ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 21, No. 5, pp. 379-386, 2015. 5
http://dx.doi.org/10.5626/KTCP.2015.21.5.379

# 스마트폰 응용 프로그램의 사용자 경험 향상을 위한 사용자 중심 반응 시간 분석 도구

## (A User-Centric Response Time Analyzer for Improving User Experience of Android Applications)

송 욱 [†] 성 노 섭 [††] 김 지 홍 [†††]
(Wook Song) (Nosub Sung) (Jihong Kim)

**요 약** 본 논문에서는 스마트폰 사용자 중심의 반응 완료 시간에 대한 동적 분석을 활용하여 사용자가 실제 인지하는 성능 중심의 새로운 최적화 프레임워크를 제안한다. 이를 위하여 먼저 스마트폰 응용 프로그램에서 사용자가 실제 인지하는 성능에 대한 평가 지표로써 사용자 중심 반응 시간을 정의한다. 또한, 이러한 사용자 중심 반응 시간의 동적 탐색에 기반하여 사용자가 인지할 수 있는 성능 병목 지점을 최적화의 힌트로써 개발자에게 제공하는 사용자 중심 반응 시간 분석 도구의 설계와 개발에 대하여 소개한다. 제안한 사용자 중심 반응 시간 분석 도구를 갤럭시 넥서스 스마트폰에 구현하여 그 정확도와 계산 부하를 평가한 결과, 전체 반응 시간의 1% 미만의 계산 부하로 카메라를 이용하여 측정한 결과 대비 92%의 정확도를 보였다. 제안한 도구의 효율성 평가를 위하여 소스 코드가 공개되어 있는 안드로이드 응용 프로그램의 성능 개선에 제안한 도구를 활용하여 최대 16.4%의 성능 향상을 달성하였다.

**키워드:** 스마트폰, 성능 측정, 사용자 중심 최적화, 스마트폰 응용, 개발자 도구

***Abstract*** We propose a novel user-perceived performance optimization framework for the Android platform that takes advantage of the user-centric response time analysis. To this end, we propose a new definition of response time, which we call the user-centric response time, as a metric for the quality of user-perceived performance of the smartphone application. In this paper, we describe the design and implementation of an on-line user-centric response time analyzer for Android-based smartphones, which provides smartphone application developers with valuable insight for user-perceived performance optimization. We implemented the user-centric response time analyzer on the Android platform, version 4.0.4 (ICS) running on a Galaxy Nexus smartphone. From our experimental results, the proposed user-centric response time analyzer accurately estimates user-centric response times with an accuracy of 92.0% compared to manually measured times with less than 1% performance penalty. In order to evaluate the efficiency of the proposed framework, we were able to reduce the user-centric response time of the target application by up to 16.4% based on the evaluation results by the proposed framework.

**Keywords:** smartphone, performance evaluation metric, user-centric system optimization, android

· 이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터 연구소에 감사드립니다. 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구입니다(No. 2010-0020724).
· 이 논문은 2014 한국컴퓨터종합학술대회에서 '모바일 응용의 사용자 중심 반응 시간 분석 도구'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 컴퓨터공학부
wooksong@davinci.snu.ac.kr
†† 비 회 원 : 서울대학교 컴퓨터공학부
nssung@davinci.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수(Seoul National Univ.)
jihong@davinci.snu.ac.kr
(Corresponding author임)

논문접수 : 2014년 9월 15일
(Received 15 September 2014)
논문수정 : 2015년 1월 29일
(Revised 29 January 2015)
심사완료 : 2015년 2월 16일
(Accepted 16 February 2015)


정보과학회 컴퓨팅의 실제 논문지 제21권 제5호(2015. 5)

## 1. 서 론

스마트폰은 상호작용 기반의 모바일 장치로서 대부분의 사용 시나리오가 사용자의 입력과 그에 대한 기기의 반응으로 이루어져 있다. 또한 각각의 상호작용 과정에서 사용자는 자신의 입력에 대한 반응이 즉각 이루어질 것으로 기대하고 있기 때문에 사용자가 인지하는 입력인가 시점부터 이에 대한 반응 완료 시점까지의 시간은 사용자 경험을 결정하는 매우 중요한 성능 평가 지표이다. 즉, 사용자의 연속된 입력에 대한 스마트폰 응용 프로그램의 반응 시간 각각에 의해 사용자 경험이 결정되므로, 스마트폰 응용에서 사용자가 실제로 느끼는 성능의 향상을 위해서는 이러한 사용자 중심의 반응 시간에 대한 분석이 선행되어야 한다.

