https://doi.org/10.7840/kics.2025.50.4.676

서비스 호출 분석을 통한 모놀리식 시스템의 도메인 기반 마이크로서비스 전환 자동화 시스템 연구

박 선 철◆, 김 영 한°

Automating Domain-Based Microservices Conversion from Monolithic Systems through Service Call Analysis

Sun-chul Park◆, Young-han Kim°

요 약

기존 시스템을 클라우드 네이티브 환경으로 전환할 때 개별 도메인의 리소스를 식별하기 위한 도메인 경계 분석이 필수적이다. 경계 분석은 모놀로식 구조를 컨테이너 단위로 분해하기 위한 것으로 소스코드뿐 아니라 연관 DB를 통합한 분석이 필요하다. 기존 연구들은 대부분 소스코드의 정적분석 방식에 의존했으나, 이는 특정 대상에 맞춘 방식으로 재사용이 어렵고, 중대형 개발에서 사용하는 개발 프레임워크 환경에는 적용할 수 없어 불완전하다. 본 연구는 실시간으로 서비스의 이벤트를 추적하여 소스코드와 DB를 통합하여 분석하는 동적분석 기반 경계 분석 시스템을 제안한다. 제안시스템은 WAS와 DB를 연결하는 미들웨어의 래퍼 클래스에서 이벤트 발생에 대한 소스파일, 호출된 함수, DB 처리정보를 결합하여 호출 간의 연관관계를 분석한다. 성능시험 결과 정적분석 대비 18% 더 정확도가 높고, 개발 프레임워크 환경도 지원함을 증명하였다. 또한, 제안기법은 수작업 분석 대비 저비용의 신속함과 다양한 개발환경에 적용할 수 있어 내부 전문가도 손쉽게 분석할 수 있는 장점을 제공한다.

키워드 : 클라우드 네이티브, 클라우드 마이그레이션, MSA, 도메인 경계 분석
Key Words : Cloud-Native, Cloud Migration, MSA, Domain Boundary Analysis

ABSTRACT

This research focuses on automating domain boundary analysis and facilitating the migration of existing systems to a cloud-based microservices architecture (MSA). Previous research has relied on static analysis methods that have various limitations, but this research employs dynamic analysis to overcome these constraints. Experiments show that database-backed source code is used 18% more compared to static analysis methods. Moreover, this approach makes it easier to modify code and allows in-house experts to easily perform the analysis. This methodology not only streamlines the migration process but also increases the accessibility of analysis to developers.

Ⅰ. 서 론

클라우드 네이티브인 마이크로서비스 아키텍처(MSA)로의 전환을 위해서는 기존 서비스를 도메인 단위로 분할하는 서비스 경계 분석이 필수적이다. 도메인 단위 분할은 서비스 기능과 유관한 소스코드와 데이터

◆ First Author : Soongsil University Graduate School of AI IT Convergence, sunpark@soongsil.ac.kr, 학생회원
° Corresponding Author : Soongsil University School of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
논문번호 : 202410-234-C-RN, Received October 8, 2024; Revised November 18, 2024; Accepted December 10, 2024

베이스(DB) 같은 리소스를 분류해야 하는데, 기존 시스템에서 리소스를 분류하는 과정은 복잡하여 숙련된 외부 전문가에 의존하는 고비용의 원인이 되고 있다. 또한, 기존 연구들은 대부분 정적분석 방식으로 리소스의 의존성을 분석해 왔으나, 이런 방식은 특정 개발환경에 맞춰져서 비효율적이며, 동적으로 코드가 완성되는 개발 프레임워크 환경에서는 적용할 수 없는 한계가 있다.

본 연구는 실시간으로 서비스의 이벤트를 추적하여 서비스와 관련된 리소스를 분석할 수 있는 동적분석 기반 시스템을 제안한다. 제안 기법은 실시간 생성 코드 호출을 추적하므로 개발 프레임워크 환경도 지원하고, 별도의 전용환경이나 메타데이터 구축같은 부가적인 선행 작업을 필요로 하지 않는다. 성능시험은 오픈소스에 적용 후 기능을 검증하였고, 정적분석과 비교하여 제안기법이 정확도가 높고 신속하여 서비스 경계 분석에 더 효율적임을 증명하였다. 제안 시스템을 통해 전환 비용 절감으로 클라우드 최적화 전환에 효과적임을 기대할 수 있다.

