웹기반 정보시스템의 내부시스템 재구성을 위한 폼 클러스터링 방법론

최상수, 박현우, 이강수
한남대학교 컴퓨터공학과

Form Clustering Methodology for Internal System Reengineering of Web-based Information System

Sang-Su Choi, Hyun-Woo Park, Gang-Soo Lee
Department of Computer Engineering, Hannam University
E-mail : gcss09@se.hannam.ac.kr, tankboy@se.hannam.ac.kr, gslee@eve.hannam.ac.kr

요 약

최근 대부분의 정보시스템은 웹기반 정보시스템으로 이주하고 있으며 이의 개발과 유지보수시에 "웹 위기" 현상이 발생하고 있다. 이를 해결하기 위한 웹엔지니어링 기술 중 웹기반 정보시스템의 내부시스템을 재구성하기 위한 방법이 필요하다. 따라서 본 논문에서는 웹기반 정보시스템의 내부시스템을 재구성하기 위한 폼 클러스터링 방법을 제시한다. 폼 클러스터링 방법은 기존의 소프트웨어 분할 및 태스크 클러스터링 기술의 개념을 적용하여 웹 기능구조를 실제 하드웨어에 할당하기 위한 최적의 응답시간 성능을 갖는 웹 소프트웨어 구조를 생성하는 방법이다. 본 논문에서 제시하는 폼 클러스터링 방법은 웹기반 정보시스템의 내부시스템을 신규개발 및 유지보수시에 적용할 수 있다.

1. 서론

인터넷을 기반으로 하는 정보통신 인프라의 발달로 인하여 90년대 중반 이후부터는 대부분의 정보시스템이 ebXML 프레임워크 기반의 웹기반 정보시스템 형태로 이주되고 있다. 그러나, 웹기반 정보시스템 개발 및 유지보수에 대한 체계적인 개발방법의 부재로 인하여 개발비용 및 기간이 초과되고 유지보수 비용이 증가하는 웹 위기(web crisis) 현상이 발생함에 따라 이를 해결하기 위한 웹공학 기술이 활발하게 연구 및 개발되고 있다[1]. 이는 60년대 말의 소프트웨어위기 발생과 소프트웨어공학 기술의 등장 상황이 30년 만에 재현되고 있는 것이다.

웹기반 정보시스템(WBIS: web-based information system)은 e-비즈니스 시스템의 전형적인 구조이며 비즈니스 프로세스(BP: business process)에 따라 B2B, B2C, B2G 등으로 구분할 수 있다. 특히, e-비즈니스 시스템의 성능과 수익성은 이를 운영하는 조직의 성패를 좌우하기 때문에 체계적이고 비용-효과적인 웹공학 기술들이 연구 및 개발되고 있다[2~4].

특히, WBIS를 개발하거나 또는 유지보수시 BP 및 항해구조(Navigation Structure)를 모델링함으로써 사용자 관점에서 WBIS를 구성하고[5], 이를 기반으로 개발자 관점에서 WBIS의 내부시스템 구조를 구성할 수 있다.

WBIS의 내부시스템 구조는 웹의 Back-End 시스템에 해당하며, 고객에게 서비스를 제공하기 위한 구조로써 폼(페이지)으로부터 입력을 받고 처리하여 다시 폼에 그 결과를 보여주는 부분이다 이러한 내부시스템 구조는 특정 유형의 입력이 요구될 때에만 해당 기능이 제공되며, 호출된 이벤트들에 대하여 미리 정해진 순서로 각 입력에 대하여 처리가 이루어지기 때문에[6], WBIS의 내부시스템 구조를 재구성하기 위하여 기존의 실시간 시스템에 대한 소프트웨어 분할(software partition) 기술[6~8]과 태스크 클러스터링(task clustering)[9~11] 및 할당(allocation)[8,12,13] 기술을 활용할 수 있다.

