정보 산업 동향

# 분산환경에서의 효율적인 균형을 위한 Service Operation Factory 구현

목 차



홍 민 석
(누리어시스템)

## 1. 서 론

SOA는 현재 많은 회사에서 적용되고 있으며, 다양한 구조로 설계되어 있다. 대부분의 어플리케이션은 각 담당업무와 담당프로세스를 가지고 있으며, 해당 프로세스를 원활하게 완료하는데 목적이 있었다[1]. 비즈니스가 커지고, 해당업무가 방대해지며, 공통되는 요소가 늘어남에 따라 SOA를 도입하여, 회사의 IT를 발전시키고자 하는 노력이 시작되었다. 또한 B2B의 인터페이스 및 상호 다른 플랫폼에서의 데이터 인터페이스도 SOA가 담당하게 되었다[2].

이런 SOA를 사용함에 따라 하나의 어플리케이션에서만 사용하지 않고 다수의 어플리케이션과 다수의 SOA서버로 운영하게 되었다. 더욱이 물리적인 위치가 다르며, 분산환경에서의 어플리케이션서버와 SOA서버의 효율을 향상시켜야 하는 문제가 발생하게 되었다.

다수의 어플리케이션과 SOA를 사용함에 따라 기존 방식에서는 어플리케이션에서 단순히 SOA서버의 Action을 요청/응답하는 방식이었다. 즉 하나의 어플리케이션은 하나의 SOA서버와 통신을 하며, 비즈니스 및 프로세스를 진행하였다.

이런 환경에서의 운영은 복합적이고 환경이 다른 상황에서 프로세스의 경쟁이나 프로세스 역행이 일어나게 되었으며 어플리케이션에서 요청을 보낸 SOA서버가 다운되면, 더 이상 진행이 안되는 문제점을 가지게 되었다. 현황을 제어하고, 보다 나은 방식으로 개선하기 위해 새로운 연구가 필요하게 되었다. 본 논문에서는 다수의 어플리케이션과 다수의 SOA서버를 운영함에 있어 기존 방식보다 개선된 Service Operation Factory (SOF)를 사용하여 비즈니스의 흐름을 효율성 있게 변화하고, 어플리케이션과 SOA서버의 요청/응답 방식을 변경하고자 한다.

SOF(Service Operation Factory)는 어플리케이션과 SOA 사이에서 효율적인 균형을 위한 구현을 말한다. 기존 서버구성도와 동일하며, 어플리케이션과 SOA 사이에서 운영된다. 어플리케이션이 SOF서버에 프로세스를 요청하면, SOF는 SOA의 현황을 체크한다. 기준은 쓰레드, 메모리, 응답시간이다. 이런 SOA의 현황에서 어플리케이션이 요청한 중요도를 적용하고, SOA를 선택하여 프로세스를 호출한다.

## 2. 관련 연구

### 2.1 커뮤니케이션 팩토리

SOF에서 사용하는 팩토리의 기술을 고려하기 위하여 팩토리 연구 중 하나인 커뮤니케이션 시스템의 가상 팩토리 기술을 살펴 보겠다. 가상 팩토리 는 네트워크 상에 팩토리 머신을 두어 커뮤니케이션을 핸들링하는 것이다. 가상머신 환경은 네트워크 상의 제어 통제를 함으로 데이터의 흐름을 원활하게 한다. MMS 와 MMS-CS를 사용하였다[6]. 팩토리를 사용했을때의 장점으로는 관계도, 유연성, 친화적인 인터페이스가 증가했다. 모든 네트워크의 가상 머신들은 작동 시나리오에 따라 스케줄되어 동작하게 된다. 모든 데이터는 가상 팩토리로 정보를 전달한다[6].