서론에 이어 2장에서는 클라우드 및 MSA 전환 자동화와 관련된 선행 연구와 개발모델, 백트레이스에 대해 살펴보며, 3장에서는 제안기법을 상세히 설명하고 4장에서는 구현 및 실험 평가 내용을 정리 후 마지막으로 5장에서 결론을 제시한다.

Ⅱ. 관련 연구

2.1 클라우드 마이그레이션

기존 서비스의 클라우드 전환은 대부분 단순전환(Lift-and-Shift)에 중점을 두고 있다. 최적화전환(Optimization)은 서비스의 개선과 향후 확장성은 좋지만, 서비스 재구조화로 인한 높은 전환비용이 요구된다. M. H. Hasan et al.은 MSA 전환시 서비스 경계 정의, 재구조화 및 재설계 요구, 기존 코드의 복잡성과 문서화 부재 같은 분석단계에서의 문제를 장애요소로 분석하였다[1]. J. Fritzsch et al.의 연구에서 재구조화를 위한 대부분의 연구가 특정한 조건에서만 적용할 수 있는 점을 문제로 지적하였다[2]. 이렇듯 최적화 전환은 서로 다른 운영환경, 개발환경, 데이터 처리 방식 등의 차이를 극복해야 한다. 특히 서비스는 소스코드와 DB를 따로 떼어놓을 수 없는데, 기존 연구들이 소스코드의 정적분석에 집중하고 있어 동적으로 동작하는 개발 프레임워크 환경에는 적합하지 않다. 하지만, DB 전환분야는 방법론외 자동화를 다루는 연구가 거의 없는데, DB는 대상 시스템의 비즈니스에 대한 이해 없이는 분석할 수 없어 범용화하기 더더욱 어려운 분야이다. L. R. Parvathy은 연구에서 DB를 클라우드로 전환에 단계적 전환이 가장 효율이 높다고 평가하였다[3]. 하지만, 데이터 정합성이 핵심인 DB는 호출 주체인 소스코드에 대한 고려 없이 단순 자료의 이전으로만 측정한 점에서 서비스 측면의 고려가 미흡하다 하겠다. M. S. Heo et al.은 DB 전환을 시스템화 하기위해 레파지토리 시스템을 제안하였다[4]. 하지만, 제안기법이 산출물을 DB화하는 관리 방법론적에 국한된 점은 아쉽다. S. C. Park et al.은 DB의 사용빈도에 따라 용도를 분류하는 머신러닝 기반 자동화 시스템을 제안하였다[5]. 제안방식은 DB 분류를 자동화하였으나, 도메인으로 세분화하지 못한 점은 아쉽다.

2.2 MSA 마이그레이션 자동화

모놀로식(Monolithic) 서비스의 MSA 전환은 서비스를 도메인 단위로 식별한 후 기능과 연관된 소스코드와 DB를 분리하여 컨테이너로 구성한다. 기존 연구들은 이 과정을 구문트리(AST: Abstract Syntax Tree) 같은 정적분석 방식에 집중하고 있다. M. Borges et al.은 재구조화를 AST로 기존 소스코드에 클라우드 공급자에 맞춘 어댑터 코드를 삽입하는 방식을 제안하였다[6]. 이 방식은 클라우드 공급자를 바꿀 때마다 재개발을 반복해야 하는 한계가 있다. A. Megargel et al.은 모놀로식 소스코드에 대응되는 기능 어댑터를 개발하여 점진적으로 교체하는 전환방식을 제안하였다[7]. 하지만, 이 방식은 완료 전까지 기존 환경과 클라우드 양쪽에 데이터가 분산되어 정합성 문제의 위험이 있다. Z. Cai et al은 특정문자열를 탐색하는 정규화 패턴 코드를 활용한 템플릿 기반을 제안하였다[8]. 이 방식은 소스코드를 분석 후 딕셔너리와 템플릿을 사전에 생성해야 하고, 특정소스에 맞춰져 범용화 하기 어렵다. A. Bucchiarone et al.은 통합개발환경의 애드온 기능으로 AST를 생성 후 MSA 적용가능 패턴을 탐색하여 추천하는 방식을 제안하였다[9]. 이 방식은 기존 소스코드를 AST로 작성하고 소스마다 패턴매치를 적용해야 하여 소규모 시스템에만 적합하며 매칭 패턴을 개발을 해야 하는 한계가 있다. M. Gysel et al.은 서비스의 경계분석을 위한 도구 연구에서 경로탐색 알고리즘을 통해 경로를 분석하고 결과를 표시하는 인터페이스를 제안하였다[10]. 이 방식은 전체 소스코드를 스캔하는 기존 방식을 이용해 결과를 GUI로 표시하는 방식으로 구성되어 있다. H. knoche et al.은 MSA 전환을 위한 파사드 방식의 단계적 이전을 제안하였다[11]. 하지만 이 방식도 점진적, 단계적 이전 방식으로 서비스 데이터 이원화 문제를 고려하지 못한 점은 아쉽다.

