박사학위논문

공개형 PaaS 클라우드 컴퓨팅 기반 공간정보 처리 시스템 개발 및 성능 평가

2018년

한성대학교 일반대학원

정보컴퓨터공학과

정보시스템공학전공

김 광 섭

박사학위논문
지도교수 이기원

공개형 PaaS 클라우드 컴퓨팅 기반 공간정보 처리 시스템 개발 및 성능 평가

Development of Geo-based Information Processing System based on Open PaaS Cloud Computing and Its Performance Evaluation

2017년 12월 일

한성대학교 일반대학원

정보컴퓨터공학과

정보시스템공학전공

김 광 섭

박사학위논문
지도교수 이기원

공개형 PaaS 클라우드 컴퓨팅 기반 공간정보 처리 시스템 개발 및 성능 평가

Development of Geo-based Information Processing System based on Open PaaS Cloud Computing and Its Performance Evaluation

위 논문을 공학 박사학위 논문으로
제출함

2017년 12월 일

한성대학교 일반대학원

정보컴퓨터공학과

정보시스템공학전공

김 광 섭

김광섭의 공학 박사학위 논문을 인준함

2017년 12월 일

심사위원장 _____(인)

심 사 위 원 _____(인)

심 사 위 원 _____(인)

심 사 위 원 _____(인)

심 사 위 원 _____(인)

국 문 초 록

공개형 PaaS 클라우드 컴퓨팅 기반
공간정보 처리 시스템 개발 및 성능 평가

한 성 대 학 교 일 반 대 학 원
정 보 컴 퓨 터 공 학 과
정 보 시 스 템 공 학 전 공
김 광 섭

클라우드 컴퓨팅 기술을 통해 제공되는 서비스는 모델에 따라 각기 다른 특징을 지니고 있다. 기술이 복잡해지면서 여러 서비스 모델이 있지만 미국 국립표준기술 연구소의 표준 서비스 모델은 IaaS, PaaS, SaaS 이다. 서비스 모델을 묘사할 때 대부분 계층으로 나타내고 있지만 서로간에 반드시 관계될 필요는 없다. 전 세계적으로 클라우드 컴퓨팅 서비스 모델 중 IaaS와 SaaS는 상용화 및 안정화되어 많이 활용되고 있다. 국내에서도 클라우드 컴퓨팅 기술을 활용된 서비스가 급부상하고 있다. IaaS는 인프라 환경을 제공하는 서비스로 사용자들이 테스트 및 서비스를 제공하기 위해 필요한 컴퓨팅 물리적 자원을 복잡한 과정 없이 바로 가져다 쓸 수 있는 모델이다. 하드웨어를 구입하여 사용하는 방식에 비해 비용이 절감되는 것뿐만 아니라 필요에 따라 적절하게 자원을 확장하거나 감소할 수 있기 때문에 효율성 측면에서도 큰 장점을 지니고 있다. SaaS는 제공하기 위한 컴퓨팅 자원에 소프트웨어가 설치되어 사용자가 바로 사용할 수 있도록 제공되는 서비스 모델이다. 그렇기

때문에 사용자는 자신의 장치에 소프트웨어를 설치할 필요가 없다. 대부분 웹 브라우저를 통해 제공되므로 브라우저가 탑재된 다양한 장치에 제공되며, 소프트웨어를 구매하는 것이 아닌 사용 기능 및 시간에 따라 요금이 청구되는 방식이다. 이러한 형태를 주문형 소프트웨어(On-demand Software)라고 한다. PaaS는 SaaS가 제공되는 형태를 개발자와 운영자 관점으로 이동한 서비스 모델로 볼 수 있다. 사용자들의 서비스에 대한 요구 조건이 점점 다양해지고 복잡해지지만 사용면에서는 편리한 서비스를 제공받기를 원한다. 그러다보니 서비스를 개발하거나 운영하는 부분이 점점 복잡해지고 있다. 하지만 이를 일반적인 환경에서는 해결할 수 있는 방법이 쉽지 않다. PaaS를 사용한다면 개발 및 운영에 필요한 환경을 서비스 형태로 제공받을 수 있어 개발자는 실 서비스에 필요한 기능을 좀 더 집중할 수 있으며, 운영자는 서비스 관리를 체계적으로 수행할 수 있다. 아직까지는 많이 활용되고 있는 IaaS와 SaaS에 비해 PaaS 모델은 필요성이 최근 부각되면서 전 세계적으로 초기 시장으로 접어들고 있다. 클라우드 기술을 선도하고 있는 기업에서도 PaaS와 관련된 상업적 서비스를 발표하기 시작하였고, 이와 관련된 오픈소스 소프트웨어에 대한 관심도 증가되면서 활동이 점차 넓어지고 있다. 공간정보 분야 관련 서비스는 일반 사용자에게 제공되는 서비스부터 복잡한 처리를 제공하는 서비스까지 광범위하다. 공간 데이터는 지리, 위치, 속성 등 여러 형태가 포함되어 있어 데이터를 시각화하거나 처리하는 공간정보 서비스를 제공하기 위해 여러 기술 활용이 필요하다. 하지만, 서비스를 제공하기 위해 사용되는 기술들은 데이터 특성으로 인해 개발자 및 운영자가 기술에 대한 설치부터 설정 등 모든 지식을 습득하기에는 어려움이 있다. 그렇기 때문에 공간정보 관련 기술을 PaaS를 통해 제공한다면 개말자 및 운영자 입장에서 큰 이점을 얻을 수 있다고 생각된다. 공간정보 분야에서도 클라우드 컴퓨팅 기술을 활용한 서비스들이 전 세계적으로 발표되고 있지만, 대부분 데이터 중심의 SaaS 서비스이고 IaaS 장점을 활용한 연구가 몇 차례 수행되었다. 이에 반해 PaaS 관련된 연구 및 사례는 전 세계적으로 수행된 바 없다. 공간 데이터는 개인보다는 국가 차원으로 데이터가 수집되거나 관리되는 경우가 대부분이다. 그러므로 공공사업에 처음으로 사용되는 경우가 상대적으로 많을 수밖에 없다. 국내에

서는 공공사업 정보시스템 웹 개발 표준 정립을 위해 전자정부 표준프레임워크를 제공하고 있다. 하지만, 전자정부 표준프레임워크에는 공간정보 웹 시스템 개발 시 필요한 공통적인 기능에 대해 고려하고 있지 않아 관련된 기술이 정의 및 구축되어 있지 않다. 공간정보를 다루기 위한 전 세계적 공통 웹 표준 인터페이스가 발표되어 있으며, 이를 표준프레임워크에서 활용할 수 있도록 제공함으로써 공간정보 서비스가 필요한 전자정부 웹 서비스 개발에 가속도를 붙일 수 있다. 표준프레임워크 목적인 품질 및 재사용 향상에도 부합되는 기술이다. 이번 연구에서는 앞서 언급한 여러 가지 사항을 고려하여 향후 공간정보 관련된 서비스를 PaaS에서 제공하기 위해 필요한 사항들을 정리하고 직접 구축하였다. PaaS 시스템은 여러 방법을 통해 구축될 수 있다. 이번 연구에서는 IaaS 기반에 PaaS를 구축하였다. IaaS 구축을 위해 오픈스택을 활용하였고, PaaS는 클라우드 파운더리를 활용하였다. 이 두 시스템은 클라우드 컴퓨팅 서비스 모델에 대표적인 오픈소스이다. 이 외에도 향후 실제 활용 및 확장 가능성을 위해 모든 기술은 오픈소스를 사용하였으며, 공간정보 처리 서비스 개발 시 국내 전자정부 표준프레임워크에 적용 가능하도록 설계하였다. 설계 및 구축된 시스템은 공간정보 분야에서 사용되는 특정 기능을 대상으로 한 것이 아닌, 공간정보 표준을 활용하여 향후 어디든지 활용할 수 있거나 확장 가능한 형태로 구축하였다. 또한, 향후 실제 구축된 시스템에 대해 클라우드 컴퓨팅 서비스 모델에 따른 성능과 표준프레임워크 적용에 대한 성능 테스트를 수행하였다. PaaS는 관심이 급부상되고 있는 서비스 종류이다. IaaS와 PaaS 모두 최종 사용자에게는 SaaS 형태로 서비스가 제공된다. PaaS로 제공된 시스템이 IaaS로 제공된 시스템보다 심각한 성능 저하 요소가 존재한다면 이를 활용하는데 어려움이 있을 수 있다. 그러므로 구축된 시스템에 대해서 성능 테스트에 대한 검증이 반드시 필요하다. 아직까지 이러한 테스트가 수행된 사례가 없다. 국내 전자정부 표준프레임워크도 최근 PaaS 환경에서 제공하기 위한 연구가 진행 중이다. 공간정보 관련된 기능을 추가와 동시에 표준프레임워크 기반으로 서비스 제공 시 이점이 많지만 적용 전과 후에 내부적으로 변경되는 사항들이 있기 때문에 실제 서비스에 대한 성능 테스트 또한 반드시 필요하다. 향후 표준프레임워크 활용에 중요한 자료이지만 이러한 연

구가 수행된 바 없다. 이번 연구에서 수행된 시스템 설계 및 구축과 성능 테스트 결과는 향후 공간정보 분야와 국내 전자정부 표준프레임워크에 대한 클라우드 컴퓨팅 적용에 중요한 연구 중 하나가 될 것으로 예상한다.

【주요어】 클라우드 컴퓨팅, 플랫폼 서비스, 전자정부 표준프레임워크, 공간정보 웹 표준, 오픈소스

목 차

표 목 차

그 림 목 차

제 1 장 서 론

제 1 절 연구 배경

정보통신기술(ICT: Information and Communication Technology)이 급속하게 성장을 하면서 이를 제공 받는 사용자들에게도 몇 가지 변화들이 생기고 있다. 가장 큰 변화로는 사용자들이 업무 또는 삶의 질을 높이기 위해 사용하는 소프트웨어들이 구매하여 개인용 장치에 설치하는 것이 아닌 서비스 형태로 변화되고 있다는 것이다. 사용자들은 소프트웨어를 사용하기 위해 구입처를 통해 구입하고 이를 사용하고 있는 장치에 설치하여 사용하였다. 소프트웨어를 사용하는 장치는 대부분 데스크톱에 한정되어 있었다. 하지만 IT 기술이 발전되면서 사용자가 직접 소프트웨어 설치가 가능하고 쉽게 사용할 수 있는 스마트 장치들이 출시되었다. 이로 인해 제공되어야 하는 장치 종류가 증가되어 장치마다 사용할 수 있는 소프트웨어를 개발하기 위해 많은 개발 인력이 필요하게 되었다. 이러한 환경으로 변화되면서 설치형 소프트웨어가 아닌 비 설치형 소프트웨어를 제공할 수 있는 웹 서비스에 대한 관심이 높아지고 있다. 인터넷을 기반으로 하는 웹 서비스는 이메일, 스케줄 등과 같은 서비스를 시작으로 최근에는 복잡한 기능을 포함하고 있는 서비스를 일부 추가 금액을 청구하여 제공하고 있다. 소프트웨어를 구입하여 설치하는 것이 아니라, 기능이 구축된 서비스에 접속하여 원하는 기능을 사용하고 사용한 만큼 요금을 지불하는 것이다. 이러한 방식을 주문형 소프트웨어(On-demand Software)라고 부른다. 온디맨드 방식은 설치형 소프트웨어 제공 방식과는 많은 차이를 볼 수 있다. 설치형 소프트웨어는 버전별로 구성되어 있고 사용자가 하나의 기능만을 필요로 하더라도 그 기능만을 구입하는 것이 아닌 기능이 포함된 버전 모두를 구입해야한다. 하지만 온디맨드 방식은 원하는 기능에 대해서만 요청하여 사용하고 이에 대한 사용료를 지불하는 방식이다. 실제 최근에 서비스 되는 대표적인 예로 마이크로소프트사에서 제공되고 있던 엑셀, 파워포인트, 워드 등 오피스 소프트웨어를 웹에서 바로 사용할 수 있도록 서

비스 형식으로 제공하는 오피스 365(Office 365)와 어도비사에서 제공하는 이미지 편집 및 동영상 편집 소프트웨어들을 서비스 형식으로 변경한 어도비 크레이티브 클라우드(Adobe Creative Cloud)가 있다. 이러한 변화에서 정보통신기술은 소프트웨어에서 서비스로 변화되고 있는 과정에서 중요한 역할을 하고 있다.

클라우드 컴퓨팅(Cloud Computing)은 변화되고 있는 생태계에 중요한 역할을 하는 정보통신기술 중 하나이다. 미국 국립 표준 기술연구소(NIST)에서 정의한 클라우드 컴퓨팅은 공유된 컴퓨팅 자원들(네트워크, 서버, 스토리지, 서비스 및 어플리케이션 등)을 언제 어디서나 편리하게 이용할 수 있는 온디맨드 방식 네트워크 접근 모델이라고 정의하고 있다. 표준으로 정의된 서비스 모델은 인프라 서비스 (IaaS: Infrastructure as a Service), 플랫폼 서비스 (PaaS: Platform as a Service), 소프트웨어 서비스 (SaaS: Software as a Service)로 제공된다. 미사용 자원에 대한 비용 절감, 스토리지 공간 확장성 등에 대한 장점으로 활용이 점차 확대되고 있으며 2011년부터 현재까지 Gartner의 120대 전략기술 트렌드로 자리 잡고 있다(Chou, 2015; 한국 클라우드컴퓨팅연구조합 연구팀, 2017). 물론 실제 활용 측면에서 고려되어야할 보안, 신뢰성, 상호 운용에 대한 고려가 반드시 선행 연구가 수행되어야 하며 Avram(2014)은 이러한 사항을 정리하였다. 국내에서도 클라우드 도입 및 이용확산을 위해 2009년도부터 클라우드 종합계획을 수립하고, 클라우드 컴퓨팅 발전 및 이용을 촉진하여 서비스를 안전하게 이용할 수 있도록 환경을 조성하는 클라우드 컴퓨팅 법이 2015년에 제정되었다(과학기술정보통신부, 2017). 또한, 실효성 확보와 서비스 품질, 성능 기준 등에 대한 정책 연구도 수행된 바 있다(서광규 등, 2016). 최근에는 클라우드 생태계를 조성하기 위해 공공 클라우드 지원센터가 설립되었다. 설립 이후 다양한 서비스가 출시되고 있으며 출시된 서비스는 국내 클라우드 스토어 씨앗(https://ceart.kr)을 통해 클라우드 컴퓨팅 기술을 기반으로 제공되는 서비스를 한 곳에서 검색, 선정, 체험, 구매할 수 있도록 제공하고 있다. 국내에서는 공공사업 시스템 개발 시 표준 확립을 목표로 표준프레임워크를 지정하여 추가 기능 개발 후 제공하고 있다. 전자정부 표준프레임워크는 전자정부 웹 시스템 개발 시 개방형

표준 준수, 상호 운용성 보장, 국가적 표준화 지향 등에 대한 유연성을 강조하기 위해 기술을 국내 상황에 맞춰 추진되고 있다(http://www.egovframe.go.kr). 그렇기 때문에 정보통신기술 동향에 맞는 연구 및 서비스가 진행되고 있으며 최근 표준프레임워크도 클라우드 환경으로 제공되기 위한 연구가 진행 중이다. 이를 파스타(PaaS-TA)라고 부르며, 관련 기술을 모두 오픈소스로 공개하고 실제 사례를 제공하면서 다양한 시도를 진행하고 있다.

공간정보 분야에서도 데이터를 시각화하거나 처리하는 데 있어 다양한 소프트웨어들이 사용되고 있다. 전 세계적으로 가장 많이 사용되고 있는 공간정보 처리 소프트웨어로는 ESRI 사에서 개발한 ArcGIS, Exelis Visual Information Solutions에서 개발한 ENVI, Hexagon Geospatial에서 개발한 ERDAS IMAGINE이 있다. 상용 소프트웨어 외 오픈소스 소프트웨어로 QGIS, GeoTools GDAL, GeoServer, OTB(The ORFEO Tool Box) 등 다양한 소프트웨어가 출시되어 있다. 대부분 소프트웨어는 공간정보를 시각화하고 영상 전체 또는 일부를 처리하는 기능을 포함하고 있다. 앞서 언급했듯이 정보통신 기술 변화로 소프트웨어들이 서비스 형식으로 제공되면서 공간정보 분야에서도 활용 가능한 수준에서 생태계가 변화되고 있다. 하지만 아직 많이 활용되고 있는 서비스는 공간정보를 직접 사용하거나 실시간 처리보다는 배경지도로 활용하거나 이미 처리된 데이터를 시각화하는 정도로 그치고 있다. 물론 최근에 정보통신 기술 동향에 맞춰 공간정보 분야에서도 클라우드 컴퓨팅 기술을 적용하려는 서비스 및 연구가 수행되고 있다. 공간정보는 대용량 데이터로 취급되는 경우가 많다. 그렇기 때문에 데이터 관점에서 클라우드 컴퓨팅을 접목하여 수행된 연구 사례가 있다(Yang et al., 2011; Li et al., 2016; Huang et al., 2017; Yang et al., 2017). 국내에서는 공간정보 IaaS 클라우드 서비스 기술 개발 방안을 연구한 사례(윤준희 등, 2017)와 데이터 서비스를 중심으로 클라우드와 공간정보 분야 연계 방안을 연구한 국내 사례도 수행된 바 있다(이인수 등, 2016). 실제 상용 소프트웨어에 대한 사례도 존재한다. ArcGIS는 ArcGIS Enterprise in the cloud 서비스를 통해 소프트웨어 서비스 제공을 시작하였고 ENVI 소프트웨어도 클라우드 서비스를 통해

소프트웨어를 제공하고 있다. 오픈소스로 제공되고 있는 QGIS 경우에는 데이터 관점으로 클라우드 컴퓨팅 서비스 모델인 데이터 서비스(DaaS)를 제공하고 있다. 공간정보 서비스와 클라우드 IaaS 관련된 연구는 클라우드 기술이 실질적으로 활용되면서 수행된 국내 연구들이 있다(Kim et al, 2013; 강상구와 이기원, 2014; 윤구선과 이기원, 2015; 윤구선 등, 2016). 특히, 실제 국내 클라우드 컴퓨팅 연구 및 적용 사례로 강상구(2016)는 클라우드 컴퓨팅 서비스 모델인 IaaS를 통해 모바일 클라우드 환경에서 공간정보 처리 분석 서비스에 대한 응용을 설계 및 구현한 연구를 수행하고 이에 대한 결과를 국내 항공우주 과학기술을 담당하고 있는 항공우주연구원(KARI: Korea Aerospace Research Institute)에서 제공하는 서비스인 위성정보 활용지원 서비스(http://ksatdb.kari.re.kr)에 적용한 사례가 있다(Lee et al., 2017).

이처럼 전 세계뿐만 아니라 국내에서도 공간정보 분야에서 제공되는 서비스를 클라우드 컴퓨팅 기술을 적용하여 서비스하기 위한 연구가 끊임없이 진행되고 있다. 하지만 한국클라우드 산업 협회에 가입된 회원사로 판단해보았을 때, 공간정보 관련 서비스를 전문으로 제공하는 업체에서는 아직 많이 참여하지 않은 것으로 파악되고 있다. 클라우드 기술은 SaaS 시장을 시작으로 IaaS에 대한 관심이 높아지고 있다. PaaS의 경우 초반 시장에는 큰 관심을 갖지 못했지만, IDG 2017 IT 전망 보고서에 따르면 2017년도 클라우드 컴퓨팅 트렌드 6가지 중 하나로 PaaS 기술 중 중요 기술인 컨테이너 기술을 보고 있다. 실제로 클라우드 컴퓨팅 서비스 형태인 IaaS는 안정화 단계에 이르렀고 이젠 PaaS에 대한 관심이 증가하고 있는 실정이다(Boultion, 2017). PaaS는 일반 사용자를 대상하는 것이 아니며, 개발자를 위한 서비스이다. 그렇기 때문에 PaaS가 모든 분야에서 필요한 사항은 아니다. 클라우드 컴퓨팅 기술 스택은 광범위하다. 또한 모든 서비스 모델은 계층적 구조가 아니기 때문에 관계없이 활용될 수 있다. 반드시 모든 분야에서 활용되는 서비스가 반드시 클라우드 컴퓨팅 서비스에 적용될 필요는 없다. 해당 분야에 특성에 맞춰 이를 적용하기 위한 선행 연구가 이루어지고 여러 정책이 고려 된 후 적용하는 것이 순서이다. 공간정보 분야에서 활용되는 기술들은 전문가적 요소가 상당히 많이 포함되어 있다. 공간정보 관련 서비스를 사용자들에게 웹 서비스로 제공

하기 위해서는 활용 기술들이 필요하지만 일반 개발자들이 모든 기술적 내용을 습득하기에는 어려운 측면이 있다. 이러한 사항을 고려해보았을 때 공간정보 분야에서는 PaaS 적용이 필요할 수 있다. 하지만 아직 공간정보 분야에서 PaaS를 적용하기 위한 연구 및 서비스는 수행된 바 없다. 그렇기 때문에 PaaS를 구성하기 위한 전반적인 내용을 정리하고 공간정보 분야에서 활용되는 소프트웨어 또는 기술들이 PaaS에 적용되기 위해 고려해야 되는 점 등 다양한 선행연구가 필요하다. 또한, 국내 기술 동향을 파악해보았을 때 공공사업에 많이 활용되는 전자정부 표준프레임워크에는 공간정보 관련된 기능이 고려되어 있지 않다. 공간정보 관련 서비스는 데이터를 보유한 공공기관에서 1차로 제공해야 민간에서 사용이 가능한 경우가 대부분이다. 그렇기 때문에 공공 시스템 표준 정립을 목표로 하고 있는 전자정부 표준프레임워크에 공간정보 서비스를 위한 기능이 포함되어 사용된다면 이를 통해 좀 더 다양한 서비스가 제공될 수 있다. 하지만 이러한 내용에 대해 실질적으로 언급되고 있지 않고 있다. PaaS는 개발자 및 운영자에 대한 편의 서비스로 최종 사용자에게 제공되는 서비스는 달라지는 사항이 없다. 그러므로 기존 다른 클라우드 컴퓨팅 서비스 방식에 대한 성능에 대한 테스트가 반드시 수행되어야한다. 전자정부 표준프레임워크도 마찬가지이다. 표준프레임워크를 사용함으로써 개발적인 측면에 이득을 보는 것이 대부분이다. 만약 적용되었을 때 성능에 대한 부분이 현저히 떨어진다면 아무리 개발적인 측면에 장점이 있다고 해도 활용하기 어렵다. 이러한 성능 테스트는 실제 서비스화 되기 전 다양한 형태로 수행이 되어야 한다.

제 2 절 연구내용 및 방법

이번 연구에서는 공간정보 분야에서 활용 가능한 정보통신기술 기술 중 최근 가장 관심도가 높고 활용성이 높다고 생각되는 기술인 클라우드 컴퓨팅을 접목하여 시대 흐름에 맞춰 나가기 위한 연구 중 하나로 볼 수 있다. 그렇기 때문에 전반적인 내용부터 설계 및 구축까지 수행하여 여러 방면에서 가능성을 검토하였다. 먼저, 클라우드 컴퓨팅에 대한 전반적인 내용을 정리하였

다. 이번 연구에서는 클라우드 컴퓨팅 서비스 모델 중 PaaS를 중심으로 공간정보 분야 관련 서비스와 통합하기 위해 집중적으로 고찰하였다. 실질적으로 PaaS를 구축하기 전에 전반적인 내용, 현재 동향 등을 조사 및 정리하고 기술적인 내용을 서술하였다. 또한, 실제 구축이 가능한 공개형 PaaS를 검토하여 여러 플랫폼 소프트웨어 중 많이 활용되고 검증된 공개형 PaaS를 기반으로 구축하였다. 구축된 PaaS 위에 공간정보 및 공공 데이터를 융합 시각화할 수 있는 웹 시스템과 위성영상 처리 웹 시스템을 설계 및 구현 후 배포하여 PaaS 기반의 공간정보 처리 시스템에 대한 일반적인 방안을 제시하였다. 특히, 국내 공공기관 동향에 맞춰 공간정보 분야의 전자정부 표준프레임워크에 대한 도입을 고려하여 웹 시스템을 향후 표준프레임워크 적용 가능성을 고려하여 설계 및 구축하였다.

연구 방법은 다음과 같다. PaaS가 관심이 증가하고 실제 서비스가 되고 있지만 아직 이에 대한 논술 정리가 부족하다. 그렇기 때문에 이번 연구에서는 제일 먼저 클라우드 PaaS에 대한 개념적 및 기술적 정리를 수행하였다. 추가로 서비스가 제공되고 있는 PaaS 상용 서비스와 오픈소스 기반 PaaS를 파악하였다. PaaS는 물리적 하드웨어에 바로 구축되거나, IaaS 위에 구축되는 방식으로 크게 두 가지로 나눌 수 있다. 이번 연구에서는 구축 방법 중 IaaS 위에 PaaS를 구축하는 시스템을 선택하였다. 그렇기 때문에 이번 연구에서는 실제 오픈소스 기반의 IaaS 및 PaaS를 모두 구축하였다. IaaS는 오픈스택(OpenStack)을 기반으로, PaaS는 클라우드 파운더리(Cloud Foundry)를 사용하였다. 이번 연구의 목적은 PaaS 가능성을 확인하기 위함이므로 오픈스택을 설치하는 하드웨어 구성은 PaaS를 구축하기 위한 최소 하드웨어 및 서비스로 구성하였다. 오픈스택 설치에 필요한 하드웨어 요구사항으로는 컨트롤 노드(Controller Node)와 컴퓨트 노드(Compute Node)이며, PaaS를 구축하기 위한 필수요소인 블록 스토리지 노드(Block Storage Node)가 추가되었다. 오픈스택 설치 시 필수적인 요소인 인증 서비스(identity service)인 키스톤(Keystone), 이미지 서비스(image service)인 노바(Nova), 컴퓨트 서비스(Conpute service)인 글라스(Glance), 네트워크 서비스(networking service)인 뉴트론(Neutron), 블록 스토리지 서비스(block Storage service)인 신더

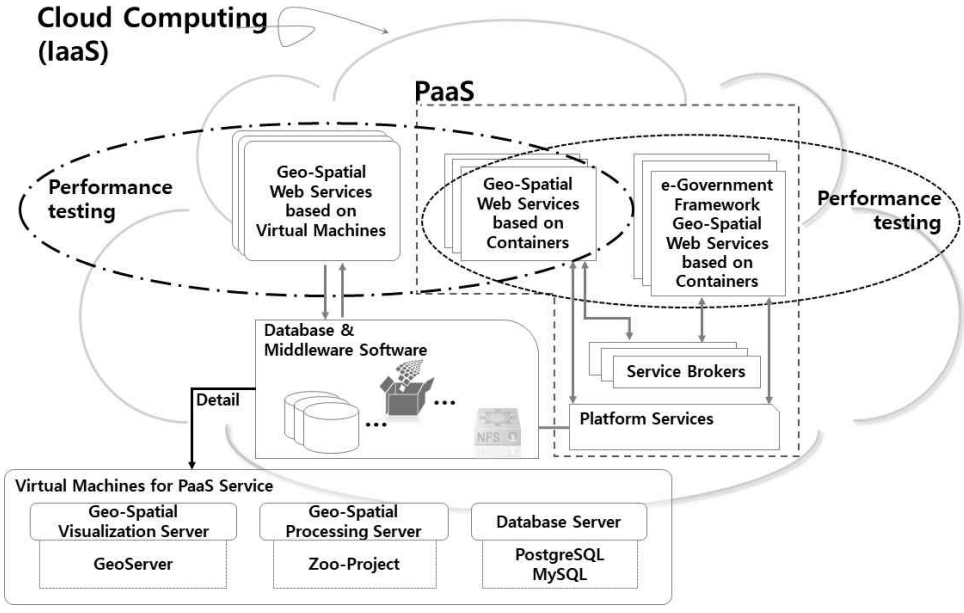
(Cinder)를 설치하였다. 추가로 공간정보 웹 서비스에서 사용될 수 있는 공유 파일 시스템 서비스(shared file systems service)인 마닐라(Manila)도 추가 설치하였다. 클라우드 파운더리 PaaS를 구축하기 위해서는 최소 가상화로 생성되어야 하는 자원이 vCPU 56개, 메모리 118GB가 필요하다. 이번 연구에서는 이 사양을 충족시키기 위해 총 네 대의 워크스테이션 급 하드웨어 장비를 컴퓨트 노드로 활용하였다. IaaS를 구축한 후 클라우드 파운더리를 설치하였다. 클라우드 파운더리는 마이크로 서비스 단위로 구성되어 여러 인스턴스를 생성하고 설정해야하므로 BOSH를 통해 설치되게 되어 있다. BOSH는 시스템 릴리즈를 위한 오픈소스 툴로 엔지니어링, 개발, 라이프 사이클 관리와 모니터링을 하도록 도움을 주는 툴이다. 생성할 인스턴스에 대한 크기, 설정 등을 YAML 설정으로 미리 작성 후 수행하여 클라우드 파운더리를 구동하기 위한 인스턴스들을 자동으로 생성하고 테스트를 수행한다.

IaaS와 PaaS 구축이 모두 완료되면 공간정보 분야에서 활용될 미들웨어 소프트웨어와 여러 기술을 PaaS에서 서비스로 활용하기 위해 설계 및 구축하고, 최종 결과로 공간정보 웹 시스템을 설계 및 구축하여 기능 구동 확인과 동시에 성능 테스트를 수행하였다. 데이터베이스 서비스를 PaaS에서 사용할 수 있도록 인스턴스를 구성 및 서비스화하고 공간정보 시각화 및 처리를 위한 소프트웨어 또한 PaaS에서 사용할 수 있도록 인스턴스를 구성하였다. 공간정보 시각화 인스턴스에는 테스트에 활용될 시각화 데이터를 다운로드 및 전처리 과정을 거친다. 벡터 데이터로 사용될 데이터는 다운로드 후 바로 데이터베이스에 삽입함으로써 시각화 준비가 모두 완료되고, 래스터 데이터의 경우 전국 데이터를 사용하기 위해 공개된 위성영상을 다운로드 받아 이를 병합하는 과정을 전처리 과정으로 먼저 수행하였다. 수행된 데이터를 공간정보 시각화 서버에 업로드하였다. 공간정보 처리를 위한 인스턴스에는 처리를 위한 서버가 구축되고 실제 처리를 위한 소프트웨어들이 서버와 연계되어 있다. 연계를 위해 추가 서비스 개발이 필요하며 모든 시스템을 설계 및 구축할 때 언어의 일관성을 위해 개발이 필요한 언어는 모두 자바로 통일하였다. 처리 서비스 연계 시 처리된 데이터는 공간정보 시각화 서버에 업로드 및 배포되어 향후 사용자가 처리된 원본 데이터를 다운로드 받거나 웹에서 시각화할

수 있도록 연계되어 있다. 처리 서버에는 얼마든지 사용자가 원하는 알고리즘 또는 소프트웨어를 추가할 수 있도록 구축되어 있으며 이번 연구에서는 오픈소스 공간정보 처리 소프트웨어를 활용하여 테스트를 위한 처리 기능을 네 가지를 연계하여 웹 시스템에서 활용할 수 있도록 구성하였다. 이렇게 구축된 모든 시스템을 웹 브라우저에서 실제로 확인하기 위해 미들웨어 어플리케이션 서버와 웹 클라이언트를 설계 및 구현하였다. 웹 서비스 구축 시에도 자바 기반으로 어플리케이션을 구축하였다. 공간정보 및 공공 데이터 융합 웹 서비스와 위성정보 처리 웹 시스템으로 나눠진다. 공간정보에서 활용되는 서비스를 크게 두 가지로 나누면 벡터와 래스터 서비스로 구분할 수 있다. 이러한 점을 고려하여 PaaS에 장점을 확인하기 위해 하나로 통합된 서비스를 구축하는 것이 아닌 기능에 따라 어플리케이션을 나누어 구축하였다. 공간정보 및 공공 데이터 융합 웹 시스템은 공공 데이터를 수집 및 관리하고 수집된 데이터를 공간정보와 융합하여 시각화하는 웹 서비스이다. 위성정보 처리 웹 시스템은 위성영상을 시각화하고 이를 처리하여 결과를 시각화하는 웹 서비스이다. PaaS의 전반적인 구조를 파악하고 공간정보 분야 특성에 맞게 추가 후 실제 테스트 시스템을 구현하여 구동 결과를 확인하였다.

마지막 연구결과로 구축된 PaaS에 대한 성능 테스트를 수행하였다. PaaS로 구동될 경우 성능이 기존 서비스와 비교하였을 때 다른 부분이 존재하는지를 확인하고자 한다. 이를 비교하기 위해 IaaS로 제공되는 웹 서비스를 추가 설계 및 구축하여 PaaS에서 구동되는 웹 서비스와 성능을 비교 분석하였다. 국내 전자정부 표준프레임워크에 대한 사용이 권장되고 있는 시점에서 이번 연구에서는 표준프레임워크를 고려하여 시스템을 설계하였다. 앞서 언급된 웹 시스템의 경우 모두 스프링 부트를 기반으로 구축하였다. 전자정부 표준프레임워크 또한 자바 기반의 스프링 프레임워크를 기반으로 구성되어 있고 기존 구축된 시스템을 설계할 때 표준프레임워크 적용 가능성을 고려하였으므로 표준프레임워크를 기반으로 추가 구축하였다. 구축된 표준프레임워크 기반 위성정보 처리 웹 시스템을 공통 컴포넌트를 추가하여 PaaS에 배포하였다. 배포된 시스템을 가지고 성능 테스트를 수행하였다. 이번 연구에서 성능 테스트한 목적은 어느 시스템이 더 좋은지를 판단하기보다는 성능 저하에 대한

여부를 확인하기 위한 테스트이며, PaaS로 구축이 되었을 때 장점으로 뽑히는 마이크로 서비스 형태의 컨테이너 방식에 대한 성능 확인으로 볼 수 있다. 그림 1-1은 이번 연구에서 수행되는 전체적인 목표를 도식화 한 것으로 클라우드 컴퓨팅 기반의 공간정보 처리 시스템에 대한 전체적인 구성도와 성능 테스트 대상에 대해 나타내었다. 논문은 다음과 같이 구성된다. 2장에서는 공간정보 웹 서비스를 PaaS에 배포하기 위한 클라우드 컴퓨팅과 PaaS에 대한 현재 동향을 파악하고 기술적인 내용을 분석한 내용을 정리한다. 국내 전자정부 표준프레임워크에 대한 기본적인 내용과 표준프레임워크의 PaaS 환경 변화 동향에 대해서도 정리되었다. 3장에서는 이번 연구에서 제안하는 공간정보 서비스를 위한 PaaS 설계 및 구축에 대한 내용에 관해 설명한다. 실제 IaaS 및 PaaS 구축과 공간정보 관련 소프트웨어를 서비스화하기 위한 구축 내용과 공간정보 웹 서비스에 대한 설계 부분을 설명한다. 4장에서는 구축된 결과에 대한 기능별 구동 확인 결과를 확인하고, 성능 테스트 결과와 활용 가능성을 서술한다. 마지막 5장은 수행 결과에 대한 결론 서술로 정리된다.



[그림 1-1] 공개형 PaaS 클라우드 기반 공간정보 처리 시스템 연구 체계 및 성능 테스트 구성.

제 2 장 클라우드 컴퓨팅 PaaS

제 1 절 클라우드 컴퓨팅

클라우드 컴퓨팅 기술은 인터넷 기반 컴퓨팅으로 여러 기술이 집합되어 있다. 서버, 스토리지, 데이터베이스 등과 같은 물리적 또는 소프트웨어적 사항들을 인터넷을 통해 간단하게 접근할 수 있으며, 사용자에게 제공하는 범위 또한 데스크톱, 모바일 등으로 넓힐 수 있다. 서비스 형태로 제공되기 때문에 바로 데이터 처리 시 분산 처리가 필요하거나 높은 성능이 요구되는 경우 적은 비용으로 유용하게 활용될 수 있다(Hussain et al, 2013; Zhan et al., 2015). 주요 서비스 모델은 세 개로 구분될 수 있다. 인프라 서비스(IaaS: Infrastructure as a Service), 플랫폼 서비스(PaaS: Platform as a Service), 소프트웨어 서비스(SaaS: Software as a Service)로 구분되며, 최근에 복잡해지는 구조로 인해 Baas(Backend as a Service), DaaS(Data as a Service)와 같은 모델들이 추가되고 있다. 서비스 모델 외 배치 모델에 따라 폐쇄 클라우드(private cloud), 퍼블릭 클라우드(public cloud), 혼합형 클라우드(hybrid cloud)로 나눌 수 있다[그림 2-1].

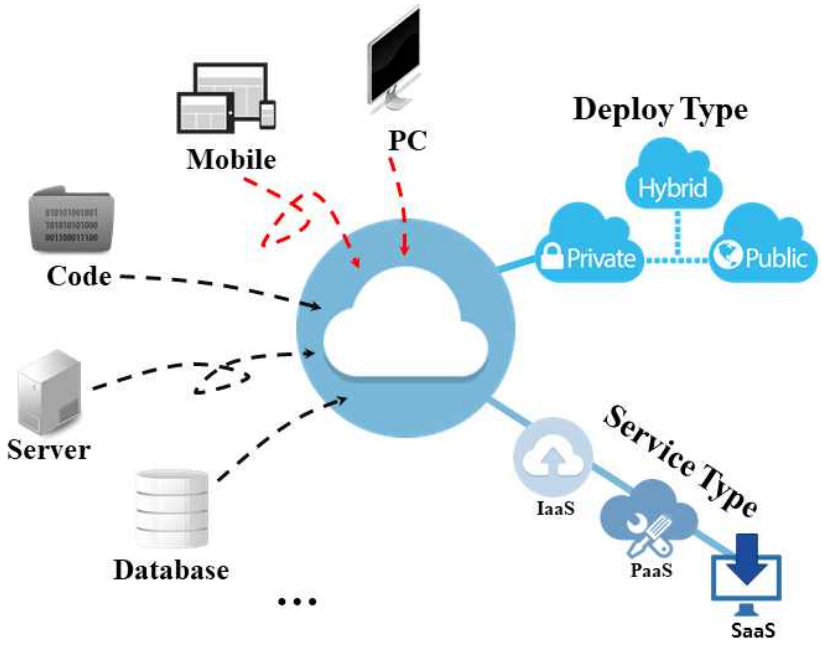
클라우드 컴퓨팅 기술이 발표되기 초기 시장은 대부분 바로 사용할 수 있는 SaaS에 대한 연구가 수행된 바 있다(신동희와 김경민, 2014). SaaS뿐만 아닌 다양한 업체에서 직접 클라우드 컴퓨팅 기술을 활용하여 서비스를 제공할 수 있도록 IaaS가 제공되기 시작되었다. 전 세계적으로 대표적인 상용 서비스로는 아마존의 아마존 웹 서비스(AWS: Amazon Web Service), 마이크로소프트사의 애저(Azure) 등이 존재한다. IaaS 상용 서비스는 사용 요금이 부과되지만 가입과 동시에 마우스 클릭만으로 가상 물리 서버가 제공되고 이를 유연하게 확장 및 축소할 수 있어 물리적 요소를 빠르게 구축할 수 있다. 요금에 대한 부담이 있고 물리적 요소를 보유하고 있을 경우 IaaS를 직접 구

축할 수 있다. IaaS를 직접 구축할 수 있는 오픈소스 소프트웨어로는 오픈스택(OpenStack), 클라우드스택(CloudStack) vSphere 등이 제공되고 있다. 직접 설치해야하므로 기술적인 내용을 파악하고 습득해야 한다는 점이 존재하지만, 구축하여 사용함으로써 완벽한 폐쇄형 IaaS를 제공할 수 있을 뿐만 아니라 확장하여 좀 더 특정 분야에 맞춰 서비스할 수 있다는 장점이 있다. 이러한 IaaS는 국내에서도 활용할 수 있도록 기업에서 국내에 데이터센터를 구축하여 서비스하고 있다. 이 외 국내 업체에서 직접 IaaS를 제공하는 서비스도 존재한다. 대표적인 예로 KT에서는 오픈소스 IaaS 소프트웨어를 활용해 인프라 환경을 직접 구축하여 서비스 형태로 기업 또는 일반인들에게 제공하고 있다. PaaS도 여러 상용 서비스가 최근 발표되고 있다. 신속한 모바일화, 소프트웨어 서비스에 필수 기능 추가, 사물 인터넷 활용, 새로운 비즈니스 프로세스 생성, 신속한 앱 출시, 데이터 시각화를 PaaS로 해결할 수 있는 비즈니스 현안으로 본 바 있다(Carr, 2015). 대부분 IaaS를 제공하고 있는 기업에서 추가로 구축하여 제공하고 있다.

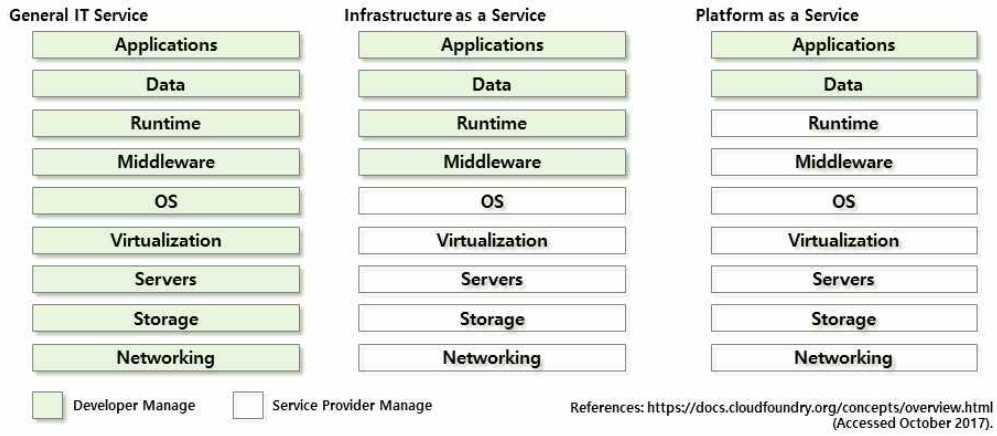
IDG Market Pulse (2017)에서 클라우드에 대한 기대 효과, 향후 추진 계획 등을 알아보고자 설문 조사를 진행한 바 있다. 70% 이상이 이미 클라우드 컴퓨팅을 도입하거나 계획 중이라고 응답하였다. 운영 방식으로는 하이브리드 클라우드를 41.1% 선호하였으며, 형태는 IaaS가 39.6%로 가장 많이 차지하였으며 PaaS는 24.9%로 적게 차지하였다. 특히, 클라우드 발전법이 통과되면서 클라우드 제한구역이었던 금융 업계에서도 기술을 적용하려고 하고 있다(서상수, 2016). 설문 조사에서도 확인할 수 있듯이 아직 클라우드 컴퓨팅 기술이 도입되는 형태는 IaaS를 대다수 생각하고 있다. 하지만 PaaS를 도입함으로써 좀 더 나은 서비스를 사용자에게 서비스할 수 있다. 그림 2-2는 전통 방식과 클라우드 서비스 모델에 따른 제공자 및 개발자 관리 범위에 대해 정리한 것이다. 일반적으로 웹 서비스를 위해서는 네트워킹, 저장소, 서버, 가상화, 운영체제, 미들웨어, 런타임, 데이터, 어플리케이션 9개의 레이어가 필요하다. 클라우드 컴퓨팅 기술이 적용되지 않은 전통 방식에 대한 서비스는 이 모든 것을 개발자가 직접 구축하고 관리해왔다. 하지만 클라우드 컴퓨팅 기술이 도입되면 서비스 제공자가 해당 하드웨어 및 소프트웨어를 구축하여

서비스 형태로 제공해주기 때문에 개발자가 관리해야 하는 범위가 점차 줄어들게 된다. 먼저 IaaS는 물리적 가상화를 지원하는 서비스이다. 그렇기 때문에 네트워킹, 저장소, 서버, 가상화와 추가로 운영체제에 대한 일부 지원이 서비스로 변경되어 제공한다. 더 나아가 PaaS는 미들웨어와 런타임 환경을 서비스 방식으로 제공한다. 그렇기 때문에 개발자는 최종적으로 사용자에게 제공되는 어플리케이션과 사용하는 데이터에 대해서만 구축하고 관리하면 된다. 개발자가 구축 및 관리해야 하는 레이어 범위가 줄어들면 자연스럽게 어플리케이션 개발에 집중할 수 있기 때문에 좀 더 나은 서비스를 제공할 수 있다.

클라우드 컴퓨팅에 포함된 기술은 데이터베이스, 가상화, 네트워크, 대용량 데이터, 분산 데이터 등 다양하게 존재한다. 이러한 기술로 가공된 클라우드 컴퓨팅 PaaS들이 오픈소스로 공개되어 있다. 표 2-1은 2017년 09월 기준으로 오픈소스 클라우드 플랫폼에 목록에 대해 정리한 것이다.



[그림 2-1] 클라우드 컴퓨팅 기술 개념 및 서비스 모델.



[그림 2-2] 전통 방식과 서비스 모델에 따른 제공자 및 개발자 관리 범위.

[표 2-1] 클라우드 PaaS 주요 오픈소스 목록(공개 SW 포털, 2017)

Name	License	Technology Support	Note
Apache Jclouds	Apache 2.0	Community	• Multi Cloud Toolkit
Apache Stratos	Apache 2.0	Community	• PaaS Framework
Cloudify	Apache 2.0	Prof/ Community	• Based on Python TOSCA • Cloud PaaS Solution
Nimbus	Apache 2.0	Community	• Open Source Toolkit that provides users with IaaS Cloud through WSDL Web Service API
Open Nebula	Apache 2.0	Community /Blog	• Visualization Solution to manage the Infrastructure of the Data Center to configure IaaS
Open QRM	GPL v2 & Commercial	Prof/ Community	• Data Center Management System
OpenStack	Apache 2.0	Community /Blog/ Wiki	• Project started by Rackspace and NASA • Control and Operation about Server, Storage, Network, Visualization
OpenShift	Apache 2.0	Prof/ Community	• PaaS and Cloud Operation Platform based on Standard Container
Osv	BSD	Prof/ Community	• Operation System • Supported Cluster Deploy Management based on Hypervisor
Eucalyptus	GPL & Commercial	Prof/ Community	• Cloud Platform about Integration Management of Server, Storage and Network infrastructure using NASA
Cloud Stack	Apache 2.0	Community	• Cloud Platform of Management Console, Various Hypervisors, Firewall, Load Balancing
Cloud Foundry	Apache 2.0	Prof/ Community	• Cloud Platform Service Solution

제 2 절 플랫폼 서비스

클라우드 컴퓨팅 PaaS는 발전하기 전 여러 기업에서 개발자들에게 제공해 주는 Open API(Application Programming Interface) 형태로 시작되었다. PaaS는 물리적 구조 위에 설치 및 구축되는 어플리케이션들을 개발, 실행, 관리할 수 있도록 도움을 주는 클라우드 컴퓨팅 서비스 분류 중 하나이다. 이러한 서비스 제공은 개발자들의 편의를 주는 것으로 IaaS가 안정화되면서 소프트웨어 서비스를 더욱더 원활하기 위한 방법의 하나로 최근 많은 관심을 받고 있다. Goncalves and Ballon (2011)은 소프트웨어 서비스와 PaaS를 비교하여 양면 시장 모델에 적합함을 보여주고 있다는 사례 연구를 제시한 바 있다. 또한 PaaS는 Gartner의 조사 결과에 따르면 클라우드 컴퓨팅 기술이 발표됨과 동시에 급성장하는 것이 아닌 가장 인상적인 성장세를 보이고 있다고 보도된 바 있다(Ferrer et al., 2016). PaaS는 몇 번의 클릭만으로 사용자가 원하는 개발 환경을 구축할 수 있어 운영 속도를 향상하기 위해 필요한 기술로 소개된 바 있다(김학진, 2017). 이렇게 적절한 시기를 맞은 PaaS는 여러 분야에서 활용될 수 있으며, 이에 대한 장점을 부각해 여러 분야 발전에 좋은 영향을 줄 수 있다. 이번 절에서는 공간정보 웹 서비스 배포 시 효율적으로 설계 및 구축에 대한 기초적 단계인 PaaS를 효율적으로 활용하기 위해 이에 대한 서비스 특징과 종류에 대해 정리하였다. 또한, 국내 시장을 고려하여 전자정부 표준프레임워크에 대해 개념 및 프레임워크에 대한 클라우드 전환 동향을 정리하였다.

