Embedded환경에서 CORBA POA의 효율적인 자원활용을 위한 POA Monitoring기법

권성현° 김재훈
아주대학교 정보통신공학과
{shkwun°, jaikim }@ajou.ac.kr

A POA Monitoring scheme for effective resource utilization of CORBA POA in embedded environment

Sung-Hyun Kwun°, Jai-Hoon Kim
Graduate School of Information and Communication, Ajou University

요 약

최근 유비쿼터스(Ubiquitous) 시대가 도래함에 따라 홈 네트워킹(Home Networking)환경과 이동컴퓨팅 환경에 주로 사용되는 임베디드 시스템을 효과적으로 이용하기 위해서 미들웨어가 필요하게 되었다. 그러나 임베디드 시스템의 제한된 자원을 효율적으로 이용하기 위해 미들웨어의 최적화가 중요하다. 본 논문에서는 CORBA 컴포넌트(Component)중 POA(Portable Object Adapter)와 ORB(Object Resource Broker)사이의 상황을 모니터링 함으로써 기존 POA의 지속적인 서비스정책을 유동적으로 설정하고 특정 POA에 집중된 Servant들을 로드 밸런스를 한다. 이를 통해 임베디드환경에 맞는 효과적인 CORBA POA를 구성할 수 있는 기법을 제안한다.

1. 서 론

유비쿼터스 컴퓨팅이란 사용자가 다양한 형태의 컴퓨터를 원하거나 의식하지 않아도 접속할 수 있는 컴퓨팅 환경을 말한다. 현재 유비쿼터스 컴퓨팅은 차세대 정보기술의 핵심으로 떠오르면서 임베디드 시스템의 역할이 중요하게 되었다. 그러나 임베디드 시스템의 제한된 자원에 의해 이용 가능한 서비스의 영역이 축소 됨에도 불구하고 사용자의 다양한 서비스 요구를 만족시키기 위해 미들웨어의 중요성이 더욱 부각되고 있다. 그래서 CORBA[1]를 통해 유비쿼터스 환경에 적절한 서비스를 얻을 수 있도록 전통적인 CORBA 통합환경 보다, Microkernel 시스템처럼 필요한 모듈만 채택할 수 있는 유연한 시스템이 필요하게 되었다. 이러한 임베디드 환경에 맞추어진 CORBA와 관련된 연구로 Real-Time CORBA ZEN Project[2]가 진행 중에 있다. ZEN에서는 Plug-in을 활용한 시스템의 모듈화된 설계를 지향 하며 실시간 환경에서 예상할 수 있는 일의 흐름도를 찾아 효율적인 구성을 할 수 있는 것을 강조한다. 임베디드 환경에 최적화한 RT(Real-Time)-CORBA ZEN은 POA[1]의 해석(Demultiplexing)단계의 비용을 줄이기 위해 ORB[1]에서 실제 Skeleton[1]까지의 각 레벨을 최적화하고 ZEN구조를 모듈 형식으로 만들어 유연한 접근방법을 제공 한다. POA가 정책을 한번 설정하면 Servant [1]는 그 정책을 가지고 계속 작동 하는 특성이 있고, 자원 상태에 따라 한 POA당 Servant가 효과적으로 위치하기 위한 방법이 없다. 본 논문에서는 POA 모니터링을 해서 자원 상태에 따라 POA의 정책을 상황에 맞게 재설정 하고, 한쪽에 너무 많이 몰려 있거나 덩치가 큰 Servant가 있는 POA를 찾아서 새로운 POA를 만들고 Servant를 분산시킨다. 이를 통해 Servant를 효율적으로 배치하고 상황에 맞는 정책을 재설정 함으로서 효과적인 자원 분배와 빠른 응답 시간을 기대할 수 있다.

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진 되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크원천기반기술 개발사업의 지원에 의한 것임

2. 관련연구

2.1 Real-Time CORBA System

ZEN[1]은 ACE ORB(TAO)[3]의 많은 패턴들과 기술 등을 토대로 만들어진 open-source RT-CORBA이다. ZEN은 선택 가능한 컴포넌트(Component)들을 Plug-in 가능하게 하고, 자원을 적게 사용하여 확장의 용이성, 자원의 효율성, 구성의 유연성을 제공한다.

ZEN에서 POA를 위한 디자인 특징은 다음과 같다.

- 자원활용의 최소화
  ZEN POA의 가장 중요한 목표는 DRE(Distributed, Real-time, and Embedded)에 알맞은 미들웨어를 위해 자원을 적게 소모하는 디자인으로 만드는 것이다. ZEN에서 사용하는 각 응용프로그램을 위해 최소한의 메모리를 사용하도록 설계하고 이를 위해 각 응용 프로그램에서 필요한 미들웨어 코드부분만 사용하게 한다. 그리고 POA를 가상 컴포넌트(Core ORB를 구성하는 핵심 Component)와 함께 사용 하도록 하여, 각 응용프로그램들은 최소한의 메모리만 을 요구한다.