한편, 전통적인 PC 환경에서의 반응 시간은 일반적으로 해당 작업의 시작 시점으로부터 그 작업이 명시적으로 완료되는 시점까지의 시간으로 정의된다. 하지만 스마트폰 환경에서는 응용이 제공하는 UI(User Interface) 중심으로 상호작용이 이루어지기 때문에 자신의 입력으로 인한 UI의 갱신이 완료되는 시점을 반응 완료 시점으로 판단하는 경향이 강하다. 좀 더 시스템적인 측면에서 접근하면, 화면에 표시되는 UI의 갱신에 필요한 작업들이 끝나면 이러한 작업들 이외의 다른 백그라운드 작업들이 현재 수행 중이라고 하더라도 이와 무관하게 사용자는 자신의 입력에 대한 반응이 완료되었다고 느낀다는 것이다. 이러한 이유로 전통적인 PC 환경에서의 작업 완료 중심의 반응 시간(Computation-Centric Response Time)으로는 사용자가 주관적으로 느끼는 성능을 평가하기에 적합한 평가 지표가 아니며, UI의 갱신을 기준으로 한 새로운 반응 시간에 대한 정의가 필요하다.

본 연구에서와 같이 사용자와의 밀접한 상호작용에 기반하여 동작하는 응용 프로그램에서 사용자의 인지 중심으로 성능을 평가하고자 하는 연구는 계속하여 진행되어 왔다. Time-to-Interact (TTI) [1]는 본 연구에서 제안한 사용자 중심 반응 시간과 유사한 개념으로써 사용자에게 있어 중요한 컨텐츠가 사용자에게 표시되고, 또한 가용하게 되기까지의 대기 시간을 나타내며, 이는 페이스북이 몇 년 전부터 자신들의 웹사이트 성능을 평가하기 위해 사용한 상위 수준 평가 지표이다. TTI가 웹사이트의 성능을 평가하기 위한 노력에 일환이었다면, AppInsight [2]는 스마트폰 대상응용 프로그램에서 사용자가 실제 인지하는 성능을 중심으로 Critical Path를 탐색하고 이를 활용한 최적화 사례를 제시하였다는 것에서 우리 연구와 직접적인 연관성을 찾을 수 있다. 하지만, AppInsight가 대상 응용 프로그램의 코드를 직접 수정하는 방식을 취한 반면에, 우리가 제안한 기법은 스마트폰 운영체제 또는 프레임워크 수준에서 동적 탐색을 진행한다는 것이 큰 차이점이며, 이러한 차이로 인해 AppInsight보다 다양한 화면 갱신 시나리오를 지원할 수 있다는 것이 본 연구의 장점이다. 본 논문의 구성은 다음과 같다. 2장에서는 사용자 중심의 반응 시간을 소개하고 기존의 작업 중심의 반응 시간과의 차이를 기술한다. 3장에서는 본 논문에서 제안한 사용자 중심 반응 시간 분석 도구를 소개하고 구현을 위한 설계와 동작 원리에 대해 설명한다. 4장에서는 제안한 도구의 정확도와 계산 부하를 평가한 후, 5장에서 사용자 중심 반응 시간 분석 도구를 활용한 사례 연구를 기술한다. 마지막으로 6장에서는 결론을 맺는다.

## 2. 사용자 중심의 반응 시간

본 장에서는 먼저 앞서 언급하였던 작업 중심의 반응 시간과 사용자 중심의 반응 시간에 대하여 각각 정의한다. 또한, 각각의 정의에 맞추어 몇 가지 스마트폰 응용의 특정 상호작용에 대한 반응시간을 측정한 후, 이를 비교함으로써 스마트폰 환경에서는 새로운 성능 평가 기준, 즉 사용자 중심의 반응 시간을 활용하는 것이 필요함을 밝힌다.

앞서 언급한 것과 같이 스마트폰 응용과 사용자 사이의 상호작용은 스마트폰 화면 터치, 키 누름 등과 같은 사용자 입력과 이에 따른 스마트폰 응용의 반응으로 생각할 수 있다. 이러한 환경에서 작업 중심의 반응 시간은 새로운 사용자 입력이 인가되는 시점부터 해당 입력으로 인해 생성되어 수행되는 모든 작업의 완료 시점까지의 시간으로 정의한다. 한편, 스마트폰 응용의 경우에는 사용자의 입력을 감지하거나 애니메이션 등을 표현하기 위함 등의 이유로 그 시작과 끝이 명확하지 않은 작업들이 존재하는데, 이러한 경우에는 위에서 정의한 작업 중심의 반응 시간을 적용하기 어렵다. 따라서, 본 논문에서는 작업 중심의 반응 시간은 위와 같이 정의하

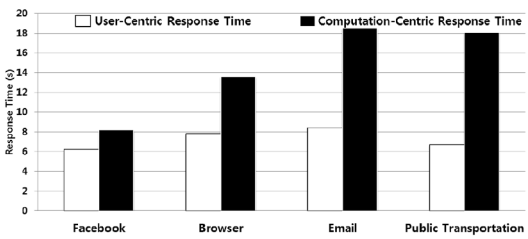


그림 1 4가지 안드로이드 응용에 대한 작업 중심 반응 시간과 사용자 중심 반응 시간의 비교

Fig. 1 Differences between computation-centric response times and user-centric response times

되, 실제 측정은 새로운 사용자의 입력이 인가되는 시점부터 해당 입력으로 인해 호출되는 메소드의 수가 더 이상 증가하지 않는 시점까지로 하였다.

