모바일 환경에서 웹 서비스 품질보장을 위한 동적 분산적응 프레임워크

이 승 화[†] · 조 재 우[††] · 이 은 석[†††]

요 약

최근 무선기기의 다양한 제약 사항을 극복하고, 수시로 변화하는 주변 환경에 따라 항상 적절한 서비스 레벨을 유지하기 위한 상황인식형 적응 서비스가 중요한 이슈가 되고 있다. 그러나 대부분의 기존 연구들은 적응 모듈이 클라이언트나 프록시, 서버 중 한 위치에 집중되어 있어, 사용자가 증가하는 경우에 작업부하가 한 곳에 몰리고, 결과적으로 사용자의 요청에 대한 응답시간을 증가시키는 문제를 가지고 있었다. 따라서 본 논문에서는 적응 모듈을 클라이언트, 프록시, 서버 측에 분산배치하고 시스템의 상황을 모니터링하여, 가장 적절한 시스템이 작업을 처리하는 분산적응 프레임워크를 제안한다. 이를 통해, 사용자가 증가하는 경우와 같이 작업부하가 증가하는 상황에서도 보다 빠른 적응작업이 가능해지며, 부하가 분산되어 안정적인 시스템 운영이 가능해진다. 본 논문에서는 제안프레임워크의 평가를 위해 프로토타입을 구현하고, 크기가 큰 이미지파일을 포함하는 멀티미디어 기반 학습콘텐츠를 이용하여 분산처리를 테스트하였다. 그리고 서버의 과부하를 시뮬레이팅하여, 기존 적응시스템들과의 응답시간과 시스템 안정성측면의 비교를 수행하였으며, 이 실험결과를 통해 제안프레임워크의 유효성을 증명하였다.

키워드 : 상황인식, 적응형 시스템, 분산 컴퓨팅, 유비쿼터스 컴퓨팅

Dynamic Distributed Adaptation Framework for Quality Assurance of Web Service in Mobile Environment

Seunghwa Lee[†] · Jaewoo Cho[††] · Eunseok Lee[†††]

ABSTRACT

Context-aware adaptive service for overcoming the limitations of wireless devices and maintaining adequate service levels in changing environments is becoming an important issue. However, most existing studies concentrate on an adaptation module on the client, proxy, or server. These existing studies thus suffer from the problem of having the workload concentrated on a single system when the number of users increases and, and as a result, increases the response time to a user's request. Therefore, in this paper the adaptation module is dispersed and arranged over the client, proxy, and server. The module monitors the context of the system and creates a proposition as to the dispersed adaptation system in which the most adequate system for conducting operations. Through this method faster adaptation work will be made possible even when the numbers of users increase, and more stable system operation is made possible as the workload is divided. In order to evaluate the proposed system, a prototype is constructed and dispersed operations are tested using multimedia based learning content, simulating server overload and compared the response times and system stability with the existing server based adaptation method. The effectiveness of the system is confirmed through this results.

Key Words : Context-Awareness, Adaptive System, Distributed Computing, Ubiquitous Computing

1. 서 론

오늘날 컴퓨팅 기술의 급격한 발전을 통해 다양한 형태의 무선 컴퓨팅 기기가 확산되고 있으며, 우리는 이를 통해 언제 어디서나 인터넷에 접속할 수 있게 되었다. 또한 기존에 고정된 장소에서 유선인터넷과 데스크 탑 PC를 이용하여 행해지던 많은 작업들이 시공간의 한계를 극복하여 자유로운 작업형태로 변화 하고 있다. 그러나 이러한 무선 컴퓨팅 기기는 '휴대성'을 위해 작은 화면 사이즈, 제한된 배터리 용량과 같은 제약사항을 가지고 있으며, 아직까지 낮은 CPU와 적은 메모리 용량과 같은 제한된 성능을 가지고 있다. 또한 무선 네트워크 환경은 고정된 유선 네트워크 환경과는 다르게 사용자의 이동에 따라서 성능이 수시로 변하는 특성을 갖고 있다.

※ 본 연구는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발사업과 정보통신부 및 정보통신연구진흥원의 대학IT연구센터 지원사업 IITA-2005-(C1090-0501-0019)의 연구결과로 수행되었음.
† 준 회 원 : 성균관대학교 대학원 컴퓨터공학과(박사과정)
†† 준 회 원 : 성균관대학교 대학원 컴퓨터공학과(석사과정)
††† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2006년 7월 27일, 심사완료 : 2006년 9월 18일

따라서 이러한 무선 컴퓨팅 환경의 다양한 제약사항을 극복하고 항상 적절한 서비스 레벨을 유지하기 위한 적응시스템 관련연구가 전 세계적으로 많은 기관에 의해 다양한 방법으로 진행되고 있다[4-13]. 이 연구들은 적응 모듈의 위치에 따라 크게 클라이언트 측 적응, 프록시 측 적응, 서버 측 적응의 형태로 구분할 수 있다. 그런데 이들 대부분의 연구는 적응 모듈이 한 곳에 몰려있어 사용자의 요청이 증가할 경우 부하가 집중되는 문제를 가지고 있다. 실제로 멀티미디어 웹 컨텐츠의 포맷을 변환하는 작업 등은 많은 리소스를 필요로 하는 작업이며, 이는 사용자의 요청에 대한 응답시간을 증가시키는 결과를 초래한다.

