멀티 클라우드 서비스 연동을 위한 게이트웨이 시스템 개발

김바울*, 김승한*, 구원본*, 김명진*
*이노그리드 클라우드컴퓨팅연구센터
baul@innogrid.com, ksh1006@innogrid.com, wbkoo@innogrid.com,
tough105@innogrid.com

Development of gateway system for multi-cloud federation

Baul Kim*, SeungHan Kim*, Won BonKoo*, MyoungJin Kim*
*Cloud Computing R&D Center, Innogrid

요 약

멀티 클라우드는 다양한 클라우드 제공자들의 클라우드 서비스를 연동하여 사용자에게 제공하는 기술로 최근 4차 산업혁명 기술의 발전에 따라 대용량 데이터의 수용, 대용량 계산 등을 단일 클라우드에서 활용하기 어려워 데이터를 가까운 클라우드 자원을 활용하여 분석하거나 서비스를 가까운 지역에서 제공하기 위해 등장한 기술이다. 그러나 멀티 클라우드 제공처마다 동일한 기능을 서로 다른 인터페이스와 파라미터를 사용해야 하며, 시스템의 버전 증가에 따른 인터페이스 변화 시 멀티 클라우드 연동을 다시 해주어야 하는 번거로움이 있다. 본 논문에서는 멀티 클라우드마다 다른 인터페이스를 분석하고 공통 요소를 도출 및 규격화하여 연동을 위한 게이트웨이를 제안한다. 인터페이스 제공을 위해 필수적인 요소와 부가적인 요소를 정리하여 공통 규격으로 만들어 멀티 클라우드 시스템 구성 및 서비스 이용에 편의성을 향상시킬 수 있다. 또한, 멀티 클라우드 인터페이스를 마이크로 서비스로 구성하여 새로운 클라우드 서비스 등록과 기존 등록된 서비스의 인터페이스 버전 변화에도 대응 가능할 수 있도록 설계하여 손쉽게 연동 서비스를 이용할 수 있는 시스템을 제안한다.

1. 서론

최근 빅데이터·인공지능·IoT 등의 4차 산업혁명 관련 기술이 확산함에 따라 단일 클라우드 상에서의 계산, 데이터 수용 한계로 인해 다양한 클라우드를 동시에 활용할 수 있는 멀티 클라우드 기술에 대한 사용자 요구가 증가하였다.

멀티 클라우드 서비스는 클라우드 서비스 사업자들이 제공하는 다양한 클라우드 서비스를 등록·관리할 수 있어 사용자가 요구하는 서비스 사업자와 필요 자원을 제공해줄 수 있다. 사용자의 입장에서는 가상 인스턴스 생성과 같은 기능을 클라우드 사업자에 상관없이 동일하게 이용할 수 있는 장점이 있다. 하지만 클라우드 연동을 위해서는 클라우드 제공자마다 다른 인터페이스를 분석하여 연동 프레임워크를 개발해야 하며, 멀티 클라우드 연동 시스템을 구축하더라도 신규 클라우드 제공처를 추가하거나 기존에 등록돼 서비스의 인터페이스 버전이 변경될 경우 새로운 인터페이스 혹은 SDK를 활용하여 추가 개발 및 시스템 재구성을 위한 비용이 소요된다.

본 논문에서는 멀티 클라우드 연동 시스템 구축의 어려움을 해결하기 위한 멀티 클라우드 서비스 연동 게이트웨이 시스템을 제안한다. 멀티 클라우드 연동의 편의성을 향상 및 연동 기능 개발의 파편화를 보완하기 위해 공통 규격 및 응답 코드 도출과 클라우드 서비스 제공처 확장이 쉽도록 마이크로 서비스 아키텍처 형식으로 시스템을 구성하여 신규 서비스 추가 연동이 편리한 시스템을 제안한다.

본 논문의 2장에서는 멀티 클라우드와 관련된 주요 연구를 살펴보고, 3장에서는 멀티 클라우드 연동을 위한 게이트웨이 시스템을 제안하며, 4장에서는 개발결과를 평가하고, 5장에서는 결론을 내린다.