사용자 중심의 반응 시간은 여러 스마트폰 응용 개발 가이드[3]에서 사용자가 실제로 느끼는 성능과 가장 유사할 것으로 추정하여 권고하는 기준을 따른다. 즉, 사용자의 입력이 인가되는 시점부터 이로 인해 발생하는 UI의 갱신이 더 이상 이루어 지지 않는 시점까지를 사용자 중심 반응 시간으로 정의한다[2]. 작업 중심 반응 시간과의 비교를 위해서는 일반적인 안드로이드 응용의 사용자 중심 반응 시간 또한 측정하는 것이 필요하다. 본 논문에서는 먼저 운영체제 수준의 프레임 버퍼를 주기적으로 샘플링 한 후, 이를 이전 프레임과 계속하여 비교하는 방법을 사용함으로써 더 이상 UI에 변화가 없는 시점을 찾고자 하였다. 그림 1은 4가지의 안드로이드 응용(Facebook, Web Browser, Email, Public Transportation)에 대한 작업 중심 반응 시간과 사용자 중심 반응 시간의 비교를 나타낸 것이다. 작업 중심 반응 시간은 모든 경우에서 사용자 중심 반응 시간보다 길게 측정되었으며, Public Transportation 응용의 경우에 그 차이가 최대로 작업 중심 반응 시간이 약 11.4초 정도 긴 것으로 나타났다. 이러한 결과는 UI 갱신 이후에도 상당한 기간 동안 백그라운드 작업이 수행되는 것이 그 이유인데 실제로 Web Browser 응용과 Email 응용을 분석한 결과, UI의 갱신이 완료된 이후에도 웹 캐시 및 방문한 웹 사이트 목록 갱신, 첨부 파일의 백그라운드 다운로드 등으로 인해 계속하여 작업이 이루어졌음을 확인할 수 있었다[4]. UI의 갱신 완료 시점이 사용자가 실제로 느끼는 반응 완료 시점임을 감안하면, 이러한 결과를 통하여 다음과 같은 사실의 확인이 가능하다.

첫째, 기존의 작업 완료 중심의 반응 시간으로는 사용자 관점에서 스마트폰 응용의 성능을 올바르게 측정하기 어렵다. 둘째, 사용자가 실제 느끼는 성능을 최적화하기 위해서는 작업 중심의 반응 시간이 아닌 사용자 중심의 반응 시간을 측정하고, 이를 줄이려는 노력이 필요하다. 따라서, 본 논문에서는 스마트폰 응용을 위한 새로운 성능 평가 지표로서 사용자 중심의 반응 시간을 제안하고, 이에 대한 개발자 수준의 분석이 가능토록 하는 도구인 사용자 중심 반응 시간 분석 도구를 소개한다.

## 3. 사용자 중심의 반응 시간 분석 도구

### 3.1 기본 개념

본 논문에서 개발의 대상으로 선정한 스마트폰 운영체제인 안드로이드의 경우, 사용자의 입력과 화면의 갱신 모두가 메인 쓰레드에서 처리된다. 메인 쓰레드에서의 UI 갱신은 효율성을 위해 무효(Invalidate)-갱신(Update) 메커니즘을 사용한다. 즉, 개발자 수준에서 UI 컴포넌트에 대한 갱신을 요청하면, 안드로이드 프레임워크는 해당 UI 컴포넌트를 무효화 처리한 후, 주기적으로 이러한 무효화 된 UI 컴포넌트들에 대한 갱신을 한꺼번에 처리하는 방식으로 동작한다. 또한, 네트워크나 저장 장치에 대한 I/O와 같이 어느 정도의 시간이 소요되는 작업을 메인 쓰레드에서 수행할 경우에는 메인 쓰레드가 담당하는 사용자 입력 처리 및 UI 갱신 작업을 지연시킬 가능성이 존재하므로 이러한 작업은 반드시 작업 쓰레드를 이용하여 처리할 것을 개발자 가이드 수준에서 권고한다. 이러한 상황에서 사용자의 입력과 이로 인하여 발생한 모든 UI의 갱신까지는 다수의 UI 컴포넌트와 작업 쓰레드들로 인해 다양한 시나리오가 존재한다. 따라서, 정확한 사용자 중심의 반응 시간 측정 및 분석을 위해서는 이런 다양한 시나리오와 무관하게 사용자 입력으로부터 발생한 UI의 갱신을 추적할 수 있어야 한다.

