클라우드 컴퓨팅에서 사용자 작업환경의 끊김 없는 연계를 위한 웹 서비스 연결 관리 기법

최 민*

Web Service Connection Management Scheme for Seamless Migration of User Workspace in Cloud Computing

Min Choi*

■ Abstract ■

Cloud computing emerges as a new computing paradigm which targets reliable and customizable services. The term builds on decades of research in virtual machine, distributed and parallel computing, utility computing, and more recently networking, web service, and software as a service. In this paper, we provide a seamless connection migration of web services. This is useful for cloud computing environment in which many client terminals have mobility. With the wireless internet facility, those mobile users can move place to place during internet communication. Therefore, we provide solutions to the two major problems in current virtualization based migration : communication failure problems and connection re-establishment. Communication channel flushing by zero window notification helps to resolve the communication failure problems and TCP port inheritance prevents connection re-establishment errors during socket reconstruction. Thus, our web service migration facility is now able to preserve open network connections, and even for server sockets. This is a highly transparent approach, in that we did not introduce additional messages for channel flushing and did not make any modification to the TCP protocol stack. Experimental results show that the overhead due to connection migration of web services is almost negligible when compared with time to take the conventional web service migration.

Keyword : Software as a Service(SaaS), Web Service, Cloud Computing, Service Migration

논문투고일 : 2009년 01월 17일 논문수정완료일 : 2009년 03월 10일 논문게재확정일 : 2009년 03월 12일

* 삼성전자 반도체총괄 메모리사업부 책임연구원

1. 서 론

우리나라는 수천만명의 인터넷 사용자수를 바탕으로 빠른 IT성장을 이끌고 있다. 게다가, 최근 국내 인터넷 환경은 이미 포화상태에 이른 유선 인터넷을 넘어, 시간과 장소의 제약 없이 사용 가능한 무선 인터넷 환경으로 진화하고 있다. 전세계 모바일 와이맥스(Mobile WiMAX) 802.16e(와이브로, Wibro) 서비스는 2010~2013년경에 이르면 사용자수가 8천만명을 초과할 것으로 예상된다[10]. 이와 같은 첨단 네트워크 인프라를 기반으로 오늘날 많은 인터넷 사용자는 수시로 사용하거나 혹은 언제 어디서 사용할지 알 수 없는 자료에 대해 인터넷 상의 블로그나 홈페이지에 이를 저장해 두기 시작했다. 장소에 구분없이 인터넷 접속이 가능해진 오늘날과 같은 환경에서는 필요시 언제든 자신의 블로그에서 자료를 다운로드하여 사용할 수 있다. 이러한 개념은 최근 각광받고 있는 차세대 컴퓨팅 환경이라 불리우는 클라우드 컴퓨팅[9, 14]의 기반이 되었다.

클라우드 컴퓨팅(Cloud Computing)은 웹 2.0, 서비스로서의 소프트웨어(Software as a Service : SaaS), 유틸리티 컴퓨팅(Utility Computing)[13]의 개념이 복합적으로 결합된 개념이다. 복수의 데이터센터를 가상화 기술로 통합하여 개인 사용자에게 각종 소프트웨어와 보안 솔루션, 컴퓨팅 능력까지 필요에 따라 온디맨드 방식으로 제공한다. 과거에 썬 마이크로시스템(Sun Microsystems)사에서 주창하던 씬 클라이언트(Thin Client)를 개선 및 확장한 개념으로 볼 수도 있다. 당시에는 네트워크와 컴퓨팅 성능에 대한 우려가 많아 성공하지 못했지만, 오늘날과 같은 고성능의 컴퓨팅 환경과 초고속 인터넷 환경에서는 그 실현가능성에 아무런 제약이 없다. 이와 같은 클라우드 컴퓨팅은 인터넷만 연결되어 있으면, 클라이언트 단말이 고성능 기기가 아니라도 원격으로 원하는 작업을 얼마든지 수행할 수 있다. 즉, 언제 어디서나 자신이 사용하는 자료를, 인터넷만 접속되어 있다면 어떤 단말기로도 원하는 문서 작업이 가능하다.

클라우드 컴퓨팅은 최신 웹 서비스[2, 12, 15] 기술을 적용하여 구글(Google)이나 IBM과 같은 세계적인 기업들이 차세대 컴퓨팅 패러다임으로 선정하며 관련기술 개발을 선도하고 있다[6], [7]. 실제로, 구글 캘린더[5]는 PC나 PDA에 매번 동기화하는 불편함을 없애고 자신의 스케쥴을 다른 사람과 데이터를 공유할 수 있는 기능을 앞세워 시장을 점유하고 있다. 페이스북(facebook)은 문서작성기를 비롯한 6,000가지 애플리케이션 웹 인터페이스를 통해 제공한다. 특히, 페이스북에서 제공하는 다양한 웹 서비스 애플리케이션은 본 연구에서 주목하고 있는 차세대 클라우드 컴퓨팅을 위한 대표적인 응용이다.

