JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 2, pp. 125-134, February 2022
https://doi.org/10.9708/jksci.2022.27.02.125

Extending the BR2K technique to enhance the robustness of blockchain application services

Min-Ho Kwon*, Myung-Joon Lee*

*Student, Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea
*Professor, Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea

[Abstract]

In this paper, we propose an extension method of the BR2K technique for enhancing the robustness of blockchain application services. The BR2K (Blockchain application, Replication & Recovery technique using Kubernetes) technique was recently developed to support the robustness of blockchain services through service replication and rapid restart. The proposed technique extends the existing BR2K technique to provide a state version, which is meta-information about the service state, and a method for safely managing it, and use the state version to determine the timing for service state recovery. Also, the technique provides systematic service state backup for service recovery and joining of new service nodes by utilizing the version information and the service registry which acts as a service recovery center in the BR2K technique. Based on this, it is possible to support new service nodes to join the replication service with consistency. As a result, new service nodes can be quickly added to the BR2K service in operation, enhancing the robustness of the BR2K service. In addition, the extended method is applied to the pilot blockchain application service and tested in a Kubernetes environment composed of virtual machines to confirm the validity of service replication consistency and rapid service recovery in the event of node failures.

▸ **Key words**: Blockchain Service, Service Robustness, Service Replication, Service Recovery, BR2K

[요 약]

본 논문에서는 블록체인 응용 서비스의 견고성을 제고하기 위한 BR2K 기법의 확장 방법을 제안한다. BR2K(Blockchain application, Replication & Recovery technique using Kubernetes) 기법은 서비스 복제와 신속 재가동을 통해 블록체인 서비스의 견고성을 지원하기 위하여 최근에 개발된 기법이다. 제안하는 기법은 기존 BR2K 기법을 확장하여 서비스 상태에 대한 메타 정보인 상태 버전 및 이를 안전하게 관리하기 위한 방법을 제공하고, 이를 이용하여 서비스 상태 복구에 대한 시점을 결정한다. 또한 이 기법은 BR2K 기법에서 서비스 복구 센터 역할을 수행하는 서비스 레지스트리와 상태 버전 정보를 활용하여 서비스 복구와 새로운 서비스 노드의 합류를 위한 체계적인 서비스 상태 백업을 제공한다. 이를 바탕으로 새로운 서비스 노드가 일관성을 가지며 새로운 복제 서비스에 합류하도록 지원할 수 있으며, 그 결과 새로운 서비스 노드가 운영 중인 BR2K 서비스에 신속하게 추가될 수 있어 서비스의 견고성이 제고된다. 또한 시범적인 블록체인 응용 서비스에 확장된 기법을 적용하고 가상 머신으로 구성된 쿠버네티스 환경에서 이를 테스트하여, 장애 상황 속에서 서비스 복제의 일관성과 신속한 서비스 복구에 대한 유효성을 확인한다.

▸ **주제어**: 블록체인 서비스, 서비스 견고성, 서비스 복제, 서비스 복구, BR2K

• First Author: Min-Ho Kwon, Corresponding Author: Myung-Joon Lee
*Min-Ho Kwon (alsgh458@gmail.com), Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan
*Myung-Joon Lee (mjlee@ulsan.ac.kr), Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan
• Received: 2022. 01. 04, Revised: 2022. 01. 20, Accepted: 2022. 01. 20.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

블록체인 기술은 주식 거래, 의약품 관리 및 추적, 디지털 자산 거래 등과 같은 다양한 분야에서 활용되고 있다.[1-3] 또한, 블록체인 기반의 암호 신원 인증[4], 블록체인의 합의 공격 분석과[5] 같이 블록체인의 활용도 및 견고성 등을 보완하는 연구들도 다양한 형태로 진행되고 있다. BR2K 기법은[6] 블록체인과 관련된 연구로 네트워크 분할, 실행 노드 중지와 같은 상황에서 블록체인 응용 서비스의 견고성 및 신속 재가동을 지원하는 기법이다. BR2K는 장애 상황에서도 상태에 대한 일관성을 보장하는 자체적인 서비스 복제를 통해 서비스의 연속성을 지원할 수 있다. 그러나 실행 노드 중지와 같은 상황에서 일관성을 보장하기 위한 각 사용자 요청에 대한 BR2K 기법의 롤백 작업은 다양한 분야의 블록체인 응용 서비스에 적용할 때, 롤백 작업의 수동 설정으로 인한 번거로움이 존재하고 잘못된 설정으로 인하여 여러 예외적인 문제도 발생할 수 있다.

본 논문은 기존의 BR2K 기법의 견고성 제고를 위한 확장된 BR2K 기법에 대하여 설명한다. 확장된 기법에서는 블록체인 응용 서비스의 상태에 대한 메타 정보인 상태 버전과 BR2K 기법에서 재난 복구 센터 역할을 수행하는 서비스 레지스트리를 통해 체계적인 서비스 상태 백업을 지원한다. 또한, 이 확장된 기법은 백업된 상태를 이용하여 BR2K 기법에 의하여 복제된 블록체인 응용 서비스의 기민한 동적 확장을 지원하고, 서비스 복제 과정에서 발생할 수 있는 실패 상황을 보다 신속하고 효율적으로 극복하기 위한 서비스 상태 복구 작업을 지원한다.

본 논문의 1장 및 2장에서는 서론과 배경지식이 소개된다. 그리고 3장에서는 기존의 BR2K 기법에 대하여 간략하게 소개하고 4장에서는 체계적인 서비스 상태 백업, 서비스 상태 복구 등을 지원하는 확장된 BR2K 기법을 기술한다. 마지막 6장에서는 확장된 기법의 서비스 상태 복구에 대한 유효성을 확인하기 위한 테스트를 소개하고 7장에서는 본 논문의 결론에 대하여 기술한다.

