안드로이드 시스템에서 프로그램 언어 비교 연구

장승주*, 김성진*
*동의대학교 컴퓨터공학과

Comparison Program Language in the Android System

Janf, Seung Ju*, Kim Seung Jin*
*Dongeui Univ., Dept. of Computer Engirneering

요 약

본 논문은 안드로이드 시스템에서 자바 프로그램 환경과 NDK를 이용한 C 프로그램 환경에 대해서 비교한다. 안드로이드 시스템에서 NDK를 이용한 C 프로그램의 성능 개선이 어느 정도 이루어질 수 있는지를 실험한다. 본 논문은 1에서 n까지의 숫자의 합을 계산하는 프로그램을 자바 프로그램과 NDK를 이용한 C 프로그램으로 작성한다. 각각 작성된 프로그램에서 n 값을 가변적으로 변화시킴으로써 성능에 어떤 영향을 주는지 실험한다.

Abstract

This paper compares the Java program environments and NDK program environments in the Android system. This paper experiments that how much is the performance enhancement in the NDK C programming of the Android system. I program sum of 1 to n for Java program and NDK C program. I experiment whether the influence an effect on a performance by changing n value. In experiment, NDK C program is over 50% performance enhancement than Java program.

키워드

안드로이드 시스템, 자바 프로그램, NDK 프로그램, C 프로그램, 성능 비교

Ⅰ. 서 론

최근에 전 세계적으로 많은 사람들이 안드로이드 스마트 폰을 사용하고 있다. 안드로이드 스마트 폰 시스템이 급속하게 확산이 되고 있는 추세이다. 안드로이드 시스템이 급속하게 확산이 되는 이유로 안드로이드 시스템이 개방형 시스템을 지향하고 있기 때문이다. 개방형 시스템은 기반이 되는 운영체제와 같은 소프트웨어의 소스 코드를 공개하고 있다. 이와 같이 개방형 시스템은 많은 사용자들이 자유롭게 시스템의 소프트웨어를 이용할 수 있다 [1-3].

원래 스마트 폰 시장을 초창기에 주도를 했던 애플의 경우는 폐쇄형 시스템을 가지고 시장에 진입을 하게 된다. 처음에는 애플이 성공하는 듯이 보였지만 시간이 지나면서 안드로이드 시스템의 개방형 시스템이 위력을 발휘하고 있다. 안드로이드 시스템이 스마트 폰 시스템의 주도권을 장악하면서 안드로이드 시스템 환경에서 구동되는 소프트웨어 시스템 구조에 대한 연구도 활발하게 이루어지고 있다.

그중에 하나가 안드로이드 시스템에서 주로 사용하는 자바 프로그램 구조에 대한 연구이다. 자바 프로그램은 사용자들이 편리하게 프로그램을 할 수 있는 장점은 있지만 거치는 인터페이스가 많다는 단점을 가지고 있다. 본 논문은 이러한 관점에서 안드로이드 시스템에서 많이 사용하는 자바 프로그램 시스템 환경에 대한 문제점을 짚어보고, 이것의 대안으로써 NDK를 이용한 C 프로그램과의 비교를 통해서 어떤 방향으로 소프트웨어 구조가 나아가야할지를 제시한다.

본 연구 논문은 안드로이드 시스템에서 C 프로그램을 이용한 성능 개선이 어느 정도 이루어질 수 있는지를 실험한다. 본 논문은 1에서 n까지의 숫자의 합을 계산하는 프로그램을 자바 프로그램과 NDK를 이용한 C 프로그램으로 작성한다. 각각 작성된 프로그램에서 n 값을 가변적으로 변화시킴으로써 성능에 어떤 영향을 주는지 실험한다. 실험 결과 NDK를 이용한 C 프로그램이 자바를 이용한 프로그램보다 50%이상의 성능 개선이 됨을 확인할 수 있었다.

본 논문의 구성은 2장에서 관련 연구를 언급한다. 3장에서는 본 논문에서 제시하는 자바 프로그램과 C 프로그램의 동작 구조 및 설계된 내용에 대해서 설명한다. 4장은 실험 내용을 언급한다. 5장에서 결론을 내린다.

