The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 14, No. 6, pp.131-137, Dec. 31, 2014. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2014.14.6.131
JIIBC 2014-6-20

서비스 지향 아키텍처를 위한 경량 ESB 엔진의 설계 및 구현

Design and Implementation of Lightweight ESBus Engine for Service Oriented Architecture

김윤호*, 조성환**

Yoon-Ho Kim*, Seong-Hwan Cho**

요 약 서비스 지향 아키텍처란 서비스들이 표준 방식에 의해 서로 느슨하게 연결(loosely coupled)되어 특정 구현에 종속되지 않은 중립적인 인터페이스를 제공함으로써, 특정 서비스를 변경 하더라도 연결된 다른 서비스에는 영향을 주지 않는 유연한 구조를 의미한다. ESB(Enterprise Service Bus)는 서비스 지향 아키텍처를 실현하기 위한 중요한 관련 기술 중 핵심요소로서 위치를 확보해가고 있으나, 국내 서비스 지향 아키텍처를 위한 ESB에 대한 개발과 연구는 부족한 실정이다. 본 논문에서는 ESB의 중계 서비스, 전송서비스, 운영서비스, 모니터링 서비스, 애플리케이션 접속 서비스, 데이터 접속 서비스를 가능하게 하는 각 주요 응용 컴포넌트의 설계 및 구현을 하였으며, 메시지 건수에 대한 데이터의 크기별에 따른 처리시간을 측정하여 성능평가를 실시하였다.

Abstract Service Oriented Architecture(SOA) is a flexible structure which does not affect other services even when a specific service is changed. It provides the platform with neutral interfaces independent of any others, where services are loosely coupled in a standard way. While ESB (Enterprise Service Bus) is a key technology for service-oriented architecture, few research and development for ESB has been done. In this paper, we designed and implemented the key components of ESB such as mediation service, message oriented middleware service, operation service, monitoring service, application connection service, and database connection service. The performance evaluation was done by measuring the message processing time for the number of messages with various sizes.

Keywords : Service Oriented Architecture(SOA), Enterprise Service Bus(ESB), Performance Evaluation

I. 서 론

최근의 비즈니스 컴퓨팅 환경은 외부 및 내부의 요구사항 변화에 대하여 보다 빠른 대응을 필요로 하고 있다. 기존의 비즈니스 컴퓨팅 패러다임으로는 이러한 변화에 신속한 대응을 할 수 없으며, 이는 기업의 경쟁력 저하를 가져오는 경향이 있다. 이에 따라 변화 환경에 대한 유연하고도 신속한 대응을 위하여 새로운 패러다임인 서비스 지향 아키텍처(Service Oriented Architecture: SOA)가 제시되었다[1][12][13].

서비스 지향 아키텍처란, 비즈니스 프로세스를 수행하는 하나의 소프트웨어 컴포넌트로 구성된 서비스들이 표

*정회원, 상명대학교 컴퓨터과학부
**정회원, 금강대학교 교양학부(교신저자)
접수일자: 2014년 11월 5일, 수정일자: 2014년 12월 5일
게재확정일자: 2014년 12월 12일

Received: 5 November, 2014 / Revised: 5 December, 2014
Accepted: 12 December, 2014
**Corresponding Author: shcho@ggu.ac.kr
School of Liberal Arts & Sciences, Geumgang University, Kore

준 방식에 의해 서로 느슨하게 연결(loosely coupled)되어 특정 구현에 종속되지 않은 중립적인 인터페이스를 제공함으로써, 특정 서비스를 변경 하더라도 연결된 다른 서비스에는 영향을 주지 않는 유연한 구조를 의미한다[2]. 이러한 변화에 대한 빠른 대응은 민첩성, 재사용 및 통합의 용이성으로 직결되며, 이런 점에서 서비스 지향 아키텍처는 산업분야에 관계없이 모든 기업들이 추구하는 실시간 기업(Real-time Enterprise)을 실현하기위한 최적의 방법론으로서 위치를 확보하고 있으며, 웹 서비스의 성공적인 적용과 함께 중요성과 관심이 더욱 부각되고 있다[3,6].

