MSA(Microservices Architecture)를 활용한 스마트팜 시스템 구축 제안

곽두일[1], 박광영[1]
[1]숭실대학교 AI테크노융합학과
ai@soongsil.ac.kr, 1004pky@ssu.ac.kr

Proposal for building a smart farm system using Microservices Architecture (MSA)

Doo-il Kwak[1], Kwang-Young Park[1]
[1]Dept. of AI Techno Convergence, Soongsil University

요 약

스마트팜 혁신밸리는 스마트팜에 특화된 스마트팜 기자재 연구·실증 기능의 집약을 위해 추진되었다. 하지만 기존의 스마트팜 실증단지 구축 시스템은 모놀리식 접근 방식을 사용해서 시스템 내 다양한 기능의 업데이트와 유지보수를 하는데 많은 시간과 인력을 필요로 한다. 이에 본 논문은 차후 스마트팜의 실증단지 운영시스템 구축에 마이크로 서비스 아키텍처를 활용한 '마이크로 서비스' 시스템을 제시하는 것을 목적으로 한다. 이를 통해 스마트팜 기능의 통합을 개선하고 대규모 배포를 위한 성능을 향상시킬 수 있다. 이를 위해 MSA 설계 전략에 대한 제언을 한다. 향후 실제 실증단지 운영시스템 구축에 본 제안의 유용성을 확인하는 작업이 필요하다.

1. 서론

스마트팜 혁신밸리는 스마트팜에 특화된 청년농육성, 스마트팜 기자재 연구·실증 기능을 집약해 농업인-기업-연구기관 간 시너지를 창출하는 거점[1]으로 정의된다. 이 스마트팜 혁신밸리는 「스마트팜 확산방안('18.4월, 부처 합동)」에 따라 스마트농업 인력·기술의 확산 거점으로 「스마트팜 혁신밸리」 조성이 추진(~'22년)되었다. 이에 김제·상주(1차지역, '18.8월), 밀양·고흥(2차지역, '19.3월) 등 4개소가 선정이 되었다. 1차 지역 실증단지의 운영 시스템은 모놀리식 접근 방식이어서 스마트 농업의 여러 기능들을 효과적으로 통합하고 대규모 배포를 위한 성능을 향상시키는 것이 어려웠다.

이에 본 논문은 'MSA 기반 배포 모델'을 소개하여 2차지역 운영시스템 구축에 도입하는 것을 제안하고자 한다. 이 모델은 클라우드 네이티브로 다수의 농장을 관리하고자 하는 분산형 시스템을 사용하여 수집된 모든 데이터를 클라우드에서 처리해야 한다[2]. 무엇보다 다양한 기능이 통합되어 있는 실증단지 시스템 운영에서 발생하는 각 기능상의 오류에 대해 빠른 유지보수가 가능하다는 장점이 있다.

2. 관련 연구

모카누 외[3]는 기존 농장 관리 시스템을 개선하고, 모니터링, 농장 운영 관리와 같은 프로세스를 클라우드의 특정 기능과 통합하여 생산성을 향상시킬 수 있는 클라우드 아키텍처를 제안했다. 콜제아 외[4]는 농민들이 농장을 어디서든 쉽게 관리하고 감독할 수 있는 서비스를 제공하는 CLUeFARM 플랫폼을 제안한다.

오그래디와 오헤어[5]는 다양한 농업 활동 모델을 통합하는 의사 결정 지원 도구와 농장별 모델의 개발을 제안한다. 또한 농업 기업 내 모델에 대한 개요를 제시하고 스마트 기술의 최신 상태를 검토한다. 이 연구의 흥미로운 결론 중 하나는 많은 모델의 모놀리식 특성으로 인해 개별 농가가 이러한 모델을 실제로 적용하는 데 어려움을 겪고 있다는 것이다. 따라서 마이크로 서비스 접근 방식을 사용하면 이러한 모델과 시스템을 실제로 더 잘 채택할 수 있다.

그 외 MSA에 기반한 시스템 아키텍쳐를 제안하는 연구는 찾아보기가 어려웠다.

