소프트웨어 온디맨드 스트리밍 시스템 성능평가

김영만[0]*, 허성진**, 최완**, 박홍재*
*국민대학교 컴퓨터학부, **한국전자통신연구원 디지털홈 연구단
*{ymkim[0], hjpark0}@kookmin.ac.kr, **{sjheo, wchoi}@etri.re.kr

Performance of Software On-Demand Streaming System

Youngman Kim*, Seongjin Heo**, Wan Choi**, Hongjae Park*
*School of Computer Science, Kookmin University
**Digital Home Division, Electronics and Telecommunications Research Institute

요 약

소프트웨어 스트리밍[1][2]은 프로그램 설치 및 실행을 위하여 서버로부터 데이터 전송이 진행중인 동안에도 PC, PDA, 휴대용 개인 컴퓨터, 휴대 전화 등과 같은 데스크 탑 또는 이동 연산장치상에서 미설치 소프트웨어의 실행이 즉각적으로 이루어지도록 하는 기능이다. 즉, 소프트웨어 스트리밍 기술을 사용하면 사용자가 다운로드, 압축해제, 인스톨과 시스템 재구성이라는 일련의 과정이 완료될 때까지 기다릴 필요 없이 최소한의 다운로드 후에 해당 소프트웨어가 곧바로 실행될 수 있도록 해준다. 응용프로그램을 실행하는데 필요한 첫번째 실행 블록이 메모리에 적재되고 기본적인 환경설정을 마치자마자 나머지 블록들이 다운로드되고 설치되기도 전에 실행될 수 있기 때문에 스트리밍 시스템은 응용 프로그램의 실행준비 시간을 대폭 줄일 수 있게 해준다. 게다가, 응용프로그램 실행시 실제로 사용되지 않는 대부분의 블록들은 서버로부터 다운로드 받지 않아도 된다. 그 결과, 메모리와 대역폭 같은 리소스의 활용이 절약된다. 이러한 스트리밍 시스템을 사용하면 사용자는 다양한 공개 또는 상업용 응용프로그램을 광범위하게 지원하는 사용자 투명성을 가진 가상 소프트웨어 컴퓨팅 환경을 만들 수 있다. 본 논문에서는 프로그램 등록, 환경 변수 설정, 그리고 구성 파일과 관련된 컴포넌트들의 자동 설치 기능들을 제공함으로서 네트워크를 통하여 소프트웨어를 스트리밍하고 실행해주는 Software On-Demand(SOD)스트리밍 시스템을 설계 및 구현한다. 또한 구현된 SOD 스트리밍 시스템의 성능측정 실험환경을 구축하고 실험 결과를 이용하여 성능분석을 행한다.

1. 서론

PC, PDA, 휴대용 개인 컴퓨터, 휴대 전화 등과 같이 데스크 탑 또는 이동식의 연산 장치에서 서버로부터 응용프로그램을 다운로드하고 설치하는 작업은 많은 시간이 소모되며, Windows 와 Linux 와 같은 대표적인 컴퓨팅 환경에서 공개 혹은 상업용 응용프로그램의 설치는 초보 사용자가 이해하기 힘든 지식을 요구한다. 그러나 이러한 문제는 서버로부터 응용프로그램의 전송 및 설치가 진행 중인 동안에도 해당 응용 프로그램이 실행될 수 있도록 해주는 소프트웨어 스트리밍 기술을 통해 해결 가능하다. 또한 소프트웨어 스트리밍은 응용프로그램의 실행에 불필요한 대부분의 블록들을 서버로부터 다운받지 않아도 되기 때문에 메모리와 대역폭 같은 리소스를 매우 효율적으로 활용할 수 있게 해준다.