서비스 지향 아키텍처를 실현하기 위한 중요한 관련 기술 중 ESB(Enterprise Service Bus)는 핵심요소로서 위치를 확보해가고 있다. 그러나 IBM, Oracle 등의 외국 기업들이 ESB 시장을 선점하여 독점하고 있으며, 국내 서비스 지향 아키텍처를 위한 ESB 엔진에 대한 개발과 연구는 부족한 실정이다. 따라서 본 논문에서는 국내 비즈니스 컴퓨팅 환경에 맞는 ESB의 주요 컴포넌트를 설계 및 구현함으로써, 기업 내외부의 표준화된 정보 공유 프레임워크 구축과 시스템 통합을 용이하게 하고자 한다. 이를 위하여 본 논문에서는 ESB의 중계 서비스, 전송서비스, 운영서비스, 모니터링 서비스, 애플리케이션 접속 서비스, 데이터 접속 서비스를 가능하게 하는 각 주요 응용 컴포넌트의 설계 및 구현을 하였으며, 메시지 건수에 대한 데이터의 크기별에 따른 처리시간을 측정하여 성능평가를 실시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 ESB의 개념 정의와 국내외의 기술동향에 대한 분석이 이루어지며, 3장에서는 ESB의 주요 컴포넌트에 대한 설계와 구현에 대하여 설명하며, 4장에서는 성능평가에 대한 환경설명 및 메시지 건수에 대한 데이터의 크기별에 따른 처리시간에 대한 성능평가가 이루어지며, 마지막으로 5장에서는 결론과 향후 연구에 대하여 언급한다.

II. ESB(Enterprise Service Bus)의 개념과 동향

기업에서 분산된 시스템들 간에 point-to-point 연결 모델에 의한 응용통합은 시스템들이 복잡해짐에 따라 많은 유지보수 비용을 초래하기 때문에, 전사적 어플리케이션 통합(EAI: Enterprise Application Integration)이라는 미들웨어기반의 spoke and wheel 모델로 발전되었다. 이 구조에서는 메시지 브로커(message broker)라는 미들웨어가 모든 응용간의 통신을 책임지며 라우팅과 자료변환을 하게 된다.그러나 이 모델은 메시지 브로커의 단일 장애점(SPOF: single point of failure) 문제와 확장성(scalability) 문제가 존재하게 된다. 초창기의 서비스 지향 아키텍처가 실패한 원인도 이러한 문제점에 기인한다. ESB는 이러한 두 모델의 문제점을 극복하고 서비스 지향 아키텍처 방식으로서 서비스와 이기종간의 응용을 연결하는데 초점을 둔 구조라 할 수 있다[11].

ESB에 대한 정확한 정의는 기능들이 구현된 형태에 따라 차이가 있기 때문에 공통된 정의를 내리기는 어렵다[5]. 이는 ESB라는 용어가 2002년 가트너그룹 분석자들로부터 처음 사용되면서, 서비스 지향 아키텍처를 위한 백본(backbone)으로서 메시지 기반 미들웨어, 웹서비스, 변환과 라우팅의 지능화 등을 통합하는 새로운 형태의 인프라구조에 대한 필요성이 처음 언급이 된 후에, 어떤 제품은 웹 서비스 기반구조나 경량 메시징 제품으로부터 진화 발전시켰고, 어떤 제품은 EAI제품에 서비스 지향 아키텍처를 위한 기능들을 추가하는 등의 다양한 접근방법을 통하여 구현이 된 결과라고 볼 수 있다. ESB에 대한 다양한 정의와 접근방법들이 있음에도 불구하고, 주요 개념과 목적은 서로 비슷하다고 볼 수 있다. 본 논문에서는 ESB는 안전하고 신뢰성 있는 방법으로 분산응용과 서비스의 상호작용을 용이하게 해주는 라우팅, 호출, 중재서비스를 제공하여주는 개방형 표준 메시지기반의 분산통합 인프라 구조로 정의 한다.

ESB가 서비스 지향 아키텍처를 실현하기 위한 중요한 기술 중 핵심요소로서 위치를 확보함에 따라 상용화된 ESB 제품군으로부터 오픈소스 ESB 제품군까지 다양한 ESB제품이 존재하고 있다.