페이스북의 경우와 같은 온 디맨드(On Demand) SaaS 컴퓨팅 패러다임을 궁극적으로 실현하기 위해서는 클라이언트의 위치나 접속형태에 관계없이 지속적이고도 끊김없는(Seamless) 서비스를 제공하는 것이 필요하다. 왜냐하면, 사용자는 익숙한 데스크톱 컴퓨팅 환경을 클라우드 컴퓨팅 환경에서도 그대로 제공받기를 원하기 때문이다. 이를 위해서 클라이언트 웹 애플리케이션은 서버측 웹 서비스와의 통신 과정에서 두 가지 사항에 유의해야 한다.

첫째는 클라이언트측 웹 애플리케이션의 이동으로 인해 발생하는 통신 단절(Communication Failure) 현상이다. 둘째는 이동된 클라이언트 애플리케이션을 새로운 클라이언트 단말에서 재시작(Restart)할 때 발생하는 연결 재설정 오류(Connection Re-establishment Error)이다. 본 연구는 이와 같은 웹 서비스 애플리케이션의 이동시 발생할 수 있는 문제를 효율적으로 해결하는 방법을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서 본 연구의 동기와 관련연구에 대해 소개한다. 제 3장에서는 웹 서비스 연결 관리 기법(Web Service Connection Management Scheme : WSCMS)을 제안하고 제 4장에서 웹 서비스 연결 관리 기법의 실험 결과를 제시한다. 그리고, 제 5장에서 결론을 맺는다.

2. 배경지식 및 연구동기

차세대 웹 애플리케이션은 웹 서비스의 집합인 서비스 기반 아키텍처(Service Oriented Architecture : SOA)에 기반한다. 이와 같은 서비스 기반 아키텍처(SOA)의 바탕위에 클라우드 컴퓨팅의 개념을 실현하고자 하는 것이 오늘날 차세대 컴퓨팅 환경을 이야기하는 많은 연구자들의 최종적인 목표이다. 최근 급속도로 발전하는 IT환경에서는 가정에서 고성능 PC를 위해 투자하는 것을 꺼려한다. 대신에 필요한 만큼의 자원을 매우 저렴한 가격으로 활용할 수 있는 클라우드 컴퓨팅의 개념이 유용하며, 기업에서는 웹 서비스 제공을 위해 자체적으로 대규모의 웹 서버를 구축/유지/운용 하거나 호스팅 서비스 업체나 IDC 업체에 값 비싼 비용을 지불하지 않아도 된다. 실제로 아마존에서 제공하는 Elastic Computing Platform(Amazon EC2)[1], [15] 등은 서버 호스팅 비용 측면에서 기존 방식에 비해 저렴하다[8]. 클라우드 호스팅 (Cloud Hosting) [11]이라는 용어를 만들어 낼 정도로 최근 클라우드 컴퓨팅을 웹 서버 호스팅으로의 사용하는 것이 각광받는 추세이다. 클라우드 호스팅은 서비스 사용량에 따라 비용을 산정하는 방식으며, 온 디맨드로 서비스 용량 증설을 원할 때 언제든지 가능하다. 게다가 서버 시스템 및 인프라에 신경 쓸 필요가 없는 등 향후 웹 서비스 호스팅 업계에 큰 변화가 있을 것으로 전망한다.

클라우드 컴퓨팅의 위와 같은 다양한 응용 범위 외에도, 본 연구에서는 차세대 개인용 컴퓨팅 플랫폼을 대체하는 씬 클라이언트 환경을 주목한다. 클라이언트측 사용자는 연산능력(Computing Power)이 부족한 PDA나 스마트폰 등에서도 엑셀이나 파워포인트와 같은 방대한 애플리케이션을 수행할 수 있는 장점이 있다. 현재는 페이스북에서 제공하는 웹 애플리케이션 서비스가 가장 대표적이다. 이 서비스는 내부적으로 서버측 웹 서비스와 클라이언트측 라이브러리가 서로 결합되어 통신을 하며 동작한다. 클라우드 컴퓨팅 환경에서는 다수의 서버 노드가 수 많은 웹 서비스를 제공한다. 따라서, 특정 웹 서비스가 동작중이라 할지라도, 이를 잠시 중단했다 다시 실행하거나, 부하분산 기능 등의 목적으로 다른 컴퓨팅 노드로 이동하여 재실행 하는 것이 필요하다. 이러한 경우, 연결중인 네트워크 상태를 유지할 수 있는가에 대한 문제가 제기된다.

클라우드 컴퓨팅 사용자가 회사에서 페이스북에서 제공하는 온라인 워드프로세서를 사용하다 급히 퇴근하거나 장소를 이동하는 경우, 다른 곳에서 해당 작업을 계속해서 수행하고자 할 경우. 자리를 이동하기 전에 작업중인 상태를 저장하지 않았기 때문에 다른 곳에서 가장 최근의 작업 상태를 이어 받아 작업할 수 없는 상황이 발생한다. 이러한 문제와 관련된 연구로는 TCP 연결 이동시 네트워크 상태를 유지하기 위한 것으로 모바일 IP와 TCP-R 등이 있다.