팩토리의 일부분으로 자바에서 팩토리를 사용하면 다음과 같은 장점이 있다. 팩토리를 쓰면 객체 생성을 캡슐화 할 수 있다. 간단한 팩토리는 엄밀하게 말해 디자인 패턴은 아니지만, 클라이언트와 구상 클래스를 분리시키기 위한 간단한 기법으로 활용할 수 있다. 팩토리 메소드 패턴에서는 상속을 활용한다. 객체 생성이 서브클래스에게 위임된다. 서브클래스에서는 팩토리 메소드를 구현하여 객체를 생산한다. 추상 팩토리 패턴에서는 객체 구성을 활용한다. 객체 생성이 팩토리 인터페이스에서 선언한 메소드들에서 구현된다. 모든 팩토리 패턴에서는 애플리케이션의 구상 클래스에 대한 의존성을 줄여줌으로써 느슨한 결합을 도와준다. 팩토리 메소드 패턴에서는 어떤 클래스에서 인스턴스를 만드는 일을 서브클래스한테 넘긴다. 추상 팩토리 패턴은 구상 클래스에 직접 의존하지 않고도 서로 관련된 객체들로 이루어진 제품군을 만들기 위한 용도로 쓰인다. 의존성 뒤집기 원칙을 따르면 구상 형식에 대한 의존을 피하고 추상화를 지향할 수 있다. 팩토리는 구상 클래스가 아닌 추상 클래스/인터페이스에 맞춰서 구현할 수 있게 해 주는 강력한 기법이다.

### 2.2 서비스지향 아키텍처

서비스 지향 아키텍처에서는 보다 다양한 서비스 및 애플리케이션에서 추적성을 고려해야만 하며 이를 통해 서비스간의 상관성을 분석해야만 변경에 대한 영향을 쉽게 도출해낼 수 있고 새로운 서비스를 구축할 때도 보다 빠르게 원하는 형태의 서비스 모형을 구축해 볼 수 있을 것이다[5]. 서비스는 발견이 가능 하고 필요로 하는 서비스를 찾고, 그것을 사용할 수 있어야 한다. 때문에 서비스는 비즈니스 요구사항과 그 하부의 요구사항들에 의해 발견이 가능해져야 한다[5].

웹 서비스 및 SOA 구현에 적용하는 모든 우수 사례들은 처음에 B2B 통신을 구현하는데 사용되었던 것으로 생각된다. 일부 사람들은 서비스를 중요한 부분의 분리를 이행하는 모듈로 정의하는 것 즉, 느슨한 연결(loose coupling)과 재사용에 집중하기도 하지만 모듈화, 객체 또는 컴포넌트 기반의 개발에 있어서 SOA와 이전 방식은 차이가 있다.[7] SOA는 이들 모듈이 서로 관련이 없는 다수의 개발 팀에 의해 개발, 관리, 전개 및 유지 보수될 수 있도록 설계된 것이다. 때로 이들 개발팀은 조직 내의 서로 다른 부서 또는 사업단위가 되기도 한다. 그렇지만, 이러한 다수의 개발팀은 다른 조직의 일부인 경우도 종종 있다. 단일포장된 애플리케이션의 공급업체일 수도 있고 다수의 애플리케이션을 제공한 시스템 통합 업체일 수도 있다. 또한 레지스트리는 서비스를 수월하게 통제할 수 있도록 해주며, 서비스의 관리와 재활용을 가능하게 해준다. 따라서 강력하고 진보한 서비스 레지스트리를 배포할 수 있느냐에 따라 SOA 효율이 결정될 것으로 보인다.

### 2.3 선행연구 개선점

분산환경과 같은 비즈니스로직을 수행하는 어플리케이션과 SOA서버상의 팩토리 기법과 통신흐름의 제어에 대한 기법이 더 요구되고 있는 환경이다. 본 논문에서 이야기 하고자하는 환경과는 다소 상이하다. 이러한 케이스에 대하여 다른 방식이 필요하다.

더 복잡하고 지역적으로 떨어져 있으며 프로세스의 흐름에 영향을 적게 줄 수 있는 방법을 생각하고, 또한 안정성까지 고려할 필요가 있게 되었다. SOA는 같은 프로세스를 수행하는 동일한 서버가 존재하며, 각 SOA의 응답시간을 체크하여 해당 SOA의 상태를 알아야 하고 다른 서버와의 상태를 비교하기 위하여 SOA 서버 자체의 여러 현황을 별도로 저장하여야 한다. 파일기반으로 XML을 작성하는 방법과 해당 파일이 Properties로 변화되어서 사용되는 방법도 있다. 해당 XML을 Properties와 자유롭게 변환되어 사용되어지도록 해야 한다. 선행연구의 팩토리에서 사용하는 것은 데이터 및 메시지를 통합하지만 어플리케이션과 SOA의 프로토콜 사이에 흐름을 제어할 수 있는 팩토리가 요구 되어 진다.

