에서 직접적으로 1PC(one-phase commit) 동작을 인보크(invoke)하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 8

제1항에 있어서, 상기 방법은,

원격 트랜잭셔널 리소스들을 릴리즈(release)시키기 위해 원격 노드 상의 참여자에게 포겟 요청(forget request)을 전송하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 9

제1항에 있어서, 상기 방법은,

서로 다른 도메인들의 복수의 참여자들이 상기 글로벌 트랜잭션에 관여되도록 하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 10

제1항에 있어서, 상기 방법은,

원격 도메인으로의 게이트웨이를 통해, 상기 공통 트랜잭션 식별자 및 상기 공유 리소스 관리자 인스턴스에 대한 식별 정보를 수신하는 단계를 더 포함하는 것을 특징으로 하는 트랜잭셔널 시스템에서 트랜잭션 프로세싱을 지원하기 위한 방법.

청구항 11

삭제

청구항 12

트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템으로서,

각각 하나 이상의 마이크로 프로세서 및 메모리를 포함하는 복수의 네트워크 컴퓨터 시스템들;

상기 복수의 네트워크 컴퓨터 시스템들에서 동작하는 복수의 트랜잭셔널 서버들 -상기 복수의 트랜잭셔널 서버들은 제1 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제1 그룹, 및 제2 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제2 그룹을 포함함-;

데이터베이스;

데이터베이스에 대한 액세스를 관리하기 위한 복수의 리소스 관리자 인스턴스들을 포함하며,

상기 복수의 트랜잭셔널 서버들은 상기 제1 트랜잭션 관리자 및 상기 제2 트랜잭션 관리자를 통해 제1 복수의 리소스 관리자 인스턴스들 및 제2 복수의 리소스 관리자 인스턴스들의 공유 리소스 관리자 인스턴스로 향하는 글로벌 트랜잭션을 생성하도록 구성되고;

제1 트랜잭션 관리자는 글로벌 트랜잭션에 대한 코디네이터(coordinator)로서 기능하고;

상기 글로벌 트랜잭션에 대한 코디네이터는 상기 공유 리소스 관리자 인스턴스에 대한 공통 트랜잭션 식별자 및 정보를 트랜잭셔널 환경에서 상기 글로벌 트랜잭션에 참여하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자 모두에게 전파하도록 구성되고;

공유 리소스 관리자 인스턴스를 공유하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자는 글로벌 트랜잭션을 코디네이트(coordinate)하기 위해 공통 트랜잭션 식별자를 사용하도록 구성되고; 그리고

상기 코디네이터는 하나의 트랜잭션 브랜치를 사용하여, 코디네이터 및 공유 리소스 관리자 인스턴스를 통해 데이터베이스에 대한 글로벌 트랜잭션을 프로세싱하도록 구성되는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 13

제12항에 있어서, 상기 데이터베이스는 클러스터화된 데이터베이스인 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 14

제13항에 있어서,

트랜잭셔널 서버들의 제1 그룹에서 제1 트랜잭션 관리자에 의해 상기 글로벌 트랜잭션에 할당된 트랜잭션 식별자는 상기 공통 트랜잭션 식별자로서 이용되는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 15

제14항에 있어서,

상기 데이터베이스는 클러스터화된 데이터베이스를 포함하는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 16

제12항에 있어서,

상기 코디네이터는 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자에 대한 상기 글로벌 트랜잭션을 커미트하거나 롤백하는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 17

제12항에 있어서,

상기 공유 리소스 관리자 인스턴스를 공유하지 않는 다른 참여자에 대한 상기 글로벌 트랜잭션을 프로세싱하기 위해 다른 트랜잭션 브랜치가 이용되는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 18

제12항에 있어서,

상기 글로벌 트랜잭션에 참여하는 모든 트랜잭션 관리자들이 공유 리소스 관리자 인스턴스를 공유할 때 공유 리소스 관리자 인스턴스 상에서 직접적으로 1PC(one-phase commit) 동작이 인보크되는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 19

제12항에 있어서,

상기 코디네이터는 원격 트랜잭셔널 리소스들을 릴리즈시키기 위해 원격 노드 상의 트랜잭션 관리자에게 포겟 요청을 전송하는 동작을 하는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 20

제12항에 있어서,

제1 트랜잭션 관리자 및 제2 트랜잭션 관리자는 서로 다른 도메인들에 있는 것을 특징으로 하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템.

청구항 21

제1 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제1 그룹과 제2 트랜잭션 관리자를 포함하는 트랜잭셔널 서버들의 제2 그룹을 포함하는 복수의 트랜잭셔널 서버들과, 데이터베이스에 대한 액세스를 관리하기 위한 복수의 리소스 관리자 인스턴스들을 포함하는 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 명령어들이 저장된 비일시적 머신 판독가능 저장 매체로서, 상기 명령어들은 실행될 때 시스템으로 하여금 단계들을 수행하게 하며, 상기 단계들은,

상기 제1 트랜잭션 관리자 및 상기 제2 트랜잭션 관리자를 통해 상기 복수의 리소스 관리자 인스턴스들의 공유 리소스 관리자 인스턴스로 향하는(directed) 글로벌 트랜잭션을 상기 복수의 트랜잭셔널 서버들로부터 수신하는 단계;

상기 글로벌 트랜잭션에 대한 코디네이터(coordinator)로서 트랜잭셔널 서버들의 제1 그룹의 제1 트랜잭션 관리자를 사용하는 단계;

상기 글로벌 트랜잭션에 대한 코디네이터를 통해, 상기 공유 리소스 관리자 인스턴스에 대한 공통 트랜잭션 식별자 및 정보를 트랜잭셔널 환경에서 상기 글로벌 트랜잭션에 참여하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자 모두에게 전파하는 단계;

상기 공유 리소스 관리자 인스턴스를 공유하는 상기 제1 트랜잭션 관리자 및 제2 트랜잭션 관리자가 공통 트랜잭션 식별자를 이용하게 하는 단계; 그리고

하나의 트랜잭션 브랜치를 사용하는 공유 리소스 관리자 인스턴스 및 상기 코디네이터를 통해, 상기 글로벌 트랜잭션을 프로세싱하는 단계를 포함하는 것을 특징으로 하는 비일시적 머신 판독가능 저장 매체.

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] 기술 분야

[0004] 본 발명은 일반적으로, 컴퓨터 시스템들 및 소프트웨어에 관한 것이며, 특히 트랜잭셔널 시스템(transactional system)에 관한 것이다.

배 경 기 술

[0005] 트랜잭셔널 미들웨어 시스템 또는 트랜잭션 지향 미들웨어(transaction oriented middleware)는 조직 내에서 다양한 트랜잭션들을 프로세싱할 수 있는 엔터프라이즈 어플리케이션 서버들을 포함한다. 고성능 네트워크 및 멀티프로세서 컴퓨터들과 같은 신기술들의 발전에 따라, 트랜잭셔널 미들웨어의 성능을 더 향상시킬 필요가 존재한다. 이들이 본 발명의 실시예들이 해결하고자 하는 일반적인 영역들이다.

발명의 내용

[0006] 트랜잭셔널 환경의 트랜잭션 프로세싱을 지원할 수 있는 시스템들 및 방법들이 본 명세서에 기술된다. 글로벌 트랜잭션에 대한 코디네이터(coordinator)가 트랜잭셔널 환경에서 글로벌 트랜잭션의 하나 이상의 참여자들에게 공통 트랜잭션 식별자 및 리소스 관리자 인스턴스에 대한 정보를 전파하는 동작을 한다. 상기 코디네이터는 리소스 관리자 인스턴스를 상기 코디네이터와 공유하는 상기 하나 이상의 참여자들이 상기 공통 트랜잭션 식별자를 이용할 수 있게 하며, 상기 코디네이터는 하나의 트랜잭션 브랜치를 이용하여 리소스 관리자 인스턴스를 공유하는 상기 하나 이상의 참여자들에 대한 글로벌 트랜잭션을 프로세싱할 수 있다.

[0007] 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원할 수 있는 시스템들 및 방법들이 본 명세서에 기술된다. 트랜잭셔널 시스템은 트랜잭셔널 서버에 요청을 라우팅하는 동작을 하며, 상기 트랜잭셔널 서버는 리소스 관리자(RM) 인스턴스에 연결된다. 더욱이, 트랜잭셔널 시스템은 트랜잭셔널 서버에 친화성 컨텍스트(affinity context)를 할당할 수 있는 바, 상기 친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타내고, 상기 트랜잭셔널 시스템은 친화성 컨텍스트에 기초하여 트랜잭셔널 서버에 상기 요청과 관계된 하나 이상의 후속적인 요청들을 라우팅할 수 있다.

[0008] 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원할 수 있는 시스템들 및 방법들이 본 명세서에 기술된다. 트랜잭셔널 서버는 하나 이상의 RM 인스턴스들과 관련된 리소스 관리자(RM) 인스턴스 정보를 데이터 소스로부터 수신하는 동작을 하는 바, 상기 수신된 인스턴스 정보는 트랜잭셔널 서버로 하여금 트랜잭셔널 서버와 현재 연결된 RM 인스턴스를 인지할 수 있게 한다. 더욱이, 트랜잭셔널 서버는 트랜잭셔널 서버와 관련된 하나 이상의 테이블들에 수신된 인스턴스 정보를 저장하는 동작을 한다. 그 다음, 트랜잭셔널 서버는 하나 이상의 테이블들에 저장된 인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱할 수 있다.

도면의 간단한 설명

[0009] 도 1은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 리소스 관리자(RM) 인스턴스 인지를 지원하는 예를 도시한다.

도 2는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 다양한 상태 테이블들을 유지하는 예를 도시한다.

도 3은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 서버 테이블(ST)을 지원하는 예를 도시한다.

도 4는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 정보를 갱신하는 예를 도시한다.

도 5는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 다양한 체크 포인트들을 갖는 트랜잭션 프로세스를 지원하는 예를 도시한다.

도 6은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 리소스 관리자(RM) 인스턴스 인지를 지원하기 위한 예시적인 순서도를 도시한다.

도 7은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 인지에 기초하여 트랜잭션 친화성을 지원하는 예를 도시한다.

도 8은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 트랜잭션 친화성 라우팅을 지원하는 예를 도시한다.

도 9는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 친화성 컨텍스트를 갖는 메시지를 전송하는 예를 도시한다.

도 10은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 친화성 컨텍스트를 갖는 메시지를 수신하는 예를 도시한다.

도 11은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 클라이언트 컨텍스트 내에서 친화성 라우팅을 지원하는 예를 도시한다.

도 12는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 서로 다른 도메인들에 걸쳐 친화성 컨텍스트를 전파하는 예를 도시한다.

도 13은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 어플리케이션 서버에 친화성 컨텍스트를 전파하는 예를 도시한다.

도 14는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 인지에 기초하여 트랜잭션 친화성을 지원하기 위한 예시적인 순서도를 도시한다.

도 15는 본 발명의 실시예에 따른 서로 다른 트랜잭션 식별자들(XID들)을 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하는 예를 도시한다.

도 16은 본 발명의 실시예에 따른 공통 트랜잭션 식별자(XID)를 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하는 예를 도시한다.

도 17은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 데이터베이스 인스턴스 인지에 기초하여 1PC(one-phase

commit) 프로세싱 모델을 지원하는 예를 도시한다.

도 18운 본 발명의 실시예에 따른 트랜잭셔널 환경에서 데이터베이스 인스턴스 인지에 기초하여 글로벌 트랜잭션을 프로세싱하는 예를 도시한다.

도 19는 본 발명의 실시예에 따른 공통 XID를 이용하여 트랜잭셔널 환경에서 복수의 도메인들에 걸쳐 글로벌 트랜잭션을 프로세싱하는 예를 도시한다.

도 20은 본 발명의 실시예에 따른 공통 XID를 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하기 위한 예시적인 흐름도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0010] 본 발명은 첨부 도면들에서 제한이 아닌 예로서 예시되며, 이 도면들에서 유사한 도면 부호들은 유사한 요소들을 나타낸다. 주목할 점으로서, 본 명세서에서 "일" 또는 "하나" 또는 일부" 실시예(들)에 대한 참조는 반드시 동일한 실시예들에 대한 참조가 아니며, 이러한 참조들은 적어도 하나를 의미하는 것이다.

[0011] 다음의 본 발명의 설명은 트랜잭셔널 미들웨어 머신 환경에 대한 예로서 턱시도(Tuxedo)환경을 이용한다. 트랜잭셔널 미들웨어 머신 환경들의 다른 타입들이 제한이 없이 이용될 수 있음이 이 기술 분야의 숙련자들에게 분명할 것이다.

[0012] 트랜잭셔널 미들웨어 머신 환경과 같은 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원할 수 있는 시스템들 및 방법들이 본 명세서에서 기술된다.

[0013] 트랜잭셔널 미들웨어 머신 환경

[0014] 본 발명의 실시예에 따르면, 시스템은 빠르게 프로비져닝되고(provisioned) 요구에 따라(on demand) 스케일링할 수 있는 대량의 병렬 인-메모리 그리드(massively parallel in-memory grid)를 포함하는 완전한 자바 EE 어플리케이션 서버 컴플렉스를 제공하기 위하여, 웹로직과 같은 어플리케이션 서버 또는 미들웨어 환경과 함께 고성능 하드웨어, 예컨대 64-비트 프로세서, 고성능 대용량 메모리 및 리던던트 인피니밴드 및 이더넷 네트워킹의 조합을 포함한다. 실시예에 따르면, 시스템은 어플리케이션 서버 그리드, 저장 영역 네트워크 및 인피니백드(IB) 네트워크를 제공하는 제 풀(full), 하프(half) 또는 쿼터 랙(quarter rack) 또는 다른 구성으로서 디플로이(deploy)될 수 있다. 미들웨어 머신 소프트웨어는 예컨대, 웹로직 서버, JRockit 또는 핫스팟 JVM, 오라클 리눅스 또는 솔라리스 및 오라클 VM과 같은 어플리케이션 서버, 미들웨어 및 다른 기능을 제공할 수 있다. 실시예에 따르면, 시스템은 IB 네트워크를 통해 서로 통신하는 복수의 컴퓨트 노드들, IB 스위치 게이트웨이 및 저장 노드들 또는 유닛들을 포함할 수 있다. 랙 구성으로서 구현될 때, 랙의 사용되지 않은 부분들은 비어진 채로 있거나 또는 필러들에 의해 점유될 수 있다.

[0015] 본 발명의 실시예에 따르면, 시스템은 오라클 미들웨어 SW 수트 또는 웹로직과 같은 미들웨어 또는 어플리케이션 서버 소프트웨어를 호스팅하기 위한 디플로이가 용이한(easy-to-deploy) 솔루션이다. 여기에서 서술된 것처럼, 일 실시예에 따라, 상기 시스템은, 하나 이상의 서버들, 저장 유닛들, 저장 네트워킹을 위한 IB 패브릭, 및 미들웨어 애플리케이션을 호스팅하기 위해 요구되는 모든 다른 컴포넌트들을 포함하는 "박스 내 그리드"이다. 중요한 성능은, 예를 들어, 리얼 애플리케이션 클러스터들 및 엑사로직 오픈 저장소를 사용하여 대량의 병렬 그리드 아키텍쳐를 활용함으로써 모든 타입들의 미들웨어 애플리케이션들에 대해 전달될 수 있다. 상기 시스템은 선형 I/O 확장성을 갖는 개선된 성능을 전달하고, 사용하고 관리하기 단순하며, 미션에 중대한 이용가능성 및 신뢰성을 전달한다.

[0016] 본 발명의 실시예에 따르면, 오라클 턱시도 시스템과 같은 트랜잭셔널 미들웨어 시스템은 오라클 엑사로직 미들웨어 머신과 같은 복수의 프로세서들 및 IB 네트워크와 같은 고성능 네트워크 연결을 갖는 고속 머신들의 장점을 취할 수 있다. 추가적으로, (간단히 "턱시도"로서도 지칭되는) 오라클 턱시도 시스템은 공유된 캐시 아키텍쳐를 갖는 클러스터화된 데이터베이스이며 클라우드 아키텍쳐의 컴포넌트일 수 있는 오라클 리얼 어플리케이션(RAC) 엔터프라이즈 데이터베이스와 같은 클러스터화된 데이터베이스의 장점을 취할 수 있다. 오라클 RAC은, 매우 스케일러블하고 비즈니스 어플리케이션들에 이용가능한 데이터베이스를 제공하기 위해 종래의 SN(shared-nothing) 및 SD(shared-disk) 접근법들의 제한들을 극복할 수 있다.

[0017] 본 발명의 실시예에 따르면, 오라클 턱시도 시스템은 고성능 분산 비즈니스 어플리케이션들의 구성, 실행 및 어드미니스트레이션이 이루어지게 하는 소프트웨어 모듈들의 세트를 제공하며, 다수의 멀티-티어 어플리케이션 개

발 툴들에 의해 트랜잭셔널 미들웨어로서 이용되어 왔다. 텍시도는 분산 컴퓨팅 환경들에서 분산 트랜잭션 프로세싱을 관리하기 위해 이용될 수 있는 미들웨어 플랫폼이다. 이는 엔터프라이즈 레거시 어플리케이션들을 언로킹하고 이들을 서비스 지향 아키텍쳐로 확장하면서도 비제한된 스케일러빌리티 및 표준 기반 상호운용성을 제공하는 입증된 플랫폼이다.

[0018] 추가적으로, 오라클 텍시도 시스템은 2PC(two-phase commit) 프로세싱을 위한 XA 표준, X/Open ATMI API 및 언어 국제화(internationalization)를 위한 X/Open 포터빌리티 가이드(XPG) 표준들의 지원을 포함하여 Open 그룹들의 X/Open 표준들에 부합할 수 있다. 트랜잭셔널 어플리케이션 서버는 XA 표준을 이용할 때 XA 서버로서 지칭될 수 있다. 예를 들어, 텍시도 그룹에 속하는 각각의 텍시도 어플리케이션 서버는 OPENINFO 속성을 이용하여 구성될 수 있다. 텍시도 그룹 내의 모든 XA 서버들은 리소스 관리자(RM)로의 연결을 확립하기 위해 OPENINFO 속성을 이용할 수 있다.

[0019] 인스턴스 인지

[0020] 도 1은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 리소스 관리자(RM) 인스턴스 인지를 지원하는 예를 도시한다. 도 1에 도시된 바와 같이, 트랜잭셔널 환경(100)의 트랜잭셔널 시스템(101)은 데이터베이스와 같은 데이터 소스(102)와 관련된 하나 이상의 리소스 관리자(RM) 인스턴스들을 이용하여 트랜잭션 프로세싱을 지원할 수 있다.

[0021] 본 발명의 실시예에 따르면, 트랜잭셔널 시스템(101)은 데이터 소스(102) 내의 RM 인스턴스 정보(104)를 인지할 수 있다. 예를 들어, 트랜잭셔널 시스템(101) 내의 트랜잭션 서버(103)는 사용자 콜백(110)의 장점을 취함으로써 데이터 소스(102)로부터 RM 인스턴스 정보(104)를 획득할 수 있다. 트랜잭셔널 시스템(101)은 사용자 콜백(110)을 레지스트래이션하기 위해 서로 다른 메커니즘들, 예컨대, 정적 레지스트래이션 메커니즘 및 동적 레지스트래이션 메커니즘을 이용할 수 있다.

[0022] 정적 레지스트래이션 메커니즘은 XA 서버가 지원될 때 이용될 수 있다. XA 서버는 트랜잭션을 제어하기 위해 균일한(uniform) XA 인터페이스를 이용하는 서버이다. 예를 들어, 텍시도에서, 정적 레지스트래이션은 xa_open() 함수가 콜된 후 tpopen() 함수에서 인보크될 수 있다. 일단, 레지스트래이션이 성공적이면, 사용자 콜백(110)은 트랜잭셔널 서버(103)가 데이터베이스, 예컨대 오라클 데이터베이스로의 연결을 확립할 때 인보크될 수 있다. 추가적으로, 사용자 콜백(110)은 xa_close() 함수가 콜되기 전에 tpclose() 함수에서 디레지스트래이션될(deregistered) 수 있다.

[0023] 대안적으로는, 트랜잭션 서버(103)는 예컨대 데이터 소스(102)와 관련된 공유 라이브러리(105)에 기초하여 사용자 콜백(110)을 동적으로 레지스터할 수 있다. 예를 들어, 텍시도는 사용자가 (예컨대, OCI 또는 Pro*c/c++을 통해) 비-XA 서버를 이용하여 오라클 데이터베이스에 연결될 때 콜백을 동적으로 레지스터할 수 있다. 텍시도는 먼저, 오라클 OCI 라이브러리 OCI API를 동적으로 로드하고, 관련 OCI 환경 핸들을 획득할 수 있다. 그 다음, 텍시도는 OCISessionBegin 함수에서 OCIUserCallbackRegister를 통해 사용자 콜백을 레지스터할 수 있다.