IBM이나 Oracle과 같은 EBS 제품을 제공하는 회사들은 각 사의 도구들을 이용한 자체적 구축 방법론을 수립하고 있으며, 주요 솔루션 벤더들이 서비스지향 아키텍처 기능을 확장하거나 통합한 제품군을 출시하고 있다. 대표적인 오픈소스 기반의 ESB로는 Mulesoft의 Mule ESB, Apache ServiceMix, Red Hat의 JBoss ESB 등의 있다[11].

그러나 현실에서 이러한 다양한 상용화제품들과 오픈소스기반의 제품들로부터 기업의 통합문제에 알맞은 ESB 제품을 선택한다는 일은 기능과 성능차이로 인하여

매우 어려운 문제가 된다. 또한 ESB의 핵심기능 외에도 새로운 비즈니스 로직을 위하여 기존의 존재하는 기능과 연동하거나 재사용하는 문제와 같이 기업에 따라 상이한 ESB 핵심기능이 아닌 외적인 문제로 인하여 더욱 복잡한 문제가 된다.

ESB를 근간으로 한 서비스지향 아키텍처 구축에 대한 국내동향은 관련 구축 방법론 및 서비스지향 아키텍처 로드맵이 마련되고 향상된 EBS 도구들이 출시되고 있으며, 다양한 프로젝트를 통해 구축 경험을 축적하고 있으며, 향후 서비스지향 아키텍처 구축 사업의 기반을 마련하고 있다. EBS를 제공하는 회사들은 각 사의 도구들을 이용한 자체적 구축 방법론을 수립하고 있으며, 주요 솔루션 벤더들이 서비스지향 아키텍처 기능을 확장하거나 통합한 제품군을 출시하고 있다. 서비스지향 아키텍처 기술 적용이 가장 활발히 논의되고 도입에 적극적인 분야는 금융/통신부분이라 할 수 있으며, 이는 기업환경의 변화와 서비스지향 아키텍처 기술이 사업영역과 밀접한 관계가 있기 때문으로 분석되며, 공공/제조부분은 상대적으로 서비스지향 아키텍처 기술의 도입에 신중한 접근을 하고 있다.

III. ESB의 주요 컴포넌트의 설계 및 구현

1. 시스템 구조도 및 주요 기능

ESB의 주요 컴포넌트에 대한 시스템 구조는 그림 1과 같다.

메시지의 변환, 암/복호화, 라우팅, 압축/분할, 로그생성 등의 메시지 브로커 기능을 수행하는 중계서비스 컴포넌트, 메시지 큐 기반의 미들웨어기능을 수행하는 전송서비스 컴포넌트, 버전관리, 배포, 제어 등의 기능을 수행하는 운영 서비스 컴포넌트, 웹 애플리케이션 서버나 그룹웨어와 같은 응용 서비스 접속 서비스 기능을 수행하는 응용접속 서비스 컴포넌트, 데이터베이스, XML, EDI 등과 접속하는 기능을 하는 데이터 접속 서비스컴포넌트 등으로 구성된다.

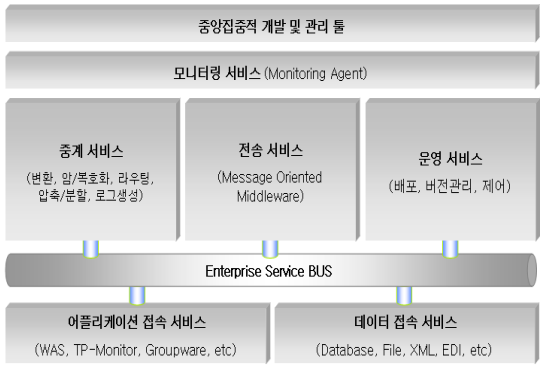


그림 1. ESB의 주요 컴포넌트 구조
Fig. 1. Main Component Structure of ESB

ESB의 중계 서비스, 전송서비스, 운영서비스, 모니터링 서비스, 애플리케이션 접속 서비스, 데이터 접속 서비스를 가능하게 가는 각 응용 컴포넌트의 구현을 위한 기능과 주요 내용과 기능은 표 1, 표2와 같다.

