3A-6

Ubuntu 上で動作する Android のリソース制限方式の提案

勅使河原　佑美　　出原　章雄　　山本　整　　落合　真一
三菱電機株式会社　情報技術総合研究所

1. はじめに

弊社では、マルチコア CPU 対応の組込み Linux 搭載機器へ新しい機能を追加するために、3rd Party アプリの利用を検討している。

従来のシステムでは、自製アプリ(主機能)のみが動作していた。しかし、新しい構成では、主機能と 3rd Party アプリが動作し、3rd Party アプリによりリソースが専有される可能性があり、主機能がリアルタイム性を維持できないといった問題がある。そのため、3rd Party アプリが使用するリソースを制限する方式を検討している。

本稿は、Linux 上で動作する 3rd Party アプリが使用するリソースの制限方式の提案である。今回の評価では Linux ディストリビューションの 1 つである Ubuntu を用いて実行環境を構築する。リソースの制限には、Linux に搭載されているリソース管理機能である cgroups[1]を利用する。

2. 目的

主機能のリアルタイム性確保のために、cgroups 管理下プロセスのリアルタイム性を評価する必要がある。その前段階として、まずは Ubuntu 上で cgroups が持つリソース管理機能の性能評価を行った[2]。cgroups は、CPU、メモリ、ディスク、ネットワークのリソース使用量・割合を特定のプロセスグループに割当てることができる。

本稿では、Ubuntu 上で 3rd Party アプリケーションとして Android を動作させ、cgroups により、リアルタイム優先度が設定された主機能の CPU 時間を確保することができるか評価する手段について述べる。

3. 実行環境

3.1. 構成

評価機器として、組込み向けマルチコア CPU の TI 社製 OMAP4430 を搭載した PandaBoard を用いた。使用するハードウェアは表 1のとおり。

A proposal of resource management of Android on Linux
Yumi TESHIGAWARA, Akio IDEHARA, Hitoshi YAMAMOTO, Shinichi OCHIAI
Information Technology R&D Center, Mitsubishi Electric Corporation

ソフトウェア構成は表 2のとおり。

表 1：使用するハードウェア

項目	詳細
Pandaboard	CPU : Dual-core ARM Cortex-A9 1GHz
SD カード	Pandaboard 上で OS を動作させるために使用。8GB

表 2：使用するソフトウェア

項目	詳細
Ubuntu	Ubuntu10.10 を使用。カーネルが cgroups に対応しているもの。
Android	Android2.2 を使用。カーネルが cgroups に対応しているもの。

3.2. Ubuntu 上での Android の実行方式

① OS のインストール

SD カードに Ubuntu をインストールする(図 1左図)。次に、カーネルを Android 用のものに変更し、Ubuntu のユーザランドに Andorid のユーザランドのファイルを追加する(図 1右図)。

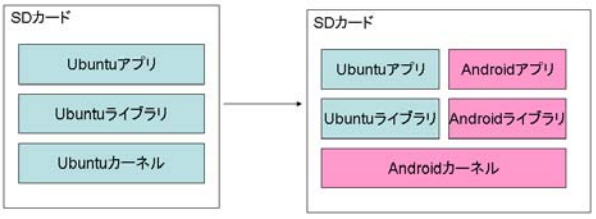

図 1：ソフトウェア構成

②設定

OS を起動するときのハードウェア関係図は図2のとおり。ディスプレイ、USB キーボード、USB マウスは Android の管理下におく。

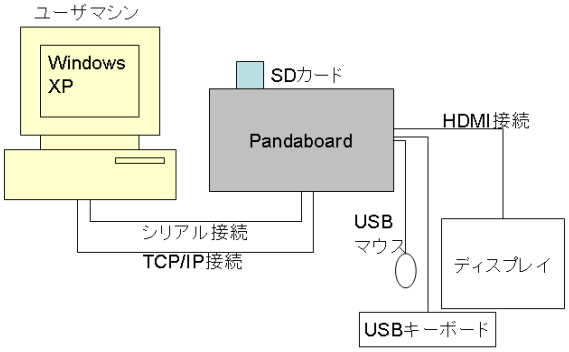

図 2：ハードウェア関係図

Ubuntu は、ユーザマシンからシリアルコンソールでログインできるように設定し、Android 管理下におくディスプレイにはグラフィック表示しないようにする。また、Android と Linux のデバイス管理機構には互換性がなく、Linux でデバイス管理機構を実行すると Android のネットワークデーモンなどで障害が発生するため、今回は回避策として、Linux のデバイス管理機構を停止する。

