다중 사용자/서비스 지원을 위한 멀티 스레드 기반 윈도우 매니저 연구

Research on Multithread-based Window Manager for Supporting Multiple Users and Services

저자
(Authors)

김인혁, 김정한, 김태형, 엄영익
Inhyeok Kim, Junghan Kim, Taehyoung Kim, Young Ik Eom

출처
(Source)

한국정보과학회 학술발표논문집 , 2014.6, 1291-1293 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE02444357

APA Style

김인혁, 김정한, 김태형, 엄영익 (2014). 다중 사용자/서비스 지원을 위한 멀티 스레드 기반 윈도우 매니저 연구. 한국정보과학회 학술발표논문집, 1291-1293.

다중 사용자/서비스 지원을 위한 멀티 스레드 기반 윈도우 매니저 연구

김인혁, 김정한, 김태형, 엄영익
성균관대학교 정보통신대학
{kkojiband, gtgkjh, kim15m, yieom}@skku.edu

Research on Multithread-based Window Manager for Supporting Multiple Users and Services

Inhyeok Kim, Junghan Kim, Taehyoung Kim, Young Ik Eom
College of Information & Communication Engineering, Sungkyunkwan University

요 약

대형 멀티터치 스크린과 투명 디스플레이의 대중화는 테이블탑과 같은 새로운 컴퓨팅 환경을 만들어내고 있지만, 이를 효과적으로 지원하는 개방성과 사용성을 지닌 플랫폼 기술에 대한 연구는 아직 미흡하다. 특히, 기존의 개인 사용자 중심에서 다중 사용자 중심으로 컴퓨팅 환경이 변화함에 따라 윈도우 매니저와 툴킷 엔진 계층에서도 기존에는 관심을 두지 않았던 병렬화와 같은 최적화 기법이 필요하게 되었다. 이에 본 논문에서는 기존의 싱글 스레드 기반의 윈도우 매니저를 멀티 스레드 환경에서 동작시킬 수 있는 다양한 동기화 및 작업 분배 기법들을 제안하였고, 이를 간단한 실험을 통해 검증하였다. 향후에는 싱글 스레드 기반의 다중 사용자와 서비스를 지원하는 윈도우 매니저에 본 논문에서 제안한 기법들을 실제로 적용하여 연구를 진행해 나갈 계획이다.

1. 서 론

대형 멀티터치 스크린과 투명 디스플레이 시장의 성장은 스마트폰이 우리의 삶을 변화시킨 것처럼 우리가 생활하는 곳곳에 새로운 가능성을 제시하고 있다. 바로 카페/사무실의 테이블이나 편의점의 냉장고 앞유리가 새로운 컴퓨팅 환경이 되는 시대가 온 것이다. 하지만 이러한 하드웨어를 제대로 활용할 수 있는 소프트웨어 기술의 발전 속도는 여전히 느리고, 더군다나 기존의 데스크탑/스마트폰 환경에서 다루지 않던 다중 사용자/서비스 지원은 아직 많은 연구가 필요하다[1, 2]. 특히, 디스플레이 장치의 크기가 커지고, 동시에 장치에 접근하는 사용자(혹은 입력)의 수가 많아질수록 기존의 싱글 스레드 기반의 윈도우 매니저(혹은 툴킷 엔진)는 심각한 병목 현상을 발생시킬 수 밖에 없다.

