특화된 데이터 처리를 위한 크롬 OS 의 최적화 방안

이슬기*, 유헌창**
*고려대학교 컴퓨터정보통신대학원
**고려대학교 대학원 컴퓨터학과
e-mail : {punky111, yuhc}@korea.ac.kr

Optimization Technique of Chrome OS for Specialized Data Processing

SeulGi Lee*, HeonChang Yu**
* Graduate School of Computer & Information Technology, Korea University
** Department of Computer Science and Engineering, Korea University

요 약

크롬 OS 는 구글에서 개발한 운영체제로서 리눅스와 HTML5 기술 등을 도입해 빠른 부팅과 안정적인 프로그램 실행 등을 통해서 윈도우로 대표되어왔던 기존 컴퓨터 운영체제들의 불편함을 개선한 것이 가장 큰 특징이다. 그러나 클라이언트용 OS 라는 점을 감안하였을 때 사용자층이 점차 늘어나고 있는 그래픽이나 암호화, 복호화 같은 보다 세분화된 다양한 데이터 처리를 하는데 있어 요구사항을 만족 시키기에는 부족한 부분이 많았다. 이 같은 문제점들은 운영체제 최적화를 통해 해결할 수 있으며, 이 논문에서는 크롬 OS 의 가상메모리 설정 변수의 최적화 수치를 찾아 개선하는 방안을 제안한다.

1. 서론

온라인 환경이 발전을 거듭하면서 그 만큼 소프트웨어의 성능도 나날이 발전을 거듭하고 있다. 하지만 대부분의 사람들은 소프트웨어를 사용하면서 자신이 사용하는 하드웨어 성능을 최대한 끌어내어 사용하지 못하고 있다. 그로 인해 자신도 모르게 하드웨어의 전력소모 낭비, 처리속도 지연, 불필요한 자원 점유 등을 초래하고 있다. 그래서 이 문제를 크롬 OS 가상메모리 (Virtual Memory) 디렉토리 내부에 존재하는 설정 변수 dirty_ratio, dirty_writeb ack_centisecs, Swappiness, vfs_cache_pressure 를 사용해 처리하고자 한다. 프로세스 처리 향상의 경우에는 디스크로 내보내는 설정 값들을 낮추고, 그래픽 처리 향상의 경우에는 설정 값들을 반대로 설정해 나가는 방식을 취할 것이다. 이를 통해 최적의 결과값 확인 및 효율적인 운영이 가능하도록 유지시키기 위한 방안들을 살펴보겠다

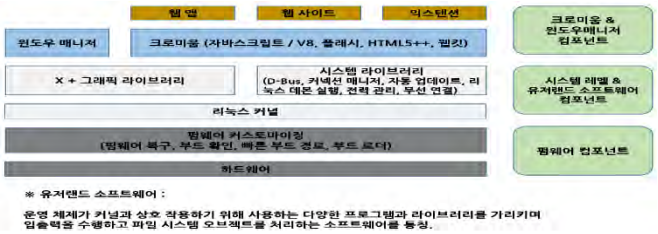
본 논문의 구성은 다음과 같다. 2 장에서는 기존의 크롬 OS 와 리눅스의 최적화와 관련된 논문 확인 및 최적화 방안을 소개하고 있으며, 3 장에서는 실험 환경 및 연구 방법에 대해서 소개하고 있다. 4 장에서는 위에서 소개한 시스템에서의 실험 결과 및 평가에 대해서 소개하고 있으며, 마지막 5 장에서는 논문에 대한 결론을 내린다.

2. 관련 연구

2.1 구글 크롬 OS

현재 사용되고 있는 많은 클라우드 OS 들중 오픈소스로 공개된 크롬 OS 는 기존의 OS 들과는 달리 PC 의 저장장치에 소프트웨어를 설치하지 않고, 구글 크롬 웹 스토어에서 필요한 앱들을 설치할 수 있다는 점이 큰 특징이다. 여기에 SSD 만을 지원하는 크롬 OS 의 특성을 활용해 크롬북 사용시 빠른 부팅이[1] 가능하다는 것과 와이파이나 LTE 통신망으로 인터넷에 접속할 경우 구글 서버에 보관되어 있는 유저들의 개인 자료를 공간 제약 없이 자유롭게 업로드와 다운로드 할 수 있다는 점에서 큰 주목을 받고 있다.