따라서 본 논문에서는 이러한 문제를 해결하기 위해 클라이언트, 프록시, 서버에 각각 적응 모듈을 분산시키고, 모듈간 상호작용을 통해 각 기기의 상황에 따라 적응 작업을 적절히 분산 처리하는 프레임워크를 제안한다. 이를 통해 사용자가 증가하여 서버에 부하가 집중 되는 등 각 위치에서 적응작업의 수행이 어려운 상황에도 일관된 속도의 적응 작업이 가능해지며, 병렬 작업처리에 의해 적응 모듈이 한 곳에 집중되어 있는 경우에 비해 빠른 적응 작업이 가능해진다. 그리고 결과적으로 전체 시스템의 안정적인 운영이 가능해지며, 또한 다양한 상황에서 각 적응 작업을 가장 잘 수행할 수 있는 모듈이 선택됨으로써, 보다 정확한 적응 결과물이 만들어진다.

본 논문에서는 제안프레임워크의 평가를 위해 프로토타입을 구현하여, 크기가 큰 이미지 컨텐츠를 포함하는 멀티미디어 기반 학습컨텐츠의 변환작업을 수행하였다. 그리고 서버에 과부하를 시뮬레이팅하여 기존에 한 곳에 적응 모듈이 집중되어 있는 방식과 응답시간, 서버의 리소스 상태변화를 비교하였으며, 실험결과를 통해 시스템의 유효성을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 대표적인 관련 연구들을 적응 모듈의 위치에 따라 세 가지 형태로 구분하여 장단점을 기술하였으며, 3장에서는 이러한 관련연구의 문제를 보완하기 위해 설계된 분산 프레임워크를 제안하였다. 4장에서는 제안 시스템의 평가를 위해 프로토타입 구현과 수행된 실험결과를 소개하였으며, 결론과 향후 과제를 5장에 기술하였다.

2. 관련 연구

최근 유비쿼터스 컴퓨팅 패러다임의 등장과 함께, 상황인식(Context-Awareness)이나 조용한 컴퓨팅(Calm Computing)[1], 오토노믹 컴퓨팅(Autonomic Computing)[2]의 개념이 많은 애플리케이션에 적용되고 있다. 이들은 공통적으로 시스템 내외부의 환경 변화를 인식하여, 그 변화에 대한 적절한 대응방안을 스스로 결정하고 수행하는 특성을 가지고 있으며, 시스템의 행동을 변경하거나 시스템의 모듈을 재구성한다. 이러한 상황-적응형 시스템의 패러다임은 다양한 분야에 적용되고 있으며, 특히 최근에는 언제 어디서나, 어떤 기기를 사용하든지 관계없이, 현재 사용자의 주변 환경에 보다 적합한 형태로 정보를 전송해주기 위해, 적응형 웹 정보 시스템에 적용되고 있다.

웹 컨텐츠 적응 시스템들은 사용자의 취향과 컴퓨팅 기기의 특성 등을 반영하여, 가장 적합한 컨텐츠를 연결해주며, 동적인 환경변화에 따라 일정한 서비스 레벨을 유지하기 위해 적응작업을 수행한다.

각 기기의 특성과 사용자의 취향에 맞게 컨텐츠를 개인화하는 가장 명확한 방법은 웹 서버에 접속할 수 있는 사용자의 모든 상황을 고려하여 여러 버전의 컨텐츠를 미리 작성하는 것이다. 그러나 이는 컨텐츠 저작자의 많은 노력을 필요로 하며, 서버 측에 많은 스토리지 낭비를 초래할 있다. 또한 오늘날과 같이 급변하는 컴퓨팅 환경에서 새롭게 출시될 기기들을 예측하고 대처하는 것은 매우 어려운 일이다. 따라서 대부분의 시스템들은 동적으로 컨텐츠를 변환하며, 이러한 기존연구들은 적응을 수행하는 모듈의 위치에 따라 크게 클라이언트 측 적응, 프록시 측 적응, 서버 측 적응으로 구분할 수 있다[3].

먼저, 클라이언트 측 적응 시스템은 적응모듈이 사용자의 기기에 내장되어, 시스템 내외부의 환경변화를 모니터링하고 그에 맞게 다운로드받은 컨텐츠를 변환하여 보여주는 방식이다[4]. 이 방식은 사용자 기기의 여러 속성들과 사용자 데이터를 외부로 전달하지 않아도 되므로 프라이버시와 관련된 컨텍스트 정보를 보호할 수 있다. 또한 사용자의 동적인 상황변화를 가장 빠르게 인식할 수 있으므로, 내부의 컴포넌트를 재구성하여 기능을 변환할 때 주로 이용된다. 또한 기기에 내장된 스타일 시트를 이용하여 컨텐츠를 사용자의 취향이나 기기의 특성에 맞게 재배치하여 볼 때 이용될 수 있다[5]. 본 연구에서 클라이언트는 일반적으로 휴대용 단말기를 의미하며, 데스크탑 PC에 비해 상대적으로 열악한 계산 처리 능력을 가지고 있다. 따라서 컨텐츠의 포맷 변환과 같은 적응은 어려운 일이며, 이미 클라이언트에 컨텐츠가 도착한 이후에 이를 변환하는 것은 컨텐츠의 크기가 변하지 않기 때문에 무의미한 작업일 수 있다. 이처럼 클라이언트 적응방식은 내용의 구성과 관련되고, 계산 리소스를 많이 소비하지 않는 부분적인 적응만이 적합하다.

다음 서버 측 적응 방식은 컨텐츠를 가지고 있는 서버에서 적응이 이루어지는 방식으로, 사용자의 상황정보에 따라 미리 작성된 적절한 컨텐츠를 연결해주거나, 동적으로 컨텐츠를 생성한다[6, 7]. 이 방식은 컨텐츠의 구조를 잘 알고 있는 저작자에 의해 변환이나 변환 규칙이 만들어지므로, 보다 정확한 적응 컨텐츠를 생성할 수 있다는 특징을 가지고 있다. 그러나 서버가 컨텐츠의 제공 외에 '적응'의 기능을 수행함으로써, 서버의 작업부하가 가중될 수 있고, 결과적으로 응답시간을 증가시킬 수 있는 문제가 있다.

