2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

분산 환경에서 소프트웨어 컴포넌트의 동적인 재구성 프레임웍에 관한 연구

○
권 기현*, 최 형진*
* 강원대학교 대학원 전산과학과
E-mail: kweon@tongwon.ac.kr

A Study on Framework of Dynamic Reconfiguration of Software Component on Distributed Environment

Ki-Hyeon Kweon*, Hyung-Jin Choi*
* Dept of Computer Science, Kangwon Natl. Univ.

요 약

분산 환경에서 수행되는 컴포넌트 중에서 수행 시간이 매우 길거나 이전의 작업에 관련되어 연속적인 내용을 처리하는 경우 이러한 컴포넌트의 상태를 변경하기 위해서는 실행 시간에 동적인 변경 방법을 사용할 수밖에 없다. 동적인 변경에서 가장 중요한 고려요소는 컴포넌트의 상태나 통신 채널에 대한 상태 정보 유지하는 것이고 이것을 위해 실행 시간에 구조을 변경하는 메커니즘이 필요하다. 본 논문에서는 컴포넌트를 서비스하는 프레임웍을 소개하고 이 프레임웍 하에서 컴포넌트를 변경, 컴포넌트의 추가 및 이동을 가능하게 할 수 있는 구조에 대해 제시한다.

Ⅰ. 서 론

다 계층 구조로 분산환경에서 수행되는 애플리케이션은 각 소프트웨어 컴포넌트들의 협력에 의해 수행된다. 이들 각 컴포넌트간의 협력 수행은 컴포넌트간의 연결 설정(소켓, RMI, CORBA의 접속)에 의해 상호 운영되게 된다. 이 경우 특정 컴포넌트는 수행 시간이 매우 길거나 이전의 작업에 관련되어 연속적인 내용을 처리하는 경우가 있을 수 있는데 이러한 컴포넌트에 대한 상태를 변경하기 위해서는 실행 시간에 동적인 변경 방법을 사용할 수밖에 없다.

분산 애플리케이션의 동적인 변경은 실행 중에 있는 애플리케이션의 작업 내용을 변경하는 작업이다. 구조 변경의 예로는 컴포넌트를 변경, 실행중인 컴포넌트를 다른 컴퓨터로 이동 및 컴포넌트를 추가하거나 삭제하는 것을 들 수 있다. 동적인 변경에서 가장 중요한 고려요소는 컴포넌트 내부 또는 컴포넌트간의 통신 채널에 대한 상태 정보가 새로운 컴포넌트로 전이 될 때 이전의 애플리케이션의 상태와 같은 상태를 유지하기 위해서 전이 될 필요가 있다. 이것을 위해 실행 시간에 구조를 변경하는 메커니즘에 대한 것이 필요한데, 동적인 변경은 컴포넌트들이 상태 정보를 제공하거나 설치하는 능력을 요구하며 변경 동안에 통신에 대한 설정을 조율하는 메커니즘이 필요하다.

본 논문에서는 컴포넌트를 서비스하는 프레임웍을 소개하고 이 프레임웍 하에서 컴포넌트를 변경, 컴포넌트의 추가 및 이동을 가능하게 할 수 있는 구조에 대해 제시한다.

Ⅱ. 관련 연구

2.1 동적인 변경

1) 동적인 변경 유형

분산 애플리케이션 프레임웍 안에서 세 가지 일반적인 변경 유형에 대한 신뢰성 있는 변경 기법을 필요로 하게 된다[1][5][6].

① 컴포넌트 구현 변경 사항

시스템의 전체적인 구조에 대한 변경은 없으나 사용자는 개별적인 컴포넌트에 대한 변경을 요구할 수 있을 것이다. 수행시간이 긴 프로그램의 개선을 위해 수행 프로그램 안에서 지속성 있는 상태의 손실 없이 애플리케이션의 기능을 확장시키기 원하는 경우이다.

② 구조 변경

시스템의 논리적인 구조(컴포넌트의 구조나 토폴로지)는 변경 가능하다. 컴포넌트 인터페이스 사이의 연결 설정은 변경 될 수 있고 새로운 컴포넌트는 추가되고 이전

의 컴포넌트는 삭제되기도 한다. 물론, 구조적인 변경은 컴포넌트 구현 사항이 변경됨을 의미한다.

③ 기하 구조 변경

논리적인 애플리케이션의 구조는 고정되어 있으나 분산 아키텍처 상으로 맵핑되는 기하구조는 변경 될 수 있다. 기하 재배치 방법은 로드 분할이나 소프트웨어 결함 허용, 가용한 통신 자원에 대한 변경 적응, 보호되는 자원에 접근하는 프로세스의 재 할당에 유용한 기법이다.