본 논문에서는 프로그램 등록, 환경 변수 설정, 그리고 구성 파일과 관련된 컴포넌트들을 자동 설치하는 작업들을 진행하는 동안에도 프로그램 실행에 필요한 최소한의 준비가 갖추어지는 순간 프로그램 실행을 제공하는 소프트웨어 스트리밍 실행 서비스를 제공하는 Software On-Demand (SOD) 스트리밍 시스템을 설계 및 구현한다. 본 시스템에서는 직관적인 look-and-click 방식의 가상 응용 프로그램 컴퓨팅 환경을 사용자에게 제공하며, 소프트웨어 다운로드와 인스톨 절차 및 페이지 요구작업이 백그라운드에서 자동적으로 진행되도록 리눅스 환경에서 설계 및 구현된다. 또한 구현된 SOD 스트리밍 시스템의 성능측정 실험환경을 구축하고 실험한 결과를 이용하여 성능분석을 행한다.

SOD 스트리밍 시스템을 지원하는 스트림 서버로서 Z!Stream[3]서버를 사용한다. Z!Stream 서버는 마운팅, 인증, 감사작업, 접속 관리, 세션 관리, 부하 조절 등과 같은 스트리밍과 관련된 풍부한 함수 집합을 제공한다.

2. Software On-Demand (SOD) Streaming System 설계 및 구현

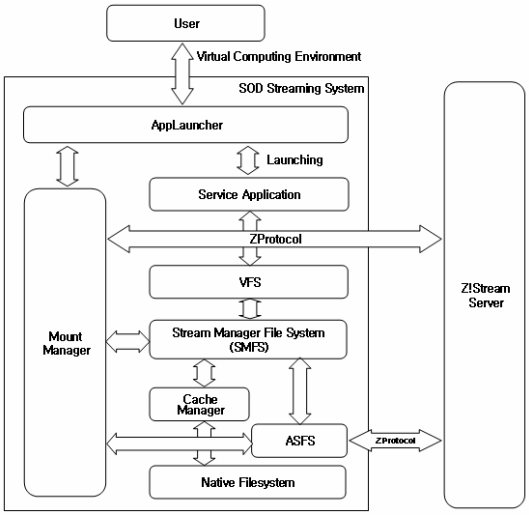
이상적인 응용프로그램 컴퓨팅 환경에서는 사용자가 다운로드, 인스톨 등과 같은 부가적인 업무에 시간과 노력을 투입하지 않아도 본 업무를 처리할 수 있도록 부수적인 작업을 자동으로 처리하는 기능을 지원해준다. 사용자가 문제를 해결하기 위해 특정 응용 프로그램을 실행하고자 할 때 데스크 탑 상에서 해당 응용프로그램을 상징하는 아이콘을 클릭하면 프로그램 초기 실행에 필요한 모듈들이 메모리에 로딩되자마자 프로그램의 UI 윈도우가 화면에 나타나게 되며 이 프로그램이 최초로 호출되는 경우에는 추가적으로 프로그램 다운로드와 인스톨 절차가 백그라운드에서 진행된다.

이번 장에서는 Software On-Demand(SOD) Streaming System이라 불리는 스트리밍 시스템을 제안한다. 제안된 SOD 시스템은 다음과 같은 특징을 가진다. 첫째로, SOD 시스템은 복잡한 인스톨 작업으로부터 사용자의 수고를 덜어준다. 두

번째로, 소프트웨어 개발자에게 새로운 제품의 광고와 유통을 위한 강력한 수단을 제공한다. 세 번째로 사용자는 새로운 프로그램을 실행하는 경우에 UI 윈도우의 빠른 팝업을 경험하게 된다. 네 번째로 SOD 스트리밍은 프로그램이 사용하는 데이터 파일을 모두 다운로드하지 않고도 응용 프로그램이 데이터 파일을 액세스할 수 있도록 해준다. 마지막으로 SOD 스트리밍은 서버에서 직접 스트리밍되기 때문에 소프트웨어 배포와 업데이트를 용이하게 한다.