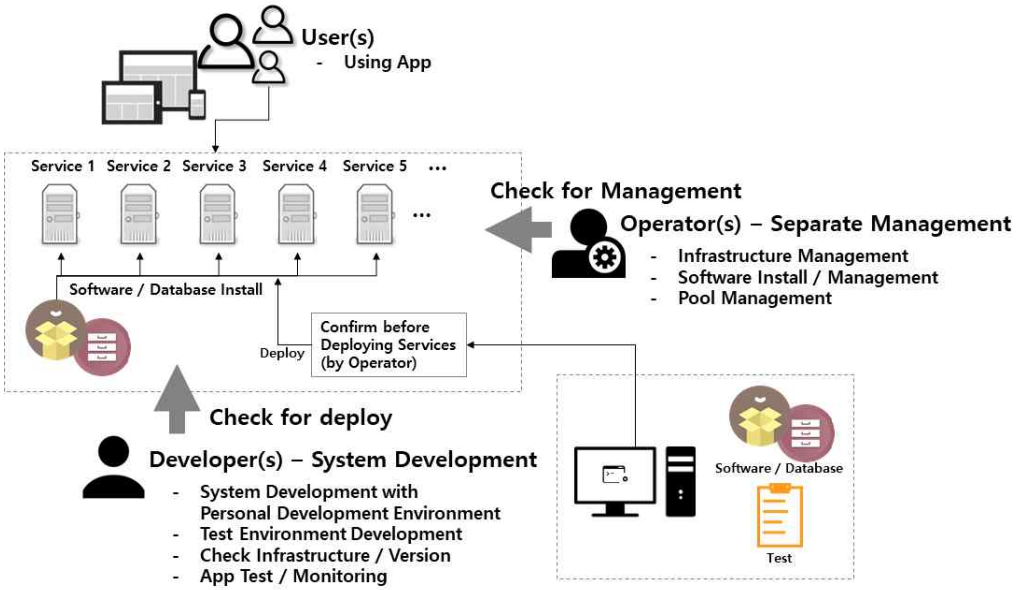
1) 플랫폼 서비스 특징

PaaS 개념은 일반 사용자에게 제공되는 소프트웨어 서비스에 대한 개념을 플랫폼에 적용한 것이다. IaaS에서 제공되는 요소에서 미들웨어와 런타임 환경을 서비스화하여 제공하는 방식을 PaaS이다. 런타임 환경은 자바(Java), 파이썬(Python), C 등과 같은 프로그래밍 언어들이 구동되는 환경을 말한다. 미들웨어의 경우 데이터베이스, 웹 컨테이너, 아파치 웹 서버 등과 같이 소프트

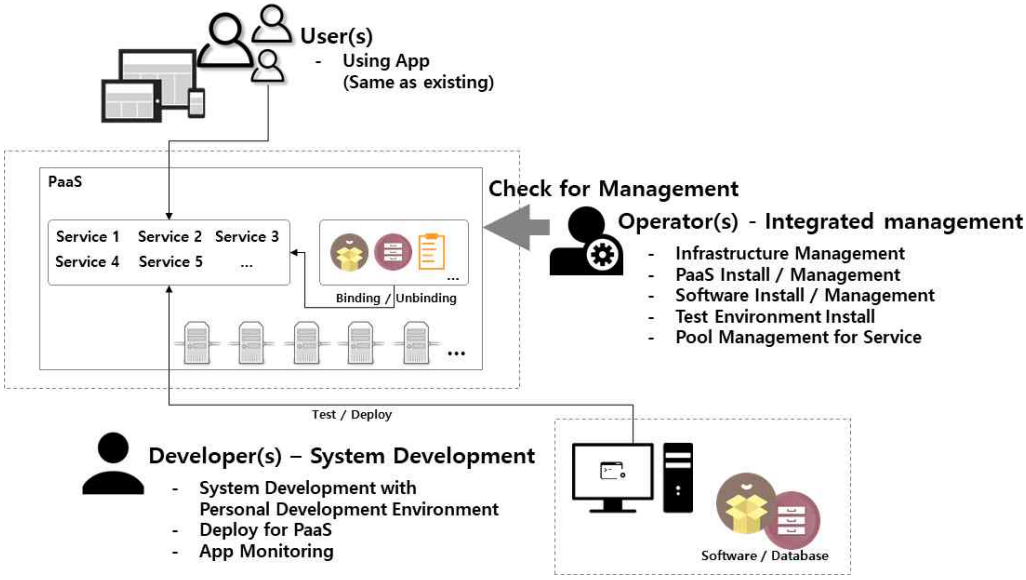
웨어 형식으로 제공되는 것으로 개발하거나 서비스 제공 시 여러 형태로 활용되는 소프트웨어들이다. 미국 국립 표준 기술연구소 정의에 따르면, PaaS의 주요 목표로는 기본 리소스 관리하는 부담을 줄여주는데 있다(Mell and Grance, 2011). IT 기술이 발전되고 사용자가 사용하기 편리한 서비스가 제공되고 있다. 하지만 이를 개발하고 구축하는 입장에서는 규모가 크고 복잡한 형태가 되고 있다. 그러다보니 미들웨어나 런타임 환경을 관리하는 측면 또한 중요한 이슈 중 하나이다. 특히, 성능이 보장되고 최소한의 비용으로 유지되는 것이 최대 관점이라 볼 수 있다. 이를 해결할 수 있는 여러 방안 중 하나로 PaaS를 볼 수 있으며, 자원 배분, 할당, 자동화 등에 대해 고려하여 실제 출시된 PaaS 및 소프트웨어를 분석한 연구 사례가 존재한다(Costache et al., 2017). PaaS는 독립적으로 구성되거나 IaaS 위에 구성되는 경우 두 가지로 나눌 수 있다. PaaS는 IaaS에서 제공되는 오토 스케일링과 같은 기능들이 서비스로 제공되고 있어 이 둘에 대한 비교 분석한 연구 사례 또한 존재한다(Vaquero et al., 2011). PaaS도 클라우드 컴퓨팅 공개 형태인 공개 및 폐쇄로 제공할 수 있다. 두 형태 모두 PaaS가 가지는 특성인 API, 개발과 배치 도구에 대해서는 공유가 되어 있다.

전반적인 웹 시스템 제공 시 전통적인 시스템과 클라우드 컴퓨팅 시스템에 대한 차이가 존재한다. 일반적인 웹 시스템 제공 시 개발자 및 운영 환경 관리자가 고려해야 하는 사항에 대해 그림 2-3과 같이 그림으로 정리하였다. 가상 하드웨어를 제공하는 IaaS 경우에도 시스템 구성 시에도 전통적인 시스템과 크게 달라지지 않을 것으로 예상한다. 먼저 개발자는 개인 개발 환경에 개발 시스템을 모두 구축하여야 한다. 구축 후 운영 시스템에 배포 전 테스트를 위한 환경도 직접 구축해야한다. 이렇게 구축된 환경은 시스템 운영 환경에서도 동일하게 고려되어야 하며 배포 시 우발적인 상황에 대해 대비해야 한다. 서비스가 진행 중인 인프라 환경의 운영 환경 관리를 위해 고려해야 할 사항이 많다. 먼저 각 물리 또는 가상 서버에 대해 관리를 별도로 해야 한다. 또한, 서비스가 배포되었을 때 사용되는 런타임 환경 및 미들웨어를 해당 서비스에 맞게 설치가 되었는지 확인해야 하며 이 또한 버전 별로 각각 관리가 되어야 한다. 하지만 그림 2-4에서 보면 알 수 있듯이 이를 PaaS로 전환하였

을 때 개발자 및 운영자가 고려해야 되는 사항이 크게 변경된다. 일단 먼저, 개인 개발환경은 구축이 되어야 한다. 물론 PaaS에서 제공해주는 범위에 따라 다르겠지만, PaaS의 경우 개발 환경을 지원해줄 뿐만 아니라 개발에 필요한 도구를 지원해주는 서비스도 존재한다. 그렇기 때문에 이 또한 선택적인 사항으로 볼 수 있다. 그리고 테스트 환경 또한 개인이 구축하는 것이 아닌 PaaS를 통해 제공될 수 있다. 지속적 통합(CI: Continuous Integration) 서비스인 젠킨스(Jenkins)와 동일하게 소스 코드 변경 사항이 있을 경우 내부적으로 테스트를 구동하고 이를 운영 환경에 배포해주는 역할을 서비스로 제공받을 수 있다. 물론, 여기서 PaaS를 사용하려면 앱을 배포하고 서비스를 사용하는 기본적인 PaaS에 대한 지식이 있어야 한다. 운영 환경을 관리하는 관리자의 경우 IaaS 설치 및 관리뿐만 아니라 PaaS를 설치 및 관리해야 하므로 기술적으로는 좀 더 복잡해졌다고 볼 수 있다. 하지만, 관리 측면에서 보았을 때 그림에서도 볼 수 있듯이 PaaS를 통해 IaaS 환경이 통합되는 것을 확인할 수 있다. 인프라 환경뿐만 아니라 각 웹 서비스에 사용되는 런타임 및 미들웨어 시스템도 통합되어 유지보수 및 관리가 편리하다. IaaS를 설치하는 데 어려움이 있겠지만 향후 유지보수를 고려해보았을 때 수많은 장점이 존재한다. 이때 가장 상단에 존재하는 최종 사용자는 전통적인 방법으로 제공받거나 PaaS로 제공 받을 때 크게 다르게 느껴지는 것 없이 동일한 방식으로 사용한다는 것이다. 하지만 사용자는 유동적이므로 이를 효율적으로 대처하기 위해서는 관리자가 얼마나 편리하게 사용할 수 있는지에 대한 고려가 필요하다.



[그림 2-3] 웹 시스템 제공 시 개발자 및 운영자 고려 사항 (전통방식 웹 시스템).



[그림 2-4] 웹 시스템 제공 시 개발자 및 운영자 고려 사항 (PaaS).

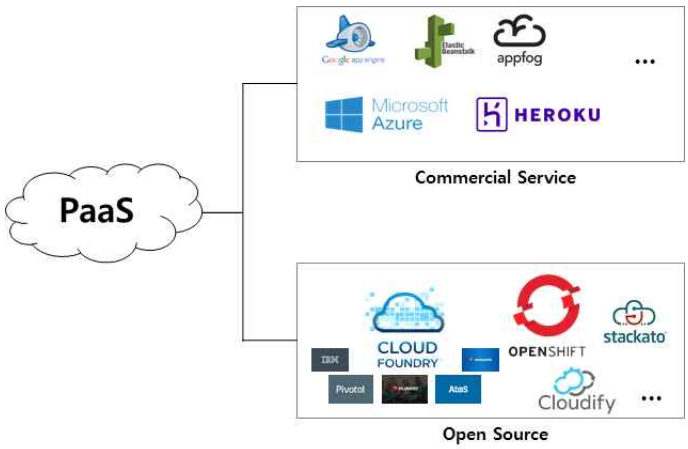
2) 플랫폼 서비스 현황

상용 및 오픈소스로 출시된 IaaS들이 안정화되면서 이를 활용하거나 독립적인 PaaS가 최근 많이 발표되고 있다. 대표적인 상용 서비스로는 그림 2-5에 정리한 바와 같이 구글 앱 엔진(Google App Engine), 마이크로소프트사의 애저(Azure), 아마존 엘라스틱 빈스토크(Amazon Elastic Beanstalk), 헤로쿠(Heroku), 앱포그(Appfog) 등이 있다. 구글 앱 엔진은 개발자들이 구글 서비스를 활용하여 웹 서비스를 배포할 수 있도록 도와준다. 주요 기능으로는 동적 웹, 쿼리 스토리지 등이 있으며 파이썬과 구굴에서 직접 만든 고우 언어를 중심적으로 지원하며 그 외 언어도 지원하고 있다. 윈도우 애저는 마이크로소프트사가 관리하는 데이터센터를 통해 제공되는 PaaS로 윈도우 서버에서 구동되는 앱을 중심으로 서비스를 시작하였다. 아마존에서 제공하는 엘라스틱 빈스토크는 여러 아마존 서비스를 동시에 사용하면서 아마존 웹 서비스 클라우드에 앱을 배치하여 관리할 수 있도록 도와주는 도구이다. 헤로쿠는 고객관계 관리 솔루션을 중심으로 한 클라우드 컴퓨팅 서비스를 제공하는 기업인 세일즈포스(salesforce)에서 인수하여 개발환경 및 프로세스 관리 기능을 갖추고 있는 PaaS이다. 마지막으로 앱포그는 오픈소스 PaaS 솔루션인 클라우드 파운더리 기반으로 상용화 PaaS를 제공하고 있는 서비스이다. 이러한 상용 서비스들은 일정 요금을 지불하면 개발자가 직접 개인적으로 개발환경을 구축하거나 운영환경을 구축하지 않아도 바로 서비스를 개발하고 배포할 수 있다는 장점이 존재한다.

시간적인 부분과 기술적인 부분에서 많은 요구 사항이 필요하지만, 상용 서비스 외 오픈 소스로 제공되는 솔루션들이 제공되고 있다. 대표적인 예로 클라우드 파운더리(Cloud Founry), 오픈 쉬프트(Open Shift), 클라우드파이(Cloudify), 스택카토(Stackato) 등이 출시되어 있다. 이러한 오픈소스 소프트웨어들은 하드웨어를 보유하고 있다면 직접 설치하여 공개뿐만 아니라 폐쇄형태의 PaaS를 내부적으로 구축할 수 있다는 장점이 있다. 클라우드 파운더리는 VMWare에서 개발하여 Linux 단체 프로젝트로 이관하고 현재 피보탈 기업에서 주로 관리하는 PaaS이다. IaaS 위에서 구동되는 PaaS로 마이크로

서비스 단위의 가상 인스턴스들로 구동된다. 오픈 쉬프트는 레드햇에서 주관하고 있는 소프트웨어이다. 컨테이너 기술을 바로 활용하여 하드웨어 위에서 구동되도록 설계되어 있으며 폐쇄 형태의 PaaS를 위한 엔터프라이즈 버전과 공개형 PaaS를 위한 버전이 공개되어 있다. 클라우드파이는 파이썬으로 만들어진 TOSCA(Topology and Orchestration Specification for Cloud Application) 기반으로 여러 기업이 후원하고 있으며 가상 IaaS 환경으로 앱을 배포하는 과정을 자동화할 수 있는 PaaS이다. 클라우드파이의 특징으로는 특정 IaaS에 의존하지 않는다는 점이다. 스택카도 또한 클라우드 파운더리와 비슷하게 원하는 하이퍼바이저에 설치하여 PaaS를 제공할 수 있다. 표 2-2는 상용 및 오픈소스 PaaS에 대해 배포되는 앱 종류와 배포 시 제공되는 자원, 배포 제공 언어를 정리한 것이다. PaaS에 배포되는 앱 종류는 대부분 웹 형식의 어플리케이션을 지원한다. 대부분 모든 솔루션에서 대중적 언어인 자바와 파이선, PHP에 대한 프로그래밍 언어 빌드를 지원하고 있다. 제공되는 자원에 대해서는 크게 컨테이너와 가상 머신 타입으로 구분될 수 있다. 컨테이너의 경우 하나의 운영체제 안에 독립적으로 생성된다는 장점이 있으며, 가상 머신은 물리적인 요소와 동일한 레이어가 구성된다는 장점이 있다. 상용 PaaS을 사용하거나 공개형 PaaS를 구축할 때 각 플랫폼의 특징을 정확히 판단 후 사용하는 것이 중요하다.

최근 국내에서도 PaaS에 대한 중요성을 느끼고 시장이 넓어지고 있다. KT에서 2016년 5월에 국내 기업 최초로 오픈소스 기반의 클라우드 PaaS인 데브팩(devpack)을 출시하였다. 데브팩은 KT IaaS인 유클라우드 비즈를 통해 제공되는 PaaS이다. 또한, 이와 비슷한 시기인 2016년 9월에 국내 금융권 기업인 코스콤(koscom)에서도 오픈소스 기반의 클라우드 PaaS을 활용하여 K PaaS-TA 서비스를 출시하여 현재 운영 중이다.



[그림 2-5] 상용 및 오픈소스 PaaS 종류.

[표 2-2] 상용 및 오픈소스 PaaS 사양 (Costache et al., 2017)

Solution	App types	Resource Isolation Support	Support Languages
AppEngine	Web	Containers	Java; Python; PHP; Go
Appfog	Web	Containers	Java; Python; PHP; Go; Node.js; Ruby
Amazon	Web; Data Analytics	Virtual Machine	Java; Python; PHP; Go; Node.js; Ruby; .Net
Azure	Web; Data Analytics	Virtual Machine	Java; Python; PHP; Go; Node.js; Ruby; .Net
Heroku	Web	Containers	Java; Python; PHP; GO; Node.js; Ruby; Scala and Play; Clojure
OpenShift	Web	Containers	Java; Python; PHP; Node.js; Ruby; Perl; Ceylon
Cloudify	Web	Virtual Machine	Based on TOSCA
Cloud Foundry	Web	Containers	Java; Python; PHP; Go; Node.js; Ruby; NET; Binary; Staticfile
Stackato	Web	Containers	Java; Python; PHP; Go; Node.js; Ruby; .NET; Perl; Clojure

제 3 절 공개형 PaaS

오픈소스 소프트웨어(OSS: Open Source Software), 공개형 소프트웨어는 소스 코드가 모두 공개되어있고 라이선스를 침해하지 않는 범위에서 누구나 자유롭게 사용할 수 있다. IT 트렌드를 기반으로 다양한 오픈소스가 확산되고 있어 최근에는 개인 개발자만이 아닌 기업들도 필수적인 요소로 변화되고 있다(박준완, 2015). PaaS를 제공하는데 있어 상용 서비스를 사용할 수 있지만 앞서 언급했듯이 클라우드파이, 오픈시프트, 클라우드 파운데이션 등 공개되어 있는 소프트웨어가 많이 존재한다. 그 중에서 가장 많이 이슈가 되고 있는 공개형 PaaS 소프트웨어로는 클라우드 파운더리와 오픈쉬프트가 있다. 이 두 종류의 소프트웨어는 아키텍처와 클라우드 서비스에 대한 상호작용 서로 다른 특징을 가지고 있다[표 2-3]. 클라우드 파운더리는 가상 머신을 기반으로 내부에 컨테이너 기술을 활용하여 구동되는 PaaS이고 오픈쉬프트는 가상 머신이 필요 없이 델타클라우드(Detacloud) 기술을 활용해 바로 컨테이너 형식으로 구동되는 PaaS 소프트웨어이다. 소프트웨어마다 특징이 존재하기 때문에 직접 PaaS를 구축하기 전 해당 특성을 잘 파악 후 적용해야 한다.

이번 연구에서는 클라우드 파운더리를 활용하여 공개형 PaaS를 구축하였다. IaaS 위에서 구동되기 때문에 클라우드 컴퓨팅을 적용할 때 이점이 존재한다. 특정 IaaS에 제한되어 있지 않고 대부분 모두 지원하고 있다. 공간정보 처리 서비스를 제공하기 위해서는 시각화를 위한 서버, 처리를 위한 서버 및 소프트웨어들이 설치되어야 한다. 아직 공간정보 분야에서는 이러한 기술 및 소프트웨어들이 클라우드 컴퓨팅 기술에 적용하기 위한 준비가 미흡하다. 그렇기 때문에 컨테이너 기술을 통해 바로 구동되는 오픈 시프트를 활용하기에는 어려움 점이 많을 것으로 여겨진다.

클라우드 파운더리는 Routung, Authentication, App Lifecycle, App Storage & Execution, Services, Messaging, Metrics & Logging 7개로 기능을 구분할 수 있다[그림 2-6]. Routing 기능에 있는 라우터(router) 구성 요소는 내부로 들어오는 트래픽을 App Liefcycle 기능 내부에 있는 구성 요소로 연결해주는 역할을 하며 로드밸런싱 기능이 내장되어 있다.

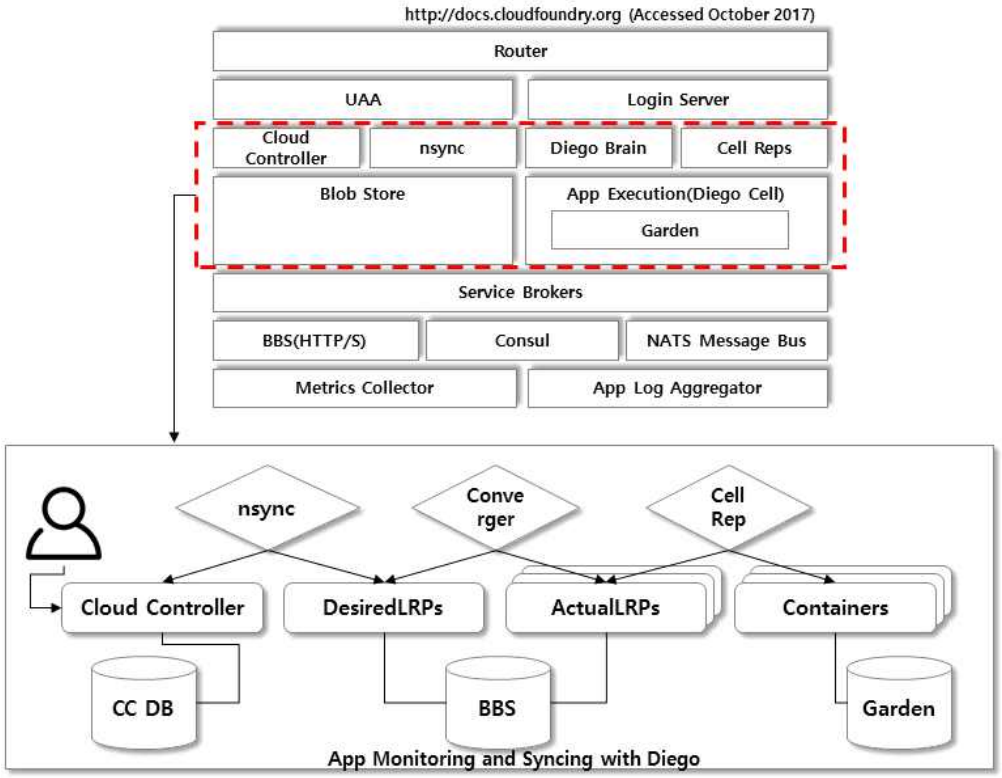
Authentication 기능은 PaaS 아이디 관리와 같은 인증과 관련된 구성요소이다. App Lifecycle 기능에 포함되어 있는 Cloud Controller 구성요소는 클라우드 파운더리의 가장 중요한 역할 중 하나로 어플리케이션 배포를 컨트롤하는 역할과 앱들의 공간을 구분 짓는 조직, 공간, 사용자 역할 서비스 등에 대한 기록을 보유하고 있다. nsync, Diego Brain, Cell Reps 구성 요소는 사용자가 앱 배포를 수행하였을 때 구성 요소들 간에 연결고리 역할과 앱들을 배포된 상태로 유지하기 위해 모니터링한다. App Storage & Execution은 실제 앱이 저장되는 구성요소인 Blob Store, 컨테이너가 생성되어 앱이 구동되는 구성요소 App Execution이 있다. Services 기능에 있는 Service Brokers는 PaaS에서 외부적 중요 요소 중 하나이다. 앱에서 사용되는 데이터베이스, 소프트웨어 등 외부 서비스들을 컨트롤하는 구성요소이다. Messaging과 Metrics & Logging에 포함되어 있는 구성요소는 대기열 메시징 시스템과 클라우드 파운더리를 구성하는 가상 인스턴스 통신 수단, 로그 수집 역할을 한다. 특히 메시징 시스템으로 사용되는 NATS는 클라우드 파운더리에서 모든 컴포넌트들이 사용하고 있는 통신프로토콜로 이 구성요소가 제대로 구동되지 않으면 PaaS 또한 구동되지 않는다.

각 구성 요소들은 구축될 때 가상 인스턴스들로 구성된다. 가상 인스턴스들은 마이크로 서비스 단위로 구성되므로 서비스를 추가하거나 삭제를 서비스 정지 없이 유연하게 할 수 있다. 그림 2-7은 사용자가 앱을 배포하였을 때 클라우드 파운더리가 수행하는 과정을 시퀀스 다이어그램으로 나타낸 것이다. 클라우드 파운더리를 활용해 앱을 배포하기 위해서는 커맨드 라인(Command Line)의 명령이 필요하다. 사용자가 앱 배포 명령어를 함과 동시에 클라우드 컨트롤러에서는 앱을 생성하기 위해 제일 먼저 앱을 배포할 공간을 생성한 후 저장 공간에 업로드하게 된다. 업로드하게 되면 Diego Cell 구성요소를 통해 Staging 단계로 진입한다. Staging 단계는 배포된 앱을 빌드하는 단계로 배포 시 설정된 빌드팩을 통해 빌드를 시작하고 빌드 시 연결된 서비스를 검색하여 서비스 사용 환경을 구성한다. Staging 단계에서 이상 없이 마무리가 된다면 앱 상태는 Running 단계로 변경이 되면서 클라우드 컨트롤러에 앱 상태를 보낸다. Running 상태를 한번 보내고 마무리되는 것이

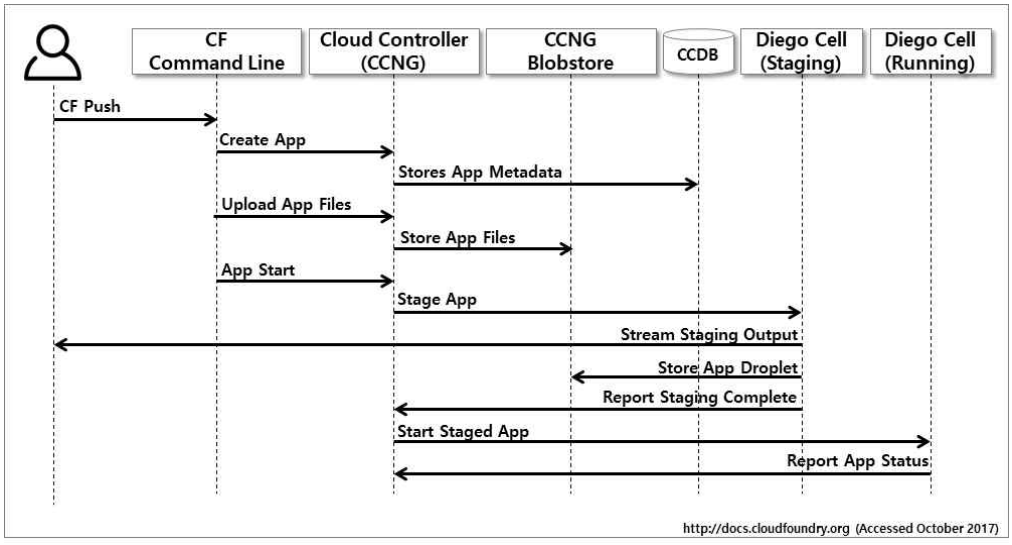
아니며, Diego Braing을 통해 주기적으로 앱 상태를 확인하여 클라우드 컨트롤러에 앱 상태를 보낸다.

[표 2-3] 클라우드 파운더리 및 오픈쉬프트 아키텍처와 상호운용 (Fowley et al., 2013)

	Cloud Foundry	OpenShift
Architecture	• Console pushes app to cloud, deployment, management / configuration through console • Controller runs as a cloud VM on the target IaaS • Controls all Cloudify spawned cloud VMs	• Divided into control plane (Broker) and messaging / application hosting infrastructure (Nodes). • Controller is command CLI shell, used to create apps. GIT for app management /deployment. • Gear is application container and a virtual server/node accessed via ssh
Clouds supported / Interoperability	• Supports AWS, vSphere, Openstack , Rackspace. • Private cloud is available.	• Uses Deltacloud • App runs on Red-Hat certied public cloud (needs Deltacloud support).



[그림 2-6] 클라우드 파운더리 아키텍처.



[그림 2-7] 클라우드 파운더리 앱 배포 과정.

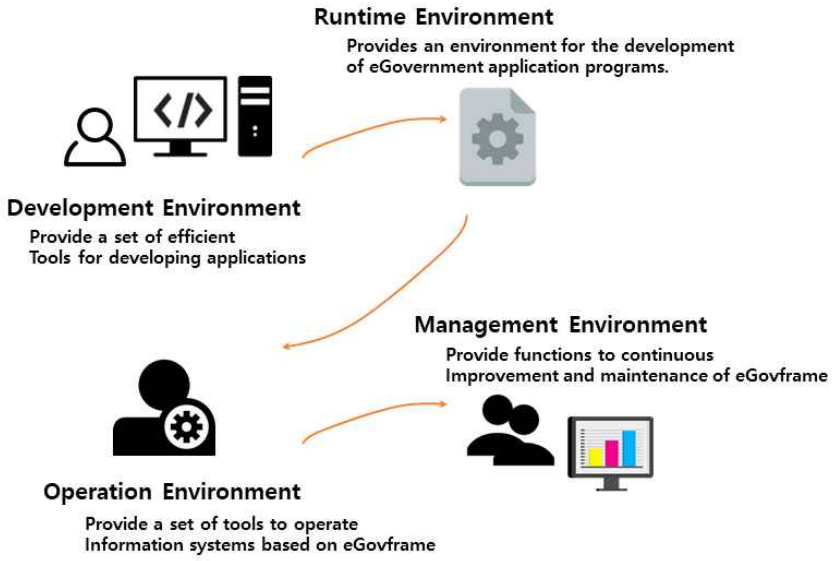
클라우드 파운더리 아키텍처에서 볼 수 있듯이 PaaS를 구축하기 위해 IaaS에서 수많은 가상 인스턴스를 설치하고 설정해야 한다. 그렇기 때문에 이 작업은 쉽지 않다. 그렇기 때문에 설치를 위한 소프트웨어가 제공되고 있다. 이를 BOSH라고 하며, BOSH를 사용하여 설치 설정 파일을 생성하여 클라우드 파운더리를 구축할 수 있다. 하지만 설정 파일을 생성하는 과정 또한 복잡하다. 클라우드 파운더리에 대한 개발을 지원하고 있는 기업에서는 복잡도를 조금이나마 해소하기 위한 추가 인터페이스를 지원해주고 있다. 대표적인 예로 피보탈 클라우드 파운더리(PCF: Pivotal Cloud Foundry)이다. 피보탈 클라우드 파운더리에서는 설치를 위해 Ops Manager를 제공하고 있으며, 설치 후 관리를 위한 Apps Manager도 제공하고 있다(Pivotal, 2017).

제 4 절 전자정부 표준프레임워크

1) 전자정부 표준프레임워크 개념

국내에서는 정부 3.0 정책에 맞춰 국민 개개인 또는 기업별로 맞춤형 서비스를 제공하는 것으로 목표하고 있다. 이는 투명성 있는 정부라는 큰 표제를 통해 국가가 보유하고 있는 데이터를 공개할 수 있는 범위 선에서 공개하고 2차 가공물로 활용할 수 있도록 제공하는 것이다. 이렇게 데이터를 공개하는 것과 별계로 국가 정보 시스템들에 대한 특정 업체 종속성, 비용증가, 중소기업에 대한 입찰 제한 등의 문제를 해결하기 위해 개발 프레임워크의 표준 정립으로 응용 소프트웨어 표준화, 품질 및 재사용성 향상을 목표로 전자정부 표준프레임워크에 대한 모델을 정립하고 관련 기술들을 2009년부터 개발하여 민간에게 제공하고 있다(이봉옥, 2009). 행정자치부 (2016) 발표에 따르면 2009년 표준프레임워크 기술 공개 이후 표준프레임워크 적용·지원 및 상용 SW 호환성 확인에 대한 성과로 공공사업 적용 648 사업, 상용 SW · 솔루션 호환 확인 96개 제품, 약 6천 명 개발자 교육 이후 및 오픈 커뮤니티를 통한 만 5천 명 개발자 참여 등 영향력 있는 성과를 보여주고 있으며, 국내뿐만 아니라 불가리아 등 해외 9개국 14개 사업에 적용하고 있다. 이러한

사례는 국내 IT 기업의 장점을 살리면서 단점을 극복한 대표적인 성공사례로 꼽고 있다(전형철과 강선무, 2013). 표준프레임워크는 그림 2-8에 정리한 것과 같이 네 가지 환경으로 구성되어 있다. 먼저, 개발 환경은(Development Environment) 개발자들에게 개발 툴을 제공하기 위한 환경으로 오픈소스로 제공되고 있는 설계, 테스트, 설정 관리, 개발 툴을 표준프레임워크에 맞게 추가 및 수정하여 제공하고 있다. 실행 환경(Runtime Environment)이 개발 시 가장 많이 사용되는 환경이다. 실행환경은 업무 실행에 필요한 공통 모듈로 화면, 업무, 데이터 처리와 모바일에 대한 화면처리도 지원하고 있다. 여기서 제공되는 모바일은 앱이 아닌 웹을 지원하고 있다. 추가사항으로 실행환경에는 공통 컴포넌트가 추가 개발되어 있다. 공통 컴포넌트는 전자정부 시스템 개발 시 공통으로 사용되는 기능을 실행환경 기반으로 프로젝트 구축 시 사용 가능하도록 독립적인 모듈로 개발하여 활용할 수 있도록 만든 컴포넌트이다. 항목별로 구분되어 있고 웹 공통 기술 136종과 모바일 웹 공통 기술 11종, 요소 기술 104종이 제공되고 있다[표 2-4].



[그림 2-8] 전자정부 표준 프레임워크 제공 환경.

[표 2-4] 전자정부 표준 프레임워크 공통 컴포넌트 목록
(전자정부 표준프레임워크, 2017)

Categories		Components
Technical components (136)	User authentication	3 components, including integrated user authentication, general login, etc.
	Security	8 components, including role/authorization, encryption/decryption, etc.
	Statistics	6 components, including access statistics, etc.
	Collaboration	28 components, including notice board, community management, etc.
	User support	55 components, including user management, FAQ/Q&A, etc.
	System management	27 compoents , including menu, log, system management, etc.
	Integration	4 compoents , including system access, mobile open API, etc.
	Digital asset management	5 components, including knowledge management, etc.
Mobile Technical components (11)		11 componments , including mobile menu, mobile real-time notice, etc.
Utility components (104)		104 compoents , including calendar, web editor, format convertor, etc.

전자정부 표준프레임워크 센터에서는 기술적 지원뿐만 아니라 이와 관련된 정책에 대한 지원 서비스를 도입하고 있으며, 공공 정보화 사업 참여시 이를 적극적으로 활용할 수 있도록 제공한다. 개발된 기술을 활용할 수 있는 무료 교육 지원을 하고 있으며, 표준프레임워크 기반으로 개발된 상용 솔루션에 대해 상호호환성 인증 제도를 도입하고 있다. 인증된 솔루션에 한해서 국내 소프트웨어 스토어인 씨앗(http://www.seart.kr)에 업로드된다. 공간정보 분야와 관련된 표준프레임워크 적용 연구 사례는 2015년부터 계속 진행됐다. 김광섭과 이기원(2015)은 공간정보 시각화를 중심으로 모바일 웹 환경에서 구동하기 위해 표준프레임워크를 활용하였고, Yoon et al. (2017)은 공간 처리와 관련된 시스템을 표준프레임워크를 활용하여 구현한 사례가 존재한다.

2) PaaS 및 표준프레임워크

전자정부 표준프레임워크 목표에 관한 내용은 클라우드 컴퓨팅과 매우 밀접한 관계를 가지고 있다. 그렇기 때문에 클라우드 컴퓨팅 시대를 맞이한 생태계에 맞춰 표준프레임워크가 향후 가야 할 방향 중 가장 큰 이슈 중 하나이며, 이러한 내용으로 전 세계적으로 관련된 연구 사례들이 존재한다. 특정한 목적을 두고 하둡(Hadoop)을 이용한 클라우드 컴퓨팅 기반 전자정부 틀을 제안한 연구로 미래의 전자정부 솔루션으로써 소개한 연구가 있으며(Mukherjee and Sahoo, 2010; Smitha et al., 2012), 전반적인 모델을 고려하여 전자정부와 클라우드를 고려한 사례도 존재한다(Khare et al., 2012). 인도에서는 서비스 모델 중 IaaS를 통해 개선할 수 있는 전자정부 서비스에 대한 아이디어를 제시한 바 있다(Dash and Pani, 2016). IaaS뿐만 아니라 PaaS를 개발자에게 필요한 기능을 서비스 형태로 제공하는 측면에서 접근하였을 때 전자정부 표준프레임워크는 빠른 서비스를 제공할 수 있는 밑바탕이 될 수 있다.

국내에서는 전자정부 표준프레임워크를 글로벌 경쟁력 강화하기 위해 오픈 PaaS 생태계로 진화하기 위한 프로젝트를 시작하였다(전형철과 강선무, 2013). 이와 관련된 결과물로 2016년도 후반에 파스타(PaaS-TA)라는 PaaS

를 구축할 수 있는 기술을 공개하였다. 이는 공개형 PaaS 소프트웨어인 클라우드 파운더리를 기반으로 전자정부 표준프레임워크를 적용한 PaaS를 구축할 수 있는 기술이다. 기본적인 아키텍처는 클라우드 파운더리와 동일하고 인프라 제어 및 관리환경, 실행환경, 개발환경, 서비스 환경, 운영환경으로 구성되어 있다[그림 2-9].

[그림 2-9] 국내 PaaS 솔루션 파스타(PaaS-TA) 구성도.

제 3 장 공개형 PaaS 기반 공간처리 서비스 설계 및 구축

제 1 절 시스템 개념

이번 연구에서 설계 및 구축한 공개형 PaaS 기반 공간처리 서비스는 향후 활용성과 확장성을 고려하였다. 활용성을 위해 모든 기술 스택을 오픈소스로 구성하였다. 특히, 오픈소스를 선택할 때 커뮤니티 활동이 가장 활발하면서 실제 업무 활용 시 제한이 없는 라이선스 중심으로 활용하였다. 시스템은 클라우드 환경과 시스템 환경 두 개로 나눠진다. 이번 장에서는 나뉘는 환경에 대해 설계 및 구축에 대한 내용을 정리하였다. 클라우드 환경은 IaaS 및 PaaS, 시스템 환경은 웹 표준 서버 및 공간정보 웹 시스템으로 구성된다.

첫 번째, 하드웨어 구성 및 클라우드 환경을 설계 및 구축한 결과이다. 클라우드는 배포 타입에 따라 공개(public), 폐쇄(private), 하이브리드(hybrid) 환경으로 서비스할 수 있다. 공간정보의 경우 일반적으로 공개될 수 없는 데이터가 많이 존재하기 때문에 이번 연구에서는 보안성을 중요하는 폐쇄 클라우드 환경으로 구성하였다. 하드웨어는 워크스테이션급 장비를 여러 대 활용하였다. 클라우드 서비스 IaaS 및 PaaS 관련 소프트웨어로는 오픈스택(OpenStack)과 클라우드 파운더리(Cloud Foundry)를 사용하였다.

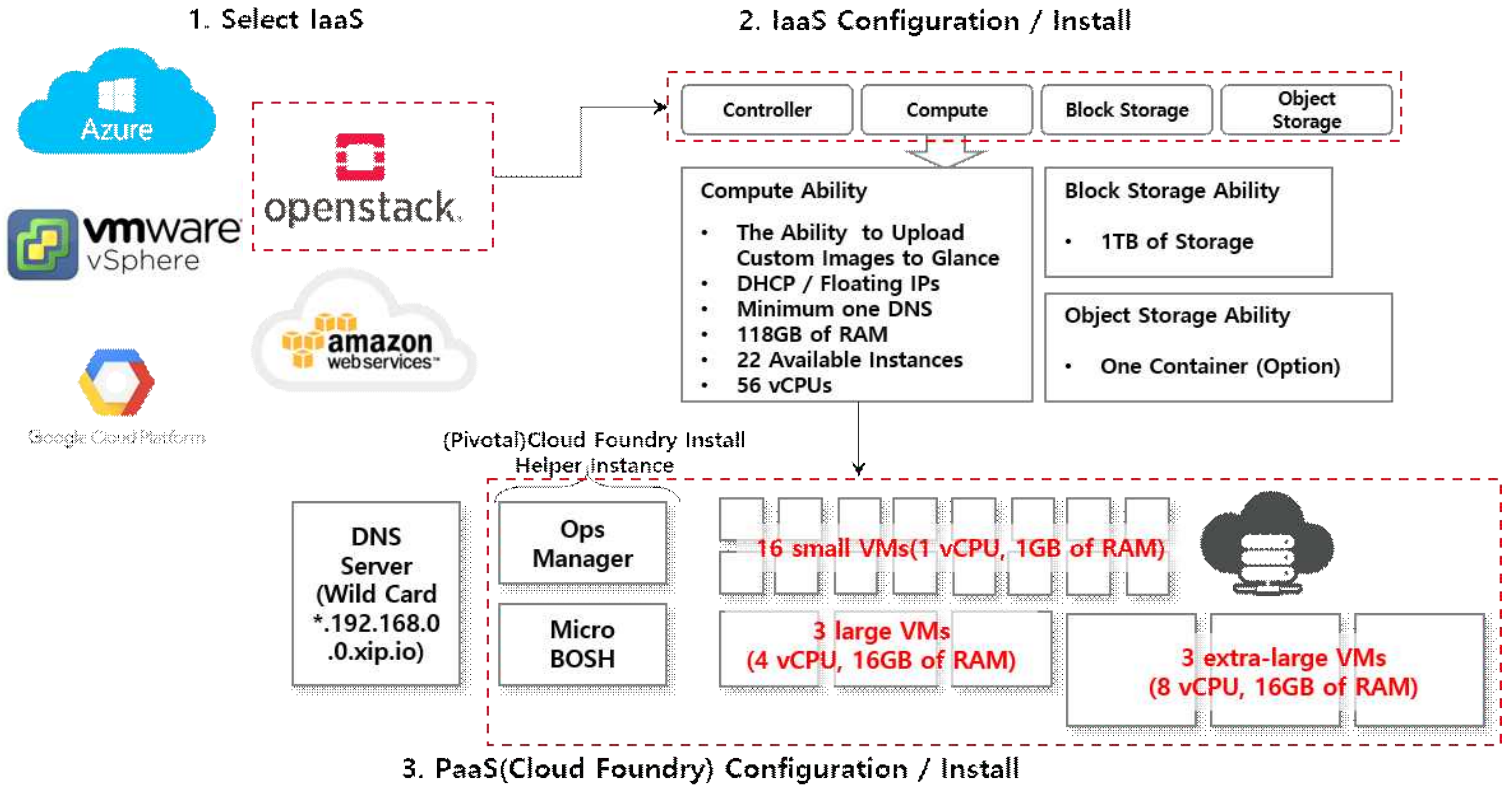
두 번째, 공간정보를 웹에서 시각화 또는 처리하기 위해서는 여러 방법이 존재한다. 그중에서 표준을 활용하는 것이 향후 활용성 및 확장성에 대한 장점이 존재한다. 특히, 이번 연구에서 제안하는 시스템의 경우 특정 기능에 중점을 둔 것이 아닌 일반적인 기능에 대해 초점을 맞춰 시스템을 설계 및 구축하였다. 가장 일반적인 기능으로 공간정보에 대한 시각화 및 처리를 생각할 수 있으며 이와 관련된 공간정보 웹 표준을 활용하기 위해 기존 존재하는 공간정보 시각화 및 처리 서버를 구축하고 이를 PaaS 환경에서 활용할 수 있도록 구성하였다. 활용된 웹 표준 서버는 GeoServer와 Zoo-Project이다. PaaS

환경에서 공간정보 웹 서비스에 대한 가능성을 확인하기 위해 실제 공간정보 융합 및 처리 웹 어플리케이션을 설계 및 구축하여 PaaS에 배포하여 기능에 대한 구동 결과를 확인하였다.

제 2 절 클라우드 환경 설계 및 구축

1) 공개형 PssS 환경 설계 및 구축

공개형 PaaS 환경을 구축하기 위해서는 크게 두 가지 방법을 존재한다. 하드웨어 바로 위에 컨테이너 기술을 사용하여 PaaS 환경을 구축할 수 있으며, 또한 클라우드 IaaS에서 PaaS를 구축하는 방법이다. 이번 연구에서는 IaaS에 PaaS를 구축하는 방법을 선택하였다. IaaS 환경에서 PaaS 환경을 구성하면 시스템이 컨테이너 기술을 사용한 것보다 상대적으로 복잡하지만, IaaS에서 제공되는 서비스를 통해 좀 더 확장성 있는 서비스를 제공할 수 있다. 그렇기 때문에 이번 연구에서는 이와 같은 방식을 채택하였다. IaaS 및 PaaS 환경을 구축하기 위해 사용된 소프트웨어로는 오픈스택과 클라우드 파운더리를 사용하였다. PaaS 환경을 구축하는 순서는 그림 3-1에서 볼 수 있듯이 크게 3가지로 나눠진다. 먼저, PaaS 환경을 설치할 IaaS를 선택한다. 선택 후 해당 IaaS에 맞춰 하드웨어를 구성하고 설치한다. 설치 시 가상 컴퓨팅 환경이 요구되는 사항으로 사용자 이미지 업로드 기능, DHCP(Dynamic Host Configuration Protocol) 및 유동 IP 생성 기능, 와일드카드(Wilde Card)로 구성된 DNS(Domain Name System) 서버, 최소 118GB 램(RAM), 최소 56개 vCPU, 최소 22개 인스턴스(instance)를 생성할 수 있는 환경, 블록 스토리지(Block Storage)를 생성할 수 있는 환경 최소 7가지가 구성되어 있어야 한다. 마지막으로 설치된 IaaS 환경 내부에 클라우드 파운더리 설치에 도움을 주는 BOSH를 통해 22개의 인스턴스가 생성되면서 PaaS 환경이 설치된다.



[그림 3-1] 공개형 PaaS 환경 구축 순서 예시.