2. 관련 연구

2.1 멀티 클라우드 관련 기술 동향

클라우드 자원 연동을 위한 기술 중 Cascading 프로젝트는 오픈스택으로 구축된 클라우드의 자원을 중앙 집중적으로 제어하기 위한 기술을 개발하고 있고[1], 다중 Kubernetes 기반 서비스 운영을 위해

멀티 클라우드 환경에서 쿠버네티스 애플리케이션 컨테이너 자원들을 통합 관리 지원을 위한 다중 쿠버네티스 연동 기술을 개발하고 있다[2]. 또한, 이종 멀티 클라우드 서비스 브로커리지 및 클라우드 인프라 스토리지 연동을 위한 기술과 연구와 퍼블릭 클라우드 상에서 운영되는 애플리케이션들을 관리하는 연구가 있으며, 분산 애플리케이션을 배포 운영하기 위해 하이브리드나 멀티 클라우드로 구성된 클라우드 인프라를 관리하는 연구도 이루어지고 있다[3,4].

3. 본론

3.1 멀티 클라우드 제공 API 공통 요소 분석

본 연구에서는 멀티 클라우드 서비스를 연동하기 위해 각 퍼블릭 클라우드 제공처 2종과 프라이빗 클라우드 1종에 대한 인터페이스를 분석하였다.

표1은 가상 인스턴스 생성에 필요한 파라미터 정보를 비교하여 공통적인 요소와 서비스별 추가로 필요한 정보들을 정리한 표 일부이다. name과 같이 가상 인스턴스 생성에 필수적인 정보는 모든 서비스에서 입력받지만, region 등 일부 속성들은 특정 클라우드 서비스에서만 이용된다. 따라서 공통 요소들과 부가 요소를 분석하여 연동에 필요한 정의 규격화와 게이트웨이를 통해 처리할 수 있도록 모듈화하였다. 또한, 신규 클라우드 서비스를 연동할 경우 공통 규격에 대한 조건을 만족하고 이외에 정보는 부가적인 파라미터로 설정하여 마이크로 서비스로 등록하게 하였다.

<표 1> 멀티 클라우드 파라미터 비교(VM 생성)

속성	타입	클라우드		
		퍼블릭		프라이빗
		AWS	Azure	OpenStack
name	String	M	M	M
region	String	NP	M	NP
resourceGroupName	String	NP	M	NP
instanceCount	Int	M	NP	NP
flavorName	String	M	M	M
sourceType	String	NP	NP	M
volumeCreated	Boolean	NP	NP	M
imageType	String	NP	M	NP
osType	String	NP	M	NP
imageId	String	M	M	M
subnetName	String	M	M	NP
networkId	String	NP	M	M
securityGroupName	String	M	NP	M
publicIpType	String	NP	M	NP
publicIp	String	NP	M	NP
keypair	String	M	NP	M

(M: Mandatory, NP: Not Present)

3.2 멀티 클라우드 제공 응답 코드 구성

본 연구에서 개발한 멀티 클라우드 서비스에서 사용자에게 요청 메시지 수행에 대한 결과를 코드 값으로 반환한다. 응답 코드는 성공 응답 코드와 오류 응답 코드가 있으며, 성공 응답 코드와 오류 응답 코드는 name, type, title로 구분되어 있다. 멀티 클라우드 서비스의 인터페이스에 대하여 본 연구에서 구성 및 정의한 응답 코드를 따른다.

표2는 멀티 클라우드 서비스 사용자의 요청이 성공적으로 수행된 경우 반환하는 응답 코드를 정리한 표이다.