본 연구는 기존 연구와 달리 실시간 동적 분석을 통해 개발 프레임워크 환경에서도 적용가능하며, DB와 소스코드 간의 연관성을 실시간으로 도출하므로 전환 과정의 효율성을 높인다.

Ⅲ. 제안 기법

3.1 제안 시스템

본 연구는 기존 서비스를 클라우드의 MSA로 전환할 때 발생하는 도메인 경계 분석을 지원하는 동적 시스템 구현을 목적으로 한다. 개발 프레임워크는 사용자의 요청에 대해 WAS의 컨트롤러 단계에서 동적 파라미터를 전달한 함수호출이 일어나고 모델 단계에서 대상 테이블과 변수를 조합한 명령 쿼리를 동적으로 생성하여 DB를 호출한다. 제안 시스템은 그림 1과 같이 모델 단계에서 DB처리 이벤트가 발생하면 백트레이스 정보와 쿼리명령을 조합하여 분석에 사용한다.

백트레이스는 함수에 대한 호출스택을 배열로 관리하므로 호출 이력에 접근하는데 용이하고, 배열을 탐색하면 모든 호출 관계를 추출할 수 있어 분석의 정확도를 높이는데 효과적이다.

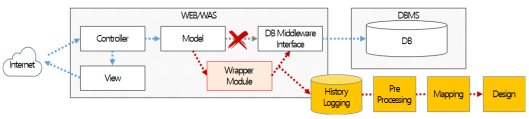

그림 1. 시스템 처리 흐름도
Fig. 1. System flowchart

3.2 개발모델

모델-뷰-컨트롤(MVC)은 강응집과 약연결을 위한 개발모델로 모놀로식에서 널리 사용된다. 과거의 개발 방식이 화면, 비지니스 로직과 DB처리가 하나의 소스코드에 모여있어 개선이 어려운 문제를 해소하기 위해 MVC 모델은 3-Tier 구조에 맞춰 그림 2처럼 각 역할에 맞춰 소스코드를 구분한 프레임워크다.

MSA의 도메인 주도 개발(DDD) 모델은 관련 기능을 묶은 역할 도메인을 정의하고 도메인 상호간 API G/W로 통신하는 어그리게이션 모델이다. 따라서, MSA로의 전환은 기존 시스템의 서비스 경계분석으로 역할 도메인과 연관된 리소스를 분해하여 재구성하는 배치모델로 아키텍트 재구조화가 필요하다. 이 과정이 고난이도로 고비용의 요인이 되어 기존 시스템의 MSA 전환 활성화의 장애가 되고 있다.