그림 1은 이러한 병목 현상을 간단한 실험을 통해 보여주고 있다. 해당 실험은 멀티코어(8개) 환경에서 윈도우 매니저에 반복적으로 이벤트를 전달하는 스트레스 테스트를 통해 윈도우 매니저가 얼마나 비효율적으로 동작하는지를 검증하는 것이다. 그림 1을 보면 윈도우 매니저가 동작하는 코어(CPU0)은 100%의 사용률을 보이지만, 나머지 7개의 코어는 전혀 사용되고 있지 않다. 이유는 윈도우 매니저에 과부하가 걸려도 싱글 스레드 방식으로 동작하는 기존의 윈도우 매니저는 부하를 전혀 분산할 수 없기 때문이다. 이는 현재까지 제한된 사용자(혹은 입력)만을 지원하는 컴퓨팅 환경에서는 큰 문제가 되지 않았지만, 향후 대형 멀티터치 스크린을 활용하는 다양한 컴퓨팅 환경에서는 여러 사용자에 의해 실행되는 윈도우의 수가 증가하고, 동시에 처리해야할 입력 이벤트의 양이 많아짐에 따라 심각한 문제를 발생시킬 수 있다.

이에 본 논문에서는 기존의 싱글 스레드 환경에서만 동작하는 윈도우 매니저의 한계점을 극복할 수 있도록 멀티 스레드 환경에서 동작 가능한 새로운 형태의 윈도우 매니저를 제안하였고, 간단한 윈도우 프로토콜을 처리할 수 있는 윈도우 매니저를 구현하여 이를 실험을 통해 검증하였다.

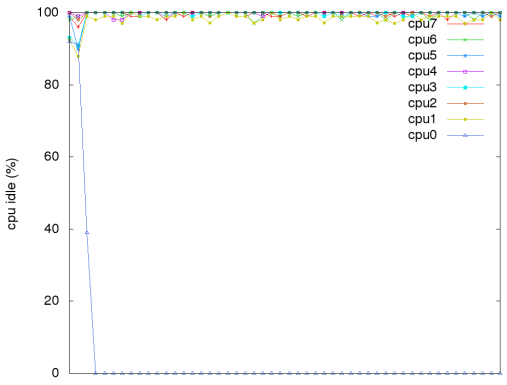

그림 2 멀티코어 환경에서 윈도우 매니저 스트레스 테스트

이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업지원을 받아 수행된 것임 (NRF-2013R1A1A2012790)

2. 관련 연구

아직 윈도우 매니저 계층에서 다중 사용자를 지원하는 연구는 초기 단계여서 멀티 스레드를 적용한 사례가 없기 때문에 툴킷 계층에서 유사한 연구를 수행한 사례를 몇 가지 소개하도록 하겠다. PyMT[3]는 별도의 스레드에서 TUIO[4] 이벤트를 받아 큐에 저장하고, 메인 스레드에서 이를 받아서 나머지 모든 작업들을 처리한다. 이는 단순히 입력 장치 이벤트를 받아들이는 부분만 별도의 스레드로 분리한 것이기 때문에 실질적으로 동시에 접근하는 사용자의 수나 서비스의 수가 많아지는 문제를 해결하긴 힘들다. 그리고 DiamondSpin[5]은 사용자 시나리오를 수행하는 입력 이벤트를 위한 별도의 스레드와 레이어를 생성하고, 컴포지터는 모든 레이어를 합쳐서 결과 화면을 생성하게 된다. 이러한 접근 방식은 여러 사용자가 독립적인 이벤트를 발생시킬 때 개별 이벤트를 처리하는 과정에서 발생하는 병목 현상을 줄일 수는 있지만, 모든 레이어를 합치는 과정에서 발생하는 병목 현상은 여전히 남아있게 된다. 또한 여러 위젯 간의 협업에서는 추가적인 동기화가 필요하게 된다.

이처럼 기존 연구들은 박물관/전시회 같은 특정 공간에서 사용되는 제한된 서비스를 목적으로 진행되어 왔기 때문에 다양한 서비스를 동시에 제공하는 것을 목적으로 하는 윈도우 매니저에 그대로 적용하기는 미흡하다.