- 유연한 모듈 디자인
  POA가 Plug-in이 가능한 모듈식 디자인을 통해 변경된 CORBA 스펙을 쉽게 탑재 할 수 있다. 그래서 ZEN POA안에 가상 컴포넌트들은 특정 POA정책을 위한 구현부분과 함께 포함 시킬 수 있다.

- 실험이 용이한 디자인
  쉽게 변형 가능한 POA정책을 적용할 수 있는 시스템 디자인을 통해, 새로운 알고리즘과 자료구조를 가진 실험 적인 POA정책을 짧은 기간 내에 테스트할 수 있다.

기존의 ORB, POA, Servant간의 레이어(layered)환경은 해석 과정을 예상할 수 없고 서버 사용시간이 길어서 비효율적이기에 실시간 환경에는 맞지 않다. 따라서 현재 ZEN project에서는 클라이언트 요청의 객체 키(Object key)에 저장되어 있는 POA ID를 사용하여 요청에 맞는 POA를 찾는 효과적인 방법인 액티브 디멀티플렉싱(Active Demultiplexing) 전략을 통해 예상 가능한 상황을 만들어 실시간 환경에 맞는 POA를 구현하고 있다.[4]

본 논문에서는 ZEN의 모듈 디자인에 추가적인 디자인을 가정한다. ORB와 POA사이의 모니터링을 추가함으로써 좀 더 효율 적인 방안을 모색하고자 한다.

2.2 POA의 기능

CORBA는 OMG(Object Management Group)[5]로부터 나온 분산 객체에 대한 명세서이다. CORBA specification 3.0에 기반한 Object Adapter인 POA는 근본적으로 ORB와 서버 응용프로그램 사이의 중계자 역할을 한다. 즉, POA는 CORBA 객체에서 프로그래밍 언어로 구성된 Servant 까지 클라이언트 요청 메시지를 전달하기 위해 객체 참조(Object refenece), 객체 ID, Servant 사이를 번역해준다.[6]

POA는 다음과 같은 기능을 수행한다.

- 객체 참조(Object reference)생성
  CORBA 객체를 관리하기 위한 객체 참조를 만든다. 이것은 원격 클라이언트들이 분산환경에서 각각의 개체들을 요청 하기 위한 정보들을 포함하고 있다.

- 7개의 표준 POA정책으로 Servant의 행동을 결정
  POA는 Servant들의 행동을 결정하기 위해 다음 7개의 서버 정책을 사용한다.
  Thread, lifespan, object id uniqueness, object id assignment, servant retention, request processing, implicit activation

- 객체의 활성과 비활성
  CORBA 객체들을 만들었을 때 활성(activation)과 비활성(deactivation) 상태를 설정하여 서비스를 수행 유무를 결정한다.

- Servant의 생성과 제거
  POA가 객체를 위한 요청을 받았을 때 Servant를 생성한다. 그리고 Servant의 제거는 객체와 Servant 사이의 관계를 해제하는 것을 의미한다.

- Servant요청을 위한 해석
  클라이언트의 요청 메시지는 운영체제를 거치고 ORB를 거치는데 POA는 이 메시지를 해석해서 Servant에게 적절한 동작을 실행하도록 호출한다.

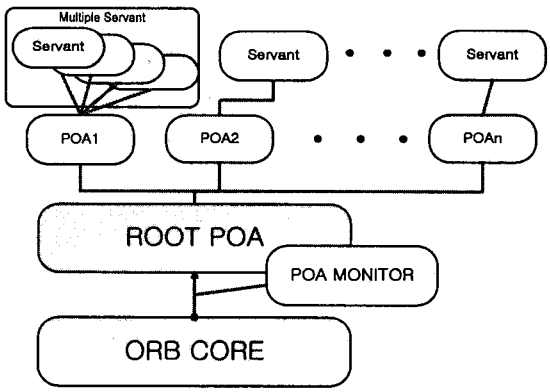
- SSI와 DSI
  POA는 Static Skeleton Interface(SSI)와 Dynamic Skeleton Interface(DSI)로부터 상속받아 Skeleton을 구성한다. 고정된 Interface나 (CORBA Server 응용프로그램 안에서 Compile됨) 동적인 Interface (Runtime때 객체 interface를 정해 Compile됨)를 가진 Skeleton은 그것이 나타내는 인터페이스에 의해 모든 동작에 대한 특정 정보 (argument type과 개수)를 포함하고 있다. 그래서 클라이언트 요청에 의한 특정 Servant에게 연결 시켜주는 POA를 돕는 역할을 한다.