모바일 IP(Mobile IP)[3]는 IP 터널링 프로토콜을 활용하는 네트워킹과 라우팅 메커니즘으로, 이동성있는 모바일 노드가 네트워크 연결을 계속적으로 유지할 수 있도록 한다. 즉, 서로 다른 네트워크에 걸쳐 이동하는 모바일 노드라 할지라도 동일한 IP 주소를 가지며 끊김없는 네트워크 연결을 제공하는 기법이다. 모바일 IP 네트워크는 홈 에이전트(HA)와 외부 에이전트(FA)가 존재하는데, 이들 에이전트들은 서로 협력하면서 패킷을 모바일 노드에게 포워딩한다. 이렇게, 모바일 호스트가 어떤 한 IP 서브 네트넷으로부터 다른 서브넷으로 이동할 때 IP 주소를 변경하지 않고 가능하도록 한다. 모바일 IP는 이동중에 수신된 송신측 패킷을 홈 에이전트가 가로채어 보관하다 모바일 노드가 새로운 네트워크 서브넷에서 등록 완료될 때 까지 기다린다. 하지만, 이 방법은 클라우드 컴퓨팅의 웹 서비스 애플리케이션의 연계처럼 통신 단절 구간이 다소 길어지는 경우에는 채널 혼잡상황으로 인식하여 윈도우크기를 급격히 줄여 성능이 저하되는 단점이 있다.

TCP-R[4] 은 핸드오프 시간동안 종단간(end-to-

end) 어드레스를 변경해주는 방식으로 현재 액티브하게 사용되는 TCP 연결을 유지하는 방식으로 TCP 핸드오프(Hand Off)를 수행한다. 이를 위해, TCP 유한상태머신(Finite State Machine : FSM)에 세 가지 추가적인 상태(RD_WAIT, RD_SENT, RD_RCVD)를 도입하여 네트워크 통신 단절구간을 표현할 수 있도록 하였다. 또한 다섯가지 옵션을 도입하여 IP 변환 기능을 수행할 수 있는 메시지를 전달하도록 하였다. 그러나, 이 방법은 TCP 프로토콜 상에 변화를 가하는 방식이므로 시스템 상에서 OS나 디바이스 드라이버 수준의 변화를 주어야하는 제약사항이 있다.

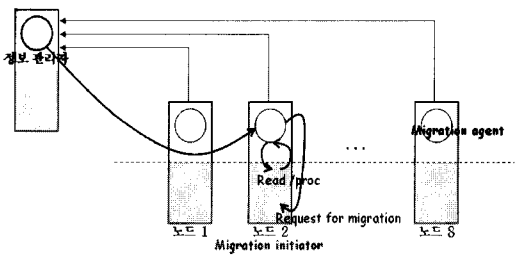
모바일 IP와 TCP-R은 공통적으로 네트워크를 사용중인 상태에서 모바일 시스템 자체가 이동하는 상황을 가정한다. 따라서, 소프트웨어 실행상태를 그대로 유지하면서, 변화하는 네트워크 주소를 반영해주는 테크닉이라 할 수 있다.

하지만, 본 논문에서는 IDC 센터내에서 부하분산 및 시스템 활용률(utilization) 최적화를 위해 어떤 웹 서비스가 한 서버 노드로부터 다른 서버 노드로 이동하는 상황을 가정한다. 이 때에는, 서버 자체가 움직이는 것이 아니라, 실행중인 웹 서비스 애플리케이션 프로세스가 현재 실행중인 컨텍스트를 이미지로 생성한다. 이러한 컨텍스트 이미지를 다른 서버 노드에서 재생성하는 방식으로 시스템 부하 및 활용률을 조정한다. 이를 위해서는 컨텍스트 이미지를 생성하고, 다른 노드로 전송하며 메모리에 컨텍스트를 재생성하기까지 접속 연계(Connection Migration) 시간이 필요하다. 결국, 웹 서비스가 클라이언트들과의 통신 단절 (Communication Failure) 구간이 발생하는데, 본 논문에서는 이러한 통신 단절 전 후의 네트워크 상태를 일관되게(Consistent) 유지하는 방법을 제안한다.

그리고, 연결 재설정 오류 문제는 웹 서비스 애플리케이션 자체가 다른 호스로 이동하여 재생성(Reconstruction) 될 때 발생한다. 즉, 모바일 IP와 TCP-R은 웹 서비스 프로세스가 이동한 후 소켓 컨텍스트를 다시 생성하는 상황을 고려하지 않기 때문에, 서버 측 연결이 이동하여 재생성될 때 발생하는 연결 재설정 오류(Connection Re-establishment Error) 문제를 해결할 수 없다. 하지만, 본 연구는 연결 상속(Connection Inheritance) 기법을 이용하여 이를 해결한다.