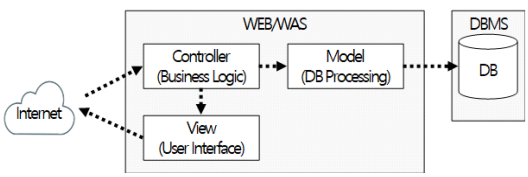

그림 2. MVC 모델의 구조
Fig. 2. Structure of the Model-View-Control Model

3.3 소스코드 스택 백트레이스(Stack Backtrace)

대다수의 개발언어는 실행중 발생하는 오류의 디버깅을 위해 호출스택(Call-stack)을 통한 스택 백트레이스 기능을 제공한다[12]. 백트레이스 정보에는 호출순서, 소스파일, 함수명, 소스코드 라인수, 호출 파라미터가 저장되어 있어, 알고리즘 1처럼 PHP의 deubg_backtrace() 내장함수를 호출하여 배열로 처리할 수 있다.

Algorithm 1 Backtrace Example in PHP Pseudo code

```
1: procedure VAR_BACKTRACE
2:     back_stack ← READ_FROM_STACK_FUNC()
3:     i ← 0
4:     for each node in back_stack do
5:         func ← node.function
6:         path ← node.file
7:         file ← BASENAME(node.file)
8:         line ← node.line
9:         Print [i++] file:func() path(line)\n
10:    end for
11: end procedure
```

3.4 DB 래퍼 클래스(Wrapper Class)

WAS에서 DB에 접근할때 DBMS에 제공하는 미들웨어 API를 사용하여 통신한다. Java의 JDBC와 PHP의 PDO는 개발언어에서 제공하는 범용 DB 인터페이스이며, Oracle의 OCI와 Mysql의 Mysqli등 특정 DBMS 전용 인터페이스도 있다. 개발 프레임워크는 DB 인터페이스의 래퍼클래스를 제작하여 DB 호출을 표준화하여 처리하는데, 제안 시스템은 이러한 호출과정을 활용하여 백트레이스와 SQL을 수집한다. 표 1은 호출이력 저장용 테이블 구조로 DB호출이 발생할 때마다 호출된 소스파일, 호출한 함수명과 대상 DB정보를 저장한다.

표 1. 호출이력 저장용 DB_TRACE 스키마
Table 1. Schema for DB_TRACE

Filed Name	Attribute	Comments
SEQ	int(11)	PK, Auto-Increment
Call_Table	varchar(128)	Callee Table-name
Call_Func	varchar(128)	Called Function name
Call_File	varchar(128)	File path of Function
Query	varchar(256)	Full text query string

Ⅳ. 구현 및 시험평가

4.1 시험 개요

본 연구의 성능시험은 오픈소스 CMS(Content Management System)를 사용하여 검증한다. WAS에서 DB를 처리하는 DB 래퍼 클래스를 수정하여 이벤트가 발생하면 백트레이스에 저장된 호출기록과 SQL문을 기록하도록 한 후 정적분석의 탐색결과와 비교하여 성능을 평가한다.

그림 3은 본 시험에서 사용한 이벤트 기록을 위한 흐름을 나타낸다. 웹서버는 사용자의 이벤트가 발생하면 WAS에서 비즈니스 로직에 맞추어 동적 코드가 생성되고 이를 처리하기 위해 DB에 자료 입출력을 요청한다. 시험은 DB를 처리하는 단계에서 백트레이스 정보와 쿼리문을 조합하여 호출이력을 구성한다.

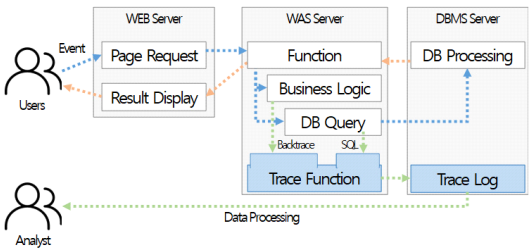

그림 3. 시험환경 데이터처리 흐름도
Fig. 3. Data processing flowchart

4.2 시험환경 구성

시험환경은 표 2와 같이 가상서버 환경에 우분투 리눅스를 설치 후 웹 서비스를 구성하였다.