II. Background Knowledge

1. Ethereum and Smart contract

이더리움은[7] 2세대 블록체인으로, 트랜잭션 기능뿐만 아니라 제 3자의 개입 없이 투명한 거래를 지원하도록 하는 스마트 컨트랙트를 제공하여 여론 조사, 각종 금융 거래 그리고 전자투표 등 다양한 분야에서 응용할 수 있는 블록체인 플랫폼이다. 또한, 이더리움은 p2p 네트워크, 암호 해시 등 기본적인 기능을 제공하며 생성된 트랜잭션과 트랜잭션에 관련된 데이터들이 모여 있는 블록의 유효성을 검증하는 합의 엔진 등이 포함되어있다. 최근에는 낮은 처리 속도와 트랜잭션의 높은 비용을 벗어나기 위하여, 기존의 해시 알고리즘 기반의 작업증명(PoW) 방식에서 벗어나 지분증명(PoS) 방식의 이더리움 2.0을 출시하여 다양한 프로덕션 환경에서 실용적으로 응용될 수 있는 주요한 블록체인이 되었다.

스마트 컨트랙트란[8] 중개자에 의한 기존 계약 방식에서 벗어나 이더리움 블록체인 상에서 조건에 따라 자동으로 계약 수행을 지원하는 기능이다. 스마트 컨트랙트는 이더리움 블록체인의 가상 머신 기반에서 실행되어 무결성 및 조작 방지에 대한 보장이 가능한 실행 코드이다. 계약을 원하는 사용자는 스마트 컨트랙트를 이용하기 위하여 트랜잭션을 보내고 스마트 컨트랙트에 명시된 내용에 따라 기능을 제공받을 수 있다. 또한, 스마트 컨트랙트는 기능을 수행하기 전후로 특정한 조건을 검증할 수 있도록 하는 함수 변경자, 트랜잭션에 로그 형태로 저장되어 클라이언트에게 메시지를 전달할 수 있는 이벤트 기능도 제공하여 다양한 형태로 사용될 수 있다.

2. ETCD

ETCD는[9-10] 서비스 설정 정보, 서비스 검색 그리고 분산 시스템 및 클러스터의 스케줄러 조정을 위한 오픈소스 기반의 키-값 분산 저장소이다. 이 저장소는 강한 일관성을 보장하는 분산 컨센서스 알고리즘인 Raft[11] 프로토콜 기반으로 구현되어 데이터를 신뢰성 있는 형태로 복제하여 관리한다. 또한, ETCD는 프로덕션 환경에서 컨테이너 오케스트레이션 도구인 쿠버네티스[14], 우버의 메트릭 플랫폼인 M3와[15] 같은 복잡한 응용 서비스에서도 사용되어 실용성이 검증되었다. ETCD는 저장된 데이터 변화에 대한 이벤트 API, 사용자 인증서를 통한 자동 TLS 그리고 분산 환경에서 발생할 수 있는 실패에 대비한 정책 및 복구 기능 등을 지원하여 데이터를 안전하고 체계적인 형태로 관리할 수 있다.

3. BR2K

다양한 분야에서 블록체인 응용 서비스가 개발되고 있지만, 아직 블록체인 응용서비스의 견고성이나 서비스 실패 시의 복구에 관한 체계적인 연구는 거의 이루어지고 있지 않은 실정이다. 클라우드 기반으로 구성된 블록체인 서비스는 개별 노드보다는 클라우드의 특성상 가용성과 견

고성을 어느 정도 지원하지만, 개별 기업의 서비스에 대한 의존은 2021년 12월 아마존 클라우드 서비스의 대규모 접속장애와 같은 위협에 근본적으로 노출되며 응용 서비스 관련 데이터 유출의 한계가 존재한다. BR2K 기법은 이를 극복하기 위하여 블록체인 응용 서비스의 견고성을 보장하는 기법이다. 이는 자체적인 동기화 방법을 통한 서비스 복제와 서비스 실패에 대한 신속 재가동 방법을 통해 블록체인 응용 서비스의 견고성을 지원한다. BR2K는 장애 상황에서도 블록체인 응용 서비스의 연속성을 보장하기 위하여 ETCD 분산 스토리지를(\$2.2) 이용한 사용자 요청의 복제, 사용자 요청의 체계적인 처리 절차 그리고 쿠버네티스를 이용한 배포 방법 등을 통해 서비스를 복제한다. 또한, 이 기법은 이더리움 블록체인의 스마트 컨트랙트로(\$2.1) 개발된 서비스 레지스트리를 통해 서비스를 이용하는 사용자에게 최신 서비스 접속 정보를 지속적으로 제공할 수 있으며, 최악의 경우를 대비한 서비스 복구 정보를 안전하게 관리할 수 있다. 그리고 BR2K 기법은 백업된 복구 정보를 이용하여 블록체인 응용 서비스가 중지되는 상황에서 서비스를 신속 재가동할 수 있는 체계적인 절차도 지원할 수 있다.

III. Replication in BR2K

본 장에서는 BR2K 기법의 서비스 복제 방법에 대하여 간략하게 설명한다. 또한, 이 기법이 적용된 블록체인 응용 서비스가 복제 과정에서 발생할 수 있는 실행 노드의 중지와 같은 상황에서 일관성을 어떠한 방식으로 보장하는지에 대하여 설명한다.

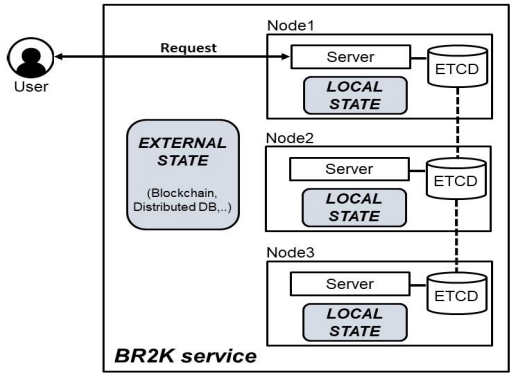

Fig. 1. BR2K service

BR2K 기법이 적용되어 복제된 블록체인 응용 서비스를 BR2K 서비스라고 부르며, 그림 1과 같이 분산 환경을 구성하는 각 노드에 블록체인 응용 서비스의 서버와 ETCD가(\$2.2) 함께 중복되어 실행되는 구조를 가진다. BR2K 서비스를 구성하는 각 서버는 사용자에게 서비스를 제공하는 리더 역할과 실시간 백업 서버인 팔로워 역할 중 하나의 역할을 가진다. 이때, BR2K 서비스에서 리더 역할을 수행하여 사용자에게 서비스를 제공하는 서버는 오직 하나이다. 만약, 서비스를 제공하는 리더가 중지되면, BR2K 서비스는 팔로워 중에서 새로운 리더를 선출하여 서비스를 재개한다.