(그림 1)의 크롬 OS 구조를 살펴보면 리눅스를 사용한다는 것을 확인할 수 있다. 한편 크롬 OS 는 펌웨어 컴포넌트, 시스템 레벨 & 유저랜드 소프트웨어 컴포넌트, 크로미움 & 윈도우 매니저 컴포넌트로 구성되어 있다. 각 컴포넌트들의 특징은 다음과 같다.



(그림 1) 크롬 OS 구조

펌웨어 컴포넌트

: OS 를 부팅하는데 있어 불필요한 부분을 제거해서 더 빠르고 Secure 하게 동작하도록 이루어져 있다.

[1] Chromebooks boot up in less than ten seconds, Samsung Electronics Corporation, “Samsung Chromebook for Education”, 2013.

①시스템 복구: 시스템 이상이 발생할 경우 크롬OS 재설치를 통해 펌웨어를 복구한다.

②부트 확인: 펌웨어에서 실행되며 기기의 펌웨어, 커널, 시스템 이미지의 변경 및 손상 여부를 확인한다.

③빠른 부트: 일반적인 PC 펌웨어와 비교했을 때 (플로피 드라이브 탐지 같은) 복잡하거나 필요치 않은 부분들을 제거해 부트 성능을 개선한다.

시스템 레벨 & 유저랜드 소프트웨어 컴포넌트

: 부트 성능 개선을 위해 리눅스 커널과 드라이버, 유저랜드 소프트웨어를 차용해 만든 컴포넌트이다. 서비스들의 병행 실행과 충돌 발생시 재실행을 시키는 역할을 하는 Upstart 로부터 관리되는 유저랜드 서비스들의 초기화 과정은 간결하다. 하지만 실행을 담당하는 서비스들은 매우 중요한 역할을 차지한다.

①D-Bus: 브라우저를 사용해 시스템의 유휴 상호작용을 관리하는 기능을 담당한다.

②커넥션 매니저: 공통 API 제공을 통해 네트워크 장치와 DNS 프록시 공급 등을 담당한다.

③WPA (Wi-Fi Protected Access) Supplicant: 무선랜 네트워크 WPA 암호화 인증 사용을 관리한다.

④자동 업데이트: 새로운 시스템 이미지 설치를 담당하는 데몬이다.

⑤전력관리: 전원이 실행되거나 종료될 경우 발생하는 전력관리 이벤트를 담당한다.

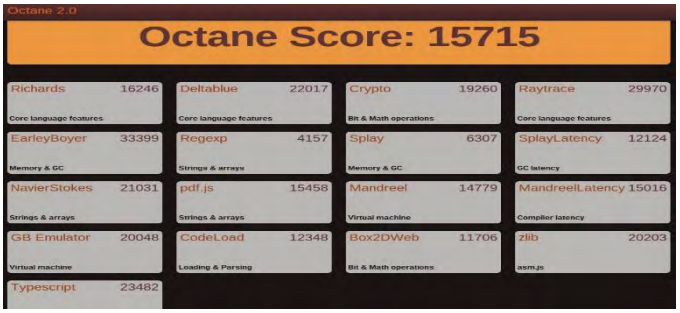
⑥리눅스 데몬 실행: NTP, Syslog, Cron. 같은 리눅스 표준 서비스들을 담당한다.

크로미움 & 윈도우매니저 컴포넌트

윈도우 매니저는 사용자와의 다중 윈도우 클라이언트 상호작용에 응답하기 위해서 처리하는 역할을 하는 컴포넌트이다. 윈도우 브라우저의 범위를 벗어난 단축키 확인 입력 포커스 (input focus) 할당, 윈도우 배치 컨트롤러의 기능을 함으로써 다른 X 윈도우 매니저들과 비슷한 기능을 갖고 있다.

2.2 크롬 OS 및 리눅스 최적화 방법

크롬 OS 는 젠투 리눅스를 변형해 개발된 운영체제다[1]. 젠투 리눅스의 큰 특징이라면 흔히 사용하는 레드햇 엔터프라이즈 리눅스 또는 우분투 리눅스와는 달리 바이너리 파일 배포판 없이 직접 컴파일을 통해 최적화 시켜 설치한다는 것이다. 따라서 크롬 OS 역시 컴파일링과 알맞은 패키지 설치 등을 통해 사용하고자 하는 기기에 적합하도록 최적화 시키는 기술이 필요하다. 다만 웹 브라우징에 맞게 릴리즈된 크롬 OS 에서는 젠투 리눅스와 달리 성능 측정을 위한 패키지들 설치나 접속을 하는데 있어 많은 제약조건들이 존재한다. 그래서 크롬 OS 에서는 성능 측정이 가능한 Octane 라는 서비스를 제공하고 있다.