프록시 측 적응 방식은 이러한 서버의 작업부하를 줄이기 위해, 컨텐츠 적응을 수행하는 기능을 서버로부터 독립시켜, 서버와 클라이언트 사이에서 중간 관문역할을 하는 프록시 서버에서 수행하는 방식이다[8-11]. 이를 통해 서버에 집중

되던 부하를 줄일 수 있으며, 서버가 상황적응 능력이 없고, 단순히 정보만을 제공하는 일반 컨텐츠 제공자일 경우, 프록시 서버는 중간 매개체 역할을 하며, 클라이언트에게 적응 컨텐츠를 제공해준다. 또한 이 방식은 클라이언트의 계산능력이 모자라 수행하지 못하는 다양한 형태의 적응서비스 (예: 번역, 컨텐츠 포맷 변환 등)이 가능하다. 그러나 이는 서버에서 컨텐츠에 대한 자세한 메타데이터가 제공되지 않으면 잘못된 적응결과물을 만들어 낼 수 있으며, 따라서 컨텐츠의 구성 내용을 몰라도 수행 가능한 번역, 컨텐츠 포맷 변환 등에 적합하다.

이처럼 기존 적응 시스템들은 적응 모듈이 클라이언트나 프록시, 서버 중 한 위치에 집중되어 있으며, 사용자가 증가하거나 어느 한쪽의 계산능력이 낮은 경우에는 많은 양의 작업부하가 한쪽으로 치우쳐져, 결과적으로 응답시간이 증가하는 문제가 있다. 본 제안시스템은 이러한 기존 연구의 단점을 보완하기 위해 설계되었으며, 다음 장에서 자세한 특징을 설명한다.

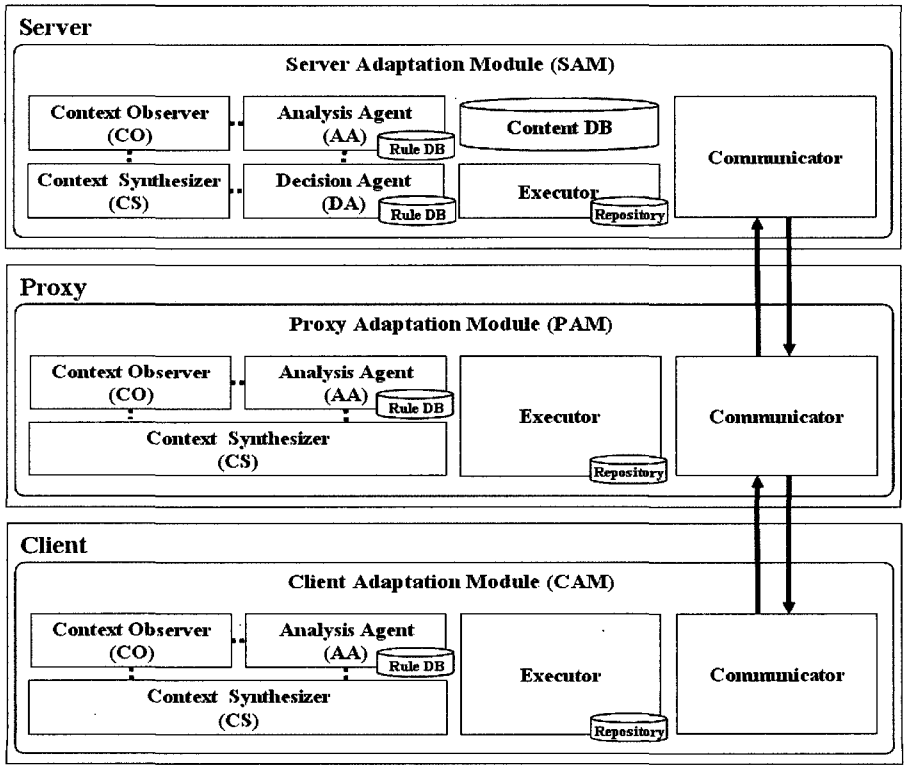
3. 동적 분산적응 프레임워크

제안프레임워크는 모바일 환경에서 발생할 수 있는 다양한 제약사항을 극복하기 위해, 사용자의 다양한 컨텍스트 정보를 인식하여 그에 적절한 컨텐츠 적응 서비스를 수행한다. 또한 제안프레임워크는 기존 연구에서 적응 모듈이 한 곳에 집중되어 있어 발생했던 여러 문제를 효율적으로 개선하기 위해 설계되었으며, 보다 빠른 응답속도와 각 시스템의 안정성을 유지하는 것을 목표로 한다.

3.1 프레임워크 구성

제안 프레임워크는 크게 클라이언트 기기에 내장되는 클라이언트 적응모듈(Client Adaptation Module: CAM)과 클라이언트와 웹 서버 중간 계층에 존재하는 프록시 적응모듈(Proxy Adaptation Module: PAM), 그리고 서버에 내장되는 서버 적응모듈(Server Adaptation Module: SAM) 세 부분으로 구성되며, 전체적인 구조는 (그림 1)과 같다.