## 3. S O F 구현

### 3.1 어플리케이션 구성

서울, 부산, 대전의 지사들은 각각의 어플리케이션을 가지고 있으며, 해당 DB로 액세스하며, SOA로 공통된 프로세스를 호출한다. 지사의 어플리케이션의 사양은 비슷하며, 공통된 내용으로 되어 있다. SOF는 서울에 위치해 있으며, 각 지사와 SOA서버와의 통신을 담당하게 된다. Linux로 되어 있으며, 대용량 RAID로 스토리지는 연결되어 있다. 다른 시스템과의 간섭을 피하기 위해 독립된 서버를 단독으로 사용하며, 대역폭 또한 많은 비중으로 할당 되어 있다. 대부분 2cpu의 4core로 되어 있다. 어플리케이션 및 SOA를 실행하기에는 무리가 없는 권장사양으로 되어 있으며, 자동백업등 서버관리 프로세스를 갖추고 있다.

### 3.2 통신흐름 제어

SOF의 가장 큰 원리는 데이터 통신의 흐름을 제어하는 것이다. 지사의 어플리케이션 HTTP Request를 받아서, SOA의 SOAP로 통신하며, 통신한 결과값을 다시 XML 형태로 만들어서 지사 어플리케이션에 HTTP로 보낸다.

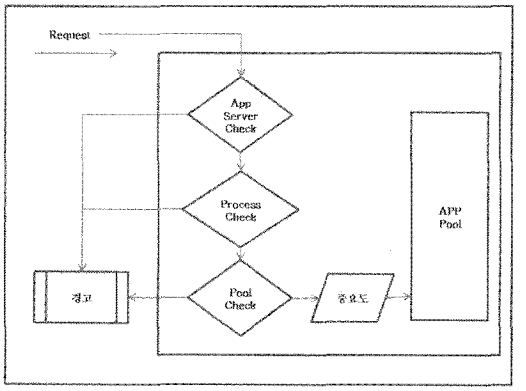
어플리케이션에서 http://1.1.1.1/wsae/ping.action 이라는 url을 호출하게 된다. ping.action 이라는 정보를 was의 proxy에서 찾는다. 이는 처음에 로드되는 정보이며, web path와 id로 구분되어 진다. ping과 연관된 정보를 proxy에서 찾은다음 실제 요청하는 클래스로 보낸다. proxy에서는 넘어오는 파라미터의 값을 한번 더 필터링 하고, 필터링을 통과한 데이터만 다음 단계로 넘기게 된다. 파라미터가 없거나 정보가 틀리다면 WAS는 잘못된 정보라고 표시를 하게된다. proxy 와 파라미터 검증을 끝내고 , 다음 단계로 진행하게 된다. 이렇게 처음에 접근하는 방식은 앞에서 말한 HTTP 방식으로 접근한다. HTTP의 Post로 요청한다. 요청을 받은 SOF는 다음장의 알고리즘 단계를 거쳐 적절한 SOA서버를 찾게 된다. 서버의 기본정보를 가지고 SOF는 프로세스, 파라미터를 전송하게 된다. 이때 사용하는 방식은 앞에서 말한 SOAP로써 Axis2를 사용하게 된다.

### 3.3 App Pool - SOA Pool

APP Pool이란 SOF에서 각 지사 어플리케이션에서 요청한 프로세스를 저장하는 곳이다. 저장되는 순서대로 처리가 되며, 최소 1개부터 최대 300개까지로 셋팅이 되어 있다. 300개가 넘는

경우 대기 상태로 전환하며, 이후에 요청한 쓰레드가 50개가 넘으면 관리자에게 경고 메시지를 이메일과 핸드폰으로 보내게 된다. APP Pool은 싱글톤으로 생성되며, SOF WAS 중 하나의 인스턴스를 가지게 된다. 총 3개의 Hashtable을 가지며 처리해야 할 프로세스 , 대기중 프로세스, 재시도 해야할 프로세스로 저장하게 된다. 2개의 Boolean을 가지며 현재 run 중인지 체크하는것과 에러를 체크하는 항목이다. 2개의 java.util.Vector를 가지며 완료한 비즈니스 프로세스 정보와 에러 발생한 비즈니스 프로세스 정보이다.