II . 관련 연구

안드로이드 시스템은 일반적으로 자바 프로그램을 이용하여 프로그램을 한다. 안드로이드 시스템 구조는 다음 그림 1과 같다.

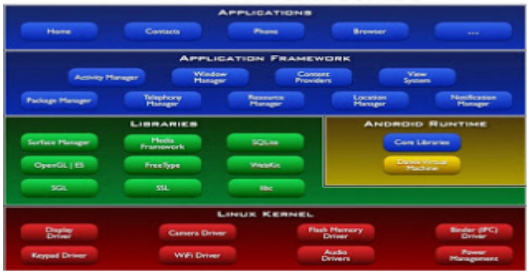

그림 1. 안드로이드 시스템 구조

안드로이드는 JAVA 언어를 기반으로 하고 있으며 개발에 필요한 모든 과정이 무료이다. 즉, 개발자가 개발자 마음대로 개량과 수정이 가능하다는 이야기이다. 하지만 모든 것들을 내 마음대로 할 수 있다는 것은 아니다. 이런 안드로이드는 Linux Kernel, HAL(Hardware Abstraction Layer), Libraries, Application Framework, Application 등 5개의 계층으로 이루어져 있다.

안드로이드의 NDK는 JAVA의 JNI 기술을 이용하여 안드로이드 App(Java)과 안드로이드 Library(C/C++) 계층을 연결해주는 역할을 한다. JVM에서 C/C++과 같은 언어로 구현된 것을 네이티브 함수(네이티브 메소드)라 한다. JDK는 표준 프로그래밍 인터페이스인 JNI를 지원한다. 즉, JVM에서 C/C++ 코드를 다소 이식성 있게 호출할 수 있다 [4-5].

NDK(Native Development Kit)는 안드로이드 시스템의 경우는 응용 프로그램이 Dalvik VM 상에서 수행이 된다. 또한 프로그램 언어는 자바를 사용하게 된다. 자바 언어를 사용하게 되면 편리하게 프로그램을 할 수 있는 측면이 있어서 사용자들이 선호한다. 하지만 VM를 거치게 되고 여러 단계의 수행코드를 거치는 문제를 가지고 있다. 이러한 NDK 프로그램은 안드로이드 프로그램에 특화해서 프로그램이 가능하다. NDK 프로그램 언어로 C나 C++를 사용 가능하다. NDK 프로그램의 장점은 기존의 C나 C++프로그램을 이용 가능하다는 것이다 [6-7].

우리가 시스템 디바이스에 접근할 필요가 있거나 혹은 자바의 성능을 넘어선 플랫폼 특정적인 작업을 수행할 경우에 JNI를 사용할 수 있다. 사실 많은 자바 라이브러리 루틴들이 결과적으로는 내부(private) 네이티브 메소드를 이러한 목적으로 호출하고 있다. 자바로 구현하기엔 어플리케이션이 매우 느릴 수 있으며, C++에서 time-critical 코드를 구현함으로써 성능의 향상을 가져올 수 있을 경우에 JNI를 사용한다.

또한 NDK를 사용할 경우 단점도 있다. 이식성(portability)을 잃게 된다. 많은 자바 라이브러리가 네이티브 메소드를 사용하고 있는데, 새로 추가된 네이티브 메소드가 모든 플랫폼에서 호출 가능한 동일한 C++ 코드를 가진다면, 그 코드는 자바에서 가장 먼저 구현되어질 수 있다. 네이티브 메소드는 자바 메소드처럼 동일한 보호를 보장받지 못한다.

III . 자바 프로그램과 NDK를 이용한 C 프로그램 설계

본 논문은 안드로이드 시스템 환경에서 NDK 환경을 이용하는 것이 유리하다는 것을 프로그램 설계를 통해서 구현하였다. 본 논문에서 제시하는 안드로이드 시스템에서 자바로 작성한 프로그램과 NDK를 이용하여 C언어로 프로그램을 작성한다. 우선 동일한 기능을 하는 프로그램을 작성한다. 구동되는 프로그램은 다음과 같은 연산을 수행한다.

