

Android 端末における Thick/Thin クライアント制御機能の一検討

本橋 史帆[†] 小口 正人[†]

[†]お茶の水女子大学

1. はじめに

クライアント端末は、モバイル端末のように端末上にデータやアプリケーションを保持し、全ての処理を端末上で行う Thick クライアント端末と、データ管理やほとんどの処理を全面的にサーバ側に頼りクライアント側は処理結果を表示するだけという Thin クライアント端末の 2 種類に分類できる。

近年、無線通信やモバイル通信が高速化し Thin クライアントとして動作する端末が増加した。一方で、Android 端末などのスマートフォンが普及し、端末の高機能化が進んでいる。それによりモバイル端末の Thick クライアント化が進み、「持ち運べるコンピュータ」として注目されてきた。しかしスマートフォンのアーキテクチャは、リソースの制約や、プログラミングおよび実行環境が組込みシステム向けとなっていることなどから、汎用 PC とは大きく異なる。そのため、これまでの Thick/Thin クライアントモデルの議論もクライアント側は基本的に汎用 PC である事が多かった。そこで本研究では、汎用 PC とは環境が大きく異なる Android 端末において Thick/Thin クライアントモデルを議論し、環境に応じてどちらとしても動作させることができるような制御機能を提案する。

2. Android アプリケーション

Android のアーキテクチャを図 1 に示す [1]。Android と汎用 PC の大きく異なる部分は、独自に開発された Android Runtime に Java で記述された Android アプリケーションの実行環境である Dalvik VM(Dalvik 仮想マシン) を搭載している点である。

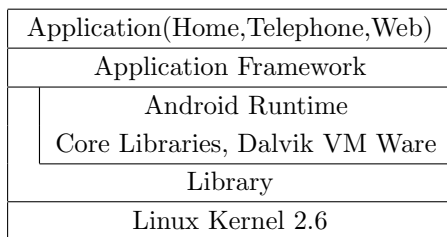


図 1: Android のアーキテクチャ

Android アプリケーションは Java で記述され、コンパイルにより Java クラスファイルができる。クラスファイルは Android 向けソフトウェアを開発するための開発環境である Android SDK に含まれる dx と呼ばれるツールで Dalvik Executable(DEX) と呼ばれる形式のファイルに変換され、Dalvik VM を介して実行される [2]。

3. Thick/Thin クライアント性能評価実験

3.1 実験目的

扱うデータ量が多く、CPU に負荷をかけるグラフィックス処理に焦点を当て、Thick/Thin クライアントの処理性能比較を行う。それにより Thick クライアント端末で十分な速度で実行できるアプリケーションと、Thick クライアント端末では実行速度が遅く、Thin クライアントモデルの形で処理を行った方が好ましい Thin クライアント向けのアプリケーションがあることを明らかにする。

3.2 実験環境

本実験で使用した実験環境を表 1 に示す。サーバ側に Android アプリケーション開発環境を整えた。本実験ではグラフィックス処理を扱うため、コンピュータビジョンに必要な機能を揃えた C/C++ ライブラリ集である OpenCV をインストールし、さらに Java から OpenCV ライブラリを呼び出せるよう開発された OpenCV for Android をインストールした。また、実機で OpenCV を利用したグラフィックス処理アプリケーションを実行するために必要となる OpenCV Manager をインストールした。

表 1: 実験環境

Android	Model number	Galaxy Nexus
	Android version	4.2.2
	Application	OpenCV Manager
server	OS	Ubuntu 13.04 (64bit)
	CPU	3.60GHz
	Main Memory	4GB
	Application Development Tool	JDK, Android SDK, Eclipse, ADT
	Library	OpenCV-2.2.0, OpenCV for Android-2.4.7

3.3 実験結果と考察

図 2 に実験結果を示す。図から明らかなように、Thin クライアントとしてサーバで処理をさせた場合には、Thick クライアントとして Android 端末に処理をさせたときの半分以下の処理時間で処理できていることがわかる。

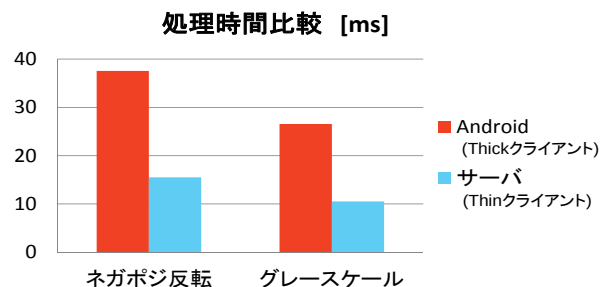


図 2: Thick/Thin クライアント処理時間比較

この結果から、本アプリケーションに関しては Thin クライアントモデルの方が処理性能が大幅に良いことを示すことができた。

本実験の考察として、1枚の静止画に対しこれだけの差が生じているため、1秒間に20～30枚の静止画を連続表示する動画を処理するプログラムは大きな処理時間差が表れ、本研究目的である切換え制御機能を提案することに意義があると視覚的に示すことができると考えられる。また、今後切換え制御機能を実装し、その性能評価を行う時にこのような Thin クライアント向けのアプリケーションを必要とするため、改めて処理対象を動画に変えたアプリケーションを実装し、本実験と同様の処理時間比較を行っていく。

