설계 패턴을 활용한 메시지 시스템의 소프트웨어 아키텍처 접근법 식별 및 평가☆

Architectural Approach Identification and Evaluation of Message System Using Design Pattern

고 현 희* 궁 상 환** 박 재 년***
Koh Hyon Hee Kung Sang Hwan Park Jae Nyon

요 약

성공적인 소프트웨어 아키텍처의 설계를 위해서는 아키텍처 설계의 기반이 되는 아키텍처 접근법의 선정이 우선 이루어져야 한다. 이 때 다양한 아키텍처 스타일 즉, 아키텍처 접근법들 중 어떤 것을 선택 할 것인가는 완성될 시스템이 어떤 기능적, 비 기능적 품질요구사항을 만족시켜야 하는지에 따라 달라지게 된다. 본 논문에서는 아키텍처 접근법 선정을 위한 평가 모델을 제안하고, 다양한 인터넷 활용과 엔터프라이즈 어플리케이션 통합에 활발히 사용되고 있는 메시지 시스템의 소프트웨어 아키텍처 접근법을 선정하고자 한다. 즉 여러 아키텍처 접근법 대안들이 존재할 경우 평가를 통해 시스템의 요구사항 만족 여부를 분석하여 대안들 중 가장 효율적이고 최적화 된 아키텍처 접근법을 선정하여 메시지 시스템의 아키텍처 설계 시 활용할 수 있도록 하고자 한다. 또한 아키텍처 접근법 식별시 스레드 기반의 설계 패턴을 활용하여 상세 설계와 구현 시 개발자들이 아키텍처를 바로 연계 시킬 수 있도록 하고자 한다.

Abstract

To design a software system in success, architectural approaches which are in basis of architectural design, must be primarily selected. What to choose among various architecture styles as specific as architectural approaches, varies along with the system status of what kind of functional or non-functional quality requirements should satisfy. In this study, we propose the evaluation model for making a selection of architectural approaches, and select an architectural approach for message system actively using in diversified internet utilization and enterprise application integration. In other words, if there may be possibly existed several architectural approaches, we present the most suitable method of architectural approach out of them through an evaluation of analyzing the system requirements satisfaction level. In addition, when it performs specific design and implementation utilizing design patterns based on thread, developers would be able to link up the architecture design directly.

☞ Keyword : 소프트웨어 아키텍처, 메시지 시스템, 설계 패턴

1. 서 론

복잡하고 방대해진 시스템 개발에서 소프트웨어 아키텍처는 데이터 구조나 알고리즘의 선택보다 중요한 부분이 되고 있다. 즉, 다양한 이해 관계자들의 요구사항을 시스템에 정확히 반영해야 하고, 이를 위해 시스템의 품질 속성과 이해관계자들의 이해관계를 반영한 소프트웨어 아키텍처의 설계가 성공적인 프로젝트를 위한 중요한 이슈가 되었다

아키텍처의 설계를 위해서는 우선 아키텍처 설

* 정 회 원 : 숙명여자대학교 대학원 컴퓨터과학과 박사과정
hhkoh@sookmyung.ac.kr(제 1저자)
** 정 회 원 : 천안대학교 정보통신학과 조교수
kung@cheonan.ac.kr(공동저자)
*** 정 회 원 : 숙명여자대학교 이과대학 학장
jnpark@sookmyung.ac.kr(공동저자)
[2005/03/08 투고 - 2005/03/28 1차 - 2005/04/12 2차 - 2005/04/25 심사완료]
☆ 이 논문은 2004년도 한국학술재단의 지원에 의하여 연구되었음(KRF-2004-003-D00347)

계의 기반이 되는 아키텍처 접근법을 선정하여야 한다. 다양한 아키텍처 접근법들 중 어떤 것을 선택 할 것인가는 완성될 시스템이 어떤 기능적, 비기능적 품질요구사항을 만족시켜야 하는지에 따라 달라지게 된다. 따라서 시스템의 요구사항을 만족시키기 위한 여러 아키텍처 접근법 대안 들 중 가장 적합한 아키텍처를 선정하기 위해서는 시스템의 요구사항에 따라 아키텍처를 검증하여 품질 요구사항을 포함하는 요구사항을 만족할 수 있는 아키텍처를 선정해야 한다.

본 논문에서는 설계 단계에서 아키텍처 설계의 기반이 되는 아키텍처 접근법 대안들을 식별하고 평가하여 가장 적합한 아키텍처 접근법을 선정함으로써 아키텍처 설계의 신뢰도를 높이고자 한다. 또한 본 논문에서 제안하고 있는 아키텍처 접근법 선정을 위한 평가모델을 통해 다양한 인터넷 활용과 엔터프라이즈 어플리케이션 통합에 활발히 사용되고 있는 메시지 시스템의 소프트웨어 아키텍처 접근법 대안들을 식별하고 평가하여 효율적이고 최적화된 아키텍처 접근법을 선정하고, 아키텍처 접근법 식별 시 스레드 기반의 설계 패턴을 참조함으로써 개발자들이 향후 상세 설계와 구현 시 아키텍처 설계와의 연계를 용이하도록 하며, 아키텍처 접근법 대안들의 평가를 위한 프로그램을 구현 시에도 활용하여 평가의 효율성을 높이고자 한다.

2. 관련연구

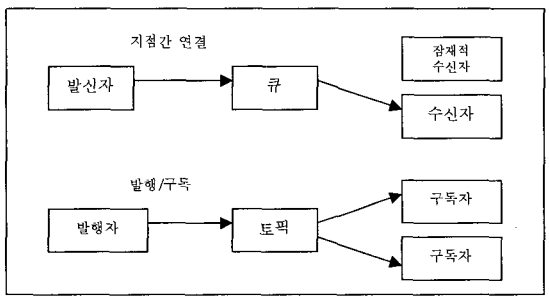
2.1 메시지 시스템

메시징은 네트워크에서 클라이언트와 클라이언트 간에 주고 받는 메시지와, 이를 처리하는 서비스 제공자로 구성된다[5].