- Context Observer (CO): 클라이언트, 프록시, 서버의 적응 모듈에 각각 삽입되어 시스템의 리소스 정보와 작업양과 같은 동적으로 변화하는 상황 정보를 수집한다. 각 시스템의 컨텍스트 정보는 적응 서비스와 적응을 수행하는 모듈의 위치를 결정하기 위해 사용된다.
- Analysis Agent (AA): CO에 의해 수집된 정보를 기반으로 현재 상황을 분석한다. 또한 클라이언트의 AA는 각 상황에서 클라이언트가 수행할 수 있는 적응작업을 결정한다. 이 결정은 서버에서 최종 적응작업의 리스트와 위치를 결정할 때 가장 우선시된다.
- Context Synthesizer (CS): 각각의 CO로부터 수집된 정보와 AA로부터 분석된 결과를 취합한다. 이는 RDF (Resource Description Framework)를 사용하여 기술된다.
- Decision Agent (DA): 취합된 상황 정보와 저장된 규칙을 기반으로, 현재 필요한 적응 서비스의 종류와 강도를 선택하고, 이를 수행하는 위치를 결정한다.



(그림 1) 제안프레임워크의 전체적인 구조

- Rule DB: 각 상황에 대한 대응방안이 규칙으로 저장된다.
- Repository: 컨텐츠 적응을 수행하는 다양한 컴포넌트들이 저장된다.
- Content DB: 웹 컨텐츠의 원본이 저장되는 공간을 의미한다.
- Executor: Repository에 저장된 적응 서비스 컴포넌트들을 호출하여 실행한다.
- Communicator: 각 모듈 간 상호작용을 수행한다.

3.2 프레임워크 동작과정

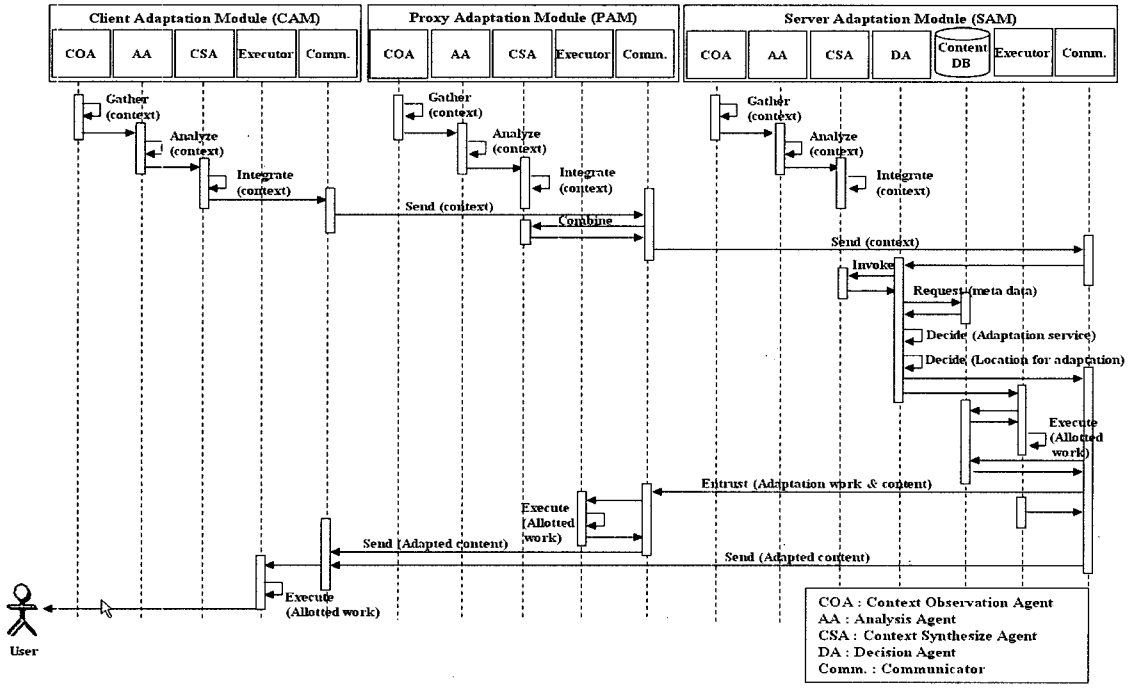
제안프레임워크의 전체적인 동작과정은 다음과 같다. 먼저 각 시스템에 내장된 Context Observer(CO)는 자신의 컨텍스트 정보를 수집한다. 수집되는 정보는 <표 1>과 같으며, 이 중 클라이언트의 정적인 성능을 표현하는 부분은 HTTP 1.1을 이용해서 서버 측으로 표현 가능한 컨텍스트이다. 그러나 본 제안프레임워크에서는 동적인 컨텍스트도 수집하고, 적응서비스의 확장에 따라 계속 새로운 컨텍스트를 추가하기 위해, Context Observer를 이용한다.

수집된 정보 중 일부는 Analysis Agent(AA)에 의해 계산되고 분석된다. 클라이언트의 AA는 분석과정에서 자신이 수행 가능한 적응서비스를 결정하는 기능이 추가되며, 동적인 리소스의 변화를 계산하는 자세한 과정은 3.3절에서 설명한다. 이후, Context Synthesizer(CS)는 CO에 의해 수집된 정적인 정보와 AA에 의해 계산된 정보를 취합한다.