표 1. ESB 컴포넌트의 주요 기능
Table 1. Main fuction of ESB Component

기능	주요 기능
ESB Architecture	· 웹 서비스 허브, 서비스 레지스트리 기능을 포함 · 지원되는 운영체제 및 하드웨어의 다양성을 제공 · 부하분산 메커니즘의 적설성과 안정성을 고려한 부하균형 지원 · 높은 가용성 구성을 위한 제품 아키텍처의 적합성 및 소프트웨어 클러스터링 기능 제공을 통한 Fail Over지원
Connectivity	· 표준 프로토콜을 지원 · 기존 EAI 솔루션과 연동할 수 있는 방법의 다양성, 연동 메커니즘의 적절성 제공 · 다수 밴더의 J2EE 시스템과의 연동이 가능하기 위한 애플리케이션 플랫폼 연동 기능 제공
Service/Message 처리	· Conditional/Parallel Process · Content base routing을 위한 기능 제공 · Rule base routing을 위한 기능 제공 · 동기화된 한 개 이상의 프로세스 동시 수행, N:1 인터페이스 구현을 위한 서비스 Join 컴포넌트 제공 · Synchronous/Asynchronous서비스 패턴에 대한 기능 제공 · 네트워크 장애, 타깃 시스템 장애 시 메시지 보존, 롤백, 재전송 등을 통한 메시지 전송 보장 기능 제공 · 에러처리를 위한 기능 제공 · 대용량 XML 문서처리 지원

표 2. ESB 운영 관리 기능
Table 2. Operations management fuction of ESB Component

기능	주요 내용
보안	• 메시지 전송 시 보안기능 제공(XML Encryption, SSL) • 전자서명, SOAP 메시지 암호화 기능제공 • 사용자 인증, 서비스 권한 관리, 사용자 인증시스템 연동 (LDAP)기능 제공
Management	• 서비스 버전관리, 동적 서비스 변경관리 기능 제공 • Hot Deploy기능 지원 • 실시간 서비스 통계 기능, 원격지 접속 기능 제공
모니터링	• 서비스 장애 시 운영자 통지를 위한 기능 제공 • 서비스 장애, 지연 시 서비스 장애 지점에 대한 빠른 식별을 지원할 수 있는 트랙킹의 편리성, 다양성 제공 • 실행시점에 각 서비스 in/out 데이터에 대한 확인 기능 제공

표 3은 각 모듈 단위별로 기능을 분석한 내용과 기능 구성이다.

표 3. 모듈에 따른 기능 및 기능구성
Table 3. Functions and Function Sets in accordance with the Module

모듈	기능	기능 구성
Workbench	디자인 인터페이스	디자인 인터페이스 구성 및 저장
	배포 인터페이스	인터페이스 배포 및 재배포, 취소
	운영 인터페이스	배포된 인터페이스를 종료 및 시작
	모듈 관리 GUI	adapter 및 모듈을 GUI관리
	User 및 Group 관리	User, Group및 추가 삭제 및 관리
	Adapter 추가 삭제 기능	Adapter 추가 삭제 및 관리
	해당 Adapter 설정 기능	Adapter에서 필요로 하는 속성 정의
Server	인터페이스 버전 관리	내부 알고리즘에 의한 인터페이스 관리
	Protocol 처리 및 라우팅	Workbench 및 Agent에서 사용되는 protocol 처리 및 라우팅
	User 로그인 관리	User에 대한 권한 처리 및 로그인 제어
	Workbench connection 관리	Socket 통신에 의한 Workbench 관리
	Agent Connection 관리	Socket 통신에 의한 Agent 관리
	Service 관리	adapter 및 module 관리
Agent	Runner 관리 및 제어	Agent에 속해 Runner 관리
	Runner에 필요 정보 라우팅	Server에서 받은 정보를 Runner에 라우팅
Runner	Load 및 parsing	Runner의 필요정보를 load및 parsing
	Adapter SDK 제공	Adapter에서 사용할 SDK 제공
	flow 전송 및 처리	메시지를 해당 Adapter에 전송 및 처리
	데이터 라우팅	데이터 특정 값에 따라 해당 연계 목적지로 분기
	JSON 메시지 처리 기법	JSON 사용으로 인한 처리속도 향상
	메시지 error 핸들링	error 형식에 따른 명확한 로그
Monitoring	데이터 전송 현황	연계 데이터 실시간 진행상황 추적
	인터페이스 데이터 통계	다양한 조건(월,일,시)에 따른 통계 정보
	통계 엑셀 저장 및 뷰어	조건에 따른 통계 정보를 엑셀에 저장
	시스템 등록 및 뷰어	시스템 정보 관리를 위한 등록 및 뷰어
	게시판 및 뉴스	필요한 정보를 게시판 및 뉴스로 뷰어
Integration Link	메시지 저장 및 복원	메시지를 저장 하며 복원
	메시지 트랜잭션	메시지를 xml, hexa값으로 구성 가능
	큐 관리	큐를 관리(추가/삭제/수정)
	채널 관리 및 데이터 전송	채널에 의해 데이터 100%전송 가능
Adapter	데이터 연계	시스템 간의 정보를 XML로 연계
	데이터 가공	데이터 변환 및 라우팅
	데이터 송수신	실시간 메시지 전송 및 sync/async 가능