표 2. 시험환경
Table 2. Test environment

Type	Spec.
Host	Intel Core i7 2.10 GHz / 16GB
Host O/S	Windows 11 Pro 23H2
Guest	Windows Hyper-V 8 Core / 4 GB
Guest O/S	Ubuntu 22.04 TLS

4.3 콘텐츠 관리시스템(CMS)

CMS는 웹 서비스의 생명주기 관리 솔루션으로 개발의 편의성과 확장의 용이성을 지원하기 위해 관련 기능을 추상화하여 제공하는 개발 프레임워크다. 본 연구에서 사용한 오픈소스 CMS인 워드프레스는 다양한 애드온의 조합으로 쇼핑몰과 같은 비지니스 사이트와 프로젝트관리, 인력관리 등 업무 사이트 구축에 널리 활용되

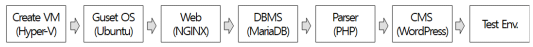

그림 4. 시험환경 환경구성 흐름도
Fig. 4. Test environment configuration flow

고 있다. 워드프레스는 그림 4와 같은 순서로 설치한 LEMP(Linux, NginX, Mysql, PHP) 오픈소스 기반 소프트웨어 스택 환경이 필요하다.

워드프레스의 애드온인 쿼리모니터(QM: QueryMonitor) 플러그인은 백트레이스를 활용하여 실시간 디버깅을 지원한다. 그림 5는 시험환경에 구성된 QM db 클래스와 wpdb 클래스의 의존관계를 나타낸다.

알고리즘 2는 호출 이력 저장을 위해 추가한 소스코

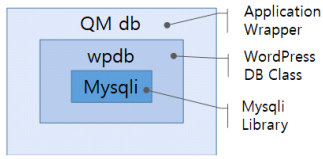

그림 5. 시험환경의 DB Wrapper 구조
Fig. 5. Structure of the DB Wrapper for test

Algorithm 2 Store the combination of Backtrace and Table to db_trace

```
procedure LOG_BACKTRACE_TO_DB(sql)
    back_trace ← DEBUG_BACKTRACE()                    ▷ backtrace to array
    function ← ""
    file ← ""
    table ← EXTRACT_TABLE_FROM_SQL(sql)               ▷ Table from SQL
    for each node in back_trace do
        if node.function ≠ NULL and node.file ≠ NULL then
            count ← count + 1
            if count = 3 then
                function ← node.function
                file ← node.file
                break                                 ▷ Stop after finding the third value
            end if
        end if
    end for
    format ← "INSERT INTO db_trace
              (Call_Table, Call_Func, Call_File, Query)
              VALUES ('%s', '%s', '%s', '%s')"
    query ← SPRINTF(format, table, function, file, sql)
    results ← wpdb.execute(query)                     ▷ Insert data into db_trace
    return results
end procedure
procedure EXTRACT_TABLE_FROM_SQL(sql)
    from_position ← STRPOS(sql, "FROM")
    if from_position = NULL then
        return "ERROR"
    end if
    table ← EXTRACT_WORD_AFTER(sql, from_position)
    return table
end procedure
```

표 3. CMS에 설치한 시험용 플러그인
Table 3. CMS Plug-in for test

Functions	Add-On
WebSite	Hello Dolly
e-Commerce Site	WooCommerce
PMS Site	Zephyr Project Manager
Monitor	QueryMonitor

드로 이력정보를 DB에 저장하는 역할을 한다. 기존 소스코드에서 DB처리가 성공한 후 백트레이스 정보를 구하고 쿼리문에서 목적 테이블 이름을 얻어내어 db_trace 테이블에 저장한다.

시험에서는 애드온 환경에서의 동작 시험을 위해 표 3과 같이 구성하였다. 모든 DB를 처리하는 워드프레스의 class-wpdb.php 파일에 백트레이스 이력 코드를 삽입하여 수집하였다. 관련된 소스코드는 저자의 깃허브[13]에서 확인할 수 있다.

4.4 데이터 수집

구성한 웹사이트의 페이지 호출을 통해 이력정보를 저장하여 그림 6과 같이 약 7천 5백건의 호출이력 데이터를 저장하였다.