2.1 SOD Streaming System Architecture

[그림 1]은 SOD 스트리밍 시스템의 구조를 보여주고 있으며 가상 컴퓨팅 환경을 만들어 다운로드와 인스톨 절차를 사용자로부터 은폐시킨다. 가상 컴퓨팅 환경의 주된 목적은 각 응용 프로그램 소프트웨어를 위해 응용 프로그램에 종속적인 프로그램 레지스트리, 환경 변수, 구성 파일, 관련 컴포넌트에 대한 컴퓨팅 환경을 사용자가 인식하지 않는 상태에서 자동적으로 설정·구축하는데 있다.



[그림 1] SOD 스트리밍 시스템

사용자가 새로운 응용 프로그램을 최초로 선택하면 SOD 시스템은 환경 관련 데이터를 스트리밍 서버로부터 수신한다. 환경 설정이 끝나게 되면 SOD 시스템은 서버에 프로그램 실행을 위한 최소한의 바이너리 실행 페이지들을 요청한다.

SOD 사용자 인터페이스 프로그램(AppLauncher)은 사용자에게 SOD 를 통하여 실행가능한 소프트웨어 아이콘들을 보여준다. 사용자가 AppLauncher 윈도우에서 임의의 아이콘을 선택하면 SOD 시스템 컴포넌트 중의 하나인 Mount Manager(MM)은 응용 프로그램 환경 데이터를 다운로드하고 인스톨하는 작업을 수행한다. MM 은 이러한 작업을 위해 Z 프로토콜을 이용하여 해당 데이터를 스트림 서버에 요청한다. 응용 프로그램 컴퓨팅 환경에서 응용프로그램을 위한 초기 설정이 끝나면 AppLauncher 는 응용 프로세스를 발진하기 위하여 실행 이미지 블록들을 OS 에게 요청한다. 이때 프로그램 이미지와 데이터 파일을 위한 I/O 가 발생하고 해당 요청은 Linux Virtual File System(VFS) 모듈을 경유하여 SOD 시스템의 세번째 컴포넌트인 Stream Manager File System (SMFS)에 도착한다. SMFS 는 우선 SOD 시스템의 네 번째 컴포넌트인 Cache Manager(CM)를 경유하여 로컬 디스크 캐시를 검색한다. 만일 페이지가 로컬 캐시에서 발견되지 않으면 SMFS 는 스트리밍 서버에게 해당 페이지 요구 메시지를 보내기 위해 다섯번째 컴포넌트인 Application Streaming File System(ASFS)를 호출하며 ASFS 는 Z 프로토콜을 통해 서버로부터 해당 페이지를 전송받는다. ASFS 를 통하여 입수된 페이지는 재사용을 위해 로컬 디스크 캐시에 저장되고 동일한 내용이 CM 에 생성된 후 프로그램 실행이 중지된 위치에서 재개된다. 현재 SOD 시스템은 리눅스 상에서 구현완료된 상태에 있다.

2.2 SOD System Modules

이 절에서는 본 시스템을 구성하는 SOD 각 모듈에 대하여 자세히 설명한다.

2.2.1 AppLauncher

AppLauncher 는 사용자가 원하는 응용 프로그램을 손쉽게 찾아내고 선택할 수 있도록 웹 브라우저와 유사한 유저 인터페이스를 제공한다. 사용자가 실행하고자 하는 응용 소프트웨어를 상징하는 아이콘을 클릭할 때, AppLauncher 는 Mount Manager 에게 응용 소프트웨어 실행을 위한 가상 컴퓨팅 환경을 확립하도록 지시한다. 그 후 fork()와 exec() 시스템 호출에 의해 응용 프로그램 프로세스를 개시한다.

2.2.2 Mount Manager (MM)

리눅스는 다중 사용자용 실행 환경을 제공한다. 그러므로, Mount Manager 역시 다중 AppLauncher 프로세스 정보를 유지해야 한다. 각 AppLauncher 가 개별 응용 프로그램 컴퓨팅 환경을 생성하도록 MM 에게 지시하면 MM 은 로컬 캐시로부터 상응하는 환경 데이터를 검색한다. 만일 데이터가 캐시에서 발견되지 않으면, MM 은 스트리밍 서버에게 환경 데이터 요구 메시지를 보낸다. MM 은 도착된 환경 데이터를 사용하여 가상 컴퓨팅 환경을 구축하고 관리한다.