이러한 배경에서, 본 논문에서는 웹공학 기술중의 일부인 WBIS의 내부시스템 구조 재구성을 위한 폼 클러스터링 방법론을 제시한다. 폼 클러스터링 방법은 기존의 소프트웨어 분할 기술과 태스크 클러스터링 기술의 개념을 적용 및 응용하여, BP 및 항해구조 모델링 결과로부터 생성된 웹 기능구조를 실제 하드웨어로 할당하기 위한 최적의 성능값을 갖는 웹 소프트웨어 구조로 클러스터링하는 방법이다.

본 논문의 2장에서는 관련연구로써 기존의 소프트웨어 분할 및 태스크 클러스터링 기술에 대한 기본 개념을 조사하고, 3장에서는 폼 클러스터링 방법론을 제시한다. 4장에서는 폼 클러스터링 방법론을 적용하여 일반적인 "온라인 쇼핑몰"의 내부시스템 구조에 대한 폼 클러스터링 사례를, 끝으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 소프트웨어 분할(Software Partition)

소프트웨어 분할(Software Partition)이란 사용자 관점을 반영하는 논리적 모듈 집합을 소프트웨어 개발자 관점을 반영하는 소프트웨어 태스크(task) 집합으로 매핑하는 것을 의미한다[6~8]. 따라서, 소프트웨어 분할은 구현된 시스템으로부터 사용자 요구사항들에 대한 추적성(traceability)을 제공하는 소프트웨어 개발을 위한 실용적인 방법이다.

소프트웨어 분할을 위한 기준은 ① 응답시간의 최소화, ② 누적되는 실행시간의 감소, ③ 효율적인 자원의 이용, ④ 태스크들 사이의 통신 감소, 그리고 ⑤ 병목(bottleneck) 프로세서에 대한 부하 감소(off-loading) 등이다. 다시 말하면, 소프트웨어 분할의 목

(*) 본 연구는 한국과학재단 목적기초연구사업 지역대학 우수과학자 지원 연구(R05-2001-000-01492-0) 지원으로 수행된 결과의 일부임.

표는 로드밸런싱(load balancing)과 인터럽트 서비스 시간(대기시간)을 제한하는 것이라 할 수 있다.

특히, Nissanke는 소프트웨어 분할을 위하여 2가지 스키마를 제시하였다[7]. 스키마 I 은 관련된 태스크들이 종료되었는지 여부와 상관없이 오직 이미 실행된 모듈들에 따라서 태스크 실행가능성이 확립되는 반면, 스키마II는 어떠한 태스크도 종료 이전에 결과를 다른 태스크들에 의존하지 않는다는 규칙 내에서 스키마 I 과 동일한 내용이 엄격하게 확립된다.

2.2 태스크 클러스터링(Task Clustering)

태스크 클러스터링(Task Clustering)이란 병렬 머신에 존재하는 높은 통신 오버헤드로 인하여 작은 태스크들을 하나의 큰 태스크로 합체함으로써 전체 실행시간을 최소화하는 절차를 말한다[9~11]. 특히, 프로세서의 수가 무한하거나 태스크 중복을 허용하는 경우 태스크 클러스터링 문제는 NP-hard 문제와 유사하다. 일반적으로, 태스크 클러스터링 알고리즘은 크게 중복을 허용하지 않는 방법과 중복을 허용하는 방법으로 구분할 수 있다[8].

중복을 허용하지 않는 태스크 클러스터링 알고리즘에는 DSC (dominant sequence clustering) 알고리즘과 ETF(earliest task first) 알고리즘 등이 제안되었고, 중복을 허용하는 태스크 클러스터링 알고리즘으로는 PY 알고리즘과 Greedy 알고리즘 등이 제안되었다[8].

그러나, 웹은 순수 어플리케이션(프로그램) 이외에도 추가적인 객체(HTML, 그래픽, 오디오, 비디오 등)들이 포함된 형태이기 때문에, 기존의 소프트웨어 분할 기술과 태스크 클러스터링 기술들을 WBIS에 그대로 적용할 수 없다.

따라서, 본 논문에서는 기존의 소프트웨어 분할 기술과 태스크 클러스터링 기술의 개념을 적용하여 WBIS의 내부시스템 구조를 재구성하기 위한 폼 클러스터링 방법론을 제시한다.