BR2K 서비스는 서비스가 운영되는 동안 유지해야 하는 서비스 운영 정보들을 가지며, 이 정보들은 사용자 요청으로 업데이트된다. BR2K 서비스의 서비스 운영 정보는 외부 상태(External state)와 로컬 상태(Local state)로 구분된다. 외부 상태는 모든 서버가 공유하는 서비스 운영 정보들이며 각 서버의 실행 노드 외부에 있는 블록체인이나 외부 분산 데이터베이스에서 유지된다. 반면에, 로컬 상태는 각 서버가 자신의 실행 노드에서 독립적으로 유지하는 서비스 운영 정보이다. 이 로컬 상태들은 응용 서비스의 일관성을 보장하기 위하여, 실행 노드 중지와 같은 장애 상황에서도 올바르게 동기화되어야 한다. BR2K 서비스에서 서로 독립적인 로컬 상태들을 동기화하는 간략한 절차는 다음과 같다.

[(l) in fig. 2] BR2K 서비스에서 서비스 제공자 역할을 수행하는 리더 서버가 사용자 요청을 받으면 자신과 연결된 ETCD에 이 요청을 저장하여 각 서버의 실행 노드로 복제한다.

[(2) in fig. 2] 복제된 사용자 요청은 리더 서버가 먼저 처리하여 자신의 실행 노드에 대한 로컬 상태를 변경시키거나 외부 상태를 변경시키고 최신 서비스 상태를 업데이트한다. 그리고 BR2K 서비스의 최신 상태가 변경되었음을 실시간 백업 서버 역할을 하는 팔로워들에게 알리기 위하여, 리더 서버는 ETCD에 저장되어 BR2K 서비스의 최신 서비스 상태를 나타내는 SSI를 업데이트한다.

[(3) in fig. 2] 각 팔로워는 자신과 연결된 ETCD를 통해 SSI의 업데이트 여부를 지속적으로 확인하여 서비스 상태가 변경됨을 감지하고 BR2K 서비스의 최신 서비스 상태를 따라가는 작업을 수행한다. 최신 상태를 따라가는 작업은 각 팔로워가 처리하지 못한 사용자 요청들에 대하여 순차적으로 처리하여 자신의 로컬 상태를 변경하여 이루어진다.

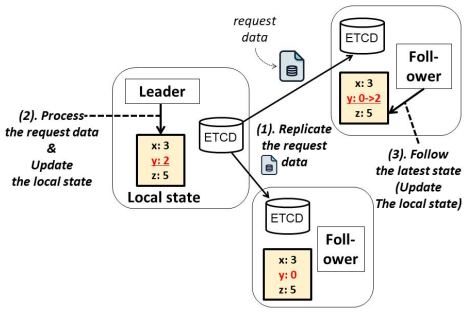

Fig. 2. Synchronization process for the local state

이와 같은 동기화 과정에서, 실행 노드의 중지와 같은 상황에서 의하여 BR2K 서비스의 서버는 사용자 요청을 처리하는 중에 중지될 수 있다. 이러한 상황이 발생한 서비스는 상태에 대한 일관성을 보장할 수 없을 수 있다. 이는 BR2K 기법에서 사용자 요청에 대한 처리 작업은 이 기법이 적용된 블록체인 응용 서비스가 제공하는 서비스에 따라 각기 달라져, 그림 3과 같이 정확하게 어떠한 시점에서 서비스 상태를 얼마나 업데이트하였는지 예측할 수 없기 때문이다. 이러한 상황을 극복하려면 크게 2가지 방법이 존재한다.

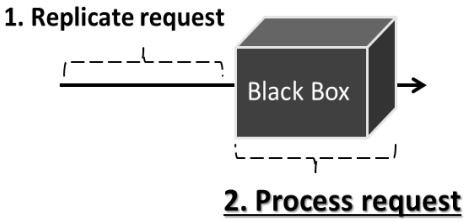

Fig. 3. The processing request in BR2K

첫 번째 방법은 사용자 요청의 처리 작업에 대하여 롤백 작업을 수행하여 이전 상태로 되돌리는 방법이다. BR2K 기법은 이와 같은 방법을 통해, 요청 처리 중에 중지된 서버가 발생하여도 서비스 상태에 대한 일관성을 보장하도록 한다. 이 롤백 작업은 사용자 요청을 처리하기 전에 이 요청이 변경하는 상태를 임시로 로컬 환경에 저장하고 이를 이용하여 사용자 요청에 대한 처리 작업 전으로 되돌리는 작업이다. 이 롤백 작업에 대한 정의는 서비스 내에 있는 각 사용자 요청마다 서비스 개발자에 의하여 개별적으로 설정된다. 그러나 각 사용자 요청에 대하여 롤백 작업을 정의하는 것은 BR2K 기법을 다양한 분야의 블록체인 서비스에 실제로 적용할 때 일반적이지 못하여 실용성이 떨어진다. 이는 번거롭게 모든 사용자 요청에 대한 롤백 작업에 대하여 설정해야 하고, 의도치 않게 잘못된 설정을 하게 되면 서비스에 대한 일관성을 보장할 수 없는 상황이 발생할 수 있다. 또한, 사용자 요청의 처리 작업마다 관련된 기존 상태의 저장 절차가 필요하여 상당한 오버헤드가 발생한다.

사용자 요청을 처리하는 중에 서버가 중지되는 상황을 극복하기 위한 두 번째 방법은 특정 시점의 상태를 백업하고 이를 이용하여 서비스 상태를 복구하는 방법이 있다. 이 방법은 일정 주기나 조건에 따라 상태 백업 작업을 수행하여야 하지만, 롤백 작업 정의와 같은 작업을 서비스 개발자가 직접 할 필요 없어, 다양한 블록체인 응용 서비스에 일반적으로 적용될 수 있다. 이뿐만 아니라, 서비스 상태 복구를 수행하기 위하여 백업 된 상태는 기존의 BR2K 서비스가 오랫동안 실행될 때 발생할 수 있는 새로운 서버의 신속하지 못한 합류와 실행 노드의 용량 부족과 같은 예외적인 문제점들을 개선할 수 있다.