(그림 1)과 같이 지사 어플리케이션의 요청이 들어오면, 해당 어플리케이션의 인증정보, 기본정보를 체크하고, 요청한 프로세스의 정보를 확인하고 검증하고 Pool의 상태를 체크, 검증한 후에 중요도 순서대로 Pool에 저장하게 된다.




(그림 1) APP Pool 저장순서

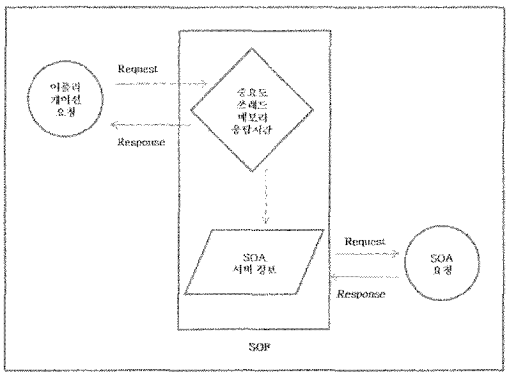
SOA Pool은 SOA 서버의 정보를 저장하고 있으며, 가장 원활한 서버가 최상위에 존재하게 된다. SOA Pool은 싱글톤[3]으로 생성되며, SOF WAS 중 하나의 인스턴스를 가지게 된다. Pool에는 4개의 Hashtable이 존재 하게 되며 서버현황 정보, 서버의 연결정보 비즈니스 호출목록, 수행해야 할 목록, 재시도 목록을 가지고 있다. Hashtable중 하나인 서버현황 정보는 각 SOA 서버의 상태를 저장하고 있으며, 최초 생성시 정보가 없으면 Properties에 등록된 서버 순서대로 정보를 수집하게 된다. 수집되는 정보는 각 SOA 서버의 쓰레드, 메모리, 응답시간이다. 서버의 연결정보 와 비즈니스 호출목록은 각 서버의 아이피, 포트 이며, 호출목록은 클라이언트에서 사용해야 할 프로세스 Action 명이다. 수행해야 할 목록은 지사에서 요청한 프로세스를 서버현황에서 체크해서 원활한 서버로 연결해서 Response를 받기 위해 대기중인 프로세스 목록이다. 최대 300개이며 300개초과시 50개의 대기 프로세스를 가진다. 50개의 대기가 넘게되면 관리자에게 이메일과 핸드폰으로 경고메시지를 전달하게 된다. 재시도 목록 Hashtable은 SOA서버로 요청했을 경우 재시도 목록으로 들어가게 되며, 3번의 재시도에도 에러가 발생할 경우, 로그에 기록하고 에러로 진행하게 된다.

### 3.4 분석 및 수행

SOF는 각 지사 어플리케이션에서 요청하는 정보를 수집한다. 요청된 Request에서 중요도를 수집하고, SOA 서버의 상태를 점검한다. SOA 서버의 상태를 점검하는 방법은 특정포트와 특정패스로 연결된 방법으로 상태정보를 서로 동기화 한다. 데이터의 비교로 어느 서버가 요청을 받아야 하는지 결정하게 된다. 합계로 우위를 결정하며, 같을 경우 설정된 서버의 우선순위로 결정한다. 우선순위 항목은 중요도, 쓰레드, 메모리, 응답시간이 있으며, 앞의 순서대로 중요하다.

각 SOA 서버별 체크항목으로는 서버에 대한 응답시간은 3초로 제한되어 있으며, SOA Library는 AXIS2 1.4.1을 사용하며, 호출용 ping Action(pingTest)을 사용하며, 세션의 아이디를 가진다. 각 서버별로 AXIS2 버전을 체크하고 ping을 확인한다. 이상이 없으면 세션아이디를 공유하게 된다. 각 서버별 정보를 SOF는 수집하게 된다.

어플리케이션 서버에서 요청한 Request와 각 서버별 체크항목으로 요청한 프로세스를 진행하게 된다. 어플리케이션이 요청한 시간과 응답한 시간을 기록하여 통계데이터로 SOF는 가지게 된다. SOA 서버에 전달한 프로세스와 시작시간, 완료시간을 통계데이터로 기록한다. 어플리케이션 통계데이터 SOA 통계데이터는 각각의 안정성을 확보하는 데이터로 활용된다.