[0024] 도 1에 도시된 바와 같이, 시스템은 트랜잭셔널 서버(103)와 관련된 관련 컨텍스트(106)에 획득된 인스턴스 정보(104)를 저장할 수 있다. 추가적으로, 트랜잭셔널 서버(103)는 공유 메모리(107) 내의 서로 다른 상태 테이블들(108)(예컨대, 텍시도에서 글로벌 BB(bulletin board))에 인스턴스 정보(104)를 저장할 수 있다. 이 테이블들(108)은 서로 다른 노드들에 동기화될 수 있고, 복수의 트랜잭셔널 서버들(예컨대, 서버들(111 내지 112)) 및/또는 네이티브 클라이언트들에 의해 액세스될 수 있다.

[0025] 도 2는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 다양한 상태 테이블들을 유지하는 예를 도시한다. 도 2에 도시된 바와 같이, 트랜잭셔널 시스템(200)은 공유 메모리(201) 내의 서로 다른 상태 테이블들에 인스턴스 정보(220)를 저장할 수 있다.

[0026] 이 상태 테이블들은 고유한 RM/데이터베이스 이름들을 저장하는 리소스 관리자 테이블(211), RM/데이터베이스 인스턴스 이름들을 저장하는 인스턴스 테이블(212) 및 RM/데이터베이스 서비스 이름들을 저장하는 서비스 테이블(213)을 포함할 수 있다. 이러한 정보는 다른 서버들이 특정 서버와 관련된 인스턴스 정보를 알도록 도울 수 있다.

[0027] 추가적으로, 트랜잭셔널 시스템(200)은 공유 메모리(201)에 서버 테이블(ST)(214)을 유지할 수 있다. ST(214)는 하나 이상의 서버 테이블 엔트리(STE)들을 포함할 수 있고, 이들 각각은 인스턴스 테이블(212)에 인덱스를 저장할 수 있다. 예를 들어, 각각의 STE는 서버가 단일-스레드 서버인 경우 인스턴스 식별자(ID)를 저장할 수 있다.

[0028] 도 2에 도시된 바와 같이, 서버 테이블(214)은 공유 메모리(201) 내의 다른 테이블들(211 내지 213)을 포인팅할 수 있다. 따라서, 트랜잭셔널 시스템(200)은 (특정 서버와 현재 연결된 RM 인스턴스에 관한 정보와 같은) 인스턴스 정보를 획득하기 위해 서버 테이블(214)을 이용할 수 있고, 트랜잭셔널 시스템(200)은 서로 다른 상태 테이블들(211 내지 213)에 저장된 인스턴스 정보를 직접적으로 이용하지 않을 수 있다.

[0029] 도 3은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 서버 테이블(ST)을 지원하는 예를 도시한다. 도 3에 도시된 바와 같이, 트랜잭셔널 시스템(300) 내의 서버 테이블(ST)(310)은 하나 이상의 서버 테이블 엔트리(STE)들(311 내지 312)을 포함할 수 있고, 이들 각각은 하나 이상의 인스턴스 식별자(ID)들을 포함할 수 있다. 예를 들어, STE(311)는 인스턴스 ID들(321 내지 322)을 포함할 수 있고, STE(312)는 인스턴스 ID들(323 내지 324)을 포함할 수 있다.

[0030] 본 발명의 실시예에 따르면, 각각의 인스턴스 ID(321 내지 324)는 다양한 인스턴스 정보를 포함할 수 있다. 도 3에 도시된 바와 같이, 인스턴스 ID(322)는 인스턴스 이름(301), 데이터베이스 이름(302) 및 서비스 이름(303)을 식별할 수 있다.

[0031] 예를 들어, 인스턴스 ID는 정수, 예컨대 세 개의 섹션들(비트 0 내지 11, 비트 12 내지 19 및 비트 20 내지 31)을 포함하는 32 비트 정수를 이용하여 구현될 수 있다. 제1 섹션, 비트 0 내지 11은 RM/데이터베이스 인스턴스 이름(301)에 대한 엔트리 인덱스를 저장할 수 있다. 제2 섹션, 비트 12 내지 19는 RM/데이터베이스 이름(302)에 대한 엔트리 인덱스를 저장할 수 있다. 제3 섹션, 비트 20 내지 31은 RM/데이터베이스 서비스 이름(303)에 대한 엔트리 인덱스를 저장할 수 있다. 추가적으로, 특별한 값 0xFFFFFFFF이 무효 인스턴스 ID를 표시하기 위해 이용될 수 있다.

[0032] 본 발명의 실시예에 따르면, 트랜잭셔널 시스템(300)은 단순히 관련 비트들을 비교함으로써 인스턴스 ID(322)의 인스턴스 정보를 체크할 수 있다. 따라서, 시스템은, 스트링 비교 동작이 비트 비교 동작보다 더욱 비용이 많이 들기(expensive) 때문에, 스트링 비교로 인한 성능 문제를 회피할 수 있다.

[0033] 도 4는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 정보를 갱신하는 예를 도시한다. 도 4에 도시된 바와 같이, 트랜잭셔널 환경(400) 내의 트랜잭셔널 시스템(401)은 리소스 관리자(RM)(402)로부터, 예컨대 데이터베이스 내의 관련 인스턴스 정보(404)를 획득할 수 있다.

[0034] 사용자 콜백(410)이 콜될 때, 최근 인스턴스 정보(404)가 RM(402)으로부터 검색될 수 있고, 컨텍스트(405)에 저장될 수 있다. 추가적으로, 트랜잭셔널 시스템(401)은 최근 인스턴스 정보(404)의 수신을 나타내는 플래그(409)를 설정할 수 있다.

[0035] 본 발명의 실시예에 따르면, 트랜잭셔널 프로세스(403)가 하나 이상의 체크포인트들과 함께 구성될 수 있다. 예를 들어, 체크포인트들은 서비스 인보케이션(invocation) 전 및 후 그리고 초기화 루틴 후에 트리거될 수 있다. 또한, 체크 포인트들은 연결이 확립 또는 드롭될 때 트리거될 수 있다.

[0036] 도 4에 도시된 바와 같이, 체크 포인트(408)에서, 트랜잭셔널 프로세스(403)는 플래그(409)를 체크할 수 있다. 플래그(409)가 업(up)인 경우, 트랜잭셔널 서버(403)는 검색된 인스턴스 정보(404)에 기초하여 트랜잭션 컨텍스트(407)를 갱신하고 (공유 메모리 내의) 상태 테이블들(406)에 검색된 인스턴스 정보(404)를 저장할 수 있다.

[0037] 도 5는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 다양한 체크 포인트들을 갖는 트랜잭션 프로세스를 지원하는 예를 도시한다. 도 5에 도시된 바와 같이, 단계(501)에서, 트랜잭셔널 프로세스가 시작된다. 그 다음, 트랜잭셔널 프로세스는 단계(502)에서 초기화 프로세스로 진행될 수 있다.

[0038] 초기화 프로세스(502)는 하나 이상의 인스턴스 체크 포인트들, 예컨대 체크 포인트(510)를 포함할 수 있다. 예를 들어, 턱시도에서, 동적 레지스트래이션이 사용될 때, 체크 포인트(510)는 초기화 루틴(예컨대, tpsvrinit) 후 서버의 스타트업 루틴에 위치될 수 있다. 또한, 체크 포인트(510)는 서버가 RM으로의 연결을 확립하는 것을 시도할 때(예컨대, xa_open() 함수 콜이 tpopen() 함수 콜에서 성공적으로 인보크된 후) 트리거될 수 있다.

[0039] 추가적으로, 단계(512)에서, 트랜잭셔널 프로세스는 초기화 프로세스(502) 동안 인스턴스 인지 능력을 인에이블(enable)시키기 위해 인스턴스 정보를 검색할 수 있다.

[0040] 더욱이, 단계(503)에서, 트랜잭셔널 프로세스는 서비스 요청이 존재하는지를 체크할 수 있다. 단계(505)에서, 트랜잭셔널 프로세스는 서비스 디스패쳐, 예컨대, tmsvcdsp()를 인보크할 수 있다. 도 5에 도시된 바와 같이, 체크 포인트(506)는 서비스 루틴(507)이 요청 메시지를 프로세싱하기 위해 인보크되기 전 트리거될 수 있다. 추가적으로, 다른 체크 포인트(508)는 서비스 루틴(507)이 완료된 후 트리거될 수 있다. 단계(509)에서, 트랜잭셔

널 프로세스는 서비스 디스패처를 종료할 수 있다.

[0041] 그렇지 않으면, 서비스 요청이 존재하지 않고, 단계(504)에서, 트랜잭셔널 프로세스가 셧다운되어야 하면, 트랜잭셔널 프로세스가 셧다운 프로세스(511)를 시작할 수 있다. 셧다운 프로세스(511)는 인스턴스 ID 정보를 클리어하기 위해 체크 포인트(513)를 트리거할 수 있다. 예를 들어, 턱시도에서, 체크 포인트(513)는 서버가 RM으로의 연결을 닫아야 할 때(예컨대, xa_close() 함수 콜이 tpclose() 함수 콜에서 콜되기 전에) 트리거될 수 있다. 마지막으로, 트랜잭셔널 프로세스가 단계(514)에서 종료된다.

[0042] 본 발명의 실시예에 따르면, 시스템이 동적 레지스트래이션 대신 정적 레지스트래이션을 이용할 때, 시스템은 체크 포인트(512) 및 체크 포인트(513)에서 서로 다르게 거동(behave)할 수 있다. 인스턴스 정보는 개시자를 체킹함이 없이 직접적으로 검색 및 갱신될 수 있다. 예를 들어, 정적 레지스트래이션이 이용될 때, XA 서버는 RM으로의 연결들을 확립/삭제하기 위해 tpopen()/tpclose() 함수 콜을 이용할 수 있다. 또한, tpopen() 함수는 커스토마이징된 tpsvrinit() 함수 콜에서 콜될 수 있고, tpclose() 함수는 커스토마이징된 tpsvrdone() 함수에서 콜될 수 있다.

[0043] 도 6은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 리소스 관리자(RM) 인스턴스 인지를 지원하는 예시적인 순서도를 도시한다. 도 6에 도시된 바와 같이, 단계(601)에서, 트랜잭셔널 서버는 하나 이상의 리소스 관리자(RM) 인스턴스들과 관련된 RM 인스턴스 정보를 리소스 소스로부터 수신할 수 있고, 수신된 인스턴스 정보는 트랜잭셔널 서버가 현재 연결된 RM 인스턴스를 인지할 수 있게 한다. 그 다음, 단계(602)에서, 시스템은 트랜잭셔널 서버와 관련된 한 이상의 테이블들에 수신된 인스턴스 정보를 저장할 수 있다. 더욱이, 단계(603)에서, 시스템은 트랜잭셔널 서버가 하나 이상의 테이블들에 저장된 인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱하게 한다.

[0044] 트랜잭션 친화성

[0045] 도 7은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 인지에 기초하여 트랜잭션 친화성을 지원하는 예를 도시한다. 도 7에 도시된 바와 같이, 트랜잭셔널 환경(700)은 하나 이상의 리소스 관리자들(RMs), 예컨대, 데이터베이스(704)와 관련된 RM 인스턴스(702)를 이용하여 글로벌 트랜잭션(710)의 프로세싱을 지원할 수 있다.

[0046] 트랜잭셔널 시스템(701)은 (예컨대, 디폴트 라우팅 정책을 이용하여) 트랜잭셔널 서버(703)에 데이터베이스 연결을 위한 요청(711)을 라우팅할 수 있다. 추가적으로, 시스템은 트랜잭셔널 서버(703)에 친화성 컨텍스트(705)를 할당할 수 있다. RM 인스턴스(702)를 식별하는 정보를 포함하는 친화성 컨텍스트(705)는 공유 메모리(예컨대, 턱시도에서 글로벌 트랜잭션 테이블(GTT))에 저장될 수 있고, 메시지를 이용하여 전파될 수 있다. 예를 들어, 트랜잭셔널 서버(703)는 상기 인스턴스 인지 특징에 기초하여 친화성 컨텍스트(705)를 통해 RM 인스턴스 이름, RM/데이터베이스 이름 및 RM/데이터베이스 서비스 이름을 획득할 수 있다.

[0047] 더욱이, 후속적인 요청(712)은 친화성 컨텍스트(705)에 기초하여 트랜잭셔널 서버(703)에 라우팅될 수 있다. 추가적으로, 글로벌 트랜잭션(710)이 완료될 때까지(또는 클라이언트 컨텍스트가 종료될 때까지) 다른 후속적인 요청들이 또한, RM 인스턴스(702)에 연결된 트랜잭셔널 서버(703)에 라우팅될 수 있다.

[0048] 본 발명의 실시예에 따르면, 트랜잭션 친화성은 RM 인스턴스(702)에 연결된 관련 데이터베이스 요청들(711 내지 712)이 동일한 트랜잭셔널 서버(703)에 라우팅될 수 있도록 한다. 따라서, 트랜잭션 친화성이 캐시 히트(cache hit)들의 가능성을 증가시킴으로써 데이터베이스 성능을 향상시킬 수 있기 때문에, 트랜잭션 친화성은 데이터베이스 클러스터 활용을 최대화함으로써 어플리케이션 성능을 향상시킬 수 있다.

[0049] 본 발명의 실시예에 따르면, 시스템은 글로벌 트랜잭션(710)을 수행하기 위해 다른 라우팅 정책들과 함께 트랜잭션 친화성 라우팅 정책을 적용할 수 있다. 예를 들어, 다음 라우팅 우선순위(precedence)가 턱시도에서 지원될 수 있다.

[0050] 1. 도메인에 대한 트랜잭션 우선순위 라우팅

[0051] 2. 클라이언트/서버 친화성 라우팅

[0052] 3. 트랜잭션 친화성 라우팅

[0053] 4. 서비스 로드에 따른 로드 밸런싱

[0054] 도 8은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 트랜잭션 친화성 라우팅을 지원하는 예를 도시한다. 도 8에 도시된 바와 같이, 트랜잭셔널 시스템은 트랜잭션 친화성 라우팅 정책을 이용하여 글로벌 트랜잭션의 프로세

싱을 지원할 수 있다.

[0055] 단계(801)에서, 시스템은 데이터베이스 연결을 수반하는 트랜잭션 요청을 수신할 수 있다. 그 다음, 단계(802)에서, 시스템은 기존 친화성 컨텍스트들이 존재하는지를 체크할 수 있다.

[0056] 친화성 컨텍스트가 글로벌 트랜잭션에 관여되지 않으면, 시스템은 단계(809)에서 로드 밸런싱 라우팅을 수행할 수 있다.

[0057] 글로벌 트랜잭션에 관여된 기존 친화성 컨텍스트들이 존재하면, 시스템은 서버를 발견하기 위해 트랜잭션 친화성 라우팅 정책을 적용할 수 있다.

[0058] 단계(803)에서, 시스템은 동일한 인스턴스 이름, 동일한 데이터베이스(DB) 이름 및 동일한 서비스 이름과 관련된 서버를 발견하는 것을 시도할 수 있다.

[0059] 시스템이 서버를 발견하지 못하면, 단계(804)에서 시스템은 동일한 DB 이름 및 동일한 서비스 이름과 관련되고 현재의 글로벌 트랜잭션에 관여되지 않는 그룹에 있는 서버를 발견하는 것을 시도할 수 있다.

[0060] 시스템이 서버를 발견하지 못하면, 단계(805)에서 시스템은 동일한 DB 이름 및 동일한 인스턴스 이름과 관련된 서버를 발견하는 것을 시도할 수 있다.

[0061] 트랜잭션 시스템이 서버를 발견하지 못하면, 단계(806)에서 시스템은 동일한 DB 이름과 관련된 서버를 발견하는 것을 시도할 수 있다.

[0062] 단계(807)에서, 시스템은 기존 친화성 컨텍스트들에 기초하여 서버를 발견할 수 있다. 한편, 단계(808)에서 시스템은 서버를 발견하지 못할 수 있다. 그 다음, 단계(809)에서 시스템은 로드 밸런싱 라우팅에 따라 서버를 발견하는 것을 시도할 수 있다.

[0063] 도 9는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 친화성 컨텍스트와 함께 메시지를 전송하는 예를 도시한다. 도 9에 도시된 바와 같이, 트랜잭셔널 시스템(901)은 트랜잭션 환경(900)에서 트랜잭션 프로세싱을 지원할 수 있다. 더욱이, 트랜잭셔널 시스템(901)에서 트랜잭셔널 서버(903)는 트랜잭션 컨텍스트(907)(예컨대, 턱시도에서 TUXC)에 기초하여 트랜잭션 프로세싱을 지원할 수 있다.

[0064] 도 9에 도시된 바와 같이, 트랜잭셔널 서버(903)는 공유 메모리(902)(예컨대, 턱시도에서 GTT)로부터 관련 친화성 컨텍스트(904)를 획득할 수 있고, 관련 친화성 컨텍스트(914)를 이용하여 트랜잭션 컨텍스트(907)를 갱신할 수 있다. 체크포인트(908)가 트리거될 때, 시스템은 트랜잭션 컨텍스트(907)로부터의 관련 친화성 컨텍스트(914)를 메시지 큐(905)의 메시지(906)에 복사할 수 있다.

[0065] 따라서, 트랜잭셔널 시스템(901)은 서비스에 메시지(906)를 전송하기 전에 서비스 라우팅을 위해 트랜잭션 컨텍스트(907)에서 관련 친화성 컨텍스트(914)를 참조할 수 있다.

[0066] 도 10은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 친화성 컨텍스트와 함께 메시지를 수신하는 예를 도시한다. 도 10에 도시된 바와 같이, 트랜잭셔널 환경(1000)의 트랜잭셔널 시스템(101)은 하나 이상의 메시지들(예컨대, 메시지(1006))을 수신하기 위해 메시지 큐(1005)를 이용할 수 있다.

[0067] 트랜잭셔널 시스템(1001)의 트랜잭셔널 서버(1003)는 최초 부트(initial boot) 후 (요청들을 포함하는) 메시지들을 연속적으로 디큐잉(dequeue)할 수 있다. 도 10에 도시된 바와 같이, 트랜잭셔널 서버(1003)는 메시지 큐(1005)로부터 메시지(1006)를 판독하고 메시지(1006) 내의 서비스 요청을 프로세싱할 수 있다.

[0068] 서비스 요청의 프로세싱 동안, 시스템은 메시지(1006)로부터의 친화성 컨텍스트를 트랜잭션 컨텍스트(1007)에 카피하기 위해 체크포인트(1008)를 트리거할 수 있다. 그 다음, 시스템은 공유 메모리(1002) 내의 친화성 컨텍스트(1004)를 갱신하기 위해 트랜잭션 컨텍스트(1007) 내의 친화성 컨텍스트(1014)를 이용할 수 있다.

[0069] 일단, 메시지(1006)의 요청이 프로세싱되면, 트랜잭셔널 서버(1003) 프로세스는 메시지 큐(1005)로부터 메시지들을 더 판독할 수 있다. 그렇지 않으면, 트랜잭셔널 서버(1003)는 다음 요청이 도착할 때까지 메시지 큐(1005) 상에서 대기할 수 있다.

[0070] 도 11은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 클라이언트 컨텍스트 내의 친화성 라우팅을 지원하는 예를 도시한다. 도 11에 도시된 바와 같이, 트랜잭셔널 환경(1100)의 트랜잭셔널 시스템(1101)은 하나 이상의 리소스 관리자(RM)들, 예컨대 데이터베이스(1104)와 관련된 RM 인스턴스(1102)를 이용하여 클라이언트 컨텍스트(1110)에서 트랜잭션 프로세싱을 지원할 수 있다. 예를 들어, 클라언트 컨텍스트(1110) 내에서, 웹 대화(web

conversation)들이 다수회 연결 및 연결해제될 수 있다. 이러한 연결들 각각 동안, 대화는 쇼핑 카트와 같은 동일한(또는 유사한) 데이터를 참조 및/또는 이에 액세스할 수 있다.

[0071] 본 발명의 실시예에 따르면, 시스템은 (예컨대, 디폴트 라우팅 정책에 기초하여) 트랜잭셔널 서버(1103)에 데이터베이스 연결을 위한 요청(1111)을 라우팅할 수 있다. 추가적으로, 시스템은 트랜잭셔널 서버(1103)에 RM(1102)을 나타내는 친화성 컨텍스트(1105)를 할당할 수 있다.

[0072] 더욱이, 클라이언트 컨텍스트(1110) 내의 하나 이상의 후속적인 요청들(예컨대, 요청(1112))은 클라이언트 컨텍스트(1110)가 종료되거나 또는 관련 트랜잭션이 완료될 때까지 친화성 컨텍스트(1105)에 기초하여 트랜잭셔널 서버(1103)에 라우팅될 수 있다. 따라서, 트랜잭셔널 시스템(1101)은 클라이언트 컨텍스트(1110) 내의 다양한 데이터베이스 동작들이 동일한 RM 인스턴스(1102)에 디렉팅(direct)될 수 있게 한다.