3. 모노리스 시스템의 한계

1차지역 실증단지 운영시스템 구축은 모노리스(Monolithic) 기반으로 시스템이 구성되었고, 다음과 같은 문제점이 도출되었다.

- 탄력적인 클라우드 인프라를 위해 설계되지 않음
- 온실 복합환경 제어기는 기성품 형태로 조립 설치되는 구조로 원천데이터 소스가 제공되나, 수정 및 변경이 어려움
- 온실 기자재 업체와 운영시스템 구축업체 상호간 프로그램 이해에 대한 부족으로 프로토콜을 맞추는데 어려움
- 외산 장비의 경우 원천데이터 제공 불가로 관리자 화면에서 데이터 연계 구성이 어려움

4. MSA 설계 이슈 및 전략

MSA는 하나의 애플리케이션을 여러 개의 작은 서비스로 개발하는 접근 방식이다.[6] 모노리식 설계의 단점을 극복한 MSA는 각 기능을 독립적으로 분리한만큼 다양한 이슈가 존재한다.

설계이슈	세부 설명	설계 전략
마이크로서비스 분리	• MSA 구조로 적용하기 위하여 전체 서비스 및 데이터를 마이크로서비스로 분리 적용 필요 • 어떠한 기준으로 서비스를 분리할 것인지 분리 전략 도입 필요	이벤트 스토밍 DDD 업무기능분해
서비스간 통신방식	• 각 서비스 요소 간 loosely coupled 통신 필요 • 상황에 따라 동기 통신 또는 비동기 통신 방식 적용 필요	동기통신 비동기통신
쿼리패턴	• 서비스와 정보가 분리됨에 따라 하나의 기능을 위해 여러 서비스 호출이 한꺼번에 필요함 • 특정 서비스의 호출에 성능 이슈가 발생할 수 있으며 이에 대한 대응 전략 필요	API 패턴 조합 CQRS
트랜잭션 관리	• 하나의 Command 호출에 대해 여러 마이크로서비스가 함께 트랜잭션 서비스 구동 • Command가 비정상 처리 시 각 마이크로서비스의 트랜잭션에 대해 복구 전략 도입 필요	SAGA Event Sourcing

표 1 MSA 설계 이슈 및 전략

4.1. 마이크로서비스 분리

업무 기능 분해 기법은 기관 보유 업무 프로세스 관련 자료, 애플리케이션 앱, 체계도를 활용하여 마이크로서비스를 나누는데 유용한 소규모 인력으로 수행 가능한 기법으로 판단된다. 아래 표는 업무 기능 분해 기법과 사업 기간, 업무 우선순위를 고려하여 분류된 마이크로 서비스 그룹의 사례이다.

마이크로서비스 그룹	마이크로서비스
시설 및 장비 관리	시설물 현황
	임대장비 및 비용 관리
일정 및 신청 관리	일정 관리
	방문 신청 및 입주 신청 관리
사용자와 회원 관리	회원 관리
	게시판 정보
	경영일지
과금 및 분석	과금 및 ROI 분석
	작물 생육 분석 및 밸런스 분석
시스템 접근 및 모니터링	사용자 관리
	메뉴 및 접근 권한 관리
	캘린더 정보 관리
	설비 모니터링 및 제어 관리
데이터 관리 및 연계 정보	입주 기업 관리
	데이터 관리
	내외부 연계 정보
	노지 데이터 수집
AI 분석 및 시설 관리	시설 이력 관리
	작물 기본 정보 및 병해충 정보
	통계 및 대시보드
	실증단지 AI 분석 관리
	AI 분석 환경(PaaS) 구성

표 2 업무기능 분해에 따른 마이크로서비스 그룹

4.2. 서비스간 통신방식

통신방식은 요청과 응답 기반 프로토콜을 사용하는 동기식 통신과 메시지 교환을 위해 카프카와 같은 메시지 브로커를 사용하는 비동기 통신을 상황에 따라 사용한다. 각 서비스간 통신방식은 둘 다 사용한다.

4.3. 쿼리패턴

마이크로서비스로 API 호출된 결과를 조합하여 결과를 생성하거나, CQRS(Command Query Responsibility Segregation; 명령과 조회의 책임 분리)를 사용하여 입력, 수정, 삭제(CUD)를 위한 DB와 조회(R)를 위한 DB를 분리하여 구성한다.

