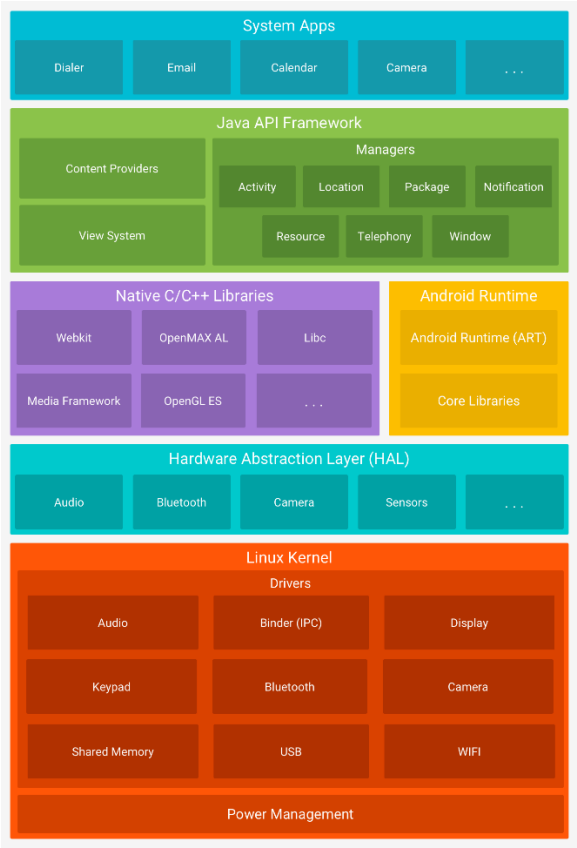
문제정의서(연구계획서)

과제명	AOSP (ANDROID OPEN SOURCE PROJECT) 프레임워크 개선

조	AOSP (ANDROID OPEN SOURCE PROJECT) 조
지도교수	조은선 교수님 (서명)
조원	201503069 김재현 201502088 이송무 201502104 임진현

1. 연구의 필요성



<안드로이드 계층구조>

안드로이드 프레임워크레벨(java api framework)계층은 안드로이드
시스템서비스실행, 자원관리, 애플리케이션의 제작, 실행, 같은
안드로이드가 구동하기위한 기본적인 기능들을 제공하는 역할을 하는 계층이다.
프레임워크 계층을 최적화한다면, 안드로이드 전반적인 성능향상을 기대할 수 있다.

코드에 기여

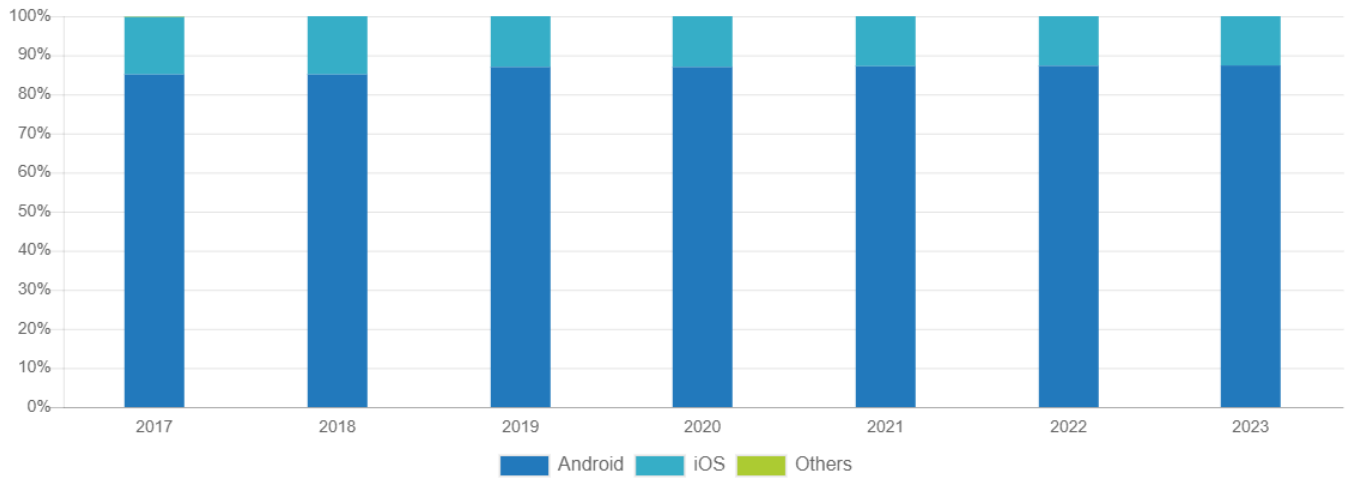
가장 중요한 건 코드입니다. Google은 보내주신 모든 변경사항을 흔쾌히 검토할 용의가 있으니 소스를 확인하고 버그나 특징을 선택한 후 코딩을 확보하세요. 제출하는 패치가 더 작고 타겟팅이 더 잘 되어 있으면 검토하기가 훨씬 수월합니다.