3. POA Monitoring을 통한 효율적 자원할당 기법

임베디드 시스템은 제한된 환경 속에서 실시간 응답성과 성능의 효율성을 가져야 한다. 본 논문에서는 효율적 자원 할당 을 위해 다음과 같이 두 가지 방법을 제시한다.

1) POA 정책 재설정
POA 모니터는 ORB와 루트(Root) POA사이에서 클라이언트의 요청을 모니터링한다. 그리고 현재 POA가 사용하는 자원과 시스템에 남아 있는 자원을 조사한 후 현재 POA가 사용하는 정책으로 인해 자원을 많이 사용하면 그 정책을 사용하는 POA를 정지시키고 새로운 정책을 부여한다. 자원이 부족한 상황에서 클라이언트로부터 요청된 서비스가 빠른 응답을 요구하는 경우라면 현재 사용하는 POA들의 우선순위를 조사해서 우선순위가 낮은 POA를 정지시키거나 자원을 적게 차지하는 정책으로 변경시킨다. 즉, 현재 요청된 서비스를 우선 실행시켜 실시간을 요하는 환경에 효과적인 대응을 한다.

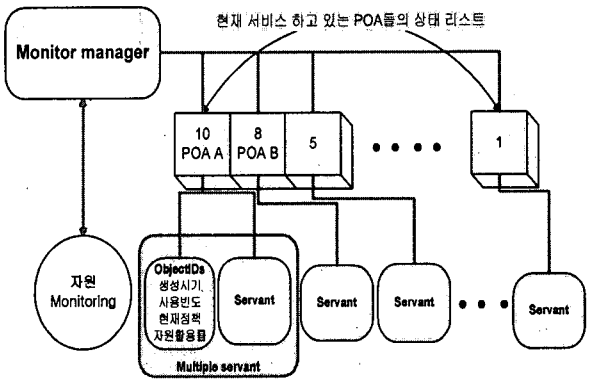
2) POA 로드 밸런스(Load balance)
같은 POA에 여러 개의 Servant가 올려 있을 경우나, 용량이 큰 Servant를 가지고 있는 POA에 Servant가 올려 있을 경우 <그림1>에서 POA1과 같이 한쪽 POA의 과부하로 인한 성능 저하가 발생 할 수 있다. 이를 막기 위한 방법으로 같은 역할을 하는 새로운 POA를 생성하여 부하를 분산 시킨다.



<그림 1> POA Monitor 구조도

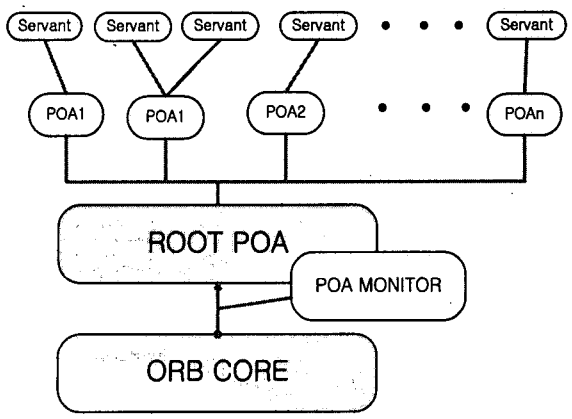
POA정책 재설정의 예로 지속적(Persistent)정책과 일시적(transient)정책을 들 수 있다. <그림1>과 같이 현재 ORB 코어를 통해서 클라이언트의 요청이 올 때, ORB와 루트 POA사이에 위치하는 POA 모니터는 현재 클라이언트 요청 빈도와 현재 자원 상태를 파악하고 있다. 기존에 사용되고 있는 객체는 지속적 정책으로 있으면서 서비스를 요청하는 클라이언트 객체에게 콜백(callback)정보를 넘겨주는 역할을 한다. 그러나 두 번째 객체에서 더 이상의 콜백정보를 원하지 않는 경우에도 첫 번째 객체가 지속적 정책으로 서비스 되어, POA에 계속 남아 있기 때문에 불필요하게 자원을 소모하는 문제가 있다. 이 문제를 해결하기 위해 <그림2>의 구조에서 볼 수 있듯이 현재 서비스되고 있는 상태 리스트에서 POA의 서비스 중요도 또는 사용빈도 등을 확인하여 그 객체를

비활성화시키거나 일시적 정책으로 변경시킨다. 그 이후 새롭게 들어온 클라이언트의 요청이나 이전에 일시 정지된 다른 Servant들을 POA 모니터를 통해 서비스 우선순위를 평가한 후 다시 활성화 시키고, 클라이언트가 요청한 서비스를 즉시 수행하거나 자원 상황을 보고 POA 메니저를 실행시켜 POA 메니저 큐(Queue)에 넣어 대기상태로 만든다. 이 방법을 통해 실시간성을 요구하는 임베디드 환경에서 실시간 응답 속도를 확보할 수 있을 뿐만 아니라 효율적인 자원분배가 가능하다.