사용자가 정보를 요청할 때, Client Adaptation Module (CAM)은 그 동안 수집된 클라이언트 측의 컨텍스트 정보를 Proxy Adaptation Module(PAM)로 전송한다. 이후, 프록시의 CS는 클라이언트의 요청정보에 자신이 수집한 프록시의 컨텍스트를 첨부하고, 이 정보를 Server Adaptation Module (SAM)로 전송한다. 서버의 Decision Agent(DA)는 요청받은 정보의 메타데이터를 분석하고, 요청 메시지와 함께 첨부된 클라이언트, 프록시의 컨텍스트 정보, 서버의 컨텍스트 정보를 기반으로 적응 서비스의 종류와 강도를 결정한다. 이 결정은 전문가에 의해 미리 구축된 규칙을 이용하며, 규

〈표 1〉 각 위치에서 수집되는 컨텍스트 정보

위치	컨텍스트 종류	적 용
클라이언트	CPU-RAM 용량, 현재 사용량	적응서비스의 종류와 강도를 결정할 때 반영
	디스플레이 사이즈, 표현 가능한 컬러	
	표현 가능한 미디어포맷	
	직접수행을 원하는 적응서비스	페이지 재구성과 같은 적응작업의 분산을 결정할 때 반영
프록시	CPU-RAM 용량, 현재 사용량, 작업 Queue 사이즈	미디어 포맷 변환과 같은 리소스를 많이 소비하는 작업의 분산을 결정할 때 반영
	수행 가능한 적응서비스 목록	
서버	CPU-RAM 용량, 현재 사용량, 작업 Queue 사이즈	미디어 포맷 변환과 같은 리소스를 많이 소비하는 작업의 분산을 결정할 때 반영



(그림 2) 제안 프레임워크의 기본 동작과정을 나타내는 시퀀스 다이어그램

칙의 자가 확장에 대한 연구[14]가 향후에 통합될 것이다.

적응서비스의 종류와 강도가 결정된 이후, 제안프레임워크는 컨텍스트 정보를 기반으로 적응이 수행되는 위치를 결정한다. 이때 클라이언트가 요청한 1차 전략이 가장 우선시되며, 이후 각 위치의 작업부하를 계산하여 적응작업이 분산 처리된다. 이에 따라 Content DB의 원본 컨텐츠는 각 위치로 분배된다. 이러한 제안프레임워크의 전체적인 동작과정이 (그림 2)에 시퀀스다이어그램으로 표현 되어있다.

3.3 적응 서비스의 위치결정 과정

제안 프레임워크는 적응 작업의 위치결정을 위해서 각 시스템의 리소스 사용량과 작업부하를 반영한다. 먼저 리소스 사용량의 분석을 위해서 기존 그리드 컴퓨팅 관련연구[15]에서 사용하는 식을 이용하였다. 그리드 컴퓨팅은 작은 컴퓨팅 파워를 가진 다수의 기기를 연결하여 고성능의 컴퓨팅 성능을 얻기 위한 연구이다. 따라서 하나의 작업을 다수의 기기로 분산시키고, 이를 스케쥴링하며 다시 통합하는 등의 알고리즘 들은 본 제안시스템의 동작과정과 유사하다. 본 연구에서는 클라이언트, 프록시 서버, 컨텐츠 제공서버가 각각 분산된 작업자가 되며, 보다 효율적인 작업의 할당을 위하여, 각 시스템은 자신의 상태를 모니터하여 최종 결정자인 컨텐츠 서버에게 알린다. 이후 서버는 이 정보에 다음과 같은 식을 적용하고, 계산 결과에 따라 작업량의 분배와 작업수행 위치를 결정한다. 본 논문에서는 리소스 계산을 위해서 각 시스템의 CPU와 RAM 정보를 사용하며, 각각 식(2)와 식(3)을 이용하여 계산을 수행한다. 그리고 전체 리소스 사용량 계산은 식(1)을 이용한다.

$$Current_{resource} = \frac{Current_{CPU} + Current_{RAM}}{W_{CPU} + W_{RAM}} \quad (1)$$

$$Current_{CPU} = W_{CPU}(1 - CPU_{load})\frac{CPU_{speed}}{CPU_{\min}} \quad (2)$$

$$Current_{RAM} = W_{RAM}(1 - RAM_{usage})\frac{RAM_{size}}{RAM_{\min}} \quad (3)$$

여기서 W는 CPU와 RAM에 할당된 가중치이다. 예를 들어, 결정된 적응작업 *image_converter*가 소비하는 리소스를 10으로 볼 때 CPU, RAM에서 소비되는 가중치는 각각 4, 6으로 정의할 수 있다. 이는 적응서비스의 개발자가 결정하는 것을 가정하였다. W_{CPU}는 CPU에 할당된 처리 양을, CPU_{load}는 CPU의 현재 사용량을 나타내며, CPU_{speed}는 CPU의 동작 속도를 나타내고, $CPU_{\min}$은 요청 작업에 필요한 CPU의 최소 양을 나타낸다. 또한 W_{RAM}은 RAM에 할당된 처리 양을, RAM_{usage}는 RAM의 현재 사용량을, RAM_{size}는 RAM의 사이즈를, $RAM_{\min}$은 요청 작업에 필요한 RAM의 최소 양을 나타낸다.

이러한 리소스 상태와 함께 작업 수행 위치를 결정하는데 반영되는 또 하나의 척도는 현재 수행 중인 작업의 양이다. 이는 현재 작업 큐의 길이를 통해서 파악한다. 큐의 길이는 Little 법칙[16]을 통해 정의된 식 (4)와 같은 큐 길이 계산식을 이용하여 얻는다.

$$r = \lambda Tr \quad (4)$$

여기서 r은 현재 큐의 길이를 나타내며, λ는 시스템에 도착하는 패킷의 도착시간의 비율 을, Tr은 평균 대기시간을 나타낸다.

