Journal of Korea Multimedia Society Vol. 24, No. 1, January 2021(pp. 75-94)
https://doi.org/10.9717/kmms.2020.24.1.075

멀티 클라우드 서비스 공통 플랫폼 설계 및 구현

김수영†, 김병섭†, 손석호†, 서지훈†, 김윤곤†, 강동재†

Design and Implementation of Multi-Cloud Service Common Platform

Sooyoung Kim†, Byoungseob Kim†, Seokho Son†, Jihoon Seo†, Yunkon Kim†, Dongjae Kang†

ABSTRACT

The 4th industrial revolution needs a fusion of artificial intelligence, robotics, the Internet of Things (IoT), edge computing, and other technologies. For the fusion of technologies, cloud computing technology can provide flexible and high-performance computing resources so that cloud computing can be the foundation technology of new emerging services. The emerging services become a global-scale, and require much higher performance, availability, and reliability. Public cloud providers already provide global-scale services. However, their services, costs, performance, and policies are different. Enterprises/ developers to come out with a new inter-operable service are experiencing vendor lock-in problems. Therefore, multi-cloud technology that federatively resolves the limitations of single cloud providers is required. We propose a software platform, denoted as Cloud-Barista. Cloud-Barista is a multi-cloud service common platform for federating multiple clouds. It makes multiple cloud services as a single service. We explain the functional architecture of the proposed platform that consists of several frameworks, and then discuss the main design and implementation issues of each framework. To verify the feasibility of our proposal, we show a demonstration which is to create 18 virtual machines on several cloud providers, combine them as a single resource, and manage it.

Key words: Multi-cloud, Multi-cloud infrastructure, Multi-cloud application, Multi-cloud monitoring

1. 서 론

IT 분야에서 대부분의 서비스는 클라우드 컴퓨팅 인프라를 활용하여 운영되고 있다. 최근 각광 받기 시작한 인공지능 서비스, 빅데이터 서비스, 그리고 엣지 컴퓨팅 서비스 등 제4차 산업혁명의 근간을 이루는 서비스들 역시, 보다 광범위한 지역을 서비스 대상으로 하거나 높은 성능이나 가용성, 그리고 안정성 등을 제공하기 위해서 클라우드 컴퓨팅 인프라를 기반으로 설계되고 있다.

그러나 클라우드 사업자마다 상이한 인프라 서비스(Infrastructure-as-a-Service, 이하 IaaS)를 제공

※ Corresponding Author : Sooyoung Kim, Address: (34129) 218 Gajeong-ro, Yuseong-gu, Daejeon, KOREA, TEL : +82-42-860-1682, FAX : +82-42-860-6699, E-mail : sykim@etri.re.kr
Receipt date : Oct. 26, 2020, Revision date : Dec. 7, 2020
Approval date : Jan. 4, 2021
† Cloud Computing SW Research Section, Electronics and Telecommunications Research Institute
(E-mail : powerkim@etri.re.kr)
(E-mail : shsonkorea@etri.re.kr)
(E-mail : jihoon.seo@etri.re.kr)
(E-mail : yunkon.kim@etri.re.kr)
(E-mail : djkang@etri.re.kr)
※ This work was supported by Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Korea government(MSIT) (No.1711102975, Development of Multi-Cloud Service Common Framework Technology, Maximizing Utilization and Spreading of Various Multi-Cloud Services)

하기 때문에 자원의 위치, 성능, 비용에 대한 제약이 발생할 수 있고, API 또한 상이하여 다수의 클라우드 인프라 연동이 쉽지 않다. 따라서, 앞서 언급한 서비스들을 광범위한 지역에 대규모로 구축하거나 탄력적으로 운영하는데 어려움이 발생하고 있다. 이러한 문제점을 해결하기 위해 글로벌 스케일의 서비스 제공자들은 다양한 퍼블릭 클라우드를 대상으로 최적의 컴퓨팅 인프라 환경을 구축하고 운영할 수 있는 멀티 클라우드 기술에 많은 관심을 기울이고 있다.

멀티 클라우드 기술은 두 개 이상의 퍼블릭 또는 프라이빗 클라우드를 연계, 운용, 활용, 관리하기 위한 차세대 클라우드 기반 기술로서, 다수의 퍼블릭 클라우드의 인프라(IaaS) 서비스를 연동하여 통합 운영하고, 구성된 멀티 클라우드 인프라상에서 클라우드 응용(PaaS, SaaS)의 유연한 배치, 운용 및 제공을 가능케 하는 기술이다[1].

멀티 클라우드 기술을 활용하면 다양한 지역의 퍼블릭 클라우드들을 유연하게 활용하여 단일 클라우드가 갖는 지역적인 한계성을 극복하여 사용자와 가장 가까운 위치의 퍼블릭 클라우드를 통한 서비스 제공이 가능하므로 서비스 지연 문제를 개선할 수 있다. 또한 특정 클라우드의 장애로 인해 생길 수 있는 서비스 장애를 완화할 수 있을 뿐 아니라 서비스가 필요로 하는 컴퓨팅 인프라 자원을 선정하여 활용 가능하므로 저렴한 운영 비용으로 최적의 서비스 품질을 제공할 수 있게 된다[2].

또한, 클라우드 협업 도구 개발 연구[3]에서는 다수 사용자들간의 공동 작업이 가능한 클라우드 컴퓨팅 기반 협업 이미지 제작 도구를 개발하는 과정의 어려움으로, 여러 클라우드 서비스 제공자들의 서로 다른 정책과 기술을 제공함에 따라 특정 클라우드 서비스에 기반하여 개발한 응용 소프트웨어가 특정 서비스 업체에 종속되는 문제점(Vendor Lock-in Problem)을 호소하였다.

본 논문에서는 이러한 멀티 클라우드를 연동하여 다양한 서비스에 활용할 수 있도록 하는 멀티 클라우드 서비스 공통 플랫폼의 주요 설계 내용과 구현 이슈를 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안한 멀티 클라우드 서비스 공통 플랫폼의 관련 기술과 연구를 소개하고, 3장에서는 멀티 클라우드 서비스 공통 플랫폼의 구성 요소들의 설계와 구현의 주요 사항들을 설명한다. 그리고 4장에서는 플랫폼의 개발 현황과 시험 방법 및 결과를 살펴보고 5장에서 결론을 맺는다.

2. 관련 기술 및 연구

멀티 클라우드를 구성하여 컴퓨팅 성능을 극대화하거나 비용을 절감하고, 서비스 범위를 확대하거나 품질을 향상시키기 위한 개념적인 여러 연구들이 지속되어 왔으며[4,5], 이러한 연구들이 구체화되어 다양한 클라우드 인프라를 사용할 수 있도록 지원하는 Apache의 libCloud[6], jCloud[7], 그리고 CloudVE의 CloudBridge[8] 기술들이 소개되어 활용되고 있다. 이들 기술은 라이브러리 형태로 제공되어 클라우드 응용이 동일한 API를 기반으로 여러 클라우드 인프라에 접근하여 해당 인프라 자원을 사용할 수 있도록 지원하고 있다. jCloud는 자바 언어를 기반으로 하는 라이브러리이며, libCloud와 CloudBridge는 파이썬 언어를 기반으로 하는 라이브러리이다. 이러한 라이브러리 방식은 언어 의존성이 강하고 특정 클라우드에 종속된 기능 제공에 초점을 두고 있어 상호운용성 이슈가 존재한다.

한편, 다양한 클라우드 인프라에 공통된 방식으로 접근하여 클라우드 자원을 획득한 후 해당 자원에 응용을 배치하고 이들의 생애주기까지 관리하는 기술이 소개되었다. 대표적으로 Scalr CMP[9]와 Flexera의 RightScale[10]가 있으며, 동일한 사용자 인터페이스를 통해 여러 클라우드 인프라에서 제공하는 자원들을 손쉽게 접근하여 여러 인프라 자원을 효과적으로 관리할 수 있도록 지원한다. Dynamic Over Cloud[11]는 기존 다중 클라우드와 사용자간 커뮤니케이션을 위해 인터페이스 프록시(Interface Proxy)라는 경량 계층을 추가하였고, 이를 바탕으로 마이크로서비스 기반 사물인터넷 클라우드 서비스 구성 환경을 제공한다. 쿠버네티스(Kubernetes), 세프(Ceph), 컨테이너 네트워킹(Container Networking), 엔보이(Envoy) 등 기존 툴을 구동할 수 있는 멀티 클라우드 개념 및 컴포넌트, 서비스의 운영 라이프사이클, 그리고 스마트 에너지 IoT-Cloud 서비스에 대해 우수한 연구 결과를 보이고 있다. 하지만, 제한된 클라우드 사용으로 보다 다양한 클라우드를 활용한 연구 개발이 필요하다. 또한 최근 많이 사용되는 Harshi-

corp의 Terraform[12]는 선언적 스크립트 언어 형식을 통해 다양한 클라우드 인프라 생성 및 관리를 지원한다. 그러나 이러한 방식은 애플리케이션의 실행 범위가 각 클라우드 내의 인프라 자원으로 국한되어 있다.

본 연구에서는 앞선 연구들과는 달리 여러 클라우드 서비스 제공자들이 제공하는 클라우드 인프라를 공통된 방식으로 접근할 수 있는 프레임워크를 제공하여 라이브러리 방식보다 언어 독립성을 제공하고, 이를 기반으로 동일한 사용자 인터페이스를 통해 멀티 클라우드 상의 인프라들을 하나의 논리적 객체로 손쉽게 묶고 관리할 수 있는 프레임워크를 제공한다. 또한 이러한 하나의 논리적 객체를 토대로, 사용자들이 이질적인 클라우드 인프라 플랫폼에 신경을 쓰지 않고 다양한 지역적, 비용적 요구사항 등을 만족하도록 클라우드 애플리케이션의 실행을 지원하는 프레임워크를 제공하는 공통된 멀티 클라우드 서비스 플랫폼을 설계하여 제안한다.

3. 플랫폼 설계 및 구현

3.1 플랫폼 설계 목표

멀티 클라우드 서비스 공통 플랫폼(이하 Cloud-Barista)은 단일 클라우드 활용의 한계성을 극복하고 멀티 클라우드의 활용·확산을 위해 다양한 멀티 클라우드 인프라 서비스와 멀티 클라우드 애플리케이션의 운용 및 관리에 필수적인 공통 기반 기술과 누구나 손쉽게 사용할 수 있는 개방형 API를 제공한다. 사용자들은 이 플랫폼을 통해 멀티 클라우드 서비스 및 애플리케이션의 실행/관리에 공통적으로 필요한 기능들에 대한 중복 개발의 비효율성을 제거하여 생산성을 크게 향상할 수 있으며 다양한 산업 및 서비스 분야에서 멀티 클라우드 활용이 가능하다.

Cloud-Barista는 상기 목표를 실현을 위해 다음과 같은 세부 목표를 달성하도록 설계하였다.

- 공통 핵심 기능 제공 (Commonness) : 다양한 멀티 클라우드 서비스의 운용 및 관리에 필수적으로 요구되는 공통 핵심 기능 제공
- 효율성 제공 (Efficiency) : 멀티 클라우드 서비스에 필요한 공통 기능들을 중복 개발해야 하는 비효율성을 제거함으로써 생산성 향상
- 개방성 제공 (Openness) : 누구나 손쉽게 사용할 수 있는 개방형 API를 제공하며, 개방형 프레임워크 아키텍처 기반의 공개 SW로 개발

또한 아래의 목적을 만족할 수 있는 SW 구조를 갖도록 설계하였다.