2.2.3 Stream Manager File System (SMFS)

SMFS 는 리눅스 VFS 와 네이티브 파일 시스템(예를 들면, ext2) 사이에 위치하는 SOD 전용 가상 파일 시스템 모듈이다. 응용 프로그램 프로세스 혹은 MM 이 프로그램을 시작하거나 데이터 페이지를 요구할 때 해당 호출은 리눅스 VFS 를 경유하여 SMFS 에 도착한다. SMFS 는 우선 Cache Manager 를 호출하여 로컬 캐시내에서 해당 페이지를 찾는다. CM 은 페이지의 내용 혹은 미발견 오류정보를 SMFS 에게 리턴한다. 후자의 경우 SMFS 는 해당 페이지 데이터를 얻기 위해 ASFS 를 통하여 스트리밍 서버에게 요구하며 서버로부터 새로운 페이지 데이터가 도착하면 ASFS 를 통하여 SMFS 에게 배달된다. 그 후 해당 페이지는 CM 을 경유하여 로컬 디스크 캐시에 저장되고, 메모리에도 로딩된 후에 응용 프로그램 프로세스 혹은 MM 의 실행이 재개된다.

2.2.4 Cache Manager (CM)

데이터 재사용성을 증진시키기 위해 로컬 디스크 캐시에 저장되는 데이터는 응용 프로그램 스트리밍 서비스가 보다 신속히 진행될 수 있도록 도와준다. 즉, 응용 프로그램 프로세스는 네트워크 라운드 트립 지연을 생략함으로써 사용자에게 신속한 프로그램 실행을 제공하며 귀중한 네트워크 자원을 절약한다. CM 은 빠른 페이지 검색을 위해 내부적으로 인덱스 구조를 가지고 있다.

2.2.5 Application Streaming File System (ASFS)

ASFS 는 데이터 페이지 요구를 Z 프로토콜 메시지에 실어서 서버로 보내주며, 서버로부터 도착한 메시지 내의 페이지 데이터를 SMFS 에 전달한다.

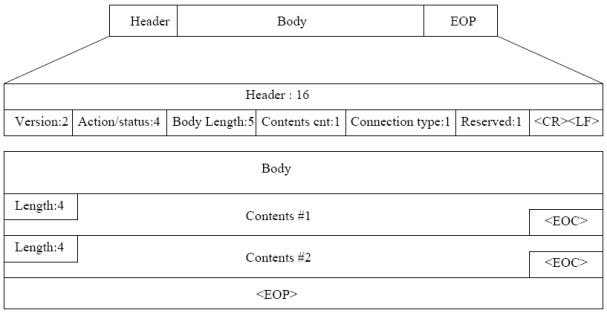
2.3 Z Protocol

Z 프로토콜은 SOD 스트리밍을 위해 설계된 응용 프로토콜

이다. 실시간 QoS 필요조건을 만족시켜야만 하는 멀티미디어 스트리밍과는 대조적으로 SOD 스트리밍은 신뢰할 수 있는 메시지 전달을 필요로 한다. 따라서 Z 프로토콜은 TCP 트랜스포트 프로토콜 상위에 구축된다.

메시지에는 제어와 데이터 메시지 2 가지 종류가 있다. 제어(데이터) 메시지는 RASCP(RASP) 보조 프로토콜에서 정의되고, MM(ASFS)과 스트리밍 서버 사이에서 사용된다. MM 은 RASCP 를 경유하여 환경 데이터, 인증, 접속 관리, 세션 관리 등의 제어 정보를 송신한다. 반면에 ASFS 는 프로그램 또는 데이터 페이지에 RASP 을 사용하여 요구 메시지를 보낸다.

RASCP 와 RASP 모두 공통의 요구-응답 통신 패러다임을 따르므로 공통 Z 프로토콜 해더 포맷을 공유한다. Z 프로토콜의 포맷은 [그림 2]와 같이 구성된다.