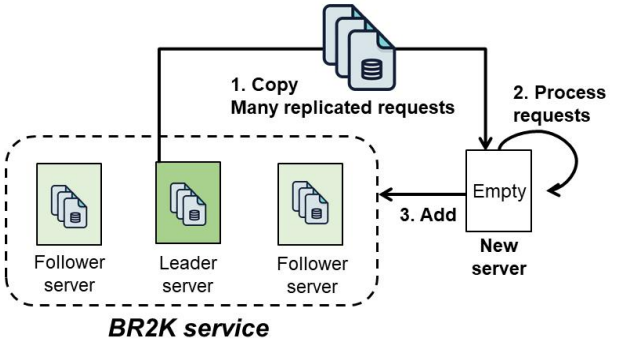

Fig. 4. Add new server in BR2K service

그림 4와 같이 지속된 기존의 BR2K 서비스에 한 서버가 복제 멤버로 새롭게 합류하려면, BR2K 서비스가 처리한 모든 사용자 요청을 새롭게 합류할 서버의 실행 노드로 복제하여야 하고 해당 서버가 많은 사용자 요청을 처리하여 BR2K 서비스의 최신 상태로 동기화하여야 한다. 이와 같은 이유로, 기존의 BR2K 기법은 서비스가 지속할수록 새로운 복제 노드를 신속하게 추가하지 못한다. 또한, 서비스가 오랫동안 지속한 BR2K 서비스는 각 서버의 실행 노드에 수많은 사용자 요청이 복제하고 이를 실행 노드에 저장하여 실행 노드의 용량 초과 문제가 발생하여 서비스 중지로 이어질 수 있다.

IV. Extension of BR2K

본 장은 기존의 BR2K 서비스의 견고성을 제고하기 위한 확장된 BR2K 기법의 상태 버전과 이를 활용한 상태 백

업 과정에 대하여 설명한다. 또한, 백업된 상태를 이용한 새로운 서버의 신속한 합류와 서비스 상태 복구를 지원하는 방법에 관하여 기술한다.

1. State version

확장된 BR2K 기법에서는 서비스 상태에 대한 백업 작업이 몇 번 수행하였는지를 나타내는 상태 버전이 존재한다. 이 상태 버전에는 그림 5와 같이 BR2K 서비스의 상태 버전을 나타내는 SV(State version)와 각 서버가 관리하는 상태 버전인 SSV(Server state version)이 있다. SSV는 서버의 실행 노드에 있는 로컬 상태에($3) 대한 버전을 나타내며, 각 서버에 의하여 비동기적으로 버전이 올라간다. SV는 BR2K 서비스의 최신 상태에 대한 버전을 나타내며, 각 서버의 로컬 상태 버전 중에 가장 높은 버전으로 업데이트된다.

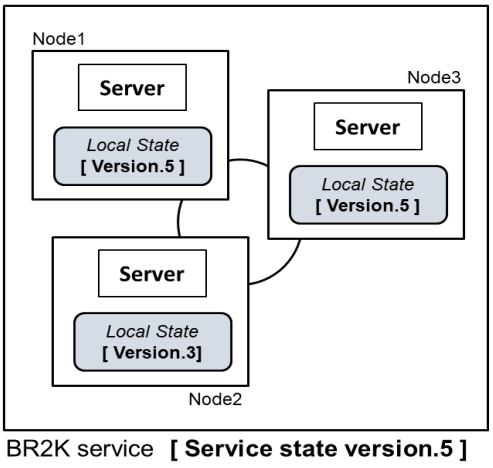

Fig. 5. State versions in BR2K service

모든 상태 버전들은 상태와 관련된 중요한 메타 값이므로, 확장된 BR2K 기법에서는 해당 버전 정보들을 ETCD를($2.2) 이용하여 BR2K 서비스의 각 실행 노드로 복제하여 관리한다. 특히, BR2K 서비스의 최신 상태 버전을 나타내는 중요한 정보인 SV는 이더리움 블록체인($2.1) 기반에서 서비스 재난 복구 센터 역할을 수행하는 서비스 레지스트리에서도($2.3) 저장하여 관리한다. 모든 상태 버전 정보는 그림 6과 같이 키-값 형태로 구성되며, BR2K 서비스가 처음 시작될 때 모든 버전의 값은 1로 시작하고 한 번의 백업 작업이 일어날 때마다 1씩 증가한다.

State versions stored in ETCD
(Data format is key-value)

SV(State version)	Version of the latest service state in BR2K service Key(String): "Service-State", Value(Number): version
SSV(Server state version)	Local state version on each server Key(String): "State"-(ID of the server), Value(Number): version

Fig. 6. Structure of state versions

2. State backup using state versions

BR2K 서비스에서는 서비스 상태 복구 작업, 새로운 서버의 신속한 합류, 서비스 복구 작업 그리고 실행 노드의 정리 작업을 진행하기 위하여, 특정 시점마다 서비스 상태에 대한 백업 작업이 필요하다. 확장된 BR2K 기법에서는 서비스 상태에 대한 백업 작업을 *State version up* 액션이라고 하며, 오직 리더 서버에 의하여 수행된다. 그림 7은 확장된 기법에서 상태 버전에 대한 히스토리의 예시를 나타낸다.