3. 폼 클러스터링 방법론

3.1 문제 정의

폼 클러스터링(form clustering)이란 WBIS에 대하여 BP(비즈니스 프로세스) 및 항해구조를 모델링하고 분석한 결과를 토대로 생성된 웹 기능구조를 실제 하드웨어에 할당하기 위한 최적의 웹 소프트웨어 구조를 생성하는 방법이다.

본 논문에서는, 폼 클러스터링을 수행하기 위한 기본 가정사항을 다음과 같이 정의한다.

[가정 1] 웹 소프트웨어 "컴포넌트"는 1개 이상의 웹 "기능객체"(모듈)들로 구성된다.

기존의 컴포넌트 공학기술(예컨대, Sun 사의 EJB와 IBM의 샌프란시스코 등)에서도 하나의 컴포넌트는 1개 이상의 객체로 구성되기 때문에 [가정 1]은 타당하며, 이를 확장하여 다음과 같은 가정사항을 추가로 정의할 수 있다.

[가정 2] 하나의 웹 소프트웨어 컴포넌트는 하나의 "소프트웨어 서버"이다.

[가정 3] 하나의 "하드웨어"에는 1개 이상의 "소프트웨어 서버"가 포함된다.

본 논문에서는, 소프트웨어 컴포넌트를 "소프트웨어 서버"라 정의하며, 실제 하드웨어에는 웹 기능객체들로 구성된 소프트웨어 서버가 할당되게 된다. 따라서, [가정 2]와 [가정 3]으로부터 다음과 같은 정의를 유도할 수 있다.

[정의 1] 기능 객체(FO: functional object) = {FO_1, FO_2, ···, FO_i, ···, FO_n}

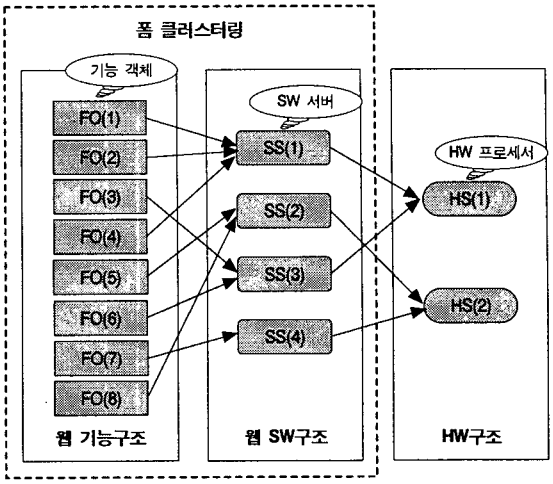

그림 1. 폼 클러스터링 문제 개념도

[정의 2] 소프트웨어 서버(SS: software server) 또는 소프트웨어 컴포넌트 = {SS_1, SS_2, ···, SS_j, ···, SS_m}

[정의 3] 하드웨어 서버(HS: hardware server) = {HS_1, HS_2, ···, HS_k, ···, HS_p}

[정의 2]에서 SS_j = {SS_i | $SS_i \in FO$} 즉, SS는 FO의 파워셋(power set)에 해당하며, [정의 3]에서 HS_k = {HS_i | $HS_i \in SS$} 즉, HS는 SS의 파워셋(power set)에 해당한다. 따라서, 본 논문에서 제시하는 폼 클러스터링 방법은 FO들을 실제 HS에 할당하기 위한 최적의 성능값을 갖는 SS로 클러스터링하는 방법이라 할 수 있으며, 전체 개념도는 <그림 1>과 같다.

3.2 폼 클러스터링 방법론

폼 클러스터링 방법은 WBIS의 웹 기능구조로부터 실제 하드웨어에 할당하기 위한 최적의 웹 소프트웨어 구조를 생성하는 방법이다. 따라서, 본 논문에서는 생성된 SS 내의 FO간의 "결합도"(coupling)를 최소화하고 SS 자체의 "응집력"(cohesion)을 극대화시키기 위하여 각 FO들의 성능특성(입력 및 출력 링크수, 실행시간, 기능목표)을 고려하여 폼 클러스터링을 수행한다.