2) 동적인 변경 환경의 요구사항

동적인 변경을 지원하기 위한 환경은 다음의 요구사항에 부합하여야한다.

① 이질적인 호스트에 대한 통신
② 현재의 구조에 대한 접근성
③ 연결 설정은 컴포넌트 자체에 포함되지 않음
④ 컴포넌트의 추가 및 삭제와 연결 설정
⑤ 전달 메시지의 접근
⑥ 동기화 작용에 대한 메커니즘
⑦ 컴포넌트 상태 정보에 대한 접근

III. 동적인 변경 프레임웍 구조

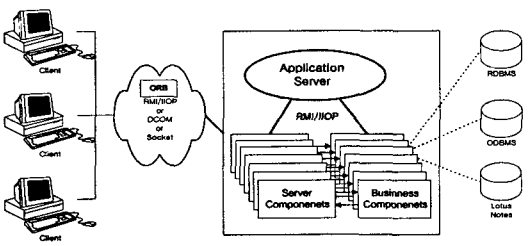
3.1 동적인 변경 프레임웍의 요구사항

동적인 변경을 지원하기 위한 프레임웍의 구조의 요구사항은 다음과 같다.

① 클라이언트가 서버의 부담을 최소로 하는 구조
② 비즈니스 컴포넌트의 추가 및 삭제가 용이한 구조
③ 비즈니스 컴포넌를 컴포넌트 서버에 의해 처리 구조
④ 복수개의 컴포넌트 서버를 이용한 부하 분산 구조
⑤ 컴포넌트 서버를 동적으로 관리할 수 있는 구조

3.2 동적인 변경 프레임웍의 구조

동적인 재구성 프레임웍의 구조도는 그림 1과 같이 컴포넌트 서버와 비즈니스 컴포넌트를 분리하여 컴포넌트 서버의 인터페이스 호출(RMI/IIOP)에 의해 비즈니스 컴포넌트가 요청되는 구조이다[4].

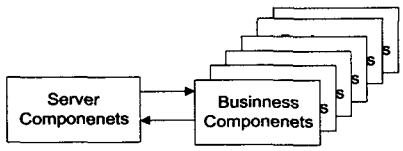


(그림 1) 컴포넌트의 동적인 변경 프레임웍 구조

3.3 비즈니스 컴포넌트의 관리

클라이언트는 서버상의 애플리케이션의 비즈니스 로직을 검사하는 비즈니스 컴포넌트에 대한 삭제가 용이한 구조를 가지는 것이 필요하다. 이를 위해 다음과 같은 사항을 만족하여야 한다.

① 애플리케이션의 인터페이스를 컴포넌트로 세분화
② 부분적인 비즈니스 컴포넌트의 추가 및 변경 용이
③ 비즈니스 컴포넌트의 오류를 지역화

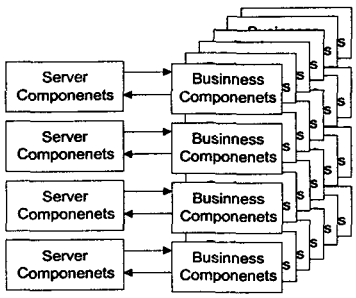


(그림 2) 비즈니스 컴포넌트의 관리

3.4 컴포넌트 서버의 관리

컴포넌트 서버는 비즈니스 컴포넌트를 관리하는 서비스 핸들러로 애플리케이션 서버에서 특정 애플리케이션에 대한 처리를 담당한다. 컴포넌트 서버에 대한 요구사항은 다음과 같다.

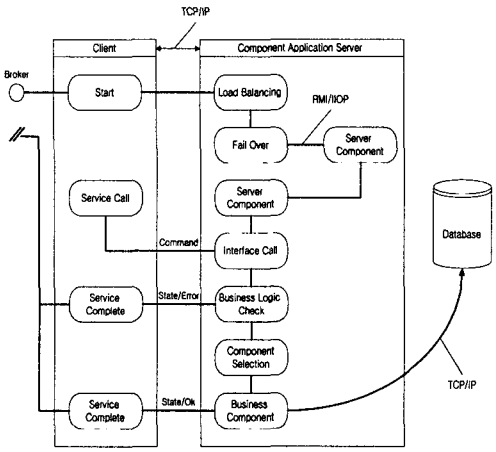
① 복수개의 컴포넌트 서버를 이용한 부하 분산 구조
② 비즈니스를 관리하는 컴포넌트 서버를 제공
③ 컴포넌트 서버를 복수개 유지 가능
④ 클라이언트의 요구를 컴포넌트 서버로 분배하여 클라이언트의 부하를 관리
⑤ 컴포넌트 서버는 애플리케이션 자체 시스템 내 뿐만 아니라 다른 컴퓨터 상에서도 수행이 가능



(그림 3) 컴포넌트 서버의 관리

3.5 UML을 이용한 애플리케이션 서버 모델링

클라이언트와 애플리케이션 서버의 Use Navigation 다이어그램은 다음과 같다[2][3].