[0073] 본 발명의 실시예에 따르면, 트랜잭셔널 시스템(1101)은 클라이언트 컨텍스트(1110) 내의 친화성에 대한 힌트를 주는 다양한 로드 밸런스 자문 이벤트(load balance advisory event)들을 데이터베이스로부터 수신할 수 있다. 예를 들어, 턱시도에서, 데이터베이스로부터 수신된 로드 밸런싱 자문 이벤트는 파라미터, AFFINITY HINT를 포함할 수 있는 바, 이 파라미터는 친화성이 특정한 인스턴스 및 서비스 결합에 대해 활성 또는 비활성인지를 나타내는 플래그이다. 웹 세션의 기간 동안 지속되는 임시적인 친화성인 AFFINITY HINT 파라미터는 로드 밸런싱 자문이 서비스 상의 목표(goal)를 설정함으로써 인에이블될 때 자동으로 인에이블될 수 있다. 추가적으로, 동일한 서비스를 제공하는 서로 다른 인스턴스들이 AFFINITY HINT에 대한 서로 다른 설정들을 가질 수 있다.

[0074] 본 발명의 실시예에 따르면, 트랜잭셔널 시스템(1101)은 관련 데이터베이스 동작들이 트랜잭션 내에 존재하는 경우, 클라이언트 컨텍스트 기반의 친화성 정책 대신, 트랜잭션 친화성 라우팅 정책을 적용할 수 있다. 한편, 시스템은 디폴트 턱시도 로드 밸런스 라우트 정책에 기초하여 친화도 라우팅 정책에 기초한 클라언트 컨텍스트를 구현할 수 있다.

[0075] 도 12는 본 발명의 실시예에 따른 트랜잭셔널 환경의 서로 다른 도메인들에 걸쳐 친화성 컨텍스트를 전파하는 예를 도시한다. 도 12에 도시된 바와 같이, 트랜잭셔널 환경(1200)은 하나 이상의 리소스 관리자(RM)들, 예컨대 데이터베이스(1215)와 관련된 RM 인스턴스(1205)를 이용하여 트랜잭션 프로세싱을 지원할 수 있다.

[0076] 본 발명의 실시예에 따르면, 시스템은 (예컨대, 디폴트 라우팅 정책을 이용하여) 트랜잭셔널 서버(1203)에 데이터베이스 연결을 위한 요청(1211)을 라우팅할 수 있다. 추가적으로, 시스템은 트랜잭셔널 서버(1203)에 RM 인스턴스(1202)를 나타내는 친화성 컨텍스트(1207)를 할당할 수 있다.

[0077] 더욱이, 트랜잭셔널 환경(1200)의 트랜잭셔널 도메인은 요청(1211)이 서로 다른 도메인들 간에 전달되어야 하는 경우 도메인들에 걸쳐 친화성 컨텍스트 정보를 전파할 수 있다.

[0078] 도 12에 도시된 바와 같이, 트랜잭션 도메인(1201)은 원격 도메인(1202)에 친화성 키 스트링(1208)을 전송하기 전에 친화성 컨텍스트(1207)를 친화성 키 스트링(1208)로 변환(translate)할 수 있다. 친화성 키 스트링(1208)을 수신한 후, 트랜잭션 도메인(1202)은 친화성 키 스트링(1208)을 친화성 컨텍스트(1206)로 변환할 수 있고, 이 친화성 컨텍스트는 트랜잭션 도메인(1202)에서 트랜잭셔널 서버(1004)에 의해 이용될 수 있다.

[0079] 따라서, 하나 이상의 후속적인 요청들(예컨대, 요청(1212))이 친화성 컨텍스트(1206)에 기초하여 RM 인스턴스(1202)에 디렉팅될 수 있다.

[0080] 도 13은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 어플리케이션 서버에 친화성 컨텍스트를 전파하는 예를 도시한다. 도 13에 도시된 바와 같이, 트랜잭셔널 환경(1300)의 트랜잭셔널 시스템(1301)은 하나 이상의 리소스 관리자(RM)들, 예컨대, 데이터베이스(1315)와 관련된 RM 인스턴스(1305)를 이용하여 트랜잭션 프로세싱을 지원할 수 있다.

[0081] 본 발명의 실시예에 따르면, 시스템은 (예컨대, 디폴트 라우팅 정책을 이용하여) 트랜잭셔널 서버(1303)에 데이터베이스 연결을 위한 요청(1311)을 라우팅할 수 있다. 추가적으로, 시스템은 트랜잭셔널 서버(1303)에 RM 인스턴스(1305)를 나타내는 친화성 컨텍스트(1307)를 할당할 수 있다.

[0082] 더욱이, 트랜잭셔널 시스템(1301)(예컨대, Tuxedo TDomain)은 친화성 키 스트링(1308)을 통해 어플리케이션 서버(1302)(예컨대, 웹로직 어플리케이션 서버)에 친화성 컨텍스트(1307) 정보를 전파할 수 있다. 예를 들어, 친화성 컨텍스트(1307)는 커넥터(1304)(예컨대, Tuxedo WTC)를 통해 트랜잭셔널 시스템(1301)과 어플리케이션 서버(1302) 간에 교환될 수 있다.

[0083] 트랜잭셔널 시스템(1301)이 커넥터(1304)에 요청을 전송할 때, 트랜잭셔널 시스템(1301)은 친화성 컨텍스트(1307)를 친화성 키 스트링(1308)으로 변환할 수 있다. 커넥터(1304)가 트랜잭셔널 시스템(1301)으로부터 요청을 수신할 때, 커넥터(1306)는 친화성 키 스트링(1308)을 어플리케이션 서버(1302)에 의해 이용될 수 있는 트랜잭션 컨텍스트(1306)로 변환할 수 있다.

[0084] 커넥터(1304)가 트랜잭셔널 시스템(1301)에 요청을 전송할 때, 커넥터(1304)는 어플리케이션 서버(1302)와 관련된 트랜잭션 컨텍스트(1306)로부터 친화성 컨텍스트를 획득할 수 있다. 트랜잭셔널 시스템(1301)이 커넥터(1304)로부터 요청을 수신할 때, 트랜잭셔널 시스템(1301)은 친화성 키 스트링(1308)을 친화성 컨텍스트(1307)로 변환할 수 있다.

[0085] 따라서, 어플리케이션 서버(1312) 내의 하나 이상의 후속적인 요청들, 예컨대 요청(1312)이 친화성 컨텍스트(1305)에 기초하여 RM 인스턴스(1302)에 디렉팅될 수 있다.

[0086] 도 14는 본 발명의 실시예에 따른 트랜잭셔널 환경에서 인스턴스 인지에 기초하여 트랜잭션 친화성을 지원하기 위한 예시적인 순서도를 도시한다. 도 14에 도시된 바와 같이, 단계(1401)에서, 시스템은 요청을 트랜잭셔널 서버에 라우팅할 수 있고, 트랜잭셔널 서버는 리소스 관리자(RM) 인스턴스에 연결된다. 그 다음, 단계(1402)에서, 시스템은 트랜잭셔널 서버에 친화성 컨텍스트를 할당할 수 있고, 친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타낸다. 더욱이, 단계(1403)에서, 시스템은 친화성 컨텍스트에 기초하여 트랜잭셔널 서버에 요청과 관련된 하나 이상의 후속적인 요청들을 라우팅할 수 있다.

[0087] 공통 트랜잭션 식별자(Common Transaction Identifier(XID))

[0088] 도 15는 본 발명의 실시예에 따른 서로 다른 트랜잭션 시별자(XID)들을 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하는 예를 도시한다. 도 15에 도시된 바와 같이, 트랜잭셔널 시스템(1500)은 서로 다른 리소스 관리자(RM) 인스턴스, 예컨대 데이터베이스(1506)에 연결된 RM 인스턴스들(1504 내지 1505)을 이용하여 글로벌 트랜잭션(1510)의 프로세싱을 지원할 수 있다.

[0089] 본 발명의 실시예에 따르면, 글로벌 트랜잭션(1510)은 글로벌 트랜잭션 식별자(GTRID 1520)와 관련될 수 있다. 글로벌 트랜잭션(1510) 내에서, 동일한 그룹의 관여된 트랜잭셔널 서버들은 하나의 트랜잭션 브랜치(branch)를 공유할 수 있고, 서로 다른 그룹들의 트랜잭셔널 서버들은 서로 다른 트랜잭션 브랜치들을 이용할 수 있다.

[0090] 도 15에 도시된 바와 같이, 트랜잭셔널 시스템(1500)은 글로벌 트랜잭션(1510)을 프로세싱하기 위해 복수의 브랜치들(예컨대, 브랜치들 A 내지 C(1521 내지 1523))을 이용할 수 있다. 브랜치들 A 내지 C(1521 내지 1523) 각각은 브랜치 한정자(qualifier), 예컨대 BQUAL들 A 내지 C(1531 내지 1533)과 관련될 수 있다.

[0091] 본 발명의 실시예에 따르면, 트랜잭셔널 시스템(1500)은 서로 다른 브랜치들(A 내지 C)(1521 내지 1523) 상에서 글로벌 트랜잭션(1510)의 프로세싱을 관리하기 위해 서로 다른 트랜잭셔널 관리자(TM)들, 예컨대 TM A 내지 C(1501 내지 1503)를 이용할 수 있다.

[0092] 예를 들어, 트랜잭션 식별자(XID) A(1511)과 관련된 TM A(1501)은 브랜치 A(1521) 상의 글로벌 트랜잭션(1510)의 프로세싱을 관리하는 역할을 할 수 있다. 트랜잭션 식별자(XID) B(1512)와 관련된 TM B(1502)는 브랜치 B(1522) 상의 글로벌 트랜잭션(1510)의 프로세싱을 관리하는 역할을 할 수 있다. 트랜잭션 식별자(XID) C(1513)과 관련된 TM C(1503)는 브랜치 C(1523) 상의 글로벌 트랜잭션(1510)의 프로세싱을 관리하는 역할을 할 수 있다.

[0093] 도 15에 도시된 바와 같이, 글로벌 트랜잭션(1510)에서 서로 다른 브랜치들 A 내지 C(1521 내지 1523)에 대한 XID들 A 내지 C(1511 내지 1513)는 동일한 GTRID(1520)(및 포맷 ID)를 공유할 수 있고, 서로 다른 브랜치 한정자들(즉, BQUAL들 A 내지 C(1531 내지 1533))를 가질 수 있다.

[0094] 본 발명의 실시예에 따르면, 시스템은 하나보다 많은 트랜잭션 서버들의 그룹들이 글로벌 트랜잭션(1510)에서 인보크될 때, 글로벌 트랜잭션(1510) 상에서 2PC(two-phase commit) 프로세스를 인보크할 수 있다.

[0095] 도 15에 도시된 바와 같이, 시스템은 심지어 글로벌 트랜잭션(1510)에서 서로 다른 참여된 트랜잭셔널 그룹들(예컨대, TM들 A 내지 B(1501 내지 1502))이 사실상 동일한 RM 인스턴스(1504)와 관련될 때에도 글로벌 트랜잭션(1510)을 프로세싱하기 위해 2PC 모델을 이용할 수 있다.

[0096] 본 발명의 실시예에 따르면, 시스템은 하나보다 많은 그룹들이 동일한 리소스 관리자 인스턴스(1504) 상에서 실행될 때 공통 XID를 이용하는 것을 통해 글로벌 트랜잭션(1510)을 프로세싱하는 성능을 향상시킬 수 있다.

[0097] 도 16은 본 발명의 실시예에 따른 공통 XID를 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하는 예를 도시한다. 도 16에 도시된 바와 같이, 트랜잭셔널 시스템(1600)은 데이터베이스(1606)에 연결된 리소스 관리자(RM) 인스턴스들(1604 내지 1605)를 이용하여 GRTID(1620)와 관련된 글로벌 트랜잭션(1610)의 프로세싱을 지원할 수 있다.

[0098] 더욱이, 트랜잭셔널 시스템(1600)은 서로 다른 트랜잭션 그룹들(예컨대, 브랜치들 A 내지 C(1621 내지 1623))에서 서로 다른 트랜잭셔널 어플리케이션 서버들 상의 글로벌 트랜잭션(1610)의 프로세싱을 관리하기 위해 트랜잭셔널 관리자(TM)들 A 내지 C(1601 내지 1603)를 이용할 수 있다.

[0099] 본 발명의 실시예에 따르면, 리소스 관리자(RM)들(1604 내지 1605) 각각은 예컨대, 데이터베이스 이름, 서버 이름 및 인스턴스 이름에 기초하여 고유하게 식별될 수 있다. 데이터베이스 인스턴스 인지 능력에 기초하여, 트랜잭셔널 시스템(1600) 내의 트랜잭셔널 서버는 자신이 현재 연결된 RM 인스턴스를 알 수 있다.

[0100] 도 16에 도시된 바와 같이, 트랜잭셔널 시스템(1600)은 TM A(1601)와 같은 코디네이터를 포함할 수 있다. 코디네이터 TM A(1601)는 RM 인스턴스(16054)를 식별하는 인스턴스 ID(1641)와 관련된다.

[0101] 추가적으로, 글로벌 트랜잭션(1610)은 하나 이상의 참여 서버들(예컨대, 참여자 TM들 B 내지 C(1602 내지 1603))를 포함할 수 있는 바, 이 참여 서버들은 코디네이터(1601)가 위치된 로컬 서버 노드 상에 위치될 수 있거나 또는 원격 서버 노드들에 위치될 수 있다. 참여자 TM들 B 내지 C(1602 내지 1603) 각각은 또한, 연결된 RM 인스턴스를 식별하는 인스턴스 ID(예컨대, 인스턴스 ID들(1642 내지 1643))과 관련될 수 있다.

[0102] 본 발명의 실시예에 따르면, 공통 XID 특징이 글로벌 트랜잭션(1610)을 프로세싱하기 위해 인에이블될 때, 코디네이터 TM A(1601)에 대한 XID(1611)는 글로벌 트랜잭션(1610) 내에서 공유될 수 있다(즉, XID(1611)는 공통 XID로서 이용된다). 따라서, 복수의 그룹들을 수반하고 오라클 데이터베이스와 같은 클러스터화된 데이터베이스 상에서 실행되는 트랜잭셔널 어플리케이션은 데이터베이스 인스턴스 인지의 장점을 취함으로써 트랜잭션 성능을 향상시킬 수 있다.

[0103] 도 16에 도시된 바와 같이, 글로벌 트랜잭션(1610)에 대한 코디네이터 TM A(1601)는 XID(1611) 및 인스턴스 ID(1641)과 같은 다양한 타입의 정보를 글로벌 트랜잭션(1610)의 수명(life cycle) 내에서 다른 참여자 TM들 B 내지 C(1602 내지 1603)에 전파할 수 있다.

[0104] 더욱이, 참여자 TM들 B 내지 C(1602 내지 1603) 각각은 수신된 인스턴스 ID(1641)를 자신 소유의 인스턴스 ID와 비교함으로써 코디네이터 TM A(1601)와 동일한 RM을 공유하는지를 결정할 수 있다. 인스턴스 ID들이 동일하면, 참여자 TM들 B 내지 C(1602 내지 1603)은 공통-XID 서버(또는 그룹)으로서 자신을 마킹할 수 있다.

[0105] 예를 들어, 시스템은 TM B(1602)가 TM A(1601)와 동일한 RM 인스턴스(1604)를 공유하기 때문에 브랜치 B(1622) 상에서 매치를 발견할 수 있다. 따라서, TM B(1602)가 트랜잭션 프로세싱을 지원하기 위해 자신만의 XID 대신 공통 XID(1611)를 이용할 수 있다. 그 다음, TM B(1602)는 자신이 공통 XID(1611)를 이용함을 코디네이터 TM A(1601)에 통지할 수 있다. 그러한 경우에, 코디네이터 TM A(1601)가 글로벌 트랜잭션(1610)을 커미트 또는 롤백하도록 동작할 때, 시스템은 (BQUAL A(1631)에 기초하여) 공통 XID(1611)를 이용하기 때문에 브랜치 B(1622)를 무시할 수 있다.

[0106] 한편, 시스템은 TM C(1603)가 서로 다른 RM(1605)와(인스턴스 ID(1643)과) 관련되기 때문에 (XID C(1613) 및 BQUAL C(1633)를 이용하는) 브랜치 C(1623) 상에서 매치를 발견하지 않을 수 있다. 옵션에 따라서는, TM C(1603)는 자신이 공통 XID(1611)를 이용하지 않음을 코디네이터 TM A(1601)에 통지할 수 있다. 그 다음, 시스템은 2PC 프로세싱 모델에 따라 트랜잭션 브랜치 C(1623)를 프로세싱할 수 있다.

[0107] 본 발명의 실시예에 따르면, 서로 다른 공통-XID 그룹들, 즉 코디네이터(1601)와 동일한 RM 인스턴스(1604)와 관련된 트랜잭셔널 서버들의 그룹들은 공통 XID(1611)를 통해 RM 인스턴스(1604)에 액세스할 수 있다.

[0108] 더욱이, 커미트 요청이 인보크될 때, 코디네이터(1601)는 로컬 공통-XID 그룹들에 어떤 메시지들을 전송하지 않을 수 있다. 시스템은 각각의 로컬 공통-XID 그룹의 상태를 판독 전용(read-only)으로 한 번에 변경할 수 있다. 또한, 원격 공통-XID 그룹들은 코디네이터(1601)로부터 준비 요청(prepare request)을 수신할 수 있고, 어떤 실제 데이터베이스 동작이 없이 자신의 상태를 판독전용으로 변경할 수있다. 따라서, 시스템은 오직, 이 그룹들 중 하나(예컨대, 코디네이터의 그룹)를 준비/커미트시키기만 하면 된다.

[0109] 추가적으로, 시스템은 인스턴스 ID가 변경되는 경우 공통-XID 그룹을 비-공통-XID 그룹이 되도록 변경할 수 있다. 예를 들어, 브랜치 B(1622)가 다른 RM 인스턴스를 이용하도록 변경되면, 시스템은 (예컨대, BQUAL B(1632)

에 기초하여) 2PC 프로세스를 대신 인보크할 수 있다.

[0110] 도 17은 본 발명의 실시예에 따른 트랜잭셔널 환경에서 데이터베이스 인스턴스 인지에 기초하여 1PC 프로세싱 모델을 지원하는 예를 도시한다. 도 17에 도시된 바와 같이, 트랜잭셔널 시스템(1700)은 데이터베이스(1706)에 연결되는 리소스 관리자(RM) 인스턴스(1704)를 이용하여 GRTID(1720)와 관련된 글로벌 트랜잭션(1710)의 프로세싱을 지원할 수 있다.

[0111] 트랜잭셔널 시스템(1701)은 복수의 트랜잭셔널 관리자(TM)들 A 내지 C(1701 내지 1703)를 포함할 수 있고, 이 복수의 TM들은 서로 다른 트랜잭셔널 그룹들(즉, 브랜치들 A 내지 C(1721 내지 1723))에서의 글로벌 트랜잭션(1710)의 프로세싱을 관리하기 위해 이용된다.

[0112] 더욱이, TM들 A 내지 C(1701 내지 1703)는 단일 리소스 관리자(RM) 인스턴스(1704)에 기초하여 글로벌 트랜잭션(1710)의 프로세싱을 관리할 수 있다. TM들 A 내지 C(1701 내지 1703) 각각은 인스턴스 식별자(ID), 예컨대 인스턴스 ID들(1741 내지 1743)을 유지할 수 있다.

[0113] 도 17에 도시된 바와 같이, 트랜잭셔널 시스템(1700)은 코디네이터(예컨대, TM A(1701))를 포함할 수 있다. 글로벌 트랜잭션(1710)에 대한 코디네이터 TM A(1701)는 글로벌 트랜잭션(1710)의 수명 내에서 공통 XID(1711) 및 인스턴스 ID(1741)와 같은 다양한 타입의 정보를 서로 다른 참여 트랜잭셔널 서버들(예컨대, TM들 B 내지 C(1702 내지 1703))에 전파할 수 있다.

[0114] 본 발명의 실시예에 따르면, 트랜잭션 친화성 능력에 기초하여, 시스템은 글로벌 트랜잭션(1710) 내의 모든 관련 요청들을 동일한 RM 인스턴스(1704)에 라우팅할 수 있다. 더욱이, 인스턴스 인지 능력에 기초하여, TM들 A 내지 C(1701 내지 1703)는 서로 다른 인스턴스 ID들(1741 내지 1743) 모두가 동일한 RM 인스턴스(1704)를 식별하기 때문에 단일 RM 인스턴스(1704)만이 글로벌 트랜잭션(1710)에 이용됨을 알 수 있다. 따라서, 코디네이터 TM A(1701)는 트랜잭셔널 환경(1700)에서 글로벌 트랜잭션(1710)을 코디네이팅하기 위한 공통 XID(1711)(예컨대, BQUAL A(1731) 및 GRTID(1720)에 기초한 자신만의 XID)를 이용할 수 있다.

