모바일 웹 서비스를 위한 요소분할 캐싱 기법

심 근 정[†] · 강 의 선[††] · 김 종 근[†††] · 고 희 애[†] · 임 영 환[††††]

요 약

본 논문은 기존의 Mobile Gate System에서 모바일 웹 서비스를 위해 사용하던 Contents Cache의 성능을 향상시킴으로 보다 더 빠르게 모바일 웹 서비스를 하는 데 그 목적을 두고 있다. Contents Generator에 의해 변환된 마크-업 페이지에는 두 요소가 존재함을 알 수 있었다. 하나의 요소는 단지 요청된 DIDL 페이지와 마크-업 종류에만 의존적인 것이었고, 다른 하나의 요소는 요청된 DIDL 페이지, 마크-업 종류, 서비스를 요청한 모바일 장치의 Display 크기, 지원되는 이미지 형식, 지원되는 이미지의 색 심도 수에 각각 의존적이었다. 기존의 Contents Cache는 이렇게 두 요소를 함께 가진 마크-업 페이지 전체를 모두 저장했다. 이는 다른 요소들이 모두 같다 하더라도 하나의 요소가 달라지면 그 요소 때문에 재사용 가능한 요소들까지 Cache 메모리 영역에 중복 저장함으로써 저장 공간을 효율적으로 사용하지 못하게 하는 문제를 발생시켰다. 이 때문에 동일한 Cache 메모리 크기 하에 더 많은 변환된 마크-업 페이지를 저장할 수 있었음에도 그렇게 하지 못했다. 따라서 본 논문에서는 Contents Generator에 의해 변환된 마크-업 페이지를 두 요소로 나누고 각각을 분류하여 저장하였다. 또한 Cache 내의 데이터와 신규 데이터간의 대체 요구에 응하기 위해 LFU, LRU 두 대체 알고리즘을 적용하였다. 이를 통해 동일한 Cache 저장 공간 내에 더 많은 변환된 마크-업 페이지를 저장하게 함으로 더 빠른 속도의 Cache 성능을 구현할 수 있는 방법을 제안하였다.

키워드 : 모바일, 웹, 캐싱, 캐시, 요소, 분할

Separate Factor Caching Scheme for Mobile Web Service

Kun-Jung Sim[†] · Eui-Sun Kang[††] · Jong-Keun Kim[†††] · Hee-Ae Ko[†] · Young-Hwan Lim[††††]

ABSTRACT

The objective of this study is to provide faster mobile web service by improving performance of Contents Cache used for mobile web service in the existing Mobile Gate System. It was found that two elements existed in Mark-Up page transcoded by Contents Generator. One of the elements was dependent only on the requested DIDL page and Mark-Up type. The other was dependent on each of the requested DIDL page, Mark-Up type, size of mobile display 모바일 장치to request service, type of images available and color depth count of the images available. The conventional Contents Cache saved the entire Mark-Up page to hold both of the two elements. This caused the problem where storage space was not effectively used because reusable elements were repetitively saved in cache memory domain due to change in one of the elements even though all the other elements were the same. As a result, a larger number of transcoded Mark-Up pages could not be saved in the same cache memory size. Therefore, in this study, Mark-Up pages transcoded by Contents Generator were divided into two elements and were separately saved. Also, in order to respond to the demand for replacing data in cache with new data, this study applied two algorithms of LFU and LRU. This study proposed the method to implement cache performance of faster speed by enabling to save more number of the transcoded Mark-Up pages in the same cache storage space.

Key Words : Mobile, Web, Caching, Cache, Factor, Separate

1. 서 론

모바일 기기에서 웹 페이지를 표현하기 위해서는 많은 제약이 따른다. 너무 많은 모바일 기기들이 단일 형식으로 표준화되지 못한 채 각각 독자적이고 독립적인 형식으로 웹 페이지들을 표현하고 있다. 크게 보면 이동 통신사별로 나눠지고, 그 또한 같은 이동 통신사의 모바일 기기들일지라도 탑재된 브라우저나 브라우저 버전에 따라 다른 방식으로 웹 페이지를 표현하기도 한다.

이는 각 이동통신사들이 자사의 시스템에 적합한 마크-업 언어를 선택적으로 사용하고 있기 때문이며, 이러한 연유로 모바일에서 웹 서비스를 제공하고자 할 때 각 이동통신사와 모바일 기기에 적합한 마크-업으로 여러 가지의 웹을 만들어 제공해야 한다. 또한 모바일 기기들의 풍부하지 못한 하드웨어 구성 때문에 지원되는 리소스의 제약도 많다.

이러한 문제를 해결하기 위해 Mobile Gate System이 제

※ 본 연구는 서울시 산학연 협력사업(10581 cooperate Org 93112)지원으로 수행되었음.
† 준 회 원 : 숭실대학교 미디어학과 박사과정
†† 정 회 원 : 숭실대학교 미디어학부 전임교수
††† 정 회 원 : 숭실대학교 미디어학과 박사과정
†††† 종신회원 : 숭실대학교 미디어학부 교수 (교신저자)
논문접수 : 2007년 3월 16일, 심사완료 : 2007년 6월 1일

안되었다. Mobile Gate System은 MPEG21의 표준에 근거한 DIDL(Digital Item Description Language)이라 불리는 중간 언어(Intermediate Language)로 표현된 모바일 웹 페이지를 서비스를 요청한 모바일 기기가 처리할 수 있는 각각의 마크-업 언어로 실시간 변환하여 서비스하는 시스템이다. 그러나 Mobile Gate System의 Contents Generator도 서비스 요청이 있을 때마다 매번 마크-업 변환을 실시간으로 처리하기에는 사용되는 자원들이 너무 많이 낭비되었다.

Mobile Gate System의 낭비되는 자원을 줄이고 서비스의 응답시간을 향상시키고자 Contents Cache가 제안되었다. Contents Cache는 단순히 Mobile Gate System에서 DIDL 페이지를 해당 접속 모바일 장치에 맞는 마크-업 언어로 재구성한 마크-업 전체를 저장, 관리한다. 물론 교체 알고리즘 등은 여러 가지로 지원, 제공되지만 실제 Caching되는 정보는 변환된 마크-업 페이지 전체가 저장된다. 그러다 보니 다음과 같은 문제가 대두되었다.

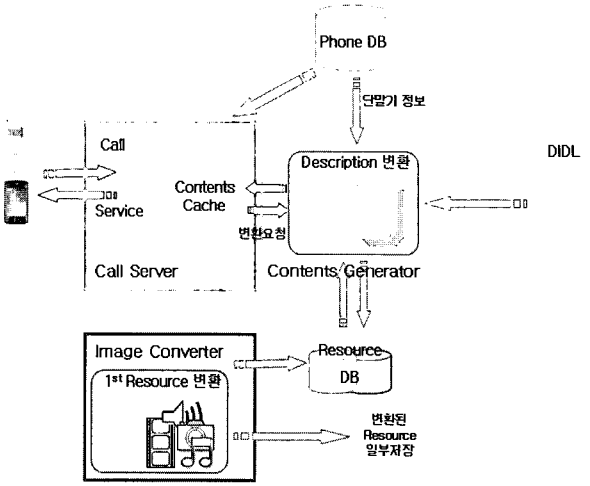
일반적으로 Cache 영역에 저장하기 위해 사용되는 데이터의 저장 공간은 서버 시스템의 주 기억 장치인 메모리 영역이다. 이는 보조 기억 장치 보다는 주 기억 장치의 접근 속도(Access Time)가 더 빠르기 때문이다. 그런데 안타깝게도 이처럼 중요한 Cache 공간의 활용을 위한 어떠한 해결책도 기존의 Mobile Gate System에 적용된 Contents Cache에서는 제공하지 못하고 있다.

본 논문은 이러한 기존의 Mobile Gate System의 Contents Cache가 가지고 있던 문제들을 해결하고 보다 적은 Cache 공간에 보다 많은 변환된 마크-업 페이지를 저장하도록 하는데 목적이 있으며, 더 나아가 Mobile Gate System의 전반적인 성능 향상을 도모하는데 목적이 있다.