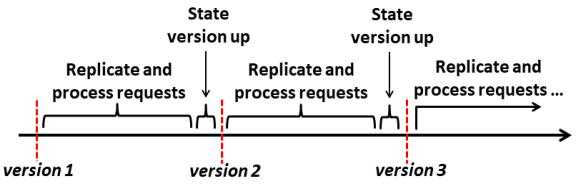

Fig. 7. History of state versions in extended BR2K

State version up 액션은 아래의 두 가지 조건 중 하나의 조건만 만족하면 시작된다. 첫 번째는 최대 복제 개수만큼 사용자 요청이 실행 노드로 복제될 때이다. 이때, 사용자 요청의 최대 복제 개수는 서비스 배포자가 각 서버가 실행되는 노드에 대한 용량을 고려하여 설정된다. 두 번째는 BR2K 서비스를 구성하는 모든 서버에서 중지된 서버가 과반수 일 때이다. 중지된 서버가 과반수인 BR2K 서비스는 일반적인 상황의 BR2K 서비스보다 정상적인 실행 노드의 개수가 적어 로컬 상태에 대한 소실 확률이 높다. 이러한 경우에는 신속히 서비스 상태에 대한 백업 작업이 이루어져야 한다. State version up 액션에 대한 시작 조건을 만족한 BR2K 서비스는 표 1과 같은 과정을 거쳐 서비스 상태에 대한 백업 작업을 수행한다.

[line 1-2 in table.1] State version up 액션을 시작한 리더는 사용자 요청을 더는 받지 않고 복제된 사용자 요청까지만 처리하여 서비스 상태를 업데이트한다. 그리고 리더는

ETCD에 저장되어 각 서버가 사용자 요청을 어디까지 처리하였는지를 나타내는 LPI와 BR2K 서비스의 최신 서비스 상태를 나타내는 SSI를 이용하여, 리더를 제외한 모든 팔로워 서버가 최신 서비스 상태를 따라올 때까지 기다린다.

[line 3 in table.1] 모든 팔로워가 최신 서비스 상태가 되면, 리더는 모든 팔로워에게 서비스 상태에 대한 백업 및 실행 노드의 용량 정리를 시작한다고 알리는 작업을 수행한다. 해당 작업은 리더가 최신 서비스 상태를 나타내는 SSI의 값을 -1로 업데이트하여 ETCD 내부에서 업데이트 이벤트를(§2.2) 발생시켜서 각 팔로워 서버에게 알린다. 각 팔로워는 자신과 연결된 ETCD를 통해 SSI의 업데이트 이벤트를 확인하면, 주기적으로 수행하는 최신 서비스 상태를 따라가는 작업을 중지하고 리더가 서비스 상태의 백업이 종료될 때까지 어떠한 작업도 수행하지 않고 대기한다.

Table 1. Pseudo code for state version up

```
PROGRAM STATE-VERSION-UP
1: Stop receiving requests
2: Wait until all alive followers are in the latest
   service state
3: Notify followers to start the state backup
   (SSI = -1)
4: Back up the current service state and snapshot
   ETCD to off-chain storage
5: Store the state-backup-log in the service registry
6: Clean up the replicated requests related to the
   backed up state
7: Increase the SV and the SSV of leader server
8: Notify followers to end the state backup
   (SSI = 0)
END
```

[line 4 in table.1] State version up 액션에서 서비스 상태의 백업 작업은 리더 서버의 실행 노드에 있는 로컬 상태, 외부 상태, 그리고 서비스 작업에 대한 히스토리 추적에 쓰이는 ETCD의 스냅샷(Snapshot) 데이터를 오프체인 스토리지(Off-chain storage) 저장하는 것이다. 오프체인은[16] 블록체인 이외의 외부 특정 스토리지에 대용량 데이터를 기록하는 방식이다. 블록체인에 대용량 데이터를 저장하는 작업은 많은 시간과 비용이 발생하기 때문에, 확장된 BR2K 기법에서는 오프체인 스토리지를 이용한다.

[line 5 in table.1] 서비스 상태와 ETCD의 스냅샷이 오프체인 스토리지에 저장되면 리더 서버는 이와 관련된 작업에 대한 로그인 상태 백업 로그를(table. 2, State-backup-log) 생성하고, 이를 비대칭 암호화의 공개 키로(Public key) 이용하여 암호화한다. 그리고 암호화된 이 로그를 이더리움 블록체인 기반의 서비스 레지스트리에(§2.3) 저장하여 보관한다. 서비스 레지스트리에 저장된 로그는 이더리움 블록체인을 구성하는 모든 노드에 복제되어 그림 8과 같이 이더리움 블록체인을 구성하는 어떠한 노드에 접근하더라도 가져올 수 있다. 또한, 해당 로그는 오직 BR2K 서비스가 소유한 비밀 키로(Private key) 상태 백업 로그를 복호화 할 수 있어, 임의 주체가 해당 로그를 획득하더라도 사용할 수 없다.

Table 2. State backup log

Attribute	Description
Start	Time of backup started
End	Time of backup ended
Version	State version of BR2K service when performing state back up
Members	Server ID list in BR2K service
Alive members	Alive server ID list in BR2K service
Snapshot meta	Meta-information of ETCD snapshot (size, type, owner, creation date, file storage path, ...)
State data meta	Meta-information of service state (size, type, owner, creation date, file storage path, ...)
storage	Meta-information on back up storage [off-chain] (access location, authentication, authorization, saved form, ...)
Subject	ID of the leader server performed the state back up

[line 6-7 in table.1] 서비스 레지스트리에 상태 백업 로그에 대한 저장 작업이 완료되면, 리더 서버는 ETCD를 이용하여 모든 서버의 실행 노드로 복제된 모든 사용자 요청과 이와 관련된 모든 메타 값을 삭제하여, 모든 실행 노드의 용량을 확보한다. 그리고 리더는 자신의 실행 노드에 대한 로컬 상태 버전인 SSV와 BR2K 서비스의 상태 버전을 나타내는 SV를 1 증가시켜, BR2K 서비스의 최신 상태에 대한 버전을 업데이트한다.