- 마이크로서비스 아키텍처의 프레임워크 기반의 SW 구조 : 개별 기능 모듈을 독립적인 서비스 단위로 구조화하여 개별 프레임워크별 형상과 개발에 대한 독립성을 확보하고, 유연한 통합 및 개발 복잡도 완화하기 위한 구조
- 플러그인을 통한 손쉬운 확장 가능한 SW 구조 : 플러그인이 가능한 표준 구조를 제시하여, 기능 확장 편의성을 제공하기 위한 구조

3.2 전체 플랫폼 구성

Cloud-Barista는 4개의 주요 프레임워크로 구성되어 있으며, 전체적인 기능 구조는 Fig. 1.과 같다. 즉 Cloud-Barista는 멀티 클라우드 인프라 연동 프레임워크(Multi-Cloud Infrastructure Federation Framework)와 멀티 클라우드 인프라 서비스 통합 관리 프레임워크(Multi-Cloud Infrastructure Service Management Framework), 멀티 클라우드 애플리케이션 통합 관리 프레임워크(Multi-Cloud Application Management Framework), 그리고 멀티 클라우드 통합 모니터링 프레임워크(Multi-Cloud Service Monitoring Framework)로 구성되어 있다. 그리고 추가로 멀티 클라우드 서비스 운용 관리 기능(Cloud-Barista Operation and Management)과 개방형 API 기반 사용자 인터페이스(User Interface)를 제공하고 있다[13].

멀티 클라우드 인프라 연동 프레임워크(이하 CB-Spider)는 이종의 멀티 클라우드 인프라를 연동하고 동일 방법으로 제어하고 관리하는 인터페이스를 제공하고, 멀티 클라우드 인프라 서비스 통합 관리 프레임워크(이하 CB-Tumblebug)는 CB-Spider를 통해 생성한 멀티 클라우드 인프라 서비스들을 효과적 운용하고 통합적으로 관리하는 인터페이스를 제공하며, 멀티 클라우드 애플리케이션 통합 관리 프레임워크(이하 CB-Ladybug)는 CB-Tumblebug을 통해 생성되고 관리되는 멀티 클라우드 인프라 서비스와 자원들을 기반으로 멀티 클라우드 애플리케이션들을 효과적 운용하고 관리하기 위한 인터페이스를 제공한다. 그리고 멀티 클라우드 통합 모니터링 프레임

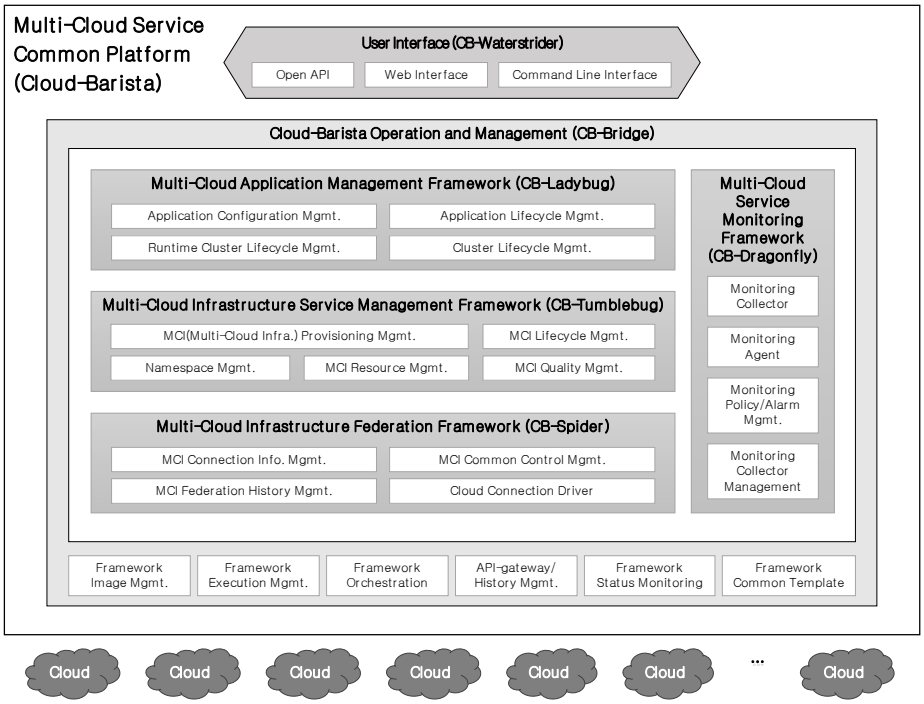

Fig. 1. The Functional Architecture of Multi-Cloud Service Common Platform (Cloud-Barista).

워크(이하 CB-Dragonfly)는 인프라 서비스 및 애플리케이션과 관련한 모니터링 정보를 수집하고 사용자에게 해당 정보를 제공하는 역할을 담당한다. 또한 멀티 클라우드 서비스 운용 관리 기능(이하 CB-Bridge)은 운영자의 손쉬운 Cloud-Barista 실행 및 개방형 API의 중계를 위해 각 프레임워크들 및 관련 구성 요소들을 실행하고 관리하는 운용 관리 도구와 API 호출 중계기를 제공한다. 그리고 개방형 API 기반 사용자 인터페이스는 사용자가 Cloud-Barista에 접근하기 위해 GUI(Graphic User Interface) 기반의 웹 도구와 명령어 기반의 CLI(Command-Line Interface) 도구를 제공한다.

3.3 멀티 클라우드 인프라 연동 프레임워크 (CB-Spider)

3.3.1 개요 및 기능

최근 단일 클라우드의 기능 제약 및 자원 한계를 극복하기 위하여 멀티 클라우드 인프라 환경을 활용하고자 하는 수요가 증가하고 있다[2]. 그러나, 각 클라우드별로 연동 및 활용 방법이 상이하여 다양한 멀티 클라우드 인프라의 활용에 어려움이 있다. 멀티 클라우드 인프라 연동 프레임워크(이하 CB-Spider)는 이러한 이종 멀티 클라우드 인프라 자원을 추상화하여 동일한 API 및 동일한 방법으로 운영 관리할 수 있는 기능들을 제공한다[13].

CB-Spider에서 관리하는 주요 대상은 다음과 같다.

- 멀티 클라우드 인프라 연동 정보 (Multi-Cloud Infrastructure Connection Information) : 이종의 여러 클라우드 인프라 연동을 위해서 필요한 각 클라우드 연결 설정 정보
- 멀티 클라우드 공통 제어 인터페이스 (Multi-Cloud Common Control Interface) : 서로 다른 멀티 클라우드 인프라 자원을 동일한 자원 규격으로 추상화하고 동일한 방식으로의 제어를 위한 단일 인터페이스
- 멀티 클라우드 연동 이력 정보 (Multi-Cloud Federation History) : 효과적인 멀티 클라우드 인프라의 활용 정책 마련을 위해 각 클라우드 연결 및 호출 이력 정보 수집

CB-Spider는 Fig. 2와 같이 크게 3가지 기능을 제공한다. (1) 이종의 여러 클라우드 인프라 연동을 위해서 필요한 연동 설정 정보를 통합 관리하는 기능을

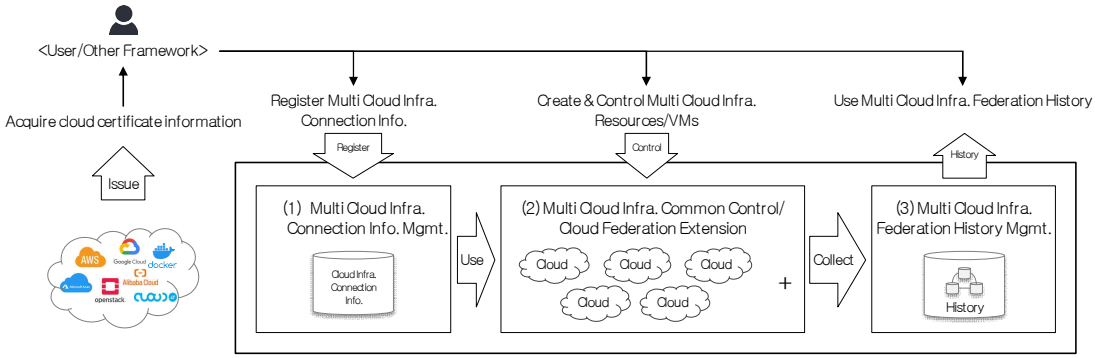

Fig. 2. An Overview of Multi-Cloud Infrastructure Federation Framework.

제공하고, (2) 서로 다른 멀티 클라우드 인프라 자원을 동일한 자원 규격으로 추상화하여 단일 인터페이스 및 공통 제어 방법을 통하여 멀티 클라우드를 제어할 수 있는 기능과 지속적인 클라우드 추가 연동을 위한 클라우드 연동 확장 기능을 제공하며, (3) 운영중인 여러 멀티 클라우드 서비스 공통 플랫폼에서 연동된 멀티 클라우드에 연결 요청 및 API 호출시마다 이에 대한 이력 정보를 통합 저장 관리하는 기능을 제공한다. 사용자는 연동 이력 정보 분석을 활용하여 멀티 클라우드 인프라 활용 정책 등에 활용할

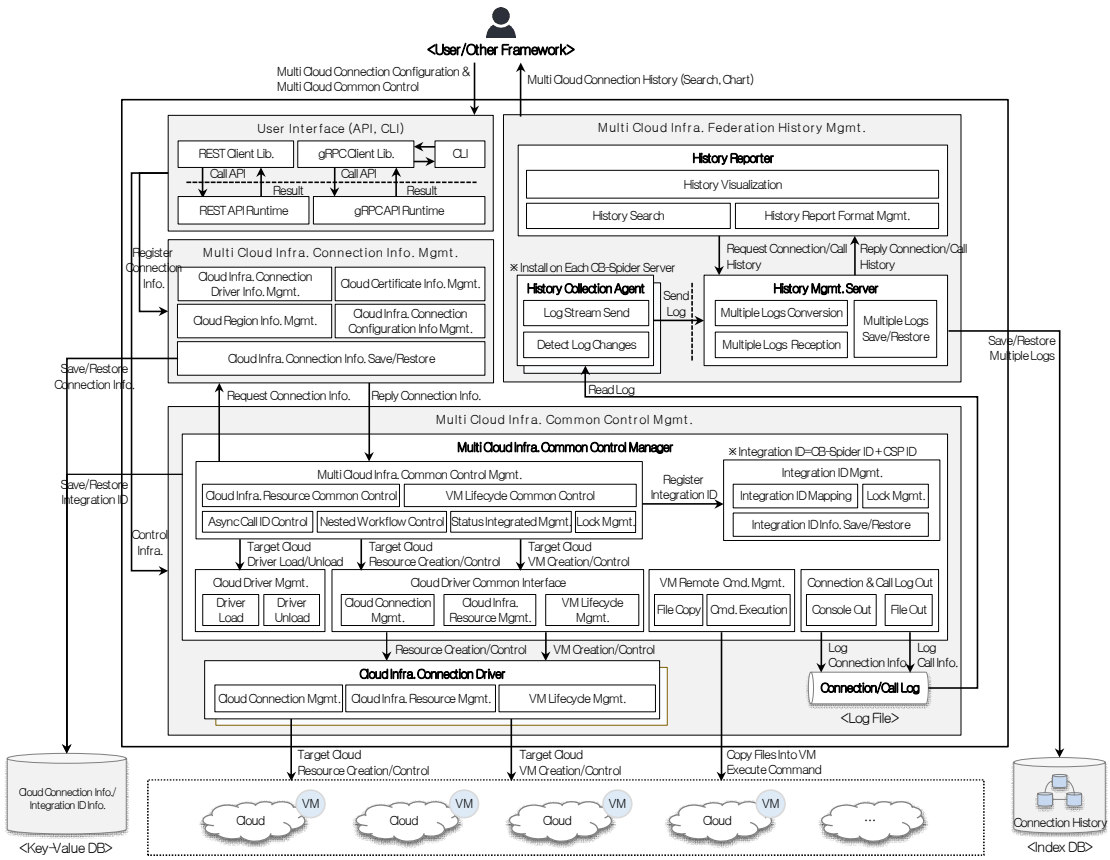

Fig. 3. The Functional Architecture of Multi-Cloud Infrastructure Federation Framework.

수 있다.