2. 관련연구

2.1 Mobile Gate System

Mobile Gate System은 모바일 웹 페이지를 구성하는 각 요소들의 정보를 포함하는 DIDL 문서를, 서비스를 요청한 모바일 장치에서 처리 가능한 적합한 형태로 마크-업 변환 해 제공하는 시스템이다. Mobile Gate System의 구성은 (그림 1)과 같이 크게 3가지 모듈로 구성되어 있다. 모바일 장치가 Mobile Gate System에 특정 DIDL 페이지를 요청하면 첫 번째 모듈인 Call Server가 해당 모바일 장치의 특성을 분석, 추출하게 된다. 이렇게 추출된 특성을 바탕으로 Call Server는 모바일 장치의 추가 특성 정보를 저장하고 있는 PhoneDB에 추가 정보를 질의, 응답 받게 된다. 추출된 모바일 장치의 특성과 PhoneDB로부터 응답 받은 추가 특성을 조합하여 Contents Cache에 서비스를 요청한 모바일 장치의 특성과 요청된 DIDL 문서에 해당하는 변환된 마크-업 페이지 데이터가 존재하는지 질의하게 된다. 이 때 만일 해당 마크-업 페이지 데이터가 존재하면 추가 마크-업 변환 없이 Contents Cache에서 응답 받은 데이터를 서비스를 요청한 모바일 장치에 제공하게 된다. 그러나 이 때 Contents Cache에 서비스를 요청한 모바일 장치의 특성에 부합하는 마크-업 페이지가 존재하지 않으면 두 번째 구성 모듈인 Contents Generator에 요청된 DIDL 문서를 요청한 모바일 장치의 특성에 적합하도록 마크-업 변환을 요청하게 된다. Contents Generator는 요청된 DIDL 문서를 <표 1>과 같이 서비스를 요청하는 모바일 장치의 특성에 적합하도록 마크-업 변환을 수행한다. 이 때 DIDL 문서를 구성하는 특정 리소스의 변환 요구가 발생하면 세 번째 구성 모듈인 Image Converter에 해당 리소스의 변환을 요구한다. 변환된 리소스 정보를 Contents Generator는 변환된 마크-업 페이지에 적용하여 하나의 완전한 모바일 웹 페이지로 재구성해 서비스를 요청한 모바일 장치에 서비스를 제공하게 된다.



(그림 1) Mobile Gate System 구성도

<표 1>서비스를 요청하는 모바일 장치의 특성

· Mark Up 형식 · Resource 형식 · Image 색 심도 · Display 크기 · Memory 크기 · Network 환경

2.1.1 Contents Cache

하나의 Mobile Web Site에 접속한 모바일 장치는 한동안 그 Mobile Web Site에 서비스를 요청할 확률이 높기 때문에 동일한 모바일 장치의 특성에 대해 DIDL 문서를 계속적으로 재구성해야 하는 Overhead가 발생된다. 이러한 Overhead는 결국 응답 시간의 지연으로 이어져 요청에 응답하는데 필연적으로 Waiting-Time이 발생하게 된다. Contents Cache는 이러한 동일한 모바일 장치의 특성에 대해 동일한 DIDL 문서를 재구성하는 Overhead 및 응답 시간에 대해 Waiting-Time을 최소화하기 위해 제안되었다. 특히 제한된 Cache 메모리 크기를 효과적으로 사용하기 위해 <표 3>과 같은 여러 알고리즘이 포함되었다.

〈표 2〉 Mobile Web Site 구성의제약 요소

· 제한된 Display 크기 · 불편한 UI(User Interface) · 유선에 비해 느린 무선 Network 환경

〈표 3〉 Contents Cache에 적용된대체 알고리즘

· IFO(First In First Out) · RU(Least Recently Used) · FU(Least Frequency Used)

2.2 Contents Cache의 문제점

현재 Mobile Gate System에 적용된 Contents Cache의 Caching 기법은 단순히 Mobile Gate System에서 DIDL 문서를 서비스를 요청한 모바일 장치의 특성에 맞는 마크-업 페이지로 재구성한 마크-업 페이지 전체를 Caching한다. 대체 알고리즘(Replacement Algorithm)이 지원되지만 실제 Caching되는 정보는 마크-업 페이지 전체가 저장된다.

이는 서비스를 요청한 모바일 장치의 특성 중 하나라도 다르다면 그 이유 때문에 변환된 전체 마크-업 페이지를 동일하게 Cache에 저장하는 문제를 야기시킨다. 이 문제는 결국 제한된 크기의 Cache 메모리 영역에 동일한 정보를 저장함으로 인해 더 많은 정보의 저장을 제약하는 문제로 볼 수 있다. 결국 중복되는 내용 때문에 새로운 정보의 저장이 막히고, 이것은 Mobile Gate System 성능에 영향을 끼치게 된다.

3. 모바일 웹 페이지의 구성 Factor

모바일 웹 페이지는 PC 웹 페이지와 마찬가지로 순수한 텍스트로 작성된 태그들로 구성되어 있다. 각 마크-업 언어에 따라 구문이 달라지며 그 의미 또한 다르다. 마크-업 언어에 따라 구문과 의미가 다르지만 동일한 것은 그 안에 정보가 담겨 있으며, 그 정보는 크게 두 가지 요소로 나누어 볼 수 있다는 것이다.

같은 마크-업 언어를 사용하는 모바일 장치라 할지라도, 단지 마크-업 언어에만 의존적인 요소가 있는 반면에, 마크-업 언어에도 의존적이면서 또한 각 모바일 장치의 특성에 의존적인 요소가 있다. 단지 마크-업 언어에만 의존적인 요소를 FLDD(Factor of Loosely-Defended on Device)라 명하고, 마크-업 언어에 의존적이면서도 동시에 각 모바일 기기의 특성에 의존적인 요소를 FSDD(Factor of Strongly-Defended on Device)라 명한다.

3.1 Factor of Loosely-Defended on Device (FLDD)

FLDD는 동일한 마크-업을 지원하는 모바일 장치 군 내에서 각 모바일 장치의 특성에 보다 덜 영향을 받는 요소들을 의미한다. 이것을 풀어 설명하면 같은 마크-업 언어를 사용하는 모바일 장치에는 해당 요소가 특정한 가공 없이 모두 동일하게 적용될 수 있다는 것이다. 예를 들면, WML 마크-업 언어의 구문 중에서 <TABLE></TABLE> 태그를 보면 WML을 지원하는 모든 모바일 장치는 이 태그를 TABLE로 동일하게 인식하며, 모바일 브라우저 상에 동일하게 표현한다. 이러한 요소는 다른 마크-업 언어를 지원하는 모바일 장치에 사용되기 전에는 항상 동일하게 사용될 수 있다.

이런 특성을 가진 요소들은 초기에 각 마크-업에 따라 변환된 모바일 웹 페이지를 2차 가공 없이 같은 마크-업 언어를 지원하는 모든 모바일 장치에 사용할 수 있는 것이다.

3.2 Factor of Strongly-Defended on Device (FSDD)

FSDD는 동일한 마크-업을 지원하는 모바일 장치군 내에서도 각 모바일 장치의 특성에 따라 보다 더 영향을 받는 요소들을 의미한다. FSDD는 같은 마크-업 언어로 표기되어 있다 하더라도 각 모바일 장치의 특성 즉, Display 크기나 Memory 크기, 혹은 지원되는 Resource의 형식 등에 따라 영향을 받는 요소들이다. FSDD들은 모바일 웹 서비스를 요청한 모바일 장치에 따라 다르게 제공되어야 하며, 이 요소들 때문에 보다 더 많은 모바일 웹 페이지들이 제작되어야 한다. 예를 들어, WML 마크-업 언어의 구문 중에서 이미지를 표현해 주는 <IMG></IMG> 태그 중에서 이미지의 Resource 경로를 표기하는 SRC 속성을 들 수 있다. 이 SRC 속성에는 실제로 표현될 이미지의 경로 정보를 담고 있지만 이 이미지는 모바일 장치에 따라 각기 다르게 제공되어야 한다.

이런 특성을 가진 요소들은 초기에 각 마크-업에 따라 변환된 모바일 웹 페이지를 모바일 장치들 각각의 특성에 맞도록 2차 가공해야 해당 모바일 장치에 서비스할 수 있다. 그러나 이러한 FSDD 요소가 한 모바일 웹 페이지에서 차지하는 비율이 높지 않다.

