졸업작품/논문 제안서

2017 년도 제 1 학기

다중 사용자 환경에서의 Window Decoration 방식에 따른 성능 분석 및 DWD 방식 도입

황인규(2012310466)

GitHub URL : https://github.com/dlsrb6342

2017 년 3 월 23 일

지도교수 : 엄 영 익 서명

1. 과제의 필요성

1.1 Abstraction

테이블 탑 디스플레이 등 다중 사용자가 동시에 사용할 수 있는 기기들이 증가하고 있다. 이러한 디스플레이의 GUI 환경에서 각 윈도우를 관리할 목적으로 만들어진 소프트웨어를 Window Manager라고 한다. 대표적으로 X Window와 Wayland Compositor가 있고 각각 Window Decoration Protocol인 X와 Wayland를 적용한 Window Manager이다. 이러한 Window Manager들은 단일 사용자 환경에서 최적화되어있는 것이고 다중 사용자 환경에서는 수정이 필요하다. 또 Window Decoration의 방식인 CSD(Client Side Decoration), SSD(Server Side Decoration) 두 방식을 어떻게 적용하느냐에 따라 성능이 좌우될 수 있다. CSD 방식을 선호하는 Wayland와 기존의 SSD 방식을 선호하는 X Window 사이에 어떤 방식이 더 효율적인가에 대한 논쟁이 있어왔다. 따라서 이번 논문에서는 다중 사용자 환경에서 Window Manager인 X Window와 Wayland Compositor에서의 CSD 방식과 SSD 방식에서의 성능 차이에 대해 분석하고 최근 KDE에서 발표한 DWD 방식을 어떤 방향으로 적용했을 때 성능 향상이 있을 지에 대해 연구하고자 한다.

1.2 서론

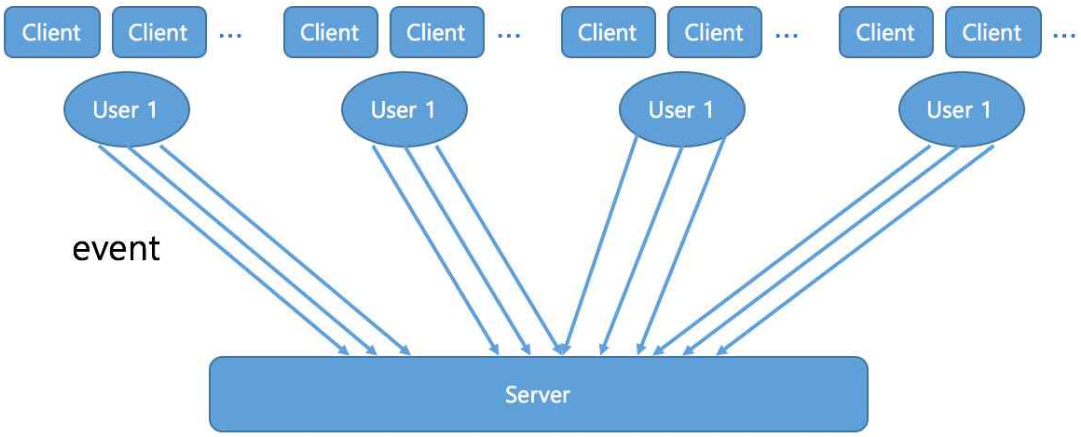

그림 1 다중 사용자 환경에서 발생하는 이벤트

단일 사용자 환경의 경우, 서버에 요청을 하는 클라이언트의 수가 윈도우 매니저가 감당할 수 있을 만큼으로 유지된다. 따라서 이미 윈도우 매니저가 최적화되어있는 단일 사용자 환경에서는 사용자가 느끼는 디스플레이 응답성은 매우 높게 측정되어진다. 하지만 다중 사용자 환경은 여러 사용자의 여러 프로세스가 서로 상호작용을 해야 하는 복잡한 상황에 놓여있다. 예를 들어 테이블 탑 디스플레이에서 여러 사용자가 수많은 프로그램을 실행하고 한 사용자가 다른 사용자에게 자신이 실행한 프로그램을 넘겨준다든지 서로 다른 사용자가 같은 프로그램을 실행하는 것 같은 윈도우 매니저가 처리해야 하는 이벤트가 증가하게 된다.