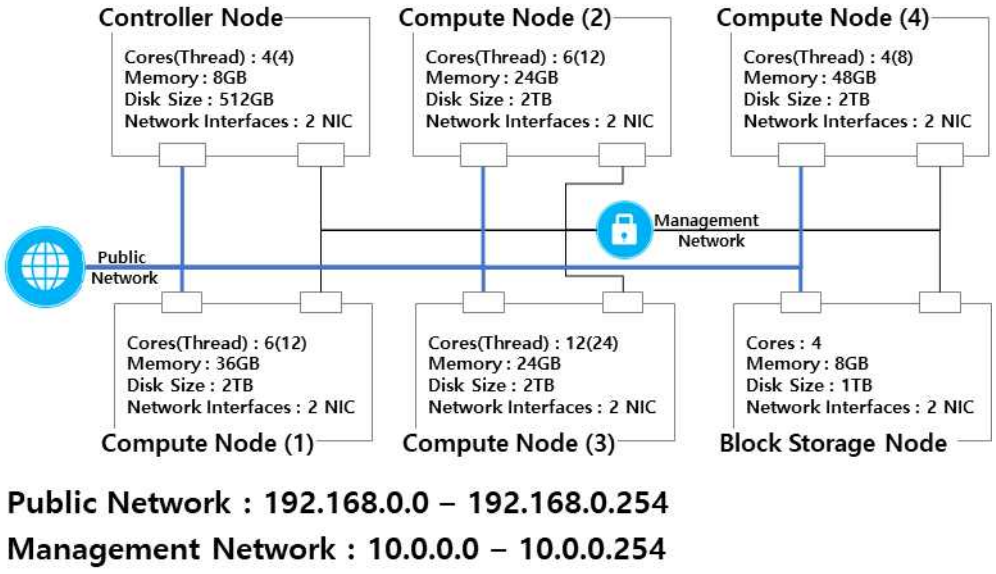
가) 오픈스택 기반 IaaS 환경 설계 및 구축

클라우드 파운더리 기반 PaaS 환경을 구축하기에 앞서 IaaS를 구축하거나 기존 서비스를 활용해야 한다. 제공되는 IaaS는 아마존 AWS, 마이크로소프트 애저(Azure), 오픈스택, vSphere 등 다양하게 존재한다. 이번 연구에서는 직접 설치하여 서비스할 수 있는 오픈스택을 활용하였다. 오픈스택을 활용하여 IaaS를 구축하기 위해서는 최소 각 하나의 컨트롤러(controller), 컴퓨트(compute) 노드가 필요하다. 추가사항으로 PaaS를 구축하기 위해 블록 스토리지 노드와 오브젝트 스토리지(Object Storage) 노드가 필요하다. 하지만 오브젝트 스토리지 노드의 경우 블록 스토리지로 대체 가능하므로 이번 연구에서는 제외하였다. 모든 하드웨어 노드는 두 개의 네트워크 인터페이스 카드(NIC: Network Interface Card)로 구성되어 있으며, Public Network와 Management Network로 구성된다. Public Network는 인터넷과 연결된 네트워크로 폐쇄 환경을 구성하기 위해 C 클래스 폐쇄형 IP 주소 대역 중 일부를 사용하였다. Management Network는 클라우드 노드들 간 빠른 통신을 위해 사용되는 네트워크로 A 클래스 폐쇄형 IP 주소 대역 중 일부를 사용하였다. 그림 3-2는 이번 연구에서 구성된 하드웨어 구성을 도식화한 것이다.

표 3-1은 이번 연구에서 구축한 IaaS 환경에서 CPU와 램에 대한 가상화 능력치를 이론적으로 계산한 결과값이다. 오픈스택은 별도 설정이 없을 경우 CPU는 16배, 램은 1.5배까지 가상화가 가능하다. 이번 연구에서 구성한 컴퓨트 노드는 총 4개이며 이를 합한 값은 CPU 56개, 램 132GB 이다. 이를 수치에 맞춰 계산하면 총 896 vCPU, 198GB vRAM을 사용할 수 있다. IaaS 환경이 구동되는 리소스와 가상 인스턴스 생성 시 필요한 노드에 성능에 따라 달라질 수 있으므로 이론값보다 낮은 컴퓨팅 능력을 갖출 수밖에 없다. 이번 연구에서 구축될 클라우드 파운더리 기반 PaaS에 대한 사양으로썬 충분한 가상 수치다.

오픈스택은 6개월 주기로 새로운 버전이 계속 출시되고 있으며 2017년 10월 기준으로 Pike 버전이 최신으로 출시된 상태이다. Pike 버전은 2017년 8월에 출시되었다. 공개형 PaaS 클라우드 파운더리를 설치하기 위해서는

BOSH 소프트웨어가 필요하다. BOSH는 가상 환경에서 인스턴스에 대한 관리를 자동으로 해주는 소프트웨어이다. 그렇기 때문에 BOSH는 IaaS에서 제공하는 API를 지원하고 있어야 한다. 공식적으로 BOSH가 구동되는 환경이 테스트 된 오픈스택 버전은 Newton, Mitaka, Liberty 이다. 이 중에서 이번 연구에서 활용한 오픈 스택 버전은 Mitaka 버전이다. PaaS을 구축하기 위한 오픈스택 기본 설치 구성으로는 신원 서비스(identity service), 이미지 서비스(image service), 컴퓨트 서비스(compute service), 네트워킹 서비스(networking service), 블록 스토리지 서비스(block Storage service)이다. 이외 추가로 인스턴스 생성 및 관리를 편리하게 해주는 대시보드 서비스(dashboard service)와 향후 인스턴스 간 데이터를 공유하기 위해 공유 파일 시스템 서비스(shared file Systems service)를 추가로 설치하였다.



[그림 3-2] 오픈스택 기반 PaaS 설치를 위한 하드웨어 구성.

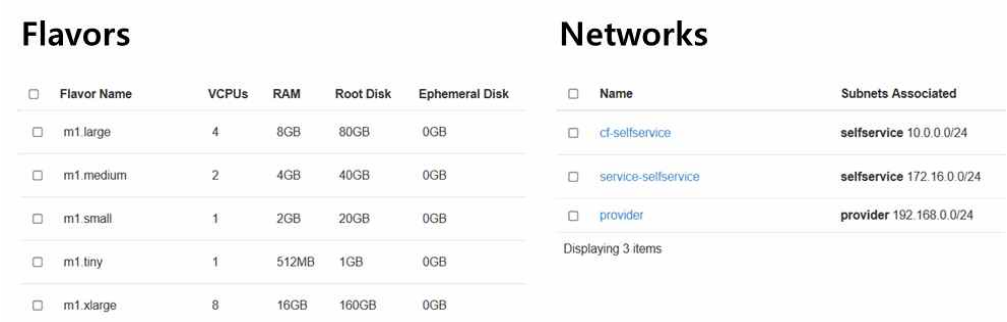
[표 3-1] IaaS 컴퓨트 노드 가상화 수치 계산 값

	CPU / vCPU(x16.0) (unit: Number)	RAM / vRAM(x.1.5) (unit: GB)
Total	56 / 896	132 / 198
Compute Node 1	12 / 192	36 / 54
Compute Node 2	12 / 192	24 / 36
Compute Node 3	24 / 384	24 / 36
Compute Node 4	8 / 128	48 / 72

그림 3-3과 그림 3-4는 오픈스택을 설치 후 대시보드 서비스인 Horizon을 통해 가상화할 수 있는 환경을 확인하고 PaaS를 위한 인스턴스 생성 크기 및 네트워크를 생성한 결과이다. 앞서 언급했듯이 클라우드 파운더리 기반 PaaS를 구축하기 위해서는 최소 22개의 인스턴스가 생성될 수 있도록 구성되어야 한다. 그림 3-3에서 볼 수 있듯이 이번 연구에서는 최대 100개까지 생성할 수 있도록 설정되어 있다. vCPU 및 메모리 또한 클라우드 파운더리를 설치하기 위한 최소 범위를 넘어서는 것을 확인할 수 있다. 하지만 메모리의 경우 기본적으로 사용되고 있는 용량으로 인해 이론적으로 표시되었던 수치보다는 낮게 나오는 것을 확인할 수 있다. 이 외에도 유동 아이피, 블록 스토리지 등 모두 생성할 수 있도록 구축하였다. PaaS가 설치되면 여러 인스턴스가 생성된다. 이때 생성되는 인스턴스에 대한 vCPU, 메모리, 디스크 크기는 Flavors를 통해 선택된다. 그림 3-4에 보이는 Flavors는 총 5개 인스턴스 생성 크기는 오픈스택을 설치하면 기본적으로 설정되어 있는 크기이다. 클라우드 파운더리는 기본 설정된 5개 Flavors를 활용하기 때문에 별도로 추가할 사항은 존재하지 않는다. 하지만 절대 이름을 변경하거나 세부적인 설정을 변경해서는 안된다. 오픈스택 설치 시 가상 네트워크 옵션으로 공급자 네트워크(provider networks)와 셀프 서비스 네트워크(self-service netowkrs)를 선택할 수 있다. 공급자 네트워크 옵션의 경우 레이어 2 서비스와 네트워크 분할을 통해 편리하게 활용할 수 있다. 셀프 서비스 네트워크의 경우 레이어 3

서비스로 복잡한 구조로 구성되어 있지만 훨씬 더 고급 기능을 활용할 수 있다. 이번 연구에서는 셀프 서비스 네트워크를 통해 여러 가상 네트워크를 생성하여 망을 분리하였다. 그림 3-4 네트워크 구성에서 볼 수 있듯이 총 3개의 네트워크로 구성하였다. Proivder 네트워크는 사설 IP 192 대역으로 어플리케이션을 접근하기 위한 네트워크이다. service-selfservice는 PaaS 구축 후 운영자가 관리할 미들웨어 소프트웨어(예: MySQL, MongoDB, RabbitMQ 등)가 서비스로 배포되기 위해 실제 구동이 되는 인스턴스들이 연결되는 네트워크이다. cf-selfservice는 클라우드 파운더리 PaaS가 실제 설치될 때 생성되는 마이크로 서비스 단위 인스턴스들이 연결되는 네트워크이다. 이렇게 구성함으로써 좀 더 체계적이고 구조적인 환경으로 PaaS를 구축할 수 있다.

[그림 3-3] 연구에 구축된 OpenStack IaaS 가상화 가용 능력 확인 결과.



[그림 3-4] 인스턴스 생성 크기 및 가상 네트워크 구성.

IaaS 구축 마지막 단계로 보안 그룹을 설정해야 한다. 보안 그룹은 외부 또는 내부 통신을 위한 필수적인 작업으로 필요 포트(port)만 개방하는 것이 중점 사항 중 하나이다. 표 3-2는 클라우드 파운더리 및 공간정보 처리 시스템과 관련된 보안 그룹을 정리한 사항이다. 클라우드 파운더리에 배포되는 앱은 기본적으로 보안 통신(HTTPS: Hypertext Transfer Protocol Secure) 형식으로 제공된다. 그렇기 때문에 외부에서 들어오는 보안 통신 포트 번호인 443을 개방해야 한다. 일반 서비스의 경우 일반 통신도 활용하기 때문에 HTTP 포트(80) 또한 개방되어야 한다. 또한 가상 라우팅, BOSH 통신, Ops Manager 통신을 위해 8443, 6868, 25555 포트도 개방해야한다. 그리고 클라우드 파운더리 구축 시 기본적으로 필요한 DNS 서버 인스턴스에서는 53번 포트를 개방해야 외부에서 도메인으로 접속할 수 있다.

공간정보 처리 시스템 관련된 포트는 MySQL 포트 3306, PostgreSQL 포트 5432, Tomcat 포트 8080, MongoDB 포트 27017이 개방되어 있어야 한다. 추가로 리눅스 인스턴스에 접근하거나 배포된 앱 컨테이너에 직접 접근하여 관리하기 위해서는 쉘(Shell)을 통해 가능하다. 이를 허용하기 위해 SSH(Secure Shell)인 포트 22를 개방한다.

[표 3-2] PaaS 구축 및 공간처리 서비스를 위한 IaaS 보안 그룹 설정

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	8443(HTTP ROUTING)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	6868 (BOSH)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	25555 (Ops Manager)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	53 (DNS)	0.0.0.0/0
Ingress / Egress	IPv4	UDP	53 (DNS)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	3306 (MYSQL)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	5432 (POSTGRESQL)	0.0.0.0/0
Ingress / Egress	IPv4	TCP	8080 (TOMCAT)	0.0.0.0/0

나) 클라우드 파운더리 기반 PaaS 환경 설계 및 구축

이번 연구에서는 오픈스택 기반 IaaS 내부에 직접 클라우드 파운더리를 설치하여 PaaS를 구축하였다. 클라우드 파운더리를 바로 사용할 수 있지만, 활용성 측면으로 보았을 때 피보탈(Pivotal)에서 제공하고 있는 PCF(Pivotal Cloud Foundry)를 이용하여 PaaS을 구축함으로써, 좀 더 안정적인 PaaS를 구축할 수 있다. 피보탈 업체에서 클라우드 파운더리를 관리하고 있기 때문에 PCF를 사용하더라도 유지보수가 지원되므로 업그레이드되었을 경우 이를 적용하기 편리하다. 클라우드 파운더리는 일반 소프트웨어와는 다르게 릴리즈 업데이트가 자주 일어난다. 그렇기 때문에 유지보수에 대해 충분히 고려가 되

어야 한다. 표 3-3은 이번 연구에서 구축된 PaaS 환경을 정리한 것이다.

[표 3-3] 피보탈 클라우드 파운더리 서비스 구축 환경

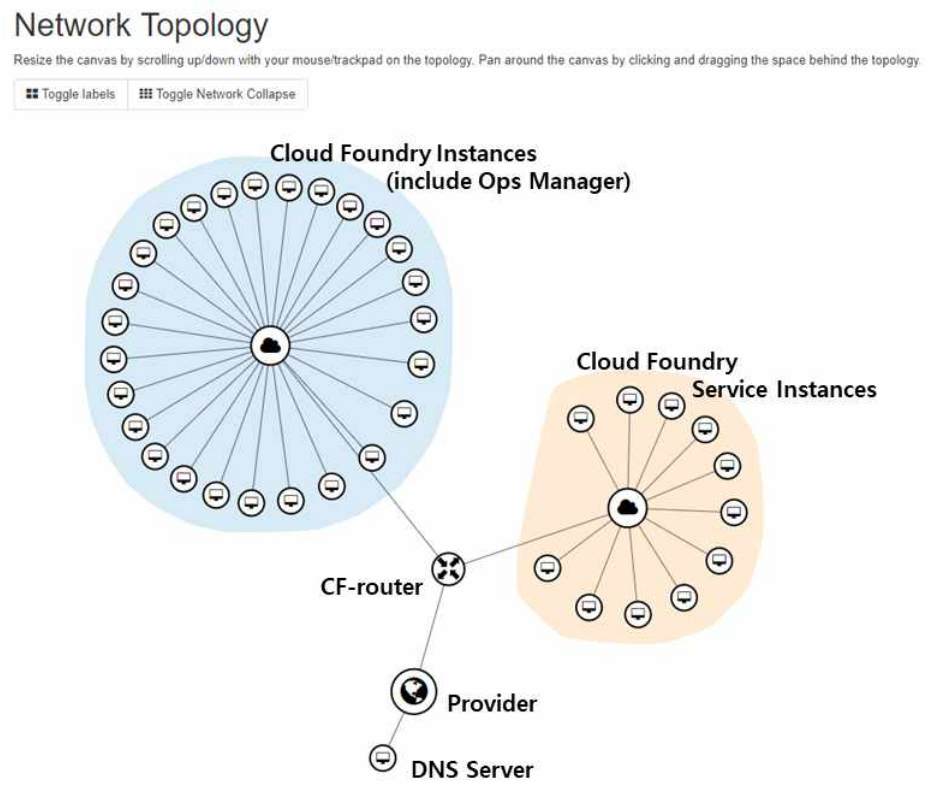
	Name	Version	Detail Note
Infrastructure as a Service	OpenStack	Mitaka	- Keystone - Nova - Glance - Neutron - Cinder - Manila
Platform as a Service	Ops Manager	1.11.10.0	- BOSH Installer
	Pivotal Elastic Runtime	1.11.11	- Cloud Foundry
	MySQL for PCF	1.10.2	- Cloud Foundry Service

Ops Manager는 클라우드 파운더리르 설치하기 위해 필요한 BOSH를 구성해주는 피보탈에서 제공하는 BOSH 설치 인스턴스이다. IaaS에 대한 기본 정보 및 네트워크 설정을 하면 BOSH가 설치된 디렉터(director) 인스턴스가 생성된다. 디렉터 설치가 끝나면 Pivotal Elastic Runtime 이미지를 업로드 후 BOSH 디렉터를 사용하여 클라우드 파운더리를 설치하도록 한다. Pivotal Elastic Runtime은 클라우드 파운더리를 설치하기 위한 설정 파일(YAML)을 생성하도록 도와주며, 향후 클라우드 파운더리가 구동되는 인스턴스들을 관리할 수 있는 인터페이스를 제공한다.

피보탈 클라우드 파운더리를 설치하게 되면 일반 클라우드 파운더리를 설치하는 것보다 더 많은 가상 인스턴스가 필요하다. 하지만 이는 vCPU 6개, RAM 8GB 정도이기 때문에 향후 유지보수 및 활용성을 보았을 때 적용가치가 충분하다. 그림 3-5는 이를 통해 구축된 PaaS에 대한 전체적인 네트워크 토폴로지를 오픈스택 대시보드를 통해 시각환 결과이다. 최상위 DNS 서버 인스턴스 한 대가 존재하며 이는 Provider 네트워크와 직접 연결되어있다. 구축한 IaaS가 폐쇄형으로 구성되어 있어 PaaS 또한 폐쇄 서비스를 제공한다. 네트워크 구성 시 관리를 고려하여 네트워크를 분리하기 위해 가상 라우터

(CF-router)를 생성하였다. 가상 라우터에는 사설 네트워크 두 개(172대역, 10대역)가 서비스되고 있다. 피보탈 클라우드 파운더리가 구동되기 위해서는 최소 22개의 인스턴스가 생성되어야 한다. 이번 연구에서는 Ops Manager, BOSH 인스턴스 포함 총 26개의 인스턴스를 생성하여 구축하였다.

추가된 네 개의 인스턴스는 앱 배포 시 중요한 역할을 하는 Diego Brain 한 개, Diego Cell 두 개, Doppler Server 한 개이다. 특히, Diego Cell의 경우 앱 배포 시 실제로 구동되는 컨테이너들이 생성되는 인스턴스이다. 그렇기 때문에 가장 크기가 큰 flavor로 구축되도록 설정되어 있으며, 기본 한 개이지만 이번 연구에서는 두 개를 추가 생성하였다. 생성된 인스턴스에 대한 개수(index), IP, Persistent disk, VM Type은 표 3-4와 같다. 모든 설정 타입은 클라우드 파운더리 설치 시 자동으로 설정된 것이지만 사용자가 원하는 크기로 변경할 수 있다.



[그림 3-5] 오픈스택 IaaS 환경 PaaS 구축 네트워크 토폴리지 결과.

[표 3-4] 피보탈 클라우드 파운더리 마이크로 서비스 인스턴스 생성 결과

VM	Index	IP	Persistent disk	VM Type
Consul	0	10.0.0.56	1GB	m1.small
NATS	0	10.0.0.57	None	m1.small
Etcd Server	0	10.0.0.58	1GB	m1.small
File Storage	0	10.0.0.59	100GB	m1.medium
MySQL Proxy	0	10.0.0.62	None	m1.small
MySQL Server	0	10.0.0.63	100GB	m1.large
Diego BBS	0	10.0.0.52	None	m1.small
UAA	0	10.0.0.53	None	m1.medium
Cloud Controller	0	10.0.0.54	None	m1.medium
HAProxy	0	10.0.0.60	None	m1.small
Router	0	10.0.0.61	None	m1.small
MySQL Monitor	0	10.0.0.68	None	m1.small
Clock Global	0	10.0.0.64	None	m1.small
Cloud Controller Worker	0	10.0.0.65	None	m1.small
Diego Brain	0	10.0.0.72	1GB	m1.small
	1	10.0.0.73	1GB	m1.small
Diego Cell	0	10.0.0.86	None	m1.xlarge
	1	10.0.0.69	None	m1.xlarge
	2	10.0.0.87	None	m1.xlarge
Loggregator Trafficcontroller	0	10.0.0.70	None	m1.small
Syslog Adapter	0	10.0.0.88	None	m1.small
Syslog Scheduler	0	10.0.0.74	None	m1.small
Doppler Server	0	10.0.0.75	None	m1.small
	1	10.0.0.76	None	m1.small

클라우드 파운더리를 구축한 후에는 PaaS에서 제공할 미들웨어 소프트웨어를 서비스로 제공하기 위한 작업이 필요하다. PaaS에서 서비스를 제공하기 위한 방법은 클라우드 파운더리에서 제공하는 서비스, 직접 서비스 브로커(Service Broker)를 구축하여 인스턴스와 연결, 단순 인스턴스를 통해 제공하는 방법 세 가지가 존재한다. 이번 연구에서는 이 세 가지 방법을 모두 사용하였다. 첫 번째 방법은 피보탈 클라우드 파운더리에서 제공하는 피보탈 네트워크(Pivotal Network) 포털에 업로드되어 있는 이미지 파일을 사용하는 것이다. 피보탈 네트워크에 있는 서비스 이미지들을 모두 자유롭게 사용할 수 있는 것은 아니지만 자유롭게 사용할 수 있는 서비스들은 활용할 경우 설치부터 관리까지 편리하게 사용할 수 있다. 이번 연구에서는 자유롭게 사용할 수 있는 서비스 중 MySQL을 사용하였다. MySQL은 향후 전자정부 표준프레임워크에서 사용될 데이터베이스로 사용될 서비스이다.

두 번째 방법은 직접 서비스 브로커를 구축하여 구동되고 있는 인스턴스 또는 컨테이너와 연결하는 방법이다. 이 방법은 기존 미들웨어 소프트웨어를 PaaS로 변경하기 위해 브로커를 설계하고 구현해야 하는 번거로움이 있다. 하지만 모든 소프트웨어가 클라우드 파운더리를 통해 제공될 수 없기 때문에 사용자 정의 방법이 필요하다. 이번 연구에서 사용되는 데이터베이스는 공간정보 기능이 특화되어 있는 관계형 데이터베이스인 PostgreSQL과 비 관계형 데이터베이스인 MongoDB이다. PostgreSQL은 이번 연구에서 설계 및 구축된 모든 앱에서 활용되며, MongoDB는 공간정보 및 공공데이터 융합 앱에서 공공데이터를 저장하는 용도로 사용된다. 서비스 브로커를 직접 구축하여 앱으로 배포 후 서비스로 활성화함으로써 활용되는 앱들이 서비스와 연결되었을 때 임의에 데이터베이스를 생성하거나 삭제하는 역할을 자동으로 수행해준다.

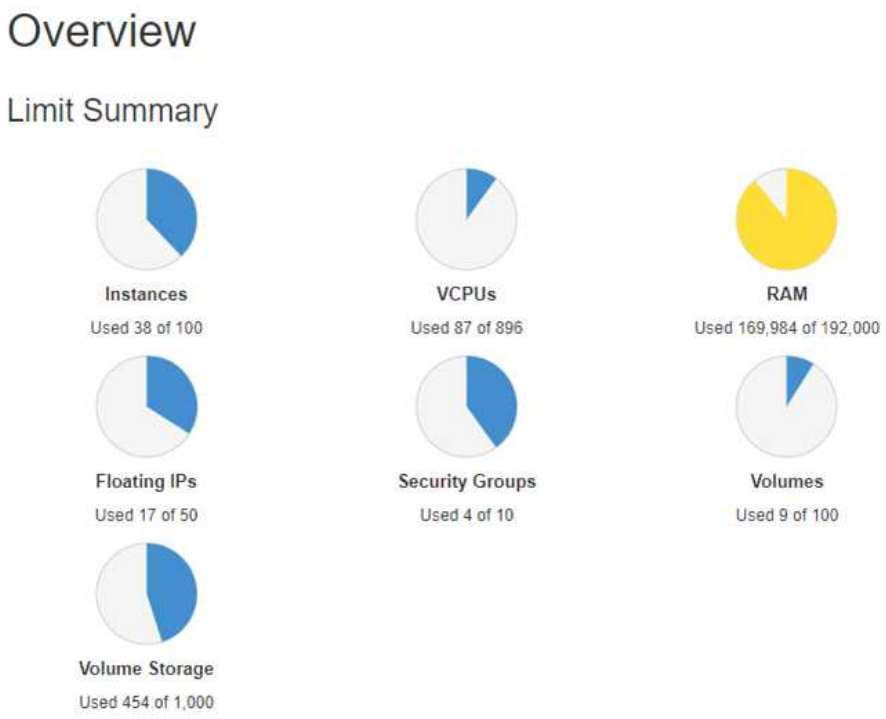
마지막으로 PaaS 환경에서 작업하지 않고 인스턴스로 직접 제공하는 것이다. 이는 제일 간편한 방법이지만 앞서 설명된 방법처럼 서비스를 관리하는 부분에 있어 수동으로 이루어져야 한다. 하지만 모든 서비스가 서비스 브로커를 구축할 수 있는 것은 아니기 때문에 이 또한 배제할 수 없는 방법 중 하나이다. 이번 연구에서는 공간정보에 특화된 미들웨어인 공간 웹 표준 시각화

및 처리 서버에 대해 이와 같이 제공하였다. 표 3-5는 PaaS에서 제공될 서비스를 위해 오픈스택 내부에 생성된 인스턴스를 정리한 것이다. 총 10개의 인스턴스가 생성되었으며 피보탈 클라우드 파운더리에서 제공하여 구축한 MySQL을 경우 단순히 데이터베이스만 설치되는 것이 아닌 관리를 위한 모니터링 서버, 향후 확장성을 위해 Proxy 서버가 추가로 생성된 것을 볼 수 있으며, 서비스 브로커 또한 인스턴스로 생성된 것을 확인할 수 있다. 제공되지 않은 미들웨어의 경우 직접 서비스로 구축하기 위해 인스턴스를 생성하였으며, 이 중 공간정보 처리 서버는 향후 확장성을 고려하여 HAProxy 서버를 구축하여 운영하도록 설계하였다.

[표 3-5] PaaS 제공을 위한 미들웨어 인스턴스 생성 결과

VM	Index	IP	Persistent disk	VM Type
Services (Provided Pivotal Cloud Foundry)				
MySQL Server	0	172.16.0.51	100GB	m1.large
Proxy	0	172.16.0.52	None	m1.medium
Monitoring	0	172.16.0.53	None	m1.small
Broker for MySQL	0	172.16.0.54	None	m1.small
Services (Custom / Manuual)				
GeoServer	0	172.16.0.11	100GB(NFS)	m1.medium
MongoDB	0	172.16.0.8	None	m1.medium
HAProxy for ZooProject	0	172.16.0.15	None	m1.small
Zoo Project	0	172.16.0.3	100GB(NFS)	m1.large
	1	172.16.0.14	100GB(NFS)	m1.large
	2	172.16.0.16	100GB(NFS)	m1.large

그림 3-6은 공간정보 처리 시스템을 PaaS에서 구동하기 위해 피보탈 클라우드 파운더리 구축 후 오픈스택 가상화 사용 현황을 대시보드를 통해 시각화 한 결과이다. PaaS를 구축하는 방법은 다양하게 구성할 수 있다. 이번 연구에서 설계 및 구축된 PaaS는 여러 방법 중 한 가지로 볼 수 있으며 생성된 인스턴스 중 미들웨어 서비스를 제공하기 위한 인스턴스 10개를 제외하고는 공간정보 처리 시스템뿐만 아닌 일반적인 PaaS를 위한 기본적인 사양이다. 인스턴스는 총 38개를 생성하였고, vCPU는 87개를 사용하였다. 가장 많이 사용된 메모리는 약 169GB 정도 사용하였고 Provider 사설 IP와 연결된 Floating IP는 총 17개를 사용하였다. Floating IP는 주로 서비스로 사용될 미들웨어 소프트웨어 인스턴스에 연결되어 있다. 보안 그룹은 클라우드 파운더리, 서비스 그룹 이렇게 2개로 나눠 각각 알맞은 인스턴스에 연결하였다. 공간정보 처리를 위해 파일 공유를 위해 사용된 네트워크 파일 시스템으로 오픈스택 서비스 마닐라(Manila)를 제외하고 클라우드 파운더리를 위한 블록 스토리지 볼륨은 총 9개가 사용되었으며 사용된 크기는 약 400GB 정도이다. 그림에는 포함되어 있지 않지만, 마닐라 서비스 또한 별도 인스턴스가 생성되어 관리되며 이는 블록 스토리지 100GB를 할당하여 향후 공간처리 서비스에 사용되는 영구 저장소로 활용되고 있다. 이는 IaaS에 구축된 결과이며, PaaS로 활용하기 위해서는 네트워크 파일 시스템 또한 서비스로 제공되어야 한다. 이번 연구에서 구축된 PaaS에서는 네트워크 파일 시스템이 포함되어 있다.

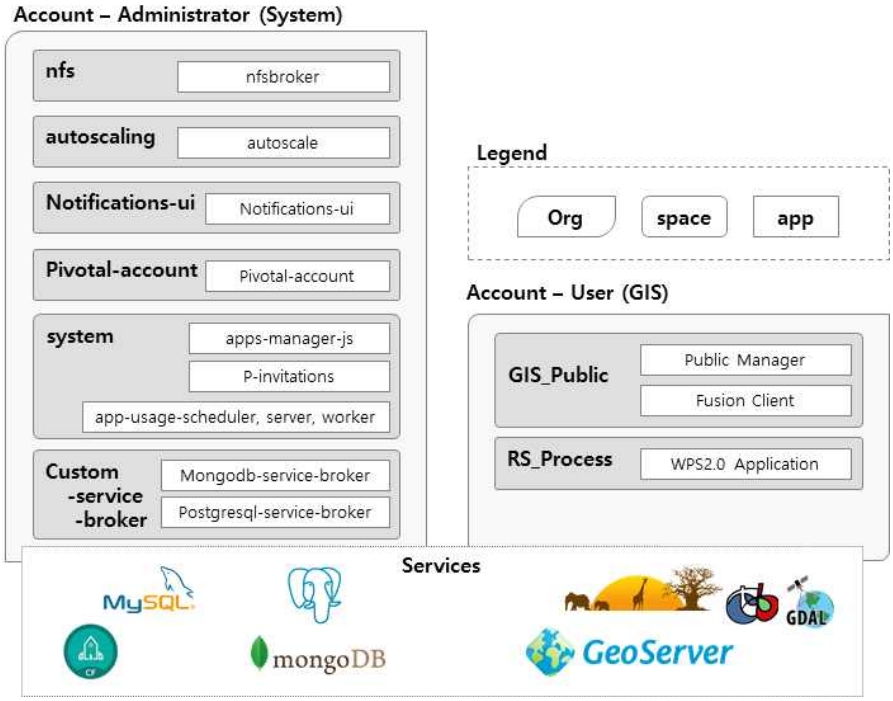


[그림 3-6] 피보탈 클라우드 파운더리 구축 후 오픈스택 가상화 사용 현황 시각화.

2) 공간정보 처리 시스템 서비스 및 앱 환경 설계

PaaS 구축이 완료됨으로써 개발된 어플리케이션을 배포하고 웹에서 활용할 수 있다. PaaS에 앱이 배포되는 공간은 크게 Org, Space, App으로 구분되어 있다. 구분되어 있는 공간에 대해 명확한 의미를 알고 배포가 되어야 좀 더 효율적인 앱 관리가 이루어질 수 있다. 그림 3-7은 이번 연구에서 설계 및 구축된 PaaS에서 Org, Space, App, 서비스 구성을 그림으로 도식화한 결과이다. 클라우드 파운더리가 구축되면 기본적으로 System 이라는 Org 그룹이 생성되어 있다. 이는 시스템 전반적인 관리를 위해 만들어진 그룹이다. 이 그룹 내부에 향후 PaaS 시스템 관리에 필요한 사항을 추가할 수 있다. 피보탈 클라우드 파운더리를 사용하여 PaaS를 구축할 경우 기본적으로 추가되는 서비스 및 앱들이 존재한다. 이러한 서비스와 앱들을 클라우드 파운더리에서

는 커맨드 라인 인터페이스(CLI: Command Line Interface)로 모두 확인해야 하지만 피보탈에서 제공하는 클라우드 파운더리는 그래픽 사용자 인터페이스(GUI: Graphic User Interface)를 제공해주며 오토스케일과 NFS 서비스가 기본으로 구축되어 있다. 피보탈에서 사용자 인터페이스를 제공하기 위해 기본적으로 배포되어 있는 앱들이 존재한다. 이러한 앱들은 모두 System Org에 여러 공간(Space)들로 배포되어 있다. 이는 피보탈 클라우드 파운더리를 설치할 때 추가 설정하여 제외할 수 있다. 그래픽 인터페이스를 제공하고 있는 apps-manager-js 앱이 배포되어 있으며, PaaS 시스템 상황을 점검하는 P-invitations, app-usage-scheduler, app-usage-server, app-usage-worker 앱들이 클라우드 파운더리를 설치할 때 자동으로 설치된다. 또한, 계정을 관리하기 위한 인터페이스를 별도로 제공하는데 이는 Pivotal-account 공간 안에 Pivotal-account 앱으로 배포되어 있다. 그렇기 때문에 사용자 계정 관리 또한 편리하게 추가 및 관리를 할 수 있다.



[그림 3-7] 구축된 클라우드 파운더리 PaaS에서 Org, Space, App 구분에 따른 공간정보 처리 서비스 구성도.

PaaS를 관리 및 유지하기 위해 설치되어 있는 앱 외에도 사용자가 개발한 앱 배포 시 활용할 수 있는 서비스가 추가로 존재한다. 기본적으로 구축되어 있는 서비스를 확인해보면 먼저 네트워크 파일 시스템인 NFS 서비스, 배포된 앱들을 네트워크 트래픽 및 CPU 점유율에 따라 확장 및 축소를 할 수 있는 오토스케일링(auto scaling) 서비스가 기본으로 설치되는 서비스이다. 이렇게 배포된 앱들은 모두 Diego Cell 인스턴스에 컨테이너(container) 방식으로 배포가 되어 있다. 기본으로 구축된 서비스들 외 공간정보 처리 서비스를 구동하기 위해 활용되는 미들웨어 소프트웨어 또한 서비스로 활용되기 위해 구성되어야 한다. 이번 연구에서 사용되는 미들웨어 소프트웨어로는 MySQL[1], MongoDB[2], PostgreSQL[3], GeoServer[4], Zoo-Project[5], OTB[6], GDAL[7]이다. 앞서 언급했듯이 PaaS에서 활용하기 위한 서비스를 구축하는 방법은 크게 세 가지로 구성되어 있다. 이번 연구에서는 이 세 가지를 모두 활용하였다. 세 가지 방법 중 공통점으로는 서비스로 제공하기 위해서는 해당 미들웨어 소프트웨어가 컨테이너 또는 인스턴스 형식으로 구축이 되어 있어야 한다. 예외적으로 클라우드 파운더리에서 직접 서비스를 제공하거나 제공할 수 있도록 구축한 경우 BOSH를 통해 미들웨어 시스템 또한 자동으로 생성하도록 구성할 수 있다. 이번 연구에 활용되는 미들웨어 소프트웨어 중 MySQL은 피보탈에서 서비스 형태로 제공하기 때문에 Ops Manager를 통해 서비스를 구축하였다. 그림 3-8은 MySQL 서비스를 구축한 결과이다.

---

1) MySQL: 오픈소스 관계형 데이터베이스 (https://www.mysql.com/)

2) MongoDB: 도큐먼트 지향 비관계형 오픈소스 데이터베이스 시스템으로 JSON과 같은 동적 스키마형 문서들을 선호 (https://www.mongodb.com)

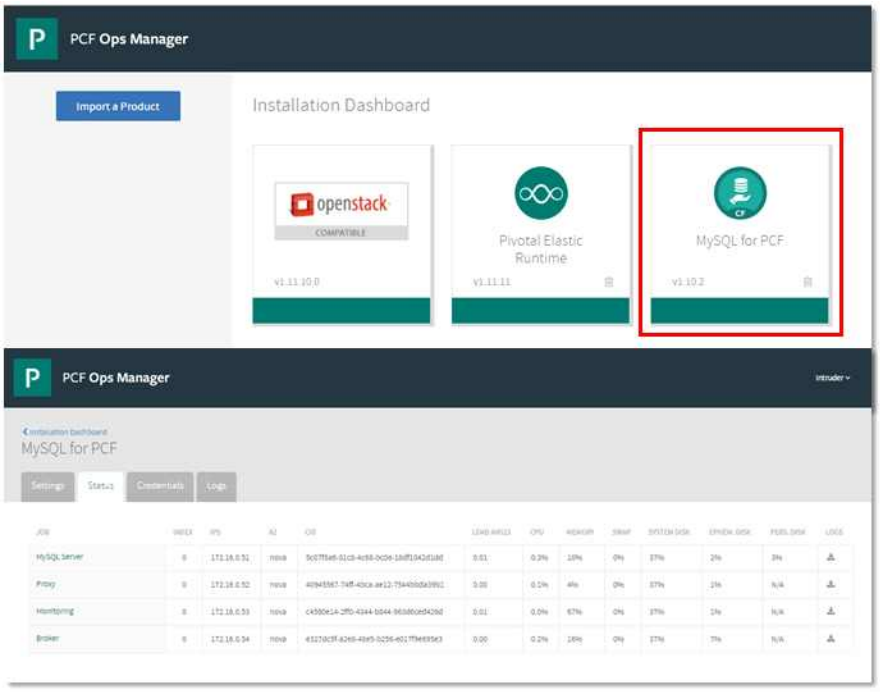
3) PostgreSQL: 오픈소스 객체-관계형 데이터베이스 시스템으로 공간정보 데이터를 저장할 수 있는 확장 기능을 제공 (https://www.postgresql.org)

4) GeoServer: 공간정보를 시각화 및 공유할 수 있는 Java 기반 오픈소스 서버 소프트웨어 (https://geoserver.org)

5) Zoo-Project: 공간 웹 처리 표준인 WPS 인터페이스를 제공하는 오픈소스 WPS 플랫폼으로 Kernel, Services, API, Client 로 구분되어 있음 (http://www.zoo-project.org/)

6) OTB: 위성영상을 시각화하거나 처리할 수 있는 기능이 탑재된 오픈소스 소프트웨어 (https://www.orfeo-toolbox.org/)

7) GDAL: 공간정보 데이터를 서로 변환하거나 원하는 형태로 자를 수 있는 기능이 탑재된 입·출력 오픈소스 소프트웨어 (http://www.gdal.org/)

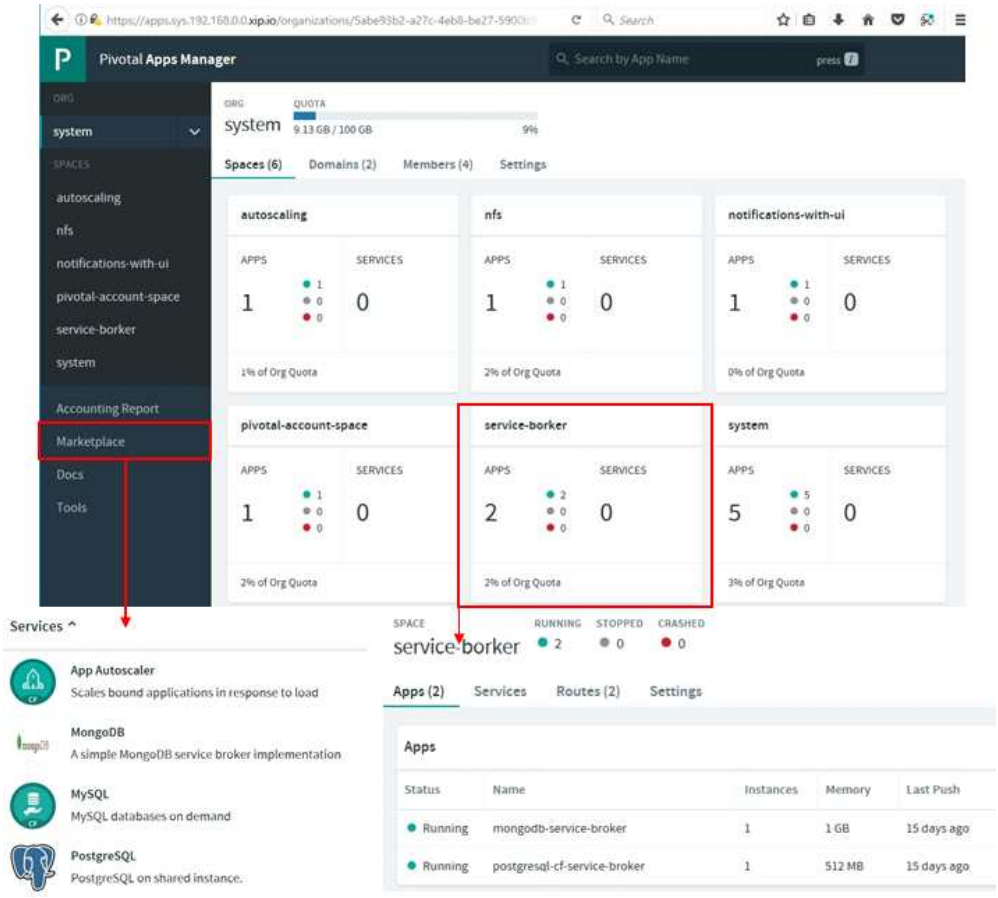


[그림 3-8] MySQL 서비스 구축 결과.

그 외 데이터베이스 소프트웨어인 PostgreSQL과 MongoDB의 경우 직접 서비스 브로커(Service Broker)를 PaaS에 앱 형태로 배포하여 활용할 수 있도록 구축하였다. 클라우드 파운더리에서 제공하는 서비스 설치 방식은 서비스 브로커가 인스턴스로 생성되어 제공되지만, 별도로 서비스 브로커를 구축할 경우 이를 선택할 수 있다. 이번 연구에서는 PaaS 위 앱 형태로 배포하여 서비스를 활성화하였다. PaaS 위에 구축할 경우 관련된 서비스 브로커를 PaaS에서 관리할 수 있다. 그림 3-9는 피보탈 클라우드 파운더리에서 제공하는 Ops Apps Manager를 통해 배포된 PostgreSQL과 MongoDB 서비스 브로커 앱에 대한 결과이다. service-broker 이름으로 공간(Space)를 생성 후 내부에 스프링 프레임워크 기반으로 만들어진 PostgreSQL 서비스 브로커 앱과 MongoDB 서비스 브로커 앱을 배포하였다. 이 앱은 PaaS에서 배포된 앱들이 서비스를 바인딩(Binding)할 때 앱에서 사용할 수 있는 전용 데이터베이스를 자동으로 생성해주는 역할을 하는 서비스 브로커이다. 언바인딩(unBinding)되

었을 때에는 데이터베이스를 삭제해주는 기능도 포함되어 있다. 이렇게 하여 PaaS에서 직접 관리가 가능한 서비스로 오토스케일링, MongoDB, MySQL, PostgreSQL 총 4개의 서비스가 구축된 것을 마켓플레이스(Marketplace)를 통해 확인할 수 있다. 공간정보 표준과 관련된 미들웨어 서버 소프트웨어들은 별도 서비스 브로커를 구축하지 않고 인스턴스에서 직접 제공하도록 하였다. 사용자 관리 및 구분이 필요할 경우 서비스 브로커를 추가 구축하여 제공하는 것이 이점이 많지만 이번 연구에서는 별도 사용자 관리를 고려하지 않았으므로 공간정보 웹 표준 서비스 제공 시 관련 URL만 가지고 서비스할 수 있으므로 PaaS 서비스로 구축하지 않았다. 향후 구축을 위해서는 MongoDB 또는 PostgreSQL 서비스와 동일하게 서비스 브로커를 설계 및 구축하여 배포 후 활용하면 사용자 관리 측면에서 효율을 극대화할 수 있다.

직접 개발한 앱을 배포하기 위해서는 별도 Org를 생성하여 관리하는 것이 효율적이다. PaaS에서 구동되는 앱을 배포하기 위해 생성한 Org 이름은 GIS이다. 하나의 Org는 몫(Quota)이라고 하는 메모리를 할당하여 제한을 둘 수 있으며 이번 연구에서는 Org마다 최대 50GB 메모리를 사용할 수 있도록 구성하였다. 이 설정은 클라우드 파운더리 설치 전 설정을 통해 확장 및 축소할 수 있다. 내부에는 여러 공간(Space)를 생성할 수 있으며 공간에서 사용하는 앱에 대한 메모리 크기는 Org Quota에 초과하지 않은 선에서 자유롭게 사용할 수 있다. 한 공간에 모든 앱을 배포할 수 있지만 각 주제에 맞춰 생성하는 것이 관리에 효율적이다. 이번 연구에서 설계 및 구축된 앱은 공간정보 및 공공데이터를 융합하는 웹 앱과 위성영상을 처리하는 웹 앱으로 구분된다. 그렇기 때문에 GIS_Public과 RS_Process로 구분하여 관련된 앱을 배포하도록 하였다. 하나의 공간에서는 서비스들을 공유하도록 되어 있다. 그렇기 때문에 공간 내부에 여러 앱들이 배포되고 그 앱들이 같은 데이터베이스 및 테이블을 공유해야 한다면 반드시 같은 공간 내부에 앱을 배포해야한다.



[그림 3-9] PostgreSQL / MongoDB PaaS 서비스 구축 결과.

제 3 절 PaaS 기반 공간정보 웹 표준 및 웹 시스템 설계 및 구축

공간정보 웹 시스템을 제공하기 위해서는 데이터를 시각화하는 방법과 처리하는 방법에 대해 먼저 고려되어야 한다. 공간정보 분야에서 사용되는 지리데이터들은 크게 벡터와 레스터로 구분될 수 있다. 벡터 데이터는 지형도 형태로 제공되며 점, 선, 면으로 이루어진 데이터로 관심지점(POI: Point of Interest), 건물, 길 등을 표현할 수 있다. 일반인에게 제공될 때는 캐드(CAD) 형식 데이터 파일 구조인 DXF(Drawing exchange Format)으로 제공되거나 공간정보 벡터형 파일구조에 특화된 Shape 파일로 제공된다. 레스터 데이터는 이미지 데이터라고 볼 수 있다. 일반 지도 서비스에서 많이 사용되는 항공

지도 또는 위성 지도 데이터들을 예로 들을 수 있으며, 이러한 데이터는 여러 장의 데이터들을 타일 형식으로 전처리 가공되어 제공하는 것이다. 일반인에게 제공되는 레스터 원본 데이터는 이미지 파일 포맷 중 손상이 없는 TIFF(Tagged Image File Format) 형태로 제공되며 일반 TIFF 형식에 지리적 해더 정보를 포함하여 GeoTiff 형식으로 제공된다. 이렇게 제공되는 벡터, 레스터 데이터는 웹상에서 바로 시각화할 수 없는 형태의 데이터들이다. 그렇기 때문에 이를 시각화하기 위해서는 처리가 필요하다. 향후 시스템에 대한 확장성이나 활용성을 고려할 경우를 위해 표준화된 방법을 수용하는 것이 적절하다. 공간 정보 웹 표준은 전 세계 여러 기관, 기업, 대학이 참여하고 있는 세대 최대 공간정보 산업 표준화 추진 기구인 OGC(Open Geospatial Consortium)에서 관리하고 있으며, 대표적인 서비스로는 웹 지도 서비스(WMS: Web Map Service), 웹 피처 서비스(WFS: Web Feature Service), 웹 지도 타일 서비스(WMTS: Web Map Tile Service), 웹 처리 서비스(WPS: Web Process Service)가 있다. 이러한 표준은 공간정보를 웹에서 시각화하거나 처리할 때 공간정보에 대한 특성을 데이터를 접근하고 시각화할 수 있는 방법에 대해 기술하고 활용할 수 있도록 인터페이스를 정리하고 있다. 이러한 표준 스팩에 맞춰 활용할 수 있도록 제공하는 오픈소스 소프트웨어들이 출시되어 있다. 먼저, 데이터 시각화를 위해 데이터를 관리하고 처리해주는 소프트웨어로는 GeoServer와 MapServer가 있으며, 공간정보 처리에 특화된 소프트웨어로는 Zoo Project, 52 North, PyWPS 등이 있다. 이러한 소프트웨어들을 활용하여 웹에서 데이터를 시각화하거나 처리함으로써 향후 확장성이나 활용성이 확보될 수 있다.