[0115] 도 17에 도시된 바와 같이, 코디네이터 TM A(1701)는 커미트 스테이지에서 다른 그룹들에 어떤 "준비/커미트" 요청을 전송하지 않을 수 있는 바, 그 이유는 이들이 모두 공통-XID 그룹들이기 때문이다. 더욱이, 시스템은 1PC 프로세싱 모델의 장점을 취할 수 있다.

[0116] 본 발명의 실시예에 따르면, 판독 전용 1PC 최적화는 시스템 성능을 현저하게 향상시킬 수 있다. 이는 모든 다른 그룹들이 판독전용으로 리턴되는 경우 예약된 그룹(reserved group) 상에서 1PC 프로세싱을 수행할 수 있다. 성능은 글로벌 트랜잭션의 모든 브랜치들이 예컨대, 동일한 인스턴스 또는 동일한 RAC에서 타이트하게 결합될 때, 향상될 수 있다.

[0117] 예를 들어, 트랜잭셔널 환경(1700)은 글로벌 트랜잭션(1710)을 프로세싱하기 위해 N 개(N>1)의 참여된 그룹들을 가질 수 있다. 이들 중, M 개(M<N)의 참여된 그룹들이 코디네이터와 동일한 인스턴스 ID를 가질 수 있다.

[0118] 데이터베이스 인스턴스 인지가 없이 2PC 프로세싱 모델을 이용하여 시스템은 (예컨대, 도 15에 도시된 바와 같이) 데이터베이스 상에서 N개의 준비 동작들 및 하나의 커미트 동작을 수행할 수 있다. 또한, 시스템은 트랜잭션 로그를 기록할 필요가 있을 수 있다.

[0119] 대안적으로는, (예컨대, 도 16에 도시된 바와 같이) 데이터베이스 인스턴스 인지에 기초하여, 시스템은 (감소된 M개의 준비 동작들과 함께) 데이터베이스 상에서 N-M개의 준비 동작들 및 하나의 커미트 동작을 수행할 필요가 있을 수 있다.

[0120] 더욱이, M = N-1일 때(글로벌 트랜잭션 내의 모든 다른 참여된 그룹들이 코디네이터와 동일한 트랜잭션 브랜치를 공유할 수 있음을 나타냄), 글로벌 트랜잭션을 프로세싱함에 있어서 단 하나의 브랜치가 존재한다. 시스템은 오직, 감소된 N개(또는 M+1개)의 준비 동작들과 함께 하나의 커미트 동작을 수행할 필요가 있을 수 있다. 또한, 시스템은 트랜잭션 로그를 기록할 필요가 없을 수 있다.

[0121] 도 18은 본 발명의 실시예에 따른 트랜잭셔널 미들웨어 환경에서 데이터베이스 인스턴스 인지에 기초하여 글로벌 트랜잭션을 프로세싱하는 예를 도시한다. 도 18에 도시된 바와 같이, 트랜잭셔널 시스템, 예컨대 오라클 턱시도 시스템은 복수의 트랜잭셔널 그룹들, 예컨대 턱시도 그룹들 A 내지 B(1802 내지 1803)를 이용하여 글로벌 트랜잭션의 프로세싱을 지원할 수 있다.

[0122] 더욱이, 턱시도 그룹들 A 내지 B(1802 내지 1803) 각각은 트랜잭션 관리자 서버(TMS)들의 세트를 가질 수 있다.

예를 들어, 그룹 A(1802)는 서버 A(1804) 및 TMS A(1806)를 포함하며, TMS A는 코디네이터로서 역할을 한다. 추가적으로, 그룹 A(1802)는 공유 메모리, 예컨대 턱시도 BB(bulletin board) A(1808)를 포함할 수 있다. 더욱이, 그룹 B(1803)는 서버 B(1805) 및 TMS B(1807), 및 공유 메모리, 예컨대 턱시도 BB B(1809)를 포함한다.

[0123] 도 18에 도시된 바와 같은 예에서, 단계(1811)에서, 클라이언트(1801)는 함수 콜, tpcall(서비스 A)를 인보크함으로써 서버 A(1804) 상에서 서비스(예컨대, 서비스 A)에 액세스할 수 있다. 그 다음, 단계(1812)에서, 서버 A(1804)는 턱시도 BB A(1808)에 관련된 글로벌 트랜잭션 테이블 엔트리(GTTE)들을 생성할 수 있다.

[0124] 추가적으로, 단계(1813)에서, 클라이언트(1801)는 함수 콜, tpcall(서비스 B)를 인보크함으로써 서버 B(1805) 상에서 다른 서비스(예컨대, 서비스 B)에 액세스할 수 있다. 단계(1814)에서, 서버 A(1804)는 그룹 A(1802)에 관한 관련 정보를 턱시도 BB B(1809)에 추가할 수 있다. 또한, 단계(1815)에서, 클라이언트(1801)는 그룹 B(1803)에 관한 관련 정보를 턱시도 BB A(1808)에 추가할 수 있다.

[0125] 더욱이, 단계(1816)에서, 클라이언트(1801)는 함수 콜 tpcommit()을 인보크함으로써 트랜잭션을 커미트하는 것을 요청할 수 있다. 턱시도는 글로벌 트랜잭션에 관여된 모든 그룹들이 동일한 RM 인스턴스 상에서 실행되는 경우 글로벌 트랜잭션 상에서 직접적으로 1PC를 인보크할 수 있다. 단계(1817)에서, 코디네이터 TMS A(1806)는 글로벌 트랜잭션을 커미트하는 것을 진행할 수 있다.

[0126] 1PC 콜을 성공하면, 단계(1818)에서, 코디네이터 TMS A(1806)는 로컬 노드에서 GTTE를 제거할 수 있다. 그 다음, 단계(1819)에서, 코디네이터(1806)는 그룹 B(1803), 원격 공통-XID 그룹에 자신의 브랜치를 잊어릴 것(forget)을 통지할 수 있다. 마지막으로, 단계(1820)에서, TMS B(1807)는 턱시도 BB B(1808)를 갱신할 수 있다.

[0127] 도 19는 본 발명의 실시예에 따른 공통 XID를 이용하여 트랜잭셔널 환경에서 복수의 도메인들에 걸쳐 글로벌 트랜잭션을 프로세싱하는 예를 도시한다. 도 19에 도시된 바와 같이, 트랜잭셔널 시스템(1900)은 데이터베이스(1906)에 연결된 리소스 관리자 인스턴스(1904)에 기초하여 복수의 도메인들, 예컨대 도메인들 A 내지 B(1951 내지 1952)에 걸쳐 글로벌 트랜잭션(1910)의 프로세싱을 지원할 수 있다.

[0128] 추가적으로, 서로 다른 브랜치들 A 내지 B(1921 내지 1923)은 트랜잭셔널 시스템(1900)에서 GTRID(1920)를 공유할 수 있다. 로컬 도메인 A(1951)에서 코디네이터 TM A(1901)는 인스턴스 ID(1941)에 기초하여 트랜잭션 식별자(XID)(1911) 및 인스턴스 정보를 원격 도메인 B(1952)에 전파할 수 있다.

[0129] 본 발명의 실시예에 따르면, 도메인 A(1951) 내에서 고유한 인스턴스 ID(1941)는 서로 다른 서버 부트 업 시퀀스들로 인해 서로 다른 도메인들(예컨대, 도메인 B(1952))에서 서로 다를 수 있다.

[0130] 도 19에 도시된 바와 같이, 도메인들에 걸쳐 직접적으로 인스턴스 ID(1941)를 전파하는 대신, 코디네이터 TM A(1901)는 도메인들에 걸쳐 전파되기 전에 인스턴스 ID(1941)를 포맷화된 스트링(1908)으로 변환할 수 있다. 예를 들어, 포맷화된 스트링(1908)은 데이터베이스 이름, 서버 이름 및 인스턴스 이름을 포함할 수 있다.

[0131] 추가적으로, 도메인 게이트웨이 서버(1905)는 로컬 도메인 A(1951)과 원격 도메인 B(1952) 사이의 통신을 지원하기 위해 이용될 수 있다. 도메인 게이트웨이 서버(1905)의 아웃바운드 인터페이스는 인스턴스 ID(1941)로부터의 인스턴스 정보를 포맷화된 스트링(1908)에 매핑할 수 있다. 도메인 게이트웨이 서버(1905)의 인바운드는 포맷화된 스트링(1908)으로부터의 인스턴스 정보를 인스턴스 ID로 매핑할 수 있다.

[0132] 예를 들어, 턱시도에서, 사용자는 턱시도 그룹들을 비즈니스 이유들로 서로 다른 도메인들로 내눌 수 있다. GWTDOMAIN 서버와 같은 게이트웨이 서버가 서로 다른 도메인들 간의 통신을 지원하기 위해 이용될 수 있다. 더욱이, 코디네이터 도메인에서의 GWTDOMAIN 서버는 프록시로서 기능을 할 수 있다. 추가적으로, GWTDOMAIN 서버는 GWTDOMAIN 서버를 통한 다른 도메인들 내의 모든 관여된 서버들이 공통 XID를 이용하도록 설정될 때 공통 XID를 이용하도록 구성될 수 있다.

[0133] 본 발명의 실시예에 따르면, 원격 도메인 B(1952)은 임포트(import)된 XID(1913)로서 공통 XID(1911)를 저장할 수 있다. 도 19에 도시된 바와 같이, 브랜치 B(1923)(즉, 공통-XID 그룹)은 임포트된 XID(1913)가 존재하고 임포트된 XID(1913)와 관련된 BQUAL A(1931)가 유효하면, 데이터베이스에 액세스하기 위해 임포트된 XID(1913)를 이용할 수 있다.

[0134] 본 발명의 실시예에 따르면, 데이터베이스 인스턴스 인지에 기초하여, 시스템은 또한, 인터-도메인 트랜잭션(inter-domain transaction)이 단일 RM 인스턴스를 수반할 때, 인터-도메인 트랜잭션을 프로세싱함에 있어서

1PC 모델의 장점을 취할 수 있다.

[0135] 도 20은 본 발명의 실시예에 따른 공통 XID를 이용하여 트랜잭셔널 환경에서 글로벌 트랜잭션을 프로세싱하기 위한 예시적인 순서도를 도시한다. 도 20에 도시된 바와 같이, 단계(2001)에서, 글로벌 트랜잭션에 대한 코디네이터는 트랜잭셔널 환경에서 글로벌 트랜잭션의 하나 이상의 참여자들에게 공통 트랜잭션 식별자 및 리소스 관리자 인스턴스에 대한 정보를 전파할 수 있다. 그 다음, 단계(2002)에서, 시스템은 코디네이터와 리소스 관리자 인스턴스를 공유하는 상기 하나 이상의 참여자들이 공통 트랜잭션 식별자를 사용할 수 있게 한다. 더욱이, 단계(2003)에서, 코디네이터는 하나의 트랜잭션 브랜치를 이용하여 리소스 관리자 인스턴스를 공유하는 상기 하나 이상의 참여자들에 대한 글로벌 트랜잭션을 프로세싱할 수 있다.

[0136] 본 발명의 실시예에 따르면, 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 방법이 제공되는 바, 상기 방법은 트랜잭셔널 서버에 요청을 라우팅하는 단계 - 상기 트랜잭셔널 서버는 리소스 관리자(RM) 인스턴스에 연결됨 - 와, 상기 트랜잭셔널 서버에 친화성 컨텍스트를 할당하는 단계 - 상기 친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타냄 - 와, 그리고 상기 친화성 컨텍스트에 기초하여 상기 트랜잭셔널 서버에 상기 요청과 관계된 하나 이상의 후속적인 요청들을 라우팅하는 단계를 포함한다.

[0137] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 RM 인스턴스가 클러스터화된 데이터베이스 내의 인스턴스가 되도록 하는 단계를 포함한다.

[0138] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 요청 및 상기 하나 이상의 후속적인 요청들이 글로벌 트랜잭션과 관련되도록 하는 단계를 포함한다.

[0139] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 하나 이상의 후속적인 요청들을 라우팅하기 위해 친화성 라우팅 정책을 이용하는 단계를 포함하며, 상기 친화성 라우팅 정책은 RM 이름, 인스턴스 이름, 및 서비스 이름에 기초한다.

[0140] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 요청 및 상기 하나 이상의 후속적인 요청들이 클라이언트 컨텍스트와 관련되도록 하는 단계를 포함한다.

[0141] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 트랜잭셔널 서버를 통해 메시지를 전송하기 전에 상기 메시지에 상기 친화성 컨텍스트를 포함시키는 단계를 포함한다.

[0142] 본 발명의 실시예에 따르면, 상기 방법은 또한, 다른 트랜잭셔널 서버로부터 수신된 메시지에 기초하여 상기 트랜잭셔널 서버에서 상기 친화성 컨텍스트를 갱신하는 단계를 포함한다.

[0143] 본 발명의 실시예에 따르면, 상기 방법은 또한, 트랜잭셔널 도메인으로부터의 상기 친화성 컨텍스트를 다른 트랜잭셔널 도메인에 전달하기 위해 친화성 컨텍스트 스트링을 이용하는 단계를 포함한다.

[0144] 본 발명의 실시예에 따르면, 상기 방법은 또한, 트랜잭셔널 시스템으로부터의 상기 친화성 컨텍스트를 어플리케이션 서버에 전달하기 위해 커넥터를 이용하는 단계를 포함한다.

[0145] 본 발명의 실시예에 따르면, 상기 방법은 또한, 어플리케이션 서버로부터의 친화성 컨텍스트를 수신하기 위해 커넥터를 이용하는 단계를 포함한다.

[0146] 본 발명의 실시예에 따르면, 컴퓨터 프로그램은, 시스템에 의해 실행될 때, 상기 시스템으로 하여금 선행하는 방법들 중의 방법을 수행하도록 하는 명령어들을 포함한다.

[0147] 본 발명의 실시예에 따르면, 컴퓨터 프로그램은, 시스템에 의해 실행될 때, 상기 시스템으로 하여금 선행하는 방법들 중의 방법을 수행하도록 하는 명령어들을 포함한다. 본 발명의 실시예에 따르면, 트랜잭셔널 미들웨어 환경에서 글로벌 트랜잭션을 프로세싱하는 것을 지원하기 위한 시스템이 제공되는 바, 상기 시스템은, 하나 이상의 마이크로 프로세서들, 상기 하나 이상의 마이크로 프로세서들 상에서 실행되는 트랜잭셔널 시스템을 포함하며, 상기 트랜잭셔널 시스템은 트랜잭셔널 서버에 요청을 라우팅하는 단계 - 상기 트랜잭셔널 서버가 리소스 관리자(RM) 인스턴스에 연결됨 - 와, 상기 트랜잭셔널 서버에 친화성 컨텍스트를 할당하는 단계 - 상기 친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타냄 - 와, 그리고 상기 친화성 컨텍스트에 기초하여 상기 트랜잭셔널 서버에 상기 요청에 관계된 하나 이상의 후속적인 요청들을 라우팅하는 단계를 수행하도록 동작한다.

[0148] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 RM 인스턴스는 클러스터화된 데이터베이스 내의 인스턴스이다.

[0149] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 요청 및 상기 하나 이상의 후속적인 요청은 글로벌 트랜잭션과 관련된다.

[0150] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 시스템은 상기 하나 이상의 후속적인 요청들을 라우팅하기 위해 친화성 라우팅 정책을 이용하는 동작을 하고, 상기 친화성 라우팅 정책은 RM 이름, 인스턴스 이름, 및 서비스 이름에 기초한다.

[0151] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 요청 및 상기 하나 이상의 후속적인 요청들이 클라이언트 컨텍스트와 관련된다.

[0152] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 시스템은 메시지를 전송하기 전에 상기 메시지에 상기 친화성 컨텍스트를 포함시키도록 동작한다.

[0153] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 시스템은 다른 트랜잭셔널 서버로부터 수신된 메시지에 기초하여 상기 트랜잭셔널 서버에서 상기 친화성 컨텍스트를 갱신하는 동작을 한다.

[0154] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 시스템은 트랜잭셔널 도메인으로부터의 상기 친화성 컨텍스트를 다른 트랜잭셔널 도메인에 전달하기 위해 친화성 컨텍스트 스트링을 이용하는 동작을 한다.

[0155] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 시스템은 트랜잭셔널 시스템으로부터의 상기 친화성 컨텍스트를 어플리케이션 서버에 전달하는 것 및 어플리케이션 서버로부터의 친화성 컨텍스트를 수신하기는 것 중 적어도 하나를 위해 커넥터를 이용하는 동작을 한다.

[0156] 본 발명의 실시예에 따르면, 명령어들이 저장된 비 일시적 머신 판독가능 저장 매체가 제공되며, 상기 명령어들은 실행될 때, 시스템으로 하여금, 트랜잭셔널 서버에 요청을 라우팅하는 단계 - 상기 트랜잭셔널 서버는 리소스 관리자(RM) 인스턴스에 연결됨 - 와, 상기 트랜잭셔널 서버에 친화성 컨텍스트를 할당하는 단계 - 상기 친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타냄 - 와, 그리고 상기 친화성 컨텍스트에 기초하여 상기 트랜잭셔널 서버에 상기 요청과 관계된 하나 이상의 후속적인 요청들을 라우팅하는 단계를 수행하도록 한다.

[0157] 본 발명의 실시예에 따르면, 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 방법이 제공되는 바, 상기 방법은 트랜잭셔널 서버를 통해 하나 이상의 리소스 관리자(RM) 인스턴스들과 관련된 RM 인스턴스 정보를 데이터 소스로부터 수신하는 단계 - 상기 수신된 인스턴스 정보는 상기 트랜잭셔널 서버가 현재 연결된 RM 인스턴스를 인지하게 함 - 와, 상기 트랜잭셔널 서버와 관련된 하나 이상의 테이블들에 상기 수신된 인스턴스 정보를 저장(save)하는 단계와, 그리고 상기 트랜잭셔널 서버가 상기 하나 이상의 테이블들에 저장된 인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱하게 하는 단계를 포함한다.

[0158] 본 발명의 실시예에 따르면, 상기 방법은 또한, 복수의 트랜잭셔널 서버들에 의해 액세스가능한 공유 메모리에 상기 하나 이상의 테이블들을 저장하는 단계를 포함한다.

[0159] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 RM에 콜 백 동작(call back operation)을 레지스터하는 단계를 포함하고, 상기 트랜잭셔널 서버는 상기 콜 백 동작이 트리거될 때 RM으로부터 인스턴스 정보를 검색하는 동작을 한다.

[0160] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 콜 백 동작이 정적으로 또는 동적으로 레지스터되도록 하는 단계를 포함한다.

[0161] 본 발명의 실시예에 따르면, 상기 방법은 또한, 컨텍스트에 상기 수신된 인스턴스 정보를 저장하는 단계 및 상기 인스턴스 정보의 수신을 나타내는 플래그를 인에이블시키는 단계를 포함한다.

[0162] 본 발명의 실시예에 따르면, 상기 방법은 또한, 트랜잭션을 수행하는 동안 하나 이상의 체크 포인트들에서 상기 플래그를 체킹하는 단계 및 상기 인스턴스 정보가 갱신될 때 상기 하나 이상의 테이블들에 저장된 인스턴스 정보를 갱신하는 단계를 포함한다.

[0163] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 하나 이상의 테이블들이 하나 이상의 고유한 데이터베이스 이름들을 저장한 RM 테이블, 하나 이상의 인스턴스 이름들을 저장한 인스턴스 테이블 및 하나 이상의 데이터베이스 서비스 이름들을 저장한 서비스 테이블 중 적어도 하나를 포함하도록 하는 단계를 포함한다.

[0164] 본 발명의 실시예에 따르면, 상기 방법은 또한, 상기 하나 이상의 테이블들이 하나 이상의 서버 테이블 엔트리

들을 포함하는 서버 테이블을 포함하도록 하는 단계를 포함한다.

[0165] 본 발명의 실시예에 따르면, 상기 방법은 또한, 각각의 상기 서버 테이블이 상기 트랜잭셔널 서버와 현재 연결된 RM 인스턴스와 관련된 인스턴스 식별자(ID)를 포함하도록 하는 단계를 포함한다.

[0166] 본 발명의 실시예에 따르면, 상기 방법은 또한, 각각의 상기 인스턴스 ID가 복수의 섹션들을 갖는 정수(integer)가 되도록 하는 단계를 포함한다.

[0167] 본 발명의 실시예에 따르면, 컴퓨터 프로그램이 명령어들을 포함하고, 상기 명령어들은 시스템에 의해 실행될 때 상기 시스템으로 하여금 선행하는 방법들 중의 방법을 수행하도록 한다.