이러한 다중 사용자 환경에서는 모든 사용자에 대해 대화형 프로세스의 Latency를 줄여 응답성을 높이는 것이 중요하다. 그 일환으로 다중 사용자 환경에서 최적화된 소프트웨어 플랫폼이나 입출력 스케줄러에 대한 연구는 활발히 진행 중에 있다. Window Manager에 대한 다중 사용자 최적화로 단일 스레드로 동작하는 윈도우 매니저를 프로세서마다 스레드를 하나씩 둬서 멀티 스레드로 동작시키는 연구가 있다[1]. 다중 사용자로 인해 너무 많은 이벤트가 쌓여 윈도우 매니저가 병목이 되는 것을 막기 위해 병렬화를 통해 대화형 프로세스의 응답성을 높인 연구이다.

이 논문에서는 다중 사용자 환경에서 기존 CSD(Client Side Decoration)이나 SSD(Server Side Decoration)을 사용했을 때, memory나 CPU 등 리소스를 더 효율적으로 사용하고 있는 방식이 어떤 것인지, 왜 이런 결과가 나왔는지에 대한 분석을 하고 이를 통해 KDE가 제안한 DWD(Dynamic Window Decoration)을 어떻게 적용시키면 다중 사용자 환경에서 모든 사용자에 공정하게 대화형 프로세스의 응답성을 높여줄지 예측해보고자 한다.

2. 선행연구 및 기술현황

2.1 Window Manager

Window Manager란 GUI 환경에서 각 윈도우를 관리할 목적으로 만들어진 소프트웨어를 말한다. Windows나 MacOS는 운영체제에 Window Manager가 포함되어 있지만 Linux에서는 여러 Window Manager 중에 선택하여 사용할 수 있다. Window Manager는 서버와 클라이언트 사이의 규약인 Window Protocol, 클라이언트에 요구에 따라 화면에 출력해주는 Compositor, Window Decoration을 담당하는 Shell, 이 3가지로 이루어져 있다.

2.2 X Window / Wayland Compositor

X Window는 1984년부터 개발되기 시작한 X 윈도우 시스템 프로토콜을 기반으로 하는 클라이언트와 서버 모델의 네트워크 지향 윈도 시스템이다. X Window는 개발이 오래되었기 때문에 예전부터 존재는 하지만 지금은 쓰지 않은 함수들도 포함되어있다. 기본적으로 SSD 방식을 택해서 자칫 서버 무겁게 한다. 프로그램들이 클라이언트가 되어 서버에 요청을 보내고 서버는 해당 요청을 처리하고 그것을 컴포지터에 보내고 화면에 출력되는 과정을 갖는다. Wayland Compositor는 오래 되고 무거운 X Window를 대체하고자 만들어진 프로토콜인 Wayland의 윈도우 매니저이다. 서버를 가볍게 유지하는 것이 이 프로토콜의 목적이기 때문에 서버의 Window Decoration 작업을 뺏어오는 CSD 방식을 선호한다. X Window와 달리 Wayland Compositor는 컴포지터 자체가 서버의 역할을 맡고 있는 구조이다.