[그림 2] Z 프로토콜

3. SOD 스트리밍 시스템 성능평가

이 장에서는 2 장에서 설계 및 구현한 SOD 스트리밍 시스템을 성능평가한다. SOD 스트리밍 시스템 사용자의 만족도를 평가할 수 있는 성능평가 실험을 행하고 결과분석을 통해 사용자에게 가상 컴퓨팅 환경에서 보다 높은 서비스 만족도를 제공할 수 있는 성능개선 방법을 제안한다.

3.1 Experimental Setup

[표 1]에서 보여주는 것처럼 SOD 성능측정 실험은 두 가지 다른 네트워크 환경(인터넷과 인트라넷)에서 진행하였다. 보다 설득력 있는 측정을 위해 응용 프로그램 패키지의 크기가 2.4MB 에서 438MB 까지 분포되어 있는 여섯 가지 소프트웨어들을 사용하였고 실시간 패킷 추출과 분석을 위한 도구로서 Ethereal Network Protocol Analyzer 를 사용했다. SOD 테스트는 대전에서 이루어졌고 인터넷 스트리밍 서버는 대전에서 140 km 떨어진 곳에 위치한 서울의 Top-Mania IDC 센터에 설치된 서버를 사용하였다.

System Environment		
Streaming Server	Internet Server	Top-Mania IDC Center in Seoul(serviceonnet.net)
	Intranet Server	Pentium 4 2.8GHz, 512M, Redhat 9.0
Client	Pentium 4 3GHz, 512M, Windows xp	
Network Environment		
Internet	www.serviceonnet.net (218.234.21.123) Connected to 1Gbps Backbone	
Intranet	soda6.etri.re.kr (129.254.189.126) Intel Pro/1000 Ethernet(Gigabit), Connected to 100Mbps Hub	
Application		
Burn4Free v 1.0	2.4 MB	Burning CD-ROM Utility
Piscasa v 1.6	8.5 MB	Management Tool for Digital Image
Nvu v 0.41	20.6 MB	HTML Tool
Acrobat Reader v 6.0	87.6 MB	PDF viewer
UCAD v 4.2	141 MB	Design tool
Hwp 2004	438 MB	Word Processor
Measure Time and Policy		
morning(09:00~12:00), afternoon(14:00~17:00), night(19:00~22:00)		

[표 1] SOD 시스템 성능 측정 실험 환경

3.2 Performance Metrics

대화형 응용 프로그램 환경이 사용자에게 신속히 제공되는데 사용되는 성능평가 요소로서 fast UI Show-up 과 최소 대기 시간을 반영하는 측정지수들을 선정하였다.

Application Load Time(ALT)은 프로그램이 선택되어 실행되고 난 후부터 초기 UI 윈도우가 화면에 표시될 때까지의 경과시간이다. ALT 는 환경 데이터의 크기, 프로그램의 초기화 또는 런칭에 관련된 데이터에 직접적으로 비례하고 전송 매체의 속도에 반비례한다. 따라서 낮은 ALT 값은 프로그램이 빨리 실행된다는 것을 뜻한다.

보통 응용 프로그램 실행시 다음 명령이 메모리에 로드 되어 있지 않거나 부분적으로 로드 되었을 때 응용 프로그램의 실행 중단이 발생하며 요구된 페이지가 스트리밍되고 있는 동안에 응용 프로그램은 중단되어야만 한다. 이때 메모리나 디스크 캐시에 해당 페이지가 없을 때 최악의 정지 시간이 발생한다. 이러한 경우 정지 시간은 해당 페이지의 스트리밍 전송 완료 시간이 된다.