3. 멀티 스레드 기반 윈도우 매니저 설계

앞에서 설명한 것처럼 일반적인 윈도우 매니저는 윈도우 혹은 다양한 입출력 장치로부터 들어오는 모든 이벤트를 하나의 스레드에서 순차적으로 처리한다. 본 논문에서는 이를 해결하기 위해 모든 이벤트들을 미리 생성해둔 스레드풀을 이용하여 각각 처리하도록 하였다. 그리고 본 장에서는 이를 위해 어떠한 동기화 정책이 필요한지, 입력 장치 관리와 컴포지팅에서는 어떠한 점을 고려해야 하는지를 상세히 설명하고자 한다.

3.1. 동기화 정책

싱글 스레드 환경으로 인한 가장 큰 장점은 윈도우 매니저 내에서 관리하는 수많은 자료구조에 대한 동기화를 하지 않아도 된다는 것이다. 하지만 리눅스 커널이 멀티코어 환경을 제대로 지원하기 위해 선점형 커널로 바뀐 것처럼 윈도우 매니저도 멀티 스레드 환경을 효과적으로 지원하기 위해서는 적절한 동기화 정책이 필요하다. 현재 본 논문에서 제안하는 윈도우 매니저의 기본적인 동기화 정책은 윈도우 그룹 단위의 동기화이다. 이는 하나의 윈도우 그룹이 기본적으로 한 명의 사용자에 의해 실행/사용되는 윈도우들이므로 사용자 시나리오 관점에서 연관성이 가장 높기 때문이다. 예를 들어 사용자가 원하는 위치로 모든 윈도우를 재정렬하는 경우, 해당 윈도우 그룹에 대한 뮤텍스만 획득하면 한번에 모든 처리가 가능해진다.

3.2. 입력 장치 관리

본 논문에서 고려하고 있는 가장 중요한 입력 장치는 멀티터치이다. 스마트폰에서 사용되는 멀티터치는 개인 사용자를 대상으로 하기 때문에 모든 탭들이 연관되어 있지만, 대형 스크린에서 사용되는 멀티터치는 여러 사용자에 의해 탭이 생성되므로 기본적으로 탭 간의 연관성이 없다고 본다. 이에 본 논문에서는 개별 탭들이 별도의 사용자 시나리오를 처리한다는 가정 하에 TUIO를 통해 들어온 탭들을 개별적으로 큐에 할당한다. 만약 기존의 키보드, 마우스처럼 TUIO 전체를 하나의 큐에 할당할 경우, 다수의 사용자에 의해 생성된 다수의 탭이 하나의 스레드에서 순차적으로 처리되기 때문에 반응성이 떨어질 가능성이 높아진다.

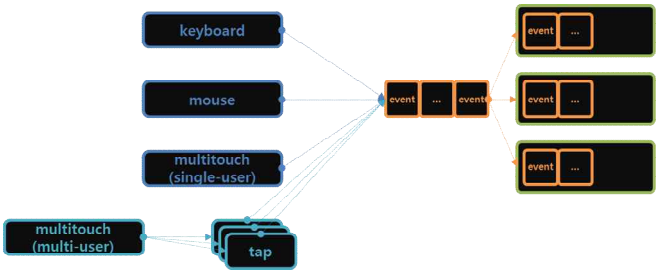

그림 3 입력 장치 이벤트 분배 방식

3.3. 컴포지팅 기법

윈도우 매니저의 가장 중요한 역할은 모든 윈도우를 하나의 화면으로 합쳐주는 컴포지팅이다. 이로 인해 윈도우 매니저는 화면 갱신 비율에 따라 매초 20회 이상 모든 윈도우의 속성과 버퍼에 접근해야 한다. 이는 윈도우의 속성에 접근하는 모든 스레드와 컴포지팅 스레드 간의 과도한 동기화를 유발한다. 본 논문에서는 이를 해결하기 위해 RCU(READ-COPY-UPDATE) 기반 컴포지팅 기법을 제안하고자 한다[6]. 그림 3은 RCU 기법을 이용한 기본적인 컴포지팅 방식을 나타내고 있다. 우선 컴포지터는 윈도우별로 RCU를 위한 큐를 가지고 있다. 그리고 윈도우는 화면이 갱신될 때마다 해당 큐에 새로운 버퍼를 추가하고, 컴포지터는 윈도우의 동작과 상관없이 매번 프레임을 갱신할 때마다 모든 큐의 최근 복사본을 이용하여 화면을 갱신하게 된다. 이럴 경우 RCU는 READ-SIDE에서는 WAITING-FREE를 보장하기 때문에 컴포지터는 윈도우의 수와 상관없이 빠르게 결과 화면을 만들어낼 수 있다.