3.3.2 구조 및 상세 기능

CB-Spider의 기능적 구조는 Fig. 3에서 보는 바와 같이 4개의 주요 블록으로 구성되며, 세부 내용은 다음과 같다.

(1) 사용자 인터페이스(User Interface) 블록

사용자에게 HTTP 기반 REST(REpresentational State Transfer) 사용자 API 및 gRPC(google Remote Procedure Call) 기반 사용자 API를 제공한다.

(2) 멀티 클라우드 인프라 연동 정보 통합 관리(Multi Cloud Infrastructure Connection Info. Management) 블록

사용자로부터 연동 대상 클라우드의 연동 정보를 입력받아 관리하며, 해당 정보는 클라우드 연동시 활용된다. 클라우드 인프라 연동 정보에는 대표적으로 대상 클라우드 연동 드라이버 정보, 대상 클라우드 접속에 필요한 사용자의 계정이나 비밀번호, 인증키 등의 인증 정보, 대상 클라우드 서비스 지역 정보 등이다.

(3) 멀티 클라우드 인프라 공통 제어 관리(Multi Cloud Infrastructure Common Control Management) 블록

멀티 클라우드 인프라 공통 제어 관리기(Multi Cloud Infrastructure Common Control Manager)와 클라우드 인프라 연동 드라이버(Cloud Infrastructure Connection Driver)로 구성된다. 멀티 클라우드 인프라 공통 제어 관리기는 서로 다른 클라우드의 인프라 자원 및 기능을 추상화하여 자원 규격을 정의하고 공통 인터페이스로 제공하여 단일 인터페이스를 통해 가상 머신의 라이프사이클 제어나 원격 명령 제어 등 동일한 방식의 제어가 가능하도록 지원하고, 클라우드 드라이버 공통 인터페이스 규격을 정의하여 제공한다. 클라우드 인프라 연동 드라이버는 연동하고자 하는 대상 클라우드 별 클라우드 드라이버 공통 인터페이스의 구현체이며, 각 연동 드라이버의 구현은 대상 클라우드의 전용 인터페이스를 이용하여 대상 클라우드로부터 자원 및 가상 머신을 제어하며 필요시 드라이버 계층에서 기능을 보완 및 은닉하여 공통 규격으로 추상화한다. 또한 통합 ID 관리(Integrated ID Management)를 통해 다양한 멀티 클라우드에서 서로 다른 자원의 ID 관리 체계를 통합하여 통일된 ID 관리 체계로 제공한다.

(4) 멀티 클라우드 인프라 연동 이력 정보 관리(Multi Cloud Infrastructure Federation History Management) 블록

연동 이력 수집 에이전트(History Collection Agent), 멀티 클라우드 인프라 연동 이력 정보 관리 서버(History Management Server)와 멀티 클라우드 인프라 연동 이력 정보 리포터(History Reporter)로 구성된다. 연동 이력 수집 에이전트는 가동 중인 연동 프레임워크 서버 마다 운영되며, 대상 연동 프레임워크 서버가 제공하는 연결 및 호출 로그 스트림을 수집하여 연동 이력 정보 관리 서버에 전송한다. 멀티 클라우드 인프라 연동 이력 정보 관리 서버는 연동 이력 수집 에이전트를 통하여 도처에서 운영 중인 여러 연동 프레임워크 서버로부터 연결 및 호출 로그를 수집함으로써 모든 멀티 클라우드 연동 이력을 아카이빙하여 사용자에게 통합 제공한다. 마지막으로 멀티 클라우드 인프라 연동 이력 정보 리포터는 연동 이력 정보 관리 서버를 통해서 아카이빙된 연결 및 호출 연동 이력 정보에 대한 다양한 형태의 분석 정보를 제공하며, 사용자는 연동 이력 정보 분석을 통하여 향후 사용자의 응용 패턴에 따른 멀티 클라우드 인프라 선택 등 멀티 클라우드 인프라 활용 정책 결정에 활용할 수 있다.

3.3.3 주요 설계 및 구현 이슈

CB-Spider의 주요 설계 및 구현 이슈는 멀티 클라우드 인프라 자원 추상화 및 인터페이스 규격과 멀티 클라우드 인프라 연동 확장 구조에 관한 것이다.

(1) 멀티 클라우드 인프라 자원 추상화 및 인터페이스 규격

멀티 클라우드 인프라 자원 추상화 방안 및 활용 구성 방법은 다음과 같다. 멀티 클라우드들이 제공하는 인프라 자원의 제공 기능은 Fig. 4(a)에서 보는 바와 같이 (A) 공통 기능, (S) 일부 공통 기능 및 (E) 개별 클라우드만 제공하는 특화 기능으로 구분할 수 있다. 만약 CB-Spider가 공통 기능만을 제공한다면,

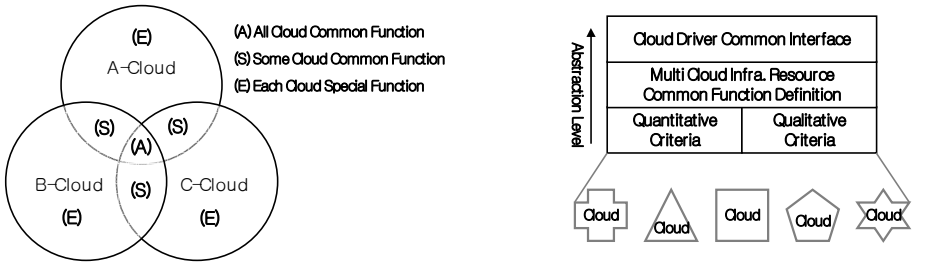

Fig. 4. Relationship and Abstraction Methods of Multi Cloud Infrastructure's Functions.

멀티 클라우드 인프라 자원 제공의 범용성이나 확장성 측면에서는 최적일 수 있지만, 사용자에게 각 클라우드가 제공하는 특화 기능을 제공할 수 없어 효율성이 감소한다. 반면 개별 클라우드가 제공하는 모든 특화 기능을 제공한다면, 클라우드별로 다른 인터페이스 및 방법을 제공해야 함으로써 시스템의 복잡도가 증가하고 확장성이 축소된다. 이러한 이슈를 해결하기 위해 CB-Spider는 일부 공통 기능을 중심으로 공통 기능을 제공하지 않는 클라우드에 대해서는 드라이버 계층에서의 기능 보완을 통해 범용성과 확장성, 그리고 자원 활용 효율성을 제공하도록 추상화하여 설계/개발을 진행 중이다. 추상화 방법은 Fig. 4 (b)와 같이 연동 대상 클라우드 인프라 자원의 정량적 기준(기능 제공 여부 등)과 정성적 기준(자원 생성 순서 규칙 등)에서 기능을 분석하고, 멀티 클라우드 인프라 자원의 공통 기능을 추출하고 정의하였다. 인프라 자원 공통 기능은 시스템화를 위하여 Fig. 5와 같은 클라우드 드라이버 공통 인터페이스로 형상화하였으며 연동 대상 클라우드 별로 드라이버를 구현하여 추가 연동할 수 있도록 설계하였다.

사용자는 최종적으로 CB-Spider를 통해 Fig. 6과 같은 멀티 클라우드 인프라 자원 및 가상 머신 구성이 가능하다. 사용자가 정의한 'Seoul-connection' 및 'Ohio-connection'과 같은 연결 설정 정보를 중심으로 VPC(Virtual Private Cloud), 서브넷 등 가상 네트워크 구성이 가능하며, 사설 IP(Private IP) 및 공용 IP(Public IP) 기반의 망 운영이 가능하다. 사용자는 연동 프레임워크 인터페이스를 사용하여 동일한 방법으로 인프라 자원을 생성할 수 있으며, 실질적인 자원의 생성은 각 클라우드 서비스 제공자의 리전에서 생성 관리된다.

(2) 멀티 클라우드 인프라 연동 확장 구조

CB-Spider는 지속적인 클라우드 인프라 추가 연동 지원을 위해 Fig. 7과 같은 클라우드 인프라 연동 드라이버 기반의 플러그인이 가능한 구조를 제공한다. 클라우드 인프라 연동 드라이버는 클라우드 인프라 연동 드라이버 공통 인터페이스 규격을 준수하여 개발할 수 있으며, 개발된 드라이버는 인프라 연동 정보 통합 관리 블록을 통해서 CB-Spider에 동적으로 등록/사용될 수 있다. 이러한 인터페이스-드라이버 구조를 통하여 다양한 멀티 클라우드 인프라 자원 및 가상 머신을 동일한 방법으로 공통 제어 및 관리할 수 있다.

3.4 멀티 클라우드 인프라 서비스 통합 관리 프레임워

[Cloud Driver Common API Example]

```
type CloudConnection interface {
    CreateImageHandler() (irs.ImageHandler, error)
    CreateVPCHandler() (irs.VPCHandler, error)
    CreateSecurityHandler() (irs.SecurityHandler, error)
    CreateKeyPairHandler() (irs.KeyPairHandler, error)
    CreateVMHandler() (irs.VMHandler, error)
    CreateVMSpecHandler() (irs.VMSpecHandler, error)
    IsConnected() (bool, error)
    Close() error
}

type VPCHandler interface {
    CreateVPC(vpcReqInfo VPCReqInfo) (VPCInfo, error)
    ListVPC() ([]*VPCInfo, error)
    GetVPC(vpcIID IID) (VPCInfo, error)
    DeleteVPC(vpcIID IID) (bool, error)
}

type VMHandler interface {
    StartVM(vmReqInfo VMReqInfo) (VMInfo, error)
    SuspendVM(vmIID IID) (VMStatus, error)
    ResumeVM(vmIID IID) (VMStatus, error)
    RebootVM(vmIID IID) (VMStatus, error)
    TerminateVM(vmIID IID) (VMStatus, error)
    ListVMStatus() ([]*VMStatusInfo, error)
    GetVMStatus(vmIID IID) (VMStatus, error)
    ListVM() ([]*VMInfo, error)
    GetVM(vmIID IID) (VMInfo, error)
}
...
```

Fig. 5. An Example of Cloud Driver Common Interface.

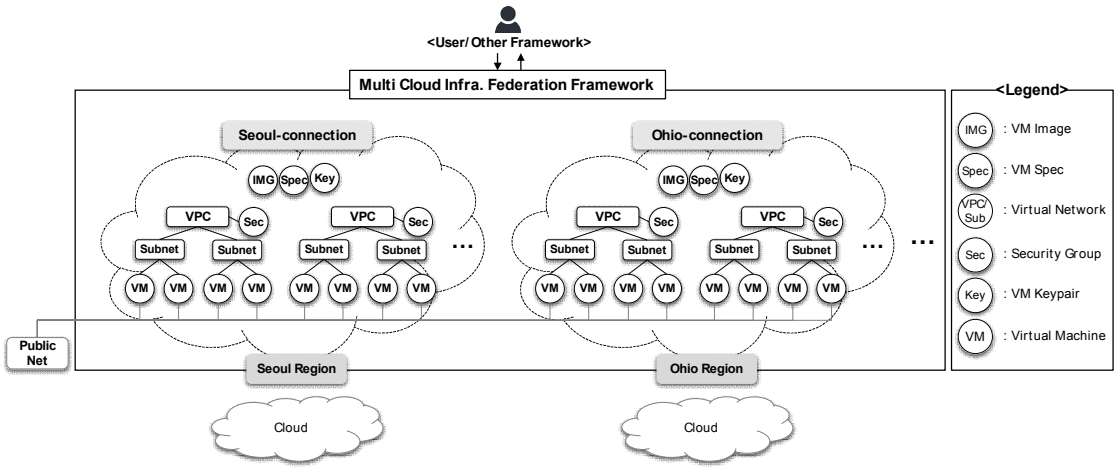

Fig. 6. The Abstracted Multi Cloud Infrastructure Resources.