(그림 2) 크롬 OS Octane 의 사용과 출력 예시

논문 [2] 에서는 크롬 OS 코드 변형을 통한 부트 시간 감소와 관련된 연구를 목표로 한다. 크롬 OS 코드 변형은 초소형 PC 인 비글보드에서 사용하는 리눅스 변형 웹 운영체제 'Awesom-O' OS 의 최적화 방식을 이용해 하드웨어 부트 체킹 제거 방식으로 부트 시간은 35 초에서 26.2 초로 단축된다. 하지만 하드웨어 부트 체킹 과정 없이 부트되는 시간만 줄이는 것을 목표로한점은 많은 개선점이 필요하다.

논문 [3] 에서는 앞서서 소개했던 논문들과 비교했을 때 시스템 부하 감소, 디스크 • 가상 메모리 시스템 • 커널 등을 튜닝하는 방안에 대해서 자세히 소개한다. 다양한 튜닝 툴과 모니터링 툴을 사용해 설정값들을 파악한 후 인프라 환경을 고려해 알맞은 설정값 지정을 통한 성능 향상이 목적이다. 다만 레드햇 엔터프라이즈 리눅스 위주로 작성되어 있어 클라이언트 OS 시점이 아닌 서버 OS 시점이 주를 이루어 신중한 검토가 요구된다.

논문 [4] 에서는 리눅스 가상 메모리의 Drop_caches 와 Vfs_cache_pressure 변수를 사용해 캐시 메모리 공간의 낭비를 줄이고 컴퓨터의 속도를 올렸다. 논문의 목적은 작성한 스크립트에 가상 메모리 변수 값을 입력시켜 메모리와 디스크를 효율적으로 사용함과 동시에 CPU 사용률과 메모리 가용량을 안정적으로 유지시키는 것이다. 그러나 리눅스의 설정 변수들 중에서 Drop_caches 와 Vfs_cache_pressure 변수만을 사용하였기에 OS 의 풀 오토매틱 제어를 기대하기에는 한계가 있다. 따라서 본 논문에서는 이외의 가상메모리 설정 변수들도 이용한다.

논문 [5] 는 리눅스의 더티 페이지가 디스크에 쓰이도록 해주는 커널 쓰레드를 이용해 캐시의 현재상태를 조절하고자 하는데 있다. 연구 내용을 살펴보면 각 Ratio 가중치에 따라 비율과 쓰기 응답시간에서 변화가 있다. 그렇지만 논문 [3]과 마찬가지로 SSD 기반인 크롬 OS 에 적용하는데 많은 보완점이 필요하다.

논문 [6] 에서는 MS-DOS 호환 1.44MB 디스켓 내부에 삽입할 수 있는 저용량의 내장형 리눅스 최적화

기술에 대한 연구를 목적으로 하고 있다. 연구 방법은 커널 최적화, 라이브러리 최적화, 컴파일 최적화 & 부팅 절차 단순화를 하는 방식을 선택했다. 그러나 MS-DOS 방식과 1.44MB 방식의 디스켓을 사용하는 환경이라는 점과 컴파일러나 라이브러리 등의 환경에 의해서 결과값은 전혀 달라질 수 있거니와 SSD 만이 사용 가능한 크롬 OS 에 적용하기에는 많은 한계점이 존재하기에 다른 차원의 연구가 필요하다.

크롬 OS 의 VM (가상메모리) 관리 설정 변수

크롬 OS 의 VM 을 설정하는 디렉터리들은 /proc/sys /vm 에 위치해 있으며 캐쉬 메모리를 비우거나 디스크 공간 설정 등을 통해서 하드웨어의 성능을 조절할 수 있다. 본 논문에서 사용하고자 하는 설정 값들과 기능은 <표 1>과 같다. (기본값은 크롬 OS 기준이다.)