이력 DB에 저장된 데이터에는 그림 7과 같이 표 1에서 정의한 테이블명, 함수명, 파일명이 순서대로 저장되어 있다.

```
MariaDB [wordpress]> select count(*) from db_trace;
+----------+
| count(*) |
+----------+
|     7572 |
+----------+
1 row in set (0.001 sec)
```

그림 6. 시험데이터 데이터 건수
Fig. 6. Data count for testing

seq Sequence Number	Call_Table Table Name	Call_Func Call Function Name	SUBSTRING(Call_File, 14, 50)
1	wp_posts	get_row	/wp-includes/class-wp-post.php
2	wp_posts	get_instance	/wp-includes/post.php
3	wp_posts	get_post	/wp-content/plugins/woocommerce/includes/class-wc
4	wp_posts	register_post_types	/wp-includes/class-wp-hook.php
5	wp_posts	apply_filters	/wp-includes/class-wp-hook.php
6	wp_posts	do_action	/wp-includes/plugin.php
7	wp_posts	do_action	/wp-settings.php
8	wp_posts	require_once	/wp-config.php
9	wp_posts	require_once	/wp-load.php
10	wp_posts	require_once	/wp-blog-header.php
11	wp_posts	require	/index.php
12		get_var	/wp-includes/class-wp-query.php
13		set_found_posts	/wp-includes/class-wp-query.php
14		get_posts	/wp-includes/class-wp-query.php
15		query	/wp-includes/class-wp.php
16		query_posts	/wp-includes/class-wp.php
17		main	/wp-includes/functions.php
18		wp	/wp-blog-header.php
19		require	/index.php
20	wp_posts	get_results	/wp-includes/post.php
21	wp_posts	_prime_post_caches	/wp-includes/class-wp-query.php
22	wp_posts	get_posts	/wp-includes/class-wp-query.php
23	wp_posts	query	/wp-includes/class-wp.php

그림 7. 시험데이터 데이터 저장현황
Fig. 7. Recorded data list

4.5 데이터 전처리

데이터는 판다스(Pandas)로 전처리를 수행하였다. DB를 포함한 연관분석을 위해 DB호출이 포함되지 않은 함수간의 호출은 대상에서 제외하였다. 또한, 개발 프레임워크는 공통 정보를 환경DB에 저장 후 호출시 DB로부터 읽어 동적 코드를 생성하는 특성이 있어 다수의 중복데이터가 발생한다. 이런 중복데이터도 제거 후 표 4와 같이 분석 데이터를 선별하였다.

표 4. 결측 및 중복제거가 데이터 현황
Table 4. Data after preprocessing

Total Count	Shortage	Duplication	Left
7,572	126	7,281	165

4.6 데이터 연관분석

DB 저장 데이터는 가공하는 방식에 따라 함수와 DB의 호출 연관관계를 분석할 수 있다. 알고리즘 3은 DB에 저장된 정보를 조회하는 결과조회 코드이다. 소스코드의 함수명이나 DB명을 선택하면 연관된 결과를 조회할 수 있다. 이 경우 전체가 중복된 결과는 제거하기 위해 중복제거 명령을 사용한다. 이러한 방식으로 소스코드와 DB의 연관을 정확히 식별할 수 있음을 증명하였다.

결과 데이터는 그래프 분석도구로 가시화도 가능하다. 그림 8은 사각형 DB를 중심으로 연관된 함수명를 연결한 네트워크 그래프다. 네트워크 그래프 결과화면은 저자의 깃허브에서 확인 가능하다.

Algorithm 3 Search by Function or Table

```
Prompt the user to choose either [By function] or [By table] and input the Target name.
if selection = function then
    SELECT DISTINCT Call_Func, Call_Table, Call_File
    FROM db_trace
    WHERE Call_Func = target_name;
    Display the results in the order: Call_Func, Call_Table, Call_File.
else if selection = table then
    SELECT DISTINCT Call_Table, Call_Func, Call_File
    FROM db_trace
    WHERE Call_Table = target_name;
    Display the results in the order: Call_Table, Call_Func, Call_File.
end if
```

