모바일 환경에서 최적화된 리소스 제공을 위한 능동적 빌드 시스템 설계

이현섭* · 김진덕*

*동의대학교

The design of Active build system For the provision of the optimized resources in a mobile environment

Hyoun-sup Lee* · Jin-deog Kim*

*Dongeui University

E-mail : lhskmj@deu.ac.kr

요 약

모바일 어플리케이션에 활용되는 리소스의 경우 사용되지 않는 데이터가 포함되는 경우가 많이 존재한다. 시스템 환경에 대응하기 위해 여러 가지 형태의 리소스를 구축하고 어플리케이션 실행 시 시스템 환경을 분석하여 구축된 리소스 중 최적화된 일부를 활용하여 어플리케이션이 구동된다.

이 경우 사용되지 않는 리소스 데이터 증가로 인해 한정적인 시스템 저장소에 영향을 주게 되며 설치된 어플리케이션의 수가 증가될수록 낭비되는 공간 또한 늘어난다.

본 논문에서는 이러한 불용 리소스의 증가로 인한 시스템 성능 저하를 해결하기 위해 능동적인 빌드 시스템에 대하여 제안한다. 제안하는 기법은 어플리케이션 설치 이전에 시스템의 정보를 확인하고 이에 따라 최적화된 리소스만을 활용하여 실행파일을 빌드하고 사용자에게 제공하는 방법이다. 이 방법을 적용할 경우 어플리케이션 설치 용량이 줄어들고 불필요한 리소스의 제공으로 인해 발생할 수 있는 시스템 용량 부족 문제를 해결할 수 있을 것으로 판단된다.

키워드

안드로이드, 리소스, 퀄리파이어, 사용자 분석, 능동적 빌드

Ⅰ. 서 론

모바일 어플리케이션에 활용되는 리소스의 평균 용량은 과거에 비하여 많이 증가되었다[1]. 하드웨어의 성능 증가에 따라 데이터 처리량이 높아졌고 이로 인해 고수준 리소스를 요구하는 어플리케이션의 수도 증가되었다[2].

일반적으로 어플리케이션의 원활한 실행을 위해 리소스를 사용하지만 모바일 환경에 능동적으로 대응하기 위해 리소스를 활용하는 경우도 많다. 즉, 실행과정에서 어플리케이션이 가진 전체 리소스의 일부만이 사용된다는 의미이다.

안드로이드 시스템의 퀄리파이어 정보를 활용한 리소스 처리 기법이 앞서 언급한 일부 리소스 일부 사용의 대표적인 예이다. 퀄리파이어는 안드로이드의 시스템 정보를 중요도에 따라 레벨로 구분하여 관리하기 위한 구조를 의미한다. 최상위 레벨에서는 국가코드, 통신 회사 정보 통신 코드 정보 등이 있으며 이후 화면 크기, 해상도, 방향, 언어정보, 특정 API 활용 등의 정보 등의 시스템 정보를 구분한다.

어플리케이션에 사용될 리소스 구축과정에서 특정 단말기 및 구동 환경을 구분하여 리소스폴더를 구성한다. 이후 어플리케이션 구동 과정에서 퀄리파이어 정보를 수집하고 이를 활용해 리소스 폴더를 선택하여 최적화 실행을 지원한다.

그러나 이 경우 퀄리파이어 정보 분석을 위한 연산 시간과 선택되지 못한 리소스의 공간 낭비가 발생한다.

본 논문에서는 이러한 연산 비용의 문제를 해결하기 위해 능동적 빌드 시스템에 대하여 제안한다, 2장에서 리소스 관련 비용 감소를 위한 기

존 연구의 특징과 본 연구와의 차이점을 설명한다. 3장에서는 본 논문에서 제안하는 능동적 빌드 시스템의 구조와 기능에 대하여 설명하며 이어오는 4장에서 결론을 맺는다.

Ⅱ. 관련연구

리소스의 크기를 줄이는 방법 중에서 사용되지 않는 메소드를 삭제하여 코드의 최적화를 수행하는 기법에 대한 연구 [3]에서는 비용 감소를 위해 프로그램된 코드에서 호출되지 않는 메소드와 리소스를 검출해서 제거하는 방법에 대하여 제안하고 있다.

이 경우 불필요 코드를 제거함으로써 시스템의 안정화와 용량의 최적화 등을 이루어 낼 수 있지만 한번이라도 호출되는 메소드들로 이루어져있는 시스템의 경우 적용할 수 없다.