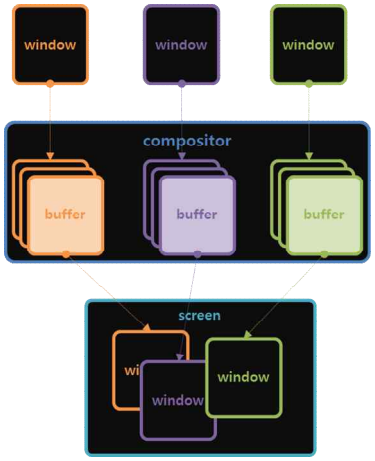

그림 4 RCU 기반 컴포지팅

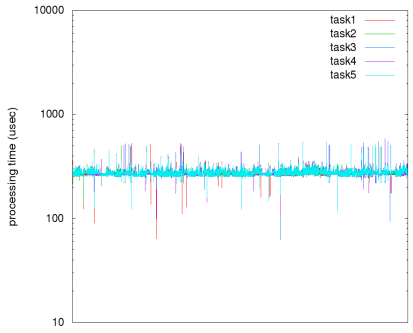

그림 5 싱글 스레드 환경에서 요청별 응답시간

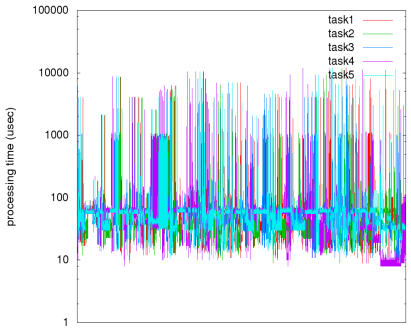

그림 6 멀티 스레드 환경에서 요청별 응답시간

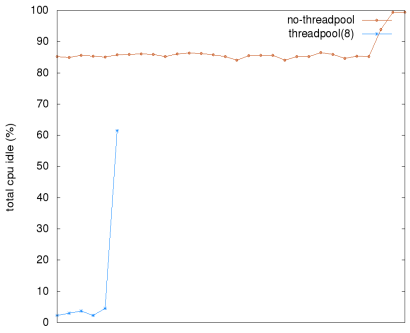

그림 7 CPU 사용률 싱글 스레드 vs 멀티 스레드

4. 실 험

본 장에서는 X 프로토콜을 대체하는 차세대 윈도우 프로토콜인 WAYLAND[7] 프로토콜을 지원하고 윈도우 클라이언트와 입력 장치로부터 들어오는 이벤트를 스레드 풀을 이용하여 처리하는 간단한 윈도우 서버를 구현하여 기본적인 성능 평가를 실시하였다.

실험은 8개의 코어를 가지는 데스크탑 환경에서 수행되었고, 5개의 윈도우 클라이언트가 각각 10000번씩 WAYLAND 프로토콜을 서버에게 요청하고 응답을 받기까지 걸리는 시간을 측정하였다. 싱글 스레드 환경은 기존의 윈도우 매니저와 동일하게 스레드풀 없이 하나의 스레드로 전체가 동작하도록 구현하였고, 멀티 스레드 환경은 스레드풀을 이용하여 작업을 분배할 수 있도록 구현하였다. 멀티 스레드 환경에서의 스레드풀 최대값은 기본적으로 코어의 수와 동일하게 설정하였고, 특정 코어에 고정(affinity)시키지는 않았다. 그리고 소켓과 장치 파일로부터 들어오는 모든 이벤트는 하나의 스레드에서 EPOLL을 이용하여 멀티플렉싱하고, 실제 작업 내용을 스레드풀에 전달한다.