메시지 시스템의 아키텍처는 그림 1에서 보는 바와 같이 크게 하나는 일대일 접속을 위한 지점간 연결(point-to-point) 모델과, 또 하나는 일대다 또는 다대일 접속 형태를 지원하는 발행/구독(publish-and-subscribe) 모델이다[5].

지점간 연결(point-to-point) 모델은 메시징 클라이언트가 큐(queue)라고 불리는 가상채널을 이용하여 송신자와 수신자가 일대일의 관계로, 동기식 또는 비동기식으로 메시지를 송수신하는 방식이다. 발행/구독(publish-and-subscribe) 모델은 메시지 출판자와 구독자 사이의 관례가 일대다의 형태를 구성하며, 이들 간 토픽(topic)이라는 가상채널을 통해 메시지를 전송하는 통신방식이다.

선 마이크로시스템사를 중심으로 규격화된 JMS (Java Message Service)는 메시지 시스템이 갖춰야 할 API차원의 표준 규격을 제공한다 따라서 JMS의 기본적인 API는 메시지 시스템의 아키텍처 접근법을 식별하기위한 기본적인 요구사항이 된다. JMS를 기반으로 한 메시지 시스템의 요구사항은 4절에서 설명한다.



〈그림 1〉 메시지 시스템 모델

2.2 설계 패턴

설계 패턴은 객체지향 소프트웨어의 구조 설계 및 세부 설계를 지원하여 유연성이 있고 재사용이 가능한 객체지향 소프트웨어를 가능하게 한다. 즉, 객체지향의 특성인 클래스 또는 객체의 개념이나 상속, 연관 등 클래스간의 관계성, 인터페이스의 이용 등을 통해 개개의 객체를 모듈화 하는 차원에서 여러 클래스들의 조립을 통해 경우마다 단순히 파라미터 등을 활용해서 일련의 클래스의 집합을 재사용 할 수 있도록 한다.

설계 패턴은 아키텍처 스타일과 비교해 볼 때 소프트웨어의 구조적 패턴을 지원하는 개념을 포

함하고 있다는 점은 비슷하나, 어느 정도의 추상화 레벨에서 구조적인 패턴을 지원하는 가에는 서로 차이가 있다. 그러나 아키텍처 스타일과 설계 패턴은 별개의 개념이라기 보다는 연계, 활용이 가능하다고 보인다. 즉, 아키텍처 스타일은 빌딩 블록을 위한 디자인 요소들의 집합 또는 이러한 빌딩 블록을 구성하기 위한 규칙 및 제약 뿐 아니라 아키텍처 스타일 내에서 생성되는 설계 요소를 분석하고 관리하는 데 필요한 도구를 제공한다. 한편 설계 패턴은 특정한 하나 또는 여러 개의 아키텍처 스타일 안의 더 작고 세부적인 문제를 해결하는 문제에 초점을 맞추고 있으므로 설계 단계에서 아키텍처 접근법 식별 시 설계 패턴과 아키텍처 스타일, 이 두 방법을 효과적으로 활용할 수 있다.

최근의 설계 패턴에 대한 연구는 스레드를 기반으로 하는 병렬 처리 환경에서의 설계 패턴을 포함한다. 이것은 사실상 아키텍처 스타일이나 패턴에 포함 될 내용이나, 프로그래밍까지 연결되는 상세한 표현을 하고 있고, 특히 스레드 사이에 함수 호출이나 자료 접근이 가능하다는 점으로 인해 설계 패턴의 범위에 포함 시켜야 하는지 혹은 아키텍처 패턴의 범위에 포함 시켜야 하는지 판단하기 어려운 부분이다. 스레드 기반의 설계 패턴을 기반으로 하는 아키텍처 스타일은 소프트웨어 설계자나 프로그래머가 그 아키텍처를 바로 프로그램에서 이용할 수 있다는 커다란 장점을 제공한다는 점이 중요하다.

자바환경의 경우는 스레드 기반의 설계 패턴이 소개되고 있는데, 이것은 설계 패턴을 벗어나 아키텍처 패턴의 영역에 포함될 내용이다. 이러한 접근은 아키텍처를 상세설계 및 구현과 쉽게 연계시킬 수 있는 매우 바람직한 방법으로 본 연구에서도 이를 활용하여 메시지 시스템의 아키텍처 접근법을 식별하고자 한다.

3. 아키텍처 접근법 선정을 위한 평가 방법

아키텍처 설계는 시스템이 만족해야 하는 기능 요구사항과 비 기능 품질 속성요구사항을 만족시키는 아키텍처 접근법 식별을 통해 아키텍처 설계를 확장해 나가게 된다.

본 논문에서는 설계 패턴을 기반으로 식별된 아키텍처 접근법 대안들에 대한 품질 평가 모델을 만들고, 이를 가지고 달성하고자 하는 품질 속성 요구사항에 대한 객관적 검증을 통해 아키텍처 접근법을 선정하고 설계를 발전시켜 나갈 수 있도록 한다.

설계 단계에서의 아키텍처 접근법 선정을 위한 평가 모델은 다음의 그림 2와 같다.

시스템의 기능 요구사항과 비 기능 요구사항으로부터 아키텍처 동인을 얻어 아키텍처 접근법을 식별하게 되고, 아키텍처 접근법이 여러 개의 대안이 존재할 경우 각 대안에 대한 평가 과정을 거쳐 해당 품질 속성에 가장 적합한 하나의 아키텍처 접근법을 선정하게 된다. 이렇게 선정된 아키텍처 접근법에 기능을 할당하여 아키텍처를 확장 설계 하고, 설계된 아키텍처에 대해서는 다시 한번 평가 과정을 거쳐 최종적으로 설계를 완성하게 된다.