4.4. 트랜잭션 관리

마이크로 서비스는 구조적으로 트랜잭션 구현 이슈를 가지고 있다. 이것은 SAGA 패턴과 이벤트 소싱으로 극복 가능하다. SAGA 패턴은 다시 이벤트 기반 SAGA와 오케스트레이션 SAGA로 나뉘는데, 2차지역 실증단지 운영시스템의 경우, 복잡성을 줄이기 위해 이벤트 기반 SAGA가 보다 더 용이하다 할 수 있다.

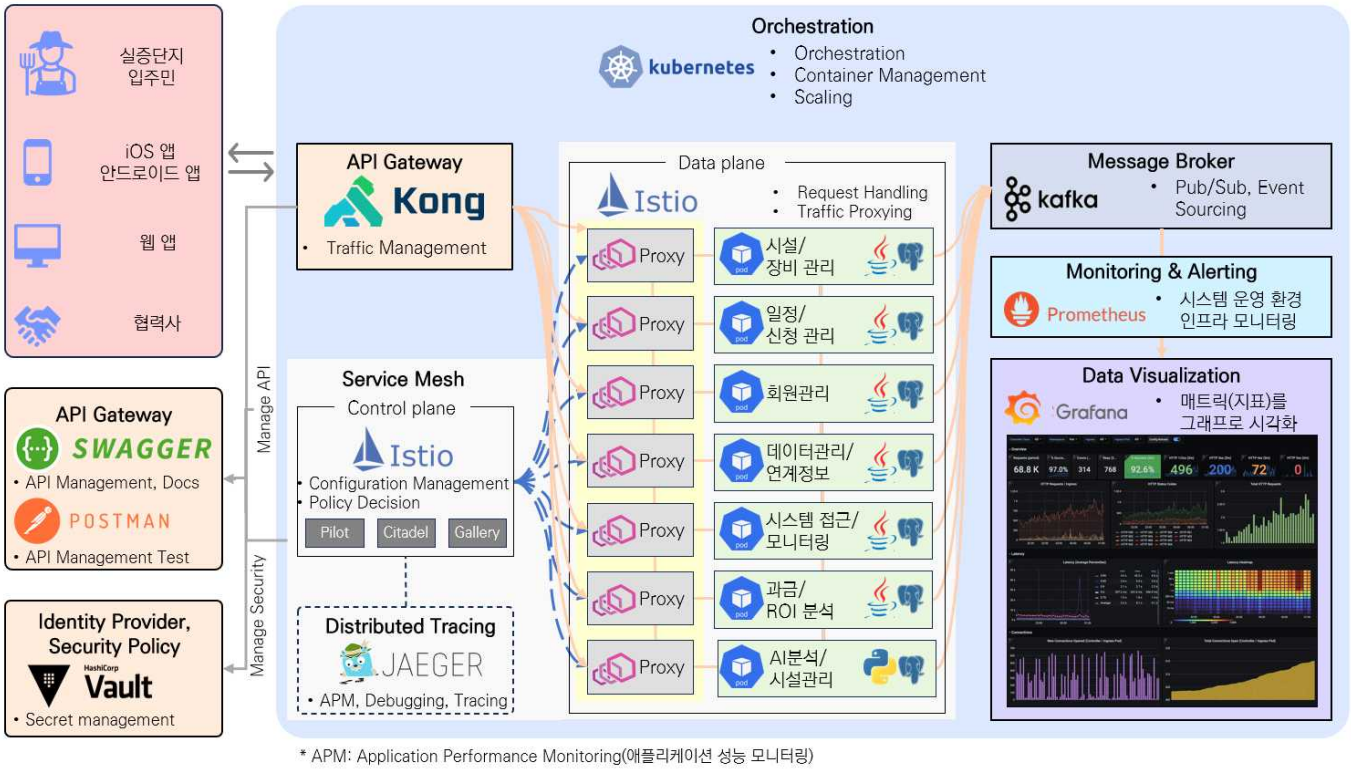

그림 1 MSA 아키텍처 개요

5. 클라우드 네이티브 인프라 구성

MSA는 클라우드 네이티브와 뗄려야 뗄 수 없는 관계에 있다. 위 그림 1와 같이 클라우드 인프라를 구성한다. 클라우드의 경우, 프라이빗 클라우드를 기본으로 하였다.