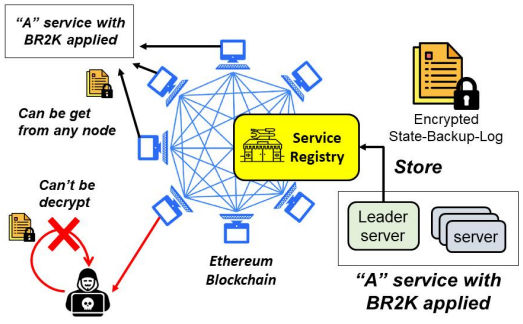

Fig. 8. State-backup-log stored in service registry

[line 8 in table.1] 모든 작업이 완료되면 리더는 state version up 액션이 종료되었음을 최신 서비스 상태를 나타내는 SSI 값을 0으로 업데이트하여 각 팔로워에게 알린다. 이를 확인한 팔로워는 자신의 실행 노드에 대한 로컬 상태 버전인 SSV를 1 증가시키고, 대기 상태에서 벗어나 주기적으로 실행하는 최신 서비스 상태를 따라가는 작업을 재개한다.(fig.2, Follow the latest state)

3. Using state version

서비스 상태에 대한 백업이 완료된 BR2K 서비스는 사용자 요청을 처리하는 중에 중지된 서버에 대한 서비스 상태 복구 작업을 지원할 수 있다. 확장된 기법의 서비스 상태 복구 작업은 *Get the latest state* 액션 기반으로 진행된다. 그림 9는 Get the latest state 액션을 나타낸다.

Get the latest state 액션은 중지되어 다시 시작된 서버가 이더리움 블록체인을 구성하는 임의의 노드에 접근하여 최신 상태 백업 로그를 가져올 때 시작된다. 상태 백업 로그를 가져온 서버는 BR2K 서비스가 소유한 비밀 키로 해당 정보를 복호화하여, 최신 서비스 상태가 백업된 오프체인 스토리지에 대한 메타 정보를 확인한다.

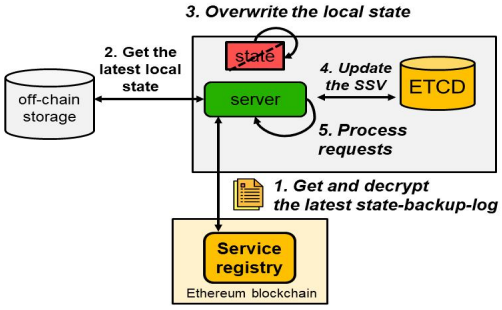

Fig. 9. Get the latest state action

이 메타 정보는 스토리지 접근 정보, 상태를 가져오기 인증 및 권한 정보 그리고 백업된 형태 등이 포함된다. 서버는 이 정보들을 이용하여 오프체인 스토리지에 접근하고 백업된 최신 로컬 상태를 가져온다. 그리고 해당 서버는 가져온 상태를 통해 자신의 실행 노드에 대한 로컬 상태를 변경하고, 자신이 관리하는 SSV를 가져온 로컬 상태의 버전으로 업데이트해준다. 이 작업을 마무리한 서버는 이 로컬 상태를 시작점으로 하여 자신의 실행 노드에 존재하는 모든 사용자 요청을 하나씩 처리하여 다른 서버의 로컬 상태와 동기화한다.

이처럼 백업된 상태는 서비스 상태 복구 작업뿐만 아니라, BR2K 서비스에 새로운 서버가 신속한 합류를 지원하기 위한 용도로 사용될 수 있다. 확장된 기법에서 BR2K 서비스에 새로운 서버가 합류되는 과정은 다음과 같다.

1) 새롭게 합류할 서버는 Get the latest state 액션을 수행하여 자신의 실행 노드에 로컬 상태를 최신 로컬 상태로 업데이트한다.

2) 위의 작업을 완료한 서버는 합류할 BR2K 서비스에 존재하는 사용자 요청들을 자신의 실행 노드로 가져온다.

3) 자신의 실행 노드로 사용자 요청들을 가져온 서버는 BR2K 서비스의 최신 상태와 동기화하기 위하여 해당 요청들 순차적으로 처리한다.

위의 과정은 기존의 BR2K 서비스에서 새로운 서버가 합류되는 과정에서 최신 상태를 가져오는 절차가 하나 추가되었지만, 새로운 서버가 자신의 실행 노드에 복제하여 처리해야 할 사용자 요청의 개수가 현저히 적기 때문에 새로운 서버의 신속한 합류를 지원할 수 있다. 그림 10은 기존의 BR2K 서비스와 확장된 기법에서 BR2K 서비스가 새로운 서버를 합류하는 과정을 비교하여 나타낸 것이다.

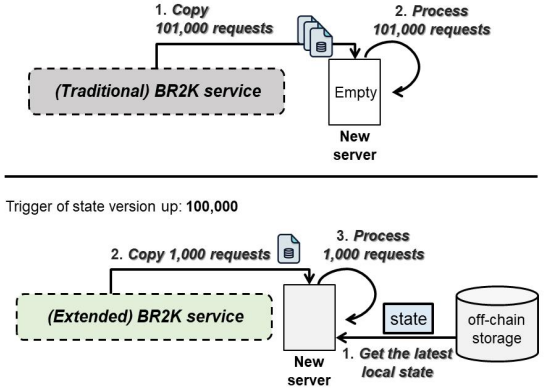

Fig. 10. Add new server in extended BR2K

기존의 BR2K 서비스는 그림 10과 같은 상황에서 약 10만 개의 사용자 요청을 새로운 서버의 실행 노드로 복제하고 해당 요청들을 서버가 처리하는 과정을 거쳐 BR2K 서비스로 합류된다. 이때, 약 10만 개의 사용자 요청을 새로운 서버의 실행 노드로 복제하는 시간은 백업된 최신 서비스 상태를 가져오는 작업보다 현저히 느리다. 사용자 요청은 서비스 상태를 변경하는 값을 포함하여 사용자 요청을 보낸 주체, 상태를 변경하는 절차와 관련된 추가적인 메타 정보가 포함되어 있어, 약 10만 개의 사용자 요청에 대한 데이터는 서비스 상태와 관련된 데이터보다 매우 크기 때문이다. 또한, 기존 방식은 새로운 서버가 많은 사용자 요

청을 순차적으로 처리하는 과정 자체도 상당한 시간이 소요될 수 있다. 한 개의 사용자 요청을 처리하는데 약 1ms가 걸린다고 가정한다면, 10만 개의 사용자 요청을 처리하는데 약 100초 정도 소요될 수 있다. 그러나 주어진 상황에서 확장된 기법에서는 약 1초 정도밖에 소요되지 않아 기존의 방식보다 빠르게 서버를 합류시킬 수 있다.