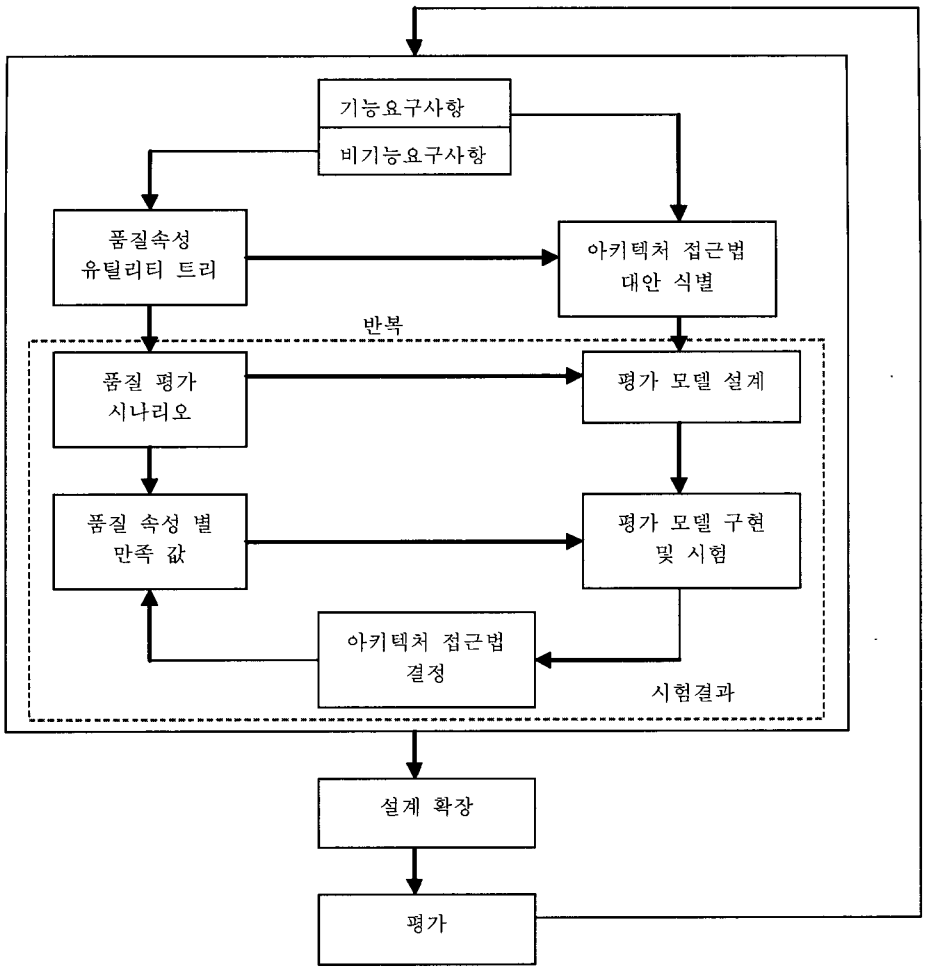
4. 메시지 시스템의 요구사항 분석

메시지 시스템의 특성과 메시징 시스템이 갖춰야 할 API차원의 표준 규격을 제공하고 있는JMS의 요구사항을 기반으로 메시지 시스템의 기능, 비기능 요구사항에 대해 정리하면 다음과 같다,

4.1 기능요구사항

메시지 시스템의 기능 요구사항은 다음과 같다.

1) 메시지가 한 시스템에서 다른 시스템으로 네트워크를 타고 비동기식으로 전달 된다.
2) 일대다(브로드케스팅) 전송이 가능하다.
3) 일대일(p2p) 전송이 가능하다.
4) JMS 클라이언트는 생산자와 소비자 둘 다 될 수 있다.
5) 메시지전달에 대한 확인 응답을 한다.
6) 토픽을 구독하는 모든 클라이언트는 그 토픽



〈그림 2〉 아키텍처 접근법 선정을 위한 평가 모델

으로 발행된 메시지에 대해 자신만의 복사본을 갖는다. 즉, 한명의 발행자에 의해 생산된 하나의 메시지가 수백 또는 수천개의 구독자에게 복사되어 전달 될 수 있다.

7) 메시지는 무조건 한번만 전달된다.
8) JMS 클라이언트는 지속적 구독(durable subscription)을 할 수 있다.

4.2 비 기능 요구사항

메시지 시스템의 비 기능 품질 요구사항으로는 대량의 데이터를 지연 없이 빠르게 목적지에 전달하여야 한다는 성능 측면의 요구사항과, 동시에 연결될 수 있는 클라이언트 수에 제한이 있어서는 안 되는 확장성, 장애 발생시에도 다른 시스템에 영향을 주어서는 안 되는 가용성, 그리고 복구 후 정확한 전달이 다시 이루어 져야 한다는 신뢰성과 보안관련 요구사항이 있다.

메시징 시스템의 비 기능 요구사항을 정리해 보면 다음과 같이 분류해 볼 수 있다.

1) 성능

JMS 서비스 제공자가 생산자로부터 소비자에게 시

스템을 통해 메시지를 전달하는 속도를 말한다[5].

2) 확장성

JMS 서비스 제공자가 지원하는, 동시에 연결할 수 있는 클라이언트의 수를 말한다[5]. 즉, 연결된 많은 생산자와 소비자를 위해 많은 양의 메시지를 동시에 처리할 수 있는지에 대한 효율적인 비율을 나타낸다[5].