(그림 4) Use Navigation 다이어그램

Use Navigation 다이어그램의 처리 흐름은 다음과 같다.

- 클라이언트는 접속을 개시
- 애플리케이션 서버는 부하 균등 및 연결 복구 처리
- 서버 컴포넌트를 선택하여 서비스 객체 획득
- 클라이언트는 명령을 보내 인터페이스 요청
- 서비스 요청에 대한 비즈니스 로직 검사
- 서비스 요청에 따른 비즈니스 컴포넌트 선택
- 비즈니스 컴포넌트는 데이터베이스와 연동하며 결과와 처리 상태를 클라이언트에 리턴

Ⅳ. 컴포넌트 변경 재구성 이벤트 설계

재구성 이벤트는 사용자로 하여금 재구성 동안에 컴포넌트간의 통신을 지연, 애플리케이션의 구조를 변경 그리고 한 컴포넌트에서 다른 컴포넌트로 상태 정보를 전이하도록 하는 것을 통하여 재구성 작업에 대한 애플리케이션 수준의 방법을 제공한다.

4.1 컴포넌트 변경을 위한 재구성 이벤트

컴포넌트 대치 재구성 작업을 위한 재구성 이벤트는 다음과 같다.

<표 1> 재구성 변경을 위한 이벤트

정의 이벤트	기 능
getComponentState()	컴포넌트의 상태 얻음
setComponentState()	컴포넌트의 상태를 지정
getNewComponent()	새 컴포넌트 생성
insertComponent()	컴포넌트 추가
deleteComponent()	컴포넌트 삭제

4.2 연결 설정 변경을 위한 재구성 이벤트

컴포넌트 연결 설정을 위한 재구성 이벤트는 다음과 같다.

<표 2> 연결 설정 변경을 위한 재구성 이벤트

정의 이벤트	기 능
getChannelState()	컴포넌트의 채널 정보 얻음
setChannelState()	컴포넌트의 채널 정보 지정

4.3 동기화 변경을 위한 재구성 이벤트

컴포넌트 동기화를 위한 재구성 이벤트는 다음과 같다.

<표 3> 동기화 변경을 위한 재구성 이벤트

정의 이벤트	기 능
getChannelState()	컴포넌트의 채널 정보 얻음
setChannelState()	컴포넌트의 채널 정보 지정

표 3의 이벤트 그룹은 애플리케이션 수준에서 컴포넌트 또는 인터페이스를 홀딩함으로서 재구조를 위한 동기화를 제공한다. 홀딩이 인터페이스에 적용되었을 경우에 이 인터페이스에 통신하려고 하는 컴포넌트는 블록킹된다.

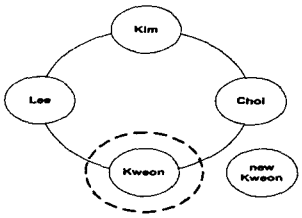
4.4 재구성 알고리즘

재구성 알고리즘은 새로운 컴포넌트의 파일이나 위치를 요청하여 새로운 버전의 컴포넌트를 생성하고 이전 버전의 모든 정보를 새 컴포넌트로 이전하고 이전의 컴포넌트를 삭제한 후 새로운 컴포넌트를 수행을 개시시킨다. 사이트 정보가 있는 경우는 해당 사이트에서 컴포넌트를 개시시키며 없는 경우는 이전의 컴포넌트와 같은 위치에서 컴포넌트를 개시시킨다.