③Android の起動

Android は、まず Ubuntu を起動し、その上で、ルートディレクトリを Andorid のユーザランドを追加したディレクトリに変更することで、起動できる。

3.3. リソース制限の実現方式

評価は図 3の構成で行う。cgroups はプロセスのように階層的に構成されており、子は親の属性を受け継ぐ。親のグループ「cg」の下に子のグループ「cg0」「cg1」「cgA」を作成する。cg0 と cg1 にて主機能、cgA にて Android のリソース管理を行うことを想定している。

cgroups では使用するコアを割り当てることができ、マルチコア構成上で、cg0 をコア 0 に、cg1 と cgA をコア 1 に割り当てる。cgA とは異なるコア 0 上にある cg0 と、同じコア 1 上にある cg1 がそれぞれ、cgA 上で動作する Android の影響をどのように受けるか確認する。

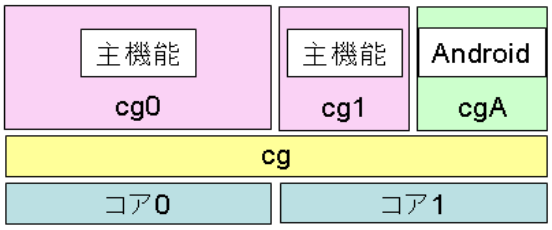

図 3：cgroups の構成

3.4. 評価内容

この構成で、リアルタイム優先度が設定されたプロセスが動作する時間を確保できるか評価する。cgroups のパラメータの中でリアルタイム優先度に関する以下の 2 つのパラメータ値を設定することで、cg0 上と cg1 上の主機能の CPU 時間を確保することができるか確認する。

- cpu.rt_period_us
  各プロセスグループに対して設定する CPU リソース制御周期時間(μ秒)
- cpu.rt_runtime_us
  各プロセスグループに設定された周期時間における CPU 利用可能時間(μ秒)

種々のパターンの設定値で、以下の手順で評価を行う。

1. cgA 上で、リアルタイム優先度を設定した無限ループを行うプログラムを実行し、負荷をかける。
2. cgA と別のコア上にある cg0 が、cgA の影響を受けるか確認するため、1 の負荷をかけた状態で、cg0 上でリアルタイム優先度を設定した無限ループを行うプログラムを実行し、CPU 使用率を測定する。また、1 の負荷をかけた状態で、cg0 上で CPU コアのベンチマークを実行し、ベンチマーク値を計測する。
3. cgA と同じのコア上にある cg1 が、cgA の影響を受けるか確認するため、1 の負荷をかけた状態で、cg1 上でリアルタイム優先度を設定した無限ループを行うプログラムを実行し、CPU 使用率を測定する。また、1 の負荷をかけた状態で、cg1 上で CPU コアのベンチマークを実行し、ベンチマーク値を計測する。

CPU 使用率が設定したパラメータ値のとおりになれば、cg0 と cg1 の CPU 時間を確保できたといえる。また、計測した CPU 使用率を、無負荷の状態で計測したベンチマーク値に対する、2、3 で計測したベンチマーク値の比率とも比較する。

4. まとめ

本稿では、Ubuntu 上で Android を動作させ、cgroups により、リアルタイム優先度が設定された主機能の CPU 時間を確保することができるか評価する手段について述べた。今後、実際に評価していく予定である。

また、主機能のメモリの確保や、主機能の使用するデバイスが Android に使用されないようにするための保護など、cgroups 管理下におけるプロセスのリアルタイム性を詳細評価していく予定である。

参考文献

[1] Paul Menage: “Linux Kernel Documentation/cgroups/cgroups.txt” http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt(2011).

[2]勅使河原，茂田井，山本，落合，松本：マルチコア対応組込み Linux におけるリソース制限方式の検討，情報処理学会第 74 回全国大会(2012)