2. 주요 클래스의 구조와 기능

본 절에서는 ESB의 주요 컴포넌트 구현을 위한 주요 클래스의 구조와 기능을 설명한다.

Main은 시작을 위한 Bootstrap 클래스이며, 실행환경을 검사하고 설정하며 ClassLoader 생성, Startup class 생성, 생성된 Startup 클래스의 start() method를 호출하는 클래스 이다.

Shutdown은 시스템을 shutdown하기 위한 클래스 이며, Engine이 명령형 인자를 파싱하고, 시스템을 구성하는 초기 클래스이다.

Configurator는 환경구성에 관한 key 값을 가지고 있으며, 환경구성 데이터를 파싱하여 필요한 구성 정보를 얻는다.

Diagnostics는 시스템에 대한 진단보고서를 작성하며, 진단보고서에는 Product version, Directory information, Adapter information, System properties를 포함한다.

Switch는 컴포넌트에 대한 참조를 제공한다. 인터페이스에 대한 참조를 가지고 있으며, 각 컴포넌트 간에 사용할 일이 있을 경우 Switch를 통해서 참조를 얻는다.

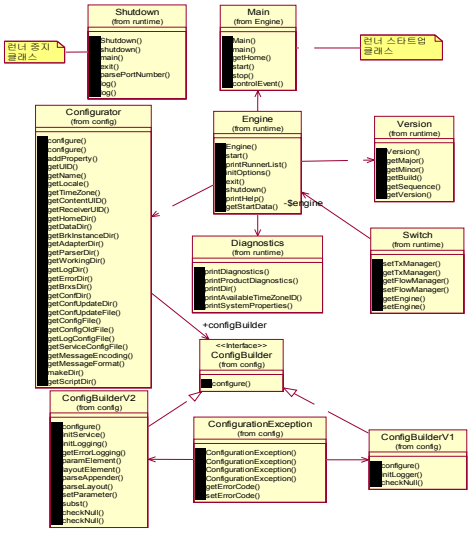


그림 2. 런너 클래스 다이어그램
Fig. 2. Runner Class diagram

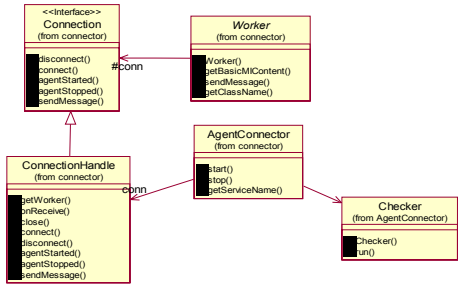


그림 3. 에이전트 커넥터 클래스 다이어그램
Fig. 3. Agent Connector Class diagram

ConnectionHandle는 커넥션 핸들을 위한 클래스이며, AgentConnector 에이전트와 Connection을 연결하고, 명령을 주고받는 통신 클래스이며, Checker 클래스는 연결 상태에 대해서 주기적으로 점검하여 연결이 끊어지면 재연결을 한다.

IV. 성능평가

1. 테스트 환경 및 시나리오

테스트를 위해 웹서비스 클라이언트, ESB, 목표 서버의 세 개의 시스템으로 구성되는 테스트 시스템 환경을 구축하였다.