웹 시스템을 설계 및 구축할 때 고려해야 하는 시스템 레이어는 데이터, 미들웨어 소프트웨어, 미들웨어 비즈니스 어플리케이션, 사용자 인터페이스로 구분될 수 있다. 물론 시스템 복잡도, 고유 특성에 따라 각 각의 레이어는 제외될 수 있다. 이번 연구에서 설계 및 구축된 공간정보 처리 시스템은 특정 기능에 대한 시스템이기는 보단 향후 시스템을 기반으로 특정 기능을 채택할 수 있도록 범용성에 대해 고려하였다. 그렇기 때문에 모든 레이어들이 존재한다는 가정 하에 공간정보 시스템을 설계하였다. 그림 3-10과 그림 3-11은

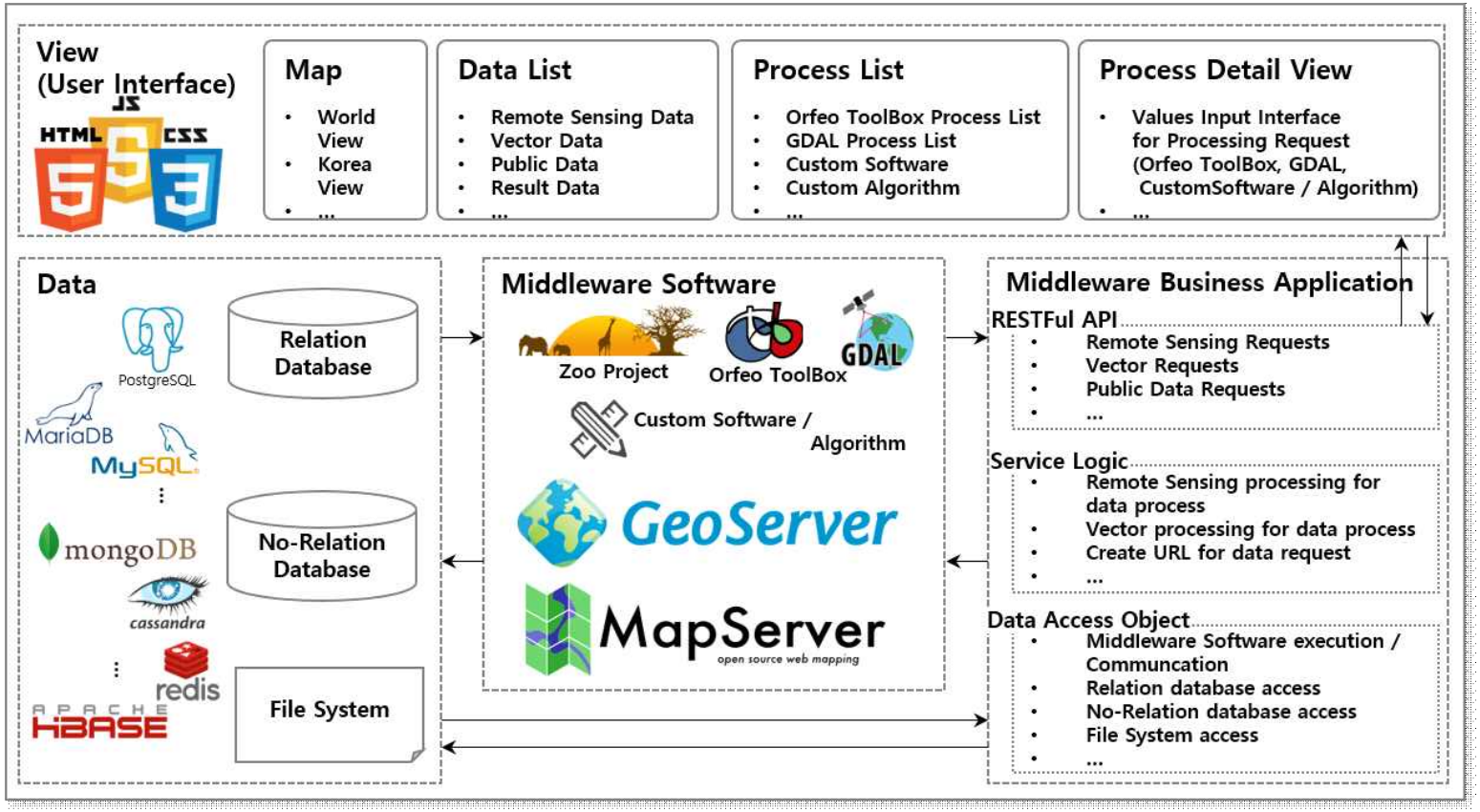
일반 환경과 클라우드 환경에서 서비스하기 위한 각 레이어들에서 사용되는 공간정보 관련된 소프트웨어들과 이 레이어들이 통신하는 방향에 대해 간단하게 구성도를 도식화한 것이다.

데이터 레이어는 데이터베이스와 파일 시스템이 포함되어 있으며 미들웨어 소프트웨어에는 공간정보를 시각화하거나 처리하기 위한 소프트웨어들이 포함되어 있다. 데이터와 미들웨어 소프트웨어를 좀 더 효율적으로 관리하기 위해 비즈니스 어플리케이션이 중간에 위치하고 있으며 RESTFul API 방식으로 설계함으로써 여러 클라이언트에서 접근하여 활용할 수 있다. 클라이언트는 이젠 일반화된 HTML5를 활용하여 사용자가 웹에서도 다양한 경험을 할 수 있도록 제공할 수 있다. 그림에서 확인할 수 있듯이 일반 환경과 클라우드 환경을 큰 차이점으로는 각 레이어 내부에 있는 소프트웨어 및 어플리케이션의 단위이다. 일반 환경에서는 모든 레이어가 같은 물리적 하드웨어에서 구동되도록 설계하는 경우가 대부분이다. 분리된다면 데이터 레이어는 별도 하드웨어를 구성하여 데이터 서버로 구축되고는 한다. 이렇게 구축할 경우 하나의 시스템이 죽거나 과부하가 걸릴 경우 나머지 시스템에도 모두 영향을 줄 수 있다. 하지만 그림 3-11과 같이 클라우드 환경으로 변화되면서 모두 가상 하드웨어 형태인 인스턴스로 구축할 수 있으며 이는 마이크로 서비스(micro service)의 기초라고 볼 수 있다. 물론 마이크로 서비스 단위로 제공하기 위해서는 각 인스턴스마다 독립적으로 설계되어야 하며 관련 있는 인스턴스들은 서로 상호호환이 이루어져야 하므로 복잡도가 높아질 수 있다. 하지만 이렇게 구성함으로써 각 인스턴스간 확장성을 확보할 수 있을 뿐만 아니라, 과부화 또는 문제가 생겼을 때 빠르게 대처할 수 있는 구조가 될 수 있다.

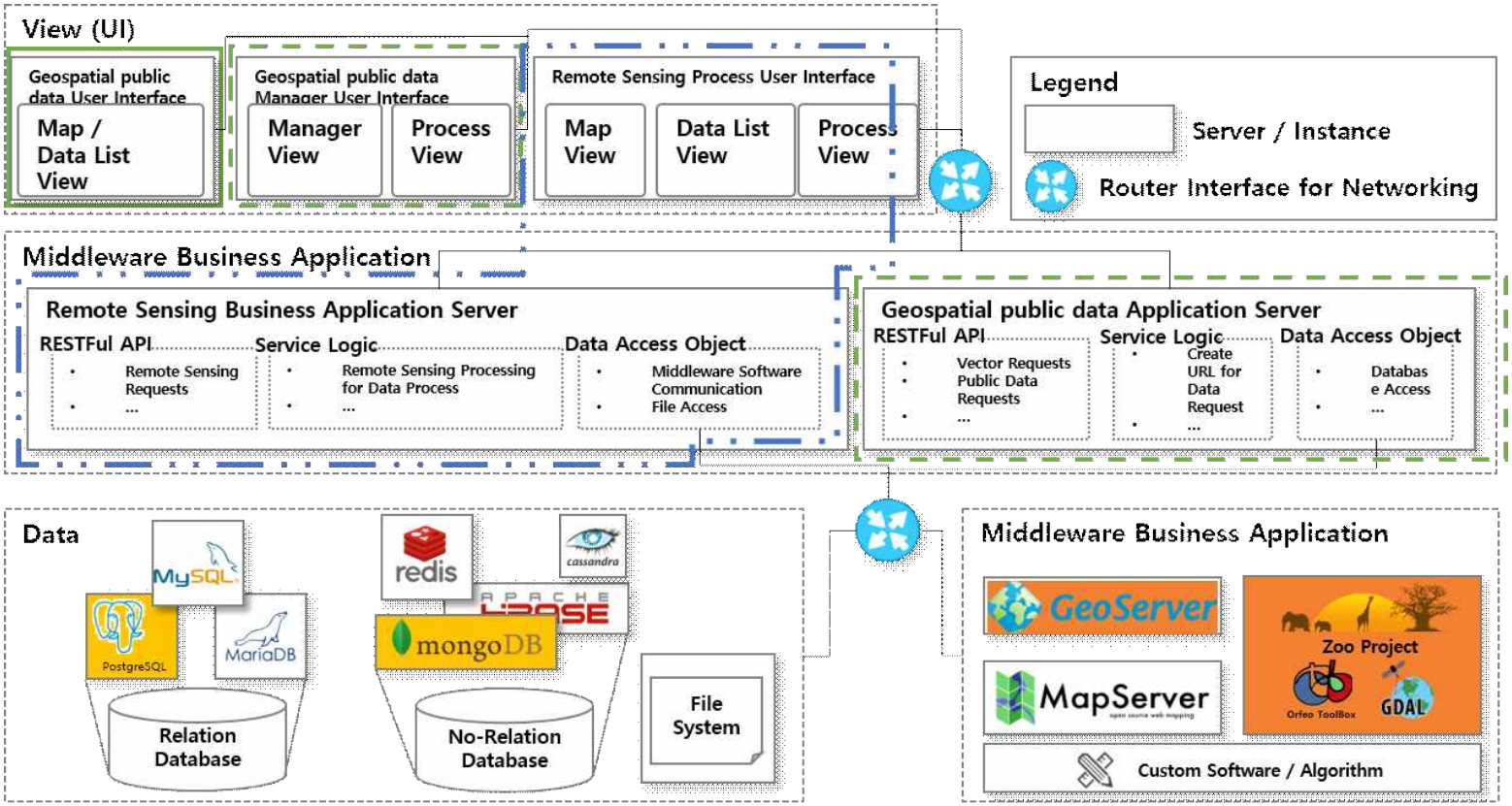
이번 연구에서 설계 및 구축된 공간정보 처리 서비스는 PaaS 기반으로 구성하면서 기본적으로 그림 3-11과 동일한 방식으로 설계되었다. 사용된 데이터 레이어는 PaaS 서비스를 구축을 통해 모두 인스턴스화 또는 컨테이너화하여 이미 마이크로 서비스 단위로 구축이 된 상태이다. 실제 웹 서비스에 중심적인 역할을 하는 비즈니스 어플리케이션과 사용자 인터페이스 또한 마이크로 단위로 설계하였다. 공간정보와 공공 데이터를 관리할 수 있는 비즈니스 어플리케이션과 수집된 데이터를 융합하여 시각화해주는 웹 어플리케이션, 공

간정보 처리 미들웨어 소프트웨어와 통신하여 위성영상을 처리하도록 인터페이스 역할을 해주는 비즈니스 어플리케이션 및 인터페이스 총 세 개 단위로 나누었다.

이번 절에서는 PaaS에서 제공되는 서비스 중 공간정보에 특화된 미들웨어 소프트웨어인 공간정보 웹 표준 시각화 서버와 웹 표준 처리 서버에 대한 서비스를 위해 인스턴스를 설계 및 구축과정을 정리하였다. 또한, 마이크로 단위로 설계 및 구축된 비즈니스 어플리케이션과 사용자 인터페이스 클라이언트 앱을 실제 PaaS 위에 배포하기 위한 단계와 구축 결과를 정리하였다.



[그림 3-10] 단일 서버에서의 공간정보 웹 서비스 일반 구성도.



[그림 3-11] 클라우드 환경에서의 공간정보 웹 서비스 일반 구성도 및 마이크로 단위 구분.

1) 공간정보 웹 표준 서비스 구축

공간정보 분야에서 사용되는 데이터는 웹에서 바로 사용할 수 없는 데이터들이 존재한다. 이러한 데이터들을 웹에서 상호운영을 고려하여 시각화 및 처리하기 위해 OGC에서는 공간정보 웹 표준을 정의하여 배포하고 있고 전 세계적으로 많이 활용되고 있다(Zhao et al., 2012; Lopez-Pellicer et al., 2012; Klug and Kmoch, 2014; Hare et al., 2017). 이를 실제로 구현한 구현체로 여러 상용 및 오픈소스 소프트웨어들이 출시되어 있다. 이번 연구에서는 제한 없이 활용할 수 있는 오픈소스 소프트웨어를 대상으로 시스템을 구축하였다. 공간정보를 시각화할 수 있는 오픈소스 소프트웨어로는 대표적으로 GeoServer와 MapServer가 존재한다. 소프트웨어 모두 공간정보에 대한 시각화 웹 표준인 WMS, WTMS, WFS 등을 모두 지원을 하고 있으며 라이선스 또한 자유롭게 활용할 수 있는 라이선스를 채택하고 있다. 이번 연구에서는 GeoServer를 활용하여 공간정보 처리 서비스를 구현하는데 활용하고자 한다.

공간정보를 처리하기 위한 표준 인터페이스로 WPS가 존재한다. WPS의 실제 구현 소프트웨어로는 GeoServer, PyWPS, 52 North 등이 출시되어 있다. 웹 표준 시각화 소프트웨어로 GeoServer를 선택하였기 때문에 처리 또한 GeoServer를 사용하면 편리할 수 있다. 하지만 이번 연구에서 사용된 웹 표준 처리 소프트웨어는 Zoo Project를 활용하였다. WPS는 오랫동안 1.0이 활용됐고 이를 활용한 사례 연구들이 대부분이다(Rautenbach et al., 2012; Giuliani et al. 2012; Castronova et al., 2013). OGC에서는 2015년 2.0을 발표하였고(Mueller and Pross, 2015) 이에 대해 관련 소프트웨어들도 따라가고 있는 추세이다. 2.0 버전으로 업그레이드되면서 비동기식 처리 방식을 지원하며 이와 관련된 인터페이스 사양이 추가되었다. GeoServer의 경우 아직 2.0 개발이 완벽하게 이루어지지 않았지만 Zoo Project의 경우 WPS 2.0에 대한 소프트웨어 지원이 완벽하게 구성되어 있다(Zoo Project team, 2017). WPS 2.0에 대한 적용 관련된 연구 및 사례로는 현재 많지 않지만 이에 대한 적용 및 성능 측정에 대한 수행 연구가 진행된바 있으며 비적용, 1.0 적용, 2.0 적용 세 가지로 구분하였을 때 2.0 적용한 경우가 좀 더 나은 결과

를 보여주었다(윤구선 , 2017). 물론 해당 연구만을 보고 반드시 WPS 2.0을 적용하였을 때 좋은 결과를 나타낸다고 볼 수 없다. 하지만 비동기식 지원과 요청 방식에 대한 변경으로 인해 좀 더 안전한 처리 요청을 할 수 있으며 확장성에 대한 장점이 크게 잡고 있으므로 사용자에게 나은 서비스를 제공할 수 있다. 그리고 표준을 수용하고 있는 소프트웨어의 경우 향후 표준에 맞춰 기능이 지원될 것으로 예상되므로 얼마든지 변경하여 설계 및 구축가능하다. Zoo Project는 처리에 대한 인터페이스를 제공해주는 것이다. 실제 처리를 위해서는 별도 처리 알고리즘 또는 소프트웨어가 필요하다. 공간정보를 입출력 및 처리하기 위한 소프트웨어를 추가로 설치하여 Zoo Project WPS 2.0 처리 인터페이스와 연결하여 실제 수행 결과를 확인하였다.

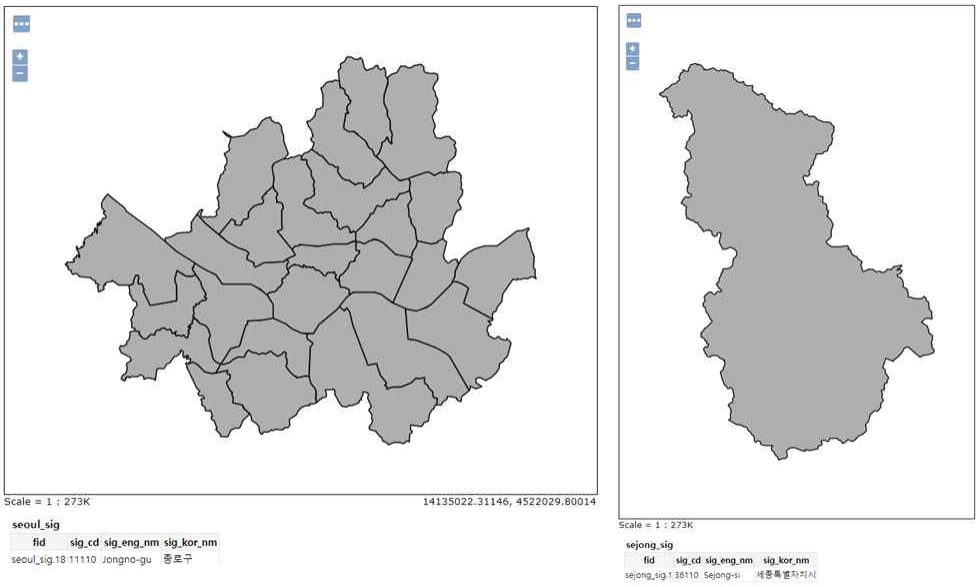
가) 공간정보 웹 표준 시각화 서버 및 데이터 환경 구축

구축된 공간정보 웹 표준 시각화 서버는 GeoServer를 활용하였다. GeoServer는 자바 환경의 웹 어플리케이션이다. 2017년 10월 기준으로 2.11.2 버전이 출시되어 있으며 이번 연구에서는 2.7.6 버전을 활용하였다. 2.7.6버전을 설치와 동시에 기본 활용 가능한 웹 표준 서비스로는 WMS-C, WMTS, WCS, WFS, WMS이다. 추가적으로 웹 표준은 아니지만 TMS(Tiled Map Service)도 활용 가능하다. WAS(Web Application Server) 형태로 압축되어 있기 때문에 구축하기에도 크게 불편한 점이 없다. 공간정보 처리 서비스를 위해 활용된 웹 표준 서비스로는 WMTS와 WFS이다. WMTS는 위성정보를 시각화하기 위해 활용하였고, WFS는 공간정보를 시각화하기 위해 활용하였다. 웹 표준 시각화 서버가 구축되면 사용할 데이터를 업로드 해야한다. GeoServer는 파일 단위 데이터뿐만 아니라 데이터베이스 단위 데이터도 지원한다. 데이터베이스는 기본적으로 PostgreSQL을 지원하며 플러그인을 통해 다른 데이터베이스도 추가할 수 있다.

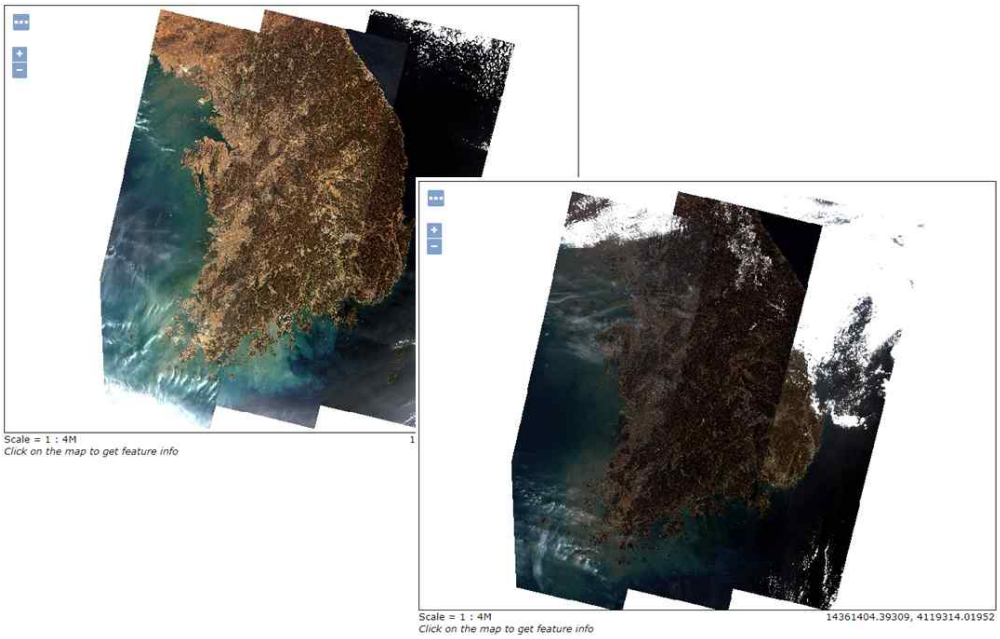
이번 연구에서 테스트를 위해 공개된 데이터를 다운 받아 활용하였다. 공간정보의 경우 국내 국가 공간 정보 포털(http://market.nsdi.go.kr)에서 제공하는 전국 데이터를 사용하였다. 국가 공간 정보 포털에서는 공간정보와 관련

된 데이터를 텍스트 또는 실제 벡터 데이터로 제공해주는 오픈 마켓이다. 제공하는 범위로는 도시 및 지역개발, 도로 교통물류, 환경 자연 및 기후 재난방재 공공안전 등 매우 다양하다. 이러한 데이터 중 도로명주소 전자지도로 건물, 건물군, 도로구간, 길폭도로, 기초구간, 출입구, 기초구역, 행정구역경계(시도, 시군구, 읍면동, 법지리) 등 11종에 대한 데이터를 벡터 데이터로 다운로드 받아 사용할 수 있는 데이터가 존재한다. 이번 연구에서는 Shape 파일로 구성되어 있는 도로명주소 전자지도를 웹 표준으로 시각화할 수 있도록 PostgreSQL / PostGIS에 저장하여 GeoServer와 연결하였다. 그림 3-12는 이번 연구에서 활용된 데이터 중 일부로 서울시 구 단위 경계면과 세종시 경계면 벡터 데이터를 WFS로 시각화한 결과이다. 사용된 레스터 데이터 또한 공개된 데이터를 사용하였다. 그림 3-13은 레스터 데이터인 위성정보를 웹 표준 서비스 WMTS를 활용해서 시각화한 결과이다. 사용한 레스터 데이터는 Landsat 데이터로 USGS와 NASA가 지구 지원 데이터를 얻기 위해 발사한 인공위성으로 촬영된 데이터이다. 이는 극궤도 위성으로 전 지구 영역 관측을 하고 있다. Landsat 1부터 8까지 발사되었고 촬영된 데이터는 누구든지 USGS에서 제공하는 홈페이지(http://earthexplorer.usgs.gov)를 통해 원본 데이터를 쉽게 검색 및 다운로드 받을 수 있고 사용할 수 있다. 최근에는 USGS 외 클라우드 컴퓨팅 IaaS를 제공하고 있는 아마존 웹 서비스(AWS: Amazon Web Service)에서도 S3(Simple Storage Service)를 통해 Landsat 데이터를 다운로드할 수 있도록 제공하고 있다(AWS, 2017). 이번 연구에서 활용된 테스트 데이터의 경우 가장 최근에 발사된 Landsat 8 영상으로 제주도를 제외한 국내 전국 지역을 커버할 수 있도록 최대한 근접한 날짜에 촬영된 한반도 9개의 데이터를 연결하여 하나의 데이터로 구성하였다. 단일 데이터를 활용하여 연구에 활용할 수 있지만, 국내에서는 정지궤도 위성이 존재하고 데이터를 보유하고 있으므로 얼마든지 한반도 데이터를 서비스할 수 있으므로 한반도 전체를 커버하는 데이터 중 공개되어 있는 데이터로 구성하여 테스트하였다. Landsat 데이터는 총 12개의 밴드로 구성되어 있다. 그중 사람 눈으로 식별 가능한 가시광선 데이터 범위가 저장된 밴드는 2(blue), 3(green), 4(red)번이다. 시각화를 위해서는 이 가시광선 범위의 밴드를 활용하여 데이

터를 가공해야 한다. 물론 향후 처리 시 데이터 정확도를 위해 가공된 데이터는 시각화에만 사용되고 원본데이터를 동시 보유하여 처리에 활용될 수 있다. 사용된 데이터는 날짜를 월 단위로 구분하였을 때 2015년 2월에서 3월과 2017년 2월에서 3월을 기준으로 촬영된 영상들을 활용하였다. 극궤도 위성이기 때문에 동일 날짜에 모든 지역을 촬영할 수 없으므로 월 단위로 검색한 데이터 중 가장 구름 범위가 낮은 데이터들을 활용하였다. 활용된 데이터에 대한 상세 날짜에 대해서는 표 3-6에 정리하였다. 데이터 획득 날짜 범위는 대부분 02월에서 03월 사이인 것을 볼 수 있다. Scene Cloud Cover는 영상에 대한 구름 영역을 0~100%로 수치화한 것이다. 2017년 3월 21일 데이터에 대한 영역이 86%로 가장 많지만 이 영상은 영상이 통합될 때 아주 일부만 사용되기 때문에 크게 문제가 되지 않는다. Image Quality는 9, 0, -1 이 세 가지 수치로 측정된다. 9가 가장 좋은 품질의 영상, 0이 나쁜 품질의 영상, -1은 품질을 측장할 수 없는 영상으로 구분된다.



[그림 3-12] GeoServer 공간정보 WFS 시각화 결과: (a) 서울시 경계면, (b) 세종시 경계면.



[그림 3-13] GeoServer WMTS 시각화 결과: (a) Landsat 8 2015년 2-3월 RGB 밴드, (b) Landsat 8 2017년 2-3월 RGB밴드.

[표 3-6] 실험에 사용된 Landsat 8 데이터 고유번호 및 촬영 날짜

Landsat Scene Identifier	Date Acquired	Scene Cloud Cover	Image Quality
LC81140342015043LGN00	2015-02-12	0.41	9
LC81140352015043LGN00	2015-02-12	6.72	9
LC81140362015043LGN00	2015-02-12	0.28	9
LC81150342015082LGN00	2015-03-23	0.07	9
LC81150352015082LGN00	2015-03-23	0.01	9
LC81150362015082LGN00	2015-03-23	11.55	9
LC81160342015073LGN00	2015-03-14	4.69	9
LC81160352015073LGN00	2015-03-14	0.9	9
LC81160362015073LGN00	2015-03-14	30.62	9
LC81140342017080LGN00	2017-03-21	86.76	9
LC81140352017080LGN00	2017-03-21	21.74	9
LC81140362017080LGN00	2017-03-29	1.96	9
LC81150342017055LGN00	2017-02-24	4.08	9
LC81150352017055LGN00	2017-02-24	0.54	9
LC81150362017055LGN00	2017-02-24	1.22	9
LC81160342017062LGN00	2017-03-03	24.12	9
LC81160352017062LGN00	2017-03-03	1.76	9
LC81160362017062LGN00	2017-03-03	2.51	9

Image Quality(9=Best, 0 = Worst, -1 Image Quality not calculated or assessed)

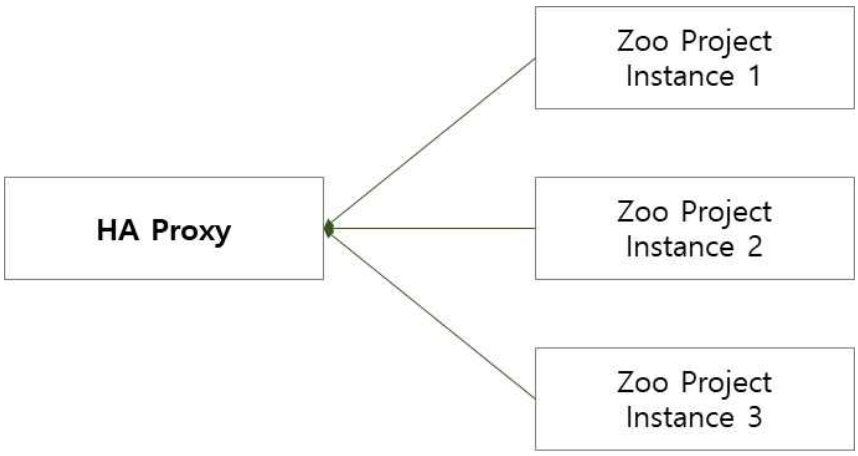
나) 공간정보 웹 표준 처리 서버 환경 구축

Zoo Project는 공간정보 웹 표준인 WPS 사양을 1.0과 2.0 모두 완벽하게 지원하고 있다. 특히 2015년 6월에 공식적으로 채택된 WPS 2.0의 경우 1.0 버전에는 존재하지 않았던 비동기식 방식이 추가되면서 활용도가 높아졌다. 실제 처리가 수행되는 공간이기 때문에 확실한 테스트를 위해 향후 확장 가능한 방식으로 인스턴스를 구성하였다. 단순히 처리 인스턴스 하나를 구성하는 것이 아닌 그림 3-14에서 보이는 것과 같이 소프트웨어 적으로 L4, L7 기능 및 로드 밸런서 기능을 제공하는 HAProxy 서버를 구축하여 총 세 대의 가상 인스턴스를 연결하였다. 이렇게 구축된 부분은 IaaS의 장점을 극대화한 것으로 한 대의 인스턴스를 모두 구축 후 이를 이미지화하여 추가한 것이고 향후 이 또한 IaaS의 오토 스케일링 서비스를 통해 확장이 가능하다. Zoo Project는 크게 Zoo Kernal, Zoo Service, Zoo Client로 나눠진다. Zoo Kernal에서 실제로 WPS 2.0 사양에 맞는 Capabilities, Execute, Status Result와 같이 처리에 필요한 인터페이스를 지원하고 있다. 이렇게 처리에 필요한 인터페이스만 존재하므로 실제 처리할 수 있는 기능을 포함되어 있지 않다. 그렇기 때문에 동일 인스턴스 내부에 실제 처리 가능한 소프트웨어 또는 알고리즘이 포함되어야 한다. 실제 테스트를 위해 이번 연구에서는 OTB와 GDAL을 설치하여 활용하였으며 사용자가 웹에서 간단한 인터페이스를 통해 공간정보를 처리할 수 있는 기능을 구현하였다. 그림 3-15와 그림 3-16은 실제 Zoo Project를 통한 WPS 2.0 수행 결과이다. 그림 3-15는 설치된 Zoo Project에서 제공하고 있는 처리 목록을 보여준다. WPS 2.0 요청은 기본적으로 XML 바인딩으로 요청되고 응답한다. 이번 연구에서 테스트될 처리 기능으로는 OTB 처리 기능 중 윤곽선 추출(Edge Extraction), 영상 흐림화(Mean Shift Segmentation), 영상 분류(K means) 기능과 GDAL의 영상 자르기(Crop) 기능을 WPS 처리 인터페이스에 맞게 사용 가능하도록 구축하였다. Zoo Project에서 이와 같이 처리 기능을 추가하기 위해서는 Zoo Service를 통해 Zoo Server와 직접 연결 또는 간접적으로 연결되어야 한다. 간접적으로 연결이 될 경우 Zoo Service에서 제공하고 있는 언어를 통해 직

접 구현해야 한다. 이번 연구에서는 모든 기능을 간접적으로 사용할 수 있도록 구축하였고 개발 언어 관련하여 통일성을 주기 위해 자바를 활용하여 해당 기능들을 Zoo Service에 연결하였다.

그림 3-16은 네 개의 처리 기능 중 윤곽선 추출에 대한 입력 값 및 출력 값에 대한 정보를 요청하는 WPS 인터페이스 사양 중 Describe Process 결과이다. 요청 URL은 동일하지만 바인딩 되는 XML이 변경됨으로써 응답 값 또한 달라지는 것을 확인할 수 있다. Describe Process에는 처리를 위해 필요한 입력 값들이 wps:input을 통해 설명되어 있고 처리된 후 반환 값에 대해서는 wps:output을 통해 설명되어 있다. 실제 입력 값과 일치하지 않을 경우 수행이 되지 않기 때문에 필요한 정보 중 하나이며, 클라이언트에서 사용자 인터페이스를 구성하는 데 중요한 역할을 한다.

공간정보를 웹에서 시각화 및 처리하기 위한 웹 표준 관련 소프트웨어를 PaaS에서 활용하기 위해 구축하였다. 구축 결과에서 보면 확인할 수 있듯이 시각화하거나 처리할 때 사용되는 WMTS, WFS, WPS는 모두 고유 URL만을 가지고 사용하는 것을 확인할 수 있다. 먼저 구축된 MySQL, PostgreSQL, MongoDB의 경우 사용자 요청에 따라 별도 데이터베이스가 생성되어야 하고 관리되어야 한다. 그렇기 때문에 서비스 브로커를 설계 및 구축하였고 추가하였다. 하지만 GeoServer 나 Zoo Project의 경우 단순 URL 제공이기 때문에 서비스 브로커가 반드시 필요하지 않다. 물론 향후 GeoServer에 데이터를 직접 올리거나 처리를 위한 알고리즘을 직접 활용한다면 서비스 브로커를 이에 맞춰 설계 및 구축하여 추가하여 제공할 수 있다.



[그림 3-14] HAProxy 적용된 Zoo Porject 인스턴스 구성도.

Request URL: cgi-bin/zoo_loader.cgi (POST)

Request Data:

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:GetCapabilities
      xmlns:wps="http://schemas.opengis.net/wps/2.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://schemas.opengis.net/wps/2.0
      ./xsd/wps.xsd" service="WPS">
</wps:GetCapabilities>
```

Response Data:

```
▼<wps:ProcessOfferings>
  ▼<wps:Process wps:processVersion="1.0">
     <ows:Identifier>OTBPROC.EdgeExtraction</ows:Identifier>
     <ows:Title>EdgeExtraction</ows:Title>
     <ows:Abstract>Raster data edge Extraction</ows:Abstract>
     <ows:Metadata xlink:title="Metadata"/>
   </wps:Process>
  ▼<wps:Process wps:processVersion="1.0">
     <ows:Identifier>OTBPROC.MeanShiftSegmentation</ows:Identifier>
     <ows:Title>MeanShiftSegmentation</ows:Title>
     <ows:Abstract>Raster data MeanShift Segmentation (Smooth)</ows:Abstract>
     <ows:Metadata xlink:title="Metadata"/>
   </wps:Process>
  ▼<wps:Process wps:processVersion="1.0">
     <ows:Identifier>OTBPROC.KMeans</ows:Identifier>
     <ows:Title>KMeans</ows:Title>
     <ows:Abstract>Raster dataClassification based on KMeans</ows:Abstract>
     <ows:Metadata xlink:title="Metadata"/>
   </wps:Process>
  ▼<wps:Process wps:processVersion="1.0">
     <ows:Identifier>GDALPROC.CropbasedShape</ows:Identifier>
     <ows:Title>CropbasedShape</ows:Title>
     <ows:Abstract>Raster data crop based on shape file</ows:Abstract>
     <ows:Metadata xlink:title="Metadata"/>
   </wps:Process>
 </wps:ProcessOfferings>
▶<wps:Languages>...</wps:Languages>
</wps:Capabilities>
```

[그림 3-15] WPS 2.0 GetCapabilities 요청 및 응답 일부 결과.

Request URL: cgi-bin/zoo_loader.cgi (POST)

Request Data:

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:DescribeProcess
      xmlns:ows="http://www.opengis.net/ows/2.0"
      xsi:schemaLocation="http://www.opengis.net/wps/2.0 ./xsd/wps.xsd"
      service="WPS" version="2.0.0">
   <ows:Identifier>OTBPROC.EdgeExtraction</ows:Identifier>
</wps:DescribeProcess>
```

Response Data:

```
▼<wps:ProcessOfferings xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:wps="http://w
 xsi:schemaLocation="http://www.opengis.net/wps/2.0 http://schemas.opengis.net/wps/2.
  ▼<wps:ProcessOffering processVersion="1.0.0.0" jobControlOptions="sync-execute asyr
    ▼<wps:Process>
       <ows:Title>EdgeExtraction</ows:Title>
       <ows:Abstract>Raster data edge Extraction</ows:Abstract>
       <ows:Identifier>OTBPROC.EdgeExtraction</ows:Identifier>
      ▼<wps:Input minOccurs="1" maxOccurs="1">
         <ows:Title>The map of digital map</ows:Title>
         <ows:Abstract>The map of digial map</ows:Abstract>
         <ows:Identifier>SHAPE</ows:Identifier>
        ▶<wps:LiteralData>...</wps:LiteralData>
       </wps:Input>
      ▼<wps:Input minOccurs="1" maxOccurs="1">
         <ows:Title>The map of Raster map</ows:Title>
         <ows:Abstract>The map of Raster map</ows:Abstract>
         <ows:Identifier>RASTER</ows:Identifier>
        ▶<wps:LiteralData>...</wps:LiteralData>
       </wps:Input>
      ▼<wps:Input minOccurs="1" maxOccurs="1">
         <ows:Title>channel</ows:Title>
         <ows:Abstract>channel (help:1)</ows:Abstract>
         <ows:Identifier>CHANNEL</ows:Identifier>
        ▶<wps:LiteralData>...</wps:LiteralData>
       </wps:Input>
      ▼<wps:Input minOccurs="1" maxOccurs="1">
         <ows:Title>RESULT FILE NAME PREFIX</ows:Title>
         <ows:Abstract>RESULT FILE NAME TO DISTINGUISH</ows:Abstract>
         <ows:Identifier>USER</ows:Identifier>
        ▶<wps:LiteralData>...</wps:LiteralData>
       </wps:Input>
```

[그림 3-16] WPS 2.0 Describe Process 요청 및 응답 일부 결과.

2) 공간정보 처리 웹 시스템 설계 및 배포

오픈스택 구축부터 클라우드 파운더리 구축, 클라우드 파운더리 서비스 구축은 모두 클라우드 PaaS를 제공하기 위한 작업이다. 이는 향후 시스템 관리자가 수행하는 몫으로 웹 시스템 개발자는 이렇게 구축된 PaaS를 사용하기만 하면 된다. 이번 연구에서는 두 종류의 공간정보 처리 웹 시스템을 설계 및 구현하여 PaaS에 배포하였다. PaaS에 앱을 배포하기 위해서는 몇 가지 고려해야 할 사항이 존재한다. 첫 번째로 PaaS에서 빌드 가능한 언어를 사용해야 한다. 두 번째 앱 내부에는 영구 저장소가 없으므로 저장 데이터를 피해야 한다. 만약 저장 데이터가 존재할 경우 IaaS 또는 PaaS에서 제공하는 네트워크

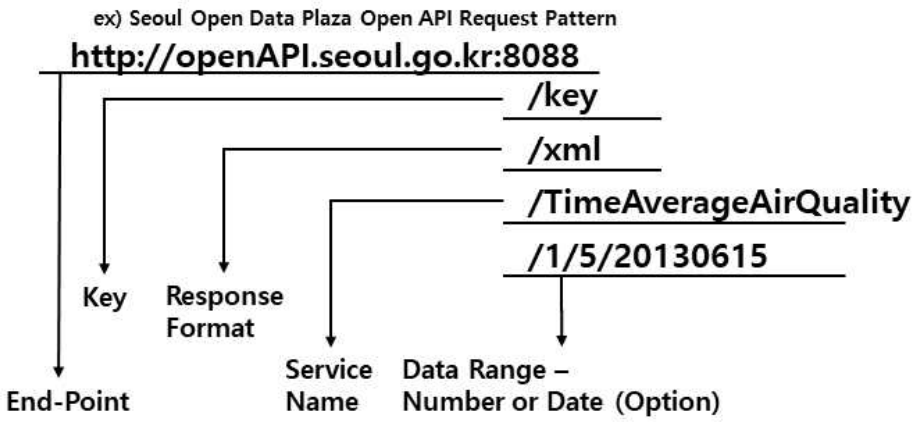
파일 시스템 또는 데이터베이스화하여 앱을 설계해야 한다. 세 번째 데이터베이스 또는 미들웨어 소프트웨어를 사용할 경우 앱 배포 후 바인딩 수행되므로 공간(Space)에서 사용되는 서비스 정보를 미리 파악해놓고 있어야한다. 이번 연구에서 설계 및 구축된 웹 시스템을 PaaS에 맞게 설계하는 과정과 이를 배포한 결과에 대해 정리하였다.

가) 공간정보 및 공공데이터 융합 시각화 웹 시스템 설계

공간정보 및 공공데이터 융합 시각화 웹 시스템 주요 기능은 Open API로 제공되고 있는 공공 데이터를 일정 주기에 맞춰 자동으로 수집하고, 수집된 데이터를 시각화하는 데 있어 단순 차트로 시각화하는 기능과 공간정보와 융합하여 시각화하는 기능을 제공한다. 웹 시스템에 사용된 데이터는 서울 열린 데이터 광장(http://data.seoul.go.kr)에서 제공하는 Open API 데이터를 사용하였다. 공공 데이터 수집을 위해서는 제공하는 기관의 Open API 엔드 포인트(End-Point)에 대한 조사가 필요하다. 서울 열린 데이터 광장의 경우 데이터 제공 시 그림 3-17과 같은 패턴으로 구성된다. 기본적으로 모든 기관이 그렇듯이 Open API 제공을 위해 엔드 포인트(End-Point)를 제공한다. 그 뒤에 파라미터 형식으로 사용자가 원하는 입력 값을 삽입함으로써 데이터를 응답받을 수 있다. 기본적으로 키 값, 응답 형식, 데이터 이름에 대한 값은 반드시 포함되도록 되어 있다. 서울시 열린 데이터 광장의 경우 이 값을 모두 URI 방식으로 제공하기 때문에 주소처럼 작성하여 요청하도록 되어 있다. 이외에도 XML(Extensible Markup Language) 또는 JSON(JavaScript Object Notation) 데이터를 이용하여 요청하는 경우도 존재한다. 마지막 옵션으로 데이터 범위를 상세하게 정할 수도 있다. 이렇게 조사된 내용을 기반으로 패턴을 분석하여 사용자가 직접 등록하고 관리할 수 있도록 시스템을 구성하였다.

전체적인 시스템 환경은 표 3-7과 같은 환경에서 구현되었다. 웹 시스템은 Geo Data & Open Data Management Server(GOMS)와 웹 클라이언트(Web Client)로 나눌 수 있다. 융합 시각화 웹 시스템 GOMS에는 공공데이터를 관리할 수 있는 대시보드 인터페이스가 포함되어 있어 융합 시각화

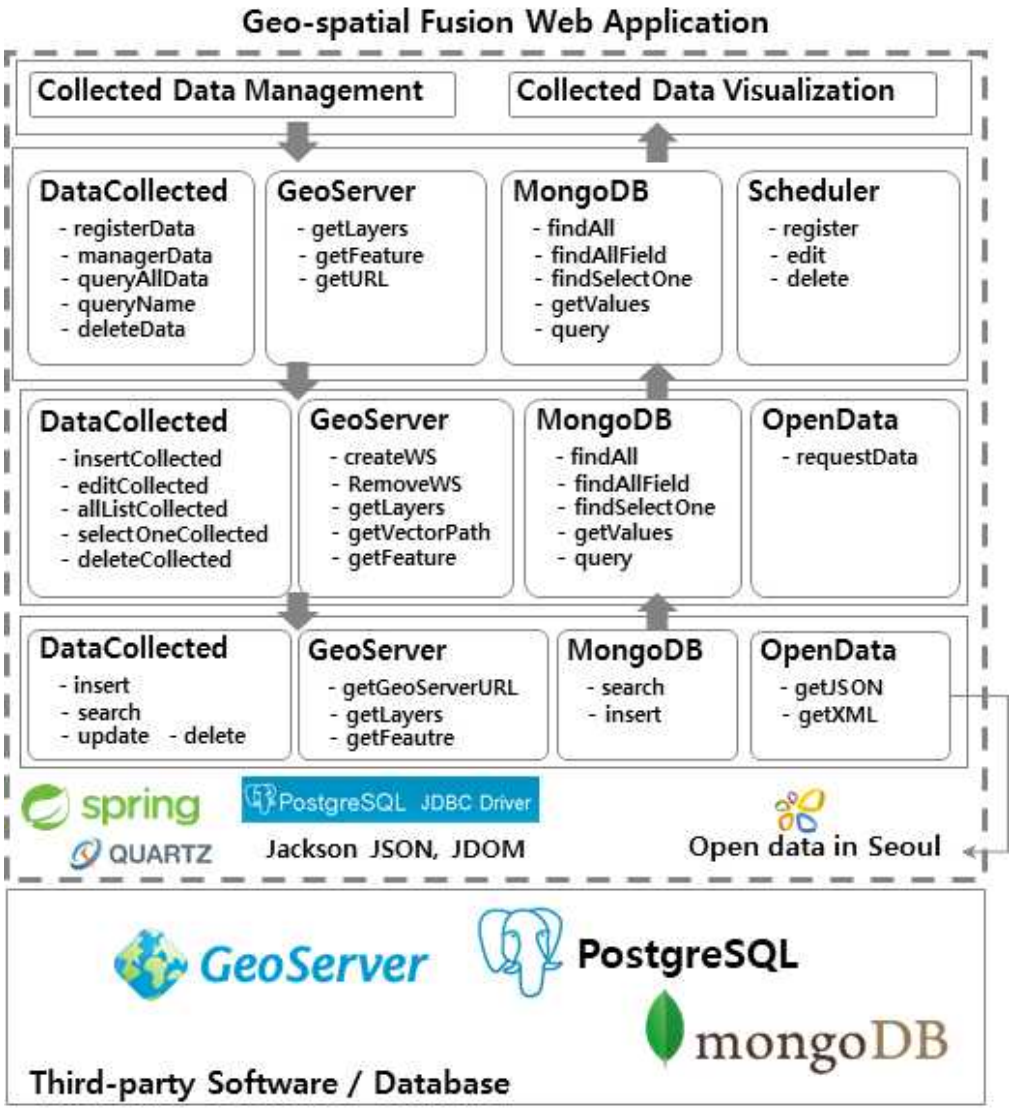
PaaS 배포 시 각각 독립적으로 구동되도록 설계되어 있다. GOMS는 자바 기반의 스프링 부트(Spring Boot) 프레임워크로 구축된 웹 어플리케이션이다. 공공 데이터를 일정 시간에 맞춰 자동으로 수집하기 위해 스케줄로 라이브러리인 쿼츠(Quartz)를 사용하였고 관리 인터페이스는 스프링 부트에서 지원하는 뷰(View) 엔진 중 Thymeleaf를 활용하였다. 개발자가 직접 RESTFul API를 이용하여 수집 데이터를 관리할 수 있지만, 일반 사용자의 경우 사용자 인터페이스가 제공되어야 한다. Thymeleaf를 기반으로 생성한 대시보드 인터페이스에서는 현재 수집 중인 공공 데이터에 대한 정보를 통합적으로 보여주고 수집 데이터에 대한 추가 및 관리를 편리하게 할 수 있다. 또한 수집된 원본 데이터를 시각화할 수 있다. PaaS에 배포할 것이기 때문에 GOMS에서 필요한 미들웨어 소프트웨어에 대해서는 환경 및 버전에 대해 정리되지 않았다. GOMS는 향후 PaaS에 배포될 때 자바 빌드팩(Buildpack)으로 배포되도록 기본 설정되어 있다. 웹 클라이언트는 순수 HTML, JavaScript, CSS만을 활용해서 만들어진 클라이언트 웹 어플리케이션이다. 사용자 인터페이스를 구성하기 위해 Bootstrap을 사용하였고 공간정보 시각화 및 차트 시각화를 위해 OpenLayers 와 D3 라이브러리를 사용하였다. 세부적인 시스템 설계는 그림 3-18로 도식화하였다. 대부분 앱에서 많이 활용되고 있는 패턴인 MVC(Model-View-Controller) 패턴으로 기본 설계되어 있으며 Data Collected, GeoServer, MongoDB, Scheduler 기능으로 구분할 수 있다. 맨 위에서부터 View, Controller, Model(Service, DAO)순으로 정리된다. 데이터 수집 관리에서는 등록, 관리, 검색, 삭제 기능이 구현되어 있으며, GeoServer는 저장된 레이어 목록, 공간정보 객체에 대한 값을 받을 수 있는 기능이 구현되어 있다. MongoDB는 실제 수집된 데이터 전체를 검색하거나, 특정 조건에 맞는 데이터를 검색하는 기능이 구현되어있다. 마지막으로 Scheduler에서는 Quartz를 통해 정해놓은 시간에 맞춰 공공 데이터를 실제로 요청하고 저장하는 기능을 호출하도록 구현되어 있다. 관리 서버의 경우 자바 기반의 빌드팩을 활용한 배포가 이루어지고 클라이언트의 경우 정적 파일(HTML, JavaScript, CSS)만을 활용하였기 때문에 Static 빌드팩을 활용한 배포가 이루어진다.