그림 2는 안드로이드 응용에서 발생 가능한 UI 갱신 시나리오들을 나타낸 것으로 UI 컴포넌트에 대한 무효화 요청을 UI_invalidate, 갱신 처리를 UI_update로 표기하였다. 가장 기본적인 시나리오의 형태는 (a)이며 이는 메인 쓰레드에서 UI 컴포넌트에 대한 무효화 요청과 갱신 처리가 모두 일어나는 상황을 나타낸다. 사용자의 입력으로 인해 호출된 콜백 메소드에 의해 UI_invalidate 1, UI_invalidate 2가 발생하였고, 작업 쓰레드의 개입이 없었으므로 얼마 간의 시간이 지난 후, 해당 요청은 UI_update 1, UI_update 2로 처리 완료된다. 이러한 경우에는 각 UI 갱신 요청마다 해당 UI 컴포넌트(예: View)에 대한 식별자(예: View ID)를 저장하고 UI 갱신 시점에 미리 저장해 두었던 식별자들이 모두 처리되었는지를 확인함으로써 사용자 입력으로 인해 발생한 모든 UI 갱신이 완료되었는지 확인할 수 있다. (b)의 경우는 (a) 시나리오에 작업 쓰레드 하나가 개입된 형태이다. 이러한 시나리오에서도 모든 UI 컴포넌트 무효화 요청은 사용자 입력으로 인해 발생하였다고 가정한다. 작업 쓰레드는 Async 1 시점에서부터 개입을 시작하며 위임 받은 작업을 완료하는 시점에 UI 무효화 요청 invalidate 3을 발생시키고 있다. 이러한 시나리오를 처리하기 위해, 사용자 입력으로 인해 메인 쓰레드로부터 작업을 위임 받는 모든 작업 쓰레드를 추적하는 것이 필요하다. 특히, 안드로이드의 개발자 API의 경우, 작업 쓰레드를 사용하기 위한 자바 수준의 쓰레드 생성부터 프레임워크에서 관리하고 있는 쓰레드 풀을 활용하기 위한 것까지 다양한 방식을 지원하고 있으므로 이에 대한 고려가 필요하다. 지금까지의 (a)와 (b) 시나리오는 모든 UI 컴포넌트 무효화 요청이 사용자 입력으로부터