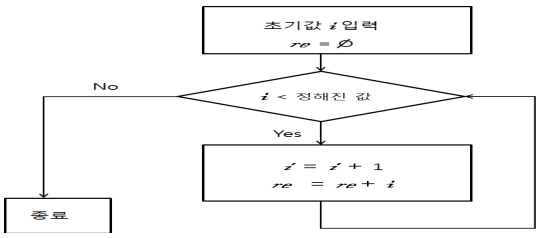

그림 2. 프로그램 동작 흐름도

그림 2는 본 논문에서 구현하고자 하는 프로그램의 흐름도이다. 이것은 1에서 n까지의 합을 계산하는 과정이다. 이 기능은 계산량이 방대하고 n 값이 클 경우에 계산 결과에 시간이 많이 소요된다. 따라서 본 논문에서 수행하고자 하는 성능 평가에 적합하다고 할 수 있다.

위에서 제시한 기능에 대해서 자바 프로그램 언어로 설계하고, 같은 기능으로 JNI를 이용한 C 언어로 프로그램을 설계한다. 아래 <그림 3>은 자바 프로그램으로 설계된 프로그램이다.

```
package com.example.sum_java;

//안드로이드 관련 클래스 선언

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Button btn = (Button) findViewById(R.id.button1);
      btn.setOnClickListener(new Button.OnClickListener() {
      @Override
          public void onClick(View v) {
          // TODO Auto-generated method stub
          long sum = 0;
          EditText edit = (EditText) findViewById(R.id.editText1);
```

```
            String temp1 = edit.getText().toString();
            long temp2 = Long.parseLong(temp1);
                for (long i = 1; i <= temp2; i++) {
                    sum = sum + i;
                }
                String res = Long.toString(sum);
                Toast.makeText(MainActivity.this,                    res,
Toast.LENGTH_SHORT)
                .show();
            }

        });
    }
.....
}
```

그림 3. 자바 언어로 작성된 프로그램

그림 3은 안드로이드 시스템에서 자바로 작성한 프로그램이다. 프로그램 기능은 그림 2에서 설명한 바와 같이 1에서 n까지 숫자의 합을 계산하는 것이다. 먼저 안드로이드 관련 클래스를 정의한다. 그리고, 버튼을 통해서 프로그램이 실행될 수 있도록 선언한다. 그리고, 그림 3에서 for 문장에서 실제 수행하게 되는 최종 값은 temp2까지이다. 이 값이 최종적으로 더하게 되는 마지막 n 값에 해당한다. 최종적으로 계산한 값이 sum 변수에 저장이 된다. 이 변수의 값이 최종 계산 값으로 출력이 되게 된다.

마찬가지로 동일한 기능을 하는 프로그램에 대해서 NDK를 이용해서 설계된 C 프로그램이다. 아래 그림 4는 NDK를 이용하여 C 언어로 설계된 프로그램이다.

```
package com.example.sumjni;

public class sumJNI {
    public native long sum(long first, long sec);
    static {
       System.loadLibrary("sumJNI");
    }
}
package com.example.sumjni;
public class sumJNI {
    public native long sum(long first, long sec);
       static {
          System.loadLibrary("sumJNI");
       }
}
```

그림 4. NDK를 이용한 C 프로그램

위 그림 4는 NDK를 이용한 C 프로그램이다.

Ⅳ. 실 험

본 논문에서 제시하는 설계 내용을 구현하고 실험을 하기 위하여 다음과 같은 환경을 구축하였다. 먼저 우선적으로 안드로이드 시스템 환경을 위한 NDK 환경을 설치하였다.

그림 5는 자바 환경에서 실험 화면이다.

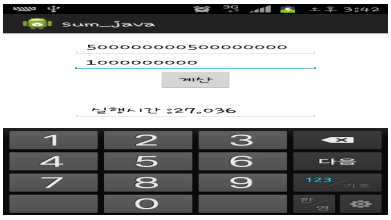

그림 5. 자바 환경에서 실험 화면

그림 5는 자바 환경에서 성능 측정을 위한 화면이다. 프로그램에서 연산 수행 횟수를 10억으로 설정한다. 10억으로 설정했을 때 실행 시간이 27초가량 나오는 것을 확인할 수 있었다. 이러한 실험 결과 20회 수행한 결과를 정리한 것이 표 1이다.