왼쪽에 있는 링크를 사용하여 패치 처리 과정, Git 및 Repo, 기타 도구를 학습함으로써 Android를 시작할 수 있습니다. Gerrit 서버에서 모든 기여에 관한 활동을 볼 수도 있습니다. Android의 일부에서는 업스트림 프로젝트에 패치를 제출하도록 요구합니다. 중간에 도움이 필요한 경우 토론방에 참여할 수 있습니다.

<android source 페이지 기여부분 내용>

오픈소스이기 때문에 환경만 되면 누구나 AOSP (Android Open Source Project) 개선에 기여할 수 있다. AOSP (Android Open Source Project)는 구글 안드로이드에서 구글서비스가 제외된 코어부분이다. 인터넷에서도 수십가지의 최적화 기법들이 올라 와있으며 구글 개발자사이트에서도 주요한 최적화 기법들을 소개해 놓았으며 계속 추가되고 있다.

대규모로 진행중인 가장 대표적인 프로젝트는 구글에서 진행중인
[안드로이드 GO Edition] 라고 할 수 있다. GO EDITION은 인도와 같이 소비력이 높지 않은 시장을 공략하기위해 저전력, 저성능 기기에서도 무리없이 작동할 수 있게 기존 안드로이드(android one)에 전반적인 최적화를 적용한 버전. 기존 버전과는 분리되어 있지만. GO에 적용된 기법들은 종종 기존 버전에 적용되기도 한다
안드로이드 8.1 같은 경우는 GO에디션의 메모리 최적화, 뉴럴 네트워크 api,
공유메모리 개선 api 같은 go에 적용된 최적화 기법들이 적용되었다.