[그림 3-17] 서울 열린 데이터 광장 Open API 패턴 분석 결과.

[표 3-7] 공간정보 및 공공데이터 융합 시각화 웹 시스템 환경

System environment			Version
Geo Data & Open Data Management Server	Web framework	Spring boot	1.3.3
	Scheduler Library	Quartz	2.2.1
	Template Engine	Thymeleaf	2.1.4
	Need GeoServer		
	Need None-Relation Database(MongoDB)		
	Need Relation Database(PostgreSQL)		
Web Client	Markup Language	HTML	5
	Script language	JavaScript	ECMA5
	User Interface	Bootstrap	3.3.6
	Mapping library	OpenLayers	3.10.2
	Chart library	D3	4



[그림 3-18] 공간정보 및 공공데이터 융합 시각화 시스템 설계도.

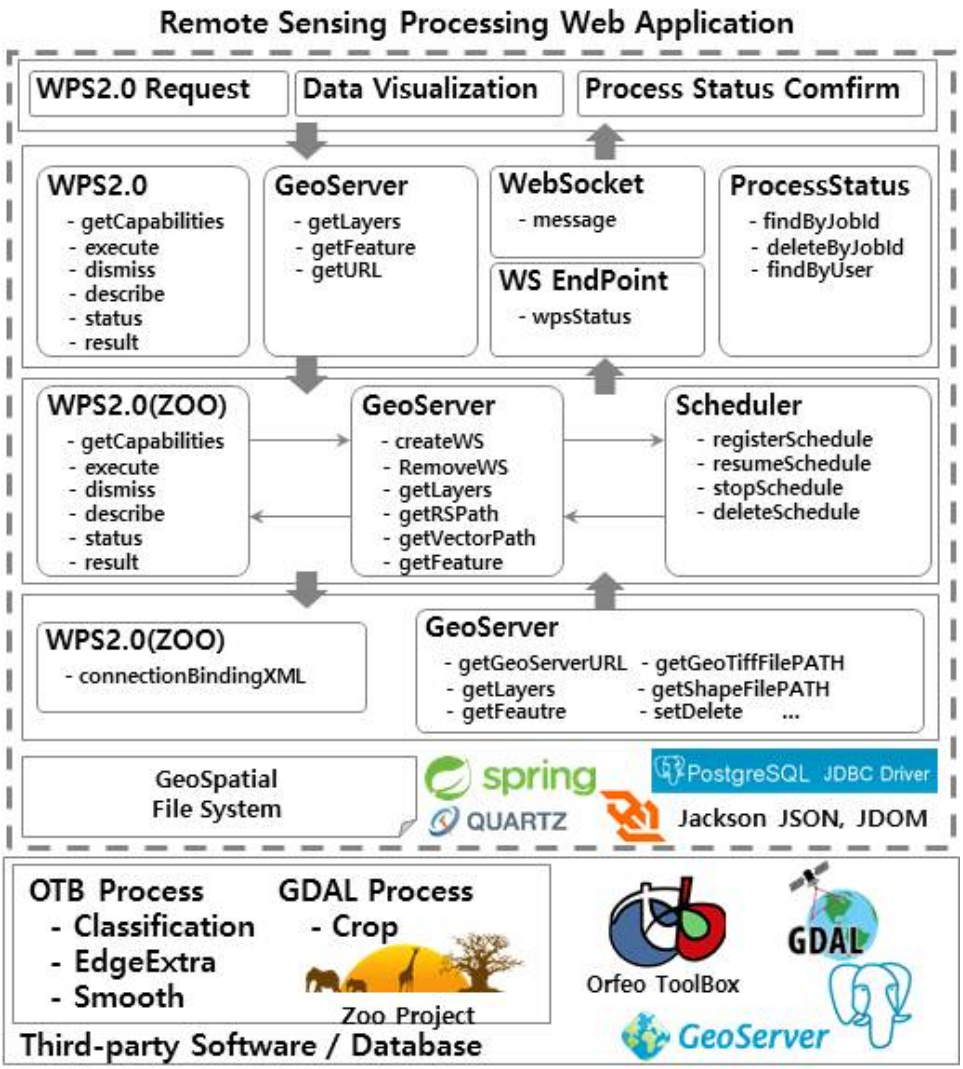
나) 위성정보 처리 웹 시스템 설계

공간정보 처리 웹 시스템도 융합 웹 시스템과 동일한 방식인 자바 기반의 스프링 부트 프레임워크로 구성하였다. 처리 웹 시스템의 경우 수집 관리와 융합 시각화를 나눈 것처럼 서버와 클라이언트를 구분하지 않고 처리 서버와 클라이언트가 하나로 구성되어 구동되도록 설계하였다. 사용된 라이브러리 환경은 표 3-8에서 볼 수 있듯이 GOMS와 거의 비슷하다. 특별한 사항 중 하나로 처리 웹 시스템의 경우 WebSocket을 사용하였다. WebSocket은 HTML5가 발표되면서 활용되기 시작된 기술 중 하나이다. 기존 HTTP 통신을 개선하기 위한 방법으로 기존 Pull 형식에 대한 통신 프로토콜을 소켓 통신 방식인 Push 형식으로 제공할 수 있는 기술이다. WebSocket을 공간정보 분야에 활용한 사례 연구로 Kim and Lee (2016)는 공간 속성정보를 실시간으로 관리하기 위해 기술에 대한 사양을 정리하고 실제 구축한 바 있다. 이를 기반으로 WebSocket에 대한 활용성을 입증하였으며, 이번 연구에서는 WebSocket을 처리 결과를 클라이언트에게 요청을 받을 경우만 데이터를 전송해주는 것이 아닌 서버 상태가 변경되었을 때 자동으로 클라이언트에게 변경사항을 알려주기 위해 활용되었다. 처리 웹에서는 위성영상을 처리하기 위해 처리 웹 표준인 WPS 2.0을 활용하였다. WPS 2.0에는 처리 상태를 확인할 수 있는 상태 인터페이스(getStatus), 결과 인터페이스(getResult)가 존재한다. 이는 처리 서버에서 이루어지는 것으로써 클라이언트는 처리가 되었는지 확인할 방법이 없다. 이를 처리가 완료되면 클라이언트에게 알려주기 위해 WebSocket 기술을 활용하여 처리가 완료되면 클라이언트에게 알림을 주는 방식을 채택하였다. 영상처리를 할 경우 많은 시간이 걸리는 경우가 있다. 이런 경우를 대비하여 WebSocket 기술과 WPS 2.0 비동기식 방식을 활용함으로써 사용자는 처리한 후 반드시 웹 시스템에 접속을 유지할 필요가 없어진다. 웹 시스템 클라이언트 접속이 끊어져도 서버에서는 처리 상태가 유지되고 모든 처리가 끝나면 데이터를 제공할 수 있는 환경으로 모두 알아서 이뤄지도록 설계되었다.

[표 3-8] WPS 2.0 기반 공간정보 처리 시스템 환경

System environment			Version
Remote Sensing Processing Server & Client	Web framework	Spring boot	1.3.3
	Scheduler Library	Quartz	2.2.1
	WebSocket	Stomp	2.3.3
	Template Engine	Thymeleaf	2.1.4
	Markup Language	HTML	5
	Script language	JavaScript	ECMA5
	User Interface	Bootstrap	3.3.6
	Mapping library	OpenLayers	3.10.2

그림 3-19는 위성영상 처리 웹 시스템에 대한 설계도이다. 마찬가지로 MVC 패턴으로 시스템을 구성하였고 WebSocket, Scheduler을 통해 내부적인 데이터 교환도 이루어지는 것을 확인할 수 있다. 기능으로는 WPS 2.0, GeoServer, WebSocket, ProcessStatus으로 나눌 수 있다. WPS 2.0 기능은 WPS2.0 인터페이스를 요청할 수 있도록 클라이언트에서 받은 인자를 알맞게 가공하는 역할을 한다. GeoServer는 융합 웹 시스템과 동일하게 데이터 목록을 시각화하거나 데이터값을 받아오는 기능이 존재하며, WebSocket은 클라이언트에서 요청받을 End-Point에 대한 기능 서버에서 WebSocket Push를 요청할 때 수행되는 기능이 포함되어 있다. ProcessStatus는 처리 중간 상태를 확인할 수 있는 스케줄러 기능이 있다. 이 웹 시스템은 향후 WPS 2.0 인터페이스를 잘 모르는 상황에서 위성영상 처리 클라이언트를 구축하기 위한 미들웨어 비즈니스 어플리케이션으로 활용할 수 있도록 RESTFul API를 구성하였다. 그렇기 때문에 클라이언트 개발자는 WPS 2.0 인터페이스 사양을 모를지라도 단순 웹 시스템 개발하듯이 요청함으로써 위성영상 처리를 수행할 수 있다는 장점을 지니고 있다.



[그림 3-19] WPS 2.0 활용 공간정보 처리 미들웨어 시스템 설계도.

공간정보 및 공공 데이터 융합 웹 시스템 설계 시 분리 형태로 설계하였기 때문에 다양한 시스템 설계 구조를 배포하기 위해 위성영상 처리 웹 시스템은 서버와 클라이언트를 하나로 통합하여 설계하였다. 통합된 시스템은 PaaS에 배포될 때 자바 기반의 빌드팩을 활용하여 배포될 수 있다. 위성영상 처리 웹 시스템도 융합 웹과 마찬가지로 서버와 클라이언트는 언제든지 분리될 수 있는 구조로 설계되었다.

제 4 장 구현결과 및 성능 테스트

제 1 절 공간정보 처리 PaaS 서비스 구현 결과

이번 장에서는 3장에서 설계 및 구축한 IaaS 및 PaaS에 웹 서비스를 배포한 결과를 정리하였다. 웹 서비스는 직접 설계 및 구축한 것으로 향후 공간정보 분야에서 범용으로 활용될 수 있는 공간정보 및 공공 데이터 융합 웹 시스템과 위성정보 처리 시스템이다. PaaS에 웹 시스템을 배포함과 동시에 각 웹 앱에서 사용되는 서비스(미들웨어 소프트웨어)들을 바인딩하였다. 서비스 구축방법에 따라 바인딩에 대한 차이점을 확인하였다. 배포된 웹 앱에 대한 기능 구동 여부 결과를 확인하였다. 마지막으로 클라우드 서비스 형태와 국내 표준프레임워크 적용 여부에 대해 성능 테스트를 수행하였다. 본 연구에서 수행된 성능 테스트는 PaaS와 IaaS를 통해 배포되는 앱은 크게 차이가 존재한다. 일반적으로 IaaS는 가상 머신에 직접 앱에 배포되는 형태로 제공되고, PaaS는 컨테이너에 앱이 배포되는 형태이다. 이러한 차이점이 앱 제공 시 영향을 미치는지에 대한 성능 테스트를 수행하였다. 두 번째, 성능 테스트는 국내 전자정부 시스템에 적용되는 표준프레임워크로 현재 PaaS로 전환하기 위한 연구가 진행 중이다. 공간정보 분야에서 적용될 수 있는 서비스들은 정부를 통해 제공될 수 있으므로 해당 기술을 적용하였을 때 성능에 대해 확인을 해 볼 필요가 있다고 본다. 다양한 방식으로 시스템을 설계하고 구축하여 실제 구동 가능한 형태로 가공하고 테스트 소프트웨어를 통해 성능 테스트를 수행하였다.

1) PaaS 기반 공간정보 앱 배포 및 서비스 바인딩 결과

이번 연구에서 구축된 PaaS인 클라우드 파운더리에 앱을 구동하기 위해서는 첫 번째로 구동될 앱을 배포한 후 필요한 서비스에 대해 바인딩하는 과정이 수행된다. 앱을 배포하는 과정은 간단하다. 클라우드 파운더리 CLI 명령어

인 cf push를 통해 배포가 가능하다. 배포 후 cf bind-service CLI 명령어를 통해 원하는 서비스를 바인딩할 수 있다. 표 4-1은 앱이 배포될 때 클라우드 파운더리에서 자동으로 매칭되는 빌드팩 목록을 정리한 것이다. 기본적으로 제공해주는 빌드팩은 9개가 존재한다. StaticFile은 HTML, JavaScript, CSS와 같이 정적파일로 구성된 파일을 구동하는 빌드팩이다. Java 빌드팩은 Grail, Spring 등과 같이 자바 기반 언어 또는 프레임워크를 지원하는 빌드팩이다. 나머지 빌드팩의 경우에도 해당 빌드팩 이름과 동일하게 각 언어를 지원하는 빌드팩이다. 이때 Position 숫자가 높을수록 우선순위가 낮으며 나중에 검색을 하게 된다. 빌드팩은 cf push를 통해 개발된 앱이 업로드되면서 프로젝트의 고유한 특성을 판단하여 자동으로 빌드하기 위한 빌드팩을 연결한다. 예를 들어 자바 기반 메이븐 프로젝트에 대해 빌드팩을 자바로 선택하기 위해서 메이븐 프로젝트에 필요한 pom.xml 파일이 존재하는지 확인한다. 클라우드 파운더리에서 제공해주는 외에도 사용자 정의 빌드팩을 추가할 수 있다.

[표 4-1] 클라우드 파운더리 제공 빌드팩

Buildpack	Position	Enable	Locked	filename
Staticfile_buildpack	1	true	false	Staticfile_buildpack-cached-v1.4.11.zip
Java_buildpack_offline	2	true	false	java-buildpack-offline-v3.18.zip
Ruby_buildpack	3	true	false	Ruby_buildpack-cached-v1.6.44.zip
Nodejs_buildpack	4	true	false	Nodejs_buildpack-cached-v1.6.3.zip
Go_buildpack	5	true	false	Go_buildpack-cached-v1.8.5.zip
Python_buildpack	6	true	false	Python_buildpack-cached-v1.5.20.zip
Php_buildpack	7	true	false	Php_buildpack-cached-v4.3.38.zip
Dotnet_core_buildpack	8	true	false	Dotnet-core_buildpack-cached-v1.0.22.zip
Binary_buildpack	9	true	false	Binary_buildpack-cached-v1.0.13.zip

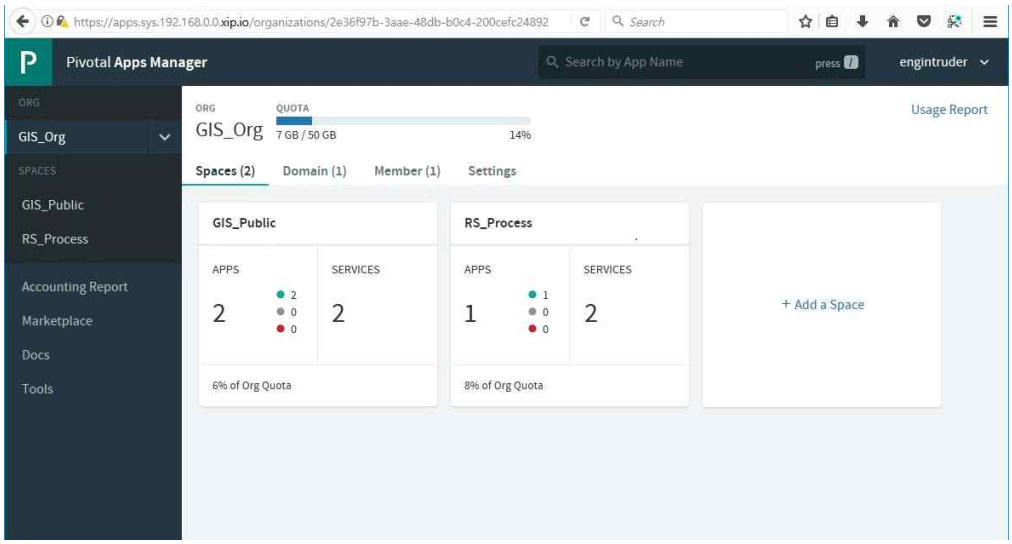
구축된 클라우드 파운더리에 이번 연구에서 설계 및 구현한 공간정보 처리 웹 어플리케이션을 배포하였다. 클라우드 파운더리에 앱을 배포할 때 배포되는 앱에 대해 상세 설정을 위해 manifest 파일이 필요하다. manifest 설정 파일은 YAML 형식으로 구성되어 있다. 배포하려고 하는 앱 최상위 폴더에 manifest.yml 파일을 생성하여 설정 후 배포하면 설정에 맞게 앱이 배포되고 그렇지 않을 경우 클라우드 파운더리 기본설정에 맞춰 배포된다. 설정 파일이 없으면 접속 URL 또한 임의로 생성되기 때문에 manifest 파일을 생성하여 배포하기를 권장하고 있다. 이번 연구에서 배포한 앱은 총 세 개다. 모든 앱에 manifest 파일을 표 4-2와 같이 생성하여 원하는 구성에 맞춰 배포할 수 있도록 구축하였다. 스프링 기반의 앱으로 구성되어 있는 공간정보 및 공공데이터 융합 웹 관리 시스템과 위성정보 처리 웹 시스템은 기본 메모리를 2GB로 설정하고 HTML, JavaScript, CSS로 구성된 클라이언트 앱은 1GB로 설정하였다. 배포가 이상 없이 되어 라우터를 통해 서비스 가능한지 확인하기 위한 방법으로 기본 방법인 port로 설정하였다. 이는 배포 빌드팩에 따라 관련된 포트가 열려있어 제대로 구동되는지 확인하는 설정으로 스프링 프레임워크의 경우 톰캣 기반 컨테이너를 사용하여 배포되기 때문에 8080포트로 구동 여부를 확인하며 클라이언트의 경우 nginX(Engine X)컨테이너를 구동되기 때문에 80포트로 구동 여부를 확인한다. 확인 방법으로는 port 외에도 process, http를 선택할 수 있다. http는 GET 방식으로 URL을 요청함에 있어 이상 없음(HTTP 200)으로 응답 결과가 오는 것이다. process는 어플리케이션이 TCP 연결을 제공하고 있지 않을 때 사용하는 것으로 Diego에서 프로세스 상태를 확인한다.

[표 4-2] 클라우드 파운더리 배포 앱 manifest 설정

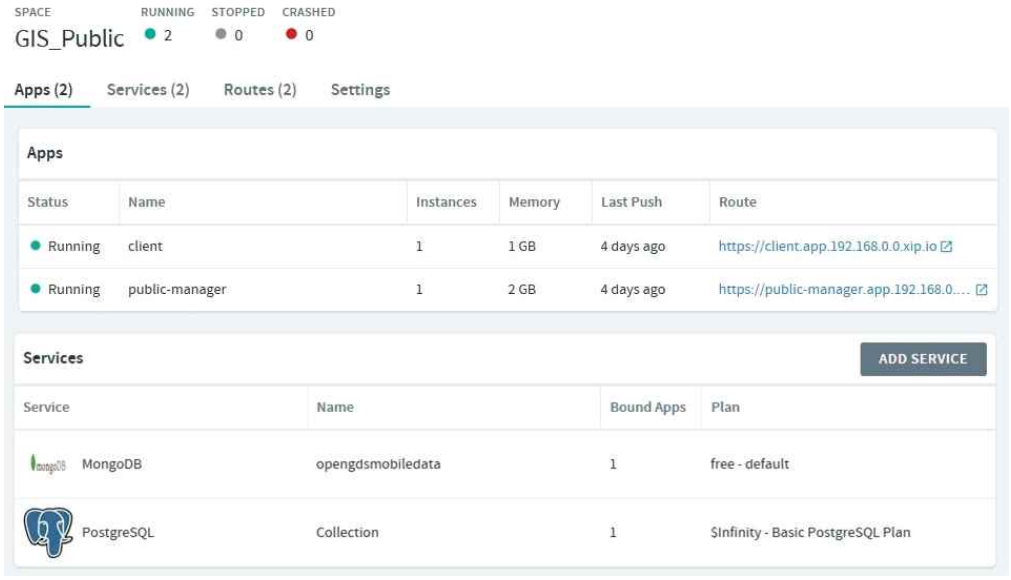
Name	Memory	Route	Health Check Type
public-manager	2GB	public-manager.app.192.168.0.0.xip.io	Port
client	1GB	client.app.192.168.0.0.xip.io	Port
wps2	2GB	wps2.app.192.168.0.0.xip.io	Port

그림 4-1은 배포를 위해 생성된 Org와 Space를 생성한 결과이다. 공간정보 처리 시스템을 배포하기 위한 Org를 GIS_Org로 생성하였다. 이렇게 생성된 Org 내부에는 두 개의 공간이 존재한다. 먼저, 공간정보 및 공공 데이터 융합 시스템 앱이 배포될 공간으로는 GIS_Public 이다. 그리고 위성정보 처리 웹 시스템 앱이 배포될 공간은 RS_Process이다. GIS_Public 공간에는 두 개의 앱과 두 개의 서비스를 사용하고 RS_Process 공간에는 한 개의 앱과 두 개의 서비스를 사용한다.

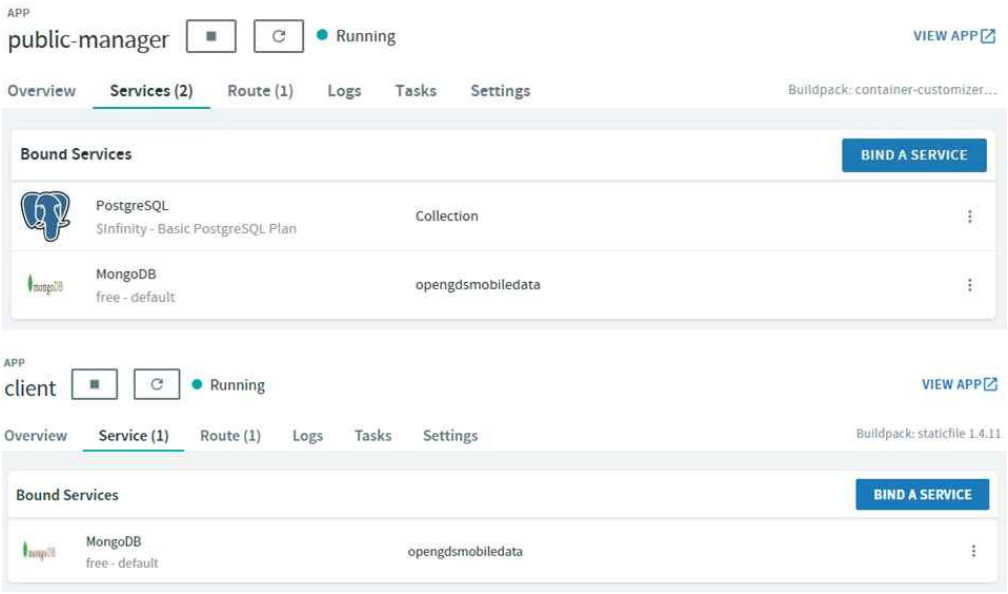
그림 4-2와 그림 4-3은 공간정보 및 공공데이터 융합 웹 시스템 공간 내부에 앱을 배포 및 서비스 바인딩한 결과이다. 그림 4-2에서 확인할 수 있듯이 공간정보 및 공공데이터 융합 웹 시스템 관련된 앱은 두 개로 배포된 것을 확인할 수 있다. 공공데이터 수집을 관리하기 위한 앱인 public-manager와 수집된 데이터를 융합 시각화하는 client 앱이다. 메모리와 라우트는 설정된 값에 맞춰 배포되었으며 초기 생성 인스턴스가 한 개로 되어 있는 것을 확인할 수 있다. 그림 4-2에서 보이는 서비스 탭은 공간 앱으로 바인딩한 결과가 아닌 공간에서 사용할 서비스를 추가한 결과이다. 서비스를 추가할 때 Plan을 지정할 수 있다. Plan은 서비스 구축 시 사용자가 서비스를 사용할 수 있는 범위를 지정해놓을 것으로 여러 Plan을 지정해놓음에 따라 서비스 사용에 따른 관리를 할 수 있다. 예를 들어 데이터베이스 서비스에서 사용될 수 있는 Plan은 사용 공간 크기에 대한 범위로 Space마다 사용될 데이터베이스 테이블의 저장 크기를 제한할 수 있다. 이번 연구에서는 전반적인 테스트를 위해 Plan을 지정하지 않고 제한 없이 사용할 수 있는 Plan만을 활용하였다. public-manager와 client 앱은 동일한 MongoDB 데이터베이스에 접근하여 활용하는 것이 적절하다. 서비스 사용 구분은 Org가 아닌 Space에 따라 구분된다. 같은 공간에서는 동일한 서비스를 공유하도록 되어있다. 그림 4-3에서 확인할 수 있듯이 public-manager 앱은 PostgreSQL과 MongoDB 서비스를 모두 활용한다. 하지만 Client에서는 관리를 위해 사용되는 PostgreSQL은 사용할 필요 없으므로 MongoDB 서비스만 바인딩하였다. 그리하여 public-manager 앱에서 사용되는 서비스는 두 개이며 client 앱에서 사용되는 서비스는 한 개가 되는 것을 확인할 수 있다.



[그림 4-1] 피보탈 클라우드 파운더리 기반 공간정보 앱 배포를 위한 Org / Space 생성 결과.

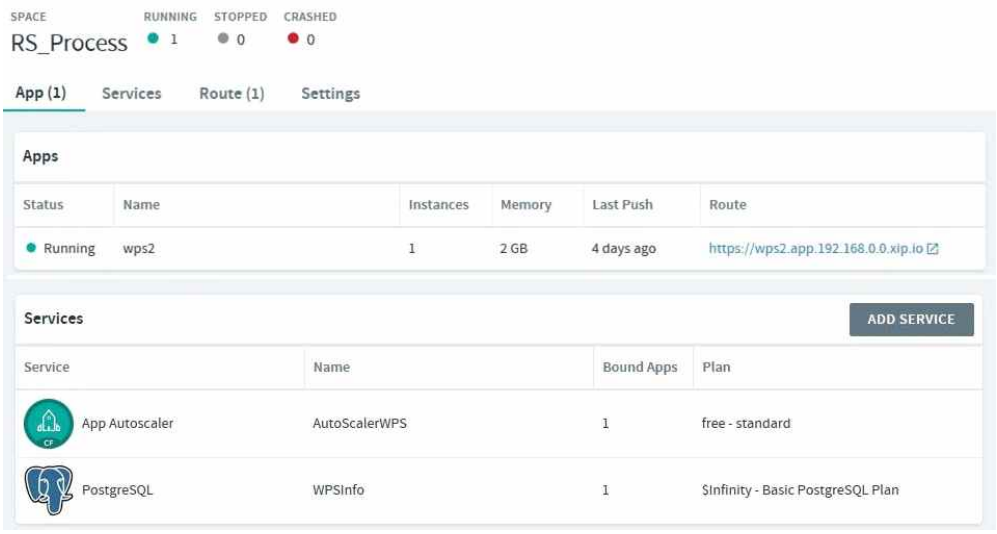


[그림 4-2] GIS_Public Space 공간정보 및 공공데이터 융합 관리 및 시각화 앱 배포 및 서비스 추가 결과.

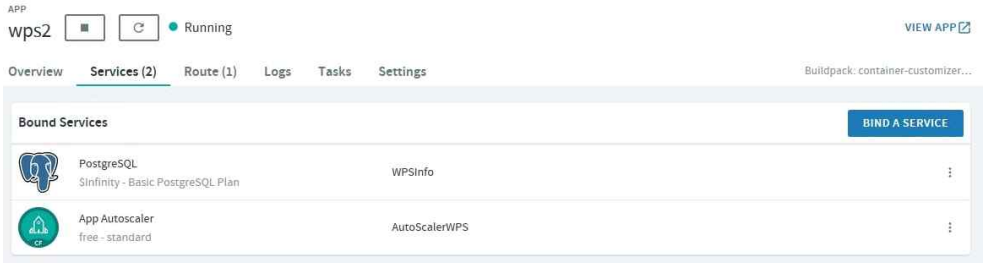


[그림 4-3] 공간정보 및 공공데이터 융합 관리 및 시각화 앱 배포 상세 시각화 및 서비스 연결 결과.

그림 4-4와 그림 4-5는 공간정보 처리 웹 시스템 공간 내부에 앱을 배포 및 서비스 바인딩한 결과이다. 공간정보 처리 웹 시스템의 경우 서버와 클라이언트를 별도로 구분하지 않고 하나의 앱으로 설계하였다. 그렇기 때문에 그림 4-4에서 확인 할 수 있듯이 배포되는 앱 결과는 하나이다. 사용되는 서비스도 PostgreSQL 하나만 사용하도록 되어있다. 추가적으로 이번 연구에서는 피보탈 클라우드 파운더리를 설치하면 기본으로 구성되어 있는 서비스인 오토스케일러를 공간정보 처리 웹 시스템에 바인딩하여 구동 방식을 확인하고자 한다. 그림 4-5는 공간정보 처리 웹 시스템에서 실제 바인딩 된 결과를 보여준다. 공간정보 처리 앱에서는 실제로 PostgreSQL 서비스와 App Autoscaler 서비스를 활용하는 것을 확인할 수 있다.



[그림 4-4] 공간정보 처리 앱 배포 결과.



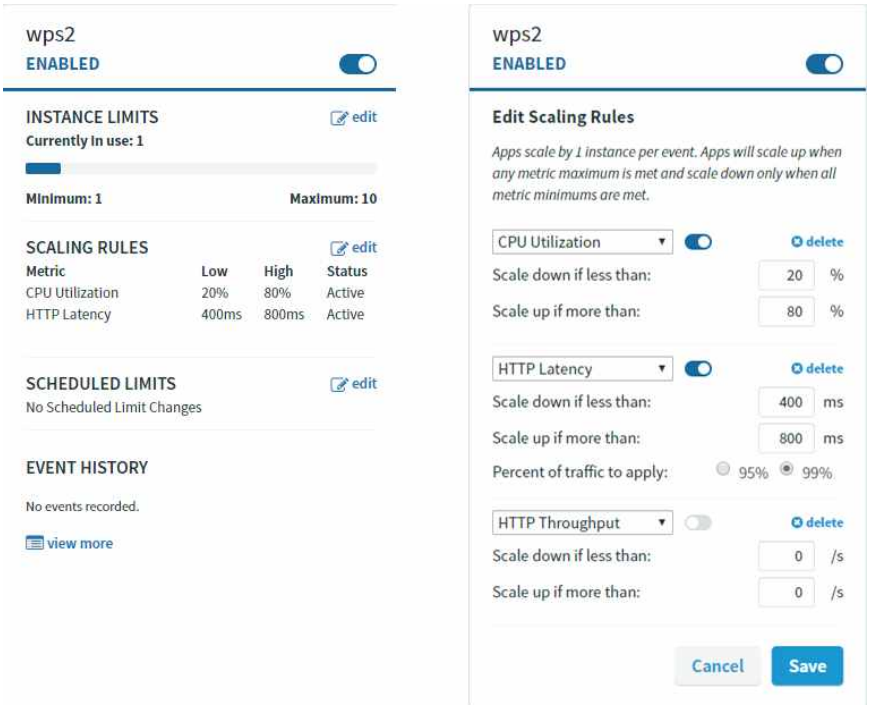
[그림 4-5] 공간정보 처리 앱 배포 상세 및 서비스 연결 결과.

PaaS 기반 공간정보 앱 배포 및 서비스 바인딩 마지막 결과로 오토스케일러를 바인딩한 결과로 오토스케일러 서비스는 바인딩 후 기본적으로 비활성화되어 있다. 그림 4-6에서 볼 수 있듯이 메니저 사용자 인터페이스를 통해 이를 활성화할 수 있으며 설정할 수 있는 기능은 크게 4개로 구성되어 있다. 제일 먼저 설정할 수 있는 사항으로는 최소/최대 인스턴스 생성 개수이다. 그리고 CPU Utilzation, HTTP Latency, HTTP Throughput 세 가지의 스케일 규칙에 따라 구동되는 것을 확인할 수 있다.

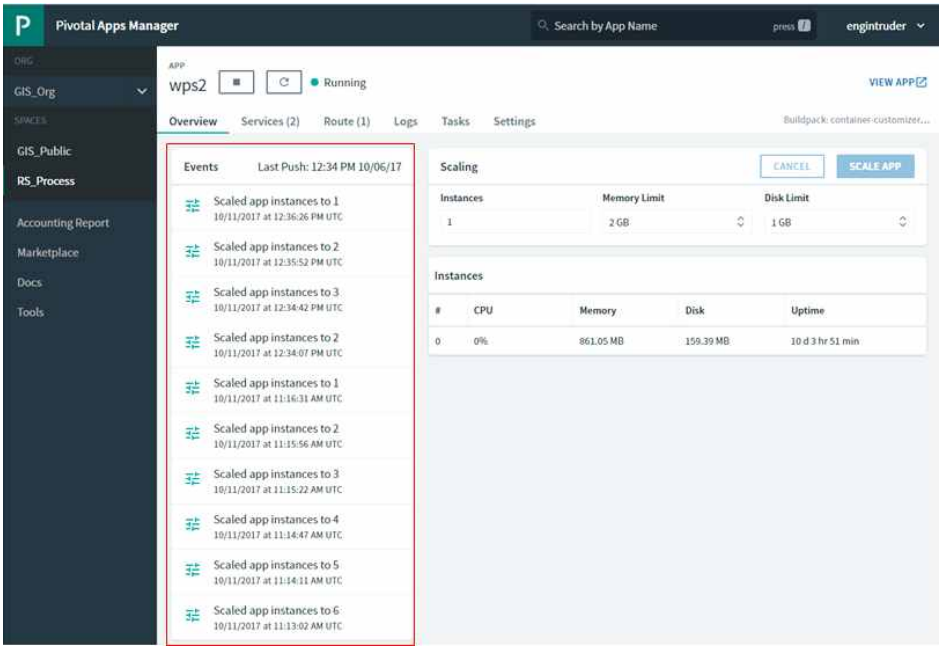
그림 4-7은 실제 오토 스케일 서비스가 구동된 결과로 설정한 값에 맞춰 앱이 배포된 컨테이너가 증가하고 다시 감소한 것이다. 오토 스케일 테스트를

위해 HTTP 트래픽을 임의로 보내 자동 생성 및 삭제되는 사항을 확인하였다. CPU Utilization이 20% 작을 경우 인스턴스 삭제, 80% 이상일 경우 인스턴스 생성으로 설정하였으며 최대 10개까지 생성하도록 설정하였다. 인스턴스는 실제 가상머신이 생성되는 것이 아닌 컨테이너 방식으로 생성되므로 생성과정이 매우 빠른 것을 확인했으며 가상 라우터를 통해 알아서 로드 밸런싱이 이루어진다. 수동으로도 앱 인스턴스를 추가할 수 있으며, 초기 설정된 값이 아닌 메모리나 디스크 공간을 변경하여 인스턴스를 추가할 수 있다.

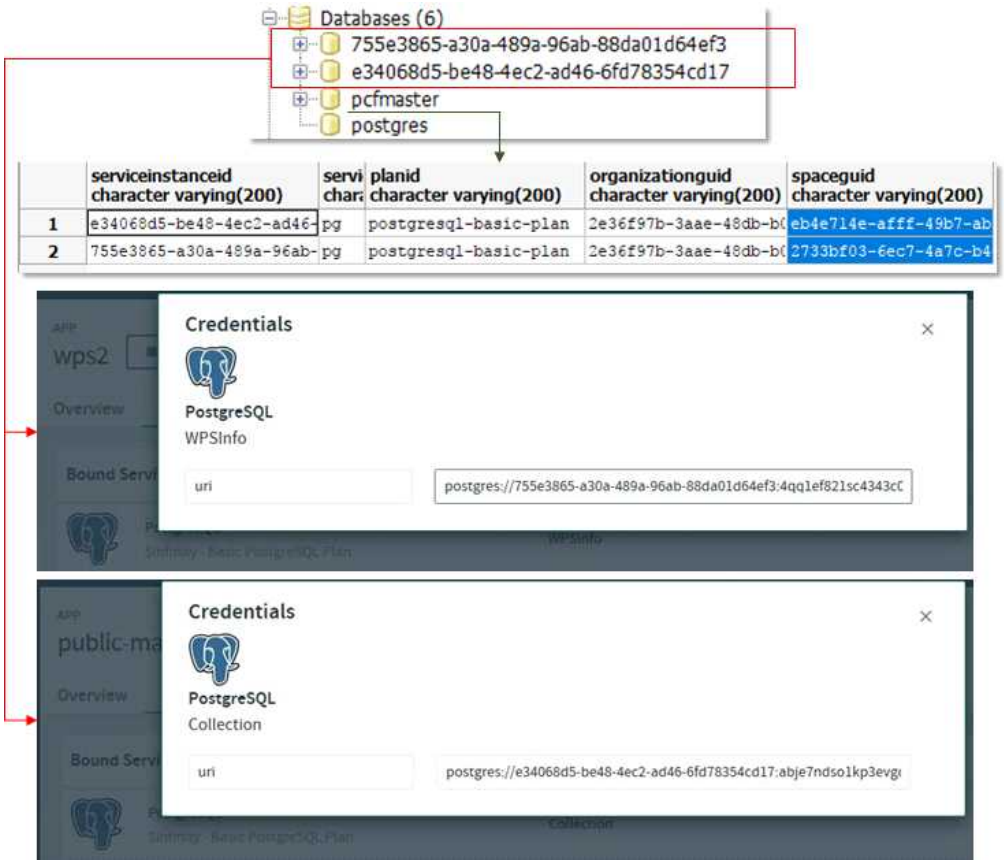
실제 서비스 구동 결과로 그림 4-8은 앱에서 PostgreSQL 서비스를 바인딩했을 때 PostgreSQL에서 수행된 결과를 확인한 그림이다. GIS_Public과 RS_Process Space 모두에서 PostgreSQL은 사용된다. PostgreSQL 서비스 브로커에서는 서비스가 공간에 추가될 때마다 새로운 데이터베이스를 생성하고 이 데이터베이스를 관리할 수 있는 사용자를 추가하도록 구축되어 있다. 이를 실제적으로 확인하기 위해서는 앱이 배포된 상세 페이지에서 Credentials 메뉴를 클릭하여 접속 정보(데이터베이스 이름, 사용자 이름, 사용자 비밀번호)를 확인할 수 있다. 이렇게 생성된 정보는 클라우드 파운더리 PostgreSQL 서비스 구축 시 자동으로 생성된 데이터베이스(pcfmaster) service 테이블에서 관리되는 것을 확인할 수 있다. 총 두 개의 서비스 인스턴스가 생성되고 이에 따른 두 개의 데이터베이스가 생성된 결과이다. 추가된 서비스를 삭제하면 해당 데이터베이스가 삭제되고 service 테이블에서도 정보가 삭제된다. MongoDB도 마찬가지로 구동된다. MongoDB 서비스를 추가하게 되면 관련된 데이터베이스가 임의의 텍스트를 통해 생성되고 이 또한 MongoDB 내부 데이터베이스를 통해 관리되는 것을 확인할 수 있다.



[그림 4-6] App Autoscaler 서비스 매니저 인터페이스.



[그림 4-7] App Autoscaler 서비스 구동 결과.



[그림 4-8] 공간 융합 시각화 및 처리 시스템 서비스 연결에 따른 데이터베이스 자동 생성 결과.

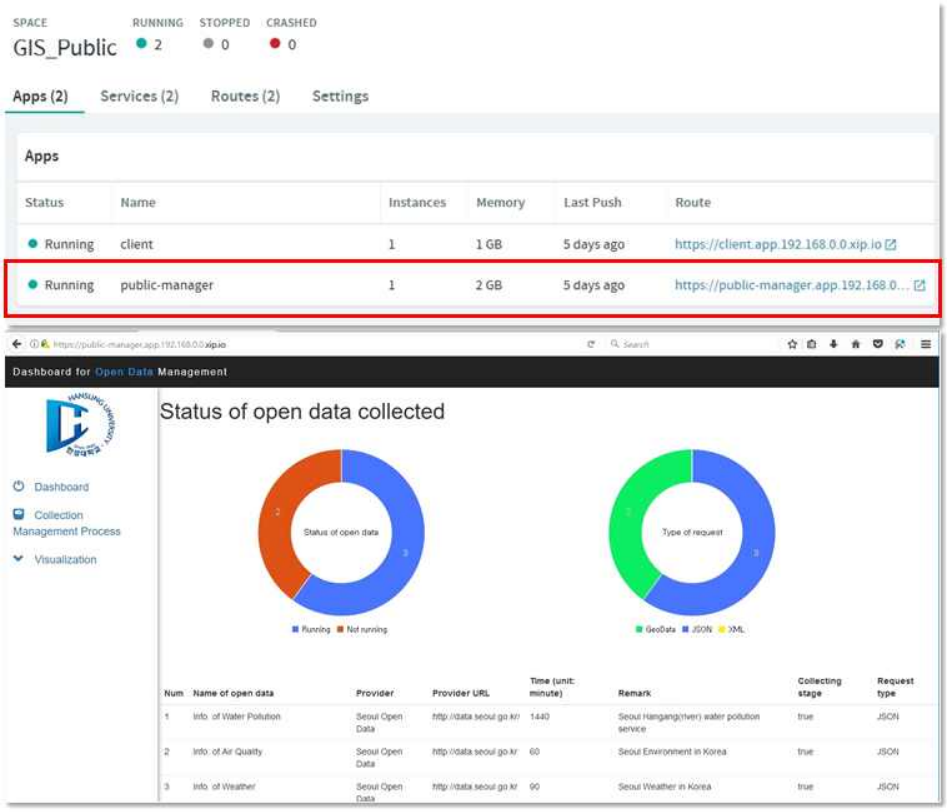
2) 공간정보 처리 서비스 구현 결과

이번 연구에서 설계 및 구현된 웹 앱은 총 세 개다. 공간정보 및 공공 데이터 융합 웹 시스템은 사용자가 마우스 및 키보드로 컨트롤 할 수 있는 관리와 시각화 웹 인터페이스 앱으로 따로 존재하며, 위성정보 처리 서버 시스템은 처리 서버와 웹 클라이언트 인터페이스가 통합되어있다. PaaS에 배포된 시스템이 제대로 구동되는지에 대해 기능별로 수행 결과를 확인하였다. 이번 연구에서 구축된 클라우드 및 시스템 환경은 폐쇄 망으로 구축되었다. 그렇기 때문에 배포된 앱을 확인하기 위한 경로는 사설 네트워크에 구축되어 있으며 외부에서는 접근이 불가하다. 실제 서비스가 필요하다면 외부에서 접속 가능

한 DNS 서버를 구축함으로써 서비스가 가능하다. DNS 서버 구축 시 필요한 사항은 다음과 같다. 또한, 클라우드 파운더리에서 기본적으로 보안 통신인 HTTPS 방식을 채택하고 있으므로 모든 결과에서 접속하기 위한 URL은 HTTPS로 구성되어 있다. 그렇기 때문에 향후 실제 서비스를 위해서는 이에 대한 인증서도 필요하다.