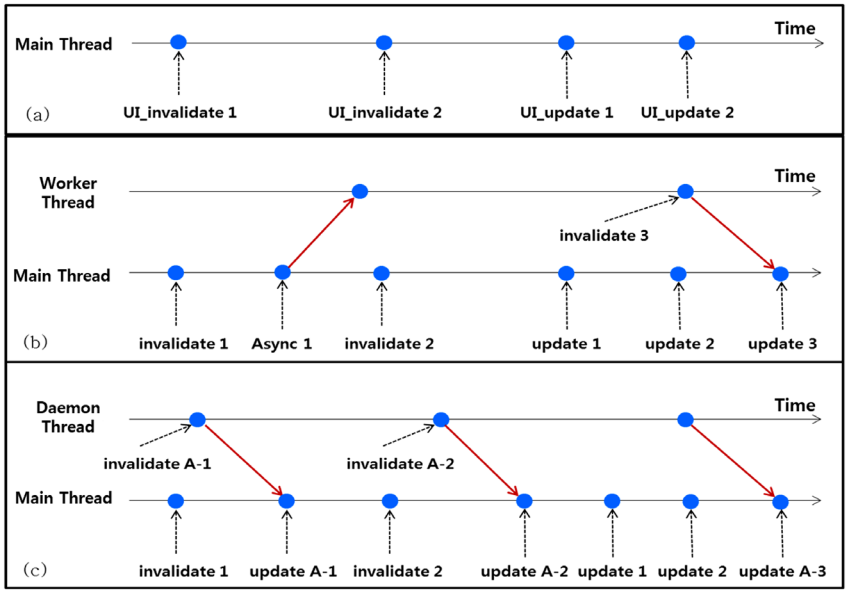


그림 2 다양한 화면 갱신 시나리오의 예제

Fig. 2 Examples of how UI update requests are handled in the Android framework

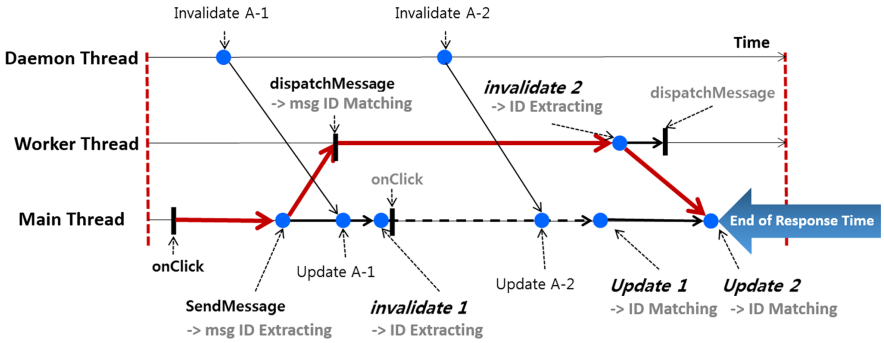


그림 3 사용자 중심 반응 시간 판별의 예시

Fig. 3 An example of identifying the user-centric response time

발생한 것으로 가정하였으나, (c)의 경우에는 invalidate A-1, invalidate A-2와 같이 사용자의 입력 처리와 무관한 쓰레드(데몬 쓰레드)에서 무효화 요청이 발생한 상황을 나타내고 있다. 비단 데몬 쓰레드뿐만 아니라 안드로이드 응용을 구성하는 다양한 UI 컴포넌트에 대하여 사용자 입력과 무관한 여러 무효화 요청이 발생할 수도 있는데, 이러한 상황을 처리하기 위해서는 사용자 입력이 들어오는 시점에 호출되는 콜백 메소드가 수행되는 동안 메인 쓰레드와 다른 쓰레드 간에 주고 받은 메시지를 추적하였다. 안드로이드 프레임워크에서는 이러한 내부적인 메시지의 교환이 sendMessage와 dispatch-Message 메소드로 이루어지므로 두 메소드의 인자와 관계를 분석함으로써 동적인 처리가 가능하도록 하였다. 그림 3은 (a), (b), (c) 시나리오 및 각각의 시나리오에서 사용자 입력 인가 시점부터 이로 인해 발생한 모든 UI 갱신 시점을 추적하는 과정을 요약한 것이다.

### 3.2 설계 및 구현

그림 4는 본 논문에서 제안한 사용자 중심 반응 시간 분석 도구의 구조를 나타낸 것이다. 사용자의 입력, 사용자의 입력의 처리와 연관되어 있는 쓰레드와의 메시지 교환, UI 컴포넌트에 대한 무효화 요청 및 갱신 처리들을 동적으로 감지하기 위하여, 안드로이드 가상 머신의 메소드 호출 및 반환 명령어 번역기를 수정하였다. 이러한 기능은 메소드 호출 후킹 모듈에서 담당한다. 사

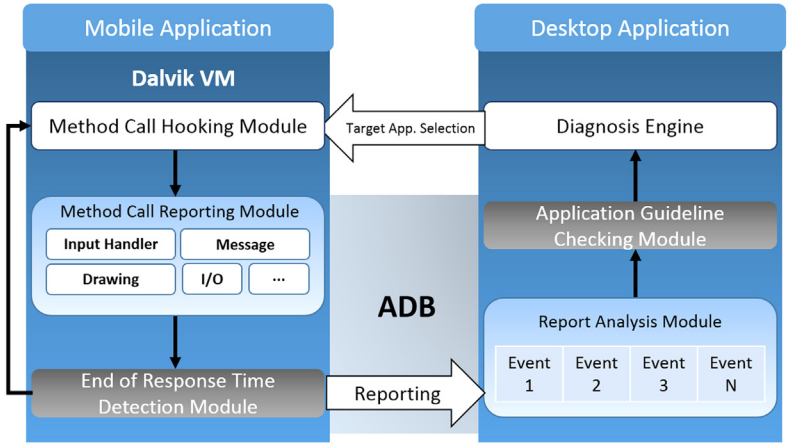


그림 4 사용자 중심 반응 시간 분석 도구의 구조
Fig. 4 An architectural overview of the proposed framework

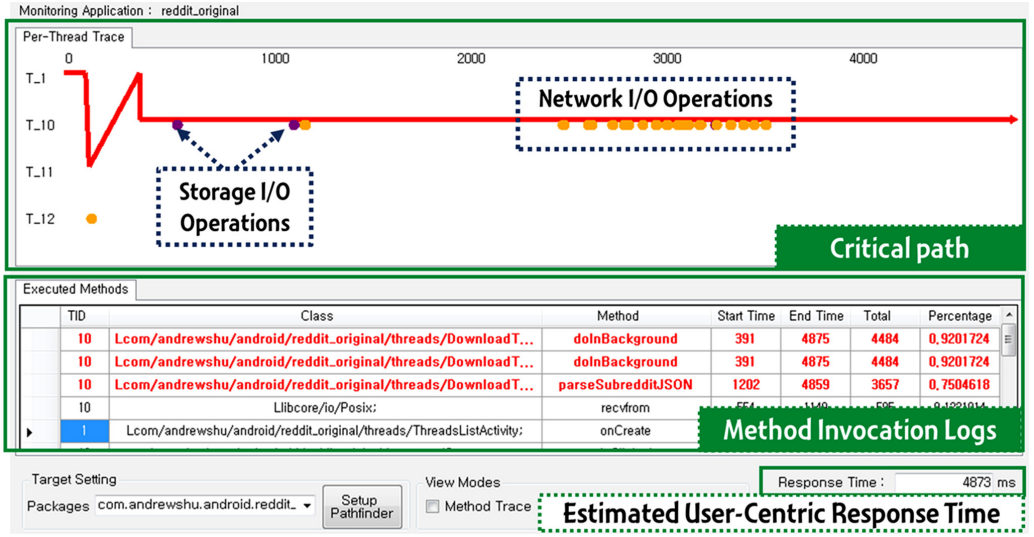


그림 5 사용자 중심 반응 시간 분석 도구가 스마트폰 응용 개발자에게 전달하는 정보의 예시
Fig. 5 An example result from the user-centric response time analyzer (host-side)