3. 웹 서비스 연결 관리 기법

웹 서비스 서버 시스템의 원활한 운용을 위한 웹서비스 실행 관리 기법은 다음과 같다. [그림 1]과 같이 각 서버 노드별 에이전트들 사이의 정보 교환 및 웹 서비스 실행을 제어하는 한 대의 정보 관리자(Information Manager)와 다른 여러 클라이언트 단말(Client Terminal)가 존재한다.



[그림 1] 웹 서비스 연결관리를 위한 구성요소

각 클라이언트 단말에는 현재 노드에서 작업환경 연계 요청이 들어올 때, 이를 서버측 정보 관리자와 통신하여 서비스 연계 작업을 초기화(Initiation)하는 역할을 수행한다. 일반적인 서비스 작업환경 연계를 위한 구현은 가상화 기술에 의거하여 중앙의 서버가 없이도 가능하지만, 본 연구에서는 두 개 클라이언트 에이전트 외에 중앙의 정보관리자 서버를 경유한다. 이러한 구조는 본 연구가 기반하고 있는 온디맨드 서비스 환경이 서버를 중앙에 두어 응용프로그램 저장소(Web Service Repository)와 작업공간 저장소(Workspace Repository)역할을 수행하기 때문이다. 웹 서비스 연계 과정은 서비스 중단(Suspend)과 재실행(Restart)의 두가지로 분리되어야 할 필요가 있으며, 연계하고자 하는 서버에서

에이전트가 서비스 중단 요청을 내리면(Initiate) 옮겨질 대상 노드에서 웹 서비스가 실행중이던 시점의 실행상태로 복귀하여 재실행한다.

웹서비스 작업환경 마이그레이션이 발생할 때 현재 실행중인 웹 서비스에서 사용중인 데이터는 저장해 두었다가 다시 복구되어야 한다. 이러한 정보들에는 주소 공간, 레지스터 집합, 열린 파일/파이프/소켓, SystemV IPC 구조체들, 현재 작업 디렉토리, 시그널 핸들러, 타이머, 터미널 설정값, 사용자 식별 정보(UID, GID 등), 서비스 식별 정보(SID, SGRP, SID 등), RLIMIT 등 그 외에도 여러가지 데이터들이 있다. 몇 가지 정보들은 응용 서비스에 따라 선택되어질 수 있는 가능성이 있는데 비해, 주소 공간과 레지스터 집합들은 서비스 실행에 직접적인 연관이 있는 것으로 반드시 저장되고 복구해야 하는 구성 요소이다. 본 연구에서 제안하는 서비스 연계 과정을 체크포인트와 재실행 부분으로 분리하여 설계한 것이다.

3.1 웹 서비스 연계 과정

웹 서비스에서 네트워크 연결을 가지는 서비스 연계 기능은 통신망을 통해 패킷을 교환하는 서버측 웹 서비스와 클라이언트측 라이브러리가 다른 단말로 매끄럽게 연계할 수 있도록 하는 것이다. 이를 위해서는 통신중인 클라이언트 서비스를 연계할 때 서비스의 실행 상태뿐만 아니라 전송 중인 통신 채널의 상태까지 저장 및 복원하는 기능이 필요하다. 이러한 메커니즘을 포함하는 서비스 연계 수행 순서는 다음과 같다.

① 연계할 서비스 혹은 서비스 그룹을 요청한다. 이 요청은 홈(Home) 혹은 대상(Target) 노드에 의해 요청될 수도 있고, 사용자에 의해 명시적으로 혹은 시스템에 의해 자동적으로 이루어질 수도 있다.

② 연계 명령이 홈 노드의 연계 지원 모듈에게 보내진다.

③ 홈 노드의 연계 지원 모듈은 해당 웹 서비스를 중단시킨다.

④ 홈 노드의 연계 지원 모듈은 웹 서비스 상태를 파일 혹은 네트워크에 저장한다. 그리고 본래의 서비스는 제거한다.

⑤ 홈 노드의 연계 지원 모듈은 해당 서비스의 통신 채널과 파일 상태 정보를 저장한다.

⑥ 서비스가 옮겨갈 대상 노드에 새로운 서비스를 생성한다.

⑦ 대상 노드의 연계 지원 모듈은 중단된 서비스 상태 정보를 홈 노드 연계 지원 모듈로부터 가져 온다.

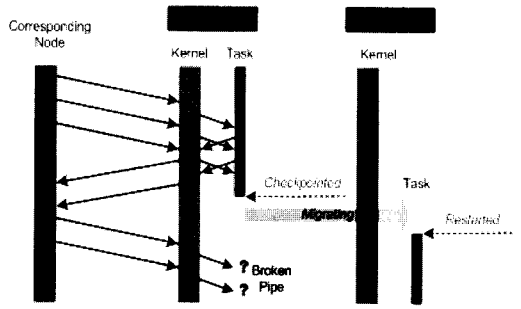
⑧ 대상 노드 연계 지원 모듈은 네트워크 주소를 대상 노드에 맞게 번역한다.