3.3 모바일 장치와 각 Factor들 사이의 의존성

모바일 장치에서 모바일 웹 페이지를 보기 위해서는 <표 1>과 같은 몇 가지 제약 요소들이 있다.

마크-업 형식은 각 모바일 장치의 브라우저에서 지원하는 마크-업의 종류를 말한다. 말할 것도 없이 마크-업의 종류는 모바일 장치에 의존적이다.

Display 크기는 각 모바일 장치가 지원하는 Display의 가로, 세로 크기를 말한다. 기존 PC 웹 브라우저들은 Scroll Bar를 통해 Display 크기를 넘어서는 Resource들을 볼 수 있도록 지원했다. 그러나 모바일 장치에 탑재된 브라우저들은 아직 Display 크기를 넘어서는 Resource들을 지원할 수 있는 방법을 제공하지 못하고 있다. 그러므로 Display 크기 역시 모바일 장치에 의존적이다.

Resource 형식은 각 모바일 장치의 브라우저에서 표현할 수 있는 Resource의 형식을 말한다. 아직까지도 많은 모바일 웹 브라우저들이 동영상이나 플래시(Flash)와 같은 보다 더 동적인 정보를 표현하지 못하고 있다. 그러나 또 한편으로 특정 브라우저들에서는 플래시 정도의 동적인 표현을 가능하게 하고 있다. 또한 이미지를 예를 들어 보면 어떤 모바일 장치에서는 JPEG 형식은 지원하면서도 GIF 형식은 지원하지 못하는 경우가 있다. 아니 오히려 여러 이미지 형식 중에서 특정 형식만 선택적으로 지원하는 경우가 대다수이다. 그렇기 때문에 Resource 형식 또한 모바일 장치에 의

존적임을 알 수 있다.

이미지의 색 심도는 각 이미지가 하나의 픽셀(pixel) 정보를 표현하기 위해 사용하는 비트(bit)의 수를 말한다. 보통은 1비트, 2비트, 4비트, 16비트, 24비트, 32비트 등으로 하나의 픽셀을 표현한다. 그러나 이처럼 다양한 색 심도를 모바일 웹 브라우저들은 모두 처리해 주지 않는다. 대신에 특정 색 심도만 선택적으로 지원한다. 그렇기 때문에 이미지의 색 심도 역시 모바일 장치에 많이 의존적일 수밖에 없다.

3.2.1 FLDD와 모바일 장치간의 의존성

FLDD는 모바일 웹 페이지를 표현하는 각 마크-업 언어의 태그들 중 모바일 장치에 덜 의존적인 요소들을 말한다. 결국 특정 마크-업 언어로 구성된 모든 모바일 웹 페이지들은 모바일 장치에 의존적일 수밖에 없지만 그 외의 요소들에 보다 덜 의존적인 태그들을 살펴보면 다음과 같다.

실제로 마크-업을 제외한 모바일 장치에 의존적인 요소들의 특징을 살펴보면 대체로 Resource와 관련된 것이다. Display 크기도 Resource의 표현 범위를 한정 짓기 때문이고 메모리 크기도 물리적, 논리적 Resource의 표현 범위를 한정 짓기 위한 것이다. 또한 Resource의 형식이나 이미지의 색 심도 역시 Resource에 해당한다.

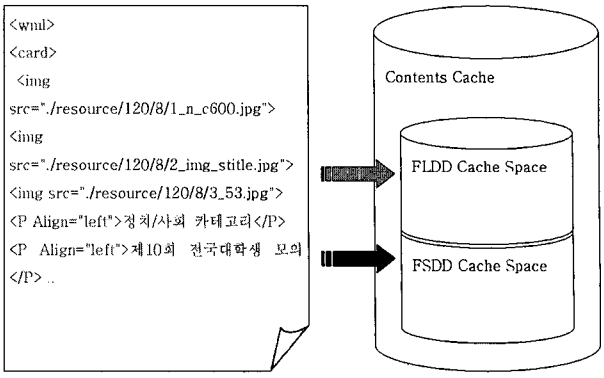
이러한 사실들에 비추어 볼 때 Resource와 관련된 태그들을 제외 하고는 거의 모든 태그들이 단지 마크-업 언어에만 의존적이다. 그러므로 Resource 관련된 태그 외에는 FLDD라 할 수 있다.

3.2.2 FSDD와 모바일 장치간의 의존성

FSDD는 모바일 웹 페이지를 표현하는 각 마크-업 언어의 태그들 중 모바일 장치에 보다 더 의존적인 요소들을 말한다. 앞서 FLDD를 설명하면서 Resource와 관련 없는 태그들은 모두 FLDD라고 하였다. 그렇다면 FSDD는 FLDD를 제외한 나머지 태그들이 될 것이다. 이 태그들은 위에서 언급한 각 모바일 장치의 의존적 요소들에 따라 각각 달라질 수 있는 것이다. 예를 들어 이미지를 표현하는 마크-업 태그인 <IMG> 태그의 요소 중에서 Resource의 경로를 표현하는 SRC 속성의 데이터만인 FSDD가 된다. 왜냐하면 <IMG> 태그와 <IMG> 태그 내의 속성들 중 SRC를 제외한 다른 태그들은 모두 마크-업 언어에만 의존적이고 모바일 장치에 의존적인 다른 요소들에는 의존적이지 않기 때문이다. 즉, <IMG> 태그의 속성 중 SRC 속성만 해당 모바일 장치에 맞도록 바꾸어준다면 다른 문제없이 해당 모바일 장치의 브라우저에서 표현될 수 있다.

4. Factor 분할 Caching 기법

Factor 분할 Caching 기법은 (그림 2)와 같이 하나의 모바일 웹 페이지 내에 공존하는 두 요소, FLDD와 FSDD를 분리하여 Cache에 저장함으로써 Cache 내에 동일 저장 공간에 저장되는 모바일 웹 페이지의 개수를 늘리고자 하는



(그림 2) Factor 분할 Caching 기법

것이다.

다시 말해 Factor 분할 Caching 기법은, Contents Cache에 변환된 특정 마크-업 페이지를 저장 하고자 하는 요청이 있을 경우, 이 모바일 웹 페이지의 구성 태그 요소들을 FLDD와 FSDD로 분류하고, 분류된 정보에 따라 각각의 문자열을 FLDD는 FLDD Cache Space에, FSDD는 FSDD Cache Space에 분류하여 저장한다. 이 때 FLDD 문자열 사이에 FSDD의 위치 정보를 Tagging하여 저장하게 되며, FSDD를 저장할 때에는 어떤 FLDD와 연관되어 있는 FSDD인지, 그리고 어떤 모바일 기기의 특성에 따른 FSDD인지를 명기하여 저장하게 된다.

실제로 Contents Cache에 저장된 모바일 웹 페이지를 서비스하기 위한 요청이 있을 때 Contents Cache는 해당 페이지가 저장된 FLDD Cache Space에서 저장된 FLDD 정보를 추출하고, 해당 페이지와 해당 모바일 기기의 특성에 부합하는 저장된 FSDD를 FSDD Cache Space에서 추출하여 통합한 후 서비스하게 된다.

이렇게 Contents Cache를 모바일 웹 페이지의 특정 Factor들로 분류하여 저장함으로 동일 크기의 Contents Cache에 저장할 수 있는 모바일 웹 페이지의 수를 증가시킬 수 있다.