3) 신뢰성

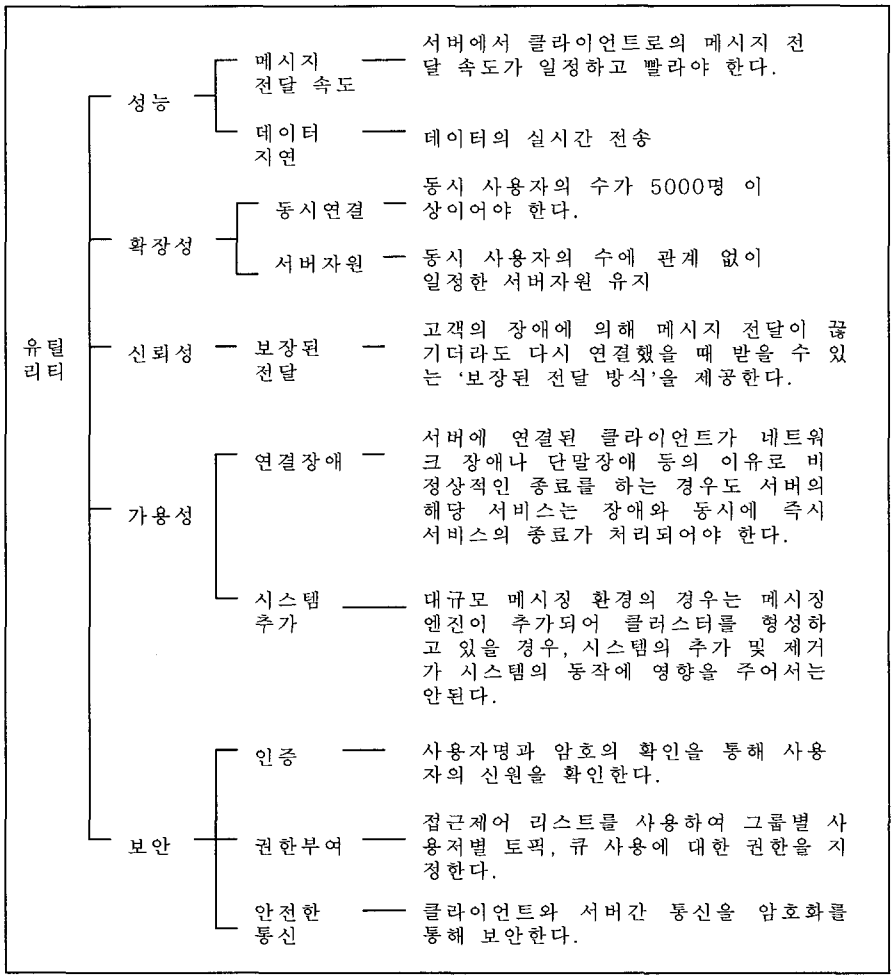
지속적 구독자와 JMS서버와의 연결이 끊어지더라도 서버는 구독자에게 전달할 메시지를 저장할 책임이 있다. '저장 후 전달 메시징 (store -and -forward messging)'은 보장된 메시징을 위한 중요한 요소이다.

4) 가용성

비동기식 메시징 시스템에서 각 하위 시스템은 다른 시스템과 분리된다. 각자 메시징 서버를 통해 통신하고 있기 때문에 한곳의 장애가 다른 곳의 운영에 영향을 주지 않는다[5].

(5) 보안 (Security)

JMS 서비스 제공자가 일반적으로 제공하는 보안적인 측면으로는 인증, 권한부여, 안전한 통신 등 세가지가 있다[5].



〈그림 3〉 메시지 시스템의 품질 속성 유틸리티 트리

4.3 품질 속성 유틸리티 트리에 의한 아키텍처 동인 결정

아키텍처 동인 결정을 위해 메시징 시스템의 비기능 요구사항 시나리오를 품질 속성별로 분류하여 유틸리티 트리를 그려보면 그림 3과 같다.

엔터프라이즈 애플리케이션의 성능, 확장성, 신뢰도는 실제 배치 환경에서 가장 중요하게 고려되어야 한다[5]. 메시징 미들웨어에 기초한 이런 환경은 중요한 부분이다. 메시징 엔진은 이러한 요구사항을 공통적으로 지원하는 기반 시스템이 된다. 또한 메시지 전달 속도와 관련된 성능이나 동시 연결자 수는 메시지 시스템의 평가에 중요한 요소 이므로 아키텍처 동인으로 정할 수 있다. 또한 정확한 메시지 전달도 메시지 시스템의 중요한 요소이다. 가용성의 경우 비동기식 메시징 시스템은 밀접하게 결합된 RPC와는 달리 메시지 시스템의 아키텍처 자체가 각 하위 시스템과 다른 시스템과 분리하여 각자 메시징 서버를 통해 통신하고 있기 때문에 한곳의 장애가 다른 곳의 운영에 영향을 주지 않는다. 따라서 가용성의 경우는 소프트웨어 아키텍처에서 더 이상 고려하지 않아도 된다. 또한 보안의 경우는 어떻게 구현하느냐는 벤더가 정하며, 각 벤더들은 JMS 클라이언트간에 서로 인증하고 권한을 부여하고, 안전한 통신을 할 때 사용할 수 있는 기술을 나름대로 조합하여 사용하고 있으므로, 여기서 아키텍처 설계를 위한 동인으로 성능, 확장성, 신뢰성 시나리오를 선택하도록 한다.

5. 메시지 시스템의 아키텍처 접근법 식별

메시지 시스템은 먼저 메시지 송신 역할을 하는 발행 부분과 메시지 수신 역할을 하는 구독부분, 그리고 메시지 발행자와 메시지 구독자를 관리하는 메시지 관리부, 그리고 메시지의 저장과 검색을 지원하는 부분으로 구분된다[1].