웹서비스 클라이언트에서 메시지를 발생시켜 ESB 시스템으로 전송하면, ESB 시스템에서는 해당 메시지를 변환 및 처리하여 메시지를 목표 서버로 보내게 되며 메시지 발생부터 목표 서버 시스템에서 처리 될 때까지의 메시지 처리 및 변환에 소모된 시간을 측정하였다. 메시지 건수에 대한 데이터 크기 별로 성능 테스트를 하였으며 데이터 크기는 2Kb, 10Kb, 100Kb, 500Kb, 1000Kb 별로 구성하였으며 메시지 건수는 500건, 1000건, 3000건, 10000건 별로 시간을 측정하였다. 또한 발생되는 메시지는 두 가지의 형태로 발생시켜 ESB 시스템의 처리성능을 측정하였다. 웹서비스 클라이언트로부터 발생되는 메시지는 다음과 같은 두 가지 형태로 구성된다.

Type 1 메시지: JSON(JavaScript Object Notation)을 이용한 key:value 형식의 메시지
Type 2 메시지: JDOM(Java Document Object Model)을 이용한 XML 메시지

2. 테스트 결과

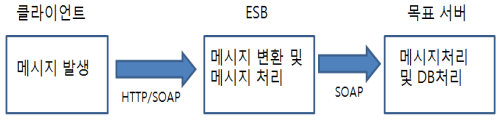
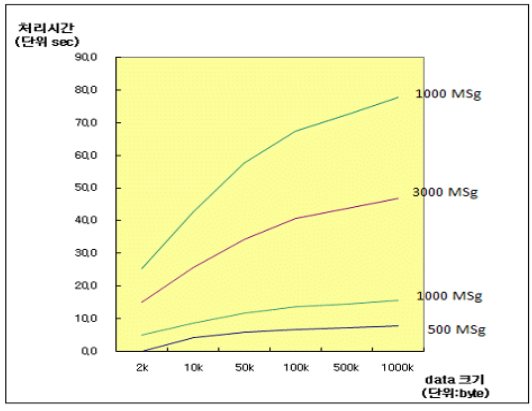


그림 4. 테스트 환경 구성도
Fig. 4. Block diagram of Test environment

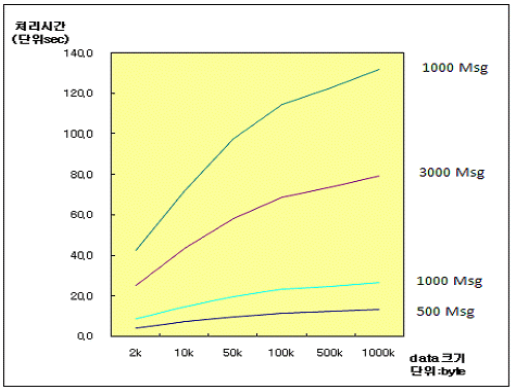
다음은 Type 1 메시지에 대한 데이터 크기에 따른 처리시간을 보여주고 있다.



건수 \ 크기	2k	10k	50k	100k	500k	1000k
500	2.5	4.3	5.8	6.8	7.3	7.8
1000	5.0	8.5	11.5	13.5	14.5	15.5
3000	14.9	25.6	34.2	40.4	43.5	46.6
5000	25.1	42.3	57.4	67.2	72.1	77.5

그림 5. Type 1 메시지에 대한 처리시간
Fig. 5. Processing time of Type 1 message

다음은 Type 2 메시지에 대한 데이터 크기에 따른 처리시간을 보여주고 있다.



건수 \ 크기	2k	10k	50k	100k	500k	1000k
500	4.3	7.2	9.8	11.5	12.3	13.2
1000	8.5	14.5	19.6	23.0	24.7	26.4
3000	25.3	43.5	58.1	68.7	74.0	79.2
5000	42.7	71.9	97.6	114.2	122.6	131.8

그림 6. Type 2 메시지에 대한 처리시간
Fig. 6. Processing time of Type 2 message

Type 1 메시지와 Type 2 메시지에 대하여 모두 데이터의 크기와 건수가 증가함에 따라 처리시간이 증가함을 알 수 있다. JDOM을 이용한 Type 2 메시지인 경우에 경량 데이터 교환포맷인 JSON 형태의 Type 1 메시지에 비하여 동일한 크기와 건수인 경우에 약 70% 정도의 처리시간의 증가를 가져오며, 이는 XML 파싱과 데이터 변환처리로 인한 오버헤드에 기인한다.