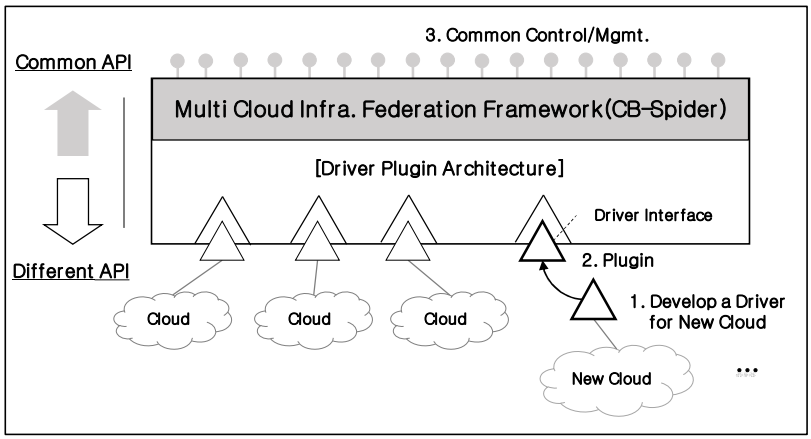

Fig. 7. The Driver Plugin Architecture.

크 (CB-Tumblebug)

3.4.1 개요 및 기능

다수의 클라우드에서 제공되는 컴퓨팅 인프라 서비스(IaaS)를 동시에 사용하면 사용자는 다양한 컴퓨팅 리소스(예를 들면, 다양한 사양의 가상 머신)로 구성된 대규모 컴퓨팅 인프라를 구성 및 사용할 수 있다. 그러나, 다수의 클라우드에서 제공되는 클라우드 서비스를 배치 및 운용하기는 쉽지 않다. 멀티 클라우드 환경에서는 선택이 필요한 컴퓨팅 리소스의 종류도 더 다양화되고 관리의 대상도 증가하므로 컴퓨팅 인프라의 구조 및 운용을 위한 복잡성이 상승하기 때문이다. 따라서, 사용자의 필요에 따라 멀티 클라우드 기반의 컴퓨팅 인프라의 구성을 최적화하고 멀티 클라우드에 걸친 컴퓨팅 자원들을 통합적으로 관리할 방법이 필요하다.

멀티 클라우드 인프라 서비스 통합 관리 프레임워크(이하 CB-Tumblebug)는 사용자의 요구사항에 적합한 최적의 멀티 클라우드 인프라를 구성하고 멀티 클라우드 인프라를 구성하는 요소 자원인 가상 머신(Virtual Machine, 이하 VM)이나 컨테이너들을 통합적으로 배치하며, 생성된 멀티 클라우드 인프라를 통합적으로 제어 및 관리하여 멀티 클라우드 인프라의 관리를 자동화하는 프레임워크이다[10]. 사용자는 이 프레임워크를 통해 손쉽게 전 세계에 걸친 멀티 클라우드 인프라를 배치 및 운용할 수 있다.

CB-Tumblebug에서 관리하는 주요 대상은 다음

과 같다.

- 멀티 클라우드 인프라 (Multi-Cloud Infrastructure, 이하 MCI) : 지역적으로 격리된 다수의 클라우드 상에서 단일 목적(응용서비스, 애플리케이션 등)을 위해 상호 연계된 하나 이상의 클라우드 인프라 서비스(VM, 컨테이너 등) 그룹
- 멀티 클라우드 인프라 자원 (Multi-Cloud Infrastructure Resource, 이하 MCIR) : 다수의 클라우드 상에서 제공되는 MCI를 생성 및 운용하기 위한 클라우드 인프라 자원들(VM 이미지, VM 사양, 네트워크, VM 접속을 위한 자원 등)

CB-Tumblebug는 크게 3단계로 기능들을 구분할 수 있다. 1단계는 멀티 클라우드 인프라 배치 계획 단계이고, 2단계는 멀티 클라우드 인프라 배치 수행 단계, 3단계는 멀티 클라우드 인프라 운용 및 관리 자동화 단계이다. CB-Tumblebug는 멀티 클라우드 인프라 배치 계획을 위해서 클라우드 서비스 제공자(Cloud Service Provider, 이하 CSP)별로 제공하는 VM 사양에 대한 성능 평가 기능을 제공하고, 해당 성능 평가를 기반으로 MCI를 최적으로 구성하고 스케줄링한다. 멀티 클라우드 인프라 배치 수행 단계에서는 계획된 MCI 구성을 실제로 인스턴스화하기 위해서, MCIR을 생성 및 관리하고 CB-Spider를 통해서 MCI에 포함된 개별 VM을 생성하는 기능을 제공한다. 그리고 개별 자원들을 MCI라는 논리적인 객체로 구성하여 저장하는 기능을 제공한다. 최종적으로, 멀티 클라우드 인프라 운용 및 관리 자동화 단계에서는 생성된 MCI의 라이프사이클을 관리하고 사용자가 지정한 정책에 따라 자동적으로 제어하는 기능을 제공한다.

3.4.2 구조 및 상세 기능

CB-Tumblebug의 기능적 구조는 Fig. 8에서 보는 바와 같이 6개의 주요 블록으로 구성되며, 세부 내용은 다음과 같다.

(1) 사용자 인터페이스(User Interface) 블록

사용자에게 HTTP 기반 REST 사용자 API 및 gRPC 기반 사용자 API를 제공한다.

(2) 네임스페이스 관리(Namespace Management) 블록

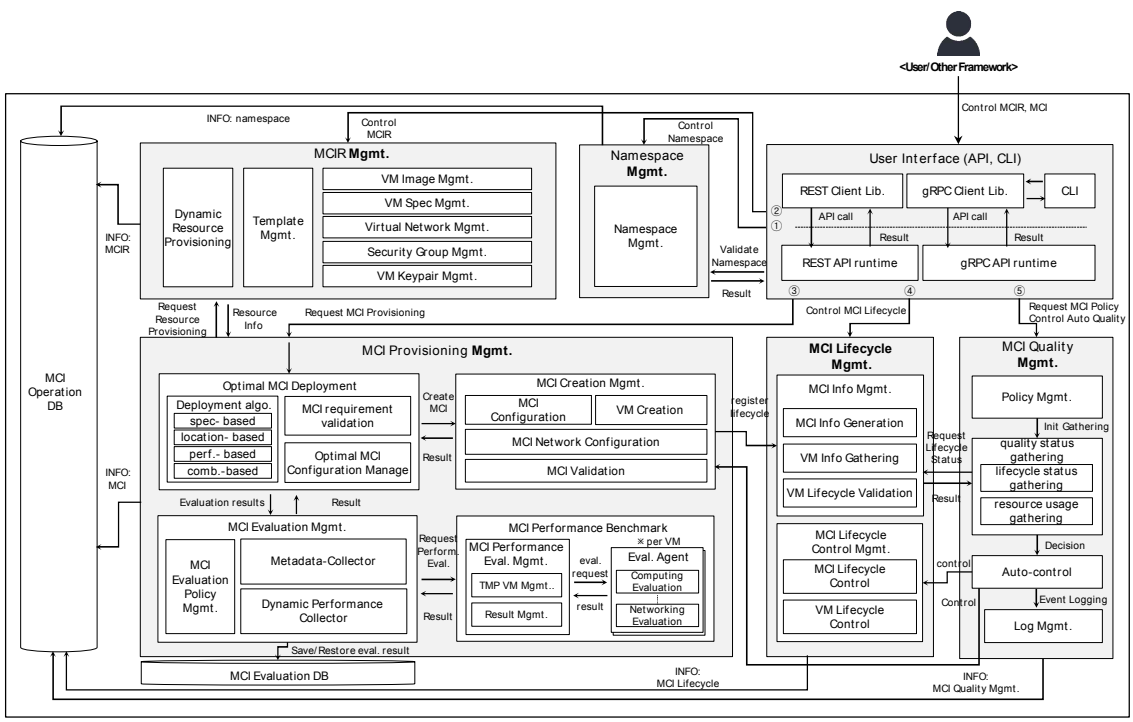

Fig. 8. The Functional Architecture of Multi-Cloud Infrastructure Service Management Framework.

네임스페이스의 생성, 조회, 검토, 수정, 삭제 등의 관리 기능을 제공한다. 네임스페이스는 CB-Tumblebug가 처리하는 객체를 논리적으로 격리하기 위한 개념이다. 예를 들어 CB-Tumblebug의 사용자가 2개의 독립적인 조직인 경우, 서로 각자가 운용하는 객체들이 노출되지 않도록 하고자 할 때 CB-Tumblebug의 네임스페이스 개념을 통해서 사용자별 자원 객체들을 격리할 수 있도록 기능을 지원한다. 예를 들어 A 회사에서 Cloud-Barista를 사용하는 경우, 인프라 관리 조직에서 단일 Cloud-Barista 계정을 개발 조직과 서비스 조직에 알려 주면, 각 조직은 Cloud-Barista의 논리적 자원 격리 단위인 네임스페이스를 필요에 따라 1개 또는 여러 개 생성하여 사용한다.

CB-Tumblebug이 처리하는 객체의 논리적 계층 구조는 다음과 같다. 먼저 최상위에 네임스페이스가 존재하며, 네임스페이스의 하위에는 멀티 클라우드 인프라(MCI)와 멀티 클라우드 인프라 자원 (MCIR)이라는 객체가 존재한다. MCI는 서로 다른 클라우드에 있는 여러 개의 VM으로 구성될 수 있고, MCIR은 이 VM들을 만들 때 필요한 자원들에 매핑되는 객체이다. 네임스페이스는 논리적 격리를 제공하기 때문에, 서로 다른 네임스페이스에 속하는 MCI, VM 및 MCIR은 ID가 동일하더라도 서로 다른 객체로 구분된다.

(3) 멀티 클라우드 자원 관리(MCIR Management) 블록

MCI에 속하는 VM을 만들기 위해서는 먼저 그에 필요한 MCIR의 생성이 필요하다. CSP를 통해서 VM을 생성하기 위해서는 해당 VM이 어떤 가상 네트워크와 서브넷을 사용할지, 어떤 보안 그룹을 사용할지, 어떤 SSH 키페어를 사용할지가 명시되어야 하며, 이를 위해 사용자는 MCIR 생성 기능을 이용해 CSP에 해당 자원을 생성함과 동시에 CB-Tumblebug에서 관리할 수 있도록 Tumblebug 객체로 등록한다. 또한, CSP에 VM을 생성할 때 어떤 OS 이미지와 VM 사양을 사용할지도 지정해 주어야 하므로, 사용자가 CB-Tumblebug의 이미지/스펙 등록 기능을 이용해 해당 이미지/스펙의 정보를 CB-Tumblebug에 등록해 놓고 MCI/VM 생성시 이 객체를 활용한다.

(4) MCI 프로비저닝 관리 (MCI Provisioning Management) 블록

임의의 VM은 하나의 MCI에 속하도록 (즉, 어느 MCI에도 속하지 않는 VM은 없고, 하나의 VM이 2개 이상의 MCI에 속하는 경우도 없도록) 정책을 정하였다. 이와 같이 VM은 MCI에 종속되는 논리적 구조이기 때문에, 사용자는 1) MCI를 생성할 때 구성 VM도 함께 생성하거나 2) 생성되어 있는 MCI에 VM을 추가해야 한다.

CB-Tumblebug에 MCI 생성 요청이 들어오면, 프로비저닝 관리기가 현재 등록되어 있는 MCIR 정보를 조회하고, 이를 이용하여 CB-Spider에 VM 생성 요청을 보낸다. CB-Spider에서 VM 생성 성공 응답을 보내면, CB-Tumblebug은 이 응답을 바탕으로 MCI/VM 객체를 업데이트하고 사용자에게 결과를 반환한다. 생성된 MCI 객체는 포함된 VM에 접속 가능한 SSH Key를 정보를 제공하므로 사용자는 이를 활용하여 MCI의 각 VM에 접속할 수 있으며, 필요에 따라서 MCI 내부의 모든 VM들에 원격 명령을 통해 사용자가 원하는 명령을 수행할 수 있다.