특히, WBIS은 "8초 규칙"(즉, 8초 이내에 결과가 처리되어야 함)과 같이 응답시간이 매우 중요하기 때문에, 폼 클러스터링을 위한 제약조건(SS에 대한 최대허용 실행시간, 최대 클러스터링 가능한 FO의 수)이 개발자에 의하여 이미 설정된 상태라는 가정 하에서, 응답시간을 최소화하고 SS간의 복잡도를 최소화하는 것을 목표로 한다.

(1) 웹 기능구조

폼 클러스터링을 수행하기 위하여, 해당 WBIS에 대하여 BP 및 항해구조를 모델링하고 분석[5]한 결과를 토대로 생성된 웹 기능구조를 입력으로 사용한다. 웹 기능구조를 모형화한 WFSD(web functional structure diagram)의 정의는 다음과 같다.

[정의 4] WFSD = {FO, Arc}

- FO = {FO_1, FO_2, ···, FO_n}는 특정 기능을 수행하는 FO들의 집합이며, 끝이 둥근 사각형으로 표기한다. 각 FO에는 기능객체명, 입력 및 출력 링크수, 실행시간, 기능목표가 매핑된다. 즉,

FO' = {FO_Name, I_Link, O_Link, RT, FT}

- Arc는 FO의 실행순서를 나타내며 단방향 또는 양방향 화살표로 표기한다. 하나의 FO는 다수의 아크를 가질 수 있으며, 아크의 수는 FO의 I_Link 및 O_Link 속성과 대응관계에 있다.

WFSD에 대한 세부 속성들(FO_Name, I_Link, O_Link, RT, FT)은 WFSM(web functional structure matrix) 형태로 표현할 수 있다. 즉, WFSM은 WFSD와 동일한 정보를 가지며 WFSD를 위한 내부 자료구조로 활용할 수 있다.

본 논문에서 제시하는 폼 클러스터링 방법은 WFSD와 WFSM가 주어졌을 때 FO들을 실제 HS에 할당하기 위한 최적의 성능값을 갖는 SS를 구하는 문제이며, 다음과 같이 2단계 접근방법을 사용한다.

(2) 1단계 : 기능목표 기반 그룹화

우선, 주어진 WFSD와 WFSM으로부터 동일한 FT(기능목표)를 갖는 FO들을 그룹화한다. 이때, 고려해야할 사항은 다음과 같다.

- 응집력을 최대화하기 위하여 동일한 기능목표를 갖는 FO들을 그룹화한다(informational cohesion: 구성된 클러스터의 모든 원소들이 단일 기능을 발휘하도록 하여 응집력 최대화).
- 그룹화된 FO들의 실행순서가 유지되는지 여부를 검사하여, 유지되지 않을 경우 FO를 서로 분리하여 그룹화한다(data coupling: 그룹화된 클러스터 사이의 통신을 위해 변수목록이나 테이블을 주고받도록 하여 결합도 최소화).

(3) 2단계 : 제약조건 검사 및 재그룹화

1단계 클러스터링을 통하여 생성된 클러스터 각각에 대하여, 개발자들에 의하여 설정된 제약조건(MAX_FO, MAX_RT)을 만족하는지 여부를 검사하여 위배되는 클러스터를 재그룹화한다. 이때, 고려해야할 사항은 다음과 같다.

- 1단계 클러스터 각각에 대하여 초기에 설정된 제약조건을 만족하는지 여부를 검사하고, 위배되는 클러스터 각각에 대하여 내부 FO들을 재그룹화한다.
- 재그룹화할 FO의 상세 속성 중에서 I_Link 및 O_Link 수가 큰 FO들을 제약조건을 고려하여 그룹화한다(입력 및 출력 링크수가 큰 FO들을 그룹화함으로써 생성된 클러스터간의 복잡도를 줄일 수 있다).