[0168] 본 발명의 실시예에 따르면, 트랜잭셔널 환경에서 트랜잭션 프로세싱을 지원하기 위한 시스템이 제공되는 바, 상기 시스템은 하나 이상의 마이크로프로세서들 및 상기 하나 이상의 마이크로프로세서 상에서 실행되는 트랜잭셔널 서버를 포함하고, 상기 트랜잭셔널 서버는 하나 이상의 리소스 관리자(RM) 인스턴스들과 관련된 RM 인스턴스 정보를 데이터 소스로부터 수신하는 단계 - 상기 수신된 인스턴스 정보는 상기 트랜잭셔널 서버가 현재 연결된 RM 인스턴스를 인지하게 함 - 와, 상기 트랜잭셔널 서버와 관련된 하나 이상의 테이블들에 상기 수신된 인스턴스 정보를 저장하는 단계와, 그리고 상기 하나 이상의 테이블들에 저장된 인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱하는 단계를 포함하는 단계들을 수행하도록 동작한다.

[0169] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 서버는 복수의 트랜잭셔널 서버들에 의해 액세스가능한 공유 메모리에 상기 하나 이상의 테이블들을 저장하는 동작을 한다.

[0170] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 서버는 상기 RM에 콜 백 동작을 레지스터하는 동작을 하며, 상기 트랜잭셔널 서버는 상기 콜 백 동작이 트리거될 때 RM으로부터 인스턴스 정보를 검색하는 동작을 한다.

[0171] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 콜 백 동작은 정적으로 또는 동적으로 레지스터되도록 구성된다.

[0172] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 서버는 컨텍스트에 상기 수신된 인스턴스 정보를 저장 및 상기 인스턴스 정보의 수신을 나타내는 플래그를 인에이블시키도록 동작한다.

[0173] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 트랜잭셔널 서버는 트랜잭션을 수행하는 동안 하나 이상의 체크 포인트들에서 상기 플래그를 체킹하고 상기 인스턴스 정보가 갱신될 때 상기 하나 이상의 테이블들에 저장된 인스턴스 정보를 갱신하는 동작을 한다.

[0174] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 하나 이상의 테이블들은 하나 이상의 고유한 데이터베이스 이름들을 저장한 RM 테이블, 하나 이상의 인스턴스 이름들을 저장한 인스턴스 테이블 및 하나 이상의 데이터베이스 서비스 이름들을 저장한 서비스 테이블 중 적어도 하나를 포함한다.

[0175] 본 발명의 실시예에 따르면, 상기 시스템에서, 상기 하나 이상의 테이블들은 하나 이상의 서버 테이블 엔트리들을 포함하는 서버 테이블을 포함한다.

[0176] 본 발명의 실시예에 따르면, 상기 시스템에서, 각각의 상기 서버 테이블은 상기 트랜잭셔널 서버와 현재 연결된 RM 인스턴스와 관련된 인스턴스 식별자(ID)를 포함하고, 상기 인스턴스 ID는 복수의 섹션들을 갖는 정수가 된다.

[0177] 본 발명의 실시예에 따르면, 명령어들이 저장된 비일시적 컴퓨터 판독가능 저장 매체가 제공되며, 상기 명령어들은 실행될 때, 시스템으로 하여금 트랜잭셔널 서버를 통해 하나 이상의 리소스 관리자(RM) 인스턴스들과 관련된 RM 인스턴스 정보를 데이터 소스로부터 수신하는 단계 - 상기 수신된 인스턴스 정보는 상기 트랜잭셔널 서버가 현재 연결된 RM 인스턴스를 인지하게 함 - 와, 상기 트랜잭셔널 서버와 관련된 하나 이상의 테이블들에 상기 수신된 인스턴스 정보를 저장(save)하는 단계와, 그리고 상기 트랜잭셔널 서버가 상기 하나 이상의 테이블들에 저장된 인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱하게 하는 단계를 포함하는 단계들을 수행하도록 한다.

[0178] 본 발명의 실시예에 따르면, 컴퓨터 프로그램이, 시스템에 의해 실행될 때 상기 시스템으로 하여금 선행하는 방법들 중의 방법을 수행하도록 하는 명령어들을 포함한다.

[0179] 본 발명의 많은 특징들이 하드웨어, 소프트웨어, 펌웨어 또는 이들의 조합들에서, 이들을 이용하여 또는 이들의

도움으로 수행될 수 있다. 따라서, 본 발명의 특징들은 (하나 이상의 프로세서들을 포함하는) 프로세싱 시스템을 이용하여 구현될 수 있다.

[0180] 당업자는 상기 특징들이 특정한 상황들 및 어떤 소정 구현의 요건에 따라 서로 함께 그리고/또는 첨부된 특허청구범위에 정의된 특징들과 함께 적절하게 조합될 수 있음을 인지할 것이다.

[0181] 본 발명의 특징들은 컴퓨터 프로그램 물로, 이를 이용하여 또는 이의 도움으로 구현될 수 있는 바, 상기 컴퓨터 프로그램 물은 명령어들이 저장된/본 명세서에 제시된 특징들 중 어느 것을 수행하도록 프로세싱 시스템을 프로그래밍하기 위해 이용될 수 있는 저장 매체(매체들) 또는 컴퓨터 판독가능 매체(매체들)이다. 저장 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.

[0182] 머신 판독가능 매체(매체들) 중 어느 하나에 저장된 경우, 본 발명의 특징들은 프로세싱 시스템의 하드웨어를 제어하기 위해 그리고 프로세싱 시스템으로 하여금 본 발명의 결과들을 활용하는 다른 메커니즘과 상호작용하도록 하기 위해 소프트웨어 및/또는 펌웨어에 통합될 수 있다. 이러한 소프트웨어 또는 펌웨어는 이들로만 한정되는 것은 아니지만, 어플리케이션 코드, 디바이스 드라이버들, 운영 체제들 및 실행 환경들/컨테이너들을 포함할 수 있다.

[0183] 본 발명의 특징들은 또한, 예컨대, ASIC과 같은 하드웨어 컴포넌트들을 이용하여 하드웨어로 구현될 수 있다. 본 명세서에 기술된 기능들을 수행하도록 하드웨어 상태 머신을 구현하는 것은 관련 기술 분야의 당업자들에게 분명할 것이다.

[0184] 추가적으로, 본 발명은 본 발명의 교시들에 따라 프로그램된 하나 이상의 프로세서들, 메모리 및/또는 컴퓨터 판독가능 저장 매체들을 포함하여 하나 이상의 종래의 범용 또는 특수용 디지털 컴퓨터, 컴퓨팅 디바이스, 머신 또는 마이크로프로세서를 이용하여 편리하게 구현될 수 있다. 적절한 소프트웨어 코딩이 소프트웨어 분야의 숙련자들에게 분명할 바와 같이 본 발명의 교시들에 기초하여 숙련된 프로그래머들에 의해 쉽게 준비될 수 있다.

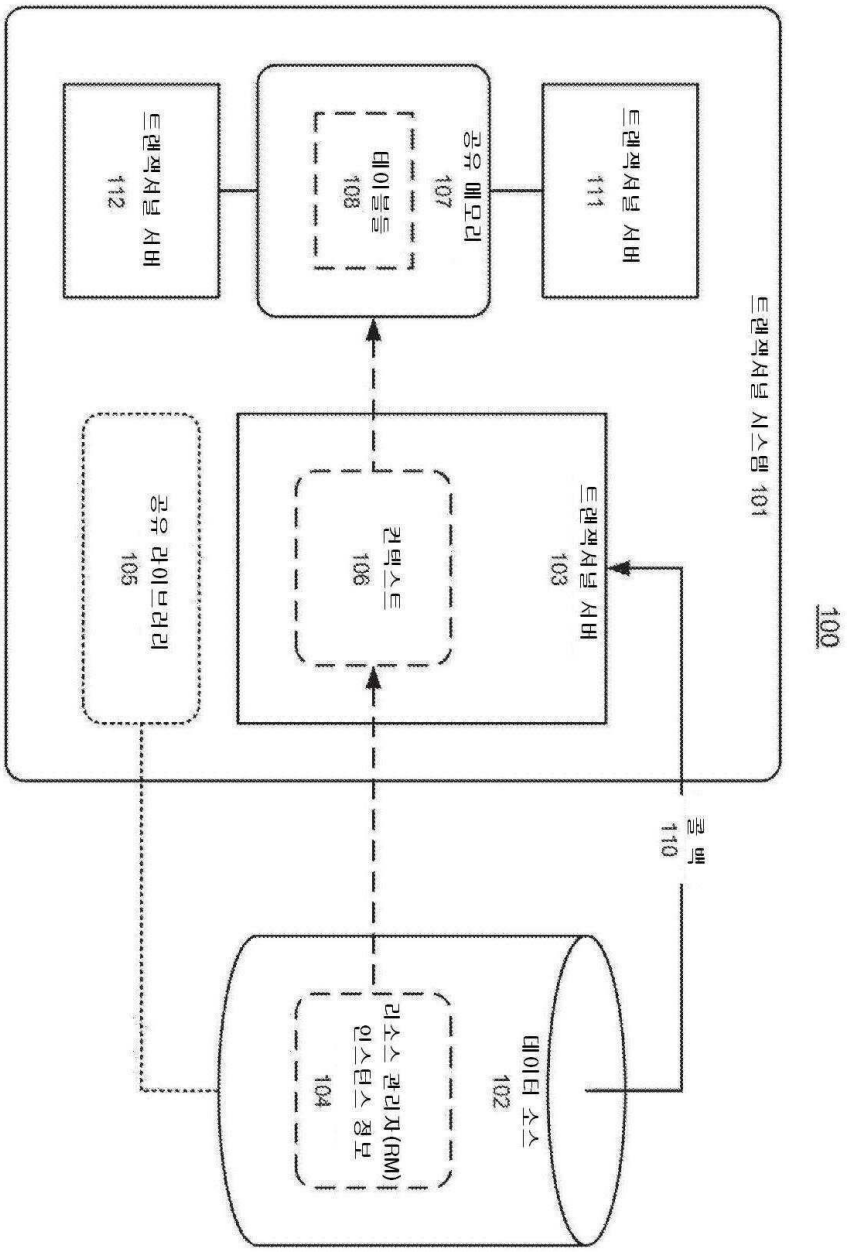
[0185] 본 발명의 다양한 실시예들이 상기에 기술되었지만, 이들은 제한이 아닌 예로서 제시되었음이 이해되어야 한다. 형태 및 세부사항의 다양한 변경들이 본 발명의 사상 및 범위를 벗어남이 없이 이 기술 분야의 숙련자들에게 분명해질 것이다.

[0186] 본 발명은 특정 기능들 및 이들의 관계들의 퍼포먼스를 예시하는 기능 빌딩 블록들의 도움으로 상기에 기술되었다. 이 기능 빌딩 블록들의 경계들(boundaries)은 종종 설명의 편의를 위해 본 명세서에 임의로 정의된다. 대안적인 경계들이 특정 기능들 및 이들의 관계들이 적절하게 수행될 수 있는 한 정의될 수 있다. 따라서, 어떤 이러한 대안적인 경계들은 본 발명의 범위 및 사상 내에 있다.

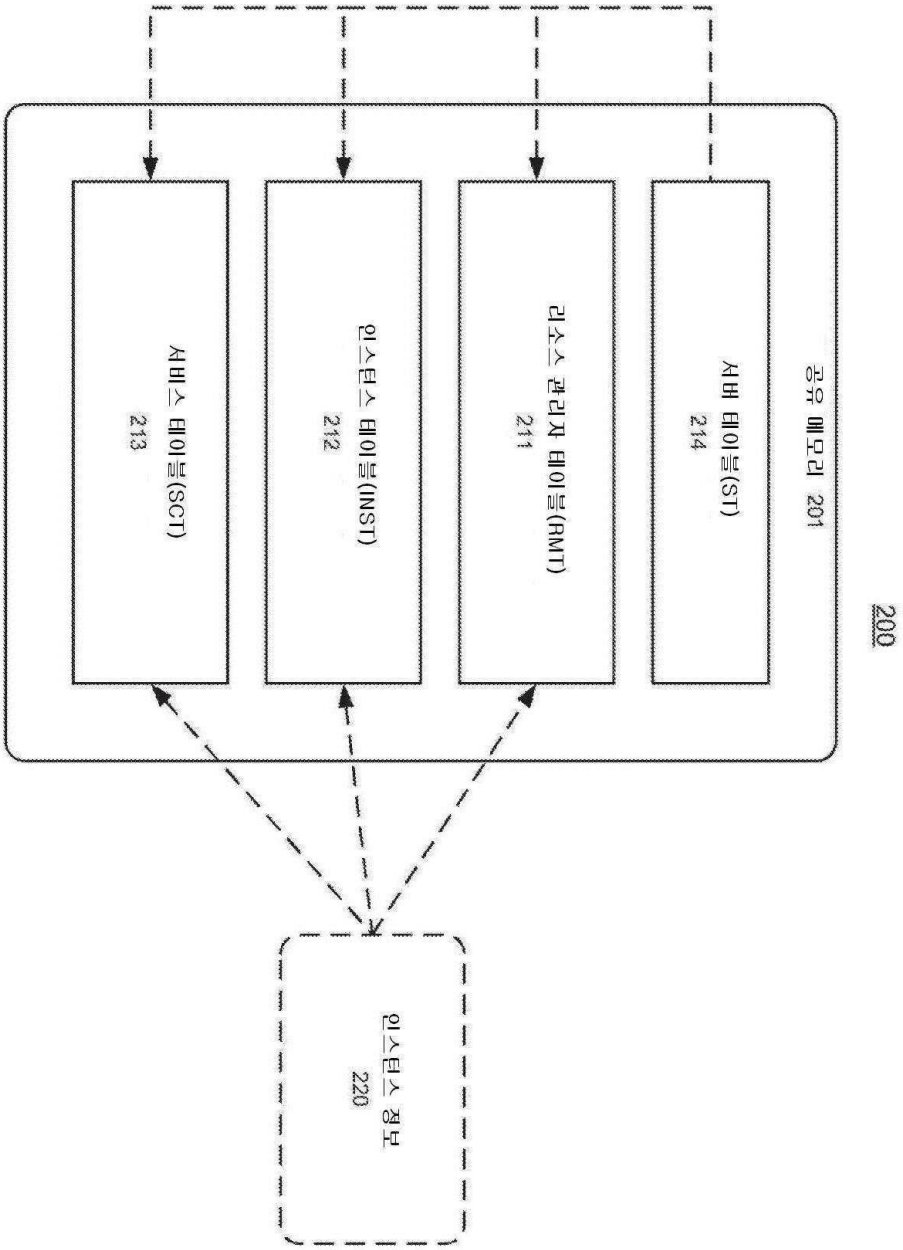
[0187] 본 발명의 상기 설명은 예시 및 설명을 목적으로 제공되었다. 본 설명은 완전한 것(exhaustive)으로 의도되거나 또는 정확히 개시된 형태들로만 본 발명을 제한하고자 의도된 것이 아니다. 본 발명의 범주 및 범위는 상기 설명된 예시적인 실시예들 중 어느 것에 의해 제한되어서는 안된다. 많은 수정들 및 변형들이 이 기술분야의 숙련자에게 분명할 것이다. 수정들 및 변형들은 개시된 특징들의 어떤 관련 조합을 포함한다. 위 실시예들은 본 발명의 원리 및 이의 실용적 응용을 가장 잘 설명하기 위해 선택 및 기술되었으며, 그럼으로써 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

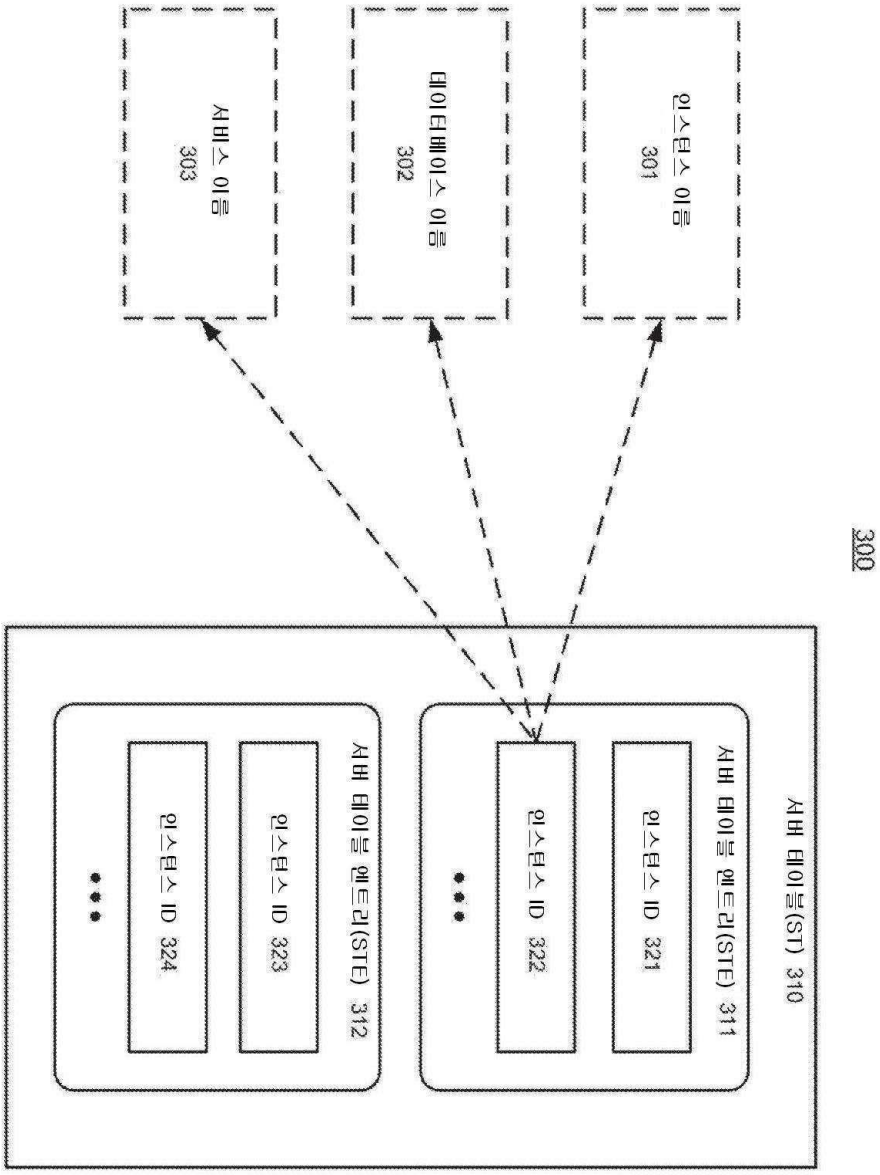
도면1


100
트랜잭셔널 시스템 101
트랜잭셔널 서버
111
공유 메모리
107
테이블들
108
트랜잭셔널 서버
112
트랜잭셔널 서버
103
컨텍스트
106
공유 라이브러리
105
콜백
110
데이터 소스
102
리소스 관리자(RM)
인스턴스 정보
104

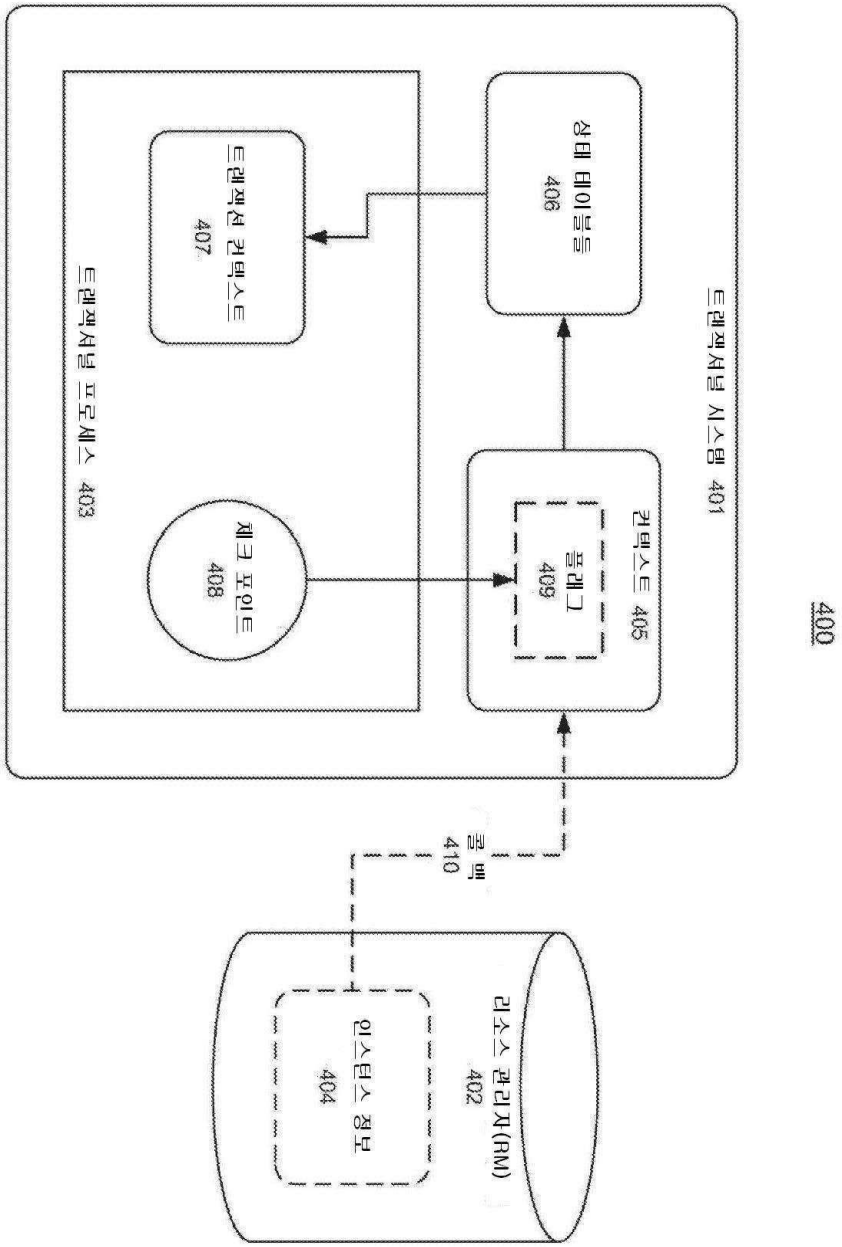
도면2


200
공유 메모리 201
서버 테이블(ST)
214
리소스 관리자 테이블(RMT)
211
인스턴스 테이블(INST)
212
서비스 테이블(SCT)
213
인스턴스 정보
220