(그림 2) 요청처리 프로세스

## 4. 실험 결과 및 분석

### 4.1 실험 환경

테스트를 위한 비즈니스 프로세스는 50개의 프로세스 중 사용빈도가 높은 프로세스를 기준으로 한다. 총 10개의 프로세스를 선별하였다. 선별된 프로세스명은 airAvail, airBookModify, airBookTps, airCommon, airDetail, airFlightInfo, airSchedule, airCancel, airQPlace, airReadTps 이다. 해당 10개의 프로세스로 하여 범위를 정한다. 각 지사의 어플리케이션의 요청도 위와 같은 프로세스의 범위이며, 각 SOA 서버의 Part Action도 위와 같은 프로세스로 정한다.

테스트용 데모 SOA 서버는 tomcat 이며, 총 4개의 데모용 SOA 서버가 구동된다. tomcat의 버전은 5.5.27이고 포트는 35600, 35700, 35800, 35900 이다. 데모용 SOA 서버의 프레임워크는 hibernate3, Axis2이며 기본 라이브러리를 가지고 있다. 각각 SOA서버마다 위에서 정의한 프로세스를 실행하기 위한 Axis2의 설정파일을 가지고 있으며 tomcat의 SOAP의 서비스에 올라가 있다. 구동시 해당 SOA 서비스는 자동 Deploy 되며, 서비스의 이상유무를 확인 할 수 있다. 사용메모리는 1GB씩 총 4GB를 사용한다. 실제 데이터베이스의 연동은 하지 않으며 해당 서비스의 호출에 따른 로직을 수행하고 끝내는 방식으로 하겠다. 데모용 SOF는 tomcat을 사용하며 버전은 5.5.27 이다. 사용 포트는 35100 이며 사용메모리는 1GB 이다. 사용프레임워크는 struts2, axis2를 주요 프레임워크로 사용하며 기본적인 라이브러리를 사용한다. 각 SOA서버의 클라이언트를 라이브러리로 가지고 있으며, 4개의 endpoint를 가진다. SOA에 요청하는 Part Action은 위에서 정의한 프로세스이다. Request Response는 모두 Log4j로 기록되며, 통계데이터로 해당 프로세스의 시작시간 완료시간을 측정하게 되며 리소스로 저장하게 된다.

### 4.2 실험 실행

기존방식으로 5번 SOF 방식으로 5번 테스트를 수행하게 된다. 아래 〈표 1〉과 같이 총 5번을 실행하며 동시사용자, 딜레이, 반복회수 에 대하여 변경하여 테스트를 수행한다.

〈표 1〉 실험 실행을 위한 변수

| | 기존방식 | | | SOF방식 | | |
|---|---|---|---|---|---|---|
| | 동시 사용자 | 딜레이 | 반복 회수 | 동시 사용자 | 딜레이 | 반복 회수 |
| 1 | 5 | 1 | 1 | 5 | 1 | 1 |
| 2 | 10 | 1 | 5 | 10 | 1 | 5 |
| 3 | 10 | 1 | 10 | 10 | 1 | 10 |
| 4 | 20 | 1 | 5 | 20 | 1 | 5 |
| 5 | 20 | 3 | 10 | 20 | 3 | 10 |

테스트용 어플리케이션에서 먼저 기존방식으로 위의 〈표 1〉과 같이 값을 셋팅한 후 순서대로

클릭한다. 그러면 테스트용 어플리케이션은 지사의 데모용 어플리케이션에 해당 값으로 요청을 한다. 기존방식은 지사의 어플리케이션에서 바로 SOA 서버로 요청하며 SOF방식은 지사의 어플리케이션이 SOF에게 요청하고 SOF는 각 SOA서버에게 요청하게 된다.

### 4.3 지표 검증

아래 〈표 2〉는 총 5번씩 테스트 후 데이터의 통계이다. 각 4개의 지사 어플리케이션의 데이터를 수집하고 통계를 한 데이터 이며 기록된 쓰레드, 메모리, 응답시간의 통계이다. 단위는 쓰레드는 개수 , 메모리는 MB , 응답시간은 초이다.