또한 코드의 요류를 수정한 경우라 하더라도 시스템에 대응하기 위해 저장한 리소스는 실행을 위해 꼭 필요한 요소기 때문에 제안된 방법으로는 용량을 최적화 할 수 없다.

안드로이드 시스템에서 어플리케이션의 응답시간 향상을 위한 실행시간 선행 컴파일 체계에 관한 연구[4]에서는 특정 응용이 실행되기 전에 주요 바이트코드들을 VM 위에서 미리 컴파일 하는 AOT(Ahead-of-Time)컴파일 체계에 대해 제안하였다. 이 체계를 활용할 경우 어플리케이션의 응답시간이 약 15% 정도 단축되는 결과를 보였다.

그러나 필요 리소스를 구분하고 저장소에서 읽어 오는 과정의 경우 주요 바이트 코드와는 별도로 구성되어 있기 때문에 리소스로 인한 시스템 저하 문제는 해결할 수 없다.

이어오는 3장에서 서론과 관련연구에서 언급한 불용 리소스 문제를 해결하기 위한 능동적 빌드 시스템 ABaR(Active Build and Resource)구조에 대하여 설명하고 처리 기능에 대하여 제안한다.

Ⅲ. ABaR 구조

그림 1은 본 논문에서 제안하는 ABaR의 구조도이다.

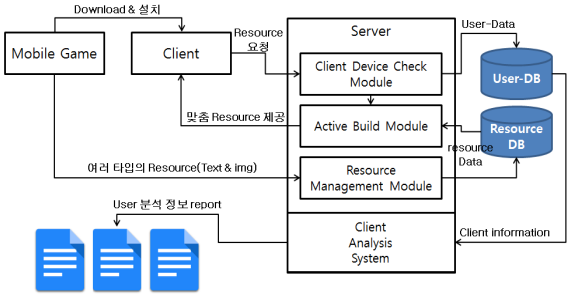

그림 1. ABaR 구조

리소스를 가장 많이 소비하는 어플리케이션 영역인 모바일 게임을 기반으로 설계된 시스템 구조이며 총 4개의 핵심 모듈로 구성된다.

첫 번째 모듈은 Client Device Check Module로 ABaR에 접근하는 사용자의 정보를 체크하는 모듈이다. 디바이스의 정보 중 개인정보(사용자)를 제외한 특정 시스템 정보들을 수집하는 기능을 수행한다.

표 1은 수집정보 리스트 이며 수집된 정보를 ABaR User DB에 저장하고 Active Build Module을 호출한다.

표 1. ABaR 수집 디바이스 정보

항목	구체적인 수집 정보
시스템 정보	OS 타입 정보 (ex : window, ios, android ... etc)
국가 정보	언어 정보 및 사용 국가 정보
디바이스 정보	현재 설치된 단말기의 정보 (ex : 아이폰 5s, 갤럭시 노트, G, 샤오미 ... etc)
build 버전 정보	설치된 모바일 게임의 build 버전 정보
HW 정보	해상도(LCD, LED 타입), 활용 센서 정보, 네트워크 디바이스 타입, 프로세서 처리 정보

두 번째 모듈은 Active Build Module이며 Client Device Check Module 통해 입력된 Client정보를 토대로 매치되는 Build 정보에 해당하는 리소스 및 어플리케이션 데이터, 실행 파일을 제공하는 모듈이다.

국가별, 빌드별로 따로 색인 및 파일서버를 구축할 필요 없이 본 모듈을 통해 하나의 서버 및 시스템으로 사용자 정보에 따라 데이터 제공 한다.

개발자는 어플리케이션 build 시에 한 종류의 method, 알고리즘을 구성하더라도 client 정보에 따라 ABaR이 필요 정보를 제공할 수 있어 여러 타입의 빌드 및 변수, method를 고려할 필요가 없다.

세 번째 모듈은 리소스를 저장하고 관리하기 위한 Resource Management Module이다. 개발자는 제공하기 위한 여러 가지 build 타입의 리소스를 ABaR에 저장하기 위해 본 모듈을 호출한다.

ABaR은 본 모듈을 통해 입력된 정보를 Resource DB에 저장하고 Active Build Module에서 저장된 정보와 Client Device Check Module에서 수집된 정보를 비교하여 Client 정보에 따라 데이터를 제공하게 된다.