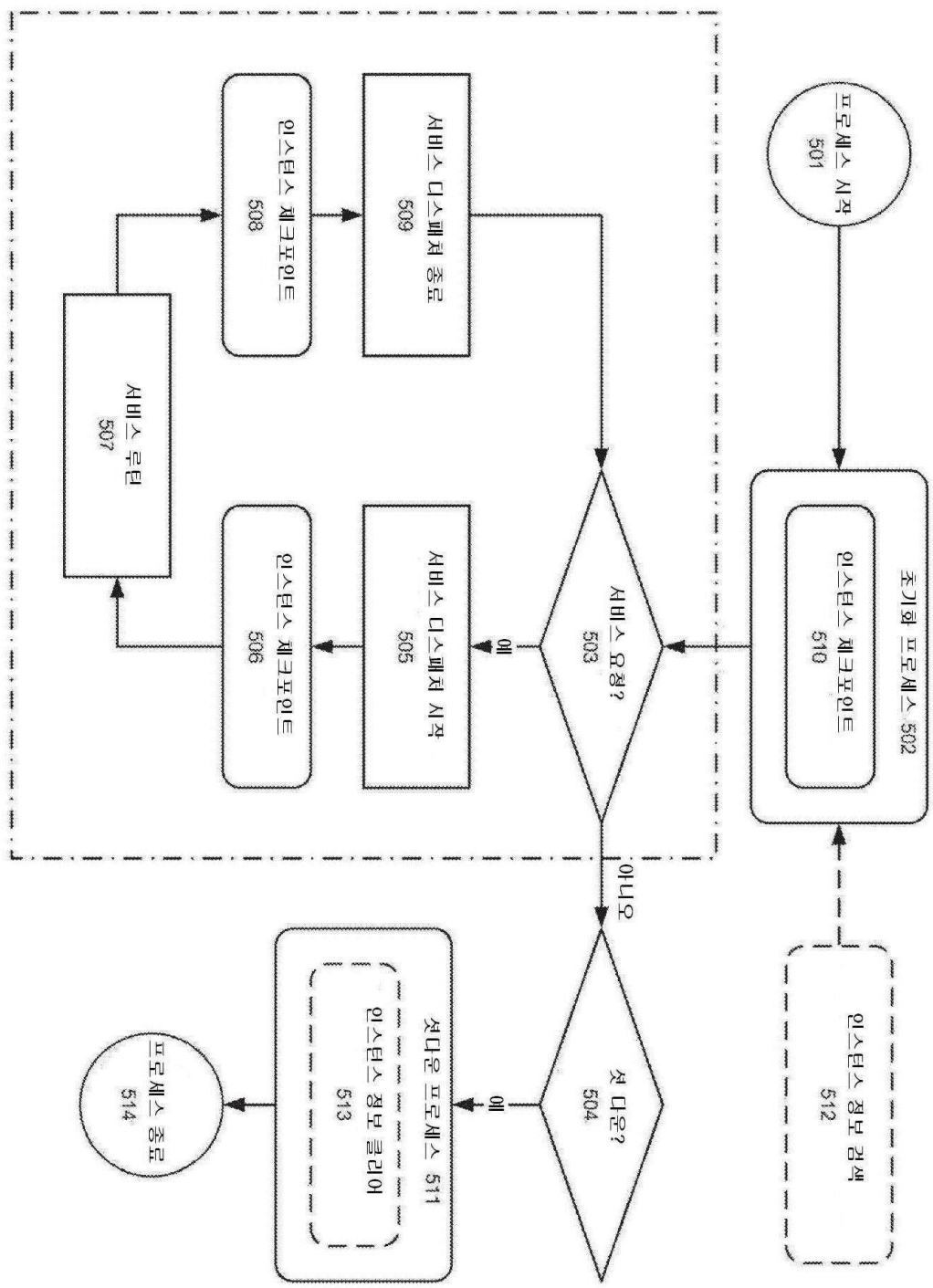
도면3


300
서버 테이블(ST) 310
서버 테이블 엔트리(STE) 311
인스턴스 ID 321
인스턴스 ID 322
서버 테이블 엔트리(STE) 312
인스턴스 ID 323
인스턴스 ID 324
인스턴스 이름 301
데이터베이스 이름 302
서비스 이름 303

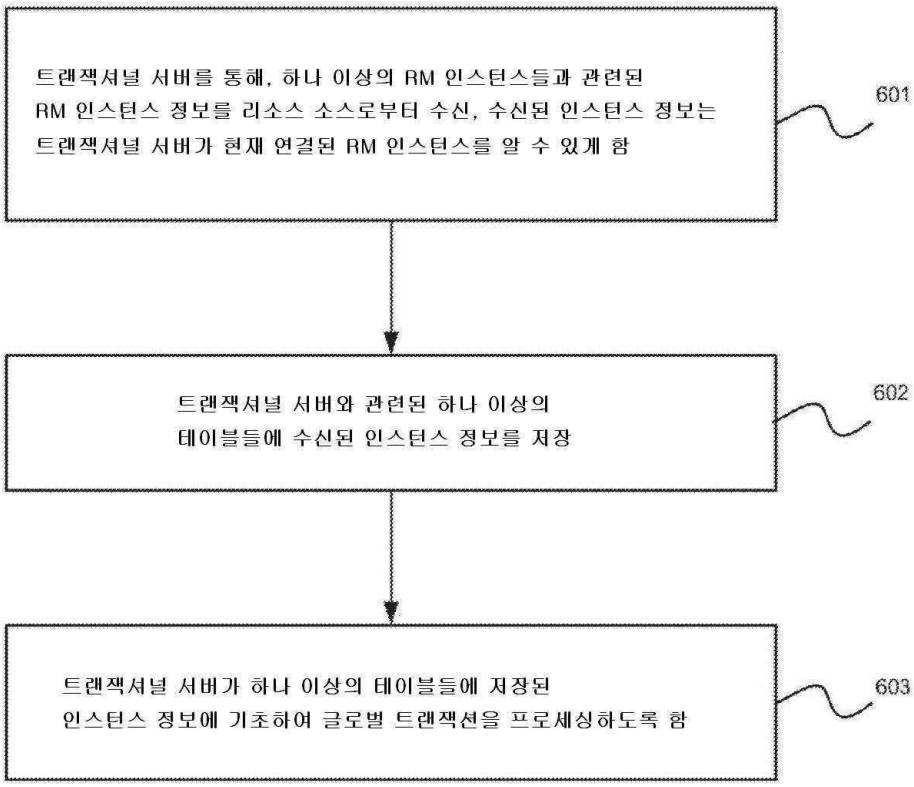
도면4


400
트랜잭셔널 시스템 401
상태 테이블들
406
트랜잭션 컨텍스트
407
트랜잭셔널 프로세스 403
체크 포인트
408
컨텍스트 405
플래그
409
콜 백
410
리소스 관리자(RM)
402
인스턴스 정보
404

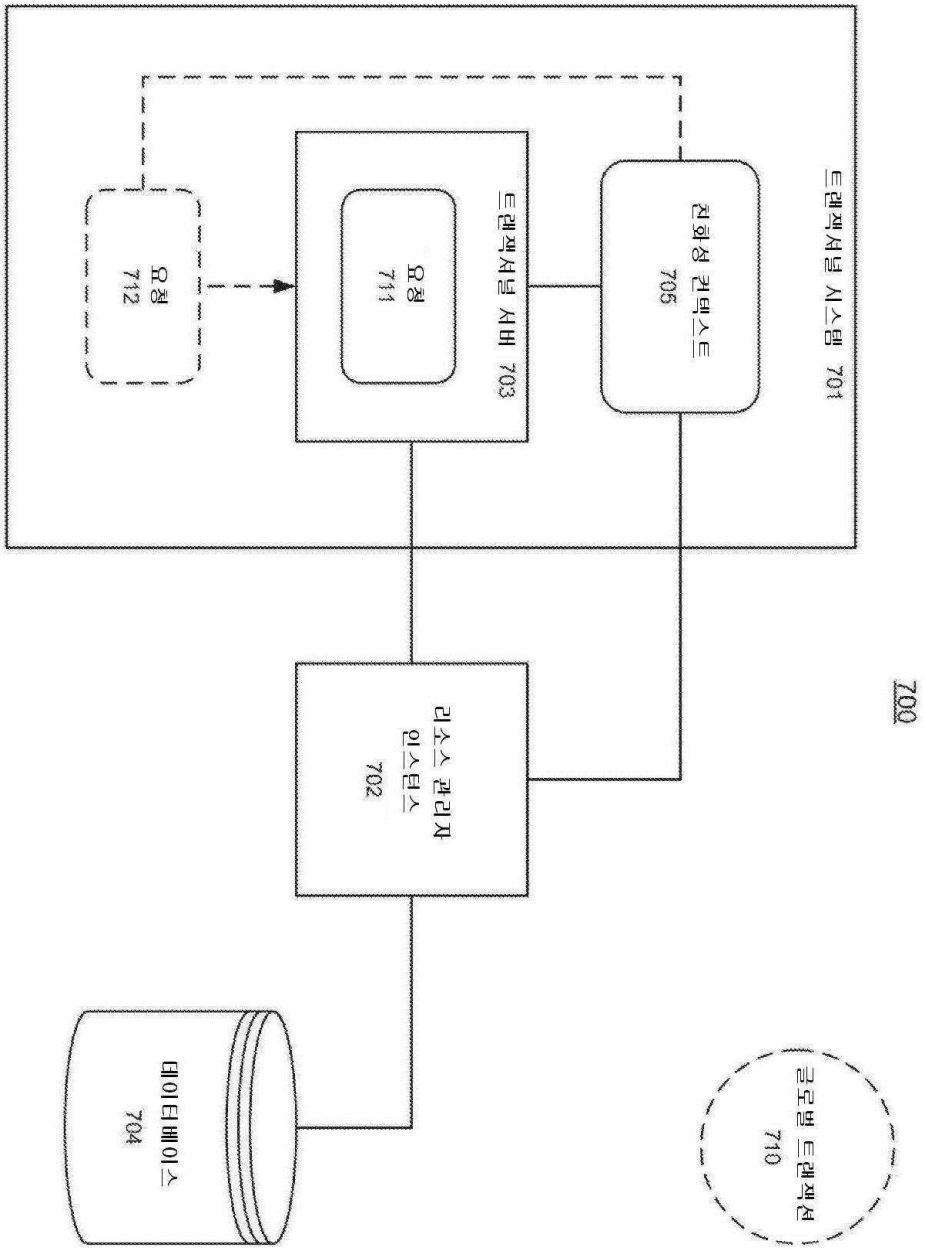
도면5


프로세스 시작
501
초기화 프로세스 502
인스턴스 체크포인트
510
서비스 요청?
503
예
서비스 디스패처 시작
505
인스턴스 체크포인트
506
서비스 유틸
507
인스턴스 체크포인트
508
서비스 디스패처 종료
509
아니오
셧 다운?
504
예
셧다운 프로세스 511
인스턴스 정보 클리어
513
프로세스 종료
514
인스턴스 정보 검색
512

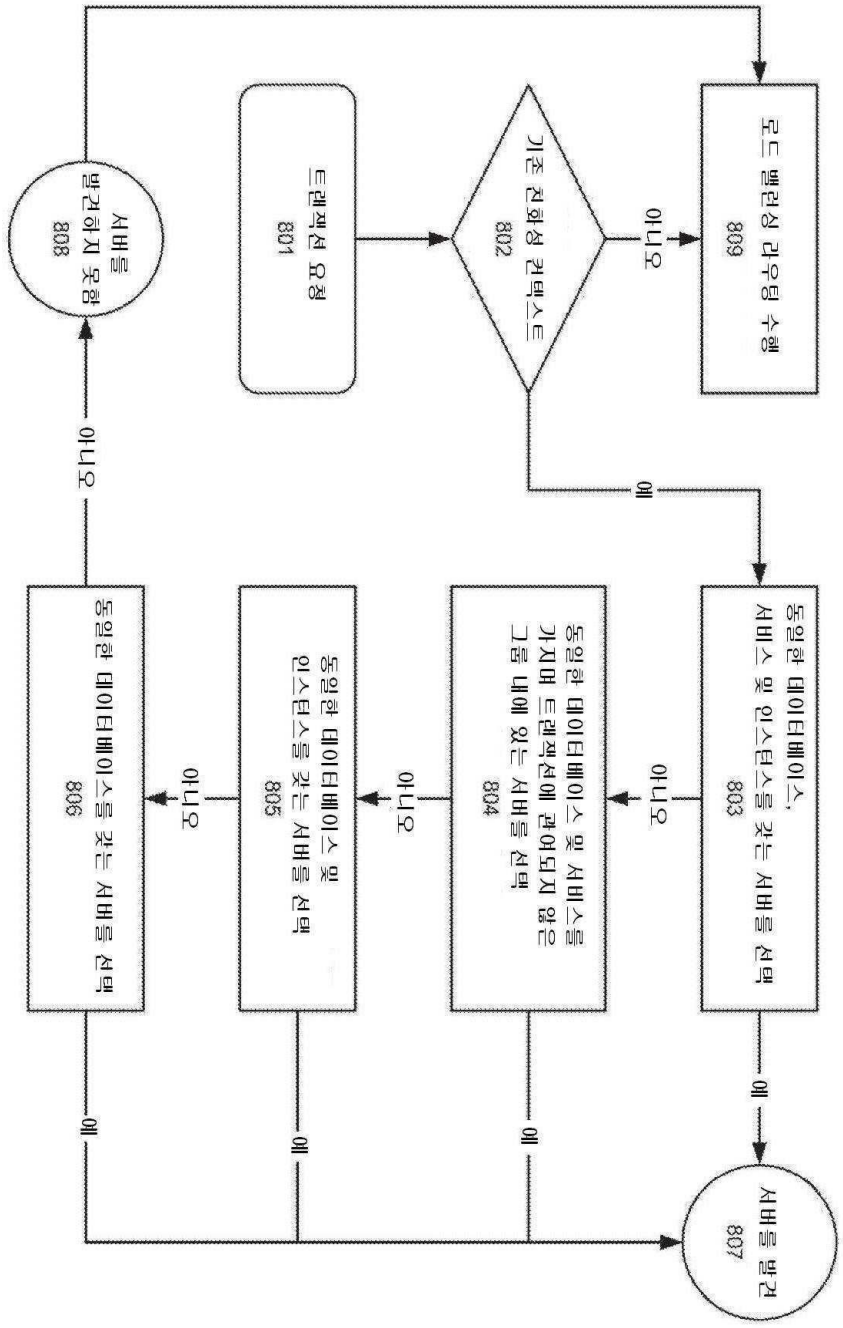
도면6


트랜잭셔널 서버를 통해, 하나 이상의 RM 인스턴스들과 관련된
RM 인스턴스 정보를 리소스 소스로부터 수신, 수신된 인스턴스 정보는
트랜잭셔널 서버가 현재 연결된 RM 인스턴스를 알 수 있게 함
601
트랜잭셔널 서버와 관련된 하나 이상의
테이블들에 수신된 인스턴스 정보를 저장
602
트랜잭셔널 서버가 하나 이상의 테이블들에 저장된
인스턴스 정보에 기초하여 글로벌 트랜잭션을 프로세싱하도록 함
603

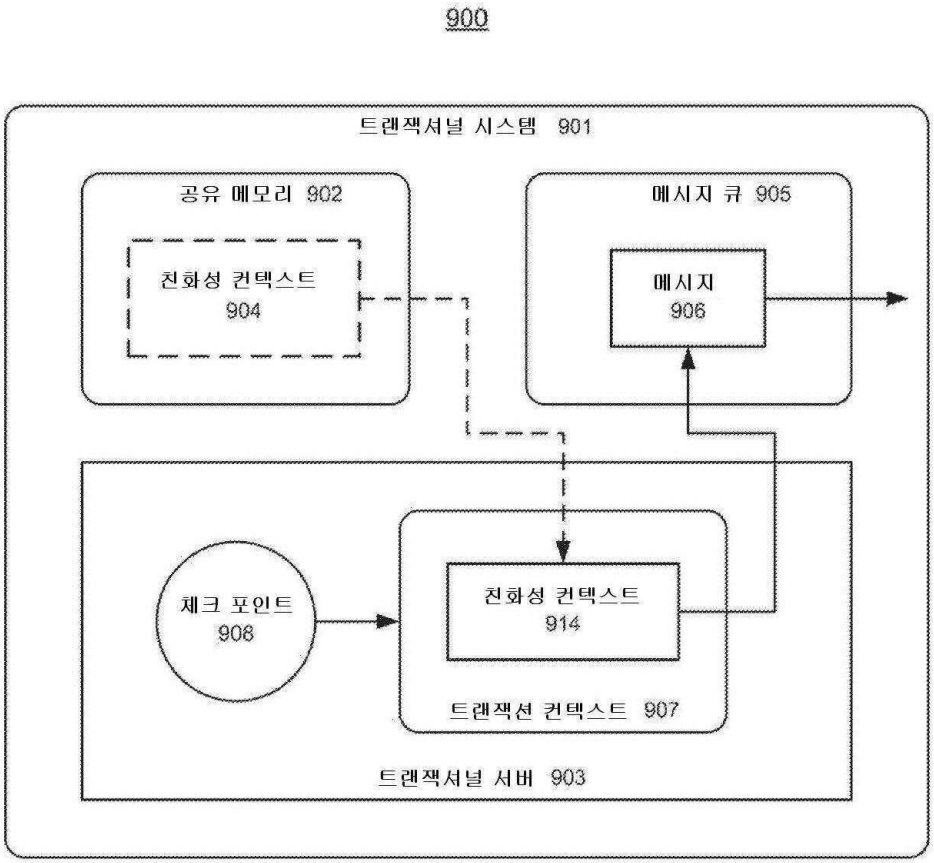
도면7


700
트랜잭션 시스템 701
요청 712
트랜잭션널 서버 703
요청 711
전파성 컨텍스트 705
리소스 관리자 인스턴스 702
데이터베이스 704
글로벌 트랜잭션 710