<표 1> 크롬 OS VM (가상메모리) 관리 설정 변수

VM 설정 변수	기능
dirty_ratio	Pdflush 를 통해 더티 데이터 생성기가 전체 메모리에서 더티 데이터의 라이트백이 (Write -back) 시작되어 디스크로 옮겨지는 최대 비율 설정 변수다. 기본값은 60 이다.
dirty_writeback_centisecs	정의된 Pdflush 데몬 웨이크업 인터벌 사이에, 더티 인-메모리 데이터들을 주기적으로 디스크로 내보내 라이팅을 시키기 위한 설정 변수이다. 기본값은 60000 이며, 단위는 0.01 초 이다. (60000 = 10 분).
Swappiness	사용중인 컴퓨터에 스왑을 얼마나 사용할지에 대해 설정하는 변수. 기본값 60 보다 높으면 스와핑을 메모리에서 하드로 자주 전송할 것이며, 작게 한다면 반대 상황이 발생한다.
vfs_cache_pressure	아이노드 오브젝트와 디렉터리 캐싱을 사용해 메모리를 반환시키는 커널 성향을 조절한다. 기본값은 100 이다.

3. 실험

3.1 실험 환경

본 논문에서는 위에서 설명한 VM (가상메모리) 관리 설정 변수를 이용하여 실험을 진행했으며 게스트 모드로 로그인하여 영향을 줄 수 있는 요인들을 최대한 줄일 수 있도록 했다. 환경에 따라서 다양한 결과값을 보기 위해 가상 머신과 크롬북에서 실험한다. 사용한 각각의 모델 및 시스템 환경은 <표 2> 및 <표 3>과 같다. 괄호 안은 오라클 버추얼 박스 가상 머신 환경이다.

<표 2> 실험환경 1 (LG 전자 15ND530-PX7SK) (16GB 로 업그레이드)

CPU	인텔 i7-4702MQ 2.20Ghz 쿼드코어 (Intel i7-4702MQ 2.20Ghz 듀얼코어)
RAM	16 GB (2 GB)
HDD	500 GB (6 GB)
크롬 OS 버전	28.0.1484.1 4028.0.2013_04_20_1746 (Developer Build - hexxeh) vanilla x86-generic
커널 버전	Linux Version 3.4.0 (Chrome-bot@build107-m2)

크롬북의 일반 모드에서는 쉘을 사용할 수 없도록 설정되어 있다. 따라서 크롬북의 크롬 OS 에서 VM 을 (가상메모리) 최적화 시키기 위해 크롬 OS 디벨로퍼 모드로 진입한 후 실험한다[7].

<표 3> 실험환경 2 (삼성전자 시리즈 3 XE303C12 크롬북)

CPU	삼성 엑시노스 5 1.7 GHz 듀얼코어
RAM	2 GB
HDD	16 GB
크롬 OS 버전	48.0.2564.116 7647.84.0 (Official Build) stable-channel daisy
커널 버전	Linux Version 3.8.11 (Chrome-bot@build72-m2)

3.2 실험방법

크롬 OS 최적화를 실험하기 위해서 Sysctl 커맨드를 통해 각각의 VM 설정 변수에 알맞은 변경값을 삽입하여 Crypto 등의 해당하는 영역에 크롬 OS Octane 수치를 기준으로 처리 능력이 더 뛰어난 경우를 알아낸다. 할당한 변경 값은 최적화 수치를 입력해가며 최상의 값을 찾았으며 관련 값은 <표 4> 및 <표 5>와 같다. 나머지 하위 /proc/sys/vm 파일 설정값들은 기본값으로 두었다.

<표 4> 프로세스 성능 최적화의 VM 설정 변수값

VM 설정 변수	기본값	변경값
dirty_writeback_centisecs	60000	87966
dirty_ratio	60	38
Swappiness	60	49
vfs_cache_pressure	100	80

<표 5> 그래픽 성능 최적화의 VM 설정 변수값

VM 설정 변수	기본값	변경값
dirty_writeback_centisecs	60000	60150
dirty_ratio	60	90
Swappiness	60	87
vfs_cache_pressure	100	116

