DEIM Forum2014 E8-5

Android端末においてThick/Thinクライアントの切換え制御を行うソフトウェア実装の提案

本橋　史帆†　　小口　正人†

† お茶の水女子大学　〒 112-8610 東京都文京区大塚 2-1-1
E-mail: †g1020536@is.ocha.ac.jp, ††oguchi@computer.org

あらまし　近年，無線通信やモバイル通信の高速化が急速に進んだことにより，クライアントでは限定的な処理のみを行い，サーバ側の処理に全面的に頼る，Thin クライアントモデルとして機能する端末が増加した．一方，モバイル端末では，ガラパゴス携帯から Android をはじめとするスマートフォンへと流行が移り，近年スマートフォンの高機能化が進んでいる．スマートフォンは，クライアント側にデータを保持し計算処理も主にクライアント側で行う Thick クライアントモデルとして動作し,「持ち運べるコンピュータ」としての役割が期待されている．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．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 である事が多く，サーバ側に近いアーキテクチャのコンピュータをクライアント側でも用いていた．これに対し本研究では，汎用 PC と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，両者を切り換えて扱うことを可能にする制御ソフトウェアの実装を提案する．またこれを実現するための各要素の具体的な実装方法について議論を行い，実装した例を示す．

キーワード　Android, モバイル端末, Thick/Thin クライアント, 制御機能

A Proposal of Software Implementation for Switching Control of Thick/Thin Client in Android Terminal

Shiho MOTOHASHI† and Masato OGUCHI†

† Ochanomizu University　　2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610, JAPAN
E-mail: †g1020536@is.ocha.ac.jp, ††oguchi@computer.org

Key words　Android, Mobile phone, Thick/Thin client, Control Mechanism

1. はじめに

クライアント・サーバ型のネットワークコンピューティングにおいては，クライアント側にデータを保持し，計算処理も主にクライアント側で行う Thick クライアントモデルと，クライアントではコマンド入力や画面表示などの限定的な処理のみを行い，サーバ側の処理に全面的に頼る Thin クライアントモデルが提唱され，各時代のコンピュータやネットワーク環境においてそれぞれの実装が行われてきた．Thick クライアント端末はデータやアプリケーションを端末内に保持するため，オフラインでも操作が可能というメリットを持つが，データ管理・セキュリティ面には不安が残る．それに対し，Thin クライアント端末はサーバ側でデータを管理するため，セキュリティ面で注目を集めている．また，サーバ側のリッチなリソースを用いた高機能/高性能処理が可能というメリットも持つ．しかし Thin クライアント端末はサーバに接続するための一定レベルの通信帯域が必須となり，オフラインでは使用できないことがデメリットとなる．

研究背景として，近年，スマートフォンが爆発的に増加し，その高機能化が進んでいることが挙げられる．それに伴い，Thick クライアントとして動作するスマートフォンは「持ち運べるコンピュータ」としての役割が期待され，Thick クライアント化がより進んでいる．その一方で，無線通信やモバイル通信の高速化が進み，通信帯域が太く安定したため，セキュリティ面で注目を集めた Thin クライアント端末も急速に普及した．モバイル端末を Thin クライアント化してしまおうという動きも見られる．しかし，前述のように Thick/Thin クライアントモデルにはどちらにもメリット・デメリットが挙げられるため，どち

らが良い/悪いという優劣をつけることは難しいと言える．そこで本研究では，世界のスマートフォンシェア率の約 80 %を占める Android 端末 [1] を用いて，ネットワーク環境や起動するアプリケーションの処理の重さ，端末性能を考慮し，環境に応じて Thick クライアントモデルから Thin クライアントモデルへと切換えて処理を実行することができるような切換え制御機能を提案する．この制御機能を実現させることにより，Thick/Thin クライアント両者のメリットを生かした端末の使用ができるのではないかと考える．