<표 2> 성공 응답 코드 구성

name	type	title
OK	200(OK)	Retrieve Successes
Created	201(Created)	Created Successes
Deleted	200(OK)	Deleted Successes
Changed	200(OK)	Changed Successes

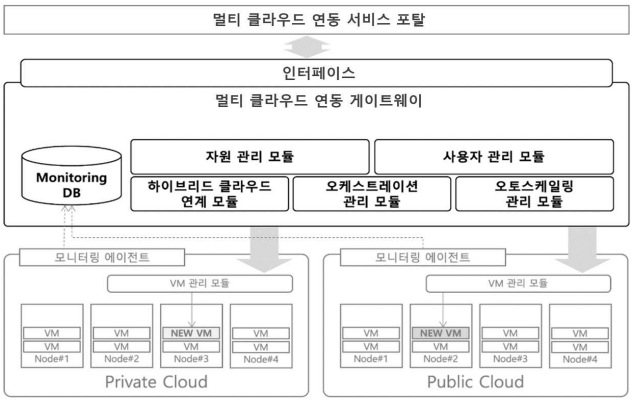
표3은 멀티 클라우드 서비스 사용자의 요청이 오류로 인하여 요청 메시지 수행이 실패한 경우 반환하는 응답 코드를 정리한 표이다.

<표 3> 오류 응답 코드 구성

name	type	title
InvalidRequest	400 (Bad Request)	Invalid Request
BadRequest Data	400 (Bad Request)	Bad Request Data
Unauthorized	401 (Unauthorized)	Unauthorized
MethodNot Allowed	405(Method Not Allowed)	Requested API is not defined
AlreadyExists	409(Conflict)	Already Exists
InternalError	500(Internal Server Error)	Internal Error

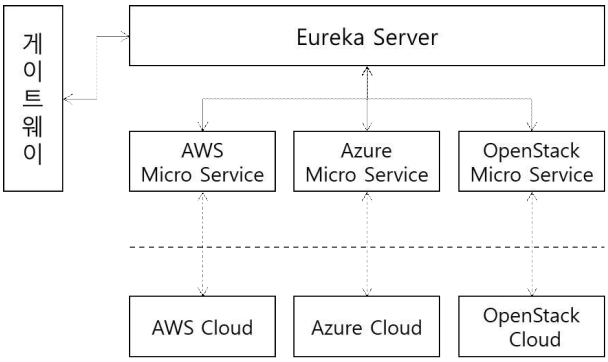
3.3 멀티 클라우드 연동 시스템 구성

본 연구에서 개발한 멀티 클라우드 서비스 연동 게이트웨이의 구조는 그림 1과 같다. 멀티 클라우드 연동 시스템은 멀티 클라우드 자원을 관리할 수 있는 모듈, 사용자 인증 관리 모듈, 하이브리드 클라우드 연계 모듈, 오케스트레이션 관리 모듈, 오토스케일링 관리 모듈이 있다.



(그림 1) 멀티 클라우드 연동 시스템 구성도

하이브리드 클라우드 연계 모듈은 마이크로 서비스 형식으로 설계하여 각각의 클라우드 서비스마다 연동할 마이크로 서비스를 구현하였다. 게이트웨이에서는 표 1과 같이 공통 규격을 통해 도출한 파라미터와 서비스 명세를 기반으로 호출하도록 구성한다. 그림 2와 같이 클라우드마다 연동을 위한 마이크로 서비스를 두고 이를 연계하여 서비스를 등록 관리할 수 있는 서버로 구성된다. 마이크로 서비스로 구성된 자원들을 동적으로 등록하고 관리할 수 있도록 Eureka Sever-Client를 활용하였다. 서비스 등록 관리를 위해 서비스의 이름, 설명, 버전 정보를 메타정보로 등록 관리하여 게이트웨이의 요청에 따라 필요한 서비스로 연결되는 구조이다.



(그림 2) 하이브리드 클라우드 연계 구조

4. 구현 및 검증

4.1 구현 환경

본 실험에 사용된 환경은 표4와 같다. 서비스 포탈과 게이트웨이 시스템은 스프링 프레임워크로 구현하였고 마이크로 서비스 등록 관리를 위해 eureka를 이용하여 구성하였다. 멀티 클라우드 연동 대상은 퍼블릭 클라우드 2종, 프라이빗 클라우드 1종을 연동하였다.

<표 4> 멀티 클라우드 시스템 환경

서비스 포탈	게이트웨이 시스템	클라우드		
		퍼블릭		프라이빗
Ubuntu	Ubuntu	AWS	Azure	OpenStack
Spring Framework	eureka-server 2.1.1	awssdk 2.5.51	azure 1.20.0	openstack4j 3.2.0

4.2 구현 결과

본 실험에서는 멀티 클라우드 연동 테스트 결과를 서비스 대시보드를 통해 검증하였다. 그림 3은 대시보드 메인 화면으로 퍼블릭 클라우드 3종, 오픈스택 1종에 자원 운용 현황을 보여준다.