<2017-2023(예측) 스마트폰 os 점유율표>

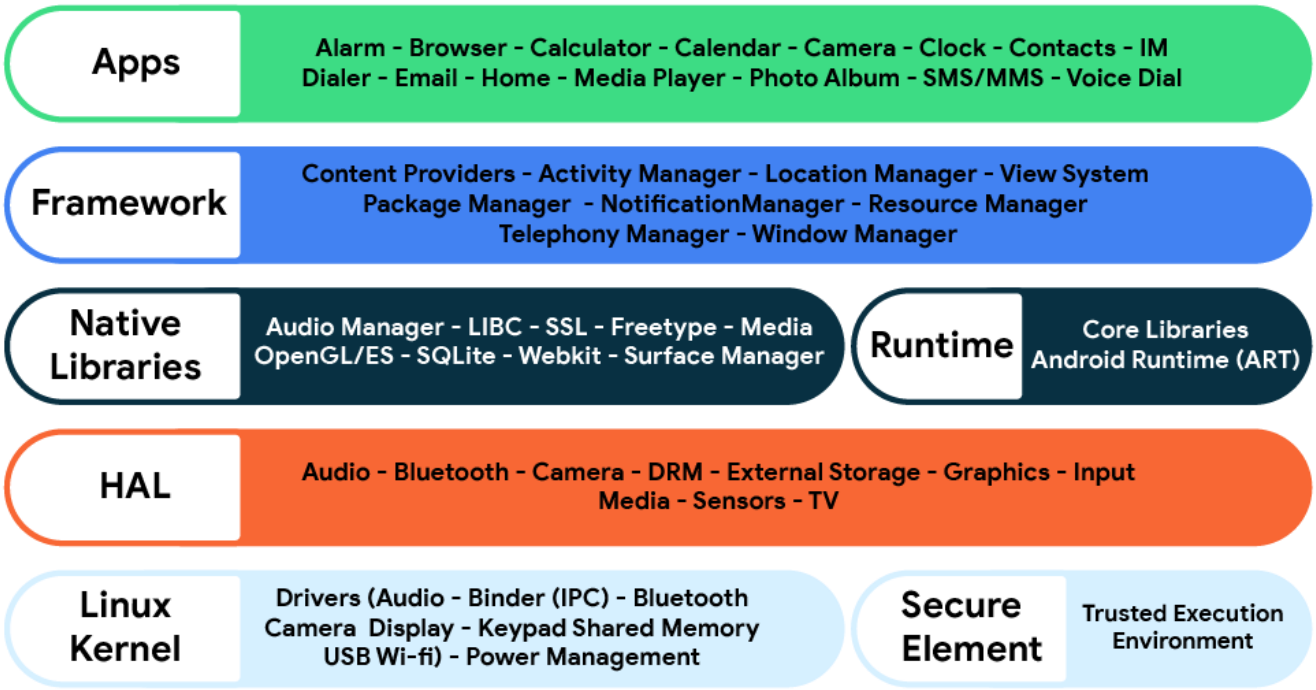
안드로이드는 갈수록 시장에서 독점적인 지위를 굳혀가고 있으며, 모바일을 넘어서 차량, 스마트시계, 스마트tv 같이 다른 기기들로 확장되고 있다. 실제로 구글이 진행하는 차량용 인포테인먼트 플랫폼인 안드로이드 오토는 이미 국내에서도 지원 차량이 점점 늘어나는 중이다. 이런 식으로 안드로이드의 영역이 점점 확장되고, 그 사용자가 증가함에 따라 안드로이드의 성능개선은 점점 더 중요한 과제가 되어가고 있다. 그리고 이런 시장에서의 성공문제가 아니더라도 연구과정에서 안드로이드OS 뿐아니라 OS전반적으로 적용가능한 기술이 나올 수도 있기에 상업적 혹은 기술적 연구가치가 있는 주제이다.

2. 연구의 목표 및 내용

목표:

안드로이드 프레임워크의 성능개선을 통해서

궁극적으로 애플리케이션들의 로딩시간을 단축시키는 것



<안드로이드 프레임워크의 기능들>

프레임 워크가 제공하는 여러 기능 중에 화면의 출력, 관리와 관련된

1.(View System) -(애플리케이션 사용자 인터페이스 생성에 사용되는 확장 가능한 뷰들의 집합)

2.(Window Manager) –(화면에 대한 정보, 배치 등을 관리하는 시스템 서비스)

이 두가지의 소스를 개선하는 데에 초점을 두기로 함.

그리고 애플리케이션 개발언어는 자바지만 안드로이드는 기본적으로 C/C++로 만들어져 있기 때문에 java framework에서 제어할 수 없는 하드웨어적인 부분들은 JNI를 통해 Native library를 호출해서 제어한다. 그렇기 때문에 메인주제인 java framework layer를 넘어선 native library layer와 JNI 관련 내용들도 필요한 부분들은 연구범위에 포함하기로 함

속도를 위해 NDK를 사용하는 방법도 고려해보았으나, Java의 생산성, 호환성이 매우 뛰어나서 ndk는 속도에 정말로 민감한, 게임 같은 앱을 작성하는 경우, java에서 제어가 완전 불가능한 하드웨어를 가진 기기를 제어해야 하는 경우 아니면 일반적으로 쓰이지 않기 때문에 ndk는 고려하지 않는 방향으로 정했다.