(5) MCI 라이프사이클 관리(MCI Lifecycle Management) 블록

생성된 MCI는 기본적으로 라이프사이클(예를 들면, 생성, 동작, 재시작, 종료 등에 대한 상태)에 대한 관리가 필요하다. 기존에는 여러 클라우드에 존재하는 VM들의 라이프사이클을 관리하기 위해 각 클라우드의 콘솔/CLI/API/SDK 등을 이용해 개별적으로 제어가 필요하였다. 특히 멀티 클라우드 환경에서는 CSP별로 인터페이스가 다르기 때문에 이를 통합적으로 확인하고 제어하기가 쉽지 않다. 그러나 CB-Tumblebug을 통하면, 다수의 클라우드에 존재하는 VM들을 통합적으로 관리할 수 있는 MCI 객체를 제공하기 때문에, 각 VM의 라이프사이클 확인 및 제어뿐만 아니라 MCI에 포함된 모든 VM의 라이프사이클을 통합적으로 조회하고 제어하는 기능을 제공한다.

CSP마다 VM 액션/상태 표기 방식이 약간씩 상이하여, CB-Tumblebug에서는 Fig. 9와 같이, VM에 대한 액션을 5가지(Create, Terminate, Suspend, Resume, Reboot), 상태를 9가지(Running, Suspended, Failed, Terminated, Creating, Suspending, Resuming, Rebooting, Terminating)로 정의하였다.

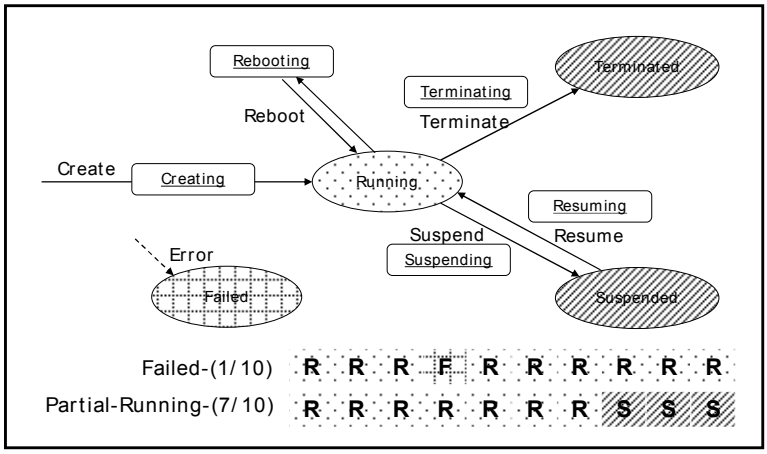

Fig. 9. The MCI Lifecycle Transition Graph.

또한, 여러 VM들을 포함하는 MCI의 상태도 정의하기 위해, VM 중 일부가 특정 상태인 경우 'Partial-Suspended' 와 같이 표시하도록 하였다.

(6) MCI 품질 자동 관리(MCI Quality Management) 블록

Cloud-Barista가 멀티 클라우드를 지원하므로, 사용자는 조건과 상황에 따라 CSP를 자유롭게 선택할 수 있다. 여기에서 나아가, 특정 클라우드에서 장애가 발생하거나, 더 저렴한 다른 클라우드가 있거나, 특정 클라우드 간의 네트워크가 느려지는 등의 상황이 발생하는 경우 이를 감지하고 자동으로 다른 클라우드를 이용하는 시나리오를 고려해 볼 수 있다.

CB-Tumblebug의 멀티 클라우드 인프라 서비스 자동 제어기는 Cloud-Barista의 모니터링 프레임워크인 CB-Dragonfly로부터 MCI 모니터링 정보를 수집하여, 지표에 특이사항이 있을 경우 정책을 기반으로 MCI에 속한 VM을 자동적으로 제어한다.

3.4.3 주요 설계 및 구현 이슈

최적의 멀티 클라우드 인프라 구성을 위해서 클라우드별로 제공하는 인프라 서비스의 특성을 정확하게 파악할 필요가 있다. 특히 클라우드의 데이터센터들은 세계적으로 광범위하게 퍼져 있으므로 클라우드 서비스 접근에 소요되는 시간(latency)이 서로 상이하다. 또한, CSP별로 제공하는 컴퓨팅 서비스의 종류(예: GPU 제공 서비스, 고성능 네트워크 속도 제공 서비스 등)와 성능(예: 계산 성능, 메모리 속도, DB 처리 속도 등)이 다르기 때문에 사용자의 요구사항에 적합한 클라우드 서비스를 선정할 필요가 있다.

더불어 멀티 클라우드 인프라 서비스는 여러 개별 서비스의 조합으로 구성되므로 개별 클라우드에서 지원하지 못하는 서비스간 연계 기술을 제공할 필요가 있다. 예를 들어, 아마존 클라우드(이하, AWS)에서 제공하는 VM과 구글 클라우드(이하, GCP)에서 제공하는 VM을 하나의 멀티 클라우드 인프라 서비스로 조합하는 경우, AWS의 VM과 GCP의 VM 사이의 통신은 보안이 구성되지 않은 공개 네트워크를 사용하게 된다. 이를 보완하기 위해서는 클라우드간 VM에 보안을 위한 VPN(Virtual Private Network) 기술 등을 접목하여 제공할 필요가 있다.

이와 같이, 전세계에 걸쳐 퍼져있는 다양한 클라우드의 형상과 특징을 고려하여 사용자에게 하나의 멀티 클라우드 인프라를 제공하기 위한 방법을 새롭게 고안할 필요가 있다.

3.5 멀티 클라우드 애플리케이션 통합 관리 프레임워크 (CB-Ladybug)

3.5.1 개요 및 기능

사용자는 앞에서 설명한 CB-Tumblebug를 통해서 전 세계에 배치된 최적의 멀티 클라우드 인프라를 활용하여 자신의 애플리케이션을 손쉽게 배치하고 효과적으로 운용 및 관리할 수 있는 방법이 필요하다.

멀티 클라우드 애플리케이션 통합 관리 프레임워크(CB-Ladybug)는 사용자에게 컨테이너화된 애플

리케이션들을 활용하여 멀티 클라우드 애플리케이션을 구성하도록 지원하고 실행 요구사항을 전달받아, 사용자 요구를 충족하는 최적의 멀티 클라우드 인프라 서비스를 요청하여 할당받고 사용자의 멀티 클라우드 애플리케이션을 멀티 클라우드 인프라상에 배치/실행하고 통합적으로 제어하고 자동적으로 관리해 주는 기능을 제공한다.

CB-Ladybug에서 관리하는 주요 대상은 다음과 같다.

- 멀티 클라우드 애플리케이션 (Multi-Cloud Application, 이하 MC-App) : 클라우드 네이티브 방식을 기반으로[14], 멀티 클라우드 인프라상에서의 운용을 위해 생성, 배포, 실행되는 애플리케이션
- 멀티 클라우드 애플리케이션 실행환경 (Multi-Cloud Application Runtime, 이하 MCAR) : 멀티 클라우드 애플리케이션을 구성하는 컨테이너화된 애플리케이션들을 효과적으로 배치/실행하고 통합적인 제어와 관리를 지원하는 컨테이너 오케스트레이션 엔진을 의미하며, 대표적으로 쿠버네티스와 도커 스웜(Docker Swarm), 그리고 아파치 메소스(Apache Mesos)가 해당함

CB-Ladybug는 크게 3단계로 기능들을 구분할 수 있다. 1단계는 MC-App을 구성하고 실행 요구사항을 등록하는 단계이고, 2단계는 MCAR을 생성하고 해당 MC-App을 배치하는 단계이며, 3단계는 MC-App의 운용 및 관리 자동화 단계이다. CB-Ladybug는 사용자로부터 MC-App의 구성 정보와 단위 App들의 실행선후관계나 요구 기능/성능, 또는 실행 제약 조건 등의 실행 요구사항을 입력받고, CB-Tumblebug 서브시스템에 이를 충족시킬 수 있는 MCI의 생성을 요청하여 할당받고 MCAR을 설치한 후 해당 MC-App을 프로비저닝하여 실행하고, 사용자가 지정한 품질 정책에 따라 자동적으로 제어할 수 있는 기능을 제공한다.

3.5.2 구조 및 상세 기능

CB-Ladybug의 기능적 구조는 Fig. 10에서 보는 바와 같이 5개의 주요 블록으로 구성되며, 세부 내용은 다음과 같다.

(1) 사용자 인터페이스(User Interface) 블록

사용자에게 HTTP 기반 REST 사용자 API 및 gRPC 기반 사용자 API를 제공한다.

(2) MC-App 구성 관리(MC-App Configuration Management) 블록

사용자가 컨테이너화된 단위 애플리케이션(이하

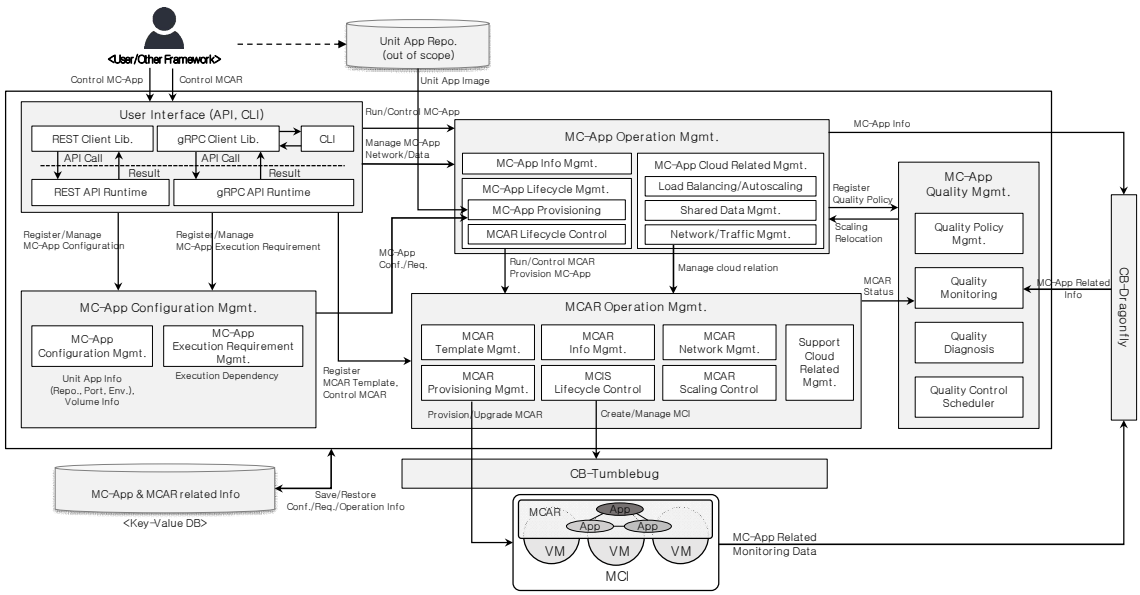

Fig. 10. The Functional Architecture of Multi-Cloud Application Management Framework.

단위 App)들로 구성한 MC-App을 등록/관리하고 MC-App의 실행 요구사항을 등록/관리하는 역할을 담당한다. MC-App을 구성하는 단위 App들은 도커 허브 저장소와 같은 별도의 저장소(단위 App 저장소)에 저장/관리되며, MC-App의 실행 요청이 발생하였을 때 해당 단위 App의 컨테이너 이미지들이 다운로드되어 실행된다. MC-App의 구성 정보는 단위 App 정보(저장소 위치와 파일명, 접속 포트 번호, 환경변수 값 등)나 볼륨 정보 등이 해당되며, MC-App의 실행 요구사항은 단위 App들간의 실행 선후관계나 요구 기능 및 성능, 실행 제약 조건 등이 해당된다.