⑨ 대상 노드의 연계 지원 모듈은 옮겨진 서비스와 통신 중이던 서버측 웹 서비스에게 갱신된 통신채널 정보를 전달하고 이 정보를 받은 서비스는 이에 해당하는 채널 정보를 갱신한다.

위와 같이 서비스를 연계하게 되면 연계과정에서 옮겨진 클라이언트측 서비스의 네트워크 정보가 통신 중이던 서버측 연계 지원 모듈에 전송되어 갱신되므로 네트워크 상에서 위치 이동에 따른 통신 채널 재설정을 무리 없이 적용할 수 있다. 그러나 위와 같이 서비스를 연계할 때 채널 및 통신 상태를 완벽히 복원하기 위해서는 통신 단절(Communication Failure) 문제와 연결 재설정 오류(Connection Re-establishment Error) 문제를 해결해야 한다.

3.2 통신 단절 문제 및 해결 방법

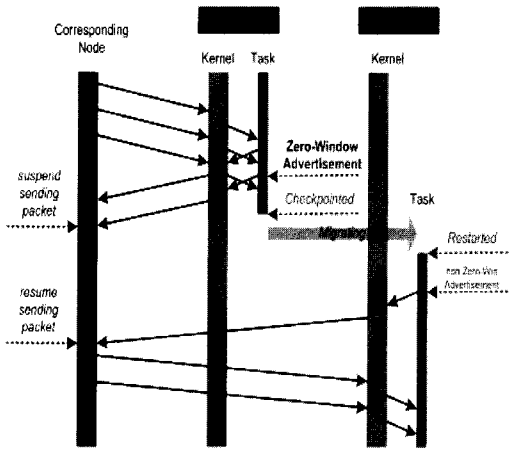
통신 채널을 사용하는 서비스 연계에서 주요 문제 중 하나는 옮겨질 서비스가 통신 채널을 사용하여 전송중인 데이터가 존재하는 경우이다. 통신 채널을 사용하여 데이터를 전송 중인 서비스를 연계할 때 네트워크 상태를 고려하지 않고 연계하게 되면 송신 중의 데이터가 소실되는 통신 단절 문제가 발생한다.



[그림 2] 통신 단절 현상

[그림 2]는 클라이언트 측에서 서비스 연계 시 통신 단절 문제가 발생하는 모습을 보여 주고 있다. 그림에서 서비스는 노드 1에서 노드 2로 옮겨졌으나 상대편으로부터 받는 데이터는 여전히 노드 1의 서버로 전달되기 때문에 데이터가 손실되는 현상을 보여 주고 있다. 이러한 통신 단절 문제를 해결하기 위해서는 서비스 연계작업 전에 통신 채널 비움(Communication Channel Flushing)을 수행해야 한다. 통신 채널 비움이란 서비스 이동이 일어나는 동안 통신 채널을 통해 메시지가 전송되지 않도록 하는 방법이다. 송신전용 서버측 서비스를 이동할 때에는 단순히 해당 서비스를 가상머신의 API를 사용하여 이동하는 것만으로도 통신 채널이 비워지는 것을 보장할 수 있다. 하지만 클라이언트 측 서비스나 서버 측이라도 수신기능을 수행하는 웹 서비스를 이동하는 경우, 이 과정에서 상대측으로부터 데이터를 송신하지 않을 것을 보장 받아야만 통신 채널이 비워짐을 확신할 수 있다. 따라서 본 논문에서는 수신측 서비스 이동 시에 송신 윈크기를 0으로 설정하는 방법을 사용하여 통신 채널을 플러쉬(Flushing)하는 방법을 제안한다.

이 방법은 수신 서비스에서 윈도우 크기를 0으로 알려줌으로써 송신측 웹 서비스 애플리케이션이 전송 윈도우 크기를 0으로 조절하도록 하여 송신측 웹 서비스로부터의 데이터전송을 중단시키는 것이다. 이 기법은 네트워크 수준의 접근방식이기 때문에 통신 채널에 있는 전송중인 데이터를 고려



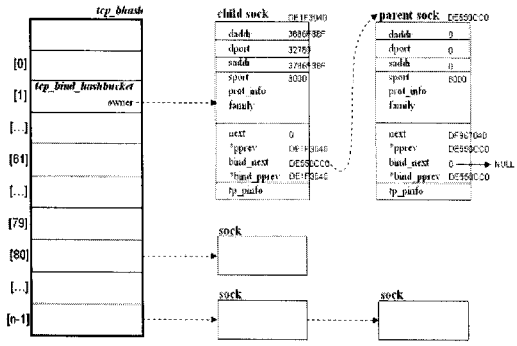
[그림 3] 제로 윈도우 알림을 이용한 데이터 손실 방지 예

할 필요 없이 정확한 서비스 연계를 가능하도록 하는 장점이 있다. [그림 3]은 이 방법을 사용하여 서비스 이동 중 패킷 전송을 중단시켜 데이터 손실을 막는 과정을 보여 준다. 지금까지 살펴본 통신 채널 비움 절차와 관련하여 통신에 참여하는 서비스가 사용중인 소켓의 역할에 따라 서비스 이동 시 수행해야 하는 작업을 구분해 보면 다음과 같다.