これまで，Thick/Thin クライアントモデルの議論では，クライアント側は基本的に汎用 PC であることが多く，サーバ側に近いアーキテクチャのコンピュータをクライアント側で用いてきた．スマートフォンのアーキテクチャは I/O インタフェースの機能の乏しさ，限られたハードウェア資源，プログラミングおよび実行環境が組込みシステム向けになっているなどの点で，従来の汎用 PC のアーキテクチャとは大きく異なり複雑だからである．本研究では，このような理由であまり議論されてこなかった Android 端末をクライアント側に用いて Thick/Thin クライアントモデルの評価や切換えを行っていく．

2. 関 連 研 究

携帯性とサーバの利便性を両立する Thin クライアント端末としてモバイル端末を利用した Thin クライアントシステムの検討がなされてきた [2]．関連研究のほとんどはモバイル端末を Thin クライアント化させることに焦点を置き，そのための要件やセキュリティ・データ管理に関するものである．実際に近年ではモバイル端末を Thin クライアント化させ，端末の紛失・盗難対策やデータやアプリケーションの管理などセキュリティ機能を考慮したものも提供されている．それに対し，本研究ではモバイル端末を完全に Thin クライアント化するのではなく，Thick クライアントとしての機能を残しつつ，環境に応じて Thin クライアントとして動作させることを目標とする．

3. Android のアーキテクチャ

Android のアーキテクチャを図 1 に示す [3]．Android は Linux カーネルをベースとし，スマートフォンやタブレット端末をターゲットに，それらに適したコンポーネントが追加されている．Linux OS と大きく異なる部分は，独自に開発された，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 Android Runtime に搭載している点である．基本的にすべての Android アプリケーションは Java で記述され，この Dalvik VM を介して実行される．そのため，ネイティブコードで記述されるプログラムに比べ，Android アプリケーションは実行速度が遅くなる．特に，処理をするデータ量が多く，CPU に負荷をかけるようなグラフィクス処理や数値計算処理，データベース処理などに関しては，ネイティブコードで記述されたプログラムと Android アプリケーションとでは実行速度に大きな差が現れ，Android 端末での処理は不向きであると考えられる．本提案制御ソフトウェアはこれらの処理を Android 端末からの利用でも十分な速度で実行させることを可能にすると考える．

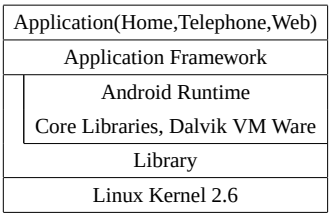

図 1　Android のアーキテクチャ

4. Thick/Thin クライアント性能評価

4.1 実 験 目 的

Android 端末において，Thick/Thin クライアントモデルを切換えて扱う制御機能を提案することに意義があることを示す．そのためにまず，両者の処理性能比較を行い，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示す．

4.1.1 Android アプリケーション

Android アプリケーションの開発環境は無償で提供されており，誰でも気軽にアプリケーションを開発できるようになっている．また，Android アプリケーションはキャリア間の制約がなく，自由度も汎用性も高い．そのため，様々な機能を持つ数多くのアプリケーションが市場に出回っており，ユーザは自由に欲しいアプリケーションを入手できる．しかし，それらはもちろん Android 端末上で処理することを前提としており，十分な速さで処理ができるものがほとんどであるため，Thick/Thin クライアントモデルの切換えは必要としないと考えられる．そこで本研究では，このアプリケーション開発環境を利用して，2 節で挙げたような実行条件によっては Android には不向きと考えられる処理のうちグラフィクス処理に焦点を当て，あえて実行速度が遅くなる可能性のある重い処理のアプリケーションを実装する．