우선 그림 4와 그림 5는 싱글 스레드와 멀티 스레드 환경에서 각각 윈도우 클라이언트가 요청에 대한 응답을 받는데 걸린 시간을 나타내고 있다. 그림 4에 비해 그림 5가 응답 시간의 변동 폭이 큰 이유는 현재 윈도우 서버에서 요청을 처리하는 시간에 비해 요청을 처리할 스레드를 할당받고 스케줄링되기까지의 시간이 더 크기 때문이다. 그리고 그림 6은 싱글 스레드와 멀티 스레드 환경에서 전체 작업을 처리하는데 걸린 시간과 전체 CPU 사용률을 나타내고 있다. 이를 보면 앞에서 얘기한 것처럼 싱글 스레드 환경은 8개의 코어 중 1개만을 사용하기 때문에 CPU 사용률이 20% 가 되지 않지만, 멀티 스레드 환경은 전체 CPU를 골고루 사용하기 때문에 CPU 사용률이 거의 100%가 되는 것을 볼 수 있다. 이러한 차이로 인해 싱글 스레드 환경에서 30초 정도 걸리는 작업을 멀티 스레드 환경에서는 4초 정도의 시간으로 처리할 수 있게 된다.

5. 결 론

본 논문에서는 기존의 데스크탑/스마트폰 환경이 아닌 대형 멀티터치 스크린이 활용될 수 있는 다양한 컴퓨팅 환경을 목표로 다중 사용자/서비스를 지원할 때 발생하는 윈도우 매니저(혹은 툴킷 엔진) 계층에서의 병목 현상을 해결할 수 있는 방법으로 멀티 스레드 환경을 기반으로 한 병렬화를 제안하고 있다. 이를 위해 윈도우 매니저의 가장 중요한 역할인 윈도우 관리, 입력 장치 이벤트 처리 그리고 컴포지팅에서 효과적인 스레드 분배 기법 및 동기화 기법을 연구하였고, 간단한 실험을 통해 멀티 스레드 환경에서 얻을 수 있는 성능적인 이점을 평가하였다.

향후에는 기존의 싱글 스레드 기반으로 개발 중이던 다중 사용자/서비스 지원 윈도우 매니저에 본 논문에서 제안한 기법들을 실제로 적용하여 스레드풀 관리, 작업 분배 정책 그리고 캐시 등을 고려한 코어 할당 정책 등 세부적인 최적화 기법에 대한 연구도 함께 진행해나갈 계획이다.

참 고 문 헌

[1] N. A. Hamdan, S. Voelker and J. Borchers, “Conceptual Framework for Surface Manager on Interactive Tabletops,“ In Ext. Abstracts CHI, 1527-1532, 2013.

[2] R. Wimmer and F. Hennecke, “Everything is a Window: Utilizing the Window Manager for Multi-Touch Interaction,“ In Proc. EICS 2010.

[3] T. E Hansen, J. P. Hourcade, M. Virbel, S. Patali and T. Serra, “PyMT: A Post-WIMP Multi-Touch User Interface Toolkit,“ In Proc. ITS, 17-24, 2009.

[4] tuio, http://www.tuio.org, 15 April, 2014.

[5] C. Shen, F. D. Vernier, C. Forlines, M. Ringel, “DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction,“ In Proc. CHI, 167-174, 2004.

[6] M. Desnoyers, P. E. McKenney, A. S. Stern, M. R. Dagenais, and J. Walpole, “User-Level Implementations of Read-Copy Update,“ IEEE Transactions on Parallel and Distributed Systems, Vol. 23, No. 2, 2012.

[7] wayland, *http://wayland.freedesktop.org*.