5.1. 사용자 진입점:

실증단지 입주자는 iOS 앱, Android 앱 또는 웹 앱을 통해 애플리케이션에 접속한다.

5.2. API Gateway와 인그래스 및 이그래스:

사용자 요청은 API Gateway를 통해 진입한다. Kong API Gateway를 통해 요청을 받아들이고, 인그래스 컨트롤러를 통해 들어온 트래픽을 분산시킨다. 이때, Swagger를 통해 API 문서를 작성하여 유지보수를 대비하고, Postman을 통해 API 관리 테스트를 시행한다. 보안 비밀 관리, 암호화와 보안 토큰 생성 및 관리와 동적 보안을 위해 Hashicorp Valut라는 오픈소스 도구를 활용한다.

인그래스 컨트롤러(이스티오):

API Gateway에서 받은 요청을 기반으로 경로 및 라우팅 규칙을 확인한다. 이후 해당 요청을 처리하기 위한 적절한 마이크로서비스로 라우팅한다.

이그래스 트래픽 관리(이스티오):

인그래스 컨트롤러는 이그래스 트래픽 관리를 수행하여 외부에서 들어오는 트래픽을 관리하고 분배한다. 이를 통해 사용자 요청이 적절한 마이크로서비스로 전달된다.

5.3. 프록시(엔보이):

인그래스 컨트롤러에 의해 라우팅된 요청은 엔보이 프록시를 통해 마이크로서비스로 중개된다. 요청 및 응답을 관리하며, 서비스 메시와 함께 작동, 보안 및 모니터링을 제공한다.

5.4. 마이크로서비스 그룹:

다양한 비즈니스 로직을 담당하며, 각각은 특정한 기능 또는 역할을 수행한다. 엔보이 프록시를 통해 중개된 요청은 해당 마이크로서비스 그룹으로 전달되고 처리된다. 각각의 마이크로서비스 그룹은 기능에 따라 개발 언어와 데이터베이스가 달라질 수 있다. AI분석/시설관리 마이크로서비스 그룹을 보면 자바를 사용한 다른 그룹과 달리 파이썬 언어로 개발하게끔 되어 있음을 볼 수있다.

5.5. 메시지 브로커(카프카):

일부 마이크로서비스는 카프카 메시지 브로커를 통해 이벤트 기반 통신을 수행한다. 이를 통해 비동기

적인 작업 및 데이터 흐름이 관리된다.

5.6. 텔레메트리; 로깅, 모니터링, 추적(프로메테우스, 그라파나):

프로메테우스가 시스템 운영환경과 인프라에 대해 모니터링을 하고, 프로메테우스를 통해 수집된 로깅, 모니터링 및 추적정보를 그라파나가 그래프로 시각화한다. 이것을 통해 애플리케이션의 성능 및 상태를 실시간으로 모니터링할 수 있다.

이러한 네트워크 구성은 사용자의 요청을 안전하게 관리하고, 마이크로서비스 간 효율적인 통신을 보장하며, 텔레메트리를 통해 시스템을 모니터링하고 최적화하는 데 중요한 역할을 한다.

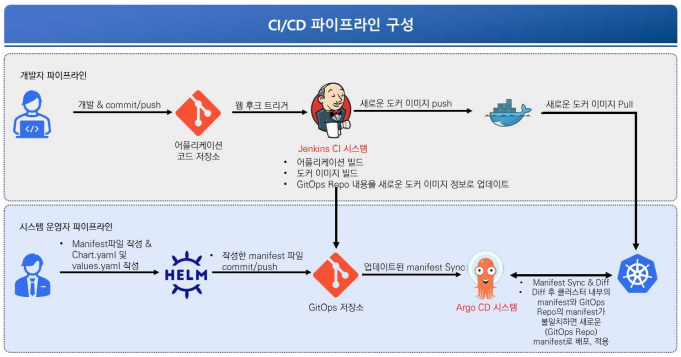

그림 2 CI/CD 파이프라인 구성

위 그림 2[7]는 CI/CD 파이프라인 구성에 대한 구성도로 이렇게 할 경우, 효율적인 SW개발이 가능하고, 반복적인 작업을 컴퓨터가 자동화함으로서 개발자들은 보다 더 창의적인 개발을 할 여유가 생기게 된다.