4.2 実 験 概 要

PNG 形式 182KB の画像を 1 枚用意し，グレースケール化するプログラムとネガポジ反転をするプログラムを作成した．Thick クライアント側は Java で Android アプリケーションとして実装し，Android 端末実機で実行させる．一方，Thin クライアント側はネイティブコードで記述したプログラム (今回は C で書いたプログラムのバイナリ) をサーバ側で処理させ，処理結果をクライアント端末に表示させる．それぞれ画像処理にかかる時間を計測することで，このような単純な画像処理における Android 端末とサーバの処理性能の比較を行う．

4.3 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

Android アプリケーション用に 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

4.4　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と比較して半分以下の処理時間で処理できていることがわかる．

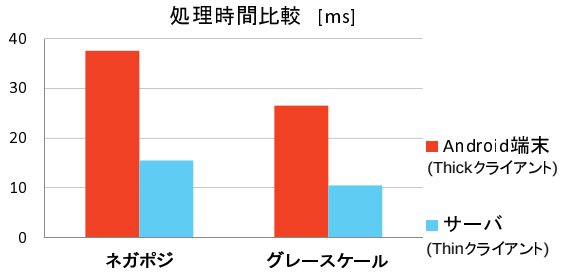

図 2　Thick/Thin クライアント処理時間比較

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 枚の画像に対してこれだけの差が表れていることから，より大きな画像，もしくは 1 秒間に 20～30 枚の静止画を連続表示する動画を処理対象にすることで処理時間差がより大きく表れ，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際にこのような 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，処理対象を動画に移したアプリケーションを実装し，本実験と同様の評価実験を行っていく．

5.　動画を処理対象にした性能評価

5.1　実 験 概 要

AVI 形式の動画 (20 秒，6.43MB) を用意し，動画のフレームを取り出し，画像処理を加えて，処理を施したフレームを表示させるという動画処理を行うプログラムを実装する．本実験では，フレームにソーベルフィルタをかける処理を施した．

5.2　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．サーバ側の処理でも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なり，大きな処理時間差が生まれると考えられる．しかし，両者の実行コード生成手順が異なることや，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アプリケーションを実装するのは容易ではなく，アプリケーションプログラマに大きな負荷がかかる．そこで本研究では Thick クライアント側のアプリケーションをより容易に実装できるようなフレームワークを用意する．

5.2.1　プログラム実行手順

Thick クライアント向けの Java で記述される Android アプリケーションと Thin クライアントのサーバ向けのネイティブコード (C,C++) で記述されたプログラムの実行コード生成までの開発手順をそれぞれ図 3，図 4 に示す [4] .

2 つの図を見てわかるように，ネイティブコードで記述されたプログラムは 1 度のコンパイルで実行コードが生成されるのに対し，Java で記述されるプログラムは Android アプリケーションパッケージが生成されるまでにいくつかの段階を経なくてはならず，その構成は複雑である．このような実行コード生成手順の相違から，異なる環境で同等の処理を行うプログラムを作成することは難しいとわかる．

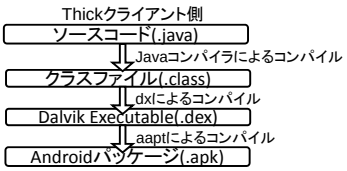

図 3　Android アプリ実装

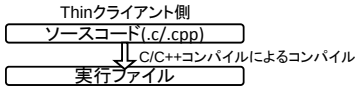

図 4　C/C++プログラム実装

5.2.2　Android NDK

5.2 節冒頭で，ネイティブコードと Java のプログラムに互換性がないことが問題であると述べたが，その解決策として Android NDK(Native Development Kit) を用いることにした．本節では，図 5 を用いて Android NDK の仕組みを説明する．

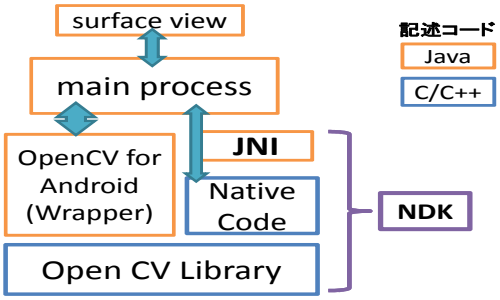

図 5　Android NDK の仕組み

Android NDK とは 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のことであり，Java プログラムからは JNI(Java Native Interface) を利用してネイティブコードを呼び出すことができる．ネイティブコードで記述された部分は Dalvik VM を介さずに実行されるため，アプ