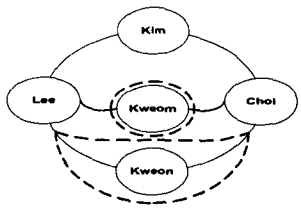
```
m =재구조 가능한 컴포넌트
M = { m | m은 재구성 가능한 컴포넌트 }
mi = {컴포넌트 m의 i번째 연결 설정 }
R = {변경되어지는 컴포넌트의 집합, R⊆M }
R' = {대치되는 컴포넌트의 집합}
각 루프에서 M은 (M-R)∪R'에 의해 변경됨

while(true) {
        R을 얻음
        for each m∈R {
                기능이 강화된 m'에 대한 이름,
                컴포넌트, 위치 구함
                기능이 강화된 m' 생성
        }
        for each m'∈R' {
                for 각 연결설정 mi' {
                        mi를 mi'에 바운드
                }
        }
        for each m'∈R' {
                m' 시작
                m 제거
        }
}
```

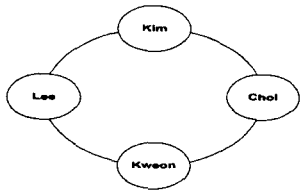
(그림 5) 컴포넌트 재구성 알고리즘



(그림 6) 컴포넌트 변경 준비



(그림 7) 컴포넌트 상태 변경



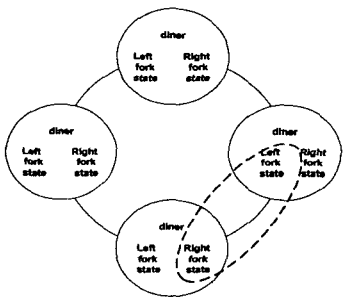
(그림 8) 컴포넌트 변경 완료

4.5 식사하는 철학자의 변경 재구성

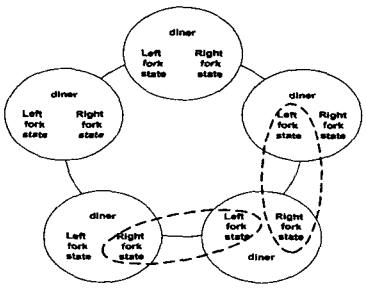
식사하는 철학자의 주변의 상태를 받아 새로운 컴포넌트로 대치 될 수 있다. 재구성 알고리즘에 의해 컴포넌트 변경의 경우를 식사하는 철학자로 적용하면 그림 6, 7, 8과 같다.

4.6 식사하는 철학자의 추가 재구성

컴포넌트가 추가되어 구조 변경되는 경우 인접한 컴포넌트의 상태 정보로 설정한 후 추가된다.



(그림 9) 컴포넌트 추가 전 상태



(그림 10) 컴포넌트 추가 후 상태

V. 결론 및 연구 방향

본 논문에서는 분산 3 계층 환경에서 컴포넌트를 서비스하는 동적인 재구성 프레임웍을 설계하고 동적인 재구성 이벤트를 소개하였다. 컴포넌트 서버의 동적 재구조에 요구되는 프레임웍을 설계함으로서 분산 환경에서 애플리케이션 서버가 컴포넌트를 서비스할 때 어떻게 동적인 재구조가 가능할 것인지 방향을 살펴보았으며 개별적인 컴포넌트에 대한 변경 요구에 대한 재구성 방법, 시스템의 논리적인 구조(컴포넌트의 구조나 토폴로지)에 대한 변경 그리고 분산 아키텍처 상으로 맵핑되는 네트웍 구조에 대한 재구성 방법을 제시하였다.

향후 연구 방향으로는 컴포넌트 재구조화에 따른 구현 및 컴포넌트의 네트웍에 전이 및 재구조화에 대한 연구가 필요하다.

참 고 문 헌

[1] Jeff sutherland, *Business Object Component Architecture: A Target Application Area for Complex Adaptive Systems Reserach*, Available by web server from http://jeffsutherland.org/oopsla98/boca_cas.html, 1998.

[2] Desmond D'Souza& Alan Wiils, *Objects, Component, and Frameworks with UML*, Addison-Wesley, 1998.

[3] Martin Fowler with Kendall Scott, *UML Distilled Applying the Standard Object Modeling Language*, Addison wesley, 1997.

[4] Gene Fodor, *Frameworks for component-based client/server computing*; Scott M.Lewandowski; ACM Comput. Surv. 30, 1 (Mar. 1998), Pages 3 - 27.

[5] P. Oreizy, R. N. Taylor, *On the Role of Software Architectures in Runtime System Reconfiguration,* In Proceedings of the International Conference on Configurable Distributed Systems (ICCDS 4). Annapolis, Maryland, May 4-6, 1998.

[6] H. Fossa, M. Sloman, *Implementing Interactive Configuration Management for Distributed Systems,* In Proceedings of 3rd International Conference on Configurable Distributed Systems (ICCDS), Annapolis, Maryland, May 1996.