4.1 변환된 모바일 웹 페이지의 Factor 분할

Mobile Gate System의 Call Server는 특정 모바일 장치가 특정 모바일 웹 페이지의 서비스를 요청 할 경우 해당 모바일 웹 페이지가 Contents Cache 내에 존재하는지 확인한다. 확인 후 해당 모바일 웹 페이지가 Contents Cache에 존재하지 않을 경우 Contents Generator에 신규 변환을 요청한다. 이 때 Contents Generator는 서비스를 요청한 모바일 장치의

〈알고리즘 1〉 요소분할 Caching 알고리즘

입력: - 서비스를 요청한 모바일 장치의 특성
- 서비스 되어야 할 DIDL 문서

출력: - 서비스를 요청한 모바일 장치에 적합한 변환된 마크-업

단계: (1) 요청된 DIDL 문서가 재구성될 때 FLDD와 FSDD 분할
(2) 분할된 FLDD와 FSDD를 해당Caching Space에 저장
(3) 재요청시 Contents Cache에 Query하여 모바일 장치와 DIDL 문서에 해당하는 FLDD, FSDD 추출
(4) FSDD들을 FLDD의 각 영역에 병합하여 반환

특성에 따라 Description과 Resource를 변환하고 종말에는 두 결과물을 하나의 변환된 마크-업 Page로 결합하여 서비스하게 된다. 모바일 웹 페이지의 Factor 분할은 Description 변환과 Resource 변환 시에 이루어지게 된다. Description 변환이 진행되는 중간 중간마다 Resource 변환이 필요할 때 변환된 경로를 즉시 Description에 병합하는 것이 아니라 병합 직전에 따로 분류하여 번호를 발급하고 그 번호에 따라 Caching하는 것이다.

4.2 Contents Cache Space 분할

Contents Cache의 저장 공간은 (그림 2)와 같이 분할된 두 Factor들을 저장할 수 있도록 수정되어야 한다. 기존 Contents Cache는 마크-업 정보 전체를 단일 저장 공간에 저장하였기 때문에 본 논문에서 제안하는 Factor 기반 Caching 알고리즘을 적용하기 위해서는 저장 공간의 분할이 반드시 선행되어야 하는 것이다.

Contents Cache의 저장 공간을 다음과 같이 두 부분으로 분할한다. 하나는 FLDD를 위한 Caching Space로, 다른 하나는 FSDD를 위한 Caching Space로 분할한다.

FLDD Caching Space는 <표 1>에서와 같이 서비스를 요청한 모바일 장치의 특성 중 마크-업 형식과 요청된 DIDL 문서 명을 조합하여 키(Key)로 하고 순수한 FLDD 마크-업 요소들을 데이터로 하는 자료구조를 사용한다.

FSDD Caching Space는 <표 1>에서와 같이 서비스를 요청한 모바일 장치의 모든 특성과 요청된 DIDL 문서 명, 그리고 서비스 요청된 DIDL 문서의 해당 위치 데이터를 조합해 키로 하고 순수한 FSDD 마크-업 요소들을 데이터로 하는 자료구조를 사용한다.

4.3 Factor 분할 Caching 알고리즘

Factor 분할 Caching은 기본적으로 4.1장에서 설명한 것과 같이 요청된 DIDL 문서를 요청한 모바일 장치의 특성에 적합하도록 변화할 때 FLDD와 FSDD로 분할한다. 그런 후 분할된 각 Factor들을 각각의 Caching Space에 분할하여 저장, 관리하게 된다. 즉, 4.2장에서 설명한 바와 같이 FLDD는 FLDD Caching Space에, FSDD는 FSDD Caching Space에 분할하여 저장 관리하게 된다. 후에 동일한 특성을 가진 모바일 장치가 동일한 DIDL 문서를 요청하게 되면 Call Server는 Contents Cache에 Query하여 분할된 Caching Space에서 각 요소들을 추출하고 병합하여 온전한 마크-업 데이터를 서비스하게 된다.

4.4 대체 알고리즘

Cache에 주어진 메모리 크기는 한정되어 있기 때문에 모든 마크-업 페이지를 다 저장할 수 없다. 그렇기 때문에 Cache 내에 이미 저장된 마크-업 페이지와 새롭게 저장하려고 하는 신규 마크-업 페이지간의 대체(Replacement)가 필요하다. 이러한 과정을 처리 해주는 알고리즘을 추가하였다. 기존의 Contents Cache에 적용되었던 두 대체 알고리즘을 본 논문에서 제안하는 요소분할 Caching 기법에 적용하였다.

4.4.1 LFU(Least Frequency Used)

LFU 알고리즘은 각 데이터 별로 참조된 횟수를 특정 Key로 기록하여 보관해 두었다가 대체 요구가 발생했을 경우 가장 적게 참조된 데이터를 신규 데이터와 대체하는 알고리즘이다. 요소분할 Caching 기법에서는 FSDD 요소가 FLDD 요소에 의존적이기 때문에 FLDD 요소는 제외하고 FSDD 요소의 참조 횟수를 기록하여 우선적으로 FSDD 요소를 제거함으로 신규 데이터의 저장 공간을 마련한다. 즉, 대체 요구가 발생했을 경우 FSDD 요소 중 참조 횟수가 적은 순으로 데이터를 제거한다. 그런 후에 제거된 FSDD 요소와 연관된 FLDD 요소가 다른 FSDD 요소들에 의해 사용되는지 확인한 후 사용되면 그대로 두는 반면, 사용되지 않을 경우는 FLDD 요소도 함께 제거한다. 그런 후 Cache에서 사용할 수 있는 크기를 측정하고 신규 데이터를 저장할 수 있을 만큼의 여유 공간이 생길 때까지 위의 작업을 반복한다.

4.4.2 LRU(Least Recently Used)

LRU 알고리즘은 각 데이터들이 최근에 사용되었는지 여부를 List-Up한다. 대체 요구가 발생하면 가장 오래 전에 참조된 데이터를 신규 데이터와 대체하는 알고리즘이다. 요소분할 Caching 기법에서는 LFU 알고리즘과 마찬가지로 우선적으로 FSDD 요소에 대해서 LRU 알고리즘을 적용한다. 그런 후 제거된 FSDD 요소가 의존하고 있던 FLDD 요소가 다른 FSDD 요소들에 의해 사용되고 있는지 확인한 후 사용되고 있으면 그대로 두는 반면, 사용되고 있지 않으면 함께 제거한다. 그런 후 Cache에서 사용할 수 있는 크기를 측정하고 신규 데이터를 저장할 수 있을 만큼의 여유 공간이 생길 때까지 위의 작업을 반복한다.

5. 실험 결과

5.1 실험환경

하드웨어 환경은 펜티엄4 3.0GHz CPU와 1GB 램으로 구성된 PC를 사용했다. 운영체제로는 Microsoft Windows XP Professional을 사용했다. 실험을 위한 DIDL 페이지 파일은 총 30개로 이미지와 텍스트로만 구성된 DIDL 페이지를 사용했다. PhoneDB는 HDML 정보 119개, mHTML 정보 283개, WML 정보 629개, XHTML 정보 1개로 구성했다. 그리고 Cache 대체 알고리즘은 기존의 LFU(Least Frequency Used)와 LRU(Least Recently Used) 두 가지 경우 모두 적용했다. 메모리 크기는 1Kb, 10Kb, 20Kb, 30Kb, 40Kb, 50Kb 총 6가지 경우로 실험했다. 요청 회수는 100건, 500건, 1000건, 5000건 총 4가지 경우로 실험했다.

5.2 성능평가

성능평가는 전체적으로 총 3번에 이루어 졌다. 3번의 결과에서 가장 높은 결과 치와 가장 낮은 결과 치를 제외한

결과 치를 가지고 성능을 평가했다.

5.2.1 Cache 적중률

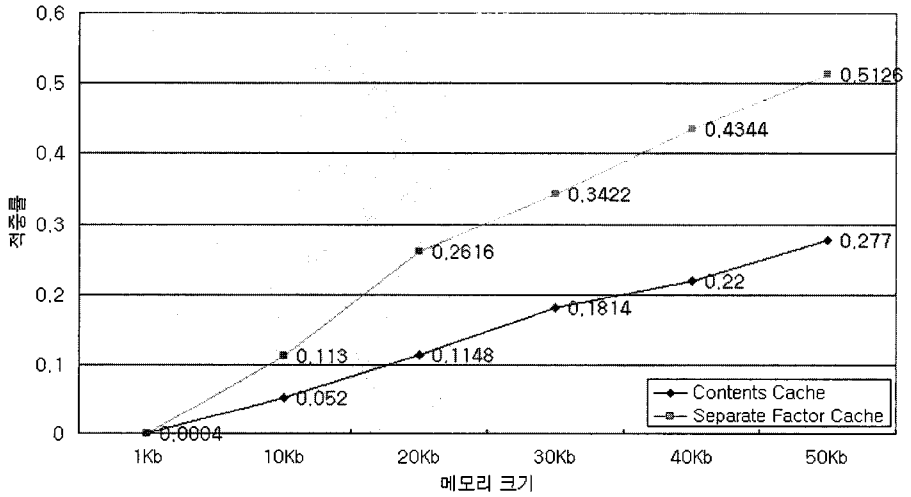
Cache 적중률은 서비스 요청 시 Contents Generator에 의해 재구성되지 않고 Cache 내에 저장된 마크-업으로 바로 서비스한 회수를 전체 서비스 요청 회수로 나눈 것을 말한다. 이번 실험에서는 대체 알고리즘 LFU와 LRU 각각에 대해 5000건의 요청 회수를 기준으로 1Kb, 10Kb, 20Kb, 30Kb, 40Kb, 50Kb 각각 용량 별로 Cache 적중률을 구해 비교했다. 또한 50Kb 메모리를 기준으로 100건, 500건, 1000건, 5000건 각각의 요청회수 별로 Cache 적중률을 구해 비교했다.