네 번째 모듈은 Client Analysis System으로 ABaR의 기능을 보조하기 위한 독립된 모듈이다.

일정 수준 이상 리소스 제공 이후 User-DB에

저장된 정보를 토대로 User Report를 제공하는 모듈로서 사용자의 패턴 정보, 다운로드 받은 디바이스의 종류 카운트, 국가 정보 등을 토대로 향후 서비스 방향과 리소스 관리 정책을 선정할 수 있다.

오류 및 실행 문제에서도 가장 우선 서비스해야 하는 대상 정보와 향후 개발 사이클에 올라있는 어플리케이션의 예측 디바이스 정보 등을 판단하는데 지원하는 휴리스틱 정보로 활용된다.

그림 2와 같이 ABaR을 활용할 경우 하나의 빌드 타입 에서 여러 타입의 build resource를 제공하게 된다.

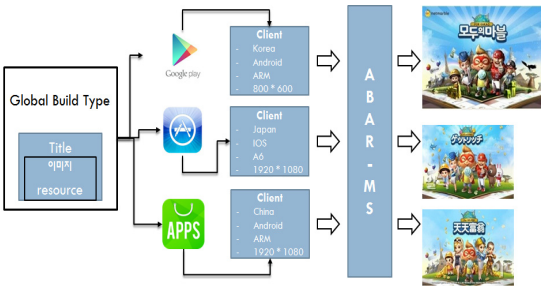

그림 2. ABaR 활용도

ABaR을 통해 하나의 Build 타입이 여러 개의 실행 환경에 대응되는 리소스를 제공하는 구조를 볼 수 있다.

실행 시 리소스를 탐색할 필요가 없으며 어플리케이션 구동에 필요한 최적화된 리소스를 하나만 가질 수 있도록 지원하는 것이 ABaR의 핵심 기능이다.

Ⅳ. 결 론

기존에는 특정 환경과 디바이스에 대응하기 위한 어플리케이션을 개발하기 위해서는 활용되는 리소스를 모두 저장하고 구동환경을 체크하여 저장된 리소스에서 환경에 대응되는 데이터를 추출하여 실행하는 구조가 대부분이었다.

특히 안드로이드 환경의 경우 파편화 문제로 인하여 이러한 다중 리소스 구성 구조가 개발자에게는 선택이 아닌 필수 해결 문제로 항상 제기되었다.

이로 인해 개발자는 리소스 판별을 위한 불필요한 알고리즘의 구현을 수행해야하며 시스템 환경에 원활한 대응을 위해 복잡한 시스템 구조가 구성되는 문제가 발현된다.

사용자 입장에서도 어플리케이션의 구동 속도의 저하와 디바이스의 저장 공간의 불필요한 소모가 발생하여 어플리케이션 사용에 대한 만족도가 떨어지는 문제가 발생한다.

본 논문에서는 이러한 문제점을 해결하기 위해 능동적 빌드 시스템 ABaR에 대하여 제안하였다.

개발자는 하나의 개발 버전을 구현하고 ABaR 내부에 여러 타입에 맞춘 리소스를 저장한다. 어플리케이션은 최초 실행 과정에서 ABaR 서버에 접근하여 현재 설치된 디바이스 환경에 알맞은 리소스를 내려 받아 최적화된 구조로 구동된다.

또한 사용자 정보 분석 모듈을 통해 개인 및 소규모 개발자는 어플리케이션에 대한 분석된 설치 정보를 활용하여 서비스의 질을 높일 뿐만 아니라 어플리케이션의 수명주기, 디버깅, 개발 사이클 등에 효과적으로 대응할 수 있는 정보를 제공 받을 수 있게 되며 개발 비용을 효율적으로 절감할 수 있을 것으로 사료된다.

참고문헌

[1] 안드로이드 개발자 지원 공식 사이트, http://developer.android.com/

[2] 포커스온, http://focuson50.tistory.com/21

[3] 윤용호, 김진영, 이광근, “안드로이드 어플리케이션의 불필요한 코드를 찾는 방법”, 한국정보과학회 학술발표논문집, 제39권 1A호, pp.440-442,2012

[4] 홍성길, 김강희, “안드로이드 응용의 응답시간 향상을 위한 실행시간 선행 컴파일 체계”, 한국정보과학회논문지, 제20권 제1호 pp.6-10, 2014