이처럼 리소스사용량과 큐의 길이는 현재 시스템이 부하를 어느 정도 부담하고 있는지를 파악하기 위해 필요하다. 제안 프레임워크는 이를 기반으로 분산 처리의 결정을 수행한다. 최종 결정을 위해 사용되는 계산식은 식 (5)와 같다.

$$System_{context} = Current_{resource} + r \quad (5)$$

4. 구현 및 평가

본 논문에서는 제안프레임워크의 평가를 위해 프로토타입을 구현하여 멀티미디어기반의 학습 컨텐츠 제공 시스템에 적용하였으며, 응답속도와 시스템 안정성측면의 이득을 평가하였다.

프로토타입 구현에 사용된 기기의 사양은 다음과 같다. 먼저, 클라이언트 디바이스는 현재 일반적으로 많이 사용되고 있는 PDA를 대상으로 하였으며, 240 * 320의 화면크기와 400MHz의 CPU, 128M RAM을 가진 HP 5500 PDA를 이용하였다. 그리고 프록시서버는 AP나 AP주변에 위치하는 서버를 의미하며, 일반적인 관문역할 외에 제안시스템의 적응 서비스를 수행한다. 실험에서 프록시 서버는 2.8GHz CPU와 512 RAM을 가진 일반적인 데스크탑 PC를 이용하였으며, 컨텐츠 서버는 3.4GHz CPU와 2G RAM을 가진 HP 4300 워크스테이션을 이용하였다.

본 실험에서, 각 서버는 Microsoft Windows 2000을 운영체제로 사용하며, PDA 도 Microsoft Windows CE를 사용하기 때문에, 각 Context Observer는 C++을 이용하여 구현하였다. 나머지 모듈은 모두 Java를 이용하여 개발하였으며, JADE-LEAP을 기반으로 상호작용을 수행한다. 따라서 CO는 컨텍스트 정보를 RDF로 생성하고, Java로 구현된 모듈들은 이 문서를 읽어서 이용하도록 하였다.

실험에서 이용된 적응서비스는 자바의 JAI (Java Advanced Imaging)[17] 기술을 이용하여 구현된 이미지 변환기를 주로 이용하였으며, 이는 이미지의 크기를 변환하는 *Image_size_convert*, 컬러를 변환하는 *Image_color_depth_convert*, 포맷을 변환하는 *Image_format_convert*를 포함한다. 또한 제공 컨텐츠는 동영상과 이미지, 문서를 포함하는 멀티미디어 학습 컨텐츠를 대상으로 하였으며, XML로 기술하여, 레이아웃과 관련된 부분은 스타일 문서를 따로 분리하여 클라이언트에서 처리할 수 있도록 하였다.

4.1 응답속도 평가

첫 번째 실험은 프레임워크의 응답속도에 대한 이득을 평가하기 위한 것이며, 클라이언트 적응방식과 프록시 적응방식, 서버 적응방식을 제안프레임워크와 비교하였다. 그리고 일반적인 상황에서 응답시간 측정을 수행하고, 서버에 과부하를 발생시킨 이후에 응답시간을 각각 비교하였다.

실험에는 1600 * 1200 픽셀로 이루어져있고, 각각 900k bytes 와 1060kbytes 크기의 고용량 이미지를 포함하는 두 개의 문서를 이용하였다. 먼저 일반적인 상황에서의 응답시간을 확인하기 위해, 컨텐츠를 요청하고 클라이언트 화면에 내용 출력이 완료되는 시점을 체크하였다. 그리고 각 단계에서 응답속도에 직접적인 영향을 주는 요소를 찾기 위해, 작업을 분리하여 재시험을 수행하였다. 응답속도에 영향을 주는 요소는 각각의 위치에서 수행되는 '작업처리시간(Processing time)'과 각 시스템 간 '파일전송시간(Response time)'이며, 실험환경과 평가결과에 대한 내용이 각각 (그림 3)과 (그림 4)에 나타나있다.

컨텐츠는 미리 구축된 규칙에 따라 400 * 300 픽셀로 이미지 크기를 감소시켰고, 품질은 50%를 낮추는 적응을 수행하였다. 이에 따라 변환된 이미지는 각각 24kbytes, 32k bytes로 감소되었다.

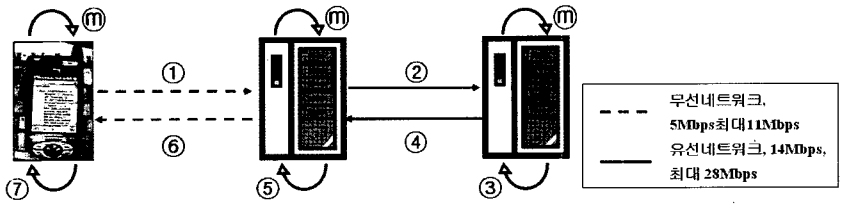
(그림 3)에서 ⓜ에 해당하는 단계는 시스템이 평소에 컨텍스트를 수집하는 단계이거나 100ms이하의 짧은 처리시간이 소요되는 단계이다.

실험결과, 클라이언트 적응 시스템은 이미지 컨텐츠를 변환하고, 이를 화면에 출력하는데 평균적으로 19초가 소요되었으며, 그 결과 전체 응답시간이 20초가 넘게 소요되었다.