파이프라인의 구조를 살펴보자.

젠킨스가 가장 우선적으로 새로운 코드를 빌드한 후 도커 이미지를 만들고 이것을 도커허브에 저장한다. 이에 대한 새로운 도커 이미지 버전 태그 내용을 깃옵스 리포지토리 내용에 업데이트한다. 그리고 아르고CD에서는 신규 업데이트 된 깃옵스 리포지토리의 내용을 감지해서 현재 쿠버네티스에 배포된 메니페스트 내용과 새로 업데이트 된 깃옵스 리포지토리의 매니페스트 내용을 비교(Diff) 하여 업데이트 된 내역을 확인하면 운영자의 조작에 따라서 혹은 자동으로 업데이트 된 내용을 적용(Sync)하여 쿠버네티스에 배포한다.

6. 결론

본 논문은 스마트팜 혁신밸리 실증단지 운영시스템을 MSA기반으로 개발하는 방안을 제안하는 것을 목표로 한다. 이를 위해 MSA 설계상의 이슈와 그에 따른 전략을 소개했다. 이어 클라우드 네이티브 참조 모델을 제시하여 프라이빗 클라우드 상에서의 운영 시스템 구축을 제안하고 CI/CD 파이프라인 구성을 제안하였다. 따라서 각 MSA 설계 전략과 클라우드 네이티브 상의 MSA 요소기술을 활용하여 기존 1차 지역 실증단지 운영시스템보다 한단계 진보된 시스템 구축이 가능해보인다.

향후 본 논문에서 제시된 설계가 적용된 운영시스템을 구축하고 그에 따른 개발 이슈에 대한 후속 연구가 필요하다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 지역지능화혁신인재양성사업의 연구결과로 수행되었음" (IITP-2023-RS-2022-00156360)

참고문헌

[1] "사업내용." 2023. Mafra.go.kr. 2023. https://www.mafra.go.kr/home/5281/subview.do.
[2] Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. Big data in smart farming – A review. *Agricultural Systems*, *153*, (2017): 69 - 80. https://doi.org/10.1016/j.agsy.2017.01.023
[3] M. Mocanu, V. Cristea, C. Negru, F. Pop, V. Ciobanu and C. Dobre, "Cloud-Based Architecture for Farm Management," 2015 *20th International Conference on Control Systems and Computer Science*, Bucharest, Romania, (2015), 814-819, doi: 10.1109/CSCS.2015.55.
[4] Negru, Catalin, George Musat, Madalin Colezea, Constantin Anghel, Alexandru Dumitrascu, Florin Pop, Carmen De Maio, and Aniello Castiglione. "Dependable workflow management system for smart farms." *Connection Science* 34, no. 1 (2022): 1833-1854.
[5] O'Grady, M. J., & O'Hare, G. M. (2017). Modelling the smart farm. *Information Processing in Agriculture*, *4*(3), 179 - 187. https://doi.org/10.1016/j.inpa.2017.05.001
[6] Lewis, James. "Microservices." Martinfowler.com, 2014, martinfowler.com/articles/microservices.html. Accessed 23 Sept. 2023.
[7] Bang, ShinChul. 2020. "[FINDA] MSA를 위한 Kubernetes 세팅과 CI/CD Pipeline 구성, 그리고 Monitoring 시스템 구축 – 2." Medium. finda 기술 블로그. February 24, 2020. https://medium.com/finda-tech/finda-msa%EB%A5%BC-%EC%9C%84%ED%95%9C-kubernetes-%EC%84%B8%ED%8C%85%EA%B3%BC-ci-cd-pipeline-%EA%B5%AC%EC%84%B1-%EA%B7%B8%EB%A6%AC%EA%B3%A0-monitoring-%EC%8B%9C%EC%8A%A4%ED%85%9C-%EA%B5%AC%EC%B6%95-2-ef29380ec474.