〈표 2〉 실험 완료 후 통계데이터

| | 기존방식 | | | SOF방식 | | |
|---|---|---|---|---|---|---|
| | 쓰레드 | 여유 메모리 | 응답 시간 | 쓰레드 | 여유 메모리 | 응답 시간 |
| 1 | 81ea | 238mb | 8.3초 | 84ea | 240mb | 8.3초 |
| 2 | 147ea | 222mb | 10.3초 | 138ea | 227mb | 8.4초 |
| 3 | 150ea | 217mb | 11.8초 | 147ea | 223mb | 9.6초 |
| 4 | 182ea | 200mb | 21.9초 | 148ea | 214mb | 17.3초 |
| 5 | 238ea | 150mb | 29.1초 | 190ea | 172mb | 22.7초 |

〈표 2〉에서 볼수 있듯이 기존방식 대비 SOF 방식은 응답시간 19%, 여유메모리 5%, 쓰레드 12%가 향상되었다. 5번의 테스트를 하고 그에 대한 통계를 기록한 것이다. 쓰레드수는 계속 증가하고 메모리는 계속 감소하며, 응답시간의 기준은 어플리케이션의 시작시간과 끝시간을 기록한 결과이다. 실제 사용자가 인지할 수 있는 시간이 응답시간이므로 응답시간을 기준으로 19% 향상이라는 결과가 되었으며 사용자가 많아 질수록 효율은 더 증가함을 볼 수 있다.

## 5. 결론 및 향후연구

SOA를 사용하는 기업과 이기종간의 데이터를 교환해야 할 경우가 많아지고 있다[4]. 사용자의 요구와 복잡한 환경에 영향을 받고 있으며 비즈니스 흐름도 다변화 하고 있다. 기존의 단방향 어플리케이션에서 SOA서버를 호출하는 경우도 있지만, 대형화 되고 복잡한 환경이라면 각각 물리적인 위치가 다른곳에서 같은 비즈니스를 수행해야 할 경우가 생기게 마련이다. 이에 따라 어플리케이션서버와 SOA서버들 사이에 Service Operation Factory를 사용함으로 인하여 비즈니스의 흐름을 원활히 하고 통신의 흐름을 제어 하고자 연구를 하게 되었다.

테스트를 실행하고 데이터를 검증한 결과 분산환경에서의 SOF로 인하여 어플리케이션의 요청 대비 효율이 응답시간 기준 19% 향상함을 보인다. 동시접속의 사용자가 증가함에 따라 각 서버의 성능을 반영하여 SOA 서버의 부하를 줄일 수 있다. 기존 1:1 매핑으로는 Request Response를 그대로 처리 하였으나, SOF는 각 서버의 상태를 점검하여 보다 효율적으로 접근함으로 다수의 동시사용자가 사용할 경우 효율이 증가한다.

향후 연구에는 현재까지의 변수를 바탕으로 좀더 세밀한 변수가 필요하며 SOF를 미러링 하는 방법이 더 연구되어 비즈니스를 향상시킬 수 있는 기법이 연구되어야 한다.

## 참고문헌

[1] L. Pouridas, "SOAP and Web Services", Software, IEEE Volume 23, Issue 6, 2006

[2] C. Janner, "Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services", IEEE IT Professional Volume 9, Issue 3, 2007

[3] T. Tajima, "Efficient Web Services Message Exchange by SOAP Bundling Framework", Enterprise Distributed Object Computing Conference, 2007

[4] N. Hochstein and A. Brenner, "Where to Start with SOA: Criteria for Selecting SOA Projects" , Hawaii International Conference on System Sciences, 2008

[5] 안동길,"서비스지향 아키텍처 기반의 요구사항 추적 모델 연구", 서강대학교 정보통신대학원, 2007

[6] D.S. Kim, "Virtual Factory Communication System and Its Application to Networked Factory Machine" , Industrial Electronics Society IECON, 2004

[7] D. 한센, SOA 자바 웹서비스로 통하는 서비스 지향 아키텍처, 위키북스, 2008

## 저자약력

**홍 민 석**

1999년 군산대학교 해생과(학사)
2009년 서강대학교 소프트웨어공학과(석사)
2000년~현재 누리어시스템 금융사업부 팀장
관심분야 : Service-Oriented Computing, Web Services Composition, 요구사항분석
이 메 일 : hms0402@gmail.com