2단계 클러스터링을 통하여 생성된 전체 클러스터 구조가 웹 소프트웨어 구조가 되며, 생성된 클러스터 각각이 소프트웨어 서버(SS)가 된다.

(4) FCA: form clustering algorithm

본 논문에서 제시하는 폼 클러스터링 방법을 정형화한 알고리즘은 <표 1>과 같다.

결론적으로, 폼 클러스터링 방법은 해당 WBIS에 대하여 BP 및 항해구조를 모델링하고 분석한 결과를 토대로 생성된 웹 기능구조로부터, FO들을 실제 HS에 할당하기 위한 최적의 응답시간 성능을 갖고 복잡도를 최소화하여 SS를 생성하는 방법이다. 특히, 웹 기반 정보시스템의 응답시간 성능은 이를 운용하는 기업의 성패를 좌우하기 때문에, 본 논문에서 제시한 폼 클러스터링 방법은 웹기반 정보시스템의 내부시스템의 신규개발 및 유지보수시에 활용할 수 있다.

4. 사례연구

본 장에서는 일반적인 "온라인 쇼핑몰"에 대하여 BP 및 항해구조를 모델링하고 분석한 결과를 토대로 생성된 웹 기능구조 모델(WFSD, WFSM)에 대하여 폼 클러스터링 사례를 보인다.

표 1. FCA : form clustering algorithm

```
// 1단계 기능목표 기반 그룹화
// 제약조건 초기화
MAX_FO = '                    // 최대 클러스터링 가능한 FO 수
MAX_RT = '                    // 최대허용 실행시간

// n개의 FO에 대하여 m개의 클러스터 생성
FOR i=1 to n, FOR j=1 to m DO
  // 클러스터의 기능목표(FT)와 실행시간(RT) 초기화
  Ci = null, Ci.FT = null, Ci.RT = 0, Ci.FO = 0
  IF Ci.FT = Null THEN                // 새로운 클러스터 생성
    Ci = Ci + FOi
    Ci.FT = FOi.FT
    Ci.RT = Ci.RT + FOi.RT
    Ci.FO = Ci.FO + 1, i = i + 1
  ELSE IF Ci.FT = FOi.FT AND Ci.FO < MAX_FO THEN // 클러스터에 추가
    Ci = Ci + FOi
    Ci.RT = Ci.RT + FOi.RT
    Ci.FO = Ci.FO + 1, i = i + 1
  ELSE
    j = j + 1
  END_IF
END_DO END_DO

// 2단계 제약조건 검사 및 재그룹화
// 생성된 n개의 1단계 클러스터에 대하여 최대허용 실행시간 검사
FOR i=1 to n DO
  IF Ci.RT > MAX_RT THEN          // 최대허용 실행시간 초과
    FOR j=1 to MAX_FO, FOR k=1 to m, FOR x=1 to y DO
      // 2차 클러스터 초기화
      Cx = null
      // 최대허용 실행시간을 초과하지 않는 FO들의 조합에 대하여
      IF Ci.FO, RT + Cj.FO, RT <= MAX_RT THEN
        // FO간의 복잡도 계산 결과 가장 큰 FO들의 조합
        Cx = Check(Ci.FO, Cj.FO)
        k = k + 1
      END_IF
    END_DO END_DO END_DO
  END_IF
END_DO
```

조를 모델링하고 분석한 결과를 토대로 생성된 웹 기능구조 모델(WFSD, WFSM)에 대하여 폼 클러스터링 사례를 보인다.

개발자들은 폼 클러스터링을 수행하기 위한 제약조건으로 최대 허용 실행시간 = 30, 최대 클러스터링 가능 FO수 = 3으로 설정하였으며, 입력으로 사용될 '온라인 쇼핑몰'에 대한 WFSD와 WFSM은 각각 <그림 2>, <표 2>와 같다.