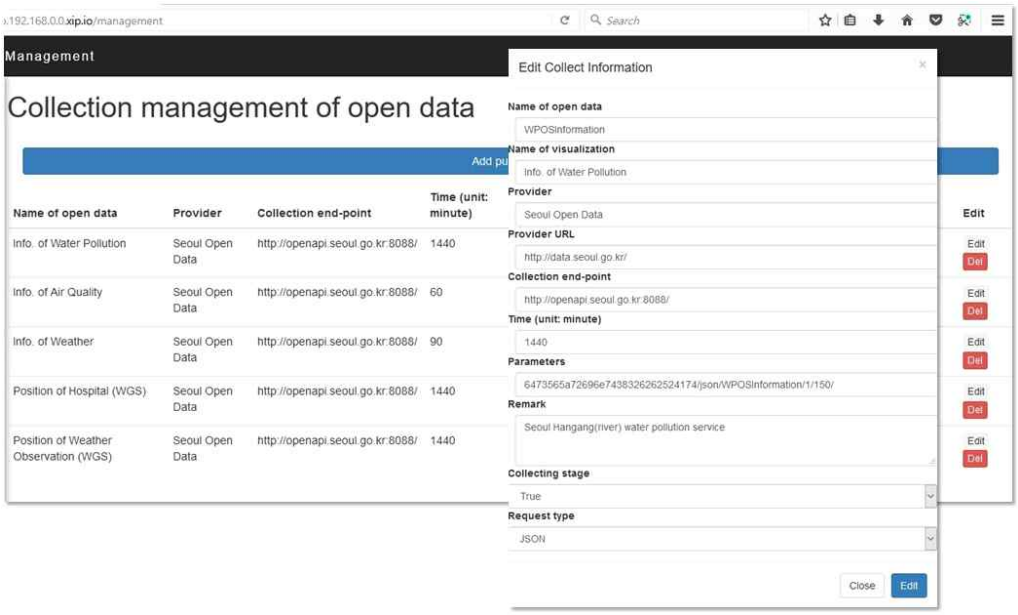
가) 공간정보 및 공공데이터 융합 처리 앱 결과

공간정보 및 공공데이터 융합을 위해 관리하는 시스템은 public-manager 이름으로 배포되었다. public-manager 앱을 접속하기 위해서는 https://public-manager .app.192.168.0.0.xip.io를 통해 접속할 수 있다. 그림 4-9는 공공 데이터 수집 관리를 위한 관리 대시보드 웹 인터페이스 메인 화면이다. 메인 화면에서는 현재 수집이 진행되고 있는 공공 데이터를 전반적으로 확인할 수 있도록 차트 시각화 및 테이블이 존재한다. 시각화되는 차트로는 현재 수집이 되고 있는 공공 데이터 개수와 공공 데이터 요청에 대한 응답 타입을 확인할 수 있다. 테스트를 위해 수집 가능한 데이터를 추가해 놓은 상태이다. 총 5개의 공공 데이터가 수집되는 것을 확인할 수 있으며, 그중에서 3개가 일정 시간에 맞춰 수집되는 것을 확인할 수 있다. 수집되는 공공 데이터로는 서울 열린 데이터 광장의 측정소별 시간 평균 대기질 값, 지역별 세 시간마다 업데이트되는 날씨 정보, 한강 근처 수질 오염도에 대한 데이터이다. 측정되는 일정은 단위가 분으로 설정할 수 있다. 대기질 값의 경우 시간별로 되어 있으므로 60분마다 측정되도록 되어있고 날씨의 경우 세 시간마다 업데이트되므로 90분마다 수집되도록 등록하였다. 마지막으로 수질 오염의 경우 하루에 한 번 갱신되기 때문에 1440분마다 수집되도록 등록하였다.

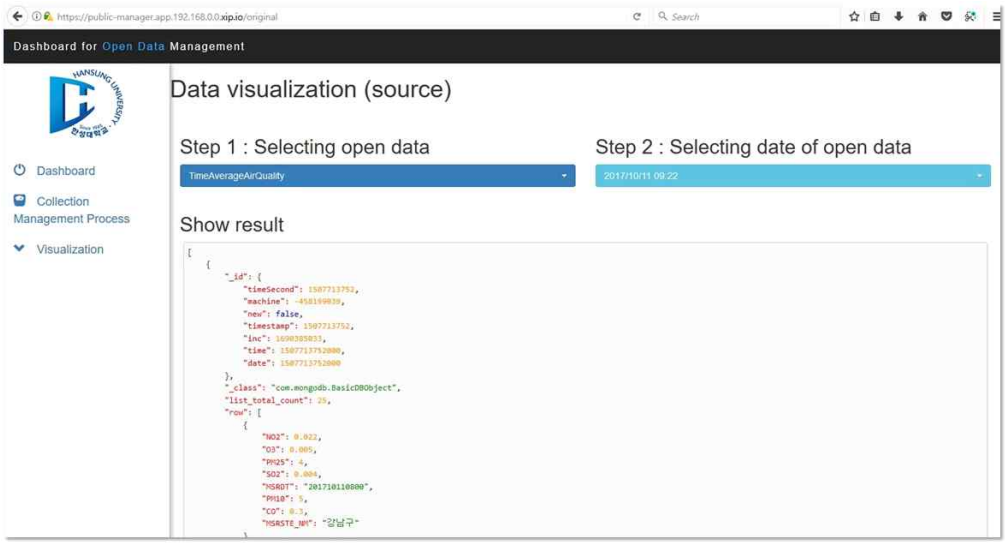


[그림 4-9] 공간정보 및 공공데이터 융합 처리 앱 접속 Route 확인 및 관리자 웹 대시보드 인터페이스.

그림 4-10은 수집 데이터에 대한 정보를 등록, 수정, 삭제하는 인터페이스이다. 공공 데이터 URL 패턴을 분석할 때 확인되었던 사항으로 앤드포인트, 키 값, 응답 포맷, 서비스 이름, 데이터 범위 등이 있었다. 이러한 사항을 각 변수로 저장해놓고 설정해놓은 시간에 따라 요청할 수 있도록 입력하는 인터페이스이다. 여기에서는 실제로 데이터 수집에 대한 스케줄을 시작 및 정지할 수도 있다. 그림 4-11은 수집된 원본 데이터를 확인할 수 있는 페이지이다. 데이터를 선택하면 수집된 날짜가 목록으로 시각화된다. 이 중 원하는 날짜를 선택하면 수집된 데이터 원본을 확인할 수 있다. 서울 열린 데이터 광장의 경우 대부분 데이터를 JSON 형식으로 응답하는 것이 가능하다. 그렇기 때문에 원본 데이터는 JSON 형식으로 저장되며 실제 데이터를 별도 가공하지 않고 바로 row 키 값으로 저장한다.

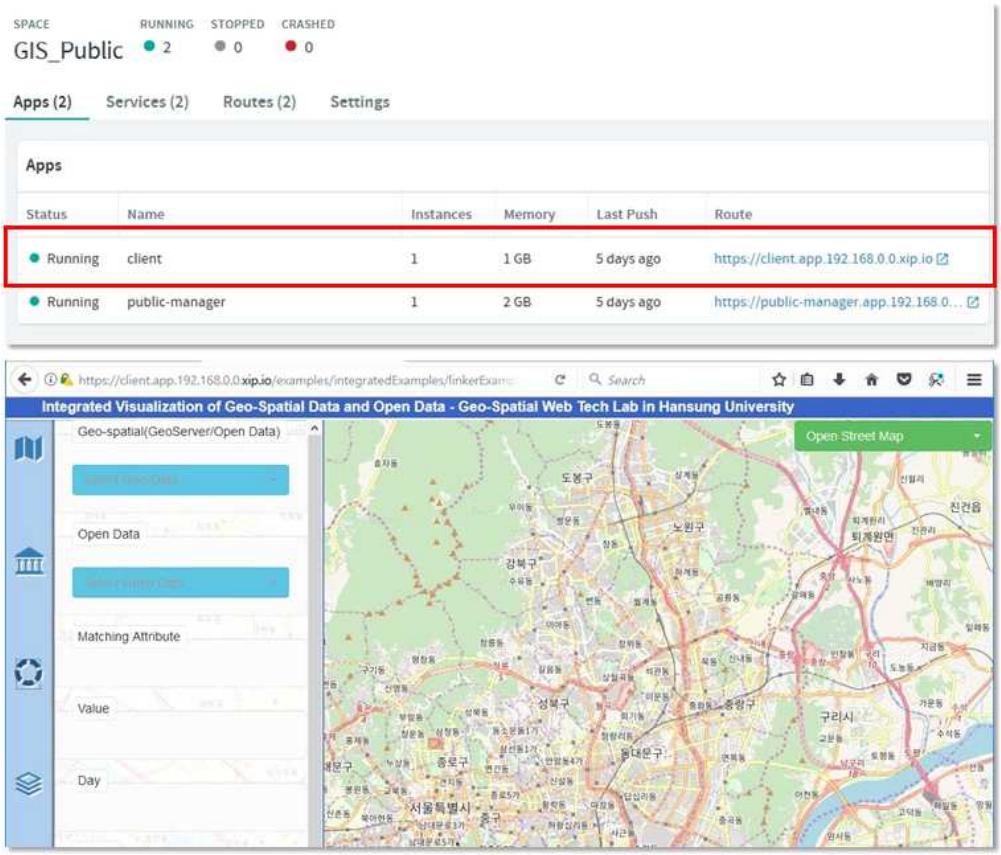


[그림 4-10] 공간정보 및 공공데이터 융합 처리 앱 데이터 관리 인터페이스.

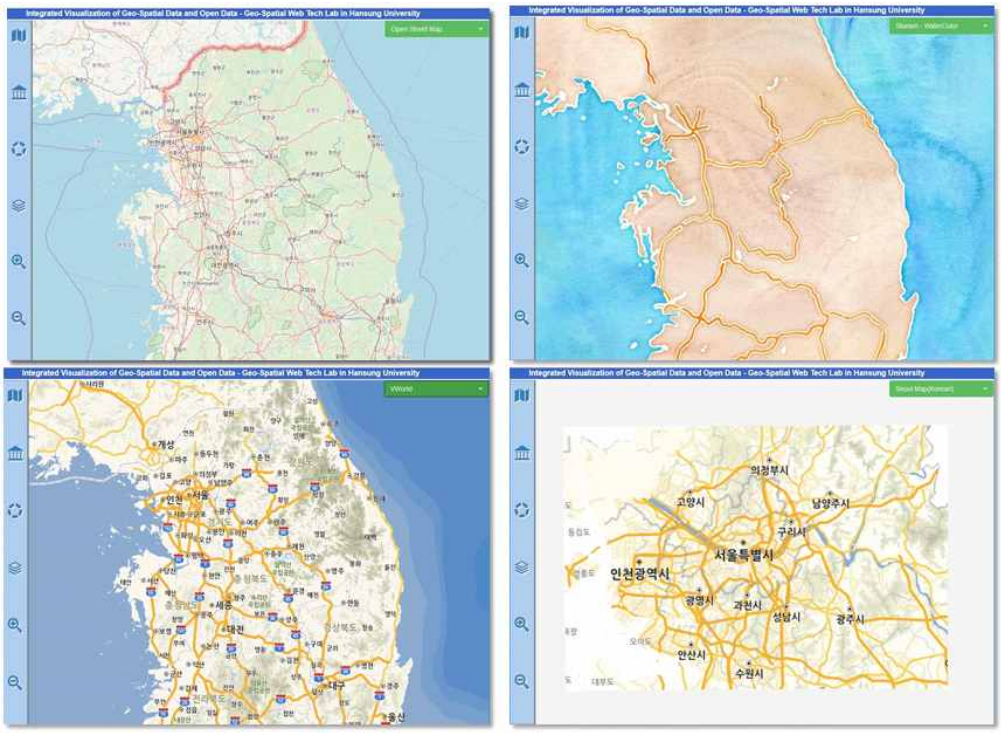


[그림 4-11] 공간정보 및 공공데이터 융합 처리 원본 데이터 시각화.

public-manager 앱은 PaaS 서비스로 제공되는 PostgreSQL과 MongoDB에 공공 데이터 관리 및 수집을 수행하도록 도와주는 인터페이스이다. MongoDB에 저장된 공공 데이터를 클라이언트에서 사용하여 공간정보와 융합하여 시각화할 수 있는 웹 클라이언트 인터페이스는 clinet 앱으로 별도 구성하였다. client 앱은 https://client.app.192.168.0.0.xip.io를 통해 접속할 수 있다. 융합 웹 클라이언트는 공공 데이터를 차트로도 시각화할 수 있으며 공간 속성정보와 매칭하여 시각화할 수 있다. 그림 4-12는 시각화 클라이언트 메인 페이지이다. 지도 기반으로 인터페이스를 구성하였고 그림 4-13에서 보이는 것과 같이 전 세계 데이터(OSM, Stamen 등)와 국내 데이터(브이월드, 서울 지도) 배경 지도를 지원하도록 구성하였다. 지도 매핑 라이브러리로는 OpenLayers를 활용하였다.



[그림 4-12] 데이터 시각화 클라이언트 앱 Route 확인 및 웹 인터페이스.

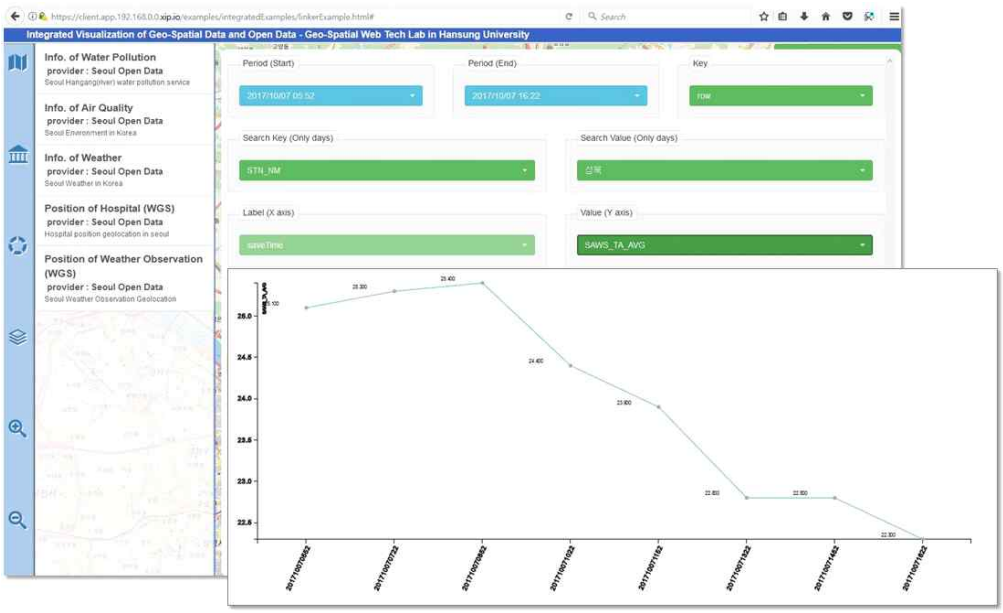


[그림 4-13] 데이터 시각화 클라이언트 앱 배경지도 화면.

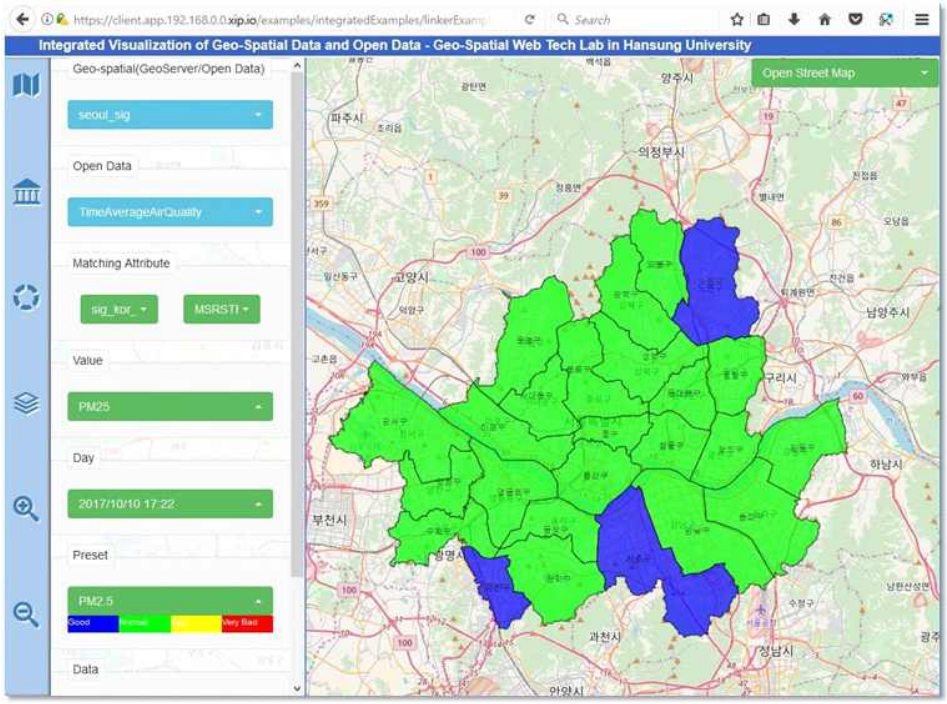
데이터 시각화 클라이언트 앱에서는 수집된 데이터를 차트로 시각화할 수 있다. 차트로 시각화하기 위해 D3.js 차트 라이브러리를 사용하였다. D3는 HTML5 SVG를 사용하여 어느 장치든지 웹 표준을 사용하고 있는 브라우저에서는 적절한 시각화가 가능한 시각화 오픈소스 라이브러리이다(김광섭과 이기원, 2014). 원하는 단일 날짜에 원하는 데이터를 x축 y축을 설정하여 차트로 시각화할 수 있으며, 기간별로 시각화도 가능하다. 그림 4-14는 기간별로 데이터를 시각화한 결과이다. 시각화 된 데이터로는 서울 날씨 정보이다. 먼저, 차트로 시각화할 원하는 시작 일시(2017/10/07 05:52)와 종료 일시(2017/10/07 16:22)를 선택한다. 구축된 웹 클라이언트의 경우 기간으로 데이터를 보여줄 경우 모든 지역에 대해 데이터를 시각화할 수 없으므로 지역에 대해 설정이 필요하다. 날씨 데이터의 경우 STN_NM 키 값에 지역에 대한 정보가 포함되어 있으므로 이를 선택하여 시각화될 지역에 대해 선택한다. 선택이 되면 날씨 정보가 포함되어 있는 SAWS_TA_AVG 키 값을 Y 축 데

이터 값으로 선택한다. 총 10시간에 대한 데이터를 시각화하는 것으로 90분마다 수집을 하므로 8개에 대한 데이터가 차트로 보여진다. 차트 종류는 기본 차트인 막대, 선, 영역 차트로 선택할 수 있으며 그림 4-14는 선 차트로 시각화 한 결과이다. 차트 시각화 시 최소 값과 최대 값을 계산하여 차트 범위가 변경되도록 설정하였다.

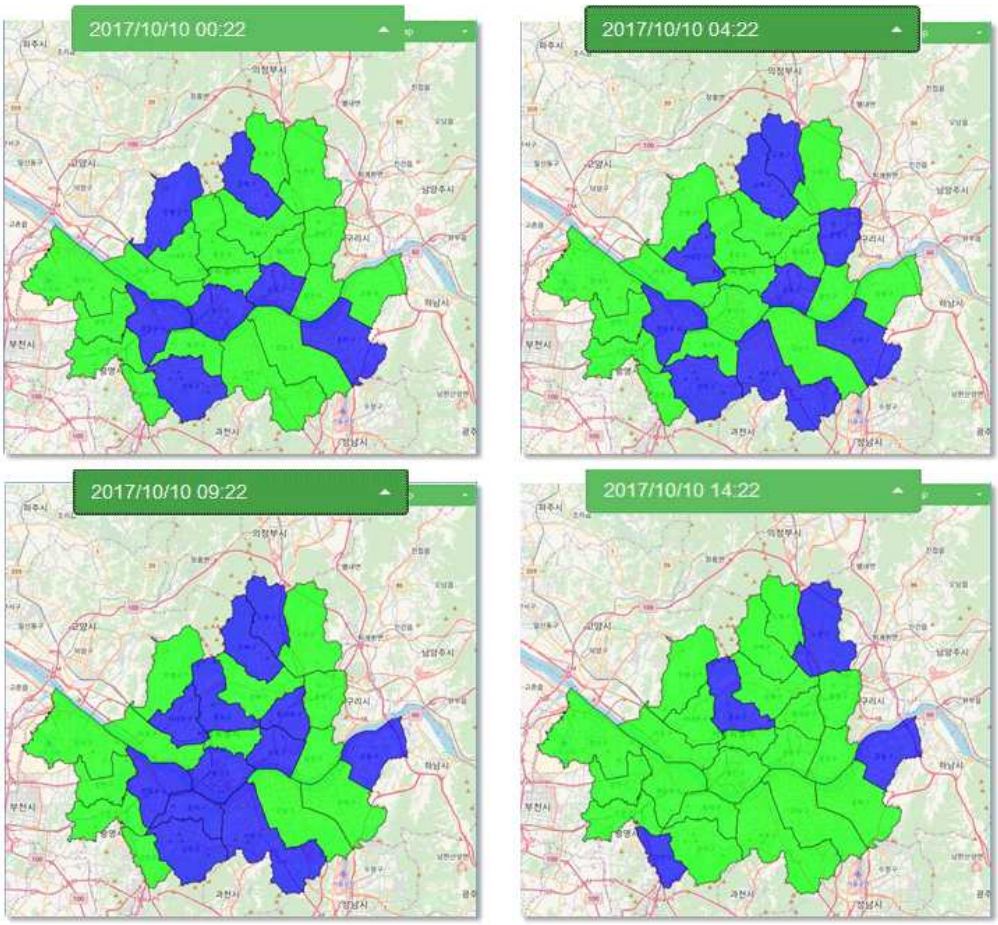
그림 4-15는 공간정보와 공공 데이터를 융합하여 보여주는 결과이다. 간단한 속성정보 매칭이므로 클라이언트에서 실시간으로 수행된다. 공공정보는 PaaS 서비스에서 제공하는 GeoServer에서 제공하는 데이터이다. 서울 열린 데이터 광장의 데이터의 경우 서울 전 지역에 대한 정보를 보여주기 때문에 GeoServer 데이터 중 서울 구 지역 경계구역 데이터를 선택한다. 공간정보 선택 후 공공 데이터에 대해 선택을 한다. 선택한 데이터는 서울 대기질 정보이다. 공간정보와 공공 데이터가 선택되면 두 데이터 속성정보 중 매칭이 되는 키 값을 선택한다. 공간정보에서 구 이름을 저장한 속성 키 값은 sig_kor_nm이고, 대기질 정보에서 구 이름을 저장한 속성 키 값은 MSRSTE_NM이다. 기본적인 데이터가 모두 선택된 후에 실제 보여질 데이터에 대해 선택한다. PM1.0, PM2.5, O3, SO2, CO, NO2에 대한 대기질 정보를 제공하고 있다. 대기질 값을 선택 후에는 데이터 날짜를 선택한다. 대기질의 경우 값에 따라 좋음, 보통, 나쁨, 매우 나쁨을 색상으로 표현할 수 있다. 대기 종류에 따라 값의 범위가 다르므로 선택한 대기 종류에 따른 프리셋을 정하면 그림 4-15와 같이 구 단위로 시각화된 공간정보 위 대기질소 값이 매핑되어 시각화된다. 이후 추가로 날짜를 변경하면 해당 날짜에 맞는 데이터가 실시간으로 시각화된다. 그림 4-16은 모든 설정이 끝난 데이터를 날짜만 변경하여 융합 시각화한 결과로 2017년 10월 10일 00시 22분, 2017년 10월 10일 04시 22분, 2017년 10월 10일 09시 22분, 2017년 10월 10일 14시 22분 데이터를 순차적으로 시각화한 결과이다. 해당 결과를 확인해보았을 때 MongoDB 데이터를 제공 받을 때 공간정보 및 공공데이터 융합 처리 앱을 통해서도 받거나 서비스를 통해 직접 받는 것이 가능하다. 또한 간단하게 확인할 수 있듯이 웹 표준에 대한 서비스가 제공되므로 클라이언트에서 얼마든지 공간정보를 시각화할 수 있다는 장점이 존재한다.



[그림 4-14] 데이터 시각화 클라이언트 앱 날씨 정보 차트 시각화.



[그림 4-15] 데이터 시각화 클라이언트 앱 공간정보(서울) 및 공공 데이터(PM2.5) 융합 시각화.

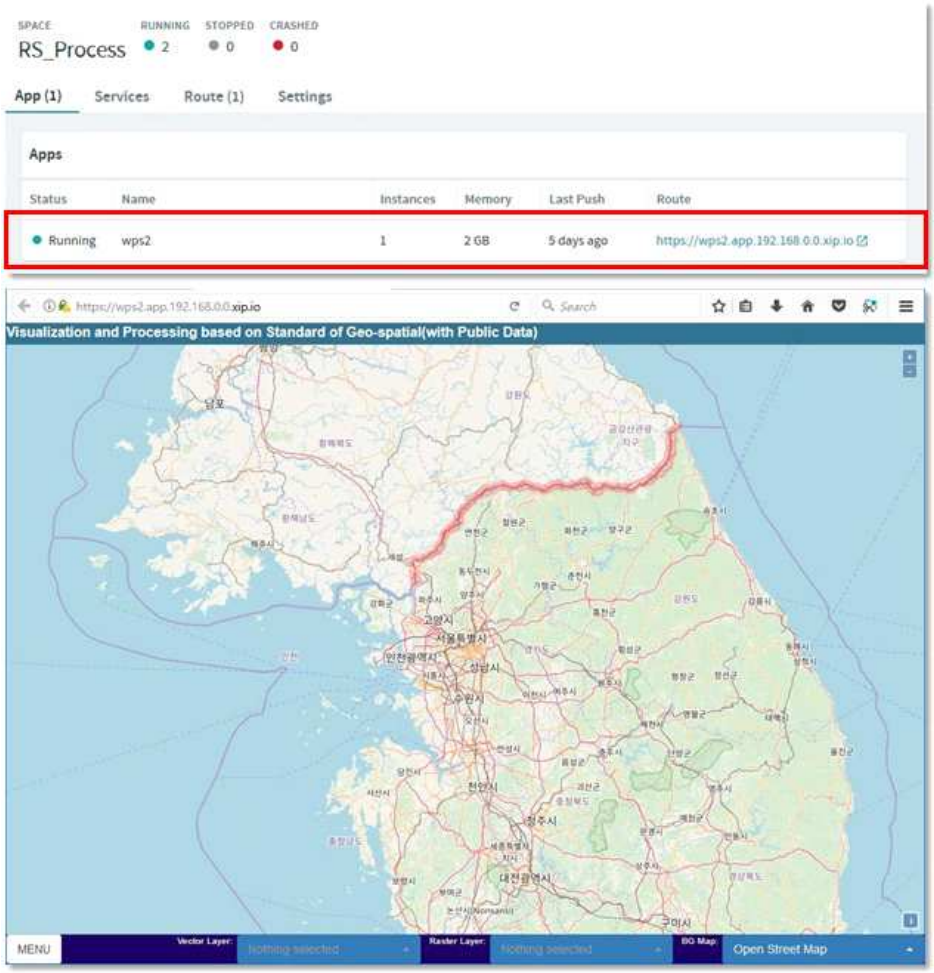


[그림 4-16] 데이터 시각화 클라이언트 앱 날짜 별 융합 시각화 결과.

나) 공간정보 웹 표준 활용 위성영상 처리 앱 결과

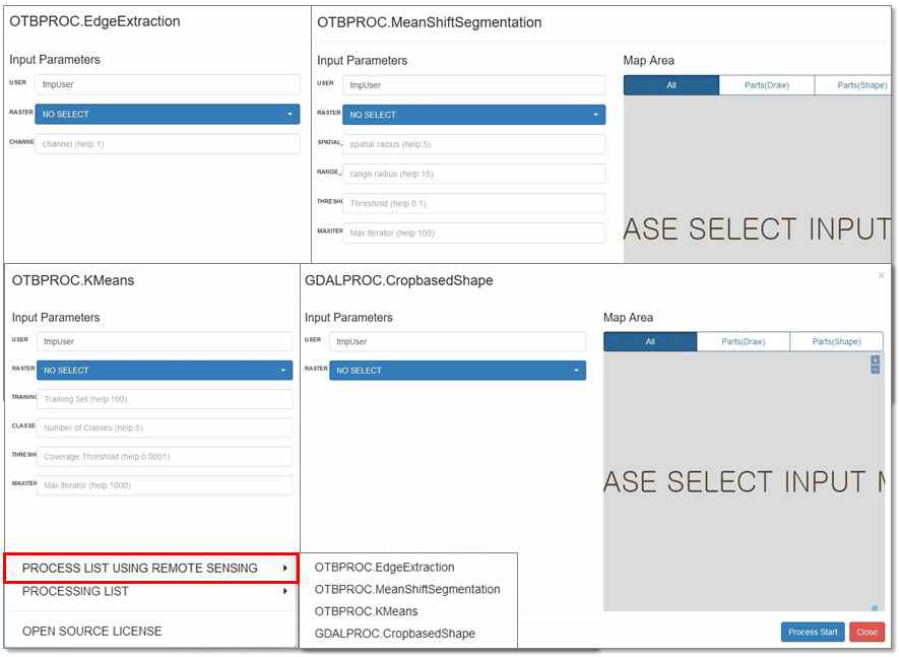
세 번째 배포 결과로 위성영상 처리 웹 시스템 기능 구동에 대해 확인하였다. 위성영상 처리 앱의 경우 서버와 클라이언트가 통합되어 구동되고 있다. 처리를 위해 처리 현황을 저장하는 테이블로 PostgreSQL 서비스를 활용하였다. 공간정보 처리 앱에 접근하기 위해서는 https://wps2.app.192.168.0.0.xip.io를 통해 웹 인터페이스를 확인할 수 있다. 대부분 공간정보 웹 서비스의 경우 지도 기반으로 서비스되는 경우가 대부분이다. 처리 앱 같은 경우에도 배경지도를 기반으로 서비스가 구동되는 것을 그림 4-17에서 확인할 수

있다. 지도 기능으로는 데이터 시각화 클라이언트 앱과 동일하게 다양한 배경 지도를 제공하고 처리를 위해 제공되는 레이어를 선택할 수 있는 데이터 목록이 존재한다. 이 데이터는 인스턴스 기반으로 구축되어 있는 PaaS 서비스 GeoServer를 통해 실시간으로 데이터를 받아올 수 있다. 처리 앱에서 시각화할 수 있는 데이터는 서울과 세종 지역의 벡터 데이터, Landsat 8 데이터로 가공된 2015년 2017년 위성영상을 시각화할 수 있다. GeoServer에서 제공하는 웹 표준을 활용하여 벡터 데이터는 WFS로 시각화되며, 래스터 데이터는 WMTS로 시각화된다.



[그림 4-17] 위성영상 처리 앱 경로 확인 및 처리 웹 인터페이스.

공간정보 처리 앱에는 테스트를 위한 총 네 개의 처리 기능 인터페이스를 지원하고 있다. 먼저 공간정보 입출력 소프트웨어인 GDAL을 Zoo Service에서 사용할 수 있도록 구축하였으며 기능은 GDAL Crop으로 데이터를 원하는 데로 자를 수 있는 기능이다. 나머지 세 가지는 모두 OTB 소프트웨어를 Zoo Service에서 사용할 수 있도록 구축한 것으로 OTB에서 제공하고 있는 기능 중 외곽선 추출, 영상 소프트화, 영상 구분 세 가지를 처리할 수 있도록 인터페이스를 구성하였다. 변수 입력 인터페이스 구성시 WPS 2.0 인터페이스 중 Describe Process를 통해 해당 기능에 대한 입력 변수 정보를 받아와 자동으로 값을 입력받을 수 있도록 엘리먼트를 구성하였다. 그렇기 때문에 향후 Zoo Service에 기능이 추가되더라도 얼마든지 인터페이스를 구성하는데 어려움이 없다. 그림 4-18에서는 공간정보 처리 앱에서 제공하고 있는 위성영상 처리 기능 목록과 기능별로 처리 수행을 위해 필요한 변수 입력 사용자 인터페이스가 생성된 결과를 보여주고 있다.



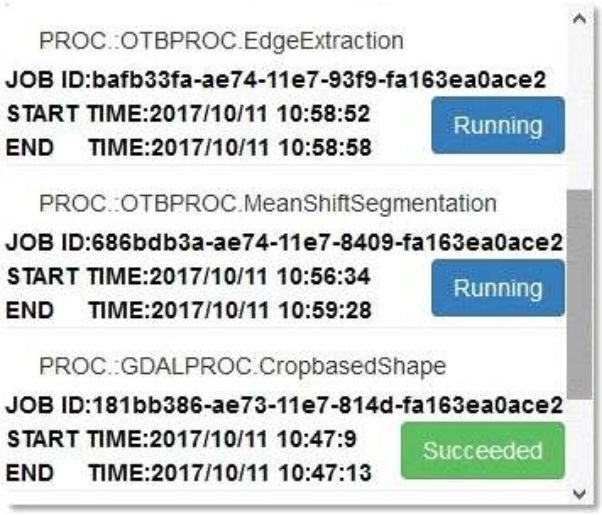
[그림 4-18] 위성영상 처리 앱 처리 목록 시각화 및 변수 입력 인터페이스.

WPS 2.0 인터페이스를 활용하여 위성영상을 처리하는 상태는 총 네 가지 상태로 구분되어진다. 일단 처리가 요청되면 Accepted 단계이다. 이때 사용자가 요청한 변수가 해당 처리를 위해 필요한 변수 개수 및 키 값이 동일한지 확인한다. 확인이 되어 이상이 없을 경우 Running 상태로 처리가 수행되기 시작한다. 만약 해당 값이 문제가 생길 경우 바로 Failed 단계로 변경된다. Running 상태가 처리가 되면서 유지되고 중간에 오류가 날 경우 Failed를 반환하지만 이상 없이 처리가 마무리되면 Succeeded로 변경되는 것을 확인할 수 있다. 이러한 상태는 처리를 수행하는 Execute 인터페이스를 수행하여 반환되는 JobID 값을 입력 값으로 사용하는 getStatus 인터페이스를 통해 상태를 확인할 수 있다. 상태 변화를 확인할 수 있도록 공간정보 처리 앱에 사용자 인터페이스를 구성하였다[그림 4-19].

[그림 4-19] WPS2.0 인터페이스 지원 처리 순서 시각화 인터페이스.

그림 4-20 결과처럼 WPS 2.0의 장점 중 하나인 비동기식 처리로 인해 동시 다중 처리 수행을 할 수 있는 것을 확인하였다. 공간정보 처리 앱에서는 상태 변화를 스케줄러 라이브러리인 쿼츠를 활용하여 수시 Zoo Project 서버

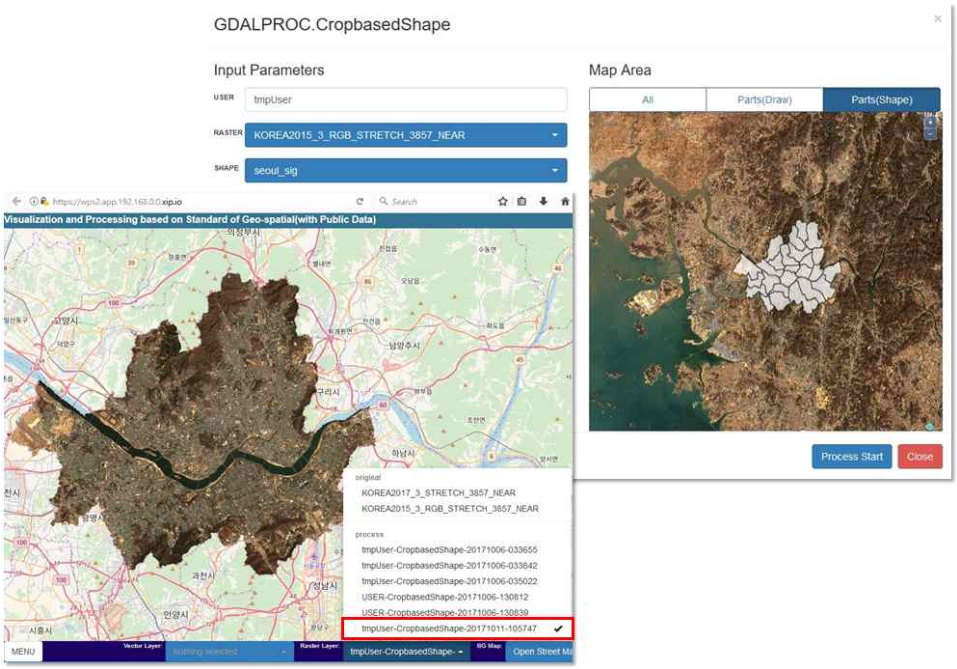
에 getStatus를 통해 확인하고 응답 결과가 Accepted에서 Failed, Accepted에서 Running, Running에서 Succeeded 또는 Failed로 상태 변화가 일어날 때 상태를 저장하는 데이터베이스에 종료 시점 및 상태를 업데이트하고 이를 WebSocket 통신으로 클라이언트에게 Push 한다.



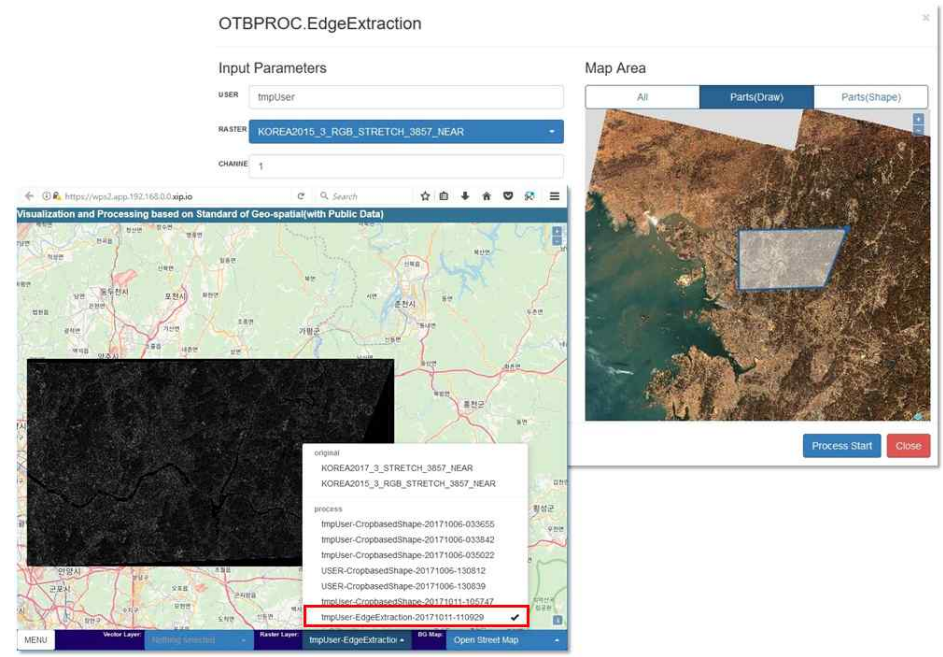
[그림 4-20] 다중 처리 지원 확인 결과.

그림 4-21에서 그림 4-24까지는 간단한 시나리오를 구성하여 기능 처리 결과를 확인한 것이다. 처리 기능을 수행할 때 위성영상 영역에 대해 모든 데이터를 처리할 수 있고 영역을 선택할 수 있다. 영역을 선택할 때에는 지도 위에 폴리곤을 직접 생성하여 영역을 생성할 수 있으며, GeoServer에 저장된 벡터 데이터를 활용할 수도 있다. 처리된 데이터는 메인 화면 하단 부분 레스터 레이어를 선택하는 목록에 PROCESS 타이틀 밑에 차례로 추가된다. 추가된 레이어 이름은 아이디-처리명-처리요청일시 순으로 파일명이 생성되며 아이디의 경우 현재 로그인 기능이 없기 때문에 tmpUser로 임시 작성해놓은 상태이다. 처리 결과 데이터는 GeoTiff 형식으로 GeoServer에 자동으로 등록된다. 그렇기 때문에 사용자는 GeoServer에 직접 접속하여 데이터를 다운로드 받거나 다른 웹 어플리케이션에서도 활용할 수 있다. 먼저 그림 4-21은 2015년 위성영상 데이터를 서울시 지역에 대해서만 확인하기 위해 GDAL 기

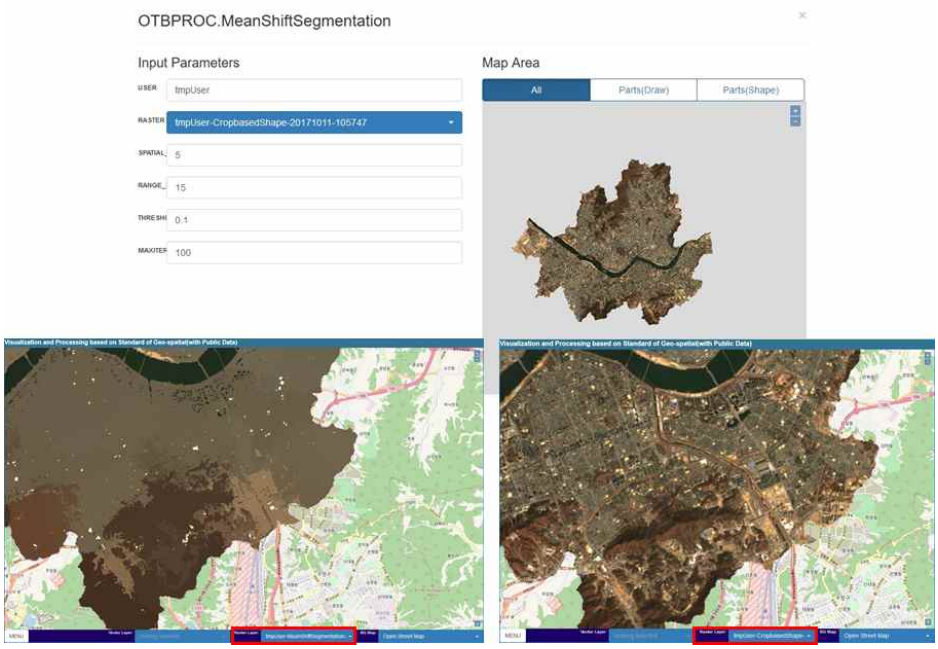
능을 사용하여 자른 결과이다. 입력 변수로 반드시 위성 데이터가 필요하며 자를 영역에 대한 선택이 필요하다. 자를 영역에 대한 선택은 직접 Map Area 부분에 자를 영역을 폴리곤 형태로 생성하거나 GeoServer에 저장되어 있는 전국구 경계면 데이터를 활용할 수 있다. 그림 4-22는 OTB 기능인 엣지 추출 기능을 수행한 결과이다. OTB 위성영상 처리 사용은 사용자 인터페이스를 활용하는 방법과 커맨트 라인 인터페이스를 사용하는 방법 두 가지가 존재한다. 이번 연구에서는 사용자 인터페이스는 웹 클라이언트에서 선택하기 때문에 실제 기능 처리에 대해서는 커맨드 라인 인터페이스를 활용하여 모든 기능을 수행하였다. 사용 방법은 소프트웨어 가이드(OTB Development Team, 2016)를 통해 확인할 수 있다. 모든 기능은 기본 필터링(Basic Filtering) 기능에 포함되어 있다. 엣지 추출을 수행할 때에는 단일 밴드만이 필요하다. 그렇기 때문에 입력 변수에 밴드 번호를 입력하는 인터페이스가 존재하며, 지도 영역 선택은 직접 폴리곤으로 생성하여 처리 영역을 선택하였다. 세 번째 처리 결과로는 영상을 데이터를 평활화한 결과이다[그림 4-23]. Mean Shift Segmentation 처리를 할 때는 기존 전국 데이터를 사용한 것이 아닌 이전 처리 결과로 GDAL Crop 기능으로 잘라진 서울시 데이터를 사용하였다. 이와 같이 처리된 데이터를 연결하여 바로 다시 두 번째 처리로 활용할 수도 있도록 설계하였다. Mean Shift Segmentation 알고리즘은 Spatial, Range, Threshold, Maxiter 총 네 개의 입력 인자를 받는다. 마지막 처리 결과로 그림 4-24는 K means 알고리즘을 수행한 결과이다. K means 알고리즘 또한 Training Set, Classes, Threshold, Maxiter 네 개 입력 인자를 받는다. K means는 Classes 개수에 따라 영상을 분류 한다. 그림 4-24는 결과에서는 총 5개로 분류한 결과이다. GeoServer에 저장되어 있는 서울시 구 데이터를 선택하여 전국 위성 데이터를 서울 지역에 대해서는 자른 후 처리를 수행하였다. 그림 4—25에서 볼 수 있듯이 처리된 영상 데이터를 중첩하여 시각화도 가능하다. 지도 위에 여러 데이터를 동시 시각화하고 웹 앱의 메뉴에 있는 레이어 리스트 항목을 통해 현재 시각화되고 있는 레이어 목록을 확인할 수 있다. 이를 드래그 앤 드롭으로 순서를 변경하여 레이어 순서를 변경할 수 있으며, 투명도 또한 조절할 수 있다.



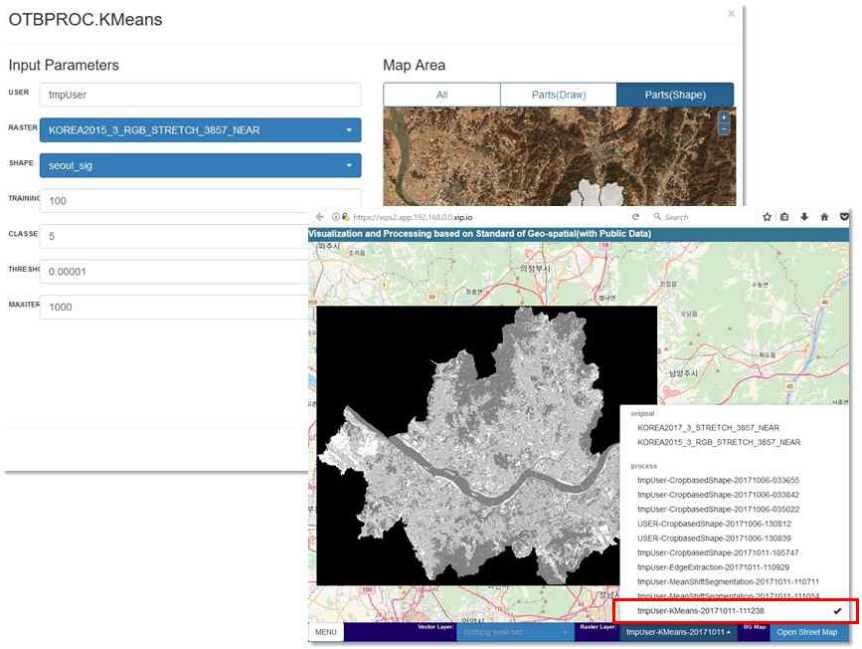
[그림 4-21] GDAL Crop 기능 처리 결과.



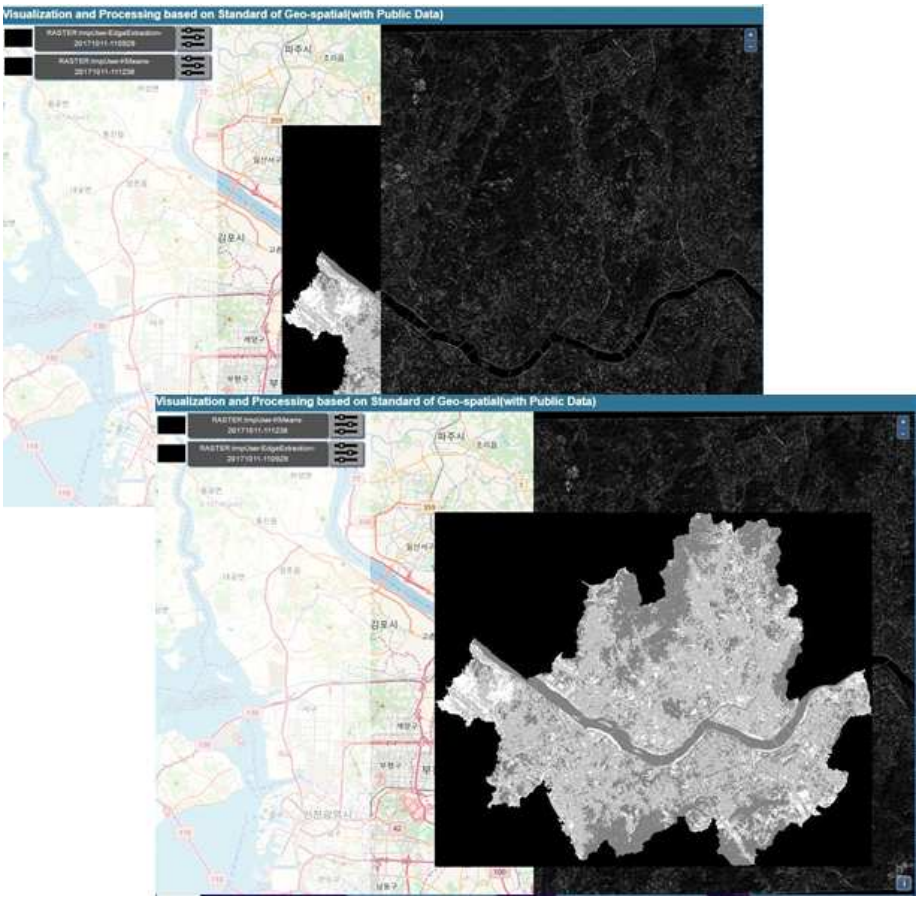
[그림 4-22] OTB Edge Extraction 기능 처리 결과.