용자 중심 반응 시간의 판별과 연관이 있는 모든 메소드들을 그 호출과 반환 시점에 메소드 호출 후킹 모듈에 의하여 감지되고, 호출한 메소드의 종류, 메소드와 함께 전달되는 인자 등이 메소드 호출 기록 모듈에 전달되며 이러한 정보들을 종합하여 반응 시간 종료 판단 모듈은 사용자의 입력을 처리하기 위한 콜백 메소드가 종료되었는지, 또한 이로 인해 작업을 위임 받은 쓰레드들이 자신의 작업을 완료하였는지, 마지막으로 각 실행 흐름으로부터 발생한 모든 UI 갱신이 완료되었는지 여부를 판별한다. 사용자 중심의 반응 완료 시점이 감지되면 메소드 호출 기록 모듈에 저장되어 있는 정보를 호스트 PC의 응용에 전달한다. 이러한 정보들은 ADB 인터페이스를 통하여 전달되며 실제 사용자에게 유용한 정보의 형태로 가공하여 보여주는 역할은 이곳에서 담당한다.

그림 5는 호스트 PC에서 동작하는 응용 프로그램이 개발자에게 보여주는 정보를 나타낸 것이다. 위에서 언급한 Critical Path 정보 이외에도 네트워크 및 저장장치에 대한 I/O, 사용자 중심 반응 완료 시점까지의 메소드 호출 기록 정보 등을 함께 개발자에게 제공한다. 호스트 PC에서 동작하는 응용은 C#을 이용하여 구현하였다.

## 4. 실험 및 평가

본 논문에서 제안한 사용자 중심 반응 시간 분석 도구는 안드로이드 4.0.4가 탑재된 갤럭시 넥서스 스마트폰에 구현하였다. 정확도와 성능 부하를 평가하기 위한 대상 응용은 Google Play Store에서 백만 건 이상의 설치된 응용들 중에서 선정한 것이며, 각 응용의 사용 시나리오는 런칭과 해당 응용을 사용하는 주요 과정 중 한 가지를 사용하였다.

정확도의 평가는 제안한 사용자 중심 반응 시간 분석 도구를 사용하여 측정한 시간과 카메라를 이용하여 30fps 촬영한 동영상과의 비교를 통하여 진행하였다. 카메라를 이용하여 찍은 영상은 프레임 별로 분석하여 사용자가 입력을 인가하는 시점과 이로 인해 발생한 모든 UI의 갱신 시점을 탐지하였다. 그림 6은 이러한 실험을 통하여 확인한 제안한 도구의 정확도 평가를 나타낸 것이다. 총 14개의 응용 사용 시나리오에 대하여 평가한 결과, 평균 92%의 높은 정확도로 사용자 중심 반응 시간 측정이 가능하며 그 크기 역시 100 ms 이내로 매우 정확함을 확인할 수 있었다. 또한 제안한 사용자 중심 반응 시간 분석 도구는 메소드 호출 수준에서의 마지막 UI 갱신 처리 시점을 감지하기 때문에 실제로 스마트폰 화면에 해당 갱신이 반영되어 사용자가 인지하는 시점보다는 항상 조금 먼저 판단을 내리는 것을 확인할 수 있었다. 사용자 중심 반응 시간의 측정을 위해서는 안드로이드 가상 머신 수준에서부터 프레임워크 수준까지 다양한 계층에서의 연산을 필요로 한다. 이에 따른 성능 부하를 평가하기 위하여 제안한 사용자 중심 반응 시간 분석 도구를 제거하였을 때와 포함하였을 때의 수행 시간 분석을 진행한 결과, 전체 반응 시간에서 약 1% 미만의 성능 부하만이 발생함을 확인할 수 있었다.

## 5. 사례 연구: 사용자 중심 반응 시간 분석 도구를 활용한 스마트폰 응용 성능 최적화

본 절에서는 제안한 사용자 중심 반응 시간 분석 도구를 사용하여 스마트폰 응용의 성능 병목을 분석하고 이를 개선하는 사례를 소개한다. 마켓에서 설치 가능한 대부분의 스마트폰 응용은 그 소스 코드가 공개되어 있지 않아 성능 병목을 탐지하였다 하더라도 그것을 개선하여 평가하기 어렵다. 따라서 본 사례 연구에서는 마켓에서 설치 가능하며 널리 사용되는 응용 중에서도 그 소스가 공개되어 있는 응용을 성능 병목 평가 및 최적화 대상으로 선정하였다.

