MSF/CD기반 물리 설계 시 CID (Component Interaction Diagram) 성능 향상에 대한연구

정현규*, 이송희**, 최진영**
*고려대학교 컴퓨터정보통신대학원 디지털정보공학과
**고려대학교 컴퓨터학과
email: *ygestapo@korea.ac.kr, **{shlee, choi}@formal.korea.ac.kr

A Study of performance improvement for CID of Physical design based on MSF/CD

Hyun-Kyoo Jeong*, Song-Hee Lee**, Jin-Young Choi**
*Dept. of Digital Information Engineering, Korea University
**Dept. of Computer Science and Engineering, Korea University

요 약

과거의 기능 중심의 소프트웨어 개발방법론에서 벗어나 최근에는 서비스 중심의 컴포넌트 기반 소프트웨어 개발 (CBD: Component Based Development) 방법론이 새로운 개발 패러다임으로 각광을 받고 있다. 본 논문은 CBD 방법론 중 마이크로 소프트사의 MSF/CD 방법론을 기반으로 한 물리적인 설계를 할 때 기존의 CID (Component Interaction Diagram)에 Dispatch Agent 를 제안하였다. 실험 결과를 통해 제안된 기법이 component 서비스 속도를 향상시키고 보다 신속한 트랜잭션 처리를 제공함을 확인하였다.

1. 서론

인터넷 비즈니스 구현을 위한 새로운 IT 패러다임으로 인식되어온 'Web Service' 는 게임기, 개인용 PDA, 휴대폰에서부터 유무선 통신 단말기까지 활용되고 있으며 각 층의 사용자들이 생활과 노하우를 컴포넌트(component)화하여 한 차원 높은 서비스를 제공하고 있다. 소프트웨어에서 한계를 뛰어 넘는 컴포넌트는 독립적이고 부품화 되어있어 재사용이 쉬운 특징을 가지고 있어 향후 소프트웨어의 유일한 대안으로 전망되고 있다[1].

컴포넌트 기반 소프트웨어 개발 (CBD: Component Based Development) 방법론은 검증받은 컴포넌트를 조립함으로써 구축 비용을 획기적으로 줄이고 시스템 전체의 품질 확보와 함께 "Plug-in-Play" 방식의 SW 부품 교체를 통해 높은 시스템의 유지 보수성을 제공한다. 이러한 CBD 기반에서 시스템을 분석하고 설계하는데 있어 객체의 핵심이고, 비즈니스 로직을 담는 부분이 컴포넌트 설계이다. 따라서 비지니스 설계시 컴포넌트의 재활용성을 활용하고, 고객의 중요한 요구사항 중의 하나인 속도를 개선하는데 프로젝트의 성패가 달려 있다고 해도 과언이 아닐 것 이다.

본 논문에서는 수 년 동안 마이크로소프트에서 사용되어온 CBD 개발방법론MSF/CD (Microsoft Solution Framework/ Component Design)를 기반으로 하고 있으며, 컴포넌트의 물리 설계 시에 논리 설계의 자료를 실제 물리 CID를 작성하는데 있어 많은 제약 사항을 가진 물리적인 환경에서 UI와 컴포넌트간에 트래픽을 감소하고, 불필요한 Round-Trip을 줄이기 위해 Dispatch Agent 를 제안하였다. Dispatch Agent의 성능을 검증하기 위하여 실제 운영 시스템에서 각각의 데이터 건수 별 속도를 측정하고 각 Dispatch Agent를 사용하기 이전의 속도와 사용 후 응답 속도 개선을 통하여 보다 빠르게 향상된 물리 CID 설계를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 MSF/CD방법론 기반의 관련된 연구들에 대해 살펴보았다. 3장에서는 제안된 Dispatch agent를 실제 운영 시스템에 적용하고 실험 결과를 제시하였다. 마지막으로 4장은 본 논문의 결론 및 향후 연구 과제에 대해서 논의하였다.