메시지 시스템의 성능과 확장성 요구사항을 고려하여 메시지 시스템의 성능과 확장성 요구사항을 고려하여 아키텍처 접근법은 메시지 발행이나 구독을 위한 클라이언트로부터 요청이 들어올 때마다 복수의 클라이언트의 요구를 동시에 처리하기 위해 각 클라이언트에게 스레드를 할당하는 멀티 스레드 패턴을 활용한 아키텍처 접근법을 선택한다.

멀티 스레드 패턴은 멀티 스레드의 생성이나 처리 방법에 따라 여러 대안이 나올 수 있다. 또한 클라이언트의 수가 많아짐으로써 스레드가 무한정 생김으로써 발생할 수 있는 문제 등을 고려하여야한다. 본 논문에서는 이러한 사항들을 고려하여 다음과 같은 3개의 아키텍처 접근법 대안을 식별하였다.

1) 클라이언트로부터 요청이 들어올 때마다 메시징 서버에서 클라이언트 에게 서비스를 담당할 스레드를 하나씩 할당하는 Thread-per-Client 아키텍처
2) 여러 개의 스레드를 미리 가동해 두고 워커 스레드로서 기다리게 하는 Thread Pool 아키텍처
3) 서버측의 단일 스레드를 사용하는 Single Thread 아키텍처

5.1 Thread-per-Message 패턴을 이용한 Thread-per-Client 아키텍처

메시지 시스템의 가장 간단한 구조는 출판이나 구독을 위한 클라이언트로부터 요청이 들어올 때마다 메시징 서버에서 클라이언트에게 서비스를 담당할 스레드를 하나씩 할당하는 방법이다[1]. 즉, 복수의 클라이언트의 요구를 처리하는 서버를 실현하기 위해서 Thread-per-message 패턴이 사용된다.

Thread-Per-Message 패턴을 이용하면 클라이언트에 대한 호스트의 응답성이 좋아지고 지연 시간이 줄어든다고 볼 수 있다. 그러나 클라이언트로부터 요청이 올 때마다 호스트내에서 새로운 스레드를 기동하게 되고 스레드의 기동에는 시간이 걸리므로 클라이언트의 수가 많아질수록 응답성이 높아진다고 볼 수는 없다. 즉, 대규모 접속을 가정하는 메시지 시스템에서 메시징 서버의 성능에 영

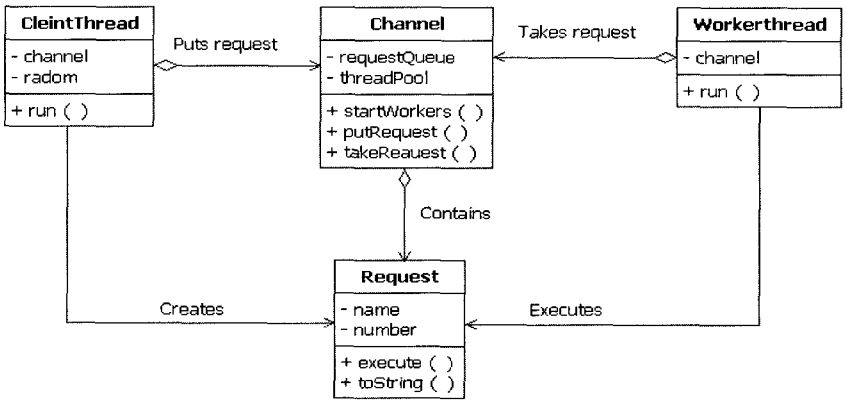
향을 주는 상황은 동시에 많은 사용자가 서버에 접속 할 때 발생한다.

여기서 고려해야 하는 문제는 우선 스레드가 무한정 생김으로써 발생할 수 있는 장애이며, 또 한 가지는 스레드를 생성하고 기동하는 데 소요되는 시간의 문제이다. 즉 Thread-per-Client 아키텍처는 Persistent Message를 통해 "보장된 전달"과 "지속적 구독"등 신뢰성 요구사항을 반영하고 있고, 아키텍처는 성능에 있어서의 응답성과 지연시간은 고려하고 있지만, 확장성과 확장에 따른 성능은 고려하지 못하고 있다.

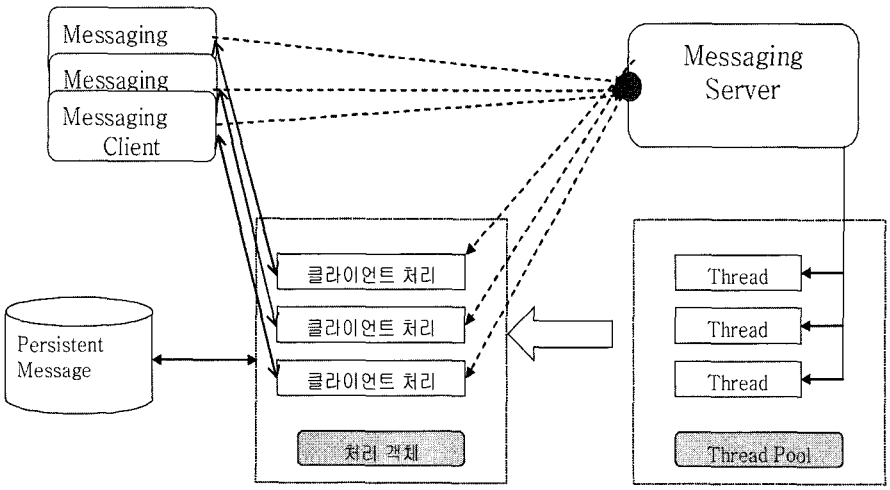
5.2 Worker-Thread 패턴을 이용한 Thread Pool 아키텍처

Worker Thread 패턴에서는Thread-per-Meaasge 패턴과 같이 다른 스레드에 실제의 처리를 맡긴 그러나 스레드는 매회 기동하는 것이 아니라 미리 가동해 두고 Worker 스레드로서 기다리게 한다. 이것에 의해서 스레드의 기동에 걸리는 시간을 절약할 수 있다.

