다중 사용자 테이블탑 환경에서의 윈도우 데코레이션 방식에 따른 성능 분석

Performance Analysis of Window Decoration Method in Multi-user Tabletop Systems

요 약

기존의 리눅스 시스템은 단일 사용자 컴퓨팅 환경에 개발되어왔다. 하지만 최근 대형 멀티 터치스크린과 같은 다중 입력을 위한 폼팩터 (Form Factor)가 등장하면서 기존의 리눅스 환경과는 다른 새로운 환경이 증가하였다. 다중 사용자 환경에서는 여러 프로세스와 여러 사용자가 상호작용하기 때문에 사용자들에게 낮은 응답시간을 제공하는 것이 중요하다. 모든 사용자의 응답시간을 최소화해 성능을 높여주기 위해서는 테이블 탑 디스플레이와 같은 환경에서 빈번하게 일어날 수 있는 윈도우 데코레이션 이벤트를 처리하는 성능이 중요하다. 이에 본 논문에서는 다중 사용자 환경에서 Wayland와 X Window의 성능을 비교 분석하였다. 실험 결과, X Window가 Wayland에 비해 스핀락에 대한 호출이 많아 X Window가 Wayland 보다 121.39% 만큼 CPU Clock을 더 소모하는 것을 확인하였다.

1. 서 론

최근 테이블 탑 디스플레이 등 다중 입력을 받을 수 있는 폼팩터가 등장하면서 동시에 다중 사용자 컴퓨팅 환경이 증가하고 있다. 이러한 GUI (Graph User Interface) 환경에서 각 윈도우를 관리할 목적으로 만들어진 소프트웨어를 윈도우 매니저라고 한다. 윈도우 매니저는 화면상에 나타나는 여러 가지 창 (Window) 의 크기, 이동, 배열 등을 관리하여 보다 효율적으로 모니터에 출력하게 해준다. 대표적인 윈도우 매니저로는 X Window와 Wayland가 있다 [1].

단일 사용자 환경은 윈도우 매니저 서버에 요청을 보내는 클라이언트의 수가 단일 사용자만 존재하기 때문에 윈도우 매니저가 감당할 수 있을 만큼으로 유지된다. 따라서 윈도우 매니저가 최적화되어있는 단일 사용자 환경에서는 사용자가 느끼는 디스플레이 응답성은 높게 측정된다. 하지만 다중 사용자 환경은 여러 사용자의 여러 프로세스가 서로 상호작용을 해야 하는 복잡한 상황에 놓여있다. 테이블 탑 디스플레이에서 여러 사용자가 수많은 프로그램을 실행하고 한 사용자가 다른 사용자에게 자신이 실행한 프로그램을 넘겨주는 작업, 서로 다른 사용자가 윈도우 (클라이언트)에 대한 이벤트들을 동시에 발생시키는 작업 등 윈도우 매니저가 처리해야 하는 이벤트가 증가하게 된다. 윈도우 매니저가 이러한 이벤트들을 처리하는 것을 윈도우 데코레이션이라고 한다. 이에 본 논문에서는 다중 사용자 환경에서 어떤 윈도우 데코레이션 방식이 더 효율적인지 비교 분석하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 대표적 윈도우 매니저인 X Window 와 Wayland 에 대해 소개하고 3 장에서는 각각 X Window 와 Wayland 환경에서 각각의 윈도우 데코레이션 이벤트들에 대한 벤치마크 테스트를 한 결과를 분석하고 4 장에서는 본 논문의 결론과 향후 계획으로 마무리한다.

2. 배경 지식

X Window 는 X 윈도우 시스템 프로토콜을 기반으로 컴포지터, 서버, 클라이언트로 구성되는 윈도우 시스템이다 [2]. X Window 는 개발이 진행한지 오래 지났기 때문에 예전에 작성되었던 함수들 중에 지금은 쓰지 않은 함수들도 포함되어있다. 이로 인해 코드의 양이 많아 윈도우 시스템 자체가 무겁다. Wayland 는 긴 코드로 인해 무거운 X Window 를 대체하고자 만들어진 윈도우 매니저이다. X Window 와 달리 Wayland 컴포지터는 컴포지터 자체가 서버의 역할을 맡고 있는 구조이다.

Wayland와 X Window가 보이는 가장 큰 차이점은 윈도우 데코레이션 방식이다. 윈도우 데코레이션이란 윈도우 시스템에서 클라이언트의 제목 표시줄이나 경계를 수정하는 일을 일컫는다 [3]. 이러한 윈도우 데코레이션을 수행하는 방식에는 X Window가 사용하는 SSD (Server Side Decoration)와 Wayland가 사용하는 CSD (Client Side Decoration)이 존재한다. X Window가 사용하고 있는 SSD의 경우, 윈도우 데코레이션을 서버에서 진행하는 방식으로, 서버에 부하가 커지는 문제가 있지만, 모든 윈도우의 제목표시줄을 서버에서 그리기 때문에 일관성 있게 관리하기에 좋다. 서버가 클라이언트로부터 종료 이벤트 요청을 받았을 때 클라이언트에 오류가 발생하더라도 종료 요청을 처리할 수 있다. Wayland가 사용하는 방식인 CSD는 서버가 하던 윈도우 데코레이션을 클라이언트가 직접 하는 방식이다. 클라이언트가 서버의 일을 가져오기 때문에 서버를 더 가볍게 사용할 수 있지만, 클라이

표 1 실험 환경

OS	Ubuntu Desktop 16.04
Kernel	Linux version 4.10.0-30-generic
CPU	Intel(R) Core(TM) i5-5257U CPU @ 2.70GHz / 2 Core
RAM	2GB

언트끼리 일관성 있는 제목 표시줄을 가지려는 노력이 필요하다. 또 CSD는 윈도우 프레임을 클라이언트가 그리기 때문에 윈도우 데코레이션 이벤트를 클라이언트가 확인해서 서버에 요청해야 한다. 따라서 클라이언트에 오류가 발생하면 정상적인 이벤트 처리를 할 수 없다.