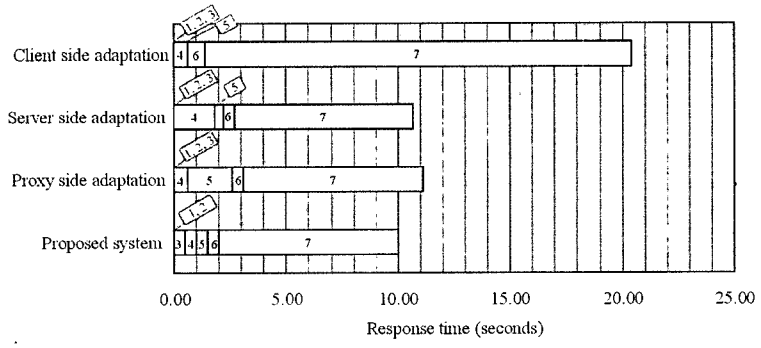
다음, 서버 적응방식은 서버에서 이미지 처리시간이 평균 1.5초 내외로 소요되었으며, 클라이언트에서 화면 로딩시간은 8초 내외로 소요되었다. 프록시 적응방식도 서버 적응과 유사한 결과가 측정되었으며, 프록시에서의 이미지 처리시간은 평균 1.8초로 체크되었다. 제안프레임워크는 이 두 문서를 병렬로 처리하도록 규칙을 생성하였으며, 서버와 프록시의 이미지 처리시간은 각각 평균 1초 내외로 소요되었다.

(그림 3)에서 네트워크 전송시간인 ④와 ⑥과정은 각각 유선과 무선 환경으로 각각 속도차이가 있었지만, 본 실험에서는 둘 다 1초 내외의 빠른 전송속도를 보였으며, 전체 응답속도에 큰 영향을 미치지 않았다.

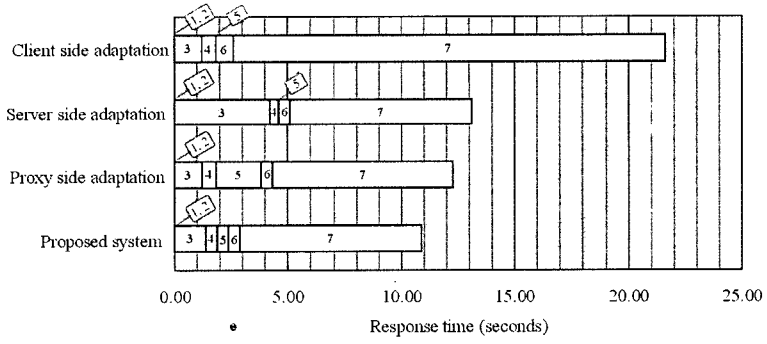
이후, 서버에 작업부하가 집중되는 상황의 변화를 실험하



(그림 3) 응답속도에 영향을 주는 요소



(그림 4) 일반적인 상황에서 응답시간 비교 그래프



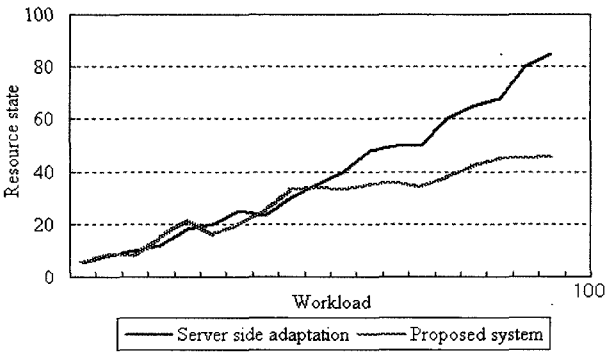
(그림 5) 서버에 과부하를 발생시킨 이후의 응답시간 비교 그래프

기 위해, 서버에서 dummy process를 생성하는 *process_generator*를 실행하였다. 이는 서버에 접속한 사용자 수와 리소스 사용량을 기록한 로그를 이용하여, 접속사용자의 증가를 시뮬레이팅 하기위해 개발된 애플리케이션이다. 실험이 진행되면서 각 방식 모두 응답시간이 증가하였지만, 제안프레임워크가 가장 적은 응답시간을 나타내었고, 서버에 작업부하가 가중되어 느려질수록 시간격차는 커졌다. 이러한 실험결과가 (그림 5)에 나와 있으며, 위와 같은 두 실험결과를 통해 응답속도 측면에서 제안프레임워크의 효율성을 확인할 수 있었다.

4.2 안정성 유지 평가

두 번째 실험은 제안프레임워크를 통해 서버의 안정적인 상태가 유지되는 특징을 입증하기 위한 것이다. 이를 위해 응답시간 측정 실험에서 사용되었던 *process_generator*를 이용하여 서버에 과부하를 발생시켰으며, 서버적응방식과 제안프레임워크의 서버 리소스 상태 변화를 지속적으로 체크하였다.

과부하를 발생시키는 실험이 진행되면서, 기존의 서버적응 방식은 지속적으로 리소스 상태가 증가하였다. 이와 반면에 제안프레임워크는 작업부하를 계속 발생시켜도, 리소스 상태가 조금씩 높아지며, 서버적응시스템에 비해 안정적인 상태를 유지함을 알 수 있었다. 본 실험결과를 통해, 제안프레임워크는 응답시간의 이득뿐만 아니라, 서버의 안정성을 유지시키는 특징을 가지고 있음을 확인할 수 있었으며, 나아가서는 시스템 다운과 같은 고장발생율을 낮출 수 있을 것임을 기대할 수 있었다. 본 실험의 결과가 (그림 6)에 그래프로 나타나있다.



(그림 6) 시스템 안정성 평가 결과

5. 결론 및 향후과제

본 논문에서는 기존에 다양한 방법으로 연구되고 있는 적응시스템들의 단점을 보완하여, 클라이언트, 프록시, 서버에 각각 적응모듈을 분산배치하고, 각 시스템의 다양한 상황에 따라 적응작업을 분산 처리하는 프레임워크를 제안하였다.