(3) MC-App 운영 관리(MC-App Operation Management) 블록

사용자가 실행 요청한 MC-App의 요구 기능 및 성능, 실행 제약 조건 등에 적합하도록 MCAR을 생성한 후 해당 MC-App을 프로비저닝하고 운영 관리하는 역할을 담당한다. 또한 클라우드간 효과적인 연계 운영을 위해 품질 자동 제어기의 요청에 따른 오토스케일링이나 단위 App 컨테이너들의 동적 재배치, 공유데이터의 관리, 네트워크/트래픽 제어관리 등의 기능을 수행한다.

(4) MCAR 운영 관리(MCAR Operation Management) 블록

MC-App의 요구 성능 및 기능을 제공하기 위해 CB-Tumblebug에 적합한 MCI의 생성을 요청하여 할당받고, 해당 MCI에 컨테이너 오케스트레이션 엔진의 프로비저닝을 수행한다. 생성된 MCAR을 대상으로 동일 서브넷 구성 등을 위한 네트워크 관리 기능을 제공하며 MCAR에 대한 오토스케일링 요청에 대한 스케일링 관리를 담당한다. 또한 클라우드간 연계 운영을 위한 기반 기능을 제공한다.

(5) MC-App 품질 자동 제어(MC-App Quality Management) 블록

MC-App의 품질 관리를 위해 MC-App의 품질 상태를 모니터링하고 사용자가 요구한 품질 제어 정책에 적합하게 동작하는지 MC-App의 품질을 진단한다. 진단 결과에 따라 MCAR 차원의 오토스케일링의 수행을 요청하거나 단위 App 컨테이너들의 동적 재배치 수행을 요청하여 MC-App의 종합적인 품질을 유지시킨다.

3.5.3 주요 설계 및 구현 이슈

CB-Ladybug에서 활용 가능한 MCAR의 대표적인 구현체인 쿠버네티스는 동일 서브넷 환경에서 효과적으로 동작하도록 설계 및 구현되었기 때문에 멀티 클라우드 환경에서 그대로 활용하기에는 적합하지 않다. 쿠버네티스 커뮤니티는 최근 이러한 요구를 반영하여 멀티 클라우드 환경에서도 효과적으로 사용하기 위한 다양한 방안들이 논의되고 있다.

CB-Ladybug는 CB-Tumblebug을 통해서 할당받은 MCI를 대상으로 현재 널리 사용되고 있는 컨테이너 오케스트레이션 엔진인 쿠버네티스를 프로비저닝한 후 이를 기반으로 MC-App을 프로비저닝하고 운용/관리한다. MCI가 각종 CSP로부터 할당받은 VM들로 구성될 수 있기 때문에, 이들은 기본적으로는 서로 공개 IP망을 기반으로 통신할 수 밖에 없다. 물론 동일 CSP의 리전에 있는 VM과는 사설 IP망으로 통신하도록 하여 네트워크 성능 저하가 발생하지 않도록 해야 한다. 따라서 공개 IP망을 기반으로 오버레이 네트워크 구성하면서도 사설 IP망을 활용할 수 있는 방안, 그리고 보안성 제공을 위한 방안 등을 접목하여 함께 고려할 필요가 있다.

또한 구성된 MCI 내 각 VM간 네트워크 지연 시간이 서로 상이하므로, 이러한 사항을 고려하여 쿠버네티스의 컨트롤 플레인 노드와 워커 노드들 간의 배치뿐만 아니라 워커 노드들에서 운용되는 단위 애플리케이션들 간의 효과적인 배치 방안이 수립되어야 한다.

한편, 멀티 클라우드 환경을 고려하지 않고 있는 설계 및 구현된 컨네이터 오케스트레이션 엔진들을 대상으로, MC-App 구성과 실행 요구사항에 관한 규격을 기준으로 하는 MC-App의 효과적인 배치 및 실행 방안도 수립되어야 한다.

3.6 멀티 클라우드 통합 모니터링 프레임워크 (CB-Dragonfly)

3.6.1 개요 및 기능

다양한 클라우드상에 대규모로 배치되어 운용되는 멀티 클라우드 인프라 서비스들뿐만 아니라, 해당 멀티 클라우드 인프라에 배치되어 운용되는 수많은 애플리케이션들의 상태와 성능 정보를 수집, 저장,

관리하는 기술이 요구된다. 멀티 클라우드 통합 모니터링 프레임워크(이하 CB-Dragonfly)는 전세계 다양한 지역에서 제공되는 수많은 클라우드 서비스들을 대상으로 대규모 모니터링 데이터의 수집과 처리, 그리고 지리적 차이로 인해 발생하는 지연을 극복할 수 있는 안정적인 모니터링 성능을 보장하고 멀티 클라우드에 특화된 신규 모니터링 항목 제공을 위한 확장성 있는 구조와 방식을 제공한다[13]. 또한 주기적인 모니터링과 실시간 모니터링을 지원하기 위하여 푸쉬(PUSH) 방식과 풀(PULL) 방식의 모니터링 데이터 수집 기술을 제공한다.

CB-Dragonfly에서 관리하는 주요 대상은 다음과 같다.

- 가상 머신(VM)/컨테이너 모니터링 데이터 : 가상 머신과 컨테이너 단위의 모니터링 메트릭(Metric)에 대한 측정 결과
- 멀티 클라우드 인프라(MCI) 모니터링 데이터 : CB-Tumblebug에서 제공하는 MCI 단위의 모니터링 메트릭에 대한 측정 결과
- 멀티 클라우드 애플리케이션(MC-App) 모니터링 데이터 : CB-Ladybug에서 제공하는 MC-App 단위의 모니터링 메트릭에 대한 측정 결과
- 요구 기반 (On-Demand) 모니터링 데이터 : 필요한 시점에 특정 모니터링 메트릭에 대한 측정 결과

3.6.2 구조 및 상세 기능

CB-Dragonfly의 기능적 구조는 Fig. 11에서 보는 바와 같이 9개의 블록으로 구성되며, 세부 내용은 다음과 같다.

(1) 사용자 인터페이스(User Interface) 블록

사용자에게 HTTP 기반 REST 사용자 API 및 gRPC 기반 사용자 API를 제공한다.

(2) 모니터(Monitor) 블록

사용자의 요청에 따라 가상 머신(VM) 또는 컨테이너의 모니터링 데이터, MCI와 관련한 모니터링 데이터, MC-App과 관련한 모니터링 데이터, 그리고 온디맨드 모니터링 데이터를 제공한다.

(3) 모니터링 데이터 관리기(Monitoring Data Manager) 블록

시계열 데이터베이스(Time Series Database)를 관리하며, 모니터링 데이터를 저장하고 탐색하는 기능을 제공한다.

(4) 모니터링 수집기(Monitoring Collector) 블록

데이터 수집 주기에 따라 모니터링 에이전트로부터 전달받은 데이터를 가공하여 시계열 데이터베이스에 저장한다

(5) 모니터링 수집기 관리기(Monitoring Collector Manager) 블록

모니터링 수집기의 부하분산을 목적으로, 부하분산 정책에 따라 모니터링 수집기의 스케일-인/아웃(Scale-in/out)을 수행하거나 메시지 브로커의 토픽을 관리한다.

(6) 모니터링 정책 관리기(Monitoring Policy Manager) 블록

모니터링 수집기의 확장 정책, 모니터링 수집기의 데이터 가공 주기 정책, 에이전트의 데이터 수집 정책 등을 관리한다.

(7) 모니터링 데이터 부하분산 관리기(Load Balance (LB) Manager) 블록

주기적인 모니터링 데이터의 경우 수많은 에이전트가 전달하는 모니터링 데이터를 안정적으로 처리하기 위해 적합한 모니터링 수집기에 분산하여 전달되도록 메시지 브로커를 관리한다.

(8) 모니터링 에이전트(Monitoring Agent) 블록

각 VM 또는 컨테이너에 설치되며, VM이나 컨테이너, 그리고 멀티 클라우드 애플리케이션과 관련한 모니터링 데이터를 수집하여 모니터링 수집기에 푸쉬 또는 풀 방식으로 전달하거나 온디맨드 모니터에게 풀 방식으로 전달한다.

(9) 모니터링 알람 관리기(Monitoring Alarm Manager) 블록

사용자가 설정한 모니터링 알람 정책에 따라 성능 임계치 등을 진단하여 이메일, 슬랙(Slack) 등 다양한 방법으로 알람을 전달한다.

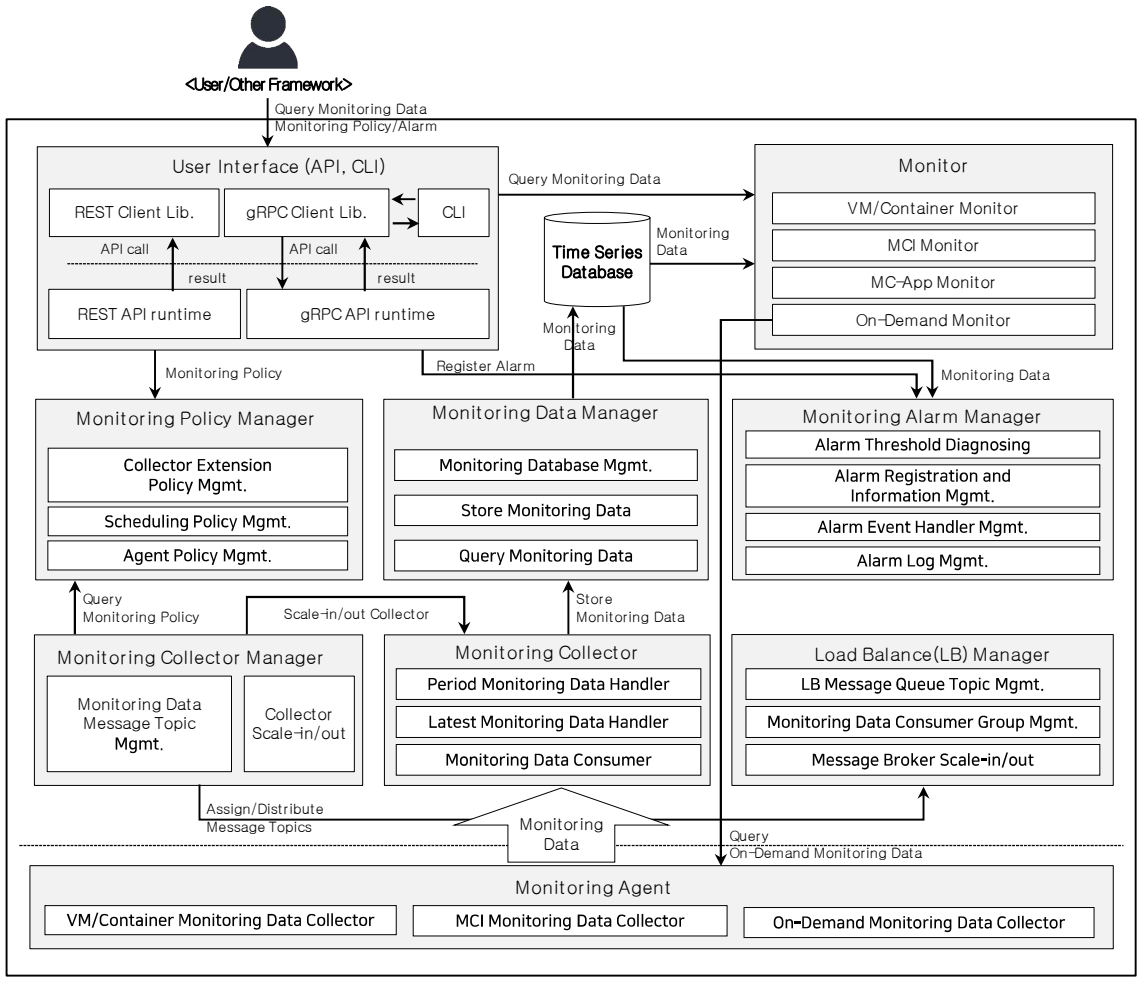

Fig. 11. The Functional Architecture of Multi-Cloud Service Monitoring Framework.