- 송신 서비스 응용 : 송신 서비스의 경우 해당 서비스를 체크포인팅 하게 되면 더 이상 새로운 패킷을 보내지 않으므로 특별한 전처리 없이 서비스 이동을 수행할 수 있다.
- 수신 서비스 응용 : 수신 서비스를 이동하는 경우엔 송신 측 서비스에게 0 윈도우로 설정하도록 하여 통신 채널 비움을 수행한다.
- 송수신 서비스 응용 : 서비스가 데이터를 송수신 하는 경우로 일반적인 웹 서비스 애플리케이션에서 빈번하게 발생하는 경우이다. 이 때는 앞서 나온 송신 서비스 연계에서 사용하는 방법과 수신 서비스 연계에 사용하는 0 윈도우 조정 기법을 동시에 사용하여 서비스 연계 중 데이터가 전송되는 상황을 방지한다.

3.3 연결 재설정 문제 및 해결 방법

가상화 기술 기반의 기존 서비스 연계 기법을 이용하여 서버 소켓(Server Socket)을 사용하는 서비스를 다른 서버로 이동시키면 소켓 구조를 복원하는 단계에서 클라이언트와 연결된 소켓과 감시자 소켓이 같은 포트 번호를 이용함으로 인해 포트 중복(Port Duplication) 문제가 발생한다. 이러한 연결 재설정 문제는 지금까지의 가상화 기반 서비스 이동 기법들이 네트워크 서비스에 대한 이동을 지원하지 못하던 큰 걸림돌로 존재해 왔다. 따라서 본 논문에서는 이를 해결하기 위해 소켓의 재생성시 TCP 연결 상속 (TCP Connection Inheritance) 기법을 이용하여 서버 소켓을 사용하는 서비스를 이동할 때 발생하는 연결 재설정 문제를 해결한다.



[그림 4] 연결 상속 구현을 위한 자료구조

소켓 객체의 효율적인 검색을 위해서 [그림 4]와 같은 해쉬테이블(Hashtable) 구조를 사용한다. 해쉬 키(Hash Key)로는 소켓 객체의 근원지(Source) 및 목적지(Destination)에 대한 정보가 사용되는데 바인딩(Binding)과 감시(Listening) 중에는 근원지 포트(Source Port)만을 사용하지만, 이미 연결된 소켓에 대해서는 근원지 정보와 목적지 정보의 튜플(Tuple)인 <saddr, sport, daddr, dport>을 사용하게 된다. 소켓의 바인딩 요청이 들어오면 리눅스 커널은 해쉬 함수 tcp_bhashfn에 통과시켜 얻은 해쉬 값 (Hash Value)을 바탕으로 해쉬테이블인 tcp_bhash를 검색하여 그 위치에 새로운 해쉬 버킷(Hash Bucket)을 할당한다. 만약, 새로 바인딩 하려는 소켓 객체의 정보가 이미 바인딩 되어 있는 것과 동일한 경우, 즉 서버 소켓과 일반 소켓의 경우와 같이 동일한 소스 포트를 사용하는 경우에는 해쉬 함수를 통한 해쉬 값이 일치하기 때문에 두 소켓이 같은 해쉬버킷으로 사상되어 충돌 문제가 발생한다.

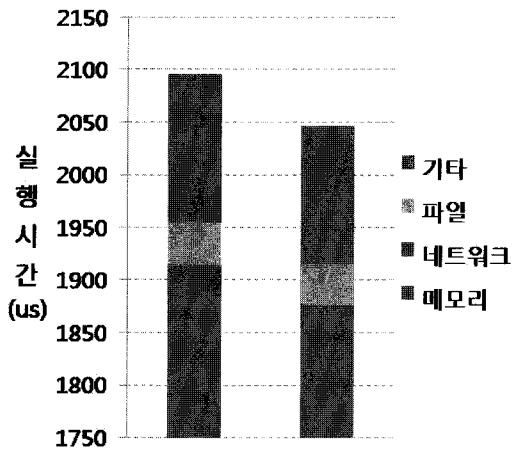
이러한 문제를 해결하기 위해 소켓 정보를 복원할 때 포트 중복으로 인해 발생하는 연결 재설정 문제를 TCP 연결 상속을 사용한다. TCP 연결 상속은 부모 소켓(Parent Socket)의 근원지 포트를 자식 소켓에서 상속받는 것으로써 tcp_bhash의 동일한 해쉬버킷에 같은 포트를 공유하는 둘 이상의 소켓 객체를 체인(Chain)으로 연결하는 방법이다. 모두 8000번의 포트를 사용하고 두 소켓 구조를 가정할 때, 이 상태에서 서비스가 이동하면 서비스 중단 및 복원 과정에서 이러한 구조를 유지해야 한다. 이를 위해, 서비스 연계 후, 서버 소켓을 먼저 생성하고 TCP_LISTEN 상태로 설정한다. 그리고 두 번째로 실제 웹 서비스의 소켓 객체를 생성하고 포트를 상속받는다. 소켓 객체의 필드(Field) 중에서 bind_next와 bind_pprev는 바인딩과 관련된 해쉬테이블에서 사용되며, next와 pprev는 서버 소켓과 개설된 소켓(Established Socket)을 위한 연결 리스트를 유지하기 위해서 사용된다. 서버 소켓의 prev에 저장 되어 있던 tb->owner에 대한 주소를 웹 서비스 소켓이 받고 자식 소켓의 bind_next 필드에 서버 소켓의 주소를 적는다. 이렇게 하면 포트 중복 문제를 유발하지 않으면서도 웹 서비스 소켓 객체를 해쉬테이블 구조에 넣을 수 있게 되어 연결 재설정 오류 없이 서버 소켓과 웹 서비스 소켓 객체를 모두 옮길 수 있다.