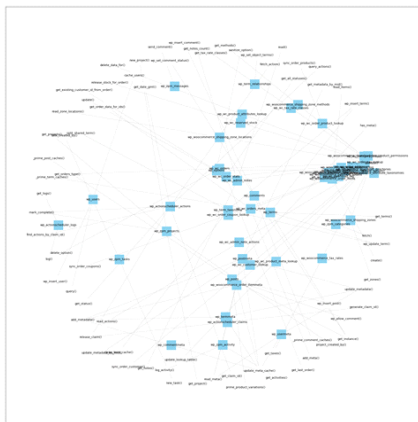

그림 8. 연관관계의 네트워크그래프
Fig. 8. Association analysis network graph

4.7 성능비교

기존 연구에서 제안된 정적분석 기법과 본 연구의 제안시스템을 비교하여 성능을 평가하였다. 시험 대상은 표 5에 제시된 오픈소스 CMS의 PHP 언어로 작성된 웹서비스용 소스파일 약 3천4백개를 분석대상으로 하였다.

표 5. 성능비교 대상 파일 수
Table 5. Number of Files for Performance Comparison

Types	Count
Total	18,842
PHP	9,957
Service Page	(3,428)
Admin Page	(6,529)
Others	8,885

성능시험 결과는 표 6과 같이 나타났다. 정적분석 기법은 소스코드를 탐색하여 82개의 SQL 구문이 탑재된 소스를 검출하였다. 하지만, 프레임워크는 실시간 동적으로 파라미터를 생성하는 코드 재사용성 특성이 있어 DB까지는 분석할 수 없다. 이에 비하여 제안시스템은 약 18% 더 많게 DB를 사용하는 소스파일을 검출하였는데, 이는 개발 프레임워크의 동적코드 생성 특성까지 반영된 결과이다. 표 6의 SQL호출처럼 한 개의 소스파일에서 평균 18번의 동적으로 DB를 사용함을 알 수 있다.

성능시험을 통해 제안 시스템이 소스파일과 함수간의 호출관계와 연관된 DB까지 포함하여 분석하는데 더 효율적임을 알 수 있다. 또한, 개발 프레임워크의 동적 코드 생성 특성까지 반영하여 소스와 DB간의 연관성을 더 많이 검출할 수 있었다. 이러한 결과로 제안 기법이 서비스 경계 분석의 목적에 더 적합한 시스템임을 증명하였다.

표 6. 성능비교 결과표
Table 6. Performance Comparison Results Table

Method	SQL-Utilizing Files	SQL Calls
Static Code Analysis	82	-
Proposal	97	1,795

V. 결 론

본 연구는 모놀리식 시스템을 클라우드 네이티브 환경의 MSA로 전환할 때, 필수적인 도메인 경계 분석을 자동화하기 위한 연구이다. 기존 연구의 정적분석 기반 자동화가 가지고 있는 특정 개발환경 전용으로 인한 범용성 결여, 개발 프레임워크 미지원 문제에 대하여, 실시간 웹서비스 이벤트로부터 호출을 추적하여 분석하는 하향식 동적분석 방식의 해결방법을 제안하였다. 성능시험 결과처럼 정적분석보다 18% 더 높은 정확도를 나타냈으며, 개발 프레임워크에 관계없이 적용가능함을 증명하였다. 분석결과도 소스파일, 함수와 함께 DB까지 포함하여 서비스와 연관된 모든 자원을 통합적으로 분석할 수 있음이 확인되었다.

제안된 시스템은 분석 전용 환경구축이나 사전 메타데이터도 요구되지 않아 더 신속하고 효율적이다. 또한, 외부 전문가보다 기존 시스템에 대한 이해도 높은 내부 전문가로도 분석할 수 있도록 하는 전환비용 절감으로, 실질적인 클라우드 최적화 전환에 효과적으로 활용될 것으로 기대할 수 있다. 다만, 호출 관계의 시각화에 대규모 데이터 표현의 한계가 있으므로, 복잡한 호출 관계를 시각화하는 추가 연구가 필요하다.