동일한 환경 하에서 Cache 적중률이 높다는 것은 그만큼 Contents Generator가 마크-업 페이지를 재구성해야 할 횟수가 줄어든다는 것을 의미한다. 결국 응답 시간을 향상시킴으로 Mobile Gate System의 성능을 향상시킴을 의미한다.

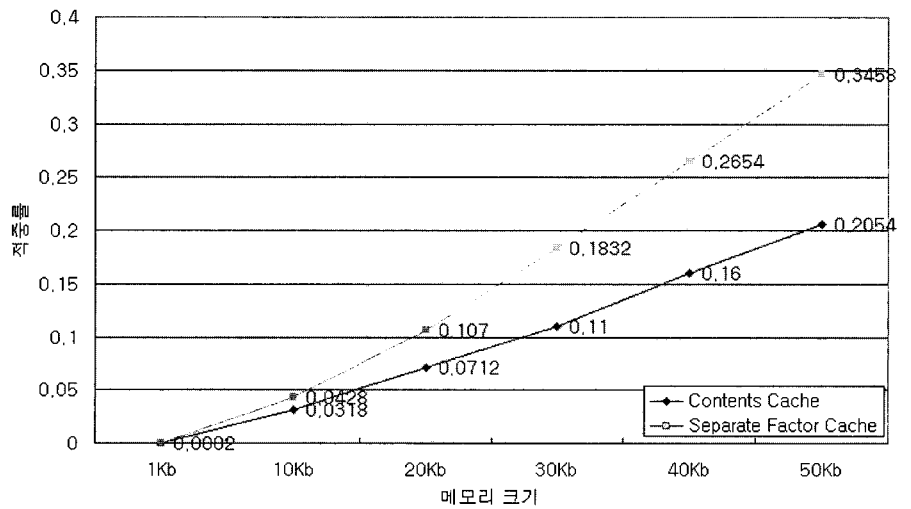
(그림 3)의 그래프에서 보는 바와 같이 LFU 대체 알고리즘을 적용했을 때 각 메모리 크기에 따라 Cache 내의 마크-업으로 요청에 대처한 비율이 기존의 Contents Cache보다는 Separate Factor Cache가 더 높다는 것을 알 수 있다.

(그림 4)의 그래프에서 보는 바와 같이 LRU 대체 알고리즘도 비슷한 양상을 보이기는 하지만 LFU 대체 알고리즘보다는 그 상승세가 덜함을 알 수 있다. 이것은 LRU 대체 알고리즘보다는 LFU 대체 알고리즘이 본 Cache에서는 더 적합함을 보여주는 것이다. 페이지 요청 회수를 고정하고 각 Cache의 메모리 크기에 변화를 주어 실험한 결과 대체적으로 기존의 Contents Cache보다는 Separate Factor Cache가 더 좋은 적중률을 보여줬다.

같은 적중률을 평가한 것이지만 위의 (그림 5)의 그래프는 Cache의 메모리 크기를 고정하고 요청 수를 증가시켜 평가한 결과 데이터이다. 요청수가 100건일 때와 요청수가 500건일 때 가장 큰 변화를 보여줬지만 지속적으로 상승되고 있음을 볼 수 있다. 이것은 두 Cache 다 비슷한 양상이지만 그 상승세는 Contents Cache보다는 Separate Factor Cache가 더 가파름을 볼 수 있다. 결국 요청수가 늘어날수록 적



(그림 3) LFU가 적용된 요청 수 5000건 기준의 메모리 크기 별 적중률



(그림 4) LRU가 적용된 요청 수 5000건 기준의 메모리 크기 별 적중률

중률은 Contents Cache보다는 Separate Factor Cache가 더 높아진다는 것이다.

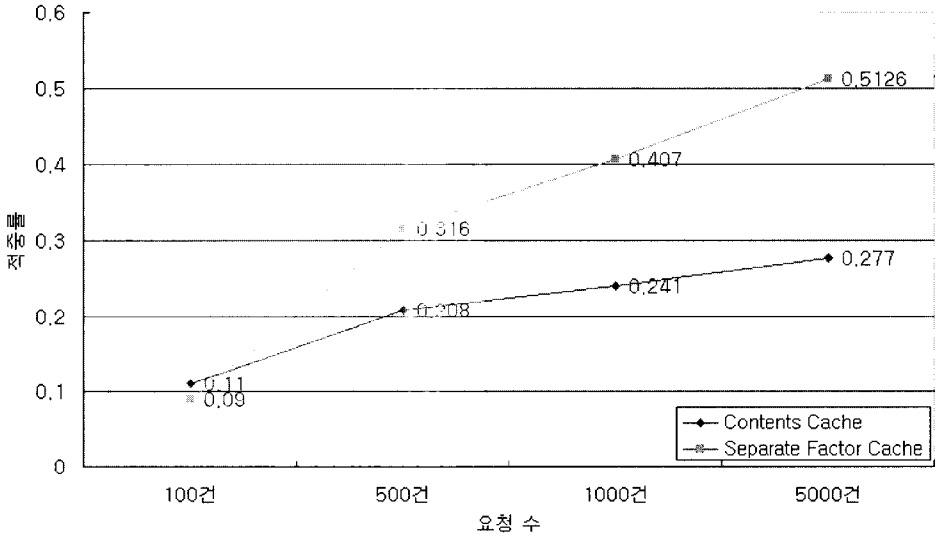
(그림 6)의 그래프에서는 대체 알고리즘을 LFU에서 LRU로 바꾸어 평가한 결과 데이터이다. LRU 대체 알고리즘을 사용했을 때는 또 다른 양상을 보였다. 물론 요청 수 100건에서 500건 사이의 상승세가 가장 가파른 것은 같았지만 그 후의 상승세는 LFU 대체 알고리즘과는 달리 완만해짐을 볼 수 있다. 이것은 요청 수를 고정하고 메모리 크기를 변화시켜 평가했던 이전의 그림들과 마찬가지로 LFU 대체 알고리즘이 LRU 대체 알고리즘보다 Separate Factor Cache에 더 적합함을 보여주는 것이다.

5.2.2 동일 메모리 크기 내에 저장된 마크-업 페이지 수

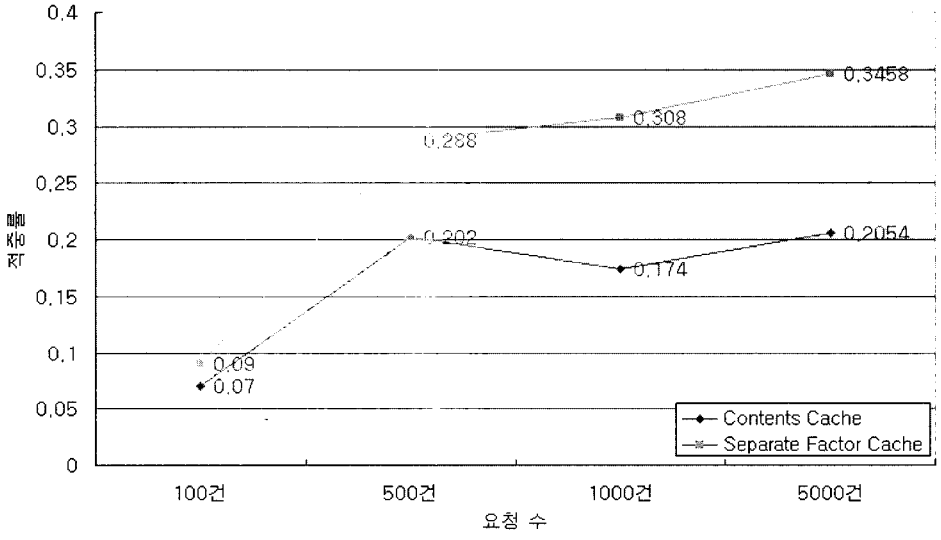
동일 메모리 크기 내에 저장된 마크-업 페이지 수는 기존의 Contents Cache와 Separate Factor Cache가 둘 다 동일한 메모리 크기 내에서 Cache 영역에 저장하고 있는 마크-업 페이지의 개수를 말한다. 이 때 Separate Factor Cache는 FLDD 저장 공간에 저장된 FLDD 페이지 요소와 FSDD 저장 공간에 저장된 FSDD 페이지 요소를 결합하여 하나의 페이지를 구성하기 때문에 구성되어 서비스될 수 있는 페이지의 개수로 저장된 마크-업 페이지의 개수를 결정한다.