4. 실 험

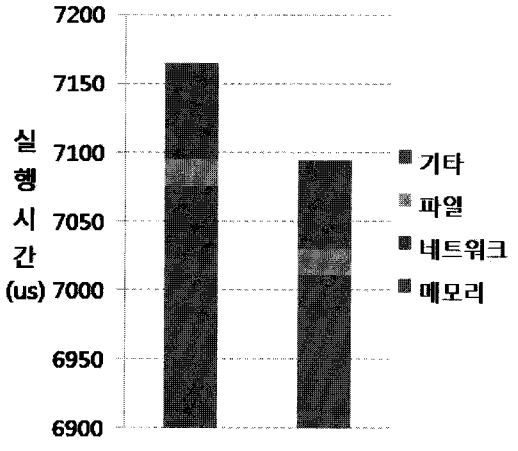
앞에서 언급한 바와 같이, 통신 단절 문제와 연

결 재설정 문제로 인해 네트워크 연결을 갖는 웹 서비스의 연계를 수행할 수 없기 때문에 다른 관련 연구와 직접적인 비교를 수행할 수 없다. 차선책으로, 네트워크 연결을 가지는 웹 서비스 연계 기능이 네트워크를 사용하지 않는 웹 서비스를 마이그레이션 하는 것에 비해 어느 정도의 오버헤드를 가지는지 평가한다. 이 때, "네트워크를 사용하지 않는 기존 웹 서비스"란 HTTP 프로토콜을 통해 제공되는 비연결 기반(Connectionless) 웹 서비스를 뜻한다. 많은 웹 사이트에서 고유의 비즈니스 로직을 구현하는데 웹 서비스를 많이 활용하고 있는데, 이러한 웹 서비스의 대다수는 주어진 상황/입력에 대해 연산을 수행하고 결과값을 반환하는 형태로 동작한다. 클라이언트 측에서 보면, 이와 같은 웹 서비스는 네트워크를 사용하지 않는 웹 서비스가 된다. 반면, 클라우드 컴퓨팅 환경의 페이스북 웹 서비스는 온라인 워드프로세서를 구현하기 위해 클라이언트측과 지속적으로 네트워크 연결 설정을 유지하는 연결기반(Connection Oriented) 웹 서비스이다. 물론, 연결기반 네트워크를 사용하지 않는 기존 웹 서비스의 경우는 별도의 연결관리 기법이 필요하지 않다. 하지만, 본 연구를 통해 제안된 웹 서비스 연결관리 기법의 통작 오버헤드를 측정하기 위해서, 네트워크를 사용하지 않는 웹 서비스에 대해서 서비스 마이그레이션 성능을 평가하였다. 결과적으로 두 방법의 실행시간의 차이를 계산하면, 순수하게 네트워크 연결을 연계하는 구현 오버헤드가 시간 단위로 나타나게 되므로 이를 제시하도록 한다.

제안하는 서비스 연계 기능은 리눅스 커널 2.6.2 기반의 클러스터 환경에서 구현하고 성능을 평가하였다. 클라우드 시스템을 타겟으로 하지만 국내에서 시스템 소프트웨어 수준에서 작업할 수 있는 실제적인 클라우드 컴퓨팅 환경이 없기에 클러스터 시스템에서 평가하였다. 본 실험에서는 서버측 웹 서비스와 클라이언트 웹 애플리케이션 사이에 핑퐁(Ping-Pong) 메시지를 주고받는 서비스를 구현하여 다른 노드로 이동시킬 때 발생하는 오버헤드를 측정하였다. 시스템 환경은 펜티엄 코어 2 듀오에서 Linux 커널 2.6.2 운영체제, 메모리 1GB를 사용하였다. 리눅스 운영체제를 채택한 것은 대표적인 오피스 웹 애플리케이션 서비스를 제공하고 있는 페이스북(facebook)의 모든 서비스가 리눅스 서버 및 클라이언트 환경을 기반으로 하기 때문이다.