반면에 페이지가 메모리에 이미 존재하고 있을 때 최적의 상황이 발생한다. 그러므로 실제 응용 프로그램의 정지시간은 0 에서부터 전체 페이지를 스트림 서버로부터 네트워크를 경유하여 로드하는 시간 사이에 분포한다. Application Suspension Time(AST) 혹은 어플리케이션 중단시간은 임의의 기능이 선택된 순간부터 그 기능이 실행된 결과 화면이 나올 때까지의 시간으로서 스트리밍된 페이지의 크기에 비례한다. 응용 프로그램은 중지하고 있는 동안 사용자와 상호 작용 할 수 없다. 많은 상호작용을 포함하고 있는 응용 프로그램에 있어서 사용자 만족도 향상을 위하여 낮은 중단시간은 매우 중요하다.

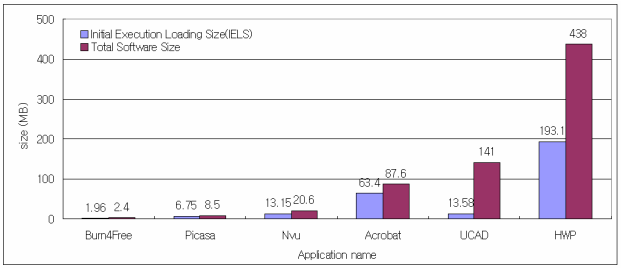
3.3 Experimental Results and Analysis

이번 절에서는 Application Load Time(ALT)을 측정하였다. 측정 데이터의 정확성을 높이기 위해 측정한 데이터를 몇 개의 세트로 구성하여 이들의 평균을 구하였다. 새로운 응용 소프트웨어의 최초 실행시의 초기 ALT 를 측정하였다. 또한 동일한 프로그램에 대하여 재실행시 페이지 캐싱 메커니즘이 가진 성능향상 정도를 밝히기 위하여 재실행 ALT 값을 측정하였다. Initial Execution Loading Size(IELS), 즉 최초의 ALT 기간 동안 다운로드 되는 패킷 데이터의 크기 또한 측정하였다.

[표 2]와 [그림 3]은 각 응용 프로그램의 전체 소프트웨어 크기에 대한 IELS 를 보여준다. UCAD 를 제외한 모든 응용 프로그램의 IELS 가 전체 소프트웨어 크기의 40~80%에 이른다. 이는 신규 프로그램의 초기 실행을 위하여 클라이언트가 많은 패킷을 다운로드해야만 하는 것을 의미한다.

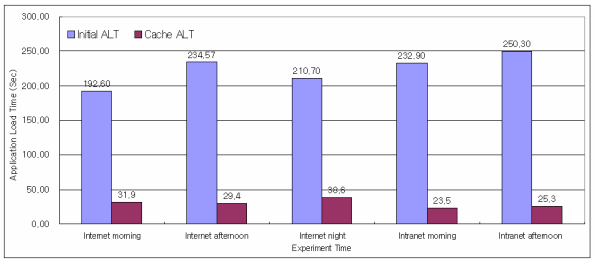
Application program	Initial execution loading size (IELS)
Burn4Free v 1.0	2,057,364 Byte (1.96 MB), of total size 81%
Piscasa v 1.6	7,084,438 Byte (6.75 MB), of total size 79%
Nvu v 0.41	13,788,539 Byte (13.15MB), of total size 63%
Acrobat Reader v 6.0	66,102,317 Byte (63.04 MB), of total size 72%
UCAD v 4.2	14,246,774 Byte (13.58 MB), of total size 9%
Hwp 2004	202,483,982 Byte (193.1 MB), of total size 44%