동일 메모리 크기에 저장된 마크-업 페이지 수가 높을수록 Cache 적중률을 높이게 된다. Cache 적중률이 높아지면 5.2.1장에서 언급한 바와 같이 Contents Generator의 재구성 횟수를 줄일 수 있고, 이것은 결국 Mobile Gate System의 성능 향상을 가져올 수 있다.

(그림 7)의 그래프는 LFU 대체 알고리즘을 적용하고 요청 수를 고정해 각 Cache의 메모리 크기에 변화를 주어 평



(그림 5) LFU가 적용된 메모리 50Kb 기준의 요청 수 별 적중률



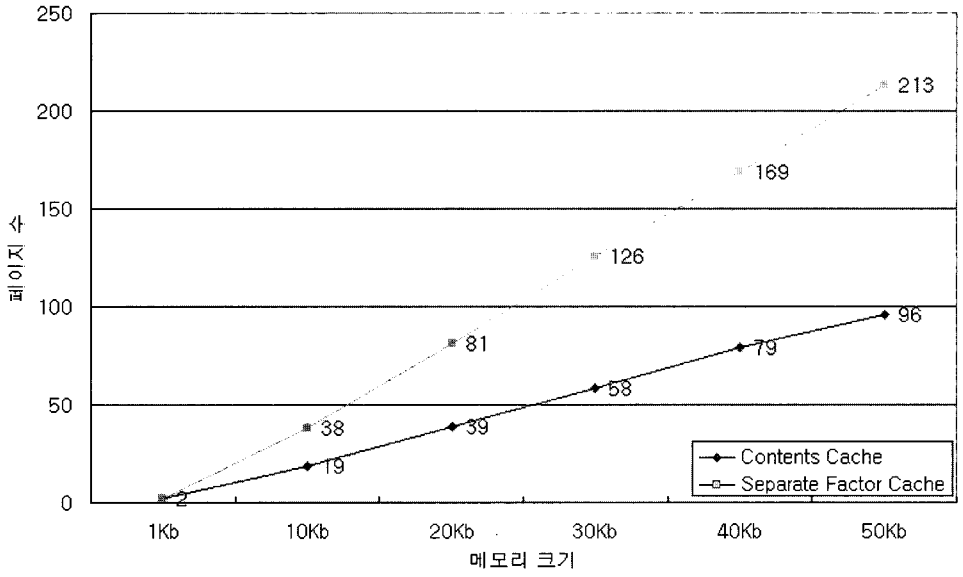
(그림 6) LRU가 적용된 메모리 50Kb 기준의 요청 수 별 적중률

가한 결과 데이터이다. 그림에서 보는 바와 같이 1Kb일 때는 비슷한 개수로 시작하지만 50Kb가 되었을 때는 저장되는 페이지의 개수가 현저히 달라짐을 볼 수 있다. Separate Factor Cache가 기존의 Contents Cache에 비해 약 2배 가량 더 많은 변환된 마크-업 페이지를 저장하고 있음을 확인할 수 있다. 더 많은 변환된 마크-업 페이지를 저장하고 있다는 것은 결국 적중률의 향상을 가져오고 Mobile Gate System의 성능 향상을 가져온다.

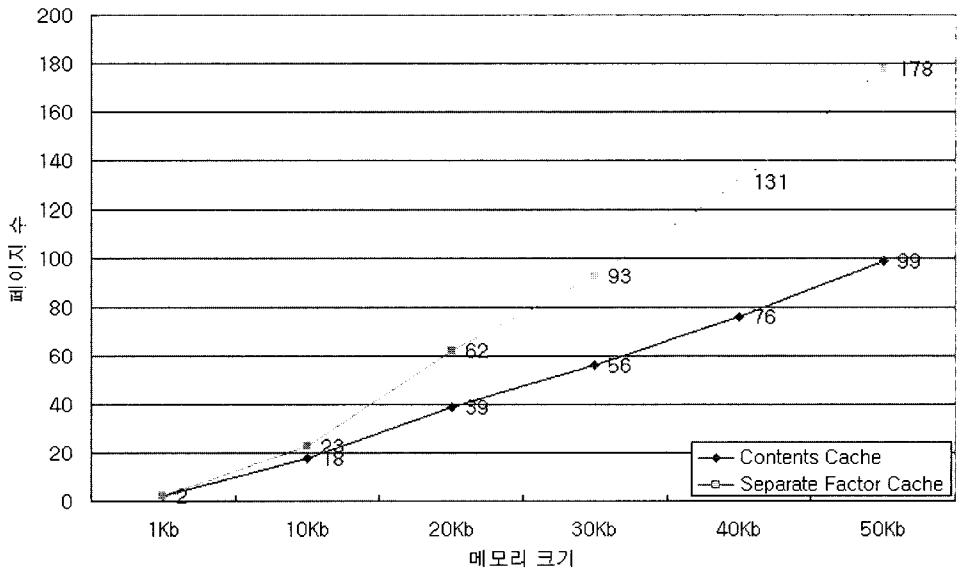
(그림 8)은 (그림 7)에서 대체 알고리즘을 LRU로 바꾼 결과이다. 결론적으로는 (그림 7)의 그래프와 비슷한 모양을 하고 있지만 실제 저장되는 마크-업 페이지의 수는 더 적다. 이것은 앞에서 살펴본 적중률과 밀접한 관계가 있는데, 적중률의 그것과 같이 페이지 저장 개수도 LRU 대체 알고리즘 보다는 LFU 대체 알고리즘이 더 좋은 성능을 보였다.

5.2.3 응답시간

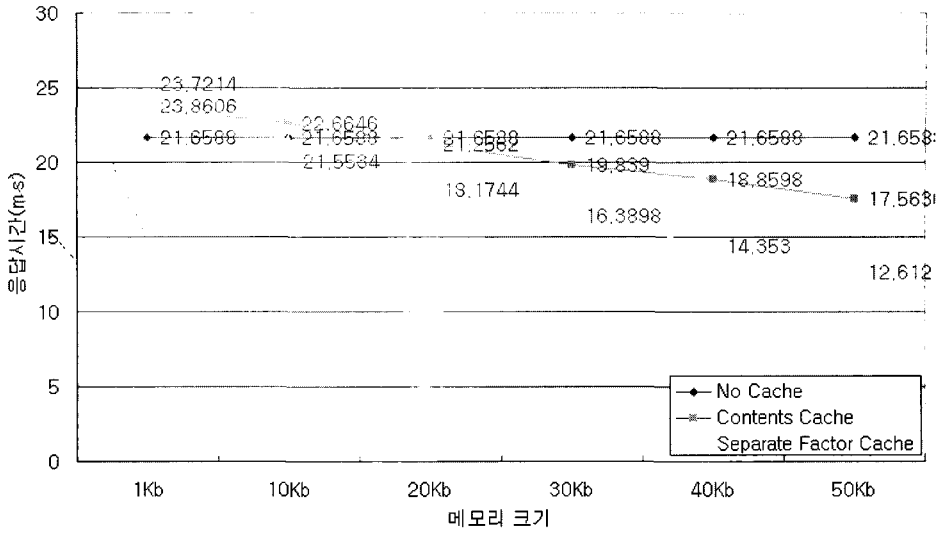
응답시간은 크게 보면 서비스 요청이 시작된 시점부터 서비스 요청이 끝나는 시점까지를 말한다. 그러나 본지에서는 Cache의 성능만을 평가하기 때문에 (그림 3)에서 보는 바와 같이 서비스를 요청한 시점으로부터 Contents Generator에게 Cache의 정보를 가져오거나 DIDL 페이지를 마크-업 페이지로 재구성하기 위해 요청하기 직전까지의 준비 과정을 평가 시간에 넣지 않았다. 또한 실제로 Cache 내에 저장되어 있던 마크-업 페이지나 재구성된 마크-업 페이지가 서비스를 요청한 요청 자에게 전달되는 시간도 포함하지 않았다. 즉, 순전히 Contents Generator 모듈이 Cache 내에 원하는