V. Test

본 장에서는 확장된 BR2K 기법의 서비스 상태 복구 기법의 유효성을 검증하기 위하여 시범적인 블록체인 응용 서비스인 InfoDID에[12] 해당 기법을 적용하여 테스트한다. InfoDID는 분산 식별자 기술인 DID(Decentralized identifiers)[13] 기술을 통해 사용자의 개인 정보를 신뢰성 있게 관리하고 사용자가 자신의 정보를 다른 서비스에 편리하게 제공할 수 있도록 지원하는 서비스이다. 이 서비스는 블록체인 기반의 사용자 정보 저장의 비용 및 용량 한계 때문에 LDAP(Lightweight Directory Access Protocol) 기반의 데이터베이스를 통해 사용자 정보를 저장하여 관리한다. 이러한 민감한 사용자 정보를 관리하는 InfoDID 서비스는 서비스 실행에 대한 결함이 발생하여도 사용자에게 지속적인 서비스가 가능하여야 하며 어떠한 상황에서도 결함에 대한 복구를 진행할 수 있어야 한다.

테스트 환경은 12개의 가상 머신으로 구성된 쿠버네티스 환경에 확장된 BR2K 기법을 적용하여 그림 11과 같이 하나의 실행 노드에 ETCD와 InfoDID 서비스를 복제하여 구성한다. 해당 서비스에 요청을 보내는 사용자는 5명이며, 이 사용자들은 15개의 노드로 구성된 이더리움 블록체인에서 실행되는 서비스 레지스트리에($2.3) 서비스 제공 위치를 확인하여 사용자 요청을 보내게 된다.

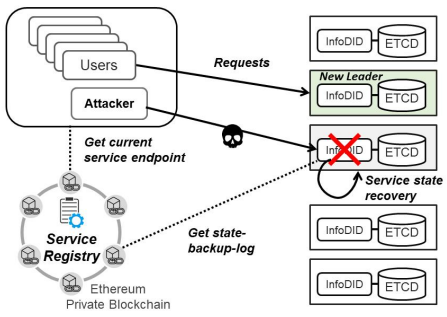

Fig. 11. Test environment

그림 11에 공격자는 30~120초 사이의 랜덤한 주기로 서비스 레지스트리를 통해 서비스를 제공하는 리더 서버를 찾아 해당 서버의 실행 노드를 일시적으로 중지시키는 프로세스이다. 공격당한 실행 노드는 15초 후에 다시 재실행되며, 사용자 요청을 처리하는 중에 중지된 서버는 그림 11과 같이 확장된 BR2K 기법의 서비스 상태 복구를 수행한다. 만약 사용자가 현재 이용 중인 서비스에 요청을 보내는 것이 실패한다면 블록체인 서비스 레지스트리에서 BR2K 기법에 의하여 조정된 새로운 서비스 제공 위치를 가져와 다시 요청을 보낸다. 본 논문의 테스트 과정은 다음과 같다.

(1) 각 사용자는 총 10,000번의 사용자 요청을 초당 2번의 속도로 InfoDID 서버에게 보낸다. 반면에 공격자는 사용자가 요청을 보내는 동안 랜덤한 주기마다 서비스 제공자 역할을 수행하는 리더 서버의 실행 노드를 일시적으로 중지시키는 장애를 발생시킨다.

(2) 사용자 요청을 받은 InfoDID 서버는 요청에 대해서 처리하고 사용자에게 해당 요청에 대한 성공 응답을 보낸다. 이때, InfoDID 서버에게 보내지는 사용자 요청은 사용자의 개인 정보를 서비스에게 등록하는 요청이며, 이 요청으로 인하여 서비스의 최신 상태가 업데이트된다.

(3) 모든 사용자가 10,000번의 요청을 완료하면, 복제된 각 InfoDID 서비스에 저장된 사용자의 개인 정보들이 서로 같은지, 누락된 부분이 없는지 검사하여 서비스 복제가 정확하게 이루어졌는지 확인한다.

Table 3. Test notations

Sign	Explanation
T*service*	Total test time
T*downtime*	Service downtime
NP*frequecy*	Frequency of pauses on execution nodes
NP*number*	Number of pauses on execution nodes
REQ*null*	Number of requests that have only been replicated
REQ*success*	Number of requests successfully processed
REQ*pause*	Number of requests paused while processing
SR	Service replication

Table 4. Test results

	results
T*service*	about 83min
T*downtime*	about 16min
NP*frequecy*	30~120sec
NP*number*	67
REQ*null*	10,193(Out of 50,000 requests)
REQ*success*	39.805(Out of 50,000 requests)
REQ*pause*	2(Out of 50,000 requests)
SR	success

표 4는 테스트 결과에 대한 요약이다. 이 테스트에서 전체 테스트 시간 동안 공격자에 의한 67번의 실행 노드 중지가 발생하였고 이에 따른 서비스 장애 시간은 약 5분 1의 정도 비율로 발생하였다. 또한, 1번 장애가 발생할 때마다 약 15초간 서비스를 제공하지 못함을 확인하였다. (table.4, T*service*, T*downtime*, NP*frequecy*, NP*number*) 이는 BR2K 기법에서 새로운 서비스를 제공하는 리더에 대한 선출 및 이더리움 블록체인 노드들 간의 합의에 필요한 시간에 의존하여 새로운 접속 정보가 갱신되는 것으로 확인되었다. 특히, 대부분의 경우 이더리움 블록체인의 합의 시간으로 인하여 새로운 서비스 제공 정보가 다소 느리게 갱신되었으며, 만약 본 기법을 합의 속도가 이더리움 블록체인보다 빠른 블록체인에 적용하면 서비스 장애 시간은 더욱 줄어들 것으로 판단된다.