2. 관련 연구

MCS/CD기반의 설계 방법론에 대한 연구는 미흡하지만, 실제적인 제품의 툴로 개발된 가장 대표적인 것은 Aurora.net(Desinger, Modeler, Developer)이다. Aurora.net은 객체지향적(Object Oriented) 개발 방법론을 적용하여 시스템을 개발할 때 소프트웨어 시스템을 분석/설계하는 비쥬얼 모델링 도구로 기존의 UML 모델링 솔루션들인 Rational Rose 나 Together가 J2EE 환경의 EJB를 만드는데 그 기능을 하였다면, Aurora.net Designer는 MS의 .NET 환경에서 운영되는 컴포넌트인 COM+를 설계, 생성하는 유일한 제품이다 [3]. 또한 [2]에서는 CBD 기술과 .NET 관련기술, Web 구축 기술을 기업의 업무를 전산화하는 ERP에 접목시켰다. 따라서 MSF/CD 기반 개발, .NET Platform, Web

기술이 적용된 시스템이 갖는 모든 장점을 수용하는 ERP 솔루션을 의미한다. 이는 개발 생산성을 향상시키고, 시스템 적용 후의 유지 보수가 용이한 시스템을 구축하고 표준 프로세스의 경영관리 업무의 전산화를 전사적 자원관리차원에서 접근하였다.

3. CID(Component Interaction Diagram) 설계 및 구현

3.1 MSF/CD 물리설계

MSF/CD는 크게 개념 설계(Conceptual Design), 논리 설계(Logical Design), 물리 설계(Physical Design)로 나뉘어지며 다음과 같다.

- 개념 설계는 기업이 비즈니스를 수행하기 위해서 필요한 상세 업무절차를 파악하는 것이 주목적이다. 이 단계의 주요 세부 활동은 업무 프로세스 모델과 초기 객체 모델을 설계한다.
- 논리 설계는 파악된 업무 프로세스를 대상으로 하여 크게는 누가 어떤 역할을 할 것이며, 세부적으로는 각각의 역할에 따라서 어떠한 일을 수행할 것인가를 정하는 일이다. 주요 세부 활동, 클래스 모델 설계, 논리UI설계, 논리 DB설계, 논리 인터페이스 설계의 활동을 한다.
- 물리 설계는 실제 실행 계획을 세우는 것으로 발생할 만한 여러 가지 실제 상황을 가정하고, 운영에 따른 문제점을 사전 파악하고 대책을 반영하는 일이다. 주요 세부 활동은 컴포넌트 모델 설계, 물리 인터페이스 설계, 물리UI설계, 물리 DB설계를 한다.

3.1 Dispatch Agent 테스트 환경

Dispatch Agent테스트를 위한 환경은 다음의 표1과 같다

<표 1> Dispatch Agent 실험 환경

Dispatch Agent 실험 환경		
H/W	기종	UnisysES3220
	CPU	Xeon 1.6GHz x 2way
	Memory	2G
	OS	Windows 2000
Middle Tier		.NET Framework , COM+

실험을 통해UI와 컴포넌트간의 데이터 전송 속도를 전송 데이터 크기 별로 측정한 결과 값은 표 2에서 보여진다.

<표 2> 성능 측정 테이블

(1분당)	총요청수	총연결수	평균RPS	페이지크기	평균TTLB [단위MS]
6KB/10Row	4,608	4,658	76.80	7.8KB	527.69
18KB/30Row	1,601	1,651	26.68	19.8KB	1,499.37
30KB/50Row	1,103	1,153	18.36	31.7KB	2,153.91
60KB/100Row	551	601	9.18	62.9KB	4,610.27
188KB/300Row	146	196	2.43	183.8KB	14,178.62
314KB/500Row	73	123	1.22	299.0KB	22,765.66