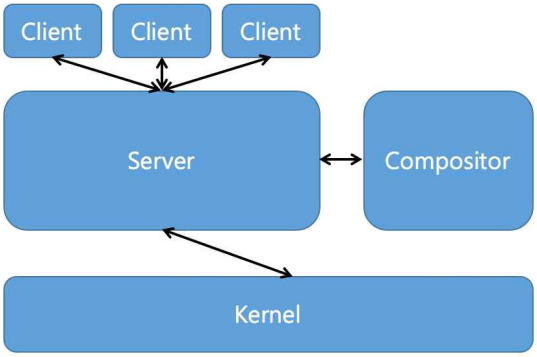

그림 2 X Window의 구성

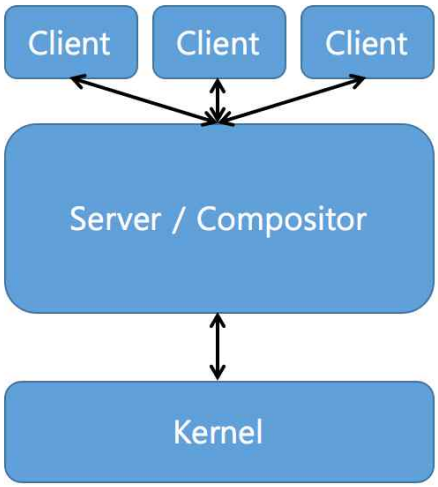

그림 3 Wayland Compositor의 구성

2.3 CSD 방식 / SSD 방식

먼저 Window Decoration이란 Windowing System에서 윈도우의 title bar나 border를 수정하는 일을 일컫는다. 윈도우를 최소화, 최대화, 종료, 사이즈 조절 등의 일들을 말한다. 이러한 Window Decoration을 수행하는 방식에 Server Side Decoration과 Client Side Decoration이 있는 것이다. SSD(Server Side Decoration)의 경우, Window Decoration을 서버 사이드에서 진행하는 방식으로, 서버에 부하가 커지는 문제가 있지만, 모든 윈도우의 title bar를 서버에서 그리기 때문에 똑같이 일관성있게 관리하기에 좋다. CSD(Client Side Decoration)은 서버가 하던 Window Decoration을 클라이언트가 직접하게 하는 방식이다. 클라이언트가 서버의 일을 가져오기 때문에 서버를 더 가볍게 사용할 수 있지만, 클라이언트들끼리 일광성있는 title bar를 가지려는 노력이 필요하다.

2.4 DWD 방식

Dynamic Window Decoration 방식이란 최근에 KDE(K Desktop Environment)에서 발표한 Window Decoration 방식의 하나이다. DWD(Dynamic Window Decoration)는 2.2의 SSD와 CSD의 장점을 취해 만든 방식이다. 클라이언트는 어떤 아이콘을 쓸 것인지 서버에 보내주고 서버는 이 아이콘들을 이용해 Window Decoration을 한다. 이렇게 한다면 클라이언트가 그리는 일을 담당하게 되어 서버의 부하가 떨어지고 서버는 해당 아이콘의 이벤트를 안정적으로 담당할 수 있게 된다.

3. 작품/논문 전체 진행계획 및 구성

표 1 논문 전체 진행계획

월별 / 내용	3	4	5	6	7	8	9	10	11
Github (연구노트) 작성	■	■	■	■	■	■	■	■	■
논문 제안서, 서약서	■								
관련 논문 및 기술 분석		■	■	■					
기존 방식 실험 및 개선 방안 제안					■	■			
논문 중간보고서							■		
논문 최종보고서, 학회발표자료								■	■
논문 발표회									■

이 논문의 전체 진행계획은 위의 표1과 같다. 3월에는 기존 Linux Kernel의 리소스 관리에 대한 이해 및 윈도우 매니저의 작동 방식을 숙지하며 제안서를 작성한다. 4월부터 6월까지는 단일 사용자 환경에 최적화되어있는 윈도우 매니저들의 기술 및 방식을 분석한다. CSD 방식과 SSD 방식의 차이 및 장단점을 확인하고 DWD 방식에 대해 분석하고 각 기술 간의 관계를 확인한다. 윈도우 매니저 관련 논문들을 읽고 다중 사용자 환경에서의 적용 방안에 대해 탐색한다. 7월부터 8월까지 공부한 기술들을 토대로 윈도우 데코레이션 방식에 따른 CPU, 메모리 사용량 등 리소스에 대해 실험하고 다중 사용자 환경에 적합한 방식을 탐구한다. 9월에는 논문 작성 및 중간보고서 작성을 하고 10월부터 11월까지 최종보고서 및 학회발표자료를 준비한다.

4. 기대효과 및 개선방향

윈도우 매니저 X Window, Wayland Compositor의 어떤 Window Decoration이 다중 사용자 환경에 적합한지 알 수 있다. 그 방식을 고려하여 새로운 Window Decoration 방식인 DWD를 다중 사용자 환경에 어떻게 적용시킬 수 있을지 알 수 있을 것이다. 현재 단일 사용자 환경에 적합한 윈도우 매니저를 다중 사용자 환경에 맞춰 개선할 수 있을 것이다.

5. 참고문헌

[1] I. Kim, J. Kim, T. Kim, and Y. I. Eom, "Research on Multithread-based Window Manager for Supporting Multiple Users and Services," *Proc. of the KIISE Korea Computer Congress*, pp. 1291-1293, 2014.

[2] R. Wimmer and F. Hennecke, "Everything is a Window: Utilizing the Window Manager for Multi-Touch Interaction," *Proc. of the EICS*, pp. 1-4, 2010.

[3] 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.

[4] M. Lee, I. Kim, and Y. I. Eom, "Analyses of I/O Schedulers for Multi-user Systems Considering Periodicity of Interactive Processes," *Proc. of the KIISE Winter Conference*, pp. 1278-1280, 2015.

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

[6] x.org. [Online]. Available: https://www.x.org/wiki (downloaded 2016, Nov. 10)