V. 결론

본 논문에서는 서비스 지향 아키텍처를 실현하기 위한 중요한 기술인 ESB를 핵심기능만을 구현하였다. 따라서 중계 서비스, 전송서비스, 운영서비스, 모니터링 서비스, 애플리케이션 접속 서비스, 데이터 접속 서비스를 가능하게 하는 각 응용 컴포넌트를 구현하였으므로 경량 ESB라 할 수 있다.

성능측정을 위하여 key:value 형식의 단순 메시지와 파싱이 필요한 XML 메시지, 두 가지의 형태의 메시지로 발생시켜 데이터 크기에 따른 메시지 건수 별로 ESB 시스템의 처리성능을 측정하였다. 본 논문에서 구현한 ESB는 메시지 중심의 ESB라 할 수 있으며, 따라서 메시지전송에 기반을 둔 처리시간에 초점을 맞추어 성능을 측정하였다.

ESB에 대한 메시지 중심의 접근방법은 서비스를 충분히 고려하지 않고 트랜스포트 부분의 구현과 성능에 초점을 맞추는 경향이 있다. 따라서 향후 연구로서 서비스를 생성, 배치, 관리하는 컴포넌트들을 고려한 종합적인 성능평가가 필요하며 이에 따른 분석과 향상이 필요하다.

References

[1]http://www.ibm.com/developerworks/kr/series/soa/index.html

[2] Footen J, An Introduction to Service Oriented Architecture, SMPTE MOTION IMAGING JOURNAL, Vol.119 No.4, 2010

[3] OSOA(Open Service Oriented Architecture), "http://www.osoa.org"

[4] Dirk Krafzig, Karl Banke, Dirk Slama, “Enterprise SOA: Service Oriented Architecture Best Practices”, Prentice Hall PTR, 2004

[5] David A. Chappell, “Enterprise Service Bus”, O'Reilly, 2004.

[6] Tony Baer, The road to SOA CBR Research 2006
[7] SOAP version1.2, http://www.w3.org/2000/TR/Group.
[8] Thomas Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, 2005.
[9]Http://esbperformance.org/wiki/ESB_Performance_Testing
[10] Themba Shezi, et al "Performance Evaluation of Enterprise Service Buses towards Support of Service Orchestration", International Conference on Computer Engineering and Network Security (ICCENS'2012) December
[11] Sanjay P. Ahuja, Amit Patel, "Enterprise Service Bus: A Performance Evaluation", Communications and Network, 2011, 3, 133-140
[12] Won-kyu Park, Young-bum Park, "Design and Implementation of SOA based S/W Services for Dynamic Behavior of Embedded System," The Journal of The Institute of Internet, Broadcasting and Communication (JIIBC), Vol. 10, No. 4, pp.29-34, August 2010
[13] Lee Sang Hyo, Yang Hae Sool, " Usability Evaluation Method for SOA Software." Journal of the Korea Academia-Industrial cooperation Society, Vol. 10, No. 7, pp. 1575-1584, 2009

저자 소개

김 윤 호(정회원)

- 1985년 : 서울대학교 계산통계학과 (학사)
- 1987년 : 서울대학교 대학원 계산통계학과(이학석사)
- 1996년 : 서울대학교 대학원 전산과학 박사
- 1997년 ~ 현재 : 상명대학교 소프트웨어대학 컴퓨터과학부 교수

<주관심분야 : 분산시스템, 저작권보호기술, 디지털콘텐츠>

조 성 환(정회원)

- 1980년 : 성균관대학교 전자공학과 (학사)
- 1982년 : 성균관대학교 대학원 전자공학과(공학석사)
- 1991년 : 성균관대학교 대학원 전자공학과(공학박사)
- 1982년 ~ 1985년 : 해군사관학교 전기 및 전자공학과 전임강사
- 1997년 : 미국 Columbia 대학 CATT Visiting Scholar
- 1985년 ~ 2002년 : 동서울대학 컴퓨터공학과 부교수
- 2002년 ~ 현재 : 금강대학교 교수

<주관심분야 : 영상통신, 무선네트워크, 저작권보호기술 (DRM)>