[그림 5] 서비스 연계시 중단(suspend)과정의 소요 시간 비교



[그림 6] 서비스 연계시 재실행(restart)과정의 소요 시간 비교

[그림 5]와 [그림 6]에서 왼쪽 그래프는 네트워크 웹 서비스를 연계할 때를 오버헤드를 나타내고

오른쪽 그래프는 일반 웹 서비스를 이동할 때 오버헤드를 나타낸다. 웹 서비스 연계 기능은 [그림 5]와 [그림 6]에서 나타나는 바와 같이 대부분의 시간을 메모리 내의 데이터를 저장하고 복구하는데 사용된다. 실행 중인 서비스 연계에 있어 네트워크 연결을 유지해주는 핵심 기능을 구현한 소켓 구조의 저장 및 복구 기능은 중단과 재실행 과정에 대해서 각각 0.7%, 1.1%의 오버헤드만을 가진다. 따라서 본 연구에서 개발한 웹 서비스 연계기능이 효율적으로 구현되었음을 보여준다.

6. 결 론

클라우드 컴퓨팅은 차세대 컴퓨팅 패러다임에 변화와 혁명을 예고하고 있다. 와이브로와 같은 무선인터넷 서비스가 클라우드 컴퓨팅 개념과 결합하면 무한한 가능성이 열린다. 구글, 애플, 삼성, LG 등이 잇달아 내놓고 있는 스마트폰 시장에서도 클라우드 컴퓨팅이 킬러 응용(Killer Application)으로서 서비스 대중화에 기여할 것으로 보인다.

본 연구는 이동형 웹 서비스 구축을 위한 필수적인 기능인 사용자 작업환경의 끊김없는 연계를 위한 웹 서비스 연결 관리 기법을 제안하였다. 이 연구를 통해 공헌(Contribution)하는 바는 TCP 및 IP 프로토콜에 수정을 가하지 않으면서도 적은 오버헤드로 웹 서비스 애플리케이션 연계에 있어 통신단절과 연결 재설정오류를 해결하는데 있다. 본 연구에서 제안하는 통신 채널 비움과 TCP 연결 상속 기법은 적은 연결관리 오버헤드로 위 문제들을 해결함을 실험을 통해 보였다.

참 고 문 헌

[1] "Amazon's S3 Cloud Service Turns Into A Puff of Smoke", InformationWeek NewsFilter, 2008.

[2] C. Hewitt, "ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing", IEEE Internet Computing, 2008.

[3] C. E. Perkins, "Mobile IP", IEEE Communications Magazine, Vol.35, No.5(1997).

[4] D. Funato, K. Yasuda, and H. Tokuda, "TCP-R : TCP Mobility Support for Continuous Operation", IEEE International Conference on Network Protocols, 1997.

[5] Google Calandar, http://www.google.com/calendar/render?hl=ko.

[6] IBM, "Google and IBM Announced University Initiative to Address Internet-Scale Computing Challenges", http://www-03.ibm.com/press/en/pressrelease/22414.wss. 2007.

[7] IBM, "IBM Introduces Ready-to-Use Cloud Computing", http://www-03.ibm.com/press/us/en/pressrelease/22613.wss, 2007.

[8] J. Bezos, "Jeff Bezos' Risky Bet", BussinessWeek Cover Story, 2006.

[9] K. A. Deli and M. A. Walker, "Emergence of The Academic Computing Cloud", ACM Ubiquity Vol.9, No.31(2008).

[10] Mobile WiMAX : Global Opportunities, Strategies and Forecasts 2007~2013, Juniper Research, 2007.

[11] MOSSO, http://www.mosso.com

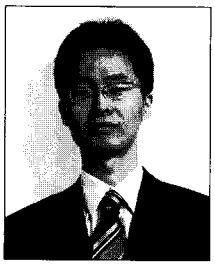
[12] P. Mika, "Web Semantics in the Clouds", IEEE Inteligent Systems, 2008.

[13] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing : Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", IEEE International Conference on High Performance Computing and Communications, 2008.

[14] W. Kim, "Cloud Computing : Status and Prognosis", Journal of Object Technology, Vol.8, No.1(2009).

[15] 이원석, 이강찬, 전종홍, 이승윤, "웹 서비스를 이용한 서비스 기반 디바이스 연동 기술", 한국IT서비스학회지, 제4권, 제2호(2005).

◈ 저 자 소 개 ◈



최 민 (miin.choi@samsung.com)

광운대학교에서 전자계산학과 학사, 한국과학기술원 전산학과 석사, 한국과학기술원(KAIST) 전산학전공 박사학위를 취득하였다. 현재는 삼성전자 반도체총괄 메모리사업부에서 책임연구원으로 재직중이다. 주요 연구관심 분야로는 서비스컴퓨팅, 웹 서비스, 클라우드 컴퓨팅, 컴퓨터구조 등이다.