여기서 고려해야 하는 문제는 Worker 즉, Thread Pool 안에 기동 되어 있는 스레드의 수이다.



〈그림 4〉 Worker Thread 패턴의 클래스 다이어그램[6]



〈그림 5〉 메시지 시스템의 Thread Pool 아키텍처[1]

Worker의 수가 많으면 그만큼 병행해서 일을 실행할 수 있다. 그러나 동시에 요구되는 일의 수보다도 많은 Worker가 있어 도움이 되지는 않는다. 너무 많은 Worker는 일을 하지 않고 메모리를 점유하고 있을 뿐이다. 즉, Thread Pool 아키텍처의 경우 확장성과 성능 측면을 모두 고려하고 있지만, Worker의 수에 따라 성능에 저하를 가져 올 수 있는 문제가 있다.

5.3 Event Dispatching Thread 패턴을 이용한 Single Thread 아키텍처

서버측의 단일 스레드를 사용하는 경우 송신자 클라이언트로부터 들어오는 메시지는 채널에 메시지가 들어오는 순서대로 받아오면 되므로 단일 스레드로 처리하는 데는 별다른 처리가 필요하지 않다. 그러나 수신자의 경우 단일 스레드가 여러 수신자에게 메시지를 송신하기 위해서는 어떤 한 클라이언트에게 집중되지 않도록 스케줄링을 해주어야 한다. 이 패턴에서는 수신자 클라이언트가 메시지를 받기를 원할 경우 메시지를 끝까지 받았다는 EndMessage를 받기 전까지 계속 서버에게 Request 메시지를 보내게 하고, 서버는 Request Message를 받을 때마다 해당 클라이언트에게 메시지를 송신하는 구조를 이용하고자 한다.

Single Thread 아키텍처를 구현하기 위해 사용하고 있는Event Dispatching Thread 패턴은 이벤트 큐에서 이벤트를 하나 꺼내어 그것을 실행하고, 실행이 끝나면 다시 이벤트 큐로 돌아와 다음의 이벤트를 집어서 실행하는 과정을 반복한다. 만일 이벤트 큐에 이벤트가 하나도 없다면 Event Dispatching Thread는 이벤트가 오는 것을 기다린다. 여기서 이벤트는 Worker Thread 패턴의 Request 메시지에 해당하고 이벤트 큐는 채널에 해당한다고 볼 수 있다. 이 패턴에서Event Dispatching Thread는 단일 스레드 즉, 하나의 Worker Thread가 된다.

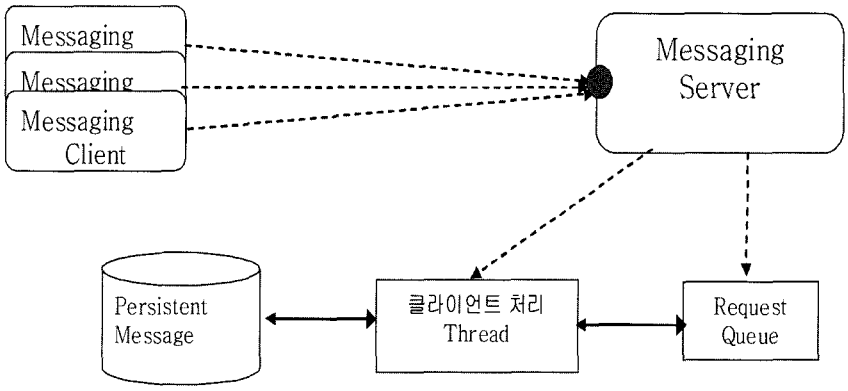
Worker가 하나면 멀티 스레드의 장점을 가지고 있지 않은 것 같지만, 배타제어가 불필요하게 되고 있다는 설계 덕분에 Event Dispatching Thread가 실행되기 전의 메소드는 Worker Thread끼리의 배타 베어가 불필요하게 된다.

6. 평가를 통한 아키텍처 접근법 선정

6.1 평가 시나리오 작성

식별된 3개의 아키텍처 접근법 대안은 Persistent Message를 통해 신뢰성 요구사항을 만족시키고 있다. 따라서 확장성과 성능 요구사항의 평가 결과에 따라 가장 효율적인 아키텍처 접근법을 선정하도록 한다. 확장성과 성능은 동시 사용자 수에 따라 성능이 영향을 받을 수 있는 속성이므로 두 속성을 같이 고려하여 평가 시나리오를 작성하여야 한다. 즉, 확장성과 성능 요구사항을 평가하기 위해서는 연결된 송신자, 수신자의 수에 따른 메시지 전달 속도를 측정해 보아야 한다. 연결된 송신자와 수신자는 메시징 서버의 스레드를 기동시키므로 서버 측의 스레드 기동에 드는 시간과 서버자원의 결과를 가지고 확장성을 예측 해볼 수 있고, 성능은 메시징 서버에서 기동된 스레드가 메시지를 생성해서 전달하는 속도를 가지고 예측해 볼 수 있다.

ES1. 메시징 서버의 스레드 생성과 기동 시간 측정한다.
ES2. 클라이언트 수에 따른 메시지 생성 및 전달 속도를 측정한다.
- 3가지 경우의 "one-to-one" 테스트를 실행한다. (10/10/10, 100/100/100, 1000/1000/1000)
- 테스트 메시지는 10MB의 파일을 512Byte를 메시지 단위로 전송하는 Byte형태의 메시지로 테스트한다.
- 메시지 전송 결과를 측정하는 과정을 5번 반복하고 그 평균값을 최종 결과로 한다.