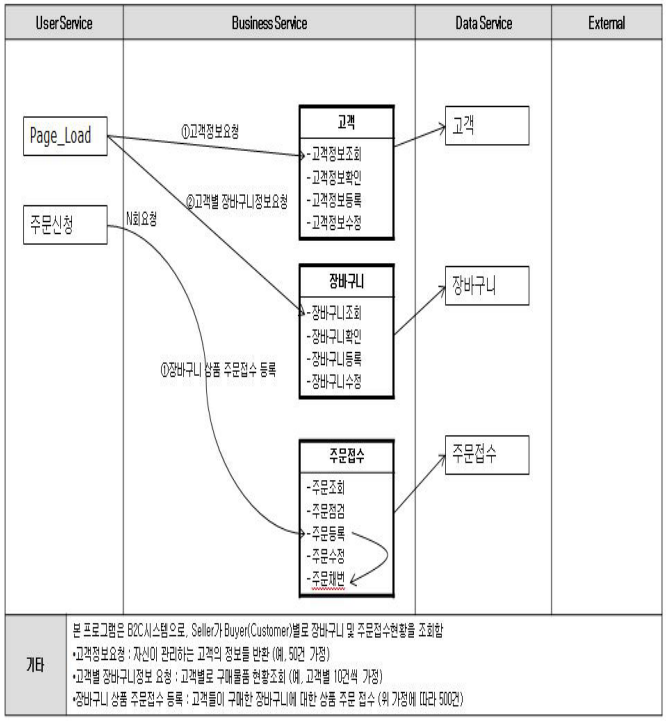
표2에서 RPS(Request Per Second)는 초당 처리 요청 수를 나타내며, TTLB(Time To Last Byte)는 Client가 서버로부터 마지막 응답을 받는 시간을 나타낸다. 6KB(10Row)를 전송할 때 1분당 총 요청 수는 4,608이고, 이는 초당 76.80RPS를 나타내며 마지막 바이트 응답 받는 시간이 527.69 Millisecond로 0.5초 정도 걸리는 것을 의미 한다. 또한 314KB(500Row)를 전송할 때 1분당 총 요청 수는 73이고, 이는 초당 1.22RPS를 나타내며 마지막 바이트 응답 받는 시간이 22,765.66 Millisecond로 22초 정도 걸리는 것을 의미 한다. 즉 적은 데이터 전송 시 처리 속도가 빠르며 많은 서비스 요청에 응답 할 수 있음을 나타낸다.

3.2 시나리오

Client에서 Form Load시 Seller가 자신의 관리 고객별로 일괄 조회하고, 고객별 장바구니 정보를 일괄 등록하는 시나리오를 CID로 작성 하였고 그림 1에서 보여진다. 이는 B2C시스템으로, Seller가 Buyer(Customer)별로 장바구니 및 주문 접수 현황을 조회 한다.

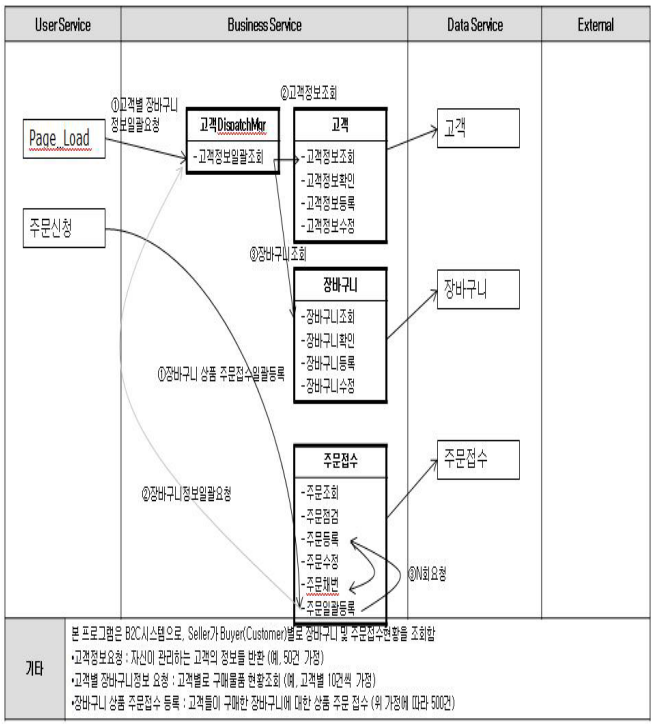
- 고객정보요청: 자신이 관리하는 고객의 정보들 반환 (예, 50건 가정)
- 고객별 장바구니정보 요청: 고객별로 구매물품 현황조회 (예, 고객별 10건씩 가정)
- 장바구니 상품 주문접수 등록: 고객들이 구매한 장바구니에 대한 상품 주문 접수 (위 가정에 따라 500건)