Reddit은 소셜 뉴스 웹사이트로 사용자가 주제별로 Article을 작성하여 공유할 수 있으며, 그 내용에 대한 다른 사용자들의 Feedback에 의해 순위가 결정되고 이 순위에 따라 Article들을 정렬하여 보여주는 서비스이다. 서비스 주체인 Reddit에서 직접 개발하여 배포하는 공식 모바일 응용은 아직 공개된 것이 없지만, 서비스의 규모가 크고, 사용자가 많다 보니 많은 개발자들에 의하여 이를 지원하기 위한 모바일 응용들이 활발히 개발되어 배포되고 있는 실정이다. Reddit-is-fun은 이러한 응용들 중에 가장 유명한 응용 중 하나로 Open-Source임과 동시에 Reddit 웹사이트 에서 제공하는 다양한 기능들을 모바일 응용 수준에서 폭넓게 제공하는 등의 이유로 매우 높은 설치 수를 기록하고 있다.

### 5.1 사용자 중심 반응 시간 분석 도구를 활용한 성능 평가 및 병목 지점 분석

성능 병목 분석을 위하여 Reddit-is-fun 응용의 런칭 과정을 제안한 사용자 중심 반응 시간 분석 도구를 활용하여 평가한 결과는 다음과 같다. 총 5회 수행하여 측정한 사용자 중심 반응 시간은 평균 5860.4ms이며 Critical Path와 네트워크와 저장 장치에 대한 I/O 분석

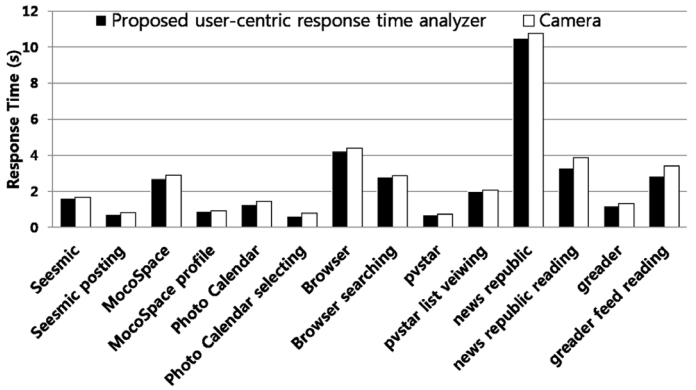


그림 6 사용자 중심 반응 시간 분석 도구와 카메라를 이용하여 각각 측정한 반응 시간 비교

Fig. 6 Response time differences between the proposed user-centric response time analyzer and manual measurements

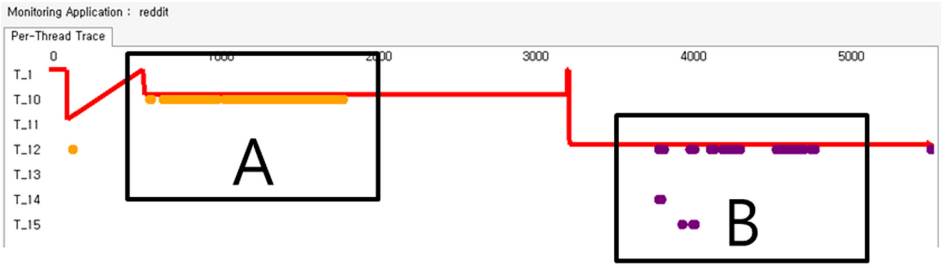


그림 7 사용자 중심 반응 시간 분석 도구를 활용한 Reddit-is-fun 응용의 런칭 과정에 대한 분석
Fig. 7 An analysis result of launching session of Reddit-Is-Fun using the user-centric response time analyzer

은 그림 7과 같다. 그림 7에서의 A 영역은 저장 장치에 대한 I/O를 B 영역은 네트워크를 통한 I/O를 나타내며, Critical Path 안에서 I/O를 처리하는데 상당한 시간이 소요됨을 확인할 수 있다. Critical Path에 속한 메소드의 호출 기록을 살펴보면 Critical Path에서 가장 큰 비중을 차지하는 메소드는 Download ThreadsTask 쓰레드의 doInBackground 메소드와 Show ThumbnailTask 쓰레드의 doInBackground 메소드이다. 먼저, Download-ThreadTask의 doInBackground 메소드를 살펴보면, HTTP를 이용한 네트워크 I/O와 네트워크 I/O를 통하여 내려 받은 JSON에 대한 파싱 작업, 이렇게 크게 두 가지 파트로 구성되어 있다.