또한, 이 테스트에서는 리더 서버가 요청을 처리하는 중에 공격자 때문에 중지되는 상황이 발생하여도, 복제된 각 서비스의 상태가 정확하게 동기화되었는지 확인한다. 표 5에 있는 테스트 결과에서 볼 수 있듯이 빈번한 리더의 변경과 요청 처리 중에 중지된 리더 서버가 발생하여 서비스 상태 복구 작업이 일어났음에도(table 4, REQ*pause*), 처리에 성공한 사용자 요청의 개수와(table 4, REQ*success*) 복제된 각 서비스에 기록된 사용자의 개인 정보에 대한 개수가 서로 일치함을 확인하였다.(table 4, SR) 이는 장애 상황에서도 상태 복제가 정확하게 이루어져 복제된 서비스 간의 일관성이 보장되고 지속적으로 서비스가 제공할 수 있음을 나타낸다.

VI. Conclusions

본 논문에서는 블록체인 응용 서비스의 견고성 제고를 위하여 BR2K 기법의 확장 방법을 제안하였다. 기존의 BR2K 기법을 확장하기 위하여, 서비스 상태에 대한 버전들을 정의하여 실행 노드의 용량 정리 작업 및 상태에 대한 백업 시점을 결정하도록 하였고 이 버전 정보와 블록체인 기반의 서비스 레지스트리를 통해 서비스 상태의 백업을 체계적이고 안전하게 지원하도록 하였다. 또한, 백업된 상태를 이용한 체계적인 서비스 상태 복구와 새로운 서버의 신속한 합류 방법을 제안하여, BR2K 기법의 견고성을 높이고 다양한 블록체인 응용 서비스에 해당 기법을 손쉽게 적용할 수 있도록 하였다. 더불어 시범적인 블록체인 응용 서비스인 InfoDID 서비스에 확장된 BR2K 기법을 적용하고 이를 테스트하여 해당 기법의 유효성을 확인하였다. 추후 연구에서는 본 기법에 대한 서비스 복구에 대한 시간을 줄이고 활용성을 높이기 위하여, 이더리움보다 합의 속도가 빠른 글로벌 블록체인인 Klaytn[17]과 EOS[18] 블록체인을 이용하여 해당 기법을 확장하고 추가적인 활용 도구도 개발하여 실용성을 제고할 예정이다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(No. 2019R1I1A3A01052970)

REFERENCES

[1] Lee, Jinwook, et al. "Price-Bands: A Technical Tool for Stock Trading", Blockchain Technologies, pp. 221-246, 2021. DOI: 10.1007/978-981-33-6137-9_10

[2] Liu Weiwei, et al. "AucSwap: A Vickrey auction modeled decentralized cross-blockchain asset transfer protocol", Journal of Systems Architecture, Vol. 117, pp. 102102, Aug. 2021. DOI: 10.1016/j.sysarc.2021.102102

[3] Khan, I.R, Baig, M.A, "Managing Medical Supply Chain Using Blockchain Technology", Blockchain for Healthcare Systems, pp. 149-158, Jul. 2021. DOI: 10.1201/9781003141471-10

[4] Anon, "A Blockchain-Based Distributed Authentication System for Healthcare", International Journal of Healthcare Information Systems and Informatics, Vol. 16, No. 4, Oct. 2021. DOI: 10.4018/ijhisi.20211001oa04

[5] Brown, M. , et al, "BLOCKCHAIN DOUBLE-SPEND ATTACK DURATION", Probability in the Engineering and Informational Sciences, Vol. 35, No. 4, pp. 858-866, May 2020. DOI: 10.1017/s0269964820000212

[6] MH Kwon, MJ Lee. "BR2K: A Replication and Recovery Technique Using Kubernetes for Blockchain Services", Proceedings of the Korean Society of Computer information Conference, Vol. 25, No. 10, pp. 77-86, Oct 2020. DOI: 10.9708/jksci.2020.25.10.077

[7] Sagar,V. , et al, 2020. "Ethereum 2.0 Blockchain in Healthcare and Healthcare Based Internet-of-Things Devices", Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences, pp.225-233, 2021. DOI: 10.1007/978-981-15-7533-4_17.

[8] Zheng, G. et al., 2020. "Operation Principles of Smart Contract",

Ethereum Smart Contract Development in Solidity, pp.159-195, Sep. 2020. DOI: 10.1007/978-981-15-6218-1_6.

[9] ETCD, https://etcd.io/

[10] Larsson, L. et al., 2020. "Impact of etcd deployment on Kubernetes, Istio, and application performance", Software: Practice and Experience, Vol. 50, No. 10, pp.1986-2007, Aug. 2020. DOI: 10.1002/spe.2885

[11] Howard, H. , Mortier, R., "Paxos vs Raft", Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data, Apr. 2020. DOI: 10.1145/3380787.3393681

[12] MH Kwon, MJ Lee. "InfoDID: A robust user information management serivce based on Decentralized Identifiers", Journal of the Korea Society of Computer and Information, Vol. 26, No. 4, pp. 75-84, Apr. 2021. DOI: 10.9708/jksci.2021.26.04.075

[13] Brunner, C. et al., "DID and VC:Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust", 2020 the 3rd International Conference on Blockchain Technology and Applications, Dec. 2020. DOI: 10.1145/3446983.3446992

[14] David Balla, et al. "Adaptive scaling of Kubernetes pods", IEEE/IFIP Network Operations and Management Symposium, pp. 20-24, Apr. 2020. DOI: 10.1109/NOMS47738.2020.9110428

[15] M3, https://m3db.io/

[16] Hepp, Thomas, et al. "On-chain vs. off-chain storage for supply-and blockchain integration.", it-Information Technology, pp. 283-29,2018. DOI: 10.1515/itit-2018-0019

[17] Klaytn, *https://www.klaytn.com/*

[18] EOS, *https://eos.io/*

Authors

Min-Ho Kwon received the B.S/B.A degrees in IT convergence/Economics from University of Ulsan, Korea, in 2020. He is currently an M.S student in Dept. of Electrical/Electronic and C omputer Engineering, University of Ulsan.

He is interested in blockchain technology, distributed comput ing, and cloud computing.

Myung-Joon Lee received the B.S. degree in Mathematics from Seoul National University in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST in 1982 and 1991, respectively.

Dr. Lee joined the faculty of the Department of Computer S cience at University of Ulsan, Ulsan, Korea, in 1982. He is currently a Professor in the School of IT Convergence, Univ ersity of Ulsan. He is interested in blockchain technology, di stributed computing, and mobile/cloud service.