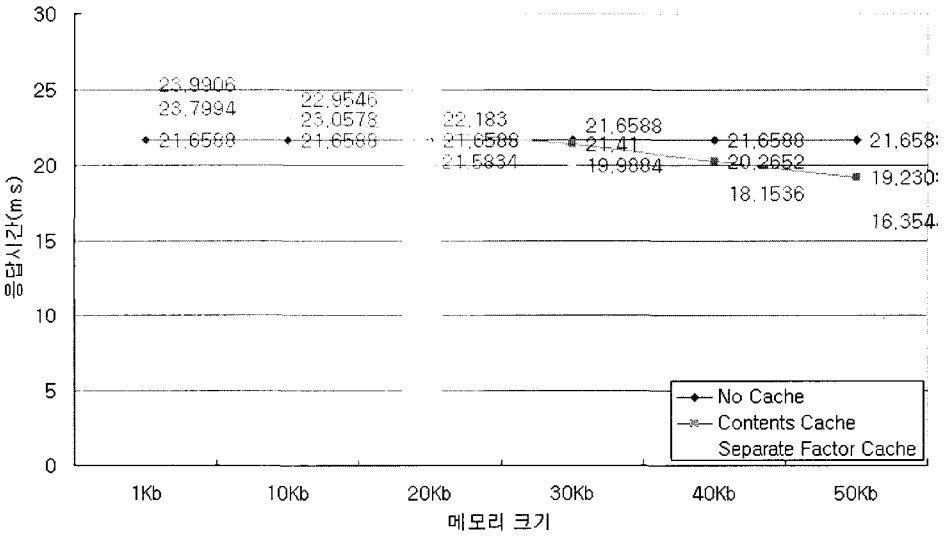
(그림 7) LFU가 적용된 요청 수 5000건 기준의 메모리 크기 별 페이지 수



(그림 8) LRU가 적용된 요청 수 5000건 기준의 메모리 크기 별 페이지 수



(그림 9) LFU가 적용된 요청 수 5000건 기준의 메모리 크기 별 응답시간



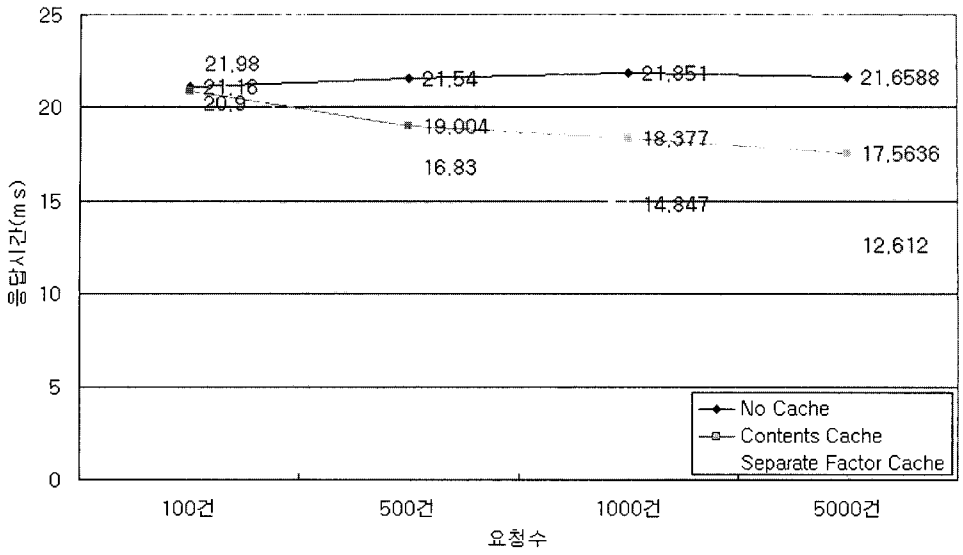
(그림 10) LRU가 적용된 요청 수 5000건 기준의 메모리 크기 별 응답시간

마크-업 페이지를 질의 하거나 신규 마크-업 페이지를 재구성하는 시간만을 응답시간을 평가하는 기준으로 삼았다.

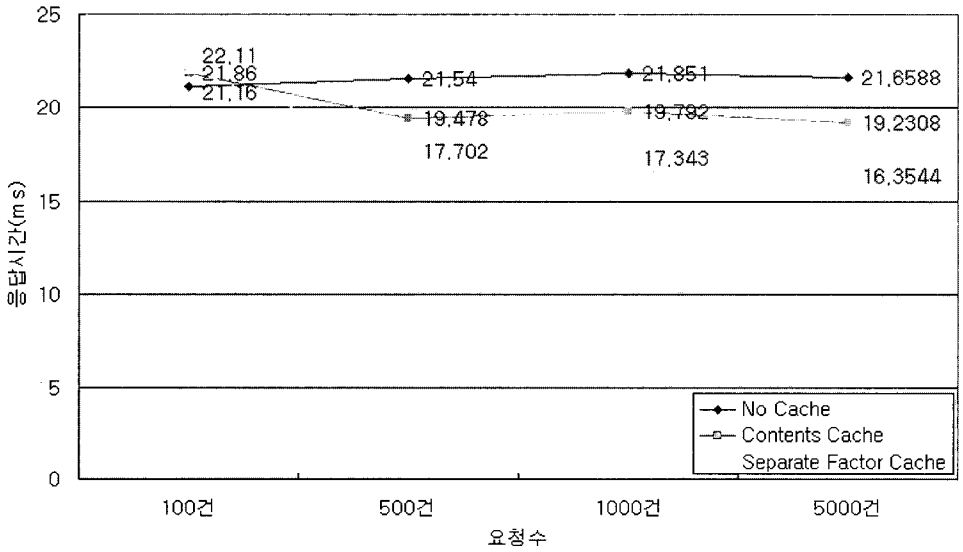
응답시간은 Cache를 사용하지 않은 경우, 기존의 Contents Cache를 사용한 경우, 본지에서 제안하는 Separate Factor Cache를 사용한 경우 이렇게 총 3가지의 경우를 가지고 평가했다. 이번 실험에서는 Replacement Algorithm LFU와 LRU 각각에 대해 5000건의 요청 회수를 기준으로 1Kb, 10Kb, 20Kb, 30Kb, 40Kb, 50Kb 각각 용량 별로 응답시간을 구해 비교했다. 또한 50Kb 메모리를 기준으로 100건, 500건, 1000건, 5000건 각각의 요청회수 별로 응답시간을 구해 비교했다.

(그림 9)는 LFU 대체 알고리즘을 사용하고 5000건의 요청 수를 기준으로 각 Cache의 메모리 크기를 증가시키면서 평가한 결과 그래프이다. Cache를 사용하지 않았을 때는 메모리 크기를 결정할 수 없으므로 모두 동일한 결과를 가져왔다. 또한 초기의 1Kb일 때는 앞에서 본 것과 같이 적중률이 0에서 0.0004이기 때문에 거의 적중하지 못했으므로 대부분의 페이지를 Cache 내의 데이터에서 사용하지 못하고 새롭게 재구성했음을 알 수 있다. 이 때 Cache는 재구성하는 시간보다 오래 걸렸는데 그 이유는 먼저 Cache에 해당 데이터의 존재 유무를 질의하는 시간과 재구성된 마크-업 페이지를 Cache에 저장하는 시간이 추가로 포함되었기 때문이다. 그래서 1Kb일 때는 재구성하는 시간보다 오히려 더 많은 시간을 사용했음을 알 수 있다. 그러나 Cache의 메모리 크기가 커짐에 따라 Cache 내에 저장되는 페이지 수가 늘어나고 자연스레 적중률이 함께 높아지면서는 그 양상이 달라진다.