그리고 제안프레임워크의 프로토타입을 구현하여, 응답시간과 시스템 안정성 유지 측면의 이득을 테스트하였으며, 실험 결과를 통해 제안방식이 보다 빠르고 효율적인 적응작업이 가능하다는 것을 입증하였다. 이러한 제안프레임워크의 특성은 모바일 컴퓨팅 환경에서의 많은 제약사항을 개선시켜줄 것이며, E-learning과 같은 다양한 웹 컨텐츠에 적용되어, 보다 편리한 컴퓨팅 환경을 제공해줄 것으로 기대된다.

향후과제로는 유비쿼터스 컴퓨팅 환경에서 발생할 수 있는 보다 다양한 컨텍스트를 활용하는 방법과 이를 수집하고 분석, 관리하기 위한 보다 효율적인 알고리즘 연구를 수행할 것이다. 또한 패턴을 분석하여 상황 변화를 미리 예측하는 부분과 모듈 간에 전송되는 메시지의 보안 문제도 연구할 것이다.

참 고 문 헌

[1] Mark Weiser, "The Computer of 21st Century," Scientific American, 265(3), pp.94-104, Sep., 1991.

[2] Paul Horn, "Autonomic Computing : IBM's Perspective on the State of Information Technology," IBM White paper, 2001.

[3] Margaritis Margaritidis and George C. Polyzos, "Adaptation techniques for ubiquitous Internet multimedia," Wireless Communications and Mobile Computing, Vol.1, No.2, pp.141-163, Jan., 2001.

[4] Daniel Billsus, Clifford A. Brunk, Craig Evans, Brian Gladish, and Michael Pazzani, "Adaptive Interfaces for Ubiquitous Web Access," Communication of the ACM, Vol.45, No.5, pp.34-38, May., 2002.

[5] Mark Butler, Fabio Giannetti, Roger Gimson, and Tony Wiley, "Device Independence and the Web," IEEE Internet Computing, Vol.6, No.5, pp.81-86, Sep., 2002.

[6] Ariel Pashtan, Shriram Kollipara, and Michael Pearce, "Adapting Content for Wireless Web Services," IEEE Internet Computing, Vol.7, No.5, pp.79-85, Sep., 2003.

[7] Rakesh Mohan, John R. Smith, and Chung-Sheng Li, "Adapting Multimedia Internet Content for Universal Access," IEEE Trans. on Multimedia, Vol.1, No.1, Mar., 1999.

[8] IBM WebSphere Transcoding Publisher, http://www-306.ibm.com/software/pervasive/transcoding_publisher

[9] Timo Laakko and Tapio Hiltunen, "Adapting Web Content to Mobile User Agents," IEEE Internet Computing, Vol.9, No.2, pp.46-53, Mar., 2005.

[10] Wai Yip Lum and Francis C.M. Lau, "A Context-Aware Decision Engine for Content Adaptation," IEEE Pervasive Computing, Vol.1, No.3, pp.41-49, Jul., 2002.

[11] Bjorn Knutsson, Honghui Lu, Jeffrey Mogul, and Bryan Hopkins, "Architecture and Performance of Server-Directed Transcoding," ACM Trans. on Internet Technology, Vol.2, No.4, pp.392-424, Nov., 2003.

[12] Yonghyun Hwang, Jihong Kim, and Eunkyong Seo, "Structure-Aware Web Transcoding for Mobile Devices," IEEE Internet Computing, Vol.7, No.5, pp.14-21, Sep., 2003.

[13] Alvin T.S. Chan and Siu-Nam Chuang, "MobiPADS: A

Reflective Middleware for Context-Aware Mobile Computing," IEEE Trans. on Software Engineering. Vol.29, No.12 pp.1072-1085, Dec., 2003.

[14] Seunghwa Lee, Jehwan Oh, and Eunseok Lee, "An Architecture for Multi-agent based Self-adaptive System *in Mobile Environment*," LNCS 3578, pp.494-500, Jul., 2005.

[15] Maozhen Li, and Mark A. Baker, 'The Grid: Core Technologies: Chapter 6 Grid Scheduling and Resource Management,' Wiley, pp.243-300, 2005.

[16] Little, J. D., "A proof of the queueing formula L = Lambda * W", Operations Research, Vol.9, pp.383-387, 1961.

[17] Sun Microsystems - Java Advanced Imaging(JAI) API, http://java.sun.com/products/java-media/jai/

이 승 화

e-mail : shlee@ece.skku.ac.kr

2003년 2월 배재대학교 정보통신공학과 (공학사)

2005년 2월 성균관대학교 대학원 컴퓨터공학과(공학석사)

2005년 3월~현재 성균관대학교 대학원 컴퓨터공학과(박사과정)

관심분야 : 지능형에이전트, 상황적응형 시스템, 오토노믹컴퓨팅, 소프트웨어공학 등

조 재 우

e-mail : jwcho@ece.skku.ac.kr

2004년 2월 상지대학교 컴퓨터정보공학부(공학사)

2004년 9월~현재 성균관대학교 대학원 컴퓨터공학과(석사과정)

관심분야 : 유비쿼터스컴퓨팅, 오노토믹 컴퓨팅, 소프트웨어 공학, 웹컨텐츠적응 등

이 은 석

e-mail : eslee@ece.skku.ac.kr

1985년 2월 성균관대학교 전자공학과 (공학사)

1988년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학석사)

1992년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학박사)

1992년~1994년 일본 미쯔비씨 정보전자연구소 특별연구원

1994년~1995년 일본 동북(Tohoku)대학교 Assistant Prof.

1995년 3월~현재 성균관대학교 정보통신공학부 교수

관심분야 : 소프트웨어공학, 오토노믹/유비쿼터스 컴퓨팅, 에이전트지향 지능형시스템 등