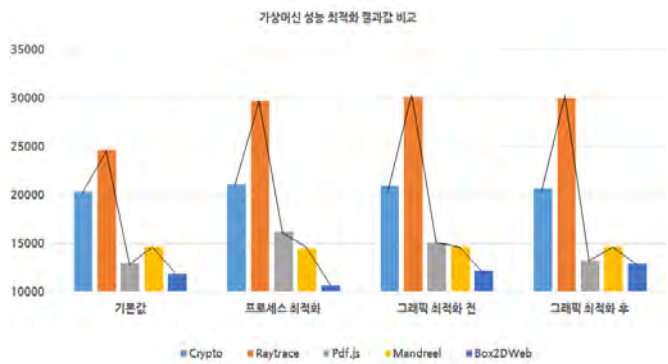
4. 실험 결과 및 평가

<표 6> 크롬 OS Octane 의 가상머신 프로세스 및 그래픽 성능 최적화 결과값

	기본값	프로세스 최적화	그래픽 최적화 전	그래픽 최적화 후
Crypto	20326	21128	20961	20687
Raytrace	24716	29792	30131	30044
Pdf.js	12968	16162	15127	13260
Mandreel	14651	14469	14665	14636
Box2DWeb	11869	10684	12238	12871

<표 7> 크롬 OS Octane 의 ARM 크롬북 프로세스 및 그래픽 성능 최적화 결과값

	기본값	프로세스 최적화	그래픽 최적화 전	그래픽 최적화 후
Crypto	7677	8064	8109	8070
Raytrace	9204	12149	12444	12825
Pdf.js	2961	4308	3830	3855
Mandreel	2570	4680	4626	4372
Box2DWeb	3546	5853	7035	7056



(그림 3) 가상머신 성능 최적화 결과값 비교

(그림 4) ARM 크롬북 성능 최적화 결과값 비교

실험 결과값들을 살펴보면 디스크로 내보내지 않고 캐시의 성능을 일정 부분 높인 경우 프로세스 처리 성능은 높았지만 그래픽 처리 성능은 낮았다. 반대로 디스크로 내보내는 설정 값들을 일정부분 높인 경우 그래픽 성능은 높았지만 프로세스 처리 성능 결과 값이 낮음을 수치상으로 확인할 수 있다.

다만 dirty_writeback_centisecs 만을 적용하였을 경우 3D 연산을 통한 빠른 비트 연산으로 인하여 처리 능력이 더 좋았음을 알 수 있다. 하지만 2D 연산에 있어서는 나머지 3 개의 연산도 같이 적용하였을 경우 처리 능력이 더 좋았음을 확인할 수 있다. 그래서 이 방법은 크롬 OS 에서 그래픽 작업을 할 경우 용도에 따라 알맞은 최적화를 통해 빠른 처리가 가능함을 알 수 있다. 반대로 수 많은 배열 값들을 처리할 경우 프로세스 최적화 기법을 통해 보다 더 빠른 처리가 가능함을 보여준다.

5. 결론 및 향후과제

본 논문에서는 SSD 기반의 크롬 OS 최적화 방법에 대해서 제시했다. 가상메모리를 설정하기 위한 변수인 dirty_writeback_centisecs, dirty_ratio, Swappiness, vfs_cache_pressure 에 변경값을 삽입하여 그래픽이나 프로세스 처리 같은 특수 목적분야에서 적절한 처리가 가능함을 실험을 통해 확인했다. 하지만 현재 변수 값들만으로는 다양한 목적의 처리를 하는데 있어 한계가 있을 뿐만 아니라 SSD 환경에서 완벽하게 데이터를 처리하는데 부족한 부분이 많다. 그래서 향후에는 커널 접근을 통한 최적화 및 프로세스 처리 속도 향상과 관련하여 연구하는 것이 향후 과제이다.

Acknowledgment

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 ICT/SW 창의연구과정지원사업(SW 중심대학)의 연구결과로 수행되었음"(R2215-15-1007)

참고문헌

[1] https://www.chromium.org/chromium-os/how-tos-and-troubleshooting/chromiumos-architecture-porting-guide
[2] Alexis Emperador, Alison N. Norman “Analyzing ChromeOS’s Boot Performance” Texas University, 2013 (학술논문)
[3] 이상진 “효율적인 서버 운영을 위한 Linux OS Tuning 의 관한 연구” 연세대학교, 2009 (석사논문)
[4] 서재우 “리눅스 운영체제에서 효과적인 캐시 메모리 공간 관리에 대한 연구” 고려대학교, 2013 (석사논문)
[5] 김영곤, 김성진, 박찬익 “리눅스의 실시간 성능 향상을 위한 적응적 커널 쓰레드 제어 기법” 포항공과대학교, 2009 (학술논문)
[6] 김용운, 박정수, 김용진 “내장형 리눅스를 위한 시스템 최적화 기술” 한국전자통신연구원 표준연구센터, 2000 (학술논문)
[7] https://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices/samsung-arm-chromebook