위의 시나리오를 MCS/CF의 물리 디자인 적용 시 기본 CID는 다음 (그림1)같이 나타난다.



(그림 1) 주문접수 CID

시나리오는2개의 Event로 구성되며 Page Load event와 주문신청 event로 구성 된다. 이를 본 논문에서 제안한 Dispatch Agent를 이용하여, 물리적 환경을 고려하여 재 작성하면 그림2와 같다.



(그림 2) Dispatch Agent 주문접수 CID

- 고객정보요청: 자신이 관리하는 고객의 정보들 반환 (예, 고객정보 10건 , 장바구니 정보500건)
- 장바구니 상품 주문접수일괄등록: 500개의 장바구니 내역을 Dispatch로 부터 일괄적으로 조회하여 서버에서 바로 처리

3.3 실험의 성능 비교

개선 후 Logic을 Pseudo Code로 (그림3-3)에 자세히 나타나 있다. 그림 1은 dsCustomer가 50건을 UI에 반환 하고UI로직은 다시 이를 서버로 보내어 장바구니 정보를 10건씩 개별적 요청UI의 주문신청 이벤트는 이전 Page_Load이벤트에서 넘겨받는 Basket의 건수만큼 반복하여 주문접수객체를 호출하는 불합리한 흐름을 나타낸다.

그림2는 고객정보를 한번에 일괄 요청하여 Round Trip을 줄이고, 고객정보 일괄 등록 처리가 서버 내부에서 일어나는 IPC메커니즘으로 네트웍 시간 손실을 줄였다. 그림 3은 Dispatch Agent 주문접수 Pseudo Code를 보여준다.

```
void Page_Load()
{
    고객DispatchMgr객체 customer = new
    고객DispatchMgr객체 ();
    DataSet dsBasketInfo =
    customer.고객정보일괄조회(@SellerID);
    Reponse.Write(dsBasketInfo);
}

DataSet 고객정보일괄조회()
{
    고객객체 customer = new 고객객체();
    장바구니객체 basket = new 장바구니객체();
    DataSet dsCustomer =
    customer.고객정보조회(@SellerID);
    DataSet dsBasketInfo =
    basket.장바구니조회(dsCustomer);
    return dsBasketInfo;
}

DataSet 고객정보조회()
{
    string SQL;
    SQL = "SELECT 고객정보 FROM 고객 WHERE
    SellerID=@SellerID";
    DataSet dsCustomer = EXEC(SQL);
    return dsCustomer;
}

DataSet 장바구니조회(DataSet 고객정보)
{
    string SQL;
    DataSet dsBasket;
    for (int i=0 ; i < 고객정보.Count ; i++)
    {
        SQL = "SELECT 장바구니정보 FROM 장바구니
              WHERE 고객ID=@고객ID";
         dsBasket.Add(EXEC(SQL);
    }
    return dsBasket;
```

```
}

void 주문신청()
{
    주문접수객체 order = new 주문접수객체();
    order.주문일괄등록(dsBasketInfo[i]);
}


void 주문일괄등록(주문정보)
{
   고객DispatchMgr객체 customer = new
   고객DispatchMgr객체 ();
    DataSet dsBasketInfo =
    customer.고객정보일괄조회(@SellerID);
    for(int i = 0; i < dsBasketInfo.Count; i++)
    {
         order.주문등록(dsBasketInfo[i]);
    }
}
```

(그림 3) Dispatch Agent 주문접수 Pseudo Code

표2를 바탕으로 각각의 CID의 Logic을 비교하면 표 3과 같다.