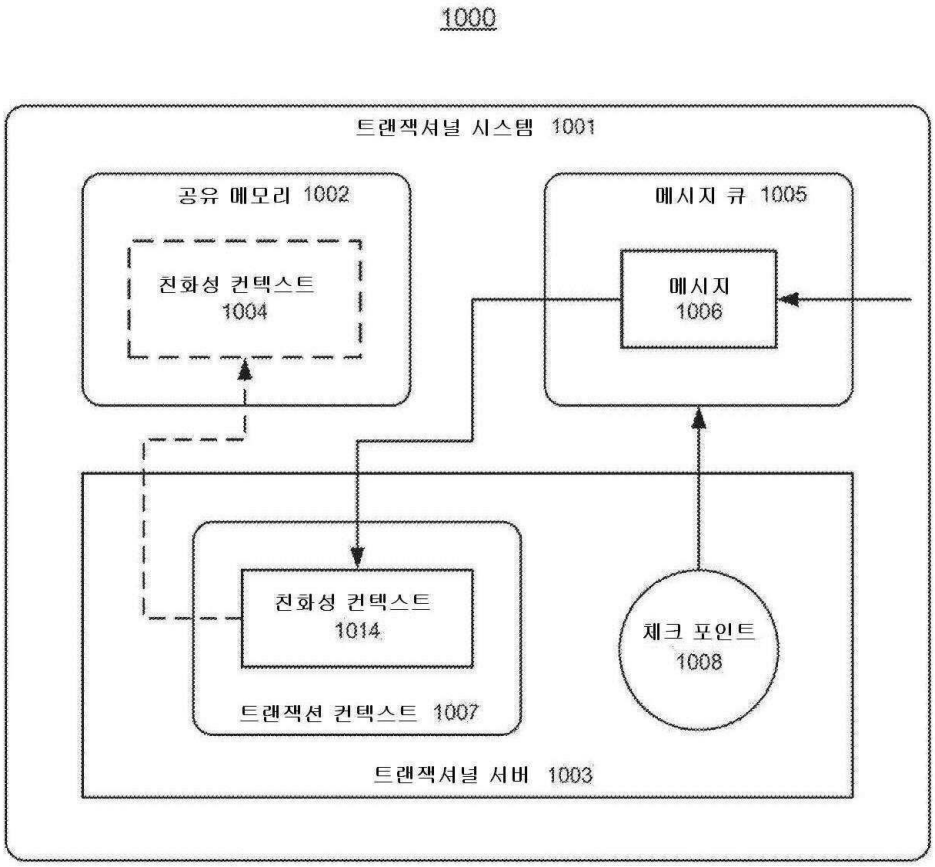
도면8


로드 밸런싱 라우팅 수행
809
기존 컨텍스트
802
아니오
예
트래픽성 요청
801
서버를 발견
807
동일한 데이터베이스, 서비스 및 인스턴스를 갖는 서버를 선택
803
동일한 데이터베이스 및 서비스를 가지며 트래픽선에 편향되지 않은 그룹 내에 있는 서버를 선택
804
동일한 데이터베이스 및 인스턴스를 갖는 서버를 선택
805
동일한 데이터베이스를 갖는 서버를 선택
806
서버를 발견할지 못함
808

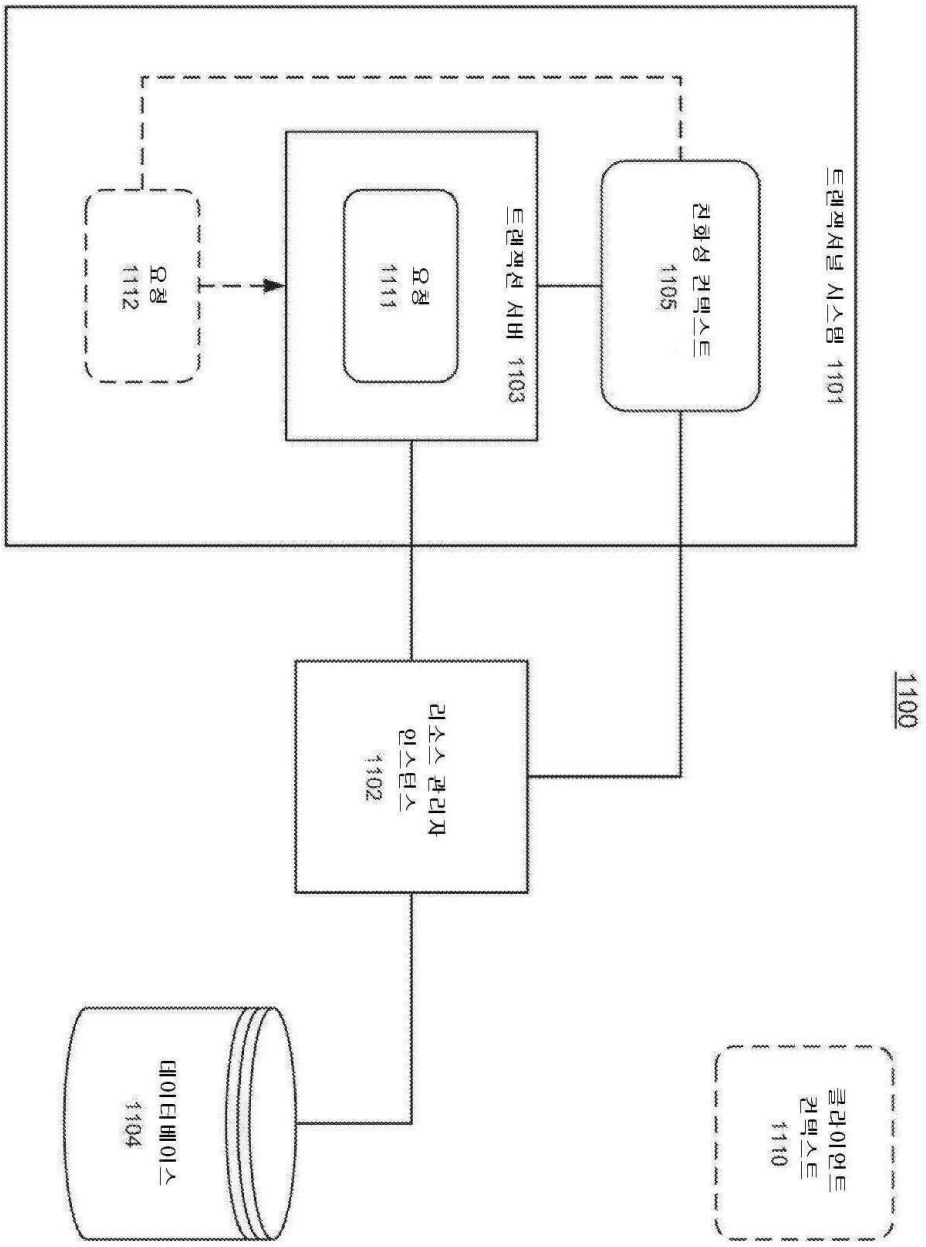
도면9


900
트랜잭셔널 시스템 901
공유 메모리 902
친화성 컨텍스트
904
메시지 큐 905
메시지
906
체크 포인트
908
친화성 컨텍스트
914
트랜잭션 컨텍스트 907
트랜잭셔널 서버 903

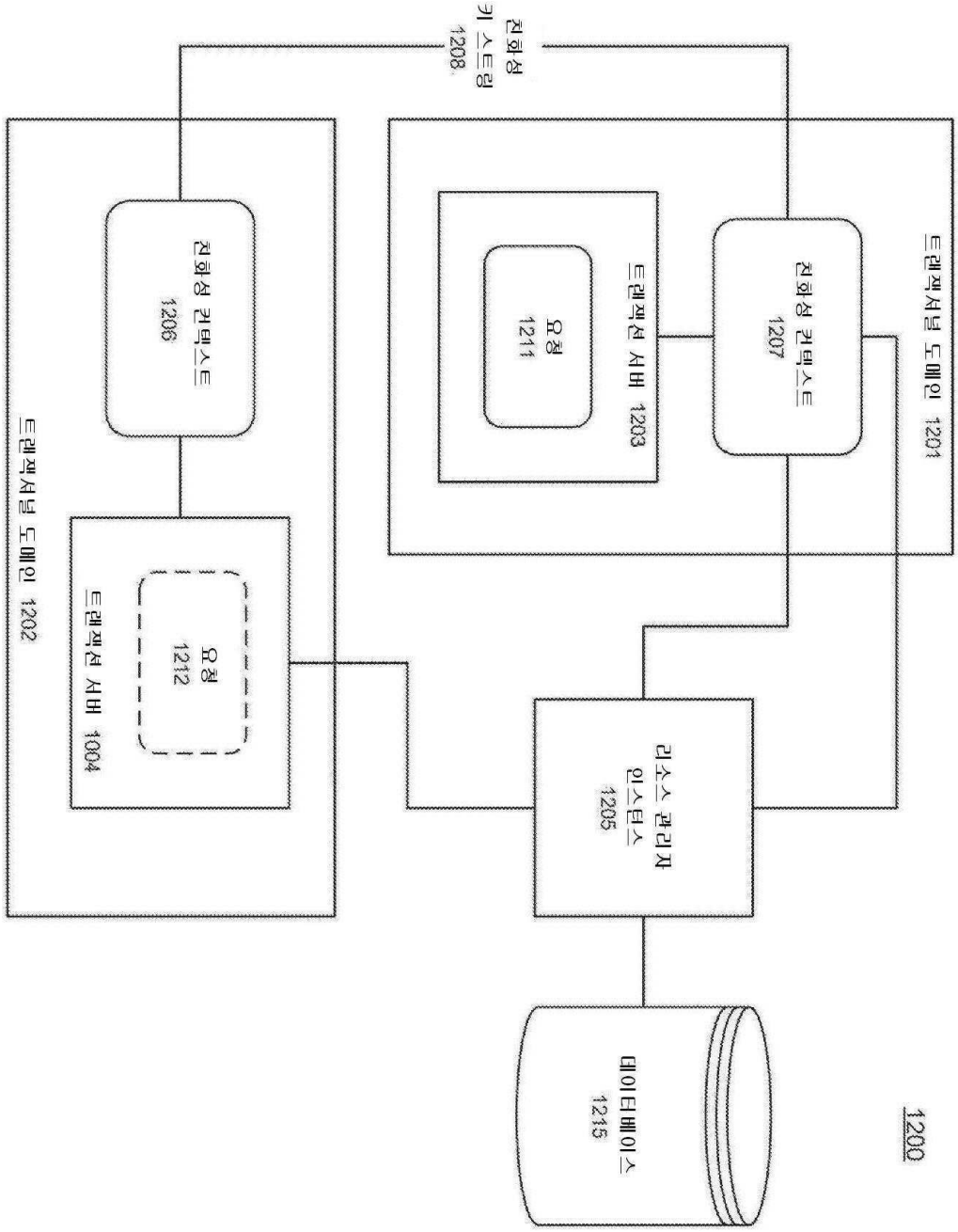
도면10


1000
트랜잭셔널 시스템 1001
공유 메모리 1002
친화성 컨텍스트
1004
메시지 큐 1005
메시지
1006
친화성 컨텍스트
1014
트랜잭션 컨텍스트 1007
체크 포인트
1008
트랜잭셔널 서버 1003

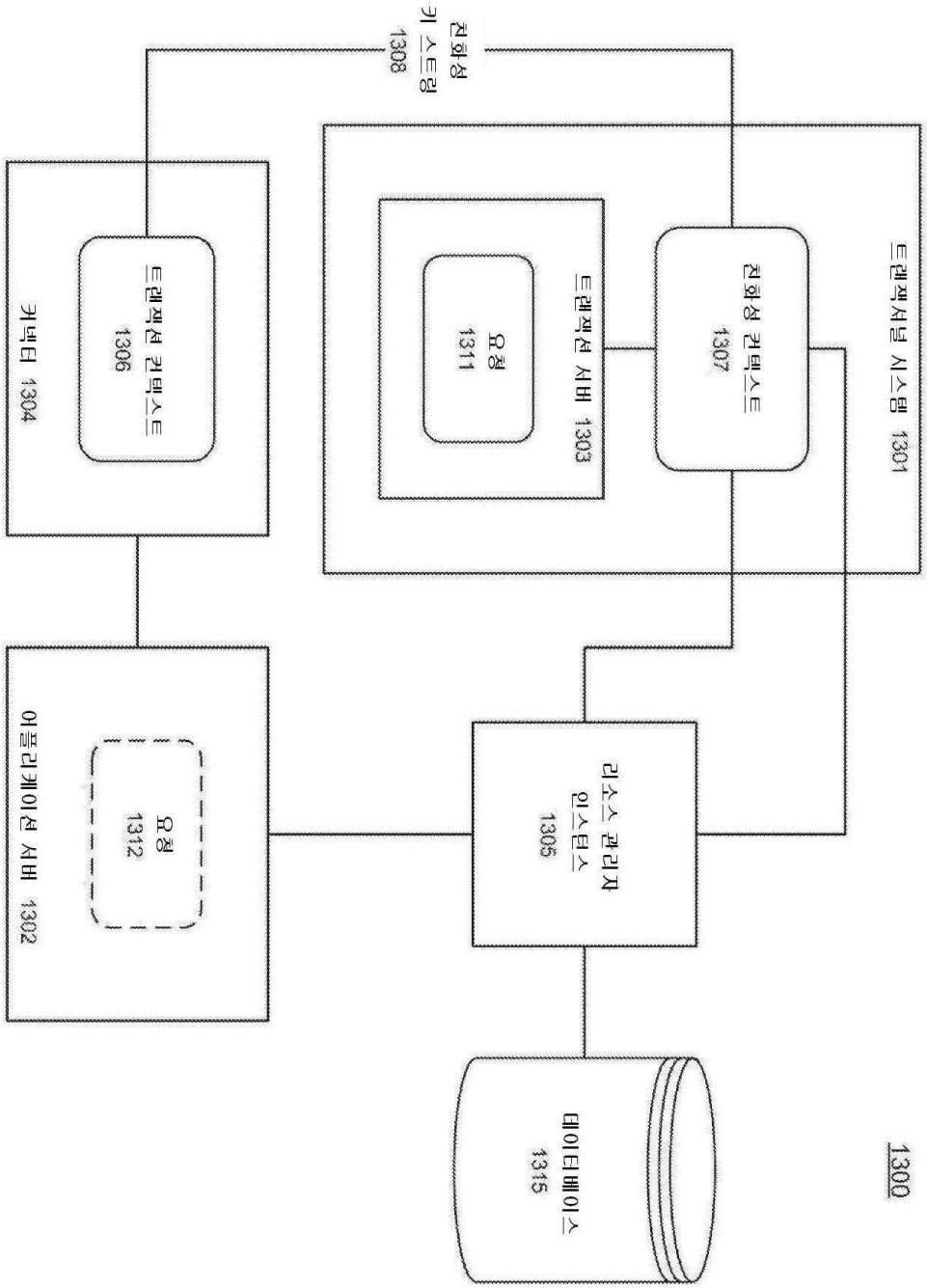
도면11


1100
트랜잭셔널 시스템 1101
컨텍스트 1105
트랜잭션 서버 1103
요청 1111
요청 1112
리소스 관리자 인스턴스 1102
데이터베이스 1104
클라이언트 컨텍스트 1110

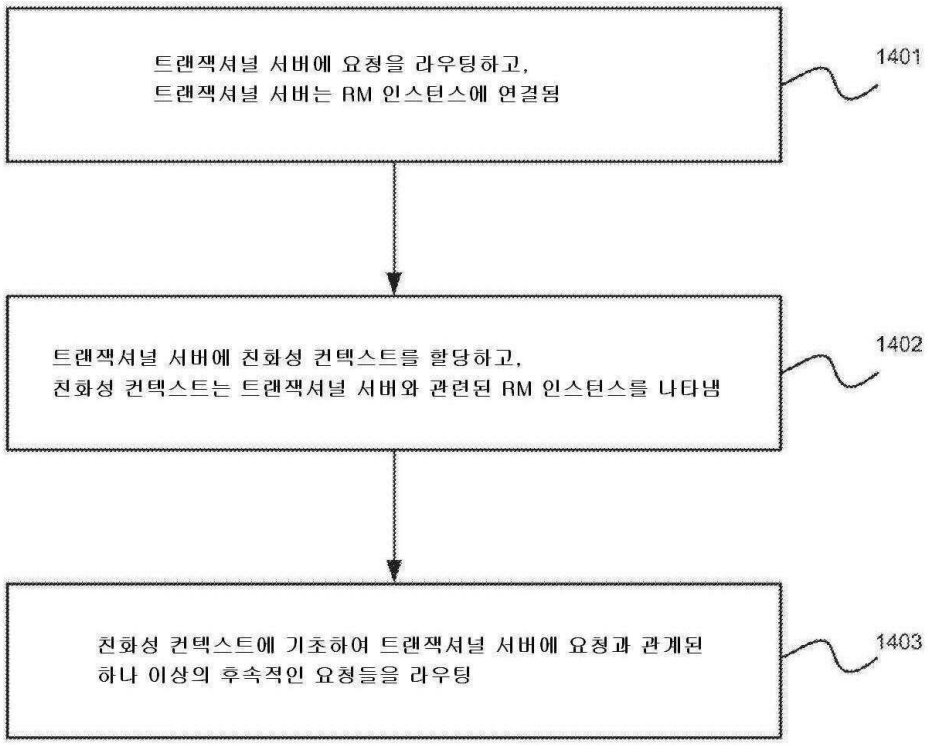
도면12


트랜잭셔널 도메인 1201
전화성 컨텍스트 1207
트랜잭션 서버 1203
요청 1211
전화성 키 스트링 1208
트랜잭셔널 도메인 1202
전화성 컨텍스트 1206
요청 1212
트랜잭션 서버 1004
리소스 관리자 인스턴스 1205
데이터베이스 1215
1200

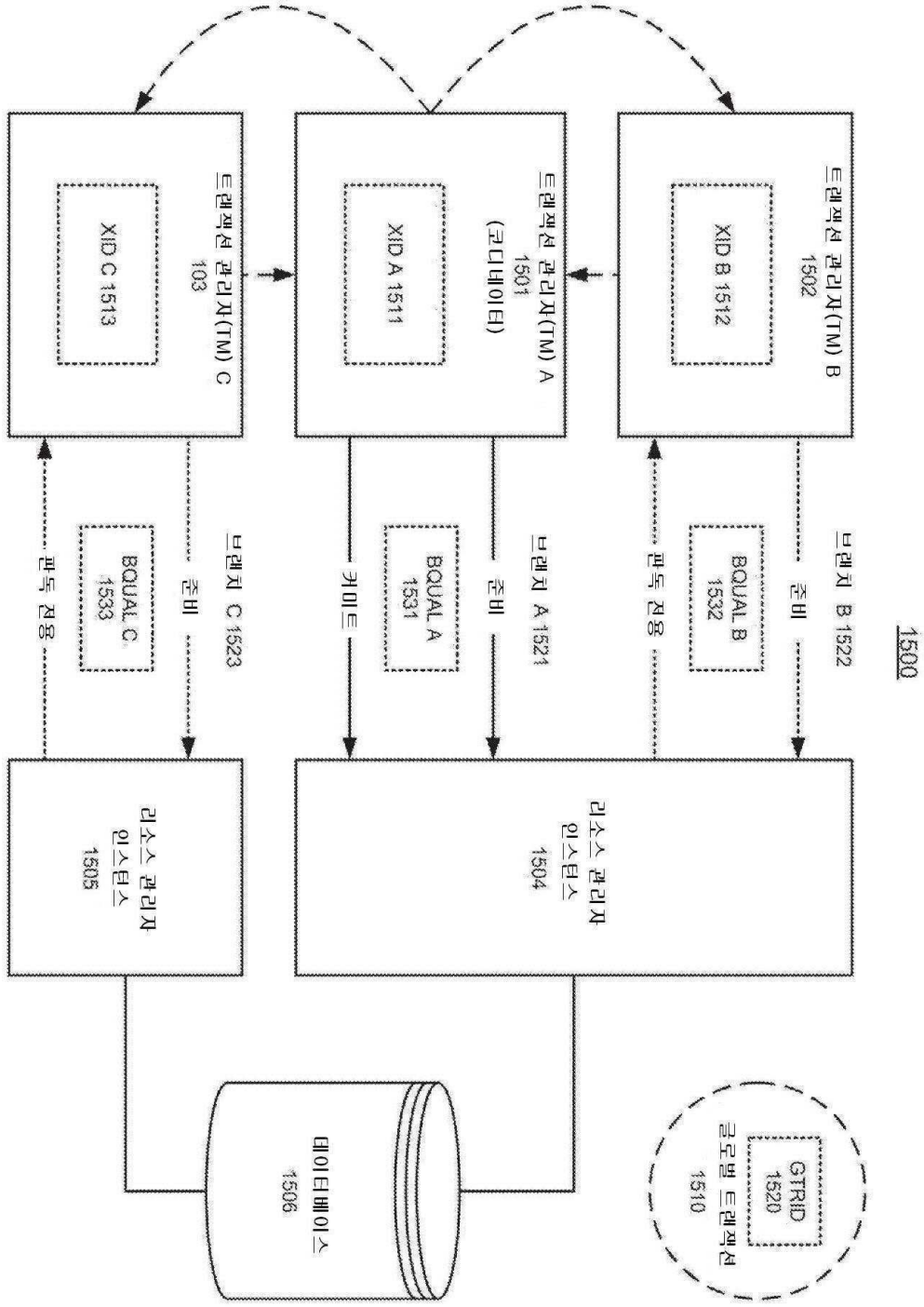
도면13


1300
트랜잭셔널 시스템 1301
천화성 컨텍스트 1307
트랜잭션 서버 1303
요청 1311
천화성 키 스트링 1308
커넥터 1304
트랜잭션 컨텍스트 1306
리소스 관리자 인스턴스 1305
어플리케이션 서버 1302
요청 1312
데이터베이스 1315