〈그림 6〉 메시지 시스템의 Single Thread 아키텍처

따라서 평가 시나리오는 다음과 같이 만들 수 있다.

"one-to-one" 테스트는 클라이언트가 짝을 이루는 즉 송신자/수신자가 짝을 이루어 하나의 토픽으로 메시지를 교환하는 것이다.[10]. 예를들어 "10/10/10"(10개의 송신자(publisher), 10개의 수신자(subscriber), 10개의 토픽(topic))테스트는 10개의 출판과 구독 쌍이 10개의 토픽으로 메시지를 교환하는 것이다.

6.2 평가 결과 및 아키텍처 접근법 선정

평가시나리오 ES1.에 대한 시험은 먼저 스레드를 생성시키고 start()를 호출하여 실제 스레드가 동작 되기까지의 시간 소요를 프로그램으로 구현하여 실시하였다[1]. 그 결과는 다음과 같다[1].

1) 1000개의 문자열을 만드는데 소요되는 시간 : 0.006 milisecond
2) 1000개의 스레드를 만드는데 소요되는 시간 : 0.066 milisecond
3) 1000개의 스레드를 만듬과 동시에 실행시키는 시간 : 2.433 milisecond

위의 결과로 메시지 시스템에서 스레드를 실행시키는 시간이 성능저하에 중요한 요인이 될 수 있음을 알 수 있다. 즉 접속자 수가 늘어나 스레드 할당이 계속 증가할 경우 성능이 저하 될 수 있다.

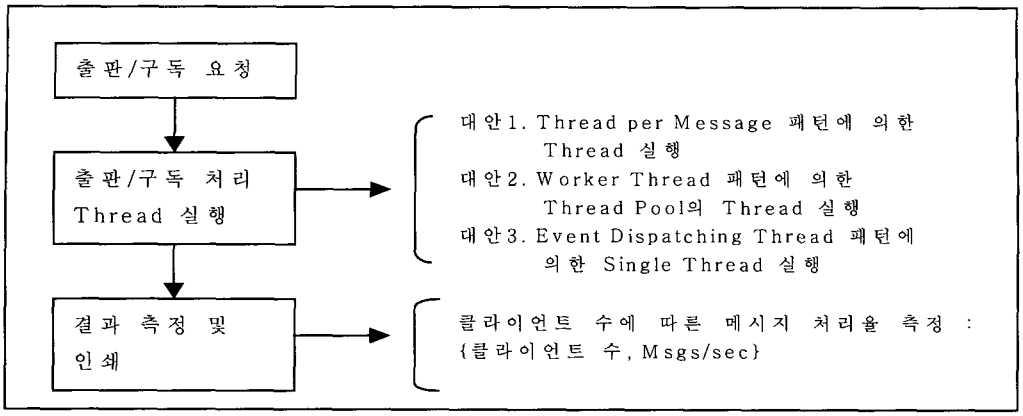
평가 시나리오 ES2.에 의해 확장성과 그에 따른 성능 품질 속성을 평가하기 위한 모델은 그림 7과 같고, 평가를 위한 구현은 각 참조 패턴을 활용하여 구현하였다.

구현된 평가 모델에 의한 시험 결과는 표 1과 같다.

위의 표에 나타난 시험 결과를 볼 때 접속자 수가 작을 때는 Thread per Client가 좋은 성능을 보여주고 있으나, 접속자의 수가 많아져서 스레드 할당이 많아 지는 경우 Thread Pool의 성능이 좋아지는 걸로 볼 수 있다. 본 사례연구에서 Thread Pool의 스레드 수를 100개로 하였으나, Thread Pool의

〈표 1〉 메시지 전송 테스트 결과

P/S/T	MsgSize (Bytes)	Msgs/sec		
		Thread per Client	Thread Pool	Single Thread
10/10/10	512	913	624	103
100/100/100	512	1406	4338	76
1000/1000/1000	512	1240	2145	18



〈그림 7〉 평가 시나리오 ES2.에 의한 평가 모델

스레드 수를 다르게 할 경우 다른 결과를 보일 수도 있다. 스레드 Pool의 스레드 개수를 100개로 한 경우 100/100/100의 상황에서 좋은 성능을 보여주고 접속자 수가 늘어남에 따라 성능이 저하되는 것을 볼 수 있다. 따라서 Thread Pool 아키텍처의 경우 Thread Pool의 스레드 수를 정하는 것이 중요한 요소라 볼 수 있다.

Thread per Client 아키텍처의 경우는 접속자 수가 증가함에 따라 성능의 큰 차이는 보이지 않지만, 성능의 감소하는 추세를 보이는 것으로 보아 접속자 수가 많아 짐에 따라 스레드 수가 증가하여 성능이 감소하고 있음을 예측할 수 있다.

Single Thread 아키텍처의 경우는 접속자의 수가 많아질수록 스케줄링에 의한 부하가 많아 성능이 감소하는 결과를 보이고 있다.

따라서 세가지 아키텍처 접근법 대안 중 Thread Pool 아키텍처가 가장 좋은 평가 결과를 보이고 있고, Pool에 생성된 스레드의 수를 다르게 하면 평가 결과가 더 좋아질 수도 있다.

위의 결과에 의해 본 연구에서는 Thread Pool 아키텍처 접근법을 선정한다.

7. 결론 및 향후 연구과제

본 논문에서는 아키텍처 접근법 선정을 위한 평가 모델을 제안하고, 설계 패턴을 기반으로 메시지 시스템의 아키텍처 접근법을 식별하여, 여러 아키텍처 접근법 대안이 존재할 경우 평가 과정을 거쳐 결과를 분석하여 가장 적합한 아키텍처 접근법을 선정하도록 하였다.