3. 연구의 추진전략 및 방법

소스에서 효율을 저하시키는 부분을 찾아내고, 수정하는 식으로 개선
테스트는 프로젝트비로 구입한 공기계로 진행.
구글에서 나온 레퍼런스폰인 픽셀, 넥서스X5 쓰기로 함
버전은 8.1 (api27)

소스코드는 androidrefx.com 이라는 사이트를 이용 위에서 정한 window manager, view system의 코드를 분석 어플리케이션 코드 분석하여 실행 시 window manager와 view system의 동작을 파악
로딩속도에 영향을 주는 부분들을 파악하기
해당 부분들에 여러가지 최적화 기법들을 적용해보면서 실행시간 변화를 측정
반복하며 이해도가 높아지면 최적화기법을 개선하거나, 직접 개발해보기

```
public void put(String key, String value) {
    mValues.put(key, value);
}
public void put(String key, Byte value) {
    mValues.put(key, value);
}
public void put(String key, Short value) {
    mValues.put(key, value);
}
```

그림 1 ContentValues 클래스에서 발견된
코드스멜(이하 중략)[4]
(컨텐츠 프로바이더, 코드스멜 리팩터링)

3. 메모리 복사 기반 제안 방식

본 논문에서는 메모리 블록을 복사하는 memcpy() 함수를 사용해 메모리에 접근하는 횟수를 줄이는 방식을 사용한다. 메모리 접근 횟수를 최소화하여 패키지 스캐닝 속도를 향상시키는 방법이다.

바이트(byte) 단위를 사용하여 복사하는 strlcpy() 함수를 대신해 보다 큰 워드(word) 단위의 복사를 하는 memcpy() 함수를 활용하여 [그림 3]과 같이 수정하여 구현하였다. 기존 strlcpy() 함수와 동일한 기능을 유지하기 위해 복사된 문자열의 마지막 값은 널로 끝나도록 구현하였고 복사하려는 문자열(src)의 길이를 반환한다.

```
size_t
strlcpy(char* dst, const char* src, size_t siz)
{
    size_t srclen, returnV = strlen(src);

        siz--;

        srclen = returnV;

        if (srclen > siz)
            srclen = siz;

        memcpy(dst, src, srclen);
        dst[srclen] = '₩0';

        return returnV;
}
```

[그림 3] 개선 후 소스코드

(메모리접근 최소화, 패키지 매니저)

불필요한 개체 생성 방지 virtual보다는 static 사용 상수에 static final 사용 비공개 내부 클래스의 비공개 액세스 대신 패키지 고려 부동 소수점 사용 피하기, 등등

4. 연구 팀의 구성 및 과제 추진 일정

팀구성 : 팀원이 각자 맡은 소스코드를 분석해와서 설명하고,
함께 문제점, 최적화방법 토론

1학기:

안드로이드 프레임워크의 기본적인 개념공부.

기본적인 최적화 익히기

2학기:

본격적인 코드분석과 최적화

기존 최적화기법의 분석, 개선

1학기 상세일정

3월 – 5월20일

안드로이드의 기초를 공부,

조은선 교수님과 매주 hangout으로

공부한내용을 간단히 브리핑하는 식으로 스터디진행

https://sites.google.com/cs-cnu.org/plas/undergraduate-projects/android-framework

깃허브 블로그를 이용해 내용을 정리하는중

5월20일 – 6월:

코드분석과 성능분석을 위한 환경구성

6월이후

본격적 코드분석,

프레임워크 개선방법공부

적용해보면서 성능변화 테스트

분석 – 개선 – 적용을 계속 반복하여 점점 개선하기

- 참고문헌(Reference)

https://developer.android.com/guide/platform?hl=ko

https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09301959

http://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09301958

프로젝트주소

https://github.com/keelim/project_aosp

https://www.youtube.com/channel/UC05ce0W79PQWj6H86cN4_LQ/videos