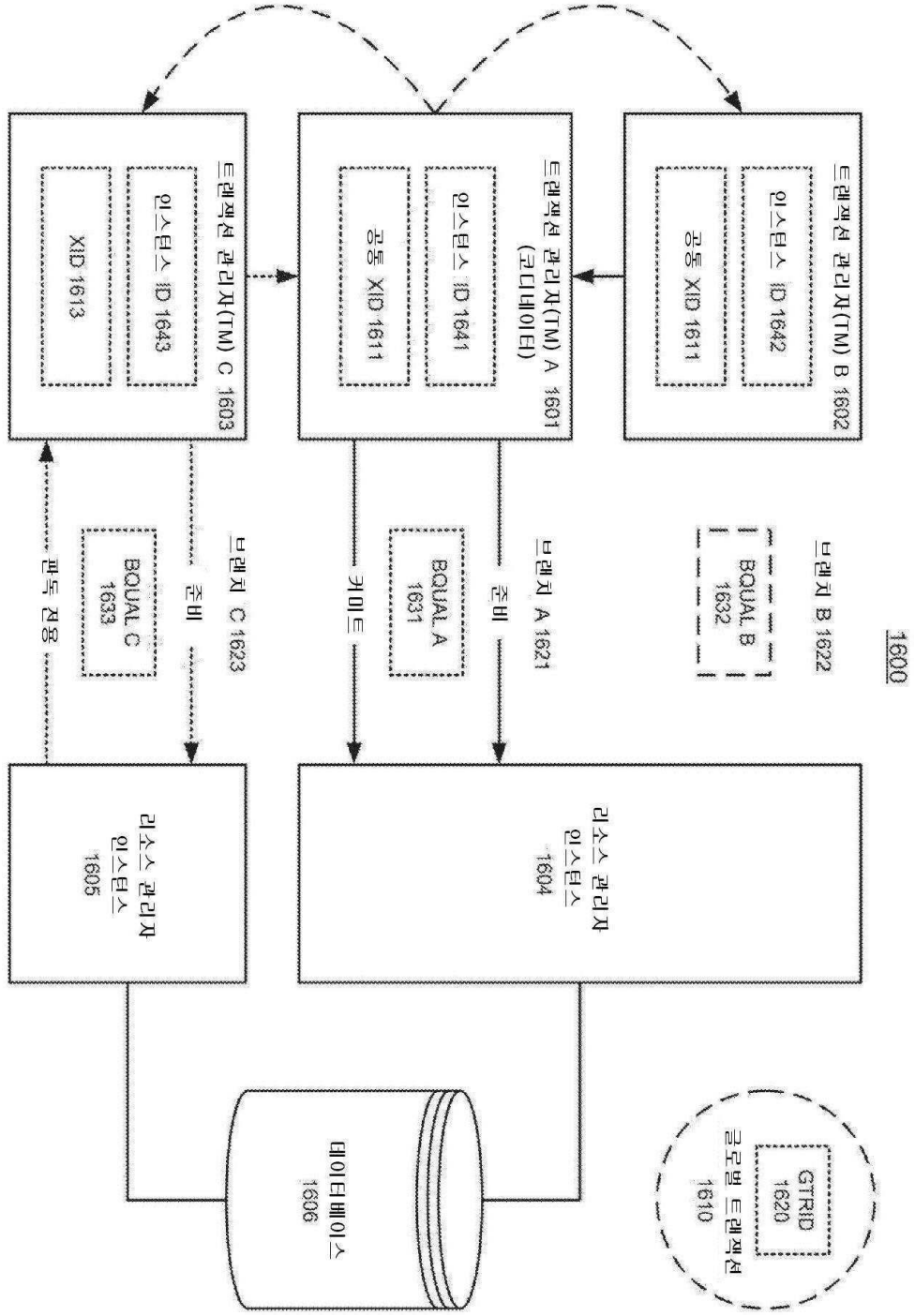
도면14


트랜잭셔널 서버에 요청을 라우팅하고,
트랜잭셔널 서버는 RM 인스턴스에 연결됨
1401
트랜잭셔널 서버에 친화성 컨텍스트를 할당하고,
친화성 컨텍스트는 트랜잭셔널 서버와 관련된 RM 인스턴스를 나타냄
1402
친화성 컨텍스트에 기초하여 트랜잭셔널 서버에 요청과 관계된
하나 이상의 후속적인 요청들을 라우팅
1403

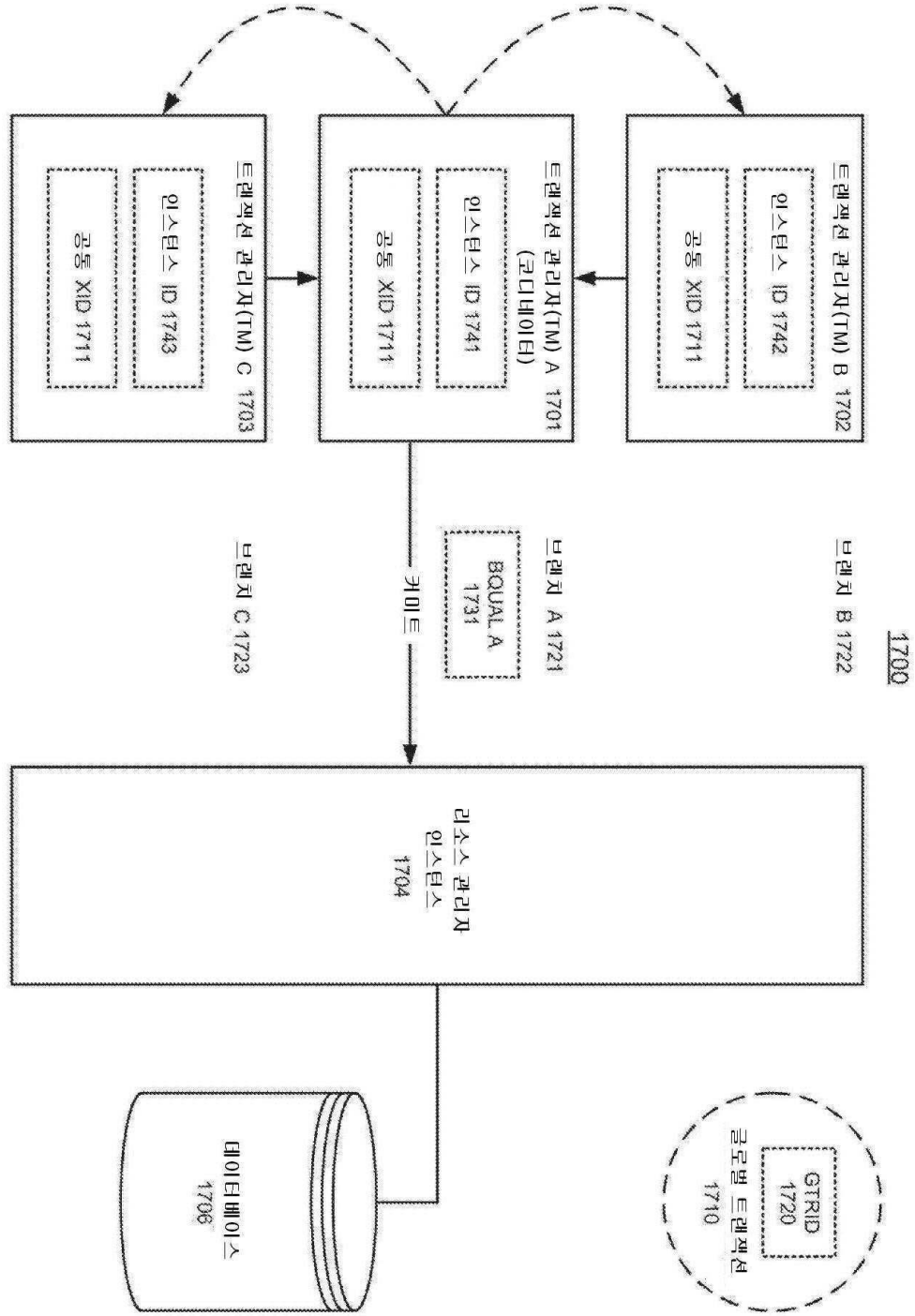
도면15


1500
트랜잭션 관리자(TM) B
1502
XID B 1512
트랜잭션 관리자(TM) A
1501
(코디네이터)
XID A 1511
트랜잭션 관리자(TM) C
103
XID C 1513
브랜치 B 1522
준비
BQUAL B
1532
편독 전용
브랜치 A 1521
준비
BQUAL A
1531
커미트
브랜치 C 1523
준비
BQUAL C
1533
편독 전용
리소스 관리자
인스턴스
1504
리소스 관리자
인스턴스
1505
데이터베이스
1506
글로벌 트랜잭션
1510
GTRID
1520

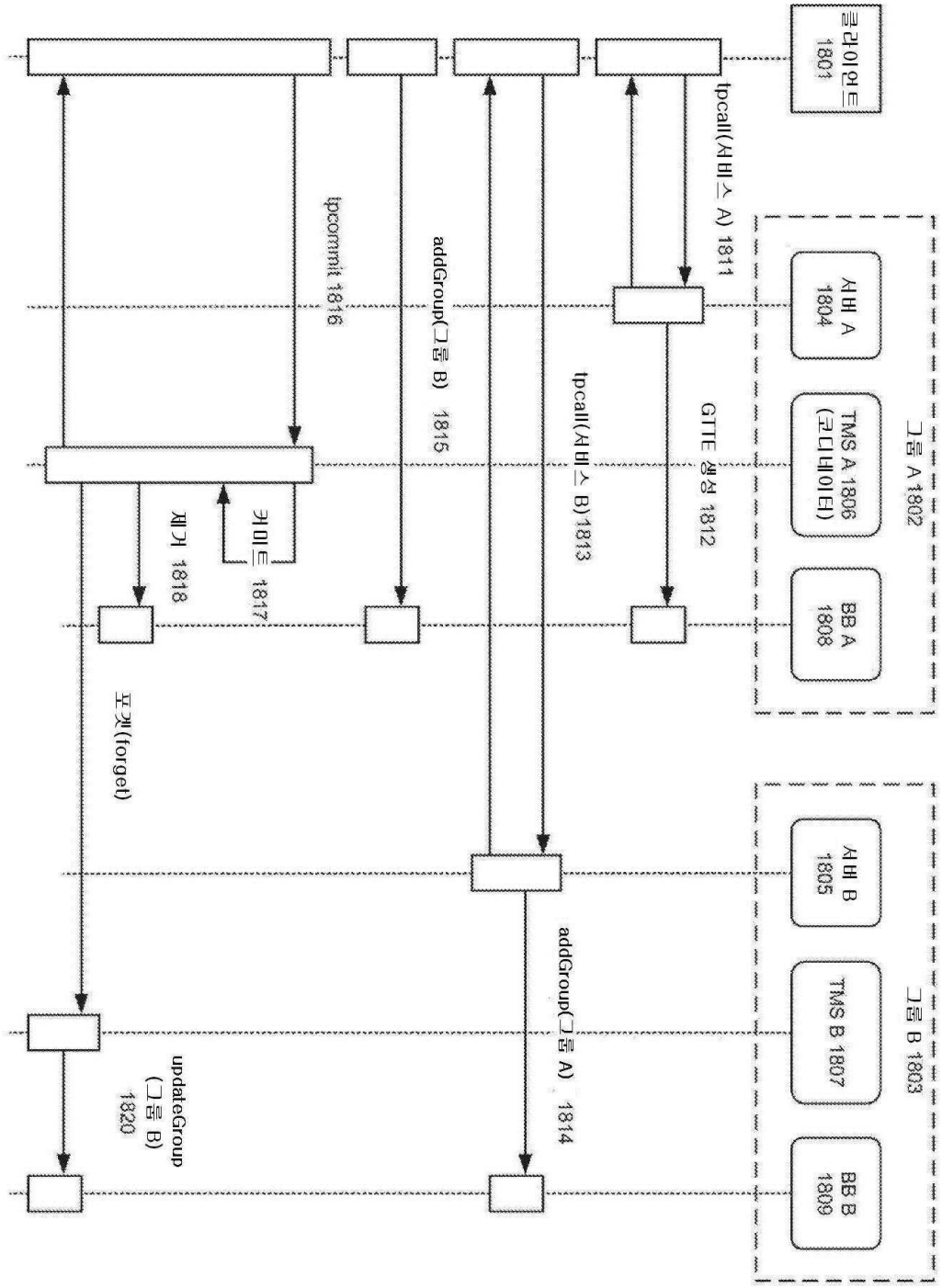
도면16


1600
트랜잭션 관리자(TM) B 1602
인스턴스 ID 1642
공통 XID 1611
트랜잭션 관리자(TM) A 1601
(코디네이터)
인스턴스 ID 1641
공통 XID 1611
트랜잭션 관리자(TM) C 1603
인스턴스 ID 1643
XID 1613
브랜치 B 1622
BQUAL B
1632
브랜치 A 1621
준비
BQUAL A
1631
커미트
브랜치 C 1623
준비
BQUAL C
1633
판독 전용
리소스 관리자
인스턴스
1604
리소스 관리자
인스턴스
1605
글로벌 트랜잭션
1610
GTRID
1620
데이터베이스
1606

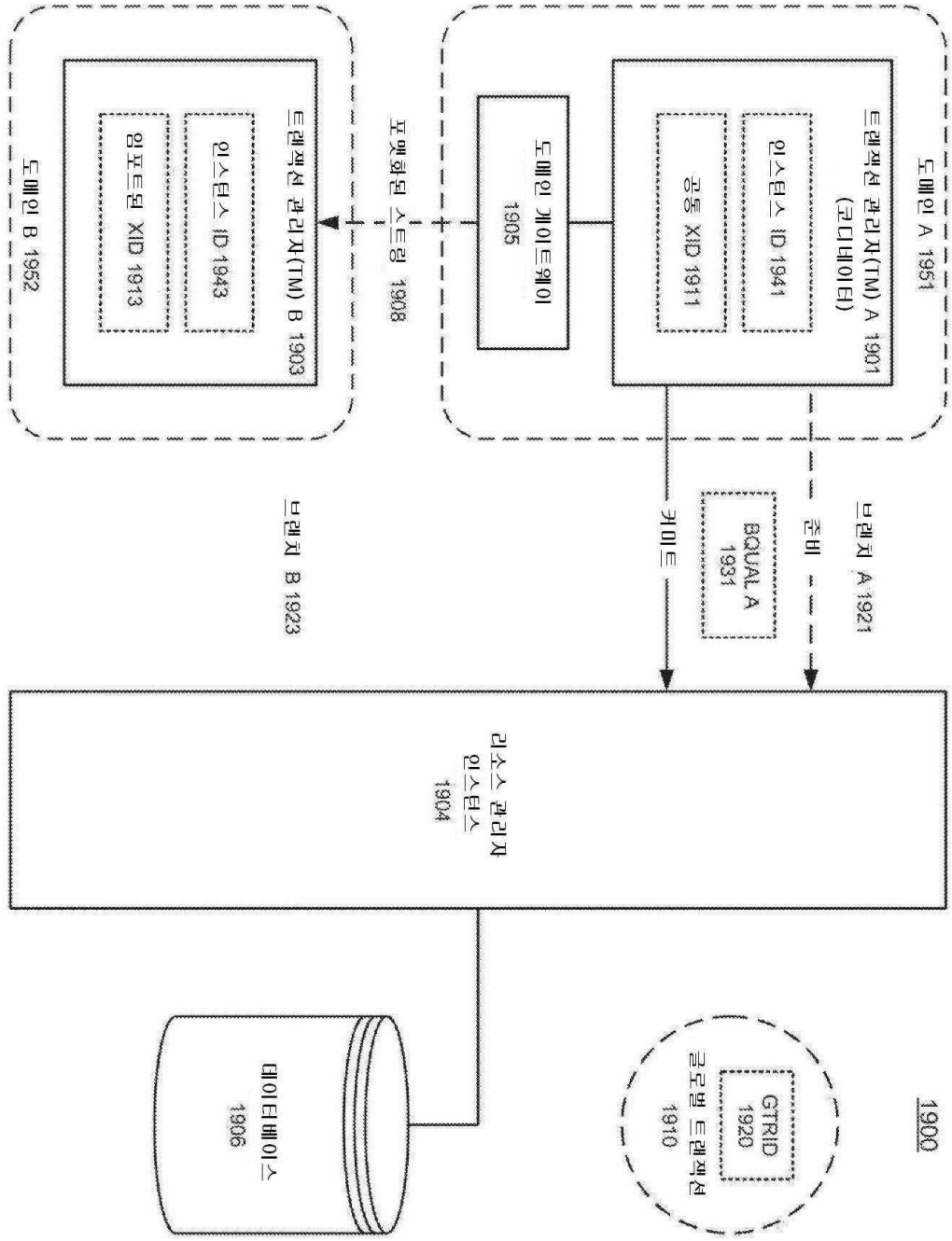
도면17


트랜잭션 관리자(TM) B 1702
인스턴스 ID 1742
공동 XID 1711
트랜잭션 관리자(TM) A 1701
(코디네이터)
인스턴스 ID 1741
공동 XID 1711
트랜잭션 관리자(TM) C 1703
인스턴스 ID 1743
공동 XID 1711
브랜치 B 1722
브랜치 A 1721
BQUAL A
1731
커미트
브랜치 C 1723
1700
리소스 관리자
인스턴스
1704
데이터베이스
1706
글로벌 트랜잭션
1710
GTRID
1720

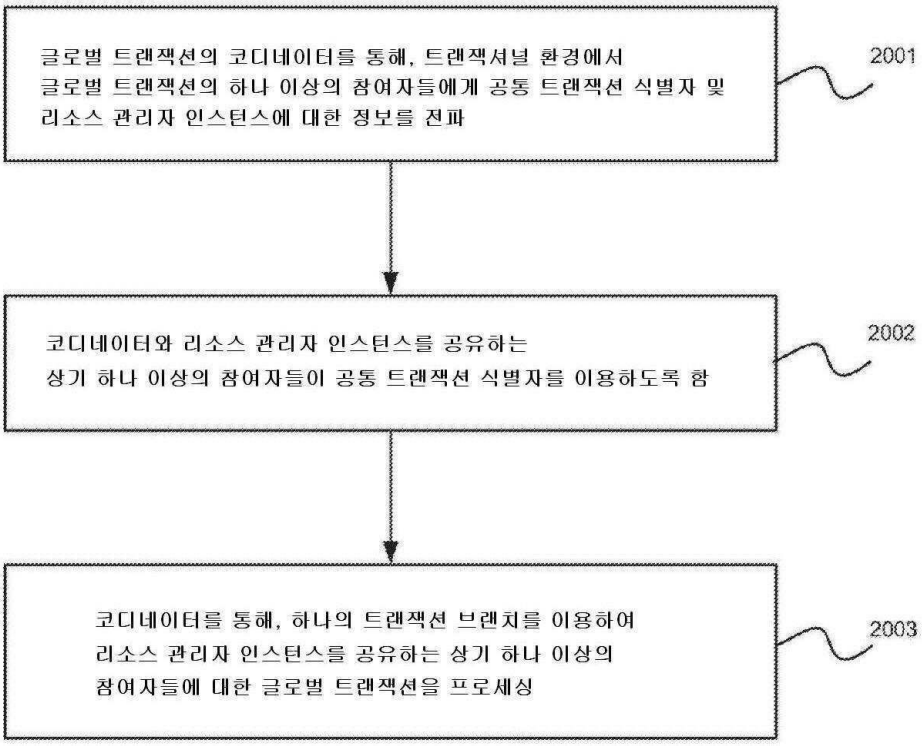
도면18


클라이언트 1801
그룹 A 1802
서버 A 1804
TMS A 1806 (코디네이터)
BB A 1808
그룹 B 1803
서버 B 1805
TMS B 1807
BB B 1809
tpcall(서비스 A) 1811
GTTE 생성 1812
tpcall(서비스 B) 1813
addGroup(그룹 A) 1814
addGroup(그룹 B) 1815
tpcommit 1816
커미트 1817
제거 1818
포겟(forget)
updateGroup (그룹 B) 1820

도면19


1900
도메인 A 1951
트랜잭션 관리자(TM) A 1901
(코디네이터)
인스턴스 ID 1941
공동 XID 1911
도메인 게이트웨이
1905
포맷팅된 스트림 1908
도메인 B 1952
트랜잭션 관리자(TM) B 1903
인스턴스 ID 1943
임포트된 XID 1913
브랜치 B 1923
키미트
준비
BQUAL A
1931
브랜치 A 1921
리소스 관리자
인스턴스
1904
GTRID
1920
글로벌 트랜잭션
1910
데이터베이스
1906

도면20


글로벌 트랜잭션의 코디네이터를 통해, 트랜잭셔널 환경에서
글로벌 트랜잭션의 하나 이상의 참여자들에게 공통 트랜잭션 식별자 및
리소스 관리자 인스턴스에 대한 정보를 전파
2001
코디네이터와 리소스 관리자 인스턴스를 공유하는
상기 하나 이상의 참여자들이 공통 트랜잭션 식별자를 이용하도록 함
2002
코디네이터를 통해, 하나의 트랜잭션 브랜치를 이용하여
리소스 관리자 인스턴스를 공유하는 상기 하나 이상의
참여자들에 대한 글로벌 트랜잭션을 프로세싱
2003