(그림 10)의 그래프에서 확인할 수 있듯이 LRU 대체 알고리즘 역시 Cache의 메모리 크기가 커짐에 따라 두 Cache 또한 응답시간이 줄어듦을 알 수 있다. 그러나 그 중에서도 Separate Factor Cache의 응답시간이 기존의 Contents



(그림 11) LFU가 적용된 메모리 50Kb 기준의 요청 수 별 응답시간



(그림 12) LRU가 적용된 메모리 50Kb 기준의 요청 수 별 응답시간

Cache의 응답시간보다 더 줄어듦을 볼 수 있다. 이것 역시 Cache 내의 페이지 저장 개수와 적중률과 밀접한 연관이 있는 것이다.

(그림 11)의 그래프는 앞에서 본 그래프들과 비슷하지만 Cache의 메모리 크기를 50Kb로 정하고 요청 수를 증가시키면서 응답시간을 평가한 결과이다. 이 역시 앞에서 보았던 적중률 결과에서 예상할 수 있듯이 요청 수 100건과 500건 사이의 응답시간이 가장 가파르게 줄어들고 있음을 볼 수 있다. 또한 Cache를 사용하지 않았을 때에는 평균적으로 비슷한 응답시간을 보여줬지만 Cache를 사용한 두 경우는 모두 요청수가 늘어남에 따라 응답시간이 점차 줄어들었다. 그러나 그래프에서 보는 바와 같이 기존의 Contents Cache보다는 본 논문에서 제안하는 Separate Factor Cache의 경우 더 낮은 응답시간을 보여준다.

(그림 12)의 그래프는 (그림 11)의 그래프와 비슷하지만 대체 알고리즘을 LFU에서 LRU로 바꾼 것이다. 역시 앞의 적중률에서 보았던 것과 같이 LRU 대체 알고리즘을 적용했을 때보다는 LFU 알고리즘을 적용했을 때 응답시간이 더 줄어듦을 볼 수 있다.

6. 결 론

본 논문에서는 Mobile Gate System에서 사용되던 기존의 Contents Cache의 성능을 향상시키는 기법을 제안하고 이를 구현, 평가하였다.

모든 DIDL 페이지를 모든 모바일 장치의 성능에 적합하도록 항상 재구성 하는 것은 Mobile Gate Server의 불필요한 자원 사용과 Mobile Gate System의 성능 저하를 초래하

고 종말에는 모바일 웹 페이지를 요청한 사용자들의 신뢰성을 저하시키게 된다. 또한 동시 접속자 수가 늘어나면 늘어날수록 이를 해결하고자 하드웨어의 추가 구입 등 불필요한 지출도 발생된다. 이러한 상황은 Cache 기법을 통해 어느 정도 해결할 수 있다. 그러나 이러한 Cache도 주어진 저장 공간의 한계 때문에 그 성능에는 제약이 따를 수밖에 없다.

본 논문에서 제시하는 Factor 분할 Cache 기법은 이러한 주어진 제한된 Cache 저장 공간 내에서 보다 많은 정보를 저장하게 하여 더 빠르게 요구에 응답할 수 있게 한다. Factor 분할 Cache 기법은 변환 마크-업 페이지를 Cache에 저장함에 있어 보다 덜 모바일 장치에 의존적인 요소와 보다 더 모바일 장치에 의존적인 요소를 분류, 따로 저장함으로 동일한 메모리 크기에 기존의 Cache보다 더 많은 마크-업 페이지를 저장하게 한다. 이는 결국 적중률의 향상과 응답시간의 단축으로 Cache와 Mobile Gate System 전반의 성능 향상을 가져온다. 또한 Cache 기법에서 떼어 놓을 수 없는 대체 알고리즘을 비교 평가함으로 보다 더 나은 대체 알고리즘을 선택할 수 있게 하였다.

실험 결과를 통해 기존의 Contents Cache보다 Factor 분할 Cache 방법이 동일 메모리 하에 Cache에 저장되는 페이지 개수가 더 많았고, 적중률의 향상과 결과적으로 응답시간을 향상시킬 수 있었다.

참 고 문 헌

[1] Elizabeth J. O'Neil, and Gerhard Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, p.297-306, May 1993.

[2] Theodore Johnson and Dennis Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," In Proceedings of the 20th VLDB Conference, p.439-450, 1994.

[3] 김현섭, "웹 Caching을 위한 교체 알고리즘," 동국대학교 석사 학위 논문, 2004

[4] J.T.Robinson and N. V. Devarakonda, "Data cache Management Using Frequency-Based Replacement," In Proceedings of the 1990 ACM SIGMETRICS Conference, p134-142, 1990.

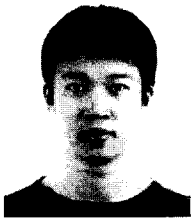
[5] Kai Cheng and Yahilo Kambayashi, "Advanced Replacement Policies for WWW Caching" In Proc. 1st International Conference on Web Age Information management(WAIM'2000), June 2000.

[6] Pei Cao, Sandy Irani, "Cost-Aware WWW Proxy Caching Algorithms" Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Dec 1997.

[7] Ludmila Cherkasova and Gianfranco Ciardo, "Role of Aging, Frequency, and Size in Web Cache Replacement Policies" In Proceedings of the Sixth International Symposium on Computers and Communications (ISCC'01), Hammamet, Tunisia, July 3-5, 2001.

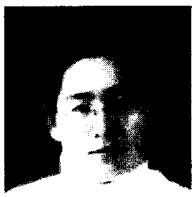
[8] Kai Cheng, Yahiko Kambayashi, "Enhanced Proxy Caching with content Management" Knowledge and Information Systems, April 2002.

[9] HaiYang Hu, JiDong Ge, Ping Lu, XianPing Tao, and Jian Lu, "Supporting Wireless Web Page Access in Mobile Environments Using Mobile Agents*" State Key Laboratoryfor Novel Software Technology, Nanjing University, 210093, China.



심 근 정

e-mail : inyourhands@nate.com
2004년 고려대학교 국어국문학과(학사)
2007년 숭실대학교 컴퓨터학과(석사)
2007년~현 재 숭실대학교 미디어학과 박사과정
관심분야 : 멀티미디어, 유비쿼터스, 모바일 컴퓨팅



강 의 선

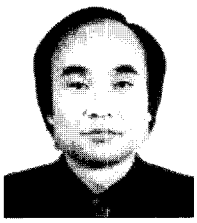
e-mail : iami86@ssu.ac.kr
2002년 숭실대학교 컴퓨터공학(석사)
2007년 숭실대학교 미디어학과(박사)
2007년~현 재 숭실대학교 미디어학부 전임교수
관심분야 : 멀티미디어 기술, 무선 인터넷, DMB

김 종 근

e-mail : jongni@ssu.ac.kr
2000년 숭실대학교 전자공학과(학사)
2007년 숭실대학교 미디어학과(석사)
2007년~현재 숭실대학교 미디어학과 박사과정
관심분야 : 모바일 시스템, 멀티미디어

고 희 애

e-mail : heeae9484@korea.com
2002년 숭실대학교 정보과학대학원 (미디어공학석사)
1999년~2006년 Ogilvy&Mather Korea 근무 Digital marketing팀
2007년~현 재 숭실대학교 미디어학과 박사과정
관심분야 : 디지털컨텐츠, 모바일컨텐츠, 유저인터페이스, 디지털마케팅, 멀티미디어, 전자상거래



임 영 환

e-mail : yhlim@ssu.ac.kr
1977년 경북대학교 수학과(학사)
1979년 한국과학원 전산학과(석사)
1985년 Northwestern University 전산학과(박사)
1979년~1996년 한국전자통신연구소 책임연구원
1996년~현 재 숭실대학교 미디어학부 교수
관심분야 : 멀티미디어 기술, 유비쿼터스, 모바일, 감성 컨텐츠 분야