[그림 4-23] OTB Mean Shift Segmentation 기능 처리 결과.



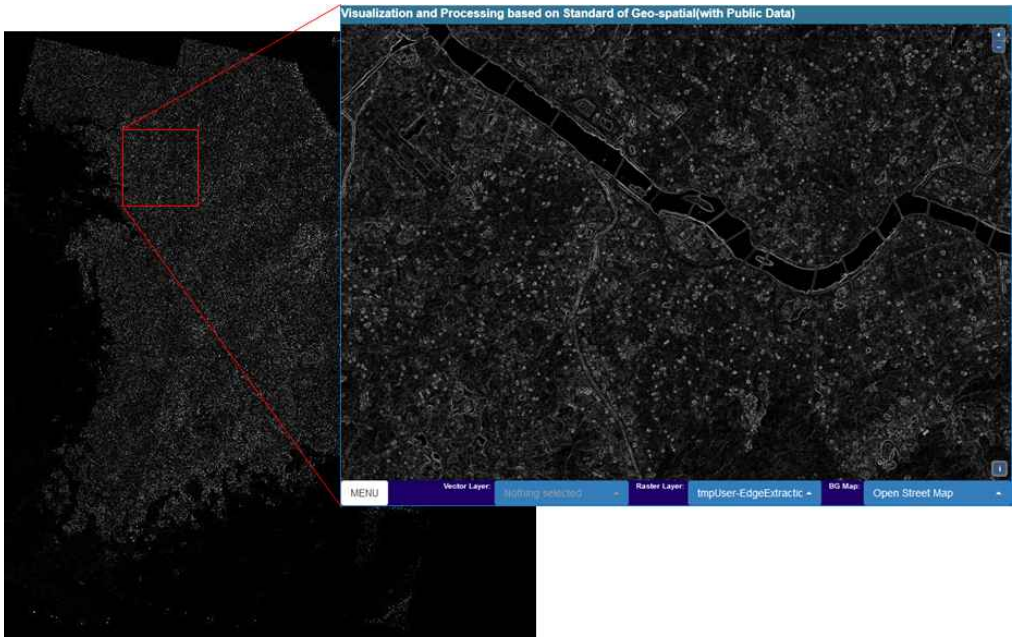
[그림 4-24] OTB K Means 기능 처리 결과.



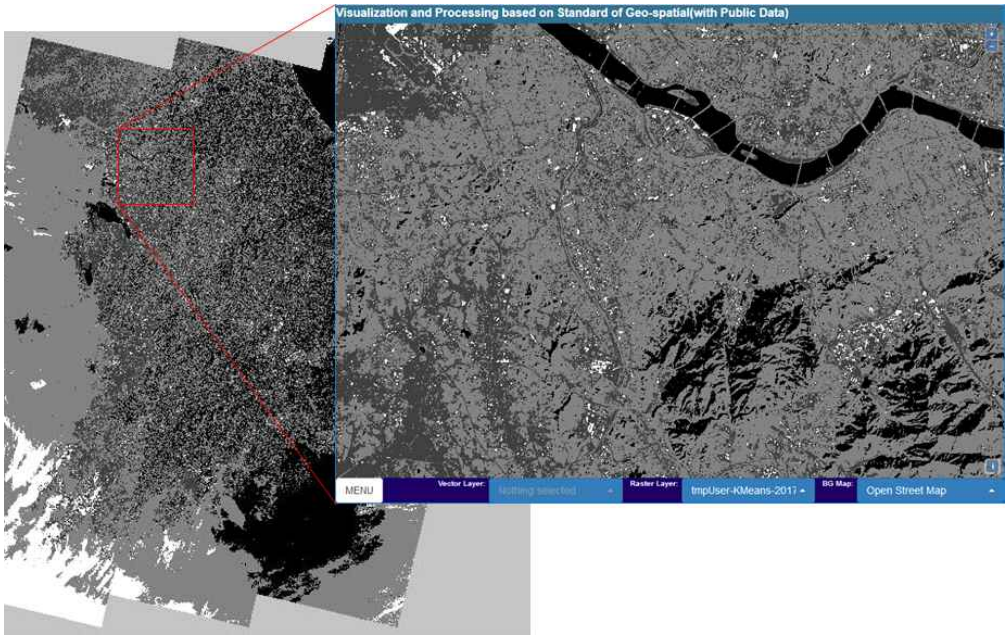
[그림 4-25] 다중 레이어 시각화 관리 결과.

총 세 종류에 대한 설계 및 구축된 공간정보 처리 서비스를 PaaS에 배포하여 기능에 대한 수행 여부 테스트를 수행하였다. 공간정보 및 공공 데이터 융합 시각화 앱의 경우 공간정보 서비스에 기본이 되는 기능을 포함하고 있으며, 위성영상 처리 서비스에 구축된 기능들은 클라이언트 기반 공간정보 처리 소프트웨어에서 일반적으로 제공되는 사항과 동일한 방식으로 처리되도록 설계되었다. 사용자가 단순히 활용하는 입장에서 고려해보았을 때 큰 차이가 나지 않을 것이다. PaaS로 배포가 되더라도 사용자는 별다른 변화를 느끼지 않으며 단순히 웹을 사용하는 수준에 그칠 것이다. 하지만 개발자나 운영자 관점에서 보았을 때는 상당히 많은 부분이 변경되고 관리되는 부분 또한 편리해진 것을 확인할 수 있다. 마지막 결과로 그림 4-26부터 그림 4-28은 국내 데이터 전체 기능별로 처리한 결과이다. Landsat 8, 2015년도 전국 데이터를 사용하였다. 이는 이번 웹 시스템에서 비동기식 처리를 제공하는 WPS 2.0을 활용하고 WebSocket을 사용하기 때문에 수행할 수 있는 결과이다. 영상 처리의 경우 지역 크기는 시간에 비례할 수밖에 없다. 대부분 소프트웨어의 경우 처리할 동안 컴퓨팅 리소스를 잡아먹기 때문에 다른 작업을 동시에 수행하기 어려운 상황이 될 수 있다. 클라우드 컴퓨팅 서비스는 컴퓨팅 자원이 외부 시스템에 의존되기 때문에 처리 후 자신의 컴퓨팅 자원이 낭비되지 않는다. 그러므로 다른 작업을 수행할 수 있는데 이때 WPS 2.0과 WebSocket 사용으로 반드시 웹 시스템에 접속해야 할 상황조차 없애준다. 처리가 시작되면 처리 서버의 웹 표준 인터페이스를 통해 비동기식 처리가 수행된다. 그러므로 동시다발적으로 처리를 받을 수 있고, WebSocket을 통해 처리가 완료되면 알아서 클라이언트에게 Push를 줄 수 있다. 만약 클라이언트가 접속이 되어 있지 않은 상황일지라도 접속을 하였을 때 실시간 상태를 확인할 수 있다. 물론 인스턴스에 대한 컴퓨팅 능력, 알고리즘 복잡도, 처리에 대한 개수에 따라 처리 시간은 상이할 수 있다. 그림 4-29는 처리를 수행한 알고리즘별 시작시간과 종료 시간을 보여주고 있다. Edge Extraction 처리를 한반도 전체 지역에 대해 처리한 결과로 총 시간은 약 5분 정도, K Means 처리 또한 5분 정도 시간이 수행되었다. Mean Shift Segmentation의 경우 가장 많은 시간이 소요된 것을 확인할

수 있다.

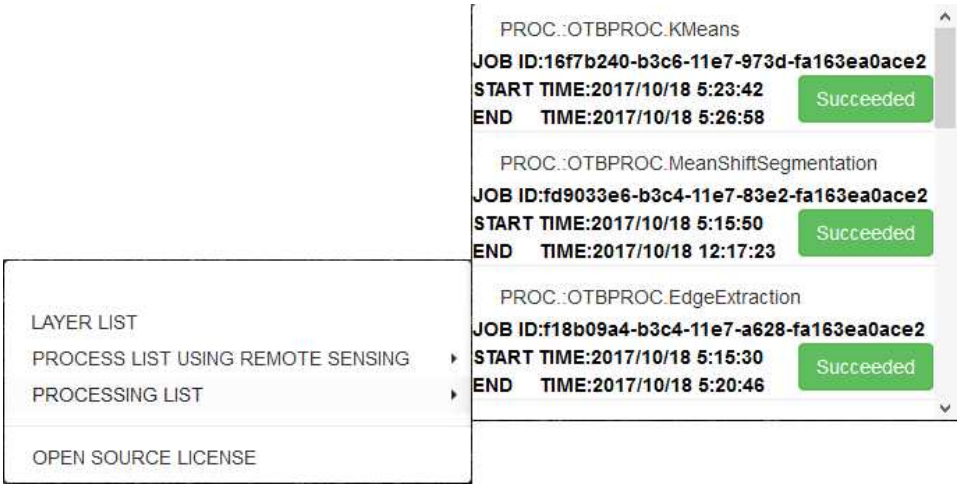


[그림 4-26] 국내 데이터 Edge Extraction 전체 처리 결과.



[그림 4-27] 국내 데이터 K Means 전체 처리 결과.

[그림 4-28] 국내 데이터 Mean Shift Segmentation 전체 처리 결과.



[그림 4-29] 한반도 전체 데이터 처리 결과 목록.

제 2 절 성능 테스트

이번 절에서는 이번 연구에서 PaaS에 배포된 웹 시스템 특정 요청에 대한 성능 테스트 결과를 서술하였다. 성능 테스트를 위해서 비교 대상이 되는 웹 시스템을 추가 설계 및 구축하였다. 테스트에 대한 배포 시스템 차이는 크게 두 가지를 고려하였다. 첫 번째, IaaS로 구축된 웹 서비스와 PaaS로 구축된 웹 서비스에 대한 성능 비교이다. 상대적으로 둘 중 하나의 서비스로 구축된 웹 서비스가 성능이 좋다는 비교를 위한 목적이 아닌 PaaS로 구동되었을 때 저하가 있는지에 대한 것을 확인하는데 중점을 두었다. IaaS와 PaaS는 사용자에게 제공되는 최종 결과는 동일하다. PaaS는 개발자 및 운영자에 편의를 고려한 서비스이므로 관리 측면에서 많은 부분이 개선되었다. 하지만 성능 저하가 생기거나 문제가 있으면 향후 활용하는 데 문제가 있을 수 있기 때문에 이번 절에서는 이러한 사항을 고려하였다. IaaS는 가상 인스턴스 기반이고 PaaS는 컨테이너 기반으로 시스템이 배포되기 때문에 사양을 동일하게 맞추기 어렵다. 최대한 동일한 사양을 갖는 다양한 환경에서 서비스를 구축하고, 네트워크 통신 기반의 스트레스를 주어 성능 저하 여부에 대한 테스트를 수행하고 이를 정리하였다. 두 번째 성능 테스트는 PaaS 내부 배포 시스템 간 테스트로 전자정부 표준프레임워크 기반과 비 전자정부 표준프레임워크 기반의 공간정보 웹 시스템에 대한 성능 테스트이다. 국내 공공기관에서는 전자정부 웹 시스템 개발 시 전자정부 표준프레임워크를 많이 활용하고 있다. 활용하는 이유 중 하나로는 공통 컴포넌트 활용이라고 볼 수 있다. 편리하게 활용할 수 있지만 사용자에게 배포 시 성능 저하가 있을 경우 향후 활용성에 문제가 있을 수 있다. 당연히 기능이 추가될수록 프로젝트 단위가 커지기 때문에 이에 대한 저하는 있을 수 있다. 이번 연구에서는 여러 공통 컴포넌트 중 가장 기본이 되는 통합 로그인 기능을 영상정보 처리 웹 시스템에 적용하여 성능 부하 테스트를 수행하였다. 이 또한 성능 테스트 시 처리에 대한 속도를 비교하는 것이 아닌 처리 요청에 대한 안정성 성능 테스트이다. 처리가 수행되는 서버는 별도의 인스턴스로 구축되어 있고 이는 실제 처리 요청 시 변경 대상이 아니기 때문에 이번 연구에서는 처리 속도 및 결과에 대해서는 모든

시스템에서 동일하다고 볼 수 있다.

테스트 수행에서 사용된 소프트웨어는 오픈소스로 많이 사용되고 있는 성능 테스트 소프트웨어인 jMeter를 활용하였다. jMeter를 사용할 때 적용되는 기본적인 내용으로 Thread, Ramp-Up Period, Loop Count를 조절하여 성능 테스트를 수행하였다. Thread는 동시 사용자 수를 말하며, Ramp-Up Period는 사용자가 해당 명령을 수행하는 시간 간격을 말한다. Loop Count는 테스트 시나리오를 몇 번 실행할 것인지에 대한 설정값이다. 총 세 가지에 대한 웹 시스템에 대해 성능 테스트 시나리오는 사용자가 처리를 위해 필요한 입력 값이 무엇인지 요청하는 WPS2.0 요청인 Describe Process와 실제 처리 요청을 하는 Execute 요청이다. 요청에 대한 Thread, Ramp-Up Period, Loop Count는 표 4-3과 같이 구성하였다. 세 가지 사항으로 나누기 위한 구분은 Threads의 개수이다. Ramp-Up과 Loop Count는 모두 동일하다. 이렇게 구분을 한 이유로는 동시 사용자가 증가할 때 시스템에 대한 요청 응답 변화를 확인하기 위해서이다. 최종으로 테스트 되는 사용자 수는 Thread와 Loop Count의 곱으로 결정되며 3000명, 5000명, 7000명으로 구분될 수 있다. 하지만 Ramp-Up을 설정하였기 때문에 동시 사용자수는 3600초(1시간) 안에서 적절하게 분배되어 처리가 된다. 그러므로 동시 사용자에 대한 값 또한 향후 테스트 결과에 포함된다.

[표 4-3] 성능 테스트 Thread Group 설정 값

Test Systems	Infrastructure, Platform(none e-govframework, e-govframework)		
Threads (number)	300	500	700
Ramp-Up Period(second)	3600		
Loop Count (number)	10		

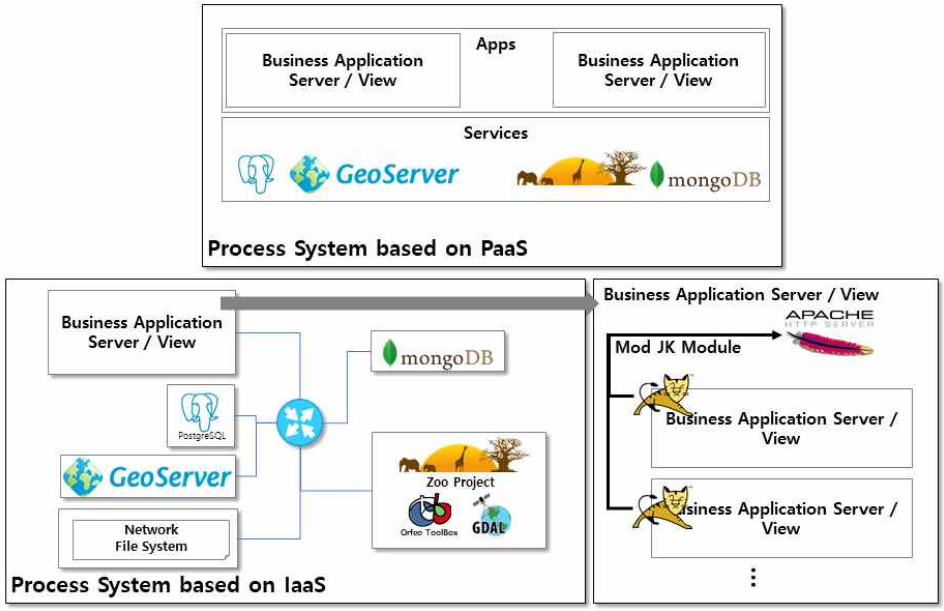
1) 클라우드 서비스 형식에 따른 공간정보 처리 시스템 성능 테스트

가) IaaS 기반 공간정보 처리 시스템 설계 및 구축

이번 연구에서 테스트 대상이 되는 웹 시스템은 구축된 시스템 중 공간정보 처리 앱인 위성영상 처리 웹 서비스이다. IaaS는 물리적 요소를 서비스형태로 지원해주는 서비스이다. 이번 연구에서는 PaaS를 위해 구축된 오픈스택에 가상 머신 하나를 추가로 생성하여 성능 테스트에 활용하였다. 비교 대상이 되는 PaaS의 컨테이너 및 IaaS의 인스턴스 사양은 표 4-4와 같다. IaaS로 바로 구동되는 위성영상 처리 웹 시스템 사양은 가상 머신 flavor 중에 m1.medium을 사용하여 인스턴스를 생성하였다. m1.medium은 vCPU 2개, 메모리 4GB, 디스크 용량 40GB로 설정되는 인스턴스이다. PaaS 앱 사양은 한 개의 컨테이너가 잡는 리소스로 메모리 1GB 및 저장소 1GB로 설정하였다. 컨테이너가 구동되는 가상 인스턴스 flavor은 m1.xlarge로 구성되어 있다. IaaS 환경과 동일하게 구동되기 위해 총 네 개의 컨테이너로 구성하여 4GB로 구동되는 앱을 구성하였다. PaaS와 IaaS에 대한 설계도를 간단히 그림 4-30에 표현하였다. 먼저, PaaS는 앞서 구축된 방식과 동일하다. IaaS로 설계된 공간정보 처리 시스템은 하나의 인스턴스 안에 자바 기반 웹 어플리케이션이 구동 가능하도록 환경을 구성한다. 제일 먼저 자바 런타임 환경을 설치한 후 웹 어플리케이션 서버(WAS: Web Application Server)인 톰캣을 설치한다. 웹 어플리케이션 서버에 대한 성능을 고려하여 서비스마다 독립적으로 서버를 구성할 수 있도록 설계되어 있다. 설치 후 웹 서버로 들어오는 요청을 위임할 수 있는 바이너리 프로토콜인 AJP(Apache Jserv Protocol) 통신으로 서로 다른 웹 어플리케이션 서버 포트를 묶어 웹 서버 포트인 80포트로 모두 요청할 수 있다는 장점이 있다. 처리를 위해 사용되는 미들웨어 소프트웨어는 GeoServer, PostgreSQL, Zoo Project를 사용하였다. Zoo Project의 경우 처리에 대한 효율을 위해 총 세 개의 인스턴스를 HAProxy를 통해 로드 밸런싱 되도록 구성하였다. 성능 테스트 수행은 순차적으로 진행되기 때문에 미들웨어 소프트웨어는 서로 공유하도록 구성하였다.

[표 4-4] 성능 테스트를 위한 IaaS 및 PaaS 웹 시스템 사양

VM / Container	Index	IP	Persistent disk	VM Type or RAM
WPS Container App (In PCF)				
WPS2 Container	4	-	1GB	1GB
WPS Instance				
WPS2 Instance	0	172.16.0.5	80GB	m1.medium
Shared Instances / Services				
GeoServer (PostgreSQL)	0	172.16.0.11	100GB(NFS)	m1.medium
MongoDB	0	172.16.0.8	None	m1.medium
HAProxy for ZooProject	0	172.16.0.15	None	m1.small
ZooProject	0	172.16.0.3	100GB(NFS)	m1.large
	1	172.16.0.14	100GB(NFS)	m1.large
	2	172.16.0.16	100GB(NFS)	m1.large



[그림 4-30] 성능 테스트를 위한 IaaS 공간정보 처리 시스템 설계.

나) 성능 테스트 결과

성능 테스트 수행 결과는 jMeter Listener 중 Response Latencies Over Time, Times vs Threads, Summary Report를 참고하여 정리하였다. Latencies는 응답 지연율 시간으로 HTTP 요청을 수행하였을 때 응답을 받을 때까지 시간을 측정한 것으로 서버에 대한 능력을 쉽고 빠르게 판단할 수 있는 내용 중 하나이다. Times vs Threads는 시간에 따른 동시 사용자 접속 수에 대한 개수를 결과로 나타내 주는 Listener이다. 테스트 측정 시 사용자를 300, 500, 700으로 구분하여 측정하였다. 사용자 요청이 한꺼번에 이루어지는 것이 아닌 Ramp-Up 수치에 따라 간격이 생기게 된다. 그리고 동시 사용자 수도 성능에 큰 영향을 미치기 때문에 이번 연구에서는 이를 측정하였다. 마지막으로 Summary Report에는 모든 결과에 대한 최소, 최대, 평균값과 처리 응답에 대한 오류율을 계산하여 보여주는 Listener이다. 표 4-5와 표 4-6은 IaaS와 PaaS 각 환경에서 Threads마다 성능 테스트한 결과를 표로 나타낸 것이다. 정리한 결과 내용으로는 응답 지연율(Latencies over time) 평균 값, 에러율(Error), 최대 동시 사용자 접속 수(Max active threads)이다. 수행 요청은 WPS 2.0 요청 중 처리 기능에 대한 세부 입력 값을 응답하는 Describe Process와 실제 처리를 요청하는 Execute이다. 전체적으로 바로 처리가 가능한 Describe Process 경우 Threads 변경에 따른 응답 지연율은 크게 다르지 않는 것을 확인할 수 있었다. 하지만 Execute 응답 지연율에 대한 평균 결과는 Threads마다 큰 차이를 보이고 있다. 최대 동시 사용자 또한 300과 500 Threads는 큰 차이가 나지 않지만 700 Thread일 경우 큰 차이를 보이고 있다. 좀 더 나은 비교를 위해 그림 4-31과 같이 IaaS와 PaaS 지연 응답률과 에러율 결과를 그래프로 나타낸 것이다. 그래프를 보면 알 수 있듯이 Describe Process와 같이 간단한 응답인 경우 전반적으로 IaaS 환경이 빠른 것을 확인할 수 있었다. 하지만 리소스가 필요한 Execute에 대한 지연 응답률은 PaaS 환경이 더 빠르게 나타나는 것을 확인할 수 있다. 이 외에 오류율과 최대 동시 사용자의 경우에도 PaaS 환경이 더 나은 결과를 보여주고 있다. 오류율 같은 경우 공간정보 처리에 대한 결과가 아닌 공간정보 처리 요청

에 대한 성공 여부이다. IaaS 환경의 경우 두 요청 모두 Threads가 높아질수록 처리에 대한 오류율도 높아지는 것을 확인할 수 있었다. 하지만 PaaS 환경은 오류율에 대한 큰 변화가 없는 것을 확인하였다. 최대 동시 사용자는 두 환경 모두 증가되긴 하지만 PaaS 환경에서 좀 더 낮은 증가하는 것을 확인할 수 있다. 동일 Threads에 대해 더 낮은 최대 사용자는 그만큼 서버가 요청에 대한 처리를 빠르게 할 수 있다는 말과 동일하다고 볼 수 있다. 최고 Threads 수인 700에서 제일 큰 차이가 나타나는 것으로 볼 때 시스템이 허용하는 범위 내 많은 사용자 접속 시 효율적인 것을 볼 수 있다. 이러한 결과는 PaaS 환경에서 배포될 때 분배되는 컨테이너 인스턴스 환경이 되므로 좀 더 나은 환경으로 시스템이 제공될 수 있는 것으로 예상할 수 있다.

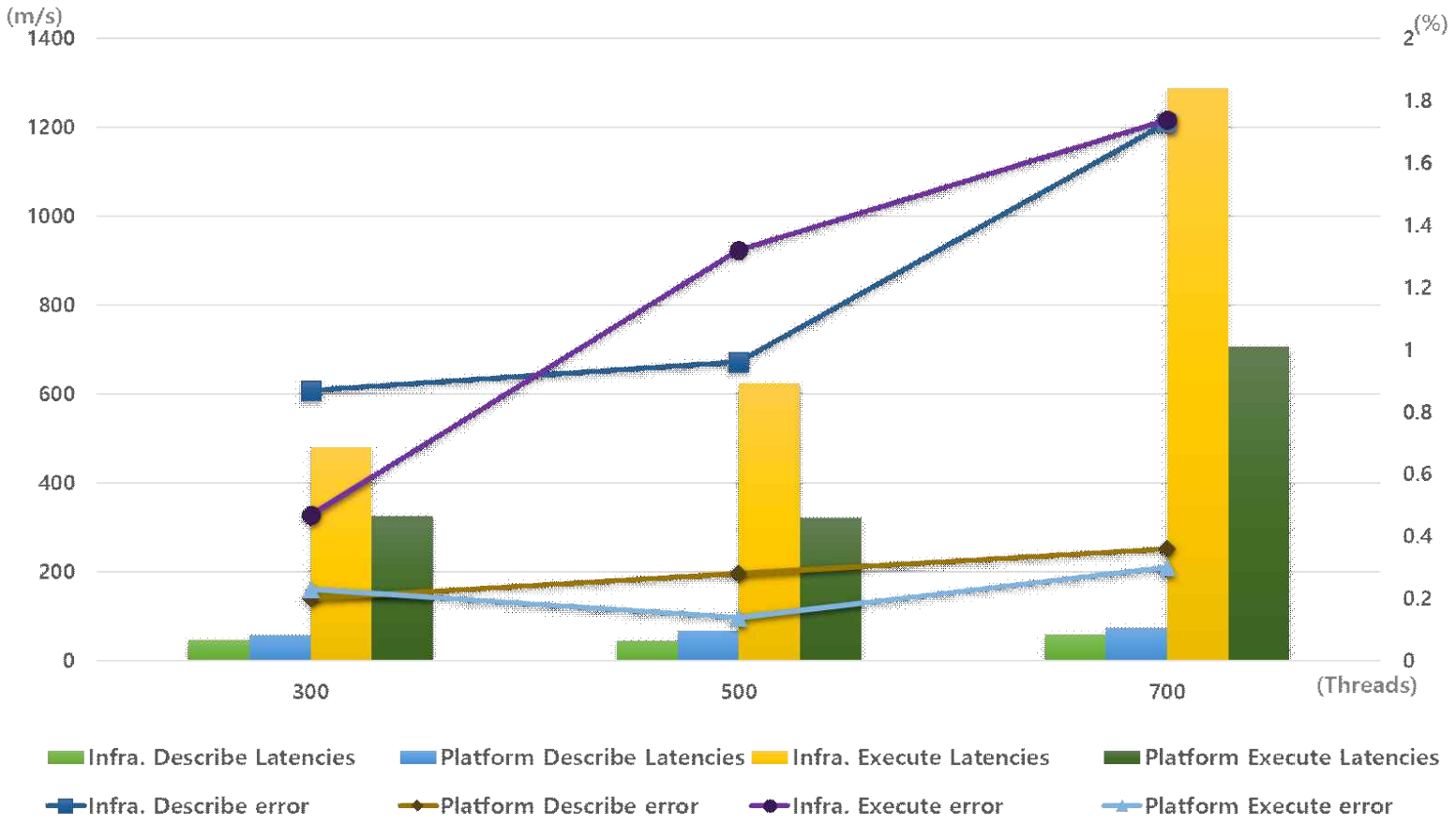
[표 4-5] IaaS 환경 성능 테스트 동시 사용자, 응답 시간 평균 결과

Threads(Infra.)		Describe Process	Execute
300	Latencies over time(ms)	48.91	481.88
	Error (%)	0.87	0.47
	Max active threads(number)	3	
500	Latencies over time(ms)	46.62	624.70
	Error (%)	0.96	1.32
	Max active threads(number)	5	
700	Latencies over time(ms)	59.67	1289.46
	Error (%)	1.73	1.74
	Max active threads(number)	11	

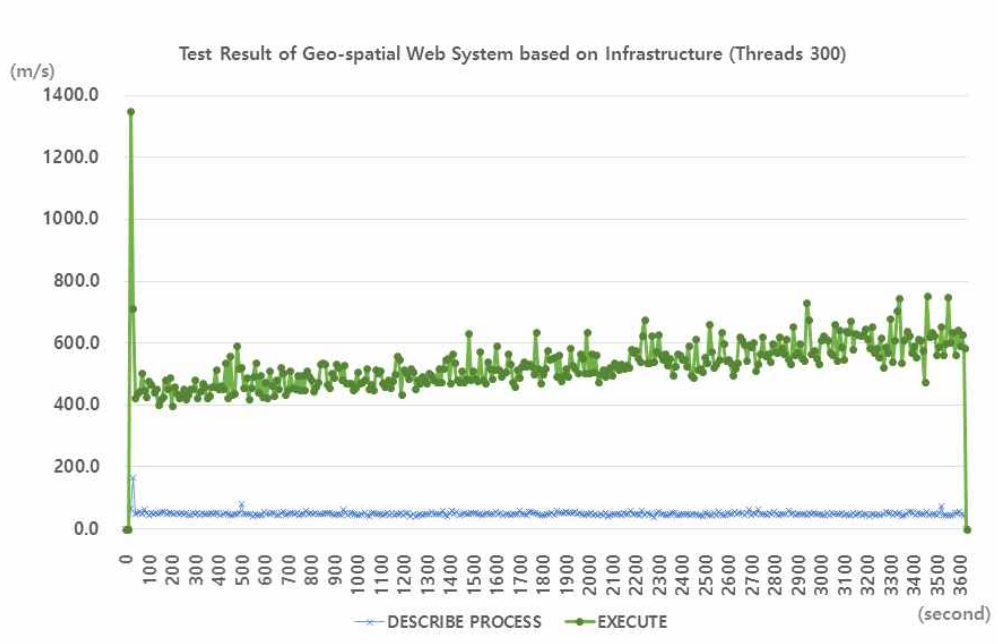
[표 4-6] PaaS 환경 성능 테스트 동시 사용자, 응답 시간 평균 결과

Threads(Platform)		Describe Process	Execute
300	Latencies over time(ms)	58.24	326.08
	Error (%)	0.20	0.23
	Max active threads(number)	3	
500	Latencies over time(ms)	67.81	322.24
	Error (%)	0.48	0.30
	Max active threads(number)	6	
700	Latencies over time(ms)	73.81	707.25
	Error (%)	0.36	0.30
	Max active threads(number)	7	

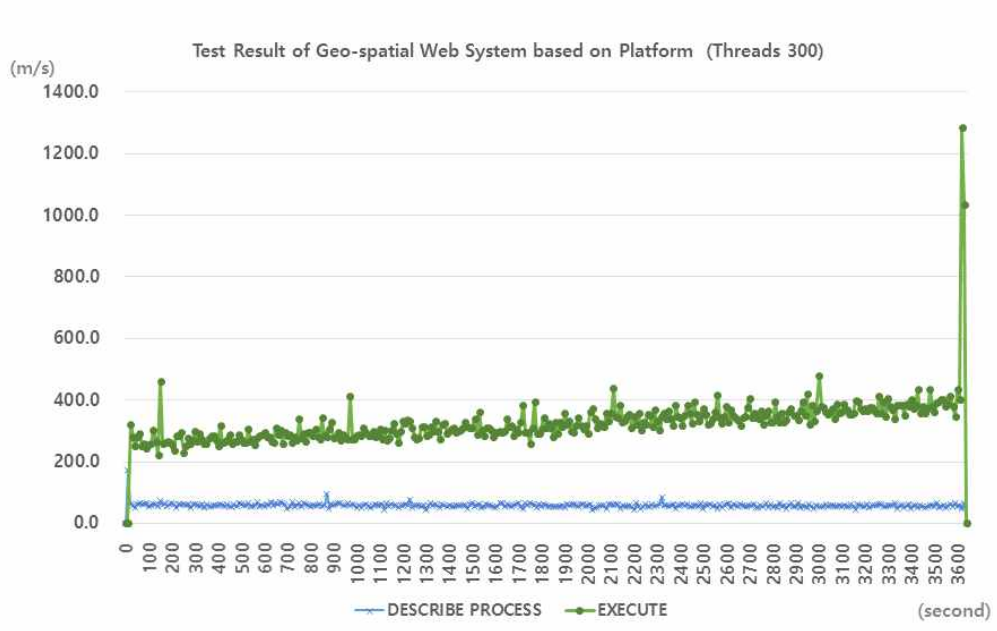
그림 4-32부터 그림 4-37은 성능 테스트에 대한 지연 응답률에 대한 결과를 100초 단위로 측정한 결과를 그래프로 나타낸 것이다. 이 또한 Threads가 300, 500, 700 결과에 대해 각각 나타내었다. Y축은 m/s 단위 응답 시간이며, x축은 시간을 나타낸다. 시간에 대한 주축은 100초, 응답 시간에 대한 주축은 200(m/s) 단위로 구성되어 있다. 1시간(3600초)에 모든 처리가 끝난 것을 확인할 수 있다. 파란색 선이 Describe Process이고, 초록색 선이 Execute 결과이다. Describe Process는 증가율 없이 응답 시간이 거의 동일한 것을 확인할 수 있었고, Execute는 두 시스템 모두 시간이 지나감에 따라 응답 시간이 증가하는 것을 확인할 수 있다. 300의 경우 IaaS와 PaaS 모두 비슷한 증가율을 보이고 있다. 하지만 500부터는 IaaS 환경에서 좀 더 급격한 증가를 확인할 수 있다. 특히, 700인 경우 PaaS 환경은 300과 500과 거의 동일하게 증가율을 보이고 있지만, IaaS 환경은 급격하게 올라가는 것을 확인할 수 있다. 최종적으로 모든 그래프에서 확인할 수 있듯이 PaaS 환경에서 좀 더 사용자 증가에 대한 대처가 좋을 것을 확인할 수 있다.



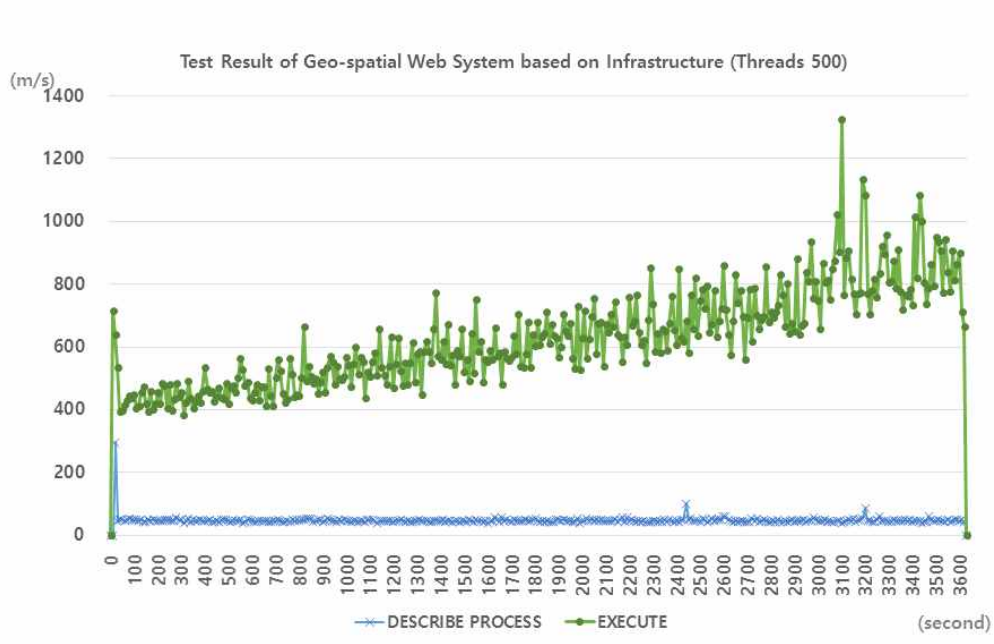
[그림 4-31] IaaS 및 PaaS 환경 Edge Extraction 기능 Describe Process, Execute 요청 성능 비교 평균 결과 차트.



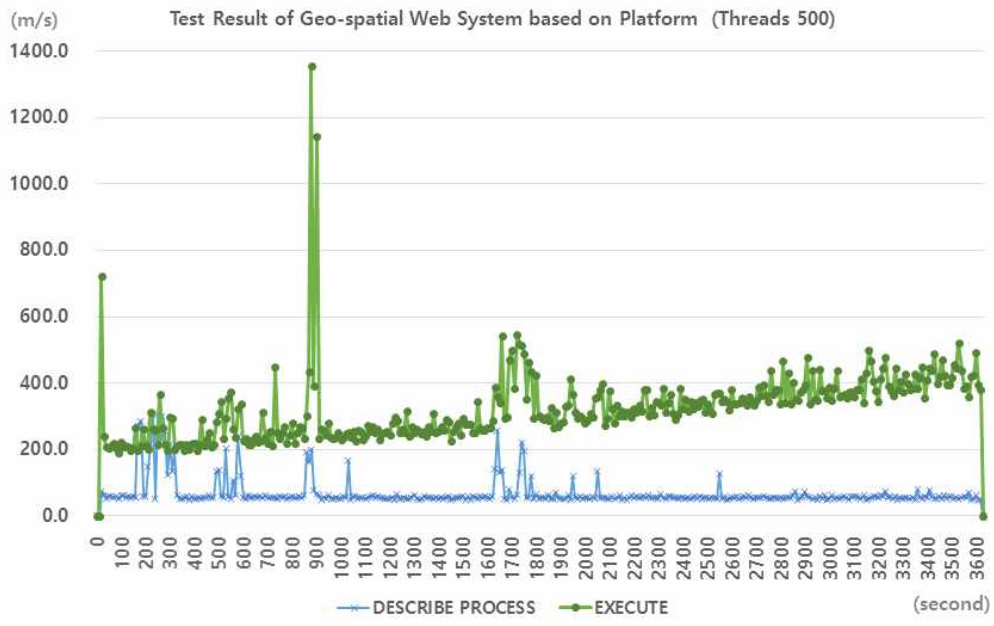
[그림 4-32] IaaS 환경 성능 테스트 결과(Threads 300).



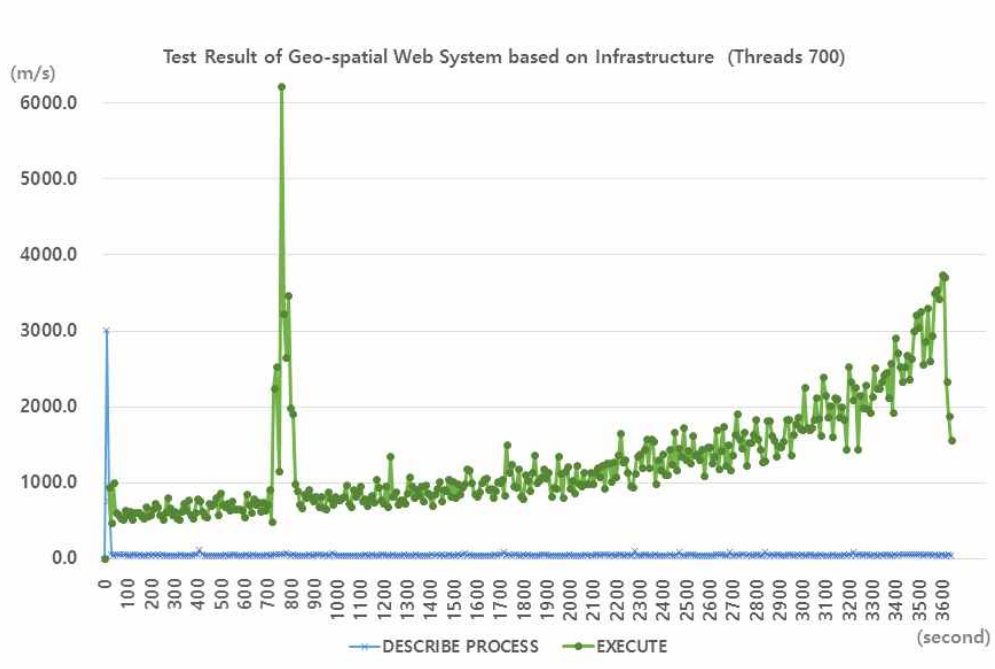
[그림 4-33] PaaS 환경 성능 테스트 결과(Threads 300).



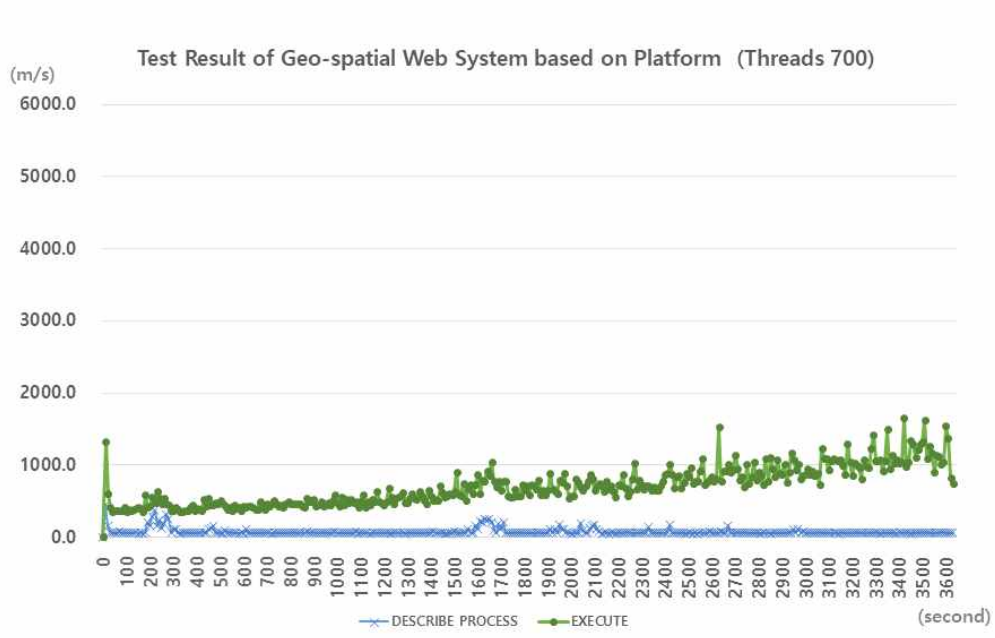
[그림 4-34] IaaS 환경 성능 테스트 결과(Threads 500).



[그림 4-35] PaaS 환경 성능 테스트 결과(Threads 500).



[그림 4-36] IaaS 환경 성능 테스트 결과(Threads 700).



[그림 4-37] PaaS 환경 성능 테스트 결과(Threads 700).

2) 표준프레임워크 여부에 따른 시스템 성능 테스트

가) 표준프레임워크 기반 공간정보 처리 시스템 설계 및 구축

이번 성능 테스트는 동일한 PaaS 앱 컨테이너에서 국내 전자정부 표준프레임워크 적용 여부에 대한 성능 테스트이다. 동일 언어 기반으로 구성되어 있지만 전자정부 표준프레임워크를 기반으로 공통적인 기능에 대해 빠르게 적용이 가능하다. 성능 테스트를 위해 위성영상 처리 웹 시스템을 전자정부 표준프레임워크 기반 웹 시스템으로 추가 구축하고 공통 컴포넌트 중 로그인 기능을 적용하여 테스트를 수행하였다. 성능 테스트를 위한 컨테이너 및 서비스 사양은 표 4-7과 같다. 두 웹 시스템 모두 PaaS 기반의 컨테이너 형식으로 구동이 되고 설정되는 메모리 또한 동일하게 구성하였다. 공유되는 서비스는 앞서 측정을 위해 사용된 서비스 사양과 동일하게 구성되어 있다. 그림 4-38은 성능 테스트를 위한 두 시스템에 대한 간단한 구성도이다. 그림에서도 확인할 수 있듯이 외관상으로 보았을 때에는 크게 다르지 않다. 하지만 스프링 프레임워크를 사용하는 것이 아닌 표준프레임워크를 활용하였고, Apps 내부에 표준프레임워크에서 제공하는 여러 기능을 바로 사용할 수 있도록 구성되므로 내부적으로는 많은 부분이 변경되어 있다. 표준프레임워크 기반으로 설계된 웹 시스템을 PaaS에서 구동하기 위해서는 표준프레임워크를 컴파일할 수 있는 빌드팩이 필요하다. 두 시스템 모두 자바 기반 톰캣 컨테이너로 구동되지만 표준프레임워크에 대한 특성이 포함되어 있어 별도 컴파일 환경이 추가로 필요하다. 이 빌드팩은 전자정부 표준프레임워크를 지원하고 있는 PaaS인 파스타(PaaS-TA)에서 모듈로 제공하고 있으며, 표준프레임워크 3.5 버전까지 지원하고 있다. 그렇기 때문에 위성영상 처리 웹 시스템을 전자정부 표준프레임워크 3.5 버전으로 구축하고, 그림 4-39와 같이 이번 연구에서 구축된 PaaS에 전자정부 표준프레임워크 3.5 빌드팩을 추가하였다. 이때 전자정부 표준프레임워크 빌드팩은 자바 빌드팩과 동일한 구성을 하고 있기 때문에 자바 빌드팩보다 우선순위가 낮아한다. 그렇지 않으면 자바 기반 시스템이 배

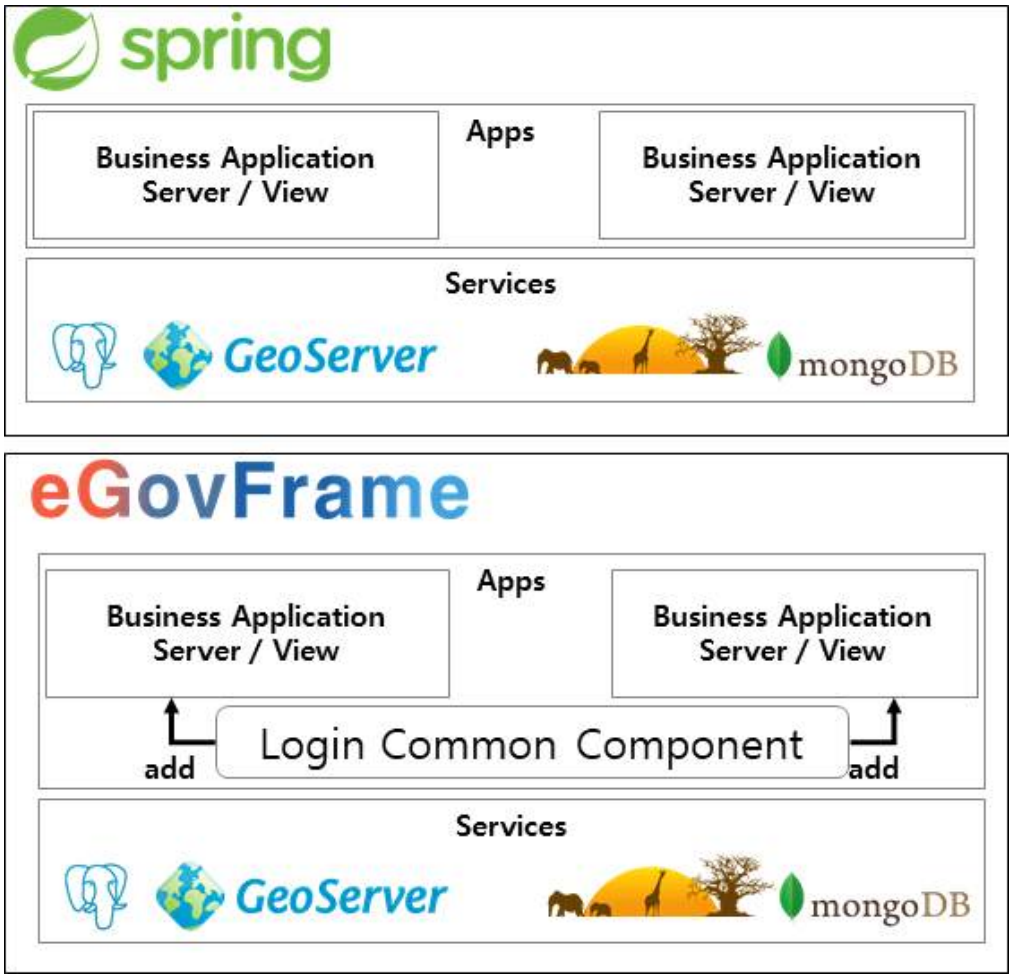
포될 때 표준프레임워크 빌드팩을 이용하기 때문에 문제가될 수 있다. 그림 4-40은 이번 연구에서 구축된 피보탈 클라우드 파운더리 PaaS에 전자정부 표준프레임워크 기반의 위성영상 처리 웹 시스템을 배포한 결과이다. 결과에서 볼 수 있듯이 배포 시 전자정부 표준프레임워크 빌드팩인 egov_buildpack_v35를 사용한 것을 확인할 수 있다.