표 1. JAVA 환경에서 수행 시간 측정 결과 (단위 : 초)

100000 (십만)	1000000 (백만)	10000000 (천만)	100000000 (억)	1000000000(십억)
0.014	0.091	0.346	2.868	27.036
0.012	0.12	0.367	2.91	27.571
0.015	0.045	0.382	2.806	28.282
0.019	0.079	0.395	2.862	26.776
0.021	0.101	0.375	2.801	26.574
0.006	0.08	0.378	2.808	27.77
0.04	0.078	0.378	2.824	26.173
0.032	0.116	0.437	2.77	26.673
0.038	0.09	0.34	2.938	26.612
0.032	0.139	0.374	2.78	26.641
0.015	0.12	0.401	2.849	27.154
0.028	0.078	0.356	2.874	26.541
0.031	0.085	0.338	2.899	26.12
0.011	0.086	0.349	2.79	26.874
0.032	0.081	0.366	2.824	26.658
0.014	0.116	0.381	2.989	26.451
0.031	0.127	0.384	2.788	26.657
0.008	0.093	0.344	2.76	26.74
0.012	0.109	0.367	2.802	27.011
0.04	0.136	0.405	2.778	26.891
평균 : 0.023	평균 : 0.099	평균 : 0.373	평균 : 2.836	평균 : 26.860

표 1은 자바 환경에서 성능 측정을 수행한 결과를 보여준다. 성능 측정은 십만, 백만, 천만, 1억, 십억까지의 계산을 했을 경우의 성능 측정 결과를 보여준다. 같은 숫자에 대해서 20회 성능 측정을 수행하였다. 20회 성능 측정을 수행한 평균값을 하단에 보여주고 있다. 그림 6은 NDK환경에서 실험 화면을 보여주고 있다.

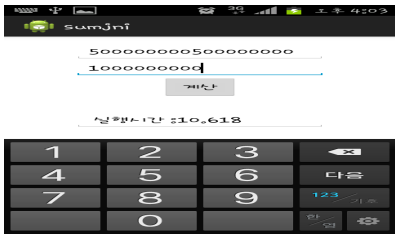

그림 6. NDK 환경에서 실험 화면

그림 6은 NDK 환경에서 성능 측정을 위한 화면이다. 프로그램에서 연산 수행 횟수를 10억으로 설정한다. 10억으로 설정했을 때 실행 시간이 10초가량 나오는 것을 확인할 수 있었다. 이러한 실험 결과 20회 수행한 결과를 정리한 것이 표 2이다.

표 2. NDK 환경에서 수행 시간 측정 결과 (단위 : 초)

100000 (십만)	1000000 (백만)	10000000 (천만)	100000000 (억)	1000000000 (십억)
0.004	0.035	0.19	1.174	10.646
0.025	0.095	0.215	1.211	10.772
0.014	0.134	0.174	1.182	10.689
0.003	0.092	0.182	1.204	10.604
0.026	0.105	0.187	1.257	10.556
0.004	0.03	0.168	1.206	10.643
0.026	0.037	0.144	1.263	10.63
0.009	0.06	0.213	1.206	11.087
0.021	0.088	0.213	1.173	10.769
0.027	0.032	0.188	1.164	10.618
0.01	0.081	0.23	1.167	10.55
0.017	0.105	0.186	1.221	10.597
0.004	0.038	0.196	1.151	10.682
0.02	0.03	0.188	1.229	10.635
0.02	0.045	0.164	1.23	10.526
0.018	0.083	0.23	1.133	10.417
0.031	0.039	0.185	1.261	10.566
0.009	0.032	0.163	1.171	10.686
0.003	0.041	0.188	1.233	10.745
0.014	0.099	0.195	1.184	10.525
평균 : 0.015	평균 : 0.065	평균 : 0.190	평균 : 1.201	평균 : 10.647

표 2는 NDK 환경에서 성능 측정을 수행한 결과를 보여준다. 성능 측정은 십만, 백만, 천만, 1억, 십억까지의 계산을 했을 경우의 성능 측정 결과를 보여준다. 같은 숫자에 대해서 20회 성능 측정을 수행하였다. 20회 성능 측정을 수행한 평균값을 하단에 보여주고 있다.