아키텍처 접근법 식별 시 설계패턴을 기반으로 할 경우 개발자들이 향우 상세설계와 구현까지의 연계를 용이하게 할 수 있다. 또한 여러 대안이 존재할 경우 아키텍처 접근법 선정을 위한 평가 모델을 제시함으로써 개발자들이 설계 과정 중에 가장 적합한 접근법을 선정하도록 도와 줌으로써 아키텍처 설계의 신뢰성을 높이고 있다. 아키텍처 접근법 평가 시 설계 패턴에 기반한 평가 프로그램 구현과 시험 결과에 의해 선정함으로써 기존의 경험에 의존한 검토 방식의 평가방법 보다 객관적인 방법으로 결과의 신뢰도 또한 높이고 있다.

본 논문에서는 평가 결과에 따라 Thread Pool 아키텍처가 가장 양호한 성능을 보여주는 것으로 아키텍처 설계 대안으로 최종 선정되었다. 이 연구를 통해 중요한 품질 요구사항을 실현하는 아키텍처 접근법이 설계 단계에서 결정되어 많은 비용 절감이 예상될 수 있다.

향후 연구과제로는 메시지 시스템 이외의 다른 도메인의 아키텍처 설계 시 설계 패턴을 활용하는 방안과 평가 방법에 대한 연구가 이루어져야 한다.

참 고 문 헌

[1] 궁상환, "모바일 인터넷 환경에서 Dynamic Scalable 메시지 성능에 관한 연구", 한국전자 통신 연구원, 2002

[2] Jayatirtha Asundi, Rick Kazman, Mark Klein, "Using Economic Considerations to Choose Among Architecture Design Alternatives" Technical Report CMU/SEI, pp.1-17, 2001

[3] Len Bass, Paul Clements, Rick Kazman, "Software Architecture in Practice Second Edition", Addison Wesley, pp.307-324, 2003

[4] Paul Clements, Rick Kazman, Klein, "Evaluating Software Architectures : Methods and Case Studies", Addison Wesley, pp. 43-84, 2002

[5] Richard Monson-Hadefel, Davi Chappell, "Java Message Service" O'Reilly, 2000

[6] Yuki Hiroshi 저 , 조해미 역 "Java 언어로 배우는 디자인 패턴 입문-멀티 스레드 편" 영진출판사, pp.247-292, 2003

[9] Jahming Technologies, "SonicMQ 4.0과 IBM MQSeries 5.2에 대한 벤치마크 비교", Jahming Technologies 2001

[10] Sonic Software Corporation, "JMS Performance Comparison: Publish/Subscribe Messaging", Sonic Software Corporation, pp.4-5, 2003

[11] Gregor Hohpe, Bobby woolf, "Enterprise Integration Patterns : Designing Building, and Deploying Messaging Solutions" Addison-Wesley, 2003

[12] Mark Klein, Rick Kazman, "Attribute-Based Architectural Styles", Technical Report CMU/SEI, 1999

[13] Len Bass, Mark Klein, Felix Bachmann, "Quality Attribute Design Primitives and the Attribute Driven Design Method", 4th International Wotkshop on Product Family Engineering Bilbao, Spain, 3-5 October 2001

[14] Steve G. Woods, Maro R. Barbacci, "Architectural Evaluation of Collaborative Agent-Based Systems", CMU SEI, 1999

◐ 저 자 소 개 ◑

고 현 희 (Koh Hyon Hee)

1992년 숙명여자대학교 전자계산학과 졸업(이학사)
2000년 숙명여자대학교 대학원 컴퓨터과학과 졸업(이학석사)
1993년 코오롱 정보통신 연구원
1999년 LG산전연구소 주임연구원
2002년 LG전자 정보통신 연구소 선임연구원
2002년 미국 카네기 멜론 대학교 MSE과정 수료
2001년 ~ 현재 숙명여자대학교 대학원 컴퓨터과학과 박사과정
관심분야 : 소프트웨어 아키텍처, 설계 패턴
E-mail : hhkoh@sookmyung.ac.kr

궁 상 환 (Kung Sang Hwan)

1977년 숭실대학교 전자계산학과 졸업(이학사)
1983년 고려대학교 대학원 전자정보차리학과 졸업(경영학석사)
1998년 충북대학교 대학원 전자계산학과 졸업(이학박사)
1981년 제2군수지원사령부 제원처리실 프로그램장교
1998년 한국전자통신원 책임연구원
1997년 미국 스탠포드 연구소(SRI) 초빙연구원
2000년 중소기업청 정보화지원과 전산사무관
2000년 미국 카네기 멜론 대학교 MSE과정 수료
2001년 ~ 현재 천안대학교 정보통신학과 조교수
관심분야 : 소프트웨어 구조, 분산시스템
E-mail : kung@cheonan.ac.kr

박 재 년 (Park Jae Nyon)

1966년 고려대학교 물리학과 졸업 (이학사)
1969년 고려대학교 대학원 고에너지 물리학과 (이학석사)
1972년 독일 함부르크대학 전산학과
1981년 고려대학교 대학원 고에너지 물리학과 (이학박사)
1972년 독일 국립 입자 가속기 연구소(DESY) 연구원
1979년 고려대학교 전자계산소 총간사
1983년 전남대학교 계산통계학과 교수
1998년 숙명여자대학교 전자계산소 원장
1983년 ~ 현재 숙명여자대학교 정보과학부 교수
2002년 ~ 현재 숙명여자대학교 이과대학 학장
관심분야 : 시스템 분석 및 설계, 정보구조도, 컴포넌트 기반 개발 방법론
E-mail : jnpark@sookmyung.ac.kr