[표 4-7] 표준프레임워크 성능 테스트를 위한 PaaS 사양

VM / Container	Index	Persistent disk	RAM
WPS Container App (In PCF)			
WPS2 Container	4	1GB	1GB
eGov WPS2 Container	4	1GB	1GB
Shared Services			
GeoServer	0	100GB(NFS)	-
MongoDB	0	None	-
HAProxy for ZooProject	0	None	-
ZooProject	0	100GB(NFS)	-
	1	100GB(NFS)	-
	2	100GB(NFS)	-

그림 4-41과 그림 4-42은 성능 테스트 수행 전 구동 여부 확인을 위한 전자정부 표준프레임워크 기반 위성영상 처리 웹 시스템에 대한 기능 수행 결과이다. 로그인 기능을 추가하여 위성영상 처리 시스템 접근하려면 그림 4-41에서 볼 수 있듯이 로그인이 필요하다. 로그인에 필요한 사용자 인터페이스는 데모로 제공해주는 로그인 페이지를 활용하였다. 로그인이 완료되면 처리가 수행되는데 이때 사용자에 따른 처리 결과를 확인할 수 있도록 구축하였다. 그림 4-42 결과처럼 표준프레임워크 로그인 시 사용된 사용자 아이

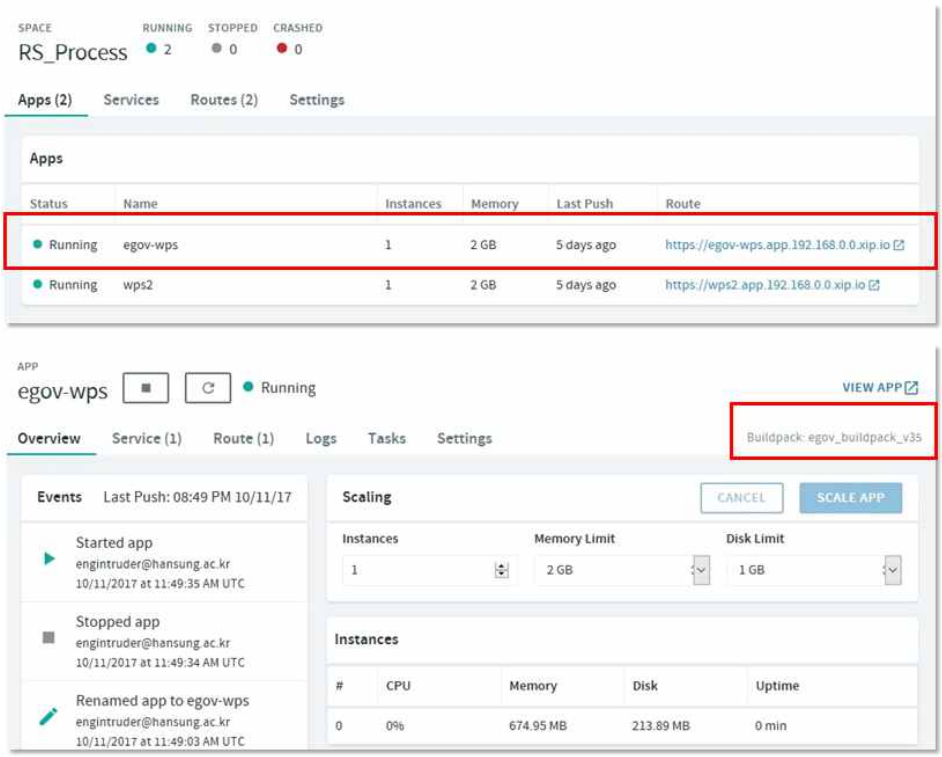
디 USER를 통해 접근하면 처리에 대한 결과를 불러올 때 USER 사용자가 처리한 결과만을 불러온다. 이러한 기능은 기본적으로 영상처리 시스템 설계 시 고려되었던 사항이었지만 로그인 기능이 없어 활성화되지 않았다. 표준프레임워크 공통 컴포넌트 통합로그인이 추가됨으로써 사용자별 처리 결과 기능이 자동으로 활성화된 것을 확인할 수 있다.



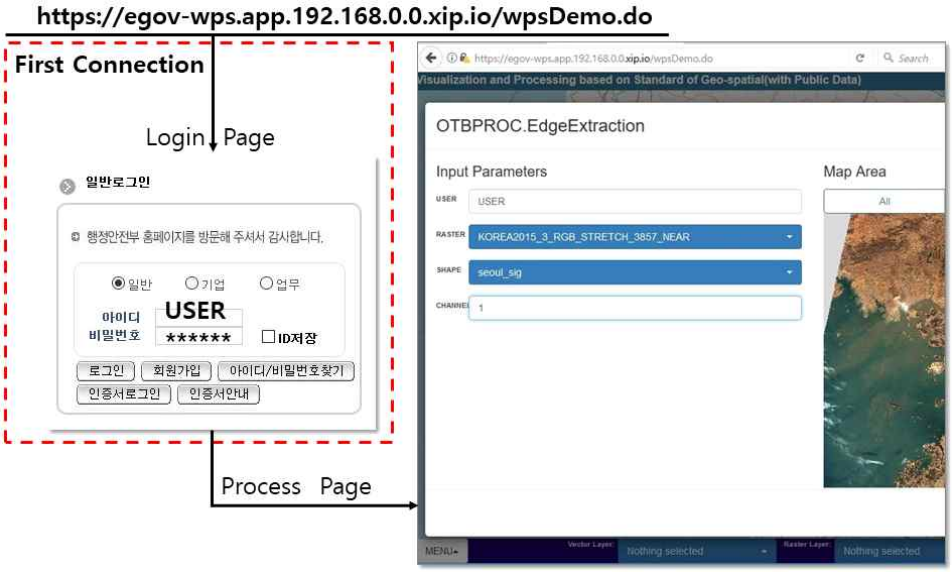
[그림 4-38] 성능 테스트를 위한 표준프레임워크 공간 표준 처리 시스템 설계.

buildpack	position	enabled	locked	filename
staticfile_buildpack	1	true	false	staticfile_buildpack-cached-v1.4.11.zip
java_buildpack_offline	2	true	false	java-buildpack-offline-v3.18.zip
ruby_buildpack	3	true	false	ruby_buildpack-cached-v1.6.44.zip
nodejs_buildpack	4	true	false	nodejs_buildpack-cached-v1.6.3.zip
go_buildpack	5	true	false	go_buildpack-cached-v1.8.5.zip
python_buildpack	5	true	false	python_buildpack-cached-v1.5.20.zip
php_buildpack	6	true	false	php_buildpack-cached-v4.3.38.zip
dotnet_core_buildpack	7	true	false	dotnet-core_buildpack-cached-v1.0.22.zip
binary_buildpack	8	true	false	binary_buildpack-cached-v1.0.13.zip
egov_buildpack_v35	9	true	false	egov-buildpack-offline-egov3.5.zip

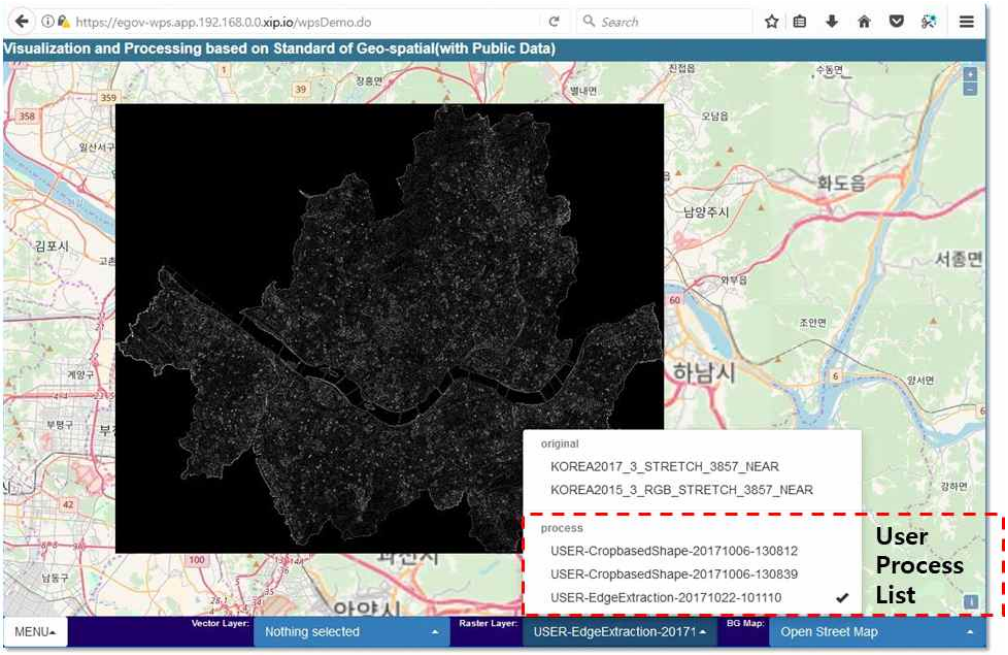
[그림 4-39] 전자정부 표준프레임워크 PaaS를 위한 빌드팩 업로드 결과.



[그림 4-40] 표준프레임워크 성능 테스트를 위한 PaaS 배포 결과.



[그림 4-41] 위성영상 처리 웹 시스템 전자정부 표준프레임워크 공통 컴포넌트 적용 결과(사용자 통합로그인).



[그림 4-42] 공통 컴포넌트 추가에 따른 처리 결과 목록 사용자 별 구분 기능.

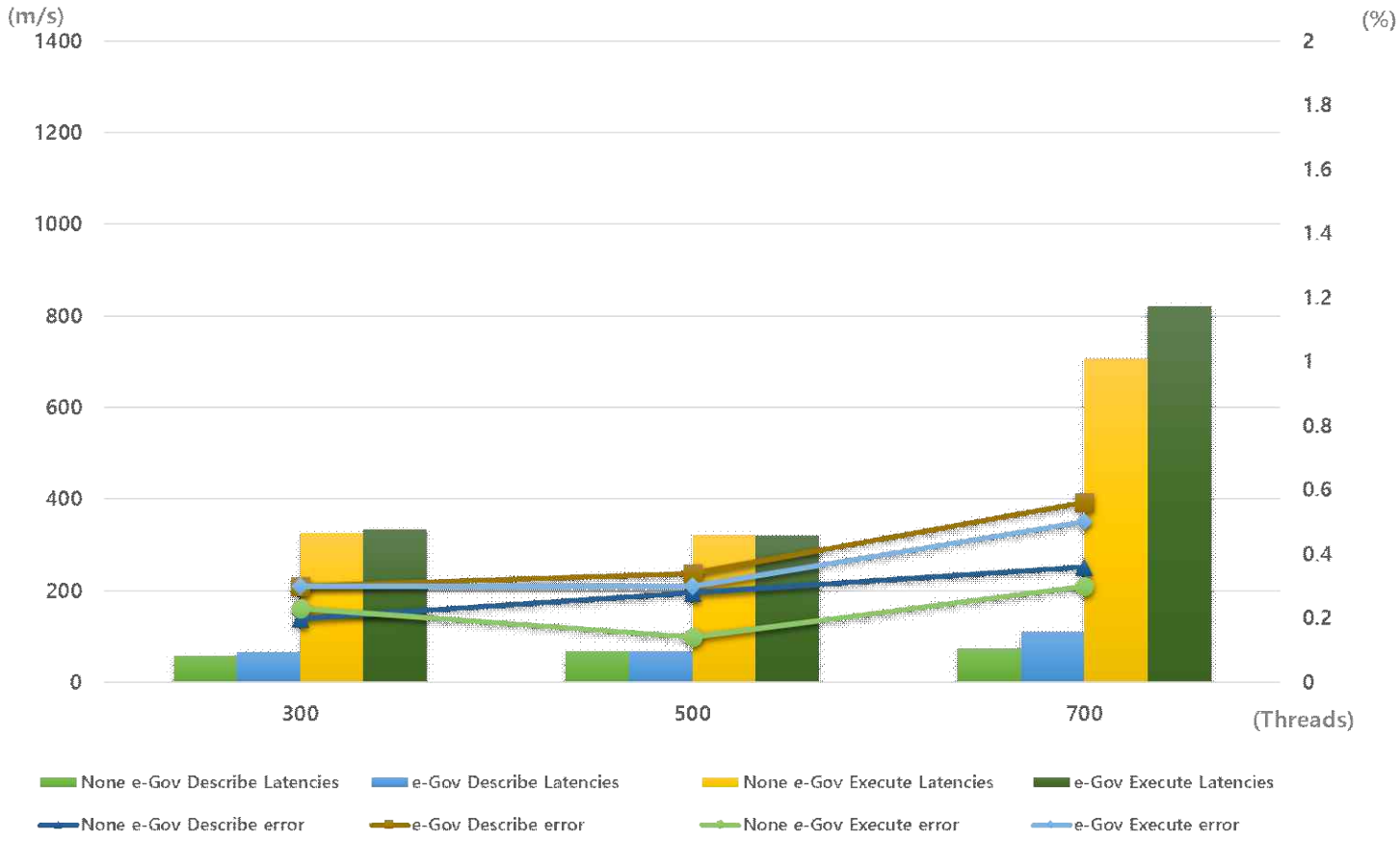
나) 성능 테스트 결과

표 4-8은 전자정부 표준프레임워크를 기반으로 구성된 영상정보 처리 시스템 성능 테스트 수행에 대한 결과를 나타낸다. 전자정부 표준프레임워크 기반으로 구축한 시스템에는 통합 로그인 공통컴포넌트를 추가했기 때문에 로그인 수행이 선행되어야 처리가 가능하다. 그렇기 때문에 성능 테스트에서 로그인에 대한 요청이 선행되어야 한다. 비교 대상 되는 비 전자정부 표준프레임워크 PaaS에 구축된 시스템은 IaaS 환경에 구축된 시스템과 비교한 성능 테스트 결과와 동일한 사양이기 때문에 이를 사용하였다. 표준프레임워크가 추가 되더라도 응답 평균 시간 에러율, 최대 동시 사용자를 확인해보았을 때 이전과 비슷한 상승세를 보이고 있는 것을 확인할 수 있다. 하지만 700 Threads Describe Process와 Execute 결과에 대해서는 상대적으로 500 Threads와 비교해 보았을 때 급격하게 증가 되는 것을 확인할 수 있었다. 비 표준프레임워크와 비교를 위해 그림 4-43에서 보이는 것과 같이 그래프로 표현해보았다. 응답 시간을 보았을 때 300 Threads 일 경우 모든 처리에 대해 비슷한 결과를 보이고 있지만 Threads가 증가될수록 그 격차가 벌어지는 것을 확인할 수 있다. 에러율을 비교하여 보았을 때 표준프레임워크에 대한 에러율이 높긴 하지만 비슷한 값 및 증가를 보이고 있는 것을 확인할 수 있다. 응답 시간의 경우 로그인 기능에 대한 예제 로직이 추가됨에 따라 그만큼 리소스 양이 많아졌기 때문에 증가된 것으로 예상되며 이는 개발자가 예제를 바로 사용하는 것이 아닌 적절한 최적화를 통해 충분히 좀 더 나은 결과를 낼 수 있다.

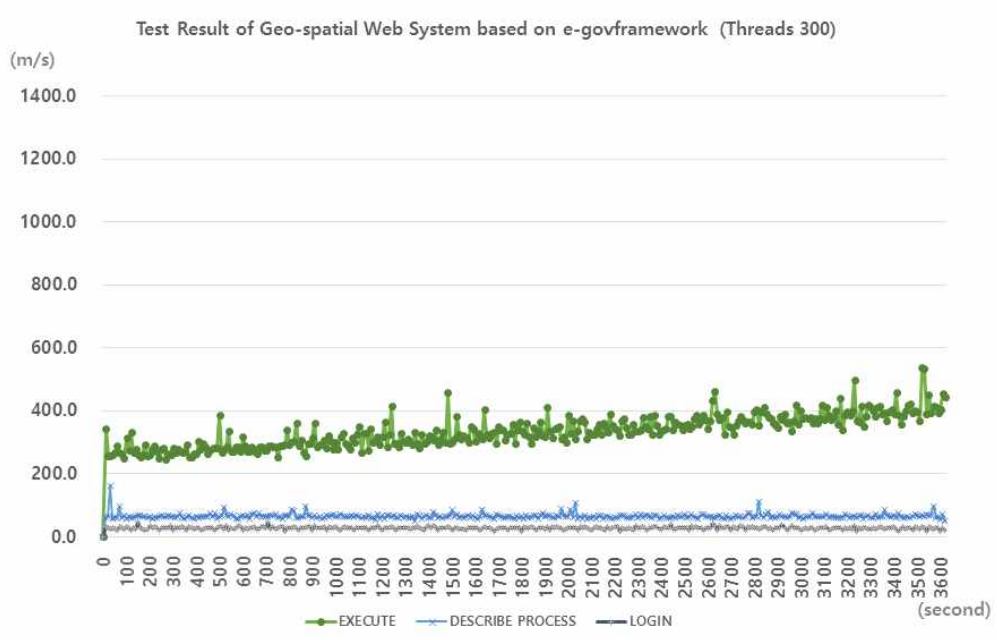
그림 4-44부터 그림 4-46은 PaaS 환경에서 전자정부 표준프레임워크 기반으로 구성된 영상처리 비즈니스 어플리케이션에 대해 Threads가 300, 500, 700개로 성능 테스트를 수행하였을 때 전체 응답 결과를 그래프로 표현한 것이다. 표준프레임워크와 비 표준프레임워크에 대한 전체적인 성능에 대한 결과는 큰 차이가 없는 것으로 확인된다. 서비스 개발 시 필요한 기능이 표준프레임워크 공통 컴포넌트 기능으로 제공되고 있다면 이를 활용하는 것이 서비스 제공에 많은 도움을 줄 수 있다.

[표 4-8] 표준프레임워크 기반 성능 테스트 동시 사용자, 응답 시간 평균 결과

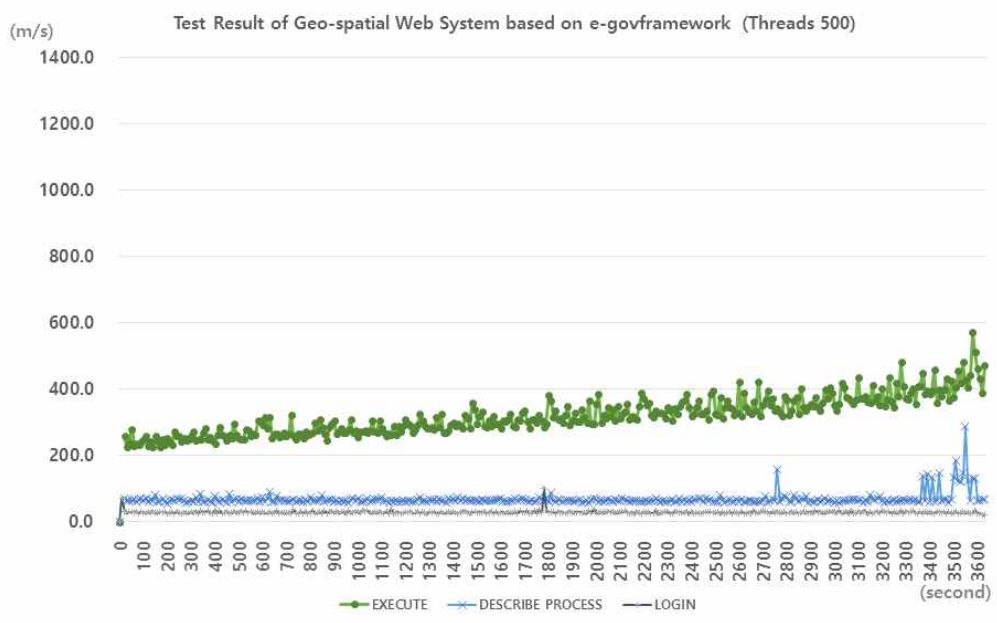
Threads		Login	Describe Process	Execute
300	Latencies over time(ms)	27.90	65.70	335.20
	Error (%)	0.00	0.30	0.30
	Max active threads(number)	3		
500	Latencies over time(ms)	28.8	68.2	319.9
	Error (%)	0.00	0.3	0.3
	Max active threads(number)	4		
700	Latencies over time(ms)	28.0	111.5	822.0
	Error (%)	0.00	0.6	0.5
	Max active threads(number)	8		



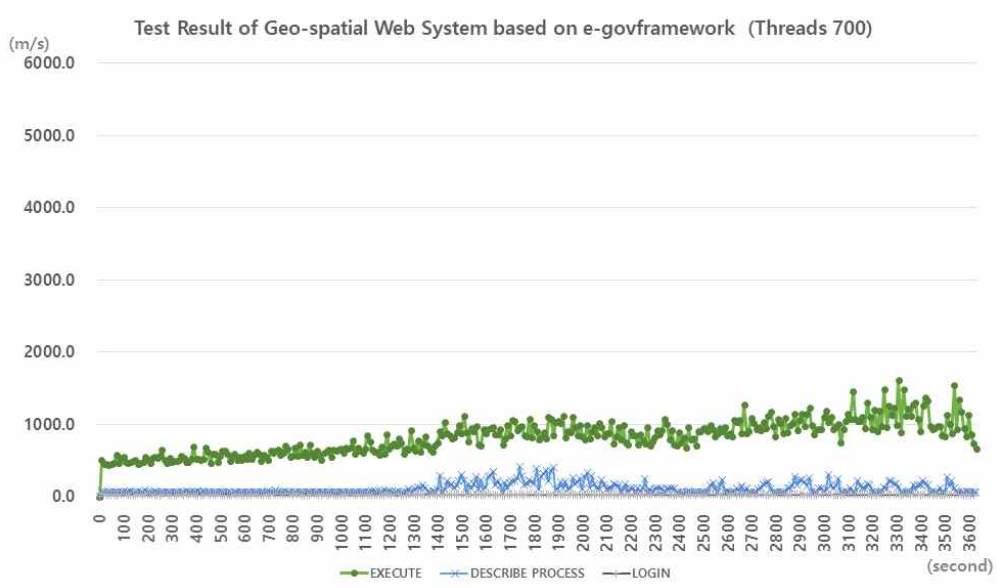
[그림 4-43] 표준프레임워크 및 비 표준프레임워크 환경 Edge Extraction 기능 Describe Process, Execute 요청 성능 비교 평균 결과 차트.



[그림 4-44] 표준프레임워크 기반 시스템 성능 테스트 결과(Threads 300).



[그림 4-45] 표준프레임워크 기반 시스템 성능 테스트 결과(Threads 500).



[그림 4-46] 표준프레임워크 기반 시스템 성능 테스트 결과(Threads 700).

제 3 절 연구 성과 활용 가능성

이번 연구에서 실제 구축된 시스템은 개발자가 개발한 공간정보 관련 웹 시스템을 배포할 수 있는 PaaS 시스템과 구축한 PaaS에 배포한 세 가지 공간정보 처리 웹 시스템이다. PaaS 클라우드 컴퓨팅 구축을 위해 사용된 오픈스택과 클라우드 파운더리는 실제 산업계에서도 많이 활용되고 있는 오픈소스 기술이다. 특히, PaaS 플랫폼인 클라우드 파운더리는 국내 연구 사업으로 진행되고 있는 파스타(PaaS-TA) 기반 기술로도 사용되고 있다. 그렇기 때문에 파스타를 사용하더라도 연구에서 제안된 시스템 설계를 적용하여 공간정보 관련된 플랫폼 서비스를 제공할 수 있다. 또한, 연구에서 제안한 PaaS 구축에 필요한 최소 사양인 물리적 하드웨어를 보유하고 있는 기관 또는 민간기업에서는 이번 연구에서 구축한 내용을 토대로 구축하여 PaaS 시스템을 제공할 수 있다. 추가로 제안한 공간정보 관련 서비스를 위해 추가한 미들웨어 서비스를 공간정보 관련 기관에서 PaaS 형태로 제공한다면 국내 금융 플랫폼 서비스인 K-PaaS-TA와 비슷하게 공간정보 웹 서비스 배포가 가능한 공간정보 PaaS로 발전할 수 있다. 물론 실제 미들웨어 서비스를 제공하기 위해서는 사용자에 따른 서비스 Plan을 추가 설계하여 서비스 브로커가 구축되어야 한다. 설계 및 구축하여 배포한 공간정보 관련 웹 시스템은 사용자에게 제공될 수 있는 공간정보 처리 웹 서비스 중 일부분이며 특정 목적이나 기능을 고려하여 시스템 개발을 한 것이 아니다. 그렇기 때문에 실제 서비스를 위해서는 사용자 필요에 따라 별도로 시스템이 설계 및 구축되어야 한다. 하지만 웹 시스템을 설계할 때 공간정보 웹 표준을 활용하였기 때문에 구축된 시스템을 활용하여 얼마든지 확장할 수 있다. 예를 들어 사용자가 보유하고 있는 공간정보를 시각화하기 위해서 별도의 시스템을 구축하는 것이 아닌 GeoServer 서비스에 데이터를 업로드 함으로써 배포된 웹 시스템에서 바로 시각화할 수 있다. 공공 데이터 수집 또한 마찬가지로 현재 수집 가능한 OpenAPI는 서울열린 데이터 광장에서 제공하고 있는 데이터이다. 하지만 다른 기관에서 제공하고 있는 OpenAPI 제공 방식 패턴을 고려하여 해당 방식에 대해 비즈니스 서비스 모듈을 추가한다면 배포된 웹 서비스에서 수집을 할 수 있다. 마지막

으로 위성정보 처리 웹 서비스의 경우에도 자신이 원하는 처리 기능을 얼마든지 추가 가능하다. CLI 형태로 제공되는 소프트웨어 및 알고리즘을 가지고 있다면 공간정보 웹 표준 처리 소프트웨어인 Zoo-Project 서비스에 추가하고 입력받을 수 있는 인터페이스를 추가 개발하여 사용자가 원하는 처리 기능을 탑재한 웹 서비스가 가능해진다. 설계 시 표준을 활용하였기 때문에 국내 전자정부 표준프레임워크에서 채택하여 활용 가능하다. 연구에서도 언급했듯이 전자정부 표준프레임워크는 국내 공공 웹 시스템 개발 시 응용 소프트웨어 표준화, 품질 및 재 사용성 향상을 목표로 하고 있다. 연구에서 사용되고 있는 공간정보 웹 표준은 전 세계적으로도 공통 사용되고 있는 표준이다. 공간정보 관련 웹 서비스 제공 시 활용되는 표준을 전자정부 표준프레임워크에서 정의한다면 여러 공간정보 웹 시스템 개발에 대해 공유가 가능할 뿐만 아니라 각 기관 간 데이터 공유에도 상당한 이점이 있을 것이다. 특히, 맨 마지막에 수행된 시스템 성능 평가는 다양한 자료로 활용될 수 있다. 클라우드 컴퓨팅 도입이 필요한 기관이나 활용 예정 중인 곳에서 IaaS와 PaaS를 선택하거나 전자정부 표준프레임워크 도입에 대해 중요한 기초 자료가 될 수 있다.

제 5 장 결 론

이번 연구에서는 클라우드 컴퓨팅 기술에 대해 전반적인 내용과 세부 서비스 형태인 PaaS 관련한 이론적 및 기술적 내용을 정리하였다. 정리된 내용을 바탕으로 공간정보 분야에서 적용하기 위한 클라우드 및 웹 서비스의 다양한 설계 방식을 제시하고 이를 실제로 구축하여 PaaS에 직접 배포하여 구동 결과를 확인하였다. PaaS를 활용한다는 것을 전체적인 결과로 보았을 때 최종 사용자가 이용하는 관점으로 보았을 때 큰 변화를 찾아볼 수 없다고 느낄 수 있다. 하지만 다양한 측면에서 보았을 때 여러 장점을 지니고 있다는 것을 확인하였다. 먼저, PaaS를 구축한 결과 하드웨어 최대 능력을 끌어올릴 수 있다는 것을 확인하였다. 클라우드 컴퓨팅 기술은 하드웨어 능력을 최대로 활용하기 위해 만들어진 기술이다. 하드웨어 기술을 제공하는 IaaS만을 가지고 서비스를 구축하였을 경우에도 잠재되어 있는 하드웨어 능력을 활용할 수 있다. 하지만 IaaS만으로는 물리적 하드웨어에 이론적으로 계산된 값을 완벽하게 활용할 수 있다는 보장이 없다. 구축된 PaaS 위 미들웨어 소프트웨어와 어플리케이션을 배포하는 과정에서 컨테이너 기술을 활용하고 있다. 컨테이너 기술은 이번 연구에서 설계 및 구축된 시스템을 보면 확인할 수 있듯이 배포 및 확장 등 여러 관리 측면으로 보았을 때 가상 머신보다 가볍기 때문에 빠르다는 것을 확인하였다. 또한, 개발자 및 운영자 관점으로는 상당한 변화를 가져다준다는 점이다. PaaS를 사용한다면 개발자 관점에서 시스템 개발 시 테스트 및 운영환경에 대한 고려가 현저히 낮아진 것을 확인하였다. 기존에 하드웨어 및 소프트웨어 형태로 테스트 및 운영환경이 제공되어 직접 구축하여 사용되는 어려운 점이 존재하였지만, PaaS로 제공됨으로써 이 부분이 모두 서비스 형태로 변경되어 개발 또한 온디맨스 방식으로 제공받을 수 있다는 점이다. 특히, 서비스를 운영하기 위해 직접 빌드 및 운영 환경을 구축했던 개발자는 필요 없어졌다는 점이다. 운영환경을 관리하는 입장에서도 큰 변화를 확인할 수 있었다. 운영자는 미들웨어 소프트웨어를 설치 및 관리하는데 있어 통합적으로 관리할 수 있을 뿐만 아니라, 이번 연구에서 구축된

PostgreSQL 서비스에서 확인할 수 있듯이 배포된 앱은 공간에 따라 데이터베이스가 알아서 생성되고 삭제될 수 있는 기능이 포함될 수 있다. 추가로 물리적 용량을 Plan에 맞춰 제공함으로써 하드웨어 관리를 효율적으로 할 수 있다.

공간정보 분야에서 제공되는 서비스를 PaaS에 적용하였을 때에 대한 장점 또한 존재한다. 공간정보를 제공하기 위해서는 데이터 시각화 방법부터 관련 소프트웨어를 설치하거나 사용하는 방법에 대한 기술까지 추가로 습득해야 되는 사항이 너무 많이 존재한다. 특히, 이번 연구에 활용된 미들웨어 소프트웨어들은 이를 관리하는 전문적인 업체가 있을 정도로 복잡한 구성으로 구현되어 있다. 공간정보에 특화된 PaaS를 한다면 이렇게 복잡한 구성으로 구현되어 있는 미들웨어 소프트웨어를 개발자는 고민할 필요 없이 바로 가져다 사용할 수 있다는 장점이 존재한다. 이번 연구에서 활용된 서비스는 공간정보 웹 표준을 모두 지원하고 있어 이를 사용함으로써 제공 지역 범위가 넓어질 수 있어 국내뿐만 아니라 전 세계적으로 서비스할 수 있다.

마지막으로 국내에서 제공되고 있는 전자정부 표준프레임워크에 대한 지원을 고려하여 설계 및 구축되었으므로 향후 연결 잠재성을 가지고 있다. 전자정부 표준프레임워크는 많이 활용되고 있는 오픈소스를 사용하여 이에 대한 기능에 대한 성능은 보장받을 수 있으며, 최근에는 국내뿐만 아니라 세계적으로 활용이 되고 있는 표준프레임워크이다. 국내 공공기관에서 구축되는 전자정부 시스템은 전자정부 표준프레임워크를 기반으로 구축하는 것을 권장하고 있다. 그리고 표준프레임워크는 PaaS로 이전하기 위한 노력이 현재 수행되고 있다. 이번 연구에서도 이러한 동향을 고려하여 PaaS에서 제공되는 기술들을 사용하고 공간정보 처리 웹 시스템을 전자정부 표준프레임워크 실행환경을 기반으로 설계 및 구축해보았다. 공간정보 시스템은 데이터를 보유하고 있는 공공기관에서 시스템을 제공함으로써 좀 더 서비스 콘텐츠에 대한 극대화를 볼 수 있으며, 일반 기업에서도 전자정부 표준프레임워크에서 제공하는 공통 컴포넌트들을 활용하여 서비스를 구축할 수 있다. 국내에서 제공되고 있는 금융기관 관련 PaaS인 K-PaaS-TA와 같이 특정 분야에 맞춰 서비스될 수 있다. 이와 같이 일반 서비스에서 공간정보 관련된 미들웨어 소프트

웨어들을 서비스화하여 사용할 뿐만 아니라 공간정보 분야를 많이 활용하는 특정 전문기관에서 이번 연구에서 제시한 PaaS와 같이 폐쇄적 형태의 PaaS로 활용할 수 있다. 이번 연구에서 구축된 모든 시스템은 향후 공간정보 분야를 포함하여 모든 분야 PaaS 시스템에 적용될 수 있으며, 특히 공간정보 분야에서는 정보통신기술에 대한 동향에 맞춰 이를 실제로 구축할 수 있는 가능성을 확인할 수 있는 연구로 향후 관련 서비스에 활용되기를 기대할 수 있다.

참 고 문 헌

1. 국내문헌

강상구, 이기원. (2014). 오픈소스 모바일 클라우드 서비스 R 기반 공간영상정보 필터링 사례.『한국지리정보학회지』, 22(5), 1-8.

강상구. (2016).『모바일 클라우드 환경에서 공간정보 처리 분석 서비스 응용』. 한성대학교.

공개 SW 포털. (2017). 클라우드/빅데이터 분야 공개 SW 솔루션 목록. http://www.oss.kr/oss_intro12 (Accessed October 2017).

과학기술정보통신부. (2017).『클라우드컴퓨팅 발전 및 이용자 보호에 관한 법률』. 국가법령정보센터.

김광섭, 이기원. (2014). R 이용 오픈데이터 시각화 웹 응용.『한국지리정보학회지』, 17(2), 72-81.

김광섭, 이기원. (2015). 전자정부 표준 프레임워크 모바일 실행환경 기반 공공데이터와 공간데이터 시각화.『한국공간정보학회지』, 23(1), 9-17.

김학진. (2017).『기업 생전을 위한 클라우드, 제대로 준비하고 계십니까』. (10). IDG Summary.

박준환. (2015).『이제는 오픈소스 시대 기업의 오픈소스 도입 전략』. (11). IDG Summary.

서광규, 장진산, 이재웅, 김동현. (2016).『클라우드 컴퓨팅 서비스 품질·성능 기준 연구』. 상명대학교 산학협력단.

서상수. (2016).『유행 댓딘 보안, 컨셉 아닌 실용 한국형 프라이빗 클라우드의 핵심 요건』. IDG Summary.

신동희, 김경민. (2014). 서비스형 소프트웨어 기술(연구)동향.『한국인터

넷정보학회지』, 15(1), 24-37.

윤구선, 이기원. (2015). 클라우드 컴퓨팅과 웹 프레임워크 환경에서 WPS 기반 위성영상 정보처리.『한국원격탐사학회지』, 31(6), 561-570.

윤구선, 김광섭, 이기원. (2016). 오픈스택 클라우드 환경 OGC WPS 2.0 기반 위성영상처리 성능측정 시험. 『한국원격탐사학회지』, 32(6), 617-627.

윤구선. (2017).『모바일 웹 환경 OGC-WPS 표준 적용 위성영상처리 시스템 구현』. 한성대학교.

윤준희, 김창윤, 문현석. (2017). 국가 공간정보 인프라의 클라우드 서비스 기술개발 방안 수립.『한국산학기술학회논문지』, 18(3), 469-477.

이봉옥. (2009). 전자정부 표준프레임워크 소개 및 활성화 계획.『한국지역정보개발원』, 58(0), 46-51.

이인수, 허용, 이재강, 이재원. (2016). 클라우드의 공간정보분야 연계 방안 연구.『한국지적정보학회지』, 18(2), 3-16.

전자정부 표준프레임워크. (2017).『전자정부 표준프레임워크 국문 리플릿』. (8). 행정자치부.

전형철, 강선무. (2013). 전자정부 표준프레임워크와 IT서비스 기업 글로벌 경쟁력 강화 방안.『한국통신학회지』. 30(9), 72-77.

한국클라우드컴퓨팅연구조합 연구개발팀. (2017).『클라우드컴퓨팅 기술 스택』. (108). 한국클라우드 컴퓨팅연구조합.

행정자치부. (2016).『전자정부 표준프레임워크 2016 최신버전 및 우수사례 설명회: 표준프레임워크 현황 및 차세대 프레임워크 발전전략』. (136). NIA 한국정보화진흥원.

Boulton, C. (2017).『2017 IT 전망 보고서: 두 번째 파도 온다 2017년 클라우드 컴퓨팅 트렌드 6가지』. (19-21). IDG Deep Dive.

Carr, D. F. (2015).『2015 클라우드 컴퓨팅의 과제와 전망: PaaS로 해결

할 수 있는 6가지 비즈니스 현안』. (7-12). IDG Deep Dive.
IDG Market Pulse. (2017). 『Ems 구름 잡던 시절은 지났다. 2017년 한국의 클라우드 현주소』. IDG Research.

2. 국외문헌

Avram, M. G. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective, Procedia Technology(The 7th International Conference Interdisciplinarity in Engineering), 12. 529-534.

AWS. (2017). Landsat on AWS. https://aws.amazon.com/public-data-sets/landsat/ (Accessed October 2017).

Castronova, A. M., Goodall, J. L., and Elag, M. M. (2013). Models as web services using the Open Geospatial Consortium Web Processing Service standard. Environmental Modelling & Software, 47, 72-83.

Chou, D. C. (2015). Cloud computing: A balue creation model. Computer Standards & Interface, 38, 72-77.

Costache, S., Dib, D., Parlavantzas, N., and Morin, C. (2017). Resource management in cloud Platform as a Service systems: analysis and opportunities. The Journal of Systems & Software, 132, 98-118.

Dash, S., and Pani S. K. (2016). E-Governance paradigm using cloud infrastructure: benefits and challenges. Procedia Computer Science(International Conference on Computational Modelling and Security), 85, 843-855.

Ferrer, A. J., Perez, D. G., and Gonzalez, R. S. (2016). Multi-Cloud Platform-as-a-Service model, functionalities and approaches. Procedia Computer Science(Cloud Forword: from Distributed to Complete Computing), 97, 63-72.

Fowley, F., Pahl, C., and Zhang, L. (2013). A Comparison framework and review of service brokerage solutions for cloud architectures. (137-149). Service Oriented Computing(ICSOC 2013 Workshops).

Gigliani, G., Nativi, S., Lehmann, A., and Ray, N. (2012). Computers and Geosciences, 47, 20-33.

Goncalves, V., and Ballon, P. (2011). Adding value to the network: Mobile operators' experiments with Software-as-a-Service and Platform-as-a-Service models. Telematics and Informatics, 28, 12-21.

Hare, T. M., Rossi, A. P., Frigeri, A., and Marmo, C. (2017). Interoperability in planetary research for geospatial data analysis. Planetary and Space Science, https://doi.org/10.1016/j.pss.2017.01.016 (Accessed October 2017).

Huang, W., Zhang, W., Zhang, D., and Meng, L. (2017). Elastic spatial query processing in OpenStack cloud computing environment for time-constraint data analysis. International Journal of Geo-Information, 6(3), 84-101.

Hussain, H., and Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., Qureshi, M. B., Zhang, L., Yongji, W., Ghani, N., Kolodziej, J., Zomaya, A. Y., Xu, C., Balaji, P. Vishnu, A., Pinel, F., Pecero, J. E., Kliazovich, D. Boucry, P., Li, H., Wang, L., Chen, D., and Rayes, A. (2013). A surcey on resource allocation in high performanace distributed computing systems. Parallel Computing, 39, 709-736.

Khare, A. B., Raghav, V., and Sharma, P. (2012). Cloud computing based rural E-Governance model. Journal of Information and

Operations Management, 3(1), 89-91.

Kim, K., and Lee, K. (2016). Real-time processing of spatial attribute information for mobile web based on standard web framework and HTML5. Spatial Information Research, 24(2), 93-101.

Kim, K., Kang, S., and Lee, K. (2013). Geo-based image blending in a mobile cloud environment. Remote Sensing Letters, 4(11), 1117-1126.

Klug, H., and Kmoch, A. (2014). A Smart groundwater portal: An OGC web services orchestration framework for hydrology to improve data access and visualization in New Zealand. Computers and Geosciences, 69, 78-86.

Lee, K., Kang, S., Kim, K., and Chae, T. (2017). Cloud-based statellite image processing service by open source stack: A KARI Case. Korean Journal of Remote Sensing, 33(4), 339-350.

Li, Z., Yang, C., Liu, K., Hu, F., and Jin, B. (2016). Automatic scaling Hadoop in the cloud for efficient process of big Geospatial data. International Journal of Geo-Information, 5, 173-187.

Lopez-Pellicer, F. J., Renteria-Agualimpia, W., Bejar, R. Muro-Medrano, P. R., and Zarazage-Soria, F. J. (2012). Availability of the OGV geoprocessing standard: March 2011 reality check. Computers and Geosciences, 47, 13-19.

Mell, P., and Grance, T. (2011). The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology.

Mueller, M., and Pross, B. (2015). OGC WPS 2.0 Interface Standard. (113). Open Geospatial Consortium.

Mukherjee, K., and Sahoo, G. (2010). Cloud computing: feature framework for e-Governance. International Journal of Computer Applications, 7(7). 31-34.

OTB Development Team. (2016). The ORFEO Tool Box software guide. Centre National D'etudes Spatiales.

Pivotal. (2017). Pivotal Cloud Foundry Documentation. Pivotal Software.

Rautenbach, V., Coetzee, S., Strzelecki, M., and Iwaniak, A. (2012). Results of an evaluation of the orchestration capabilities of the Zoo Project and The 52 North framework for an intelligent geoportal. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, I-4, 163-168.

Smitha, K. K., Tony, T., and Chitharanjan K. (2012). Cloud based E-Governance system: A survey. Procedia Engineering(International Conferemce on Modelling, Optimization and Computing), 38, 3816-3823.

Vaquero, L. M., Rodero-Merino, L., and Buyya, R. (2011). Dynamically scaling applications in the cloud. ACM SIGCOMM Computer Communication Review, 41, 45-52.

Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., and Bambacus, M. (2011). Spatial Cloud Computing: How geospatial sciences cloud use and help to shape cloud computing. International Journal on Digital Earth, 4, 305-329.

Yang, C., Yu, M., Hu, F., Jiang, Y., and Li Y. (2017). Utilizing cloud computing to address big geospatial data challenges. Computers, Environment and Urban Systems, 61, 120-128.

Yoon, G., Kim, K., and Lee, K. (2017). Linkage of OGC WPS 2.0 to the e-Government standard framework in Korea: An Implementation case for geo-spatial image processing. International Journal of Geo-Information, 6(1), 25-39.

Zhan, Z., Liu, X., Gong, Y., and Zhang, J. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surceys, 47(4), 1-25.

Zhao, P., Foerster, T., and Yue, P. (2012). The Geoprocessing web. Computer and Geosciences, 47, 3-12.

Zoo-Project Team, 2017. ZOO-Project Document, Release 1.5. ZOO-Project.

ABSTRACT

Development of Geo-based Information Processing System based on Open PaaS Cloud Computing and Its Performance Evaluation

Kim, Kwang-Seob

Major in Information System Engineering

Dept. of Information and Computer Engineering

The Graduate School

Hansung University

Recently, the software development scheme and its usage pattern are changing due to the Internet environment. It is the on-demand way to use the software directly through a web browser without any installation. Evolution of information and communication technology plays a huge role in the Web-based applications or service system development. In particular, cloud computing technology leads this advancement. Cloud computing is a collection of core technologies such as virtualization, network, storage, and distributed computing. It can be basically divided into three types of service model: Infrastructure as a Service(IaaS), Platform as a Service(PaaS), and Software as a Service(SaaS). Also, it can be classified into three deployment types, depending on the degree of openness: public, public, and hybrid. There are lots of research issues and development approaches in the theme of broad cloud computing. In case of Infrastructure as a Service, its related technologies and services have reached the stable stage; for instance, Amazon Web Service (AWS) and Google Cloud Platform. On the contrary, Platform as a Service standing for PaaS is an emerging cloud computing scheme. It provides

technological bases and practical computing resources for Web-based applications developers and on-line system operators to design, build, and operate. Moreover, the e-government standard framework is evolving into the open cloud platform. Despite these huge paradigm movements in the information technology, the geo-spatial domains are still at an early stage in all aspects of core technology and application model development. Especially, PaaS is a new concept in the geo-spatial application fields. This study explores basic components for the geo-spatial application of PaaS cloud computing from the viewpoint of full open source: PaaS container, PaaS application performance, and PaaS cloud application model. To solve this problem, the test system on Spring framework was implemented using the pure open source base containing OpenStack and Cloud Foundry. As the results, the web-based geo-spatial application service development by PaaS container shows the advantageous points for system availability and extensibility. Experiments for PaaS performance test show significant results that both IaaS application and PaaS application are within the given tolerance level. PaaS cloud application model proposed in this study can be regarded as a reference model for the geo-spatial application service system development with many requirements including open source, PaaS, e-government framework, and OGC-based geo-data processing.

KEYWORD: Cloud Computing, Platform as a Service, E-Government Standard Framework, Geo-spatial Web Standard, Open Source