3. 실험

실험 환경은 표 1과 같은 환경에 X Window와 Wayland를 설치해 실험을 진행하였다. 벤치마크 툴로는 x11perf를 사용하였다. x11perf는 X11 서버 성능 테스트 프로그램이지만 Wayland가 X11에 맞춰진 프로그램도 지원하기 때문에 Wayland의 성능 테스트에도 사용할 수 있다. 다중 사용자 환경에 맞는 테스트를 하기 위해 윈도우 창을 최소 10개에서 최대 250개까지 생성하는 테스트를 진행하였다. x11perf로는 벤치마크 테스트의 실행시간만 알 수 있기 때문에 리눅스 성능 측정 도구인 perf로 벤치마크 테스트 실행 간의 CPU Clock을 측정하였다.

벤치마크 테스트를 위한 x11perf의 옵션은 표 2와 같이 선택하였다. 다중 사용자 환경에서 발생할 가능성이 높은 윈도우 데코레이션 이벤트이기 때문에 이러한 옵션들을 선택하였다. 이에 따라 X window와 Wayland의 차이점인 윈도우 데코레이션 방식에 따른 성능 차이를 확인할 수 있기 때문이다.

실험 결과, Circulate, Create, Resize, Destroy 옵션에서는 비슷한 성능을 보였지만 Popup과 Move 옵션에서는 큰 차이를 보였다. 그림 1과 그림 2는 각각 Popup, Move 옵션으로 실험했을 때 CPU Clock을 정규화한 그래프이다.. CPU Clock이 큰 상위 5개의 함수는 따로 표현하였다. Popup 옵션에서는 __lock_text_start 함수를 X Window가 Wayland 대비 130.82% 더 호출하였고 Move 옵션에서는 X Window가 Wayland 대비 239.09% 더 호출하였다. 기타 다른 함수의 경우, 차지하는 비중이 작고 Wayland와 X Window 사이의 적은 차이를 보였다. __lock_text_start는 스핀락이 걸린 변수를 획득하기 위해서 호출되는 함수이다. X Window가 Wayland에 비해 스핀락에 대한 요청이 많아 CPU Clock이 큰 것으로 분석되었다.

표 2 x11perf 옵션

옵션	설명
Create	새로운 윈도우 창을 생성하는 작업
Destroy	윈도우 창을 종료하는 작업
Popup	윈도우 창을 최소화, 활성화를 반복하는 작업
Move	활성화된 윈도우 창을 움직이는 작업
Resize	활성화된 윈도우 창의 크기를 조절하는 작업
Circulate	활성화되어있는 윈도우 창 중에 밑에 있는 윈도우를 위로 올리는 작업

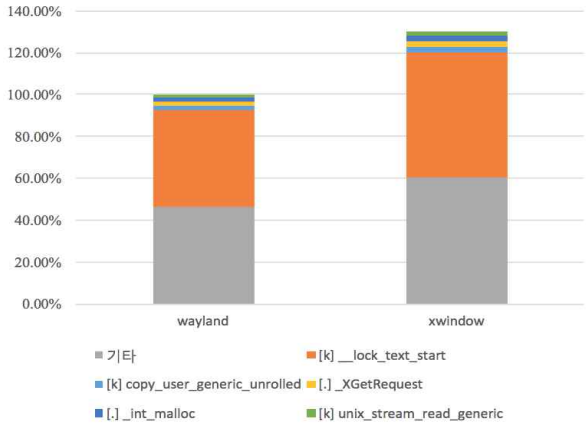

그림 1 Popup 옵션의 함수별 CPU Clock

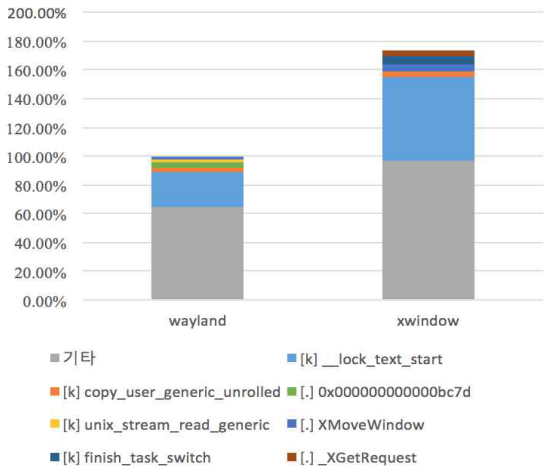

그림 2 Move 옵션의 함수별 CPU Clcok

4. 결론 및 향후 연구

본 논문에서는 윈도우 데코레이션 방식이 다중 사용자 환경에서 미치는 영향에 대해 분석하기 위한 실험을 진행하였다. 실험 결과 Wayland가 X Window에 비해 좋은 성능을 보였다. 향후 KDE (K Desktop Environment)에서 제안한 새로운 윈도우 데코레이션 방식인 DWD (Dynamic Window Decoration)에 대한 분석을 진행할 것이다.

참고 문헌

[1] H. S. Kim, S. J. Noh, and Y. I. Eom, “Comparative Analysis of the Performance of X Window and Wayland Systems,” *Proc. of the KIISE Winter Conference*, pp. 1791-1793, 2016.

[2] Wayland.[Online]. Available: https://wayland.freedesktop.org/ (Downloaded 2017, Mar. 10)

[3] R. Wimmer and F. Hennecke, “Everything is a Window: Utilizing the Window Manager for Multi-Touch Interaction,“ *Proc. of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pp. 1-4, 2010.