3.6.3 주요 설계 및 구현 이슈

이종 클라우드 서비스 제공자가 제공하는 서로 다른 모니터링 메트릭으로 인해 멀티 클라우드 서비스에 대한 일관된 통합 모니터링이 어렵다. 따라서 CB-Dragonfly는 에이전트 기반의 멀티 클라우드 통합 모니터링 구조를 활용하여 서로 다른 클라우드 서비스에 대해 통합 모니터링 메트릭과 보관 정책을 적용함으로써 일관된 모니터링을 가능하도록 설계하였다.

또한 수많은 모니터링 에이전트로부터 전달되는 대규모 모니터링 데이터를 부하분산 처리를 하고, VM 개수, 클라우드 서비스 제공자 종류, 지역 등과 같이 확장 정책 기준에 따라 전달받은 모니터링 데이터를 처리하는 모니터링 수집기의 확장성을 제공함으로써, 대규모 멀티 클라우드 서비스 환경에서도 안정적인 모니터링 기능을 제공하도록 설계하였다.

그리고 상용 퍼블릭 클라우드 서비스 제공자의 모니터링은 AWS의 경우 5분 주기는 무료로 1분 주기는 유료로 제공하는 등 최신의 모니터링 정보를 수집하기 어렵고, 보존 기간도 클라우드 서비스 제공자별로 서로 상이하다. CB-Dragonfly는 2초 단위로 모니터링 수집하여 10초 단위로 시계열 데이터베이스에 저장할 수 있으며, 수집 주기는 사용자가 설정이 가능하다. 또한, 실시간 모니터링 또는 온디맨드 모니터링 데이터 수집을 위해 풀 방식을 지원함으로써 사용자는 가장 최신의 모니터링 데이터를 수집할 수 있도록 설계하였다.

3.7 멀티 클라우드 서비스 운용 관리 도구

3.7.1 개요 및 기능

멀티 클라우드 서비스 운용 관리 도구(이하 cb-operator)는 Fig. 12와 같이 앞서 설명한 각 프레임워크들과 추가로 필요한 구성 요소들을 컨테이너 형태로 구동하여 전체 플랫폼을 실행하고, 각 프레임워크의 실행 상태 및 모니터링 정보 제공 등의 운용 관리 기능을 제공하여 Cloud-Barista의 사용 편의성을 증대시키는 역할을 담당한다[13].

Cloud-Barista를 실행하기 위해서는 CB-Spider, CB-Tumblebug, CB-Ladybug, CB-Dragonfly 등 주요 프레임워크들을 실행해야 하는데, 각 프레임워크의 소스코드 다운로드부터 빌드, 환경설정과 실행 등 단계가 복잡하고 난해하며 또 다른 구성 요소(예, CB-Dragonfly 동작을 위해 InfluxDB 필요)와의 의존성 이슈도 있다.

이러한 실행 과정의 어려움을 줄이기 위해, 각 프레임워크들과 추가 구성 요소들을 주기적으로 컨테이너화하여 해당 컨테이너 이미지를 공개 컨테이너 이미지 저장소인 도커 허브(Docker Hub)에 공개하고 있으며[15], cb-operator를 통해 Cloud-Barista를 처음 접하는 사용자들이 손쉽게 Cloud-Barista를 실행해 볼 수 있도록 지원한다. cb-operator는 도커 컴포즈(Docker Compose) 또는 쿠버네티스(Kubernetes)를 기반으로 동작하며, 컨테이너 이미지화되어 있는 각 프레임워크들과 추가적인 구성 요소들을 적절한 환경설정과 함께 동작시켜 Cloud-Barista를 실행하거나 중지하는 등 전반적인 라이프사이클을 관리할 수 있다.

3.7.2 구조 및 상세 기능

cb-operator는 Fig. 13에서 보는 바와 같이 3개의 주요 블록으로 구성되며, 세부 내용은 다음과 같다.

(1) 플랫폼 실행 관리 (Execution Management) 블록

사용자는 Cloud-Barista의 실행을 명령하면 로컬 이미지 저장소에 Cloud-Barista 구성 요소 컨테이너 이미지가 있는지 확인하고, 존재하지 않으면 공개 이

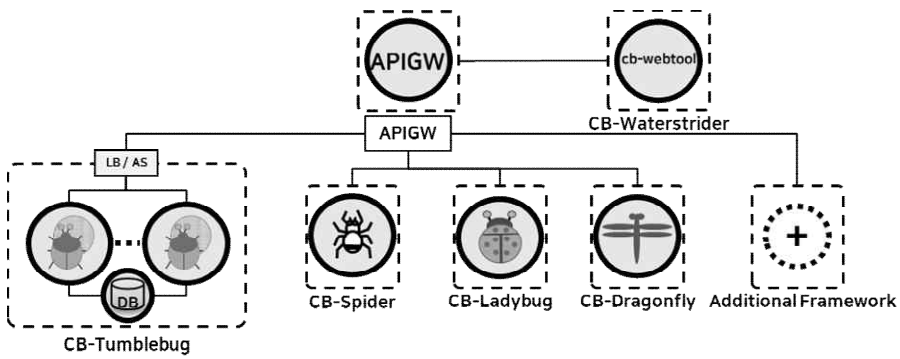

Fig. 12. A Deployment Concept of Cloud-Barista.

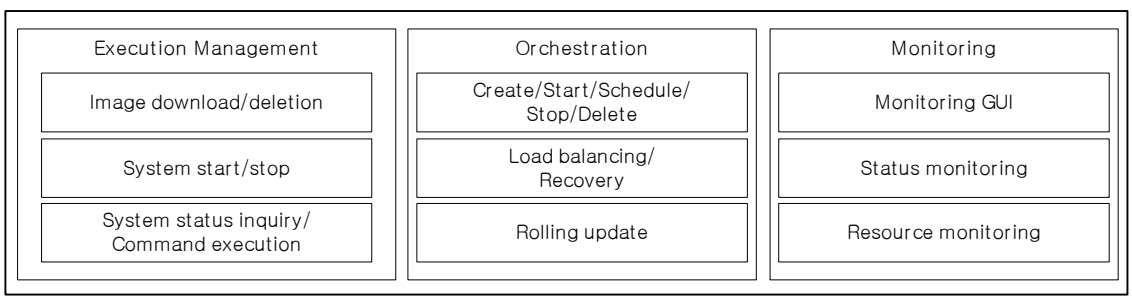

Fig. 13. The Functional Architecture of cb-operator.

미지 저장소에서 로컬 이미지 저장소로 이미지를 다운로드한 후 Cloud-Barista의 각 구성 요소들을 실행하여 서비스를 수행한다. 그리고 Cloud-Barista가 실행된 상태에서 상태 조회 기능을 통해 현재 실행되어 있는 Cloud-Barista 구성 요소 컨테이너들의 상태와 로컬 이미지 저장소에 있는 컨테이너 이미지들의 상태를 확인할 수 있다. 또한 실행 중인 각 프레임워크 컨테이너에 접속하여 명령을 내리거나 모든 프레임워크 컨테이너를 중지하거나 삭제하는 등의 기능을 제공한다.

(2) 프레임워크 오케스트레이션 (Orchestration) 블록

각 프레임워크를 도커 컴포즈나 쿠버네티스 기반으로 실행할 수 있도록 지원한다. 사용자가 Cloud-Barista의 프레임워크를 직접 개발하거나 간단한 시험 등을 목적으로 단일 노드에서 Cloud-Barista를 실행시키고자 할 때는 도커 컴포즈 기반 동작 모드를 이용하는 것이 유용하고, 제품 수준의 서비스를 제공하고자 하는 경우 쿠버네티스 기반 동작 모드를 선택하여 각 프레임워크 컨테이너들에 대해 부하 분산이나 장애 복구, 롤링 업데이트 등의 기능을 활용할 수 있다.

(3) 모니터링 (Monitoring) 블록

도커 컴포즈나 쿠버네티스와 연동하여 실행할 수 있는 프로메테우스(Prometheus)[16], 그라파나(Grafana)[17], 예거(Jaeger)[18] 등의 도구들을 통해 복합적인 모니터링 기능을 사용할 수 있다.

3.7.3 주요 설계 및 구현 이슈

Cloud-Barista의 구동을 위해서는 주요 프레임워크인 CB-Spider, CB-Tumblebug, CB-Ladybug, CB-Dragonfly가 실행되어야 할 뿐 아니라 추가적인 구성 요소인 각종 데이터베이스(예, etcd, InfluxDB 등)와 사용 편의성 등을 위한 Cloud-Barista 웹 GUI 도구(cb-webtool)와 Cloud-Barista API 게이트웨이(cb-apigw) 등도 실행되어야 한다. 이러한 구성 요소들의 정상적인 실행을 위해서는 각 구성 요소에서 필요로 하는 쉘 환경변수들 뿐만 아니라 YAML 파일을 통해 환경설정값을 적합하게 설정해 주어야 한다.

cb-operator에서는 이러한 복잡한 부분을 보다 쉽게 처리하기 위해, 사용자가 cb-operator 소스코드를 깃허브로부터 복제하면 도커 컴포즈 기반 동작 모드 구동을 위한 docker-compose.yaml 파일과 쿠버네티스 기반 동작 모드 구동을 위한 헬름(Helm) 차트도 함께 다운로드되도록 하였고, 주요 프레임워크에서 필요로 하는 쉘 환경변수를 docker-compose.yaml 또는 쿠버네티스의 헬름 차트 안에서 설정 및 로드하도록 하였으며, YAML 파일은 볼륨 마운트를 통해 설정 및 로드할 수 있도록 하였다. 추가적인 구성 요소에 대해서는 docker-compose.yaml에는 다른 서비스 형태로, 헬름 차트에서는 서브차트의 형태로 명시하였다.

또한, 각 구성 요소가 실행되어 동작할 때 메타정보 등을 저장하는데, 이를 컨테이너 안에 저장하면 추후 컨테이너가 삭제될 시 메타정보도 함께 삭제되기 때문에, 이를 방지하기 위해 도커 컴포즈 기반 동작 모드에서는 호스트 컴퓨터의 특정 디렉토리를, 쿠버네티스 기반 동작 모드에서는 외부 저장소인 쿠버네티스의 퍼시스턴트볼륨(PersistentVolume)를 설정하여 컨테이너의 특정 디렉토리로 마운트하여 그곳에 메타정보가 저장되도록 하였다.

4. 플랫폼 개발 및 시험

4.1 개발 현황

Cloud-Barista는 Cloud-Barista 커뮤니티 주도로 깃헙(GitHub)에서 개발이 이루어지고 있으며, 각 프레임워크 단위로 구분하여 저장소를 개설하여 개발을 진행하고 있다[19]. 또한, 개발 시 필요한 논의는 깃헙 이슈와 슬랙[21]을 통해서 이루어지고 있다.

Cloud-Barista 커뮤니티에서는 결과물의 공개와 확산, 의견 청취, 커뮤니티 기여자 확대 등을 위해 Cloud-Barista 릴리즈 행사를 약 6개월 간격으로 개최하고 있으며, 2019년 11월 제1차 릴리즈 행사(ver 0.1.0-americano)를 시작으로 2020년 6월에 제2차 릴리즈 행사(ver0.2.0-cappuccino)를 개최하였으며, 2020년 11월 제3차 릴리즈 행사(ver0.3.0-espresso)를 개최하였다. 2021년 5월 제4차 릴리즈 행사를 계획하고 있다. 릴리즈 행사에서 사용된 발표자료나 동영상, 그리고 관련 기술 문서와 API 문서, 소스코드 등은 Cloud-Barista 커뮤니티 웹사이트에 게시되어 있어 누구나 활용할 수 있다[20].

Table 1. The Test Environment.