4. 動画処理に関する性能評価実験

4.1 実験概要

AVI形式の動画(20秒, 6.43MB)を用意し、動画の各フレームを取り出し、フレーム毎に処理を加えるといった動画処理を行うプログラムを実装する。本実験では、OpenCV ライブラリを利用して各フレームにソーベルフィルタをかける処理を施した。

4.2 実験結果と考察

Thin クライアント側のプログラムを C 言語で実装した。10 回行った平均処理時間をとったところ、動画読込からフレーム取得・処理、画面表示にかかる時間は 10.36[s] で、そのうちフレーム取得・処理にかかる時間は 4.90[s] という結果を得た。これだけ大きな処理時間が出ていることから、Thick クライアント側の Android アプリケーションの実行速度はより遅く、大きな処理時間差が生じると考えられる。しかし、C 言語のプログラムと Java のプログラムの互換性がないことなどから、異なる環境である Thick クライアントにおいて同等の処理を行う Android アプリケーションを実装するのは難しいと言える。

そこで本研究では、Google 社が無償で公開している、Android 環境において端末のハードウェア上で実行できるネイティブコードのプログラムを開発するための開発環境である Android NDK(Native Development Kit) を用いることにした。これを利用することで、Java プログラムから JNI(Java Native Interface) を利用してハードウェア上で実行できるネイティブコードを呼び出すことができるため、Thin クライアント側で実装したプログラムの一部の再利用が可能となる [3]。

5. 提案制御機能の概要

ここで、本研究で提案する Thick/Thin クライアント制御機能の概要を説明する。

本提案機能のフレームワークを図 3 に示す。Android には向いていないと考えられる、実験に用いてきたグラフィックス処理や数値計算処理、データベース処理などといった処理を要するアプリケーションを起動する際に、本提案切換え制御機能を介して、ネットワーク環境や端末情報、そして処理の重さなどといった情報を取得し、その情報を元にそのまま Thick クライアントの形でアプリケーションを起動するか、または Thin クライアントの形で起動するためにサーバへ接続しサーバ側に処理をさせるかを決定する。

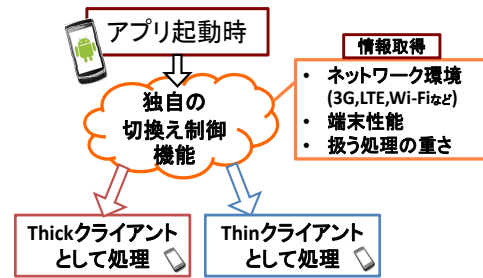


図 3: Thick/Thin クライアント制御機能概要

この切換え制御機能を図 4 のような切換えアプリケーションとして実装する。ユーザはまずこの切換えアプリケーションを起動する。このアプリケーションは、起動時にネットワーク環境・端末情報を取得し、ユーザに起動したいアプリケーションを選択させ、さらに選択されたアプリケーション情報を取得する。これらの得られた情報を元に Thick/Thin クライアントモデルの切換えを行うものとする。

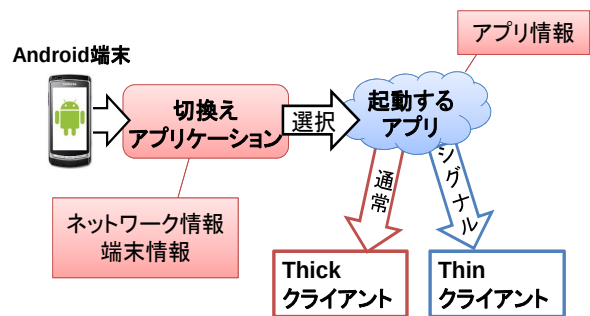


図 4: 切換えアプリケーション概要

6. まとめと今後の課題

本研究では、Thick クライアントである Android 端末を、ネットワーク環境や端末性能、起動するアプリケーションの処理の重さなどに応じて、Thin クライアントモデルとして動作させることで、両者のメリットを生かすことができると考え、Thick/Thin クライアント制御機能を提案した。簡単な画像処理を行うプログラムを実装し実験を行った結果、Thin クライアント端末としてサーバ側に処理をさせた方が、Thick クライアント端末として端末上で処理をさせるよりも実行速度が速く、処理性能が良いことがわかった。この結果から、本提案制御機能を実装することに十分な意義があることが示せた。

今後は、本提案機能の性能評価に利用するために Thick/Thin クライアントの処理性能差が大きく出る Android アプリケーションを実装する。それと並行して、切換え制御自体がボトルネックになる可能性を踏まえた切換えアプリケーションの実装も行っていきたい。

参考文献

- [1] Android: <http://www.techfirm.co.jp/lab/android/outline.html>
- [2] DalvikVM: <http://readwrite.jp/archives/1970>
- [3] NDK: <http://e-words.jp/w/Android20NDK.html>