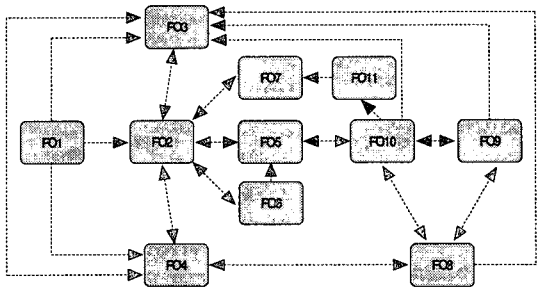

그림 2. 사례 웹 기능구조 다이어그램(WFSD)

표 2. 사례 WFSD에 대한 매트릭스(WFSM)

FO_ID	FO_Name	I_Link	O_Link	RT	FT
FO1	HTTP 서버	0	3	10	웹 서비스
FO2	홈페이지 관리자	6	5	10	웹 서비스
FO3	브라우징 관리자	5	1	10	웹 서비스
FO4	검색엔진	4	3	30	검색
FO5	로그인 관리자	3	2	20	인증
FO6	등록엔진	1	1	10	인증
FO7	내역조회 관리자	2	1	30	검색
FO8	선택 관리자	3	4	20	카트
FO9	보관 관리자	2	3	10	카트
FO10	주문 관리자	3	5	20	카트
FO11	지불 관리자	1	1	10	지불

주어진 WFSD와 WFSM에 대하여 1단계 클러스터링을 수행한 결과 <그림 3>과 같이 6개의 클러스터가 생성되었다.

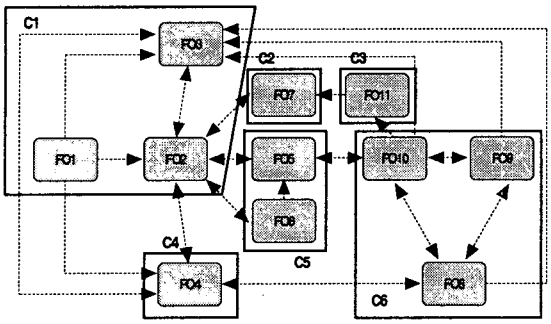

그림 3. 사례 웹 기능구조에 대한 1단계 클러스터링 결과

<그림 3>에서, FO4(내역조회 관리자)와 FO7(검색엔진)은 동일한 기능목표(검색)를 가지지만, 실행순서를 유지하기 위하여 개별적인 클러스터로 그룹화되었다.

2단계 클러스터링을 수행하기 위하여, 1단계에서 생성된 각 클러스터에 대하여 제약조건 만족 여부를 분석한 결과 <그림 4>와 같이 C6이 제약조건(MAX_RT = 30)에 위배됨을 알 수 있다.

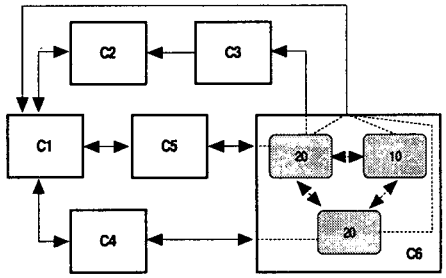

그림 4. 제약조건에 위배되는 클러스터(C6)

따라서, C6에 대하여 제약조건을 만족시키도록 2단계 클러스터링을 수행하면 최종적으로 7개의 클러스터가 생성되며, 이렇게 생성된 클러스터 각각이 SS에 해당된다. 폼 클러스터링을 통해 생성된 웹 소프트웨어 구조는 <그림 5>, <표 3>과 같으며, 생성된 SS는 다음과 같다.

SS1 = {FO1, FO2, FO3}　　SS2 = {FO7}
SS3 = {FO11}　　SS4 = {FO4}
SS5 = {FO5, FO6}　　SS6 = {FO8}
SS7 = {FO9, FO10}

5. 결론

본 논문에서 제시하는 폼 클러스터링 방법은 웹기반 정보시스템의 내부시스템 구조를 재구성하기 위한 방법으로 제시하였으며, 제시한 방법의 사용방법을 보이기 위하여 일반적인 "온라인 쇼핑몰"을 예로 하여 전체 적용과정을 보였다.