リケーションの高速化につながり，また，ネイティブで記述されたコードの再利用が可能となる [5] ．NDK を用いることで，本家の OpenCV ライブラリを利用できるだけでなく，Thin クライアント側で実装したプログラムの一部を再利用できるようになる．

本研究ではこの開発環境を用いて，グラフィクス処理部分をネイティブコードで記述し JNI を介して呼び出す形で Thin クライアント側のプログラムと同等の処理を行う Android アプリケーションをより容易に実装できるようにする [6] ．

6. 切換え制御ソフトウェア

6.1 ソフトウェアの構成

ここで，本研究で提案する切換え制御ソフトウェアの概要を説明する．

Thick/Thin クライアント切換え制御ソフトウェアの動きを図 6 に示す．実験に用いてきたような Android に不向きな処理を要するアプリケーションを起動する際に，本提案ソフトウェアを介して，3G,LTE,Wi-Fi などといった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．

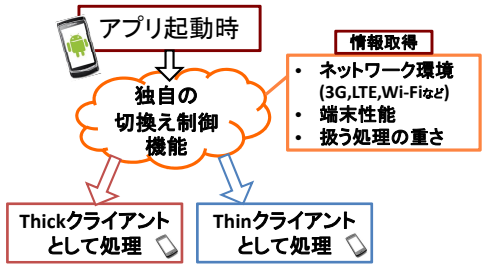

図 6　Thick/Thin クライアント切換え制御ソフトウェア

この切換え制御ソフトウェアを図 7 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．

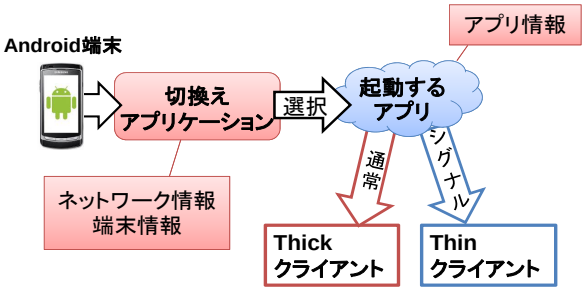

図 7　切換えアプリケーション概要

このアプリケーションは，起動時にネットワーク環境・端末情報を取得する．そしてユーザに起動したいアプリケーションを選択させ，実行時の予測が可能な場合は選択されたアプリケーション情報を取得する．今後の評価実験により定める切換え基準と得られたこれらの情報を元に Thick/Thin クライアントモデルの切換えを行い，環境に応じた形でアプリケーションを実行するものとする．次節ではこのフレームワーク実現のために用いる機能を挙げる．

6.2 ソフトウェア実装方法

6.2.1 情 報 取 得

本実験で実装した具体的な情報取得手法を説明する．ネットワーク情報としてネットワークの種類 (Wi-Fi，モバイル通信等) を取得し，さらに Wi-Fi の場合は，RSSI(Received Signal Strength Indication) と呼ばれる受信信号強度を取得する．RSSI の単位は dBm で，RSSI 値は 1mW のときの信号強度を 0dBm としたときの相対数値となっており，取得した値が 0 に近いほど電波強度が強いという指標となる．これらの情報から，Thin クライアントとしてサーバに接続するために十分なネットワーク状況下であるかどうかを判断する．

また，端末情報として，搭載メモリ容量，使用可能メモリ量，端末名などを取得し，選択したアプリケーションを実行するのに十分なメモリが確保できるか，端末に十分な処理能力が備わっているか等の判断基準とする．