위 실험 결과에서 자바를 사용한 환경이 NDK를 사용한 환경보다 전반적으로 50%이상 느림을 알 수 있다. 10억을 수행한 경우에는 100%이상 성능이 느림을 확인할 수 있었다. 따라서 안드로이드 환경에서 프로그램을 할 때 자바 환경보다 NDK환경을 이용한 프로그램이 성능 등의 측면에서 유리함을 알 수 있다. 특히, 계산량이 많은 환경에서는 자바 프로그램 환경이 불리함을 확인할 수 있다.

V. 결 론

본 논문은 안드로이드 시스템에서 자바 프로그램 대신 C 프로그램을 사용할 경우 성능 개선이 어느 정도 되는지에 대한 연구를 수행하였다. 본 논문에서 제안하는 방식은 기존의 자바 프로그램을 이용하는 대신에 NDK를 이용한 C 프로그램 방식이 어느정도 성능 개선에 도움이 되는지에 대한 연구이다.

기존의 안드로이드 시스템은 대부분이 자바 언어로 프로그램하도록 되어 있다. 이것은 사용자들이 쉽고 편리하게 안드로이드 시스템에서 프로그램할 수 있도록 하기 위함이다. 그러나 이와 같은 방식은 VM을 거치게 되면서 시스템 상에 오버헤더가 발생한다는 것이다.

본 논문에서는 1에서부터 n까지 더하는 프로그램을 자바 언어와 NDK를 이용한 C 언어로 실제 안드로이드 시스템 상에서 구현하였다. 이렇게 구현된 프로그램을 이용하여 자바 프로그램의 실행 결과에 소요되는 시간과 C 언어에서 실행하는데 소요되는 시간을 측정하였다. 본 논문에서 실험을 수행한 결과 NDK를 이용한 C 언어 프로그램 방식이 자바 언어 프로그램 방식 보다 50%에서 100% 정도 성능 개선이 있음을 확인하였다. 본 논문에서 실제 구현하여 성능 측정한 결과에서 제시하듯이 안드로이드 시스템에서 빠르게 실행이 필요한 프로그램은 자바 언어보다 C언어로 프로그램하는 것이 성능적인 측면에서 유리함을 알 수 있다.

참고문헌

[1] 서화정, 김호원, “NDK 기반 공개키 암호를 위한 곱셈기 구현 및 분석”, 한국정보통신학회논문지, 제16권 11호, pp. 2347-2354, 2011.11.

[2] 여지환, 정원기, 문수묵 “LLVM을 활용한 안드로이드 NDK의 이식성 향상”, 한국정보과학회 2012 가을 학술발표논문집 제39권 제2호(A), pp.179-181, 2012.11

[3] 정원기, 김범준, 문수묵, “LLVM의 안드로이드 호환성 확보를 위한 연구”, 한국정보과학회 2011가을 학술발표 논문집 제38권 제2호(A), pp.238-240, 2011. 11.

[4] 서화정, 김호원 “안드로이드 기반 공개키 암호를 위한 곱셈기 구현 및 분석”, 한국통신학회논문지 제37권 제10호(융합기술), pp.940-948, 2012. 10.

[5] 김태권, 김봉완, 최대림, 이용주 “안드로이드 OS기반 한국어 TTS 서비스의 설계 및 구현”, 한국콘텐츠학회논문지 제12권 제1호, pp.9-16, 2012. 1.

[6] 손미경, 강남희 “안드로이드 플랫폼을 위한 자바 보안 프로바이더 설계 및 구현”, 한국통신학회논문지 제37권 제9호(융합기술), pp.851-858, 2012. 9.

[7] 손기철 “안드로이드 NDK를 이용한 어플리케이션 속도 향상 예측기”, 전북대학교 국회도서관 (321.381 -12-6), 2012.