폼 클러스터링 방법은 기존의 소프트웨어 분할 기술과 태스크 클러스터링 기술의 개념을 적용하여, 웹 기능구조를 실제 하드웨어에 할당하기 위한 최적의 응답시간을 갖는 웹 소프트웨어 구조를 생성하는 방법이며, 웹기반 정보시스템의 내부시스템을 새로 개발하거나 유지보수시에 적용할 수 있다.

그러나, 본 논문에서 제안하는 폼 클러스터링 방법에서는 수행알고리즘에 대한 정형화가 부족하며 FO(기능객체)의 실행시간을 계산하기 위한 방법과 실제 FO간에 전송되는 메시지 유형 및 크기에 따른 통신 지연시간을 고려하지 않았다. 따라서 이에 대한 연구를 향후 연구과제로 남긴다.

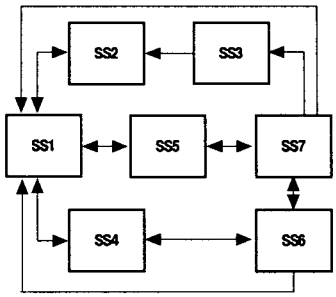

그림 5. 사례 웹 기능구조에 대한 폼 클러스터링 결과
(웹 소프트웨어 구조)

표 3. 웹 소프트웨어 구조에 대한 상세 속성 매트릭스

SS_ID	SS_Name	I_Link	O_Link	RT
SS1	웹 서버	5	3	30
SS2	거래정보검색 서버	2	1	30
SS3	지불 서버	1	1	10
SS4	제품검색 서버	2	2	30
SS5	인증 서버	2	2	30
SS6	카트 서버	2	3	20
SS7	주문 서버	2	4	30

참고문헌

[1] A. Ginige, S. Murugesan, "Web Engineering: An Introduction, ", *IEEE Multimedia*, Vol.8, No.1, pp.14-18, Jan-Mar. 2001.

[2] P. Killela, *Web Performance Turning*, O'reiley, 1998.

[3] N. V. Flor, *Web Business Engineering*, Addison Wesley, 2000.

[4] D. Menasce, *Scaling for E-business: Technologies, Models, Performance, and capacity Planning*, Prentice-Hall, 2000.

[5] 최상수, 이강수 외, "디지털 컨텐츠를 포함한 웹기반 정보시스템의 재구성을 위한 비즈니스 프로세스 및 항해 모델링", 디지털컨텐츠학회 논문지, Vol.3, No.1, 2002(게재예정)

[6] J. P. Huang, "Modeling of Software Partition for Distributed Real-Time Application", *IEEE Transactions on Software Engineering*, Vol.11, No.10, pp.1113-1126, Oct. 1985.

[7] B. Indurkhya, et. al, "Optimal Partitioning of Randomly Generated Distributed Programs", *IEEE Transaction on Software Engineering*, Vol.12, No.3, pp.483-495, Mar. 1986.

[8] Dr. Nimal Nissanke, *Realtime Systems*, Prentice Hall, 1997.

[9] M. A. Palis, et. al, "Task Clustering and Scheduling for Distributed Memory Parallel Architectures", *IEEE Transactions on Parallel and Distributed Systems*, Vol.7, No.1, pp.46-55, Jan. 1996.

[10] A. Gerasoulis, T. Yang, "A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors", Journal of Parallel and Distributed Computing, Vol.16, pp.276-291, 1992.

[11] W. Lowe, W. Zimmermann, "On Finding Optimal Clusterings of Task Graphs", In Aizu International Symposium on Parallel Algorithm and Architecture Synthesis, pp.241-247, IEEE Computer Society Press, 1995.

[12] W. W. Chu, L. M-T. Lan, "Task Allocation and Precedence Relations for Distributed Real-Time Systems", *IEEE Transactions on Computers*, Vol.C-36, No.6, pp.667-679, June. 1987.

[13] C. E. Houstis, "Module Allocation of Real-Time Applications to Distributed Systems", *IEEE Transactions on Software Engineering*, Vol.16, No.7, pp.699-709, Jul. 1990.