위의 분석 결과를 활용하여 크게 두 가지 방향으로 성능을 개선하였다. 첫 번째는 캐싱 메커니즘을 적용하는 것이다. HTTP를 이용한 네트워크 I/O와 네트워크 I/O를 통하여 내려 받은 JSON에 대한 파싱 작업은 매우 부하가 큰 작업이다. 매번 새로운 컨텐츠로 UI가 구성된다면 이는 타당한 동작이지만, 어느 정도 시간이 지나지 않는 경우에는 동일한 컨텐츠 또는 일부만 변경된 컨텐츠로 UI가 구성될 확률이 높다. 성능 개선을 위하여 이 과정에 캐싱 메커니즘을 적용하고 그 향상 정도를 평가하였다. 그 결과 사용자 중심 반응 시간은 4898.6ms로 약 16.4% 향상됨을 확인할 수 있었다. 두 번째는 Down-loadThreadTask 쓰레드와 Show ThumbnailsTask 쓰레드를 분리함으로써 병렬성 향상하는 것이다. 기존의 동작은 DownloadThreadTask 쓰레드의 모든 작업이 끝난 후에, ShowThumbnailsTask 쓰레드가 자신의 작업을 시작하는 것인데, 이는 병렬성을 최대로 살릴 수 있는 구조가 아니다. 따라서, DownloadThreadTask에 의해 모든 데이터가 생성될 때까지 ShowThumbnails-Task 쓰레드가 기다리는 것이 아닌 데이터 각각이 생성될 때마다 Show ThumbnailsTask 쓰레드에 의해 그 데이터들이 처리되도록 병렬성을 향상하였다. 이 경우의 사용자 중심 반응 시간은 평균적으로 4964.6ms로 기존 대비 약 15.3% 향상됨을 확인할 수 있었다.

## 6. 결 론

본 논문에서는 스마트폰 응용에서의 성능 평가 기준으로 사용자 중심의 반응 시간을 제안하였고, 이를 활용한 사용자 중심 반응 시간 측정 및 분석 도구를 제안하였다. 제안한 사용자 중심 반응 시간 측정 도구는 개발자에게 사용자가 체감하는 성능에 대한 정보와 더불어 다양한 최적화 힌트를 제공함으로써 개발자로 하여금 사용자 경험 향상에 활용할 수 있도록 하였다. 또한, 높은 정확도와 낮은 성능 부하로 동작함을 보임으로써 개발자들이 실제 스마트폰 응용을 개발하는데 사용하기에 부족함이 없음을 증명하였다. 또한, 제안한 사용자 중심 반응 시간 분석 도구를 활용하여 특정 응용의 성능 병목을 분석하고 이를 개선하는 과정을 소개하여, 제안한 도구의 효용성 역시 증명하였다. 하지만, 제안한 사용자 중심 반응 시간 측정 도구는 안드로이드 가상 머신 수준에서 메소드의 호출을 추적하는 방식으로 반응 시간을 추적하기 때문에 네이티브 메소드로 구성된 응용에 대해서는 동작하기 어려운 태생적 한계가 존재한다. 향후, 이러한 한계를 보완하여 도구의 활용성을 높일 계획이다.

## References

[1] J. Sobel. (2010, Oct. 19), Making Facebook 2x Faster [Online]. Available: https://code.facebook.com/posts/557831680945510/m aking-facebook-2x-faster/

[2] L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajanm, I.Obermiuller, and S. Shayandeh, "AppInsight: Mobile App Performance Monitoring in the Wild," *Proc. of the International Conference on Operating System Design and Implementation*, pp. 107-120, 2012.

[3] Apple Inc. (2014, Mar. 10), Launch Time Performance Guidelines [Online]. Available: https://developer.apple.com/legacy/library/documentation/Performance/Conceptual/LaunchTime/LaunchTime.pdf (downloaded 2015, Feb. 26)

[4] G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh

N.Thiagarajan, "Who Killed My Battery?: Analyzing MobileBrowser Energy Consumption," *Proc. of the ACM InternationalConference on World Wide Web*, pp. 41-50, 2012.

송 욱

2007년 성균관대학교 정보통신공학부 졸업(학사). 2009년 서울대학교 컴퓨터공학부 졸업(석사). 2009년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 모바일 시스템, 사용자 중심 성능/전력 최적화

성 노 섭

2012년 성균관대학교 정보통신공학부 졸업(학사). 2014년 서울대학교 컴퓨터공학부 졸업(석사). 2014년~현재 SAP Labs Korea 연구원. 관심분야는 모바일 시스템, 소프트웨어 성능 분석

김 지 홍

1986년 서울대학교 계산통계학과 졸업(학사). 1988년 University of Washington 컴퓨터과학과 졸업(석사). 1995년 University of Washington 컴퓨터과학 및 공학과 졸업(박사). 1995년~1997년 미국 Texas Instruments 선임연구원 1997년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 임베디드 소프트웨어, 저전력 시스템, 멀티미디어 시스템, 컴퓨터 구조