Type	Contents		
Cloud-Barista Version	ver0.2.0-cappuccino		
Test Cloud Service Provider & Region & The number of VMs	4 CSPs	10 Regions	18 VMs
	Amazon AWS	ap-us-east-1 (Virginia, US)	2
		ap-us-west-1 (California, US)	2
		ap-northeast-2 (Seoul, Korea)	2
	Microsoft Azure	koreacentral (Seoul, Korea)	2
		northcentralus (Wyoming, US)	2
		candaeas (Quebec City, Canada)	2
	Google GCP	asia-east1 (Taiwan)	2
		asia-east2 (Hongkong)	2
		europe-west3 (Frankfurt, Germany)	1
	ALIBABA	ap-northeast-1 (Tokyo, Japan)	1
Test Application	Server	NGINX Web Server on each VMs	
	Client	Google Chrome Web Browser	

4.2 시험 환경, 방법 및 결과

멀티 클라우드 환경에서 Cloud-Bairsta의 동작 확인을 위해서 다양한 위치의 클라우드 리전에 가상 머신(VM)을 생성하고 각 가상 머신이 정상적으로 동작하는지 확인하였다. 제2차 릴리즈 행사의 데모 세션에서는 Table 1과 같이 전 세계 10개 리전에 분산하여 총 18개의 VM을 생성하고 이를 하나의 MCI로 묶고, 해당 MCI에 웹 서버(NGINX)를 일괄 배포한 후 웹 브라이저를 통해 MCI에 속한 각 VM에서 동작하는 웹서버에 각각 접속하여 웹 서버와 VM이 정상적으로 동작하는지 확인하는 시험을 수행하였다.

각 리전에 VM들이 할당되어 동작하는지 확인하기 위해 Fig. 14와 같이 세계 지도에 VM의 위치를 표시하고 각 VM들을 포함하는 MCI를 다각형으로 시각화하여 직관적으로 동작 여부를 확인할 수 있도록 하였다. 그리고 MCI 단위로 생성(Create), 일시중지(Suspend), 재개(Resume), 재부팅(Reboot), 종료(Terminate) 명령을 내려, MCI 단위의 VM 라이프사이클 제어가 잘 동작함을 확인하였다. 그림에 표현된 문구는 '[MCI 식별자] 현재 상태-(상태 변경 완료 VM 수/전체 VM 수)'을 의미한다.

또한 멀티 클라우드 상에서만 발생할 수 있는 MCI 단위의 부분 상태 관리 기능과 MCI 품질 자동 관리를 위한 MCI 동적 성능 벤치마킹 기능을 개발하여 공개하였다. 동적 성능 벤치마킹 기능은 사용자가 최적의 MCI를 구성할 수 있도록 조건에 맞는 CSP별 VM 스펙을 벤치마킹을 통해 찾아 주는 기능으로, 시험용 VM들을 생성하여 벤치마킹 에이전트를 설치하고, 이 에이전트를 통해 CPU 성능(소수 계산 시간 등)과 네트워크 지연시간과 같은 개별 VM 및 VM들 간의 성능을 벤치마킹하고 그 결과를 취합한다. 사용자는 벤치마킹의 결과인 각 리전별 VM의 CPU 성능과 VM 및 클라우드 간 통신 성능 등을 고려하면서 VM을 배치하는데 활용하는 등 보다 효과적으로 MCI를 구성할 수 있다.

5. 결론 및 향후 과제

근래 융합 IT 서비스의 수요 증가는 응용 서비스

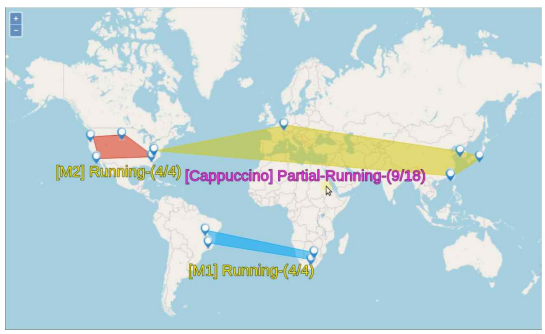

Fig. 14. The Demonstration Result of Cloud-Barista ver0.2.0-cappuccino.

들의 글로벌 스케일 확대를 야기시키고 있으며, 이는 응용 서비스들의 컴퓨팅 인프라를 멀티 클라우드 환경으로 이동시키고 있다. 그러나 단일 클라우드 인프라상에서의 운영에 최적화되어 있는 기존의 퍼블릭 클라우드들의 서로 다른 기능 및 인터페이스들은 이러한 멀티 클라우드 인프라 환경으로의 새로운 변화에 걸림돌이 되고 있다.

본 논문에서는 이와 같은 문제 해결을 위하여 전세계 (이종) 멀티 클라우드 인프라의 연동 및 공통 방식 제어 기술과 멀티 클라우드 인프라의 효과적인 배치 및 운용 기술, 그리고 이러한 멀티 클라우드 인프라 상에서 애플리케이션의 효과적인 배치 및 운용 기술과 대규모 멀티 클라우드 서비스에 대한 통합 모니터링 기술에 대한 설계 및 주요 구현 이슈를 소개하였다. 이를 통해 단일 클라우드의 기능 제약 및 자원 한계를 뛰어넘는 글로벌 스케일의 멀티 클라우드 인프라 환경에서 운용되는 서비스들을 손쉽게 개발 및 운영할 수 있을 것으로 기대된다.

앞으로는 국내외 다양한 클라우드 서비스 확대를 위해 클라우드 연동 드라이버들의 추가 개발과 클라우드 서비스별로 불규칙한 연동 성능 문제나 전체 VM 이미지 목록과 같은 대규모 정보 제공에 따른 성능 문제 등 이번 연구 과정에서 도출된 이슈들을 해결하기 위해 엣지 기술이나 분산 협업 기술을 접목할 계획이며, 멀티 클라우드 인프라의 최적 배치를 위한 효과적인 모니터링 메트릭을 추가 발굴하고, 플랫폼 운영자들의 효과적 운용을 지원하기 위한 멀티 클라우드 인프라 운영 자동화 기능을 개발할 계획에 있다. 그리고 멀티 클라우드 애플리케이션의 실행 관리 방식을 구체화하는 등 멀티 클라우드 애플리케이션의 효과적인 운용을 위한 추가 연구를 진행할 예정이다. 또한 정량적/정성적인 측면의 성능 비교를 통해 개선할 부분을 파악하여 보다 효과적인 방법에 대한 연구를 진행할 예정이다.

REFERENCE

[1] Multicloud. https://en.wikipedia.org/wiki/Multicloud (accessed Oct., 15, 2020).

[2] B.S. Kim, Y.W. Jung, B.T. Oh, S.Y. Kim, S. Son, J.H. Seo, et al., "Multi-cloud Technology Introduction and Research Trends," *Electronics and Telecommunications Trends*, Vol. 35, No. 3, pp. 45-54, 2020.

[3] Y.M. Lim, "The Collaborative Image Editing Tool based On the Cloud Computing," *Journal of Korea Multimedia Society*, Vol. 20, No. 8, pp. 1456-1463, 2017.

[4] D. Petcu, "Multi-Cloud: Expectations and Current Approaches," *Proceedings of the 2013 International Workshop on Multi-cloud applications and federated clouds*, pp. 1-6, 2013.

[5] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems," *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 887-894, 2013.

[6] Apache libCloud. https://libcloud.readthedocs.io/en/stable/ (accessed Oct., 15, 2020).

[7] Apache jCloud. https://jclouds.apache.org/guides/ (accessed Oct., 15, 2020).

[8] N. Goonasekera, A. Lonie, J. Taylor, and E. Afgan, "CloudBridge: A Simple Cross-cloud Python Library," *Proceedings of the XSEDE 16 Conference on Diversity, Big Data, and Science at Scale*, Article No. 37, pp. 1-8, 2016.

[9] Scalr CMP. https://cmp.docs.scalr.com/en/latest/ (accessed Oct., 15, 2020).

[10] Flexera RightScale. https://docs.rightscale.com/api/ (accessed Oct., 15, 2020).

[11] J. Han, S. Park, J. Kim, "Dynamic OverCloud: Realizing Microservices-Based IoT-Cloud Service Composition over Multiple Clouds," *Electronics*, 9, 969, 2020.

[12] Hashicorp Terraform. https://www.terraform.io/docs/index.html (accessed Oct., 15, 2020).

[13] System Design for Multi Cloud Service Common Framework (2019). https://github.com/cloud-barista/archive/blob/master/americano/docs/design/(Cloud-Barista)%EC%8B%9C%EC%8A%A4%ED%85%9C%EC%84%A4%EA%B3%84%EC%84%9C(v0.5)-2019-10-06.pdf (accessed Oct., 15, 2020).

[14] N. Kratzke, P.C. Quint, "Understanding Cloud-

native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study," *Journal of Systems and Software*. Vol. 126, pp. 1-16, 2017.
[15] Cloud-Barista Public Container Image Registry. https://hub.docker.com/u/cloudbaristaorg (accessed Oct., 15, 2020).
[16] Prometheus. https://prometheus.io/ (accessed Oct., 15, 2020).
[17] Grafana. https://grafana.com/ (accessed Oct., 15, 2020).
[18] Jaeger. https://www.jaegertracing.io/ (accessed Oct., 15, 2020).
[19] Cloud-Barista Repositories. https://github.com/cloud-barista (accessed Oct., 15, 2020).
[20] Cloud-Barista Community Website. https://cloud-barista.github.io/ (accessed Oct., 15, 2020).
[21] Cloud-Barista Slack Workspace. https://cloud-barista.slack.com/ (accessed Oct., 15, 2020).

김 수 영

2005년 한국과학기술원 전산학과 (석사)
2015년 충북대학교 정보통신공학 전공 (박사수료)
2005년～현재 한국전자통신연구원 클라우드기반SW연구실 선임연구원
2020년～현재 클라우드바리스타 CB-Ladybug 프레임워크 리더
관심분야 : 분산/고성능/클라우드 컴퓨팅, 운영체제, 시스템소프트웨어

김 병 섭

1999년 전북대학교 정보통신공학과 (석사)
1999년～현재 한국전자통신연구원 클라우드기반SW연구실 책임연구원
2019년～현재 클라우드바리스타 CB-Spider 프레임워크 리더
관심분야 : 클라우드 컴퓨팅, 고성능 컴퓨팅, 분산 컴퓨팅

손 석 호

2009년 광주과학기술원 정보통신공학과 (석사)
2012년 광주과학기술원 정보통신공학과 (박사)
2014년～현재 한국전자통신연구원 클라우드기반SW연구실 선임연구원
2019년～현재 클라우드바리스타 CB-Tumblebug 프레임워크 리더
2020년～현재 쿠버네티스 SIG-Docs 한글화팀 팀장
2020년～현재 한국전자통신연구원 오픈소스전문위원
관심분야 : 분산/고성능/클라우드 컴퓨팅, 스케줄링 메커니즘, 멀티 에이전트 시스템

서 지 훈

2014년 서울대학교 전기·컴퓨터공학부 (석사)
2014년～현재 한국전자통신연구원 클라우드기반SW연구실 연구원
2019년～현재 클라우드바리스타 CB-Bridge 프레임워크 리더
관심분야 : 분산/고성능/클라우드 컴퓨팅, 무선통신, 유선통신

김 윤 곤

2016년 경희대학교 컴퓨터공학과 (석사)
2020년 경희대학교 컴퓨터공학과 (박사)
2020년 경희대학교 컴퓨터공학과 연구박사
2020년～현재 한국전자통신연구원 클라우드기반SW연구실 연구원
2020년～현재 클라우드바리스타 CB-Larva 인큐베이터 리더
관심분야 : 클라우드 컴퓨팅, 분산 클라우드, 멀티 클라우드 네트워크, 소프트웨어 디자인 패턴

강 동 재

2010년 인하대학교대학원 정보공학과 (박사)
2011년～2018년 과학기술연합대학원대학교(UST) 겸임교수
2001년～현재 한국전자통신연구원 클라우드기반SW연구실 실장
2019년～현재 클라우드바리스타 커뮤니티 리더
관심분야 : 클라우드 컴퓨팅, 공개 소프트웨어, 시스템 소프트웨어