(그림 3) 멀티 클라우드 연동 대시보드 구현 결과

그림 4는 멀티 클라우드 기능 중 가상 자원을 위한 화면이다. 가상 자원 생성은 기본 정보 설정, 이미지 설정, 사양 설정, 보안 설정, 기타 설정으로 구분되어 있다. 사용자는 가상 자원 생성 시 공통 규격 외에 클라우드 독자적인 설정은 자동으로 선택되거나 기타 설정에서 추가할 수 있게 구현하였다.

(그림 4) 가상 인스턴스 생성 화면

그림 5는 생성된 가상 인스턴스 결과 화면으로 멀티 클라우드 연동 시스템이 요청한 자원이 연동된 멀티 클라우드에서 생성된 결과이다.

(그림 5) 클라우드 자원 생성 결과 화면

표 5는 본 논문에서 개발한 멀티 클라우드 연동 게이트웨이를 통해 호출한 시간과 클라우드 중 오픈스택에 대한 직접 인터페이스 직접호출 시간을 비교한 결과이다. 게이트웨이 호출 시간 평균 3.361초, 직접 호출 시간 3.35초로 평균 오버헤드는 0.011초로 측정되었다.

<표 5> 게이트웨이 호출과 직접호출 시간 비교

구분	게이트웨이 기반 API 호출 시간	OPENSTACK API 호출 시간	오버헤드
1	3.44	3.424	0.016
2	3.38	3.372	0.008
3	3.66	3.648	0.012
4	3.24	3.232	0.008
5	3.56	3.549	0.011
6	3.16	3.15	0.01
7	3.45	3.436	0.014
8	3.13	3.125	0.005
9	3.38	3.372	0.008
10	3.21	3.192	0.018
평균	3.361	3.35	0.011

5. 결론

본 논문에서는 멀티 클라우드 연동의 어려움을 해결하기 위해 연동에 필요한 인터페이스를 분석하여 공통 요소를 도출하고, 시스템을 마이크로 서비스 아키텍처로 구성하여 인터페이스 버전 변경이나 새로운 멀티 클라우드 추가 시 유연 대응이 가능한 구조로 설계하였다. 또한, 멀티 클라우드 연동 게이트웨이 시스템을 구현하고 이를 확인하기 위한 대시보드를 개발하여 본 논문에서 제안한 시스템의 유용성을 확인할 수 있었고, 게이트웨이 오버헤드도 직접호출하는 것과 비교하여 상당히 낮은 것으로 확인되었다. 하지만 본 논문에서 연동한 클라우드가 3종류로 제한되어 있어 다양한 멀티 클라우드 서비스 연동에 대한 검증이 부족한 부분이 있어 향후 추가 연구를 통해 다양한 클라우드 서비스를 추가 연동하여 확장 가능한지에 대해 검증을 추진할 것이다.

Acknowledgment

본 연구는 국토교통부/과학기술정보통신부/국토교통과학기술진흥원의 스마트시티 혁신성장동력 프로젝트 지원으로 수행되었음(과제번호 20NSPS-B149391-03).

참고문헌

[1] Couto, Rodrigo S., et al. "Building an IaaS cloud with droplets: a collaborative experience with OpenStack." Journal of Network and Computer Applications 117 (2018): 59-71.

[2] Kim, Dongmin, et al. "TOSCA-based and federation-aware cloud orchestration for Kubernetes container platform." Applied Sciences 9.1 (2019): 191.

[3] 강동재, et al. "클라우드 서비스 브로커 기술 및 사례 분석." 한국통신학회지 (정보와통신) 30.4 (2013): 7-15.

[4] Quint, Peter-Christian, and Nane Kratzke. "Towards a Description of Elastic Cloud-native Applications for Transferable Multi-Cloud-Deployments." Small 147 (2017): 34.