References

[1] M. H. Hasan, M. H. Osman, N. I. Admodisastro, and M. S. Muhammad, “Legacy systems to cloud migration: A review from the architectural perspective,” *J. Syst. and Software*, vol. 202, Apr. 2023. (https://doi.org/10.1016/j.jss.2023.111702)

[2] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, “From monolith to microservices: A classification of refactoring approaches,” *Software Eng. Aspects of Continuous Development and New Paradigms of Software Production and Deployment (DEVOPS 2018)*, vol. 11350, pp. 128-141, Jan. 2019. (https://doi.org/10.1007/978-3-030-06019-0_10)

[3] L. R. Parvathy, “A comprehensive analysis of cloud migration strategies: Efficiency comparison of trickle, big bang, refactoring, lift and shift, replatforming approaches,” *2023 9th Int. Conf. Smart Structures and Syst. (ICSSS)*, pp. 1-7, Nov. 2023. (https://doi.org/10.1109/ICSSS58085.2023.10407074)

[4] M. S. Heo, D. S. Kim, and H. W. Kim, “Data conversion automation tool based on re-

pository and processes," *J. Service Research and Studies*, vol. 10, no. 2, pp. 17-29, Jun. 2020.
(https://doi.org/10.18807/jsrs.2020.10.2.017)

[5] S. C. Park and Y. H. Kim "A design and implementation of automated classification and analysis system using K-Means clustering for transition of database to micro-services-architecture," *J. KICS*, vol. 49, no. 9, pp. 1306-1314, Sep. 2024.
(https://doi.org/10.7840/kics.2024.49.9.1306)

[6] M. Borges, E. Barros and, Paulo Henrique Maia, "Cloud restriction solver: A refactoring-based approach to migrate applications to the cloud," *Inf. Software Technol.*, vol. 95, pp. 346-365, Feb. 2018.
(https://doi.org/10.1016/j.infsof.2017.11.014)

[7] A. Megargel, V. Shankararaman, and D. K. Walker, "Migrating from monoliths to cloud-based microservices: A banking industry example," *Software Eng. in the Era of Cloud Comput.*, pp. 85-108, Jan. 2020.
(https://doi.org/10.1007/978-3-030-33624-0_4)

[8] C. Zhengong, Z. Liping, W. Xinyu, Y. Xiaohu, Q. Juntao, and Y. Keting, "A pattern-based code transformation approach for cloud application migration," *IEEE 8th Int. Conf. Cloud Comput.*, pp. 33-40, Aug. 2015.
(https://doi.org/10.1109/CLOUD.2015.15)

[9] A. Bucchiarone, K. Soysal, and C. Guidi, "A model-driven approach towards automatic migration to microservices," *Software Eng. Aspects of Continuous Development and New Paradigms of Software Production and Deployment (DEVOPS 2019)*, vol. 12055, pp. 15-36, Jan. 2020.
(https://doi.org/10.1007/978-3-030-39306-9_2)

[10] M. Gysel, L. Kölbener, W. Giersche and, O. Zimmermann, "Service cutter: A systematic approach to service decomposition," *Service-Oriented and Cloud Comput. (ESOCC)*, vol. 9846, pp. 185-200, Aug. 2016.
(https://doi.org/10.1007/978-3-319-44482-6_12)

[11] H. Knoche and W. Hasselbring, "Using microservices for legacy software modernization," *IEEE Software*, vol. 35, no. 3, pp. 44-49, Jun. 2018.
(https://doi.org/10.1109/MS.2018.2141035)

[12] WIKIPEDIA, *Stack-trace*, Retrieved Nov. 15, 2024, from https://en.wikipedia.org/wiki/Stack_trace

[13] service-boundary-analysis, *Dynamic Service Boundary analysis with Backtrace*, Retrieved Nov. 18, 2024, from https://github.com/sun-iron/service-boundary-analysis

박 선 철 (Sun-chul Park)

2022년 : 숭실대학교 공학석사
2023년~현재 : 숭실대학교 일반대학원 인공지능IT융합학과 박사과정
<관심분야> 인공지능, 클라우드, 데이터센터, 정보시스템 분석 및 설계
[ORCID:0000-0003-1838-974X]

김 영 한 (Young-han Kim)

1984년 : 서울대학교 졸업
1986년 : 한국과학기술원 공학석사
1990년 : 한국과학기술원 공학박사
1994년~현재 : 숭실대학교 전자정보공학부 교수
<관심분야> 차세대 인터넷 프로토콜, 이동/무선 네트워크, 인터넷 텔레포니, 센서/모바일 애드혹 네트워크