以上に述べたネットワークや端末の情報取得機能の実装を Android 端末上で行った．選択したアプリケーションの情報については，実行時に情報取得が可能な場合には，メモリ使用量等を取得し，切換え判断基準に追加する．Thick/Thin クライアントモデル切換えの条件は評価実験を重ねて定めていくことにする．

6.2.2 Thick クライアントとして動かす場合

Android 特有のインテントという機能を用いる．インテントとはアプリケーションの中の 1 つ 1 つの機能を橋渡しする仕組みのことであり，各アプリケーションはインテントフィルタに自身が反応できるイベントの種類を格納している [7]．インテントが発行されると，OS は端末内の全アプリケーションのインテントフィルタを調べ，そのインテントを処理できるアプリケーションを呼び出す仕組みになっている．ここで呼び出されたアプリケーションを選択し起動することで，本切換えアプリケーションを介して Thick クライアントモデルの形でアプリケーションを実行させることが可能となった．

6.2.3 Thin クライアントとして動かす場合

サーバに命令を送るために socket 通信を利用する．具体的には，切換えアプリケーションで得た端末情報，ネットワーク情報等から Thin クライアントとして動作させることにした際に，起動したいアプリケーションを選択すると，そのアプリケーション名が socket 通信によりサーバに送られ，サーバ側は受け取ったアプリケーション名に対応するプログラムを実行するという仕組みである．同時に，Android 端末側ではデスクトップ共有アプリケーションを起動し，サーバ側の画面を表示することで処理結果を共有する．

socket 通信を利用して，サーバ・クライアント間でデータ送受信を行い，サーバ側で指定プログラムを実行するためのプログラムをサーバ側とクライアント側にそれぞれ C 言語で実装した．シンプルなプログラム実装を行い，その実行例を図 8 に示す．このプログラムは，サーバ側・クライアント側でそれぞれ

socket を生成/接続した状態でクライアントである Android 側からアプリケーション名をサーバ側へ送信し，サーバ側が受信したアプリケーション名と同名のプログラムを実行する．このクライアント側のプログラムを Java で実装しなおし，切換えアプリケーションに組み込むことで，Thin クライアントとして動作させることが可能となる．

図 8　socket 通信を利用したアプリケーション起動例

7.　まとめと今後の課題

本研究では，Thick クライアントモデルとして動作する 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 クライアント切換え制御ソフトウェアを提案した．また，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で処理を行った方が Thick クライアント端末として処理を行うよりも処理性能が良く，実行速度も速いことがわかった．この結果から，本提案ソフトウェアを実装することに十分な意義があることが示せた．

また，本提案ソフトウェアフレームワークを提案し，そのうちの情報取得部分，Thick クライアントとして動作させる部分，Thin クライアントとして動作させる部分をそれぞれ実装した．

今後の課題としては，個々の実装を統一し，1 つの切換えアプリケーションとして完成させる．また，Android NDK を利用して Thick/Thin クライアントモデルの処理性能差が大きく出るような様々な Android アプリケーションを実装し，本提案ソフトウェアの評価実験を行っていく．この評価実験を通して，Thick/Thin クライアント切換え判断基準を調整する．それと同時に切換え制御自体がボトルネックになってしまう可能性も合わせて調査し，より完成度を高めていく．

文　　献

[1] Android シェア率:http://jbpress.ismedia.jp/articles/-/38424
[2] 高橋竜男，高橋修，水野忠則:モバイル向けシンクライアントシステムの検討，情報処理学会論文誌，Vol.45 No.5 pp.1417-1431(May 2004)
[3] Android アーキテクチャ:http://www.techfirm.co.jp/lab/android/outline.html
[4] Android application:http://www.atmarkit.co.jp/ait/articles/0812/08/news123.html
[5] Android NDK:http://e-words.jp/w/Android20NDK.html
[6] NDK 実装:http://www.kkaneko.com/rinkou/js/andk.html
[7] インテント:http://techbooster.org/android/application/8346/