<표 3> CID의 Logic의 성능 비교

	주문접수CID	Dispatch Agent주문접수CID
Page_Load Event	1. 고객정보요청 : 50건 반환하여 2.1초. 이는 다시 아래 요청으로 입력 되므로 2.1초 추가 2.고객별장바구니 정보요청 : 고객별10건씩이므로 500건 반환하여22.7초 소요	1. 고객별 장바구니 정보일괄 요청 : 고객정보 10건 + 장바구니 500건 -->500건 22.7초 소요
소요시간	26.9 초	22.7 초
주문신청 Event	1. 장바구니 상품 주문접수등록 : 장바구니에 있는 물품 500건에 대하여 일괄 등록 (장바구니에 있는 물품등록 개개별로 등록할 경우 500번 네트워크통신 반복되며, 22.7초 이상 소요)	1. 장바구니 상품 주문접수일괄등록 : 500개의 장바구니 내역을 Dispatch로부터 일괄적으로 조회하여 서버에서 바로 처리. 서버의 요청은IPC에 의한 내부통신으로 네트워크 제약에 구애받지 않으므로 일괄처리 후, 성공여부 반환하는 것이므로 네트워크 시간 0.5초이내로 종료될 것 예측 가능
소요시간	22.7	0.5초
총 시간	49.6초	23.2초

총 소요시간이 49.6초 와 23.2초의 많은 시간의 차이가 나는 것을 알 수 있다. 이는 네크워크 Traffic 상의 Round Trip 과 통신 데이타 양을 줄이고, 가능한 서버에서 내부 처리에 의한 Logic으로 바꾼 결과, 속도의 향상을 가져 옴을 알 수 있다.

4. 결론 및 향후 과제

본 논문에서는CBD 개발방법론 중 마이크로소프트사의MSF/CD를 기반의 환경에서, 논리 설계의 자료를 이용하여 실제 물리적인 컴포넌트 CID를 작성할 때 UI와 컴포넌트간에 트래픽을 감소하고, 불필요한 Round-Trip을 줄이기 위한 Dispatch Agent 를 제안하였다. 본 논문에서 제안된 기법은 단순 조회 성 업무인 경우에 Round trip 을 줄였으며, 또한 서버 측에서 업무를 수행 하는 단순 서비스 호출에 비하여 53.23%이상의 속도 개선 효과가 실제적인 실험 결과를 통해서 보여주고 있다.

향후 연구 과제로는 기업의 분산 환경에서 서버 측의 여러 서비스들을 한번에 호출해 줄 수 있는 분산 Dispatch Agent의 적용에 대한 연구가 필요하다.

참고문헌

[1] 이윤철,“Coarse-Grained Agent를 이용한 웹 서비스 성능 향상 방안”, P3-P4, 2006

[2] 이은성, 박병형, 이승호, “MSF/CD를 기반으로 한 ERP 솔루션 개발 사례”, 정보처리학회지 제 10권 제 3호, 2003. 5

[3] 이승호, 박병형, 김영희, “MSF/CD방법론 기반의 분석/설계 툴 설계에 관한 연구”, 정보처리학회지 제 10권 제3호, 2003. 5

[4] 김권기, 김기범, 김도협, 김준환, 박수연, 손중모, 안진만, “ Microsoft .NET시스템 구축 방법론”, 정보문화사, 2002

[5] 김기범, 김세웅, 박수연, 손중모, 안진만, 이준규, 편도산, “MSF/CD기반의 컴포넌트 설계 방법론”, 정보문화사, 2004

[6] Dirk Krafzig ,Karl Banke , Dirk Slama, “Enterprise SOA: Service Oriented Architecture Best Practices”, Prentice Hall, p67-p81, 2004

[7] 최성, “CBD엔지니어링 기초”, 2005

[8] John Cheesman & John Daniels, “UML Components: A simple process for specifying component-based software”, Addison-Wesley, 2000