[표 2] 초기 ALT 동안 다운로드 된 IELS

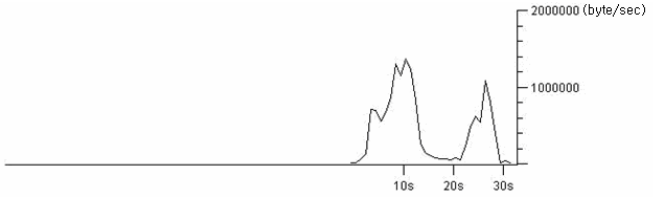


[그림 3] 응용 프로그램 별 IELS 와 전체 크기

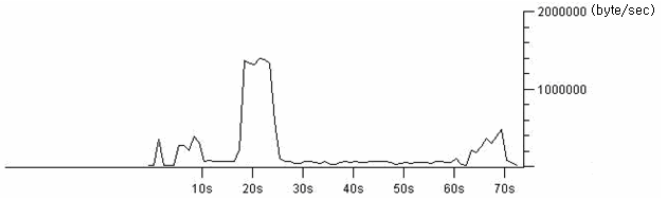
[그림 4]는 가장 큰 IELS 와 ALT 를 가지는 HWP2004 에 관한 실험 결과를 보여준다. HWP2004 의 최초 실행은 약 3~4 분이 걸린다. 그러나 첫 번째 실행 이후 캐시 ALT 는 필요한 데이터 페이지를 로컬 디스크 혹은 캐시에서 가져오기 때문에 실행시간이 30 초로 줄어든 것을 확인할 수 있다.



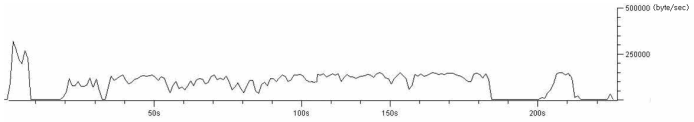
[그림 4] HWP2004 의 초기 ALT 와 캐시된 ALT 의 평균



[그림 5] 초기 ALT 동안 Nvu 의 트래픽 흐름



[그림 6] 초기 ALT 동안 UCAD 의 트래픽 흐름



[그림 7] 초기 ALT 동안 HWP2004 의 트래픽 흐름

[그림 5,6,7]은 세 가지 대표적인 응용 프로그램(Nvu, UCAD, HWP2004)의 초기 ALT 기간 동안 패킷 전송속도(byte/sec)를 보여준다. 초기의 ALT 기간 중 모든 응용 프로그램은 최대 네트워크 대역폭보다 훨씬 작은 일부분의 대역폭만을 사용한다. 이러한 현상은 응용 프로그램 프로세스가 간헐적으로 페이지를 요구한다는 것을 보여준다. 한편, 서버는 대부분의 ALT 기간 동안 CPU 시간과 네트워크 대역폭을 충분히 활용하지 못하고 리소스 사용도가 매우 적은 상태에 머물러 있다.

따라서 응용 프로그램의 로드 시간을 최적으로 줄이기 위하여 응용 프로그램의 실행에 필요한 페이지를 초기 ALT 기간에 집중적으로 요구하는 Accelerator Initiation Accelerator(AIA) 방법과 응용 프로그램의 정지시간을 줄이기 위하여 응용 프로그램이 자주 사용되는 페이지를 통계적인 접근을 통하여 미리 다운받을 수 있도록 하는 Statistical Predictor Prefetching(SPP)을 성능개선안으로 제안한다.

4. 결론

본 논문에서는 프로그램 등록, 환경 변수 설정, 구성 파일 및 관련 컴포넌트의 자동 인스톨 기능을 자동적으로 실행함으로서 응용 프로그램의 신속한 스트림 실행을 제공하는 Software On-Demand Streaming System (SOD)을 제안하였다. 또한 다양한 리눅스 응용 프로그램 소프트웨어에 대한 SOD 스트리밍 실험을 통하여 성능측정한 실험의 결과 분석을 행하였다. 분석 결과를 바탕으로 두가지 성능개선 방법(AIA, SPP)을 제안하였으나 향후 실험을 통한 검증이 필요하다.

참고문헌

[1] Kuacharoen, P., Mooney, V., and Madisetti, V., "Software streaming via block streaming," Proceedings of the Design Automation and Test in Europe, pp. 912-917, Mar. 2003.

[2] Kuacharoen, P., Mooney, V., and Madisetti, V., "Efficient Execution of Large Applications on Portable and Wireless Clients,", Proceedings of the Mobility Conference & Exhibition, Aug. 2004.

[3] SOFTonNET Inc., "Z!Stream Technology," http://www.softonnet.com.