<그림 2> POA Monitor 내부 구조도

POA 로드 밸런스의 예로 <그림1>의 POA1과 같이 하나의 POA에 여러 개의 Servant가 있는 경우와 덩치가 큰 Servant와 여러 개의 Servant가 같이 있는 경우 과부하가 발생 할 수 있다. 따라서 POA 모니터를 통해 <그림3>처럼 한쪽에 몰려 있는 Servant들을 분산시킴으로써 POA1의 과부하를 줄일 수 있다. 로드 밸런스를 하기 위해 <그림2>의 구조도에서 보여주는 것과 같이, 모니터 메니저가 정렬된 POA 리스트들을 조사하고 자원을 관찰한 결과값을 참조하여 Servant가 몰려 있거나 자원소모가 큰 Servant가 존재하는 POA를 찾는다. 그러므로 현재 POA의 상태리스트 우선순위, Servant의 자원 활용율, Servant사용 비율 등을 보고 모니터 메니저에게 새로운 POA를 만들도록 요청하면 모니터 메니저는 루트 POA에게 새로운 POA를 생성하게 한다. 새롭게 만들어진 POA에 한쪽에 몰려있던 Servant들을 관련된 Servant들로만 묶어서 넣거나, 단일 Servant의 대비 자원 사용 비율에 따라 분리해서 넣는다.



<그림 3> POA Monitoring을 통한 로드밸런스

POA 모니터에서 POA상태 리스트에 가중치를 정하는 방법은 Servant생성 시기와, 환경정보를 가지고 평가한다. <그림2>에서 보는 것처럼 루트 POA와 통신을 하는 모니터 매니저는 각 POA 상태 리스트에 있는 Servant 정보를 가지고 생성시기가 최근이거나 자원활용률과 사용빈도가 크면 우선순위 점수를 높게 정한다. 이렇게 우선순위를 정하여 생성시기, 사용빈도, 자원사용률 등을 비교하여 항상 최신의 정보를 가지고 POA 상태 리스트를 정렬한다.

결과적으로 POA의 현재 자원 상황에 맞지 않는 정책으로 자원을 낭비하거나, 한 POA에 과중한 Servant들이 몰려있을 때 새로운 POA를 생성해서 로드밸런스를 구현한다. 즉, 자원 모니터링을 통해 현재 남아 있는 자원정보와 POA에서 사용하는 자원활용률을 보고 정책 변경 및 로드밸런스를 한다. 이렇게 함으로써 실시간을 요하는 환경 및 제한된 환경을 가지고 있는 임베디드 시스템에서, POA 모니터링을 통해 효과적으로 자원을 분배 할 수 있고, 실시간성에 알맞은 응답을 할 수 있다.

4. 결론 및 향후 과제

본 논문에서는 임베디드 환경에서 CORBA POA를 효과적으로 사용하기 위해 POA 모니터링 기법을 제안한다. 이를 통해 임베디드 시스템 자원환경에 적합하고 실시간성을 보장하는 CORBA POA를 구성할 수 있다. 향후 과제로 POA 모니터를 RT-CORBA ZEN과 연계 가능한 기능으로 구현하고, POA 상태리스트를 정렬하기 위해 영향을 많이 받는 요소(사용빈도, 생성시기, 자원사용률, 정책에 따른 자원소모 비율 등)들과 POA 상태리스트의 알맞은 업데이트 시기를 구체적으로 찾을 것이다. 그 다음 합당한 상태 리스트를 생성하고, 로드 밸런스를 위한 최적화된 Servant 분할 규칙을 정할 것이다.

참고문헌

[1] CORBA specification 3.0, http://www.omg.org/technology/documents/corba_spec_catalog.html"Common Object Request Broker Architecture: Core specification", OMG, March 2004

[2] R. Klefstad, A.S.Krishna, D.C.Schmid, "Design and Performance of a Modular Portable Object Adapter for Distributed, Real-Time and Embedded CORBA Applications", Springer-Verlag GmbH, pp. 549 – 567, June 2003

[3] D.C.Schmidt, D.L.Levine and S.Mungee, "The Design and Performance of Real-Time Object Request Brokers", Computer Communications, vol. 21, pp.294-324, Apr.1998

[4] R. Klefstad, A.S.Krishna, D.C.Schmidt, A.Corsaro, "Towards Predictable Real-time Java Object Request Brokers", IEEE, 2003

[5] http://www.omg.org/

[6] Steve Vinoski, "New Features for CORBA 3.0" Communications of the ACM , 1988